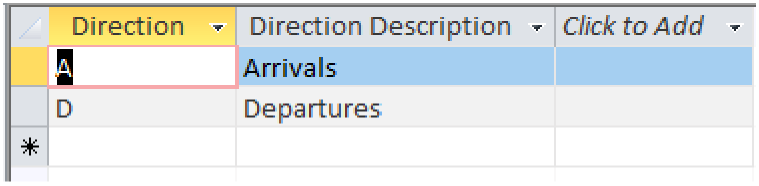
Databases Assignment

Part 1

Question 2

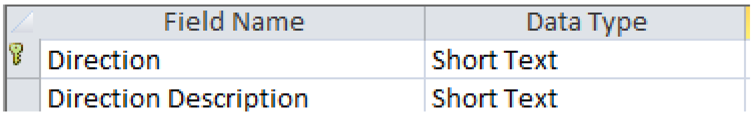
Directions Table

a. Design mode



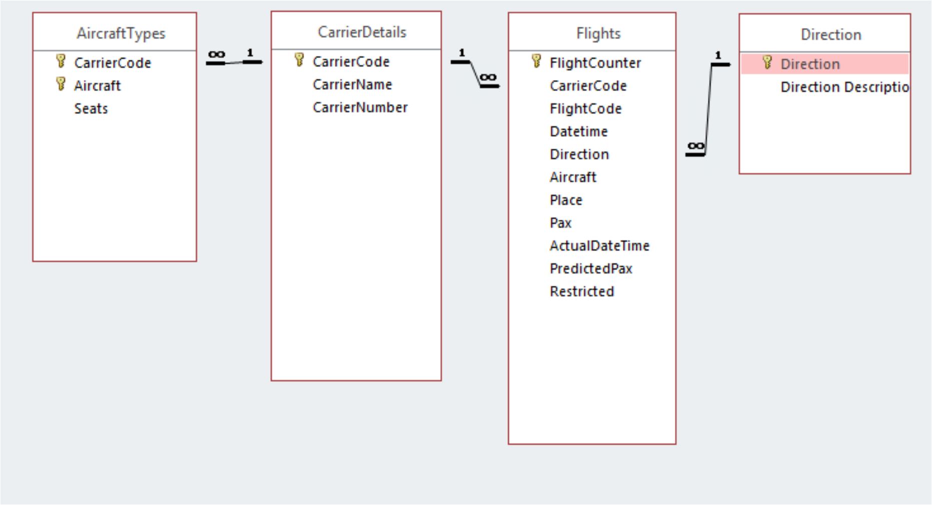
Direction	Direction Description	*Click to Add*
A	Arrivals	
D	Departures	

b. Database mode



Field Name	Data Type
Direction	Short Text
Direction Description	Short Text

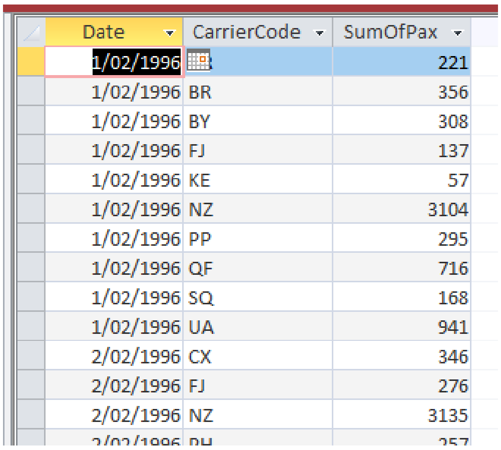
c. New Relationship



Question 3

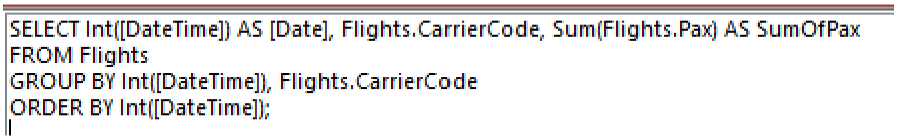
AirlineDailyTotals

d. Datasheet View



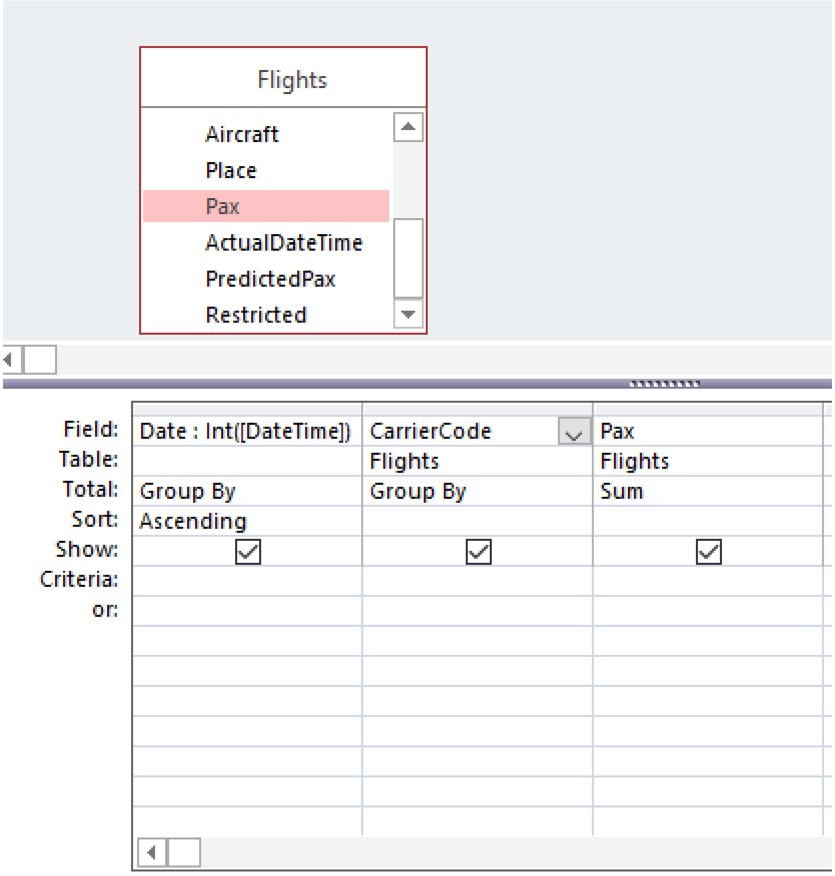
Date	CarrierCode	SumOfPax
1/02/1996	[illegible]	221
1/02/1996	BR	356
1/02/1996	BY	308
1/02/1996	FJ	137
1/02/1996	KE	57
1/02/1996	NZ	3104
1/02/1996	PP	295
1/02/1996	QF	716
1/02/1996	SQ	168
1/02/1996	UA	941
2/02/1996	CX	346
2/02/1996	FJ	276
2/02/1996	NZ	3135
2/02/1996	PH	257

e. SQL View



```
SELECT Int([DateTime]) AS [Date], Flights.CarrierCode, Sum(Flights.Pax) AS SumOfPax
FROM Flights
GROUP BY Int([DateTime]), Flights.CarrierCode
ORDER BY Int([DateTime]);
```

f. Design View

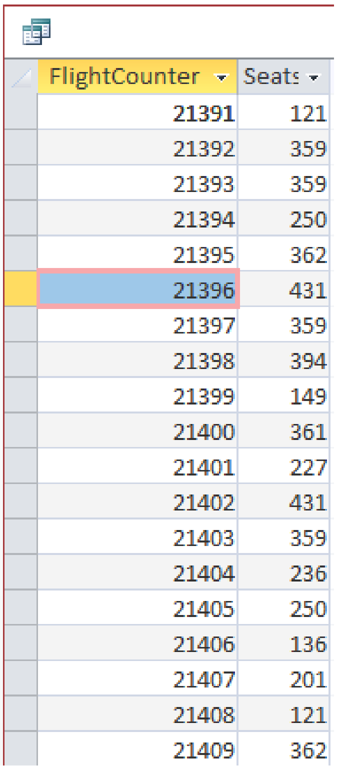


Field:	Date : Int([DateTime])	CarrierCode	Pax
Table:		Flights	Flights
Total:	Group By	Group By	Sum
Sort:	Ascending		
Show:	☑	☑	☑
Criteria:			
or:			

Question 5

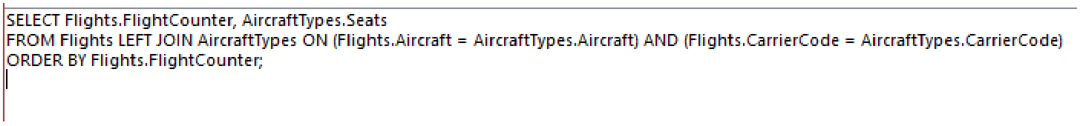
FlightSeats

g. Database view



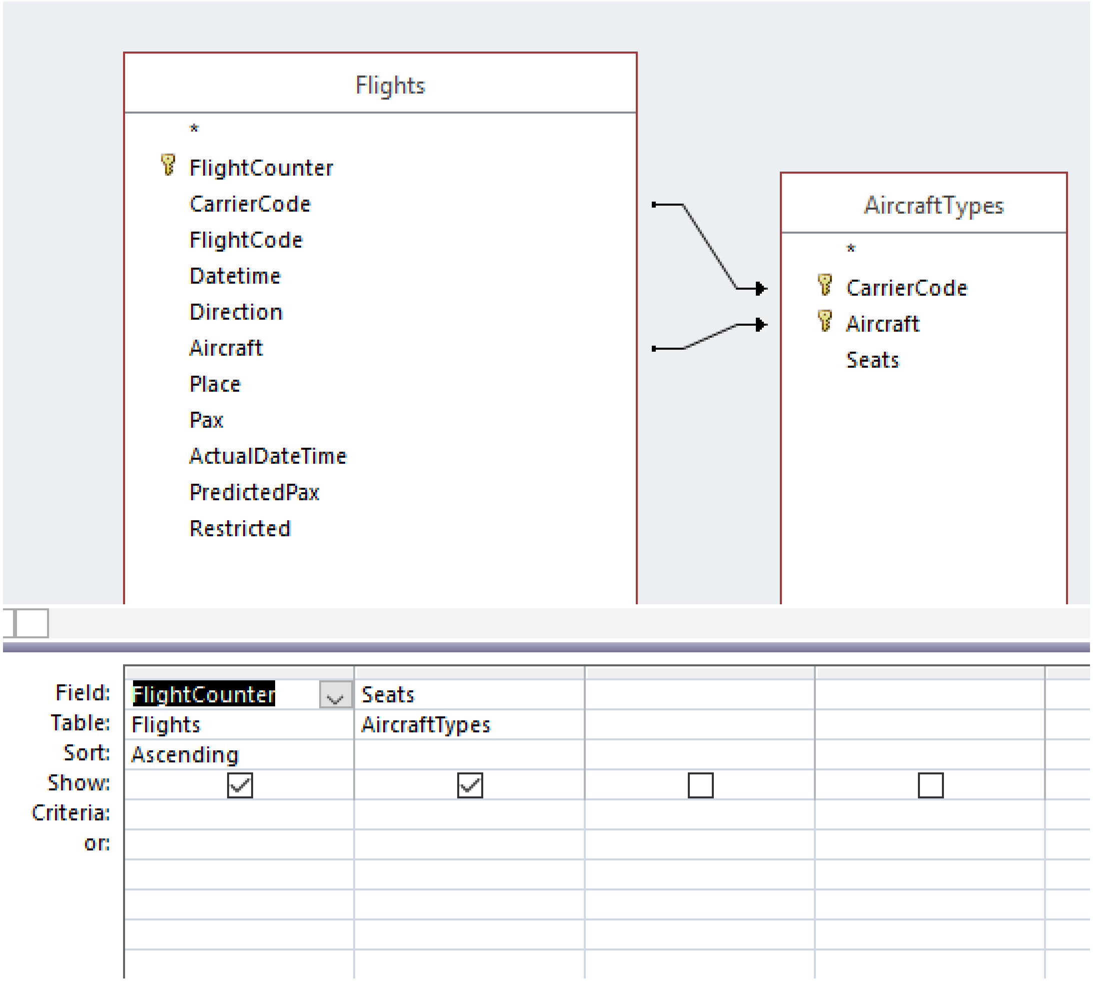
FlightCounter	Seats
21391	121
21392	359
21393	359
21394	250
21395	362
21396	431
21397	359
21398	394
21399	149
21400	361
21401	227
21402	431
21403	359
21404	236
21405	250
21406	136
21407	201
21408	121
21409	362

h. SQL view



```
SELECT Flights.FlightCounter, AircraftTypes.Seats
FROM Flights LEFT JOIN AircraftTypes ON (Flights.Aircraft = AircraftTypes.Aircraft) AND (Flights.CarrierCode = AircraftTypes.CarrierCode)
ORDER BY Flights.FlightCounter;
```

i. Design view

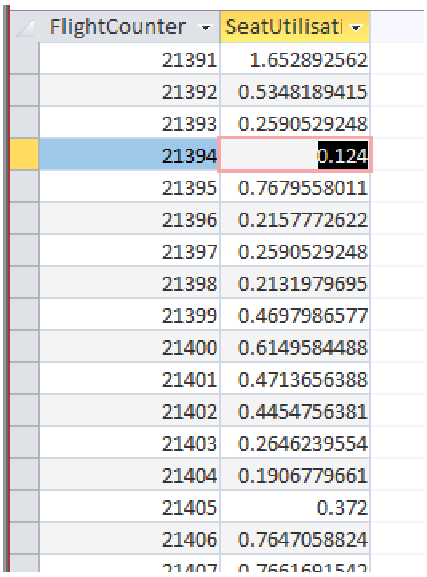


Field:	FlightCounter	Seats		
Table:	Flights	AircraftTypes		
Sort:	Ascending			
Show:	☑	☑	□	□
Criteria:				
or:				

Question 6

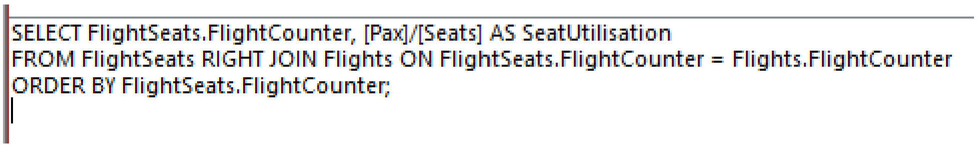
FlightSeatUtilisation

j. Database view



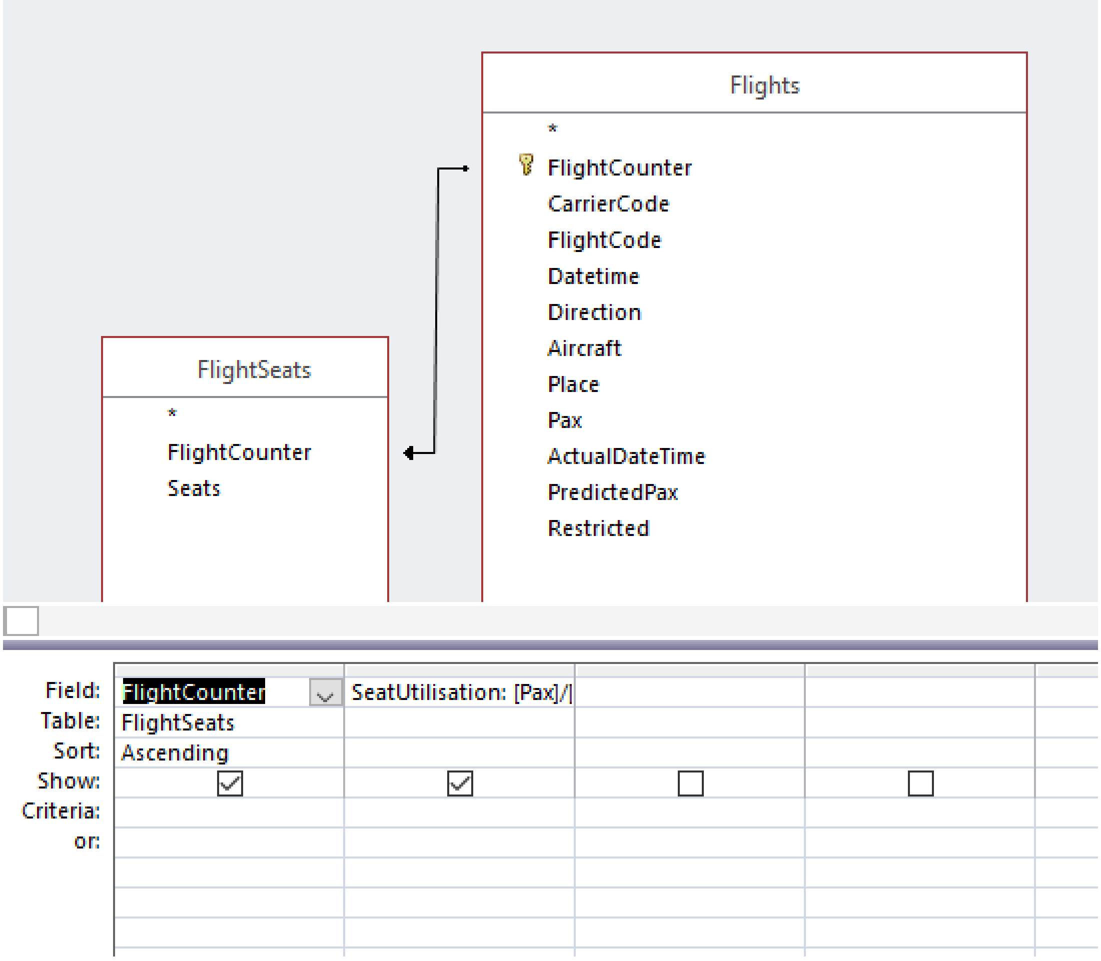
FlightCounter	SeatUtilisati
21391	1.652892562
21392	0.5348189415
21393	0.2590529248
21394	0.124
21395	0.7679558011
21396	0.2157772622
21397	0.2590529248
21398	0.2131979695
21399	0.4697986577
21400	0.6149584488
21401	0.4713656388
21402	0.4454756381
21403	0.2646239554
21404	0.1906779661
21405	0.372
21406	0.7647058824
21407	0.7661691542

k. SQL view



```
SELECT FlightSeats.FlightCounter, [Pax]/[Seats] AS SeatUtilisation
FROM FlightSeats RIGHT JOIN Flights ON FlightSeats.FlightCounter = Flights.FlightCounter
ORDER BY FlightSeats.FlightCounter;
```

l. Design view

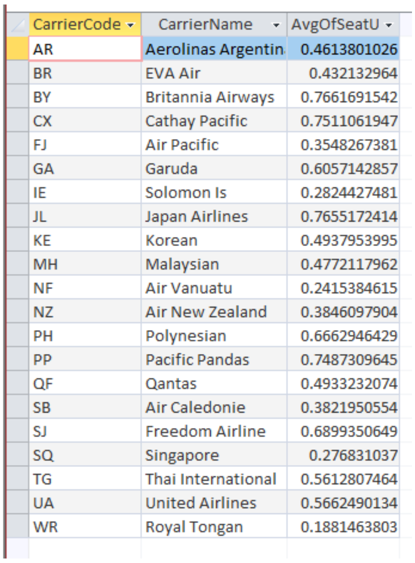


Field:	FlightCounter	SeatUtilisation: [Pax]/[		
Table:	FlightSeats			
Sort:	Ascending			
Show:	☑	☑	□	□
Criteria:				
or:				

Question 7

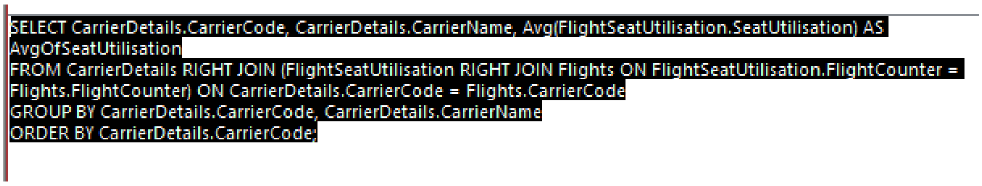
CarrierUtilisation

m. Database view



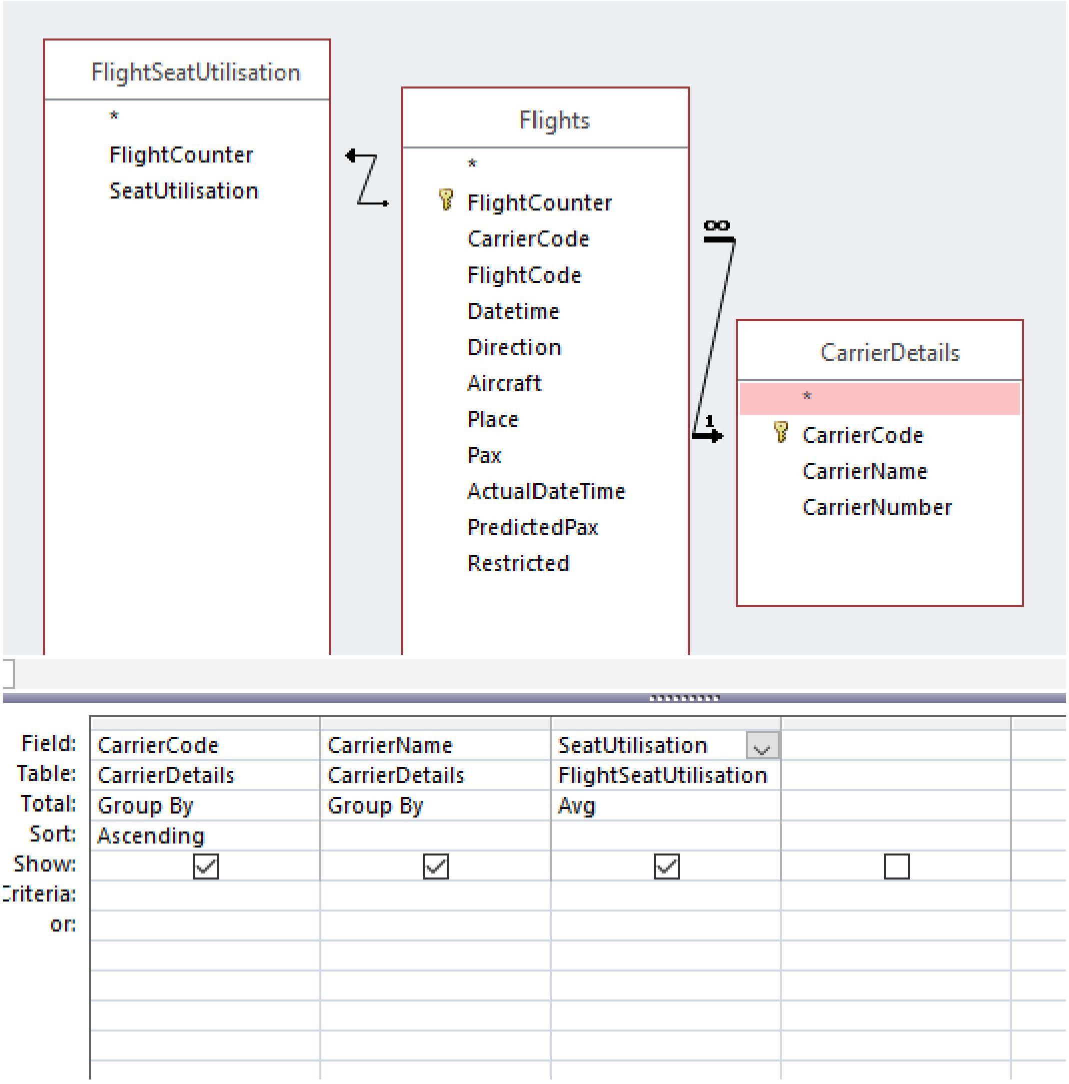
CarrierCode	CarrierName	AvgOfSeatU
AR	Aerolinas Argentin	0.4613801026
BR	EVA Air	0.432132964
BY	Britannia Airways	0.7661691542
CX	Cathay Pacific	0.7511061947
FJ	Air Pacific	0.3548267381
GA	Garuda	0.6057142857
IE	Solomon Is	0.2824427481
JL	Japan Airlines	0.7655172414
KE	Korean	0.4937953995
MH	Malaysian	0.4772117962
NF	Air Vanuatu	0.2415384615
NZ	Air New Zealand	0.3846097904
PH	Polynesian	0.6662946429
PP	Pacific Pandas	0.7487309645
QF	Qantas	0.4933232074
SB	Air Caledonie	0.3821950554
SJ	Freedom Airline	0.6899350649
SQ	Singapore	0.276831037
TG	Thai International	0.5612807464
UA	United Airlines	0.5662490134
WR	Royal Tongan	0.1881463803

n. SQL view



```
SELECT CarrierDetails.CarrierCode, CarrierDetails.CarrierName, Avg(FlightSeatUtilisation.SeatUtilisation) AS
AvgOfSeatUtilisation
FROM CarrierDetails RIGHT JOIN (FlightSeatUtilisation RIGHT JOIN Flights ON FlightSeatUtilisation.FlightCounter =
Flights.FlightCounter) ON CarrierDetails.CarrierCode = Flights.CarrierCode
GROUP BY CarrierDetails.CarrierCode, CarrierDetails.CarrierName
ORDER BY CarrierDetails.CarrierCode;
```

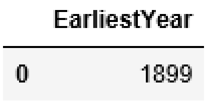
o. Design view



Field:	CarrierCode	CarrierName	SeatUtilisation	
Table:	CarrierDetails	CarrierDetails	FlightSeatUtilisation	
Total:	Group By	Group By	Avg	
Sort:	Ascending			
Show:	☑	☑	☑	□
riteria:				
or:				

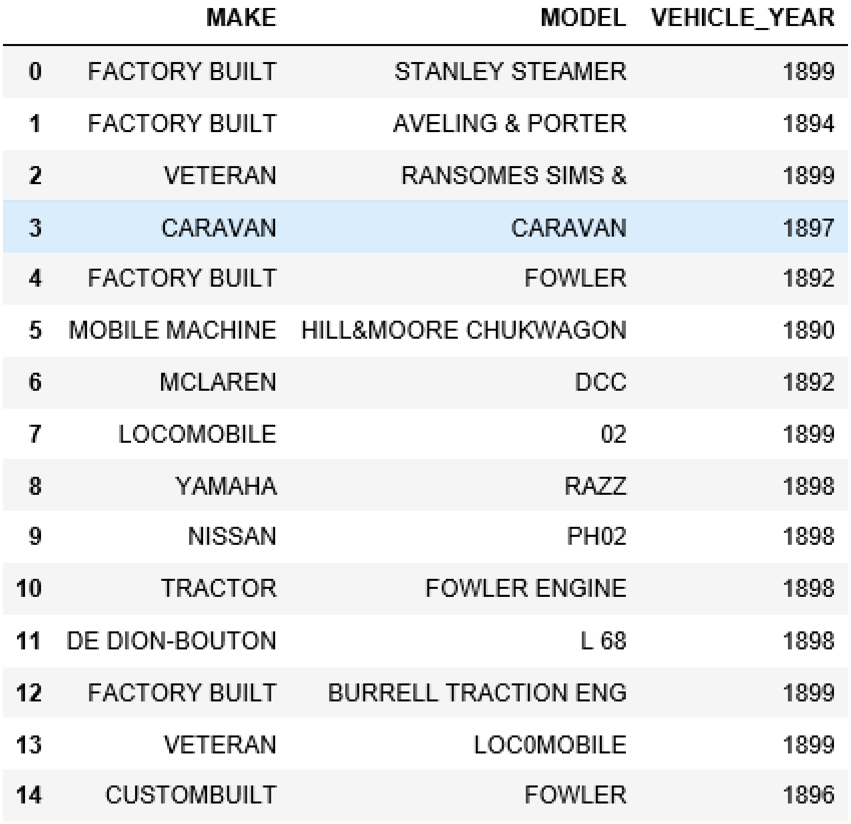
Part 2

1. # Earliest year of first registration for using the minimum function
   dataframe = pandas.read_sql_query('SELECT MIN(FIRST_NZ_REGISTRATION_YEAR) AS EarliestYear FROM Fleet ', connection)
   dataframe



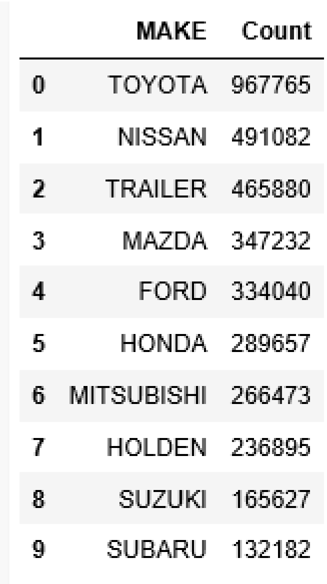
	EarliestYear
0	1899

2.
```
# Make, model and vehicle year of all the cars with a vehicle year earlier than 1900?
dataframe = pandas.read_sql_query('SELECT MAKE, MODEL, VEHICLE_YEAR  FROM Fleet WHERE VEHICLE_YEAR <1900 ', connection)
dataframe
```



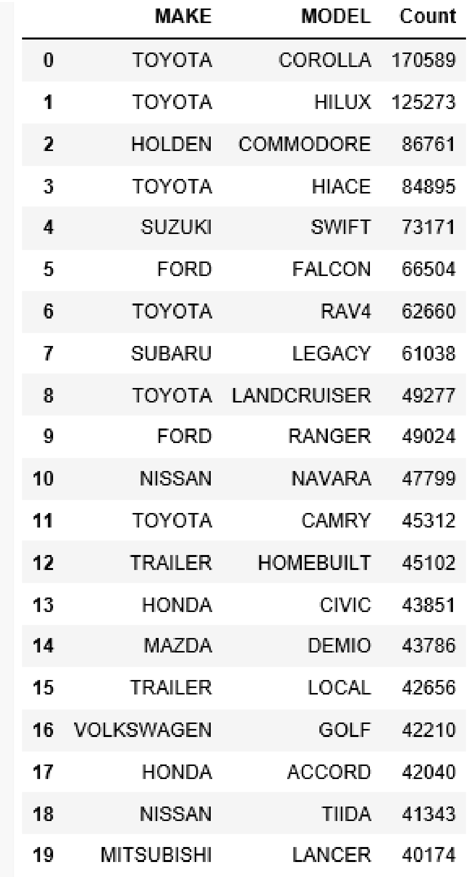
	MAKE	MODEL	VEHICLE_YEAR
0	FACTORY BUILT	STANLEY STEAMER	1899
1	FACTORY BUILT	AVELING & PORTER	1894
2	VETERAN	RANSOMES SIMS &	1899
3	CARAVAN	CARAVAN	1897
4	FACTORY BUILT	FOWLER	1892
5	MOBILE MACHINE	HILL&MOORE CHUKWAGON	1890
6	MCLAREN	DCC	1892
7	LOCOMOBILE	02	1899
8	YAMAHA	RAZZ	1898
9	NISSAN	PH02	1898
10	TRACTOR	FOWLER ENGINE	1898
11	DE DION-BOUTON	L 68	1898
12	FACTORY BUILT	BURRELL TRACTION ENG	1899
13	VETERAN	LOC0MOBILE	1899
14	CUSTOMBUILT	FOWLER	1896

3.
```
# What are the 10 most popular (by count) car makes, and the counts of these?
dataframe = pandas.read_sql_query('SELECT MAKE, COUNT(MAKE) AS Count FROM Fleet GROUP BY MAKE ORDER BY Count DESC LIMIT 10', connection)
dataframe
```



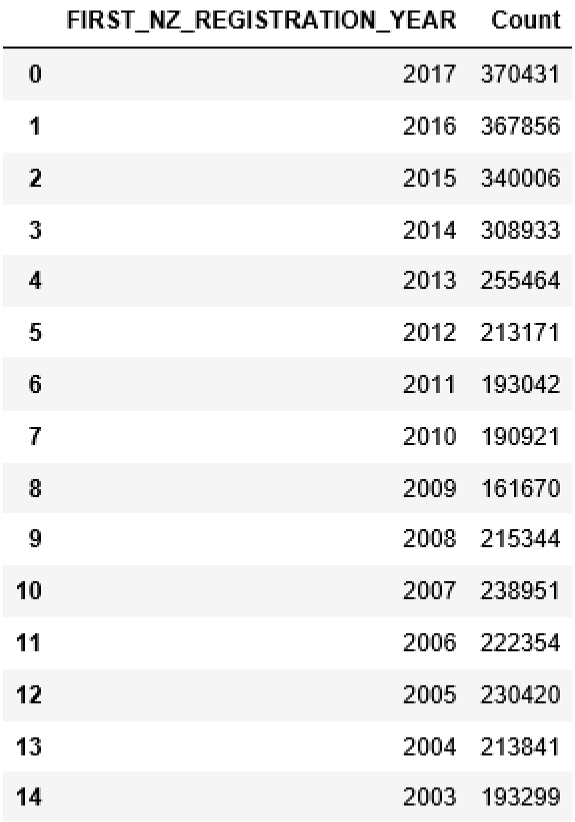
	MAKE	Count
0	TOYOTA	967765
1	NISSAN	491082
2	TRAILER	465880
3	MAZDA	347232
4	FORD	334040
5	HONDA	289657
6	MITSUBISHI	266473
7	HOLDEN	236895
8	SUZUKI	165627
9	SUBARU	132182

4. # What are the 20 most popular (by count) car models (where each (make, model) tuple counts as a different model), and the counts of these?
   dataframe = pandas.read_sql_query('SELECT MAKE,MODEL, COUNT(MODEL) AS Count FROM Fleet GROUP BY MAKE,MODEL ORDER BY Count DESC LIMIT 20', connection)
   dataframe



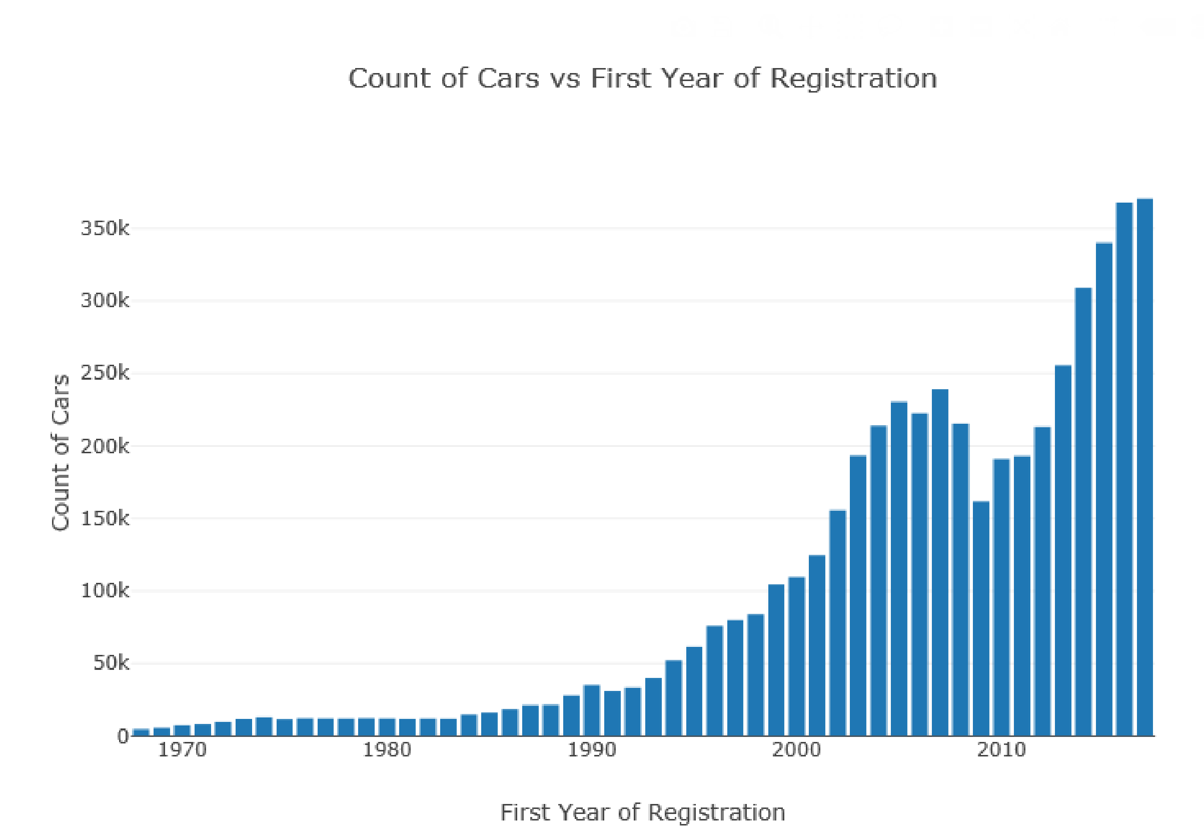
	MAKE	MODEL	Count
0	TOYOTA	COROLLA	170589
1	TOYOTA	HILUX	125273
2	HOLDEN	COMMODORE	86761
3	TOYOTA	HIACE	84895
4	SUZUKI	SWIFT	73171
5	FORD	FALCON	66504
6	TOYOTA	RAV4	62660
7	SUBARU	LEGACY	61038
8	TOYOTA	LANDCRUISER	49277
9	FORD	RANGER	49024
10	NISSAN	NAVARA	47799
11	TOYOTA	CAMRY	45312
12	TRAILER	HOMEBUILT	45102
13	HONDA	CIVIC	43851
14	MAZDA	DEMIO	43786
15	TRAILER	LOCAL	42656
16	VOLKSWAGEN	GOLF	42210
17	HONDA	ACCORD	42040
18	NISSAN	TIIDA	41343
19	MITSUBISHI	LANCER	40174

5.
   ```
   # How many cars are first registered in each of the most recent 50 years?
   dataframe = pandas.read_sql_query('SELECT
   FIRST_NZ_REGISTRATION_YEAR,
   COUNT(FIRST_NZ_REGISTRATION_YEAR) AS Count FROM Fleet WHERE
   FIRST_NZ_REGISTRATION_YEAR <>"" GROUP BY
   FIRST_NZ_REGISTRATION_YEAR ORDER BY
   FIRST_NZ_REGISTRATION_YEAR DESC LIMIT 50 ', connection)
   dataframe
   ```

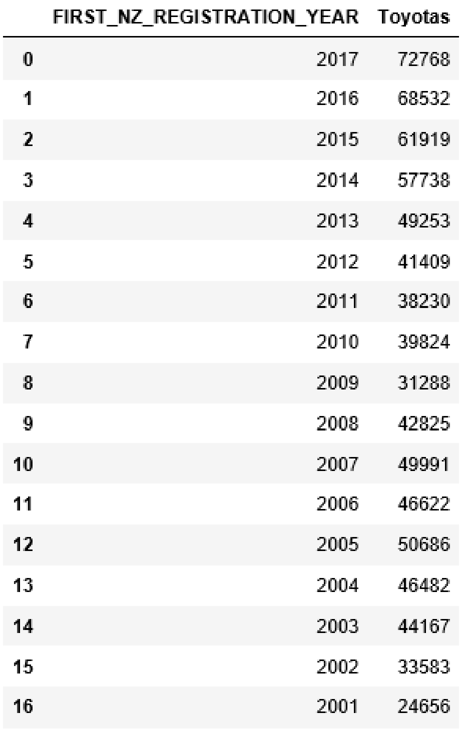


	FIRST_NZ_REGISTRATION_YEAR	Count
0	2017	370431
1	2016	367856
2	2015	340006
3	2014	308933
4	2013	255464
5	2012	213171
6	2011	193042
7	2010	190921
8	2009	161670
9	2008	215344
10	2007	238951
11	2006	222354
12	2005	230420
13	2004	213841
14	2003	193299

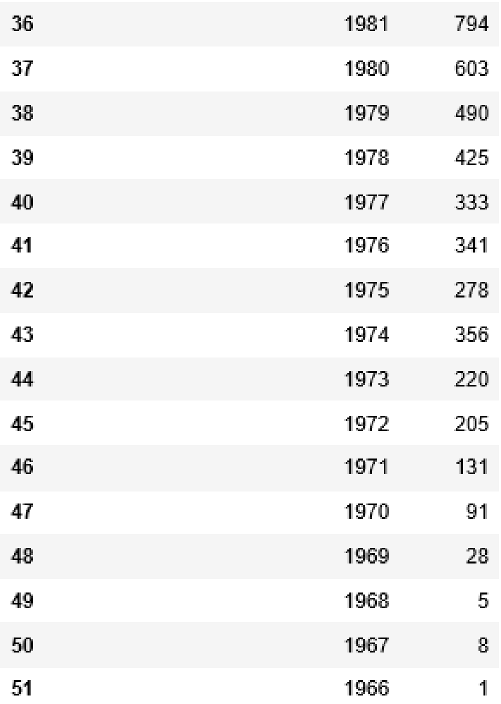
6.
   ```
   # Generate a plot with the previous answers
   dataframeNew = pandas.read_sql_query('SELECT
   FIRST_NZ_REGISTRATION_YEAR,
   COUNT(FIRST_NZ_REGISTRATION_YEAR) AS Count FROM Fleet WHERE
   FIRST_NZ_REGISTRATION_YEAR >=1968 GROUP BY
   FIRST_NZ_REGISTRATION_YEAR ORDER BY
   FIRST_NZ_REGISTRATION_YEAR ASC ', connection)
   trace =
   plotly.graph_objs.Bar(x=dataframeNew.FIRST_NZ_REGISTRATION_YEAR,
   y=dataframeNew.Count)
   layout = plotly.graph_objs.Layout(title="Count of Cars vs First Year of
   Registration",
               xaxis=dict(title='First Year of Registration'),
               yaxis=dict(title='Count of '))
   fig = plotly.graph_objs.Figure(data=[trace], layout=layout)
   plotly.offline.iplot(fig)
   ```



7.
   ```
   # How many Toyota cars were first registered in each year from 1950
   onwards?
   Toyota = pandas.read_sql_query('SELECT
   FIRST_NZ_REGISTRATION_YEAR, COUNT(MAKE) AS Toyotas FROM Fleet
   WHERE MAKE = "TOYOTA" AND FIRST_NZ_REGISTRATION_YEAR <> ""
   AND FIRST_NZ_REGISTRATION_YEAR > 1949 GROUP BY
   FIRST_NZ_REGISTRATION_YEAR ORDER BY
   FIRST_NZ_REGISTRATION_YEAR DESC' , connection)
   Toyota
   # No registered toyotas in New Zealand before 1966.
   ```

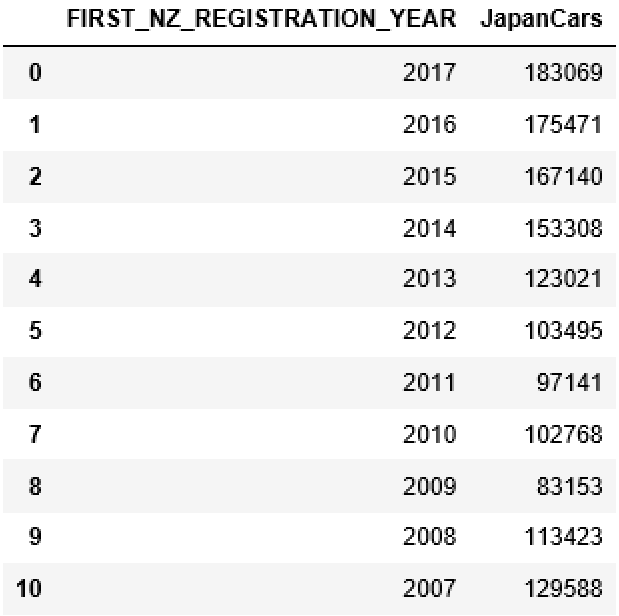


	FIRST_NZ_REGISTRATION_YEAR	Toyotas
0	2017	72768
1	2016	68532
2	2015	61919
3	2014	57738
4	2013	49253
5	2012	41409
6	2011	38230
7	2010	39824
8	2009	31288
9	2008	42825
10	2007	49991
11	2006	46622
12	2005	50686
13	2004	46482
14	2003	44167
15	2002	33583
16	2001	24656



36	1981	794
37	1980	603
38	1979	490
39	1978	425
40	1977	333
41	1976	341
42	1975	278
43	1974	356
44	1973	220
45	1972	205
46	1971	131
47	1970	91
48	1969	28
49	1968	5
50	1967	8
51	1966	1

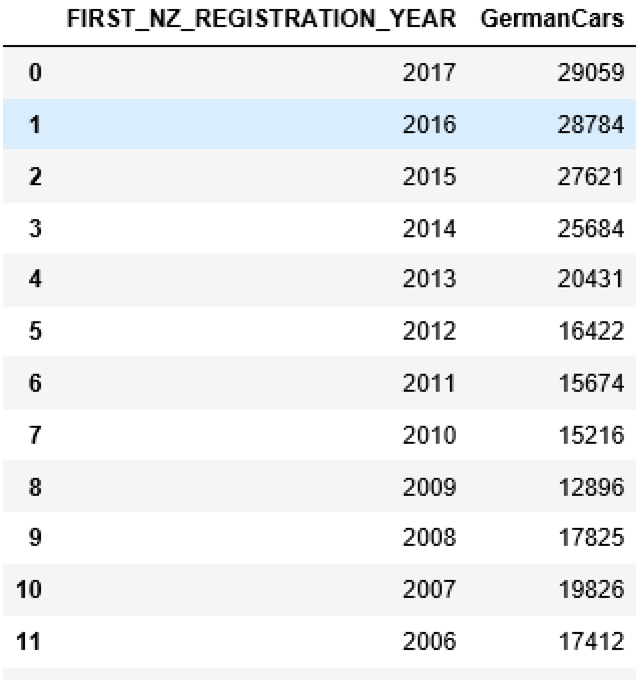
8. # How many cars from Japan (ORIGINAL_COUNTRY=”JAPAN”) were first registered in each year from 1950 onwards?
   Jap = pandas.read_sql_query('SELECT FIRST_NZ_REGISTRATION_YEAR, COUNT(ORIGINAL_COUNTRY) AS JapanCars FROM Fleet WHERE ORIGINAL_COUNTRY = "JAPAN" AND FIRST_NZ_REGISTRATION_YEAR <> "" AND FIRST_NZ_REGISTRATION_YEAR > 1949 GROUP BY FIRST_NZ_REGISTRATION_YEAR ORDER BY FIRST_NZ_REGISTRATION_YEAR DESC', connection)
   Jap



	FIRST_NZ_REGISTRATION_YEAR	JapanCars
0	2017	183069
1	2016	175471
2	2015	167140
3	2014	153308
4	2013	123021
5	2012	103495
6	2011	97141
7	2010	102768
8	2009	83153
9	2008	113423
10	2007	129588

9. # How many cars from Germany (ORIGINAL_COUNTRY=”GERMANY”) were first registered in each year from 1950 onwards?

```
Ger = pandas.read_sql_query('SELECT FIRST_NZ_REGISTRATION_YEAR,
COUNT(ORIGINAL_COUNTRY) AS GermanCars FROM Fleet WHERE
ORIGINAL_COUNTRY = "GERMANY" AND FIRST_NZ_REGISTRATION_YEAR
<> "" AND FIRST_NZ_REGISTRATION_YEAR > 1949 GROUP BY
FIRST_NZ_REGISTRATION_YEAR ORDER BY
FIRST_NZ_REGISTRATION_YEAR DESC', connection)
Ger
```

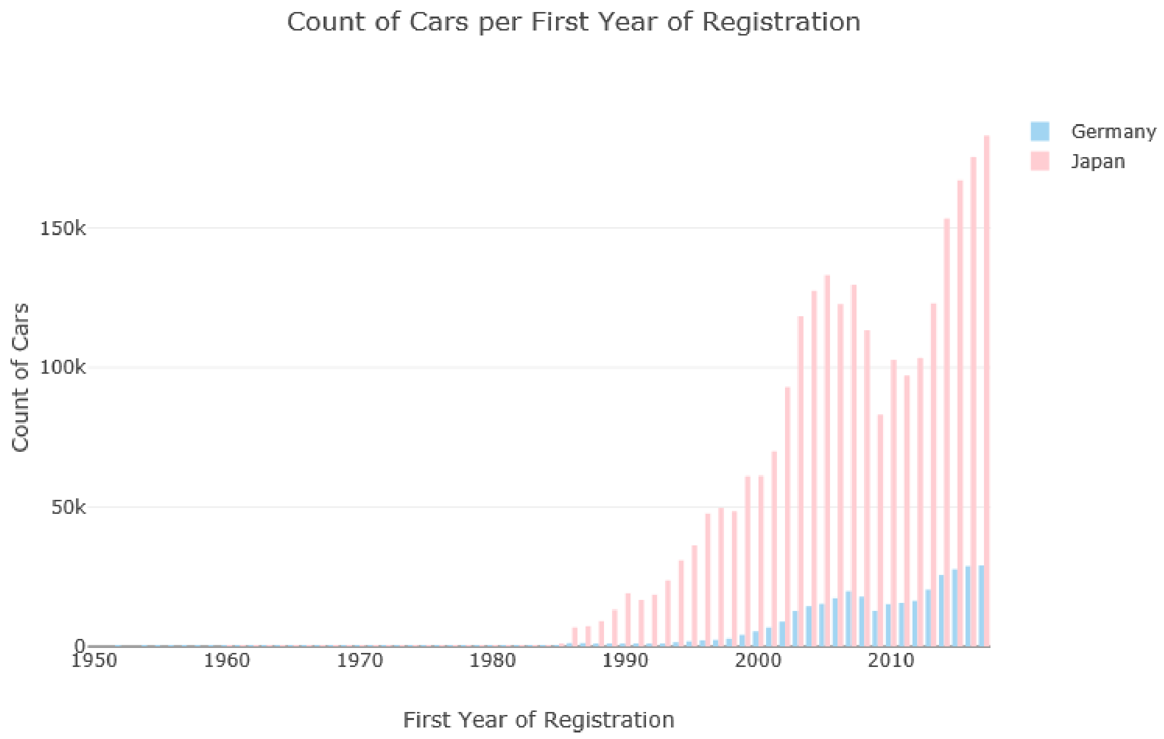


	FIRST_NZ_REGISTRATION_YEAR	GermanCars
0	2017	29059
1	2016	28784
2	2015	27621
3	2014	25684
4	2013	20431
5	2012	16422
6	2011	15674
7	2010	15216
8	2009	12896
9	2008	17825
10	2007	19826
11	2006	17412

10.

```
# 10. Generate a labelled bar plot (with a legend) showing this first-
registered data for Japan and Germany
trace_germany =
plotly.graph_objs.Bar(x=Ger.FIRST_NZ_REGISTRATION_YEAR,y=Ger.Ger
manCars,name = 'Germany',marker=dict(color='#A2D5F2'))
trace_japan =
plotly.graph_objs.Bar(x=Jap.FIRST_NZ_REGISTRATION_YEAR,y=Jap.Japa
nCars,name = 'Japan',marker=dict(color='#ffcdd2'))

layout = plotly.graph_objs.Layout(title="Count of Cars per First Year of
Registration",
              xaxis=dict(title='First Year of Registration'),
              yaxis=dict(title='Count of Cars'))
fig = plotly.graph_objs.Figure(data=[trace_germany,trace_japan],
layout=layout)
plotly.offline.iplot(fig)
```

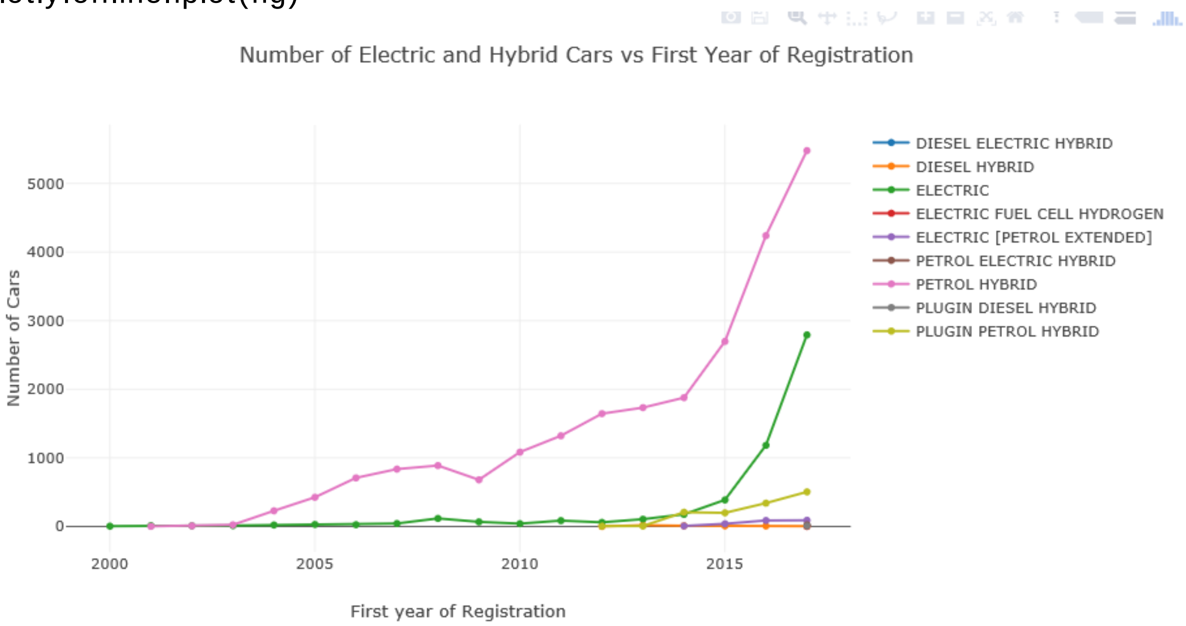


11.
```
# Create the scatter plot for the hybrids
# Find the list of all motive powers,
motive = pandas.read_sql_query('SELECT MOTIVE_POWER FROM Fleet
WHERE MOTIVE_POWER <> "DIESEL" AND MOTIVE_POWER <> "PETROL"
AND MOTIVE_POWER <> "" AND MOTIVE_POWER <> "CNG" AND
MOTIVE_POWER <> "LPG" AND MOTIVE_POWER <> "OTHER"  GROUP BY
MOTIVE_POWER', connection)

# Initialise trace storage vector
traces =[];
# For loop to run to create traces to append to a list of traces
for power in motive.MOTIVE_POWER:
    energy = pandas.read_sql_query('SELECT
FIRST_NZ_REGISTRATION_YEAR, COUNT(MOTIVE_POWER) AS Count
FROM Fleet WHERE MOTIVE_POWER = "{}" AND
FIRST_NZ_REGISTRATION_YEAR >=2000 GROUP BY
FIRST_NZ_REGISTRATION_YEAR ORDER BY
FIRST_NZ_REGISTRATION_YEAR ASC'.format(power), connection)
    # Create the plotting option
    energy_plot =
plotly.graph_objs.Scatter(x=energy.FIRST_NZ_REGISTRATION_YEAR,
                y=energy.Count,
                name = power)
    #Append to the traces for plotting
    traces.append(energy_plot);

layout = plotly.graph_objs.Layout(title="Number of Electric and Hybrid
Cars vs First Year of Registration",
                    xaxis=dict(title='First year of Registration'),
                    yaxis=dict(title='Number of Cars'))

# Plot them all
fig = plotly.graph_objs.Figure(data=traces, layout=layout)
plotly.offline.iplot(fig)
```


Number of Electric and Hybrid Cars vs First Year of Registration
Number of Cars
First year of Registration
DIESEL ELECTRIC HYBRID
DIESEL HYBRID
ELECTRIC
ELECTRIC FUEL CELL HYDROGEN
ELECTRIC [PETROL EXTENDED]
PETROL ELECTRIC HYBRID
PETROL HYBRID
PLUGIN DIESEL HYBRID
PLUGIN PETROL HYBRID