E-COMMERCE DATA PIPELINE

By: Prayoga Agusto Haradi

01

PROJECT OVERVIEW

E-commerce Data Pipeline and Analytics Dashboard

Objectives

- Develop an automated data pipeline for processing e-commerce transaction data
- Transform raw data into a structured format suitable for analytics (star schema)
- Load processed data into a cloud-based data warehouse
- Create insightful visualizations for business intelligence

Analysis Overview

- Sales performance across different product categories
- Revenue distribution by order priority
- Customer device preferences (web vs. mobile)
- Profit trends over time

E-commerce Data Pipeline and Analytics Dashboard (Continued)

Key Results

- Automated monthly data processing
- Enhanced data quality and consistency by data cleaning.
- Improved pipeline reliability and scalability
- Optimized data warehouse performance by creating a star schema.
- Delivered actionable business insights.

Platforms

- Apache Airflow
- Python
- AWS S3
- Docker
- AWS Redshift
- Amazon QuickSight

02

PROJECT BACKGROUND

Background

E-Commerce platforms often face critical challenges with data silos hindering decision-making and manual reporting causing delays

This project develops an end-to-end data pipeline for a growing e-commerce platform, transforming raw transaction data into actionable insights. By automating data processing and creating an analytics dashboard.

The project will benefit multiple stakeholders, such as: business leaders gain real-time performance data, marketing teams access customer insights, and operations optimize inventory management.

Problem Statement

01

Data Source

Local CSV files processed monthly

02

Data Quality

Inconsistent and potentially erroneous data

03

Data Storage

Lack of scalable storage solution

04

Data Processing

Manual, time-consuming ETL processes

05

Data Analysis

Lack of a centralized, optimized data warehouse hinders quick insights

06

Reporting

Limited ability to generate real-time, actionable insights

Success Metrics

Automation

Reducing the percentage of manual ETL Processes.

Data Accuracy

Increase the percentage of records accurately processed and transformed.

Query Performance

Improve the average response time for standard analytical queries.

Business Impact

Increase the number of data-driven decisions made using insights from the new system.

Data Platforms Used

Data Source and Extraction

Apache Airflow orchestrates the extraction of data from monthly updated CSV files.

Data Transformation and Staging

Python scripts clean and transform the data before storing it in AWS S3.

Data Loading to Warehouse

Processed data is efficiently transferred from S3 to Amazon Redshift.

Data Warehouse Optimization

Redshift organizes data in a star schema for optimized analytics performance.

Data Analysis and Visualization

Amazon QuickSight connects to Redshift to create interactive dashboards and reports.

Data Source

- The data source used is a E-Commerce dataset stored in a local computer downloaded from the website Kaggle. It consists of the columns:
  Order_Date, Time, Aging, Customer_Id, Gender, Device_Type, Customer_Login_type, Product_Category, Product, Sales, Quantity, Discount, Profit, Shipping_Cost, Order_Priority, Payment_method.

- Below is a short snippet of the dataset used.

Order_Date	Time	Aging	Customer_	Gender	Device_Type	Customer_Login_type	Product_Category	Product	Sales	Quantity
1/2/2018	10:56:33	8	37077	Female	Web	Member	Auto & Accessories	Car Media	140	1
7/24/2018	20:41:37	2	59173	Female	Web	Member	Auto & Accessories	Car Speake	211	1
11/8/2018	8:38:49	8	41066	Female	Web	Member	Auto & Accessories	Car Body C	117	5
4/18/2018	19:28:06	7	50741	Female	Web	Member	Auto & Accessories	Car & Bike	118	1

Data Transformation

1. Convert dates and numeric fields to appropriate data types
2. Remove null values and duplicates
3. Calculate profit margin and categorize it
4. Create dimension tables for customers, products, and dates
5. Build a fact table for sales with calculated metrics.

Code Snippet

```
def transform_to_star_schema_and_save(input_file: str, output_dir: str, ti: TaskInstance):
    # Read the CSV file
    df = pd.read_csv(input_file)

    print("Original number of rows:", len(df))
    print("Columns in the input file:", df.columns.tolist())

    # Data Cleaning Steps
    # Convert Order_Date to datetime
    df['Order_Date'] = pd.to_datetime(df['Order_Date'], errors='coerce')

    # Remove rows with null values
    df.dropna(inplace=True)

    print("Number of rows after removing nulls:", len(df))

    # Convert numeric columns and handle potential errors
    numeric_columns = ['Sales', 'Quantity', 'Discount', 'Profit', 'Shipping_Cost']
    for col in numeric_columns:
        df[col] = pd.to_numeric(df[col], errors='coerce')

    # Remove duplicates
    df.drop_duplicates(inplace=True)

    print("Final number of rows:", len(df))

    # Calculate Profit Margin
    df['Profit_Margin'] = df['Profit'] / df['Sales']

    # Add Profit Margin Category
    df['Profit_Margin_Category'] = pd.cut(
        df['Profit_Margin'],
        bins=[-np.inf, 0, 0.1, 0.2, 0.3, np.inf],
        labels=['Loss', 'Low', 'Medium', 'High', 'Very High']
    )

    # Create dimension tables
    dim_customer = df[['Customer_Id', 'Gender', 'Customer_Login_type']].drop_duplicates().reset_index(drop=True)
    dim_customer['customer_key'] = dim_customer.index + 1
```

Transformation Tools and Considerations

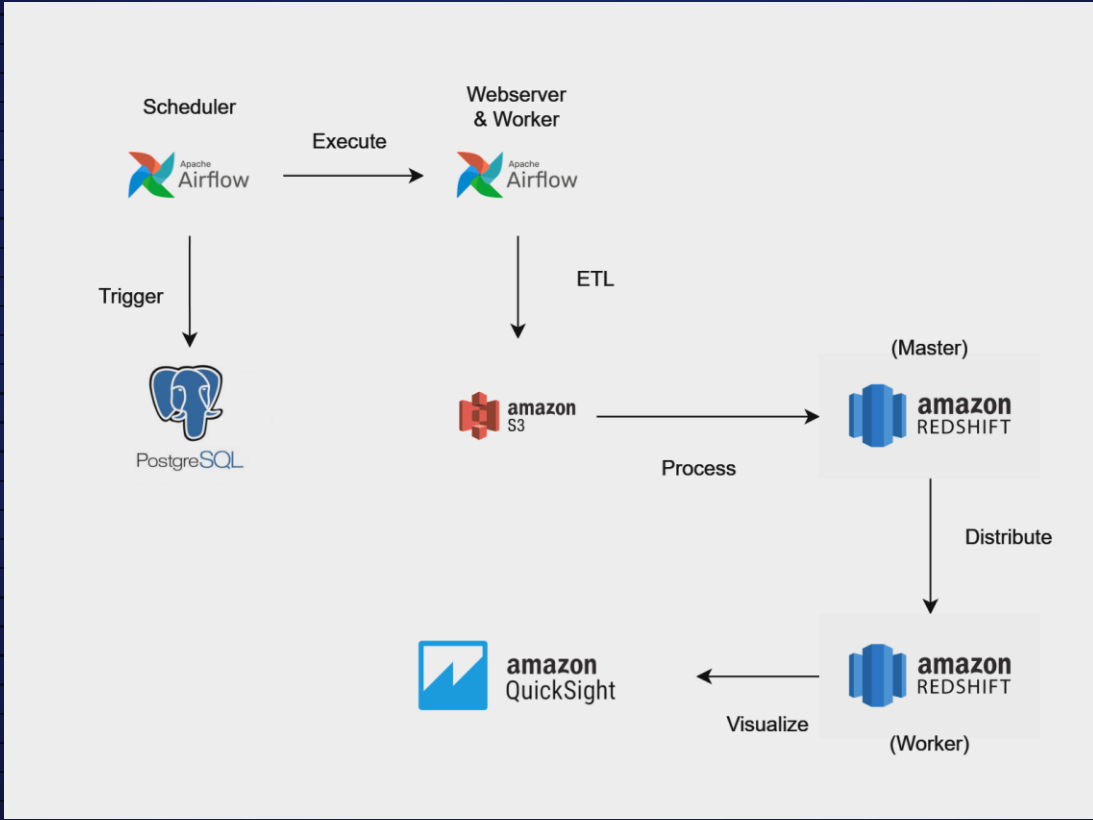
1. Python with Pandas
   Chosen for its powerful data manipulation capabilities and ease of use in ETL processes, offering more flexibility than SQL-based transformations alone.

2. Apache Airflow
   Selected for workflow orchestration, offering better scheduling and monitoring compared to cron jobs, with better error handling and dependency management.

3. AWS S3
   Used as a data lake for its scalability and seamless integration with other AWS services. S3 offers virtually unlimited storage and easy integration with Redshift, making it superior to on-premises storage solutions.

4. Amazon Redshift
   Preferred for its column-oriented storage and MPP (massively parallel processing) architecture, optimizing analytical query performance.

Architecture



Data Modelling

1. Star Schema Design
   Optimizes for quick analysis of sales performance across multiple dimensions.

2. Fact Table: Sales
   Central table capturing key metrics like revenue, profit, and quantity sold.

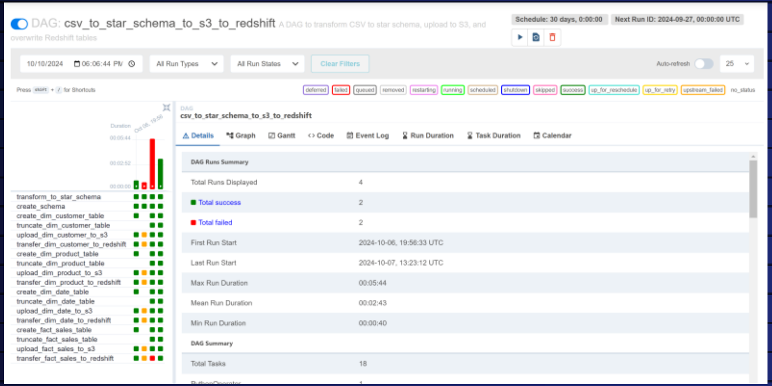
3. Dimension Tables
   a. Customer: Enables customer segmentation and behavior analysis
      Columns: Gender, Customer_Login_type, customer_key
   b. Product: Facilitates product category performance evaluation
      Columns: Product_Category, Product, product_key
   c. Date: Allows for time-based trend analysis and seasonality detection
      Columns: date_key, full_date, year, month, day, weekday

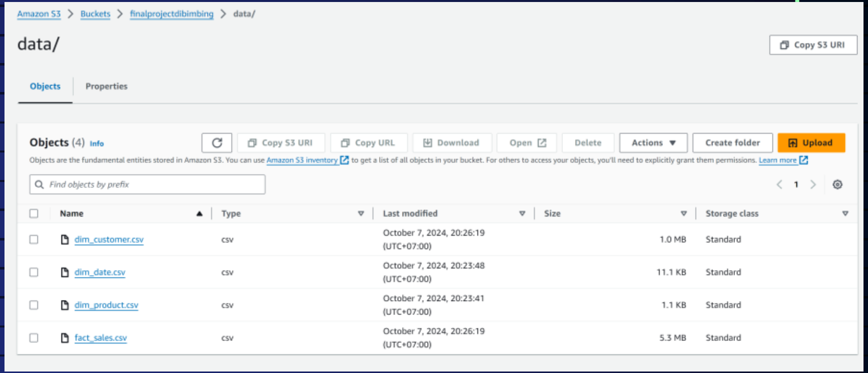
4. Key Performance Indicators (KPIs)Includes calculated metrics like profit margin and total revenue for easy reporting.

03

Result & Analysis

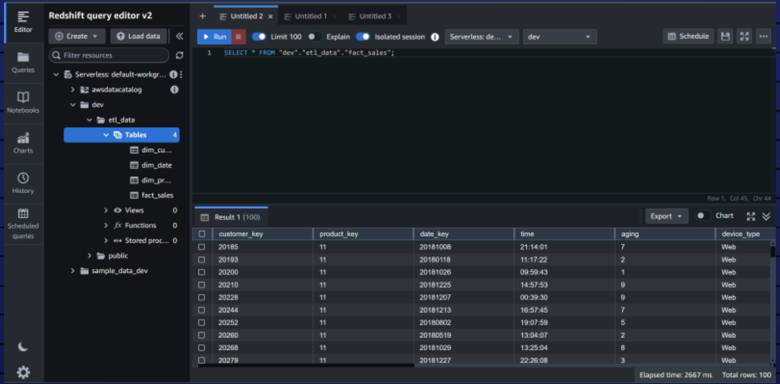
End Result





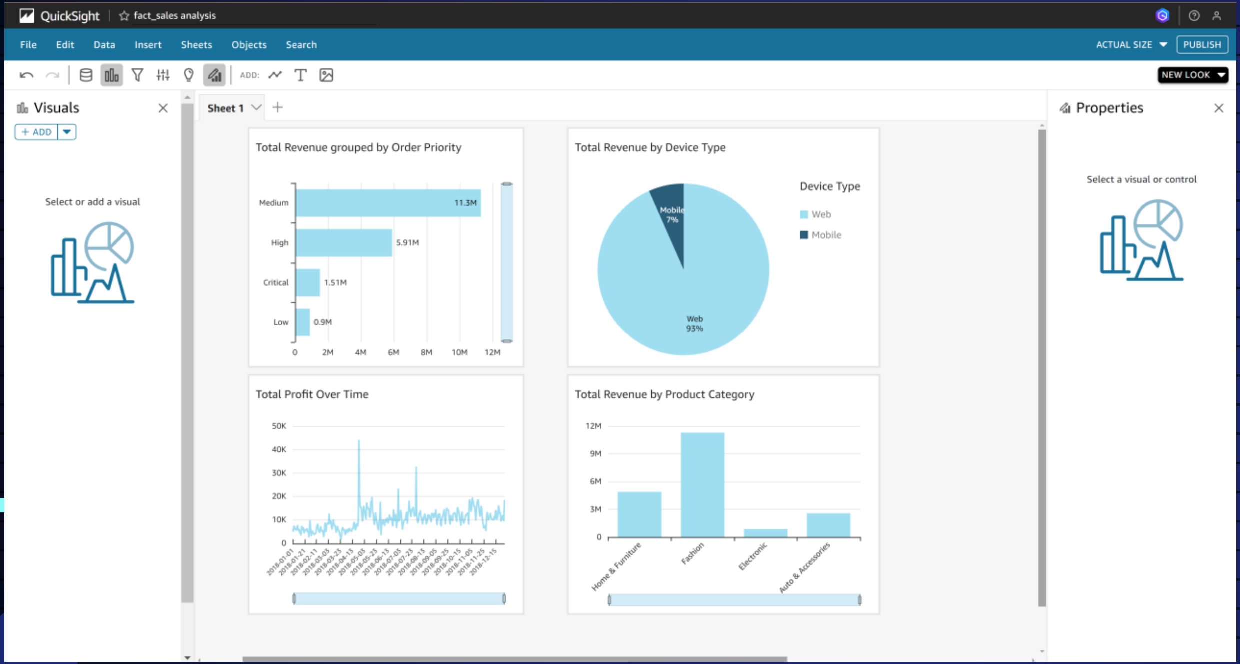
DAG Running

S3 Bucket



Data loaded into Redshift

Visualization in QuickSight



1. Medium priority orders generate the highest revenue at **11.3M.**

2. Web orders dominate with **93%** of total revenue.

3. Profit shows fluctuations with occasional high peaks between April-May.

4. Fashion category leads in revenue generation, followed by Home & Furniture.

04

Conclusion

Conclusion

1. Successful Implementation
   Automated ETL pipeline from CSV to Redshift established using python and airflow.

2. Improved Data Accessibility
   Star schema design enables efficient querying and analysis, comprised of 1 fact table and 3 dimension table.

3. Business Impact
   Enhanced ability to derive insights from sales data.

4. Scalable Solution
   Platform capable of handling growing data volumes.

5. Foundation for Advanced Analytics
   Prepared data structure supports future ML and AI initiatives.

Main Problems in the Project

1. Creating a connection for Amazon S3 and Redshift.

2. Authentication for AWS, creating a user and credentials (secret key and access key ID) that can be accessed by the docker container.

3. Security groups, VPC and Redshift networking so that the docker container can connect to AWS using the credentials.

Limitations & Recommendations

Limitations

- Potential for high costs with increasing data volumes
- Limited to structured CSV data from a single source

Recommendations

- Optimize AWS resource usage and explore reserved instances
- Integrate additional data types and sources for comprehensive analysis