```
create database set2;

use set2;

51. Write an SQL query to report the name, population, and area of the big countries.
Return the result table in any order.

create table gdp
(
    name varchar(30),
    continent varchar(30),
    area bigint,
    population bigint,
    gdp bigint,
    PRIMARY KEY(name)
);

insert into gdp VALUES
('Afghanistan','Asia',652230,25500100,20343000000),
('Albania','Europe',28748,2831741,12960000000),
('Algeria','Africa',2381741,37100000,188681000000),
('Andorra','Europe',468,78115,3712000000),
('Angola','Africa',1246700,20609294,100990000000);


Ans. select gdp.name,gdp.population,gdp.area
     from gdp
     where gdp.area>=3000000 OR gdp.population>=25000000;


52. Write an SQL query to report the names of the customer that are not referred by the customer with id
= 2.

create table customer
(
    id int,
    name varchar(30),
    referee_id int,
    PRIMARY KEY (id)
);

insert into customer VALUES
(1,'Will',null),
(2,'Jane',null),
(3,'Alex',2),
(4,'Bill',null),
(5,'Zack',1),
(6,'Mark',2);


Ans. select name from customer
    where name not IN(select name from customer
    where referee_id=2);


53. Write an SQL query to report all customers who never order anything.

create table customers
(
    id int,
    name varchar(30),
    PRIMARY KEY (id)
);

create table orders
(
    id int,
    customerId int,
    PRIMARY KEY (id),
    Foreign Key (customerId) REFERENCES customers(id)
);

insert into customers VALUES
(1,'Joe'),
(2,'Henry'),
(3,'Sam'),
(4,'Max');

insert into orders VALUES
(1,3),
(2,1);
```

```
select name from customers
where id not IN(select customerId from orders);


54. Write an SQL query to find the team size of each of the employees.

create table employee
(
    employee_id int,
    team_id int,
    PRIMARY KEY (employee_id)
);

insert into employee VALUES
(1,8),
(2,8),
(3,8),
(4,7),
(5,9),
(6,9);


select employee_id,count(employee_id) over(partition by team_id)
from employee
ORDER BY employee_id;


55. Write an SQL query to find the countries where this company can invest.

create table person
(
    id int,
    name varchar(30),
    phone_number varchar(11),
    PRIMARY KEY (id)
);

create table country
(
    name varchar(30),
    country_code varchar(3),
    PRIMARY KEY (country_code)
);

create table calls
(
    caller_id int,
    callee_id int,
    duration int
);

insert into person VALUES
(3,'Jonathan','051-1234567'),
(12,'Elvis','051-7654321'),
(1,'Moncef','212-1234567'),
(2,'Maroua','212-6523651'),
(7,'Meir','972-1234567'),
(9,'Rachel','972-0011100');

insert into country VALUES
('Peru',51),
('Israel',972),
('Morocco',212),
('Germany',49),
('Ethiopia',251);

insert into calls VALUES
(1,9,33),
(2,9,4),
(1,2,59),
(3,12,102),
(3,12,330),
(12,3,5),
(7,9,13),
(7,1,3),
(9,7,1),
(1,7,7);

select * from (select p.name person,p.id id,c.name country from(select name,
case
    when LENGTH(country_code)=1 then concat('00',country_code)
    when LENGTH(country_code)=2 then concat('0',country_code)
    else country_code
end as country_code
from country) c
JOIN
person p
on substr(p.phone_number,1,3)=c.country_code) tb_in
JOIN
calls cl
on tb_in.id = cl.caller_id
GROUP BY tb_in.country;
```

57. Write an SQL query to find the customer_number for the customer who has placed the largest *number* of orders.

```
CREATE table orders
(
    order_number int,
    customer_number int,
    PRIMARY KEY (order_number)
);

insert into orders VALUES
(1,1),
(2,2),
(3,3),
(4,3);

select o1.customer_number from
(select customer_number,count(customer_number) tot_orders
from orders
group by customer_number
ORDER BY tot_orders DESC
LIMIT 1) o1;
```

58. Write an SQL query to report all the consecutive available seats in the cinema.

```
create table cinema
(
    seat_id int AUTO_INCREMENT,
    free bool,
    PRIMARY KEY (seat_id)
);

insert into cinema VALUES
(1,1),
(2,0),
(3,1),
(4,1),
(5,1);

select distinct tb1.seat_id from (select
case
    when c1.seat_id+1 = c2.seat_id then c1.seat_id
    when c1.seat_id-1 = c2.seat_id then c1.seat_id
end as seat_id
from cinema c1
join
cinema c2
on c1.seat_id != c2.seat_id
WHERE c1.free =1 and c2.free =1) tb1
where tb1.seat_id is not NULL
ORDER BY tb1.seat_id ASC;
```

59. Write an SQL query to report the names of all the salespersons who did not have any orders related to the company with the name "RED".

```
create table SalesPerson
(
    sales_id int,
    name varchar(30),
    salary int,
    commission_rate int,
    hire_date date,
    PRIMARY KEY (sales_id)
);

create table company
```

```
(
    com_id int,
    name varchar(30),
    city varchar(30),
    PRIMARY KEY (com_id)
);

create table orders
(
    order_id int,
    order_date date,
    com_id int,
    sales_id int,
    amount int,
    PRIMARY KEY (order_id),
    Foreign Key (com_id) REFERENCES company(com_id),
    Foreign Key (sales_id) REFERENCES SalesPerson(sales_id)
);

insert into SalesPerson VALUES
(1,'John',100000,6,'2006-01-04'),
(2,'Amy',12000,5,'2010-01-05'),
(3,'Mark',65000,12,'2008-12-25'),
(4,'Pam',25000,25,'2005-01-01'),
(5,'Alex',5000,10,'2007-03-02');

insert into company VALUES
(1,'RED','Boston'),
(2,'ORANGE','New York'),
(3,'YELLOW','Boston'),
(4,'GREEN','Austin');

insert into orders VALUES
(1,'2014-01-01',3,4,10000),
(2,'2014-01-02',4,5,5000),
(3,'2014-01-03',1,1,50000),
(4,'2014-01-04',1,4,25000);

select name from SalesPerson
where name not in (select s.name from
(select c.com_id cid,c.name cname,o.sales_id sid from company c
JOIN
orders o
on c.com_id = o.com_id
WHERE c.name='RED') tb1
JOIN
SalesPerson s
on tb1.sid = s.sales_id);



60. Write an SQL query to report for every three line segments whether they can form a triangle.

CREATE table triangle
(
    x int,
    y int,
    z int,
    PRIMARY KEY (x, y, z)
);


insert into triangle VALUES
(13,15,30),
(10,20,15);

--Using the Triangle Inequality theorem we can easily check if 3 given sides forms a triangle, where it states
--if we add two sides of a triangle then it is always greater than the 3rd side
select x,y,z,
CASE
    when x+y>z and y+z>x and x+z>y then 'Yes'
    else 'No'
end as triangle
from triangle;


61. Write an SQL query to report the shortest distance between any two points from the Point table.

create table POINT
(
    x INT,
    PRIMARY KEY (x)
);

insert into POINT VALUES
(-1),
(0),
(2);

select * from (select abs(abs(p1.x)-abs(p2.x)) as distance
from POINT p1
JOIN
POINT p2
ORDER BY distance) tb1
WHERE tb1.distance != 0
limit 1;



62. Write a SQL query for a report that provides the pairs (actor_id, director_id) where the actor has
cooperated with the director at least three times.

create table ActorDirector
(
    actor_id int,
    director_id int,
    timestamp int,
    PRIMARY KEY (timestamp)
);

insert into ActorDirector VALUES
(1,1,0),
(1,1,1),
(1,1,2),
(1,2,3),
(1,2,4),
(2,1,5),
(2,1,6);


select tb1.actor_id,tb1.director_id
from(select actor_id,director_id,
count(timestamp) cont
from ActorDirector
group by actor_id,director_id
having cont>=3
order by cont desc) tb1;



63. Write an SQL query that reports the product_name, year, and price for each sale_id in the Sales table.

create table product
(
    product_id int,
    product_name varchar(30),
    PRIMARY KEY (product_id)
);

create table sales
(
    sale_id int,
    product_id int,
    year int,
    quantity int,
    price int,
    primary key (sale_id, year),
    Foreign Key(product_id) REFERENCES product(product_id)
);

insert into product VALUES
(100,'Nokia'),
(200,'Apple'),
(300,'Samsung');

insert into sales VALUES
(1,100,2008,10,5000),
(2,100,2009,12,5000),
(7,200,2011,15,9000);

select p.product_name, s.year, s.price from sales s
JOIN
product p
on s.product_id = p.product_id;
```

```
64. Write an SQL query that reports the average experience years of all the employees for each project,

create table employee
(
    employee_id int,
    name varchar(30),
    experience_years int,
    PRIMARY key (employee_id)
);

create table project
(
    project_id int,
    employee_id int,
    PRIMARY key (project_id, employee_id),
    Foreign Key(employee_id) references employee(employee_id)
);


insert into employee VALUES
(1,'Khaled',3),
(2,'Ali',2),
(3,'John',1),
(4,'Doe',2);


insert into project values
(1,1),
(1,2),
(1,3),
(2,1),
(2,4);

select p.project_id project_id,round(avg(e.experience_years),2) average_years
from employee e
JOIN
project p
on e.employee_id = p.employee_id
GROUP BY p.project_id;




65. Write an SQL query that reports the best seller by total sales price, If there is a tie, report them all.

create table product
(
    product_id int,
    product_name varchar(30),
    unit_price int,
    PRIMARY key (product_id)
);

create table sales
(
    seller_id int,
    product_id int,
    buyer_id int,
    sale_date date,
    quantity int,
    price int,
    Foreign Key(product_id) references product(product_id)
);

insert into product VALUES
(1,'S8',1000),
(2,'G4',800),
(3,'iPhone',1400);

insert into sales VALUES
(1,1,1,'2019-01-21',2,2000),
(1,2,2,'2019-02-17',1,800),
(2,2,3,'2019-06-02',1,800),
(3,3,4,'2019-05-13',2,2800);


select seller_id
from(select seller_id,
dense_rank() OVER (order by sum(price) desc) as rnk
```

```
from sales
GROUP BY seller_id) tb1
where tb1.rnk=1;




66. Write an SQL query that reports the buyers who have bought S8 but not iPhone.

create table product
(
    product_id int,
    product_name varchar(30),
    unit_price int,
    PRIMARY KEY (product_id)
);

create table sales
(
    seller_id int,
    product_id int,
    buyer_id int,
    sale_date date,
    quantity int,
    price int,
    Foreign Key (product_id) REFERENCES product(product_id)
);

insert into product VALUES
(1,'S8',1000),
(2,'G4',800),
(3,'iPhone',1400);

insert into sales VALUES
(1,1,1,'2019-01-21',2,2000),
(1,2,2,'2019-02-17',1,800),
(2,1,3,'2019-06-02',1,800),
(3,3,3,'2019-05-13',2,2800);

select distinct buyer_id
from sales s
join
product p
on p.product_id = s.product_id
where p.product_name = 'S8'
and buyer_id not in (select buyer_id
from sales s
join
product p
on p.product_id = s.product_id
where p.product_name = 'iPhone');




67. Write an SQL query to compute the moving average of how much the customer paid in a seven days
window (i.e., current day + 6 days before). average_amount should be rounded to two decimal places.

create table customer
(
    customer_id int,
    name varchar(30),
    visited_on date,
    amount int,
    PRIMARY KEY (customer_id, visited_on)
);

insert into customer VALUES
(1,'Jhon','2019-01-01',100),
(2,'Daniel','2019-01-02',110),
(3,'Jade','2019-01-03',120),
(4,'Khaled','2019-01-04',130),
(5,'Winston','2019-01-05',110),
(6,'Elvis','2019-01-06',140),
(7,'Anna','2019-01-07',150),
(8,'Maria','2019-01-08',80),
(9,'Jaze','2019-01-09',110),
(1,'Jhon','2019-01-10',130),
(3,'Jade','2019-01-10',150);

with sum_result as(SELECT
    visited_on,
    SUM(amount) as amount
```

```
FROM customer
GROUP BY visited_on)

select tb1.visited_on,tb1.amount,tb1.average_amount
from (select visited_on,
sum(amount) over(ORDER BY visited_on rows BETWEEN 6 preceding and current row)
as amount,
ROUND(AVG(amount) OVER(ORDER BY visited_on ROWS BETWEEN 6 PRECEDING AND CURRENT ROW),2)
as average_amount
from sum_result) tb1
where tb1.visited_on > '2019-01-06';




68. Write an SQL query to find the total score for each gender on each day.

create table scores
(
    player_name varchar(30),
    gender varchar(1),
    day date,
    score_points int,
    PRIMARY KEY (gender, day)
);

insert into scores VALUES
('Aron','F','2020-01-01',17),
('Alice','F','2020-01-07',23),
('Bajrang','M','2020-01-07',7),
('Khali','M','2019-12-25',11),
('Slaman','M','2019-12-30',13),
('Joe','M','2019-12-31',3),
('Jose','M','2019-12-18',2),
('Priya','F','2019-12-31',23),
('Priyanka','F','2019-12-30',17);

select gender,day,
sum(score_points) over(partition by gender ORDER BY day)
from scores;




69. Write an SQL query to find the start and end number of continuous ranges in the table Logs.

CREATE table logs
(
    log_id int,
    PRIMARY KEY (log_id)
);

insert into logs VALUES
(1),
(2),
(3),
(7),
(8),
(10);

select min(tb_out.log_id) start_id,max(tb_out.log_id) end_id
from(select tb_in.log_id log_id,(tb_in.log_id-tb_in.rnk) as diff
from(select log_id,
dense_rank() over(ORDER BY log_id) rnk
from logs) tb_in)tb_out
GROUP BY tb_out.diff;




70. Write an SQL query to find the number of times each student attended each exam.

create table students
(
    student_id int,
    student_name varchar(30),
    PRIMARY KEY (student_id)
);

create table subjects
(
    subject_name varchar(30),
    PRIMARY KEY (subject_name)
);

create table examinations
(
    student_id int,
    subject_name varchar(30)
);

insert into students VALUES
(1,'Alice'),
(2,'Bob'),
(13,'John'),
(6,'Alex');

insert into subjects VALUES
('Math'),
('Physics'),
('Programming');

insert into examinations VALUES
(1,'Math'),
(1,'Physics'),
(1,'Programming'),
(2,'Programming'),
(1,'Physics'),
(1,'Math'),
(13,'Math'),
(13,'Programming'),
(13,'Physics'),
(2,'Math'),
(1,'Math');

select s.student_id,s.student_name,sub.subject_name,count(e.subject_name) as attended_exams
from students s
join 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,s.student_name,sub.subject_name;
```

71. Write an SQL query to find employee_id of all employees that directly or indirectly report their work to the head of the company.

```
create table employees
(
    employee_id int,
    employee_name varchar(30),
    manager_id int,
    PRIMARY KEY (employee_id)
);

insert into employees VALUES
(1,'Boss',1),
(3,'Alice',3),
(2,'Bob',1),
(4,'Daniel',2),
(7,'Luis',4),
(8,'Jhon',3),
(9,'Angela',8),
(77,'Robert',1);

with recursive emp_hir as
(
   select employee_id from employees where employee_name='Boss'
   UNION
   select em.employee_id
   from emp_hir eh inner join employees em on eh.employee_id = em.manager_id
)

select * from emp_hir where employee_id != 1;
```

72. Write an SQL query to find for each month and country, the number of transactions and their total amount, the number of approved transactions and their total amount.

```
CREATE table transactions
```

```
(
    id int,
    country varchar(30),
    state enum("approved", "declined"),
    amount int,
    trans_date date,
    PRIMARY KEY (id)
);

insert into transactions VALUES
(121,'US','approved',1000,'2018-12-18'),
(122,'US','declined',2000,'2018-12-19'),
(123,'US','approved',2000,'2019-01-01'),
(124,'DE','approved',2000,'2019-01-07');

select DATE_FORMAT(trans_date, '%Y-%m') month,country,count(id)
from transactions
GROUP BY month,country;




73. Write an SQL query to find the average daily percentage of posts that got removed after being
reported as spam, rounded to 2 decimal places.

CREATE DATABASE set2;
use set2;

create table actions
(
    user_id int,
    post_id int,
    action_date date,
    action enum ('view', 'like', 'reaction', 'comment', 'report', 'share'),
    extra varchar(10)
);

create table removals
(
    post_id int,
    remove_date date,
    PRIMARY KEY (post_id)
);

insert into actions VALUES
(1,1,'2019-07-01','view','null'),
(1,1,'2019-07-01','like','null'),
(1,1,'2019-07-01','share','null'),
(2,2,'2019-07-04','view','null'),
(2,2,'2019-07-04','report','spam'),
(3,4,'2019-07-04','view','null'),
(3,4,'2019-07-04','report','spam'),
(4,3,'2019-07-02','view','null'),
(4,3,'2019-07-02','report','spam')
(5,2,'2019-07-03','view','null'),
(5,2,'2019-07-03','report','racism'),
(5,5,'2019-07-03','view','null'),
(5,5,'2019-07-03','report','racism');

insert into removals VALUES
(2,'2019-07-20'),
(3,'2019-07-18');

select round(avg(daily_count), 2) as average_daily_percent
from
(select count(r.post_id)/count(a.post_id)*100 as daily_count
from actions a
left join removals r
on a.post_id = r.post_id
where extra = 'spam'
group by action_date
) tb1;



76. Write an SQL query to find the salaries of the employees after applying taxes. Round the salary to the
nearest integer.

CREATE TABLE salaries
(
    company_id int,
    employee_id int,
    employee_name varchar(30),
    salary int,
    PRIMARY KEY (company_id, employee_id)
);

insert into salaries VALUES
(1,1,'Tony',2000),
(1,2,'Pronub',21300),
(1,3,'Tyrrox',10800),
(2,1,'Pam',300),
(2,7,'Bassem',450),
(2,9,'Hermione',700),
(3,7,'Bocaben',100),
(3,2,'Ognjen',2200),
(3,13,'Nyan Cat',3300),
(3,15,'Morning Cat',7777);

SELECT s.company_id,s.employee_id,s.employee_name,
case
    when max_sal<1000 then round(s.salary)
    when max_sal BETWEEN 1000 and 10000 then round(s.salary - (0.24*s.salary))
    when max_sal>10000 then s.salary - round((0.49*s.salary))
end as salary
from(select company_id,max(salary) max_sal
from salaries
GROUP BY company_id) tb1
JOIN
salaries s
on s.company_id = tb1.company_id;




77. Write an SQL query to report the difference between the number of apples and oranges sold each day.

CREATE table sales
(
    sale_date date,
    fruit enum('apples','oranges'),
    sold_num int,
    PRIMARY KEY (sale_date, fruit)
);

insert into sales VALUES
('2020-05-01','apples',10),
('2020-05-01','oranges',8),
('2020-05-02','apples',15),
('2020-05-02','oranges',15),
('2020-05-03','apples',20),
('2020-05-03','oranges',0),
('2020-05-04','apples',15),
('2020-05-04','oranges',16);

select tb_a.sale_date sale_date, (tb_a.sold_num-tb_o.sold_num) diff
from(select sale_date,sold_num from sales
WHERE fruit = 'apples'
GROUP BY sale_date)tb_a
JOIN
(select sale_date,sold_num from sales
WHERE fruit = 'oranges'
GROUP BY sale_date) tb_o
ON tb_a.sale_date=tb_o.sale_date;




78. Write an SQL query to evaluate the boolean expressions in Expressions table.

create table variables
(
    name varchar(2),
    value int,
    PRIMARY KEY (name)
);

create table expressions
(
    left_operand varchar(1),
    operator enum('<', '>', '='),
    right_operand varchar(1),
    PRIMARY KEY (left_operand, operator, right_operand)
);
```

```
insert into variables VALUES
('x',66),
('y',77);

insert into expressions VALUES
('x','>','y'),
('x','<','y'),
('x','=','y'),
('y','>','x'),
('y','<','x'),
('x','=','x');

select e.left_operand, e.operator, e.right_operand,
    case
        when e.operator = '<' then if(l.value < r.value,'true','false')
        when e.operator = '>' then if(l.value > r.value,'true','false')
        else if(l.value = r.value,'true','false')
    end as value
from expressions e
left join variables l on e.left_operand = l.name
left join variables r on e.right_operand = r.name;



81. Query the Name of any student in STUDENTS who scored higher than 75 Marks

create table students
(
    ID int,
    Name VARCHAR(30),
    Marks int
);

insert into students VALUES
(1,'Ashley',81),
(2,'Samantha',75),
(4,'Julia',76),
(3,'Belvet',84);

SELECT name from students
where marks>75
order by substr(name,-3),id asc;



82. Write a query that prints a list of employee names (i.e.: the name attribute) from the Employee table in
alphabetical order.

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 name from employee
order by name ASC;




83. Write a query that prints a list of employee names (i.e.: the name attribute) for employees in
Employee having a salary greater than $2000 per month who have been employees for less than 10
months.

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 name from employee
where salary>2000 and months<10
order by employee_id ASC;



84. Write a query identifying the type of each record in the TRIANGLES table using its three side lengths.
create table triangles
(
    a int,
    b INT,
    c INT
);

insert into triangles VALUES
(20,20,23),
(20,20,20),
(20,21,22),
(13,14,30);

SELECT
case
when a+b>c and b+c>a and a+c>b
    then
        case
        when a=b and b=c then 'Equilateral'
        when a=b or b=c or a=c then 'Isosceles'
        else 'Scalene'
        end
else 'Not a Triangle'
end as output
from triangles;




85. Write a query to obtain the year-on-year growth rate for the total spend of each product for
each year.

create table user_transactions
(
    transaction_id integer,
    product_id integer,
    spend decimal,
    transaction_date datetime
);

insert into user_transactions VALUES
(1341,123424,1500.60,'2019-12-31 12:00:00'),
(1423,123424,1000.20,'2020-12-31 12:00:00'),
(1623,123424,1246.44,'2021-12-31 12:00:00'),
(1322,123424,2145.32,'2022-12-31 12:00:00');


with temp as
(select year(transaction_date) year,product_id,spend
from user_transactions)

select t1.year,t1.product_id,t1.spend curr_year_spend,t2.spend prev_year_spend,
CASE
    when t2.spend is NULL then NULL
    else round(((t1.spend-t2.spend)/t2.spend*100),2)
end as yoy_rate
from temp t1
left JOIN
```

```
temp t2
on t1.year-1 = t2.year and t1.product_id = t2.product_id;



86. Write a SQL query to find the number of prime and non-prime items that can be stored in the 500,000
square feet warehouse.


87. Write a query to obtain the active user retention in July 2022.

create table user_actions
(
    user_id int,
    event_id int,
    event_type ENUM ("sign-in", "like", "comment"),
    event_date datetime
);

insert into user_actions VALUES
(445,7765,'sign-in','2022-05-31 12:00:00'),
(742,6458,'sign-in','2022-06-03 12:00:00'),
(445,3634,'like','2022-06-05 12:00:00'),
(742,1374,'comment','2022-06-05 12:00:00'),
(648,3124,'like','2022-06-18 12:00:00');

select max(month(event_date)) month,count(distinct user_id) monthly_active_users
from user_actions
group by user_id
having count(distinct month(event_date)) >= 2;




88. Write a query to report the median of searches made by a user. Round the median to one decimal
point.

create table search_frequency
(
    searches int,
    num_users integer
);

insert into search_frequency VALUES
(1,2),
(2,2),
(3,3),
(4,1);

WITH recursive decompose_count AS
  (
    SELECT 1 AS n
    UNION ALL
    SELECT n+1 AS n FROM decompose_count
    WHERE n < (SELECT MAX(frequency) AS max_frequency FROM search_frequency)
  ),
  table_expand as(
    SELECT b.num ,
           ROW_NUMBER() OVER(ORDER BY b.num) AS row_seq
    FROM decompose_count AS a
    JOIN search_frequency AS b
    ON a.n <= b.frequency
  )

select round(sum(num)/2,1) as median
from table_expand te1
JOIN
(select
    case
        when count(num)%2=1 then (count(num)/2)+1
        else count(num)/2
    end as lower_bound,
    CASE
        when count(num)%2=1 then (count(num)/2)+1
        else (count(num)/2)+1
    end as upper_bound
from table_expand) te2
on te1.row_seq = te2.lower_bound or te1.row_seq = te2.upper_bound;



89. Write a query to update the Facebook advertiser's status using the daily_pay table.
```

```
90. Write a query that calculates the total time that the fleet of servers was running. The output should be
in units of full days.



91. Using the transactions table, identify any payments made at the same merchant with the same credit
card for the same amount within 10 minutes of each other. Count such repeated payments.



92. Write a query to find the bad experience rate in the first 14 days for new users who signed up in June
2022.



95. Write an SQL query to report the median of all the numbers in the database after decompressing the
Numbers table.

create table numbers
(
    num int,
    frequency int,
    PRIMARY KEY (num)
);

insert into numbers VALUES
(0,7),
(1,1),
(2,3),
(3,1);

WITH recursive decompose_count AS
  (
    SELECT 1 AS n
    UNION ALL
    SELECT n+1 AS n FROM decompose_count
    WHERE n < (SELECT MAX(frequency) AS max_frequency FROM numbers)
  ),
  table_expand as(
    SELECT b.num ,
           ROW_NUMBER() OVER(ORDER BY b.num) AS row_seq
    FROM decompose_count AS a
    JOIN numbers AS b
    ON a.n <= b.frequency
  )

select round(sum(num)/2,1) as median
from table_expand te1
JOIN
(select
    case
        when count(num)%2=1 then (count(num)/2)+1
        else count(num)/2
    end as lower_bound,
    CASE
        when count(num)%2=1 then (count(num)/2)+1
        else (count(num)/2)+1
    end as upper_bound
from table_expand) te2
on te1.row_seq = te2.lower_bound or te1.row_seq = te2.upper_bound;


96. Write an SQL query to report the comparison result (higher/lower/same) of the average salary of
employees in a department to the company's average salary.

create table salary
(
    id int,
    employee_id int,
    amount int,
    pay_date date,
    PRIMARY KEY (id)
);

create table employee
(
    employee_id int,
    department_id int,
    PRIMARY KEY (employee_id)
);

insert into salary VALUES
(1,1,9000,'2017-03-31'),
(2,2,6000,'2017-03-31'),
(3,3,10000,'2017-03-31'),
(4,1,7000,'2017-02-28'),
(5,2,6000,'2017-02-28'),
(6,3,8000,'2017-02-28');

insert into employee VALUES
(1,1),
(2,2),
(3,2);

with company_avg as(
    select CONCAT(year(pay_date),"-",month(pay_date)) as pay_month,
    avg(amount) as com_avg
    from salary
    GROUP BY pay_month
),avg_tb as (
    select *
    from(select e.department_id,month(s.pay_date) pay_month_s,
        sum(amount) amount,count(e.employee_id) count_emp
    from employee e
    JOIN
    salary s
    on e.employee_id = s.employee_id
    GROUP BY month(s.pay_date),e.department_id) tb1
    JOIN
    company_avg ca
    on tb1.pay_month_s = substr(ca.pay_month,-1)
)

select department_id,pay_month,
case
    when amount/count_emp > com_avg then 'higher'
    when amount/count_emp < com_avg then 'lower'
    else 'same'
end as comparison
from avg_tb;



97. Write an SQL query to report for each install date, the number of players that installed the game on
that day, and the day one retention.

create table activity
(
    player_id int,
    device_id int,
    event_date date,
    games_played int,
    PRIMARY KEY (player_id, event_date)
);

insert into activity VALUES
(1,2,'2016-03-01',5),
(1,2,'2016-03-02',6),
(2,3,'2017-06-25',1),
(3,1,'2016-03-01',0),
(3,4,'2016-07-03',5);

select tb1.install_dt,tb1.installs,
case
    when tb2.installs is NULL then 0
    else round(tb2.installs/tb1.installs,2)
end as Day1_retention
from(select count(player_id) installs,min(event_date) as install_dt
from activity
GROUP BY event_date) tb1
left JOIN
(select count(player_id) installs,min(event_date) as install_dt
from activity
GROUP BY event_date) tb2
on tb1.install_dt + 1 = tb2.install_dt
where tb1.install_dt in (select min(event_date) as install_dt
from activity
GROUP BY player_id);


99. Write an SQL query to report the students (student_id, student_name) being quiet in all exams. Do not
```

```
return the student who has never taken any exam.

create table student
(
    student_id int,
    student_name varchar(30),
    PRIMARY KEY (student_id)
);

CREATE table exam
(
    exam_id int,
    student_id int,
    score int,
    PRIMARY KEY (exam_id, student_id)
);

insert into exam VALUES
(10,1,70),
(10,2,80),
(10,3,90),
(20,1,80),
(30,1,70),
(30,3,80),
(30,4,90),
(40,1,60),
(40,2,70),
(40,4,80);

insert into student VALUES
(1,'Daniel'),
(2,'Jade'),
(3,'Stella'),
(4,'Jonathan'),
(5,'Will');

select
   s.*
from exam e
inner join
student s
on s.student_id=e.student_id
group by student_id
having max(score) not in (select max(score) from exam)
   and min(score) not in (select min(score) from exam);
```