LABORATORY 1 (THEORY)

DDL ⇒ Data Manipulation Language

↳ It is a set of instructions to create, modify and remove DB objects such as tables, indexes & users.

eg:- DROP, CREATE, ALTER.

CREATE :→ It is used to create table in a DataBase (DB).

Syntax: create table Table_Name (
"column 1" "data type"
"column 2" " data type"
"column 3" " data type"
- - - - - - - -
"column N" "data type");

⊛NOTE:- We can create a copy of an existing table using the create table command. The new table gets the same column signature as the old table.

ALTER:- It is used to add, delete or modify columns in an existing table.

ADD → ALTER TABLE table_name
ADD column_name datatype;

DROP → ALTER TABLE table_name
DROP COLUMN column_name;

MODIFY → ALTER TABLE table_name
MODIFY column_name
datatype;

CONSTRAINTS:→ It is used to specify rules for the data in a table.

NOT NULL ⇒ Ensure that a column cannot have a NULL Values

UNIQUE ⇒ Ensure that all column's value are distinct / different.

PRIMARY KEY ⇒ A combination of NOT NULL & UNIQUE.
Each row is unique. || Only one in table || No duplicacy

FOREIGN KEY ⇒ Prevents action that would destroy links b/w tables.

↳ is a field, that refers to PRIMARY KEY in another table.

FK = Foreign key.

PK = Primary key.

A table with FK is ⇒ Child

A table with PK is ⇒ parent

CHECK ⇒ is a constraint which is used to limit the value range that can be placed in a ~~col~~ column.

{IF we apply on a column it will allow only certain values for this columns;}

INSERT;- It is used to insert new records in a table.

```
INSERT INTO table_name
(column1, column2, ------)
VALUES (values 1, values 2, ----);
```

```
INSERT INTO table_name
values (value 1, value 2, ---);
```

LABORATORY - 6)

SELECT:- is used to select data from DB

```
Select column1, column2,.....
FROM table_name;
```

SQL OPERATORS:

Arithmetic → Add (+), Subtract (-), Multiply (*), Divide (/), Modulo (%)

Comparison → Equal to (=), Greater than (>), Less than (<), Greater than or equal to (>=)
Less than or equal to (<=), NOT equal to (<>)

Logical → ALL → TRUE if all of the subquery values meet the condition

AND → TRUE if all the conditions seperated by AND is TRUE

ANY → TRUE if any of the subquery values meet the condition

BETWEEN → TRUE if the operand is within the range of comparisons.

EXISTS → TRUE if the subquery returns one or more records

IN → TRUE if the operand is equal to one of a list of expression

LIKE → TRUE if the operand matches a pattern

NOT → Displays a record if the conditions is NOT TRUE.

OR → TRUE if any of the conditions seperated by OR is TRUE

REFERENTIAL INTEGRITY :- (RI)

It refers to the accuracy and consistency of data within a relationship.

In Relationships, data is linked b/w two or more tables. This is achieved by having the foreign key reference a primary key value. Because of this we need to ensure that data on both sides of the relationship remain intact.

Laboratory (3)

UPDATE :- It is used to modify the existing records in a table.

Syx → UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;

DELETE :- Is used to delete the existing records in a table

Syx :- DELETE FROM table_name } Deleting a particular record
WHERE condition;

DELETE FROM table_name; } if there is no where clause then the DELETE command will delete the entire ~~ent~~ record into that table.

SELECT is a limited form of DML statement in that it can access only data in the DB. It cannot manipulate data in the DB, although it can operate on the accessed data before returning the results of the query.

As per SQL Standard select is an DML.

SELECT → DML {limited form} i.e, it is DQL

SELECT * → DML.

LABORATORY (4)

DUAL is a table that is automatically created by Oracle DB along with data dictionary.

It has one column, DUMMY defined to be VARCHAR2 (1) and contains one row with a value X.

Aggregate functions :- It is used to perform the calculations on multiple rows of a single column of a table.

eg:- COUNT(), SUM(), MIN()
AVG() , MAX(),

STRING FUNCTIONS :-

LOWER() → Converts a string to lowercase

UPPER() → Converts a string to uppercase

TRIM() → Removes leading and trailing spaces from a string.

LTRIM() → Removes leading spaces from a string.

RTRIM() → Removes trailing spaces from a string.

TIME FUNCTIONS :-

ADD_MONTHS → Add specific time interval to a date.

```
SELECT ADD_MONTHS ( date, no.of months) FROM table_name;
```

MONTHS_BETWEEN :- Difference b/w two dates.

```
SELECT MONTHS_BETWEEN (date, date2) FROM table_name;
```

TO_DATE :-

```
SELECT TO_DATE ( Date, Format) FROM Table.;
```

* TO_char :- Is used to type cast a numeric or date input to character type with format model.

```
TO_CHAR (date, 'format');
```

GROUP BY :- Statement groups rows that have the same values into summary rows.

SYNTAX

```
SELECT column_name
FROM table_name
WHERE condition
Group by column_name
Order by column_name;
```

(*) Group by uses the aggregate functions.

Format

YYYY → Full Yr

YEAR → Year spelled out

MM. → two digit month

MONTH → Full name of month

MON → 3 digit month

DD → 2 digit date

DY → 3 letter of day of week.

DAY → Full name of week

LABORATORY (5)

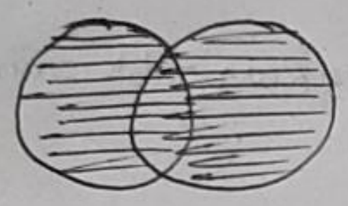
A join is mean for combining fields from the two tables or more by using values common to each.

Types:- Inner, Left, Right, Full, Natural, Cross

INNER  Returns all rows when there is at least one match in Both fields

LEFT  Return all rows from the left table, and the matched rows from the ~~right~~ left table

RIGHT  Returns all rows from the right table, and the matched rows from the right table

FULL  Return all rows when there is a match in ONE of the table.

RIGHT = RIGHT OUTER

LEFT = LEFT OUTER

JOIN = INNER JOIN

FULL = FULL OUTER.

Natural join:- It is a join operator that creates an implicit join clause for you based on the common column in the two tables being joined.

A Natural join can be Inner, Left, Right join.

The default is Inner Join.

→ Natural Join is Better than Inner Join

NATURAL LEFT JOIN → All Left.
NATURAL RIGHT JOIN → All Right
NATURAL FULL JOIN → Full

} Natural Join doesn't have any WHERE clause.

CROSS JOIN :- It returns the cartesian product of the sets of record from the two or more joined tables.

SELF JOIN :- SQL self join is used to join a table to itself as if the table were two tables.

```
SELECT a.column_name, b.column_name,.....
FROM table a, table b.
WHERE a.common_field = b.common_field;
```

CONDITIONAL JOIN :- A join that is not a natural join should have an explicit condition for joining is called conditional join.

INNER → Returns the combined tuple b/w 2 or more tables.

OUTER → Returns the combined tuple from a specified table even if join condition fail.

Alias → It is used to give a table, column of a table, a temporary name.

COLUMN

```
SELECT column_name
AS alias_name
FROM table_name;
```

TABLE

```
SELECT column_name
FROM table_name
AS alias_name;
```

LABORATORY (6):

(U) UNION :- Combines distinct results of two or more SELECT statements.

(U) UNION ALL :- Combines all results of two or more SELECT statement including duplicates.

(∩) Intersect :- Returns only the common records obtained from two or more SELECT statement.

(—) Minus :- Return only those records which are exclusive to the first table.

→ Select * from One UNION Select * from Second;

→ Select * from One UNION ALL Select * from Two;

→ Select * from One INTERSECT Select * from Two;

→ Select * from One MINUS Select * from Two;

LABORATORY (7)

A subquery is a query within another SQL query and embedded with in the WHERE clause.

SubQuery can be used with the SELECT, INSERT, UPDATE and DELETE statements along with the operators like =, <, >, <=, >=, IN, BETWEEN etc.

Types: →

Single: → Sub Query which returns single row output.

Multiple: → multiple row output

Correlated → Correlated subqueries depend on data provided by the outer query.

→ **Operators used:** IN, ANY, ALL → **Operator used** → **Exists**.

Simple Subquery: the sub query is first processed and then the main query is processed

Correlated SubQuery: → A row from the main outer table, is selected and is used for in the **subquery**.

↓

Outer Table

LABORATORY (8)

VIEWS:- It is a virtual table, in the Query / Temporary table

```
CREATE VIEW view_name
AS SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

```
DROP VIEW views_
        name;
```