Kamani Science College And Prataprai Arts College - Amreli

"VIDYAVIHAR", OPP. ITI, BHAVNAGAR ROAD, AMRELI, GUJARAT, INDIA, PIN CODE 365601

Computer Science Department

B.C.A.

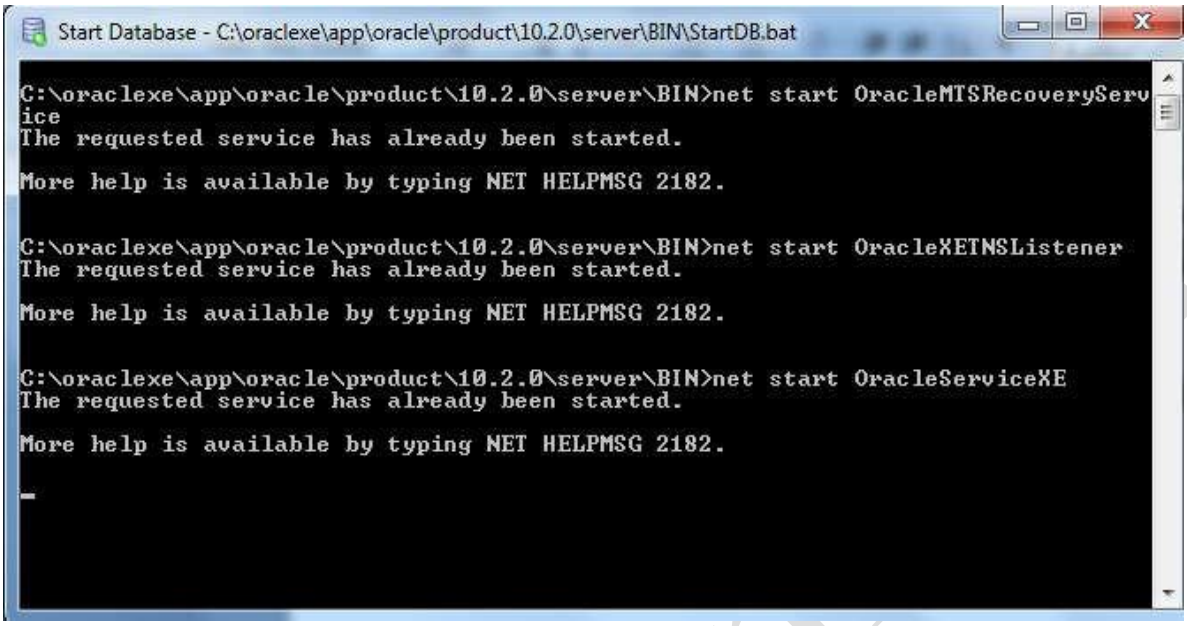
Lab Manual

Oracle

INDEX

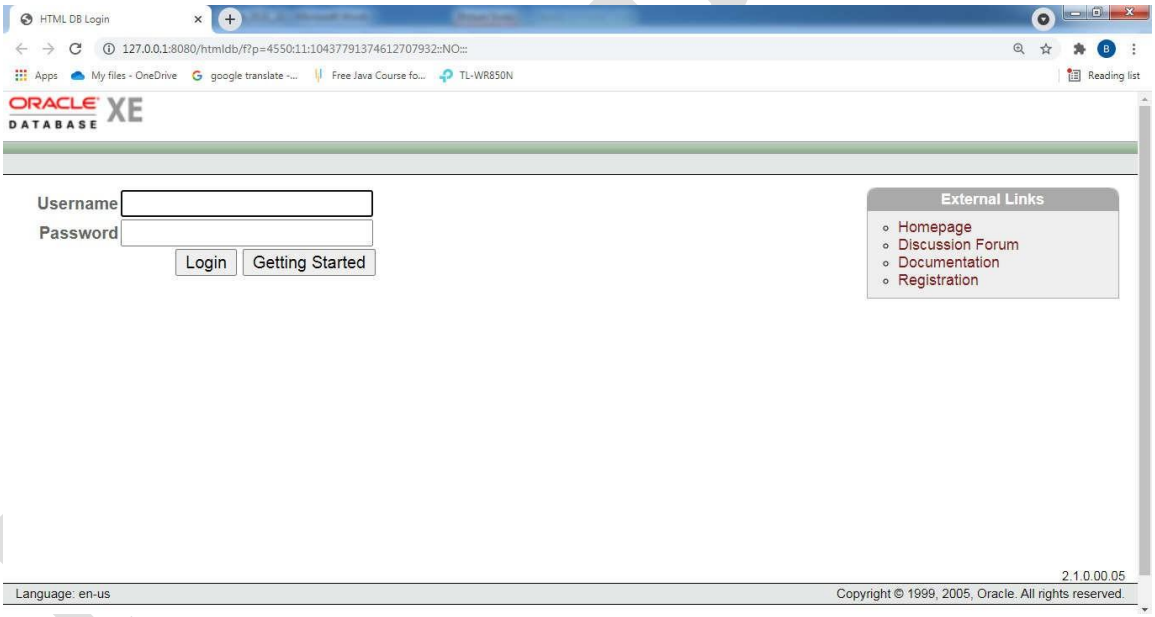
Step:1

Click on start→All Programs→Oracle Database10g express edition→Start Database



Step 2:

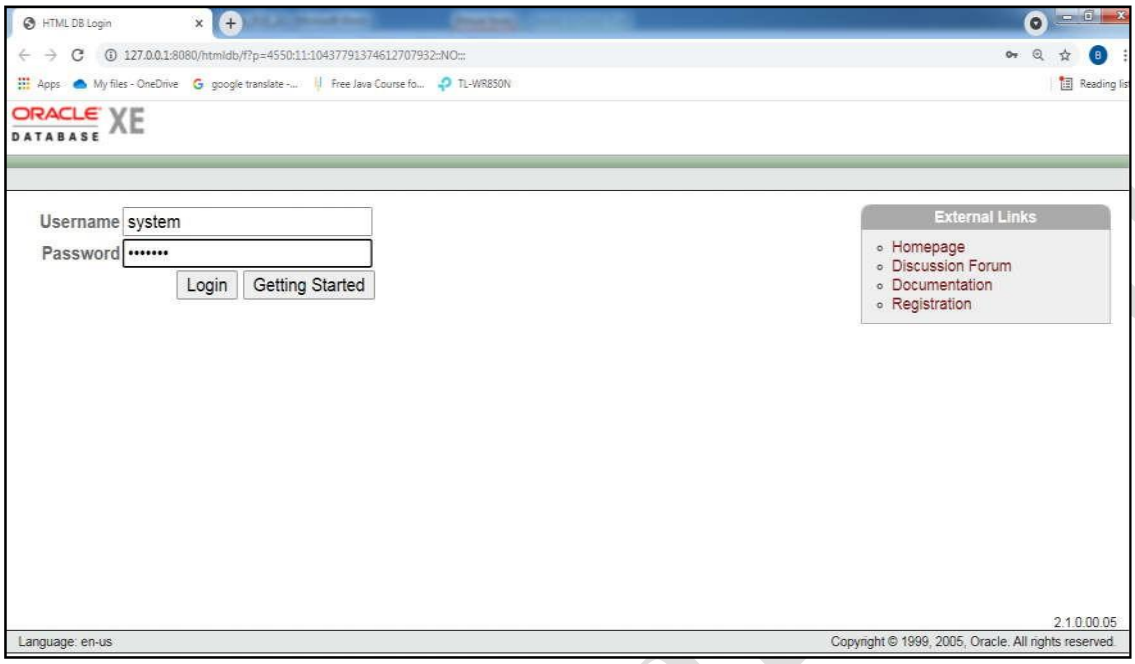
Click on Go To Database Home Page icon on desktop.



Step 3:

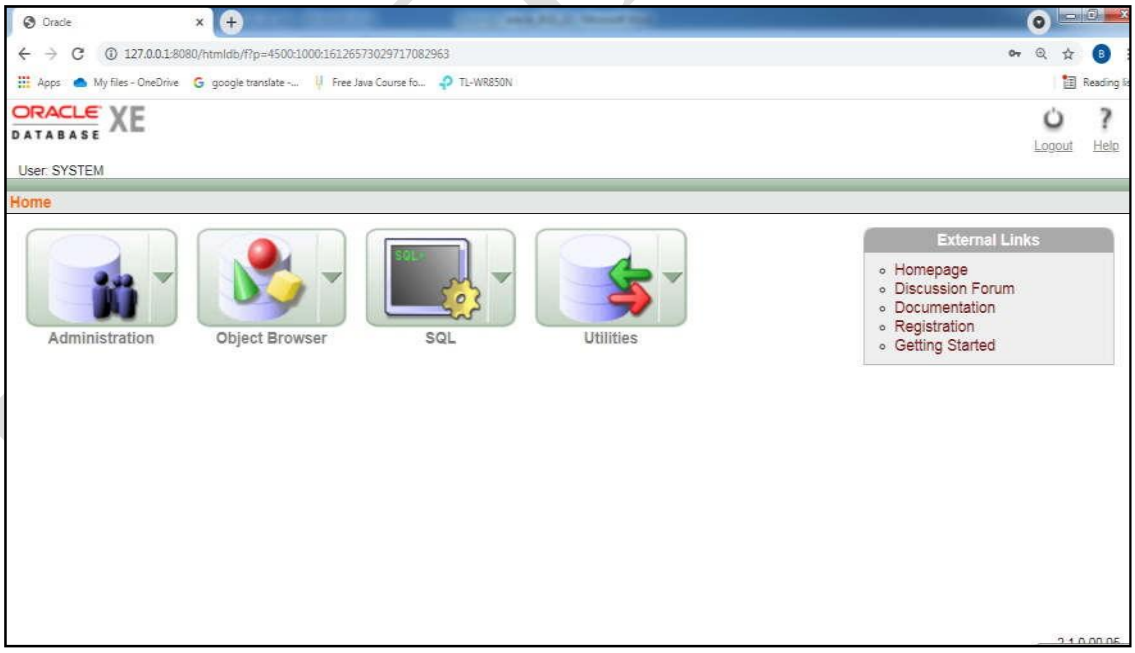
Give Username : system

& Password : manager and click on login button.

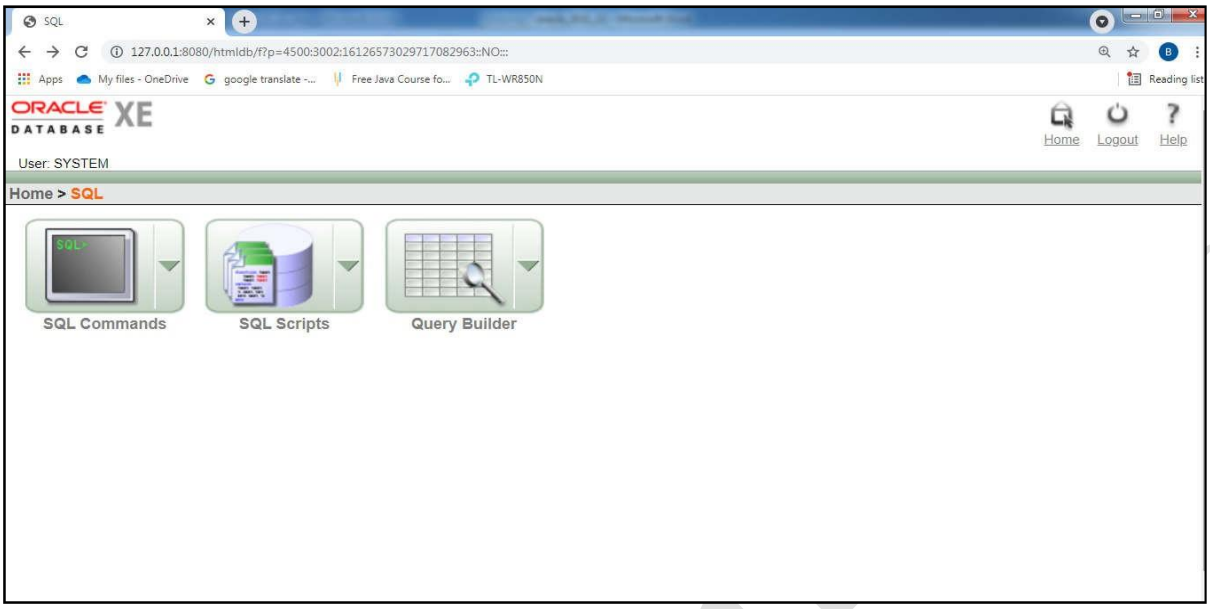


Step 4:

You will get following Screen. Now click on SQL.

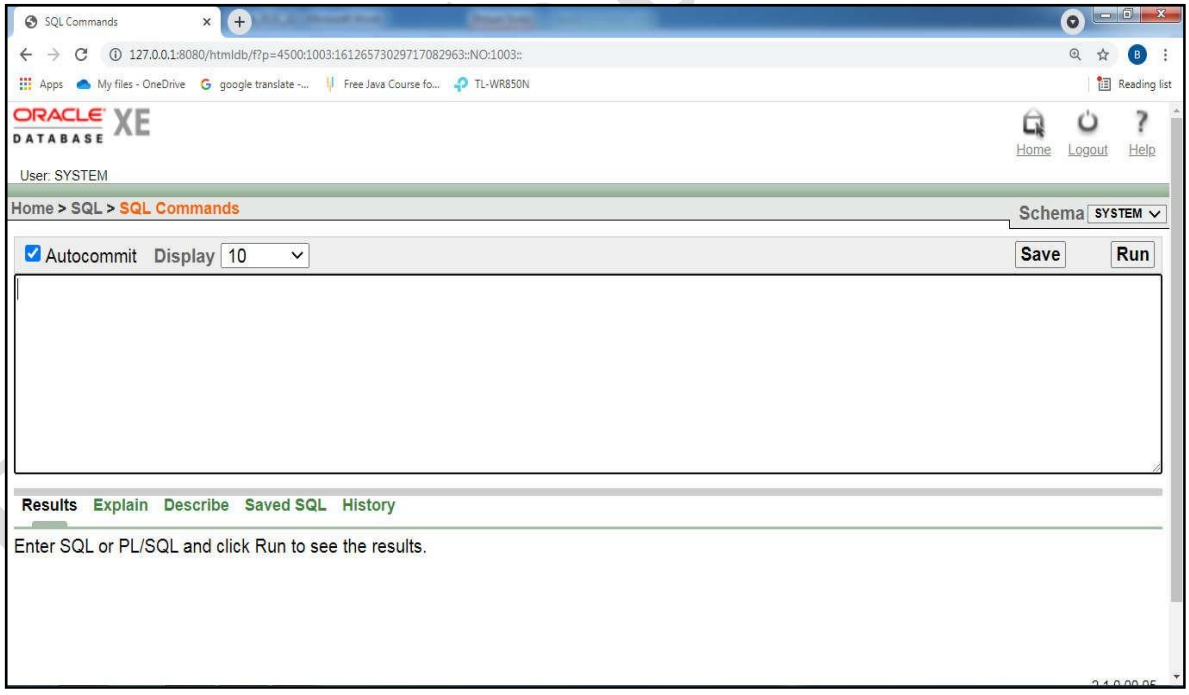


Then You Click on SQL Command → Enter Command



Step 5:

You will get following Screen. Now, perform Your Queries.



UNIT : 2
Managing Tables and Data Control and Transaction Control Command
Exercise-1

Create a table Student which include these fields, Insert records in the table then perform a given queries:

A. Create the table described below:

Table Name : Student
Description : Used to store Student information.

Column Name	Data Type	Size
Roll_No	Varchar2	6
Name	Varchar2	20
Marks	Number	3

Create Table:

```
create table Student
(Roll_No varchar2 (6),
Name varchar2 (20),
Marks number (3));
```

Output:

Table Created.

B. Data for Student table:

ROLL_NO	NAME	MARKS
S00001	Ishita	70
S00002	Ruhi	65
S00003	Twinkal	80

```
insert into Student (Roll_No,Name,Marks)values('S00001', 'Ishita',70);
```

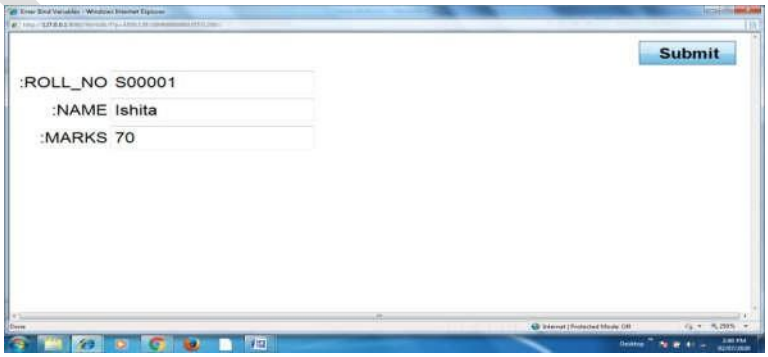
or

```
insert into Student values('S00001', 'Ishita',70);
```

or

```
insert into Student values (:Roll_No,:Name,:Marks);
```

Note: Below screen will be display and click on submit button then following output will come



Output:

1 row(s) inserted.

C. Exercise on retrieving records from a table

1. To view the structure of table.
   Describe Student;

2. Retrieve the entire contents of the Student table.
   Select * from Student;

3. Retrieve the only Roll_No column from Student table.
   Select Roll_No from Student;

4. Change the marks of Roll_No 'S00001' to 80.
   Update Student Set Marks = 80 Where Roll_No = 'S00001';

5. Delete from Student where the column name holds the value Twinkal
   Delete from Student Where name= 'Twinkal';

6. Change the name of the Student table to Stud
   Rename Student to Stud;

Exercise-2

Create a table Client which include these fields, Insert records in the table then perform a given queries:

A. Create the table described below:

Table Name : Client
Description : Used to store Client information.

Column Name	Data Type	Size
Client_No	Varchar2	6
Name	Varchar2	20
City	Varchar2	15
PinCode	Number	6
State	Varchar2	15
BalDue	Number	10

Create Table:

```
create table Client
(Client_No varchar2 (6),
Name varchar2 (20),
City varchar2 (15),
PinCode number (6),
State varchar2 (15),
BalDue number (10));
```

Output:

Table Created.

B. Data for Client table:

CLIENT_NO	NAME	CITY	PINCODE	STATE	BALDUE
C00001	Ivan Bayross	Mumbai	400054	Maharashtra	15000
C00002	Mamta Muzumdar	Madras	780001	TamilNadu	0
C00003	Chhaya Bankar	Mumbai	400057	Maharashtra	5000
C00004	Ashwini Joshi	Bangalore	560001	Karnataka	0
C00005	Hansel Colaco	Mumbai	400060	Maharashtra	20000

```
insert into Client(Client_No,Name,City,PinCode,State,BalDue)values('C00001','Ivan
Bayross','Mumbai',400054,'Maharashtra',15000);
```

Output:

1 row(s) inserted.

C. Exercise on retrieving records from a table

1. Find out the names of all the Clients.
   Select Name from Client;

2. Retrieve the entire contents of the Client table.
   Select * from Client;

3. Retrieve the list of Names, City and the State of all the Clients.
   Select Name, City, State from Client;

4. List all the Clients who are located in Mumbai.
   Select Client_No, Name from Client Where City = 'Mumbai';

5. Change the size of city column to 20 in Client.
   Alter table Client Modify (city varchar2(20));

6. Change the city of Client_No ‘C00005’ to ‘Pune’.
   Update Client Set City = 'Pune' Where Client_No = 'C00005';

7. Change the BalDue of Client_No ‘C00001’ to Rs. 1000.
   Update Client Set BalDue = 1000 Where Client_No = 'C00001';

8. Delete from Client where the column state holds the value ‘TamilNadu’.
   Delete from Client Where State = 'TamilNadu';

9. Add a column called ‘Contact_no’ of data type ‘number’ and size = ‘10’ to the Client table.
   Alter table Client Add (Contact_no number (10)) ;

10. Change the column name Contact_no to Mobile_no
    Alter table Client Rename Column Contact_no to Mobile_no;

11. Delete the column Mobile_no from Client.
    Alter table Client Drop Column Mobile_no;

12. Destroy the table Client along with its data.
    Drop table Client;

Exercise-3

Create a table Client_Master which include these fields, Insert records in the table then perform a given queries:

A. Create the table described below:

Table Name : Client_Master
Description : Used to store Client information.

Column Name	Data Type	Size	Attributes
Client_No	Varchar2	6	Primary key/First Letter Start with C
Name	Varchar2	20	Not null
City	Varchar2	15	
Pincode	Number	6	
State	Varchar2	15	
Baldue	Number	10,2	

Create Table:

```
create table Client_Master
(Client_No varchar2(6) Primary Key,
Name varchar2(20) not null,
City varchar2(15),
Pincode number(6),
State varchar2(15),
Baldue number(10,2),
Check(Client_No like 'C%'));
```

Output:

Table Created.

B. Data for Client_Master table:

CLIENT_NO	NAME	CITY	PINCODE	STATE	BALDUE
C00001	Ivan Bayross	Mumbai	400054	Maharashtra	15000
C00002	Mamta Muzumdar	Madras	780001	TamilNadu	0
C00003	Chhaya Bankar	Mumbai	400057	Maharashtra	5000
C00004	Ashwini Joshi	Bangalore	560001	Karnataka	0
C00005	Hansel Colaco	Mumbai	400060	Maharashtra	20000
C00006	Deepak Sharma	Mangalore	560050	Karnataka	0

```
insert into Client_Master(Client_No,Name,City,PinCode,State,BalDue)values('C00001','Ivan
Bayross','Mumbai',400054,'Maharashtra',15000);
```

Output:

1 row(s) inserted.

C. Exercise on retrieving records from a table

1. List the name of all Clients having ‘a’ as the second letter in their names.
   Select Name from Client_Master where Name like '_a%';

2. List the Client no, Name who stay in a city whose first letter is ‘M’.
   Select Client_no, Name from Client_Master where City like 'M%';

3. List all Clients who stay in ‘Bangalore’ or ‘Mangalore’.
   Select * from Client_Master where City in ('Bangalore', 'Mangalore');

4. List all Clients whose Baldue is grater than value 10000.
   Select * from Client_Master where Baldue>10000;

5. List the name, city and state of Clients who are not in the state of ‘Maharashtra’.
   Select Name, City, State from Client_Master where State not in ('Maharashtra');

Exercise-4

Create a table Product_Master which include these fields, Insert records in the table then perform a given queries.

A. Create the table described below:

Table Name : Product_Master
Description : Used to store Product information.

Column Name	Data Type	Size	Attributes
Product_No	Varchar2	6	Primary key/ First Letter Start with P
Description	Varchar2	15	Not null
ProfitPercent	Number	4,2	Not null
UnitMeasure	Varchar2	10	Not null
QtyOnHand	Number	8	Not null
ReOrderLvl	Number	8	Not null
SellPrice	Number	8,2	Not null, Cannot be 0
CostPrice	Number	8,2	Not null, Cannot be 0

Create Table:

```
create table Product_Master
(Product_No varchar2(6) Primary Key,
Description varchar2(15) not null,
ProfitPercent number(4,2) not null,
UnitMeasure varchar2(10) not null,
QtyOnHand number(8) not null,
ReOrderLvl number(8)not null,
SellPrice number(8,2)not null,
CostPrice number(8,2)not null,
Check(Product_No like 'P%'),
Check(SellPrice<>0),
Check(CostPrice<>0));
```

Output:

Table Created.

B. Data for Product_Master table:

PRODUCT_NO	DESCRIPTION	PROFITPERCENT	UNITMEASURE	QTYONHAND	REORDERLVL	SELLPRICE	COSTPRICE
P00001	T-Shirt	5	Piece	200	50	350	250
P00002	Shirt	6	Piece	150	50	500	350
P00003	Cotton Jeans	5	Piece	100	20	600	450
P00004	Jeans	5	Piece	100	20	750	500
P00005	Trousers	2	Piece	150	50	850	550
P00006	Pull Overs	2.5	Piece	80	30	700	450
P00007	Denim Shirts	4	Piece	100	40	350	250
P00008	Lycra Tops	5	Piece	70	30	300	175
P00009	Skirts	5	Piece	75	30	450	300

```
insert into Product_Master (Product_No,Description,ProfitPercent,UnitMeasure,QtyOnHand,ReOrderLvl,
SellPrice,CostPrice)values('P00001','T-Shirt',5,'Piece',200,50,350,250);
```

Output:

1 row(s) inserted.

C. Exercise on retrieving records from a table

1. List products whose selling price is greater than 500 and less than or equal to 750.
   Select Product_No, Description from Product_Master Where SellPrice >= 500 and SellPrice < =750;
   Or
   Select * From Product_Master Where Sellprice Between 500 And 750;

2. List products whose selling price is more than 500. Calculate a new selling price as original selling price * 15. Rename the new column in the output of the above query as New_Price.
   Select Product_No, Description, SellPrice, SellPrice*15 "New_Price" from Product_Master Where SellPrice > 500;

3. Calculate the average price of all the products.
   Select Avg (SellPrice), Avg(CostPrice) from Product_Master;

4. Determine the maximum and minimum product prices. Rename the output as Max_Price and Min_Price respectively.
   Select Max(SellPrice) "Max_Price", Min(SellPrice) "Min_Price" from Product_Master;

5. Calculate square root of price of each product.
   Select Sqrt(SellPrice),Sqrt(CostPrice) from Product_Master;

6. Count the number of products having price less than or equal to 500
   Select Count (Product_No) from Product_Master Where SellPrice <= 500;

7. List all the products whose QtyOnHand is less than ReOrderLevel.
   Select Product_No, Description from Product_Master Where QtyOnHand < ReOrderLvl;

Exercise-5

Create a table Salesman_Master which include these fields, Insert records in the table then perform a given queries:

A. Create the table described below:

Table Name: Salesman_Master
Description: Used to store salesman information working for the company.

Column Name	Data Type	Size	Attributes
Salesman_No	Varchar2	6	Primary key/ First Letter Start with S
SalesmanName	Varchar2	20	Not null
City	Varchar2	20	
State	Varchar2	20	
SalAmt	Number	8,2	Not null , Cannot be 0
TgtToGet	Number	6,2	Not null , Cannot be 0
YtdSales	Number	6,2	Not null
Remarks	Varchar2	60	

Create Table:

```
Create table Salesman_Master
(Salesman_No varchar2 (6) Primary Key,
SalesmanName varchar2 (20) Not Null,
City varchar2 (20),
State varchar2 (20),
SalAmt number (8, 2) Not Null,
TgtToGet number (6,2) Not Null,
YTDSales number (6, 2) Not Null,
Remarks varchar2 (60),
Check (Salesman_No like 'S%'),
Check (SalAmt <> 0),
Check (TgtToGet <> 0));
```

Output:

Table Created.

B. Data for Salesman_Master table:

SALESMAN_NO	SALESMANNAME	CITY	STATE	SALAMT	TGTTOGET	YTDSALES	REMARKS
S00001	Sunita	Mumbai	Maharashtra	3000	100	50	Good
S00002	Sonu	Mumbai	Maharashtra	3000	200	100	Good
S00003	Sagar	Mumbai	Maharashtra	3000	200	100	Good
S00004	Sadip	Mumbai	Maharashtra	3500	200	150	Good

```
insert into Salesman_Master(Salesman_No,SalesmanName,City,State,SalAmt,TgtToGet,YTDSales,
        Remarks )values('S00001','Sunita','Mumbai','Maharashtra',3000,100,50,'Good');
```

Output:

1 row(s) inserted.

Exercise-6

Create a table Sales_Order which include given fields, Insert records in the table and use foreign key concept.

A. Create the table described below:

Table Name: Sales_Order
Description: Used to store client's orders.

Column Name	Data Type	Size	Default	Attributes
Order_No	Varchar2	6		Primary key/ First Letter Start with O
Client_No	Varchar2	6		Foreign Key references Client_No of Client_Master
OrderDate	Date			Not null
Salesman_No	Varchar2	6		Foreign Key references Salesman_No of Salesman_Master
DelyType	Char	1	F	Delivery: part(P)/full(F)
BillYN	Char	1		
DelyDate	Date			
OrderStatus	Varchar2	10		Values('In Process' , 'Fulfilled' ,' Backorder ', 'Cancelled')

Create Table:

```
Create table Sales_Order
(Order_No varchar2 (6) Primary key,
Client_No varchar2 (6) References Client_Master,
OrderDate date not null,
Salesman_No varchar2 (6) References Salesman_Master,
DelyType char (1) Default 'F',
BillYN char (1),
DelyDate date,
OrderStatus varchar2 (10),
Check (Order_No like 'O%'),
Check (DelyType In ('P','F')),
Check (Orderstatus In ('In Process','Fulfilled', 'Backorder', 'Cancelled')));
```

Output:

Table Created.

B. Data for Sales_Order table:

ORDER_NO	CLIENT_NO	ORDERDATE	SALESMAN_NO	DELYTYPE	BILLYN	DELYDATE	ORDERSTATUS
O19001	C00001	12-JUN-04	S00001	F	N	20-JUL-02	In Process
O19002	C00002	25-JUN-04	S00002	P	N	27-JUN-02	Cancelled
O19003	C00003	18-FEB-04	S00003	F	Y	27-FEB-02	Fulfilled
O19004	C00004	03-APR-04	S00001	F	Y	07-APR-02	Fulfilled
O19005	C00005	20-MAY-04	S00002	P	N	22-MAY-02	Cancelled
O19006	C00006	24-MAY-04	S00003	F	N	26-JUL-02	In Process

```
insert into Sales_Order
(Order_No,Client_No,OrderDate,Salesman_No,DelyType,BillYN,DelyDate,OrderStatus)
values('O19001','C00001','12-June-04','S00001','F','N','20-July-02','In Process');
```

Output:

```
1 row(s) inserted.
```

Exercise-7

Create a table Sales_Order_Details which include these fields use foreign key, Insert records in the table then perform a given queries:

A. Create the table described below:

Table Name: Sales_Order_Details
Description: Used to store client's orders with details of each product ordered.

Column Name	Data Type	Size	Attributes
Order_No	Varchar2	6	Foreign Key references Order_No of Sales_Order
Product_No	Varchar2	6	Foreign Key references Product_No of Product_Master
QtyOrdered	Number	8	
QtyDisp	Number	8	
ProductRate	Number	10,2	

Create Table:

```
Create table Sales_Order_Details
(Order_No varchar2 (6) References Sales_Order,
Product_No varchar2 (6) References Product_Master,
QtyOrdered number (8),
QtyDisp number (8),
ProductRate number (10, 2));
```

Output:

Table Created.

B. Data for Sales_Order_Details table:

ORDER_NO	PRODUCT_NO	QTYORDERED	QTYDISP	PRODUCTRATE
O19001	P00001	4	4	525
O19001	P00006	2	1	8400
O19002	P00001	2	1	5250
O19003	P00005	10	0	525
O19003	P00006	3	3	3150
O19004	P00001	3	1	5250
O19004	P00002	10	10	525
O19006	P00003	4	4	1050

```
insert into Sales_Order_Details
(Order_No,Product_No,QtyOrdered,QtyDisp,ProductRate)val
ues ('O19001','P00001',4,4,525);
```

Output:

1 row(s) inserted.

C. Exercise on retrieving records from a table

1. List all information from the Sales_Order table for orders place in the month of June.
   Select * from Sales_Order Where to_char (OrderDate, 'Mon') = 'Jun';

2. Count the total number of order.
   Select count(*) "Total Order" from Sales_Order;

3. List the order information for clientno ‘C00001’ and ‘C00002’.
   Select * from Sales_Order Where Client_No In ('C00001',
   'C00002');

4. List the order no and day on which client placed their order.
   Select Order_No, to_char (OrderDate, 'DAY') from
   Sales_Order;

5. List the month (in alphabets) and date when the orders must be delivered.
   Select to_char(DelyDate, 'MONTH'),DelyDate from Sales_Order order by to_char
   (DelyDate, 'MONTH');

6. List the OrderDate in the format ‘DD – Month – YY’ e.g. 12 – February -
   02 Select to_char (OrderDate,'DD – MONTH – YY') from Sales_Order;

7. List the date, 15 days after today
   date. Select sysdate + 15 from dual;

8. Print the description and total qty sold for each product.
   select description, sum(qtydisp) from product_master pm, sales_order_details sod where
   pm.product_no = sod.product_no group by description;

Exercise-8

Perform clause and special operators using Emp table.

A. Create the table described below:

Table Name: Emp
Description: Used to store employees details.

Column Name	Data Type	Size
EmpNo	Number	6
Ename	Varchar2	20
Job	Varchar2	15
MGR	Number	10
HireDate	Date	
Sal	Number	10
Comm	Number	10
DeptNo	Number	10

Create Table:

```
Create table Emp
(EmpNo number(6),
Ename varchar2(20),
Job varchar2(15),
MGR number(10),
HireDate Date,
Sal number(10),
Comm number(10),
DeptNo  number(10));
```

Output:

Table Created.

B. Data for Emp table:

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	Smith	Clerk	7902	17-DEC-80	800	-	20
7499	Allen	Salesman	7698	20-FEB-81	1600	300	30
7521	Ward	Salesman	7698	22-FEB-81	1250	500	30
7566	Jones	Manager	7839	02-APR-81	2975	-	20
7654	Martin	Salesman	7698	28-SEP-81	1250	1400	30
7698	Blake	Manager	7839	01-MAY-81	2850	-	30
7782	Clark	Manager	7839	09-JUN-81	2450	-	10
7788	Scott	Analyst	7566	19-APR-87	3000	-	20
7799	King	President	-	17-NOV-81	5000	-	10
7844	Turner	Salesman	7698	08-SEP-81	1500	0	30

insert into Emp(EmpNo,Ename,Job,MGR,HireDate,Sal,Comm,DeptNo) values
(7369,'Smith','Clerk',7902, '17-DEC-1980',800, '',20);

Output:

1 row(s) inserted.

C. Exercise to perform with select statement

1. Where

Select * From Emp Where Deptno>20;

Output:

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7499	Allen	Salesman	7698	20-FEB-81	1600	300	30
7521	Ward	Salesman	7698	22-FEB-81	1250	500	30
7654	Martin	Salesman	7698	28-SEP-81	1250	1400	30
7698	Blake	Manager	7839	01-MAY-81	2850	-	30
7844	Turner	Salesman	7698	08-SEP-81	1500	0	30
7900	James	Clerk	7698	03-DEC-81	950	-	30

2. Group By & Having Clause

Select Job, Avg(Sal) From Emp Group By Job;

Output:

JOB	AVG(SAL)
Clerk	1037.5
Manager	2758.33333333333333333333333333333333333
Analyst	3000
President	5000
Salesman	1400

Select Job, Avg (Sal) From Emp Group By Job Having Job='Analyst';

Output:

JOB	AVG(SAL)
Analyst	3000

3. Roll Up

Select EmpNo, Ename, Sum(sal) From Emp Group By Rollup(EmpNo,Ename);

Output:

EMPNO	ENAME	SUM(SAL)
7900	James	950
7900	-	950
7369	Smith	800
7369	-	800
7499	Allen	1600
7499	-	1600
7521	Ward	1250
7521	-	1250
7566	Jones	2975
7566	-	2975
More than 10 rows available. Increase rows selector to view more rows.		

4. Cube

Select EmpNo, Ename, Sum(sal) From Emp Group By Cube(EmpNo,Ename);

Output:

EMPNO	ENAME	SUM(SAL)
-	-	29025
-	Ford	3000
-	King	5000
-	Adams	1100
-	Allen	1600
-	Blake	2850
-	Clark	2450
-	James	950
-	Jones	2975
-	Scott	3000
More than 10 rows available. Increase rows selector to view more rows.		

5. Order By

Select Job From Emp Order By Job;

Output:

JOB
Analyst
Analyst
Clerk
Clerk
Clerk
Clerk
Manager
Manager
Manager
President

Select Ename From Emp Order By Ename Desc;

Output:

ENAME
Turner
Smith
Scott
Miller
King
Jones
James
Ford
Clark
Blake
More than 10 rows available. Increase rows selector to view more rows.

6. Distinct

Select Distinct Job From Emp;

Output:

JOB
Clerk
Manager
Analyst
President
Salesman

Special Operator

1. **In**

List all employee whose salary in 1500 to 2500

Select Ename, Sal From Emp Where Sal In (1500,2500);

Output:

ENAME	SAL
Turner	1500

2. **Any**

Select Empno, Sal From Emp Where Sal > Any (2000, 3000, 4000);

Output:

EMPNO	SAL
7566	2975
7698	2850
7782	2450
7788	3000
7799	5000
7902	3000

3. **All**

Select Empno, Sal From Emp Where Sal > All (2000, 3000, 4000);

Output:

EMPNO	SAL
7799	5000

4. **Between**

Select * From Emp Where Sal Between 1500 And 2500;

Output:

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7499	Allen	Salesman	7698	20-FEB-81	1600	300	30
7782	Clark	Manager	7839	09-JUN-81	2450	-	10
7844	Turner	Salesman	7698	08-SEP-81	1500	0	30

5. **Exists & Not Exists**

SELECT SalesmanName FROM Salesman_Master WHERE EXISTS (SELECT Order_No

FROM Sales_Order WHERE Sales_Order.Salesman_No = Salesman_Master.Salesman_No AND OrderStatus = 'Fulfilled');

Output:

SALESMANNAME
Sunita
Sagar

Not Exists

SELECT SalesmanName FROM Salesman_Master WHERE NOT EXISTS (SELECT Order_No FROM Sales_Order WHERE Sales_Order.Salesman_No = Salesman_Master.Salesman_No AND OrderStatus = 'Fulfilled');

Output:

SALESMANNAME
Sonu
Sadip

6. Like

List the name of all client having 'a' as the second letter in their names.
Select * From Emp Where Ename Like '_a%';

Output:

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7900	James	Clerk	7698	03-DEC-81	950	-	30

List the employee whose job Starts with 'M'.

Select EName, Job From Emp Where Job Like 'M%';

Output:

ENAME	JOB
Jones	Manager
Blake	Manager
Clark	Manager

Exercise-9

Perform following queries using all different kinds of Joins

[A] Inner Join [Equi Join]:

```
SELECT Salesman_Master.Salesman_No, Salesman_Master.SalesmanName,
Sales_Order.Order_No
FROM Salesman_Master
INNER JOIN Sales_Order
ON Salesman_Master.Salesman_No = Sales_Order.Salesman_No;
```

Output:

SALESMAN_NO	SALESMANNAME	ORDER_NO
S00001	Sunita	O19001
S00002	Sonu	O19002
S00003	Sagar	O19003
S00001	Sunita	O19004
S00002	Sonu	O19005
S00004	Sadip	O19006

[B] Outer Join:

1. Left Outer Join:

```
SELECT Salesman_Master.Salesman_No, Salesman_Master.SalesmanName, Sales_Order.Order_No
FROM Salesman_Master
LEFT OUTER JOIN Sales_Order
ON Salesman_Master.Salesman_No = Sales_Order.Salesman_No;
```

Output:

SALESMAN_NO	SALESMANNAME	ORDER_NO
S00001	Sunita	O19001
S00002	Sonu	O19002
S00003	Sagar	O19003
S00001	Sunita	O19004
S00002	Sonu	O19005
S00003	Sagar	O19006
S00004	Sadip	-

2. Right Outer Join:

```
SELECT Sales_Order.Order_No, Sales_Order.DelyType,
    Salesman_Master.SalesmanName
FROM Salesman_Master
RIGHT OUTER JOIN Sales_Order
ON Salesman_Master.Salesman_No = Sales_Order.Salesman_No;
```

Output:

ORDER_NO	DELYTYPE	SALESMANNAME
O19001	F	Sunita
O19002	P	Sonu
O19003	F	Sagar
O19004	F	Sunita
O19005	P	Sonu
O19006	F	Sadip

3. Full Outer Join:

```
SELECT Salesman_Master.SalesmanName, Sales_Order.Salesman_No FROM
Salesman_Master
FULL OUTER JOIN Sales_Order ON
Salesman_Master.Salesman_No=Sales_Order.Salesman_No
ORDER BY Salesman_Master.SalesmanName;
```

Output:

SALESMANNAME	SALESMAN_NO
Sadip	-
Sagar	S00003
Sagar	S00003
Sonu	S00002
Sonu	S00002
Sunita	S00001
Sunita	S00001

[C] Self Join:

```
SELECT a.Salesmanname, b.city, a.SalAmt
FROM Salesman_Master a, Salesman_Master b
WHERE a.SalAmt >  b.SalAmt;
```

Output:

SALESMANNAME	CITY	SALAMT
Sadip	Mumbai	3500
Sadip	Mumbai	3500
Sadip	Mumbai	3500

Exercise-10

Perform following query using the concepts of Subquery, Minus, Intersect, Union

1. **Subquery**

```
SELECT *    FROM Client
WHERE Client_no IN
(SELECT Client_no  FROM Client WHERE Baldue > 1500) ;
```

Output:

CLIENT_NO	NAME	CITY	PINCODE	STATE	BALDUE
C00001	Ivan Bayross	Mumbai	400054	Maharashtra	15000
C00003	Chhaya Bankar	Mumbai	400057	Maharashtra	5000
C00005	Hansel Colaco	Mumbai	400060	Maharashtra	20000

2. **Minus**

```
SELECT Salesman_No
FROM Salesman_Master
MINUS
SELECT Salesman_No
FROM Sales_order;
```

Output:

SALESMAN_NO
S00004

3. **Intersect**

```
SELECT Salesman_No
FROM Salesman_Master
INTERSECT
SELECT Salesman_No
FROM Sales_order;
```

Output:

SALESMAN_NO
S00001
S00002
S00003

4. **Union**

```
SELECT Salesman_No
FROM Salesman_Master
UNION
SELECT Salesman_No
```

FROM Sales_order;

Output:

SALESMAN_NO
S00001
S00002
S00003
S00004

Exercise-11

Perform following queries using different types function such as Numeric function, Character function, Date function, Aggregate function, General function.

• Numeric Function

A. Create the Number_1 described below:

Table Name: Number_1
Description: Used to Store Number_1

Column Name	Data Type	Size
No1	Number	5,2
No2	Number	10

Create Table:

```
Create table Number_1
(No1 number(5,2),
No2 number (10));
```

Output:

Table Created.

B. Data for Number table:

No1	No2
6.28	5
7.28	6
-8.28	3
-8.28	3
0	3

```
insert into Number_1(No1,No2)values(6.28,5);
```

Output:

1 row(s) inserted.

1. Abs

```
Select No1, Abs(No1) From Number_1;
```

Output:

NO1	ABS(NO1)
6.28	6.28
7.28	7.28
-8.28	8.28
-8.28	8.28
0	0

2. **Ceil**

Select No1, Ceil(No1) From Number_1;

Output:

NO1	CEIL(NO1)
6.28	7
7.28	8
-8.28	-8
-8.28	-8
0	0

3. **Exp**

Select No2, Exp(No2) From Number_1;

Output:

NO2	EXP(NO2)
5	148.413159102576603421115580040552279624
6	403.428793492735122608387180543388279609
3	20.085536923187667740928529654581717897 1
3	20.085536923187667740928529654581717897 1
3	20.085536923187667740928529654581717897 1

4. **Floor**

Select No1, Floor(No1) From Number_1;

Output:

NO1	FLOOR(NO1)
6.28	6
7.28	7
-8.28	-9
-8.28	-9
0	0

5. **Greatest**

Select Greatest (1, '3.925', '2.4') "Greatest" From Dual;

Output:

Greatest
3.925

```
Select Greatest('Harry', 'Harriot', 'Harold') "Greatest" From Dual;
```

Output:

Greatest
Harry

6. **Least**

```
Select Least('Harry','Harriot','Harold') "Least" From Dual;
```

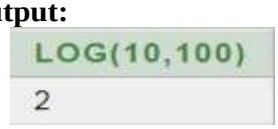
Output:

Least
Harold

7. **Log**

```
Select Log(10,100) From Dual;
```

Output:

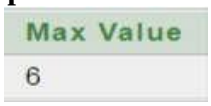


LOG(10,100)
2

8. **Max**

```
Select Max(No2) "Max Value" From Number_1;
```

Output:



Max Value
6

9. **Min**

```
Select Min(No2) "Min Value" From Number_1;
```

Output:

Min Value
3

10. **Remainder**

```
Select Remainder(15, 6)"Remainder" From Dual;
```

Output:

Remainder
3

11. **Round**

```
Select No1, Round(No1) From Number_1;
```

Output:

NO1	ROUND(NO1)
6.28	6
7.28	7
-8.28	-8
-8.28	-8
0	0

12. Sign

Select Sign(-5) From Dual;

Output:

SIGN(-1)
-1

13. Sqrt

Select No1,Abs(No1),Sqrt(Abs(No1)) From Number_1;

Output:

NO1	ABS(NO1)	SQRT(ABS(NO1))
6.28	6.28	2.505992817228333557699072963150573 1722
7.28	7.28	2.698147512646408293110061122299108994 49
-8.28	8.28	2.877498913987631724958462838497616352
-8.28	8.28	2.877498913987631724958462838497616352
0	0	0

- ## Character Function

A. Create the Number_1 described below:

Table Name: Stud_List
Description: Used to Store Name

Column Name	Data Type	Size
Name	Varchar2	15

Create Table:

```
Create table Stud_List
(Name varchar2(15));
```

Output:

Table Created.

B. Data for Number table:

Name
mahesh
dipesh
Ramesh
PRITESH
HARESH

```
insert into Stud_List(Name)values('mahesh');
```

Output:

1 row(s) inserted.

1. Chr

```
Select Chr(65) As "Character" From Dual;
```

Output:

Character
A

2. Concat

```
Select Concat('Welcome ','To Oracle') "Concat" From Dual;
```

Output:

Concat
Welcome To Oracle

3. Initcap

```
Select Name,Initcap(Name) From Stud_List;
```

Output:

NAME	INITCAP(NAME)
mahesh	Mahesh
dipesh	Dipesh
Ramesh	Ramesh
PRITESH	Pritesh
HARESH	Haresh

4. Lower

Select Name,Lower(Name) From Stud_List;

Output:

NAME	LOWER(NAME)
mahesh	mahesh
dipesh	dipesh
Ramesh	ramesh
PRITESH	pritesh
HARESH	haresh

5. Upper

Select Name,Upper(Name) From Stud_List;

Output:

NAME	UPPER(NAME)
mahesh	MAHESH
dipesh	DIPESH
Ramesh	RAMESH
PRITESH	PRITESH
HARESH	HARESH

6. Lpad

Select Lpad(Name,10,'*')"Lpad" From Stud_List;

Output:

Lpad
****mahesh
****dipesh
****Ramesh
***PRITESH
****HARESH

7. Rpad

Select Rpad(Name,10,'*')"Rpad" From Stud_List;

Output:

Rpad
mahesh****
dipesh****
Ramesh****
PRITESH***
HARESH****

8. Ltrim

Select Name,Ltrim (Name,'P') "Ltrim" From Stud_List;

Output:

NAME	Ltrim
mahesh	mahesh
dipesh	dipesh
Ramesh	Ramesh
PRITESH	RITESH
HARESH	HARESH

9. Rtrim

Select Name,Rtrim (Name,'h') "Rtrim" From Stud_List;

Output:

NAME	Rtrim
mahesh	mahes
dipesh	dipes
Ramesh	Rames
PRITESH	PRITESH
HARESH	HARESH

10. Trim

select TRIM(' ' FROM ' tech ')AS TrimmedString from dual;

Output:

TRIMMEDSTRING
tech

11. Replace

Select replace('PGDCA College','PGDCA','BCA')AS Replace from dual;

Output:

REPLACE
BCA College

12. Substr

Select Name,substr(Name, 3,4) "Substring" from Stud_List;

Output:

NAME	Substring
mahesh	hesh
dipesh	pesh
Ramesh	mesh
PRITESH	ITES
HARESH	RESH

- ## Date Function

1. Add_months

Select add_months (sysdate ,4) "Add_Months" from dual ;

Output:

Add_Months
02-NOV-21

2. Last_day

Select last_day (sysdate)"Last_day" from dual;

Output:

Last_day
31-JUL-21

3. Months_between

Select months_between ('02-april-21','02-july-21')"Months_between" from dual;

Output:

Months_between
-3

4. Next_day

Select next_day ('02-july-21','wednesday')"Next_day" from dual;

Output:

Next_day
07-JUL-21

5. Round(date)

Select ROUND(TO_DATE ('22-june-21'),'YEAR')"Round" from dual;

Output:

Round
01-JAN-22

6. Sysdate

Select SYSDATE From Dual;

Output:

SYSDATE
02-JUL-21

7. Systimestamp

Select systimestamp from dual;

Output:

SYSTIMESTAMP
02-JUL-21 11.26.00.356000 AM +05:30

8. Trunc(date)

Select TRUNC(TO_DATE('22-JULY-21'), 'MONTH')"Truncate" from dual;

Output:

Truncate
01-JUL-21

9. To_date

Select To_Date('09/07/2021', 'Dd/Mm/Yyyy') To_Date From Dual;

Output:

TO_DATE
09-JUL-21

10. To_char

Select To_Char(33459, '$99,999') To_Char From Dual;

Output:

TO_CHAR
$33,459

Select To_Char(sysdate, 'dd/mm/yyyy') To_Char From Dual;

Output:

TO_CHAR
02/07/2021

Select To_Char(sysdate, 'MON DDth, YYYY') To_Char From Dual;

Output:

TO_CHAR
JUL 02ND, 2021

- ## Aggregate Function

1. Sum

Select Sum(SalAmt)As "Total Salary" From Salesman_Master;

Output:

Total Salary
12500

2. Count

Select Count(Salesman_No)As "Total Salesman" From Salesman_Master;

Output:

Total Salesman
4

3. Avg

Select Avg(SalAmt)As "Average Salary" From Salesman_Master;

Output:

Average Salary
3125

4. Max

Select Max(SalAmt)As "Maximum Salary" From Salesman_Master;

Output:

Maximum Salary
3500

5. Min

Select Min(SalAmt)As "Minimum Salary" From Salesman_Master;

Output:

Minimum Salary
3000

- ## General Function

1. Coalesce

```
Select Coalesce(NULL, NULL, NULL, 'W3Schools.com', NULL, 'Example.com') As
"Coalesce" From Dual;
```

Output:

Coalesce
W3Schools.com

2. Case

```
SELECT Product_No, QtyOnHand,
CASE
   WHEN QtyOnHand > 100 THEN 'The quantity is greater than 100'
   WHEN QtyOnHand = 100 THEN 'The quantity is 100'
   ELSE 'The quantity is either greater than 100 nor less than 100'
END AS QuantityText
FROM Product_Master;
```

Output:

PRODUCT_NO	QTYONHAND	QUANTITYTEXT
P00001	200	The quantity is greater than 100
P00002	150	The quantity is greater than 100
P00003	100	The quantity is 100
P00004	100	The quantity is 100
P00005	150	The quantity is greater than 100
P00006	80	The quantity is either greater than 100 nor less than 100
P00007	100	The quantity is 100
P00008	70	The quantity is either greater than 100 nor less than 100
P00009	75	The quantity is either greater than 100 nor less than 100

3. Decode

```
SELECT SalesmanName,
DECODE(Salesman_No, 'S00001', 'Aman',
                    'S00002', 'Omkar',
                    'S00003', 'Raj',
                    'Aarav') result
FROM SALESMAN_MASTER;
```

Output:

SALESMANNAME	RESULT
Sunita	Aman
Sonu	Omkar
Sagar	Raj
Sadip	Aarav

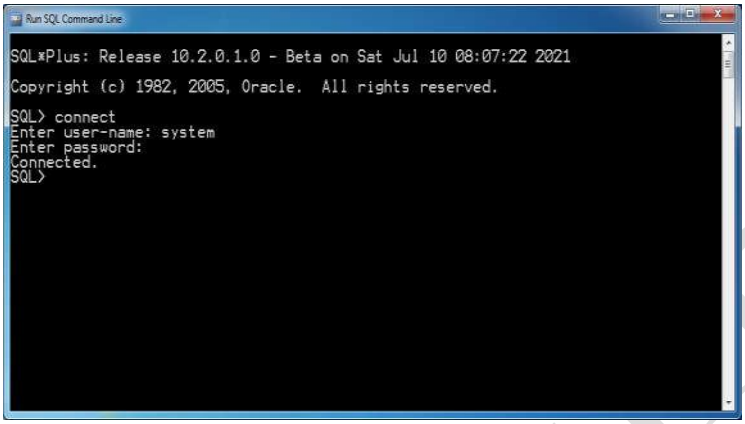
Exercise-12

Data Control and Transaction Control Command

A. Create user and grant user and revoke grant on object.

To perform this command open Run SQL Command Line window.
To open that window click on start-> Oracle Database 10g Express Edition-> Run SQL Command Line

First give connect and then give username.




```
SQL> connect
Enter user-name: system
Enter password: manager
Connected.
SQL> create user angel identified by angel123;

User created.

SQL> grant connect to angel;

Grant succeeded.

SQL> connect
Enter user-name: scott
Enter password: tiger
Connected.
SQL> create table test_data
  2  (no number(5),
  3  name varchar2(15));

Table created.

SQL>  insert  into  test_data(no,name)values(1,'aarva');

1 row created.

SQL>  insert  into  test_data(no,name)values(2,'prathvi');

1 row created.
```

```
SQL> select * from test_data;

        NO NAME
---------- ----------_ _ .
         1 aarva
         2 prathvi

SQL> grant select,insert on test_data to angel;

Grant succeeded.

SQL> commit;

Commit complete.

SQL> connect
Enter user-name: angel
Enter password: angel123
Connected.
SQL> select *  from scott.test_data;

        NO NAME
---------- ----------_ _ .
         1 aarva
         2 prathvi

SQL> insert into scott.test_data(no,name)values(3,'freya');

1 row created.

SQL> select * from scott.test_data;

        NO NAME
---------- ----------_ _ .
         1 aarva
         2 prathvi
         3 freya

SQL> delete from scott.test_data where no=3;
delete from scott.test_data where no=3
            *
ERROR at line 1:
ORA-01031: insufficient privileges
```

Note: We have not given delete grant to user angel so it will generate error insufficient privileges.

Note: we take back insert privilege from user angel

```
SQL> connect
Enter user-name: scott
Enter password: tiger
Connected.
SQL> revoke insert on test_data from angel;

Revoke succeeded.
```

```
SQL> connect
Enter user-name: angel
Enter password: angel123
Connected.
SQL> insert into scott.test_data (no,name) values (4,'mann');
insert into scott.test_data (no,name) values (4,'mann')
            *
ERROR at line 1:
ORA-01031: insufficient privileges
```

B. Create role and grant role to user.

```
SQL> connect
Enter user-name: system
Enter password: manager
Connected.
SQL> create role testing;

Role created.

SQL> grant select on stud to testing;

Grant succeeded.

SQL> grant select on client_master to testing;

Grant succeeded.

SQL> create user mmg identified by mmghodasara;

User created.

SQL> grant connect to mmg;

Grant succeeded.

SQL> grant testing to mmg;

Grant succeeded.

SQL> commit;

Commit complete.

SQL> connect
Enter user-name: mmg
Enter password: mmghodasara
Connected.
SQL> select * from system.stud;

ROLL_N NAME                 MARKS
------ -------------------- -----
S00001 Ishita                  80
S00002 Ruhi                    65

SQL> connect
Enter user-name: system
Enter password: manager
Connected.
SQL> revoke select on stud from testing;

Revoke succeeded.
```

C. Perform a Commit & rollback command on the table.

```
SQL> connect
Enter user-name: scott
Enter password: tiger
Connected.
SQL> create table stud_data
  2  (no number(5),
  3  name varchar2(15),
  4  marks number (5));

Table created.

SQL> insert into stud_data(no,name,marks)values(1,'sachi',75);
1 row created.

SQL> insert into stud_data(no,name,marks)values(2,'radha',80);
1 row created.

SQL> insert  into  stud_data(no,name,marks)values(3,'krishna',80);
1 row created.

SQL> commit;
Commit complete.

SQL> select * from stud_data;

NO    NAME          MARKS
------ - - ---------- ----------
1     sachi          75
2     radha          80
3     krishna        80


SQL> insert  into  stud_data(no,name,marks)values(4,'pakhi',90);
1 row created.

SQL> select * from stud_data;

NO    NAME          MARKS
------ - - ---------- ----------
1     sachi          85
2     radha          80
3     krishna        80
4     pakhi          90

SQL> rollback;
Rollback complete.

SQL> select * from stud_data;

NO    NAME          MARKS
------ - - ---------- ----------
1     sachi          85
2     radha          80
3     krishna        80
```

D. Perform a savepoint command on the table.

```
SQL> connect
Enter user-name: scott
Enter  password:  tiger
Connected.

SQL> select * from stud_data;

      RNO NAME          MARKS
---------- ---------- ----------
         1 sachi             85
         2 radha             80
         3 krishna           80

SQL> savepoint sp1;

Savepoint created.

SQL> delete from stud_data where rno=1;

1 row deleted.

SQL> select * from stud_data;

      RNO NAME          MARKS
---------- ---------- ----------
         2 radha             80
         3 krishna           80

SQL> savepoint sp2;

Savepoint created.

SQL> delete from stud_data where rno=2;

1 row deleted.

SQL> select * from stud_data;

      RNO NAME          MARKS
---------- ---------- ----------
         3 krishna           80

SQL> savepoint sp3;

Savepoint created.

SQL> delete from stud_data where rno=3;

1 row deleted.

SQL> select * from stud_data;

no rows selected
```

```
SQL> rollback to sp2;

Rollback complete.

SQL> select * from stud_data;

       RNO NAME           MARKS
---------- ---------- ----------
         2 radha              80
         3 krishna            80
```

UNIT : 3
Other ORACLE Database Objects Concurrency Control Using Lock
Exercise-1

Perform View.

A. View - Create, Insert, Update, Delete, Drop on Single Table:

create view vw_sales as select * from sales_order;

View created.

select * from vw_sales;

ORDER_NO	CLIENT_NO	ORDERDATE	SALESMAN_NO	DELYTYPE	BILLYN	DELYDATE	ORDERSTATUS
O19001	C00001	12-JUN-04	S00001	F	N	20-JUL-02	In Process
O19002	C00002	25-JUN-04	S00002	P	N	27-JUN-02	Cancelled
O19003	C00003	18-FEB-04	S00003	F	Y	27-FEB-02	Fulfilled
O19004	C00004	03-APR-04	S00001	F	Y	07-APR-02	Fulfilled
O19005	C00005	20-MAY-04	S00002	P	N	22-MAY-02	Cancelled
O19006	C00006	24-MAY-04	S00003	F	N	26-JUL-02	In Process

create view vw_sales1 as select order_no,client_no from sales_order;

View created.

select * from vw_sales1;

ORDER_NO	CLIENT_NO
O19001	C00001
O19002	C00002
O19003	C00003
O19004	C00004
O19005	C00005
O19006	C00006

insert into vw_sales values('O19007','C00001','25-DEC-10','S00001','F','N','3-JAN-2011', 'In Process');
1 row(s) inserted.

select * from vw_sales;

ORDER_NO	CLIENT_NO	ORDERDATE	SALESMAN_NO	DELYTYPE	BILLYN	DELYDATE	ORDERSTATUS
O19001	C00001	12-JUN-04	S00001	F	N	20-JUL-02	In Process
O19002	C00002	25-JUN-04	S00002	P	N	27-JUN-02	Cancelled
O19003	C00003	18-FEB-04	S00003	F	Y	27-FEB-02	Fulfilled
O19004	C00004	03-APR-04	S00001	F	Y	07-APR-02	Fulfilled
O19005	C00005	20-MAY-04	S00002	P	N	22-MAY-02	Cancelled
O19006	C00006	24-MAY-04	S00003	F	N	26-JUL-02	In Process
O19007	C00001	25-DEC-10	S00001	F	N	03-JAN-11	In Process

update vw_sales set orderstatus='Cancelled' where order_no='O19007';
1 row(s) updated.

select * from vw_sales;

ORDER_NO	CLIENT_NO	ORDERDATE	SALESMAN_NO	DELYTYPE	BILLYN	DELYDATE	ORDERSTATUS
O19001	C00001	12-JUN-04	S00001	F	N	20-JUL-02	In Process
O19002	C00002	25-JUN-04	S00002	P	N	27-JUN-02	Cancelled
O19003	C00003	18-FEB-04	S00003	F	Y	27-FEB-02	Fulfilled
O19004	C00004	03-APR-04	S00001	F	Y	07-APR-02	Fulfilled
O19005	C00005	20-MAY-04	S00002	P	N	22-MAY-02	Cancelled
O19006	C00006	24-MAY-04	S00003	F	N	26-JUL-02	In Process
O19007	C00001	25-DEC-10	S00001	F	N	03-JAN-11	Cancelled

delete from vw_sales where order_no='O19007';
1 row(s) deleted.

select * from vw_sales;

ORDER_NO	CLIENT_NO	ORDERDATE	SALESMAN_NO	DELYTYPE	BILLYN	DELYDATE	ORDERSTATUS
O19001	C00001	12-JUN-04	S00001	F	N	20-JUL-02	In Process
O19002	C00002	25-JUN-04	S00002	P	N	27-JUN-02	Cancelled
O19003	C00003	18-FEB-04	S00003	F	Y	27-FEB-02	Fulfilled
O19004	C00004	03-APR-04	S00001	F	Y	07-APR-02	Fulfilled
O19005	C00005	20-MAY-04	S00002	P	N	22-MAY-02	Cancelled
O19006	C00006	24-MAY-04	S00003	F	N	26-JUL-02	In Process

Drop view vw_sales;
View dropped.

B. View- Creating on Multiple Table:

Create View mvw_sales As
Select Salesman_Master.Salesman_No, Salesman_Master.Salesmanname, Sales_Order.Order_No
From Salesman_Master, Sales_Order Where Salesman_Master.Salesman_No =
Sales_Order.Salesman_No;

View created.

select * from mvw_sales;

SALESMAN_NO	SALESMANNAME	ORDER_NO
S00001	Sunita	O19001
S00002	Sonu	O19002
S00003	Sagar	O19003
S00001	Sunita	O19004
S00002	Sonu	O19005
S00003	Sagar	O19006

Exercise-2

Perform Sequence.

A. Create Sequence

```
CREATE SEQUENCE sequence_1
start with 1
increment by 1
minvalue 0
maxvalue 100
cycle;
```

B. To view sequence with table.

create a table named supplier with columns as id and name.

```
CREATE TABLE supplier
(
ID number(10),
NAME varchar2(20)
);
```

Now insert values into table

```
INSERT into supplier VALUES(sequence_1.nextval,'Mann');
INSERT into supplier VALUES(sequence_1.nextval,'Aarush');
```

where sequence_1.nextval will insert id's in id column in a sequence as defined in sequence_1.

To view records in table

```
SELECT * FROM supplier;
```

ID	NAME
1	Mann
2	Aarush

C. To view sequence with next value.

```
SELECT sequence_1.NEXTVAL FROM dual;
```

NEXTVAL
3

D. To view sequence with Current value.

```
SELECT sequence_1.CURRVAL FROM dual;
```

CURRVAL
3

E. Alter sequence

```
ALTER SEQUENCE sequence_1
INCREMENT BY 2
```

```
MAXVALUE 1000
CYCLE;
```

Sequence altered.

```
SELECT sequence_1.NEXTVAL FROM dual;
```

NEXTVAL
5

```
SELECT sequence_1.NEXTVAL FROM dual;
```

NEXTVAL
7

F. Drop sequence

```
DROP SEQUENCE sequence_1;
```

Sequence dropped.

Exercise-3

Perform Synonyms.

A. Create Synonyms

```
CREATE OR REPLACE SYNONYM synon1
FOR stud;
```

Synonym created.

Select * From synon1;

ROLL_NO	NAME	MARKS
S00001	Ishita	80
S00002	Ruhi	65

Drop Synonym synon1;

Synonym dropped.

Exercise-4

Perform Database Link

```
CREATE DATABASE LINK stud
CONNECT TO scott IDENTIFIED BY tiger
USING 'stud_list';
```

Output:

```
Database Link created.
```

Exercise-5

Perform Index.

A. Simple Index

CREATE INDEX emp_ename_i ON emp(ename);
Index created.

Select Empno,Job From Emp Where Ename='King';

EMPNO	JOB
7799	President

B. Composite Index:

CREATE INDEX emp_eno_job_i ON emp(empno,job);
Index created.

Select Empno,Ename,Job From Emp Where Job IN('Analyst') And Empno>7700;

EMPNO	ENAME	JOB
7788	Scott	Analyst
7902	Ford	Analyst

EMPNO	ENAME	JOB
7788	Scott	Analyst
7876	Adams	Clerk
7900	James	Clerk
7902	Ford	Analyst
7934	Miller	Clerk

C. Unique Index:

Create Unique Index Empno On Emp(Empno);
Index created.

Select * From Emp Where Empno=7566;

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7566	Jones	Manager	7839	02-APR-81	2975	-	20

D. Function base Index:

CREATE INDEX client_name_fbi ON client(UPPER(name));
Index created.

Select * From client Where Upper(name)='MAMTA MUZUMDAR';

CLIENT_NO	NAME	CITY	PINCODE	STATE	BALDUE
C00002	Mamta Muzumdar	Madras	780001	TamilNadu	0

Exercise-6

Perform cluster.

A. Create cluster

```
Create Cluster C1(D Number(2));

Cluster created.
```

B. Now create index on cluster…

```
create index ci1 on cluster c1;

Index created.
```

C. Create dept table with cluster and insert records

```
create table dept
(dno number(2),
dname varchar2(20))
cluster c1(dno);

Table  created.
```

Insert records into table…

```
insert into dept values(10,'Acct');

1 row created.

insert into dept values(20,'Research');

1 row created.
```

D. Create emp_1 table with cluster and insert records

```
create table emp_1
(empno number(4),
ename varchar2(20),
dno number(2))
cluster c1(dno);

Table created.
```

Insert records into table…

```
insert into emp_1 values(1,'A',10);
1 row created.

insert into emp_1 values(2,'B',20);

1 row created.
```

E. select rowid,dno,dname from dept;

ROWID	DNO	DNAME
AAAE4rAABAAAKD6AAA	10	ACCT
AAAE4rAABAAAKD7AAA	20	RESEARCH

F. select rowid,empno,ename,dno from emp_1;

ROWID	EMPNO	ENAME	DNO
AAAE4rAABAAAKD6AAA	1	A	10
AAAE4rAABAAAKD7AAA	2	B	20

UNIT : 4
Introduction to PL/SQL, Advanced PL/SQL
Exercise-1

Simple PL/SQL Block structure.

Write A Program To Print "Welcome To PL/SQL Block".

```
Begin
        --this is first PL/SQL Block form
        dbms_output.put_line ('Welcome to PL/SQL Block');
End;
```

Output:

```
Welcome to PL/SQL Block
```

Exercise-2

Variable Declaration

Write A Program To Count Square.

```
Declare
        x number (5);
        ans number (5);
Begin
        x:= :x;
        ans:= x*x;
        dbms_output.put_line ('Square is = '||ans);
End;
```

Output:

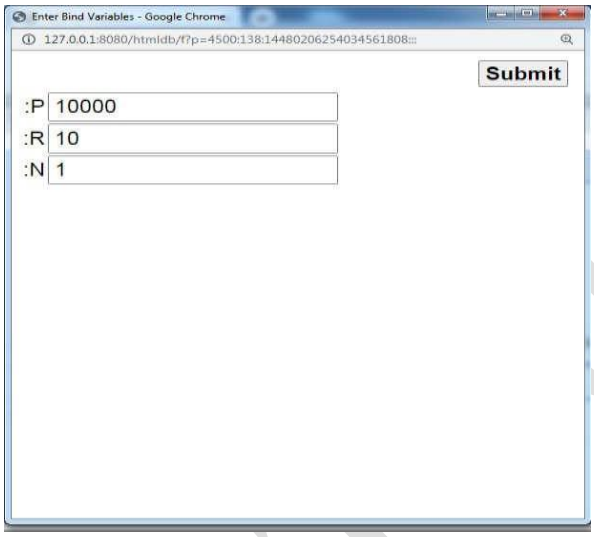

```
Square is = 100
```

Exercise-3

Write A Program To Find Simple Interest.

```
Declare
        p number:= :p;
        r number:= :r;
        n number:= :n;
        ans number;
Begin
        ans:= (p*r*n)/100;
        dbms_output.put_line('Simple Interest is = '||ans);
End;
```

Output:




```
Simple Interest is = 1000
```

Exercise-4

Write a program for global and local variable.

```
DECLARE
        -- Global variables
          num1 number := 95;
          num2 number := 85;
BEGIN
          dbms_output.put_line('Outer Variable num1: ' || num1);
          dbms_output.put_line('Outer Variable num2: ' || num2);
        DECLARE
                -- Local variables
                num1 number := 195;
                num2 number := 185;
        BEGIN
                dbms_output.put_line('Inner Variable num1: ' || num1);
                dbms_output.put_line('Inner Variable num2: ' || num2);
        END;
END;
```

Output:

```
Outer Variable num1: 95
Outer Variable num2: 85
Inner Variable num1: 195
Inner Variable num2: 185
```

Exercise-5

Write a program to give a declare constant variable.

```
DECLARE
  -- constant declaration
        pi constant number := 3.141592654;
   -- other declarations
        radius number(5,2);
        dia number(5,2);
        circumference number(7, 2);
        area number (10, 2);
BEGIN
        radius :=  9.5;
        dia := radius * 2;
        circumference := 2.0 * pi * radius;
        area := pi * radius * radius;

        dbms_output.put_line('Radius: ' || radius);
        dbms_output.put_line('Diameter: ' || dia);
        dbms_output.put_line('Circumference: ' || circumference);
        dbms_output.put_line('Area: ' || area);
END;
```

Output:

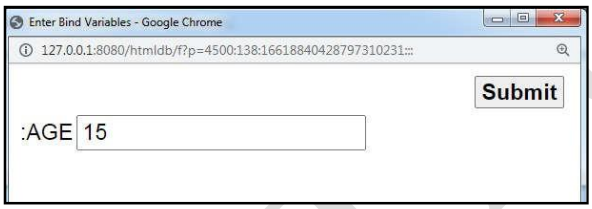
```
Radius: 9.5
Diameter: 19
Circumference: 59.69
Area: 283.53
```

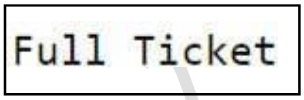
Exercise-6

Write a program to check ticket age wise with the use of IF-THEN-ELSIF-ELSE

```
Declare
        age number:= :age;

Begin
        if age<=4 then
                dbms_output.put_line ('No Ticket');

        elsif age between 5 and 11 then
                dbms_output.put_line  ('Half  Ticket');

        elsif age between 12 and 59 then
                dbms_output.put_line  ('Full  Ticket');

        else
                dbms_output.put_line ('Senior citizen Discount');

        end if;
End;
```

Output:



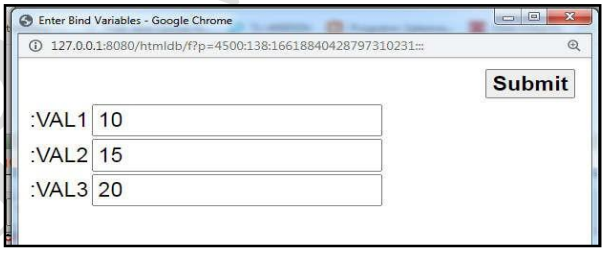


Exercise-7

Write a program to input three no and find out maximum and minimum from it.

```
Declare
        val1 number(5);
        val2 number(5);
        val3 number(5);
        ma number(5);
        mi number(5);
Begin
        val1:=:val1;
        val2:=:val2;
        val3:=:val3;

--for maximum value
        if val1>val2 and val1>val3 then
                ma:=val1;
        elsif val2>val3 and val2>val1 then
                ma:=val2;
        else
                ma:=val3;
        end if;

--for minimum value
        if val1<val2 and val1<val3 then
                mi:=val1;
        elsif val2<val3 and val2<val1 then
                mi:=val2;
        else
                mi:=val3;
        end if;

dbms_output.put_line ('Maximum=' ||ma);
dbms_output.put_line ('Minimum=' ||mi);

End;
```

Output:



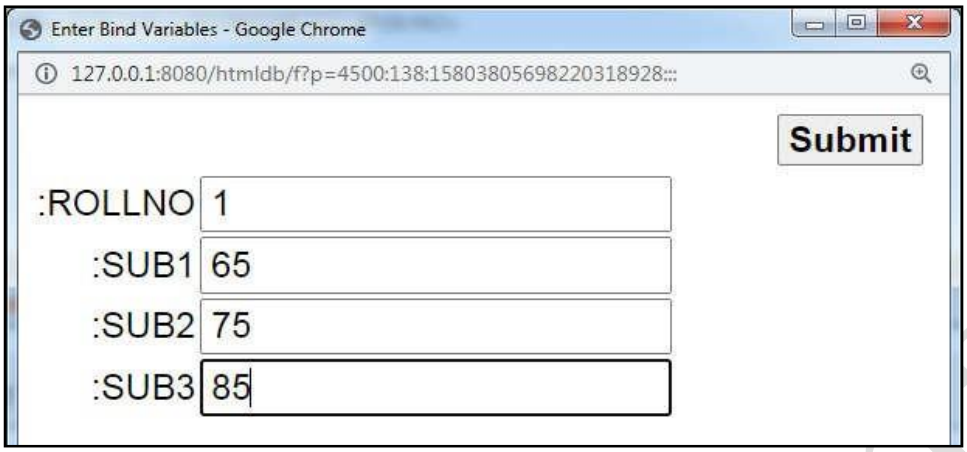

```
Maximum=20
Minimum=10
```

Exercise-8

Write a program to input rollno and three subject marks. find out total, percentage, result and grade for the student from the entered data.

```
Declare
        rollno number(5);
        sub1 number(10);
        sub2 number(10);
        sub3 number(10);
        total number(10);
        per number(5,2);
        result char(4);
        grade char(25);

Begin
        rollno:=:rollno;
        sub1:=:sub1;
        sub2:=:sub2;
        sub3:=:sub3;
        total:= sub1 + sub2 + sub3;
        per:= total/3;

        if sub1>=40 and sub2>=40 and sub3>=40 then
                result:= 'Pass';

                if per >= 70.0 then
                        grade:='Distinction';

                elsif per >=60.0 then
                        grade:='First Class';

                elsif per >=50.0 then
                        grade:='Second Class';

                elsif per >=40.0 then
                        grade:='Pass Class';

                end if;

        else
                result:= 'Fail';
                grade:= 'No Grade';
        end if;

dbms_output.put_line ('Roll No.: '|| rollno);
dbms_output.put_line ('Subject1: '|| sub1);
dbms_output.put_line ('Subject2: '|| sub2);
dbms_output.put_line ('Subject3: '|| sub3);
dbms_output.put_line ('Total: '|| total);
dbms_output.put_line ('Percentage: '|| per);
dbms_output.put_line ('Result: '|| result);
dbms_output.put_line ('Grade: '|| grade);

End;
```

Output:



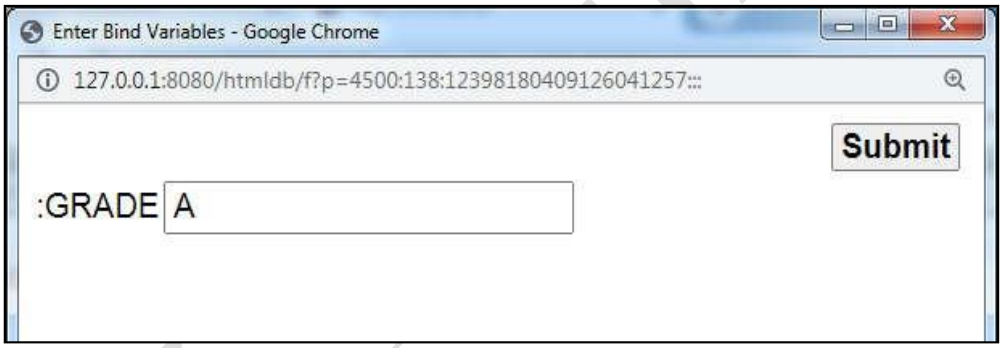

```
Roll No.: 1
Subject1: 65
Subject2: 75
Subject3: 85
Total: 225
Percentage: 75
Result: Pass
Grade: Distinction
```

Exercise-9

Write a program for the use of PL/SQL Case Statement

```
DECLARE
        grade char(1):=:grade;
BEGIN
        CASE grade
                when 'A' then
                        dbms_output.put_line('Excellent');
                when 'B' then
                        dbms_output.put_line('Very good');
                when 'C' then
                        dbms_output.put_line('Good');
                when 'D' then
                        dbms_output.put_line('Average');
                when 'F' then
                        dbms_output.put_line('Passed with Grace');
                else dbms_output.put_line('Failed');
        END CASE;
END;
```

Output:




```
Excellent
```

Exercise-10

Looping

Write a program to print first 10 numbers using basic loop.. exit loop.

```
Declare
        i number := 1;
Begin
        Loop
                Exit When i>10;
                Dbms_Output.Put_Line(i);
        i := i+1;
        End Loop;
End;
```

Output:

```
1
2
3
4
5
6
7
8
9
10
```

Exercise-11

Write a program to print first 10 numbers using while loop.

```
Declare
        i integer := 1;
Begin
        while i <= 10
        loop
        dbms_output.put_line(i);
        i := i+1;
        end loop;
End;
```

Output:

```
1
2
3
4
5
6
7
8
9
10
```

Exercise-12

Write a program to print first 10 numbers using for loop.

```
Declare
        Var1 Number;
Begin
        Var1:=10;
        For Var1 In 1..10
        Loop
        Dbms_Output.Put_Line (Var1);
        End Loop;
End;
```

Output:

```
1
2
3
4
5
6
7
8
9
10
```

Exercise-13

Write a program to print first 1 to 10 numbers in reverse using for loop.

```
Declare
        x number (5);
Begin

        For x in reverse 1..10
        Loop
                dbms_output.put_line(x);
        End loop;
End;
```

Output:

```
10
9
8
7
6
5
4
3
2
1
```

Exercise-14

Write a program to print first 10 prime number using while loop.

```
Declare
        c number (4);
        cntr number (4);
        a number (4);
        flag number (2);
Begin
        c:=0;
        cntr:=3;
        While c<10
        loop
                flag:=0;
                For i in 2.. (cntr-1)
                        loop
                        a:=cntr/i;
                        if (a*i) = cntr then
                                Flag:=1;
                        end if;
        End loop;
        if flag = 0 then
                dbms_output.put_line (cntr);
                C:=c+1;
        end if;
        cntr:=cntr+1;
        End loop;
End;
```

Output:

```
3
5
7
11
13
17
19
23
29
31
```

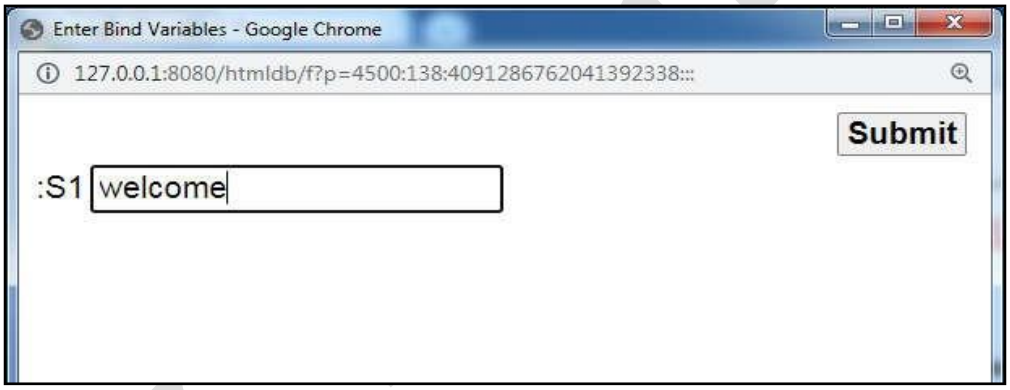
Exercise-15

String reverse

Write a program to accept a string from user and display reverse of it

```
Declare
        s1 varchar2 (20);
        s2 varchar2 (20);
        x number (20);

Begin

        s1:=:s1;
        x:=length (s1);
        For i in reverse 1..x
        loop
                s2:=s2 || substr (s1, i, 1);
        End loop;
        dbms_output.put_line ('Reverse string is ' || s2);
End;
```

Output:




```
Reverse string is emoclew
```

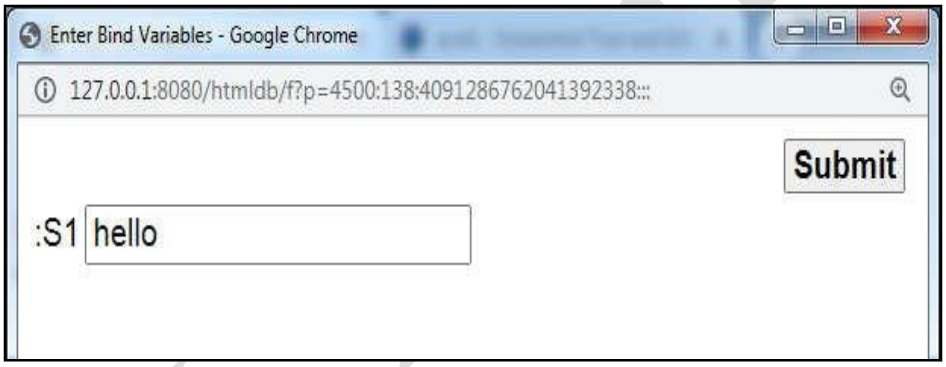
Exercise-16

Triangle

Write a program to input a string and display its Pyramid.

```
Declare
        s1 varchar2 (10);
        s2 varchar2 (10);
        x number (10);

Begin

        s1:=:s1;
        x:=length (s1);
        for i in 1..x
        loop
                s2:= substr (s1, 1, i);
                dbms_output.put_line (s2);
        end loop;
End;
```

Output:




```
h
he
hel
hell
hello
```

Exercise-17

Write a program to print number pyramid

```
Declare

Begin

        For i in 1..5 loop
        For j in 1..i loop
                dbms_output.put (" || j || "); -- Double single quote
        End loop;
        dbms_output.put_line (' ');
        End loop;
        dbms_output.put_line (' ');
End;
```

Output:

```
1
12
123
1234
12345
```

Exercise-18

Write a program for the use of goto statement.

```
DECLARE
        a number(2) := 10;
BEGIN
        <<loopstart>>
        -- while loop execution
        WHILE a < 20 LOOP
                dbms_output.put_line ('value of a: ' || a);
                a := a + 1;
                IF a = 15 THEN
                        a := a + 1;
                GOTO loopstart;
                END IF;
        END LOOP;
END;
```

Output:

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

Exercise-19

Looping to insert record into specified table

Write a program to insert 10 row into square table.

Create table square

```
create table square
(no number(5),
s number(5));
```

Table created.

Program

```
Declare
        no number (5):=1;
        s number (5):=0;

Begin

        While no<=10
        Loop
                s:=no*no;
                Insert into square values (no, s);
                No:=no+1;
        End loop;
End;
```

Output:

Select * from square;

NO	S
1	1
2	4
3	9
4	16
5	25
6	36
7	49
8	64
9	81
10	100

Exercise-20

Looping to update specified table.

Write a program to update row of square table

```
Declare
  n number (5):=1;
Begin
  While n<=10 Loop

  Update square set s = no*2 where no = n;
          n:=n+1;
  End loop;
End;
```

Output:

Select * from square;

NO	S
1	2
2	4
3	6
4	8
5	10
6	12
7	14
8	16
9	18
10	20

Exercise-21

--Looping to delete record from specified table

Write a program to delete row from square table

```
Declare
        n number(5):=10;

Begin
        n:=:n;
        Delete from square where no =n;
End;
```

Output:

Select * from square;

NO	S
1	2
2	4
4	8
5	10
6	12
7	14
8	16
9	18
10	20

Exercise-22

%TYPE

Write a program to use of %type attribute

Create table....

```
create table emp1
(emp_no number(5),
name varchar2(10),
salary number(5),
dept_no number(5),
dept varchar2(20));
```

Now Insert records into the table ....

```
insert into emp1 values(1,'krishna',5000,10,'sales');
```

EMP_NO	NAME	SALARY	DEPT_NO	DEPT
1	krishna	5000	10	sales
2	radha	5500	20	manager
3	mann	4000	10	account

PL/SQL block

```
Declare
        eno emp1.emp_no %TYPE;
        ename emp1.name %TYPE;
Begin
        eno:= '5';
        ename:= 'lavya';

        dbms_output.put_line('Employee No: ' || eno);
        dbms_output.put_line('Employee name: ' || ename);

End;
```

Output:

```
Employee No: 5
Employee name: lavya
```

Exercise-23

%ROWTYPE

Write a program to fetch records from table using %rowtype.

```
Declare
        -- declare variables
        -- declare record variable that represents a row fetched from the emp1 table
        emp_rec emp1%ROWTYPE; -- declare variable with %ROWTYPE attribute
Begin
        SELECT * INTO emp_rec FROM emp1 WHERE emp_no = 1; -- retrieve record
        dbms_output.put_line('Employee name: ' || emp_rec.name );
        dbms_output.put_line('Employee salary: ' || emp_rec.salary); -- display
End;
```

Output:

```
Employee name: krishna
Employee salary: 5000
```

Exercise-24

Implicit Cursor [%found]

Write a program to accept empno. If the employee is found then update the salary of employee by 15% and display a message base on the record.

```
Declare
        Empno emp1.emp_no %type;
Begin
        Update emp1 set salary =salary+ salary * 0.15 where emp_no = :empno;
        If sql%found then
                dbms_output.put_line ('Record is updated');

        Else
                dbms_output.put_line ('Record does not exist');

        End if;
End;
```

Output:

Empno 1 found in table so below output comes…

```
Record is updated
```

Now view record is updated in table we use below query…

Select * from emp1;

EMP_NO	NAME	SALARY	DEPT_NO	DEPT
1	krishna	5750	10	sales
2	radha	5500	20	manager
3	mann	4000	10	account

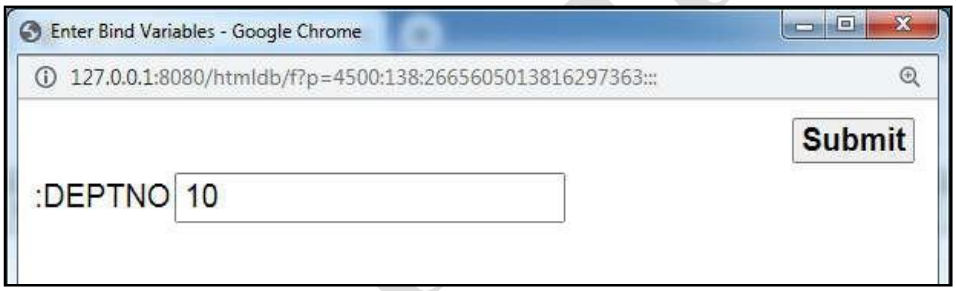
Exercise-25

Implicit Cursor [%rowcount]

Write a program to accept department number of employee and update the salary of that employee to increment by 1000. Display appropriate message and display how many rows are updated. (Using %rowtype attribute)

```
Declare
        rows_update char (3);
        deptno number (10);
Begin
        Update emp1 set salary = salary+1000 where dept_no=:deptno;
        rows_update:= to_char (sql%rowcount);

        If sql%rowcount >0 then
                dbms_output.put_line (rows_update || 'Rows are updated');
        Else
                dbms_output.put_line ('No employee in deptartment no' || deptno);
        End if;
End;
```

Output:




```
2  Rows are updated

1 row(s) updated.
```

Now view in table

Select * from emp1;

EMP_NO	NAME	SALARY	DEPT_NO	DEPT
1	krishna	6750	10	sales
2	radha	5500	20	manager
3	mann	5000	10	account

Exercise-26

Explicit Cursor [%isopen]

Write a program to find that explicit cursor is open or not if cursor is open then display appropriate message.

```
Declare
        Cursor cur_emp is select name from emp1 where dept_no=10;

Begin
        Open cur_emp;

        If cur_emp %isopen then
                dbms_output.put_line ('The cursor is opened');

        Else
                dbms_output.put_line ('The cursor is closed');
        End if;
End;
```

Output:

```
The cursor is opened
```

Exercise-27

Explicit Cursor [%notfound]

Write a program to display name and salary of emp1 table. (using %rowtype)

```
Declare
        xemp emp1 %rowtype;
        Cursor c_emp1 is select * from emp1;
Begin
        Open c_emp1;
        Loop
                Fetch c_emp1 into xemp;
                Exit when c_emp1 %notfound;
                dbms_output.put_line ('Salary of ' || xemp.name || ' is ' || xemp.salary);
         End loop;
        Close c_emp1;
End;
```

Output:

```
Salary of krishna is 6750
Salary of radha is 5500
Salary of mann is 5000
```

Exercise-28

Explicit cursor [%found]

Write a program to update the salary of employee and insert the record in bonus table (using explicit cursor %found).

We need 2 tables emp1 and bonus. emp1 table we created in the above exercise so create bonus table.

```
create table bonus
(emp_no number(10),
bonus number(10),
bonus_date date);
```

--PL/SQL Block

```
Declare
        Cursor cur_emp is Select emp_no,salary from emp1 where dept='sales';
        e_no emp1.emp_no %type;
        e_sal emp1.salary %type;

Begin
        Open cur_emp;

        Loop
                Fetch cur_emp into e_no, e_sal;

                If cur_emp % found then
                        Update emp1 set salary = salary + 2000 where emp_no = e_no;
                        Insert into bonus values (e_no, e_sal+2000, sysdate);
                Else
                        Exit;
                End if;
        End loop;

        Close cur_emp;
End;
```

Output:

```
1 row(s) updated.
```

Select * from emp1;

EMP_NO	NAME	SALARY	DEPT_NO	DEPT
1	krishna	8750	10	sales
2	radha	5500	20	manager
3	mann	5000	10	account

Select * from bonus;

EMP_NO	BONUS	BONUS_DATE
1	8750	27-SEP-21

Exercise-29

Explicit cursor [%notfound]

Write a program for the use of parameterized cursor. (In this program insert record in the branch_master table that is exists in the branch table. If the record does not exist then display message.

Table : branch

```
Create table branch
(branch_no varchar2(10),
name varchar2(10));
```

insert records into branch table

BRANCH_NO	NAME
1	BCA
2	MCA
3	BCOM

```
insert into branch values('1','BCA');
```

Table : branch_master

```
Create table branch_master
(branch_no varchar2(10),
name varchar2(10));
```

insert record into branch_master table

BRANCH_NO	NAME
1	BCA

```
insert into branch_master values('1','BCA');
```

--PL/SQL Block

```
Declare
        Cursor cur_branch is select * from branch;
        Cursor cur_branch_mstr(str_branch_name varchar2) is select branch_no from branch_master
        where name = str_branch_name;
        str_branch_no branch.branch_no %type;
        str_branch_name branch.name %type;
        mast_view varchar2 (10);
Begin
        Open cur_branch;
        Loop
        Fetch cur_branch into str_branch_no,str_branch_name;
        Exit when cur_branch %notfound;

        Open cur_branch_mstr(str_branch_name);
        Fetch cur_branch_mstr into mast_view;

        If cur_branch_mstr %found then
```

```
                dbms_output.put_line('Branch: ' || str_branch_name || ' exist');
        Else
                dbms_output.put_line('Branch: ' || str_branch_name || ' '||'does not exist.');
                dbms_output.put_line('Inserting new branch in the branch_master table');
                Insert into branch_master values(str_branch_no,str_branch_name);
        End if;

        Close cur_branch_mstr;
        End loop;
        Close cur_branch;
        Commit;
End;
```

Output:

```
Branch: BCA exist
Branch: MCA does not exist.
Inserting new branch in the branch_master table
Branch: BCOM does not exist.
Inserting new branch in the branch_master table

1 row(s) inserted.
```

Select * from branch_master;

BRANCH_NO	NAME
1	BCA
2	MCA
3	BCOM

Exercise-30

--Cursor for loops

/*Write a program to give a bonus of rs. 1000 to employee who are in the sales department and store that record in the bonus table using cursor for loops.*/

```
Declare
        Cursor cur_emp is select emp_no,salary from emp1 where dept='sales';
Begin
        For emp_rec in cur_emp
        Loop
                Update emp1 set salary=salary + 1000 where emp_no=emp_rec.emp_no;
                Insert into bonus values (emp_rec.emp_no, emp_rec.salary + 1000, sysdate);
        End loop;
End;
```

Output:

```
1 row(s) updated.
```

Select * from emp1;

EMP_NO	NAME	SALARY	DEPT_NO	DEPT
1	krishna	9750	10	sales
2	radha	5500	20	manager
3	mann	5000	10	account

Select * from bonus;

EMP_NO	BONUS	BONUS_DATE
1	8750	27-SEP-21
1	9750	27-SEP-21

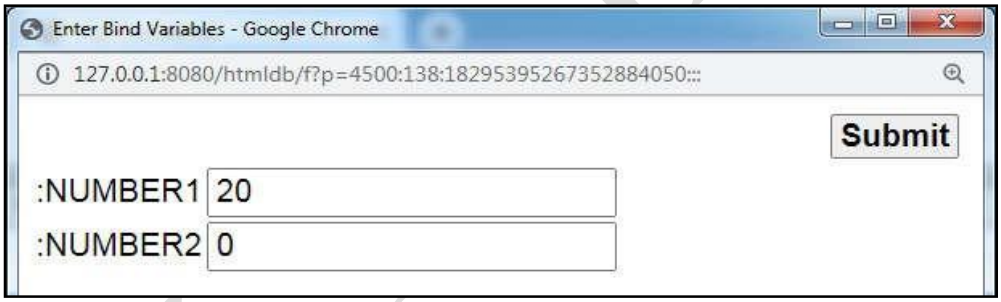
Exercise-31

Exception Handling

/*Write a program to enter two numbers and divide the no1 by no2. If the user enters the zero value of no2 then display appropriate error message using the exception handler. */

```
Declare
        no1 number(10);
        no2 number(10);
        ans number(10);

Begin
        no1:=:number1;
        no2:=:number2;
        ans:= no1/no2;
        dbms_output.put_line ('Division is = '|| ans);
        Exception

        When zero_divide then
                dbms_output.put_line ('you have entered no2 as zero');
                dbms_output.put_line ('Please entered another value');
End;
```

Output:




```
You have entered no2 as zero
Please enter another value
```

Exercise-32

Procedure (with in parameter)

/*Write a PL/SQL statement to create procedure called p1 which accept a number & print multiply by 2 on the screen*/

--Procedure

```
Create or replace procedure p1(a in number)is
        ans number;
Begin
        ans:= a*2;
        dbms_output.put_line ('The Answer is:'|| ans);
End;
```

```
Procedure created.
```

--PL/SQL Block

```
Begin
p1(5);
End;
```

Output:

```
The Answer is:10
```

Exercise-33

Procedure (with in out parameter)

/*Write a PL/SQL statement to create procedure which accept no from the user and multiply by 3 on the screen. (Use in out parameter)*/

--Procedure

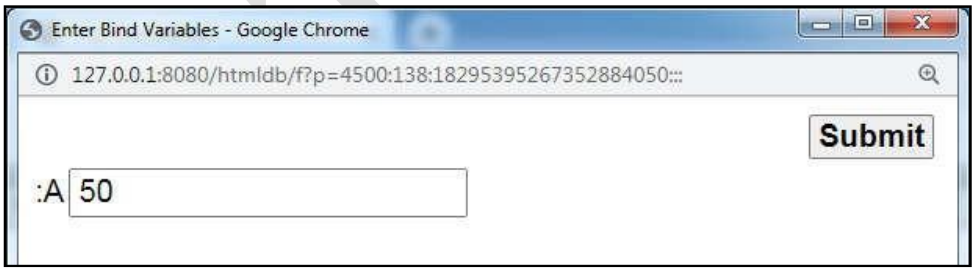
```
Create or replace procedure p2(a in out number)is
        ans number;
Begin
        ans:= a*2;
        dbms_output.put_line ('The Answer is:'|| ans);
End;
```

```
Procedure created.
```

--PL/SQL Block

```
Declare
        a number;
Begin
        a:=:a;
        dbms_output.put_line ('The Real no is: '||a);
        p2(a);
End;
```

Output:




```
The Real no is: 50
The Answer after multiply is:100
```

Exercise-34

Procedure (with out parameter)

/*Write a PL/SQL statement to create procedure which accept no from the user and multiply by 3 on the screen. (Use out parameter)*/

--Procedure

```
Create or replace procedure p3(a out number) is
        b number;
Begin
        b:=4;
        a:=b*b;
End;
```

```
Procedure created.
```

--PL/SQL Block

```
Declare
        a number;
Begin
        p3(a);
        dbms_output.put_line ('The answer is '||a);
End;
```

Output:

```
The answer is 16
```

Exercise-35

Procedure to insert value in table.

--Create a procedure which insert value in temp table.

--Create table temp:

```
create table temp
(no number(10),
name varchar2(20));
```

```
Table created.
```

--Procedure

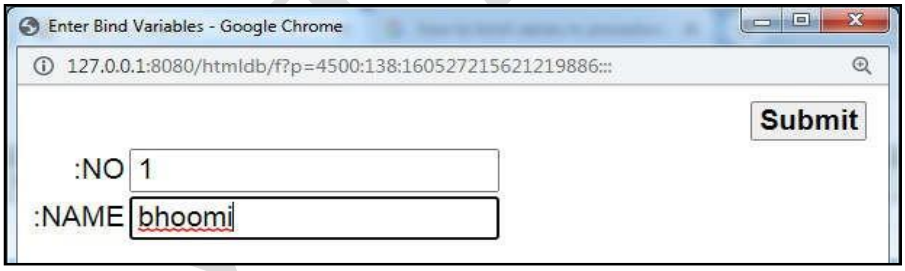
```
Create or replace procedure in_record(num temp.no %type, nm temp.name %type) As
Begin
        Insert into temp values(num, nm);
End;
```

```
Procedure created.
```

--PL/SQL block

```
Begin
        in_record(:no,:name);
End;
```

Output:



Run the PL/SQL block again and insert more records given below…

NO	NAME
1	bhoomi
2	dimple
3	neha

```
select * from temp;
```

Exercise-36

Procedure to update value in table.

--Create a procedure which update value in temp table.

--Procedure

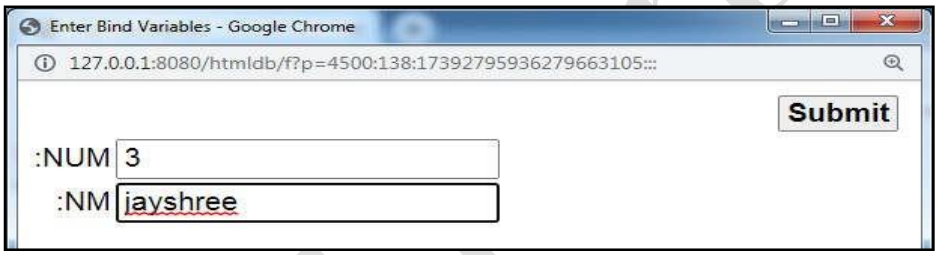
```
Create or replace procedure up_record(num temp.no %type, nm temp.name %type) As
Begin
        Update temp set name = nm where no = num;
End;
```

```
Procedure created.
```

--PL/SQL block

```
Begin
        up_record (:num,:nm);
End;
```

Output:



select * from temp;

NO	NAME
1	bhoomi
2	dimple
3	jayshree

Exercise-37

Procedure to delete value in table

--Create a procedure which delete value from temp table

--Procedure

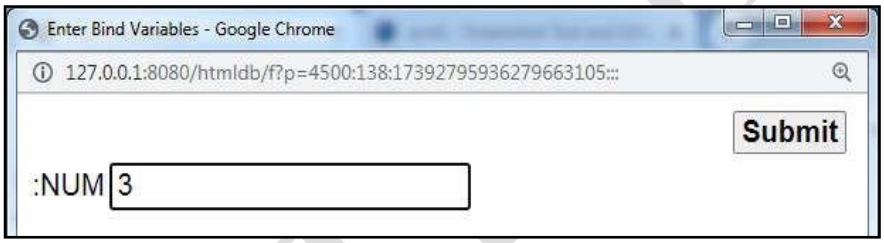
```
Create or replace procedure del_record (num temp.no %type)As
Begin
        Delete from temp where no = num;
End;
```

```
Procedure created.
```

--PL/SQL block

```
Begin
        del_record (:num);
End;
```

Output:



select * from temp;

NO	NAME
1	bhoomi
2	dimple

Exercise-38

Function

-- Create a function for addition of two values

--Function

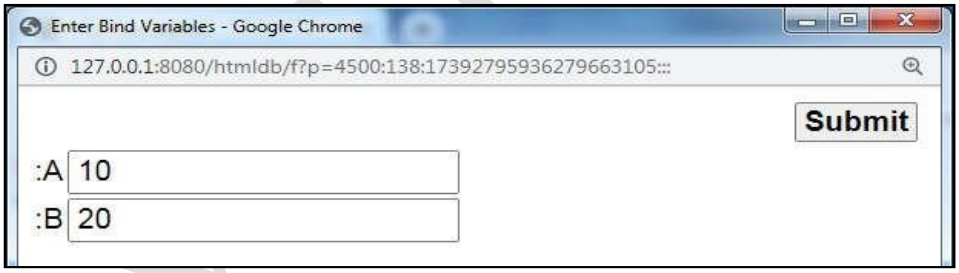
```
Create or replace function fun_add(a in out number, b in out number)
        Return number is c number;
Begin
        c:=a+b;
        Return c;
End;
```

```
Function created.
```

--PL/SQL block

```
Declare
        a   number:=:a;
        b   number:=:b;
        ans number(3);
Begin
        ans:=func_add(a, b);
        dbms_output.put_line ('The addition is:-' || ans);
End;
```

Output:




```
The addition is:-30
```

Exercise-39

--Package
-- Create a package to display appropriate message.
-- Package Definition:

```
Create or replace package pack1 as
        Procedure p1;
        Function f1 return varchar;
End;
```

```
Package created.
```

--Package body

```
Create or replace package body pack1 as
Procedure p1 is
Begin
        dbms_output.put_line ('Hi this is from procedure');
End p1;

Function f1 return varchar is Message varchar(50);
Begin
        message:='Hello this is a function';
        Return (message);
End f1;

End pack1;
```

```
Package Body created.
```

--PL/SQL block

```
Declare
--msg varchar(50);

Begin
--msg:=pack1.f1();
        dbms_output.put_line(pack1.f1 ());
        pack1.p1 ();
End;
```

Output:

```
Hello this is a function
Hi this is from procedure
```

Exercise-40

-- Package
-- Create a package for addition and subtraction of two values.
-- Package Definition:

```
Create or replace package pack2 as
        Procedure p2(a number, b number);
        Function f2 (a number, b number)return number;
End;
```

```
Package created.
```

--Package body

```
Create or replace package body pack2 as
Procedure p2(a in number, b in number) is ans number;
Begin
        ans:=a+b;
        dbms_output.put_line('The addition is:-'|| ans);
End p2;

Function f2 (a in number, b in number)return number is ans number;
Begin
        ans:=a-b;
        return(ans);
End f2;

End pack2;
```

```
Package Body created.
```

--PL/SQL block

```
Declare
        a number:=3;
        b number:=2;
        ans number;
Begin
        pack2.p2 (a,b);
        ans:=pack2.f2(a,b);
        dbms_output.put_line ('The Substraction is:-' || ans);
End;
```

Output:

```
The addition is:-5
The Substraction is:-1
```

Exercise-41

-- Trigger

--Create a trigger to display salary changes in the customer table.

--First create customer table and insert records.

```
create table customers
(id number(3),
name varchar2(20),
age number(2),
address varchar2(20),
salary number(10));
```

ID	NAME	AGE	ADDRESS	SALARY
1	aarva	25	ahmedabad	50000
2	vedanshu	29	pune	75000
3	denil	35	surat	85000

```
insert into customers values(1,'aarva',25,'ahmedabad',50000);
```

--Create a trigger

```
CREATE OR REPLACE TRIGGER display_salary_changes
        BEFORE DELETE OR INSERT OR UPDATE ON customers
        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);
END;
```

```
Trigger created.
```

--PL/SQL block to update customer table..

```
DECLARE
        total_rows number(2);
BEGIN
        UPDATE  customers SET salary = salary + 5000;
        IF sql%notfound THEN
                dbms_output.put_line('no  customers  updated');
        ELSIF sql%found THEN
                total_rows := sql%rowcount;
        dbms_output.put_line( total_rows || ' customers updated ');
        END IF;
END;
```

Output:

```
Old salary: 50000
New salary: 55000
Salary difference: 5000
Old salary: 75000
New salary: 80000
Salary difference: 5000
Old salary: 85000
New salary: 90000
Salary difference: 5000
3 customers updated
```

Exercise-42

PLSQL TABLES [Nested table]

Create a nested table called dept and use it in to query.

```
Create or replace type dept1 as table of varchar2 (10);


-- Now create a table

Create table emp1_1 (emp_no number (10), ename varchar2 (10), deptartment dept1)
Nested table deptartment store as dept_tab;

-- After it insert a record in it

Insert into emp1_1 values (1,'aaa', dept1 ('Account','Cleark'));

-- After it show the record
```

Output:

```
Select * from emp1_1;

EMP_NO ENAME DEPTARTMENT
--------- ------------------------------
      1         AAA        DEPT1 ('Account', 'Cleark')
```

Exercise-43

PLSQL TABLES [Varray]

create a varray called marks_va of size 5 & apply on the table

```
Create or replace type marks_va as varray (5) of number (5);

-- After varray created then create a table student

Create table student (std_no number (10) primary key, name varchar2 (15), marks
marks_va);

-- After table is created then insert a row in the table

Insert into student values (1,'aaa', marks_va (50, 60, 70));

insert into student values(2,'bbb',marks_va(50,60,65,70,75));

-- After row is inserted then

Select * from student;
```

Output:

```
  STD_NO NAME MARKS
--------- ------- ------------------------
       1         AAA     MARKSVA (50, 60, 70)
```