Objective

The goal of this data analysis project using sql would be to identify opportunities to increase the occupancy rate on low-performing flights, which can ultimately lead to increased profitability for the airline.

Importing Libraries

In [1]:
```
import sqlite3
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

Database Connection

In [2]:
```
connection = sqlite3.connect('travel.sqlite')
cursor = connection.cursor()
```

In [3]:
```
# extracting table names from the database
cursor.execute("""SELECT name FROM sqlite_master WHERE type='table';""")
print('List of Tables present in the Database')
table_list = [table[0] for table in cursor.fetchall()]
table_list
```

```
List of Tables present in the Database
```

Out[3]:
```
['aircrafts_data',
 'airports_data',
 'boarding_passes',
 'bookings',
 'flights',
 'seats',
 'ticket_flights',
 'tickets']
```

Data Exploration

In [4]:
```
aircrafts_data = pd.read_sql_query(f"""SELECT * FROM aircrafts_data""", connection)
aircrafts_data.head()
```

Out[4]:

	aircraft_code	model	range
0	773	{"en": "Boeing 777-300", "ru": "Боинг 777-300"}	11100
1	763	{"en": "Boeing 767-300", "ru": "Боинг 767-300"}	7900
2	SU9	{"en": "Sukhoi Superjet-100", "ru": "Сухой Суп...	3000
3	320	{"en": "Airbus A320-200", "ru": "Аэробус A320-...	5700
4	321	{"en": "Airbus A321-200", "ru": "Аэробус A321-...	5600

In [5]:
```
airports_data = pd.read_sql_query(f"""SELECT * FROM airports_data""", connection)
airports_data.head()
```

Out[5]:

	airport_code	airport_name	city	coordin
0	YKS	{"en": "Yakutsk Airport", "ru": "Якутск"}	{"en": "Yakutsk", "ru": "Якутск"}	(129.77099609375,62.0932998657226
1	MJZ	{"en": "Mirny Airport", "ru": "Мирный"}	{"en": "Mirnyj", "ru": "Мирный"}	(114.03900146484375,62.534698486328
2	KHV	{"en": "Khabarovsk-Novy Airport", "ru": "Хабар...	{"en": "Khabarovsk", "ru": "Хабаровск"}	(135.18800354004,48.5279998779300
3	PKC	{"en": "Yelizovo Airport", "ru": "Елизово"}	{"en": "Petropavlovsk", "ru": "Петропавловск-К...	(158.453994750976562,53.167900085449
4	UUS	{"en": "Yuzhno-Sakhalinsk Airport", "ru": "Хом...	{"en": "Yuzhno-Sakhalinsk", "ru": "Южно-Сахали...	(142.718002319335938,46.888698577880

In [6]:
```
boarding_passes = pd.read_sql_query(f"""SELECT * FROM boarding_passes""", connectio
boarding_passes.head()
```

Out[6]:

	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

In [7]:
```
bookings = pd.read_sql_query(f"""SELECT * FROM bookings """, connection)
bookings.head()
```

Out[7]:

	book_ref	book_date	total_amount
0	00000F	2017-07-05 03:12:00+03	265700
1	000012	2017-07-14 09:02:00+03	37900
2	000068	2017-08-15 14:27:00+03	18100
3	000181	2017-08-10 13:28:00+03	131800
4	0002D8	2017-08-07 21:40:00+03	23600

In [8]:
```
flights = pd.read_sql_query(f"""SELECT * FROM flights  """, connection)
flights.head()
```

Out[8]:

	flight_id	flight_no	scheduled_departure	scheduled_arrival	departure_airport	arriv
0	1185	PG0134	2017-09-10 09:50:00+03	2017-09-10 14:55:00+03	DME	
1	3979	PG0052	2017-08-25 14:50:00+03	2017-08-25 17:35:00+03	VKO	
2	4739	PG0561	2017-09-05 12:30:00+03	2017-09-05 14:15:00+03	VKO	
3	5502	PG0529	2017-09-12 09:50:00+03	2017-09-12 11:20:00+03	SVO	
4	6938	PG0461	2017-09-04 12:25:00+03	2017-09-04 13:20:00+03	SVO	

In [9]:
```
seats = pd.read_sql_query(f"""SELECT * FROM seats  """, connection)
seats.head()
```

Out[9]:

	aircraft_code	seat_no	fare_conditions
0	319	2A	Business
1	319	2C	Business
2	319	2D	Business
3	319	2F	Business
4	319	3A	Business

In [10]:
```
ticket_flights = pd.read_sql_query(f"""SELECT * FROM ticket_flights  """, connectio
ticket_flights.head()
```

Out[10]:

	ticket_no	flight_id	fare_conditions	amount
0	0005432159776	30625	Business	42100
1	0005435212351	30625	Business	42100
2	0005435212386	30625	Business	42100
3	0005435212381	30625	Business	42100
4	0005432211370	30625	Business	42100

In [11]:
```
tickets = pd.read_sql_query(f"""SELECT * FROM tickets  """, connection)
tickets.head()
```

Out[11]:

	ticket_no	book_ref	passenger_id
0	0005432000987	06B046	8149 604011
1	0005432000988	06B046	8499 420203
2	0005432000989	E170C3	1011 752484
3	0005432000990	E170C3	4849 400049
4	0005432000991	F313DD	6615 976589

In [12]:
```
for table in table_list:
    print("\ntable: " + table)
    columns_info = connection.execute("PRAGMA table_info({})".format(table))
    for column in columns_info.fetchall():
        print(column[1:3])
```

```
table: aircrafts_data
('aircraft_code', 'character(3)')
('model', 'jsonb')
('range', 'INTEGER')

table: airports_data
('airport_code', 'character(3)')
('airport_name', 'jsonb')
('city', 'jsonb')
('coordinates', 'point')
('timezone', 'TEXT')

table: boarding_passes
('ticket_no', 'character(13)')
('flight_id', 'INTEGER')
('boarding_no', 'INTEGER')
('seat_no', 'character varying(4)')

table: bookings
('book_ref', 'character(6)')
('book_date', 'timestamp with time zone')
('total_amount', 'numeric(10,2)')

table: flights
('flight_id', 'INTEGER')
('flight_no', 'character(6)')
('scheduled_departure', 'timestamp with time zone')
('scheduled_arrival', 'timestamp with time zone')
('departure_airport', 'character(3)')
('arrival_airport', 'character(3)')
('status', 'character varying(20)')
('aircraft_code', 'character(3)')
('actual_departure', 'timestamp with time zone')
('actual_arrival', 'timestamp with time zone')

table: seats
('aircraft_code', 'character(3)')
('seat_no', 'character varying(4)')
('fare_conditions', 'character varying(10)')

table: ticket_flights
('ticket_no', 'character(13)')
('flight_id', 'INTEGER')
('fare_conditions', 'character varying(10)')
('amount', 'numeric(10,2)')

table: tickets
('ticket_no', 'character(13)')
('book_ref', 'character(6)')
('passenger_id', 'character varying(20)')
```

Data Cleaning

In [13]:
```
# checking for missing values in each column for every table
for table in table_list:
    print(f'\nMissing Values in table {table}')
    df_table = pd.read_sql_query(f"""SELECT * FROM {table}""", connection)
    print(df_table.isnull().sum())
```

```
Missing Values in table aircrafts_data
aircraft_code    0
model            0
range            0
dtype: int64

Missing Values in table airports_data
airport_code    0
airport_name    0
city            0
coordinates     0
timezone        0
dtype: int64

Missing Values in table boarding_passes
ticket_no      0
flight_id      0
boarding_no    0
seat_no        0
dtype: int64

Missing Values in table bookings
book_ref        0
book_date       0
total_amount    0
dtype: int64

Missing Values in table flights
flight_id              0
flight_no              0
scheduled_departure    0
scheduled_arrival      0
departure_airport      0
arrival_airport        0
status                 0
aircraft_code          0
actual_departure       0
actual_arrival         0
dtype: int64

Missing Values in table seats
aircraft_code      0
seat_no            0
fare_conditions    0
dtype: int64

Missing Values in table ticket_flights
ticket_no          0
flight_id          0
fare_conditions    0
amount             0
dtype: int64

Missing Values in table tickets
ticket_no       0
book_ref        0
```

```
passenger_id    0
dtype: int64
```

Basic Analysis

How many planes have more than 100 seats?

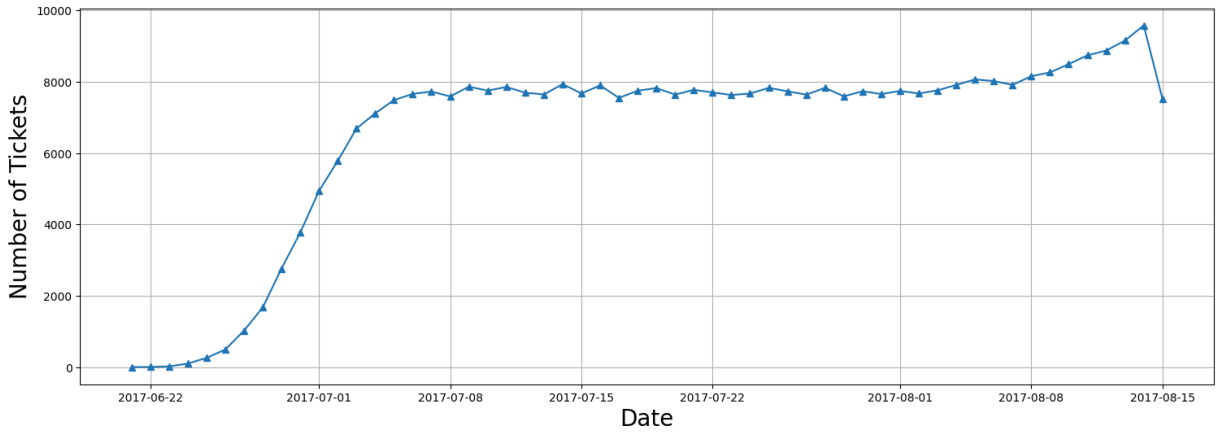
In [14]:
```
pd.read_sql_query(f"""SELECT aircraft_code, COUNT(*) as num_seats FROM seats
                      GROUP BY aircraft_code
                      HAVING num_seats > 100
                      ORDER BY num_seats DESC""", connection)
```

Out[14]:

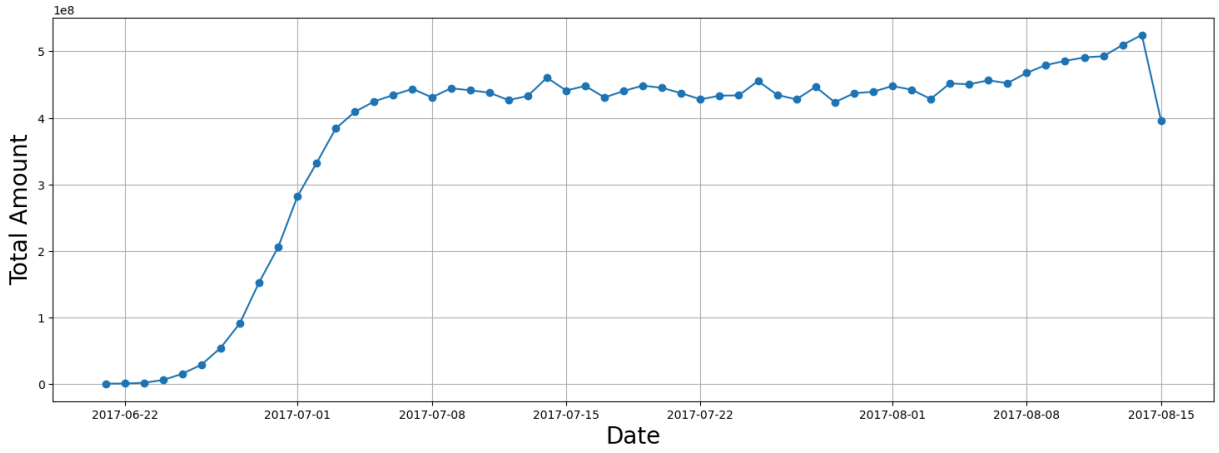
	aircraft_code	num_seats
0	773	402
1	763	222
2	321	170
3	320	140
4	733	130
5	319	116

How the number of tickets booked and total amount earned changed with the time.

In [15]:
```
tickets = pd.read_sql_query(f"""SELECT *
                                FROM tickets
                                INNER JOIN bookings
                                ON tickets.book_ref=bookings.book_ref;""", conn
tickets['book_date'] = pd.to_datetime(tickets['book_date'])
tickets['date'] = tickets['book_date'].dt.date
x =  tickets.groupby('date')[['date']].count()
plt.figure(figsize = (18,6))
plt.plot(x.index,x['date'], marker = '^')
plt.xlabel('Date', fontsize = 20)
plt.ylabel('Number of Tickets', fontsize = 20)
plt.grid('b')
plt.show()
```



In [16]:
```
bookings = pd.read_sql_query(f"""SELECT * FROM bookings""", connection)
bookings['book_date'] = pd.to_datetime(bookings['book_date'])
bookings['date'] = bookings['book_date'].dt.date
y =  bookings.groupby('date')[['total_amount']].sum()
plt.figure(figsize = (18,6))
plt.plot(y.index,y['total_amount'], marker = 'o')
plt.xlabel('Date', fontsize = 20)
plt.ylabel('Total Amount', fontsize = 20)
plt.grid('b')
plt.show()
```

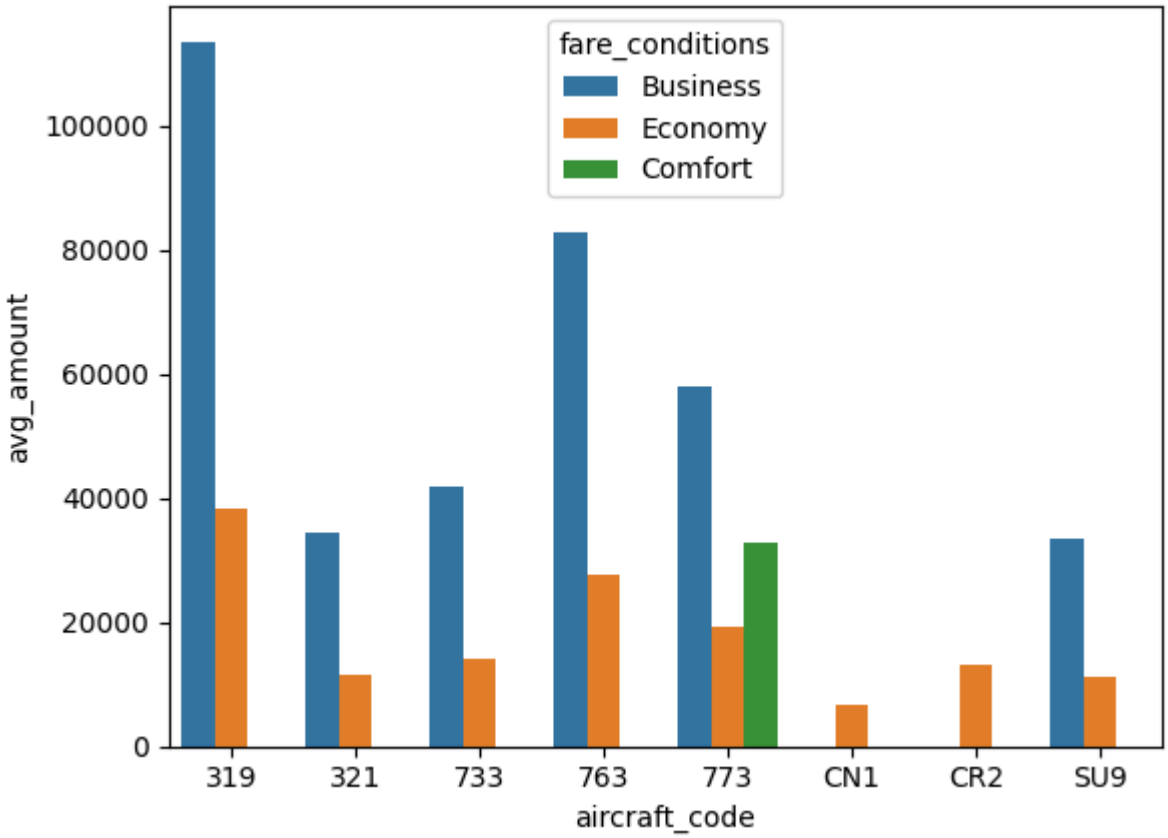


Calculate the average charges for each aircraft with different fare conditions.

In [17]:
```
df = pd.read_sql_query(f"""SELECT fare_conditions, aircraft_code, AVG(amount) as av
                           JOIN flights
                           ON ticket_flights.flight_id=flights.flight_id
                           GROUP BY aircraft_code, fare_conditions""", connection)

sns.barplot(data = df, x = 'aircraft_code', y ='avg_amount', hue = 'fare_conditions
```

Out[17]:
```
<AxesSubplot:xlabel='aircraft_code', ylabel='avg_amount'>
```



Analyzing occupancy rate

For each aircraft, calculate the total revenue per year and the average revenue per ticket.

In [18]:
```
pd.set_option('display.float_format', str)
```

In [19]:
```
pd.read_sql_query(f"""SELECT aircraft_code, total_revenue, ticket_count, total_reve
                      FROM
                      (SELECT aircraft_code, COUNT(*) as ticket_count, SUM(amount
                      JOIN flights
                      ON ticket_flights.flight_id=flights.flight_id
                      GROUP BY aircraft_code)""", connection)
```

Out[19]:

	aircraft_code	total_revenue	ticket_count	avg_revenue_per_ticket
0	319	2706163100	52853	51201
1	321	1638164100	107129	15291
2	733	1426552100	86102	16568
3	763	4371277100	124774	35033
4	773	3431205500	144376	23765
5	CN1	96373800	14672	6568
6	CR2	1982760500	150122	13207
7	SU9	5114484700	365698	13985

Calculate the average occupancy per aircraft.

In [20]:
```
occupancy_rate = pd.read_sql_query(f"""SELECT a.aircraft_code, AVG(a.seats_count) a
AVG(a.seats_count)/b.num_seats as occupancy_rate
                            FROM (
                                SELECT aircraft_code, flights.flight_id, COUNT(*) a
                                FROM boarding_passes
                                INNER JOIN flights
                                ON boarding_passes.flight_id=flights.flight_id
                                GROUP BY aircraft_code, flights.flight_id
                                ) as a INNER JOIN
                                (
                                SELECT aircraft_code, COUNT(*) as num_seats FROM se
                                GROUP BY aircraft_code
                                ) as b
                                ON a.aircraft_code = b.aircraft_code
                            GROUP BY a.aircraft_code""", connection)
occupancy_rate
```

Out[20]:

	aircraft_code	booked_seats	num_seats	occupancy_rate
0	319	53.58318098720292	116	0.46192397402761143
1	321	88.80923076923077	170	0.5224072398190045
2	733	80.25546218487395	130	0.617349709114415
3	763	113.93729372937294	222	0.5132310528350132
4	773	264.9258064516129	402	0.659019419033863
5	CN1	6.004431314623338	12	0.5003692762186115
6	CR2	21.48284690220174	50	0.42965693804403476
7	SU9	56.81211267605634	97	0.5856918832583128

Calculate by how much the total annual turnover could increase by giving all aircraft a 10% higher occupancy rate.

In [21]:
```
occupancy_rate['Inc occupancy rate'] = occupancy_rate['occupancy_rate'] + occupancy
occupancy_rate
```

Out[21]:

	aircraft_code	booked_seats	num_seats	occupancy_rate	Inc occupancy
0	319	53.58318098720292	116	0.46192397402761143	0.508116371430
1	321	88.80923076923077	170	0.5224072398190045	0.57464796380
2	733	80.25546218487395	130	0.617349709114415	0.679084680025
3	763	113.93729372937294	222	0.5132310528350132	0.564554158118
4	773	264.9258064516129	402	0.659019419033863	0.724921360937
5	CN1	6.004431314623338	12	0.5003692762186115	0.550406203840
6	CR2	21.48284690220174	50	0.42965693804403476	0.472622631848
7	SU9	56.81211267605634	97	0.5856918832583128	0.64426107158

In [22]:
```
total_revenue = pd.read_sql_query("""SELECT aircraft_code, SUM(amount) as total_rev
                        JOIN flights
                        ON ticket_flights.flight_id=flights.flight_id
                        GROUP BY aircraft_code""", connection)

occupancy_rate['Inc Total Annual Turnover'] = (total_revenue['total_revenue']/occup
occupancy_rate
```

Out[22]:

	aircraft_code	booked_seats	num_seats	occupancy_rate	Inc occupancy
0	319	53.58318098720292	116	0.46192397402761143	0.508116371430
1	321	88.80923076923077	170	0.5224072398190045	0.57464796380
2	733	80.25546218487395	130	0.617349709114415	0.679084680025
3	763	113.93729372937294	222	0.5132310528350132	0.564554158118
4	773	264.9258064516129	402	0.659019419033863	0.724921360937
5	CN1	6.004431314623338	12	0.5003692762186115	0.550406203840
6	CR2	21.48284690220174	50	0.42965693804403476	0.472622631848
7	SU9	56.81211267605634	97	0.5856918832583128	0.64426107158

In [3]:
```

```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[3], line 1
----> 1 select * aircrafts_data;

NameError: name 'select' is not defined
```

In [ ]: