Spark Preparation

We check if we are in Google Colab. If this is the case, install all necessary packages.

To run spark in Colab, we need to first install all the dependencies in Colab environment i.e. Apache Spark 3.3.2 with hadoop 3.3, Java 8 and Findspark to locate the spark in the system. The tools installation can be carried out inside the Jupyter Notebook of the Colab. Learn more from A Must-Read Guide on How to Work with PySpark on Google Colab for Data Scientists!

```
try:
  import google.colab
  IN_COLAB = True
except:
  IN_COLAB = False
```

```
if IN_COLAB:
    !apt-get install openjdk-8-jdk-headless -qq > /dev/null
    !wget -q https://dlcdn.apache.org/spark/spark-3.3.2/spark-3.3.2-bin-hadoop3.tgz
    !tar xf spark-3.3.2-bin-hadoop3.tgz
    !mv spark-3.3.2-bin-hadoop3 spark
    !pip install -q findspark
    import os
    os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
    os.environ["SPARK_HOME"] = "/content/spark"
```

Start a Local Cluster

```
import findspark
findspark.init()
```

```
spark_url = 'local'
```

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

```
spark = SparkSession.builder\
        .master(spark_url)\
        .appName('Spark SQL')\
        .getOrCreate()
```

Spark Assignment

Based on the movie review dataset in 'netflix-rotten-tomatoes-metacritic-imdb.csv', answer the below questions.

Note: do not clean or remove missing data

```
path = 'netflix-rotten-tomatoes-metacritic-imdb.csv'
```

```
df = spark.read.option("header", True).csv(path)
```

```
df.show(3)
```

```
+-------------------+--------------------+--------------------+----------------+---------------+----------------+-
|              Title|               Genre|                Tags|       Languages|Series or Movie|Hidden Gem Score|
+-------------------+--------------------+--------------------+----------------+---------------+----------------+-
|   Lets Fight Ghost|Crime, Drama, Fan...|Comedy Programmes...|Swedish, Spanish|         Series|             4.3|
|HOW TO BUILD A GIRL|              Comedy|Dramas,Comedies,F...|         English|          Movie|             7.0|
|         Centigrade|     Drama, Thriller|           Thrillers|         English|          Movie|             6.4|
+-------------------+--------------------+--------------------+----------------+---------------+----------------+-
only showing top 3 rows
```

```
df.columns
```

```
['Title',
 'Genre',
 'Tags',
 'Languages',
 'Series or Movie',
 'Hidden Gem Score',
 'Country Availability',
 'Runtime',
 'Director',
 'Writer',
 'Actors',
 'View Rating',
 'IMDb Score',
 'Rotten Tomatoes Score',
 'Metacritic Score',
 'Awards Received',
 'Awards Nominated For',
 'Boxoffice',
 'Release Date',
 'Netflix Release Date',
 'Production House',
 'Netflix Link',
 'IMDb Link',
 'Summary',
 'IMDb Votes',
 'Image',
 'Poster',
 'TMDb Trailer',
 'Trailer Site']
```

What is the maximum and average of the overall hidden gem score?

```
df = df.withColumn('Hidden Gem Score', df['Hidden Gem Score'].cast('float'))
```

```
from pyspark.sql.functions import avg, min, max, countDistinct
df.select(avg('Hidden Gem Score'), max('Hidden Gem Score')).show()
```

```
+---------------------+---------------------+
|avg(Hidden Gem Score)|max(Hidden Gem Score)|
+---------------------+---------------------+
|    5.937551392466598|                  9.8|
+---------------------+---------------------+
```

How many movies that are available in Korea?

Consider "Language" column

```
from pyspark.sql.functions import array_contains
from pyspark.sql.functions import split
from pyspark.sql.functions import split, explode

movies_df = df.withColumn("Language", split("Languages", ",\s*"))
korean_movies_df = movies_df.filter(array_contains("Language", "Korean"))
# korean_movies_df = korean_movies_df.filter(col("Series or Movie") == "Movie")
korean_movie_count = korean_movies_df.distinct().count()
print("Number of movies available in Korea:", korean_movie_count)

```

```
Number of movies available in Korea: 735
```

Which director has the highest average hidden gem score?

```
director_scores_df = df.groupBy("Director").agg(avg("Hidden Gem Score").alias("Avg Hidden Gem Score"))
sorted_director_scores_df = director_scores_df.sort(col("Avg Hidden Gem Score").desc()).first()
top_director = sorted_director_scores_df["Director"]
top_score = sorted_director_scores_df["Avg Hidden Gem Score"]
print("The director with the highest average hidden gem score is", top_director, "with an average score of", top_scc

```

```
The director with the highest average hidden gem score is Dorin Marcu with an average score of 9.800000190734863
```

```
dir_df = df.withColumn("Director", split(df["Director"], ",\s*"))
dir_df = dir_df.select(explode(dir_df["Director"]).alias("Director"), dir_df["Hidden Gem Score"])
director_scores = dir_df.groupBy("Director").agg({"Hidden Gem Score": "avg"})
top_director = director_scores.orderBy(director_scores["avg(Hidden Gem Score)"].desc()).first()
print(f"The director with the highest average Hidden Gem Score is {top_director['Director']} with an average score c
```

```
The director with the highest average Hidden Gem Score is Dorin Marcu with an average score of 9.800000190734863
```

How many genres are there in the dataset?

```
genres_df = df.select(split(col("Genre"), ",\s*").alias("Genres_array"))
exploded_genres_df = genres_df.select(explode(col("Genres_array")).alias("Genres"))
num_genres = exploded_genres_df.select("Genres").distinct().count()
print("There are", num_genres, "genres in the dataset.")
```

```
There are 28 genres in the dataset.
```

✓ 0s completed at 8:35 PM