Date

Practical NO. 1

* Aim :- Write SQL queries to implement DDL commands.

* Theory :-

• DDL (Data Definition Language)

→ It is used to defines and modify databases structures.

① CREATE TABLE :

→ This command creates a new table in database.

```
CREATE TABLE employees (
    employee_id INT PRIMARY KEY,
    first_name VARCHAR (50),
    last_name VARCHAR (50),
    birth_date DATE,
    hire_date DATE,
    Salary DECIMAL (10, 2)
);
```

② ALTER TABLE : (ADD)

→ This command is used alter the structure of table.

```
ALTER TABLE employees
ADD email VARCHAR (100);
```

④ ALTER TABLE (MODIFY):

→ This command modifies an existing column in table.

```
ALTER TABLE employees
MODIFY COLUMN salary DECIMAL(12,2);
```

④ ALTER TABLE (DROP):

→ This commands drop an existing column from tables.

```
ALTER TABLE employees
DROP COLUMN birth_date;
```

⑤ DROP_TABLE :

→ This command deletes an entires table and All it's data.

```
DROP TABLE employees;
```

⑥ TRUNCATE TABLE :-

→ This command removes all rows from table resetting to initial state but without removing table structure itself.

```
TRUNCATE TABLE employees;
```

Date

⑤ RENAME :

→ This command changes name of database object such as table or column.

RENAME TABLE old_table_name TO new_table_name ;

ALTER TABLE table_name
RENAME COLUMN old_col_name TO new_col_name ;

* Result :

→ The SQL queries to implement DDL commands executed successfully.

By

Practical No. 2

* Aim :- Write SQL queries to implement DML commands.

* Theory :

• DML (Data Manipulation Language):

→ It is used for managing data with database objects.

① INSERT :

```
INSERT INTO employees (employee_id, first_name, last_name, birth_date, hire_date, Salary)
VALUES (1, 'John', 'DOE', '1980-01-05', '2023-03-01', 60000.0);
```

→ This command adds new rows of data to tables.

② UPDATE :

→ This command modifies existing data in table.

```
UPDATE employees
SET Salary = 65000.0
WHERE employee_id = 1;
```

(3) DELETE:

→ This command removes existing rows from table.

```
DELETE FROM employees
WHERE employee_id = 1;
```

(4) SELECT:

→ This command retrieves data from table.

```
SELECT employee_id, first_name, last_name,
salary FROM employees
WHERE salary > 50000.00;
```

* Result:

→ The SQL queries to implement DML commands executed successfully.

Practical No. 3

* Aim : Write SQL queries to implement DCL commands.

* Theory :

• DCL (Data control language) :

→ It is used to control access to data in database.

① GRANT :

→ This command gives privileges to users.

```
GRANT SELECT, INSERT, UPDATE ON
employes To 'username';
```

② REVOKE :

→ This command removes privileges from users.

```
REVOKE, INSERT, UPDATE ON employees
FROM 'username';
```

③ CREATE USER :

→ This command create new user in database.

```
CREATE USER 'new_user' @ 'localhost'
IDENTIFIED BY 'password';
```

Date :

④ DROP USER :

→ This command deletes user from database.

DROP USER 'username' @ 'localhost';

* Result :

→ The SQL queries to implement DCL commands executed successfully.

Date

Practical No. 4.

* Aim:- Write SQL queries to implement key constraints.

* Theory:

The SQL constraints are used to specify user for the data in a table.
This ensures the accurance and reliability of the data in the table.
of there is any violation bet^ the contraint and the data action the action is aborted. In the key constraints there are two types of keys:

① Primary key
② Unique key.

The primary key the not null and unique values.
The unique key constraints the null or unique value.

The following constraints are commonly used in SQL.

- Not Null:- Ensures that a column cannot have a null values.

- Unique :- Ensures that all values in a column different.

- primary key :- A combination of not null and unique.
- foreign key :- Prevents action that would destroy link bet^n tables.
- check :- Ensures that values in a column satisfies a specific conditions.
- Default :- Sets a default values for a column if no values is specified.
- Syntax :-

  Create table tablename (c1 datatype contraints, colum2 datatype contraints, ...);
- Examples :-

```
Create table employee (
employee_id int primary key,
email varchar2(100) unique,
first_name varchar2(40) Not Null,
last_name varchar2(50) Not Null,
Salary decimal (10 2));
```

* Result :- The command has been executed successfully.

Practical NO. 5

* Aim : Write SQL queries using AGGREGATE functions.

* Theory :

An agregate function is a function that performs a calculation on a set of values, and returns a single value.

Aggregate functions are often used with the 'GROUP BY' clause of the 'SELECT' statement. The 'GROUP BY' clause splits the result-set into groups of values & aggregate funct" can be used to return a single value for each group.

The most commonly used SQL aggregate functions are :

- MIN () - return the smallest value within the selected column.

- MAX () - returns the largest value within the selected column.

- COUNT() - returns the largest number of rows in a set.

- SUM () - returns the total sum of a numerical column.

* AVG() - returns the average value of a numerical column.

- VAR_POP() / (Population Variance) : Calculate the the variance for entire populations of set of rules.

- VAR_SAMP() / (Sample Variance) : Calculate the variances for a sample of a set of values.

- STDDEV_POP() / (Population standard Deviation) :

Calculateds standard deviation for the entire population of a set of values.

- 'STDDEV_SAMP()/(Sample standard Deviation)

Calculate the standard deviation for a sample of a set of values.

- 'MEDIAN()' (supported in oracle) :

Calculates the Median values of a set of numbers.

- STDDEV() : standard deviation.

* Result : Hence, the SQL queries of aggregate function are executed successfully.

Date

Practical NO. 6

* Aim :- Write SQL queries to SELECT data using SET UNION, INTERSECTION and MINUS operation

* Theory :-

The SQL set operation is used to combine the two or more SQL select statements

types of set operations.

- union
- union All
- Intersect
- Minus

① Union :- The SQL union operations is used to combine the result of two or more SQL select queries.

The union operation diminates the duplicate rows from its result set.

Syntax :- Select column 1, column 2 ... from table 1, union select column 1, column 2 ... from table 2.

② Union All :- Union All operation is equal to the union operation. It returns the set without removing duplicate & sorting the data.

Syntax :- Select column 1, column 2, from table
Union All Select column 1, column 2,
... from tables.

③ Intersect :- It is used to combine two Select statement. The intersect operations return the common row from both the Select statements.

Syntax :- Select column_name from table 1
intersect.
Select column_name
from table 2;

④ Minus :- It combines the result of two select statements, minus operator is used to display the rows which are present in the first query but absent in two second query.

Syntax :- select column-name from
table, minus select column-name
from table 2.

* Result :- The command has been executed Successfully.

Date

Practical No. 7.

* Aim :- Write a SQL queries to implement operators Between & In.

* Theory :-

① In ()
② Not In ()
③ Between ()
④ NOT Between ()

- In () → The 'In' operator allows you to specify multiple values in a where clauses.

Syntax: select * from table_name
where column_name In
(values, values);

- Not In () → The 'Not In' operator is used in combination with other operator to give the opposite result, also called a negative result.

Syntax: Select * from table name
where column_name not in
(values, values);

- Between (): Between operator selects values within a given range. The values can be numbers, text or data

syntax: Select * from table_name
where SQL.

between 45000 & 50000);

- Not Between :- The SQL 'Not Between' operator is used to getting the values as part of result set which is outside of the range specified by the 'Between' operator.

Syntax :- Select * from table_name
where column_name Not between
(value, value);

* Results- The SQL commands are executed successfully.

By :

Date

Practical No. 8.

* Aim :- To study and implemented order by group by and having SQL clause.

* Theory :-

① Order by clause

The order by clause is used in SQL queries to set the data returned by a query in ascending and descending order based on one or more columns.

- Syntax :- Select column1, column2.... from table_name order by column1 (ASC/DESC), column2 (ASC/DESC], ...;

- Example :- Select name, salary from employees order by salary DESC;

② Group by clause.

The group by clause is used to group rows that have the same values into summary rows of used with aggregate functions.

count(), Sum(), Avg(), etc.

Syntax:- Select column_name : aggregate function (column_name 2) from table_name
Group by column 1 ;

Example:- Select department, count (*)
from employees
Group by department ;

③ Having Clause :

The Having clause is used to filter records after the group by claud is applied . It works similarity to the where valuer but is specifically used with aggregate.

Syntax :- Selec- column 1 , aggregate_function. (column 2) from table_name
group by column 1
Homing aggregate_function (column 2) condition ;

Example :- Select department, count (*)
from employees
Group by department
Having count (*) > 10 ;

* Result :- Hence the Order by, Group by in SQL has been executed & implemented Successfully.

Date

Practical No. 9.

* Aim :- Write SQL queries to implement joins.

* Theory :-

- A join is a fundamental operation that combines rows from two or more tables based on a related column bet^ them. Joins are essential for retrieving compre-hensive data spans multiple tables, enabling more complex queries.

* Types of joins :-

① Inner Join

→ An inner join returns only the rows that rows matching values in both table being joins.

Use when you need to retrieve records that have corresponding entries in both tables.

Syntax :-

```
Select table1.column1, table2.column2,...
from table1
Inner Join table2
ON table1.common_column = table2.
                          common_column;
```

② Left Join (Left outer join)

Date

↔ A left join returns all records from left table & the matched records from the right table of there is no match, the result is Null on the right side.

Syntax:- Select table.column, table2.colmn2.
from table1
left Join table2
ON table1.common_column = table2.

③ Right Join (Right outer Join)

- A right join returns all records from the right table & the matched records from the left table of there is no match, the result is NULL on left side.

Use when you want all records from the right table regardless of whether there is a matching record in the left table.

Syntax:-

Select table.column, table2.column2,..
from table1
Right Join table2
ON tables.common_column = table2.

④ Full Join (Full outer join)

A full join returns all records when there is a match in either the left or right table of there is no match, the result is

NULL for the side that lacks a matching record, like when you need to retrieve all records from both tables, matching where possible & filling with NULL where there is no match.

Syntax:- SELECT table1.column1, table2.column2
FROM table1,
full outer JOIN table2.
ON table1.common_column = table2.

⑤ Cross Join.

A Cross Join return the product of the two tables, meaning it combines every row of the first table with every row of second table.

Syntax:- SELECT table1.column1, table2.column2
from table1.
CROSS JOIN table1;

⑥ Natural Join

A automatically join tables based on all columns with same name & types in tables.

Syntax:- Select * from table1
Natural join table2.

* Result:- Hence SQL queries executed successfully.

Date

Practical No. 10.

* Aim :- Write SQL queries using different operators.

* Theory :

1. Arithmetic Operators :

Arithmetic operators perform mathematical operations on numeric data. Common arithmetic operators are :

- '+' (Addition)
- '-' (Substraction)
- '*' (Multiplication)
- '/' (Division)
- '%' (Modulus)

2. Comparison operators :

Comparison operators compare two values. Common comparison operators are :

- "=" (equal)
- '!=' or '<>' (Not equal)
- '>' (Greater than)
- '<' (less than
- '>=' (greater than or equal to)
- '<=' (less Than or equal To)

3. Bitwise operators :

Bitwise operators perform bit manipulation between two integer values common bitwise operators are :

- '&' (Bitwise AND)
- '|' (Bitwise OR)
- '^' (Bitwise XOR)
- '~' (Bitwise NOT)
- '<<' (Bitwise Left shift)
- '>>' (Bitwise Right Shift)

4. Logical Operators :

Logical operators in sql are used to combine multiple condition multiple in a 'where' clause to filter the results of a query.

1) AND : Return 'TRUE' if all the conditions seperated by 'AND' are 'TRUE'.

2) OR : Returns 'TRUE' if any of the conditions seperated by 'OR' is 'TRUE'.

3) NOT : Reverse the result of any condition ; return 'TRUE' if the condition is 'FALSE'.

4) BETW

4) BETWEEN : The 'BETWEEN' operator is used to select values within a range.

5) IN : The 'IN' operator is used to specify multiple possible values for a column.

6) LIKE : The 'LIKE' operators is used in a 'WHERE' clause to search for a specified pattern in a column.

7) IS NULL : The 'Is NULL' operator is used to filter records with NULL values in a column.

8) EXITS : The 'EXITS' operators is used to negate a condition:

* Result: The SQL queries to implement operators are executed successfully.