UNION CLAUSE

The SQL UNION clause/operator is used to combine the results of two or more SELECT statements without returning any duplicate rows.

To use UNION, each SELECT must have the same number of columns selected, the same number of column expressions, the same data type, and have them in the same order but they do not have to be the same length.

```
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
 [WHERE condition]

UNION

SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
```

UNION ALL CLAUSE

The SQL UNION ALL clause/operator is used to combine the results of two SELECT statements ***including*** duplicate rows.

The same rules that apply to UNION apply to the UNION ALL operator.

```
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
 [WHERE condition]

UNION ALL

SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
```

INTERSECT CLAUSE

The SQL INTERSECT clause/operator is used to combine two SELECT statements, but returns rows only from the first SELECT statement that are identical to a row in the second SELECT statement. This means INTERSECT returns only common rows returned by the two SELECT statements.

The same rules that apply to UNION apply to the INTERSECT operator.

```
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
 [WHERE condition]

INTERSECT

SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
```

EXCEPT CLAUSE

The SQL EXCEPT clause/operator is used to combine two SELECT statements and returns rows from the first SELECT statement that are not returned by the second SELECT statement. This means EXCEPT returns only rows which are not available in second SELECT statement.

The same rules that apply to UNION apply to the EXCEPT operator.

```
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
 [WHERE condition]

EXCEPT

SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
```

ASSIGNMENT

ASSIGNMENT – 5

IN THE EMP TABLE DISPLAY :

1) EID NAME CITY DOJ DEPT DESI SALARY OF THE DELHI EMPLOYEES

2) DETAILS OF ALL THE EMPLOYEES WHOSE SALARY DETAILS ARE NOT AVAILABLE.

IN THE INVENTORY STRUCTURE DISPLAY :

1) PID, PDESC, CATEGORY, SNAME, SCITY

2) DISPLAY OID , ODATE , CNAME, CADDRESS, CPHONE, PDESC, PRICE,OQTY, AMT

INDEXES

Indexes

Indexes are special lookup tables that the database search engine can use to speed up data retrieval.

An index helps speed up SELECT queries and WHERE clauses, but it slows down data input, with UPDATE and INSERT statements. Indexes can be created or dropped with no effect on the data.

The CREATE INDEX Command:

```
CREATE INDEX index_EID ON table_EID (column_EID);
```

Composite Indexes:

```
CREATE INDEX index_EID on table_EID (column1, column2);
```

Implicit Indexes: Implicit indexes are indexes that are automatically created by the database server when an object is created. Indexes are automatically created for primary key constraints and unique constraints.

DROP INDEX Command:

```
DROP INDEX index_EID ON table_EID;
```

SQL VIEWS

VIEWS

A view is nothing more than a SQL statement that is stored in the database with an associated ID.

A view can contain all rows of a table or select rows from a table. A view can be created from one or many tables which depends on the written SQL query to create a view.

Views which are kind of virtual tables, allow users to do the following:

- Structure data in a way that users or classes of users find natural or intuitive.
- Restrict access to the data such that a user can see and (sometimes) modify exactly what they need and no more.
- Summarize data from various tables which can be used to generate reports.

VIEWS

CREATE VIEW view_EID AS

(SELECT column1, column2.....

FROM table_EID

WHERE [condition]

);

The WITH CHECK OPTION:

The WITH CHECK OPTION is a CREATE VIEW statement option. The purpose of the WITH CHECK OPTION is to ensure that all UPDATE and INSERTs satisfy the condition(s) in the view definition.

CREATE VIEW view_EID AS

SELECT column1, column2.....

FROM table_EID

WHERE [condition]

WITH CHECK OPTION

;

Updating a Views

A view can be updated under certain conditions:

- The SELECT clause may not contain the keyword DISTINCT.
- The SELECT clause may not contain summary functions.
- The SELECT clause may not contain set operators.
- The FROM clause may not contain multiple tables.
- The query may not contain GROUP BY or HAVING.
- Calculated columns may not be updated.
- All NOT NULL columns from the base table must be included in the view in order for the INSERT query to function.
- The SELECT clause may not contain an ORDER BY clause.

Dropping Views

DROP VIEW view_EID;

SQL HAVING CLAUSE

SQL HAVING CLAUSE

The HAVING clause enables we to specify conditions that filter which group results appear in the final results

The WHERE clause places conditions on the selected columns, whereas the HAVING clause places conditions on groups created by the GROUP BY clause

The following is the position of the HAVING clause in a query

SELECT

FROM

WHERE

GROUP BY

HAVING

ORDER BY

SQL HAVING CLAUSE

The HAVING clause must follow the GROUP BY clause in a query and must also precede the ORDER BY clause if used.

The following is the syntax of the SELECT statement, including the HAVING clause:

```
SELECT column1, column2
FROM table1, table2
WHERE [ conditions ]
GROUP BY column1, column2
HAVING [ conditions ]
ORDER BY column1, column2
```

e to Think!!

Create a View for the below queries:

From the employee salary table, we need to see the total salary as “TOTAL COST” for each department arranged in the descending order of total salary .

Also just show only those departments where “TOTAL COST” is greater than 50000.

ASSIGNMENT

ASSIGNMENT – 6

1) CREATE A VIEW *EMP_SAL_DETAILS* TO GET EID NAME DOJ DEPT DESI SALARY AS BASIC. ALSO CALCULATE HRA (15% OF BASIC), PF (9% OF BASIC), NET(BASIC+HRA+PF), GROSS(NET-PF).

2) CREATE A VIEW TO DISPLAY EID,NAME, DOJ, DESI, DEPT OF ALL THE MANAGERS JOINED IN 2019.

3) CREATE A VIEW TO HOW MANY TEAM MEMBERS ARE THERE IN EACH DEPARTMENTS IN EACH CITY, ALONG WITH THERE TOTAL & AVERAGE SALARY.

4) IN THE INVENTORY STRUCTURE GENERATE A VIEW *BILL*. IT SHOULD DISPLAY OID,ODATE,CNAME,ADDRESS,PHONE,PDESC, PRICE, OQTY, AMOUNT

SQL FUNCTIONS

SQL FUNCTIONS

•**COUNT** Function - The SQL Server COUNT aggregate function is used to count the number of rows in a database table.

•**MAX** Function - The SQL Server MAX aggregate function allows to select the highest (maximum) value for a certain column.

•**MIN** Function - The SQL Server MIN aggregate function allows to select the lowest (minimum) value for a certain column.

•**AVG** Function - The SQL Server AVG aggregate function selects the average value for certain table column.

•**SUM** Function - The SQL Server SUM aggregate function allows selecting the total for a numeric column.

•**SQRT** Function - This is used to generate a square root of a given number.

•**RAND** Function - This is used to generate a random number using SQL command.

•**CONCAT** Function - This is used to concatenate multiple parameters to a single parameter.

•**RANK** Function - This is used to assign the rank to each row

•select RANK() over (ORDER BY MARKS DESC) 'POSITION', * from STU_MARKS;

•**DENSE_RANK** Function - This is used to assign the rank to each row without skipping the rank

•select DENSE_RANK() over (ORDER BY MARKS DESC) 'POSITION', * from STU_MARKS;

•**ROW_NUMBER** Function - This is used to add the row no to each row.

•select row_number() over (ORDER BY MARKS DESC) 'ROWNO', * from STU_MARKS;

SQL STRING FUNCTIONS

- **ASCII()** Ascii code value will come as output for a character expression..
- **CHAR()** Character will come as output for given Ascii code or integer.
- **CHARINDEX()** Starting position for given search expression will come as output in a given string expression. EX: Select CHARINDEX('G', 'KING')
- **LEFT()** Left part of the given string till the specified number of characters
- **RIGHT()** Right part of the given string till the specified number of characters.

LEN() Number of characters will come as output for a given string expression

LOWER() Lowercase string will come as output for a given string data.

UPPER() Uppercase string will come as output for a given string data.

SQL STRING FUNCTIONS

SUBSTRING() Part of a string based on the start position value and length value.

Ex: Select SUBSTRING ('WORLD', 1,3)

REPLACE() String expression will come as output for a given string data after replacing all occurrences of specified character with specified character.

Ex: Select REPLACE('INDIA', 'I', 'K')

REVERSE() Reverse string expression will come as output for a given string data

STUFF() String expression will come as output for a given string data after replacing from starting character till the specified length with specified character.

Ex Select STUFF('ABCDEFGH', 2,4,'IJK')