Date: 29-08-25

Task 1:

~~Title~~ Aim: Conceptual design using ER model - college slot booking and management system

Tools required: https://drawio

step involved in creating ER Diagram

step 1: problem Understanding and requirement analysis

* Analyse real world application: college slot booking and magement system.
* understanding domain: student, Dept course, slot.

Step 2: Identify major entities

* STUDENT
* DEPARTMENT
* COURSE
* SLOT

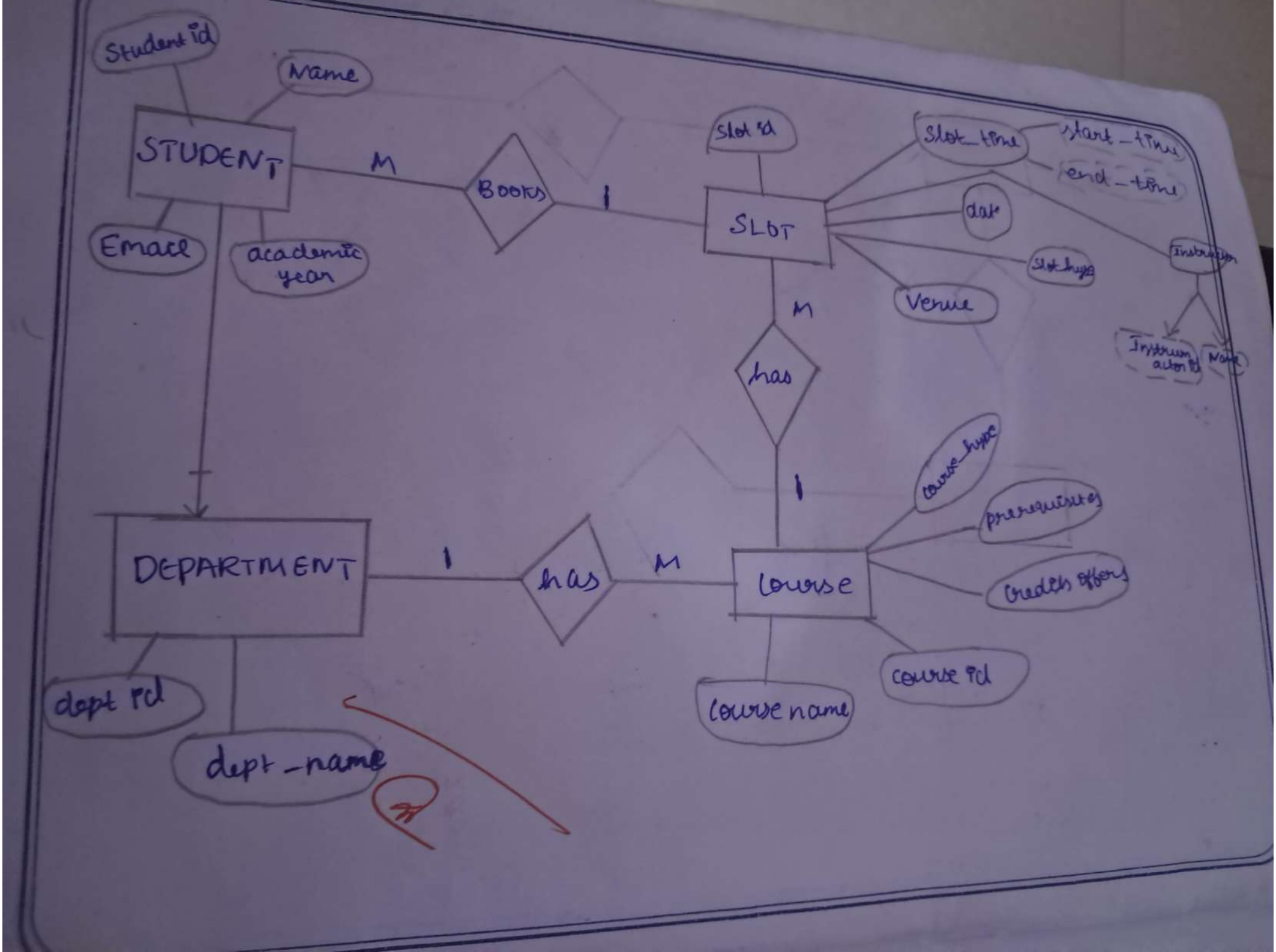
Step 3: Entity Attributes.

→ STUDENT: Student_id (PK), name, email, academic year.

→ Department: dept_id (PK), deppt-name

→ course: course_id (PK), course_name, credits-offered prerequisites, course-hyp

→ SLOT: slot-id (PK), slot-time, Instructor, date, venue.


Student id
Name
STUDENT
M
BOOKS
1
Slot id
Slot_time
start_time
end_time
SLOT
date
Slot type
Venue
Instructor
Instructor id
Name
Email
academic year
M
has
1
course_type
prerequisites
credits offers
DEPARTMENT
1
has
M
Course
course id
course name
dept id
dept_name

Step 4:
→ A student has one dept
→ One department has many courses
→ A course has many slots
→ One or more student chooses one slot

Step 5: Draw ER Diagram using draw.io
* open https://draw.io
* choose Blank diagram → click create
* From left panel, drag the following:
→ use rectangle for entities (Student, Dept)
→ use ellipses for Attributes (student_id, dept_id)
→ use diamonds for relationships (has, books)
→ connect using lines.
→ use PK or underline to denote primary key
→ use labels such as (1:N), (M:N)

Step 6: Relationships :-
→ Student (1) → (1) department
→ Department (1) → has → (M) courses
→ Course (1) – has → (M) slots
→ Student (M) – Books → (1) slot

VEL TECH	
EX NO.	1
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	—
TOTAL (20)	15
SIGN WITH DATE	

Input: College slot Management System
scenario user requirements student slot booking, faculty availability, Room scheduling, Time table management)
Database design rules (Entity – Attribute relationship, Identification, normalisation considerations

Result: This task helped us to understand the importance of conceptual design in database system using draw.io. We able to book real time slot into an ER diagram

Output Theory [illegible]

Clearly [illegible]

- All [illegible]
- All [illegible] conditions
- Concept [illegible] Analysis [illegible]

Task 1.2

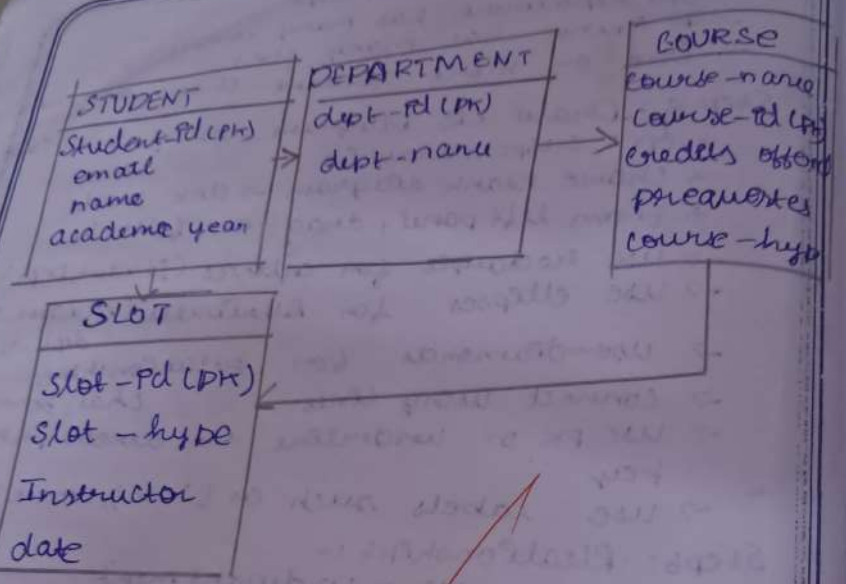
Aim: Steps for converting the ER diagrams for table

* Entity type b
* All single-valued attribute becomes column for table
* A key attribute of entity type represented by prim ary key
* The multivalued attribute is represented by separate value.
* Derived attribute represented by separate value.
* Composite attribute represented by separate
* Derived attributed are not considered in table

using the values you convert ER diagram and assign mapping between tables.

VEL TECH	
EX NO.	1
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	—
TOTAL (20)	15
SIGN WITH DATE	23/7/25

Result: Hence, the relationship model of College slot booking and management system using ER model was completed


STUDENT
Student-Id (PK)
email
name
academic year
DEPARTMENT
dept-Id (PK)
dept-name
COURSE
course-name
course-Id (PK)
credits offered
prerequisites
course-hyde
SLOT
Slot-Id (PK)
Slot-hyde
Instructor
date

Output :
Table employee successfully
Create

output

EMPLOYEE_ID	Int
EMPLOYEE_NAME	VARCHAR(50)
ADDRESS_Varchar	VARCHAR(100)
AGE NUMBER	INT

~~12/8/25~~ 5/9/25

Task 2.8

Aim : To Generating design of other traditional database Model. ~~to~~ implement DDL command and DML command of SQL and database system

Implementation of DDL commands

of SQL with example

* Create
* Alter
* Drop

DDL commands: It is used to create new table

1. Create table

Query :

Sol :

```
CREATE TABLE EMPLOYEE (EMPLOYEE_ID
                       NUMBER(5)
                       PRIMARY KEY
EMPLOYEE_NAME VARCHAR(50),
ADDRESS VARCHAR2(100),
AGE NUMBER(3) );
```

2. DESCRIBE TABLE : Display the struct of table.

Query :

Sol

```
DESC Employee.
```

Output
Table EMPLOYEE dropped.

Output
Table altered

3. DROP TABLE : Delete the entire table structure.

Query:

DROP TABLE EMPLOYEE(

4. ALTER TABLE : used to add or modify column in an existing table.

Query:

Sol

ALTER TABLE EMPLOYEE ADD SALARY NUMBER (8,2).

EMPLOYEE_ID	EMPLOYE_NAME	SALARY
101	Ravi	12000

DML commands

1. Insert INTO: Adding a new table

QUERY:

INSERT INTO EMPLOYEE (EMPLOYEE_ID, EMPLOYEE_NAME, ADDRESS, AGE).

VALUES (101, 'Ravi', 'Chennai', 28);

Output

EMP ID	EMPLOYEE_NAME	ADDRESS	AGE
101	RAVI	Chennai	28

Output

1 row updated

Output

1 row deleted.

	20
	23

VIVA TASK	
	2
	3
	4
	3
	3
	15
	20

5/8/25

DML commands are executed

DML

create, delete and

are executed

SELECT with WHERE clause

SELECT * FROM EMPLOYEE Where

Age = 28

output:

EMPLOYEE_ID	EMPLOYEE_NAME	Age
101	Ram	28

VEL TECH	
EX NO.	2
PERFORMANCE (5)	3
RESULT AND ANALYSIS (5)	3
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	18
SIGN WITH DATE	5/8/25

Thus the conceptual design of task DDL and DML commands are executed sucessfully.

Result

DDL & DML

~~The task to create, delete and alter the table are executed successfully~~

19/08/25

Task 2.1 DDL and DML commands with constraints

Aim: To Implement DDL and DML commands with constraints.

DDL - (Data definition language) -> Create, Alter, Drop, Truncate, Rename

DML - (Data manipulation language) -> Insert, update, Delite select

Constraints -> Primary Key, Foreign Key, Not Null, Unique, Check, Default

1. DDL commands for employee management system

1.1 Create table

```
Create Table Employee (
    Employee ID INT primary key,
    Employee Name VARCHAR (50) NOT NULL,
Gender char(1) Check (Gender In ('M','F'))
Age INT CHECK (Age >0 ),
Contact Number VARCHAR 2(20) UNIQUE,
Address VARCHAR2(100)
);
```

```
CREATE TABLE DEPARTMENT (
    Department ID Int primary key,
    Department VARCHAR 2 (50) NOT NULL.
    Manager ID INT
);
```

```
CREATE TABLE ASSIGNMENT(
  Assignment ID INT PRIMARY KEY,
  EMPLoyee INT NOT NULL,
  Department INT NOT NULL,
  Assigned DATE DEFault SysDATE,
  Role VARCHAR2(50),
  FOREIGN KEY (EMPloyee ID) Employee
                                (Employee ID)
  FOREIGN key (Department ID) REFERENCES
                  Department (Department
                                ID)
  );
```

output:
Table Employee created.

1.2 ALTER TABLE:

```
ALTER TABLE EMPLOYEE ADD
                Email VARCHAR2(50)

ALTER TABLE Employee MODIFY
                Contact number
                VARCHAR(20
```

Output

Table Employee altered.

Emp_ID	Emp_NAME	E-mail	Emp_contact number
101	Rani	Rani@gmail.com	7012571234

1.3 TRUNCATE Table

TRUNCATE TABLE Appointment

1.4 RENAME TABLE

ALTER TABLE EMPLOYEE RENAME TO Employees;

Table Employees created successfully

2. DML commands for employee management system

2.1 INSERT DATA

INSERT INTO EMPLOYEES (EmployeeID, Employee ID, Employee name, Age,

Ram, M, 35, 101, chennai, Ram@gmail.com.

Insert INTO Department (Department ID, Department name) manager ID.

Employee ID	Employee name	Gender	Age	Email
103	Rafesh	M	37	Rafesh@mail

2.2 UPDATE data

update Employees
SET Age = 34, Address = mumbai
where Employee ID = 1;

2.3 DELETE Data

DELETE FROM Assignment
Where Assignment ID = 109
Result: No rows returned empty table

2.4 SELECT Data

SELECT e. Employee name, Department name, a. Assignment table
FROM Assignment a
Join Employees e on a. Employee ID = e. Employee ID
Join Department d ON e. Department = d. Department

Employee Name	Department name	Assigned date	Role
Rane	CSE	2025	Developer

Output:

Employee name	ID	Age	place	gender	Role
Rajesh	102	35	chennai	M	Developer
Rane	103	30	Trichy	M	Developer.

VEL TECH	
EX NO.	2.1
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	
TOTAL (20)	15
SIGN WITH DATE	13/8/25

Result: The task to Implement DDL and dml commands for our task-1 Entity in RDBMS has been completed successfully

19/8/25

Task 3.1 Using clauses, operators and functions in Queries

Title: Implementation of DML commands using clauses, operators and functions in queries.

* Insert table
* Select table
* Update table
* Delete table

Objective:

* To understand the different issues involved in the design and implementation of a database system.

DML:
1. Insert 2. delete 3. Update 4. delete.

Insert Into: This is used to add records

INSERT INTO < relation | table name > (field-1, field-2, field-n) values (data-1, data-2, ... data-n);

Example: SQL > Insert Into member value (104, shan, HR, male);

EMP

Employee name	ID	Age	place	gender	Role.
Rajesh	102	35	chennai	M	Developer
Ravi	103	38	Trichy	M	Developer
Shaan	104	30	Karur	M	HR

update - SET - WHERE : Used to update.

Syntax : SQL > update relation name SET field name = data, field-name 2 = data, where field - name = data; WHERE field-name = data;

Example: UPDATE Employee SET Employee name = 'valan' WHERE EMP ID = 104;

Employee name	Emp-ID	Age	place	gender	Role
Rajesh	102	35	Chennai	M	Developer
Ravi	103	38	Trichy	M	Developer
valan	104	30	Karur	M	HR

DELETE FROM : Used to delete all the records

~~DELE~~ Syntax: DELETE FROM relation - name WHERE condition.

~~Output~~

~~Example~~ : DELETE FROM _ Where : Used to delete a selected record from a relation.

DELETE FROM Employee where age = 35

output

Employee name	Emp-ID	Age	place	gender	Role
Ravi	103	38	Trichy	M	Developer
valan	104	30	Karur	M	HR

Truncate : Delete all values and structure remains same.

Example : TRUNCATE Employee

Queries

1. Retrieve member name and with letter 'a' and member no between 101 and 103

Query :

```
SELECT first name.
last_name , salary
FROM employees
  where first_name
LIKE '%.m';
```

Output

Employee name	ID	Age	Place	gender	Role
Rani	103	38	Trichy	M	Developer

2. List the salary where pagecount between clause and operator.

Query :

```
select * from book where
salary
pagecount between 19000 and 22000;
```

Output

id	Employee name	Salary
103	Ramp	20,000

5. Find the records who has minimum number of ~~[illegible]~~ salary.

Output

Id	Employee name	Salary
104	Rafesh	15,000

4. Find the records whose ~~[illegible]~~ do date of joining is greater than 2022-02-07

ID	Employee name	doj
103	Ramp	2022-03-07

5. Combine the records of member id and employee relation union

Query.

SELECT membno FROM members
union select membno FROM borrowed;

Output

employee id
101
102
103
104

6. Find the employee and their place details using group by and oderby clauses

Query.

SELECT Employee, places, Count (*) no FROM book places Employee, publisher Group By ~~authors~~ Order by ~~author~~ Employee.

Output

Employee name	places
Raul	chennai
Rajesh	chennai

VEL TECH	
EX NO.	3.2
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	20
SIGN WITH DATE	

Result: To implement the DML commands executed. successfully using clauses, operators and function in Queries completed successfully

25/8/25

Task 3.2 Aggregate Functions

Aim:
To study and implement aggregate functions (Count (), sum, AVG() MIN(), MAX ()) on a sample student database

procedure
1. Create a table named students
2. Insert sample records
3. Write queries using aggregate functions
4. Observe and record the output

Commands with explanation

1) Count total number of employees

SELECT COUNT (*) AS Total_students
FROM ~~Students~~ Employee ;

Explanation:
* Count (*) counts how many rows (Employee) are in the table.
* As Total_~~Students~~ Employee gives a user friendly column name.

2) Find the highest ~~marks~~ salary obtained by a student.

SELECT MAX(salary) AS Highest - salary
FROM EMPLOYEE

Output

Employee name	Id	Salary
Rane	103	40,000

3) Find the minimum salary among the employees

Employee name	ID	Salary
Rajesh	104	15,000

4) Find the highest salary among the employees

Employee name	ID	Salary
Rane	103	40,000

5) Find the average age of Employees

~~Employee name~~

SELECT AVG (Age) AS AVG - ~~Marks~~ Age
FROM employees

output

AVG - AGE
32

VEL TECH

EX No.	3.2
PERFORMANCE	5
RECORD	5
VIVA	5
RESULT	15
TOTAL	

26/8/24

Result : Thus ~~SQL~~ commands Aggregate functions Executed successfully based on Employee is created successfully.

2/9/25

Task 4: Independent and correlated Nested Queries

Aim: To implement and understand Nested Queries in SQL, including independent and correlated subqueries with practical examples in a university database scenario.

Procedure

1. Create required tables. (eg, Students, Departments, Courses Enrollments)
2. Insert sample data.
3. Write and execute independent nested queries.
4. Write and execute correlated nested queries.
5. Observe and analyse the differences in execution and results

Explanation for nested query

A nested query (subquery) is a query written inside another query. it is used when the output of one query is required for another query.

1. Independent (Simple) Queries.

* Inner query runs once independently of the outer query.
* The result of inner query runs once independently.

Syntax:

```
SELECT column_list
FROM table
Where column operator (SELECT column
                        FROM table
 where condition);
```

Example 1: Independent Nested Query

Query:

```
SELECT Employee Name
FROM Employee
Where EMP_ID = (101
    SELECT EMP_ID
    FROM Employee
    WHERE EMP_NAME = 'Rani');
```

Output
Rani

2. Correlated Nested Queries

The inner query is executed repeatedly, once for each row of the outer query.

* Inner query depends on a value from the outer query.

Syntax:

```
SELECT column_list
FROM table1 t1
WHERE EXISTS
    SELECT 1
    FROM table2 t2
    WHERE condition involving t1.column
                              = t2.column
```

Query

```
SELECT s.~~Student~~ Employee Name
FROM ~~Students~~ Employees emp
WHERE EXISTS(
  SELECT 1
  FROM Enrollment projects
  Where pa.EmployeeID= emp EmployeeID
```

Explanation:

* For each student in the outer query, the inner query checks if that student exists in Enrollments project.
* If yes → the student is returned.

Answer output for query

Rajesh
Rani
Valan
tamilarasan
Shaan

Defference between the query of independent correlated

Feature	Independent subquery	Correlated Subquery
Execution	Inner query runs once	Inner query runs for each other
dependency	Independent of outer query	Dependent on outer query values

performance	Faster	slower
Example usage in table	Finding dept of sec in employee	checking enrollment of Employee

Output

Rajesh
Ravi
tamilarasan
shaan
kaavanan
valan

VEL TECH	
EX NO.	4
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	1
TOTAL (20)	15
SIGN WITH DATE	26/3

Result :

To implement the independent and correlated Nested queries executed successfully

9/9/25

Task 5 - Writing Join Queries equivalent and for Recursive Queries

Aim: To implement of different types of queries using Employee scenario.

SQL Join clause

Step 1:

Syntax:

SELECT column 1, column 2, column 3..
FROM table-name1, table_name 2
Where table-name1. column name = table-name2. columnname;

Types of joins:

1. simple Join
2. Outer and self join

Simple Join:

Query

Select * from item, cust where emp_id = cust. id;

SELF Join

Query

Select * from emp x, emp y where x. salary >= (select avg (salary) from x. emp where x. deptno

Outer join:

Extends the result of simple join.

Types of join

Inner join: Return Records

Query

FROM Employee1

INNER join Employee2 ON

Employee1. column emp_id

Employee2. Dept_id;

LEFT outer join:

Query

SELECT Employee_name(s) FROM

Employee1. columnname.

Employee2. column_name.

Right outer Join:

Query

SELECT column_name FROM table1

Right join table2 ON

table1. column_name

table2. column_name;

Full outer Join

Syntax

```
SELECT column_name FROM table 1
FULL outer join table 2 ON
table 1. column_name
table 2. column_name;
```

SQL Join clause

```
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    EmployeeName VARCHAR(50),
    DeptID INT,
    ManagerID INT,
    Foreign key (DeptID)
CREATE TABLE projects (
    Project ID VARCHAR(10), PRIMARY KEY,
    Project Name VARCHAR(50),
    DeptID INT,
    FOREIGN KEY (Dept ;
CREATE TABLE Assignments
    AssignmentID INT PRIMARY KEY,
    Employee ID. INT,
    PROJECT ID VARCHAR(10);
CREATE Table Hierarchy (
    Employee INT,
), Manager ID INT
```

Insert into Departments
(201, 'Technology')
(202, 'Human Resources'),
(203, Marketing');
Insert into Employees
(1, 'Alice', 201, 3),
(2, 'Bob', 202, 4),
(3, 'Charlie' 201, NULL),
(4, Daned, 202, NULL);
Insert into projects values
('P1', Website Redesign', 201),
(P2, Hiring, 202),
(P3, API Development', 201)
('P4', Social Media campaign', 203);
Insert into Heirarchy values.
(1, 3),
(2, 4);

SELECT * FROM employeeheirarcy;

VEL TECH	
EX NO.	5
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	—
TOTAL (20)	15
SIGN WITH DATE	

Result: The implementation of SQL commands for Employee using Joins and recursive queries successfully

16/9/25

Exception,

Task 6 Triggers and view query for employee

Aim: To perform an action whenever seconds

View:

It is a virtual table based on the result-set of an sql statement.

Creating a view

SQL

```
CREATE VIEW employee_public_info AS
SELECT
  employee_id,
  first_name
  last_name,
  email
FROM
  employees

where
  is_active = 1
```

Query to view

```
SELECT first_name, email FROM employee
  public info where employee_id=104;
```

Output

employee_id	first_name	last_name	email	salary
101	Rajesh	Khanna	RK@gmail.com	30,000
102	Valan	uday	VV@gmail.com	45,000
103	Ramc	-	Ravi@gmail.com	15,000
104	Shaan	-	Shaan@gmail.com	40,000

Treggers :

Special type stored procedure that automatically runs occur on database

Query.

```
CREATE TRIGGER before_employee_salary_update
BEFORE UPDATE ON employees
For each row
Begin
  IF OLD.salary <> NEW.salary THEN
    INSERT INTO salary_audit (emp_id,
                              old salary, new_salary
    values (OLD.employee_id, OLD.salary,
                              NEW.salary);
  END IF;
END;
```

Query :

```
UPDATE employees
SET salary = 75000
where employee_id = 101;
```

Output

emp_id	old_salary	new-salary	Changed at
101	60000	75000	2025-09-15

VEL TECH	
EX NO.	6
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	-
TOTAL (20)	15
SIGN WITH DATE	16/9/25

Result : The Implementation of triggers and views on the database has been successfully completed and verified

Date: 23/9/25

Task No: 7 PL/SQL procedure, functions, loops

Aim:
To implement PL/SQL procedure, functions and loops on number theory and business scenarios

Procedure:
PL/SQL is combination of SQL along with the procedural features of programming languages. It was developed by Oracle corporation in the early 90's to one of three key programming languages.

Syntax: Declare
<declare session>
Begin
<exutable command>
Exception
<exception handling>
end;

Program:
Declare
message varchar(20); slot closed"

Begin
dbms_output_put_line (messages
End;
output - slot closed
Dynamic Input :- -set server output
on declare
x number (5);
y number (5)
z (number(9))
begin
x=10
y=12
z: x+y
dbms.output

multiplication of x an of x and y "(112)
end;

output multiplication of x and y 120

Declare
Mind number, 3) :100;
Begin: If (hrd =10) then
dbms_output.put_line
Else if (hrd=50) then ('value of hrd is 50);
dbms=output put line (*value of hrd is 50");

```
Else If (bed -110) then
    dbms_output.line ('False');

Else:
    dbms_output.line('NOW');

END IF;
```

output: None

Exact Value is 100

PL/SQL procedure successfully completed

Loop :-

```
Declare
    led number(1);
    oed number(1);
Begin <<outloop>>
    For led IN 1...2 loop
            <<Inner_loop>>
        For Pd IN 1....2 loop
            dbms_output line
wed '' and oed is ;||oed);
    End loop inner_loop;

  END loop outer_loop;
End;
```

hid is : 1 and oid is : 1
hid is : 1 and oid is : 2
hid is : 2 and oid is : 1
hid is : 2 and oid is : 2

PL/SQL procedure successfully completed.

Function :-

```
Create or replace function.

Begin
If id > 200 then
    Return ('No slot available');
Else
    Return ('slot open');
End If;
End;
```

```
SQL> Create or replace procedure
prin_odd nos If cursor num_cs is
    Select she_id from students;
v_id Number;
v_id odd Boolean
v_i Number;
```

```
Begin
    open num_cus;
    Loop
        Fetch num_cus into v_id;
            Exit.
    while v_count < n loop
            v_is_prime := TRUE

    for i in 2 .. Trunc(sqrt(v_num))
                                    Loop
        IF MOD (v_num, i) = 0 THEN
            v_is_prime := FALSE;
            EXIT;
        END IF;
    If v_is_prime Then
            DBMS_Output.PUT_LINE
            v_Count := v_Count+1;
        End IF;
    END;
BEGIN:
        print_n(4)
End;
```

Output :- 1

2

3

VEL TECH	
EX NO.	7
PERFORMANCE (5)	5
RESULT AND ANALYS'S (5)	5
VIVA VOCE (5)	3
RECORD (5)	5
TOTAL (20)	18
SIGN WITH DATE	

23/09/25

Result: Thus the Implementation of PL/SQL function and loops on database has been completed succesfully.