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-3.5.1/spark-3.5.1-bin-hadoop3.tgz
    !tar xf spark-3.5.1-bin-hadoop3.tgz
    !mv spark-3.5.1-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('Spark Assignment')\
        .config('spark.ui.port', '4040')\
        .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

```
sc = spark.sparkContext
```

```
spark
```

SparkSession - in-memory

SparkContext

Spark UI

```
Version
     v3.5.1
Master
     local
AppName
     Spark Assignment
```

```
sc
```

SparkContext

Spark UI

Version
    v3.5.1
Master
    local
AppName
    Spark Assignment

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

```
df = spark.read.csv(data_path, header=True, inferSchema=True)
```

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

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

hidden_gem_stats = df.agg(
    max("Hidden Gem Score").alias("Maximum Hidden Gem Score"),
    avg("Hidden Gem Score").alias("Average Hidden Gem Score")
)

hidden_gem_stats.show()
```

```
+------------------------+------------------------+
|Maximum Hidden Gem Score|Average Hidden Gem Score|
+------------------------+------------------------+
|                     9.8|       5.937551386501226|
+------------------------+------------------------+
```

How many movies that are available in Korea?

```
from pyspark.sql.functions import col

korea_movie_count = df.filter(col("Languages").contains("Korea")).count()
print(f"Number of movies available in Korea: {korea_movie_count}")
```

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

Which director has the highest average hidden gem score?

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

director_avg_scores = df.groupBy("Director") \
                        .agg(avg("Hidden Gem Score").alias("Average Hidden Gem Score")) \
                        .orderBy(col("Average Hidden Gem Score").desc())

top_director = director_avg_scores.first()

print(f"Director with the highest average hidden gem score: {top_director['Director']} (Score: {top_director['Average Hi
```

```
Director with the highest average hidden gem score: Dorin Marcu (Score: 9.8)
```

How many genres are there in the dataset?

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

genre_df = df.withColumn("Genre", explode(split(col("Genre"), ",\s*")))
genre_count = genre_df.select("Genre").distinct().count()

print(f"Total number of unique genres: {genre_count}")
```

```
Total number of unique genres: 28
```