**Problem Statement:**

Imagine you are a data engineer working for AdvertiseX, a digital advertising technology company. AdvertiseX specializes in programmatic advertising and manages multiple online advertising campaigns for its clients. The company handles vast amounts of data generated by ad impressions, clicks, conversions, and more. Your role as a data engineer is to address the following challenges:

Data Sources and Formats:

Ad Impressions:

Data Source: AdvertiseX serves digital ads to various online platforms and websites.

Data Format: Ad impressions data is generated in JSON format, containing information such as ad creative ID, user ID, timestamp, and the website where the ad was displayed.

Clicks and Conversions:

Data Source: AdvertiseX tracks user interactions with ads, including clicks and conversions (e.g., sign-ups, purchases).

Data Format: Click and conversion data is logged in CSV format and includes event timestamps, user IDs, ad campaign IDs, and conversion type.

Bid Requests:

Data Source: AdvertiseX participates in real-time bidding (RTB) auctions to serve ads to users.

Data Format: Bid request data is received in a semi-structured format, mostly in Avro, and includes user information, auction details, and ad targeting criteria.

Case Study Requirements:

Data Ingestion:

Implement a scalable data ingestion system capable of collecting and processing ad impressions (JSON), clicks/conversions (CSV), and bid requests (Avro) data.

Ensure that the ingestion system can handle high data volumes generated in real-time and batch modes.

Data Processing:

Develop data transformation processes to standardize and enrich the data. Handle data validation, filtering, and deduplication.

Implement logic to correlate ad impressions with clicks and conversions to provide meaningful insights.

Data Storage and Query Performance:

Select an appropriate data storage solution for storing processed data efficiently, enabling fast querying for campaign performance analysis.

Optimize the storage system for analytical queries and aggregations of ad campaign data.

Error Handling and Monitoring:

Create an error handling and monitoring system to detect data anomalies, discrepancies, or delays.

Implement alerting mechanisms to address data quality issues in real-time, ensuring that discrepancies are resolved promptly to maintain ad campaign effectiveness.

**Solution:**

We can make use of multiple open source tools to achieve the required goal like,

* Apache Kafka: For real-time data streaming.
* Apache Flink: For real-time data processing.
* Apache NiFi: For batch ingestion and initial data transformations.
* Amazon S3 / ADLS Gen2 : For data lake storage.
* Apache Hudi / Delta Lake: For handling upserts and time-travel data.

I propose a solution to accommodate the tools in one single cloud provider for easier handling. Azure provides a comprehensive suite of services that can be leveraged to build a robust data engineering pipeline.

**1. Data Ingestion**

For data ingestion, we can use Azure Event Hubs for real-time data streaming and Azure Data Factory for batch data ingestion.

**Components:**

* **Azure Event Hubs**: For real-time data streaming.
* **Azure Data Factory (ADF)**: For orchestrating batch data ingestion and initial transformations.
* **Azure Blob Storage**: For storing raw data.
* **Azure Data Lake Storage (ADLS)**: For storing processed data in a structured format.

**Implementation:**

**Azure Event Hubs Setup:**

* Create Event Hubs for each data source:
  + **ad-impressions**
  + **clicks**
  + **conversions**
  + **bid-requests**

**Azure Data Factory Setup:**

* Create ADF pipelines to ingest CSV and JSON data into Event Hubs.
* Use ADF activities like **Copy Data**, **Data Flow**, and **Mapping Data Flows** to handle file ingestion and data format conversions.

**2. Data Processing**

For data transformation, we'll use Azure Databricks, which provides a unified analytics platform that integrates well with Azure services.

**Azure Databricks Setup:**

* Set up Databricks clusters to process data from Event Hubs and ADLS.
* Implement data validation, filtering, and deduplication using Databricks notebooks.

A small snippet of databricks notebook:

from pyspark.sql.functions import col, from\_json, when

from pyspark.sql.types import StructType, StructField, StringType, TimestampType

# Define schema for ad impressions

schema = StructType([

StructField("ad\_creative\_id", StringType(), True),

StructField("user\_id", StringType(), True),

StructField("timestamp", TimestampType(), True),

StructField("website", StringType(), True)

])

# Read data from Event Hubs

ad\_impressions = spark.readStream.format("eventhubs").options(\*\*event\_hub\_config).load()

ad\_impressions\_df = ad\_impressions.select(from\_json(col("body").cast("string"), schema).alias("data")).select("data.\*")

# Data validation and filtering

valid\_ad\_impressions = ad\_impressions\_df.filter(col("ad\_creative\_id").isNotNull() & col("user\_id").isNotNull())

# Write to Azure Data Lake Storage

valid\_ad\_impressions.writeStream.format("parquet").option("path", "adl://path/to/ad-impressions").option("checkpointLocation", "adl://path/to/checkpoints").start()

**3. Data Storage and Query Performance**

For storing processed data, we can use Azure Data Lake Storage (ADLS) for raw and processed data, and Azure Synapse Analytics for fast querying.

**Data Storage (Azure Data Lake Storage and Azure Synapse Analytics):**

* Store raw and processed data in ADLS using Parquet format.
* Load processed data into Azure Synapse Analytics for fast analytical queries.
* Use Delta Lake for handling upserts and time-travel queries in Databricks.

**Schema Design:**

* Design schemas to reflect the hierarchical structure of the data (e.g., fact tables for events, dimension tables for ad creatives, users).

**4. Error Handling and Monitoring**

Implementing a robust error handling and monitoring system using Azure Monitor and Azure Log Analytics.

**Error Handling:**

* Implement try-except blocks in Databricks notebooks to catch and log errors.

**Monitoring:**

* Set up monitoring and alerting with Azure Monitor and Azure Log Analytics.
* Use built-in monitoring tools to track data flows and detect anomalies.
* Implement custom metrics to track data processing health.