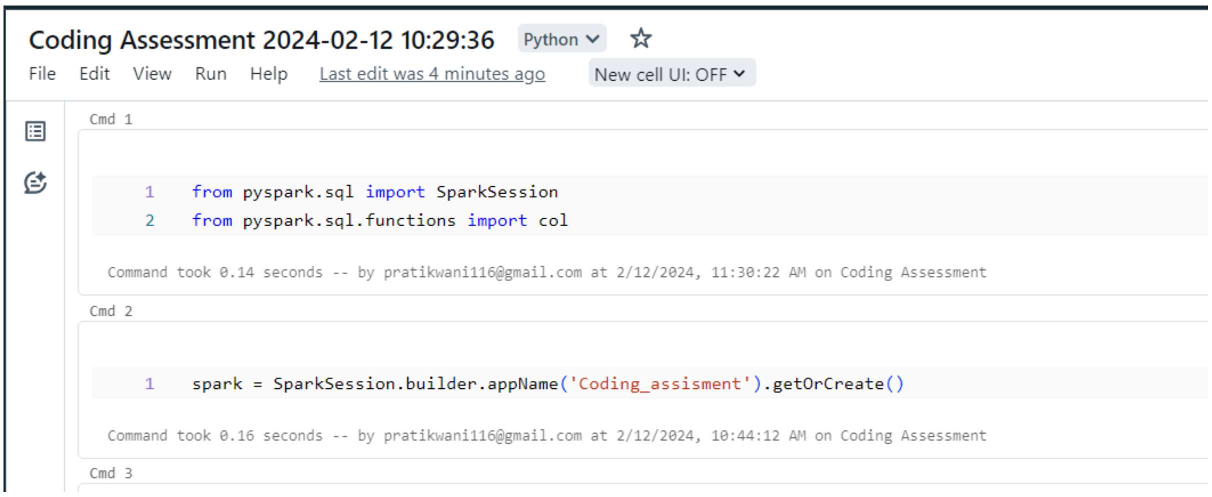
PySpark Coding Challenge

Pratik Wani

Question No 1

- Created two dataframes with the named as df1 and df2
- All the operations (Manipulating, Droping, Sorting, Aggregations, Joining, GroupeBy) performed on these dataframes

- Creating Session:
    - After importing necessary Libraries Spark Session is created using getOrCreate()


Coding Assessment 2024-02-12 10:29:36
Python
File
Edit
View
Run
Help
Last edit was 4 minutes ago
New cell UI: OFF
Cmd 1
Cmd 2
Cmd 3

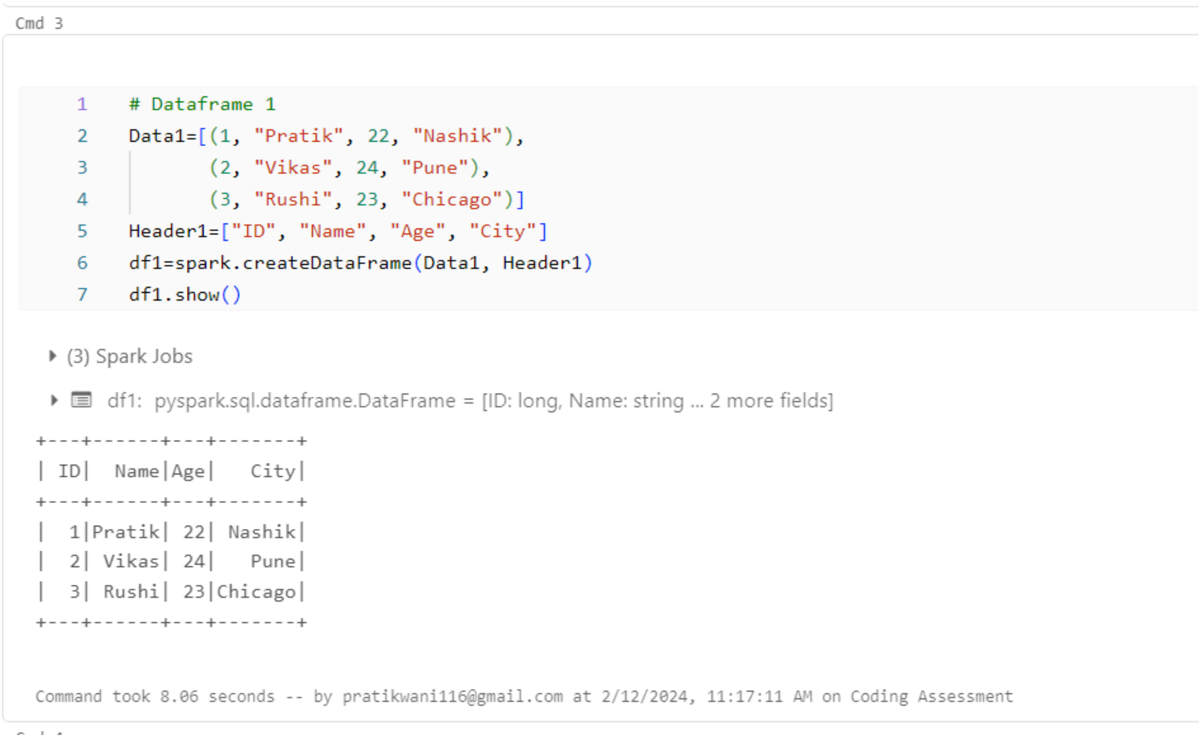
```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col
```

```
Command took 0.14 seconds -- by pratikwani116@gmail.com at 2/12/2024, 11:30:22 AM on Coding Assessment
```

```
spark = SparkSession.builder.appName('Coding_assisment').getOrCreate()
```

```
Command took 0.16 seconds -- by pratikwani116@gmail.com at 2/12/2024, 10:44:12 AM on Coding Assessment
```

- Creating DataFrames:
    - Two DataFrames are created as df1 and df2 using the method createDataFrame by passing the data and the header



Cmd 3

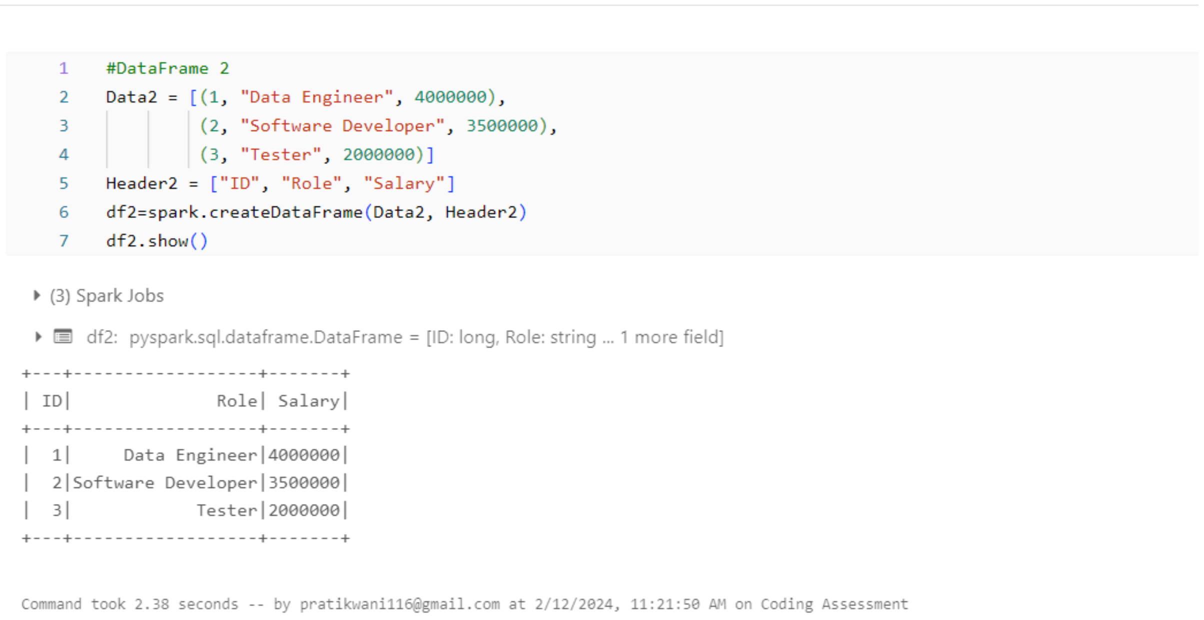
```
# Dataframe 1
Data1=[(1, "Pratik", 22, "Nashik"),
       (2, "Vikas", 24, "Pune"),
       (3, "Rushi", 23, "Chicago")]
Header1=["ID", "Name", "Age", "City"]
df1=spark.createDataFrame(Data1, Header1)
df1.show()
```

▸ (3) Spark Jobs

▸ df1: pyspark.sql.dataframe.DataFrame = [ID: long, Name: string ... 2 more fields]

```
+---+------+---+-------+
| ID|  Name|Age|   City|
+---+------+---+-------+
|  1|Pratik| 22| Nashik|
|  2| Vikas| 24|   Pune|
|  3| Rushi| 23|Chicago|
+---+------+---+-------+
```

Command took 8.06 seconds -- by pratikwani116@gmail.com at 2/12/2024, 11:17:11 AM on Coding Assessment



```
#DataFrame 2
Data2 = [(1, "Data Engineer", 4000000),
         (2, "Software Developer", 3500000),
         (3, "Tester", 2000000)]
Header2 = ["ID", "Role", "Salary"]
df2=spark.createDataFrame(Data2, Header2)
df2.show()
```

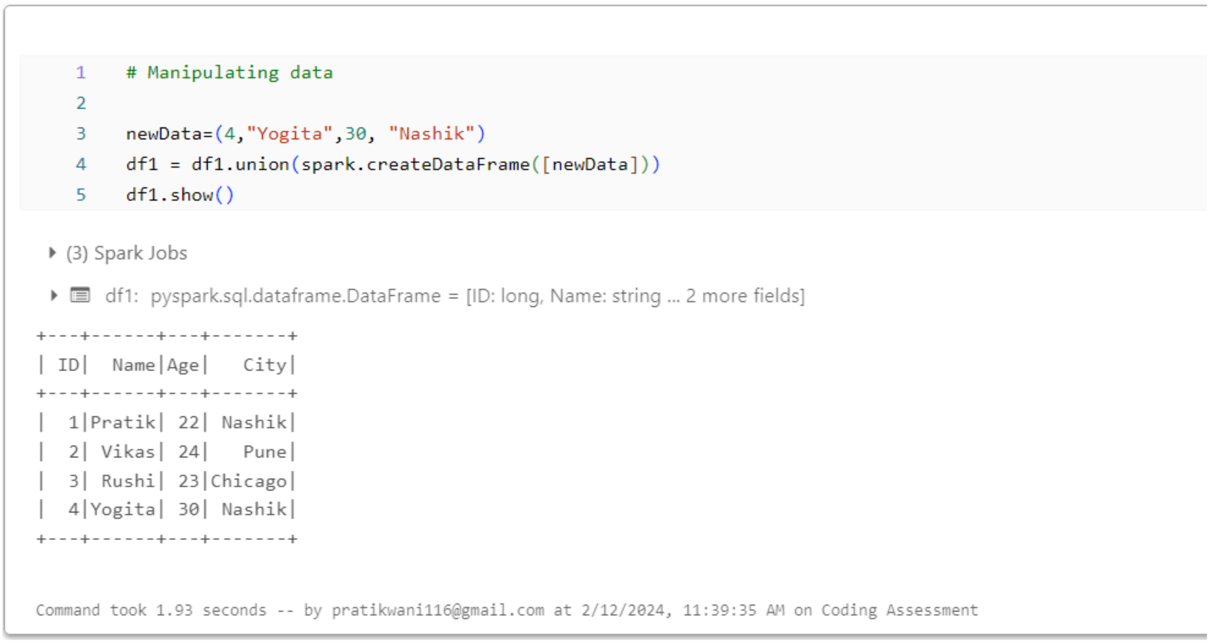
▸ (3) Spark Jobs

▸ df2: pyspark.sql.dataframe.DataFrame = [ID: long, Role: string ... 1 more field]

```
+---+------------------+-------+
| ID|              Role| Salary|
+---+------------------+-------+
|  1|     Data Engineer|4000000|
|  2|Software Developer|3500000|
|  3|            Tester|2000000|
+---+------------------+-------+
```

Command took 2.38 seconds -- by pratikwani116@gmail.com at 2/12/2024, 11:21:50 AM on Coding Assessment

- Manupulating Data:
    - After creating the df1 added the new data with ID 4 and Name Yogita in df1
    - To add the data I created new dataframe with newdata and then combined it with df1 using union method



```
# Manipulating data

newData=(4,"Yogita",30, "Nashik")
df1 = df1.union(spark.createDataFrame([newData]))
df1.show()
```

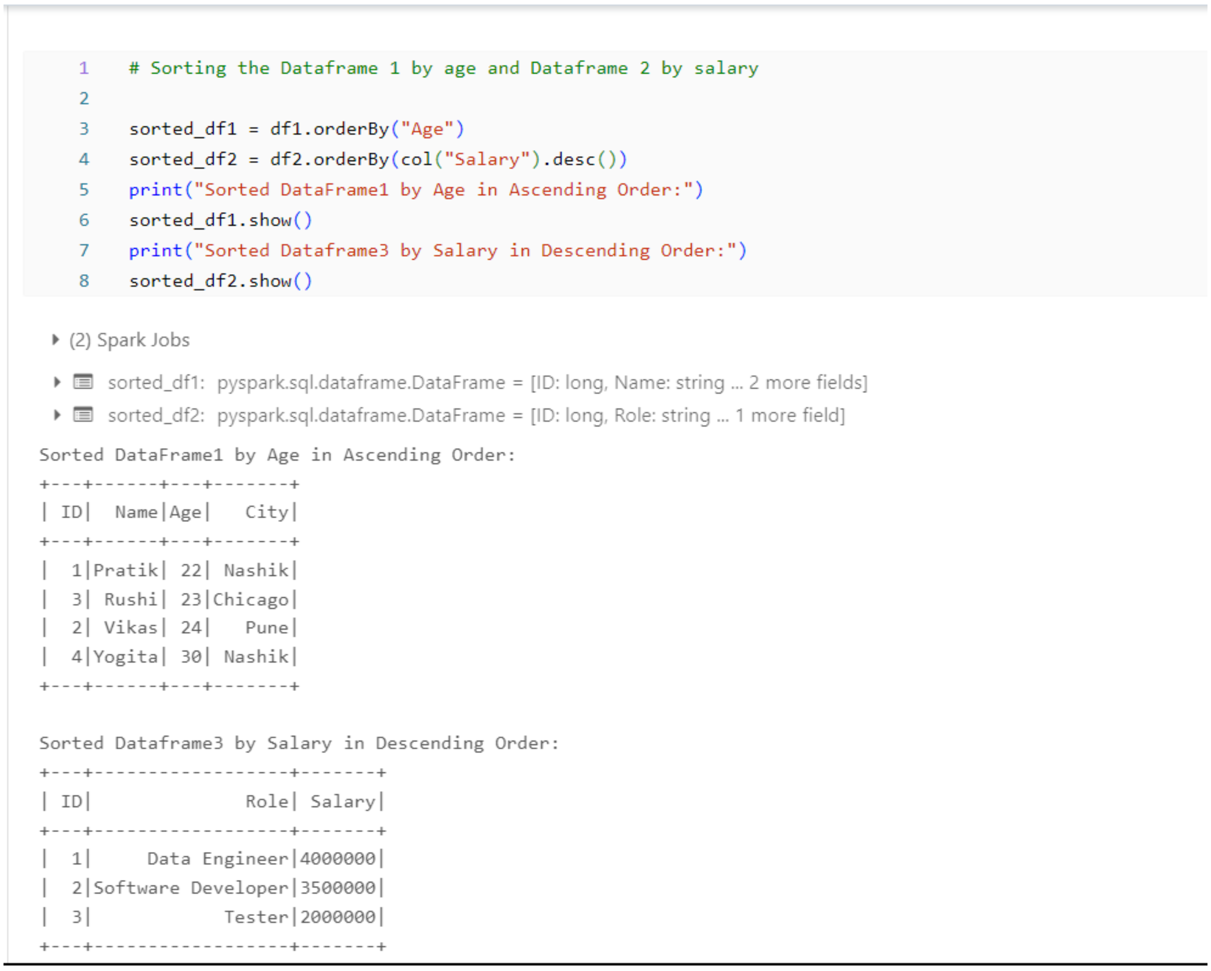
(3) Spark Jobs

df1: pyspark.sql.dataframe.DataFrame = [ID: long, Name: string ... 2 more fields]

```
+---+------+---+-------+
| ID|  Name|Age|   City|
+---+------+---+-------+
|  1|Pratik| 22| Nashik|
|  2| Vikas| 24|   Pune|
|  3| Rushi| 23|Chicago|
|  4|Yogita| 30| Nashik|
+---+------+---+-------+
```

Command took 1.93 seconds -- by pratikwani116@gmail.com at 2/12/2024, 11:39:35 AM on Coding Assessment

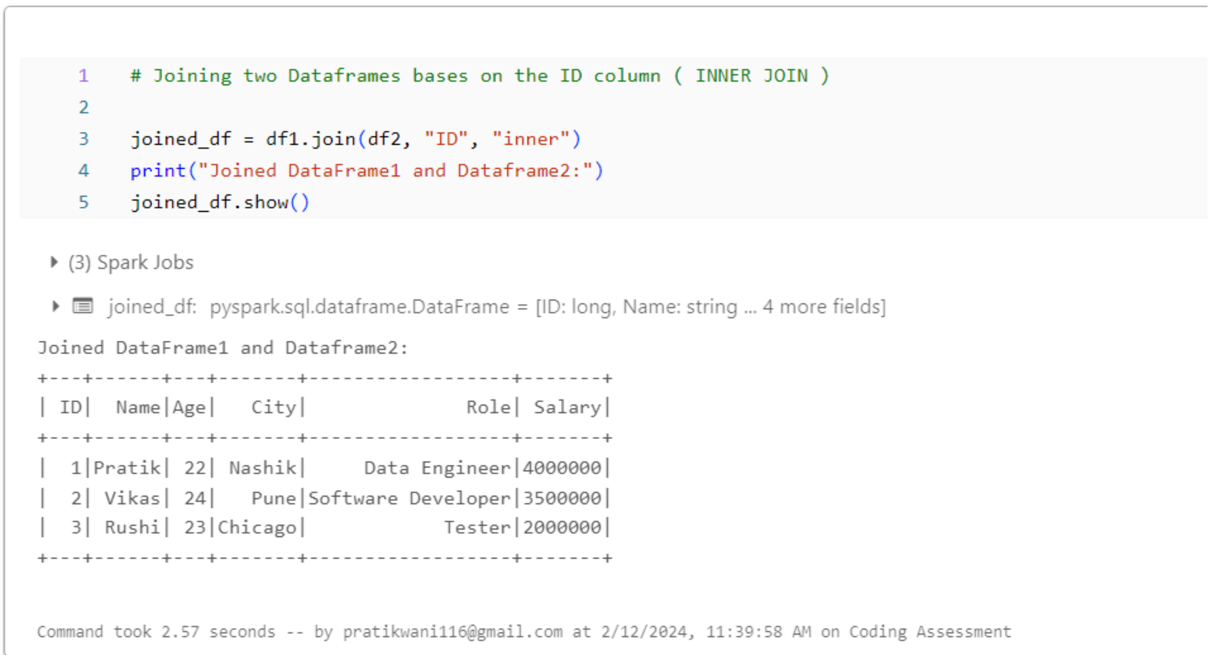
- Sorting:
    - The df1 is sorted by Age in Ascending Order while the df2 is sorted by Salary in Descending Order
    - To sort the data I used orderBy() method



```python
# Sorting the Dataframe 1 by age and Dataframe 2 by salary

sorted_df1 = df1.orderBy("Age")
sorted_df2 = df2.orderBy(col("Salary").desc())
print("Sorted DataFrame1 by Age in Ascending Order:")
sorted_df1.show()
print("Sorted Dataframe3 by Salary in Descending Order:")
sorted_df2.show()
```

```
▸ (2) Spark Jobs
▸ sorted_df1:  pyspark.sql.dataframe.DataFrame = [ID: long, Name: string ... 2 more fields]
▸ sorted_df2:  pyspark.sql.dataframe.DataFrame = [ID: long, Role: string ... 1 more field]
Sorted DataFrame1 by Age in Ascending Order:
+---+------+---+-------+
| ID|  Name|Age|   City|
+---+------+---+-------+
|  1|Pratik| 22| Nashik|
|  3| Rushi| 23|Chicago|
|  2| Vikas| 24|   Pune|
|  4|Yogita| 30| Nashik|
+---+------+---+-------+

Sorted Dataframe3 by Salary in Descending Order:
+---+------------------+-------+
| ID|              Role| Salary|
+---+------------------+-------+
|  1|     Data Engineer|4000000|
|  2|Software Developer|3500000|
|  3|            Tester|2000000|
+---+------------------+-------+
```

- Joining:
    - The df1 and df2 both having one column in common which is ID based on column ID both DataFrames are joined
    - To join the DataFrames we have to use the join() method and pass the name of second DataFrame with the Column name and type of join
    - In this example Inner join is performed



```
# Joining two Dataframes bases on the ID column ( INNER JOIN )

joined_df = df1.join(df2, "ID", "inner")
print("Joined DataFrame1 and Dataframe2:")
joined_df.show()
```

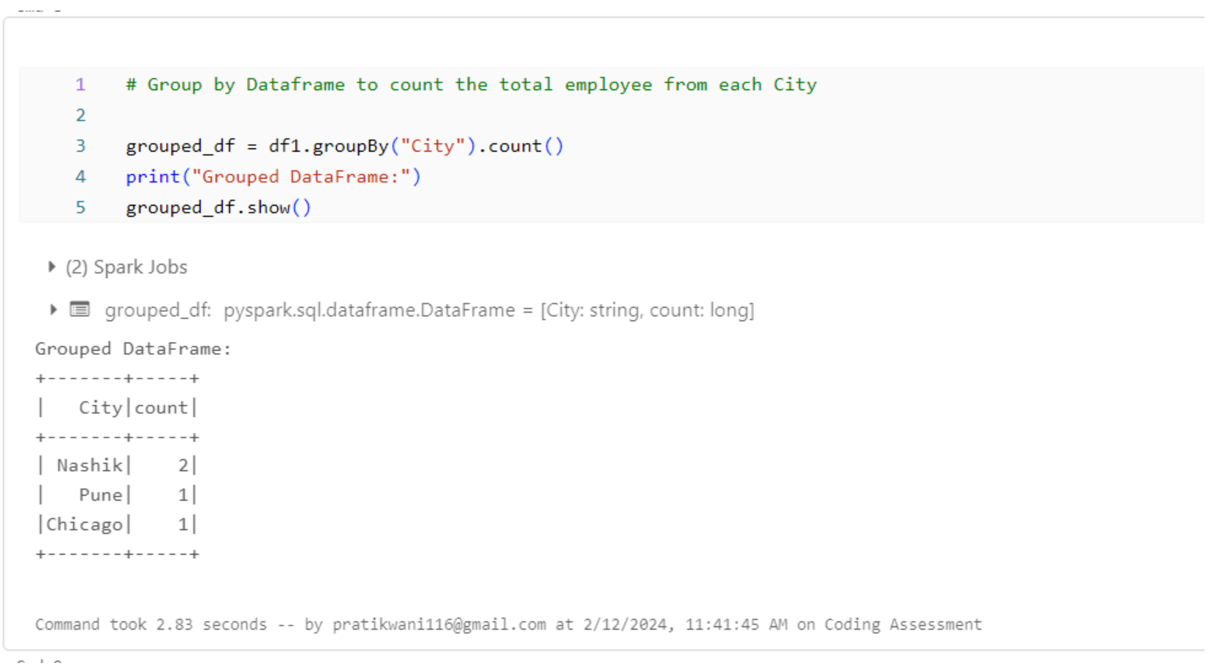
▸ (3) Spark Jobs

▸ joined_df: pyspark.sql.dataframe.DataFrame = [ID: long, Name: string ... 4 more fields]

```
Joined DataFrame1 and Dataframe2:
+---+------+---+-------+------------------+-------+
| ID|  Name|Age|   City|              Role| Salary|
+---+------+---+-------+------------------+-------+
|  1|Pratik| 22| Nashik|     Data Engineer|4000000|
|  2| Vikas| 24|   Pune|Software Developer|3500000|
|  3| Rushi| 23|Chicago|            Tester|2000000|
+---+------+---+-------+------------------+-------+
```

Command took 2.57 seconds -- by pratikwani116@gmail.com at 2/12/2024, 11:39:58 AM on Coding Assessment

- Group By:
    - To group the data bases on some field we have to use groupBy() method by passing the column name
    - We can perform different aggregate functions on such grouped data
    - In this example DataFrame df1 is groupby citywise and citywise total number of employees are counted.



```
# Group by Dataframe to count the total employee from each City

grouped_df = df1.groupBy("City").count()
print("Grouped DataFrame:")
grouped_df.show()
```

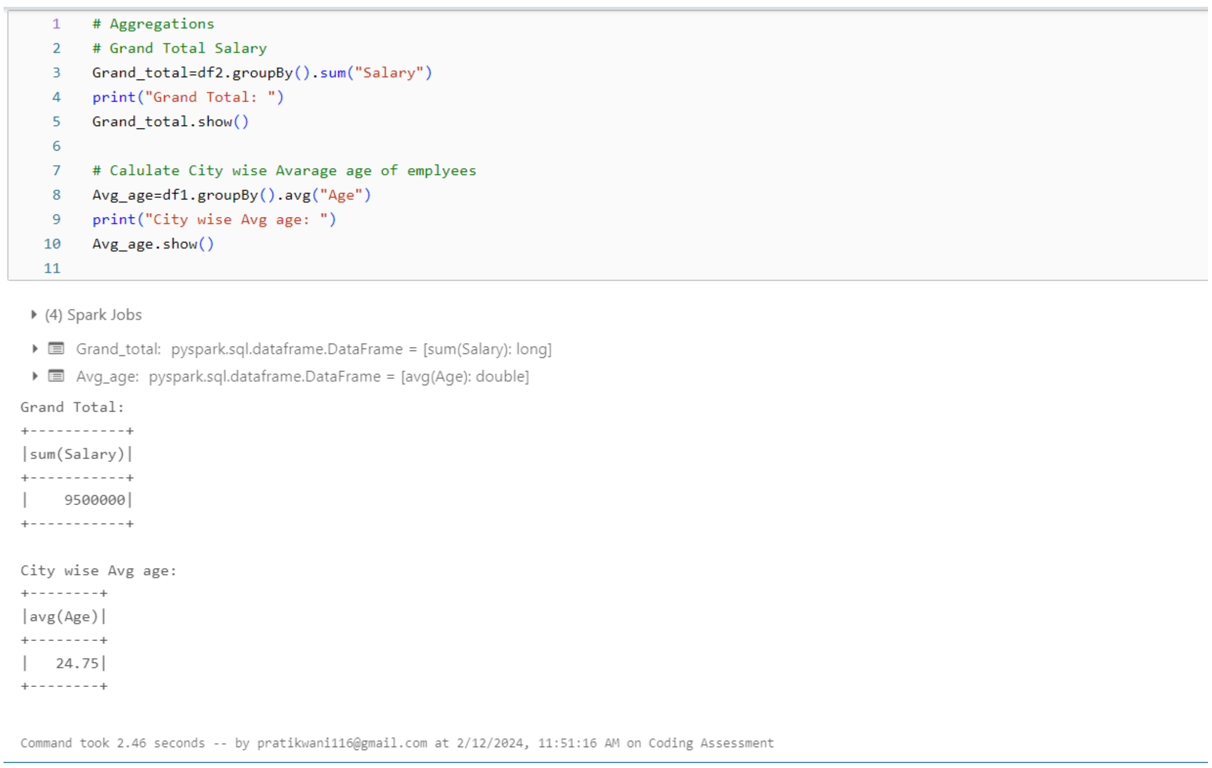
(2) Spark Jobs

grouped_df: pyspark.sql.dataframe.DataFrame = [City: string, count: long]

```
Grouped DataFrame:
+-------+-----+
|   City|count|
+-------+-----+
| Nashik|    2|
|   Pune|    1|
|Chicago|    1|
+-------+-----+
```

Command took 2.83 seconds -- by pratikwani116@gmail.com at 2/12/2024, 11:41:45 AM on Coding Assessment

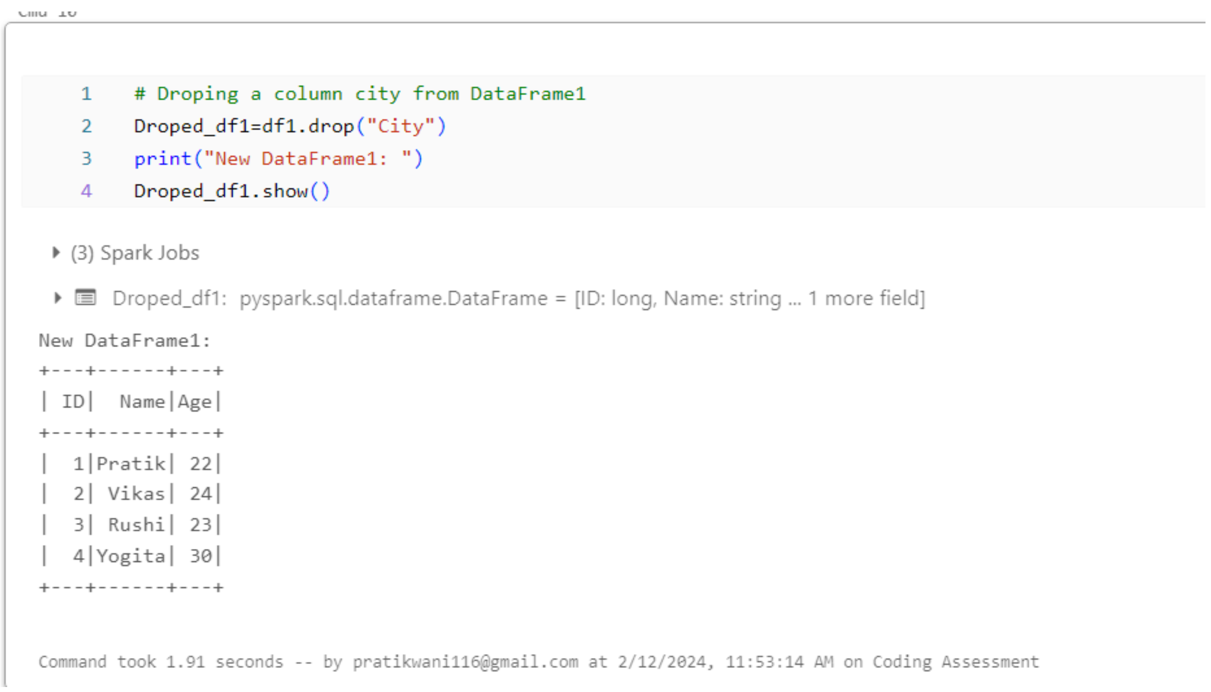
- Aggregation:
    - After grouping the data we can perform different aggregations on the data
    - Such as Sum(), Max(), Min(), Avg()
    - Here 2 examples are performed one is finding Grand Total and another one is finding Citywise Average age of employees



```
# Aggregations
# Grand Total Salary
Grand_total=df2.groupBy().sum("Salary")
print("Grand Total: ")
Grand_total.show()

# Calulate City wise Avarage age of emplyees
Avg_age=df1.groupBy().avg("Age")
print("City wise Avg age: ")
Avg_age.show()

```

```
▸ (4) Spark Jobs
▸ Grand_total:  pyspark.sql.dataframe.DataFrame = [sum(Salary): long]
▸ Avg_age:  pyspark.sql.dataframe.DataFrame = [avg(Age): double]
Grand Total:
+-----------+
|sum(Salary)|
+-----------+
|    9500000|
+-----------+

City wise Avg age:
+--------+
|avg(Age)|
+--------+
|   24.75|
+--------+

Command took 2.46 seconds -- by pratikwani116@gmail.com at 2/12/2024, 11:51:16 AM on Coding Assessment
```

- Droping:
    - To drop any column from the DataFrame we have to use drop method by passing the name of the columns we want to delete.
    - In this example column name City is deleted from the DataFrame df1



```
# Droping a column city from DataFrame1
Droped_df1=df1.drop("City")
print("New DataFrame1: ")
Droped_df1.show()
```

▸ (3) Spark Jobs

▸ Droped_df1: pyspark.sql.dataframe.DataFrame = [ID: long, Name: string ... 1 more field]

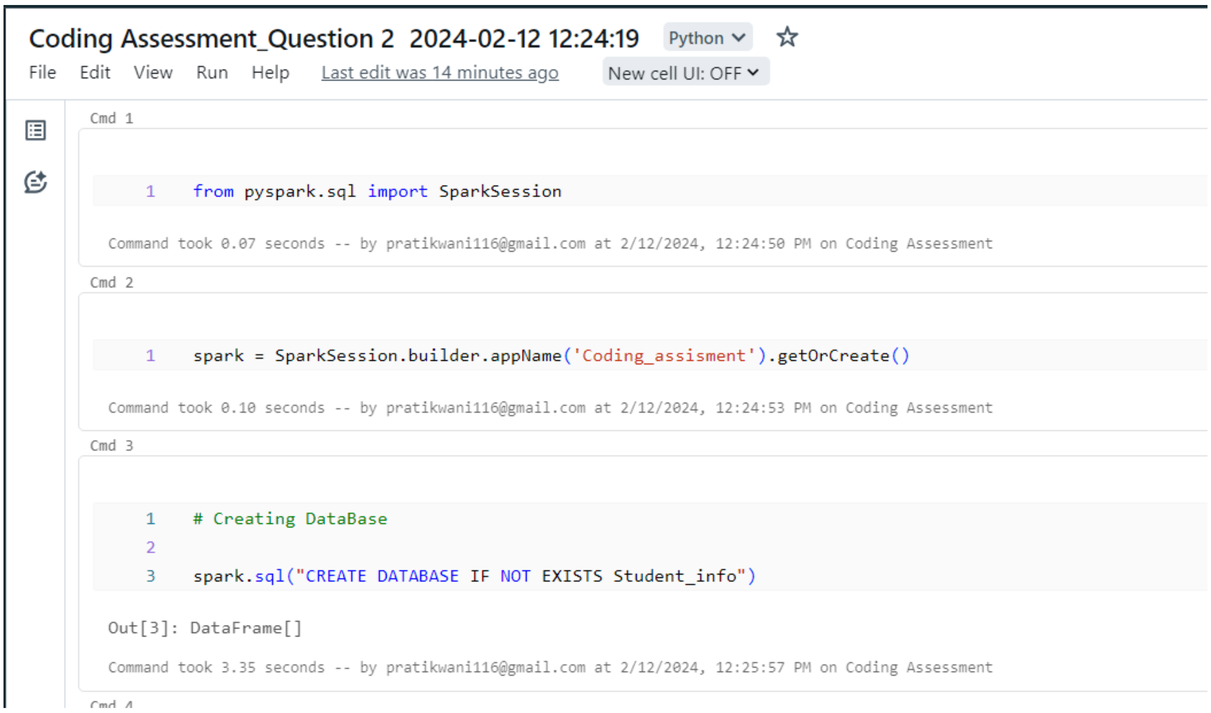
```
New DataFrame1:
+---+------+---+
| ID|  Name|Age|
+---+------+---+
|  1|Pratik| 22|
|  2| Vikas| 24|
|  3| Rushi| 23|
|  4|Yogita| 30|
+---+------+---+
```

Command took 1.91 seconds -- by pratikwani116@gmail.com at 2/12/2024, 11:53:14 AM on Coding Assessment

Question No 2

- The Database Student_info is created in that database two tables named as Students and Courses are created
- Join operations performed on these tables

- Create Database:


Coding Assessment_Question 2 2024-02-12 12:24:19
Python
File Edit View Run Help
Last edit was 14 minutes ago
New cell UI: OFF

Cmd 1

```
from pyspark.sql import SparkSession
```

```
Command took 0.07 seconds -- by pratikwani116@gmail.com at 2/12/2024, 12:24:50 PM on Coding Assessment
```

Cmd 2

```
spark = SparkSession.builder.appName('Coding_assisment').getOrCreate()
```

```
Command took 0.10 seconds -- by pratikwani116@gmail.com at 2/12/2024, 12:24:53 PM on Coding Assessment
```

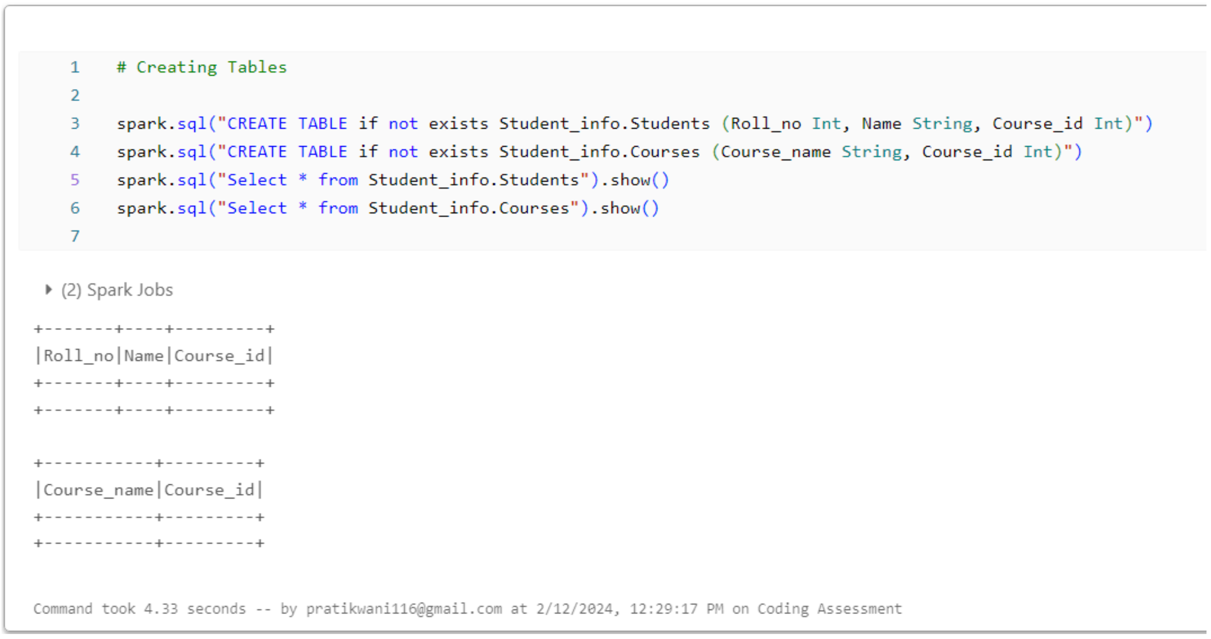
Cmd 3

```
# Creating DataBase

spark.sql("CREATE DATABASE IF NOT EXISTS Student_info")
```

```
Out[3]: DataFrame[]
Command took 3.35 seconds -- by pratikwani116@gmail.com at 2/12/2024, 12:25:57 PM on Coding Assessment
```

Cmd 4

- Create Tables



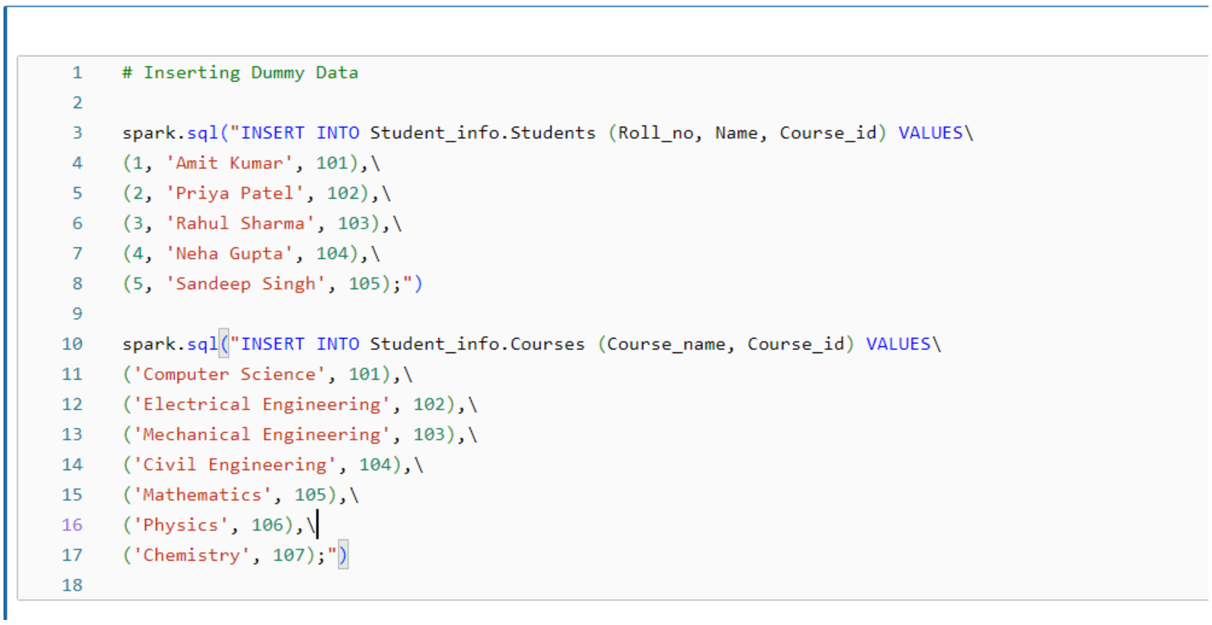
```
# Creating Tables

spark.sql("CREATE TABLE if not exists Student_info.Students (Roll_no Int, Name String, Course_id Int)")
spark.sql("CREATE TABLE if not exists Student_info.Courses (Course_name String, Course_id Int)")
spark.sql("Select * from Student_info.Students").show()
spark.sql("Select * from Student_info.Courses").show()
```

(2) Spark Jobs

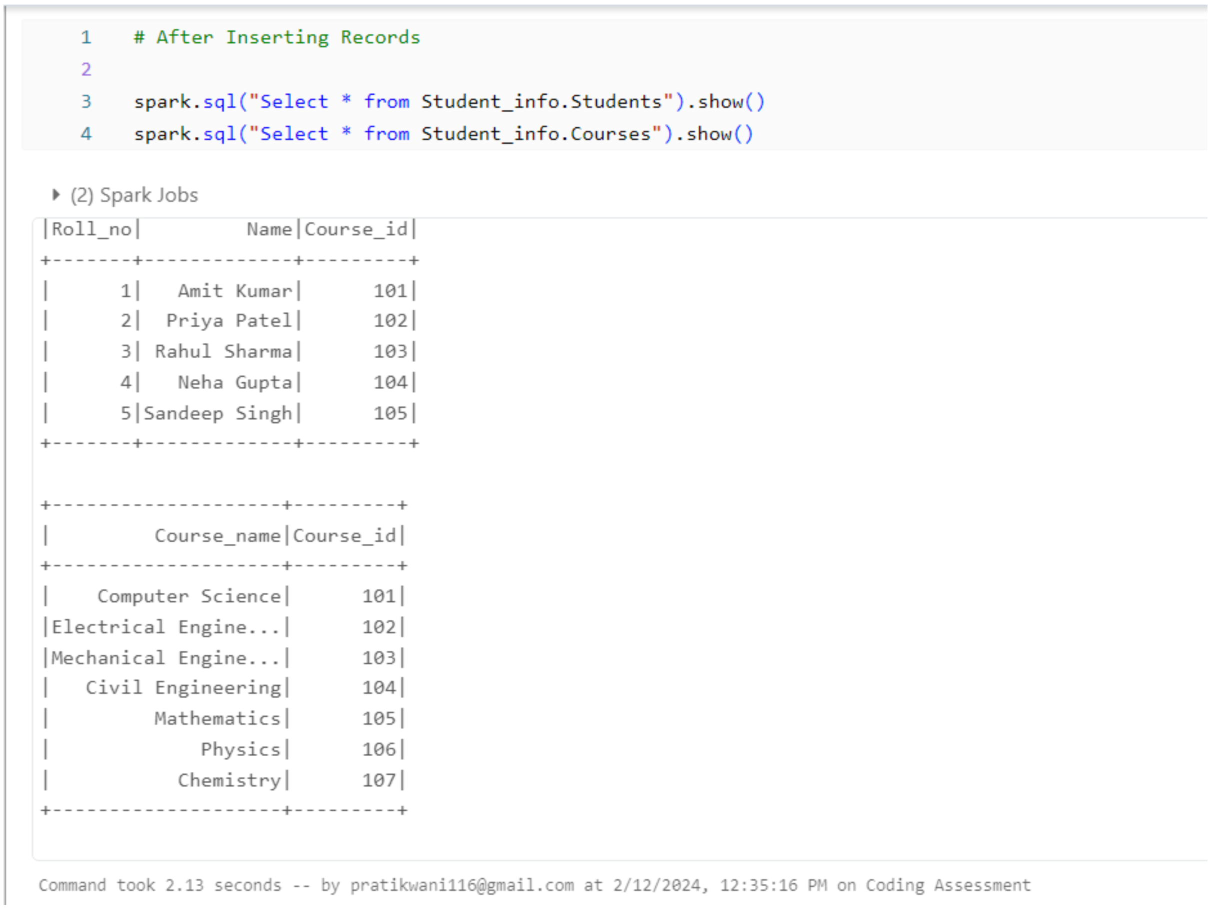
```
+-------+----+---------+
|Roll_no|Name|Course_id|
+-------+----+---------+
+-------+----+---------+

+-----------+---------+
|Course_name|Course_id|
+-----------+---------+
+-----------+---------+
```

Command took 4.33 seconds -- by pratikwani116@gmail.com at 2/12/2024, 12:29:17 PM on Coding Assessment

- Insert Records



```
# Inserting Dummy Data

spark.sql("INSERT INTO Student_info.Students (Roll_no, Name, Course_id) VALUES\
(1, 'Amit Kumar', 101),\
(2, 'Priya Patel', 102),\
(3, 'Rahul Sharma', 103),\
(4, 'Neha Gupta', 104),\
(5, 'Sandeep Singh', 105);")

spark.sql("INSERT INTO Student_info.Courses (Course_name, Course_id) VALUES\
('Computer Science', 101),\
('Electrical Engineering', 102),\
('Mechanical Engineering', 103),\
('Civil Engineering', 104),\
('Mathematics', 105),\
('Physics', 106),\
('Chemistry', 107);")
```



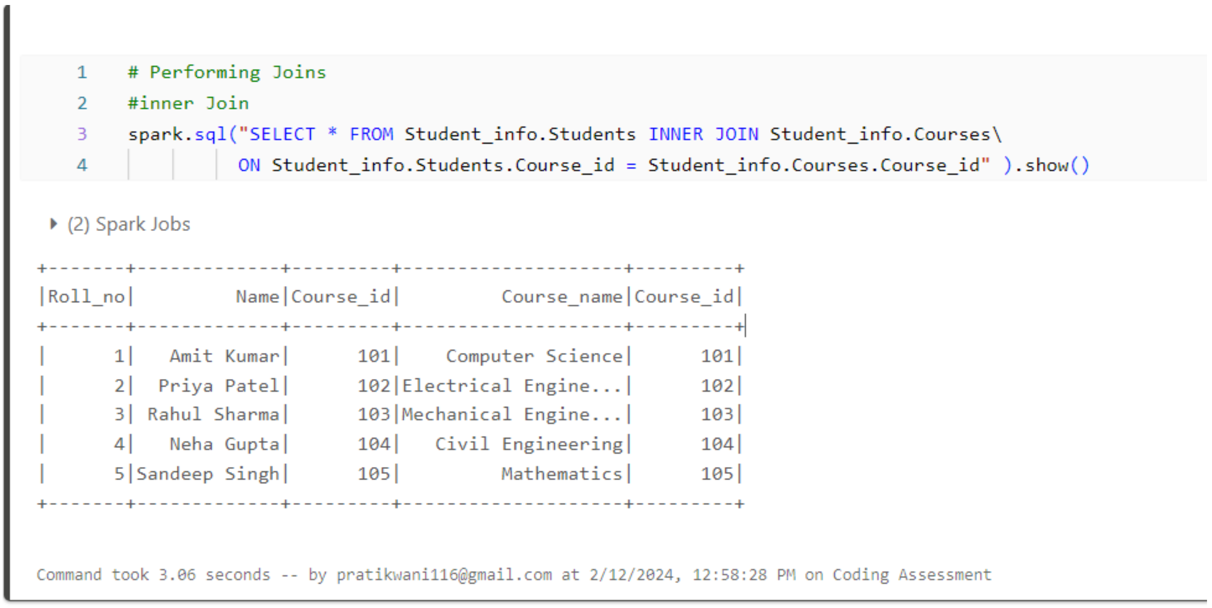
```python
# After Inserting Records

spark.sql("Select * from Student_info.Students").show()
spark.sql("Select * from Student_info.Courses").show()
```

(2) Spark Jobs

```
|Roll_no|        Name|Course_id|
+-------+------------+---------+
|      1|  Amit Kumar|      101|
|      2| Priya Patel|      102|
|      3|Rahul Sharma|      103|
|      4|  Neha Gupta|      104|
|      5|Sandeep Singh|      105|
+-------+------------+---------+

+--------------------+---------+
|         Course_name|Course_id|
+--------------------+---------+
|    Computer Science|      101|
|Electrical Engine...|      102|
|Mechanical Engine...|      103|
|   Civil Engineering|      104|
|         Mathematics|      105|
|             Physics|      106|
|           Chemistry|      107|
+--------------------+---------+
```

Command took 2.13 seconds -- by pratikwani116@gmail.com at 2/12/2024, 12:35:16 PM on Coding Assessment

- Inner Join:
  - Inner Join takes all the matching rows from the both tables
  - student is enrolled in courses with course_id 106, 107 so in the result set of inner join these two coursed will not include



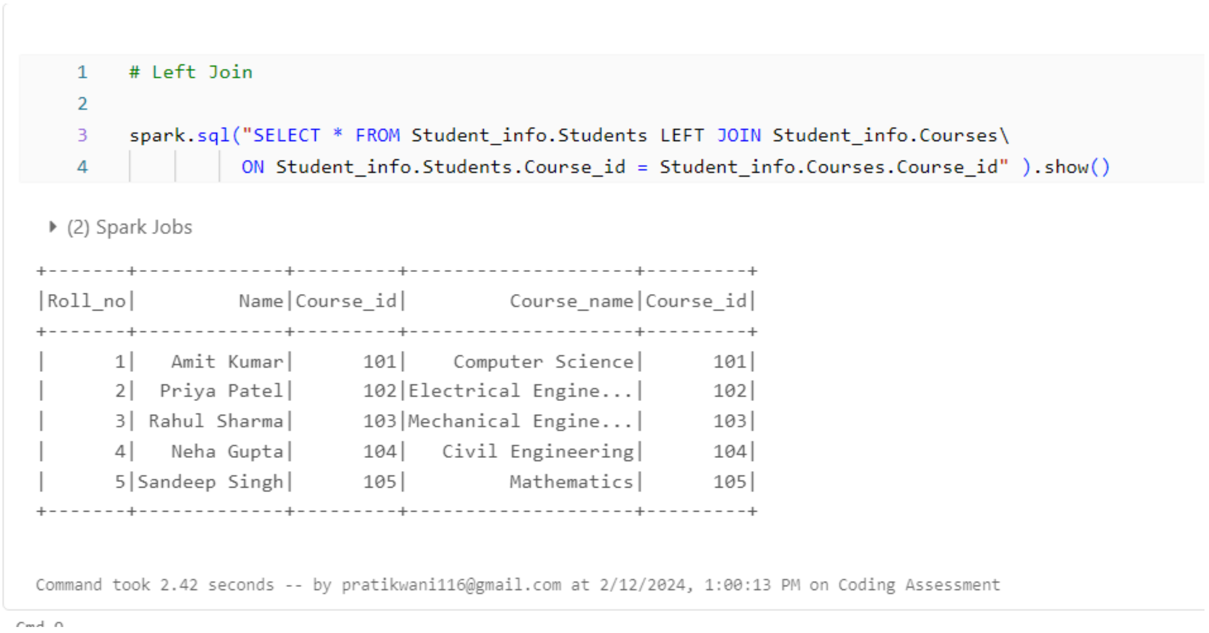
```
# Performing Joins
#inner Join
spark.sql("SELECT * FROM Student_info.Students INNER JOIN Student_info.Courses\
            ON Student_info.Students.Course_id = Student_info.Courses.Course_id" ).show()
```

▸ (2) Spark Jobs

```
+-------+-------------+---------+--------------------+---------+
|Roll_no|         Name|Course_id|         Course_name|Course_id|
+-------+-------------+---------+--------------------+---------+
|      1|   Amit Kumar|      101|    Computer Science|      101|
|      2|  Priya Patel|      102|Electrical Engine...|      102|
|      3| Rahul Sharma|      103|Mechanical Engine...|      103|
|      4|   Neha Gupta|      104|   Civil Engineering|      104|
|      5|Sandeep Singh|      105|         Mathematics|      105|
+-------+-------------+---------+--------------------+---------+
```

Command took 3.06 seconds -- by pratikwani116@gmail.com at 2/12/2024, 12:58:28 PM on Coding Assessment

- Left Join:
    - Left Join takes all the rows from Left Table and Matching Rows from the Right Table
    - The Unmatched fields are filled with Null values



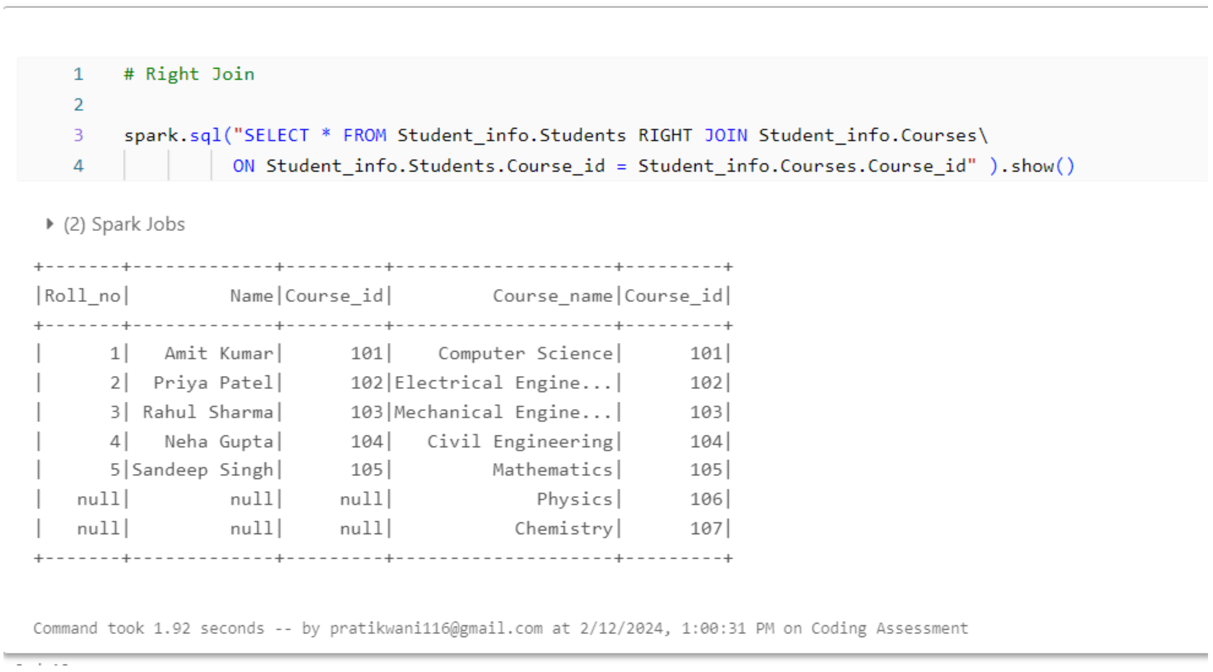
```
# Left Join

spark.sql("SELECT * FROM Student_info.Students LEFT JOIN Student_info.Courses\
        ON Student_info.Students.Course_id = Student_info.Courses.Course_id" ).show()
```

▸ (2) Spark Jobs

```
+-------+------------+---------+-------------------+---------+
|Roll_no|        Name|Course_id|        Course_name|Course_id|
+-------+------------+---------+-------------------+---------+
|      1|  Amit Kumar|      101|   Computer Science|      101|
|      2| Priya Patel|      102|Electrical Engine...|      102|
|      3|Rahul Sharma|      103|Mechanical Engine...|      103|
|      4|  Neha Gupta|      104|  Civil Engineering|      104|
|      5|Sandeep Singh|      105|        Mathematics|      105|
+-------+------------+---------+-------------------+---------+
```

Command took 2.42 seconds -- by pratikwani116@gmail.com at 2/12/2024, 1:00:13 PM on Coding Assessment

- Right Join:
  - Right Join takes all the rows from Right Table and Matching Rows from the Left Table
  - The Unmatched fields are filled with Null values



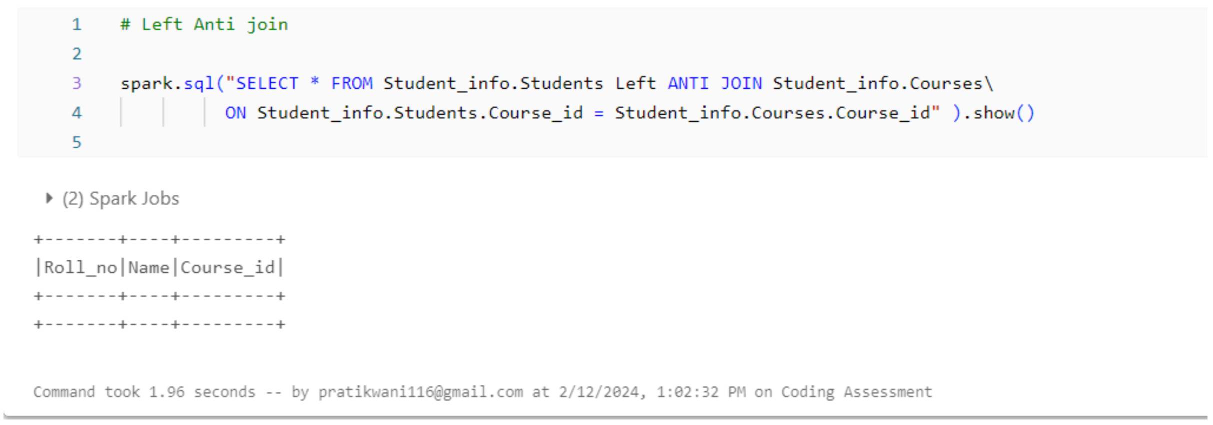
```
# Right Join

spark.sql("SELECT * FROM Student_info.Students RIGHT JOIN Student_info.Courses\
        ON Student_info.Students.Course_id = Student_info.Courses.Course_id" ).show()
```

▸ (2) Spark Jobs

```
+-------+------------+---------+-------------------+---------+
|Roll_no|        Name|Course_id|        Course_name|Course_id|
+-------+------------+---------+-------------------+---------+
|      1|  Amit Kumar|      101|   Computer Science|      101|
|      2| Priya Patel|      102|Electrical Engine...|      102|
|      3|Rahul Sharma|      103|Mechanical Engine...|      103|
|      4|  Neha Gupta|      104|  Civil Engineering|      104|
|      5|Sandeep Singh|      105|        Mathematics|      105|
|   null|        null|     null|            Physics|      106|
|   null|        null|     null|          Chemistry|      107|
+-------+------------+---------+-------------------+---------+
```

Command took 1.92 seconds -- by pratikwani116@gmail.com at 2/12/2024, 1:00:31 PM on Coding Assessment

- Left Anti Join:
    - Left Anti Join takes all the unmatched rows from left Table
    - The Unmatched fields are filled with Null values
    - In this example there is no such unmatched row hence result set is empty



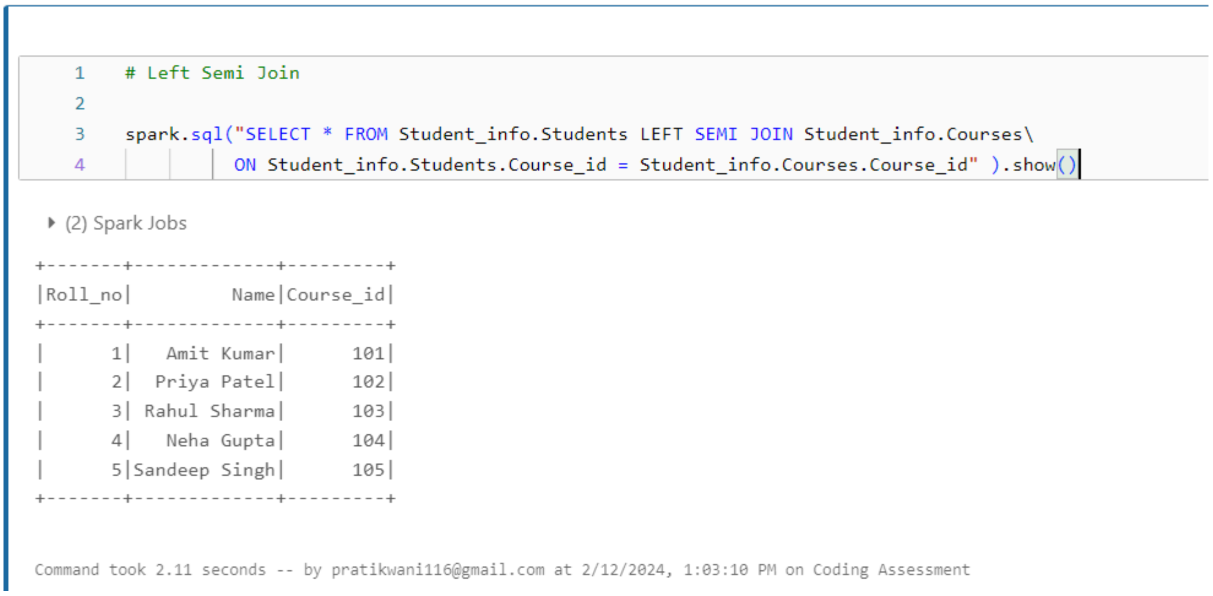
```
# Left Anti join

spark.sql("SELECT * FROM Student_info.Students Left ANTI JOIN Student_info.Courses\
            ON Student_info.Students.Course_id = Student_info.Courses.Course_id" ).show()

```

▸ (2) Spark Jobs

```
+-------+----+---------+
|Roll_no|Name|Course_id|
+-------+----+---------+
+-------+----+---------+
```

Command took 1.96 seconds -- by pratikwani116@gmail.com at 2/12/2024, 1:02:32 PM on Coding Assessment

- Left Semi Join:
    - Left Semi Join takes all the matched rows from left Table



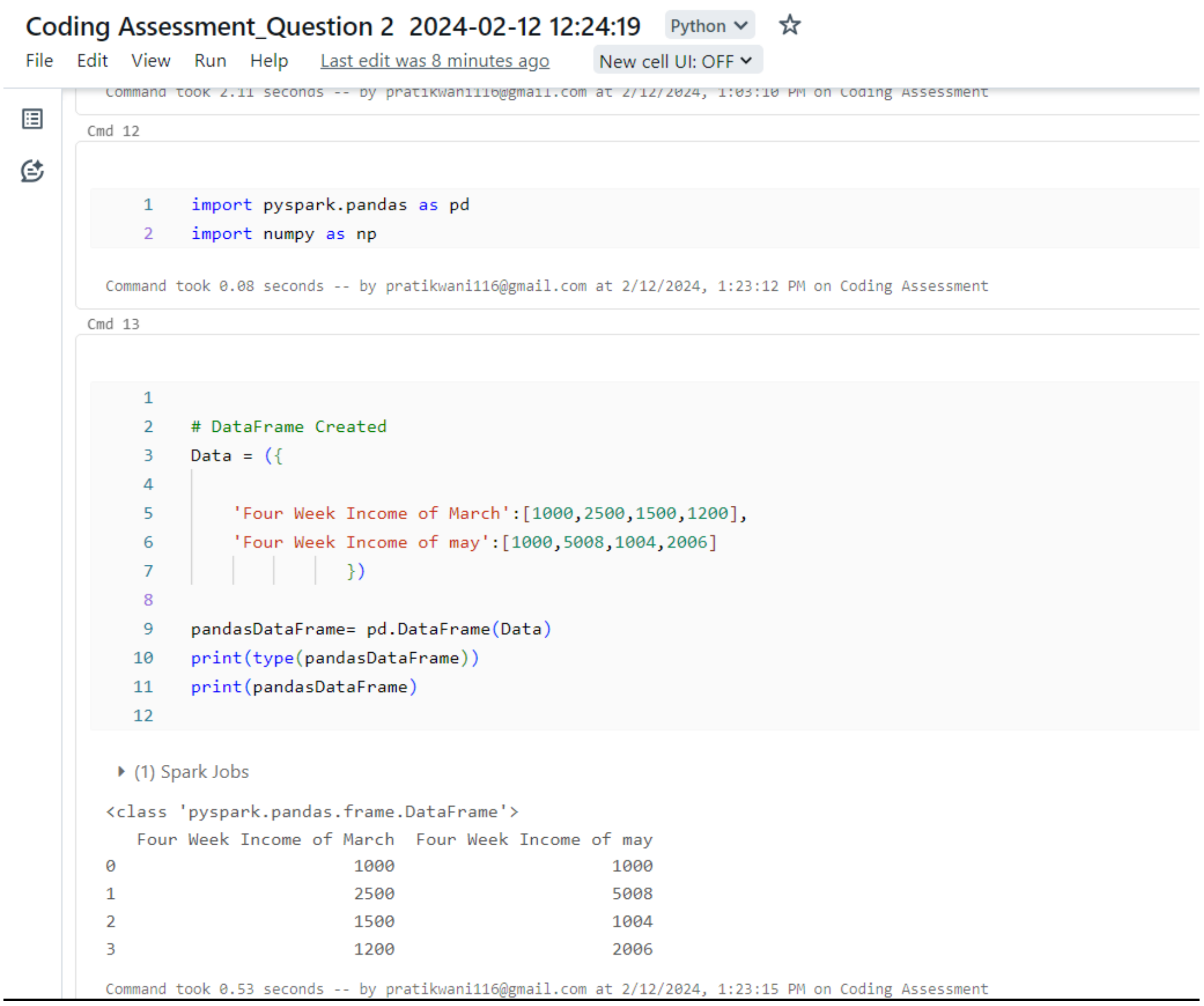
```
# Left Semi Join

spark.sql("SELECT * FROM Student_info.Students LEFT SEMI JOIN Student_info.Courses\
          ON Student_info.Students.Course_id = Student_info.Courses.Course_id" ).show()
```

▸ (2) Spark Jobs

```
+-------+------------+---------+
|Roll_no|        Name|Course_id|
+-------+------------+---------+
|      1|  Amit Kumar|      101|
|      2| Priya Patel|      102|
|      3|Rahul Sharma|      103|
|      4|  Neha Gupta|      104|
|      5|Sandeep Singh|      105|
+-------+------------+---------+
```

Command took 2.11 seconds -- by pratikwani116@gmail.com at 2/12/2024, 1:03:10 PM on Coding Assessment

- Applying Functions in a Pandas DataFrame
    - DataFrame is created in which we have weekly incomes
    - The function is created to get the total income of each month



Coding Assessment_Question 2 2024-02-12 12:24:19

Command took 2.11 seconds -- by pratikwani116@gmail.com at 2/12/2024, 1:03:10 PM on Coding Assessment

Cmd 12

```python
import pyspark.pandas as pd
import numpy as np
```

Command took 0.08 seconds -- by pratikwani116@gmail.com at 2/12/2024, 1:23:12 PM on Coding Assessment

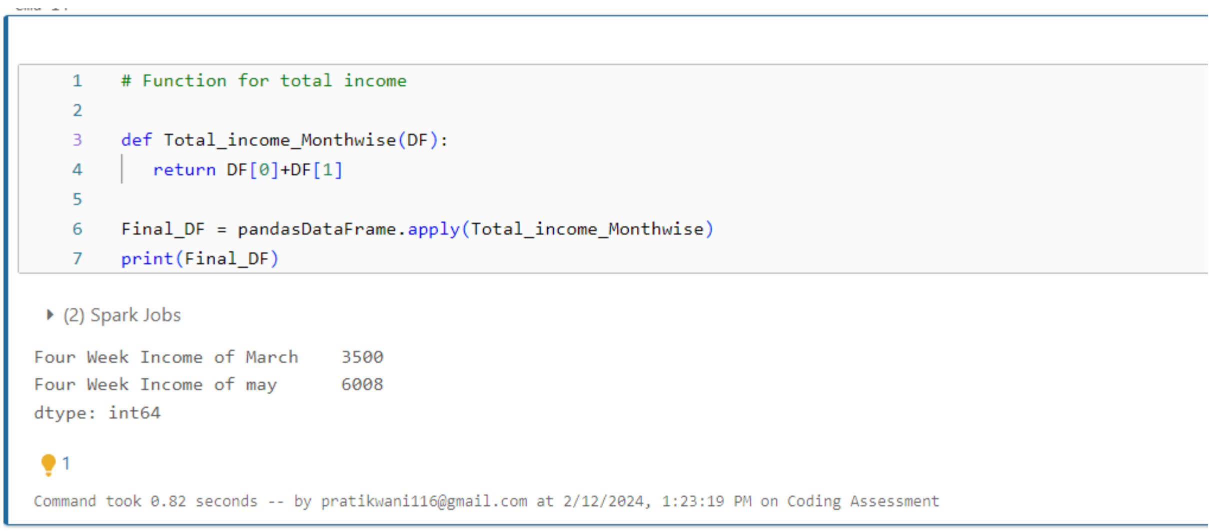
Cmd 13

```python

# DataFrame Created
Data = ({

    'Four Week Income of March':[1000,2500,1500,1200],
    'Four Week Income of may':[1000,5008,1004,2006]
                })

pandasDataFrame= pd.DataFrame(Data)
print(type(pandasDataFrame))
print(pandasDataFrame)

```

(1) Spark Jobs

```
<class 'pyspark.pandas.frame.DataFrame'>
   Four Week Income of March  Four Week Income of may
0                       1000                     1000
1                       2500                     5008
2                       1500                     1004
3                       1200                     2006
```

Command took 0.53 seconds -- by pratikwani116@gmail.com at 2/12/2024, 1:23:15 PM on Coding Assessment



```python
# Function for total income

def Total_income_Monthwise(DF):
    return DF[0]+DF[1]

Final_DF = pandasDataFrame.apply(Total_income_Monthwise)
print(Final_DF)
```

(2) Spark Jobs

```
Four Week Income of March    3500
Four Week Income of may      6008
dtype: int64
```

Command took 0.82 seconds -- by pratikwani116@gmail.com at 2/12/2024, 1:23:19 PM on Coding Assessment