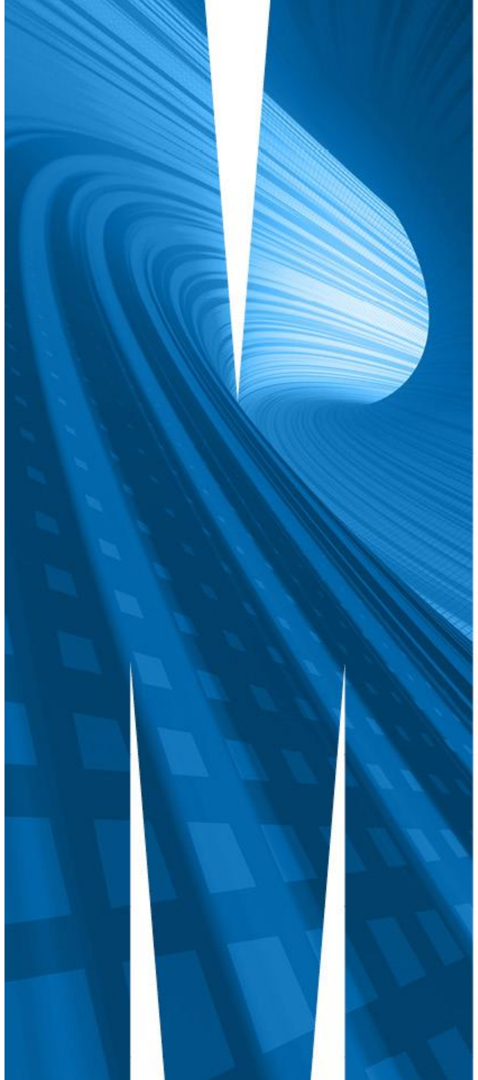
MONASH University

Week 3

FIT5202 Big Data Processing



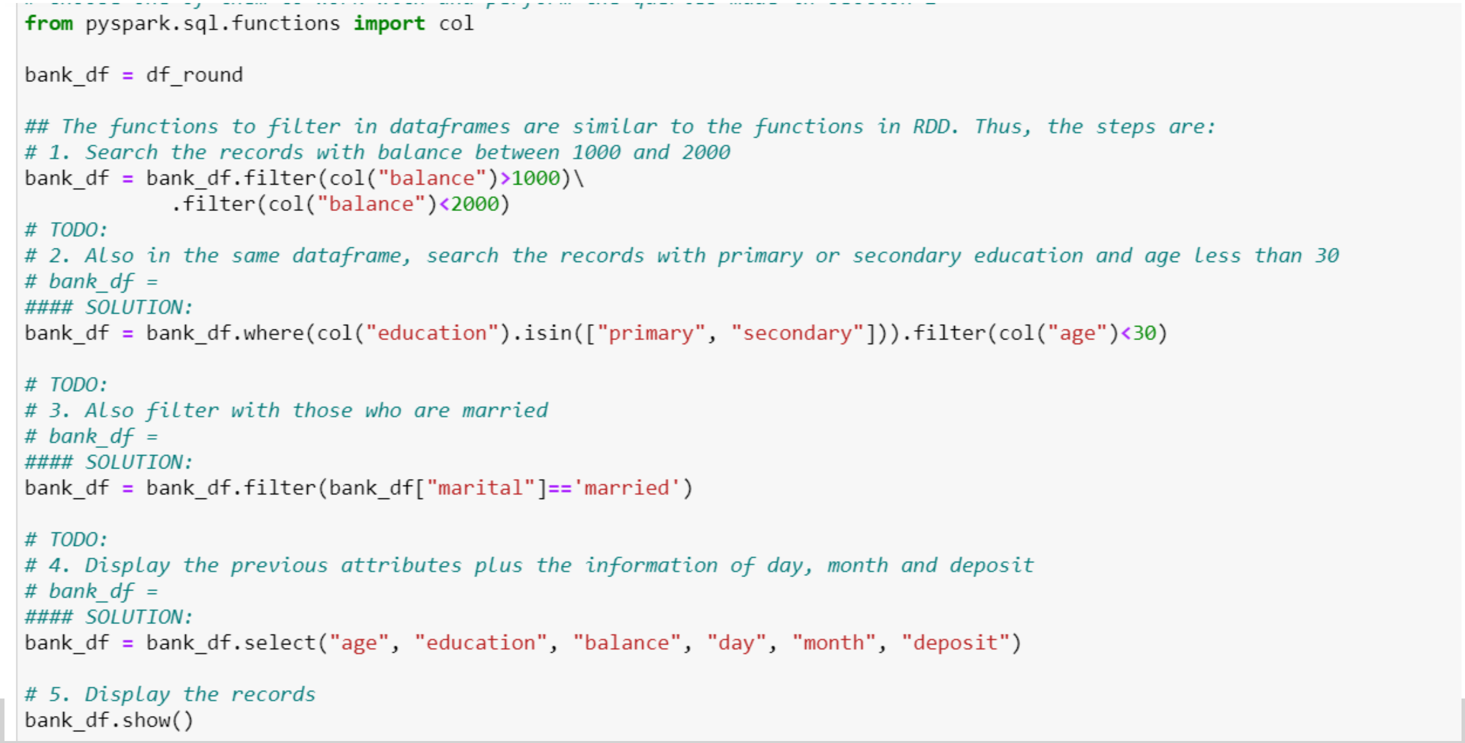
Week 3 Agenda

- Week 2 Review
- Assignment 1 Briefing
- Join Strategies in Spark
- Joins used in Spark
- Understanding Query Execution Plan and DAG in Spark UI (Demo)

Week 2 Review

- Understanding SparkSession and SparkContext
- Data Partitioning in RDDs
- Parallel Search and RDDs with multiple conditions
- RDDs vs DataFrames
- Partitioning in DataFrames
- Searching and Filtering in DataFrames
- Using SparkSQL With DataFrames.

Solution – Week 2



```python
from pyspark.sql.functions import col

bank_df = df_round

## The functions to filter in dataframes are similar to the functions in RDD. Thus, the steps are:
# 1. Search the records with balance between 1000 and 2000
bank_df = bank_df.filter(col("balance")>1000)\
            .filter(col("balance")<2000)
# TODO:
# 2. Also in the same dataframe, search the records with primary or secondary education and age less than 30
# bank_df =
#### SOLUTION:
bank_df = bank_df.where(col("education").isin(["primary", "secondary"])).filter(col("age")<30)

# TODO:
# 3. Also filter with those who are married
# bank_df =
#### SOLUTION:
bank_df = bank_df.filter(bank_df["marital"]=='married')

# TODO:
# 4. Display the previous attributes plus the information of day, month and deposit
# bank_df =
#### SOLUTION:
bank_df = bank_df.select("age", "education", "balance", "day", "month", "deposit")

# 5. Display the records
bank_df.show()
```

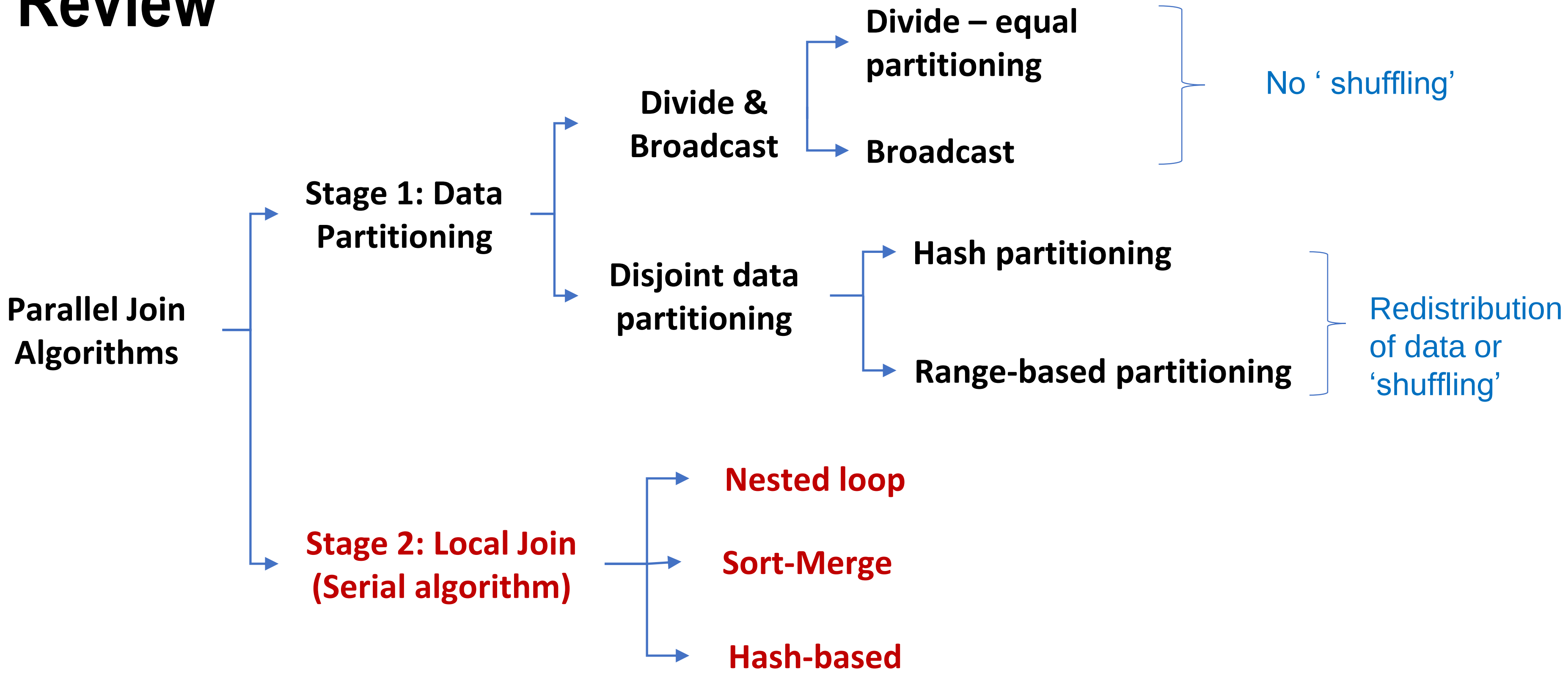
Joining Strategies By Spark

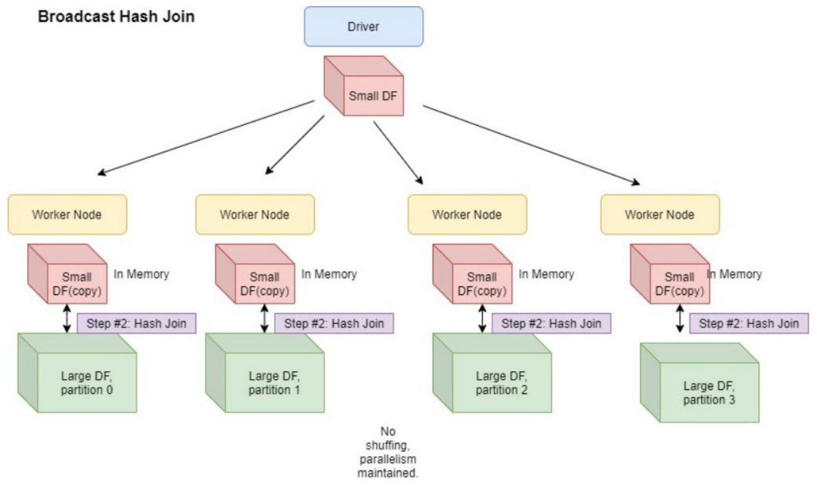
Why do we need different join strategy?

1. Shuffle Sort Merge Join (default)
2. Broadcast Hash Join
3. Shuffle Hash Join

https://towardsdatascience.com/strategies-of-spark-join-c0e7b4572bcf

Review


Parallel Join Algorithms
Stage 1: Data Partitioning
Divide & Broadcast
Divide – equal partitioning
Broadcast
No ‘ shuffling’
Disjoint data partitioning
Hash partitioning
Range-based partitioning
Redistribution of data or ‘shuffling’
Stage 2: Local Join (Serial algorithm)
Nested loop
Sort-Merge
Hash-based



Broadcast Hash Join

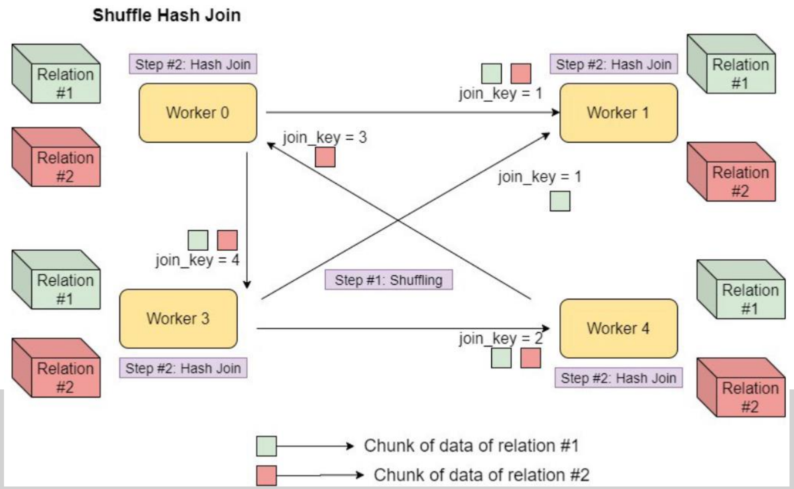
(1) Partitioning: large table is partitioned, broadcast small table to each node (processor)

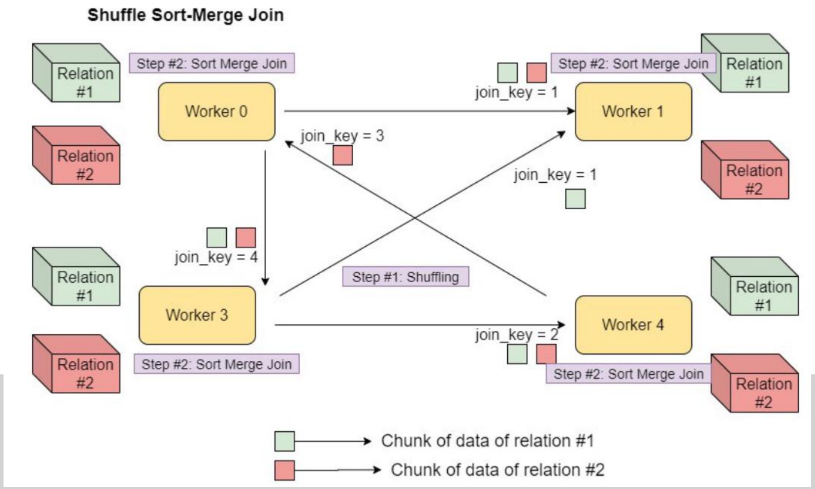
(2) Local join (hash join)

Shuffle Hash (Sort-Merge) Join

(1) Redistribute data until data in each partition have the same join-key value (e.g., hash value)

(2) Local join (hash join/sort-merge)



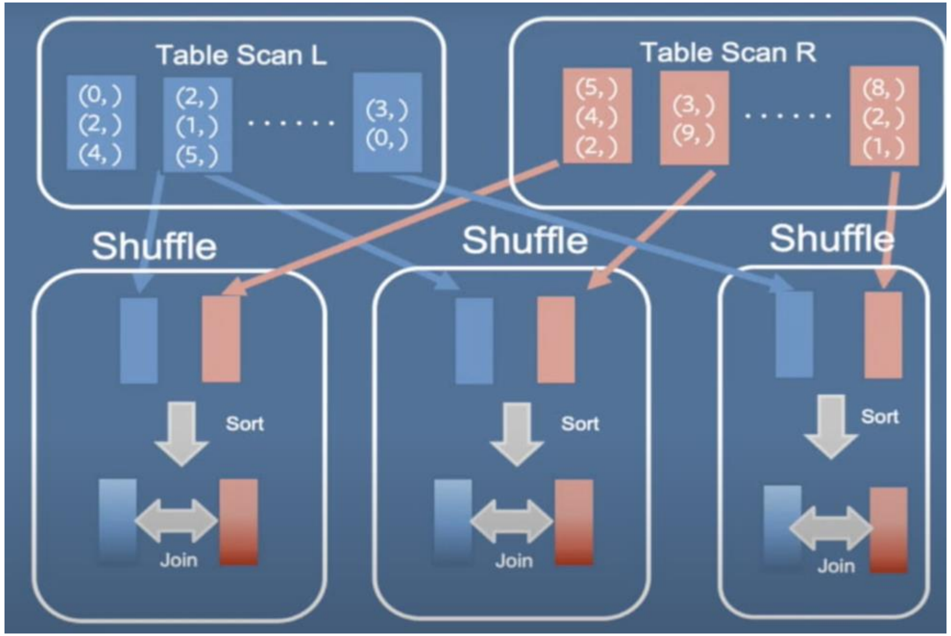


Join Strategies in Spark

Sort-Merge Join

1. Data from both tables are shuffled (when no bucketing was done)
2. Data are then sorted based on the key and perform sort-merge join on each executer

- Most Scalable joining strategy in Spark
- spark.sql.join.preferSortMergeJoin is set to true by default

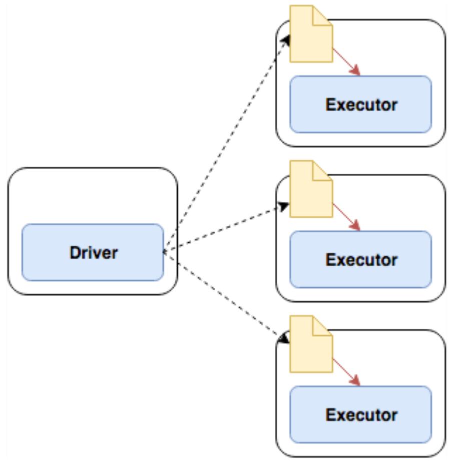


https://databricks.com/session_eu19/spark-sql-bucketing-at-facebook

Join Strategies in Spark

Broadcast Hash

1. Smallest table is broadcasted to each node (using broadcast variable)
2. Joining then takes place
3. Eliminates the shuffling process for bigger dataset.
4. Threshold of size of smaller table can be Configured using "spark.sql.autoBroadcastJoinThreshold"
5. Default is 10 MB (if one of the tables is less than 10 MB, that can be broadcasted)

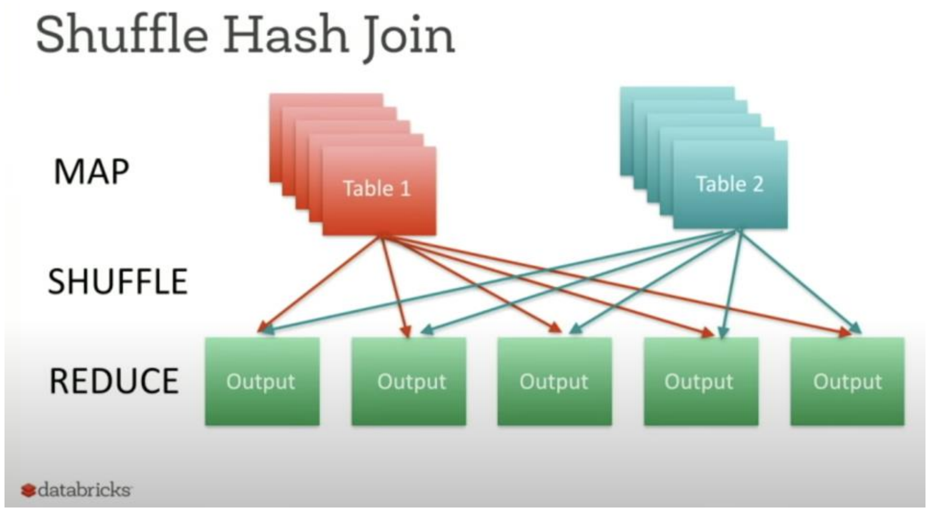


https://henning.kropponline.de/2016/12/11/broadcast-join-with-spark/

Join Strategies in Spark

Shuffle Hash

1. When both data frames are partitioned using same partitioner. So join keys will fall into the same partition.



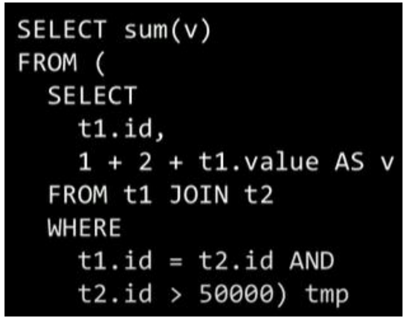
https://databricks.com/session/optimizing-apache-spark-sql-joins

Join Strategies in Spark

When to use which?

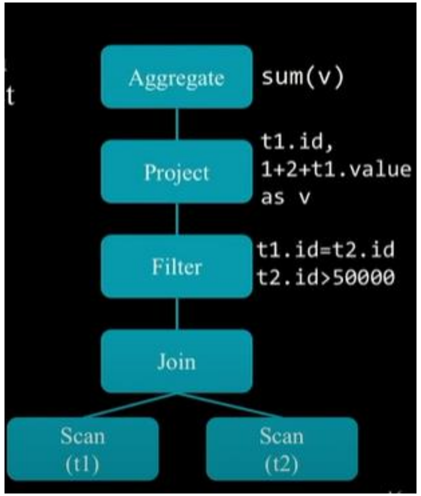
1. When both datasets are too big : Sort-merge join/ Shuffle-hash join
2. When one table is big and another is small : Broadcast Hash Join

Spark SQL Optimization: An Join Example

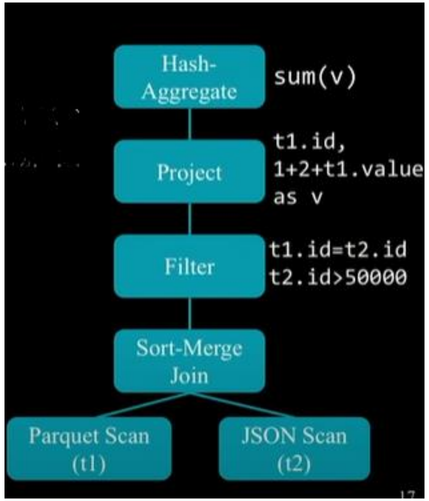


```
SELECT sum(v)
FROM (
  SELECT
    t1.id,
    1 + 2 + t1.value AS v
  FROM t1 JOIN t2
  WHERE
    t1.id = t2.id AND
    t2.id > 50000) tmp
```

SQL Query



Logical Plan – describes computation on datasets without defining how to conduct computation



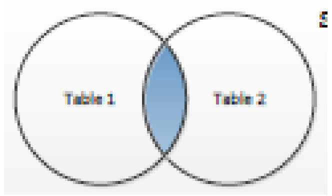
Physical Plan – describes computation on datasets with specific definitions on how to conduct computation

DAG of RDDs

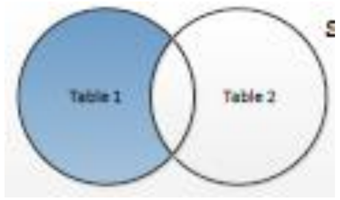
https://www.databricks.com/session/a-deep-dive-into-spark-sqls-catalyst-optimizer

Other Core Joins

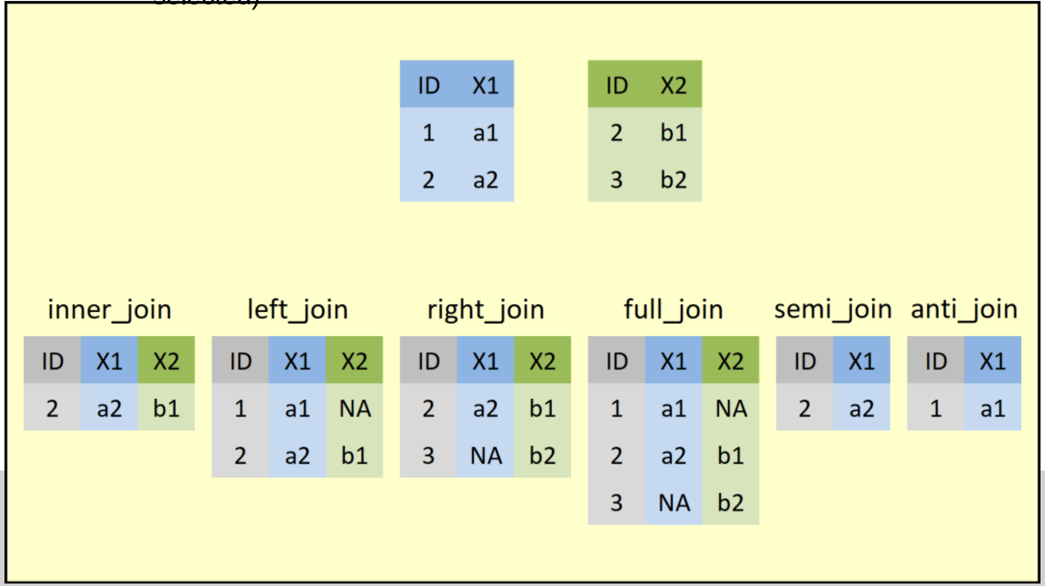
- INNER
- OUTER
- LEFT
- RIGHT
- LEFT SEMI
- LEFT ANTI



Left Semi Join (similar with inner join, but only the left table columns and values are selected)



Left anti Join



ID	X1
1	a1
2	a2

ID	X2
2	b1
3	b2

inner_join

ID	X1	X2
2	a2	b1

left_join

ID	X1	X2
1	a1	NA
2	a2	b1

right_join

ID	X1	X2
2	a2	b1
3	NA	b2

full_join

ID	X1	X2
1	a1	NA
2	a2	b1
3	NA	b2

semi_join

ID	X1
2	a2

anti_join

ID	X1
1	a1

Thank You!

See you next week.