databricks Google play project

(https://databricks.com)

```
import pyspark
from pyspark.sql import SparkSession
from pyspark.sql.types import StructField ,StructType,IntegerType, StringType
from pyspark.sql.functions import *



#Create Data Frame by loading the dataset to databriicks


df = spark.read.load('/FileStore/tables/googleplaystore.csv',format ='csv',header= 'true',escape='"',inferschema = 'true')
```

Show result

```
df.count()
```

```
Out[24]: 10841
```

```
df.show(1)
```

```
+--------------------+--------------+------+-------+----+---------+----+-----+--------------+------------+---------------+
-----------+------------+
|                 App|      Category|Rating|Reviews|Size|Installs|Type|Price|Content Rating|      Genres|   Last Updated|
Current Ver| Android Ver|
+--------------------+--------------+------+-------+----+---------+----+-----+--------------+------------+---------------+
-----------+------------+
|Photo Editor & Ca...|ART_AND_DESIGN|   4.1|    159| 19M| 10,000+|Free|    0|      Everyone|Art & Design|January 7, 2018|
1.0.0|4.0.3 and up|
+--------------------+--------------+------+-------+----+---------+----+-----+--------------+------------+---------------+
-----------+------------+
only showing top 1 row
```

```
df.printSchema()
```

```
root
 |-- App: string (nullable = true)
 |-- Category: string (nullable = true)
 |-- Rating: double (nullable = true)
 |-- Reviews: string (nullable = true)
 |-- Size: string (nullable = true)
 |-- Installs: string (nullable = true)
 |-- Type: string (nullable = true)
 |-- Price: string (nullable = true)
 |-- Content Rating: string (nullable = true)
 |-- Genres: string (nullable = true)
 |-- Last Updated: string (nullable = true)
 |-- Current Ver: string (nullable = true)
 |-- Android Ver: string (nullable = true)
```

```
#Data Cleaning

#1 Deleting unwanted rows


df = df.drop("Size","Android Ver","Last Updated")
df = df.drop("Content Rating","Current Ver")
```

```
df.show(2)
```

```
+--------------------+--------------+------+-------+--------+----+-----+--------------------+
|                 App|      Category|Rating|Reviews|Installs|Type|Price|              Genres|
+--------------------+--------------+------+-------+--------+----+-----+--------------------+
```

```
|Photo Editor & Ca...|ART_AND_DESIGN|   4.1|    159| 10,000+|Free|    0|        Art & Design|
| Coloring book moana|ART_AND_DESIGN|   3.9|    967|500,000+|Free|    0|Art & Design;Pret...|
+--------------------+--------------+------+-------+--------+----+-----+--------------------+
only showing top 2 rows
```

```
df.printSchema()
```

```
root
 |-- App: string (nullable = true)
 |-- Category: string (nullable = true)
 |-- Rating: double (nullable = true)
 |-- Reviews: string (nullable = true)
 |-- Installs: string (nullable = true)
 |-- Type: string (nullable = true)
 |-- Price: string (nullable = true)
 |-- Genres: string (nullable = true)
```

```
df.show(1)
```

```
+--------------------+--------------+------+-------+--------+----+-----+------------+
|                 App|      Category|Rating|Reviews|Installs|Type|Price|      Genres|
+--------------------+--------------+------+-------+--------+----+-----+------------+
|Photo Editor & Ca...|ART_AND_DESIGN|   4.1|    159| 10,000+|Free|    0|Art & Design|
+--------------------+--------------+------+-------+--------+----+-----+------------+
only showing top 1 row
```

```
# 2) Changing Data Types for analysis
```

```
df = df.withColumn("Reviews",col("Reviews").cast(IntegerType()))\
 .withColumn("Installs",col("Installs").cast(IntegerType()))\
.withColumn("Price",col("Price").cast(IntegerType()))
```

```
df.show(1)
```

```
+--------------------+--------------+------+-------+--------+----+-----+------------+
|                 App|      Category|Rating|Reviews|Installs|Type|Price|      Genres|
+--------------------+--------------+------+-------+--------+----+-----+------------+
|Photo Editor & Ca...|ART_AND_DESIGN|   4.1|    159|   10000|Free|    0|Art & Design|
+--------------------+--------------+------+-------+--------+----+-----+------------+
only showing top 1 row
```

```
df.printSchema()
```

```
root
 |-- App: string (nullable = true)
 |-- Category: string (nullable = true)
 |-- Rating: double (nullable = true)
 |-- Reviews: integer (nullable = true)
 |-- Installs: string (nullable = true)
 |-- Type: string (nullable = true)
 |-- Price: string (nullable = true)
 |-- Genres: string (nullable = true)
```

```
# Use Regex to Validate correct data in the clolmns
```

```
from pyspark.sql.functions import regexp_replace, col

df = df.withColumn("Installs", regexp_replace(col("Installs"), "[^0-9]", "")) \
    .withColumn("Price", regexp_replace(col("Price"), "[$]", ""))
```

```
df.show(1)
```

```
+--------------------+--------------+------+-------+--------+----+-----+------------+
|                 App|      Category|Rating|Reviews|Installs|Type|Price|      Genres|
+--------------------+--------------+------+-------+--------+----+-----+------------+
|Photo Editor & Ca...|ART_AND_DESIGN|   4.1|    159|   10000|Free|    0|Art & Design|
+--------------------+--------------+------+-------+--------+----+-----+------------+
only showing top 1 row
```

```
df.printSchema()
```

```
root
 |-- App: string (nullable = true)
 |-- Category: string (nullable = true)
 |-- Rating: double (nullable = true)
 |-- Reviews: integer (nullable = true)
 |-- Installs: string (nullable = true)
 |-- Type: string (nullable = true)
 |-- Price: string (nullable = true)
 |-- Genres: string (nullable = true)
```

```
# Expoloratory Data Analysis and Charts
```

```
# Create a SQL TEMP View for the Dataset
```

```
df.createOrReplaceTempView("apps")
```

```
# Run Sql Queries on dataset
```

Top 10 Reviews given to to a app

```
df.createOrReplaceTempView("apps")

query = '''
    SELECT App, SUM(Reviews)
    FROM apps
    GROUP BY App
    ORDER BY SUM(Reviews) DESC
    LIMIT 10
'''
```

```
result = spark.sql(query)
result.show()
```

```
+--------------------+------------+
|                 App|sum(Reviews)|
+--------------------+------------+
|           Instagram|   266241989|
|  WhatsApp Messenger|   207348304|
|      Clash of Clans|   179558781|
|Messenger – Text ...|   169932272|
|      Subway Surfers|   166331958|
|    Candy Crush Saga|   156993136|
|            Facebook|   156286514|
|         8 Ball Pool|    99386198|
|        Clash Royale|    92530298|
|            Snapchat|    68045010|
+--------------------+------------+
```

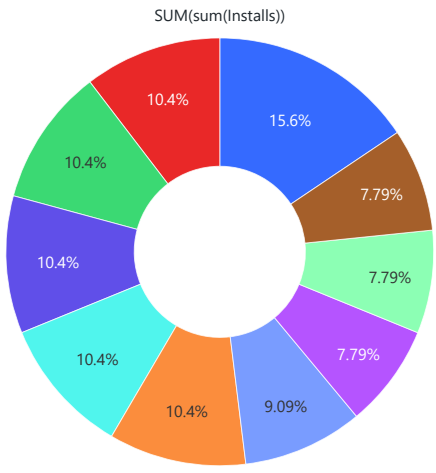
Top 10 installed app

```
df.createOrReplaceTempView("apps")  # Assuming `df` is your DataFrame containing the "apps" data

query = '''
    SELECT App, Type, SUM(Installs)
    FROM apps
    GROUP BY 1,2
    ORDER BY 3 DESC
'''

result = spark.sql(query)
result.show()
```

```
+--------------------+----+-------------+
|                 App|Type|sum(Installs)|
+--------------------+----+-------------+
|      Subway Surfers|Free|        6.0E9|
|           Instagram|Free|        4.0E9|
|        Google Drive|Free|        4.0E9|
|            Hangouts|Free|        4.0E9|
|       Google Photos|Free|        4.0E9|
|         Google News|Free|        4.0E9|
|    Candy Crush Saga|Free|        3.5E9|
|  WhatsApp Messenger|Free|        3.0E9|
|               Gmail|Free|        3.0E9|
|        Temple Run 2|Free|        3.0E9|
|Skype - free IM &...|Free|        3.0E9|
|Google Chrome: Fa...|Free|        3.0E9|
|Messenger – Text ...|Free|        3.0E9|
|Maps - Navigate &...|Free|        3.0E9|
|     Viber Messenger|Free|        2.5E9|
|   Google Play Games|Free|        2.0E9|
|            Facebook|Free|        2.0E9|
|            Snapchat|Free|        2.0E9|
```

```
%sql SELECT App, Type, SUM(Installs)
    FROM apps
    GROUP BY 1,2
    ORDER BY 3 DESC
    LIMIT 10
```

Table **Visualization 1**

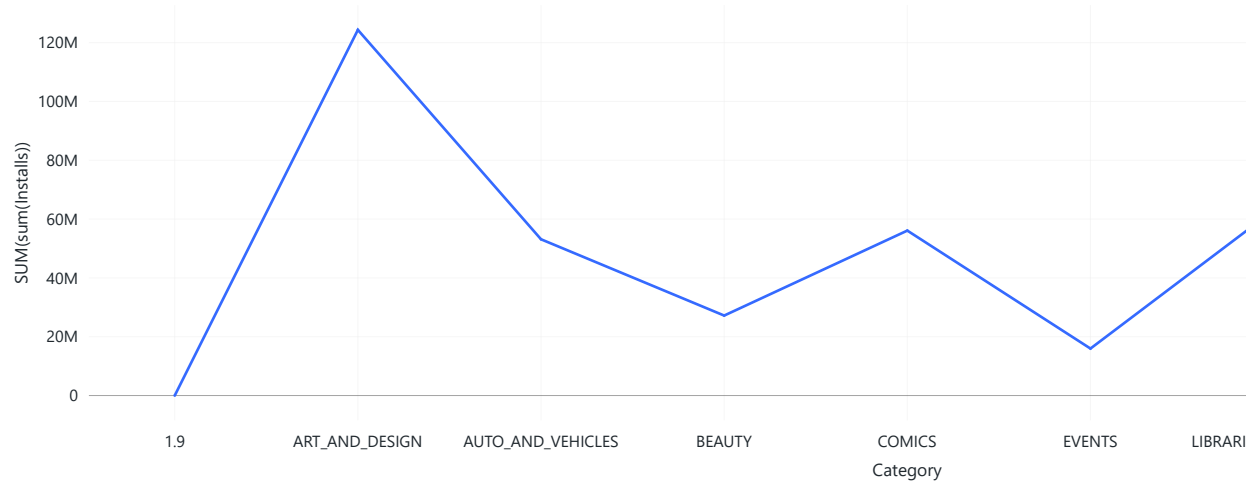


10 rows

Category wise distribution

```
%sql select Category , sum(Installs) from apps
group by 1
order by 2 asc
limit 9
```

Table **Visualization 1**



9 rows