In [50]:

```
import pandas as pd
import sqlite3
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

In [100…

```
connection = sqlite3.connect('travel.sqlite')
cursor = connection.cursor()
#using sql to connect to the database
```

In [27]:

```
cursor.execute("""SELECT name FROM sqlite_master where type = 'table';""")
print('Tables present are;')
table_list = [table[0] for table in cursor.fetchall()]
table_list
```

```
Tables present are;
```

Out[27]:

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

Pursing through briefly to see the tables from the database

BRIEF OVERVIEW OF THE DATA

EXTRACTING THE TABLES FROM THE DATABASE

In [28]:

```
aircrafts = pd.read_sql_query("SELECT * FROM aircrafts_data", connection)
aircrafts
```

Out[28]:

	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

In [29]:

```
airports = pd.read_sql_query("SELECT * FROM airports_data", connection)
airports
```

Out[29]:

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

104 rows × 5 columns

In [30]:

```
boarding_passes = pd.read_sql_query("SELECT * FROM boarding_passes", connection)
boarding_passes
```

Out[30]:

	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

In [31]:

```
bookings = pd.read_sql_query("SELECT * FROM bookings", connection)
bookings
```

Out[31]:

	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

In [32]:

```
flights =pd.read_sql_query("SELECT * FROM flights",connection)
flights
```

Out[32]:

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

33121 rows × 10 columns

In [33]:

```
seats = pd.read_sql_query("SELECT * FROM seats", connection)
seats
```

Out[33]:

	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

In [34]:

```
ticket_flights = pd.read_sql_query("SELECT * FROM ticket_flights", connection)
ticket_flights
```

Out[34]:

	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

In [14]:

```
tickets = pd.read_sql_query("SELECT * FROM tickets", connection)
tickets
```

Out[14]:

	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

In [134…

```
for table in table_list:
    print('\ntable:', table)
    column_info = connection.execute("PRAGMA table_info({})".format(table))
    for column in column_info.fetchall():
        print(column[1:3])
#looping through all the tables to generate the column names
```

```
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)')
```

In [133…

```python
for table in table_list:
    print('\ntable:', table)
    df_table = pd.read_sql_query(f"SELECT * FROM {table}", connection)
    print(df_table.isnull().sum())
# checking for any null values or zeros in the tables
```

```
table: aircrafts_data
aircraft_code    0
model            0
range            0
dtype: int64

table: airports_data
airport_code    0
airport_name    0
city            0
coordinates     0
timezone        0
dtype: int64

table: boarding_passes
ticket_no      0
flight_id      0
boarding_no    0
seat_no        0
dtype: int64

table: bookings
book_ref        0
book_date       0
total_amount    0
dtype: int64

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

table: seats
aircraft_code      0
seat_no            0
fare_conditions    0
dtype: int64

table: ticket_flights
ticket_no          0
flight_id          0
fare_conditions    0
amount             0
dtype: int64

table: tickets
ticket_no       0
book_ref        0
passenger_id    0
dtype: int64
```

ANALYSIS

This will involve finding planes with the seat numbers being above 100, tickets booked, amounts earned by the airlines according to the classes of tickets bought.

How many planes have 100 seats

In [102…

```
df1 = pd.read_sql_query("""SELECT aircraft_code, count(*) as num_seats
    from seats group by aircraft_code having num_seats>100""",connection)
df1
```

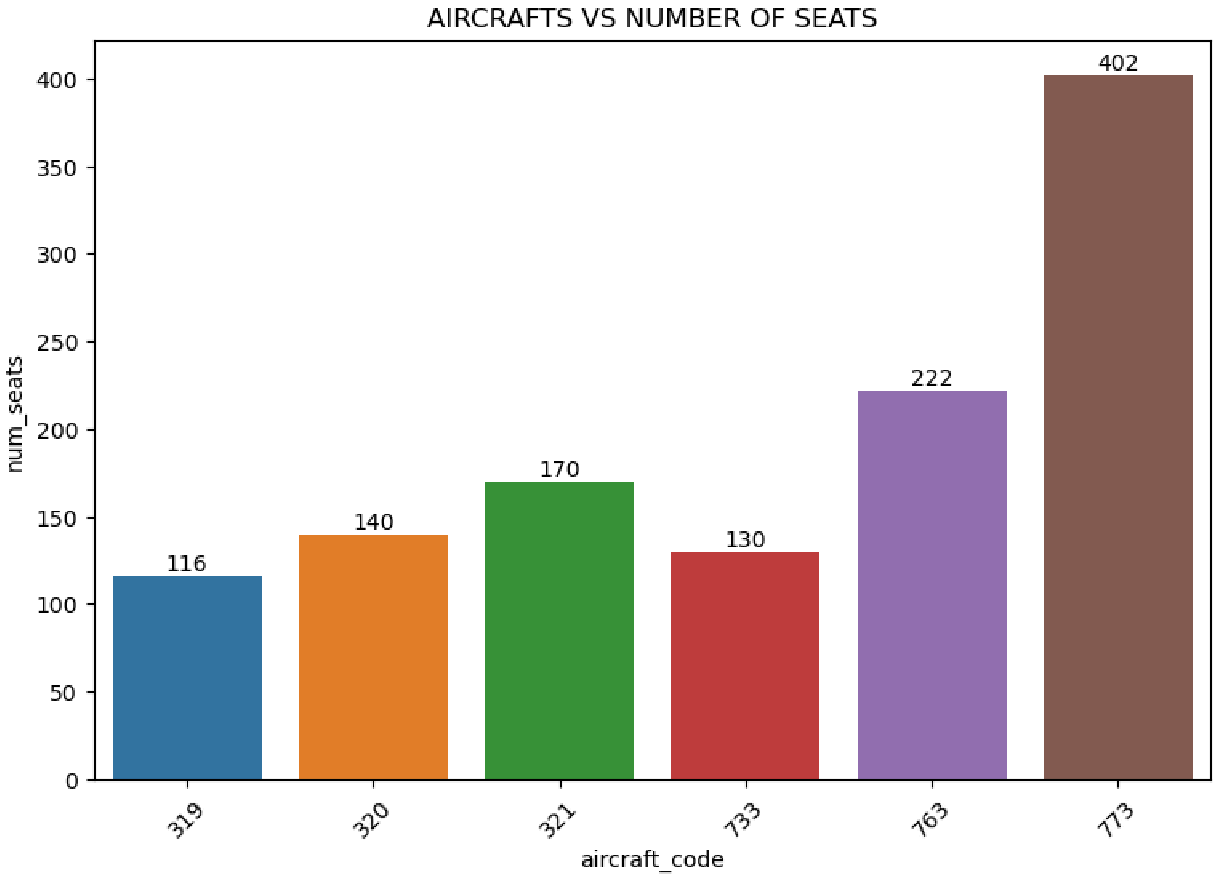
Out[102]:

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

Number of tickets booked alongside amount earned overtime

In [107…

```
fig,axes = plt.subplots(figsize=(9,6))
ax = sns.barplot(data = df1,x='aircraft_code',y='num_seats')
for container in ax.containers:
    ax.bar_label(container)
plt.title('AIRCRAFTS VS NUMBER OF SEATS')
plt.xticks(rotation = 45)
plt.show()
```



In [38]:

```
pd.read_sql_query("""SELECT * FROM tickets inner join bookings
on tickets.book_ref = bookings.book_ref""", connection)
```

Out[38]:

	ticket_no	book_ref	passenger_id	book_ref	book_date	total_amount
0	0005432000987	06B046	8149 604011	06B046	2017-07-05 20:19:00+03	12400
1	0005432000988	06B046	8499 420203	06B046	2017-07-05 20:19:00+03	12400
2	0005432000989	E170C3	1011 752484	E170C3	2017-06-29 01:55:00+03	24700
3	0005432000990	E170C3	4849 400049	E170C3	2017-06-29 01:55:00+03	24700
4	0005432000991	F313DD	6615 976589	F313DD	2017-07-03 04:37:00+03	30900
...	...	...	...	...	...	...
366728	0005435999869	D730BA	0474 690760	D730BA	2017-08-14 11:50:00+03	210600
366729	0005435999870	D730BA	6535 751108	D730BA	2017-08-14 11:50:00+03	210600
366730	0005435999871	A1AD46	1596 156448	A1AD46	2017-08-13 03:49:00+03	45900
366731	0005435999872	7B6A53	9374 822707	7B6A53	2017-08-15 15:54:00+03	219400
366732	0005435999873	7B6A53	7380 075822	7B6A53	2017-08-15 15:54:00+03	219400

366733 rows × 6 columns

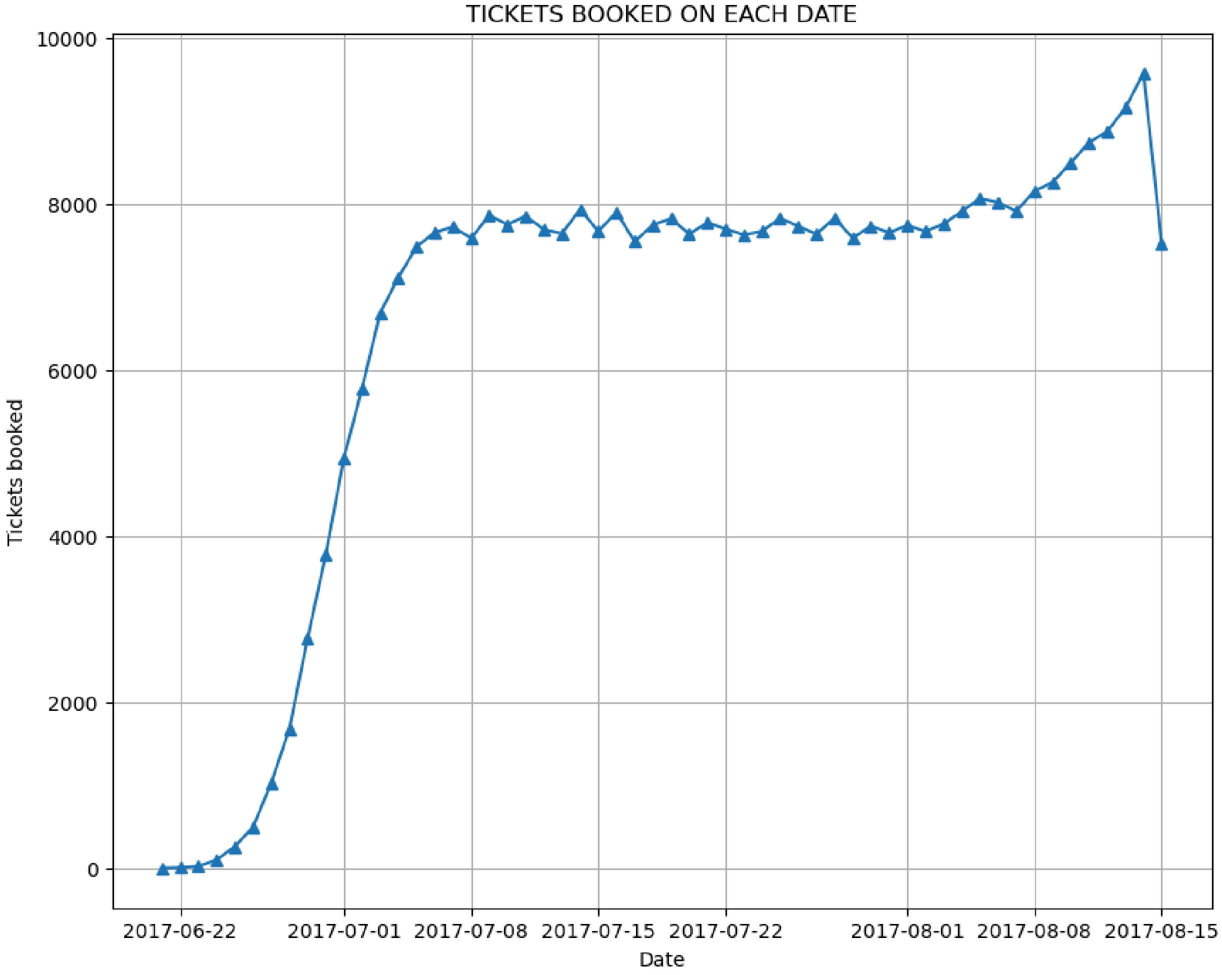
In [39]:

```
tickets = pd.read_sql_query("""SELECT * FROM tickets inner join bookings
on tickets.book_ref = bookings.book_ref""", connection)
tickets.dtypes
```

```
Out[39]: ticket_no        object
         book_ref         object
         passenger_id     object
         book_ref         object
         book_date        object
         total_amount      int64
         dtype: object
```

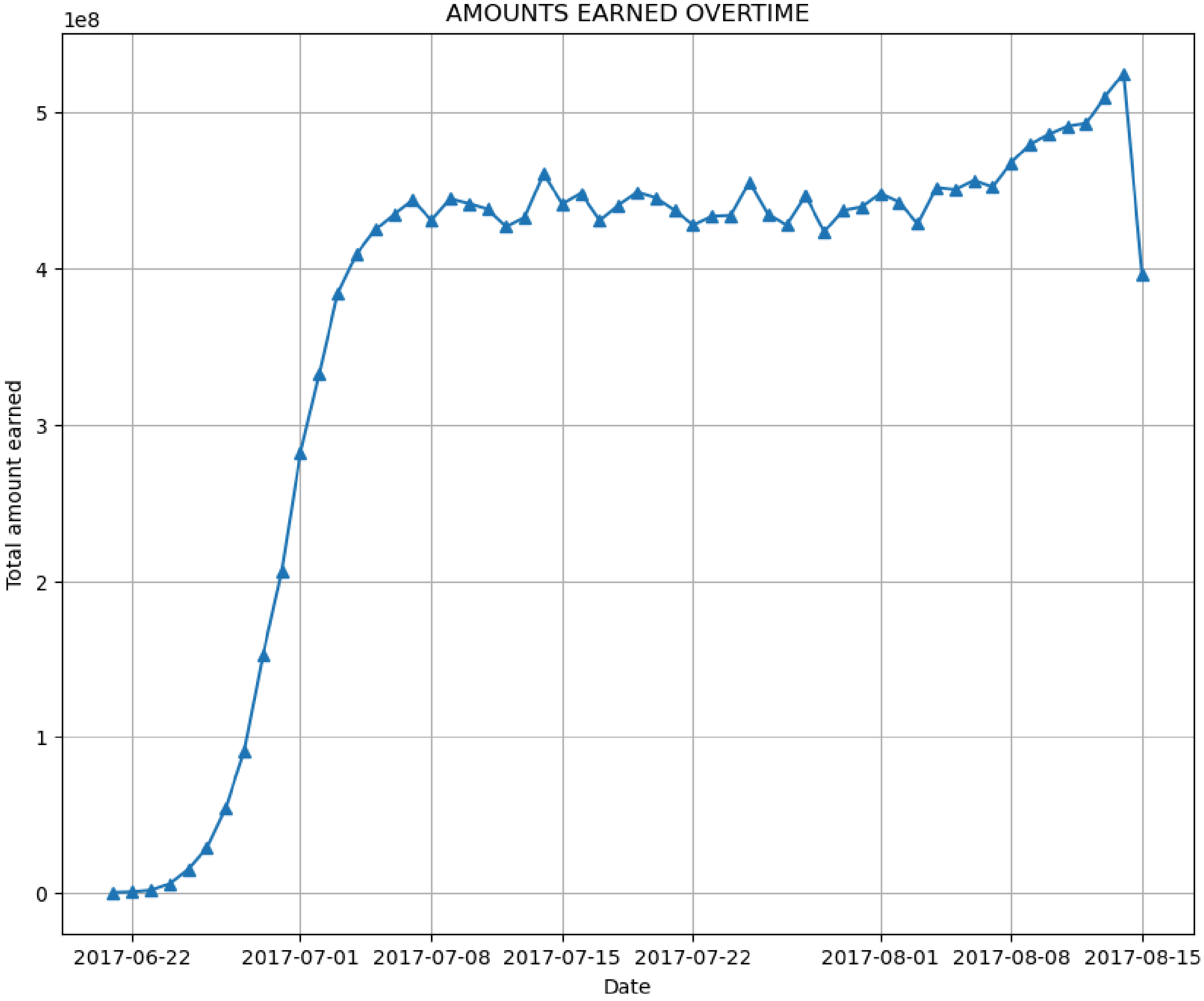
```
In [40]: tickets['book_date'] = pd.to_datetime(tickets['book_date'])
         tickets['date'] = tickets['book_date'].dt.date
```

```
In [41]: ticket_count = tickets.groupby('date')[['date']].count()
         plt.figure(figsize = (10,8))
         plt.plot(ticket_count.index, ticket_count['date'], marker = '^')
         plt.title('TICKETS BOOKED ON EACH DATE')
         plt.xlabel('Date' )
         plt.ylabel('Tickets booked')
         plt.grid('b')
         plt.show()
```



Amounts earned overtime from tickets

```
In [42]: bookings = pd.read_sql_query("SELECT * FROM bookings", connection)
         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= (10,8))
         plt.plot(booking_amount.index, booking_amount['total_amount'], marker = '^')
         plt.title('AMOUNTS EARNED OVERTIME')
         plt.xlabel('Date')
         plt.ylabel('Total amount earned')
         plt.grid('b')
         plt.show()
```



1. I utilized a line chart to visualize these trends and it shows an increase from june 22nd to july 7th on tickets booked.
2. we can as well see that there is a very close relationship/correlation between the booked tickets and amounts earned by the airlines overtime.

Average charge of each airline regardless of class

In [47]:
```
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 a
```

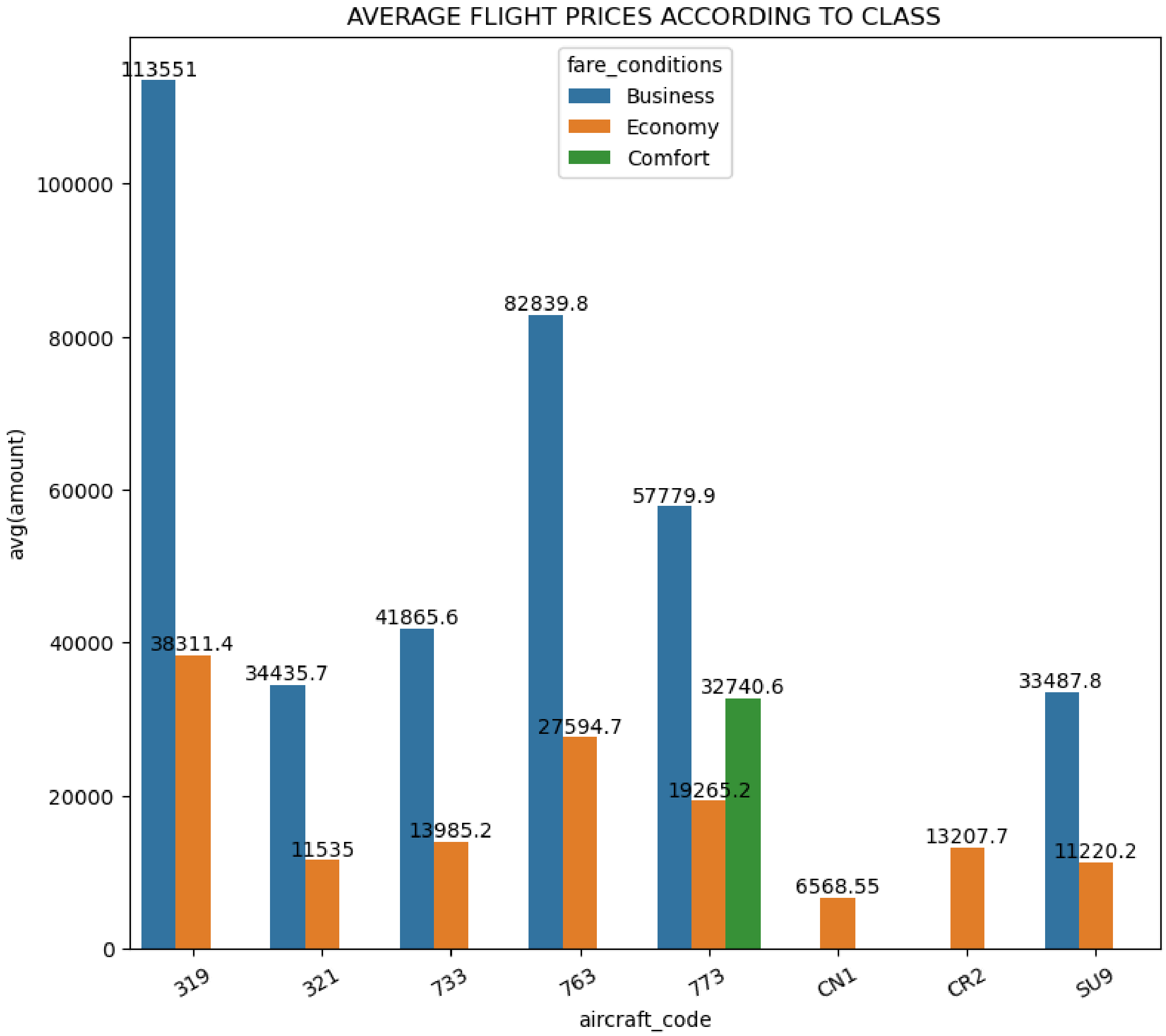
In [48]:
```
df
```

Out[48]:

	fare_conditions	aircraft_code	avg(amount)
0	Business	319	113550.557703
1	Economy	319	38311.402347
2	Business	321	34435.662664
3	Economy	321	11534.974764
4	Business	733	41865.626175
5	Economy	733	13985.152000
6	Business	763	82839.842866
7	Economy	763	27594.721829
8	Business	773	57779.909435
9	Comfort	773	32740.552889
10	Economy	773	19265.225693
11	Economy	CN1	6568.552345
12	Economy	CR2	13207.661102
13	Business	SU9	33487.849829
14	Economy	SU9	11220.183400

In [109…

```
import seaborn as sns
fig,axes = plt.subplots(figsize=(9,8))
ax = sns.barplot(data = df, x='aircraft_code', y='avg(amount)', hue = 'fare_conditions
for container in ax.containers:
    ax.bar_label(container)
plt.title('AVERAGE FLIGHT PRICES ACCORDING TO CLASS')
plt.xticks(rotation = 30)
plt.show()
```



1. The plane with aircraft code 773 seems to be the only plane with a third class;'comfort class'
2. CN1 and CR2 only have one class which is economy

Total Revenue and Occupancy Rates

Total revenue

calculating the total revenue per year and average revenue per ticket

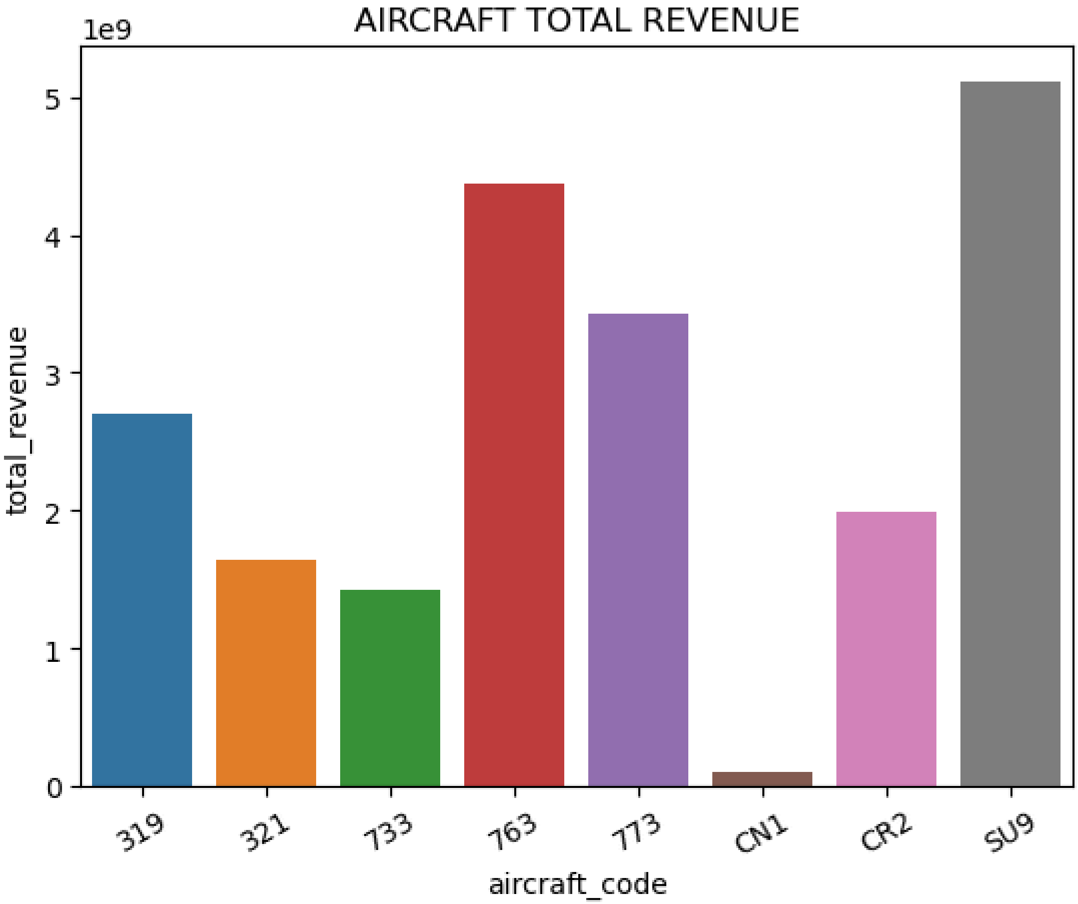
In [86]:
```
df2 =pd.read_sql_query("""SELECT aircraft_code,ticket_count, total_revenue, total_reve
(SELECT aircraft_code, count(*) as ticket_count,
sum(amount) as total_revenue from ticket_flights join flights on ticket_flights.flight
= flights.flight_id group by aircraft_code)""",connection)
df2
```

Out[86]:

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

In [89]:

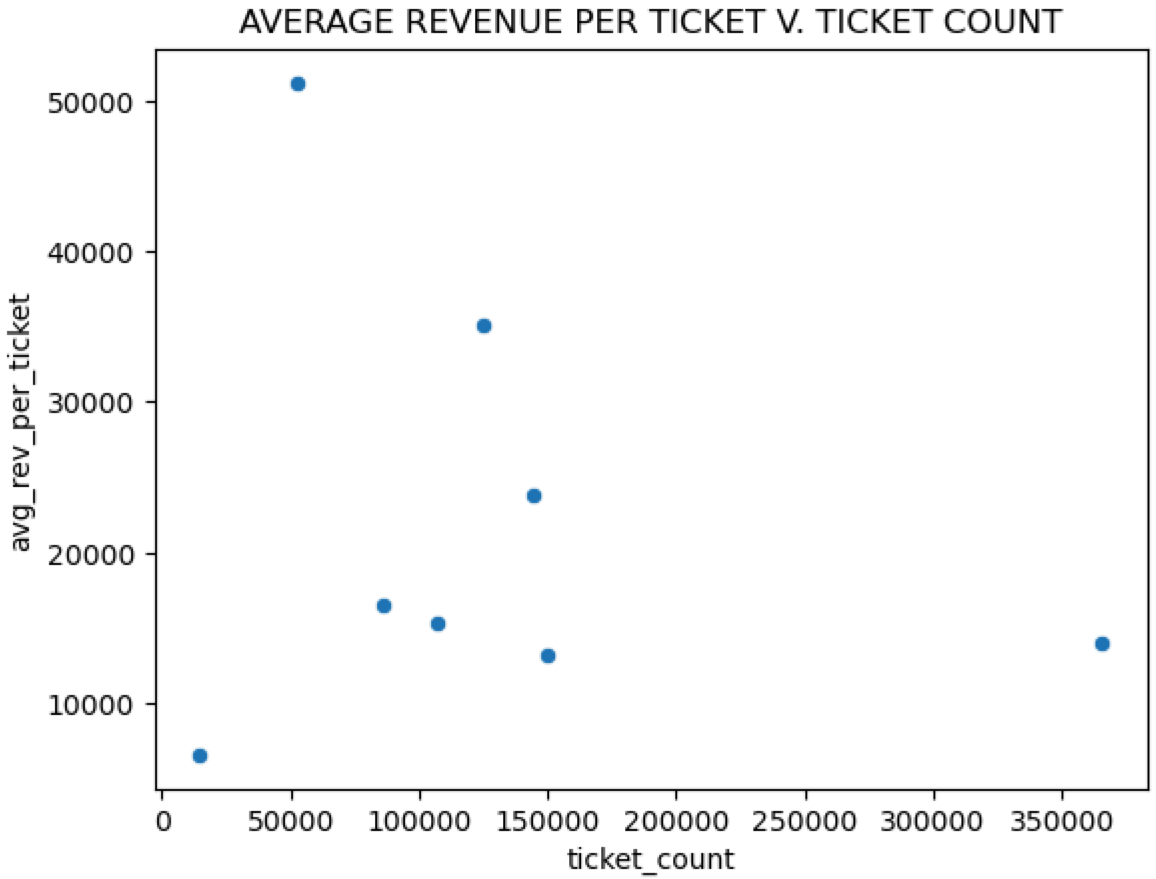
```
sns.barplot(data = df2, x ='aircraft_code',y='total_revenue')
plt.xticks(rotation=30)
plt.title('AIRCRAFT TOTAL REVENUE')
plt.show()
```



1. SU9 has the most revenue likely due to its lower ticket prices.
2. CN1 has the lowest due to its limited economy class offerings.

In [99]:

```
sns.scatterplot(data= df2,x='ticket_count', y ='avg_rev_per_ticket')
plt.title('AVERAGE REVENUE PER TICKET V. TICKET COUNT')
plt.show()
```



negtive correlation between ticket count and average revenue per ticket meaning as more tickets are sold, the revenue per ticket reduces showing that higher sales are potentially driven by lower prices.

Calculating average occcupancy rate in each aircraft

In [117…

```
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_seats as occupancy_rate from
   (SELECT aircraft_code, flights.flight_id, count(*) as seats_count from boarding_pas
   inner join flights on boarding_passes.flight_id = flights.flight_id group by aircra
   as a inner join(SELECT aircraft_code, count(*) as num_seats from seats group by air
   as b on a.aircraft_code = b.aircraft_code group by a.aircraft_code""", connection)
occupancy_rate
```

Out[117]:

	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

total annual revenue that could increase by aircrafts having higher occupancy rates

In [118…

```
occupancy_rate['Increased occupancy rate'] = occupancy_rate['occupancy_rate']+occupanc
occupancy_rate
```

Out[118]:

	aircraft_code	booked_seats	num_seats	occupancy_rate	Increased occupancy rate
0	319	53.58318098720292	116	0.46192397402761143	0.5081163714303726
1	321	88.80923076923077	170	0.5224072398190045	0.574647963800905
2	733	80.25546218487395	130	0.617349709114415	0.6790846800258565
3	763	113.93729372937294	222	0.5132310528350132	0.5645541581185146
4	773	264.9258064516129	402	0.659019419033863	0.7249213609372492
5	CN1	6.004431314623338	12	0.5003692762186115	0.5504062038404727
6	CR2	21.48284690220174	50	0.42965693804403476	0.4726226318484382
7	SU9	56.81211267605634	97	0.5856918832583128	0.644261071584144

In [119…

```
pd.set_option("display.float_format",str)
```

In [120…

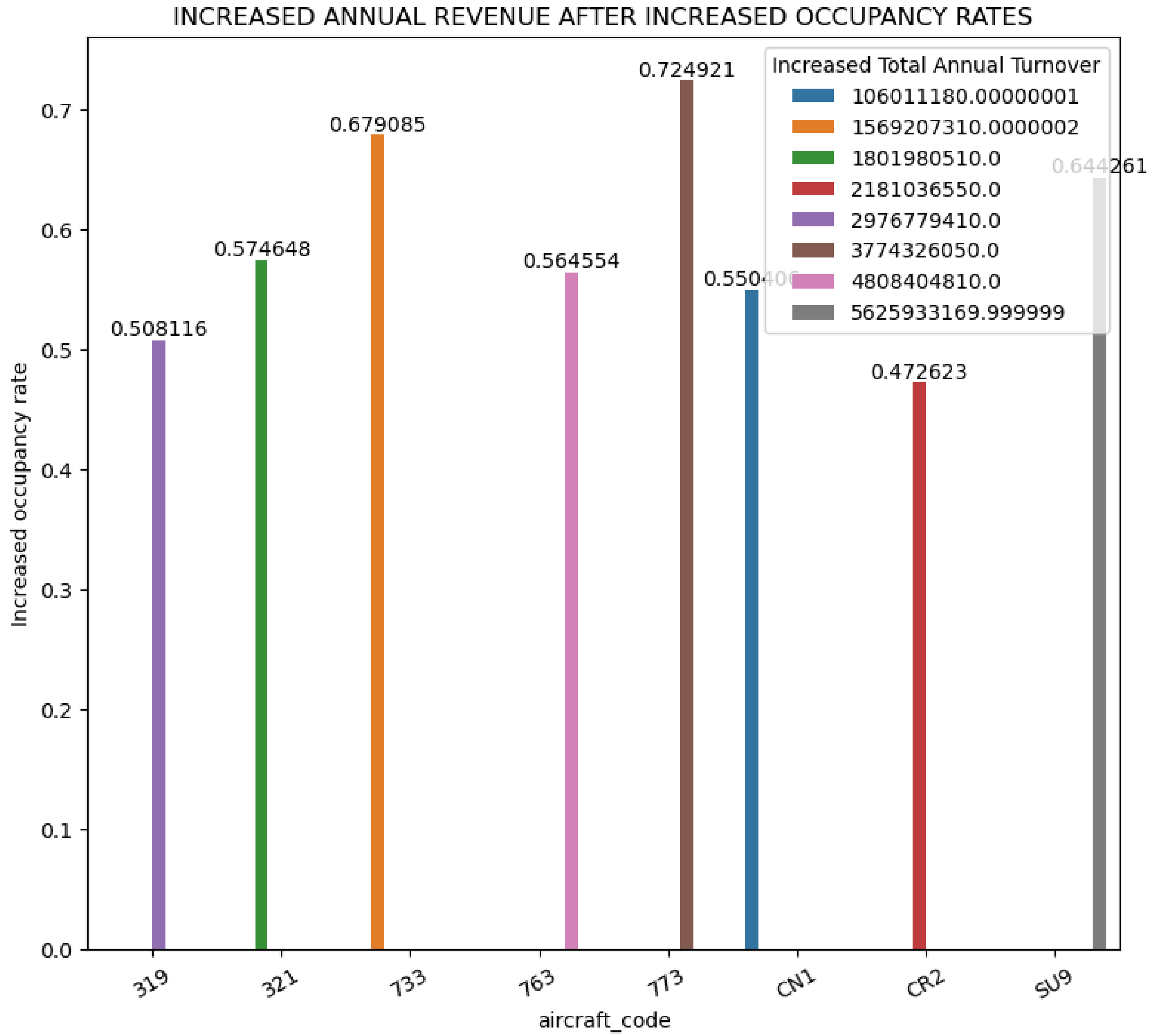
```
total_revenue = pd.read_sql_query("""SELECT aircraft_code, sum(amount) as total_revenu
flights on ticket_flights.flight_id = flights.flight_id group by aircraft_code""",conn
occupancy_rate['Increased Total Annual Turnover'] = (total_revenue['total_revenue']/oc
occupancy_rate
```

Out[120]:

	aircraft_code	booked_seats	num_seats	occupancy_rate	Increased occupancy rate	Increa Annual
0	319	53.58318098720292	116	0.46192397402761143	0.5081163714303726	297
1	321	88.80923076923077	170	0.5224072398190045	0.574647963800905	180
2	733	80.25546218487395	130	0.617349709114415	0.6790846800258565	156920731
3	763	113.93729372937294	222	0.5132310528350132	0.5645541581185146	480
4	773	264.9258064516129	402	0.659019419033863	0.7249213609372492	377
5	CN1	6.004431314623338	12	0.5003692762186115	0.5504062038404727	106011180
6	CR2	21.48284690220174	50	0.42965693804403476	0.4726226318484382	218
7	SU9	56.81211267605634	97	0.5856918832583128	0.644261071584144	56259331

In [131…

```
fig,axes = plt.subplots(figsize=(9,8))
ax = sns.barplot(data = occupancy_rate, x=('aircraft_code'), y=('Increased occupancy r
for container in ax.containers:
    ax.bar_label(container)
plt.title('INCREASED ANNUAL REVENUE AFTER INCREASED OCCUPANCY RATES')
plt.xticks(rotation = 30)
plt.show()
```



TO CONCLUDE, ACCORDING TO THE DATA, AIRLINES COULD;

1. OFFER MORE CLASSES AT COMPETITIVE PRICES TO BOOST SALES AS WE CAN SEE FROM AIRCRAFT 773 THAT HAS A THIRD OFFERING('COMFORT) THAT DOES AS WELL AS AIRCRAFT SU9'S BUSINESS CLASS OFFERINGS
2. DUE TO THE NEGEATIVE CORRELATION BETWEEN TICKET COUNT AND REVENUE, THE COMPANIES COULD AS WELL LOOK INTO PRICING STRATEGIES TO FIND BALANCES BETWEEN TICKET COUNT AND REVENUES FOR EXAMPLE VALUE BASED PRICING, BUNDLES ETC
3. INCREASE OCCUPANCY RATES THAT OFFSET VACANT SEATS AS WELL AS LEAD TO INCREASED PROFITABILITY.
4. STRIKE A BALANCE BETWEEN CONSIDERING CONSUMER SATISFACTION AS WELL AS IMPLEMENTING THESE STRATEGIES

In [ ]: