

```
In [2]: import sqlite3
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
In [3]: connection = sqlite3.connect("Airline_data.sqlite")
cursor = connection.cursor()
print("Connected to the database")
```

Connected to the database

```
In [4]: 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[4]: ['aircrafts_data',
 'airports_data',
 'boarding_passes',
 'bookings',
 'flights',
 'seats',
 'ticket_flights',
 'tickets']
```

Data Exploration

```
In [7]: # --example

aircrafts_data = pd.read_sql_query("select * from aircrafts_data", connection)
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	5700
4	321	{"en": "Airbus A321-200", "ru": "Аэробус A321-..."} 5600	5600
5	319	{"en": "Airbus A319-100", "ru": "Аэробус A319-..."} 6700	6700
6	733	{"en": "Boeing 737-300", "ru": "Боинг 737-300"}	4200
7	CN1	{"en": "Cessna 208 Caravan", "ru": "Сессна 208..."} 1200	1200
8	CR2	{"en": "Bombardier CRJ-200", "ru": "Бомбардье ..."} 2700	2700

```
In [6]: airports_data = pd.read_sql_query("select * from airports_data", connection)
airports_data
```

Out[6]:	airport_code	airport_name	city	coordinates
0	YKS	{"en": "Yakutsk Airport", "ru": "Якутск"} "Yakutsk Airport", "ru": "Якутск"}	{"en": "Yakutsk", "ru": "Якутск"} {"en": "Yakutsk", "ru": "Якутск"}	(129.77099609375, 62.0932998657226562)
1	MJZ	{"en": "Mirny Airport", "ru": "Мирный"} "Mirny Airport", "ru": "Мирный"}	{"en": "Mirnyj", "ru": "Мирный"} {"en": "Mirnyj", "ru": "Мирный"}	(114.03900146484375, 62.534698486328125)
2	KHV	{"en": "Khabarovsk-Novy Airport", "ru": "Хабаровск-Новый Аэропорт"} "Khabarovsk-Novy Airport", "ru": "Хабаровск-Новый Аэропорт"}	{"en": "Khabarovsk", "ru": "Хабаровск"} {"en": "Khabarovsk", "ru": "Хабаровск"}	(135.18800354004, 48.527999877930001) Asia
3	PKC	{"en": "Yelizovo Airport", "ru": "Елизово"} "Yelizovo Airport", "ru": "Елизово"}	{"en": "Petropavlovsk", "ru": "Петропавловск-Камчатский"} "Petropavlovsk", "ru": "Петропавловск-Камчатский"}	(158.453994750976562, 53.1679000854492188) Asi...
4	UUS	{"en": "Yuzhno-Sakhalinsk Airport", "ru": "Южно-Сахалинск"} "Yuzhno-Sakhalinsk Airport", "ru": "Южно-Сахалинск"}	{"en": "Yuzhno-Sakhalinsk", "ru": "Южно-Сахалинск"} "Yuzhno-Sakhalinsk", "ru": "Южно-Сахалинск"}	(142.718002319335938, 46.8886985778808594) ,
...
99	MMK	{"en": "Murmansk Airport", "ru": "Мурманск"} "Murmansk Airport", "ru": "Мурманск"}	{"en": "Murmansk", "ru": "Мурманск"} "Murmansk", "ru": "Мурманск"}	(32.7508010864257812, 68.7817001342773438) Eur...
100	ABA	{"en": "Abakan Airport", "ru": "Абакан"} "Abakan Airport", "ru": "Абакан"}	{"en": "Abakan", "ru": "Абакан"} "Abakan", "ru": "Абакан"}	(91.3850021362304688, 53.7400016784667969) Asia
101	BAX	{"en": "Barnaul Airport", "ru": "Барнаул"} "Barnaul Airport", "ru": "Барнаул"}	{"en": "Barnaul", "ru": "Барнаул"} "Barnaul", "ru": "Барнаул"}	(83.5384979248046875, 53.363800048828125) Asia
102	AAQ	{"en": "Anapa Vityazevo Airport", "ru": "Витязево"} "Anapa Vityazevo Airport", "ru": "Витязево"}	{"en": "Anapa", "ru": "Анапа"} "Anapa", "ru": "Анапа"}	(37.3473014831539984, 45.002101898192997) Eur...
103	CNN	{"en": "Chulman Airport", "ru": "Чульман"} "Chulman Airport", "ru": "Чульман"}	{"en": "Neryungri", "ru": "Нерюнгри"} "Neryungri", "ru": "Нерюнгри"}	(124.914001464839998, 56.9138984680179973)

104 rows × 5 columns

In [8]: `boarding_passes = pd.read_sql_query("select * from boarding_passes", connection)`
`boarding_passes`

Out[8]:

	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 [9]:

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

Out[9]:

	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	FFFFEF3	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	FFFAA8	2017-08-08 04:45:00+03	28800
262787	FFFFF7	2017-07-01 22:12:00+03	73600

262788 rows × 3 columns

In [10]:

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

Out[10]:

	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	BT
1	3979	PG0052	2017-08-25 14:50:00+03	2017-08-25 17:35:00+03	VKO	HM
2	4739	PG0561	2017-09-05 12:30:00+03	2017-09-05 14:15:00+03	VKO	AE
3	5502	PG0529	2017-09-12 09:50:00+03	2017-09-12 11:20:00+03	SVO	UF
4	6938	PG0461	2017-09-04 12:25:00+03	2017-09-04 13:20:00+03	SVO	UL
...
33116	33117	PG0063	2017-08-02 19:25:00+03	2017-08-02 20:10:00+03	SKX	SV
33117	33118	PG0063	2017-07-28 19:25:00+03	2017-07-28 20:10:00+03	SKX	SV
33118	33119	PG0063	2017-09-08 19:25:00+03	2017-09-08 20:10:00+03	SKX	SV
33119	33120	PG0063	2017-08-01 19:25:00+03	2017-08-01 20:10:00+03	SKX	SV
33120	33121	PG0063	2017-08-26 19:25:00+03	2017-08-26 20:10:00+03	SKX	SV

33121 rows × 10 columns

In [11]:

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

Out[11]:

	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 [12]: ticket_flights = pd.read_sql_query("select * from ticket_flights", connection)
ticket_flights
```

Out[12]:

	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 [13]: tickets = pd.read_sql_query("select * from ticket_", connection)
tickets
```

Out[13]:

	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 [18]: #Example for which type column available in table
```

```
for table in table_list:
    print('\n




```

```

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 [20]: #Example -- For checking

```

for table in table_list:
    print('\n




```

```
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
```

Basic analysis

How many planes have more than 100 seats?

```
In [21]: pd.read_sql_query("""select aircraft_code, count(*) as num_seats from seats
group by aircraft_code having num_seats > 100""", connection)
```

Out[21]:

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

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

```
In [38]: tickets = pd.read_sql_query("""Select * from tickets inner join bookings
on tickets.book_ref = bookings.book_ref""", connection)

tickets.dtypes #for plotting the graph we need to know data types of table.

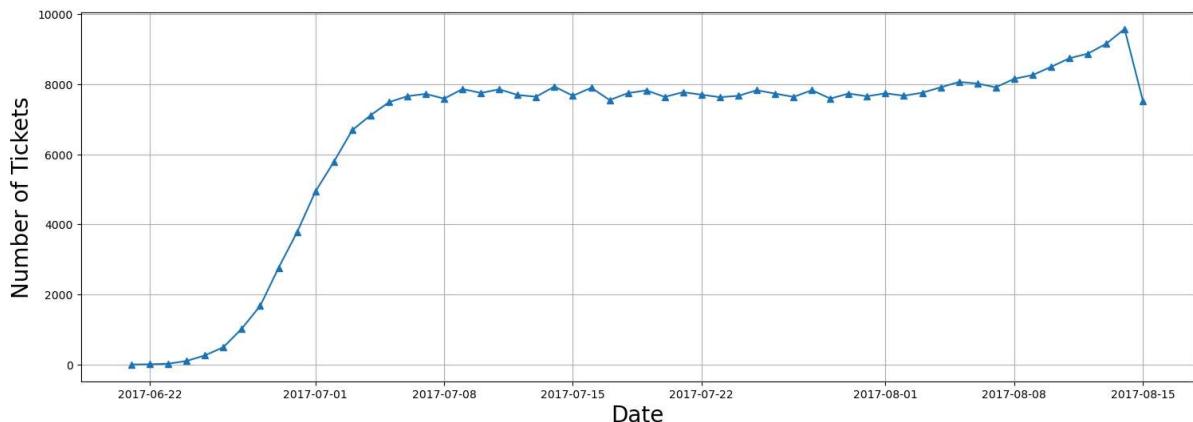
tickets['book_date'] = pd.to_datetime(tickets['book_date'])
tickets.dtypes #checking

#For using groupby we need to remove time from the date

tickets['date'] = tickets['book_date'].dt.date #Return date
```

```
In [48]: x = tickets.groupby("date")["date"].count()
```

```
plt.figure(figsize=(18,6))
plt.plot(x.index, x, marker = '^')
plt.xlabel('Date', fontsize = 20)
plt.ylabel('Number of Tickets', fontsize = 20 )
plt.grid('b')
plt.show()
```



```
In [23]: bookings = pd.read_sql_query("""Select * from bookings""", connection)
```

```
bookings['book_date'] = pd.to_datetime(bookings['book_date'])
bookings.dtypes #checking

bookings['date'] = bookings['book_date'].dt.date #Return date

bookings
```

Out[23]:

	book_ref	book_date	total_amount	date
0	00000F	2017-07-05 03:12:00+03:00	265700	2017-07-05
1	000012	2017-07-14 09:02:00+03:00	37900	2017-07-14
2	000068	2017-08-15 14:27:00+03:00	18100	2017-08-15
3	000181	2017-08-10 13:28:00+03:00	131800	2017-08-10
4	0002D8	2017-08-07 21:40:00+03:00	23600	2017-08-07
...
262783	FFFEF3	2017-07-17 07:23:00+03:00	56000	2017-07-17
262784	FFFF2C	2017-08-08 05:55:00+03:00	10800	2017-08-08
262785	FFFF43	2017-07-20 20:42:00+03:00	78500	2017-07-20
262786	FFFA8	2017-08-08 04:45:00+03:00	28800	2017-08-08
262787	FFFFF7	2017-07-01 22:12:00+03:00	73600	2017-07-01

262788 rows × 4 columns

In [54]:

```
x = bookings.groupby('date')[['total_amount']].sum()  
x
```

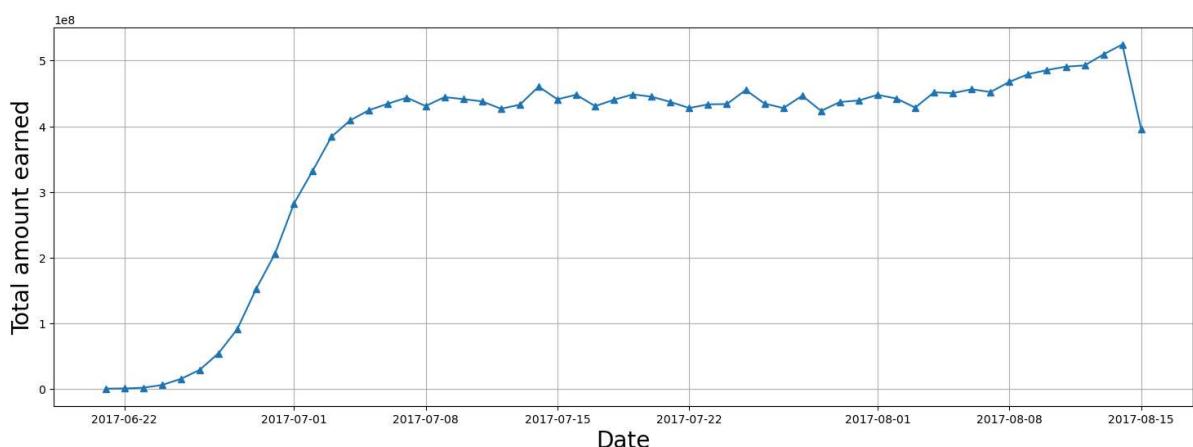
Out[54]:

date	total_amount
2017-06-21	441900
2017-06-22	775300
2017-06-23	1822000
2017-06-24	5977000
2017-06-25	15305400
2017-06-26	29049100
2017-06-27	54339900
2017-06-28	91256400
2017-06-29	152484000
2017-06-30	205910000
2017-07-01	281864700
2017-07-02	332231100
2017-07-03	384108400
2017-07-04	409175900
2017-07-05	424680000
2017-07-06	434393100
2017-07-07	443575300
2017-07-08	430814500
2017-07-09	444432000
2017-07-10	441434900
2017-07-11	437876000
2017-07-12	426721800
2017-07-13	432826700
2017-07-14	460449000
2017-07-15	441120100
2017-07-16	448014600
2017-07-17	430686400
2017-07-18	440378400
2017-07-19	448547900
2017-07-20	445062300
2017-07-21	437084500
2017-07-22	427789700
2017-07-23	433387400
2017-07-24	433862000
2017-07-25	455336100

total_amount

date	total_amount
2017-07-26	434484900
2017-07-27	427743700
2017-07-28	446519200
2017-07-29	423439400
2017-07-30	436956700
2017-07-31	439255100
2017-08-01	447861400
2017-08-02	442331900
2017-08-03	428452000
2017-08-04	451766500
2017-08-05	450449600
2017-08-06	456387300
2017-08-07	452158300
2017-08-08	467666600
2017-08-09	479377800
2017-08-10	485604400
2017-08-11	490795600
2017-08-12	492638500
2017-08-13	509633600
2017-08-14	524600500
2017-08-15	395644100

```
In [58]: plt.figure(figsize=(18,6))
plt.plot(x.index, x, marker = '^')
plt.xlabel('Date', fontsize = 20)
plt.ylabel('Total amount earned', fontsize = 20 )
plt.grid('b')
plt.show()
```

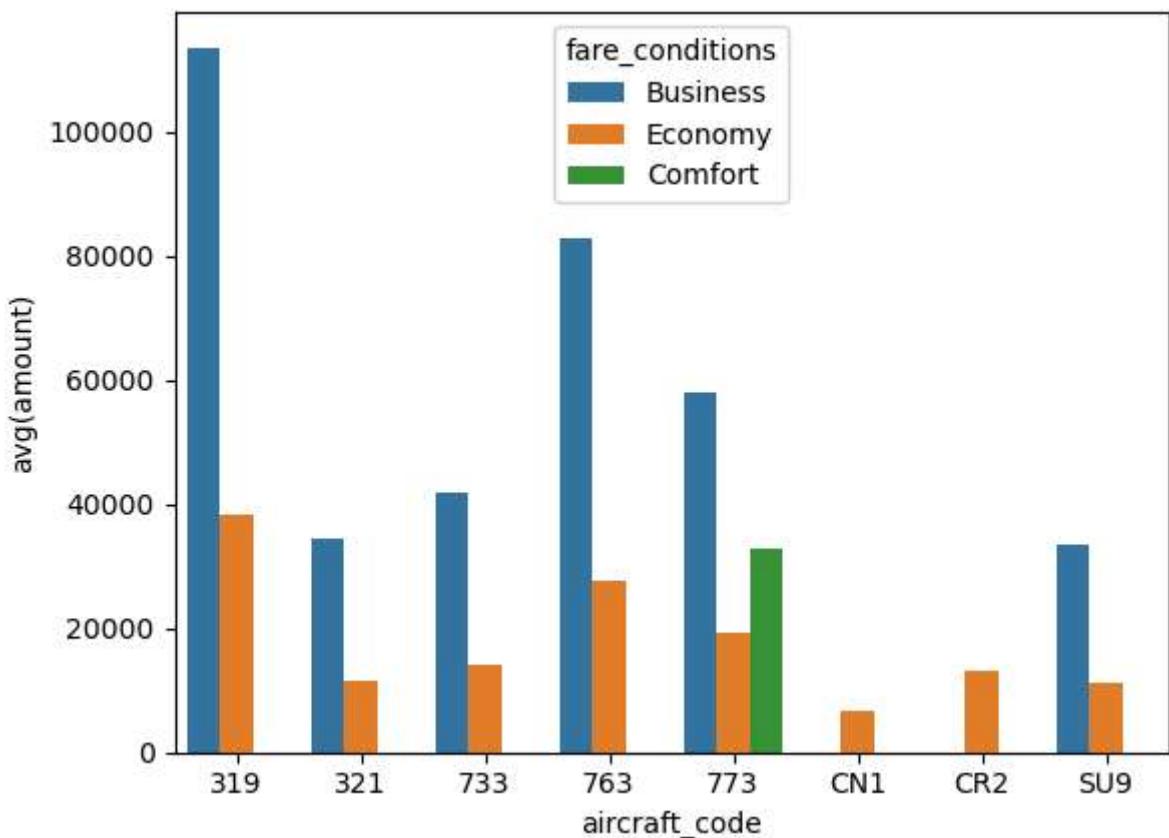


Calculate the average charges for each aircraft with different fare conditions.

```
In [63]: df = pd.read_sql_query("""select fare_conditions, aircraft_code, avg(amount) from t
join flights on ticket_flights.flight_id = flights.flight_i
group by aircraft_code, fare_conditions""", connection)
```

```
In [64]: sns.barplot(data = df, x = 'aircraft_code', y = 'avg(amount)', hue = 'fare_conditi
```

```
Out[64]: <Axes: xlabel='aircraft_code', ylabel='avg(amount)'>
```



Analyzing occupancy rate

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

```
In [67]: pd.read_sql_query("""select aircraft_code, ticket_count, total_revenue, total_rever
(select aircraft_code, count(*) as ticket_count, sum(amount) as
join flights on ticket_flights.flight_id = flights.flight_id
```

Out[67]:

	aircraft_code	ticket_count	total_revenue	avg_revenue_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

Calculate the average occupancy per aircraft.

In [15]:

```
occupancy_rate = pd.read_sql_query("""select a.aircraft_code, avg(a.seats_count) as
from
(select aircraft_code, flights.flight_id, count(*) as seats_c
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
on a.aircraft_code = b.aircraft_code group by a.aircraft_code
occupancy_rate
```

Out[15]:

	aircraft_code	booked_seats	num_seats	occupancy_rate
0	319	53.583181	116	0.461924
1	321	88.809231	170	0.522407
2	733	80.255462	130	0.617350
3	763	113.937294	222	0.513231
4	773	264.925806	402	0.659019
5	CN1	6.004431	12	0.500369
6	CR2	21.482847	50	0.429657
7	SU9	56.812113	97	0.585692

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

In [18]:

```
occupancy_rate["Inc occupancy rate"] = occupancy_rate['occupancy_rate'] + occupancy_rate
```

Out[18]:

	aircraft_code	booked_seats	num_seats	occupancy_rate	Inc occupancy rate
0	319	53.583181	116	0.461924	0.508116
1	321	88.809231	170	0.522407	0.574648
2	733	80.255462	130	0.617350	0.679085
3	763	113.937294	222	0.513231	0.564554
4	773	264.925806	402	0.659019	0.724921
5	CN1	6.004431	12	0.500369	0.550406
6	CR2	21.482847	50	0.429657	0.472623
7	SU9	56.812113	97	0.585692	0.644261

In [20]: `pd.set_option("display.float_format", str)`

In [21]: `total_revenue = pd.read_sql_query("""select aircraft_code, sum(amount) as total_rev
join flights on ticket_flights.ticket_id = flights.flight_id
occupancy_rate['Inc Total Annual Turnover'] = (total_revenue['total_revenue']/occupy
occupancy_rate`

Out[21]:

	aircraft_code	booked_seats	num_seats	occupancy_rate	Inc occupancy rate	Inc
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	156920
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	106011
6	CR2	21.48284690220174	50	0.42965693804403476	0.4726226318484382	
7	SU9	56.81211267605634	97	0.5856918832583128	0.644261071584144	56259

In []: