AdventureWorks Business DB ETL Pipeline – Design Document

Authors: Gabriel Mancillas, Duy Nguyen, Jorge Roldan

Date: Feb 24, 2025

1. Repository Overview

- **GitHub Repository:**
  Your GitHub Repo URL Here
  This repository contains:
  - Python scripts for Extract, Transform, Load (ETL)
  - Database schema (.sql files)
  - Documentation for deployment and usage

2. Source Datasets

2.1. Dataset Origin & Rationale

We use AdventureWorks CSV files, which are a well-known sample dataset provided by Microsoft. The dataset includes tables such as:

- Employee.csv
- Vendor.csv
- ShipMethod.csv
- Product.csv
- PurchaseOrderHeader.csv
- PurchaseOrderDetail.csv
- Sales.csv (optional for simulation)
- SalesTarget.csv (monthly/quarterly targets)

- Customer.csv
- WeeklySalesSummary.csv (aggregated data)

Why Choose AdventureWorks?

- **Realistic Business Environment:** Simulates manufacturing, sales, and related processes.
- **Rich Relationships:** Ideal for practicing SQL joins, foreign keys, and building data warehouses.
- **Standard Example:** Widely recognized in the SQL community.

3. Pipeline Output

3.1. What the Pipeline Produces

Post-ETL execution, cleaned and transformed data are loaded into MySQL, enabling the following reports:

- **Weekly Sales:** Revenue, order counts, average sale values.
- **Purchase Order Analysis:** Vendor-product relations, spending totals, order statuses.
- **Employee & Sales Performance:** Comparison of SalesTarget vs. actual Sales.

3.2. Benefits of the Pipeline

- **Business Insights:** Informs decisions on inventory, vendor management, and employee performance.
- **Automation:** Ensures up-to-date metrics through scheduled reporting.
- **Scalability:** Designed to extend with additional tables and data sources.

4. Architecture Diagram

```
            ┌──────────────┐
            │ CSV Datasets │
            │(Employee.csv,│
            │ Vendor.csv,  │
            │    ...     )│
            └─────┬────────┘
```

```
          │  (Extract)
          ▼
┌──────────────────────┐
│  Python ETL Scripts  │
│(extract, transform,  │
│        load)         │
└──────────────────────┘
      │  (Transform/Load)
      ▼
┌─────────────────────────┐
│    MySQL Database       │
│ (Cleaned & Final Tables)│
└─────────────────────────┘
      │  (Query for Reports)
      ▼
┌──────────────────────────┐
│   Automated Reports      │
│  (Sales, PO analysis,    │
│   Employee insights)     │
└──────────────────────────┘
```

5. Final Schema Diagram

Key Tables and Relationships:

- **Employee:** EmployeeID (PK)
- **Vendor:** VendorID (PK)
- **ShipMethod:** ShipMethodID (PK)
- **Product:** ProductID (PK)
- **PurchaseOrderHeader:**
  - PurchaseOrderID (PK)

  - EmployeeID (FK → Employee(EmployeeID))
  - VendorID (FK → Vendor(VendorID))
  - ShipMethodID (FK → ShipMethod(ShipMethodID))
- **PurchaseOrderDetail:**
  - PurchaseOrderDetailID (PK)
  - PurchaseOrderID (FK → PurchaseOrderHeader(PurchaseOrderID))
  - ProductID (FK → Product(ProductID))
- **Sales:**
  - SaleID (PK)
  - EmployeeID (FK → Employee(EmployeeID))
  - CustomerID (FK → Customer(CustomerID)) *(optional)*
- **SalesTarget:**
  - SalesTargetID (PK) or composite key (EmployeeID, Year, Month)
  - EmployeeID (FK → Employee(EmployeeID))
- **WeeklySalesSummary:**
  - Composite Key: (Year, Week)
- **Customer:** CustomerID (PK)

(If additional relationships or bridging tables are necessary (e.g., for ShipMethod ↔ Product), adjust accordingly.)

---

6. System Considerations and Future Improvements

6.1. Scalability

- **Current Approach:**
  A single MySQL instance with Python-based ETL handles moderate data volumes.
- **Potential Bottlenecks and Enhancements:**
  - Table sharding or partitioning for scaling.

    - Migrating to distributed databases (e.g., Amazon Redshift, BigQuery).
    - Implementing chunk-based or incremental loading to optimize performance.

6.2. Security

```
- AWS RDS & RDS Proxy:
- The RDS Proxy can manage database credentials via AWS Secrets Manager, limiting direct
access to the DB.
•       Connections can be IAM-authenticated or use token-based security.
•       Token-Based Access:
•       The application can request temporary credentials/tokens from IAM, reducing the
need to store static passwords.
•       Network:
•       Deploy RDS in a private subnet; only the proxy endpoint is exposed to the
application.
•       Use security groups to restrict inbound connections to known IPs or VPC
resources.
•       Encryption:
•       At Rest: Use KMS to encrypt RDS data.
•       In Transit: Enforce SSL/TLS connections between the application and the proxy.
```

6.3. Extensibility

- **Adding New Tables:**
  The schema is flexible and can integrate additional CSV inputs.
- **Adapting Transformations:**
  Python scripts are modular, allowing new transformation functions.
- **Alternate Data Outputs:**
  Possibility to connect with BI dashboards or load data into a data warehouse for advanced analytics.

---

7. Conclusion

This design document describes a robust ETL pipeline that: 1. Extracts AdventureWorks CSV data. 2. Transforms it via Python scripts. 3. Loads final tables into AWS RDS (MySQL) behind an RDS Proxy. 4. Secures credentials using token-based access and AWS Secrets Manager. 5. Produces automated reports on sales, purchase orders, employee metrics, etc.

Strengths: • AWS-based deployment with RDS Proxy improves security and performance. • Token-based authentication removes the need for static credentials. • Pipeline is modular, allowing easy extension.

Areas for Improvement: • Scaling for very large datasets may require read replicas or a data warehouse approach. • Additional security layers (audit logging, stricter IAM roles) can further reduce risk.

This pipeline meets typical business intelligence needs and is structured for future expansion and scalability.