SQL CONCEPTS

UNIT - 9

Prepared By – Prof. Rutika Patel

DDL(Data Definition Language)

- **DDL Commands :**
  - Create
  - Alter
  - Truncate
  - drop

Create

This command is used to create a new table in SQL. The user has to give information like table name, column names, and their datatypes.

Syntax –

```
CREATE TABLE table_name (
column_1 datatype,
column_2 datatype,
column_3 datatype, .... );
```

Example-

We need to create a table for storing Student information of a particular College. Create syntax would be as below

```
CREATE TABLE Student_info
( College_Id number(2),
College_name varchar(30),
Branch varchar(10));
```

ALTER

This command is used to add, delete or change columns in the existing table. The user needs to know the existing table name and can do add, delete or modify tasks easily.

Syntax –

Syntax to add a column to an existing table.

```
ALTER TABLE table_name
ADD column_name datatype;
```

Example –

In our Student_info table, we want to add a new column for CGPA. The syntax would be as below as follows.

```
ALTER TABLE Student_info
ADD CGPA number;
```

TRUNCATE

This command is used to remove all rows from the table, but the structure of the table still exists.

Syntax –
Syntax to remove an existing table.

```
TRUNCATE TABLE table_name;
```

Example –

The College Authority wants to remove the details of all students for new batches but wants to keep the table structure. The command they can use is as follows.

```
TRUNCATE TABLE Student_info;
```

DROP

This command is used to remove an existing table along with its structure from the Database.

Syntax –

Syntax to drop an existing table.

```
DROP TABLE table_name;
```

Example –

If the College Authority wants to change their Database by deleting the Student_info Table.

```
DROP TABLE Student_info;
```

DML(Data Manipulation Language)

- **DML Commands :**
  - SELECT
  - INSERT
  - UPDATE
  - DELETE

SELECT

SELECT:
This command is used to get data out of the database. It helps users of the database to access from an operating system, the significant data they need. It sends a track result set from one tables or more.

Syntax :

SELECT * FROM <table_name>;

Example :

SELECT * FROM student_info;
OR
SELECT * FROM students where due_fees <=20000;

INSERT

This command is used to enter the information or values into a row. We can connect one or more records to a single table within a repository using this instruction. This is often used to connect an unused tag to the documents.

Syntax :

```
INSERT INTO <table_name> (column_name1, column_name2) VALUES
('value1', 'value2');
```

Example :

```
INSERT INTO student_info (stu_id, stu_name , city)
VALUES (1, ‘Riha', ‘Ahmedabad');
```

UPDATE

This command is used to alter existing table records. Within a table, it modifies data from one or more records. This command is used to alter the data which is already present in a table.

Syntax :

```
UPDATE <table_name> SET <column_name = value> WHERE condition;
```

Example :

```
UPDATE student_info SET due_fees = 20000 WHERE stu_name = ‘Jeni';
```

DELETE

It deletes all archives from a table. This command is used to erase some or all of the previous table’s records. If we do not specify the ‘WHERE’ condition then all the rows would be erased or deleted.

Syntax :

```
DELETE FROM <table_name> WHERE <condition>;
```

Example :

```
DELETE FROM students WHERE stu_id = '001';
```

DCL Commands

- DCL (Data Controlling Language) is a query language that allows users to retrieve and edit data held in databases. The types of Data Controlling Language commands include Grant and Revoke.

GRANT Command

User access privileges to a database are given by this command. It can be used to grant SELECT, INSERT, UPDATE, and DELETE privileges to a user on a single table or several tables.

- **Syntax:**

```
    GRANT SELECT, UPDATE ON MY_TABLE TO SOME_USER,
ANOTHER_USER;
```

- **Example:**

```
        GRANT INSERT, SELECT on accounts to Alex
```

- Using this command, Alex has been granted permissions on accounts database objects like he can query or insert into accounts.

REVOKE Command:

- To take back permissions from the user REVOKE command is used. It is used to revoke a privilege (by default) or a specific command, such as UPDATE or DELETE, depending on the situation.
- **Syntax:**

  REVOKE privilege_name ON object_name FROM {user_name |PUBLIC | role_name}

- **Example:**

  REVOKE INSERT, SELECT on accounts from John

- Using this command, the permissions of John like query or insert on accounts database objects has been removed.

Logical Operators

- The logical operators are used to perform operations such as ALL, ANY, NOT, BETWEEN etc.

Operator	Description
ALL	Used to compare a specific value to all other values in a set
ANY	Compares a specific value to any of the values present in a set.
IN	Used to compare a specific value to the literal values mentioned.
BETWEEN	Searches for values within the range mentioned.
AND	Allows the user to mention multiple conditions in a WHERE clause.
OR	Combines multiple conditions in a WHERE clause.
NOT	A negate operators, used to reverse the output of the logical operator.
EXISTS	Used to search for the row’s presence in the table.
LIKE	Compares a pattern using wildcard operators.

Example[ANY] :

```
SELECT * FROM Students
WHERE Age > ANY (SELECT Age FROM Students WHERE Age > 21);
```

- **OUTPUT:**

StudentID	FirstName	LastName	Age
1	Atul	Mishra	23
5	Vaibhav	Gupta	25

Example[BETWEEN & AND] :

```
SELECT * FROM Students
WHERE Age BETWEEN 22 AND 25;
```

- **OUTPUT:**

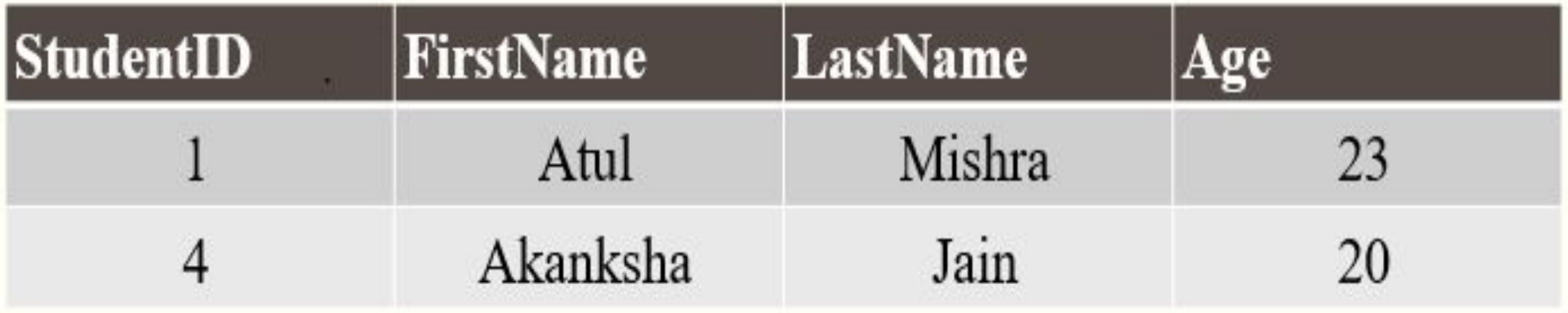
StudentID	FirstName	LastName	Age
1	Atul	Mishra	23

IN Operator

- **IN :**

```
SELECT  * FROM  Students
WHERE  Age IN   ( ’23’, ’20’);
```

- **OUTPUT:**

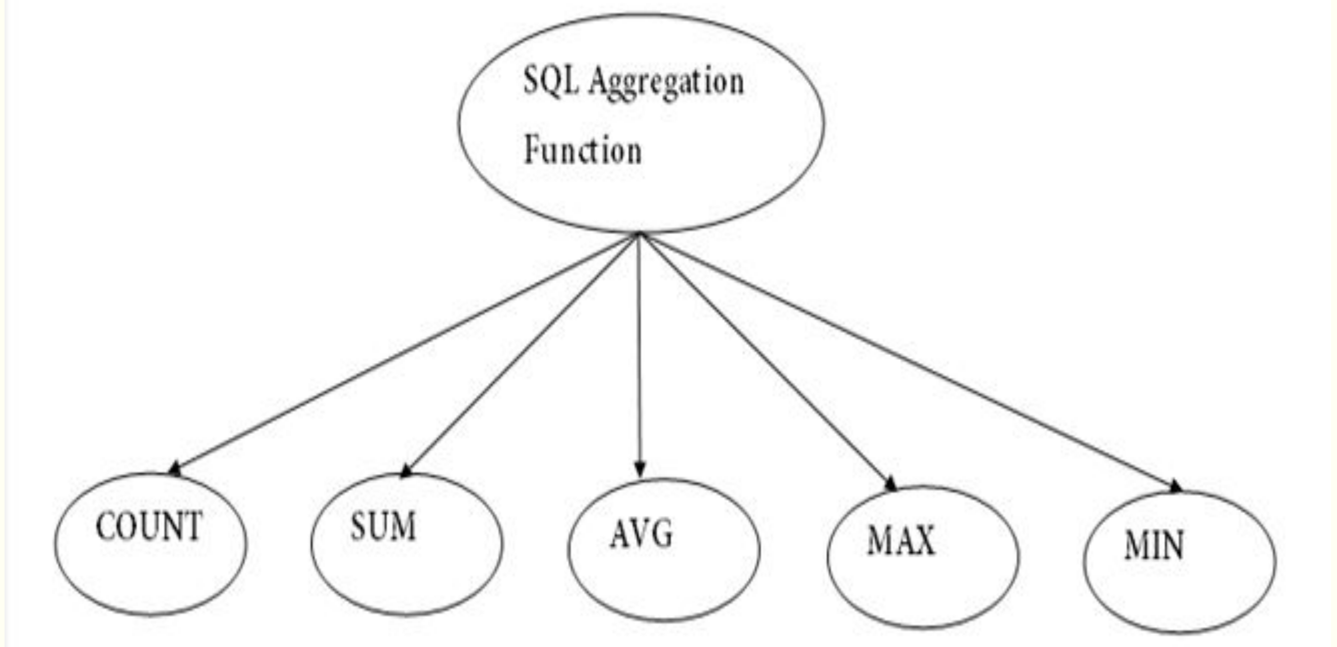


StudentID	FirstName	LastName	Age
1	Atul	Mishra	23
4	Akanksha	Jain	20

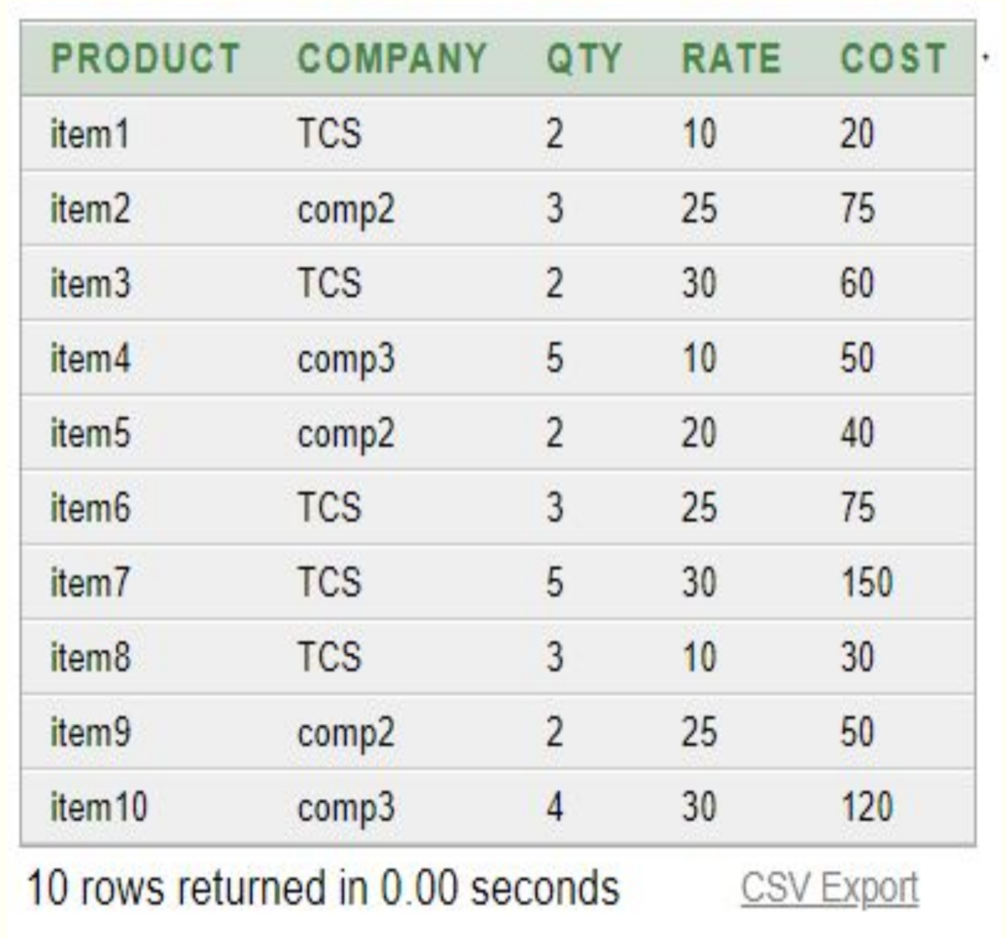
Aggregate functions

- SQL aggregation function is used to perform the calculations on multiple rows of a single column of a table. It returns a single value.

- It is also used to summarize the data.



Cont…



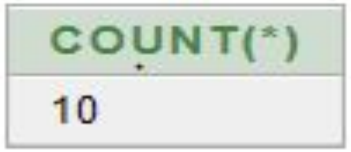
PRODUCT	COMPANY	QTY	RATE	COST
item1	TCS	2	10	20
item2	comp2	3	25	75
item3	TCS	2	30	60
item4	comp3	5	10	50
item5	comp2	2	20	40
item6	TCS	3	25	75
item7	TCS	5	30	150
item8	TCS	3	10	30
item9	comp2	2	25	50
item10	comp3	4	30	120

10 rows returned in 0.00 seconds CSV Export

1. Count

SELECT COUNT(*)

FROM Product_master;

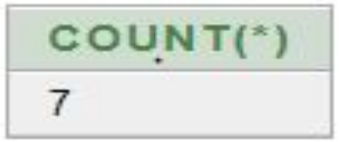


COUNT(*)
10

Example: COUNT with WHERE

SELECT COUNT(*) FROM Product_master

WHERE rate>=20;



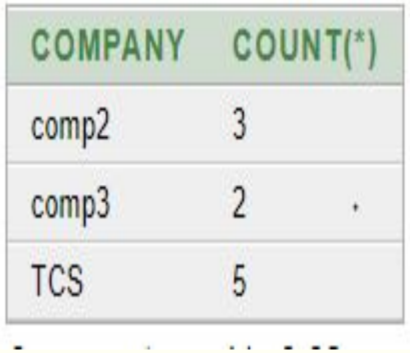
COUNT(*)
7

Example: COUNT() with GROUP BY

SELECT company, COUNT(*)

FROM Product_master

GROUP BY company;



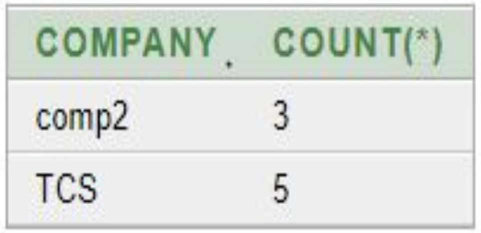
COMPANY	COUNT(*)
comp2	3
comp3	2
TCS	5

Cont…

Example: COUNT() with HAVING

SELECT company, COUNT(*) FROM Product_master

GROUP BY company HAVING COUNT(*)>2;



COMPANY	COUNT(*)
comp2	3
TCS	5

2. SUM Function

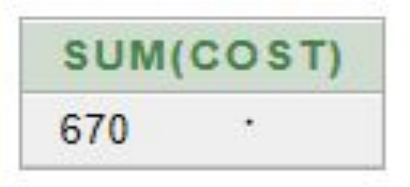
- Sum function is used to calculate the sum of all selected columns. It works on numeric fields only.
- **Syntax**

SUM() or

SUM([ALL|DISTINCT] expression)

Example: SUM()

SELECT SUM(COST) FROM Product_master ;



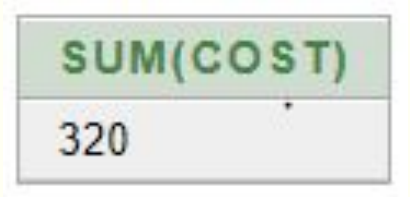
SUM(COST)
670

Example: SUM() with WHERE

SELECT SUM(COST)

FROM Product_master

WHERE qty>3;



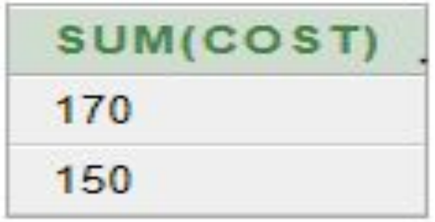
SUM(COST)
320

Cont…

- **Example: SUM() with GROUP BY**

SELECT SUM(COST) FROM Product_master

WHERE QTY>3 GROUP BY company;



SUM(COST)
170
150

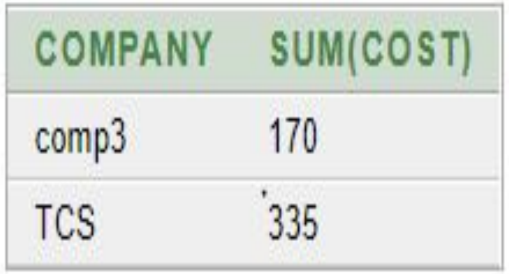
Example: SUM() with HAVING

SELECT company, SUM(COST)

FROM Product_master

GROUP BY company

HAVING SUM(COST)>=170;



COMPANY	SUM(COST)
comp3	170
TCS	335

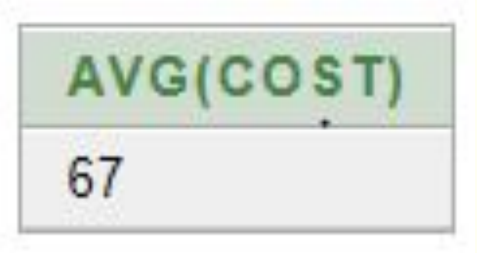
3. AVG function

- The AVG function is used to calculate the average value of the numeric type. AVG function returns the average of all non-Null values.
- **Syntax**

AVG() or AVG([ALL|DISTINCT] expression)

- **Example**

SELECT AVG(COST) FROM Product_master ;



AVG(COST)
67

4. MAX Function

- MAX function is used to find the maximum value of a certain column. This function determines the largest value of all selected values of a column.
- **Syntax**

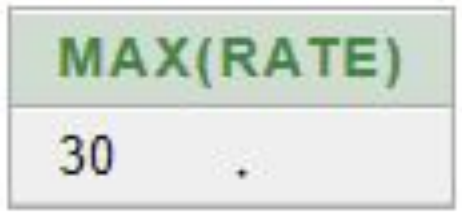
MAX() or

MAX([ALL|DISTINCT] expression)

- **Example**

SELECT MAX(RATE)

FROM Product_master;



MAX(RATE)
30 .

5. MIN Function

- MIN function is used to find the minimum value of a certain column. This function determines the smallest value of all selected values of a column.
- **Syntax**

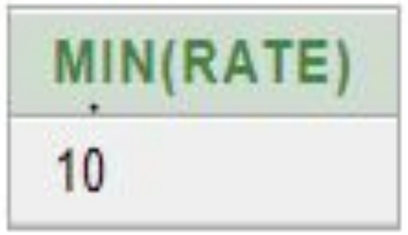
MIN() or

MIN([ALL|DISTINCT] expression)

Example

SELECT MIN(RATE)

FROM Product_master;



MIN(RATE)
10

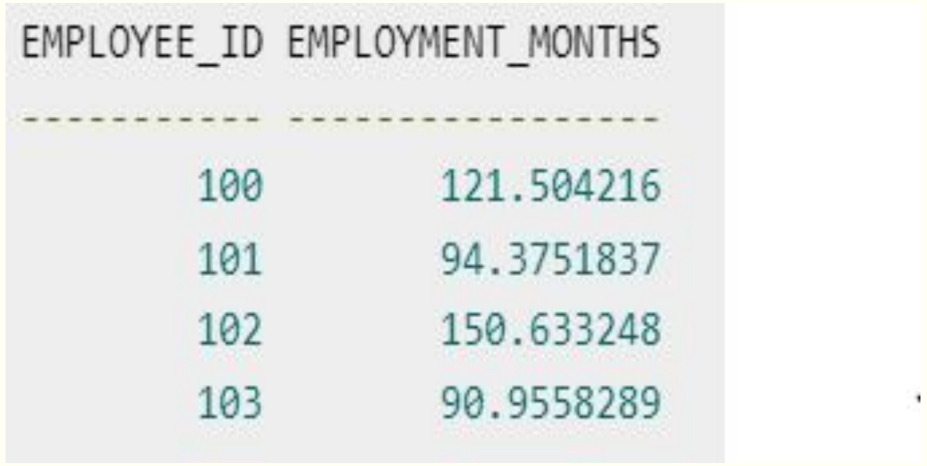
Date functions

- Date arithmetic operations return date or numeric values. Functions under the category are MONTHS_BETWEEN, ADD_MONTHS, ROUND and TRUNC.

  - MONTHS_BETWEEN function returns the count of months between the two dates.
  - ADD_MONTHS function add 'n' number of months to an input date.
  - ROUND and TRUNC functions are used to round and truncates the date value.

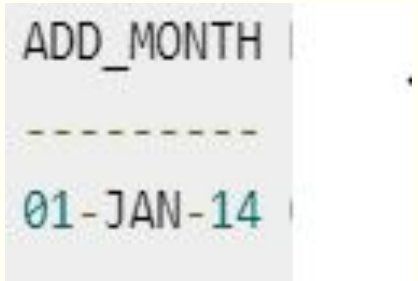
Cont…

- **Example:**
  - SELECT employee_id, MONTHS_BETWEEN (sysdate, hire_date) Employment_months FROM employees WHERE rownum < 5;



```
EMPLOYEE_ID EMPLOYMENT_MONTHS
----------- -----------------
        100        121.504216
        101        94.3751837
        102        150.633248
        103        90.9558289
```

  - SELECT ADD_MONTHS (sysdate, 5) ;



```
ADD_MONTH
---------
01-JAN-14
```

Cont…

- **Example:**
  - SELECT SYSDATE, ROUND(SYSDATE,’MONTH’) “ROUND_MONTH” FROM DUAL;

SYSDATE	ROUND_MONTH
28-APR-20	01-MAY-20

  - SELECT SYSDATE, TRUNC(SYSDATE,’MONTH’) “TRUNC_MONTH” FROM DUAL;

SYSDATE	TRUNC_MONTH
28-APR-20	01-APR-20

Numeric Function

- **Numeric Functions** are used to perform operations on numbers and return numbers.

 ABS(): It returns the absolute value of a number.

Syntax: SELECT ABS(-243.5);

Output: 243.5

 POWER(): It returns m raised to the nth power.

Syntax: SELECT POWER(4, 2);

Output: 16

Cont..

- **ROUND():** It returns a number rounded to a certain number of decimal places.

  Syntax: SELECT ROUND(5.553);

  Output: 6

- **TRUNCATE():** This doesn’t work for SQL Server. It returns 7.53635 truncated to 2 places right of the decimal point.

  Syntax: SELECT TRUNCATE(7.53635, 2);

  Output: 7.53

- **SQRT():** It returns the square root of a number.

  Syntax: SELECT SQRT(25);

  Output: 5

Character Functions

- SQL provides a rich set of character functions that allow you to get information about strings and modify the contents of those strings in multiple ways.

- **INITCAP :** This function converts alpha character values to uppercase for the first letter of each word and all others in lowercase. The words in the string is must be separated by either # or _ or space.

Syntax: INITCAP(SQL course)

Input1: SELECT INITCAP(‘pacific school of engineering') FROM DUAL;

Output1: Pacific School Of Engineering

Input2:SELECTINITCAP(‘PACIFIC_SCHOOL_OF_ENGINEERING')FROM DUAL;
Output2: Pacific_School_Of _Engineering

- **LOWER :** This function converts alpha character values to lowercase. LOWER will actually return a fixed-length string if the incoming string is fixed-length. LOWER will not change any characters in the string that are not letters, since case is irrelevant for numbers and special characters, such as the dollar sign ($) or modulus (%).

Syntax: LOWER(SQL course)

Input1: SELECT LOWER('PACIFIC SCHOOL OF ENGINEERING') FROM DUAL;

Output1: pacific school of engineering

Input2: SELECT LOWER('PACIFIC SCHOOL OF ENGINEERING @123') FROM DUAL;

Output2: pacific school of engineering@123

 UPPER : This function converts alpha character values to uppercase. Also UPPER function too, will actually return a fixed-length string if the incoming string is fixed-length. UPPER will not change any characters in the string that are not letters, since case is irrelevant for numbers and special characters, such as the dollar sign ($) or modulus (%).

Syntax: UPPER(SQL course)

Input1: SELECT UPPER(‘pacific') FROM DUAL;
Output1: PACIFIC

Input2: SELECT UPPER(‘pacific$508%7') FROM DUAL;
Output2: PACIFIC$508%7

 REPLACE : This function searches for a character string and, if found, replaces it with a given replacement string at all the occurrences of the string. REPLACE is useful for searching patterns of characters and then changing all instances of that pattern in a single function call. If a replacement string is not given, then REPLACE function removes all the occurrences of that character string in the input string. If neither a match string nor a replacement string is specified, then REPLACE returns NULL.

Syntax: REPLACE(Text, search_string, replacement_string)

Input1: SELECT REPLACE('DATA MANAGEMENT','DATA','DATABASE')
FROM DUAL;
Output1: DATABASE MANAGEMENT

Input2: SELECT REPLACE('abcdeabcccabdddeeabcc', 'abc') FROM DUAL;
Output2: deccabdddeec

- **SUBSTR :** This function returns a portion of a string from a given start point to an end point. If a substring length is not given, then SUBSTR returns all the characters till the end of string (from the starting position specified).

Syntax: SUBSTR('String', start-index, length_of_extracted_string)

Input1: SELECT SUBSTR('Database Management System', 9) FROM DUAL;

Output1: Management System

- **INSTR :** This function returns numeric position of a character or a string in a given string. Optionally, you can provide a position *m* to start searching, and the occurrence *n* of string. Also, if the starting position is not given, then it starts search from index 1, by default. If after searching in the string, no match is found then, INSTR function returns 0.

Syntax: INSTR(Column|Expression, 'String', [,m], [n])

Input: SELECT INSTR('Google apps are great applications','app',1,2) FROM DUAL;

Output: 23

Cont…

LTRIM():

- The **SQL LTRIM** function serves to remove unnecessary spaces on the left side of the string. The syntax is as follows:

LTRIM(string)

- ***string*** is the mandatory parameter that specifies the target string of characters we need to trim on the left side. The output is a copy of the specified string, but without the spaces at the beginning:

Syntax: SELECT LTRIM('   SQL Function');

Output: **‘SQL Function’**

Cont…

 RTRIM() :

- The **SQL RTRIM** function works in the same way as LTRIM – the difference is, it removes spaces on the right side of the string. The syntax is below:

  RTRIM(string)

- ***string*** is the required parameter that points to the string of characters where we need to remove the trailing spaces.

  Syntax: SELECT RTRIM('SQL Server   ');

  Output: **'SQL Server'**

Joins

- As the name shows, JOIN means to combine something. In case of SQL, JOIN means "to combine two or more tables".
- In SQL, JOIN clause is used to combine the records from two or more tables in a database.

EQUI Join

- EQUI JOIN creates a JOIN for equality or matching column(s) values of the relative tables. EQUI JOIN also create JOIN by using JOIN with ON and then providing the names of the columns with their relative tables to check equality using equal sign (=).

Syntax :

```
    SELECT column_list FROM table1, table2.... WHERE table1.column_name
= table2.column_name;
```

Example –

```
SELECT student.name, student.id, record.class, record.city FROM student, record
WHERE student.city = record.city;
```

Or

Syntax :

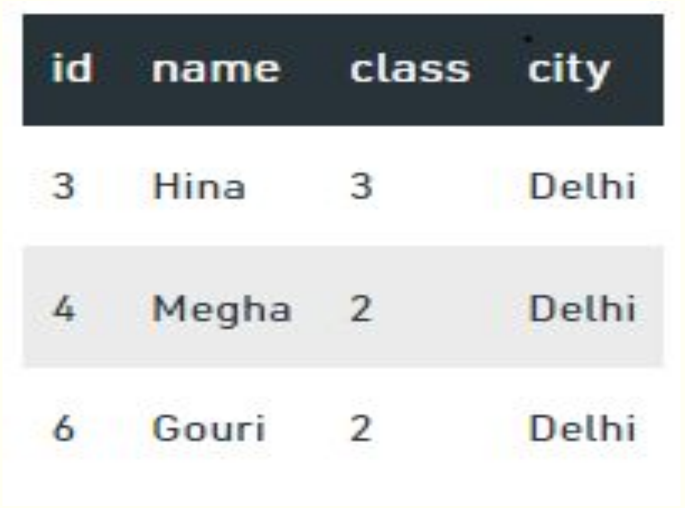
```
SELECT column_list FROM table1 JOIN table2 [ON(join_condition)]
```

Example –

```
SELECT student.name, student.id, record.class, record.city FROM
student JOIN record ON student.city = record.city;
```

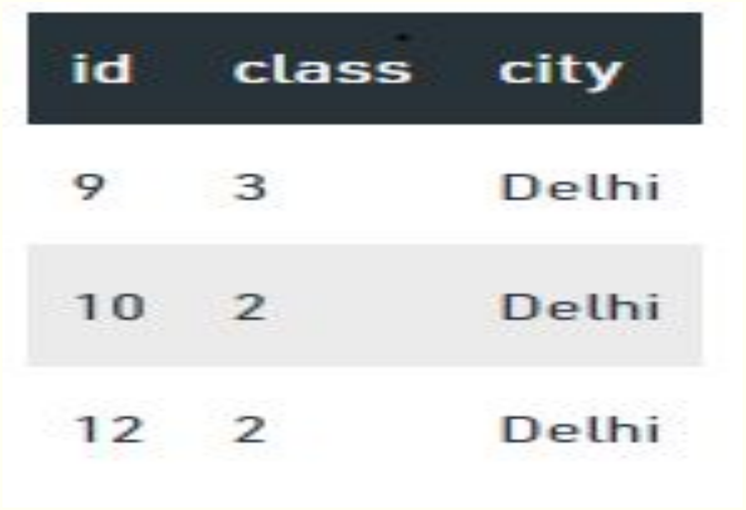
Example:

Table name — Student



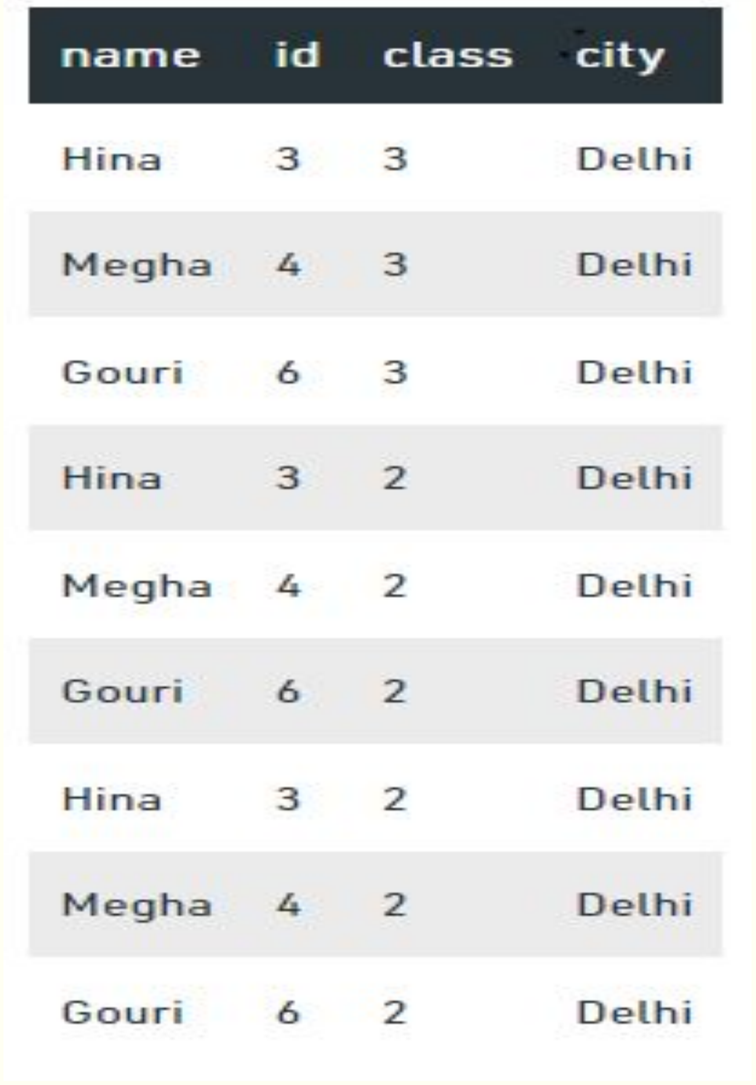
id	name	class	city
3	Hina	3	Delhi
4	Megha	2	Delhi
6	Gouri	2	Delhi

Table name — Record



id	class	city
9	3	Delhi
10	2	Delhi
12	2	Delhi

Output:



name	id	class	city
Hina	3	3	Delhi
Megha	4	3	Delhi
Gouri	6	3	Delhi
Hina	3	2	Delhi
Megha	4	2	Delhi
Gouri	6	2	Delhi
Hina	3	2	Delhi
Megha	4	2	Delhi
Gouri	6	2	Delhi

NON EQUI JOIN

NON EQUI JOIN performs a JOIN using comparison operator other than equal(=) sign like >, <, >=, <= with conditions.

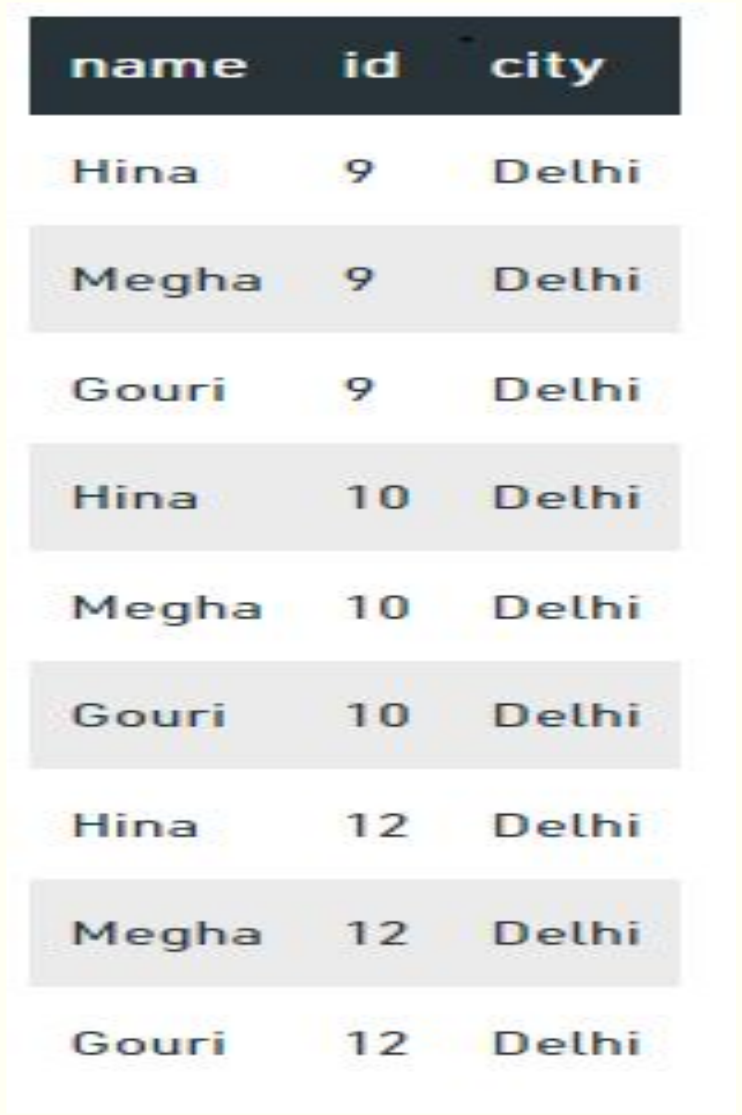
Syntax:

SELECT * FROM table_name1, table_name2 WHERE table_name1.column [> | < | >= | <=] table_name2.column;

Example:

SELECT student.name, record.id, record.city FROM student, record WHERE Student.id < Record.id ;

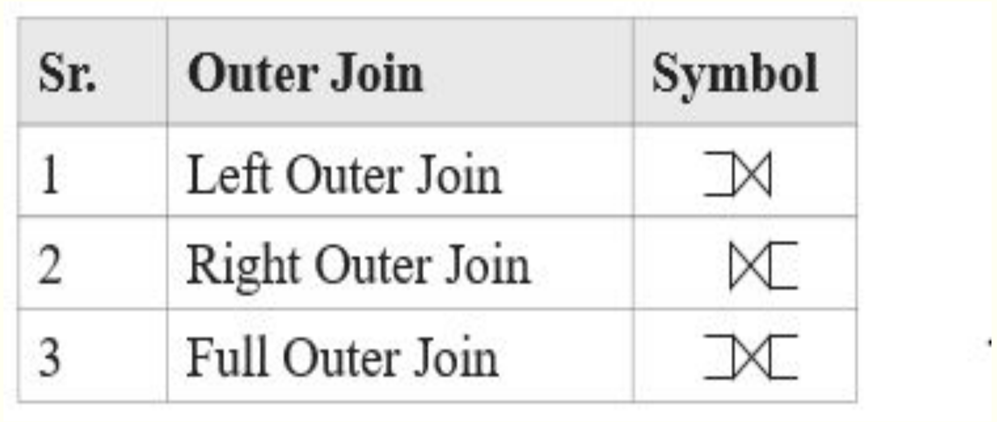
Output:



name	id	city
Hina	9	Delhi
Megha	9	Delhi
Gouri	9	Delhi
Hina	10	Delhi
Megha	10	Delhi
Gouri	10	Delhi
Hina	12	Delhi
Megha	12	Delhi
Gouri	12	Delhi

Outer Join

- In natural join some records are missing, if we want that missing records than we have to use outer join.
- Three types of Outer Join

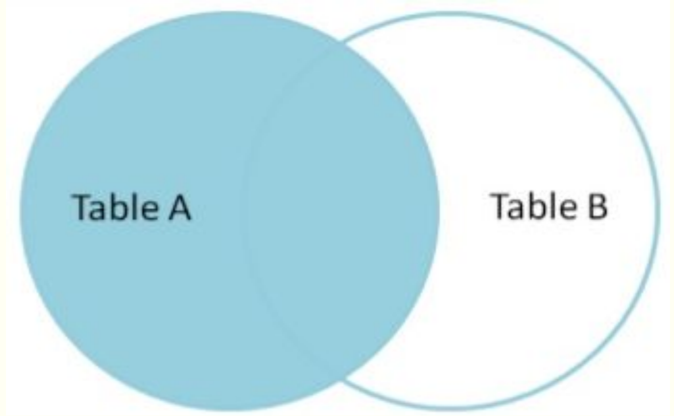


Sr.	Outer Join	Symbol
1	Left Outer Join	⟕
2	Right Outer Join	⟖
3	Full Outer Join	⟗

To perform a Outer Join there must be **one common attribute (column)** between two relations.

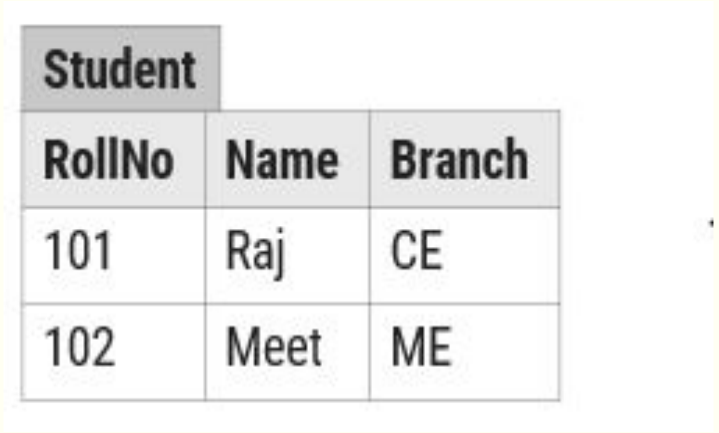
1) Left Outer Join

Symbol	⋈
Notation	*Relation-1 (R1)* ⟕ *Relation-2 (R2)* OR *Algebra-1* ⟕ *Algebra-2*
Operation	• Display all the tuples of the left relation even through there is no matching tuple in the right relation. • For such kind of tuples having no matching, the attributes of right relation will be padded with NULL in resultant relation.



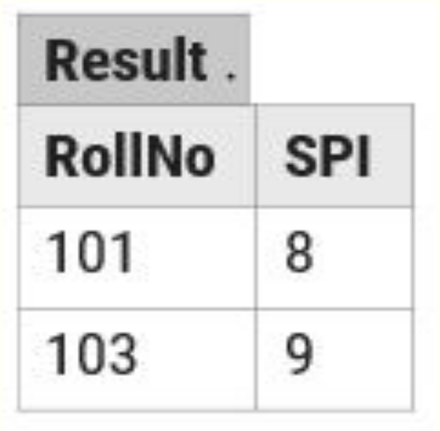
Cont…

- **Example:** Perform Left Outer Join between Student and Result.



Student

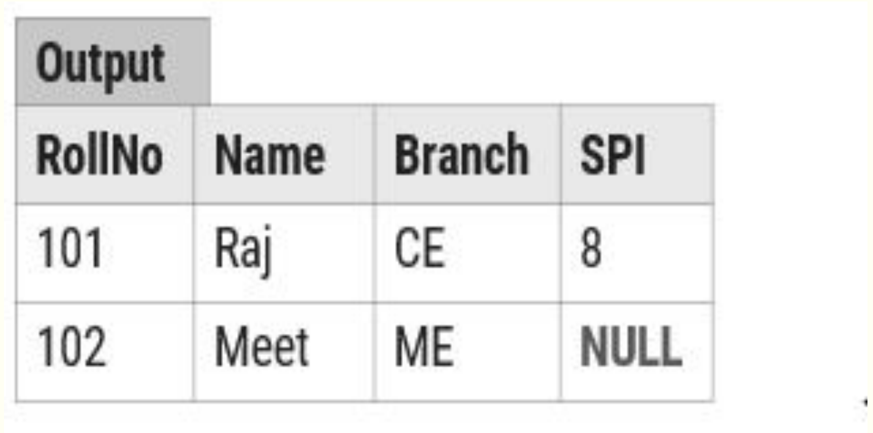
RollNo	Name	Branch
101	Raj	CE
102	Meet	ME



Result

RollNo	SPI
101	8
103	9

Answer: (Student) ⟕ (Result)

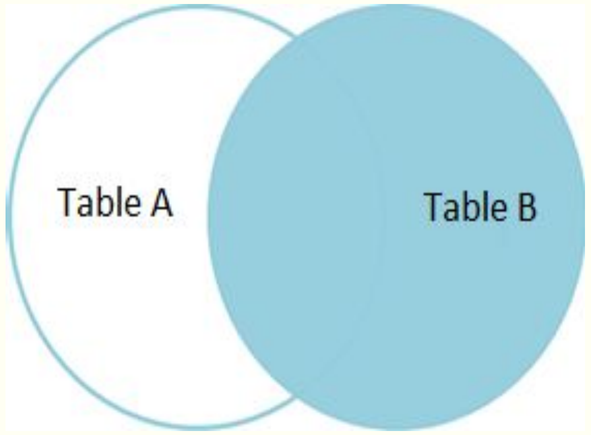


Output

RollNo	Name	Branch	SPI
101	Raj	CE	8
102	Meet	ME	NULL

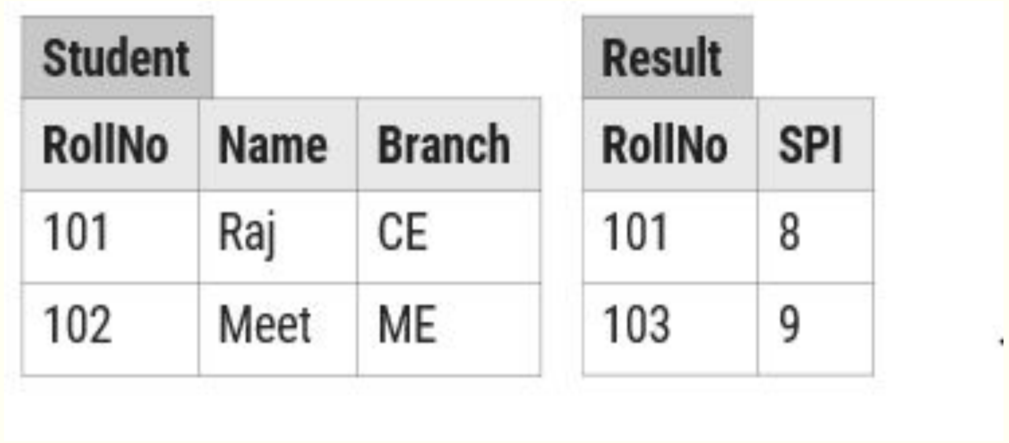
2) Right Outer Join

Symbol	$\bowtie$
Notation	*Relation-1 (R1)*⟖ *Relation-2 (R2)* OR *Algebra-1* ⟖*Algebra-2*
Operation	• Display all the tuples of right relation even through there is no matching tuple in the left relation. • For such kind of tuples having no matching, the attributes of left relation will be padded with NULL in resultant relation.



Cont…

- **Example:** Perform Right Outer Join between Student and Result.



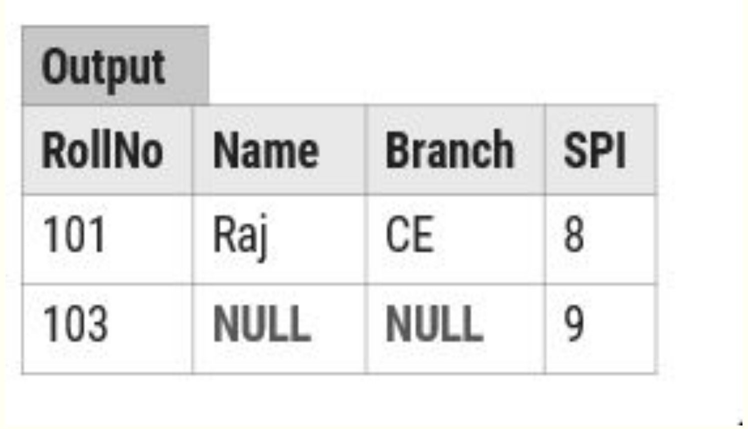
Student

RollNo	Name	Branch
101	Raj	CE
102	Meet	ME

Result

RollNo	SPI
101	8
103	9

Answer: (Student) ⟖ (Result)

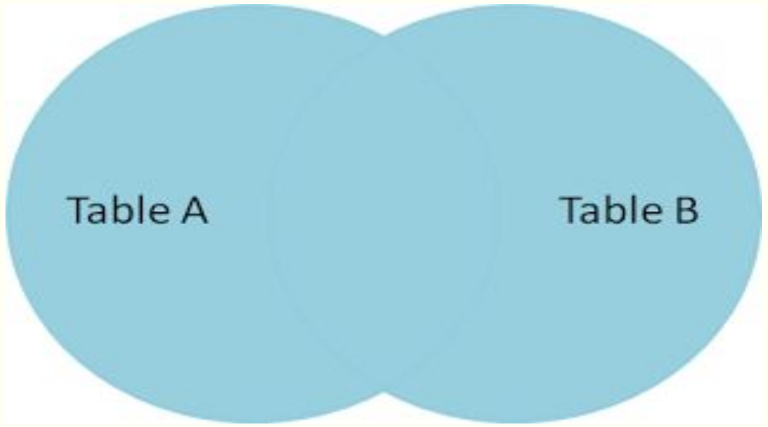


Output

RollNo	Name	Branch	SPI
101	Raj	CE	8
103	NULL	NULL	9

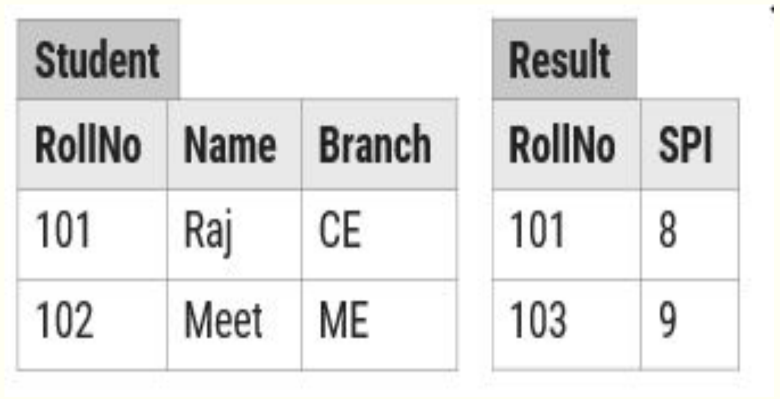
3) Full Outer Join

Symbol	⋈
Notation	*Relation-1 (R1)* ⟗ *Relation-2 (R2)* **OR** *Algebra-1* ⟗ *Algebra-2*
Operation	• Display **all the tuples of both of the relations**. It also pads null values whenever required. (Left outer join + Right outer join). • For such kind of **tuples having no matching**, it will be **padded with NULL** in resultant relation.



Cont…

- **Example:** Perform Full Outer Join between Student and Result.



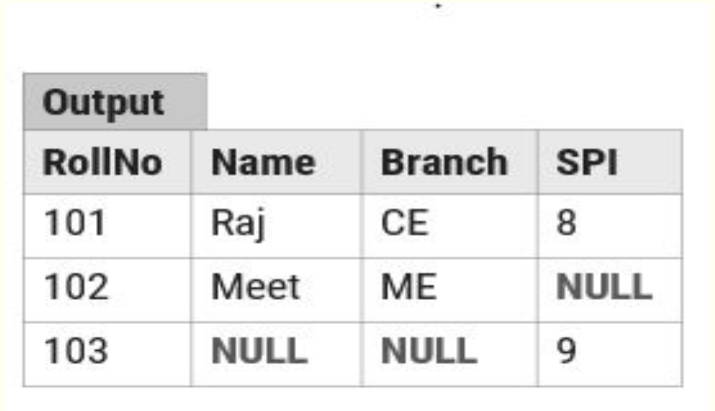
Student

RollNo	Name	Branch
101	Raj	CE
102	Meet	ME

Result

RollNo	SPI
101	8
103	9

Answer: (Student) ⟗ (Result)



Output

RollNo	Name	Branch	SPI
101	Raj	CE	8
102	Meet	ME	NULL
103	NULL	NULL	9

SELF JOIN:

As the name signifies, in SELF JOIN a table is joined to itself. That is, each row of the table is joined with itself and all other rows depending on some conditions. In other words we can say that it is a join between two copies of the same table.

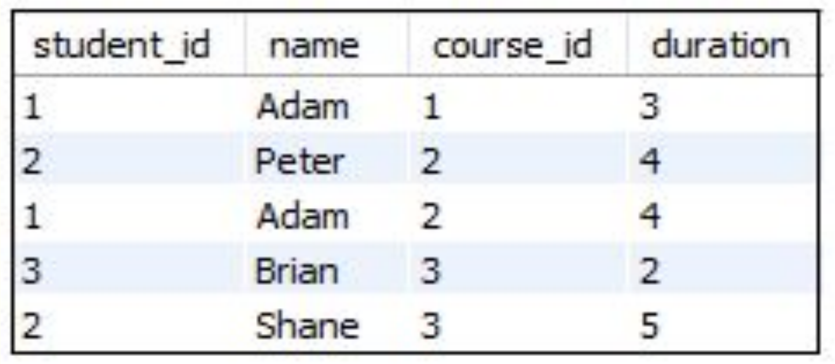
Syntax:

```
SELECT a.coulmn1 , b.column2 FROM table_name a, table_name b WHERE
some_condition;
```

table_name: Name of the table. **some_condition**: Condition for selecting the rows.

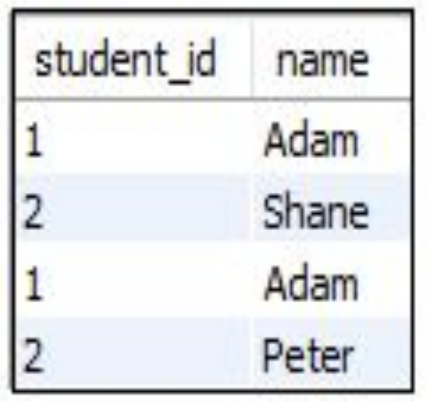
Cont…

Now, we are going to get all the result (student_id and name) from the table where **student_id** is equal, and **course_id** is not equal.



student_id	name	course_id	duration
1	Adam	1	3
2	Peter	2	4
1	Adam	2	4
3	Brian	3	2
2	Shane	3	5

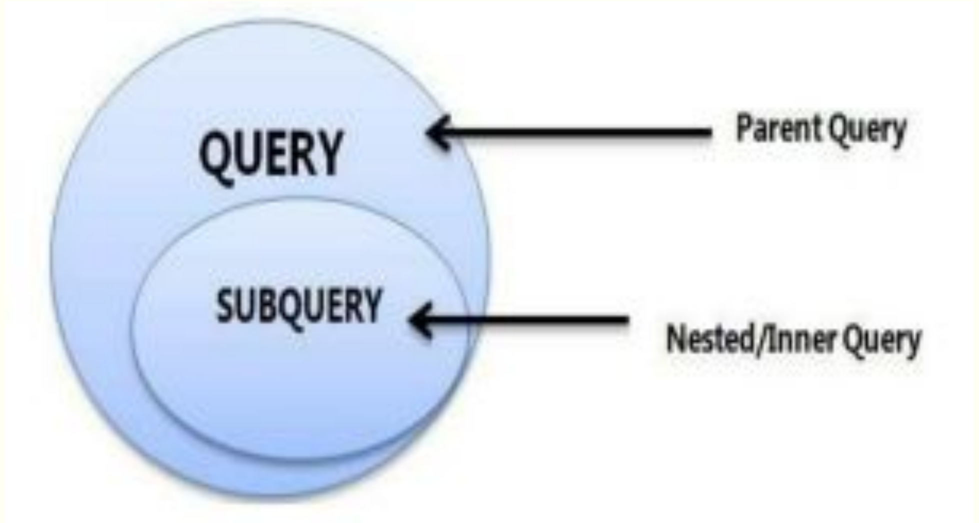
Example: SELECT s1.student_id, s1.name
FROM student AS s1, student s2
WHERE s1.student_id=s2.student_id
AND s1.course_id<>s2.course_id;



student_id	name
1	Adam
2	Shane
1	Adam
2	Peter

Sub queries

- Sub query or Inner query or Nested query is a query in a query.
- SQL sub query is usually added in the WHERE Clause of the SQL statement.
- Most of the time, a sub query is used when you know how to search for a value using a SELECT statement, but do not know the exact value in the database.



Cont…

- **Types of sub query**
  1. Single row sub query
  2. Multiple row sub query
  3. Correlated sub query

4. **<u>Single row sub query</u>**
   - Return 0 or 1 row
   - Can be used with <,>,<=,>= etc operators.

   - **Example:** Find out the name of staff whose salary is maximum

```
SELECT STAFF_NAME FROM STAFF
 WHERE STAFF_SALARY=(SELECT MAX (STAFF_SALARY) FROM STAFF);
```

Cont…

2. Multiple row sub query

- Return one or more rows
- Can be used with IN,NOT IN, ANY, ALL etc operators.

- **Example:** Find out the name of staff who are from “Computer” department using sub query.

```
SELECT * FROM STAFF WHERE DEP_ID IN

(SELECT DEP_ID FROM DEPARTMENT WHERE DEP_ID=‘2’);
```

Cont…

3. Correlated sub query

- If a sub query references columns in the parent query.
- This makes it impossible evaluate the sub query before evaluating the parent query.
- **Example:** Find out the name of staff who earn more salary then average salary.

```
SELECT STAFF_NAME FROM STAFF WHERE
STAFF_SALARY < (SELECT AVG(STAFF_SALARY) FROM STAFF);
```

Set Operators

Set operators combine the results of two or more queries into a single result.

Three types of Set operator

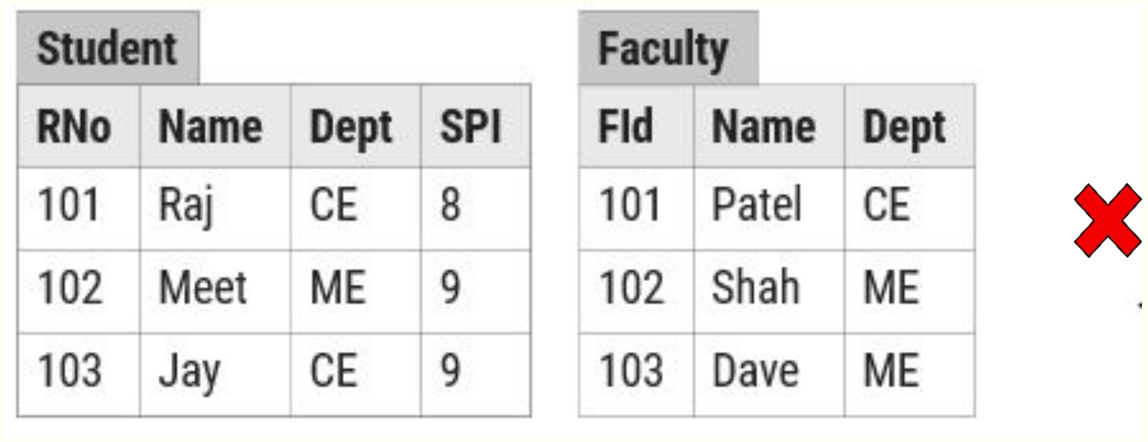
Sr No	Set Operator	Symbol
1	Union	U
2	Intersect / Intersection	∩
3	Minus / Set difference	–

Conditions: Set operators will take two or more queries as input, which must be union-compatible.

- Both queries should have **same (equal) number of columns**
- Corresponding **attributes should have the same data type** or **domain**

Conditions to perform Set Operators

Condition 1: Both queries should have same (equal) number of columns.



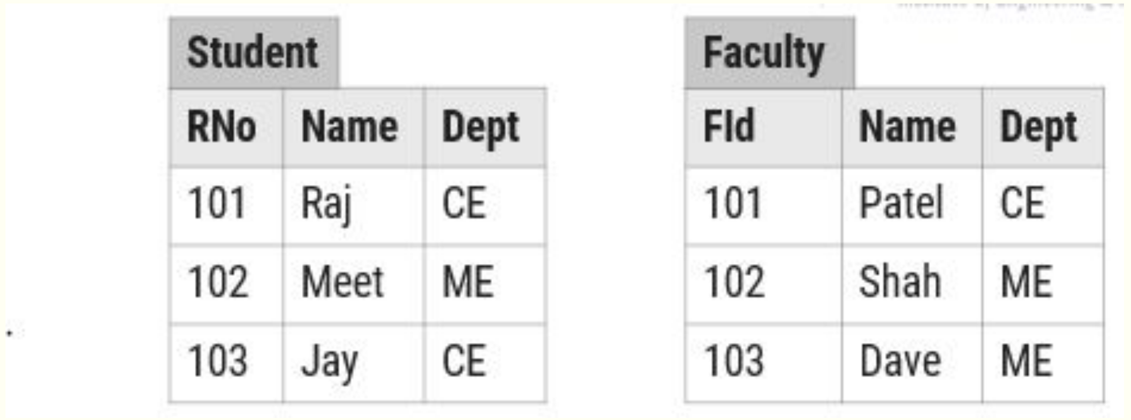
Student

RNo	Name	Dept	SPI
101	Raj	CE	8
102	Meet	ME	9
103	Jay	CE	9

Faculty

FId	Name	Dept
101	Patel	CE
102	Shah	ME
103	Dave	ME

✗



Student

RNo	Name	Dept
101	Raj	CE
102	Meet	ME
103	Jay	CE

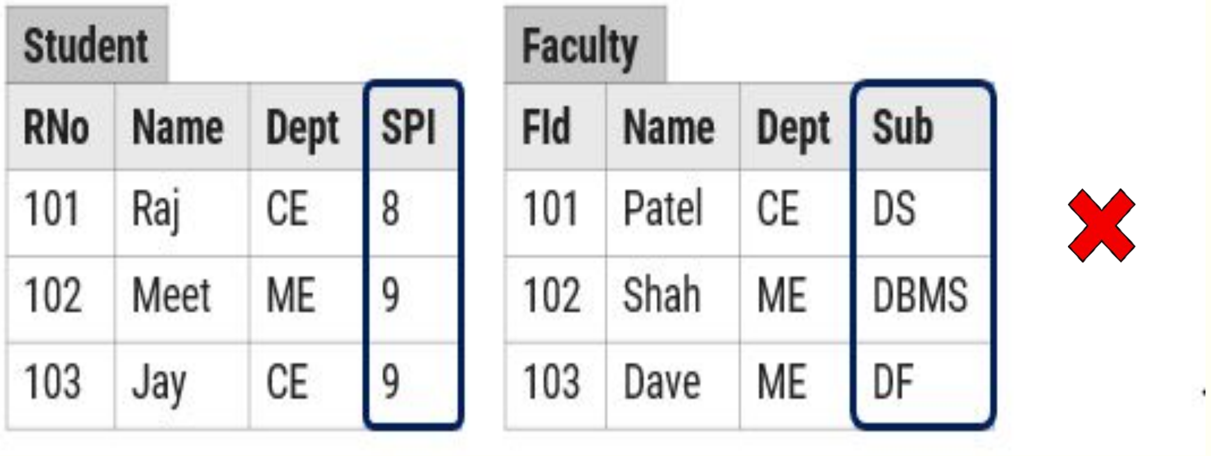
Faculty

FId	Name	Dept
101	Patel	CE
102	Shah	ME
103	Dave	ME

✓

Cont…

Condition 2: Corresponding attributes should have the same data type.



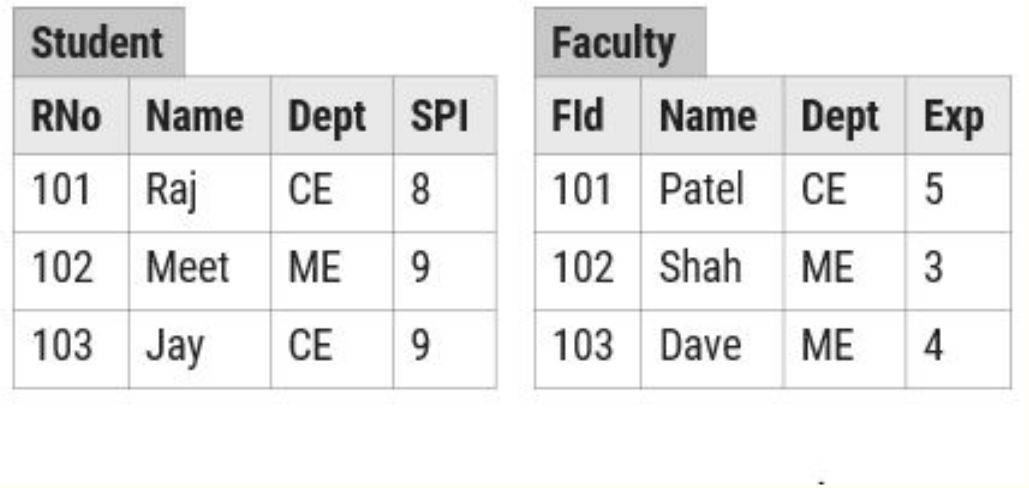
Student

RNo	Name	Dept	SPI
101	Raj	CE	8
102	Meet	ME	9
103	Jay	CE	9

Faculty

FId	Name	Dept	Sub
101	Patel	CE	DS
102	Shah	ME	DBMS
103	Dave	ME	DF

✗



Student

RNo	Name	Dept	SPI
101	Raj	CE	8
102	Meet	ME	9
103	Jay	CE	9

Faculty

FId	Name	Dept	Exp
101	Patel	CE	5
102	Shah	ME	3
103	Dave	ME	4

✓

Cont..

Exercise: Check whether following tables are compatible or not.

- A: (First_name(char), Last_name(char), Date_of_Birth(date))
- B: (FName(char), LName(char), PhoneNumber(number))

- A: (First_name(char), Last_name(char), Date_of_Birth(date))
- B: (FName(char), LName(char), DOB(date))

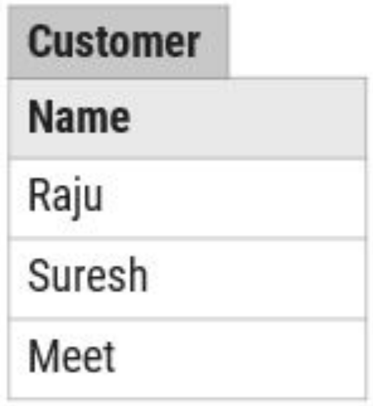
- Person (PersonID, Name, Address, Hobby)
- Professor (ProfessorID, Name, OfficeAddress, Salary)

Union Operator

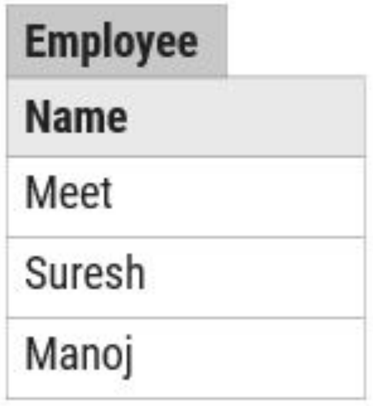
Symbol:	U
Notation:	*Relation-1 (R1)* U *Relation-2 (R2)* **OR** *Algebra-1* U *Algebra-2*
Operation:	• It displays all the tuples/records belonging to the first relation (left relation) or the second relation (right relation) or both. • It also eliminates duplicate tuples (tuples present in both relations appear once).

Example: Perform Union between Customer and Employee.

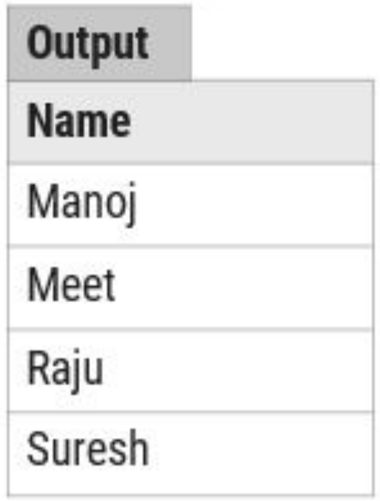
Answer: (Customer) U (Employee)



Customer
Name
Raju
Suresh
Meet



Employee
Name
Meet
Suresh
Manoj



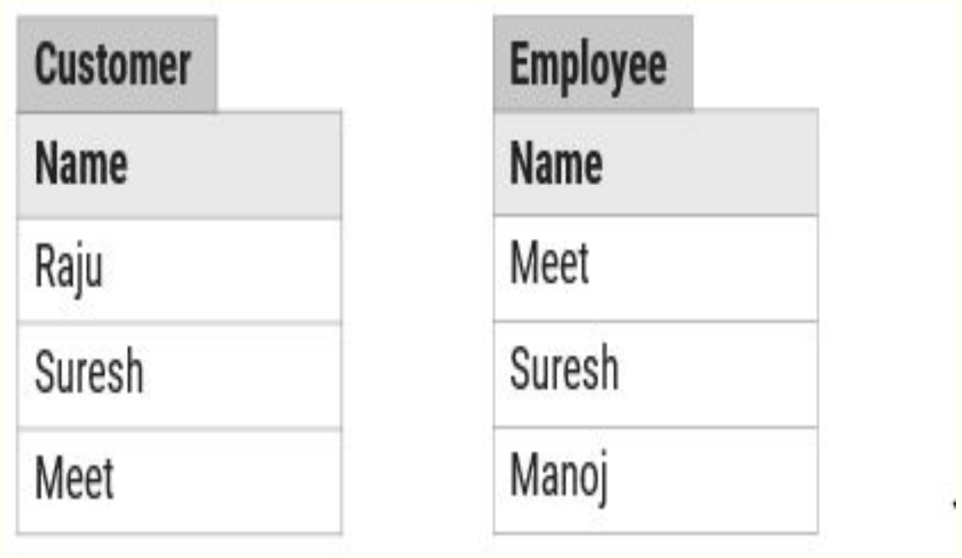
Output
Name
Manoj
Meet
Raju
Suresh

Intersect/ Intersection Operator

Symbol:	∩
Notation:	Relation-1 (R1) ∩ Relation-2 (R2) OR Algebra-1 ∩ Algebra-2
Operation:	• It displays all the tuples/records belonging to both relations. OR • It displays all the tuples/records which are common from both relations.

Example: Perform Intersection between Customer and Employee.

Answer: (Customer) ∩ (Employee)

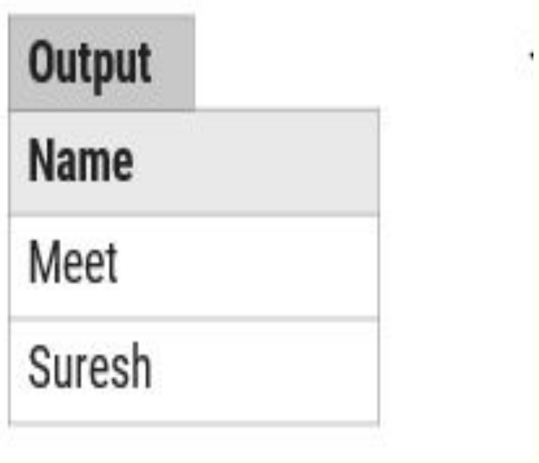


Customer

Name
Raju
Suresh
Meet

Employee

Name
Meet
Suresh
Manoj



Output

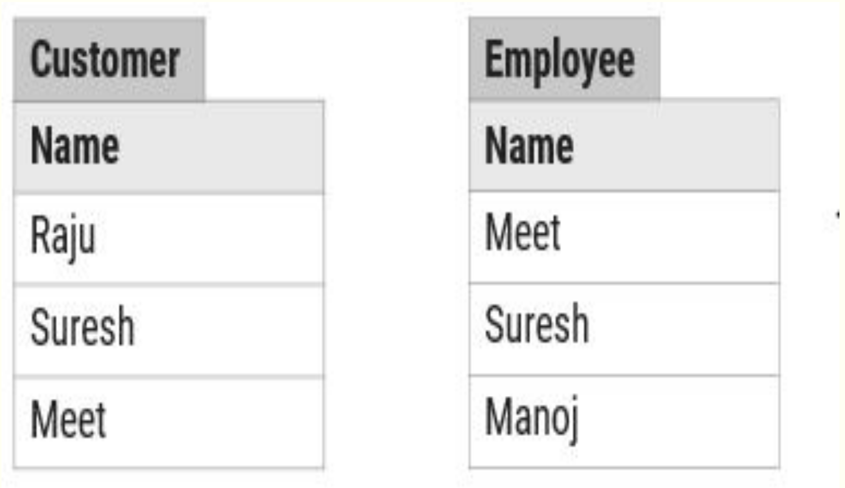
Name
Meet
Suresh

Minus/ Set difference Operator

Symbol:	−
Notation:	*Relation-1 (R1)* − *Relation-2 (R2)* **OR** *Algebra-1* − *Algebra-2*
Operation:	It displays all the tuples/records belonging to the first relation (left relation) but not in the second relation (right relation).

Example: Perform Set difference between Customer and Employee

Answer: (Customer) − (Employee)

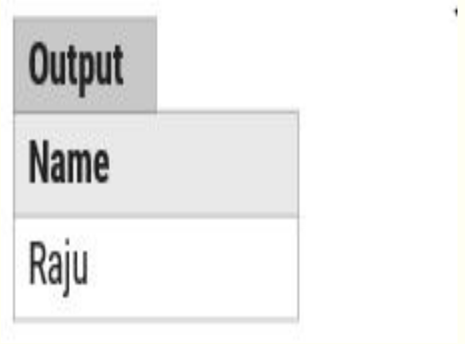


Customer

Name
Raju
Suresh
Meet

Employee

Name
Meet
Suresh
Manoj



Output

Name
Raju

Transaction control commands

- **TCL** stands for **Transaction Control Languages**. These commands are used for maintaining consistency of the database and for the management of transactions made by the DML commands.
- A **Transaction** is a set of SQL statements that are executed on the data stored in DBMS. Whenever any transaction is made these transactions are temporarily happen in database.So to make the changes permanent, we use **TCL** commands.
- The TCL commands are:
  - COMMIT
  - ROLLBACK
  - SAVEPOINT

COMMIT

- This command is used to save the data permanently.
- Whenever we perform any of the DML command like -INSERT, DELETE or UPDATE, these can be rollback if the data is not stored permanently. So in order to be at the safer side COMMIT command is used.

Syntax:

```
commit;
```

ROLLBACK

- This command is used to get the data or restore the data to the last savepoint or last committed state.

- If due to some reasons the data inserted, deleted or updated is not correct, you can rollback the data to a particular savepoint or if savepoint is not done, then to the last committed state.

Syntax:

```
    rollback;
```

SAVEPOINT

This command is used to save the data at a particular point temporarily, so that whenever needed can be rollback to that particular point.

Syntax:

```
    Savepoint A;
```

Thank You…!