```
# Importing Libraries

import sqlite3
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import json

import warnings
warnings.filterwarnings('ignore')
```

```
#Connecting to database

conn = sqlite3.connect('/content/travel.sqlite')
cursor = conn.cursor()
```

```
# List of tables

tables = pd.read_sql("""SELECT *
                        FROM sqlite_master
                        WHERE type='table';""", conn)
tables
```

	type	name	tbl_name	rootpage	sql
0	table	aircrafts_data	aircrafts_data	2	CREATE TABLE aircrafts_data (\r\n aircraft_...
1	table	airports_data	airports_data	3	CREATE TABLE airports_data (\r\n airport_co...
2	table	boarding_passes	boarding_passes	4	CREATE TABLE boarding_passes (\r\n ticket_n...
3	table	bookings	bookings	5	CREATE TABLE bookings (\r\n book_ref charac...
4	table	flights	flights	6	CREATE TABLE flights (\r\n flight_id intege...
5	table	seats	seats	7	CREATE TABLE seats (\r\n aircraft_code char...
6	table	ticket_flights	ticket_flights	8	CREATE TABLE ticket_flights (\r\n ticket_no...
7	table	tickets	tickets	9	CREATE TABLE tickets (\r\n ticket_no charac...

Next steps: Generate code with tables | View recommended plots | New interactive sheet

```
# Data Exploration

aircrafts_data = pd.read_sql_query("select * from aircrafts_data", conn)
aircrafts_data
```

	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
5	319	{"en": "Airbus A319-100", "ru": "Аэробус A319-...	6700
6	733	{"en": "Boeing 737-300", "ru": "Боинг 737-300"}	4200
7	CN1	{"en": "Cessna 208 Caravan", "ru": "Сессна 208...	1200
8	CR2	{"en": "Bombardier CRJ-200", "ru": "Бомбардье ...	2700

Next steps: Generate code with aircrafts_data | View recommended plots | New interactive sheet

```
aircrafts_data['model'] = aircrafts_data['model'].apply(lambda x: json.loads(x)['en'])
aircrafts_data
```

	aircraft_code	model	range
0	773	Boeing 777-300	11100
1	763	Boeing 767-300	7900
2	SU9	Sukhoi Superjet-100	3000
3	320	Airbus A320-200	5700
4	321	Airbus A321-200	5600
5	319	Airbus A319-100	6700
6	733	Boeing 737-300	4200
7	CN1	Cessna 208 Caravan	1200
8	CR2	Bombardier CRJ-200	2700

Next steps: Generate code with aircrafts_data | View recommended plots | New interactive sheet

```
airports_data = pd.read_sql_query("select * from airports_data", conn)
airports_data
```

	airport_code	airport_name	city	coordinates	timezone
0	YKS	{"en": "Yakutsk Airport", "ru": "Якутск"}	{"en": "Yakutsk", "ru": "Якутск"}	(129.77099609375,62.0932998657226562)	Asia/Yakutsk
1	MJZ	{"en": "Mirny Airport", "ru": "Мирный"}	{"en": "Mirnyj", "ru": "Мирный"}	(114.03900146484375,62.534698486328125)	Asia/Yakutsk
2	KHV	{"en": "Khabarovsk-Novy Airport", "ru": "Хабар...	{"en": "Khabarovsk", "ru": "Хабаровск"}	(135.18800354004,48.5279998779300001)	Asia/Vladivostok
3	PKC	{"en": "Yelizovo Airport", "ru": "Елизово"}	{"en": "Petropavlovsk", "ru": "Петропавловск-К...	(158.453994750976562,53.1679000854492188)	Asia/Kamchatka
4	UUS	{"en": "Yuzhno-Sakhalinsk Airport", "ru": "Хом...	{"en": "Yuzhno-Sakhalinsk", "ru": "Южно-Сахали...	(142.718002319335938,46.8886985778808594)	Asia/Sakhalin
...	...	...	...	...	...
99	MMK	{"en": "Murmansk Airport", "ru": "Мурманск"}	{"en": "Murmansk", "ru": "Мурманск"}	(32.7508010864257812,68.7817001342773438)	Europe/Moscow
100	ABA	{"en": "Abakan Airport", "ru": "Абакан"}	{"en": "Abakan", "ru": "Абакан"}	(91.3850021362304688,53.7400016784667969)	Asia/Krasnoyarsk
101	BAX	{"en": "Barnaul Airport", "ru": "Барнаул"}	{"en": "Barnaul", "ru": "Барнаул"}	(83.5384979248046875,53.363800048828125)	Asia/Krasnoyarsk
102	AAQ	{"en": "Anapa Vityazevo Airport", "ru": "Витяз...	{"en": "Anapa", "ru": "Анапа"}	(37.3473014831539984,45.002101898192997)	Europe/Moscow
103	CNN	{"en": "Chulman Airport", "ru": "Чульман"}	{"en": "Neryungri", "ru": "Нерюнгри"}	(124.914001464839998,56.9138984680179973)	Asia/Yakutsk

104 rows × 5 columns

Next steps: Generate code with airports_data | View recommended plots | New interactive sheet

```
airports_data['airport_name'] = airports_data['airport_name'].apply(lambda x: json.loads(x)['en'])
airports_data['city'] = airports_data['city'].apply(lambda x: json.loads(x)['en'])
```

```
airports_data
```

	airport_code	airport_name	city	coordinates	timezone
0	YKS	Yakutsk Airport	Yakutsk	(129.77099609375,62.0932998657226562)	Asia/Yakutsk
1	MJZ	Mirny Airport	Mirnyj	(114.03900146484375,62.534698486328125)	Asia/Yakutsk
2	KHV	Khabarovsk-Novy Airport	Khabarovsk	(135.18800354004,48.5279998779300001)	Asia/Vladivostok
3	PKC	Yelizovo Airport	Petropavlovsk	(158.453994750976562,53.1679000854492188)	Asia/Kamchatka
4	UUS	Yuzhno-Sakhalinsk Airport	Yuzhno-Sakhalinsk	(142.718002319335938,46.8886985778808594)	Asia/Sakhalin
...	...	...	...	...	...
99	MMK	Murmansk Airport	Murmansk	(32.7508010864257812,68.7817001342773438)	Europe/Moscow
100	ABA	Abakan Airport	Abakan	(91.3850021362304688,53.7400016784667969)	Asia/Krasnoyarsk
101	BAX	Barnaul Airport	Barnaul	(83.5384979248046875,53.363800048828125)	Asia/Krasnoyarsk
102	AAQ	Anapa Vityazevo Airport	Anapa	(37.3473014831539984,45.002101898192997)	Europe/Moscow
103	CNN	Chulman Airport	Neryungri	(124.914001464839998,56.9138984680179973)	Asia/Yakutsk

104 rows × 5 columns

Next steps: Generate code with `airports_data` | View recommended plots | New interactive sheet

```
boarding_passes = pd.read_sql_query("select * from boarding_passes", conn)
boarding_passes
```

	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
...	...	...	...	...
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

579686 rows × 4 columns

```
bookings = pd.read_sql_query("select * from bookings", conn)
bookings
```

	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
...	...	...	...
262783	FFFEF3	2017-07-17 07:23:00+03	56000
262784	FFFF2C	2017-08-08 05:55:00+03	10800
262785	FFFF43	2017-07-20 20:42:00+03	78500
262786	FFFFA8	2017-08-08 04:45:00+03	28800
262787	FFFFF7	2017-07-01 22:12:00+03	73600

262788 rows × 3 columns

```
flights = pd.read_sql_query("select * from flights", conn)
flights
```

	flight_id	flight_no	scheduled_departure	scheduled_arrival	departure_airport	arrival_airport	status	aircraft_code	a
0	1185	PG0134	2017-09-10 09:50:00+03	2017-09-10 14:55:00+03	DME	BTK	Scheduled	319	
1	3979	PG0052	2017-08-25 14:50:00+03	2017-08-25 17:35:00+03	VKO	HMA	Scheduled	CR2	
2	4739	PG0561	2017-09-05 12:30:00+03	2017-09-05 14:15:00+03	VKO	AER	Scheduled	763	
3	5502	PG0529	2017-09-12 09:50:00+03	2017-09-12 11:20:00+03	SVO	UFA	Scheduled	763	
4	6938	PG0461	2017-09-04 12:25:00+03	2017-09-04 13:20:00+03	SVO	ULV	Scheduled	SU9	
...	...	...	...	...	...	...	...	...	
33116	33117	PG0063	2017-08-02 19:25:00+03	2017-08-02 20:10:00+03	SKX	SVO	Arrived	CR2	
33117	33118	PG0063	2017-07-28 19:25:00+03	2017-07-28 20:10:00+03	SKX	SVO	Arrived	CR2	
33118	33119	PG0063	2017-09-08 19:25:00+03	2017-09-08 20:10:00+03	SKX	SVO	Scheduled	CR2	
33119	33120	PG0063	2017-08-01 19:25:00+03	2017-08-01 20:10:00+03	SKX	SVO	Arrived	CR2	
33120	33121	PG0063	2017-08-26 19:25:00+03	2017-08-26 20:10:00+03	SKX	SVO	Scheduled	CR2	

33121 rows × 10 columns

Next steps: Generate code with flights | View recommended plots | New interactive sheet

```
seats = pd.read_sql_query("select * from seats", conn)
seats
```

	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
...	...	...	...
1334	773	48H	Economy
1335	773	48K	Economy
1336	773	49A	Economy
1337	773	49C	Economy
1338	773	49D	Economy

1339 rows × 3 columns

Next steps: Generate code with seats | View recommended plots | New interactive sheet

```
ticket_flights = pd.read_sql_query("select * from ticket_flights", conn)
ticket_flights
```

	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
...	...	...	...	...
1045721	0005435097522	32094	Economy	5200
1045722	0005435097521	32094	Economy	5200
1045723	0005435104384	32094	Economy	5200
1045724	0005435104352	32094	Economy	5200
1045725	0005435104389	32094	Economy	5200

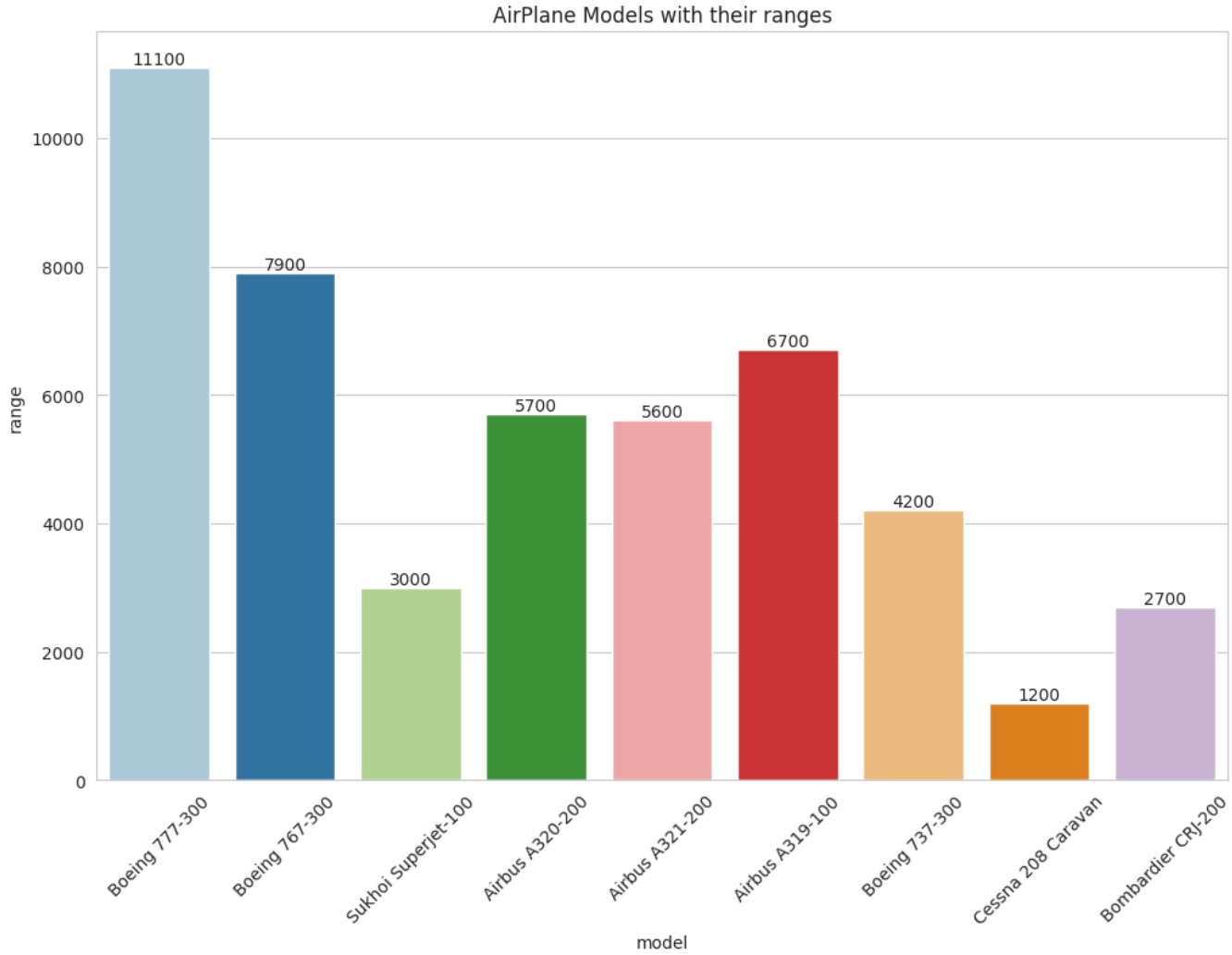
1045726 rows × 4 columns

```
tickets = pd.read_sql_query("select * from tickets", conn)
tickets
```

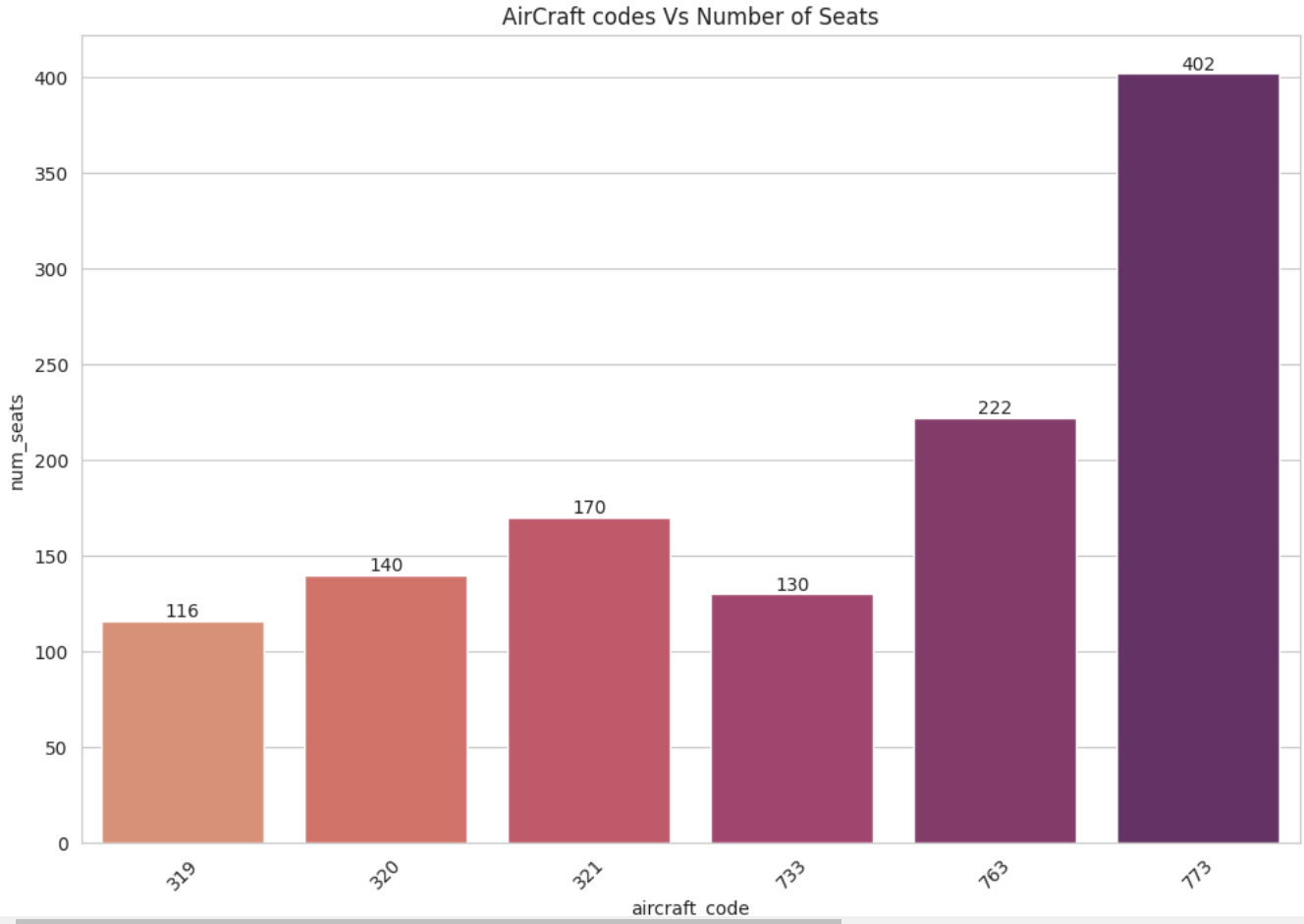
	ticket_no	book_ref	passenger_id
0	0005432000987	06B046	8149 604011
1	0005432000988	06B046	8499 420203
2	0005432000989	E170C3	1011 752484
3	0005432000990	E170C3	4849 400049
4	0005432000991	F313DD	6615 976589
...	...	...	...
366728	0005435999869	D730BA	0474 690760
366729	0005435999870	D730BA	6535 751108
366730	0005435999871	A1AD46	1596 156448
366731	0005435999872	7B6A53	9374 822707
366732	0005435999873	7B6A53	7380 075822

366733 rows × 3 columns

```
# Visualizations

sns.set_style('whitegrid')
fig,axes = plt.subplots(figsize=(12,8))
ax = sns.barplot(x='model',y='range', data=aircrafts_data, palette = 'Paired')
for container in ax.containers:
    ax.bar_label(container)
plt.title('AirPlane Models with their ranges')
plt.xticks(rotation=45)
plt.show()
```




```
# Planes having more than 100 seats

df = pd.read_sql_query("""select aircraft_code, count(*) as num_seats from seats
                        group by aircraft_code having num_seats >100""", conn)

df.to_csv('aircraft_seats.csv')


sns.set_style('whitegrid')
fig,axes = plt.subplots(figsize=(12,8))
ax = sns.barplot(x='aircraft_code',y='num_seats', data=df, palette = 'flare')
for container in ax.containers:
    ax.bar_label(container)
plt.title('AirCraft codes Vs Number of Seats')
plt.xticks(rotation=45)
plt.show()
```

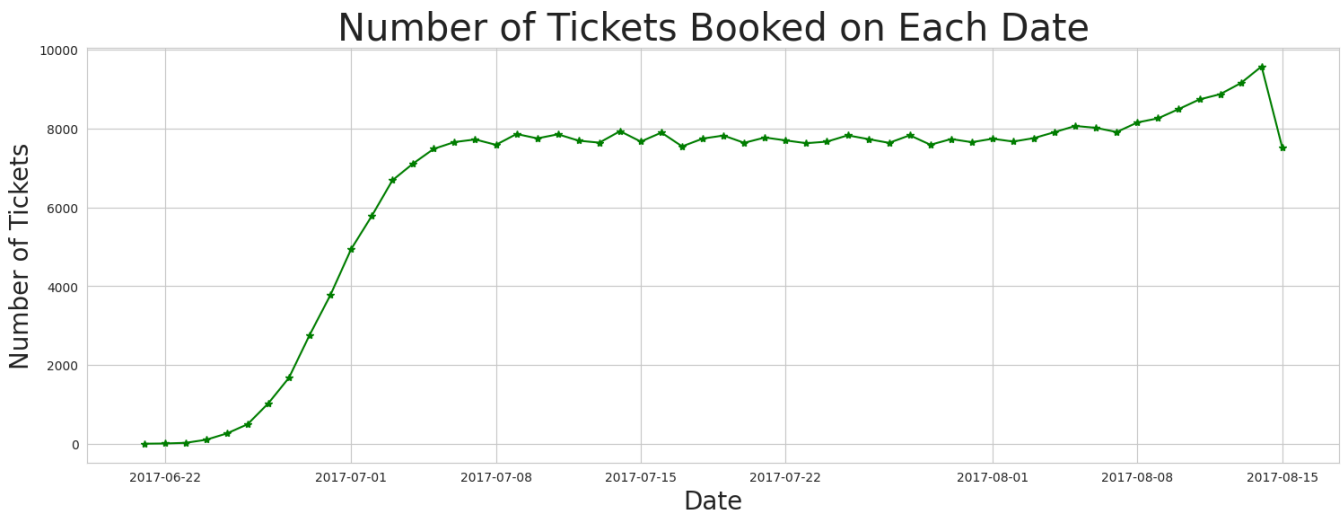



```
crafts = pd.read_sql("""SELECT aircraft_code, json_extract(model, '$.en')
                        FROM aircrafts_data
                        where aircraft_code IN (319, 320, 321, 733, 763, 773);""", conn)
crafts
```

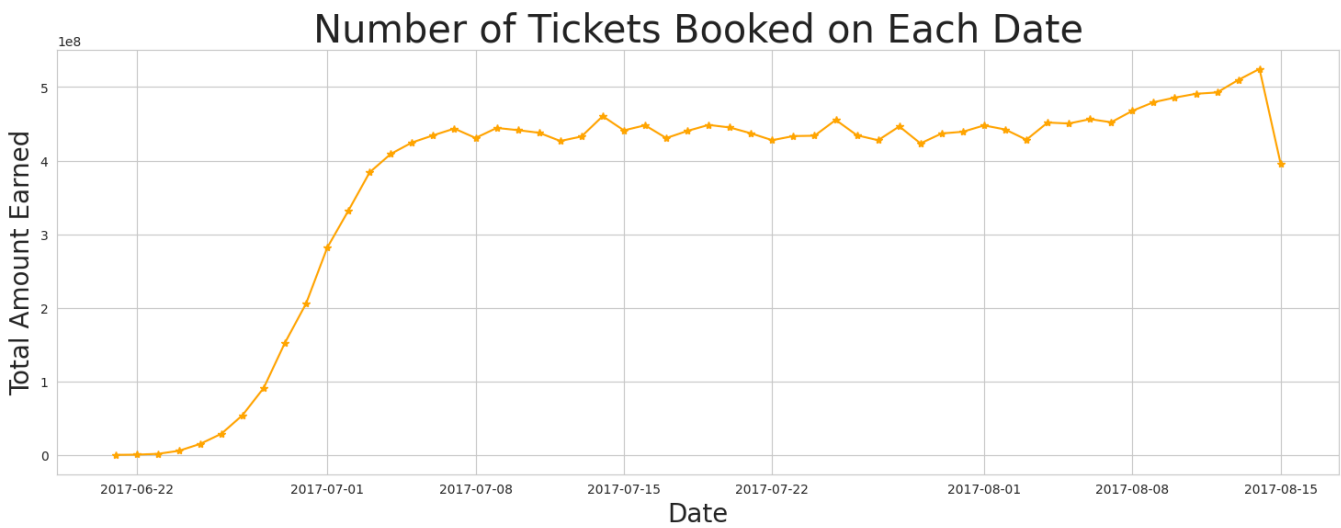
	aircraft_code	json_extract(model, '$.en')
0	773	Boeing 777-300
1	763	Boeing 767-300
2	320	Airbus A320-200
3	321	Airbus A321-200
4	319	Airbus A319-100
5	733	Boeing 737-300

Next steps: Generate code with crafts | View recommended plots | New interactive sheet

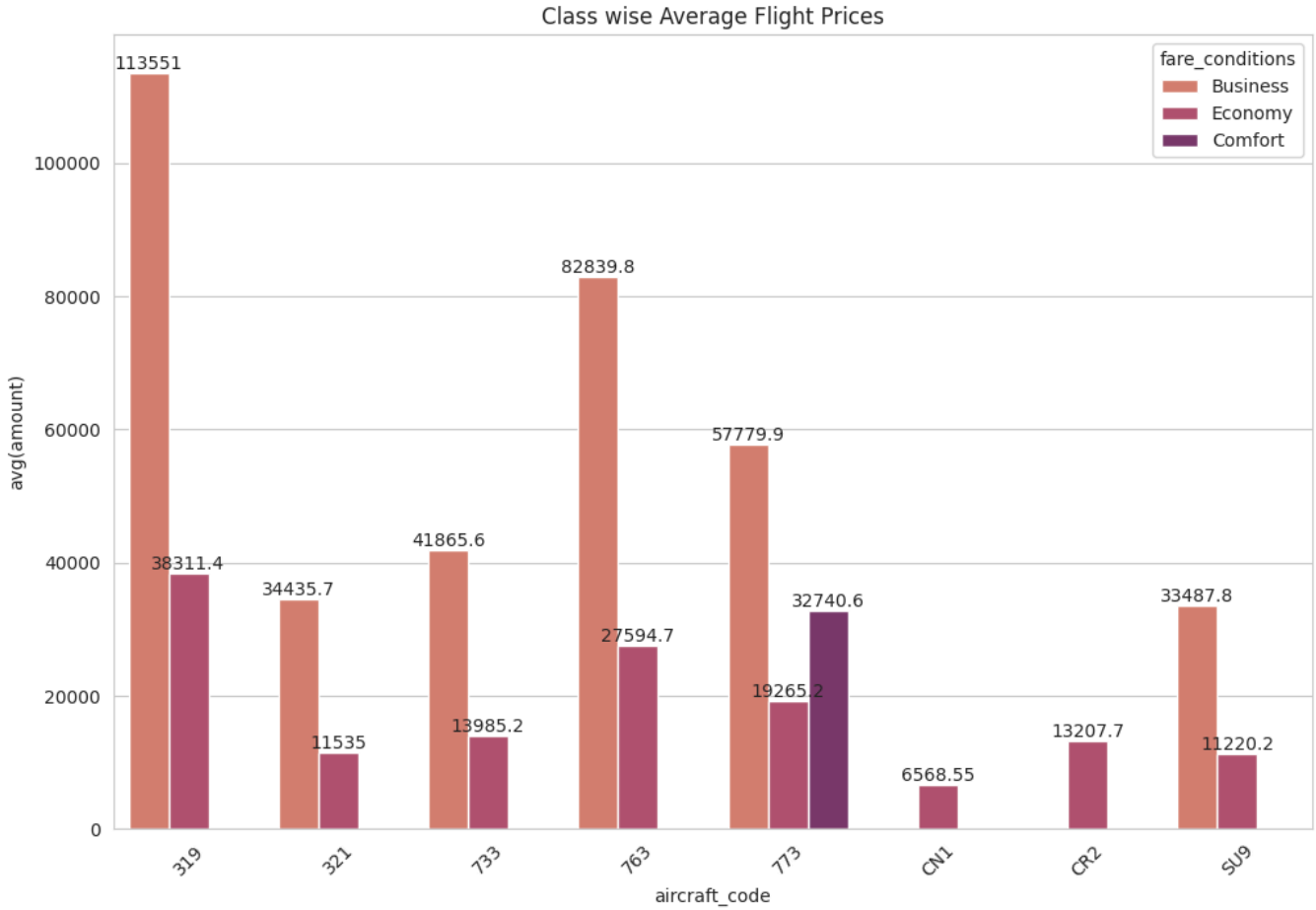
```
# Number of tickets booked and total amount earned changed with the time

tickets = pd.read_sql_query("""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
tickets_count = tickets.groupby('date')[['date']].count()
plt.figure(figsize=(18,6))
plt.plot(tickets_count.index, tickets_count['date'], color='green', scalex=True, marker = "*")
plt.title('Number of Tickets Booked on Each Date', fontsize=30)
plt.xlabel('Date', fontsize=20)
plt.ylabel('Number of Tickets', fontsize=20)
plt.grid('b')
plt.show()
```




```
bookings = pd.read_sql_query("select * from bookings", conn)

bookings['book_date'] = pd.to_datetime(bookings['book_date'])
bookings['date'] = bookings['book_date'].dt.date
booking_amount = bookings.groupby('date')[['total_amount']].sum()

plt.figure(figsize=(18,6))
plt.plot(booking_amount.index, booking_amount['total_amount'],color='orange',scalex=True, marker = '*')
plt.title('Number of Tickets Booked on Each Date', fontsize=30)
plt.xlabel('Date', fontsize=20)
plt.ylabel('Total Amount Earned', fontsize=20)
plt.grid('b')
```




```
# Fare Distribution for the Flights

df = pd.read_sql_query("""select fare_conditions, aircraft_code,avg(amount)
                        from ticket_flights join flights
                        on ticket_flights.flight_id = flights.flight_id
                        group by aircraft_code, fare_conditions""", conn)
df.to_csv('fare_avg_amount.csv')

sns.set_style('whitegrid')
fig,axes = plt.subplots(figsize=(12,8))
ax = sns.barplot(x='aircraft_code',y='avg(amount)',hue='fare_conditions', data=df, palette = 'flare')
for container in ax.containers:
    ax.bar_label(container)
plt.title('Class wise Average Flight Prices')
plt.xticks(rotation=45)
plt.show()
```




```
crafts = pd.read_sql("""SELECT aircraft_code, json_extract(model, '$.en')
                        FROM aircrafts_data
                        where aircraft_code IN (319, 321, 733, 763, 773, 'CN1', 'CR2', 'SU9');""", conn)
crafts
```

	aircraft_code	json_extract(model, '$.en')
0	773	Boeing 777-300
1	763	Boeing 767-300
2	SU9	Sukhoi Superjet-100
3	321	Airbus A321-200
4	319	Airbus A319-100
5	733	Boeing 737-300
6	CN1	Cessna 208 Caravan
7	CR2	Bombardier CRJ-200

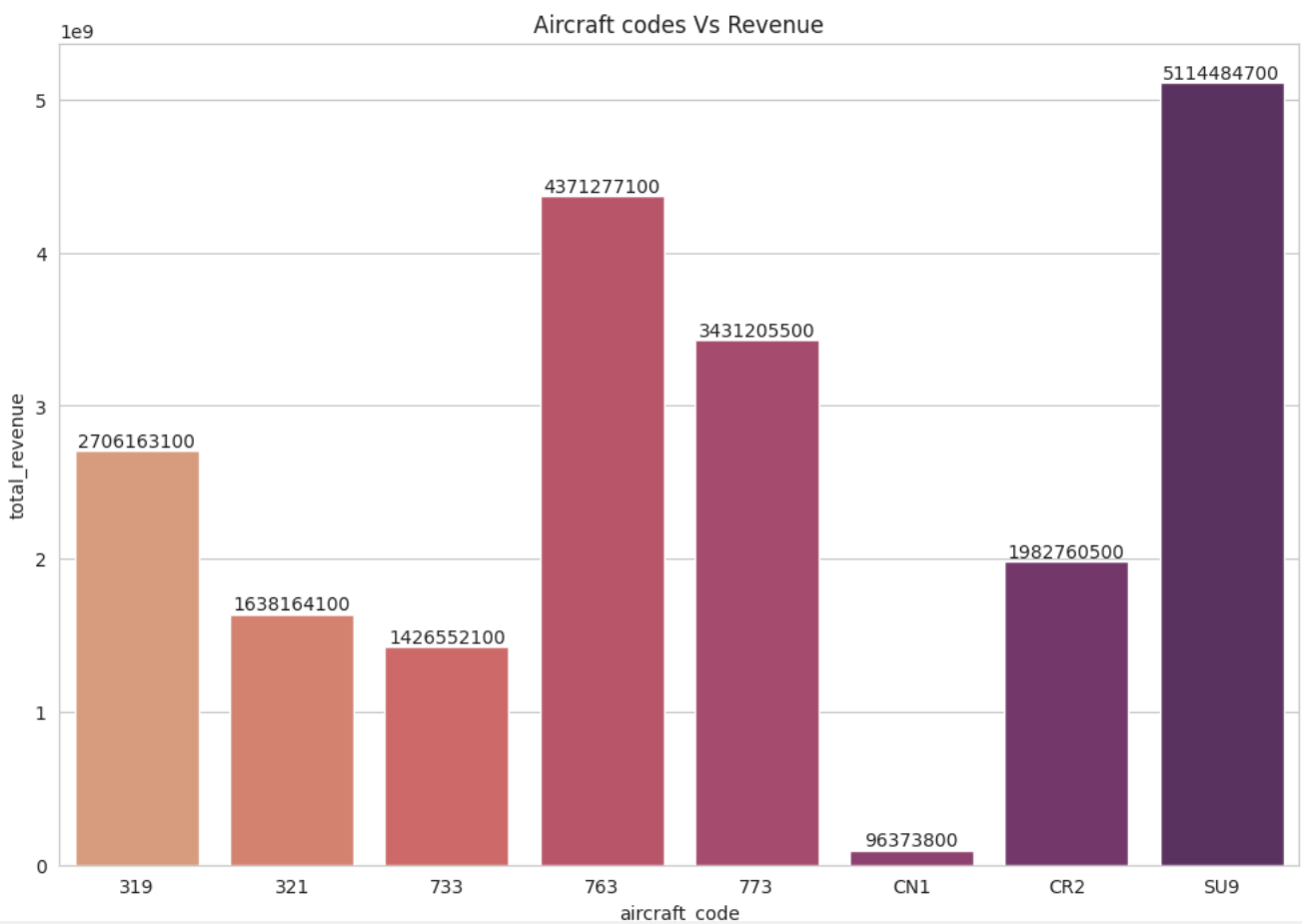
Next steps: Generate code with crafts | View recommended plots | New interactive sheet

```
# Total revenue per year and the average revenue per ticket.

revenue = pd.read_sql_query("""select aircraft_code,ticket_count,total_revenue,total_revenue/ticket_count as avg_revenue_per_ticket fror
                    (select aircraft_code, count(*) as ticket_count, sum(amount) as total_revenue from ticket_flights
                        join flights on ticket_flights.flight_id = flights.flight_id
                        group by aircraft_code)""", conn)
revenue.to_csv('revenue.csv')


df_revenue = pd.read_csv('revenue.csv')
df_revenue
```

	Unnamed: 0	aircraft_code	ticket_count	total_revenue	avg_revenue_per_ticket
0	0	319	52853	2706163100	51201
1	1	321	107129	1638164100	15291
2	2	733	86102	1426552100	16568
3	3	763	124774	4371277100	35033
4	4	773	144376	3431205500	23765
5	5	CN1	14672	96373800	6568

Next steps: Generate code with df_revenue | View recommended plots | New interactive sheet

```
# Aircraft codes vs revenue

sns.set_style('whitegrid')
fig,axes = plt.subplots(figsize=(12,8))
ax = sns.barplot(x='aircraft_code',y='total_revenue', data=df_revenue, palette = 'flare')
for container in ax.containers:
    ax.bar_label(container, fmt='%.0f')  # Format labels as integers
plt.title('Aircraft codes Vs Revenue')
plt.show()
```




```
# Calculate the average occupancy per aircraft

occupancy_rate = pd.read_sql_query("""select a.aircraft_code,avg(a.seats_count) as booked_seats, b.num_seats, avg(a.seats_count)/b.num_s
                (select aircraft_code,flights.flight_id,count(*) as seats_count 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 seats
                    group by aircraft_code) as b
                    on a.aircraft_code = b.aircraft_code group by a.aircraft_code""", conn
                  )
occupancy_rate
```