SI564 FINAL PROJECT

junhuuu 12113312

I. Database Setup and Basic Information Preview

In my final project, I pretend to use a `airline_delay.csv` from Kaggle.

In [ ]:
```python
import sqlite3
import pandas as pd
```

In [ ]:
```python
# Load the CSV file
df = pd.read_csv('./airline_delay.csv')

# Preview the first few rows of each column
df.head()
```

Out[ ]:

	year	month	carrier	carrier_name	airport	airport_name	arr_flights	arr_del15	carrier_ct	weather_ct	...	security_ct	late_aircraft_ct	arr_cancelled	arr_diverted	arr_delay
0	2020	12	9E	Endeavor Air Inc.	ABE	Allentown/Bethlehem/Easton, PA: Lehigh Valley ...	44.0	3.0	1.63	0.0	...	0.0	1.25	0.0	1.0	89.0
1	2020	12	9E	Endeavor Air Inc.	ABY	Albany, GA: Southwest Georgia Regional	90.0	1.0	0.96	0.0	...	0.0	0.00	0.0	0.0	23.0
2	2020	12	9E	Endeavor Air Inc.	AEX	Alexandria, LA: Alexandria International	88.0	8.0	5.75	0.0	...	0.0	0.65	0.0	1.0	338.0
3	2020	12	9E	Endeavor Air Inc.	AGS	Augusta, GA: Augusta Regional at Bush Field	184.0	9.0	4.17	0.0	...	0.0	3.00	0.0	0.0	508.0
4	2020	12	9E	Endeavor Air Inc.	ALB	Albany, NY: Albany International	76.0	11.0	4.78	0.0	...	0.0	1.00	1.0	0.0	692.0

5 rows × 21 columns

In [ ]:
```python
# Basic information of the dataframe
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3351 entries, 0 to 3350
Data columns (total 21 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   year                3351 non-null   int64
 1   month               3351 non-null   int64
 2   carrier             3351 non-null   object
 3   carrier_name        3351 non-null   object
 4   airport             3351 non-null   object
 5   airport_name        3351 non-null   object
 6   arr_flights         3343 non-null   float64
 7   arr_del15           3343 non-null   float64
 8   carrier_ct          3343 non-null   float64
 9   weather_ct          3343 non-null   float64
 10  nas_ct              3343 non-null   float64
 11  security_ct         3343 non-null   float64
 12  late_aircraft_ct    3343 non-null   float64
 13  arr_cancelled       3343 non-null   float64
 14  arr_diverted        3343 non-null   float64
 15  arr_delay           3343 non-null   float64
 16  carrier_delay       3343 non-null   float64
 17  weather_delay       3343 non-null   float64
 18  nas_delay           3343 non-null   float64
 19  security_delay      3343 non-null   float64
 20  late_aircraft_delay  3343 non-null   float64
dtypes: float64(15), int64(2), object(4)
memory usage: 549.9+ KB
```

Therefore, we know that this dataset has 21 columns and 3351 rows. ANd following are the meaning of each column.

`year` Year data collected

`month` Numeric representation of the month

`carrier` Carrier.

`carrier_name` Carrier Name.

`airport` Airport code.

`airport_name` Name of airport.

`arr_flights` Number of flights arriving at airport

`arr_del15` Number of flights more than 15 minutes late

`carrier_ct` Number of flights delayed due to air carrier. (e.g. no crew)

`weather_ct` Number of flights due to weather.

`nas_ct` Number of flights delayed due to National Aviation System (e.g. heavy air traffic).

`security_ct` Number of flights canceled due to a security breach.

`late_aircraft_ct` Number of flights delayed as a result of another flight on the same aircraft delayed

`arr_cancelled` Number of cancelled flights

`arr_diverted` Number of flights that were diverted

`arr_delay` Total time (minutes) of delayed flight.

`carrier_delay` Total time (minutes) of delay due to air carrier

`weather_delay` Total time (minutes) of delay due to inclement weather.

`nas_delay` Total time (minutes) of delay due to National Aviation System.

`security_delay` Total time (minutes) of delay as a result of a security issue .

`late_aircraft_delay` Total time (minutes) of delay flights as a result of a previous flight on the same airplane being late.

i.Set up SQLite connection and Normalize the dataset to Create the Database

To reduces redundancy and ensures data integrity,given the structure of your dataset here's my normalization schema:

`Carriers Table` : Contains information about the airline carriers.
Columns: Carrier ID (Primary Key), Carrier Name

`Airports Table` : Contains information about the airports.
Columns: Airport Code (Primary Key), Airport Name

`Flights Table` : Contains detailed information about each flight.
Columns: Flight ID (Primary Key), Year, Month, Carrier ID (Foreign Key referencing Carriers Table), Airport Code (Foreign Key referencing Airports Table), Arrived Flights, Arrival Delays over 15 minutes, Arrival Cancellations, Arrival Diversions, Arrival Delays

`Delays Table` : Contains detailed information about the delays.
Columns: Delay ID (Primary Key), Flight ID (Foreign Key referencing Flights Table), Carrier Delay Count, Weather Delay Count, National Air System Delay Count, Security Delay Count, Late Aircraft Delay Count, Carrier Delay (Minutes), Weather Delay (Minutes), National Air System Delay (Minutes), Security Delay (Minutes), Late Aircraft Delay (Minutes)

Table Documentation

Carriers

Field Name	Data Type	Values	Note
carrier_id	TEXT	id of carrier	PRIMARY KEY
carrier_name	TEXT	carrier name	

Airports

Field Name	Data Type	Values	Note
airport_code	TEXT	airport code	PRIMARY KEY
airport_name	TEXT	name of airport	

Flights

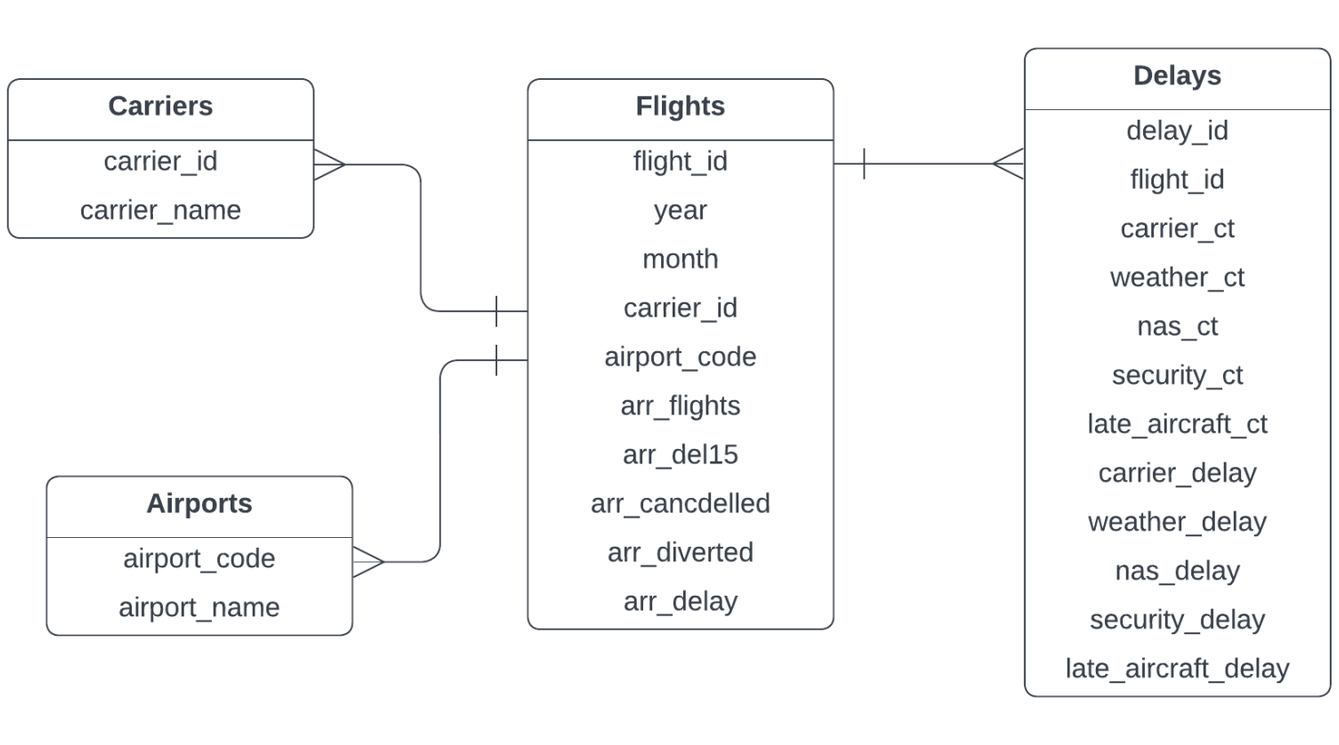
Field Name	Data Type	Values	Note
flight_id	INTEGER	id of flight	PRIMARY KEY
year	INTEGER	year data collected	
month	INTEGER	numeric representation of the month	
carrier_id	TEXT	id of carrier	
airport_code	TEXT	airport code	
arr_flights	REAL	number of flights arriving at airport	
arr_del15	REAL	number of flights more than 15 minutes late	
arr_cancelled	REAL	number of cancelled flights	
arr_diverted	REAL	number of flights that were diverted	
arr_delay	REAL	total time (minutes) of delayed flight.	

Delays

Field Name	Data Type	Values	Note
delay_id	INTEGER	id of delay	PRIMARY KEY
flight_id	INTEGER	id of flight	
carrier_ct	REAL	number of flights delayed due to air carrier. (e.g. no crew)	
weather_ct	REAL	number of flights due to weather.	
nas_ct	REAL	number of flights delayed due to National Aviation System (e.g. heavy air traffic).	
security_ct	REAL	number of flights canceled due to a security breach.	
late_aircraft_ct	REAL	number of flights delayed as a result of another flight on the same aircraft delayed	
carrier_delay	REAL	total time (minutes) of delay due to air carrier	
weather_delay	REAL	total time (minutes) of delay due to inclement weather.	
nas_delay	REAL	total time (minutes) of delay due to National Aviation System.	
security_delay	REAL	total time (minutes) of delay as a result of a security issue .	
late_aircraft_delay	REAL	total time (minutes) of delay flights as a result of a previous flight on the same airplane being late.	

ERD

airline_delay



In [ ]:
```python
import sqlite3

# Connect to a SQLite database
conn = sqlite3.connect('./final.db')
cursor = conn.cursor()

# Create the Carriers table
cursor.execute('''
    CREATE TABLE Carriers (
        carrier_id TEXT PRIMARY KEY,
        carrier_name TEXT
    )
''')

# Create the Airports table
cursor.execute('''
    CREATE TABLE Airports (
        airport_code TEXT PRIMARY KEY,
        airport_name TEXT
    )
''')

# Create the Flights table
cursor.execute('''
    CREATE TABLE Flights (
        flight_id INTEGER PRIMARY KEY AUTOINCREMENT,
        year INTEGER,
        month INTEGER,
        carrier_id TEXT,
        airport_code TEXT,
        arr_flights REAL,
        arr_del15 REAL,
        arr_cancelled REAL,
        arr_diverted REAL,
        arr_delay REAL,
        FOREIGN KEY (carrier_id) REFERENCES Carriers(carrier_id),
        FOREIGN KEY (airport_code) REFERENCES Airports(airport_code)
    )
''')

# Create the Delays table
cursor.execute('''
    CREATE TABLE Delays (
        delay_id INTEGER PRIMARY KEY AUTOINCREMENT,
        flight_id INTEGER,
        carrier_ct REAL,
        weather_ct REAL,
        nas_ct REAL,
        security_ct REAL,
        late_aircraft_ct REAL,
        carrier_delay REAL,
        weather_delay REAL,
        nas_delay REAL,
        security_delay REAL,
        late_aircraft_delay REAL,
        FOREIGN KEY (flight_id) REFERENCES Flights(flight_id)
    )
''')

# Insert data into Carriers and Airports tables
unique_carriers = df[['carrier', 'carrier_name']].drop_duplicates()
unique_airports = df[['airport', 'airport_name']].drop_duplicates()

for _, row in unique_carriers.iterrows():
    cursor.execute("INSERT INTO Carriers (carrier_id, carrier_name) VALUES (?, ?)", (row['carrier'], row['carrier_name']))

for _, row in unique_airports.iterrows():
    cursor.execute("INSERT INTO Airports (airport_code, airport_name) VALUES (?, ?)", (row['airport'], row['airport_name']))

# Insert data into Flights and Delays tables
for _, row in df.iterrows():
    # Insert into Flights table and get the inserted flight_id
    cursor.execute("""
        INSERT INTO Flights (year, month, carrier_id, airport_code, arr_flights, arr_del15, arr_cancelled, arr_diverted, arr_delay)
        VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)
    """, (row['year'], row['month'], row['carrier'], row['airport'], row['arr_flights'], row['arr_del15'], row['arr_cancelled'], row['arr_diverted'], row['arr_delay'
    flight_id = cursor.lastrowid

    # Insert into Delays table
    cursor.execute("""
        INSERT INTO Delays (flight_id, carrier_ct, weather_ct, nas_ct, security_ct, late_aircraft_ct, carrier_delay, weather_delay, nas_delay, security_delay, late_a
        VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
    """, (flight_id, row['carrier_ct'], row['weather_ct'], row['nas_ct'], row['security_ct'], row['late_aircraft_ct'], row['carrier_delay'], row['weather_delay'], ro

# Commit the changes
conn.commit()
```

ii.Export the sql file

In [ ]:
```python
# File path for the SQL dump
sql_dump_file_path = './airline_delay.sql'

# Open a file to write the SQL dump
with open(sql_dump_file_path, 'w') as file:
    for line in conn.iterdump():
        file.write(f'{line}\n')
```

iii.preview all the tables

In [ ]:
```python
# Define the function to preview tables with column names
def preview_table(cursor, table_name):
    cursor.execute(f"PRAGMA table_info({table_name});")
    columns = [col[1] for col in cursor.fetchall()]  # Extract column names

    cursor.execute(f"SELECT * FROM {table_name} LIMIT 5;")
    rows = cursor.fetchall()

    return columns, rows

# Preview the tables with column names
tables = ['Carriers', 'Airports', 'Flights', 'Delays']
table_previews = {}
for table in tables:
    table_previews[table] = preview_table(cursor, table)

table_previews
```

Out[ ]:
```
{'Carriers': (['carrier_id', 'carrier_name'],
  [('9E', 'Endeavor Air Inc.'),
   ('AA', 'American Airlines Inc.'),
   ('AS', 'Alaska Airlines Inc.'),
   ('B6', 'JetBlue Airways'),
   ('DL', 'Delta Air Lines Inc.')]),
 'Airports': (['airport_code', 'airport_name'],
  [('ABE', 'Allentown/Bethlehem/Easton, PA: Lehigh Valley International'),
   ('ABY', 'Albany, GA: Southwest Georgia Regional'),
   ('AEX', 'Alexandria, LA: Alexandria International'),
   ('AGS', 'Augusta, GA: Augusta Regional at Bush Field'),
   ('ALB', 'Albany, NY: Albany International')]),
 'Flights': (['flight_id',
   'year',
   'month',
   'carrier_id',
   'airport_code',
   'arr_flights',
   'arr_del15',
   'arr_cancelled',
   'arr_diverted',
   'arr_delay'],
  [(1, 2020, 12, '9E', 'ABE', 44.0, 3.0, 0.0, 1.0, 89.0),
   (2, 2020, 12, '9E', 'ABY', 90.0, 1.0, 0.0, 0.0, 23.0),
   (3, 2020, 12, '9E', 'AEX', 88.0, 8.0, 0.0, 1.0, 338.0),
   (4, 2020, 12, '9E', 'AGS', 184.0, 9.0, 0.0, 0.0, 508.0),
   (5, 2020, 12, '9E', 'ALB', 76.0, 11.0, 1.0, 0.0, 692.0)]),
 'Delays': (['delay_id',
   'flight_id',
   'carrier_ct',
   'weather_ct',
   'nas_ct',
   'security_ct',
   'late_aircraft_ct',
   'carrier_delay',
   'weather_delay',
   'nas_delay',
   'security_delay',
   'late_aircraft_delay'],
  [(1, 1, 1.63, 0.0, 0.12, 0.0, 1.25, 56.0, 0.0, 3.0, 0.0, 30.0),
   (2, 2, 0.96, 0.0, 0.04, 0.0, 0.0, 22.0, 0.0, 1.0, 0.0, 0.0),
   (3, 3, 5.75, 0.0, 1.6, 0.0, 0.65, 265.0, 0.0, 45.0, 0.0, 28.0),
   (4, 4, 4.17, 0.0, 1.83, 0.0, 3.0, 192.0, 0.0, 92.0, 0.0, 224.0),
   (5, 5, 4.78, 0.0, 5.22, 0.0, 1.0, 398.0, 0.0, 178.0, 0.0, 116.0)])}
```

II. Solve Problems

i. Determine which airline has the highest average delay.

In [ ]:
```python
query = '''
    SELECT c.carrier_name,AVG(f.arr_delay) AS avg_delay
    FROM Flights f
    JOIN Carriers c ON f.carrier_id = c.carrier_id
    GROUP BY c.carrier_name
    ORDER BY avg_delay DESC
    LIMIT 1;
    '''
cursor.execute(query)
cursor.fetchone()
```

Out[ ]: `('Southwest Airlines Co.', 7713.165745856353)`

ii. Top 5 Carriers with the Most Delays.

In [ ]:
```python
query = '''
    SELECT c.carrier_name, COUNT(*) AS delay_count
    FROM Flights f
    JOIN Carriers c ON f.carrier_id = c.carrier_id
    WHERE f.arr_del15 > 0
    GROUP BY f.carrier_id
    ORDER BY delay_count DESC
    LIMIT 5;
'''
cursor.execute(query)
cursor.fetchall()
```

Out[ ]:
```
[('SkyWest Airlines Inc.', 459),
 ('Envoy Air', 274),
 ('Delta Air Lines Inc.', 241),
 ('Allegiant Air', 228),
 ('Endeavor Air Inc.', 217)]
```

iii. Most Common Cause of Delays

In [ ]:
```python
query = '''
    SELECT 'Carrier' AS reason, SUM(carrier_ct) FROM Delays
    UNION ALL
    SELECT 'Weather', SUM(weather_ct) FROM Delays
    UNION ALL
    SELECT 'NAS', SUM(nas_ct) FROM Delays
    UNION ALL
    SELECT 'Security', SUM(security_ct) FROM Delays
    UNION ALL
    SELECT 'Late Aircraft', SUM(late_aircraft_ct) FROM Delays
    ORDER BY 2 DESC
    LIMIT 1;
'''
cursor.execute(query)
cursor.fetchone()
```

Out[ ]: `('Late Aircraft', 57386.16999999997)`

iv. Airports with the Highest Number of Cancellations

In [ ]:
```python
query = '''
    SELECT a.airport_name, SUM(f.arr_cancelled) AS total_cancellations
    FROM Flights f
    JOIN Airports a ON f.airport_code = a.airport_code
    GROUP BY f.airport_code
    ORDER BY total_cancellations DESC
    LIMIT 5;
'''
cursor.execute(query)
cursor.fetchall()
```

Out[ ]:
```
[('Boston, MA: Logan International', 467.0),
 ("Chicago, IL: Chicago O'Hare International", 457.0),
 ('San Francisco, CA: San Francisco International', 433.0),
 ('Dallas/Fort Worth, TX: Dallas/Fort Worth International', 428.0),
 ('Denver, CO: Denver International', 396.0)]
```

v. Total Delays per Year

In [ ]:
```python
query = '''
    SELECT year, SUM(arr_del15) AS total_delays
    FROM Flights
    GROUP BY year;
'''
cursor.execute(query)
cursor.fetchall()
```

Out[ ]: `[(2019, 126945.0), (2020, 43532.0)]`

vi. Ratio of Delayed to Non-Delayed Flights for Each Carrier

In [ ]:
```python
query = '''
    SELECT
        c.carrier_name,
        SUM(CASE WHEN f.arr_del15 > 0 THEN 1 ELSE 0 END) / COUNT(f.flight_id) AS delay_ratio
    FROM
        Flights f
    JOIN
        Carriers c ON f.carrier_id = c.carrier_id
    GROUP BY
        c.carrier_name
    ORDER BY
        delay_ratio DESC;
    '''
cursor.execute(query)
cursor.fetchall()
```

Out[ ]:
```
[('Spirit Air Lines', 1),
 ('Southwest Airlines Co.', 1),
 ('JetBlue Airways', 1),
 ('Hawaiian Airlines Inc.', 1),
 ('United Air Lines Inc.', 0),
 ('SkyWest Airlines Inc.', 0),
 ('Republic Airline', 0),
 ('PSA Airlines Inc.', 0),
 ('Mesa Airlines Inc.', 0),
 ('Frontier Airlines Inc.', 0),
 ('ExpressJet Airlines LLC', 0),
 ('Envoy Air', 0),
 ('Endeavor Air Inc.', 0),
 ('Delta Air Lines Inc.', 0),
 ('American Airlines Inc.', 0),
 ('Allegiant Air', 0),
 ('Alaska Airlines Inc.', 0)]
```

vii. Year-wise Breakdown of Delays by Reason

In [ ]:
```python
query = '''
    SELECT
        f.year,
        SUM(d.carrier_ct) AS carrier_delays,
        SUM(d.weather_ct) AS weather_delays,
        SUM(d.nas_ct) AS nas_delays,
        SUM(d.security_ct) AS security_delays,
        SUM(d.late_aircraft_ct) AS late_aircraft_delays
    FROM Delays d
    JOIN Flights f ON d.flight_id = f.flight_id
    GROUP BY f.year;
'''
cursor.execute(query)
cursor.fetchall()
```

Out[ ]:
```
[(2019,
  36488.05000000001,
  3161.159999999996,
  38651.82999999999,
  277.28999999999996,
  48366.720000000016),
 (2020,
  17218.369999999995,
  1663.2700000000007,
  15449.220000000005,
  181.76999999999995,
  9019.449999999995)]
```

viii. Least Delay-Prone Airport

In [ ]:
```python
query = '''
    SELECT
        a.airport_name,
        AVG(f.arr_delay) AS avg_delay
    FROM
        Flights f
    JOIN
        Airports a ON f.airport_code = a.airport_code
    GROUP BY
        a.airport_name
    ORDER BY
        avg_delay ASC
    LIMIT 1;
    '''
cursor.execute(query)
cursor.fetchone()
```

Out[ ]: `('Greenville, NC: Pitt Greenville', 0.0)`