BookMyShow Real-Time Data Pipeline

1. Project Overview

This project builds a **real-time data pipeline** for **streaming booking and payment data** in an event-driven manner. Using **Azure Event Hub, Stream Analytics, and Synapse**, the pipeline processes incoming events and transforms them for storage and analysis.

Key Objectives:

- Capture booking and payment data in real-time.
- Transform and enrich event data using **Azure Stream Analytics**.
- Store structured event data in **Azure Synapse Analytics**.
- Enable **real-time reporting and insights** for event-based transactions.

2. Architecture & Components

The project consists of the following key Azure services:

1. Event Hubs (Message Streaming Layer)

- **Created two Event Hub topics:**
  - bookmyshowbookingtopic → Receives booking data.

    - bookmyshowpaymenttopic → Receives payment data.
- These topics act as real-time event brokers where incoming booking and payment events are published.

2. Azure Stream Analytics (Data Processing Layer)

- A **Stream Analytics job** is created to consume data from both Event Hub topics.
- The job transforms and joins booking and payment streams based on order_id.
- The processed data is **stored in an Azure Synapse Analytics table** for analysis.

3. Azure Synapse Analytics (Storage & Analysis Layer)

- A dedicated **schema bookymyshow** is created inside Synapse.
- The bookings_fact table stores **enriched booking and payment data**.
- SQL queries enable further **analytics and reporting**.

3. Data Flow Explanation

1. **Data Generation & Publishing:**
    - Python scripts are used to simulate booking and payment events.

    - Each event contains relevant details such as order ID, customer info, event details, seat details, and payment details.
    - The scripts publish events into the corresponding Event Hub topics.

2. **Real-time Processing in Stream Analytics:**
    - Stream Analytics Job reads data from both topics (bookingtopic and paymenttopic).
    - Transforms the data (casting timestamps, categorizing event types, extracting structured fields).
    - Joins booking and payment events based on order_id within a 2-minute window.
    - Sends processed data to Azure Synapse Analytics.

3. **Data Storage in Synapse:**
    - The transformed dataset is stored in the bookings_fact table in Synapse.
    - The schema includes customer details, event details, seat price, payment details, timestamps, and event categories.
    - Queries allow reporting, aggregation, and analysis on booking and payment trends.

4. SQL Queries & Schema in Synapse

- Schema Definition (bookings_fact table):

    - Contains columns for **order details, customer details, event details, seat details, payment details, and timestamps**.
- **Example Queries:**
    - SELECT COUNT(*) FROM bookymyshow.bookings_fact; → To check total processed records.
    - SELECT event_category, COUNT(*) FROM bookymyshow.bookings_fact GROUP BY event_category; → To analyze booking trends per category.

5. Python Scripts for Data Publishing

- **Booking Data Publisher:**
    - Uses Faker library to generate **random booking data**.
    - Sends booking events to **Event Hub (bookmyshowbookingtopic)**.
- **Payment Data Publisher:**
    - Generates **mock payment transactions**.
    - Sends payment events to **Event Hub (bookmyshowpaymenttopic)**.
- Both scripts run continuously, sending new events every 5 seconds.

6. Error Handling & Monitoring

- Event Hub Monitoring:
  - Azure **Monitor** and **Log Analytics** track message flow.
- Stream Analytics Debugging:
  - Errors are handled using **TRY_CAST()** for data type conversions.
  - **System.Timestamp** is used to track event processing time.
- Synapse Data Validation:
  - Periodic queries verify **data completeness and consistency**.

7. Conclusion:

Current Achievements:

✓ Successfully **streamed and processed** booking/payment data in real-time.

✓ Integrated **Azure Stream Analytics** for **event transformation and enrichment**.

✓ Stored structured data in **Azure Synapse** for reporting and analysis.