**JOINS**

**Type of Joins**

1. Inner Join
2. Outer (Full) Join
3. Left Join
4. Right Join
5. Left Semi Join
6. Left Anti Join
7. Cross Join

**Creating Dataframe:**

**A screenshot of a computer code

AI-generated content may be incorrect.**

**Inner Join**

An inner join returns rows from both dataframes that have matching keys. In other words, it returns only the rows that have common keys in both dataframes. This is the default join type in PySpark.

A screenshot of a computer

AI-generated content may be incorrect.

**Outer (Full) Join**

An outer join, also known as a full join, returns all rows from both dataframes. If a key is present in one dataframe but not in the other, the missing values are filled with nulls.

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Left Join**

A left join returns all rows from the left dataframe and the matched rows from the right dataframe. If no match is found for a key in the right dataframe, the result will contain null values.

A screenshot of a computer program

AI-generated content may be incorrect.

**Right Join**

A right join returns all rows from the right dataframe and the matched rows from the left dataframe. If no match is found for a key in the left dataframe, the result will contain null values.

A screenshot of a computer

AI-generated content may be incorrect.

**Left Semi Join**

A left semi join returns only the columns from the left dataframe for the rows with matching keys in both dataframes. It is similar to an inner join but only returns the columns from the left dataframe.

A screenshot of a computer

AI-generated content may be incorrect.

**Left Anti Join**

A left anti join returns the rows from the left dataframe that do not have matching keys in the right dataframe. It is the opposite of a left semi join.

A screenshot of a computer

AI-generated content may be incorrect.

**Cross Join**

A cross join, also known as a cartesian join, returns the cartesian product of both dataframes. It combines each row from the left dataframe with each row from the right dataframe.

A screenshot of a computer

AI-generated content may be incorrect.

**overwriteSchema:**

overwriteSchema is used when overwriting a Delta table with a new schema. It replaces the entire schema of the existing table with the new schema from the DataFrame.

A computer code with red text

AI-generated content may be incorrect.

* Replaces the existing schema entirely.
* Useful when you want to reset the table schema.

**mergeSchema in Delta Table:**

mergeSchema is used when appending data with a modified schema. It merges new columns into the existing schema without dropping old ones.

A screenshot of a computer code

AI-generated content may be incorrect.

* Adds new columns if they don't exist in the existing schema.
* Keeps old schema intact (no deletion or modification of existing columns).
* Ensures schema compatibility for appending new data.

**1.2 How Spark Distributes Work**

1️. **Data is divided into partitions** and distributed across nodes.  
2️. The **Driver program** schedules tasks to process partitions in parallel.  
3️. **Executors** run tasks and store intermediate results in memory (RDDs).  
4️. The results are collected and returned to the Driver.