EXPERIMENT 3

AIM: PySpark program to perform inner join

THEORY:

PySpark Join is used to combine two DataFrames and by chaining these you can join multiple DataFrames; it supports all basic join type operations available in traditional SQL like INNER, LEFT OUTER, RIGHT OUTER, LEFT ANTI, LEFT SEMI, CROSS, SELF JOIN. PySpark Joins are wider transformations that involve data shuffling across the network.

INNER JOIN: The INNER JOIN keyword selects all rows from both the tables as long as the condition satisfies. This keyword will create the result-set by combining all rows from both the tables where the condition satisfies i.e value of the common field will be same.

LEFT JOIN: This join returns all the rows of the table on the left side of the join and matching rows for the table on the right side of join. The rows for which there is no matching row on right side, the result-set will contain null. LEFT JOIN is also known as LEFT OUTER JOIN

RIGHT JOIN: RIGHT JOIN is similar to LEFT JOIN. This join returns all the rows of the table on the right side of the join and matching rows for the table on the left side of join. The rows for which there is no matching row on left side, the result-set will contain null. RIGHT JOIN is also known as RIGHT OUTER JOIN

FULL JOIN: FULL JOIN creates the result-set by combining result of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from both the tables. The rows for which there is no matching, the result-set will contain NULL values

CODE:

```
from pyspark import SparkConf, SparkContext

from pyspark.sql import SparkSession

from pyspark.sql.functions import col
```

```
conf = SparkConf()
conf.setMaster("local").setAppName("Joins")
sc = SparkContext.getOrCreate(conf=conf)
spark = SparkSession(sc)
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","10","F",2000), \
   (5,"Brown",2,"2010","40","",-1), \
     (6,"Brown",2,"2010","50","",-1) \
 ]
empColumns = ["emp_id","name","superior_emp_id","year_joined", \
      "emp_dept_id","gender","salary"]

empDF = spark.createDataFrame(data=emp, schema = empColumns)
dept = [("Finance",10), \
   ("Marketing",20), \
   ("Sales",30), \
   ("IT",40) \
 ]
deptColumns = ["dept_name","dept_id"]
deptDF = spark.createDataFrame(data=dept, schema = deptColumns)
#Inner Join
empDF.join(deptDF,empDF.emp_dept_id == deptDF.dept_id,"inner").show()
```

OUTPUT:

Employee Table

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

DEPARTMENT TABLE

dept_name	dept_id
Finance	10
Marketing	20
Sales	30
IT	40

Inner Join Output

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

CONCLUSION:

We have created two tables named Employee and Department using createDataFrame function, and performed inner join using join function in pyspark.