Qusetion 101:

```
create table UserActivity
( username varchar(20),
activity varchar(20),
startdate date,
enddate date);

insert into UserActivity values
('Alice','Travel','2020-02-12','2020-02-20'),
('Alice','Dancing','2020-02-21','2020-02-23'),
('Alice','Travel','2020-02-24','2020-02-28'),
('Bob','Travel','2020-02-11','2020-02-18');
```

```
select * from UserActivity where (username, startDate) in (
    select u1.username, max(u1.startDate) from UserActivity u1
        where (u1.username, u1.startDate) not in (
            select u2.username, max(u2.startDate) from UserActivity u2
                group by u2.username
                having count(u2.username) > 1
        )
        group by u1.username
);

# Approach 2:

select distinct username, activity, startDate, endDate
from
    (select u.*,
           rank() over (partition by username order by startDate desc) as rnk,
           count(activity) over (partition by username) as num
    from UserActivity u) t
where (num <> 1 and rnk = 2) or (num = 1 and rnk = 1);
```

Question 102:

****Same as 101 in Set 3**

Question 103:

****Same as 81 in Set 2**

Question 104:

****Same as 82 in Set 2**

Question 105:

****Same as 83 in Set 2**

Question 106:

****Same as 84 in Set 2**

Question 107:

```
CREATE TABLE EMPLOYEES
(ID int,
name varchar(20),
salary int CONSTRAINT c1_chk CHECK (1000 < salary < 100000)
);

insert into EMPLOYEES values (1,'Kristeen',1420),
(2,'Ashley',2006),
(3,'Julia',2210),
(4,'Maria',3000);
```

```
select ceil(avg(salary) - avg(replace(salary, '0', ''))) from EMPLOYEES;
```

Question 108:

```
create table employee(employee_id int,name varchar(30),months int,salary int);

insert into employee values
(12228,'Rose',15,1968),
(33645,'Angela',1,3443),(45692,'Frank',17,1608),
(56118,'Patrick',7,1345),(59725,'Lisa',11,2330),
(74197,'Kimberly',16,4372),(78454,'Bonnie',8,1771),
(83565,'Michael',6,2017),(98607,'Todd',5,3396),
(99989,'Joe',9,3573);
```

```
select months*salary as salary, count(*) as count from employee
group by salary
order by salary desc
limit 1;
```

Question 109:

```
create table occupations(
name varchar(30),
occupation varchar(30),
check(occupation in ('Doctor', 'Professor', 'Singer','Actor'))
);

insert into occupations values
('Samantha','Doctor'),('Julia','Actor'),
('Maria','Actor'),('Meera','Singer'),
('Ashley','Professor'),('Ketty','Professor'),
('Christeen','Professor'),('Jane','Actor'),
('Jenny','Doctor'),('Priya','Singer');
```

```
select concat(name, '(', substring(occupation, 1, 1), ')') as name
from occupations
order by name;

select concat('There are a total of', ' ', count(occupation), ' ',
lower(occupation), 's.') as profession
from occupations
group by occupation
order by profession;
```

Question 110:

```
create table occupations(
name varchar(30),
occupation varchar(30),
check(occupation in ('Doctor', 'Professor', 'Singer','Actor'))
);

insert into occupations values
('Samantha','Doctor'),('Julia','Actor'),
('Maria','Actor'),('Meera','Singer'),
('Ashley','Professor'),('Ketty','Professor'),
('Christeen','Professor'),('Jane','Actor'),
('Jenny','Doctor'),('Priya','Singer');
```

```
select
    Doctor,
    Professor,
    Singer,
    Actor
from (
    select
        NameOrder,
        max(case Occupation when 'Doctor' then Name end) as Doctor,
        max(case Occupation when 'Professor' then Name end) as Professor,
        max(case Occupation when 'Singer' then Name end) as Singer,
        max(case Occupation when 'Actor' then Name end) as Actor
    from (
            select
                Occupation,
                Name,
                row_number() over(partition by Occupation order by Name ASC) as
NameOrder
            from occupations
         ) as NameLists
    group by NameOrder
    ) as Names;
```

Question 111:

```
create table BST(N int, P int);

insert into BST VALUES
(1,2),(3,2),(6,8),(9,8),(2,5),(8,5),(5,null);
```

```
SELECT N,
CASE
   WHEN P IS NULL THEN 'Root'
   WHEN (SELECT COUNT(*) FROM BST WHERE B.N=P)>0 THEN 'Inner'
   ELSE 'Leaf'
END AS PLACE
FROM BST B
ORDER BY N;
```

Qusetion 112:

```sql
create table company (company_code varchar(10),founder varchar(30));

create table lead_manager (lead_manager_code varchar(30),company_code
varchar(10));

create table senior_manager (senior_manager_code varchar(30),lead_manager_code
varchar(30),company_code varchar(10));

create table manager (manager_code varchar(3),senior_manager_code
varchar(30),lead_manager_code varchar(30),company_code varchar(10));

create table employee1 (employee_code varchar(30),manager_code
varchar(3),senior_manager_code varchar(30),lead_manager_code
varchar(30),company_code varchar(10));

insert into company values('C1','Monika'),
('C2','Samantha');

insert into lead_manager values('LM1','C1'),
('LM2','C2');
insert into senior_manager values ('SM1','LM1','C1'),
('SM2','LM1','C1'),
('SM3','LM2','C2');

insert into manager values ('M1','SM1','LM1','C1'),
('M2','SM3','LM2','C2'),
('M3','SM3','LM2','C2');

insert into employee1 values ('E1','M1','SM1','LM1','C1'),
('E2','M1','SM1','LM1','C1'),
('E3','M2','SM3','LM2','C2'),
('E4','M3','SM3','LM2','C2');
```

```sql
select c.company_code, c.founder,
       count(distinct l.lead_manager_code)lead_manager_count,
       count(distinct s.senior_manager_code)senior_manager_count,
       count(distinct m.manager_code)manager_count,
       count(distinct e.employee_code)employee_count
from company as c
join lead_manager as l
on c.company_code = l.company_code
join senior_manager as s
on l.lead_manager_code = s.lead_manager_code
```

```
join manager as m
on m.senior_manager_code = s.senior_manager_code
join employee1 as e
on e.manager_code = m.manager_code
group by c.company_code, c.founder
order by c.company_code;
```

Question 113:

```
--Approach 1 :

WITH RECURSIVE T (I) AS (SELECT 1 UNION ALL SELECT I+1 FROM T WHERE I < 100)
SELECT group_concat (I SEPARATOR '&') AS LST
FROM
( SELECT A.I FROM T A
  LEFT JOIN T B ON B.I BETWEEN 2 AND A.I - 1
  -- Actually, 1 is prime as well...
  WHERE A.I > 1 GROUP BY A.I
  HAVING COUNT (CASE MOD (A.I, B.I) WHEN 0 THEN 1 END) = 0
  ORDER BY A.I) A;

--Approach 2 :

SELECT GROUP_CONCAT(NUMB SEPARATOR '&')
FROM (
    SELECT @num:=@num+1 as NUMB FROM
    information_schema.tables t1,
    information_schema.tables t2,
    (SELECT @num:=1) tmp
) tempNum
WHERE NUMB<=1000 AND NOT EXISTS(
        SELECT * FROM (
            SELECT @nu:=@nu+1 as NUMA FROM
                information_schema.tables t1,
                information_schema.tables t2,
                (SELECT @nu:=1) tmp1
                LIMIT 1000
            ) tatata
        WHERE FLOOR(NUMB/NUMA)=(NUMB/NUMA) AND NUMA<NUMB AND NUMA>1
    );
```

Question 114:

```sql
-- Approach 1

WITH RECURSIVE cte AS
(
  SELECT 1 AS n, CAST('*' AS CHAR(100)) AS str
  UNION ALL
  SELECT n + 1, concat('* ',str) FROM cte WHERE n < 20
)
SELECT str FROM cte;

-- Approach 2

set @number = 0;
select repeat('* ', @number := @number + 1)
from information_schema.tables
where @number < 20;
```

Question 115:

```sql
WITH Recursive CTE AS (
    SELECT 20 AS counter
    UNION ALL
    SELECT counter - 1
    FROM CTE
    WHERE counter > 0
)
SELECT repeat('* ', counter)
FROM CTE;
```

Question 116:

```sql
create table functions (X int, Y int);

insert into functions values (20,20),
(20,20),(20,21),(23,22),(22,23),(21,20);
```

```sql
SELECT f1.X, f1.Y FROM functions AS f1
WHERE f1.X = f1.Y AND
(SELECT COUNT(*) FROM functions WHERE X = f1.X AND Y = f1.Y) > 1
UNION
SELECT f1.X, f1.Y from functions AS f1
WHERE EXISTS(SELECT X, Y FROM functions WHERE f1.X = Y AND f1.Y = X AND f1.X < X)
ORDER BY X;
```

Question 117:

****Same as 81 in set 2**

Question 118:

****Same as 82 in set 2**

Question 119:

****Same as 83 in set 2**

Question 120:

****Same as 84 in set 2**

Question 121:

****Same as 85 in set 2**

Question 122:

****Same as 86 in set 2**

Question 123:

****Same as 87 in set 2**

Question 124:

****Same as 88 in set 2**

Question 125:

****Same as 89 in set 2**

Question 126:

****Same as 90 in set 2**

Question 127:

****Same as 91 in set 2**

Question 128:

****Same as 92 in set 2**

Question 129:

****Same as 68 in set 2**

Question 130:

****Same as 55 in set 2**

Question 131:

****Same as 95 in set 2**

Question 132:

****Same as 96 in set 2**

Question 133:

****Same as 97 in set 2**

Question 134:

****Same as 98 in set 2**

Question 135:

****Same as 99 in set 2**

Question 136 – Question 149 are repeated in Set 3

Question : 150

```
CREATE TABLE Students (id int, name varchar(20));
CREATE TABLE Packages (id int, salary float);
CREATE TABLE Friends (id int, friend_id int);
```

```
INSERT INTO Students values(1,'Ashley'),(2,'samantha'),(3,'Julia'),(4,'Scarlet');
INSERT INTO Friends values(1,2),(2,3),(3,4),(4,1);
INSERT INTO Packages values(1,15.20),(2,10.06),(3,11.55),(4,12.22);
```

```
SELECT s.Name FROM Students s
JOIN Packages p1 ON s.ID = p1.ID
JOIN Friends f ON s.ID = f.ID
JOIN Packages p2 ON f.Friend_ID = p2.ID
WHERE p2.Salary > p1.Salary
ORDER BY p2.Salary;
```

Question : 151

```
CREATE table hackers (hacker_id int, name varchar(20));
create table difficulty (difficulty_level int , score int);
create table challenges (challenge_id int , hacker_id int, difficulty_level int);
create table submission (submission_id int , hacker_id int, challenge_id int,
score int);
```

```
insert into hackers values (5580,'Rose'),(8439,'Angela'),(27205,'Frank'),
(52243,'Patrick'),(52348,'Lisa'),(57645,'Kimberly'),(77726,'Bonnie'),
(83082,'Michael'),(86870,'Todd'),(90411,'Joe');

insert into difficulty values (1,20),(2,30),(3,40),(4,60),(5,80),(6,100),(7,120);

insert into challenges values(4810,77726,4),(21089,27205,1),(36566,5580,7),
(66730,52243,6),(71055,52243,2);

insert into submission values (68628,77726,36566,30),(65300,77726,21089,10),
(40326,52243,36566,77),(8941,27205,4810,4),(83554,77726,66730,30),
(43353,52243,66730,0),(55385,52348,71055,20),(39784,27205,71055,23),
(94613,86870,71055,30),(45788,52348,36566,0),(93058,86870,36566,30),
(7344,8439,66730,92),(2721,8439,4810,36),(523,5580,71055,4),
(49105,52348,66730,0),(55877,57645,66730,80),(38355,27205,66730,35),
(3924,8439,36566,80),(97397,90411,66730,100),(84162,83082,4810,40),
(97431,90411,71055,30);
```

```
SELECT h.hacker_id,h.name
FROM  hackers h,challenges c ,difficulty d,submission s
WHERE h.hacker_id=s.hacker_id
AND c.challenge_id=s.challenge_id
AND c.difficulty_level=d.difficulty_level
AND s.score=d.score
GROUP BY h.hacker_id,h.name
HAVING  COUNT(h.hacker_id)>1
ORDER BY COUNT(c.challenge_id) DESC, h.hacker_id;
```

Question : 152

```
create table project
(task_id int, start_date date, end_date date);
```

```
insert into project values (1,'2015-10-01','2015-10-02'),
(2,'2015-10-02','2015-10-03'),(3,'2015-10-03','2015-10-04'),
(4,'2015-10-13','2015-10-14'),(5,'2015-10-14','2015-10-15'),
(6,'2015-10-28','2015-10-29'),(7,'2015-10-30','2015-10-31');
```

```
SELECT Start_Date, min(End_Date) as End_Date
FROM
```

```
 (SELECT Start_Date FROM project WHERE Start_Date NOT IN (SELECT End_Date FROM
project)) a ,
 (SELECT End_Date FROM project WHERE End_Date NOT IN (SELECT Start_Date FROM
project)) b
WHERE Start_Date < End_Date
GROUP BY Start_Date
ORDER BY DATEDIFF(min(End_Date), Start_Date) ASC, Start_Date ASC;
```

Using Joins:

```
Select Start_Date, MIN(End_Date) as End_Date
From
    (Select b.Start_Date From project as a RIGHT Join project as b
    ON b.Start_Date = a.End_Date WHERE a.Start_Date IS NULL) sd,
    (Select a.End_Date From project as a Left Join project as b
    ON b.Start_Date = a.End_Date WHERE b.End_Date IS NULL) ed
Where Start_Date < End_Date
GROUP BY Start_Date
ORDER BY datediff(MIN(End_Date), Start_Date), Start_Date;
```

Question 153:

```
create table transactions (user_id int,amount float, transaction_date timestamp);

insert into transactions values (1,9.99,'2022-08-01 10:00:00'),
(1,55,'2022-08-17 10:00:00'),
(2,149.5,'2022-08-05 10:00:00'),
(2,4.89,'2022-08-06 10:00:00'),
(2,34,'2022-08-07 10:00:00');
```

```
SELECT DISTINCT T1.user_id
FROM transactions AS T1
INNER JOIN transactions AS T2
  ON DATE(T2.transaction_date) = DATE(T1.transaction_date) + 1
INNER JOIN transactions AS T3
  ON DATE(T3.transaction_date) = DATE(T1.transaction_date) + 2
ORDER BY T1.user_id;
```

Question 154:

```
create table payments(payer_id int,recipient_id int,amount int);
```

```
insert into payments values (101,201,30),
(201,101,10),
(101,301,20),
(301,101,80),
(201,301,70);
```

```
WITH cte AS (
SELECT payer_id, recipient_id
FROM payments
INTERSECT
SELECT recipient_id, payer_id
FROM payments)

SELECT COUNT(payer_id)/2 AS unique_relationships
FROM cte;
```

Question: 155

```
CREATE table user_logins (user_id int , login_date datetime);

insert into user_logins values (725,'2022-03-03 12:00:00'),
(245,'2022-03-28 12:00:00'),(112,'2022-03-05 12:00:00'),(245,'2022-04-29
12:00:00'),
(112,'2022-04-05 12:00:00');
```

```
SELECT
  MONTH(current_month.login_date)AS curr_month,COUNT(current_month.user_id) AS
reactivated_users
FROM user_logins as current_month
WHERE
  NOT EXISTS (SELECT *
    FROM user_logins AS last_month
    WHERE current_month.user_id = last_month.user_id
      AND month(last_month.login_date)= MONTH(current_month.login_date- 1))
GROUP BY curr_month
ORDER BY curr_month ASC;
```

Question 156:

```
create table user_transactions (transaction_id int, user_id int,
spend float,transaction_date timestamp);

insert into user_transactions values
```

```
(759274,111,49.50,'2022-02-03 00:00:00'),
(850371,111,51.00,'2022-03-15 00:00:00'),
(615348,145,36.30,'2022-03-22 00:00:00'),
(137424,156,151.00,'2022-04-04 00:00:00'),
(248475,156,87.00,'2022-04-16 00:00:00');
```

Using SubQuery:

```
  SELECT COUNT(DISTINCT user_id) AS users
FROM (SELECT
      user_id,
      spend,
      RANK() OVER (
        PARTITION BY user_id
        ORDER BY transaction_date ASC) AS row_num
    FROM user_transactions) AS transactions
WHERE row_num = 1
  AND spend >= 50;
```

Using CTE:

```
WITH transactions AS (
  SELECT
    user_id,
    spend,
    RANK() OVER (
      PARTITION BY user_id
      ORDER BY transaction_date ASC) AS row_num
  FROM user_transactions)
SELECT COUNT(DISTINCT user_id) AS users
FROM transactions
WHERE row_num = 1
  AND spend >= 50;
```

Question 157:

```
create table measurements
(measurement_id int,measurement_value FLOAT,measurement_time datetime);

insert into measurements values
(131233,1109.51,'2022-07-10 09:00:00'),
(135211,1662.74,'2022-07-10 11:00:00'),
(523542,1246.24,'2022-07-10 13:15:00'),
(143562,1124.50,'2022-07-11 15:00:00'),
(346462,1234.14,'2022-07-11 16:45:00');
```

```sql
WITH ranked_measurements AS (
  SELECT
    CAST(measurement_time AS DATE) AS measurement_day, measurement_value,
    ROW_NUMBER() OVER (PARTITION BY CAST(measurement_time AS DATE)
      ORDER BY measurement_time) AS measurement_num
  FROM measurements
) SELECT
  measurement_day,
  round(SUM(
      CASE WHEN measurement_num % 2 != 0 THEN measurement_value
      ELSE 0 END),2) AS odd_sum,
  round(SUM(
      CASE WHEN measurement_num % 2 = 0 THEN measurement_value
      ELSE 0 END),2) AS even_sum
FROM ranked_measurements
GROUP BY measurement_day;
```

Qusetion 159:

```sql
create table rental_amenities
(rental_id  int,amenity varchar(30));

insert into rental_amenities values
(123,'pool'),
(123,'kitchen'),
(234,'hot tub'),
(234,'fireplace'),
(345,'kitchen'),
(345,'pool'),
(456,'pool');
```

```sql
WITH airbnb_amenities AS (
SELECT
  rental_id,
  group_concat(amenity ORDER BY amenity) AS amenities
FROM rental_amenities
GROUP BY rental_id)
SELECT COUNT(*) AS matching_airbnb
FROM airbnb_amenities AS airbnb1 JOIN airbnb_amenities AS airbnb2
ON airbnb1.amenities = airbnb2.amenities
WHERE airbnb1.rental_id > airbnb2.rental_id;
```

Question 160:

```
Create table ad_campaigns
(campaign_id int, spent int,
revenue float, advertiser_id int);

insert into ad_campaigns values (1,5000,7500,3),(2,1000,900,1),
(3,3000,12000,2),(4,500,2000,4),(5,100,400,4);
```

```
SELECT
  advertiser_id,
  ROUND(((SUM(revenue) / SUM(spent))), 2) AS ROAS
FROM ad_campaigns
GROUP BY advertiser_id
ORDER BY advertiser_id;
```

Question 161:

```
create table employee_pay(employee_id int, salary int,title varchar(20));

insert into employee_pay values
(101,80000,'Data Analyst'),
(102,90000,'Data Analyst'),
(103,100000,'Data Analyst'),
(104,30000,'Data Analyst'),
(105,120000,'Data Scientist'),
(106,100000,'Data Scientist'),
(107,80000,'Data Scientist'),
(108,310000,'Data Scientist');
```

```
WITH payout AS (
SELECT
  employee_id,
  salary,
  title,
  (AVG(salary) OVER (PARTITION BY title)) * 2 AS double_average,
  (AVG(salary) OVER (PARTITION BY title)) / 2 AS half_average
FROM employee_pay)

SELECT
```

```
  employee_id,
  salary,
  CASE WHEN salary > double_average THEN 'Overpaid'
    WHEN salary < half_average THEN 'Underpaid'
  END AS outlier_status
FROM payout
WHERE salary > double_average
  OR salary < half_average;
```

Question 163:

```
create table purchases
(user_id int,product_id int,quantity int,purchase_date datetime);

insert into purchases values
(536,3223,6,'2022-01-11 12:33:44'),
(827,3585,35,'2022-02-20 14:05:26'),
(536,3223,5,'2022-03-02 09:33:28'),
(536,1435,10,'2022-03-02 08:40:00'),
(827,2452,45,'2022-04-09 00:00:00');

--Solution #1: Using Subquery

SELECT COUNT(DISTINCT users) AS repeated_purchasers
FROM (
  SELECT DISTINCT user_id AS users
  FROM purchases
  GROUP BY user_id, product_id
  HAVING COUNT(DISTINCT purchase_date) > 1
) AS repeat_purchases;

--Solution #2: Using CTE

WITH repeat_purchases AS (
SELECT DISTINCT user_id AS users
FROM purchases
GROUP BY user_id, product_id
HAVING COUNT(DISTINCT purchase_date) > 1
)
SELECT COUNT(DISTINCT users) AS repeated_purchasers
FROM repeat_purchases;

--Solution #3: Using Self-Join

SELECT COUNT(DISTINCT p1.user_id) AS repeated_purchasers
```

```
FROM purchases AS p1
INNER JOIN purchases AS p2
  ON p1.product_id = p2.product_id
    AND p1.purchase_date <> p2.purchase_date;
```

Question 164:

```
create table search_category
(country varchar(20),
search_cat varchar(20),
num_search int,
invalid_result_pct float);


insert into search_category values
('UK','home',null,null),
('UK','tax',98000,1.00),
('UK','travel',100000,3.25);


With invalid_results
AS(
select
    country,
    num_search,
    invalid_result_pct,
    CASE WHEN invalid_result_pct IS NOT NULL THEN num_search
    ELSE NULL END AS num_search_2,
    ROUND((num_search * invalid_result_pct)/100.0,0) AS invalid_search
FROM search_category
WHERE num_search IS NOT NULL AND invalid_result_pct iS NOT NULL
)
SELECT
    country,
    SUM(num_search_2) AS toatal_search,
    ROUND (SUM(invalid_search)/SUM(num_search_2) * 100.0,2) AS invalid_result_pct
    FROM invalid_results
    GROUP BY country ORDER BY country;
```

Question 165:

```sql
create table transactions
(
transaction_id integer,
type enum('deposit','withdrawal'),
amount float,
transaction_date timestamp
);

insert into transactions values
(19153,'deposit',65.90,'2022-07-10 10:00:00'),
(53151,'deposit',178.55,'2022-07-08 10:00:00'),
(29776,'withdrawal',25.90,'2022-07-08 10:00:00'),
(19153,'withdrawal',45.99,'2022-07-08 10:00:00'),
(77134,'deposit',32.60,'2022-07-10 10:00:00');
```

```sql
WITH daily_balances AS (
  SELECT transaction_date,
    EXTRACT(DAY FROM transaction_date) AS transaction_day,
    EXTRACT(MONTH FROM transaction_date) AS transaction_month,
   ROUND((SUM(CASE WHEN type = 'deposit' THEN amount
      WHEN type = 'withdrawal' THEN -amount END)),2) AS balance
  FROM transactions
  GROUP BY
  transaction_date,
    transaction_day,
    transaction_month)

SELECT
  (transaction_date),
  SUM(balance) OVER (
    PARTITION BY transaction_month
    ORDER BY transaction_day) AS balance
FROM daily_balances
ORDER BY transaction_day;
```

Question 166:

```sql
create table product_spend
(
category varchar(20),
product varchar(20),
```

```
user_id int,
spend float,
transaction_date timestamp
);

insert into product_spend VALUES
('appliance','refrigerator',165,246.00,'2021-12-26 12:00:00'),
('appliance','refrigerator',123,299.99,'2022-03-02 12:00:00'),
('appliance','washing machine',123,219.80,'2022-03-02 12:00:00'),
('electronics','vacuum',178,152.00,'2022-04-05 12:00:00'),
('electronics','wireless headset',156,249.90,'2022-08-07 12:00:00'),
('electronics','vacuum',145,189.00,'2022-07-15 12:00:00');
```

```
--Solution #1: Using CTE

WITH product_category_spend AS (
SELECT
  category,
  product,
  ROUND((SUM(spend)),2) AS total_spend
FROM product_spend
WHERE transaction_date >= '2022-01-01'
  AND transaction_date <= '2022-12-31'
GROUP BY category, product
),
top_spend AS (
SELECT *,
  RANK() OVER (
    PARTITION BY category
    ORDER BY total_spend DESC) AS ranking
FROM product_category_spend)

SELECT category, product, total_spend
FROM top_spend
WHERE ranking <= 2
ORDER BY category, ranking;

--Solution #2: Using Subquery

SELECT
  category,
  product,
  total_spend
```

```
FROM (
    SELECT  *,
      RANK() OVER (
        PARTITION BY category
        ORDER BY total_spend DESC) AS ranking
    FROM (
        SELECT
          category,
          product,
          ROUND((SUM(spend)),2) AS total_spend
        FROM product_spend
        WHERE transaction_date >= '2022-01-01'
          AND transaction_date <= '2022-12-31'
        GROUP BY category, product) AS total_spend
  ) AS top_spend
WHERE ranking <= 2
ORDER BY category, ranking;
```

Question 167:

```
create table users (user_id int,signup_date datetime, last_login datetime);

insert into users values
(1001,'2022-06-01 12:00:00','2022-07-05 12:00:00'),
(1002,'2022-06-03 12:00:00','2022-06-15 12:00:00'),
(1004,'2022-06-02 12:00:00','2022-06-15 12:00:00'),
(1006,'2022-06-15 12:00:00','2022-06-27 12:00:00'),
(1012,'2022-06-16 12:00:00','2022-07-27 12:00:00');
```

Question 168:

```
create table songs_history
(history_id int,user_id int,song_id int,song_plays int);

insert into songs_history VALUES
(10011,777,1238,11),(12452,695,4520,1);

create table songs_weekly
(user_id int,song_id int,listen_time datetime);

insert into songs_weekly values
(777,1238,'2022-08-01 12:00:00'),(695,4520,'2022-08-04 08:00:00'),
(125,9630,'2022-08-04 16:00:00'),(695,9852,'2022-08-07 12:00:00');
```

```sql
--Solution #1: Using CTE

WITH history AS (
SELECT user_id, song_id, song_plays
FROM songs_history
UNION ALL
SELECT user_id, song_id, COUNT(song_id) AS song_plays
FROM songs_weekly
WHERE listen_time <= '2022-04-08 23:59:59'
GROUP BY user_id, song_id
)
SELECT user_id, song_id, SUM(song_plays) AS song_count
FROM history
GROUP BY user_id, song_id
ORDER BY song_count DESC;

--Solution #2: Using Subquery

SELECT user_id, song_id, SUM(song_plays) AS song_count
FROM (
  SELECT user_id, song_id, song_plays
  FROM songs_history
  UNION ALL
  SELECT user_id, song_id, COUNT(song_id) AS song_plays
  FROM songs_weekly
  WHERE listen_time <= '2022-04-08 23:59:59'
  GROUP BY user_id, song_id
) AS report
GROUP BY user_id, song_id
ORDER BY song_count DESC;
```

Question 169:

```sql
create table emails(email_id int,user_id int,signup_date datetime);

insert into emails values
(125,7771,'2022-06-14 00:00:00'),
(236,6950,'2022-07-01 00:00:00'),
(433,1052,'2022-07-09 00:00:00');

CREATE table texts (text_id int, email_id int,signup_action varchar(20));

insert into texts values
(6878,125,'Confirmed'),(6920,236,'Not Confirmed'),
```

```
(6994,236,'Confirmed');
```

```
--Solution #1: Using CTE

WITH rate AS (
SELECT
  user_id,
  CASE WHEN texts.email_id IS NOT NULL THEN 1 ELSE 0 END AS signup
FROM emails
LEFT JOIN texts
  ON emails.email_id = texts.email_id
  AND signup_action = 'Confirmed')
  SELECT ROUND(SUM(signup) / COUNT(user_id), 2) AS confirm_rate
FROM rate;
--Solution #2: Using Subquery

SELECT
  ROUND(SUM(signup) / COUNT(user_id), 2) AS confirm_rate
FROM (
  SELECT
    user_id,
    CASE WHEN texts.email_id IS NOT NULL THEN 1
      ELSE 0 END AS signup
  FROM emails
  LEFT JOIN texts
    ON emails.email_id = texts.email_id
    AND signup_action = 'Confirmed'
) AS rate;
```

Question 170 :

```
create table tweets (tweet_id int,user_id int, tweet_date timestamp);

insert into tweets values
(214252,111,STR_TO_DATE('06/01/2022 12:00:00','%m/%d/%Y %H:%i:%s')),
(739252,111,STR_TO_DATE('06/01/2022 12:00:00','%m/%d/%Y %H:%i:%s')),
(846402,111,STR_TO_DATE('06/02/2022 12:00:00','%m/%d/%Y %H:%i:%s')),
(241425,254,STR_TO_DATE('06/02/2022 12:00:00','%m/%d/%Y %H:%i:%s')),
(137374,111,STR_TO_DATE('06/04/2022 12:00:00','%m/%d/%Y %H:%i:%s'));
```

```
--Solution #1: Using CTE

WITH tweet_count
```

```
AS (
  SELECT
    user_id, tweet_date,
    COUNT(DISTINCT tweet_id) AS tweet_num
  FROM tweets
  GROUP BY user_id,tweet_date
)
SELECT
  user_id, tweet_date,
  ROUND(
    AVG(tweet_num) OVER (
      PARTITION BY user_id
      ORDER BY user_id, tweet_date
      ROWS BETWEEN 2 PRECEDING AND CURRENT ROW), 2)
  AS rolling_avg_3d
FROM tweet_count;


--Solution #2: Using Subquery

SELECT
  user_id,
  tweet_date,
  ROUND(
    AVG(tweet_num) OVER (
      PARTITION BY user_id
      ORDER BY user_id, tweet_date
      ROWS BETWEEN 2 PRECEDING AND CURRENT ROW), 2)
  AS rolling_avg_3d
FROM (
  SELECT
    user_id,
    tweet_date,
    COUNT(DISTINCT tweet_id) AS tweet_num
  FROM tweets
  GROUP BY user_id, tweet_date) AS tweet_count;
```

Qusetion 171:

```
create table activities
(activity_id int,user_id int,
activity_type enum('send','open','chat'),
time_spent float,
activity_date datetime);
```

```
insert into activities values
(7274,123,'open',4.50,'2022-06-22 12:00:00'),
(2425,123,'send',3.50,'2022-06-22 12:00:00'),
(1413,456,'send',5.67,'2022-06-23 12:00:00'),
(1414,789,'chat',11.00,'2022-06-25 12:00:00'),
(2536,456,'open',3.00,'2022-06-25 12:00:00');

create table age_breakdown
(user_id int,
age_bucket enum('21-25','26-30','31-35'));

insert into age_breakdown values
(123,'31-35'),
(456,'26-30'),
(789,'21-25');
```

```
WITH snaps_statistics AS (
  SELECT
    age.age_bucket,
    SUM(CASE WHEN activities.activity_type = 'send'
      THEN activities.time_spent ELSE 0 END) AS send_timespent,
    SUM(CASE WHEN activities.activity_type = 'open'
      THEN activities.time_spent ELSE 0 END) AS open_timespent,
    SUM(activities.time_spent) AS total_timespent
  FROM activities
 INNER JOIN age_breakdown AS age
    ON activities.user_id = age.user_id
  WHERE activities.activity_type IN ('send', 'open')
  GROUP BY age.age_bucket)

SELECT
  age_bucket,
  ROUND(100.0 * send_timespent / total_timespent, 2) AS send_perc,
  ROUND(100.0 * open_timespent / total_timespent, 2) AS open_perc
FROM snaps_statistics;
```

Question 172:

```
create table personal_profiles (profile_id int,name varchar(30),followers int);

insert into personal_profiles VALUES(1,'Nick Singh',92000),
```

```
(2,'Zach Wilson',199000),(3,'Daliana Liu',171000),(4,'Ravit Jain',107000),
(5,'Vin Vashishta',139000),(6,'Susan Wojcicki',39000);

create table employee_company (personal_profile_id int,company_id int);

insert into employee_company VALUES (1,4),(1,9),(2,2),(3,1),(4,3),(5,6),(6,5);

create table company_pages(company_id int,name varchar(50),followers int);

insert into company_pages VALUES (1,'The Data Science Podcast',8000),
(2,'Airbnb',700000),
(3,'The Ravit Show',6000),
(4,'DataLemur',200),
(5,'Youtube',16000000),
(6,'DataScience.Vin',4500),
(9,'Ace The Data Science Interview',4479);
```

```
select
      profiles.profile_id
FROM
      personal_profiles AS profiles
JOIN
      employee_company as employee
ON
      profiles.profile_id = employee.personal_profile_id
JOIN
      company_pages AS pages
ON
      employee.company_id = pages.company_id
WHERE
       profiles.followers > pages.followers
GROUP BY
      profiles.profile_id
ORDER BY
      profiles.profile_id;
```