Operators:

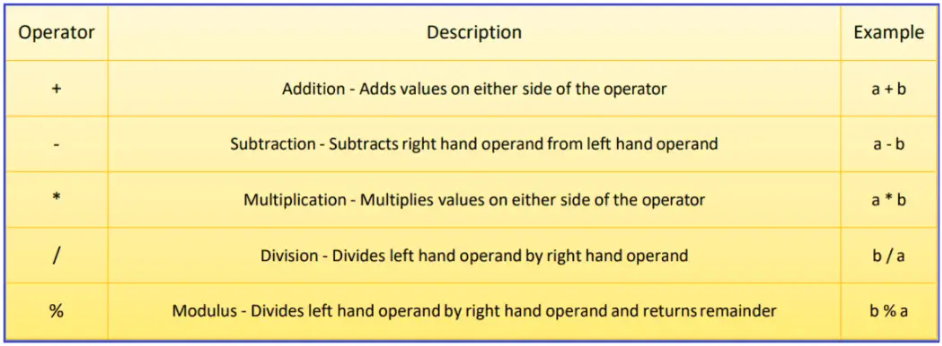
Operators are used to express the conditions in Select statements. The operator manipulates individual data items and returns a result. The data items are called operands or arguments.

* **Arithmetic operators**
* **Assignment operator**
* **Relational operators**
* **Logical operators**
* **Special Operators**
* **Set Operators**

**Arithmetic Operators in Oracle:**

The Arithmetic Operators in Oracle are used for performing mathematical calculations such as Addition, Subtraction, Multiplication, Module, and Division represented by the expected +, -, \* , %, and / respectively on the given operand values.

1. The arithmetic operations can be used to create expressions on number and date data.
2. The arithmetic operators can be used in any clause of a SQL statement.
3. SQL \* Plus ignores the blank spaces before and after the arithmetic operator.



 Operator Precedence

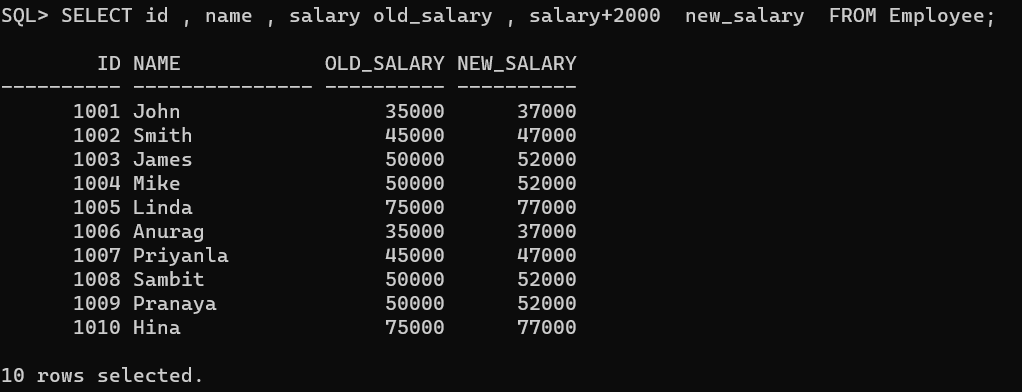
If the arithmetic expression contains more than one operator, then the multiplication and division operators are the highest priority and are evaluated first, and then the addition and minus (subtraction) operators are evaluated.

When two operators are having the same priority, then the expression is going to be evaluated from left to right in the Oracle database. The Parentheses in Oracle can also be used to force an operation to take priority over any other operators.

##### ****Arithmetic Operator Addition (+)****

Example: Display salary of employees with 2000 increments in their salary.

**SELECT ID, Name, Salary, Salary + 2000 “new\_salary” FROM Employee;**

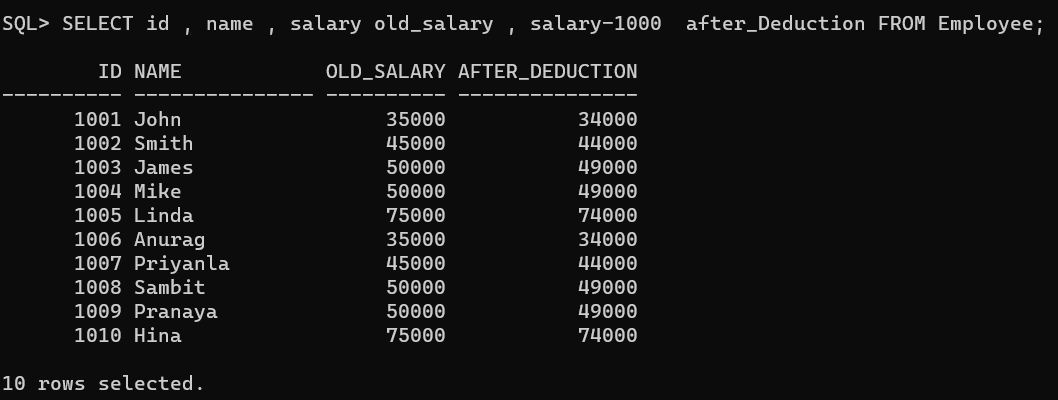


##### ****Arithmetic Operator Subtraction (-):****

This operator is used to perform the subtraction between two numbers

**Example:**Display the details of employees decreasing their salary by 200.

**SELECT ID, Name, Salary, Salary – 200 “after\_deduction” FROM Employee;**

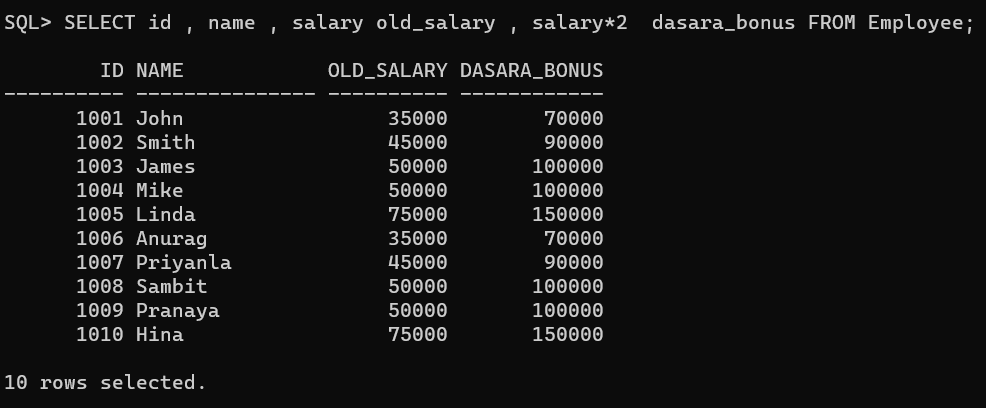


##### ****Arithmetic Operator Multiplication (\*):****

This operator is used to perform multiplication.

**Example**: Display the details of the employees Incrementing their salary two times.

**SELECT ID, Name, Salary, Salary \* 2 “dasara\_bonus” FROM Employee;**

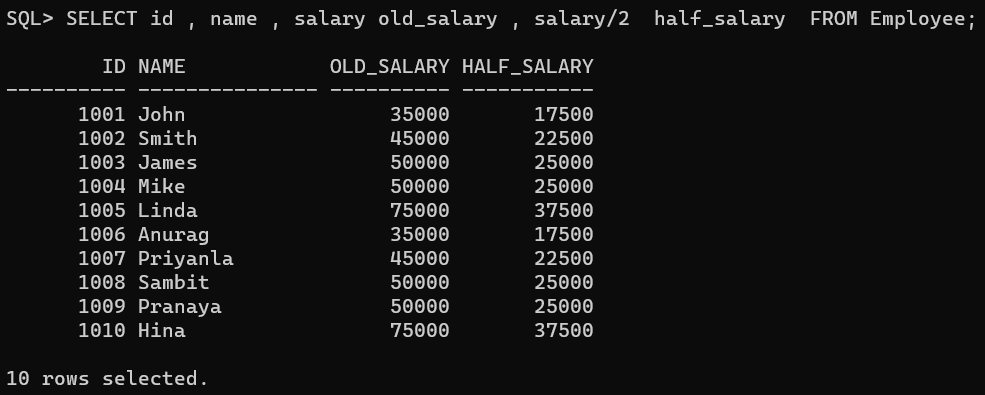


##### ****Arithmetic Operator Division (/):****

This operator is used to perform a Division test. The division will display only the Quotient value, not the remainder value

**Example**: Display half of the salary of employees.

**SELECT ID, Name, Salary, Salary / 2 “half\_salary” FROM Employee;**



In oracle, it is not possible to perform any calculations on String values (such as Varchar or Char).

##### ****Assignment Operator in Oracle****

The Assignment Operator in Oracle is used to assign or compare a value to a column or a field of a table. The equal sign (=) is the assignment operator where the value on the right is assigned to the value on the left.

Relational Operators:

 operators are used for comparing one expression with another expression. The relational operators determine whether the two values are equal or a value is greater than the other, or less than the other. The result of a comparison can be TRUE, FALSE, or NULL (When one or both the expressions contain NULL values).

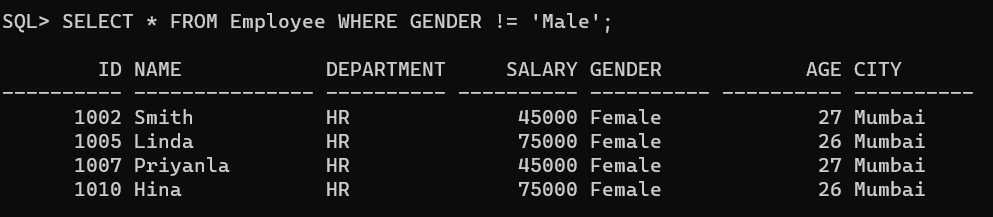
##### ****Types of Relational Operators in Oracle****

1. **Not Equal (!= or <>) Operator**
2. **Greater Than (>) Operator**
3. **Less Than (<) Operator**
4. **Greater Than or Equal To (>=) Operator**
5. **Less Than or Equal To (<=) Operator**

##### ****Not Equal (!=) Relational Operator in Oracle****

The Not Equal (!=) Operator in Oracle is just the opposite of the equal operator. That means this operator is used to check whether the two expressions are equal or not. If both the expressions are not equal then the condition becomes true and will return the not-matched records.

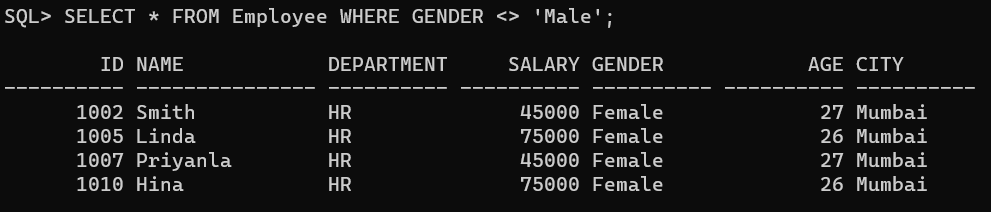
**SELECT \* FROM Employee WHERE Gender != ‘Male’;**



##### ****Not Equal (<>) Relational Operator in Oracle****

The Not Equal (<>) Operator in Oracle is the same as the Not Equal (!=) operator.

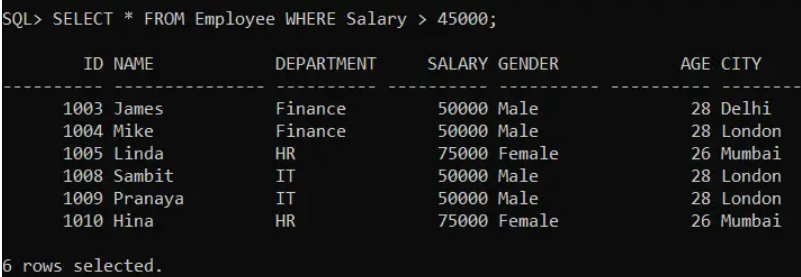
**SELECT \* FROM Employee WHERE Gender <> ‘Male’;**



##### ****Greater Than (>) Relational Operator in Oracle****

The Greater Than (>) Operator in Oracle is used to check whether the left-hand expression value is higher than the right-hand expression value. If the left-hand expression value is higher than the right-hand expression value then the condition becomes true and it will return the matched records

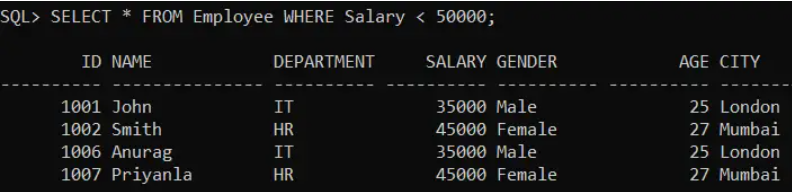
**SELECT \* FROM Employee WHERE Salary > 45000;**



##### ****Less Than (<) Relational Operator in Oracle****

The Less Than (>) Operator in Oracle is used to check whether the left-hand expression value is lower than the right-hand expression value. If the left-hand expression value is lower than the right-hand expression value then the condition becomes true and will return the matched records

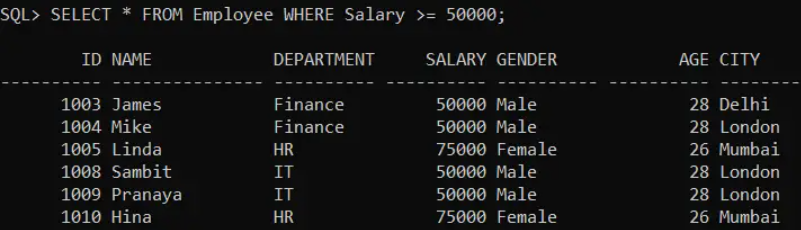
**SELECT \* FROM Employee WHERE Salary < 50000;**



##### ****Greater Than or Equal To (>=) Operator in Oracle****

The Greater than or Equal To (>=) Operator in Oracle is used to check whether the left-hand expression value is higher than or equals to the right-hand expression value or not. If the left-hand expression value is higher than or equals to the right-hand expression value then the condition becomes true and will return all the matched records.

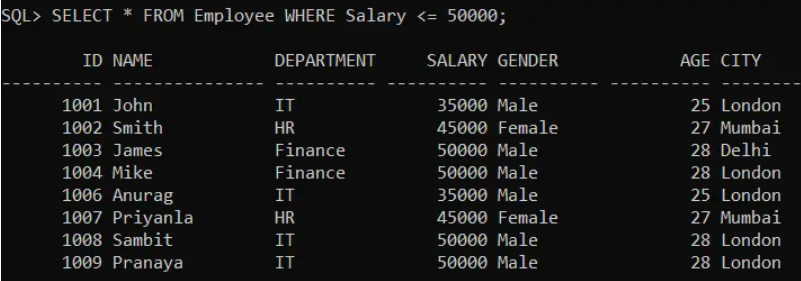
**SELECT \* FROM Employee WHERE Salary >= 50000;**



##### ****Less Than or Equal To (<=) Operator in Oracle****

The Less than or Equal To (<=) Operator in Oracle is used to check whether the left-hand expression value is lower than or equal to the right-hand expression value or not. If the left-hand expression value is lower than or equals to the right-hand expression value then the condition becomes true and it will return all the matching records.

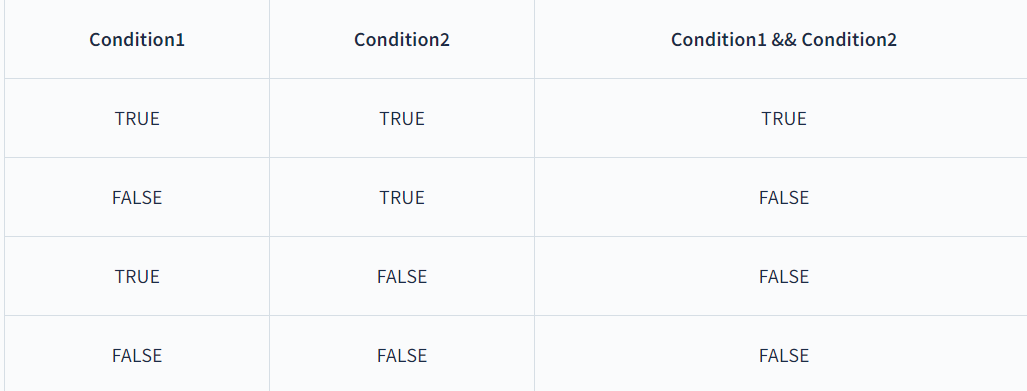
**SELECT \* FROM Employee WHERE Salary <= 50000;**



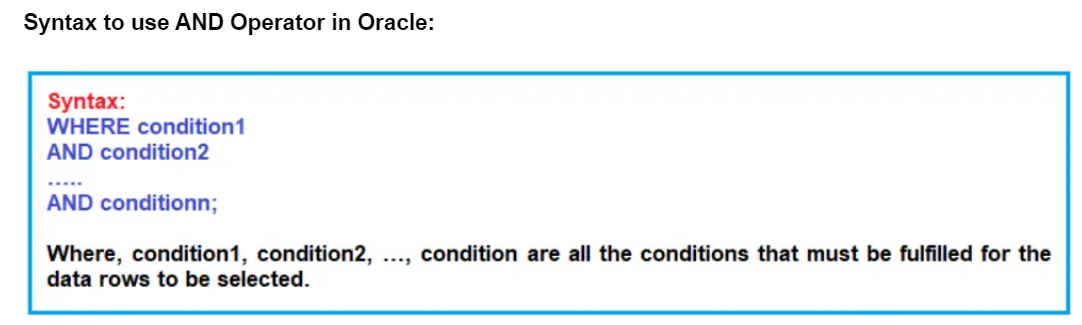
Logical Operators :

If you want to combine more than one condition, then you need to use the Logical Operators in Oracle. The Logical Operators in Oracle are basically used to check for the truth-ness of some conditions. Logical operators return a Boolean data type with a value of TRUE, or FALSE. In Oracle, there are three Logical Operators available.

Logical AND :

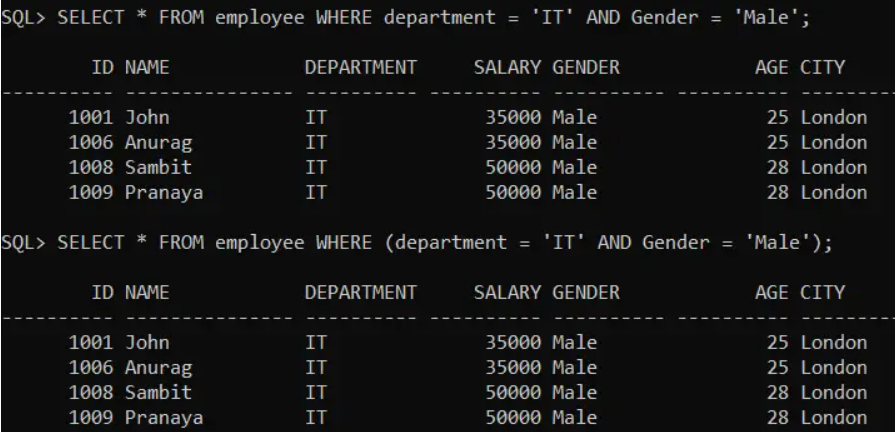


We can use AND condition with SELECT, INSERT, UPDATE or DELETE statements to test two or more conditions in an individual query.

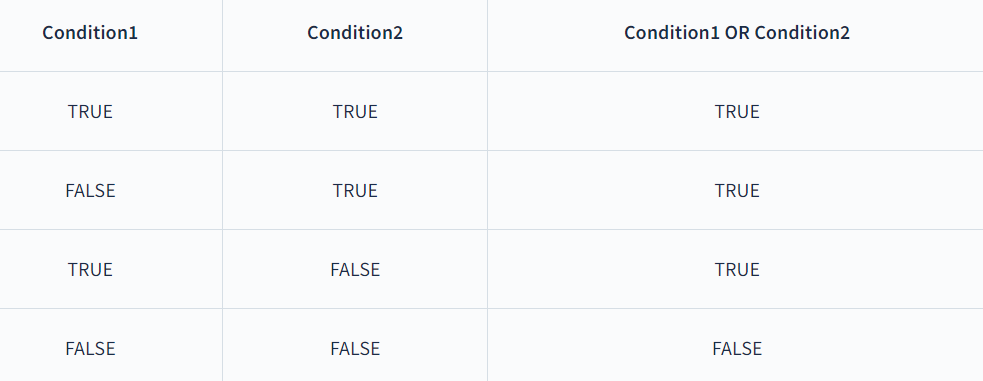


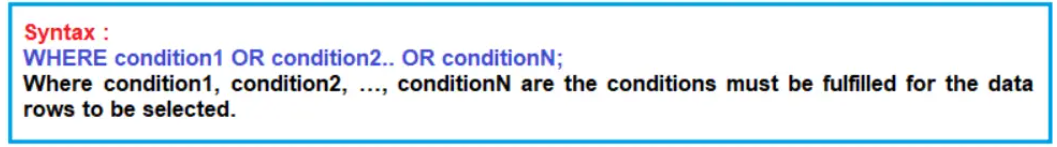
SELECT \* FROM employee WHERE (department = 'IT' AND Gender = 'Male'); --OR

SELECT \* FROM employee WHERE department = 'IT' AND Gender = 'Male'; -- Bracket is optional



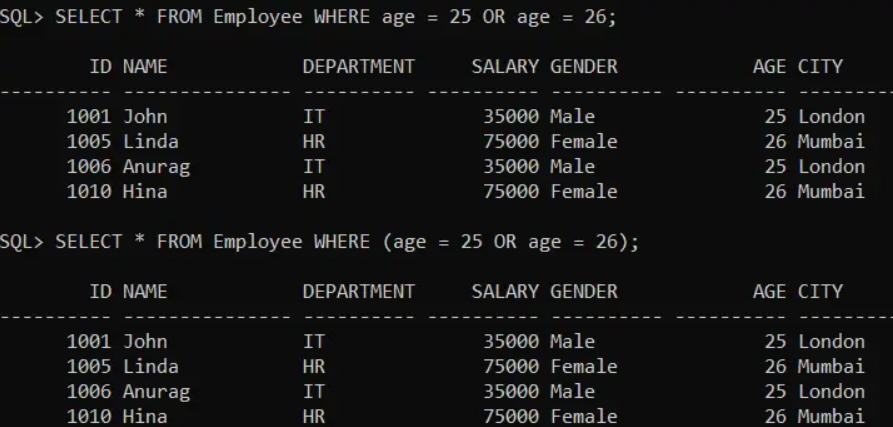
Logical OR :





SELECT \* FROM Employee WHERE age = 25 OR age = 26; -- OR

SELECT \* FROM Employee WHERE (age = 25 OR age = 26); -- Bracket is optional



##### ****Logical NOT****

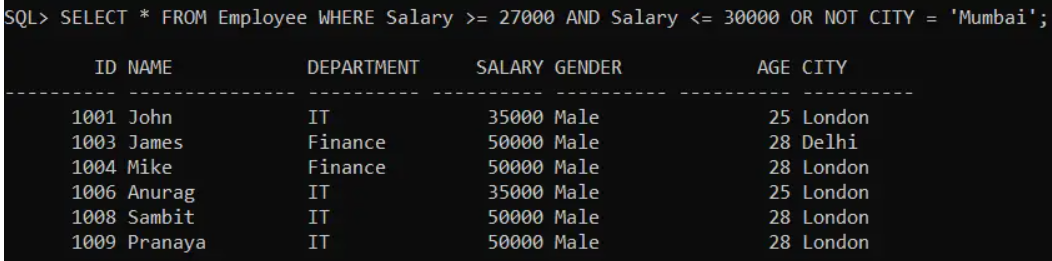
##### 

##### 

**Nested Logical Operators in Oracle:**

We can also use multiple logical operators in a single SQL statement in Oracle. When we combine the logical operators in a SELECT statement, the order in which the statement is processed is

1. **NOT**
2. **AND**
3. **OR**



Set operators :

**Types of SET Operators in Oracle:**

There are four types of SET Operators available in Oracle. They are as follows:

1. **UNION**: It Returns all distinct rows selected by either query
2. **UNION ALL:** It Returns all rows selected by either query, including all duplicates
3. **INTERSECT**: It Returns all distinct rows selected by both queries
4. **MINUS**: It Returns all distinct rows selected by the first query but not the second

You can combine multiple queries using the set operators UNION, UNION ALL, INTERSECT, and MINUS. All set operators have equal precedence. If a SQL statement contains multiple set operators, then Oracle Database evaluates them from the left to right unless parentheses explicitly specify another order.

SQL statements containing these set operators are referred to as compound queries, and each SELECT statement in a compound query is referred to as a component query. Two SELECTs can be combined into a compound query by a set operation only if they satisfy the following two conditions:

1. The result sets of both the queries must have the same number of columns.
2. The data type of each column in the second result set must match the data type of its corresponding column in the first result set.

The corresponding expressions in the select lists of the component queries of a compound query must match in number and must be in the same datatype group.

If component queries select character data, then the data type of the return values are determined as follows:

1. If both queries select values of datatype CHAR of equal length, then the returned values have datatype CHAR of that length. If the queries select values of CHAR with different lengths, then the returned value is VARCHAR2 with the length of the larger CHAR value.
2. If either or both of the queries select values of datatype VARCHAR2, then the returned values have datatype VARCHAR2.

Column names from the first query appear in the result.

**Advantage of SET operators in Oracle:**

1. Use a set operator to combine multiple queries into a single query
2. These operators are used to combine the information of similar data types from one or more than one table.

**Restrictions on the Set Operators:**

1. The ORDER BY clause doesn’t recognize the column names of the second SELECT
2. The set operators are not valid on columns of type BLOB, CLOB, BFILE, VARRAY, or nested table.
3. The UNION, INTERSECT, and MINUS operators are not valid on LONG columns.
4. Set operations are not allowed on SELECT statements containing TABLE collection expressions.
5. SELECT statements involved in set operations can’t use the FOR UPDATE clause.

##### ****Syntax of SET Operators in Oracle:**** **<Component Query>** **{UNION | UNION ALL | MINUS | INTERSECT}** **<Component Query>**

CREATE TABLE EmployeeUK

(

EmployeeId INT,

FirstName VARCHAR(20),

LastName VARCHAR(20),

Gender VARCHAR(10),

Department VARCHAR(20)

);

INSERT INTO EmployeeUK VALUES(1, 'Pranaya', 'Rout', 'Male','IT');

INSERT INTO EmployeeUK VALUES(2, 'Priyanka', 'Dewangan', 'Female','IT');

INSERT INTO EmployeeUK VALUES(3, 'Preety', 'Tiwary', 'Female','HR');

INSERT INTO EmployeeUK VALUES(4, 'Subrat', 'Sahoo', 'Male','HR');

INSERT INTO EmployeeUK VALUES(5, 'Anurag', 'Mohanty', 'Male','IT');

INSERT INTO EmployeeUK VALUES(6, 'Rajesh', 'Pradhan', 'Male','HR');

INSERT INTO EmployeeUK VALUES(7, 'Hina', 'Sharma', 'Female','IT');

CREATE TABLE EmployeeUSA

(

EmployeeId INT,

FirstName VARCHAR(20),

LastName VARCHAR(20),

Gender VARCHAR(10),

Department VARCHAR(20)

);

INSERT INTO EmployeeUSA VALUES(1, 'James', 'Pattrick', 'Male','IT');

INSERT INTO EmployeeUSA VALUES(2, 'Priyanka', 'Dewangan', 'Female','IT');

INSERT INTO EmployeeUSA VALUES(3, 'Sara', 'Taylor', 'Female','HR');

INSERT INTO EmployeeUSA VALUES(4, 'Subrat', 'Sahoo', 'Male','HR');

INSERT INTO EmployeeUSA VALUES(5, 'Sushanta', 'Jena', 'Male','HR');

INSERT INTO EmployeeUSA VALUES(6, 'Mahesh', 'Sindhey', 'Female','HR');

INSERT INTO EmployeeUSA VALUES(7, 'Hina', 'Sharma', 'Female','IT');

##### ****UNION Operator in Oracle****

The UNION operator is used to combine the result set of two or more SELECT statements into a single result set and then eliminates any duplicate rows from the final result set.

##### 

##### **Syntax :**

##### 

##### 

/\* check union with ORDER BY clause after going learning ORDER BY clause \*/

##### ****UNION ALL Operator in Oracle****

The UNION ALL operator is used to combine the result set of two or more SELECT statements into a single result including the duplicate values.

##### 

##### **Syntax:**

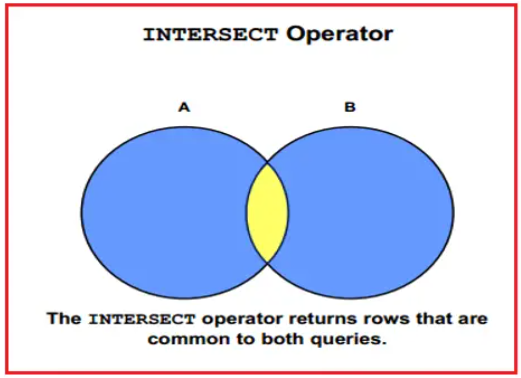
##### 

##### 

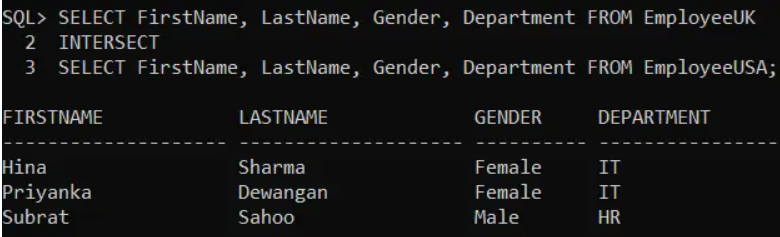
/\* check unionall with ORDER BY clause after going learning ORDER BY clause \*/

##### ****INTERSECT Operator in Oracle****

The INTERSECT operator in Oracle is used to combine two result sets and returns the data which are common in both the result set.







##### /\* after learning inner join go through the difference between inner join and intersect \*/

##### ****MINUS Operator in Oracle****

The MINUS operator in Oracle is used to return unique rows from the left query which isn’t present in the right query’s results.

##### 

##### 

##### 

/\* after learning order by learn its integration with Minus and difference between minus and not in \*/