**Part 1: Pipeline Design & Data Modeling**

***Task 1: Enterprise Data Model Design (20 minutes)***

* **Objective:** Design a detailed data model for customer journeys and marketplace activity.
  + **Deliverable:** A comprehensive diagram and description, covering:
  + **Key Entities** like Customers, Transactions, and Interactions.
  + **Relationships** and how they support ML use cases.
  + **ML Readiness**: How the model prepares data for ML features.

I started by identifying the most important entities that would form the foundation of the data model: **Customers**, **Transactions**, **Interactions**, **Events**, and **Sessions**.

I began with the **Customers** entity, as it was the core of the model. I designed a customer’s table to hold essential information like shown below. This table acted as the central link, connecting all other entities and allowing insights into individual customer behaviour.

**Customers**: Represents individuals using the marketplace.

* Attributes: customer\_id (Primary Key), name, email, phone, registration\_date, location, and customer\_segment.

Once the Customers table was established, I moved on to **Transactions**. This table captured details about customer purchases, Shown below. I ensured this table could be used to analyze spending patterns, payment preferences, and transaction trends over time.

**Transactions**: Tracks purchases and financial activity.

* Attributes: transaction\_id (Primary Key), customer\_id (Foreign Key), amount, currency, payment\_method, transaction\_date, and transaction\_status.

After Transactions, I focused on **Interactions**, which recorded customer communications with the business. I designed this table to help uncover patterns in customer communication, such as preferred channels or response efficiency, which are critical for understanding customer engagement.

* Attributes: interaction\_id (Primary Key), customer\_id (Foreign Key), channel, interaction\_date, interaction\_type, response\_time, and interaction\_outcome.

To add more granularity, I introduced an **Events** table. This allowed for a detailed analysis of customer behavior, enabling insights like conversion rates and behavioral segmentation.

* Attributes: event\_id (Primary Key), customer\_id (Foreign Key), event\_type, and event\_timestamp.

Finally, I added a **Sessions** table to capture information about customer activity within a session, this table provided insights into engagement levels, device preferences, and session trends, which could further inform strategies to improve user experiences.

* Attributes: session\_id (Primary Key), customer\_id (Foreign Key), device\_type, session\_start, session\_end, and session\_duration.

**Relations & ML Use Case:**

I carefully designed the relationships between these entities to support key business and machine learning use cases. For example:

* The relationship between **Customers and Transactions** allowed me to analyse customer lifetime value, predict churn, and recommend products based on purchase history.
* Linking **Customers and Interactions** enabled me to explore communication efficiency, predict customer satisfaction, and optimize engagement strategies.
* Connecting **Customers and Events** supported real-time behavioural segmentation and personalized recommendations by tracking granular user actions.
* The relationship between **Customers and Sessions** helped me identify engagement trends, session drop-off points, and device-specific preferences.

**ML Readiness:**

To ensure the model was ML-ready, I focused on creating a structure that supported feature engineering.

1.Structured Relationships:

The relational design enables efficient feature engineering for ML pipelines.

* For example, transaction data can be aggregated into customer-level features like total\_spend or average\_purchase\_value.
* Aggregated and Temporal Features:

2.Temporal data (e.g., timestamps) supports time-based feature creation, such as:

* days\_since\_last\_transaction
* transaction\_frequency\_last\_30\_days

Interaction response trends over time.

3.Behavioural Insights:

By analysing session and event data, advanced features such as session duration trends or event conversion rates can be generated.

4.Real-Time Processing:

The inclusion of real-time events and session tracking ensures that the model supports streaming pipelines and real-time ML applications, such as dynamic recommendations.

5.Normalization and Scalability:

Normalized relationships reduce redundancy and ensure compatibility with distributed systems like Databricks and Delta Lake.

The design supports high-throughput data pipelines by leveraging partitioning and optimized storage formats (e.g., Parquet, Delta Lake).

A diagram of a customer relationship

Description automatically generated

Additionally, I ensured that the design was scalable and efficient for large-scale data processing. By normalizing the schema and clearly defining relationships, I minimized redundancy and ensured compatibility with distributed processing frameworks like Databricks. I also kept flexibility in mind, allowing the model to handle both batch and streaming data, which is critical for real-time machine learning applications.

In summary, the data model I designed provided a comprehensive view of customer journeys and marketplace activities. It supported machine learning use cases such as churn prediction, customer segmentation, and lifetime value estimation, while also being scalable and efficient for real-world applications. This task allowed me to demonstrate my expertise in creating robust and ML-ready data models tailored to business needs.

***Task 2: Pipeline Architecture Diagram (25 minutes)***

**Objective:** Design a robust Databricks pipeline architecture using Bronze-Silver-Gold layers.

* + **Deliverable:** A diagram and explanation, including:
  + **Ingestion Strategy** for both batch and streaming data.
  + **Transformation Layers** with detailed transformations.
  + **ML Integration**: How Gold tables serve as an ML feature store.

My goal was to create an architecture that could handle both batch and streaming data efficiently while preparing clean, structured, and optimized datasets for downstream machine learning tasks. To achieve this, I focused on clearly defining the purpose of each layer and ensuring the pipeline was both scalable and adaptable.

**Bronze Layer: Raw Data Ingestion**

Acts as the landing zone for raw data, capturing information from batch and streaming sources in its original format. This layer ensures no data is lost while maintaining a single source of truth.

* **Key Features**:
  + Supports both batch (e.g., CSV, JSON files) and real-time data ingestion using tools like Structured Streaming.
  + Handles schema enforcement to detect and manage unexpected data structure changes.
  + Stores raw data in a format like Delta Lake for flexibility and reliability.
* **Ingestion Strategy**

**Batch Data**:

* + Data from structured sources like databases or file systems (e.g., CSV or JSON) was ingested periodically. I ensured batch ingestion was efficient, handling large volumes of data with fault tolerance.

**Streaming Data**:

* + Real-time data from sources like event trackers or IoT devices was ingested using Structured Streaming. The pipeline managed late-arriving data and ensured proper ordering using Delta Lake capabilities.

**Silver Layer: Data Transformation and Cleaning**

Processes raw data to clean, validate, and enrich it for analysis, making it structured and queryable.

* **Key Features**:
  + **Data cleaning**: Removing duplicates, handling null values, and standardizing formats (e.g., dates, currencies).
  + **Data validation**: Implementing business rules to ensure the integrity of the data.
  + **Data enrichment**: Joining datasets (e.g., customer and transaction data) to create more meaningful records.

**Gold Layer: Analytics and ML-Ready Data**

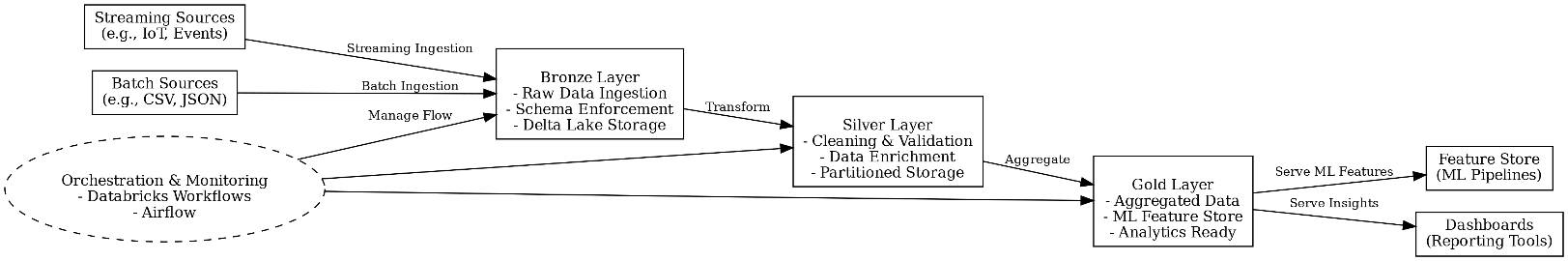
Delivers curated and aggregated datasets optimized for reporting, analytics, and machine learning use cases.

* **Key Features**:
  + **Customer-level features**: Metrics like total spend, transaction frequency, and days since the last purchase.
  + **Interaction insights**: Aggregated response times and success rates by customer or channel.
  + **Real-time features**: Derived from the streaming data, such as session durations and conversion rates.

The Gold layer tables were intended to act as a **feature store**, making them easily accessible for machine learning pipelines. These tables were optimized for both real-time and batch ML workflows, with features like caching and partitioning to improve performance.

**Orchestration**

* I envisioned using orchestration tools like **Databricks Workflows** or **Apache Airflow** to automate and monitor the data flow.
* Resilience features such as monitoring schema changes and alerting on failed jobs were incorporated to ensure smooth pipeline execution.



The pipeline diagram illustrates the **Bronze-Silver-Gold framework** for data processing. Data is ingested from batch sources (e.g., CSV, JSON) and streaming sources (e.g., IoT events) into the **Bronze Layer**, which stores raw data with schema enforcement. The **Silver Layer** cleans, validates, and enriches this data, making it structured and queryable. Finally, the **Gold Layer** aggregates the data into ML-ready feature tables and analytics datasets. The process is orchestrated and monitored using tools like Databricks Workflows or Airflow, ensuring scalability and reliability for both real-time and batch use cases.

**Part 2: Data Processing & Feature Engineering**

**Task 3: Real-Time Data Integration Task (Updated)**

**Objective:** Set up a real-time ingestion pipeline using Structured Streaming and Delta Lake with advanced validation to ensure data quality.

**Deliverable:** Code with explanations, including:

* Real-Time Ingestion: Using Structured Streaming setup.
* Advanced Validation Apply schema enforcement, null handling, and deduplication.

**1. Real-Time Ingestion Setup**

I began by setting up a **Structured Streaming pipeline** to handle real-time data from a streaming source, such as IoT devices or event trackers. The pipeline ingests data in JSON format, continuously writing it to a Delta Lake table in the **bronze layer**. This ensured:

* Efficient handling of real-time data.
* Schema enforcement to manage unexpected data structures.
* Scalability for high-throughput scenarios.

**2. Data Validation Using Delta Lake**

To maintain data quality, I used Delta Lake’s advanced features, including:

* **Schema Enforcement**: Automatically rejecting data that doesn’t match the expected schema.
* **Deduplication**: Handling duplicate records with dropDuplicates(), ensuring clean, unique data.
* **Null Checks**: Rejecting rows with missing critical fields using validation logic.
* **Late-Arriving Data Handling**: Ensuring proper ordering and timestamping with Delta Lake’s ACID properties.

3. Code Implementation