"Data is the new Oil"

You might have heard that statement quite a few times. World is increasingly generating more and more data every second. Did you know that as of right now, in one second, more than 10 hours of video are being uploaded to YouTube, 9190 Tweets are being sent out on Twitter, 1026 Instagram photos are being uploaded on Instagram and Google processes over 40,000 search queries. As you can see there are tons of different types of data which gets generated - video, text, image, transaction etc. And this is increasing exponentially with every year as the world is becoming more and more digital. Tools like SQL help us in getting meaningful insights out of this huge mountain of information.

- 1,209,600 new data producing social media users each day.
- 682 million tweets per day!
- More than 4 million hours of content uploaded to Youtube every day, with users watching 5.97 billion hours of Youtube videos each day.
- 67,305,600 Instagram posts uploaded each day
- There are over 2 billion monthly active facebook users, compared to 1.44 billion at the start of 2015 and 1.65 at the start of 2016.
- Facebook has 1.58 billion daily active users on average as of Q2 2019
- 4.3 BILLION Facebook messages posted daily!
- 5.76 BILLION Facebook likes every day.

Structured Query Language

For any analytics professional, Structured Query Language i.e. SQL is the most basic and a must have skill. Almost all the companies store their data in databases which are generally accessed by SQL. Unlike Python, SQL is not versatile or general purpose as it can only be used to manipulate data. And SQL is pretty powerful at manipulating data.

There are many dialects of SQL e.g. MySQL, T-SQL, PL/SQL, Hive SQL etc. Each of these dialects have most things in common, except for few syntax differences here and there. Depending on which software you are using for storing data(e.g. Hadoop/Azure/SAS/Teradata etc), you will be using different SQL dialects inside different softwares.

To keep things simple, here we will run SQL inside Python using pandasql module. We will highlight things which are currently not supported by pandasql module, but are by many other widely known SQL dialects.

In [2]:
```
pip install pandasql
```

```
Defaulting to user installation because normal site-packages is not writeable
```

```
Requirement already satisfied: pandasql in /home/devilinyou/.local/lib/python3.6/site-packages (0.7.3)
Requirement already satisfied: numpy in /home/devilinyou/.local/lib/python3.6/site-packages (from pandasql) (1.
19.4)
Requirement already satisfied: pandas in /home/devilinyou/.local/lib/python3.6/site-packages (from pandasql)
(1.1.4)
Requirement already satisfied: sqlalchemy in /home/devilinyou/.local/lib/python3.6/site-packages (from pandasq
l) (1.4.1)
Requirement already satisfied: python-dateutil>=2.7.3 in /home/devilinyou/.local/lib/python3.6/site-packages (f
rom pandas->pandasql) (2.8.1)
Requirement already satisfied: pytz>=2017.2 in /home/devilinyou/.local/lib/python3.6/site-packages (from pandas
->pandasql) (2020.4)
Requirement already satisfied: six>=1.5 in /home/devilinyou/.local/lib/python3.6/site-packages (from python-dat
eutil>=2.7.3->pandas->pandasql) (1.15.0)
Requirement already satisfied: importlib-metadata in /home/devilinyou/.local/lib/python3.6/site-packages (from
sqlalchemy->pandasql) (3.4.0)
Requirement already satisfied: greenlet!=0.4.17 in /home/devilinyou/.local/lib/python3.6/site-packages (from sq
lalchemy->pandasql) (1.0.0)
Requirement already satisfied: zipp>=0.5 in /home/devilinyou/.local/lib/python3.6/site-packages (from importlib
-metadata->sqlalchemy->pandasql) (3.4.0)
Requirement already satisfied: typing-extensions>=3.6.4 in /home/devilinyou/.local/lib/python3.6/site-packages
(from importlib-metadata->sqlalchemy->pandasql) (3.7.4.3)
Note: you may need to restart the kernel to use updated packages.
```

In [3]:
```
import pandas as pd
import pandasql as ps
```

In [4]:
```
pwd
```

Out[4]:
```
'/home/devilinyou/Downloads/DEsktop'
```

In [5]:
```
url = pd.read_csv(r"/home/devilinyou/Downloads/DEsktop/train.csv")
pd.options.display.max_columns = None
display(url)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

In [6]:
```
url.head(10) #choosing 1st 10 list of data
```

Out[6]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C

In [7]:
```
ps.sqldf("""
SELECT * FROM url
""")
```

Out[7]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	None	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	None	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	None	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	None	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	None	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	None	Q

891 rows × 12 columns

Reduce code using Url So we don't have to be As

In [8]:
```
ps.sqldf("""
SELECT df.* FROM url as df
""")
```

Out[8]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	None	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	None	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	None	S

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	None	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	None	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	None	Q

891 rows × 12 columns

Using Limit

Only using 10 row of data

In [9]:
```
ps.sqldf("""
SELECT df.* FROM url as df LIMIT 10
""")
```

Out[9]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	None	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	None	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	None	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	None	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	None	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	None	S

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	None	C

In [10]:
```
ps.sqldf("""
SELECT df.PassengerId, df.Name, df.Age
FROM url as df WHERE df.Age <1

""") #using where condition to filter
```

Out[10]:

	PassengerId	Name	Age
0	79	Caldwell, Master. Alden Gates	0.83
1	306	Allison, Master. Hudson Trevor	0.92
2	470	Baclini, Miss. Helene Barbara	0.75
3	645	Baclini, Miss. Eugenie	0.75
4	756	Hamalainen, Master. Viljo	0.67
5	804	Thomas, Master. Assad Alexander	0.42
6	832	Richards, Master. George Sibley	0.83

In [11]:
```
ps.sqldf("""
SELECT df.PassengerId, df.Name, df.Age  FROM url as df WHERE df.Age IS NULL

""")# using where statement to filter to find null value in dataframe
```

Out[11]:

	PassengerId	Name	Age
0	6	Moran, Mr. James	None
1	18	Williams, Mr. Charles Eugene	None
2	20	Masselmani, Mrs. Fatima	None
3	27	Emir, Mr. Farred Chehab	None
4	29	O'Dwyer, Miss. Ellen "Nellie"	None
...	...	...	...
172	860	Razi, Mr. Raihed	None
173	864	Sage, Miss. Dorothy Edith "Dolly"	None

	PassengerId	Name	Age
174	869	van Melkebeke, Mr. Philemon	None
175	879	Laleff, Mr. Kristo	None
176	889	Johnston, Miss. Catherine Helen "Carrie"	None

177 rows × 3 columns

In [12]:
```
ps.sqldf("""
SELECT df.PassengerId, df.Name, df.Age  FROM url as df WHERE df.Age  BETWEEN 5 AND 10

""")
```

Out[12]:

	PassengerId	Name	Age
0	25	Palsson, Miss. Torborg Danira	8.0
1	51	Panula, Master. Juha Niilo	7.0
2	59	West, Miss. Constance Mirium	5.0
3	148	Ford, Miss. Robina Maggie "Ruby"	9.0
4	166	Goldsmith, Master. Frank John William "Frankie"	9.0
5	183	Asplund, Master. Clarence Gustaf Hugo	9.0
6	234	Asplund, Miss. Lillian Gertrud	5.0
7	238	Collyer, Miss. Marjorie "Lottie"	8.0
8	279	Rice, Master. Eric	7.0
9	420	Van Impe, Miss. Catharina	10.0
10	449	Baclini, Miss. Marie Catherine	5.0
11	481	Goodwin, Master. Harold Victor	9.0
12	490	Coutts, Master. Eden Leslie "Neville"	9.0
13	536	Hart, Miss. Eva Miriam	7.0
14	542	Andersson, Miss. Ingeborg Constanzia	9.0
15	550	Davies, Master. John Morgan Jr	8.0
16	635	Skoog, Miss. Mabel	9.0
17	721	Harper, Miss. Annie Jessie "Nina"	6.0

	PassengerId	Name	Age
18	752	Moor, Master. Meier	6.0
19	778	Emanuel, Miss. Virginia Ethel	5.0
20	788	Rice, Master. George Hugh	8.0
21	814	Andersson, Miss. Ebba Iris Alfrida	6.0
22	820	Skoog, Master. Karl Thorsten	10.0
23	853	Boulos, Miss. Nourelain	9.0

In [13]:
```
#You can check for missing values by using "IS NULL"
ps.sqldf("""

SELECT
    df.*
FROM url as df
WHERE df.age IS NULL

""")
```

Out[13]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	6	0	3	Moran, Mr. James	male	None	0	0	330877	8.4583	None	Q
1	18	1	2	Williams, Mr. Charles Eugene	male	None	0	0	244373	13.0000	None	S
2	20	1	3	Masselmani, Mrs. Fatima	female	None	0	0	2649	7.2250	None	C
3	27	0	3	Emir, Mr. Farred Chehab	male	None	0	0	2631	7.2250	None	C
4	29	1	3	O'Dwyer, Miss. Ellen "Nellie"	female	None	0	0	330959	7.8792	None	Q
...	...	...	...	...	...	...	...	...	...	...	...	...
172	860	0	3	Razi, Mr. Raihed	male	None	0	0	2629	7.2292	None	C
173	864	0	3	Sage, Miss. Dorothy Edith "Dolly"	female	None	8	2	CA. 2343	69.5500	None	S
174	869	0	3	van Melkebeke, Mr. Philemon	male	None	0	0	345777	9.5000	None	S
175	879	0	3	Laleff, Mr. Kristo	male	None	0	0	349217	7.8958	None	S
176	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	None	1	2	W./C. 6607	23.4500	None	S

177 rows × 12 columns

In [14]:
```
ps.sqldf("""

SELECT
    df.*
FROM url as df
WHERE df.age IS NOT NULL

""")
```

Out[14]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	None	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	None	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	None	S
...	...	...	...	...	...	...	...	...	...	...	...	...
709	886	0	3	Rice, Mrs. William (Margaret Norton)	female	39.0	0	5	382652	29.1250	None	Q
710	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	None	S
711	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
712	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
713	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	None	Q

714 rows × 12 columns

In [15]:
```
ps.sqldf("""

SELECT
    df.*
FROM url AS df
WHERE df.age BETWEEN 5 AND 10

""")
```

Out[15]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	25	0	3	Palsson, Miss. Torborg Danira	female	8.0	3	1	349909	21.0750	None	S
1	51	0	3	Panula, Master. Juha Niilo	male	7.0	4	1	3101295	39.6875	None	S
2	59	1	2	West, Miss. Constance Mirium	female	5.0	1	2	C.A. 34651	27.7500	None	S
3	148	0	3	Ford, Miss. Robina Maggie "Ruby"	female	9.0	2	2	W./C. 6608	34.3750	None	S
4	166	1	3	Goldsmith, Master. Frank John William "Frankie"	male	9.0	0	2	363291	20.5250	None	S
5	183	0	3	Asplund, Master. Clarence Gustaf Hugo	male	9.0	4	2	347077	31.3875	None	S
6	234	1	3	Asplund, Miss. Lillian Gertrud	female	5.0	4	2	347077	31.3875	None	S
7	238	1	2	Collyer, Miss. Marjorie "Lottie"	female	8.0	0	2	C.A. 31921	26.2500	None	S
8	279	0	3	Rice, Master. Eric	male	7.0	4	1	382652	29.1250	None	Q
9	420	0	3	Van Impe, Miss. Catharina	female	10.0	0	2	345773	24.1500	None	S
10	449	1	3	Baclini, Miss. Marie Catherine	female	5.0	2	1	2666	19.2583	None	C
11	481	0	3	Goodwin, Master. Harold Victor	male	9.0	5	2	CA 2144	46.9000	None	S
12	490	1	3	Coutts, Master. Eden Leslie "Neville"	male	9.0	1	1	C.A. 37671	15.9000	None	S
13	536	1	2	Hart, Miss. Eva Miriam	female	7.0	0	2	F.C.C. 13529	26.2500	None	S
14	542	0	3	Andersson, Miss. Ingeborg Constanzia	female	9.0	4	2	347082	31.2750	None	S
15	550	1	2	Davies, Master. John Morgan Jr	male	8.0	1	1	C.A. 33112	36.7500	None	S
16	635	0	3	Skoog, Miss. Mabel	female	9.0	3	2	347088	27.9000	None	S
17	721	1	2	Harper, Miss. Annie Jessie "Nina"	female	6.0	0	1	248727	33.0000	None	S
18	752	1	3	Moor, Master. Meier	male	6.0	0	1	392096	12.4750	E121	S
19	778	1	3	Emanuel, Miss. Virginia Ethel	female	5.0	0	0	364516	12.4750	None	S
20	788	0	3	Rice, Master. George Hugh	male	8.0	4	1	382652	29.1250	None	Q
21	814	0	3	Andersson, Miss. Ebba Iris Alfrida	female	6.0	4	2	347082	31.2750	None	S

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
22	820	0	3	Skoog, Master. Karl Thorsten	male	10.0	3	2	347088	27.9000	None	S
23	853	0	3	Boulos, Miss. Nourelain	female	9.0	1	1	2678	15.2458	None	C

In [16]:
```
ps.sqldf("""

SELECT
    df.*
FROM  url AS df
WHERE df.age < 1
    AND df.Sex = "female"

""")
```

Out[16]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	470	1	3	Baclini, Miss. Helene Barbara	female	0.75	2	1	2666	19.2583	None	C
1	645	1	3	Baclini, Miss. Eugenie	female	0.75	2	1	2666	19.2583	None	C

In [17]:
```
ps.sqldf("""

SELECT
    df.*
FROM  url AS df
WHERE df.age < 1
    AND df.Sex = "male"

""")
```

Out[17]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	79	1	2	Caldwell, Master. Alden Gates	male	0.83	0	2	248738	29.0000	None	S
1	306	1	1	Allison, Master. Hudson Trevor	male	0.92	1	2	113781	151.5500	C22 C26	S
2	756	1	2	Hamalainen, Master. Viljo	male	0.67	1	1	250649	14.5000	None	S
3	804	1	3	Thomas, Master. Assad Alexander	male	0.42	0	1	2625	8.5167	None	C
4	832	1	2	Richards, Master. George Sibley	male	0.83	1	1	29106	18.7500	None	S

In [18]:
```
ps.sqldf("""
```

```
SELECT
    df.*
FROM url AS df
WHERE (df.age < 1 AND df.Sex = "female")
    OR (df.age > 70 AND df.Pclass = 1)

""")
```

Out[18]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	97	0	1	Goldschmidt, Mr. George B	male	71.00	0	0	PC 17754	34.6542	A5	C
1	470	1	3	Baclini, Miss. Helene Barbara	female	0.75	2	1	2666	19.2583	None	C
2	494	0	1	Artagaveytia, Mr. Ramon	male	71.00	0	0	PC 17609	49.5042	None	C
3	631	1	1	Barkworth, Mr. Algernon Henry Wilson	male	80.00	0	0	27042	30.0000	A23	S
4	645	1	3	Baclini, Miss. Eugenie	female	0.75	2	1	2666	19.2583	None	C

In [19]:

```
ps.sqldf("""

SELECT
    df.*
FROM url AS df
WHERE  (df.age > 70 AND df.Pclass = 1)

""")
```

Out[19]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	97	0	1	Goldschmidt, Mr. George B	male	71.0	0	0	PC 17754	34.6542	A5	C
1	494	0	1	Artagaveytia, Mr. Ramon	male	71.0	0	0	PC 17609	49.5042	None	C
2	631	1	1	Barkworth, Mr. Algernon Henry Wilson	male	80.0	0	0	27042	30.0000	A23	S

In [20]:

```
ps.sqldf("""

SELECT
    df.*
FROM url AS df
WHERE   df.Pclass = 1
```

```
""")
```

Out[20]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
1	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
2	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
3	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.5500	C103	S
4	24	1	1	Sloper, Mr. William Thompson	male	28.0	0	0	113788	35.5000	A6	S
...	...	...	...	...	...	...	...	...	...	...	...	...
211	872	1	1	Beckwith, Mrs. Richard Leonard (Sallie Monypeny)	female	47.0	1	1	11751	52.5542	D35	S
212	873	0	1	Carlsson, Mr. Frans Olof	male	33.0	0	0	695	5.0000	B51 B53 B55	S
213	880	1	1	Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)	female	56.0	0	1	11767	83.1583	C50	C
214	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
215	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C

216 rows × 12 columns

Using "AND" and "OR" together might be tricky at times. It is best to use parantheses to enforce which conditions will be evaluated together. E.g. in the above example we have four conditions in total. First the two conditions inside the parantheses will be evaluated separately using "AND". And then the output of those two will be evaluated using "OR" statement

In [21]:

```
ps.sqldf("""

SELECT
    df.*
FROM url AS df
WHERE df.Cabin IN ('A5', 'A6', 'A7','E46','C103','C85')

""")
```

Out[21]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
1	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
2	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.5500	C103	S
3	24	1	1	Sloper, Mr. William Thompson	male	28.0	0	0	113788	35.5000	A6	S
4	97	0	1	Goldschmidt, Mr. George B	male	71.0	0	0	PC 17754	34.6542	A5	C
5	175	0	1	Smith, Mr. James Clinch	male	56.0	0	0	17764	30.6958	A7	C

If there are multiple values you want to filter in a column, you can use "IN" keyword

In [22]:
```
ps.sqldf("""

SELECT
    df.*
FROM  url AS df
WHERE df.Age IN (65, 70, 71, 80)

""")
```

Out[22]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	55	0	1	Ostby, Mr. Engelhart Cornelius	male	65.0	0	1	113509	61.9792	B30	C
1	97	0	1	Goldschmidt, Mr. George B	male	71.0	0	0	PC 17754	34.6542	A5	C
2	281	0	3	Duane, Mr. Frank	male	65.0	0	0	336439	7.7500	None	Q
3	457	0	1	Millet, Mr. Francis Davis	male	65.0	0	0	13509	26.5500	E38	S
4	494	0	1	Artagaveytia, Mr. Ramon	male	71.0	0	0	PC 17609	49.5042	None	C
5	631	1	1	Barkworth, Mr. Algernon Henry Wilson	male	80.0	0	0	27042	30.0000	A23	S
6	673	0	2	Mitchell, Mr. Henry Michael	male	70.0	0	0	C.A. 24580	10.5000	None	S
7	746	0	1	Crosby, Capt. Edward Gifford	male	70.0	1	1	WE/P 5735	71.0000	B22	S

For text/string columns, a powerful way of filtering is using "LIKE" keyword. It works by the specifying the substring you want in a string and surrounding the substring with '%'s. Here is how it looks with examples below -

In [23]:
```
ps.sqldf("""
```

```
SELECT
    df.*
FROM url AS df
WHERE df.Name LIKE "%MRS.%"
LIMIT 5

""")
```

Out[23]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
1	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
2	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	None	S
3	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	None	C
4	16	1	2	Hewlett, Mrs. (Mary D Kingcome)	female	55.0	0	0	248706	16.0000	None	S

In [24]:

```
ps.sqldf("""

SELECT
    df.Age,
    df.PassengerID,
    df.Fare,
    df.Age * df.Age AS age_sq,
    df.Age + 1 AS age_plus_one,
    df.Fare / df.PassengerID AS PerHead_person
FROM url AS df
LIMIT 5

""")
```

Out[24]:

	Age	PassengerId	Fare	age_sq	age_plus_one	PerHead_person
0	22.0	1	7.2500	484.0	23.0	7.250000
1	38.0	2	71.2833	1444.0	39.0	35.641650
2	26.0	3	7.9250	676.0	27.0	2.641667
3	35.0	4	53.1000	1225.0	36.0	13.275000

	Age	PassengerId	Fare	age_sq	age_plus_one	PerHead_person
4	35.0	5	8.0500	1225.0	36.0	1.610000

In [25]:
```
#Concatenating Sex and Ticket
ps.sqldf("""

SELECT
    df.Sex,
    df.Ticket,
    df.Sex || df.Ticket AS Sex_Plus_ticket
FROM url AS df
LIMIT 5

""")
```

Out[25]:

	Sex	Ticket	Sex_Plus_ticket
0	male	A/5 21171	maleA/5 21171
1	female	PC 17599	femalePC 17599
2	female	STON/O2. 3101282	femaleSTON/O2. 3101282
3	female	113803	female113803
4	male	373450	male373450

In [26]:
```
ps.sqldf("""

SELECT
    df.Name,
    df.PassengerID,
    df.Age,
    CASE WHEN df.Age IS NULL THEN '0. Missing'
        WHEN df.Age <18 THEN '1. 1-17'
        WHEN df.Age <60 THEN '2. 18-60'
        WHEN df.Age =60 THEN '3. 60'
    ELSE '4. 60+' END AS age_bucket
FROM url AS df
LIMIT 50

""")#Feature engineering
```

Out[26]:

	Name	PassengerId	Age	age_bucket

	Name	PassengerId	Age	age_bucket
0	Braund, Mr. Owen Harris	1	22.0	2. 18-60
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	2	38.0	2. 18-60
2	Heikkinen, Miss. Laina	3	26.0	2. 18-60
3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	4	35.0	2. 18-60
4	Allen, Mr. William Henry	5	35.0	2. 18-60
5	Moran, Mr. James	6	NaN	0. Missing
6	McCarthy, Mr. Timothy J	7	54.0	2. 18-60
7	Palsson, Master. Gosta Leonard	8	2.0	1. 1-17
8	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	9	27.0	2. 18-60
9	Nasser, Mrs. Nicholas (Adele Achem)	10	14.0	1. 1-17
10	Sandstrom, Miss. Marguerite Rut	11	4.0	1. 1-17
11	Bonnell, Miss. Elizabeth	12	58.0	2. 18-60
12	Saundercock, Mr. William Henry	13	20.0	2. 18-60
13	Andersson, Mr. Anders Johan	14	39.0	2. 18-60
14	Vestrom, Miss. Hulda Amanda Adolfina	15	14.0	1. 1-17
15	Hewlett, Mrs. (Mary D Kingcome)	16	55.0	2. 18-60
16	Rice, Master. Eugene	17	2.0	1. 1-17
17	Williams, Mr. Charles Eugene	18	NaN	0. Missing
18	Vander Planke, Mrs. Julius (Emelia Maria Vande...	19	31.0	2. 18-60
19	Masselmani, Mrs. Fatima	20	NaN	0. Missing
20	Fynney, Mr. Joseph J	21	35.0	2. 18-60
21	Beesley, Mr. Lawrence	22	34.0	2. 18-60
22	McGowan, Miss. Anna "Annie"	23	15.0	1. 1-17
23	Sloper, Mr. William Thompson	24	28.0	2. 18-60
24	Palsson, Miss. Torborg Danira	25	8.0	1. 1-17
25	Asplund, Mrs. Carl Oscar (Selma Augusta Emilia...	26	38.0	2. 18-60
26	Emir, Mr. Farred Chehab	27	NaN	0. Missing

	Name	PassengerId	Age	age_bucket
27	Fortune, Mr. Charles Alexander	28	19.0	2. 18-60
28	O'Dwyer, Miss. Ellen "Nellie"	29	NaN	0. Missing
29	Todoroff, Mr. Lalio	30	NaN	0. Missing
30	Uruchurtu, Don. Manuel E	31	40.0	2. 18-60
31	Spencer, Mrs. William Augustus (Marie Eugenie)	32	NaN	0. Missing
32	Glynn, Miss. Mary Agatha	33	NaN	0. Missing
33	Wheadon, Mr. Edward H	34	66.0	4. 60+
34	Meyer, Mr. Edgar Joseph	35	28.0	2. 18-60
35	Holverson, Mr. Alexander Oskar	36	42.0	2. 18-60
36	Mamee, Mr. Hanna	37	NaN	0. Missing
37	Cann, Mr. Ernest Charles	38	21.0	2. 18-60
38	Vander Planke, Miss. Augusta Maria	39	18.0	2. 18-60
39	Nicola-Yarred, Miss. Jamila	40	14.0	1. 1-17
40	Ahlin, Mrs. Johan (Johanna Persdotter Larsson)	41	40.0	2. 18-60
41	Turpin, Mrs. William John Robert (Dorothy Ann ...	42	27.0	2. 18-60
42	Kraeff, Mr. Theodor	43	NaN	0. Missing
43	Laroche, Miss. Simonne Marie Anne Andree	44	3.0	1. 1-17
44	Devaney, Miss. Margaret Delia	45	19.0	2. 18-60
45	Rogers, Mr. William John	46	NaN	0. Missing
46	Lennon, Mr. Denis	47	NaN	0. Missing
47	O'Driscoll, Miss. Bridget	48	NaN	0. Missing
48	Samaan, Mr. Youssef	49	NaN	0. Missing
49	Arnold-Franchi, Mrs. Josef (Josefine Franchi)	50	18.0	2. 18-60

SUBSTR() function can be used to take out a part of a string from the text columns in a dataset. SUBSTR() function requires three inputs - SUBSTR(column_name, starting_point, number_of_characters)

In [27]:
```
ps.sqldf("""
```

```
SELECT
    df.Name,
    SUBSTR(df.Name,2, 4) AS out
FROM url AS df
LIMIT 5

""")
```

Out[27]:

	Name	out
0	Braund, Mr. Owen Harris	raun
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	umin
2	Heikkinen, Miss. Laina	eikk
3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	utre
4	Allen, Mr. William Henry	llen

In [28]:

```
ps.sqldf("""

SELECT
    df.*
FROM url AS df LIMIT 100

""")
```

Out[28]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	None	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	None	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	None	S
...	...	...	...	...	...	...	...	...	...	...	...	...
95	96	0	3	Shorney, Mr. Charles Joseph	male	NaN	0	0	374910	8.0500	None	S
96	97	0	1	Goldschmidt, Mr. George B	male	71.0	0	0	PC 17754	34.6542	A5	C

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
97	98	1	1	Greenfield, Mr. William Bertram	male	23.0	0	1	PC 17759	63.3583	D10 D12	C
98	99	1	2	Doling, Mrs. John T (Ada Julia Bone)	female	34.0	0	1	231919	23.0000	None	S
99	100	0	2	Kantor, Mr. Sinai	male	34.0	1	0	244367	26.0000	None	S

100 rows × 12 columns

lets find Distinct keyword for more knowleage about matter

You can get all the unique values in a column using "DISTINCT" keyword

In [29]:
```
ps.sqldf("""

SELECT
    DISTINCT df.Embarked
FROM url AS df

""")
```

Out[29]:

	Embarked
0	S
1	C
2	Q
3	None

aggregation

SQL provides multiple ways in which you can aggregate information at different levels. You will need to use "GROUP BY" in the end and specify on which column(s) you are aggregating the data.

In [30]:
```
ps.sqldf("""

SELECT
```

```
    df.Embarked,
    AVG(df.Age) AS avg_age,
    SUM(df.Age) AS tot_age,
    MIN(df.Age) AS min_age,
    MAX(df.Age) AS max_age,
    COUNT(df.PassengerId) AS tot_passengers,
    COUNT(DISTINCT df.Ticket) AS dist_tickets
FROM url AS df
GROUP BY df.Embarked

""")
```

Out[30]:

	Embarked	avg_age	tot_age	min_age	max_age	tot_passengers	dist_tickets
0	None	50.000000	100.00	38.00	62.0	2	1
1	C	30.814769	4005.92	0.42	71.0	168	122
2	Q	28.089286	786.50	2.00	70.5	77	66
3	S	29.445397	16312.75	0.67	80.0	644	494

An aggregate function performs a calculation on a set of values, and returns a single value. Except for COUNT(*) , aggregate functions ignore null values. Aggregate functions are often used with the GROUP BY clause of the SELECT statement. All aggregate functions are deterministic Apart from the aggregate functions used above, many SQL dialects also support STDEV for standard deviation, VAR for variance, SQRT for square root and many more. You can always google to find out which aggregate functions are supported by the SQL dialect you are using

In [31]:

```
ps.sqldf("""

SELECT
    df.Embarked,
    df.Sex,
    AVG(df.Age) AS avg_age,
    SUM(df.Age) AS tot_age,
    MIN(df.Age) AS min_age,
    MAX(df.Age) AS max_age,
    COUNT(df.PassengerId) AS tot_passengers,
    COUNT(DISTINCT df.Ticket) AS dist_tickets

FROM url AS df
GROUP BY df.Embarked,
    df.Sex

""")
```

Out[31]:

	Embarked	Sex	avg_age	tot_age	min_age	max_age	tot_passengers	dist_tickets
0	None	female	50.000000	100.00	38.00	62.0	2	1
1	C	female	28.344262	1729.00	0.75	60.0	73	54
2	C	male	32.998841	2276.92	0.42	71.0	95	88
3	Q	female	24.291667	291.50	15.00	39.0	36	35
4	Q	male	30.937500	495.00	2.00	70.5	41	37
5	S	female	27.771505	5165.50	1.00	63.0	203	158
6	S	male	30.291440	11147.25	0.67	80.0	441	394

In [32]:

```
ps.sqldf("""

SELECT
    df.Embarked,
    df.Sex,
    AVG(df.Age) AS avg_age,
    SUM(df.Age) AS tot_age,
    MIN(df.Age) AS min_age,
    MAX(df.Age) AS max_age,
    COUNT(df.PassengerId) AS tot_passengers,
    COUNT(DISTINCT df.Ticket) AS dist_tickets

FROM url AS df

WHERE age >= 18
    AND age <=60

GROUP BY df.Embarked,
    df.Sex

ORDER BY df.Embarked,
    df.Sex

""")
```

Out[32]:

	Embarked	Sex	avg_age	tot_age	min_age	max_age	tot_passengers	dist_tickets
0	None	female	38.000000	38.0	38.0	38.0	1	1
1	C	female	35.295455	1553.0	18.0	60.0	44	35

	Embarked	Sex	avg_age	tot_age	min_age	max_age	tot_passengers	dist_tickets
2	C	male	33.838983	1996.5	18.0	60.0	59	56
3	Q	female	27.166667	244.5	18.0	39.0	9	9
4	Q	male	33.850000	338.5	19.0	57.0	10	10
5	S	female	31.909396	4754.5	18.0	58.0	149	136
6	S	male	31.941368	9806.0	18.0	60.0	307	292

HAVING

Lastly, once you have aggregated the information, you can further put filters on the aggregated output using "HAVING". Always remember the difference between "WHERE" and "HAVING" as it is one of the most commonly asked interview questions. "WHERE" is used to filter the data directly from the tables while "HAVING" is used to filter the aggregated data you get as the output of a SQL

First we used "WHERE", then "GROUP BY", then "HAVING" and lastly "ORDER BY"

In [33]:
```
ps.sqldf("""

SELECT
    df.Embarked,
    df.Sex,
    AVG(df.Age) AS avg_age,
    SUM(df.Age) AS tot_age,
    MIN(df.Age) AS min_age,
    MAX(df.Age) AS max_age,
    COUNT(df.PassengerId) AS tot_passengers,
    COUNT(DISTINCT df.Ticket) AS dist_tickets

FROM url AS df

WHERE age >= 18
    AND age <=60

GROUP BY df.Embarked,
    df.Sex
HAVING avg_age >= 30

ORDER BY df.Embarked,
    df.Sex

""")
```

Out[33]:

	Embarked	Sex	avg_age	tot_age	min_age	max_age	tot_passengers	dist_tickets
0	None	female	38.000000	38.0	38.0	38.0	1	1
1	C	female	35.295455	1553.0	18.0	60.0	44	35
2	C	male	33.838983	1996.5	18.0	60.0	59	56
3	Q	male	33.850000	338.5	19.0	57.0	10	10
4	S	female	31.909396	4754.5	18.0	58.0	149	136
5	S	male	31.941368	9806.0	18.0	60.0	307	292

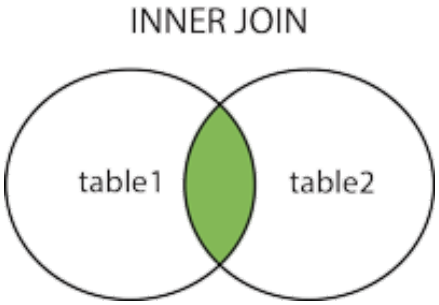
In [34]:
```
#Joints
cities = pd.DataFrame({'code':['S', 'C', 'L'],'city':['Southampton', 'Cherbourg', 'London']})
cities
```

Out[34]:

	code	city
0	S	Southampton
1	C	Cherbourg
2	L	London

INNER JOIN

INNER JOIN gives rows corresponding to only the common values(in the columns used in the inner join condition) between the two tables



In [35]:
```
ps.sqldf("""

SELECT
    c.city,
    df.Embarked,
    df.Sex,
    AVG(df.Age) AS avg_age,
```

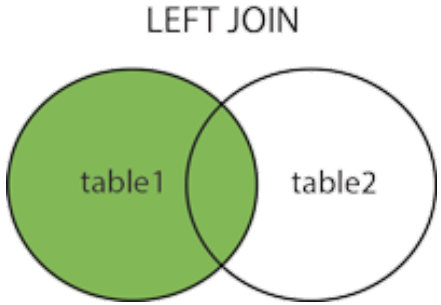
```
    SUM(df.Age) AS tot_age,
    MIN(df.Age) AS min_age,
    MAX(df.Age) AS max_age,
    COUNT(df.PassengerId) AS tot_passengers,
    COUNT(DISTINCT df.Ticket) AS dist_tickets

FROM url AS df

INNER JOIN cities AS c
    ON df.Embarked = c.code

GROUP BY df.Embarked,
    df.Sex

""")
```

Out[35]:

	city	Embarked	Sex	avg_age	tot_age	min_age	max_age	tot_passengers	dist_tickets
0	Cherbourg	C	female	28.344262	1729.00	0.75	60.0	73	54
1	Cherbourg	C	male	32.998841	2276.92	0.42	71.0	95	88
2	Southampton	S	female	27.771505	5165.50	1.00	63.0	203	158
3	Southampton	S	male	30.291440	11147.25	0.67	80.0	441	394

LEFT JOIN

LEFT JOIN gives all rows from the left table and only those rows from right table where the values are common in the columns



used in left join condition

In [36]:

```
ps.sqldf("""

SELECT
    c.city,
    df.Embarked,
    df.Sex,
    AVG(df.Age) AS avg_age,
```

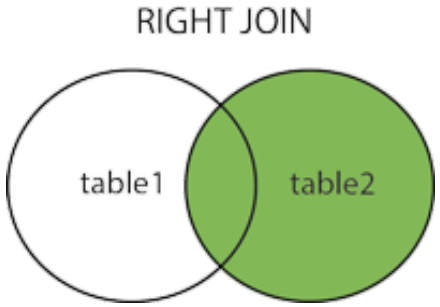
```
    SUM(df.Age) AS tot_age,
    MIN(df.Age) AS min_age,
    MAX(df.Age) AS max_age,
    COUNT(df.PassengerId) AS tot_passengers,
    COUNT(DISTINCT df.Ticket) AS dist_tickets

FROM url AS df

LEFT JOIN cities AS c
    ON df.Embarked = c.code

GROUP BY df.Embarked,
    df.Sex

""")
```

Out[36]:

	city	Embarked	Sex	avg_age	tot_age	min_age	max_age	tot_passengers	dist_tickets
0	None	None	female	50.000000	100.00	38.00	62.0	2	1
1	Cherbourg	C	female	28.344262	1729.00	0.75	60.0	73	54
2	Cherbourg	C	male	32.998841	2276.92	0.42	71.0	95	88
3	None	Q	female	24.291667	291.50	15.00	39.0	36	35
4	None	Q	male	30.937500	495.00	2.00	70.5	41	37
5	Southampton	S	female	27.771505	5165.50	1.00	63.0	203	158
6	Southampton	S	male	30.291440	11147.25	0.67	80.0	441	394

RIGHT JOIN

RIGHT JOIN is the mirror opposite of LEFT JOIN. RIGHT JOIN gives all rows from the right table and only those rows from left table where the values are common in the columns used in right join condition.



In [37]:

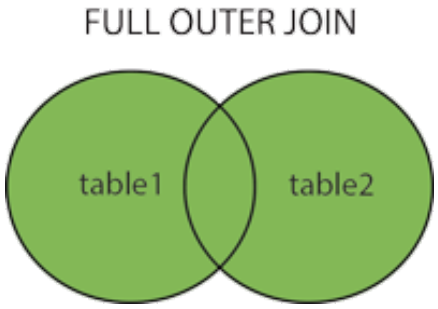
```
ps.sqldf("""
```

```
SELECT
    c.city,
    df.Embarked,
    df.Sex,
    AVG(df.Age) AS avg_age,
    SUM(df.Age) AS tot_age,
    MIN(df.Age) AS min_age,
    MAX(df.Age) AS max_age,
    COUNT(df.PassengerId) AS tot_passengers,
    COUNT(DISTINCT df.Ticket) AS dist_tickets

FROM cities AS c

LEFT JOIN url AS df
    ON df.Embarked = c.code

GROUP BY df.Embarked,
    df.Sex

""")
```

Out[37]:

	city	Embarked	Sex	avg_age	tot_age	min_age	max_age	tot_passengers	dist_tickets
0	London	None	None	NaN	NaN	NaN	NaN	0	0
1	Cherbourg	C	female	28.344262	1729.00	0.75	60.0	73	54
2	Cherbourg	C	male	32.998841	2276.92	0.42	71.0	95	88
3	Southampton	S	female	27.771505	5165.50	1.00	63.0	203	158
4	Southampton	S	male	30.291440	11147.25	0.67	80.0	441	394

FULL JOIN



FULL JOIN gives all rows from the from both tables, irrespective of common values

In [38]:

```
pd.merge(url.groupby('Embarked')['Age'].mean().reset_index(),
         cities.rename({'code':'Embarked'}, axis = 'columns'),
```

```
          on='Embarked',
          how='outer')
```

Out[38]:

	Embarked	Age	city
0	C	30.814769	Cherbourg
1	Q	28.089286	NaN
2	S	29.445397	Southampton
3	L	NaN	London

In [39]:

```
#Using "UNION" to stack datasets
ps.sqldf("""

SELECT DISTINCT df.Embarked FROM url AS df
UNION
SELECT DISTINCT c.code FROM cities AS c

""")
```

Out[39]:

	Embarked
0	None
1	C
2	L
3	Q
4	S

In [40]:

```
#For all rows
ps.sqldf("""

SELECT DISTINCT df.Embarked FROM url AS df
UNION ALL
SELECT DISTINCT c.code FROM cities AS c

""")
```

Out[40]:

	Embarked
0	S

	Embarked
1	C
2	Q
3	None
4	S
5	C
6	L

In [41]:
```
#Using "EXCEPT" to take rows from first dataset and then remove rows which are present in both datasets
ps.sqldf("""

SELECT DISTINCT df.Embarked FROM url AS df
EXCEPT
SELECT DISTINCT c.code FROM cities AS c

""")
```

Out[41]:

	Embarked
0	None
1	Q

In [42]:
```
#Using "EXCEPT" to take rows from first dataset and then remove rows which are present in both datasets
ps.sqldf("""

SELECT DISTINCT c.code FROM cities AS c
EXCEPT
SELECT DISTINCT df.Embarked FROM url AS df

""")
```

Out[42]:

	code
0	L

In [ ]:

In [ ]: