Project Documentation: SQL and PySpark Data Analysis on Kaggle Airline Dataset

Overview

This project aims to assess and demonstrate skills in relational database operations using SQL and data analysis using PySpark. The dataset used is an airline-related dataset from Kaggle, consisting of eight tables. The tasks include setting up the database, performing basic exploratory data analysis (EDA), writing SQL queries, performing similar operations in PySpark, and formulating and solving a set of 15 questions.

Dataset Schema

The dataset consists of the following eight tables:

1. **aircrafts_data**: Information about aircrafts.
   - `aircraft_code` (character(3)) - Primary Key
   - `model` (jsonb)
   - `range` (integer)
2. **airports_data**: Information about airports.
   - `airport_code` (character(3)) - Primary Key
   - `airport_name` (jsonb)
   - `city` (jsonb)
   - `coordinates` (point)
   - `timezone` (text)
3. **boarding_passes**: Information about boarding passes.
   - `ticket_no` (character(13)) - Primary Key, Foreign Key
   - `flight_id` (integer) - Primary Key, Foreign Key
   - `boarding_no` (integer)
   - `seat_no` (character varying(4))
4. **bookings**: Information about bookings.
   - `book_ref` (character(6)) - Primary Key
   - `book_date` (timestamp with time zone)
   - `total_amount` (numeric(10,2))
5. **flights**: Information about flights.
   - `flight_id` (integer) - Primary Key
   - `flight_no` (character(6))
   - `scheduled_departure` (timestamp with time zone)
   - `scheduled_arrival` (timestamp with time zone)
   - `departure_airport` (character(3)) - Foreign Key
   - `arrival_airport` (character(3)) - Foreign Key
   - `status` (character varying(20))
   - `aircraft_code` (character(3)) - Foreign Key
   - `actual_departure` (timestamp with time zone)
   - `actual_arrival` (timestamp with time zone)
6. **seats**: Information about seats in aircrafts.
   - `aircraft_code` (character(3)) - Primary Key, Foreign Key
   - `seat_no` (character varying(4)) - Primary Key

  - `fare_conditions` (character varying(10))

7. **ticket_flights**: Information about ticket flights.
   - `ticket_no` (character(13)) - Primary Key, Foreign Key
   - `flight_id` (integer) - Primary Key, Foreign Key
   - `fare_conditions` (character varying(10))
   - `amount` (numeric(10,2))
8. **tickets**: Information about tickets.
   - `ticket_no` (character(13)) - Primary Key
   - `book_ref` (character(6)) - Foreign Key
   - `passenger_id` (character varying(20))

Project Tasks

1. **Database Setup**
   - Load Kaggle database to SQLite .
   - SQLite database setup in jupyter-notebook

```
import sqlite3
import pandas as pd
# Reconnect to the SQLite database
conn = sqlite3.connect('Airlines_data.sqlite') #It contains 8 tables
conn.close()

%load_ext sql
#load sql module to ipython

%sql sqlite:///Airlines_data.sqlite
```

2. **Exploratory Data Analysis (EDA)**

   General Pandas Operations:

   - Loaded the dataset into pandas DataFrames to explore the data.
   - Displayed basic statistics and checked for null values.

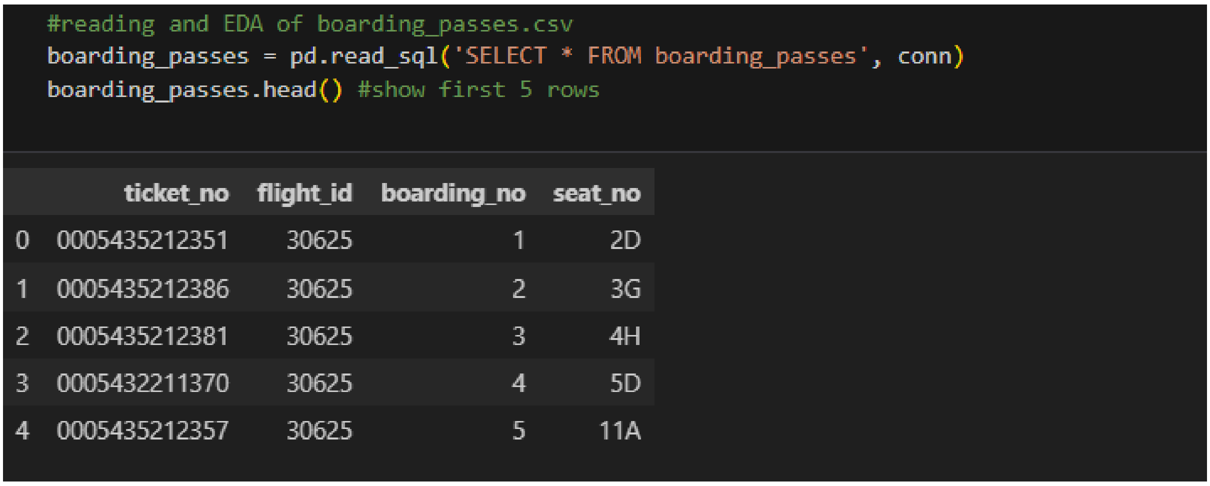
   Schema Familiarization:

   - Reviewed the schema definitions and relationships between tables.

   Primary and Foreign Key Relationships:

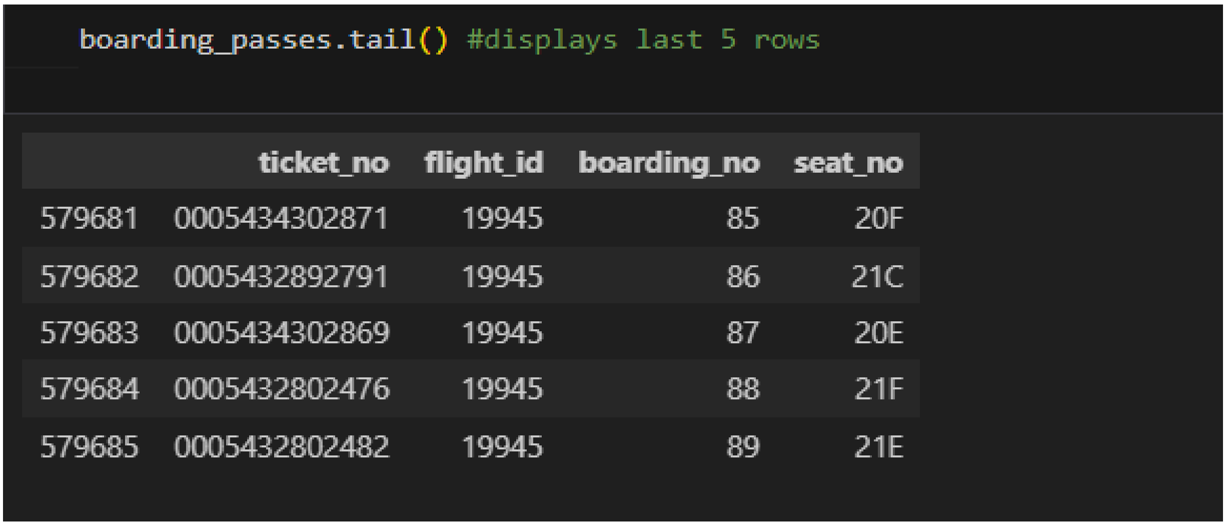
   - Identified primary keys for each table.
   - Identified foreign key relationships using `PRAGMA foreign_key_list`.

Example **: EDA of boarding_passes.csv**



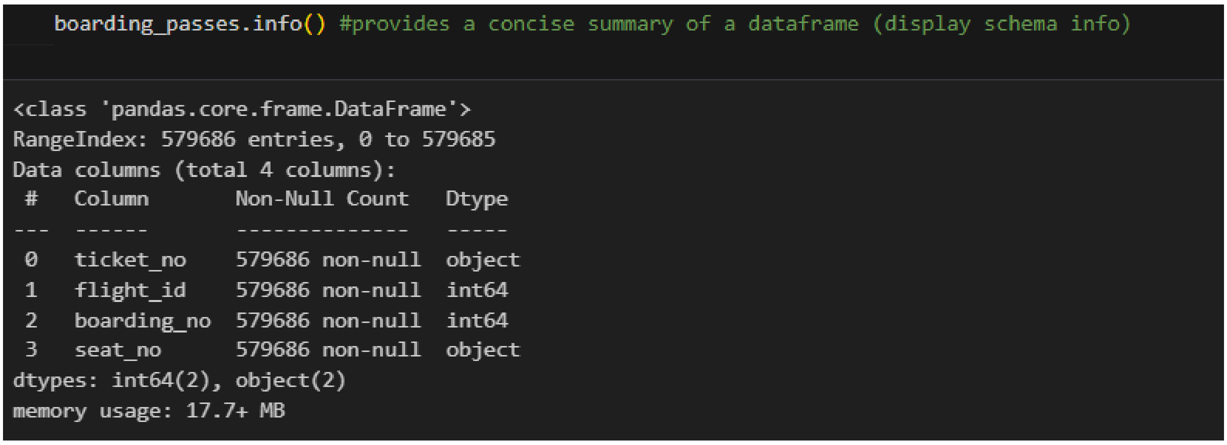
```python
#reading and EDA of boarding_passes.csv
boarding_passes = pd.read_sql('SELECT * FROM boarding_passes', conn)
boarding_passes.head() #show first 5 rows
```

	ticket_no	flight_id	boarding_no	seat_no
0	0005435212351	30625	1	2D
1	0005435212386	30625	2	3G
2	0005435212381	30625	3	4H
3	0005432211370	30625	4	5D
4	0005435212357	30625	5	11A



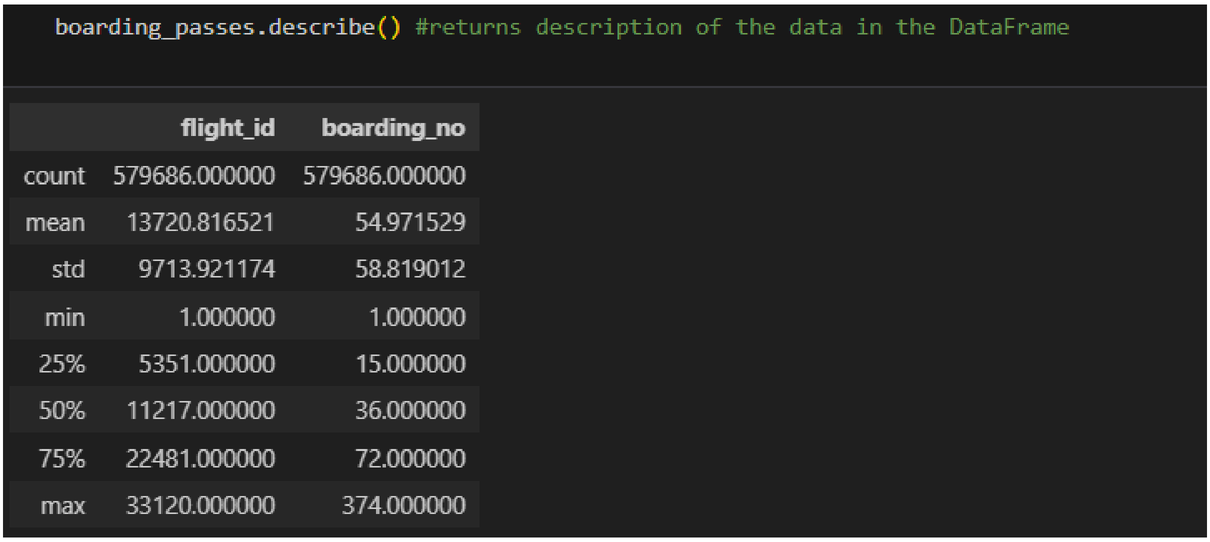
```python
boarding_passes.tail() #displays last 5 rows
```

	ticket_no	flight_id	boarding_no	seat_no
579681	0005434302871	19945	85	20F
579682	0005432892791	19945	86	21C
579683	0005434302869	19945	87	20E
579684	0005432802476	19945	88	21F
579685	0005432802482	19945	89	21E



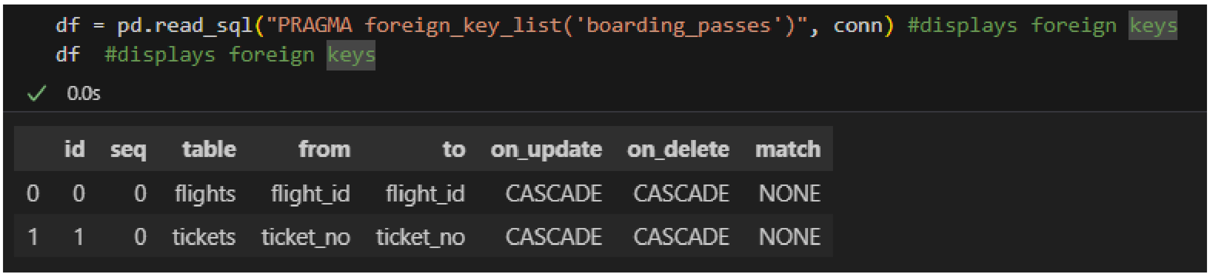
```python
boarding_passes.info() #provides a concise summary of a dataframe (display schema info)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 579686 entries, 0 to 579685
Data columns (total 4 columns):
 #   Column       Non-Null Count   Dtype 
---  ------       --------------   ----- 
 0   ticket_no    579686 non-null  object
 1   flight_id    579686 non-null  int64 
 2   boarding_no  579686 non-null  int64 
 3   seat_no      579686 non-null  object
dtypes: int64(2), object(2)
memory usage: 17.7+ MB
```



```
boarding_passes.describe() #returns description of the data in the DataFrame
```

```
            flight_id    boarding_no
count   579686.000000  579686.000000
mean     13720.816521      54.971529
std       9713.921174      58.819012
min          1.000000       1.000000
25%       5351.000000      15.000000
50%      11217.000000      36.000000
75%      22481.000000      72.000000
max      33120.000000     374.000000
```



```
df = pd.read_sql("PRAGMA foreign_key_list('boarding_passes')", conn) #displays foreign keys
df  #displays foreign keys
```

```
✓ 0.0s
```

```
   id  seq    table       from         to  on_update  on_delete  match
0   0    0  flights  flight_id  flight_id    CASCADE    CASCADE   NONE
1   1    0  tickets  ticket_no  ticket_no    CASCADE    CASCADE   NONE
```

NOTE – Check ‘EDA & SQL.ipynb’ file to see other table’s EDA

3. SQL Operations

- **Distinct Values:**

```
Table-1 [Key column: aircraft_code | Table: aircrafts_data]
SELECT DISTINCT aircraft_code FROM aircrafts_data;

Table-2 [Key column: airport_code | Table: airports_data]
SELECT DISTINCT airport_code FROM airports_data;

Table-3 [Key column: ticket_no | Table: boarding_passes]
SELECT DISTINCT ticket_no FROM boarding_passes;

Table-4 [Key column: book_refs | Table: bookings]
SELECT DISTINCT book_ref FROM bookings;



Table-5 [Key column: flight_id | Table: flights]
SELECT DISTINCT flight_id FROM flights;
```

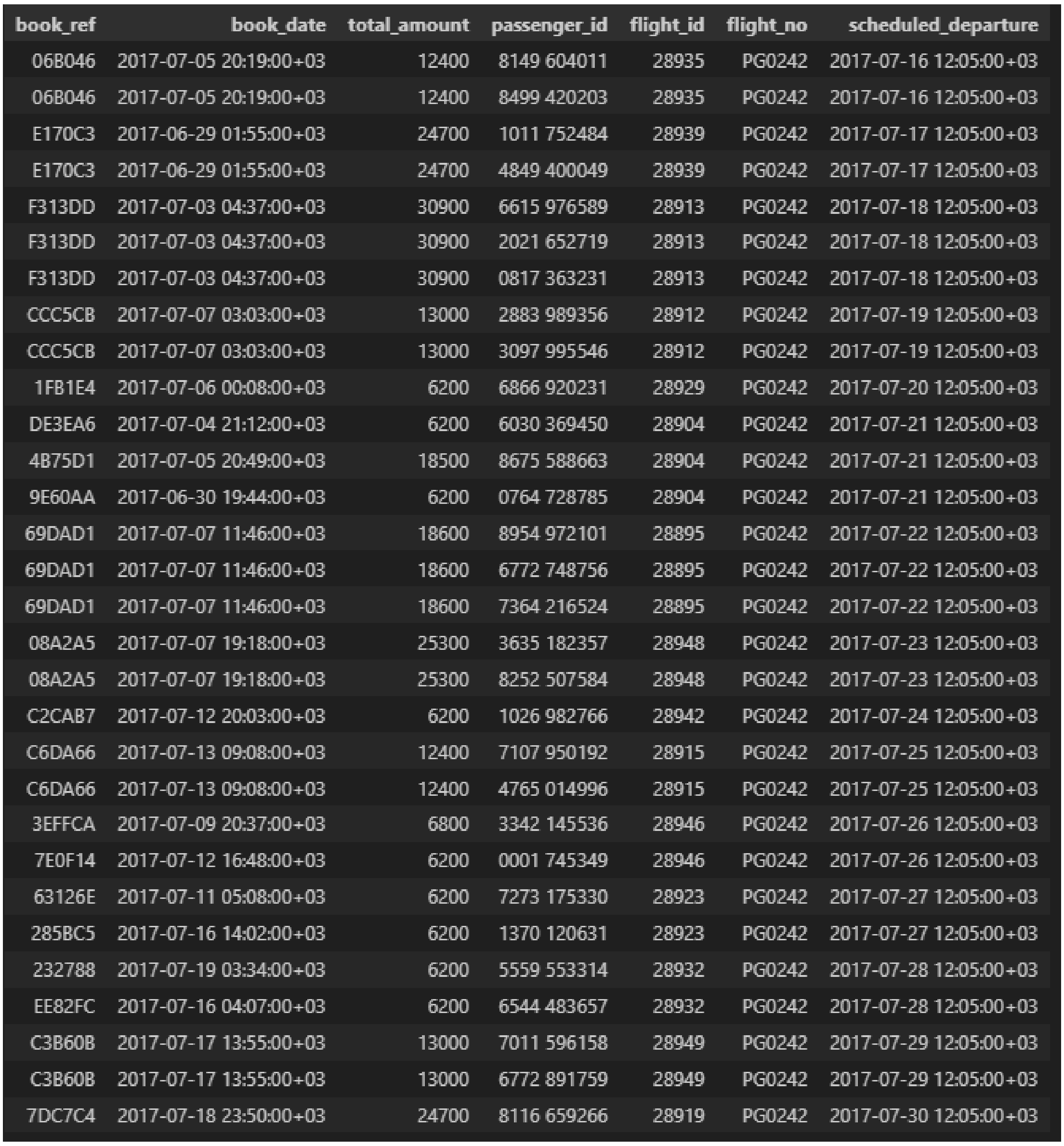
```
Table-6 [Key column: seat_no | Table: seats]
SELECT DISTINCT seat_no FROM seats;

Table-7 [Key column: ticket_no | Table: ticket_flights]
SELECT DISTINCT ticket_no FROM ticket_flights;

Table-8 [Key column: ticket_no | Table: tickets]
SELECT DISTINCT ticket_no FROM tickets;
```

Consolidated View(JOIN):

```
SELECT b.book_ref, b.book_date, b.total_amount, t.passenger_id,
tf.flight_id, f.flight_no, f.scheduled_departure
FROM ( bookings b
JOIN tickets t ON b.book_ref = t.book_ref
JOIN ticket_flights tf ON t.ticket_no = tf.ticket_no
JOIN flights f ON tf.flight_id = f.flight_id )
```

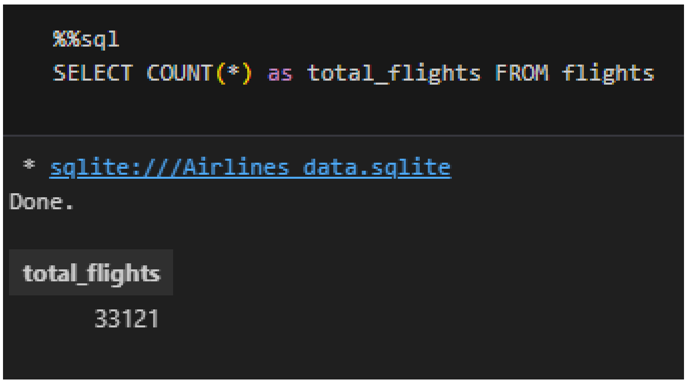
Explanation

- The book_ref column represents unique booking references.
- The book_date column shows when each booking was made.
- The total_amount column contains the total cost of each booking.
- The passenger_id column provides the ID of the passenger who made the booking.
- The flight_id column shows the unique identifier of the flight associated with the booking.
- The flight_no column gives the flight number.• The scheduled_departure column provides the scheduled departure time of the flight.



book_ref	book_date	total_amount	passenger_id	flight_id	flight_no	scheduled_departure
06B046	2017-07-05 20:19:00+03	12400	8149 604011	28935	PG0242	2017-07-16 12:05:00+03
06B046	2017-07-05 20:19:00+03	12400	8499 420203	28935	PG0242	2017-07-16 12:05:00+03
E170C3	2017-06-29 01:55:00+03	24700	1011 752484	28939	PG0242	2017-07-17 12:05:00+03
E170C3	2017-06-29 01:55:00+03	24700	4849 400049	28939	PG0242	2017-07-17 12:05:00+03
F313DD	2017-07-03 04:37:00+03	30900	6615 976589	28913	PG0242	2017-07-18 12:05:00+03
F313DD	2017-07-03 04:37:00+03	30900	2021 652719	28913	PG0242	2017-07-18 12:05:00+03
F313DD	2017-07-03 04:37:00+03	30900	0817 363231	28913	PG0242	2017-07-18 12:05:00+03
CCC5CB	2017-07-07 03:03:00+03	13000	2883 989356	28912	PG0242	2017-07-19 12:05:00+03
CCC5CB	2017-07-07 03:03:00+03	13000	3097 995546	28912	PG0242	2017-07-19 12:05:00+03
1FB1E4	2017-07-06 00:08:00+03	6200	6866 920231	28929	PG0242	2017-07-20 12:05:00+03
DE3EA6	2017-07-04 21:12:00+03	6200	6030 369450	28904	PG0242	2017-07-21 12:05:00+03
4B75D1	2017-07-05 20:49:00+03	18500	8675 588663	28904	PG0242	2017-07-21 12:05:00+03
9E60AA	2017-06-30 19:44:00+03	6200	0764 728785	28904	PG0242	2017-07-21 12:05:00+03
69DAD1	2017-07-07 11:46:00+03	18600	8954 972101	28895	PG0242	2017-07-22 12:05:00+03
69DAD1	2017-07-07 11:46:00+03	18600	6772 748756	28895	PG0242	2017-07-22 12:05:00+03
69DAD1	2017-07-07 11:46:00+03	18600	7364 216524	28895	PG0242	2017-07-22 12:05:00+03
08A2A5	2017-07-07 19:18:00+03	25300	3635 182357	28948	PG0242	2017-07-23 12:05:00+03
08A2A5	2017-07-07 19:18:00+03	25300	8252 507584	28948	PG0242	2017-07-23 12:05:00+03
C2CAB7	2017-07-12 20:03:00+03	6200	1026 982766	28942	PG0242	2017-07-24 12:05:00+03
C6DA66	2017-07-13 09:08:00+03	12400	7107 950192	28915	PG0242	2017-07-25 12:05:00+03
C6DA66	2017-07-13 09:08:00+03	12400	4765 014996	28915	PG0242	2017-07-25 12:05:00+03
3EFFCA	2017-07-09 20:37:00+03	6800	3342 145536	28946	PG0242	2017-07-26 12:05:00+03
7E0F14	2017-07-12 16:48:00+03	6200	0001 745349	28946	PG0242	2017-07-26 12:05:00+03
63126E	2017-07-11 05:08:00+03	6200	7273 175330	28923	PG0242	2017-07-27 12:05:00+03
285BC5	2017-07-16 14:02:00+03	6200	1370 120631	28923	PG0242	2017-07-27 12:05:00+03
232788	2017-07-19 03:34:00+03	6200	5559 553314	28932	PG0242	2017-07-28 12:05:00+03
EE82FC	2017-07-16 04:07:00+03	6200	6544 483657	28932	PG0242	2017-07-28 12:05:00+03
C3B60B	2017-07-17 13:55:00+03	13000	7011 596158	28949	PG0242	2017-07-29 12:05:00+03
C3B60B	2017-07-17 13:55:00+03	13000	6772 891759	28949	PG0242	2017-07-29 12:05:00+03
7DC7C4	2017-07-18 23:50:00+03	24700	8116 659266	28919	PG0242	2017-07-30 12:05:00+03

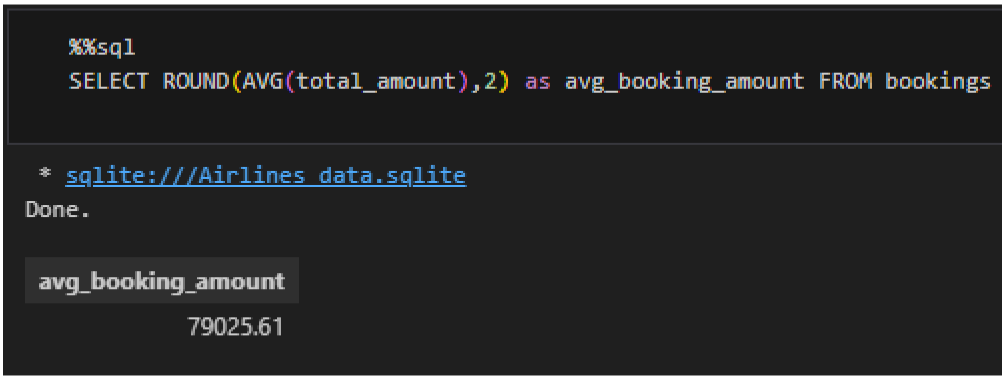
Summary Statistics:



```
%%sql
SELECT COUNT(*) as total_flights FROM flights
```

```
 * sqlite:///Airlines_data.sqlite
Done.
```

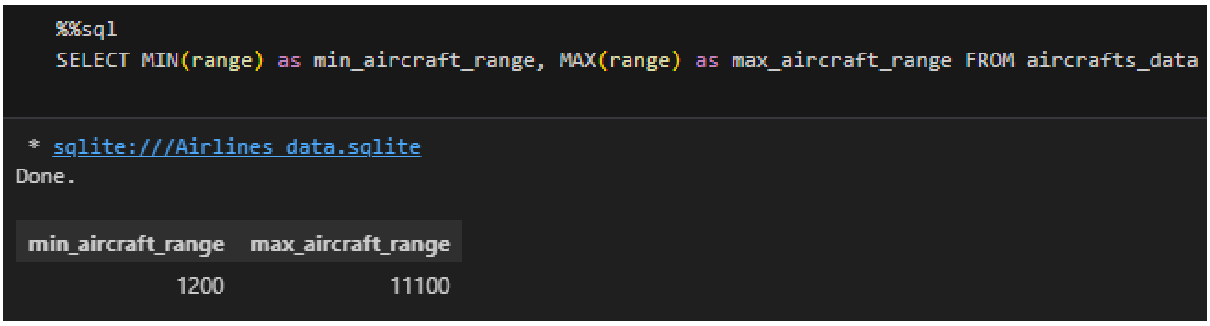
total_flights
33121



```
%%sql
SELECT ROUND(AVG(total_amount),2) as avg_booking_amount FROM bookings
```

```
 * sqlite:///Airlines_data.sqlite
Done.
```

avg_booking_amount
79025.61



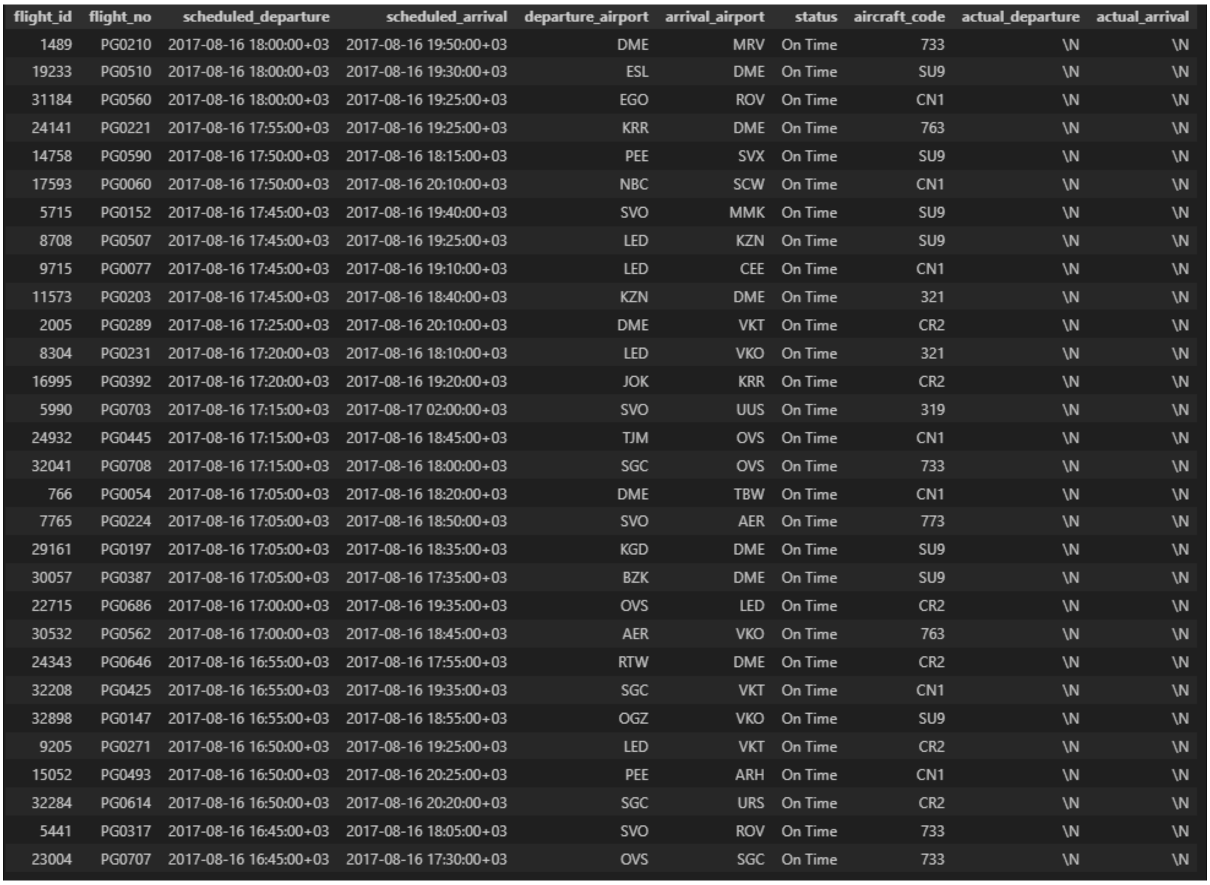
```
%%sql
SELECT MIN(range) as min_aircraft_range, MAX(range) as max_aircraft_range FROM aircrafts_data
```

```
 * sqlite:///Airlines_data.sqlite
Done.
```

min_aircraft_range	max_aircraft_range
1200	11100

- **Filter and Sort Data:**

```
%%sql

SELECT * FROM flights
WHERE status = 'On Time'
ORDER BY scheduled_departure DESC
```



flight_id	flight_no	scheduled_departure	scheduled_arrival	departure_airport	arrival_airport	status	aircraft_code	actual_departure	actual_arrival
1489	PG0210	2017-08-16 18:00:00+03	2017-08-16 19:50:00+03	DME	MRV	On Time	733	\N	\N
19233	PG0510	2017-08-16 18:00:00+03	2017-08-16 19:30:00+03	ESL	DME	On Time	SU9	\N	\N
31184	PG0560	2017-08-16 18:00:00+03	2017-08-16 19:25:00+03	EGO	ROV	On Time	CN1	\N	\N
24141	PG0221	2017-08-16 17:55:00+03	2017-08-16 19:25:00+03	KRR	DME	On Time	763	\N	\N
14758	PG0590	2017-08-16 17:50:00+03	2017-08-16 18:15:00+03	PEE	SVX	On Time	SU9	\N	\N
17593	PG0060	2017-08-16 17:50:00+03	2017-08-16 20:10:00+03	NBC	SCW	On Time	CN1	\N	\N
5715	PG0152	2017-08-16 17:45:00+03	2017-08-16 19:40:00+03	SVO	MMK	On Time	SU9	\N	\N
8708	PG0507	2017-08-16 17:45:00+03	2017-08-16 19:25:00+03	LED	KZN	On Time	SU9	\N	\N
9715	PG0077	2017-08-16 17:45:00+03	2017-08-16 19:10:00+03	LED	CEE	On Time	CN1	\N	\N
11573	PG0203	2017-08-16 17:45:00+03	2017-08-16 18:40:00+03	KZN	DME	On Time	321	\N	\N
2005	PG0289	2017-08-16 17:25:00+03	2017-08-16 20:10:00+03	DME	VKT	On Time	CR2	\N	\N
8304	PG0231	2017-08-16 17:20:00+03	2017-08-16 18:10:00+03	LED	VKO	On Time	321	\N	\N
16995	PG0392	2017-08-16 17:20:00+03	2017-08-16 19:20:00+03	JOK	KRR	On Time	CR2	\N	\N
5990	PG0703	2017-08-16 17:15:00+03	2017-08-17 02:00:00+03	SVO	UUS	On Time	319	\N	\N
24932	PG0445	2017-08-16 17:15:00+03	2017-08-16 18:45:00+03	TJM	OVS	On Time	CN1	\N	\N
32041	PG0708	2017-08-16 17:15:00+03	2017-08-16 18:00:00+03	SGC	OVS	On Time	733	\N	\N
766	PG0054	2017-08-16 17:05:00+03	2017-08-16 18:20:00+03	DME	TBW	On Time	CN1	\N	\N
7765	PG0224	2017-08-16 17:05:00+03	2017-08-16 18:50:00+03	SVO	AER	On Time	773	\N	\N
29161	PG0197	2017-08-16 17:05:00+03	2017-08-16 18:35:00+03	KGD	DME	On Time	SU9	\N	\N
30057	PG0387	2017-08-16 17:05:00+03	2017-08-16 17:35:00+03	BZK	DME	On Time	SU9	\N	\N
22715	PG0686	2017-08-16 17:00:00+03	2017-08-16 19:35:00+03	OVS	LED	On Time	CR2	\N	\N
30532	PG0562	2017-08-16 17:00:00+03	2017-08-16 18:45:00+03	AER	VKO	On Time	763	\N	\N
24343	PG0646	2017-08-16 16:55:00+03	2017-08-16 17:55:00+03	RTW	DME	On Time	CR2	\N	\N
32208	PG0425	2017-08-16 16:55:00+03	2017-08-16 19:35:00+03	SGC	VKT	On Time	CN1	\N	\N
32898	PG0147	2017-08-16 16:55:00+03	2017-08-16 18:55:00+03	OGZ	VKO	On Time	SU9	\N	\N
9205	PG0271	2017-08-16 16:50:00+03	2017-08-16 19:25:00+03	LED	VKT	On Time	CR2	\N	\N
15052	PG0493	2017-08-16 16:50:00+03	2017-08-16 20:25:00+03	PEE	ARH	On Time	CN1	\N	\N
32284	PG0614	2017-08-16 16:50:00+03	2017-08-16 20:20:00+03	SGC	URS	On Time	CR2	\N	\N
5441	PG0317	2017-08-16 16:45:00+03	2017-08-16 18:05:00+03	SVO	ROV	On Time	733	\N	\N
23004	PG0707	2017-08-16 16:45:00+03	2017-08-16 17:30:00+03	OVS	SGC	On Time	733	\N	\N

4. PySpark Setup

- **Installed and Configured PySpark:**
  - Used `pip` to install PySpark.
  - Configured PySpark in Jupyter Notebook.
- **Loaded Dataset:**

```
from pyspark.sql import SparkSession
spark =
SparkSession.builder.appName("AirlineDataAnalysis").getOrCreate()
df_aircrafts = spark.read.csv("path/to/aircrafts_data.csv",
header=True, inferSchema=True)
df_airports = spark.read.csv("path/to/airports_data.csv",
header=True, inferSchema=True)


# Load data into PySpark DataFrames

aircrafts_data = spark.read.csv('AIR_CSV/aircrafts_data.csv',
header=True, inferSchema=True)

airports_data = spark.read.csv('AIR_CSV/airports_data.csv',
header=True, inferSchema=True)

boarding_passes = spark.read.csv('AIR_CSV/boarding_passes.csv',
header=True, inferSchema=True)
```

```
bookings = spark.read.csv('AIR_CSV/bookings.csv', header=True,
inferSchema=True)

flights = spark.read.csv('AIR_CSV/flights.csv', header=True,
inferSchema=True)


seats = spark.read.csv('AIR_CSV/seats.csv', header=True,
inferSchema=True)

ticket_flights = spark.read.csv('AIR_CSV/ticket_flights.csv',
header=True, inferSchema=True)

tickets = spark.read.csv('AIR_CSV/tickets.csv', header=True,
inferSchema=True)
```

# Show schema and data Of each table

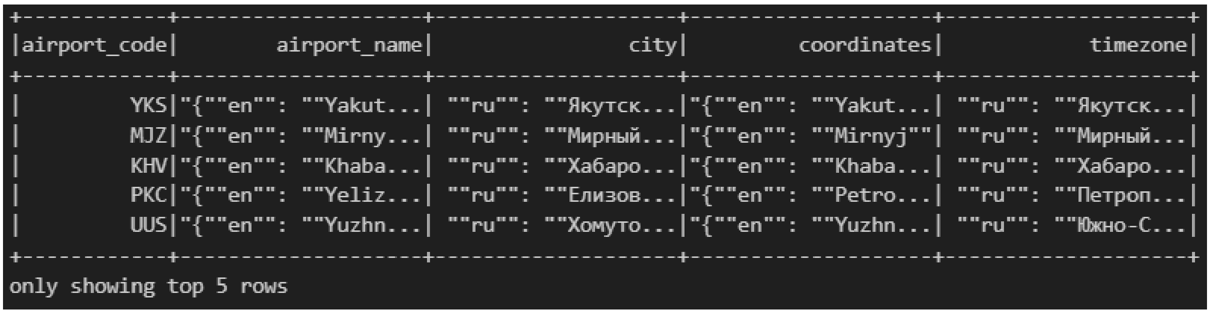
1. Table-01

```
aircrafts_data.printSchema()
root
 |-- aircraft_code: string (nullable = true)
 |-- model: string (nullable = true)
 |-- range: string (nullable = true)


aircrafts_data.show(5)
+-------------+--------------------+--------------------+
|aircraft_code|               model|               range|
+-------------+--------------------+--------------------+
|          773|"{""en"": ""Boein...| ""ru"": ""Боинг ...|
|          763|"{""en"": ""Boein...| ""ru"": ""Боинг ...|
|          SU9|"{""en"": ""Sukho...| ""ru"": ""Сухой ...|
|          320|"{""en"": ""Airbu...| ""ru"": ""Аэробу...|
|          321|"{""en"": ""Airbu...| ""ru"": ""Аэробу...|
+-------------+--------------------+--------------------+
only showing top 5 rows
```

2. Table-02

```
airports_data.printSchema()
root
 |-- airport_code: string (nullable = true)
 |-- airport_name: string (nullable = true)
 |-- city: string (nullable = true)
 |-- coordinates: string (nullable = true)
 |-- timezone: string (nullable = true)
```

```
#display the contents of a DataFrame in a tabular format
airports_data.show(5)
```



```
+------------+--------------------+--------------------+--------------------+--------------------+
|airport_code|        airport_name|                city|         coordinates|            timezone|
+------------+--------------------+--------------------+--------------------+--------------------+
|         YKS|"{""en"": ""Yakut...| ""ru"": ""Якутск...|"{""en"": ""Yakut...| ""ru"": ""Якутск...|
|         MJZ|"{""en"": ""Mirny...| ""ru"": ""Мирный...|"{""en"": ""Mirnyj""| ""ru"": ""Мирный...|
|         KHV|"{""en"": ""Khaba...| ""ru"": ""Хабаро...|"{""en"": ""Khaba...| ""ru"": ""Хабаро...|
|         PKC|"{""en"": ""Yeliz...| ""ru"": ""Елизов...|"{""en"": ""Petro...| ""ru"": ""Петроп...|
|         UUS|"{""en"": ""Yuzhn...| ""ru"": ""Хомуто...|"{""en"": ""Yuzhn...| ""ru"": ""Южно-С...|
+------------+--------------------+--------------------+--------------------+--------------------+
only showing top 5 rows
```

3. Table-3

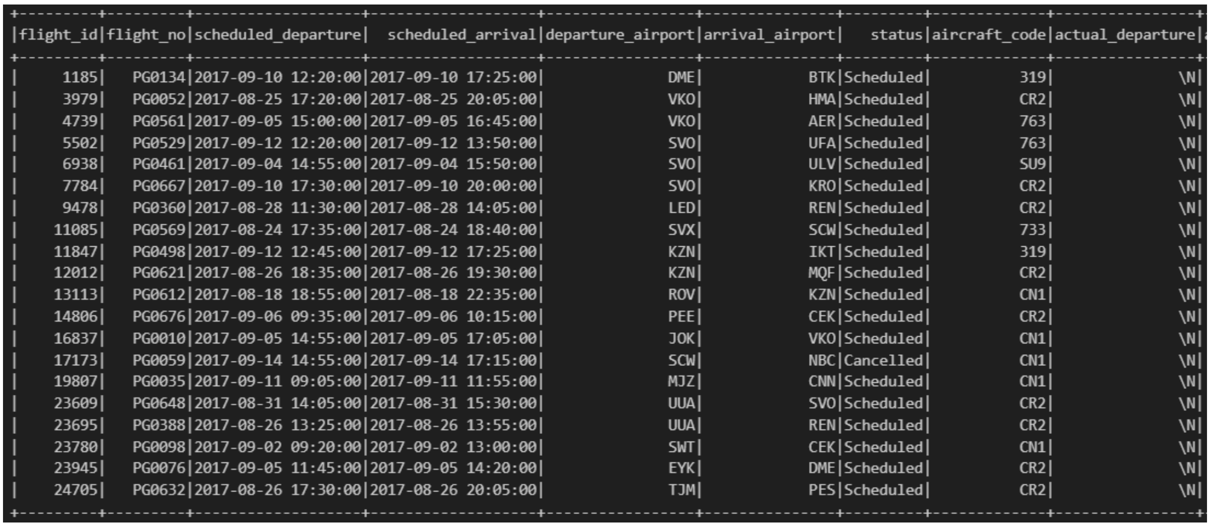
```
boarding_passes.printSchema()
root
 |-- ticket_no: long (nullable = true)
 |-- flight_id: integer (nullable = true)
 |-- boarding_no: integer (nullable = true)
 |-- seat_no: string (nullable = true)

#display the contents of a DataFrame in a tabular format
boarding_passes.show()
+----------+---------+-----------+-------+
| ticket_no|flight_id|boarding_no|seat_no|
+----------+---------+-----------+-------+
|5435212351|    30625|          1|     2D|
|5435212386|    30625|          2|     3G|
|5435212381|    30625|          3|     4H|
|5432211370|    30625|          4|     5D|
|5435212357|    30625|          5|    11A|
|5435212360|    30625|          6|    11E|
|5435212393|    30625|          7|    11H|
|5435212374|    30625|          8|    12E|
|5435212365|    30625|          9|    13D|
|5435212378|    30625|         10|    14H|
|5435212362|    30625|         11|    15E|
|5435212334|    30625|         12|    15F|
|5435212370|    30625|         13|    15K|
|5435212329|    30625|         14|    15H|
|5435725513|    30625|         15|    16D|
|5435212328|    30625|         16|    16C|
|5435630915|    30625|         17|    16E|
|5435212388|    30625|         18|    17E|
|5432159775|    30625|         19|    17D|
|5435212382|    30625|         20|    17H|
+----------+---------+-----------+-------+
only showing top 20 rows
```

4. Table-4

```
bookings.printSchema()
root
 |-- book_ref: string (nullable = true)
 |-- book_date: timestamp (nullable = true)
 |-- total_amount: integer (nullable = true)

#display the contents of a DataFrame in a tabular format
boarding_passes.show()
+----------+---------+-----------+-------+
| ticket_no|flight_id|boarding_no|seat_no|
+----------+---------+-----------+-------+
|5435212351|    30625|          1|     2D|
|5435212386|    30625|          2|     3G|
|5435212381|    30625|          3|     4H|
|5432211370|    30625|          4|     5D|
|5435212357|    30625|          5|    11A|
|5435212360|    30625|          6|    11E|
|5435212393|    30625|          7|    11H|
|5435212374|    30625|          8|    12E|
|5435212365|    30625|          9|    13D|
|5435212378|    30625|         10|    14H|
|5435212362|    30625|         11|    15E|
|5435212334|    30625|         12|    15F|
|5435212370|    30625|         13|    15K|
|5435212329|    30625|         14|    15H|
|5435725513|    30625|         15|    16D|
|5435212328|    30625|         16|    16C|
|5435630915|    30625|         17|    16E|
|5435212388|    30625|         18|    17E|
|5432159775|    30625|         19|    17D|
|5435212382|    30625|         20|    17H|
+----------+---------+-----------+-------+
only showing top 20 rows
```

5. Table-05:

```
flights.printSchema()
root
 |-- flight_id: integer (nullable = true)
 |-- flight_no: string (nullable = true)
 |-- scheduled_departure: timestamp (nullable = true)
 |-- scheduled_arrival: timestamp (nullable = true)
 |-- departure_airport: string (nullable = true)
 |-- arrival_airport: string (nullable = true)
 |-- status: string (nullable = true)
 |-- aircraft_code: string (nullable = true)
 |-- actual_departure: string (nullable = true)
 |-- actual_arrival: string (nullable = true)

#display the contents of a DataFrame in a tabular format
flights.show()
```



```
+---------+---------+-------------------+-------------------+-----------------+---------------+---------+-------------+----------------+
|flight_id|flight_no|scheduled_departure|  scheduled_arrival|departure_airport|arrival_airport|   status|aircraft_code|actual_departure|
+---------+---------+-------------------+-------------------+-----------------+---------------+---------+-------------+----------------+
|     1185|   PG0134|2017-09-10 12:20:00|2017-09-10 17:25:00|              DME|            BTK|Scheduled|          319|              \N|
|     3979|   PG0052|2017-08-25 17:20:00|2017-08-25 20:05:00|              VKO|            HMA|Scheduled|          CR2|              \N|
|     4739|   PG0561|2017-09-05 15:00:00|2017-09-05 16:45:00|              VKO|            AER|Scheduled|          763|              \N|
|     5502|   PG0529|2017-09-12 12:20:00|2017-09-12 13:50:00|              SVO|            UFA|Scheduled|          763|              \N|
|     6938|   PG0461|2017-09-04 14:55:00|2017-09-04 15:50:00|              SVO|            ULV|Scheduled|          SU9|              \N|
|     7784|   PG0667|2017-09-10 17:30:00|2017-09-10 20:00:00|              SVO|            KRO|Scheduled|          CR2|              \N|
|     9478|   PG0360|2017-08-28 11:30:00|2017-08-28 14:05:00|              LED|            REN|Scheduled|          CR2|              \N|
|    11085|   PG0569|2017-08-24 17:35:00|2017-08-24 18:40:00|              SVX|            SCW|Scheduled|          733|              \N|
|    11847|   PG0498|2017-09-12 12:45:00|2017-09-12 17:25:00|              KZN|            IKT|Scheduled|          319|              \N|
|    12012|   PG0621|2017-08-26 18:35:00|2017-08-26 19:30:00|              KZN|            MQF|Scheduled|          CR2|              \N|
|    13113|   PG0612|2017-08-18 18:55:00|2017-08-18 22:35:00|              ROV|            KZN|Scheduled|          CN1|              \N|
|    14806|   PG0676|2017-09-06 09:35:00|2017-09-06 10:15:00|              PEE|            CEK|Scheduled|          CR2|              \N|
|    16837|   PG0010|2017-09-05 14:55:00|2017-09-05 17:05:00|              JOK|            VKO|Scheduled|          CN1|              \N|
|    17173|   PG0059|2017-09-14 14:55:00|2017-09-14 17:15:00|              SCW|            NBC|Cancelled|          CN1|              \N|
|    19807|   PG0035|2017-09-11 09:05:00|2017-09-11 11:55:00|              MJZ|            CNN|Scheduled|          CN1|              \N|
|    23609|   PG0648|2017-08-31 14:05:00|2017-08-31 15:30:00|              UUA|            SVO|Scheduled|          CR2|              \N|
|    23695|   PG0388|2017-08-26 13:25:00|2017-08-26 13:55:00|              UUA|            REN|Scheduled|          CR2|              \N|
|    23780|   PG0098|2017-09-02 09:20:00|2017-09-02 13:00:00|              SWT|            CEK|Scheduled|          CN1|              \N|
|    23945|   PG0076|2017-09-05 11:45:00|2017-09-05 14:20:00|              EYK|            DME|Scheduled|          CR2|              \N|
|    24705|   PG0632|2017-08-26 17:30:00|2017-08-26 20:05:00|              TJM|            PES|Scheduled|          CR2|              \N|
+---------+---------+-------------------+-------------------+-----------------+---------------+---------+-------------+----------------+
```

6. Table-06

```
seats.printSchema()
root
 |-- aircraft_code: string (nullable = true)
 |-- seat_no: string (nullable = true)
 |-- fare_conditions: string (nullable = true)
```

```
#display the contents of a DataFrame in a tabular format
seats.show()
```

```
+-------------+-------+---------------+
|aircraft_code|seat_no|fare_conditions|
+-------------+-------+---------------+
|          319|     2A|       Business|
|          319|     2C|       Business|
|          319|     2D|       Business|
|          319|     2F|       Business|
|          319|     3A|       Business|
|          319|     3C|       Business|
|          319|     3D|       Business|
|          319|     3F|       Business|
|          319|     4A|       Business|
|          319|     4C|       Business|
|          319|     4D|       Business|
|          319|     4F|       Business|
|          319|     5A|       Business|
|          319|     5C|       Business|
|          319|     5D|       Business|
|          319|     5F|       Business|
|          319|     6A|        Economy|
|          319|     6B|        Economy|
|          319|     6C|        Economy|
|          319|     6D|        Economy|
+-------------+-------+---------------+
only showing top 20 rows
```

7. Table-07

```
ticket_flights.printSchema()
root
 |-- ticket_no: long (nullable = true)
 |-- flight_id: integer (nullable = true)
 |-- fare_conditions: string (nullable = true)
 |-- amount: integer (nullable = true)



#display the contents of a DataFrame in a tabular format
ticket_flights.show()
```

```
+----------+---------+---------------+------+
| ticket_no|flight_id|fare_conditions|amount|
+----------+---------+---------------+------+
|5432159776|    30625|       Business| 42100|
|5435212351|    30625|       Business| 42100|
|5435212386|    30625|       Business| 42100|
|5435212381|    30625|       Business| 42100|
|5432211370|    30625|       Business| 42100|
|5435212357|    30625|        Comfort| 23900|
|5435212360|    30625|        Comfort| 23900|
|5435212393|    30625|        Comfort| 23900|
|5435212374|    30625|        Comfort| 23900|
|5435212365|    30625|        Comfort| 23900|
|5435212378|    30625|        Comfort| 23900|
|5435212362|    30625|        Comfort| 23900|
|5435212334|    30625|        Comfort| 23900|
|5435212329|    30625|        Comfort| 23900|
|5435212370|    30625|        Comfort| 23900|
|5435212328|    30625|        Comfort| 23900|
|5435725513|    30625|        Comfort| 23900|
|5435630915|    30625|        Comfort| 23900|
|5435212388|    30625|        Economy| 14000|
|5432159775|    30625|        Economy| 14000|
+----------+---------+---------------+------+
only showing top 20 rows
```

8. Table-08

```
tickets.printSchema()
root
 |-- ticket_no: long (nullable = true)
 |-- book_ref: string (nullable = true)
 |-- passenger_id: string (nullable = true)
```

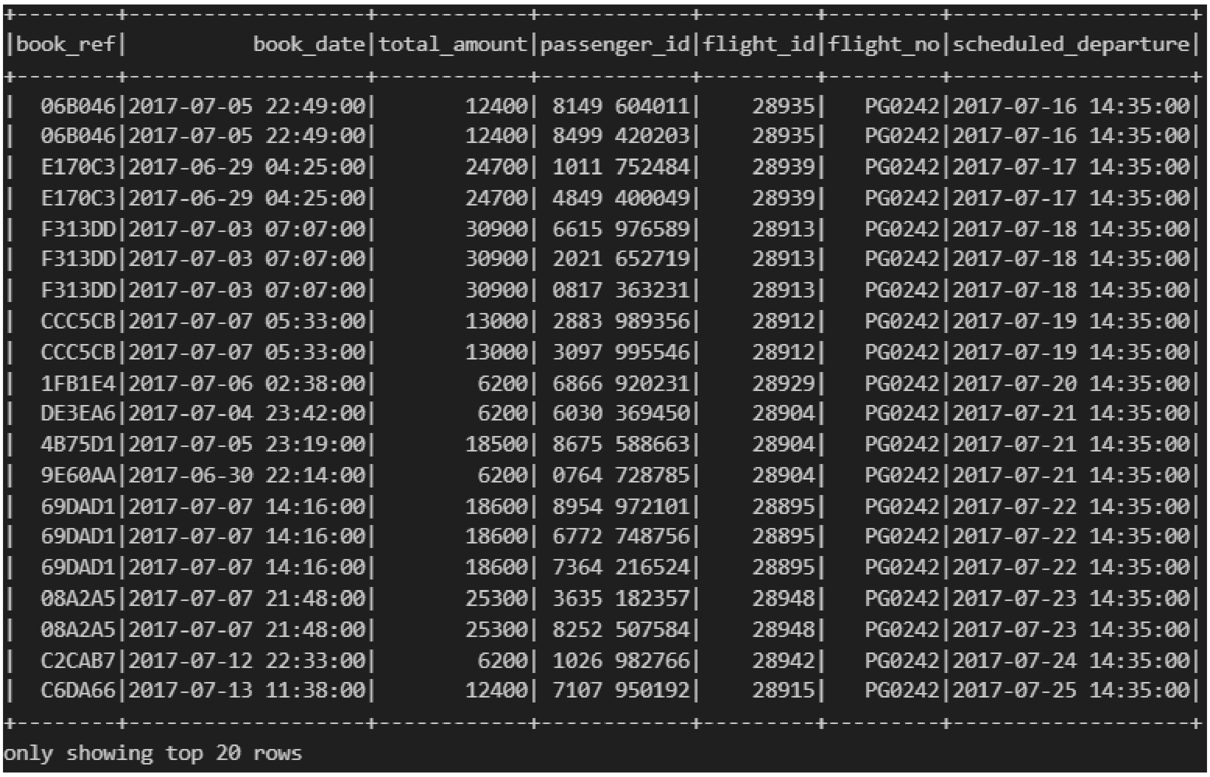
```
#display the contents of a DataFrame in a tabular format
tickets.show()
```

```
+----------+--------+------------+
| ticket_no|book_ref|passenger_id|
+----------+--------+------------+
|5432000987|  06B046| 8149 604011|
|5432000988|  06B046| 8499 420203|
|5432000989|  E170C3| 1011 752484|
|5432000990|  E170C3| 4849 400049|
|5432000991|  F313DD| 6615 976589|
|5432000992|  F313DD| 2021 652719|
|5432000993|  F313DD| 0817 363231|
|5432000994|  CCC5CB| 2883 989356|
|5432000995|  CCC5CB| 3097 995546|
|5432000996|  1FB1E4| 6866 920231|
|5432000997|  DE3EA6| 6030 369450|
|5432000998|  4B75D1| 8675 588663|
|5432000999|  9E60AA| 0764 728785|
|5432001000|  69DAD1| 8954 972101|
|5432001001|  69DAD1| 6772 748756|
|5432001002|  69DAD1| 7364 216524|
|5432001003|  08A2A5| 3635 182357|
|5432001004|  08A2A5| 8252 507584|
|5432001005|  C2CAB7| 1026 982766|
|5432001006|  C6DA66| 7107 950192|
+----------+--------+------------+
only showing top 20 rows
```

JOIN RELEVANT TABLE(PySpark)

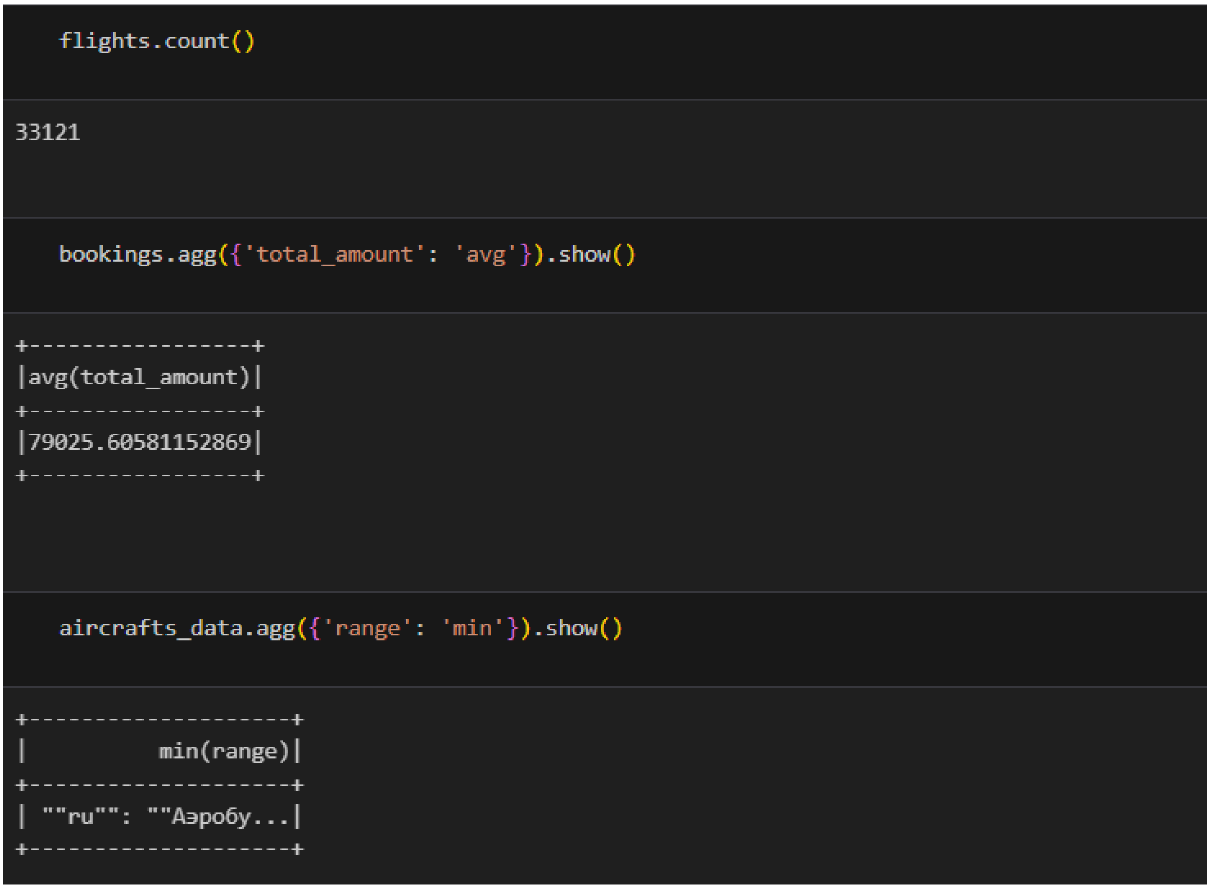
```
#Here I have joined bookings table with tickets_flights, ticket_no, flight_id tables
```

```
joined_df = bookings.join(tickets, 'book_ref') \
    .join(ticket_flights, 'ticket_no') \
    .join(flights, 'flight_id')
joined_df.select('book_ref', 'book_date', 'total_amount', 'passenger_id',
'flight_id', 'flight_no', 'scheduled_departure').show()
```



```
+--------+-------------------+------------+------------+---------+---------+-------------------+
|book_ref|          book_date|total_amount|passenger_id|flight_id|flight_no|scheduled_departure|
+--------+-------------------+------------+------------+---------+---------+-------------------+
|  06B046|2017-07-05 22:49:00|       12400| 8149 604011|    28935|   PG0242|2017-07-16 14:35:00|
|  06B046|2017-07-05 22:49:00|       12400| 8499 420203|    28935|   PG0242|2017-07-16 14:35:00|
|  E170C3|2017-06-29 04:25:00|       24700| 1011 752484|    28939|   PG0242|2017-07-17 14:35:00|
|  E170C3|2017-06-29 04:25:00|       24700| 4849 400049|    28939|   PG0242|2017-07-17 14:35:00|
|  F313DD|2017-07-03 07:07:00|       30900| 6615 976589|    28913|   PG0242|2017-07-18 14:35:00|
|  F313DD|2017-07-03 07:07:00|       30900| 2021 652719|    28913|   PG0242|2017-07-18 14:35:00|
|  F313DD|2017-07-03 07:07:00|       30900| 0817 363231|    28913|   PG0242|2017-07-18 14:35:00|
|  CCC5CB|2017-07-07 05:33:00|       13000| 2883 989356|    28912|   PG0242|2017-07-19 14:35:00|
|  CCC5CB|2017-07-07 05:33:00|       13000| 3097 995546|    28912|   PG0242|2017-07-19 14:35:00|
|  1FB1E4|2017-07-06 02:38:00|        6200| 6866 920231|    28929|   PG0242|2017-07-20 14:35:00|
|  DE3EA6|2017-07-04 23:42:00|        6200| 6030 369450|    28904|   PG0242|2017-07-21 14:35:00|
|  4B75D1|2017-07-05 23:19:00|       18500| 8675 588663|    28904|   PG0242|2017-07-21 14:35:00|
|  9E60AA|2017-06-30 22:14:00|        6200| 0764 728785|    28904|   PG0242|2017-07-21 14:35:00|
|  69DAD1|2017-07-07 14:16:00|       18600| 8954 972101|    28895|   PG0242|2017-07-22 14:35:00|
|  69DAD1|2017-07-07 14:16:00|       18600| 6772 748756|    28895|   PG0242|2017-07-22 14:35:00|
|  69DAD1|2017-07-07 14:16:00|       18600| 7364 216524|    28895|   PG0242|2017-07-22 14:35:00|
|  08A2A5|2017-07-07 21:48:00|       25300| 3635 182357|    28948|   PG0242|2017-07-23 14:35:00|
|  08A2A5|2017-07-07 21:48:00|       25300| 8252 507584|    28948|   PG0242|2017-07-23 14:35:00|
|  C2CAB7|2017-07-12 22:33:00|        6200| 1026 982766|    28942|   PG0242|2017-07-24 14:35:00|
|  C6DA66|2017-07-13 11:38:00|       12400| 7107 950192|    28915|   PG0242|2017-07-25 14:35:00|
+--------+-------------------+------------+------------+---------+---------+-------------------+
only showing top 20 rows
```

Some statistics summary



```
flights.count()
```

```
33121
```

```
bookings.agg({'total_amount': 'avg'}).show()
```

```
+-----------------+
|avg(total_amount)|
+-----------------+
|79025.60581152869|
+-----------------+
```

```
aircrafts_data.agg({'range': 'min'}).show()
```

```
+--------------------+
|          min(range)|
+--------------------+
| ""ru"": ""Аэробу...|
+--------------------+
```

5. Formulated Questions (4 easy , 5 medium , 6 hard level)

1. Retrieve the total number of flights in the dataset.
2. Find the average booking amount.
3. List distinct aircraft codes.
4. Identify distinct airport codes.
5. Count the number of flights per status.
6. Calculate the total booking amount per day.
7. Retrieve flights with a specific aircraft code.
8. List the top 5 busiest airports based on departures.
9. Find flights that departed on time.
10. Calculate the average delay time per airline.
11. Identify the day with the highest number of cancellations.
12. Calculate the percentage of on-time arrivals per airport.
13. Find the longest flight (based on scheduled time).
14. Determine the average flight range for each aircraft model.
15. Calculate the total revenue generated by each aircraft model.

6. SQL Solutions

- **Total Number of Flights:**

```
%%sql

SELECT COUNT(*) AS total_flights FROM flights;
```

- **Average booking amount:**

```
%%sql

SELECT ROUND(AVG(total_amount),3) as avg_booking_amount FROM bookings
```

- **List distinct aircraft codes:**

```
%%sql

SELECT DISTINCT aircraft_code FROM aircrafts_data
```

- **List distinct airports codes:**

```
%%sql

SELECT DISTINCT airport_code FROM airports_data
```

- **Count the numbers of flights per status:**

```
%%sql

SELECT status, COUNT(*) as flight_count FROM flights GROUP BY status;
```

- **Calculate the total booking amount per day**

```
%%sql

SELECT DATE(book_date) as booking_day, SUM(total_amount) as
total_amount FROM bookings GROUP BY booking_day;
```

- **Retrieve flights with a specific aircraft code**

```
%%sql

SELECT * FROM flights WHERE aircraft_code = 'CN1'
```

- **List the top 5 busiest airports based on departures**

```
%%sql

SELECT departure_airport, COUNT(*) as departure_count FROM flights
GROUP BY departure_airport ORDER BY departure_count DESC LIMIT 5
```

- **Find flights that departed on time**

```
%%sql

SELECT * FROM flights WHERE status = 'On Time'
```

- **Identify the day with the highest number of cancellation**

```
%%sql

SELECT DATE(scheduled_departure) as day, COUNT(*) as cancellations
FROM flights WHERE status = 'Cancelled' GROUP BY day ORDER BY
cancellations DESC LIMIT 1
```

- **Calculate the percentage of on-time arrivals per airport**

```
%%sql

SELECT f.arrival_airport, ROUND((SUM(CASE WHEN f.actual_arrival <=
f.scheduled_arrival THEN 1 ELSE 0 END) * 100.0 / COUNT(*)),2) AS
on_time_percentage FROM flights f WHERE f.actual_arrival IS NOT NULL
AND f.scheduled_arrival IS NOT NULL GROUP BY f.arrival_airport
```

- **Find the longest flight(Based on scheduled time)**

```
%%sql

SELECT f.flight_id, f.flight_no, f.scheduled_departure,
f.scheduled_arrival, (f.scheduled_arrival - f.scheduled_departure) AS
flight_duration FROM flights f ORDER BY flight_duration DESC LIMIT 1
```

- **Determine the average flight range for each aircraft model**

```
SELECT model, AVG(range) as avg_range FROM aircrafts_data GROUP BY
model
```

- **Calculate the total revenue generated by each aircraft model**

```
SELECT model, SUM(amount) as total_revenue FROM aircrafts_data ad
JOIN flights f ON ad.aircraft_code = f.aircraft_code JOIN
ticket_flights tf ON f.flight_id = tf.flight_id GROUP BY model
```

- **Calculate the average delay time per airline.**

```
%%sql

SELECT f.aircraft_code, AVG(EXTRACT(EPOCH FROM (f.actual_arrival -
f.scheduled_arrival))/60) AS avg_delay_minutes FROM flights f WHERE
f.actual_arrival IS NOT NULL AND f.scheduled_arrival IS NOT NULL
GROUP BY f.aircraft_code;
```

7. PySpark Solutions

1. Retrieve the total number of flights in the dataset.

```
  flights.count()
```

2. Find the average booking amount.

```
 bookings.agg({'total_amount': 'avg'}).show()
```

3. List distinct aircraft codes.

```
aircrafts_data.select('aircraft_code').distinct().show()
```

4. Identify distinct airport codes.

```
airports_data.select('airport_code').distinct().show()
```

5. Count the number of flights per status.

```
   flights.groupBy('status').count().show()
```

6. Calculate the total booking amount per day.

```
bookings.groupBy(bookings.book_date.cast('date')).agg({'total_
amount': 'sum'}).show()
```

7. Retrieve flights with a specific aircraft code.

```
flights.filter(flights.aircraft_code == 'CR2').show()
```

8. List the top 5 busiest airports based on departures.

```
flights.groupBy('departure_airport').count().orderBy('count',
ascending=False).limit(5).show()
```

9. Find flights that departed on time.

```
flights.filter(flights.status == 'On Time').show()
```

10. Calculate the average delay time per airline.

```
flights.withColumn('delay', (unix_timestamp('actual_arrival')
- unix_timestamp('scheduled_arrival')) /
60).groupBy('airline').avg('delay').show()
```

11. Identify the day with the highest number of cancellations.

```
flights.filter(flights.status ==
'Cancelled').groupBy(flights.scheduled_departure.cast('date'))
.count().orderBy('count', ascending=False).limit(1).show()
```

12. Calculate the percentage of on-time arrivals per airport.

```
flights.groupBy('arrival_airport').agg(expr("round(sum(case
when status = 'On Time' then 1 else 0 end) / count(*) * 100,
2)").alias('on_time_percentage')).show()
```

13. Find the longest flight (based on scheduled time).

```
flights.withColumn('flight_duration',
expr("(unix_timestamp(scheduled_arrival) -
unix_timestamp(scheduled_departure)) /
60")).orderBy('flight_duration',
ascending=False).limit(1).show()
```

14. Determine the average flight range for each aircraft model.

```
aircrafts_data.groupBy('model').agg({'range': 'avg'}).show()
```

15. Calculate the total revenue generated by each aircraft model.

```
aircrafts_data.join(flights,
'aircraft_code').join(ticket_flights,
'flight_id').groupBy('model').agg({'amount': 'sum'}).show()
```

Assumptions and Notes

- The dataset tables are properly normalized and foreign keys are correctly set.

- For SQL solutions, we assumed the use of standard SQL queries which should be compatible with SQLite.

- For PySpark solutions, the operations were performed on DataFrames, and common PySpark functions were used.

- The exact columns and data types were used as provided in the dataset schema.

- For some questions, additional columns like `delay` and `flight_duration` were computed based on timestamp differences.

- All calculations are rounded to two decimal places where applicable for better readability.

Conclusion

This project demonstrates the ability to handle a relational database and perform complex data analysis using both SQL and PySpark. The comprehensive approach ensures that we can derive meaningful insights from the dataset, leveraging the strengths of both SQL for relational data manipulation and PySpark for large-scale data processing.

All code, queries, and additional resources are available in the project repository on GitHub.