Operators in Oracle

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

The different types of Operators available in Oracle SQL are:-

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

Arithmetic operators: -

- **The arithmetic operations can be used to create expressions on number and date data.**
- **The arithmetic operations can be Used to perform any Arithmetic Operations like Addition, subtraction, Multiplication and Divided by.**
- **The arithmetic operators can be used in any clause of a sql statement.**
- **Sql * plus ignores the blank spaces before and after the arithmetic operator .**

Example:-

Display salary of employees with 2000 increment in their salary.

```
Sql> SELECT ename,sal,sal + 2000 "Incremented salary" FROM emp;
```

Explination:-in emp table every employee salary sum it 2000.

Arithmetic Operator Subtraction (-):-

- **used to perform subtraction between two numbers and dates.**

Example:

Display the details of employees decreasing their salary by 200.

```
Sql> select ename,sal,sal-200 from emp;
```

Explination:-in emp table every employee salary subtracted with 200.

Arithmetic Operator Multiplication(*) :-Used to perform multiplication.

Example:-

Display the details of the employees Incrementing their salary two times.

Sql> SELECT sal * 2 FROM emp;

Explination:-every emp table salary is multiplied by 2.

Arithmetic Operator Division (/):-

Used to perform Division test. Division will display only the Quotient value not the remainder value. Example 6/2 gives 3 because 2 divides 6 by 3 times.

Example:-

Display half of the salary of employees.

Sql> SELECT sal, sal/2 FROM emp;

Examples:-

Sql> select empno,ename,sal,12*sal+100 from emp;

Sql> select empno,ename,sal,(12*sal)+100 from emp;

Sql> select empno,ename,sal,12*(sal+100) from emp;

Assignment operators:-

This operator is used for equality test. Used to test the equality of two operands.

Example:-

Display the details of Employees whose salary is equal to 2000.

Sql> SELECT *FROM emp WHERE sal=950;

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

---------- ---------- --------- ---------- --------- ---------- ---------- ----------

7900 JAMES CLERK 7698 03-DEC-81 950 30

Relational Operator Lessthan(<):-

This operator is used for less than test. Example a<b checks that operand 'a' is less than 'b' or not.

Example: Display the details of the employees whose salary is less than 3000.

Sql> SELECT * FROM emp WHERE sal < 3000;

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

---------- ---------- --------- ---------- --------- ---------- ---------- ----------

7369 SMITH CLERK 7902 17-DEC-80 800 20

7499 ALLEN SALESMAN 7698 20-FEB-81 1600 300 30

7521 WARD SALESMAN 7698 22-FEB-81 1250 500 30

7566 JONES MANAGER 7839 02-APR-81 2975 20

7654 MARTIN SALESMAN 7698 28-SEP-81 1250 1400 30

7698 BLAKE MANAGER 7839 01-MAY-81 2850 30

7782 CLARK MANAGER 7839 09-JUN-81 2450 10

7844 TURNER SALESMAN 7698 08-SEP-81 1500 0 30

7876 ADAMS CLERK 7788 23-MAY-87 1100 20

7900 JAMES CLERK 7698 03-DEC-81 950 30

7934 MILLER CLERK 7782 23-JAN-82 1300 10

Here if you observe we got a result values whose salary is less than the operand 3000.

Relational Operator Greater than(>):-

This operator is used for Greater than test. For example a>b checks the operand 'a' is greater than 'b' or not.

Example:

Display the details of Employees whose salary is greater than 3000

Sql> SELECT * FROM emp WHERE sal > 3000;

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

7839 KING PRESIDENT 17-NOV-81 5000 10

Relational Operator Less than or Equals to(<=):

This operator is used for Less than or Equal to test. For example a<=b, here checks whether operand 'a' is less than or equals to operand 'b'. If a<b then condition is true and if a=b then also condition is true but if a>b then condition is false.

Example :-

Display the details of Employees whose salary is less than or equal to 3000.

Sql> SELECT * FROM EMP WHERE sal <= 3000;

Relational Operator Greater than or Equals to(>=):

This operator is used to check the Greater than or equal test. For example a>=b checks the operand 'a' is greater than operand 'b' or operand 'a' is equals to the operand 'b'.

Example:-

Display the details of Employees whose salary is greater than or equal to 3000.

Sql> SELECT * FROM emp WHERE sal >= 3000;

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7902	FORD	ANALYST	7566	03-DEC-81	3000		20

<u>Relational Operator Not Equals to(!= or ^= or <>):</u>

This operator is used for inequality test.

Examples:

Display the details of employees whose salary is not equals to 2000.

Sql> SELECT * FROM emp WHERE sal != 3000;

Sql> SELECT * FROM emp WHERE sal ^= 2000;

Sql> SELECT * FROM emp WHERE sal <> 2000;

<u>Logical operators:-</u>

<u>AND operator:-</u>

Returns 'True' if both component conditions are true. Returns 'False' if any one component condition or Both Component conditions are False.

Example:-

Display the details of Employees whose salary is Greater than 1000 AND also whose salary is less than 2000.

Sql> SELECT *FROM emp WHERE sal > 1000 AND sal <2000;

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

7499 ALLEN SALESMAN 7698 20-FEB-81 1600 300 30

7521 WARD SALESMAN 7698 22-FEB-81 1250 500 30

7654 MARTIN SALESMAN 7698 28-SEP-81 1250 1400 30

7844 TURNER SALESMAN 7698 08-SEP-81 1500 0 30

7876 ADAMS CLERK 7788 23-MAY-87 1100 20

7934 MILLER CLERK 7782 23-JAN-82 1300 10

Sql> select ename,sal,job from emp

where (sal>=1500 and sal<=5000) and

job='MANAGER';

<u>The Logical OR operator:-</u>

- **Returns True if either component conditions become TRUE. Returns False if both the component conditions becomes False.**

Example:

Display the details of Employees whose salary is Greater than 1000 OR also whose salary is less than 2000.

Sql> SELECT *FROM emp WHERE sal> 1000 OR sal < 2000;

Explination:-whose salaries more than 1000 or less than 2000 that all emp table display.

SQL> select empno,ename,job,hiredate from emp

where job='MANAGER' or deptno=20;

sql> select empno,ename,job,hiredate from emp

where (job='MANAGER' or deptno=10);

sql> select empno,ename,job,hiredate from emp

where (job='CLERK' or job='SALESMAN' or job='ANALYST');

SQL> select empno,ename,job,hiredate from emp

where (sal<=2500 or sal>=5000) or job='MANAGER';

sql> select ename,job ,sal from emp

where job='CLERK' or job='MANAGER' and sal>1500;

<u>The Logical NOT operator:-</u>

The NOT operator returns 'True' if the condition is False and returns 'False' if the following condition is True.

Example:

Display the details of employees whose salary is Greater than or Equals to 3000.

Sql> SELECT * FROM emp WHERE sal < 3000;

Explination:-whose salary less than 3000 that salaries all are comming.

SQL> select empno,ename,job,sal from emp

where not ename='SMITH';

SQL> select empno,ename,job,sal from emp

where not sal>=5000;

sql> select empno,ename,job,sal,deptno from emp

where not job='CLERK' and deptno=20;

<u>Special operators:-</u>

<u>IN operator:-</u>

- **Returns true if value is available in given list of values**
- **Supports with all types of data (data types)**

In the below example only employees whose empno is (7125,7369,7782) are fetched.

Sql> SELECT *FROM emp WHERE empno IN (7125, 7369, 7782);

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

7369 SMITH CLERK 7902 17-DEC-80 800 20

7782 CLARK MANAGER 7839 09-JUN-81 2450 10

Inside DML statements:-

Sql> UPDATE emp SET sal=sal+200 WHERE ename IN ('SMITH','ALLEN','WARD');

Sql> DELETE FROM emp WHERE hiredate IN ('22-DEC-82','17-NOV-81');

Not in operator:-

not in' operator is quite opposite to 'IN' clause.

Sql> SELECT *FROM emp WHERE empno NOT IN (7125, 7369,7782);

Inside DML statements:-

Sql> UPDATE emp SET sal=sal+200 WHERE ename NOT IN

('SMITH','ALLEN','WARD');

Sql> DELETE FROM emp WHERE hiredate NOT IN ('22-DEC-82',' 17-NOV-81');

BETWEEN Operator:-

- **Returns true if value specified is within the specified range.**
- **Supports with numbers and date values.**
- **Returns all values from given range including source & destination values.**
- **Always apply on low value to high value.**

Example:- in this example all employee records are fetched whose salary is between 2000 and 3000

Sql> SELECT *FROM emp WHERE sal BETWEEN 2000 AND 3000;

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

7566 JONES MANAGER 7839 02-APR-81 2975 20

7698 BLAKE MANAGER 7839 01-MAY-81 2850 30

7782 CLARK MANAGER 7839 09-JUN-81 2450 10

7788 SCOTT ANALYST 7566 19-APR-87 3000 20

7902 FORD ANALYST 7566 03-DEC-81 3000 20

Whenever lower bound value is larger than upper bound then it shows 'no rows selected'

Example:-

Sql> SELECT *FROM emp WHERE sal BETWEEN 3000 AND 2000;

Output:

-- No rows selected

SQL> select ename,sal,job from emp

where job between 'MANAGER' and 'SALESMAN';

sql> select ename,sal,job,hiredate from emp

where hiredate between '17-DEC-81' and '20-JUN-83';

<u>Not between operator:-</u>

- **Returns true if value specified is not within the specified range.**
- **Supports with numbers and date values.**

- **Not between is an exclusive operator which eliminates range limits from Output.**

Example:-

Sql> SELECT *FROM emp WHERE sal NOT BETWEEN 2000 AND 3000;

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

7369 SMITH CLERK 7902 17-DEC-80 800 20

7499 ALLEN SALESMAN 7698 20-FEB-81 1600 300 30

7521 WARD SALESMAN 7698 22-FEB-81 1250 500 30

7654 MARTIN SALESMAN 7698 28-SEP-81 1250 1400 30

7839 KING PRESIDENT 17-NOV-81 500010

7844 TURNER SALESMAN 7698 08-SEP-81 1500 0 30

7876 ADAMS CLERK 7788 23-MAY-87 1100 20

7900 JAMES CLERK 7698 03-DEC-81 950 30

7934 MILLER CLERK 7782 23-JAN-82 1300 10

Note:-

Lower bound – 'value 'must be lower when compare to 'upper bound 'value

Upper bound- 'value' must be higher when compare to 'lower bound 'value

Sql> select ename,sal,job from emp

where job not between 'MANAGER' and 'SALESMAN';

sql> select ename,sal,job,hiredate from emp

where hiredate not between '17-DEC-81' and '20-JUN-83';

LIKE operator: -

Used to search for specific pattern in a given input.

% (percentage) and _ (underscore) are two wildcard characters.

% (percentage) represents "remaining group of characters "in the given input

_ (underscore) represents "one character" in given input.

Syntax:-

Select *From <tableName>

Where <character data type column> like '<value>';

Example:- Display the employees whose name is starting with 'S' in EMP table.

```
Sql> SELECT * FROM emp WHERE ename LIKE 'S%'

EMPNO ENAME      JOB              MGR HIREDATE         SAL  COMM DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
```

Display the employees whose name ends with 'S' in EMP table

```
Sql> SELECT * FROM emp WHERE ename LIKE '%S'

EMPNO ENAME      JOB              MGR HIREDATE         SAL  COMM DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
```

Display the employees whose names are having second letter as 'L' in EMP table

Sql> SELECT * FROM emp WHERE ename LIKE '_L%'

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10

Sql> select ename ,hiredate from emp where hiredate like '%JAN%';

Sql> select empno,ename,job from emp where job like '_____';

EX:

WHOSE EMPLOYEE NUMBER STARTS WITH 7 AND ENDS WITH 8 DIGIT?

SELECT * FROM EMP WHERE EMPNO LIKE '7%8';

EX:

WHO ARE JOININED IN THE YEAR 1981?

SELECT * FROM EMP WHERE HIREDATE LIKE '%81';

EX:

WHO ARE JOINED IN THE MONTH OF DEC?

SELECT * FROM EMP WHERE HIREDATE LIKE '%DEC%';

EX:

WHO ARE JOINED IN MONTH OF "FEB" , "MAY" ?

SELECT * FROM EMP WHERE HIREDATE LIKE '%FEB%' OR HIREDATE LIKE '%MAY%';

SQL> select empno,ename,job ,hiredate from emp where hiredate like '%-FEB-81';

Like operator with special char's:

EX: WHOSE EMPLOYEE NAME IS HAVING "#" CHAR?

SELECT * FROM EMP77 WHERE ENAME LIKE '%#%';

EX: WHOSE EMPLOYEE NAME IS HAVING "@" CHAR?

SELECT * FROM EMP77 WHERE ENAME LIKE '%@%';

EX: WHOSE EMPLOYEE NAME IS HAVING "_" CHAR?

SELECT * FROM EMP77 WHERE ENAME LIKE '%_%'ESCAPE '\';

EX: WHOSE EMPLOYEE NAME IS HAVING "%" CHAR?

SELECT * FROM EMP77 WHERE ENAME LIKE '%\%%'ESCAPE '\';

NOTE:

Generally oracle db server will treat these symbol(%, _) as "wildcard operators" but not "special char's". To overcome this problem we must use a pre-defined statement is "escape '\' " .

Not like operator: -

syntax:-

seleclt *from <table Name>

where <character data type column> not like '<value>';

Display the employees whose name is not ends with 'S' in EMP table?

Sql> SELECT *FROM emp WHERE ename NOT LIKE '%S';

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20

7499 ALLEN SALESMAN 7698 20-FEB-81 1600 300 30

7521 WARD SALESMAN 7698 22-FEB-81 1250 500 30

7654 MARTIN SALESMAN 7698 28-SEP-81 1250 1400 30

7698 BLAKE MANAGER 7839 01-MAY-81 2850 30

7782 CLARK MANAGER 7839 09-JUN-81 2450 10

7788 SCOTT ANALYST 7566 19-APR-87 3000 20

7839 KING PRESIDENT 17-NOV-81 5000 10

7844 TURNER SALESMAN 7698 08-SEP-81 1500 30

7902 FORD ANALYST 7566 03-DEC-81 3000 20

7934 MILLER CLERK 7782 23-JAN-82 1300 10

Display the employees whose names are not having second letter as 'L' in EMP table?

Sql> SELECT *FROM emp WHERE ename NOT LIKE '_L%';

Display the employees whose names are not start with 'S' in EMP table.?

Sql> SELECT *FROM emp WHERE ename NOT LIKE 'S%';

Sql> select ename ,hiredate from emp where hiredate not like '%JAN%';

Sql> select empno,ename,job from emp where ename not like '_O%';

Display the employees whose names are second letter start with 'R' from ending.?

Sql> SELECT *FROM emp WHERE ename LIKE '%R_';

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

7521 WARD SALESMAN 7698 22-FEB-81 1250 500 30

7782 CLARK MANAGER 7839 09-JUN-81 2450 10

7902 FORD ANALYST 7566 03-DEC-81 3000 20

Display the names in EMP table whose names having 'LL'.?

Sql> SELECT *FROM emp WHERE ename LIKE '%LL%';

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

7499 ALLEN SALESMAN 7698 20-FEB-81 1600 300 30

7934 MILLER CLERK 7782 23-JAN-82 1300 10

<u>Is null Operator:</u> Comparing nulls in a table.NULL is unknow (or) undefined value in database,NULL != 0 & NULL != space.so use " IS " keyword.

<u>syntax:</u>

where <column name> is null;

Ex:

Waq to display employee whose commission is NULL ?

SELECT * FROM EMP WHERE COMM IS NULL;

Ex:

Waq to display employee whose commission is NOT NULL ?

SELECT * FROM EMP WHERE COMM IS NOT NULL;

Ex:

Waq to display ename,job,sal,comm and also sal+comm from emp table whose ename is "SMITH" ?

SQL> SELECT ENAME,JOB,SAL,COMM,COMM+SAL FROM EMP WHERE ENAME='SMITH';

ENAME JOB SAL COMM COMM+SAL

--------- --------- --------- --------- --------------

SMITH CLERK 800

<u>Note:</u> If any arithmetic operator is performing some operation with NULL then it agian returns NULL only.

Ex: If x=1000;

i) x+null -------> 1000+0 -----------> 1000

ii) x-null -------> 1000-null ------------> null

iii) x*null ------> 1000*null ------------> null

iv) x/null --------> 1000/null ------------> null

- To overcome the above problem we should use a predefined function is called as "NVL()"

<u>Nvl(Exp1,Exp2):</u> Nvl stands for NULL VALUE. It is pre-defined function. is used to replace a user defined value in place of NULL in the expression. This function is having two arguments are Expression1 and Expression2.

> If Exp1 is null ------------> returns Exp2 value (user defined value)

> If Exp1 is not null -------> returns Exp1 only.

Ex:

SQL> SELECT NVL(NULL,0) FROM DUAL;

NVL(NULL,0)

0

SQL> SELECT NVL(NULL,100) FROM DUAL;

NVL(NULL,100)

100

SQL> SELECT NVL(0,100) FROM DUAL;

NVL(0,100)

0

SQL> SELECT ENAME, JOB, SAL, COMM, NVL (COMM,0)+SAL FROM EMP WHERE ENAME='SMITH';

ENAME JOB SAL COMM NVL(COMM,0)+SAL

---------- --------- ---------- ---------- ---------- ----------

SMITH CLERK 880 880

<u>Nvl2 (Exp1, Exp2, Exp3):</u> Pre-defined function which is an extension of NVL () having 3 arguments are Exp1, Exp2, Exp3.

- **If Exp1 is Null ----------> Exp3 value (User Defined value)**
- **If Exp1 is not null ------> Exp2 value (User Defined value)**

Ex: Waq to update all employee commissions in a table based on the following conditions?

i) IF employee comm is null then update those employees comm as 600.

ii) IF employee comm is not null then update those employees comm as comm+500.

Sol:- UPDATE EMP SET COMM=NVL2 (COMM, COMM+500,600);

<u>Set Operators:</u>

SQL set operators allows combine results from two or more SELECT statements. At first sight this looks similar to SQL joins although there is big difference. SQL joins tends to combine columns i.e. with each additionally joined table it is possible to select more and more columns. SQL set operators on the other hand combine rows from different queries with strong preconditions - all involved SELECTS must. Joins we are collecting the data from two tables when there is a common data. But in set operators the data is not joined, in this the data is merged

• Retrieve the same number of columns and

● The data types of corresponding columns in each involved SELECT must be compatible (either the same or with possibility implicitly convert to the data types of the first SELECT statement).

Operator	Returns
UNION	All distinct rows selected by either query
UNION ALL	All rows selected by either query, including all duplicates
INTERSECT	All distinct rows selected by both queries
MINUS	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.

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 datatype of the return values are determined as follows:

● 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 CHARvalue.

● If either or both of the queries select values of datatype VARCHAR2, then the returned values have datatype VARCHAR2.

In queries using set operators, Oracle does not perform implicit conversion across datatype groups. Therefore, if the corresponding expressions of

component queries resolve to both character data and numeric data, Oracle returns an error.

Set Operator Guidelines:-

- **The expressions in the SELECT lists must match in number and data type.**
- **Parentheses can be used to alter the sequence of execution.**
- **The ORDER BY clause:**
  - **Can appear only at the very end of the statement**
  - **Will accept the column name, aliases from the first SELECT statement, or the positional notation**
- **Column names from the first query appear in the result.**

Advantage of set operator:-

- **Use a set operator to combine multiple queries into a single query**
- **These operators are used to combine the information of similar data type from One or more than one table.**

Restrictions on the Set Operators:-

The set operators are subject to the following restrictions:

- **The ORDER BY clause doesn't recognize the column names of the second SELECT**
- **The set operators are not valid on columns of type BLOB, CLOB, BFILE, VARRAY, ornested table.**
- **The UNION, INTERSECT, and MINUS operators are not valid on LONG columns.**
- **Set operations are not allowed on SELECT statements containing TABLE collection expressions.**
- **SELECT statements involved in set operations can't use the FOR UPDATE clause.**

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 datatype of each column in the second result set must match the datatype of its corresponding column in the first result set.

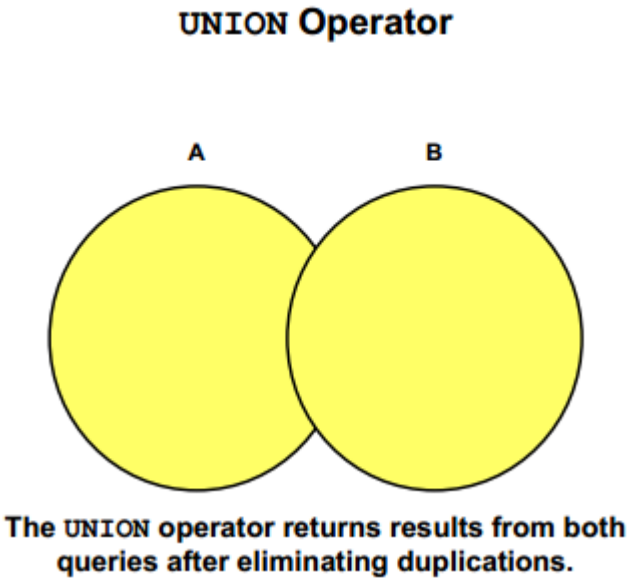
The generic syntax of a query involving a set operation is:

<component query>

{UNION | UNION ALL | MINUS | INTERSECT}

<component query>

- **UNION:- UNION Operator combines the results of two select statements into one result set, and then eliminates any duplicates rows from the final result set.**



Example:-

- **Sql> select empno,ename from emp where deptno=20**
- **union**

• select empno,ename from emp where deptno=30 order by 1;

Explination:- The above statement combines the results of two queries with the UNION operator, which eliminates duplicate selected rows. This statement shows that you must match datatype (using the TO_CHAR function) when columns do not exist in one or the other table:

Sql> select empno,ename,job from emp

where deptno=(select deptno from dept

Where dname='SALES')

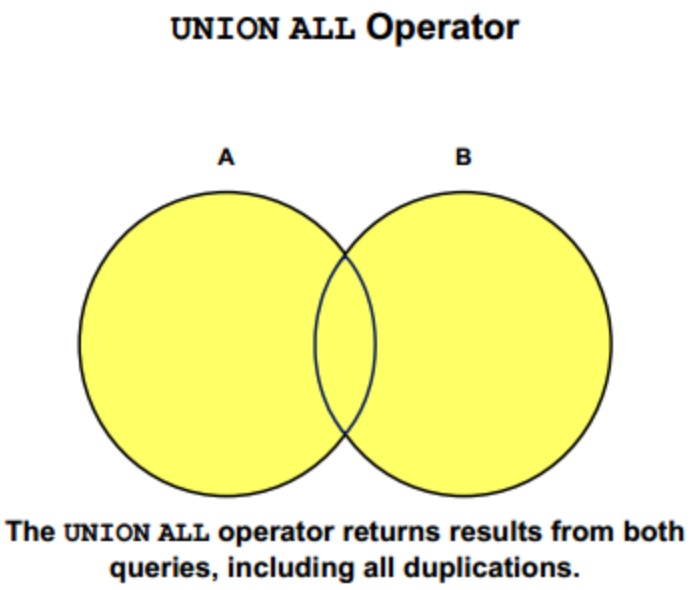
union

select empno,ename,job from emp

where deptno=(select deptno from dept where dname='ACCOUNTING') order by 1;

Union All:-

UNION ALL Operator combines the results of two select statements into one result set including Duplicates.



Example:-

Sql> select empno,ename from emp where deptno=10

union all

select empno,ename from emp where deptno=30

order by 1;

Explination:- The UNION operator returns only distinct rows that appear in either result, while the UNION ALL operator returns all rows. The UNION ALL operator does not eliminate duplicate selected rows:

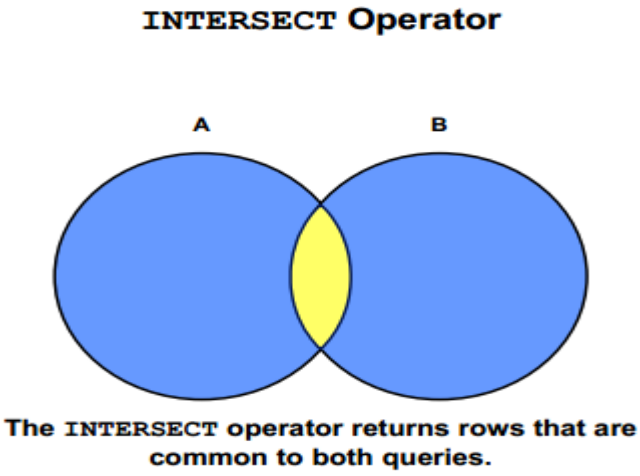
sql> select job from emp where deptno=20

union all

select job from emp where deptno=30;

INTERSECT:-

INTERSECT Operator returns only those rows that are common in both tables.



Sql> select empno,ename from emp where deptno=10

intersect

select empno,ename from emp where deptno=30

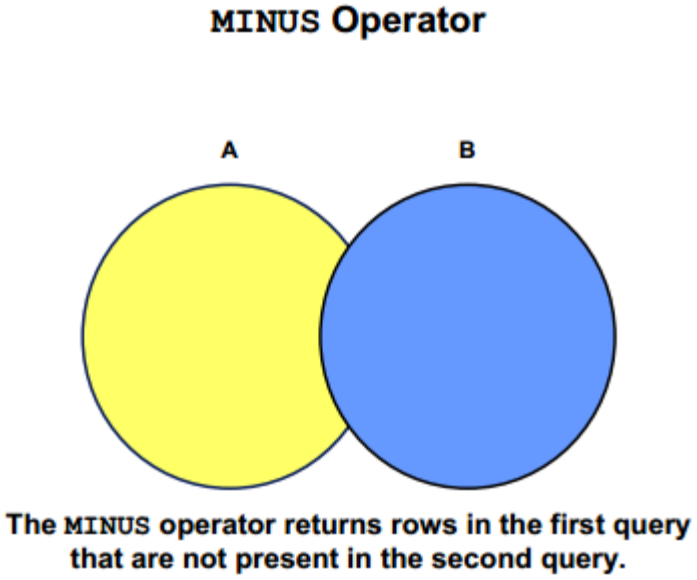
order by 1;

Explination:- The above statement combines the results with the INTERSECT operator, which returns only those rows returned by both queries.

```
Sql> select job from emp where deptno=20
        intersect
        select job from emp where deptno=30;
```

MINUS:-

MINUS Operator takes the result set of first select statement and removes those rows that are returned by a second select statement.



Example:-

```
Sql> select empno,ename from emp
        Where    deptno=10
        minus
        select empno,ename from emp
        where deptno=30 order by 1;
```

Explination:- The above statement combines results with the MINUS operator, which returns only unique rows returned by the first query but not by the second:

```
Sql>select job from emp where deptno=20

        minus

        select job from emp where deptno=30;
```

Using Set Operations to Compare Two Tables:-

Developers, and even DBAs, occasionally need to compare the contents of two tables to determine whether the tables contain the same data. The need to do this is especially common in test environments, as developers may want to compare a set of data generated by a program under test with a set of "known good" data. Comparison of tables is also useful for automated testing purposes, when we have to compare actual results with a given set of expected results. SQL's set operations provide an interesting solution to this problem of comparing two tables.

The following query uses both MINUS and UNION ALL to compare two tables for equality. The query depends on each table having either a primary key or at least one unique index.

Example:-

```
Sql>DESC CUSTOMER_KNOWN_GOOD

Name                    Null?    Type

------------------------- -------- ----------------

CUST_NBR                  NOT NULL NUMBER(5)

NAME                    NOT NULL VARCHAR2(30)
```

```
SELECT * FROM CUSTOMER_KNOWN_GOOD;


  CUST_NBR NAME

---------- ----------------------------

       1 Sony

       1 Sony
```

2 Samsung

3 Panasonic

3 Panasonic

3 Panasonic

Sql>DESC CUSTOMER_TEST

Name Null? Type

----------- -------- ----------

CUST_NBR NOT NULL NUMBER(5)

NAME NOT NULL VARCHAR2(30)

Sql>SELECT * FROM CUSTOMER_TEST;

CUST_NBR NAME

---------- ----------

1 Sony

1 Sony

2 Samsung

2 Samsung

3 Panasonic

As we can see the CUSTOMER_KNOWN_GOOD and CUSTOMER_TEST tables have the same structure, but different data. Also notice that none of these tables has a primary or unique key; there are duplicate records in both. The following SQL will compare these two tables effectively.

Example:-

```
Sql>(SELECT * FROM CUSTOMER_KNOWN_GOOD
    MINUS
   SELECT * FROM CUSTOMER_TEST)
```

```
UNION ALL
(SELECT * FROM CUSTOMER_TEST
MINUS
SELECT * FROM CUSTOMER_KNOWN_GOOD);
```

Explination:-Let's talk a bit about how this query works. We can look at it as the union of two compound queries. The parentheses ensure that both MINUS operations take place first before the UNION ALL operation is performed. The result of the first MINUS query will be those rows in CUSTOMER_KNOWN_GOOD that are not also in CUSTOMER_TEST. The result of the second MINUS query will be those rows in CUSTOMER_TEST that are not also in CUSTOMER_KNOWN_GOOD. The UNION ALL operator simply combines these two result sets for convenience. If no rows are returned by this query, then we know that both tables have identical rows. Any rows returned by this query represent differences between the CUSTOMER_TEST and CUSTOMER_KNOWN_GOOD tables.

```
Sql>(SELECT C1.*, COUNT(*)
FROM CUSTOMER_KNOWN_GOOD C1
GROUP BY C1.CUST_NBR, C1.NAME
MINUS
SELECT C2.*, COUNT(*)
FROM CUSTOMER_TEST C2
GROUP BY C2.CUST_NBR, C2.NAME)
UNION ALL
(SELECT C3.*, COUNT(*)
FROM CUSTOMER_TEST C3
GROUP BY C3.CUST_NBR, C3.NAME
MINUS
SELECT C4.*, COUNT(*)
FROM CUSTOMER_KNOWN_GOOD C4
```

GROUP BY C4.CUST_NBR, C4.NAME);

Explination:- These results indicate that one table (CUSTOMER_KNOWN_GOOD) has one record for "Samsung", whereas the second table (CUSTOMER_TEST) has two records for the same customer. Also, one table (CUSTOMER_KNOWN_GOOD) has three records for "Panasonic", whereas the second table (CUSTOMER_TEST) has one record for the same customer. Both the tables have the same number of rows (two) for "Sony", and therefore "Sony" doesn't appear in the output.

TIP: Duplicate rows are not possible in tables that have a primary key or at least one unique index. Use the short form of the table comparison query for such tables.

Type conversion functions:-

If corresponding expressions does not belongs to same data type also we can restrictive data from multiple queries ,in this case we use appropriate type conversion function.

Example:-

```
Sql> select ename "name",to_number(null) "deptno" from emp

                union

    select to_char(null),deptno from dept;



sql> select empno,ename,sal,to_number(null) from emp

          where deptno=20

      union

  select empno,ename,to_number(null),hiredate from emp

          where deptno=30;
```

Using NULLs in Compound Queries

We discussed union compatibility conditions at the beginning of this chapter. The union compatibility issue gets interesting when NULLs are involved. As we know, NULL doesn't have a datatype, and NULL can be used in place of a value of any datatype. If we purposely select NULL as a

column value in a component query, Oracle no longer has two datatypes to compare in order to see whether the two component queries are compatible. For character columns, this is no problem. For example:

```
Sql>SELECT 1 NUM, 'DEFINITE' STRING FROM DUAL

                UNION

          SELECT 2 NUM, NULL STRING FROM DUAL;

        NUM STRING

---------- --------

         1 DEFINITE

         2
```

Notice that Oracle considers the character string 'DEFINITE' from the first component query to be compatible with the NULL value supplied for the corresponding column in the second component qery. However, if a NUMBER or a DATE column of a component query is set to NULL, we must explicitly tell Oracle what "flavor" of NULL to use. Otherwise, we'll encounter errors. For

example:

```
sql>SELECT 1 NUM, 'DEFINITE' STRING FROM DUAL

          UNION

       SELECT NULL NUM, 'UNKNOWN' STRING FROM DUAL;
```

ERROR at line 1:

ORA-01790: expression must have same datatype as corresponding expression

Note that the use of NULL in the second component query causes a datatype mismatch between the first column of the first component query, and the first column of the second component query. Using NULL for a DATE column causes the same problem, as in the following

Example:-

```
          sql>SELECT 1 NUM, SYSDATE DATES FROM DUAL

                UNION
```

SELECT 2 NUM, NULL DATES FROM DUAL;

SELECT 1 NUM, SYSDATE DATES FROM DUAL

ERROR at line 1:

ORA-01790: expression must have same datatype as corresponding expression

In these cases, we need to cast the NULL to a suitable datatype to fix the problem, as in the following examples:-

Sql>SELECT 1 NUM, 'DEFINITE' STRING FROM DUAL

UNION

SELECT TO_NUMBER(NULL) NUM, 'UNKNOWN' STRING FROM DUAL;

NUM STRING

---------- --------

1 DEFINITE

UNKNOWN

Sql>SELECT 1 NUM, SYSDATE DATES FROM DUAL

UNION

SELECT 2 NUM, TO_DATE(NULL) DATES FROM DUAL;

NUM DATES

---------- ---------

1 06-JAN-02

2

This problem of union compatibility when using NULLs is encountered in Oracle8i. However, there is no such problem in Oracle9i, as we can see in the following examples generated from an Oracle9i database.

Sql>SELECT 1 NUM, 'DEFINITE' STRING FROM DUAL

UNION

SELECT NULL NUM, 'UNKNOWN' STRING FROM DUAL;

NUM STRING

---------- --------

1 DEFINITE

UNKNOWN

Sql>SELECT 1 NUM, SYSDATE DATES FROM DUAL

UNION

SELECT 2 NUM, NULL DATES FROM DUAL;

NUM DATES

---------- ---------

1 06-JAN-02

2

EX:-

- **sql>select job from emp where deptno=20**

  union

  select job from emp where deptno=30;

EX:-

SQL> SELECT * FROM EMP_HYD;

EID ENAME SAL

---------- ---------- ----------

1021 SAI 85000

1022 JONES 48000

1023 ALLEN 35000

SQL> SELECT * FROM EMP_CHENNAI;

EID	ENAME	SAL
1021	SAI	85000
1024	WARS	36000
1025	MILLER	28000

EX: Waq to display all employee details who are working in NARESHIT organization?

SELECT * FROM EMP_HYD UNION SELECT * FROM EMP_CHENNAI;

(OR)

SELECT * FROM EMP_HYD UNION ALL SELECT * FROM EMP_CHENNAI;

EX: Waq to display employee details who are working in both branchs ?

SELECT * FROM EMP_HYD INTERSECT SELECT * FROM EMP_CHENNAI;

EX: Waq to display employee details who are working in HYD BUT NOT IN CHENNAI branchs ?

SELECT * FROM EMP_HYD MINUS SELECT * FROM EMP_CHENNAI;

EX: Waq to display employee details who are working in CHENNAI BUT NOT IN HYD branchs ?

SELECT * FROM EMP_CHENNAI MINUS SELECT * FROM EMP_HYD;