Desing and Develop Data Processing – Azure Event Hubs, Stream Analytics

Real-Time Data Processing ?

◇ What is Data Processing?

When devices, websites, or applications send data constantly (like temperature sensors, user clicks, or vehicle speed), we need a way to:

1. Collect that data
2. Process it (like calculate average or filter out unwanted data)
3. Store or display it in real-time (like in a dashboard or database)

This is called real-time data processing.

◇ 1. What is Azure Event Hubs?

Think of it like a “data collector”.

- It collects live data from many devices or applications.
- It can receive millions of messages per second.
- It does not process the data — just collects and holds it for a short time.

Example:

Imagine you have 1000 temperature sensors sending data every second.
All these messages go into Event Hub, which holds them like a mailbox.

◇ 2. What is Azure Stream Analytics?

Think of it like a “data processor”.

- It reads the data from Event Hubs.
- It processes it in real-time using simple SQL-like queries.
- It can do things like:
    - Find the average temperature every 1 minute.

- Filter only data where temperature > 50.
- Count how many times a device sent data.

Then, it sends the final output to:

- A database (like Azure SQL)
- A dashboard (like Power BI)
- A storage file (like Azure Blob)

🔁 Overall Flow

Let's look at the flow like a pipeline:

```text
[ Devices or Apps ]
        ↓
[ Azure Event Hub ]
  (Collects real-time data)
        ↓
[ Azure Stream Analytics ]
  (Processes data using SQL)
        ↓
[ Output: Database, Dashboard, File ]
```

🔧 Step-by-Step Example: Real-Time Temperature Monitoring

Let's imagine you're creating a system for monitoring temperatures from sensors in different rooms.

▶ Step 1: Set up Event Hub

1. Go to Azure Portal → Create a Namespace → Create an Event Hub.
2. This will be the place where all your sensors send data.

▶ Step 2: Send Data to Event Hub

Each sensor sends data like this every few seconds:

You can send this data using:

- Python
- JavaScript
- C#
  (Example code is available — I can give it if you need.)

▶ Step 3: Set up Stream Analytics Job

Now we create a **Stream Analytics job** to **read and process** the data.

☑ Input:

- Source: **Event Hub**
- This tells Stream Analytics where to get the live data from.

☑ Query (like SQL):

Here's a simple query:

```sql
SELECT
  deviceId,
  AVG(temperature) AS averageTemp
INTO
  sqlOutput
FROM
  eventHubInput TIMESTAMP BY enqueuedTime
GROUP BY
  deviceId, TumblingWindow(minute, 1)
```

🥏 What it does:

- For every room (deviceId),
- It calculates the **average temperature every 1 minute**.

☑ **Output:**

- Destination: **Azure SQL Database** (or Power BI, Excel, etc.)
- This is where your processed data goes.

▶ **Step 4: Start and Monitor**

- Start the Stream Analytics job.
- Watch as your system receives data → processes it → stores it → visualizes it (like on a chart or dashboard).

🧠 **Simple Analogy**

Imagine you're running a water-purifying plant:

Part	Real Life	Azure Service
Water from many taps	Live data from devices/apps	Event Hubs (collects water/data)
Water filter	Cleans & processes the water	Stream Analytics (filters/processes)
Water tank	Stores clean water	SQL DB / Power BI (stores result)

What Do You Pay For in Streaming Services?

1. How much data you send (Ingress)

- The more **events or messages** you send into the service, the higher the cost.
- 📦 Think of it like paying for how many **packages you drop off** at a post office.

---

2. How much data you read or pull out (Egress)

- If you read the data from Event Hub or stream it to another service, that costs too.
- 📦 Like paying the **post office to deliver** your packages.

---

3. Retention Time (How long data is kept)

- You can choose how many days your messages are stored.
- Longer retention = more cost.
- 🗂️ Like renting extra lockers to keep your old mail for later use.

---

4. Throughput Units (Speed + Power)

- Streaming services let you **buy power units** to control how fast data can flow.
- More **throughput units (TUs)** = more speed = more cost.
- 🚚 Like paying more for faster internet or extra lanes on a highway.

---

5. Features Used (like capture, encryption, geo-replication)

- Some **extra features** cost more.
- For example:
    - Saving raw data to storage (called “capture” in Azure Event Hub)
    - Keeping copies in multiple regions (geo-replication)

📝 **Example in Simple Words:**

Let’s say you run an online app:

- You send **1 million messages per day** to Azure Event Hub ☑
- You want to **store those messages for 2 days** ☑
- And your app **reads the messages in real time** ☑

👉 You will be charged for:

- The **amount of data in and out**
- The **storage time**
- The **speed (throughput)** needed
- Any **extra features** you enable

🎯 **Summary:**

Cost Area	**What it Means**	**Analogy**
Ingress	Messages going in	Dropping packages at post office
Egress	Messages being read	Packages being delivered
Retention	How long messages are stored	Renting lockers
Throughput Units	Speed and power of processing	Internet speed lanes
Add-ons	Extra services	Premium delivery options

Batch and Real-Time Processing explain in simple words and in detail

What is Batch Processing?

➤ **Simple Definition:**

Batch Processing means collecting **a large amount of data** and **processing it all at once** at a scheduled time (like every hour, day, or week).

🔘 **Key Features:**

- Data is **processed in groups (batches)**.
- Not instant — there's **some delay**.
- Great for tasks that don't need immediate results.

📦 **Real-life Example:**

Think of it like **doing laundry** — you collect dirty clothes and wash them **in one go**, not one shirt at a time.

💻 **Tech Example:**

- Processing all bank transactions at the end of the day.
- Generating salary slips for all employees at month-end.

⚡ **What is Real-Time Processing?**

➤ **Simple Definition:**

Real-Time Processing means **handling data immediately as it arrives**, with very little delay (milliseconds to seconds).

🔘 **Key Features:**

- Data is **processed instantly**.
- Useful when immediate action or response is needed.
- Often used in **live systems**.

📦 **Real-life Example:**

Imagine you're on a **video call** — the audio and video must be processed in real-time, or there will be a delay.

💻 **Tech Example:**

- Fraud detection in online banking (alerting while a transaction is happening).
- Uber showing your ride moving on the map in real-time.

📊 **Comparison Table**

Feature	Batch Processing	Real-Time Processing
Speed	Slower (delayed)	Fast (instant)
Data Handling	Processes large chunks at once	Processes each event/message immediately
Use Case	Reports, payroll, backups	Stock trading, live tracking, fraud detection
Tools	Hadoop, Azure Data Factory	Apache Kafka, Azure Event Hub, AWS Kinesis
Complexity	Simpler	More complex setup
Cost	Generally cheaper	Can be more expensive

What is Azure Event Hubs?

➤ **Simple Definition:**

Azure Event Hubs is a **big mailbox for messages** (events) that come from many sources — like apps, sensors, or websites — and sends them to other systems that need them.

You can think of it as a **central hub** for collecting and distributing huge amounts of data **in real time**.

📦 **Real-Life Example (Simple Analogy):**

Imagine a busy **airport**:

- ✈️ Flights (data producers like apps, IoT devices) arrive at the airport.
- 🧳 Passengers (messages/events) go into the **baggage claim (Event Hub)**.
- 🚕 Taxis (data consumers like databases or analytics tools) pick up passengers and take them to their destinations.

This is how **Azure Event Hubs works** — it takes messages from many sources and sends them where they are needed.

🛠️ What Can Send Data to Event Hubs?

- Mobile apps 📱
- Websites 🌐
- IoT devices (e.g., sensors) 🔌
- Games 🎮
- Software systems 💻

These are called **"producers"**.

🎯 Where Does the Data Go After?

- To a **database**
- To **analytics tools** like Azure Stream Analytics or Power BI
- To **real-time dashboards**
- To **machine learning models**
- To **alerts/notifications**

These are called **"consumers"**.

⚙️ Key Features:

Feature	Explanation
Real-time data	Sends and receives data almost instantly
High throughput	Can handle **millions of messages per second**
Scalable	Grows with your app's data
Durable	Keeps data safe for a limited time (default 1 day, up to 7 days or more)
Multiple readers	Many systems can read the same data

📬 What is MSMQ (Microsoft Message Queuing)?

MSMQ is a **message queue system developed by Microsoft** that allows applications to send messages to each other, even if they are not running at the same time.

Simple Explanation:

- Works like a **mailbox for programs** on **Windows**.
- App A drops a message into MSMQ → MSMQ stores it → App B reads it later.
- Good for **offline communication** between apps.

🐰 What is RabbitMQ?

- **RabbitMQ** is a **popular open-source message broker** that helps systems communicate by **passing messages asynchronously** using a protocol called **AMQP**.

Creating an instance of Event hub

🧿 Steps to Create an Event Hub

Step 1: Go to Azure Portal

- Open https://portal.azure.com
- Sign in with your Azure account

Step 2: Create a Namespace

A namespace is like a **container** for your Event Hubs.

1. In the search bar, type Event Hubs and click **Event Hubs**.
2. Click **+ Add** to create a new namespace.
3. Fill in the details:
    - **Subscription**: Choose your Azure subscription.
    - **Resource group**: Create a new one or select existing.

- **Namespace name**: Give a unique name (e.g., myeventhubns).
- **Location**: Choose a region (like Central India).
- **Pricing Tier**: Select Basic or Standard (Basic is fine for learning).

4. Click **Review + Create**, then click **Create**.

⏳ Wait a few seconds for the deployment to finish.

Step 3: Create an Event Hub

Now that you have a namespace, let's add an Event Hub inside it.

1. Open the **namespace** you just created.
2. In the left menu, click **Event Hubs** under **Entities**.
3. Click **+ Event Hub**.
4. Fill in the details:
   - **Name**: Enter a name (e.g., myhub).
   - **Partition count**: Leave default (usually 2–4 is enough).
   - **Message retention**: Leave default (1 day or adjust).
5. Click **Create**.

☑ Your Event Hub is now ready to receive and stream events!

🎯 **Optional: Get Connection String**

To send or receive data using code, you'll need the **connection string**:

1. Go to your Event Hub namespace.
2. Click **Shared access policies**.
3. Click the **policy name** (e.g., RootManageSharedAccessKey).
4. Copy the **Connection string – primary key**.

Azure Event Hubs - Sending data

🎯 Goal:

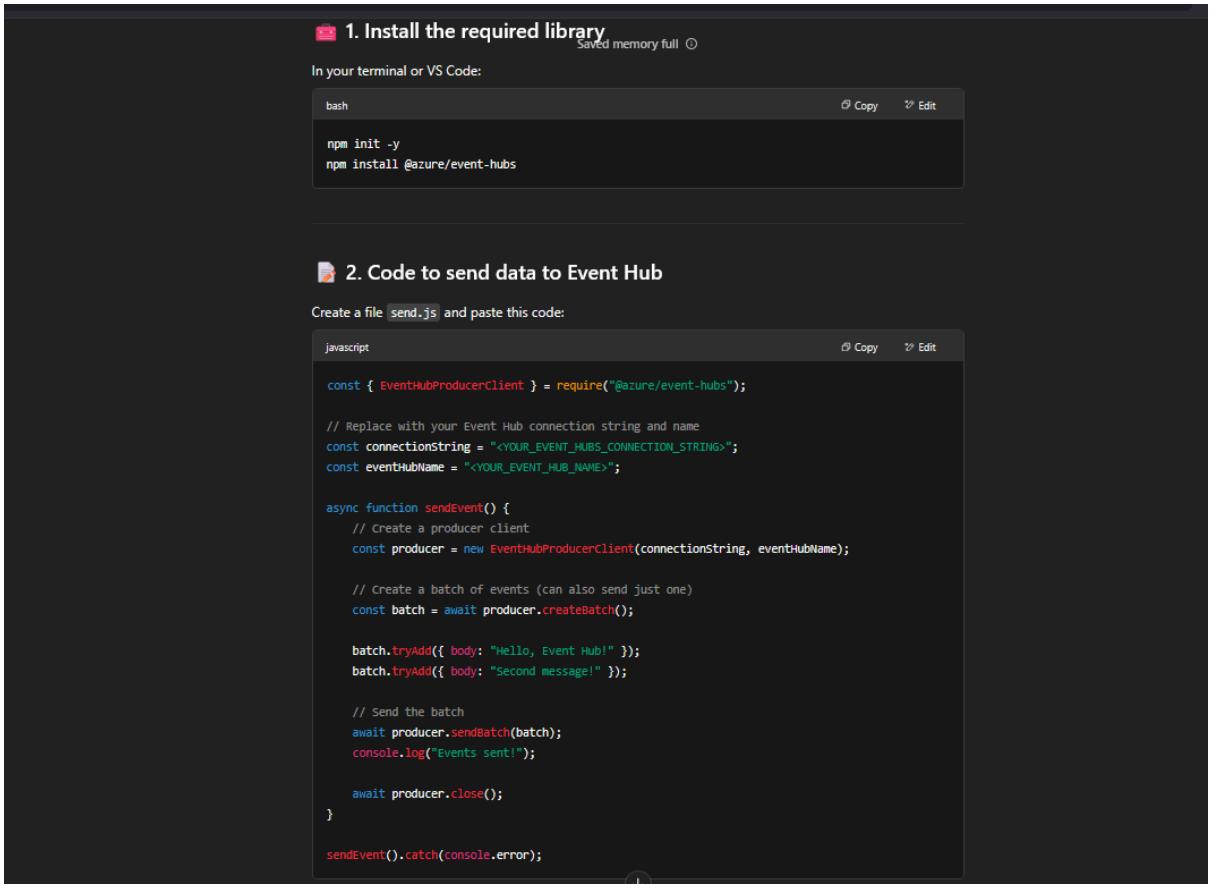
We will send some sample events (messages) to Azure Event Hub using code.

☑️ Prerequisites:

- You already created an **Event Hub** and have its **connection string**.
- You have **Node.js** installed on your system.
- Your Event Hub name and namespace are ready.

🧰 1. Install the required library

In your terminal or VS Code:



🔐 Replace:

- <YOUR_EVENT_HUBS_CONNECTION_STRING> with your connection string from Azure.
- <YOUR_EVENT_HUB_NAME> with your Event Hub's name.

🩹 Where does this data go?

The data is now inside **Azure Event Hub**, and you can:

- Monitor it via **Azure Portal**
- Connect it to a **consumer** (like Azure Stream Analytics, Databricks, or custom code) to read and process it.

💡 What is Azure Stream Analytics?

Azure Stream Analytics is a **real-time data processing tool** from Microsoft Azure.
It helps you **analyze and process data as it flows in**, rather than after it's stored.

🧠 Simple Definition:

Azure Stream Analytics lets you take fast-moving data (like from apps, IoT devices, or sensors), **filter it**, **analyze it**, and **send it** somewhere useful — like dashboards, databases, or alerts.

🎯 Real-Life Analogy:

Imagine you work at a **water treatment plant**:

- Water is coming in through pipes (like data from devices or apps).
- You have filters and sensors (Stream Analytics) to clean or check the water **as it flows**.
- Clean water goes to houses (processed data goes to dashboards, storage, or alerts).

So, **Stream Analytics** is like the real-time filter and monitor for your fast-flowing data.

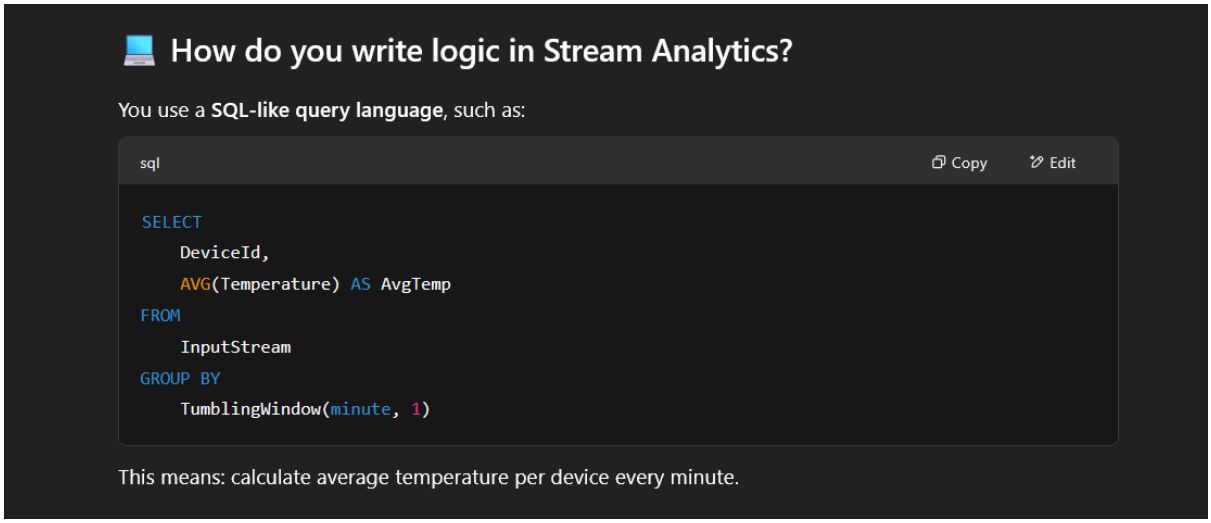
📥 What kind of data does it work on?

Azure Stream Analytics works best with:

- **Event Hubs** (for event streams)
- **IoT Hub** (for data from sensors/devices)
- **Blob Storage** (optional, for static data comparison)

🛠️ What can you do with it?

Task	Example
Filter data	Only take sensor readings above 100°C
Aggregate values	Calculate average temperature every 10 seconds
Join data	Combine device info with sensor data
Detect patterns	Find anomalies in user clicks or vehicle speed
Trigger alerts/actions	If heart rate > 180, send alert
Send data	To Power BI, SQL, Storage, Functions, etc.



This means: calculate average temperature per device every minute.

🔁 Input → Process → Output

◇ Inputs:

- Event Hub
- IoT Hub
- Blob Storage

◇ Query Processing:

- SQL-like logic (filters, joins, aggregations)

◇ Outputs:

- Power BI (for live dashboards)
- Azure SQL Database
- Azure Functions (to take action)
- Azure Blob Storage
- Data Lake

🔘 Pricing (Simple Idea)

You pay for:

- How many **Streaming Units** (SUs) you use — based on data size + speed.
- Can scale up or down as per your data volume.

Feature	Explanation
Real-time processing	Handles data as it arrives
Easy SQL syntax	Write logic with simple SQL queries
Multiple inputs/outputs	Flexible data flow
Fully managed	No server setup needed
Scalable & Reliable	Can handle millions of events per second

What is a Stream Analytics Job?

A **Stream Analytics job** is like a mini-program in Azure that:

- **Takes real-time data** from a source (like Event Hub or IoT Hub),
- **Processes it** using SQL-like queries (filter, group, average, etc.),
- **Sends the result** to an output (like Power BI, Azure SQL, or Storage).

Now let's see how to **create one step-by-step**!

🛠️ Step-by-Step: How to Create a Stream Analytics Job

☑️ Step 1: Go to Azure Portal

- Open your browser and go to https://portal.azure.com
- Log in with your Azure account

☑️ Step 2: Search for "Stream Analytics Jobs"

- On the left menu or top search bar, type: **Stream Analytics Jobs**
- Click on it

☑️ Step 3: Click + Create

- You'll see a button that says **+ Create**
- Click on it to start creating your job

☑️ Step 4: Fill in Basic Details

You'll be asked to enter some simple info:

Field	What to Fill
Subscription	Choose your Azure subscription
Resource Group	Choose an existing group or create a new one
Job Name	Give a name like MyStreamJob
Region	Choose where you want to run the job (like India, US, etc.)
Streaming Units	Leave as 1 (you can change later)

Click Review + Create, then Create

🎉 Congrats! You've created your Stream Analytics Job!

🎯 What's next?

After creating the job, you'll set up:

1. Input (Where the data comes from – like Event Hub)
2. Output (Where the results go – like Power BI or SQL)
3. Query (How you want to process the data – using SQL)

If you want, I can guide you step-by-step for each next part:
➞ Setting Input
➞ Setting Output
➞ Writing the Query
➞ Running the job

Would you like to continue with that?

Perfect! Let's continue and complete the whole process **step-by-step** after creating your **Stream Analytics job**. You'll learn how to:

1. ☑ Set the **Input** (from Event Hub)
2. ☑ Set the **Output** (to Power BI, SQL, or Storage)
3. ☑ Write a **Query** to process the data
4. ☑ Start the **job**

◇ Step 1: Add Input (from Event Hub)

This tells the job **where the data is coming from**.

▶ How to do it:

1. Go to your **Stream Analytics job** (open it from the portal)
2. On the left menu, click **Inputs**
3. Click **+ Add stream input** → Select **Event Hub**
4. Fill the form:

- **Input alias**: inputhub (or any name)
- **Subscription**: select yours
- **Event Hub namespace**: choose the one you created
- **Event Hub name**: select your event hub
- **Policy name**: choose the one with **Listen** permission

5. Click **Save**

☑ Done! Now your job knows where to read incoming data from.

◇ Step 2: Add Output (to Power BI, SQL, etc.)

This tells the job **where to send processed results**.

Let's use **Power BI** for live visualization.

▶ How to do it:

1. In your Stream Analytics job, click **Outputs**
2. Click **+ Add > Power BI**
3. Sign in to your **Power BI account**
4. Fill the form:
   - **Output alias**: pbioutput
   - **Group workspace**: (use default or any other)
   - **Dataset name**: e.g., SensorData
   - **Table name**: e.g., LiveTemp
5. Click **Save**

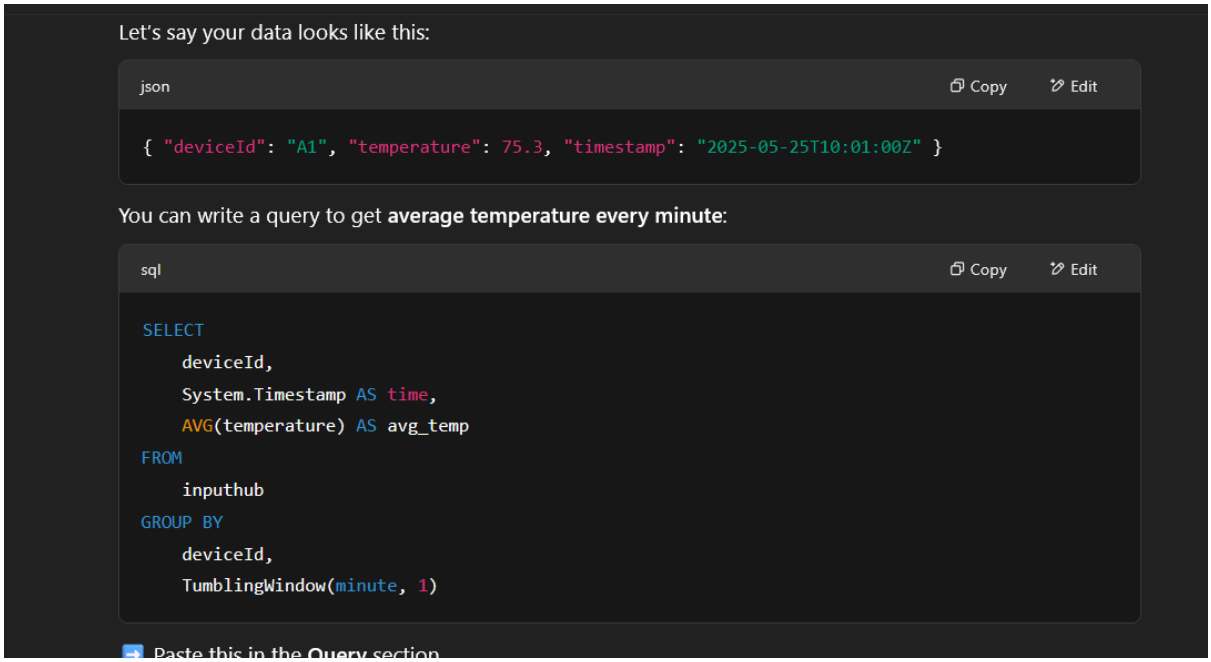
☑ Now your results will go directly to Power BI!

◇ Step 3: Write the Query

This is the brain of your job. You use SQL-like commands to filter, group, or process the data.

▶ Example Query:

Let's say your data looks like this:



➞ Paste this in the **Query** section

☑ Now your job will group data every 1 minute and calculate average temperature.

◇ Step 4: Start the Stream Analytics Job

1. Click **Start**
2. Choose:
    - **Input**: your Event Hub
    - **Output**: your Power BI
3. Click **Start**

That's it!

🎉 Final Result:

- Your **devices send data to Event Hub**
- The **Stream Analytics job processes it**
- Power BI **shows the result in real time** (like a live chart)

Let me know if you want help with:

- Sending sample data to Event Hub
- Creating a chart in Power BI
- Using different outputs like Blob storage or SQL

What Does "Defining the Input" Mean?

In **Azure Stream Analytics**, **defining the input** means telling Azure:

“Where is the data coming from?”

Your **Stream Analytics job** needs a **data source** to read from. This source is called the **input**.

🧠 Why Is It Important?

Stream Analytics is used to process **real-time streaming data**.
But before processing, it must know:

- What is the source of the data?
- What format is the data in?
- How to connect to it?

This is why **defining the input** is the first and most important step.

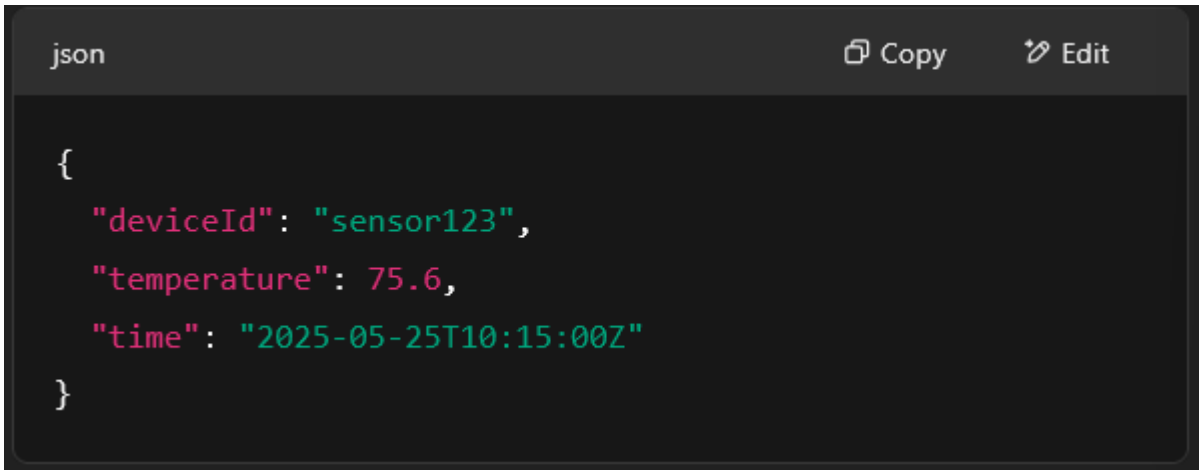
📥 Types of Inputs in Stream Analytics

You can choose from 3 common input types:

Input Type	Used For
Azure Event Hub	Collecting data from apps, sensors, logs, etc.
Azure IoT Hub	Collecting data from IoT devices like sensors
Azure Blob Storage	Reading data from files (JSON, CSV, etc.) stored in Azure

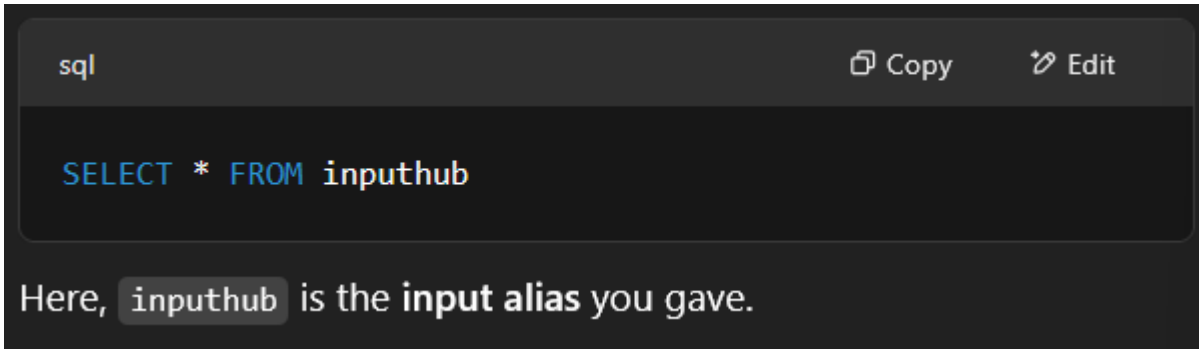
Example

Imagine your Event Hub receives data like this:



Once you define Event Hub as input:

- Stream Analytics will start **reading this JSON data**
- You can now use it in your query like:



☑ **Summary (in Simple Words)**

What You Do	Why You Do It
Define Input in Stream Analytics	To tell Azure where your streaming data is coming from
Use Event Hub, IoT Hub, or Blob	Based on real-time or file-based data

What You Do	Why You Do It
Enter connection details	So Azure can connect and read the data
Save and test the input	To confirm it is connected and ready to use

☑ What Does "Defining the Output" Mean?

Once Stream Analytics **reads data** (input) and **processes it** using a query, it needs to **send the final result somewhere**.

That **“somewhere”** is called the **output**.

So, **defining the output** means telling Azure:

"Where should the processed data go?"

🧠 Why Is It Important?

After your Stream Analytics job processes data (like filtering, aggregating, or calculating), the **results need to be stored, displayed, or used** by other apps.

Without an output:

- The processed data will go nowhere.
- Your Stream Analytics job won’t be useful.

📤 Where Can the Output Go?

Stream Analytics supports many **output destinations**, such as:

Output Type	Used For
Power BI	To visualize data in charts and dashboards
Azure Blob Storage	To store output as files (e.g., JSON/CSV)
Azure SQL Database	To insert processed data into SQL tables
Azure Data Lake Storage	For big data storage and further analytics

Output Type	Used For
Azure Table Storage	For storing data in table format
Azure Cosmos DB	For NoSQL-style output
Service Bus / Event Hub	To send processed data to another system

You can also send it to **multiple outputs** at once!

🛠️ How to Define Output (Step-by-Step)

Let's say we're sending the output to **Azure Blob Storage** (most commonly used).

☑️ Step 1: Open the Stream Analytics Job

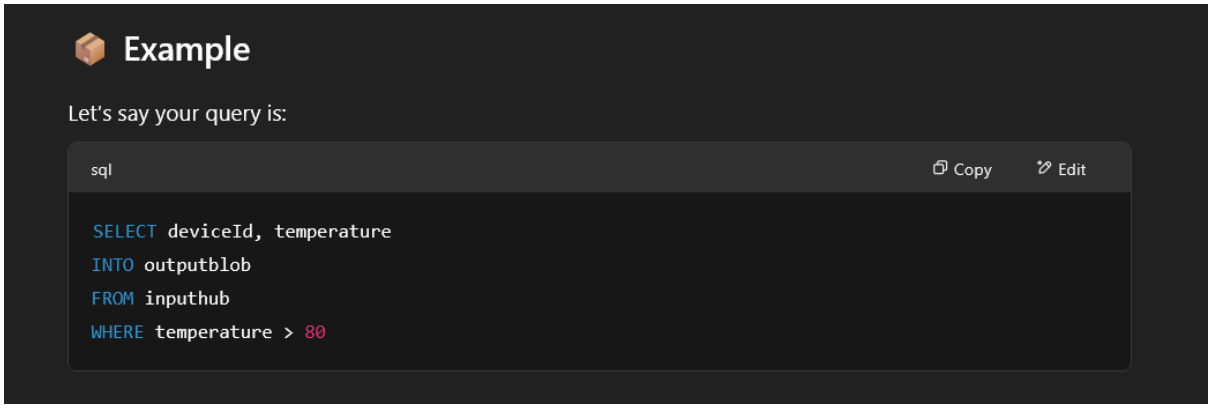
- Go to https://portal.azure.com
- Open your **Stream Analytics job**

☑️ Step 2: Add an Output

1. In the left menu, click **Outputs**
2. Click **+ Add**
3. Choose your desired output — let's pick **Blob Storage**

☑️ Step 3: Fill in the Output Settings

Field	What It Means
Output alias	Nickname for your output (e.g., outputblob)
Subscription	Your Azure subscription
Storage account	Select your storage account
Container	The folder where files will be saved
Path pattern	Format like date/hour for organizing files
Event serialization	Usually **JSON**, **CSV**, or **Avro**
Encoding	Usually **UTF-8**



- You are filtering only high temperatures
- Output alias is outputblob
- Azure will now **save those results into Blob Storage** in a selected container

---

☑ **Summary (In Simple Words)**

What You Do	Why You Do It
Define output in Stream Analytics	To tell Azure where to send final results
Choose destination (Blob, SQL, Power BI, etc.)	Based on your use case
Fill connection details	So Azure can store or send the output data
Save and test	To confirm it is connected and working

What Does "Defining the Query" Mean?

Once your Stream Analytics job:

- **Gets input** (from Event Hub, IoT Hub, or Blob Storage)
- And **knows the output** (where to send the results)

You need to write a **query** that tells Azure **what to do with the input data**.

The query is like a **set of instructions** or a **filtering/processing rule** for your data.

It uses a language **similar to SQL** (Structured Query Language), which is easy to read and understand.

🤔 Why Is the Query Important?

Without the query:

- Azure won't know what you want to do with the data.
- You won't get filtered, useful, or summarized results.

The query lets you:

- Filter data (e.g., only temperature > 100)
- Select specific fields
- Group data (e.g., by hour or location)
- Apply calculations (e.g., average, count)

✨ Query Language = SQL-Like

Azure Stream Analytics uses a **SQL-like query language**, but it's designed for **real-time, streaming data**.

🛠️ Structure of a Basic Query

Here's the basic format:

```sql
SELECT column1, column2, ...
INTO output_alias
FROM input_alias
WHERE condition
```

📌 Terms:

- SELECT: Choose what fields/columns you want

- INTO: Where to send the result (your output alias)
- FROM: Where to get the data from (your input alias)
- WHERE: Add conditions to filter data

☑ Example 1: Filter High Temperatures

```sql
SELECT deviceId, temperature
INTO outputblob
FROM inputhub
WHERE temperature > 80
```

📌 Explanation:

- Take data from inputhub
- If temperature is more than 80
- Send only the deviceId and temperature fields
- To the output named outputblob

☑ Example 2: Calculate Average Every 5 Seconds

```sql
SELECT
  System.Timestamp AS eventTime,
  AVG(temperature) AS avgTemp
INTO outputblob
FROM inputhub
GROUP BY TumblingWindow(second, 5)
```

📌 Explanation:

- For every **5-second window**, calculate the **average temperature**
- Include the event time
- Send to outputblob

🧠 What Are Windowing Functions?

Since Stream Analytics deals with **continuous data**, it needs to group data by **time windows**.

Here are common window functions:

Function	Purpose
TumblingWindow	Fixed-size window (e.g., every 5 seconds)
HoppingWindow	Overlapping windows (e.g., every 5s, lasting 10s)
SlidingWindow	Window moves with every event
SessionWindow	Groups events by activity session

Review on what we have seen so far

🔄 1. What is Azure Event Hubs?

- **Azure Event Hubs** is a **big data streaming platform**.
- It helps you **collect a huge amount of real-time data** from apps, websites, IoT devices, etc.
- Think of it as a **message collector** where all data-producing apps send their data continuously.

◇ Why use it?

Because it can handle **millions of events per second** and send them to other systems for processing.

🏐 Example: A fitness app sending live heartbeat data from 10,000 users to Event Hub.

🛠️ 2. How to Create an Event Hub Instance?

To use Event Hubs, you create:

- A **Namespace**: Like a container (a folder) that holds your event hubs.
- An **Event Hub**: The actual pipeline where the data flows in.

💡 Steps:

1. Go to Azure Portal
2. Create a **Namespace**
3. Inside it, create an **Event Hub**
4. Set things like name, region, message retention, etc.

📤 3. Sending Data to Event Hub

- You can send data from apps, scripts, or sensors using:
    - **Azure SDKs** (C#, Python, Java, etc.)
    - **Event Hub REST API**
    - **Azure IoT Hub**, **Log Analytics**, etc.

🟢 Example: You write a Python script that reads data from a temperature sensor and sends it to the Event Hub.

🌼 4. What is Azure Stream Analytics?

- **Azure Stream Analytics (ASA)** is a **real-time data processing engine**.
- It reads live data (from Event Hubs, IoT Hubs, or Storage) and lets you **filter, transform, and analyze** it using simple **SQL-like queries**.
- Then it sends the result to outputs like **Power BI**, **Blob Storage**, or **SQL Database**.

🔁 Think of it like:

Data comes from Event Hub → Stream Analytics processes it → Sends results to your chosen output.

🛠️ 5. Creating a Stream Analytics Job

To use ASA, you create a **job**.

💡 A Job is like a mini-program that:

1. Takes **input** from a source (e.g., Event Hub)
2. Runs a **query** on the data

3. Sends **output** to a destination (e.g., Power BI)

💡 In Azure Portal:

- Click "Create Stream Analytics Job"
- Set name, location, and resource group
- After creation, configure input, output, and query

📥 6. Defining the Input

Input = Where the job reads data from.

You define:

- **Input alias**: A nickname like inputhub
- **Source type**: Choose from Event Hub, Blob Storage, IoT Hub, etc.
- **Connection info**: Event Hub name, consumer group, authentication key

🟢 Example: Connect to an Event Hub that receives temperature data.

📤 7. Defining the Output

Output = Where the processed data will go.

You can send results to:

- **Power BI** – for dashboards
- **Azure Blob Storage** – for storing in files
- **Azure SQL Database** – to insert data in tables
- **Cosmos DB**, **Service Bus**, and more

🟢 Example: Send results to a Power BI dashboard to show live charts.

🔍 8. Defining the Query

Query = Logic you write to tell ASA how to process the data.

Written using SQL-like language:

```sql
SELECT deviceId, temperature
INTO outputblob
FROM inputhub
WHERE temperature > 80
```

You can:

- Filter (WHERE)
- Calculate averages, counts (AVG, COUNT)
- Use time-based windows (TumblingWindow, HoppingWindow)
- Group data (GROUP BY)

🧠 The power of ASA lies in this query.

💰 9. Costing for Streaming Services

Azure charges based on:

- **Data volume** (how much data you send)
- **Throughput units** (capacity for processing)
- **Running time** (how long your job runs)

◇ Stream Analytics: Charged per streaming unit per hour
◇ Event Hubs: Charged per **Throughput Unit**, data ingress (in) & egress (out), and retention

---

🆚 10. Batch vs Real-Time Processing

Feature	Batch Processing	Real-Time (Streaming) Processing
📦 Data	Collected over time (e.g., daily, weekly)	Processed instantly as it arrives
⏱️ Speed	Slower, after data is collected	Fast, real-time
🔧 Use Case	Payroll, monthly reports	Fraud detection, live analytics

Feature	Batch Processing	Real-Time (Streaming) Processing
⚙️ Tools	Azure Data Factory, Databricks	Azure Event Hubs + Stream Analytics

☑ **Final Summary**

What We Learned	Key Takeaway
Azure Event Hubs	Collects real-time data at large scale
Creating Event Hub	Set up a namespace and event hub
Sending Data	Use SDK or script to send data to Event Hub
Azure Stream Analytics	Processes data in real-time using SQL queries
Creating ASA Job	Define input, output, and query
Input & Output	Connect to sources and destinations
Writing Query	Use SQL to filter, group, and process data
Costing	Based on data volume and processing units
Batch vs Streaming	Streaming is fast, continuous; batch is delayed

What is a Blob?

- **Blob** stands for **Binary Large Object**.
- In **Azure Blob Storage**, you store big files like:
    - Documents
    - Images
    - Videos
    - Logs
    - CSV or JSON data files

Think of Blob Storage as a **cloud-based hard drive** where you store your files.

What does “Reading Blob Data” mean?

It simply means:

Accessing the file stored in Azure Blob Storage and getting its contents to use in your app, tool, or analytics job.

How do we read blob data?

You can read blob data using different tools or methods:

1. Azure Portal (Manual)

- Go to your Storage Account → Blob Container
- Click on a blob file (like a .csv)
- Click **"Download"** or **"View"** to read its contents

---

3. Azure Stream Analytics (Live Processing)

You can **connect Blob Storage as an input** in Stream Analytics to **read new data as it comes**, like logs or CSVs.

You define:

- Blob path (e.g., /container/*.csv)
- Date format in file name (optional)

- Start time for processing

🧠 Why read blob data?

You read blob data to:

- Analyze logs
- Process large data files (CSV, JSON)
- Show data in your application
- Use it in **data pipelines** or **real-time analytics**

🔑 Summary

Concept	Meaning
Blob Storage	Cloud place to store files
Read Blob	Getting file contents to use in app or job
Tools	Azure Portal, SDKs (Python, C#, etc.), Stream Analytics
Use Cases	Data analysis, dashboard, automation, ML

Sending Diagnostic log data

🩺 What are Diagnostic Logs?

Diagnostic logs are **logs created by Azure services** that help you monitor and troubleshoot what's happening.

📝 These logs can include:

- Who accessed a resource?
- What operations were performed?
- Were there any errors?

- How long did something take?

For example:

- A web app log might show how many users visited.
- A storage account log might show file upload/download attempts.

🚀 What does "Sending Diagnostic Log Data" mean?

It means:

Automatically forwarding logs from Azure services to a place where you can store, view, or analyze them.

🔧 Where Can You Send Diagnostic Logs?

You can send them to:

1. **Log Analytics** (to analyze with queries)
2. **Storage Account** (to store for long-term)
3. **Event Hub** (to stream logs to other tools)

🔄 Real-World Analogy

Imagine your house has CCTV cameras:

- **Diagnostic Logs** = The footage or logs the cameras record.
- You might:
    - Store it on a hard drive (like **Blob Storage**)
    - Send it to a monitor system (like **Log Analytics**)
    - Stream it live to a control room (like **Event Hub**)

🛠️ Steps to Send Diagnostic Log Data

1. **Go to the Azure resource** (e.g., a storage account or a virtual machine)
2. Click on **"Diagnostic settings"**
3. Click **"+ Add diagnostic setting"**

4. Choose:
    - What logs to send (e.g., audit logs, metrics)
    - Where to send (Log Analytics / Storage / Event Hub)
5. Save the setting

From now on, Azure will **automatically send logs** to the selected destination.

🎯 **Why is This Useful?**

- **Troubleshooting**: Helps you find and fix errors
- **Security Monitoring**: See who accessed what and when
- **Performance Tuning**: Know which operations are slow
- **Auditing & Compliance**: Keep logs for legal requirements

---

☑ **Summary**

Term	Meaning
Diagnostic Logs	Detailed logs from Azure services
Sending Logs	Forwarding logs to another service
Destinations	Log Analytics, Storage Account, Event Hub
Use Cases	Monitoring, security, performance, auditing
Setup	Done via **Diagnostic settings** in portal

🧠 What Does "Formulating a Query" Mean?

It means:

Writing a set of instructions (like a question) to tell the system what data you want and how you want to process it.

You use a language **similar to SQL** (Structured Query Language).

💡 Why Do We Need a Query?

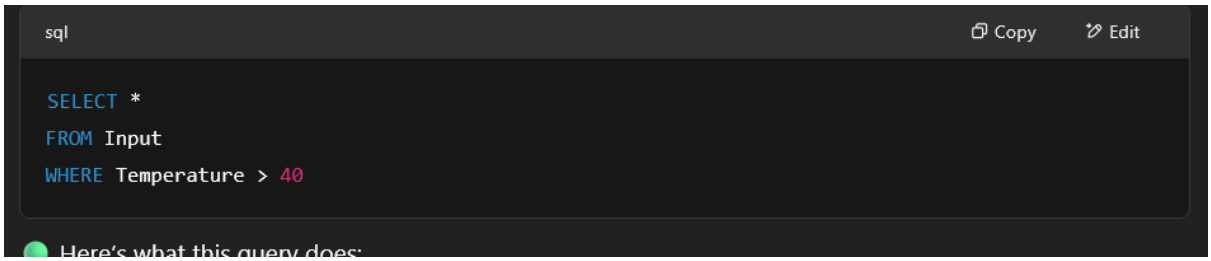
When data is flowing into your system (like from **Event Hub**, **IoT devices**, or **Blob files**), you need to:

- Pick only useful data
- Filter out unwanted data
- Group data (e.g., by time or location)
- Apply logic (like sum, count, average)
- Send it to output (like Power BI or a database)

You do this with a query.

📝 Simple Example

Let’s say you're getting temperature data from sensors and you only want readings above 40°C.



🔍 Here’s what this query does:

- SELECT * → Pick all columns
- FROM Input → Use data coming from a stream named Input
- WHERE Temperature > 40 → Filter and only keep high temperatures

---

📦 Common Query Tasks

Task	Example SQL Query
Filtering	WHERE City = 'Agra'
Selecting fields	SELECT DeviceId, Temperature
Aggregation	SELECT COUNT(*) or AVG(Temperature)
Grouping by time	GROUP BY TumblingWindow(minute, 5)
Joining streams	JOIN two inputs to compare or combine their data
Sending to output	The **result of this query** goes to your selected output

Azure Event Hubs Capture

🌟 What is Azure Event Hubs?

- **Event Hub** is like a big **data pipeline** that collects a lot of data from different sources in real-time.
- It's used to handle data like:
    - Sensor readings
    - App logs
    - Website clicks
    - IoT messages

📻 What is "Capture" in Event Hubs?

Event Hubs Capture is a **built-in feature** that:

Automatically saves the data received by Event Hub to a storage account (like Azure Blob or Data Lake).

So instead of writing custom code to store all incoming data, **Capture handles it for you**.

🎯 Why Use Capture?

- ☑ **Save raw data** for later use (e.g., reporting, analysis, ML)
- ☑ **Replay data** for debugging or reprocessing
- ☑ Automatically organize data into **folders and files** by time

---

🧱 How It Works – Simple Steps

1. Your devices or apps send data to **Event Hub**.
2. You enable **Capture** and link it to a **Storage Account** or **Data Lake**.
3. Capture will **automatically save incoming data** in files every few minutes or after a certain size (e.g., every 5 minutes or 100 MB).
4. These files can be **read later** by tools like:
    - Azure Stream Analytics
    - Data Factory
    - Azure Synapse
    - Databricks
    - Power BI

🗂 What Do the Saved Files Look Like?

- Files are saved in **Avro format** (a compressed data format good for big data).
- The folder path usually looks like this:

swift Copy Edit

```
/<eventhub-namespace>/<eventhub-name>/YYYY/MM/DD/HH/mm/filename.avro
```

Example:

swift Copy Edit

```
/myeventhubs/myhub/2025/05/25/15/30/data.avro
```

This tells us the file contains data from May 25, 2025 at 3:30 PM.

🔧 How to Enable Capture?

1. Go to your **Event Hub** in the Azure Portal.
2. Click on **"Capture"** from the left menu.
3. Turn it **On**.
4. Choose:
    - A **Storage Account or Data Lake**
    - The **capture frequency** (time/size)
5. Save settings.

🧠 Summary

Feature	What It Does
Event Hubs Capture	Auto-saves incoming data to storage
Format	Avro (used for big data processing)
Destinations	Azure Blob Storage or Data Lake
Benefits	Easy storage, replay data, process later
Setup	Just enable it in Azure Portal

What Does "Debugging Your Job" Mean?

Debugging a job means:

Finding and fixing mistakes in your Stream Analytics job so that it works correctly.

Just like checking your school project for spelling or logic errors, you check your job to:

- See if the **query is correct**

- Check if **data is coming in properly**
- Ensure that the **output is what you expect**

🧠 Why Is Debugging Needed?

Because sometimes:

- Your **query might have a small mistake**
- The **data format is wrong or missing**
- The **output doesn’t look right**
- The **job fails to run properly**

Debugging helps you **find out what's wrong and fix it quickly**.

🪟 What Are Windowing Functions

Windowing functions are used to **group streaming data by time**.

Imagine you’re watching a live security camera. The video never stops, right?
Now, imagine you want to:

- Count how many people enter **every 5 minutes**
- Or check the average temperature **every hour**

To do that, you need to **divide the never-ending stream of data into small "windows"** of time.

Windowing = Breaking the stream into small time periods so you can do calculations on each part.

🧠 Why Do We Use Windowing?

Real-time data is always coming, like:

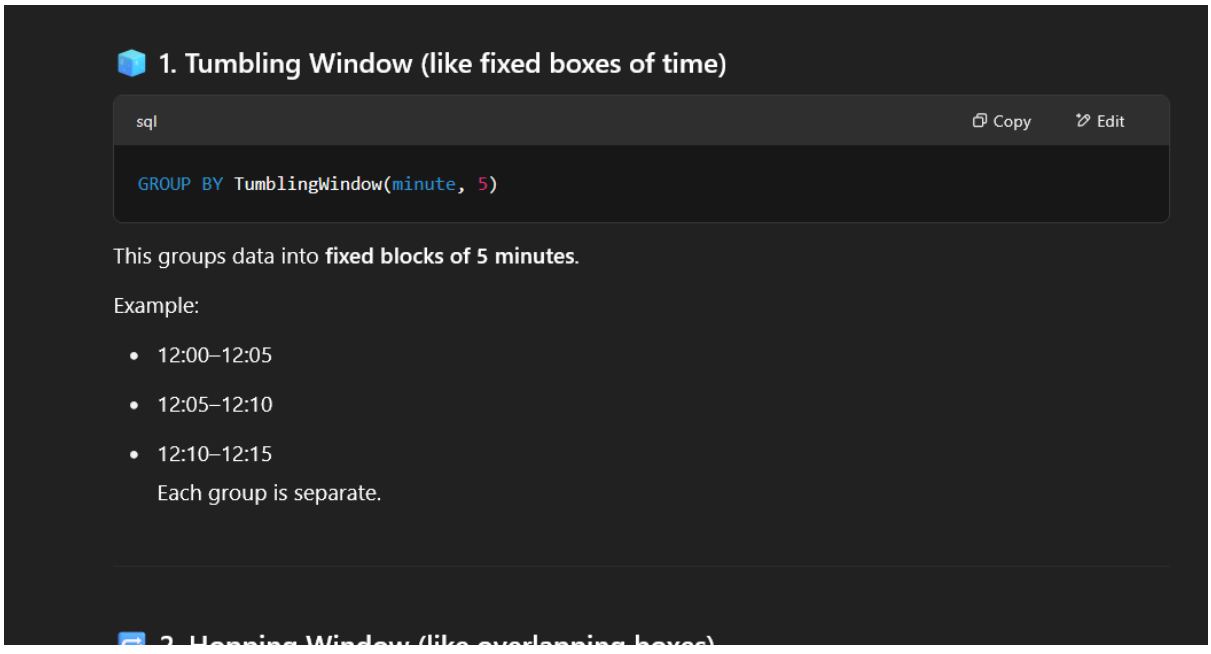
- Sensor readings
- App clicks
- Payment logs

We use **windows** to:

- Count, sum, or average values **in time ranges** (like 1 minute, 5 minutes)
- Make data easier to understand
- Create meaningful insights

📦 Types of Windows (in Azure Stream Analytics)

Window Type	What It Does
Tumbling Window	Breaks time into **fixed, non-overlapping chunks**. Like: 0–5 mins, 5–10 mins
Hopping Window	Windows **overlap**. Like every 5 mins, but each window is 10 mins long
Sliding Window	A window **moves by every event**. Good for detecting sudden spikes
Session Window	Based on **activity gaps** (e.g., no data for 5 mins means session ended)



1. Tumbling Window (like fixed boxes of time)

```sql
GROUP BY TumblingWindow(minute, 5)
```

This groups data into **fixed blocks of 5 minutes.**

Example:

- 12:00–12:05
- 12:05–12:10
- 12:10–12:15
  Each group is separate.

---

2. Hopping Window (like overlapping boxes)

```sql
GROUP BY HoppingWindow(minute, 5, 2)
```

This groups every **5 minutes**, but with a **hop of 2 minutes** — so windows overlap.

Example:

- 12:00–12:05
- 12:02–12:07
- 12:04–12:09

This is useful when you want **more frequent updates** with some **overlap.**

What is a Tumbling Window? (Quick Recap)

A **Tumbling Window** breaks a continuous stream of data into **equal, non-overlapping chunks** of time.

Example:
If you use a **5-minute tumbling window**, it will group data like this:

- 12:00 – 12:05
- 12:05 – 12:10

- 12:10 – 12:15
  Each group is **separate** and does **not overlap**.

☑ Why Use It?

You use a **Tumbling Window** when you want to:

- Count events (like logins) every 5 minutes
- Get average temperature every hour
- Track total sales every 10 minutes

❁ How to Set Up a Tumbling Window in Azure Stream Analytics

Step 1: Create Your Stream Analytics Job

(You may have already done this.)

Step 2: Define Your Input

Connect to data source like:

- Event Hub
- IoT Hub
- Blob Storage

This is where your **streaming data** is coming from.

Step 3: Define Your Output

Choose where the results will go, like:

- Power BI
- Blob Storage
- SQL Database

Step 4: Write a Query Using Tumbling Window

Here's the **main part**: your **SQL-like query**.

What Does “Tumbling Window Implementation” Mean?

It means you're going to **set up and run** a tumbling window to process live streaming data in **fixed time blocks** like:

- Every 5 minutes
- Every 1 hour
- Every 10 seconds

You’ll use this to:

- Count things
- Calculate average, max, min
- Group events that happen in each time block

🛠️ Steps to Implement Tumbling Window in Azure

◇ Step 1: Set Up a Stream Analytics Job

This is where all the work happens.

1. Go to Azure portal
2. Click on **“Stream Analytics Job”**
3. Give it a name and resource group
4. Choose location and click **Create**

◇ Step 2: Add Input

This is where your streaming data comes from (for example, an Event Hub).

1. In your job, click **“Inputs”**
2. Click **“+ Add stream input” → Event Hub (or IoT Hub, Blob, etc.)**
3. Fill in Event Hub details (name, namespace, etc.)
4. Click **Save**

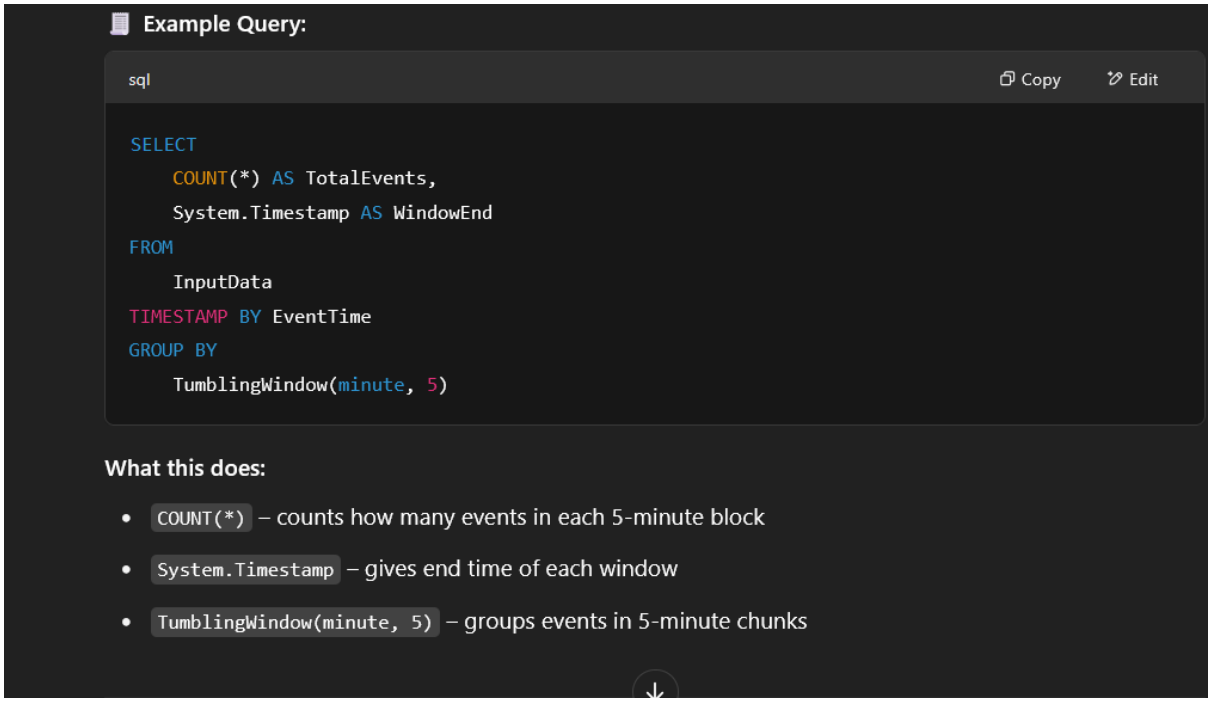
◇ Step 3: Add Output

This is where you want the result to go (for example, a Blob Storage, Power BI, or SQL Database).

1. Click **“Outputs”**
2. Click **“+ Add”**
3. Choose the type (e.g., Blob storage)
4. Enter storage account details and container name
5. Click **Save**

◇ Step 4: Write the Query with Tumbling Window

Now comes the main part – writing the query to process the data using a tumbling window.



What this does:

- COUNT(*) – counts how many events in each 5-minute block
- System.Timestamp – gives end time of each window
- TumblingWindow(minute, 5) – groups events in 5-minute chunks

---

◇ Step 5: Start the Job

Once everything is ready:

1. Click **“Start”** at the top of the job
2. Choose the **input start time** (Now or Custom)
3. Click **Start**

Now your Stream Analytics job will start collecting and processing data in 5-minute blocks.

Having multiple queries in the job

🧠 What does it mean?

Normally, in an Azure Stream Analytics job, you write one SQL-like query to process your streaming data.

But Azure also lets you write more than one query in the same job — each query can:

- Do something different
- Work on the same or different data
- Send results to different outputs

This is called having multiple queries in one job.

🎯 Why Use Multiple Queries?

Imagine you have a stream of live temperature data. You can:

- Use Query 1 to calculate average temperature every 5 minutes
- Use Query 2 to detect high temperature alerts
- Use Query 3 to store raw data in Blob Storage

Each query works independently inside the same job.

🔍 What Are Windowing Functions?

Windowing functions help you group real-time data into time-based chunks so you can:

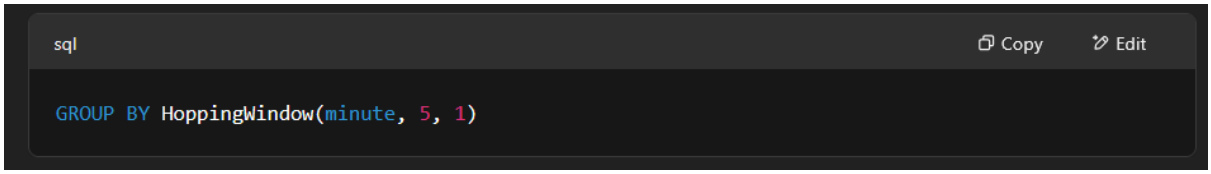
- Count events
- Calculate averages
- Detect patterns
  over a specific period.

🔲 Other Window Types (Besides Tumbling Window)

1. Hopping Window

- Think of it like: A moving window that overlaps.
- Example: A window of 5 minutes that moves every 1 minute.
- So you'll get results every 1 minute for the last 5 minutes.

🧠 Use when: You want to catch changes or trends in a rolling time period.



2. Sliding Window

- **Think of it like:** A window that only gives results **when new data comes in**.
- It **moves with the data**, not on a fixed time schedule.

🧠 **Use when:** You want **event-driven** calculations.

3. Session Window

- **Think of it like:** A window that **groups data when it's coming in close together** (no long gaps).
- If there is **no data** for a certain time (e.g., 10 seconds), the window **closes**.

```sql
GROUP BY SessionWindow(second, 10)
```

☑ What is Reference Data?

Reference data is **static or slow-changing data** that you use to **enrich** your live streaming data.

You can think of it like a **lookup table** that helps give **context** or **meaning** to the streaming data.

📦 Example in Real Life

Let's say:

- You have **real-time streaming data** from temperature sensors.
- Each sensor only sends a **sensor ID**.

Now, you have another table (reference data) like:

SensorID	Location
101	Delhi
102	Mumbai

You use **reference data** to **join** with the streaming data, so you can say:

"Sensor 101 is in Delhi and its temperature is 38°C"

🧠 Why Use Reference Data?

To **add more details** to streaming events using **joins**.

Streaming Data	Reference Data	Enriched Result
Sensor ID + Temperature	Sensor ID + Location	Sensor ID + Temperature + Location

💡 Where Is Reference Data Stored?

You upload reference data in **Blob storage** (as a CSV file) and tell Azure Stream Analytics to use it.

🔐 What is a Network Security Group (NSG)?

An **NSG** in Azure is like a **firewall**. It controls:

- What kind of traffic (data) can **come in** or **go out** of your server or virtual machine (VM).
- Based on **rules** (like allow or deny specific ports, IPs, etc.)

NSG logs help you **track**:

- What traffic was allowed or denied.
- When and from where it happened.

📖 What Do We Mean by "Reading NSG Logs"?

It means:

- **Collecting and viewing** the log data created by the NSG.
- So you can **analyze**, **monitor**, or **troubleshoot** network issues.

👨‍💻 Server Setup to Read NSG Logs – Step-by-Step

☑️ Step 1: Enable Diagnostic Logs for NSG

You need to **turn on logging** for your Network Security Group:

- Go to your NSG in the Azure portal.
- Enable **"Diagnostic settings"**.
- Choose to send logs to **Storage Account**, **Log Analytics**, or **Event Hub**.

👉 Most common: Send to **Storage Account** (cheaper) or **Log Analytics** (easier to query).

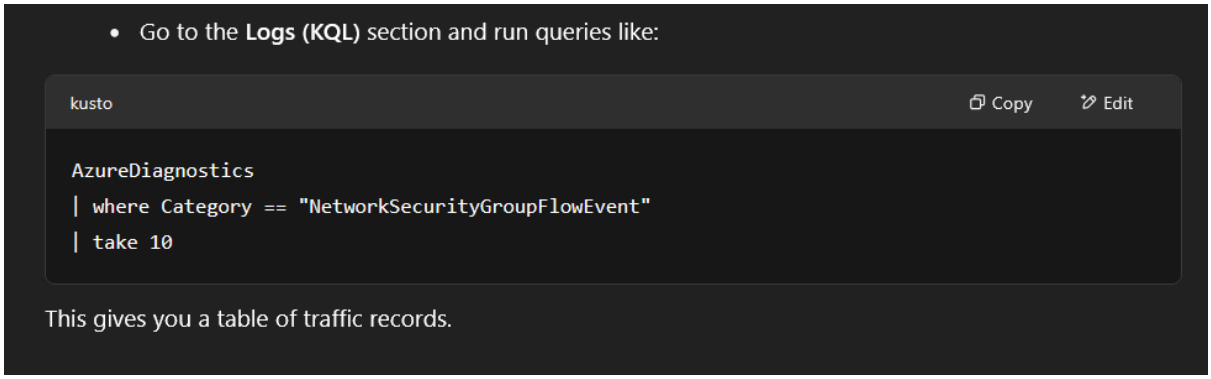
☑️ Step 2: Choose Where to Store the Logs

- **Storage Account**: Saves logs in a folder format (as JSON files).
- **Log Analytics**: Lets you run queries to search logs.
- **Event Hub**: Used when you're streaming logs to other tools (like SIEM tools or Stream Analytics).

☑ Step 3: Access the Logs

Depending on where you stored them:

- If in **Storage Account**:
    - Use tools like **Azure Storage Explorer** to download or view logs.
    - Or use a **Stream Analytics job** to process them live.
- If in **Log Analytics**:



🔍 What's Inside the NSG Log?

Each log entry can tell you:

- Source and destination IP
- Port numbers
- Protocol (TCP/UDP)
- Whether it was **allowed or denied**
- VM name and location

📊 Why Is This Important?

- To **monitor attacks** (blocked traffic)

- To **debug connectivity issues**
- To **audit and improve security rules**

Reading Network Security Group Logs - Enabling NSG Flow Logs

What are NSG Flow Logs?

- NSG = **Network Security Group** (works like a **firewall** for your virtual machines).
- **Flow Logs** = These logs record the **traffic (data)** that is either **allowed** or **denied** by the firewall.

Think of it like a **CCTV camera for your network** – it watches what traffic comes in and out.

🎯 **Why Enable NSG Flow Logs?**

To:

- **Track** what's happening with your server/network traffic
- **Find problems** (like blocked traffic)
- **Improve security** by analyzing unexpected access

☸️ **How to Enable NSG Flow Logs — Step-by-Step**

☑️ **Step 1: Go to Your NSG**

- Open the **Azure Portal**
- Search for **"Network Security Groups"**
- Click on the NSG you want to monitor

☑️ **Step 2: Open Diagnostic Settings**

- On the left side, click **"Diagnostics settings"**
- Click **"+ Add diagnostic setting"**

☑️ **Step 3: Choose What to Log**

- **Check the box** for **"NetworkSecurityGroupFlowEvent"**
  (This is the actual flow log data)

☑ **Step 4: Choose Where to Store the Logs**

Pick one or more of the following:

- **Storage Account** – stores raw files (best for long-term)
- **Log Analytics** – you can run queries on logs (easy to analyze)
- **Event Hub** – useful if sending logs to other systems

☑ **Step 5: Save the Settings**

- Click **Save**
- Now Azure will start **recording flow logs** for that NSG

1. Understanding the NSG Flow Log Structure

🔍 **What it means:**

- NSG Flow Logs are **JSON files**.
- They contain **details about each connection attempt**, like:
  - **Source IP address** (where the request came from)
  - **Destination IP address** (where it was going)
  - **Port number**
  - **Protocol** (TCP/UDP)
  - **Action** – was it **allowed** or **denied**?

📦 Example:

json Copy Edit

```
{
  "srcIP": "10.0.0.5",
  "destIP": "10.0.0.7",
  "srcPort": 12345,
  "destPort": 80,
  "action": "A"
}
```

2. Starting with the Query

💡 **What it means:**

- You will start writing a **basic query** using **Kusto Query Language (KQL)** in **Log Analytics** to search flow logs.

📓 Example:

kusto Copy Edit

```
AzureDiagnostics
| where Category == "NetworkSecurityGroupFlowEvent"
```

This query filters the logs to **only show NSG Flow Events**.

3. Formulating Our Query

🛠️ **What it means:**

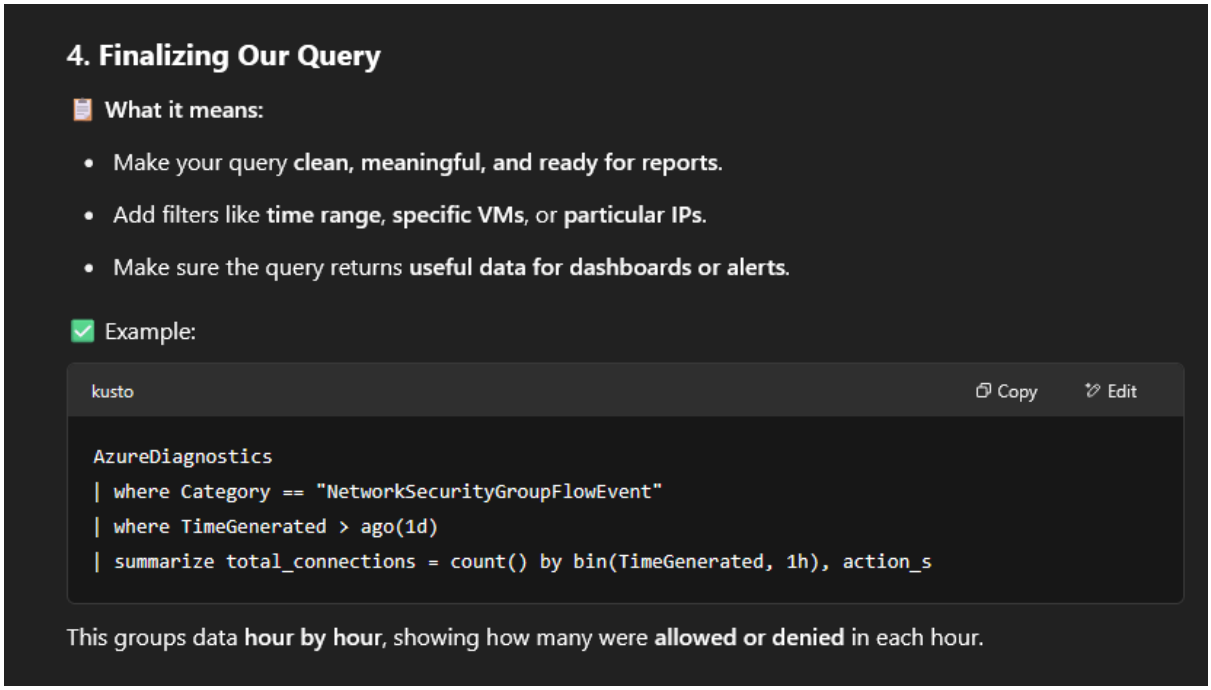
- Add more filters and logic to get **useful information**.
- You can **group by IPs, count allowed vs denied**, or **track specific ports**.

📌 Example:

kusto Copy Edit

```
AzureDiagnostics
| where Category == "NetworkSecurityGroupFlowEvent"
| summarize count() by action_s
```

This tells you **how many requests were allowed or denied**.



Azure data bricks

🧠 **Simple Explanation:**

Azure Databricks is a tool on Microsoft Azure that helps you analyze big data (huge amounts of data) and build machine learning models.
It gives you a place to write code in Python, SQL, Scala, or R and process data quickly and easily.

Think of it like a smart notebook in the cloud where:

- **You write your data code (like Python).**
- **It processes your data very fast (even if it's massive).**
- **You can collaborate with your team in real time.**

🔍 **Detailed Explanation:**

☑ 1. What is it Built On?

Azure Databricks is built on Apache Spark, a powerful open-source engine used for:

- **Big data processing**
- **Real-time analytics**
- **Machine learning**

Azure combines Spark with easy-to-use features, security, and integration with other Azure services.

☑ 2. Main Features of Azure Databricks:

◇ Feature	🕮 Meaning
Collaborative notebooks	Code + text + results in one place (like Jupyter Notebooks). Great for team work.
Auto-scaling clusters	Adds/removes compute power depending on your data workload. Saves cost.
Multi-language support	Write code in Python, SQL, Scala, or R. All in one notebook.
Built-in ML tools	Supports machine learning libraries (like scikit-learn, TensorFlow).
Connects with Azure	Works well with services like Azure Data Lake, Event Hubs, Blob Storage, and Power BI.

Building a solution architecture for a data engineering solution using Azure Databricks, Azure Data Lake Gen2, Azure Data Factory and Power BI

Sure! Let's break this down **step by step** in **easy words**, and then go through how **all the tools work together** to build a **data engineering solution architecture** using:

- ☑ Azure Databricks
- ☑ Azure Data Lake Storage Gen2
- ☑ Azure Data Factory
- ☑ Power BI

🧱 What Is a Solution Architecture?

A **solution architecture** is like a **blueprint** or **plan** for how your **data** will flow across different services, from where it is collected to where it is analyzed and visualized.

🛠️ Tools Overview (Simple Definitions):

Tool	What It Does (Simple Explanation)
Azure Data Factory	A tool to **move and schedule** data from one place to another. (Think of it like a delivery guy)
Azure Data Lake Gen2	A **storage system** to hold huge amounts of data. (Like a giant cloud hard disk)
Azure Databricks	A place to **clean, process, and analyze data**. (Think of it as a smart kitchen that cooks your data)
Power BI	A tool to **visualize the final data** in charts, graphs, and dashboards. (Like a beautiful report card)

🧭 Step-by-Step Solution Architecture Flow:

Here's how all these tools work **together** to create a data pipeline.

◇ 1. Data Ingestion (Data Factory)

- You use **Azure Data Factory** to **bring raw data** from different sources:

    - Excel/CSV files
    - Databases
    - APIs
- Data Factory **moves this raw data** and **stores it into** Azure Data Lake Gen2.

📦 **Think of Data Factory as the truck that picks up your groceries (data) and delivers it to your storage (Data Lake).**

◇ 2. Data Storage (Azure Data Lake Gen2)

- The raw data is stored in folders (containers) in Azure Data Lake.
- You can organize it as:
    - **Raw zone**: untouched data
    - **Clean zone**: cleaned/transformed data
    - **Curated zone**: ready-to-use data

🗃️ It's like your fridge, pantry, and kitchen shelf. You store ingredients here for later cooking (processing).

◇ 3. Data Processing (Azure Databricks)

- **Azure Databricks** connects to the Data Lake.
- It **reads raw data**, **cleans it**, **removes errors**, **joins tables**, and **transforms it**.
- You can write code in **Python**, **SQL**, or **Scala**.
- Final output can be saved back into Data Lake (in the curated zone).

👨‍🍳 This is like cooking: Databricks takes raw data (ingredients) and turns it into a delicious meal (processed data).

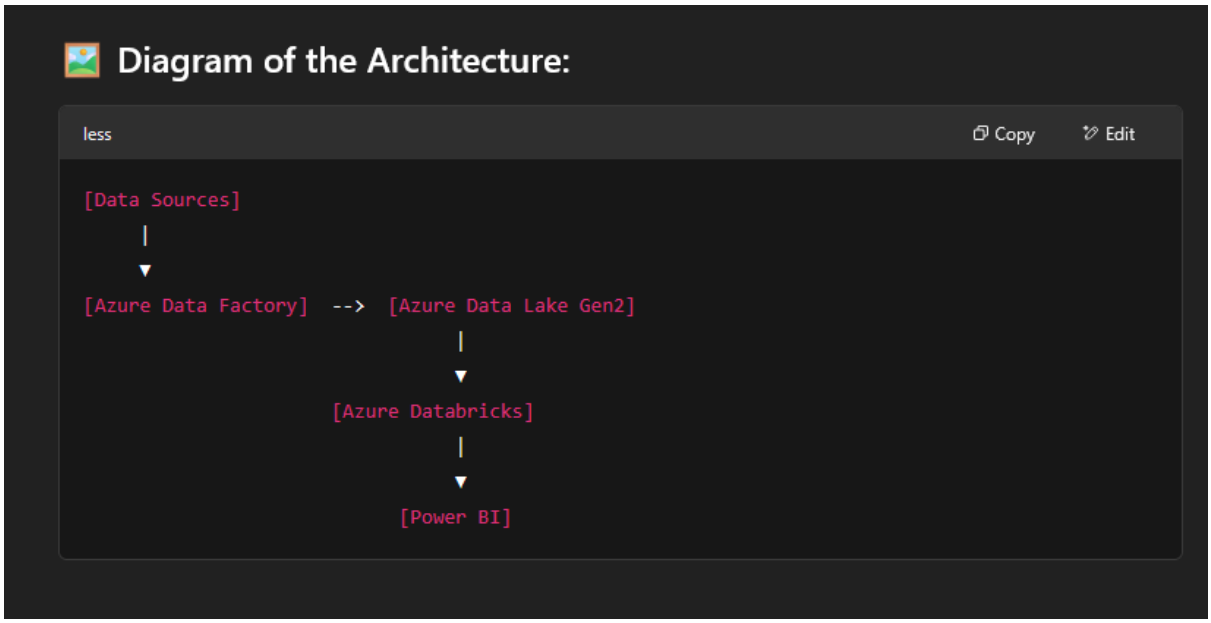
◇ 4. Data Visualization (Power BI)

- Power BI connects to the **processed data** in Azure Data Lake or Databricks.
- You use Power BI to:
    - Create dashboards

- Build charts
- Share reports with others

📊 This is the final presentation — your clean, tasty dish served beautifully on a plate!

🖼️ **Diagram of the Architecture:**



💡 **Real-Life Example:**

Let's say you work at an eCommerce company:

- **Data Factory pulls daily order data from an online system.**
- **Data Lake stores the raw order data.**
- **Databricks cleans the data, removes duplicates, calculates total revenue, top-selling products.**
- **Power BI shows dashboards for the sales team to track performance.**

☑️ **Summary**

Step	Tool	What It Does
1	Azure Data Factory	Moves data from source to storage

Step	Tool	What It Does
2	Azure Data Lake Gen2	Stores raw and processed data
3	Azure Databricks	Cleans, transforms, and analyzes data
4	Power BI	Visualizes the data for decision-making