Pizza Chain Insights

1. Business Challenge/Requirement

Objective

PizzaChain Insights is a cloud-native batch analytics pipeline designed to simulate the real-world data infrastructure needs of a multi-branch pizza retail chain. It ingests operational data such as orders, inventory, and discounts from stores across regions, processes it via AWS Glue, stores curated datasets in Amazon S3, and provides actionable insights using Athena and a scalable EC2-hosted dashboard. The project includes automated alerting for operational risks

Data Source (RDS)

- Discounts_applied
- Inventory logs
- order_items
- orders
- sku master

Below is an abstract of end to end process:

- Data is injested in to RDS database for all the required tables.
- The data is read from RDS using glue(Crawler-job) and written in to S3.
- Athena queries can be run on the S3 data
- A lambda runs on schedule to run athena queries to find thresolds and puts those orders in sqs
- EC2/Lambda will read messages from Queue and looks up RDS for information like store id and sends SNS to stores

Core Business Questions

1. **Top 5 Selling SKUs per Store in the Last 7 Days**
2. **Category-wise Revenue Breakdown with Discounts Applied**
3. **Identify Orders Where Discount > 30% of Total Value**
4. **Low Inventory Alert Based on Last Weekend's Average Sales**
5. **Revenue and Orders by Hour of Day**

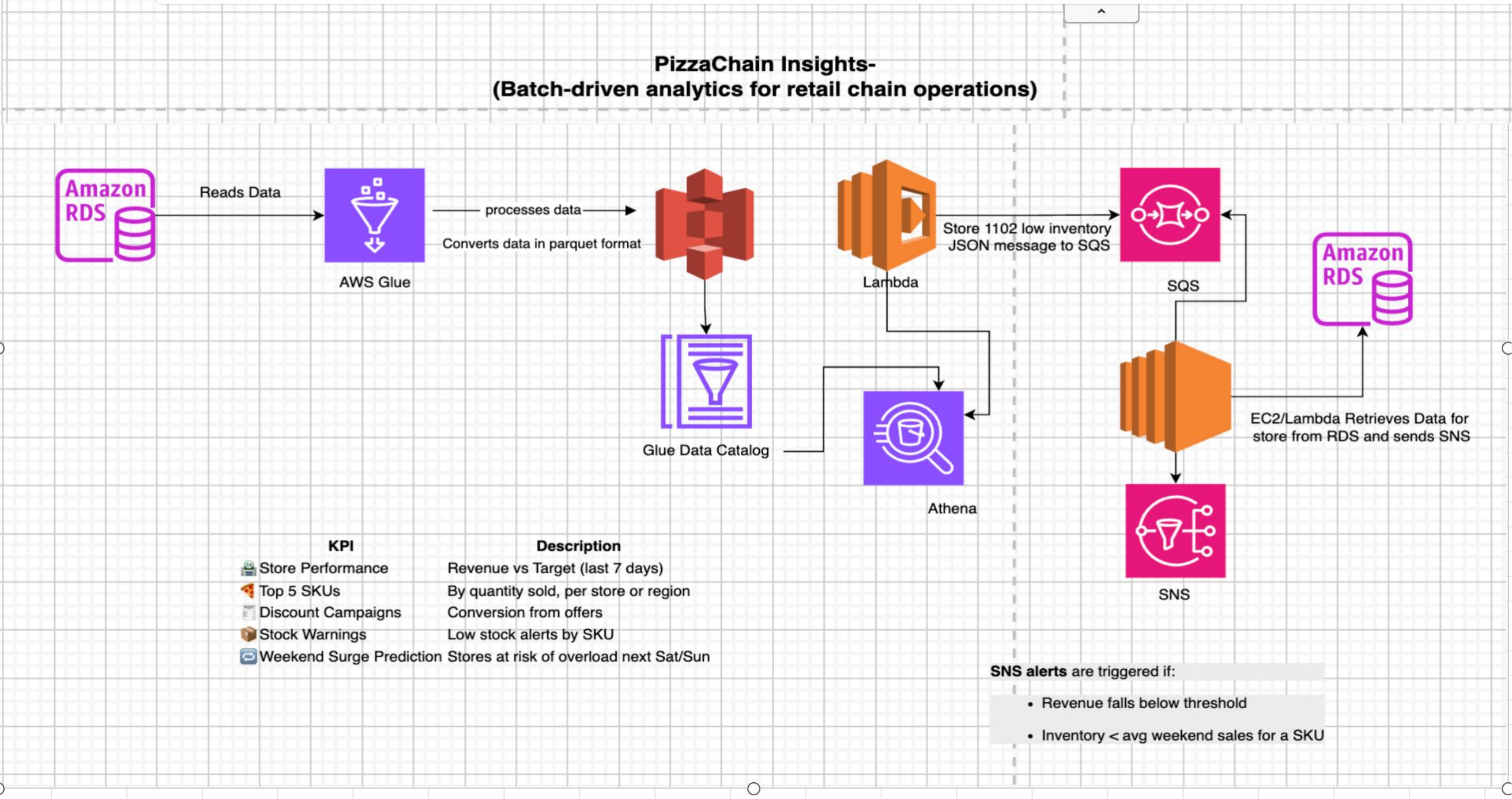
2. The Goal of the Project

Below are some of the high-level technical and non-technical goals for this project:

Build a batch-based data pipeline using Amazon RDS, AWS Glue, S3, and Athena

- Clean, transform, and partition multi-store retail data
- Automate alerts via Amazon SNS for issues like low stock or unmet revenue targets
- Deploy a KPI dashboard using EC2
- Ensure secure, modular, and extensible design using IAM roles

3. Data Flow Architecture/Process Flow


PizzaChain Insights-
(Batch-driven analytics for retail chain operations)
Amazon RDS
Reads Data
AWS Glue
processes data
Converts data in parquet format
Glue Data Catalog
Lambda
Store 1102 low inventory JSON message to SQS
SQS
Athena
Amazon RDS
EC2/Lambda Retrieves Data for store from RDS and sends SNS
SNS

KPI	Description
🏪 Store Performance	Revenue vs Target (last 7 days)
🍕 Top 5 SKUs	By quantity sold, per store or region
🧾 Discount Campaigns	Conversion from offers
📦 Stock Warnings	Low stock alerts by SKU
🔁 Weekend Surge Prediction	Stores at risk of overload next Sat/Sun

SNS alerts are triggered if:

- Revenue falls below threshold
- Inventory < avg weekend sales for a SKU

Dataset Explanation and Schema

We have I data source from RDS:

We have the below datasets -

Discounts applied

column Name	***Data type***	***Column description***	***sample value***
Discount code	***varchar***		
Discount_amount	***double***		

inventory_logs

column Name	***Data type***	***Column description***	***sample value***
log_time	***varchar***		
store_id	***int***		
sku_id	***varchar(50)***		
current_stock	***int***		
restock_thresold	***int***		

order_items

column Name	*Data type*	*Column description*	*sample value*
order_id	*varchar*		
sku_id	*varchar(10)*		
quantity	*double*		
unit_price	*double*		
discount_code	*varchar*		
discount_amount	*double*		

orders

column Name	*Data type*	*Column description*	*sample value*
order_id	*varchar*		
customer_id	*varchar(75)*		
store_id	*varchar(2)*		
order_time	*varchar(2)*		
total_amount	*double*		

sku_master

column Name	***Data type***	***Column description***	***sample value***
sku_id	***varchar***		
item_name	***varchar***		
category	***varchar***		
price	***double***		
created_at	***timestamp***		

Copy the file - generate_pizza_chain_data.py to your VM and csv files required to load in to your SQL database

Problem Statements/Tasks

1. Create scripts to load in to database

2. Load the data in to S3 using glue from RDS

3.Perform all the transformations in the glue job

4.Run the given athena queries on the enriched S3 data

5.Create Lambda function to run thresold queries and put messages in SQS

6.Send SNS from EC2 for the selected stores

Transformations for Step 3 above

Orders	*Remove records with missing order_id.* *Join with order_items to calculate total order value.* *Join with discounts_applied (aggregated) to compute total discount per order.* *Add a derived column day_of_week from order_time.*
order_items	*Remove records with missing order_id, sku_id, quantity, or unit_price.* *Convert quantity to integer and unit_price to double.* *Calculate a new column item_total = quantity × unit_price.* *Join with sku_master to add item name and category.* *Join with discounts_applied to calculate discount value per line item.*
sku_master	*Drop rows missing sku_id.* *Clean item_name and category:* *Remove leading/trailing spaces.* *Convert to lowercase.* *Use this table to enrich both order_items and inventory_logs with readable product names.*
discounts_applied	*Drop rows missing discount_code.* *Clean discount_code (lowercase and trim).* *Convert discount_amount to double.* *This data is used to:* *Join with order_items by discount_code.* *Compute discount value for each item.*
inventory_logs	*Drop rows missing sku_id, store_id, current_stock, or unit_price.* *Convert current_stock to integer (rename to stock_qty).* *Convert unit_price to double.* *Calculate stock_value = stock_qty × unit_price.* *Enrich with sku_master to get item name and category.*

Queries to run

1. **Top 5 Selling SKUs per Store in the Last 7 Days**
2. **Category-wise Revenue Breakdown with Discounts Applied**
3. **Identify Orders Where Discount > 30% of Total Value**
4. **Low Inventory Alert Based on Last Weekend's Average Sales**
5. **Revenue and Orders by Hour of Day**
6. **Running Total of Revenue by Store**
7. **Customers with High Frequency and High Value Orders**
8. **Most Discounted Products by Total Discount Given**
9. **SKU Sell-Through Rate (Sold Quantity vs. Inventory)**
10. **Order Trends with Day of Week and Category**
11. Store-wise Revenue, Discount Impact, and Inventory Status (Today)
(Combines revenue, discounts, SKU categories, and inventory for **store-level operational decisions**.)

12. Top Discounted Products with Inventory Alert (Last 30 Days)

(Helps identify **highly discounted SKUs that are also low on stock**, useful for **replenishment planning** and **promotion strategy evaluation)**