Write a recursive query to return factorial of a given number

With recursive t

As

(select 5 num, 5 as fact

Union all

Select num-1 num,fact\*(num-1)

From t where num>2

)

Select max(fact) factorial from t;

| Parts | |
| --- | --- |
| partid | ParentID |
| A | A |
| B | A |
| C | B |
| D | C |
| E | E |
| F | E |
| G | F |
| H | F |
| I | I |
| J | I |
| K | J |

| PartID | TopMostParentID |
| --- | --- |
| A | A |
| B | A |
| C | A |
| D | A |
| E | E |
| F | E |
| G | E |
| H | E |
| I | I |
| J | I |
| K | I |

Create table parts (partid varchar(1), parentid varchar(10));

insert into parts values('A','A');

insert into parts values('B','A');

insert into parts values('C','B');

insert into parts values('D','C');

insert into parts values('E','E');

insert into parts values('F','E');

insert into parts values('G','F');

insert into parts values('H','F');

insert into parts values('I','I');

insert into parts values('J','I');

insert into parts values('K','J');

With recursive t as(

Select partid originalpartid , parentid immediatepartid, parentid topmostparentid,1 level from parts

Union all

Select

originalpartid ,immediatepartid,p.parentid,level+1

From

T , parts p

Where t.topmostparentid=p.partid

And p.partid<>p.parentid)

Select originalpartid,topmostparentid from (Select \*,row\_number() over(partition by originalpartid order by level desc ) rn from t) as t2 where rn=1;

Drop procedure sp\_loademp;

Delimiter $$

Create procedure sp\_loademp(rec int)

Begin

Declare i int;

Set i=1;

Truncate table emp;

Start transaction;

while(i<=rec)

Do

Insert into emp(eid,ename,salary , deptid) values

(i,concat(convert(char(mod((i-1),26)+97), char(1)),i),

rand()\*100000,mod((i-1),3)+1);

If mod(i,5000)=0 then

Commit;

Start transaction;

End if;

Set i=i+1;

End while;

commit;

End;

$$

Delimiter ;

call sp\_loademp(2000000);

Delimiter $$

Create procedure sp\_getename()

Begin

Declare flag int default 0;

Declare v\_ename varchar(100);

Declare c1 cursor for

Select ename from emp;

Declare continue handler for not found set flag=1;

Open c1;

Myloop:loop

Fetch c1 into v\_ename;

If flag=1 then

Leave myloop;

Else

Select v\_ename;

End if;

End loop myloop;

Close c1;

End;

$$

Delimiter ;

Delimiter $$

Create function sf\_getelist\_cur(v\_dept varchar(100))

Returns mediumtext

Reads sql data

Begin

Declare v\_ename varchar(100);

Declare elist mediumtext default '';

Declare flag int default 0;

Declare c1 cursor for

Select ename from emp join dept on emp.deptid=dept.deptid

Where deptname=v\_dept;

Declare continue handler for not found set flag=1;

Open c1;

Myloop:loop

Fetch c1 into v\_ename;

If flag=1 then

Leave myloop;

Else

Set elist=concat(elist,',',v\_ename);

End if;

End loop myloop;

return(substring(elist,2));

Close c1;

End;

$$

Delimiter ;

Drop table emp\_audit;

Create table emp\_audit

(command varchar(10),

user\_name varchar(100),

Upd\_dt datetime,

Old\_eid int,

New\_eid int,

Old\_ename varchar(100),

New\_ename varchar(100),

Old\_salary int,

New\_salary int,

Old\_deptid int,

New\_deptid int);

Delimiter $$

Create trigger trigg\_emp\_upd

Before update on emp for each row

Begin

Insert into emp\_audit

(command,

user\_name,

Upd\_dt,

Old\_eid,

New\_eid,

Old\_ename,

New\_ename,

Old\_salary,

New\_salary,

Old\_deptid,

New\_deptid

)

Values

('UPDATE',

user(),

now(),old.eid,new.eid

,old.ename,new.ename

,old.salary,new.salary

,old.deptid,new.deptid);

End;

$$

Delimiter ;

Delimiter $$

Create trigger trigg\_emp\_del

Before delete on emp for each row

Begin

Insert into emp\_audit

(command,

user\_name,

Upd\_dt,

Old\_eid,

New\_eid,

Old\_ename,

New\_ename,

Old\_salary,

New\_salary,

Old\_deptid,

New\_deptid

)

Values

('DELETE' ,

user(),

now(),old.eid,null

,old.ename,null

,old.salary,null

,old.deptid,null);

End;

$$

Delimiter ;

Delimiter $$

Create trigger trigg\_emp\_ins

Before INSERT on emp for each row

Begin

Insert into emp\_audit

(command,

user\_name,

Upd\_dt,

Old\_eid,

New\_eid,

Old\_ename,

New\_ename,

Old\_salary,

New\_salary,

Old\_deptid,

New\_deptid

)

Values

('INSERT ' ,

user(),

now(),null,new.eid

,null,new.ename,

Null,new.salary,null,

new.deptid);

End;

$$

Delimiter ;

31. Find all salespeople whose name starts with ‘P’ and the fourth character is ‘l’.

Select \* from salespeople where sname like 'p\_\_l%';

32. Write a query that uses a subquery to obtain all orders for the customer named Cisneros.

Assume you do not know his customer number.

Select \* from orders where cnum in (select cnum

From customers where cname='Cisneros');

33. Find the largest orders for Serres and Rifkin.

Select sname,max(amt)

From orders o , salespeople s

Where o.snum=s.snum

And sname in ('Serres','Rifkin')

Group by sname;

34. Extract the Salespeople table in the following order : SNUM, SNAME, COMMISSION, CITY.

Select SNUM, SNAME, comm COMMISSION, CITY

From salespeople;

35. Select all customers whose names fall in between ‘A’ and ‘G’ alphabetical range.

Select \* from customers where left(cname,1) between 'a' and 'g';

36. Select all the possible combinations of customers that you can assign.

Select c1.cname,c2.cname from customers c1, customers c2

Where c1.cnum<c2.cnum;

37. Select all orders that are greater than the average for October 4.

Select \* from orders

Where amt>(select avg(amt) from orders where odate='1996-10-04');

38. Write a select command using a corelated subquery that selects the names and numbers of all

customers with ratings equal to the maximum for their city.

Select cname,cnum from customers co

Where exists

(select 1 from customers ci

Where co.city=ci.city

Group by city having max(ci.rating)=co.rating );

39. Write a query that totals the orders for each day and places the results in descending order.

Select odate,sum(amt) from orders group by odate

Order by sum(amt) desc;

40. Write a select command that produces the rating followed by the name of each customer in

San Jose.

Select rating,cname from customers where city='San Jose';

41. Find all orders with amounts smaller than any amount for a customer in San Jose.

select \* from orders where amt<any

(select amt from orders o , customers c

Where o.cnum=c.cnum and city='San Jose');

select \* from orders where amt<

(select max(amt) from orders o , customers c

Where o.cnum=c.cnum and city='San Jose');

42. Find all orders with above average amounts for their customers.

Select \* from orders

Where amt>(select avg(amt) from orders);

43. Write a query that selects the highest rating in each city.

Select city,max(rating) from customers group by city;

44. Write a query that calculates the amount of the salesperson’s commission on each order by a

customer with a rating above 100.00.

Select sname,odate,onum,amt,amt\*comm commission

From orders o , salespeople s , customers c

Where o.snum=s.snum

And o.cnum=c.cnum

And s.snum=c.snum

And rating>100;

45. Count the customers with ratings above San Jose’s average.

Select count(cnum) from customers

Where rating>(select avg(rating) from customers where city='San Jose');

46. Write a query that produces all pairs of salespeople with themselves as well as duplicate rows

with the order reversed.

Select s1.sname s1,s2.sname

From salespeople s1, salespeople s2;

47. Find all salespeople that are located in either Barcelona or London.

Select \* from salespeople where city in ('Barcelona','London');

48. Find all salespeople with only one customer.

Select \* from salespeople

Where snum in

(select snum from customers group by snum having count(distinct cnum)=1);

49. Write a query that joins the Customer table to itself to find all pairs of customers served by a

single salesperson.

Select c1.cname,c2.cname,

c1.snum,c2.snum

from customers c1, customers c2

Where c1.cnum<c2.cnum

And c1.snum=c2.snum;

50. Write a query that will give you all orders for more than $1000.00

Select \* from orders where amt>1000;

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?

Select distinct sname,s.snum from salespeople s, customers c

Where s.snum<>c.snum

And s.city=c.city;

Select sname,s.snum 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 customers c, salespeople s

Where c.snum=s.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 o1.onum,o2.onum,

C1.cname

from orders o1, orders o2, customers c1

Where o1

.onum<o2.onum

And o1.cnum=c1.cnum

And o2.cnum=c1.cnum;

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');

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.

Select \* from orders where amt=0 or amt is null;