

Date: 29-09-25

Task 1:

<sup>Alm</sup>  
~~Task~~: Conceptual design using ER model -  
college slot booking and management  
system

Tools required: <https://draw.io>

Step involved in creating ER Diagram

Step 1: Problem Understanding and  
requirement analysis

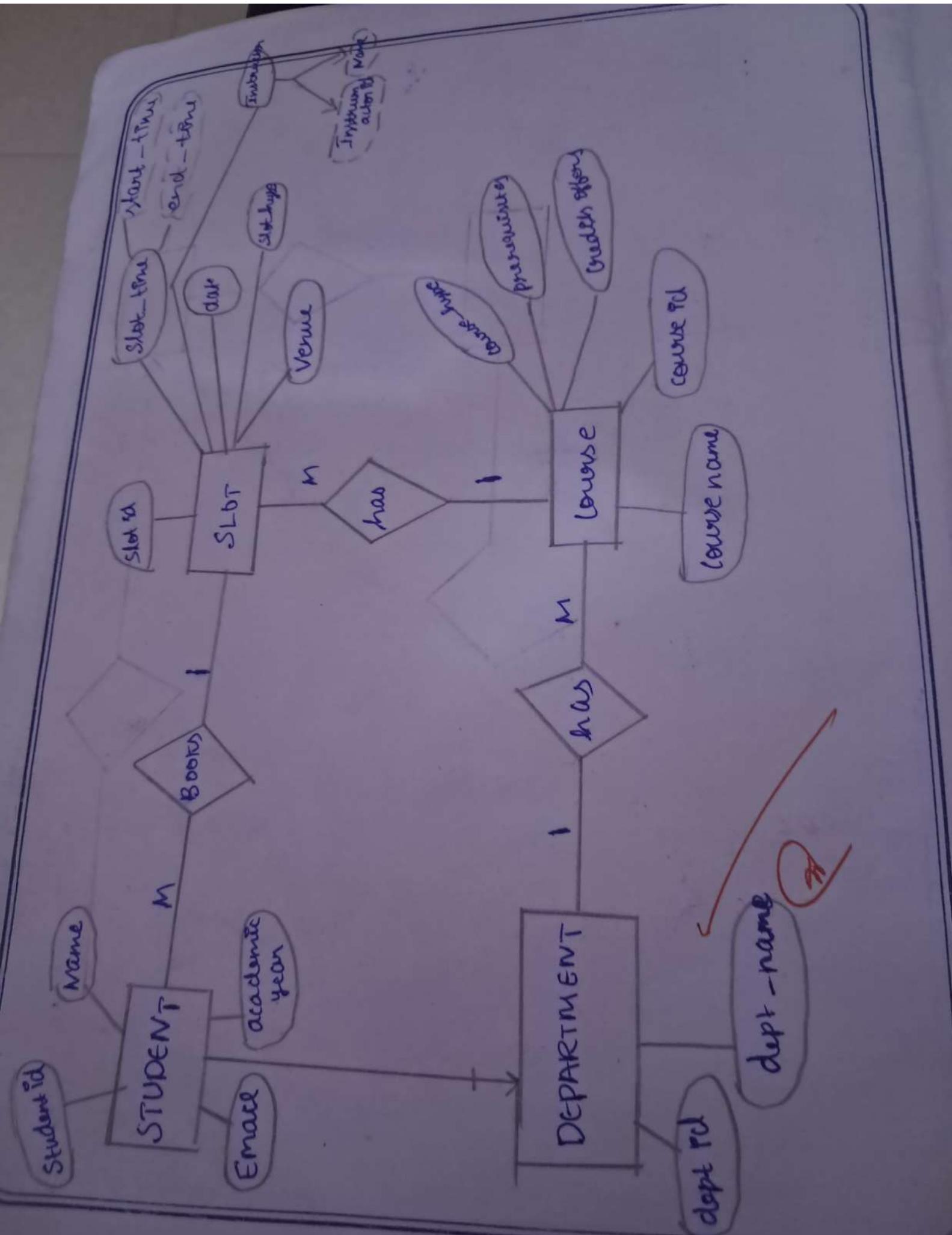
- \* Analyse real world application: college  
slot booking and management system.
- \* understanding domain: student, Dept  
course, slot.

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), dept-name
- course: course-id (PK), course-name,  
credits-offered, pre-requisites, course-type
- Slot: slot-id (PK), slot-time, Instructor,  
date, venue.



Step 4: A student has one dept

- One department has many students
- A course has many students
- One or more student courses one dept

Step 5: Draw ER Diagram using draw to

- \* Open steps: 11 draw to
- \* Create blank diagram → click create
- \* Change left panel, drag the following:
- \* From left panel, drag entities (Student, Dept)
- 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 on underline to denote primary key
- Use labels such as L1:N, L:M:N

Step 6: Relationships :-

- Student (1) → (1) department
- Department (1) → (M) courses
- Course (1) → (M) student
- Student (M) → (1) books

VET TECH	
X NO.	1
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	5

Input: College Student Management System

- Objectives: user requirements, Room
- Not working, faculty availability, Room
- Scheduling: Time table management

Database design rules (Entity - Attribute Relationships)

Result: This task helped us to understand the importance of conceptual design in database system using drawing of ER diagram, the above to book real time student info can ER diagram



By your obedient servant

*[Faint handwritten notes at the bottom of the page]*

Page 102

70 -

*[Faint handwritten notes at the bottom of the page]*

only. Modern  
to all countries  
reigning in a

1875

Output

1914-1915 10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-43-44-45-46-47-48-49-50-51-52-53-54-55-56-57-58-59-60-61-62-63-64-65-66-67-68-69-70-71-72-73-74-75-76-77-78-79-80-81-82-83-84-85-86-87-88-89-90-91-92-93-94-95-96-97-98-99-100-101-102-103-104-105-106-107-108-109-110-111-112-113-114-115-116-117-118-119-120-121-122-123-124-125-126-127-128-129-130-131-132-133-134-135-136-137-138-139-140-141-142-143-144-145-146-147-148-149-150-151-152-153-154-155-156-157-158-159-160-161-162-163-164-165-166-167-168-169-170-171-172-173-174-175-176-177-178-179-180-181-182-183-184-185-186-187-188-189-190-191-192-193-194-195-196-197-198-199-200-201-202-203-204-205-206-207-208-209-210-211-212-213-214-215-216-217-218-219-220-221-222-223-224-225-226-227-228-229-230-231-232-233-234-235-236-237-238-239-240-241-242-243-244-245-246-247-248-249-250-251-252-253-254-255-256-257-258-259-260-261-262-263-264-265-266-267-268-269-270-271-272-273-274-275-276-277-278-279-280-281-282-283-284-285-286-287-288-289-290-291-292-293-294-295-296-297-298-299-300-301-302-303-304-305-306-307-308-309-310-311-312-313-314-315-316-317-318-319-320-321-322-323-324-325-326-327-328-329-330-331-332-333-334-335-336-337-338-339-340-341-342-343-344-345-346-347-348-349-350-351-352-353-354-355-356-357-358-359-360-361-362-363-364-365-366-367-368-369-370-371-372-373-374-375-376-377-378-379-380-381-382-383-384-385-386-387-388-389-390-391-392-393-394-395-396-397-398-399-400-401-402-403-404-405-406-407-408-409-410-411-412-413-414-415-416-417-418-419-420-421-422-423-424-425-426-427-428-429-430-431-432-433-434-435-436-437-438-439-440-441-442-443-444-445-446-447-448-449-450-451-452-453-454-455-456-457-458-459-460-461-462-463-464-465-466-467-468-469-470-471-472-473-474-475-476-477-478-479-480-481-482-483-484-485-486-487-488-489-490-491-492-493-494-495-496-497-498-499-500-501-502-503-504-505-506-507-508-509-510-511-512-513-514-515-516-517-518-519-520-521-522-523-524-525-526-527-528-529-530-531-532-533-534-535-536-537-538-539-540-541-542-543-544-545-546-547-548-549-550-551-552-553-554-555-556-557-558-559-560-561-562-563-564-565-566-567-568-569-570-571-572-573-574-575-576-577-578-579-580-581-582-583-584-585-586-587-588-589-590-591-592-593-594-595-596-597-598-599-600-601-602-603-604-605-606-607-608-609-610-611-612-613-614-615-616-617-618-619-620-621-622-623-624-625-626-627-628-629-630-631-632-633-634-635-636-637-638-639-640-641-642-643-644-645-646-647-648-649-650-651-652-653-654-655-656-657-658-659-660-661-662-663-664-665-666-667-668-669-670-671-672-673-674-675-676-677-678-679-680-681-682-683-684-685-686-687-688-689-690-691-692-693-694-695-696-697-698-699-700-701-702-703-704-705-706-707-708-709-710-711-712-713-714-715-716-717-718-719-720-721-722-723-724-725-726-727-728-729-730-731-732-733-734-735-736-737-738-739-740-741-742-743-744-745-746-747-748-749-750-751-752-753-754-755-756-757-758-759-760-761-762-763-764-765-766-767-768-769-770-771-772-773-774-775-776-777-778-779-780-781-782-783-784-785-786-787-788-789-790-791-792-793-794-795-796-797-798-799-800-801-802-803-804-805-806-807-808-809-810-811-812-813-814-815-816-817-818-819-820-821-822-823-824-825-826-827-828-829-830-831-832-833-834-835-836-837-838-839-840-841-842-843-844-845-846-847-848-849-850-851-852-853-854-855-856-857-858-859-860-861-862-863-864-865-866-867-868-869-870-871-872-873-874-875-876-877-878-879-880-881-882-883-884-885-886-887-888-889-890-891-892-893-894-895-896-897-898-899-900-901-902-903-904-905-906-907-908-909-910-911-912-913-914-915-916-917-918-919-920-921-922-923-924-925-926-927-928-929-930-931-932-933-934-935-936-937-938-939-940-941-942-943-944-945-946-947-948-949-950-951-952-953-954-955-956-957-958-959-960-961-962-963-964-965-966-967-968-969-970-971-972-973-974-975-976-977-978-979-980-981-982-983-984-985-986-987-988-989-990-991-992-993-994-995-996-997-998-999-1000-1001-1002-1003-1004-1005-1006-1007-1008-1009-1010-1011-1012-1013-1014-1015-1016-1017-1018-1019-1020-1021-1022-1023-1024-1025-1026-1027-1028-1029-1030-1031-1032-1033-1034-1035-1036-1037-1038-1039-1040-1041-1042

## Task 1.2

Ans: Steps for converting the ER diagram

- \* Entity type to table
  - \* All single-valued attribute becomes column for table
  - \* A key attribute of entity type represented by primary key
  - \* The multivalued attribute is represented by separate value.
  - \* Derived attribute represented by separate value.
  - \* Composite attribute represented by separate attribute and not considered in table
- Using the values you convert the diagram and assign mapping between tables.

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

Ans/2/16

Result: Hence, the relationship model of college that existing and management system using ER model was completed

STUDENT			
Student-Id (PK)	name	email	academic year

DEPARTMENT	
dept-Id (PK)	dept-name

COURSE			
course-name	course-Id (PK)	grades offered	prerequisites
course-hypr			

SLOT	
slot-Id (PK)	slot-hypr

Instructor	
date	

Task 1

Arm: Steps

for

\* En

\* A

\*

\*

\*

\*

\*

can be

Resu

Cell

my

Output : employee successfully  
Table create

Output

EMPLOYEE-ID      Int  
EMPLOYEE-NAME    VARCHAR(50)  
ADDRESS-VOUCHER   VARCHAR(100)  
AGE NUMBER        INT



EMPLOYEE-ID	EMPLOYEE-NAME	ADDRESS-VOUCHER	AGE NUMBER
1	John Doe	123 Main St	30
2	Jane Smith	456 Oak Ave	25
3	Mike Johnson	789 Pine Rd	35
4	Sarah Brown	101 Elm St	28
5	David Wilson	202 Maple Dr	32

12/8/25 10/09  
Task 2.8

Aim: To create a database model. As an example of SQL command and PDL command and data base system implementation of SQL with example  
\* create  
\* Alter  
\* Drop

PDL commands: It is used to create new table

Query:

SQL:

```
CREATE TABLE EMPLOYEE (EMPLOYEE-ID  
NUMBER(5) PRIMARY KEY  
EMPLOYEE-NAME VARCHAR(50),  
ADDRESS VARCHAR(200),  
AGE NUMBER(3));
```

2. DESCRIBE TABLE: Display the structure of table.

Query:

SQL

DESC EMPLOYEE.



Output  
Table EMPLOYEE dropped.

Output  
Table altered



CREATE TABLE EMPLOYEE  
(  
EMPLOYEE\_ID NUMBER(4) NOT NULL,  
LAST\_NAME VARCHAR2(25) NOT NULL,  
FIRST\_NAME VARCHAR2(25) NOT NULL,  
EMAIL VARCHAR2(25) NOT NULL,  
PHONE VARCHAR2(20) NOT NULL,  
HIRE\_DATE DATE NOT NULL,  
JOB\_ID VARCHAR2(10) NOT NULL,  
SALARY NUMBER(8,2) NOT NULL,  
COMMISSION\_PCT NUMBER(2,1) NOT NULL,  
CONSTRAINT EMPLOYEE\_PK PRIMARY KEY (EMPLOYEE\_ID),  
CONSTRAINT EMPLOYEE\_FK FOREIGN KEY (JOB\_ID) REFERENCES JOB (JOB\_ID)  
)

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:

Sal

ALTER TABLE

NUMBER (8,2).

EMPLOYEE ADD SALARY

EMPLOYEE\_ID EMPLOYEE\_NAME SALARY

Ravi

12000

DML commands

1. Insert INTO:

Adding a new value

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 new updated

Output

1 new deleted.

Chronic

11/2/2008

01/07/2016	01/07/2016
------------	------------

3118

	7
	6
	5
	4
	3
	2
	1
	0

5/9/23

standards  
PAL are created

create a debate and give excellent



SELECT WITH WHERE CLAUSE

SELECT \* FROM EMPLOYEE WHERE

Age = 28

EMPLOYEE-ID	EMPLOYEE-NAME	AGE
101	RAVE	28

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

5/8/23

the conceptual design of commands. This task DDL and DML are executed successfully.

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

Ask: To implement DDL and DML commands with constraints.

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

DML - (Data manipulation language) → Insert, update, Delete, 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 C

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(20) UNIQUE,

Address VARCHAR(100)

;

CREATE TABLE DEPARTMENT C

Department ID INT Primary key,

Department VARCHAR(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)

3:

output: Table Employee created.

1.2 ALTER TABLE:

ALTER TABLE EMPLOYEE ADD

EMAIL VARCHAR2(50)

ALTER TABLE EMPLOYEE MODIFY

CONTACT NUMBER

VARCHAR2(20)

output

Table Employee altered.

EMP_ID	EMP_NAME	E-MAIL	EMP_CONTACT NUMBER
101	RAVE	RAVE@gmail.com	7012571234

### 1.3 TRUNCATE TABLE

TRUNCATE TABLE Department


### 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,

Row, M, 35, 101, Chennai, Row@gmail.com.

Insert INTO Department (Department name,  
Department ID,  
Manager ID.



Employee ID	Employee name	Gender	Age	Salary
103	Rafael	M	37	Rafael

### 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  
 Results: 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  
 JOIN Department d ON e.Department =  
 d.Department

Employee Name	Department name	Assigned date	Role
Rave	CSE	2025	Developer

Output:

Employee name	ID	Age	place	Gender	Role
Rafiah	102	35	Chennai	M	Developer
Rave	103	35	Tamil	M	Developer

VEL TECH	
EX NO.	201
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	15
SIGN WITH DATE	19/8/24

Result: The task to implement DDL card and commands for our task-1 entry in RDBMS has been completed successfully

19/8/25

5.2.4.1/8

### TASK 3-1 Using clauses, operators and functions in queries

Table: Implementation of DDL commands using clauses, operators and functions

in queries:

- \* Insert table
- \* Select table
- \* Update table
- \* Delete table

#### Objective:

- \* To understand the different phases 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 queries

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

2. field-n) values ( data-1, data-2, ..., data-n );

Example: SQL > Insert into member value (104, Shyam, HR, male);

Employee name	EMP ID	Age	Place	gender	Role.
Rajesh	102	35	Chennai	M	Developer
Ravi	103	38	Tamilnadu	M	Developer
Shaan	104	30	Karnataka	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	Tirunelveli	M	Developer
Valan	104	30	Karur	M	HR

DELETE FROM : Used to delete all the

records  
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	Tirunelveli	M	Developer
Valan	104	30	Karur	M	HR



Truncate : Delete all values and structure members  
 Example : TRUNCATE employees

### Queries

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

### Query :

```
SELECT last name,  
last - name, salary  
FROM employees  
WHERE last - name
```

Like 'Y.M';

### Output

Employee name	ID	Age	Place	gender	Religion
Ravi	103	38	Tiruchy	M	Devedar

2. List the salary where between clause and operator

### Query :

```
select * from emp where  
salary between 1900 and 2000;
```



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

Query: Employee, places;  
 SELECT FROM book places  
 COUNT (\*) NO employees, publisher  
 GROUP BY publisher  
 ORDER BY publisher

Output	places
Employee name	channel
Ravi	channel
Rajesh	channel

VEL TECH	
EX NO.	3.3
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	20
DATE	

Result: ~~SQL~~ To implement the executed.

SQL commands are successfully using clauses & operators and function in queries template 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 Employee;
```

#### Explanation:

\* Count(\*) counts how many rows (Employee) are in the table.

\* AS Total - <sup>Employee</sup>~~Students~~ gives a user friendly column name.



2) Find the highest <sup>Salary</sup> ~~marks~~ obtained by a student.

```
SELECT MAX(salary) AS Highest
- salary
FROM EMPLOYEE
```

Output

Employee name	Id	Salary
Ram	103	40000

3) Find the minimum salary among the employees

Employee name	ID	Salary
Rafesh	104	15,000

4) Find the highest salary among the employees

Employee name	Id	Salary
Ram	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

VELTECH	
EX No.	32
PERFORM	✓
RES	5
VIVA	5
REVIEW	-
TO	W
26/4/21	

Result: Thus ~~20~~ commands  
Aggregate functions  
Executed successfully based on  
Employee is created successfully.

21/9/25

## 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: Independent Nested Query  
Query:

SELECT Employee Name

FROM Employee

WHERE EMP\_ID = (101

SELECT EMP\_ID

FROM Employee

WHERE EMP\_NAME = 'Ravi');

Output

Ravi



## 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
```

Query

Employee

= t2.Column

```
SELECT s. Student 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  $\rightarrow$  the student is returned.

Answer out put for query

Rajesh  
Ravi  
Valan  
tamilarasan  
Shaan

Difference between the query of independent  
correlated

Feature	Independent	Correlated
	Subquery	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 place in  
employee

checking  
enrollment of  
Employee

Output

Rajesh  
Ravi  
tamilanathan  
sharan  
kaavanan  
valan

#### VEL TECH

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

Result :

To Implement the independent  
and correlated Nested queries executed  
successfully



9/9/25

## Task 5 - Writing Join Queries Equivalent and/or 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 - name 1, table - name 2  
Where table - name 1. column name = table -  
name 2. column name;
```

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 (  
  ProjectID VARCHAR(10), PRIMARY KEY,
```

```
  ProjectName VARCHAR(50),
```

```
  DeptID INT,
```

```
  FOREIGN KEY (DeptID)
```

```
CREATE TABLE Assignments
```

```
  AssignmentID INT PRIMARY KEY,
```

```
  EmployeeID INT,
```

```
  ProjectID VARCHAR(10);
```

```
CREATE TABLE Hierarchy (  
  Employee INT,
```

```
  ManagerID 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, 'Dana', 202, NULL);

Insert Into Projects Values

(P1, 'Website Redesign', 201),

(P2, 'Marketing', 202),

(P3, 'API Development', 201)

(P4, 'Social Media Campaign', 203);

Insert Into Hierarchy Values.

(1, 3),

(2, 4);

SELECT \* FROM employee\_hirarchy;

VEL TECH	
EX NO.	5
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	1
TOTAL (20)	16
SIGN WITH DATE	2

Result: The Implementation of SQL commands for Employee using joins and recursive queries successfully.



16/9/25

## Task 6 Exception, Triggers and View query for employee

Aim: To perform an action whenever records

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=106;



## Output

Employee-id	first-name	last-name	email	Salary
101	Rafesh	Kharina	RK@ gmail.com	30,000
102	Valan	uday	VV@ gmail.com	45,000
103	Ravi	-	Ravi@ gmail.com	15,000
104	Shaan	-	Shaan@ gmail.com	40,000

Triggers : Special type stored procedure that automatically runs occur on database

Query

CREATE TRIGGER before\_employee\_salary  
BEFORE UPDATE ON employees update  
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 :

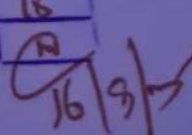
UPDATE employees

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)	8
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	1
TOTAL (20)	16
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, function, Loops

Aim:

To implement PL/SQL procedure, function 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 section>

Begin

          <executable 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 and y (10\*12)  
end;

output multiplication of x and y 120

Declare

number, 3) : 100;

Begin: If (htd=10) then

dbms-output-put-line

Else if (htd=50) then ('value of htd is 50')

dbms-output-put-line ('value of htd is

50')



```
Else If (chcd=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 ~~LLouterloop~~

For led IN 1...2 loop

LLInner-loop

For Pd IN 1...2 loop

dbms\_output.line

used "" and oed a ; || oed);

End loop inner-loop;

End loop outer-loop;

End;

hed is : 1 and oId is : 1  
hed is : 1 and oId is : 2  
hed is : 2 and oId is : 1  
hed 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

prun-add nos If cursor num-es is

select she-id from students;

v-Id Number;

v-ft Odd Boolean

v-p Number;

Begin

Open num-cur;

Loop Fetch num-cur into v-id;  
Exit.

while v-count < n loop

v-is-prime := TRUE

for i in 2 ... Trunc(sqrt(v-num))

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)	2
RESULT AND ANALYSIS (5)	6
VIVA VOCE (5)	3
RECORD (5)	1
TOTAL (20)	18
SIGN WITH DATE	

23/9/15

Result: Thus the Implementation of PUSP Function and loops on database has been completed successfully.