SQL Joins - FOREIGN KEY, INNER JOIN, LEFT OUTER JOIN

- As the name shows, JOIN means *to combine something*. In case of SQL, JOIN means **"to combine two or more tables"**.
- A `JOIN` clause in SQL is used to combine rows from two or more tables, based on a related column between them. The `JOIN` clause is a powerful tool for querying data from multiple tables.

  > it is important to join tables through `primary key` columns. `Primary key` columns are unique identifiers for each row in a table, so they ensure that the data is accurate and consistent. When you join tables through `primary key` columns, you are ensuring that the rows in the two tables are related to each other.

- In short, joining tables through primary key columns is a good practice that helps to ensure the accuracy, consistency, and referential integrity of your data.
- Here are some additional benefits of joining tables through primary key columns:
  - It can help to improve the performance of your queries.
  - It can make it easier to write complex queries.
  - It can help to reduce the risk of data corruption.

TYPES OF JOINS

- Inner Join
  - Natural Join
- Left (Outer) Join
- Right (Outer) Join

- (Full) Outer Join
- Left (Outer) Join Excluding Inner Join
- Right (Outer) Join Excluding Inner Join
- (Full) Outer Join Excluding Inner Join
- Cross Join
- Equi-Join

WHY SQL JOIN IS USED?

If you want to access more than one table through a select statement.

If you want to combine two or more table then SQL JOIN statement is used .it combines rows of that tables in one table and one can retrieve the information by a SELECT statement.

FOREIGN KEY CONSTRAINT

- The `FOREIGN KEY` constraint is used to prevent actions that would destroy links between tables.
- A `FOREIGN KEY` is a field (or collection of fields) in one table, that refers to the `PRIMARY KEY` in another table.
- The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.
- The `FOREIGN KEY` constraint prevents invalid data from being inserted into the foreign key column, because it has to be one of the values contained in the parent table.

FOREIGN KEY on CREATE TABLE

The following SQL creates a `FOREIGN KEY` on the "PersonID" column when the "Orders" table is created:

```
-- MYSQL
CREATE TABLE Orders (
    OrderID int NOT NULL,
    OrderNumber int NOT NULL,
    PersonID int,
    PRIMARY KEY (OrderID),
    FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)
);

-- SNOWFLAKE
CREATE TABLE Orders (
    OrderID int NOT NULL,
    OrderNumber int NOT NULL,
    PersonID int,
    PRIMARY KEY (OrderID),
    PersonID DATATYPE NOT NULL REFERENCES Persons
);
```

FOREIGN KEY ON ALTER TABLE

To create a `FOREIGN KEY` constraint on the "PersonID" column when the "Orders" table is already created, use the following SQL:

```
ALTER TABLE Orders
ADD FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);
```

To allow naming of a `FOREIGN KEY` constraint, and for defining a `FOREIGN KEY` constraint on multiple columns, use the following SQL syntax:

```
ALTER TABLE Orders
ADD CONSTRAINT FK_PersonOrder
FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);
```

DROP A FOREIGN KEY

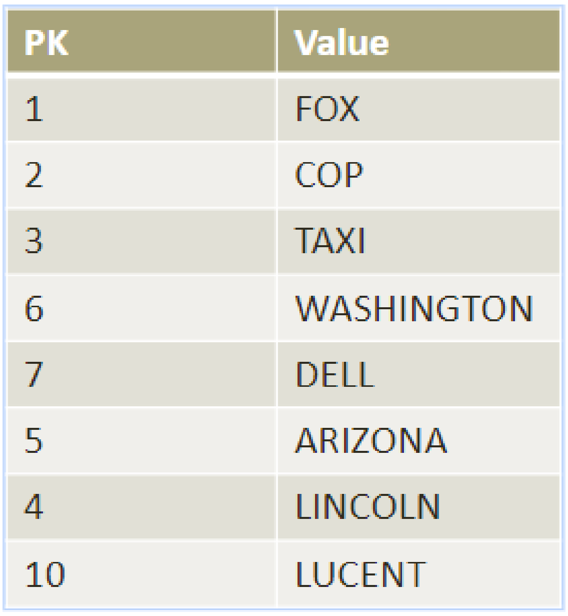
To drop a `FOREIGN KEY` constraint, use the following SQL:

```
ALTER TABLE Orders
DROP FOREIGN KEY FK_PersonOrder;

OR

ALTER TABLE Orders
DROP CONSTRAINT FK_PersonOrder;
```

LETS GET STARTED!!!

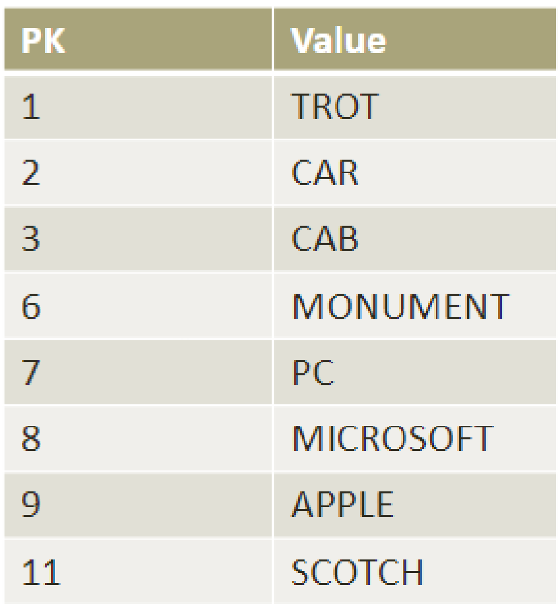
SAMPLE TABLE

TableA



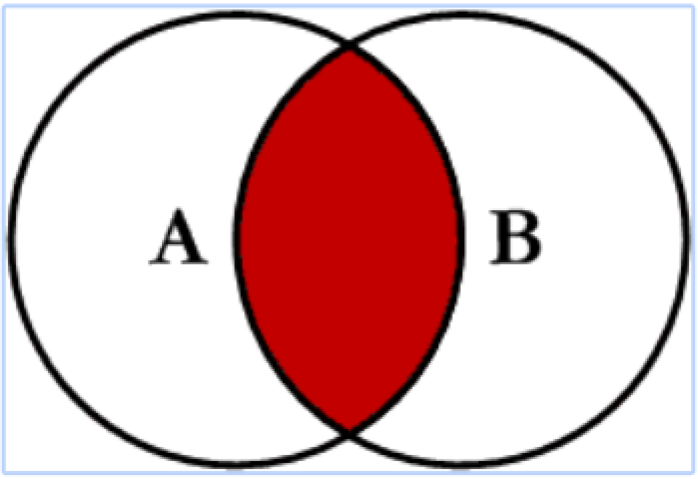
PK	Value
1	FOX
2	COP
3	TAXI
6	WASHINGTON
7	DELL
5	ARIZONA
4	LINCOLN
10	LUCENT

TableB



PK	Value
1	TROT
2	CAR
3	CAB
6	MONUMENT
7	PC
8	MICROSOFT
9	APPLE
11	SCOTCH

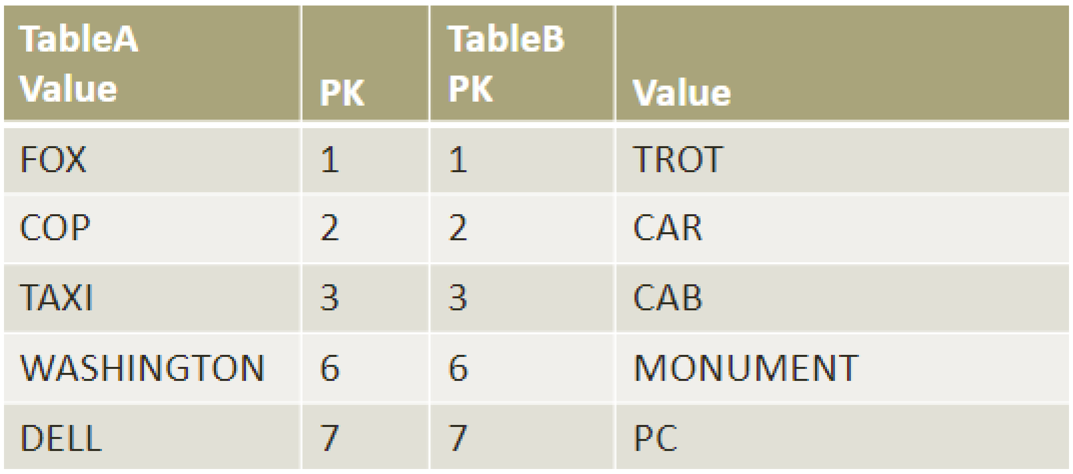
INNER JOIN



- Inner join produces only the set of records that match in both Table A and Table B
- Most commonly used, best understood join.
- If we write only `JOIN` it directly means `INNER JOIN`
- Inner joins just do not have to use equality to join the fields

- Can use <,>,<>

```
SELECT * FROM TableA INNER JOIN TableB ON
TableA.PK = TableB.PK

or

SELECT * FROM TableA JOIN TableB ON
TableA.PK = TableB.PK
```



TableA Value	PK	TableB PK	Value
FOX	1	1	TROT
COP	2	2	CAR
TAXI	3	3	CAB
WASHINGTON	6	6	MONUMENT
DELL	7	7	PC

- When your column name is same always use table name before column so that server can identify from which column you are picking

```
SELECT A.VALUE, A.PK, B.PK, B.VALUE FROM TableA AS A
INNER JOIN TableB AS B
ON TableA.PK = TableB.PK;
```

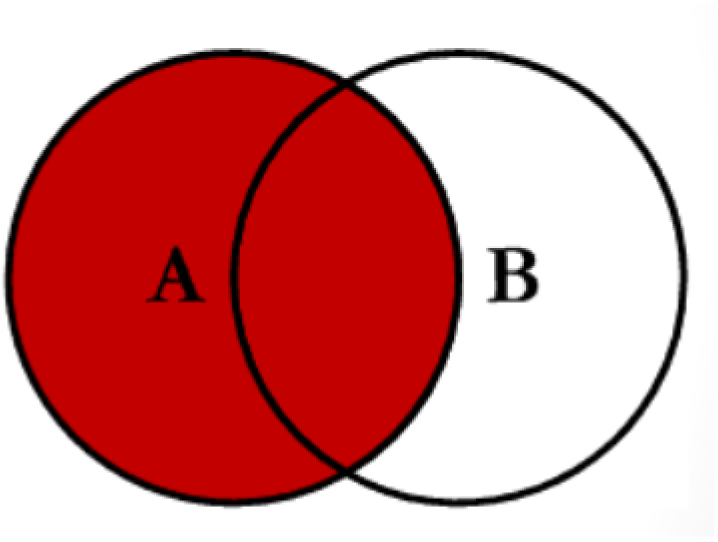
- It is a good practice or i would say a JOIN ethics, always write column name with table name so that if a fresher or anybody else look at your data, they don't get confuse by looking from which table this column came from. So always write column name with table name.
- 2nd most important point in joins, as you see in above table, we join tables using column PK in both table, so it is recommended not to show another column PK while displaying data, because it is duplicating of data, and why you need duplicate data or primary key, so its better to only use 1 column through which you have joined the tables.

EXAMPLE

- CREATE A TABLE CONTAINING ALL DATA FOR OWNER AND PETS CONTAINING OWNERS THOSE WHO ARE HAVING PETS.
    - Give alias to those which have same column name.

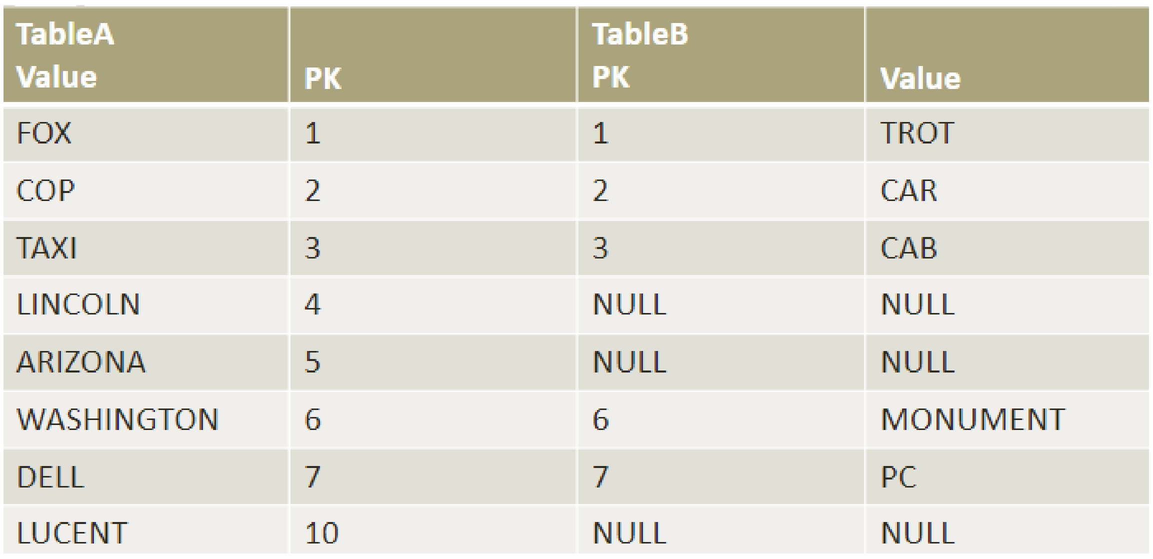
```
CREATE OR REPLACE TABLE AD_OWNER_PETS AS
SELECT OWN.*, PETS.PETID, PETS.NAME AS PET_NAME, PETS.KIND, PETS.GENDER, PETS.AGE
FROM AD_OWNER AS OWN
INNER JOIN AD_PETS AS PETS ON OWN.OWNERID = PETS.OWNERID;
```

LEFT OUTER JOIN OR LEFT JOIN



- The `LEFT JOIN` keyword returns all records from the left table (table1), and the matching records from the right table (table2). The result is 0 records from the right side, if there is no match. Then right side will contain `Null` value.

```
SELECT * FROM TableA
LEFT OUTER JOIN TableB  ON TableA.PK = TableB.PK
```



TableA		TableB	
Value	PK	PK	Value
FOX	1	1	TROT
COP	2	2	CAR
TAXI	3	3	CAB
LINCOLN	4	NULL	NULL
ARIZONA	5	NULL	NULL
WASHINGTON	6	6	MONUMENT
DELL	7	7	PC
LUCENT	10	NULL	NULL

Here you can see primary values are null too which will not happen in real word problem because you can't keep duplicate columns, we use this output just for demo purpose. Don't take it by heart.

IMPORTANT POINT

- Which Table you should choose for left side as you know it is left outer join so you need 2 table one is left and another is right, so which table you choose for left side.
- First you will see for which table stakeholder is asking, think with your brain, if there are multiple and every table is imporant then.
- `You will choose table which have maximum no. of records in it.`
  - If we don't know how tables are related then we mostly use `LEFT OUTER JOIN`

MASTER TABLE CREATION

```
-- WE HAVE 11 TABLES AND WE DON'T KNOW HOW ARE THEY RELATED TO EACH
-- OTHER WE JUST KNOW THEY HAVE COMMON COLUMNS
-- NOW WE CREATE A MASTER TABLE (MAKING ONE TABLE COMBINING ALL
-- THOSE TABLE.

CREATE OR REPLACE TABLE AD_MASTER_TABLE_NAME AS
SELECT <columns you want see in master table>
FROM AD_TABLE_1 AS T1 -- TABLE WITH MAX NO. OF RECORDS
-- we are joining on the basis of related column and they both are primary key.
-- always check from which table you are joining your column
LEFT OUTER JOIN AD_TABLE_2 AS T2 ON T1.column_x = T2.column_x
LEFT OUTER JOIN AD_TABLE_3 AS T3 ON T2.column_y = T3.column_y
LEFT OUTER JOIN AD_TABLE_4 AS T4 ON T1.column_z = T4.column_z
LEFT OUTER JOIN AD_TABLE_5 AS T5 ON T3.column_a = T5.column_a
LEFT OUTER JOIN AD_TABLE_6 AS T6 ON T2.column_b = T6.column_b;
```