

CP1370

Lab 3

John-Michael Woodrow

3.

Lab3 Python

File Edit View Run Help Last edit was now

Run all My Cluster Share Publish

< > + Code + Text

Just now (10s) 1 Python

```
from pyspark.sql import SparkSession

# Load the JSON file into a DataFrame
movies_df = spark.read.json("/FileStore/movies.json")

# Display the first 10 records
movies_df.show(10)

# Print the schema of the DataFrame
movies_df.printSchema()
```

▼ (2) Spark Jobs

▶ Job 0 View (Stages: 1/1)

▶ Job 1 View (Stages: 1/1)

▶ movies_df: pyspark.sql.dataframe.DataFrame = [actor_name: string, movie_title: string ... 1 more field]

actor_name	movie_title	produced_year
McClure, Marc (I)	Coach Carter	2005
McClure, Marc (I)	Superman II	1980
McClure, Marc (I)	Apollo 13	1995
McClure, Marc (I)	Superman	1978
McClure, Marc (I)	Back to the Future	1985
McClure, Marc (I)	Back to the Futur...	1990
Cooper, Chris (I)	Me, Myself & Irene	2000
Cooper, Chris (I)	October Sky	1999
Cooper, Chris (I)	Capote	2005
Cooper, Chris (I)	The Bourne Supremacy	2004

only showing top 10 rows

root

```
|-- actor_name: string (nullable = true)
|-- movie_title: string (nullable = true)
|-- produced_year: long (nullable = true)
```

[Shift+Enter] to run and move to next cell

[ctrl]+Shift+P] to open the command palette

4.

```

from pyspark.sql.functions import col, floor

# Add a 'decade' column based on the year
movies_with_decade_df = movies_df.withColumn("decade", (floor(col("produced_year") / 10) * 10))

# Display the first 10 records to verify the column was added
movies_with_decade_df.show(10)

```

▶ (1) Spark Jobs

▶ movies_with_decade_df: pyspark.sql.dataframe.DataFrame = [actor_name: string, movie_title: string ... 2 more fields]

actor_name	movie_title	produced_year	decade
McClure, Marc (I)	Coach Carter	2005	2000
McClure, Marc (I)	Superman II	1980	1980
McClure, Marc (I)	Apollo 13	1995	1990
McClure, Marc (I)	Superman	1978	1970
McClure, Marc (I)	Back to the Future	1985	1980
McClure, Marc (I)	Back to the Futur...	1990	1990
Cooper, Chris (I)	Me, Myself & Irene	2000	2000
Cooper, Chris (I)	October Sky	1999	1990
Cooper, Chris (I)	Capote	2005	2000
Cooper, Chris (I)	The Bourne Supremacy	2004	2000

only showing top 10 rows

[Shift+Enter] to run and move to next cell

5.

```

# Rename the first two columns to 'actor' and 'title'
renamed_movies_df = movies_with_decade_df.withColumnRenamed("actor_name", "actor").withColumnRenamed(
    "movie_title", "title")

# Display the first 10 records to verify the columns were renamed
renamed_movies_df.show(10)

```

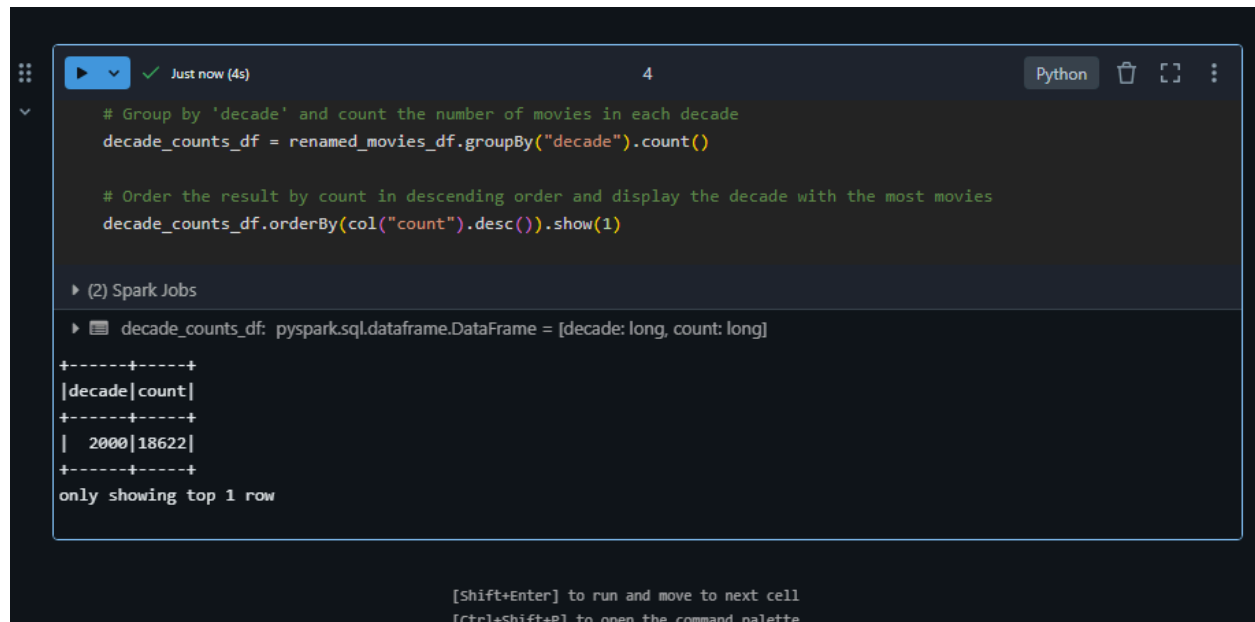
▶ (1) Spark Jobs

▶ renamed_movies_df: pyspark.sql.dataframe.DataFrame = [actor: string, title: string ... 2 more fields]

actor	title	produced_year	decade
McClure, Marc (I)	Coach Carter	2005	2000
McClure, Marc (I)	Superman II	1980	1980
McClure, Marc (I)	Apollo 13	1995	1990
McClure, Marc (I)	Superman	1978	1970
McClure, Marc (I)	Back to the Future	1985	1980
McClure, Marc (I)	Back to the Futur...	1990	1990
Cooper, Chris (I)	Me, Myself & Irene	2000	2000
Cooper, Chris (I)	October Sky	1999	1990
Cooper, Chris (I)	Capote	2005	2000
Cooper, Chris (I)	The Bourne Supremacy	2004	2000

only showing top 10 rows

6.



The screenshot shows a Jupyter Notebook interface with a Python kernel. The code in the cell groups movies by decade and counts them, then orders the results by count in descending order and shows the top 1 row. The output displays a single row for the year 2000 with a count of 18622.

```
# Group by 'decade' and count the number of movies in each decade
decade_counts_df = renamed_movies_df.groupBy("decade").count()

# Order the result by count in descending order and display the decade with the most movies
decade_counts_df.orderBy(col("count").desc()).show(1)
```

▶ (2) Spark Jobs

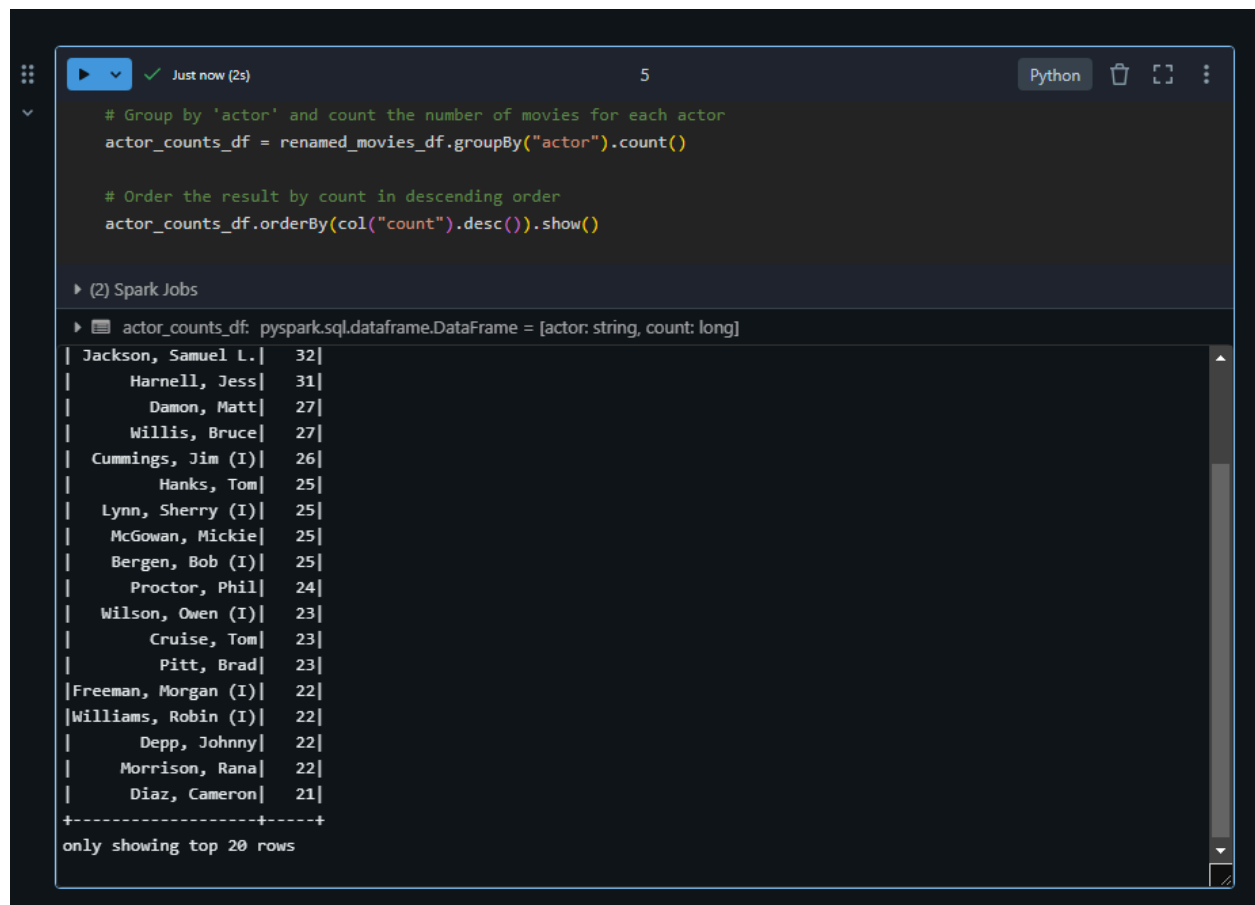
▶ `decade_counts_df: pyspark.sql.dataframe.DataFrame = [decade: long, count: long]`

decade	count
2000	18622

only showing top 1 row

[Shift+Enter] to run and move to next cell
[Ctrl+Shift+P] to open the command palette

7.



The screenshot shows a Jupyter Notebook interface with a Python kernel. The code in the cell groups movies by actor and counts them, then orders the results by count in descending order and shows the top 20 rows. The output displays a list of actors and their corresponding movie counts, starting with Samuel L. Jackson at 32.

```
# Group by 'actor' and count the number of movies for each actor
actor_counts_df = renamed_movies_df.groupBy("actor").count()

# Order the result by count in descending order
actor_counts_df.orderBy(col("count").desc()).show()
```

▶ (2) Spark Jobs

▶ `actor_counts_df: pyspark.sql.dataframe.DataFrame = [actor: string, count: long]`

actor	count
Jackson, Samuel L.	32
Harnell, Jess	31
Damon, Matt	27
Willis, Bruce	27
Cummings, Jim (I)	26
Hanks, Tom	25
Lynn, Sherry (I)	25
McGowan, Mickie	25
Bergen, Bob (I)	25
Proctor, Phil	24
Wilson, Owen (I)	23
Cruise, Tom	23
Pitt, Brad	23
Freeman, Morgan (I)	22
Williams, Robin (I)	22
Depp, Johnny	22
Morrison, Rana	22
Diaz, Cameron	21

only showing top 20 rows