Ejercicios de DataFrames

Trabajando sobre el fichero googleplaystore.csv

La Spark Session la he inicializado en un script de prueba anteriormente:

```
%python
from pyspark.sql import SparkSession

# Crear o obtener una SparkSession
spark = SparkSession.builder.appName("DataFrames").getOrCreate()
```

1. Elimina la App “Life Made WI-FI Touchscreen Photo Frame”

```
%python
from pyspark.sql import SparkSession
from pyspark.sql.functions import col

df = spark.read.csv("dbfs:/FileStore/tables/Formacion_Binaia/Practica_DataFrames/googleplaystore_2.csv", header=True, inferSchema=True)

# Filtrar el DataFrame para excluir las filas donde el nombre de la aplicación empieza con "Life Made"
df_filtered = df.filter(~col("App").startswith("Life Made"))

# Escribir el DataFrame filtrado en un archivo temporal
temp_path = "dbfs:/FileStore/tables/Formacion_Binaia/Practica_DataFrames/temp_googleplaystore_2.csv"
df_filtered.write.csv(temp_path, header=True, mode='overwrite')

# Recolectar los datos filtrados
data = df_filtered.collect()
print(data)

# Contar las filas filtradas
contador = len(data)
print(contador)

# Eliminar el archivo original
dbutils.fs.rm("dbfs:/FileStore/tables/Formacion_Binaia/Practica_DataFrames/googleplaystore_2.csv", True)

# Mover el archivo temporal al destino original
dbutils.fs.mv(temp_path, "dbfs:/FileStore/tables/Formacion_Binaia/Practica_DataFrames/googleplaystore_2.csv", True)
```

2. Sustituir los valores NaN en la columna Rating.

```
%python
#EJERCICIO 2
from pyspark.sql import SparkSession
from pyspark.sql.functions import col

# Leer el archivo CSV en un DataFrame de Spark
df = spark.read.csv("dbfs:/FileStore/tables/Formacion_Binaia/Practica_DataFrames/googleplaystore_2.csv", header=True, inferSchema=True)

# Sustituir los valores NaN en la columna 'Rating' por un valor específico (por ejemplo, 0.0)
df_filled = df.fillna({'Rating': 0.0})  #fillna sirve para reemplazar valores nulos (null o NaN) en un dataframe

# Ordenar el DataFrame por la columna 'App'
df_sorted = df_filled.orderBy("App")

temp_path = "dbfs:/FileStore/tables/Formacion_Binaia/Practica_DataFrames/temp_googleplaystore_2.csv"
df_sorted.write.csv(temp_path, header=True, mode='overwrite')

# Recolectar los datos filtrados
data = df_sorted.collect()
print(data)

# Contar las filas filtradas
contador = len(data)
print(contador)

# Eliminar el archivo original
dbutils.fs.rm("dbfs:/FileStore/tables/Formacion_Binaia/Practica_DataFrames/googleplaystore_2.csv", True)

# Mover el archivo temporal al destino original
dbutils.fs.mv(temp_path, "dbfs:/FileStore/tables/Formacion_Binaia/Practica_DataFrames/googleplaystore_2.csv", True)
```

3. Sustituir NaN en columna Type por “Unknown”.

```
%python
#EJERCICIO 2
from pyspark.sql import SparkSession
from pyspark.sql.functions import col

# Leer el archivo CSV en un DataFrame de Spark
df = spark.read.csv("dbfs:/FileStore/tables/Formacion_Binaia/Practica_DataFrames/googleplaystore_2.csv", header=True,
inferSchema=True)

# Sustituir los valores NaN en la columna 'Rating' por un valor específico (por ejemplo, 0.0)
df_filled = df.fillna({'Type': 'Unknown'})  #fillna sirve para reemplazar valores nulos (null o NaN) en un dataframe

# Ordenar el DataFrame por la columna 'App'
df_sorted = df_filled.orderBy("App")

temp_path = "dbfs:/FileStore/tables/Formacion_Binaia/Practica_DataFrames/temp_googleplaystore_2.csv"
df_sorted.write.csv(temp_path, header=True, mode='overwrite')

# Recolectar los datos filtrados
data = df_sorted.collect()
print(data)

# Contar las filas filtradas
contador = len(data)
print(contador)

# Eliminar el archivo original
dbutils.fs.rm("dbfs:/FileStore/tables/Formacion_Binaia/Practica_DataFrames/googleplaystore_2.csv", True)

# Mover el archivo temporal al destino original
dbutils.fs.mv(temp_path, "dbfs:/FileStore/tables/Formacion_Binaia/Practica_DataFrames/googleplaystore_2.csv", True)
```

4. Agregar una columna que nos indica si las características varían dependiendo del dispositivo

```
%python
#EJERCICIO 4
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, when

# Leer el archivo CSV en un DataFrame de Spark
df = spark.read.csv("dbfs:/FileStore/tables/Formacion_Binaia/Practica_DataFrames/googleplaystore_2.csv", header=True, inferSchema=True)

df_with_varies = df.withColumn("Varies_with_device", when(col("Current ver") == "Varies with device", True).otherwise(False))
#withcolumn para añadir una nueva columna o reemplazar una existente
#(nombre de columna, y el when para determinar la condición para poner los valores en la columna)

temp_path = "dbfs:/FileStore/tables/Formacion_Binaia/Practica_DataFrames/temp_googleplaystore_2.csv"
df_with_varies.write.csv(temp_path, header=True, mode='overwrite')

# Recolectar los datos filtrados
data = df_with_varies.collect()
print(data)

# Contar las filas filtradas
contador = len(data)
print(contador)

# Eliminar el archivo original
dbutils.fs.rm("dbfs:/FileStore/tables/Formacion_Binaia/Practica_DataFrames/googleplaystore_2.csv", True)

# Mover el archivo temporal al destino original
dbutils.fs.mv(temp_path, "dbfs:/FileStore/tables/Formacion_Binaia/Practica_DataFrames/googleplaystore_2.csv", True)
```

5. Crea una nueva columna llamada Frec_Download con los siguientes valores.

a. Baja => número de instalaciones < 50000
b. Media => número de instalaciones >= 50000 y < 1000000
c. Alta => número de instalaciones >= 1000000 y < 50000000
d. Muy alta => número de instalaciones >= 50000000

```python
#EJERCICIO 5
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, when, regexp_replace

# Leer el archivo CSV en un DataFrame de Spark
df = spark.read.csv("dbfs:/FileStore/tables/Formacion_Binaia/Practica_DataFrames/googleplaystore_2.csv", header=True, inferSchema=True)

# Limpiar y convertir la columna 'Installs' a un formato numérico ya que contiene valores que son cadenas de texto con caracteres especiales como (, +)...
df = df.withColumn("Installs", regexp_replace(col("Installs"), "[+,]","").cast("int"))
#regexp_replace remueve los caracteres '+' y ',' de los valores en la columna Installs y los converite a int

# Crear la nueva columna 'Frec_Download' con los valores correspondientes
df_with_frec_download = df.withColumn(
    "Frec_Download",
    when(col("Installs") < 50000, "Baja")
    .when((col("Installs") >= 50000) & (col("Installs") < 1000000), "Media")
    .when((col("Installs") >= 1000000) & (col("Installs") < 50000000), "Alta")
    .when(col("Installs") >= 50000000, "Muy alta")
)

temp_path = "dbfs:/FileStore/tables/Formacion_Binaia/Practica_DataFrames/temp_googleplaystore_2.csv"
df_with_frec_download.write.csv(temp_path, header=True, mode='overwrite')

# Recolectar los datos filtrados
data = df_with_frec_download.collect()
print(data)

# Contar las filas filtradas
contador = len(data)
print(contador)

# Eliminar el archivo original
dbutils.fs.rm("dbfs:/FileStore/tables/Formacion_Binaia/Practica_DataFrames/googleplaystore_2.csv", True)

# Mover el archivo temporal al destino original
dbutils.fs.mv(temp_path, "dbfs:/FileStore/tables/Formacion_Binaia/Practica_DataFrames/googleplaystore_2.csv", True)
```

6. Consultas sobre df_limpio

a. Muestra aquellas aplicaciones que tengan una frecuencia de descarga muy alta y una valoración mayor a 4.5

```python
%python
#EJERCICIO 6 A
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, when, regexp_replace

df = spark.read.csv("dbfs:/FileStore/tables/Formacion_Binaia/Practica_DataFrames/googleplaystore_2.csv", header=True, inferSchema=True)

# Filtrar aplicaciones con frecuencia de descarga muy alta y valoración mayor a 4.5
filtered_df = df.filter((col("Frec_Download") == "Muy alta") & (col("Rating") > 4.5))

app_names_df = filtered_df.select("App")  #se extrae solo la columna App del DataFrame filtrado y se almacena en app_names_df

app_names_df.show(n=filtered_df.count(), truncate=False)  #mostrar todos los nombres de las aplicaciones que cumplen las condiciones, con el número de filas igual al conteo de filtered_df y sin truncar el texto.
```

```
▸ (5) Spark Jobs
▸ df: pyspark.sql.dataframe.DataFrame = [App: string, Category: string ... 13 more fields]
▸ filtered_df: pyspark.sql.dataframe.DataFrame = [App: string, Category: string ... 13 more fields]
▸ app_names_df: pyspark.sql.dataframe.DataFrame = [App: string]
|Toy Blast                                          |
|Twitch: Livestream Multiplayer Games & Esports     |
|Twitch: Livestream Multiplayer Games & Esports     |
|Video Downloader for Facebook                      |
|Video Editor Music,Cut,No Crop                     |
|VideoShow-Video Editor, Video Maker, Beauty Camera |
|VivaVideo - Video Editor & Photo Movie             |
|War Robots                                         |
|Wattpad 📖 Free Books                              |
|Wattpad 📖 Free Books                              |
|Waze - GPS, Maps, Traffic Alerts & Live Navigation |
|Waze - GPS, Maps, Traffic Alerts & Live Navigation |
|YouCam Makeup - Magic Selfie Makeovers             |
|YouCam Makeup - Magic Selfie Makeovers             |
|ZEDGE™ Ringtones & Wallpapers                      |
|ZEDGE™ Ringtones & Wallpapers                      |
|ZEDGE™ Ringtones & Wallpapers                      |
|ZEDGE™ Ringtones & Wallpapers                      |
|ZenUI Launcher                                     |
+---------------------------------------------------+
```

b. Muestra el número de aplicaciones con frecuencia de descarga muy alta y de coste gratuito.

```
%python
#EJERCICIO 6 B
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, when, regexp_replace

df = spark.read.csv("dbfs:/FileStore/tables/Formacion_Binaia/Practica_DataFrames/googleplaystore_2.csv", header=True, inferSchema=True)


filtered_df = df.filter((col("Frec_Download") == "Muy alta") & (col("Type") == "Free"))

# Mostrar las aplicaciones que cumplen con las condiciones
filtered_data = filtered_df.collect()
contador = len(filtered_data)
print(contador)
```

(3) Spark Jobs

df: pyspark.sql.dataframe.DataFrame = [App: string, Category: string ... 13 more fields]

filtered_df: pyspark.sql.dataframe.DataFrame = [App: string, Category: string ... 13 more fields]

```
828
```

c. Muestra aquellas aplicaciones cuyo precio sea menor que 13 dólares

```
%python
#EJERCICIO 6 C
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, when, regexp_replace

df = spark.read.csv("dbfs:/FileStore/tables/Formacion_Binaia/Practica_DataFrames/googleplaystore_2.csv", header=True, inferSchema=True)

df = df.withColumn("Price", regexp_replace(col("Price"), "[$]", "").cast("float"))

filtered_df = df.filter(col("Price") < 13)

app_names_df = filtered_df.select("App")

app_names_df.show(n=filtered_df.count(), truncate=False)
```

(5) Spark Jobs

df: pyspark.sql.dataframe.DataFrame = [App: string, Category: string ... 13 more fields]

filtered_df: pyspark.sql.dataframe.DataFrame = [App: string, Category: string ... 13 more fields]

app_names_df: pyspark.sql.dataframe.DataFrame = [App: string]

```
+---------------------------------------------------------------------------------------------------------------------------------------
|App
+---------------------------------------------------------------------------------------------------------------------------------------
|"""i DT"" Fútbol. Todos Somos Técnicos."
|"Alphabet ""H"" Passcode Lock Screen"
|"Eat Fast Prepare ""Without Internet"""
|"Official QR Code® Reader ""Q"""
|"The FN ""Baby"" pistol explained"
|"Women""s Health Tips"
|"Yanosik: ""antyradar""; traffic jams; navigation; camera"
|+Download 4 Instagram Twitter
|- Free Comics - Comic Apps
|.R
|/u/app
|058.ba
|1. FC Köln App
|10 Best Foods for You
|10 Best Foods for You
|10 Minutes a Day Times Tables
|10 WPM Amateur ham radio CW Morse code trainer
|10,000 Quotes DB (Premium)
```

7. Dado que nuestro set de datos contiene muchos registros, para comprobar nuestros resultados puedes hacer pruebas en una porción del dataset. Prueba usando el 10% de nuestros datos y con seed = 123.

```
%python
#EJERCICIO 6 C
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, when, regexp_replace

# Leer el archivo CSV en un DataFrame de Spark
df = spark.read.csv("dbfs:/FileStore/tables/Formacion_Binaia/Practica_DataFrames/googleplaystore_2.csv", header=True, inferSchema=True)

# Obtener una muestra del 10% del DataFrame con seed 123
df_sample = df.sample(fraction=0.1, seed=123)
```