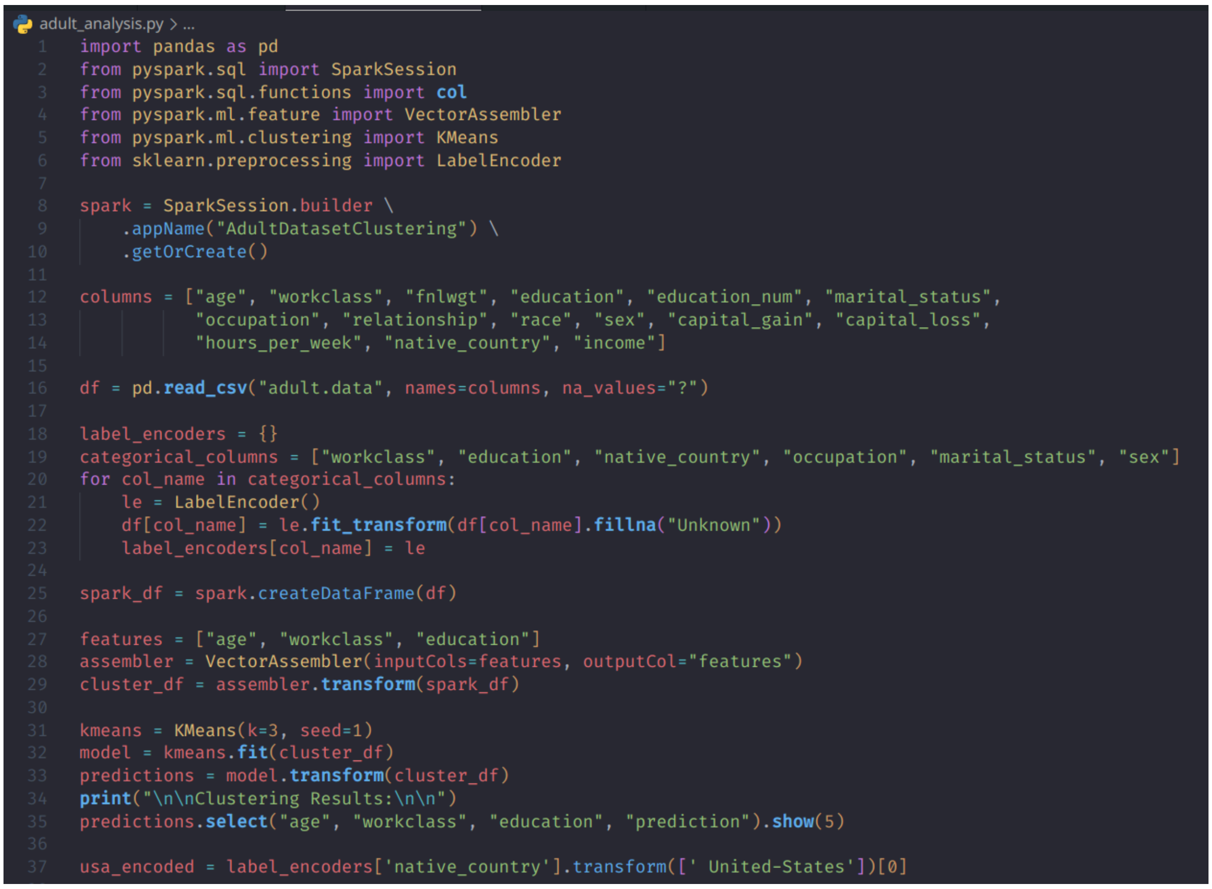
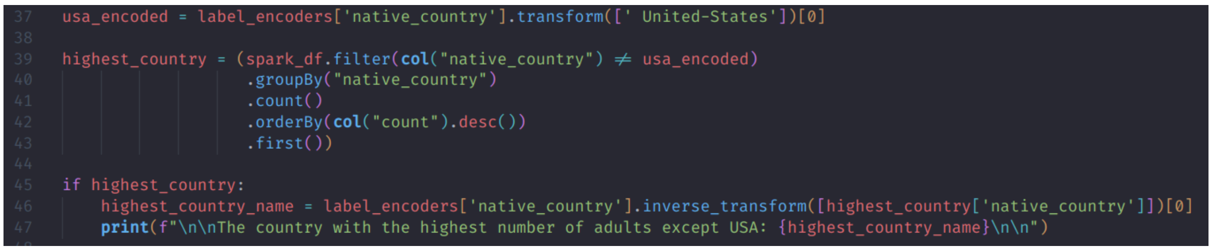
Sridhar Tuli
Assignment 9 Spark

Question:
You can find adult dataset of UCI here (adult.data)
1. Write Python/Java code to cluster the data on at least 3 attributes (for example: age, workclass, education).
2. Execute the above code in Apache Spark.
3. Using spark transformations/actions, find
i) The country from which the highest number of adults are there except USA.
ii) No of people who are at least Masters and works in “Tech-support”.
iii) No of unmarried male who works in “Local_govt”.

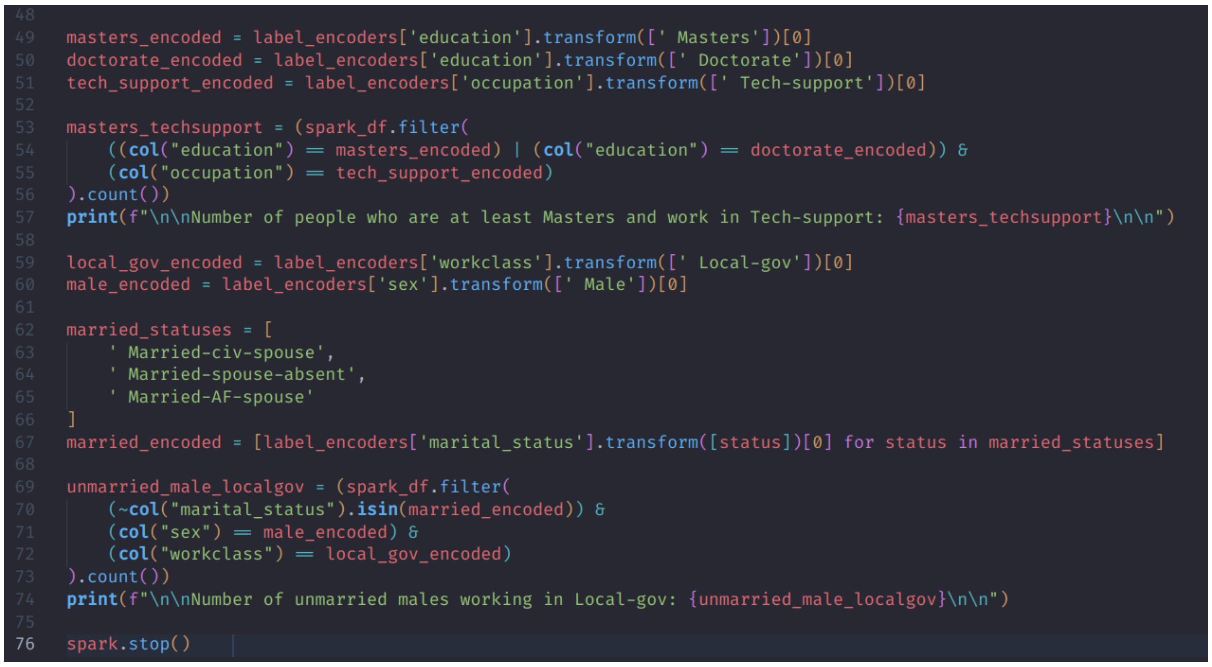


```python
import pandas as pd
from pyspark.sql import SparkSession
from pyspark.sql.functions import col
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.clustering import KMeans
from sklearn.preprocessing import LabelEncoder

spark = SparkSession.builder \
    .appName("AdultDatasetClustering") \
    .getOrCreate()

columns = ["age", "workclass", "fnlwgt", "education", "education_num", "marital_status",
           "occupation", "relationship", "race", "sex", "capital_gain", "capital_loss",
           "hours_per_week", "native_country", "income"]

df = pd.read_csv("adult.data", names=columns, na_values="?")

label_encoders = {}
categorical_columns = ["workclass", "education", "native_country", "occupation", "marital_status", "sex"]
for col_name in categorical_columns:
    le = LabelEncoder()
    df[col_name] = le.fit_transform(df[col_name].fillna("Unknown"))
    label_encoders[col_name] = le

spark_df = spark.createDataFrame(df)

features = ["age", "workclass", "education"]
assembler = VectorAssembler(inputCols=features, outputCol="features")
cluster_df = assembler.transform(spark_df)

kmeans = KMeans(k=3, seed=1)
model = kmeans.fit(cluster_df)
predictions = model.transform(cluster_df)
print("\n\nClustering Results:\n\n")
predictions.select("age", "workclass", "education", "prediction").show(5)

usa_encoded = label_encoders['native_country'].transform([' United-States'])[0]
```



```python
usa_encoded = label_encoders['native_country'].transform([' United-States'])[0]
```

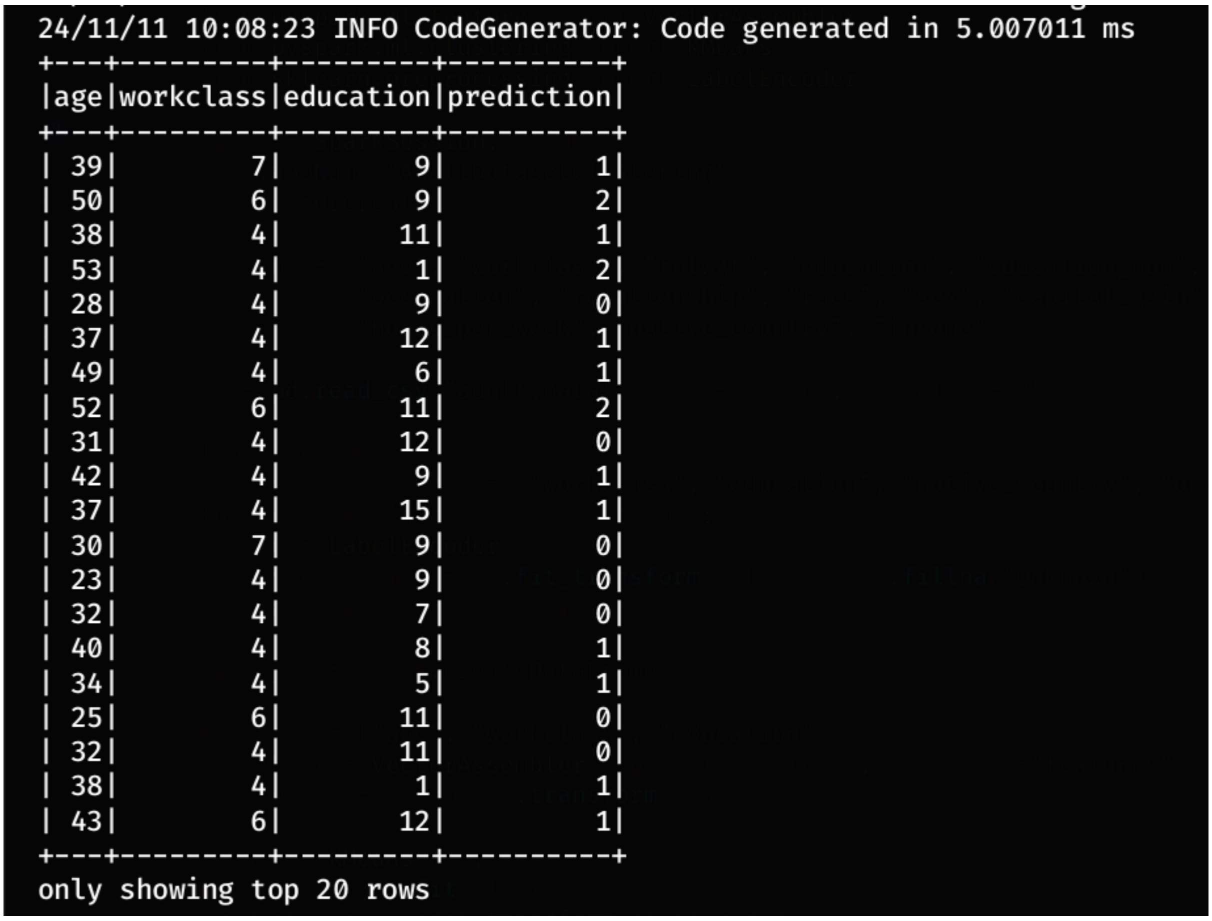
```python
highest_country = (spark_df.filter(col("native_country") != usa_encoded)
                   .groupBy("native_country")
                   .count()
                   .orderBy(col("count").desc())
                   .first())

if highest_country:
    highest_country_name = label_encoders['native_country'].inverse_transform([highest_country['native_country']])[0]
    print(f"\n\nThe country with the highest number of adults except USA: {highest_country_name}\n\n")
```



```python

masters_encoded = label_encoders['education'].transform([' Masters'])[0]
doctorate_encoded = label_encoders['education'].transform([' Doctorate'])[0]
tech_support_encoded = label_encoders['occupation'].transform([' Tech-support'])[0]

masters_techsupport = (spark_df.filter(
    ((col("education") == masters_encoded) | (col("education") == doctorate_encoded)) &
    (col("occupation") == tech_support_encoded)
).count())
print(f"\n\nNumber of people who are at least Masters and work in Tech-support: {masters_techsupport}\n\n")

local_gov_encoded = label_encoders['workclass'].transform([' Local-gov'])[0]
male_encoded = label_encoders['sex'].transform([' Male'])[0]

married_statuses = [
    ' Married-civ-spouse',
    ' Married-spouse-absent',
    ' Married-AF-spouse'
]
married_encoded = [label_encoders['marital_status'].transform([status])[0] for status in married_statuses]

unmarried_male_localgov = (spark_df.filter(
    (~col("marital_status").isin(married_encoded)) &
    (col("sex") == male_encoded) &
    (col("workclass") == local_gov_encoded)
).count())
print(f"\n\nNumber of unmarried males working in Local-gov: {unmarried_male_localgov}\n\n")

spark.stop()
```

OUTPUT:

Write Python/Java code to cluster the data on at least 3 attributes (for example: age, workclass, education).

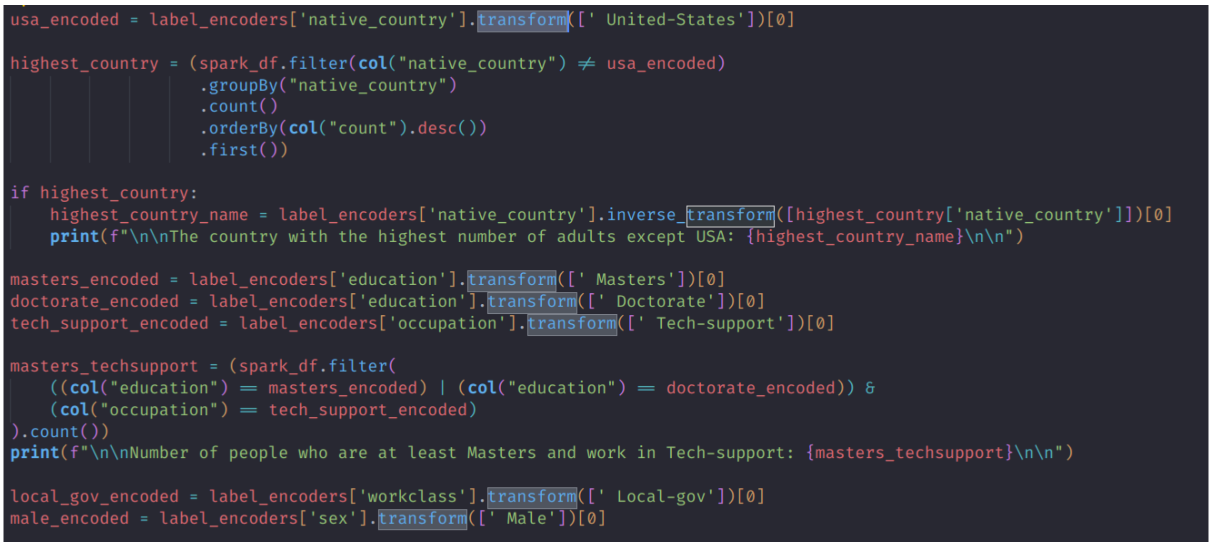


```
24/11/11 10:08:23 INFO CodeGenerator: Code generated in 5.007011 ms
+---+---------+---------+----------+
|age|workclass|education|prediction|
+---+---------+---------+----------+
| 39|        7|        9|         1|
| 50|        6|        9|         2|
| 38|        4|       11|         1|
| 53|        4|        1|         2|
| 28|        4|        9|         0|
| 37|        4|       12|         1|
| 49|        4|        6|         1|
| 52|        6|       11|         2|
| 31|        4|       12|         0|
| 42|        4|        9|         1|
| 37|        4|       15|         1|
| 30|        7|        9|         0|
| 23|        4|        9|         0|
| 32|        4|        7|         0|
| 40|        4|        8|         1|
| 34|        4|        5|         1|
| 25|        6|       11|         0|
| 32|        4|       11|         0|
| 38|        4|        1|         1|
| 43|        6|       12|         1|
+---+---------+---------+----------+
only showing top 20 rows
```

3. Using spark transformations/actions, find

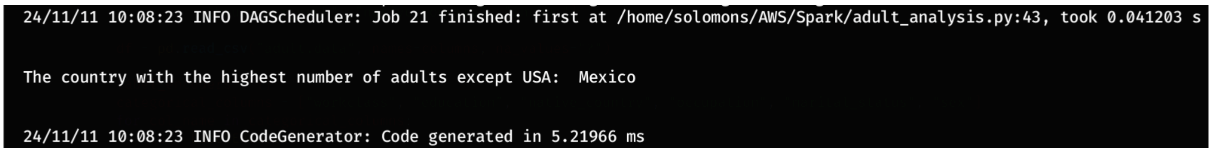
i) The country from which the highest number of adults are there except USA.

Corresponding Code:



```python
usa_encoded = label_encoders['native_country'].transform([' United-States'])[0]

highest_country = (spark_df.filter(col("native_country") != usa_encoded)
                   .groupBy("native_country")
                   .count()
                   .orderBy(col("count").desc())
                   .first())

if highest_country:
    highest_country_name = label_encoders['native_country'].inverse_transform([highest_country['native_country']])[0]
    print(f"\n\nThe country with the highest number of adults except USA: {highest_country_name}\n\n")

masters_encoded = label_encoders['education'].transform([' Masters'])[0]
doctorate_encoded = label_encoders['education'].transform([' Doctorate'])[0]
tech_support_encoded = label_encoders['occupation'].transform([' Tech-support'])[0]

masters_techsupport = (spark_df.filter(
    ((col("education") == masters_encoded) | (col("education") == doctorate_encoded)) &
    (col("occupation") == tech_support_encoded)
).count())
print(f"\n\nNumber of people who are at least Masters and work in Tech-support: {masters_techsupport}\n\n")

local_gov_encoded = label_encoders['workclass'].transform([' Local-gov'])[0]
male_encoded = label_encoders['sex'].transform([' Male'])[0]
```

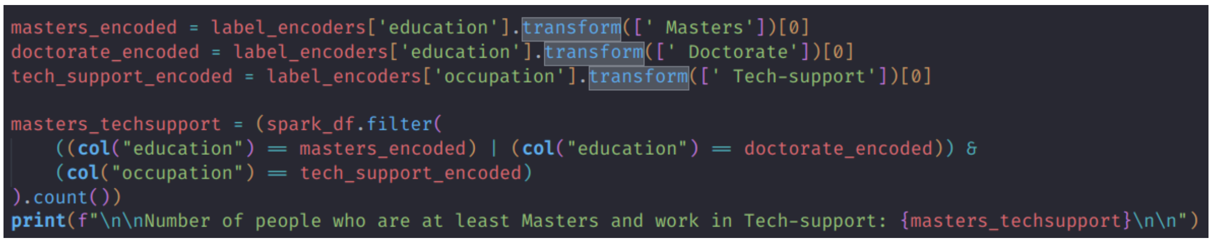
Output:



```
24/11/11 10:08:23 INFO DAGScheduler: Job 21 finished: first at /home/solomons/AWS/Spark/adult_analysis.py:43, took 0.041203 s


The country with the highest number of adults except USA:  Mexico


24/11/11 10:08:23 INFO CodeGenerator: Code generated in 5.21966 ms
```

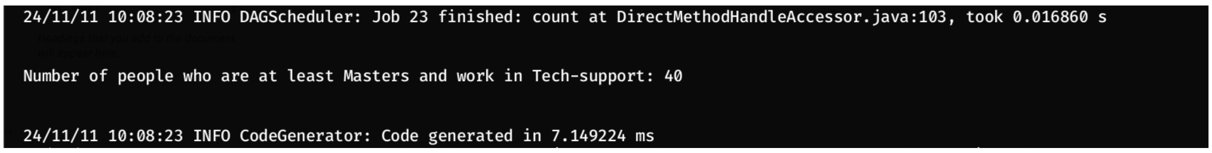
ii) No of people who are at least Masters and works in “Tech-support”.

Corresponding Code:



```python
masters_encoded = label_encoders['education'].transform([' Masters'])[0]
doctorate_encoded = label_encoders['education'].transform([' Doctorate'])[0]
tech_support_encoded = label_encoders['occupation'].transform([' Tech-support'])[0]

masters_techsupport = (spark_df.filter(
    ((col("education") == masters_encoded) | (col("education") == doctorate_encoded)) &
    (col("occupation") == tech_support_encoded)
).count())
print(f"\n\nNumber of people who are at least Masters and work in Tech-support: {masters_techsupport}\n\n")
```

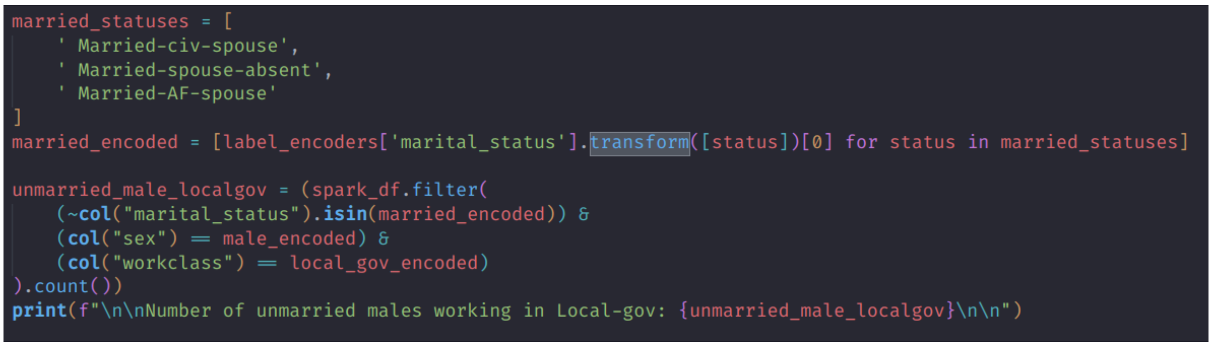
Output:



```
24/11/11 10:08:23 INFO DAGScheduler: Job 23 finished: count at DirectMethodHandleAccessor.java:103, took 0.016860 s


Number of people who are at least Masters and work in Tech-support: 40


24/11/11 10:08:23 INFO CodeGenerator: Code generated in 7.149224 ms
```

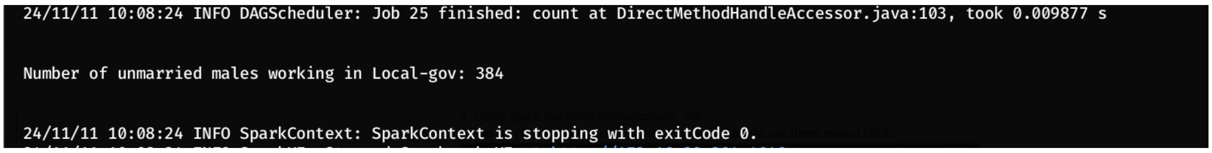
iii) No of unmarried male who works in “Local_govt”.

Corresponding Code:



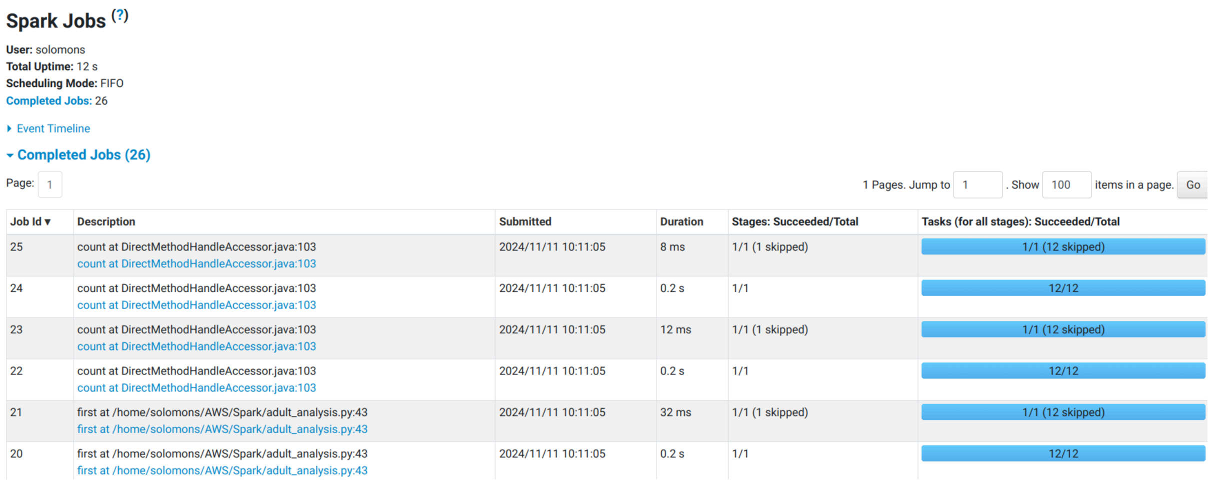
```
married_statuses = [
    ' Married-civ-spouse',
    ' Married-spouse-absent',
    ' Married-AF-spouse'
]
married_encoded = [label_encoders['marital_status'].transform([status])[0] for status in married_statuses]

unmarried_male_localgov = (spark_df.filter(
    (~col("marital_status").isin(married_encoded)) &
    (col("sex") == male_encoded) &
    (col("workclass") == local_gov_encoded)
).count())
print(f"\n\nNumber of unmarried males working in Local-gov: {unmarried_male_localgov}\n\n")
```

Output:



```
 24/11/11 10:08:24 INFO DAGScheduler: Job 25 finished: count at DirectMethodHandleAccessor.java:103, took 0.009877 s


 Number of unmarried males working in Local-gov: 384


 24/11/11 10:08:24 INFO SparkContext: SparkContext is stopping with exitCode 0.
```

AT SPARK WEBSITE



Spark Jobs (?)

User: solomons
Total Uptime: 12 s
Scheduling Mode: FIFO
Completed Jobs: 26

▸ Event Timeline

▾ Completed Jobs (26)

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
25	count at DirectMethodHandleAccessor.java:103 count at DirectMethodHandleAccessor.java:103	2024/11/11 10:11:05	8 ms	1/1 (1 skipped)	1/1 (12 skipped)
24	count at DirectMethodHandleAccessor.java:103 count at DirectMethodHandleAccessor.java:103	2024/11/11 10:11:05	0.2 s	1/1	12/12
23	count at DirectMethodHandleAccessor.java:103 count at DirectMethodHandleAccessor.java:103	2024/11/11 10:11:05	12 ms	1/1 (1 skipped)	1/1 (12 skipped)
22	count at DirectMethodHandleAccessor.java:103 count at DirectMethodHandleAccessor.java:103	2024/11/11 10:11:05	0.2 s	1/1	12/12
21	first at /home/solomons/AWS/Spark/adult_analysis.py:43 first at /home/solomons/AWS/Spark/adult_analysis.py:43	2024/11/11 10:11:05	32 ms	1/1 (1 skipped)	1/1 (12 skipped)
20	first at /home/solomons/AWS/Spark/adult_analysis.py:43 first at /home/solomons/AWS/Spark/adult_analysis.py:43	2024/11/11 10:11:05	0.2 s	1/1	12/12

We are getting the information about the task happening
My laptop has 6 Cores, therefore 12 threads
Thats why it’s showing 1/12 in the image, telling that 1 task has been completed out of 12 parallely running ones.

===========================END OF FILE================================