Final Project Queries

February 13, 2025

[1]:
```
# Importing libraries
import logging
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import pymysql
import ssl
from sqlalchemy import create_engine
from sqlalchemy import text
```

1 Extract

Retrieving data from Azure

[2]:
```
# Connection configuration for MySQL Azure Database
host = "airplane123.mysql.database.azure.com"
port = 3306
user = "student"
password = "Ads507password"
database = "airline"

# Configure logging
logging.basicConfig(filename='etl_monitoring.log', level=logging.INFO,␣
 ↪format='%(asctime)s - %(levelname)s - %(message)s')

try:
    logging.info("Starting ETL Pipeline...")

    # Connect to MySQL
    engine = create_engine(
        f"mysql+pymysql://{user}:{password}@{host}:{port}/{database}"
    )
    print("Connected to airline database on Azure without SSL")
except Exception as e:
    print("Connection failed:", e)
    logging.error(f"Error in data extraction: {e}")
```

```
Connected to airline database on Azure without SSL
```

Loading data using Pandas library into a Dataframe for data manipulation

[3]:
```
# Extract Data
logging.info("Extracting data from MySQL...")

# Query to get the data from the database
df = pd.read_sql("SHOW TABLES", engine)
logging.info(f"Extracted {len(df)} rows successfully.")
print(df)

# Loading each table into a dataframe
df_passengers = pd.read_sql("SELECT * FROM passengers", engine)
df_flights = pd.read_sql("SELECT * FROM flights", engine)
df_airport = pd.read_sql("SELECT * FROM airport", engine)
```

```
  Tables_in_airline
0           airport
1           flights
2        passengers
```

Showing the first 5 rows of each table

[4]:
```
df_passengers.head(5)
```

[4]:

	Passenger_id	First_name	Last_name	Gender	Age	Nationality
0	1	Edithe	Leggis	Female	62	Japan
1	2	Elwood	Catt	Male	62	Nicaragua
2	3	Darby	Felgate	Male	67	Russia
3	4	Dominica	Pyle	Female	71	China
4	5	Bay	Pencost	Male	21	China

[5]:
```
df_airport.head(5)
```

[5]:

	Airport_id	Airport_name	Airport_country_code	Country_name	\
0	1	Coldfoot Airport	US	United States	
1	2	Kugluktuk Airport	CA	Canada	
2	3	Grenoble-IsÃ¨re Airport	FR	France	
3	4	Ottawa / Gatineau Airport	CA	Canada	
4	5	Gillespie Field	US	United States	

	Airport_continent	Continents
0	NAM	North America
1	NAM	North America
2	EU	Europe
3	NAM	North America
4	NAM	North America

[6]:
```
df_flights.head(5)
```

[6]:
```
   Flight_id  Passenger_id  Airport_id Departure_date Arrival_airport  \
0          1             1           1     2022-06-28             CXF
1          2             2           2     2022-12-26             YCO
2          3             3           3     2022-01-18             GNB
3          4             4           4     2022-09-16             YND
4          5             5           5     2022-02-25             SEE

      Pilot_name Flight_status
0  Edithe Leggis       On Time
1    Elwood Catt       On Time
2  Darby Felgate       On Time
3  Dominica Pyle       Delayed
4    Bay Pencost       On Time
```

2 Transform

Cleaning and modifying the data for further analysis

[7]:
```
try:
    # Check for missing values
    logging.info("Transforming data...")

    # Missing values in each table
    passengers_missing = df_passengers.isnull().sum()
    flights_missing = df_flights.isnull().sum()
    airport_missing = df_airport.isnull().sum()

    logging.info(f"Missing values before cleaning: {passengers_missing},␣
 ↪{flights_missing}, {airport_missing}")

    print(passengers_missing)
    print(flights_missing)
    print(airport_missing)

    logging.info("Missing values checked successfully.")

except Exception as e:
    logging.error(f"Error in data transformation: {e}")
```

```
Passenger_id    0
First_name      0
Last_name       0
Gender          0
Age             0
Nationality     0
dtype: int64
Flight_id          0
Passenger_id       0
```

```
Airport_id         0
Departure_date     0
Arrival_airport    0
Pilot_name         0
Flight_status      0
dtype: int64
Airport_id              0
Airport_name            0
Airport_country_code    0
Country_name            0
Airport_continent       0
Continents              0
dtype: int64
```

The dataframe does not contain any null values

Data types

[8]:
```
# Checking data types
print("Passengers table:\n")
df_passengers.info()
print("\n")
print("Airport table:\n")
df_airport.info()
print("\n")
print("Flights table:\n")
df_flights.info()
```

```
Passengers table:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 98612 entries, 0 to 98611
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Passenger_id  98612 non-null  int64
 1   First_name    98612 non-null  object
 2   Last_name     98612 non-null  object
 3   Gender        98612 non-null  object
 4   Age           98612 non-null  int64
 5   Nationality   98612 non-null  object
dtypes: int64(2), object(4)
memory usage: 4.5+ MB


Airport table:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 98612 entries, 0 to 98611
```

```
Data columns (total 6 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Airport_id            98612 non-null  int64
 1   Airport_name          98612 non-null  object
 2   Airport_country_code  98612 non-null  object
 3   Country_name          98612 non-null  object
 4   Airport_continent     98612 non-null  object
 5   Continents            98612 non-null  object
dtypes: int64(1), object(5)
memory usage: 4.5+ MB


Flights table:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 98612 entries, 0 to 98611
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Flight_id        98612 non-null  int64
 1   Passenger_id     98612 non-null  int64
 2   Airport_id       98612 non-null  int64
 3   Departure_date   98612 non-null  object
 4   Arrival_airport  98612 non-null  object
 5   Pilot_name       98612 non-null  object
 6   Flight_status    98612 non-null  object
dtypes: int64(3), object(4)
memory usage: 5.3+ MB
```

[9]:
```python
# Fixing data types in Passengers
df_passengers['Gender'] = df_passengers['Gender'].astype('category')
df_passengers['Nationality'] = df_passengers['Nationality'].astype('category')

# Fixing data types in Airport
df_airport['Airport_name'] = df_airport['Airport_name'].astype('category')
df_airport['Airport_country_code'] = df_airport['Airport_country_code'].
 ↪astype('category')
df_airport['Country_name'] = df_airport['Country_name'].astype('category')
df_airport['Airport_continent'] = df_airport['Airport_continent'].
 ↪astype('category')
df_airport['Continents'] = df_airport['Continents'].astype('category')

# Fixing data types in Flights
df_flights['Departure_date'] = pd.to_datetime(df_flights['Departure_date'])
df_flights['Arrival_airport'] = df_flights['Arrival_airport'].astype('category')
df_flights['Flight_status'] = df_flights['Flight_status'].astype('category')
```

[10]:
```
# Checking data types
print("Passengers table:\n")
df_passengers.info()
print("\n")
print("Airport table:\n")
df_airport.info()
print("\n")
print("Flights table:\n")
df_flights.info()
```

```
Passengers table:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 98612 entries, 0 to 98611
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Passenger_id  98612 non-null  int64
 1   First_name    98612 non-null  object
 2   Last_name     98612 non-null  object
 3   Gender        98612 non-null  category
 4   Age           98612 non-null  int64
 5   Nationality   98612 non-null  category
dtypes: category(2), int64(2), object(2)
memory usage: 3.3+ MB


Airport table:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 98612 entries, 0 to 98611
Data columns (total 6 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Airport_id            98612 non-null  int64
 1   Airport_name          98612 non-null  category
 2   Airport_country_code  98612 non-null  category
 3   Country_name          98612 non-null  category
 4   Airport_continent     98612 non-null  category
 5   Continents            98612 non-null  category
dtypes: category(5), int64(1)
memory usage: 1.8 MB


Flights table:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 98612 entries, 0 to 98611
```

```
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Flight_id        98612 non-null  int64
 1   Passenger_id     98612 non-null  int64
 2   Airport_id       98612 non-null  int64
 3   Departure_date   98612 non-null  datetime64[ns]
 4   Arrival_airport  98612 non-null  category
 5   Pilot_name       98612 non-null  object
 6   Flight_status    98612 non-null  category
dtypes: category(2), datetime64[ns](1), int64(3), object(1)
memory usage: 4.4+ MB
```

We have efectively changed those 'object' to 'category' and 'date' types

Data Standardization

[11]:
```
# Standardizing categorical values to lowecase for consistency
df_passengers[['Gender', 'Nationality']] = df_passengers[['Gender',␣
↪'Nationality']].apply(lambda x: x.str.lower())

df_airport[['Airport_name', 'Airport_country_code', 'Country_name',␣
↪'Airport_continent', 'Continents']] = df_airport[['Airport_name',␣
↪'Airport_country_code', 'Country_name', 'Airport_continent', 'Continents']].
↪apply(lambda x: x.str.lower())

df_flights[['Arrival_airport', 'Flight_status']] =␣
↪df_flights[['Arrival_airport', 'Flight_status']].apply(lambda x: x.str.
↪lower())
```

Date Formatting

[12]:
```
# To ensure dates are consistent
df_flights['Departure_date_year'] = df_flights['Departure_date'].dt.year
df_flights['Departure_date_month'] = df_flights['Departure_date'].dt.month
df_flights['Departure_date_day'] = df_flights['Departure_date'].dt.day
df_flights.head(1)
```

[12]:

	Flight_id	Passenger_id	Airport_id	Departure_date	Arrival_airport	Pilot_name	Flight_status	Departure_date_year	Departure_date_month	Departure_date_day
0	1	1	1	2022-06-28	cxf	Edithe Leggis	on time	2022	6	28

Feature Engineering Extracting Day of week from Flights table

[13]:
```
df_flights['Departure_date_name'] = df_flights['Departure_date'].dt.day_name()
df_flights.head(1)
```

[13]:
```
   Flight_id  Passenger_id  Airport_id Departure_date Arrival_airport  \
0          1             1           1     2022-06-28             cxf

      Pilot_name Flight_status  Departure_date_year  Departure_date_month  \
0  Edithe Leggis       on time                 2022                     6

   Departure_date_day Departure_date_name
0                  28             Tuesday
```

Encoding Gender, and Age from Passengers table

[14]:
```
# Gender Encoding
df_passengers['Gender_encoded'] = df_passengers['Gender'].map({'male': 0,␣
↪'female': 1})

# Age Groups
df_passengers['Age_group'] = pd.cut(df_passengers['Age'], bins=[0, 12, 18, 60,␣
↪100], labels=['Child', 'Teen', 'Adult', 'Senior'])

df_passengers.head(1)
```

[14]:
```
   Passenger_id First_name Last_name  Gender  Age Nationality  Gender_encoded  \
0             1     Edithe    Leggis  female   62       japan               1

  Age_group
0    Senior
```

3 Load

Store the transformed data into MySQL database

[15]:
```
# Create the connection to the database
host = "airplane123.mysql.database.azure.com"
port = 3306
user = "student"
password = "Ads507password"
database = "airline-cleaned"

try:
    logging.info("Loading transformed data into MySQL...")

    engine2 = create_engine(
        f"mysql+pymysql://{user}:{password}@{host}:{port}/{database}"
    )
```

```
    logging.info("Data successfully loaded into MySQL.")

    print("Connected to airline database on Azure with SSL")

except Exception as e:
    print("Connection failed:", e)
```

```
Connected to airline database on Azure with SSL
```

[16]:
```
# Manually dropping foreign key constraints
with engine2.connect() as connection:
    connection.execute(text("SET FOREIGN_KEY_CHECKS = 0;"))
    connection.execute(text("DROP TABLE IF EXISTS flights;"))
    connection.execute(text("DROP TABLE IF EXISTS passengers;"))
    connection.execute(text("DROP TABLE IF EXISTS airports;"))
    connection.execute(text("SET FOREIGN_KEY_CHECKS = 1;"))
    connection.commit()
```

[17]:
```
# Storing the cleaned data into the database
df_passengers.to_sql('passengers', con=engine2, if_exists='replace',␣
 ↪index=False)
df_airport.to_sql('airports', con=engine2, if_exists='replace', index=False)
df_flights.to_sql('flights', con=engine2, if_exists='replace', index=False)
```

[17]:
```
98612
```

Verify Data Upload

[18]:
```
with engine2.connect() as connection:
    result = connection.execute(text("SELECT COUNT(*) FROM passengers"))
    print(f"Passengers Table Rows: {result.fetchone()[0]}")

    result = connection.execute(text("SELECT COUNT(*) FROM airports"))
    print(f"Airports Table Rows: {result.fetchone()[0]}")

    result = connection.execute(text("SELECT COUNT(*) FROM flights"))
    print(f"Flights Table Rows: {result.fetchone()[0]}")
```

```
Passengers Table Rows: 98612
Airports Table Rows: 98612
Flights Table Rows: 98612
```