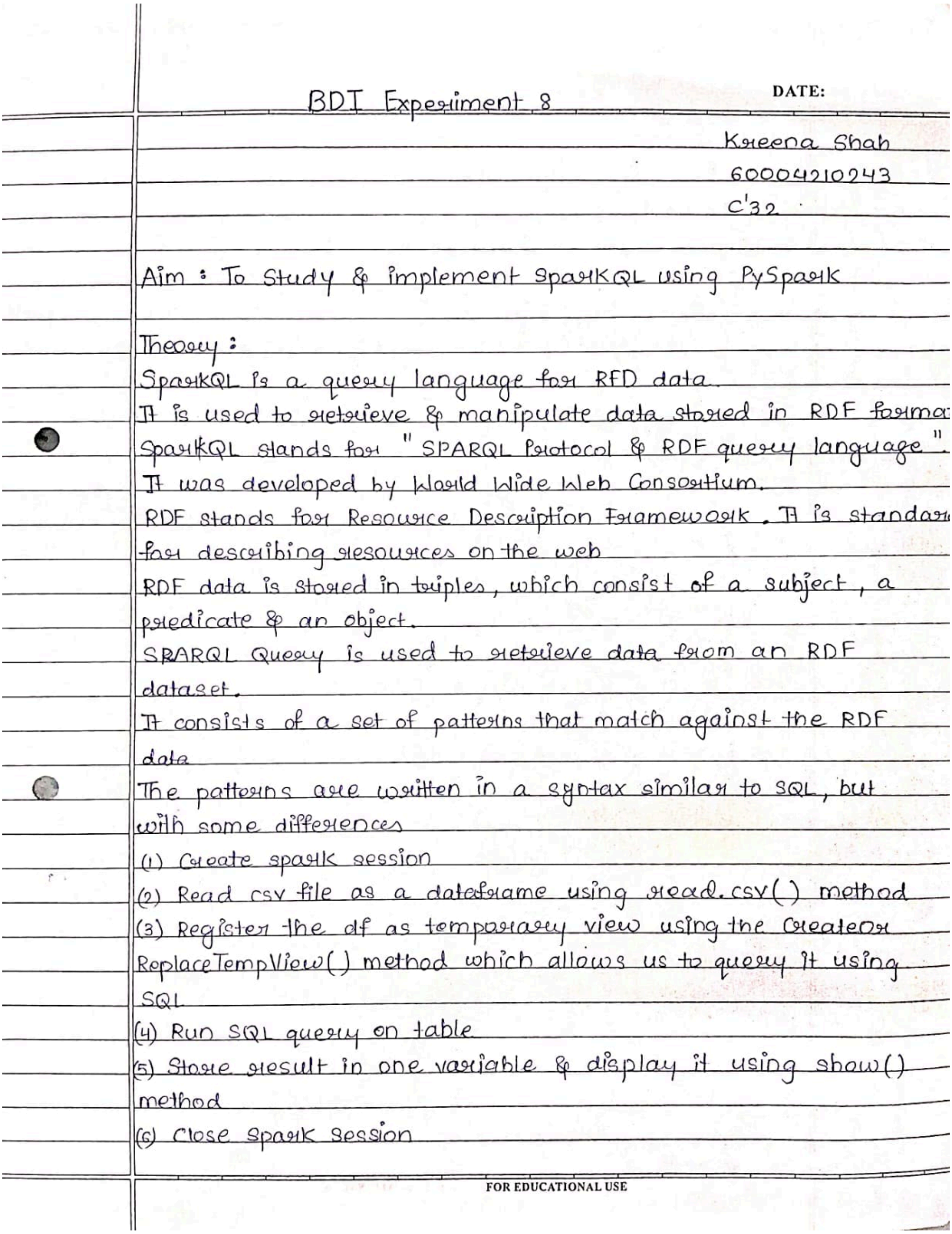
Experiment 8

Aim : To study and implement SparkQL using PySpark.



BDI Experiment 8 DATE:

Kreena Shah
60004210243
C'32

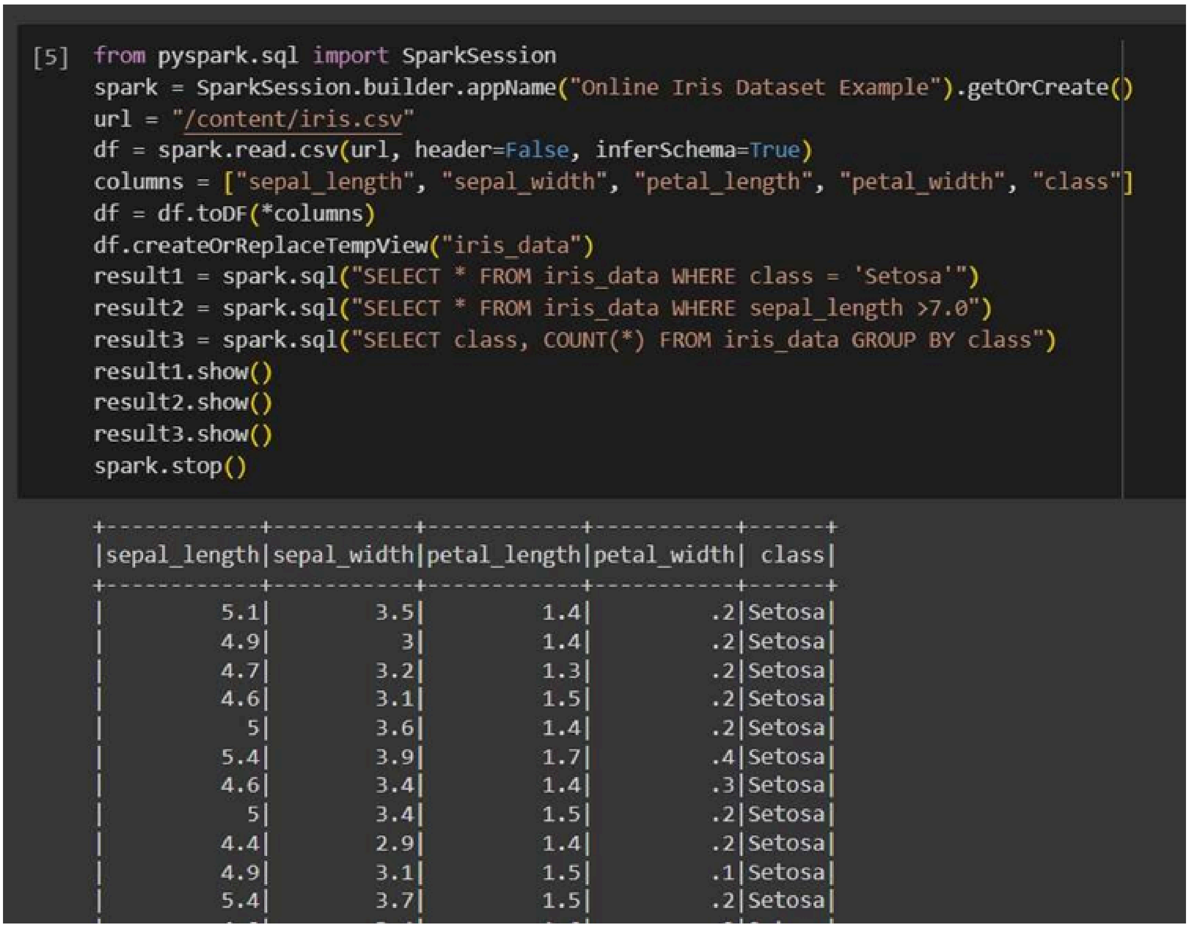
Aim : To Study & implement SparkQL using PySpark

Theory :

SparkQL is a query language for RFD data.
It is used to retrieve & manipulate data stored in RDF forma
SparkQL stands for "SPARQL Protocol & RDF query language".
It was developed by World Wide Web Consortium.
RDF stands for Resource Description Framework. It is standar
for describing resources on the web
RDF data is stored in triples, which consist of a subject, a predicate & an object.
SPARQL Query is used to retrieve data from an RDF dataset.
It consists of a set of patterns that match against the RDF data
The patterns are written in a syntax similar to SQL, but with some differences

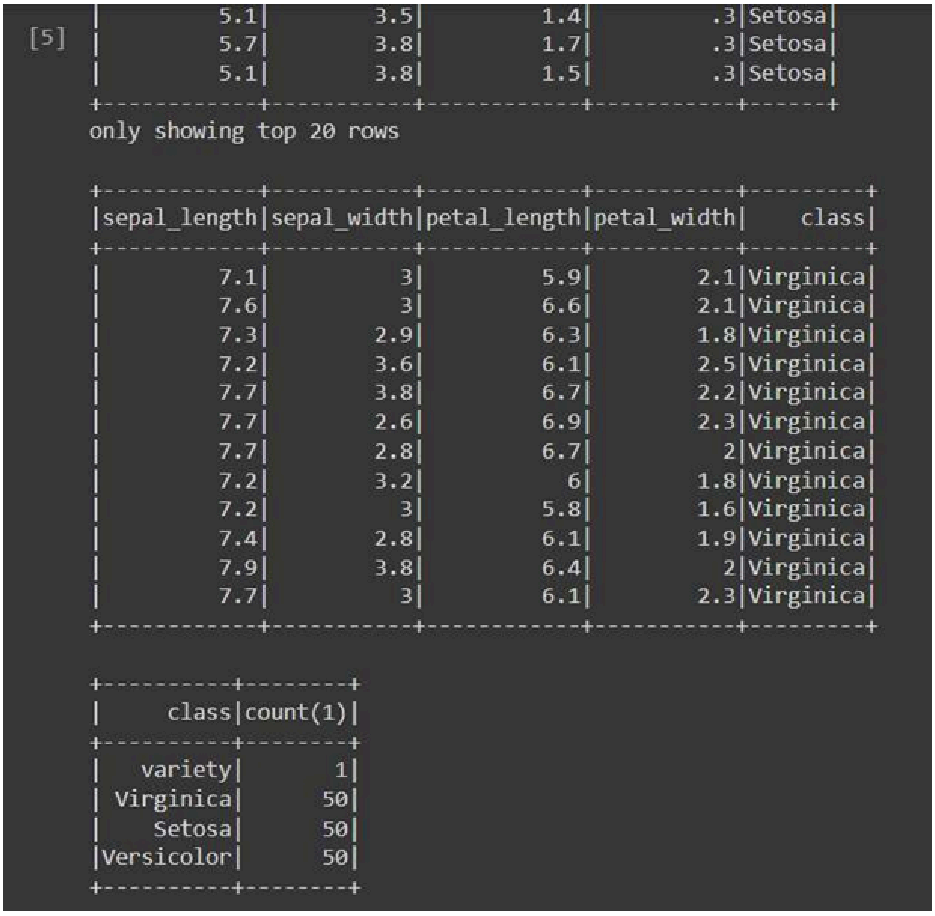
(1) Create spark session
(2) Read csv file as a dataframe using read.csv() method
(3) Register the df as temporary view using the CreateOr ReplaceTempView() method which allows us to query it using SQL
(4) Run SQL query on table
(5) Store result in one variable & display it using show() method
(6) Close spark session

FOR EDUCATIONAL USE

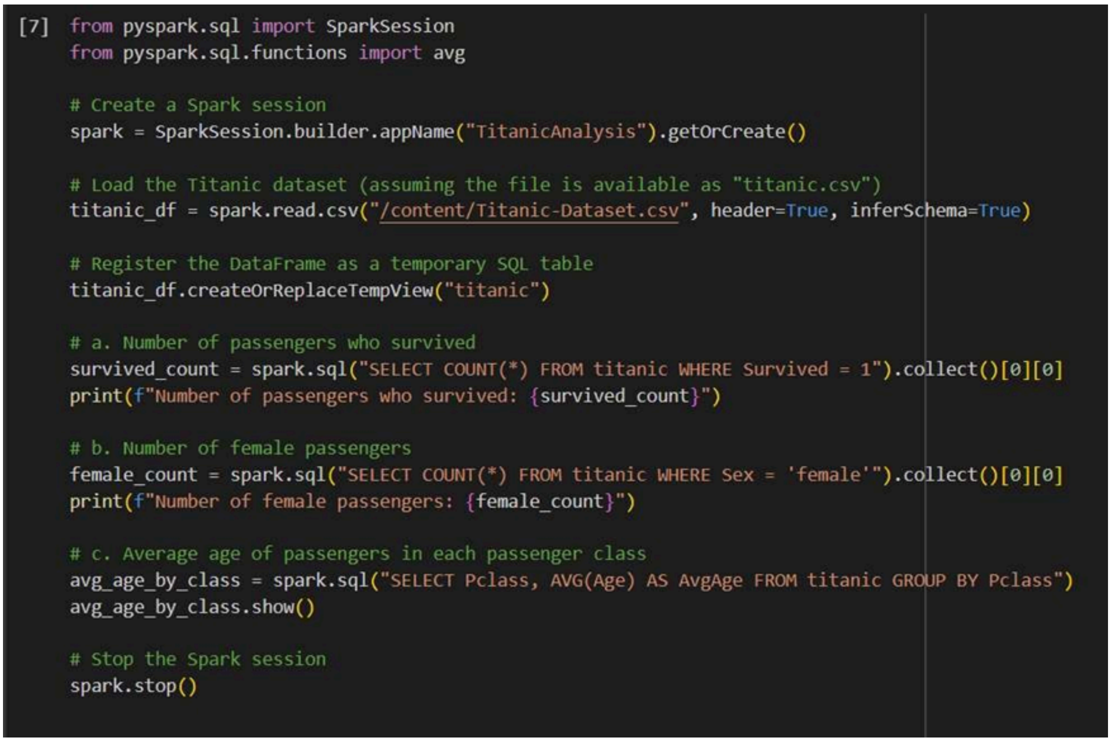


```
[5] from pyspark.sql import SparkSession
    spark = SparkSession.builder.appName("Online Iris Dataset Example").getOrCreate()
    url = "/content/iris.csv"
    df = spark.read.csv(url, header=False, inferSchema=True)
    columns = ["sepal_length", "sepal_width", "petal_length", "petal_width", "class"]
    df = df.toDF(*columns)
    df.createOrReplaceTempView("iris_data")
    result1 = spark.sql("SELECT * FROM iris_data WHERE class = 'Setosa'")
    result2 = spark.sql("SELECT * FROM iris_data WHERE sepal_length >7.0")
    result3 = spark.sql("SELECT class, COUNT(*) FROM iris_data GROUP BY class")
    result1.show()
    result2.show()
    result3.show()
    spark.stop()
```

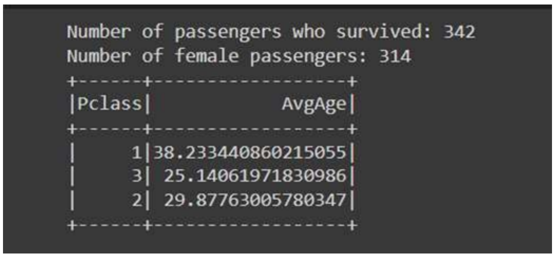
```
+------------+-----------+------------+-----------+------+
|sepal_length|sepal_width|petal_length|petal_width| class|
+------------+-----------+------------+-----------+------+
|         5.1|        3.5|         1.4|         .2|Setosa|
|         4.9|          3|         1.4|         .2|Setosa|
|         4.7|        3.2|         1.3|         .2|Setosa|
|         4.6|        3.1|         1.5|         .2|Setosa|
|           5|        3.6|         1.4|         .2|Setosa|
|         5.4|        3.9|         1.7|         .4|Setosa|
|         4.6|        3.4|         1.4|         .3|Setosa|
|           5|        3.4|         1.5|         .2|Setosa|
|         4.4|        2.9|         1.4|         .2|Setosa|
|         4.9|        3.1|         1.5|         .1|Setosa|
|         5.4|        3.7|         1.5|         .2|Setosa|
```



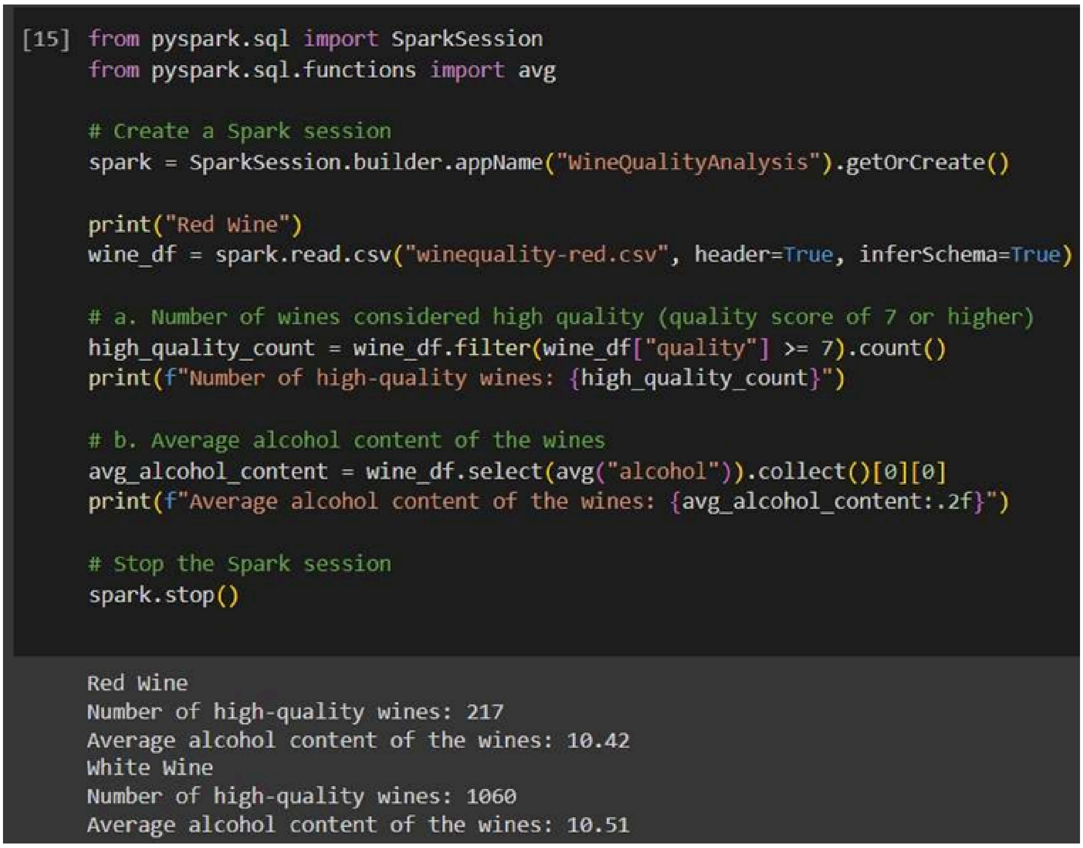
```
[5]
|         5.1|        3.5|         1.4|         .3|Setosa|
|         5.7|        3.8|         1.7|         .3|Setosa|
|         5.1|        3.8|         1.5|         .3|Setosa|
+------------+-----------+------------+-----------+------+
only showing top 20 rows

+------------+-----------+------------+-----------+---------+
|sepal_length|sepal_width|petal_length|petal_width|    class|
+------------+-----------+------------+-----------+---------+
|         7.1|          3|         5.9|        2.1|Virginica|
|         7.6|          3|         6.6|        2.1|Virginica|
|         7.3|        2.9|         6.3|        1.8|Virginica|
|         7.2|        3.6|         6.1|        2.5|Virginica|
|         7.7|        3.8|         6.7|        2.2|Virginica|
|         7.7|        2.6|         6.9|        2.3|Virginica|
|         7.7|        2.8|         6.7|          2|Virginica|
|         7.2|        3.2|           6|        1.8|Virginica|
|         7.2|          3|         5.8|        1.6|Virginica|
|         7.4|        2.8|         6.1|        1.9|Virginica|
|         7.9|        3.8|         6.4|          2|Virginica|
|         7.7|          3|         6.1|        2.3|Virginica|
+------------+-----------+------------+-----------+---------+

+----------+--------+
|     class|count(1)|
+----------+--------+
|   variety|       1|
| Virginica|      50|
|    Setosa|      50|
|Versicolor|      50|
+----------+--------+
```



```
[7] from pyspark.sql import SparkSession
    from pyspark.sql.functions import avg

    # Create a Spark session
    spark = SparkSession.builder.appName("TitanicAnalysis").getOrCreate()

    # Load the Titanic dataset (assuming the file is available as "titanic.csv")
    titanic_df = spark.read.csv("/content/Titanic-Dataset.csv", header=True, inferSchema=True)

    # Register the DataFrame as a temporary SQL table
    titanic_df.createOrReplaceTempView("titanic")

    # a. Number of passengers who survived
    survived_count = spark.sql("SELECT COUNT(*) FROM titanic WHERE Survived = 1").collect()[0][0]
    print(f"Number of passengers who survived: {survived_count}")

    # b. Number of female passengers
    female_count = spark.sql("SELECT COUNT(*) FROM titanic WHERE Sex = 'female'").collect()[0][0]
    print(f"Number of female passengers: {female_count}")

    # c. Average age of passengers in each passenger class
    avg_age_by_class = spark.sql("SELECT Pclass, AVG(Age) AS AvgAge FROM titanic GROUP BY Pclass")
    avg_age_by_class.show()

    # Stop the Spark session
    spark.stop()
```

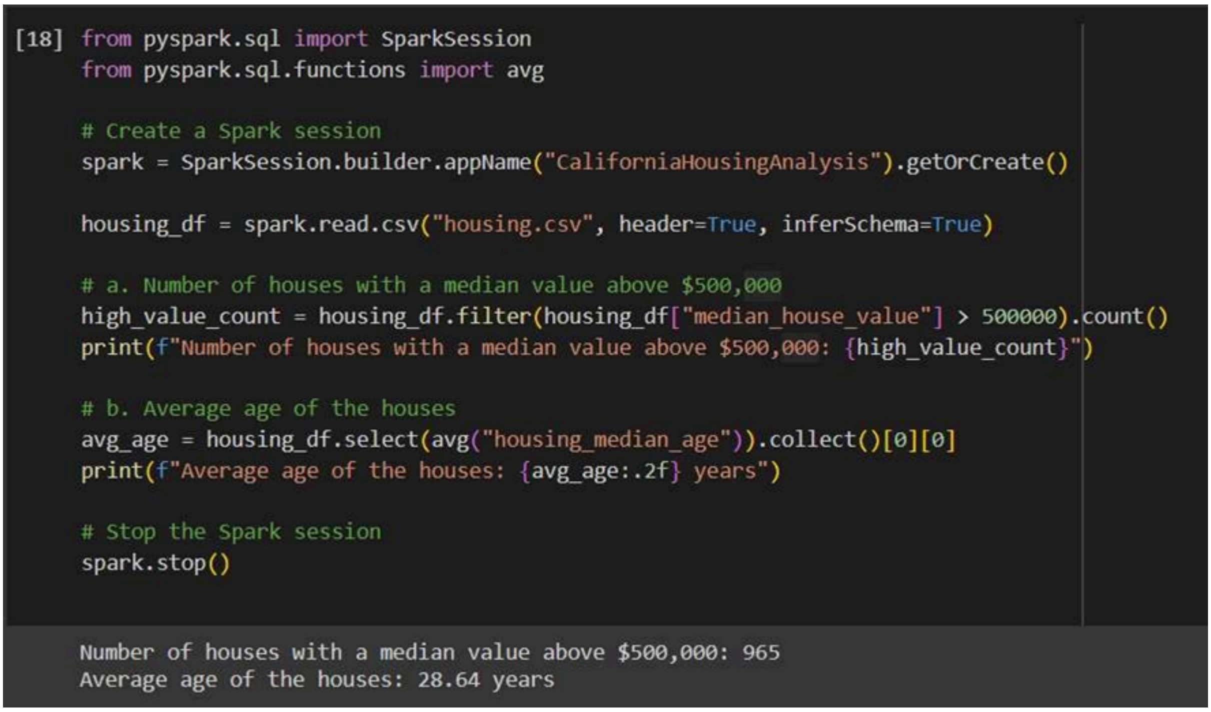


```
Number of passengers who survived: 342
Number of female passengers: 314
+------+-----------------+
|Pclass|           AvgAge|
+------+-----------------+
|     1|38.233440860215055|
|     3| 25.14061971830986|
|     2| 29.87763005780347|
+------+-----------------+
```



```
[15] from pyspark.sql import SparkSession
     from pyspark.sql.functions import avg

     # Create a Spark session
     spark = SparkSession.builder.appName("WineQualityAnalysis").getOrCreate()

     print("Red Wine")
     wine_df = spark.read.csv("winequality-red.csv", header=True, inferSchema=True)

     # a. Number of wines considered high quality (quality score of 7 or higher)
     high_quality_count = wine_df.filter(wine_df["quality"] >= 7).count()
     print(f"Number of high-quality wines: {high_quality_count}")

     # b. Average alcohol content of the wines
     avg_alcohol_content = wine_df.select(avg("alcohol")).collect()[0][0]
     print(f"Average alcohol content of the wines: {avg_alcohol_content:.2f}")

     # Stop the Spark session
     spark.stop()
```

```
Red Wine
Number of high-quality wines: 217
Average alcohol content of the wines: 10.42
White Wine
Number of high-quality wines: 1060
Average alcohol content of the wines: 10.51
```



```
[18] from pyspark.sql import SparkSession
     from pyspark.sql.functions import avg

     # Create a Spark session
     spark = SparkSession.builder.appName("CaliforniaHousingAnalysis").getOrCreate()

     housing_df = spark.read.csv("housing.csv", header=True, inferSchema=True)

     # a. Number of houses with a median value above $500,000
     high_value_count = housing_df.filter(housing_df["median_house_value"] > 500000).count()
     print(f"Number of houses with a median value above $500,000: {high_value_count}")

     # b. Average age of the houses
     avg_age = housing_df.select(avg("housing_median_age")).collect()[0][0]
     print(f"Average age of the houses: {avg_age:.2f} years")

     # Stop the Spark session
     spark.stop()
```

```
Number of houses with a median value above $500,000: 965
Average age of the houses: 28.64 years
```

Conclusion : Thus, we understood & implemented SparkQL using PySpark.