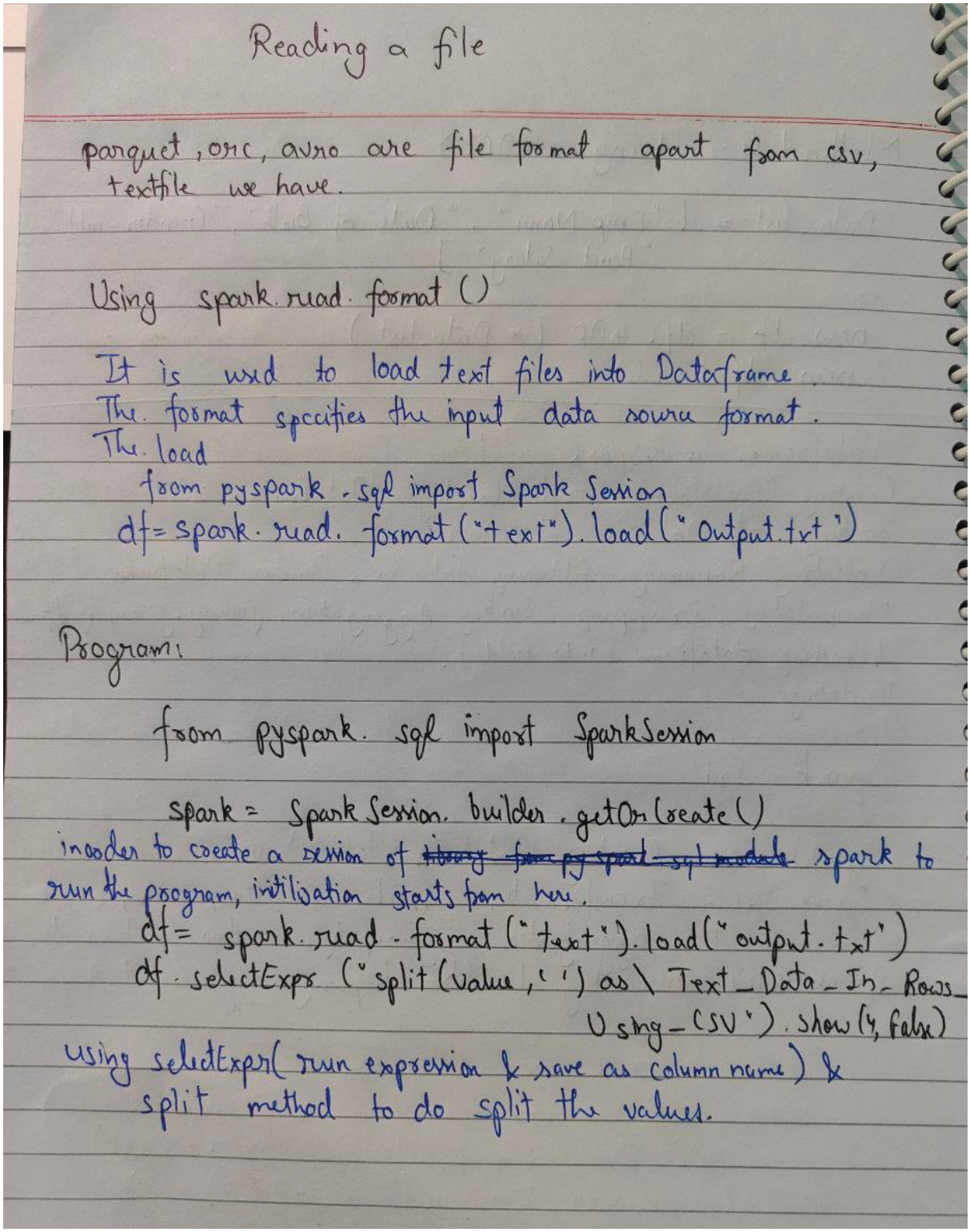
Assignment-14

Name-Ajay Chaudhary
Batch-Data Engineering(Batch-1)

Hand written notes-



Reading a file

parquet, orc, avro are file format apart from csv, textfile we have.

Using spark.read.format()

It is used to load text files into Dataframe.
The format specifies the input data source format.
The load

```
from pyspark.sql import SparkSession
df=spark.read.format("text").load("output.txt")
```

Program:

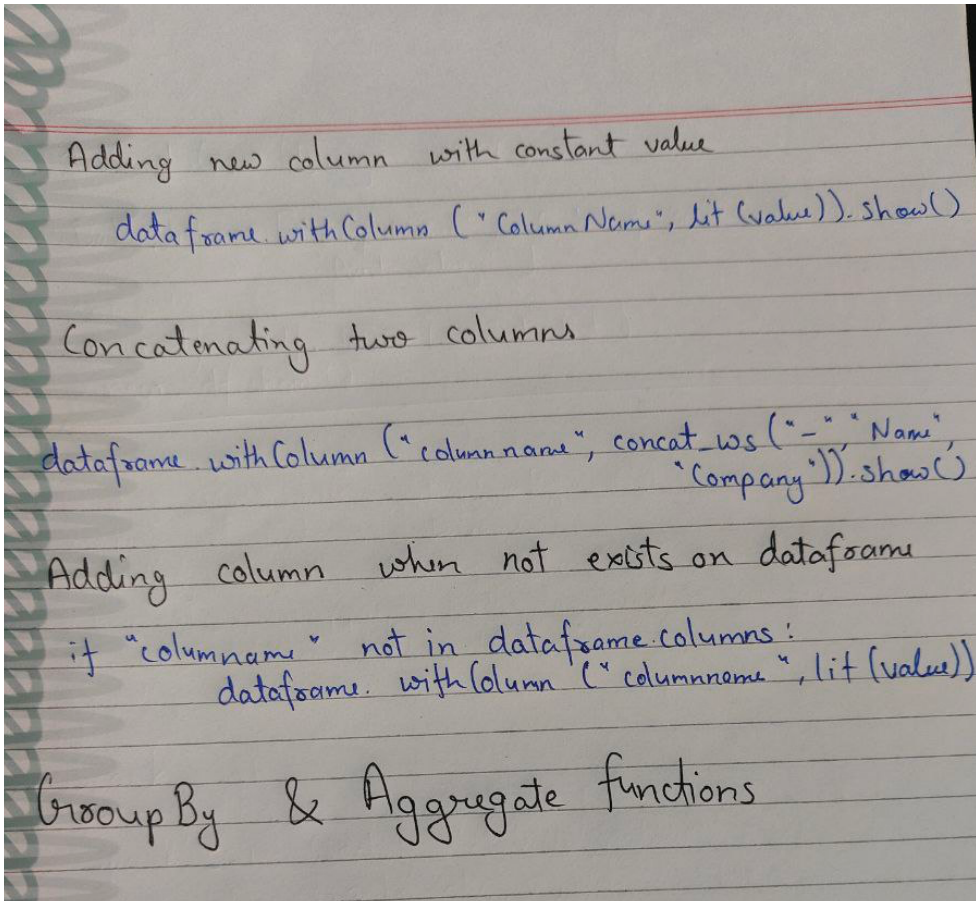
```
from pyspark.sql import SparkSession

spark = SparkSession.builder.getOrCreate()
```

inorder to create a session of ~~library from pyspark.sql module~~ spark to run the program, initialization starts from here.

```
df= spark.read.format("text").load("output.txt")
df.selectExpr("split(value,',') as \Text_Data_In_Rows_
                                    Using_CSV").show(4,False)
```

using selectExpr(run expression & save as column name) & split method to do split the values.



GroupBy and Aggregate function:

Similar to SQL GROUP BY clause, PySpark groupBy() function is used to collect the identical data into groups on DataFrame and perform count, sum, avg, min, and max functions on the grouped data.

[1]:
```
from pyspark.sql import SparkSession
```

[2]:
```
spark=SparkSession.builder.appName("PySpark_example").getOrCreate()
```

```
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
24/02/10 08:18:42 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
```

[3]:
```
df_pyspark=spark.read.csv("/Users/ajaychaudhary/jupyter/PySpark/test.csv",header=True,inferSchema=True)
```

[4]:
```
df_pyspark.show()
df_pyspark.groupBy("department").sum("salary").show()
```

```
+-------+------------+------+
|   name|  department|salary|
+-------+------------+------+
| Chandu|Data Science| 50000|
|Rashmi |         IOT| 75000|
|  Rohit|    Big Data| 55000|
| Rohit |    Big Data| 80000|
|  Ronit|         IOT| 60000|
|  Ronit|Data Science| 70000|
| Chandu|Data Science| 45000|
|Krishna|    Big Data| 65000|
| Rashmi|    Big Data| 85000|
|  Rohit|         IOT| 60000|
+-------+------------+------+
```

```
[4]: df_pyspark.show()
df_pyspark.groupBy("department").sum("salary").show()
```

```
+-------+-----------+------+
|   name|  department|salary|
+-------+-----------+------+
| Chandu|Data Science| 50000|
|Rashmi |         IOT| 75000|
|  Rohit|    Big Data| 55000|
| Rohit |    Big Data| 80000|
|  Ronit|         IOT| 60000|
|  Ronit|Data Science| 70000|
| Chandu|Data Science| 45000|
|Krishna|    Big Data| 65000|
| Rashmi|    Big Data| 85000|
|  Rohit|         IOT| 60000|
+-------+-----------+------+

+-----------+-----------+
|  department|sum(salary)|
+-----------+-----------+
|         IOT|     195000|
|    Big Data|     285000|
|Data Science|     165000|
+-----------+-----------+
```

Similarly we performed min, max, mean, avg, and count using the groupBy function

MIN() function

```
[5]: df_pyspark.groupBy("department").min("salary").show()
```

```
+-----------+-----------+
|  department|min(salary)|
+-----------+-----------+
|         IOT|      60000|
|    Big Data|      55000|
|Data Science|      45000|
+-----------+-----------+
```

Max() function

```
[6]: df_pyspark.groupBy("department").max("salary").show()
```

```
+-----------+-----------+
|  department|max(salary)|
+-----------+-----------+
|         IOT|      75000|
|    Big Data|      85000|
|Data Science|      70000|
+-----------+-----------+
```

Avg () function

```
[7]: df_pyspark.groupBy("department").avg("salary").show()
```

```
+-----------+-----------+
|  department|avg(salary)|
+-----------+-----------+
|         IOT|    65000.0|
|    Big Data|    71250.0|
|Data Science|    55000.0|
+-----------+-----------+
```

Mean() function

```
[8]: df_pyspark.groupBy("department").mean("salary").show()
```

```
+----------+-----------+
|  department|avg(salary)|
+----------+-----------+
|         IOT|    65000.0|
|    Big Data|    71250.0|
|Data Science|    55000.0|
+----------+-----------+
```

Count() function

```
[9]: df_pyspark.groupBy("department").count().show()
```

```
+----------+-----+
|  department|count|
+----------+-----+
|         IOT|    3|
|    Big Data|    4|
|Data Science|    3|
+----------+-----+
```

groupBy() using multiple columns

```
df_pyspark.groupBy("name","department").min("salary").show()
```

```
+-------+----------+-----------+
|   name|  department|min(salary)|
+-------+----------+-----------+
|  Rohit|    Big Data|      55000|
| Rashmi|    Big Data|      85000|
| Chandu|Data Science|      45000|
|  Ronit|Data Science|      70000|
| Rohit |    Big Data|      80000|
|  Ronit|         IOT|      60000|
|Krishna|    Big Data|      65000|
|Rashmi |         IOT|      75000|
|  Rohit|         IOT|      60000|
+-------+----------+-----------+
```

Using Pivot

```
Using pivot() function
```

```
11]: df_pyspark.groupBy("department").pivot("Name").sum("salary").show()
```

```
+-----------+------+-------+------+-------+-----+------+-----+
|  department|Chandu|Krishna|Rashmi|Rashmi |Rohit|Rohit |Ronit|
+-----------+------+-------+------+-------+-----+------+-----+
|         IOT|  NULL|   NULL|  NULL|  75000|60000|  NULL|60000|
|    Big Data|  NULL|  65000| 85000|   NULL|55000| 80000| NULL|
|Data Science| 95000|   NULL|  NULL|   NULL| NULL|  NULL|70000|
+-----------+------+-------+------+-------+-----+------+-----+
```

Handling Missing Values Pyspark

```
Handling missing value
```

```
df_pyspark1=spark.read.csv("/Users/ajaychaudhary/jupyter/PySpark/null_values_test.csv",header=True,inferSchema=True)
df_pyspark1.show()
```

```
+-------+----+----------+------+
|    name| age|experience|salary|
+-------+----+----------+------+
|   Ravi |  28|         5| 60000|
|  Priya |  35|      NULL| 75000|
|   Arun |  40|        12| 85000|
|   Neha |NULL|      NULL| 55000|
| Suresh |  45|        18| 90000|
|Deepika |NULL|      NULL| 65000|
| Rajesh |  30|         8| 70000|
|  Meera |NULL|      NULL| 50000|
| Sanjay |  38|        15| 80000|
|  Anita |NULL|        10| 60000|
+-------+----+----------+------+
```

```
[ ]: Dropping rows based on null values
```

```
[22]: df_pyspark1.na.drop().show()
```

```
+-------+---+----------+------+
|   name|age|experience|salary|
+-------+---+----------+------+
|  Ravi | 28|         5| 60000|
|  Arun | 40|        12| 85000|
|Suresh | 45|        18| 90000|
|Rajesh | 30|         8| 70000|
|Sanjay | 38|        15| 80000|
+-------+---+----------+------+
```

drop() has the following parameters — how, thresh, and subset

df_pyspark1.na.drop(how="all").show()
# if all values in rows are null then drop
# default any

df_pyspark1.na.drop(how="any",thresh=2).show()
#atleast 2 non null values should be present.

df_pyspark1.na.drop(how="any",subset=["salary"]).show()
# only in that column rows get deleted

Filling missing value- single value

[26]:
```
df_pyspark1.na.fill('Missing Values').show()
df_pyspark1.na.fill(0).show()
```

```
+--------+----+----------+------+
|    name| age|experience|salary|
+--------+----+----------+------+
|   Ravi |  28|         5| 60000|
|  Priya |  35|      NULL| 75000|
|   Arun |  40|        12| 85000|
|   Neha |NULL|      NULL| 55000|
| Suresh |  45|        18| 90000|
|Deepika |NULL|      NULL| 65000|
| Rajesh |  30|         8| 70000|
|  Meera |NULL|      NULL| 50000|
| Sanjay |  38|        15| 80000|
|  Anita |NULL|        10| 60000|
+--------+----+----------+------+

+--------+---+----------+------+
|    name|age|experience|salary|
+--------+---+----------+------+
|   Ravi | 28|         5| 60000|
|  Priya | 35|         0| 75000|
|   Arun | 40|        12| 85000|
|   Neha |  0|         0| 55000|
| Suresh | 45|        18| 90000|
|Deepika |  0|         0| 65000|
| Rajesh | 30|         8| 70000|
|  Meera |  0|         0| 50000|
| Sanjay | 38|        15| 80000|
|  Anita |  0|        10| 60000|
+--------+---+----------+------+
```

Filling missing values using Mean, Median, or Mode with help of the Imputer function

[28]:
```
#filling with mean
from pyspark.ml.feature import Imputer
imputer = Imputer(inputCols=["age"],outputCols=["age_imputed"]).setStrategy("mean")
imputer.fit(df_pyspark1).transform(df_pyspark1).show()
```

```
+--------+----+----------+------+-----------+
|    name| age|experience|salary|age_imputed|
+--------+----+----------+------+-----------+
|   Ravi |  28|         5| 60000|         28|
|  Priya |  35|      NULL| 75000|         35|
|   Arun |  40|        12| 85000|         40|
|   Neha |NULL|      NULL| 55000|         36|
| Suresh |  45|        18| 90000|         45|
|Deepika |NULL|      NULL| 65000|         36|
| Rajesh |  30|         8| 70000|         30|
|  Meera |NULL|      NULL| 50000|         36|
| Sanjay |  38|        15| 80000|         38|
|  Anita |NULL|        10| 60000|         36|
+--------+----+----------+------+-----------+
```

Filling missing values using Mean, Median, or Mode with help of the Imputer function

[28]:
```
#filling with mean
from pyspark.ml.feature import Imputer
imputer = Imputer(inputCols=["age"],outputCols=["age_imputed"]).setStrategy("mean")
imputer.fit(df_pyspark1).transform(df_pyspark1).show()
```

```
+--------+----+----------+------+-----------+
|    name| age|experience|salary|age_imputed|
+--------+----+----------+------+-----------+
|   Ravi |  28|         5| 60000|         28|
|  Priya |  35|      NULL| 75000|         35|
|   Arun |  40|        12| 85000|         40|
|   Neha |NULL|      NULL| 55000|         36|
| Suresh |  45|        18| 90000|         45|
|Deepika |NULL|      NULL| 65000|         36|
| Rajesh |  30|         8| 70000|         30|
|  Meera |NULL|      NULL| 50000|         36|
| Sanjay |  38|        15| 80000|         38|
|  Anita |NULL|        10| 60000|         36|
+--------+----+----------+------+-----------+
```

OrderBy() and sort() function in Pyspark

[29]:
```
df_pyspark.show()
```

```
+-------+-----------+------+
|   name|  department|salary|
+-------+-----------+------+
| Chandu|Data Science| 50000|
|Rashmi |         IOT| 75000|
|  Rohit|    Big Data| 55000|
| Rohit |    Big Data| 80000|
|  Ronit|         IOT| 60000|
|  Ronit|Data Science| 70000|
| Chandu|Data Science| 45000|
|Krishna|    Big Data| 65000|
| Rashmi|    Big Data| 85000|
|  Rohit|         IOT| 60000|
+-------+-----------+------+
```

[30]:
```
df_pyspark.sort("salary").show()
```

```
+-------+-----------+------+
|   name|  department|salary|
+-------+-----------+------+
| Chandu|Data Science| 45000|
| Chandu|Data Science| 50000|
|  Rohit|    Big Data| 55000|
|  Ronit|         IOT| 60000|
|  Rohit|         IOT| 60000|
|Krishna|    Big Data| 65000|
|  Ronit|Data Science| 70000|
|Rashmi |         IOT| 75000|
| Rohit |    Big Data| 80000|
| Rashmi|    Big Data| 85000|
+-------+-----------+------+
```

sort based on descending order

```
[31]: df_pyspark.sort(df_pyspark["salary"].desc()).show()
      # sort based on descending order
```

```
+-------+------------+------+
|   name|  department|salary|
+-------+------------+------+
| Rashmi|    Big Data| 85000|
| Rohit |    Big Data| 80000|
|Rashmi |          IOT| 75000|
|  Ronit|Data Science| 70000|
|Krishna|    Big Data| 65000|
|  Ronit|          IOT| 60000|
|  Rohit|          IOT| 60000|
|  Rohit|    Big Data| 55000|
| Chandu|Data Science| 50000|
| Chandu|Data Science| 45000|
+-------+------------+------+
```

Sort based on first column then second column

```
[32]: df_pyspark.sort("salary","Name").show()
      # Sort based on first column then second column
```

```
+-------+------------+------+
|   name|  department|salary|
+-------+------------+------+
| Chandu|Data Science| 45000|
| Chandu|Data Science| 50000|
|  Rohit|    Big Data| 55000|
|  Rohit|          IOT| 60000|
|  Ronit|          IOT| 60000|
|Krishna|    Big Data| 65000|
|  Ronit|Data Science| 70000|
|Rashmi |          IOT| 75000|
| Rohit |    Big Data| 80000|
| Rashmi|    Big Data| 85000|
+-------+------------+------+
```

Alternatively orderBy can be used to sort based on single column

```
[33]: #alternatively orderBy can be used to sort based on single column
      df_pyspark.orderBy("salary").show()
```

```
+-------+-----------+------+
|   name|  department|salary|
+-------+-----------+------+
| Chandu|Data Science| 45000|
| Chandu|Data Science| 50000|
|  Rohit|    Big Data| 55000|
|  Ronit|         IOT| 60000|
|  Rohit|         IOT| 60000|
|Krishna|    Big Data| 65000|
|  Ronit|Data Science| 70000|
|Rashmi |         IOT| 75000|
| Rohit |    Big Data| 80000|
| Rashmi|    Big Data| 85000|
+-------+-----------+------+
```

Joins

Join Type	Description
Inner Join	Join records when key columns are matched, and dropped when they are not matched
Outer join	Returns all rows from both datasets, where Join expression doesn't match it returns null or respective columns
Left Join/ Left outer join	Returns all rows from left dataset regardless of match found on right dataset, when Join doesn't match – it assigns null for that record
Right Join/ Right outer join	Returns all rows from Right dataset regardless of match found on left dataset, when Join doesn't match – it assigns null for that record
Left Semi Join	Returns columns from the only left dataset for the matched records in the right dataset on join expression
Left Anti Join	Returns only columns from left dataset for non-matched records

[35]:
```
#creating two dataframes for joins to be performed
emp = [(1,"Smith",-1,"2018","10","M",3000),(2, "Rose",1 , "2010", "20","M", 4000),(3,"Williams",1,"2010","10","M",1000),(4, "Jones",2 ,"2005"
empColumns = ["emp_id","name","superior_emp_id","year_joined", "emp_dept_id","gender","salary"]

empDF = spark.createDataFrame(data=emp, schema = empColumns)
empDF.printSchema()
empDF.show()

dept = [("Finance",10),("Marketing",20),("Sales",30),("IT",40)]
deptColumns = ["dept_name","dept_id"]
deptDF = spark.createDataFrame(data=dept, schema = deptColumns)
deptDF.printSchema()
deptDF.show()
```

```
+------+--------+---------------+-----------+-----------+------+------+
|emp_id|    name|superior_emp_id|year_joined|emp_dept_id|gender|salary|
+------+--------+---------------+-----------+-----------+------+------+
|     1|   Smith|             -1|       2018|         10|     M|  3000|
|     2|    Rose|              1|       2010|         20|     M|  4000|
|     3|Williams|              1|       2010|         10|     M|  1000|
|     4|   Jones|              2|       2005|         10|     F|  2000|
|     5|   Brown|              2|       2010|         40|      |    -1|
|     6|   Brown|              2|       2010|         50|      |    -1|
+------+--------+---------------+-----------+-----------+------+------+

root
 |-- dept_name: string (nullable = true)
 |-- dept_id: long (nullable = true)

+---------+-------+
|dept_name|dept_id|
+---------+-------+
|  Finance|     10|
|Marketing|     20|
|    Sales|     30|
|       IT|     40|
+---------+-------+
```

Inner join

:
```
#inner join
empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"inner").show()
```

```
[Stage 78:>                                                         (0 + 4) / 4]
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|emp_id|    name|superior_emp_id|year_joined|emp_dept_id|gender|salary|dept_name|dept_id|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|     1|   Smith|             -1|       2018|         10|     M|  3000|  Finance|     10|
|     3|Williams|              1|       2010|         10|     M|  1000|  Finance|     10|
|     4|   Jones|              2|       2005|         10|     F|  2000|  Finance|     10|
|     2|    Rose|              1|       2010|         20|     M|  4000|Marketing|     20|
|     5|   Brown|              2|       2010|         40|      |    -1|       IT|     40|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
```

Outer Join

```
#outer join
empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"outer").show()
```

```
[Stage 83:>                                                         (0 + 4) / 4]
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|emp_id|    name|superior_emp_id|year_joined|emp_dept_id|gender|salary|dept_name|dept_id|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|     1|   Smith|             -1|       2018|         10|     M|  3000|  Finance|     10|
|     3|Williams|              1|       2010|         10|     M|  1000|  Finance|     10|
|     4|   Jones|              2|       2005|         10|     F|  2000|  Finance|     10|
|     2|    Rose|              1|       2010|         20|     M|  4000|Marketing|     20|
|  NULL|    NULL|           NULL|       NULL|       NULL|  NULL|  NULL|    Sales|     30|
|     5|   Brown|              2|       2010|         40|      |    -1|       IT|     40|
|     6|   Brown|              2|       2010|         50|      |    -1|     NULL|   NULL|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
```

Left join

```
[39]: #left join
empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"left").show()
```

```
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|emp_id|    name|superior_emp_id|year_joined|emp_dept_id|gender|salary|dept_name|dept_id|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|     1|   Smith|             -1|       2018|         10|     M|  3000|  Finance|     10|
|     3|Williams|              1|       2010|         10|     M|  1000|  Finance|     10|
|     2|    Rose|              1|       2010|         20|     M|  4000|Marketing|     20|
|     4|   Jones|              2|       2005|         10|     F|  2000|  Finance|     10|
|     6|   Brown|              2|       2010|         50|      |    -1|     NULL|   NULL|
|     5|   Brown|              2|       2010|         40|      |    -1|       IT|     40|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
```

Right join

```
[40]: #right join
empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"right").show()
```

```
[Stage 94:>                                                         (0 + 4) / 4]
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|emp_id|    name|superior_emp_id|year_joined|emp_dept_id|gender|salary|dept_name|dept_id|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|     4|   Jones|              2|       2005|         10|     F|  2000|  Finance|     10|
|     3|Williams|              1|       2010|         10|     M|  1000|  Finance|     10|
|     1|   Smith|             -1|       2018|         10|     M|  3000|  Finance|     10|
|     2|    Rose|              1|       2010|         20|     M|  4000|Marketing|     20|
|  NULL|    NULL|           NULL|       NULL|       NULL|  NULL|  NULL|    Sales|     30|
|     5|   Brown|              2|       2010|         40|      |    -1|       IT|     40|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
```

Left semi join

[41]:
```
#left semi join

empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"leftsemi").show()
```

```
[Stage 100:>                                                        (0 + 4) / 4]
+------+--------+---------------+-----------+-----------+------+------+
|emp_id|    name|superior_emp_id|year_joined|emp_dept_id|gender|salary|
+------+--------+---------------+-----------+-----------+------+------+
|     1|   Smith|             -1|       2018|         10|     M|  3000|
|     3|Williams|              1|       2010|         10|     M|  1000|
|     4|   Jones|              2|       2005|         10|     F|  2000|
|     2|    Rose|              1|       2010|         20|     M|  4000|
|     5|   Brown|              2|       2010|         40|      |    -1|
+------+--------+---------------+-----------+-----------+------+------+
```

Left anti join

[42]:
```
#left anti join
empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"leftanti").show()
```

```
+------+-----+---------------+-----------+-----------+------+------+
|emp_id| name|superior_emp_id|year_joined|emp_dept_id|gender|salary|
+------+-----+---------------+-----------+-----------+------+------+
|     6|Brown|              2|       2010|         50|      |    -1|
+------+-----+---------------+-----------+-----------+------+------+
```

Union

PySpark union() and unionAll() transformations are used to merge two or more DataFrame's of the same schema or structure.

[43]:
```
import pyspark
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('SparkByExamples.com').getOrCreate()
simpleData = [("James","Sales","NY",90000,34,10000), \
("Michael","Sales","NY",86000,56,20000), \
("Robert","Sales","CA",81000,30,23000), \
("Maria","Finance","CA",90000,24,23000) \
]
columns= ["employee_name","department","state","salary","age","bonus"]
df = spark.createDataFrame(data = simpleData, schema = columns)
df.printSchema()
df.show(truncate=False)
```

```
24/02/09 17:20:46 WARN SparkSession: Using an existing Spark session; only runtime SQL configurations will take effect.
root
 |-- employee_name: string (nullable = true)
 |-- department: string (nullable = true)
 |-- state: string (nullable = true)
 |-- salary: long (nullable = true)
 |-- age: long (nullable = true)
 |-- bonus: long (nullable = true)

+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|James        |Sales     |NY   |90000 |34 |10000|
|Michael      |Sales     |NY   |86000 |56 |20000|
|Robert       |Sales     |CA   |81000 |30 |23000|
|Maria        |Finance   |CA   |90000 |24 |23000|
+-------------+----------+-----+------+---+-----+
```

[44]:
```
simpleData2 = [("James","Sales","NY",90000,34,10000), \
("Maria","Finance","CA",90000,24,23000), \
("Jen","Finance","NY",79000,53,15000), \
("Jeff","Marketing","CA",80000,25,18000), \
("Kumar","Marketing","NY",91000,50,21000) \
]
columns2= ["employee_name","department","state","salary","age","bonus"]
df2 = spark.createDataFrame(data = simpleData2, schema = columns2)
df2.printSchema()
df2.show(truncate=False)
```

```
root
 |-- employee_name: string (nullable = true)
 |-- department: string (nullable = true)
 |-- state: string (nullable = true)
 |-- salary: long (nullable = true)
 |-- age: long (nullable = true)
 |-- bonus: long (nullable = true)

+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|James        |Sales     |NY   |90000 |34 |10000|
|Maria        |Finance   |CA   |90000 |24 |23000|
|Jen          |Finance   |NY   |79000 |53 |15000|
|Jeff         |Marketing |CA   |80000 |25 |18000|
|Kumar        |Marketing |NY   |91000 |50 |21000|
+-------------+----------+-----+------+---+-----+
```

merge two or more dataframes using union

:
```
#merge two or more dataframes using union
unionDF = df.union(df2)
unionDF.show(truncate=False)
```

```
+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|James        |Sales     |NY   |90000 |34 |10000|
|Michael      |Sales     |NY   |86000 |56 |20000|
|Robert       |Sales     |CA   |81000 |30 |23000|
|Maria        |Finance   |CA   |90000 |24 |23000|
|James        |Sales     |NY   |90000 |34 |10000|
|Maria        |Finance   |CA   |90000 |24 |23000|
|Jen          |Finance   |NY   |79000 |53 |15000|
|Jeff         |Marketing |CA   |80000 |25 |18000|
|Kumar        |Marketing |NY   |91000 |50 |21000|
+-------------+----------+-----+------+---+-----+
```

Merge without duplicates

```
#merge without duplicates
disDF = df.union(df2).distinct()
disDF.show(truncate=False)
```

```
[Stage 13:==============================>
+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|James        |Sales     |NY   |90000 |34 |10000|
|Michael      |Sales     |NY   |86000 |56 |20000|
|Robert       |Sales     |CA   |81000 |30 |23000|
|Maria        |Finance   |CA   |90000 |24 |23000|
|Jen          |Finance   |NY   |79000 |53 |15000|
|Kumar        |Marketing |NY   |91000 |50 |21000|
|Jeff         |Marketing |CA   |80000 |25 |18000|
+-------------+----------+-----+------+---+-----+
```