Task-2.1 4/8/25

STUDY - OF SQL COMMANDS

* Data definition language. (DDL).

1) Create:-

Command name: Create

Description: The Create commands it utilized to create database, tables, triggers and other database objects.

Syntax: Database_Name;

Example: A data base is collection of data stored in it and creating a database is the first step when designing a relational data system.

2) Alter:-

Command Name: Alter

Description: The Alter command is used for the modification of the structure of the database in the existing table without delting or recreating the table.

Syntax: ALTER TABLE table_name
ADD column_name_datatype.

Example:- To add a New column in the table.

3) Drop:-

Command Name: Drop

Description:- DROP is a DDL command utilized to delete/remove database objects from the database

Syntax:- DROP DATABASE Database_Name;

Example: Removing a database from the database

4) Truncate:-

Command Name:- Truncate

Description: The Truncate command is used when we want to delete all records from a table and the structure of table is not deleted. It is faster than deleting because it removes all rows with a single click.

Syntax:- TRUNCATE TABLE table_name;

Example: Suppose you want to remove the record of the movies table.

5) Rename

Command Name: Rename

Description: The RENAME is a DDL command that is utilized to change the names of database objects such as column names and tables names.

Syntax: RENAME TABLE oldtable_name / new_table_name.

Example: Changing the old table name to the new table name.

6) Comment

Command Name: Comment

Description: Add comments to the data dictionary.

Syntax:- COMMENT 'comment_text' ON TABLE table_name;

Example: SELECT * FROM CUSTOMERS;

* Data Manipulation Language (DML)

1) Insert:-

Command name: Insert

Description: Insert data into a table.

Syntax: INSERT INTO table_name [column 1, column2, ...) VALUES (value 1, value 2, ...);

2) Update:-

Command name: Update

Description: Update existing data within a table.

Syntax: UPDATE table_name SET column 1=value1, column 2= value 2 where conditions.

3) Delete:-

Command name: delete

Description: Table control concurrency.

Syntax: DELETE FROM table_name WHERE conditions;

4) Lock:

Command name: Lock

Description: Table control concurrency.

Syntax: Lock TABLE table_name IN Lock_mode;

5) Call:-

Command name: Call

Description: Call a PL/SQL or JAVA subprogram.

Syntax: CALL produce_name (arguments);

6) Explain PLAN:-

Command name :- Explain name.

Description: Desribe the access path to data

Syntax: Explain Plan for select

* FROM table_name;

Example:-

INSERT INTO employees (first_name, last_name
Department)

VALUES ('Jane', 'smith', 'HR');

* Data Query language (DQL)

1) Select

Command Name: Select

Description: It is used to retrive data from the database

Syntax: SELECT column 1, column 2, ... from table_name WHERE condition;

2) From

Command Name: FROM

Dricription: Filters row before any grouping or aggregation.

Syntax: SELECT column1,
FROM table_name
WHERE condition;

3) Where

Command name: Where

Description: Filters rows before any grouping or aggregation.

Syntax:-
Syntax: SELECT column 1
FROM table_name;
WHERE condition;

4) Group by:-

Command name: Group by.

Description: Group rows that have the same values in specified column.

Syntax:- SELECT column 1, AGG-FUNCTION (column 2)
FROM table_name
GROUP BY column 1;

5) Having:-

Command name: Having.

Description: Filters the results of GROUP BY

Syntax: Select column 1, AGG-FUNCTION (column 1)
FROM table_name.
GROUP BY column 1
Having condition;

6) Distinct:-

Command name: Distinct.

Description: Remove duplicate rows from the result set.

Syntax: SELECT DISTINCT column 1, column_name
FROM table_name;

7) Order by:-

Command name: Order by

Description: Sorts the result set by one or more columns.

Syntax: SELECT column 1
FROM table-name.
ORDERED BY column 1

8) Limit:-

Command name:- Limit

Description: By default, it short in decending order unless specified as DESC

Syntax: SELECT * From table_name LIMIT number;

Example:-

```
SELECT first_name, last_name, hire_data
FROM employee
WHERE department = 'sales'
ORDER BY hire_data DESC;
```

* Data Control language

1) Grant:-

Command name: Grant

Description: Assigns new privileges to a user account, allowing access to specific database objects, actions or function.

Syntax: Grant privilege_type [(column_list)]

2) Revoke:-

Command name: Revoke

Description: Revoke previously granted privileges from a user account, taking away this access to certain database objects or actions.

Syntax:- REVOKE [Grant option For]
Privilge_type [(column_list)] ON
[object_type] object_name FROM user
[CASCADE]

Example:

GRANT SELECT, UPDATE ON EMPLOYEE TO (USER_NAME)

* Transaction Control Language: (TCL)

1) Commit:-

Command name:- Connit.

Description: Save all changes movie during the transaction.

Syntax: Connet;

2) Rollback:-

Command name: Rollback

Description:- Undoes all changes mode during the transaction.

Syntax:- ROLLBACK.

3) SAVEPoint:-

Command name:- savepoint.

Description: Creates a savepoint within the current transaction.

Syntax:- SAVEPOINT savepoint_name;

4) Begin Transaction:-

Command name: Begin transaction.

Description:- Start a new transaction.

syntax:- BEGIN Transaction.
[transaction_name];

Example:

BEGIN TRANSACTION;
UPDATE employee SET, department= 'Marketing' WHERE department = 'sale';
SAVEPOINT before_update;
UPPATE employee SET department = 'IT' WHERE department
ROLLBACK TO SAVEPOINT before_update;
CONNET;

Task-2.2 Design and Implementation Hiearchical / Network Model

1) Identify the specificity of each relationship in university mangement system.

Answer

① Student - Enroll - courses — Many to Many
② Department - offers - courses — one to many
③ Professor - Teaches - courses — one to many
④ courses - has - schedule — one to one
⑤ student - Makes - payment — one to many
⑥ Student - Allocated - Hostel room — one to one
⑦ student - Has - Attendence — one to many

2) Find the domain of attributes and apply suitable constraints

Ans

* Student Entity

Attribute	Domain	Constraints
1) Student_id	VARCHAR(10)	Primary key, NOT NULL, Unique
2) first_name	VARCHAR (50)	NOT NULL
3) last_name	VARCHAR (50)	NOT NULL
4) email	VARCHAR (100)	Unique, valid, NOT NULL
5) dob	DATE	NOT NULL MUST BE > age 18

6) dept_id VARCHAR(10) Foreign key to [Department (depa_id)]

* Professor Entity.

Attribute	Domain	Constraints.
Prof_id	VARCHAR(10)	Primary key, NOT NULL, Unique
first_name	VARCHAR(50)	NOTNULL
Last_name	VARCHAR(50)	NOT NULL
email	VARCHAR(100)	Unique, NOT NULL
depa_id	VARCHAR(10)	Foreign key to [Department dept_id]

* Enrollment :-

Attribute	Domain	Constraints.
enroll_id	INT	Primary key.
student_id	VARCHAR(10)	Foreign key to student (Student_id)
course_id	VARCHAR(10)	Foreign key to course (Course_id)
enroll_date	DATE	NOT NULL

* Payment Entity.

Attribute	Domain	constraints.
Payment_id	INT	Primary key.
student_id	VARCHAR(10)	Foreign key to student (student_id)
amount	DECIMAL(10,2)	NOT NULL, value > 0.
Payment_date	DATE	valid, NOT NULL

* Department Entity:-

Attribute	Domain	constraint
Department_id	INT	Primary key
Dept_name	VARCHAR(100)	Unique, NOT NULL
HOD	INT	Forigen key, -> (Faculty (faculty_id))

* Schedule Entity:-

Attribute	Domain	Constraints
Schedule_Id	INT	Primary key.
course_id	INT	Forigen key -> course (course_id)
faculty_id	INT	forigen key -> Faculty (faculty_id)

* Attendance Entity :-

Attributes	Domain	Constraints.
Attendence_id	INT	Primary key.
Student_id	INT	Foriegn key → (student (student_id))
Course_id	INT	Foriegn key → (Course (Course_id))
Status	VARCHAR(10)	Check (status IN ('Present', Absent'))

* Hostel Entity

Attributes	Domain	Constraints.
Hostel_name	VARCHAR(100)	Unique, NOT NULL
Hostel_id	INT	Unique, NOT NULL
Warden_Name	VARCHAR(100)	NOT NULL
Location	VARCHAR(200)	NOT NULL.

Perform SQL operation using DDL and DML commands:-

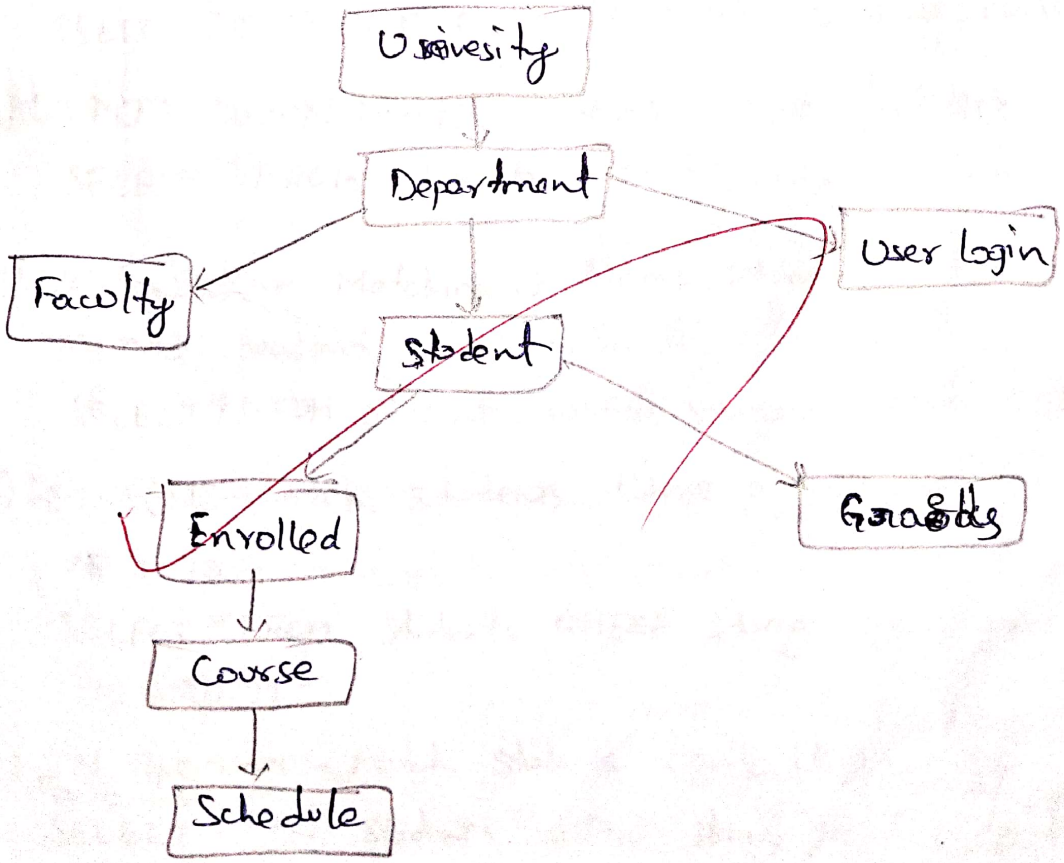
DDL Commands:-

CREATE Student (student_ID INT Primary key, First_name char(100) NOT NULL, Last_name char(100) NOT NULL, email varchar(100) NOT NULL valid);

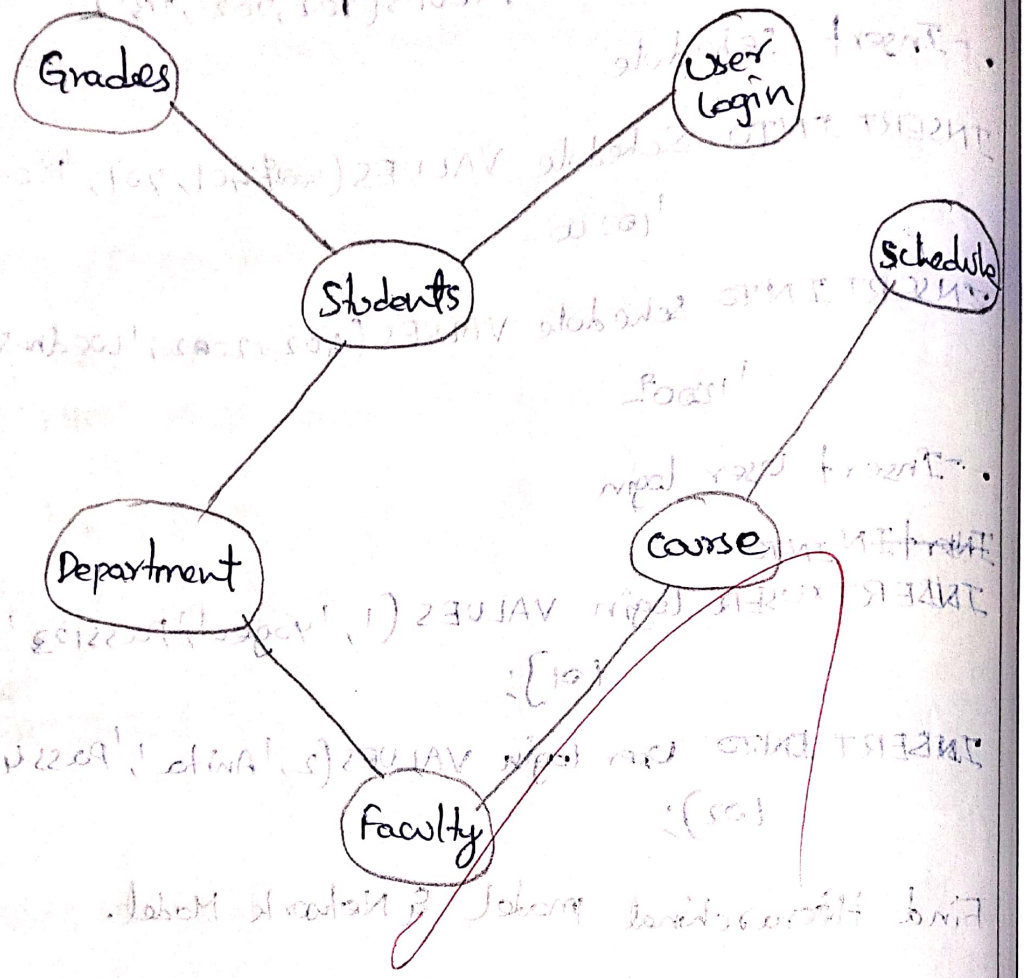
```
--Insert Enrollments
INSERT INTO Enrollment VALUES (101, 201, 'A');
INSERT INTO Enrollment VALUES (102, 202, 'B');
--Insert Schedule
INSERT INTO Schedule VALUES (401, 201, 'Monday'
                             '10:00-
INSERT INTO Schedule VALUES (402, 2002, 'Wednesday',
                             '1200-
--Insert User login
INSERT INTO User login VALUES (1, 'yogeo', 'Pass123',
                               101);
INSERT INTO User login VALUES (2, 'Anita', 'Pass456',
                               102);
```

Find Hierarchical model & Network Model.

Hierarchical Model:-



Network Model



Task: 2:3 :- Different types of select queries.

1) Basic select :- Fetches all columns & rows from the student table

SELECT * FROM students.

2) Select specific columns :- Fetchs only selected columns

SELECT * FROM students WHERE stage>20;

3) WHERE clause (Filtering data) :- Fetchs records of student solder than 20.

SELECT DISTINCT Corse_id FROM student_corse.

4) ORDER BY (sorting) :-

4) ~~DISTINCT~~ :- Sorts students by age in descending

SELECT stname, stagge FROM students, ORDER BY stage DESC;

5) DISTINCT :- Fetch only unique corse IDs (remove duplicate).

SELECT DISTINCT corse_id FROM student_corse;

6) LIMIT (Top N Results) :- Result only 5 Rows.

SELECT * FROM students LIMITS 5;

7) LIKE (Pattern Matching) :- Finds student whose names ~~student~~ start with "A".

SELECT * FROM Students WHERE Stname LIKE 'A%';

8) BETWEEN :- Fetch students whose age is between 18 & 22.

SELECT * FROM Students WHERE Stage BETWEEN 18 AND 22;

9) IN Operator :- Fetch students aged 18, 20 or 22.

SELECT * FROM Students WHERE Stage IN (18, 20, 22);

10) Aggregate Functions:- Returns total number, average age, maximum & minimum.

```
SELECT COUNT(*) AS total_student FROM students;
SELECT AVG(stage) As average_age FROM students;
SELECT MAX(stage), MIN(stage) FROM students;
```

11) GROUP BY:- Groups students by course & counts them

```
SELECT course_id, count(stid) As total_student
FROM student_course
GROUP BY course_id;
```

12) HAVING (with Groups):- Fetches only course where more than 2 students enrolled

```
SELECT course_id, count COUNT(stid) AS total_
students
FROM student_course
GROUP BY course_id
HAVING COUNT(stid) > 2;
```

13) JOIN queries:-

→ INNER JOIN:- Shows students names with their course names

```
SELECT S.studentname, c.name.
FROM students S
INNER JOIN student_course SC ON S.sto_id =
    SC.stoid.
INNER JOIN course CON Sc.Course_id = C.course_id;
```

→ LEFT JOIN:- Shows all students (even if they didn't take a course)

```
SELECT S.Stoname, C name
FROM Students S
LEFT JOIN student_course SC ON S.student_id =
    SC.stoid
```

LEFT JOIN course C ON SC.course_id = C.course_id;

14) Subquery (Nested SELECT):- Finds students older than the average age.

```
SELECT Stuname, Stoage.
FROM Students
WHERE Stoage > (SELECT AVG(Stoage) FROM
Students);
```

15) UNION (Combine Results):- Combine names of students & course names (unique values only).

```
SELECT Stoname FROM students
UNION
SELECT name FROM course;
```

VEL TECH	
EX No.	2
PERFORMANCE (5)	5
RESULT AND ANALYSIS (2)	3
VIVA VOCE (3)	3
RECORD (4)	4
TOTAL (15)	15
SIGN WITH DATE	13/10/25

Result:- Different types of select queries are proved.

13/10/25