A mysql  B sqlserver

175 (B)

SELECT p.FirstName,

p.LastName,

a.City,

a.State

From Person p

Left Join Address a ON p.PersonId = a.PersonId

176(A) sqlserver cannot use limit

Select IFNULL(

    (Select DISTINCT Salary

    From Employee

    Order by Salary

    LIMIT 1 offset 1),null) as SecondHighestSalary

177(A)

CREATE FUNCTION getNthHighestSalary(N INT) RETURNS INT

BEGIN

DECLARE M int;

set M = N-1;

  RETURN (

      # Write your MySQL query statement below.

SELECT IFNULL(

(SELECT DISTINCT Salary

From Employee

Order by Salary DESC

LIMIT 1 Offset M),null)

  );

END

178(B) Read carefully!!!!!

SELECT Score,

DENSE\_RANK() OVER(ORDER BY Score DESC) as Rank

FROM Scores

ORDER BY Score DESC

180(A)

SELECT DISTINCT l1.Num as ConsecutiveNums

FROM Logs l1

INNER JOIN Logs l2 ON l1.Id+1 = l2.Id

    AND l1.Num = l2.Num

INNER JOIN Logs l3 ON l2.Id+1 = l3.Id

    AND L2.Num = L3.Num

181(A)

SELECT a.Name AS Employee

FROM Employee a

INNER JOIN Employee b ON a.ManagerId = b.Id

    AND a.Salary > b.Salary

182(A)

SELECT Email

FROM Person

GROUP BY Email

HAVING COUNT(Email) >1

SELECT a.Email

FROM Person a

INNER JOIN Person b ON a.Email = b.Email

    AND a.Id <> b.Id

GROUP BY a.Email

SELECT DISTINCT a.Email

FROM Person a, Person b

WHERE a.Email = b.Email

    AND a.Id <> b.Id

183(A)

SELECT c.Name AS Customers

FROM Customers c

LEFT JOIN Orders o ON c.Id = o.CustomerId

WHERE o.Id IS NULL

SELECT c.Name AS Customers

FROM Customers c

WHERE NOT EXISTS (SELECT 1 FROM Orders o WHERE c.Id = o.CustomerId )

SELECT c.Name AS Customers

FROM Customers c

WHERE c.Id NOT IN (SELECT o.CustomerId From Orders o )

184(A)

SELECT d.Name AS Department,

a.Name AS Employee,

a.Salary

FROM Employee a

INNER JOIN Department d ON a.DepartmentId = d.Id

INNER JOIN (SELECT e.DepartmentId,

    MAX(e.Salary) AS Salary

    FROM Employee e

    GROUP BY DepartmentId

) cte

    ON a.DepartmentId = cte.DepartmentId

    AND a.Salary = cte.Salary

(B)

SELECT d.Name AS Department,

a.Name AS Employee,

a.Salary

FROM (SELECT DepartmentId,

    Name,

    Salary,

    RANK() OVER (PARTITION BY DepartmentId ORDER BY Salary DESC) rn

    FROM Employee) a

INNER JOIN Department d ON a.DepartmentId = d.Id

WHERE a.rn = 1

185(B)

SELECT d.Name AS Department,

c.Name AS Employee,

c.Salary

FROM (SELECT DepartmentId,

    Name,

    Salary,

    DENSE\_RANK() OVER(PARTITION BY DepartmentId ORDER BY Salary DESC) rn

    FROM Employee

) c

INNER JOIN Department d ON c.DepartmentId = d.Id

WHERE rn IN (1,2,3)

196(A)

DELETE FROM Person

WHERE Id NOT IN (SELECT MIN(a.Id) FROM (SELECT \* FROM Person) a GROUP BY Email)

197

MySQL 第一个参数减第二个参数 DATEDIFF(a.DATE, b.DATE)

Sql Server 第二个参数减第一个参数  DATEDIFF(day, a.DATE, b.DATE)

(A)

SELECT a.Id

FROM Weather a

INNER JOIN Weather b ON DATEDIFF(a.RecordDate,b.RecordDate) = 1

WHERE a.Temperature > b.Temperature

(B)

SELECT a.Id

FROM Weather a

INNER JOIN Weather b ON DATEDIFF(day,a.RecordDate,b.RecordDate) = -1

WHERE a.Temperature > b.Temperature

262(A)

SELECT t.Request\_at AS Day,

Round(COUNT(IF(t.Status = 'cancelled\_by\_driver'OR t.Status ='cancelled\_by\_client',TRUE, null))/ COUNT(\*),2) AS 'Cancellation Rate'

FROM Trips t

LEFT JOIN Users u1 ON t.Client\_Id = u1.Users\_Id

LEFT JOIN Users u2 ON t.Driver\_Id = u2.Users\_Id

WHERE t.Request\_at BETWEEN '2013-10-01' AND '2013-10-03'

    AND u1.Banned = 'No'

    AND u2.Banned = 'No'

GROUP BY Request\_at

ORDER BY Request\_at

511

(A,B)

SELECT player\_id,

min(event\_date) AS first\_login

FROM Activity

GROUP BY player\_id

ORDER BY player\_id

512

(A)

SELECT a.player\_id,

a.device\_id

FROM Activity a

INNER JOIN (

    SELECT player\_id,

    MIN(event\_date) AS Min

    FROM Activity

    GROUP BY player\_id

) b ON a.player\_id = b.player\_id

    AND a.event\_date = b.Min

ORDER BY player\_id

534

(A)

SELECT a.player\_id,

a.event\_date,

SUM(b.games\_played) AS games\_played\_so\_far

FROM Activity a

INNER JOIN Activity b ON a.player\_id = b.player\_id

    AND b.event\_date <= a.event\_date

GROUP BY a.player\_id, a.event\_date

(B)

SELECT player\_id,

event\_date,

SUM(games\_played) OVER(PARTITION BY player\_id ORDER BY event\_date) AS 'games\_played\_so\_far'

FROM Activity

ORDER BY player\_id, games\_played\_so\_far

550（A）

SELECT

ROUND(COUNT(DISTINCT b.player\_id)/ COUNT(DISTINCT a.player\_id),2) AS fraction

FROM Activity a

LEFT JOIN (

        SELECT player\_id,

        min(event\_date) AS event\_date

        FROM Activity

        GROUP BY player\_id

    ) b ON a.player\_id = b.player\_id

        AND DATEDIFF(a.event\_date, b.event\_date) = 1

569（A）

SELECT Id, Company, Salary

FROM Employee e

WHERE ABS((SELECT count(\*) FROM Employee e1 WHERE e.Company = e1.Company AND e.Salary >= e1.Salary)-(SELECT count(\*) FROM Employee e2 WHERE e.Company = e2.Company AND e.Salary <= e2.Salary)) <= 1

GROUP BY Company, Salary

570（A）

SELECT b.Name

FROM (SELECT ManagerId

    FROM Employee

    GROUP BY ManagerId

    HAVING count(ManagerId)>=5

    ) a

INNER JOIN Employee b ON a.ManagerId = b.Id

571(A)

SELECT Avg(n1.Number) AS median

FROM Numbers n1

WHERE ABS((SELECT SUM(Frequency) FROM Numbers WHERE n1.Number <= Number) -(SELECT SUM(Frequency) FROM Numbers WHERE n1.Number >= Number))<= n1.Frequency

574 (A)

SELECT c.Name

FROM Candidate c

INNER JOIN (

    SELECT CandidateId

    FROM Vote

    GROUP BY CandidateId

    ORDER BY COUNT(CandidateId) DESC

    LIMIT 1

) b ON c.Id = b.CandidateId

SELECT c.Name

FROM Candidate c

INNER JOIN Vote v ON c.Id = v.CandidateId

GROUP BY c.Id

ORDER BY COUNT(c.Id) DESC

LIMIT 1

577(A)

SELECT e.Name, b.bonus

FROM Employee e

LEFT JOIN Bonus b ON e.EmpID = b.EmpID

WHERE b.bonus <1000 OR b.Bonus IS NULL

578(A)

SELECT a.question\_id AS survey\_log

FROM (

    SELECT question\_id,

    SUM(CASE WHEN action = 'show' THEN 1 ELSE 0 END) AS num\_show,

    SUM(CASE WHEN action = 'answer' THEN 1 ELSE 0 END) AS num\_ans

    FROM survey\_log

    GROUP BY question\_id) a

ORDER BY (num\_ans/num\_show) DESC

LIMIT 1

579

(A)

SELECT e1.Id,

e1.month,

Sum(e2.Salary) AS Salary

FROM Employee e1

INNER JOIN Employee e2 ON e1.Id = e2.Id

    AND (e1.month-e2.month) BETWEEN 0 AND 2

where (e1.id,e1.month) not in (select id,max(month) from employee group by id)

GROUP BY e1.Id,e1.month

ORDER BY e1.Id,e1.month DESC

(B)

SELECT Id,

Month,

Salary

FROM (SELECT Id,

Month,

SUM(Salary) OVER (PARTITION BY Id ORDER BY Month ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) AS Salary,

ROW\_NUMBER() OVER (PARTITION BY Id ORDER BY Month DESC) rn

FROM Employee

) cte

WHERE rn>1

ORDER BY Id, Month DESC

580(A)

SELECT d.dept\_name, COUNT(s.student\_id) AS student\_number

FROM department d

LEFT JOIN Student s ON d.dept\_id = s.dept\_id

GROUP BY d.dept\_name

ORDER BY COUNT(s.student\_id) DESC, d.dept\_name

584(A)

SELECT Name

FROM customer

WHERE IFNULL(referee\_id,0) <> 2

SELECT Name

FROM customer

WHERE COALESCE(referee\_id,0) <> 2

SELECT Name

FROM customer

WHERE referee\_id <> 2 OR referee\_id IS NULL

585(A)

SELECT sum(TIV\_2016) AS TIV\_2016

FROM insurance

WHERE TIV\_2015 IN (

    SELECT TIV\_2015

    FROM insurance

    GROUP BY TIV\_2015

    HAVING COUNT(TIV\_2015)>1

) AND CONCAT(LAT,LON) IN (

    SELECT CONCAT(LAT,LON)

    FROM insurance

    GROUP BY CONCAT(LAT,LON)

    HAVING COUNT(CONCAT(LAT,LON))=1

)

586(A)

SELECT customer\_number

FROM orders

GROUP BY customer\_number

ORDER BY COUNT(order\_number) DESC

LIMIT 1

595(A)

SELECT Name,

Population,

Area

FROM World

WHERE area>3000000 OR population>25000000

ORDER BY Name

596(A)

SELECT Class

FROM courses

GROUP BY Class

HAVING COUNT(DISTINCT Student)>=5

597(A)

SELECT IFNULL(ROUND(COUNT(DISTINCT r.requester\_id,r.accepter\_id)/ COUNT(DISTINCT f.Sender\_id, f.send\_to\_id) ,2),0) AS accept\_rate

FROM friend\_request f ,request\_accepted r

601(A)

SELECT s1.id, s1.visit\_date, s1.people

FROM stadium s1

LEFT JOIN stadium s2 ON s1.id - 2 = s2.id

LEFT JOIN stadium s3 ON s1.id - 1 = s3.id

LEFT JOIN stadium s4 ON s1.id + 1 = s4.id

LEFT JOIN stadium s5 ON s1.id + 2 = s5.id

WHERE s1.people>=100

AND ((s2.people>=100 AND s3.people>=100)

    OR(s3.people>=100 AND s4.people>=100)

    OR(s4.people>=100 AND s5.people>=100))

ORDER BY s1.id

(B)

SELECT id,

visit\_date,

people

FROM (

    SELECT id,

    visit\_date,

    people,

    lag(people,2) OVER (ORDER BY id) AS prep2,

    lag(people,1) OVER (ORDER BY id) AS prep1,

    lead(people,1) OVER (ORDER BY id) AS next1,

    lead(people,2) OVER (ORDER BY id) AS next2

    FROM stadium

) a

WHERE people>=100

    AND ((prep2>=100 AND prep1>=100)

    OR (prep1>=100 AND next1>=100)

    OR (next1>=100 AND next2>=100))

ORDER BY id

602(A)

SELECT id,count(id) AS num

FROM (

    SELECT requester\_id AS id

    FROM request\_accepted

    UNION ALL

    SELECT accepter\_id AS id

    FROM request\_accepted

) cte

GROUP BY id

ORDER BY count(id) DESC

LIMIT 1

603(A)

SELECT DISTINCT c1.seat\_id

FROM cinema c1

INNER JOIN cinema c2 ON ABS(c1.seat\_id - c2.seat\_id) = 1

WHERE c1.free = 1 AND c2.free = 1

ORDER BY c1.seat\_id

(B)when tbl is subquery, need alias!

SELECT seat\_id

FROM (SELECT seat\_id,

    free,

    lag(free) OVER (ORDER BY seat\_id) AS LastSeat,

    lead(free) OVER (ORDER BY seat\_id) AS NextSeat

    FROM cinema

) a

WHERE free = 1 AND (LastSeat = 1 OR NextSeat = 1)

607(A)

SELECT Name

FROM salesperson

WHERE sales\_id NOT IN (

    SELECT DISTINCT o.sales\_id

    FROM orders o

    INNER JOIN company c ON o.com\_id = c.com\_id

    WHERE c.name = 'Red'

)

608(A)

SELECT t1.id,

CASE WHEN t1.p\_id IS NOT NULL AND t2.p\_id IS NOT NULL THEN 'Inner'

     WHEN t1.p\_id IS NOT NULL AND t2.p\_id IS NULL THEN 'Leaf'

     ELSE 'Root' END AS Type

FROM tree t1

LEFT JOIN tree t2 ON t1.id = t2.p\_id

GROUP BY t1.id

610(A)

SELECT x,

y,

z,

CASE WHEN x+y>z AND x+z>y AND y+z>x THEN 'Yes'

ELSE 'No' END AS triangle

FROM triangle

612(A)  
SELECT ROUND(SQRT(MIN(POW(p1.x-p2.x,2)+POW(p1.y-p2.y,2))),2) AS shortest

FROM point\_2d p1

INNER JOIn point\_2d p2 ON p1.x <> p2.x OR p1.y <> p2.y

613(A)

SELECT MIN(ABS(p1.x-p2.x)) AS shortest

FROM point p1

INNER JOIN point p2 ON p1.x <> p2.x

614(A)

SELECT f1.follower,

COUNT(DISTINCT f2.follower) AS num

FROM follow f1

INNER JOIN follow f2 ON f1.follower = f2.followee

GROUP BY f1.follower

618(A)

SELECT am.name AS America, a.name AS Asia, eu.name AS Europe

FROM (SELECT @am:=0,@as:=0,@eu:=0) t,

(SELECT name, @am:=@am+1 AS id FROM student WHERE continent = 'America' ORDER BY name) am

LEFT JOIN (SELECT name, @as:=@as+1 AS id FROM student WHERE continent = 'Asia' ORDER BY name) a ON am.id = a.id

LEFT JOIN (SELECT name, @eu:=@eu+1 AS id FROM student WHERE continent = 'Europe' ORDER BY name) eu ON am.id = eu.id

ORDER BY am.id

(B)

SELECT am.name AS America, a.name AS Asia, eu.name AS Europe

FROM

(SELECT name, ROW\_NUMBER() OVER(ORDER BY Name) AS id FROM student WHERE continent = 'America') am

LEFT JOIN (SELECT name, ROW\_NUMBER() OVER(ORDER BY Name) AS id FROM student WHERE continent = 'Asia') a ON am.id = a.id

LEFT JOIN (SELECT name, ROW\_NUMBER() OVER(ORDER BY Name) AS id FROM student WHERE continent = 'Europe') eu ON am.id = eu.id

ORDER BY am.id

619(A) max() can take care of null

SELECT IFNULL((SELECT num

FROM my\_numbers

GROUP BY num

HAVING COUNT(num) = 1

ORDER BY num DESC

LIMIT 1),null) AS num

SELECT max(num)AS num

FROM (

    SELECT num

    FROM my\_numbers

    GROUP BY num

    HAVING COUNT(num) = 1

    ORDER BY num DESC

    LIMIT 1

) a

620(A)

SELECT id,

movie,

description,

rating

FROM cinema

WHERE id%2 <>0

AND description <> 'boring'

ORDER BY rating DESC

626(A)

SELECT CASE WHEN id%2 = 0 THEN id-1

     WHEN id%2 = 1 AND id<(SELECT COUNT(\*) FROM SEAT) THEN id+1

     ELSE id END AS id,

student

FROM seat

ORDER BY id

1045（A）

SELECT DISTINCT customer\_id

FROM customer

GROUP BY customer\_id

HAVING COUNT(DISTINCT product\_key) = (SELECT COUNT(DISTINCT product\_key) FROM Product)

ORDER BY customer\_id

1050（A）

SELECT actor\_id, director\_id

FROM ActorDirector

GROUP BY actor\_id, director\_id

HAVING COUNT(\*)>=3

1068(A)

SELECT p.product\_name,

s.year,

s.price

FROM Sales s

INNER JOIN Product p ON s.product\_id = p.product\_id

GROUP BY p.product\_name,

s.year,

s.price

1069(A)

SELECT product\_id, sum(quantity) AS total\_quantity

FROM Sales

GROUP BY product\_id

ORDER BY product\_id

1070(A)

SELECT cte.product\_id,

cte.first\_year,

s.quantity,

s.price

FROM

(

SELECT product\_id, min(year) AS first\_year

FROM Sales

GROUP BY product\_id

) cte

LEFT JOIN Sales s ON cte.product\_id = s.product\_id

    AND cte.first\_year = s.year

ORDER BY product\_id

(B)

SELECT product\_id,

first\_year,

quantity,

price

FROM (

    SELECT product\_id,

    year AS first\_year,

    quantity,

    price,

    ROW\_NUMBER() OVER (PARTITION BY product\_id ORDER BY year) AS rn

    FROM Sales

) cte

WHERE rn = 1

1075(A)

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

ORDER BY p.project\_id

(B)

SELECT project\_id

FROM Project

GROUP BY project\_id

HAVING COUNT(employee\_id) = (SELECT count(employee\_id) FROM Project GROUP BY project\_id ORDER BY count(employee\_id) DESC LIMIT 1)

ORDER BY project\_id

1076(B)

SELECT project\_id,

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 rn

    FROM Project p

    INNER JOIN Employee e ON p.employee\_id = e.employee\_id

) cte

WHERE rn = 1

ORDER BY project\_id,employee\_id

1082(A)

SELECT seller\_id

FROM Sales

GROUP BY seller\_id

HAVING sum(price) = (SELECT sum(price) FROM Sales GROUP BY seller\_id ORDER BY sum(price) DESC LIMIT 1)

ORDER BY seller\_id

(B)

SELECT seller\_id

FROM (

    SELECT seller\_id,

    rank() OVER (ORDER BY SUM(price) DESC) AS rn

    FROM Sales

    GROUP BY seller\_id

) cte

WHERE rn = 1

ORDER BY seller\_id

1083(A)

SELECT s.buyer\_id

FROM Sales s

INNER JOIN Product p ON s.product\_id = p.product\_id

GROUP BY s.buyer\_id

HAVING SUM(IF(p.product\_name='S8',1,0))>0

    AND SUM(IF(p.product\_name='iPhone',1,0))=0

1084(a)

SELECT p.product\_id,

p.product\_name

FROM product p

INNER JOIN Sales s ON p.product\_id = s.product\_id

GROUP BY p.product\_id

HAVING min(sale\_date) >='2019-01-01' AND max(sale\_date)<='2019-03-31'

1097(A)

SELECT cte.install\_dt,

COUNT(cte.player\_id) AS installs,

ROUND(COUNT(a.event\_date)/ COUNT(cte.player\_id),2) AS Day1\_retention

FROM (

    SELECT player\_id,

    MIN(event\_date) as install\_dt

    FROM Activity

    GROUP BY player\_id

) cte

LEFT JOIN Activity a ON cte.player\_id = a.player\_id

    AND DATEDIFF(a.event\_date,cte.install\_dt) = 1

GROUP BY cte.install\_dt

ORDER BY cte.install\_dt

615(A)

SELECT d.pay\_month,

d.department\_id,

CASE WHEN d.AVG\_DEP > c.AVG\_COM THEN 'higher'

     WHEN d.AVG\_DEP = c.AVG\_COM THEN 'same'

     ELSE 'lower'

     END AS comparison

FROM (SELECT date\_format(s.pay\_date,'%Y-%m') AS pay\_month,

    e.department\_id,

    AVG(s.amount) AS AVG\_DEP

    FROM Salary s

    INNER JOIN employee e ON s.employee\_id = e.employee\_id

    GROUP BY pay\_month, e.department\_id) d

INNER JOIN (SELECT date\_format(pay\_date,'%Y-%m') AS pay\_month,

    AVG(amount) AS AVG\_COM

    FROM salary

    GROUP BY pay\_month

) c ON d.pay\_month = c.pay\_month

1098(A)

SELECT b.book\_id, b.name

FROM Books b

LEFT JOIN Orders o ON b.book\_id = o.book\_id

    AND o.dispatch\_date BETWEEN '2018-06-23' AND '2019-06-23'

WHERE DATEDIFF('2019-06-23', b.available\_from)>30

GROUP BY b.book\_id

HAVING SUM(IFNULL(o.quantity,0)) <10

ORDER BY b.book\_id

 在join的时候设定时间否则where 会不能保证id的完整性

1107（A）

SELECT login\_date, count(user\_id) AS user\_count

FROM (

    SELECT user\_id,min(activity\_date) AS login\_date

    FROM Traffic

    WHERE activity = 'login'

    GROUP BY user\_id

) a

WHERE DATEDIFF('2019-06-30',login\_date) <= 90

GROUP BY login\_date

ORDER BY login\_Date

1112

(A)

SELECT student\_id,

MIN(course\_id) AS course\_id,

grade

FROM Enrollments

WHERE (student\_id,grade) IN (SELECT student\_id,max(grade)AS grade FROM Enrollments GROUP BY student\_id)

GROUP BY student\_id

(B)

SELECT student\_id,

course\_id,

grade

FROM (

    SELECT student\_id,

    course\_id,

    grade,

    row\_number() OVER (PARTITION BY student\_id ORDER BY grade DESC, course\_id) AS rn

    FROM Enrollments

) cte

WHERE rn = 1

1113(A)

SELECT extra AS report\_reason,

COUNT(DISTINCT post\_id) AS report\_count

FROM Actions

WHERE action\_date = '2019-07-04'

    AND action = 'report'

    AND extra is not null

GROUP BY extra

1126(A)

SELECT e.business\_id

FROM Events e

INNER JOIN (SELECT event\_type,

    avg(occurences) AS AVG\_OCC

    FROM Events

    GROUP BY event\_type

) cte ON e.event\_type = cte.event\_type

WHERE e.occurences > cte.AVG\_OCC

GROUP BY e.business\_id

HAVING COUNT(\*) >1

1127(A)

SELECT d.spend\_date,

p.platform,

IFNULL(SUM(amount),0) AS total\_amount,

IFNULL(COUNT(DISTINCT user\_id),0) AS total\_users

FROM (SELECT DISTINCT spend\_date FROM Spending) d

JOIN (SELECT DISTINCT platform FROM Spending UNION SELECT 'both') p

LEFT JOIN (SELECT user\_id, spend\_date,

    CASE WHEN COUNT(DISTINCT platform) = 2 THEN 'both'

    WHEN COUNT(DISTINCT platform) = 1 AND platform = 'mobile' THEN 'mobile'

    WHEN COUNT(DISTINCT platform) = 1 AND platform = 'desktop' THEN 'desktop'

    END AS platform,

    SUM(amount) AS amount

    FROM Spending

    GROUP BY 1,2) u ON d.spend\_date = u.spend\_date AND p.platform = u.platform

    GROUP BY 1,2

ORDER BY d.spend\_date, FIELD(p.platform,'desktop','mobile','both')

(A)

SELECT d.spend\_date,

p.platform,

IFNULL(SUM(s.amount),0) AS total\_amount,

IFNULL(COUNT(s.user\_id),0) AS total\_users

FROM (SELECT DISTINCT spend\_date FROM spending) d

JOIN (SELECT 'desktop' AS platform UNION SELECT 'mobile' AS platform UNION SELECT 'both' AS platform) p

LEFT JOIN (SELECT spend\_date,

           user\_id,

           CASE WHEN COUNT(DISTINCT platform) = 2 THEN 'both'

                WHEN COUNT(DISTINCT platform) = 1 AND platform = 'mobile' THEN 'mobile'

                WHEN COUNT(DISTINCT platform) = 1 AND platform = 'desktop' THEN 'desktop'

                END AS platform,

           SUM(amount) AS amount

           FROM Spending

           GROUP BY 1,2) s ON d.spend\_date = s.spend\_date AND p.platform = s.platform

GROUP BY 1,2

ORDER BY 1, FIELD(P.platform,'desktop','mobile','both')

1132（A）

left join并不会影响主表，也就是说，无论LEFT JOIN  on后面什么条件，主表都会被查出来，所以说，这里的限制条件没用。

SELECT ROUND(100\*AVG(cte.daily\_percent),2) AS average\_daily\_percent

FROM (SELECT a.action\_date,

    IFNULL(COUNT(DISTINCT r.post\_id),0)/COUNT(DISTINCT a.post\_id) AS daily\_percent

    FROM Actions a

    LEFT JOIN Removals r ON a.post\_id = r.post\_id

        AND a.action\_date< r.remove\_date

    WHERE a.action ='report'

        AND a.extra = 'spam'

    GROUP BY a.action\_date

) cte

1137(B)

WITH cte1 AS (SELECT fail\_date AS date,

'failed' AS period\_state

FROM Failed

WHERE fail\_date BETWEEN '2019-01-01' AND '2019-12-31'

UNION ALL

SELECT success\_date AS date,

'succeeded' AS period\_state

FROM Succeeded

WHERE success\_date BETWEEN '2019-01-01' AND '2019-12-31'

), cte2 AS(SELECT date,

           period\_state,

           ROW\_NUMBER() OVER(PARTITION BY period\_state ORDER BY date) rn

           FROM cte1

)

SELECT period\_state,

MIN(date) AS start\_date,

MAX(date) AS end\_date

FROM cte2

GROUP BY period\_state, DATEDIFF(day,rn,date)

ORDER BY start\_date

1141(A)

SELECT activity\_date AS day,

count(DISTINCT user\_id) AS active\_users

FROM Activity

WHERE DATEDIFF('2019-07-27', activity\_date) <30

    AND activity\_type IS NOT NUll

GROUP BY activity\_date

1142(A)

select round(ifnull(count(distinct session\_id)/count(distinct user\_id),0),2) as average\_sessions\_per\_user

from activity

where datediff('2019-07-27',activity\_date) <30

1148(A)

SELECT author\_id AS id

FROM Views

WHERE author\_id = viewer\_id

GROUP BY author\_id

ORDER BY author\_id

1149(A)

SELECT DISTINCT viewer\_id AS id

FROM Views

GROUP BY viewer\_id, view\_date

HAVING COUNT(DISTINCT article\_id) >1

ORDER BY viewer\_id

1158(A)

SELECT u.user\_id AS buyer\_id,

u.join\_date,

count(o.order\_id) AS orders\_in\_2019

FROM Users u

LEFT JOIN Orders o ON u.user\_id = o.buyer\_id

    AND o.order\_date BETWEEN '2019-01-01' AND '2019-12-31'

GROUP BY u.user\_id

1159

(A)

SELECT u.user\_id AS seller\_id,

IF(u.favorite\_brand = i.item\_brand,'yes','no') AS '2nd\_item\_fav\_brand'

FROM Users u

LEFT JOIN (

    SELECT o1.seller\_id,

    o1.item\_id,

    o1.order\_date

    FROM Orders o1

    INNER JOIN Orders o2 ON o1.seller\_id = o2.seller\_id

        AND o1.order\_date>o2.order\_date

    GROUP BY o1.seller\_id,o1.order\_date

    HAVING COUNT(\*) = 1

) cte ON u.user\_id = cte.seller\_id

LEFT JOIN Items i ON cte.item\_id = i.item\_id

(B)

SELECT u.User\_id AS seller\_id,

CASE WHEN u.favorite\_brand = cte.item\_brand THEN 'yes' ElSE 'no' END AS  '2nd\_item\_fav\_brand'

FROM Users u

LEFT JOIN (

    SELECT seller\_id,

    item\_brand

    FROM (

        SELECT o.seller\_id,

        i.item\_brand,

        ROW\_NUMBER() OVER (Partition by o.seller\_id Order by o.order\_date) rn

        FROM Orders o

        INNER JOIN Items i ON o.item\_id = i.item\_id

    ) a

    WHERE a.rn = 2

) cte ON u.user\_id = cte.seller\_id

1164

(A)

SELECT DISTINCT p.product\_id,

COALESCE(cte.new\_price,10) AS price

FROM Products p

LEFT JOIN(SELECT product\_id,

    new\_price

    FROM Products

    WHERE (product\_id,change\_Date) IN (SELECT product\_id,max(change\_date)    FROM Products WHERE change\_Date<='2019-08-16' GROUP BY product\_id)

    GROUP BY product\_id

) cte ON p.product\_id = cte.product\_id

(B)

SELECT DISTINCT p.product\_id,

COALESCE(cte.new\_price, 10) AS price

FROM Products p

LEFT JOIN (SELECT  product\_id,

    new\_price,

    change\_date

    FROM(SELECT product\_id,

        new\_price,

        change\_date,

        ROW\_NUMBER() OVER(Partition By product\_id Order BY change\_date DESC) as rn

        FROM Products

        WHERE change\_date<='2019-08-16'

    ) a

    WHERE rn = 1

) cte ON p.product\_id = cte.product\_id

SELECT DISTINCT p.product\_id,

COALESCE(cte.new\_price, 10) AS price

FROM Products p

LEFT JOIN (SELECT product\_id,

        new\_price,

        change\_date,

        ROW\_NUMBER() OVER(Partition By product\_id Order BY change\_date DESC) as rn

        FROM Products

        WHERE change\_date<='2019-08-16'

    ) cte ON p.product\_id = cte.product\_id

AND rn = 1

1173

SELECT ROUND(100 \* SUM(order\_date = customer\_pref\_delivery\_date) / count(\*),2) AS immediate\_percentage

FROM Delivery

SELECT ROUND(100 \* AVG(order\_date = customer\_pref\_delivery\_date),2) AS immediate\_percentage

FROM Delivery

1174

(A)

SELECT ROUND(100\*Avg(order\_date = customer\_pref\_delivery\_date),2) AS immediate\_percentage

FROM (

    SELECT delivery\_id,

    order\_date,

    customer\_pref\_delivery\_date

    FROM Delivery

    WHERE (customer\_id,order\_date) IN (

        SELECT customer\_id,

        min(order\_date)

        FROM Delivery

        GROUP BY customer\_id)

) cte

1179 null not ‘null’

SELECT id,

Max(CASE WHEN month = 'Jan' THEN revenue ELSE null END) AS Jan\_Revenue,

Max(CASE WHEN month = 'Feb' THEN revenue ELSE null END) AS Feb\_Revenue,

Max(CASE WHEN month = 'Mar' THEN revenue ELSE null END) AS Mar\_Revenue,

Max(CASE WHEN month = 'Apr' THEN revenue ELSE null END) AS Apr\_Revenue,

Max(CASE WHEN month = 'May' THEN revenue ELSE null END) AS May\_Revenue,

Max(CASE WHEN month = 'Jun' THEN revenue ELSE null END) AS Jun\_Revenue,

Max(CASE WHEN month = 'Jul' THEN revenue ELSE null END) AS Jul\_Revenue,

Max(CASE WHEN month = 'Aug' THEN revenue ELSE null END) AS Aug\_Revenue,

Max(CASE WHEN month = 'Sep' THEN revenue ELSE null END) AS Sep\_Revenue,

Max(CASE WHEN month = 'Oct' THEN revenue ELSE null END) AS Oct\_Revenue,

Max(CASE WHEN month = 'Nov' THEN revenue ELSE null END) AS Nov\_Revenue,

Max(CASE WHEN month = 'Dec' THEN revenue ELSE null END) AS Dec\_Revenue

FROM Department

GROUP BY id

ORDER BY id

1193(A)

SELECT t.month,

t.country,

t.trans\_count,

IFNULL(a.approved\_count,0) AS approved\_count,

t.trans\_total\_amount,

IFNULL(a.approved\_total\_amount,0) AS approved\_total\_amount

FROM (SELECT DATE\_FORMAT(trans\_Date,'%Y-%m') AS month,

country,

count(id) AS trans\_count,

sum(amount) AS trans\_total\_amount

FROM Transactions

GROUP BY month,country

) AS t

LEFT JOIN (SELECT DATE\_FORMAT(trans\_Date,'%Y-%m') AS month,

country,

count(id) AS approved\_count,

sum(amount) AS approved\_total\_amount

FROM Transactions

WHERE state='approved'

GROUP BY month,country

) AS a ON t.month = a.month

    AND t.country = a.country

GROUP BY t.month, t.country

(A)

SELECT date\_format(trans\_Date,'%Y-%m') 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

1194

(A)

SELECT group\_id, min(player\_id) AS player\_id

FROM (SELECT p.player\_id,

p.group\_id,

sum(IF(p.player\_id = m.first\_player,first\_score,0))+sum(IF(p.player\_id = m.second\_player,second\_score,0)) AS score

FROM Players p

LEFT JOIN Matches m ON p.player\_id = m.first\_player

    OR p.player\_id = m.second\_player

GROUP BY p.player\_id) cte1

WHERE (group\_id,score) IN (SELECT group\_id,max(score)

     FROM (SELECT p.player\_id,

    p.group\_id,

    sum(IF(p.player\_id = m.first\_player,first\_score,0)) +   sum(IF(p.player\_id = m.second\_player,second\_score,0)) AS score

    FROM Players p

    LEFT JOIN Matches m ON p.player\_id = m.first\_player

        OR p.player\_id = m.second\_player

    GROUP BY p.player\_id) cte2

    GROUP BY group\_id

    )

GROUP BY group\_id

--where 里面的subquery不用alias

(B)

SELECT group\_id, player\_id

FROM (

SELECT

group\_id,

p.player\_id,

ROW\_NUMBER() OVER(PARTITION BY group\_id ORDER BY t2.score DESC, t2.player\_id) rn

FROM Players p

LEFT JOIN (

    SELECT player\_id, sum(score) AS score

    FROM (SELECT first\_player AS player\_id,

        first\_score AS score

        FROM Matches

        UNION ALL

        SELECT second\_player AS player\_id,

        second\_score AS score

        FROM Matches) t1

    GROUP BY player\_id) t2 ON p.player\_id = t2.player\_id

) t3

WHERE rn = 1

1204(A)

SELECT q1.person\_name

FROM Queue q1

INNER JOIN Queue q2 ON q1.turn >= q2.turn

GROUP BY q1.turn

HAVING Sum(q2.weight)<= 1000

ORDER BY Sum(q2.weight) DESC

LIMIT 1

1205

SELECT month,

country,

SUM(CASE WHEN state = "approved" THEN 1 ELSE 0 END) AS approved\_count, SUM(CASE WHEN state = "approved" THEN amount ELSE 0 END) AS approved\_amount,

SUM(CASE WHEN state = "back" THEN 1 ELSE 0 END) AS chargeback\_count, SUM(CASE WHEN state = "back" THEN amount ELSE 0 END) AS chargeback\_amount

FROM

(

    SELECT LEFT(chargebacks.trans\_date, 7) AS month, country, "back" AS state, amount

    FROM chargebacks

    JOIN transactions ON chargebacks.trans\_id = transactions.id

    UNION ALL

    SELECT LEFT(trans\_date, 7) AS month, country, state, amount

    FROM transactions

    WHERE state = "approved"

) s

GROUP BY month, country

1211(A)

SELECT query\_name,

ROUND(AVG(rating/position),2) AS quality,

ROUND(100\*SUM(rating<3)/COUNT(query\_name),2) AS poor\_query\_percentage

FROM Queries

GROUP BY query\_name

1212(A)

SELECT t.team\_id,

t.team\_name,

IFNULL(SUM(cte.points),0) AS num\_points

FROM Teams t

LEFT JOIN(

SELECT match\_id,

host\_team AS team\_id,

CASE WHEN host\_goals > guest\_goals THEN 3

     WHEN host\_goals = guest\_goals THEN 1

     ELSE 0 END AS points

FROM Matches

UNION ALL

SELECT match\_id,

guest\_team AS team\_id,

CASE WHEN host\_goals < guest\_goals THEN 3

     WHEN host\_goals = guest\_goals THEN 1

     ELSE 0 END AS points

FROM Matches

) cte ON cte.team\_id = t.team\_id

GROUP BY t.team\_id

ORDER BY num\_points DESC, t.team\_id

SELECT t.team\_id,

t.team\_name,

SUM(CASE WHEN t.team\_id = m.host\_team AND m.host\_goals>m.guest\_goals THEN 3

   WHEN t.team\_id = m.guest\_team AND m.host\_goals<m.guest\_goals THEN 3

   WHEN m.host\_goals=m.guest\_goals THEN 1

   ELSE 0 END) AS num\_points

FROM Teams t

LEFT JOIN Matches m ON t.team\_id = m.host\_team

    OR t.team\_id = m.guest\_team

GROUP BY t.team\_id

ORDER BY num\_points DESC, t.team\_id

1241.(A)

SELECT s1.sub\_id AS post\_id,

COUNT(DISTINCT s2.sub\_id) AS number\_of\_comments

FROM Submissions s1

LEFT JOIN Submissions s2 ON s1.sub\_id = s2.parent\_id

WHERE s1.parent\_id IS NULL

GROUP BY post\_id

ORDER BY post\_id

1251(A)

SELECT u.product\_id,

ROUND(SUM(u.units\*p.price)/SUM(u.units),2) AS average\_price

FROM UnitsSold u

INNER JOIN Prices p ON u.product\_id = p.product\_id

    AND u.purchase\_date BETWEEN p.start\_date AND p.end\_date

GROUP BY u.product\_id

1264(A) OR 的时候注意括号

SELECT DISTINCT l.page\_id AS recommended\_page

FROM Likes l

LEFT JOIN Friendship f ON l.user\_id = f.user1\_id

    OR l.user\_id = f.user2\_id

WHERE (f.user1\_id = 1 OR f.user2\_id = 1)

    AND l.user\_id <> 1

    AND l.page\_id NOT IN (SELECT DISTINCT page\_id FROM Likes WHERE user\_id = 1)

1270（A）

SELECT e1.employee\_id

FROM Employees e1

INNER JOIN Employees e2 ON e1.manager\_id = e2.employee\_id

INNER JOIN Employees e3 ON e2.manager\_id = e3.employee\_id

WHERE e3.manager\_id = 1 AND e1.employee\_id != 1

1280(A)

SELECT s.student\_id,

s.student\_name,

sub.subject\_name,

COUNT(e.subject\_name) AS attended\_exams -- 有where 子句要加ifnull才能显示为0的值

FROM (Students s, Subjects sub)

LEFT JOIN Examinations e ON s.student\_id = e.student\_id

    AND sub.subject\_name = e.subject\_name

GROUP BY s.student\_id, sub.subject\_name

ORDER BY s.student\_id, sub.subject\_name

1285(A)

SELECT l1.log\_id AS start\_id,

    MIN(l2.log\_id) AS end\_id

FROM (SELECT log\_id FROM Logs

     WHERE log\_id-1 NOT IN (SELECT log\_id FROM Logs) ) l1,

     (SELECT log\_id FROM Logs

     WHERE log\_id+1 NOT IN (SELECT log\_id FROM Logs) ) l2

WHERE l1.log\_id <= l2.log\_id

GROUP BY l1.log\_id

(B)

SELECT min(log\_id) AS start\_id,

max(log\_id) AS end\_id

FROM (SELECT log\_id,

ROW\_NUMBER() OVER (ORDER BY log\_id) AS rn

FROM Logs

) cte

GROUP BY log\_id-rn

1294(A)

SELECT c.country\_name,

CASE WHEN AVG(w.weather\_state)<=15 THEN 'Cold'

    WHEN AVG(w.weather\_state)>=25 THEN 'Hot'

    ELSE 'Warm' END AS weather\_type

FROM Countries c

INNER JOIN Weather w ON c.country\_id = w.country\_id

WHERE w.day BETWEEN '2019-11-01' AND '2019-11-30'

GROUP BY w.country\_id

1303(A)

SELECT e.employee\_id,

cte.team\_size

FROM Employee e

LEFT JOIN (SELECT team\_id,

    COUNT(employee\_id) AS team\_size

    FROM Employee

    GROUP BY team\_id

) cte ON e.team\_id = cte.team\_id

ORDER BY e.employee\_id

(B)

With cte AS(SELECT team\_id,

COUNT(employee\_id) AS team\_size

FROM Employee

GROUP BY team\_id

)

SELECT e.employee\_id,

cte.team\_size

FROM Employee e

LEFT JOIN cte ON e.team\_id = cte.team\_id

ORDER BY e.employee\_id

1308(A)

SELECT s1.gender,

s1.day,

SUM(s2.score\_points) AS total

FROM Scores s1

LEFT JOIN Scores s2 ON s1.gender=s2.gender

    AND s1.day>=s2.day

GROUP BY s1.gender,s1.day

ORDER BY s1.gender,s1.day

SELECT gender,

day,

SUM(score\_points) OVER(PARTITION BY gender ORDER BY DAY) AS total

FROM Scores

ORDER BY gender,day

1321(A)

SELECT a.visited\_on,

SUM(b.day\_sum) AS amount,

ROUND(AVG(b.day\_sum),2) AS average\_amount

FROM (SELECT visited\_on, SUM(amount) AS day\_sum FROM Customer GROUP BY visited\_on) a

INNER JOIN (SELECT visited\_on, SUM(amount) AS day\_sum FROM Customer GROUP BY visited\_on) b

    ON DATEDIFF(a.visited\_on,b.visited\_on) BETWEEN 0 AND 6

GROUP BY a.visited\_on

HAVING COUNT(b.visited\_on) = 7

ORDER BY a.visited\_on

(B)

SELECT visited\_on,

SUM(SUM(amount)) OVER(ORDER BY visited\_on ROWS BETWEEN 6 preceding and current row) AS amount,

ROUND(SUM(SUM(amount\*1.0)) OVER(ORDER BY visited\_on ROWS BETWEEN 6 preceding and current row)/7.0,2) AS average\_amount

FROM Customer

GROUP BY visited\_on

ORDER BY visited\_on

OFFSET 6 ROWS

1322(A)

SELECT ad\_id,

ROUND(100\*IFNULL(SUM(action = 'Clicked')/SUM(action ='Clicked' OR action = 'Viewed'),0),2) AS ctr

FROM Ads

GROUP BY ad\_id

ORDER BY ctr DESC, ad\_id

1327(A)

SELECT p.product\_name,

SUM(o.unit) AS unit

FROM Orders o

INNER JOIN Products p ON o.product\_id = p.product\_id

WHERE o.order\_date BETWEEN '2020-02-01' AND '2020-02-29'

GROUP BY p.product\_name

HAVING SUM(o.unit)>=100

1336(B)

WITH a AS(

SELECT v.user\_id,

v.visit\_date,

SUM(CASE WHEN t.transaction\_date IS NOT NULL THEN 1 ELSE 0 END) AS transactions\_count

FROM Visits v

LEFT JOIN Transactions t ON v.user\_id = t.user\_id AND v.visit\_date = t.transaction\_date

GROUP BY v.user\_id, v.visit\_date

), b AS (

SELECT transactions\_count,

COUNT(transactions\_count) AS visits\_count

FROM a

GROUP BY transactions\_count)

, c AS (

SELECT 0 AS i,

max(transactions\_count) AS mx

FROM b

UNION ALL

SELECT i+1, mx

FROM c

WHERE i+1<=mx

)

SELECT c.i AS transactions\_count,

ISNULL(b.visits\_count,0) AS visits\_count

FROM c

LEFT JOIN b ON c.i = b.transactions\_count

ORDER BY c.i

1341(A)

(SELECT u.name AS results

FROM Movie\_rating mr

INNER JOIN Users u ON mr.user\_id = u.user\_id

GROUP BY u.name

ORDER BY count(mr.movie\_id) DESC, u.name

LIMIT 1)

UNION ALL

(SELECT m.title AS results

FROM Movie\_rating mr

INNER JOIN Movies m ON mr.movie\_id = m.movie\_id

WHERE LEFT(created\_at,7) = '2020-02'

GROUP BY m.title

ORDER BY AVG(mr.rating) DESC, m.title

LIMIT 1)

Union：对两个结果集进行并集操作，不包括重复行，同时进行默认规则的排序。

Union All：对两个结果集进行并集操作，包括重复行，不进行排序。

(B)

SELECT name AS results

FROM (SELECT name,

    ROW\_NUMBER() OVER(ORDER BY user\_num DESC, name) AS rn

    FROM (SELECT user\_id,

          COUNT(DISTINCT movie\_id) AS user\_num

          FROM Movie\_Rating mr

          GROUP BY user\_id) un

    INNER JOIN Users u ON un.user\_id = u.user\_id) n

WHERE rn = 1

UNION ALL

SELECT title AS results

FROM (SELECT title,

     ROW\_NUMBER() OVER(ORDER BY rating DESC, title) AS rn

     FROM (SELECT movie\_id,

          AVG(rating\*1.0) AS rating 数据库里为INT型，AVG的时候要出来小数必须要转化一下为浮点型

          FROM Movie\_Rating mr

          WHERE LEFT(created\_at,7) = '2020-02'

          GROUP BY movie\_id) r

     INNER JOIN Movies m ON r.movie\_id = m.movie\_id) t

WHERE rn = 1

1350(A)

SELECT id,name

FROM Students

WHERE department\_id NOT IN (SELECT id FROM Departments)

1355(A)

SELECT a.name AS Activity

FROM Activities a

LEFT JOIN Friends f ON a.name = f.activity

GROUP BY a.name

HAVING COUNT(\*) NOT IN

(SELECT max(n)

FROM (SELECT activity, count(\*) AS n

FROM Friends

GROUP BY activity) cte1

UNION ALL

SELECT min(n)

FROM (SELECT activity, count(\*) AS n

FROM Friends

GROUP BY activity) cte1)

(B)

With cte AS (

SELECT Activity,

COUNT(\*) AS n

FROM Friends

GROUP BY Activity

)

SELECT a.name AS Activity

FROM Activities a

LEFT JOIN Friends f ON a.name = f.activity

GROUP BY a.name

HAVING COUNT(\*) NOT IN (SELECT min(n) FROM cte UNION ALL

                       SELECT max(n) FROM cte)

1364

（A）

SELECT i.invoice\_id,

c.customer\_name,

i.price,

IFNULL(COUNT(DISTINCT ct.contact\_email),0) AS contacts\_cnt,

IFNULL(COUNT(DISTINCT c2.email),0) AS trusted\_contacts\_cnt

FROM Customers c

INNER JOIN Invoices i ON c.Customer\_id = i.user\_id

LEFT JOIN Contacts ct ON i.user\_id = ct.user\_id

LEFT JOIN Customers c2 ON ct.contact\_email = c2.email

GROUP BY 1,2,3

(B)

WITH cte\_contacts AS (

SELECT c.customer\_id,

COUNT(ct.user\_id) AS contacts\_cnt

FROM Customers c

INNER JOIN Contacts ct ON c.Customer\_id = ct.User\_id

GROUP BY c.Customer\_id

),

cte\_trusted\_contacts AS (

SELECT c.customer\_id,

COUNT(ct.user\_id) AS trusted\_contacts\_cnt

FROM Customers c

INNER JOIN Contacts ct ON c.Customer\_id = ct.user\_id

INNER JOIN Customers c2 ON ct.contact\_email = c2.email

GROUP BY c.Customer\_id

)

SELECT i.invoice\_id,

c.customer\_name,

i.price,

ISNULL(c1.contacts\_cnt,0) AS contacts\_cnt,

ISNULL(c2.trusted\_contacts\_cnt,0) AS trusted\_contacts\_cnt

FROM Customers c

INNER JOIN Invoices i ON c.Customer\_id = i.user\_id

LEFT JOIN cte\_contacts c1 ON c.Customer\_id = c1.Customer\_id

LEFT JOIN cte\_trusted\_contacts c2 ON c.Customer\_id = c2.Customer\_id

ORDER BY i.invoice\_id