```
Drop  procedure sp_checkevenodd;
Delimiter $$

Create procedure sp_checkevenodd(num int)
Begin
If mod(num,2)=0 then
Select concat(num,' is an even number');
Else
Select concat(num,' is an odd number');
End if;
end;
$$
Delimiter ;


Call sp_checkevenodd(13);


Select * from information_schema.routines where routine_schema='vita';


Drop  procedure sp_fact;

-- while loop
-- factorial
delimiter $$
create procedure sp_fact(num int)
begin
declare i int;
declare fact bigint default 1;
set i=num;
while(i>=2) do
set fact=fact*i;
set i=i-1;
end while;
select fact;
end;
$$
delimiter ;

call sp_fact(5)
```

– write a procedure that takes input a number and returns whether the number is prime or not

```
Delimiter $$
```

```
Create procedure sp_checkprime(num int)
Begin
Declare i int default 2;
Declare flag int default 0;
while(i<=num/2 and flag=0) do
If mod(num,i)=0 then
Select concat(num ,' is not a prime number');
Set flag=1;
End if;
Set i=i+1;
End while;
If flag=0 then
Select concat(num ,' is  a prime number');
End if;
end;
$$
Delimiter ;

Call sp_checkprime(13);



Drop procedure sp_checkprime;

Delimiter $$
Create procedure sp_checkprime(num int)
Begin
Declare i int default 2;
Declare flag int default 0;
Myloop: loop
If i>num/2 and flag=0 then
Leave myloop;
End if;
If mod(num,i)=0 then
Select concat(num ,' is not a prime number');
Set flag=1;
End if;
Set i=i+1;
End loop;
If flag=0 then
Select concat(num ,' is  a prime number');
End if;
end;
$$
Delimiter ;

Call sp_checkprime(13);
```

```
Delimiter $$
Create procedure sp_checkcoprime(num1 int,num2 int)
Begin
Declare i int default 2;
Declare flag int default 0;
while(i<=num1 and i<num2 and flag=0) do
If mod(num1,i)=0 and mod(num2,i)=0  then
Select concat(num1,' and ' , num2  ,' are not co-prime numbers');
Set flag=1;
End if;
Set i=i+1;
End while;
If flag=0 then
Select concat(num1,' and ' , num2  ,' are co-prime numbers');
End if;
end;
$$
Delimiter ;
Call sp_checkcoprime(5,10);

Drop procedure sp_getdept;

Delimiter $$
Create procedure sp_getdept(v_eid int)
begin
Declare v_dept varchar(100);
Select deptname into v_dept from emp join dept on dept.deptid=emp.deptid
Where eid=v_eid;
Select v_dept;
End;
$$
Delimiter ;


Call sp_getdept(1);




Delimiter $$
Create procedure sp_getdept(v_eid int,  out  v_dept varchar(100))
begin
```

```
Select deptname into v_dept from emp join dept on dept.deptid=emp.deptid
Where eid=v_eid;
End;
$$
Delimiter ;
```

```
Call sp_getdept(2,@dept);
```

```
Delimiter $$
Create procedure sp_call(v_eid int)
Begin
Declare v_dept varchar(100);
Call sp_getdept(v_eid,v_dept);
Select v_dept;
End;
$$
Delimiter ;
```

```
Call sp_call(2);
```

Create a combination of built in functions to take input a number
And return the closest multiple of 5 to that number

47->45
49->50

```
Select round(47/5)*5
```

```
Select substring('abcdef',2,3);

Select substring('abcdef',2);

Select substring('abcdef',-2);

Select instr('abcdef','bc');

Select instr('abcdef','bcz');
```

```
Select replace('adddaaaa','a','z');


Select ltrim('      sdsdsd   ');
Select rtrim('      sdsdsd   ');
Select trim('      sdsdsd   ');
```

1. Find the position of the 1st occurrence of old str
2. Take the characters occurring before that position
3. Append the new str with the string taken from step 2
4. Add the length of the old string to the position taken from step 1
5. Get characters from the input string from the position got from step 4
6. Append the characters got from step 5 to characters from step 3

```
Set @str='abadecdeadedaaadebbdd';
Set @oldstr='ade';
Set @newstr='xyz';

Step 1
Select instr(@str,@oldstr);

Step 2
Select substring(@str,1,instr(@str,@oldstr)-1);


Step 3
Select concat(substring(@str,1,instr(@str,@oldstr)-1),@newstr);

Step 4
Select instr(@str,@oldstr)+length(@oldstr);;

step 5 & 6
Select concat(substring(@str,1,instr(@str,@oldstr)-1),@newstr,substring(@str,
instr(@str,@oldstr)+length(@oldstr)));


'abxyzcdeadedaaadebbdd';
'abxyzcdeadedaaadebbdd';
'

 select datediff(curdate(),'2021-10-31');
```

```
Delimiter $$

Create function sf_checkevenodd(num int)
Returns varchar(100)
deterministic
Begin
If mod(num,2)=0 then
return( concat(num,' is an even number'));
Else
return( concat(num,' is an odd number'));
End if;
end;
$$
Delimiter ;

Select eid,sf_checkevenodd(eid) from emp;


Delimiter $$
Create function sp_checkcoprime(num1 int,num2 int)
Returns varchar(100)
deterministic
Begin
Declare i int default 2;
while(i<=num1 and i<=num2 ) do
If mod(num1,i)=0 and mod(num2,i)=0  then
return(concat(num1,' and ' , num2  ,' are not co-prime numbers'));
End if;
Set i=i+1;
End while;
 return(concat(num1,' and ' , num2  ,' are co-prime numbers'));
end;
$$
Delimiter ;







drop function sf_countwords;
delimiter $$
create function sf_countwords(str text)
returns int
deterministic
```

```
begin
declare i int default 1;
declare cnt int default 1;
declare tmp_str text default trim(str);
while(i<=length(tmp_str))
do
if substring(tmp_str,i,1)=' ' and substring(tmp_str,i+1,1)<>' '  then
set cnt=cnt+1;
end if;
set i=i+1;
end while;
if trim(str)='' then
set cnt=0;
end if;
return(cnt);
end;
$$
delimiter ;

select sf_countwords('this is test');
```

```
-- write a function that takes input a department name
-- and returns the name of all the employees in that department
-- as a comma separated list
HR
a1,a4,a6,a9
```

```
drop  function sf_elist;
delimiter $$
create function sf_elist(v_dept varchar(100))
returns text
READS SQL DATA
begin
declare i int default 0;
declare elist text default '';
declare v_ename varchar(100);
myloop:loop
select ename into v_ename from emp , dept where emp.deptid=dept.deptid
and deptname=v_dept order by ename limit 1 offset i;
if v_ename is null then
leave myloop;
```

```
end if;
set elist=concat(elist,',',v_ename);
set v_ename=null;
set i=i+1;
end loop myloop;
return(substring(elist,2));
end;
$$
delimiter ;



select sf_elist('HR');


select ename from emp , dept where emp.deptid=dept.deptid
and deptname='HR' order by ename limit 1 offset 200;
```

51. Write a query that lists each order number followed by the name
of the customer who made
that order.

```
select onum,cname from
orders o ,customers c
where o.cnum=c.cnum;
```

52. Write 2 queries that select all salespeople (by name and number) who have customers in their
cities who they do not service, one using a join and one a corelated subquery. Which solution
is more elegant?

```
-- join is better
select distinct s.sname,s.snum,s.city
from salespeople s , customers c
where s.snum<>c.snum
and s.city=c.city;
```

```
select s.sname,s.snum,s.city
from salespeople s
where exists(select 1 from customers c
where s.snum<>c.snum
and s.city=c.city);
```

53. Write a query that selects all customers whose ratings are equal to or greater than ANY (in the SQL sense) of Serres'?

```
select * from customers where rating >=any(select rating from
salespeople s, customers c
where s.snum=c.snum
and sname='Serres');
```

54. Write 2 queries that will produce all orders taken on October 3 or October 4.

```
select * from orders where odate in ('1996-10-03','1996-10-04');

select * from orders where odate = '1996-10-03' or odate='1996-10-04';
```

55. Write a query that produces all pairs of orders by a given customer. Name that customer and eliminate duplicates.

```
select cname,o1.onum,o2.onum from orders o1 ,orders o2, customers c
where o1.cnum=o2.cnum
and o1.cnum=c.cnum
and o2.cnum=c.cnum
and o1.onum<o2.onum;
```

56. Find only those customers whose ratings are higher than every customer in Rome.

```
Select * from customers where rating>all(
Select rating from customers where city='Rome');

Select * from customers where rating>(
Select max(rating) from customers where city='Rome');
```

57. Write a query on the Customers table whose output will exclude all customers with a rating <=
100.00, unless they are located in Rome.

```
Select * from customers where rating>100 or city='Rome';
```

58. Find all rows from the Customers table for which the salesperson number is 1001.

```
Select * from customers where snum=1001;
```

59. Find the total amount in Orders for each salesperson for whom this total is greater than the
amount of the largest order in the table.

```
Select sname,sum(amt) from
Orders o , salespeople s
Where o.snum=s.snum
Group by sname
Having sum(amt)>(select max(amt) from orders);
```

60. Write a query that selects all orders save those with zeroes or NULLs in the amount field.

61. Produce all combinations of salespeople and customer names such that the former precedes
the latter alphabetically, and the latter has a rating of less than 200.

```
Select sname,cname
From salespeople,customers
Where sname<cname and rating<200;
```

62. List all Salespeople’s names and the Commission they have earned.

```
Select sname,case when sum(amt*comm) is null then 0 else  sum(amt*comm) end
total_comm
From orders o right join salespeople s
On o.snum=s.snum
```

Group by sname;

63. Write a query that produces the names and cities of all customers with the same rating as
Hoffman. Write the query using Hoffman's CNUM rather than his rating, so that it would still be
usable if his rating changed.

Select cname,city from customers
Where rating=(select rating from customers where cname='Hoffman');

64. Find all salespeople for whom there are customers that follow them in alphabetical order.

Select sname,cname
From salespeople s , customers c
Where s.snum=c.snum
And sname<cname;

65. Write a query that produces the names and ratings of all customers of all who have above
average orders.

Select cname, rating from customers c, orders o
Where c.cnum=o.cnum
And amt>(select avg(amt) from orders);

66. Find the SUM of all purchases from the Orders table.
Select sum(amt) from orders;

67. Write a SELECT command that produces the order number, amount and date for all rows in
the order table.

Select onum,amt,odate from orders;

68. Count the number of nonNULL rating fields in the Customers table (including repeats).

Select count(rating) from customers;

69. Write a query that gives the names of both the salesperson and the customer for each order
after the order number.

```
Select onum,sname,cname
From
Orders o , customers c , salespeople s
Where o.snum=s.snum
And o.cnum=c.cnum
And c.snum=s.snum;
```

70. List the commissions of all salespeople servicing customers in London.

```
Select sname,case when sum(amt*comm) is null then 0 else  sum(amt*comm) end
total_comm
From orders o, salespeople s, customers c
Where o.snum=s.snum
And o.cnum=c.cnum
And c.snum=s.snum
And c.city='London'
Group by sname;
```

71. Write a query using ANY or ALL that will find all salespeople who have no customers located in
their city.

```
Select * from salespeople s
Where city!=all(select city from customers c where c.snum=s.snum);
```

72. Write a query using the EXISTS operator that selects all salespeople with customers located in
their cities who are not assigned to them.

```
Select *
From salespeople s
Where exists (select 1 from customers c
Where c.city=s.city and c.snum<>s.snum);
```

73. Write a query that selects all customers serviced by Peel or Motika. (Hint : The SNUM field
relates the two tables to one another.)

```
Select cname,sname
From customers c , salespeople s
Where c.snum=s.snum
And sname in ('Peel','Motika');
```

74. Count the number of salespeople registering orders for each day. (If a salesperson has more
than one order on a given day, he or she should be counted only once.)

```
Select odate, count(distinct snum) from orders group by odate;
```

75. Find all orders attributed to salespeople in London.

```
Select odate,onum,amt
From orders c , salespeople s
Where c.snum=s.snum
And city='London';
```

76. Find all orders by customers not located in the same cities as their salespeople.

```
Select odate,onum,amt,cname
From orders o , salespeople s, customers c
Where c.snum=s.snum
And o.snum=s.snum
And o.cnum=c.cnum
And c.city<>s.city;
```

77. Find all salespeople who have customers with more than one current order.

```
Select * from salespeople
Where snum in
(select snum from customers
Where cnum in (select cnum from orders
Group by cnum having count(onum)>1));
```

78. Write a query that extracts from the Customers table every customer assigned to a salesperson who currently has at least one other customer (besides the customer being selected) with orders in the Orders table.

```
Select * from customers where snum in(Select snum from orders
Group by snum
Having count(distinct cnum)>1);
```

79. Write a query that selects all customers whose names begin with ‘C’.

```
Select * from customers where cname like 'c%';
```

80. Write a query on the Customers table that will find the highest rating in each city. Put the output
in this form : for the city (city) the highest rating is : (rating).

```
Select concat( 'for the city ',city,' the highest rating is : ',max(rating) )
From customers
Group by city;
```

81. Write a query that will produce the SNUM values of all salespeople with orders currently in the
Orders table (without any repeats).

82. Write a query that lists customers in descending order of rating. Output the rating field first,
followed by the customer’s names and numbers.

83. Find the average commission for salespeople in London.

84. Find all orders credited to the same salesperson who services Hoffman (CNUM 2001).

85. Find all salespeople whose commission is in between 0.10 and 0.12 (both inclusive).

86. Write a query that will give you the names and cities of all salespeople in London with a commission above 0.10.

87. What will be the output from the following query?

```
 SELECT * FROM ORDERS
 where (amt < 1000 OR NOT (odate = '1996-10-03' AND cnum >
2003));
```

88. Write a query that selects each customer's smallest order.

89. Write a query that selects the first customer in alphabetical order whose name begins with G.
90. Write a query that counts the number of different nonNULL city values in the Customers table.
91. Find the average amount from the Orders table.
92. What would be the output from the following query?

```
SELECT * FROM ORDERS
WHERE NOT (odate = 10/03/96 OR snum > 1006) AND amt >=
1500);
```

93. Find all customers who are not located in San Jose and whose rating is above 200.
94. Give a simpler way to write this query :

```
SELECT snum, sname city, comm FROM salespeople
WHERE (comm > + 0.12 OR comm < 0.14);
```

```
SELECT snum, sname city, comm FROM salespeople
WHERE (comm > + 0.12 OR comm < 0.14);

SELECT snum, sname city, comm FROM salespeople where comm is not null;
```

95. Evaluate the following query :

```
SELECT * FROM orders
WHERE NOT ((odate = 10/03/96 AND snum > 1002) OR amt > 2000.00);
```

96. Which salespersons attend to customers not in the city they have been assigned to?
97. Which salespeople get commission greater than 0.11 are serving customers rated less than
250?
98. Which salespeople have been assigned to the same city but get different commission percentages?

```
Select s1.sname,s2.sname,s1.comm,s2.comm,s1.city
From salespeople s1, salespeople s2
Where s1.city=s2.city
And
s1.comm<>s2.comm
And
s1.snum<s2.snum;
```

99. Which salesperson has earned the most by way of commission?
100.Does the customer who has placed the maximum number of orders have the maximum rating?

```
Select ord.cnum, case when ord.cnum=cust_rat.cnum then 'Yes' else 'No' end Result from
(Select * from (Select cnum,rank() over(order by cust_order_count desc) rn from (Select
cnum ,count(onum) cust_order_count from orders group by cnum) as t ) as t1
Where rn=1) ord left join
(select * from (select cnum,rating, rank() over (order by rating desc) rn from customers ) as
c1 where rn=1) cust_rat
On ord.cnum=cust_rat.cnum
;
```

Assignment Day 4

1. Revise all slides
2. Revise Isolation Levels, Normalization, Transaction, ACID, Data Models
3. Revise Day 1 to Day 4 queries
4. Do all 125 queries from case study
5. Complete the group project
6. Try to do hierarchical query