Guide to run the project:

1- Clone the repository :
2- Unzip the folder found in the git repository and place it under this path: ‘C:/tmp/’
3- Open a spark session
4- Run the project

Code Walkthrough:

At first we use SparkSession to create our spark session or get a reference to an already created one.

Then we define the schema of the csv file.

Then we setup our spark session to watch for file changes in a specified path with a specific csv schema that we already created.

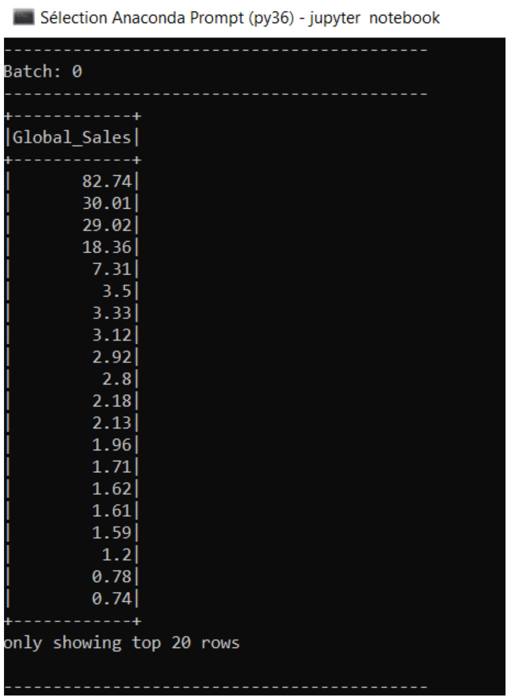
Then we create a streaming DataFrame. After that we setup our sql queries and then start the queries and instruct spark to show changes in the console.

Example of results:

I- Video Games CSV :

1- First Query (global_2006_nintendo):

```
#display global_sales in 2006 where publisher is nintendo
global_2006_nintendo = spark.sql("SELECT Global_Sales FROM VideoGames where Year=2006 and  Publisher like 'Nintendo'")
```



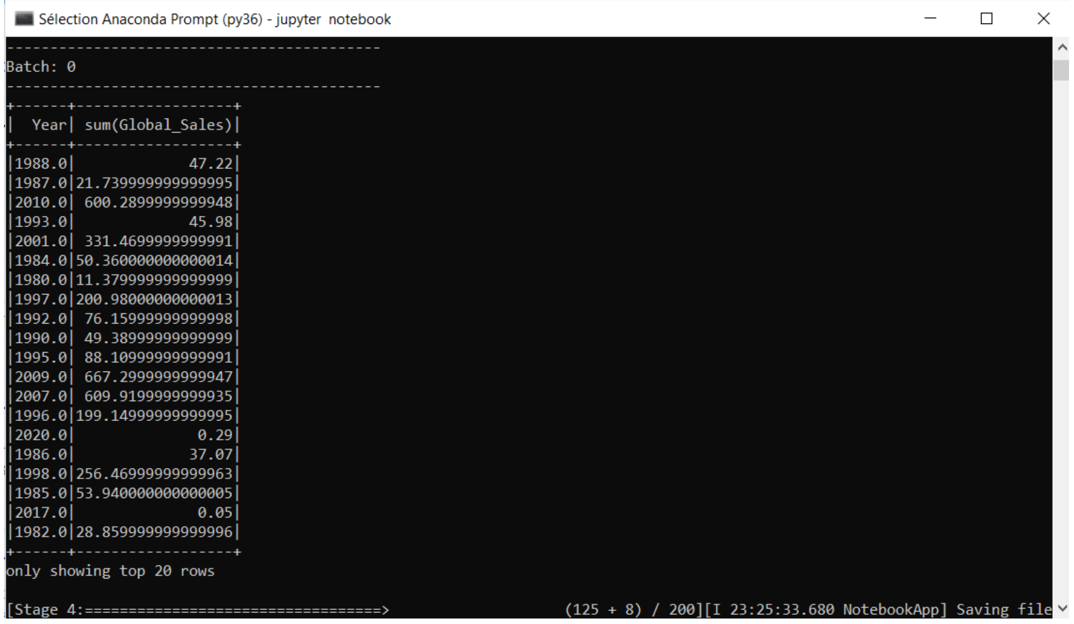
Sélection Anaconda Prompt (py36) - jupyter notebook

```
-------------------------------------------
Batch: 0
-------------------------------------------
+------------+
|Global_Sales|
+------------+
|       82.74|
|       30.01|
|       29.02|
|       18.36|
|        7.31|
|         3.5|
|        3.33|
|        3.12|
|        2.92|
|         2.8|
|        2.18|
|        2.13|
|        1.96|
|        1.71|
|        1.62|
|        1.61|
|        1.59|
|         1.2|
|        0.78|
|        0.74|
+------------+
only showing top 20 rows

-------------------------------------------
```

2- Second Query (Total_Global_Sales):

```
#display total Global Sales for each year
Total_Global_Sales = spark.sql("SELECT Year, SUM(Global_Sales) FROM VideoGames GROUP BY Year")
```

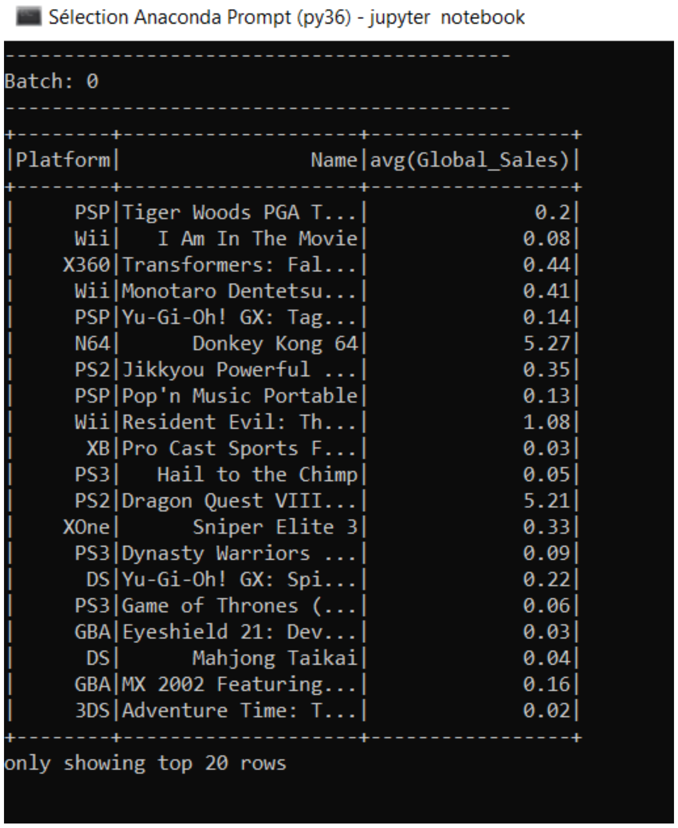


```
-------------------------------------------
Batch: 0
-------------------------------------------
+------+-----------------+
|  Year| sum(Global_Sales)|
+------+-----------------+
|1988.0|            47.22|
|1987.0|21.739999999999995|
|2010.0| 600.2899999999948|
|1993.0|            45.98|
|2001.0| 331.4699999999991|
|1984.0|50.360000000000014|
|1980.0|11.379999999999999|
|1997.0|200.98000000000013|
|1992.0| 76.15999999999998|
|1990.0| 49.38999999999999|
|1995.0| 88.10999999999991|
|2009.0| 667.2999999999947|
|2007.0| 609.9199999999935|
|1996.0|199.14999999999995|
|2020.0|             0.29|
|1986.0|            37.07|
|1998.0|256.46999999999963|
|1985.0|53.940000000000005|
|2017.0|             0.05|
|1982.0|28.859999999999996|
+------+-----------------+
only showing top 20 rows

[Stage 4:=================================>                    (125 + 8) / 200][I 23:25:33.680 NotebookApp] Saving file
```

3- Third Query (Avg_Global_sales):

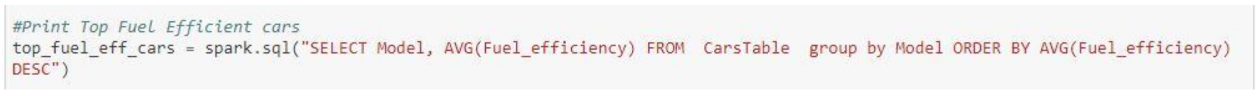
```
#Display the average of global sales for each pair of plateform and video game name
Avg_Global_sales=spark.sql("SELECT Platform, Name ,MEAN(Global_Sales) FROM VideoGames GROUP BY Platform, Name")
```



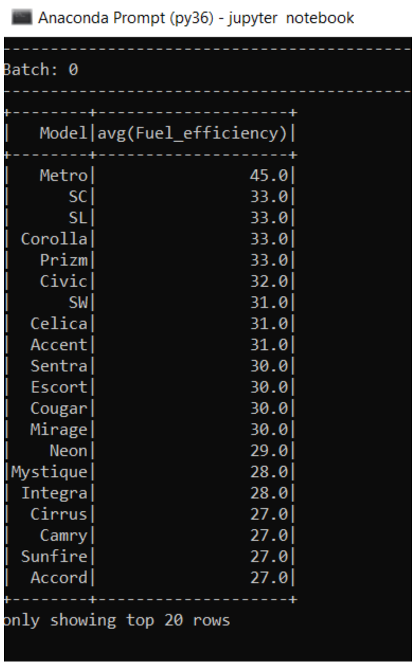
```
-------------------------------------------
Batch: 0
-------------------------------------------
+--------+--------------------+-----------------+
|Platform|                Name|avg(Global_Sales)|
+--------+--------------------+-----------------+
|     PSP|Tiger Woods PGA T...|              0.2|
|     Wii|    I Am In The Movie|             0.08|
|    X360|Transformers: Fal...|             0.44|
|     Wii|Monotaro Dentetsu...|             0.41|
|     PSP|Yu-Gi-Oh! GX: Tag...|             0.14|
|     N64|       Donkey Kong 64|             5.27|
|     PS2|Jikkyou Powerful ...|             0.35|
|     PSP|Pop'n Music Portable|             0.13|
|     Wii|Resident Evil: Th...|             1.08|
|      XB|Pro Cast Sports F...|             0.03|
|     PS3|    Hail to the Chimp|             0.05|
|     PS2|Dragon Quest VIII...|             5.21|
|    XOne|       Sniper Elite 3|             0.33|
|     PS3|Dynasty Warriors ...|             0.09|
|      DS|Yu-Gi-Oh! GX: Spi...|             0.22|
|     PS3|Game of Thrones (...|             0.06|
|     GBA|Eyeshield 21: Dev...|             0.03|
|      DS|       Mahjong Taikai|             0.04|
|     GBA|MX 2002 Featuring...|             0.16|
|     3DS|Adventure Time: T...|             0.02|
+--------+--------------------+-----------------+
only showing top 20 rows
```

I- Cars CSV:

1- First Query (top_fuel_eff_cars):



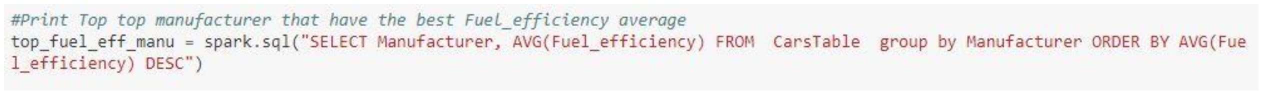
```
#Print Top Fuel Efficient cars
top_fuel_eff_cars = spark.sql("SELECT Model, AVG(Fuel_efficiency) FROM  CarsTable  group by Model ORDER BY AVG(Fuel_efficiency)
DESC")
```



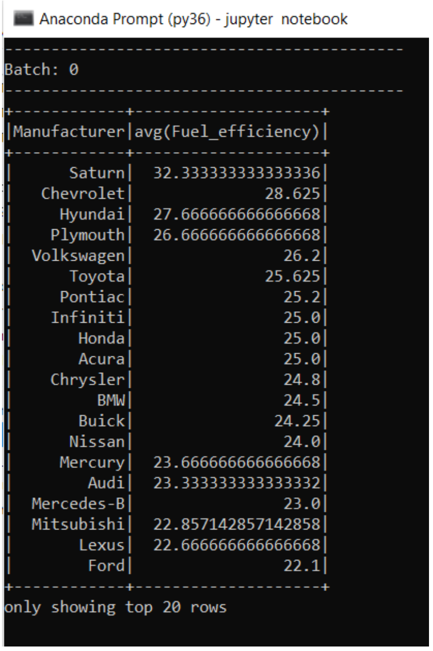
Anaconda Prompt (py36) - jupyter notebook

```
-------------------------------------------
Batch: 0
-------------------------------------------
+--------+-------------------+
|   Model|avg(Fuel_efficiency)|
+--------+-------------------+
|   Metro|               45.0|
|      SC|               33.0|
|      SL|               33.0|
| Corolla|               33.0|
|   Prizm|               33.0|
|   Civic|               32.0|
|      SW|               31.0|
|  Celica|               31.0|
|  Accent|               31.0|
|  Sentra|               30.0|
|  Escort|               30.0|
|  Cougar|               30.0|
|  Mirage|               30.0|
|    Neon|               29.0|
|Mystique|               28.0|
| Integra|               28.0|
|  Cirrus|               27.0|
|   Camry|               27.0|
| Sunfire|               27.0|
|  Accord|               27.0|
+--------+-------------------+
only showing top 20 rows
```

2- Second Query (top_fuel_eff_manu):



```
#Print Top top manufacturer that have the best Fuel_efficiency average
top_fuel_eff_manu = spark.sql("SELECT Manufacturer, AVG(Fuel_efficiency) FROM  CarsTable  group by Manufacturer ORDER BY AVG(Fue
l_efficiency) DESC")
```



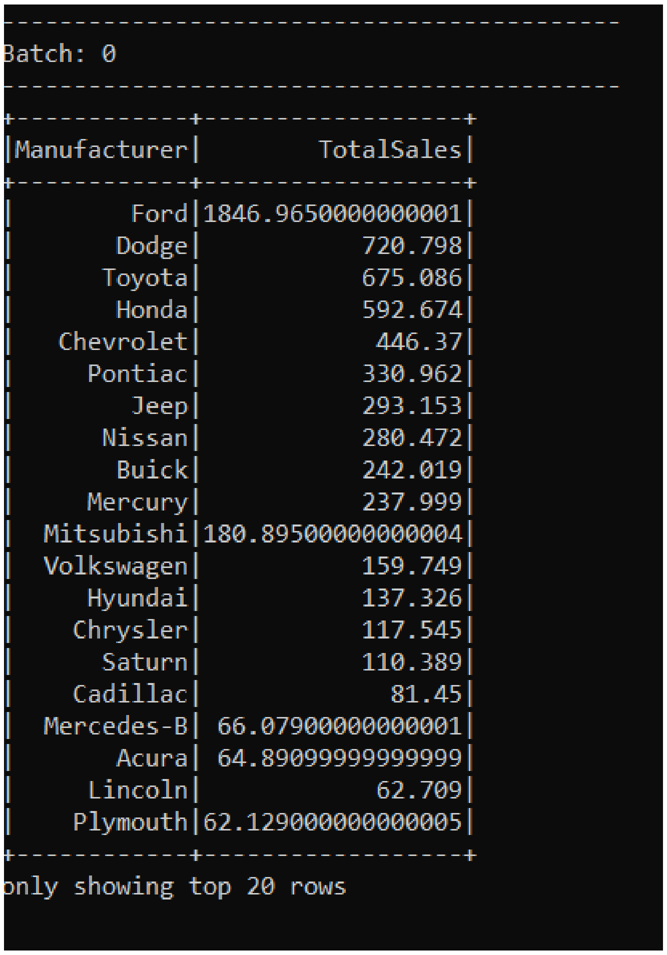
Anaconda Prompt (py36) - jupyter notebook

```
-------------------------------------------
Batch: 0
-------------------------------------------
+------------+-------------------+
|Manufacturer|avg(Fuel_efficiency)|
+------------+-------------------+
|      Saturn| 32.333333333333336|
|   Chevrolet|             28.625|
|     Hyundai| 27.666666666666668|
|    Plymouth| 26.666666666666668|
|  Volkswagen|               26.2|
|      Toyota|             25.625|
|     Pontiac|               25.2|
|    Infiniti|               25.0|
|       Honda|               25.0|
|       Acura|               25.0|
|    Chrysler|               24.8|
|         BMW|               24.5|
|       Buick|              24.25|
|      Nissan|               24.0|
|     Mercury| 23.666666666666668|
|        Audi| 23.333333333333332|
|  Mercedes-B|               23.0|
|  Mitsubishi| 22.857142857142858|
|       Lexus| 22.666666666666668|
|        Ford|               22.1|
+------------+-------------------+
only showing top 20 rows
```

3- Third Query (Total_Sales):

```
#Print Total Sales in Thousands grouped by Manufacturer and order by TotalSales
Total_Sales = spark.sql("SELECT Manufacturer, SUM(Sales_in_thousands) as TotalSales FROM CarsTable  GROUP BY Manufacturer order
by TotalSales DESC")
```

Anaconda Prompt (py36) - jupyter notebook

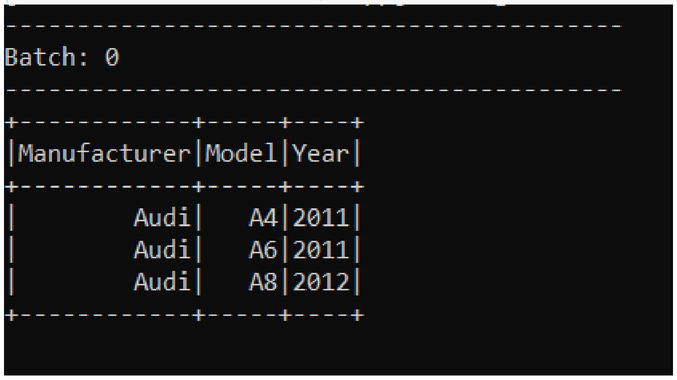


```
-------------------------------------------
Batch: 0
-------------------------------------------
+------------+------------------+
|Manufacturer|        TotalSales|
+------------+------------------+
|        Ford|1846.9650000000001|
|       Dodge|           720.798|
|      Toyota|           675.086|
|       Honda|           592.674|
|   Chevrolet|            446.37|
|     Pontiac|           330.962|
|        Jeep|           293.153|
|      Nissan|           280.472|
|       Buick|           242.019|
|     Mercury|           237.999|
|  Mitsubishi|180.89500000000004|
|  Volkswagen|           159.749|
|     Hyundai|           137.326|
|    Chrysler|           117.545|
|      Saturn|           110.389|
|    Cadillac|             81.45|
|  Mercedes-B| 66.07900000000001|
|       Acura| 64.89099999999999|
|     Lincoln|            62.709|
|    Plymouth|62.129000000000005|
+------------+------------------+
only showing top 20 rows
```

4- Forth Query (Audi_cars):

```
#Audi cars ordered by Year
Audi_cars=spark.sql("SELECT Manufacturer, Model ,SUBSTRING(Latest_Launch, length(Latest_Launch)-3 ,4 ) AS Year  FROM CarsTable w
here Manufacturer like 'Audi' ")
```

5-

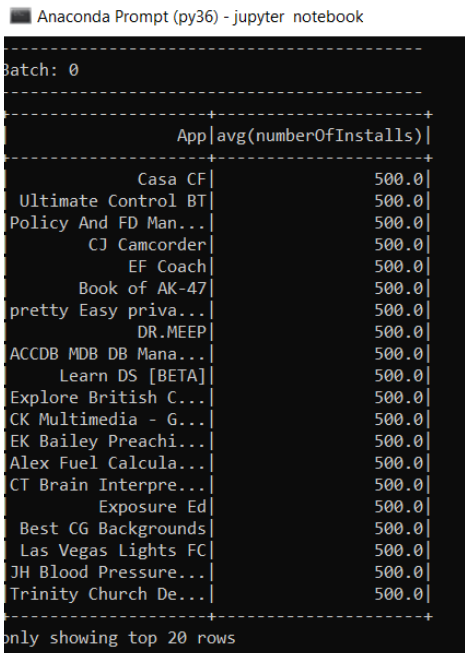


```
-------------------------------------------
Batch: 0
-------------------------------------------
+------------+-----+----+
|Manufacturer|Model|Year|
+------------+-----+----+
|        Audi|   A4|2011|
|        Audi|   A6|2011|
|        Audi|   A8|2012|
+------------+-----+----+
```

II- Google Play CSV

1- First Query (top_installs):

```
#Top installs
top_installs = spark.sql("SELECT App,AVG(numberOfInstalls) FROM Googleplay group by App order by AVG(numberOfInstalls) DESC ")
```

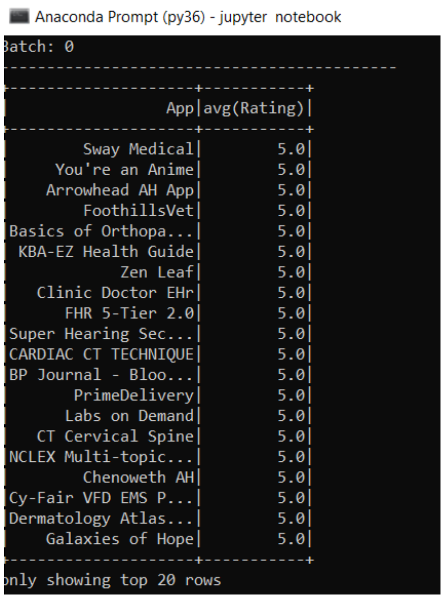


Anaconda Prompt (py36) - jupyter notebook

```
-------------------------------------------
atch: 0
-------------------------------------------
--------------------+--------------------+
                 App|avg(numberOfInstalls)|
--------------------+--------------------+
             Casa CF|               500.0|
  Ultimate Control BT|               500.0|
 Policy And FD Man...|               500.0|
         CJ Camcorder|               500.0|
             EF Coach|               500.0|
         Book of AK-47|               500.0|
 pretty Easy priva...|               500.0|
              DR.MEEP|               500.0|
 ACCDB MDB DB Mana...|               500.0|
       Learn DS [BETA]|               500.0|
 Explore British C...|               500.0|
 CK Multimedia - G...|               500.0|
 EK Bailey Preachi...|               500.0|
 Alex Fuel Calcula...|               500.0|
 CT Brain Interpre...|               500.0|
          Exposure Ed|               500.0|
  Best CG Backgrounds|               500.0|
  Las Vegas Lights FC|               500.0|
 JH Blood Pressure...|               500.0|
 Trinity Church De...|               500.0|
--------------------+--------------------+
nly showing top 20 rows
```

2- Second Query (top_rate_medapp):

```
#top Rating apps where genre = medical
top_rate_medapp = spark.sql("SELECT App , AVG(Rating) from Googleplay where Genres like 'Medical' group by App order by AVG(Rati
ng) DESC")
```

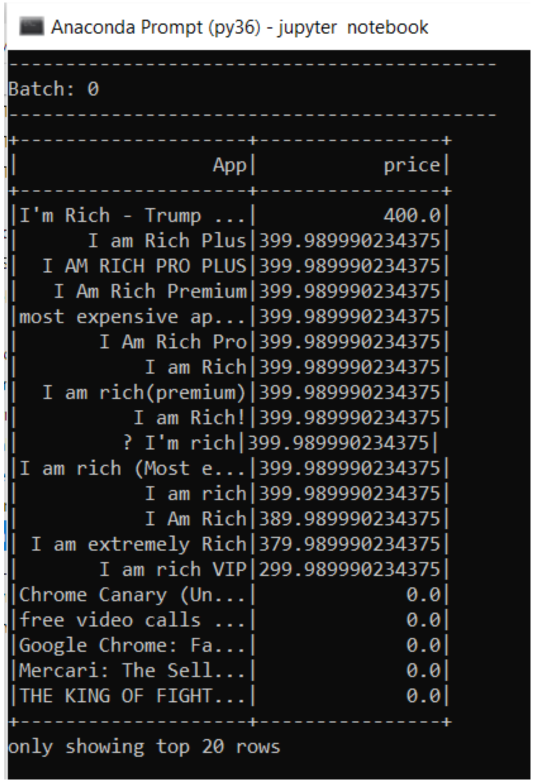


Anaconda Prompt (py36) - jupyter notebook

```
atch: 0
-------------------------------------------
--------------------+-----------+
                 App|avg(Rating)|
--------------------+-----------+
        Sway Medical|        5.0|
     You're an Anime|        5.0|
    Arrowhead AH App|        5.0|
         FoothillsVet|        5.0|
 Basics of Orthopa...|        5.0|
  KBA-EZ Health Guide|        5.0|
             Zen Leaf|        5.0|
      Clinic Doctor EHr|        5.0|
         FHR 5-Tier 2.0|        5.0|
 Super Hearing Sec...|        5.0|
 CARDIAC CT TECHNIQUE|        5.0|
 BP Journal - Bloo...|        5.0|
        PrimeDelivery|        5.0|
       Labs on Demand|        5.0|
     CT Cervical Spine|        5.0|
 NCLEX Multi-topic...|        5.0|
         Chenoweth AH|        5.0|
 Cy-Fair VFD EMS P...|        5.0|
 Dermatology Atlas...|        5.0|
      Galaxies of Hope|        5.0|
--------------------+-----------+
nly showing top 20 rows
```

3- Third Query (top_priced_app):

```
#top priced apps
top_priced_app = spark.sql("SELECT App ,AVG(price) AS price from Googleplay group by App order by price DESC")
```

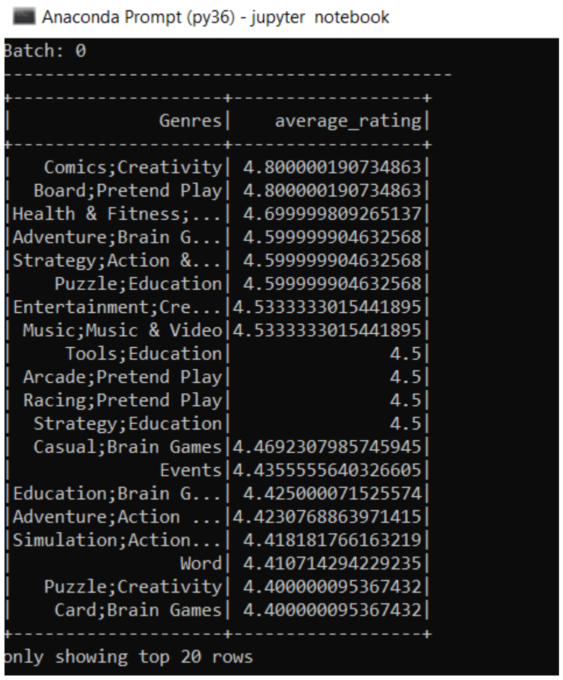


Anaconda Prompt (py36) - jupyter notebook

```
-------------------------------------------
Batch: 0
-------------------------------------------
+--------------------+----------------+
|                 App|           price|
+--------------------+----------------+
|I'm Rich - Trump ...|           400.0|
|      I am Rich Plus|399.989990234375|
|  I AM RICH PRO PLUS|399.989990234375|
|   I Am Rich Premium|399.989990234375|
|most expensive ap...|399.989990234375|
|        I Am Rich Pro|399.989990234375|
|           I am Rich|399.989990234375|
|  I am rich(premium)|399.989990234375|
|          I am Rich!|399.989990234375|
|          ? I'm rich|399.989990234375|
|I am rich (Most e...|399.989990234375|
|           I am rich|399.989990234375|
|           I Am Rich|389.989990234375|
| I am extremely Rich|379.989990234375|
|       I am rich VIP|299.989990234375|
|Chrome Canary (Un...|             0.0|
|free video calls ...|             0.0|
|Google Chrome: Fa...|             0.0|
|Mercari: The Sell...|             0.0|
|THE KING OF FIGHT...|             0.0|
+--------------------+----------------+
only showing top 20 rows
```

4- Forth Query (top_rated_cat):

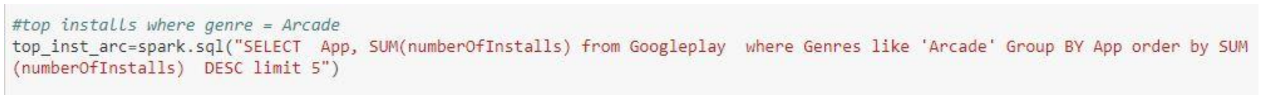
```
#Top rated categories on Google play
top_rated_cat=spark.sql("SELECT  Genres,AVG(Rating) as average_rating from Googleplay Group BY Genres order by average_rating DE
SC")
```



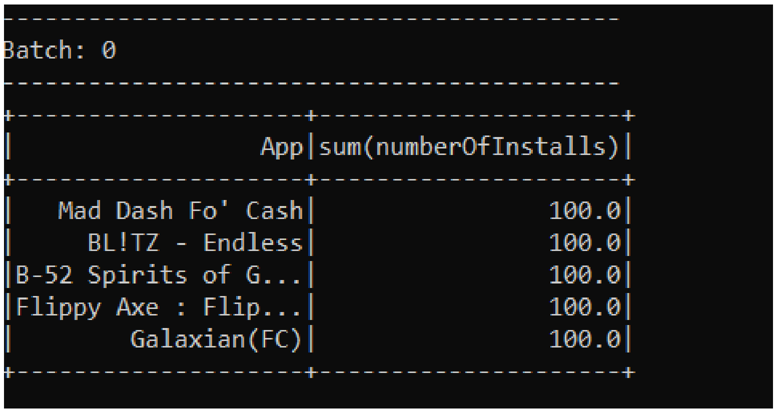
Anaconda Prompt (py36) - jupyter notebook

```
Batch: 0
-------------------------------------------
+--------------------+------------------+
|              Genres|    average_rating|
+--------------------+------------------+
|    Comics;Creativity| 4.800000190734863|
|   Board;Pretend Play| 4.800000190734863|
|Health & Fitness;...| 4.699999809265137|
|Adventure;Brain G...| 4.599999904632568|
|Strategy;Action &...| 4.599999904632568|
|     Puzzle;Education| 4.599999904632568|
|Entertainment;Cre...|4.5333333015441895|
| Music;Music & Video|4.5333333015441895|
|      Tools;Education|               4.5|
|  Arcade;Pretend Play|               4.5|
|  Racing;Pretend Play|               4.5|
|   Strategy;Education|               4.5|
|   Casual;Brain Games|4.4692307985745945|
|               Events|4.4355555640326605|
|Education;Brain G...| 4.425000071525574|
|Adventure;Action ...|4.4230768863971415|
|Simulation;Action...| 4.418181766163219|
|                 Word| 4.410714294229235|
|    Puzzle;Creativity| 4.400000095367432|
|     Card;Brain Games| 4.400000095367432|
+--------------------+------------------+
only showing top 20 rows
```

5- Fifth Query (top_inst_arc):



```
#top installs where genre = Arcade
top_inst_arc=spark.sql("SELECT  App, SUM(numberOfInstalls) from Googleplay  where Genres like 'Arcade' Group BY App order by SUM
(numberOfInstalls)  DESC limit 5")
```



```
-------------------------------------------
Batch: 0
-------------------------------------------
+--------------------+---------------------+
|                 App|sum(numberOfInstalls)|
+--------------------+---------------------+
|    Mad Dash Fo' Cash|                100.0|
|      BL!TZ - Endless|                100.0|
|B-52 Spirits of G...|                100.0|
|Flippy Axe : Flip...|                100.0|
|        Galaxian(FC)|                100.0|
+--------------------+---------------------+
```