```
!apt-get install openjdk-11-jdk-headless -qq > /dev/null
!pip install pyspark==3.5.1 -q
```

```
E: Failed to fetch http://security.ubuntu.com/ubuntu/pool/main/o/openjdk-lts/openjdk-11-jre-headless_11.0.28%2b6-1ubuntu1%7e22.0
E: Failed to fetch http://security.ubuntu.com/ubuntu/pool/main/o/openjdk-lts/openjdk-11-jdk-headless_11.0.28%2b6-1ubuntu1%7e22.0
E: Unable to fetch some archives, maybe run apt-get update or try with --fix-missing?
```

```
!java -version
!spark-submit --version
```

```
openjdk version "17.0.16" 2025-07-15
OpenJDK Runtime Environment (build 17.0.16+8-Ubuntu-0ubuntu122.04.1)
OpenJDK 64-Bit Server VM (build 17.0.16+8-Ubuntu-0ubuntu122.04.1, mixed mode, sharing)
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.5.1
      /_/

Using Scala version 2.12.18, OpenJDK 64-Bit Server VM, 17.0.16
Branch HEAD
Compiled by user heartsavior on 2024-02-15T11:24:58Z
Revision fd86f85e181fc2dc0f50a096855acf83a6cc5d9c
Url https://github.com/apache/spark
Type --help for more information.
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

1. Environment Setup & Imports

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, sum as _sum, round, split, array_contains
from pyspark.ml.feature import VectorAssembler, StandardScaler
from pyspark.ml.clustering import KMeans
```

2. Initialize Spark Session

```
spark = SparkSession.builder \
    .appName("MovieLens") \
    .getOrCreate()
```

3. Data Ingestion

```
movies_df = spark.read.csv("drive/MyDrive/spark/movies.csv",header=True,inferSchema=True)
```

```
movies_df.count()
```

```
62423
```

```
movies_df.show(5)
```

```
+-------+--------------------+--------------------+
|movieId|               title|              genres|
+-------+--------------------+--------------------+
|      1|    Toy Story (1995)|Adventure|Animati...|
|      2|      Jumanji (1995)|Adventure|Childre...|
|      3|Grumpier Old Men ...|      Comedy|Romance|
|      4|Waiting to Exhale...|Comedy|Drama|Romance|
|      5|Father of the Bri...|              Comedy|
+-------+--------------------+--------------------+
only showing top 5 rows
```

```
tags_df = spark.read.csv("drive/MyDrive/spark/tags.csv",header=True,inferSchema=True)
```

```
tags_df.count()
```

```
1093360
```

```
tags_df.show(5)
```

```
+------+-------+---------------+----------+
|userId|movieId|            tag| timestamp|
+------+-------+---------------+----------+
|     3|    260|        classic|1439472355|
|     3|    260|         sci-fi|1439472256|
|     4|   1732|    dark comedy|1573943598|
|     4|   1732| great dialogue|1573943604|
|     4|   7569|so bad it's good|1573943455|
+------+-------+---------------+----------+
only showing top 5 rows
```

```
ratings_df = spark.read.csv("drive/MyDrive/spark/ratings.csv", header=True, inferSchema=True)
```

```
ratings_df.count()
```

```
25000095
```

```
ratings_df.show(5)
```

```
+------+-------+------+----------+
|userId|movieId|rating| timestamp|
+------+-------+------+----------+
|     1|    296|   5.0|1147880044|
|     1|    306|   3.5|1147868817|
|     1|    307|   5.0|1147868828|
|     1|    665|   5.0|1147878820|
|     1|    899|   3.5|1147868510|
+------+-------+------+----------+
only showing top 5 rows
```

```
movies_df.printSchema()
ratings_df.printSchema()
tags_df.printSchema()
```

```
root
 |-- movieId: integer (nullable = true)
 |-- title: string (nullable = true)
 |-- genres: string (nullable = true)

root
 |-- userId: integer (nullable = true)
 |-- movieId: integer (nullable = true)
 |-- rating: double (nullable = true)
 |-- timestamp: integer (nullable = true)

root
 |-- userId: integer (nullable = true)
 |-- movieId: integer (nullable = true)
 |-- tag: string (nullable = true)
 |-- timestamp: string (nullable = true)
```

```
movies_df.describe().show()
ratings_df.describe().show()
tags_df.describe().show()
```

```
+-------+------------------+--------------------+------------------+
|summary|           movieId|               title|            genres|
+-------+------------------+--------------------+------------------+
|  count|             62423|               62423|             62423|
|   mean|122220.38764557935|                NULL|              NULL|
| stddev| 63264.74484425327|                NULL|              NULL|
|    min|                 1|"""BLOW THE NIGHT...|(no genres listed)|
|    max|            209171|     줄탁동시 (2012)|           Western|
+-------+------------------+--------------------+------------------+

+-------+-----------------+-----------------+-----------------+--------------------+
```

```
|summary|           userId|           movieId|            rating|           timestamp|
+-------+-----------------+------------------+------------------+--------------------+
|  count|         25000095|          25000095|          25000095|            25000095|
|   mean|81189.28115381162|21387.981943268616| 3.533854451353085|1.2156014431215513E9|
| stddev|46791.71589745776| 39198.86210105973|1.0607439611423535| 2.268758080595386E8|
|    min|                1|                 1|               0.5|           789652009|
|    max|           162541|            209171|               5.0|          1574327703|
+-------+-----------------+------------------+------------------+--------------------+

+-------+-----------------+------------------+--------------------+--------------------+
|summary|           userId|           movieId|                 tag|           timestamp|
+-------+-----------------+------------------+--------------------+--------------------+
|  count|          1093360|           1093360|             1093360|             1093360|
|   mean|67590.22463324065|  58492.7644389771|1.3406660588640274E7|1.4301154103339365E9|
| stddev|51521.13756056962|59687.312817477476| 4.563334194504871E8|1.1773852849277835E8|
|    min|                3|                 1| Alexander Skarsgård|  I am fat anyway"""|
|    max|           162534|            209063|          카운트다운|          1574316696|
+-------+-----------------+------------------+--------------------+--------------------+
```

4. Data Preprocessing & Cleaning

```
from pyspark.sql.functions import col, sum as _sum
```

```
# Helper function to check for null values
def null_count(df):
    return df.select([_sum(col(c).isNull().cast("int")).alias(c) for c in df.columns])
```

```
null_count(movies_df).show()
```

```
+-------+-----+------+
|movieId|title|genres|
+-------+-----+------+
|      0|    0|     0|
+-------+-----+------+
```

```
null_count(ratings_df).show()
```

```
+------+-------+------+---------+
|userId|movieId|rating|timestamp|
+------+-------+------+---------+
|     0|      0|     0|        0|
+------+-------+------+---------+
```

```
null_count(tags_df).show()
```

```
+------+-------+---+---------+
|userId|movieId|tag|timestamp|
+------+-------+---+---------+
|     0|      0|  0|        0|
+------+-------+---+---------+
```

Drop duplicate rows

```
movies_df = movies_df.dropDuplicates(["movieId"])
```

```
movies_df.count()
```

```
62423
```

```
ratings_df = ratings_df.dropDuplicates(["userId", "movieId"])
```

```
ratings_df.count()
```

```
25000095
```

```
tags_df = tags_df.dropDuplicates(["userId", "movieId", "tag"])
```

```
tags_df = tags_df.select("movieId", "tag")
```

```
tags_df.count()
```

```
1093360
```

Transformation on Movies

```
from pyspark.sql.functions import when,col,sum, split, avg, count, max, min, collect_list
```

```
from pyspark.sql.functions import regexp_extract, regexp_replace, lit
from pyspark.sql.types import IntegerType

movies_df = movies_df.withColumn("release_year",when(regexp_extract("title", r'\((\d{4})\)', 1).isNotNull(),
                                                     regexp_extract("title", r'\((\d{4})\)', 1).cast(IntegerType())).otherwise(
```

```
movies_df = movies_df.withColumn("release_year", when(col("release_year").isNull(), "Not Available").otherwise(col("release_yea
```

```
movies_df = movies_df.withColumn("title",  regexp_replace(col("title"), r'\s*\(\d{4}\)', ""))
```

```
movies_df = movies_df.withColumn("genres", when(col("genres").isNull(), "Unknown").otherwise(col("genres")))
```

```
movies_df = movies_df.withColumn("genres_array", split(col("genres"), r"\|"))
```

```
movies_df.show(5)
```

```
+-------+--------------------+--------------------+------------+--------------------+
|movieId|               title|              genres|release_year|        genres_array|
+-------+--------------------+--------------------+------------+--------------------+
|      1|           Toy Story|Adventure|Animati...|        1995|[Adventure, Anima...|
|      3|    Grumpier Old Men|      Comedy|Romance|        1995|   [Comedy, Romance]|
|      5|Father of the Bri...|              Comedy|        1995|           [Comedy]|
|      6|                Heat|Action|Crime|Thri...|        1995|[Action, Crime, T...|
|      9|        Sudden Death|              Action|        1995|            [Action]|
+-------+--------------------+--------------------+------------+--------------------+
only showing top 5 rows
```

```
null_count(movies_df).show()
```

```
+-------+-----+------+------------+------------+
|movieId|title|genres|release_year|genres_array|
+-------+-----+------+------------+------------+
|      0|    0|     0|           0|           0|
+-------+-----+------+------------+------------+
```

Transformation on Ratings

-> Creating new dataframe movie_ratings using ratings dataframe

```
ratings_aggregate = ratings_df.groupBy("movieId").agg(
    avg("rating").alias("avg_rating"),
    count("rating").alias("rating_count"),
    (min("timestamp")).alias("min_timestamp"),
    (max("timestamp")).alias("max_timestamp")
)
```

```
current_time_unix = int(spark.sql("SELECT UNIX_TIMESTAMP()").collect()[0][0])
ratings_aggregate = ratings_aggregate.withColumn("recency_days", round((lit(current_time_unix) - col("max_timestamp")) / (60 *
ratings_aggregate = ratings_aggregate.withColumn("rating_span_days", round((col("max_timestamp") - col("min_timestamp")) / (60
```

```
ratings_aggregate = ratings_aggregate.drop("min_timestamp")
ratings_aggregate = ratings_aggregate.drop("max_timestamp")
```

```
ratings_aggregate.show(5)
```

```
+-------+------------------+------------+------------+----------------+
|movieId|        avg_rating|rating_count|recency_days|rating_span_days|
+-------+------------------+------------+------------+----------------+
|  44022|3.2593627146699773|        4833|      2198.0|          4981.0|
|   1580|3.5817083457378187|       40308|      2197.0|          8174.0|
|   2366| 3.543409877319912|        6358|      2201.0|          7670.0|
|   8638|3.9717508278145695|        4832|      2200.0|          5614.0|
|    471|3.6579813752234034|       10631|      2197.0|          8665.0|
+-------+------------------+------------+------------+----------------+
only showing top 5 rows
```

Transformations on Tags

```
tags_aggregate = tags_df.groupBy("movieId").agg(
    count("tag").alias("tag_count"),
    collect_list("tag").alias("tag_list")
)
```

```
tags_aggregate.show(5)
```

```
+-------+---------+--------------------+
|movieId|tag_count|            tag_list|
+-------+---------+--------------------+
|      1|      697|[resourcefulness,...|
|      3|       29|[old, sequel, bes...|
|      5|       24|[sequel fever, gy...|
|      6|      621|[tense, Al Pacino...|
|      9|       13|[Jean-Claude Van ...|
+-------+---------+--------------------+
only showing top 5 rows
```

Join operations on movies_df, ratings_average and tags_aggregate

movies_df (contains movieId, title, genres, release_year, genres_array)

ratings_aggregate (contains movieId, avg_rating, rating_count, recency_days, rating_span_days)

tags_aggregate (contains movieId, tag_count, tag_list)

```
movies_clean = movies_df.join(ratings_aggregate, on="movieId", how="left")
```

```
movies_clean.show(5)
```

```
+-------+--------------------+--------------------+-----------+--------------------+------------------+------------+-----------
|movieId|               title|              genres|release_year|        genres_array|        avg_rating|rating_count|recency_day
+-------+--------------------+--------------------+-----------+--------------------+------------------+------------+-----------
|      1|           Toy Story|Adventure|Animati...|        1995|[Adventure, Anima...| 3.893707794587238|       57309|      2197.
|      3|    Grumpier Old Men|      Comedy|Romance|        1995|   [Comedy, Romance]| 3.142028126058963|       11804|      2207.
|      5|Father of the Bri...|              Comedy|        1995|           [Comedy]|3.0584343520573674|       11714|      2211.
|      6|                Heat|Action|Crime|Thri...|        1995|[Action, Crime, T...| 3.854908898649748|       24588|      2198.
|      9|        Sudden Death|              Action|        1995|           [Action]| 2.992050660199407|        3711|      2211.
+-------+--------------------+--------------------+-----------+--------------------+------------------+------------+-----------
only showing top 5 rows
```

```
movies_clean = movies_clean.join(tags_aggregate, on="movieId", how="left")
```

```
movies_clean.show(5)
```

```
+-------+--------------------+--------------------+-----------+--------------------+------------------+------------+-----------
|movieId|               title|              genres|release_year|        genres_array|        avg_rating|rating_count|recency_day
+-------+--------------------+--------------------+-----------+--------------------+------------------+------------+-----------
|      1|           Toy Story|Adventure|Animati...|        1995|[Adventure, Anima...| 3.893707794587238|       57309|      2197.
|      3|    Grumpier Old Men|      Comedy|Romance|        1995|   [Comedy, Romance]| 3.142028126058963|       11804|      2207.
|      5|Father of the Bri...|              Comedy|        1995|           [Comedy]|3.0584343520573674|       11714|      2211.
|      6|                Heat|Action|Crime|Thri...|        1995|[Action, Crime, T...| 3.854908898649748|       24588|      2198.
|      9|        Sudden Death|              Action|        1995|           [Action]| 2.992050660199407|        3711|      2211.
+-------+--------------------+--------------------+-----------+--------------------+------------------+------------+-----------
```

```
only showing top 5 rows
```

```
from pyspark.sql.functions import round

movies_clean = movies_clean.withColumn("avg_rating_rounded", round("avg_rating"))
```

```
movies_clean.show(5)
```

```
+-------+--------------------+--------------------+------------+--------------------+------------------+------------+-----------
|movieId|               title|              genres|release_year|        genres_array|        avg_rating|rating_count|recency_day
+-------+--------------------+--------------------+------------+--------------------+------------------+------------+-----------
|      1|           Toy Story|Adventure|Animati...|        1995|[Adventure, Anima...| 3.893707794587238|       57309|      2197.
|      3|    Grumpier Old Men|      Comedy|Romance|        1995|   [Comedy, Romance]| 3.142028126058963|       11804|      2207.
|      5|Father of the Bri...|              Comedy|        1995|            [Comedy]|3.0584343520573674|       11714|      2211.
|      6|                Heat|Action|Crime|Thri...|        1995|[Action, Crime, T...| 3.854908898649748|       24588|      2198.
|      9|        Sudden Death|              Action|        1995|            [Action]| 2.992050660199407|        3711|      2211.
+-------+--------------------+--------------------+------------+--------------------+------------------+------------+-----------
only showing top 5 rows
```

```
movies_clean= movies_clean.drop("avg_rating")
```

```
movies_clean = movies_clean.fillna({"avg_rating_rounded": 0, "rating_count": 0})
```

```
movies_clean.show(5)
```

```
+-------+--------------------+--------------------+------------+--------------------+------------+------------+----------------+
|movieId|               title|              genres|release_year|        genres_array|rating_count|recency_days|rating_span_days|
+-------+--------------------+--------------------+------------+--------------------+------------+------------+----------------+
|      1|           Toy Story|Adventure|Animati...|        1995|[Adventure, Anima...|       57309|      2197.0|          8697.0|
|      3|    Grumpier Old Men|      Comedy|Romance|        1995|   [Comedy, Romance]|       11804|      2207.0|          8683.0|
|      5|Father of the Bri...|              Comedy|        1995|            [Comedy]|       11714|      2211.0|          8679.0|
|      6|                Heat|Action|Crime|Thri...|        1995|[Action, Crime, T...|       24588|      2198.0|          8692.0|
|      9|        Sudden Death|              Action|        1995|            [Action]|        3711|      2211.0|          8679.0|
+-------+--------------------+--------------------+------------+--------------------+------------+------------+----------------+
only showing top 5 rows
```

```
from pyspark.sql.functions import array_contains
```

```
all_genres = ["Action","Comedy","Drama","Romance","Horror","Adventure","Children","Crime","Thriller","Animation","Fantasy","Sci
```

```
for g in all_genres:
    movies_clean = movies_clean.withColumn(f"genre_{g}", array_contains(col("genres_array"), g).cast("int"))
```

```
movies_clean.show(5)
```

```
--------+------------+-----------+-------------+------------+--------------+--------------+-----------+--------------+----------
e_Action|genre_Comedy|genre_Drama|genre_Romance|genre_Horror|genre_Adventure|genre_Children|genre_Crime|genre_Thriller|genre_Anim
--------+------------+-----------+-------------+------------+--------------+--------------+-----------+--------------+----------
       0|           1|          0|            0|           0|             1|             1|          0|             0|
       0|           1|          0|            1|           0|             0|             0|          0|             0|
       0|           1|          0|            0|           0|             0|             0|          0|             0|
       1|           0|          0|            0|           0|             0|             0|          1|             1|
       1|           0|          0|            0|           0|             0|             0|          0|             0|
--------+------------+-----------+-------------+------------+--------------+--------------+-----------+--------------+----------
```

Clustering - KMeans

```
from pyspark.ml.feature import VectorAssembler
```

```
feature_cols = [c for c in movies_clean.columns if c.startswith("genre_")] + ["avg_rating_rounded", "rating_count"]
```

```
assembler = VectorAssembler(
    inputCols=feature_cols,
```

```
    outputCol="features"
)
```

```
movies_final = assembler.transform(movies_clean)
```

```
movies_final.show()
```

```
-----+-----------+-------------+------------+---------------+-------------+-----------+-------------+---------------+----------
omedy|genre_Drama|genre_Romance|genre_Horror|genre_Adventure|genre_Children|genre_Crime|genre_Thriller|genre_Animation|genre_Fant
-----+-----------+-------------+------------+---------------+-------------+-----------+-------------+---------------+----------
    1|          0|            0|           0|              1|            1|          0|            0|              1|
    1|          0|            1|           0|              0|            0|          0|            0|              0|
    1|          0|            0|           0|              0|            0|          0|            0|              0|
    0|          0|            0|           0|              0|            0|          1|            1|              0|
    0|          0|            0|           0|              0|            0|          0|            0|              0|
    1|          0|            0|           1|              0|            0|          0|            0|              0|
    0|          0|            0|           0|              1|            1|          0|            0|              1|
    0|          0|            1|           0|              1|            0|          0|            0|              0|
    0|          1|            0|           0|              0|            0|          1|            0|              0|
    0|          1|            1|           0|              0|            0|          0|            0|              0|
    1|          0|            0|           0|              0|            0|          0|            0|              0|
    1|          1|            0|           0|              0|            0|          1|            1|              0|
    0|          1|            0|           1|              0|            0|          1|            1|              0|
    0|          1|            0|           0|              0|            0|          0|            0|              0|
    0|          1|            0|           0|              0|            1|          0|            0|              0|
    0|          1|            1|           0|              0|            0|          0|            0|              0|
    0|          1|            0|           0|              0|            0|          0|            0|              0|
    0|          1|            0|           0|              0|            1|          0|            0|              0|
    0|          1|            1|           0|              0|            0|          0|            0|              0|
    0|          0|            0|           0|              0|            0|          0|            0|              0|
-----+-----------+-------------+------------+---------------+-------------+-----------+-------------+---------------+----------
```

```
from pyspark.ml.clustering import KMeans
```

```
kmeans = KMeans(k=10, seed=42, featuresCol="features")
```

```
model = kmeans.fit(movies_final)
```

```
clusters = model.transform(movies_final)
```

```
clusters.select("movieId", "title", "prediction").show(20, truncate=False)
```

```
+-------+-----------------------------+----------+
|movieId|title                        |prediction|
+-------+-----------------------------+----------+
|1      |Toy Story                    |1         |
|3      |Grumpier Old Men             |8         |
|5      |Father of the Bride Part II  |8         |
|6      |Heat                         |6         |
|9      |Sudden Death                 |5         |
|12     |Dracula: Dead and Loving It  |5         |
|13     |Balto                        |9         |
|15     |Cutthroat Island             |5         |
|16     |Casino                       |7         |
|17     |Sense and Sensibility        |7         |
|19     |Ace Ventura: When Nature Calls|6         |
|20     |Money Train                  |5         |
|22     |Copycat                      |3         |
|26     |Othello                      |9         |
|27     |Now and Then                 |9         |
|28     |Persuasion                   |5         |
|31     |Dangerous Minds              |3         |
|34     |Babe                         |2         |
|35     |Carrington                   |9         |
|37     |Across the Sea of Time       |0         |
+-------+-----------------------------+----------+
only showing top 20 rows
```