**City**

**Ans1.**

**Create table city**

**(**

**ID number,**

**Name varchar2(17),**

**Country code varchar2(3),**

**District varchar2(20),**

**Population number**

**)**

**\***all American cities in the CITY table with populations larger than 100000

select id, name, countrycode, district,population from city where countrycode='USA'and

population>100000 ;

ID NAME COUNTRYCODE DISTRICT POPULATION

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| * + 3815 | * El Paso | * USA | * Texas | * 563662 |
| * + 3878 | * Scottsdale | * USA | * Arizona | * 202705 |
| * + 3965 | * Corona | * USA | * California | * 124966 |
| * + 3973 | * Concord | * USA | * California | * 121780 |
| * + 3977 | * CedarRapids | * USA | * Iowa | * 120758 |
| * + 3982 | * Coral Springs | * USA | * Florida | * 117549 |

Ans 2. select id, name, countrycode, district,population from city where countrycode='USA'and

Population>120000

ID NAME COUNTRYCODE DISTRICT POPULATION

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| * 3815 | * El Paso | * USA | * Texas | * 563662 |
| * 3878 | * Scottsdale | * USA | * Arizona | * 202705 |
| * 3965 | * Corona | * USA | * California | * 124966 |
| * 3973 | * Concord | * USA | * California | * 121780 |
| * 3977 | * CedarRapids | * USA | * Iowa | * 120758 |

Ans 3. Select \* from city

Ans 4. select id, name, countrycode, district,population from city where id = 1661.

Ans 5. select id, name, countrycode, district,population from city where countrycode = 'JPN'.

Ans 6. select name from city where countrycode = 'JPN'.

Ans 7. select city,state from station.

;

Ans8. SELECT DISTINCT CITY FROM STATION WHERE MOD(ID,2)=0 ORDER BY CITY ASC;

Ans9. SELECT COUNT(CITY) - COUNT(DISTINCT CITY) FROM STATION;

Ans10. select city, length(city) from station order by length(city) DESC,city ASC fetch first row only;

select city, length(city) from station order by length(city) asc ,city asc fetch first row only;

Ans 11. SELECT DISTINCT(CITY) FROM STATION WHERE CITY LIKE 'A%' OR CITY LIKE 'E%' OR CITY LIKE 'I%' OR CITY LIKE 'O%'

OR CITY LIKE 'U%' ORDER BY CITY ASC;

Ans12. SELECT DISTINCT(CITY) FROM STATION WHERE CITY LIKE '%a' OR CITY LIKE '%e' OR CITY LIKE '%i' OR CITY LIKE '%o'

OR CITY LIKE '%u';

Ans13. SELECT DISTINCT CITY FROM STATION WHERE upper(SUBSTR(CITY,1,1)) NOT IN ('A','E','I','O','U') AND lower(SUBSTR(CITY,1,1)) NOT IN

('a','e','i','o','u');

Ans 14. SELECT DISTINCT CITY FROM STATION WHERE UPPER(SUBSTR(CITY, LENGTH(CITY), 1)) NOT IN ('A','E','I','O','U') AND LOWER(SUBSTR(CITY, LENGTH(CITY),1)) NOT IN ('a','e','i','o','u');

Ans15. SELECT DISTINCT CITY FROM STATION WHERE LOWER(SUBSTR(CITY,1,1)) NOT IN ('a','e','i','o','u') OR LOWER(SUBSTR(CITY, LENGTH(CITY),1)) NOT IN ('a','e','i','o','u');

Ans16. SELECT DISTINCT CITY FROM STATION WHERE LOWER(SUBSTR(CITY,1,1)) NOT IN ('a','e','i','o','u') AND LOWER(SUBSTR(CITY,LENGTH(CITY),1)) NOT IN ('a','e','i','o','u');

Ans17. **select** p.product\_id, p.product\_name

**from** Product p

**left** **join** Sales s

**on** p.product\_id **=** s.product\_id

**group** **by** p.product\_id

**having** **sum**(s.sale\_date **between** '2019-01-01' **and** '2019-03-31') **=** **count**(s.sale\_date)

Ans 18. select author\_id as id

from views

where author\_id = viewer\_id

group by id

order by id

Ans19. select round(sum(if(order\_date = customer\_pref\_delivery\_date, 0, .4166)) / count(\*)

\* 100,2) as immediate\_percentage

from Delivery

where (customer\_id, order\_date) in (

select customer\_id, min(order\_date)

from Delivery

group by customer\_id

)

Ans 20. select t.ad\_id, (case when base != 0 then round(t.num/t.base\*100,2) else 0 end) as Ctr from (select ad\_id, sum( case when action = 'clicked' or action = 'viewed' then 1 else 0 end) as base, sum( case when action = 'clicked' then 1 else 0 end) as num from ads group by ad\_id)t order by Ctr desc, t.ad\_id asc;

Ans 21. select employee\_id, count(team\_id) over (partition by team\_id) as team\_size from employee order by employee\_id;

Ans22. select c.country\_name, case when avg(weather\_state) <= 15 then 'Cold' when avg(weather\_state) >= 25 then 'Hot' else 'Warm' end as weather\_state from countries c left join weather w on c.country\_id = w.country\_id where month(day) = 11 group by c.country\_name;

Ans23. select p.product\_id, round(sum(u.units\*p.price)/sum(u.units),2) as average\_price from prices p left join unitssold u on p.product\_id = u.product\_id where u.purchase\_date >= start\_date and u.purchase\_date <= end\_date group by product\_id order by product\_id;

Ans24. select t.player\_id, event\_date as first\_login from (select player\_id, event\_date, row\_number() over(partition by player\_id order by event\_date) as num from activity)t where t.num = 1;

Ans 25. select t.player\_id, t.device\_id from (select player\_id, device\_id, row\_number() over(partition by player\_id order by event\_date) as num from activity)t where t.num = 1;

Ans 26. select p.product\_name, sum(o.unit) as unit from Products p left join Orders o on p.product\_id = o.product\_id where month(o.order\_date) = 2 and year(o.order\_date) = 2020 group by p.product\_id having unit >= 100;

Ans 27. select user\_id, name, mail from Users where mail regexp '^[a-zA-Z]+[a-zA-Z0-9\_\.\-]\*@leetcode[\.]com' order by user\_id;

Ans 28. select t.customer\_id, t.name from (select c.customer\_id, c.name, sum(case when month(o.order\_date) = 6 and year(o.order\_date) = 2020 then p.price\*o.quantity else 0 end) as june\_spent, sum(case when month(o.order\_date) = 7 and year(o.order\_date) = 2020 then p.price\*o.quantity else 0 end) as july\_spent from Orders o left join Product p on o.product\_id = p.product\_id left join Customers c on o.customer\_id = c.customer\_id group by c.customer\_id) t where june\_spent >= 100 and july\_spent >= 100;

Ans29. select c.Title from Content c left join TVProgram t on c.content\_id = t.content\_id where c.Kids\_content = 'Y' and c.content\_type = 'Movies' and month(t.program\_date) = 6 and year(t.program\_date) = 2020;

Ans 30. select q.\*, coalesce(n.Npv,0) as Npv from Queries q left join NPV n on q.Id = n.Id and q.Year = n.Year;

Ans 31. select q.\*, coalesce(n.Npv,0) as Npv from Queries q left join NPV n on q.Id = n.Id and q.Year = n.Year;

Ans32. select u.unique\_id, e.name from employees e left join employeeUNI u on e.id = u.id;

Ans 33. select u.name, coalesce(sum(r.distance),0) as travelled\_distance from users u left join rides r on u.id = r.user\_id group by u.name order by travelled\_distance desc, u.name;

Ans 34. select p.product\_name, sum(o.unit) as unit from Products p left join Orders o on p.product\_id = o.product\_id where month(o.order\_date) = 2 and year(o.order\_date) = 2020 group by p.product\_id having unit >= 100;

Ans35. (select t1.name as Results from (select u.name, count(u.user\_id), dense\_rank() over(order by count(user\_id) desc, u.name) as r1 FROM Users u left join MovieRating m on u.user\_id = m.user\_id group by u.user\_id) t1 where r1 = 1) union (select t2.title as Results from (select mo.title, avg(m.rating), dense\_rank() over(order by avg(m.rating)desc, mo.title) as r2 from Movies mo left join MovieRating m on mo.movie\_id = m.movie\_id where month(m.created\_at) = 2 and year(m.created\_at) = 2020 group by m.movie\_id) t2 where r2 = 1);

Ans36. select u.name, coalesce(sum(r.distance),0) as travelled\_distance from users u left join rides r on u.id = r.user\_id group by u.name order by travelled\_distance desc, u.name;

Ans37. select u.unique\_id, e.name from employees e left join employeeUNI u on e.id = u.id;

Ans38. select id, name from Students where department\_id not in (select id from Departments);

Ans39. select t.person1, t.person2, count(\*) as call\_count, sum(t.duration) as total\_duration from (select duration, case when from\_id < to\_id then from\_id else to\_id end as person1, case when from\_id > to\_id then from\_id else to\_id end as person2 from Calls) t group by t.person1, t.person2;

Ans40. select p.product\_id, round(sum(u.units\*p.price)/sum(u.units),2) as average\_price from prices p left join unitssold u on p.product\_id = u.product\_id where u.purchase\_date >= start\_date and u.purchase\_date <= end\_date group by product\_id order by product\_id;

Ans41. select w.name as warehouse\_name, sum(p.width\*p.length\*p.height\*w.units) as volume from warehouse w left join products p on w.product\_id = p.product\_id group by w.name order by w.name;

Ans42. select t.sale\_date, (t.apples\_sold - t.oranges\_sold) as diff from (select sale\_date, max(CASE WHEN fruit = 'apples' THEN sold\_num ELSE 0 END )as apples\_sold, max(CASE WHEN fruit = 'oranges' THEN sold\_num ELSE 0 END )as oranges\_sold FROM sales group by sale\_date) t ORDER BY t.sale\_date;

Ans43. select round(t.player\_id/(select count(distinct player\_id) from activity),2) as fraction from ( select distinct player\_id, datediff(event\_date, lead(event\_date, 1) over(partition by player\_id order by event\_date)) as diff from activity ) t where diff = -1;

Ans44. select t.name from (select a.id, a.name, count(b.managerID) as no\_of\_direct\_reports from employee a INNER JOIN employee b on a.id = b.managerID group by b.managerID) t where no\_of\_direct\_reports >= 5 order by t.name;

Ans45. select d.dept\_name, count(s.dept\_id) as student\_number from department d left join student s on s.dept\_id = d.dept\_id group by d.dept\_id order by student\_number desc, dept\_name;

Ans46. select customer\_id from customer group by customer\_id having count(distinct product\_key)=(select count(\*) from product);

Ans47. select t.project\_id, t.employee\_id from (select p.project\_id, e.employee\_id, dense\_rank() over(partition by p.project\_id order by e.experience\_years desc) as r from project p left join employee e on p.employee\_id = e.employee\_id) t where r = 1 order by t.project\_id;

Ans48. select t1.book\_id, t1.name from ( (select book\_id, name from Books where available\_from < '2019-05-23') t1 left join (select book\_id, sum(quantity) as quantity from Orders where dispatch\_date > '2018-06-23' and dispatch\_date<= '2019-06-23' group by book\_id having quantity < 10) t2 on t1.book\_id = t2.book\_id );

Ans49. select t.student\_id, t.course\_id, t.grade from (select student\_id, course\_id, grade, dense\_rank() over(partition by student\_id order by grade desc, course\_id) as r from enrollments) t where r = 1 order by t.student\_id;

Ans50. select t2.group\_id, t2.player\_id from ( select t1.group\_id, t1.player\_id, dense\_rank() over(partition by group\_id order by score desc, player\_id) as r from ( select p.\*, case when p.player\_id = m.first\_player then m.first\_score when p.player\_id = m.second\_player then m.second\_score end as score from Players p, Matches m where player\_id in (first\_player, second\_player) ) t1 ) t2 where r = 1;

Ans51. select name, population, area from World where area >= 3000000 or population >= 25000000;

Ans52. select name from Customer where refree\_id != 2 or refree\_id is NULL;

Ans 53. select c.name from Customers c left join Orders o on c.id = o.customerID where o.id is NULL;

Ans54. count(team\_id) over(partition by team\_id) as team\_size from Employee order by employee\_id;

Ans55. select t3.Name from ( select t2.Name, avg(t1.duration) over(partition by t2.Name) as avg\_call\_duration, avg(t1.duration) over() as global\_average from ((select cl.caller\_id as id, cl.duration from Calls cl) union (select cl.callee\_id as id, cl.duration from Calls cl)) t1 left join (select p.id, c.Name from Person p left JOIN Country c ON cast(left(p.phone\_number,3) as int) = cast(c.country\_code as int)) t2 ON t1.id = t2.id) t3 where t3.avg\_call\_duration > global\_average group by t3.Name;

Ans 56. select t.player\_id, t.device\_id

from (select player\_id, device\_id, row\_number() over(partition by player\_id order by event\_date) as num from activity)t

where t.num = 1;

Ans57. select customer\_number

from

Orders

group by customer\_number

order by count(order\_number) desc

limit 1;

Ans: To find all such customers, we will use dense\_rank() order by count(order\_number desc).

Now, all those customers who have placed the largest number of orders will get rank 1. Then, we

select all those customers who are having rank 1.

Solution:

select t.customer\_number

from

(select customer\_number,

dense\_rank() over(order by count(order\_number) desc) as r

from

Orders

group by customer\_number) t

where t.r = 1;

Ans58. select t.seat\_id

from

(select seat\_id, lead(seat\_id,1,seat\_id) over(order by seat\_id) as next

from Cinema

where Free != 0

) t

where next - seat\_id in (0,1)

order by seat\_id;

Ans59. select Name from SalesPerson

where sales\_id

not in

(select o.sales\_id

from

Orders o

left join

Company c

on o.com\_id = c.com\_id

where c.Name = 'Red');

Ans60. select X, Y, Z, (case

when X+Y > Z and Y+Z > X and Z+X > Y then 'Yes'

else 'No'

end) as triangle

from Triangle;

Ans61. select min(t.diff) as shortest

from

(select lead(X,1) over(order by X) - X as diff

from

Point) t;

Ans62. select actor\_id, director\_id

from

ActorDirector

group by actor\_id, director\_id

having count(\*) >= 3;

Ans63. select p.product\_name, s.year,

sum(price) as price

from

Sales s

left join

Product p

on s.product\_id = p.product\_id

group by p.product\_name, s.year;

Ans64. select p.project\_id, round(avg(e.experience\_years),2) as average\_years

from

Project p

left join

Employee e

on p.employee\_id = e.employee\_id

group by p.project\_id;

Ans65. select t.seller\_id

from

(select seller\_id , sum(price),

dense\_rank() over(order by sum(price) desc) as r

from Sales

group by seller\_id) t

where t.r = 1;

Ans66. select buyer\_id

from

(

select t1.buyer\_id,

sum(case when t1.product\_name = 'S8' then 1 else 0 end) as S8\_count,

sum(case when t1.product\_name = 'iPhone' then 1 else 0 end) as iphone\_count

from

(

select s.buyer\_id, p.product\_name

from

Sales s

left join

Product p

on s.product\_id = p.product\_id

) t1

group by t1.buyer\_id

) t2

where t2.S8\_count = 1 and t2.iphone\_count = 0;

Ans67. select t2.visited\_on, t2.amount, t2.average\_amount

from

(select t1.visited\_on, t1.prev\_date\_interval\_6,

round(sum(amount) over(order by visited\_on range between interval '6' day preceding and current row),2) as amount,

round(avg(amount) over(order by visited\_on range between interval '6' day preceding and current row),2) as average\_amount

from

(select visited\_on, sum(amount) as amount,

lag(visited\_on,6) over(order by visited\_on) as prev\_date\_interval\_6

from Customer

group by visited\_on

order by visited\_on) t1

) t2

where prev\_date\_interval\_6 is not null;

Ans68. select gender, day,

sum(score\_points) over(partition by gender order by day) as total

from Scores

group by gender, day

order by gender, day;

Ans69. select distinct start.log\_id as start\_id,

min(end.log\_id) over(partition by start.log\_id) as end\_id

from

(select log\_id from Logs where log\_id - 1 not in (select \* from Logs)) start,

(select log\_id from Logs where log\_id + 1 not in (select \* from Logs)) end

where start.log\_id <= end.log\_id;

Ans70. select t.student\_id, t.student\_name , t.subject\_name,

count(e.subject\_name) as attended\_exams

from

(select student\_id, student\_name, subject\_name

from Students, Subjects) t

left join

Examinations e

on t.student\_id = e.student\_id and t.subject\_name = e.subject\_name

group by t.student\_id, t.subject\_name

order by t.student\_id, t.subject\_name;

Ans71. with recursive new as

(

select employee\_id from Employees where employee\_id = 1

union

select e2.employee\_id from new e1

inner join

Employees e2

on e1.employee\_id = e2.manager\_id

)

select \* from new where employee\_id <> 1;

Ans72. select month(trans\_date) as Month,

Country, count(Id) as trans\_count,

sum(case when State = 'approved' then 1 else 0 end) as approved\_count,

sum(amount) as trans\_total\_amount,

sum(case when State = 'approved' then amount else 0 end) as approved\_total\_amount

from Transactions

group by Month, Country;

Ans73. select round(avg(t.daily\_percent), 2) as average\_daily\_percent

from

(

select

sum(case when remove\_date > action\_date then 1 else 0 end)/

count(tmp.action\_date)\*100 as daily\_percent

from

(

select post\_id, action\_date, extra

from Actions where extra = 'spam') tmp

left join Removals r

on tmp.post\_id = r.post\_id

group by action\_date

) t;

Ans74. select round(t.player\_id/(select count(distinct player\_id) from activity),2) as

fraction

from

(

select distinct player\_id,

datediff(event\_date, lead(event\_date, 1) over(partition by player\_id order by event\_date)) as diff

from activity ) t

where diff = -1;

Ans75. select round(t.player\_id/(select count(distinct player\_id) from activity),2) as

fraction

from

(

select distinct player\_id,

datediff(event\_date, lead(event\_date, 1) over(partition by player\_id order by event\_date)) as diff

from activity ) t

where diff = -1;

Ans76. select company\_id, employee\_id, employee\_name,

(case when max(salary) over(partition by company\_id) < 1000 then salary

when max(salary) over(partition by company\_id) < 10000 then round(0.76\*salary)

else round(0.51\*salary)

end) as Salary

from Salaries;

Ans 77. select t.left\_operand, t.operator, t.right\_operand, (case

when t.value > v2.value and operator = '>' then "true"

when t.value < v2.value and operator = '<' then "true"

when t.value = v2.value and operator = '=' then "true"

else "false"

end) as value

from(select e.\*, v1.value

from

Expressions e

inner join

Variables v1

on e.left\_operand = v1.name) t

inner join

Variables v2

on t.right\_operand = v2.name;

Ans 78. select t3.Name from

(

select t2.Name, avg(t1.duration) over(partition by t2.Name) as avg\_call\_duration,

avg(t1.duration) over() as global\_average

from

((select cl.caller\_id as id, cl.duration

from Calls cl)

union

(select cl.callee\_id as id, cl.duration

from Calls cl)) t1

left join

(select p.id, c.Name from Person p

left JOIN

Country c

ON cast(left(p.phone\_number,3) as int) = cast(c.country\_code as int)) t2

ON t1.id = t2.id) t3

where t3.avg\_call\_duration > global\_average

group by t3.Name;

Ans 79. select name

from

Employee

order by name;

Ans80. select year, product\_id, curr\_year\_spend, coalesce(prev\_year\_spend,'') as prev\_year\_spend,

coalesce(round((curr\_year\_spend - prev\_year\_spend)/prev\_year\_spend \*100,2),'') as yoy\_rate

from

(

select year(transaction\_date) as year, product\_id, spend as curr\_year\_spend,

round(lag(spend,1) over(partition by product\_id order by transaction\_date),2) as prev\_year\_spend

from user\_transactions

) t;

Ans81. select item\_type, (case

when item\_type = 'prime\_eligible' then floor(500000/sum(square\_footage)) \* count(item\_type)

when item\_type = 'not\_prime' then floor((500000 -(select floor(500000/sum(square\_footage)) \* sum(square\_footage) from inventory where item\_type = 'prime\_eligible'))/sum(square\_footage)) \* count(item\_type)

end) as item\_count

from inventory

group by item\_type

order by count(item\_type) desc;

Ans82. select month(a.event\_date) as month, count(distinct a.user\_id) as

monthly\_active\_users

from

user\_actions a

inner join

user\_actions b

on concat(month(a.event\_date),year(a.event\_date)) = concat(1+month(b.event\_date),year(b.event\_date))

and a.user\_id = b.user\_id

where a.event\_type in ('sign-in', 'like', 'comment')

and b.event\_type in ('sign-in', 'like', 'comment')

and concat(month(a.event\_date),'/',year(a.event\_date)) = '7/2022'

and concat(1+month(b.

select month(a.event\_date) as month, count(distinct a.user\_id) as

monthly\_active\_users

from

user\_actions a

inner join

user\_actions b

on concat(month(a.event\_date),year(a.event\_date)) = concat(1+month(b.event\_date),year(b.event\_date))

and a.user\_id = b.user\_id

where a.event\_type in ('sign-in', 'like', 'comment')

and b.event\_type in ('sign-in', 'like', 'comment')

and concat(month(a.event\_date),'/',year(a.event\_date)) = '6/2022'

and concat(1+month(b.event\_date),'/',year(b.event\_date)) = '6/2022'

group by month(a.event\_date);

Ans83. using recursive cte

with recursive seq as

(

select searches, num\_users, 1 as c from search\_frequency

union

select searches, num\_users, c+1 from seq where c < num\_users

)

select round(avg(t.searches),1) as median from

(select searches,row\_number() over(order by searches, c) as r1,

row\_number() over(order by searches desc, c desc) as r2 from seq order by searches) t

where t.r1 in (t.r2, t.r2 - 1, t.r2 + 1);

Ans84.

|  |  |  |
| --- | --- | --- |
| Conditions used in case when: Previous Status | Condition | Next Status |
| New, Existing, Churn, Resurrect | Didn’t pay on day T | Churn |
| New, Existing, Resurrect | Paid on day T | Existing |
| Churn | Paid on day T | Resurrect |

select user\_id, case

when status in ('NEW','EXISTING','CHURN','RESURRECT') and user\_id not in (select user\_id from daily\_pay) then 'CHURN'

when status in ('NEW','EXISTING','RESURRECT') and user\_id in (select user\_id from daily\_pay) then 'EXISTING'

when status = 'CHURN' and user\_id in (select user\_id from daily\_pay) then 'RESURRECT'

end as new\_status

from advertiser

order by user\_id;

Ans85. select sum(t.individual\_uptime) as total\_uptime\_days

from

(

select case when session\_status = 'stop'

then

timestampdiff(day, lag(status\_time) over(partition by server\_id order by status\_time), status\_time) end as individual\_uptime

from server\_utilization

) t;

Ans86. select sum(case when (unix\_timestamp(t.next\_transaction) - unix\_timestamp(t.transaction\_timestamp))/60 <= 10 then 1 else 0 end) as payment\_count

from

(select transaction\_timestamp,

lead(transaction\_timestamp,1) over(partition by merchant\_id, credit\_card\_id, Amount order by transaction\_timestamp) as next\_transaction

from transactions)t;

Ans87. select round(avg(t1.bad\_exp\_pct\_per\_cust),2) as bad\_exp\_pct

from

(

select t.customer\_id, 100\*sum(case when o.status <> 'completed successfully' then 1 else 0 end)/count(\*) as bad\_exp\_pct\_per\_cust

from

(

select customer\_id, signup\_timestamp from customers where month(signup\_timestamp) = 6

) t

inner join

orders o

on o.customer\_id = t.customer\_id

where timestampdiff(day, t.signup\_timestamp, o.order\_timestamp) <= 13

group by t.customer\_id

) t1;

Ans88. select gender, day,

sum(score\_points) over(partition by gender order by day) as total

from Scores

group by gender, day

order by gender, day;

Ans89. select t3.Name from

(

select t2.Name, avg(t1.duration) over(partition by t2.Name) as avg\_call\_duration,

avg(t1.duration) over() as global\_average

from

((select cl.caller\_id as id, cl.duration

from Calls cl)

union

(select cl.callee\_id as id, cl.duration

from Calls cl)) t1

left join

(select p.id, c.Name from Person p

left JOIN

Country c

ON cast(left(p.phone\_number,3) as int) = cast(c.country\_code as int)) t2

ON t1.id = t2.id) t3

where t3.avg\_call\_duration > global\_average

group by t3.Name;

Ans90. using recursive cte

with recursive seq as

(

select num, frequency, 1 as c from Numbers

union

select num, frequency, c+1 from seq where c < frequency

)

select round(avg(t.num),1) as median

from

(

select num,row\_number() over(order by num, c) as r1,

row\_number() over(order by num desc, c desc) as r2 from seq order by num

) t

where t.r1 in (t.r2, t.r2 - 1,t.r2 + 1);

using cumulative sum

select round(avg(t1.num),1) as median

from

(select t.num, t.cumm\_sum,

lag(cumm\_sum,1,0) over(order by num) as prev\_cumm\_sum,

case when total % 2 = 0 then total/2 else (total+1)/2 end as pos1,

case when total % 2 = 0 then (total/2)+1 else (total+1)/2 end as pos2

from

(select num, frequency,

sum(frequency) over(order by num rows between unbounded preceding and current row) as cumm\_sum,

sum(frequency) over(order by num rows between unbounded preceding and unbounded following) as total

Ans91. select distinct concat(year(t.pay\_date),'-',month(t.pay\_date)) as pay\_month,

t.department\_id,

case

when monthly\_department\_avg\_salary > monthly\_average\_salary then 'higher'

when monthly\_department\_avg\_salary < monthly\_average\_salary then 'lower'

else 'same'

end as Comparison

from

(select s.pay\_date, e.department\_id,

avg(s.amount) over(partition by month(s.pay\_date), e.department\_id) as monthly\_department\_avg\_salary,

avg(s.amount) over(partition by month(s.pay\_date)) as monthly\_average\_salary

from Salary s

left join

Employee e

on s.employee\_id = e.employee\_id) t

order by t.department\_id;

Ans92. select t1.install\_dt, count(player\_id) as installs,

round(count(t1.next\_install)/count(t1.player\_id),2) as Day1\_retention

from

(

select t.player\_id, t.install\_dt, a.event\_date as next\_install

from

(

select player\_id, min(event\_date) as install\_dt

from Activity

group by player\_id

) t

left join

Activity a

on t. player\_id = a.player\_id and a.event\_date = t.install\_dt + 1

) t1

group by install\_dt;

Ans93. select t2.group\_id, t2.player\_id from

(

select t1.group\_id, t1.player\_id,

dense\_rank() over(partition by group\_id order by score desc, player\_id) as r

from

(

select p.\*, case when p.player\_id = m.first\_player then m.first\_score

when p.player\_id = m.second\_player then m.second\_score

end as score

from

Players p, Matches m

where player\_id in (first\_player, second\_player)

) t1

) t2

where r = 1;

Ans94. select t.student\_id, t.student\_name from

(select s.student\_name, s.student\_id, count(e.student\_id) over(partition by student\_name) as exams\_given,

case when e.score > min(e.score) over(partition by e.exam\_id) and e.score < max(e.score) over(partition by e.exam\_id) then 1 else 0 end as quiet

# 1 means student is quiet, 0 means student is not quiet

from Exam e

left join

Student s

on e.student\_id = s.student\_id)t

group by t.student\_name, t.student\_id, t.exams\_given

having sum(t.quiet) = t.exams\_given

# sum(quiet) will give the total number of exams in which student is quiet

Ans95. select t.student\_id, t.student\_name from

(select s.student\_name, s.student\_id, count(e.student\_id) over(partition by student\_name) as exams\_given,

case when e.score > min(e.score) over(partition by e.exam\_id) and e.score < max(e.score) over(partition by e.exam\_id) then 1 else 0 end as quiet

# 1 means student is quiet, 0 means student is not quiet

from Exam e

left join

Student s

on e.student\_id = s.student\_id)t

group by t.student\_name, t.student\_id, t.exams\_given

having sum(t.quiet) = t.exams\_given

# sum(quiet) will give the total number of exams in which student is quiet

Ans96. select t.user\_id, t.song\_id, sum(t.song\_plays) as song\_plays

from

(

select user\_id, song\_id, song\_plays

from songs\_history

union all

select user\_id, song\_id, 1 as song\_plays

from songs\_weekly

where date(listen\_time) <= '2022/08/04') t

group by user\_id, song\_id;

Ans97. select round(sum(case when t.signup\_action = 'Confirmed' then 1 else 0

end)/count(\*),2) as confirm\_rate

from

emails e

join

texts t

on e.email\_id = t.email\_id;

Ans98. select user\_id, date\_format(tweet\_date, '%m/%d/%Y %h:%i:%s') as tweet\_date,

round(avg(count(distinct tweet\_id)) over(order by tweet\_date rows between 2 preceding and current row),2) as rolling\_avg\_3days

from tweets

group by user\_id, tweet\_date

Ans99. select b.age\_bucket,

round(100\*sum(case when a.activity\_type = 'Send' then a.time\_spent else 0 end)/sum(a.time\_spent),2) send\_perc,

round(100\*sum(case when a.activity\_type = 'Open' then a.time\_spent else 0 end)/sum(a.time\_spent),2) open\_perc

from

activities a

join

age\_breakdown b

on a.user\_id = b.user\_id

where activity\_type in ('Open', 'Send')

group by b.age\_bucket

order by b.age\_bucket;

Ans100. select p.profile\_id

from

personal\_profiles p

join

employee\_company e

on p.profile\_id = e.personal\_profile\_id

join

company\_pages c

on e.company\_id = c.company\_id

group by p.profile\_id, p.followers

having p.followers > sum(c.followers)

order by profile\_id;

Ans101. with new as

(select t.username, t.activity, t.startDate, t.endDate

from(

select username, activity, startDate, endDate,

dense\_rank() over(partition by username order by endDate desc) as r

from UserActivity)t

where r = 2

)

select \* from new

union

select n.username, n.activity, n.startDate, n.endDate

from(

select username, activity, startDate, endDate,

dense\_rank() over(partition by username order by endDate desc) as r

from UserActivity)n

where r = 1 and username not in (select username from new);

Ans102. with new as

(select t.username, t.activity, t.startDate, t.endDate

from(

select username, activity, startDate, endDate,

dense\_rank() over(partition by username order by endDate desc) as r

from UserActivity)t

where r = 2

)

select \* from new

union

select n.username, n.activity, n.startDate, n.endDate

from(

select username, activity, startDate, endDate,

dense\_rank() over(partition by username order by endDate desc) as r

from UserActivity)n

where r = 1 and username not in (select username from new);

Ans103. select name from Students

where marks > 75

order by right(name, 3), id;

Ans104. select name

from

Employee

where salary > 2000 and months < 10

order by employee\_id;

Ans105. select case

when A+B > C and B+C > A and C+A > B then

(

case

when A != B and B != C then 'Scalene'

when A = B and B = C then 'Equilateral'

else 'Isosceles'

end

)

else 'Not A Triangle'

end as Result

from Triangles;

Ans106. select ceil(avg(salary) - avg(replace(salary, 0, '')))

as calculation\_difference

from Employees;

Ans107. select concat(max(t.earnings), ' ',

sum(case

when earnings = max\_salary then 1

else 0

end)) as Output

from

(

select max(salary\*months) over() as max\_salary,

salary\*months as earnings

from

Employee) t;

Ans108. select concat(name, '(', left(occupation,1),')') as name\_occupation)

from Occupations

order by name;

select

concat('There are a total of', ' ', count(occupation), ' ', lower(occupation), 's.') as occupation\_count

from Occupations

group by occupation

order by count(occupation), occupation;

Ans109. select max(case Occupation when 'Doctor' then Name end) as Doctors,

max(case Occupation when 'Professor' then Name end) as Professors,

max(case Occupation when 'Singer' then Name end) as Singers,

max(case Occupation when 'Actor' then Name end) as Actors

from

(

select occupation, name,

row\_number() over(partition by Occupation order by name) as r

from Occupations

) t

group by r;

Ans110. select

(

case

when P is NULL then 'Root'

when N not in (select distinct P from BST where P is not null) then 'Leaf'

else 'Inner'

end

) as Node\_Type

from BST

order by N;

Ans111. select concat(c.company\_code, ' ', c.founder, ' ',

count(distinct l.lead\_manager\_code), ' ',

count(distinct s.senior\_manager\_code), ' ',

count(distinct m.manager\_code), ' ',

count(distinct e.employee\_code)) as Output

from Company c

left outer join

Lead\_Manager l

on c.company\_code = l.company\_code

left join

Senior\_Manager s

on l.lead\_manager\_code = s.lead\_manager\_code

left join

Manager m

on s.senior\_manager\_code = m.senior\_manager\_code

left join

Employee e

on m.manager\_code = e.manager\_code

group by c.company\_code, c.founder

order by c.company\_code;

Ans112. with recursive cte as

(

select 2 as num

union

select num+1 from cte

where num+1 <= 1000

)

select GROUP\_CONCAT(num SEPARATOR "&") as prime

from

(

select 2 as num

union

select c1.num from cte c1

inner join

cte c2 on c2.num <= round(c1.num/2)

group by num

having min(c1.num % c2.num) > 0

order by num

)t;

Ans113. with recursive num(n) as

(

select 1

union

select n + 1

from num

where n + 1 <= 20

)

select lpad('', num.n, '\*') as 'P(20)'

from num;

Ans114. with recursive num(n) as

(

select 20

union

select n - 1

from num

where n - 1 >= 1

)

select lpad('', num.n, '\*') as 'P(20)'

from num;

Ans116. select distinct a.X, a.Y from

(select \*, row\_number() over(order by X) as r1 from Functions) a

inner join

(select \*,row\_number() over(order by X) as r2 from Functions) b

on a.X = b.Y and b.X = a.Y

where a.X <= a.Y and a.r1 <> b.r2

order by a.X

Ans117. select case

when A+B > C and B+C > A and C+A > B then

(

case

when A != B and B != C then 'Scalene'

when A = B and B = C then 'Equilateral'

else 'Isosceles'

end

)

else 'Not A Triangle'

end as Result

from Triangles;

Ans118. select year, product\_id, curr\_year\_spend, coalesce(prev\_year\_spend,'') as prev\_year\_spend,

coalesce(round((curr\_year\_spend - prev\_year\_spend)/prev\_year\_spend \*100,2),'') as yoy\_rate

from

(

select year(transaction\_date) as year, product\_id, spend as curr\_year\_spend,

round(lag(spend,1) over(partition by product\_id order by transaction\_date),2) as prev\_year\_spend

from user\_transactions

) t;

Ans119. select year, product\_id, curr\_year\_spend, coalesce(prev\_year\_spend,'') as prev\_year\_spend,

coalesce(round((curr\_year\_spend - prev\_year\_spend)/prev\_year\_spend \*100,2),'') as yoy\_rate

from

(

select year(transaction\_date) as year, product\_id, spend as curr\_year\_spend,

round(lag(spend,1) over(partition by product\_id order by transaction\_date),2) as prev\_year\_spend

from user\_transactions

) t;

Ans120. select item\_type, (case

when item\_type = 'prime\_eligible' then floor(500000/sum(square\_footage)) \* count(item\_type)

when item\_type = 'not\_prime' then floor((500000 -(select floor(500000/sum(square\_footage)) \* sum(square\_footage) from inventory where item\_type = 'prime\_eligible'))/sum(square\_footage)) \* count(item\_type)

end) as item\_count

from inventory

group by item\_type

order by count(item\_type) desc;

Ans121. select month(a.event\_date) as month, count(distinct a.user\_id) as

monthly\_active\_users

from

user\_actions a

inner join

user\_actions b

on concat(month(a.event\_date),year(a.event\_date)) = concat(1+month(b.event\_date),year(b.event\_date))

and a.user\_id = b.user\_id

where a.event\_type in ('sign-in', 'like', 'comment')

and b.event\_type in ('sign-in', 'like', 'comment')

and concat(month(a.event\_date),'/',year(a.event\_date)) = '7/2022'

and concat(1+month(b.event\_date),'/',year(b.event\_date)) = '7/2022'

group by month(a.event\_date);

Ans122. with recursive seq as

(

select searches, num\_users, 1 as c from search\_frequency

union

select searches, num\_users, c+1 from seq where c < num\_users

)

select round(avg(t.searches),1) as median from

(select searches,row\_number() over(order by searches, c) as r1,

row\_number() over(order by searches desc, c desc) as r2 from seq order by searches) t

where t.r1 in (t.r2, t.r2 - 1, t.r2 + 1);

Ans123. using cumulative sum

select round(avg(t1.searches),1) as median

from

(select t.searches, t.cumm\_sum,

lag(cumm\_sum,1,0) over(order by searches) as prev\_cumm\_sum,

case when total % 2 = 0 then total/2 else (total+1)/2 end as pos1,

case when total % 2 = 0 then (total/2)+1 else (total+1)/2 end as pos2

from

(select searches, num\_users,

sum(num\_users) over(order by searches rows between unbounded preceding and current row) as cumm\_sum,

sum(num\_users) over(order by searches rows between unbounded preceding and unbounded following) as total

from search\_frequency) t

) t1

where (t1.pos1 > t1.prev\_cumm\_sum and t1.pos1 <= t1.cumm\_sum) or (t1.pos2 > t1.prev\_cumm\_sum and t1.pos2 <= t1.cumm\_sum);

Ans124.

|  |  |  |
| --- | --- | --- |
| Conditions used in case when: Previous Status | Condition | Next Status |
| New, Existing, Churn, Resurrect | Didn’t pay on day T | Churn |
| New, Existing, Resurrect | Paid on day T | Existing |
| Churn | Paid on day T | Resurrect |

select user\_id, case

when status in ('NEW','EXISTING','CHURN','RESURRECT') and user\_id not in (select user\_id from daily\_pay) then 'CHURN'

when status in ('NEW','EXISTING','RESURRECT') and user\_id in (select user\_id from daily\_pay) then 'EXISTING'

when status = 'CHURN' and user\_id in (select user\_id from daily\_pay) then 'RESURRECT'

end as new\_status

from advertiser

order by user\_id;

Ans125. select sum(case when (unix\_timestamp(t.next\_transaction) - unix\_timestamp(t.transaction\_timestamp))/60 <= 10 then 1 else 0 end) as payment\_count

from

(select transaction\_timestamp,

lead(transaction\_timestamp,1) over(partition by merchant\_id, credit\_card\_id, Amount order by transaction\_timestamp) as next\_transaction

from transactions)t;

Ans126. select round(avg(t1.bad\_exp\_pct\_per\_cust),2) as bad\_exp\_pct

from

(

select t.customer\_id, 100\*sum(case when o.status <> 'completed successfully' then 1 else 0 end)/count(\*) as bad\_exp\_pct\_per\_cust

from

(

select customer\_id, signup\_timestamp from customers where month(signup\_timestamp) = 6

) t

inner join

orders o

on o.customer\_id = t.customer\_id

where timestampdiff(day, t.signup\_timestamp, o.order\_timestamp) <= 13

group by t.customer\_id

) t1;

Ans127. select gender, day,

sum(score\_points) over(partition by gender order by day) as total

from Scores

group by gender, day

order by gender, day;

Ans128. select t3.Name from

(

select t2.Name, avg(t1.duration) over(partition by t2.Name) as avg\_call\_duration,

avg(t1.duration) over() as global\_average

from

((select cl.caller\_id as id, cl.duration

from Calls cl)

union

(select cl.callee\_id as id, cl.duration

from Calls cl)) t1

left join

(select p.id, c.Name from Person p

left JOIN

Country c

ON cast(left(p.phone\_number,3) as int) = cast(c.country\_code as int)) t2

ON t1.id = t2.id) t3

where t3.avg\_call\_duration > global\_average

group by t3.Name;

Ans129. with recursive seq as

(

select num, frequency, 1 as c from Numbers

union

select num, frequency, c+1 from seq where c < frequency

)

select round(avg(t.num),1) as median

from

(

select num,row\_number() over(order by num, c) as r1,

row\_number() over(order by num desc, c desc) as r2 from seq order by num

) t

where t.r1 in (t.r2, t.r2 - 1,t.r2 + 1);

Ans130. using cumulative sum

select round(avg(t1.num),1) as median

from

(select t.num, t.cumm\_sum,

lag(cumm\_sum,1,0) over(order by num) as prev\_cumm\_sum,

case when total % 2 = 0 then total/2 else (total+1)/2 end as pos1,

case when total % 2 = 0 then (total/2)+1 else (total+1)/2 end as pos2

from

(select num, frequency,

sum(frequency) over(order by num rows between unbounded preceding and current row) as cumm\_sum,

sum(frequency) over(order by num rows between unbounded preceding and unbounded following) as total

from Numbers) t

Ans131. select t1.install\_dt, count(player\_id) as installs,

round(count(t1.next\_install)/count(t1.player\_id),2) as Day1\_retention

from

(

select t.player\_id, t.install\_dt, a.event\_date as next\_install

from

(

select player\_id, min(event\_date) as install\_dt

from Activity

group by player\_id

) t

left join

Activity a

on t. player\_id = a.player\_id and a.event\_date = t.install\_dt + 1

) t1

group by install\_dt;

Ans132. select t2.group\_id, t2.player\_id from

(

select t1.group\_id, t1.player\_id,

dense\_rank() over(partition by group\_id order by score desc, player\_id) as r

from

(

select p.\*, case when p.player\_id = m.first\_player then m.first\_score

when p.player\_id = m.second\_player then m.second\_score

end as score

from

Players p, Matches m

where player\_id in (first\_player, second\_player)

) t1

) t2

where r = 1;

Ans133. select t.student\_id, t.student\_name from

(select s.student\_name, s.student\_id, count(e.student\_id) over(partition by student\_name) as exams\_given,

case when e.score > min(e.score) over(partition by e.exam\_id) and e.score < max(e.score) over(partition by e.exam\_id) then 1 else 0 end as quiet

# 1 means student is quiet, 0 means student is not quiet

from Exam e

left join

Student s

on e.student\_id = s.student\_id)t

group by t.student\_name, t.student\_id, t.exams\_given

having sum(t.quiet) = t.exams\_given

# sum(quiet) will give the total number of exams in which student is quiet

Ans134. select s.student\_name, s.student\_id, count(e.student\_id) over(partition by

student\_name) as exams\_given,

case when e.score > min(e.score) over(partition by e.exam\_id) and e.score < max(e.score) over(partition by e.exam\_id) then 1 else 0 end as quiet

# 1 means student is quiet, 0 means student is not quiet

from Exam e

left join

Student s

on e.student\_id = s.student\_id)t

group by t.student\_name, t.student\_id, t.exams\_given

having sum(t.quiet) = t.exams\_given

# sum(quiet) will give the total number of exams in which student is quiet

Ans135. with new as

(select t.username, t.activity, t.startDate, t.endDate

from(

select username, activity, startDate, endDate,

dense\_rank() over(partition by username order by endDate desc) as r

from UserActivity)t

where r = 2

)

select \* from new

union

select n.username, n.activity, n.startDate, n.endDate

from(

select username, activity, startDate, endDate,

dense\_rank() over(partition by username order by endDate desc) as r

from UserActivity)n

where r = 1 and username not in (select username from new);

Ans136. with new as

(select t.username, t.activity, t.startDate, t.endDate

from(

select username, activity, startDate, endDate,

dense\_rank() over(partition by username order by endDate desc) as r

from UserActivity)t

where r = 2

)

select \* from new

union

select n.username, n.activity, n.startDate, n.endDate

from(

select username, activity, startDate, endDate,

dense\_rank() over(partition by username order by endDate desc) as r

from UserActivity)n

where r = 1 and username not in (select username from new);

Ans137. select ceil(avg(salary) - avg(replace(salary, 0, '')))

as calculation\_difference

from Employees;

Ans138. select concat(max(t.earnings), ' ',

sum(case

when earnings = max\_salary then 1

else 0

end)) as Output

from

(

select max(salary\*months) over() as max\_salary,

salary\*months as earnings

from

Employee) t;

Ans139. select concat(name, '(', left(occupation,1),')') as name\_occupation)

from Occupations

order by name;

select

concat('There are a total of', ' ', count(occupation), ' ', lower(occupation), 's.') as occupation\_count

from Occupations

group by occupation

order by count(occupation), occupation;

Ans140. select max(case Occupation when 'Doctor' then Name end) as Doctors,

max(case Occupation when 'Professor' then Name end) as Professors,

max(case Occupation when 'Singer' then Name end) as Singers,

max(case Occupation when 'Actor' then Name end) as Actors

from

(

select occupation, name,

row\_number() over(partition by Occupation order by name) as r

from Occupations

) t

group by r;

Ans 141. select

(

case

when P is NULL then 'Root'

when N not in (select distinct P from BST where P is not null) then 'Leaf'

else 'Inner'

end

) as Node\_Type

from BST

order by N;

Ans 142. select concat(c.company\_code, ' ', c.founder, ' ',

count(distinct l.lead\_manager\_code), ' ',

count(distinct s.senior\_manager\_code), ' ',

count(distinct m.manager\_code), ' ',

count(distinct e.employee\_code)) as Output

from Company c

left outer join

Lead\_Manager l

on c.company\_code = l.company\_code

left join

Senior\_Manager s

on l.lead\_manager\_code = s.lead\_manager\_code

left join

Manager m

on s.senior\_manager\_code = m.senior\_manager\_code

left join

Employee e

on m.manager\_code = e.manager\_code

group by c.company\_code, c.founder

order by c.company\_code;

Ans143. select distinct a.X, a.Y from

(select \*, row\_number() over(order by X) as r1 from Functions) a

inner join

(select \*,row\_number() over(order by X) as r2 from Functions) b

on a.X = b.Y and b.X = a.Y

where a.X <= a.Y and a.r1 <> b.r2

order by a.X

Ans144. select s.name

from

Students s

join

Friends f

on s.id = f.id

join

Packages sp

on sp.id = s.id

join

Packages fp

on fp.id = f.friend\_id

where fp.salary > sp.salary

order by fp.salary;

Ans 145. select concat(t1.hacker\_id, ' ', t1.name) as Result from

(

select t.hacker\_id, t.name,

dense\_rank() over(order by full\_score\_challenge\_count desc) as r

from

(

select h.hacker\_id, h.name, count(h.hacker\_id) as full\_score\_challenge\_count

from

Submissions s

join

Hackers h

on s.hacker\_id = h.hacker\_id

join

Challenges c

on s.challenge\_id = c.challenge\_id

join

Difficulty d

on d.difficulty\_level = c.difficulty\_level

where s.score = d.score

group by h.hacker\_id, h.name

having full\_score\_challenge\_count > 1

) t

) t1

where t1.r = 1

order by t1.hacker\_id;

Ans146. select s.start\_date, min(e.end\_date) as end\_date, (min(e.end\_date) - s.start\_date) as number\_of\_days

from

(select start\_date from Projects where start\_date - 1 not in (select start\_date from Projects)) s,

(select end\_date from Projects where end\_date + 1 not in (select end\_date from Projects)) e

where s.start\_date <= e.end\_date

group by s.start\_date;

Ans147. select distinct t.user\_id

from

(

select user\_id, transaction\_date as first,

lead(transaction\_date,1) over(partition by user\_id order by transaction\_date) as second,

lead(transaction\_date,2) over(partition by user\_id order by transaction\_date) as third

from transactions

) t

where timestampdiff(day, first, second) = 1 and timestampdiff(day, second, third) = 1;

Ans 148. select count(\*) as unique\_relationshis

from

(select count(\*) as relation\_count

from

(

select greatest(payer\_id, recipient\_id) as person1,

least(payer\_id, recipient\_id) as person2

from

(select distinct \* from payments) t

) t1

group by person1, person2

) t2

where relation\_count = 2;

Ans149. select count(\*) as users from

(

select transaction\_id, user\_id, spend,

row\_number() over(partition by user\_id order by transaction\_date) as r

from user\_transactions

) t

where t.r =1 and t.spend >= 50;

Ans150. select measurement\_day,

round(sum(case when r % 2 != 0 then measurement\_value else 0 end),2) as odd\_sum,

round(sum(case when r % 2 = 0 then measurement\_value else 0 end),2) as even\_sum

from

(

select date\_format(measurement\_time, '%m/%d/%Y 00:00:00') as measurement\_day,

measurement\_value, row\_number() over(partition by date(measurement\_time) order by measurement\_time) as r

from measurements

)t

group by measurement\_day;

Ans151. select distinct t.user\_id

from

(

select user\_id, transaction\_date as first,

lead(transaction\_date,1) over(partition by user\_id order by transaction\_date) as second,

lead(transaction\_date,2) over(partition by user\_id order by transaction\_date) as third

from transactions

) t

where timestampdiff(day, first, second) = 1 and timestampdiff(day, second, third) = 1;

Ans152. select count(t1.amenity\_count) as matching\_airbnb

from

(

select t.amenities, count(\*) as amenity\_count

from

(

select rental\_id, group\_concat(amenity order by amenity) amenities

from rental\_amenities

group by rental\_id

)t

group by t.amenities

)t1

where t1.amenity\_count>1;

Ans153. select advertiser\_id,

sum(revenue)/sum(spend) as ROAS

from ad\_campaigns

group by advertiser\_id

order by advertiser\_id;

Ans 154.

select t.employee\_id, t.salary, case

when t.salary > t.base\_for\_overpaid then 'Overpaid'

when t.salary < t.base\_for\_underpaid then 'Underpaid'

end as status

from

(select employee\_id, salary, 2\*avg(salary) over(partition by title) as base\_for\_overpaid,

0.5\*avg(salary) over(partition by title) as base\_for\_underpaid

from employee\_pay

)t

having status is not null

order by t.employee\_id;

Ans155. select count(\*) as unique\_relationshis

from

(select count(\*) as relation\_count

from

(

select greatest(payer\_id, recipient\_id) as person1,

least(payer\_id, recipient\_id) as person2

from

(select distinct \* from payments) t

) t1

group by person1, person2

) t2

where relation\_count = 2;

Ans156. select count(distinct t.user\_id) as repeat\_purchasers

from

(

select user\_id, product\_id, count(\*) as c

from purchases

group by user\_id, product\_id

having

Ans157. select distinct DATE\_FORMAT(transaction\_date, '%m/%d/%Y 12:00:00'),

round(sum(amount) over(partition by month(transaction\_date) order by transaction\_date),2) as balance

from

(

select transaction\_date, case when type = 'deposit' then amount else -amount end as amount

from transactions

) t;

Ans158. select t.category, t.product, t.total\_spend

from

(

select category, product, round(sum(spend),2) as total\_spend,

dense\_rank() over(partition by category order by sum(spend) desc) as r

from product\_spend

group by category, product

) t

where r <= 2

Ans159. according to week of year

select week(signup\_date),

round(100\*sum(case when timestampdiff(day,signup\_date,last\_login) <= 28 then 1 else 0 end)/count(\*),2) as churn\_rate

from users

group by week(signup\_date);

according to week of month

(select signup\_date, last\_login, case

when week(signup\_date) = 22 then 1

when week(signup\_date) = 23 then 2

when week(signup\_date) = 24 then 3

when week(signup\_date) = 25 then 4

when week(signup\_date) = 26 then 5

end as signup\_week

from users)

select signup\_week,

round(100\*sum(case when timestampdiff(day,signup\_date,last\_login) <= 28 then 1 else 0 end)/count(\*),2) as churn\_rate

from cte

group by signup\_week;