FULLY CLOUD-BASED ETL PIPELINE ON AWS

CHIRATH SETUNGE

Table of Contents

Question 1

Initialize AWS tools

Launched an EC2 instance of type t2medium with Ubuntu, and configured it with a 30GB gp3 SSD, which can easily store logs and installations. Apart from going with a free tier t2 micro, the t2 medium was selected because we have to run Apache Airflow webserver and databases, DAG execution, and SSH. It offers 4GB of memory and 2 virtual CPUs. Configured security groups that allow port 22 for SSH and port 8080 for Apache Airflow webserver. To smoothly run the boto3 executions and not save long-term AWS keys on disk, created an S3 bucket called weather-data-cm2606 and built an IAM role that granted full S3 access to EC2. Finally, connected this role to the EC2 instance.

In the EC2 instance, install Python3 virtual environment to create a dedicated virtual environment to isolate the Airflow installation and other library installations to avoid package conflicts, that rise along with the development. Airflow variables were also created to securely store the OpenWeather API key.

ETL Pipeline

Four tasks are orchestrated by the DAG file weather_etl_dag. To make sure the API is operational before extraction, an HTTP Sensor called check_weather_api_ready first polls the OpenWeatherMap endpoint until it responds with HTTP 200. After that, a GET request is sent by the HTTP Operator extract_weather_data, which then uses JSON. Loads to convert the HTTP response into structured JSON. The third step is pulling that JSON via XCom and flattening it into a dictionary including latitude, longitude, temperature, wind speed, and wind direction using a Python Operator named transform_load_weather_data. After converting this record into a one-row pandas dataframe, encode the and call boto3 to stream the CSV into the S3 bucket and give the file a timestamped name, such as current_weather_data_london_20250419093000.csv. A final Dummy Operator pipeline_success signifies the end of the flow.

Automation and Scheduling

Airflow handles automation natively. To prevent retroactive executions, set the DAG's schedule interval to run daily to trigger once daily and catch up to False. To ensure that temporary API problems are automatically retried, default arguments contain retries to 2 with a two-minute delay for retry. The downstream Dummy Operator ensures that no further runs begin until all upstream jobs have completed successfully, and the HTTP Sensor requirement ensures that the API is online before extraction. Code changes are picked up instantly since the Airflow scheduler reloads DAG files every 30 seconds. The pipeline can also be triggered from Airflow UI. And the Airflow UI can be used to monitor each task in detail with errors.

Evaluation

To evaluate the ETL pipeline the performance parameters like scheduler lag and overall run duration in order to assess our ETL pipeline. Verified the correctness by comparing a few sample temperatures with the Open Weather Map dashboard and confirming that each output CSV follows the desired structure, which includes columns for Latitude, Longitude, Temperature, Wind Speed, and Wind Direction.

Improvements

A significant enhancement would be moving our metadata backend from SQLite to PostgreSQL, allowing multi-worker parallelism. This modification significantly increases fault tolerance and throughput, enabling resource-intensive or high-volume DAGs to operate concurrently without scheduling conflict. Furthermore, incorporating real-time email or Slack alerts through email operators or even using lambda functions guarantees that messages are sent out instantly for both successes and failures. These alerts lower mean time to detection and resolution, enhance operational visibility, and facilitate adaptive handling of events in production systems. A robust, observable ETL pipeline that is prepared for production is made possible by these improvements.

Question 2

Dimension Tables

Store dimension (STORE_DIM)

The Store dimension table contains all of the location-related data related to SuperMart's stores, with the following attributes

- StoreKey (surrogate key)
- StoreID
- StoreName
- City
- State
- Country
- Region
- StoreSize

One crucial analytical element that enables SuperMart to divide up and examine performance across various geographical areas is the Store dimension. This makes it possible for management to evaluate store performance across markets, spot regional trends, and base business choices on location. In analytical reporting, the location data's hierarchical structure, country, state, city, offers versatile drill-down options.

Product Dimension (PRODUCT_DIM)

The product dimensions table contains all of the comprehensive information about all the items sold at stores with the following attributes

- ProductKey (surrogate key)
- ProductID
- ProductName
- Category
- SubCategory
- Supplier

- Brand
- UnitCost
- UnitPrice

This dimension makes it possible to analyze product performance in great detail, which helps SuperMart find top-performing products, determine which categories drive sales, and successfully manage supplier relationships. Diving from broad categories to specific products is made easier by the natural hierarchies created by the categorical attributes in the table.

Time Dimension (TIME_DIM)

To facilitate temporal analysis, the Time dimension table includes a variety of date and time attributes.

- DateKey (surrogate key)
- Date
- DayOfWeek (1-7)
- DayOfMonth (1-31)
- Month (1-12)
- Quarter (1-4)
- Year
- IsHoliday (boolean)
- Season (Spring, Summer, Fall, Winter)

Most of the business intelligence applications rely on the Time dimension since it allows for forecasting, trend analysis, seasonality detection, and period-over-period comparisons. Its hierarchical structure year, quarter, month, day) allows for versatile time-based reporting capabilities.

Customer Dimension (CUSTOMER_DIM)

Information on SuperMart's customers is kept in the Customer dimension table

- CustomerKey (surrogate key)
- CustomerID
- CustomerName

- MembershipLevel (bronze, silver, gold)
- MembershipStartDate
- Email
- CustomerSince (first purchase was made)

Customer segmentation, loyalty program research, and customized marketing tactics are made possible by this dimension. SuperMart may assess the efficacy of its reward program levels and examine purchasing trends across various client segments by connecting sales transactions to customer variables.

Fact table

Sales Fact Table (FACT_SALES)

This star schema's primary fact table includes measures and transactional data.

- SalesKey (surrogate key)
- TransactionID
- DateKey (FK - TIME_DIM)
- StoreKey (FK - STORE_DIM)
- ProductKey (FK - PRODUCT_DIM)
- CustomerKey (FK - CUSTOMER_DIM)
- QuantitySold
- SalesAmount
- DiscountAmount
- NetSalesAmount
- CostAmount
- ProfitAmount

The distinctive star pattern of dimensional modeling is created by the foreign keys, which create relationships with all dimension tables. The star schema design is based on each foreign key referring to the surrogate key in its corresponding dimension table.

The most adaptable metrics for analysis are additive facts since they can be combined across all dimensions.

Additive Facts	
QuantitySold	Unrestricted summation over all dimensions
SalesAmount	Able to be aggregated in a meaningful way across all dimensions
CostAmount	Indicates the whole, additive cost of items sold across all dimensions.
ProfitAmount	Unrestricted summation over all dimensions

Semi-Additive Facts	
Average Transaction Value	Sales amount divided by transaction count would be used to determine it; it cannot be added up over time dimensions directly, but it can be averaged.

Non-Additive Facts	
Profit Margin Percentage	It would be computed as (ProfitAmount/SalesAmount)*100; at each level, it must be recalculated because it cannot be meaningfully summed over any dimensions.

Aggregate Tables

Monthly Sales by Store and Category (AGG_MONTHLY_SALES_BY_STORE_CATEGORY)

- YearMonth
- StoreKey (FK - STORE_DIM)
- CategoryName (FK - PRODUCT_DIM)
- TotalQuantitySold
- TotalSalesAmount
- TotalTransactions
- AverageTransactionValue

For category-level analysis and monthly sales reporting, this aggregate table greatly improves query efficiency. SuperMart can produce reports far more quickly by recalculating these typical aggregations rather than processing every single specific transaction. This is especially helpful for frequent monthly performance evaluations and executive dashboards.

Daily Sales by Membership Level (AGG_DAILY_SALES_BY_MEMBERSHIP)

- DateKey (FK - TIME_DIM)
- MembershipLevel (CUSTOMER_DIM)
- TotalCustomers (count of distinct customers)
- TotalTransactions
- TotalQuantitySold
- TotalSalesAmount
- AverageSalesPerCustomer

The analysis of the effectiveness of the loyalty program is optimized by this aggregate table. SuperMart can use it to quantify the impact of membership-specific promotions, observe changes in customer behavior across membership levels, and swiftly assess how various membership tiers affect daily sales. For the management of loyalty programs and focused marketing campaigns, this information is essential.

Business Questions

1. Which product categories demonstrate the most pronounced seasonal trends, and what is the monthly sales trend per product category across various regions?

- In order to examine regional performance trends, this question makes use of the AGG_MONTHLY_SALES_BY_STORE_CATEGORY aggregate data in conjunction with DIM_STORE. Inventory planning and regional marketing tactics can be informed by the analysis's ability to pinpoint which product categories do well in which locations during particular seasons.

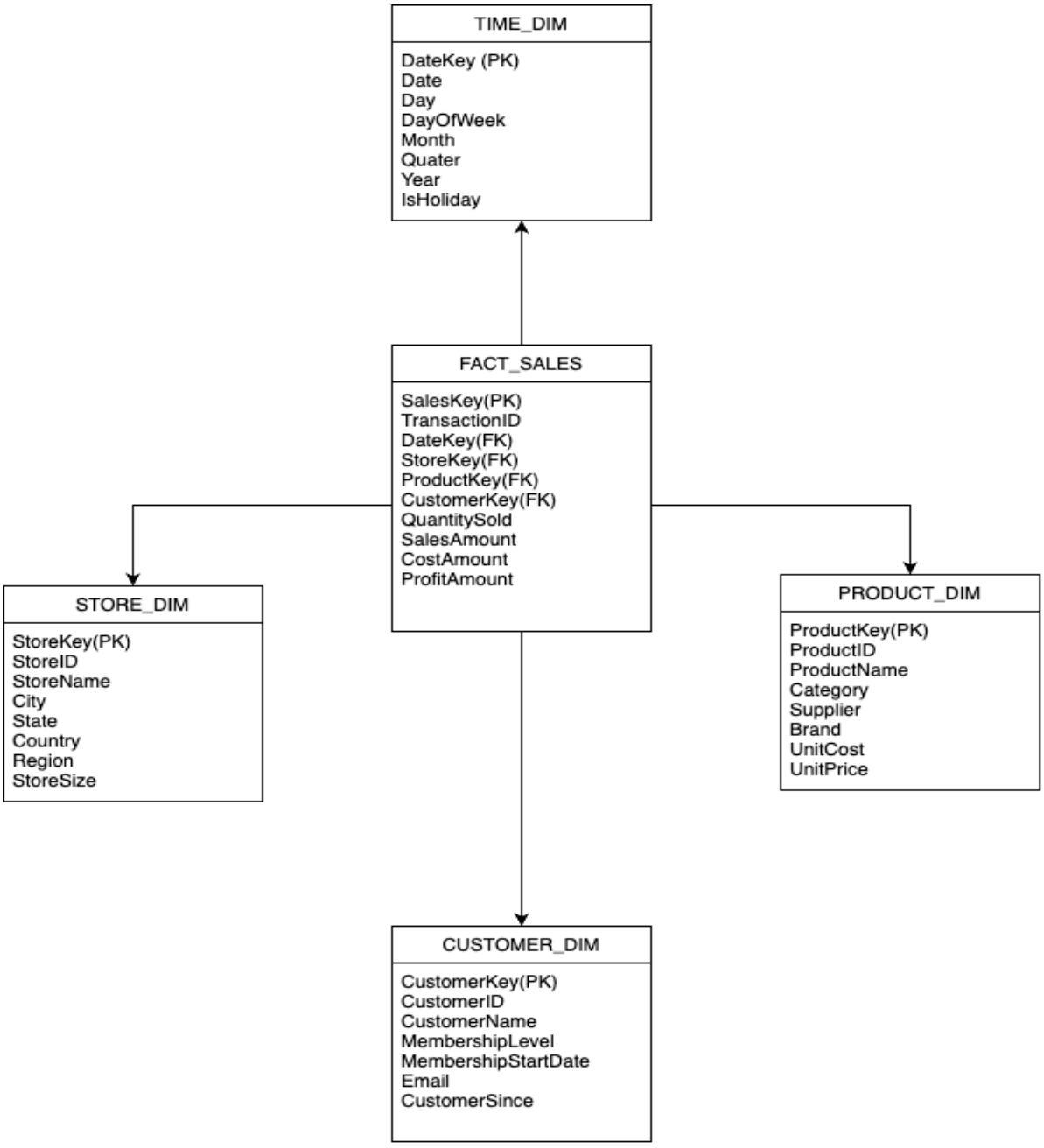
2. What effects do membership upgrades have on the frequency and amount of purchases, and how does customer purchasing behavior vary by membership tier?

- To answer this question, need to analyze spending trends by membership tier using the AGG_DAILY_SALES_BY_MEMBERSHIP aggregate dataset. The information can be used to improve the structure of the loyalty program and develop focused rewards to promote membership tier upgrades.

3. How do these trends connect to the demographics of the local market, and which retailers are seeing the fastest growth rates for particular product categories?

- In order to determine growth prospects, this query integrates store-specific data with information from the AGG_MONTHLY_SALES_BY_STORE_CATEGORY aggregate table. In areas with positive developments, the data can help guide growth planning and store-specific merchandising strategies.

Star Schema Diagram



Question 3

Suggesting a layered authentication strategy that uses Multi-Factor Authentication for users like doctors, patients, data scientists, and auditors to log in using two or more verification steps (password + TOTP). To limit reauthentication, Federated Single Sign-On (SSO) via OAuth 2.0/OpenID Connect is used only authenticate users only once through a central cooperate identity provider, such as Google or Azure AD, and then gives access to multiple systems. For securely authenticating backend services or API clients that communicate with each other, Mutual TLS with certificates (X.509) is used. Hardware security modules securely store and handle these certificates. When the login context changed, such as a new geolocation from the previous state, risk-based adaptive authentication was adapted to increase the level of verification. Suggest a hybrid RBAC + ABAC approach for authorization. Doctors, patients, data scientists, and external auditors who correspond to high-level permissions are enforced with IAM policies and database ACLs. Open policy agent engines evaluated ABAC policies, which apply fine-grained rules based on user attributes and data attributes. Under strong audit, Just-In-Time privilege elevation permits temporary access for specific tasks. This ensures HIPAA/GDPR compliance by enforcing context-aware access restrictions, rigorous identity verification, and least-privilege rules.

Role	View De-identified Data	View Full PHI	Run Analytics Jobs	Export Reports	Audit Logs Access
Data Scientist	✓	✗	✓	✓ (masked)	✗
Doctor	✓	✓	✓	✓	✗
External Auditor	✓	✗	✗	✗	✓

Thte doctors have complete access to PHI (personal health data), external auditors can only see audit logs, and data scientists can perform analytics on de-identified data but cannot view PHI.

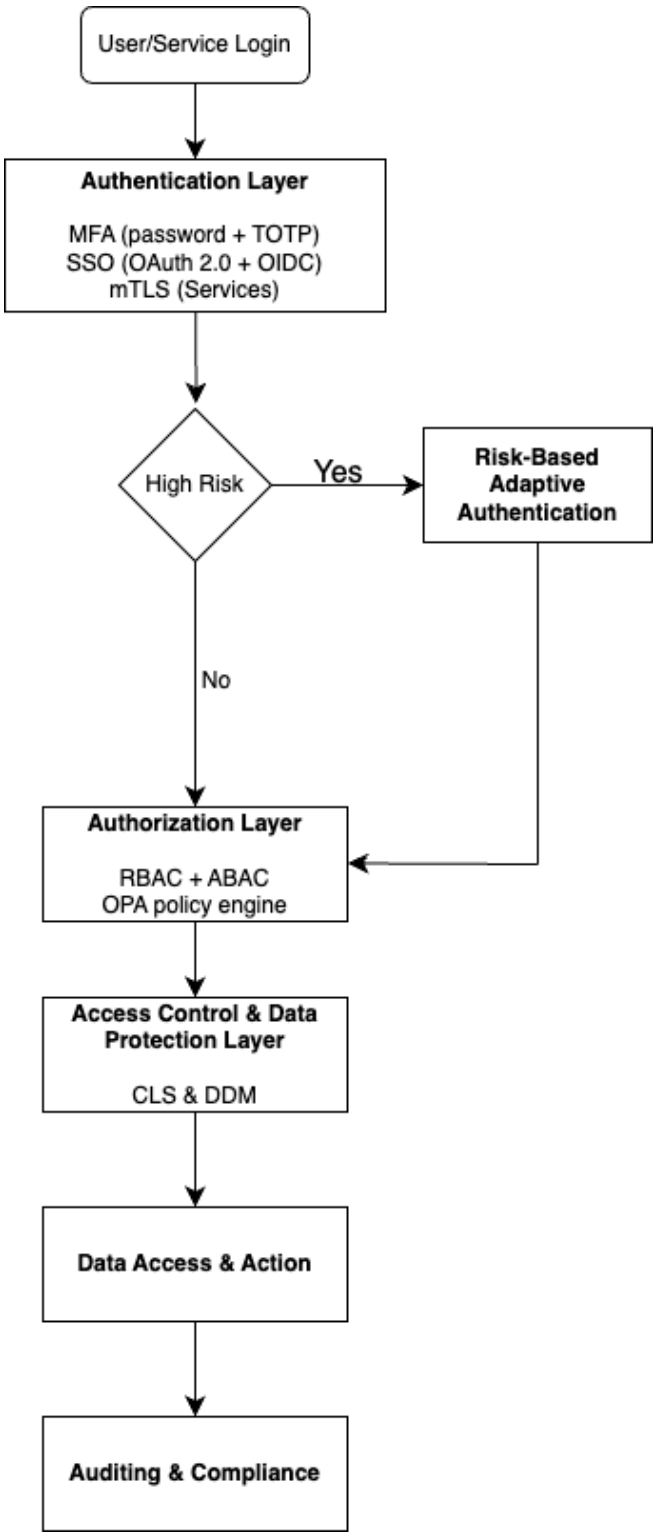
To protect HealthAnalytics Inc.'s private patient data, suggest encrypting data while it's at rest using AES-256, a strong symmetric encryption that has FIPS compliance and broad hardware acceleration. To safeguard PHI, insurance IDs, and biometrics, cloud-stored data, like as databases and object storage, will make use of services like Transparent Data Encryption or column-level encryption. To protect communication channels between client applications, microservices, and the data Lakehouse, and concurrently use TLS 1.3 for data in transit. Mutual TLS supports encryption and authentication in service-to-service communications. Key management is essential; hardware security modules (HSMs) or cloud key management services such as AWS KMS will be used to safely store keys, guaranteeing appropriate lifespan and rotation strategies. Additionally, ChaCha20-Poly1305 offers a great substitute in settings with less hardware support for AES. While guaranteeing performance and data integrity across all channels, the multilayer encryption approach additionally complies with HIPAA and GDPR regulations.

A policy that combines Column-Level Security (CLS) and Dynamic Data Masking (DDM) is suggested to safeguard sensitive fields like Patient identifiers, such as Insurance IDs, Social Security numbers and others such as medical data info and biometric data. To guarantee that only authorized roles, like doctors, can read unmasked sensitive material, CLS is used to implement stringent access control at the database level. Secure views are set up to prevent or limit access to these columns for other users, such as external auditors or data scientists. Concurrently, DDM is used to dynamically mask social security number in real time, showing them to any user without the necessary authorization as "XXX-XX-1234. By combining runtime data encryption and role-based permissions, this policy ensures that sensitive information is kept private while adhering to the least privilege principle. In order to facilitate compliance with HIPAA and GDPR and improve overall data governance and security, comprehensive audit logs are kept to track data access and any attempts at bypassing the boundaries.

At HealthAnalytics Inc., a thorough architecture that includes logging, data retention, and real-time monitoring is necessary to guarantee HIPAA and GDPR compliance. To begin, create immutable audit trails that record each user action, API call, and data access using AWS CloudTrail, Azure Monitor, or comparable solutions. This serves as the foundation for ongoing compliance evaluations and forensic investigations. Create data retention standards that specify the duration of time that each type of data should be kept on file. At the same time, use cloud

lifecycle management solutions to automatically archive or destroy data according to its age and level of sensitivity. In order to correlate and analyze events, generate alerts, and react quickly to issues, combine these measures with a strong real-time monitoring system that integrates security information and event management (SIEM) solutions like Splunk or IBM QRadar. To further secure patient records, use encryption, dynamic data masking, and strict access limits. Staff training on incident response and security best practices, as well as routine internal and external audits, strengthens the compliance posture and promotes a continuous improvement culture in data governance.

To centralize security and traffic control, are going to front the aggregated data service behind an API gateway, such AWS API Gateway or Azure API Management. Every client request passes via the gateway, which ends TLS 1.3, verifies JWT tokens or API keys, and makes sure the scopes of the tokens allow for the retrieval of aggregated data. Applied per user or API key, rate-limiting restrictions prevent misuse and preserve platform performance. De-identified, aggregated metrics (such as counts and averages) are continuously computed by an ETL process on the backend and written to a specific, non-PII schema. The gateway never routes calls to raw data stores; it only routes them to specific endpoints. Strict ABAC rules guarantee that only the "ResearchPartner" role may access these APIs, and request and response validations remove any remaining identities. All API calls, along with any authorization or throttling issues, are recorded for observability in a SIEM or CloudWatch Logs, allowing for real-time monitoring and alerts. Partners can safely use research datasets without being exposed to specific patient records thanks to a hardened, aggregated-data layer, tokenized access, rate restriction, and an API gateway.


User/Service Login
Authentication Layer
MFA (password + TOTP)
SSO (OAuth 2.0 + OIDC)
mTLS (Services)
High Risk
Yes
Risk-Based Adaptive Authentication
No
Authorization Layer
RBAC + ABAC
OPA policy engine
Access Control & Data Protection Layer
CLS & DDM
Data Access & Action
Auditing & Compliance