SQL in Pythonic way via Jupyter Notebook

@ Ramendra Kumar

Requirement

- 1.Install MySql Server in your system. Note down its root password.
- 2.Install following packages in Anaconda by running cell below.

```
In [16]: #!pip install mysqlclient
         #!pip install sqlalchemy
         #!pip install ipython-sql
```

Description

- mysqlclient -->It provides connection to MySQL database using Python. Before you can access MySQL databases using Python, you must install mysqlclient packages. Other packages are also available viz: mysql-connector-python,PyMySQL.
- sqlalchemy --> It provides a full suite of well known enterprise-level persistence patterns, designed for efficient and high-performing database access, adapted into a simple and Pythonic domain language.
- ipython-sql -->It introduces a %sql (or %%sql) magic to your notebook allowing you to connect to a database, using SQLAlchemy connect strings, then issue SQL commands within IPython or IPython Notebook.

```
In [1]: import pandas as pd
        import sqlalchemy
```

Creating a Database

- For time being, for sake of simplicity --> Database is collection of tables.
- We are going to create a database named 'Aashvi'. It is blank initialy. We will add tables in it.

```
In [3]: engine2 = sqlalchemy.create_engine('mysql://root:r1m2n3#?@localhost') # for connect
        ing to server
        engine2.execute("CREATE DATABASE Aashvi") # for creating a  database named  Aashvi
```

```
In [4]: engine2.execute("USE Aashvi") # For selecting particular db in server.Currently we
         have only one i.e. Aashvi

Out[4]: <sqlalchemy.engine.result.ResultProxy at 0x8ae4548>
```

```
In [5]: engine2.connect()
        print(engine2)

        Engine(mysql://root:***@localhost)
```

ipython-sql library is loaded using the %load_ext

In [6]:
```
%load_ext sql
```

In [11]:
```
# ipython-sql library is loaded using the %load_ext
# To connect to the database you need to pass connection string in SQLAlchemy forma
t to the
%sql mysql://root:r1m2n3#?@localhost/Aashvi
```

Loading Iris data set through pandas

In [15]:
```
data=pd.read_csv('iris.csv')
data.head()
```

Out[15]:

	SepalLength	SepalWidth	PetalLength	PetalWidth	Name
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

For making a sql table in DB from pandas dataframe

In [115]:
```
# Making a datafrme into sql table and storing in aashvi database  then working on
 it
data.to_sql('Iris_N',con=engine2)
# Now we have a sql table  named Iris_N in aashvi database we will work on it.
```

For Dropping/Deleting any table from DB

In [21]:
```
%%sql
DROP TABLE Iris_N
```

```
 * mysql://root:***@localhost/Aashvi
0 rows affected.
```

Out[21]:
```
[]
```

Note: Once a table is dropped/deleted,we have create it again to access it or to work on it.

Module 1

Selecting multiple(coma seperated) or single column from a Table

- Note: anything inside this --> /*,------*/ is comment line, not executed as code.

In [30]:
```
%%sql
SELECT SepalLength,PetalWidth FROM Iris_N
LIMIT 5           /*,For limiting the result upto only 5 rows, without  LIMIT all r
ows will be displayed*/
```

```
 * mysql://root:***@localhost/Aashvi
5 rows affected.
```

Out[30]:

SepalLength	PetalWidth
5.1	0.2
4.9	0.2
4.7	0.2
4.6	0.2
5.0	0.2

(*) wildcard character gives all colums of a table at a time, you dont have to write each column name individually

In [31]:
```
%%sql
SELECT * FROM Iris_N
LIMIT 5
```

```
 * mysql://root:***@localhost/Aashvi
5 rows affected.
```

Out[31]:

index	SepalLength	SepalWidth	PetalLength	PetalWidth	Name
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Limiting Results

In [18]:
```
%%sql
SELECT SepalLength FROM Iris_N LIMIT 5
```

```
 * mysql://root:***@localhost/chinook
5 rows affected.
```

Out[18]:

SepalLength
5.1
4.9
4.7
4.6
5.0

Creating New Table and Writing on it

In [32]:
```
%%sql
CREATE TABLE  shoes1(
    Id char(10)  PRIMARY KEY,
    Brand char(10) NOT NULL,
    Type char(250) NOT NULL,
    Color char(250) NOT NULL,
    Price decimal(8,2) NOT NULL,
    Reviews varchar(35) NOT NULL
    )
```

```
 * mysql://root:***@localhost/Aashvi
0 rows affected.
```

Out[32]:
```
[]
```

Adding data into table

In [42]:
```
%%sql
INSERT INTO shoes1
VALUES('14535974','Gucci','Slippers','Pink','695.00','I like it.')
```

In [44]:
```
%%sql
SELECT * FROM shoes1
```

```
 * mysql://root:***@localhost/Aashvi
1 rows affected.
```

Out[44]:

Id	Brand	Type	Color	Price	Reviews
14535974	Gucci	Slippers	Pink	695.00	I like it.

In [38]:
```
%%sql
CREATE TABLE shoes2(
    Id        char(10)       PRIMARY KEY,
    Brand     char(10)       NOT NULL,
    Type      char(250)      NOT NULL,
    Color     char(250)      NOT NULL,
    Price     decimal(8,2)   NOT NULL,
    Reviews   varchar(35)    NOT NULL
    )
```
```
 * mysql://root:***@localhost/Aashvi
0 rows affected.
```
Out[38]:
```
[]
```

In [39]:
```
%%sql
SELECT Brand FROM shoes2
```
```
 * mysql://root:***@localhost/Aashvi
0 rows affected.
```
Out[39]:

Brand

In [43]:
```
%%sql
INSERT INTO shoes2
VALUES('1','GUCCI','slippers','pink','650.30', 'I hate it.')
```
```
 * mysql://root:***@localhost/Aashvi
1 rows affected.
```
Out[43]:
```
[]
```

In [45]:
```
%%sql
SELECT * FROM shoes2
```
```
 * mysql://root:***@localhost/Aashvi
1 rows affected.
```
Out[45]:

Id	Brand	Type	Color	Price	Reviews
1	GUCCI	slippers	pink	650.30	I hate it.

In [46]:
```
%%sql
INSERT INTO shoes1
(Id,Brand,Type,Color,Price,Reviews)
VALUES('14535984','Nike','Slippers','Blue','695.00','I didnt like it')
```
```
 * mysql://root:***@localhost/Aashvi
1 rows affected.
```
Out[46]:
```
[]
```

In [ ]:
```
%%sql
INSERT INTO shoes2
(Id,Brand,Type,Color,Price,Reviews)
VALUES('2','r','ty','co','60','I like it')
```

```
In [ ]:  %%sql
         SELECT * FROM shoes2
```

Trying to insert row with Duplicate ID. This is not allowed and there will be an error.

```
In [48]: %%sql
         INSERT INTO shoes1
         (Id,Brand,Type,Color,Price,Reviews)
         VALUES('14535984','Gucci','Slippers','Pink','695.00','I didnt like it')
```

Inserting multiple rows at once

```
In [49]: %%sql
         INSERT INTO shoes1
         (Id,Brand,Type,Color,Price,Reviews)
         VALUES('14535985','Nike','Slippers','Blue','678.00','I like it'),
         ('14535986','Goldstar','Shoes','Black','695.00','I didnt like it'),
         ('14535987','Nike','Slippers','Blue','800.00','Ok Ok'),
         ('14535988','Puma','Slippers','Blue','870.00','Nice to have one');

          * mysql://root:***@localhost/Aashvi
         4 rows affected.

Out[49]: []
```

```
In [50]: %%sql
         SELECT * FROM shoes1

          * mysql://root:***@localhost/Aashvi
         6 rows affected.
```

Out[50]:

Id	Brand	Type	Color	Price	Reviews
14535974	Gucci	Slippers	Pink	695.00	I like it.
14535984	Nike	Slippers	Blue	695.00	I didnt like it
14535985	Nike	Slippers	Blue	678.00	I like it
14535986	Goldstar	Shoes	Black	695.00	I didnt like it
14535987	Nike	Slippers	Blue	800.00	Ok Ok
14535988	Puma	Slippers	Blue	870.00	Nice to have one

CREATING TEMPORARY TABLE

In [51]:
```
%%sql
CREATE TEMPORARY TABLE Sandals AS
(SELECT * FROM shoes1 WHERE Color='Blue')
```
```
 * mysql://root:***@localhost/Aashvi
4 rows affected.
```
Out[51]:
```
[]
```

In [52]:
```
%%sql CREATE TEMPORARY TABLE sand AS
(SELECT * FROM shoes1 WHERE NOT Color='Blue')
```
```
 * mysql://root:***@localhost/Aashvi
2 rows affected.
```
Out[52]:
```
[]
```

Now displaying above temperory table named 'Sandals'

In [54]:
```
%%sql
SELECT * FROM Sandals
```
```
 * mysql://root:***@localhost/Aashvi
4 rows affected.
```
Out[54]:

Id	Brand	Type	Color	Price	Reviews
14535984	Nike	Slippers	Blue	695.00	I didnt like it
14535985	Nike	Slippers	Blue	678.00	I like it
14535987	Nike	Slippers	Blue	800.00	Ok Ok
14535988	Puma	Slippers	Blue	870.00	Nice to have one

In [53]:
```
%%sql
SELECT * FROM sand
```
```
 * mysql://root:***@localhost/Aashvi
2 rows affected.
```
Out[53]:

Id	Brand	Type	Color	Price	Reviews
14535974	Gucci	Slippers	Pink	695.00	I like it.
14535986	Goldstar	Shoes	Black	695.00	I didnt like it

In [ ]:
```
%%sql
CREATE TEMPORARY TABLE setosa AS
(SELECT * FROM Iris_N WHERE Name='Iris-setosa')
```

In [ ]:
```
%%sql
SELECT * FROM setosa
```

Updating table and deleting rows and columns

In [55]:
```
%%sql
SELECT * FROM shoes2
```

```
 * mysql://root:***@localhost/Aashvi
1 rows affected.
```

Out[55]:

Id	Brand	Type	Color	Price	Reviews
1	GUCCI	slippers	pink	650.30	I hate it.

Dropping a column from a table

- multiple columns can be dropped by adding more drop commmnad below.

In [56]:
```
%%sql
ALTER TABLE shoes2
DROP COLUMN  Type;
```

```
 * mysql://root:***@localhost/Aashvi
0 rows affected.
```

Out[56]:
```
[]
```

In [57]:
```
%%sql
SELECT * FROM shoes2
```

```
 * mysql://root:***@localhost/Aashvi
1 rows affected.
```

Out[57]:

Id	Brand	Color	Price	Reviews
1	GUCCI	pink	650.30	I hate it.

Deleting Particular row

In [58]:
```
%%sql
DELETE FROM shoes2  WHERE Id='1';
```

```
 * mysql://root:***@localhost/Aashvi
1 rows affected.
```

Out[58]:
```
[]
```

In [59]:
```
%%sql
SELECT * FROM shoes2
```

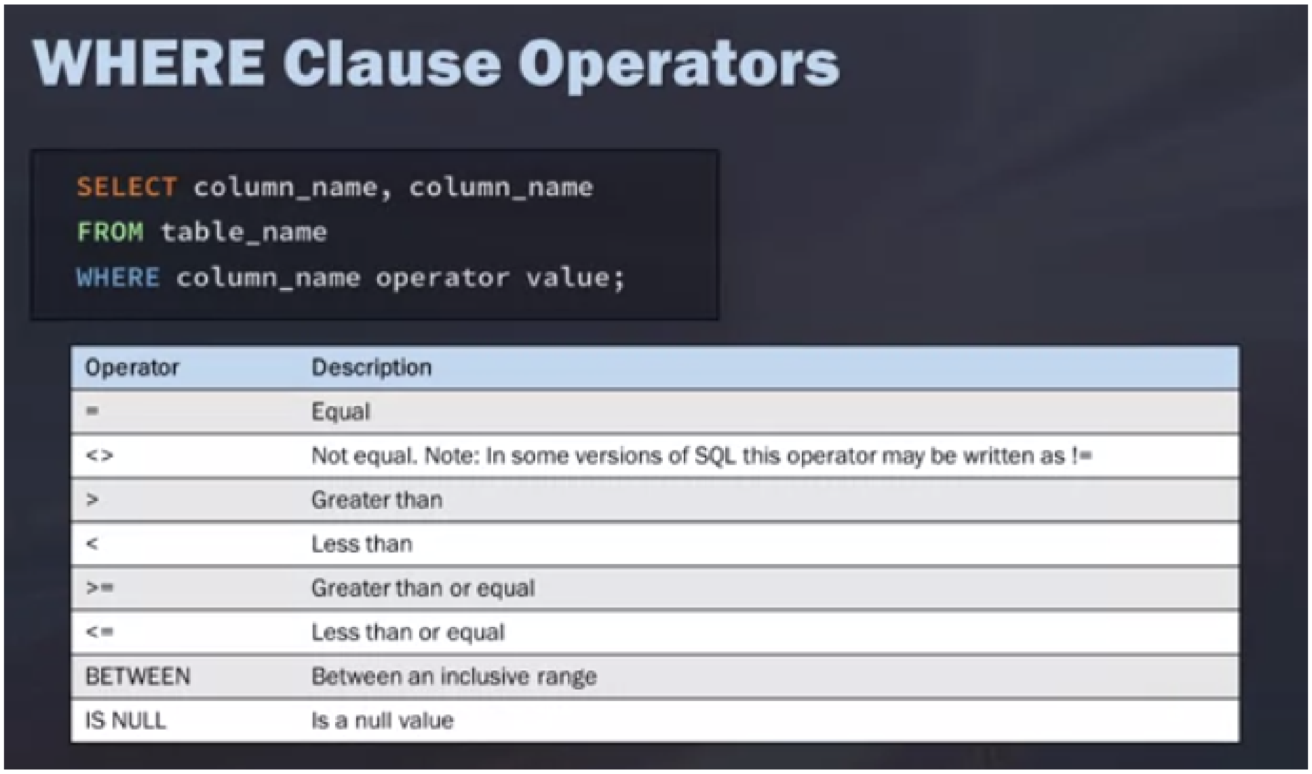
```
 * mysql://root:***@localhost/Aashvi
0 rows affected.
```

Out[59]:

Id	Brand	Color	Price	Reviews

Module 2

Filter Data



WHERE Clause Operators

```
SELECT column_name, column_name
FROM table_name
WHERE column_name operator value;
```

Operator	Description
=	Equal
<>	Not equal. Note: In some versions of SQL this operator may be written as !=
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range
IS NULL	Is a null value

In [67]:
```
%%sql
SELECT * FROM Iris_N WHERE SepalLength >5 # Selecting all column
LIMIT 5
```

```
 * mysql://root:***@localhost/Aashvi
5 rows affected.
```

Out[67]:

index	SepalLength	SepalWidth	PetalLength	PetalWidth	Name
0	5.1	3.5	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
10	5.4	3.7	1.5	0.2	Iris-setosa
14	5.8	4.0	1.2	0.2	Iris-setosa
15	5.7	4.4	1.5	0.4	Iris-setosa

In [68]:
```
%%sql
SELECT * FROM Iris_N WHERE SepalLength<>0.5 LIMIT 8
```

```
 * mysql://root:***@localhost/Aashvi
8 rows affected.
```

Out[68]:

index	SepalLength	SepalWidth	PetalLength	PetalWidth	Name
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa

In [69]:
```
%%sql
SELECT * FROM Iris_N WHERE PetalLength=1.4 LIMIT 5
```

```
 * mysql://root:***@localhost/Aashvi
5 rows affected.
```

Out[69]:

index	SepalLength	SepalWidth	PetalLength	PetalWidth	Name
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa

In [70]:
```
%%sql
SELECT SepalLength,SepalWidth
FROM Iris_N
WHERE Name = 'Iris-versicolor'
LIMIT 5
# Selecting few column
```

```
 * mysql://root:***@localhost/Aashvi
5 rows affected.
```

Out[70]:

SepalLength	SepalWidth
7.0	3.2
6.4	3.2
6.9	3.1
5.5	2.3
6.5	2.8

In [71]:
```
%%sql
SELECT SepalLength,SepalWidth
FROM Iris_N
WHERE SepalLength BETWEEN 6 AND 7
LIMIT 5
# Selecting few column
```

```
 * mysql://root:***@localhost/Aashvi
5 rows affected.
```

Out[71]:

SepalLength	SepalWidth
7.0	3.2
6.4	3.2
6.9	3.1
6.5	2.8
6.3	3.3

In [72]:
```
%%sql
SELECT SepalLength,SepalWidth
FROM Iris_N
WHERE SepalLength <6 OR SepalLength>7
LIMIT 5
```

```
 * mysql://root:***@localhost/Aashvi
5 rows affected.
```

Out[72]:

SepalLength	SepalWidth
5.1	3.5
4.9	3.0
4.7	3.2
4.6	3.1
5.0	3.6

In [73]:
```
%%sql
SELECT SepalLength,SepalWidth
FROM Iris_N
WHERE SepalLength !=5
LIMIT 5
```

```
 * mysql://root:***@localhost/Aashvi
5 rows affected.
```

Out[73]:

SepalLength	SepalWidth
5.1	3.5
4.9	3.0
4.7	3.2
4.6	3.1
5.4	3.9

Advanced Filter

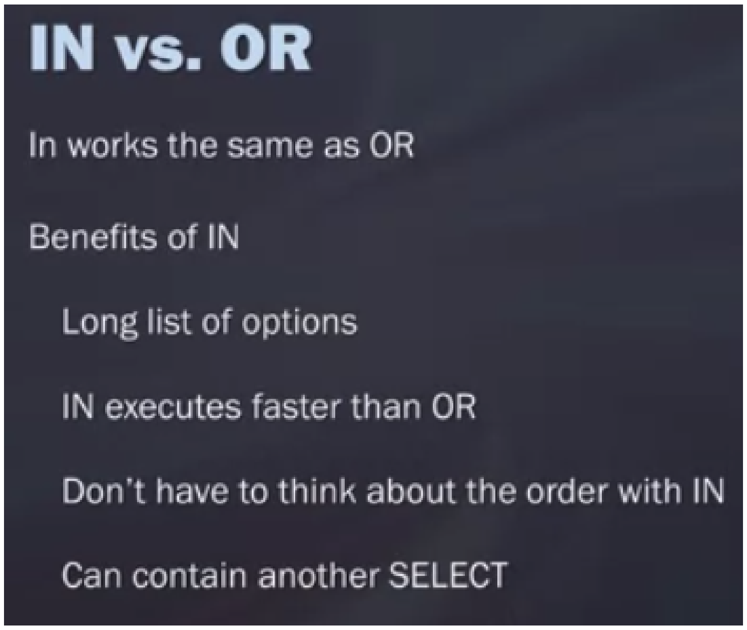
IN <-- OPERATOR

```
In [75]: %%sql
         SELECT SepalLength,PetalLength,Name
         FROM Iris_N
         WHERE Name IN ('Iris-versicolor','Iris-setosa')
         LIMIT 5
```

```
 * mysql://root:***@localhost/Aashvi
5 rows affected.
```

Out[75]:

SepalLength	PetalLength	Name
5.1	1.4	Iris-setosa
4.9	1.4	Iris-setosa
4.7	1.3	Iris-setosa
4.6	1.5	Iris-setosa
5.0	1.4	Iris-setosa


IN vs. OR

In works the same as OR

Benefits of IN

- Long list of options
- IN executes faster than OR
- Don't have to think about the order with IN
- Can contain another SELECT

OR with AND

In [76]:
```
%%sql
SELECT SepalLength,PetalLength, Name
FROM Iris_N
WHERE (SepalLength>6 OR SepalLength=5) AND  PetalLength>1.5
LIMIT 5
## USE Parenthesis for maintain order of operations
```

```
 * mysql://root:***@localhost/Aashvi
5 rows affected.
```

Out[76]:

SepalLength	PetalLength	Name
5.0	1.6	Iris-setosa
5.0	1.6	Iris-setosa
5.0	1.6	Iris-setosa
7.0	4.7	Iris-versicolor
6.4	4.5	Iris-versicolor

NOT oprators

In [78]:
```
%%sql
SELECT * FROM Iris_N
WHERE NOT Name='Iris-setosa' AND NOT Name='Iris-versicolor'
LIMIT 5
```

```
 * mysql://root:***@localhost/Aashvi
5 rows affected.
```

Out[78]:

index	SepalLength	SepalWidth	PetalLength	PetalWidth	Name
100	6.3	3.3	6.0	2.5	Iris-virginica
101	5.8	2.7	5.1	1.9	Iris-virginica
102	7.1	3.0	5.9	2.1	Iris-virginica
103	6.3	2.9	5.6	1.8	Iris-virginica
104	6.5	3.0	5.8	2.2	Iris-virginica

In [79]:
```
%%sql
SELECT SepalLength,SepalWidth, Name
FROM Iris_N
WHERE(Name<>'Iris-setosa' AND Name<>'Iris-vericolor')
LIMIT 5
```

```
 * mysql://root:***@localhost/Aashvi
5 rows affected.
```

Out[79]:

SepalLength	SepalWidth	Name
7.0	3.2	Iris-versicolor
6.4	3.2	Iris-versicolor
6.9	3.1	Iris-versicolor
5.5	2.3	Iris-versicolor
6.5	2.8	Iris-versicolor

USING Wildcaard

- Importing an excel file as pandas dataframe and then converting it into sql table of database.

In [80]:
```
edata=pd.read_excel(r'E:\Machine Learning\Downloaded video\sql_course\Email.xlsx')
edata.head()
```

Out[80]:

	Name	EmailID	mobileNumber	buyDateTime	url	orderID	price
0	Neeraj	neerajrathore27998@gmail.com	9625027500	2020-01-13 18:17:12	gtbit1?productList=1	6345076	700
1	Prayas Jain	p.jain3322@gmail.com	9718495185	2019-10-24 00:21:23	gtbit1?productList=1	6129425	700
2	Mohit	mauryamohit515@gmail.com	9015693974	2019-10-21 13:47:40	gtbit1?productList=1	6120354	700
3	Anshul Garg	04anshulgarg@gmail.com	9711944615	2019-10-23 22:49:27	gtbit1?productList=1	6129204	700
4	Tajinder Kaur	tkaur755@gmail.com	9069336966	2019-10-21 15:23:56	gtbit1?productList=1	6121173	700

In [84]:
```
edata.to_sql('Email',con=engine2)
```

```
In [83]: %%sql
         SELECT * FROM Email
         LIMIT 5
```

```
 * mysql://root:***@localhost/Aashvi
5 rows affected.
```

Out[83]:

index	Name	EmailID	mobileNumber	buyDateTime	url	orderID	pric
0	Neeraj	neerajrathore27998@gmail.com	9625027500	2020-01-13 18:17:12	gtbit1?productList=1	6345076	70
1	Prayas Jain	p.jain3322@gmail.com	9718495185	2019-10-24 00:21:23	gtbit1?productList=1	6129425	70
2	Mohit	mauryamohit515@gmail.com	9015693974	2019-10-21 13:47:40	gtbit1?productList=1	6120354	70
3	Anshul Garg	04anshulgarg@gmail.com	9711944615	2019-10-23 22:49:27	gtbit1?productList=1	6129204	70
4	Tajinder Kaur	tkaur755@gmail.com	9069336966	2019-10-21 15:23:56	gtbit1?productList=1	6121173	70

```
In [85]: %%sql
         SELECT * FROM Email
         WHERE Name LIKE '%Jain'
         LIMIT 5
```

```
 * mysql://root:***@localhost/Aashvi
5 rows affected.
```

Out[85]:

index	Name	EmailID	mobileNumber	buyDateTime	url	orderID	price
1	Prayas Jain	p.jain3322@gmail.com	9718495185	2019-10-24 00:21:23	gtbit1?productList=1	6129425	700
21	Mahima Jain	mahima.170799@gmail.com	8826617418	2019-10-18 22:17:56	gtbit1?productList=1	6113772	700
24	Srishty Jain	jsrishty2000@gmail.com	7291919621	2019-10-19 20:52:26	gtbit1?productList=1	6116192	700
42	Archit Jain	architjain798@gmail.com	8802196620	2019-10-23 14:10:31	gtbit1?productList=1	6127382	700
154	Swaasti Jain	swaastijain@gmail.com	9811761109	2019-10-25 16:08:22	gtbit1?productList=1	6134014	700

In [86]:
```
%%sql
SELECT * FROM Email
WHERE Name LIKE '%Kaur'
LIMIT 5
```

```
 * mysql://root:***@localhost/Aashvi
5 rows affected.
```

Out[86]:

index	Name	EmailID	mobileNumber	buyDateTime	url	orderID
4	Tajinder Kaur	tkaur755@gmail.com	9069336966	2019-10-21 15:23:56	gtbit1?productList=1	6121173
7	Harshleen Kaur	h.kaur.makkar16@gmail.com	9654496288	2019-10-16 20:25:18	gtbit1?productList=1	6106202
9	Dilpreet kaur	dilpreetkaurchawla.2121@gmail.com	9999473745	2019-10-17 15:15:35	gtbit1?productList=1	6109227
40	Simran Preet Kaur	simranpreetk917@gmail.com	9650308453	2019-10-20 21:19:51	gtbit1?productList=1	6118788
60	snehjeet kaur	sneh5119@gmail.com	9205745085	2019-10-21 19:07:30	gtbit1?productList=1	6122065

In [87]:
```
%%sql
SELECT * FROM Email
WHERE Name LIKE 'Rahul%'
LIMIT 5
```

```
 * mysql://root:***@localhost/Aashvi
2 rows affected.
```

Out[87]:

index	Name	EmailID	mobileNumber	buyDateTime	url	orderID	price
26	Rahul Gupta	rrahulgupta098@gmail.com	9315668416	2019-10-22 16:50:50	gtbit1?productList=1	6124913	700
231	Rahul Gupta	rahulgupta20636851@gmail.com	9540222407	2020-03-02 14:35:40	gtbit1?productList=1	6506771	700

In [88]:
```
%%sql
SELECT * FROM Email
WHERE (Name LIKE 'R%') AND (Name NOT LIKE '%Gupta')
LIMIT 7
```

```
 * mysql://root:***@localhost/Aashvi
7 rows affected.
```

Out[88]:

index	Name	EmailID	mobileNumber	buyDateTime	url	orderID	pr
8	Raunaq Kalra	kraunaq58@gmail.com	9996666582	2019-10-16 20:39:53	gtbit1? productList=1	6106259	7
58	Ritika Jindal	rjindal540@gmail.com	8587986643	2020-01-17 11:28:42	gtbit1? productList=1	6357714	7
79	Rajat Sharma	rajat.sharma4211@gmail.com	8826090016	2019-10-21 21:42:22	gtbit1? productList=1	6122659	7
87	Rishabh Sharma	therishabhsharma763@gmail.com	9971601448	2019-10-21 23:03:40	gtbit1? productList=1	6122967	7
97	Rishabh Arora	aroraris habhish@gmail.com	7011529497	2019-10-22 00:25:37	gtbit1? productList=1	6123136	7
108	Rishabh sharma	atrishabh1999@gmail.com	7042461945	2019-10-22 17:06:59	gtbit1? productList=1	6124981	7
179	Rohan	rohantanwar3103@gmail.com	9650726263	2019-11-17 00:00:29	gtbit1? productList=1	6188942	7

Sorting with ORDER BY -> DESC & ASC

In [89]:
```
%%sql
SELECT * FROM Iris_N
ORDER BY  SepalLength ASC# ACS for ascending order(Default), DESC for descending
LIMIT 5
```

```
 * mysql://root:***@localhost/Aashvi
5 rows affected.
```

Out[89]:

index	SepalLength	SepalWidth	PetalLength	PetalWidth	Name
13	4.3	3.0	1.1	0.1	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
38	4.4	3.0	1.3	0.2	Iris-setosa
42	4.4	3.2	1.3	0.2	Iris-setosa
41	4.5	2.3	1.3	0.3	Iris-setosa

In [90]:
```
%%sql
CREATE TEMPORARY TABLE Iris_tmp AS
SELECT * FROM Iris_N
WHERE  SepalLength > 6
ORDER BY SepalLength DESC
```

```
 * mysql://root:***@localhost/Aashvi
5 rows affected.
```

Out[90]:
```
[]
```

In [91]:
```
%%sql
SELECT * FROM Iris_tmp
```

```
 * mysql://root:***@localhost/Aashvi
5 rows affected.
```

Out[91]:

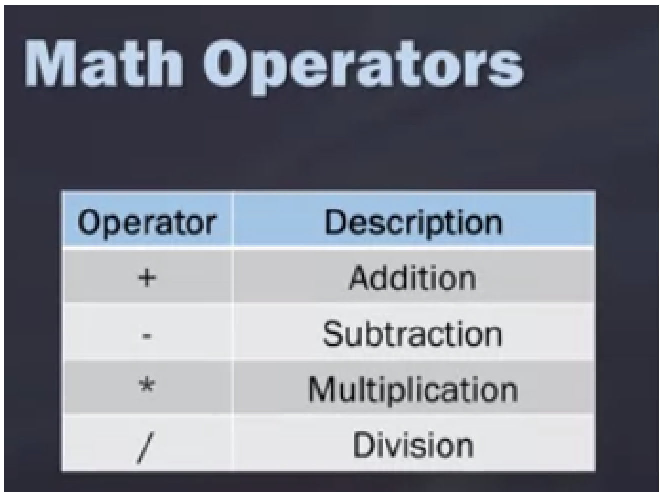
index	SepalLength	SepalWidth	PetalLength	PetalWidth	Name
131	7.9	3.8	6.4	2.0	Iris-virginica
117	7.7	3.8	6.7	2.2	Iris-virginica
118	7.7	2.6	6.9	2.3	Iris-virginica
122	7.7	2.8	6.7	2.0	Iris-virginica
135	7.7	3.0	6.1	2.3	Iris-virginica

In [92]:
```
%%sql
DROP TABLE Iris_tmp
```

```
 * mysql://root:***@localhost/Aashvi
0 rows affected.
```

Out[92]:
```
[]
```

Math operations



Math Operators

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division

In [93]:
```
%%sql
SELECT SepalLength,SepalWidth, SepalLength*SepalWidth AS Cros_feature1
FROM Iris_N
ORDER BY SepalWidth DESC
LIMIT 5
```

```
 * mysql://root:***@localhost/Aashvi
5 rows affected.
```

Out[93]:

SepalLength	SepalWidth	Cros_feature1
5.7	4.4	25.080000000000002
5.5	4.2	23.1
5.2	4.1	21.32
5.8	4.0	23.2
5.4	3.9	21.060000000000002

In [94]:
```
%%sql
SELECT SepalLength,SepalWidth, (SepalLength*SepalWidth)/PetalLength AS Cros_feature
2
FROM Iris_N
LIMIT 5
```

```
 * mysql://root:***@localhost/Aashvi
5 rows affected.
```

Out[94]:

SepalLength	SepalWidth	Cros_feature2
5.1	3.5	12.75
4.9	3.0	10.500000000000002
4.7	3.2	11.569230769230769
4.6	3.1	9.506666666666666
5.0	3.6	12.857142857142858

Check this one

In [95]:
```
%%sql
CREATE TEMPORARY TABLE Temp AS
SELECT SepalLength,PetalWidth,SepalLength-PetalWidth AS New_Feat
FROM Iris_N
WHERE SepalLength >6
ORDER BY PetalWidth DESC
```

```
 * mysql://root:***@localhost/Aashvi
6 rows affected.
```

Out[95]:
```
[]
```

In [96]:
```
%%sql
SELECT * FROM Temp
LIMIT 8
```

```
 * mysql://root:***@localhost/Aashvi
6 rows affected.
```

Out[96]:

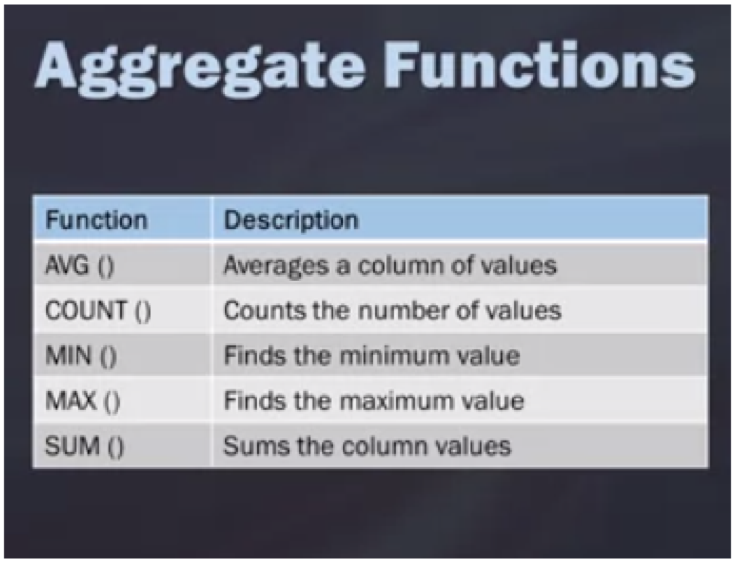
SepalLength	PetalWidth	New_Feat
7.2	2.5	4.7
6.3	2.5	3.8
6.7	2.5	4.2
6.3	2.4	3.9
6.7	2.4	4.300000000000001
6.9	2.3	4.6000000000000005

In [97]:
```
%%sql
DROP TABLE Temp
```

```
 * mysql://root:***@localhost/Aashvi
0 rows affected.
```

Out[97]:
```
[]
```

Aggregate Functions



Aggregate Functions

Function	Description
AVG ()	Averages a column of values
COUNT ()	Counts the number of values
MIN ()	Finds the minimum value
MAX ()	Finds the maximum value
SUM ()	Sums the column values

In [99]:
```
%%sql
SELECT AVG(SepalLength) AS avgspl FROM Iris_N
```

```
 * mysql://root:***@localhost/Aashvi
1 rows affected.
```

Out[99]:

avgspl
5.843333333333335

In [100]:
```
%%sql
SELECT COUNT(*) AS total_entry FROM Iris_N
##counts all rows in the table containing values or null values
```

```
 * mysql://root:***@localhost/Aashvi
1 rows affected.
```

Out[100]:

total_entry
150

In [101]:
```
%%sql
SELECT COUNT(SepalLength) AS total_spl FROM Iris_N
## counts all the rows in a specifica column ignoring NULL values
```

```
 * mysql://root:***@localhost/Aashvi
1 rows affected.
```

Out[101]:

total_spl
150

In [102]:
```
%%sql
SELECT MAX(SepalLength) AS maxspl FROM Iris_N
```

```
 * mysql://root:***@localhost/Aashvi
1 rows affected.
```

Out[102]:

maxspl
7.9

In [103]:
```
%%sql
SELECT MAX(SepalLength) AS maxspl, MIN(SepalLength) AS minspl FROM Iris_N
##Null values are ignore
```

```
 * mysql://root:***@localhost/Aashvi
1 rows affected.
```

Out[103]:

maxspl	minspl
7.9	4.3

In [104]:
```
%%sql
SELECT SUM((SepalLength)) AS sumspl FROM Iris_N
```

```
 * mysql://root:***@localhost/Aashvi
1 rows affected.
```

Out[104]:

sumspl
876.5000000000002

In [105]:
```
%%sql
SELECT SUM(SepalLength*SepalWidth) as prodsplw
FROM Iris_N
where (SepalLength)>6
```

```
 * mysql://root:***@localhost/Aashvi
1 rows affected.
```

Out[105]:

prodsplw
1219.0399999999997

In [106]:
```
%%sql
SELECT COUNT(DISTINCT SepalLength) FROM Iris_N
```

```
 * mysql://root:***@localhost/Aashvi
1 rows affected.
```

Out[106]:

COUNT(DISTINCT SepalLength)
35

In [107]:
```
%%sql
SELECT SUM(DISTINCT SepalLength) FROM Iris_N
```

```
 * mysql://root:***@localhost/Aashvi
1 rows affected.
```

Out[107]:

SUM(DISTINCT SepalLength)
210.3999999999999

Grouping Data with SQL

- Importing an excel file as pandas dataframe and then converting it into sql table of database.

In [109]:
```
tips=pd.read_csv('E:/Machine Learning/Downloaded video/sql_course/tips.csv')
tips.head()
```

Out[109]:

	total_bill	tip	smoker	day	time	size
0	16.99	1.01	No	Sun	Dinner	2
1	10.34	1.66	No	Sun	Dinner	3
2	21.01	3.50	No	Sun	Dinner	3
3	23.68	3.31	No	Sun	Dinner	2
4	24.59	3.61	No	Sun	Dinner	4

In [110]:
```
tips.to_sql('tips',con=engine2)
```

```
In [111]: %%sql
          SELECT smoker, SUM(total_bill) AS total_collection
          FROM tips
          GROUP BY smoker
```

```
 * mysql://root:***@localhost/Aashvi
2 rows affected.
```

Out[111]:

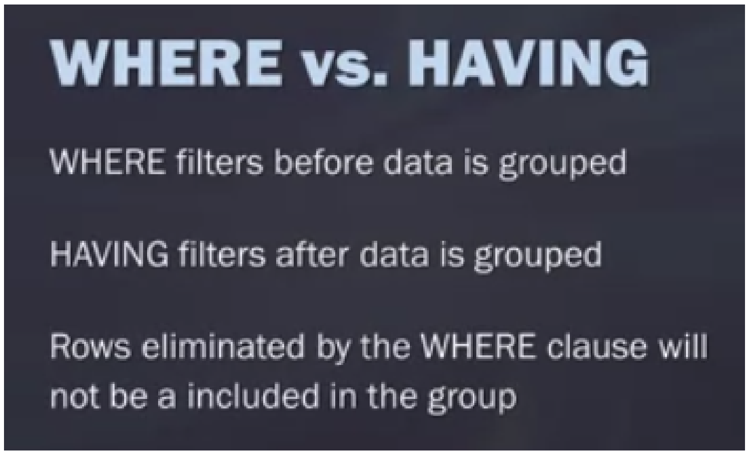
smoker	total_collection
No	2897.430000000001
Yes	1930.3400000000001

```
In [112]: %%sql
          SELECT day, SUM(total_bill) AS total_collection
          FROM tips
          GROUP BY day
```

```
 * mysql://root:***@localhost/Aashvi
4 rows affected.
```

Out[112]:

day	total_collection
Fri	325.87999999999994
Sat	1778.3999999999996
Sun	1627.1600000000003
Thur	1096.3299999999997



WHERE vs. HAVING

WHERE filters before data is grouped

HAVING filters after data is grouped

Rows eliminated by the WHERE clause will not be a included in the group

In [113]:
```
%%sql
SELECT day, SUM(total_bill) AS total_collection
FROM tips
WHERE size>3
GROUP BY day
HAVING total_collection>41
ORDER BY day DESC
```

```
 * mysql://root:***@localhost/Aashvi
3 rows affected.
```

Out[113]:

day	total_collection
Thur	282.09000000000003
Sun	609.5600000000001
Sat	416.5399999999999

In [114]:
```
%%sql
SELECT time,day,SUM(tip) AS total_tip
FROM tips
WHERE size>4
GROUP BY day
HAVING day <> 'Sun'
```

```
 * mysql://root:***@localhost/Aashvi
2 rows affected.
```

Out[114]:

time	day	total_tip
Dinner	Sat	3.0
Lunch	Thur	20.9

Module 3 : Joinings

Joins in Detail with visible example: Pandas Syntx & Equivalent SQL

A. INNER JOIN

1. Basic Example with one unique key combination:

- one-to-one joins: for example when joining two DataFrame objects on their indexes (which must contain unique values).

In [35]:
```
LEFT1 = pd.DataFrame({'key': ['K0', 'K1', 'K2', 'K3'],
                      'A': ['A0', 'A1', 'A2', 'A3'],
                      'B': ['B0', 'B1', 'B2', 'B3']})

RIGHT1 = pd.DataFrame({'key': ['K0', 'K1', 'K2', 'K3'],
                       'C': ['C0', 'C1', 'C2', 'C3'],
                       'D': ['D0', 'D1', 'D2', 'D3']})

result = pd.merge(LEFT1, RIGHT1, on='key')

print(LEFT1,'\n\n',RIGHT1)
result
```

```
  key   A   B
0  K0  A0  B0
1  K1  A1  B1
2  K2  A2  B2
3  K3  A3  B3

   key   C   D
0  K0  C0  D0
1  K1  C1  D1
2  K2  C2  D2
3  K3  C3  D3
```

Out[35]:
```
   key  A   B   C   D
0  K0   A0  B0  C0  D0
1  K1   A1  B1  C1  D1
2  K2   A2  B2  C2  D2
3  K3   A3  B3  C3  D3
```

Equivalent SQL implementation

In [38]:
```
## Making a datafrme into sql table and storing in database  then working on it
LEFT1.to_sql('left1',con=engine2)
RIGHT1.to_sql('right1',con=engine2)
```

In [40]:
```
%%sql
SELECT * FROM left1
```

```
 * mysql://root:***@localhost/chinook
4 rows affected.
```

Out[40]:
```
index  key  A   B
    0  K0   A0  B0
    1  K1   A1  B1
    2  K2   A2  B2
    3  K3   A3  B3
```

```
In [41]: %%sql
         SELECT * FROM right1
```

```
 * mysql://root:***@localhost/chinook
4 rows affected.
```

Out[41]:

index	key	C	D
0	K0	C0	D0
1	K1	C1	D1
2	K2	C2	D2
3	K3	C3	D3

```
In [46]: %%sql
         SELECT left1.key,A,B
         FROM left1
         JOIN right1
         ON left1.key=right1.key
         /*,Cmment line --> or, we can write : INNER JOIN */
```

```
 * mysql://root:***@localhost/chinook
4 rows affected.
```

Out[46]:

key	A	B
K0	A0	B0
K1	A1	B1
K2	A2	B2
K3	A3	B3

2. Complicated Example with multiple join keys. Only the keys appearing in both left and right are present (the intersection), since how='inner' by default (in pandas).

In [47]:
```
LEFT2 = pd.DataFrame({'key1': ['K0', 'K0', 'K1', 'K2'],
                     'key2': ['K0', 'K1', 'K0', 'K1'],
                     'A': ['A0', 'A1', 'A2', 'A3'],
                     'B': ['B0', 'B1', 'B2', 'B3']})

RIGHT2 = pd.DataFrame({'key1': ['K0', 'K1', 'K1', 'K2'],
                      'key2': ['K0', 'K0', 'K0', 'K0'],
                      'C': ['C0', 'C1', 'C2', 'C3'],
                      'D': ['D0', 'D1', 'D2', 'D3']})

result = pd.merge(LEFT2, RIGHT2, on=['key1', 'key2'],how='inner') ##intersection --
> default how='inner'
print(LEFT2,'\n\n',RIGHT2)
result
```

```
  key1 key2   A   B
0   K0   K0  A0  B0
1   K0   K1  A1  B1
2   K1   K0  A2  B2
3   K2   K1  A3  B3

   key1 key2   C   D
0   K0   K0  C0  D0
1   K1   K0  C1  D1
2   K1   K0  C2  D2
3   K2   K0  C3  D3
```

Out[47]:

	key1	key2	A	B	C	D
0	K0	K0	A0	B0	C0	D0
1	K1	K0	A2	B2	C1	D1
2	K1	K0	A2	B2	C2	D2

Equivalent SQL implementation

In [48]:
```
### Making a datafrme into sql table and storing in database  then working on it
LEFT2.to_sql('left2',con=engine2)
RIGHT2.to_sql('right2',con=engine2)
```

In [49]:
```
%%sql
SELECT * FROM left2
```

```
 * mysql://root:***@localhost/chinook
4 rows affected.
```

Out[49]:

index	key1	key2	A	B
0	K0	K0	A0	B0
1	K0	K1	A1	B1
2	K1	K0	A2	B2
3	K2	K1	A3	B3

In [50]:
```
%%sql
SELECT * FROM right2
```

```
 * mysql://root:***@localhost/chinook
4 rows affected.
```

Out[50]:

index	key1	key2	C	D
0	K0	K0	C0	D0
1	K1	K0	C1	D1
2	K1	K0	C2	D2
3	K2	K0	C3	D3

In [51]:
```
%%sql
SELECT left2.key1,left2.key2,A,B,C,D
FROM left2
JOIN right2
ON (left2.key1=right2.key1) AND (left2.key2=right2.key2)
```

```
 * mysql://root:***@localhost/chinook
3 rows affected.
```

Out[51]:

key1	key2	A	B	C	D
K0	K0	A0	B0	C0	D0
K1	K0	A2	B2	C1	D1
K1	K0	A2	B2	C2	D2

Note: In above Example and examples given below, we are joing on two field (on both :key1 & key2) thus 'AND' operator has been used. For joining on one key, AND part will not be there.

B. Example > how='left' /left join

- Using same above two table i.e. left2 and right2

In [53]:
```
result = pd.merge(LEFT2, RIGHT2, how='left', on=['key1', 'key2']) ##left>A,right>B
print(LEFT2,'\n\n',RIGHT2)
result
```

```
  key1 key2   A   B
0   K0   K0  A0  B0
1   K0   K1  A1  B1
2   K1   K0  A2  B2
3   K2   K1  A3  B3

   key1 key2   C   D
0   K0   K0  C0  D0
1   K1   K0  C1  D1
2   K1   K0  C2  D2
3   K2   K0  C3  D3
```

Out[53]:

	key1	key2	A	B	C	D
0	K0	K0	A0	B0	C0	D0
1	K0	K1	A1	B1	NaN	NaN
2	K1	K0	A2	B2	C1	D1
3	K1	K0	A2	B2	C2	D2
4	K2	K1	A3	B3	NaN	NaN

Equivalent SQL implementation

In [57]:
```
%%sql
SELECT left2.key1,left2.key2,A,B,C,D
FROM left2
LEFT JOIN right2
ON (left2.key1=right2.key1) AND (left2.key2=right2.key2)
ORDER BY key1
```

```
 * mysql://root:***@localhost/chinook
5 rows affected.
```

Out[57]:

key1	key2	A	B	C	D
K0	K0	A0	B0	C0	D0
K0	K1	A1	B1	None	None
K1	K0	A2	B2	C1	D1
K1	K0	A2	B2	C2	D2
K2	K1	A3	B3	None	None

C. Example > how='right'/right join

In [58]:
```
result = pd.merge(LEFT2, RIGHT2, how='right', on=['key1', 'key2'])##left>A,right>B
print(LEFT2,'\n\n',RIGHT2)
result
```

```
  key1 key2   A   B
0   K0   K0  A0  B0
1   K0   K1  A1  B1
2   K1   K0  A2  B2
3   K2   K1  A3  B3

    key1 key2   C   D
0   K0   K0  C0  D0
1   K1   K0  C1  D1
2   K1   K0  C2  D2
3   K2   K0  C3  D3
```

Out[58]:

	key1	key2	A	B	C	D
0	K0	K0	A0	B0	C0	D0
1	K1	K0	A2	B2	C1	D1
2	K1	K0	A2	B2	C2	D2
3	K2	K0	NaN	NaN	C3	D3

Equivalent SQL implementation

In [60]:
```
%%sql
SELECT right2.key1,right2.key2,A,B,C,D
FROM left2
RIGHT JOIN right2
ON (left2.key1=right2.key1) AND (left2.key2=right2.key2)
ORDER BY key1
```

```
 * mysql://root:***@localhost/chinook
4 rows affected.
```

Out[60]:

key1	key2	A	B	C	D
K0	K0	A0	B0	C0	D0
K1	K0	A2	B2	C1	D1
K1	K0	A2	B2	C2	D2
K2	K0	None	None	C3	D3

D. Example > how='outer'/outer join or Union operation

In [61]:
```
result = pd.merge(LEFT2, RIGHT2, how='outer', on=['key1', 'key2'])
print(LEFT2,'\n\n',RIGHT2)
result
```

```
  key1 key2   A   B
0   K0   K0  A0  B0
1   K0   K1  A1  B1
2   K1   K0  A2  B2
3   K2   K1  A3  B3

   key1 key2   C   D
0   K0   K0  C0  D0
1   K1   K0  C1  D1
2   K1   K0  C2  D2
3   K2   K0  C3  D3
```

Out[61]:

	key1	key2	A	B	C	D
0	K0	K0	A0	B0	C0	D0
1	K0	K1	A1	B1	NaN	NaN
2	K1	K0	A2	B2	C1	D1
3	K1	K0	A2	B2	C2	D2
4	K2	K1	A3	B3	NaN	NaN
5	K2	K0	NaN	NaN	C3	D3

Equivalent SQL implementation

- Since here is two field on which we have to join Union is used to get full outer join.

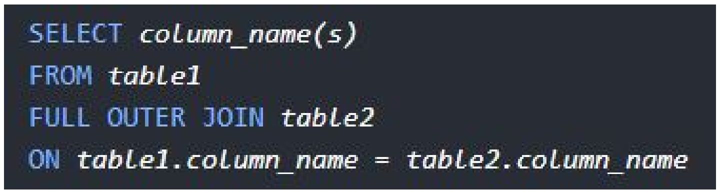
In [82]:
```
%%sql
SELECT left2.key1,left2.key2,A,B,C,D
FROM left2
LEFT JOIN right2
ON (left2.key1=right2.key1) AND (left2.key2=right2.key2)
UNION
SELECT right2.key1,right2.key2,A,B,C,D
FROM left2
RIGHT JOIN right2
ON (left2.key1=right2.key1) AND (left2.key2=right2.key2)
ORDER BY key1
```

```
 * mysql://root:***@localhost/chinook
6 rows affected.
```

Out[82]:

key1	key2	A	B	C	D
K0	K0	A0	B0	C0	D0
K0	K1	A1	B1	None	None
K1	K0	A2	B2	C1	D1
K1	K0	A2	B2	C2	D2
K2	K1	A3	B3	None	None
K2	K0	None	None	C3	D3

For joining on one field only, following syntax can be used :



```
SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2
ON table1.column_name = table2.column_name
```

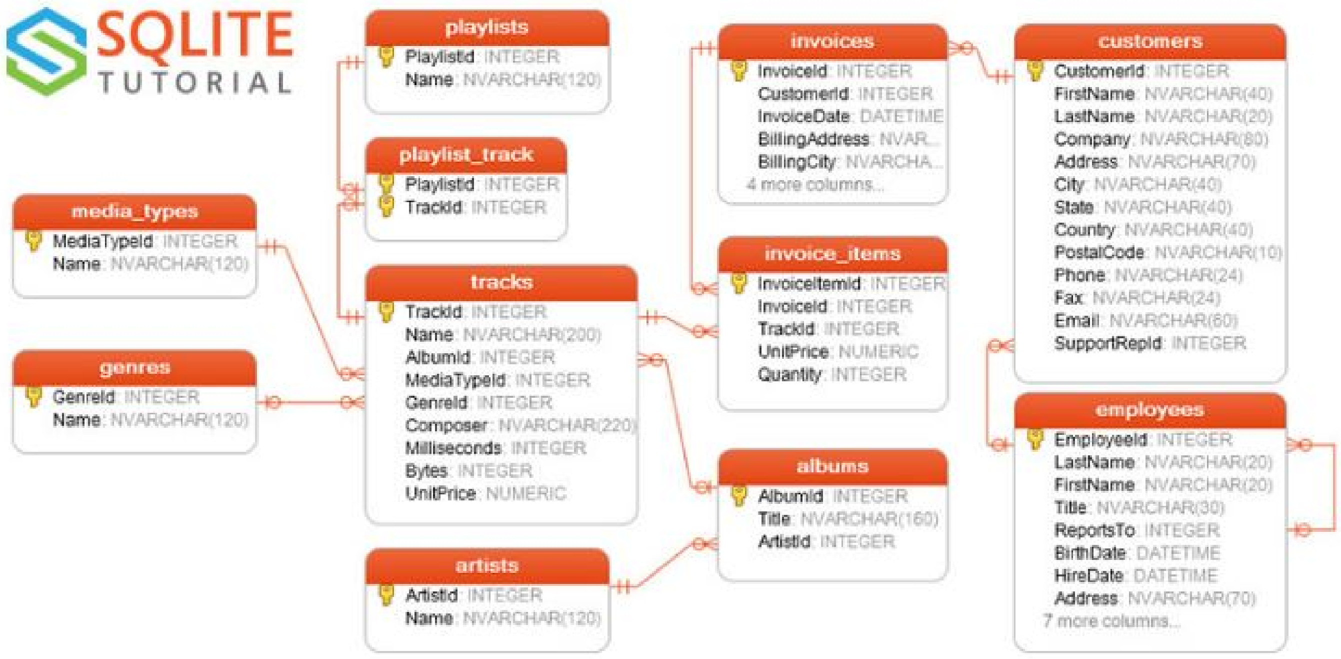
Module 4 Practice on Chinook database : Self join, Many more Different Scenarios

For this we have to first upload chinook database to our mysql server.

Connecting to Database

In [1]:
```
dialect+driver://username:password@host:port/database
engine1=sqlalchemy.create_engine('mysql+mysqldb://root:r1m2n3#?@localhost/chinook')
engine1.connect()
print(engine1)
```

Chinook Shema > Image Ref: sqlitetutorial.net



Q.1. Provide a query showing Customers (just their full names, customer ID and country) who are not in the US

In [ ]:
```
%%sql
SELECT FirstName,LastName,CustomerId,Country
FROM chinook.customer
WHERE Country<>'USA'
```

In [ ]:
```
## Check below syntax for concatenation of two string columns
```

In [ ]:
```
%%sql
SELECT CONCAT(FirstName,' ', LastName) AS Name ,CustomerId,Country
FROM chinook.customer
WHERE Country <>'USA';
```

Q.2.Provide a query only showing the Customers from Brazil.

In [ ]:
```
%%sql
SELECT FirstName,Country
FROM chinook.customer
WHERE Country ='Brazil'
```

Q.3. Provide a query showing the Invoices of customers who are from Brazil. The resultant table should show the customer's full name, Invoice ID, Date of the invoice and billing country.

In [ ]:
```
%%sql
SELECT invoice.CustomerID, CONCAT(FirstName,' ', LastName) AS Name,InvoiceId,Invoic
eDate,Country
FROM invoice
JOIN customer
ON customer.CustomerId=invoice.CustomerID
Where Country ='Brazil'
```

Q.4. Provide a query showing only the Employees who are Sales Agents.

In [ ]:
```
%%sql
SELECT * FROM chinook.employee limit 4
```

In [ ]:
```
%%sql
SELECT CONCAT(FirstName ,' ',LastName) AS Name
FROM chinook.employee
WHERE Title LIKE '%Sales%Agent%'
```

- Q.5.Provide a query showing a unique list of billing countries from the Invoice table.

In [ ]:
```
%%sql
SELECT DISTINCT BillingCountry
FROM invoice
```

Self Join :

- Find the Mangers of the empoyees

In [ ]:
```
 /*,e.EmployeeId,e.FirstName,m.FirstName AS Managers_Name*/  --> For commenting
```

In [ ]:
```
%%sql
SELECT e.EmployeeId,e.FirstName,m.FirstName AS Managers_Name
FROM employee e
JOIN employee m
on e.ReportsTo=m.EmployeeID
```

Joining Mutiple tables more than 2

Q.6. Provide a query that shows the invoices associated with each sales agent. The resultant table should include the Sales Agent's full name.

In [ ]:
```
%%sql
SELECT InvoiceId, CONCAT(employee.FirstName," ",employee.LastName) AS Name, Title
FROM customer
JOIN invoice ON invoice.CustomerId=customer.CustomerId
JOIN employee ON customer.SupportRepId=employee.EmployeeId;
```

In [ ]:
```
%%sql
SELECT InvoiceId, CONCAT(employee.FirstName," ",employee.LastName) AS Name, Title
FROM invoice
JOIN customer ON invoice.CustomerId=customer.CustomerId
JOIN employee ON customer.SupportRepId=employee.EmployeeId
```

This much for this module.

Feel Free to Share and Distribute.¶

Don't forget to follow me for more such stuff.

https://github.com/Rami-RK (https://github.com/Rami-RK)

https://www.linkedin.com/in/ramendra-kumar-57334478 (https://www.linkedin.com/in/ramendra-kumar-57334478)

Thank You !