IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 21 June 2023, accepted 8 July 2023, date of publication 13 July 2023, date of current version 20 July 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3295117

TOPICAL REVIEW

A Systematic Literature Review of Skyline Query Processing Over Data Stream

MUDATHIR AHMED MOHAMUD[1], HAMIDAH IBRAHIM[1], (Member, IEEE), FATIMAH SIDI[1], (Member, IEEE), SITI NURULAIN MOHD RUM[1], ZARINA BINTI DZOLKHIFLI[2], ZHANG XIAOWEI[1], AND MA'ARUF MOHAMMED LAWAL[3]

[1]Department of Computer Science, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Serdang, Selangor 43400, Malaysia
[2]Department of Computer and Networking, Faculty of Computing, Universiti Malaysia Pahang, Kuantan, Pahang 26300, Malaysia
[3]Department of Computer Science, Faculty of Physical Sciences, Ahmadu Bello University, Zaria, Kaduna State 810107, Nigeria

Corresponding author: Hamidah Ibrahim (hamidah.ibrahim@upm.edu.my)

This work was supported in part by the Ministry of Higher Education Malaysia through the Fundamental Research Grant Scheme under Grant FRGS/1/2020/ICT03/UPM/01/1, and in part by the Universiti Putra Malaysia.

ABSTRACT Recently, skyline query processing over data stream has gained a lot of attention especially from the database community owing to its own unique challenges. Skyline queries aims at pruning a search space of a potential large multi-dimensional set of objects by keeping only those objects that are not worse than any other. Although an abundance of skyline query processing techniques have been proposed, there is a lack of a Systematic Literature Review (SLR) on current research works pertinent to skyline query processing over data stream. In regard to this, this paper provides a comparative study on the state-of-the-art approaches over the period between 2000 and 2022 with the main aim to help readers understand the key issues which are essential to consider in relation to processing skyline queries over streaming data. Seven digital databases were reviewed in accordance with the *Preferred Reporting Items for Systematic Reviews* (*PRISMA*) procedures. After applying both the inclusion and exclusion criteria, 23 primary papers were further examined. The results show that the identified skyline approaches are driven by the need to expedite the skyline query processing mainly due to the fact that data streams are time varying (time sensitive), continuous, real time, volatile, and unrepeatable. Although, these skyline approaches are tailored made for data stream with a common aim, their solutions vary to suit with the various aspects being considered, which include the *type of skyline query*, *type of streaming data*, *type of sliding window, query processing technique, indexing technique* as well as the *data stream environment* employed. In this paper, a comprehensive taxonomy is developed along with the key aspects of each reported approach, while several open issues and challenges related to the topic being reviewed are highlighted as recommendation for future research direction.

INDEX TERMS Skyline query processing, data stream, certain data stream, uncertain data stream.

I. INTRODUCTION

In recent past, skyline queries have gained a lot of interest from the database community. Numerous applications for multi-criteria decision-making relied heavily on skyline computation. The introduction of the skyline operator by [1] has triggered an abundance of skyline algorithms being proposed [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50], [51], [52], [53], [54], [55], [56], [57], [58], [59], [60], [61]. These skyline algorithms were developed with the aim to optimize the skyline computation. However, they differ in various aspects, namely: (i) the characteristics of data they handled such as uncertain data, incomplete data, encrypted data, streaming data, big data, etc.; (ii) the platform they considered such as distributed,

The associate editor coordinating the review of this manuscript and approving it for publication was Vlad Diaconita.

cloud computing, road networks, etc.; and (iii) the type of skyline queries they processed such as range skyline, spatial skyline query, reverse skyline query, etc. The skyline operator introduced by [1] filters the collection of objects in a data set by selecting those objects that are not dominated by any other objects. An object dominates another object if it is as good as the other object in all dimensions and better in at least one dimension. This approach is commonly used to identify the best, most preferred objects known as skylines, from a given data set in satisfying a user's preferences that are specified as the *skyline query*. A classic example is identifying an apartment to rent that would meet the following user's preferences (i) apartment with the cheapest rent rate and (ii) apartment that is nearest to the city center. This is demonstrated in Figure 1(a) where the dimensions d_1 and d_2 represent the rent rate and distance, respectively. Applying the dominance analysis over the given twelve objects results in the objects o_5, o_6, o_{10}, and o_{12} as the skyline objects.

Data stream is crucial in decision making applications. These applications come in a variety of forms including sensor networks, manufacturing, web applications, telecommunications data management, security, network monitoring, and others. Unlike the conventional applications that generate data that are stored in finite persistent relations, data streams are time varying (time sensitive), continuous, real time, volatile, and unrepeatable [62]. Hence, every object o_i has a timestamp indicating the arrival time, $t_{arr}(o_i)$, and expiry time, $t_{exp}(o_i)$, of the object in the stream. Processing data stream is challenging due to several reasons: (i) the objects in the stream arrive online, (ii) the system has no control over the order in which objects arrive to be processed, either within a data stream or across data streams, (iii) data streams are potentially unbounded in size, and (iv) once an object from a data stream has been processed it is discarded or archived, it cannot be retrieved easily unless it is explicitly stored in memory, which typically is small relative to the size of the data streams [63].

The paradigm shift from static data to streaming data has recently attracted the attention of the database community simply because the traditional database techniques are inefficient at addressing the unique characteristics of data streams, such as rapid data arrivals, strict response time constraints, etc. Skyline query processing is no exception where skyline objects need to be continuously updated with objects arriving and expiring while time passes. For example, a user may ask the most preferable apartment deals advertised and want the results to be updated every 30 minutes. Using similar example as given in Figure 1(a), figures 1(b) and 1(c) depict the skyline objects at time, $t = 14$ and $t = 17$, respectively. Meanwhile, Figure 1(d) presents the arrival and expiry time of each object that are denoted by red dots and green dots, respectively. At time $t = 14$, nine objects are analyzed, i.e. $o_1, o_2, \ldots, o_9$, in which objects o_1, o_3, o_4, o_5, and o_6 are the skyline objects. Meanwhile, at time $t = 17$, the objects o_1, o_2, and o_3 are considered outdated while o_{10}, o_{11}, and o_{12} are the new arrival objects. Analyzing the valid objects at time $t = 17$ results in o_5, o_6, o_{10}, and o_{12} as the skyline objects. Obviously, a skyline query over data stream implies a continuous query which requires continuous evaluation as the query results (skylines) vary with the change of time.

A. RELATED REVIEWS

Several review studies have addressed the skyline computation issues faced during the processing of skyline queries. A recent survey conducted in [64], discusses on various flexible/restricted skyline solutions that are proposed to overcome the traditional ranking queries' inadequacies. These solutions include *Restricted Skyline queries (R-skylines), trade-off skyline, applications of ρ-dominance, Top-k dominating queries, Uncertain Top-k queries (UTK), Skyline ordering, and regret minimization.*

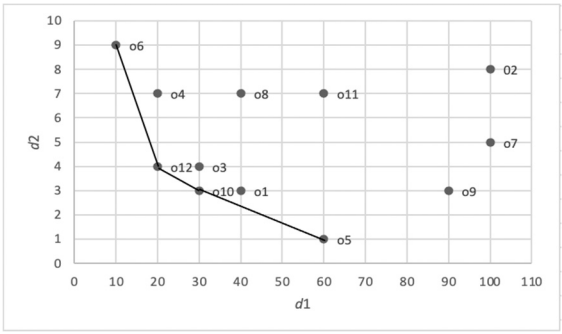
In [65], three skyline approaches are analyzed, and their main properties are highlighted. These approaches are *F-skyline*, *regret minimization*, and *skyline ranking* and they are introduced with the aim at overcoming the limitations of classic methods such as top-k and skyline queries. The following criteria are used in discussing the pros and cons of each technique: flexible input, mix top-k & skyline approaches, weight-based, multiple scoring functions, user interaction, scale invariant, stable, attribute order compensation, k selection, rank results, all-d inclusive, uncontrollable output size, and partial imprecision of output. The results of the comparison can be used as a guidelines in selecting the suitable approach in dealing with users' problems.

Dzolkhifli et al. [66] in their review has proposed a taxonomy of skyline query processing techniques over data stream. In reviewing the two main types of skyline queries over data stream, i.e. *single query* and *multi queries*; the following approaches are identified: *basic skyline*, *top-k skyline*, *n-of-N skyline*, *probabilistic skyline*, *dynamic skyline*, *group skyline*, and *reverse skyline*. The types of data stream which are categorized into two, namely: *uncertain data stream* and *certain data stream*; also play an important role in deciding the suitable approach in skyline computation. Moreover, to process the skyline queries over data stream where objects are frequently updated in rapid time, a *sliding window* approach is utilized. Two main well-known sliding window approaches that are *count based* and *time-based* are part of their developed taxonomy.

In [67], a comparative study has been conducted on skyline query processing on big data. The study has compared various skyline algorithms which include *basic distributed skyline* (*BDS*), *improved distributed skyline* (*IDS*), *progressive distributed skylining* (*PDS*), *mobile ad-hoc network* (*MANET*), *SKYPEER/SKYPEER+*, *parallel distributed skyline* (*PaDSkyline*), *feedback based distributed skyline* (*FDS*), *distributed skyline* (*DSL*), *skyline space partitioning* (*SSP*), *SKYFRAME*, and *iSKY*.

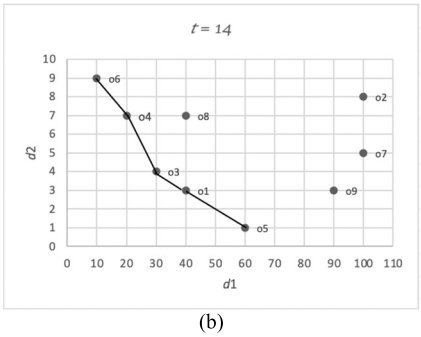
Gothwal et al. [68] presented a review on skyline algorithms that deal with varieties of data generated by different data-specific applications that include uncertain data,

Id	d_1	d_2
o_1	40	3
o_2	100	8
o_3	30	4
o_4	20	7
o_5	60	1
o_6	10	9
o_7	100	5
o_8	40	7
o_9	90	3
o_{10}	30	3
o_{11}	60	7
o_{12}	20	4

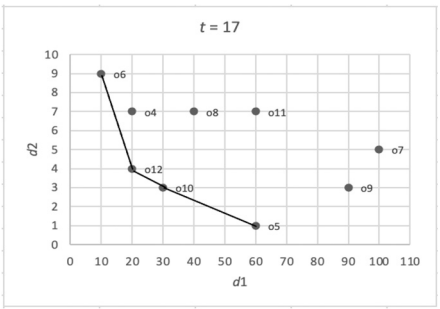


(a)

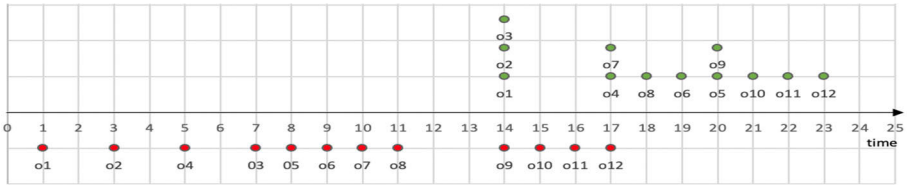
Id	d_1	d_2	$t_{arr}(o_i)$	$t_{exp}(o_i)$
o_1	40	3	1	14
o_2	100	8	3	14
o_3	30	4	7	14
o_4	20	7	5	17
o_5	60	1	8	20
o_6	10	9	9	19
o_7	100	5	10	17
o_8	40	7	11	18
o_9	90	3	14	20
o_{10}	30	3	15	21
o_{11}	60	7	16	22
o_{12}	20	4	17	23



(b)



(c)



(d)

FIGURE 1. (a) Example of skylines for non-streaming data, (b) example of skylines for streaming data at time, $t = 14$, (c) example of skylines for streaming data at time, $t = 17$, (d) the arrival and expiry time of each object.

incomplete data, time series data, and interval uncertain data. The following fundamental skyline algorithms are reviewed: *Block-Nested Loop* (*BNL*), *Divide and Conquer* (*D&C*), *Bitmap*, *Nearest Neighbor Search* (*NNS*), and *Branch and Bound* (*BBS*). Meanwhile, the following skyline algorithms *DSUD*, *e-DSUD*, *IDSUD*, and *ADSUD* which focus on processing skyline queries over uncertain data in highly distributed environment are analyzed. To compute the skyline queries for uncertain data on MapReduce framework considering both discrete and continuous models; three algorithms are discussed that are *PS-QPF-MR*, *PS-BR-MR*, and *PS-BRF-MR*. The *ISkyline* algorithm is the only skyline algorithm discussed in [68] that handled the issue of incompleteness of data. In addition, the *α/β-Dominant Skyline*, *Naïve method* (*NA*), *Bounding method* (*BM*), and *Online Skyline Query Answering Algorithm* (*OKQA*) that are devised to perform skyline computation on data arranged in series of time are also reviewed. The skyline algorithms to process skyline queries on uncertain data where the dimension of each object is represented as an interval or an accurate value are also analyzed. These include: *Branch and-Bound Interval Skyline* (*BBIS*), *CIS*, and *Constrained NN* (*CNN*).

A comprehensive survey on the state-of-the-art techniques for skyline query processing has been conducted in [69]. The survey emphasized on the numerous variations of the skyline algorithms that are proposed since the introduction of the skyline operator by [1] till the year 2013. The fundamental skyline algorithms that include *D&C*, *Bitmap*, *INDEX*, *NN*, *BBS*, *Sort-Filter-Skyline* (*SFS*), *Linear Elimination Sort for Skyline* (*LESS*), and *Sort and Limit Skyline algorithm* (*SaLSa*) are discussed. The different types of skyline queries and their applications are also presented. These include *constrained skyline queries*, *dynamic skyline queries*, *reverse skyline queries*, *group by and join skyline queries*, *top-k skyline query*, *thick skyline query*, etc.

A taxonomy of skyline algorithms over uncertain database is presented in [70]. The taxonomy divides the skyline algorithms over uncertain data into two main groups, namely: *index-based* and *non-index* approach. The *non-index-based* algorithms do not require any mechanism to organize the objects in the collections; while the *index-based* algorithms rely on an indexing technique such as *B-Tree* to expediate the process of filtering the objects in the collection. The *index-based* algorithms are further grouped into two

depending on the nature of data that they handled, i.e. *certain data* or *uncertain data*. Under the uncertainty of data, further grouping is performed reflecting two different uncertainty models assumed by the algorithms. These models are *dimension level* and *object level*. Based on the developed taxonomy, the following skyline algorithms are analyzed: *NN*, *BBS*, *Bitmap*, *probabilistic skyline* (*p-skyline*), *ε-approximate*, *U-Skyline*, *D&C*, *Effective Probabilistic Skyline Update* (*EPSU*), *LookOut*, *ISkyline*, and *Skyline Query over Uncertain Data* (*SkyQUD*).

A brief survey on skyline query processing algorithms and their different applications in various fields has been conducted in [71]. Their survey focused on *static skyline query*, *dynamic skyline query*, *spatial skyline query*, and *network skyline query*. The static skyline algorithms covered in their survey include *D&C*, *BNL*, *BBS*, *NN*, and *FAST-SKY*. Meanwhile, *I-SKY*, *N-SKY*, and *KMS* are among the dynamic skyline algorithms discussed in [71]. Furthermore, *Threshold Farthest Spatial Skyline* (*TFSS*), *Branch and Bound Farthest Spatial Skyline* (*BBFS*), *Branch and Bound Spatial Skyline* (B^2S^2), *Voronoi based Spatial Skyline* (VS^2), and *Voronoi based Continuous Spatial Skyline* (VCS^2) are the spatial skyline algorithms reviewed where spatial locations are part of the query requirements. Nonetheless, several skyline algorithms meant for network environment are also discussed which include *efficient distributed skyline based on mobile computing* (*EDS-MC*) and *Network nearest neighbor skyline* (*N3S*).

The survey in [72] reviewed the state-of-the-art of distributed skyline approaches for highly distributed environment. The analysis led to the development of a taxonomy of skyline query processing where three important factors that influence the performance of a distributed skyline approach are considered. These factors are *query routing*, *result propagation*, and *filter points*. Based on these factors, several distributed skyline approaches are compared, which include *Distributed Skyline* (*DSL*), *Skyline Space Partitioning* (*SSP*)*/Skyframe*, *iSKY*, *Semantic Small World* (*SSW*), *Single Filtering Point* (*SFP*), *Distributed Data Summaries* (*DDS*), *SKYPEER*, *SKYPEER+*, *BITPEER*, *Parallel Distributed Skyline* (*PaDSkyline*), *a grid based approach for distributed skyline* (*AGiDS*), *Feedback based Distributed Skyline* (*FDS*), and *SkyPlan*.

In [73], a survey on various skyline computation methodologies from centralized environment to modern computational environments that include distributed, real-time, mobile, and web is presented. It covers the fundamental skyline algorithms, namely: *BNL*, *D&C*, *SFS*, *LESS*, *SaLSa*, *Bitmap*, *NN*, and *BBS*. Meanwhile, the skyline algorithms proposed for processing a distributed skyline query, namely: *SkyPlan*, *PadDSkyline*, *BSkyTree-S*, *BSkyTree-P*, *Parallel Skyline* (*PSkyline*), and *Search Space Partitioning* (*SSP*) are briefly discussed. For real-time environment where the data set may be uncertain, three skyline algorithms are analyzed. They are *PSkyline*, *LookOut*, and *Constrained Skyline Computing* (*CSC*). In addition, *Ring-Skyline* (*RS*), *I-SKY*, *N-SKY*, *Range to Range Skyline Query* (*R2R*), *Point to Range Skyline Query* (*P2R*) are the skyline algorithms listed under the mobile environment. Several skyline algorithms proposed for the web environment are analyzed which include *Collaborative Filtering Skyline* (*CFS*) and *SkyRank*.

Table 1 summarizes the related review studies on skyline computation as reported in this section. For each study, the table highlights its main contribution, the year of referenced articles used in their analysis, whether *SLR* is used as the methodology in conducting the review, the nature of data under review, and the outcome of the study. From the table, it is obvious that most of the studies focus on certain and/or uncertain data. Only a few studies have constructed taxonomy [66], [70], [72] while none of the studies have applied the *SLR* methodology in conducting their review. Also, most of the review is within a short period of time (9 years – 15 years) except for [65] and [68]. Most significantly, since the majority of the articles they reviewed are older than five years [66], [67], [68], [69], [70], [71], [72], [73], it is particularly crucial to update the information presented in these studies to include the most recent works related to the study.

B. MOTIVATIONS AND CONTRIBUTIONS

Despite the development of the topic under study, to the best of our knowledge, none of the previous studies [64], [65], [66], [67], [68], [69], [70], [71], [72], [73] used the SLR methodology to systematically and comprehensively identify, classify, and compare the various skyline techniques over data stream. Thus, by providing a thorough and organized analysis of the scholarly literature between 2000 and 2022, this study seeks to help interested researchers to obtain an up-to-date review of works related to skyline query processing over data stream. The main contributions of this work are as follows:

- Proposing a comprehensive taxonomy of skyline query processing over data stream. The taxonomy is built based on the main key aspects that are identified through the literature research, namely: *type of skyline query, type of streaming data, type of sliding window, query processing technique, indexing technique* as well as the *data stream environment* employed.
- Providing a comparative study on the state-of-the-art approaches in processing skyline queries over data stream. We have developed seven key research questions that serve as the foundation for the comparative analysis of the most cutting-edge approaches.
- Highlighting open issues and challenges related to skyline computation over data stream as recommendation for future research direction. Based on the results of the *SLR*, we have identified three significant issues that warrant further investigation. They are (i) flexibility in query requirements of the users, (ii) adaptive system for extremely fast data streams, and (iii) dirty data sets with data quality problems.

TABLE 1. A summary of related review studies on skyline computation.

Reference/ year	Main contribution	Year of referenced articles	Systematic review	Nature of data	Taxonomy and comparison
[64]/2022	Survey on flexible/restricted skyline solutions	2008 – 2021 (13 years)	×	certain and uncertain data	comparison
[65]/2022	Comparative study on *F-skyline*, *regret minimization*, and *skyline ranking*	1998 – 2021 (23 years)	×	certain data	comparison
[66]/2021	Taxonomy of skyline query processing techniques over data stream	2005 – 2018 (13 years)	×	certain and uncertain data	taxonomy and comparison
[67]/2020	Comparative study on skyline query processing on big data	2008 – 2017 (9 years)	×	certain data	comparison
[68]/2018	Review on skyline algorithms that deal with varieties of data generated by different data-specific applications	2001 – 2017 (16 years)	×	uncertain data, incomplete data, time series data, and interval uncertain data	comparison
[69]/2017	Survey on the state-of-the-art techniques for skyline query	2001 – 2013 (12 years)	×	partially ordered data, incomplete data, uncertain data, data streams, time series	comparison
[70]/2017	Taxonomy of skyline algorithms over uncertain database	2001 – 2016 (15 years)	×	uncertain data	taxonomy and comparison
[71]/2014	Survey on *static skyline query*, *dynamic skyline query*, *spatial skyline query*, and *network skyline query*	2001-2013 (12 years)	×	uncertain data	comparison
[72]/2012	Taxonomy of skyline query processing for highly distributed environment	2001 – 2011 (10 years)	×	certain and uncertain data	taxonomy and comparison
[73]/2012	Survey on various skyline computation methodologies from centralized environment to modern computational environments	2000 – 2012 (12 years)	×	certain and uncertain data	Comparison
Our work	Comparative study on the state-of-the-art approaches in relation to skyline query processing over data stream	2000 – 2022 (22 years)	√	certain and uncertain data	taxonomy and comparison

C. ORGANIZATION OF THE PAPER

The rest of the paper is structured as follows. In Section II, the necessary definitions and notations, which are used throughout the paper, are set out. Section III defines the review steps of the *SLR* methodology which are based on the *Preferred Reporting Items for Systematic Reviews (PRISMA) 2020 guidelines* [74], while Section IV presents the results of classifying, analyzing, and synthesizing the selected articles based on the seven review questions that we have designed. The study results and open issues are discussed in Section V. Finally, we conclude our study in Section VI.

II. BACKGROUND

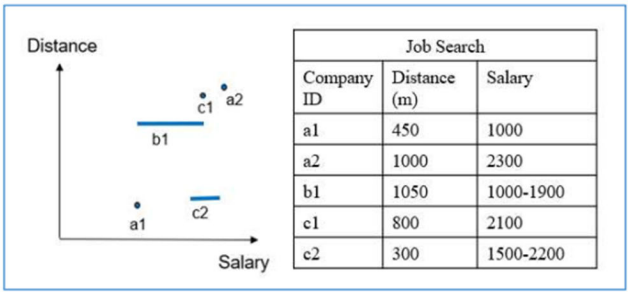
In this section, we present the necessary definitions and introduce the notations that are used throughout this paper. These definitions have been defined either formally or informally in the literature [1], [14], [22], [25], [80], [81]. Examples are provided where necessary to further clarify the definitions.

Definition 1 (Dominance): Given a database, D, with m dimensions, $d = \{d_1, d_2, \ldots, d_m\}$ and n objects $D = \{o_1, o_2, \ldots, o_n\}$, o_i is said to dominate o_j denoted by $o_i \prec o_j$ if and only if the following condition holds: $\forall d_k \in d, o_i.d_k \leq o_j.d_k \wedge \exists d_l \in d, o_i.d_l < o_j.d_l$. Here, we assume that lower values are preferred over higher ones. If otherwise, then the comparison operators $\geq$ and $>$ are used in the definition instead of $\leq$ and $<$, respectively. For instance, given the objects $o_1(6, 3, 2, 10)$ and $o_2(12, 3, 4, 10)$, then $o_1 \prec o_2$ as o_1 is better than o_2 in the first and third dimensions (true on the second part of the condition), while o_1 is equal to o_2 in the second and fourth dimensions (true on the first part of the condition).

Definition 2 (Skyline): Given a database D with n objects $D = \{o_1, o_2, \ldots, o_n\}$, o_i is a skyline of D if there is no other objects $o_j \in D$ that dominates o_i. Hence, the set of skylines of a database D can be denoted by $S = \{s_1, s_2, \ldots, s_l\}$ where each $s_i \in D$. Skylines hold the transitivity property that means if s_i dominates s_j and s_j dominates s_k, this implies that s_i dominates s_k [1].

Definition 3 (Discrete Uncertainty Model): Given a database D with n objects $D = \{o_1, o_2, \ldots, o_n\}$, an o_i is modelled as a probability distribution that is defined on a finite set of l instances (states), $o_{i1}, o_{i2}, \ldots, o_{il}$ with each o_{ij} is associated with a probability value. Here, the objects in D are said to be *uncertain* (*uncertain objects*) while D is called *uncertain database*.

Definition 4 (Continuous Uncertainty Model): Given a database D with n objects $D = o_1, o_2, \ldots, o_n$, an o_i is modelled as a continuous range of value or approximate value, v_i, which is associated with a probability density function representing the possible values of the object. Here, the value v_i is said to be *uncertain* (*uncertain value/data*) while D is called *uncertain database*.



Job Search		
Company ID	Distance (m)	Salary
a1	450	1000
a2	1000	2300
b1	1050	1000-1900
c1	800	2100
c2	300	1500-2200

FIGURE 2. Illustration of objects with certain and uncertain data.

Definition 5 (Continuous Range Value): A value $v_i = [lb{:}ub]$ is a continuous range value (*uncertain value*) where lb and ub are the lower bound and upper bound values of v_i, respectively. The possible values of v_i lie at any point within the range $[lb{:}ub]$ including the endpoints. For instance, the value of the second dimension of object $o_3(6, [3{:}7], 2, 10)$ is in the form of a continuous range value.

Definition 6 (Exact Value): A value v_i is an exact value when its precise value is known. For instance, the values of the first, third, and fourth dimensions of object $o_3(6, [3{:}7], 2, 10)$ are examples of exact values.

Figure 2 shows five objects with two dimensions, namely: salary and distance. It demonstrates that the salary values of objects $b1$ and $c2$ are in continuous range values with a lower bound and an upper bound limit. Although $c2$ is better than $b1$ in *distance*, we cannot certainly conclude that $c2$ is better than $b1$ in *salary* since the exact values of both objects are not known.

Definition 7 (Data Stream): A data stream D is an infinite sequence of objects, $D = \{o_1, o_2, \ldots\}$ with each o_i has a timestamp, $t_{arr}(o_i)$, indicating the arrival time of the object in the stream.

Definition 8 (Query Over Data Stream): A query, q_i, over a data stream, D, commonly known as *continuous query*, is specified with a *sliding window* (*moving window*), $w_j[q_i]$, that is either (i) a *timestamp-based window* of duration t; with each time-based window, $w_j[q_i]$, consists of the set of objects whose arrival timestamp is within the time t. The window $w_j[q_i]$ is defined by two parameters *range* and *slide*, where *range* specifies the length of the window extent and slide specifies the step by which the window extent moves, (ii) *sequence-based window* (*count-based window*) is a fixed window size j where each count-based window, $w_j[q_i]$, is the set of j most recent objects to arrive. Given k queries, $Q = \{q_1, q_2, \ldots, q_k\}$, if each query, q_i, in Q is processed individually/separately then the processing approach is called *single query processing*; otherwise the term *multi queries processing* is used instead. In addition, if the query utilizes the skyline operator, then it is named *continuous skyline query*. Examples of query over data stream are as follows: a user may ask the most preferable deals advertised during the recent 3 hours and want the results to be updated every 30 minutes, while another user may request the results to be updated every 15 minutes over the past 30 minutes.

III. REVIEW METHODOLOGY

To address the objectives of the research, a systematic literature review approach is adopted. To ensure reproducible, transparent, and scientifically suitable systematic reviews, the *Preferred Reporting Items for Systematic Reviews (PRISMA) 2020 guidelines* [74] is closely followed. Consequently, our review methodology consists of five steps including review questions, selection of information sources, search strategy, criteria used in making the selections, and data extraction and analysis.

First, we defined the following review questions (RQ):

RQ1: How does research in skyline queries over data stream evolve over time?
RQ2: What are the types of skyline query focused by the studies and what are the techniques employed in processing these queries?
RQ3: What are the types of data stream considered in the studies related to skyline queries?
RQ4: What is the query processing approach employed by each study in processing the continuous skyline queries?
RQ5: What are the indexing techniques employed by the studies in facilitating the skyline computation?
RQ6: What is the type of data set commonly used in the analysis conducted by the studies of skyline query on streaming data?
RQ7: What are the performance metrics frequently used in the studies of skyline query over data stream?

A. INFORMATION SOURCE AND SEARCH STRATEGY

The systematic literature review conducted in this work included an advanced search of the available literatures between the years 2000 and 2022. To ensure that none of the relevant studies are missed, we utilized the seven well-known electronic databases that are: *Web of Science* (*WoS*), *IEEE explore*, *Scopus*, *Wiley*, *Science Direct*, *ACM*, and *Springer*. Meanwhile, to avoid having a limited number of publications, a collection of digital libraries that covers a wide range of articles is selected. Furthermore, motivated by several literature reviews like [57], [66], [69], [80], and [96], a comprehensive search string is created that is relevant to the topic being investigated. These search terms are sufficient to match the article title, abstract, and keywords of a large number of papers. The exact search strings and keywords used for each information source are shown in Table 2.

To strengthen the integrity and credibility of the study, only journal articles are included. Specific inclusion and exclusion criteria are established to include only those studies that are pertinent to the aim of the study. These criteria are shown in Table 3. On the basis of exclusion and inclusion criteria, additional parameters (language: English; document type: articles) are utilized to further refine the search results in

each database. In September 2022, a final search across all databases is conducted.

B. DATA COLLECTION AND ANALYSIS

The systematic literature review is conducted in five stages in accordance to the *PRISMA 2020 guidelines* [74]. A *preliminary review* of the literature made up the first stage. A total of 1,075 articles were identified in the selected electronic databases with the following distributions: *WoS* ($n = 174$, 16.2%), *Scopus* ($n = 132$, 12.3%), *IEEE* ($n = 184$, 17.1%), *ScienceDirect* ($n = 62$, 5.8%), *ACM* ($n = 82$, 7.6%), *Springer* ($n = 379$, 35.2%), and *Wiley* ($n = 62$, 5.8%). Based on the EX1 and EX2 exclusion criteria and IC1 and IC2 inclusion criteria, 99 articles and 103 articles are ruled ineligible by human and database automation tools, respectively. These 202 articles are eliminated due to the language used and the type of publications. Meanwhile, considering the EX3 exclusion and IC3 inclusion criteria and utilizing the Microsoft Excel application, 45 articles are found to be duplicated and hence are omitted. The first stage results in 828 articles that are further reviewed in the second stage. The title and abstract of these articles are thoroughly examined to only include/exclude those that are related/not related to the *database management system* as stated by the inclusion and exclusion criteria, IC4 and EX4, respectively. We browsed the complete article in cases where the abstract is unclear or insufficient. In the third stage, we have identified 16 articles that are not fully text accessible. They are not included in the study following the EX5 exclusion criterion. In the fourth stage, the exclusion criterion, EX6, and inclusion criterion, IC6, are rigorously applied to the remaining 230 articles. This stage results in 23 articles that are related to *skyline* and *data stream*. These 23 articles are thoroughly reviewed in the final stage for relevancy in answering the research questions and the objectives that have been designed for this study. A PRISMA flow diagram as shown in Figure 3 summarized the described procedure.

The final selected articles that met all the exclusion and inclusion criteria presented in Table 3, are further analyzed to extract the required data. At this stage a full text reading is conducted where three researchers are randomly appointed to evaluate each article, after which data are extracted and crosschecked. To simplify further analyses, the data are saved in Excel spreadsheets. The following information is extracted from each article, namely: type of skyline query, proposed method, type of data, type of query processing (single/multiple), indexing technique used (if any), type of data set, and performance metrics employed. This process simplifies the analysis and classification to be conducted in answering our research questions.

IV. RESULTS

In this section, we statistically analyze the results of the SLR that are related to skyline query processing over data stream. Based on the extracted data of the 23 articles, we respond to the analytical questions RQ1 till RQ7.

RQ1: How does research in skyline queries over data stream evolve over time?

Between the year 2000 and 2022, 23 journal articles are identified to have studied the issues related to skyline queries over data stream. This is illustrated in Figure 4, where the number of articles that appeared in each year is presented. From this figure, it is obvious that the topic is attracting considerable interest starting from year 2010 and increased in numbers in year 2013, 2014, and 2016. It is worth noting that the first article was published in 2006 in the Journal *of IEEE Transactions on Knowledge and Data Engineering* [75]. In [75], the skyline queries are processed by taking into account a sliding window covering the most recent objects. Objects are continuously monitored while the skylines are maintained incrementally. Meanwhile, the first work to address skyline queries over *distributed data streams* is reported in [78]. Here, the data streams are assumed to derive from multiple horizontally split data sources. The challenges in computing skyline queries over *uncertain data stream* are first raised in [79]. Several probabilistic skyline algorithms are then developed [79], [80], [82], [83], [86], [87]. On the other hand, several works like [84], [85], and [95] investigated the parallel skyline query problem over uncertain data stream. In these works, parallelization is employed during the process of computing skylines. Nonetheless, the recent work related to skyline queries over data stream was published in 2022 in *the IEEE Transactions on Computers* [97]. The work in [97] focuses on handling the spatial-keyword skyline queries over *geo-textual streams* to continuously obtain good skyline results. Meanwhile, Figure 5 presents the total number of citations of the articles according to the year of publication. Intuitively, the recent articles show lesser number of citations as compared to the older one.

Among one of the essential technologies in today's century is sensor technology that is widely used for environmental monitoring and event surveillance, including habitat monitoring, agricultural monitoring, and forest fire detection. A skyline query can be used to monitor the extreme sensed data that are continuously generated over data streams. Since the sensor devices have limited battery power, memory size, and data processing capability, designing an energy efficient method for sensor devices to process skyline query, is one of the most challenging tasks. Hence, it is expected that research in this field will continue to evolve especially with the introduction of new technologies like sensor networks, IoT, fog computing, etc.

RQ2: What are the types of skyline query focused by the studies and what are the techniques employed in processing these queries?

Due to the unbounded length of data streams, computing the skylines is a significant challenge [75]. To process the skyline queries over a data stream, the sliding window has been widely used. For instance, in [84], [85], [86], [90], and [95], a count-based window of the most recent n received objects is employed in computing the skylines. Thus, in a count-based window, the number of active objects remains

TABLE 2. Search strategy and information source.

Information Source	Search Strings and Keywords
Scopus	TITLE-ABS-KEY (("skyline quer*" OR "skyline computation" OR "skyline query processing" OR "Query processing" OR "continuous quer*" OR "preference quer*") AND ("data stream*" OR "streaming data" OR "certain data stream*" OR "uncertain data stream*") AND (database OR "database system*"))
Web of Science	TI=("skyline quer*" OR "skyline computation" OR "skyline query processing" OR "Query processing" OR "continuous quer*" OR "preference quer*" AND "data stream*" OR "streaming data" OR "certain data stream*" OR "uncertain data stream*" AND database OR "database system*") AND AB=("skyline quer*" OR "skyline computation" OR "skyline query processing" OR "Query processing" OR "continuous quer*" OR "preference quer*" AND "data stream*" OR "streaming data" OR "certain data stream*" OR "uncertain data stream*" AND database OR "database system*") AND AK=(" skyline quer*" OR "skyline computation" OR "skyline query processing" OR "Query processing" OR "continuous quer*" OR "preference quer*" AND "data stream*" OR "streaming data" OR "certain data stream*" OR "uncertain data stream*" AND database OR "database system*")
IEEE	("Document Title":"skyline quer*" OR "Document Title":"skyline computation" OR "Document Title":"skyline query processing" OR "Document Title":"Query processing" OR "Document Title":"continuous quer*" OR "Document Title":"preference quer*" AND "Document Title":"data stream*" OR "Document Title":"streaming data" OR "Document Title":"certain data stream*" OR "Document Title":"uncertain data stream*" AND "Document Title":database OR "Document Title":"database system*") AND ("Abstract":"skyline quer*" OR "Abstract":"skyline computation" OR "Abstract":"skyline query processing" OR "Abstract":"Query processing" OR "Abstract":"continuous quer*" OR "Abstract":"preference quer*" AND "Abstract":"data stream*" OR "Abstract":"streaming data" OR "Abstract":"certain data stream*" OR "Abstract":"uncertain data stream*" AND "Abstract":database OR "Abstract":"database system*") AND ("Index Terms":"skyline quer*" OR "Index Terms":"skyline computation" OR "Index Terms":"skyline query processing" OR "Index Terms":"Query processing" OR "Index Terms":"continuous quer*" OR "Index Terms":"preference quer*" AND "Index Terms":"data stream*" OR "Index Terms":"streaming data" OR "Index Terms":"certain data stream*" OR "Index Terms":"uncertain data stream*" AND "Index Terms":database OR "Index Terms":"database system*")
Science Direct	Title, abstract or author-specified keywords (("skyline query" OR "skyline computation" OR "skyline query processing" OR "preference query") AND ("data stream" OR "certain data stream" OR "uncertain data stream") AND (database OR "database system"))
Wiley	("skyline quer*" OR "skyline computation" OR "skyline query processing" OR "Query processing" OR "continuous quer*" OR "preference quer*") AND ("data stream*" OR "streaming data" OR "certain data stream*" OR "uncertain data stream*") AND (database OR "database system*")
ACM	TITLE, Abstract and Keywords (("skyline quer*" OR "skyline computation" OR "skyline query processing" OR "Query processing" OR "continuous quer*" OR "preference quer*" AND "data stream*" OR "streaming data" OR "certain data stream*" OR "uncertain data stream*" AND database* OR "database system*"))
Springer	("skyline quer*" OR "skyline computation" OR "skyline query processing" OR "Query processing" OR "continuous quer*" OR "preference quer*") AND ("data stream*" OR "streaming data" OR "certain data stream*" OR "uncertain data stream*") AND (database OR "database system*")

TABLE 3. Inclusion and exclusion criteria used in our work.

	Inclusion criteria		Exclusion criteria
IC1	Journal articles published between 2000 and 2022	**EX1**	Conference papers, proceedings, reviews, books, other nonpeer-reviewed publications, and book chapters.
IC2	Articles written in English	**EX2**	Articles not written in English
IC3	Articles that are not in another databases	**EX3**	Articles that are in another database
IC4	Articles that are related to the *database management system*	**EX4**	Articles that are not related to *database management system*
IC5	The full text of the article is available	**EX5**	The full text of the article is not available
IC6	Articles that are related to *skyline query processing* and *data stream*	**EX6**	Articles that are not related to *skyline query processing* and *data stream*

constant. The arriving of m new objects will result in m oldest objects being discarded. Meanwhile, the continuous skyline queries are processed using a time-based window in [75], [76], [77], [78], [79], [80], [81], [82], [83], [87], [88], [89], [91], [92], [93], [94], [96], and [97] while both count-based and time-based sliding windows are used in [78]. In a time-based window, the number of active objects may not be constant since only objects with arrival time within the last t time instances are considered active.

In [75] and [76], the *Lazy and Eager* and *LookOut* algorithms that process skyline queries over data streams are proposed, respectively. The *Lazy and Eager* algorithms con-

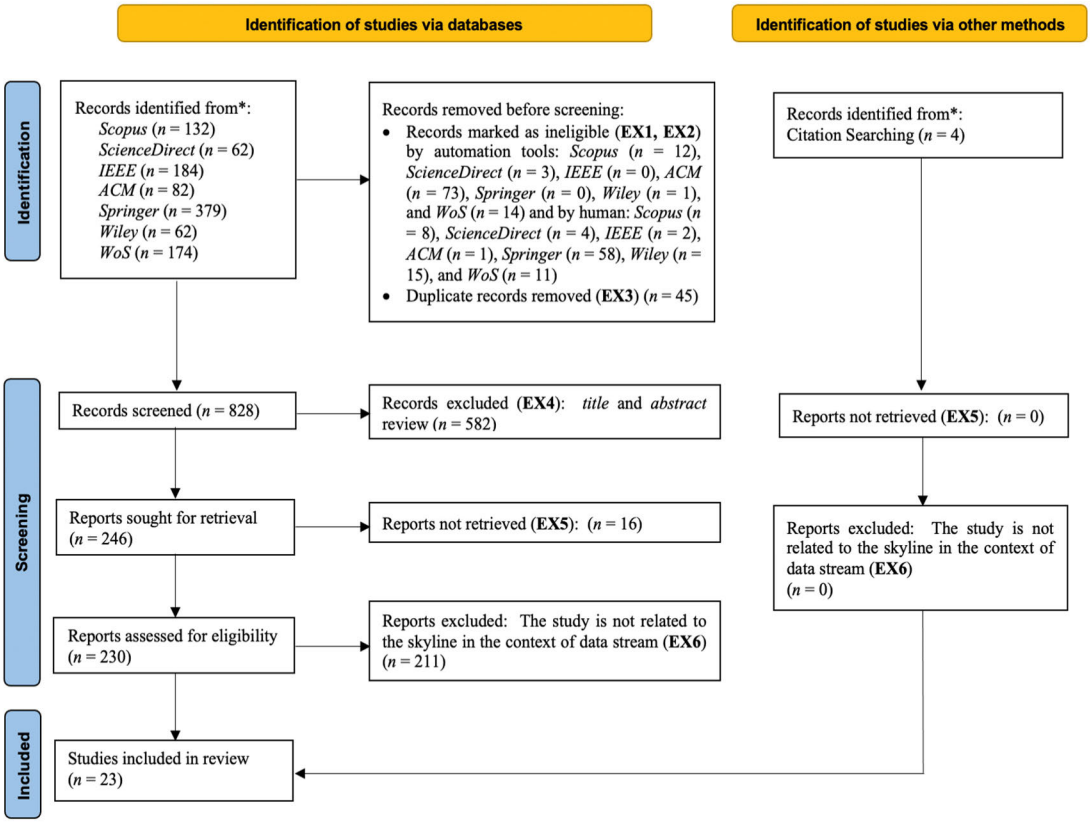


FIGURE 3. PRISMA flow diagram [74].

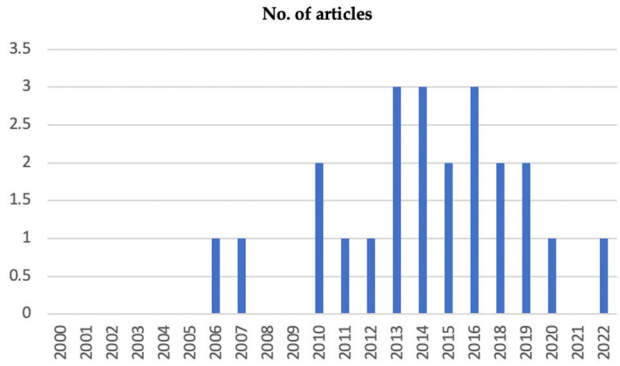


FIGURE 4. Number of articles per year.

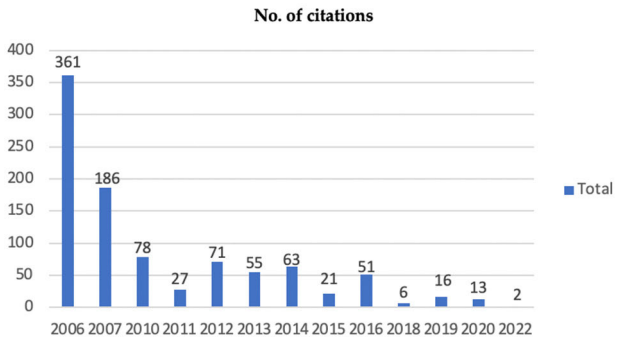


FIGURE 5. Number of citations per year.

tinuously monitor the incoming objects and incrementally maintain the skylines [75]. The database in [75] is divided into two, namely: DB_{sky} and DB_{rest} for storing objects that are and are not the current skylines, respectively. Some objects in DB_{rest} will eventually be skylines whenever a skyline object in DB_{sky} expired. While, the *LookOut* algorithm is proposed in [76] to efficiently evaluate the continuous time-interval skyline queries. Additionally, a new skyline operator is introduced named *continuous time-interval skyline* to continuously evaluate a skyline over multidimensional objects wherein each object is valid for a particular time range. Furthermore, with the assumption that the speed of skyline computation is critical, [76] has opted the quadtree as the choice of index structure.

An efficient and scalable two-phase approach to return the skyline objects in different subspaces is proposed in [77]. The *Naïve Maintenance Algorithm* (*NMA*) is employed in the first phase to produce the seed skylines from the full-space skylines. In the second phase, three novel pruning techniques, namely: *timestamp-based pruning* (*TSP*), *attribute based pruning* (*ATP*), and *timestamp-and-attribute based pruning*

(*TAP*), are utilized to improve the maintaining efficiency. Both the *TSP* and *ATP* work by pruning the cells that overlap with the dominance regions of incoming objects; while the *TAP* prunes the cells that overlap with the anti-dominance regions of incoming objects. To provide an efficient mechanism for evicting expiring valid objects, [77] stored all the valid objects in a single list and they are evicted in a first-in-first-out manner. The new arrival objects are positioned at the end of the list while those that fall out of the window are discarded from the head of the list. For each cell, three pointer lists are constructed each pointing to the corresponding inactive objects, candidate full-space skylines, and full-space skylines, respectively.

To address the issues related to skyline queries over distributed data streams, the work in [78] has presented an efficient and effective algorithm called *Based On Changes of Skylines* (*BOCS*). All objects at the remote site are saved in a local buffer and observed during the current timestamp. Once the timestamp changes, the local skylines on remote sites are updated and maintained by an efficient centralized algorithm named *GridSky*. Here, the skyline increment, *delta skyline*, is calculated by observing the difference between the current skylines and the skylines of the latest timestamp, which is then sent to the coordinator. A central buffer is utilized to preserve the *delta skylines*. This is then followed by the process of integrating the remote increments with the latest global skylines to obtain the final global skylines. Both the *GridSky* and the associated communication protocol are parts of *BOCS*.

Subsequently, skyline algorithms that are based on probability computation on uncertain data stream have been presented in [79], [80], and [82]. A probabilistic skyline algorithm, *PSkyline*, is developed in [79] to compute exact skyline probabilities of all objects in a given uncertain data set. Furthermore, a new in-memory tree structure called *Z-tree* is employed by *PSkyline* to increase the likelihood of finding incomparable groups of instances and dispensing with unnecessary dominance tests altogether. In addition, the online probabilistic skyline algorithm, *O-PSkyline*, and top-k probabilistic skyline algorithm, *K-PSkyline*, are developed to find the skyline objects for online uncertain data streams and top-k objects with the highest skyline probabilities, respectively. In [80], a novel sliding window skyline model is proposed where the *Wp-Skyline* (p, t) is calculated to identify those objects whose probabilities of becoming skylines are at least p at timestamp t. To maintain the list of candidates that might become skylines in the future sliding windows, the *candidate list* (*CL*) *approach* is introduced. Several algorithms and an enhanced refinement strategy, a combination between a multi-dimensional indexing structure and grouping-and-conquer strategy, are proposed with the aim to maintain the skyline candidate set incrementally by continuously monitoring the newly incoming and expired objects. Similar to [79] and [80], [82] investigated the problem of efficiently computing skyline against sliding windows over an uncertain data stream and formally defined the probability threshold based skyline problem. Several techniques are introduced as follows: *SSKY* to continuously compute skyline with the probability no less than a given q, q-skyline, against a sliding window, *MSKY* to continuously compute multiple q-skylines concurrently of multiple given probability thresholds, *QSKY* to process a skyline query with an ad hoc probability threshold, *TOPK* to retrieve k objects with the largest skyline probabilities, and *TIMESTAMP* to continuously compute q-skyline against the timestamp based sliding window model. Also, the minimum information needed in continuously computing the probabilistic skylines against sliding window is characterized which is efficiently kept and maintained in a candidate set.

A variety of skyline queries that employ probability computation over uncertain data stream have been investigated which include *reverse skyline queries* [83], *probabilistic subspace skyline* [86], and *n-of-N skyline queries* [87]. Focusing on the problem of continuously processing reverse skyline queries on sliding windows against uncertain data streams, [83] proposed the *Continuous Probabilistic Reverse Skyline* (*CPRS*) algorithm which is based on the R-tree to process the queries over the most recent n uncertain objects. Meanwhile, to tackle the problem of probabilistic subspace skyline query processing over sliding windows on uncertain data streams, [86] developed the *probabilistic subspace skyline* (*PSS*) algorithm using a regular grid indexing structure. *PSS* retrieves all objects from the most recent window of streaming data in a user-selected subspace with a skyline probability no smaller than a given threshold. Two variants of *PSS* called *Approximate PSS* (*APSS*) and *CUDA-based PSS* (*CPSS*) are also introduced to deal with high-dimensional databases and very large databases, respectively. On the other hand, in dealing with the problem of efficiently computing probabilistic skylines against the most recent n uncertain objects in a data stream, the following are developed in [87]: (i) an efficient pruning technique to minimize the number $\mathcal{N}(\mathcal{N} \leq N)$ of uncertain objects to be kept in the most recent N objects for processing all probabilistic n-of-N queries, (ii) a novel encoding scheme on the stored objects together with the efficient update algorithms based R-tree and interval tree techniques, and (iii) a trigger based technique for continuously processing probabilistic skyline query following the *n-of-N* model. Then, several algorithms are proposed and evaluated for *Pn-of-N* queries, that are: the stabbing query processing algorithm (*pnN*), an algorithm for continuously maintaining the data structures for supporting *Pn-of-N* queries (*pmnN*), and the continuous query processing algorithm (*pcnN*) for continuously outputting the *Pn-of-N* results. Moreover, following the probability threshold fashion, each object in the data stream is associated with an occurrence probability and maintained in a minimum candidate set.

In contrast, instead of relying on probability calculations, the works in [84], [85], and [95] employed parallelization in processing skylines over uncertain data stream. For instance, the work in [85] proposed an effective framework named

distributed parallel framework (*DPF*) to address the parallel skyline query problem. Moreover, the *parallel streaming skyline* (*PSS*) approach with an optimized streaming item mapping strategy and the grid index are proposed to further optimize the parallel skyline computation. The parallel query processing settings utilized in [85] consist of three types of nodes, namely: (i) *monitor node* that is responsible for delivering the arriving streaming objects to the parallel computing nodes, (ii) *peer nodes* that are responsible for computing the skyline probabilities for new streaming objects and maintaining the objects in its own sliding window, and (iii) *query node* that is responsible for continuously collecting the skyline results from all peer nodes. The work in [84] extended their earlier solution in [85] to handle the parallel skyline query problem over uncertain data streams in cloud computing algorithms. They proposed three parallel models, namely: *Simple Parallel Model* (*SPM*), *Alternate Parallel Model* (*APM*), and *Distributed Parallel Model* (*DPM*) based on the sliding window partitioning. Furthermore, an adaptive sliding granularity adjustment strategy and a load balance strategy are proposed to further optimize the queries. Meanwhile, to meet the query requirements of different window scales at the same time, the work in [95] proposed a framework for parallelizing the query computation for uncertain *n-of-N* skylines. Several strategies are introduced, namely: (i) a sliding window partitioning strategy and a streaming items mapping strategy to realize the load balance for each node; (ii) a spatial index structure *RST* based on *R*-tree to organize the objects within each individual sliding window and candidate set; (iii) an encoding interval scheme to transform the *n-of-N* query into stabbing query in each compute node; and (iv) a red-black tree named *RBI* to store all stabbing intervals. Similar to [84] and [85], the parallel iterative query contains three kinds of nodes that are (i) *monitor node* for maintaining the global sliding window and delivering the new arriving streaming objects to the parallel compute nodes, (ii) *compute node* for maintaining its own sliding window and computing the *n-of-N* skyline query, and (iii) *collector node* for gathering the uncertain *n-of-N* skyline query results from all compute nodes. Initially, the global sliding window is partitioned into several local sliding windows to the given strategy to achieve load balance. Meanwhile, the new arriving objects are passed alternately among the compute nodes to improve the utilization rate of the compute nodes. To maximally reduce the extra communication overhead, the non-blocking modes is used to communicate between the computation nodes. Finally, the global skyline query results are gathered by the collector node from each compute node.

Unlike, the skyline algorithms presented earlier that are tailored for processing a single continuous skyline query, the *FAst Skyline compuTation for multiple queries* (*FAST*) method in [81] is introduced for processing multiple continuous skyline queries over a data stream. *FAST* uses a filtering technique and a discriminant to discard an object that will not be a member of any future skyline of continuous queries and determine which objects in memory are skyline objects for which queries, respectively. An efficient semidominance based approach called *Semidominance Based Reverse Skyline* (*SDRS*) is proposed in [88] to efficiently evaluate continuous reverse skyline queries over sliding windows. To minimize the number of objects to be kept in the sliding window, an effective pruning approach utilizing the semidominance relationships and the first-in-first-out property of the sliding window is presented in [88]. Moreover, an extension is also proposed to handle the *n-of-N* and ($n1$, $n2$)*-of-N* reverse skyline queries. Skyline group problem which involves finding k-object groups that cannot be dominated by any other k-object group is investigated in [89]. Efficient algorithms are proposed to find and update skyline groups incrementally over data stream by reusing dominance information. Moreover, hash table, dominance graph, and a dynamic programming matrix are employed to store dominance information and update the results incrementally. Specifically, the dominance graph is used to store reusable candidate objects while the matrix is incrementally constructed based on the selected candidate objects.

The problem of calculating k *representative skyline* over data streams is explored in [90]. A new criterion is proposed to choose the k skylines as the representation to the entire data set in the stream, named *k largest dominance skyline* (*k-LDS*). To solve the *k-LDS* problem in a 2 dimensional space, the *Prefix-based Algorithm* (*PBA*) is introduced, while a *greedy algorithm* is designed to answer the *k-LDS* queries for a d-dimensional space with $d \geq 3$. The work in [91] argued that most of the past works rely on sequential algorithms in processing the continuous skyline queries over data streams. In [91], a parallelization of the eager algorithm based on the notion of *Skyline Influence Time* (*SIT*) is proposed to solve the problem of parallel skyline queries. The *SIT* of an incoming object p denoted by SIT_p is defined as the minimum time in which p may become a skyline object. If SIT_p is reached and p is still an active object, then p is added to the skyline. Nonetheless, if a younger object r dominates p before SIT_p is reached, then p is discarded. While to achieve near-optimal speedup, the optimizations of the reduce phase and several load-balancing strategies are developed. The *ρ-dominant skyline query* based on data stream is explored in [92]. The query controls the base of the result set by adjusting data proportion. A method is then proposed in [92] named *ρ-Dominant Skyline Query on data stream over Sliding Window* (*DSSW*) in order to improve the efficiency of ρ-dominant skyline query. The skyline set S_N is divided into three groups that are: (i) *ρ-Dominant Skyline Point* (D_N) – the set of objects in S_N which are not *ρ-dominated* by any other points, (ii) *Candidate Point* (C_N) – the set of objects in S_N which are *ρ-dominated* by an older object rather than a younger object, and (iii) *Candidate Point* (A_N) – the set of objects in S_N which are *ρ-dominated* by a younger object but not fully-dominated by any younger objects. The work is then extended to *n-of-N ρ-dominant skyline query* and ($n1$,$n2$)*-of-N ρ-dominant skyline query* in data stream.

Unlike most of the existing works that assume the data in the data streams are complete and available, the works in [93] and [94] focus on incomplete data sets, i.e. data sets with missing attribute values or missing objects. Two algorithms are proposed in [93], namely: *kISkyline* algorithm that is based on the traditional sliding window model with a split bucket strategy and *sISkyline* algorithm that is based on a real point, virtual point, and shadow point with a split bucket strategy along with the traditional sliding window model. The split bucket method divides the incomplete points. This is then followed by the virtual point and shadow point method with the aim to reduce the number of comparisons in the sliding window and consequently reduce the size of the candidate skyline points. As a result, the bottleneck problem of *kISkyline* algorithm in computing the global skyline points is resolved by the *sISkyline* algorithm. On the other hand, to solve the problem of skyline query over incomplete data stream (*Sky-iDS*), an efficient *Sky-iDS query answering* algorithm is proposed in [94]. The algorithm integrates several techniques, that are: (i) differential dependency (*DD*) rules to impute missing attributes of objects from incomplete data stream, (ii) effective pruning strategies, namely: *spatial pruning*, *max-corner pruning*, and *min-corner pruning* to greatly reduce the search space of the *Sky-iDS* problem, and (iii) cost-model-based index structures, i.e. data synopses and *skyline tree* (*ST*) indexes, to facilitate the data imputation via *DD* rules and skyline computation by dynamically maintained the *Sky-iDS* candidates. Meanwhile, the index structures I_j is constructed over a complete data repository R to facilitate missing data imputation. The *Sky-iDS* query answers from incomplete data stream are continuously monitored by utilizing both the data synopsis *ST* and indexes I_j, following the style of imputation and query processing at the same time.

Motivated with the problem of multidimensional skyline queries over streaming data, [96] proposed a *Multidimensional Skyline over Streaming Data* (*MSSD*) framework that contains three data structures, namely: a buffer β, a data set $\mathcal{R}$, and an index structure named *Negative SkyCube with timestamps* (*NSCt*) to store the summary of subspaces where each object is dominated during its lifetime. The micro-batch processing approach is adopted where the stream source emits one object every θ units of time. The *MSSD* gathers the objects into β during k units of time. These buffered objects are inserted into $\mathcal{R}$ while the outdated objects are removed from $\mathcal{R}$. Meanwhile, the *NSCt* is called to compute the skyline whenever a subspace skyline query is issued. On the other hand, [97] proposed a distributed skyline query processing framework based on the distributed top-k spatial-keyword query processing framework for large-scale spatial-keyword publish/subscribe systems. The framework is based on Apache Storm and consists of six components, namely: *query spouts*, *tuple spouts*, *distribution bolts*, *query bolts*, *tuple bolts*, and *aggregation bolts*. Once query bolts send requests due to new objects arrive or old objects expire, both the tuple bolts and aggregation bolts cooperatively execute the skyline computing. Each tuple bolt maintains a geo-textual index and generates a partial skyline result over its local sliding window. Then, the global skylines are computed by the aggregation bolts based on the partial skyline results from all tuple bolts. The final skyline results are then sent to the query bolts. Moreover, to efficiently index geo-textual streaming objects in the framework, an update-efficient and space-saving indexing structure, *Memo-and-Filter-based Rt-tree* (*MF-Rt-tree*), is proposed. Furthermore, to suit with the streaming setting, a fast processing approach is introduced for processing a continuous spatial-keyword skyline query. In addition, to reduce the communication cost of the proposed framework, a novel communication optimization method employing the spatial cuckoo filter and one-permutation min-wise signature method for keyword pruning is developed.

Since the introduction of the skyline operator, numerous types of skyline queries have been introduced, each focusing on solving problems related to a specific type of application. In data stream, these skyline queries differ with regard to the *features* that are considered in deriving the skyline objects which among others include the *searching space* whether total (*full space skyline queries*) or partial (*subspace skyline queries*), the occurrence probability of the objects (*probabilistic skyline queries*), the recency of the skyline objects (*n-of-N skyline queries*), the query point whether dynamic or static (*dynamic skyline query*, *reverse skyline query*), etc. These features are taken into consideration when designing the skyline algorithms. In addition, the time-varying (time-sensitive), continuous, real-time, volatile, and unpredictable characteristics of data streams have led to methods that attempt at eliminating needless skyline computation. Intuitively, indexing techniques are widely employed, along with distributed and parallel processing.

The query type and the proposed technique of each study are summarized in Table 4.

RQ3: What are the types of data stream considered in the studies related to skyline queries?

From the 23 articles being analyzed, 12 of them [75], [76], [77], [78], [81], [88], [89], [90], [91], [92], [96], [97] focused on *certain data stream*, i.e. the values of each object in the stream are assumed deterministic, precise, and available during the processing of the skyline queries. The main challenge faced by these studies is handling the data streams that are known to have the properties of time varying (time sensitive), continuous, real time, volatile, and unrepeatable. In addition, the answers to the queries are presumed to be timely and continuously updated to reflect the changes in the states of the environments. Meanwhile, with the arguments that uncertainty is inherent and inevitable in many real applications, while the amount of uncertain data collected and accumulated in a streaming fashion is rapidly increasing, 9 of the 23 articles [79], [80], [82], [83], [84], [85], [86], [87], [95] have explored the *uncertain data stream* following the discreate uncertainty model where an object o_i is modelled as a probability distribution that is defined on a finite set of l instances (states), $o_{i1}, o_{i2}, \ldots, o_{il}$ (see *Definition* 3). Uncertain data

TABLE 4. The type of skyline queries and their proposed technique.

Reference	Query Type	Sliding Window		Proposed Technique
		Time-based	Count-based	
[75]	skyline query	✓	–	*Lazy and Eager*
[76]	skyline query	✓	–	*LookOut*
[77]	subspace skyline query	✓	–	*NMA*, *TSP*, *ATP*, *TAP*
[78]	skyline query	✓	✓	*BOCS*, *GridSky*
[79]	skyline query	✓	–	*PSkyline*, *O-PSkyline*, *K-PSkyline*
[80]	skyline query	✓	–	*Wp-Skyline*, *CL approach*
[81]	multiple skyline queries	✓	–	*FAST algorithm*
[82]	skyline query	✓	–	*SSKY*, *MSKY*, *QSKY*, *TOPK*, *TIMESTAMP*
[83]	reverse skyline query	✓	–	*CPRS*
[84]	parallel skyline query	–	✓	*SPM*, *APM*, *DPM*
[85]	parallel skyline query	–	✓	*DPF*, *PSS*
[86]	subspace skyline query	–	✓	*PSS*, *APSS*, *CPSS*
[87]	probabilistic *n-of-N* queries	✓	–	*pnN*, *pmnN*, *pcnN*
[88]	reverse skyline queries, *n-of-N*, $(n1, n2)$-*of-*$\mathcal{N}$	✓	–	*SDRS*
[89]	skyline group query	✓	–	*skyline group algorithm*
[90]	k representative skyline query	–	✓	*PBA*, *greedy algorithm*
[91]	parallel skyline query	✓	–	*eager algorithm*
[92]	ρ-dominant skyline query, *n-of-N* ρ-dominant skyline query, $(n1, n2)$-*of-N*ρ-dominant skyline query	✓	–	*DSSW*
[93]	skyline query	✓	–	*kISkyline*, *sISkyline*
[94]	skyline query	✓	–	*Sky-iDS*
[95]	parallel *n-of-N* skyline query	–	✓	*A sliding window partitioning strategy, a streaming items mapping strategy, RST, RBI*
[96]	Multidimensional skyline query	✓	–	*MSSD*
[97]	Spatial keyword skyline query	✓	–	*Distributed skyline query processing framework*

are unavoidable with the emergence of a large number of practical applications in domains like sensor network, radio frequency identification (*RFID*) network, data cleaning and integration, online shopping, trend prediction, location-based service, moving object management, network traffic analysis, *GPS* systems, radar detection, economic decision making, and market surveillance [80], [82], [84], [85], [86], [87], [95]. The presence of data uncertainty is primarily due to various factors such as limitations of measuring equipment, delay or loss in data updates/transfer, data randomness and incompleteness, interference of external environment, and privacy preservation [80], [82], [84], [85], [87], [95]. Here, in dealing with skyline queries over uncertain data stream, not only the dominant relationships between objects need to be examined but also the probability of each object to be a skyline needs to be calculated which is computationally expensive. This implies that the traditional pruning strategy cannot be used directly. On the other hand, only 2 articles [93], [94] have tackled the issue of *incompleteness of data stream* where objects of the stream may have one or more missing attribute values. In practice, the incompleteness of data is often due to device anomalies, imperfect data collection techniques, environmental factors, and privacy protection [93], [94].

Data uncertainty is normally attributed to the degree to which *data* are inaccurate, imprecise, untrusted, and unknown. Data streams come from a wide variety of sources and in many different formats that they may have missing or uncertain values. Having a rigid assumption that the data streams are always certain (complete and precise) would limit the solutions to certain type of applications. Exploring the various types of data uncertainty is crucial as today's advancement in technology shows that uncertain data are inevitable in many emerging applications.

RQ4: What is the query processing approach employed by each study in processing the continuous skyline queries?

It is observed that in most articles [75], [76], [77], [78], [79], [80], [82], [83], [84], [85], [86], [87], [88], [89], [90], [91], [92], [93], [94], [95], [96], [97], the *single query processing* approach (see *Definition* 8) is employed by which the proposed techniques designed in [75], [76], [77], [78], [79], [80], [82], [83], [84], [85], [86], [87], [88], [89], [90], [91], [92], [93], [94], [95], [96], and [97] are performed in such a way that a single continuous skyline query is processed at a time over the data stream. This means given k queries, $Q = \{q_1, q_2, \ldots, q_k\}$, the proposed technique is repeated for each query, q_i, in Q, i.e. the k continuous skyline queries are

processed independently over the data stream. Nonetheless, the solution proposed in [81] applied the *multi queries processing* approach where multiple continuous skyline queries can run concurrently; with each query may have different range and slide values. This is to avoid duplication of the same or similar partial skyline computations owing to these multiple queries that are issued over the data stream may involve several objects in common [81].

To accommodate a wide range of users and applications, the skyline algorithms should be effective enough to manage multiple queries over a collection of continuously generated input data streams and analyze these data streams in close to real-time to offer fast response. It is unwise to process each user's query individually because the same streams of objects will be analyzed repeatedly. On the other hand, the objects that have been scanned might reappear in the stream at a different time. The domination analysis that has been conducted over these objects might have to be repeated at a later time, which result in unnecessary re-computation of the domination analysis. Thus, dealing with the aforementioned issues would further reduce the skyline computation time.

RQ5: What are the indexing techniques employed by the studies in facilitating the skyline computation?

A variety of indexing techniques have been employed by most of the 23 articles analyzed with the main aim to reduce the skyline computation cost by pruning irrelevant objects. Most of them made use of the *R*-tree indexing structure [79], [80], [82], [83], [87], [88], [89] due to its simplicity and its reasonable performance for real-world data [75]; while others employed the R^*-tree [75], [81], and quadtree indexing structures [76], [81]. Several notable experimental studies have shown that quadtrees can manage objects more effectively than the *R*-tree family [76]. As reported in [76], the quadtree index has significantly speeds up skyline computation by up to an order of magnitude or more in some cases, and is never slower than the R^*-tree approach. Using the quadtree also results in smaller memory consumption. Nonetheless, the work in [75] has utilized different indexing techniques to index the objects. These include *linked list* and R^*-tree to manage the objects of *Lazy* algorithm and *B*-tree and *R*-tree to index the event list of *Eager* algorithm. Similar to [75], the following studies have also incorporated several indexing techniques in their solutions: [79] (*Z*-tree, *R*-tree, *ZB*-trees), [80] (*R*-tree, hash table), [81] (R^*-tree, *PR* quadtree), and [89] (*R*-tree, hash table, dominance graph, matrix). Some of these techniques are utilized to store dominance information to update the skyline results incrementally [89]. Meanwhile, grid index is also a common indexing structure used by the studies [77], [78], [85], [86], [91] where objects are indexed in a multi-dimensional array. There are also studies that have proposed new indexing techniques based on existing established techniques such as [94], [95], [96], and [97], where *skyline tree*, *RST*, *NSCt*, and *MF-Rt-tree*, are introduced, respectively. *RST*, for instance, is a spatial indexing structure based on *R*-tree introduced in [95] to organize the objects within each individual sliding window. On the other hand, *NSCt*, an extended indexing structure of negative skycube is used in [96] to store the lossless summary of the subspaces of objects not in the skyline. Interestingly, the work in [97] has introduced an update-efficient and space-saving indexing structure, *MF-Rt-tree*, to efficiently index the geo-textual streaming objects; *MF-Rt-tree* is developed by fusing and improving the features of *Rt*-tree, cuckoo filter, and *RUM*-tree. Unlike the above studies, [84], [90], [92], and [93] are found not to use any indexing techniques. Table 5 summarizes the various indexing techniques used by each study in facilitating the process of computing skylines.

TABLE 5. Indexing techniques used.

Reference	Indexing Technique Used
[75]	*B*-tree, *R*-tree, *R**-tree, linked list
[76]	Quadtree
[77], [78], [85], [86], [91]	grid index/array
[79]	*Z*-tree, *R*-tree, *ZB*-trees
[80]	*R*-tree, hash table
[81]	*R**-tree, *PR* quadtree
[82], [83], [87], [88]	*R*-tree
[89]	*R*-tree, hash table, dominance graph, matrix
[94]	skyline tree
[95]	*RST*, *RBI*
[96]	*NSCt*
[97]	*MF-Rt-tree*
[84], [90], [92], [93]	–

Although there are various indexing techniques, the *R*-tree family has been extensively used by researchers to offer efficient processing of skyline queries in multi-dimensional data sets. Besides its simplicity, it is capable of handling diverse types of objects. An efficient filtering step is provided thru its minimum bounding rectangle (MBR) which leads to considerable decrease in computational and I/O time in comparison to other techniques.

RQ6: What is the type of data set commonly used in the analysis conducted by the studies of skyline query on streaming data?

With the intent to produce original research results and validate the research findings, two main types of data set are commonly used in the analysis of studies related to skyline query. They are *synthetic* and *real data sets*. Most of the studies [75], [76], [77], [78], [79], [80], [81], [82], [83], [84], [85], [86], [87], [88], [89], [90], [91], [92], [93], [94], [95], [96] except [97] have used synthetic data set in their analysis. The main reason is synthetic data are easier to generate and manipulate in a controlled way than real data. It is also observed that most of these studies generated different types of data distributions which are *anticorrelated*, *correlated*, and *independent* to fairly verify the effectiveness of their proposed solutions [75], [76], [77], [78], [79], [80], [81], [82], [83], [84], [85], [86], [87], [88], [90], [91], [92], [93], [94], [95], [96]. To strengthen the research findings, [79],

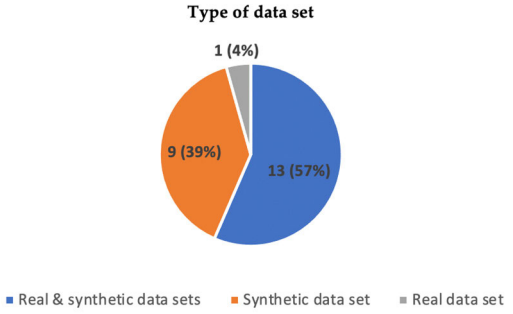


FIGURE 6. Type of data set employed by the studies.

[80], [81], [82], [83], [84], [85], [86], [87], [88], [90], [93], [94], [95], [96] have used both types of data set, i.e. synthetic and real data sets. Meanwhile, various different real data sets have been utilized by these studies, which include *NBA* [79], [93], *Long Beach county road (LBC)* [80], *New York Stock Exchange (NYSE)* [82], [87], *Zillow* [84], [85], *IPUMS* [85], [95], *e-business* [86], *Stock* [88], [90], *Yahoo Financial Website* [88], *Forest environment* [90], *Movie* [93], *Intel lab* [94], *UCI gas sensor* [94], *Antallagma time series* [94], *Pump sensor data* [94], and *tweets* [96], [97]. The only work that does not utilize the synthetic data set is [97] where a real-life data set, *tweets*, is employed to evaluate the performance of their proposed distributed skyline query processing framework. The types of data set used by each study is summarized in Table 6 while Figure 6 presents the number of studies that utilized solely the real data set or synthetic data set, and both types of data set.

The skyline operator is designed to find a set of interesting objects over a large dimensional data collection, thus to validate the efficiency of a skyline algorithm, the selected data sets should resemblance the real data collection. The standard data sets, namely: NBA, MOVIE, and TWEETS as well as synthetic data sets are the typical data sets utilized by researchers working in skyline queries. These standard data sets are also employed in studies relating to skyline query over data streams in which they are manipulated by randomly incorporating the arrival time and expiry time of each object. Moreover, since the standard data sets contain a finite set of objects, synthetic data sets are employed as additional data sets that can be easily prepared according to the distinctive characteristics of data streams, namely: continuous, unbounded, time-sensitive, and high-volume. Furthermore, depending on the type of applications targeted by a solution, the chosen data sets should closely reflect the nature of the applications besides the unique characteristics of the data streams.

RQ7: What are the performance metrics frequently used in the studies of skyline query over data stream?

The data streams are known to be time varying (time sensitive), continuous, real time, volatile, and unrepeatable, hence measuring the *processing time* incurred by any solutions is inevitable. This is evident as all studies reported the processing time [75], [76], [77], [78], [79], [80], [81], [82], [83], [84], [85], [86], [87], [88], [89], [90], [91], [92], [93], [94], [95], [96], [97] as one of the crucial performance metrics in their experiments. However, there is a slight difference between the studies with regard to the time being measured. For instances, *per-tuple* (*per-object*) *processing time* [75], [81], *query response/processing time* [77], [81], [82], [87], [88], [94], [97], *amortized time* [78], [90], *elapsed time* [79], [86], *run time* [80], [93], [97], *response time* [81], [92], *average delay* [82], [88], *time per update* [84], [85], [95], *processing speed/speedup* [84], [91], [97], *execution time* [89], [96], *maximum input rate* [91], *wall clock time* (*maintenance and query times*) [94], *processing time per update* [95], and *time ratio* [96], are among the processing time reported in the studies.

Meanwhile, to validate the effect of employing the quadtree, *heap size* and *number of pages accessed* are the two metrics used in [76] which relate to the amount of memory needed for computing skylines. Similarly, the work in [75] measured the *space consumption* which demonstrated the average amount of memory consumed by their proposed methods, *Lazy and Eager*. Also, in [81], *space efficiency* of the *FAST* method is measured based on the maximum and the average number of objects in memory during execution which include the number of objects waiting in the queue. Meanwhile, in [82], the memory usage of the *SSKY* algorithm is measured based on the maximum number of pages accessed. Furthermore, to test the impact of the grid granularity on *PPS*, *memory consumption* is measured in [86]. The *memory consumption* is also considered in [96] to measure the effect of their proposed index structure, *NSCt*, in storing a summary of subspaces where an object is dominated during its lifetime. Additionally, the *MF-Rt-tree* proposed in [97] to store keywords in indexing nodes is also evaluated with regard to *space cost*.

Nonetheless, *space consumption* is also evaluated by the studies that maintain the candidate/skyline results. This is clearly seen in [77] where the full-space skylines and candidate full-space skylines are stored in a grid-based maintenance algorithm, while in [78] local skylines on remote sites are maintained by *GridSky*. Also, the space usage of the *SSKY* algorithm in terms of maximum candidate size and maximum skyline size in $S_{N,q}$ is evaluated in [82]. Meanwhile, the space usage measured in [88] represents the maximal number of objects kept in the S_N list.

Maintenance performance is also used by several studies like [77] and [80] to evaluate the maintaining efficiency of their proposed solutions. For instances, the *amortized cost* is measured in [77] to evaluate their *NMA* maintenance algorithm in maintaining the full-space skylines and candidate full-space skylines, while the *number of quantified skylines* is the measurement used in [80] to evaluate the efficiency of the *CL* approach and *Wp-Skyline* query procedure in maintaining the candidate set, DB_{cand}. Meanwhile, in [82], the *number of the MBRs accessed* is the metric used to validate the *SSKY* algorithm in storing and maintaining

TABLE 6. Type of data set used.

Reference	Synthetic	Real
[75], [77], [78]	anticorrelated, independent	–
[76], [81], [83], [91]	anticorrelated, correlated, independent	–
[79]	anticorrelated, independent	*NBA*
[80]	anticorrelated, correlated, independent	*Long Beach county road (LBC)*
[82]	anticorrelated, independent	*New York Stock Exchange (NYSE)*
[84]	anticorrelated, independent	*Zillow*
[85]	anticorrelated, independent	*IPUMS*, *Zillow*
[86]	anticorrelated, correlated, independent	*e-business*
[87]	anticorrelated, correlated, independent	*NYSE*
[88]	anticorrelated, clustered, uniform	*Stock*, *Yahoo Financial Website*
[89]	General	–
[90]	anticorrelated, independent	*Stock*, *Forest environment*
[92]	independent, inverse correlation	–
[93]	anti-independent, independent, relevant	*NBA*, *Movie*
[94]	anticorrelated, correlated, uniform	*Intel lab*, *UCI gas sensor*, *Antallagma time series*, *Pump sensor data*
[95]	anticorrelated, correlated, independent	*IPUMS*
[96]	anticorrelated, independent	*Tweets*
[97]	–	*Tweets*

the current skyline and remaining objects. In addition, [82] measured the average maintenance cost of *MYSKY* algorithm in updating each object of the *R*-trees. On the other hand, [87] reported the efficiency of the maintenance technique, *pmnN*, based on the *maintenance time per object* to continuously maintain the data structures for supporting *Pn-of-N* queries. In [88], the *maintenance time* is reported as the time of constructing the *R*-tree index. Furthermore, to evaluate the efficiency of incrementally maintaining their proposed multi-layer tree structure, *Skyline Tree* (*ST*), [94] measured the *maintenance time* which is part of the *wall clock time*. The work in [97] also reported the *maintenance time* in maintaining the *NSCt* index structure.

To study the effect of ϵ on the accuracy of ϵ-*constraint greedy algorithm* (ϵ-*GA*), i.e. the default algorithm of *k-LDS*, [90] measured the *dominance size* and *ratio of dominance size* which are then compared to *GA*. Meanwhile, the *f-score* metric is used in [94] to test the accuracy of the *SkyiDS* approach.

On the other hand, *communication load* is a measurement used by studies that involved data streams in a distributed [78], parallel [84], [85], as well as cloud computing [84] environments in which several nodes are involved in the computation of skylines. For examples, the *total load* incurred by the *BOCS* is measured in [78], while *load balancing* in [84] and [85]. Others, like [91] measured the *offered bandwidth* as a function of the parallelism degree on the Intel architecture, while [97] reported the *communication cost* as an evaluation of their proposed *distributed skyline query processing framework*.

Table 7 presents the list of performance metrics employed by each study while Table 8 summarizes the performance metrics used by each study according to *efficiency*, *space/memory consumption*, *maintenance*, *communication*, and *effectiveness*.

Table 9 summarizes the 23 articles analyzed in this paper that are related to skyline query processing over data stream. It highlighted the reference, the query type, the type of sliding window (time-based and/or count-based), the proposed technique, the type of data stream (certain, uncertain, incomplete), the query processing approach (single or multiple), the indexing technique, the data set (synthetic and/or real), and the performance metric (efficiency, space consumption, maintenance, communication, effectiveness) used by each study. Meanwhile, Figure 7 presents the taxonomy that is derived based on the entries of Table 9. It is presented in three parts, namely: 7(a), 7(b), 7(c); each focusing on a particular environment, i.e. distributed, parallel, and centralized respectively. Based on Figure 7, it is obvious that most studies assumed a centralized environment where a single node is responsible for computing the skylines [75], [76], [77], [79], [80], [81], [82], [83], [86], [87], [88], [89], [90], [92], [93], [94], [96]. Also, most of the studies focused on certain data stream [75], [76], [77], [78], [81], [88], [89], [90], [91], [92], [96], [97]. Most profoundly, there are many unexplored paths as shown by paths without references.

V. DISCUSSION

In the following, we discuss the potential opportunities for further exploration based on the findings of the SLR performed on the 23 articles. Generally, the future directions as highlighted in these articles can be categorized into three, namely: (i) flexibility in query requirements of the users, (ii) adaptive system for extremely fast data streams, and (iii) dirty data sets with data quality problems.

A. FLEXIBILITY IN QUERY REQUIREMENTS OF THE USERS

Skyline queries aim to prune a search space of large numbers of multi-dimensional objects to a small set of interesting objects by eliminating objects that are dominated by others.

TABLE 7. List of performance metrics employed by the studies.

Reference	Performance Metrics
[75]	amortized performance (processing time, space consumption)
[76]	CPU time, maximum size of the heap, number of pages access
[77]	maintenance performance (amortized cost), query performance (query time), space consumption
[78]	processing time, space consumption, communication load, stability testing
[79]	elapsed time
[80]	number of qualified skylines, run time
[81]	per-object processing time, response time, query response time, space efficiency (number of objects)
[82]	maximum/average number of MBR accessed, average delay, space efficiency (maximum candidate size, maximum skyline size, maximum/average page number), time efficiency (delay for each element, number of elements pruned, average maintenance time, average query response time)
[83]	time efficiency (average processing time)
[84]	time per update, number of items updated per second (processing speed), load balance
[85]	time per update, load balance
[86]	elapsed time, memory consumption
[87]	processing time, query time, maintenance time
[88]	average delay, space usage, maintenance time, query processing time
[89]	execution time, number of groups and candidates
[90]	change number, amortized cost, accuracy (dominance size, ratio of dominance size)
[91]	offered bandwidth, speedup, maximum input rate, number of non-obsolete points per window
[92]	response time, the amount of data retained in a sliding window, skyline size
[93]	run time
[94]	*f*-score, wall-clock time, maintenance time, query time
[95]	time per update, processing time per update
[96]	execution time, time ratio, memory usage, maintenance time
[97]	update cost (number of operations, run time), space cost (index size), query cost (run time), and computing cost (speedup), communication cost

TABLE 8. Summarized of performance metrics employed by the studies.

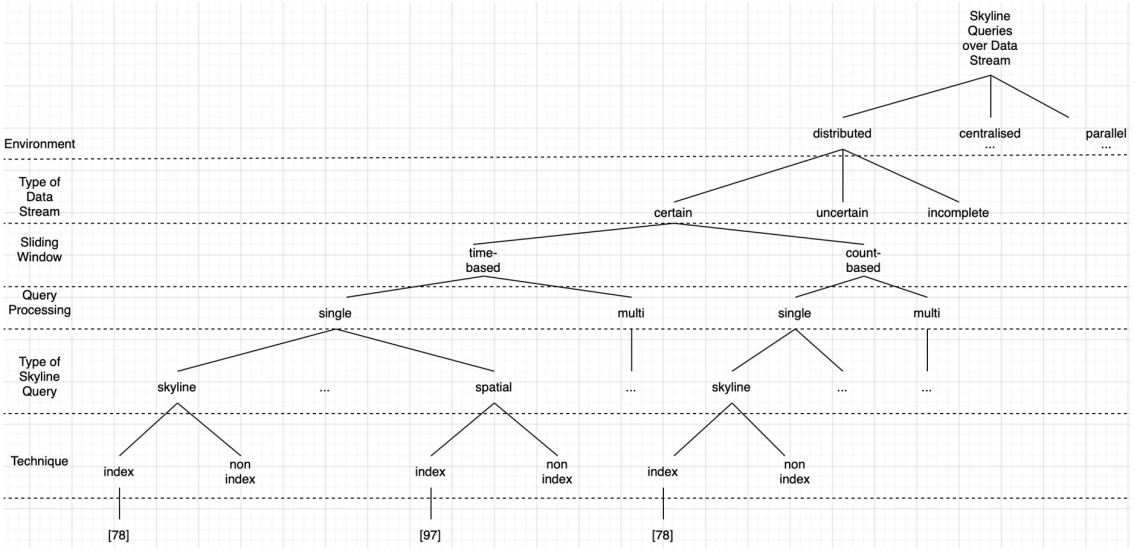
Reference	Efficiency (Processing Time)	Space Consumption		Maintenance	Communication	Effectiveness (Accuracy)
		Indexing	Results			
[75]	✓	✓	–	–	–	–
[76]	✓	✓	–	–	–	–
[77]	✓	–	✓	✓	–	–
[78]	✓	–	✓	–	✓	–
[79]	✓	–	–	–	–	–
[80]	✓	–	–	✓	–	–
[81]	✓	✓	–	–	–	–
[82]	✓	✓	✓	✓	–	–
[83]	✓	–	–	–	–	–
[84]	✓	–	–	–	✓	–
[85]	✓	–	–	–	✓	–
[86]	✓	✓	–	–	–	–
[87]	✓	–	–	✓	–	–
[88]	✓	–	✓	✓	–	–
[89]	✓	–	✓	–	–	–
[90]	✓	–	–	–	–	✓
[91]	✓	–	–	–	✓	–
[92]	✓	–	–	–	–	–
[93]	✓	–	–	–	–	–
[94]	✓	–	–	✓	–	✓
[95]	✓	–	–	–	–	–
[96]	✓	✓	–	–	–	–
[97]	✓	✓	–	✓	✓	–

Existing skyline algorithms [75], [76], [78], [79], [80], [81], [82], [84], [85], [93], [94], [96] assume that the query requirements of the users are based on the same fixed set of dimensions (all dimensions) that are available in the data set,

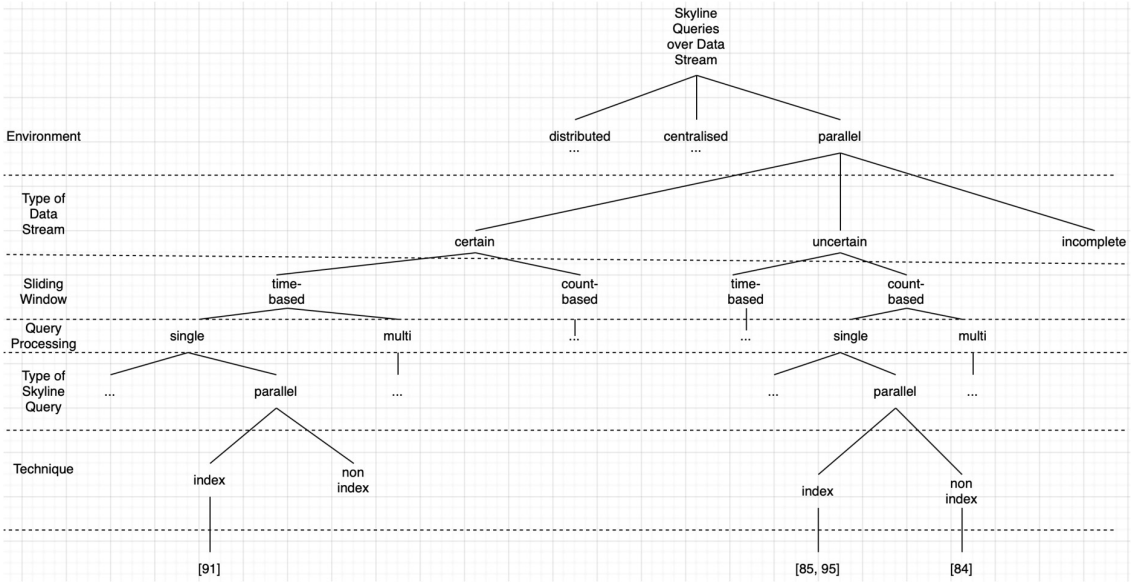
TABLE 9. Summary of studies related to skyline query processing over data stream (2000 – 2022).

Reference	Query Type	Sliding Window		Proposed Technique	Data Stream	Query Processing	Indexing Technique	Data Set		Performance Metric				
		Time-based	Count-based					Synthetic	Real	Efficiency	Space Consumption: Indexing	Space Consumption: Results	Maintenance	Communication	Effectiveness
[75]	skyline query	✓	–	*Lazy and Eager*	certain	single	*B*-tree, *R*-tree, *R**-tree, linked list	anticorrelated, independent	–	✓	✓	–	–	–	–
[76]	skyline query	✓	–	*LookOut*	certain	single	quadtree	anticorrelated, correlated, independent	–	✓	✓	–	–	–	–
[77]	subspace skyline query	✓	–	*NMA*, TSP, *ATP*, *TAP*	certain	single	grid index/array	anticorrelated, independent	–	✓	–	✓	✓	–	–
[78]	skyline query	✓	✓	*BOCS*, *GridSky*	certain	single	grid index/array	anticorrelated, independent	–	✓	–	✓	–	✓	–
[79]	skyline query	✓	–	*PSkyline*, *O-PSkyline*, *K-PSkyline*	uncertain	single	*Z*-tree, *R*-tree, *ZB*-trees	anticorrelated, independent	*NBA*	✓	–	–	–	–	–
[80]	skyline query	✓	–	*Wp-Skyline*, *CL approach*	uncertain	single	*R*-tree, hash table	anticorrelated, correlated, independent	*Long Beach county road (LBC)*	✓	–	–	✓	–	–
[81]	multiple skyline queries	✓	–	*FAST algorithm*	certain	multiple	*R**-tree, *PR* quadtree	anticorrelated, correlated, independent	–	✓	✓	–	–	–	–
[82]	skyline query	✓	–	*SSKY*, *MSKY*, *QSKY*, *TOPK*, *TIMESTAMP*	uncertain	single	*R*-tree	anticorrelated, independent	*New York Stock Exchange (NYSE)*	✓	✓	✓	✓	–	–
[83]	reverse skyline query	✓	–	*CPRS*	uncertain	single	*R*-tree	anticorrelated, correlated, independent	–	✓	–	–	–	–	–
[84]	parallel skyline query	–	✓	*SPM*, *APM*, *DPM*	uncertain	single	–	anticorrelated, independent	*Zillow*	✓	–	–	–	✓	–
[85]	parallel skyline query	–	✓	*DPF*, *PSS*	uncertain	single	grid index/array	anticorrelated, independent	*IPUMS*, *Zillow*	✓	–	–	–	✓	–
[86]	subspace skyline query	–	✓	*PSS*, *APSS*, *CPSS*	uncertain	single	grid index/array	anticorrelated, correlated, independent	*e-business*	✓	✓	–	–	–	–
[87]	probabilistic *n*-of-*N* queries	✓	–	*pnN*, *pmnN*, *pcnN*	uncertain	single	*R*-tree	anticorrelated, correlated, independent	*NYSE*	✓	–	–	✓	–	–
[88]	reverse skyline queries, *n-of-N*, (*n*1 , *n*2)-*of-N*	✓	–	*SDRS*	certain	single	*R*-tree	anticorrelated, clustered, uniform	*Stock*, *Yahoo Financial Website*	✓	–	✓	✓	–	–
[89]	skyline group query	✓	–	*skyline group algorithm*	certain	single	*R*-tree, hash table, dominance graph, matrix	General	–	✓	–	✓	–	–	–
[90]	*k* representative skyline query	–	✓	*PBA*, *greedy algorithm*	certain	single	–	anticorrelated, independent	*Stock*, *Forest environment*	✓	–	–	–	–	✓
[91]	parallel skyline query	✓	–	*eager algorithm*	certain	single	grid index/array	anticorrelated, correlated, independent	–	✓	–	–	–	✓	–
[92]	ρ-dominant skyline query, *n-of-N* ρ-dominant skyline query, (*n*1, *n*2)-*of-N*ρ-dominant skyline query	✓	–	*DSSW*	certain	single	–	independent, inverse correlation	–	✓	–	–	–	–	–
[93]	skyline query	✓	–	*klSkyline*, *slSkyline*	incomplete	single	–	anti-independent, independent, relevant	*NBA*, *Movie*	✓	–	–	–	–	–
[94]	skyline query	✓	–	*Sky-iDS*	incomplete	single	skyline tree	anticorrelated, correlated, uniform	*Intel lab*, *UCI gas sensor*, *Antallagma time series*, *Pump sensor data*	✓	–	–	✓	–	✓
[95]	parallel *n-of-N* skyline query	–	✓	*a sliding window partitioning strategy*, *a streaming items mapping strategy*, *RST*, *RBI*	uncertain	single	*RST*, *RBI*	anticorrelated, correlated, independent	*IPUMS*	✓	–	–	–	–	–
[96]	multidimensional skyline query	✓	–	*MSSD*	certain	single	*NSCt*	anticorrelated, independent	*tweets*	✓	✓	–	–	–	–
[97]	spatial keyword skyline query	✓	–	*distributed skyline query processing framework*	certain	single	*MF-Rt-tree*	–	*tweets*	✓	✓	–	✓	✓	–

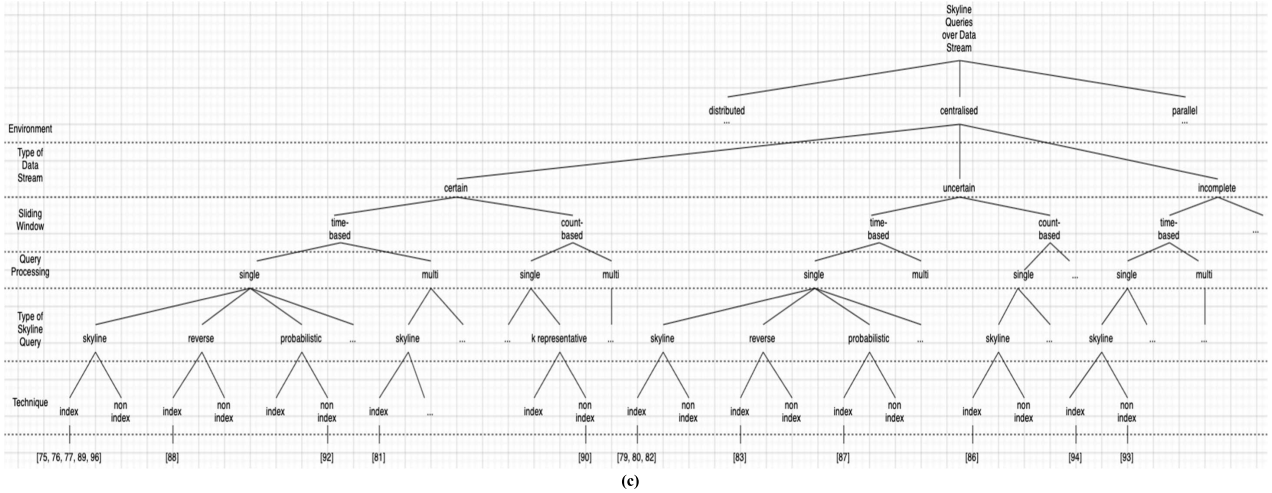
hence the full space skylines are the final outcomes of these algorithms. However, the applicability of skyline queries suffers from severe drawbacks because this rigid assumption often lead to an impractical skyline size especially with high multi-dimensional objects which no longer offer any interesting insights. Moreover, in practice different users may be interested with different dimensions of the objects. Furthermore, in some applications only certain dimensions are accessible to protect data privacy and anonymity [84]. Therefore, supporting concurrent and unpredictable subspace skyline queries over data streams has been the aim of [77], [86] to ensure users are given flexibility to select a subset of dimensions (subspace) as their query requirements. On the other hand, considering different forms of skyline retrieval like top-k skyline, k-dominate skyline, *n-of-N* skyline, and k representative skyline would also maximizing the user's preference function [75], [84], [85]. Instead of returning all skyline objects which are likely large in size, top-k skyline returns only the k most interesting skyline objects based on some kind of preference specified by the user. Inherently, the skyline computation over data stream would require less data in memory. On the other hand, to deal with dimensionality curse, one possibility is to reduce the number of dimensions considered. Nonetheless, to determine which dimensions to retain is not easy. Hence, without any intimate knowledge of the application domain, the k-dominate skyline



(a)



(b)



(c)

FIGURE 7. (a) Taxonomy of skyline queries over data stream – distributed environment. (b) Taxonomy of skyline queries over data stream – parallel environment. (c) Taxonomy of skyline queries over data stream – centralized environment.

is said to be the best option. Given an n-dimensional data set, a k-dominant skyline object is an object that is not k-dominated by any other objects where $k \leq n$. In addition, the query flexibility can be effectively enhanced by utilizing the *n-of-N* skyline operator which works by supporting the query for the most recent n ($n \leq N$) objects at the same time [95]. The k representative skyline is useful if the number of skyline objects is large. A representative skyline contains k skyline objects that can represent its corresponding full skyline. Besides, there are several works like [87] and [95] that rely on a probability threshold value in determining the skyline objects. Here, an object is a probabilistic skyline object if its skyline probability is not below a given probability threshold. Obviously, it is difficult for users to specify a meaningful threshold value. If the threshold value is set too high, then important results may be lost; while if it is too low a lot of low quality results may be returned. This means, a solution which could support several probability thresholds at the same time [95] would further enhance the flexibility of the user's query requirements.

B. ADAPTIVE SYSTEM FOR EXTREMELY FAST DATA STREAMS

A data stream is a continuous and typically rapid feed of data from variety of sources like sensors, geo-positioning devices, communication network traffic, HTTP request, etc. One of the main challenges in managing the data stream is to support real time processing. The data stream sources are often bursty, prone to dramatic spikes in volume, have limited system resources like CPU, memory, and bandwidth, with data characteristics that vary over time. Because the data arrival rates are unpredictable with peak load during a spike can be orders of magnitude higher than typical loads, the overloaded system will be unable to process all the input data and keep up with the rate of data arrival. This will cause latency in processing. Therefore, it is important for systems processing continuous monitoring queries over data streams to be adaptive to unanticipated spikes in input data rates that exceed the system capacity. In order for the system to continue to provide up-to-date query responses, load shedding technique is one of the prominent solutions for extremely fast streams. It is the process of reducing resource requirements by dropping excess load from the system when the demand on resources is above the system capacity. As reported in [75], their proposed techniques, *Lazy and Eager* algorithms, can handle very spiky traffic (up to 10^5 objects per second) and to make their solutions more meaningful in practice, they attempted to incorporate the load shedding techniques in their solutions. A different view is presented in [77], in which to accommodate the extremely fast streams, a more efficient index structures are suggested to be incorporated into their proposed grid-based maintenance algorithm, *NMA*, and pruning techniques, i.e. *TSP*, *ATP*, and *TAP*. Presently, with the arrival rate of stream objects fixed to 1,000 objects/s, the query time taken by their proposed solution does not exceed 0.01s in all cases. An efficient index structure for evaluating skylines should have the non-overlapping partitioning characteristics that leads to a natural decomposition of space that can more effectively prune the index nodes that must be searched [76]. On the other hand, developing parallel algorithms for fast data streams is one of the future studies highlighted in [93]. To realize an efficient parallel query processing, the fault-tolerant parallel techniques are crucial to ensure that the continuous skyline query is not interrupted even when failures occur during the query process [84], [85]. The interruption of the query process will not only waste plenty of the computing resources but will also seriously affect the correctness of skyline results and the query experience of the users [85].

C. DIRTY DATA SETS WITH DATA QUALITY PROBLEMS

Data sets may suffer from quality issues in particular when the data have been gathered from different data stream sources. The data may be incomplete, inaccurate, inconsistent, insecure, out-of-date, or might contain duplicates. The presence of dirty data or bad quality data in the data set would impact the computed skylines if they dominate some other objects with better quality. Currently, issues related to both uncertain data and incomplete data over data streams have been well investigated [79], [80], [82], [83], [84], [85], [86], [87], [93], [94], [95]. However, the uncertain data stream explored by these studies [79], [80], [82], [83], [84], [85], [86], [87] followed the discreate uncertainty model. There is no study that has adopted the continuous uncertainty model in computing skylines over data stream. Here, an object o_i is modelled as a continuous range of value or an approximate value, v_i, which is associated with a probability density function representing the possible values of the object (see *Definition* 4). The precise value of the object is not known during skyline computation, which can be specified in one of the following forms: range value (e.g. 100 – 200) or approximate value (eg. $\approx$100, $\pm$ 100). Therefore, developing efficient skyline algorithms on dirty data sets with data quality problems other than incompleteness and discrete uncertainty remain an open research problem [93].

VI. CONCLUSION

This study presents a systematic literature review of skyline query processing over data streams whereby 23 research articles are selected based on the 6 inclusion and 6 exclusion criteria. Seven research questions are designed and an extensive comparison is performed to address each question. This work aims at assisting interested researchers to have an up-to-date review of works related to skyline query processing over data stream by presenting a complete and systematic review of the scholarly literature over the period between 2000 and 2022. The findings demonstrate that the identified skyline techniques are motivated by the need to speed up the processing of skyline queries, primarily due to the unique characteristics of data streams that are continuously changing over time (time-sensitive), real-time, volatile, and unpredictable. Although these skyline approaches are designed

specifically for data streams and share a common goal, their solutions differ with regard to the *type of skyline query*, *type of streaming data*, *type of sliding window, query processing technique, indexing technique* as well as the *data stream environment* employed. Based on these main key aspects a comprehensive taxonomy is developed. The taxonomy revealed that most studies assumed a centralized environment with certain data stream. Furthermore, open issues and challenges as recommendations for future research direction are discussed, that include flexibility in query requirements of the users, adaptive system for extremely fast data streams, and dirty data sets with data quality problems.

ACKNOWLEDGMENT

All opinions, findings, conclusions, and recommendations in this article are those of the authors and do not necessarily reflect the views of the funding agencies.

REFERENCES

[1] S. Börzsöny, D. Kossmann, and K. Stocker, "The skyline operator," in *Proc. 17th Int. Conf. Data Eng.*, 2001, pp. 421–430.
[2] E. Tiakas, A. N. Papadopoulos, and Y. Manolopoulos, "Skyline queries: An introduction," in *Proc. 6th Int. Conf. Inf., Intell., Syst. Appl. (IISA)*, Jul. 2015, pp. 1–6.
[3] K.-L. Tan, P.-K. Eng, and B. C. Ooi, "Efficient progressive skyline computation," in *Proc. Very Large Data Bases*, 2001, pp. 301–310.
[4] D. Kossmann and F. S. Ramsak Rost, "Shooting stars in the sky: An online algorithm for skyline queries," in *Proc. Very Large Data Bases*, 2002, pp. 275–286.
[5] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with presorting," in *Proc. 19th Int. Conf. Data Eng.*, 2003, pp. 717–720.
[6] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "An optimal and progressive algorithm for skyline queries," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Jun. 2003, pp. 467–478.
[7] K. Hose, "Processing skyline queries in P2P systems," in *Proc. Very Large Data Bases, PhD Workshop*, 2005, pp. 36–40.
[8] P. Godfrey, R. Shipley, and J. Gryz, "Maximal vector computation in large data sets," in *Proc. VLDB*, vol. 5, Aug. 2005, pp. 229–240.
[9] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "Progressive skyline computation in database systems," *ACM Trans. Database Syst.*, vol. 30, no. 1, pp. 41–82, Mar. 2005.
[10] C.-Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung, and Z. Zhang, "Finding k-dominant skylines in high dimensional space," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Jun. 2006, pp. 503–514.
[11] J. Pei, Y. Yuan, X. Lin, W. Jin, M. Ester, Q. Liu, W. Wang, Y. Tao, J. X. Yu, and Q. Zhang, "Towards multidimensional subspace skyline analysis," *ACM Trans. Database Syst.*, vol. 31, no. 4, pp. 1335–1381, Dec. 2006.
[12] J. Pei, B. Jiang, X. Lin, and Y. Yuan, "Probabilistic skylines on uncertain data," in *Proc. Very Large Data Bases*, Sep. 2007, pp. 15–26.
[13] B. Joao, J. Rocha, V. Akrivi, D. Christos, and N. Kjetil, "AGiDS: A grid based strategy for distributed skyline query," in *Proc. Global Learn. Observ. Benefit Environ.*, 2009, pp. 12–23.
[14] K. C. K. Lee, W.-C. Lee, B. Zheng, H. Li, and Y. Tian, "Z-SKY: An efficient skyline query processing framework based on Z-order," *VLDB J.*, vol. 19, no. 3, pp. 333–362, Jun. 2010.
[15] M. E. Khalefa, M. F. Mokbel, and J. J. Levandoski, "Skyline query processing for uncertain data," in *Proc. 19th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2010, pp. 1293–1296.
[16] X. Li, Y. Wang, X. Li, and G. Wang, "Skyline query processing on interval uncertain data," in *Proc. IEEE 15th Int. Symp. Object/Compon./Service-Oriented Real-Time Distrib. Comput. Workshops*, Apr. 2012, pp. 87–92.
[17] N. H. M. Saad, H. Ibrahim, F. Sidi, and R. Yaakob, "Non-index based skyline analysis on high dimensional data with uncertain dimensions," in *Proc. Int. Baltic Conf. Databases Inf. Syst.* Cham, Switzerland: Springer, Jul. 2018, pp. 272–286.
[18] M. A. M. Lawal, H. Ibrahim, N. F. M. Sani, and R. Yaakob, "An indexed non-probability skyline query processing framework for uncertain data," in *Proc. Int. Conf. Adv. Mach. Learn. Technol. Appl.* Singapore: Springer, Feb. 2020, pp. 289–301.
[19] N. H. M. Saad, H. Ibrahim, F. Sidi, R. Yaakob, and A. A. Alwan, "Efficient skyline computation on uncertain dimensions," *IEEE Access*, vol. 9, pp. 96975–96994, 2021.
[20] M. A. M. Lawal, H. Ibrahim, N. F. M. Sani, and R. Yakoob, "Computing skylines over uncertain data," SSRN, Rochester, NY, USA, Tech. Rep., 2022, pp. 1–28.
[21] X. Lin, Y. Yuan, W. Wang, and H. Lu, "Stabbing the sky: Efficient skyline computation over sliding windows," in *Proc. 21st Int. Conf. Data Eng. (ICDE)*, Apr. 2005, pp. 502–513.
[22] A. Tzanakas, E. Tiakas, and Y. Manolopoulos, "Skyline algorithms on streams of multidimensional data," in *Proc. East Eur. Conf. Adv. Databases Inf. Syst.* Cham, Switzerland: Springer, Aug. 2016, pp. 63–71.
[23] P. Zou, W. Cheng, and Y. Jia, "Research on interval skyline over data stream," in *Proc. 2nd Int. Conf. Future Comput. Commun.*, vol. 3, May 2010, pp. V3-484–V3-488.
[24] L. Rudenko and M. Endres, "Real-time skyline computation on data streams," in *Proc. Eur. Conf. Adv. Databases Inf. Syst.* Cham, Switzerland: Springer, Sep. 2018, pp. 20–28.
[25] A. Wulamu, H. Li, X. Guo, Y. Xie, and Y. Fu, "Processing skyline groups on data streams," in *Proc. IEEE 12th Int. Conf. Ubiquitous Intell. Comput. IEEE 12th Int. Conf. Autonomic Trusted Comput. IEEE 15th Int. Conf. Scalable Comput. Commun. Associated Workshops (UIC-ATC-ScalCom)*, Aug. 2015, pp. 935–942.
[26] K. Alami and S. Maabout, "Multidimensional skylines over streaming data," in *Proc. Int. Conf. Database Syst. Adv. Appl.* Cham, Switzerland: Springer, Apr. 2019, pp. 338–342.
[27] L. Tian, L. Wang, A. Li, P. Zou, and Y. Jia, "Continuous skyline tracking on update data streams," in *Proc. Adv. Web Netw. Technol., Inf. Manage.* Berlin, Germany: Springer, 2007, pp. 192–197.
[28] K. Udomlamlert, T. Hara, and S. Nishio, "Candidate pruning technique for skyline computation over frequent update streams," in *Proc. Database Expert Syst. Appl.* Cham, Switzerland: Springer, Aug. 2015, pp. 93–108.
[29] J.-J. Li, S.-L. Sun, and Y.-Y. Zhu, "Efficient maintaining of skyline over probabilistic data stream," in *Proc. 4th Int. Conf. Natural Comput.*, vol. 4, Oct. 2008, pp. 378–382.
[30] L. Zhu, C. Li, and H. Chen, "Efficient computation of reverse skyline on data stream," in *Proc. Int. Joint Conf. Comput. Sci. Optim.*, vol. 1, Apr. 2009, pp. 735–739.
[31] Y. Won Lee, K. Y. Lee, and M. H. Kim, "Efficient computation of multiple sliding window skylines on data streams," in *Proc. 5th Int. Conf. Comput. Sci. Converg. Inf. Technol.*, Nov. 2010, pp. 951–956.
[32] J. Zhang, J. Gu, S. Cheng, B. Li, W. Wang, and D. Meng, "Efficient algorithms of parallel skyline join over data streams," in *Proc. Int. Conf. Algorithms Archit. Parallel Process.* Cham, Switzerland: Springer, Nov. 2018, pp. 184–199.
[33] S. Alibrandi, S. Bravo, M. Goncalves, and M. E. Vidal, "D-FOPA: A dynamic final object pruning algorithm to efficiently produce skyline points over data streams," in *Proc. Database Expert Syst. Appl.* Cham, Switzerland: Springer, Aug. 2015, pp. 117–133.
[34] J. Yang, B. Qu, C. P. Li, and H. Chen, "DC-tree: An algorithm for skyline query on data streams," in *Proc. Int. Conf. Adv. Data Mining Appl.* Berlin, Germany: Springer, Oct. 2008, pp. 644–651.
[35] H. Lu, Y. Zhou, and J. Haustad, "Continuous skyline monitoring over distributed data streams," in *Proc. Int. Conf. Sci. Stat. Database Manag.* Berlin, Germany: Springer, Jun. 2010, pp. 565–583.
[36] M. Kontaki, A. N. Papadopoulos, and Y. Manolopoulos, "Continuous processing of preference queries in data streams," in *Proc. Int. Conf. Current Trends Theory Pract. Comput. Sci.* Berlin, Germany: Springer, Jan. 2010, pp. 47–60.
[37] Y. Lu, J. Zhao, L. Chen, B. Cui, and D. Yang, "Effective skyline cardinality estimation on data streams," in *Proc. Int. Conf. Database Expert Syst. Appl.* Berlin, Germany: Springer, Sep. 2008, pp. 241–254.
[38] R. Liu and D. Li, "Dynamic dimension indexing for efficient skyline maintenance on data streams," in *Proc. Int. Conf. Database Syst. Adv. Appl.* Cham, Switzerland: Springer, Sep. 2020, pp. 272–287.
[39] M. Kontaki, A. N. Papadopoulos, and Y. Manolopoulos, "Continuous k-dominant skyline computation on multidimensional data streams," in *Proc. ACM Symp. Appl. Comput.*, Mar. 2008, pp. 956–960.
[40] G. Liang and L. Su, "Continuous adaptive mining the thin skylines over evolving data stream," in *Proc. Int. Conf. Distrib. Comput. Internet Technol.* Berlin, Germany: Springer, Dec. 2007, pp. 254–264.
[41] J.-X. Lin and J.-J. Wei, "Constrained skyline computing over data streams," in *Proc. IEEE Int. Conf. e-Bus. Eng.*, Oct. 2008, pp. 155–161.

[42] N. Sarkas, G. Das, N. Koudas, and A. K. H. Tung, "Categorical skylines for streaming data," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Jun. 2008, pp. 239–250.
[43] L. Su, P. Zou, and Y. Jia, "Adaptive mining the approximate skyline over data stream," in *Proc. Int. Conf. Comput. Sci.* Berlin, Germany: Springer, May 2007, pp. 742–745.
[44] T. D. Matteis, S. D. Girolamo, and G. Mencagli, "A multicore parallelization of continuous skyline queries on data streams," in *Proc. Eur. Conf. Parallel Process.* Berlin, Germany: Springer, Aug. 2015, pp. 402–413.
[45] L. Zhang, Y. Jia, and P. Zou, "A grid index based method for continuous constrained skyline query over data stream," in *Advances in Web and Network Technologies, and Information Management*. Berlin, Germany: Springer, 2009, pp. 185–197.
[46] H. Li and J. Yoo, "Efficient continuous skyline query processing scheme over large dynamic data sets," *ETRI J.*, vol. 38, no. 6, pp. 1197–1206, Dec. 2016.
[47] Z. Zheng, K. Ruan, M. Yu, X. Zhang, N. Wang, and D. Li, "K-dominant skyline query algorithm for dynamic datasets," *Frontiers Comput. Sci.*, vol. 15, no. 1, pp. 1–9, Feb. 2021.
[48] A. A. Safaei and M. S. Haghjoo, "Parallel processing of continuous queries over data streams," *Distrib. Parallel Databases*, vol. 28, nos. 2–3, pp. 93–118, Dec. 2010.
[49] M. Bai, J. Xin, G. Wang, X. Wang, and R. Zimmermann, "The subspace global skyline query processing over dynamic databases," *World Wide Web*, vol. 20, no. 2, pp. 291–324, Mar. 2017.
[50] G. Xiao, K. Li, X. Zhou, and K. Li, "Queueing analysis of continuous queries for uncertain data streams over sliding windows," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 30, no. 9, Nov. 2016, Art. no. 1660001.
[51] W. Zhang, X. Lin, Y. Zhang, W. Wang, and J. X. Yu, "Probabilistic skyline operator over sliding windows," in *Proc. IEEE 25th Int. Conf. Data Eng.*, Mar. 2009, pp. 1060–1071.
[52] M. Bai, J. Xin, and G. Wang, "Probabilistic reverse skyline query processing over uncertain data stream," in *Proc. Int. Conf. Database Syst. Adv. Appl.* Berlin, Germany: Springer, Apr. 2012, pp. 17–32.
[53] X. Li, Y. Wang, X. Li, Y. Wang, and R. Huang, "Parallelizing probabilistic streaming skyline operator in cloud computing environments," in *Proc. IEEE 37th Annu. Comput. Softw. Appl. Conf.*, Jul. 2013, pp. 84–89.
[54] J. Liu, X. Li, K. Ren, J. Song, and Z. Zhang, "Parallel *n*-of-*N* skyline queries over uncertain data streams," in *Proc. Int. Conf. Database Expert Syst. Appl.* Cham, Switzerland: Springer, Sep. 2018, pp. 176–184.
[55] X. Li, J. Liu, K. Ren, X. Li, X. Ren, and K. Deng, "Parallel k-dominant skyline queries over uncertain data streams with capability index," in *Proc. IEEE 21st Int. Conf. High Perform. Comput. Commun., IEEE 17th Int. Conf. Smart City, IEEE 5th Int. Conf. Data Sci. Syst. (HPCC/SmartCity/DSS)*, Aug. 2019, pp. 1556–1563.
[56] W. Hanning, X. Weixiang, J. Yang, L. Wei, and J. Chaolong, "Efficient processing of continuous skyline query over smarter traffic data stream for cloud computing," *Discrete Dyn. Nature Soc.*, vol. 2013, pp. 1–10, Jan. 2013.
[57] H. Z. Su, E. T. Wang, and A. L. Chen, "Continuous probabilistic skyline queries over uncertain data streams," in *Proc. Int. Conf. Database Expert Syst. Appl.* Berlin, Germany: Springer, Aug. 2010, pp. 105–121.
[58] Z. Dzolkhifli, H. Ibrahim, F. Sidi, L. S. Affendey, S. N. M. Rum, and A. A. Alwan, "A skyline query processing approach over interval uncertain data stream with K-means clustering technique," in *Proc. 11th Int. Conf. Adv. Databases, Knowl. Data Appl. (DBKDA)*, 2019, pp. 51–56.
[59] C.-M. Liu and S.-W. Tang, "An effective probabilistic skyline query process on uncertain data streams," *Proc. Comput. Sci.*, vol. 63, pp. 40–47, Jan. 2015.
[60] Z. Ma, K. Zhang, S. Wang, and C. Yu, "A double-index-based k-dominant skyline algorithm for incomplete data stream," in *Proc. IEEE 4th Int. Conf. Softw. Eng. Service Sci.*, May 2013, pp. 750–753.
[61] X. Li, K. Ren, X. Li, and J. Yu, "Efficient skyline computation over distributed interval data," *Concurrency Comput., Pract. Exp.*, vol. 29, no. 10, May 2017, Art. no. e4075.
[62] F. Mohamed, R. M. Ismail, N. L. Badr, and M. F. Tolba, "Data streams processing techniques," in *Multimedia Forensics and Security*. Cham, Switzerland: Springer, 2017, pp. 279–305.
[63] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, "Models and issues in data stream systems," in *Proc. 21st ACM SIGMOD-SIGACT-SIGART Symp. Princ. Database Syst.*, Jun. 2002, pp. 1–16.
[64] A. D. Giudice, "Counting stars: A survey on flexible skyline query approaches," 2022, *arXiv:2203.00331*.
[65] V. Fabris, "Flexible skylines, regret minimization and skyline ranking: A comparison to know how to select the right approach," 2022, *arXiv:2201.10179*.
[66] Z. Dzolkhifli, H. Ibrahim, and M. H. B. M. Hassin, "Review on skyline query processing techniques over data stream," in *Proc. Int. Conf. Softw. Eng. Comput. Syst. 4th Int. Conf. Comput. Sci. Inf. Manage. (ICSECS-ICOCSIM)*, Aug. 2021, pp. 443–446.
[67] P. K. Sadineni, "Comparative study on skyline query processing techniques on big data," in *Proc. 4th Int. Conf. I-SMAC (IoT Social, Mobile, Analytics Cloud) (I-SMAC)*, Oct. 2020, pp. 1045–1050.
[68] H. Gothwal, J. Choudhary, and D. P. Singh, "The survey on skyline query processing for data-specific applications," in *Proc. 3rd Int. Conf. Internet Things Connected Technol. (ICIoTCT)*. Jaipur, India: Malaviya National Institute of Technology, Apr. 2018, pp. 26–27.
[69] C. Kalyvas and T. Tzouramanis, "A survey of skyline query processing," 2017, *arXiv:1704.01788*.
[70] M. A. M. Lawal, H. Ibrahim, S. F. Mohd, and R. Yaakob, "Skyline computation of uncertain database: A survey," in *Proc. 6th Int. Conf. Comput. Informat. (ICOCI)*. Changlun, Malaysia: UUM Institutional Repository, 2017, pp. 1–7.
[71] A. C. Bency and S. D. Kanmani, "A survey of skyline processing in various environment," *Indian J. Comput. Sci. Eng.*, vol. 5, no. 1, pp. 5–8, Mar. 2014.
[72] K. Hose and A. Vlachou, "A survey of skyline processing in highly distributed environments," *VLDB J.*, vol. 21, no. 3, pp. 359–384, Jun. 2012.
[73] M. R. Kulkarni, B. F. Momin, and B. F. Momin, "Future research directions in skyline computation," *Int. J. Comput. Eng. Sci.*, vol. 2, no. 5, pp. 1–11, May 2012.
[74] M. J. Page et al., "The PRISMA 2020 statement: An updated guideline for reporting systematic reviews," *Syst. Rev.*, vol. 10, no. 1, pp. 1–11, Dec. 2021.
[75] Y. Tao and D. Papadias, "Maintaining sliding window skylines on data streams," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 3, pp. 377–391, Mar. 2006.
[76] M. Morse, J. M. Patel, and W. I. Grosky, "Efficient continuous skyline computation," *Inf. Sci.*, vol. 177, no. 17, pp. 3411–3437, Sep. 2007.
[77] Z. Huang, S. Sun, and W. Wang, "Efficient mining of skyline objects in subspaces over data streams," *Knowl. Inf. Syst.*, vol. 22, no. 2, pp. 159–183, Feb. 2010.
[78] S. Sun, Z. Huang, H. Zhong, D. Dai, H. Liu, and J. Li, "Efficient monitoring of skyline queries over distributed data streams," *Knowl. Inf. Syst.*, vol. 25, no. 3, pp. 575–606, Dec. 2010.
[79] D. Kim, H. Im, and S. Park, "Computing exact skyline probabilities for uncertain databases," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 12, pp. 2113–2126, Dec. 2012.
[80] X. Ding, X. Lian, L. Chen, and H. Jin, "Continuous monitoring of skylines over uncertain data streams," *Inf. Sci.*, vol. 184, no. 1, pp. 196–214, Feb. 2012.
[81] Y. W. Lee, K. Y. Lee, and M. H. Kim, "Efficient processing of multiple continuous skyline queries over a data stream," *Inf. Sci.*, vol. 221, pp. 316–337, Feb. 2013.
[82] W. Zhang, X. Lin, Y. Zhang, W. Wang, G. Zhu, and J. X. Yu, "Probabilistic skyline operator over sliding windows," *Inf. Syst.*, vol. 38, no. 8, pp. 1212–1233, Nov. 2013.
[83] Y. T. Yang, Y. J. Wang, M. Guo, and X. Y. Li, "Continuous probabilistic reverse skyline monitoring over uncertain data streams," in *Applied Mechanics and Materials*, vol. 380. Zürich, Switzerland: Trans Tech Publications, 2013, pp. 2681–2686.
[84] X. Li, Y. Wang, X. Li, and Y. Wang, "Parallel skyline queries over uncertain data streams in cloud computing environments," *Int. J. Web Grid Services*, vol. 10, no. 1, pp. 24–53, 2014.
[85] X. Li, Y. Wang, X. Li, and Y. Wang, "Parallelizing skyline queries over uncertain data streams with sliding window partitioning and grid index," *Knowl. Inf. Syst.*, vol. 41, no. 2, pp. 277–309, Nov. 2014.
[86] L. Zhao, Y.-Y. Yang, and X. Zhou, "Continuous probabilistic subspace skyline query processing using grid projections," *J. Comput. Sci. Technol.*, vol. 29, no. 2, pp. 332–344, Mar. 2014.
[87] W. Zhang, A. Li, M. A. Cheema, Y. Zhang, and L. Chang, "Probabilistic *n*-of-*N* skyline computation over uncertain data streams," *World Wide Web*, vol. 18, no. 5, pp. 1331–1350, Sep. 2015.
[88] J. Xin, Z. Wang, M. Bai, and G. Wang, "Reverse skyline computation over sliding windows," *Math. Problems Eng.*, vol. 2015, pp. 1–19, Jan. 2015.
[89] X. Guo, H. Li, A. Wulamu, Y. Xie, and Y. Fu, "Efficient processing of skyline group queries over a data stream," *Tsinghua Sci. Technol.*, vol. 21, no. 1, pp. 29–39, Feb. 2016.

[90] M. Bai, J. Xin, G. Wang, L. Zhang, R. Zimmermann, Y. Yuan, and X. Wu, "Discovering the k representative skyline over a sliding window," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 8, pp. 2041–2056, Aug. 2016.
[91] T. De Matteis, S. Di Girolamo, and G. Mencagli, "Continuous skyline queries on multicore architectures," *Concurrency Comput., Pract. Exp.*, vol. 28, no. 12, pp. 3503–3522, Aug. 2016.
[92] Z. Wang, J. Xin, L. Ding, J. Ba, and X. Gao, "ρ-dominant skyline computation on data stream," *IEEE Access*, vol. 6, pp. 53201–53213, 2018.
[93] H. Wang, S. Yin, M. Sun, Y. Wang, H. Wang, J. Li, and H. Gao, "Efficient computation of skyline queries on incomplete dynamic data," *IEEE Access*, vol. 6, pp. 52741–52753, 2018.
[94] W. Ren, X. Lian, and K. Ghazinour, "Skyline queries over incomplete data streams," *VLDB J.*, vol. 28, no. 6, pp. 961–985, Dec. 2019.
[95] J. Liu, X. Li, K. Ren, and J. Song, "Parallelizing uncertain skyline computation against n-of-N data streaming model," *Concurrency Comput., Pract. Exp.*, vol. 31, no. 4, Feb. 2019, Art. no. e4848.
[96] K. Alami and S. Maabout, "A framework for multidimensional skyline queries over streaming data," *Data Knowl. Eng.*, vol. 127, May 2020, Art. no. 101792.
[97] Z. Deng, Y. Wang, T. Liu, S. Dustdar, R. Ranjan, A. Zomaya, Y. Liu, and L. Wang, "Spatial-keyword skyline publish/subscribe query processing over distributed sliding window streaming data," *IEEE Trans. Comput.*, vol. 71, no. 10, pp. 2659–2674, Oct. 2022.

MUDATHIR AHMED MOHAMUD received the B.Sc. degree in information technology from the Faculty of Computing, SIMAD University, Mogadishu, Somalia, in 2015, and the M.Sc. degree in computer science from Universiti Putra Malaysia (UPM), in 2021, where he is currently pursuing the Ph.D. degree in database systems with the Department of Computer Science. He is also a Teaching Assistant and a Graduate Research Assistant with the Department of Computer Science, UPM. His research interests include query optimization, preference query evaluation, data science, skyline queries, database integration, data stream, machine learning, and information systems.

HAMIDAH IBRAHIM (Member, IEEE) received the Ph.D. degree in computer science from the University of Wales, Cardiff, U.K., in 1998. She is currently a Full Professor with the Faculty of Computer Science and Information Technology, Universiti Putra Malaysia (UPM). Her current research interests include databases (distributed, parallel, mobile, biomedical, and XML) focusing on issues related to integrity maintenance/checking, ontology/schema/data integration, ontology/schema/data mapping, cache management, access control, data security, transaction processing, query optimization, query reformulation, preference evaluation–context-aware, information extraction, concurrency control, and data management in mobile, grid, and cloud.

FATIMAH SIDI (Member, IEEE) received the Ph.D. degree in management information systems from Universiti Putra Malaysia (UPM), Malaysia, in 2008. She is currently an Associate Professor in computer science with the Department of Computer Science, Faculty of Computer Science and Information Technology, UPM. Her current research interests include knowledge and information management systems, data and knowledge engineering, database, and data warehouse.

SITI NURULAIN MOHD RUM received the Diploma degree from Universiti Teknologi MARA (UiTM), the bachelor's degree from Universiti Teknologi Malaysia (UTM), and the master's and Ph.D. degrees from the University of Malaya (UM). She is currently a Lecturer with the Department of Computer Science and Information Technology, Universiti Putra Malaysia (UPM). Before joining academia, she spent more than 15 years as an IT professional, involved in a number of IT projects, including software development, database and data center migration, virtualization infrastructure development, and procurement of hardware that includes servers and storage for the data center. She has published a number of journal articles and presented research papers at international conferences. She is also actively involved in doing research and consulting in the IT field. Her research interests include database processing, artificial intelligence, social media analytics, and data sciences.

ZARINA BINTI DZOLKHIFLI received the bachelor's degree in computer science and the M.Sc. degree in database system from Universiti Putra Malaysia. She is currently a Lecturer with the Department of Computer and Networking, Faculty of Computing, Universiti Malaysia Pahang. Her research interests include query processing, data streaming, preference query, and data networking.

ZHANG XIAOWEI received the bachelor's and master's degrees in computer science and technology from the Zhengzhou University of Light Industry, China, in 2006 and 2009, respectively. He is currently pursuing the Ph.D. degree with Universiti Putra Malaysia (UPM), Malaysia. His research interests include database systems and data mining.

MA'ARUF MOHAMMED LAWAL received the bachelor's and master's degrees in computer science from Ahmadu Bello University, Zaria, Nigeria, and the Ph.D. degree from the Faculty of Computer Science and Information Technology, Universiti Putra Malaysia. He is currently a Senior Lecturer with the Department of Computer Science, Faculty of Physical Sciences, Ahmadu Bello University. His research interests include query optimization, data science, preference evaluation-context-aware, information extraction, concurrency control, database integration, data security, transaction processing, query reformulation, preference query evaluation, and knowledge-based representation.