FA TME 6014 FR01A (Data Analytics Midterm Exam)

Abdoulie Jallow, Student ID# 3764229

Date: November 23, 2025

Question 1 and Question 4

Question 1

You are hired by a retail company that sells both online and offline. They provide you with the following data sources:

- An SQL database containing customer demographics
- CSV files with monthly sales transactions
- A REST API providing real-time product inventory
- Social media posts scraped from public Twitter accounts about the brand

Task

Design a complete end-to-end data analytics pipeline for this scenario. Your answer must include:

1. Identification and justification of the type of each data source *(structured / semi-structured / unstructured; internal vs external; streaming vs batch)*
2. A detailed data ingestion plan, specifying tools/technologies *(any are allowed)*
3. A full preprocessing workflow for each data source
4. A unified data model *(diagram or explanation)* showing how data from all sources will integrate
5. A discussion of data quality risks and how you would mitigate them

1) Data Source Classification

Before designing the ingestion and processing layers of our pipeline, we must first define the characteristics of our inputs. The architectural decisions for storage, processing engines, and scheduling depend entirely on three key factors:

1. **Structure:** How organized is the data?
2. **Origin:** Where does it come from?
3. **Velocity (Ingestion Mode):** How frequently is data generated and how quickly is it needed?

The following table summarizes the classification of the retail company's four data sources against these criteria:

Data Source	Structure Type	Origin	Ingestion Mode	Justification
SQL database (customer demographics)	Structured (relational tables)	Internal	Batch (daily/weekly extracts); optionally near-real-time via Change Data Capture	Schema-defined tables (e.g., Customers, Addresses
CSV files (monthly sales transactions)	Semi-structured (delimited text)	Internal	Batch (monthly)	Files from POS/e-commerce exports; may vary in headers, delimiters, and encodings
REST API (real-time product inventory)	Semi-structured (JSON)	Internal	Streaming/near-real-time	Endpoint(s) like /inventory return JSON with stock levels

Social media posts (Twitter/X)	Unstructured (text + metadata)	External	Streaming	Free-form text, emojis, hashtags; metadata (user, timestamp,

2) Data Ingestion Plan

The data ingestion plan, as illustrated in the diagram below, is designed around a hybrid ELT (Extract, Load, Transform) strategy, utilizing Azure Data Factory (ADF) as the central orchestration and ingestion engine. This approach ensures that data from all sources, regardless of velocity or structure, is promptly extracted and loaded into a unified raw data lake.

Ingestion Pipelines

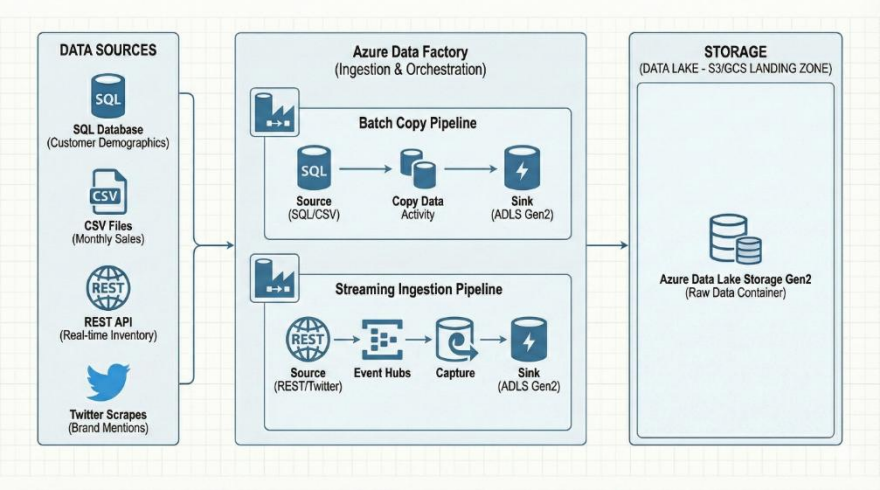
The architecture is divided into two main pipelines managed within Azure Data Factory:

- **Batch Copy Pipeline:**
  - ✓ Sources: This pipeline handles batch data from the SQL Database (Customer Demographics) and CSV Files (Monthly Sales).
  - ✓ Mechanism: ADF's Copy Data Activity is used to connect to these sources. It efficiently extracts the data and loads it directly into the destination storage. This process can be scheduled to run periodically (e.g., daily for SQL, monthly for CSVs) or triggered by file arrival events.
- **Streaming Ingestion Pipeline:**
  - ✓ Sources: This pipeline manages real-time data from the REST API (Real-time Inventory) and Twitter Scrapes (Brand Mentions).
  - ✓ Mechanism: Data from these sources is first ingested into Azure Event Hubs, a highly scalable data streaming platform. The Capture feature of Event Hubs is enabled, which automatically batches the streaming data into files (e.g., Avro format) and loads them into the destination storage. This provides a reliable and scalable way to persist real-time data streams.

Destination Storage

All data, from both the batch and streaming pipelines, is loaded into a single, centralized storage repository:

- ✓ Storage Service: Azure Data Lake Storage Gen2 (ADLS Gen2).
- ✓ Container: A dedicated "Raw Data Container" serves as the landing zone.
- ✓ Strategy: Data is stored in its native format (or a near-native format like Avro for captured streams) without any transformation at this stage. This preserves the original state of the data, providing an immutable record that is essential for auditability and allows for reprocessing if needed.



Source	Ingestion Method	Tools/Tech	Landing Zones	Orchestration
SQL (demographics)	Incremental extract via JDBC/ODBC; Change Data Capture (CDC)	Azure Data Factory / Synapse pipelines	Raw zone (Parquet/Delta); relational staging	ADF triggers; Airflow Directed Acyclic Graphs (DAGs)
CSV (sales)	Secure file pickup (SFTP/Blob) + schema-on-read	ADF copy activity; Azure Databricks Autoloader; AWS Glue; Python (pandas)	Raw file zone -> curated zone	Event-based triggers on file arrival
REST API (inventory)	Polling or webhook -> queue/stream	Azure Functions; Logic Apps; Kafka; Event Hubs	Streaming bronze (JSON) -> silver (flattened)	Serverless function timers; stream processors
Social Media Tweets (Twitter/X)	API/stream ingestion -> queue	Tweepy/official API; Kafka Connect; Event Hubs	Raw JSON stream -> NLP-ready store	Stream jobs (Databricks Structured Streaming)

3) Preprocessing Workflows

3.1 SQL (Customer Demographics)

- ✓ Profile schema & constraints
- ✓ Deduplicate by customer ID; merge households
- ✓ Standardize addresses (postal code formats), phone normalization
- ✓ Handle nulls via business rules (e.g., infer city from postal code)
- ✓ PII masking/tokenization for analytics
- ✓ Slowly Changing Dimensions (SCD Type 2) for attribute change history

3.2 CSV (Monthly Sales)

- ✓ Validate file name patterns and headers
- ✓ Normalize encodings (UTF-8), delimiter detection
- ✓ Cast types (dates, numeric); currency normalization
- ✓ Join to product and customer dimensions; create fact tables (FactSales)
- ✓ Outlier detection (returns, extreme discounts)

3.3 REST API (Inventory)

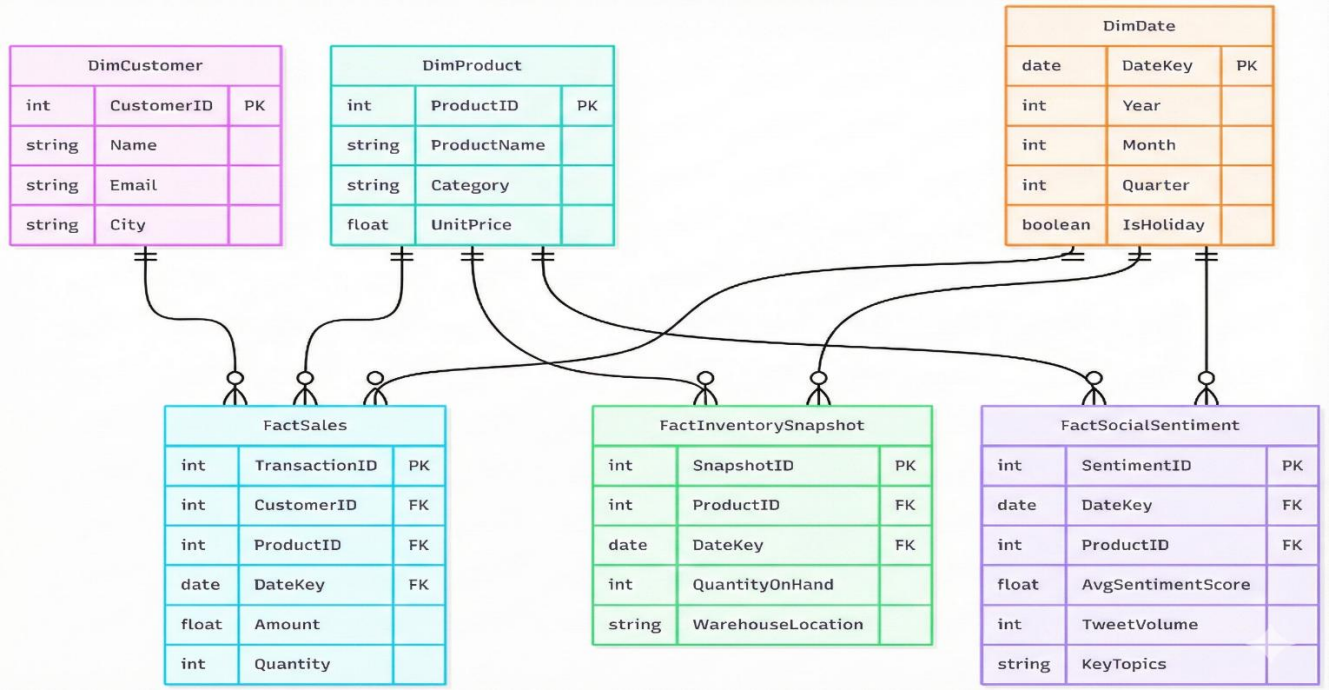
- ✓ JSON schema validation; flatten nested structures
- ✓ Upsert inventory snapshots; compute stock deltas
- ✓ Time zone normalization; latency monitoring
- ✓ Handle API rate limits/backoff; retry with idempotency keys

3.4 Social (Twitter/X)

- ✓ Language detection; remove spam/bots
- ✓ Text cleaning (URLs, emojis, hashtags); tokenization
- ✓ Sentiment analysis (rule-based/ML); topic modeling
- ✓ Map mentions to products via dictionary/embedding similarity
- ✓ Aggregate by time & campaign for marketing dashboards

4) Unified Data Model

To integrate the four disparate data sources into a single analytical environment, I have designed a Dimensional Model (specifically a Galaxy Schema). This structure is optimized for analytical querying and business intelligence reporting (e.g., Power BI, Tableau). The model centralizes measures into Fact Tables and surrounds them with descriptive Dimension Tables.



5) Data Quality Risks & Mitigations

Data quality is managed via a "defense in depth" strategy, implementing validation checks at three key stages: Ingestion (Bronze), Preprocessing (Silver), and Integration (Gold).

A. Source-Specific Risks & Technical Controls

Data Source	Key Quality Risks	Primary Technical Mitigation Strategies
CSV Files **(Sales)**	Schema mismatches (e.g., text in numeric fields), corrupted files, bad formatting.	**Strict Schema Enforcement:** Use Spark to enforce rigid data types on read. Quarantine non-conforming records to an error table for manual review instead of failing the entire pipeline.
REST API **(Inventory)**	API downtime leading to data gaps; late-arriving data due to network latency.	**Retry Logic & Event-Time:** Implement robust retry logic with exponential backoff in the polling agent. Use source timestamps and streaming watermarks to correctly order late data.

Twitter Scraper **(Social)**	High noise/spam skewing sentiment analysis; API rate limiting.	**Multi-stage Filtering:** Apply initial keyword filters at ingestion. Use NLP models in the Silver layer to identify and discard bot spam before sentiment scoring.
SQL Database **(Customers)**	Unexpected upstream schema drift breaking extraction jobs.	**Schema Evolution:** Utilize Delta Lake's schema evolution capabilities to automatically handle additive changes (like new columns) without breaking the pipeline.

B. Architectural Mitigations (System-Wide Defenses)

Beyond specific sources, the pipeline architecture itself provides systemic quality defenses:

- Immutable Bronze Layer (The "Undo Button"): By retaining raw data in an immutable Bronze layer, we ensure we can always re-process historical data if a bug is discovered in the Silver transformation logic later on.
- Handling "Late Arriving Dimensions": To ensure referential integrity when joining fast streams (Inventory) with slower batches (Product Dimensions), we use "stubbing" in the Gold layer. Temporary placeholder records are created in dimension tables, so fact records aren't dropped due to timing mismatches.
- Automated Freshness Alerts: We implement Service Level Objective (SLO) monitors on critical Gold tables to trigger alerts immediately if streaming data stops arriving within expected timeframes (e.g., no inventory updates in 15 minutes).

Question 4

Q4) *Compare the following data sources for a national health-care analytics project:*

1. *Government open-data portals*
2. *Hospital EHR (Electronic Health Record) systems*
3. *Patient-reported survey data*
4. *Wearable device sensor streams (daily step count, heart rate, etc.)*

For each data source:

1. *Evaluate accuracy, completeness, timeliness, consistency, and potential bias.*
2. *Identify one major risk that could affect data reliability.*
3. *Propose a preprocessing technique that specifically mitigates that risk (e.g., truth cross-validation, sampling correction, deduplication, time alignment, noise filtering).*

Then, choose one analytics task (e.g., predicting disease risk) and explain how weaknesses in these data sources could distort model performance—even after preprocessing.

PART 1: DATA SOURCE EVALUATION

1. Government Open-Data Portals

Dimensions:

- ✓ Accuracy: Generally high (aggregated and vetted by statistical agencies), though lacks individual granularity.
- ✓ Completeness: High at a macro level (population coverage) but often suppresses low-count data for privacy.
- ✓ Timeliness: Low. There is often a significant lag (month to years) between collection and publication.
- ✓ Consistency: High within specific datasets, but variable across different jurisdictions (e.g., different states defining "rural" differently).
- ✓ Bias: Reporting bias; relies on what institutions choose to report or are funded to track.

Major Risk: Temporal Lag. By the time data is available, it may no longer reflect current population health trends (e.g., pre-pandemic data used for post-pandemic modeling).

Preprocessing Mitigation: Nowcasting (Time-Series Extrapolation). Use historical trends combined with leading indicators (like recent search trends) to project the "stale" government data to the current date.

2. Hospital EHR (Electronic Health Record) Systems

Dimensions:

- ✓ Accuracy: Mixed. Clinical notes contain high detail, but diagnostic codes (ICD-10) are often prone to human error or "upcoding" for billing purposes.
- ✓ Completeness: Low. Fragments of data exist only for when a patient visits that specific hospital system; no data exists for healthy periods or visits to other providers.
- ✓ Timeliness: High (Real-time or near real-time).
- ✓ Consistency: Low. Formats vary wildly between EHR vendors (e.g., Epic vs. Cerner) and even between departments

- ✓ Bias: Selection bias; the data only represents sick individuals who seek care, ignoring the healthy population.

Major Risk: Data Sparsity & Fragmentation. Patients have irregular visit intervals, resulting in massive gaps in the timeline.

Preprocessing Mitigation: Longitudinal Imputation (e.g., MICE or RNN-based imputation). Instead of dropping incomplete records, use statistical techniques to infer values for missing time points based on the patient's trajectory and similar patient profiles.

3. Patient-Reported Survey Data

Dimensions:

- ✓ Accuracy: Moderate to Low. Heavily dependent on the patient's memory and honesty.
- ✓ Completeness: Variable. Respondents often skip sensitive questions (e.g., substance use).
- ✓ Timeliness: Low (usually snapshot-based).
- ✓ Consistency: Low. Subjective interpretation of questions (e.g., pain scale ratings vary by person).
- ✓ Bias: Non-response bias; those who complete surveys are often healthier, wealthier, or more motivated than the general population.

- **Major Risk: Recall Bias/Subjectivity.** Patients may misremember dates or downplay unhealthy behaviors (social desirability bias).
- **Preprocessing Mitigation: Truth Cross Validation.** Link a subset of survey respondents to their EHR records to calculate an "error rate" or "reliability score" for the survey data, then weigh the survey inputs accordingly.

4. Wearable Device Sensor Streams

Dimensions:

- ✓ Accuracy: Variable. Consumer-grade sensors (Fitbit, Apple Watch) are less accurate than clinical devices.
- ✓ Completeness: High (continuous monitoring), but breaks if the user forgets to charge or wear the device.
- ✓ Timeliness: High (streaming data).
- ✓ Consistency: High for a single device, low across different device generations/brands.
- ✓ Bias: Socioeconomic bias; data is skewed toward younger, wealthier, tech-literate populations.

- Major Risk: Signal Noise/Artifacts. Movement artifacts (e.g., shaking a hand) can register as steps, or loose fits can cause heart rate dropouts.
- Preprocessing Mitigation: Noise Filtering (e.g., Butterworth Low-Pass Filter). Apply digital signal processing to smooth the raw sensor stream and remove high-frequency noise that represents artifacts rather than physiological signals.

Part 2: Impact on Analytics Task

Selected Analytics Task: Predicting Long-Term Cardiovascular Disease (CVD) Risk

Even after applying the preprocessing techniques above (imputation, filtering, etc.), distinct weaknesses in these sources can distort model performance in the following ways:

1. The "Healthy User" Effect (Socioeconomic Distortion)

- ✓ Distortion: The model will likely be overweight with features derived from Wearable and Survey data. Because these devices are expensive and surveys take time to complete, the data predominantly comes from wealthier, health-conscious individuals.
- ✓ Model Failure: The model will accurately predict Cardiovascular Diseases (CVC) risk for affluent, tech-savvy users but will likely fail (or yield high False Negatives) for low-income populations who do not generate wearable data and visit hospitals less frequently (EHR gaps). The model learns to predict "access to technology" rather than biological risk.

2. The "Sick Care" Loop (Selection Bias)

- ✓ Distortion: EHR data is the most clinically accurate, but it relies on interaction. A patient with early stage of CVD who does not visit the doctor appears "healthy" (missing data) to the model, whereas a hypochondriac with frequent visits appears "high risk" due to data volume.
- ✓ Model Failure: The model may learn to associate number of data points with disease severity, leading to over-diagnosis of frequent utilizers and under-diagnosis of those with barriers to healthcare access.

3. The "Lag" Effect (Temporal Mismatch)

- ✓ The Distortion: Integrating Government Open Data (e.g., regional obesity rates from 2 years ago) with real-time Wearable data creates a temporal mismatch.
- ✓ Model Failure: If a sudden health event occurs (like a new environmental toxin or a pandemic), the real-time sensors will pick up physiological stress, but the model anchored by stale government baselines may suppress the risk score, assuming the regional baseline is still "normal."

Summary Table

Data Source	Major Risk	Recommended Preprocessing
Government Data	Temporal Lag (Staleness)	Nowcasting / Time-series Extrapolation
Hospital EHR	Data Sparsity / Fragmentation	Longitudinal Imputation (MICE/RNN)
Patient Surveys	Recalling / Social Desirability Bias	Truth Cross Validation (with EHR)
Wearables	Signal Noise / Artifacts	Noise Filtering (Butterworth/Smoothing)