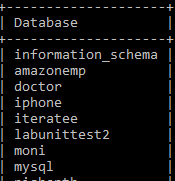
**Aim**: Exercise on set operators.

## Procedure:

**Step 1**: Create a database. mysql> create database std; mysql> use std;

Database changed mysql> show databases;



**Step 2**: Create a table.

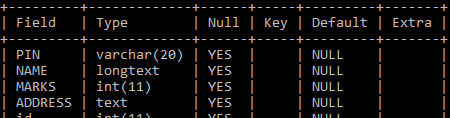
mysql> create table student(pin varchar(20),

-> name longtext,

-> marks int,

-> address text);

Query OK, 0 rows affected (0.45 sec) mysql> desc student;



**Step 3**: Insert values into the table.

mysql> insert into student values(‘20001-cm-001’,vedha’,25,’allahabad’);

mysql> insert into student values(‘20001-cm-046’,’prashanthi’,25,’hyderabad’);

mysql> insert into student values('20001-cme-003'’shail’,28,'hyderabad’);

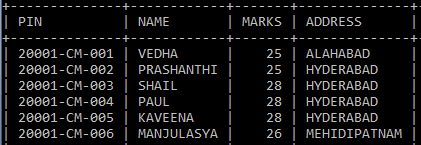
mysql> insert into student values('20001-cm=004',’paul’,28,'hyderabad');

mysql->insert into student values(‘20001-cm-005’,’kaveena’,28,’mehidipatnam’);

mysql->insert into student values(‘20001-cm-006’,’manjulasya’,28,’mehidipatnam’);

Query OK, 1 row affected (0.03 sec)

mysql> select \* from student ;



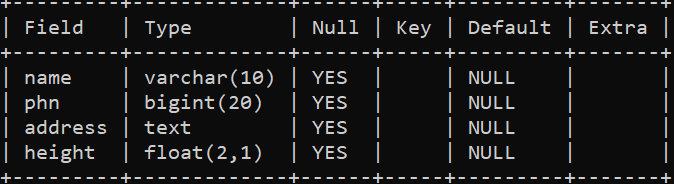
**Step 4:** Create another table.

mysql> create table teacher(name varchar(10),

->phn bigint,

->address text,

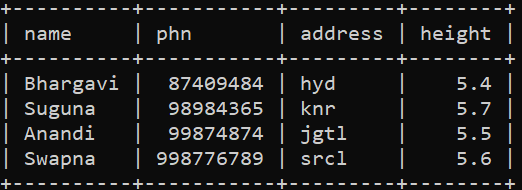
->height float(2,1)); mysql> desc teacher;



**Step 5**: Insert values into the second table.

mysql> insert into teacher values('Bhargavi',87409484,'hyd',5.4); mysql> insert into teacher values('Suguna',98984365,'knr',5.7); mysql> insert into teacher values('Anandi',99874874,'jgtl',5.5); mysql> insert into teacher values('Swapna',998776789,'srcl',5.6); Query OK, 1 row affected (0.04 sec)

mysql> select \* from teacher;

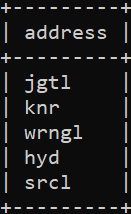


**Syntax of set operator, ’union’.** Select columnlist from table1 union

select columnlist from table2; mysql> select address from student

-> union

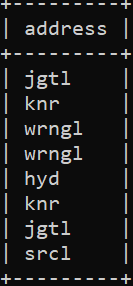
-> select address from teacher;



mysql> select address from student

-> union all

-> select address from teacher;



SQL> create table student(name varchar(10),

1. age int,
2. phn integer,
3. address varchar2(10),
4. height float);

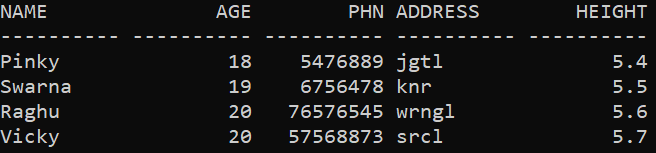
Table created.

SQL> insert into student values('Pinky',18,5476889,'jgtl',5.4); SQL> insert into student values('Swarna',19,6756478,'knr',5.5); SQL> insert into student values('Raghu',20,76576545,'wrngl',5.6); SQL> insert into student values('Vicky',20,57568873,'srcl',5.7); SQL> create table teacher(name varchar(10),

1. age int,
2. phn integer,
3. address varchar2(10),
4. height float);

Table created.

SQL> insert into teacher values('Bhargavi',1,87409484,'hyd',5.4); SQL> insert into teacher values('Suguna',2,98984365,'knr',5.5); SQL> insert into teacher values('Anandi',3,99874874,'jgtl',5.6); SQL> insert into teacher values('Swapna',4,998776789,'srcl',5.9); SQL> select \* from student;



## SYNTAX FOR SET OPERATOR ‘INTERSECT’:

Select columnlist from table1 Intersect

Select columnlist from table2; SQL> select \* from student

1. intersect
2. select \* from teacher; no rows selected

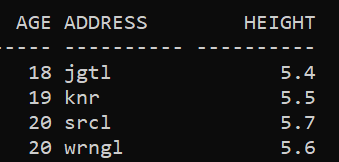
## SYNTAX FOR SET OPERATOR ‘MINUS’:

Select columnlist from table1 Minus

Select columnlist from table2;

SQL> select age,address,height from student

1. minus
2. select age,address,height from teacher;

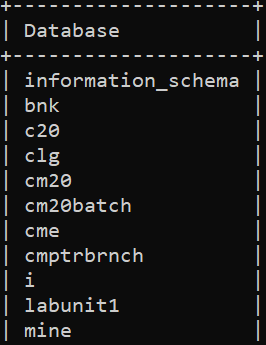


**Aim**: Exercise on orderby and group clauses.

## Procedure:

**Step 1**: Create a database. mysql> create database std; mysql> use std;

Database changed mysql> show databases;



**Step 2**: Create a table.

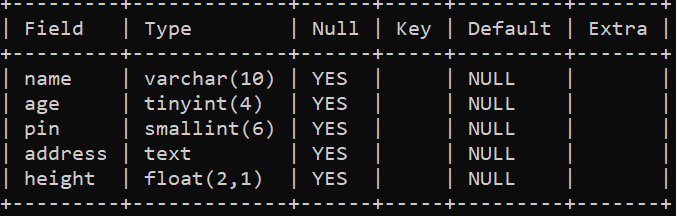
mysql> create table student(name varchar(10),

-> age tinyint,

-> pin smallint,

-> address text,

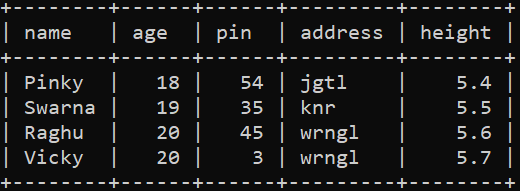
-> height float(2,1)); mysql> desc student;



**Step 3**: Insert values into the table.

mysql> insert into student values('Pinky',18,54,'jgtl',5.4); mysql> insert into student values('Swarna',19,35,'knr',5.5);

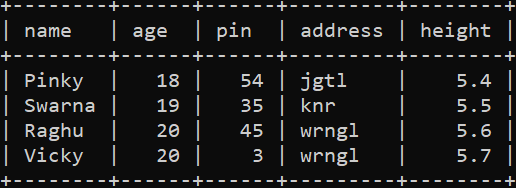
mysql> insert into student values('Raghu',20,45,'wrngl',5.6); mysql> insert into student values('Vicky',20,03,'wrngl',5.7); mysql> select \* from student ;



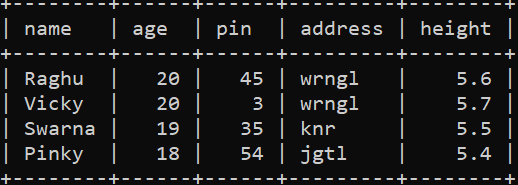
## Syntax for order by clause:

select columnlist from tablename[where condition] order by columns)[asc/desc];

mysql> select \* from student order by age asc;

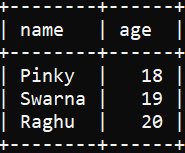


mysql> select \* from student order by age desc;



## SYNTAX FOR GROUP BY CLAUSE:

Select columnlist from tablename group by columnlist; mysql> select name,age from student group by age;



**Aim**: Exercise on sql joins.

## Procedure:

**Step 1**: Create a database. mysql> create database std; mysql> use std;

Database changed

**Step 2**: Create a table. mysql> create table student(

-> name varchar(10),

-> pin smallint,

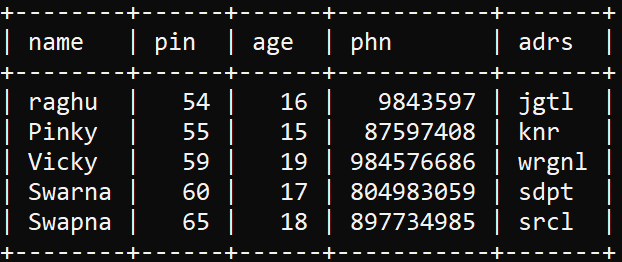
-> age tinyint,

-> phn bigint,

-> adrs text);

**Step 3**: Insert values in table.

mysql> insert into student values('raghu',54,16,09843597,'jgtl'); mysql> insert into student values('Pinky',55,15,87597408,'knr'); mysql> insert into student values('Vicky',59,19,984576686,'wrgnl'); mysql> insert into student values('Swarna',60,17,804983059,'sdpt'); mysql> insert into student values('Swapna',65,18,897734985,'srcl'); mysql> select \* from student;



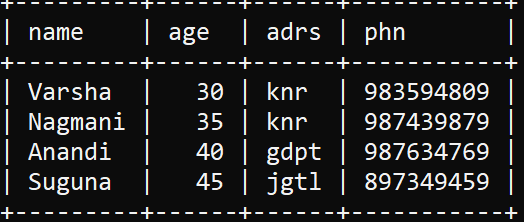
mysql> create table teacher(name varchar(10),

->age tinyint,

->adrs text,

->phn bigint);

mysql> insert into teacher values('Varsha',30,'knr',983594809); mysql> insert into teacher values('Nagmani',35,'knr',987439879); mysql> insert into teacher values('Anandi',40,'gdpt',987634769); mysql> insert into teacher values('Suguna',45,'jgtl',897349459); mysql> select \* from teacher;



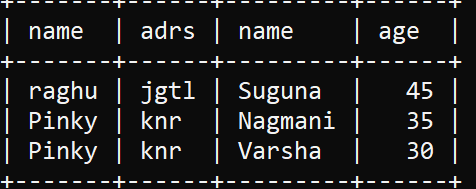
## Syntax of inner join

Select columnlist from table1 inner join table2 on table1.columname=table2.columname;

mysql> select student.name,student.adrs,teacher.name,teacher.age from student

-> inner join teacher on

-> student.adrs=teacher.adrs;



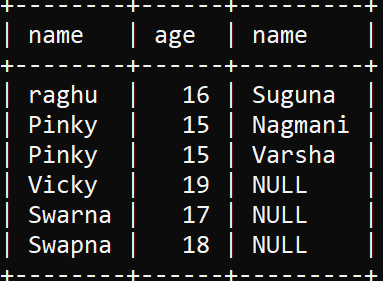
## Syntax of left join

Select columnlist from table1 left join table2 on table1.columname=table2.columname;

mysql> select student.name,student.age,teacher.name from student

-> left join teacher on

-> student.adrs=teacher.adrs;



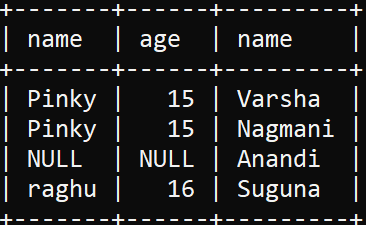
## Syntax of right join

Select columnlist from table1 right join table2 on table1.columname=table2.columname;

mysql> select student.name,student.age,teacher.name from student

-> right join teacher on

-> student.adrs=teacher.adrs;



## Syntax of full join

Select columnlist from table1 left join table2 on table1.columname=table2.columname union

Select columnlist from table1 right join table2 on table1.columname=table2.columname;

mysql> select student.name,student.age,teacher.name from student

-> left join teacher on

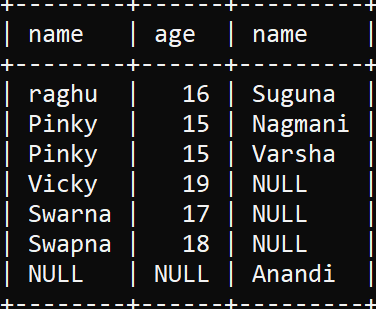
-> student.adrs=teacher.adrs

-> union

-> select student.name,student.age,teacher.name from student

-> right join teacher on

-> student.adrs=teacher.adrs;



**Aim:** Exercise on creating index.

## Procedure:

**Step 1**: Create a database. mysql> create database std; mysql> use std;

Database changed

**Step 2:** Create a table. mysql> create table student(

-> name varchar(10),

-> pin smallint,

-> age tinyint,

-> phn bigint,

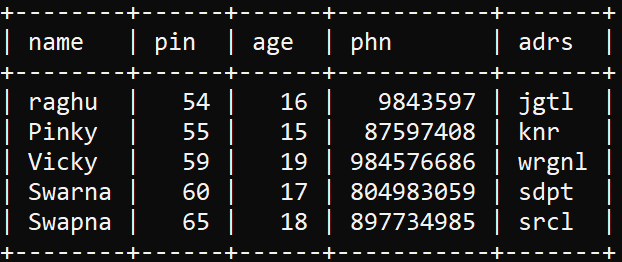
-> adrs text);

Query OK, 0 rows affected (0.40 sec)

**Step 3**: Insert values in table.

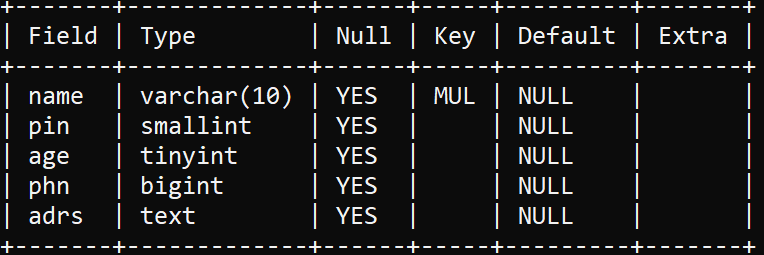
mysql> insert into student values('raghu',54,16,09843597,'jgtl'); mysql> insert into student values('Pinky',55,15,87597408,'knr'); mysql> insert into student values('Vicky',59,19,984576686,'wrgnl'); mysql> insert into student values('Swarna',60,17,804983059,'sdpt'); mysql> insert into student values('Swapna',65,18,897734985,'srcl'); Query OK, 1 row affected (0.08 sec)

mysql> select \* from student;



## SYNTAX TO CREATE INDEX:

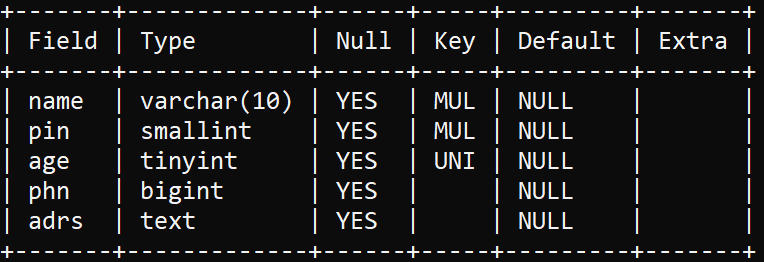
mysql> create index stdindex on student(name); mysql> desc student;



mysql> create index stddindex on student(pin);

## SYNTAX TO CREATE UNIQUE INDEX:

Create unique index indexname on tablename(columnlist); mysql> create unique index stdddindex on student(age); mysql> desc student;



**Aim:** Exercise on creating sequence.

## Procedure:

**Step 1:** Create a database. mysql> create database std; mysql> use std;

Database changed

**Step 2:** Create a table.

## SYNTAX TO CREATE SEQUENCE:

Create table tablename(columnname primary key auto\_increment); mysql> create table student(id int not null primary key auto\_increment,

->name varchar(10),

->pin tinyint,

->age smallint,

->adrs text);

**Step 3**: Insert values in table.

mysql> insert into student(name,pin,age,adrs)

-> values('raghu',45,17,'jgtl');

mysql> insert into student(name,pin,age,adrs)

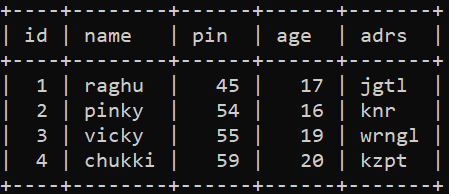
-> values('pinky',54,16,'knr');

mysql> insert into student(name,pin,age,adrs)

-> values('vicky',55,19,'wrngl');

mysql> insert into student(name,pin,age,adrs)

-> values('chukki',59,20,'kzpt'); mysql> select \* from student;



sql> create table student(

1. id int,
2. name varchar(10),
3. pin int,
4. adrs varchar(10) 6 );

table created.

## SYNTAX TO CREATE SEQUENCE

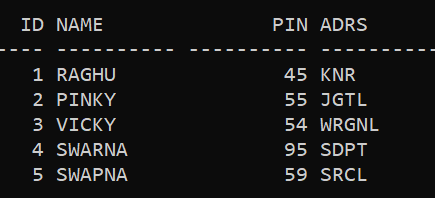
create sequence sequencename minvalue value

maxvalue value start with value increment by value cache value/nocache nocycle/cycle value;

sql> create sequence stdsqnce

1. minvalue 1
2. maxvalue 5
3. start with 1
4. increment by 1
5. cache 20
6. nocycle; sequence created.

sql> insert into student values(stdsqnce.nextval,'Raghu',45,'Knr'); sql> insert into student values(stdsqnce.nextval,'PINKY',55,'JGTL'); sql> insert into student values(stdsqnce.nextval,'VICKY',54,'WRGNL'); sql> insert into student values(stdsqnce.nextval,'SWARNA',95,'SDPT'); sql> insert into student values(stdsqnce.nextval,'SWAPNA',59,'SRCL'); sql> select \* from student;



sql> alter sequence stdsqnce

1. minvalue 0
2. maxvalue 6
3. increment by 1
4. cache 20
5. nocycle; sequence altered.

sql> drop sequence stdsqnce; sequence dropped.

**Aim:** Exercise on creating synonym.

## Procedure:

**Step 1:** Create a database. mysql> create database std; mysql> use std;

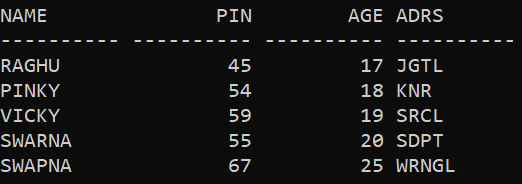
Database changed

**Step 2**: Create a table. sql> create table student(

1. name varchar(10),
2. pin int,
3. age number,
4. adrs varchar(10) 6 );

table created.

sql> insert into student values('RAGHU',45,17,'JGTL'); sql> insert into student values('PINKY',54,18,'KNR'); sql> insert into student values('VICKY',59,19,'SRCL'); sql> insert into student values('SWARNA',55,20,'SDPT');

sql> insert into student values('SWAPNA',67,25,'WRNGL'); sql> select \* from student;

## SYNTAX TO CREATE SYNONYM:

Create synonym synonymname for tablename; sql> create synonym vidyarthi for student; synonym created.

## SYNTAX TO DROP SYNONYM:

Drop synonym synonymname; sql> drop synonym vidyarthi; synonym dropped.

**Aim:** Exercise on creating view.

## Procedure:

**Step 1:** Create a database. mysql> create database std; mysql> use std;

Database changed

**Step 2**: Create a table.

mysql> create table student(name varchar(10),

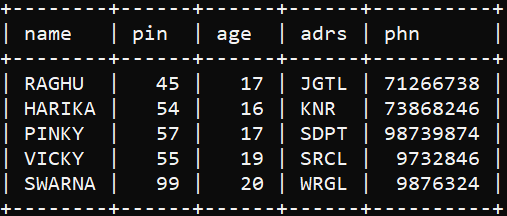
-> pin smallint,

-> age tinyint,

-> adrs text,

-> phn bigint);

mysql> insert into student values('RAGHU',45,17,'JGTL',71266738); mysql> insert into student values('HARIKA',54,16,'KNR',73868246); mysql> insert into student values('PINKY',57,17,'SDPT',98739874); mysql> insert into student values('VICKY',55,19,'SRCL',9732846); mysql> insert into student values('SWARNA',99,20,'WRGL',9876324); mysql> select \* from student;



## SYNTAX TO CREATE VIEW:

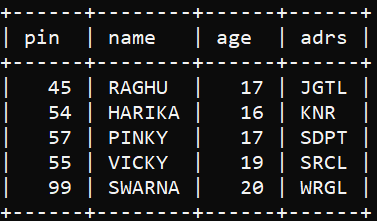
Create view viewname As

Select columnlist from tablename; mysql> create view sview

-> as

-> select pin,name,age,adrs from student; Query OK, 0 rows affected (0.48 sec)

mysql> select \* from sview;



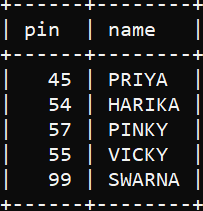
## SYNTAX TO ALTER VIEW

Alter view viewname As

Select columlist from tablename; mysql> alter view sview

-> as

-> select pin,name from student; Query OK, 0 rows affected (0.14 sec) mysql> SELECT \* FROM SVIEW;



mysql> CREATE TABLE EMPLOYEE(NAME VARCHAR(20),

-> EID INT AUTO\_INCREMENT PRIMARY KEY,

-> ADRS TEXT);

mysql> INSERT INTO EMPLOYEE(name,adrs)

-> values('raghu','jgt');

mysql> INSERT INTO EMPLOYEE(name,adrs)

-> values('raghuram','knrr');

mysql> INSERT INTO EMPLOYEE(name,adrs)

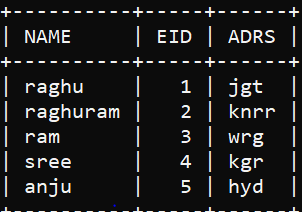
-> values('ram','wrg');

mysql> INSERT INTO EMPLOYEE(name,adrs)

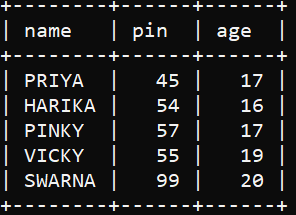
-> values('sree','kgr');

mysql> INSERT INTO EMPLOYEE(name,adrs)

-> values('anju','hyd'); mysql> select \* from employee;



mysql> alter table student drop column adrs; mysql> alter table student drop column phn; mysql> select \* from student;



## SYNTAX TO CREATE VIEW ON MULTIPLE TABLES:

Create view viewname

As select columnlist from tablename1 Union

Select columnlist from tablename2; mysql> create view eview

-> as select \* from student

-> union

-> select \* from employee;

Query OK, 0 rows affected (0.09 sec)

**Aim:** Exercise on creating cluster.

## Procedure:

**Step 1:** Create a database. mysql> create database std; mysql> use std;

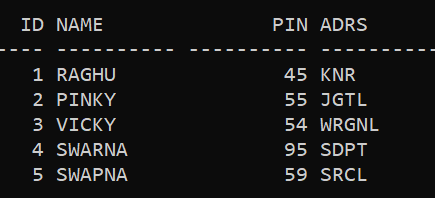
Database changed

**Step 2**: Create a table.

SQL> CREATE TABLE STUDENT(

1. ID INT,
2. NAME VARCHAR(10),
3. PIN INT,
4. ADRS VARCHAR(10) 6 );

SQL> insert into student values(1,'RAGHU',45,'KNR'); SQL> insert into student values(2,'PINKY',55,'JGTL'); SQL> insert into student values(3,'VICKY',54,'WRGNL'); SQL> insert into student values(4,'SWARNA',95,'SDPT'); SQL> insert into student values(5,'SWAPNA',59,'SRCL'); SQL> SELECT \* FROM STUDENT;



## SYNTAX TO CREATE CLUSTER:

Create cluster clustername (clusterkeycolumn datatype(size) Size 512

Storage(initial value next value);

SQL> create cluster stdcluster(PIN int)

1. size 512
2. storage(initial 100k next 50k); Cluster created.

## SYNTAX TO CREATE CLUSTER INDEX:

Create index indexname On cluster clustername;

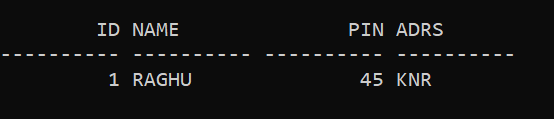
SQL> CREATE INDEX STDINDEX

2 ON CLUSTER STDCLUSTER; Index created.

SQL> CREATE TABLE STD

1. CLUSTER STDCLUSTER(PIN)
2. AS
3. SELECT \* FROM STUDENT WHERE PIN=45; Table created.

SQL> SELECT \* FROM STD;



## SYNTAX TO CREATE HASHKEY:

Create cluster clustername(columnname datatype(size) Size 512

Hashkey value

Storage(initial value next value);

SQL> CREATE CLUSTER LANGUAGE(

1. CLANG VARCHAR(10))
2. SIZE 512
3. HASHKEYS 10
4. STORAGE(INITIAL 100K NEXT 50K); SQL> DROP CLUSTER LANGUAGE; Cluster dropped.

**Aim:** Create PL/SQL procedure to perform arithmetic operation in mysql.

# Procedure:

mysql> use record; mysql> delimiter #

mysql> create procedure arithmetic\_operation(a int, b int)

-> begin

-> declare c int;

-> set c=a+b;

-> select c as 'sum';

-> set c=a-b;

-> select c as 'difference';

-> set c=a\*b;

-> select c as 'product';

-> set c=a/b;

-> select c as 'division';

-> set c=a%b;

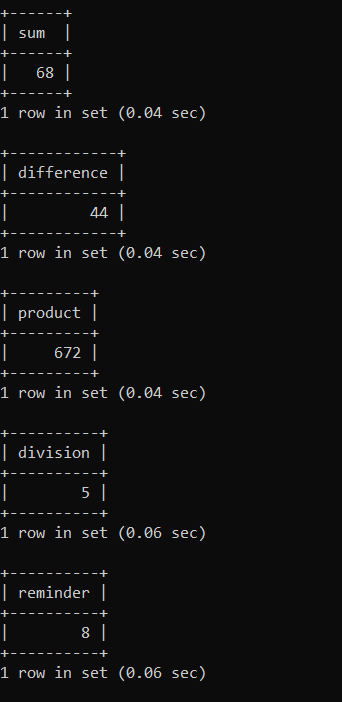
-> select c as 'reminder';

-> end;

-> #

Query OK, 0 rows affected (0.03 sec)

mysql> call arithmetic\_operation(56,12);

-> #

**Aim:** Create a PL/SQl procedure to the eligibility of voting using if statement in mysql.

# Procedure:

mysql> delimiter #

mysql> create procedure voting\_eligibility(a int)

-> begin

-> if(a>=18) then

-> select a as 'eligible for voting';

-> if(a<18) then

-> select a as 'not eligible for voting';

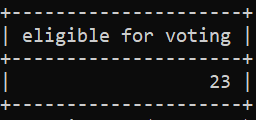
-> end if;

-> end;

-> #

Query OK, 0 rows affected (0.00 sec)

mysql> call voting\_eligibility(23);

-> #

**Aim:** Create a PL/SQL procedure to find the greatest of two numbers using if else statement in mysql.

# Procedure:

mysql> delimiter @

mysql> create procedure greatest\_two(a int,b int)

-> begin

-> if(a>b) then

-> select a as 'greatest is';

-> else

-> select b as 'greatest is';

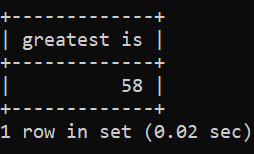
-> end if;

-> end;

-> @

Query OK, 0 rows affected (0.00 sec)

mysql> call greatest\_two(45,58);

-> @

**Aim:** Create a PL/SQL procedure to find the greatest among three numbers using nested if in mysql

# Procedure:

mysql> delimiter @

mysql> create procedure largest\_three(a int,b int,c int)

-> begin

-> if(a>b and a>c) then

-> select a as 'largrst is';

-> elseif(b>a and b>c) then

-> select b as 'largrst is';

-> else

-> select c as 'largrst is';

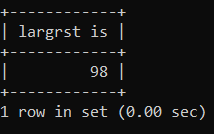
-> end if;

-> end;

-> @

Query OK, 0 rows affected (0.01 sec)

mysql> call largest\_three(26,67,98);

-> @

**Aim:** Create a PL/SQL procedure to check the current bill using CASE in mysql.

# Procedure

mysql> create procedure current\_bill(unts int)

-> begin

-> declare bill int;

-> case

-> when unts<100

-> then

-> set bill=unts\*2;

-> select bill;

-> when unts>100

-> then

-> set bill=unts\*3;

-> select bill;

-> end case;

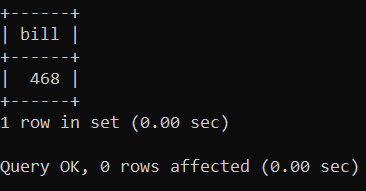
-> end;

-> @

Query OK, 0 rows affected (0.00 sec)

mysql> call current\_bill(156);

-> @



**Aim:** Create a PL/SQL procedure to find the root of a quadratic equation using nested if in mysql.

# Procedure:

mysql> delimiter @

mysql> create procedure qroot()

-> begin

-> declare a int;

-> declare b int;

-> declare c int;

-> declare discriminant int;

-> declare root1 float;

-> declare root2 float;

-> set a=2;

-> set b=4;

-> set c=1;

-> set discriminant=(b\*b)-(4\*a\*c);

-> if (discriminant>0) then

-> set root1=(-b+sqrt(discriminant))/(2\*a);

-> set root2=(-b-sqrt(discriminant))/(2\*a);

-> select root1;

-> select root2;

-> elseif (discriminant=0) then

-> set root1=-b/(2\*a);

-> set root2=-b/(2\*a);

-> select root1;

-> select root2;

-> else

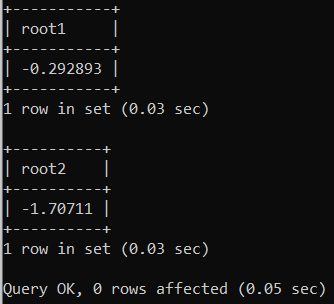
-> select 'root is imaginary';

-> end if;

-> end;

-> @

Query OK, 0 rows affected (0.01 sec) mysql> call qroot();

-> @

**Aim:** Write a PL/SQl procedure to to print Fibonacci series using loop and iterate statement in mysql.

**Procedure:** Fibonacci series using LOOp in mysql.

mysql> delimiter @

mysql> create procedure fibonaci(n int)

-> begin

-> declare i int;

-> declare a int;

-> declare b int;

-> declare c int;

-> set i=1;

-> set a=-1;

-> set b=1;

-> our\_record:

-> loop

-> set i=i+1;

-> set c=a+b;

-> select c;

-> set a=b;

-> set b=c;

-> if(i=n) then

-> leave our\_record;

-> end if;

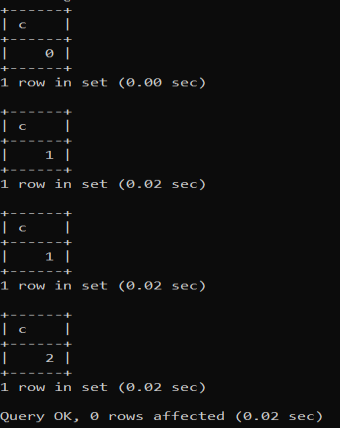
-> end loop our\_record;

-> end;

-> @

Query OK, 0 rows affected (0.02 sec)

mysql> call fibonaci(5);

-> @

Fibonacci series using LOOp in mysql

mysql> create procedure fibo(n int)

-> begin

-> declare i int;

-> declare a int;

-> declare b int;

-> declare c int;

-> set i=1;

-> set a=-1;

-> set b=1;

-> our\_record:

-> loop

-> set i=i+1;

-> set c=a+b;

-> select c;

-> set a=b;

-> set b=c;

-> if(i<=n) then

-> iterate our\_record;

-> end if;

-> leave our\_record;

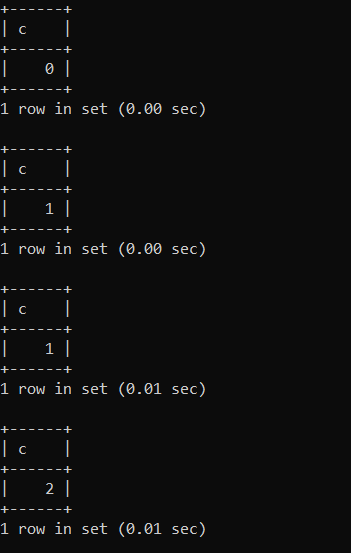
-> end loop our\_record;

-> end;

-> @

Query OK, 0 rows affected (0.00 sec)

mysql> call fibo(4);

-> @

**Aim:** Write a PL/SQL to find reverse of a number and check whether it is padindrome or not in mysql.

# Procedure:

mysql> create procedure palindrome(x int)

-> begin

-> declare i int;

-> declare num int;

-> declare rev int;

-> set rev=0;

-> set num=x;

-> while(num>0)

-> do

-> set i=mod(num,10);

-> set rev=(rev\*10)+i;

-> set num=floor(num/10);

-> end while;

-> select rev as 'reverse is';

-> if(x=rev) then

-> select x as 'it is a palindrome';

-> else

-> select x as 'it is not a palindrome';

-> end if;

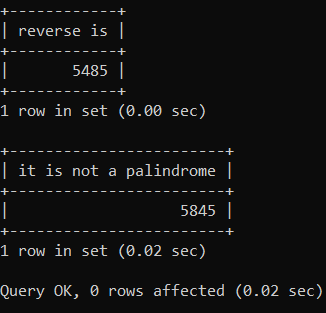
-> end;

-> @

Query OK, 0 rows affected (0.00 sec)

mysql> call palindrome(5845);

-> @



**Aim:** Write a PL/SQL procedure to find the number of digits in a given number using while loop in mysql.

# Procedure:

mysql> create procedure counting\_digit(num int)

-> begin

-> declare c int;

-> set c=0;

-> while(num>0)

-> do

-> set c=c+1;

-> set num=floor(num/10);

-> end while;

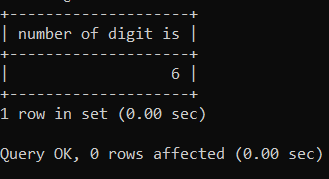
-> select c as 'number of digit is';

-> end;

-> @

Query OK, 0 rows affected (0.00 sec)

mysql> call counting\_digit(455845);

-> @

**Aim:** Write a PL/SQL procedure to print the first 10 natural numbers using repeat loop in mysql.

# Procedure:

mysql> create procedure repeat\_demo()

-> begin

-> declare i int;

-> set i=0;

-> repeat

-> set i=i+1;

-> select i;

-> until i>=10

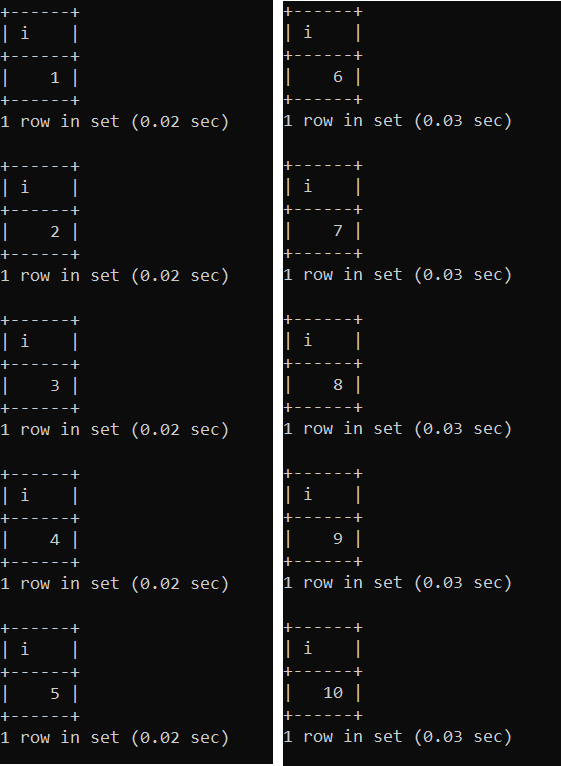
-> end repeat;

-> end;

-> @

Query OK, 0 rows affected (0.00 sec)

mysql> call repeat\_demo();

-> @

**Aim:** Write a PL/SQL procedure to select the day of a week

using case in mysql.

# Procedure:

mysql> create procedure day\_name(day int)

-> begin

-> declare day\_name varchar(10);

-> case

-> when day=1 then

-> set day\_name='Monday';

-> select day\_name;

-> when day=2 then

-> set day\_name='Tuesday';

-> select day\_name;

-> when day=3 then

-> set day\_name='Wednesday';

-> select day\_name;

-> when day=4 then

-> set day\_name='Thursday';

-> select day\_name;

-> when day=5 then

-> set day\_name='Friday';

-> select day\_name;

-> when day=6 then

-> set day\_name='Saturday';

-> select day\_name;

-> when day=7 then

-> set day\_name='Sunday';

-> select day\_name;

-> end case;

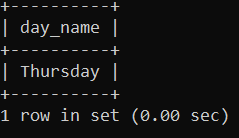
-> end;

-> @

Query OK, 0 rows affected (0.00 sec)

mysql> call day\_name(4);

-> @



**Aim:** write a PL/SQL procedure to check whether the given number is prime or not using while loop in mysql.

# Procedure:

mysql> create procedure primeno(n int)

-> begin

-> declare count int;

-> declare i int;

-> set count=0;

-> set i=2;

-> while(i<=n/2)

-> do

-> if(n%i=0)

-> then

-> set count=1;

-> end if;

-> set i=i+1;

-> end while;

-> if(count=0)

-> then

-> select n as 'prime number';

-> else

-> select n as 'not prime number';

-> end if;

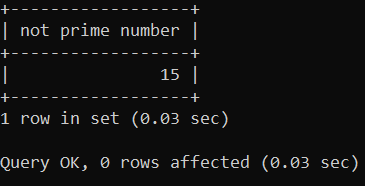
-> end;

-> @

Query OK, 0 rows affected (0.00 sec)

mysql> call primeno(15);

-> @



**Aim:** Write a PL/SQL procedure to check whether given number is Armstrong or not in mysql.

# Procedure:

mysql> create procedure armstrong(num int)

-> begin

-> declare temp int;

-> declare i int;

-> declare arm int;

-> set arm=0;

-> set temp=num;

-> while(num>0)

-> do

-> set i=mod(num,10);

-> set arm=(i\*i\*i)+arm;

-> set num=floor(num/10);

-> end while;

-> if(arm=temp) then

-> select temp as 'Armstrong number';

-> else

-> select temp as 'Not Armstrong number';

-> end if;

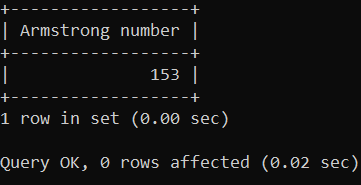
-> end;

-> @

Query OK, 0 rows affected (0.00 sec)

mysql> call armstrong(153);

-> @



**Aim:** Write a PL/SQL procedure to find the factorial of a given number using while loop in mysql.

# Procedure:

mysql> delimiter @

mysql> create procedure factorial(num int)

-> begin

-> declare n int;

-> declare f int;

-> set n=num;

-> set f=1;

-> while(n>0)

-> do

-> set f=f\*n;

-> set n=n-1;

-> end while;

-> select f as 'factorial';

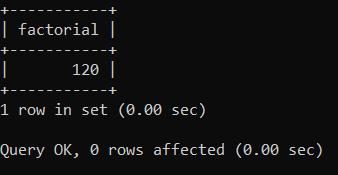
-> end;

-> @

Query OK, 0 rows affected (0.00 sec)

mysql> call factorial(5);

-> @



**Aim:** write a PL/SQl procedure to check the whether the given number is even or odd.

# Procedure:

mysql> delimiter @

mysql> create procedure odd\_even(num int)

-> begin

-> if(num%2=0) then

-> select num as 'even number';

-> else

-> select num as 'odd number';

-> end if;

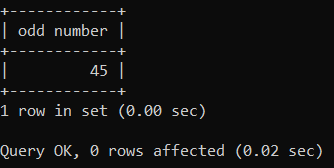
-> end;

-> @

Query OK, 0 rows affected (0.00 sec)

mysql> call odd\_even(45);

-> @



**Aim:** Write a PL/SQL to find the LCM and GCD of given two number in mysql.

# Procedure:

mysql> delimiter @

mysql> create procedure gcdlcm(n1 int,n2 int)

-> begin

-> declare i int;

-> declare j int;

-> declare lcm int;

-> declare gcd int;

-> set j=n1\*n2;

-> set i=n1%n2;

-> while(i!=0)

-> do

-> set n1=n2;

-> set n2=i;

-> set i=n1%n2;

-> end while;

-> set lcm=floor(j/n2);

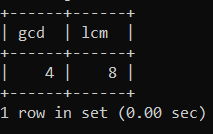
-> select n2 as 'gcd',lcm;

-> end;

-> @

Query OK, 0 rows affected (0.00 sec) mysql> call gcdlcm(4,8);

-> @



**Aim:** Write a PL/SQL procedure to find the sum of n natural numbers in mysql.

# Procedure:

mysql> delimiter @

mysql> create procedure n\_sum(x int)

-> begin

-> declare i int;

-> declare s int;

-> set i=0;

-> set s=0;

-> vivek:

-> loop

-> set i=i+1;

-> set s=s+i;

-> if i=x then

-> leave vivek;

-> end if;

-> end loop vivek;

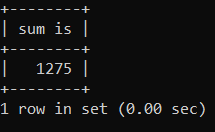
-> select s as 'sum is';

-> end;

-> @

Query OK, 0 rows affected (0.00 sec) mysql> call n\_sum(50);

-> @



**Aim:** Write a PL/SQL procedure to implement swapping of two number in mysql.

# Procedure:

mysql> delimiter @

mysql> create procedure swap(n1 int,n2 int)

-> begin

-> declare temp int;

-> select 'before swapping';

-> select n1,n2;

-> set temp=n1;

-> set n1=n2;

-> set n2=temp;

-> select 'After swapping';

-> select n1,n2;

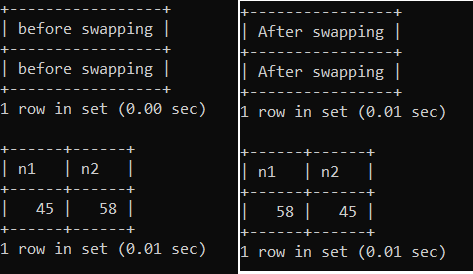
-> end;

-> @

Query OK, 0 rows affected (0.00 sec)

mysql> call swap(45,58);

-> @



**Aim:** Write a PL/SQL procedure to implement CASE statement to find whether the given alphabet is vowel or consonant in mysql.

# Procedure:

mysql> create procedure vowelornot(alph char(1))

-> begin

-> case

-> when alph='a' then

-> select alph as 'Vowel';

-> when alph='e' then

-> select alph as 'Vowel';

-> when alph='i' then

-> select alph as 'Vowel';

-> when alph='o' then

-> select alph as 'Vowel';

-> when alph='u' then

-> select alph as 'Vowel';

-> else

-> select alph as 'consonant';

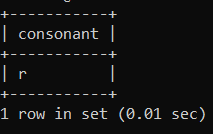
-> end case;

-> end;

-> @

Query OK, 0 rows affected (0.00 sec) mysql> call vowelornot('r');

-> @



**Aim:** Write a PL/SQL procedure to implement IN parameter in mysql.

# Procedure:

**Step-1:** creating the database mysql>create database our\_record; mysql> use our\_record;

**Step-2:** Creating the table

mysql> create table college(pin int,Name varchar(10)); Query OK, 0 rows affected (0.05 sec)

**Step-3:** Inserting the values into the table. mysql> insert into college values(12,'Vinay'),

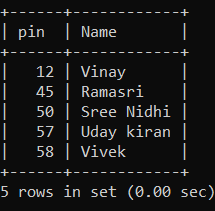
-> (45, 'Ramasri'),

-> (50, 'Sree Nidhi'),

-> (57,'Uday kiran'),

-> (58,'Vivek');

Query OK, 5 rows affected (0.04 sec) Records: 5 Duplicates: 0 Warnings: 0 mysql> select \* from college;



**Step-4:** creating procedure by using in parameter mysql> delimiter @

mysql> create procedure in\_demo(in spin int)

-> begin

-> select \* from college where spin=pin;

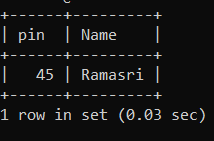
-> end;

-> @

Query OK, 0 rows affected (0.00 sec)

mysql> call in\_demo(45);

-> @



**Aim:** Write a PL/SQL procedure to implement OUT parameter in mysql.

# Procedure:

mysql> create table student(name varchar(10),

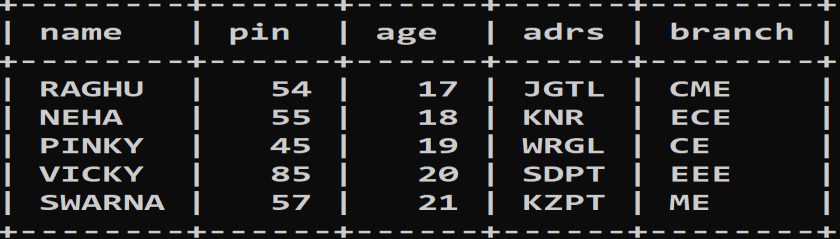
-> pin smallint,

-> age tinyint,

-> adrs text,

-> branch mediumtext);

mysql> insert into student values('RAGHU',54,17,'JGTL','CME'); mysql> insert into student values('NEHA',55,18,'KNR','ECE'); mysql> insert into student values('PINKY',45,19,'WRGL','CE'); mysql> insert into student values('VICKY',85,20,'SDPT','EEE'); mysql> insert into student values('SWARNA',57,21,'KZPT','ME'); mysql> select \* from student;



mysql> delimiter $

mysql> create procedure outpro(in spin int,out total int)

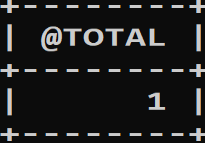
-> begin

-> select count(\*) into total from student where pin=spin;

-> end;

-> $

mysql> call outpro(54,@total); $ mysql> select @total; $



**Aim:** Write a PL/SQL procedure to implement INOUT parameter.

## Procedure:

mysql> create table student(name varchar(10),

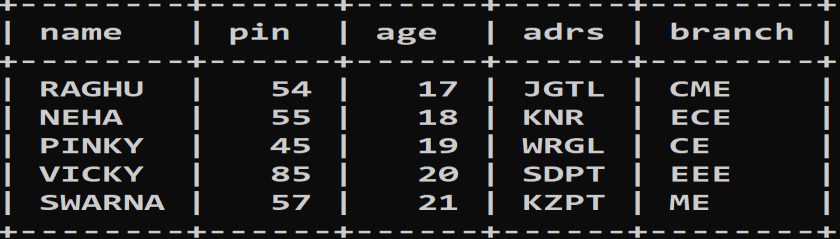
-> pin smallint,

-> age tinyint,

-> adrs text,

-> branch mediumtext);

mysql> insert into student values('RAGHU',54,17,'JGTL','CME'); mysql> insert into student values('NEHA',55,18,'KNR','ECE'); mysql> insert into student values('PINKY',45,19,'WRGL','CE'); mysql> insert into student values('VICKY',85,20,'SDPT','EEE'); mysql> insert into student values('SWARNA',57,21,'KZPT','ME'); mysql> select \* from student;



mysql> delimiter $

mysql> create procedure scounter(inout count int,in inc int)

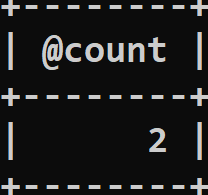
-> begin

-> set count=count+inc;

-> end; $

mysql> set @count=1; $

mysql> call scounter(@count,1); $ mysql> select @count; $



**Aim:** Write a Pl/SQL function using else if ladder else in mysql.

# Procedure:

mysql> create function income(m\_income int)

-> returns varchar(20)

-> begin

-> declare in\_level varchar(20);

-> if(m\_income<=10000) then

-> set in\_level='Low\_income';

-> elseif(m\_income>10000 and m\_income<100000) then

-> set in\_level='High\_income';

-> else

-> set in\_level='Very\_high\_income';

-> end if;

-> return in\_level;

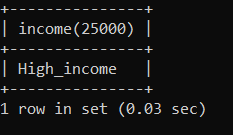
-> end;

-> $

Query OK, 0 rows affected (0.00 sec)

mysql> select income(25000);

-> $



**Aim:** Write a Pl/SQL function using iterate in mysql.

# Procedure:

mysql> delimiter @

mysql> create function sumofno(x int)

-> returns int

-> begin

-> declare i int;

-> declare s int;

-> set s=0;

-> set i=0;

-> viram: loop

-> set i=i+1;

-> set s=s+i;

-> if(i<x) then

-> iterate viram;

-> end if;

-> leave viram;

-> end loop viram;

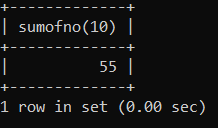
-> return s;

-> end;

-> @

Query OK, 0 rows affected (0.00 sec) mysql> select sumofno(10);

-> @



**Aim:** Write a Pl/SQL function using label in mysql. mysql> delimiter @

mysql> create function label\_demo(x int)

-> returns int

-> begin

-> viram: loop

-> set x=x+1;

-> if(x<10) then

-> iterate viram;

-> end if;

-> leave viram;

-> end loop viram;

-> return x;

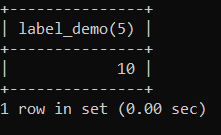
-> end;

-> @

Query OK, 0 rows affected (0.00 sec)

mysql> select label\_demo(5);

-> @



**Aim:** Write a PL/SQL function on cursor in my sql

# Procedure:

**Step-1:** creating the database mysql>create database our\_record; mysql> use our\_record;

**Step-2:** Creating the table

mysql> create table college(pin int,Name varchar(10)); Query OK, 0 rows affected (0.05 sec)

**Step-3:** Inserting the values into the table. mysql> insert into college values(12,'Vinay'),

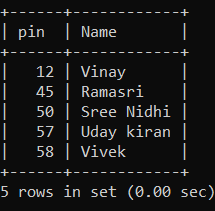
-> (45, 'Ramasri'),

-> (50, 'Sree Nidhi'),

-> (57,'Uday kiran'),

-> (58,'Vivek');

Query OK, 5 rows affected (0.04 sec) Records: 5 Duplicates: 0 Warnings: 0 mysql> select \* from college;



**Step-4:** creating function on cursor. mysql> delimiter @

mysql> create function findpin(name\_in varchar(20))

-> returns int

-> begin

-> declare id int default 0;

-> declare c1 cursor for

-> select pin from college where name=name\_in;

-> open c1;

-> fetch c1 into id;

-> close c1;

-> return id;

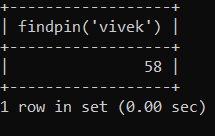
-> end;

-> @

Query OK, 0 rows affected (0.00 sec)

mysql> select findpin('vivek');

-> @



**Aim:** Write a PL/SQL program on creating triggers in mysql.

# Procedure:

**Step-1:** creating the database mysql>create database our\_record; mysql> use our\_record;

**Step-2:** Creating the table

mysql> create table account(id int,Name varchar(20), Salary bigint); Query OK, 0 rows affected (0.03 sec)

**Step-3:** Inserting the values into the table.

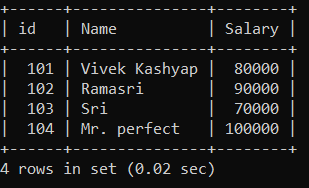
mysql> insert into account values(101,'Vivek Kashyap',80000),

-> (102,'Ramasri',90000),

-> (103,'Sri',70000),

-> (104,'Mr. perfect',100000);

Query OK, 4 rows affected (0.01 sec) Records: 4 Duplicates: 0 Warnings: 0 mysql> Select \* from account;



**Step-4:** creating the trigger.

mysql> delimiter &

mysql> create trigger update\_check

-> before

-> update

-> on

-> account

-> for each row

-> begin

-> if (new.salary<0) then

-> set new.salary=0;

-> elseif(new.salary>10000) then

-> set new.salary=40000;

-> end if;

-> end;

-> &

Query OK, 0 rows affected (0.02 sec)

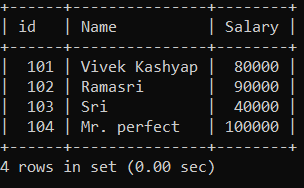
mysql> update account set salary=30000 where id=103;

-> &

Query OK, 1 row affected (0.03 sec)

Rows matched: 1 Changed: 1 Warnings: 0 mysql> select \* from account;

-> &

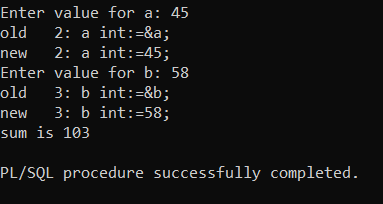


**Aim:** Write a PL/SQL procedure to add two number in oracle.

# Procedure:

SQL> set serveroutput on SQL> declare

1. a int:=&a;
2. b int:=&b;
3. c int;
4. begin 6 c:=a+b;
5. dbms\_output.put\_line('sum is '||c);
6. end; 9 /



**Aim:** Write a PL/SQL procedure to find the sum of n natural number in oracle using while loop in oracle.

# Procedure:

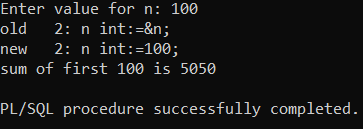
SQL> set serveroutput on SQL> declare

1. n int:=&n;
2. i int:=0;
3. s int:=0;
4. begin
5. while(i<n)
6. loop

8 i:=i+1;

9 s:=s+i;

1. end loop;
2. dbms\_output.put\_line('sum of first '||n||' is '||s);
3. end; 13 /



**Aim:** write a PL/SQL procedure to print prime number upto a given limit in oracle.

# Procedure:

SQL> create procedure primen( i in number)

1. as
2. n number;
3. m number;
4. c number;
5. begin
6. for n in 2..i loop 8 c:=0;
7. for m in 1..n loop
8. if mod(n,m)=0 then 11 c:=c+1;
9. end if;
10. end loop;
11. if c<=2 then
12. dbms\_output.put\_line(' '||n);
13. end if;
14. end loop;
15. end;

19 /

Procedure created.

SQL> execute primen(30); 2

3

5

7

11

13

17

19

23

29

PL/SQL procedure successfully completed.

**Aim:** write a PL/SQL procedure to find the factorial of a number using loop in oracle.

# Procedure:

SQL> create or replace procedure fact\_n(a number) as

1. m number:=1;
2. i number:=1;
3. begin
4. loop

6 m:=m\*i;

7 i:=i+1;

1. exit when i>a;
2. end loop;
3. dbms\_output.put\_line('factorial of '||a||' is '||m);
4. end;

12 /

Procedure created.

SQL> execute fact\_n(5); factorial of 5 is 120

PL/SQL procedure successfully completed.

**Aim:** write a PL/SQL procedure to print multiplication table of a number using loop in oracle.

# Procedure:

SQL> create or replace procedure tab\_n(a number) as

1. n number;
2. i number:=1;
3. begin
4. loop

6 n:=a\*i;

7 dbms\_output.put\_line(a||'x'||i||'='||n); 8 i:=i+1;

1. exit when i>10;
2. end loop;
3. end;

12 /

Procedure created. SQL> execute tab\_n(3); 3x1=3

3x2=6

3x3=9

3x4=12

3x5=15

3x6=18

3x7=21

3x8=24

3x9=27

3x10=30

PL/SQL procedure successfully completed.

**Aim:** write a PL/SQL procedure to print first 20 natural numbers using while loop in oracle.

# Procedure:

SQL> create or replace procedure natu(a number:=20) as

|  |  |
| --- | --- |
| 1. i number:=1; 2. begin 3. while i<=20 loop 4. dbms\_output.put\_line(i); |  |
| 6 i:=i+1; |
| 7 end loop; |
| 8 end; |
| 9 / |
| Procedure created. |
| SQL> execute natu(); |
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |
| 8 | 15 |
| 9 | 16 |
| 10 | 17 |
| 11 | 18 |
| 12 | 19 |
| 13 | 20 |
| 14 | PL/SQL procedure successfully completed. |

**Aim:** Write a Pl/SQL procedure to print Fibonacci series in oracle.

# Procedure:

SQL> create or replace procedure fibb(a number) as

1. f number:=0;
2. s number:=1;
3. c number;
4. i number;
5. begin
6. dbms\_output.put\_line(f);
7. dbms\_output.put\_line(s);
8. for i in 1..a-2 loop 10 c:=f+s;

11 f:=s;

12 s:=c;

1. dbms\_output.put\_line(c);
2. end loop;
3. end;

16 /

Procedure created. SQL> execute fibb(6); 0

1

1

2

3

5

PL/SQL procedure successfully completed.

**Aim:** Write a Pl/SQL procedure to print even numbers upto given limit using if in oracle.

# Procedure:

SQL> create or replace procedure even(a number) as

1. i number:=1;
2. begin
3. dbms\_output.put\_line('even numbers are:');
4. while i<=a loop
5. if mod(i,2)=0 then
6. dbms\_output.put\_line(i);
7. end if; 9 i:=i+1;
8. end loop;
9. end;

12 /

Procedure created. SQL> execute even(10); even numbers are:

2

4

6

8

10

PL/SQL procedure successfully completed.

**Aim:** Write a Pl/SQL procedure to find the grades of student using case in oracle.

# Procedure:

SQL> create or replace procedure grade(m in number)

1. as
2. begin
3. case
4. when m>90 and m<=100 then
5. dbms\_output.put\_line('grade is A+');
6. when m>80 and m<=90 then
7. dbms\_output.put\_line('grade is A');
8. when m>60 and m<=80 then
9. dbms\_output.put\_line('grade is B+');
10. when m>50 and m<=60 then
11. dbms\_output.put\_line('grade is B');
12. when m>30 and m<=50 then
13. dbms\_output.put\_line('grade is C');
14. when m<=30 then
15. dbms\_output.put\_line('grade is D');
16. end case;
17. end;

19 /

Procedure created.

SQL> execute grade(58); grade is B

PL/SQL procedure successfully completed.

**Aim:** Write a Pl/SQL procedure for reversing a number and check whether it is palindrome number or not using while loop in oracle.

# Procedure:

SQL> create or replace procedure palindrome(a number) as

1. i number;
2. rev number:=0;
3. t number;
4. begin 6 t:=a;

7 while t>0 loop 8 i:=mod(t,10);

9 rev:=(rev\*10)+i;

10 t:=floor(t/10);

1. end loop;
2. dbms\_output.put\_line('reverse of '||a||' is '||rev);
3. if rev=a then
4. dbms\_output.put\_line('it is palindrome');
5. else
6. dbms\_output.put\_line('it is not palindrome');
7. end if;
8. end;

19 /

Procedure created.

SQL> execute palindrome(313); reverse of 313 is 313

it is palindrome

**Aim:** Write a Pl/SQL procedure to swap two numbers in oracle.

# Procedure:

SQL> create or replace procedure swap\_t(a in number,b in number) as

1. x number:=a;
2. y number:=b;
3. t number;
4. begin 6 t:=x; 7 x:=y; 8 y:=t;
5. dbms\_output.put\_line('before swap a='||a||' and b='||b);
6. dbms\_output.put\_line('after swap a='||x||' and b='||y);
7. end;

12 /

Procedure created.

SQL> execute swap\_t(23,67); before swap a=23 and b=67 after swap a=67 and b=23

**Aim:** Write a Pl/SQL procedure for printing Armstrong number upto a given limit in oracle.

# Procedure:

SQL> create or replace procedure armstrong(a number) as

1. i number;
2. s number:=0;
3. t number;
4. c number;
5. begin

7 <<loop1>>

8 for i in 0..a loop 9 t:=i;

10 <<loop2>>

11 while t>0 loop 12 c:=mod(t,10); 13 s:=s+c\*c\*c; 14 t:=floor(t/10);

1. end loop loop2;
2. if i=s then
3. dbms\_output.put\_line(s||' is a Armstrong');
4. end if;
5. end loop loop1;
6. end;

21 /

Procedure created.

SQL> execute armstrong(200); 0 is a Armstrong

1 is a Armstrong

**Aim:** Write a Pl/SQL procedure to find greatest of three numbers using else if ladder in oracle.

# Procedure:

SQL> create or replace procedure largestt(a in int,b in int,c in int)

1. as
2. begin
3. if(a>b and a>c) then
4. dbms\_output.put\_line(a||' is the greatest');
5. elsif(b>a and b>c) then
6. dbms\_output.put\_line(b||' is the greatest');
7. else
8. dbms\_output.put\_line(c||' is the greatest');
9. end if;
10. end;

12 /

Procedure created.

SQL> execute largestt(45,58,12); 58 is the greatest

PL/SQL procedure successfully completed.

**Aim:** Write a Pl/SQL procedure to print floyds pattern using for loop in oracle.

# Procedure:

SQL> create or replace procedure floyd(a number) as

1. i number;
2. j number;
3. begin
4. for i in 1..a loop
5. for j in 1..i loop
6. dbms\_output.put(j);
7. end loop;
8. dbms\_output.new\_line;
9. end loop;
10. end;

12 /

Procedure created.

SQL> execute floyd(6); 1

12

123

1234

12345

123456

PL/SQL procedure successfully completed.

**Aim:** Write a Pl/SQL procedure to print pattern of stars using for loop in oracle.

# Procedure:

SQL> create or replace procedure floyd(a number) as

1. i number;
2. j number;
3. begin
4. for i in 1..a loop
5. for j in 1..i loop
6. dbms\_output.put('\*');
7. end loop;
8. dbms\_output.new\_line;
9. end loop;
10. end;

12 /

Procedure created.

SQL> execute floyd(5);

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

PL/SQL procedure successfully completed.

**Aim:** Write a PL/SQL procedure to find the LCM and GCD in oracle

# Procedure:

SQL> create or replace procedure lcmgcd(a in number,b in number) as

1. i number;
2. j number;
3. begin
4. if a>b then 6 i:=a;

7 j:=b;

8 else 9 i:=b; 10 j:=a;

1. end if;
2. while i<=a\*b loop
3. if mod(i,a)=0 and mod(i,b)=0 then
4. dbms\_output.put\_line('lcm is '||i);
5. exit;
6. else 17 i:=i+1;
7. end if;
8. end loop;
9. while j>=1 loop
10. if mod(a,j)=0 and mod(b,j)=0 then
11. exit;
12. else 24 j:=j-1;
13. end if;
14. end loop;
15. dbms\_output.put\_line('hcf is '||j);
16. end;

29 /

Procedure created.

SQL> execute lcmgcd(12,18); lcm is 36

hcf is 6

PL/SQL procedure successfully completed

**Aim:** Write a Pl/SQL procedure to implement IN parameter in oracle.

# Procedure:

SQL> set serveroutput on

SQL> create or replace function adder(n in number,m in number)

1. return number
2. as
3. s number(3);
4. begin

6 s:=n+m;

1. return s;
2. end;

9 /

Function created.

SQL> declare

1. s number(3);
2. begin

4 s:=adder(11,34);

1. dbms\_output.put\_line('addition is: '||s);
2. end;

7 /

addition is: 45

PL/SQL procedure successfully completed.

**Aim:** Write a Pl/SQL procedure to implement OUT parameter in oracle.

# Procedure:

SQL> create or replace procedure out(a in number,s out number) as

1. i number;
2. t number;
3. begin 5 s:=0; 6 t:=a;

7 while t>0 loop 8 i:=mod(t,10); 9 s:=s+i;

10 t:=floor(t/10);

1. end loop;
2. end;

13 /

Procedure created.

SQL>

SQL> declare

1. a number:=234;
2. b number;
3. begin
4. out(a,b);
5. dbms\_output.put\_line('sum digits is '||b);
6. end;

8 /

sum digits is 9

PL/SQL procedure successfully completed

**Aim:** Write a Pl/SQL procedure to implement INOUT parameter in oracle.

# Procedure:

SQL> create or replace procedure out(a in number,s out number) as

1. i number;
2. t number;
3. begin 5 s:=0; 6 t:=a;

7 while t>0 loop 8 i:=mod(t,10); 9 s:=s+i;

10 t:=floor(t/10);

1. end loop;
2. end;

13 /

Procedure created.

SQL>

SQL> declare

1. a number:=234;
2. b number;
3. begin
4. out(a,b);
5. dbms\_output.put\_line('sum digits is '||b);
6. end;

8 /

sum digits is 9

PL/SQL procedure successfully completed

**Aim:** Write a Pl/SQL procedure to implement EXIT in oracle.

# Procedure:

SQL> create or replace procedure tab\_n(a number) as 2 n number;

1. i number:=1;
2. begin
3. loop

6 n:=a\*i;

7 dbms\_output.put\_line(a||'x'||i||'='||n);

8 i:=i+1;

9 exit when i>10; 10 end loop;

11 end;

12 /

Procedure created. SQL> execute tab\_n(3); 3x1=3

3x2=6

3x3=9

3x4=12

3x5=15

3x6=18

3x7=21

3x8=24

3x9=27

3x10=30

PL/SQL procedure successfully completed.

**Aim:** Write a Pl/SQL procedure to implement GOTO in oracle.

# Procedure:

SQL> create or replace procedure i\_goto(b in number) as

1. i number:=0;
2. begin
3. <<loopstart>>
4. while i<=b loop
5. dbms\_output.put\_line('\*'); 7 i:=i+1;
6. goto loopstart;
7. end loop loopstart;
8. end;

11 /

Procedure created.

SQL> execute i\_goto(5);

\*

\*

\*

\*

\*

\*

PL/SQL procedure successfully completed.

**Aim:** Write a Pl/SQL procedure to implement NULL in oracle.

# Procedure:

SQL> create or replace procedure nulln(a in number,b in number) as

1. begin
2. if a>b then
3. dbms\_output.put\_line('greatest number is '||a);
4. elsif b>a then
5. dbms\_output.put\_line('greatest number is '||b);
6. else
7. null;
8. end if;
9. end;

11 /

Procedure created.

SQL> execute nulln(23,67); greatest number is 67

PL/SQL procedure successfully completed.

**Aim:** Write a Pl/SQL function to add two numbers in oracle.

# Procedure:

SQL> create or replace function add\_two(a in number,b in number)

1. return number
2. as
3. s number;
4. begin 6 s:=a+b;
5. return s;
6. end;

9 /

Function created.

SQL> declare

1. c number;
2. begin

4 c:=add\_two(23,45);

1. dbms\_output.put\_line('sum is '||c);
2. end;

7 /

sum is 68

PL/SQL procedure successfully completed.

**Aim:** Write a Pl/SQL function to find the sum of n natural numbers using while loop in oracle.

# Procedure:

SQL> create or replace function s\_of\_nat(num number)

1. return number
2. as
3. s number:=0;
4. i number:=0;
5. begin
6. while i<=num loop 8 s:=s+i;

9 i:=i+1;

1. end loop;
2. return s;
3. end;

13 /

Function created.

SQL> declare

1. a number;
2. begin

4 a:=s\_of\_nat(50);

1. dbms\_output.put\_line('sum is '||a);
2. end;

7 /

sum is 1275

PL/SQL procedure successfully completed.

**Aim:** Write a Pl/SQL function to print the prime number in oracle.

# Procedure:

SQL> create or replace function primen( i in number)

1. return number
2. as
3. n number;
4. m number;
5. c number;
6. begin
7. for n in 2..i loop 9 c:=0;

10 for m in 1..n loop 11 if mod(n,m)=0 then 12 c:=c+1;

1. end if;
2. end loop;
3. if c<=2 then
4. dbms\_output.put\_line(' '||n);
5. end if;
6. end loop;
7. return null;
8. end;

21 /

Function created.

SQL> select primen(50) from dual;

PRIMEN(50)

2

3

5

7

11

13

17

19

23

29

31

37

41

43

47

**Aim:** Write a Pl/SQL function to find factorial of a number using loop in oracle.

# Procedure:

SQL> create or replace function fact\_n(num number)

1. return number
2. as
3. f number:=1;
4. i number:=0;
5. begin
6. loop

8 i:=i+1;

9 f:=f\*i;

1. if i=num then
2. exit;
3. end if;
4. end loop;
5. return f;
6. end;

16 /

Function created.

SQL> declare

1. a number;
2. begin

4 a:=fact\_n(5);

1. dbms\_output.put\_line('factorial is '||a);
2. end;

7 /

factorial is 120.

**Aim:** Write a Pl/SQL function to print multiplication table of a number in oracle.

# Procedure:

SQL> create or replace function mulll(n number)

1. return number
2. as
3. i number;
4. begin
5. for i in 1..10 loop
6. dbms\_output.put\_line(n||'x'||i||'='||(i\*n));
7. end loop;
8. return n\*i;
9. end;

11 /

Function created.

SQL> select mulll(10) from dual; 10x1=10

10x2=20

10x3=30

10x4=40

10x5=50

10x6=60

10x7=70

10x8=80

10x9=90

10x10=100

**Aim:** Write a Pl/SQL function to print first 20 natural numbers using while loop in oracle.

# Procedure:

SQL> create or replace function one\_to\_twenty( b in out int)

1. return int
2. is
3. begin
4. while b<=20 loop
5. dbms\_output.put\_line(''||b); 7 b:=b+1;
6. end loop;
7. return null;
8. end;

11 /

Function created.

SQL> declare

1. a number;
2. begin 4 a:=&a;
3. a:a=a/a;
4. a:=one\_to\_twenty(a);
5. end;

8 /

Enter value for a: 1 old 4: a:=&a;

new 4: a:=1;

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19/

20

PL/SQL procedure successfully completed.

**Aim:** Write a Pl/SQL function to print Fibonacci series in oracle.

# Procedure:

SQL> create or replace function fibonaci\_series(n in number)

1. return number
2. as
3. f number:=0;
4. s number:=1;
5. i number;
6. t number;
7. begin
8. dbms\_output.put\_line('series');
9. for i in 2..n loop 11 t:=f+s;

12 f:=s;

13 s:=t;

1. dbms\_output.put\_line(' '||t);
2. end loop;
3. return null;
4. end;

18 /

Function created.

SQL> select fibonaci\_series(5) from dual; 0

1

2

3

5

**Aim:** Write a Pl/SQL function to print even numbers upto given limits in oracle.

# Procedure:

SQL> create or replace function eventilln(n in number)

1. return int
2. as
3. i number:=0;
4. begin
5. for i in 0..n loop
6. if mod(i,2)=0 then
7. dbms\_output.put\_line(' '||i);
8. else
9. null;
10. end if;
11. end loop;
12. return null;
13. end;

15 /

Function created.

SQL> select eventilln(10) from dual; EVENTILLN(10)

0

2

4

6

8

10

**Aim:** Write a Pl/SQL function to greatest of three among three numbers using else if ladder statement in oracle.

# Procedure:

SQL> create or replace function great\_among\_three(i in number,j in number, k in number)

1. return number
2. as
3. begin
4. if i>=j
5. and i>=k then
6. dbms\_output.put\_line('greatest is: '||i);
7. elsif j>=k
8. and j>=i then
9. dbms\_output.put\_line('greatest is: '||j);
10. else
11. dbms\_output.put\_line('greatest is: '||k);
12. end if;
13. return null;
14. end;

16 /

Function created.

SQL> select great\_among\_three(12,34,1) from dual;

GREAT\_AMONG\_THREE(12,34,1)

greatest is: 34

**Aim:** Write a Pl/SQL function to find the square and cube of a given number in oracle.

# Procedure:

SQL> create or replace function sq\_and\_cu(a in number)

1. return number
2. as
3. sq number:=a\*a;
4. cu number:=a\*a\*a;
5. begin
6. dbms\_output.put\_line('square of number is: '||sq);
7. dbms\_output.put\_line('cube of number is: '||cu);
8. return null;
9. end;

11 /

Function created.

SQL> select sq\_and\_cu(81) from dual;

SQ\_AND\_CU(81)

square of number is: 6561 cube of number is: 531441

**Aim:** Write a Pl/SQL function to implement EXT in oracle.

# Procedure:

SQL> create or replace function exitn(n in int)

1. return int
2. as
3. i number;
4. begin
5. for i in 0..n loop
6. dbms\_output.put\_line(' '||i);
7. exit;
8. end loop;
9. return null;
10. end;

12 /

Function created.

SQL> select exitn(3) from dual;

EXITN(3)

0

**Aim:** Write a Pl/SQL function to implement GOTO in oracle.

# Procedure:

SQL> set serveroutput on

SQL> create or replace function i\_goto( b in int)

1. return int
2. is
3. i number:=0;
4. begin
5. <<loopstart>>
6. while i<=b loop
7. dbms\_output.put\_line('\*'); 9 i:=i+1;
8. goto loopstart;
9. end loop loopstart;
10. return null;
11. end;

14 /

Function created.

SQL> select i\_goto(2) from dual; I\_GOTO(2)

\*

\*

\*

**Aim:** Write a Pl/SQL function to implement NULL in oracle.

# Procedure:

SQL> create or replace function nulln(a in number, b in number)

1. return int
2. as
3. begin
4. if a>b then
5. dbms\_output.put\_line('greatest number is: '||a);
6. elsif b>a then
7. dbms\_output.put\_line('greatest number is: '||b);
8. else
9. null;
10. end if;
11. return null;
12. end;

14 /

Function created.

SQL> select nulln(12,12) from dual;

NULLN(12,12)

**Aim:** Write a PL/SQL program on named exception in Oracle

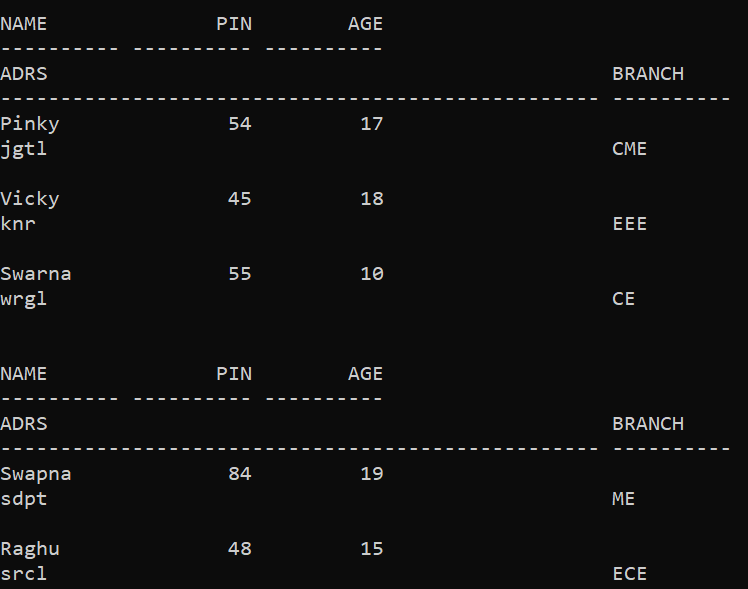
# Procedure:

SQL> create table student(name varchar(10),

1. pin int,
2. age integer,
3. adrs varchar(50),
4. branch varchar2(10) 6 );

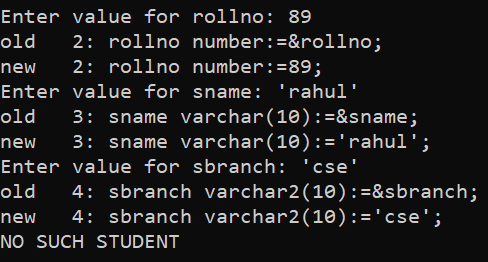
Table created.

SQL> insert into student values('Pinky',54,17,'jgtl','CME'); SQL> insert into student values('Vicky',45,18,'knr','EEE'); SQL> insert into student values('Swarna',55,10,'wrgl','CE'); SQL> insert into student values('Swapna',84,19,'sdpt','ME'); SQL> insert into student values('Raghu',48,15,'srcl''ECE'); SQL> insert into student values('Raghu',48,15,'srcl','ECE'); SQL> select \* from student;



SQL> set serveroutput on SQL> declare

1. rollno number:=&rollno;
2. sname varchar(10):=&sname;
3. sbranch varchar2(10):=&sbranch;
4. begin
5. select pin,name,branch into rollno,sname,sbranch from student where pin=rollno;
6. dbms\_output.put\_line(rollno||''||sname||''||sbranch);
7. exception
8. when no\_data\_found then
9. dbms\_output.put\_line('NO SUCH STUDENT');
10. when others then
11. dbms\_output.put\_line('ERROR');
12. end; 14 /



**Aim:** Write a PL/SQL program on unnamed exception in Oracle

# Procedure:

SQL> DECLARE

1. exp exception;
2. pragma exception\_init (exp, -20015); 4 n int:=10;
3. BEGIN
4. FOR i IN 1..n LOOP
5. dbms\_output.put\_line(i\*i);
6. IF i\*i=36 THEN
7. RAISE exp;
8. END IF;
9. END LOOP;
10. EXCEPTION
11. WHEN exp THEN
12. dbms\_output.put\_line('more than square of 6');
13. END;

16 /

1

4

9

16

25

36

more than square of 6

PL/SQL procedure successfully completed.

**Aim:** Write a PL/SQL program on raise application error in Oracle.

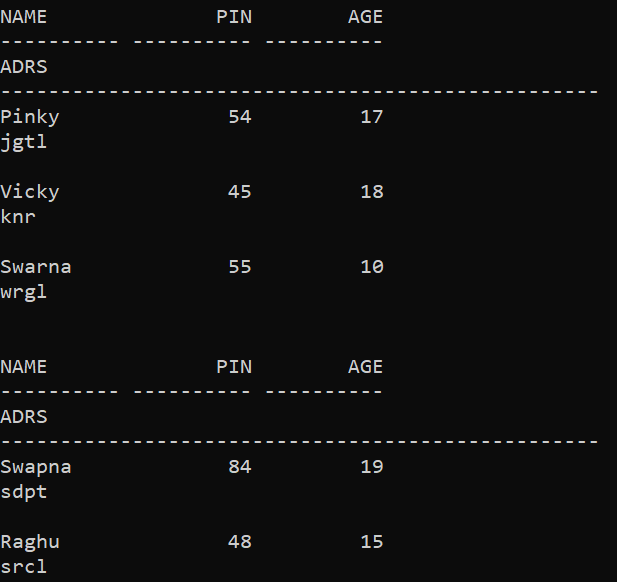
## Procedure:

SQL> create table student(name varchar(10),

1. pin int,
2. age integer,
3. adrs varchar(50) 5 );

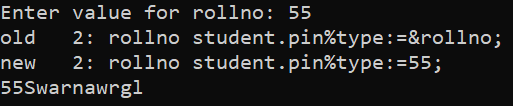
Table created.

SQL> insert into student values('Pinky',54,17,'jgtl'); SQL> insert into student values('Vicky',45,18,'knr'); SQL> insert into student values('Swarna',55,10,'wrgl'); SQL> insert into student values('Swapna',84,19,'sdpt'); SQL> insert into student values('Raghu',48,15,'srcl'); SQL> select \* from student;



SQL> set serveroutput on SQL> declare

1. rollno student.pin%type:=&rollno;
2. sname student.name%type;
3. sadrs student.adrs%type;
4. ex\_invalid\_rollno exception;
5. begin
6. if(rollno<=0) then
7. raise ex\_invalid\_rollno;
8. else
9. select pin,name,adrs into rollno,sname,sadrs from student where pin=rollno;
10. dbms\_output.put\_line(rollno||''||sname||''||sadrs);
11. end if;
12. exception
13. when ex\_invalid\_rollno then
14. dbms\_output.put\_line('ROLLNO MUST BE GREATER THAN 0');
15. when no\_data\_found then
16. dbms\_output.put\_line('NO SUCH STUDENT');
17. when others then
18. dbms\_output.put\_line('ERROR');
19. end; 21 /



**Aim:** Write a PL/SQL procedure on raise application in Oracle.

# Procedure:

SQL> create procedure raisepro(n1 in number:=0,n2 out number)

1. as
2. begin
3. if n1>100 then
4. raise\_application\_error(-20010,'NUMBER IS TOO LARGE');
5. end if;

7 n2:=n1;

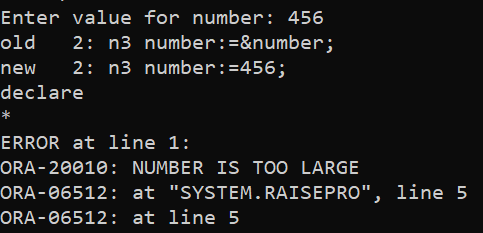
8 end;

9 /

Procedure created.

SQL> declare

1. n3 number:=&number;
2. n4 number:=0;
3. begin
4. raisepro(n3,n4);
5. dbms\_output.put\_line(n4);
6. end; 8 /



**Aim:** Write a PL/SQL program on implicit cursor in Oracle.

# Procedure:

SQL> SET SERVEROUTPUT ON SQL> DECLARE

1. VARROWS NUMBER;
2. BEGIN
3. UPDATE STUDENT SET NAME='LAXMI';
4. IF(SQL%FOUND) THEN
5. VARROWS:=SQL%ROWCOUNT;
6. DBMS\_OUTPUT.PUT\_LINE(VARROWS||'RECORDS ARE UPDATED');
7. ELSE
8. DBMS\_OUTPUT.PUT\_LINE('NO RECORDS UPDATED');
9. END IF;
10. END;

12 /

2RECORDS ARE UPDATED

PL/SQL procedure successfully completed.

**Aim:** Write a PL/SQL program on explicit cursor in Oracle.

# Procedure:

SQL> SET SERVEROUTPUT ON SQL> DECLARE

1. SROLLNO STUDENT.ROLLNO%TYPE;
2. SNAME STUDENT.NAME%TYPE;
3. SADDRESS STUDENT.ADDRESS%TYPE;
4. CURSOR STUDENT\_CURSOR
5. IS
6. SELECT ROLLNO,NAME,ADDRESS INTO SROLLNO,SNAME,SADDRESS FROM STUDENT;
7. BEGIN
8. OPEN STUDENT\_CURSOR;
9. LOOP
10. FETCH STUDENT\_CURSOR INTO SROLLNO,SNAME,SADDRESS;
11. EXIT WHEN STUDENT\_CURSOR%NOTFOUND;
12. DBMS\_OUTPUT.PUT\_LINE(SROLLNO||' '||SNAME||' '||SADDRESS);
13. END LOOP;
14. CLOSE STUDENT\_CURSOR;
15. END;

17 /

1. KOTLANIDHI HYD
2. RAMASRI MOTHKUR

**Aim:** Write a PL/SQL program on creating triggers in Oracle.

# Procedure:

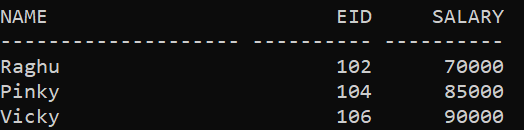
**Step1:**Create a table.

SQL> create table employee(name varchar(20),

* 1. eid int,
  2. salary number);

Table created.

SQL> insert into employee values('Raghu',102,70000); SQL> insert into employee values('Pinky',104,85000); SQL> insert into employee values('Vicky',106,90000); SQL> select \* from employee;



**Step2:**Create a trigger. SQL> set serveroutput on

SQL> create or replace trigger sal\_chng

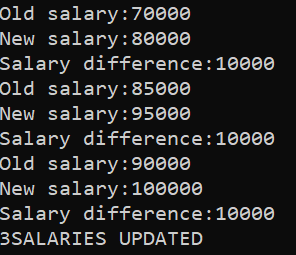
1. before
2. update or delete or insert on employee
3. for each row
4. when(NEW.EID>0)
5. declare
6. saldiff number;
7. begin
8. saldiff:=:NEW.salary-:OLD.salary;
9. dbms\_output.put\_line('Old salary:'||:OLD.salary);
10. dbms\_output.put\_line('New salary:'||:NEW.salary);
11. dbms\_output.put\_line('Salary difference:'||saldiff);
12. end;

14 /

Trigger created.

SQL> set serveroutput on SQL> declare

1. totalrows number;
2. begin
3. update employee set salary=salary+10000;
4. if sql%notfound then
5. dbms\_output.put\_line('NO SALARY UPDATED');
6. ELSIF sql%found then
7. totalrows:=sql%rowcount;
8. dbms\_output.put\_line(totalrows||'SALARIES UPDATED');
9. end if;
10. end; 12 /



PL/SQL procedure successfully completed.

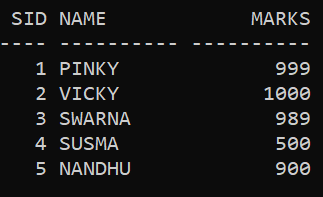
**Aim:** Write a PL/SQL procedure to create package for retrieving info from student table in Oracle.

# Procedure:

SQL> CREATE TABLE STUDENT(

1. SID INT,
2. NAME VARCHAR2(10),
3. marks number);

SQL> INSERT INTO STUDENT VALUES(1,'PINKY',999); SQL> INSERT INTO STUDENT VALUES(2,'VICKY',1000); SQL> INSERT INTO STUDENT VALUES(3,'SWARNA',989); SQL> INSERT INTO STUDENT VALUES(4,'SUSMA',500); SQL> INSERT INTO STUDENT VALUES(5,'NANDHU',900); SQL> SELECT \* FROM STUDENT;



SQL> CREATE PACKAGE STD\_MARKS

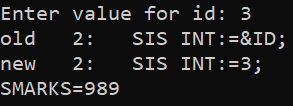
1. AS
2. PROCEDURE FIND\_MARKS(ID INT);
3. END STD\_MARKS; 5 /

Package created.

SQL> CREATE PACKAGE BODY STD\_MARKS

1. AS
2. PROCEDURE FIND\_MARKS(ID INT)
3. IS
4. SMARKS INT;
5. BEGIN
6. SELECT MARKS INTO SMARKS FROM STUDENT WHERE SID=ID;
7. DBMS\_OUTPUT.PUT\_LINE('SMARKS='||SMARKS);
8. END FIND\_MARKS;
9. END STD\_MARKS; 11 /

Package body created.



**Aim:** Write a PL/SQL procedure to create package for retrieving info from employee table in Oracle.

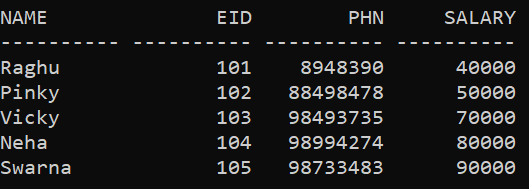
# Procedure:

SQL> create table employee(name varchar(10),

1. eid int,
2. phn number,
3. salary integer);

Table created.

SQL> insert into employee values('Raghu',101,8948390,40000); SQL> insert into employee values('Pinky',102,88498478,50000); SQL> insert into employee values('Vicky',103,98493735,70000); SQL> insert into employee values('Neha',104,98994274,80000); SQL> insert into employee values('Swarna',105,98733483,90000); SQL> select \* from employee;



SQL> create or replace package employ\_sal

1. as
2. procedure find\_salary(id int);
3. end employ\_sal;

5 /

Package created.

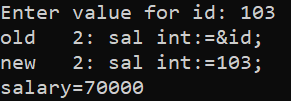
SQL> create or replace package body employ\_sal

1. as
2. procedure find\_salary(id int)
3. is
4. esalary int;
5. begin
6. select salary into esalary from employee
7. where eid=id;
8. dbms\_output.put\_line('salary='||esalary);
9. end find\_salary;
10. end employ\_sal;

12 /

Package body created. SQL> declare

1. sal int:=&id;
2. begin
3. employ\_sal.find\_salary(sal);
4. end; 6 /



PL/SQL procedure successfully completed.

**Aim:** Write a PL/SQL procedure to create package to add, remove and list the customers in customer table in Oracle.

# Procedure:

**Step1:**Create a package.

SQL> CREATE PACKAGE CUST\_SALL

1. AS
2. PROCEDURE FIND\_SAL(C\_ID CUSTOMERS.ID%TYPE);
3. END CUST\_SALL; 5 /

Package created

**Step2:**Create package body. SQL> SET SERVEROUTPUT ON

SQL> CREATE OR REPLACE PACKAGE BODY CUST\_SALL

1. AS
2. PROCEDURE FIND\_SAL(C\_ID CUSTOMERS.ID%TYPE)
3. IS
4. C\_SAL CUSTOMERS.SAL%TYPE;
5. BEGIN
6. SELECT SAL INTO C\_SAL
7. FROM CUSTOMERS
8. WHERE ID=C\_ID;
9. DBMS\_OUTPUT.PUT\_LINE('SALARY:'||C\_SAL);
10. END FIND\_SAL;
11. END CUST\_SALL; 13 /

Package body created. SQL> DECLARE

1. CODE CUSTOMERS.ID%TYPE:=&C\_ID;
2. BEGIN
3. CUST\_SALL.FIND\_SAL(CODE);
4. END;

6 /

Enter value for c\_id: 1

old 2: CODE CUSTOMERS.ID%TYPE:=&C\_ID; new 2: CODE CUSTOMERS.ID%TYPE:=1; SALARY:30000

SQL> SET SERVEROUTPUT ON

SQL> CREATE OR REPLACE PACKAGE C\_PACKAGE

1. AS
2. PROCEDURE ADDCUSTOMER(C\_ID CUSTOMERS.ID%TYPE,
3. C\_NAME CUSTOMERS.NAME%TYPE,
4. C\_AGE CUSTOMERS.AGE%TYPE,
5. C\_ADDRESS CUSTOMERS.ADDRESS%TYPE,
6. C\_SAL CUSTOMERS.SALARY%TYPE);
7. PROCEDURE DELCUSTOMER(C\_ID CUSTOMERS.ID%TYPE);
8. PROCEDURE LISTCUSTOMER;
9. END C\_PACKAGE; 11 /

Package created.

SQL> CREATE OR REPLACE PACKAGE BODY C\_PACKAGE

1. AS
2. PROCEDURE ADDCUSTOMER(C\_ID CUSTOMERS.ID%TYPE,
3. C\_NAME CUSTOMERS.NAME%TYPE,
4. C\_AGE CUSTOMERS.AGE%TYPE,
5. C\_ADDRESS CUSTOMERS.ADDRESS%TYPE,
6. C\_SAL CUSTOMERS.SALARY%TYPE)
7. IS
8. BEGIN
9. INSERT INTO CUSTOMERS(ID,NAME,AGE,ADDRESS,SALARY)VALUES(C\_ID,C\_NAME,C\_A GE,C\_ADDRESS,C\_SAL);
10. END ADDCUSTOMER;
11. PROCEDURE DELCUSTOMER(C\_ID CUSTOMERS.ID%TYPE)
12. IS
13. BEGIN
14. DELETE FROM CUSTOMERS
15. WHERE ID=C\_ID;
16. END DELCUSTOMER;
17. PROCEDURE LISTCUSTOMER IS
18. CURSOR C\_CUSTOMERS IS
19. SELECT NAME FROM CUSTOMERS;
20. TYPE C\_LIST IS TABLE OF CUSTOMERS.NAME%TYPE;
21. NAME\_LIST C\_LIST:=C\_LIST();
22. COUNTER INTEGER:=0;
23. BEGIN
24. FOR N IN C\_CUSTOMERS LOOP
25. COUNTER:=COUNTER+1;
26. NAME\_LIST.EXTEND;
27. NAME\_LIST(COUNTER):=N.NAME; 29

DBMS\_OUTPUT.PUT\_LINE('CUSTOMER('||COUNTER||')'||NAME\_LIST(CO UNTER));

1. END LOOP;
2. END LISTCUSTOMER;
3. END C\_PACKAGE; 33 /

Package body created. SQL> DECLARE

1. CODE CUSTOMERS.ID%TYPE:=8;
2. BEGIN
3. C\_PACKAGE.ADDCUSTOMER(7,'RAJINI',25,'KERALA',3500);
4. C\_PACKAGE.ADDCUSTOMER(8,'HARIKA',32,'DELHI',7500);
5. C\_PACKAGE.LISTCUSTOMER;
6. C\_PACKAGE.DELCUSTOMER(CODE);
7. C\_PACKAGE.LISTCUSTOMER;
8. END;

10 /

CUSTOMER(1)RAMESH CUSTOMER(2)KHILAN CUSTOMER(3)KOUSHIK CUSTOMER(4)CHAITALI CUSTOMER(5)HARDIK CUSTOMER(6)KOMAL CUSTOMER(7)RAJINI CUSTOMER(8)HARIKA CUSTOMER(1)RAMESH CUSTOMER(2)KHILAN CUSTOMER(3)KOUSHIK CUSTOMER(4)CHAITALI CUSTOMER(5)HARDIK

CUSTOMER(6)KOMAL CUSTOMER(7)RAJINI

PL/SQL procedure successfully completed. SQL> SET SERVEROUTPUT ON

SQL> DECLARE

1. SROLLNO STUDENT.ROLLNO%TYPE;
2. SNAME STUDENT.NAME%TYPE;
3. SADDRESS STUDENT.ADDRESS%TYPE;
4. CURSOR STUDENT\_CURSOR
5. IS
6. SELECT ROLLNO,NAME,ADDRESS INTO SROLLNO,SNAME,SADDRESS FROM STUDENT;
7. BEGIN
8. OPEN STUDENT\_CURSOR;
9. LOOP
10. FETCH STUDENT\_CURSOR INTO SROLLNO,SNAME,SADDRESS;
11. EXIT WHEN STUDENT\_CURSOR%NOTFOUND;
12. DBMS\_OUTPUT.PUT\_LINE(SROLLNO||' '||SNAME||' '||SADDRESS);
13. END LOOP;
14. CLOSE STUDENT\_CURSOR;
15. END;

17 /

1. KOTLANIDHI HYD
2. RAMASRI MOTHKUR

**Aim:** Write a PL/SQL function and procedure to create package for inserting values into the table in oracle.

# Procedure:

SQL> create or replace package college

* 1. is
  2. procedure i\_insert(no in employee.pin%type,ename in employee.name%type,esalary in employee.salary%type);
  3. function i\_select(no in employee.pin%type)
  4. return varchar2;
  5. end;

7 /

Package created.

SQL> select \* from employee;

NAME SALARY PIN

|  |  |  |
| --- | --- | --- |
| shivakumar | 10000 | 71 |
| shivasai | 20000 | 110 |
| vinay | 30000 | 12 |
| dsp | 40000 | 47 |

SQL> create or replace package body college

1. is
2. ename varchar(30);
3. esalary number;
4. procedure i\_insert(no in employee.pin%type, ename in employee.name%type,esalary in employee.salary%type)
5. is
6. begin
7. insert into employee(name,salary,pin) values(ename,esalary,no);
8. end i\_insert;
9. function i\_select(no in employee.pin%type)
10. return varchar2
11. is
12. begin
13. select name into ename from employee where pin=no;
14. return ename;
15. end i\_select;
16. end college;

18 /

Package body created.

SQL> begin

1. college.i\_insert(38,'nihal',30000);
2. end;

4 /

PL/SQL procedure successfully completed.

SQL> select \* from employee;

NAME SALARY PIN

|  |  |  |
| --- | --- | --- |
| shivakumar | 10000 | 71 |
| shivasai | 20000 | 110 |
| vinay | 30000 | 12 |
| dsp | 40000 | 47 |
| nihal | 30000 | 38 |

SQL> select college.i\_select(38) from dual; COLLEGE.I\_SELECT(38)

Nihal

**Aim:** Write a PL/SQL function and procedure to create package for exercise on exception handling in oracle.

# Procedure:

SQL> create or replace package collegen

1. is
2. procedure i\_insert(no in employee.pin%type,ename in employee.name%type,esalary in employee.salary%type);
3. function i\_select(no in employee.pin%type)
4. return varchar2;
5. end;

7 /

Package created.

SQL>

SQL> create or replace package body collegen

1. is
2. ename varchar(30);
3. esalary number;
4. procedure i\_insert(no in employee.pin%type, ename in employee.name%type,esalary in employee.salary%type)
5. is
6. begin
7. insert into employee(name,salary,pin) values(ename,esalary,no);
8. end i\_insert;
9. function i\_select(no in employee.pin%type)
10. return varchar2
11. is
12. begin
13. select name into ename from employee where pin=no;
14. exception
15. when no\_data\_found then
16. dbms\_output.put\_line(' no such employee');
17. return ename;
18. end i\_select;
19. end collegen;

21 /

Package body created.

SQL> select collegen.i\_select(14) from dual;

COLLEGEN.I\_SELECT(14)

no such employee

SQL> select \* from employee;

NAME SALARY PIN

vinay 30000 12