MoP - ETL Mini-Project (Customer Profile + Complaints Integration using MySQL)

1) **Objective**: You work for a telecom company that needs to combine customer profile data from a SQL database (structured dataset) with complaint records (unstructured dataset) stored in CSV files. Your task is to create a merged dataset that enables analysts to identify which customer segments generate the most complaints.

Students will:

- Connect Python to a MySQL database.
- Read telecom customer data from MySQL.
- Merge it with complaint data from a CSV file.
- Clean and standardize data (without deleting rows).
- Save the merged dataset for analysis in PowerBI/Tableau.

1) **Prerequisites (Setup)**

Software Required

Tool Purpose

Python 3 (Anaconda recommended) For scripting
MySQL Server (v8 or above) For storing customer data
MySQL Workbench For easy database management
Jupyter Notebook For coding
Python Libraries pandas, sqlalchemy, pymysql

1.2 Install Required Packages

Run in Anaconda Prompt or Terminal:

pip install pandas sqlalchemy pymysql jupyter

1.3 Start MySQL and use database we created in mysql MOP i.e. testdb

Open MySQL Workbench or terminal and run:

USE testdb;

1.4 update the existing customers table with region column from usage_data table

Update customers Table

-- Add region column to customers table

ALTER TABLE customers ADD COLUMN region VARCHAR(20);

-- Disable safe updates and update region

SET SQL_SAFE_UPDATES = 0; (#MySQL Workbench runs in *safe update mode* by default to protect you from accidentally updating or deleting large tables without a WHERE clause **that filters by a key (indexed column)**. – disable safe mode)

-- Update region based on usage_data
UPDATE customers c
JOIN usage_data u ON c.customer_id = u.customer_id
SET c.region = u.region;

-- View final result
SELECT * FROM customers;

1001 Asha Mehta Prepaid 2023-05-12 Delhi

1002 Ravi Kumar Postpaid 2022-12-20 Mumbai

1003 Sneha Rao Prepaid 2023-01-18 Chennai

1004 Manoj Singh Postpaid 2021-11-05 Delhi

1005 Divya Jain Prepaid 2023-03-28 Kolkata

You now have a MySQL table testdb.customers.

2. Create complaints.csv in Jupyter

Launch Jupyter

- Open Anaconda Navigator → Jupyter
- In the browser tab, navigate into test_p/.
- Open notebook -> training.ipynb we created earlier

import pandas as pd

complaints = pd.DataFrame([

{"complaint_id":"CMP-001","customer_id":1002,"category":"Billing","description":"Charged extra for data usage","created_at":"2025/09/25 10:45","status":"Open"},

{"complaint_id":"CMP-002","customer_id":1004,"category":"Network","description":"Frequent call drops in Delhi","created_at":"2025-09-25 09:30","status":"Open"},

{"complaint_id":"CMP-003","customer_id":1005,"category":"Recharge","description":"Recharge failed; amount deducted","created_at":"25-09-2025 14:00","status":"Closed"},

{"complaint_id":"CMP-004","customer_id":1002,"category":"Network","description":"Slow 4G speed at night","created_at":"2025-09-26 20:40","status":"Open"},

{"complaint_id":"CMP-005","customer_id":1003,"category":"Support","description":"No response to complaint","created_at":"2025-09-26 11:10","status":"Open"}

])

complaints.to_csv("complaints.csv", index=False)

print("✅ complaints.csv saved.")

You now have customer complaints.

3) Build the ETL Pipeline

3.1 Imports and Database Connection

```
import pandas as pd
from sqlalchemy import create_engine
# Replace with your actual credentials
user = 'root'
password = 'admin'
host = 'localhost'
database = 'testdb'
# Create SQLAlchemy engine
engine = create_engine(f"mysql+pymysql://{user}:{password}@{host}/{database}")
# Check connection
print("✅ Connected to MySQL successfully!")
```

You are now connected to MYSQL via Python!!!!!!

3.2 **EXTRACT Data**

```
# Extract customers from MySQL
customers = pd.read_sql("SELECT * FROM customers", engine)
# Extract complaints from CSV
complaints = pd.read_csv("complaints.csv")
print("Rows extracted -> Customers:", len(customers), "Complaints:", len(complaints))
display(customers.head(), complaints.head())
```

You have now extracted both forms of datasets

3.3 **TRANSFORM Data (Clean, Fix, and Merge)**

```
# --- Standardize text ---
customers['region']  = customers['region'].str.title().str.strip()
complaints['status'] = complaints['status'].str.title().str.strip()
complaints['category'] = complaints['category'].str.title().str.strip()

# --- Parse and standardize dates ---
customers['join_date']  = pd.to_datetime(customers['join_date'],  errors='coerce')
complaints['created_at'] = pd.to_datetime(complaints['created_at'], errors='coerce')

# Fill unparseable or missing dates
default_dt = pd.Timestamp('2025-09-25 00:00')
customers['join_date']  = customers['join_date'].fillna(default_dt)
complaints['created_at'] = complaints['created_at'].fillna(default_dt)

# --- Fix missing IDs or text ---
complaints['customer_id'] = complaints['customer_id'].fillna(-1).astype(int)
complaints['description'] = complaints['description'].fillna("No description provided")

# --- Merge ---
merged = customers.merge(complaints, on='customer_id', how='left')

# --- Post-merge fixes ---
merged['complaint_id'] = merged['complaint_id'].fillna("NO-COMPLAINT")
merged['category']     = merged['category'].fillna("No Complaint")
merged['status']       = merged['status'].fillna("Resolved")
merged['created_at']   = merged['created_at'].fillna(default_dt)
1
# Derived flag
merged['is_open'] = (merged['status'] == 'Open')
print(" ✅ Data transformed successfully!")
merged.head()
```

You have now transformed the datasets into uniform info

3.4 **LOAD (Save the Output)**

```
merged.to_csv("etl_output.csv", index=False)

print(f" ✅ ETL pipeline complete! Created {len(merged)} records and saved etl_output.csv.")
```

4) Validation & Quick Analytics

```
print("\nComplaints per customer:")
print(merged.groupby(['customer_id','name']).complaint_id.count().reset_index(name='compl
aint_count'))

print("\nOpen vs Closed:")
```

```
print(merged['status'].value_counts())

print("\nComplaints by region & category:")
print(merged.groupby(['region','category']).complaint_id.count().reset_index(name='count'))
```

✅ Expected:

- 5 customers visible
- “NO-COMPLAINT” for customers without complaints
- Consistent dates in YYYY-MM-DD format
- No deleted rows

5) Deliverables

Students should submit on google drive:

- xxxxx.ipynb
- complaints.csv
- etl_output.csv
- Screenshot of MySQL customers table
- A short summary (3–5 lines) of your learning from data - (e.g., “Network complaints are concentrated among Postpaid customers in Delhi; prioritize capacity checks and proactive outreach”).

MOP ETL pipeline – USING SQLITE (instead of MYSQL)

HOMEWORK – EXTRA EXERCISE – OPTIONAL

It includes exact steps, copy-paste code, and validation checks, and it fixes bad/missing data rather than deleting it.
MoP — ETL Mini-Project: Customer Profile + Complaints Integration

Goal (what you will produce)
Create an analysis-ready merged dataset by combining:

- Customers table from a small SQLite database, and
- Complaints from a CSV file,

…then save the result as etl_output.csv for dashboards/analysis.

1) Prerequisites
Tools (free):

- Python 3 (Anaconda recommended) + Jupyter Notebook
- Packages: pandas, sqlalchemy
- SQLite (built-in with Python)

Install (if needed):
pip install pandas sqlalchemy jupyter
Folder:
Create a working folder: telecom_etl/ and open it in Jupyter.

2) Create sample data (once)

2.1 Create the SQLite DB with a customers table
Run in a Jupyter cell:

```
import sqlite3
conn = sqlite3.connect("telecom.db")
cur = conn.cursor()
cur.execute("""
CREATE TABLE IF NOT EXISTS customers (
  customer_id INTEGER PRIMARY KEY,
  name TEXT, plan_type TEXT, region TEXT, join_date TEXT
);
""")
cur.executemany("""
INSERT OR REPLACE INTO customers (customer_id, name, plan_type, region, join_date)
VALUES (?,?,?,?,?)
""", [
 (1001,"Asha Mehta","Prepaid","Delhi","2023-05-12"),
 (1002,"Ravi Kumar","Postpaid","Mumbai","2022-12-20"),
 (1003,"Sneha Rao","Prepaid","Chennai","2023-01-18"),
 (1004,"Manoj Singh","Postpaid","Delhi","2021-11-05"),
 (1005,"Divya Jain","Prepaid","Kolkata","2023-03-28"),
])
```

```
conn.commit(); conn.close()
print("✅ telecom.db ready.")

2.2 Create a complaints.csv
import pandas as pd
complaints = pd.DataFrame([
  {"complaint_id":"CMP-001","customer_id":1002,"category":"Billing","description":"Charged
extra for data usage","created_at":"2025/09/25 10:45","status":"Open"},
  {"complaint_id":"CMP-
002","customer_id":1004,"category":"Network","description":"Frequent call drops in
Delhi","created_at":"2025-09-25 09:30","status":"Open"},
  {"complaint_id":"CMP-
003","customer_id":1005,"category":"Recharge","description":"Recharge failed; amount
deducted","created_at":"25-09-2025 14:00","status":"Closed"},
  {"complaint_id":"CMP-004","customer_id":1002,"category":"Network","description":"Slow 4G
speed at night","created_at":"2025-09-26 20:40","status":"Open"},
  {"complaint_id":"CMP-005","customer_id":1003,"category":"Support","description":"No
response to complaint","created_at":"2025-09-26 11:10","status":"Open"}
])
complaints.to_csv("complaints.csv", index=False)
print("✅ complaints.csv saved.")

3) Build the ETL (Extract → Transform → Load)
Create a new notebook (or new section) called etl_pipeline.

3.1 Imports
import pandas as pd
from sqlalchemy import create_engine

3.2 EXTRACT (pull data from DB + CSV)
# Customers from SQLite
engine = create_engine("sqlite:///telecom.db")
customers = pd.read_sql("SELECT * FROM customers", engine)
# Complaints from CSV
complaints = pd.read_csv("complaints.csv")
print("Rows -> customers:", len(customers), " complaints:", len(complaints))
customers.head(), complaints.head()

3.3 TRANSFORM (clean, standardize, and fix data)
We correct bad/missing values instead of dropping rows.
# --- Standardize text ---
customers['region']  = customers['region'].str.title().str.strip()
complaints['status'] = complaints['status'].str.title().str.strip()
complaints['category'] = complaints['category'].str.title().str.strip()

# --- Parse and standardize dates (multiple formats handled) ---
customers['join_date']  = pd.to_datetime(customers['join_date'],  errors='coerce')
```

```
complaints['created_at'] = pd.to_datetime(complaints['created_at'], errors='coerce')

# Fill any unparseable dates with a sensible default (keeps rows intact)
default_dt = pd.Timestamp('2025-09-25 00:00')
customers['join_date']  = customers['join_date'].fillna(default_dt)
complaints['created_at'] = complaints['created_at'].fillna(default_dt)

# --- Fix missing IDs or text (rare in this sample, but robust) ---
# If a complaint is missing customer_id, set to -1 and keep it for audit
complaints['customer_id'] = complaints['customer_id'].fillna(-1).astype(int)
complaints['description'] = complaints['description'].fillna("No description provided")

# --- Merge: keep ALL customers (left join), attach complaints where present ---
merged = customers.merge(complaints, on='customer_id', how='left')

# --- Post-merge fixups for missing complaint fields (no deletion) ---
merged['complaint_id'] = merged['complaint_id'].fillna("NO-COMPLAINT")
merged['category']     = merged['category'].fillna("No Complaint")
merged['status']       = merged['status'].fillna("Resolved")
merged['created_at']   = merged['created_at'].fillna(default_dt)

# Derived flag for analytics
merged['is_open'] = (merged['status'] == 'Open')
```

Why this is robust:

- Dates in different formats are parsed; failed parses get a default date (not dropped).
- Missing complaint fields are filled with clear placeholders.
- All customers remain in the final dataset (left join).

3.4 LOAD (save the integrated dataset)

```
merged.to_csv("etl_output.csv", index=False)
print(f" ✅ Pipeline complete! Created dataset with {len(merged)} rows and {merged.shape[1]} columns.")
```

4) Validate & sanity-check

```
print(merged.head())
print("\nComplaints per customer:")
print(merged.groupby(['customer_id','name']).complaint_id.count().reset_index(name='complaint_count'))
print("\nOpen vs Closed (including no-complaint as 'Resolved'):")
print(merged['status'].value_counts())
print("\nComplaints by category and region:")
print(merged.groupby(['region','category']).complaint_id.count().reset_index(name='count'))
```

Success criteria

- All 5 customers appear (one row per customer per complaint; some customers may repeat if they have multiple complaints).

- Customers with no complaints show NO-COMPLAINT / No Complaint / Resolved.
- Dates look consistent (YYYY-MM-DD hh:mm).
- No rows were deleted during cleaning.

5) Deliverables (what students submit)

- telecom.db
- complaints.csv
- etl_output.csv
- Notebook: etl_pipeline.ipynb
- A 3–5 line insight note