Big Data Analysis using PySpark on NYC Taxi (Jan 2015)

Introduction / Objective

Analyze NYC Taxi Data using PySpark to demonstrate scalable data processing.

```
# pip install pyspark
```

```
Requirement already satisfied: pyspark in /usr/local/lib/python3.11/dist-packages (3.5.1)
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.11/dist-packages (from pyspark) (0.10.9.7)
```

Double-click (or enter) to edit

Load the CSV file

```
data = spark.read.csv("/content/yellow_tripdata_2015-01.csv",header=True, inferSc

data.printSchema()
data.show(5)
```

```
root
 |-- VendorID: integer (nullable = true)
 |-- tpep_pickup_datetime: timestamp (nullable = true)
 |-- tpep_dropoff_datetime: timestamp (nullable = true)
 |-- passenger_count: integer (nullable = true)
 |-- trip_distance: double (nullable = true)
 |-- pickup_longitude: double (nullable = true)
 |-- pickup_latitude: double (nullable = true)
 |-- RateCodeID: integer (nullable = true)
 |-- store_and_fwd_flag: string (nullable = true)
 |-- dropoff_longitude: double (nullable = true)
 |-- dropoff_latitude: double (nullable = true)
```

```
 |-- payment_type: integer (nullable = true)
 |-- fare_amount: double (nullable = true)
 |-- extra: double (nullable = true)
 |-- mta_tax: double (nullable = true)
 |-- tip_amount: double (nullable = true)
 |-- tolls_amount: double (nullable = true)
 |-- improvement_surcharge: double (nullable = true)
 |-- total_amount: double (nullable = true)

+--------+--------------------+---------------------+---------------+-------------+------------------+------------------+-------
|VendorID|tpep_pickup_datetime|tpep_dropoff_datetime|passenger_count|trip_distance|  pickup_longitude|   pickup_latitude|RateCod
+--------+--------------------+---------------------+---------------+-------------+------------------+------------------+-------
|       2| 2015-01-15 19:05:39|  2015-01-15 19:23:42|              1|         1.59|  -73.993896484375|  40.7501106262207|
|       1| 2015-01-10 20:33:38|  2015-01-10 20:53:28|              1|          3.3|-74.00164794921875|  40.7242431640625|
|       1| 2015-01-10 20:33:38|  2015-01-10 20:43:41|              1|          1.8|-73.96334075927734| 40.80278778076172|
|       1| 2015-01-10 20:33:39|  2015-01-10 20:35:31|              1|          0.5|-74.00908660888672| 40.71381759643555|
|       1| 2015-01-10 20:33:39|  2015-01-10 20:52:58|              1|          3.0|-73.97117614746094|40.762428283691406|
+--------+--------------------+---------------------+---------------+-------------+------------------+------------------+-------
only showing top 5 rows
```

Generate

Clean the data

Close

```
df_cleaned = data.select("tpep_pickup_datetime", "total_amount", "passenger_count") \
               .dropna()
df_cleaned.show(5)
df_cleaned.printSchema()
```

```
+--------------------+------------+---------------+
|tpep_pickup_datetime|total_amount|passenger_count|
+--------------------+------------+---------------+
| 2015-01-15 19:05:39|       17.05|              1|
| 2015-01-10 20:33:38|        17.8|              1|
| 2015-01-10 20:33:38|        10.8|              1|
| 2015-01-10 20:33:39|         4.8|              1|
| 2015-01-10 20:33:39|        16.3|              1|
```

```
+--------------------+------------+---------------+
only showing top 5 rows

root
 |-- tpep_pickup_datetime: timestamp (nullable = true)
 |-- total_amount: double (nullable = true)
 |-- passenger_count: integer (nullable = true)
```

Feature Engineering: Pickup Hour

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

df_with_hour = df_cleaned.withColumn("pickup_hour", hour(col("tpep_pickup_datetime")))
df_with_hour.show(5)
```

```
+--------------------+------------+---------------+-----------+
|tpep_pickup_datetime|total_amount|passenger_count|pickup_hour|
+--------------------+------------+---------------+-----------+
| 2015-01-15 19:05:39|       17.05|              1|         19|
| 2015-01-10 20:33:38|        17.8|              1|         20|
| 2015-01-10 20:33:38|        10.8|              1|         20|
| 2015-01-10 20:33:39|         4.8|              1|         20|
| 2015-01-10 20:33:39|        16.3|              1|         20|
+--------------------+------------+---------------+-----------+
only showing top 5 rows
```

Group by pickup hour

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

hourly_summary = df_with_hour.groupBy("pickup_hour") \
    .agg(
```

```
        avg("total_amount").alias("avg_fare"),
        count("*").alias("trip_count")
    ).orderBy("pickup_hour")
hourly_summary.show()
```

```
+-----------+------------------+----------+
|pickup_hour|          avg_fare|trip_count|
+-----------+------------------+----------+
|          0|15.886732786137715|      4386|
|          1|15.418101395032073|      2939|
|          2| 15.46007056451597|      1984|
|          3| 15.63148639681478|      1507|
|          4|17.949745347698357|      1021|
|          5| 19.80913300492608|      1015|
|          6|15.995419145483885|      1849|
|          7| 14.30300642753471|      4823|
|          8| 13.98042451853378|      4829|
|          9|14.040789277736415|      5372|
|         10|13.732300063304434|      4739|
|         11|13.760901268115767|      4416|
|         12|13.939763723574393|      5629|
|         13|14.124811643835596|      5256|
|         14|15.106398791541157|      6620|
|         15| 15.17538372285789|      6033|
|         16| 15.27152296819803|      5660|
|         17|14.768366585563916|      6165|
|         18|14.674131596984688|      7295|
|         19|14.351606236403747|      8274|
+-----------+------------------+----------+
only showing top 20 rows
```

Get peak hour

```
peak_hour = hourly_summary.orderBy(col("trip_count").desc()).limit(1)
peak_hour.show()
```

```
+-----------+------------------+----------+
|pickup_hour|          avg_fare|trip_count|
+-----------+------------------+----------+
|         19|14.351606236403747|      8274|
+-----------+------------------+----------+
```

```
pandas_df = hourly_summary.toPandas()
pandas_df
```

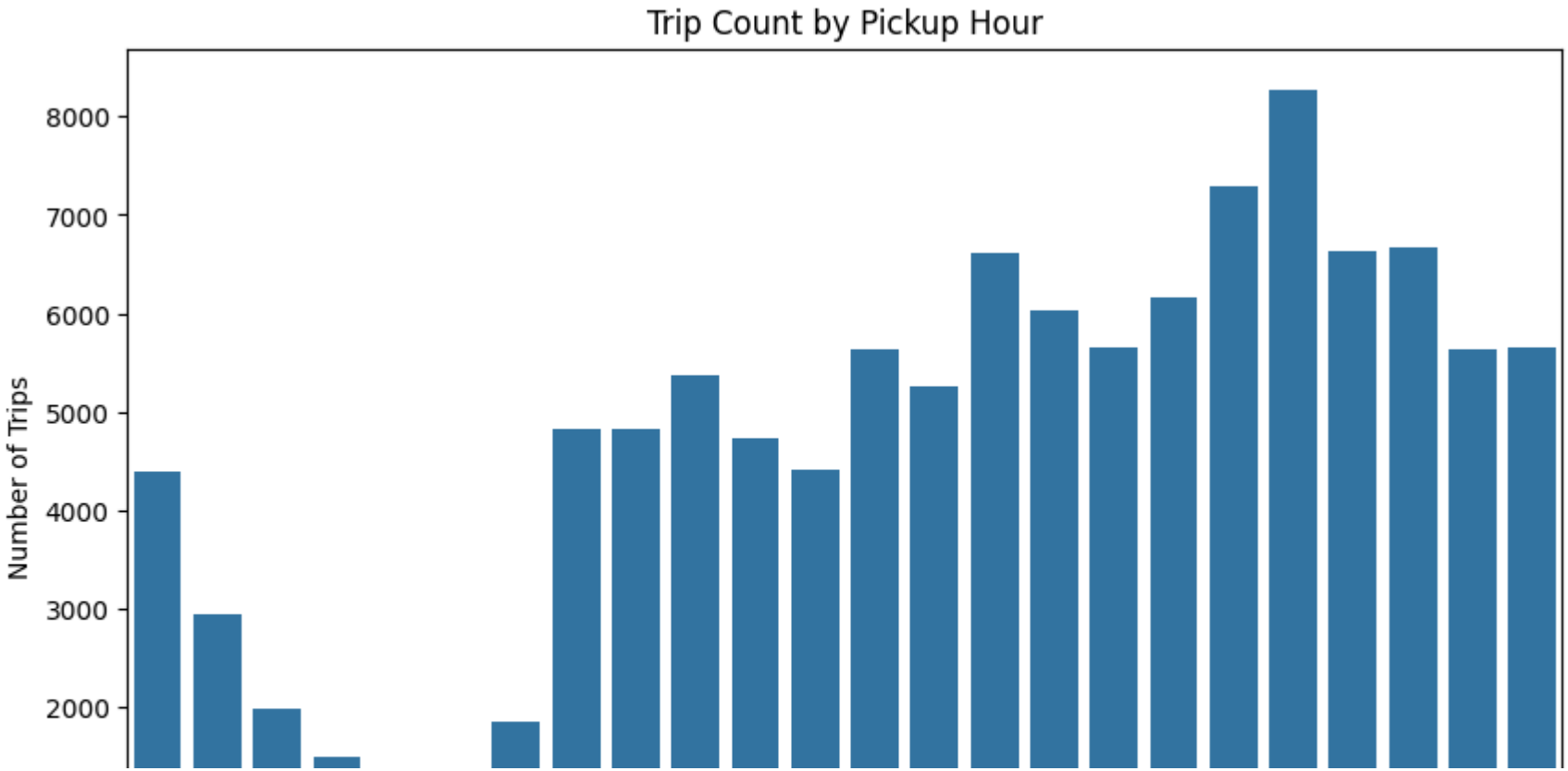
	pickup_hour	avg_fare	trip_count
0	0	15.886733	4386
1	1	15.418101	2939
2	2	15.460071	1984
3	3	15.631486	1507
4	4	17.949745	1021
5	5	19.809133	1015
6	6	15.995419	1849
7	7	14.303006	4823
8	8	13.980425	4829
9	9	14.040789	5372
10	10	13.732300	4739
11	11	13.760901	4416
12	12	13.939764	5629
13	13	14.124812	5256
14	14	15.106399	6620
15	15	15.175384	6033
16	16	15.271523	5660
17	17	14.768367	6165
18	18	14.674132	7295
19	19	14.351606	8274
20	20	14.533636	6629
21	21	14.729137	6674

22	22	15.449173	5635
23	23	15.908109	5663

Next steps: Generate code with pandas_df | View recommended plots | New interactive sheet

```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10,6))
sns.barplot(x="pickup_hour", y="trip_count", data=pandas_df)
plt.title("Trip Count by Pickup Hour")
plt.xlabel("Hour of Day")
plt.ylabel("Number of Trips")
plt.show()
```



** Average Fare per Passenger Count**

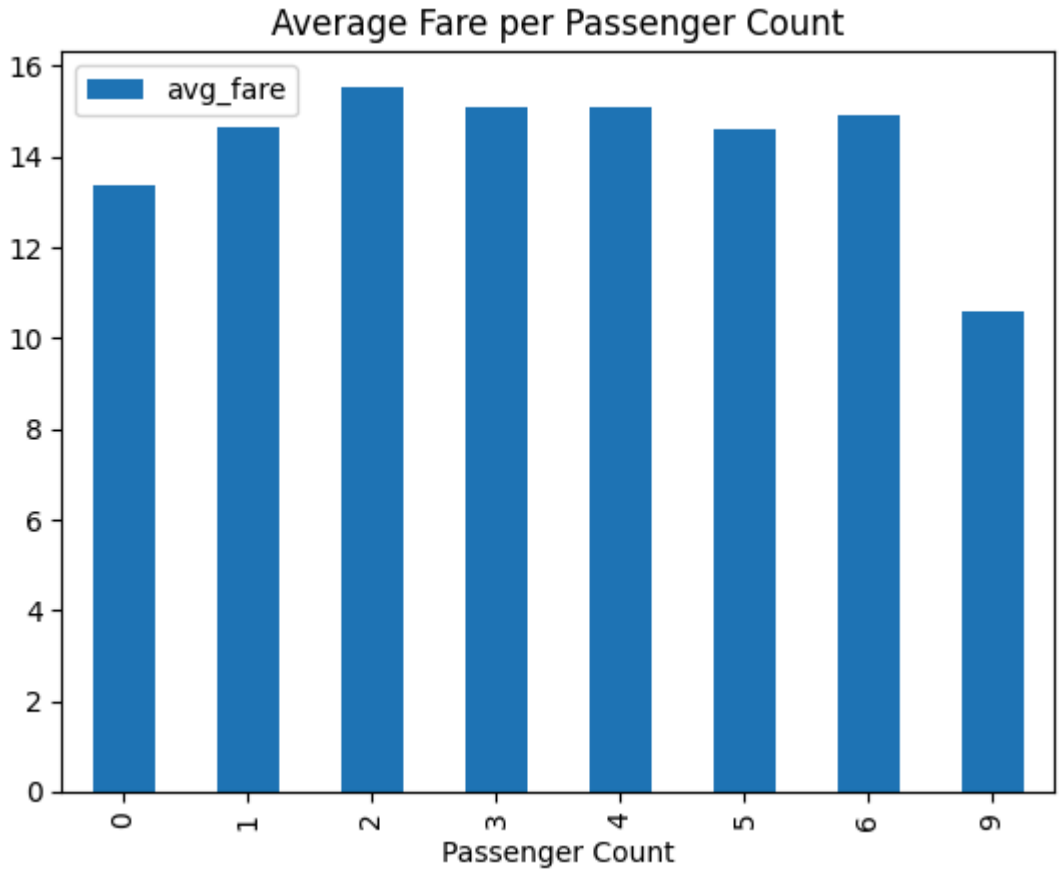
```
df_passenger = df_cleaned.groupBy("passenger_count") \
    .agg(avg("total_amount").alias("avg_fare"), count("*").alias("trip_count")) \
    .orderBy("passenger_count")
df_passenger.show()
```

```
+---------------+------------------+----------+
|passenger_count|          avg_fare|trip_count|
+---------------+------------------+----------+
|              0|13.370597014925375|        67|
|              1|14.648879103481534|     80846|
|              2| 15.53719369286493|     16299|
|              3|15.101434901878058|      4739|
```

```
|              4|15.092452243447163|      2251|
|              5|14.602242945183532|      6166|
|              6|14.911565281898957|      4044|
|              9|              10.6|         1|
+---------------+------------------+----------+
```

```
df_passenger.toPandas().plot(kind="bar", x="passenger_count", y="avg_fare", title="Average Fare per Passenger Count")
plt.xlabel("Passenger Count")
```

```
Text(0.5, 0, 'Passenger Count')
```



Trip Distance vs Fare Analysis python Copy code

```
df_distance = data.select("trip_distance", "total_amount").dropna()
df_distance = df_distance.filter((col("trip_distance") > 0) & (col("total_amount") > 0))

df_distance_stats = df_distance.agg(
    avg("trip_distance").alias("avg_distance"),
    avg("total_amount").alias("avg_fare")
)
df_distance_stats.show()
```