1. Write a Query to add a column package_stat to the table orders.

```
dbda_lab=# alter table orders add column package_stat text;
ALTER TABLE
dbda_lab=# \d orders;
                 Table "public.orders"
   Column    |       Type        | Collation | Nullable | Default
-------------+-------------------+-----------+----------+---------
 order_id    | integer           |           | not null |
package_stat | text              |           |          |
```

2. Write a Query to change the package_stat column of orders table with 'not available' for all orders.

```
dbda_lab=# update orders set package_stat = 'Not Available';
UPDATE 6
dbda_lab=# table orders;
 order_id | order_date | shipped_date | deliver | customer_id | package_stat
----------+------------+--------------+---------+-------------+---------------
   10100 | 2003-01-06 | 2003-01-13   | Shipped |       114 | Not Available
   10101 | 2003-01-09 | 2003-01-18   | Sh     +|       125 | Not Available
         |            |              | ipped   |           |
   10102 | 2003-01-10 | 2003-01-18   | Shipped |       129 | Not Available
   10103 | 2003-01-29 | 2003-02-07   | Sh     +|       121 | Not Available
```

3. Write a Query to delete a row from customers table where credit_limit is 0.00

```
delete from customers where creditlimit=0.00;
```

*****

1. Write a Query to display the first_name with the occurrence of ‘el’ in the customers tables.

```
dbda_lab=# select first_name from customers where first_name like '%el%';
 first_name
------------
 Atelier
 Nelson
 Keitel
 Saveley
```

2. Write a Query to prepare a list with customer name ,customer_id ,order_id for the customers whose delivery status is shipped.

```
bda_lab=# select first_name,c.customer_id,o.order_id from customers c join
orders o on c.customer_id = o.customer_id where deliver = 'Shipped';
 first_name | customer_id | order_id
------------+-------------+----------
 Ferguson   |         114 |    10100
 Murphy     |         129 |    10102
 Freyre     |         141 |    10104
(3 rows)
```

3. Write a Query to get the number of customers with the creditLimit greater than 50000.

```
dbda_lab=# select count(*) from customers where creditlimit > 50000;
 count
```

```
-------
    14
(1 row)
```

4. Write a Query to display the customer_id, name (first name and last name), order_id and deliver for all customers.

dbda_lab=# **select c.customer_id, c.first_name||' '||c.last_name as name, o.order_id, o.deliver from customers c join orders o on c.customer_id = o.customer_id;**

```
 customer_id |          name           | order_id | deliver
-------------+-------------------------+----------+---------
         114 | Ferguson Peter          |    10100 | Shipped
         125 | Piestrzeniewicz Zbyszek |    10101 | Sh     +
             |                         |          | ipped
         129 | Murphy Julie            |    10102 | Shipped
```

5. Write a Query to customer name in order of creditLimit smallest to highest.

dbda_lab=# **select first_name ||' '|| last_name as name,creditlimit from customers order by name;**

```
          name           | creditlimit
-------------------------+-------------
 Atelier Schmitt         |    21000.00
 Berglund Christina     +|    53100.00
                         |
 Bergulfsen Jonas        |    81700.00
```

6. Write a stored procedure by name order_day. The procedure should show the customer_id and the day on which he had made the order.

```
dbda_lab=# create table day(customer_id int, order_date date);

CREATE TABLE

dbda_lab=# create or replace procedure order_day(q integer)

language plpgsql

as $$

begin

 insert into day(customer_id,order_date) select customer_id , order_date
from orders where customer_id=q;

end; $$;

CREATE PROCEDURE


dbda_lab=# call order_day(129);

CALL

dbda_lab=# table day;

 customer_id | order_date

-------------+------------

         129 | 2003-01-10

(1 row)
```

7. Write a stored function by the name of cutomer_search. The stored function should return the maximum creditLimit made by any customer.

```
dbda_lab=# create or replace function customer_search()

dbda_lab-# returns integer

dbda_lab-# as $$

dbda_lab$# declare maxlim integer;

dbda_lab$# begin
```

```
dbda_lab$# select max(creditlimit) into maxlim from customers ;

dbda_lab$# return maxlim; end; $$

dbda_lab-# language plpgsql;

CREATE FUNCTION


dbda_lab=# select customer_search();

 customer_search

-----------------

        227600

(1 row)
```

 *****

Display only the EMPNO and ENAME columns from EMP table.

>>**select empno , ename from emp;**

Display all employees who are CLERKs and the MANAGERs
>>**select * from emp where job in ('CLERK','MANAGER');**

Display the ENAME and JOB for all employees who belong to the same DEPTNO as employee ‘KING’
>>**select ename ,job from emp where deptno in (select deptno from emp where ename= 'KING');**

Find the names of all employees hired in the month of February (of any year).
>>**SELECT ENAME FROM EMP WHERE TO_CHAR (HIREDATE,'MONTH') LIKE '%FEB%';**

Display the employees in descending order of DEPTNO
>>**SELECT * FROM EMP ORDER BY DEPTNO DESC**

Display the employee name and employee number of the employees with the headings as NUMBER and NAME
>>**SELECT ENAME AS NAME ,EMPNO AS NUMBER FROM EMP;**

Find the name of the employee who is receiving the maximum salary.
>>**SELECT ENAME FROM EMP WHERE SAL IN (SELECT MAX(SAL) FROM EMP);**

Display the sum of SAL for all the employees belonging to DEPTNO 10. ;
>>**SELECT SUM(SAL) FROM EMP WHERE DEPTNO=10;**

Display the rows where JOB column ends with the letter ‘T’

```
>>SELECT * FROM EMP WHERE JOB LIKE '%T';
```

Write a stored procedure to convert a temperature in Fahrenheit (F) to its equivalent in Celsius (C). The required formula is:- C= (F-32)*5/9 Insert the temperature in Centigrade into TEMPP table. Calling program for the stored procedure need not be written.

```
>>CREATE OR REPLACE PROCEDURE TEMP(F NUMERIC)
LANGUAGE PLPGSQL
AS $$
DECLARE C NUMERIC;
BEGIN
DROP TABLE IF EXISTS TEMPP; CREATE TABLE TEMPP (TEMP NUMERIC);
C:=(F-32)*5/9;
INSERT INTO TEMPP VALUES (C);
END; $$;
```

12. Write a stored function by the name of Num_cube. The stored function should return the cube of a number ‘N’. The number ‘N’ should be passed to the stored function as a parameter. Calling program for the stored function need not be written

```
>>CREATE OR REPLACE FUNCTION NUM_CODE (N INT)
RETURNS INT
LANGUAGE PLPGSQL
AS $$
BEGIN
RETURN N*N*N;
END; $$;
```