100 XP

Introduction

1 minute

Today, massive amounts of real-time data are generated by connected applications, Internet of Things (IoT) devices and sensors, and various other sources. The proliferation of streaming data sources has made the ability to consume and make informed decisions from these data in near-real-time an operational necessity for many organizations.

Some typical examples of streaming data workloads include:

- Online stores analyzing real-time clickstream data to provide product recommendations to consumers as they browse the website.
- Manufacturing facilities using telemetry data from IoT sensors to remotely monitor high-value assets.
- Credit card transactions from point-of-sale systems being scrutinized in real-time to detect and prevent potentially fraudulent activities.

Azure Stream Analytics provides a cloud-based stream processing engine that you can use to filter, aggregate, and otherwise process a real-time stream of data from various sources. The results of this processing can then be used to trigger automated activity by a service or application, generate real-time visualizations, or integrate streaming data into an enterprise analytics solution.

In this module, you'll learn how to get started with Azure Stream Analytics, and use it to process a stream of event data.

Next unit: Understand data streams

Continue

100 XP

Understand data streams

5 minutes

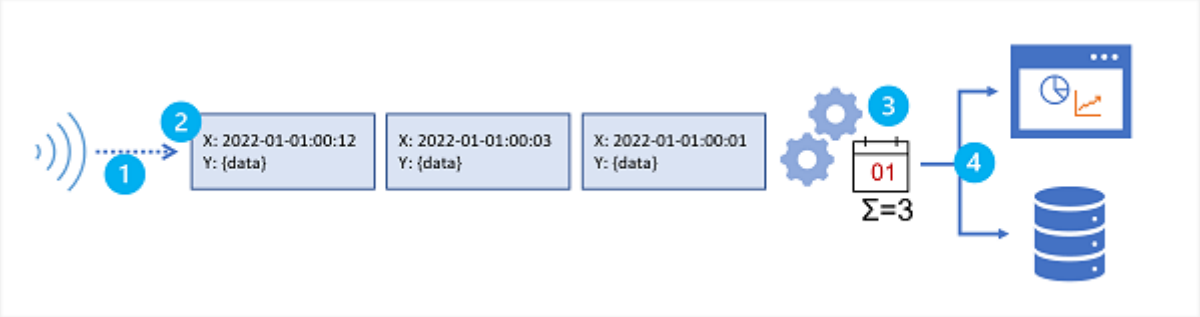
A data stream consists of a perpetual series of data, typically related to specific point-in-time events. For example, a stream of data might contain details of messages submitted to a social media micro-blogging site, or a series of environmental measurements recorded by an internet-connected weather sensor. Streaming data analytics is most often used to better understand change over time. For example, a marketing organization may perform sentiment analysis on social media messages to see if an advertising campaign results in more positive comments about the company or its products, or an agricultural business might monitor trends in temperature and rainfall to optimize irrigation and crop harvesting.

Common goals for stream analytics include

- Continuously analyzing data to report issues or trends.
- Understanding component or system behavior under various conditions to help plan future enhancements.
- Triggering specific actions or alerts when certain events occur or thresholds are exceeded.

Characteristics of stream processing solutions

Stream processing solutions typically exhibit the following characteristics:



1. The source data stream is *unbounded* - data is added to the stream perpetually.
2. Each data record in the stream includes *temporal* (time-based) data indicating when the event to which the record relates occurred (or was recorded).
3. Aggregation of streaming data is performed over temporal *windows* - for example, recording the number of social media posts per minute or the average rainfall per hour.
4. The results of streaming data processing can be used to support real-time (or *near* real-time) automation or visualization, or persisted in an analytical store to be combined with

other data for historical analysis. Many solutions combine these approaches to support both real-time and historical analytics.

Next unit: Understand event processing

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

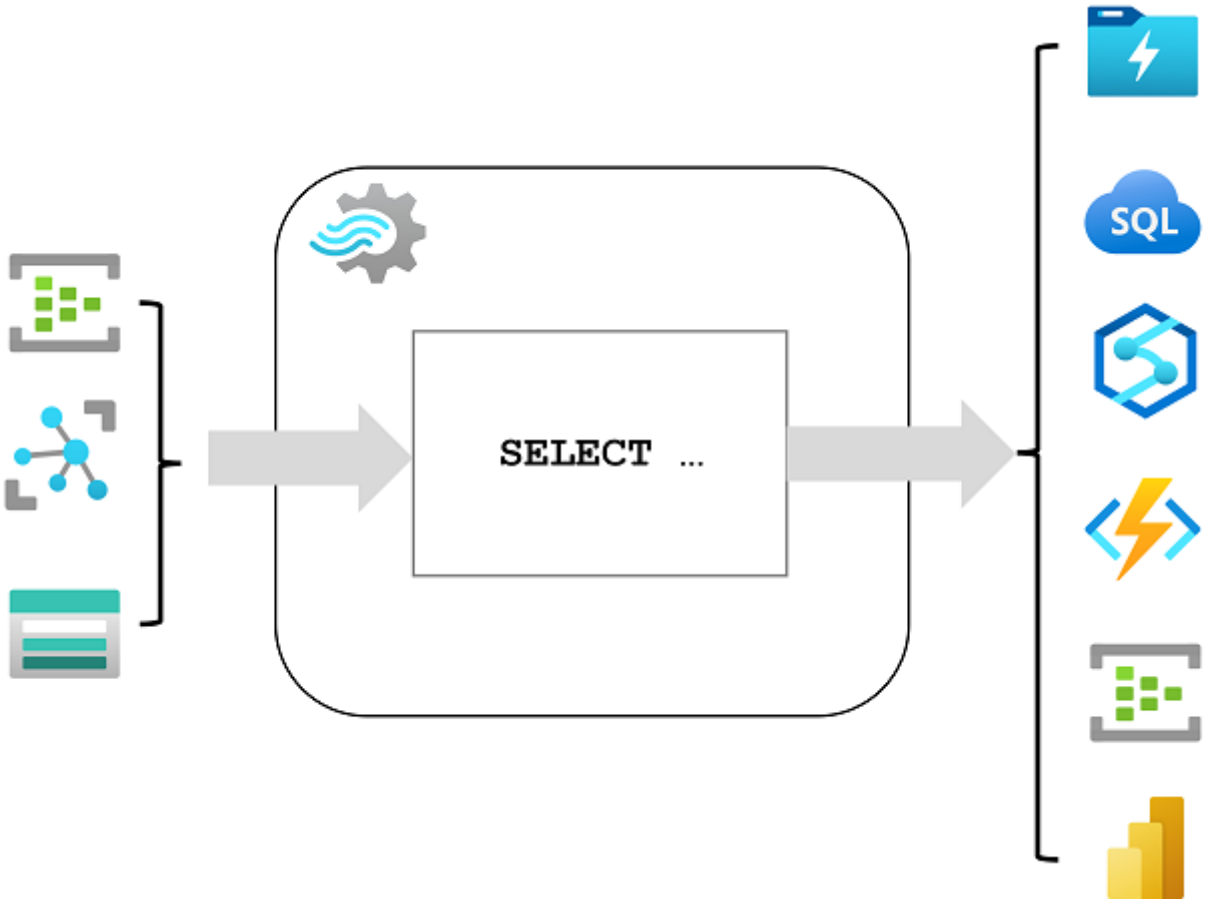
100 XP

Understand event processing

5 minutes

Azure Stream Analytics is a service for complex event processing and analysis of streaming data. Stream Analytics is used to:

- Ingest data from an *input*, such as an Azure event hub, Azure IoT Hub, or Azure Storage blob container.
- Process the data by using a *query* to select, project, and aggregate data values.
- Write the results to an *output*, such as Azure Data Lake Gen 2, Azure SQL Database, Azure Synapse Analytics, Azure Functions, Azure event hub, Microsoft Power BI, or others.



Once started, a Stream Analytics query will run perpetually, processing new data as it arrives in the input and storing results in the output.

Stream Analytics guarantees *exactly once* event processing and *at-least-once* event delivery, so events are never lost. It has built-in recovery capabilities in case the delivery of an event fails. Also, Stream Analytics provides built-in checkpointing to maintain the state of your job and produces repeatable results. Because Azure Stream Analytics is a platform-as-a-service (PaaS) solution, it's fully managed and highly reliable. Its built-in integration with various sources and

destinations and provides a flexible programmability model. The Stream Analytics engine enables in-memory compute, so it offers high performance.

Azure Stream Analytics jobs and clusters

The easiest way to use Azure Stream Analytics is to create a Stream Analytics *job* in an Azure subscription, configure its input(s) and output(s), and define the query that the job will use to process the data. The query is expressed using structured query language (SQL) syntax, and can incorporate static reference data from multiple data sources to supply lookup values that can be combined with the streaming data ingested from an input.

If your stream process requirements are complex or resource-intensive, you can create a Stream Analysis *cluster*, which uses the same underlying processing engine as a Stream Analytics job, but in a dedicated tenant (so your processing is not affected by other customers) and with configurable scalability that enables you to define the right balance of throughput and cost for your specific scenario.

Inputs

Azure Stream Analytics can ingest data from the following kinds of input:

- Azure Event Hubs
- Azure IoT Hub
- Azure Blob storage
- Azure Data Lake Storage Gen2

Inputs are generally used to reference a source of streaming data, which is processed as new event records are added. Additionally, you can define *reference* inputs that are used to ingest static data to augment the real-time event stream data. For example, you could ingest a stream of real-time weather observation data that includes a unique ID for each weather station, and augment that data with a static reference input that matches the weather station ID to a more meaningful name.

Outputs

Outputs are destinations to which the results of stream processing are sent. Azure Stream Analytics supports a wide range of outputs, which can be used to:

- Persist the results of stream processing for further analysis; for example by loading them into a data lake or data warehouse.

- Display a real-time visualization of the data stream; for example by appending data to a dataset in Microsoft Power BI.
- Generate filtered or summarized events for downstream processing; for example by writing the results of stream processing to an event hub.

Queries

The stream processing logic is encapsulated in a query. Queries are defined using SQL statements that *SELECT* data fields *FROM* one or more inputs, filter or aggregate the data, and write the results *INTO* an output. For example, the following query filters the events from the **weather-events** input to include only data from events with a **temperature** value less than 0, and writes the results to the **cold-temps** output:

SQL

```
SELECT observation_time, weather_station, temperature
INTO cold-temps
FROM weather-events TIMESTAMP BY observation_time
WHERE temperature < 0
```

A field named **EventProcessedUtcTime** is automatically created to define the time when the event is processed by your Azure Stream Analytics query. You can use this field to determine the timestamp of the event, or you can explicitly specify another DateTime field by using the *TIMESTAMP BY* clause, as shown in this example. Depending on the input from which the streaming data is read, one or more potential timestamp fields may be created automatically; for example, when using an *Event Hubs* input, a field named **EventQueuedUtcTime** is generated to record the time when the event was received in the event hub queue.

The field used as a timestamp is important when aggregating data over temporal windows, which is discussed next.

Next unit: Understand window functions

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

100 XP

Understand window functions

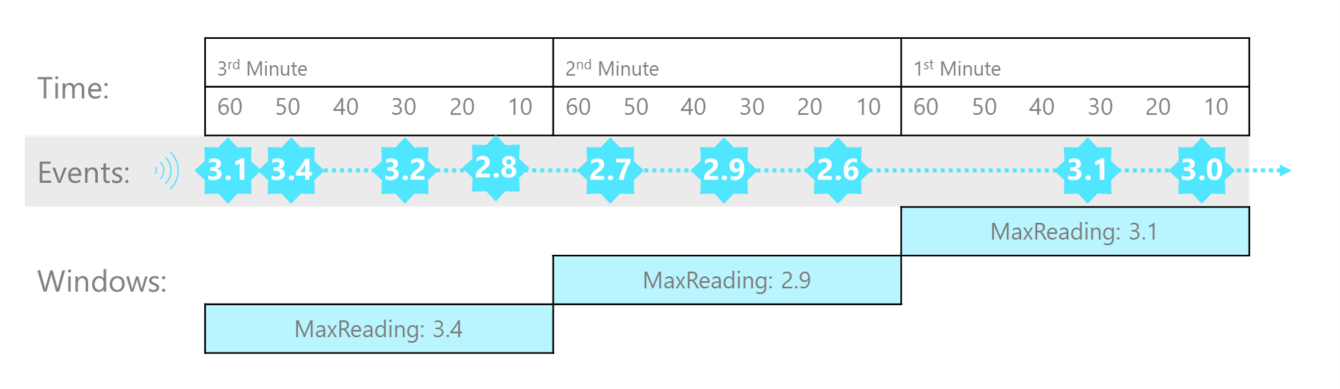
6 minutes

A common goal of stream processing is to aggregate events into temporal intervals, or *windows*. For example, to count the number of social media posts per minute or to calculate the average rainfall per hour.

Azure Stream Analytics includes native support for five kinds of temporal windowing functions. These functions enable you to define temporal intervals into which data is aggregated in a query. The supported windowing functions are Tumbling, Hopping, Sliding, Session, and Snapshot.

Tumbling

Tumbling window functions segment a data stream into a contiguous series of fixed-size, non-overlapping time segments and operate against them. Events can't belong to more than one tumbling window.



The Tumbling window example, represented by the following query, finds the maximum reading value in each one-minute window. Windowing functions are applied in Stream Analytics jobs using the GROUP BY clause of the query syntax. The `GROUP BY` clause in the following query contains the `TumblingWindow()` function, which specifies a one-minute window size.

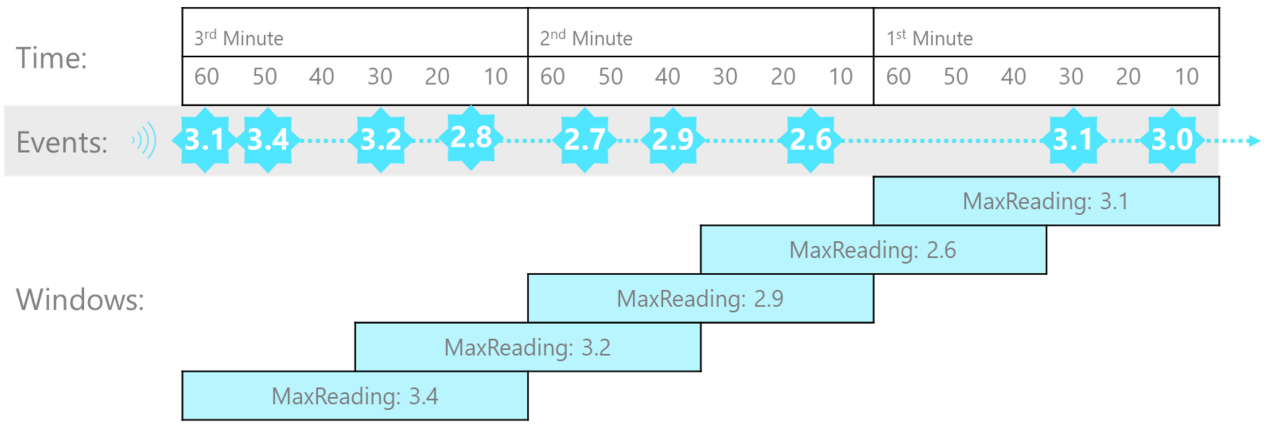
SQL

```
SELECT DateAdd(minute,-1,System.TimeStamp) AS WindowStart,
       System.TimeStamp() AS WindowEnd,
       MAX(Reading) AS MaxReading
INTO
```

```
    [output]
FROM
    [input] TIMESTAMP BY EventProcessedUtcTime
GROUP BY TumblingWindow(minute, 1)
```

Hopping

Hopping window functions model scheduled overlapping windows, jumping forward in time by a fixed period. It's easiest to think of them as Tumbling windows that can overlap and be emitted more frequently than the window size. In fact, tumbling windows are simply a hopping window whose `hop` is equal to its `size`. When you use Hopping windows, events can belong to more than one window result set.



To create a hopping window, you must specify three parameters. The first parameter indicates the time unit, such as second, minute, or hour. The following parameter sets the window size, which designates how long each window lasts. The final required parameter is the hop size, which specifies how much each window moves forward relative to the previous one. An optional fourth parameter denoting the offset size may also be used.

The following query demonstrates using a `HoppingWindow()` where the `timeunit` is set to `second`. The `windowsize` is 60 seconds, and the `hopsize` is 30 seconds. This query outputs an event every 30 seconds containing the maximum reading value that occurred over the last 60 seconds.

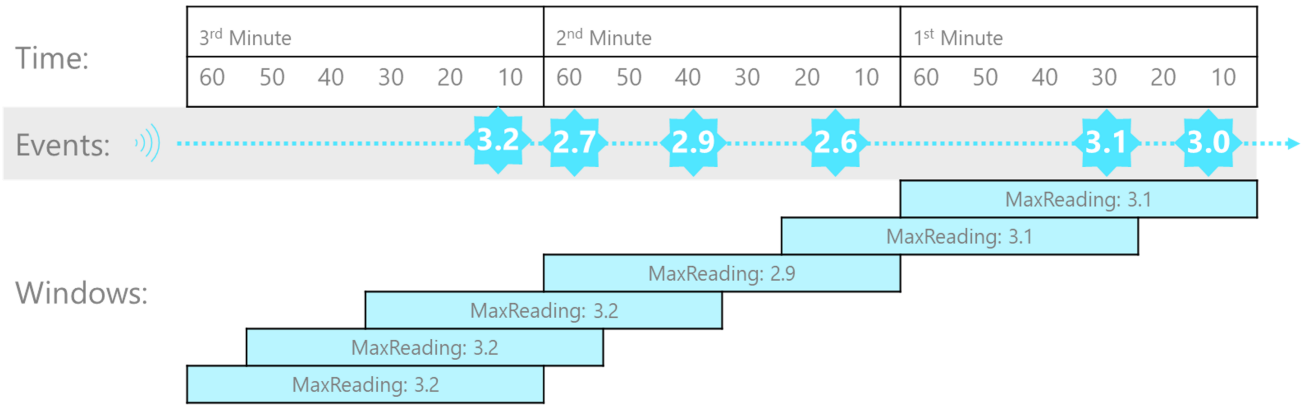
SQL

```
SELECT DateAdd(second,-60,System.TimeStamp) AS WindowStart,
       System.TimeStamp() AS WindowEnd,
       MAX(Reading) AS MaxReading
INTO
    [output]
FROM
    [input] TIMESTAMP BY EventProcessedUtcTime
```

```
GROUP BY HoppingWindow(second, 60, 30)
```

Sliding

Sliding windows generate events for points in time when the content of the window actually changes. This function model limits the number of windows that need to be considered. Azure Stream Analytics outputs events for only those points in time when an event entered or exited the window. As such, every window contains a minimum of one event. Events in Sliding windows can belong to more than one sliding window, similar to Hopping windows.



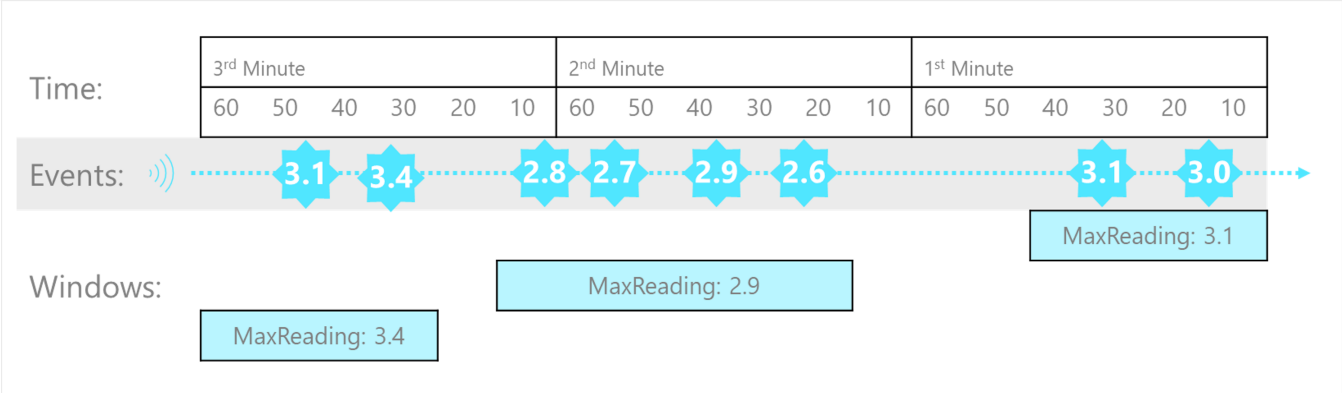
The following query uses the `SlidingWindow()` function to find the maximum reading value in each one-minute window in which an event occurred.

SQL

```
SELECT DateAdd(minute,-1,System.TimeStamp) AS WindowStart,
       System.TimeStamp() AS WindowEnd,
       MAX(Reading) AS MaxReading
INTO
    [output]
FROM
    [input] TIMESTAMP BY EventProcessedUtcTime
GROUP BY SlidingWindow(minute, 1)
```

Session

Session window functions cluster together events that arrive at similar times, filtering out periods of time where there's no data. It has three primary parameters: timeout, maximum duration, and partitioning key (optional).



The occurrence of the first event starts a session window. Suppose another event occurs within the specified timeout from the last ingested event. In that case, the window will be extended to incorporate the new event. However, if no other events occur within the specified timeout period, the window will be closed at the timeout. If events keep happening within the specified timeout, the session window will extend until the maximum duration is reached.

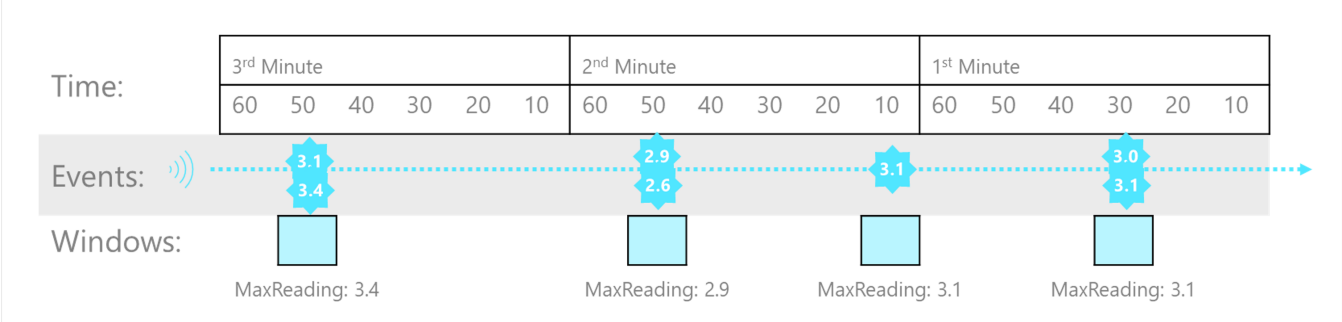
The following query measures user session length by creating a `SessionWindow` over clickstream data with a `timeoutsize` of 20 seconds and a `maximumdurationsize` of 60 seconds.

SQL

```
SELECT DateAdd(second,-60,System.TimeStamp) AS WindowStart,
       System.TimeStamp() AS WindowEnd,
       MAX(Reading) AS MaxReading
INTO
    [output]
FROM
    [input] TIMESTAMP BY EventProcessedUtcTime
GROUP BY SessionWindow(second, 20, 60)
```

Snapshot

Snapshot windows groups events by identical timestamp values. Unlike other windowing types, a specific window function isn't required. You can employ a snapshot window by specifying the `System.Timestamp()` function to your query's `GROUP BY` clause.



For example, the following query finds the maximum reading value for events that occur at precisely the same time.

SQL

```
SELECT System.TimeStamp() AS WindowTime,
       MAX(Reading) AS MaxReading
INTO
    [output]
FROM
    [input] TIMESTAMP BY EventProcessedUtcTime
GROUP BY System.Timestamp()
```

`System.Timestamp()` is considered in the `GROUP BY` clause as a snapshot window definition because it groups events into a window based on the equality of timestamps.

Next unit: Exercise - Get started with Azure Stream Analytics

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

200 XP

Knowledge check

3 minutes

Check your knowledge

1. Which definition of stream processing is correct? *

- [x] Data is processed continually as new data records arrive.

  ✔ Correct. Stream processing is used to continually process new data as it arrives.

- [ ] Data is collected in a temporary store, and all records are processed together as a batch.
- [ ] Data that is incomplete or contains errors is redirected to separate storage for correction by a human operator.

2. You need to process a stream of sensor data, aggregating values over one minute windows and storing the results in a data lake. Which service should you use? *

- [ ] Azure SQL Database
- [ ] Azure Cosmos DB
- [x] Azure Stream Analytics

  ✔ Correct. Azure Stream Analytics is a stream processing engine.

3. You want to aggregate event data by contiguous, fixed-length, non-overlapping temporal intervals. What kind of window should you use? *

- [ ] Sliding

  ✘ Incorrect. Sliding windows are not contiguous.

- [ ] Session
- [x] Tumbling

✔ Correct. Tumbling windows define contiguous, fixed-length, non-overlapping windows.

Next unit: Summary

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

100 XP

Introduction

1 minute

Suppose a retail company captures real-time sales transaction data from an e-commerce website, and wants to analyze this data along with more static data related to products, customers, and employees. A common way to approach this problem is to ingest the stream of real-time data into a data lake or data warehouse, where it can be queried together with data that is loaded using batch processing techniques.

Microsoft Azure Synapse Analytics provides a comprehensive enterprise data analytics platform, into which real-time data captured in Azure Event Hubs or Azure IoT Hub, and processed by Azure Stream Analytics can be loaded.

A typical pattern for real-time data ingestion in Azure consists of the following sequence of service integrations:

1. A real-time source of data is captured in an event ingestor, such as Azure Event Hubs or Azure IoT Hub.
2. The captured data is perpetually filtered and aggregated by an Azure Stream Analytics query.
3. The results of the query are loaded into a data lake or data warehouse in Azure Synapse Analytics for subsequent analysis.

In this module, you'll explore multiple ways in which you can use Azure Stream Analytics to ingest real-time data into Azure Synapse Analytics.

Next unit: Stream ingestion scenarios

Continue

100 XP

Stream ingestion scenarios

5 minutes

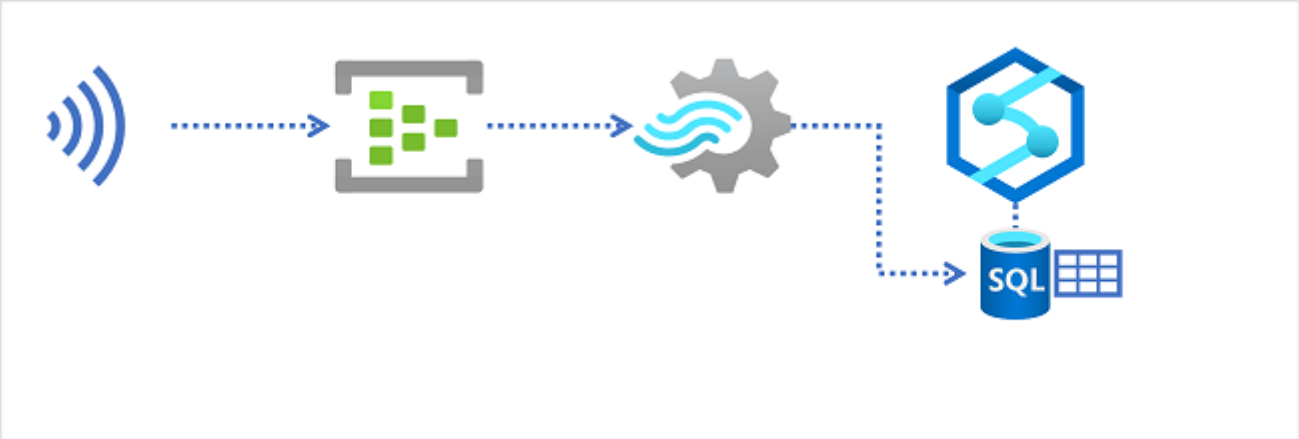
Azure Synapse Analytics provides multiple ways to analyze large volumes of data. Two of the most common approaches to large-scale data analytics are:

- **Data warehouses** - relational databases, optimized for distributed storage and query processing. Data is stored in tables and queried using SQL.
- **Data lakes** - distributed file storage in which data is stored as files that can be processed and queried using multiple runtimes, including Apache Spark and SQL.

Data warehouses in Azure Synapse Analytics

Azure Synapse Analytics provides dedicated SQL pools that you can use to implement enterprise-scale relational data warehouses. Dedicated SQL pools are based on a *massively parallel processing* (MPP) instance of the Microsoft SQL Server relational database engine in which data is stored and queried in tables.

To ingest real-time data into a relational data warehouse, your Azure Stream Analytics query must write its results to an output that references the table into which you want to load the data.

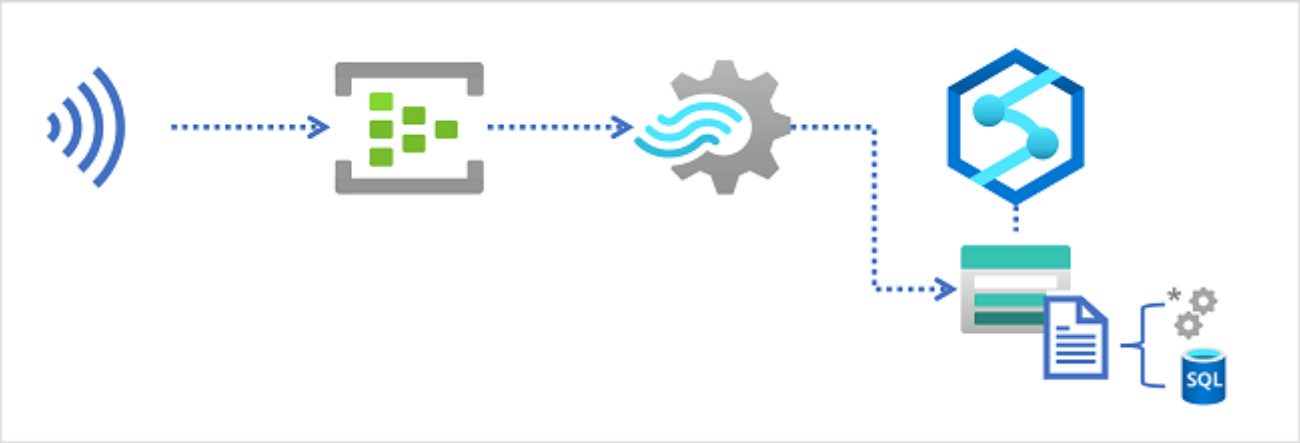


Data lakes in Azure Synapse Analytics

An Azure Synapse Analytics workspace typically includes at least one storage service that is used as a data lake. Most commonly, the data lake is hosted in an Azure Storage account using a container configured to support Azure Data Lake Storage Gen2. Files in the data lake are

organized hierarchically in directories (folders), and can be stored in multiple file formats, including delimited text (such as comma-separated values, or CSV), Parquet, and JSON.

When ingesting real-time data into a data lake, your Azure Stream Analytics query must write its results to an output that references the location in the Azure Data Lake Gen2 storage container where you want to save the data files. Data analysts, engineers, and scientists can then process and query the files in the data lake by running code in an Apache Spark pool, or by running SQL queries using a serverless SQL pool.



Next unit: Configure inputs and outputs

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

100 XP

Configure inputs and outputs

9 minutes

All Azure Stream Analytics jobs include at least one input and output. In most cases, inputs reference sources of streaming data (though you can also define inputs for static reference data to augment the streamed event data). Outputs determine where the results of the stream processing query will be sent. In the case of data ingestion into Azure Synapse Analytics, the output usually references an Azure Data Lake Storage Gen2 container or a table in a dedicated SQL pool database.

Streaming data inputs

Inputs for streaming data consumed by Azure Stream Analytics can include:

- Azure Event Hubs
- Azure IoT Hubs
- Azure Blob or Data Lake Gen 2 Storage

Depending on the specific input type, the data for each streamed event includes the event's data fields as well as input-specific metadata fields. For example, data consumed from an Azure Event Hubs input includes an **EventEnqueuedUtcTime** field indicating the time when the event was received in the event hub.

> **Note**
>
> For more information about streaming inputs, see **Stream data as input into Stream Analytics** in the Azure Stream Analytics documentation.

Azure Synapse Analytics outputs

If you need to load the results of your stream processing into a table in a dedicated SQL pool, use an **Azure Synapse Analytics** output. The output configuration includes the identity of the dedicated SQL pool in an Azure Synapse Analytics workspace, details of how the Azure Stream Analytics job should establish an authenticated connection to it, and the existing table into which the data should be loaded.

Authentication to Azure Synapse Analytics is usually accomplished through SQL Server authentication, which requires a username and password. Alternatively, you can use a managed identity to authenticate. When using an Azure Synapse Analytics output, your Azure Stream Analytics job configuration must include an Azure Storage account in which authentication metadata for the job is stored securely.

> **Note**
>
> For more information about using an Azure Synapse Analytics output, see **Azure Synapse Analytics output from Azure Stream Analytics** in the Azure Stream Analytics documentation.

Azure Data Lake Storage Gen2 outputs

If you need to write the results of stream processing to an Azure Data Lake Storage Gen2 container that hosts a data lake in an Azure Synapse Analytics workspace, use a **Blob storage/ADLS Gen2** output. The output configuration includes details of the storage account in which the container is defined, authentication settings to connect to it, and details of the files to be created. You can specify the file format, including CSV, JSON, Parquet, and Delta formats. You can also specify custom patterns to define the folder hierarchy in which the files are saved - for example using a pattern such as *YYYY/MM/DD* to generate a folder hierarchy based on the current year, month, and day.

You can specify minimum and maximum row counts for each batch, which determines the number of output files generated (each batch creates a new file). You can also configure the *write mode* to control when the data is written for a time window - appending each row as it arrives or writing all rows once (which ensures "exactly once" delivery).

> **Note**
>
> For more information about using a Blob storage/ADLS Gen2 output, see **Blob storage and Azure Data Lake Gen2 output from Azure Stream Analytics** in the Azure Stream Analytics documentation.

Next unit: Define a query to select, filter, and aggregate data

100 XP

Define a query to select, filter, and aggregate data

5 minutes

After defining the input(s) and output(s) for your Azure Stream Analytics job, you can define a *query* to process the incoming data from an input and write the results to an output.

Selecting input fields

The simplest approach to ingesting streaming data into Azure Synapse Analytics is to capture the required field values for every event using a **SELECT...INTO** query, as shown here:

SQL

```
SELECT
    EventEnqueuedUtcTime AS ReadingTime,
    SensorID,
    ReadingValue
INTO
    [synapse-output]
FROM
    [streaming-input] TIMESTAMP BY EventEnqueuedUtcTime
```

Tip

When using an **Azure Synapse Analytics** output to write the results to a table in a dedicated SQL pool, the schema of the results produced by the query must match the table into which the data is to be loaded. You can use **AS** clauses to rename fields, and cast them to alternative (compatible) data types as necessary.

Filtering event data

In some cases, you might want to filter the data to include only specific events by adding a **WHERE** clause. For example, the following query writes data only for events with a negative **ReadingValue** field value.

SQL

```
SELECT
    EventEnqueuedUtcTime AS ReadingTime,
    SensorID,
    ReadingValue
INTO
    [synapse-output]
FROM
    [streaming-input] TIMESTAMP BY EventEnqueuedUtcTime
WHERE ReadingValue < 0
```

Aggregating events over temporal windows

A common pattern for streaming queries is to aggregate event data over temporal (time-based) intervals, or *windows*. To accomplish this, you can use a **GROUP BY** clause that includes a Window function defining the kind of window you want to define (for example, *tumbling*, *hopping*, or *sliding*).

> **Tip**
>
> For more information about window functions, see **Introduction to Stream Analytics windowing functions** in the Azure Stream Analytics documentation.

The following example groups streaming sensor readings into 1 minute tumbling (serial, non-overlapping) windows, recording the start and end time of each window and the maximum reading for each sensor. The **HAVING** clause filters the results to include only windows where at least one event occurred.

SQL

```
SELECT
    DateAdd(second, -60, System.TimeStamp) AS StartTime,
    System.TimeStamp AS EndTime,
    SensorID,
    MAX(ReadingValue) AS MaxReading
INTO
    [synapse-output]
FROM
    [streaming-input] TIMESTAMP BY EventEnqueuedUtcTime
GROUP BY SensorID, TumblingWindow(second, 60)
HAVING COUNT(*) >= 1
```

> **Tip**

For more information about common patters for queries, see **Common query patterns in Azure Stream Analytics** in the Azure Stream Analytics documentation.

Next unit: Run a job to ingest data

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

100 XP

Run a job to ingest data

3 minutes

When you've created and saved your query, you can run the Azure Stream Analytics job to process events in the input(s) and write the results to output(s). Once started, the query will run perpetually until stopped; constantly ingesting new event data into your Azure Synapse Analytics workspace (into a table in relational data warehouse or files in a data lake, depending on the output type).

Working with ingested data

You can work with the ingested streaming data like any other data in Azure Synapse Analytics, combining it with data ingested using batch processing techniques or synchronized from operational data sources by using Azure Synapse Link.

Querying data in a relational data warehouse

If you used an Azure Synapse Analytics output to ingest the results of your stream processing job into a table in a dedicated SQL pool, you can query the table using a SQL query, just like any other table. The results of the query will always include the latest data to be ingested at the time the query is run. Your data warehouse can include tables for streaming data as well as tables for batch ingested data, enabling you to join real-time and batch data for historical analytics.

For example, the following SQL code could be used to query a table named **factSensorReadings** that contains the results of stream processing, and combine it with a **dimDate** table containing detailed data about the dates on which readings were captured.

SQL

```
SELECT d.Weekday, s.SensorID, AVG(s.SensorReading) AS AverageReading
FROM factSensorReadings AS s
JOIN dimDate AS d
    ON CAST(s.ReadingTime AS DATE) = d.DateKey
GROUP BY d.Weekday, s.SensorID
```

Tip

To Learn more about using a dedicated SQL pool to analyze data in a data warehouse, see the **Analyze data in a relational data warehouse** module on Microsoft Learn.

Querying data in a data lake

As streaming data is ingested into files in a data lake, you can query those files by using a serverless SQL pool in Azure Synapse Analytics. For example, the following query reads all fields from all Parquet files under the **sensors** folder in the **data** file system container.

SQL

```
SELECT *
FROM OPENROWSET(
    BULK 'https://mydatalake.blob.core.windows.net/data/sensors/*',
    FORMAT = 'parquet') AS rows
```

Tip

To Learn more about using serverless SQL pools to query files in a data lake, see the **Use Azure Synapse serverless SQL pool to query files in a data lake** module on Microsoft Learn.

You can also query the data lake by using code running in an Apache Spark pool, as shown in this example:

Python

```
%%pyspark
df =
spark.read.load('abfss://data@datalake.dfs.core.windows.net/sensors/*',
format='parquet'
)
display(df)
```

Tip

To Learn more about using Apache Spark pools to query files in a data lake, see the **Analyze data with Apache Spark in Azure Synapse Analytics** module on Microsoft Learn.

200 XP

Knowledge check

3 minutes

Check your knowledge

1. Which type of output should you use to ingest the results of an Azure Stream Analytics job into a dedicated SQL pool table in Azure Synapse Analytics? *

- Azure Synapse Analytics

  ✔ Correct. An Azure Synapse Analytics output writes data to a table in an Azure Synapse Analytics dedicated SQL pool.

- Blob storage/ADLS Gen2
- Azure Event Hubs

2. Which type of output should be used to ingest the results of an Azure Stream Analytics job into files in a data lake for analysis in Azure Synapse Analytics? *

- Azure Synapse Analytics
- Blob storage/ADLS Gen2

  ✔ Correct. A Blob storage/ADLS Gen2 output writes data to files in a data lake.

- Azure Event Hubs

  ✕ Incorrect. An Azure Event Hubs output does not write data to files in a data lake.

Next unit: Summary

Continue >

100 XP

Introduction

2 minutes

Microsoft Power BI is used by organizations all around the world to create dynamic, interactive data visualizations that reveal insights on which important business decisions are based. Access to timely data can be the difference between failure and success, so the ability to capture and visualize data in real-time, or as near as possible, is critical in many scenarios.

Azure Stream Analytics provides a way to process a stream of real-time data from an input such as Azure Event Hubs, and direct the results to an output. One possible output is a Power BI dataset, from which dashboards can consume data for real-time visualization.

In this module, we'll examine how to use Azure Stream Analytics to process a stream of real-time data, and send the results to a Power BI dataset for visualization.

Next unit: Use a Power BI output in Azure Stream Analytics

Continue

100 XP

Use a Power BI output in Azure Stream Analytics

6 minutes

All Azure Stream Analytics jobs include at least one input and output. In most cases, inputs reference sources of streaming data (though you can also define inputs for static reference data to augment the streamed event data). Outputs determine where the results of the stream processing query will be sent. To support real-time data visualization, you can use a **Power BI** output.

Streaming data inputs

Inputs for streaming data consumed by Azure Stream Analytics can include:

- Azure Event Hubs
- Azure IoT Hubs
- Azure Blob or Data Lake Gen 2 Storage

Depending on the specific input type, the data for each streamed event includes the event's data fields and input-specific metadata fields. For example, data consumed from an Azure Event Hubs input includes an **EventEnqueuedUtcTime** field indicating the time when the event was received in the event hub.

> **Note**
>
> For more information about streaming inputs, see **Stream data as input into Stream Analytics** in the Azure Stream Analytics documentation.

Power BI outputs

You can use a Power BI output to write the results of a Stream Analytics query to a table in a Power BI streaming dataset, from where it can be visualized in a dashboard. When adding a Power BI output to a Stream Analytics job, you need to specify the following properties:

- **Output alias**: A name for the output that can be used in a query.

- **Group workspace**: The Power BI workspace in which you want to create the resulting dataset.
- **Dataset name**: The name of the dataset to be generated by the output. You shouldn't pre-create this dataset as it will be created automatically (replacing any existing dataset with the same name).
- **Table name**: The name of the table to be created in the dataset.
- **Authorize connection**: You must authenticate the connection to your Power BI tenant so that the Stream Analytics job can write data to the workspace.

> **Note**
>
> For more information about Power BI outputs, see **Power BI output from Azure Stream Analytics** in the Azure Stream Analytics documentation.

Next unit: Create a query for real-time visualization

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

100 XP

Create a query for real-time visualization

6 minutes

To send streaming data to Power BI, your Azure Stream Analytics job uses a query that writes its results to a Power BI output. A simple query that forwards event data from an event hub directly to Power BI might look something like this:

SQL

```
SELECT
    EventEnqueuedUtcTime AS ReadingTime,
    SensorID,
    ReadingValue
INTO
    [powerbi-output]
FROM
    [eventhub-input] TIMESTAMP BY EventEnqueuedUtcTime
```

The results of the query determine the schema of the table in the output dataset in Power BI.

Alternatively, you might use your query to filter and/or aggregate the data, sending only relevant or summarized data to the Power BI dataset. For example, the following query calculates the maximum reading for each sensor other than sensor *0* for each consecutive minute in which an event occurs.

SQL

```
SELECT
    DateAdd(second, -60, System.TimeStamp) AS StartTime,
    System.TimeStamp AS EndTime,
    SensorID,
    MAX(ReadingValue) AS MaxReading
INTO
    [powerbi-output]
FROM
    [eventhub-input] TIMESTAMP BY EventEnqueuedUtcTime
WHERE SensorID <> 0
GROUP BY SensorID, TumblingWindow(second, 60)
HAVING COUNT(*) > 1
```

When working with window functions (such as the **TumblingWindow** function in the previous example), consider that Power BI is capable of handling a call every second. Additionally, streaming visualizations support packets with a maximum size of 15 KB. As a general rule, use

window functions to ensure data is sent to Power BI no more frequently than every second, and minimize the fields included in the results to optimize the size of the data load.

> **Note**
>
> For more information about Power BI output limitations, see **Power BI output from Azure Stream Analytics** in the Azure Stream Analytics documentation.

Next unit: Create real-time data visualizations in Power BI

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

100 XP

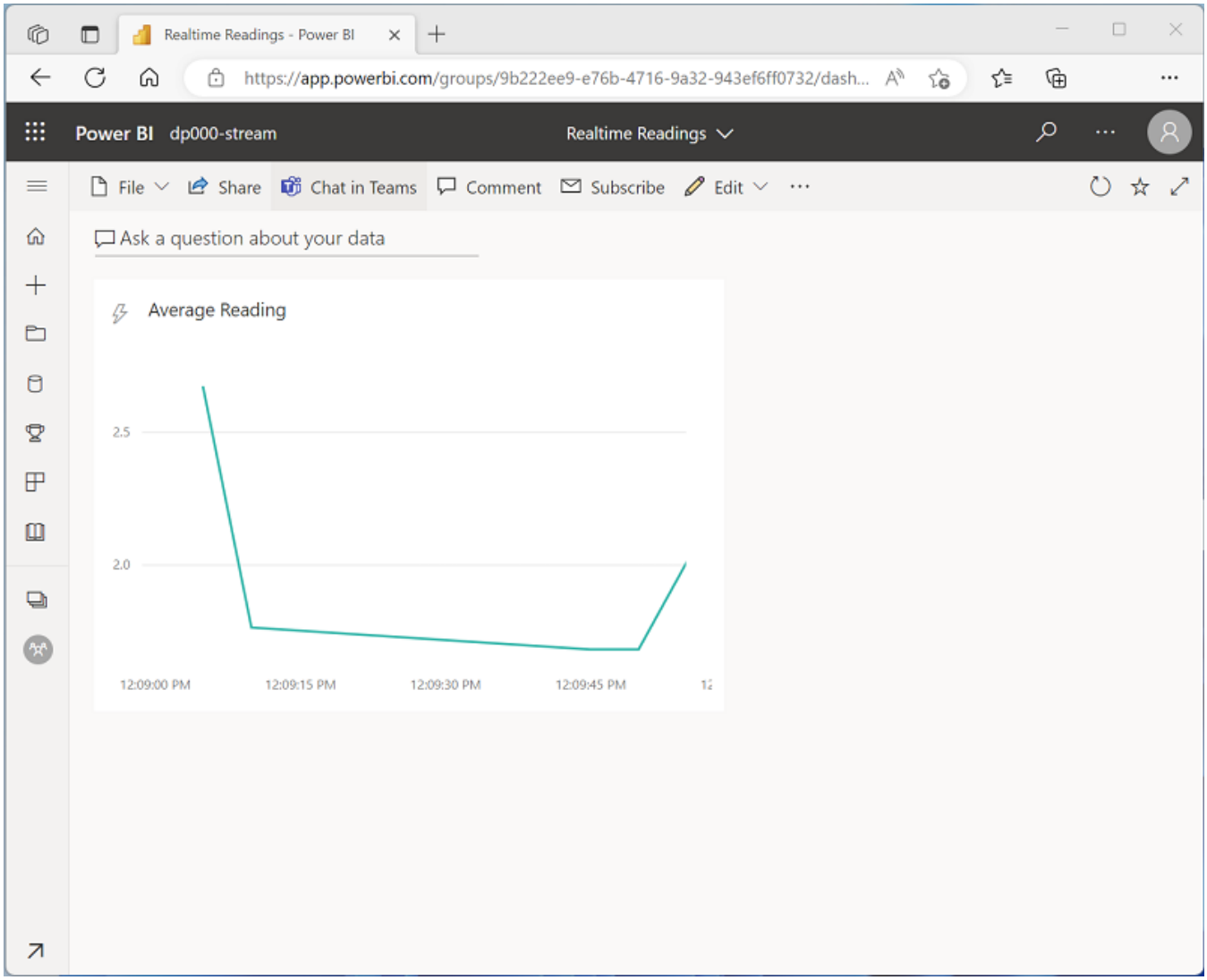
Create real-time data visualizations in Power BI

5 minutes

When you successfully run an Azure Stream Analytics job that sends results to a Power BI output, a streaming dataset containing a single table is created in the Power BI workspace specified for the output. The table contains the data produced by the Stream Analytics query.

Creating real-time visualizations in a dashboard

To visualize data in real-time, you can create a *dashboard* with a real-time visualization tile. Real-time visualizations on a dashboard show data from a streaming dataset, and are updated dynamically as new data flows into the dataset.



200 XP

Knowledge check

3 minutes

Check your knowledge

1. Which type of Azure Stream Analytics output should you use to support real-time visualizations in Microsoft Power BI? *

- ( ) Azure Synapse Analytics

  ✗ Incorrect. An Azure Synapse Analytics output writes data to a table in an Azure Synapse Analytics dedicated SQL pool.

- ( ) Azure Event Hubs
- ( ) Power BI

  ✓ Correct. A Power BI output creates a dataset with a table of streaming data in a Power BI workspace.

2. You want to use an output to write the results of a Stream Analytics query to a table named device-events in a dataset named realtime-data in a Power BI workspace named analytics workspace. What should you do? *

- ( ) Create only the workspace. The dataset and table will be created automatically.

  ✓ Correct. The dataset and table will be created dynamically by the output.

- ( ) Create the workspace and dataset. The table will be created automatically.

  ✗ Incorrect. The dataset and table will be created dynamically by the output.

- ( ) Create the workspace, dataset, and table before creating the output.

3. You want to create a visualization that updates dynamically based on a table in a streaming dataset in Power BI. What should you do? *

- ( ) Create a report from the dataset.

- [ ] Create a dashboard with a tile based on the streaming dataset.

✔ **Correct. A dashboard with a tile based on a streaming dataset updates dynamically as new data arrives.**

- [ ] Export the streaming dataset to Excel and create a report from the Excel workbook.

✕ **Incorrect. A report based on an Excel workbook does not update dynamically.**

Next unit: Summary

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆