CREATE TABLE emp2 (Emp\_id INT PRIMARY KEY,Emp\_name VARCHAR(50),Dept VARCHAR(50),Designation VARCHAR(50),Salary DECIMAL(10, 2),Address VARCHAR(100),city VARCHAR(20));

INSERT INTO emp2 values(1, 'Vikki', 'IT', 'MD', 2000000, '124 Jal vayu vihar', 'Hubli')

INSERT INTO emp2 values(2, 'Rosita', 'IT', 'Manager', 30000, '124 Jal vayu vihar', 'Hubli')

INSERT INTO emp2 values(3, 'leo', 'HR', ' Manager', 25000, '12 Leo Street', 'New York')

INSERT INTO emp2 values(4, 'Raj', 'HR', 'Employee', 20000, '5th Apple Street', 'Trichy')

INSERT INTO emp2 values(5, 'Messi', 'sales', 'Employee', 15000, '11 Orange Street' ,'Berlin')

SELECT \* FROM emp2;



**a) Select the Names of Employees who are working as Manager**

1. SELECT Emp\_name FROM emp2 WHERE Designation = 'Manager';



**B)Give all Employees a 10% rise in salary**

2. UPDATE emp2 SET salary = salary \* 1.10;



SELECT \* FROM emp2;



**C) Find the Address of employees who live in same cities**

3. SELECT emp\_name, city FROM emp2 WHERE city IN (SELECT city FROM emp2 GROUP BY city HAVING COUNT(\*) > 1);



**d) Get employees who get salary above 25000**

4. SELECT \* FROM emp2 WHERE salary > = 25000;



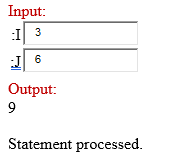
**e) List the employees who belongs to the same department**

5. SELECT emp\_name, dept FROM emp2 WHERE dept IN (SELECT dept FROM emp2 GROUP BY dept HAVING COUNT(\*) > 1);



**1. PL/SQL program to add two number.**

Declare **OUTPUT:**

i integer;

j integer;

k integer;

begin

i:=:i;

j:=:j;

k:=i+j;

dbms\_output.put\_line(k);

end;

/

**Create a procedure for student mark details.**

create or replace procedure markdetails1

(

id in number,

m1 in number,

m2 in number,

m3 in number,

m4 in number,

m5 in number,

total out number,

per out number,

grade out varchar2

)

is

gradeA number(5,2):=90;

gradeB number(5,2):=80;

gradeC number(5,2):=70;

gradeD number(5,2):=60;

begin

total:=m1+m2+m3+m4+m5;

per:=total/500\*100;

if per>=gradeA then

grade:='A';

elSif per>=gradeB and per<gradeA then

grade:='B';

elSif per>=gradeC and per<gradeB then

grade:='c';

elSif per>=gradeD and per<gradeC then

grade:='D';

else

grade:='F';

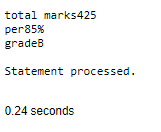
end if; **OUTPUT;**

end;

/

Declare

total1 number**; OUTPUT:**

per1 number(9,2);

grade1 varchar2(1);

begin

markdetails1(124,75,80,85,90,95, total=>total1, per=>per1, grade=>grade1);

dbms\_output.put\_line('total marks'||total1);

dbms\_output.put\_line('per'||per1||'%');

dbms\_output.put\_line('grade'||grade1);

end;

/

**EB BILL UNIT**

create or replace procedure calc\_eb\_bill(p\_units in number, p\_rate\_per\_unit in number:=3.50,p\_fixed\_charges in number:=50.00,p\_tax\_rate in number:=0.05,p\_eb\_bill out number)

is

begin

declare

energy\_charge number(10,2):=p\_units\*p\_rate\_per\_unit;

begin

declare

tax\_amount number(10,2):=energy\_charge\*p\_tax\_rate;

begin

p\_eb\_bill:=energy\_charge+tax\_amount+p\_fixed\_charges;

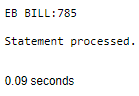
end; **OUTPUT:**

end;

end;

/

Declare **OUTPUT:**

eb\_bill number(10,2);

begin

calc\_eb\_bill(200,p\_eb\_bill=>eb\_bill);

dbms\_output.put\_line('EB BILL:'||eb\_bill);

end;

**PL/SQL program to swap two number**

declare

a number;

b number;

temp number;

begin

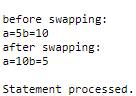
a:=5;

b:=10;

dbms\_output.put\_line('before swapping:');

dbms\_output.put\_line('a='||a||'b='||b);

temp:=a; **OUTPUT:**

a:=b;

b:=temp;

dbms\_output.put\_line('after swapping:');

dbms\_output.put\_line('a='||a||'b='||b);

end;/

create or replace function calculate\_area(length in number,width in number)

return number

is

begin **OUTPUT:**

return length\*width; 

end;

select calculate\_area(5,10) from dual;

**Trigger Creation**

create table moviess (year number(5), name varchar(20));

insert into moviess values(2014, 'Interstellar')

create trigger trigger1 before insert or update or delete on moviess for each row

begin

raise\_application\_error(-20010,'You cannot do manipulation');

end;

/ **OUTPUT:**

update moviess set year = year+1; 

delete from moviess where year= 2004;



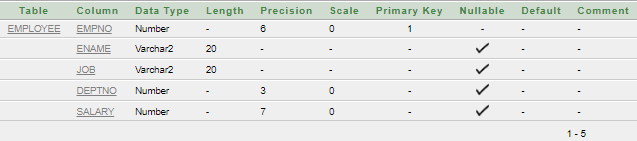
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

PREVIOUSLY IN SQL

**CREATE**

create table employee(empno number(6),ename varchar(20),job varchar(20),

deptno number (3),salary number(7),primary key(empno));

DESC emp; 

**INSERT VALUES**

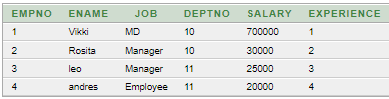
INSERT INTO employee values(1, 'Vikki', 'MD', 2000000)

INSERT INTO employee values(2, 'Rosita', 'Manager', 30000)

INSERT INTO employee values(3, 'leo', ' Manager', 25000)

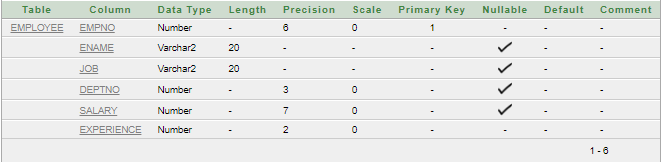
INSERT INTO employee values(4, 'andres', 'Employee', 20000)

select \* from employee;

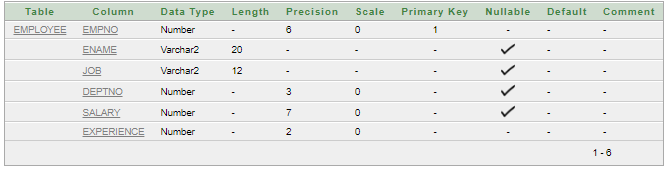
****

**a)Add a column experience to the employee table,experience numeric null allowed**

alter table **employee** add(experience number(2) not null);



**alter table employee modify(job varchar(12));**

****

truncate table employee;





drop table employee;



**VIVA VOCE**

**What is DDL?**

DDL (Data Definition Language): DDL is a subset of SQL (Structured Query Language) used to define the structure and schema of a database. It includes commands like CREATE, ALTER, DROP, and TRUNCATE, allowing users to create, modify, and delete database objects such as tables, indexes, views, and schemas. DDL statements are used to define the organization, structure, and constraints applied to the data within a database

**What is DML?**

DML(Data Manipulation Language): DML is another subset of SQL used to manage data within the database. DML commands primarily include SELECT, INSERT, UPDATE, and DELETE. These commands facilitate the retrieval, insertion, modification, and removal of data stored in the database tables. DML operations focus on manipulating the actual data contained within the database, allowing users to interact with, retrieve, modify, and delete specific data records.

**SQL Data Definition:**

- **Specifying Tables:**  Using `CREATE TABLE` to define the structure of tables, specifying columns, data types, and constraints.

-  **Data Types**: Defining the type of data each column can hold, like integers, strings, dates, etc.

- **Constraints**: Applying rules like primary keys, foreign keys, unique constraints, default values, and check constraints to maintain data integrity.

**- Simple Statements**: Basic `SELECT`, `INSERT`, `UPDATE`, and `DELETE` statements to retrieve, add, modify, and remove data from tables.

- **Complex SELECT Queries:** Utilizing `JOIN` operations (e.g., INNER JOIN, LEFT JOIN, RIGHT JOIN) and nested queries to perform more sophisticated data retrieval based on multiple tables and conditions.

- **Actions and Triggers:**  Using `TRUNCATE`, `ALTER`, `DROP` statements, and creating triggers to automate actions based on certain events like INSERT, UPDATE, or DELETE.

**Views, Schemas, and Altering:**

- **Views**: Creating virtual tables based on the result of a SELECT query, simplifying complex queries or restricting access to certain columns or rows.

- **Altering Schemas:** Modifying existing tables using `ALTER TABLE` statements to add or drop columns, modify data types, or change constraints.

**Relational Algebra:**

- **Definition**: Understanding relational algebra as a theoretical framework for working with relations (tables) using mathematical operations.

- **Relations as Sets:**  Treating tables as sets of data and applying operations (such as union, intersection) to manipulate these sets.

- **Operations**: Exploring basic operations like `SELECT`, `PROJECT`, `JOIN`, `UNION`, `INTERSECTION`, etc., to perform set-based operations on tables.

**Normalization Theory:**

- **Functional Dependencies**: Understanding how attributes in a relation depend on each other and how changes in one attribute affect others.

- **Normalization Levels (2NF, 3NF, BCNF, 4NF, 5NF)**: Studying various normal forms to eliminate data redundancy and dependency issues by organizing data into well-structured tables while ensuring data integrity.

**PL/SQL Procedure:**

A PL/SQL procedure is a named block of code within the PL/SQL language, stored in a database, that performs a specific task or set of tasks. It can take inputs (parameters), perform operations, execute SQL statements, and return results. Procedures help encapsulate logic, promoting reusability and modularity in database programming.

**Purpose of PL/SQL Procedure**:

- Encapsulate a set of SQL and procedural statements to perform specific tasks.

- Reusable blocks of code that can be called multiple times within the database or by applications.

- Promote code organization and maintenance by grouping related tasks together.

**PL/SQL Trigger Creation:**

A PL/SQL trigger is a named PL/SQL block associated with a table in a database. It automatically executes ("fires") in response to specific events (e.g., `INSERT`, `UPDATE`, `DELETE`) occurring on the associated table. Triggers can perform actions before or after the triggering event.

**Purpose of PL/SQL Trigger:**

- Enforce business rules or data validation before inserting, updating, or deleting records.

- Implement audit trails by logging changes made to a table.

- Automate actions based on specific database events without manual intervention.

- Maintain data integrity and enforce constraints.

PL/SQL procedures are reusable blocks of code designed to perform specific tasks within a database, while PL/SQL triggers are event-driven code blocks that automatically execute in response to specified database events, allowing for automated actions and enforcing rules or logic related to those events.

[AI -DBMS - Lab MAnual.pdf](AI%20-DBMS%20-%20%20Lab%20MAnual.pdf)



