Author: Gabriel Mancillas, Duy Nguyen, Jorge Roldan. Date: Feb 24, 2025

AdventureWorks Business DB ETL Pipeline – Design Document

1. GitHub Repository

GitHub Link: [Your GitHub Repo URL Here] (Example: https://github.com/Neo-and-Company/Ads507)

This repository contains all source code (Python scripts for Extract, Transform, Load), the database schema (.sql files), and documentation for deploying the pipeline.

2. Source Datasets

2.1. Dataset Origin & Rationale

We use AdventureWorks CSV files (commonly available as a sample database from Microsoft). The CSVs include realistic tables such as: • Employee.csv • Vendor.csv • ShipMethod.csv • Product.csv • PurchaseOrderHeader.csv • PurchaseOrderDetail.csv • Sales.csv (optional if you're simulating sales) • SalesTarget.csv (for monthly/quarterly targets) • Customer.csv (for customer info) • WeeklySalesSummary.csv (aggregated data if needed)

Why This Dataset? • Realistic Business Structure: AdventureWorks simulates a manufacturing and sales environment with employees, products, vendors, purchase orders, etc. • Multiple Tables & Relationships: Perfect for practicing SQL joins, foreign keys, and typical data warehouse or reporting tasks. • Widely Known: AdventureWorks is a standard example in the SQL community, making it easy to demonstrate ETL best practices.

3. Output of the Pipeline

3.1. What the Pipeline Produces

After running the ETL scripts, we load cleaned and transformed data into MySQL. Some common reports or outputs include: • Weekly Sales: Summaries of total revenue, total orders, average sale value, etc. • Purchase Order Analysis: Which vendors supply which products, total spending per vendor, order statuses. • Employee & Sales Performance: Using SalesTarget vs. actual Sales data to see which employees meet or exceed quotas.

3.2. Why the Output is Useful

```
•       Business Insights: Helps managers make decisions on inventory, vendor
negotiations, and employee performance.
•       Automation: Weekly or daily reports reduce manual effort, ensuring stakeholders
always have up-to-date metrics.
•       Scalability: The pipeline design can be extended to other tables or additional
data sources if the business grows.
```

4. Architecture Diagram

Below is a simplified architecture showing how data flows from CSV to MySQL and how it's used for reporting.

```
         ┌──────────────┐
         │ CSV Datasets  │
         │(Employee.csv, │
         │ Vendor.csv,   │
         │ etc.)         │
         └──────┬───────┘
                │
      (Extract) │
                ▼
     ┌───────────────────────┐
     │  Python Scripts        │
     │(extract, transform,    │
     │       load)            │
     └─────┬─────────────────┘
           │ (Transform/Load)
           ▼
     ┌──────────────────────────┐
     │     MySQL Database        │
     │ (Cleaned, Final Tables)   │
     └─────┬────────────────────┘
           │
```

```
            │ (Query for Reports)
            ▼
      ┌─────────────────────────┐
      │   Automated Reports     │
      │ (Sales, PO, Employee,   │
      │  Weekly Summaries)      │
      └─────────────────────────┘
```

---

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

- **Current Measures:**
    - Use of environment variables or secrets management for credentials.
    - MySQL behind a firewall/security group.
    - Enforcing SSL/TLS for connections.
- **Future Enhancements:**

    - Deploy MySQL in a private subnet to restrict public access.
    - Implement role-based access controls (e.g., read-only vs. admin).
    - Introduce audit logging for critical database operations.

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 outlines a robust ETL pipeline that:

1. Extracts AdventureWorks CSV data.
2. Transforms and cleans data using Python.
3. Loads the final dataset into MySQL.
4. Generates automated reports covering sales, purchase orders, employee metrics, etc.

Strengths:

- Simple and replicable design
- Automated reporting reduces manual processing

Areas for Improvement:

- Enhanced scalability for high data volumes.
- Advanced security practices for risk mitigation.

With these design choices, the pipeline meets typical business needs for automated reporting, while leaving room for future expansion and improvements.

End of Document