In [1]:
```
# PySpark Join is used to combine two DataFrames and by chaining these you can join multiple DataFrames; it supp
```

In [2]:
```
# PySpark Joins are wider transformations that involve data shuffling across the network.
```

In [3]:
```
# PySpark SQL Joins comes with more optimizations by default (thanks to DataFrames), but there are still some pe

# Understanding how to effectively utilize PySpark joins is essential for conducting comprehensive data analysis

# In this PySpark SQL Join, you will learn different Join syntaxes and use different Join types on two or more D

# PySpark Join Syntax
# PySpark Join Types
# Inner Join DataFrame
# Full Outer Join DataFrame
# Left Outer Join DataFrame
# Right Outer Join DataFrame
# Left Anti Join DataFrame
# Left Semi Join DataFrame
# Self Join DataFrame
# Using SQL Expression
```

1. PySpark Join Syntax

PySpark SQL join has a below syntax and it can be accessed directly from DataFrame.

In [4]:
```
# Syntax
# join(self, other, on=None, how=None)
```

In [5]:
```
# join() operation takes parameters as below and returns DataFrame.

# param other: Right side of the join
# param on: a string for the join column name
# param how: default inner. Must be one of inner, cross, outer,full, full_outer, left, left_outer, right, right_
```

In [6]:
```
# You can also write Join expression by adding where() and filter() methods on DataFrame and can have Join on mu
```

2. PySpark Join Types

In [7]:
```
# Below are the different Join Types PySpark supports.

# Join          String  Equivalent SQL Join
# inner       INNER JOIN
# outer,        full, fullouter, full_outer     FULL OUTER JOIN
# left,         leftouter, left_outer   LEFT JOIN
# right,        rightouter, right_outer RIGHT JOIN
# cross
# anti,         leftanti, left_anti
# semi,         leftsemi, left_semi
```

In [8]:
```
# Before diving into PySpark SQL Join illustrations, let’s initiate “emp” and “dept” DataFrames.
# The emp DataFrame contains the “emp_id” column with unique values, while the dept DataFrame contains the “dep
# Additionally, the “emp_dept_id” from “emp” refers to the “dept_id” in the “dept” dataset.
```

In [9]:
```
import findspark
findspark.init()
```

In [10]:
```
# Prepare Data
import pyspark
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName('Joins').master('local[*]').getOrCreate()

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)
empDF.printSchema()
empDF.show(truncate = False)
```

```
root
 |-- emp_id: long (nullable = true)
 |-- name: string (nullable = true)
 |-- superior_emp_id: long (nullable = true)
 |-- year_joined: string (nullable = true)
 |-- emp_dept_id: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- salary: long (nullable = true)

+------+--------+---------------+-----------+-----------+------+------+
|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    |
+------+--------+---------------+-----------+-----------+------+------+
```

In [11]:

```
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(truncate = False)
```

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

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

3. How Join Works?

In [12]:

```
# PySpark’s join operation combines data from two or more Datasets based on a common column or key.
# It is a fundamental operation in PySpark and is similar to SQL joins.

# Common Key:
# =============

# In order to join two or more datasets we need a common key or a column on which you want to join. This key is

# Partitioning:
# ===============

# PySpark Datasets are distributed and partitioned across multiple nodes in a cluster.
# Ideally, data with the same join key should be located in the same partition.
# If the Datasets are not already partitioned on the join key, PySpark may perform a shuffle operation to redis
# Shuffling can be an expensive operation, especially for large Datasets.

# Join Type Specification:
# ==========================

# We can specify the type of join like inner join, full join, left join, etc., by specifying on “how” parameter

# This parameter determines which rows should be included or excluded in the resulting Dataset.

# Join Execution:
# =================

# PySpark performs the join by comparing the values in the common key column between the Datasets.

# Inner Join:
# =============

# Returns only the rows with matching keys in both DataFrames.
```

```
# Left Join:
# ============

# Returns all rows from the left DataFrame and matching rows from the right DataFrame.

# Right Join:
# =============

# Returns all rows from the right DataFrame and matching rows from the left DataFrame.

# Full Outer Join:
# ==================

# Returns all rows from both DataFrames, including matching and non-matching rows.

# Left Semi Join:
# =================

# Returns all rows from the left DataFrame where there is a match in the right DataFrame.

# Left Anti Join:
# =================

# Returns all rows from the left DataFrame where there is no match in the right DataFrame.
```

4. PySpark Inner Join DataFrame

The default join in PySpark is the inner join, commonly used to retrieve data from two or more DataFrames based on a shared key. An Inner join combines two DataFrames based on the key (common column) provided and results in rows where there is a matching found. Rows from both DataFrames are dropped with a non-matching key.

In [13]:
```
# Inner Join
empDF.join(deptDF, empDF.emp_dept_id == deptDF.dept_id, "inner") \
    .show(truncate = False)
```

```
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|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     |
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
```

This example drops “emp_dept_id” with value 50 from “emp” And “dept_id” with value 30 from “dept” datasets. Following is the result of the above Join statement.

5. PySpark Left Outer Join

Left a.k.a Leftouter join returns all rows from the left dataset regardless of match found on the right dataset when join expression doesn’t match, it assigns null for that record and drops records from right where match not found.

In [14]:
```
# Left Outer Join
empDF.join(deptDF, empDF.emp_dept_id == deptDF.dept_id, "left") \
    .show(truncate = False)

empDF.join(deptDF, empDF.emp_dept_id == deptDF.dept_id, "leftouter") \
    .show(truncate = False)
```

```
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|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     |
|2     |Rose    |1              |2010       |20         |M     |4000  |Marketing|20     |
|3     |Williams|1              |2010       |10         |M     |1000  |Finance  |10     |
|4     |Jones   |2              |2005       |10         |F     |2000  |Finance  |10     |
|5     |Brown   |2              |2010       |40         |      |-1    |IT       |40     |
|6     |Brown   |2              |2010       |50         |      |-1    |NULL     |NULL   |
+------+--------+---------------+-----------+-----------+------+------+---------+-------+

+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|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     |
|2     |Rose    |1              |2010       |20         |M     |4000  |Marketing|20     |
|3     |Williams|1              |2010       |10         |M     |1000  |Finance  |10     |
|4     |Jones   |2              |2005       |10         |F     |2000  |Finance  |10     |
|5     |Brown   |2              |2010       |40         |      |-1    |IT       |40     |
|6     |Brown   |2              |2010       |50         |      |-1    |NULL     |NULL   |
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
```

In our dataset, the record with “emp_dept_id” 50 does not have a corresponding entry in the “dept” dataset, resulting in null values in the “dept” columns (dept_name & dept_id). Additionally, the entry with “dept_id” 30 from the “dept” dataset is omitted from the results. Above is the outcome of the provided join expression.

6. Right Outer Join

Right a.k.a Rightouter join is opposite of left join, here it returns all rows from the right dataset regardless of math found on the left dataset, when join expression doesn’t match, it assigns null for that record and drops records from left where match not found.

In [15]:
```
# Right Outer Join
empDF.join(deptDF, empDF.emp_dept_id == deptDF.dept_id, "right") \
    .show(truncate = False)

empDF.join(deptDF, empDF.emp_dept_id == deptDF.dept_id, "rightouter") \
    .show(truncate = False)
```

```
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|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     |
+------+--------+---------------+-----------+-----------+------+------+---------+-------+

+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|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     |
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
```

In our example, the dataset on the right, containing “dept_id” with a value of 30, does not have a corresponding record in the left dataset “emp”. Consequently, this record contains null values for the columns from “emp”. Additionally, the record with “emp_dept_id” value 50 is dropped as no match was found in the left dataset. Below is the result of the aforementioned join expression.

7. PySpark Full Outer Join

Outer a.k.a full, fullouter join in PySpark combines the results of both left and right outer joins, ensuring that all records from both DataFrames are included in the resulting DataFrame. It includes all rows from both DataFrames and fills in missing values with nulls where there is no match. In other words, it merges the DataFrames based on a common key, but retains all rows from both DataFrames, even if there’s no match. This join type is useful when you want to preserve all the information from both datasets, regardless of whether there’s a match on the key or not.

In [16]:
```
# Full Outer Join
empDF.join(deptDF, empDF.emp_dept_id == deptDF.dept_id, "outer") \
    .show(truncate = False)

empDF.join(deptDF, empDF.emp_dept_id == deptDF.dept_id, "full") \
    .show(truncate = False)

empDF.join(deptDF, empDF.emp_dept_id == deptDF.dept_id, "fullouter") \
    .show(truncate = False)
```

```
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|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   |
+------+--------+---------------+-----------+-----------+------+------+---------+-------+

+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|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   |
+------+--------+---------------+-----------+-----------+------+------+---------+-------+

+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|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   |
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
```

This code snippet performs a full outer join between two PySpark DataFrames, empDF and deptDF, based on the condition that emp_dept_id from empDF is equal to dept_id from deptDF. In our “emp” dataset, the “emp_dept_id” with a value of 50 does not have a corresponding record in the “dept” dataset, resulting in null values in the “dept” columns. Similarly, the “dept_id” 30 does not have a record in the “emp” dataset, hence you observe null values in the “emp” columns. Above Code is the output of the provided join example.

8. Left Semi Join

A Left Semi Join in PySpark returns only the rows from the left DataFrame (the first DataFrame mentioned in the join operation) where there is a match with the right DataFrame (the second DataFrame). It does not include any columns from the right DataFrame in the resulting DataFrame. This join type is useful when you only want to filter rows from the left DataFrame based on whether they have a matching key in the right DataFrame. Left Semi Join can also be achieved by selecting only the columns from the left dataset from the result of the inner join

In [17]:
```
# Left Semi Join
empDF.join(deptDF, empDF.emp_dept_id == deptDF.dept_id, "leftsemi") \
    .show(truncate = False)
```

```
+------+--------+---------------+-----------+-----------+------+------+
|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    |
+------+--------+---------------+-----------+-----------+------+------+
```

9. Left Anti Join

A Left Anti Join in PySpark returns only the rows from the left DataFrame (the first DataFrame mentioned in the join operation) where there is no match with the right DataFrame (the second DataFrame). It excludes any rows from the left DataFrame that have a corresponding key in the right DataFrame. This join type is useful when you want to filter out rows from the left DataFrame that have matching keys in the right DataFrame.

In [18]:
```
# Left Anti Join
empDF.join(deptDF, empDF.emp_dept_id == deptDF.dept_id, "leftanti") \
    .show(truncate = False)
```

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

10. PySpark Self Join

Joins are not complete without a self join, Though there is no self-join type available in PySpark, we can use any of the above-explained join types to join DataFrame to itself. below example use inner self join.

In [19]:
```python
# Self Join

from pyspark.sql.functions import col

empDF.alias("emp1").join(empDF.alias("emp2"), \
    col("emp1.superior_emp_id") == col("emp2.emp_id"), "inner") \
    .select(col("emp1.emp_id"), col("emp1.name"), \
    col("emp2.emp_id").alias("superior_emp_id"), \
    col("emp2.name").alias("superior_emp_name")) \
    .show(truncate = False)
```

```
+------+--------+---------------+-----------------+
|emp_id|name    |superior_emp_id|superior_emp_name|
+------+--------+---------------+-----------------+
|2     |Rose    |1              |Smith            |
|3     |Williams|1              |Smith            |
|4     |Jones   |2              |Rose             |
|5     |Brown   |2              |Rose             |
|6     |Brown   |2              |Rose             |
+------+--------+---------------+-----------------+
```

11. Using SQL Expression

Alternatively, you can also use SQL query to join DataFrames/tables in PySpark. To do so, first, create a temporary view using createOrReplaceTempView(), then use the spark.sql() to execute the join query.

In [20]:
```python
# Using spark.sql

empDF.createOrReplaceTempView("EMP")
deptDF.createOrReplaceTempView("DEPT")

joinDF = spark.sql("SELECT * FROM EMP e, DEPT d WHERE e.emp_dept_id == d.dept_id") \
    .show(truncate = False)

joinDF2 = spark.sql("SELECT * FROM EMP e INNER JOIN DEPT d ON e.emp_dept_id == d.dept_id") \
    .show(truncate = False)
```

```
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|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     |
+------+--------+---------------+-----------+-----------+------+------+---------+-------+

+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|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     |
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
```

12. PySpark SQL Join on multiple DataFrames

When you need to join more than two tables, you either use SQL expression after creating a temporary view on the DataFrame or use the result of join operation to join with another DataFrame like chaining them. for example# Join on multiple dataframes df1.join(df2, df1.id1 == df2.id2, "inner") \ .join(df2, df1.id1 == df3.id3, "inner")

13. PySpark SQL Join Complete Example

In [21]:
```python
import pyspark

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

spark = SparkSession.builder.appName('Join').master('local[*]').getOrCreate()

emp = [(1,"Smith",-1,"2018","10","M",3000), \
```

```python
    (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)
empDF.printSchema()
empDF.show(truncate=False)


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(truncate=False)

empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"inner") \
     .show(truncate=False)

empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"outer") \
    .show(truncate=False)
empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"full") \
    .show(truncate=False)
empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"fullouter") \
    .show(truncate=False)

empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"left") \
    .show(truncate=False)
empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"leftouter") \
   .show(truncate=False)

empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"right") \
   .show(truncate=False)
empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"rightouter") \
   .show(truncate=False)

empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"leftsemi") \
   .show(truncate=False)

empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"leftanti") \
   .show(truncate=False)

empDF.alias("emp1").join(empDF.alias("emp2"), \
    col("emp1.superior_emp_id") == col("emp2.emp_id"),"inner") \
    .select(col("emp1.emp_id"),col("emp1.name"), \
      col("emp2.emp_id").alias("superior_emp_id"), \
      col("emp2.name").alias("superior_emp_name")) \
   .show(truncate=False)

empDF.createOrReplaceTempView("EMP")
deptDF.createOrReplaceTempView("DEPT")

joinDF = spark.sql("select * from EMP e, DEPT d where e.emp_dept_id == d.dept_id") \
  .show(truncate=False)

joinDF2 = spark.sql("select * from EMP e INNER JOIN DEPT d ON e.emp_dept_id == d.dept_id") \
  .show(truncate=False)
```

```
root
 |-- emp_id: long (nullable = true)
 |-- name: string (nullable = true)
 |-- superior_emp_id: long (nullable = true)
 |-- year_joined: string (nullable = true)
 |-- emp_dept_id: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- salary: long (nullable = true)

+------+--------+---------------+-----------+-----------+------+------+
|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     |
+---------+-------+

+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|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     |
+------+--------+---------------+-----------+-----------+------+------+---------+-------+

+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|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   |
+------+--------+---------------+-----------+-----------+------+------+---------+-------+

+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|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   |
+------+--------+---------------+-----------+-----------+------+------+---------+-------+

+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|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   |
+------+--------+---------------+-----------+-----------+------+------+---------+-------+

+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|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     |
|2     |Rose    |1              |2010       |20         |M     |4000  |Marketing|20     |
|3     |Williams|1              |2010       |10         |M     |1000  |Finance  |10     |
|4     |Jones   |2              |2005       |10         |F     |2000  |Finance  |10     |
|5     |Brown   |2              |2010       |40         |      |-1    |IT       |40     |
|6     |Brown   |2              |2010       |50         |      |-1    |NULL     |NULL   |
+------+--------+---------------+-----------+-----------+------+------+---------+-------+

+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|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     |
|2     |Rose    |1              |2010       |20         |M     |4000  |Marketing|20     |
|3     |Williams|1              |2010       |10         |M     |1000  |Finance  |10     |
|4     |Jones   |2              |2005       |10         |F     |2000  |Finance  |10     |
|5     |Brown   |2              |2010       |40         |      |-1    |IT       |40     |
|6     |Brown   |2              |2010       |50         |      |-1    |NULL     |NULL   |
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
```

```
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|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     |
+------+--------+---------------+-----------+-----------+------+------+---------+-------+

+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|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     |
+------+--------+---------------+-----------+-----------+------+------+---------+-------+

+------+--------+---------------+-----------+-----------+------+------+
|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    |
+------+--------+---------------+-----------+-----------+------+------+

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

+------+--------+---------------+-----------------+
|emp_id|name    |superior_emp_id|superior_emp_name|
+------+--------+---------------+-----------------+
|2     |Rose    |1              |Smith            |
|3     |Williams|1              |Smith            |
|4     |Jones   |2              |Rose             |
|5     |Brown   |2              |Rose             |
|6     |Brown   |2              |Rose             |
+------+--------+---------------+-----------------+

+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|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     |
+------+--------+---------------+-----------+-----------+------+------+---------+-------+

+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|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     |
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
```

14. Frequently asked questions on PySpark Joins

What is the default join in PySpark?

In PySpark the default join type is “inner” join when using with .join() method. If you don’t explicitly specify the join type using the “how” parameter, it will perform the inner join. One can change the join type using the how parameter of .join().

Is join expensive in PySpark?

Yes, Join in PySpark is expensive because of the data shuffling (wider transformation) that happens between the partitioned data in a cluster. It basically depends on the data size, data skew, cluster configuration, join type being performed, partitioning, and broadcast joins.

Can we join on multiple columns in PySpark?

Yes, we can join on multiple columns. Joining on multiple columns involves more join conditions with multiple keys for matching the rows between the datasets. It can be achieved by passing a list of column names as the join condition when using the .join() method.

How do I drop duplicate columns after joining PySpark?

PySpark distinct() function is used to drop/remove the duplicate rows (all columns) from Dataset and dropDuplicates() is used to drop rows based on selected (one or multiple) columns.

What is the difference between the inner join and the left join?

The key difference is that an inner join includes only the rows with matching keys in both Datasets, while a left join includes all the rows from the left Dataset and matches them with rows from the right Dataset where there's a match. Non-matching rows in the left Dataset in a left join are included with null values in the columns from the right Dataset.

What is the difference between left join and left outer join?

Both terms refer to the same type of join operation, and they can be used interchangeably. The "OUTER" keyword is optional when specifying a "LEFT JOIN."