Q1)
Create a table customer (cust_no varchar (5), cust_name varchar (15), age number, phone varchar (10))
a) insert 5 records and display it
b) add new field d_birth with date datatype
c) create another table cust_phone with fields cust_name and phone from customer table
d) remove the field age
e) change the size of the cust_name to 25
f) delete all the records from the table
g) rename the table

Solution:

Create the customer table

```
CREATE TABLE customer (
   cust_no VARCHAR(5),
   cust_name VARCHAR(15),
   age INTEGER,
   phone VARCHAR(10)
);
```

Insert 5 records into the customer table

```
INSERT INTO customer (cust_no, cust_name, age, phone) VALUES
('C1', 'Alice', 30, '1234567890'),
('C2', 'Bob', 25, '0987654321'),
('C3', 'Charlie', 35, '1112223334'),
('C4', 'David', 28, '5556667778'),
('C5', 'Eve', 22, '9998887776');
```

Display the records

```
SELECT * FROM customer;
```

Add the d_birth field to the customer table

```
ALTER TABLE customer ADD COLUMN d_birth DATE;
```

Create the cust_phone table from customer table

```
CREATE TABLE cust_phone AS SELECT cust_name, phone FROM customer;
```

Remove the age field from the customer table

```
ALTER TABLE customer DROP COLUMN age;
```

Change the size of cust_name to 25

```
ALTER TABLE customer ALTER COLUMN cust_name TYPE VARCHAR(25);
```

Delete all records from the customer table

DELETE FROM customer;

Rename the customer table to cust

ALTER TABLE customer RENAME TO cust;

Drop the cust table

DROP TABLE cust;

Q2)

Create a table sales_man (salesman_no primary key, s_name not null, place, phone unique)
Create table sales_order(order_no primary key , order_date not null ,salesman_no foreign key references salesman_no in sales_man , del_type values should be either P or F (check constraints) ,order_status values should be 'Inprocess','Fullfilled','Backorder', 'Cancelled' (check constraints))
a) Insert few records in both tables
b) Delete primary key from sales_man table
c) Delete Foreign key and Check constraints from sales_order table
d) Add primary key in sales_man using ALTER TABLE
e) Add foreign key and CHECK constraints in sales_order table using ALTER TABLE

Solution

Create the sales_man table

CREATE TABLE sales_man (salesman_no SERIAL PRIMARY KEY, s_name VARCHAR(50) NOT NULL, place VARCHAR(50), phone VARCHAR(15) UNIQUE);

Create the sales_order table

CREATE TABLE sales_order (order_no SERIAL PRIMARY KEY, order_date DATE NOT NULL, salesman_no INTEGER, del_type CHAR(1) CHECK (del_type IN ('P', 'F')), order_status VARCHAR(20) CHECK (order_status IN ('Inprocess', 'Fullfilled', 'Backorder', 'Cancelled')),FOREIGN KEY (salesman_no) REFERENCES sales_man (salesman_no));

Insert records into sales_man table

INSERT INTO sales_man (s_name, place, phone) VALUES ('Alice', 'New York', '1234567890'), ('Bob', 'Los Angeles', '0987654321'), ('Charlie', 'Chicago', '1112223334');

Insert records into sales_order table
INSERT INTO sales_order (order_date, salesman_no, del_type, order_status) VALUES ('2023-05-01', 1, 'P', 'Inprocess'), ('2023-05-02', 2, 'F', 'Fullfilled'), ('2023-05-03', 3, 'P', 'Backorder');

Delete primary key from sales_man table

ALTER TABLE sales_man DROP CONSTRAINT sales_man_pkey;

Delete foreign key constraint from sales_order table
ALTER TABLE sales_order DROP CONSTRAINT sales_order_salesman_no_fkey;

Delete check constraints from sales_order table
ALTER TABLE sales_order DROP CONSTRAINT sales_order_del_type_check, DROP CONSTRAINT sales_order_order_status_check;

Add primary key to sales_man table
ALTER TABLE sales_man ADD CONSTRAINT sales_man_pkey PRIMARY KEY (salesman_no);

Add foreign key constraint to sales_order table
ALTER TABLE sales_order ADD CONSTRAINT sales_order_salesman_no_fkey FOREIGN KEY (salesman_no) REFERENCES sales_man (salesman_no);

Add check constraints to sales_order table
ALTER TABLE sales_order ADD CONSTRAINT sales_order_del_type_check CHECK (del_type IN ('P', 'F')), ADD CONSTRAINT sales_order_order_status_check CHECK (order_status IN ('Inprocess', 'Fullfilled', 'Backorder', 'Cancelled'));

Q3)
Create a table Hospital with the fields (doctorid,doctorname,department,qualification,experience).
Write the queries to perform the following.
a) Insert 5 records
b) Display the details of Doctors
c) Display the details of doctors who have the qualification ‘MD’
d) Display all doctors who have more than 5 years experience but do not have the qualification ‘MD’
e) Display the doctors in ‘Skin’ department
f) update the experience of doctor with doctored=’D003’ to 5
g) Delete the doctor with DoctorID=’D005’

Solution
Create the Hospital table
CREATE TABLE Hospital (doctorid VARCHAR(10) PRIMARY KEY, doctorname VARCHAR(50), department VARCHAR(50), qualification VARCHAR(50), experience INTEGER);

Insert 5 records into the Hospital table

INSERT INTO Hospital (doctorid, doctorname, department, qualification, experience) VALUES ('D001', 'Dr. Alice', 'Cardiology', 'MD', 10),('D002', 'Dr. Bob', 'Neurology', 'PhD', 8), ('D003', 'Dr. Charlie', 'Orthopedics', 'MS', 6), ('D004', 'Dr. David', 'Skin', 'MD', 4), ('D005', 'Dr. Eve', 'Pediatrics', 'MBBS', 3);

Display the details of all doctors
SELECT * FROM Hospital;

Display the details of doctors who have the qualification 'MD'

SELECT * FROM Hospital WHERE qualification = 'MD';

Display all doctors who have more than 5 years experience but do not have the qualification 'MD'

SELECT * FROM Hospital WHERE experience > 5 AND qualification != 'MD';

Display the doctors in 'Skin' department

SELECT * FROM Hospital WHERE department = 'Skin';

Update the experience of doctor with doctorid='D003' to 5

UPDATE Hospital SET experience = 5 WHERE doctorid = 'D003';

Delete the doctor with doctorid='D005'

DELETE FROM Hospital WHERE doctorid = 'D005';

Q4)

Create the following tables
Bank_customer (accno primary key, cust_name, place)
Deposit (accno foreign key, deposit_no, damount)
Loan (accno foreign key loan_no, Lamount)
Write the following queries
a) Display the details of the customers
b) Display the customers along with deposit amount who have only deposit with the bank
c) Display the customers along with loan amount who have only loan with the bank
d) Display the customers they have both loan and deposit with the bank
e) Display the customer who have neither a loan nor a deposit with the bank

Solution
Create the Bank_customer table

CREATE TABLE Bank_customer (accno VARCHAR(10) PRIMARY KEY, cust_name VARCHAR(50), place VARCHAR(50));

Create the Deposit table
CREATE TABLE Deposit (accno VARCHAR(10) REFERENCES Bank_customer(accno),deposit_no SERIAL PRIMARY KEY, damount NUMERIC);

Create the Loan table
CREATE TABLE Loan (accno VARCHAR(10) REFERENCES Bank_customer(accno), loan_no SERIAL PRIMARY KEY,Lamount NUMERIC);

Display the details of all customers
SELECT * FROM Bank_customer;

Display the customers along with deposit amount who have only deposit with the bank
SELECT bc.accno, bc.cust_name, bc.place, d.damount FROM Bank_customer bc JOIN Deposit d ON bc.accno = d.accno LEFT JOIN Loan l ON bc.accno = l.accno WHERE l.accno IS NULL;

Display the customers along with loan amount who have only loan with the bank
SELECT bc.accno, bc.cust_name, bc.place, l.Lamount FROM Bank_customer bc JOIN Loan l ON bc.accno = l.accno LEFT JOIN Deposit d ON bc.accno = d.accno WHERE d.accno IS NULL;

Display the customers who have both loan and deposit with the bank
SELECT bc.accno, bc.cust_name, bc.place, d.damount, l.Lamount FROM Bank_customer bc JOIN Deposit d ON bc.accno = d.accno JOIN Loan l ON bc.accno = l.accno;

Display the customers who have neither a loan nor a deposit with the bank
SELECT bc.accno, bc.cust_name, bc.place FROM Bank_customer bc LEFT JOIN Deposit d ON bc.accno = d.accno LEFT JOIN Loan l ON bc.accno = l.accno WHERE d.accno IS NULL AND l.accno IS NULL;

Q5)

Create a table employee with fields (EmpID, EName, Salary, Department, and Age). Insert some records. Write SQL queries using aggregate functions and group by clause
A. Display the total number of employees.
B. Display the name and age of the oldest employee of each department.
C. Display the average age of employees of each department
D. Display departments and the average salaries
E. Display the lowest salary in employee table
F. Display the number of employees working in purchase department
G. Display the highest salary in sales department;
H. Display the difference between highest and lowest salary

Solution
Create the employee table
CREATE TABLE employee (EmpID SERIAL PRIMARY KEY, EName VARCHAR(50),Salary NUMERIC, Department VARCHAR(50), Age INTEGER);

Insert some records into the employee table
INSERT INTO employee (EName, Salary, Department, Age) VALUES ('Ameen', 50000, 'HR', 30), ('B', 60000, 'Sales', 28), ('C', 55000, 'HR', 35), ('D', 70000, 'IT', 32), ('E', 80000, 'Sales', 45), ('F', 45000, 'IT', 29), ('G', 48000, 'Purchase', 25), ('H', 62000, 'Purchase', 40), ('I', 75000, 'Sales', 50),
('J', 53000, 'HR', 26);

Display the total number of employees
SELECT COUNT(*) AS total_employees FROM employee;

Display the name and age of the oldest employee of each department
SELECT Department, EName, Age FROM employee WHERE (Department, Age) IN (SELECT Department, MAX(Age) FROM employee GROUP BY Department);

Display the average age of employees of each department
SELECT Department, AVG(Age) AS average_age FROM employee GROUP BY Department;

Display departments and the average salaries
SELECT Department, AVG(Salary) AS average_salary FROM employee GROUP BY Department;

Display the lowest salary in employee table
SELECT MIN(Salary) AS lowest_salary FROM employee;

Display the number of employees working in purchase department
SELECT COUNT(*) AS purchase_department_employees FROM employee WHERE Department = 'Purchase';

Display the highest salary in sales department
SELECT MAX(Salary) AS highest_salary_sales FROM employee WHERE Department = 'Sales';

Display the difference between highest and lowest salary
SELECT MAX(Salary) - MIN(Salary) AS salary_difference FROM employee;

Q6)

Create a table product with the fields (Product_code primary key, Product_Name, Category, Quantity, Price).
Insert some records Write the queries to perform the following.
a. Display the records in the descending order of Product_Name
b. Display Product_Code, Product_Name with price between 20 and 50
c. Display the details of products which belongs to the categories of ‘bath soap’, ‘paste’, or ‘washing powder’
d. Display the products whose Quantity less than 100 or greater than 500
e. Display the products whose names starts with 's'
f. Display the products which not belongs to the category 'paste'
g. Display the products whose second letter is 'u' and belongs to the Category 'washing powder'

Solution

Create the product table
CREATE TABLE product (Product_code VARCHAR(10) PRIMARY KEY,Product_Name VARCHAR(50), Category VARCHAR(50), Quantity INTEGER,Price NUMERIC);

Insert some records into the product table
INSERT INTO product (Product_code, Product_Name, Category, Quantity, Price) VALUES('P001', 'Soap', 'bath soap', 150, 25), ('P002', 'Shampoo', 'hair care', 80, 45), ('P003', 'Toothpaste', 'paste', 200, 30),

('P004', 'Detergent', 'washing powder', 50, 40), ('P005', 'Conditioner', 'hair care', 300, 35), ('P006', 'Body Wash', 'bath soap', 90, 50), ('P007', 'Mouthwash', 'oral care', 120, 20), ('P008', 'Laundry Powder', 'washing powder', 600, 60), ('P009', 'Shaving Cream', 'shaving', 70, 25), ('P010', 'Hand Soap', 'bath soap', 450, 15);

Display the records in the descending order of Product_Name
SELECT * FROM product ORDER BY Product_Name DESC;

Display Product_Code, Product_Name with price between 20 and 50
SELECT Product_code, Product_Name FROM product WHERE Price BETWEEN 20 AND 50;

Display the details of products which belongs to the categories of 'bath soap', 'paste', or 'washing powder'
SELECT * FROM product WHERE Category IN ('bath soap', 'paste', 'washing powder');

Display the products whose Quantity is less than 100 or greater than 500
SELECT * FROM product WHERE Quantity < 100 OR Quantity > 500;

Display the products whose names start with 's'
SELECT * FROM product WHERE Product_Name ILIKE 's%';

Display the products which do not belong to the category 'paste'
SELECT * FROM product WHERE Category != 'paste';

Display the products whose second letter is 'u' and belongs to the Category 'washing powder'
SELECT * FROM product WHERE Product_Name ILIKE '_u%' AND Category = 'washing powder';

Q7)

Create table supplier(supcode,sname,city)
Create table product (pcode,pname)
Create table supl_product(supcode,pcode,qty)
a) Get all pairs of supplier numbers such that the two suppliers are located in the same city.
b) Get supplier names for suppliers who supply product P2.
c) Get product numbers supplied by more than one supplier.
d) Get supplier numbers for suppliers who are located in the same city as supplier S1.
e) Get supplier names for suppliers who supply part P1.
f) Get the number of Suppliers, who are supplying at least one product.
g) For each product supplied, get the pcode. and the total quantity supplied for that part.

Solution
Create supplier table
CREATE TABLE supplier (supcode VARCHAR(10) PRIMARY KEY,sname VARCHAR(50) NOT NULL, city VARCHAR(50) NOT NULL);

Create product table
CREATE TABLE product (pcode VARCHAR(10) PRIMARY KEY, pname VARCHAR(50) NOT NULL);

Create supl_product table with foreign key constraints

```
CREATE TABLE supl_product (supcode VARCHAR(10) REFERENCES supplier(supcode),pcode VARCHAR(10) REFERENCES product(pcode), qty INTEGER,  PRIMARY KEY (supcode, pcode));
```

Get all pairs of supplier numbers such that the two suppliers are located in the same city.

```
SELECT s1.supcode, s2.supcode FROM supplier s1 JOIN supplier s2 ON s1.city = s2.city AND s1.supcode < s2.supcode;
```

Get supplier names for suppliers who supply product P2.

```
SELECT s.sname FROM supplier s JOIN supl_product sp ON s.supcode = sp.supcode WHERE sp.pcode = 'P2';
```

Get product numbers supplied by more than one supplier.

```
SELECT pcode FROM supl_product GROUP BY pcode HAVING COUNT(supcode) > 1;
```

Get supplier numbers for suppliers who are located in the same city as supplier S1.

```
SELECT s2.supcode FROM supplier s1 JOIN supplier s2 ON s1.city = s2.city WHERE s1.supcode = 'S1' AND s2.supcode != 'S1';
```

Get supplier names for suppliers who supply part P1.

```
SELECT s.sname FROM supplier s JOIN supl_product sp ON s.supcode = sp.supcode WHERE sp.pcode = 'P1';
```

Get the number of Suppliers, who are supplying at least one product.

```
SELECT COUNT(DISTINCT supcode) AS supplier_count FROM supl_product;
```

For each product, get the supplier codes and the total quantity supplied for that product.

```
SELECT pcode, supcode, SUM(qty) AS total_quantity FROM supl_product GROUP BY pcode, supcode;
```