

RDBMS LAB LIST 2023 (S4 BCA AND BSC CS)

Part 1 SQL

Program 1 : Customer Table

AIM

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 cutomer to cust
- H. drop the table

TABLE DESIGN

Table name : customer

Name	Type	description
cust_no	varchar(5)	customer number
cust_name		customer name
age	int	age
phone	varchar(10)	phone number
d_birth	date	date of birth

QUERIES

create table customer(cust_no varchar(5),cust_name varchar(15),age numeric,phone varchar(10));

/d customer;

Table "public.customer"		
Column	Type	Collation
cust_no	character varying(5)	
cust_name	character varying(15)	
age	numeric	
phone	character varying(10)	

a)

insert into customer values(1,'Raju',23,9495823456);

insert into customer values(2,'Zara',23,9447132324);

```
insert into customer values(3,'Adam',34,8089123456);
insert into customer values(4,'Sheena',23,7259123456);
insert into customer values(5,'Lisa',23,8891223344);
```

```
select *from customer;
```

cust_no	cust_name	age	phone
1	Raju	23	9495823456
2	Zara	23	9447132324
3	Adam	34	8089123456
4	Sheena	23	7259123456
5	Lisa	23	8891223344

(5 rows)

b)

```
alter table customer add d_birth date;
\d customer;
```

Table "public.customer"			
Column	Type	Collation	
cust_no	character varying(5)		
cust_name	character varying(15)		
age	numeric		
phone	character varying(10)		
d_birth	date		

c)

```
create table cust_phone as select cust_name,phone from customer;
select *from cust_phone;
```

cust_name	phone
Raju	9495823456
Zara	9447132324
Adam	8089123456
Sheena	7259123456
Lisa	8891223344

(5 rows)

d)

```
alter table customer drop age;
\d customer;
```

Table "public.customer"			
Column	Type	Collation	
cust_no	character varying(5)		
cust_name	character varying(15)		
phone	character varying(10)		
d_birth	date		

e)

```
ALTER TABLE customer ALTER COLUMN cust_name TYPE varchar(25);
\d customer;
```

Table "public.customer"			
Column	Type	Collation	
cust_no	character varying(5)		
cust_name	character varying(25)		
phone	character varying(10)		
d_birth	date		

f)

```
TRUNCATE customer;
select *from customer;
```

cust_no	cust_name	phone	d_birth
(0 rows)			

g)

```
ALTER TABLE customer RENAME TO cust;
\d cust;
```

Table "public.cust"			
Column	Type	Collation	
cust_no	character varying(5)		
cust_name	character varying(25)		
phone	character varying(10)		
d_birth	date		

h)

```
drop table cust;
```

```
DROP TABLE
```

Program 2 : Constraints

AIM

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', or 'Cancelled' (check constraints)).

- Insert few records in both tables
- Delete primary key from sales_man table
- Delete Foreign key and Check constraints from sales_order table
- Add primary key in sales_man using ALTER TABLE
- Add foreign key and CHECK constraints in sales_order table using ALTER TABLE

TABLE DESIGN

Table name : sales_man

Name	Type	Constraints	Description
salesman_no	int	primary key	salesman number
s_name	varchar(10)	not null	salesman name
place	varchar(10)		place
phone	int	unique	phone number

Table name : sales_order

Name	Type	Constraints	Description
order_no	int	primary key	order number
order_date	date	not null	order date
order_status	char(10)	not null, check ('inprocess', 'fullfilled', 'cancelled', 'backorder')	order status
salesman_no	int	foreign key – salesman(salesman_no)	salesman number
del_type	char(1)	check ('p', 'f')	delivery type

QUERIES

create table sales_man(salesman_no int primary key,s_name varchar(10) not null,place varchar(10),phone numeric(10) unique);
\d sales_man;

```
Table "public.sales_man"
  Column      |      Type       | Collation | Nullable | Default
-----+-----+-----+-----+-----
salesman_no   | integer         |           | not null |
s_name        | character varying(10) |           | not null |
place         | character varying(10) |           |          |
phone         | numeric(10,0)   |           |          |
Indexes:
  "sales_man_pkey" PRIMARY KEY, btree (salesman_no)
  "sales_man_phone_key" UNIQUE CONSTRAINT, btree (phone)
```

create table sales_order(order_no int primary key,order_date date NOT NULL,order_status char(10) NOT NULL check(order_status IN('Inprocess', 'Fullfilled', 'Cancelled', 'Backorder')), salesman_no int references sales_man (salesman_no), del_type char(1) check (del_type='F' or del_type='P'));
\d sales_order;

```
Table "public.sales_order"
  Column      |      Type       | Collation | Nullable | Default
```

```

-----+-----+-----+-----+-----
order_no | integer | | not null |
order_date | date | | not null |
order_status | character(10) | | not null |
salesman_no | integer | | |
del_type | character(1) | | |
Indexes:
    "sales_order_pkey" PRIMARY KEY, btree (order_no)
Check constraints:
    "sales_order_del_type_check" CHECK (del_type = 'F'::bpchar OR del_type = 'P'::bpchar)
    "sales_order_order_status_check" CHECK (order_status = ANY (ARRAY['Inprocess'::bpchar,
'Fullfilled'::bpchar, 'Cancelled'::bpchar, 'Backorder'::bpchar]))
Foreign-key constraints:
    "sales_order_salesman_no_fkey" FOREIGN KEY (salesman_no) REFERENCES
sales_man(salesman_no)

```

a)

```

insert into sales_man values (101, 'ananthu', 'feroke', 8137036211);
insert into sales_man values (102, 'fariz', 'chungam', 8137036231);
insert into sales_man values( 103, 'sheena', 'chelari', 9997036231);
insert into sales_man values(104, 'asla', 'chelari', 9687036231);
insert into sales_man values(105, 'jithin', 'tanur', 9687035671);

```

```
select*from sales_man;
```

salesman_no	s_name	place	phone
101	ananthu	feroke	8137036211
102	fariz	chungam	8137036231
103	sheena	chelari	9997036231
104	asla	chelari	9687036231
105	jithin	tanur	9687035671

(5 rows)

```

insert into sales_order values(1,'01-01-17', 'Inprocess',101,'F'),(2,'03-02-17', 'Fullfilled', 102, 'F'), (3,
'03-03-17', 'Fullfilled', 103, 'P'), (4, '03-03-17', 'Cancelled',104, 'F'), (5,'05-03-17', 'Backorder', 105, 'P');

```

```
select * from sales_order;
```

order_no	order_date	order_status	salesman_no	del_type
1	2017-01-01	Inprocess	101	F
2	2017-03-02	Fullfilled	102	F
3	2017-03-03	Fullfilled	103	P
4	2017-03-03	Cancelled	104	F
5	2017-05-03	Backorder	105	P

(5 rows)

b)

```

ALTER TABLE sales_man DROP constraint sales_man_pkey cascade;
\d sales_man;

```

Column	Type	Collation	Nullable	Default
--------	------	-----------	----------	---------

```

-----+-----+-----+-----+
salesman_no | integer | | not null |
s_name      | character varying(10) | | not null |
place       | character varying(10) | | |
phone       | numeric(10,0) | | |
Indexes:
    "sales_man_phone_key" UNIQUE CONSTRAINT, btree (phone)

```

\d sales_order;

```

Table "public.sales_order"
  Column      |      Type      | Collation | Nullable | Default
-----+-----+-----+-----+-----+
order_no      | integer        |           | not null |
order_date    | date           |           | not null |
order_status  | character(10)   |           | not null |
salesman_no   | integer        |           |          |
del_type      | character(1)    |           |          |
Indexes:
    "sales_order_pkey" PRIMARY KEY, btree (order_no)
Check constraints:
    "sales_order_del_type_check" CHECK (del_type = 'F'::bpchar OR del_type = 'P'::bpchar)
    "sales_order_order_status_check" CHECK (order_status = ANY (ARRAY['Inprocess'::bpchar, 'Fullfilled'::bpchar, 'Cancelled'::bpchar, 'Backorder'::bpchar]))

```

c)

```

ALTER TABLE sales_order DROP constraint sales_order_del_type_check;
ALTER TABLE sales_order DROP constraint sales_order_order_status_check;
\d sales_order;

```

```

Table "public.sales_order"
  Column      |      Type      | Collation | Nullable | Default
-----+-----+-----+-----+-----+
order_no      | integer        |           | not null |
order_date    | date           |           | not null |
order_status  | character(10)   |           | not null |
salesman_no   | integer        |           |          |
del_type      | character(1)    |           |          |
Indexes:
    "sales_order_pkey" PRIMARY KEY, btree (order_no)

```

d)

```

ALTER TABLE sales_man ADD primary key(salesman_no);
\d sales_man;

```

```

Table "public.sales_man"
  Column      |      Type      | Collation | Nullable | Default
-----+-----+-----+-----+-----+
salesman_no   | integer        |           | not null |
s_name        | character varying(10) | | not null |
place         | character varying(10) | | |
phone         | numeric(10,0) | | |
Indexes:
    "sales_man_pkey" PRIMARY KEY, btree (salesman_no)
    "sales_man_phone_key" UNIQUE CONSTRAINT, btree (phone)

```

e)

```
ALTER TABLE sales_order ADD FOREIGN KEY (salesman_no) REFERENCES sales_man(salesman_no);
ALTER TABLE sales_order ADD CHECK (del_type = 'F' OR del_type = 'P');
ALTER TABLE sales_order ADD check(order_status IN( 'Inprocess', 'Fullfilled', 'Cancelled', 'Backorder'));
\d sales_order;
```

```
Table "public.sales_order"
  Column      |      Type      | Collation | Nullable | Default
-----+-----+-----+-----+-----
order_no      | integer        |           | not null |
order_date    | date           |           | not null |
order_status  | character(10)  |           | not null |
salesman_no   | integer        |           |          |
del_type      | character(1)   |           |          |
Indexes:
    "sales_order_pkey" PRIMARY KEY, btree (order_no)
Check constraints:
    "sales_order_del_type_check" CHECK (del_type = 'F'::bpchar OR del_type = 'P'::bpchar)
    "sales_order_order_status_check" CHECK (order_status = ANY (ARRAY['Inprocess'::bpchar, 'Fullfilled'::bpchar, 'Cancelled'::bpchar, 'Backorder'::bpchar]))
Foreign-key constraints:
    "sales_order_salesman_no_fkey" FOREIGN KEY (salesman_no) REFERENCE
sales_man(salesman_no)
```

Program 3 : Hospital Table

AIM

Create a table Hospital with the fields (doctorid, doctorname, department, qualification, experience).

Write the queries to perform the following.

- Insert 5 records
- Display the details of Doctors
- Display the details of doctors who have the qualification 'MD'
- Display all doctors who have more than 5 years experience but do not have the qualification 'MD'
- Display the doctors in 'Skin' department
- update the experience of doctor with doctorid='D003' to 5
- Delete the doctor with DoctorID='D005'

TABLE DESIGN

Table name : hospital

Name	type	Description
doctorid	char(4)	doctor id
doctorname	varchar(10)	doctor name
department	varchar(10)	department
qualification	varchar(25)	qualification
experience	int	experience in years

QUERIES

```
create table hospital(doctorid char(4),doctorname varchar(10),department varchar(25),qualification
varchar(25),experience int);
\d hospital;
```

table "public.hospital"				
column	type	collation	nullable	default
doctorid	character(4)			
doctorname	character varying(10)			
department	character varying(25)			
qualification	character varying(25)			
experience	integer			

a)

```
insert into hospital values('d001','miya','cardiologist','mbbs',5);
insert into hospital values('d002','john','orthologist','md',4);
insert into hospital values('d003','ramesh','skin','mbbs',3);
insert into hospital values('d004','madona','dentist','bds',6);
insert into hospital values('d005','manoj','optometry','md',1);
```

b)

```
select * from hospital;
```

doctorid	doctorname	department	qualification	experience
d001	miya	cardiologist	mbbs	5
d002	john	orthologist	md	4
d003	ramesh	skin	mbbs	3
d004	madona	dentist	bds	6
d005	manoj	optometry	md	1

(5 rows)

c)

```
select doctorname from hospital where qualification='md';
```

doctorname
john
manoj

(2 rows)

d)

```
select doctorname from hospital where experience>5 and qualification!='md';
```

doctorname
madona

(1 row)

e)

```
select doctorname from hospital where department='skin';
```

```
doctorname
-----
ramesh
(1 row)
```

f)

```
update hospital set experience=5 where doctorid='d003';
select * from hospital;
```

doctorid	doctorname	department	qualification	experience
d001	miya	cardiologist	mbbs	5
d002	john	orthologist	md	4
d004	madona	dentist	bds	6
d005	manoj	optometry	md	1
d003	ramesh	skin	mbbs	5

(5 rows)

g)

```
delete from hospital where doctorid='d005';
select * from hospital;
```

doctorid	doctorname	department	qualification	experience
d001	miya	cardiologist	mbbs	5
d002	john	orthologist	md	4
d004	madona	dentist	bds	6
d003	ramesh	skin	mbbs	5

(4 rows)

Program 4 : implementing sql join and set operations

AIM

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

- Display the details of the customers
- Display the customers along with deposit amount who have only deposit with the bank
- Display the customers along with loan amount who have only loan with the bank
- Display the customers they have both loan and deposit with the bank
- Display the customer who have neither a loan nor a deposit with the bank

TABLE DESIGN

Tabel name : Bank_customer

Name	Type	Constraints	Description
accno	int	primary key	account number
cust_name	varchar(25)		customer name
place	varchar(25)		place

Table name : loan

Name	Type	Constraints	Description
accno	int	foreign key – bank_customer(accno)	account number
loan_no	int		loan number
lamount	numeric		loan amount

Table name : deposit

Name	Type	Constraints	Description
accno	int	foreign key – bank_customer(accno)	account number
deposit_no	int		deposit number
damount	numeric		deposit amount

QUERIES

create table Bank_customer(accno int primary key,cust_name varchar(25),place varchar(25));
create table Deposit(accno int references Bank_customer(accno),deposit_no int, damount numeric);
create table loan(accno int references bank_customer(accno), loan_no int,lamount numeric);

insert into bank_customer values(101,'Ravi','clt');
insert into bank_customer values(102,'Adam','tvm');
insert into bank_customer values(103,'Aysha','mlprm');
insert into bank_customer values(104,'Lisa','knr');
insert into bank_customer values(105,'Shaju','klm');
insert into bank_customer values(106, 'Razeen','kch');
insert into bank_customer values(107,'Radha','tvm');
insert into bank_customer values(108,'Jose','knr');

insert into deposit values(101,15,400000);
insert into deposit values(102,13,75000);
insert into deposit values(105,12,55000);
insert into deposit values(108,16,750000);

```

insert into loan values(103,4,500000);
insert into loan values(104,2,200000);
insert into loan values(106,6,300000);
insert into loan values(108,8,600000);

```

```
select * from loan;
```

accno	loan_no	lamount
103	4	500000
104	2	200000
106	6	300000
108	8	600000

(4 rows)

```
select * from deposit;
```

accno	deposit_no	damount
101	15	400000
102	13	75000
105	12	55000
108	16	750000

(4 rows)

a)

```
select * from bank_customer;
```

accno	cust_name	place
101	Ravi	clt
102	Adam	tvm
103	Aysha	mlprm
104	Lisa	knr
105	Shaju	klm
106	Razeen	kch
107	Radha	tvm
108	Jose	knr

(8 rows)

b)

```
select b.accno,cust_name,damount from bank_customer b join deposit d on b.accno=d.accno where
b.accno not in(select accno from loan);
```

accno	cust_name	damount
101	Ravi	400000

```

102 | Adam      | 75000
105 | Shaju     | 55000
(3 rows)

```

c)

select b.accno,cust_name,lamount from bank_customer b join loan l on b.accno=l.accno where b.accno not in(select accno from deposit);

```

accno | cust_name | lamount
-----+-----+-----
103 | Aysha     | 500000
104 | Lisa      | 200000
106 | Razeen    | 300000
(3 rows)

```

d)

select cust_name from bank_customer where accno in((select accno from loan)intersect(select accno from deposit));

```

cust_name
-----
Jose
(1 row)

```

e)

select cust_name from bank_customer where accno not in((select accno from loan)union(select accno from deposit));

```

cust_name
-----
Radha
(1 row)

```

Program 5 : Aggregate Functions

AIM

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

- Display the total number of employees.
- Display the name and age of the oldest employee of each department.
- Display the average age of employees of each department
- Display departments and the average salaries
- Display the lowest salary in employee table
- Display the number of employees working in purchase department
- Display the highest salary in sales department;
- Display the difference between highest and lowest salary

TABLE DESIGN

Table name : employee

Name	Type	Constraints	Description
empid	int	Primary key	Employee id
Ename	Varchar(10)		Employee name
salary	Numeric		Salary
department	Varchar(20)		Department name
age	int		Age

QUERIES

create table employee(empid int PRIMARY KEY,ename varchar(10),salary numeric, department varchar(20) , age int);

Insert into employee values(101, 'Adam', 20000, 'Purchase', 25), (102, 'Lisa', 15000, 'Sales', 45), (103, 'Arun', 18000, 'Sales', 34), (104, 'Aysha', 25000, 'Purchase', 25), (105, 'Sheeja', 30000, 'Finance', 36), (106, 'Sagar', 28000, 'Finance', 42);

select * from employee;

```
empid | ename | salary | department | age
-----+-----+-----+-----+-----
  101 | Adam  | 20000 | Purchase   | 25
  102 | Lisa  | 15000 | Sales      | 45
  103 | Arun  | 18000 | Sales      | 34
  104 | Aysha | 25000 | Purchase   | 25
  105 | Sheeja | 30000 | Finance    | 36
  106 | Sagar | 28000 | Finance    | 42
(6 rows)
```

a)

select count(empid)from employee;

```
count
-----
      6
(1 row)
```

b)

select ename,department from employee a where age in(select max(age) from employee b group by department having a.department=b.department);

```
ename | department
-----+-----
Adam  | Purchase
```

```

Lisa | Sales
Aysha | Purchase
Sagar | Finance
(4 rows)

```

c)

```
select department,avg(age)from employee group by department;
```

```

department |          avg
-----+-----
Purchase   | 25.000000000000000
Finance    | 39.000000000000000
Sales      | 39.500000000000000
(3 rows)

```

d)

```
select department,avg (salary)from employee group by department;
```

```

department |          avg
-----+-----
Purchase   | 22500.0000000000000
Finance    | 29000.0000000000000
Sales      | 16500.0000000000000
(3 rows)

```

e)

```
select min(salary) as min_salary from employee;
```

```

min_salary
-----
15000
(1 row)

```

f)

```
select count(ename) from employee where department='Purchase';
```

```

count
-----
2
(1 row)

```

g)

```
select max(salary)from employee where department='Sales';
```

```

max
-----
18000
(1 row)

```

h)

```
select max(salary) - min(salary) as sal_difference from employee;
```

```
sal_difference
-----
15000
(1 row)
```

Program 6 : Logical Operators

AIM

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'

TABLE DESIGN

Table name : product

Name	Type	Constraints	Description
product_code	int	primary key	product code number
product_name	varchar(20)		product name
category	varchar(20)		Category
quantity	int		Quantity
price	numeric		Price

QUERIES

create table product(product_code int primary key, product_name varchar(20),category varchar(20), quantity int,price numeric(10,2));

insert into product values(1,'colgate','paste',10,100);

insert into product values(2,'close up','paste',9,90);

insert into product values(3,'nirma','bath soap',10,600);

insert into product values(4,'sunlight','washing powder',10,700);

insert into product values(5,'toy','car',1,200);

insert into product values(6,'toy','bike',3,300);

insert into product values(7,'lux','bath soap',1,20);

```
insert into product values(8,'lux','bath liquid',600,2000);
insert into product values(9,'nirma','bath liquid',300,1000);
```

```
select * from product;
```

product_code	product_name	category	quantity	price
1	colgate	paste	10	100.00
2	close up	paste	9	90.00
3	nirma	bath soap	10	600.00
4	sunlight	washing powder	10	700.00
5	toy	car	1	200.00
6	toy	bike	3	300.00
7	lux	bath soap	1	20.00
8	lux	bath liquid	600	2000.00
9	nirma	bath liquid	300	1000.00

(9 rows)

a)

```
select * from product order by product_name desc;
```

product_code	product_name	category	quantity	price
5	toy	car	1	200.00
6	toy	bike	3	300.00
4	sunlight	washing powder	10	700.00
9	nirma	bath liquid	300	1000.00
3	nirma	bath soap	10	600.00
7	lux	bath soap	1	20.00
8	lux	bath liquid	600	2000.00
1	colgate	paste	10	100.00
2	close up	paste	9	90.00

(9 rows)

b)

```
select product_code,product_name from product where price between 20 and 50;
```

product_code	product_name
7	lux

(1 row)

c)

```
select product_name,price from product where category in ('bath soap','paste','washing powder');
```

product_name	price
colgate	100.00

```
close up | 90.00
nirma | 600.00
sunlight | 700.00
lux | 20.00
(5 rows)
```

d)

select * from product where quantity<100 or quantity>500;

```
product_code | product_name | category | quantity | price
-----+-----+-----+-----+-----
1 | colgate | paste | 10 | 100.00
2 | close up | paste | 9 | 90.00
3 | nirma | bath soap | 10 | 600.00
4 | sunlight | washing powder | 10 | 700.00
5 | toy | car | 1 | 200.00
6 | toy | bike | 3 | 300.00
7 | lux | bath soap | 1 | 20.00
8 | lux | bath liquid | 600 | 2000.00
(8 rows)
```

e)

select product_name from product where product_name like 's%';

```
product_name
-----
sunlight
(1 row)
```

f)

select product_name from product where category != 'paste';

```
product_name
-----
nirma
sunlight
toy
toy
lux
lux
nirma
(7 rows)
```

g)

select product_name from product where product_name like '_u%' and category='washing powder';

```
product_name
-----
sunlight
(1 row)
```

Program 7 : Employee Table

AIM

Consider the employee database given below. Give an expression in SQL for each of the following queries:

EMPLOYEE (Employee-Name, City)

WORKS (Employee-Name, Company-Name, Salary)

COMPANY (Company-Name, City)

MANAGES (Employee-Name, Manager-Name)

- Find the names of all employees who work in Infosys
- Find the names and cities of residence of all employees who works in Wipro
- Find the names, and cities of all employees who work in Infosys and earn more than Rs. 10,000.
- Find the employees who live in the same cities as the companies for which they work.
- Find all employees who do not work in Wipro Corporation.
- Find the company that has the most employees.

TABLE DESIGN

Table name : employee

Name	Type	Constraints	Description
empname	varchar(10)	primary key	employee name
city	varchar(10)		City

Table name : company

Name	Type	Constraints	Description
company_name	varchar(10)	primary key	company name
city	varchar(10)		City

Table name : works

Name	Type	Constraints	Description
empname	varchar(10)	foreign key – employee(empname)	employee name
cname	varchar(10)	foreign key – company(company_name)	company name
salary	int		Salary

Table name : manages



Name	Type	Constraints	Description
empname	varchar(10)	foreign key – employee(empname)	Employee name
manager_name	varchar(10)	foreign key - employee(empname)	City

QUERIES

create table employee(empname varchar(10) primary key,city varchar(10));

create table company(company_name varchar(10) primary key ,city varchar(10));

```
create table works(empname varchar(10) primary key references employee(empname),cname varchar(10) references company(company_name),salary int);
```

```
create table manages(empname varchar(10) references employee(empname),manager_name varchar(10) references employee(empname),primary key(empname,manager_name));
```

```
insert into employee values('swathi','kzkd');
insert into employee values('vishnu','tvm');
insert into employee values('shreya','usa');
insert into employee values('adam','dubai');
insert into employee values('sajid','malappuram');
```

```
insert into company values('infosys','tvm');
insert into company values('chandrika','trissur');
insert into company values('wipro','kochi');
insert into company values('tata','mumbai');
insert into company values('bajaj','delhi');
```

```
insert into works values('swathi','infosys',10000);
insert into works values('vishnu','infosys',15000);
insert into works values('shreya','wipro',71502);
insert into works values('adam','infosys',8000);
insert into works values('sajid','wipro',18000);
```

```
insert into manages values('swathi','adam');
insert into manages values('vishnu','adam');
insert into manages values('sajid','shreya');
```

```
select * from employee;
```

```
empname | city
-----+-----
swathi  | kzkd
vishnu  | tvn
shreya  | usa
adam    | dubai
sajid   | malappuram
(5 rows)
```

```
select * from company;
```

company_name	city
infosys	tvm
chandrika	trissur
wipro	kochi
tata	mumbai
bajaj	delhi

(5 rows)

select * from manages;

empname	manager_name
swathi	adam
vishnu	adam
sajid	shreya

(3 rows)

select * from works;

empname	cname	salary
swathi	infosys	10000
vishnu	infosys	15000
shreya	wipro	71502
adam	infosys	8000
sajid	wipro	18000

(5 rows)

a)

select empname from works where cname='infosys';

empname
swathi
vishnu
adam

(3 rows)

b)

select employee.empname,employee.city from employee,works where employee.empname = works.empname and works.cname = 'wipro';

empname	city
shreya	usa
sajid	malappuram

(2 rows)

c)

select employee.empname,city from employee,works where employee.empname= works.empname and cname='infosys' and salary>10000;

empname	city
---------	------

```
-----+-----  
vishnu | tvm  
(1 row)
```

d)

select employee.empname from employee,works,company where employee.empname = works.empname
and employee.city = company.city and works.cname = company.company_name;

```
empname  
-----  
vishnu  
(1 row)
```

e)

select empname from works where cname!='wipro';

```
empname  
-----  
swathi  
vishnu  
adam  
(3 rows)
```

f)

select cname from works group by cname order by count(*) desc limit 1;

```
cname  
-----  
infosys  
(1 row)
```

Program 8 :

AIM

Create table supplier(supcode,sname,city)

Create table product (pcode,pname)

Create table supl_product(supcode,pcode,qty)

- Get all pairs of supplier numbers such that the two suppliers are located in the same city.
- Get supplier names for suppliers who supply product P2.
- Get product numbers supplied by more than one supplier.
- Get supplier numbers for suppliers who are located in the same city as supplier S1.
- Get supplier names for suppliers who supply part P1.
- Get the number of Suppliers, who are supplying at least one product.
- For each product supplied, get the pcode. and the total quantity supplied for that part.

TABLE DESIGN

Table Name : Supplier

Name	Type	Constraints	Description
supcode	Char(3)	Primary key	Supplier number
sname	varchar(10)		Supplier name
city	Varchar(10)		City

Table Name : Product

Name	Type	Constraints	Description
pcode	Char(3)	Primary key	product number
pname	varchar(10)		product name

Table Name : Supl_product

Name	Type	Constraints	Description
supcode	Char(3)	Foreign key – supplier (supcode)	Supplier number
pcode	char(3)	Foreign key – product(pcode)	Product number
qty	int		Quantity

QUERIES

create table supplier(supcode char(3) primary key,sname varchar(10), city varchar(10));

create table product(pcode char(3) primary key, pname varchar(10));

Create table supl_product(supcode char(3) references supplier(supcode),pcode char(3) references product(pcode),qty int);

insert into supplier values ('s1','raju','calicut'), ('s2','sheela','thrissur'), ('s3','aysha','kochi'), ('s4','anees','tirur'), ('s5','lisa','calicut'),('s6','zara','tirur');

select * from supplier;

supcode	sname	city
s1	raju	calicut
s2	sheela	thrissur
s3	aysha	kochi
s4	anees	tirur
s5	lisa	calicut
s6	zara	tirur

(6 rows)

```
insert into product values('p1','soap'), ('p2','rice'), ('p3','salt'), ('p4','sugar'), ('p5','wheat'), ('p6','colgate'), ('p7','chilly');
```

```
select * from product;
```

pcode	pname
p1	soap
p2	rice
p3	salt
p4	sugar
p5	wheat
p6	colgate
p7	chilly

(7 rows)

```
insert into supl_product values('s1','p1',34), ('s1','p2',20), ('s2','p2',20), ('s6','p4',5), ('s3','p3',10), ('s2','p7',10), ('s3','p1',12);
```

```
select * from supl_product;
```

supcode	pcode	qty
s1	p1	34
s1	p2	20
s2	p2	20
s6	p4	5
s3	p3	10
s2	p7	10
s3	p1	12

(7 rows)

a)

```
select a.supcode,b.supcode,b.city from supplier a,supplier b where a.city=b.city and a.supcode<b.supcode;
```

supcode	supcode	city
s1	s5	calicut
s4	s6	tirur

(2 rows)

b)

```
select sname from supplier where supcode in(select supcode from supl_product where pcode='p2');
```

sname
raju

```
sheela
(2 rows)
```

c)

select pcode from supl_product group by pcode having count(pcode)>1;

```
pcode
-----
p2
p1
(2 rows)
```

d)

select supcode from supplier where city=(select city from supplier where supcode='s1');

```
supcode
-----
s1
s5
(2 rows)
```

e)

select sname from supplier where supcode in(select supcode from supl_product where pcode='p1');

```
sname
-----
raju
aysha
(1 row)
```

f)

select count(distinct supcode) from supl_product;

```
count
-----
4
(1 row)
```

g)

select pcode,sum(qty) from supl_product group by pcode;

```
pcode | sum
-----+-----
p3    | 10
p4    | 5
p7    | 10
p2    | 40
p1    | 46
(5 rows)
```

Part 2 PostgreSQL

Program 9 : Salary Report of Employees

AIM

Prepare a salary report of the employees showing the details such as:

EmpNo, Name, Basic Pay, DA, Gross Salary, PF, Net Salary, Annual Salary and Tax

For this purpose, create a table named SALARIES having the following structure.

Field Name	Type	Width
EmpNo	Character	10
Name	Character	20
Basic	Numeric	6

Enter the records of at least 10 employees. Use the following information for calculating the details for the report:

- DA is fixed as the 40% of the basic pay.
- PF is fixed as 10% of the basic pay.
- Gross Salary is (Basic Pay + DA).
- Net Salary is (Gross Salary – PF)
- Annual Salary is (12 * Net Salary)

Tax is calculated using the following rules:

- If annual salary is less than 100000, No Tax
- If annual salary is greater than 100000 but less than or equal to 150000, then the tax is 10% of the excess over 100000.
- If annual salary is greater than 150000 but less than or equal to 250000, then the tax is 20% of the excess over 150000.
- If annual salary is greater than 250000, then the tax is 30% of the excess over 250000.

TABLE DESIGN

Table Name : salaries

Name	Type	Description
EmpNo	char(10)	Employee number
Name	char(20)	Name
Basic	numeric(6)	Basic pay

QUERIES

create table salaries(EmpNo char(3), Name char(10), Basic numeric(6));

Insert into salaries values('101', 'Adam', 20000), ('102', 'Lisa', 15000), ('103', 'Arun', 18000), ('104', 'Aysha', 25000), ('105', 'Sheeja', 30000), ('106', 'Sagar', 28000);

```
Insert into salaries values('111', 'Muthu', 10000), ('109', 'Hari', 2500), ('108', 'Raju', 5000), ('107', 'Sabi', 8000);
select * from salaries;
```

empno	name	basic
101	Adam	20000
102	Lisa	15000
103	Arun	18000
104	Aysha	25000
105	Sheeja	30000
106	Sagar	28000
111	Muthu	10000
109	Hari	2500
108	Raju	5000
107	Sabi	8000

(10 rows)

```
do $$
declare
dav numeric(10,2);
pfv numeric(10,2);
grossv numeric(10,2);
anv numeric(10,2);
netv numeric(10,2);
taxv numeric(10,2);
rec record;
begin
alter table salaries add column da numeric(10,2),add column pf numeric(10,2),add column gross
numeric(10,2),add column net numeric(10,2),add column annual numeric(10,2),add column tax
numeric(10,2);
for rec in select * from salaries
loop
dav:=rec.basic*0.4;
pfv:=rec.basic*0.1;
grossv:=rec.basic+dav;
netv:=grossv-pfv;
anv:=12*netv;
if anv>250000 then
taxv:=(anv-250000)*0.3+ (250000-150000)*0.2 + (150000-100000)*0.1;
elsif anv>150000 then
taxv:=(anv-150000)*0.2 + (150000-100000)*0.1;
elsif anv>100000 then
taxv:=(anv-100000)*0.1;
```

```

else
taxv:=0;
end if;
update salaries set da=dav, pf=pfv, gross=grossv, net=netv, annual=anv, tax=taxv where
empno=rec.empno;
end loop;
end $$;
select * from salaries;

```

empno	name	basic	da	pf	gross	net	annual	tax
101	Adam	20000	8000.00	2000.00	28000.00	26000.00	312000.00	43600.00
102	Lisa	15000	6000.00	1500.00	21000.00	19500.00	234000.00	21800.00
103	Arun	18000	7200.00	1800.00	25200.00	23400.00	280800.00	34240.00
104	Aysha	25000	10000.00	2500.00	35000.00	32500.00	390000.00	67000.00
105	Sheeja	30000	12000.00	3000.00	42000.00	39000.00	468000.00	90400.00
106	Sagar	28000	11200.00	2800.00	39200.00	36400.00	436800.00	81040.00
111	Muthu	10000	4000.00	1000.00	14000.00	13000.00	156000.00	6200.00
109	Hari	2500	1000.00	250.00	3500.00	3250.00	39000.00	0.00
108	Raju	5000	2000.00	500.00	7000.00	6500.00	78000.00	0.00
107	Sabi	8000	3200.00	800.00	11200.00	10400.00	124800.00	2480.00

Program 10 : Calculating Grade From Average Score

AIM

Create table exam_result(rollno, avg_score, Grade) insert 10 records. Assign null values to the field grade. Write Program block to update the grade field by using the following condition.

avg_score between 90 and 100 - A

avg_score 75 -89 - B

avg_score 60- 74 - C

avg_score 50 -59 - D

avg_score below 50 – E

TABLE DESIGN

Table Name : exam_result

Name	Type	Description
Rollno	Integer	Roll Number
avg_score	numeric(5,2)	Average Mark
Grade	char(1)	Grade

QUERIES

Create table exam_result(rollno integer,avg_score numeric(5,2), grade char(1));

```
insert into exam_result values (4, 67), (3, 35), (2, 91), (6, 45), (11, 86),(7,95), (8, 75), (9, 68), (5, 55), (12, 97), (13, 30);
```

```
select * from exam_result;
```

rollno	avg_score	grade
4	67.00	
3	35.00	
2	91.00	
6	45.00	
11	86.00	
7	95.00	
8	75.00	
9	68.00	
5	55.00	
12	97.00	
13	30.00	

(11 rows)

```
do $$
declare
rec record;
grd char(1);
begin
for rec in select * from exam_result
loop
if rec.avg_score between 90 and 100 then
grd:= 'A';
elsif rec.avg_score between 75 and 89 then
grd= 'B';
elsif rec.avg_score between 60 and 74 then
grd:= 'C';
elsif rec.avg_score between 50 and 59 then
grd:= 'D';
else
grd:= 'E';
end if;
update exam_result set grade=grd where rollno=rec.rollno;
end loop;
end;
```

```
$$ language plpgsql;
```

```
Select * from exam_result;
```

```
rollno | avg_score | grade
-----+-----+-----
      4 |      67.00 | C
      3 |      35.00 | E
      2 |      91.00 | A
      6 |      45.00 | E
     11 |      86.00 | B
      7 |      95.00 | A
      8 |      75.00 | B
      9 |      68.00 | C
      5 |      55.00 | D
     12 |      97.00 | A
     13 |      30.00 | E
(11 rows)
```

Program 11 : Area of a Circle

AIM

Write a program code to calculate the area of a circle for a value of radius varying from 3 to 7. Store the radius and the corresponding value of calculated area in an empty table named areas with field's radius and area.

TABLE DESIGN

Table Name : areas

Name	Type	Description
radius	integer	Radius of circle
area	numeric	Calculated area

QUERIES

```
do $$
declare
r integer;
ar numeric(10,2);
begin
create table areas(radius integer,area numeric(5,2));
r:=3;
for r in 3..7 loop
```

```

ar:=3.14*r*r;
insert into areas values(r,ar);
end loop;
end $$;

```

```

select * from areas;

```

```

radius | area
-----+-----
      3 |  28.26
      4 |  50.24
      5 |  78.50
      6 | 113.04
      7 | 153.86
(5 rows)

```

Program 12 : Electricity Bill Calculation

AIM

Write a program block to calculate the electricity bill by accepting cust_no and units_consumed.

ALGORITHM

Function electricity_bill(custno as int, unit as int)

- Step 1. Start
- Step 2. Check if unit<=100 then rate=3 and go to step 5
- Step 3. Check if unit<=250 then rate=4 and go to step 5
- Step 4. Check if unit<=500 then rate=5, else rate=6
- Step 5. Calculate bill_amount=rate*unit.
- Step 6. Print bill_amount
- Step 7. Stop

PROGRAM CODE

```

create or replace function electricity_bill(c int, u int) returns text as $$
declare
rate int;
amt int;
begin
if u<=100 then
rate:=3;
elsif u<=250 then
rate:=4;
elsif units_consumed<=500 then

```

```

rate:=5;
else
rate:=6;
end if;
amt:=rate*u;
return 'Customer No      : ' || c || E'\nUnits Consumed : ' || u || E'\nBill Amount      : ' || amt;
end;
$$ language plpgsql;

```

```
select electricity_bill(121,200);
```

```

electricity_bill
-----
Customer No      : 121+
Units Consumed   : 200+
Bill Amount      : 800
(1 row)

```

Program 13 : Fibonacci Numbers upto a Limit

AIM

Create a procedure to print Fibonacci number up to a limit, limit is passed as an argument

ALGORITHM

The function fibonacci(n as int)

Step 1. Start

Step 2. Declare a, b and c as integer.

Step 3. Initialize a=1, b=0 and c=0

Step 4. Repeat steps 5 through 8 until n<c

Step 5. Print c

Step 6. Calculate c=a+b

Step 7. a=b

Step 8. b=c

Step 9. Stop

PROGRAM CODE

```
create or replace function fibonacci(n int) returns setof int as $$
```

```

declare
a int:=1;
b int:=0;
c int:=0;
begin
loop
exit when n < c;
return next c;
c:=a+b;
a:=b;
b:=c;
end loop;
end;
$$ language plpgsql;

```

```
select fibonacci(8);
```

fibonacci
0
1
1
2
3
5
8
(7 rows)

Program 14 : Check Prime or Not

AIM

Create a function to check whether a given number is prime or not

ALGORITHM

Function checkprime(n as int)

Step 1. Start

Step 2. Declare i as int

Step 3. Check if $n < 2$ then, print “not prime” and go to step 9

Step 4. $i = 2$

Step 5. Repeat steps 6 & 7 while $i \leq n/2$

Step 6. If $n \bmod i = 0$ then print "not prime" and go to step 9

Step 7. $i = i + 1$

Step 8. Print "Prime". Go to step 9

Step 9. Stop

PROGRAM CODE

```
CREATE FUNCTION check_prime(n int) returns varchar(25) AS $$
```

```
DECLARE
```

```
i int;
```

```
BEGIN
```

```
if n < 2 then
```

```
return n || ' is not a prime number ';
```

```
end if;
```

```
for i in 2..n/2
```

```
loop
```

```
if mod(n,i)=0 then
```

```
return n || ' is not a prime number ';
```

```
end if;
```

```
end loop;
```

```
return n || ' is a prime number ';
```

```
end;
```

```
$$ language plpgsql;
```

```
Select check_prime(1);
```

```
Select check_prime(7);
```

```
1bsmdc=# select check_prime(1);
        check_prime
```

```
-----
1  is not a prime number
(1 row)
```

```
1bsmdc=# select check_prime(7);
        check_prime
```

```
-----
7  is a prime number
(1 row)
```

Program 15 : Student Mark List

AIM

Create a table stud_mark(regno, sname ,avg_mark)

Insert few records

Write a procedure to display number of students got Distinction, first-class, second class, third class or failed (90-100 distinction, 75-89 firstclass 60-74 second class 50-59 Third class below 50 failed)

TABLE DESIGN

Table Name : stud_mark

Name	Type	Description
Regno	char(5)	Register number
Sname	varchar(20)	Name of the student
avg_mark	numeric(5,2)	Average mark

QUERIES

Create table stud_mark(regno char(5), sname varchar(20), avg_mark numeric(5,2));

insert into stud_mark values ('S01', 'Viji', 35), ('S02', 'Adam', 91), ('S03', 'Zara', 45), ('S04', 'Lisa', 86), ('S05', 'Ishan', 95), ('S06', 'Hari', 75), ('S07', 'Haya', 68), ('S08', 'Jisha', 55), ('S09', 'Jasi', 97), ('S10', 'Pranav', 30), ('S11', 'Neeraj', 67);

select * from stud_mark;

```
regno | sname | avg_mark
-----+-----+-----
s01   | viji  | 35.00
s02   | Adam  | 91.00
s03   | Zara  | 45.00
s04   | Lisa  | 86.00
s05   | Ishan | 95.00
s06   | Hari  | 75.00
s07   | Haya  | 68.00
s08   | Jisha | 55.00
s09   | Jasi  | 97.00
s10   | Pranav | 30.00
s11   | Neeraj | 67.00
(11 rows)
```

do \$\$

declare

dist int;

first int;

```

second int;
third int;
fail int;
begin
select count(*) into dist from stud_mark where avg_mark between 90 and 100;
select count(*) into first from stud_mark where avg_mark between 75 and 89;
select count(*) into second from stud_mark where avg_mark between 60 and 74;
select count(*) into third from stud_mark where avg_mark between 50 and 59;
select count(*) into fail from stud_mark where avg_mark < 50;
raise notice '
No of Distintions      : %
No of First Classes    : %
No of Second Classes   : %
No of Third Classes    : %
No of Failures         : %',dist,first,second,third,fail;
end $$;

```

NOTICE:

```

No of Distintions      : 3
No of First Classes    : 2
No of Second Classes   : 2
No of Third Classes    : 1
No of Failures         : 3
DO

```

Program 16 Display Average Salary of a Department

AIM

Create a table emp_salary(empno,ename,dept,salary)

Write a function to return the average salary of a particular department by accepting departmentname as argument.

TABLE DESIGN

Table Name : emp_salary

Name	Type	Description
empno	int	Employee id
ename	Varchar(15)	Employee Name
dept	Varchar(15)	department
salary	int	salary

QUERIES

Create table emp_salary(empno int, ename varchar(15), dept varchar(15), salary int);

Insert into emp_salary values(101, 'Adam', 'Production', 20000), (102, 'Lisa', 'Marketing', 15000), (103, 'Arun', 'Marketing', 18000), (104, 'Aysha', 'Production', 25000), (105, 'Sheeja', 'Finance', 30000), (106, 'Sagar', 'Finance', 28000);

Select * from emp_salary;

```
empno | ename | dept | salary
-----+-----+-----+-----
  101 | Adam | Production | 20000
  102 | Lisa | Marketing | 15000
  103 | Arun | Marketing | 18000
  104 | Aysha | Production | 25000
  105 | Sheeja | Finance | 30000
  106 | Sagar | Finance | 28000
(6 rows)
```

create function avg_salary(dept_name varchar(10)) returns numeric(10,2) as \$\$

declare

avg_sal numeric(10,2);

begin

select avg(salary) into avg_sal from emp_salary group by dept having dept=dept_name;

return avg_sal;

end;

\$\$ language plpgsql;

Select avg_salary('Production');

avg_salary

```
-----  
22500.00  
(1 row)
```

Select avg_salary('Finance');

```
avg_salary  
-----  
29000.00  
(1 row)
```

Progam 17. Implementation of Trigger Before Insert

AIM

Create a table Student (regno, sname, sub1, sub2, sub3, sub4, sub5, mark_total, avg_mark)

Create a BEFORE INSERT trigger to calculate total mark and average mark and update the corresponding columns.

TABLE DESIGN

Table Name : student

Name	Type	Description
Regno	char(5)	Register number
Sname	varchar(15)	Name of the student
sub1	numeric(3)	Subject 1
sub2	numeric(3)	Subject 2
sub3	numeric(3)	Subject 3
sub4	numeric(3)	Subject 4
sub5	numeric(3)	Subject 5
mark_total	numeric(3)	Total mark
avg_mark	numeric(5,2)	Average mark

QUERIES

Create table student(regno char(5), sname varchar(15), sub1 numeric(3), sub2 numeric(3), sub3 numeric(3), sub4 numeric(3), sub5 numeric(3), mark_total numeric(3), avg_mark numeric(5,2));

create or replace function fun() returns trigger as \$\$

declare

begin

```

new.mark_total=new.sub1+new.sub2+new.sub3+new.sub4+new.sub5;
new.avg_mark=new.mark_total/5.0;
return new;
end;
$$ language plpgsql;

create trigger trig before insert on student for each row execute procedure fun();

insert into student values ('s101','adam',23,45,67,23,45), ('s102',' sheena', 96,97,89,95,67), ('s103',
'bobby', 67,52,83,91,34), ('s104','radha', 34,54,23,12,25), ('s105','zara', 86,76,82,85,34);

select * from student;

```

regno	sname	sub1	sub2	sub3	sub4	sub5	mark_total	avg_mark
s101	adam	23	45	67	23	45	203	40.60
s102	sheena	96	97	89	95	67	444	88.80
s103	bobby	67	52	83	91	34	327	65.40
s104	radha	34	54	23	12	25	148	29.60
s105	zara	86	76	82	85	34	363	72.60

(5 rows)

Program 18. Implementation of Trigger After Delete or Update

AIM

Create table phonebook (pname, mobno)

Create a Trigger to insert the old records from the table phonebook to del_phonebook (pname, mobno, modfy_date) whenever a record is deleted or updated in the phonebook table.

TABLE DESIGN

Table Name : phonebook

Name	Type	description
Pname	varchar(20)	Name of the Person
Mobno	char(10)	Mobile Number

Table Name : del_phonebook

Name	Type	description
Pname	varchar(20)	Name of the Person

Mobno	char(10)	Mobile Number
modify_date	timestamp	modified date time

QUERIES

```
create table phonebook(pname varchar(20), mobno char(10));
create table del_phonebook(pname varchar(20), mobno char(10), modify_date timestamp);
```

```
insert into phonebook values('Raju','9895324212');
insert into phonebook values('Aravind', '9435111222');
insert into phonebook values('Shreeja','9277123435');
insert into phonebook values('Arunthadi', '9867111333');
insert into phonebook values('Fida', '9234555777');
```

```
select * from phonebook;
```

pname	mobno
Raju	9895324212
Aravind	9435111222
Shreeja	9277123435
Arunthadi	9867111333
Fida	9234555777

(5 rows)

```
create or replace function fun() returns trigger as $$
begin
    insert into del_phonebook values (old.pname, old.mobno, now());
return new;
end;
$$ language plpgsql;
```

```
create trigger trig after delete or update on phonebook for each row execute procedure fun();
```

```
delete from phonebook where pname='Shreeja';
update phonebook set mobno='9292777888' where pname='Raju';
```

```
select * from del_phonebook;
```

pname	mobno	modify_date
Shreeja	9277123435	2024-05-28 13:28:21.409086
Raju	9895324212	2024-05-28 13:28:21.454148

(2 rows)