Theory -

- Stored procedure vs functions.

Stored procedure are used to run a single query again and again without writing it every time to avoid ~~time loss~~ query redundency and optimize time.

eg.
if we want to call a query to display customers on a webpage we can call the procedure rightaway without writing the query again & again.
just call storedprocedure_name;

functions are same as storedprocedure just to as adding parameters to stored procedures eg.

```
stored procedure_name ( int datatype variable name)
{
  Select from table where table col-name = var_name;
}
```

eg. if we want to pass a user value from web application we can directly pass value to stored procedure so that we can pass the variable to query & result can be displayed.

- Triggers -

Triggers are event which we can execute on a certain time or for some check values.

eg
if we want to display a ~~trigger~~ some warning on a insurance portal where the login user insurance is about to expire. you can create a trigger on date & display a message that "your insurance is about to expire".

name, Age, City
, 111, 1),
'Shreya',
Pur',

select count(*), city from customers;

3. select count(*) from product where productPrice > 50,000;

4. select productName from Products where productPrice = max(product price);

5. create view productbyCustomers
AS
(select productName, productPrice from product
where ProductId IN (select pid from customers);)
;

6. select * from customers where count(select ~~distinct~~ count(city) from customers) > 1;

7. Select customerName from Customers where pId IN (
Select productId where productName = 'mobile'));

PL/SQL

1. Delimeter $$ BEGIN
create procedure showTable
AS
Show Tables;
END $$
Delimeter;

2. Delimeter $$
BEGIN.
create procedure highestprice (int n)
BEGIN
Select max(ProductPrice) from Product where price = n;
END $$
Delimeter;

Table - Insertion -

1. insert into customers (customerId, customerName, Age, old, pId) values (1, 'Ramesh', 18, 'Pune', 111, 1), (2, 'Jayesh', 15, 'Solapur', 555, 4), (3, 'Shreya', 45, 'Pune', 222, 1), (4, 'Rani', 23, 'Nagpur', 444, 3), (5, 'Aishwarya', 23, 'Banglore', 111, 2), (6, 'Shruti', 30, 'Mumbai', 222, 5), (7, 'Tushar', 50, 'Delhi', 111, 1), (8, 'Aryan', 10, 'Mumbai', 555, 1), (9, 'Akshay', 18, 'Nagpur', 222, 2), (10, 'Varun', 25, 'Indapur', 444, 4);

Orders -

Insert into orders (orderNumber, orderDate, status) values (111, '2023-03-01', 'Yes'), (222, '2024-03-02', 'No'); (333, '2023-03-05', 'Yes'), (444, '2024-03-06', 'Yes', (555, '2024-03-07', 'No');

Product -

Insert into Product (productId, productName, productPrice) values (1, 'mobile', 50000), (2, 'TV', 30000), (3, 'Laptop', 80000), (4, 'Scooty', 150000), (5, 'AC', 25000);

Aditya J. Raut 9049052994 SQL - Assignment - 2
ITPM - Dec - 2024.

Part 1 →

1. create database finalTest
2. create three tables in finalTest Database.
   - Customers - customerId, CustomerName, Age, city.

1. create database FinalTest;

   Use FinalTest;

2. 1. create table Customers (customerId int, customerName varchar(20), Age int, city varchar(20), oId int, pId int);

   2. Create table orders (orderNumber int, orderDate date, status char(8));

3.

2. 1. create table orders (orderNumber int primary key, orderDate date, status char(8));

   2. create table product (productId int primary key, productName varchar(20), productPrice decimal(10,2));

   3. create table customers (customerId int primary key, customerName varchar(20), Age int, city varchar(20), oid int, pid int);

   → alter table customers add constraint foreign key on customers.oid refers to orders.orderNumber);

   → alter table customers add constraint foreign key on customers.pid refers to product.productId);