INDEX

MYSQL /PL

Exercise 1

---

1. Write a program that computes the perimeter and the area of a rectangle. Define your own values for the length and width. (Assuming that L and W are the length and width of the rectangle, Perimeter = 2*(L+W) and Area = L*W.

Creating Table rectangle :

```
create table rectangle(
   length float(10,2) NOT NULL,
   breadth float(10,2) NOT NULL,
   area float(10,2) NULL,
   perimeter float(10,2) NULL

);
```

Table created :

```
11 •   desc rectangle ;
```

Result Grid | Filter Rows: | Export: | W

Field	Type	Null	Key	Default	Extra
length	float(10,2)	NO		NULL	
breadth	float(10,2)	NO		NULL	
area	float(10,2)	YES		NULL	
perimeter	float(10,2)	YES		NULL	

Inserting values

```
insert into rectangle(length,breadth) values
(10,12),
(34.3, 23.3),
(20.1,45.3) ,
(15,16),
(25,35);
```

Table with inserted values :

```
select * from rectangle;
```

length	breadth	area	perimeter
10.00	12.00	NULL	NULL
34.30	23.30	NULL	NULL
20.10	45.30	NULL	NULL
15.00	16.00	NULL	NULL
25.00	35.00	NULL	NULL

Procedure to add values to table rectangle :

```
CREATE DEFINER=`root`@`localhost` PROCEDURE
`fill_perimeter_and_area`( )
BEGIN
      declare l float default 0.00;
      declare b float default 0.00;

      declare i int default 0;
      declare ar float default 0.0;
      declare pr float default 0.0;
        declare cr cursor for select length , breadth from
rectangle;
      declare continue handler for NOT FOUND  set i=1;
```

```
      open cr;

      while i=0
      do
          fetch cr into l,b;
          set ar= l*b;
          set pr=2*(l+b);
          insert into rectangle(area, perimeter) values (ar,pr);
      end while;
     close cr;
END
```

Final output after calling procedure :

```
call sql_assignment.fill_perimeter_and_area();

select * from rectangle;
```

length	breadth	area	perimeter
10.00	12.00	120.00	44.00
34.30	23.30	799.19	115.20
20.10	45.30	910.53	130.80
15.00	16.00	240.00	62.00
25.00	35.00	875.00	120.00

---

2. Convert a temperature in Fahrenheit (F) to its equivalent in Celsius (C) and vice versa. The required formulae are:- C= (F-32)*5/9 F= 9/5*C + 32

Creating table:

```
create table convert_f_To_C(
   fahrenheit float(10,2),
    celsius float(10,2)
);
```

Table structure created :

```
desc convert_f_to_c;
```

	Field	Type	Null	Key	Default	Extra
▶	fahrenheit	float(10,2)	YES		NULL	
	celsius	float(10,2)	YES		NULL	

Inserting values :

```
insert into convert_f_To_C values
(123,null),
(null,32),
(175,null),
(null, 100);
select * from convert_f_To_C;
```

Table :

fahrenheit	celsius
123.00	NULL
NULL	32.00
175.00	NULL
NULL	100.00

Procedure to convert and fill null values :

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `conert_f_to_c`()
BEGIN
    declare f_val float(10,2) default 0.00;
    declare c_val float(10,2) default 0.00;

    declare i int default 0;
    declare cr cursor for select fahrenheit,celsius from convert_f_to_c;
     declare continue handler for NOT FOUND set i=1;
       open cr;

       while i=0
       do
          fetch cr into f_val , c_val;
          if f_val is null
          then
          set f_val =(9/5*c_val + 32);
          end if;

          if c_val is null
          then
          set c_val=(f_val-32)*5/9  ;
          end if;

           update convert_f_to_c
           set fahrenheit= f_val, celsius=c_val
           where f_val=fahrenheit or c_val = celsius;



       end while;
       close cr;
```

```
END
```

After procedure call | Table looks like :

```
call sql_assignment.conert_f_to_c();
select * from convert_f_To_C;
```

fahrenheit	celsius
123.00	50.56
89.60	32.00
175.00	79.44
212.00	100.00

---

3. Write a program that enables a user to input an integer. The program should then state whether the integer is evenly divisible by 5.

Procedure which takes int as parameter and give output as ‘DIVISIBLE’ or ‘NOT ‘DIVISIBLE’

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `even_or_odd`(in n int)
BEGIN
      if n MOD 5 =0
    then select 'DIVISIBLE' as 'Result';
    else
    select 'NOT DIVISIBLE' as 'Result';
    end if;
END
```

×××

Exercise 2

---

1. Select from any table a number and determine whether it is within a given range (for example, between 1 and 10).

Creating table with column containing numbers

```
create table numbers(
  num int,
  L_limit int ,
  U_limit int,
  is_in_range varchar(10)
);
```

Table structure :

Field	Type	Null	Key	Default	Extra
num	int	YES		NULL	
L_limit	int	YES		NULL	
U_limit	int	YES		NULL	
is_in_range	varchar(10)	YES		NULL	

Inserting values :

```
insert into numbers values
(12,1,10,null),
(20, 1 , 40,null),
(132, 60, 80,null),
(143,50,150,null),
(122, 120, 140,null),
```

```
(30,140, 160,null);
```

Stored function to check if num is in range of U_ limit and L_ limit

```
CREATE DEFINER=`root`@`localhost` FUNCTION `check_within_range`(
L_limit int, U_limit int , num int) RETURNS varchar(255) CHARSET
utf8mb4
    DETERMINISTIC
BEGIN
    declare msg  varchar(255) default '';
    case
    when L_limit >=num and U_limit <=num then set msg= 'TRUE';
    else  set msg ='FALSE';
    end case;

RETURN msg ;
END
```

Stored procedure to update the values and check if its in range b

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `update_in_range`()
BEGIN
    declare n int default 0;
    declare UL int default 0;
    declare LL int default 0;
    declare msg  varchar(255) default '';
    declare i int default 0;
    declare cr cursor for select num , L_limit , U_limit from
numbers;
    declare continue handler for NOT FOUND set i=1;
    open cr;
    while i=0
    do
     fetch cr into n , LL, UL;
```

```
    set msg = check_within_range(LL , UL , n);



    update numbers
    set is_in_range= msg
    where num = n and L_limit = LL and U_limit =UL;
    end while;
    close cr;
    END
```

---

2. Select from any table three positive integers representing the sides of a triangle, and determine whether they form a valid triangle. Hint: In a triangle, the sum of any two sides must always be greater than the third side.

Creating table :

```
create table triangle(
   s1 float,
   s2 float,
   s3 float

);
```

Table structure created :

Field	Type	Null	Key	Default	Extra
s1	float	YES		NULL	
s2	float	YES		NULL	
s3	float	YES		NULL	

Inserting values into table :

```
insert into triangle(s1,s2,s3) values
(10,12,13),
(20,50,90),
(90,10,15),
(10, 20 ,20),
(10,20, 40),
(5,10,15),
(19,10,23);
```

Table :

s1	s2	s3
10	12	13
20	50	90
90	10	15
10	20	20
10	20	40
5	10	15
19	10	23

Function to check if its a triangle

```
CREATE DEFINER=`root`@`localhost` FUNCTION `is_Triangle`(s1 int , s2 int ,
s3 int ) RETURNS varchar(8) CHARSET utf8mb4
    DETERMINISTIC
BEGIN
```

```
        declare msg varchar(8) default '';
        if (s1+s2 >s3) and s2+s3 >s1 and s3+s1>s2
        then set msg = 'TRUE';
        else
        set msg ='FALSE';
        end if;
RETURN msg;
END
```

Check for the table :

```
select s1,s2,s3, is_triangle(s1,s2,s3) as IS_Triangle from triangle;
```

s1	s2	s3	IS_Triangle
10	12	13	TRUE
20	50	90	FALSE
90	10	15	FALSE
10	20	20	TRUE
10	20	40	FALSE
5	10	15	FALSE
19	10	23	TRUE

---

3. Check if a given year is a leap year. The condition is:- year should be (divisible by 4 and not divisible by 100) or (divisible by 4 and divisible by 400.). The year should be Selected from some table.

Creating ‘leapyear ’ table :

```
create table leapyear(
 year int,
 is_Leap_Year varchar(8)
);
```

Structure of leapyear table:

```
desc leapyear;
```

Field	Type	Null	Key	Default	Extra
year	int	YES		NULL	

Inserting values :

```
insert into leapyear(year) values
(2022),
(2000),
(1300),
(2015),
(2021),
(2024);
```

Table :

```
Select * from leapyear;
```

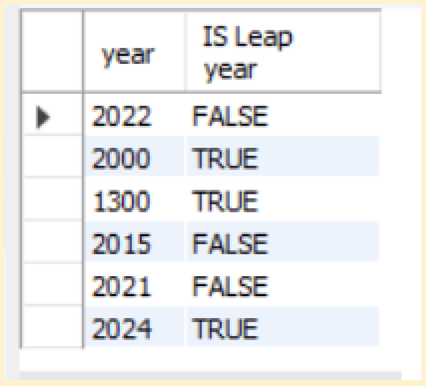
year
2022
2000
1300
2015
2021
2024

Function To check if its leap year :

```
CREATE DEFINER=`root`@`localhost` FUNCTION `check_if_leap_year`(year int)
RETURNS varchar(8) CHARSET utf8mb4
    DETERMINISTIC
BEGIN
   declare msg varchar(10) default '';
   if  (((year /100 )=0 and (year %400)=0) or (year % 4=0))
   then set msg ='TRUE';
   else set msg ='FALSE';
   end if;

RETURN msg;
END
```

OUTPUT :



year	IS Leap year
2022	FALSE
2000	TRUE
1300	TRUE
2015	FALSE
2021	FALSE
2024	TRUE

* * *

Exercise 3

---

1. Write a program containing a loop that iterates from 1 to 1000 using a variable I, which is incremented each time around the loop. The program should output the value of I every hundred iterations (i.e., the output should be 100, 200, etc.).

Creating a procedure and loop with in procedure to iterate from 1 to 1000:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE
`get_every_hundred_iterations`()
BEGIN
   declare i int default 1;
   declare res varchar(255) default '';
   loop1 : Loop
   if i>1000
   then Leave loop1;
   end if;

   if i MOD 100 =0
   then
      set res = concat(res, i, ' ');
   end if;

   set i=i+1;
   end Loop;


   select res;
END
```

Procedure call :

```
call sql_assignment.get_every_hundred_iterations();
```

Output Table :

	res
▶	100 200 300 400 500 600 700 800 900 1000

---

2. Write a program that Selects from any table a minimum and maximum value for a radius, along with an increment factor, and generates a series of radii by repeatedly adding the increment to the minimum until the maximum is reached. For each value of the radius, compute and display the circumference, area, and volume of the sphere. (Be sure to include both the maximum and the minimum values.).

Creating table circle:

```
create table circle (
  max_radus float ,
  min_radius float,
  iterator int
);
```

Table structure :

```
desc circle;
```

Field	Type	Null	Key	Default	Extra
max_radus	float	YES		NULL	
min_radius	float	YES		NULL	
iterator	int	YES		NULL	

Inserting values in table:

```
insert into circle values
(10,40,5);
```

```
select * from circle;
```

min_radius	max_radus	iterator
10	40	5

Procedure to calculate ;

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `cal_CAV`()
BEGIN
     declare area float default 0;
     declare circum float default 0;
     declare vol float default 0;

     -- var to store row wise values
     declare max float default 0;
     declare min float default 0;
     declare itr float default 0;
```

```
        declare i int default 0;
     declare cr cursor for select  max_radus,
min_radius,iterator from circle;
     declare continue handler for NOT FOUND set i=1;
     create table if not exists tempforcircle (
       minimum float ,
       maximum float ,
       iter int ,
       area_of_circle float,
       volume float,
       circum float
     );

     open cr;
     while i=0 or min<= max
     do
      fetch cr into max, min , itr;


     set area = PI()*min*min;
     set circum= 2*PI()*min;
     set vol = 4/3*PI()*pow(min,3);


     insert into tempforcircle values
     (min, max , itr , area , vol, circum);
     set min=min+itr;
     end while;
END
```

---

3. A palindrome is a word that is spelled the same forward and backward, such as level, radar, etc. Write a program to Select from any table a five letter word and determine whether it is a palindrome.

Stored procedure to check if string is palindrome

```
CREATE DEFINER=`root`@`localhost` PROCEDURE
`check_if-pallindrome`(in str varchar(255))
BEGIN
     declare res varchar(255) default 'TRUE';
     declare low int default 1;
     declare high int default length(str);

     loop1: while low <=high
     do
       if NOT(substr(str , low, 1) = substr(str, high, 1))
       then
       set res='FALSE';
       leave loop1;
       end if;
       set low=low+1;
       set high=high-1;
     end while;

     select str, res;

END
```

Test Cases to check if procedure working

Test case 1:

```
call sql_assignment.`check_if-pallindrome`('mom');
```

Output :

	str	res
▶	mom	TRUE

Test case 2 :

```
call sql_assignment.`check_if-pallindrome`('level');
```

Output :

	str	res
▶	level	TRUE

Test case 2 :

```
call sql_assignment.`check_if-pallindrome`('Maria');
```

Output :

	str	res
▶	Maria	FALSE

Exercise 4

---

1. Write a stored function to take three parameters, the sides of a triangle. The sides of the triangle should be accepted from the user. The function should return a Boolean value:- true if the triangle is valid, false otherwise. A triangle is valid if the length of each side is less than the sum of the lengths of the other two sides. Check if the dimensions entered can form a valid triangle.

Creating table :

```
create table triangle(
   s1 float,
   s2 float,
   s3 float

);
```

Table structure created :

Field	Type	Null	Key	Default	Extra
s1	float	YES		NULL	
s2	float	YES		NULL	
s3	float	YES		NULL	

Inserting values into table :

```
insert into triangle(s1,s2,s3) values
(10,12,13),
(20,50,90),
(90,10,15),
(10, 20 ,20),
(10,20, 40),
(5,10,15),
(19,10,23);
```

Table :

s1	s2	s3
10	12	13
20	50	90
90	10	15
10	20	20
10	20	40
5	10	15
19	10	23

Function to check if its a triangle

```
CREATE DEFINER=`root`@`localhost` FUNCTION `is_Triangle`(s1 int , s2 int ,
s3 int ) RETURNS varchar(8) CHARSET utf8mb4
    DETERMINISTIC
BEGIN
```

```
        declare msg varchar(8) default '';
        if (s1+s2 >s3) and s2+s3 >s1 and s3+s1>s2
        then set msg = 'TRUE';
        else
        set msg ='FALSE';
        end if;
RETURN msg;
END
```

Check for the table :

```
select s1,s2,s3, is_triangle(s1,s2,s3) as IS_Triangle from triangle;
```

s1	s2	s3	IS_Triangle
10	12	13	TRUE
20	50	90	FALSE
90	10	15	FALSE
10	20	20	TRUE
10	20	40	FALSE
5	10	15	FALSE
19	10	23	TRUE

---

2. Write a function that generates a random number between 1 and 10. Use any logic of your choice to achieve this.

Query to generate random number :

```
select lpad((floor(rand()*(10-1)+1)),25,' ') as 'Rand 1 To 10';
```

Output :

Rand 1 To 10
7

---

3. Create a function that accepts a string of n characters and exchanges the first character with the last, the second with the next – to – last, and so forth until n exchanges have been made. What will the final string look like? Write the function to verify your conclusion

---

Creating a function → reverse string

```
CREATE DEFINER=`root`@`localhost` FUNCTION `reverse_a_string`(str
varchar(255)) RETURNS varchar(255) CHARSET utf8mb4
    DETERMINISTIC
BEGIN
        declare low int default 1;
        declare high int default length(str);
        declare rev varchar(255) default '';

        while low<=high
        do
                  set rev = concat(substr(str,low,1),rev );
```

```
            set low=low+1;
        end while;
RETURN rev;
END
```

Function call 1:

```
set @str='Saurabh';
select @str as 'original str', sql_assignment.reverse_a_string(@str) as
'reversed string';
```

Output :

	original str	reversed string
▶	Saurabh	hbaruaS

Function call 2:

```
set @str='John';
select @str as 'original str', sql_assignment.reverse_a_string(@str) as
'reversed string';
```

Output :

	original str	reversed string
▶	John	nhoJ

Exercise 5

---

1. Write a stored procedure by the name of Comp_intr to calculate the amount of interest on a bank account that compounds interest yearly. The formula is:- I = p (1+ r) y – p where:- I is the total interest earned. p is the principal. r is the rate of interest as a decimal less than 1, and y is the number of years the money is earning interest. Your stored procedure should accept the values of p, r and y as parameters and insert the Interest and Total amount into tempp table.

Procedure to calculate interest and total amount and insert it into temp table

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `comp_intr`(in p float,in r
float, in y int)
BEGIN


    declare intr float default 0;
    declare tot_amt float default 0;
     create table if not exists temp(
      total float,
      interest float
     );

    select p*(1+r),y,  (p*POW((1+r), y));

    set tot_amt= p*POW((1+r), y);
    set intr=tot_amt-p;

    select tot_amt , intr;
    insert into temp values
    (tot_amt,intr );


END
```

Procedure call :

```
call comp_intr(6000, 0.09, 2);
select * from temp;
```

Output :

	total	interest
▶	7128.6	1128.6

—----------

2. Create a stored function by the name of Age_calc. Your stored function should accept the date of birth of a person as a parameter. The stored function should calculate the age of the 50 Sameer Dehadrai person in years. The stored function should return the age in years.

—----------

Procedure which date of birth as param and returns age :-

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `Age-calc`(in dob date, out age
int)
BEGIN
      declare today_date date default now();
    declare total_days int ;


    set  total_days = datediff(today_date,dob);
   set age= date_format(from_days(total_days), '%Y');

END
```

Procedure call :

```
set @age = 0;
call sql_assignment.`Age-calc`('2001-02-26',@age);
select @age as Age;
```

Output :

	Age
▶	22

×××