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-3.5.5/spark-3.5.5-bin-hadoop3.tgz
    !tar xf spark-3.5.5-bin-hadoop3.tgz
    !mv spark-3.5.5-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 import SQLContext

spark = SparkSession.builder.master(spark_url).appName('Netflix Rotten Tomatoes Metacritic IMDB').config('spark.ui.port'
```

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

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

```
file_path = '/content/netflix-rotten-tomatoes-metacritic-imdb.csv'

df = spark.read.option('header', True).csv(file_path)
```

```
df.show(5)
```

```
-----------------+------------+---------------+--------------------+--------------------+-----------+----------+---------
ıtry Availability|     Runtime|       Director|              Writer|              Actors|View Rating|IMDb Score|Rotten To
-----------------+------------+---------------+--------------------+--------------------+-----------+----------+---------
         Thailand|< 30 minutes|Tomas Alfredson|John Ajvide Lindq...|Kåre Hedebrant, P...|          R|       7.9|
           Canada|    1-2 hour|  Coky Giedroyc|       Caitlin Moran|Paddy Considine, ...|          R|       5.8|
           Canada|    1-2 hour|  Brendan Walsh|Brendan Walsh, Da...|Genesis Rodriguez...|    Unrated|       4.3|
.gium,Netherlands|< 30 minutes|           NULL|                NULL|Vahide Perçin, Go...|       NULL|       6.5|
ıuania,Poland,...|    1-2 hour|  Stephen Irwin|                NULL|        Ragga Gudrun|       NULL|       6.3|
-----------------+------------+---------------+--------------------+--------------------+-----------+----------+---------
```

```
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']
```

```
from pyspark.sql.functions import max, avg

df.select(max('Hidden Gem Score')).show()

df.select(avg('Hidden Gem Score')).show()
```

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

+---------------------+
|avg(Hidden Gem Score)|
+---------------------+
|    5.937551386501226|
+---------------------+
```

How many movies that are available in Korean Language?

```
from pyspark.sql.functions import col

df.filter(col('Languages').contains('Korea')).count()
```

```
735
```

Which director has the highest average hidden gem score?

```
df.groupBy('Director').agg(avg('Hidden Gem Score').alias('avg_score')).orderBy('avg_score', ascending=False).show(1)
```

```
+-----------+---------+
|   Director|avg_score|
+-----------+---------+
|Dorin Marcu|      9.8|
+-----------+---------+
only showing top 1 row
```

How many genres are there in the dataset?

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

df.select(explode(split(col('Genre'), ', ')).alias('genre')).distinct().count()
```

```
28
```