

|  |  |
| --- | --- |
| **Sr.No** | 01 |
| **SQL**  **Query :** | Give all the information about salesman, customer and order table. |
| **Solution :** | Select \*from salesman;  Select \*from customer;  Select \*from orderr; |
| **Output :** |  |
|  |  |
| **Sr.No** | 02 |
| **SQL**  **Query :** | Add Unique key constraint on name field of Salesman table. |
| **Solution :** | alter table salesman add constraint unique\_sname unique(sname); |
| **Output :** |  |

|  |  |
| --- | --- |
|  |  |
| **Sr.No** | 03 |
| **SQL**  **Query :** | Produce cnum and snum of customer who lives in Baroda. |
| **Solution :** | Select cnum, snum from customer where city = ‘baroda’; |
| **Output :** |  |
|  |  |
| **Sr.No** | 04 |
| **SQL**  **Query :** | Produce the order no, amount and date of all orders. |
| **Solution :** | Select onum, amount, odate from orderr; |
| **Output :** |  |
|  |  |
| **Sr.No** | 05 |
| **SQL**  **Query :** | Give all the information about all the customers with salesman number 1001. |
| **Solution :** | Select \*from customer where snum = 1001; |
| **Output :** |  |
|  |  |
| **Sr.No** | 06 |

|  |  |
| --- | --- |
| **SQL**  **Query :** | Display the information in the sequence of city, sname, snum, and Commission. |
| **Solution :** | Select city, sname, snum, commission from salesman; |

|  |  |
| --- | --- |
| **Output :** |  |
|  |  |
| **Sr.No** | 07 |
| **SQL**  **Query :** | List of rating followed by the name of each customer in Surat. |
| **Solution :** | Select cname, ratig from customer where city = ‘surat’; |
| **Output :** |  |
|  |  |
| **Sr.No** | 08 |
| **SQL**  **Query :** | List of snum of all salesmen with orders in order table without any duplicates. |
| **Solution :** | Select distinct(snum) from orderr; |
| **Output :** |  |
|  |  |
| **Sr.No** | 09 |
| **SQL**  **Query :** | List of all orders for more than Rs. 1000. |

|  |  |
| --- | --- |
| **Solution :** | Select \*from orderr where amount > 1000; |
| **Output :** |  |
|  |  |
| **Sr.No** | 10 |
| **SQL**  **Query :** | List out names and cities of all salesmen in Mumbai with commission above 10%. |
| **Solution :** | select sname, city from salesman where city ='mumbai' and commission>10; |
| **Output :** |  |
|  |  |
| **Sr.No** | 11 |
| **SQL**  **Query :** | List all customers excluding those with rating <= 100 or they are located in Rajkot. |

|  |  |
| --- | --- |
| **Solution :** | select \*from customer where rating > 100 and city != 'rajkot'; |
| **Output :** |  |
|  |  |
| **Sr.No** | 12 |
| **SQL**  **Query :** | List all order for more than Rs. 1000 except the orders of sum, 1006 of 10/03/2023. |
| **Solution :** | Select \*from orderr where amount > 1000 and not(snum =1006 and odate = ’10-mar-2023’); |

|  |  |
| --- | --- |
| **Output :** |  |
|  |  |
| **Sr.No** | 13 |
| **SQL**  **Query :** | List all orders taken on October 01st or 11th or 29th 2023. |
| **Solution :** | Select \*From orderr where odate in ( ’01-oct-2023’,’11-oct-2023’,’29-oct-2023’); |
| **Output :** |  |
|  |  |
| **Sr.No** | 14 |
| **SQL**  **Query :** | List all customers whose names begin with a letter 'R'. |
| **Solution :** | Select \*from customer where cname like ‘r%’; |

|  |  |
| --- | --- |
| **Output :** |  |
|  |  |
| **Sr.No** | 15 |
| **SQL**  **Query :** | List all customers whose names begins with letter 'A' to 'G'. |
| **Solution :** | select \*from customer where cname between 'a' and 'g'; |
| **Output :** |  |
|  |  |

|  |  |
| --- | --- |
| **Sr.No** | 16 |
| **SQL**  **Query :** | List all orders with zero or NULL amount. |
| **Solution :** | Select \*from order where amount = 0 or amount is null; |
| **Output :** |  |
|  |  |
| **Sr.No** | 17 |
| **SQL**  **Query :** | Find out the largest orders of salesman 1002 and 1007. |
| **Solution :** | Select snum,max(amount) max from orderr group by snum having snum=1002 or snum = 1007; |
| **Output :** |  |
|  |  |
| **Sr.No** | 18 |

|  |  |
| --- | --- |
| **SQL**  **Query :** | Count all orders of 01-Mar-2023. |
| **Solution :** | select count(\*) From orderr where odate = '1-mar-2023'; |
| **Output :** |  |
|  |  |
| **Sr.No** | 19 |
| **SQL**  **Query :** | Calculate the total amount ordered. |
| **Solution :** | Select sum(amount) as “Total amount of orders” from orderr; |
| **Output :** |  |
|  |  |
| **Sr.No** | 20 |

|  |  |
| --- | --- |
| **SQL**  **Query :** | Calculate the average amount ordered. |
| **Solution :** | Select avg(amount) from orderr; |
| **Output :** |  |
|  |  |
| **Sr.No** | 21 |
| **SQL**  **Query :** | Count the no. of salesmen currently having orders. |
| **Solution :** | Select count(distinct snum) from orderr; |
| **Output :** |  |
|  |  |
| **Sr.No** | 22 |

|  |  |
| --- | --- |
| **SQL**  **Query :** | Find the largest order taken by each salesman on each date. |
| **Solution :** | Select snum, odate , max(amount) from orderr group by snum, odate; |
| **Output :** |  |
|  |  |
| **Sr.No** | 23 |
| **SQL**  **Query :** | Find the largest order taken by each salesman on 10/02/2023. |
| **Solution :** | Select odate, snum, max(amount) from orderr where odate = ’10-feb-2023’ group by snum,odate; |
| **Output :** |  |
|  |  |

|  |  |
| --- | --- |
| **Sr.No** | 24 |
| **SQL**  **Query :** | Count the no. of different not-null cities in the Customer table. |
| **Solution :** | Select count(distinct city) as “No of cities” from customer where city is not null; |
| **Output :** |  |
|  |  |
| **Sr.No** | 25 |
| **SQL**  **Query :** | Find out each customer's smallest order. |
| **Solution :** | Select cnum, min(amount) as “Smallest order” from orderr group by cnum; |

# ACHARYA MOTIBHAI PATEL INSTITUTE OF COMPUTER STUDIES – MCA DEPARTMENT MCA SEM -1 P11A3DMS DATABASE MANAGEMENT SYSTEM

|  |  |
| --- | --- |
| **Output :** |  |
|  |  |
| **Sr.No** | 26 |
| **SQL**  **Query :** | Find out the customer in alphabetical order whose name begins with 'a'. |
| **Solution :** | Select \*from customer where cnaame like ‘a%’ order by cname; |
| **Output :** |  |
|  |  |
| **Sr.No** | 27 |
| **SQL**  **Query :** | Count the no. of salesmen registering orders for each day. |
| **Solution :** | Select odate, count(distinct snum) from orderr group by odate; |

| P a g e

|  |  |
| --- | --- |
| **Output :** |  |

## 

|  |  |
| --- | --- |
|  |  |
| **Sr.No** | 28 |
| **SQL**  **Query :** | List all salesmen with their % of commission. |

### 

|  |  |
| --- | --- |
| **Solution :** | Select snum,sname,city,commisiion||’%’ as commission from salesman; |
| **Output :** |  |
|  |  |
| **Sr.No** | 29 |
| **SQL**  **Query :** | Find out the salesman whose name ends with a. |
| **Solution :** | Select \*From salesman where sname like ‘a%’; |

|  |  |
| --- | --- |
| **Output :** |  |
|  |  |
| **Sr.No** | 30 |
| **SQL**  **Query :** | Assume each salesperson has a 12% commission. Write a query on the order table that will produce the order number, salesman no and amount of commission for that order. |
| **Solution :** | Select onum, snum, amount\*12/100 commission from orderr; |
| **Output :** |  |
|  |  |
| **Sr.No** | 31 |
| **SQL**  **Query :** | Find the highest rating in each city in proper format. |
| **Solution :** | select city, max(rating) from customer group by city; |
| **Output :** |  |
|  |  |
| **Sr.No** | 32 |
| **SQL**  **Query :** | List all customers in descending order of rating. |
| **Solution :** | select \*from customer order by rating desc; |

|  |  |
| --- | --- |
| **Output :** |  |
|  |  |
| **Sr.No** | 33 |
| **SQL**  **Query :** | Calculate the total of orders for each day. |
| **Solution :** | Select odate, sum(amount) from orderr group by odate; |

|  |  |
| --- | --- |
| **Output :** |  |
|  |  |
| **Sr.No** | 34 |
| **SQL**  **Query :** | Show the name of all customers with their salesman's name. |
| **Solution :** | select c.cname,s.sname from customer c, salesman s where c.snum=s.snum; |
| **Output :** |  |
|  |  |

|  |  |
| --- | --- |
| **Sr.No** | 35 |
| **SQL**  **Query :** | List all customers and salesmen who shared a same city. |
| **Solution :** | select c.cnum,c.cname,s.snum,s.sname from customer c, salesman s where c.city=s.city; |
| **Output :** |  |
|  |  |
| **Sr.No** | 36 |
| **SQL**  **Query :** | List all orders with the names of their customer and salesman. |
| **Solution :** | select o.onum,o.amount,o.odate,c.cname,s.sname from orderr o, customer c, salesman s where o.cnum=c.cnum AND o.snum=s.snum; |
| **Output :** |  |
|  |  |
| **Sr.No** | 37 |
| **SQL**  **Query :** | List all orders by the customers not located in the same city as their salesman. |
| **Solution :** | select o.onum,o.amount,o.odate,c.cname,c.city,s.sname,s.city from orderr o, customer c, salesman s where o.cnum=c.cnum AND o.snum=s.snum AND c.city<>s.city; |

|  |  |
| --- | --- |
| **Output :** |  |
|  |  |
| **Sr.No** | 38 |
| **SQL**  **Query :** | List all customers serviced by salesman with commission above 12%. |
| **Solution :** | select c.cnum,c.cname,c.city,c.rating,c.snum,s.sname,s.commission from customer c, salesman s where c.snum=s.snum and s.commission>12; |
| **Output :** |  |
|  |  |
| **Sr.No** | 39 |
| **SQL**  **Query :** | Calculate the amount of the salesman commission on each order by customer with rating above 100. |
| **Solution :** | select o.onum,c.cname,o.amount,s.sname,s.commission, (o.amount\*s.commission)/100 AS “Commission  Amount” from orderr o, customer c, salesman s where o.cnum=c.cnum AND o.snum=s.snum  AND c.rating>100; |

|  |  |
| --- | --- |
| **Output :** |  |
|  |  |
| **Sr.No** | 40 |
| **SQL**  **Query :** | Find all pairs of customers having the same rating without duplication. |
| **Solution :** | Select c1.cname,c2.cname from customer c1, customer c2 where c1.rating = c2.rating and c1.cnum < c2.cnum; |
| **Output :** |  |
|  |  |
| **Sr.No** | 41 |
| **SQL**  **Query :** | List all customers located in cities where salesman Anjali has customers. |
| **Solution :** | Select \*from customer where snum = (Select snum from salesman where sname = 'anjali'); |

|  |  |
| --- | --- |
| **Output :** |  |
|  |  |
| **Sr.No** | 42 |
| **SQL**  **Query :** | Find all pairs of customers served by a single salesman with the salesman's name and number. |
| **Solution :** | Select c.cname, d.cname, c.cnum, s.sname from salesman s, customer c, customer d where c.snum = d.snum and c.cnum > d.cnum and c.snum = s.snum; |
| **Output :** |  |
|  |  |
| **Sr.No** | 43 |
| **SQL**  **Query :** | List all salesmen who are living in the same city without duplicate rows. |
| **Solution :** | Select a. \*from salesman a, salesman b where a.snum<>b.snum and a.city = b.city; |
| **Output :** |  |
|  |  |
| **Sr.No** | 44 |
| **SQL**  **Query :** | Produce the name and city of all the customers with the same rating as ‘Hardik’. |
| **Solution :** | Select cname, city from customer where rating = (select rating from customer where cname = ‘hardik’; |
| **Output :** |  |
|  |  |

|  |  |
| --- | --- |
| **Sr.No** | 45 |
| **SQL**  **Query :** | Extract all orders of Miti. |
| **Solution :** | Select \*from orderr where snum=(select snum from salesman where sname = ‘miti’; |
| **Output :** |  |
|  |  |
| **Sr.No** | 46 |
| **SQL**  **Query :** | Extract all orders of Baroda's salesmen. |
| **Solution :** | Select \*from orderr where snum in(select snum from salesman where city = ’baroda’); |
| **Output :** |  |
|  |  |
| **Sr.No** | 47 |
| **SQL**  **Query :** | Find all orders of the salesman who services 'Hardik’. |
| **Solution :** | Select \*from orderr where snum = (select snum from customer where cname = ’hardik’); |
| **Output :** |  |
|  |  |
| **Sr.No** | 48 |
| **SQL**  **Query :** | List all orders that are greater than the average of Feb 10, 2023. |
| **Solution :** | Select \*from orderr where amount>(select avg(amount) from orderr where odate = ’10-feb-2023’); |
| **Output :** |  |

|  |  |
| --- | --- |
|  |  |
| **Sr.No** | 49 |
| **SQL**  **Query :** | Find all orders attributed to salesmen in 'Bangalore'. |
| **Solution**  **:** | Select o.\*from orderr o,salesman s where o.snum = s.snum and s.city = ’bangalore’; |
| **Output :** |  |
|  |  |
| **Sr.No** | 50 |
| **SQL**  **Query :** | List the commission of all salesmen serving customers in 'Surat'. |
| **Solution**  **:** | select s.sname, s.commission from salesman s, customer c where c.snum = s.snum and c.city = 'surat’; |
| **Output :** |  |
|  |  |
| **Sr.No** | 51 |
| **Sql**  **Query:** | Insert a row into customer table with values London, Pratik a 2005 for the ColumnsCity, name and number. |
| **Solution:** | Insert into customer(city,cname,cnum) values(‘London’,’Pratik’,2009); |

|  |  |
| --- | --- |
| **OutPut:** |  |
|  |  |
| **Sr.No** | 52 |
| **Sql**  **Query:** | Create another table London staff having same structure as salesman table. |
| **Solution:** | create table londonstaff as select \* from salesmans; |
| **Output:** |  |
|  |  |
| **Sr.No** | 53 |
| **Sql**  **Query** | Insert all the rows of salesmen table with city London in the London staff table. |
| **Solution** | insert into londonstaff select \* from salesmans where city='london'; |
| **Output** |  |
|  |  |
| **Sr.No** | 54 |
| **Sql**  **Query** | Create another table Day totals with two-attribute date and total and insert rows into this table from order table |
| **Solution** | create table daytotal as select odate,sum(amount) "sumamount" from orders group by odate; |

|  |  |
| --- | --- |
| **Output** |  |
|  |  |
| **Sr.No** | 55 |
| **Sql**  **Query** | Create a duplicate of the salesmen table with a name Multicust. Now delete all the rows from the salesmen table |
| **Solution** | Create table Mulitcast as select \* from salesmans; |
| **OutPut** |  |
|  |  |
| **Sr.No** | 56 |
| **Sql**  **Query** | Get back all the rows of salesmen table from its duplicate table. |
| **Solution** | insert into salesmans select \* from Multicast; |

|  |  |
| --- | --- |
| **Output** |  |
|  |  |
| **Sr.No** | 57 |
| **Query** | Remove all orders from customer Chandresh from the orders table. |
| **Solution** | delete from orderr where cnum in(select cnum from customers where cname='chandresh'); |
| **Output** |  |
|  |  |
| **Sr.No** | 58 |
| **Query** | Set the ratings of all the customers of Piyush to 400. |
| **Solution** | update customers set rating=400 where snum in (select snum from salesmans where sname='piyush'); |

|  |  |
| --- | --- |
|  |  |
| **Sr.No** | 59 |
| **Sql**  **Query** | Increase the rating of all customers in Rome by 100. |
| **Solution** | update customers set rating=rating+100 where city='rome'; |
| **OutPut** |  |

|  |  |
| --- | --- |
| **Output** |  |

|  |  |
| --- | --- |
|  |  |
| **Sr.No** | 60 |
| **Sql**  **Query** | Salesman Miti has resigned. Reassign her number to a new salesman Gopal whose city is Bombay and commission is 10%. |
| **Solution** | update salesmans set sname='gopal',city='bombay',commission=10 where sname='miti'; |
| **OutPut** |  |

|  |  |
| --- | --- |
|  |  |
| **Sr.No** | 61 |
| **Sql**  **Query** | Double the commission of all salesmen of London. |
| **Solution** | update salesmans set commission = commission\*2 where city='london'; |
| **OutPut** |  |

|  |  |
| --- | --- |
|  |  |
| **Sr.No** | 62 |
| **Sql**  **Query** | Set ratings for all customers in London to NULL. |
| **Solution** | update customers set rating='null' where city='london'; |
| **OutPut** |  |

|  |  |
| --- | --- |
|  |  |
| **Sr.No** | 63 |
| **Sql**  **Query** | Suppose we have a table called sales Manager with the same definition as Salesmen table. Company decides to promote salesmen having total order more than 5000 to Sales Manager. Fill up the Sales Manager table. |
| **Solution** | create table salesmanager31 as select \* from salesmans; delete salesmanager31;  insert into salesmanager31 select \* from salesmans where snum in(select snum from orders group by snum having sum(amount)>5000); |
| **OutPut** |  |

|  |  |
| --- | --- |
|  |  |
| **Sr.No** | 64 |
| **Sql**  **Query** | Assume that we have a table called smcity. Store the information of all salesmen with the customers in their home cities into smcity |
| **Solution** | create table smcity31 as select distinct s.\* from salesmans s,customers c where s.snum=c.snum and s.city=c.city; |
| **OutPut** |  |

|  |  |
| --- | --- |
|  |  |
| **Sr.No** | 65 |
| **Sql** | Create a table Bonus that contains date wise maximum amount of order for all salesmen. |
| **Query** |  |
| **Solution** | create table bonus31 as select snum,odate,max(amount) "Max" from orderr group by odate,snum; |
| **OutPut** |  |

|  |  |
| --- | --- |
|  |  |
| **Sr.No** | 66 |
| **Sql**  **Query** | Create a table Multicust containing the salesmen with more than one customer. |
| **Solution** | create table multicast31 as select \* from salesmans where snum in(select snum from customers group by snum having count(snum)>1); |
| **OutPut** |  |
|  |  |
| **Sr.No** | 67 |
| **Sql**  **Query** | New Delhi office has closed. Remove all customers assigned to salesmen in New Delhi. |
| **Solution** | delete from salesmans where snum in(select snum from salesmans where city='new delhi'); |
| **OutPut** |  |
|  |  |
| **Sr.No** | 68 |
| **Sql**  **Query** | Delete all salesmen who have at least one customer with a rating of 100 from salesmen table |
| **Solution** | delete from salesmans where snum in(select snum from customers where rating=100); |
| **OutPut** |  |

|  |  |
| --- | --- |
|  |  |
| **Sr.No** | 69 |
| **Sql**  **Query** | Delete the salesmen who produce the lowest order for each day. |
| **Solution** | delete from salesmans where snum in (select snum from orderr where amount in(select min(amount) from orderr group by odate)); |
| **OutPut** |  |
|  |  |
| **Sr.No** | 70 |
| **Sql**  **Query** | Find the smallest order for each day. Reduce the commission of all salesmen by 2% who produce this order |
| **Solution** | Update salesman s set commission-2 where exists (select \* from order o where s.snum=o.snum and amount in(select min(amount) from order group by odate)); |
| **OutPut** |  |
| **Sr.No** | 71 |
| **Sql**  **Query** | Delete all customers with no current orders. |
| **Solution** | delete from customers where cnum not in(select distinct(cnum) from orderr ); |
| **OutPut** |  |
| **Sr.No** | 72 |
| **Sql**  **Query** | Write a command to find out the orders by date. |
| **Solution** | select \* from orderr order by odate |
| **OutPut** |  |

|  |  |
| --- | --- |
| **Sr.No** | 1 |
| **Sql**  **Query** | Create a generic PL/SQL block that prints “Hello, PL/SQL!” to the console. (Generic PL/SQL Block) |
| **Solution** | DECLARE  BEGIN  dbms\_output.put\_line('Hello, PL/SQL!');  END;  / |
| **OutPut** |  |

|  |  |
| --- | --- |
| **Sr.No** | 2 |
| **Sql**  **Query** | Write a PL/SQL block to declare a variable, assign it a value, and display it. |
| **Solution** | DECLARE     num1 NUMBER:=10;     num2 NUMBER:=20;     result NUMBER;  BEGIN     result:=num1+num2;     DBMS\_OUTPUT.PUT\_LINE('The sum is:' || result);  END;  / |
| **OutPut** |  |

|  |  |
| --- | --- |
| **Sr.No** | 3 |
| **Sql**  **Query** | Write a PL/SQL block to handle an exception when a division by zero occurs. |
| **Solution** | DECLARE     v\_num1 NUMBER:=10;     v\_num2 NUMBER:=0;     v\_result NUMBER;  BEGIN     v\_result:=v\_num1/v\_num2;     DBMS\_OUTPUT.PUT\_LINE('Result: '|| v\_result);  EXCEPTION     WHEN ZERO\_DIVIDE THEN        DBMS\_OUTPUT.PUT\_LINE('Error: Division by zero');  END;  / |
| **OutPut** |  |

|  |  |
| --- | --- |
| **Sr.No** | 4 |
| **Sql**  **Query** | Write a PL/SQL block to use a FOR loop to print numbers from 1 to 10. |
| **Solution** | BEGIN     For i IN 1..10 LOOP         DBMS\_OUTPUT.PUT\_LINE('Iteration' || i || ' : Hello,this is number '|| i);      END LOOP;  END;  / |
| **OutPut** |  |

|  |  |
| --- | --- |
| **Sr.No** | 5 |
| **Sql**  **Query** | Write a PL/SQL function to add two numbers. |
| **Solution** | CREATE OR REPLACE FUNCTION add\_numbers(p\_num1 NUMBER,p\_num2 NUMBER)  RETURN NUMBER IS  BEGIN    RETURN p\_num1+p\_num2;    END;  / |
| **OutPut** |  |

|  |  |
| --- | --- |
| **Sr.No** | 6 |
| **Sql**  **Query** | Write a PL/SQL procedure that takes an employee's ID and prints their name. |
| **Solution** | DECLARE    v\_user\_id NUMBER;    v\_first\_name VARCHAR2(50);    v\_last\_name VARCHAR2(50);    v\_email VARCHAR2(100);    v\_phone\_number NUMBER(15);  BEGIN    v\_user\_id:=&user\_id;      SELECT first\_name,last\_name,email,phone\_number into v\_first\_name,v\_last\_name,v\_email,v\_phone\_number FROM users WHERE user\_id=v\_user\_id;    DBMS\_OUTPUT.PUT\_LINE('User Details;');    DBMS\_OUTPUT.PUT\_LINE('First Name' || v\_first\_name);    DBMS\_OUTPUT.PUT\_LINE('Last Name' || v\_last\_name);    DBMS\_OUTPUT.PUT\_LINE('Email' || v\_email);    DBMS\_OUTPUT.PUT\_LINE('Phone Number' || v\_phone\_number);  EXCEPTION     WHEN NO\_DATA\_FOUND THEN          DBMS\_OUTPUT.PUT\_LINE('No usr found with the provided ID,');     WHEN OTHERS THEN          DBMS\_OUTPUT.PUT\_LINE('An error occured'  ||  SQLERRM);  END;  / |
| **OutPut** |  |

|  |  |
| --- | --- |
| **Sr.No** | 7 |
| **Sql**  **Query** | Write a PL/SQL function to add two numbers. |
| **Solution** | CREATE OR REPLACE FUNCTION add\_numbers(p\_num1 NUMBER,p\_num2 NUMBER)  RETURN NUMBER IS  BEGIN    RETURN p\_num1+p\_num2;    END;  / |
| **OutPut** |  |

|  |  |
| --- | --- |
| **Sr.No** | 8 |
| **Sql**  **Query** | Create a user-defined exception and handle it in a PL/SQL block. ( Error Handling in PL/SQL (User Define)) |
| **Solution** | DECLARE      my\_exception EXCEPTION;  BEGIN      RAISE my\_exception;      EXCEPTION          WHEN my\_exception THEN              DBMS\_OUTPUT.PUT\_LINE('User defined exception caught');              DBMS\_OUTPUT.PUT\_LINE('Program Developed by Ashvin Parmar');              DBMS\_OUTPUT.PUT\_LINE('Enrollment No: 24034211039');  END;  / |
| **OutPut** |  |

|  |  |
| --- | --- |
| **Sr.No** | 9 |
| **Sql**  **Query** | Declare and use an implicit cursor in a PL/SQL block. (Types of  Cursors)  (1) Create the departments table  (2) Insert records into the departments table  (3) Create the employees table  (4) Insert records into the employees table  (5) Write a PL/SQL block for Declare and use an implicit cursor |
| **Solution** | DECLARE      v\_emp\_name VARCHAR2(50);    BEGIN      SELECT emp\_name INTO v\_emp\_name FROM employees WHERE emp\_id=105;      DBMS\_OUTPUT.PUT\_LINE('EMP Name : ' || v\_emp\_name);      DBMS\_OUTPUT.PUT\_LINE('Program Developed by Ashvin Parmar');      DBMS\_OUTPUT.PUT\_LINE('Enrollment No: 24034211039');      EXCEPTION      WHEN NO\_DATA\_FOUND THEN          DBMS\_OUTPUT.PUT\_LINE('EMPLOYEE NOT FOUND');  END;  / |
| **OutPut** |  |

|  |  |
| --- | --- |
| **Sr.No** | 10 |
| **Sql**  **Query** | Use a cursor with a loop to fetch and display all employee names.  (Cursor with Loops)  (1) Create the departments table  (2) Insert records into the departments table  (3) Create the employees table  (4) Insert records into the employees table  (5) Write a PL/SQL block for Use a cursor with a loop to fetch and  display all employee names. |
| **Solution** | DECLARE      CURSOR employee\_cursor IS          SELECT emp\_name FROM employees;      v\_emp\_name VARCHAR2(50);  BEGIN      --cursor with loop      FOR emp\_rec IN employee\_cursor loop          v\_emp\_name := emp\_rec.emp\_name;          DBMS\_OUTPUT.PUT\_LINE('Employee Name: ' || v\_emp\_name);      END LOOP;      DBMS\_OUTPUT.PUT\_LINE('Program Developed by Ashvin Parmar');      DBMS\_OUTPUT.PUT\_LINE('Enrollment No: 24034211039');  END;  / |
| **OutPut** |  |

|  |  |
| --- | --- |
| **Sr.No** | 11 |
| **Sql**  **Query** | Create a parameterized cursor to fetch details for a specific  department. (Parameterized Cursor)  (1) Create the departments table  (2) Insert records into the departments table  (3) Create the employees table  (4) Insert records into the employees table  (5) Write a PL/SQL block for create a parameterized cursor to  fetch details for a specific department |
| **Solution** | DECLARE      CURSOR dept\_cursor(p\_dept\_id NUMBER) IS      SELECT emp\_name FROM employees WHERE dept\_id=p\_dept\_id;      v\_emp\_name VARCHAR2(50);      v\_dep\_id NUMBER :=2;  BEGIN      FOR emp\_rec IN dept\_cursor(v\_dep\_id) LOOP          v\_emp\_name := emp\_rec.emp\_name;          DBMS\_OUTPUT.PUT\_LINE('Employee Name: '|| v\_emp\_name);      END LOOP;      DBMS\_OUTPUT.PUT\_LINE('Program Developed by Ashvin Parmar');      DBMS\_OUTPUT.PUT\_LINE('Enrollment No: 24034211039');  END;  / |
| **OutPut** |  |

|  |  |
| --- | --- |
| **Sr.No** | 12 |
| **Sql**  **Query** | Use a nested cursor to fetch and display employee names for  each department. (Nested Cursor)  (1) Create the departments table  (2) Insert records into the departments table  (3) Create the employees table  (4) Insert records into the employees table  (5) Write a PL/SQL block for Use a nested cursor to fetch and  display employee names for each department. |
| **Solution** | DECLARE      CURSOR dept\_cursor IS          SELECT dept\_id FROM departments;      CURSOR emp\_cursor(p\_dept\_id NUMBER) IS          SELECT emp\_name FROM employees WHERE dept\_id=p\_dept\_id;          v\_emp\_name VARCHAR2(50);  BEGIN      --NESTED CURSOR      FOR dep\_rec IN dept\_cursor LOOP          DBMS\_OUTPUT.PUT\_LINE('Dept Id: ' || dep\_rec.dept\_id);          FOR emp\_rec IN emp\_cursor(dep\_rec.dept\_id) LOOP              v\_emp\_name := emp\_rec.emp\_name;              DBMS\_OUTPUT.PUT\_LINE('Employee Name: ' || v\_emp\_name);          END LOOP;          DBMS\_OUTPUT.PUT\_LINE('Program Developed by Ashvin Parmar');          DBMS\_OUTPUT.PUT\_LINE('Enrollment No: 24034211039');      END LOOP;  END;  / |
| **OutPut** |  |

|  |  |
| --- | --- |
| **Sr.No** | 13 |
| **Sql**  **Query** | Write a PL/SQL stored procedure that takes an employee ID as an  input parameter and prints the employee’s details. |
| **Solution** | CREATE OR REPLACE PROCEDURE getEmployeeDetails(p\_emp\_id IN NUMBER ) AS  v\_emp\_name VARCHAR2(50);  v\_emp\_salary NUMBER;  BEGIN      SELECT ename,salary INTO v\_emp\_name,v\_emp\_salary FROM employee WHERE eid=p\_emp\_id;      DBMS\_OUTPUT.PUT\_LINE('EMP NAME: ' || v\_emp\_name);      DBMS\_OUTPUT.PUT\_LINE('EMP SALARY: ' || v\_emp\_salary);    END getEmployeeDetails;  /  DECLARE      v\_emp\_id NUMBER := 2;  BEGIN      getEmployeeDetails(v\_emp\_id);      DBMS\_OUTPUT.PUT\_LINE('Program Developed by Ashvin Parmar');      DBMS\_OUTPUT.PUT\_LINE('Enrollment No: 24034211039');  END;  / |
| **OutPut** |  |

|  |  |
| --- | --- |
| **Sr.No** | 14 |
| **Sql**  **Query** | Write a PL/SQL stored procedure to include an OUT parameter  for the employee’s department |
| **Solution** | CREATE OR REPLACE PROCEDURE getEmployeeDetailswithDept(p\_emp\_id NUMBER,p\_dept OUT VARCHAR2,p\_emp\_name OUT VARCHAR2) AS   BEGIN      SELECT emp\_name,dept\_id INTO p\_emp\_name,p\_dept FROM employees WHERE emp\_id=p\_emp\_id;  END getEmployeeDetailswithDept;  /  DECLARE      v\_emp\_id NUMBER := 105;      v\_ename VARCHAR2(20);      v\_dept VARCHAR2(20);  BEGIN      getEmployeeDetailswithDept(v\_emp\_id,v\_dept,v\_ename);      DBMS\_OUTPUT.PUT\_LINE('EMPLOYEE NAME: ' || v\_ename ||', '|| 'DEPT ID: ' || v\_dept);      DBMS\_OUTPUT.PUT\_LINE('Program Developed by Ashvin Parmar');      DBMS\_OUTPUT.PUT\_LINE('Enrollment No: 24034211039');  END;  / |
| **OutPut** |  |

|  |  |
| --- | --- |
| **Sr.No** | 15 |
| **Sql**  **Query** | Develop a trigger that captures the old and new values when a customer’s record is inserted, updated and updated.  1) Create a Customers table.  2) Insert record in customer table.  3) Develop a trigger that captures the old and new values when a customer’s record is inserted, updated and deleted |
| **Solution** | create or replace trigger display\_salary\_changes  before delete or insert or update on customerr  for each row  when(new.ID > 0)  DECLARE   sal\_diff number;  BEGIN   sal\_diff:= :new.salary - :old.salary;   DBMS\_OUTPUT.PUT\_LINE('Old Salary: ' || :old.salary);   DBMS\_OUTPUT.PUT\_LINE('New Salary: ' || :new.salary);   DBMS\_OUTPUT.PUT\_LINE('Salary difference: ' || sal\_diff);   DBMS\_OUTPUT.PUT\_LINE('Program Developed by Ashvin Parmar');   DBMS\_OUTPUT.PUT\_LINE('Enrollment No: 24034211039');  END;  / |
| **OutPut** |  |

|  |  |
| --- | --- |
| **Sr.No** | 16 |
| **Sql**  **Query** | Develop a trigger that captures the old and new values when an customer salary is updated. Develop a trigger that captures the old and new values when an customer salary is updated. |
| **Solution** | create or replace trigger display\_salary\_changes  before delete or insert or update on customerr  for each row  when(new.ID > 0)  DECLARE   sal\_diff number;  BEGIN   sal\_diff:= :new.salary - :old.salary;   DBMS\_OUTPUT.PUT\_LINE('Old Salary: ' || :old.salary);   DBMS\_OUTPUT.PUT\_LINE('New Salary: ' || :new.salary);   DBMS\_OUTPUT.PUT\_LINE('Salary difference: ' || sal\_diff);   DBMS\_OUTPUT.PUT\_LINE('Program Developed by Ashvin Parmar');   DBMS\_OUTPUT.PUT\_LINE('Enrollment No: 24034211039');  END;  /  update customerr set salary=salary+500 where id=1;  select \* from customerr;  DECLARE    total\_rows number(2);    BEGIN       if sql%notfound then   DBMS\_OUTPUT.PUT\_LINE('No customer update');      else        total\_rows:=sql%rowcount;   DBMS\_OUTPUT.PUT\_LINE(total\_rows || ' customer updated');   end if;   end;   / |
| **OutPut** |  |

|  |  |
| --- | --- |
| **Sr.No** | 17 |
| **Sql**  **Query** | The provided PL/SQL code defines a package named math operations that encapsulates two mathematical operations: addition and multiplication. The package consists of a specification and a body |
| **Solution** | CREATE OR REPLACE PACKAGE math\_operations IS     FUNCTION add\_numbers(num1 NUMBER,num2 NUMBER) RETURN NUMBER;     FUNCTION multiply\_numbers(num1 NUMBER,num2 NUMBER) RETURN NUMBER;  END math\_operations;  /  CREATE OR REPLACE PACKAGE BODY math\_operations IS      FUNCTION add\_numbers(num1 NUMBER,num2 NUMBER)RETURN NUMBER IS         BEGIN           RETURN num1+num2;      END add\_numbers;        FUNCTION multiply\_numbers(num1 NUMBER,num2 NUMBER)RETURN NUMBER  IS      BEGIN          RETURN num1\*num2;      END multiply\_numbers;    END math\_operations;  /  BEGIN     DBMS\_OUTPUT.PUT\_LINE('Addition: ' || math\_operations.add\_numbers(10,20));     DBMS\_OUTPUT.PUT\_LINE('Multiplication: ' || math\_operations.multiply\_numbers(10,20));     DBMS\_OUTPUT.PUT\_LINE('Program developed by Ashvin Parmar');     DBMS\_OUTPUT.PUT\_LINE('Enrollment No:24034211039');  END;  / |
| **OutPut** |  |

|  |  |
| --- | --- |
| **Sr.No** | 18 |
| **Sql**  **Query** | The provided PL/SQL code defines a package named c\_package that encapsulates use the CUSTOMERS table stored in our database and insert record |
| **Solution** | CREATE OR REPLACE PACKAGE c\_package AS  -- Adds a customer  PROCEDURE addCustomer( c\_id customerr.id%type, c\_name customerr.name%type, c\_age customerr.age%type, c\_addr customerr.address%type, c\_sal customerr.salary%type);  -- Removes a customer  PROCEDURE delCustomer(c\_id customerr.id%TYPE);  --Lists all customers  PROCEDURE listCustomer;  END c\_package;  /  CREATE OR REPLACE PACKAGE BODY c\_package AS  PROCEDURE addCustomer(c\_id customerr.id%type, c\_name customerr.name%type, c\_age customerr.age%type, c\_addr customerr.address%type, c\_sal customerr.salary%type)  IS  BEGIN  INSERT INTO customerr (id,name,age,address,salary)  VALUES(c\_id, c\_name, c\_age, c\_addr, c\_sal);  END addCustomer;  PROCEDURE delCustomer(c\_id customerr.id%type) IS  BEGIN  DELETE FROM customerr  WHERE id = c\_id;  END delCustomer;  PROCEDURE listCustomer IS  CURSOR c\_customers is  SELECT name FROM customerr;  TYPE c\_list is TABLE OF customerr.Name%type;  name\_list c\_list := c\_list(); counter integer :=0;  BEGIN  FOR n IN c\_customers LOOP counter := counter +1; name\_list.extend; name\_list(counter) := n.name;  dbms\_output.put\_line('Customer(' ||counter|| ')'||name\_list(counter));  END LOOP;  END listCustomer;  END c\_package;  /  DECLARE  code customerr.id%type:= 1;  BEGIN  c\_package.addcustomer(4, 'Rajnish', 25, 'Chennai', 3500);  c\_package.addcustomer(5, 'Subham', 32, 'Delhi', 7500);  c\_package.listcustomer; c\_package.delcustomer(code);  c\_package.listcustomer;  END;  / |
| **OutPut** |  |