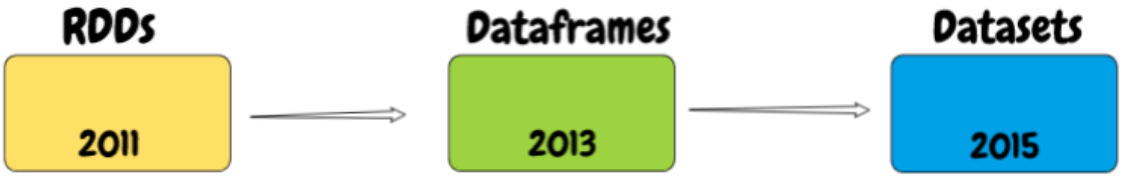
PYSPARK DATAFRAMES

DR. ALVIN ANG

CONTENTS



Source: https://www.analyticsvidhya.com/blog/2020/11/what-is-the-difference-between-rdds-dataframes-and-datasets/

- If you want a tutorial on RDD: https://sparkbyexamples.com/pyspark-rdd/
- If you want to know the differences between RDD / Dataframes / Datasets:
    - https://medium.com/quantyca/apache-spark-how-to-choose-the-correct-data-abstraction-8df7c6d8ec63
- Here, we only teach how to use Spark Dataframes.
- Because RDDs are outdated and slow. [1]

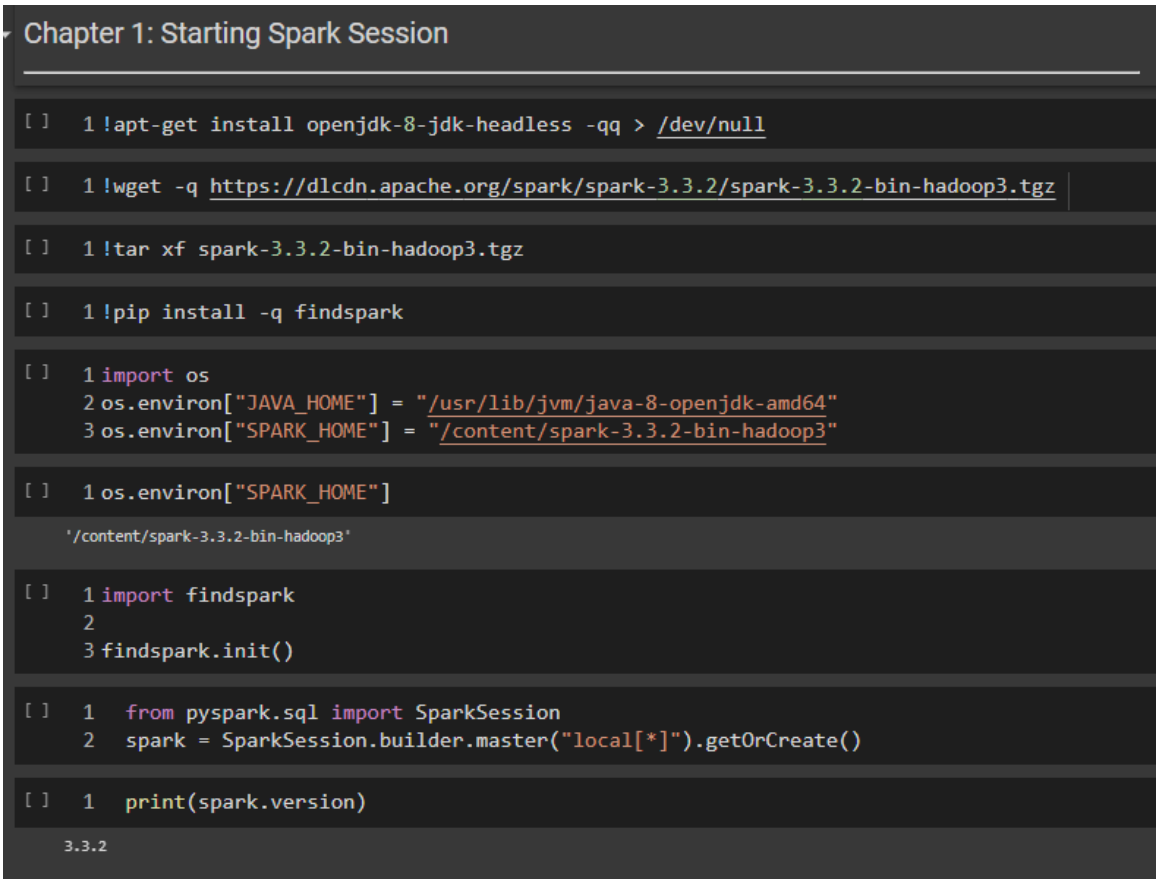
[1] https://medium.com/knoldus/apache-spark-3-reasons-why-you-should-not-use-rdds-cab554acd4c1

II. PYSPARK DATAFRAME – EXAMPLE 1

https://www.alvinang.sg/s/Dataframes_with_PySpark_Example_1_by_Dr_Alvin.ipynb

A. CHAPTER 1: START A SPARK SESSION

https://www.alvinang.sg/s/How_To_Start_A_Spark_Session__Read_in_CSV_frrom_Website.ipynb



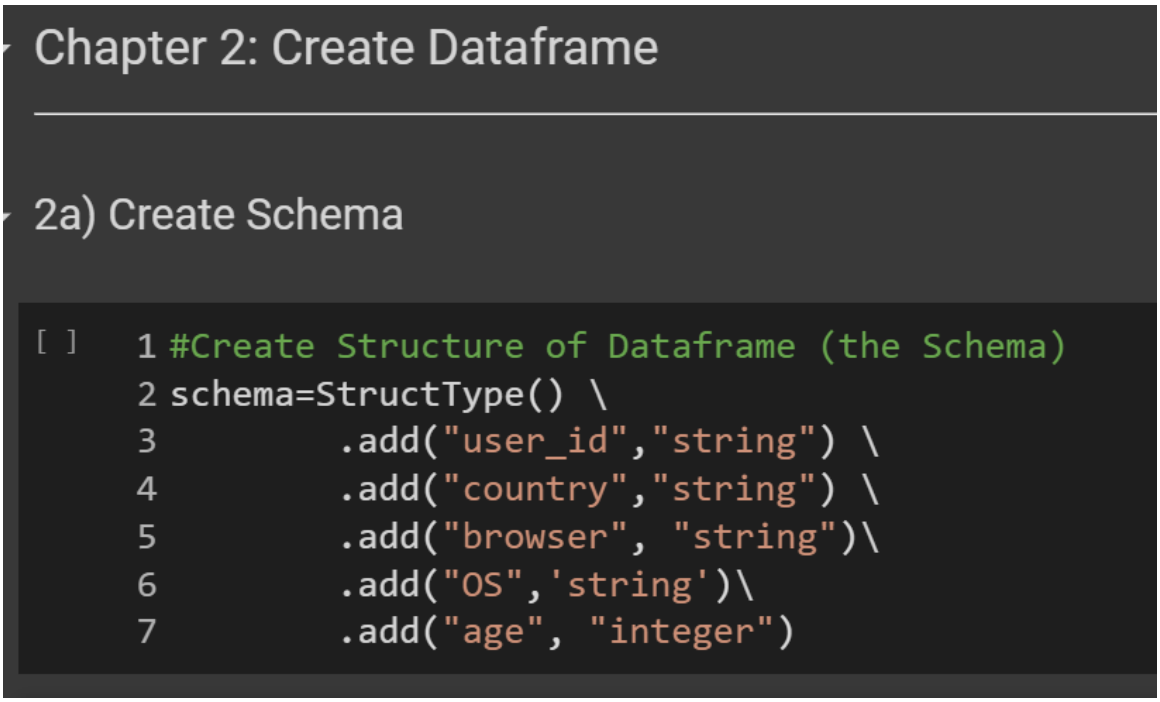
1. IMPORT LIBRARIES

1a) Importing Libraries

```
[ ]  1 from pyspark.sql import SparkSession
     2 spark=SparkSession.builder.appName('data_processing').getOrCreate()
     3
     4 import pyspark.sql.functions as F
     5 from pyspark.sql.types import *
```

B. CHAPTER 2: CREATE DATAFRAME

1. CREATING A SCHEMA



2. CREATE DATAFRAME FROM THE SCHEMA

2b) Create Dataframe From the Schema

```
[ ]  1 #Create Dataframe
     2 df=spark.createDataFrame([("A203",'India',"Chrome","WIN", 33) \
     3                          ,("A201",'China',"Safari","MacOS",35),\
     4                          ("A205",'UK',"Mozilla","Linux",25)],\
     5                         schema=schema)
```

```
     1 #Print the Schema
     2 df.printSchema()
```

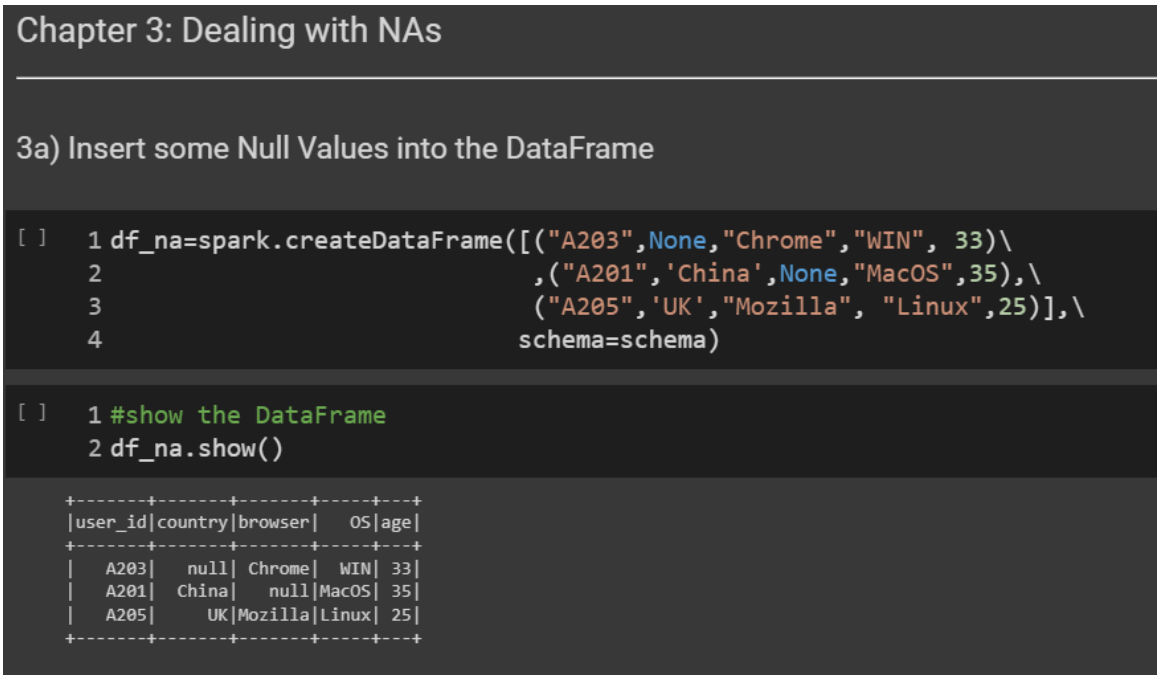
```
root
 |-- user_id: string (nullable = true)
 |-- country: string (nullable = true)
 |-- browser: string (nullable = true)
 |-- OS: string (nullable = true)
 |-- age: integer (nullable = true)
```

```
[ ]  1 #Show the Dataframe
     2 df.show()
```

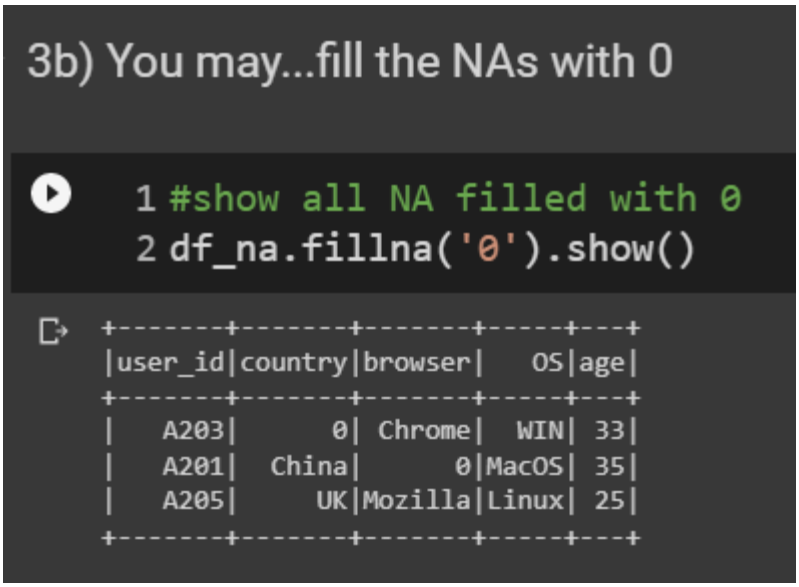
```
+-------+-------+-------+-----+---+
|user_id|country|browser|   OS|age|
+-------+-------+-------+-----+---+
|   A203|  India| Chrome|  WIN| 33|
|   A201|  China| Safari|MacOS| 35|
|   A205|     UK|Mozilla|Linux| 25|
+-------+-------+-------+-----+---+
```

C. CHAPTER 3: DEALING WITH NAS

1. INSERT SOME NULL VALUES INTO THE DATAFRAME

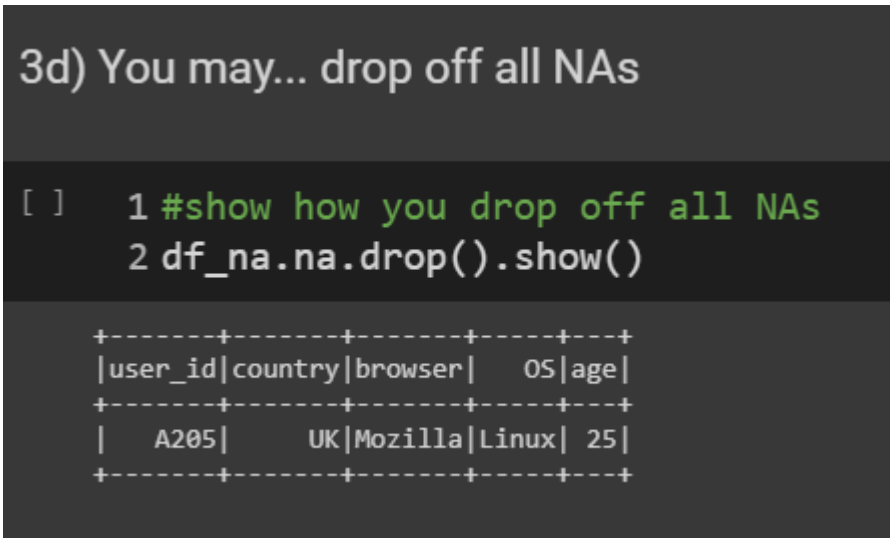


2. YOU MAY FILL NAS WITH 0

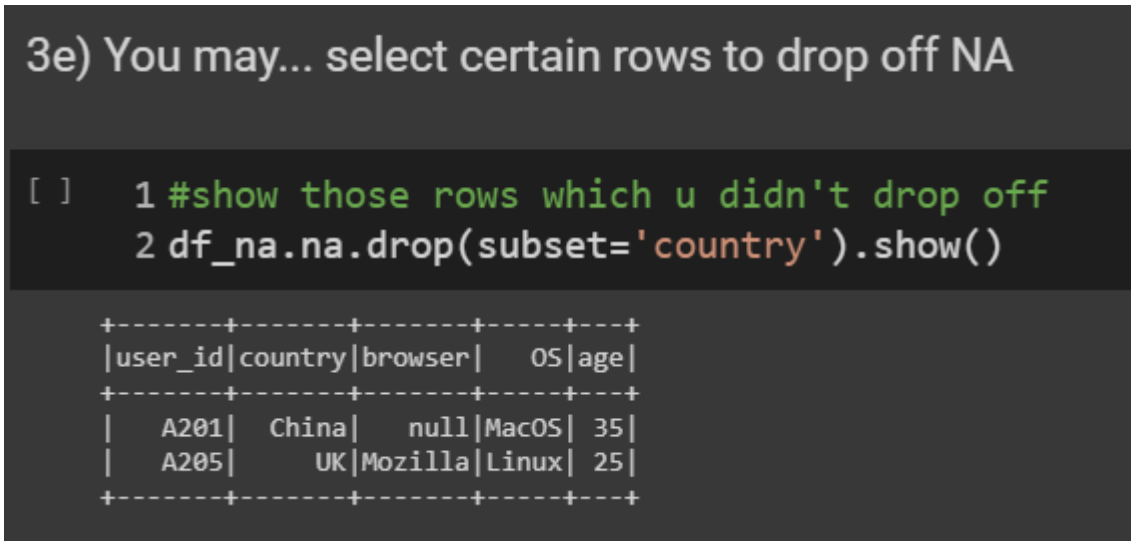


3. YOU MAY FILL NAS WITH VALUES

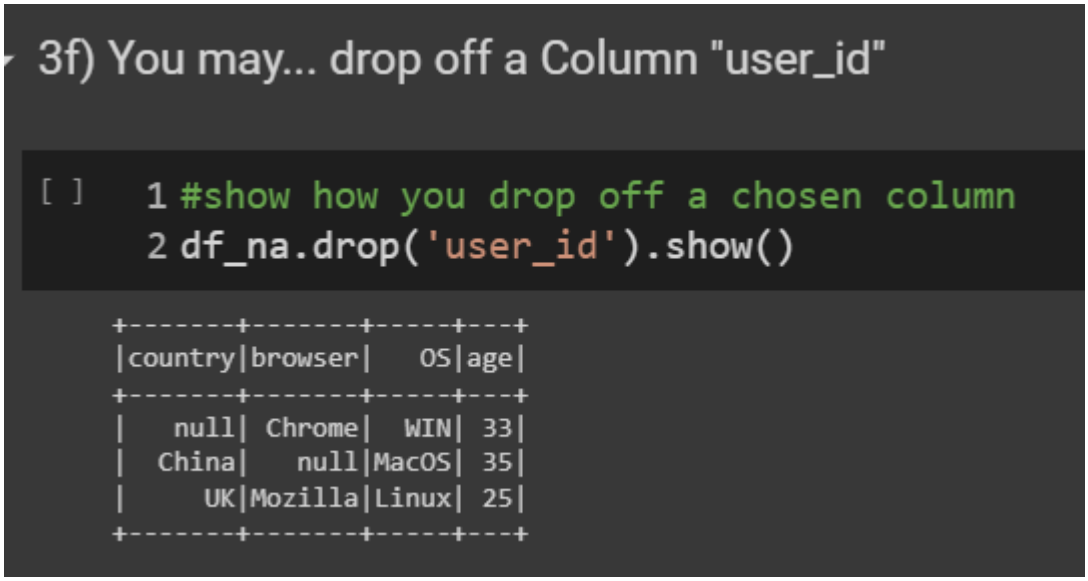
4. DROP ROWS WITH NAS



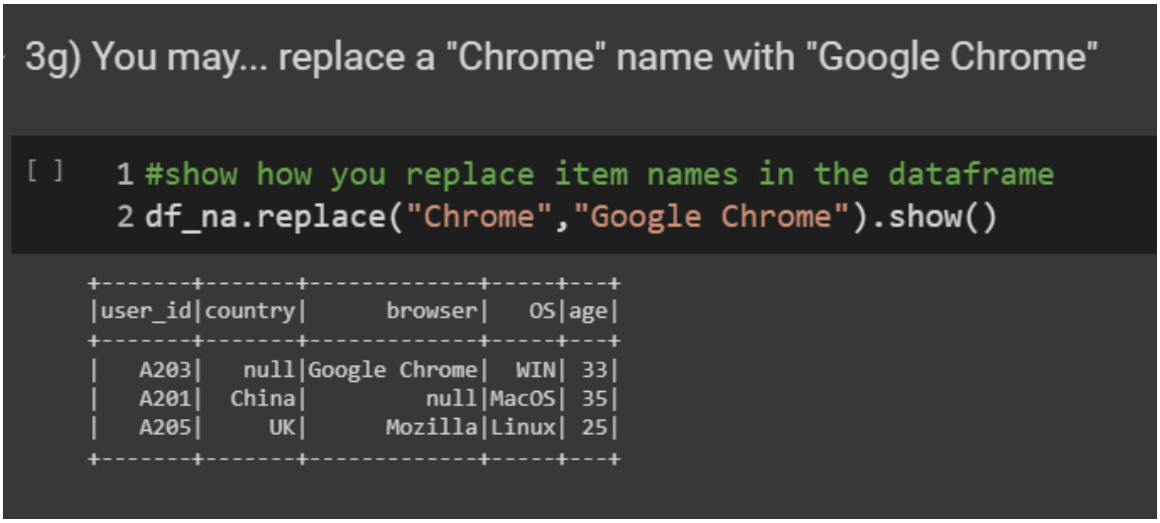
5. YOU MAY DROP A SELECTED COLUMN CONTAINING NA



6. YOU MAY DROP OFF A COLUMN



7. YOU MAY REPLACE A ‘CHROME’ NAME WITH ‘GOOGLE CHROME’



D. CHAPTER 4: HOW TO READ IN DATA FROM WEBSITE AND GET DESCRIPTIVE STATS

1. READING IN CSV FROM ALVIN'S WEBSITE

Chapter 4: How to Read in Data from Website & Get Descriptive Stats

4a) Reading in CSV from Alvin's Website

```
[ ] 1 #df=spark.read.csv("customer_data.csv",header=True, inferSchema=True)
    2
    3 #df.count()
```

```
[ ] 1 from pyspark import SparkFiles
    2
    3 url = 'https://www.alvinang.sg/s/customer_data.csv'
    4 spark.sparkContext.addFile(url)
    5
    6 df = spark.read.csv(SparkFiles.get("customer_data.csv"), header=True, inferSchema=True)
    7 df
```

```
DataFrame[Customer_subtype: string, Number_of_houses: int, Avg_size_household: int, Avg_age: string, Customer_main_type: string, Avg_Salary: int, label: int]
```

```
[ ] 1 #how many columns are there?
    2 len(df.columns)
```

```
7
```

```
[ ] 1 #what are the columns?
    2 df.printSchema()
```

```
root
 |-- Customer_subtype: string (nullable = true)
 |-- Number_of_houses: integer (nullable = true)
 |-- Avg_size_household: integer (nullable = true)
 |-- Avg_age: string (nullable = true)
 |-- Customer_main_type: string (nullable = true)
 |-- Avg_Salary: integer (nullable = true)
 |-- label: integer (nullable = true)
```

```
[ ] 1 #show the first 3 rows?
    2 df.show(3)
```

```
+--------------------+----------------+------------------+-----------+--------------------+----------+-----+
|    Customer_subtype|Number_of_houses|Avg_size_household|    Avg_age|  Customer_main_type|Avg_Salary|label|
+--------------------+----------------+------------------+-----------+--------------------+----------+-----+
|Lower class large...|               1|                 3|30-40 years|Family with grown...|     44905|    0|
|Mixed small town ...|               1|                 2|30-40 years|Family with grown...|     37575|    0|
|Mixed small town ...|               1|                 2|30-40 years|Family with grown...|     27915|    0|
+--------------------+----------------+------------------+-----------+--------------------+----------+-----+
only showing top 3 rows
```

```
[ ] 1 #show a summary of the dataset?
    2 df.summary().show()
```

```
+-------+--------------------+------------------+------------------+-----------+--------------------+-----------------+------------------+
|summary|    Customer_subtype|  Number_of_houses|Avg_size_household|    Avg_age|  Customer_main_type|       Avg_Salary|             label|
+-------+--------------------+------------------+------------------+-----------+--------------------+-----------------+------------------+
|  count|                2000|              2000|              2000|       2000|                2000|             2000|              2000|
|   mean|                null|            1.1075|            2.6895|       null|                null|    1616908.0835|            0.0605|
| stddev|                null|0.3873225521186316|0.7914562220841646|       null|                null|6822647.757312146|0.2384705099001677|
|    min|Affluent senior a...|                 1|                 1|20-30 years|     Average Family|             1361|                 0|
|    25%|                null|                 1|                 2|       null|                null|            20315|                 0|
|    50%|                null|                 1|                 3|       null|                null|            31421|                 0|
|    75%|                null|                 1|                 3|       null|                null|            42949|                 0|
|    max| Young, low educated|                10|                 5|70-80 years|Successful hedonists|         48919896|                 1|
+-------+--------------------+------------------+------------------+-----------+--------------------+-----------------+------------------+
```

2. SELECT CERTAIN COLUMNS

4b) Select Certain Columns

```
[ ]  1 #show the Avg Salary of each Customer Sub Type?
     2 df.select(['Customer_subtype','Avg_Salary']).show()
```

```
+--------------------+----------+
|    Customer_subtype|Avg_Salary|
+--------------------+----------+
|Lower class large...|     44905|
|Mixed small town ...|     37575|
|Mixed small town ...|     27915|
|Modern, complete ...|     19504|
|  Large family farms|     34943|
|    Young and rising|     13064|
|Large religious f...|     29090|
|Lower class large...|      6895|
|Lower class large...|     35497|
|     Family starters|     30800|
|       Stable family|     39157|
|Modern, complete ...|     40839|
|Lower class large...|     30008|
|        Mixed rurals|     37209|
|    Young and rising|     45361|
|Lower class large...|     45650|
|Traditional families|     18982|
|Mixed apartment d...|     30093|
|Young all america...|     27097|
|Low income catholics|     23511|
+--------------------+----------+
only showing top 20 rows
```

3. USING FILTER TO FILTER THE DATASET

4c) Using Filter to Filter the Dataset

```
[ ]  1 #how many are there with Avg Salary more than $1M?
     2
     3 df.filter(df['Avg_Salary'] > 1000000).count()
```

```
128
```

```
[ ]  1 #show all those with Avg Salary more than $1M?
     2
     3 df.filter(df['Avg_Salary'] > 1000000).show()
```

```
+--------------------+---------------+------------------+-----------+--------------------+----------+-----+
|    Customer_subtype|Number_of_houses|Avg_size_household|    Avg_age|  Customer_main_type|Avg_Salary|label|
+--------------------+---------------+------------------+-----------+--------------------+----------+-----+
| High status seniors|              1|                 3|40-50 years|Successful hedonists|   4670288|    0|
| High status seniors|              1|                 3|50-60 years|Successful hedonists|   9561873|    0|
| High status seniors|              1|                 2|40-50 years|Successful hedonists|  18687005|    0|
| High status seniors|              1|                 2|40-50 years|Successful hedonists|  24139960|    0|
| High status seniors|              1|                 2|50-60 years|Successful hedonists|   6718606|    0|
|High Income, expe...|              1|                 3|40-50 years|Successful hedonists|  19347139|    0|
|High Income, expe...|              1|                 3|40-50 years|Successful hedonists|   5243902|    0|
| High status seniors|              2|                 3|40-50 years|Successful hedonists|   6138618|    0|
|High Income, expe...|              1|                 3|50-60 years|Successful hedonists|  24930053|    0|
| High status seniors|              1|                 2|50-60 years|Successful hedonists|  12545905|    1|
|High Income, expe...|              1|                 3|40-50 years|Successful hedonists|  29976435|    0|
| High status seniors|              1|                 2|50-60 years|Successful hedonists|  24639614|    0|
| High status seniors|              1|                 2|40-50 years|Successful hedonists|  16073966|    0|
|High Income, expe...|              1|                 4|40-50 years|Successful hedonists|  35032441|    1|
|High Income, expe...|              1|                 2|50-60 years|Successful hedonists|   8354410|    0|
| High status seniors|              1|                 1|60-70 years|Successful hedonists|  20241068|    0|
| High status seniors|              1|                 1|50-60 years|Successful hedonists|  45592572|    0|
| High status seniors|              1|                 2|50-60 years|Successful hedonists|  10289449|    0|
|High Income, expe...|              1|                 2|50-60 years|Successful hedonists|   5586401|    0|
| High status seniors|              1|                 2|50-60 years|Successful hedonists|  41699271|    0|
+--------------------+---------------+------------------+-----------+--------------------+----------+-----+
only showing top 20 rows
```

```
1 #show those with Avg Salary more than $0.5M AND with more than 2 houses?
2
3 df.filter(df['Avg_Salary'] > 500000).filter(df['Number_of_houses'] > 2).show()
```

```
+--------------------+---------------+------------------+-----------+--------------------+----------+-----+
|    Customer_subtype|Number_of_houses|Avg_size_household|    Avg_age|  Customer_main_type|Avg_Salary|label|
+--------------------+---------------+------------------+-----------+--------------------+----------+-----+
|Affluent senior a...|              3|                 2|50-60 years|Successful hedonists|    596723|    0|
|Affluent senior a...|              3|                 2|50-60 years|Successful hedonists|    944444|    0|
|Affluent senior a...|              3|                 2|50-60 years|Successful hedonists|    788477|    0|
|Affluent senior a...|              3|                 2|50-60 years|Successful hedonists|    994077|    0|
+--------------------+---------------+------------------+-----------+--------------------+----------+-----+
```

4. USING WHERE TO FILTER THE DATASET

4d) Using Where to Filter the Dataset

```
[ ]  1 #show those with Avg Salary more than $0.5M and more than 2 houses?
     2
     3 df.where((df['Avg_Salary'] > 500000) & (df['Number_of_houses'] > 2)).show()
```

```
+--------------------+----------------+-----------------+-----------+--------------------+----------+-----+
|    Customer_subtype|Number_of_houses|Avg_size_household|    Avg_age|  Customer_main_type|Avg_Salary|label|
+--------------------+----------------+-----------------+-----------+--------------------+----------+-----+
|Affluent senior a...|               3|                2|50-60 years|Successful hedonists|    596723|    0|
|Affluent senior a...|               3|                2|50-60 years|Successful hedonists|    944444|    0|
|Affluent senior a...|               3|                2|50-60 years|Successful hedonists|    788477|    0|
|Affluent senior a...|               3|                2|50-60 years|Successful hedonists|    994077|    0|
+--------------------+----------------+-----------------+-----------+--------------------+----------+-----+
```

5. USING GROUPBY TO COUNT

4e) Using Groupby to Count

```
[ ]  1 #count the number of different customers
     2 df.groupBy('Customer_main_type').count().show()
```

```
+--------------------+-----+
|  Customer_main_type|count|
+--------------------+-----+
|             Farmers|   93|
|       Career Loners|   15|
|Retired and Relig...|  202|
|Successful hedonists|  194|
|         Living well|  178|
|      Average Family|  308|
|    Cruising Seniors|   60|
|Conservative fami...|  236|
|      Driven Growers|  172|
|Family with grown...|  542|
+--------------------+-----+
```

```python
#count the number of subtype customers and etc.. etc...
for col in df.columns:
  if col !='Avg_Salary':
    print(f" Aggregation for {col}")
    df.groupBy(col).count().orderBy('count',ascending=False).show(truncate=False)
```

```
 Aggregation for Customer_subtype
+----------------------------------------+-----+
|Customer_subtype                        |count|
+----------------------------------------+-----+
|Lower class large families              |288  |
|Traditional families                    |129  |
|Middle class families                   |122  |
|Large religious families                |107  |
|Modern, complete families               |93   |
|Couples with teens 'Married with children'|83   |
|Young and rising                        |78   |
|High status seniors                     |76   |
|Low income catholics                    |72   |
|Mixed seniors                           |71   |
|Village families                        |68   |
|Mixed rurals                            |67   |
|Young all american family               |62   |
|Stable family                           |62   |
|Large family, employed child            |56   |
|Young, low educated                     |56   |
|Family starters                         |55   |
|High Income, expensive child            |52   |
|Religious elderly singles               |47   |
|Mixed small town dwellers               |47   |
+----------------------------------------+-----+
only showing top 20 rows
```

```
 Aggregation for Number_of_houses
+----------------+-----+
|Number_of_houses|count|
+----------------+-----+
|1               |1808 |
|2               |178  |
|3               |12   |
|5               |1    |
|10              |1    |
+----------------+-----+

 Aggregation for Avg_size_household
+------------------+-----+
|Avg_size_household|count|
+------------------+-----+
|3                 |900  |
|2                 |730  |
|4                 |255  |
|1                 |94   |
|5                 |21   |
+------------------+-----+

 Aggregation for Avg_age
+-----------+-----+
|Avg_age    |count|
+-----------+-----+
|40-50 years|1028 |
|30-40 years|496  |
|50-60 years|373  |
|60-70 years|64   |
|20-30 years|31   |
|70-80 years|8    |
+-----------+-----+
```

```
 Aggregation for Customer_main_type
+--------------------+-----+
|Customer_main_type  |count|
+--------------------+-----+
|Family with grown ups|542  |
|Average Family      |308  |
|Conservative families|236  |
|Retired and Religious|202  |
|Successful hedonists |194  |
|Living well         |178  |
|Driven Growers      |172  |
|Farmers             |93   |
|Cruising Seniors    |60   |
|Career Loners       |15   |
+--------------------+-----+

 Aggregation for label
+-----+-----+
|label|count|
+-----+-----+
|0    |1879 |
|1    |121  |
+-----+-----+
```

6. USING .AGG TO FIND MEAN

4f) Using .AGG to find Mean

```
1 #Find the average number of houses per Customer type
2 df.groupBy('Customer_main_type').agg(F.mean('Number_of_houses')).show()
```

```
+--------------------+----------------------+
|  Customer_main_type|avg(Number_of_houses)|
+--------------------+----------------------+
|             Farmers|                   1.0|
|       Career Loners|    1.8666666666666667|
|Retired and Relig...|    1.0247524752475248|
|Successful hedonists|    1.0670103092783505|
|         Living well|    1.1629213483146068|
|      Average Family|     1.12012987012987|
|    Cruising Seniors|                   1.4|
|Conservative fami...|    1.0889830508474576|
|      Driven Growers|     1.180232558139535|
|Family with grown...|    1.0774907749077491|
+--------------------+----------------------+
```

7. USING .AGG TO FIND MAX

4g) Using .AGG to find Max

```
1 #find the Maximum number of houses per Customer type
2 df.groupBy('Customer_main_type').agg(F.max('Number_of_houses')).show()
```

```
+--------------------+----------------------+
|  Customer_main_type|max(Number_of_houses)|
+--------------------+----------------------+
|             Farmers|                     1|
|       Career Loners|                    10|
|Retired and Relig...|                     2|
|Successful hedonists|                     3|
|         Living well|                     5|
|      Average Family|                     3|
|    Cruising Seniors|                     3|
|Conservative fami...|                     2|
|      Driven Growers|                     2|
|Family with grown...|                     2|
+--------------------+----------------------+
```

8. USING .AGG TO FIND MIN

4h) Using .AGG to find Min

```
1 #find the Minimum no. of houses per Customer type
2 df.groupBy('Customer_main_type').agg(F.min('Number_of_houses')).show()
```

```
+--------------------+----------------------+
|  Customer_main_type|min(Number_of_houses)|
+--------------------+----------------------+
|             Farmers|                     1|
|       Career Loners|                     1|
|Retired and Relig...|                     1|
|Successful hedonists|                     1|
|         Living well|                     1|
|      Average Family|                     1|
|    Cruising Seniors|                     1|
|Conservative fami...|                     1|
|      Driven Growers|                     1|
|Family with grown...|                     1|
+--------------------+----------------------+
```

9. USING .AGG TO FIND SUM

4i) Using .AGG to find Sum

```
1 #find the total number of houses per Customer type
2 df.groupBy('Customer_main_type').agg(F.sum('Number_of_houses')).show()
```

```
+--------------------+----------------------+
|  Customer_main_type|sum(Number_of_houses)|
+--------------------+----------------------+
|             Farmers|                    93|
|       Career Loners|                    28|
|Retired and Relig...|                   207|
|Successful hedonists|                   207|
|         Living well|                   207|
|      Average Family|                   345|
|    Cruising Seniors|                    84|
|Conservative fami...|                   257|
|      Driven Growers|                   203|
|Family with grown...|                   584|
+--------------------+----------------------+
```

10. USING. SORT TO SORT

4j) Using .Sort to Sort

```
1 #Sort in Descending Order per Avg Salary
2 df.sort("Avg_Salary", ascending=False).show()
```

```
+--------------------+---------------+-----------------+-----------+--------------------+----------+-----+
|    Customer_subtype|Number_of_houses|Avg_size_household|    Avg_age|  Customer_main_type|Avg_Salary|label|
+--------------------+---------------+-----------------+-----------+--------------------+----------+-----+
| High status seniors|              1|                2|60-70 years|Successful hedonists|  48919896|    0|
|High Income, expe...|              1|                2|50-60 years|Successful hedonists|  48177970|    0|
|High Income, expe...|              1|                2|50-60 years|Successful hedonists|  48069548|    1|
|High Income, expe...|              1|                3|40-50 years|Successful hedonists|  46911924|    0|
| High status seniors|              1|                3|40-50 years|Successful hedonists|  46614009|    0|
|High Income, expe...|              1|                3|30-40 years|Successful hedonists|  45952441|    0|
|High Income, expe...|              1|                3|40-50 years|Successful hedonists|  45864609|    1|
| High status seniors|              1|                1|50-60 years|Successful hedonists|  45592572|    0|
| High status seniors|              1|                2|50-60 years|Successful hedonists|  45170899|    0|
|High Income, expe...|              1|                3|50-60 years|Successful hedonists|  44843830|    0|
| High status seniors|              1|                2|50-60 years|Successful hedonists|  43230349|    0|
| High status seniors|              1|                5|40-50 years|Successful hedonists|  43181830|    0|
| High status seniors|              1|                1|30-40 years|Successful hedonists|  42631926|    0|
| High status seniors|              1|                2|50-60 years|Successful hedonists|  41919020|    0|
| High status seniors|              1|                2|50-60 years|Successful hedonists|  41699271|    0|
| High status seniors|              1|                2|40-50 years|Successful hedonists|  41398953|    0|
|High Income, expe...|              1|                3|40-50 years|Successful hedonists|  41269615|    1|
| High status seniors|              1|                2|50-60 years|Successful hedonists|  41192397|    1|
| High status seniors|              1|                2|40-50 years|Successful hedonists|  40564335|    0|
| High status seniors|              1|                2|50-60 years|Successful hedonists|  40453887|    0|
```

11. USING .ORDERBY TO ORDERBY

4k) Using .Orderby to Orderby

```
[ ]  1 #Order the Customer Subtype by their Mean Salary (descending order)
     2 df.groupBy('Customer_subtype').agg(F.avg('Avg_Salary').\
     3                                    alias('mean_salary')).\
     4                                    orderBy('mean_salary',ascending=False).\
     5                                    show(50,False)
```

```
+----------------------------------------+-------------------+
|Customer_subtype                        |mean_salary        |
+----------------------------------------+-------------------+
|High status seniors                     |2.507677857894737E7 |
|High Income, expensive child            |2.3839817807692308E7|
|Affluent young families                 |662068.7777777778   |
|Affluent senior apartments              |653638.8235294118   |
|Senior cosmopolitans                    |49903.0             |
|Students in apartments                  |35532.142857142855  |
|Large family farms                      |33135.61538461538   |
|Young, low educated                     |33072.21428571428   |
|Large family, employed child            |32867.857142857145  |
|Suburban youth                          |32558.0             |
|Village families                        |32449.470588235294  |
|Middle class families                   |31579.385245901638  |
|Modern, complete families               |31576.0             |
|Etnically diverse                       |31572.0             |
|Young and rising                        |30795.897435897437  |
|Mixed seniors                           |30759.267605633802  |
|Very Important Provincials              |30548.40625         |
|Religious elderly singles               |30540.59574468085   |
|Family starters                         |30376.2             |
|Career and childcare                    |30110.939393939392  |
```

```
#Order the Customer Subtype by their Maximum Salary (descending order)
df.groupBy('Customer_subtype').agg(F.max('Avg_Salary').\
                                   alias('max_salary')).\
                                   orderBy('max_salary',ascending=False).\
                                   show()
```

```
+--------------------+----------+
|    Customer_subtype|max_salary|
+--------------------+----------+
| High status seniors|  48919896|
|High Income, expe...|  48177970|
|Affluent senior a...|    994077|
|Affluent young fa...|    991838|
|Traditional families|     49965|
|  Large family farms|     49965|
|Middle class fami...|     49932|
|Senior cosmopolitans|     49903|
|Mixed small town ...|     49901|
|Lower class large...|     49899|
|       Mixed seniors|     49876|
|    Young and rising|     49816|
|        Mixed rurals|     49785|
|Modern, complete ...|     49729|
| Young, low educated|     49626|
|Mixed apartment d...|     49621|
|     Family starters|     49602|
|    Village families|     49575|
|Religious elderly...|     49564|
|       Stable family|     49548|
+--------------------+----------+
only showing top 20 rows
```

```
df.groupBy('Customer_subtype').agg(F.max('Avg_Salary').\
                                   alias('max_salary')).\
                                   orderBy('max_salary',ascending=False).\
                                   show()
```

```
+--------------------+----------+
|    Customer_subtype|max_salary|
+--------------------+----------+
| High status seniors|  48919896|
|High Income, expe...|  48177970|
|Affluent senior a...|    994077|
|Affluent young fa...|    991838|
|Traditional families|     49965|
|  Large family farms|     49965|
|Middle class fami...|     49932|
|Senior cosmopolitans|     49903|
|Mixed small town ...|     49901|
+--------------------+----------+
```

notice that the alias changes the column name....
and the 'orderby' orders by the new column name...

E. CHAPTER 5: JOINING AND PIVOTING DATASETS

1. CREATE A NEW REGION DATASET…

Chapter 5: Joining and Pivoting Datasets

5a) Creating a New Dataset

```
[ ]  1 #we create a new region dataset
     2
     3 region_data = spark.createDataFrame([('Family with grown ups','PN'), \
     4                                       ('Driven Growers','GJ'),\
     5                                       ('Conservative families','DD'),\
     6                                       ('Cruising Seniors','DL'),\
     7                                       ('Average Family ','MN'),\
     8                                       ('Living well','KA'),\
     9                                       ('Successful hedonists','JH'),\
    10                                       ('Retired and Religious','AX'),\
    11                                       ('Career Loners','HY'),\
    12                                       ('Farmers','JH')],\
    13                                      schema=StructType().add("Customer_main_type","string").\
    14                                      add("Region Code","string"))
```

```
[ ]  1 region_data.show()
```

```
+--------------------+-----------+
|  Customer_main_type|Region Code|
+--------------------+-----------+
|Family with grown...|         PN|
|      Driven Growers|         GJ|
|Conservative fami...|         DD|
|    Cruising Seniors|         DL|
|     Average Family |         MN|
|         Living well|         KA|
|Successful hedonists|         JH|
|Retired and Relig...|         AX|
|       Career Loners|         HY|
|             Farmers|         JH|
+--------------------+-----------+
```

2. INNER JOIN 2 DATASETS

5b) Inner Join 2 Datasets

```
[ ]  1 #Inner join both 'Customer_Data.csv' and 'region_data' together
     2 #based on the 'Customer_main_type' column
     3 new_df=df.join(region_data,on='Customer_main_type')
```

```
     1 #now we have 'Region Code' new column for all rows
     2 new_df.show()
```

```
+--------------------+--------------------+---------------+------------------+-----------+----------+-----+-----------+
|  Customer_main_type|    Customer_subtype|Number_of_houses|Avg_size_household|    Avg_age|Avg_Salary|label|Region Code|
+--------------------+--------------------+---------------+------------------+-----------+----------+-----+-----------+
|Family with grown...|Lower class large...|              1|                 2|40-50 years|     25596|    0|         PN|
|Family with grown...|Mixed small town ...|              1|                 2|40-50 years|     26579|    0|         PN|
|Family with grown...|Lower class large...|              1|                 4|30-40 years|     33537|    0|         PN|
|Family with grown...|    Village families|              1|                 3|40-50 years|     22089|    0|         PN|
|Family with grown...|Lower class large...|              1|                 2|40-50 years|     38712|    0|         PN|
|Family with grown...|    Village families|              1|                 2|50-60 years|     21519|    0|         PN|
|Family with grown...|Lower class large...|              1|                 3|40-50 years|     43214|    0|         PN|
|Family with grown...|Couples with teen...|              1|                 2|50-60 years|     45028|    0|         PN|
|Family with grown...|Lower class large...|              1|                 3|40-50 years|     25747|    0|         PN|
|Family with grown...|Couples with teen...|              1|                 2|50-60 years|     42847|    0|         PN|
|Family with grown...|Lower class large...|              1|                 2|50-60 years|     31415|    0|         PN|
|Family with grown...|    Village families|              1|                 2|40-50 years|     17451|    0|         PN|
|Family with grown...|Lower class large...|              1|                 3|40-50 years|     29011|    0|         PN|
|Family with grown...|Lower class large...|              1|                 2|40-50 years|     42372|    0|         PN|
|Family with grown...|Lower class large...|              1|                 2|50-60 years|     34910|    0|         PN|
|Family with grown...|Lower class large...|              1|                 3|40-50 years|     10707|    1|         PN|
|Family with grown...|Lower class large...|              2|                 3|40-50 years|     19189|    0|         PN|
|Family with grown...|Lower class large...|              2|                 3|40-50 years|      5306|    0|         PN|
|Family with grown...|Couples with teen...|              1|                 2|40-50 years|     46423|    0|         PN|
|Family with grown...|Lower class large...|              1|                 3|40-50 years|     24911|    0|         PN|
+--------------------+--------------------+---------------+------------------+-----------+----------+-----+-----------+
only showing top 20 rows
```

From the `customer_data.csv` dataset

Customer_subtype	Number_of_houses	Avg_size_household	Avg_age	Customer_main_type	Avg_Salary	label
Lower class large...	1	3	30-40 years	Family with grown...	44905	0
Mixed small town ...	1	2	30-40 years	Family with grown...	37575	0
Mixed small town ...	1	2	30-40 years	Family with grown...	27915	0

From the new `region_data` dataframe created

Customer_main_type	Region Code
Family with grown...	PN
Driven Growers	GJ
Conservative fami...	DD
Cruising Seniors	DL
Average Family	MN
Living well	KA
Successful hedonists	JH
Retired and Relig...	AX
Career Loners	HY
Farmers	JH

did you notice that both datasets share this column?

```
[57] new_df=df.join(region_data,on='Customer_main_type')
```

this is taken from region_data

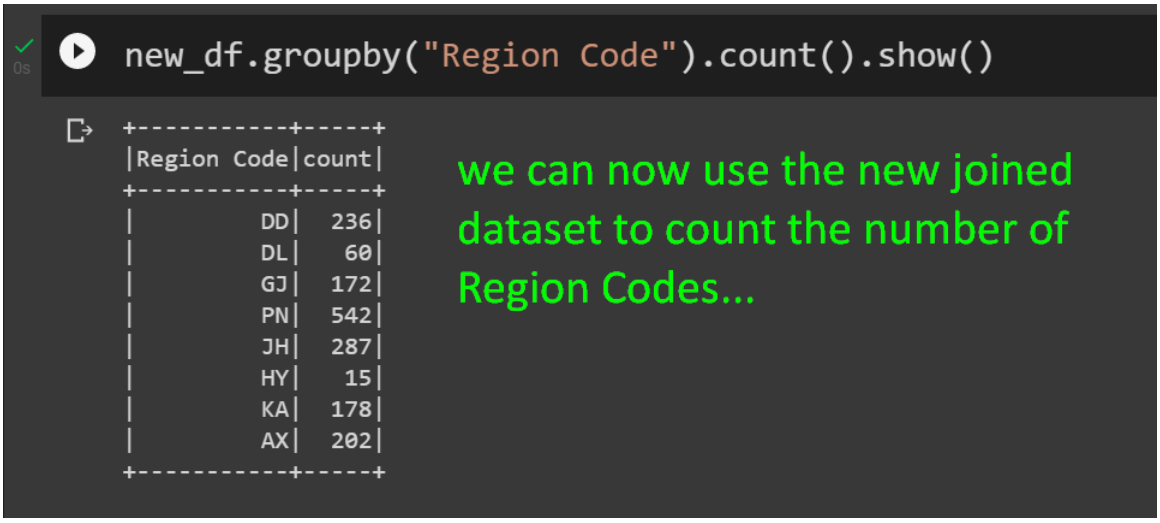
```
new_df.show()
```

this is the shared column between both datasets

this is taken from the customer_data.csv

Customer_main_type	Customer_subtype	Number_of_houses	Avg_size_household	Avg_age	Avg_Salary	label	constant	Region Code
Family with grown...	Lower class large...	1	2	40-50 years	25596	0	finance	PN
Family with grown...	Mixed small town ...	1	2	40-50 years	26579	0	finance	PN
Family with grown...	Lower class large...	1	4	30-40 years	33537	0	finance	PN
Family with grown...	Village families	1	3	40-50 years	22089	0	finance	PN
Family with grown...	Lower class large...	1	2	40-50 years	38712	0	finance	PN
Family with grown...	Village families	1	2	50-60 years	21519	0	finance	PN
Family with grown...	Lower class large...	1	3	40-50 years	43214	0	finance	PN
Family with grown...	Couples with teen...	1	2	50-60 years	45028	0	finance	PN
Family with grown...	Lower class large...	1	3	40-50 years	25747	0	finance	PN
Family with grown...	Couples with teen...	1	2	50-60 years	42847	0	finance	PN
Family with grown...	Lower class large...	1	2	50-60 years	31415	0	finance	PN
Family with grown...	Village families	1	2	40-50 years	17451	0	finance	PN
Family with grown...	Lower class large...	1	3	40-50 years	29011	0	finance	PN

- The "join" joins both datasets that share the same column "Customer_main_type"
- I believe its by default an inner join.



3. PIVOTING THE NEW DATASET

5c) Pivoting the new dataset

```
[ ]  1 #first, fill all NA with 0, then PIVOT by the "avg_age" and show the sum of the 'avg_salary'
     2 df.groupBy('Customer_main_type').pivot('Avg_age').sum('Avg_Salary').fillna(0).show()
```

```
+--------------------+----------+----------+----------+----------+----------+----------+
|  Customer_main_type|20-30 years|30-40 years|40-50 years|50-60 years|60-70 years|70-80 years|
+--------------------+----------+----------+----------+----------+----------+----------+
|            Farmers|         0|    462027|   2031235|    316206|         0|         0|
|       Career Loners|    143998|    176639|     25701|    105193|     32558|         0|
|Retired and Relig...|    126350|    336631|   2975266|   1687711|    335357|     61124|
|Successful hedonists|     42261| 171278764|1223362814|1563071675| 200340129|     15518|
|         Living well|    460528|   2965303|   1795405|    331304|         0|         0|
|      Average Family|         0|  23682805|   7789464|    412490|    226281|         0|
|    Cruising Seniors|         0|     43302|    303601|    529354|    716425|    139538|
|Conservative fami...|     69390|   2381485|   3449782|    915954|    146432|         0|
|      Driven Growers|         0|   1376260|   3407807|    424272|     83936|         0|
|Family with grown...|     16406|   2620378|   9132414|   3295378|    162820|     10496|
+--------------------+----------+----------+----------+----------+----------+----------+
```

```
 1 #first, fill all NA with 0, then PIVOT by the 'label' and show the sum of the 'avg_salary'
 2 df.groupBy('Customer_main_type').pivot('label').sum('Avg_Salary').fillna(0).show()
```

```
+--------------------+---------+--------+
|  Customer_main_type|        0|       1|
+--------------------+---------+--------+
|            Farmers|  2734832|   74636|
|       Career Loners|   484089|       0|
|Retired and Relig...|  5328410|  194029|
|Successful hedonists|2720381462|437729699|
|         Living well|  5453384|   99156|
|      Average Family| 26036999| 6074041|
|    Cruising Seniors|  1675841|   56379|
|Conservative fami...|  6595027|  368016|
|      Driven Growers|  4492465|  799810|
|Family with grown...| 14394094|  843798|
+--------------------+---------+--------+
```

```
df.groupBy('Customer_main_type').pivot('label').sum('Avg_Salary').fillna(0).show()
```

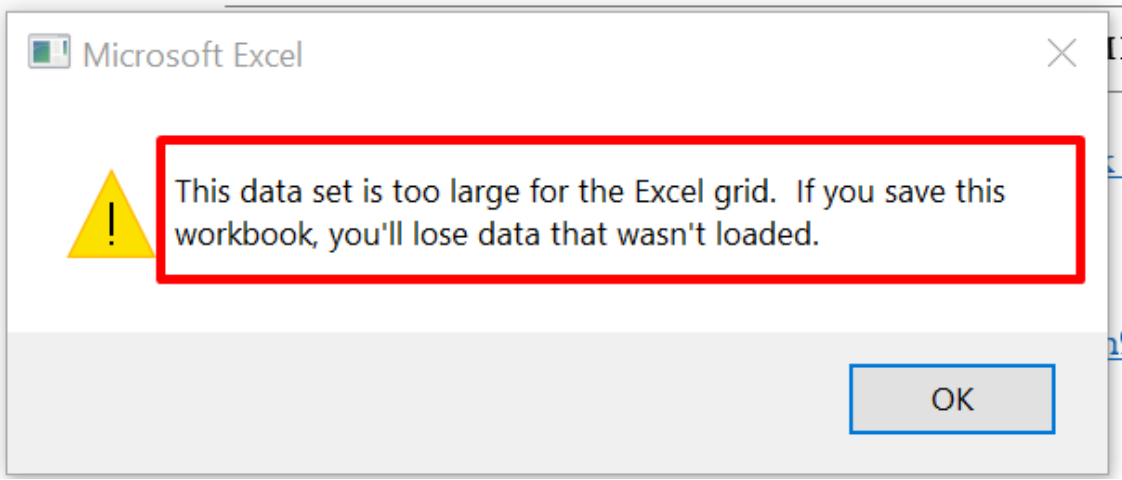
```
+--------------------+---------+--------+
|  Customer_main_type|        0|       1|
+--------------------+---------+--------+
|            Farmers|  2734832|   74636|
|       Career Loners|   484089|       0|
|Retired and Relig...|  5328410|  194029|
|Successful hedonists|2720381462|437729699|
|         Living well|  5453384|   99156|
|      Average Family| 26036999| 6074041|
|    Cruising Seniors|  1675841|   56379|
|Conservative fami...|  6595027|  368016|
|      Driven Growers|  4492465|  799810|
|Family with grown...| 14394094|  843798|
+--------------------+---------+--------+
```

THE END

https://www.alvinang.sg/s/DataFrames_with_PySpark_Example_2_by_Dr_Alvin_Ang.ipynb

Purchases.csv:

https://drive.google.com/file/d/1IlKKRkilXqUvIThm9JHcpLK6Y3-YfoI5/view?usp=share_link



- Purchases.csv is quite a big data set (4 million rows) and around 200MB.
- You will get the above error message if you try using Excel.
- This is where PySpark becomes useful.

A. CHAPTER 1: STARTING SPARK SESSION

Chapter 1: Starting Spark Session

```
[ ]  1 !apt-get install openjdk-8-jdk-headless -qq > /dev/null
```

```
[ ]  1 !wget -q https://dlcdn.apache.org/spark/spark-3.3.2/spark-3.3.2-bin-hadoop3.tgz
```

```
[ ]  1 !tar xf spark-3.3.2-bin-hadoop3.tgz
```

```
[ ]  1 !pip install -q findspark
```

```
[ ]  1 import os
     2 os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
     3 os.environ["SPARK_HOME"] = "/content/spark-3.3.2-bin-hadoop3"
```

```
[ ]  1 os.environ["SPARK_HOME"]
```

```
'/content/spark-3.3.2-bin-hadoop3'
```

```
[ ]  1 import findspark
     2
     3 findspark.init()
```

```
     1  from pyspark.sql import SparkSession
     2  spark = SparkSession.builder.master("local[*]").getOrCreate()
```

```
[ ]  1  print(spark.version)
```

```
3.3.2
```

1. IMPORT LIBRARIES

1a) Import Libraries

```
[ ]  1 #import module
     2 from pyspark.sql import SparkSession
     3 from pyspark.sql.types import *
     4
     5 # * stands for all
```

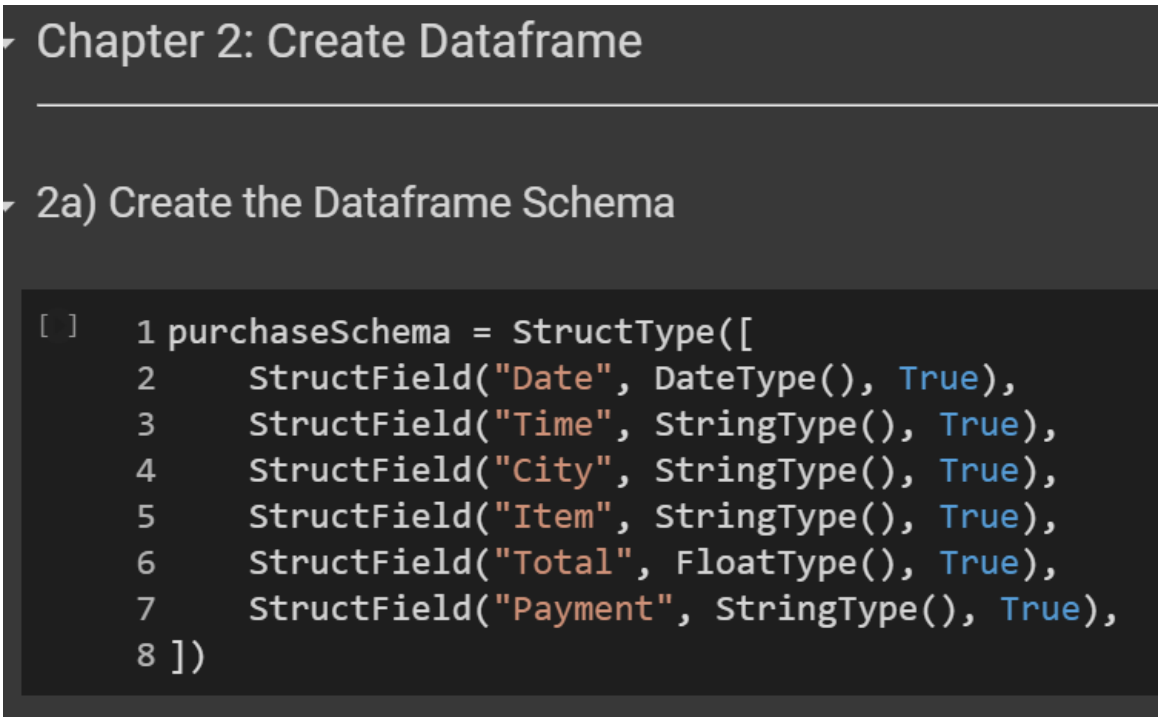
2. START SPARK SESSION

1b) Start Spark Session

```
[ ]  1 #create session in order to be capable of accessing all Spark API
     2 spark = SparkSession \
     3     .builder \
     4     .appName("Introdution to Spark DataFrame") \
     5     .config("spark.some.config.option", "some-value").getOrCreate()
```

B. CHAPTER 2: CREATE DATAFRAME

1. CREATE THE DATAFRAME SCHEMA



2. READ IN THE PURCHASES.CSV AND PUT INTO THE SCHEMA

2b) Read in the Purchases.csv and put into the Schema

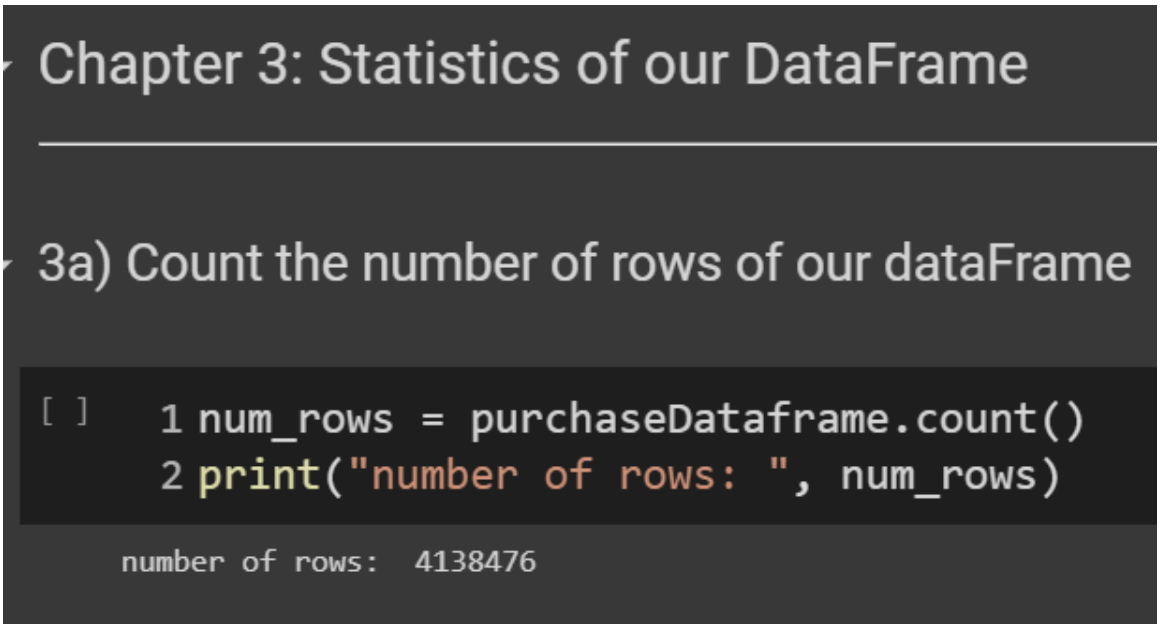
```
[ ]  1 purchaseDataframe = spark.read.csv(
     2     "/content/purchases.csv",
     3     header=True, schema=purchaseSchema, sep="\t")
```

```
     1 #Show the first 3 rows
     2 purchaseDataframe.show(3)
```

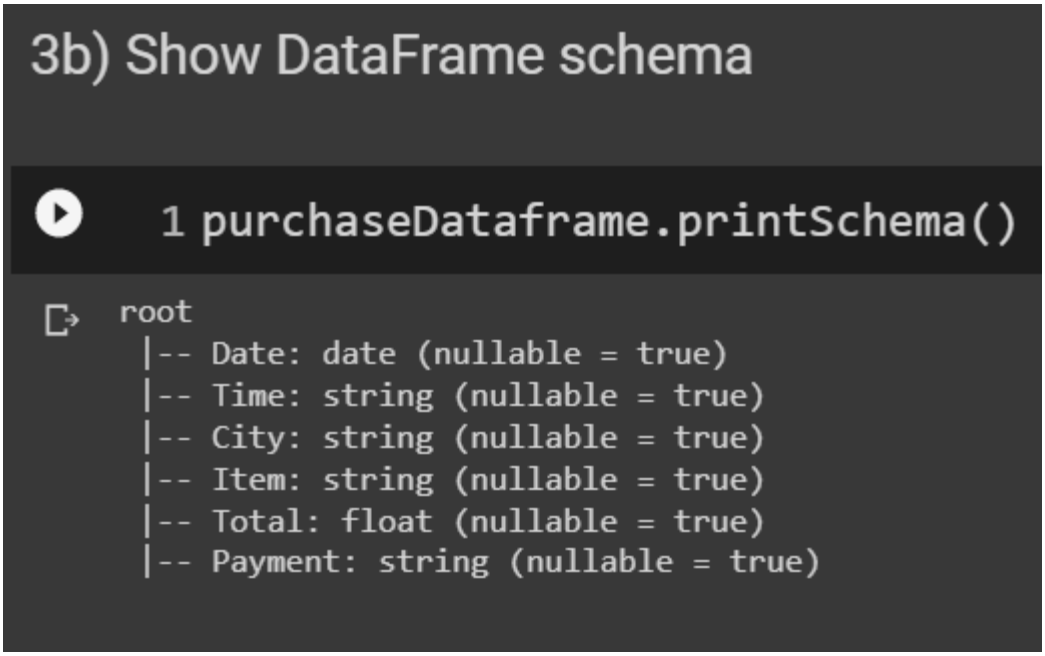
```
+----------+-----+----------+---------------+------+-------+
|      Date| Time|      City|           Item| Total|Payment|
+----------+-----+----------+---------------+------+-------+
|2012-01-01|09:00|  San Jose|  Men's Clothing|214.05|   Amex|
|2012-01-01|09:00|Fort Worth|Women's Clothing|153.57|   Visa|
|2012-01-01|09:00| San Diego|          Music| 66.08|   Cash|
+----------+-----+----------+---------------+------+-------+
only showing top 3 rows
```

C. CHAPTER 3: STATISTICS OF OUR DATAFRAME

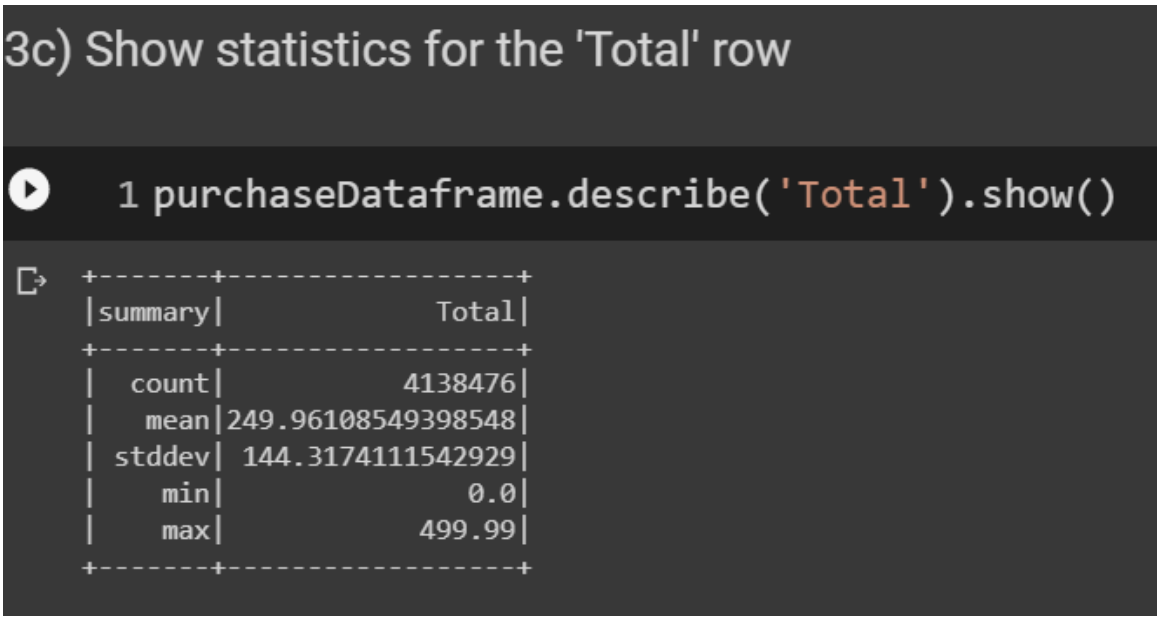
1. COUNT THE NUMBER OF ROWS OF OUR DATAFRAME



2. SHOW DATAFRAME SCHEMA



3. SHOW STATISTICS FOR THE 'TOTAL' ROW



D. CHAPTER 4: SUBSETTING OUR DATAFRAME

1. CREATE NEW DATAFRAME FROM "CITY" AND "TOTAL" COLUMNS

Chapter 4: Subsetting our DataFrame

4a) Create new dataFrame from "City" and "Total" columns

```
[ ]  1 newDataframe = purchaseDataframe.select(purchaseDataframe['City'],
     2                                          purchaseDataframe['Total'])
```

2. SHOW THE NEW DATAFRAME AND PRINT THE SCHEMA

4b) Show the New DataFrame and Print the Schema

```
1 newDataframe.show(3);
2 newDataframe.printSchema()
```

```
+----------+------+
|      City| Total|
+----------+------+
|  San Jose|214.05|
|Fort Worth|153.57|
| San Diego| 66.08|
+----------+------+
only showing top 3 rows

root
 |-- City: string (nullable = true)
 |-- Total: float (nullable = true)
```

E. CHAPTER 5: MANIPULATING OUR NEW DATAFRAME

1. ADD 10 TO EVERY ROW IN 'TOTAL' COLUMN

Chapter 5: Manipulating our New DataFrame

5a) Add 10 to every row in 'Total' column

```
1 purchaseDataframe.select(purchaseDataframe['City'],
2                          purchaseDataframe['Total']+10).show(3)
```

```
+----------+------------+
|      City|(Total + 10)|
+----------+------------+
|  San Jose|      224.05|
|Fort Worth|      163.57|
| San Diego|       76.08|
+----------+------------+
only showing top 3 rows
```

2. FILTER THOSE >20 IN "TOTAL" COLUMN

5b) Filter those > 200 in "Total" column

```
1 purchaseDataframe.filter(purchaseDataframe['Total'] > 200).show(3)
```

```
+----------+-----+----------+------------------+------+----------+
|      Date| Time|      City|              Item| Total|   Payment|
+----------+-----+----------+------------------+------+----------+
|2012-01-01|09:00|  San Jose|     Men's Clothing|214.05|      Amex|
|2012-01-01|09:00|Pittsburgh|       Pet Supplies|493.51|  Discover|
|2012-01-01|09:00|     Omaha|Children's Clothing|235.63|MasterCard|
+----------+-----+----------+------------------+------+----------+
only showing top 3 rows
```

3. ORDER BY CITY

5c) Order By City

```
[ ]  1 sortedByCity = purchaseDataframe.orderBy('City').show(4)
```

```
+----------+-----+-----------+----------------+------+----------+
|      Date| Time|       City|            Item| Total|   Payment|
+----------+-----+-----------+----------------+------+----------+
|2012-07-06|10:02|Albuquerque|Health and Beauty|  2.77|      Cash|
|2012-07-06|10:15|Albuquerque|     Pet Supplies|499.01|      Visa|
|2012-07-06|10:03|Albuquerque|            Toys| 50.47|MasterCard|
|2012-07-06|09:56|Albuquerque|           Music|425.03|      Cash|
+----------+-----+-----------+----------------+------+----------+
only showing top 4 rows
```

4. GROUPBY CITY

5d) Groupby city

```
 1 numTransactionEachCity = purchaseDataframe.groupBy("City").count()
 2 numTransactionEachCity.show(5)
```

```
+---------------+-----+
|           City|count|
+---------------+-----+
|North Las Vegas|40013|
|        Phoenix|40333|
|          Omaha|40209|
|      Anchorage|39806|
|        Anaheim|40086|
+---------------+-----+
only showing top 5 rows
```

F. CHAPTER 6: INDEXING OUR NEW DATAFRAME

Chapter 6: Indexing our New Dataframe

```
#Pyspark DataFrames are unlike Pandas Dataframe
#You cannot use .iloc / .loc to access rows or columns
#This is because you distributed the DataFrame (split it up) into multiple computers
#Thus u can't locate it using "row column" approach
#U need to create a new column called "Index ID"
```

1. CREATE A NEW INDEX COLUMN

6a) Create a New Index Column

```
#import 'monotonically_increasing_id' function
from pyspark.sql.functions import monotonically_increasing_id
```

```
#Create the New Index Column
newPurchasedDataframe = purchaseDataframe.withColumn(
    "index", monotonically_increasing_id())
```

```
#Show the new Dataframe
newPurchasedDataframe.show(7)
```

```
+----------+-----+----------+--------------------+------+----------+-----+
|      Date| Time|      City|                Item| Total|   Payment|index|
+----------+-----+----------+--------------------+------+----------+-----+
|2012-01-01|09:00|  San Jose|     Men's Clothing|214.05|      Amex|    0|
|2012-01-01|09:00|Fort Worth|   Women's Clothing|153.57|      Visa|    1|
|2012-01-01|09:00| San Diego|               Music| 66.08|      Cash|    2|
|2012-01-01|09:00|Pittsburgh|       Pet Supplies|493.51|  Discover|    3|
|2012-01-01|09:00|     Omaha|Children's Clothing|235.63|MasterCard|    4|
|2012-01-01|09:00|  Stockton|     Men's Clothing|247.18|MasterCard|    5|
|2012-01-01|09:00|    Austin|             Cameras| 379.6|      Visa|    6|
+----------+-----+----------+--------------------+------+----------+-----+
only showing top 7 rows
```

2. FILTER THE NEW DATAFRAME USING THE INDEX COLUMN

6b) Filter the New DataFrame using the Index Column

```
[ ]  1 #Filter out only Rows 2 to 4
     2 row2Till4 = newPurchasedDataframe.filter((newPurchasedDataframe['index']>=2) &
     3                                          (newPurchasedDataframe['index']<=4))
```

```
     1 #Show the new Filtered DataFrame
     2 row2Till4.show()
```

```
+----------+-----+----------+------------------+------+----------+-----+
|      Date| Time|      City|              Item| Total|   Payment|index|
+----------+-----+----------+------------------+------+----------+-----+
|2012-01-01|09:00| San Diego|             Music| 66.08|      Cash|    2|
|2012-01-01|09:00|Pittsburgh|      Pet Supplies|493.51|  Discover|    3|
|2012-01-01|09:00|     Omaha|Children's Clothing|235.63|MasterCard|    4|
+----------+-----+----------+------------------+------+----------+-----+
```

3. USE FILTER AND SELECT TO ACCESS THE PARTICULAR ROW AND COLUMN

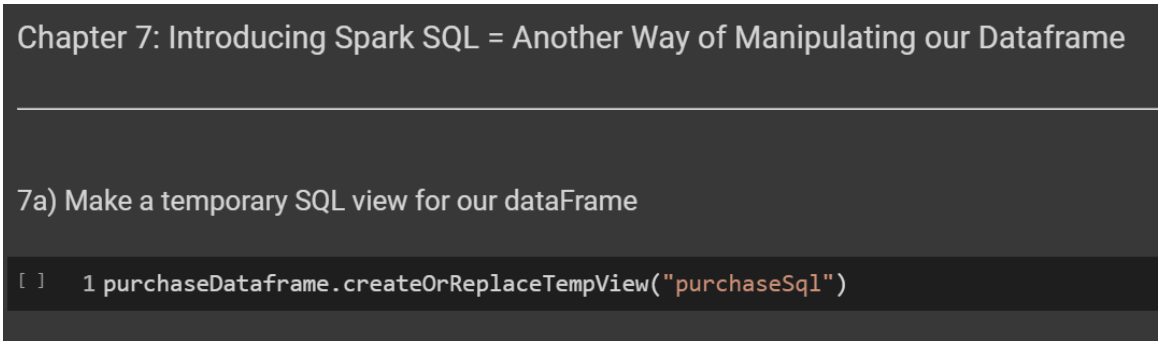
6c) Use Filter and Select to access the particular Row and Column

```
[ ]  1 dataRow2ColumnTotal = newPurchasedDataframe.filter(newPurchasedDataframe['index']==2).select('Total')
     2 dataRow2ColumnTotal.show()
```

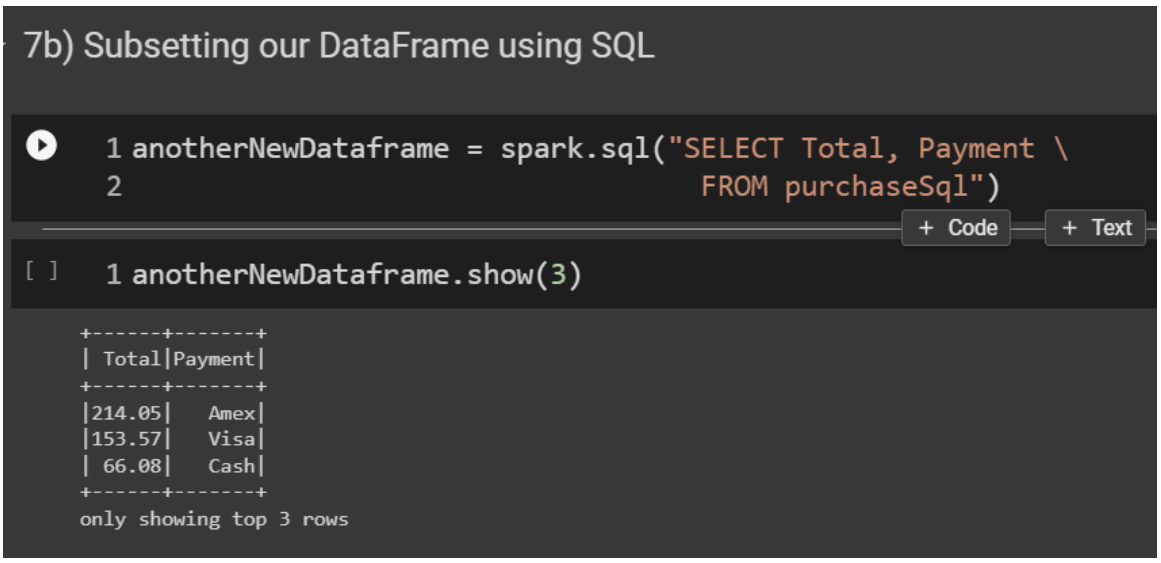
```
+-----+
|Total|
+-----+
|66.08|
+-----+
```

G. CHAPTER 7: INTRODUCING SPARK SQL = ANOTHER WAY OF MANIPULATING OUR DATAFRAME

1. MAKE A TEMPORARY SQL VIEW FOR OUR DATAFRAME



2. SUBSETTING OUR DATAFRAME USING SQL



3. ORDER BY CITY

7c) Order by City

```
[ ]  1 orderByCity = spark.sql("SELECT * FROM purchaseSql ORDER BY City")
```

```
     1 orderByCity.show(5)
```

```
+----------+-----+-----------+-------------------+------+----------+
|      Date| Time|       City|               Item| Total|   Payment|
+----------+-----+-----------+-------------------+------+----------+
|2012-07-06|10:02|Albuquerque|  Health and Beauty|  2.77|      Cash|
|2012-07-06|10:18|Albuquerque|             Crafts|  35.0|  Discover|
|2012-07-06|10:03|Albuquerque|               Toys| 50.47|MasterCard|
|2012-07-06|09:56|Albuquerque|              Music|425.03|      Cash|
|2012-07-06|10:04|Albuquerque|Children's Clothing|153.97|      Amex|
+----------+-----+-----------+-------------------+------+----------+
only showing top 5 rows
```

4. FILTER "TOTAL" COLUMN VALUE > 0 AND ORDER THEM BY "PAYMENT"

7d) Filter "Total" column value > 200, and Order them by "Payment"

```
[ ]  1 filterAndSortWithSQL = spark.sql("SELECT * FROM purchaseSql WHERE Total>200 ORDER BY Payment")
```

```
     1 filterAndSortWithSQL.show(4)
```

```
+----------+-----+--------------+---------------+------+-------+
|      Date| Time|          City|           Item| Total|Payment|
+----------+-----+--------------+---------------+------+-------+
|2012-07-06|09:46|    Long Beach|    Pet Supplies|482.56|   Amex|
|2012-07-06|09:47|       Oakland|      Computers| 338.9|   Amex|
|2012-07-06|09:46|St. Petersburg|Women's Clothing|311.51|   Amex|
|2012-07-06|09:45|       Spokane|  Men's Clothing|368.88|   Amex|
+----------+-----+--------------+---------------+------+-------+
only showing top 4 rows
```

THE END

IV. ABOUT DR. ALVIN ANG

Dr. Alvin Ang earned his Ph.D., Masters and Bachelor degrees from NTU, Singapore. He is a scientist, entrepreneur, as well as a personal/business advisor. More about him at www.AlvinAng.sg.