Sales Analytics ETL Project

1. Project Overview

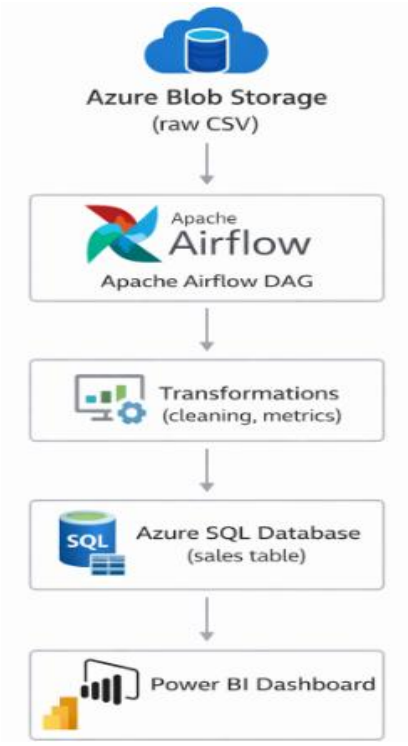
This project implements an **end-to-end ETL (Extract, Transform, Load) pipeline** using **Apache Airflow** and **Azure services**, followed by **analytical visualization in Power BI**.

The objective is to simulate a real-world data engineering workflow where raw sales data is ingested, cleaned, transformed, stored in a relational database / data warehouse, and finally analyzed through dashboards.

2. Architecture Overview

High-level flow:

1. **Data Source** – CSV files stored in Azure Blob Storage
2. **Orchestration** – Apache Airflow
3. **Transformations** – Python (Pandas)
4. **Data Warehouse / Storage** – Azure SQL Database
5. **Analytics & Visualization** – Power BI



3. Technology Stack

Layer	Technology
Orchestration	Apache Airflow
Programming	Python 3.10
Storage (Raw)	Azure Blob Storage
Database	Azure SQL Database
Analytics	Power BI
OS	WSL (Ubuntu)

4. Data Description

Raw Data (CSV)

Column	Description
order_id	Unique order identifier
product	Product name
category	Product category
quantity	Number of items ordered
price	Price per unit
order_date	Order date

5. ETL Design

5.1 Extract

- Source: Azure Blob Storage container (raw)
- Format: CSV
- Airflow task reads data using Python SDK / Pandas

Purpose:

- Bring raw data into the pipeline without modification

5.2 Transform

Transformations applied:

1. Remove rows with null values
2. Remove invalid records:
    - quantity ≤ 0
    - price ≤ 0
3. Create derived metrics:
    - revenue = quantity * price
4. Standardize column types
5. Add analytical attributes:
    - Order Value Category (Low / Medium / High)

These transformations ensure **data quality and analytics readiness**.

5.3 Load

- Target: Azure SQL Database
- Table: sales
- Mode: Append (supports incremental loads)

Purpose:

- Store clean, structured data for reporting and BI tools

6. Apache Airflow DAG Design

DAG Name

etl_sales_pipeline

Tasks

1. **extract_task**
    - Reads CSV from Blob Storage
    - Writes raw file locally (temporary staging)
2. **transform_task**
    - Cleans and enriches data
    - Produces final dataset
3. **load_task**
    - Inserts data into Azure SQL Database

DAG Properties

- Schedule: Daily
- Catchup: Disabled
- Idempotent logic supported

7. Data Model (Current)

Sales Table

Column	Type
order_id	INT
product	VARCHAR
category	VARCHAR
quantity	INT
price	FLOAT

Column	Type
revenue	FLOAT
order_date	DATE
order_value_category	VARCHAR

Note: A star schema (fact + dimensions) can be added later when scaling analytics.

8. Incremental Data Strategy

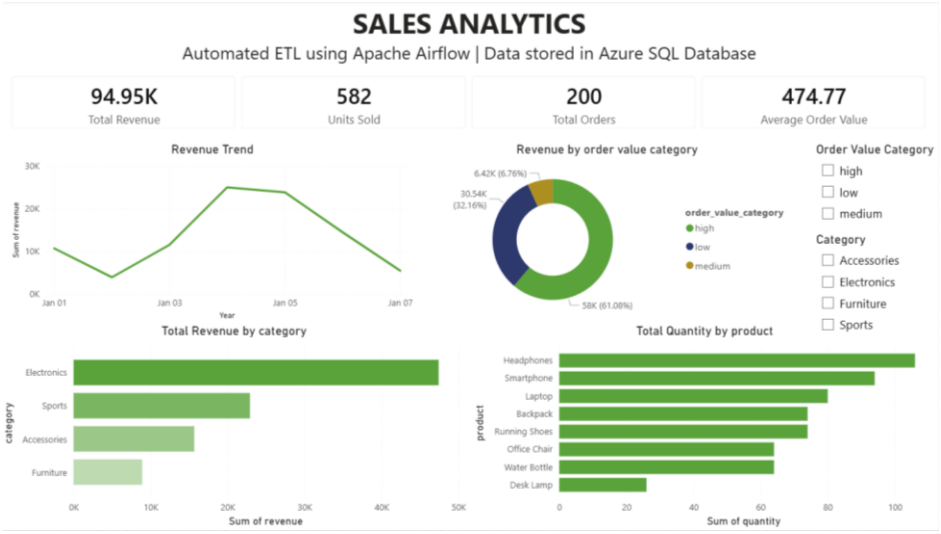
- New CSV files uploaded to Blob Storage
- Airflow DAG scheduled to run daily
- Logic ensures only new records are inserted
- No manual intervention required

This simulates **production-grade automation**.

9. Power BI Integration

Connection

- Source: Azure SQL Database
- Authentication: SQL Authentication



KPIs

- Total Revenue
- Total Orders
- Total Quantity
- Average Order Value

Visuals

- Revenue Trend (Line chart)
- Revenue by Category (Bar chart)
- Quantity by Product (Bar chart)
- Revenue by Order Value Category (Donut chart)

11. Dashboard Purpose

The dashboard enables:

- Executive-level insights
- Trend analysis
- Product and category performance evaluation
- Data-driven decision making

12. Future Enhancements

- Star schema (fact_sales + dimensions)
- Incremental loading with watermarking
- Data quality checks
- Alerts on pipeline failure
- CI/CD for DAG deployment
- Azure Synapse Analytics integration
- Azure Data Factory integration

13. Key Learning Outcomes

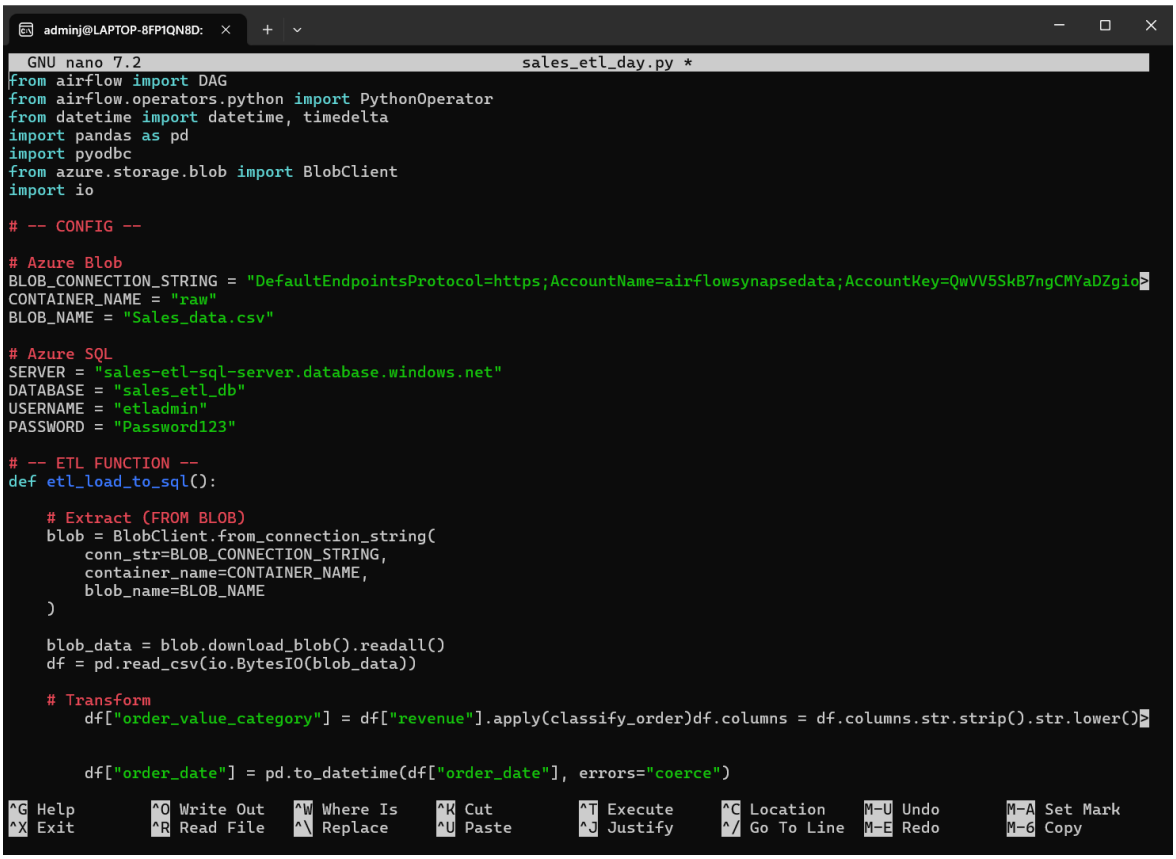
- End-to-end ETL pipeline design
- Real-world Airflow usage
- Azure cloud integration
- Data modeling fundamentals
- BI dashboard creation

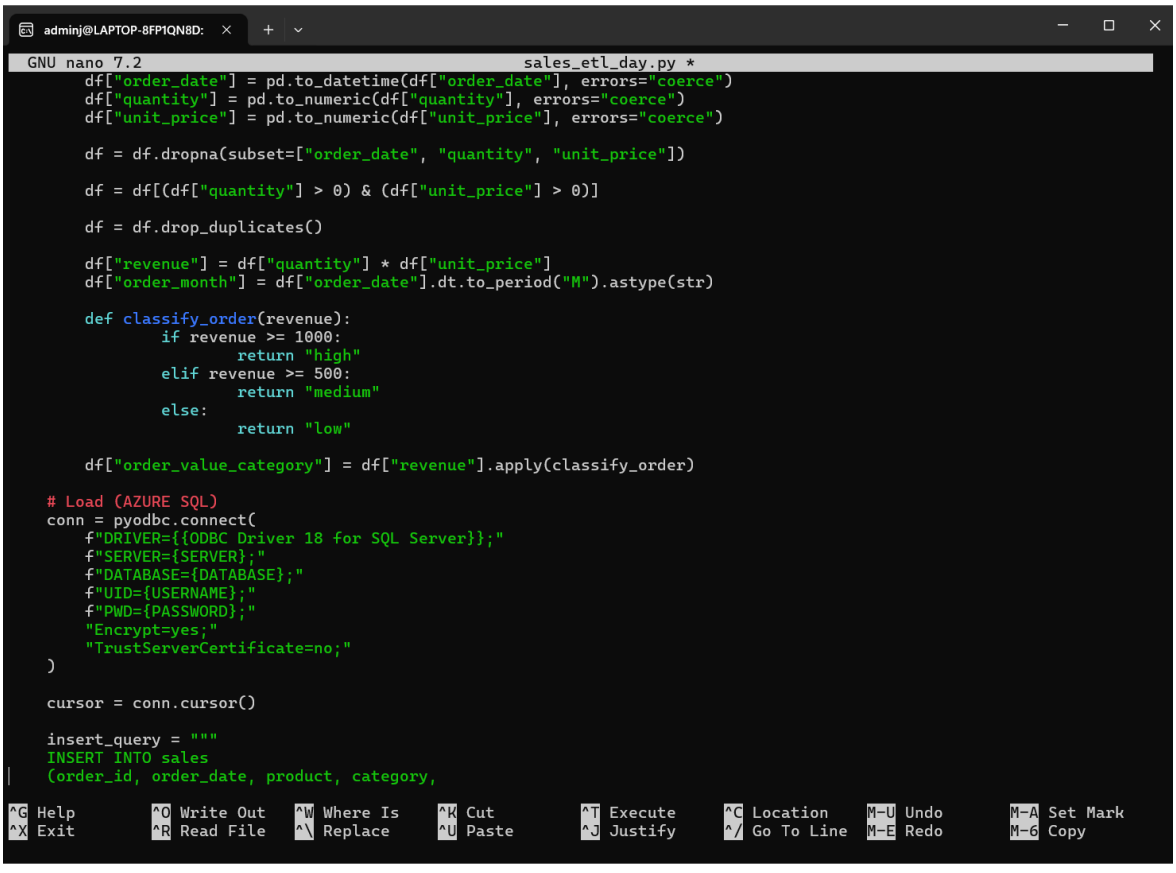
14. Conclusion

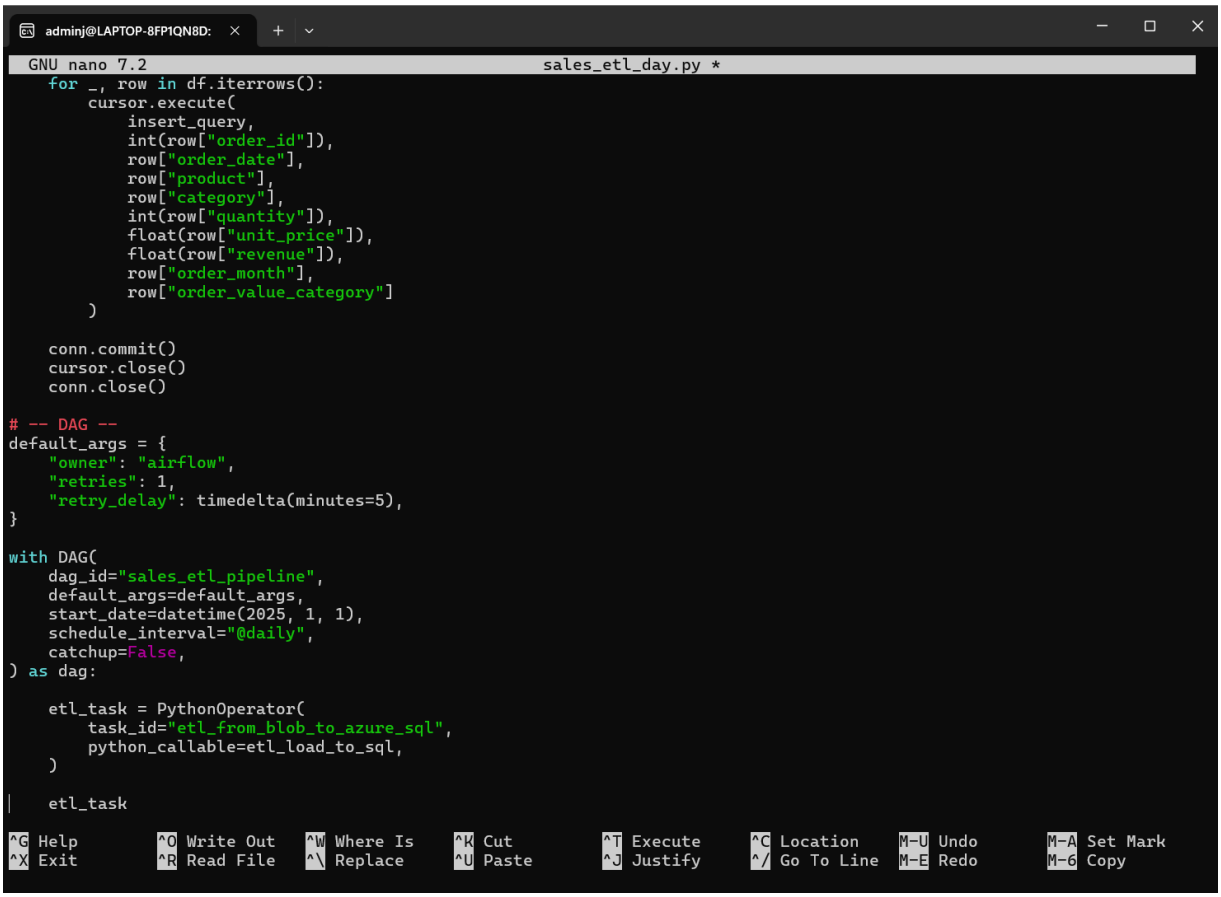
This project demonstrates a **complete data engineering lifecycle**, from raw data ingestion to business intelligence reporting. It reflects industry-aligned practices and provides a strong foundation for advanced analytics and data warehousing projects.

15. Author

Jeswin K Reji
Aspiring Data Scientist
December 2025
jeswinkr7@gmail.com
LinkedIn GitHub


adminj@LAPTOP-8FP1QN8D:
GNU nano 7.2 sales_etl_day.py *
from airflow import DAG
from airflow.operators.python import PythonOperator
from datetime import datetime, timedelta
import pandas as pd
import pyodbc
from azure.storage.blob import BlobClient
import io
# -- CONFIG --
# Azure Blob
BLOB_CONNECTION_STRING = "DefaultEndpointsProtocol=https;AccountName=airflowsynapsedata;AccountKey=QwVV5SkB7ngCMYaDZgio>
CONTAINER_NAME = "raw"
BLOB_NAME = "Sales_data.csv"
# Azure SQL
SERVER = "sales-etl-sql-server.database.windows.net"
DATABASE = "sales_etl_db"
USERNAME = "etladmin"
PASSWORD = "Password123"
# -- ETL FUNCTION --
def etl_load_to_sql():
# Extract (FROM BLOB)
blob = BlobClient.from_connection_string(
conn_str=BLOB_CONNECTION_STRING,
container_name=CONTAINER_NAME,
blob_name=BLOB_NAME
)
blob_data = blob.download_blob().readall()
df = pd.read_csv(io.BytesIO(blob_data))
# Transform
df["order_value_category"] = df["revenue"].apply(classify_order)df.columns = df.columns.str.strip().str.lower()>
df["order_date"] = pd.to_datetime(df["order_date"], errors="coerce")
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location M-U Undo M-A Set Mark
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line M-E Redo M-6 Copy


adminj@LAPTOP-8FP1QN8D:
GNU nano 7.2 sales_etl_day.py *
df["order_date"] = pd.to_datetime(df["order_date"], errors="coerce")
df["quantity"] = pd.to_numeric(df["quantity"], errors="coerce")
df["unit_price"] = pd.to_numeric(df["unit_price"], errors="coerce")
df = df.dropna(subset=["order_date", "quantity", "unit_price"])
df = df[(df["quantity"] > 0) & (df["unit_price"] > 0)]
df = df.drop_duplicates()
df["revenue"] = df["quantity"] * df["unit_price"]
df["order_month"] = df["order_date"].dt.to_period("M").astype(str)
def classify_order(revenue):
if revenue >= 1000:
return "high"
elif revenue >= 500:
return "medium"
else:
return "low"
df["order_value_category"] = df["revenue"].apply(classify_order)
# Load (AZURE SQL)
conn = pyodbc.connect(
f"DRIVER={{ODBC Driver 18 for SQL Server}};"
f"SERVER={SERVER};"
f"DATABASE={DATABASE};"
f"UID={USERNAME};"
f"PWD={PASSWORD};"
"Encrypt=yes;"
"TrustServerCertificate=no;"
)
cursor = conn.cursor()
insert_query = """
INSERT INTO sales
(order_id, order_date, product, category,
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location M-U Undo M-A Set Mark
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line M-E Redo M-6 Copy


adminj@LAPTOP-8FP1QN8D:
GNU nano 7.2 sales_etl_day.py *
for _, row in df.iterrows():
cursor.execute(
insert_query,
int(row["order_id"]),
row["order_date"],
row["product"],
row["category"],
int(row["quantity"]),
float(row["unit_price"]),
float(row["revenue"]),
row["order_month"],
row["order_value_category"]
)
conn.commit()
cursor.close()
conn.close()
# -- DAG --
default_args = {
"owner": "airflow",
"retries": 1,
"retry_delay": timedelta(minutes=5),
}
with DAG(
dag_id="sales_etl_pipeline",
default_args=default_args,
start_date=datetime(2025, 1, 1),
schedule_interval="@daily",
catchup=False,
) as dag:
etl_task = PythonOperator(
task_id="etl_from_blob_to_azure_sql",
python_callable=etl_load_to_sql,
)
etl_task
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location M-U Undo M-A Set Mark
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line M-E Redo M-6 Copy