NO.

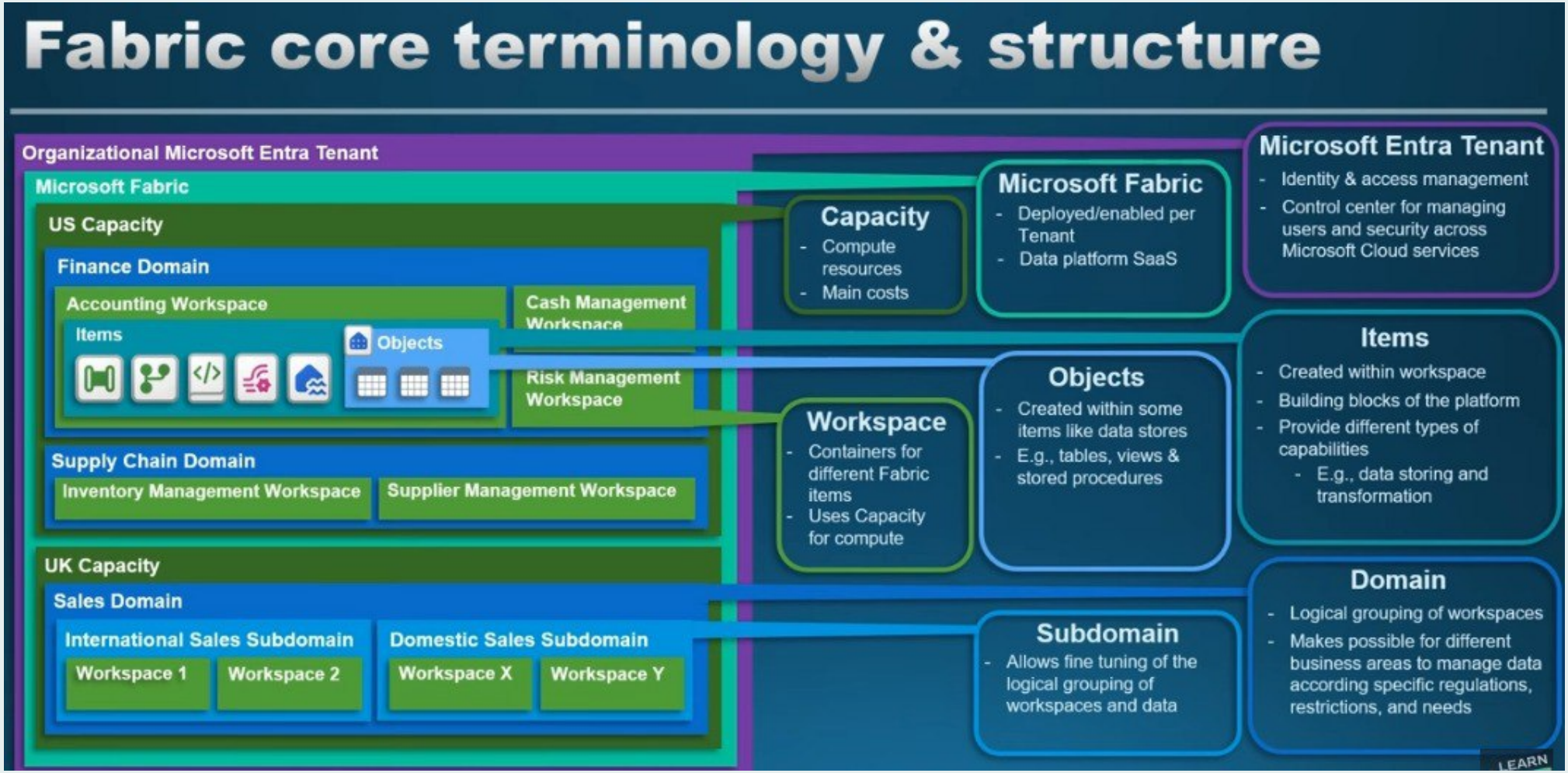
Title

Enjoy the most efficient handwriting experience! Taking notes on the go, whether for inspiration, ideas, knowledge learning, business insights, or even sketches...

NOTE IN

Creative notes

By reading we enrich the mind;
by writing we polish it.



Microsoft tenant

- This represent The organization
- Handles Identity & access management
- Controls user Permissions and security across all microsoft Cloud services including Fabric

Microsoft fabric

- A data platform (Saas) that runs within The Entra Tenant
- Enabled on a per-tenant basis
- Provides all The Combined analytics workloads (Lakehouse, Data Pipeli)

Capacity

- Represent Compute resource That microsoft fabric uses
- Consumed when running workloads (quering, Pipelines, notebooks)
- represent The main cost unit in Fabric

Domain

- A logical grouping of workspaces based on bussiness area
- Aligns with data & analytics with how the bussiness is structured
- Helps enforce governance, compliance, and responsibities

Subdomain → more fine-grained logical grouping inside a domain.

Workspace

- The main container where items and data live
- Used to organize datasets, pipelines, notebooks, models, reports, ..etc
- Used Capacity to run workload

Items

- building blocks of microsoft fabric
- created inside a workspace
- Each has capabilities like

storing data
- Lakehouse
- warehouse

Processing data
- notebook
- Data pipeline

Visualizing data
- PowerBI
- semantic model

objects

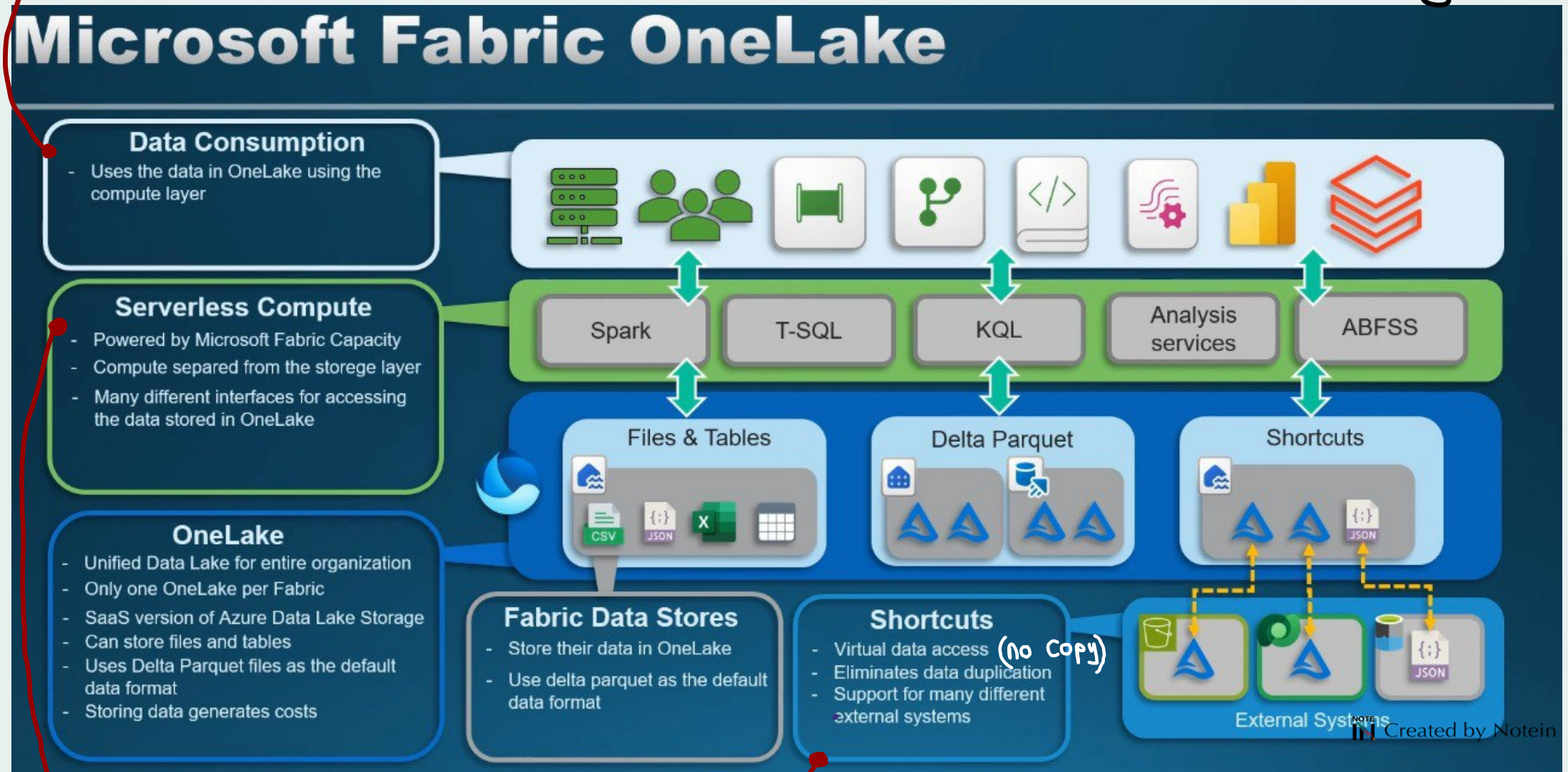
- created inside certain items like lakehouse / warehouse
- are sub-components of an item

Ex: tables, view, folders } → represent the internal structure of a data storage item.

(Top layer)

represent all The ways users & developers consume and use The data stored in OneLake

Purpose → interact with oneLake data Through compute engine used by These ways



- Compute engine layer That Process data Stored in OneLake.

- Compute separated from engine

Meaning → data stays in OneLake but you use different (interface) engines to analyze it

Create a pointer to external data Instead of copying into OneLake

Programming Languages in DP-700

SQL

- Structured Query Language used with structured data
- Two different dialects in the exam: Spark SQL and T-SQL
- Spark SQL is mainly used in Spark notebooks and interacting with Lakehouse
- T-SQL is mainly used with Warehouse, T-SQL notebooks, T-SQL endpoints and can be also used with Eventhouse/KQL databases

KQL

- Kusto Query Language
- Read only request language meant for fetching, filtering, analyzing, and visualizing data
- Capable of handling large volumes of structured, semi-structured and unstructured data
- Mainly used to query KQL databases in Eventhouse
- The easiest DP-700 language to learn for a beginner

PySpark

- PySpark is the Python API for Apache Spark
- Very versatile language that can be used for many different things and many types of data
- The most difficult language to master of the DP-700 programming languages
- Mainly used with Spark notebooks and interacting with Lakehouse

Syntax Difference Example 1

SQL

```
SELECT name, age
FROM person;
```

KQL

```
person
| project name, age
```

PySpark

```
display(person.select("name", "age"))
```

Input table/df: person

name	age	gender
John	41	Male
Jane	37	Female
Aleksi	31	Male

Query Output

name	age
John	41
Jane	37
Aleksi	31

Syntax Difference Example 2

SQL

```
SELECT count(*) AS count
FROM person;
```

KQL

```
person
| count
```

PySpark

```
display(person \
.agg({"*":"count"}) \
.withColumnRenamed("count(1)", "count"))
```

Input table/df: person

name	age	gender
John	41	Male
Jane	37	Female
Aleksi	31	Male

Query Output

count
3

Syntax Difference Example 3

SQL

```
SELECT name, age
FROM person
WHERE name = 'Aleksi';
```

KQL

```
person
| where name == "Aleksi"
| project name, age
```

PySpark

```
display(person \
.filter(person["name"] == "Aleksi") \
.select("name", "age"))
```

Input table/df: person

name	age	gender
John	41	Male
Jane	37	Female
Aleksi	31	Male

Query Output

name	age
Aleksi	31

Syntax Difference Example 4

SQL

```
SELECT name, age
FROM person
WHERE age >= 37
ORDER BY age ASC;
```

KQL

```
person
| where age >= 37
| project name, age
| order by age asc
```

PySpark

```
display(person \
.filter(person["age"] >= 37) \
.select("name", "age") \
.orderBy("age", ascending=True))
```

Input table/df: person

name	age	gender
John	41	Male
Jane	37	Female
Aleksi	31	Male

Query Output

name	age	gender
Jane	37	Female
John	41	Male

Syntax Difference Example 5

SQL

```
SELECT p.name, p.age, pi.info
FROM person AS p
JOIN person_info AS pi
ON p.name = pi.name;
```

KQL

```
person
| join person_info on name
| project name, age, info
```

PySpark

```
display(person \
.join(person_info, person["name"] == person_info["name"]) \
.select(person["name"], person["age"], person_info["info"]))
```

Input table/df: person

name	age	gender
John	41	Male
Jane	37	Female
Aleksi	31	Male

Input table/df: person_info

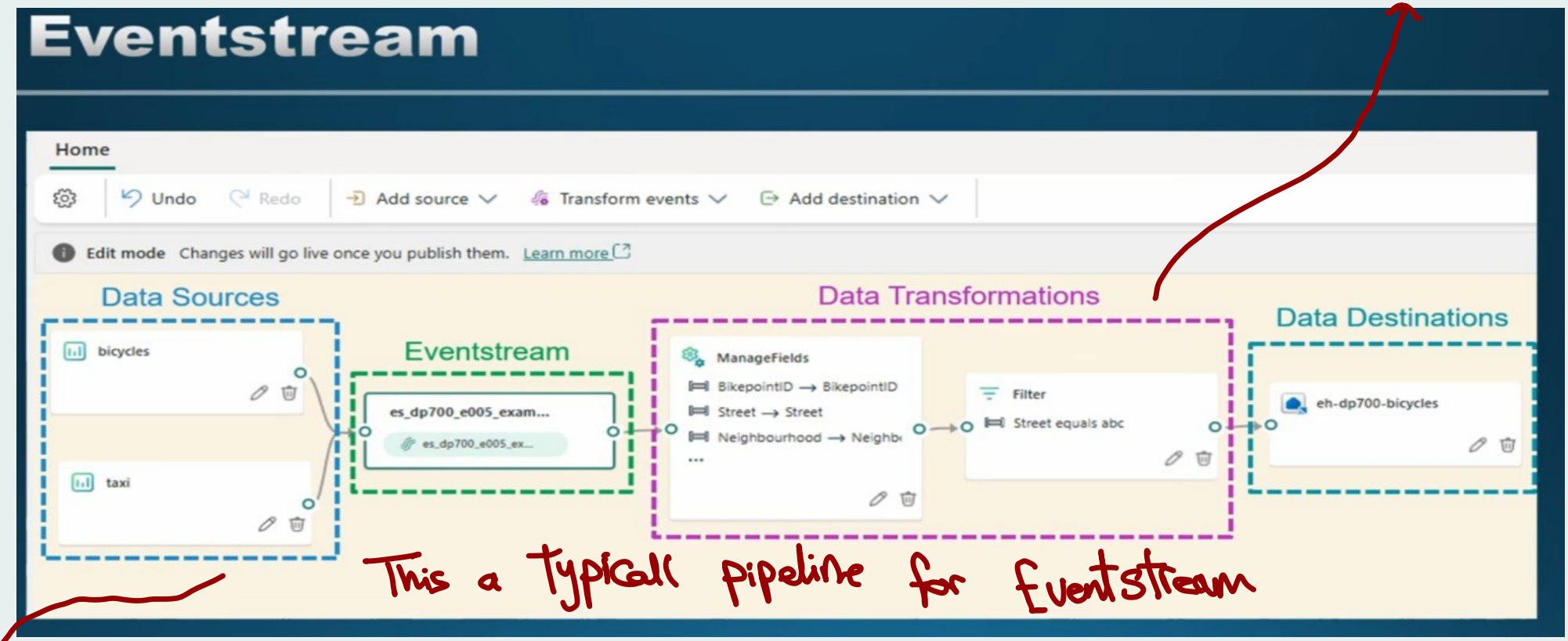
name	info
John	Sales
Jane	Management
Aleksi	IT

Query Output

name	age	info
John	41	Sales
Jane	37	Management
Aleksi	31	IT

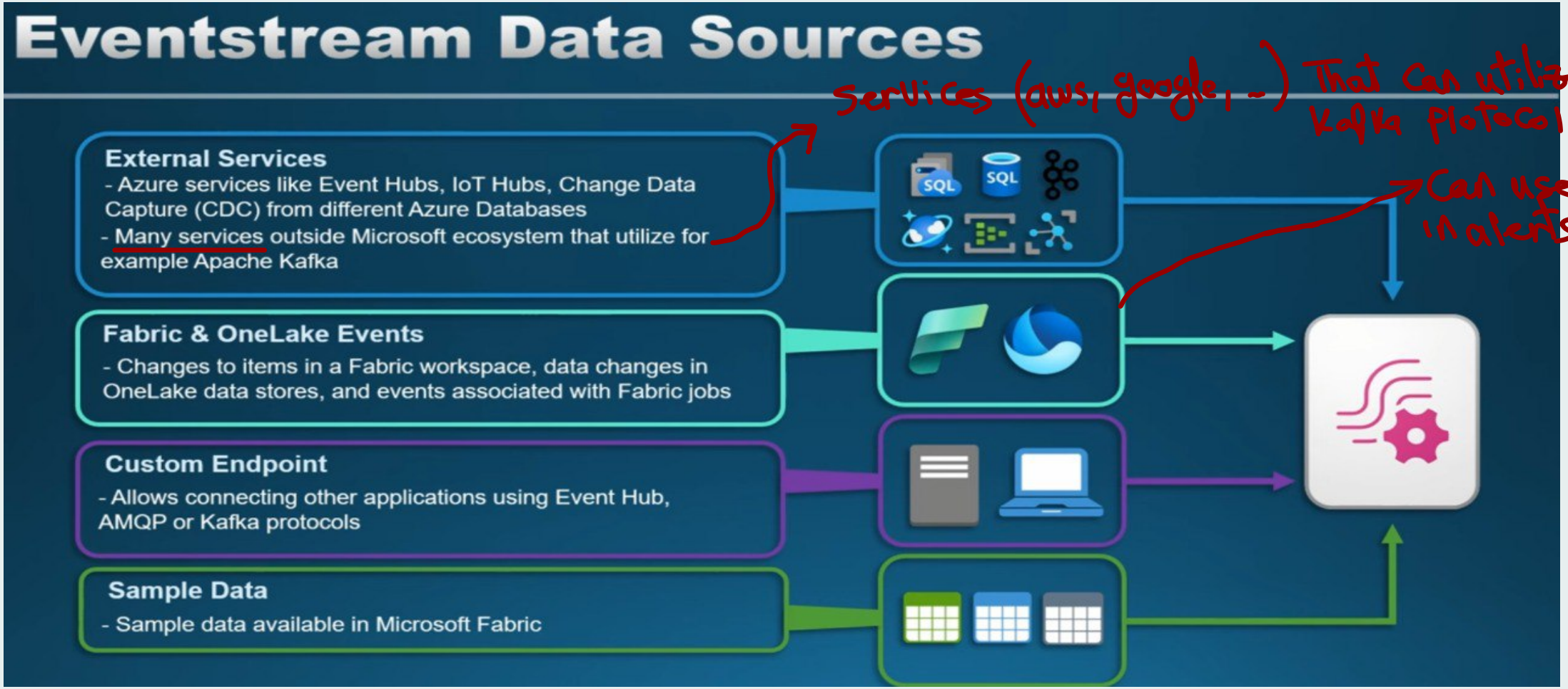
Eventstream

– we can apply transformatio
on output of eventstream



Eventstream can has multiple sources

Eventstream Data Sources



Eventstream Data Transformations

Filter: Allows filtering events based on the value of a column in the input. Depending on the data type (number or text), the transformation keeps the values that match the selected condition, such as is null or is not null.

Manage fields: Allows adding, removing, renaming and transforming columns in the data.

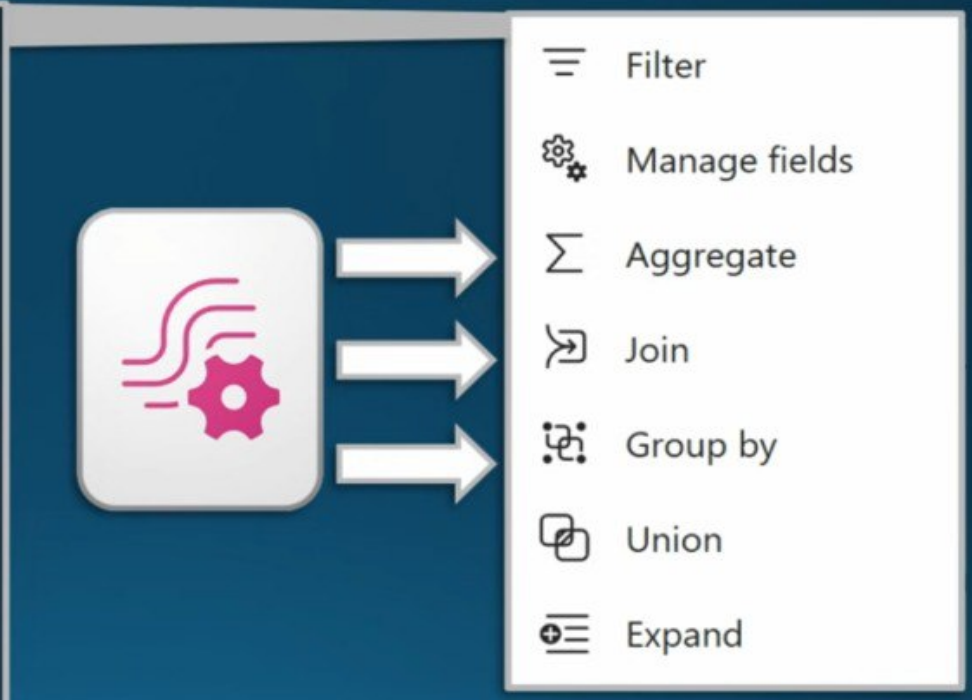
Aggregate: Allows aggregating data using aggregations sum, minimum, maximum or average over a period of time. Also, allows renaming calculated columns and partitioning aggregations based on other columns in the data.

Join: Allows combining data from two streams based on a matching condition between them.

Group by: Allows more complex aggregations using time windows (windowing functions). Also, allows renaming calculated columns and grouping aggregations based on other columns in the data.

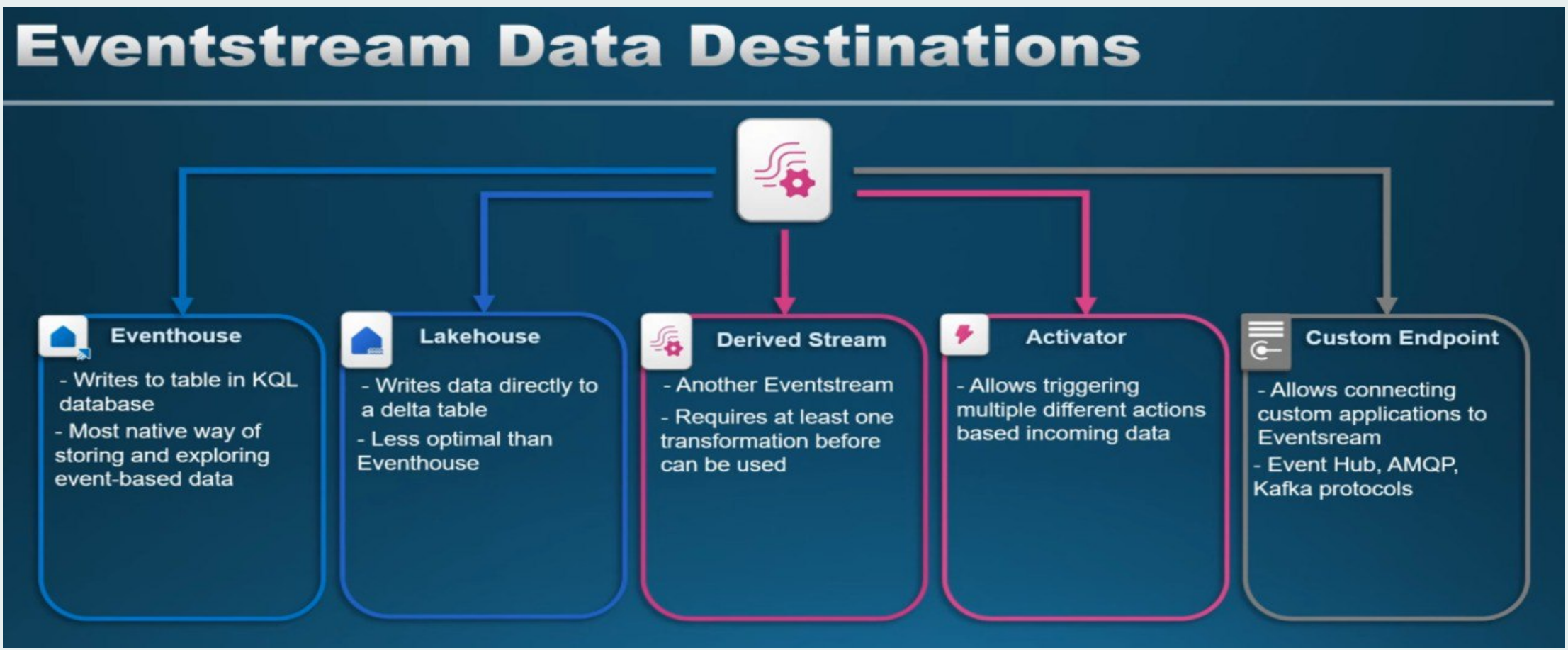
Union: Allows combining two or more inputs with shared columns that have the same name and data type. Columns that don't match are dropped and not included in the output.

Expand: Allows creating a new row for each value within an array.



- Manage fields, Aggregate, and Group by ⟶ allows renaming of columns.

Eventstream Data Destinations



What are Windowing Functions?

- Windowing functions are used in real-time stream processing to group events into time-based segments for analysis.
- They help in processing continuous, unbounded data by applying aggregations like count, sum, average, max, min, etc.
- Instead of treating the stream as an infinite flow, windowing functions divide the data into manageable chunks.
- Used in Microsoft Fabric Eventstream Group by transformation

- Tumbling Window
- Hopping Window
- Sliding Window
- Session Window
- Snapshot

Tumbling Window

Fixed, Non-Overlapping Time Intervals

- Events are grouped into fixed-size time windows (e.g., every 5 minutes).
- Each event belongs to exactly one window—no overlap.

Triggers at Regular Intervals

- A new window starts immediately after the previous one ends.
- If using a 5-minute tumbling window, the windows will be:
  - 12:00 - 12:05
  - 12:05 - 12:10
  - 12:10 - 12:15, and so on.

Best for Periodic Aggregations

- Useful for counting, summing, or averaging data over fixed time periods.

No Overlapping Windows

- Tumbling Window does not create overlapping time periods.
- Each event is processed only once in its respective time block.

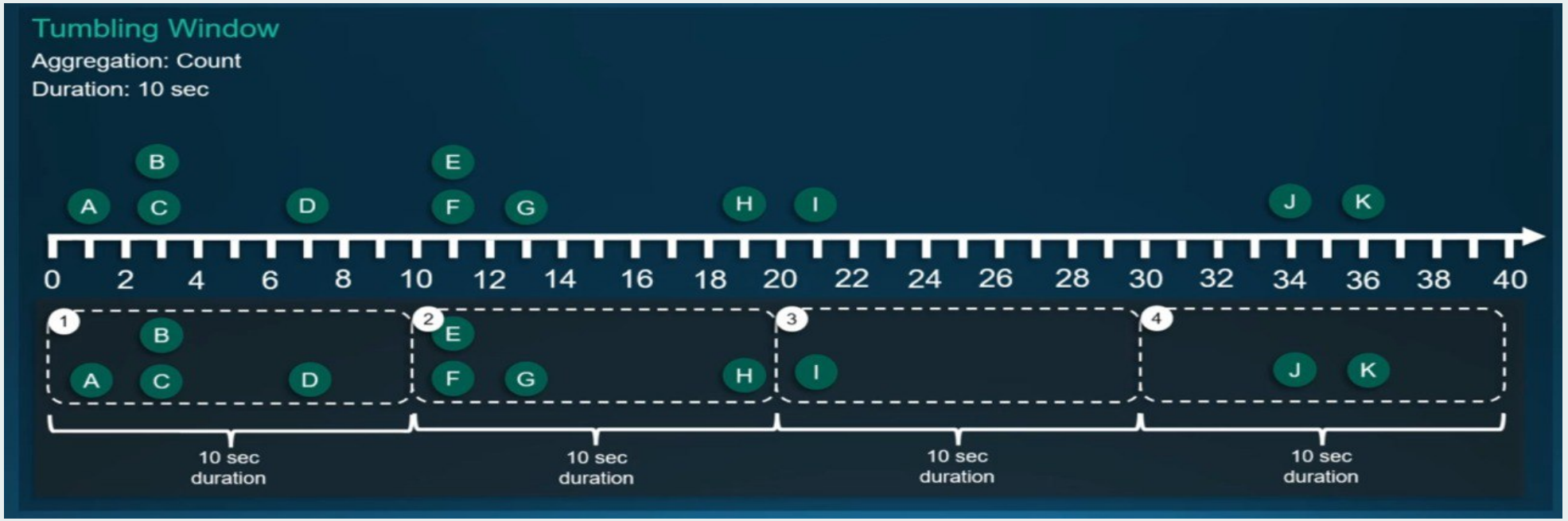
Example Questions

How many orders were placed every 10 minutes?

What is the total revenue generated every hour?

What is the average temperature recorded every 30 minutes?

How many failed login attempts occurred every 15 minutes?



Timestamp	Message
2025-01-01T00:00:01	A
2025-01-01T00:00:03	B
2025-01-01T00:00:03	C
2025-01-01T00:00:07	D
2025-01-01T00:00:11	E
2025-01-01T00:00:11	F
2025-01-01T00:00:13	G
2025-01-01T00:00:19	H
2025-01-01T00:00:21	I
2025-01-01T00:00:34	J
2025-01-01T00:00:36	K

Tumbling Window
Aggregation: Count
Duration: 10 sec

Window_End_Time	Count
2025-01-01T00:00:10	4
2025-01-01T00:00:20	4
2025-01-01T00:00:30	1
2025-01-01T00:00:40	2

Hopping Window

🔹 Fixed, Overlapping Time Intervals
- Events are grouped into fixed-size time windows (e.g., every 5 minutes).
- Windows overlap, so an event can belong to multiple windows.

🔹 Triggers at Regular Intervals
- A new window starts at a defined hop size (e.g., every 2 minutes).
- If using a 5-minute window with a 2-minute hop, the windows will be:
  - 12:00 - 12:05
  - 12:02 - 12:07
  - 12:04 - 12:09, and so on.

🔹 Best for Continuous Monitoring
- Useful for applications that need frequent updates on recent data.

🔹 Overlapping Windows for Smoother Trends
- Each event may be counted multiple times in different windows.
- Helps in smoother trend analysis by avoiding abrupt changes between time periods.

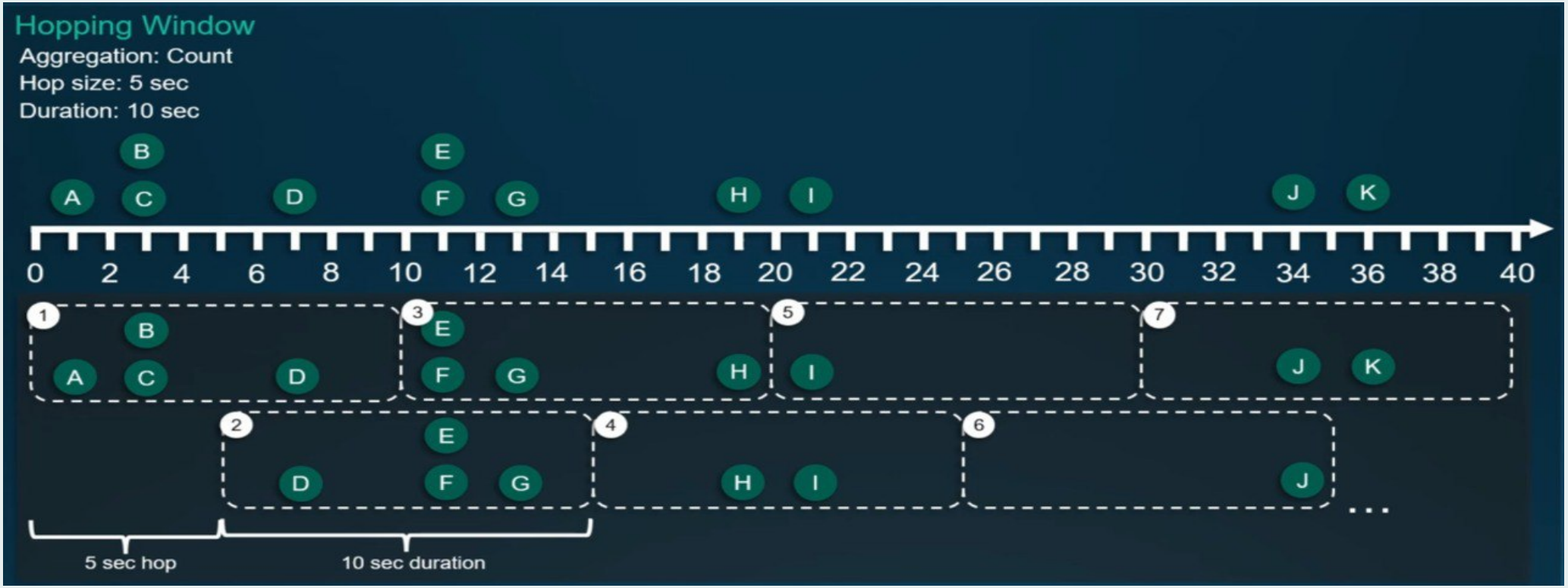
Example Questions

How many website visits occurred every 5 minutes, updating every 1 minute?

What is the moving average of CPU usage every 10 minutes, updated every 2 minutes?

What is the total revenue generated in the last hour, refreshed every 10 minutes?

What is the average temperature in the last 20 minutes, updating every 5 minutes?



Timestamp	Message
2025-01-01T00:00:01	A
2025-01-01T00:00:03	B
2025-01-01T00:00:03	C
2025-01-01T00:00:07	D
2025-01-01T00:00:11	E
2025-01-01T00:00:11	F
2025-01-01T00:00:13	G
2025-01-01T00:00:19	H
2025-01-01T00:00:21	I
2025-01-01T00:00:34	J
2025-01-01T00:00:36	K

Hopping Window
Aggregation: Count
Hop size: 5 sec
Duration: 10 sec

Window_End_Time	Count
2025-01-01T00:00:10	4
2025-01-01T00:00:15	4
2025-01-01T00:00:20	4
2025-01-01T00:00:25	2
2025-01-01T00:00:30	1
2025-01-01T00:00:35	1
2025-01-01T00:00:40	2

Sliding Window

◆ Dynamically Triggered by Incoming Events
- A new window opens only when an event arrives.
- The window looks back and forward in time for a defined duration (e.g., 5 minutes).

◆ Each Event Triggers a New Window
- Unlike Tumbling or Hopping Windows, Sliding Windows do not start at fixed intervals.
- Each event creates a new time window, covering events within the specified duration.

◆ Best for Continuous, Real-Time Calculations
- Ideal for calculating moving averages, trend analysis, and anomaly detection.

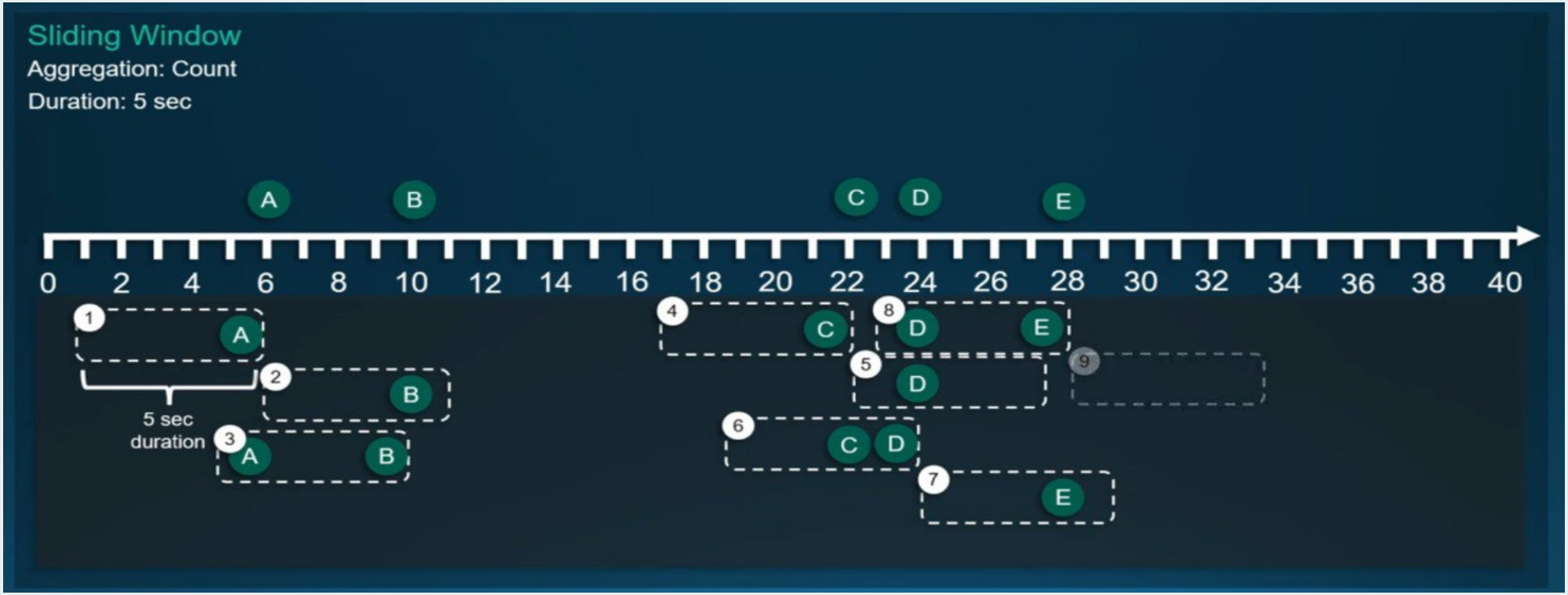
◆ Overlapping Windows for Smooth Analysis
- Windows overlap naturally when events arrive close together.
- Allows more granular real-time insights compared to fixed interval-based windows.

Example Questions

What is the moving average of stock prices in the last 10 minutes, recalculated whenever a new trade occurs?

What is the average temperature recorded in the last 20 minutes, updated every time a new sensor reading arrives?

What is the total energy consumption over the last hour, updated every time a new meter reading is received?



Timestamp	Message
2025-01-01T00:00:06	A
2025-01-01T00:00:10	B
2025-01-01T00:00:22	C
2025-01-01T00:00:24	D
2025-01-01T00:00:28	E

Sliding Window
Aggregation: Count
Duration: 5 sec

	Window_End_Time	Count
1	2025-01-01T00:00:06	1
2	2025-01-01T00:00:11	1
3	2025-01-01T00:00:10	2
4	2025-01-01T00:00:22	1
5	2025-01-01T00:00:27	1
6	2025-01-01T00:00:24	2
7	2025-01-01T00:00:29	1
8	2025-01-01T00:00:28	2

Session Window

◆ Dynamically Sized, Based on Inactivity Gaps
- Windows are not fixed; instead, they are based on periods of continuous activity.
- A session ends when there is no new event for a specified time (e.g., 10 minutes of inactivity).

◆ Windows Start and End Dynamically
- A new session starts when an event arrives after inactivity.
- The window grows as long as new events keep arriving within the inactivity and max duration threshold.

◆ Best for Analyzing User Behavior and Sessions
- Ideal for tracking user activity, customer interactions, or machine uptime.
- Example: Group website visits into sessions based on user inactivity (e.g., 30 minutes of no activity = new session).

◆ Overlapping Sessions Can Occur for Different Partitions
- Sessions for different users or devices may overlap in real-time.
- Example: User A and User B can have active sessions at the same time.

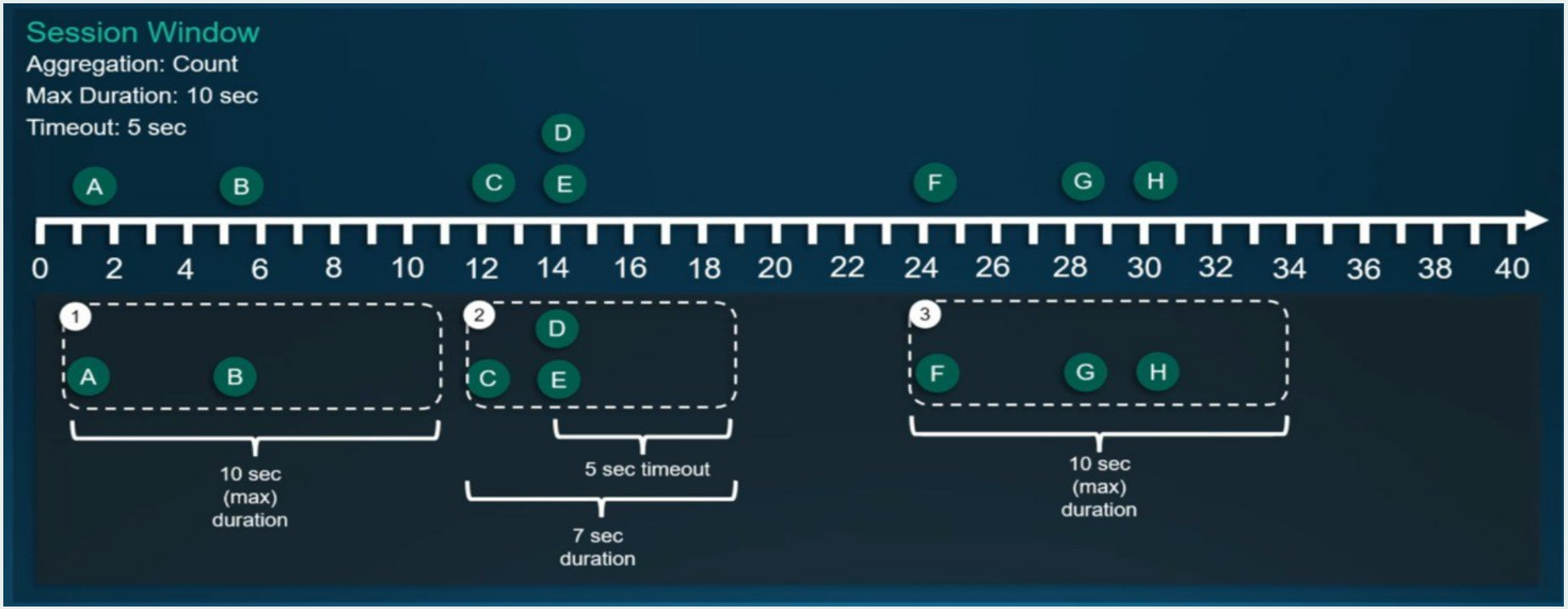
Example Questions

How many website visits were part of the same browsing session?

How long does a machine remain active before downtime?

What is the average duration of streaming sessions before a user disconnects?

How many transactions were completed within each user session?



Timestamp	Message
2025-01-01T00:00:01	A
2025-01-01T00:00:05	B
2025-01-01T00:00:12	C
2025-01-01T00:00:14	D
2025-01-01T00:00:14	E
2025-01-01T00:00:24	F
2025-01-01T00:00:28	G
2025-01-01T00:00:30	H

Session Window
Aggregation: Count
Max Duration: 10 sec
Timeout: 5 sec

Window_End_Time	Count
2025-01-01T00:00:11	2
2025-01-01T00:00:19	3
2025-01-01T00:00:34	3

Snapshot

◆ Processes Events That Share the Exact Timestamp
- A Snapshot Window groups together events that arrive at the same timestamp.
- No time duration is defined—it is purely based on simultaneous event occurrences.

◆ Triggered When Events Arrive
- A new window is created only when events share an exact arrival time.
- Example: If three events arrive at 12:00:00, they will be grouped into the same snapshot window.

◆ Best for Analyzing Simultaneous Events
- Useful for scenarios where you need to process only events that happen at the same instant.
- Example: Detecting multiple orders placed at the exact same timestamp.

◆ No Overlapping Windows
- Each event belongs to exactly one snapshot window.
- If events arrive at different timestamps, they are processed in separate windows.

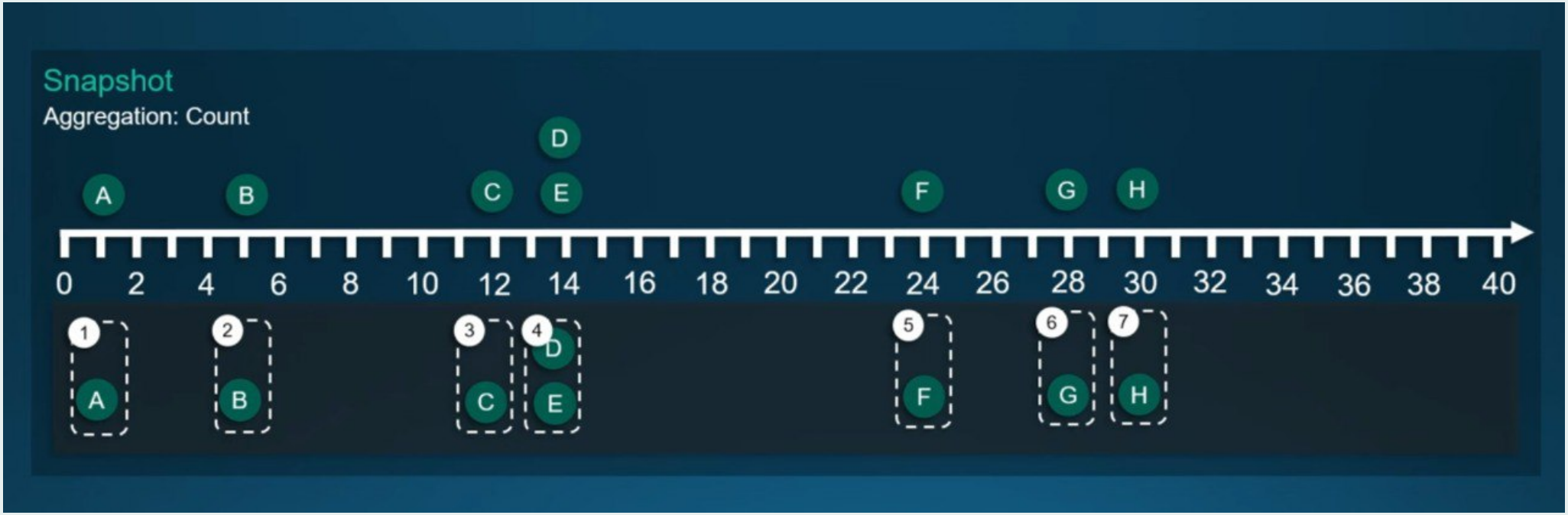
Example Questions

Which orders were placed at the exact same time?

How many error logs were generated at the exact same moment?

What is the total value of orders placed at the exact same timestamp?

What is the average temperature recorded across multiple sensors at the same time?



Timestamp	Message
2025-01-01T00:00:01	A
2025-01-01T00:00:05	B
2025-01-01T00:00:12	C
2025-01-01T00:00:14	D
2025-01-01T00:00:14	E
2025-01-01T00:00:24	F
2025-01-01T00:00:28	G
2025-01-01T00:00:30	H

Snapshot

Window_End_Time	Count
2025-01-01T00:00:01	1
2025-01-01T00:00:05	1
2025-01-01T00:00:12	1
2025-01-01T00:00:14	2
2025-01-01T00:00:24	1
2025-01-01T00:00:28	1
2025-01-01T00:00:30	1

Key Differences

Window Type	Triggered On Fixed Time Intervals?	When Does It Trigger?	Can Windows Overlap?	Fixed Window Size	Best Use Case	Example Scenario
Tumbling	✅ Yes	At regular intervals (e.g., every 5 min)	❌ No	✅ Yes	Periodic aggregations like counting, summing, or averaging over fixed time periods	Count orders every 10 minutes
Hopping	✅ Yes	At regular intervals with overlap (e.g., every 2 min with a 5-min window)	✅ Yes	✅ Yes	Continuous monitoring with frequent updates	Calculate moving average of CPU usage every 10 min, updated every 2 min
Sliding	❌ No	When an event arrives	✅ Yes	✅ Yes	Real-time trend analysis, moving averages	Average temperature over the last 5 minutes, recalculated with each new reading
Session	❌ No	When an event arrives	✅ Yes, when partitioning is being used	❌ No	Analyzing behavior or grouping activity into sessions	Group user interactions into sessions based on 15-min inactivity gaps
Snapshot	❌ No	When an event arrives	❌ No	✅ Yes	Analyzing simultaneous events	Count error logs that were generated at the exact same moment