README

This document contains SQL query by python and some use cases of SQL. The Database is PostgreSQL DB. All the test, codes are included in the following screenshots.

At first, read the csv data, import it into Jupyter notebook.

In [2]:
```
#import PostgreSQL adapter,computing package and data analysis tool
import psycopg2
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import sys

#loading data from lcoal
data = pd.DataFrame(pd.read_csv('/Users/chen/Desktop/INFO6210-DMDB/games.csv', encoding='latin1'))
data
```

Out[2]:

	Name	Platform	Year_of_Release	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales	Critic_Score	Critic_Cou
0	Wii Sports	Wii	2006.0	Sports	Nintendo	41.36	28.96	3.77	8.45	82.53	76.0	5
1	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58	6.81	0.77	40.24	NaN	N
2	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.68	12.76	3.79	3.29	35.52	82.0	7
3	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.61	10.93	3.28	2.95	32.77	80.0	7
4	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	Nintendo	11.27	8.89	10.22	1.00	31.37	NaN	N
5	Tetris	GB	1989.0	Puzzle	Nintendo	23.20	2.26	4.22	0.58	30.26	NaN	N
6	New Super Mario Bros.	DS	2006.0	Platform	Nintendo	11.28	9.14	6.50	2.88	29.80	89.0	6
7	Wii Play	Wii	2006.0	Misc	Nintendo	13.96	9.18	2.93	2.84	28.92	58.0	4
8	New Super Mario Bros. Wii	Wii	2009.0	Platform	Nintendo	14.44	6.94	4.70	2.24	28.32	87.0	8
9	Duck Hunt	NES	1984.0	Shooter	Nintendo	26.93	0.63	0.28	0.47	28.31	NaN	N
10	Nintendogs	DS	2005.0	Simulation	Nintendo	9.05	10.95	1.93	2.74	24.67	NaN	N
11	Mario Kart DS	DS	2005.0	Racing	Nintendo	9.71	7.47	4.13	1.90	23.21	91.0	6
12	Pokemon Gold/Pokemon Silver	GB	1999.0	Role-Playing	Nintendo	9.00	6.18	7.20	0.71	23.10	NaN	N
13	Wii Fit	Wii	2007.0	Sports	Nintendo	8.92	8.03	3.60	2.15	22.70	80.0	6
14	Kinect Adventures!	X360	2010.0	Misc	Microsoft Game Studios	15.00	4.89	0.24	1.69	21.81	61.0	4
15	Wii Fit Plus	Wii	2009.0	Sports	Nintendo	9.01	8.49	2.53	1.77	21.79	80.0	3

Then see the data describe and check the duplicate and the null of data, use drop function to drop the duplicated rows. Because the null data in some columns more than 5000, and these values has no effect in the sales data, so don’t need drop it.

In [39]:
```
data.describe()
```

Out[39]:

	Year_of_Release	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales	Critic_Score	Critic_Count	User_Count
count	16450.000000	16719.000000	16719.000000	16719.000000	16719.000000	16719.000000	8137.000000	8137.000000	7590.000000
mean	2006.487356	0.263330	0.145025	0.077602	0.047332	0.533543	68.967679	26.360821	162.229908
std	5.878995	0.813514	0.503283	0.308818	0.186710	1.547935	13.938165	18.980495	561.282326
min	1980.000000	0.000000	0.000000	0.000000	0.000000	0.010000	13.000000	3.000000	4.000000
25%	2003.000000	0.000000	0.000000	0.000000	0.000000	0.060000	60.000000	12.000000	10.000000
50%	2007.000000	0.080000	0.020000	0.000000	0.010000	0.170000	71.000000	21.000000	24.000000
75%	2010.000000	0.240000	0.110000	0.040000	0.030000	0.470000	79.000000	36.000000	81.000000
max	2020.000000	41.360000	28.960000	10.220000	10.570000	82.530000	98.000000	113.000000	10665.000000

In [40]:
```
#check if it has duplicated data
data.duplicated().sum()
```

Out[40]:
```
0
```

In [41]:
```
#check the situation of null in each column
data.isnull().sum()
```

Out[41]:
```
Name                   2
Platform               0
Year_of_Release      269
Genre                  2
Publisher             54
NA_Sales               0
EU_Sales               0
JP_Sales               0
Other_Sales            0
Global_Sales           0
Critic_Score        8582
Critic_Count        8582
User_Score          6704
User_Count          9129
Developer           6623
Rating              6769
dtype: int64
```

In [45]:
```
#drop the duplicated data
ddata.drop_duplicates()
```

Out[45]:

	Name	Platform	Year_of_Release	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales	Critic_Score	Critic_Coun
0	Wii Sports	Wii	2006.0	Sports	Nintendo	41.36	28.96	3.77	8.45	82.53	76.0	51.0
1	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58	6.81	0.77	40.24	NaN	NaN
2	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.68	12.76	3.79	3.29	35.52	82.0	73.0
3	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.61	10.93	3.28	2.95	32.77	80.0	73.0
4	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	Nintendo	11.27	8.89	10.22	1.00	31.37	NaN	NaN
5	Tetris	GB	1989.0	Puzzle	Nintendo	23.20	2.26	4.22	0.58	30.26	NaN	NaN
6	New Super Mario Bros.	DS	2006.0	Platform	Nintendo	11.28	9.14	6.50	2.88	29.80	89.0	65.0
7	Wii Play	Wii	2006.0	Misc	Nintendo	13.96	9.18	2.93	2.84	28.92	58.0	41.0
8	New Super Mario Bros. Wii	Wii	2009.0	Platform	Nintendo	14.44	6.94	4.70	2.24	28.32	87.0	80.0

In [46]:
```
#NAN data check and column information
ddata.isnull().sum()
```

Out[46]:
```
Name                   2
Platform               0
Year_of_Release      269
Genre                  2
Publisher             54
NA_Sales               0
EU_Sales               0
JP_Sales               0
Other_Sales            0
Global_Sales           0
Critic_Score        8582
Critic_Count        8582
User_Score          6704
User_Count          9129
Developer           6623
Rating              6769
dtype: int64
```

Then, connect the PostgreSQL database, and create a new table ‘PCgames’ with the same columns, can also use pd.to_sql to do it. After that, copy the importing data to the ‘PCgames’, then use SQL to display the table.

In [53]:
```
#connect to the postgreSQL Database in Localhost, Create a new table according to previous table and submit
#to excute it, then close the connection
conn = psycopg2.connect(database='postgres',user='postgres', password='')
cur = conn.cursor()
cur.execute('''
    CREATE TABLE PCgames
    (Name varchar, Platform varchar, Year_of_Release varchar, Genre varchar, Publisher varchar,
    NA_Sales real, EU_Sales real, JP_Sales real, Other_Sales real,
    Global_Sales real, Critic_Score varchar, Critic_Count varchar,
    User_Score varchar, User_Count varchar, Developer varchar, Rating varchar)
    ''')
conn.commit()
conn.close()
```

In [54]:
```
#COPY the data in localhost csv file to the postgreSQL Database
conn = psycopg2.connect(database='postgres',user='postgres', password='')
cur = conn.cursor()
cur.execute('''
    COPY PCgames
    FROM '/Users/chen/Desktop/INFO6210-DMDB/games.csv'
    WITH csv HEADER;
''')
conn.commit()
conn.close()
```

In [67]:
```
conn = psycopg2.connect(database='postgres',user='postgres', password='')
cur = conn.cursor()
```

In [69]:
```
#read the Table PCgames in DB and display its head
df=pd.read_sql('''SELECT * FROM PCgames''', con=conn)
df.head()
```

Out[69]:

	name	platform	year_of_release	genre	publisher	na_sales	eu_sales	jp_sales	other_sales	global_sales	critic_score	critic_count	user_score	use
0	Wii Sports	Wii	2006	Sports	Nintendo	41.36	28.96	3.77	8.45	82.53	76	51	8	
1	Super Mario Bros.	NES	1985	Platform	Nintendo	29.08	3.58	6.81	0.77	40.24	None	None	None	
2	Mario Kart Wii	Wii	2008	Racing	Nintendo	15.68	12.76	3.79	3.29	35.52	82	73	8.3	
3	Wii Sports Resort	Wii	2009	Sports	Nintendo	15.61	10.93	3.28	2.95	32.77	80	73	8	
4	Pokemon Red/Pokemon Blue	GB	1996	Role-Playing	Nintendo	11.27	8.89	10.22	1.00	31.37	None	None	None	

Then try insert a row and test if insert successfully, after that, delete the test row from table and test if delete successfully.

In [67]:
```
#insert one row, display it and delete it
cur.execute('''INSERT INTO PCgames values('Test', 'test', 2018, 'test', 'test', 1, 2, 3,
4, 5, '6', '7', '8', '9', 'test', 'Test');''')
conn.commit()
```

In [68]:
```
df_a=pd.read_sql('''SELECT * FROM PCgames WHERE Name = 'Test' ''', con=conn)
df_a
```

Out[68]:

	name	platform	year_of_release	genre	publisher	na_sales	eu_sales	jp_sales	other_sales	global_sales	critic_score	critic_count	user_score	user_count
0	Test	test	2018	test	test	1.0	2.0	3.0	4.0	5.0	6	7	8	9

In [69]:
```
cur.execute('''delete FROM PCgames WHERE Name = 'Test' ''')
```

In [70]:
```
df_b=pd.read_sql('''SELECT * FROM PCgames WHERE Name = 'Test' ''', con=conn)
df_b
```

Out[70]:

	name	platform	year_of_release	genre	publisher	na_sales	eu_sales	jp_sales	other_sales	global_sales	critic_score	critic_count	user_score	user_count	d

Then thinking about normalization. The ‘PCgames’ table has 16 columns:
Name, Platform, Year_of_Release, Genre, Publisher, NA_Sales, EU_Sales, JP_Sales, Other_Sales, Global_Sales, Critic_Score, Critic_Count, User_Score, User_Count, Developer, Rating.
It can be divided to 2 parts:
First is the basic information of PC games, including Name, Platform, Year_of_Release, Genre, Publisher and Developer.
Second is the sales data of PC games, including NA_Sales, EU_Sales, JP_Sales, Other_Sales, Global_Sales, Critic_Score, Critic_Count, User_Score, User_Count, Rating.
Based on it creating 2 tables:

```
In [70]: #normalization, choose columns to create a new table
df_n=pd.read_sql('''SELECT PCgames.name, PCgames.platform, PCgames.year_of_release, PCgames.genre, PCgames.publisher,
PCgames.developer FROM PCgames''', con=conn)
df_n
```

Out[70]:

	name	platform	year_of_release	genre	publisher	developer
0	Wii Sports	Wii	2006	Sports	Nintendo	Nintendo
1	Super Mario Bros.	NES	1985	Platform	Nintendo	None
2	Mario Kart Wii	Wii	2008	Racing	Nintendo	Nintendo
3	Wii Sports Resort	Wii	2009	Sports	Nintendo	Nintendo
4	Pokemon Red/Pokemon Blue	GB	1996	Role-Playing	Nintendo	None
5	Tetris	GB	1989	Puzzle	Nintendo	None
6	New Super Mario Bros.	DS	2006	Platform	Nintendo	Nintendo
7	Wii Play	Wii	2006	Misc	Nintendo	Nintendo
8	New Super Mario Bros. Wii	Wii	2009	Platform	Nintendo	Nintendo
9	Duck Hunt	NES	1984	Shooter	Nintendo	None
10	Nintendogs	DS	2005	Simulation	Nintendo	None
11	Mario Kart DS	DS	2005	Racing	Nintendo	Nintendo
12	Pokemon Gold/Pokemon Silver	GB	1999	Role-Playing	Nintendo	None
13	Wii Fit	Wii	2007	Sports	Nintendo	Nintendo
14	Kinect Adventures!	X360	2010	Misc	Microsoft Game Studios	Good Science Studio
15	Wii Fit Plus	Wii	2009	Sports	Nintendo	Nintendo
16	Grand Theft Auto V	PS3	2013	Action	Take-Two Interactive	Rockstar North
17	Grand Theft Auto: San Andreas	PS2	2004	Action	Take-Two Interactive	Rockstar North
18	Super Mario World	SNES	1990	Platform	Nintendo	None
19	Brain Age: Train Your Brain in Minutes a Day	DS	2005	Misc	Nintendo	Nintendo
20	Pokemon Diamond/Pokemon Pearl	DS	2006	Role-Playing	Nintendo	None
21	Super Mario Land	GB	1989	Platform	Nintendo	None
22	Super Mario Bros. 3	NES	1988	Platform	Nintendo	None
23	Grand Theft Auto V	X360	2013	Action	Take-Two Interactive	Rockstar North
24	Grand Theft Auto: Vice City	PS2	2002	Action	Take-Two Interactive	Rockstar North
25	Pokemon Ruby/Pokemon Sapphire	GBA	2002	Role-Playing	Nintendo	None

Create and Test the first table:

In [81]:
```
#transfer the new table to a csv file and then copy it to the DB
df_n.to_csv('PCgames_attribute.csv', index=False)
```

In [ ]:
```
cur.execute('''
    CREATE TABLE PCgames_attribute
    (Name varchar, Platform varchar, Year_of_Release varchar, Genre varchar, Publisher varchar, Developer varchar)
    ''')
conn.commit()
```

In [79]:
```
conn.close
```

Out[79]:
```
<function connection.close>
```

In [82]:
```
conn = psycopg2.connect(database='postgres',user='postgres', password='')
cur = conn.cursor()
cur.execute('''
    COPY PCgames_attribute
    FROM '/Users/chen/PCgames_attribute.csv'
    WITH csv HEADER;
''')
conn.commit()
```

In [83]:
```
#read and display the first new table, the name is PCgames_attribut
df_m=pd.read_sql('''SELECT * FROM PCgames_attribute''', con=conn)
df_m.head()
```

Out[83]:

	name	platform	year_of_release	genre	publisher	developer
0	Wii Sports	Wii	2006	Sports	Nintendo	Nintendo
1	Super Mario Bros.	NES	1985	Platform	Nintendo	None
2	Mario Kart Wii	Wii	2008	Racing	Nintendo	Nintendo
3	Wii Sports Resort	Wii	2009	Sports	Nintendo	Nintendo
4	Pokemon Red/Pokemon Blue	GB	1996	Role-Playing	Nintendo	None

Test the second table:

In [86]:
```
#choose the rest of Table PCgames to create the second table, call it PCgames_sales, and copy it to the DB
df_p=pd.read_sql('''SELECT Name, NA_Sales, EU_Sales, JP_Sales, Other_Sales,
    Global_Sales, Critic_Score, Critic_Count, User_Score, User_Count, Rating FROM PCgames''', con=conn)
df_p.head()
```

Out[86]:

	name	na_sales	eu_sales	jp_sales	other_sales	global_sales	critic_score	critic_count	user_score	user_count	rating
0	Wii Sports	41.36	28.96	3.77	8.45	82.53	76	51	8	322	E
1	Super Mario Bros.	29.08	3.58	6.81	0.77	40.24	None	None	None	None	None
2	Mario Kart Wii	15.68	12.76	3.79	3.29	35.52	82	73	8.3	709	E
3	Wii Sports Resort	15.61	10.93	3.28	2.95	32.77	80	73	8	192	E
4	Pokemon Red/Pokemon Blue	11.27	8.89	10.22	1.00	31.37	None	None	None	None	None

In [90]:
```
df_p.to_csv('PCgames_sales.csv', index=False)
conn.close()
```

In [91]:
```
conn = psycopg2.connect(database='postgres',user='postgres', password='')
cur = conn.cursor()
cur.execute('''
    CREATE TABLE PCgames_sales
    (Name varchar, NA_Sales real, EU_Sales real, JP_Sales real, Other_Sales real,
    Global_Sales real, Critic_Score varchar, Critic_Count varchar,
    User_Score varchar, User_Count varchar, Rating varchar)
    ''')
conn.commit()
conn.close()
```

In [92]:
```
conn = psycopg2.connect(database='postgres',user='postgres', password='')
cur = conn.cursor()
cur.execute('''
    COPY PCgames_sales
    FROM '/Users/chen/PCgames_sales.csv'
    WITH csv HEADER;
''')
conn.commit()
```

In [35]:
```
#after copy, read and display the PCgames_sales
df_q=pd.read_sql('''SELECT * FROM PCgames_sales''', con=conn)
df_q.head(4)
```

Out[35]:

	name	na_sales	eu_sales	jp_sales	other_sales	global_sales	critic_score	critic_count	user_score	user_count	rating
0	Wii Sports	41.36	28.96	3.77	8.45	82.53	76	51	8	322	E
1	Super Mario Bros.	29.08	3.58	6.81	0.77	40.24	None	None	None	None	None

PCgames use cases

Use case 1

/*

The use case 1 is to see the sales of different genre games in NA, JP in different platform.
According to the Name in two Tables, DB will find the sale data and return it.
The DB will return all the rows of data.

*/

```
SELECT PCgames_attribute.Platform, PCgames_attribute.genre,
PCgames_sales.NA_sales, PCgames_sales.JP_sales FROM PCgames_attribute join
PCgames_sales ON PCgames_attribute.Name = PCgames_sales.Name
```

/*OUTPUT

platform	genre	na_sales	jp_sales
Wii	Sports	41.36	3.77
NES	Platform	3.4	0.15
NES	Platform	29.08	6.81
Wii	Racing	15.68	3.79
Wii	Sports	15.61	3.28
GB	Role-Playing	11.27	10.22
GB	Puzzle	2.97	1.81
GB	Puzzle	23.2	4.22
DS	Platform	11.28	6.5
Wii	Misc	13.96	2.93
Wii	Platform	14.44	4.7
NES	Shooter	26.93	0.28
DS	Simulation	9.05	1.93
DS	Racing	9.71	4.13
GB	Role-Playing	9	7.2
Wii	Sports	8.92	3.6
X360	Misc	15	0.24
Wii	Sports	9.01	2.53
PS3	Action	0.39	0

Use case 2

```
/*
The use case 2 is to find the sales data in EU in decreasing order, display their publish year and developer.
The DB will return all data in the decreasing order.
*/
SELECT PCgames_attribute.Name, PCgames_attribute.year_of_release,
PCgames_attribute.developer, PCgames_sales.EU_sales
FROM PCgames_attribute JOIN PCgames_sales
ON PCgames_attribute.Name = PCgames_sales.Name
ORDER BY PCgames_sales.EU_sales DESC
/*OUTPUT:
```

name	year_of_release	developer	eu_sales
Wii Sports	2006	Nintendo	28.96
Mario Kart Wii	2008	Nintendo	12.76
Nintendogs	2005		10.95
Wii Sports Resort	2009	Nintendo	10.93
Brain Age: Train Your Brain in Minutes a Day	2005	Nintendo	9.2
Wii Play	2006	Nintendo	9.18
New Super Mario Bros.	2006	Nintendo	9.14
Grand Theft Auto V	2015	Rockstar North	9.09
Grand Theft Auto V	2013	Rockstar North	9.09
Grand Theft Auto V	2013	Rockstar North	9.09
Grand Theft Auto V	2014	Rockstar North	9.09
Grand Theft Auto V	2014	Rockstar North	9.09
Pokemon Red	1996		8.89
Wii Fit Plus	2009	Nintendo	8.49
Wii Fit	2007	Nintendo	8.03
Mario Kart DS	2005	Nintendo	7.47
New Super Mario Bros. Wii	2009	Nintendo	6.94

Use case 3

/*

The use case 3 is to find the data that the sales of Sports PC games, its sales and name.
The DB will return the data the Global_sales > 20.

*/

```
SELECT PCgames_attribute.Name, PCgames_attribute.genre,
PCgames_sales.Global_Sales
FROM PCgames_attribute INNER JOIN PCgames_sales
ON PCgames_attribute.Name = PCgames_sales.Name
WHERE PCgames_sales.Global_Sales > 20 AND PCgames_attribute.genre = 'Sports'
ORDER BY PCgames_sales.Global_Sales DESC
```

/*OUTPUT:

name	genre	global_sales
Wii Sports	Sports	82.53
Wii Sports Resort	Sports	32.77
Wii Fit	Sports	22.7
Wii Fit Plus	Sports	21.79

Use case 4

/*

The use case 4 gets the 10 highest platform sales data in NA of PC games.
The DB will return 10 rows.

*/

```
SELECT PCgames_attribute.Platform, sum(PCgames_sales.NA_sales) as num
FROM PCgames_attribute INNER JOIN PCgames_sales ON PCgames_attribute.Name =
PCgames_sales.Name
GROUP BY PCgames_attribute.Platform ORDER BY num DESC LIMIT 10
```

/*OUTPUT:

platform	num
X360	1389.43
PS3	1266.7
PS2	1156.84
Wii	958.841
PC	901.002
DS	774.653
XB	617.312
GC	517.811
GBA	459.039
PS	436.19

Use case 5

/*

The use case 5 gets the different district sales data, order it by sales of EU, and return the highest 20 PC games.

The DB will return 20 rows and order it by EU_sales.

*/

```
SELECT PCgames_attribute.Name, PCgames_sales.NA_sales,
PCgames_sales.EU_sales,
PCgames_sales.EU_sales, PCgames_sales.JP_sales, PCgames_sales.Other_sales
FROM PCgames_attribute INNER JOIN PCgames_sales ON
PCgames_attribute.Name = PCgames_sales.Name
WHERE PCgames_attribute.Platform = 'X360'
ORDER BY PCgames_sales.EU_sales DESC LIMIT 20
```

/*OUTPUT:

name	na_sales	eu_sales	eu_sales-2	jp_sales	other_sales
Grand Theft Auto V	7.02	9.09	9.09	0.98	3.96
The Sims 3	0.99	6.42	6.42	0	0.6
Grand Theft Auto V	3.96	6.31	6.31	0.38	1.97
FIFA 16	1.12	6.12	6.12	0.06	1.28
Call of Duty: Black Ops 3	6.03	5.86	5.86	0.36	2.38
FIFA 17	0.66	5.75	5.75	0.08	1.11
Call of Duty: Black Ops II	4.99	5.73	5.73	0.65	2.42
Call of Duty: Modern Warfare 3	5.54	5.73	5.73	0.49	1.57
Grand Theft Auto V	9.66	5.14	5.14	0.06	1.41
FIFA Soccer 13	1.06	5.01	5.01	0.13	1.97
Kinect Adventures!	15	4.89	4.89	0.24	1.69
Call of Duty: Black Ops	5.99	4.37	4.37	0.48	1.79
FIFA 15	0.8	4.33	4.33	0.05	0.9
FIFA 12	0.84	4.3	4.3	0.11	1.39
Call of Duty: Modern Warfare 3	9.04	4.24	4.24	0.13	1.32
FIFA 14	0.78	4.24	4.24	0.07	1.37
Call of Duty: Black Ops II	8.25	4.24	4.24	0.07	1.12
Grand Theft Auto IV	4.76	3.69	3.69	0.44	1.61
Call of Duty: Black Ops	9.7	3.68	3.68	0.11	1.13
Call of Duty: Modern Warfare 2	4.99	3.64	3.64	0.38	1.6