Crack SQL Interview in 50 Questions & Answer

Mostafa Ahmed

❖ Select Questions

Q: 01

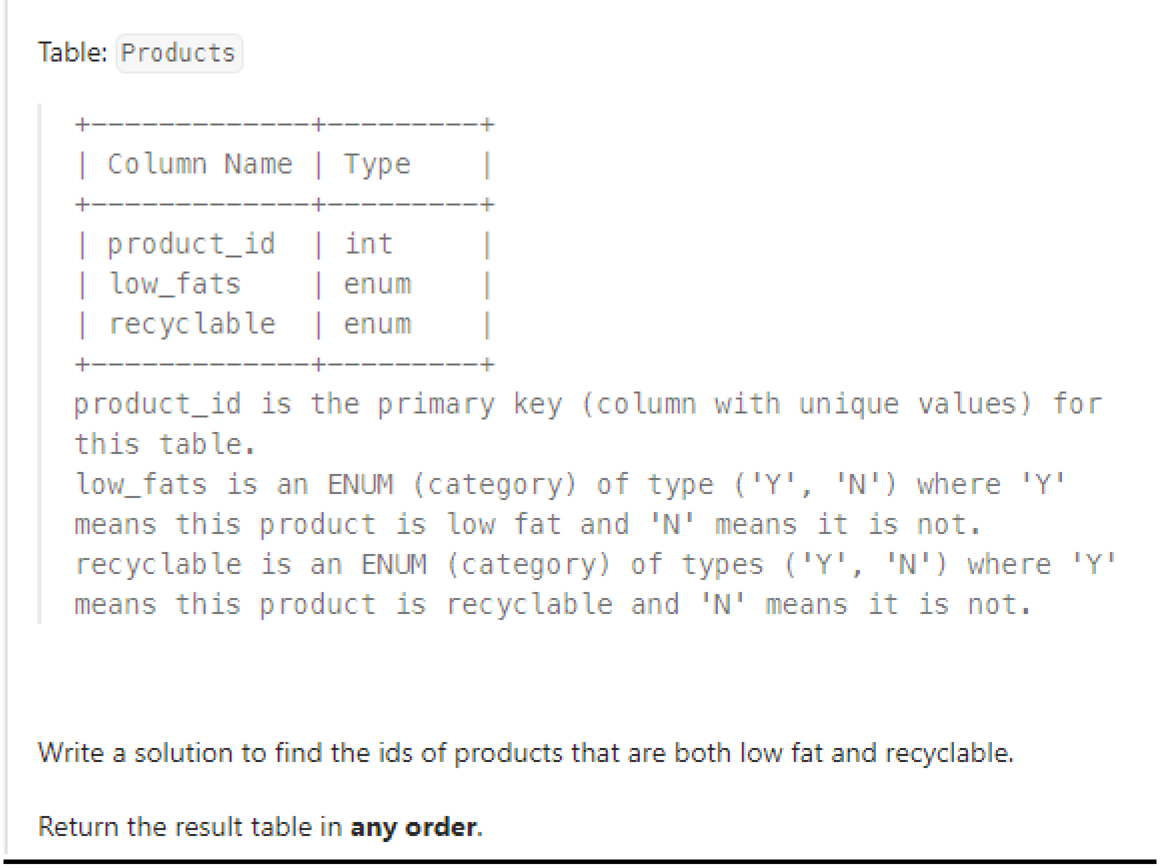


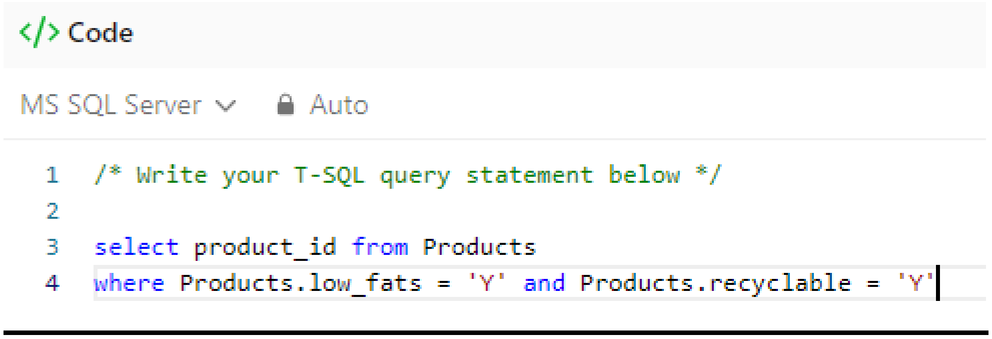
Table: `Products`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| product_id  | int     |
| low_fats    | enum    |
| recyclable  | enum    |
+-------------+---------+
product_id is the primary key (column with unique values) for
this table.
low_fats is an ENUM (category) of type ('Y', 'N') where 'Y'
means this product is low fat and 'N' means it is not.
recyclable is an ENUM (category) of types ('Y', 'N') where 'Y'
means this product is recyclable and 'N' means it is not.
```

Write a solution to find the ids of products that are both low fat and recyclable.

Return the result table in **any order**.

Answer



```sql
/* Write your T-SQL query statement below */

select product_id from Products
where Products.low_fats = 'Y' and Products.recyclable = 'Y'
```

Q: 02

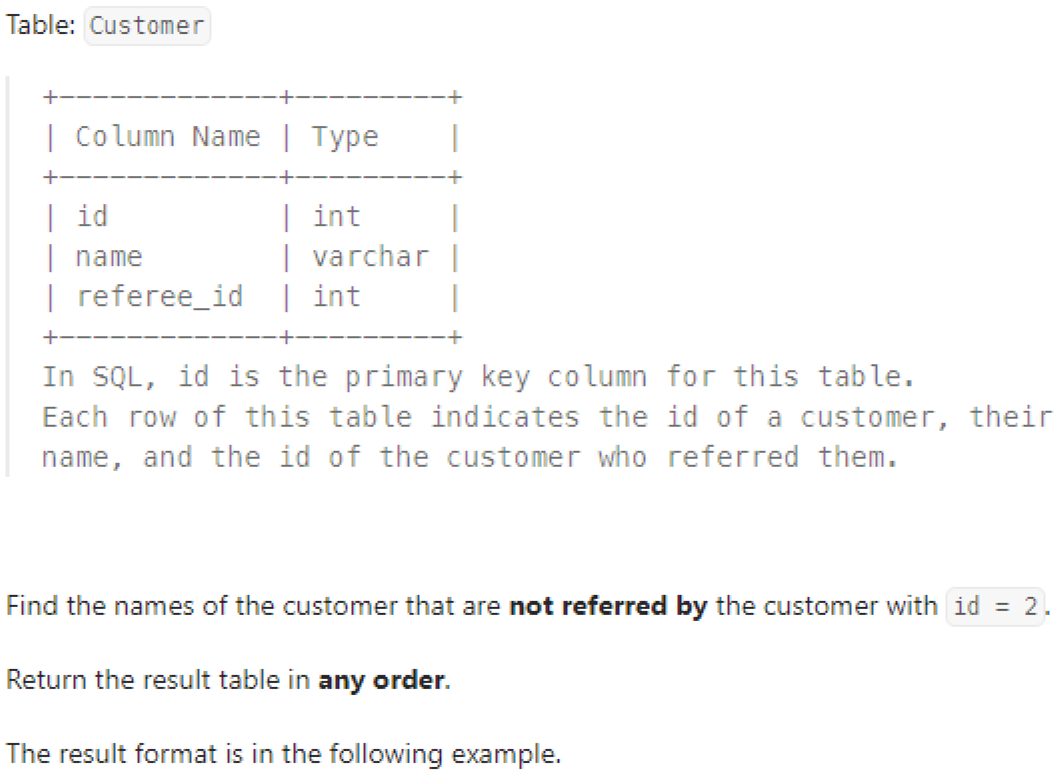


Table: `Customer`

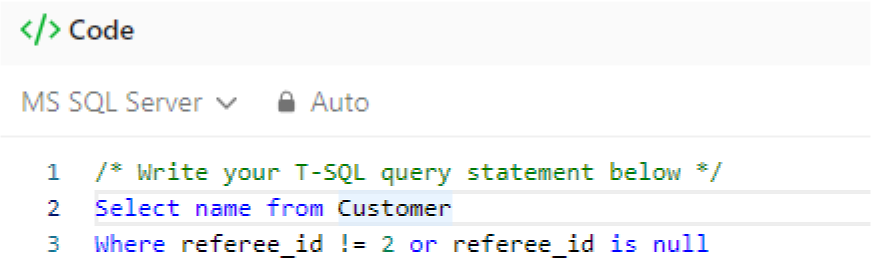
```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| id          | int     |
| name        | varchar |
| referee_id  | int     |
+-------------+---------+
In SQL, id is the primary key column for this table.
Each row of this table indicates the id of a customer, their
name, and the id of the customer who referred them.
```

Find the names of the customer that are **not referred by** the customer with `id = 2`.

Return the result table in **any order**.

The result format is in the following example.

Answer



```
/* Write your T-SQL query statement below */
Select name from Customer
Where referee_id != 2 or referee_id is null
```

Q: 03

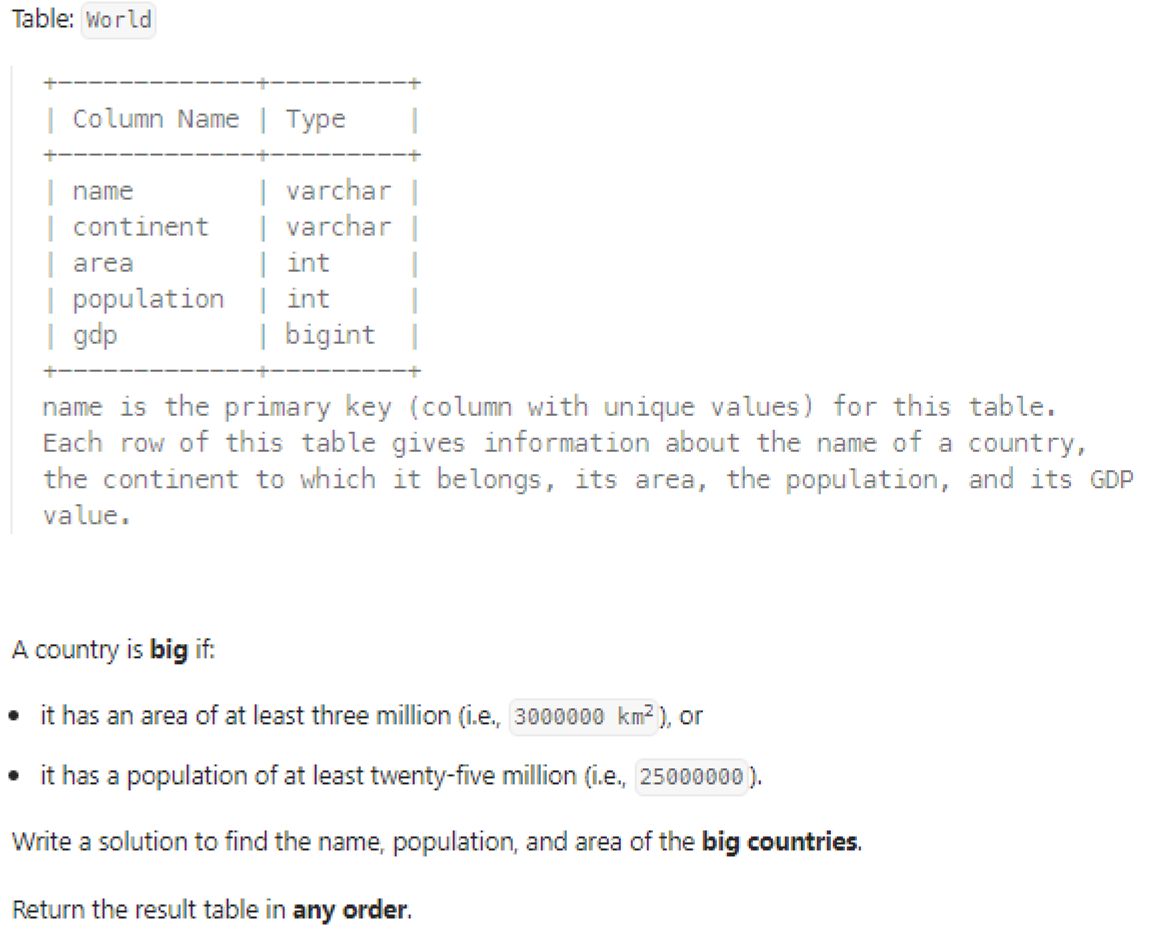


Table: `World`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| name        | varchar |
| continent   | varchar |
| area        | int     |
| population  | int     |
| gdp         | bigint  |
+-------------+---------+
name is the primary key (column with unique values) for this table.
Each row of this table gives information about the name of a country,
the continent to which it belongs, its area, the population, and its GDP
value.
```

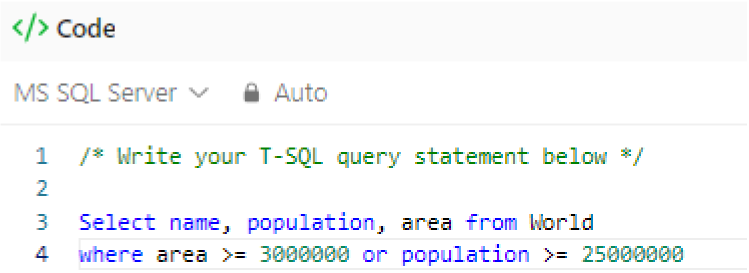
A country is **big** if:

- it has an area of at least three million (i.e., `3000000 km²`), or
- it has a population of at least twenty-five million (i.e., `25000000`).

Write a solution to find the name, population, and area of the **big countries**.

Return the result table in **any order**.

Answer



```
/* Write your T-SQL query statement below */

Select name, population, area from World
where area >= 3000000 or population >= 25000000
```

Q: 04

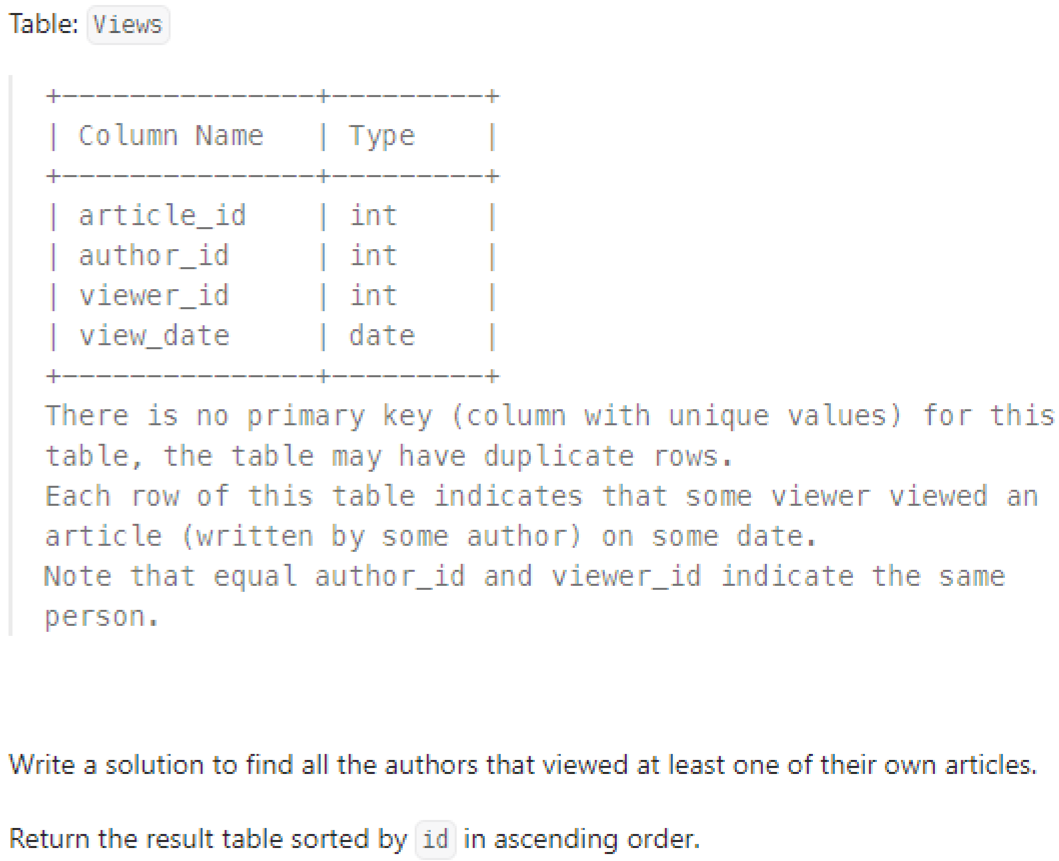


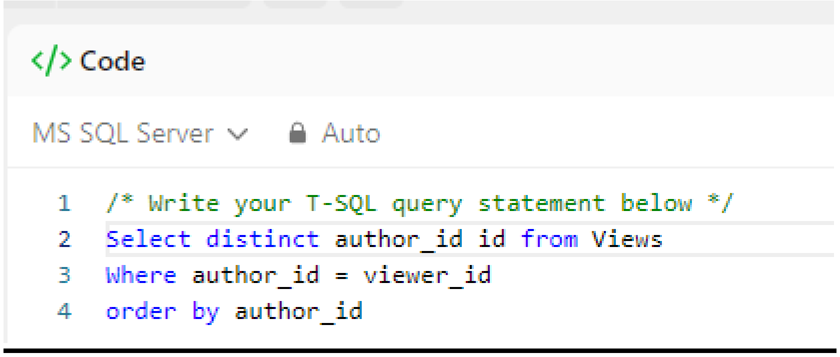
Table: Views

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| article_id    | int     |
| author_id     | int     |
| viewer_id     | int     |
| view_date     | date    |
+---------------+---------+
There is no primary key (column with unique values) for this
table, the table may have duplicate rows.
Each row of this table indicates that some viewer viewed an
article (written by some author) on some date.
Note that equal author_id and viewer_id indicate the same
person.
```

Write a solution to find all the authors that viewed at least one of their own articles.

Return the result table sorted by `id` in ascending order.

Answer



```sql
/* Write your T-SQL query statement below */
Select distinct author_id id from Views
Where author_id = viewer_id
order by author_id
```

Q: 05

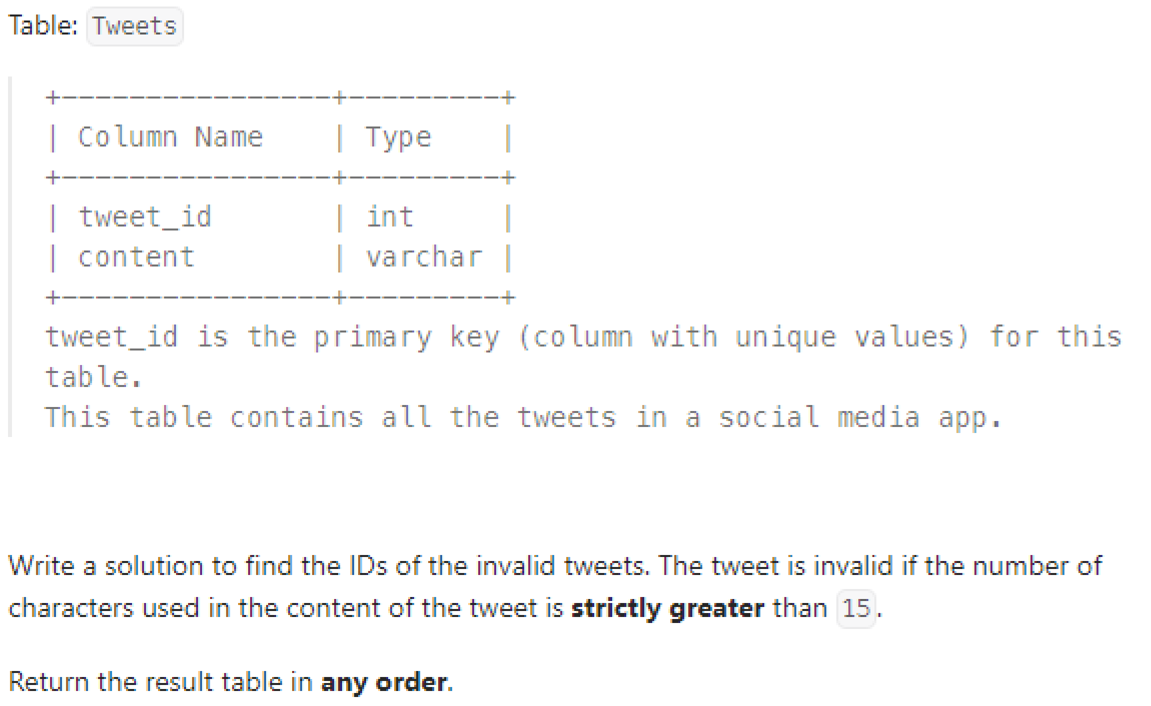


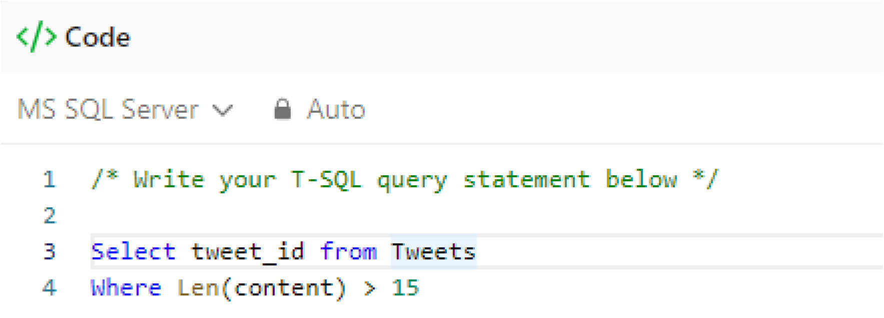
Table: `Tweets`

```
+----------------+---------+
| Column Name    | Type    |
+----------------+---------+
| tweet_id       | int     |
| content        | varchar |
+----------------+---------+
tweet_id is the primary key (column with unique values) for this table.
This table contains all the tweets in a social media app.
```

Write a solution to find the IDs of the invalid tweets. The tweet is invalid if the number of characters used in the content of the tweet is **strictly greater** than `15`.

Return the result table in **any order**.

Answer



```sql
/* Write your T-SQL query statement below */

Select tweet_id from Tweets
Where Len(content) > 15
```

❖ Basic Join

Q: 06

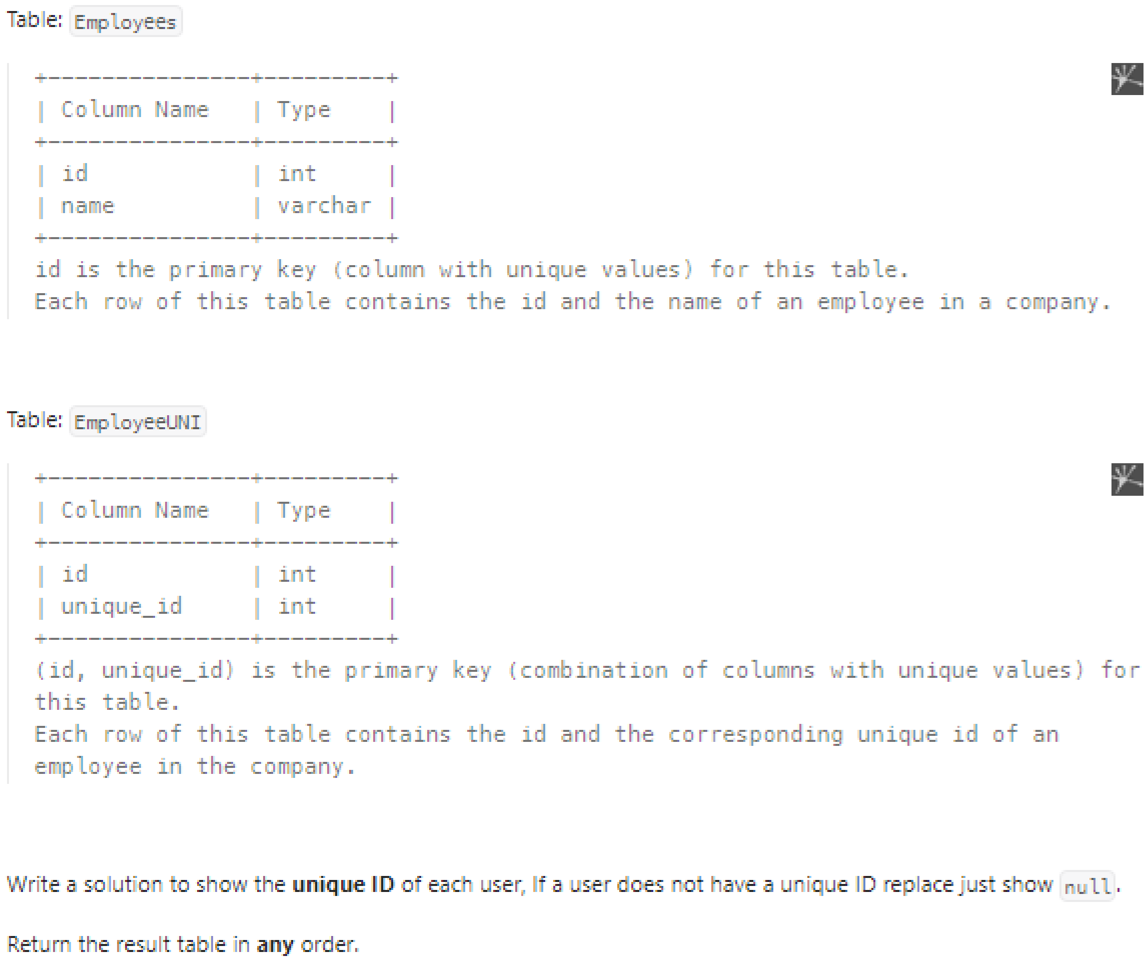


Table: `Employees`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| id            | int     |
| name          | varchar |
+---------------+---------+
id is the primary key (column with unique values) for this table.
Each row of this table contains the id and the name of an employee in a company.
```

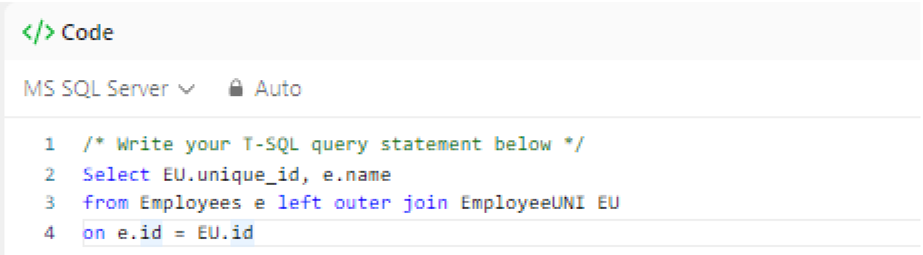
Table: `EmployeeUNI`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| id            | int     |
| unique_id     | int     |
+---------------+---------+
(id, unique_id) is the primary key (combination of columns with unique values) for
this table.
Each row of this table contains the id and the corresponding unique id of an
employee in the company.
```

Write a solution to show the **unique ID** of each user, If a user does not have a unique ID replace just show `null`.

Return the result table in **any** order.

Answer



</> Code

MS SQL Server ⌄ 🔒 Auto

```sql
/* Write your T-SQL query statement below */
Select EU.unique_id, e.name
from Employees e left outer join EmployeeUNI EU
on e.id = EU.id
```

Q: 07

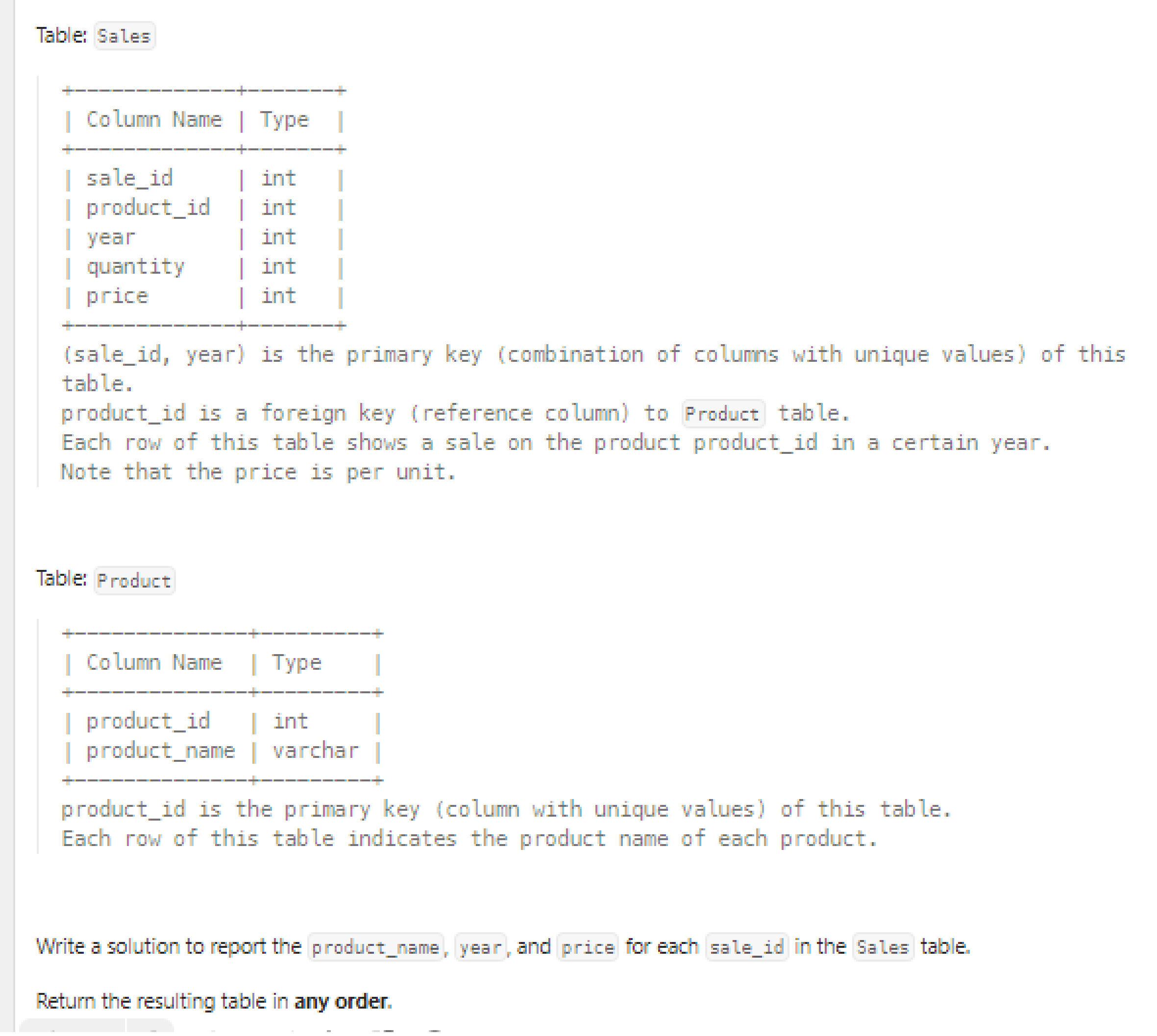


Table: `Sales`

```
+-------------+-------+
| Column Name | Type  |
+-------------+-------+
| sale_id     | int   |
| product_id  | int   |
| year        | int   |
| quantity    | int   |
| price       | int   |
+-------------+-------+
(sale_id, year) is the primary key (combination of columns with unique values) of this
table.
product_id is a foreign key (reference column) to Product table.
Each row of this table shows a sale on the product product_id in a certain year.
Note that the price is per unit.
```

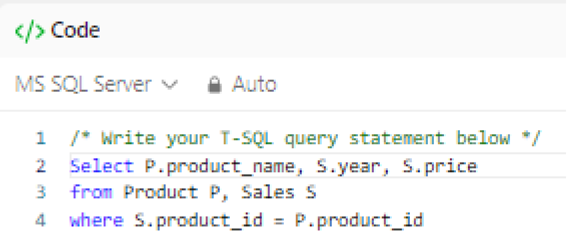
Table: `Product`

```
+--------------+---------+
| Column Name  | Type    |
+--------------+---------+
| product_id   | int     |
| product_name | varchar |
+--------------+---------+
product_id is the primary key (column with unique values) of this table.
Each row of this table indicates the product name of each product.
```

Write a solution to report the `product_name`, `year`, and `price` for each `sale_id` in the `Sales` table.

Return the resulting table in **any order**.

Answer



```sql
/* Write your T-SQL query statement below */
Select P.product_name, S.year, S.price
from Product P, Sales S
where S.product_id = P.product_id
```

Q: 08

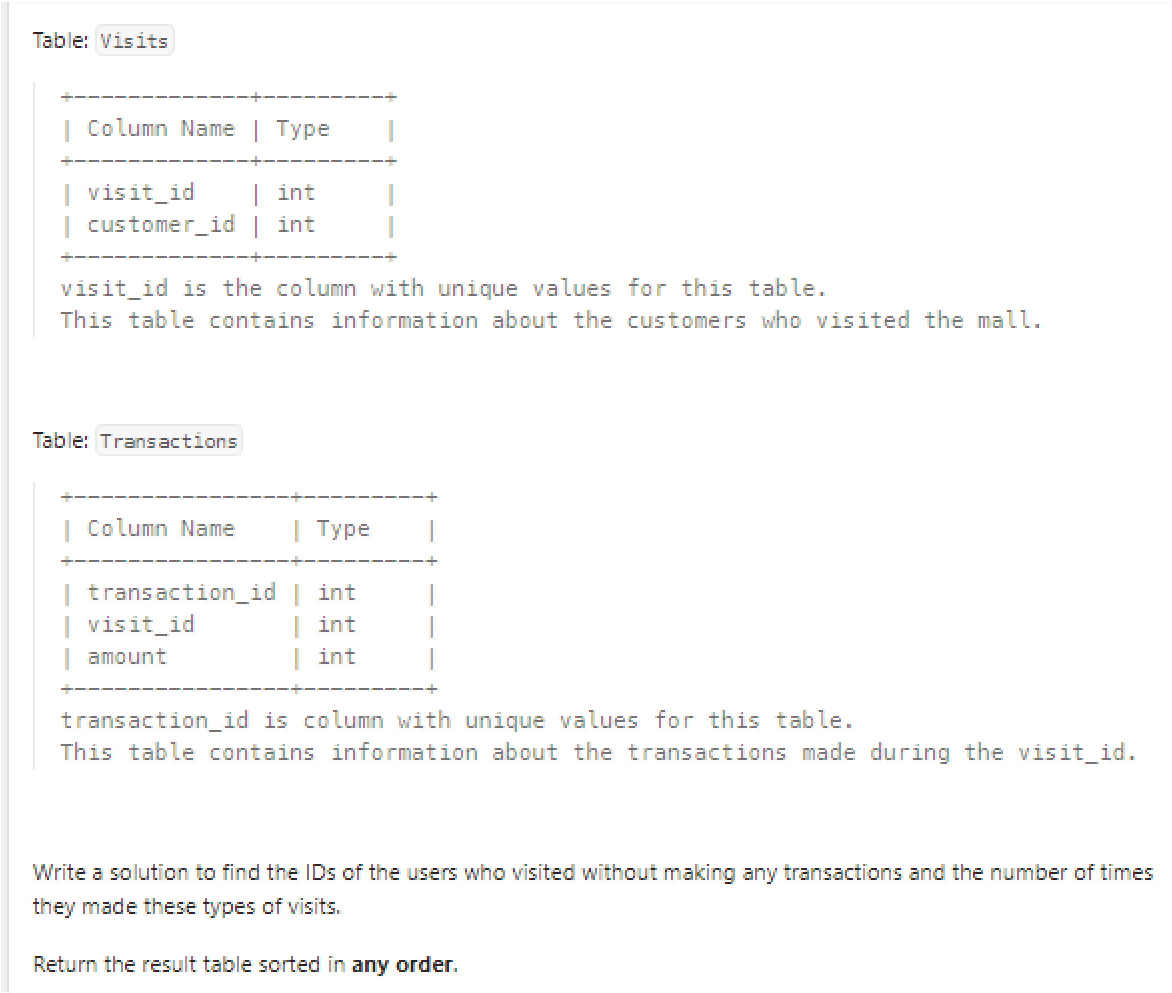


Table: `Visits`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| visit_id    | int     |
| customer_id | int     |
+-------------+---------+
visit_id is the column with unique values for this table.
This table contains information about the customers who visited the mall.
```

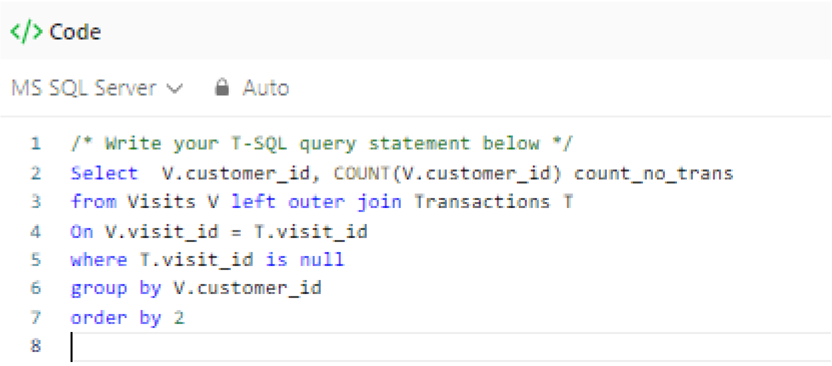
Table: `Transactions`

```
+----------------+---------+
| Column Name    | Type    |
+----------------+---------+
| transaction_id | int     |
| visit_id       | int     |
| amount         | int     |
+----------------+---------+
transaction_id is column with unique values for this table.
This table contains information about the transactions made during the visit_id.
```

Write a solution to find the IDs of the users who visited without making any transactions and the number of times they made these types of visits.

Return the result table sorted in **any order**.

Answer



```sql
/* Write your T-SQL query statement below */
Select  V.customer_id, COUNT(V.customer_id) count_no_trans
from Visits V left outer join Transactions T
On V.visit_id = T.visit_id
where T.visit_id is null
group by V.customer_id
order by 2
```

Q: 09

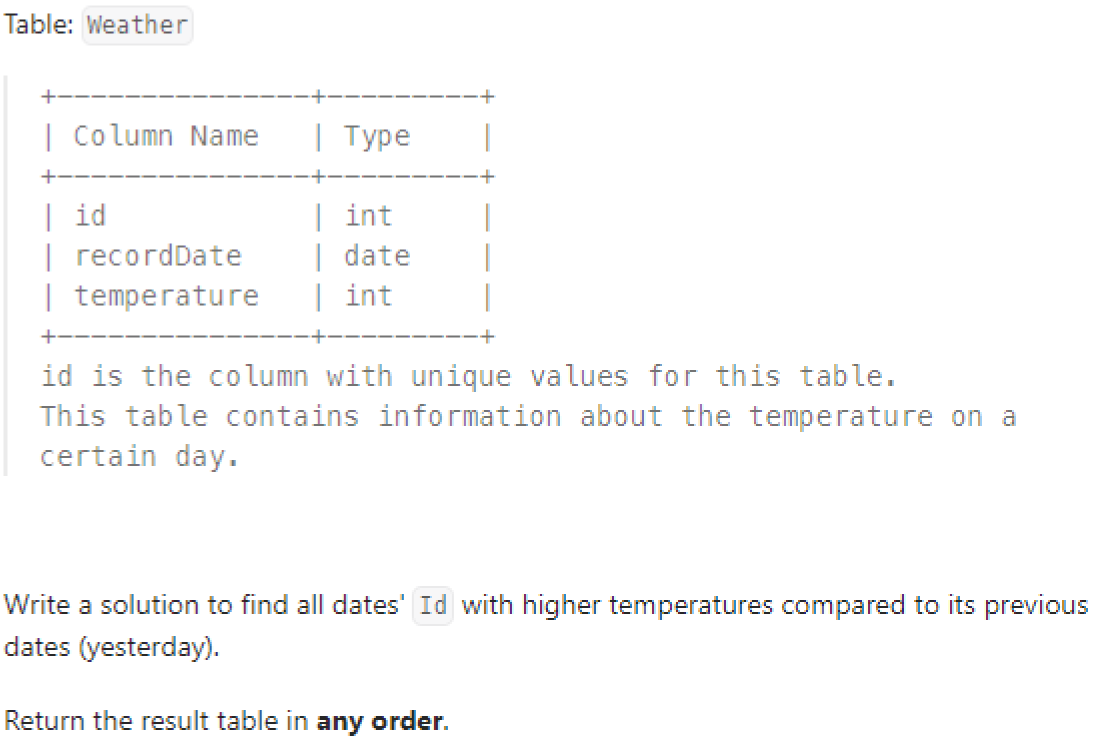


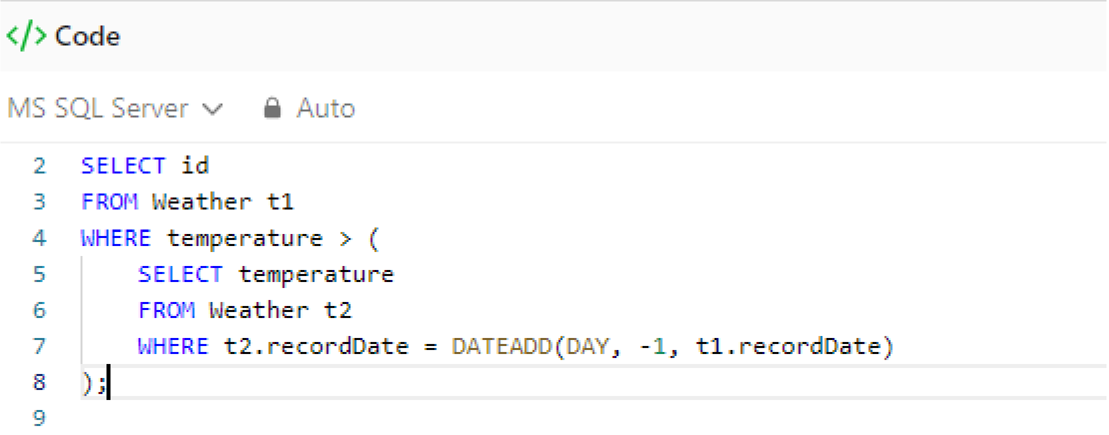
Table: `Weather`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| id            | int     |
| recordDate    | date    |
| temperature   | int     |
+---------------+---------+
id is the column with unique values for this table.
This table contains information about the temperature on a
certain day.
```

Write a solution to find all dates' `Id` with higher temperatures compared to its previous dates (yesterday).

Return the result table in **any order**.

Answer



```sql
SELECT id
FROM Weather t1
WHERE temperature > (
    SELECT temperature
    FROM Weather t2
    WHERE t2.recordDate = DATEADD(DAY, -1, t1.recordDate)
);
```

Q: 10

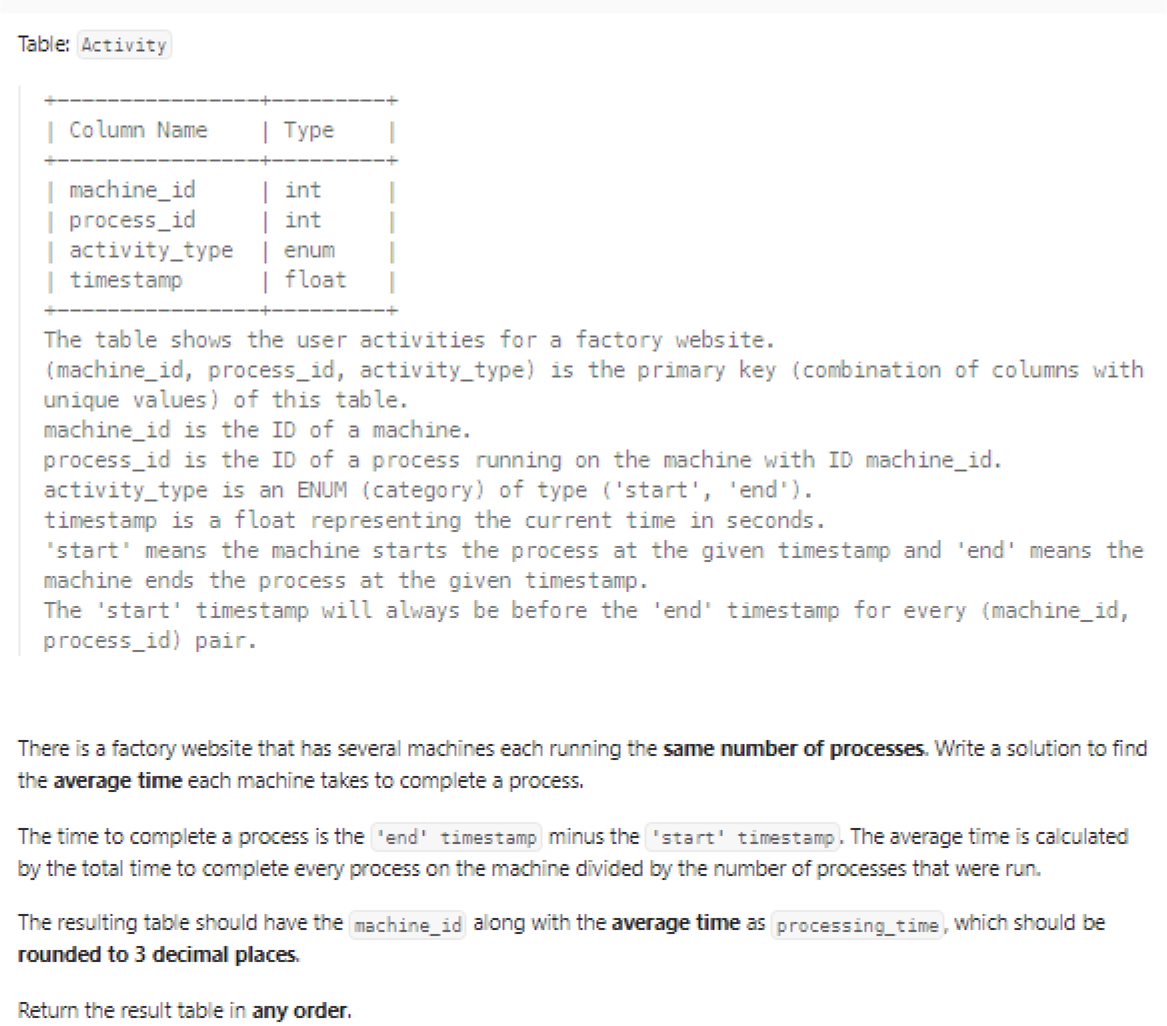


Table: `Activity`

```
+----------------+---------+
| Column Name    | Type    |
+----------------+---------+
| machine_id     | int     |
| process_id     | int     |
| activity_type  | enum    |
| timestamp      | float   |
+----------------+---------+
The table shows the user activities for a factory website.
(machine_id, process_id, activity_type) is the primary key (combination of columns with
unique values) of this table.
machine_id is the ID of a machine.
process_id is the ID of a process running on the machine with ID machine_id.
activity_type is an ENUM (category) of type ('start', 'end').
timestamp is a float representing the current time in seconds.
'start' means the machine starts the process at the given timestamp and 'end' means the
machine ends the process at the given timestamp.
The 'start' timestamp will always be before the 'end' timestamp for every (machine_id,
process_id) pair.
```

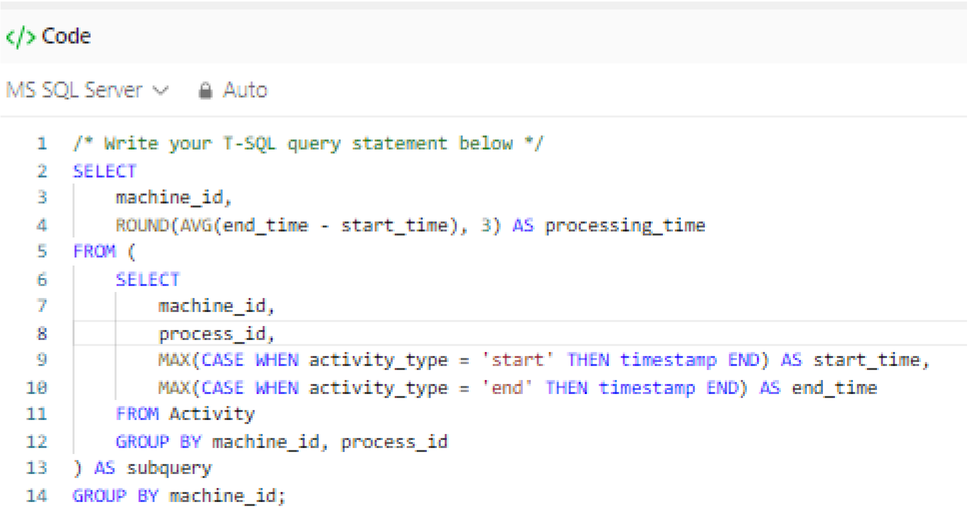
There is a factory website that has several machines each running the **same number of processes**. Write a solution to find the **average time** each machine takes to complete a process.

The time to complete a process is the `'end' timestamp` minus the `'start' timestamp`. The average time is calculated by the total time to complete every process on the machine divided by the number of processes that were run.

The resulting table should have the `machine_id` along with the **average time** as `processing_time`, which should be **rounded to 3 decimal places**.

Return the result table in **any order**.

Answer



```sql
/* Write your T-SQL query statement below */
SELECT
    machine_id,
    ROUND(AVG(end_time - start_time), 3) AS processing_time
FROM (
    SELECT
        machine_id,
        process_id,
        MAX(CASE WHEN activity_type = 'start' THEN timestamp END) AS start_time,
        MAX(CASE WHEN activity_type = 'end' THEN timestamp END) AS end_time
    FROM Activity
    GROUP BY machine_id, process_id
) AS subquery
GROUP BY machine_id;
```

Q: 11

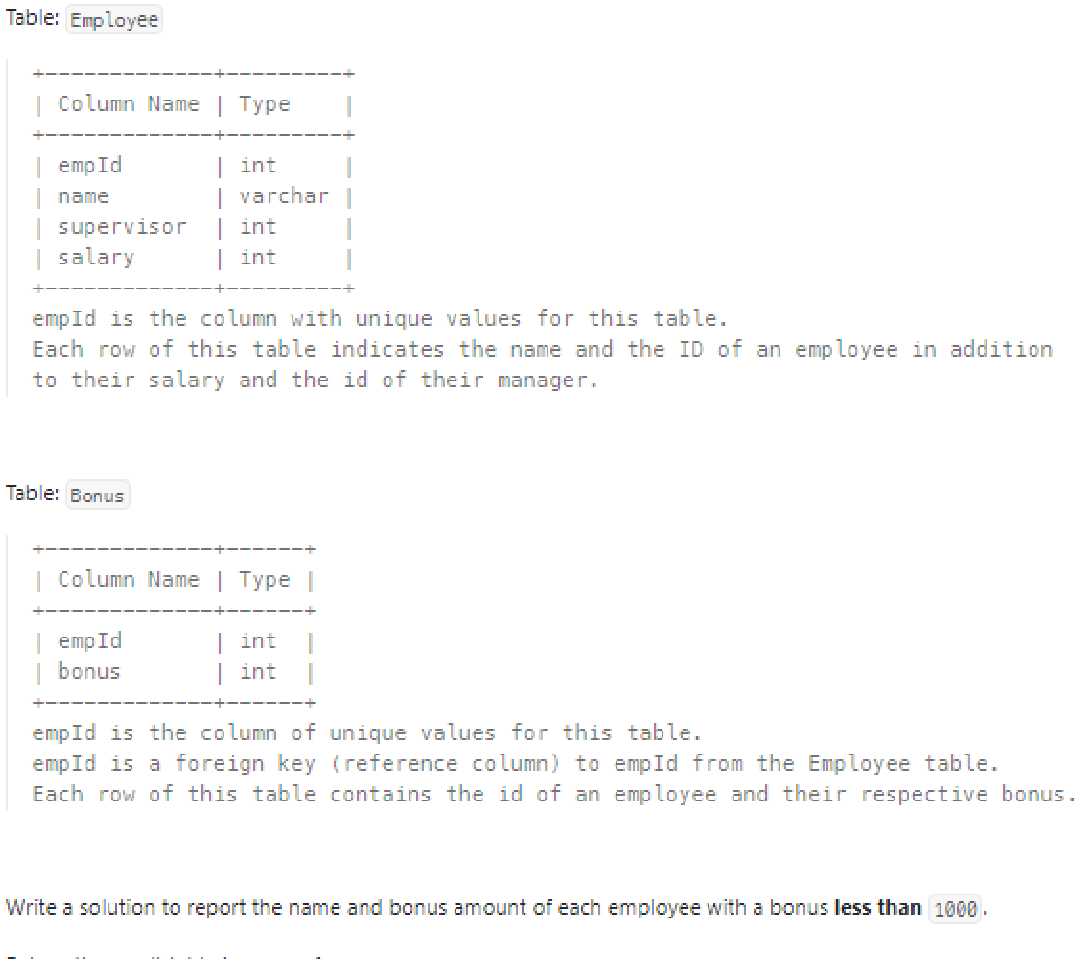


Table: `Employee`

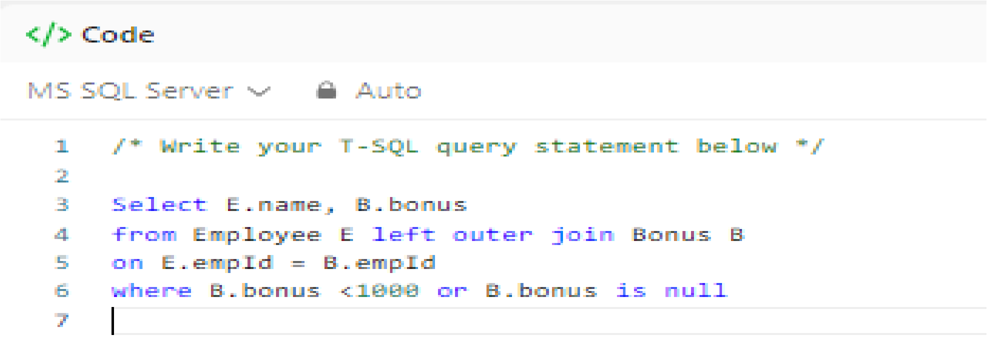
```
+--------------+---------+
| Column Name  | Type    |
+--------------+---------+
| empId        | int     |
| name         | varchar |
| supervisor   | int     |
| salary       | int     |
+--------------+---------+
empId is the column with unique values for this table.
Each row of this table indicates the name and the ID of an employee in addition
to their salary and the id of their manager.
```

Table: `Bonus`

```
+-------------+-------+
| Column Name | Type  |
+-------------+-------+
| empId       | int   |
| bonus       | int   |
+-------------+-------+
empId is the column of unique values for this table.
empId is a foreign key (reference column) to empId from the Employee table.
Each row of this table contains the id of an employee and their respective bonus.
```

Write a solution to report the name and bonus amount of each employee with a bonus **less than** `1000`.

Answer



```
/* Write your T-SQL query statement below */

Select E.name, B.bonus
from Employee E left outer join Bonus B
on E.empId = B.empId
where B.bonus <1000 or B.bonus is null
```

Q: 12

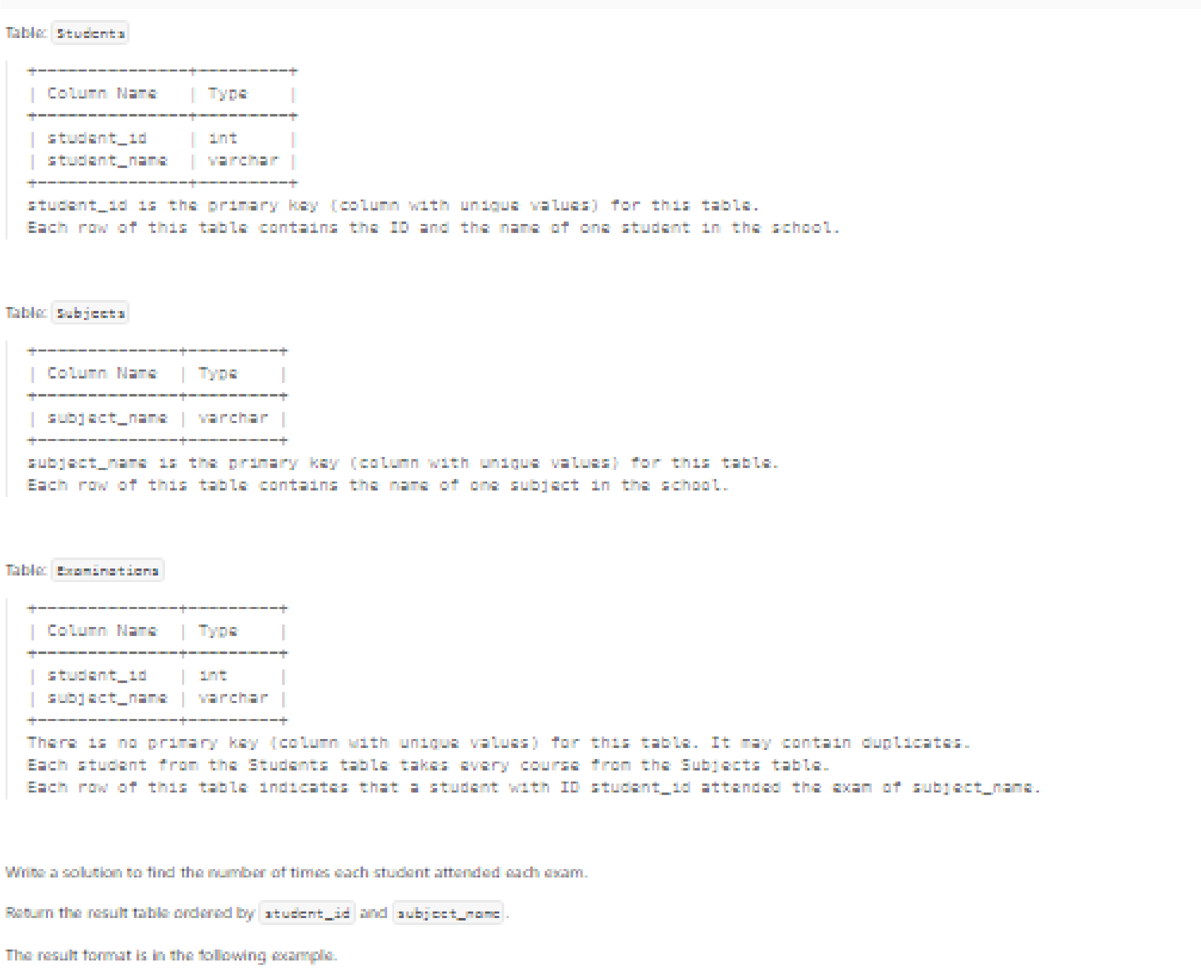


Table: `Students`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| student_id    | int     |
| student_name  | varchar |
+---------------+---------+
student_id is the primary key (column with unique values) for this table.
Each row of this table contains the ID and the name of one student in the school.
```

Table: `Subjects`

```
+--------------+---------+
| Column Name  | Type    |
+--------------+---------+
| subject_name | varchar |
+--------------+---------+
subject_name is the primary key (column with unique values) for this table.
Each row of this table contains the name of one subject in the school.
```

Table: `Examinations`

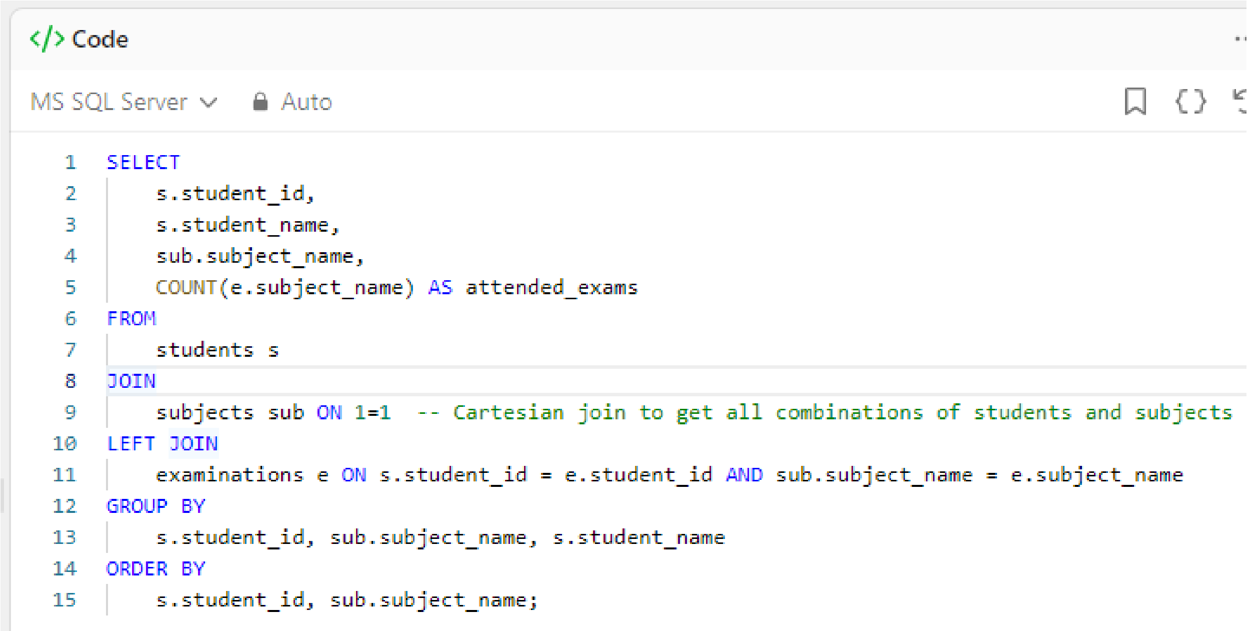
```
+--------------+---------+
| Column Name  | Type    |
+--------------+---------+
| student_id   | int     |
| subject_name | varchar |
+--------------+---------+
There is no primary key (column with unique values) for this table. It may contain duplicates.
Each student from the Students table takes every course from the Subjects table.
Each row of this table indicates that a student with ID student_id attended the exam of subject_name.
```

Write a solution to find the number of times each student attended each exam.

Return the result table ordered by `student_id` and `subject_name`.

The result format is in the following example.

Answer



```sql
SELECT
    s.student_id,
    s.student_name,
    sub.subject_name,
    COUNT(e.subject_name) AS attended_exams
FROM
    students s
JOIN
    subjects sub ON 1=1  -- Cartesian join to get all combinations of students and subjects
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, s.student_name
ORDER BY
    s.student_id, sub.subject_name;
```

Q: 13

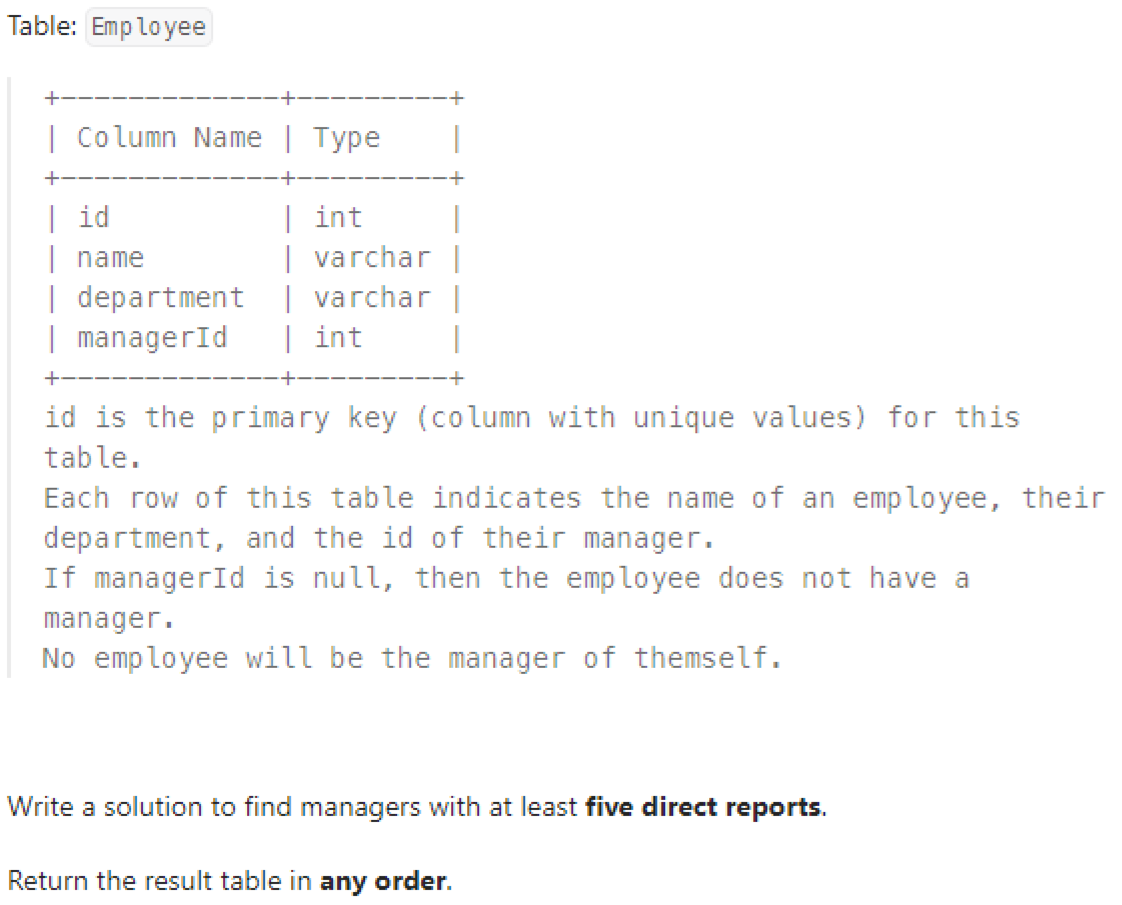


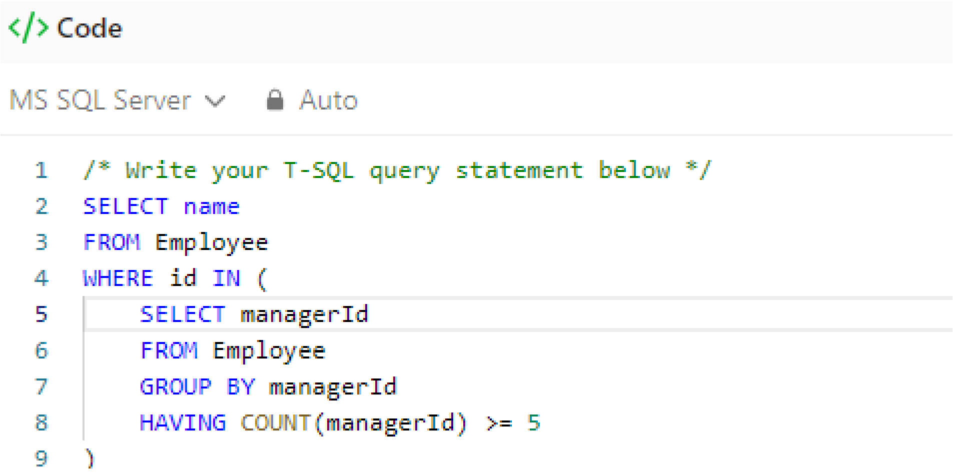
Table: `Employee`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| id          | int     |
| name        | varchar |
| department  | varchar |
| managerId   | int     |
+-------------+---------+
id is the primary key (column with unique values) for this table.
Each row of this table indicates the name of an employee, their department, and the id of their manager.
If managerId is null, then the employee does not have a manager.
No employee will be the manager of themself.
```

Write a solution to find managers with at least **five direct reports**.

Return the result table in **any order**.

Answer



```sql
/* Write your T-SQL query statement below */
SELECT name
FROM Employee
WHERE id IN (
    SELECT managerId
    FROM Employee
    GROUP BY managerId
    HAVING COUNT(managerId) >= 5
)
```

Q: 14

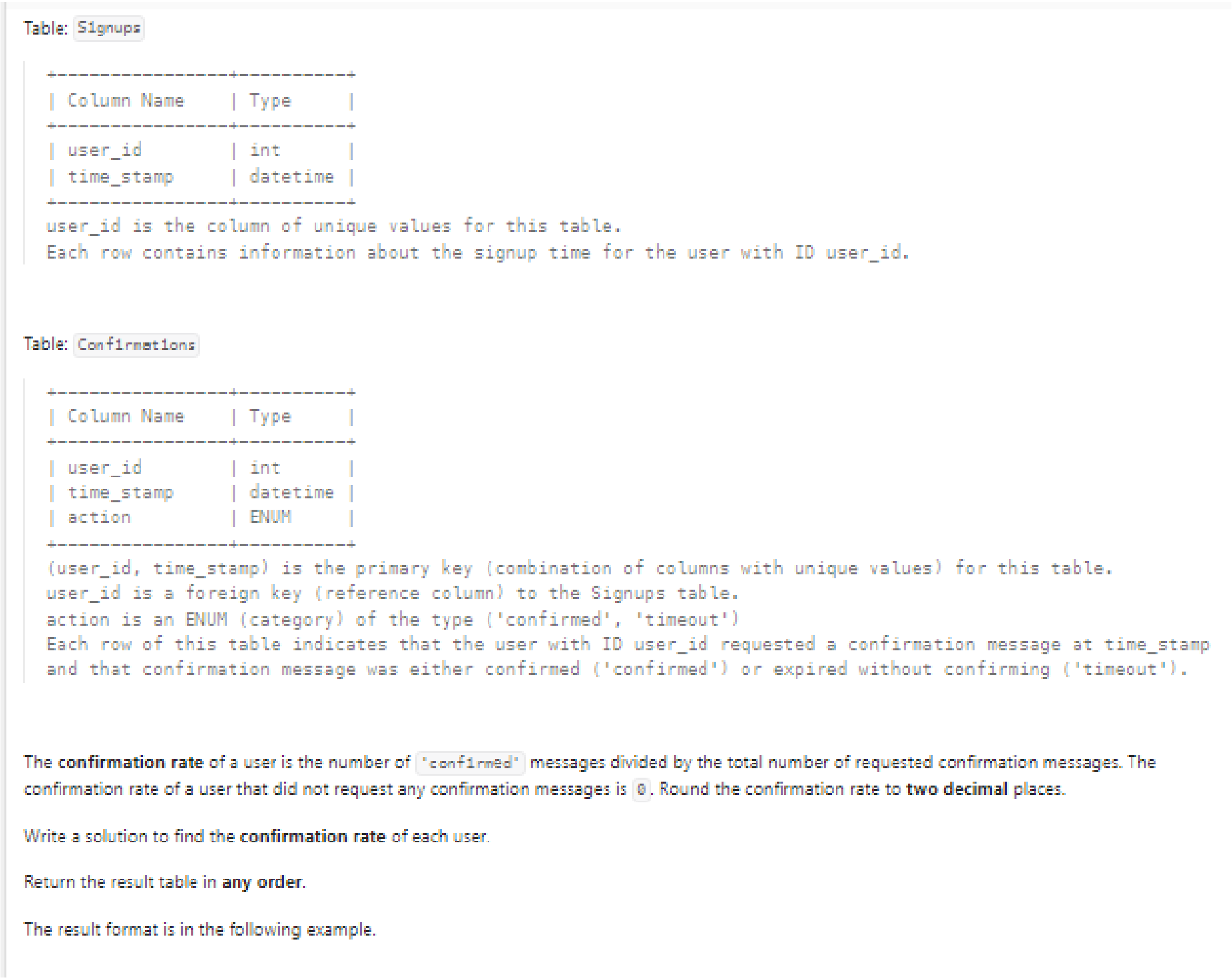


Table: `Signups`

```
+----------------+----------+
| Column Name    | Type     |
+----------------+----------+
| user_id        | int      |
| time_stamp     | datetime |
+----------------+----------+
user_id is the column of unique values for this table.
Each row contains information about the signup time for the user with ID user_id.
```

Table: `Confirmations`

```
+----------------+----------+
| Column Name    | Type     |
+----------------+----------+
| user_id        | int      |
| time_stamp     | datetime |
| action         | ENUM     |
+----------------+----------+
(user_id, time_stamp) is the primary key (combination of columns with unique values) for this table.
user_id is a foreign key (reference column) to the Signups table.
action is an ENUM (category) of the type ('confirmed', 'timeout')
Each row of this table indicates that the user with ID user_id requested a confirmation message at time_stamp
and that confirmation message was either confirmed ('confirmed') or expired without confirming ('timeout').
```

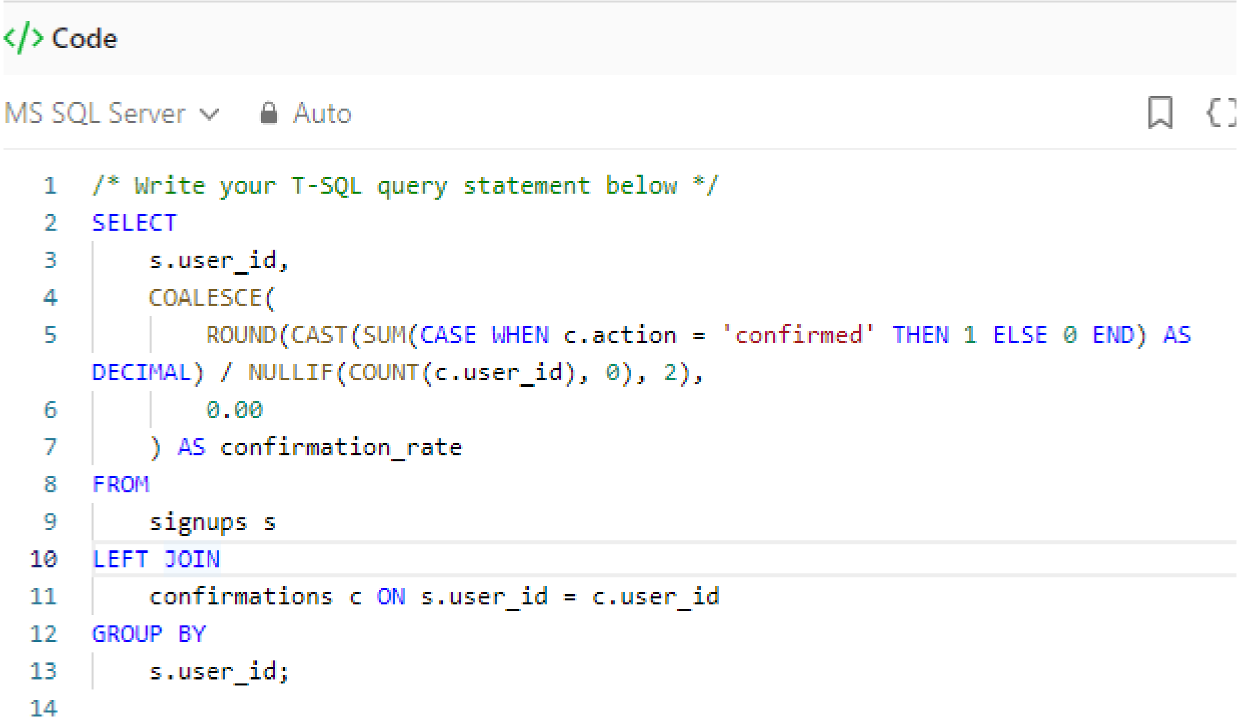
The **confirmation rate** of a user is the number of `'confirmed'` messages divided by the total number of requested confirmation messages. The confirmation rate of a user that did not request any confirmation messages is `0`. Round the confirmation rate to **two decimal** places.

Write a solution to find the **confirmation rate** of each user.

Return the result table in **any order**.

The result format is in the following example.

Answer



```sql
/* Write your T-SQL query statement below */
SELECT
    s.user_id,
    COALESCE(
        ROUND(CAST(SUM(CASE WHEN c.action = 'confirmed' THEN 1 ELSE 0 END) AS DECIMAL) / NULLIF(COUNT(c.user_id), 0), 2),
        0.00
    ) AS confirmation_rate
FROM
    signups s
LEFT JOIN
    confirmations c ON s.user_id = c.user_id
GROUP BY
    s.user_id;
```

❖ Basic Aggregate functions

Q: 15

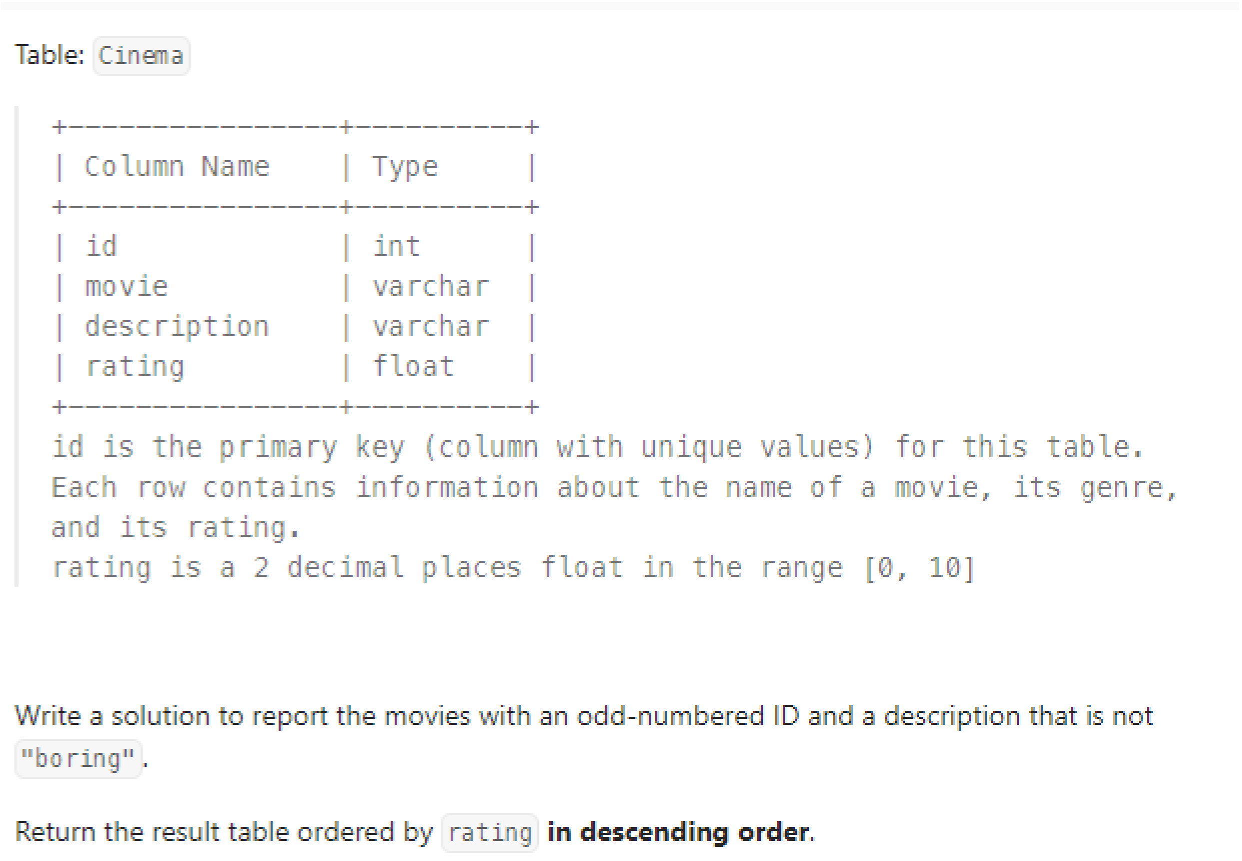


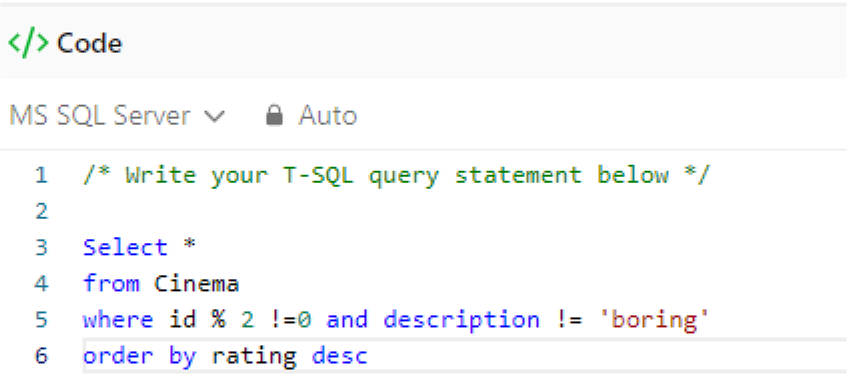
Table: `Cinema`

```
+----------------+----------+
| Column Name    | Type     |
+----------------+----------+
| id             | int      |
| movie          | varchar  |
| description    | varchar  |
| rating         | float    |
+----------------+----------+
id is the primary key (column with unique values) for this table.
Each row contains information about the name of a movie, its genre,
and its rating.
rating is a 2 decimal places float in the range [0, 10]
```

Write a solution to report the movies with an odd-numbered ID and a description that is not `"boring"`.

Return the result table ordered by `rating` **in descending order**.

Answer



```sql
/* Write your T-SQL query statement below */

Select *
from Cinema
where id % 2 !=0 and description != 'boring'
order by rating desc
```

Q: 16

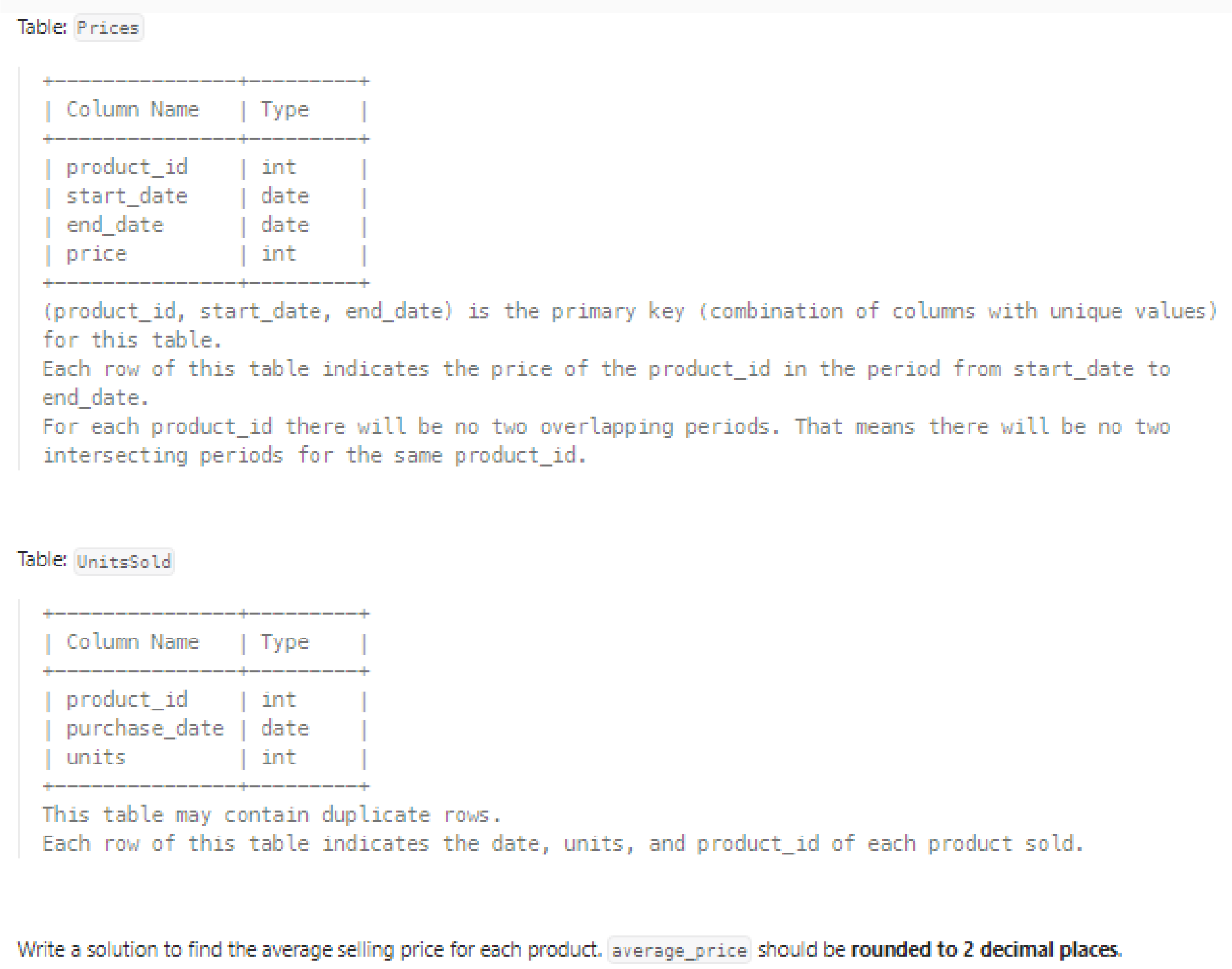


Table: `Prices`

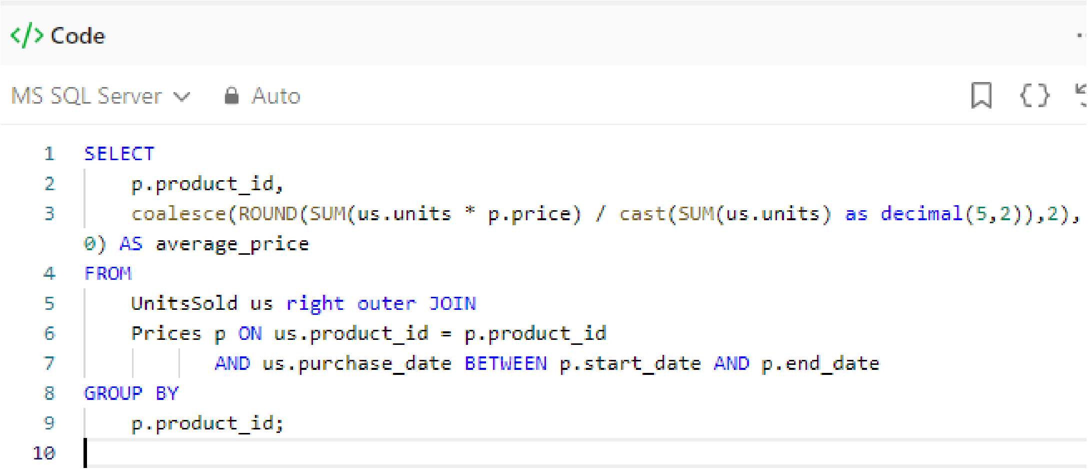
```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| product_id    | int     |
| start_date    | date    |
| end_date      | date    |
| price         | int     |
+---------------+---------+
(product_id, start_date, end_date) is the primary key (combination of columns with unique values)
for this table.
Each row of this table indicates the price of the product_id in the period from start_date to
end_date.
For each product_id there will be no two overlapping periods. That means there will be no two
intersecting periods for the same product_id.
```

Table: `UnitsSold`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| product_id    | int     |
| purchase_date | date    |
| units         | int     |
+---------------+---------+
This table may contain duplicate rows.
Each row of this table indicates the date, units, and product_id of each product sold.
```

Write a solution to find the average selling price for each product. `average_price` should be **rounded to 2 decimal places**.

Answer



```sql
SELECT
    p.product_id,
    coalesce(ROUND(SUM(us.units * p.price) / cast(SUM(us.units) as decimal(5,2)),2), 0) AS average_price
FROM
    UnitsSold us right outer JOIN
    Prices p ON us.product_id = p.product_id
        AND us.purchase_date BETWEEN p.start_date AND p.end_date
GROUP BY
    p.product_id;
```

Q: 17

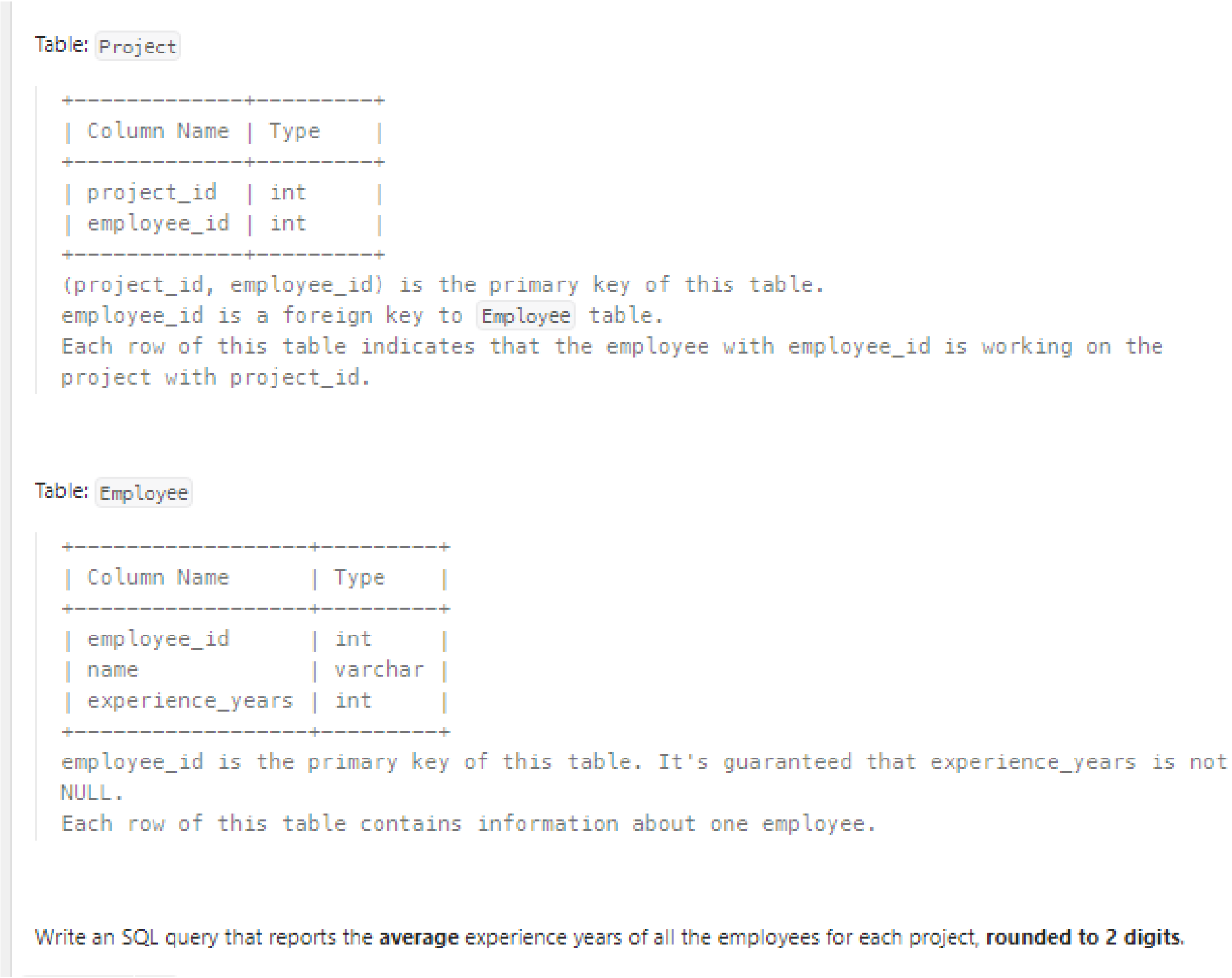


Table: `Project`

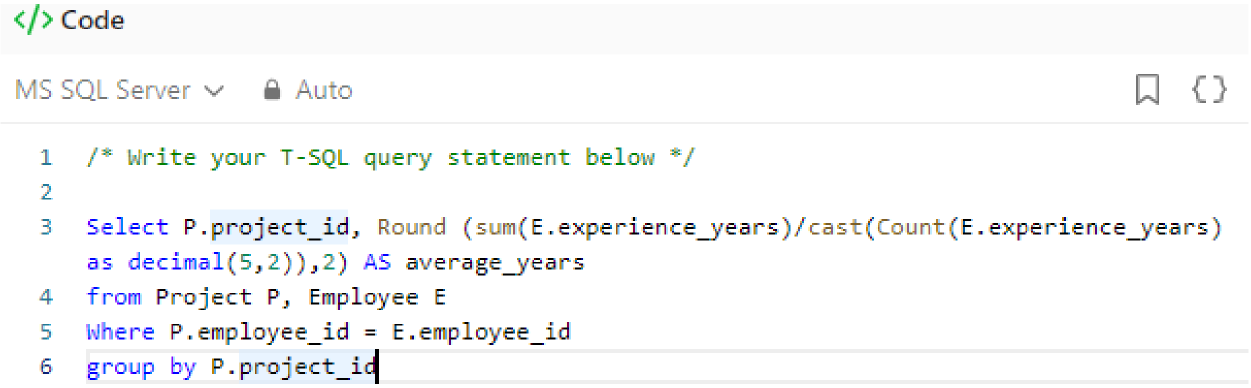
```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| project_id  | int     |
| employee_id | int     |
+-------------+---------+
(project_id, employee_id) is the primary key of this table.
employee_id is a foreign key to Employee table.
Each row of this table indicates that the employee with employee_id is working on the
project with project_id.
```

Table: `Employee`

```
+------------------+---------+
| Column Name      | Type    |
+------------------+---------+
| employee_id      | int     |
| name             | varchar |
| experience_years | int     |
+------------------+---------+
employee_id is the primary key of this table. It's guaranteed that experience_years is not
NULL.
Each row of this table contains information about one employee.
```

Write an SQL query that reports the **average** experience years of all the employees for each project, **rounded to 2 digits.**

Answer



```sql
/* Write your T-SQL query statement below */

Select P.project_id, Round (sum(E.experience_years)/cast(Count(E.experience_years) as decimal(5,2)),2) AS average_years
from Project P, Employee E
Where P.employee_id = E.employee_id
group by P.project_id
```

Q: 18

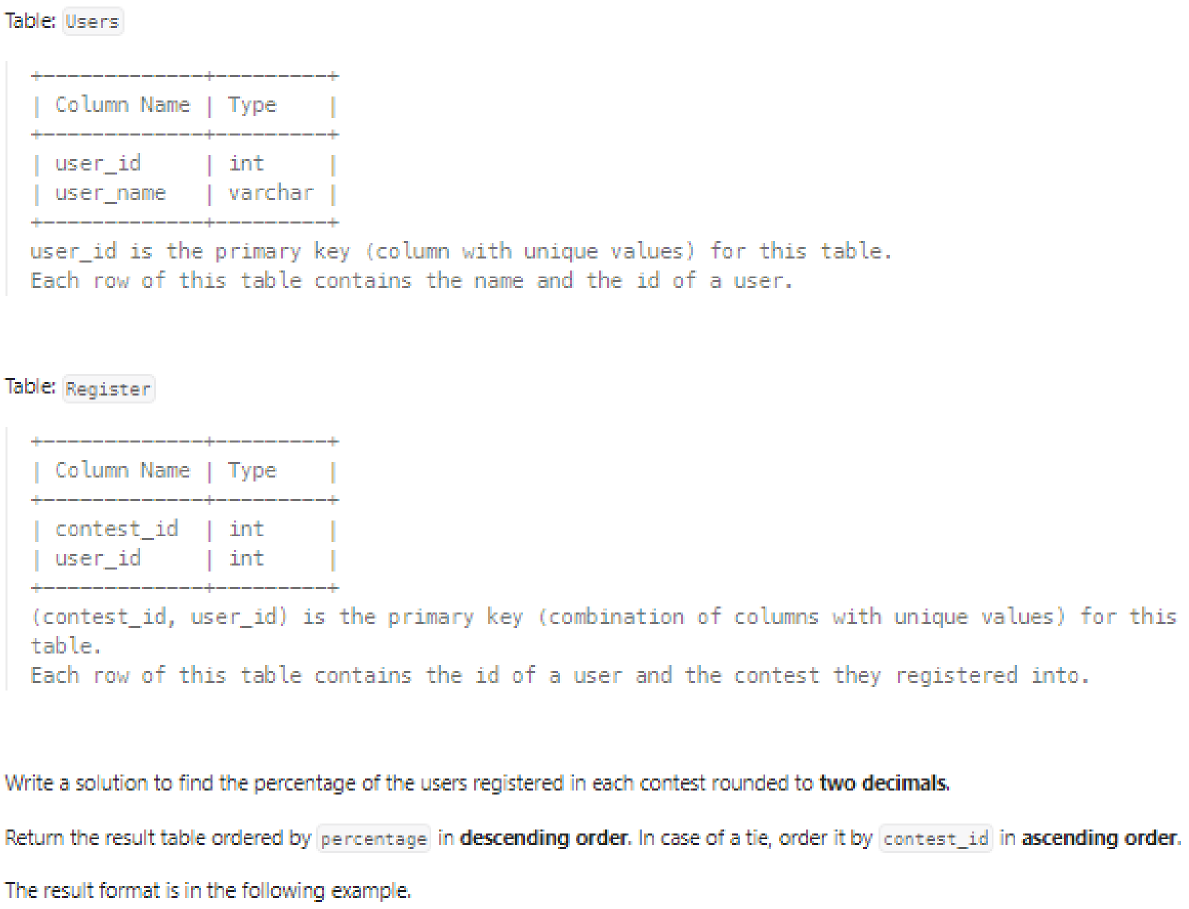


Table: `Users`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| user_id       | int     |
| user_name     | varchar |
+---------------+---------+
user_id is the primary key (column with unique values) for this table.
Each row of this table contains the name and the id of a user.
```

Table: `Register`

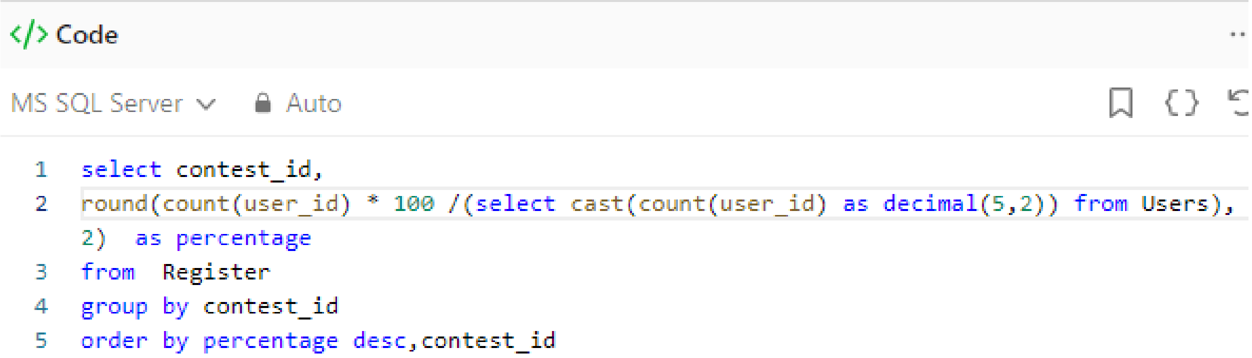
```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| contest_id    | int     |
| user_id       | int     |
+---------------+---------+
(contest_id, user_id) is the primary key (combination of columns with unique values) for this
table.
Each row of this table contains the id of a user and the contest they registered into.
```

Write a solution to find the percentage of the users registered in each contest rounded to **two decimals**.

Return the result table ordered by `percentage` in **descending order**. In case of a tie, order it by `contest_id` in **ascending order**.

The result format is in the following example.

Answer



```sql
select contest_id,
round(count(user_id) * 100 /(select cast(count(user_id) as decimal(5,2)) from Users),
2)  as percentage
from  Register
group by contest_id
order by percentage desc,contest_id
```

Q: 19

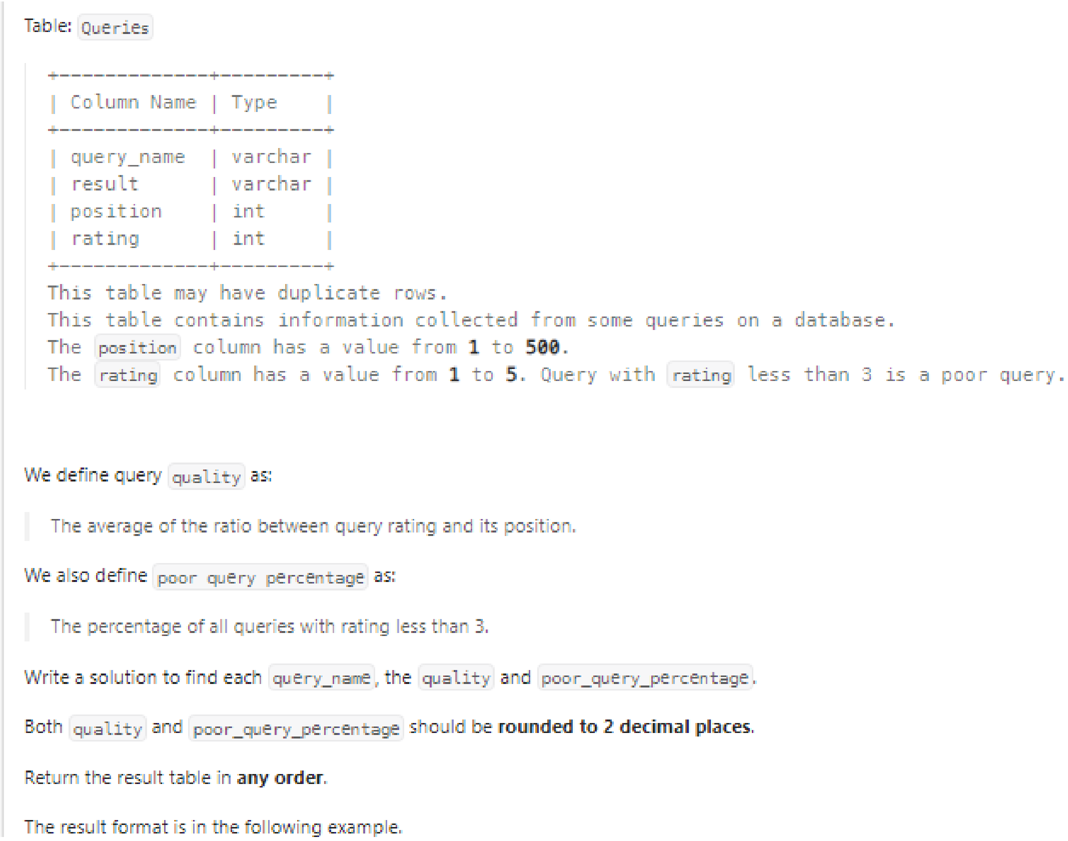


Table: `Queries`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| query_name  | varchar |
| result      | varchar |
| position    | int     |
| rating      | int     |
+-------------+---------+
This table may have duplicate rows.
This table contains information collected from some queries on a database.
The position column has a value from 1 to 500.
The rating column has a value from 1 to 5. Query with rating less than 3 is a poor query.
```

We define query `quality` as:

> The average of the ratio between query rating and its position.

We also define `poor query percentage` as:

> The percentage of all queries with rating less than 3.

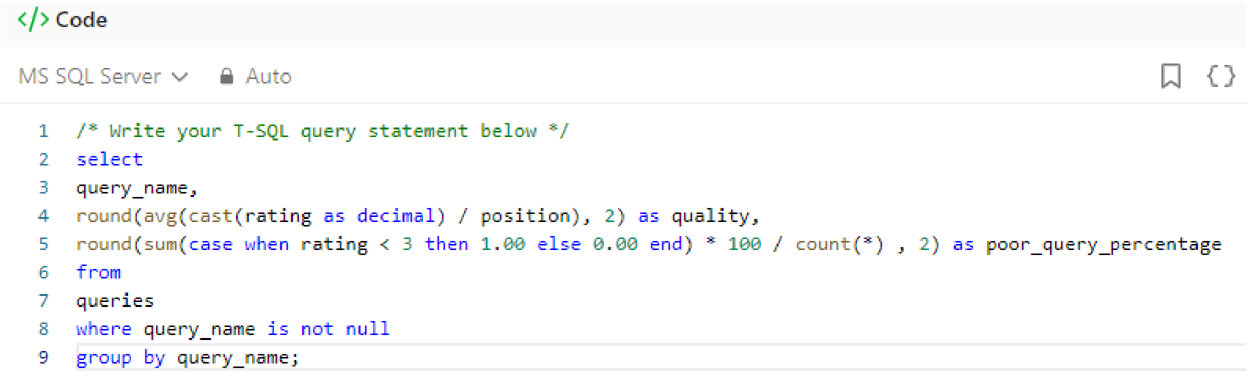
Write a solution to find each `query_name`, the `quality` and `poor_query_percentage`.

Both `quality` and `poor_query_percentage` should be **rounded to 2 decimal places**.

Return the result table in **any order**.

The result format is in the following example.

Answer

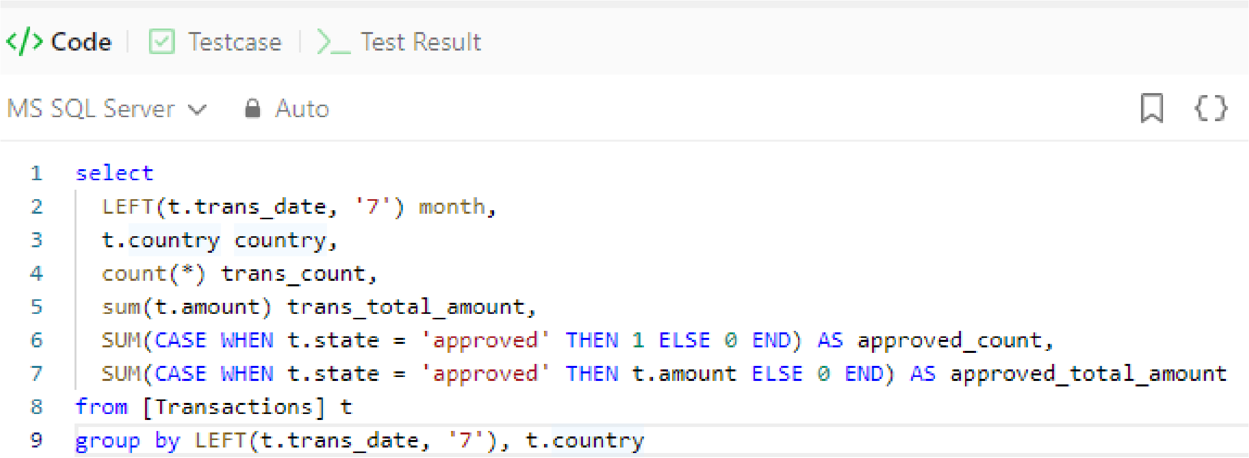


```sql
/* Write your T-SQL query statement below */
select
query_name,
round(avg(cast(rating as decimal) / position), 2) as quality,
round(sum(case when rating < 3 then 1.00 else 0.00 end) * 100 / count(*) , 2) as poor_query_percentage
from
queries
where query_name is not null
group by query_name;
```

Q: 20

> Table: `Transactions`
>
> ```
> +----------------+---------+
> | Column Name    | Type    |
> +----------------+---------+
> | id             | int     |
> | country        | varchar |
> | state          | enum    |
> | amount         | int     |
> | trans_date     | date    |
> +----------------+---------+
> id is the primary key of this table.
> The table has information about incoming transactions.
> The state column is an enum of type ["approved", "declined"].
> ```
>
> 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.

Answer



```sql
select
  LEFT(t.trans_date, '7') month,
  t.country country,
  count(*) trans_count,
  sum(t.amount) trans_total_amount,
  SUM(CASE WHEN t.state = 'approved' THEN 1 ELSE 0 END) AS approved_count,
  SUM(CASE WHEN t.state = 'approved' THEN t.amount ELSE 0 END) AS approved_total_amount
from [Transactions] t
group by LEFT(t.trans_date, '7'), t.country
```

Q: 21

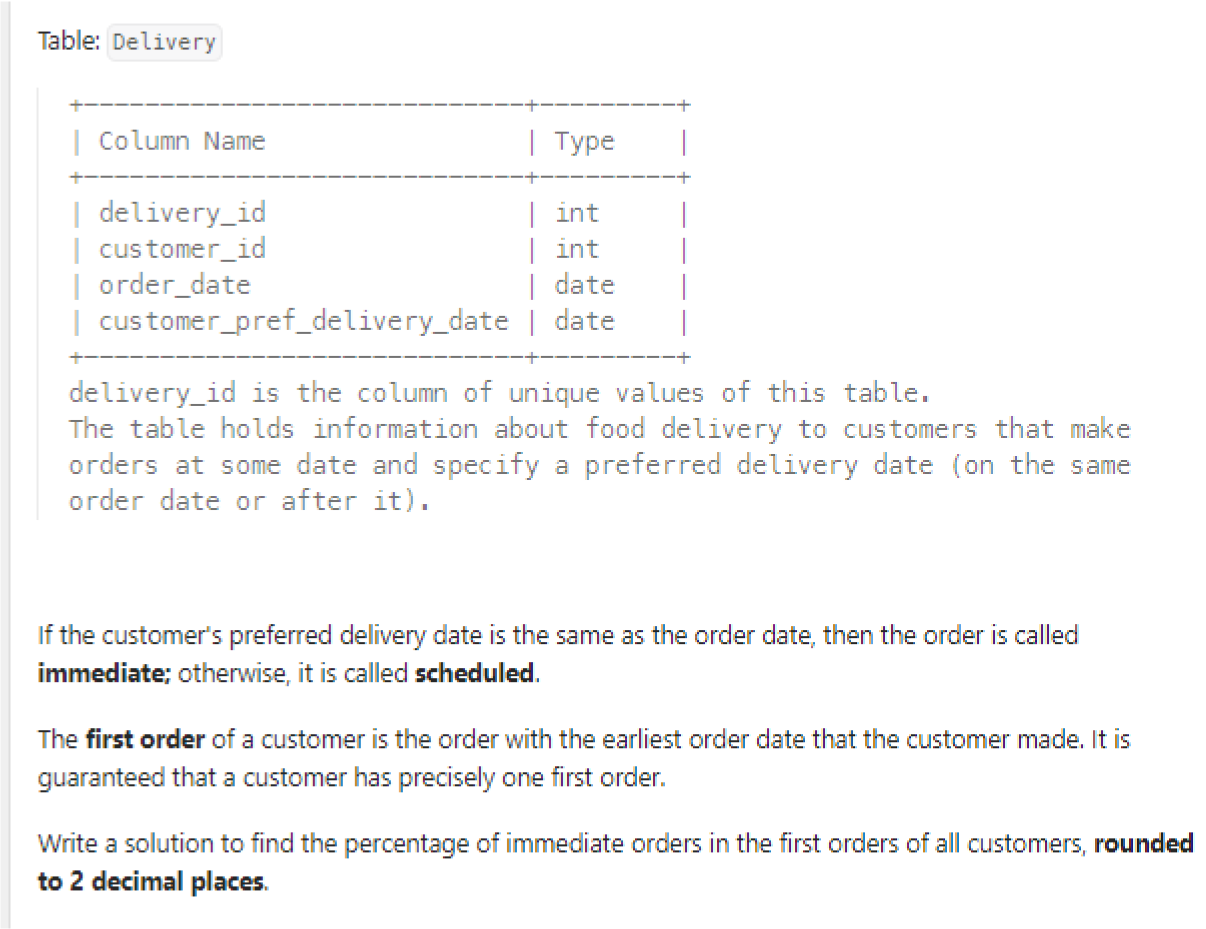


Table: `Delivery`

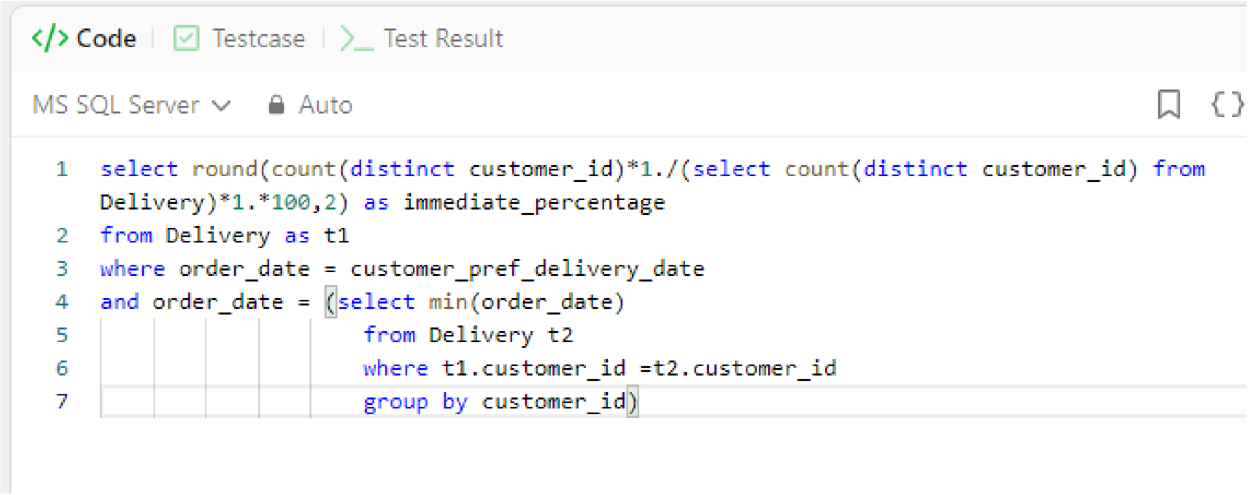
```
+-----------------------------+---------+
| Column Name                 | Type    |
+-----------------------------+---------+
| delivery_id                 | int     |
| customer_id                 | int     |
| order_date                  | date    |
| customer_pref_delivery_date | date    |
+-----------------------------+---------+
delivery_id is the column of unique values of this table.
The table holds information about food delivery to customers that make
orders at some date and specify a preferred delivery date (on the same
order date or after it).
```

If the customer's preferred delivery date is the same as the order date, then the order is called **immediate;** otherwise, it is called **scheduled**.

The **first order** of a customer is the order with the earliest order date that the customer made. It is guaranteed that a customer has precisely one first order.

Write a solution to find the percentage of immediate orders in the first orders of all customers, **rounded to 2 decimal places**.

Answer



```sql
select round(count(distinct customer_id)*1./(select count(distinct customer_id) from
Delivery)*1.*100,2) as immediate_percentage
from Delivery as t1
where order_date = customer_pref_delivery_date
and order_date = (select min(order_date)
                    from Delivery t2
                    where t1.customer_id =t2.customer_id
                    group by customer_id)
```

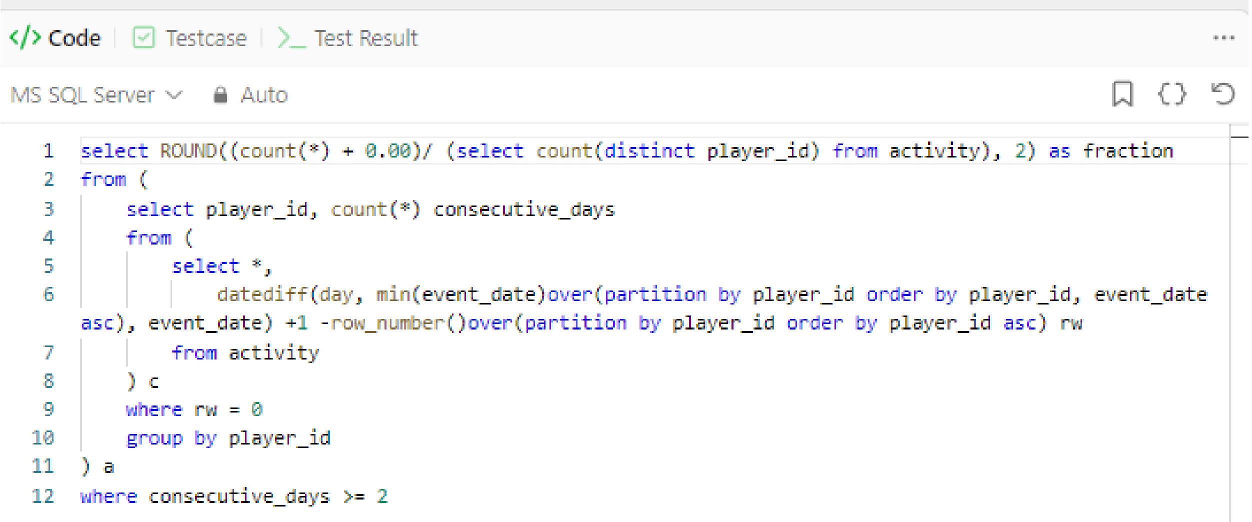
Q: 22

Table: `Activity`

```
+--------------+---------+
| Column Name  | Type    |
+--------------+---------+
| player_id    | int     |
| device_id    | int     |
| event_date   | date    |
| games_played | int     |
+--------------+---------+
(player_id, event_date) is the primary key (combination of columns with
unique values) of this table.
This table shows the activity of players of some games.
Each row is a record of a player who logged in and played a number of
games (possibly 0) before logging out on someday using some device.
```

Write a solution to report the **fraction** of players that logged in again on the day after the day they first logged in, **rounded to 2 decimal places**. In other words, you need to count the number of players that logged in for at least two consecutive days starting from their first login date, then divide that number by the total number of players.

Answer



```sql
select ROUND((count(*) + 0.00)/ (select count(distinct player_id) from activity), 2) as fraction
from (
    select player_id, count(*) consecutive_days
    from (
        select *,
            datediff(day, min(event_date)over(partition by player_id order by player_id, event_date asc), event_date) +1 -row_number()over(partition by player_id order by player_id asc) rw
        from activity
    ) c
    where rw = 0
    group by player_id
) a
where consecutive_days >= 2
```

❖ Sorting and Grouping Questions

Q: 23

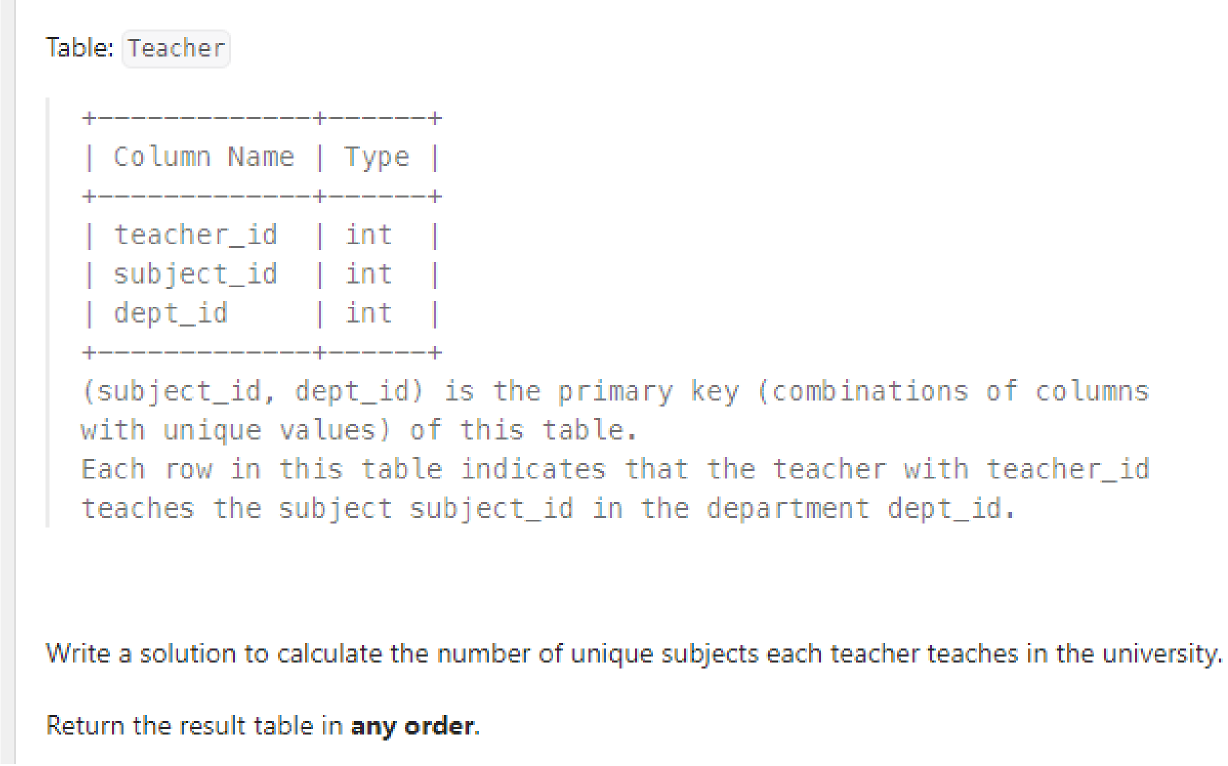


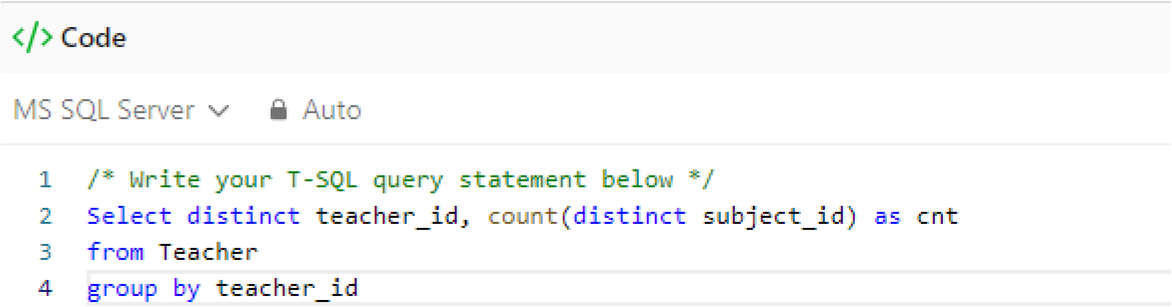
Table: `Teacher`

```
+-------------+------+
| Column Name | Type |
+-------------+------+
| teacher_id  | int  |
| subject_id  | int  |
| dept_id     | int  |
+-------------+------+
(subject_id, dept_id) is the primary key (combinations of columns
with unique values) of this table.
Each row in this table indicates that the teacher with teacher_id
teaches the subject subject_id in the department dept_id.
```

Write a solution to calculate the number of unique subjects each teacher teaches in the university.

Return the result table in **any order**.

Answer



```sql
/* Write your T-SQL query statement below */
Select distinct teacher_id, count(distinct subject_id) as cnt
from Teacher
group by teacher_id
```

Q: 24

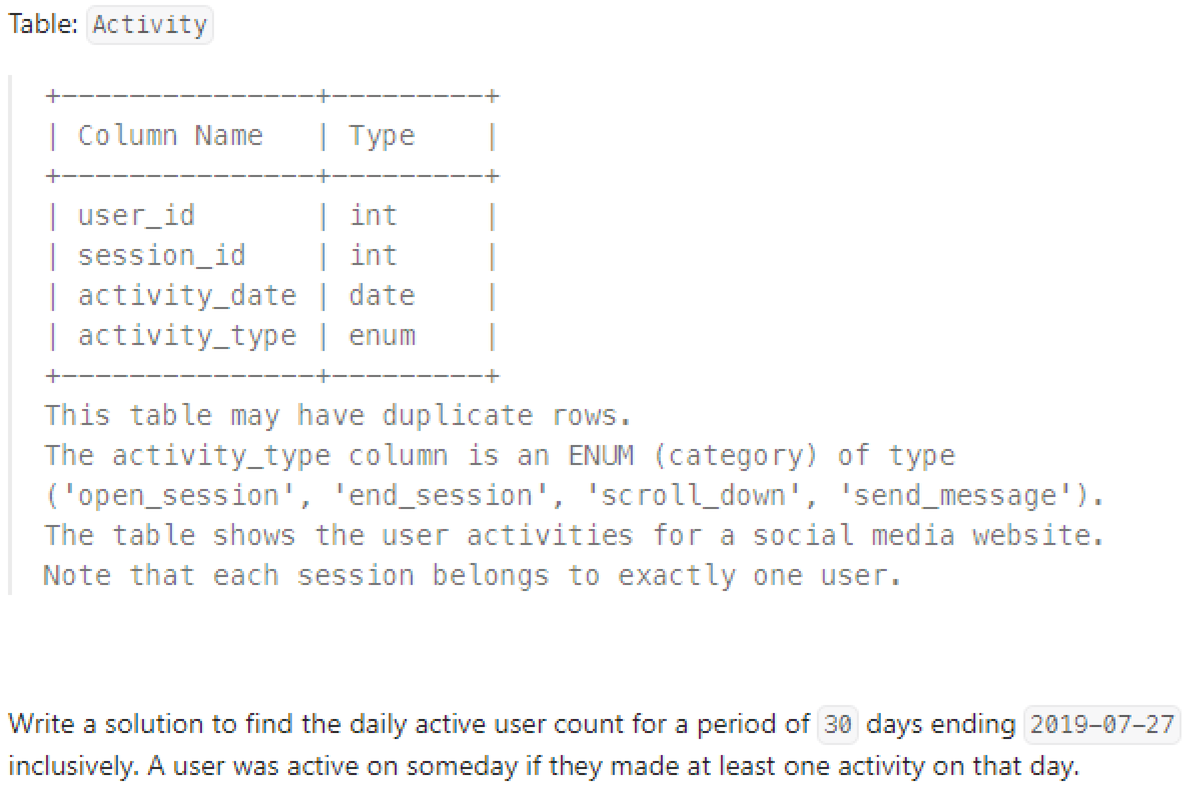
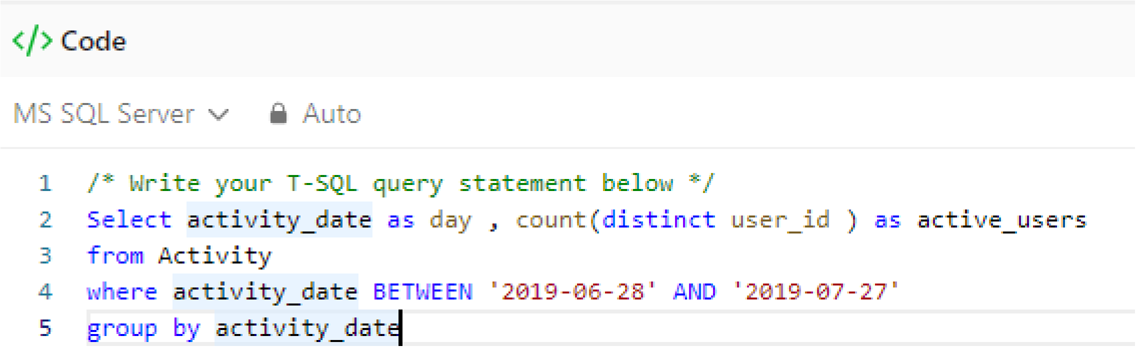


Table: `Activity`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| user_id       | int     |
| session_id    | int     |
| activity_date | date    |
| activity_type | enum    |
+---------------+---------+
This table may have duplicate rows.
The activity_type column is an ENUM (category) of type
('open_session', 'end_session', 'scroll_down', 'send_message').
The table shows the user activities for a social media website.
Note that each session belongs to exactly one user.
```

Write a solution to find the daily active user count for a period of `30` days ending `2019-07-27` inclusively. A user was active on someday if they made at least one activity on that day.

Answer



```sql
/* Write your T-SQL query statement below */
Select activity_date as day , count(distinct user_id ) as active_users
from Activity
where activity_date BETWEEN '2019-06-28' AND '2019-07-27'
group by activity_date
```

Q: 25

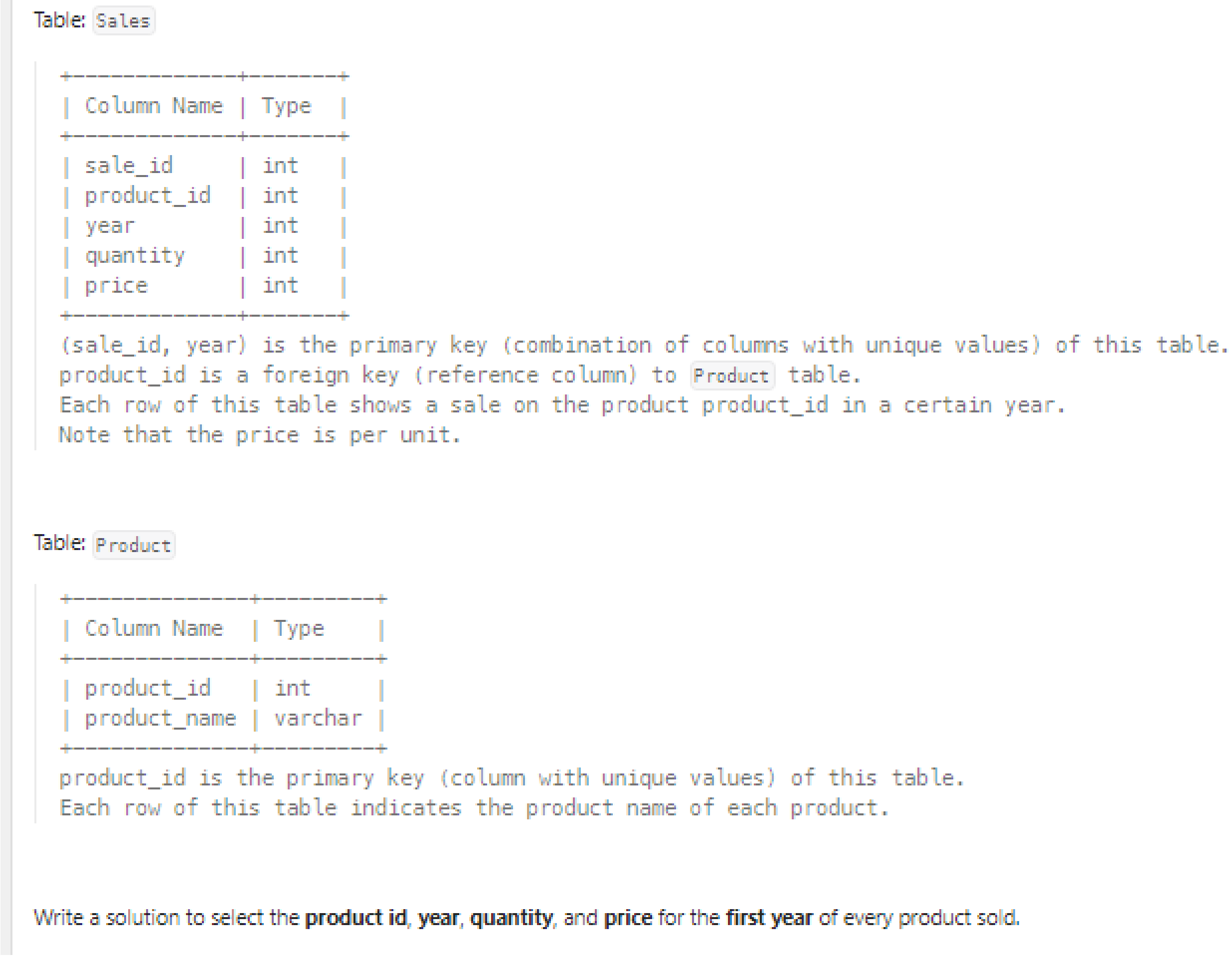


Table: Sales

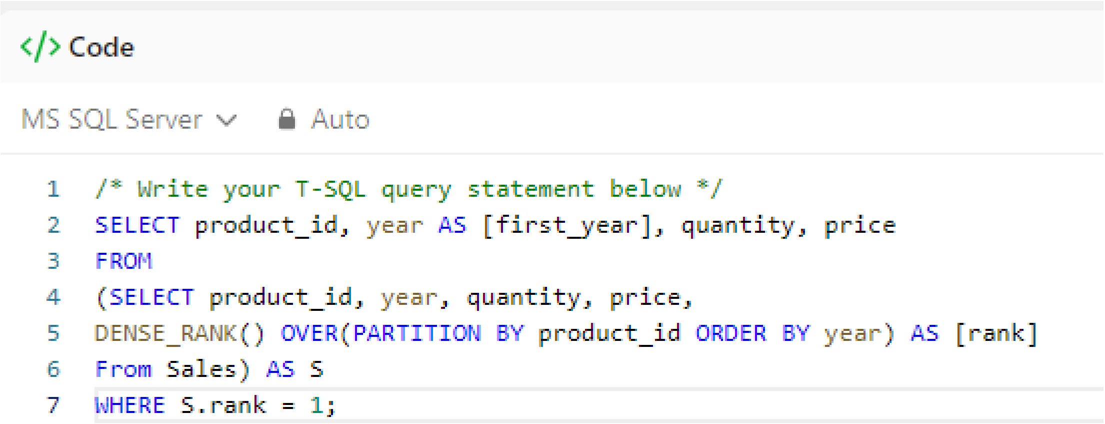
```
+-------------+-------+
| Column Name | Type  |
+-------------+-------+
| sale_id     | int   |
| product_id  | int   |
| year        | int   |
| quantity    | int   |
| price       | int   |
+-------------+-------+
(sale_id, year) is the primary key (combination of columns with unique values) of this table.
product_id is a foreign key (reference column) to Product table.
Each row of this table shows a sale on the product product_id in a certain year.
Note that the price is per unit.
```

Table: Product

```
+--------------+---------+
| Column Name  | Type    |
+--------------+---------+
| product_id   | int     |
| product_name | varchar |
+--------------+---------+
product_id is the primary key (column with unique values) of this table.
Each row of this table indicates the product name of each product.
```

Write a solution to select the **product id**, **year**, **quantity**, and **price** for the **first year** of every product sold.

Answer



```sql
/* Write your T-SQL query statement below */
SELECT product_id, year AS [first_year], quantity, price
FROM
(SELECT product_id, year, quantity, price,
DENSE_RANK() OVER(PARTITION BY product_id ORDER BY year) AS [rank]
From Sales) AS S
WHERE S.rank = 1;
```

Q: 26

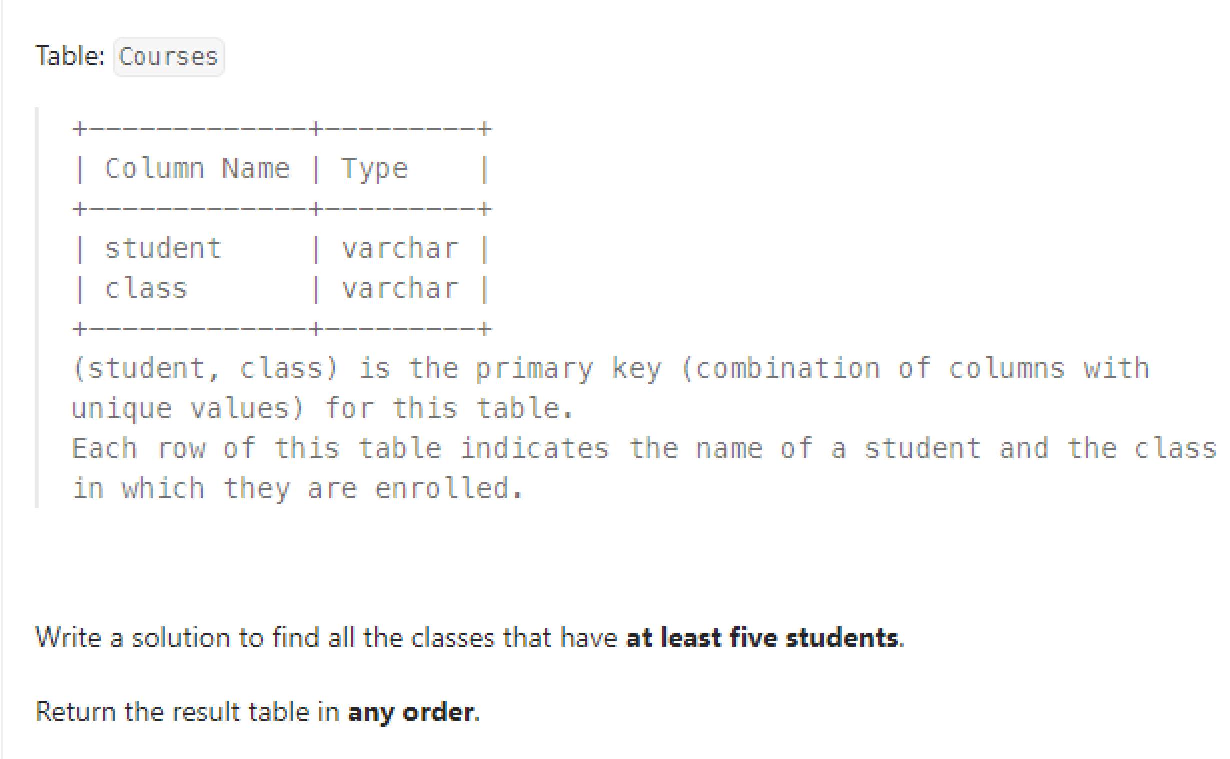


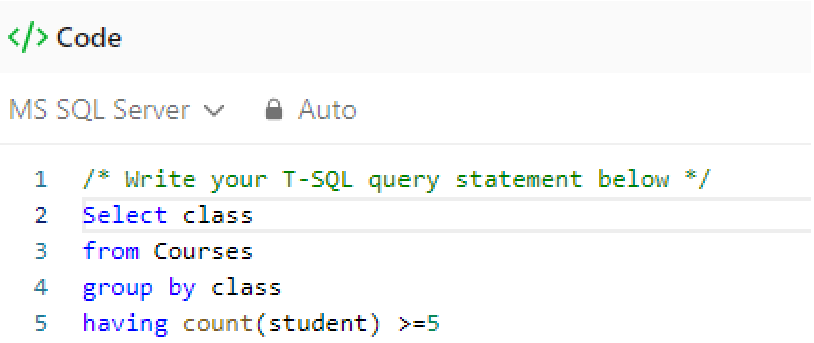
Table: `Courses`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| student     | varchar |
| class       | varchar |
+-------------+---------+
(student, class) is the primary key (combination of columns with
unique values) for this table.
Each row of this table indicates the name of a student and the class
in which they are enrolled.
```

Write a solution to find all the classes that have **at least five students**.

Return the result table in **any order**.

Answer



```sql
/* Write your T-SQL query statement below */
Select class
from Courses
group by class
having count(student) >=5
```

Q: 27

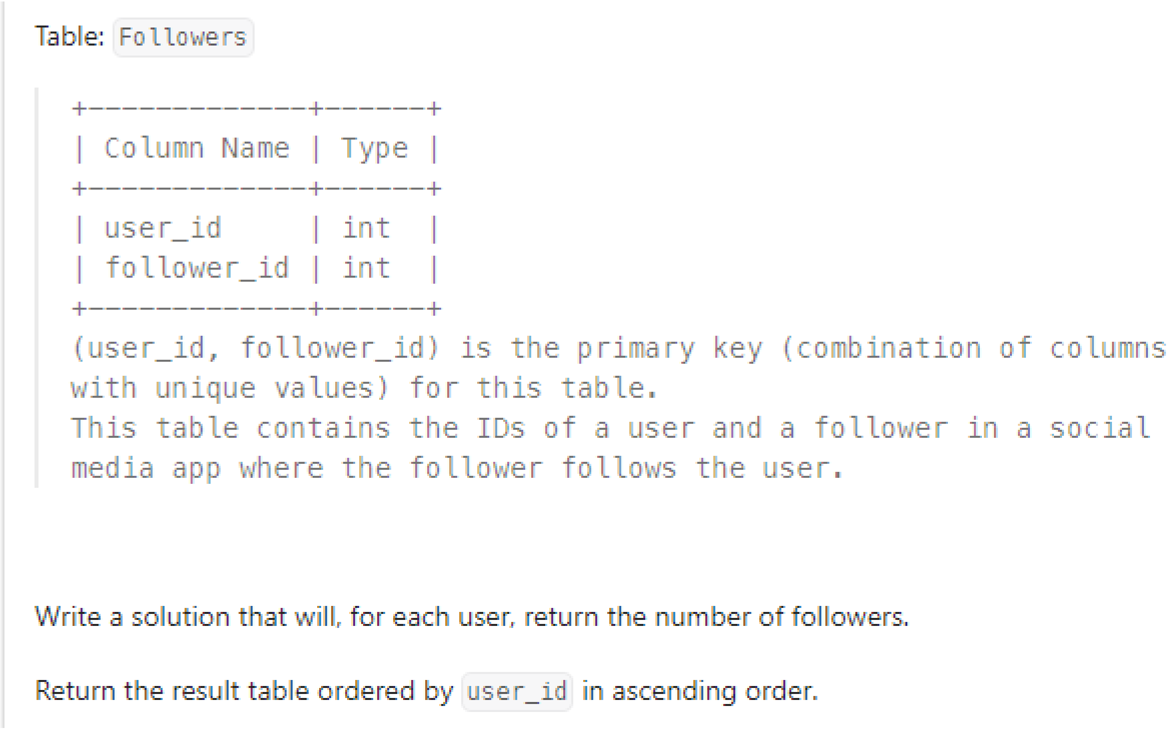


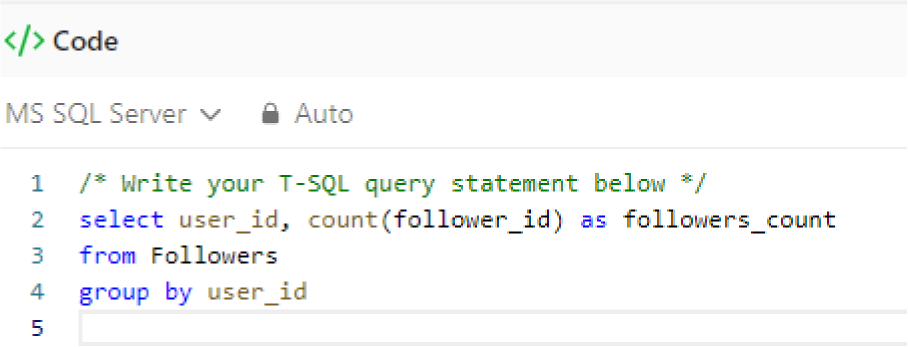
Table: `Followers`

```
+-------------+------+
| Column Name | Type |
+-------------+------+
| user_id     | int  |
| follower_id | int  |
+-------------+------+
(user_id, follower_id) is the primary key (combination of columns
with unique values) for this table.
This table contains the IDs of a user and a follower in a social
media app where the follower follows the user.
```

Write a solution that will, for each user, return the number of followers.

Return the result table ordered by `user_id` in ascending order.

Answer



```sql
/* Write your T-SQL query statement below */
select user_id, count(follower_id) as followers_count
from Followers
group by user_id

```

Q: 28

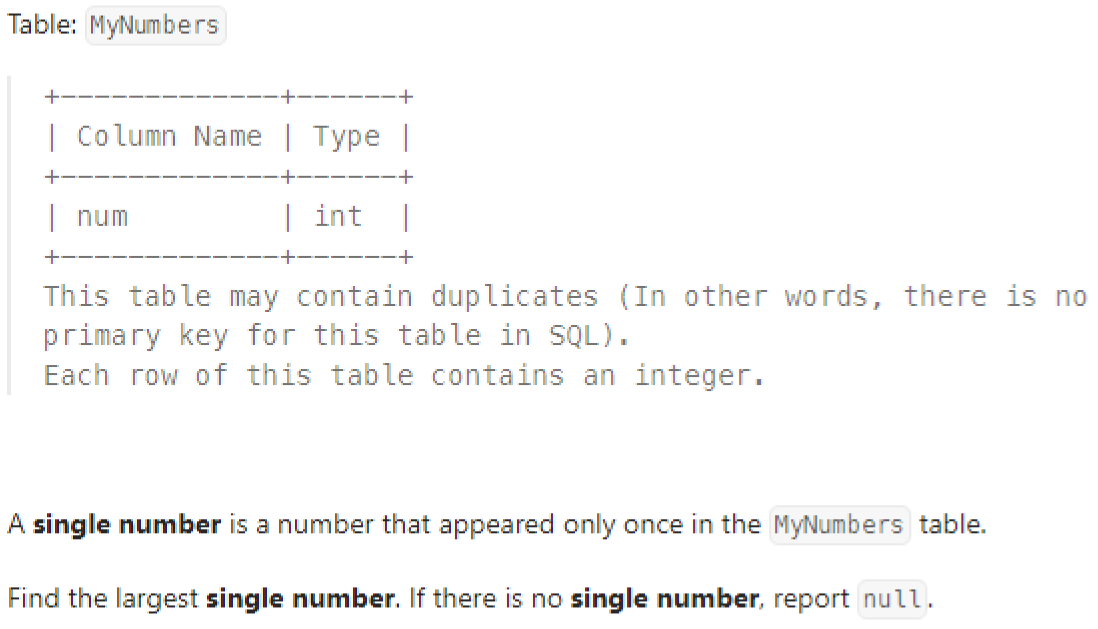


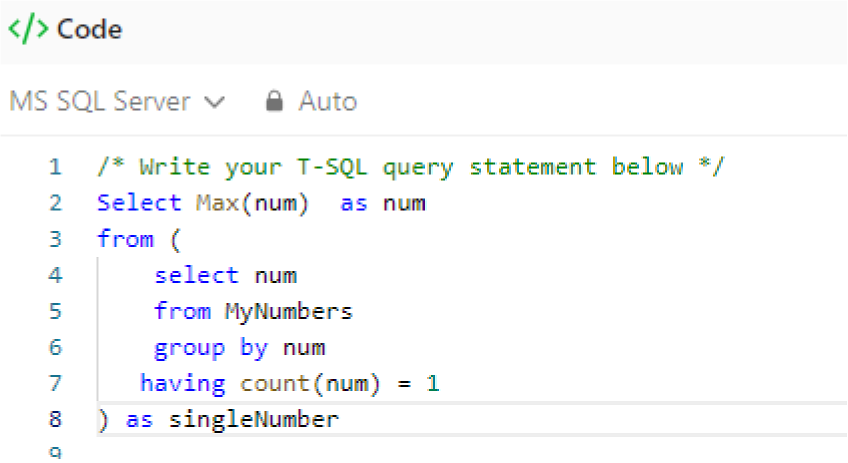
Table: `MyNumbers`

```
+-------------+------+
| Column Name | Type |
+-------------+------+
| num         | int  |
+-------------+------+
This table may contain duplicates (In other words, there is no
primary key for this table in SQL).
Each row of this table contains an integer.
```

A **single number** is a number that appeared only once in the `MyNumbers` table.

Find the largest **single number**. If there is no **single number**, report `null`.

Answer



```sql
/* Write your T-SQL query statement below */
Select Max(num)  as num
from (
    select num
    from MyNumbers
    group by num
   having count(num) = 1
) as singleNumber
```

Q: 29

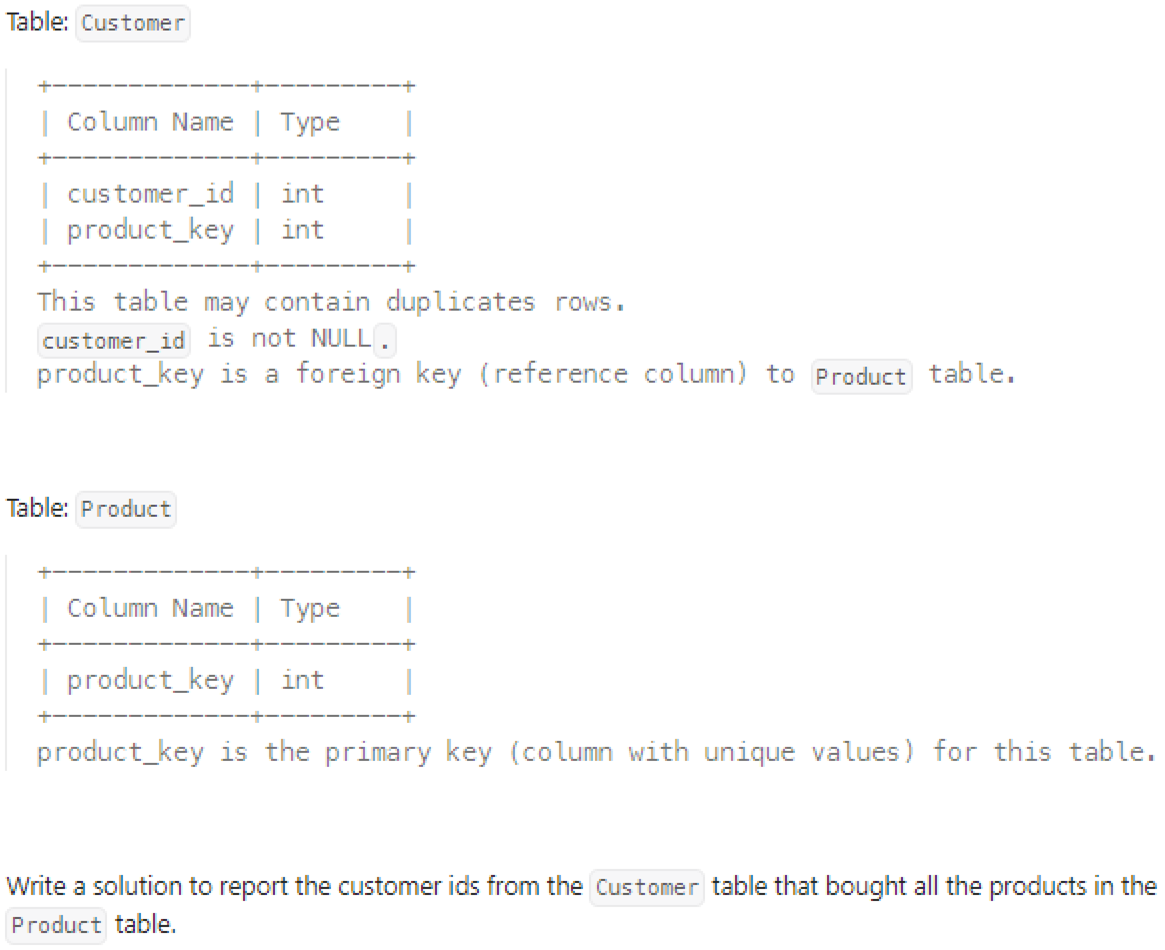


Table: `Customer`

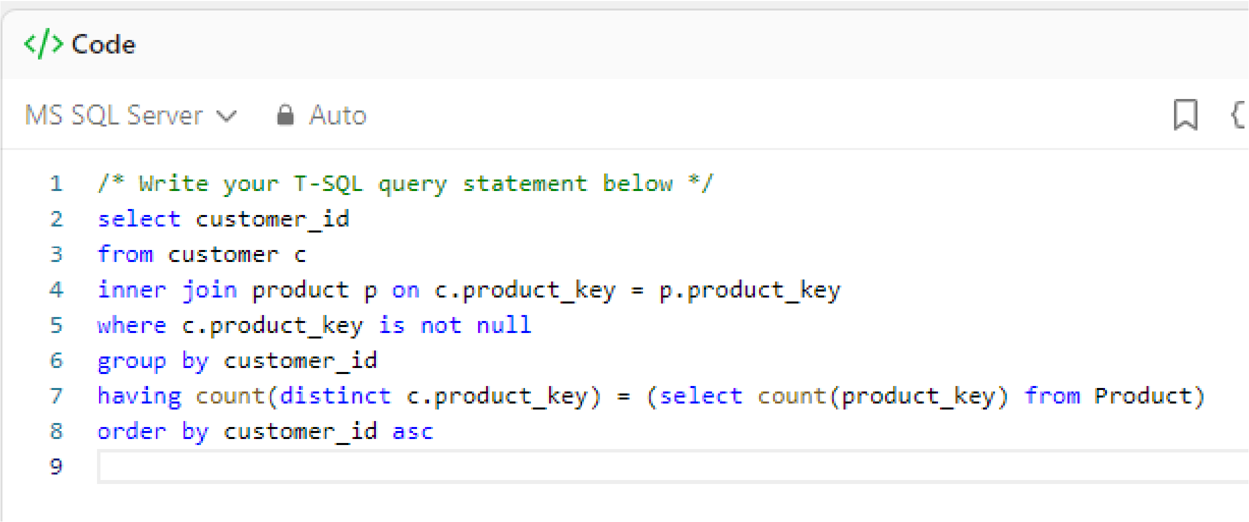
```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| customer_id | int     |
| product_key | int     |
+-------------+---------+
This table may contain duplicates rows.
customer_id is not NULL.
product_key is a foreign key (reference column) to Product table.
```

Table: `Product`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| product_key | int     |
+-------------+---------+
product_key is the primary key (column with unique values) for this table.
```

Write a solution to report the customer ids from the `Customer` table that bought all the products in the `Product` table.

Answer



```sql
/* Write your T-SQL query statement below */
select customer_id
from customer c
inner join product p on c.product_key = p.product_key
where c.product_key is not null
group by customer_id
having count(distinct c.product_key) = (select count(product_key) from Product)
order by customer_id asc

```

❖ Advanced String Functions / Regex / Clause

Q: 30

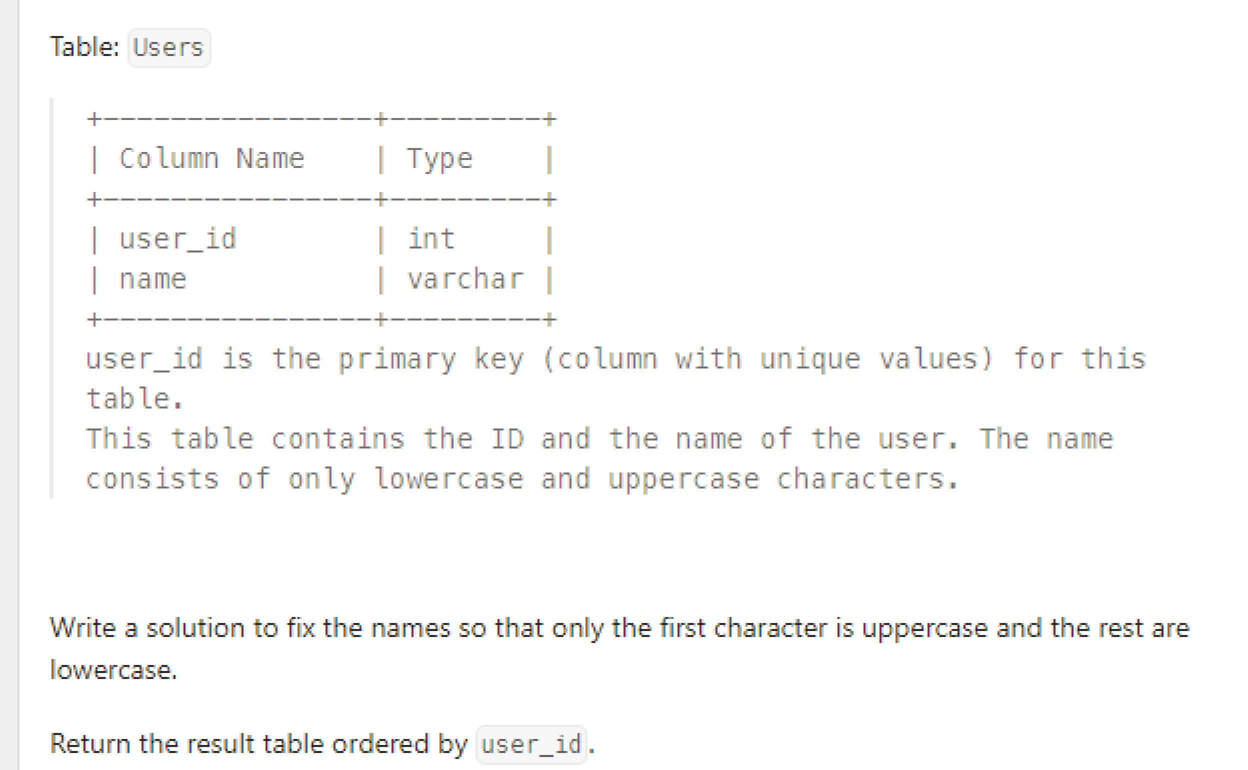


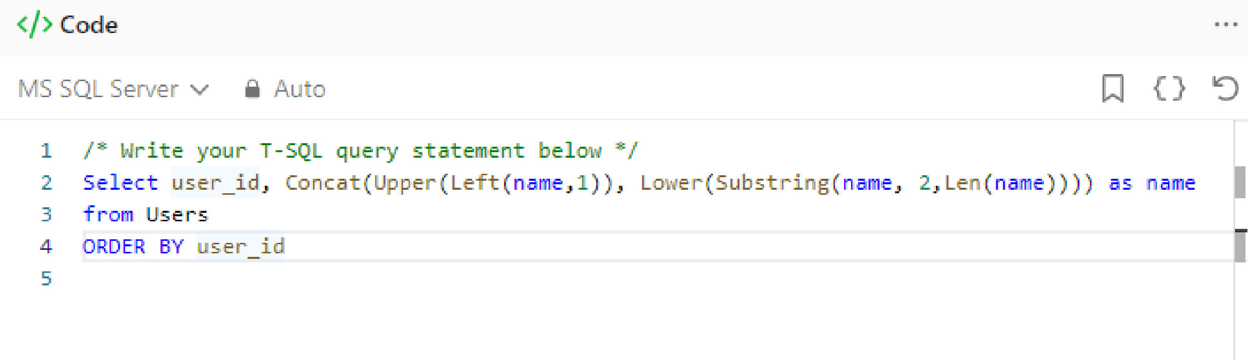
Table: `Users`

```
+-----------------+---------+
| Column Name     | Type    |
+-----------------+---------+
| user_id         | int     |
| name            | varchar |
+-----------------+---------+
user_id is the primary key (column with unique values) for this
table.
This table contains the ID and the name of the user. The name
consists of only lowercase and uppercase characters.
```

Write a solution to fix the names so that only the first character is uppercase and the rest are lowercase.

Return the result table ordered by `user_id`.

Answer



```
/* Write your T-SQL query statement below */
Select user_id, Concat(Upper(Left(name,1)), Lower(Substring(name, 2,Len(name)))) as name
from Users
ORDER BY user_id

```

Q: 31

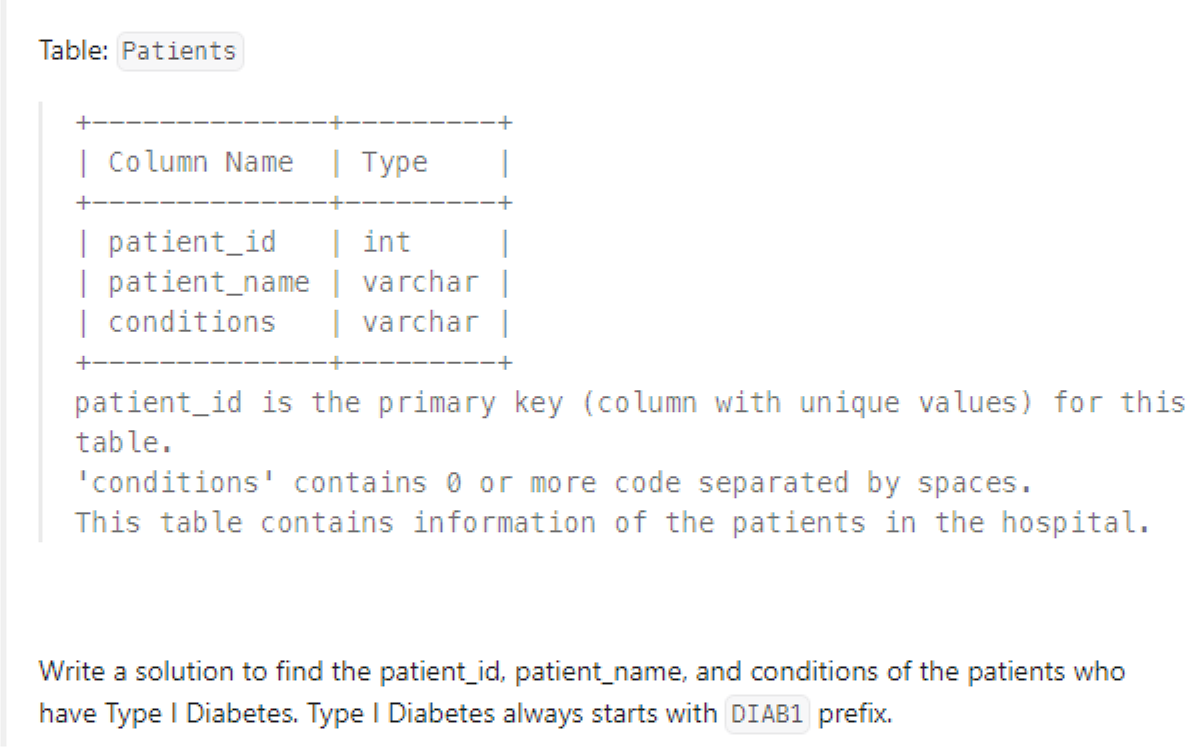
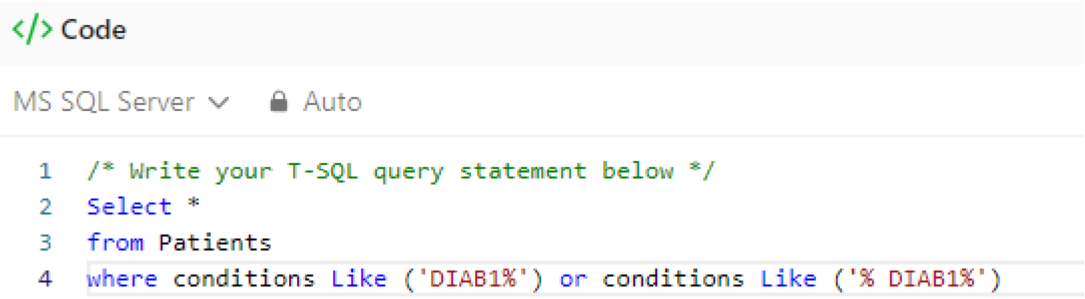


Table: `Patients`

```
+--------------+---------+
| Column Name  | Type    |
+--------------+---------+
| patient_id   | int     |
| patient_name | varchar |
| conditions   | varchar |
+--------------+---------+
patient_id is the primary key (column with unique values) for this
table.
'conditions' contains 0 or more code separated by spaces.
This table contains information of the patients in the hospital.
```

Write a solution to find the patient_id, patient_name, and conditions of the patients who have Type I Diabetes. Type I Diabetes always starts with `DIAB1` prefix.

Answer



```sql
/* Write your T-SQL query statement below */
Select *
from Patients
where conditions Like ('DIAB1%') or conditions Like ('% DIAB1%')
```

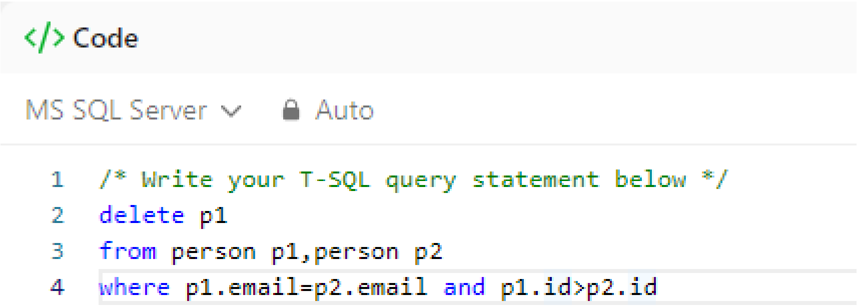
Q: 32

Table: `Person`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| id          | int     |
| email       | varchar |
+-------------+---------+
id is the primary key (column with unique values) for this table.
Each row of this table contains an email. The emails will not
contain uppercase letters.
```

Write a solution to **delete** all duplicate emails, keeping only one unique email with the smallest `id`.

Answer



```sql
/* Write your T-SQL query statement below */
delete p1
from person p1,person p2
where p1.email=p2.email and p1.id>p2.id
```

Q: 33

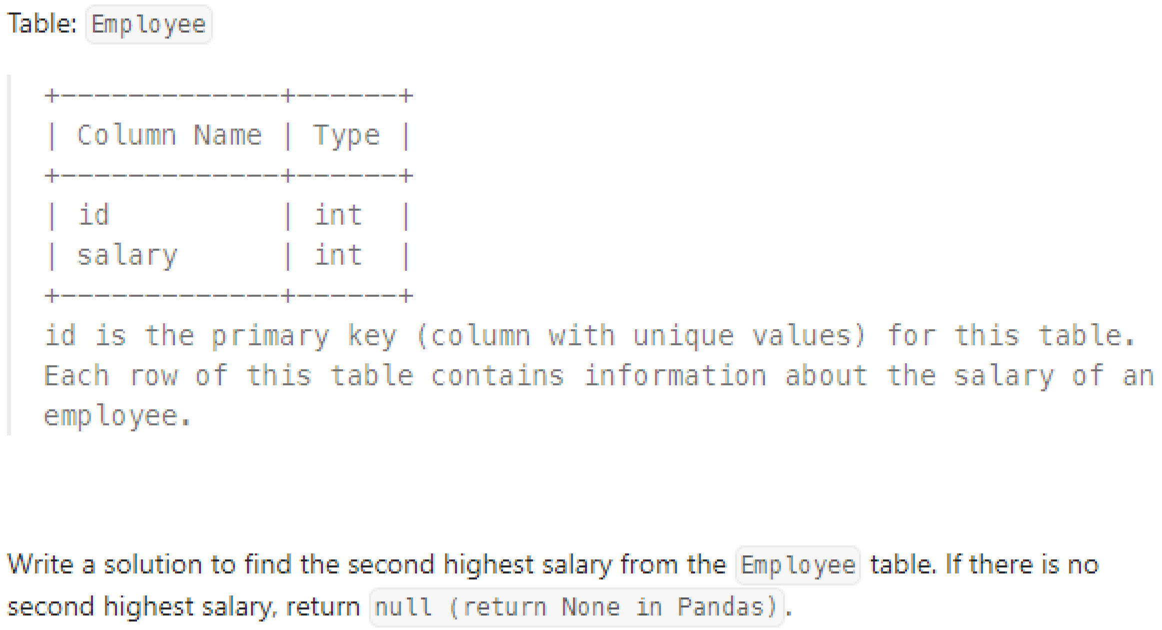
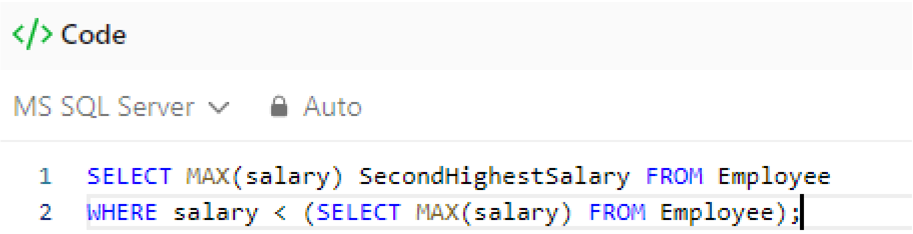


Table: `Employee`

```
+-------------+------+
| Column Name | Type |
+-------------+------+
| id          | int  |
| salary      | int  |
+-------------+------+
id is the primary key (column with unique values) for this table.
Each row of this table contains information about the salary of an
employee.
```

Write a solution to find the second highest salary from the `Employee` table. If there is no second highest salary, return `null (return None in Pandas)`.

Answer



```sql
SELECT MAX(salary) SecondHighestSalary FROM Employee
WHERE salary < (SELECT MAX(salary) FROM Employee);
```

Q: 34

Table `Activities`:

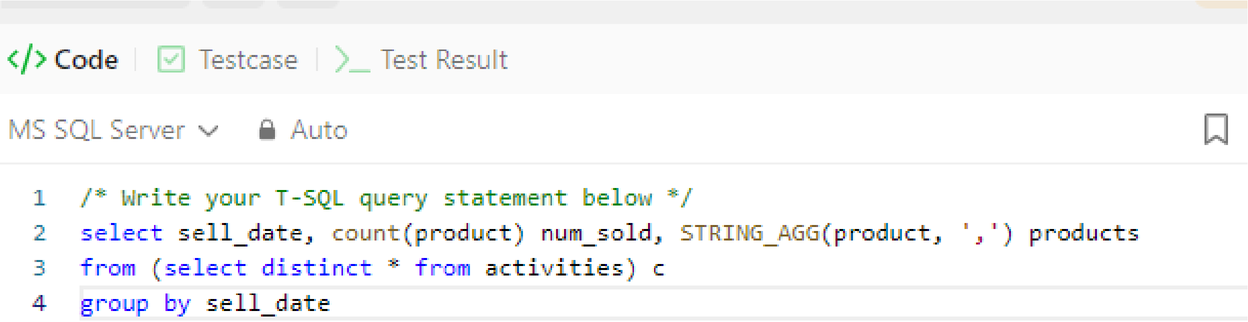
```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| sell_date   | date    |
| product     | varchar |
+-------------+---------+
There is no primary key (column with unique values) for this
table. It may contain duplicates.
Each row of this table contains the product name and the date it
was sold in a market.
```

Write a solution to find for each date the number of different products sold and their names.

The sold products names for each date should be sorted lexicographically.

Return the result table ordered by `sell_date`.

Answer



```sql
/* Write your T-SQL query statement below */
select sell_date, count(product) num_sold, STRING_AGG(product, ',') products
from (select distinct * from activities) c
group by sell_date
```

Q: 35

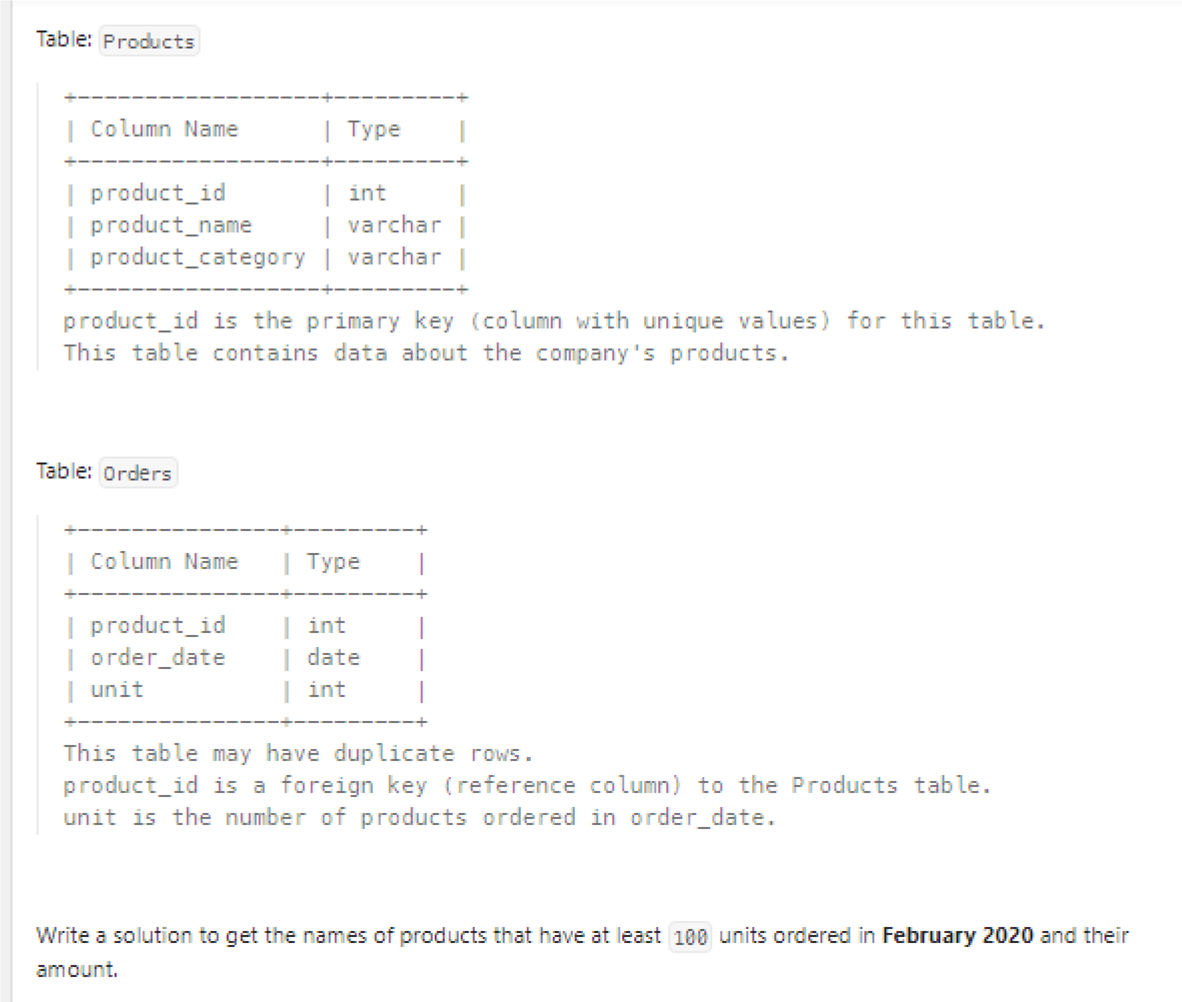


Table: `Products`

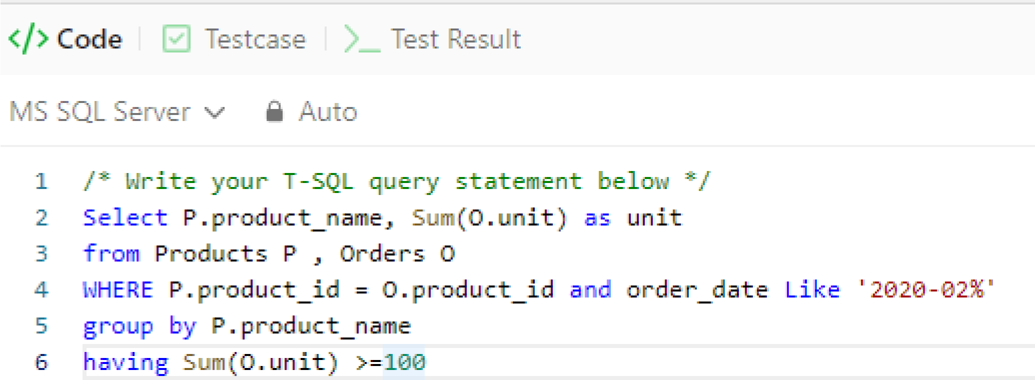
```
+------------------+---------+
| Column Name      | Type    |
+------------------+---------+
| product_id       | int     |
| product_name     | varchar |
| product_category | varchar |
+------------------+---------+
product_id is the primary key (column with unique values) for this table.
This table contains data about the company's products.
```

Table: `Orders`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| product_id    | int     |
| order_date    | date    |
| unit          | int     |
+---------------+---------+
This table may have duplicate rows.
product_id is a foreign key (reference column) to the Products table.
unit is the number of products ordered in order_date.
```

Write a solution to get the names of products that have at least `100` units ordered in **February 2020** and their amount.

Answer



```sql
/* Write your T-SQL query statement below */
Select P.product_name, Sum(O.unit) as unit
from Products P , Orders O
WHERE P.product_id = O.product_id and order_date Like '2020-02%'
group by P.product_name
having Sum(O.unit) >=100
```

Q: 36

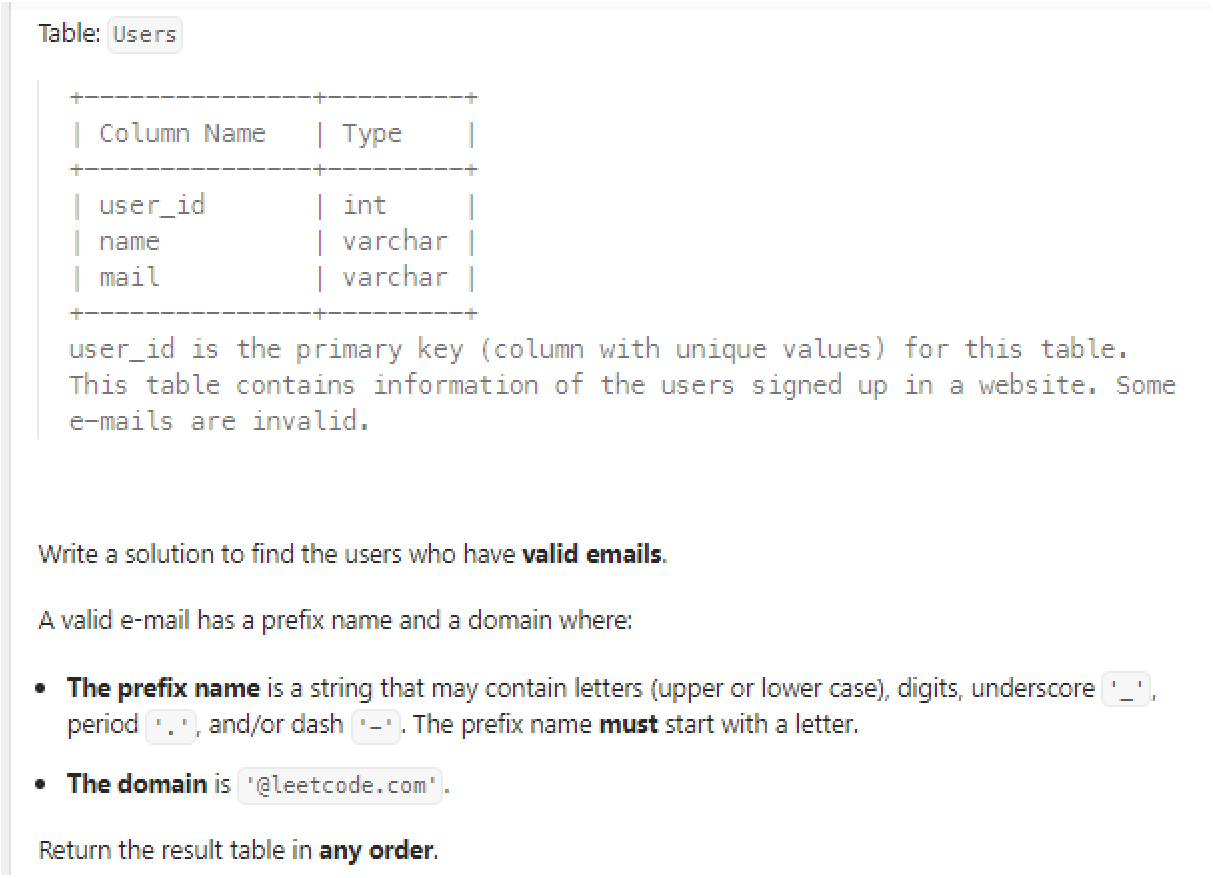


Table: `Users`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| user_id       | int     |
| name          | varchar |
| mail          | varchar |
+---------------+---------+
user_id is the primary key (column with unique values) for this table.
This table contains information of the users signed up in a website. Some
e-mails are invalid.
```

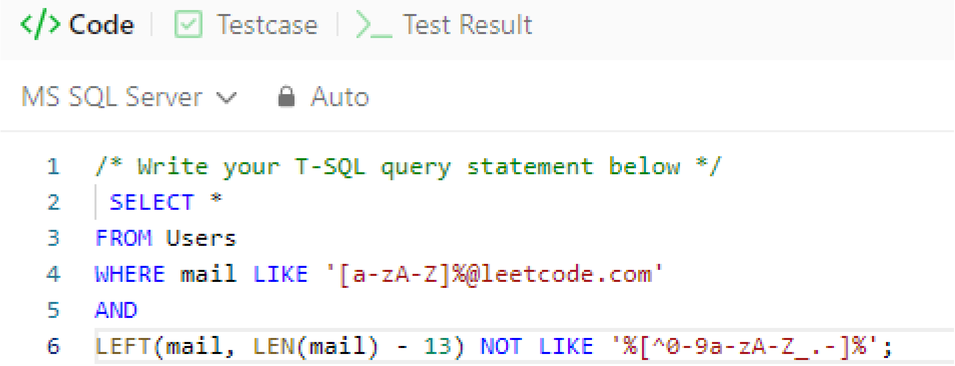
Write a solution to find the users who have **valid emails**.

A valid e-mail has a prefix name and a domain where:

- **The prefix name** is a string that may contain letters (upper or lower case), digits, underscore `'_'`, period `'.'`, and/or dash `'-'`. The prefix name **must** start with a letter.
- **The domain** is `'@leetcode.com'`.

Return the result table in **any order**.

Answer



```
/* Write your T-SQL query statement below */
 SELECT *
FROM Users
WHERE mail LIKE '[a-zA-Z]%@leetcode.com'
AND
LEFT(mail, LEN(mail) - 13) NOT LIKE '%[^0-9a-zA-Z_.-]%';
```

❖ Sub Queries

Q: 37

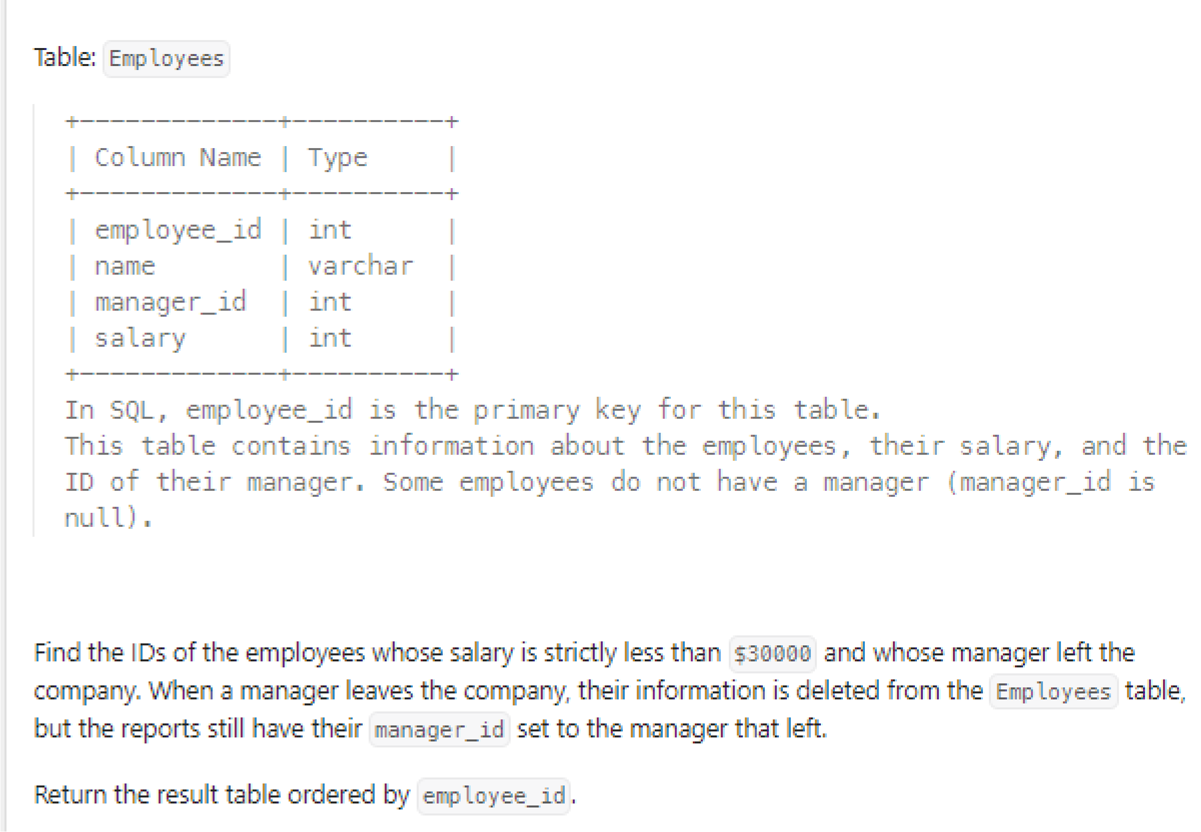


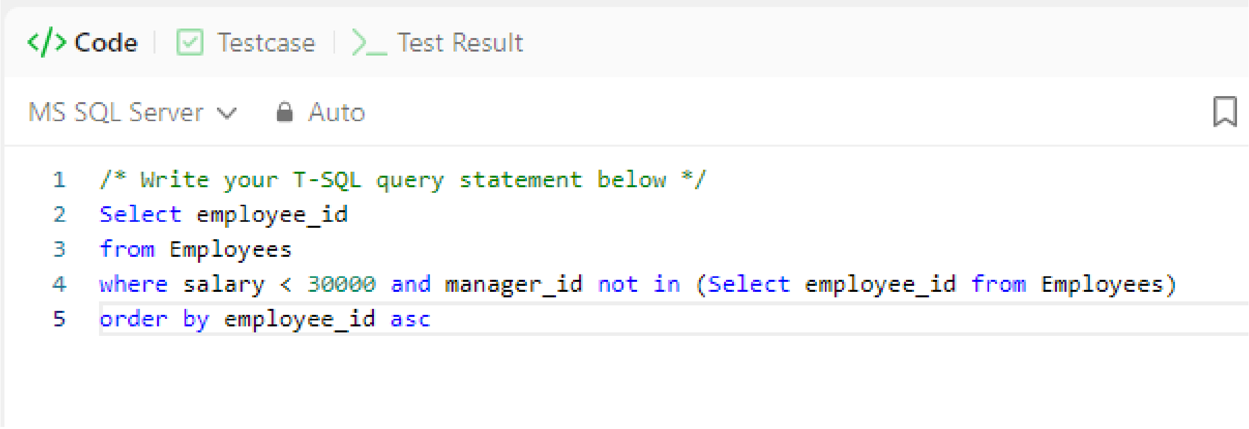
Table: `Employees`

```
+-------------+----------+
| Column Name | Type     |
+-------------+----------+
| employee_id | int      |
| name        | varchar  |
| manager_id  | int      |
| salary      | int      |
+-------------+----------+
In SQL, employee_id is the primary key for this table.
This table contains information about the employees, their salary, and the
ID of their manager. Some employees do not have a manager (manager_id is
null).
```

Find the IDs of the employees whose salary is strictly less than `$30000` and whose manager left the company. When a manager leaves the company, their information is deleted from the `Employees` table, but the reports still have their `manager_id` set to the manager that left.

Return the result table ordered by `employee_id`.

Answer



```sql
/* Write your T-SQL query statement below */
Select employee_id
from Employees
where salary < 30000 and manager_id not in (Select employee_id from Employees)
order by employee_id asc
```

Q: 38

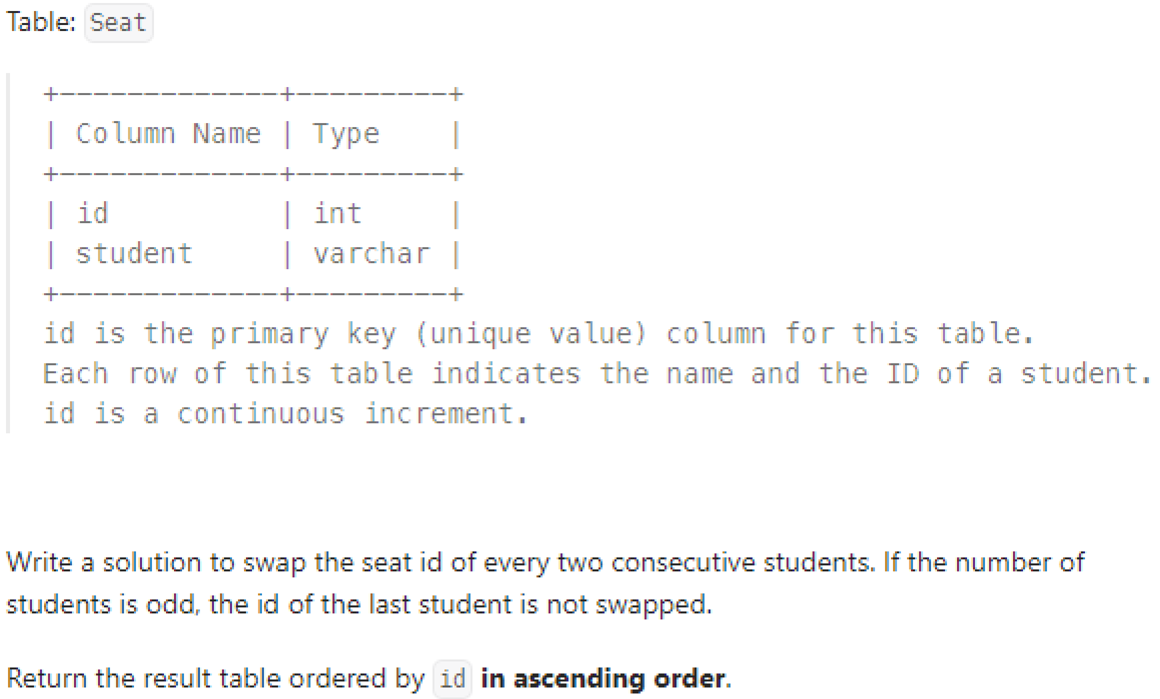


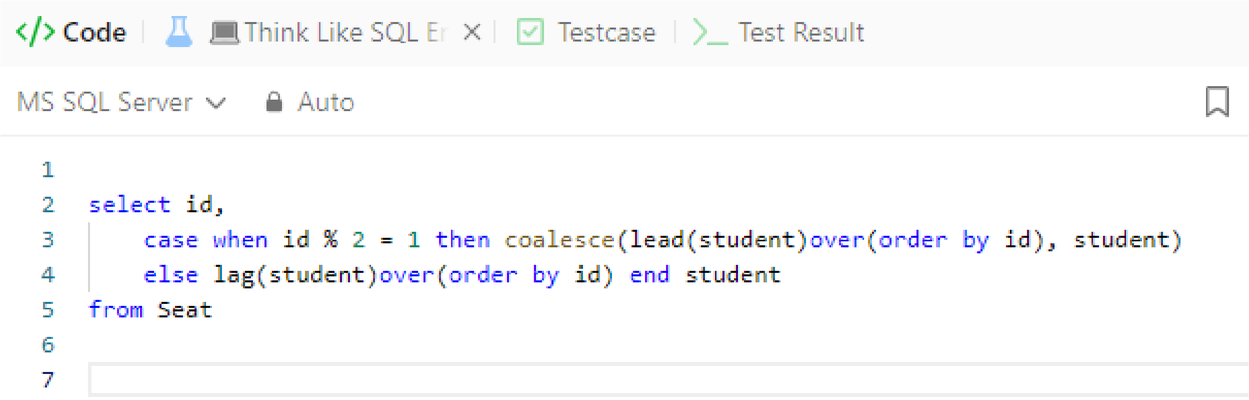
Table: Seat

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| id          | int     |
| student     | varchar |
+-------------+---------+
id is the primary key (unique value) column for this table.
Each row of this table indicates the name and the ID of a student.
id is a continuous increment.
```

Write a solution to swap the seat id of every two consecutive students. If the number of students is odd, the id of the last student is not swapped.

Return the result table ordered by `id` **in ascending order**.

Answer



```

select id,
    case when id % 2 = 1 then coalesce(lead(student)over(order by id), student)
    else lag(student)over(order by id) end student
from Seat


```

Q: 39

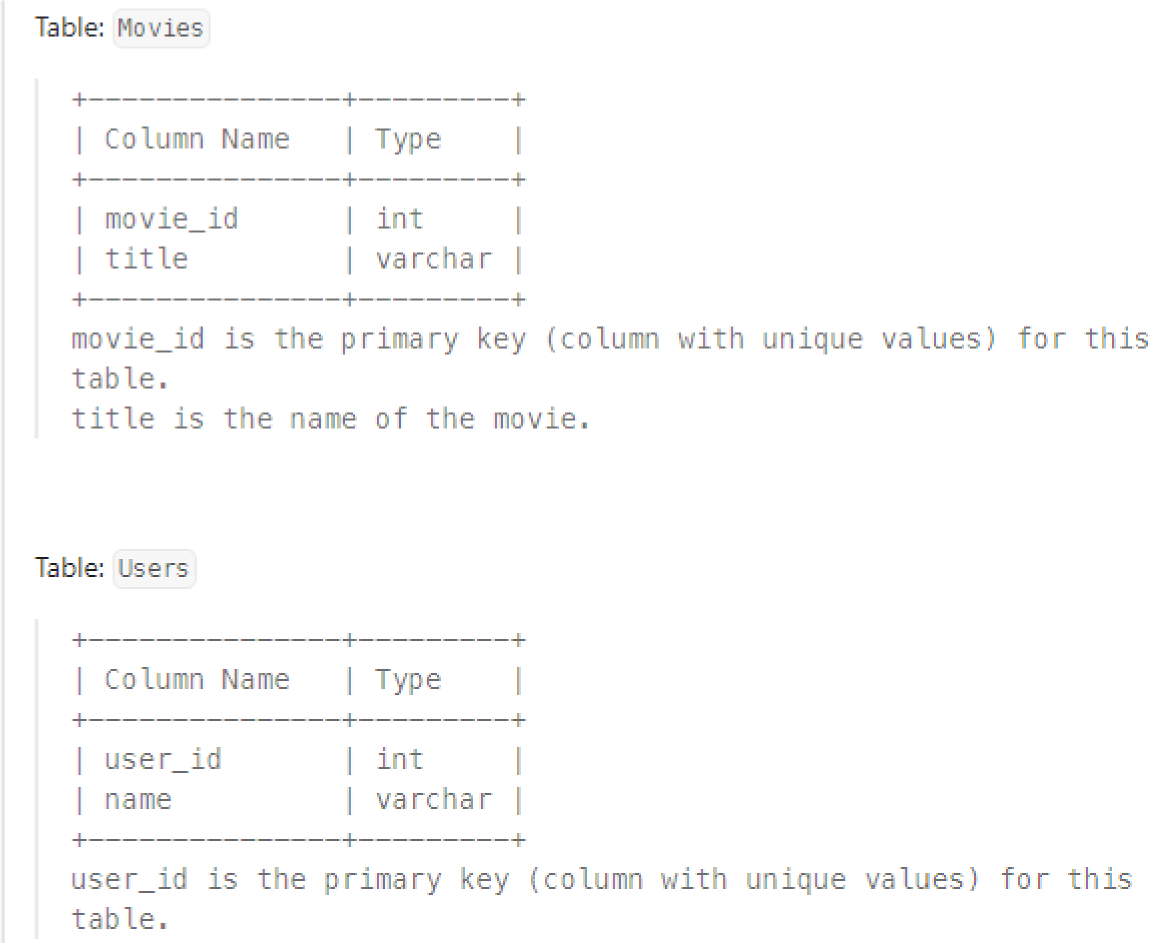


Table: Movies

```
+---------------+----------+
| Column Name   | Type     |
+---------------+----------+
| movie_id      | int      |
| title         | varchar  |
+---------------+----------+
movie_id is the primary key (column with unique values) for this
table.
title is the name of the movie.
```

Table: Users

```
+---------------+----------+
| Column Name   | Type     |
+---------------+----------+
| user_id       | int      |
| name          | varchar  |
+---------------+----------+
user_id is the primary key (column with unique values) for this
table.
```

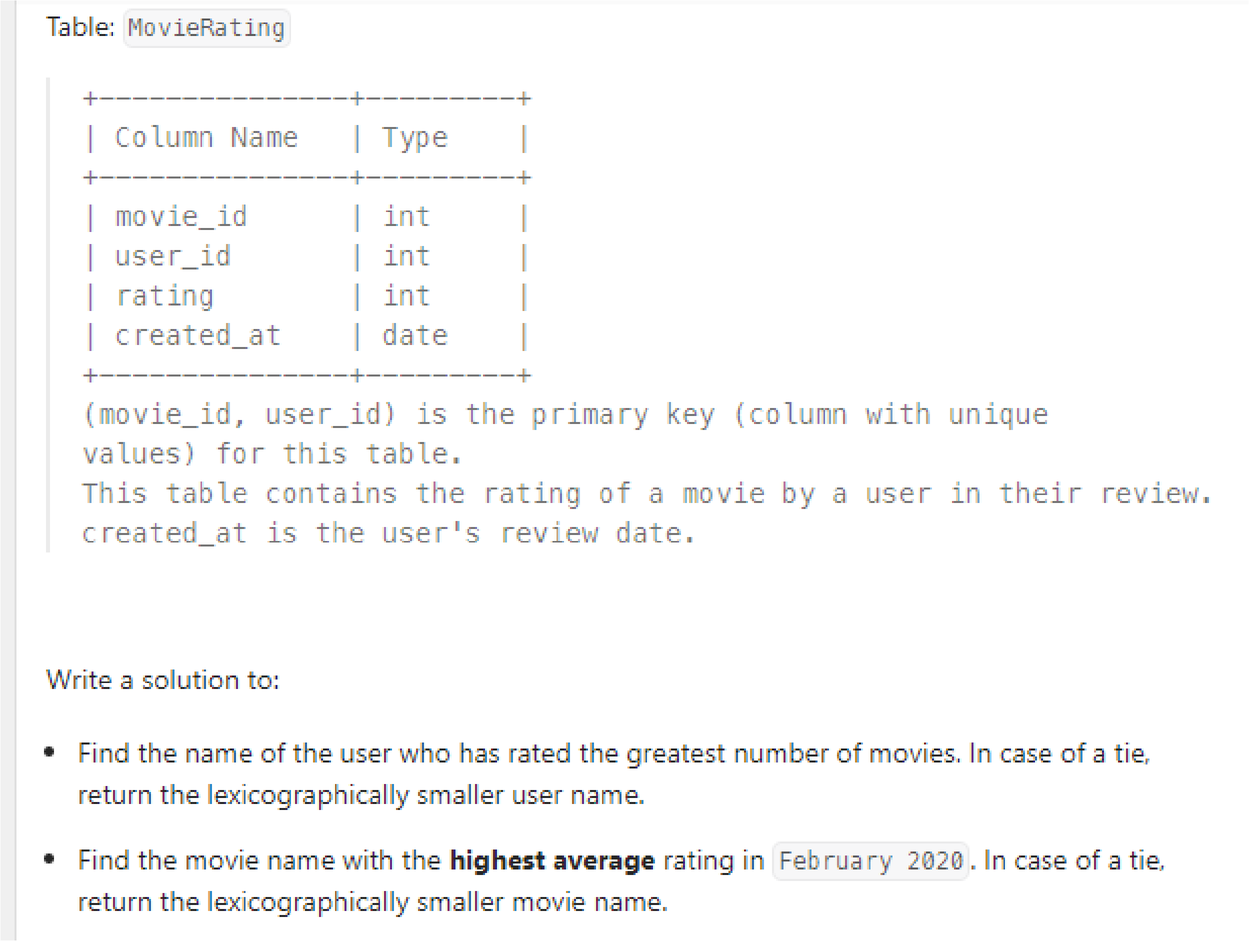


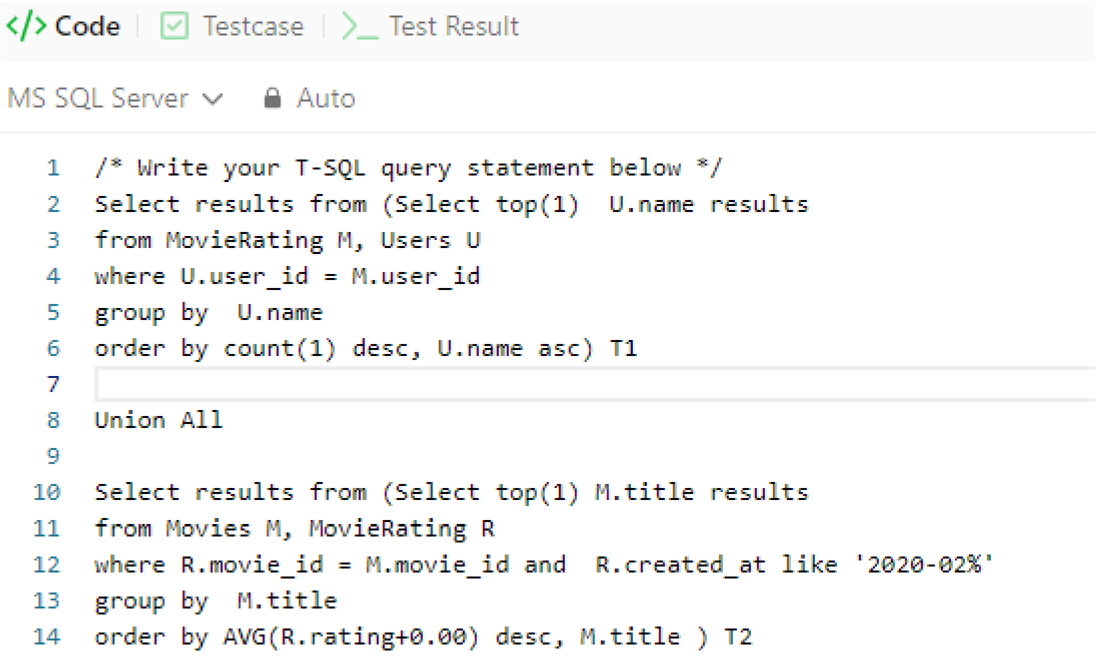
Table: `MovieRating`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| movie_id      | int     |
| user_id       | int     |
| rating        | int     |
| created_at    | date    |
+---------------+---------+
(movie_id, user_id) is the primary key (column with unique
values) for this table.
This table contains the rating of a movie by a user in their review.
created_at is the user's review date.
```

Write a solution to:

- Find the name of the user who has rated the greatest number of movies. In case of a tie, return the lexicographically smaller user name.
- Find the movie name with the **highest average** rating in `February 2020`. In case of a tie, return the lexicographically smaller movie name.

Answer



```sql
/* Write your T-SQL query statement below */
Select results from (Select top(1)  U.name results
from MovieRating M, Users U
where U.user_id = M.user_id
group by  U.name
order by count(1) desc, U.name asc) T1

Union All

Select results from (Select top(1) M.title results
from Movies M, MovieRating R
where R.movie_id = M.movie_id and  R.created_at like '2020-02%'
group by  M.title
order by AVG(R.rating+0.00) desc, M.title ) T2
```

Q: 40

Table: `Customer`

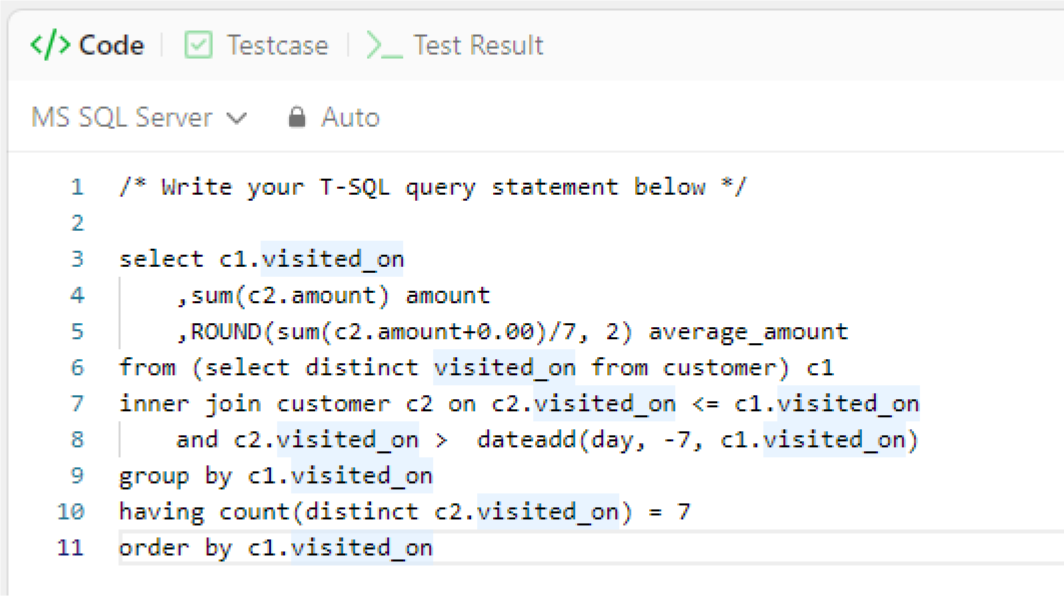
```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| customer_id   | int     |
| name          | varchar |
| visited_on    | date    |
| amount        | int     |
+---------------+---------+
In SQL,(customer_id, visited_on) is the primary key for this table.
This table contains data about customer transactions in a restaurant.
visited_on is the date on which the customer with ID (customer_id) has visited
the restaurant.
amount is the total paid by a customer.
```

You are the restaurant owner and you want to analyze a possible expansion (there will be at least one customer every day).

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**.

Return the result table ordered by `visited_on` **in ascending order**.

Answer



```sql
/* Write your T-SQL query statement below */

select c1.visited_on
    ,sum(c2.amount) amount
    ,ROUND(sum(c2.amount+0.00)/7, 2) average_amount
from (select distinct visited_on from customer) c1
inner join customer c2 on c2.visited_on <= c1.visited_on
    and c2.visited_on >  dateadd(day, -7, c1.visited_on)
group by c1.visited_on
having count(distinct c2.visited_on) = 7
order by c1.visited_on
```

Q: 41

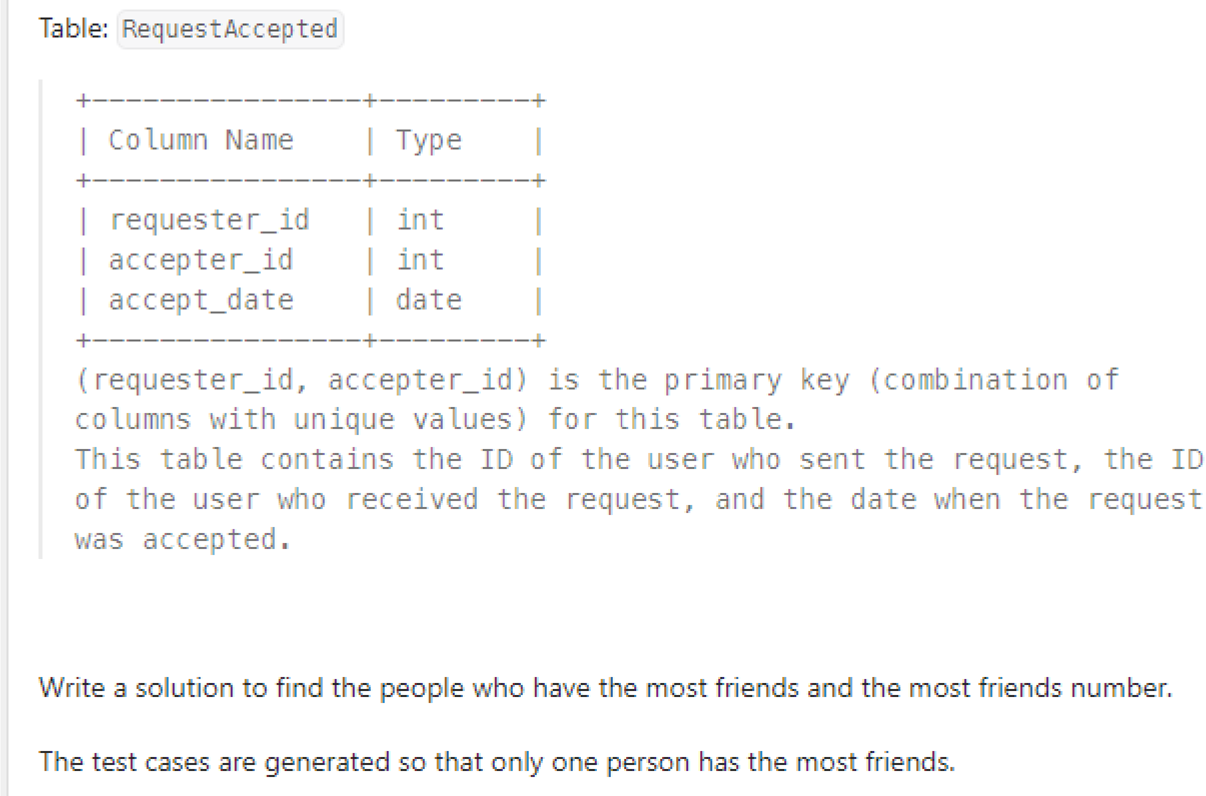


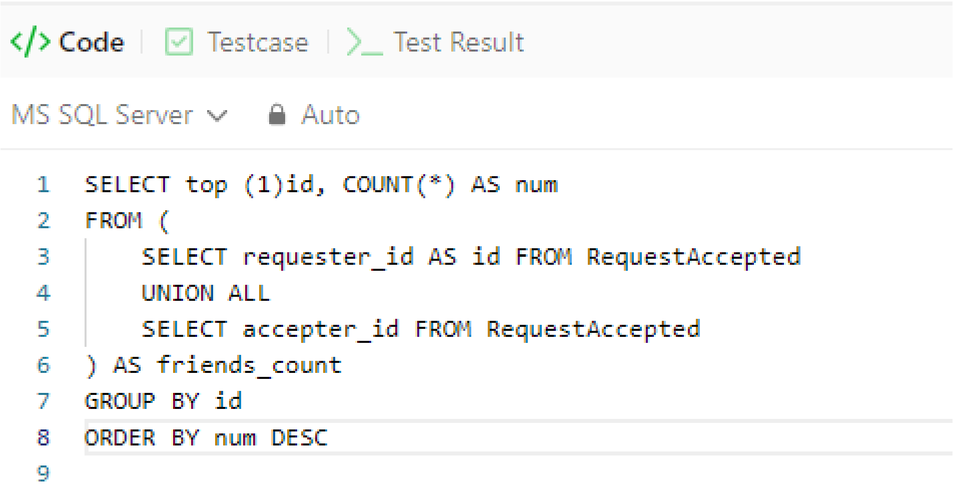
Table: RequestAccepted

```
+----------------+---------+
| Column Name    | Type    |
+----------------+---------+
| requester_id   | int     |
| accepter_id    | int     |
| accept_date    | date    |
+----------------+---------+
(requester_id, accepter_id) is the primary key (combination of
columns with unique values) for this table.
This table contains the ID of the user who sent the request, the ID
of the user who received the request, and the date when the request
was accepted.
```

Write a solution to find the people who have the most friends and the most friends number.

The test cases are generated so that only one person has the most friends.

Answer



```sql
SELECT top (1)id, COUNT(*) AS num
FROM (
    SELECT requester_id AS id FROM RequestAccepted
    UNION ALL
    SELECT accepter_id FROM RequestAccepted
) AS friends_count
GROUP BY id
ORDER BY num DESC
```

Q: 42

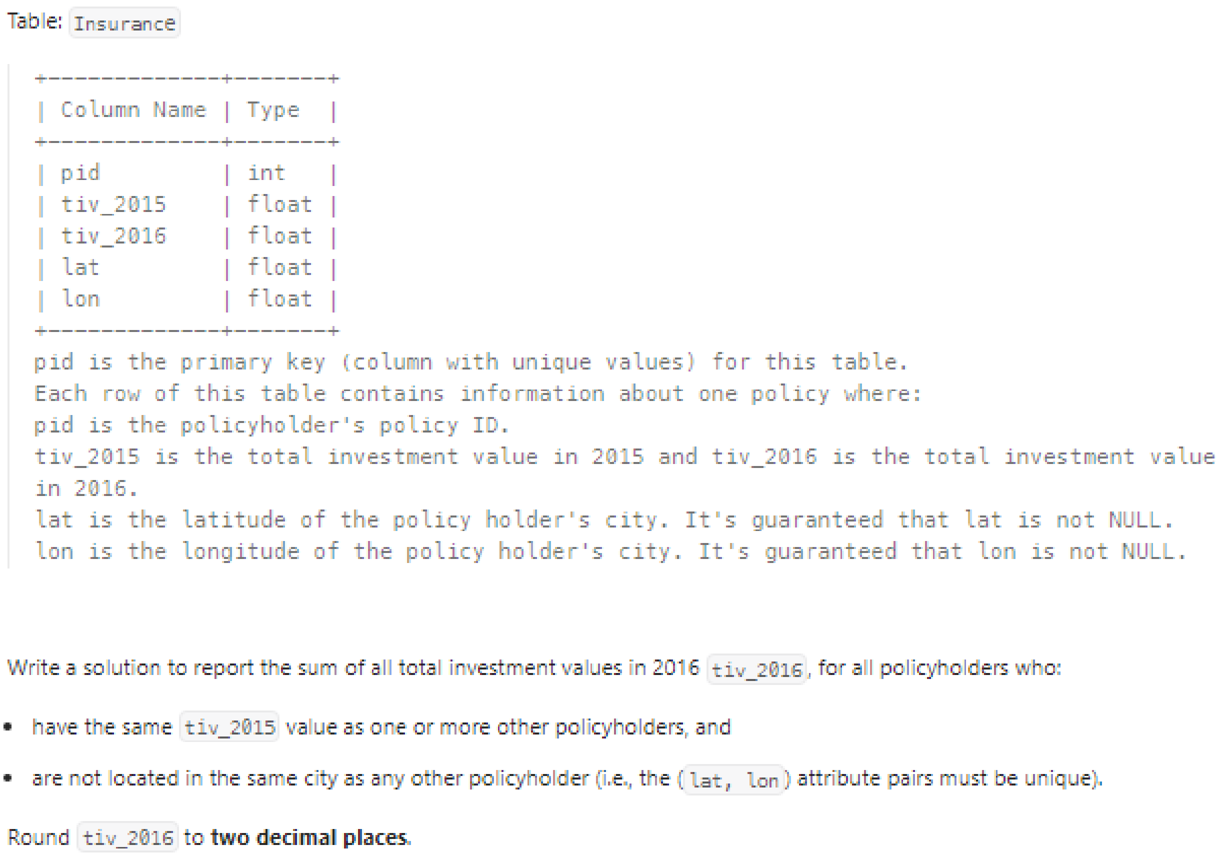


Table: `Insurance`

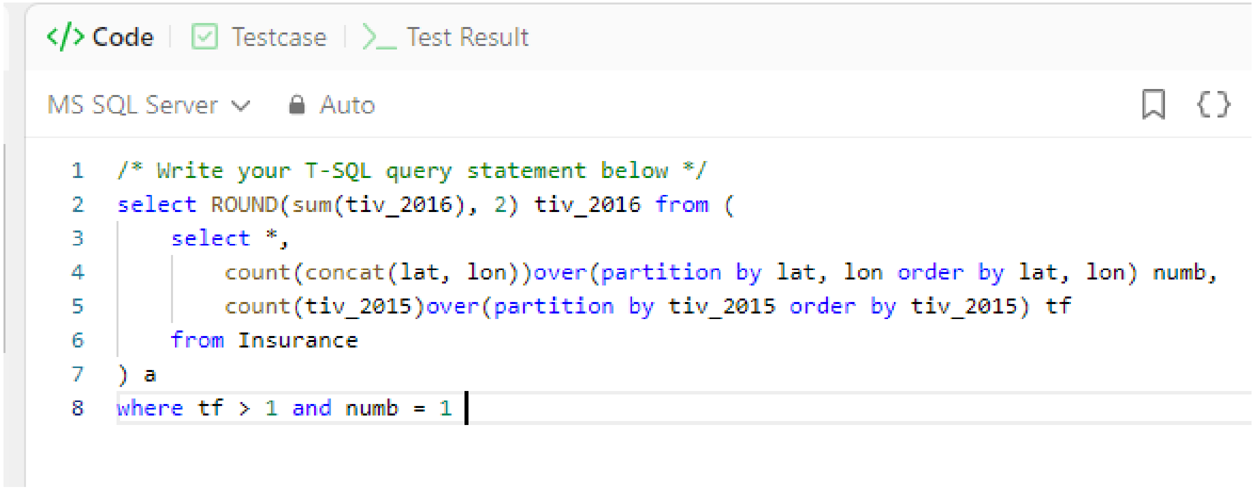
```
+-------------+-------+
| Column Name | Type  |
+-------------+-------+
| pid         | int   |
| tiv_2015    | float |
| tiv_2016    | float |
| lat         | float |
| lon         | float |
+-------------+-------+
pid is the primary key (column with unique values) for this table.
Each row of this table contains information about one policy where:
pid is the policyholder's policy ID.
tiv_2015 is the total investment value in 2015 and tiv_2016 is the total investment value
in 2016.
lat is the latitude of the policy holder's city. It's guaranteed that lat is not NULL.
lon is the longitude of the policy holder's city. It's guaranteed that lon is not NULL.
```

Write a solution to report the sum of all total investment values in 2016 `tiv_2016`, for all policyholders who:

- have the same `tiv_2015` value as one or more other policyholders, and
- are not located in the same city as any other policyholder (i.e., the (`lat, lon`) attribute pairs must be unique).

Round `tiv_2016` to **two decimal places**.

Answer



```sql
/* Write your T-SQL query statement below */
select ROUND(sum(tiv_2016), 2) tiv_2016 from (
    select *,
        count(concat(lat, lon))over(partition by lat, lon order by lat, lon) numb,
        count(tiv_2015)over(partition by tiv_2015 order by tiv_2015) tf
    from Insurance
) a
where tf > 1 and numb = 1
```

Q: 43

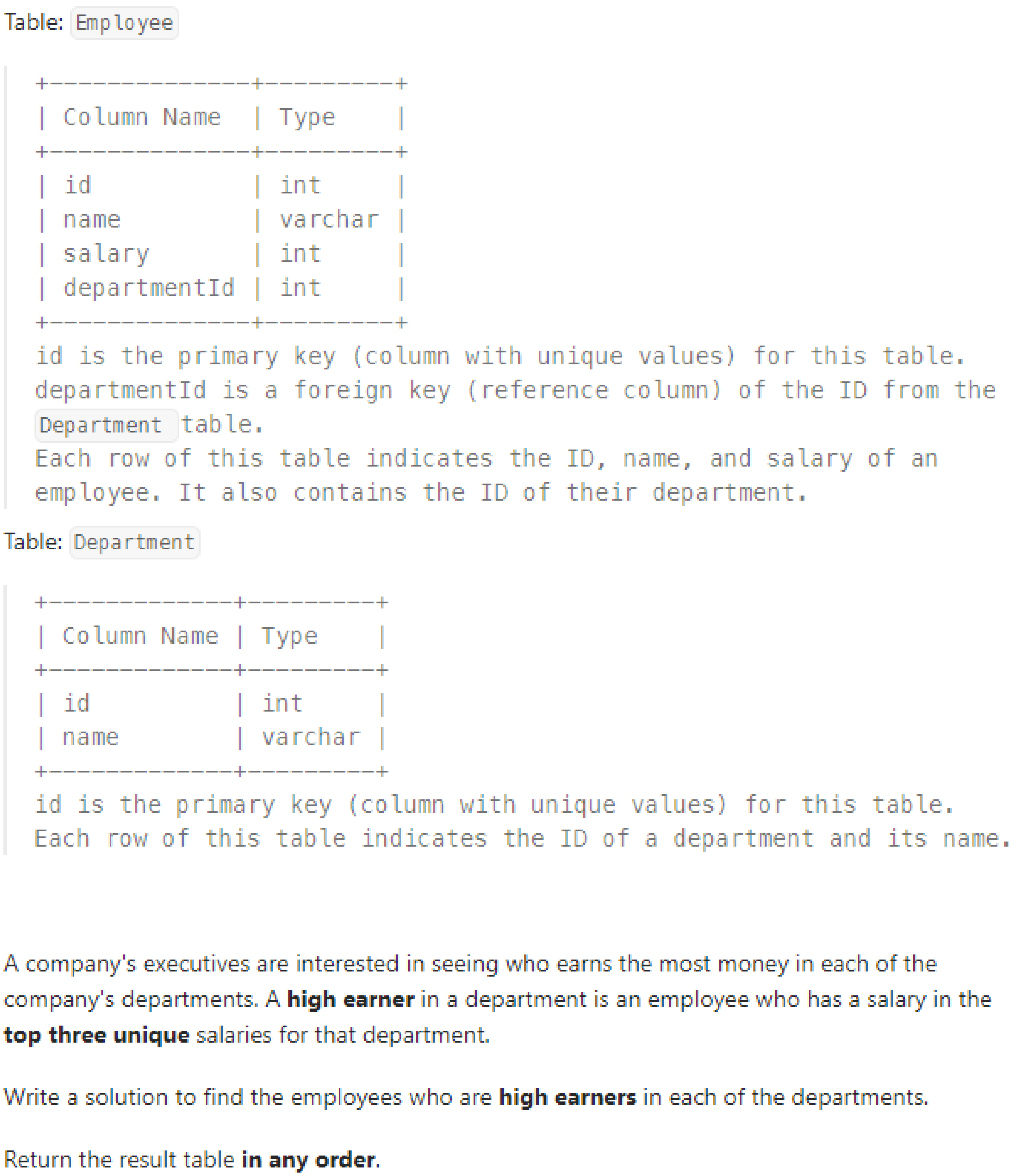


Table: `Employee`

```
+--------------+---------+
| Column Name  | Type    |
+--------------+---------+
| id           | int     |
| name         | varchar |
| salary       | int     |
| departmentId | int     |
+--------------+---------+
id is the primary key (column with unique values) for this table.
departmentId is a foreign key (reference column) of the ID from the
Department table.
Each row of this table indicates the ID, name, and salary of an
employee. It also contains the ID of their department.
```

Table: `Department`

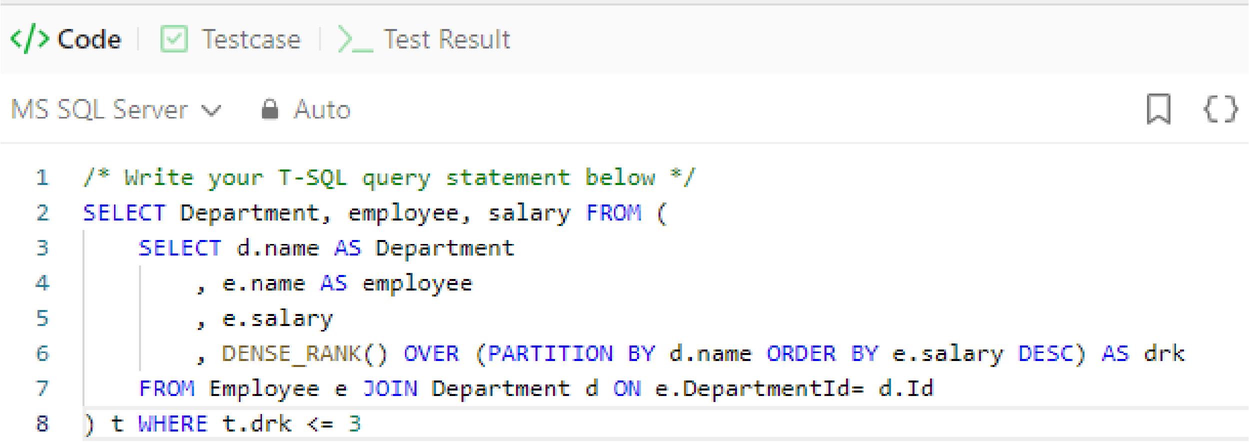
```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| id          | int     |
| name        | varchar |
+-------------+---------+
id is the primary key (column with unique values) for this table.
Each row of this table indicates the ID of a department and its name.
```

A company's executives are interested in seeing who earns the most money in each of the company's departments. A **high earner** in a department is an employee who has a salary in the **top three unique** salaries for that department.

Write a solution to find the employees who are **high earners** in each of the departments.

Return the result table **in any order**.

Answer



```sql
/* Write your T-SQL query statement below */
SELECT Department, employee, salary FROM (
    SELECT d.name AS Department
        , e.name AS employee
        , e.salary
        , DENSE_RANK() OVER (PARTITION BY d.name ORDER BY e.salary DESC) AS drk
    FROM Employee e JOIN Department d ON e.DepartmentId= d.Id
) t WHERE t.drk <= 3
```

❖ Advanced Select and Join

Q: 44

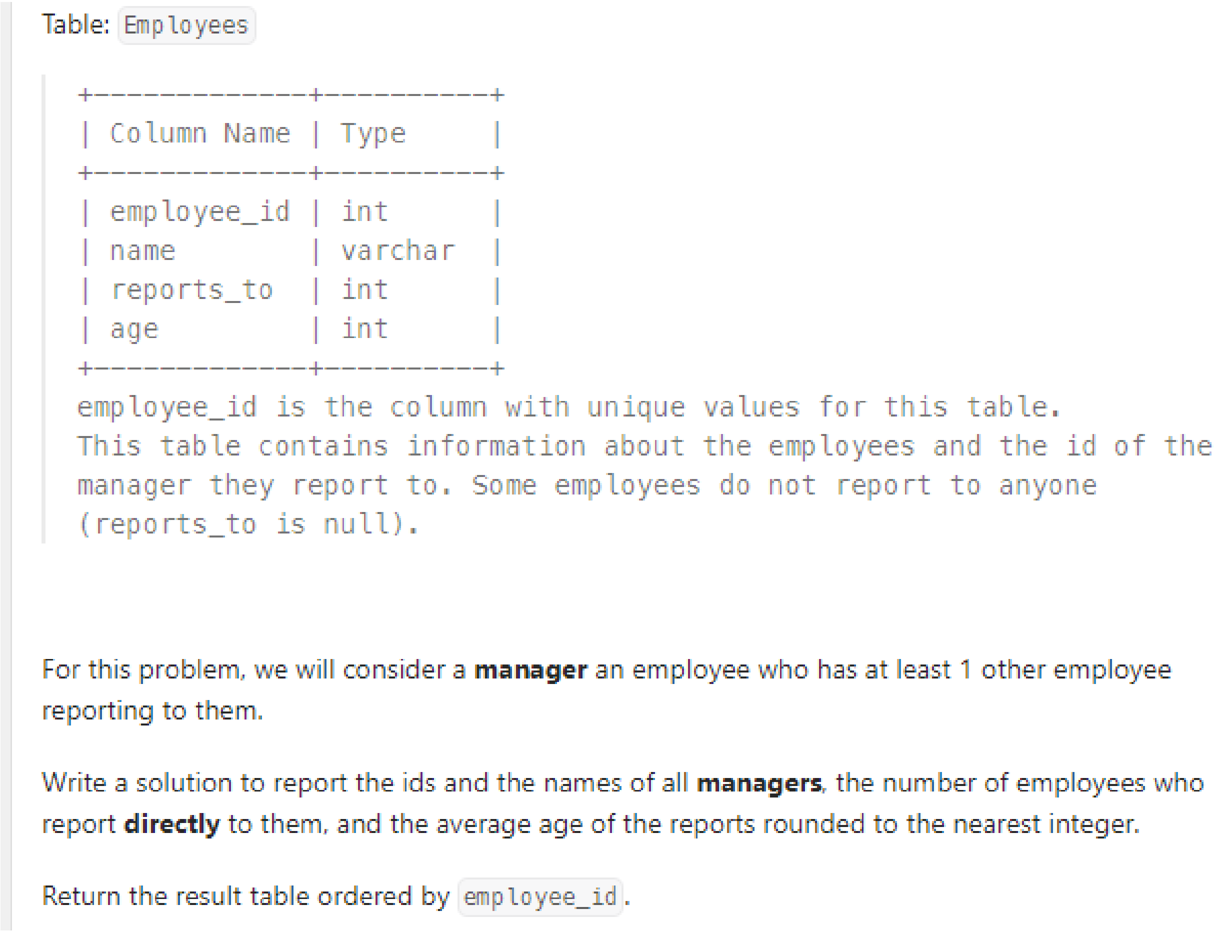


Table: `Employees`

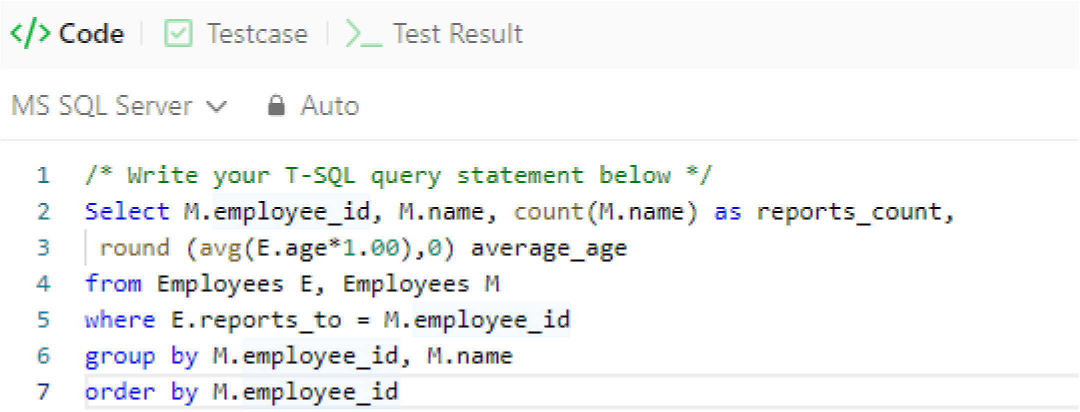
```
+-------------+----------+
| Column Name | Type     |
+-------------+----------+
| employee_id | int      |
| name        | varchar  |
| reports_to  | int      |
| age         | int      |
+-------------+----------+
employee_id is the column with unique values for this table.
This table contains information about the employees and the id of the
manager they report to. Some employees do not report to anyone
(reports_to is null).
```

For this problem, we will consider a **manager** an employee who has at least 1 other employee reporting to them.

Write a solution to report the ids and the names of all **managers**, the number of employees who report **directly** to them, and the average age of the reports rounded to the nearest integer.

Return the result table ordered by `employee_id`.

Answer



```sql
/* Write your T-SQL query statement below */
Select M.employee_id, M.name, count(M.name) as reports_count,
 round (avg(E.age*1.00),0) average_age
from Employees E, Employees M
where E.reports_to = M.employee_id
group by M.employee_id, M.name
order by M.employee_id
```

Q: 45

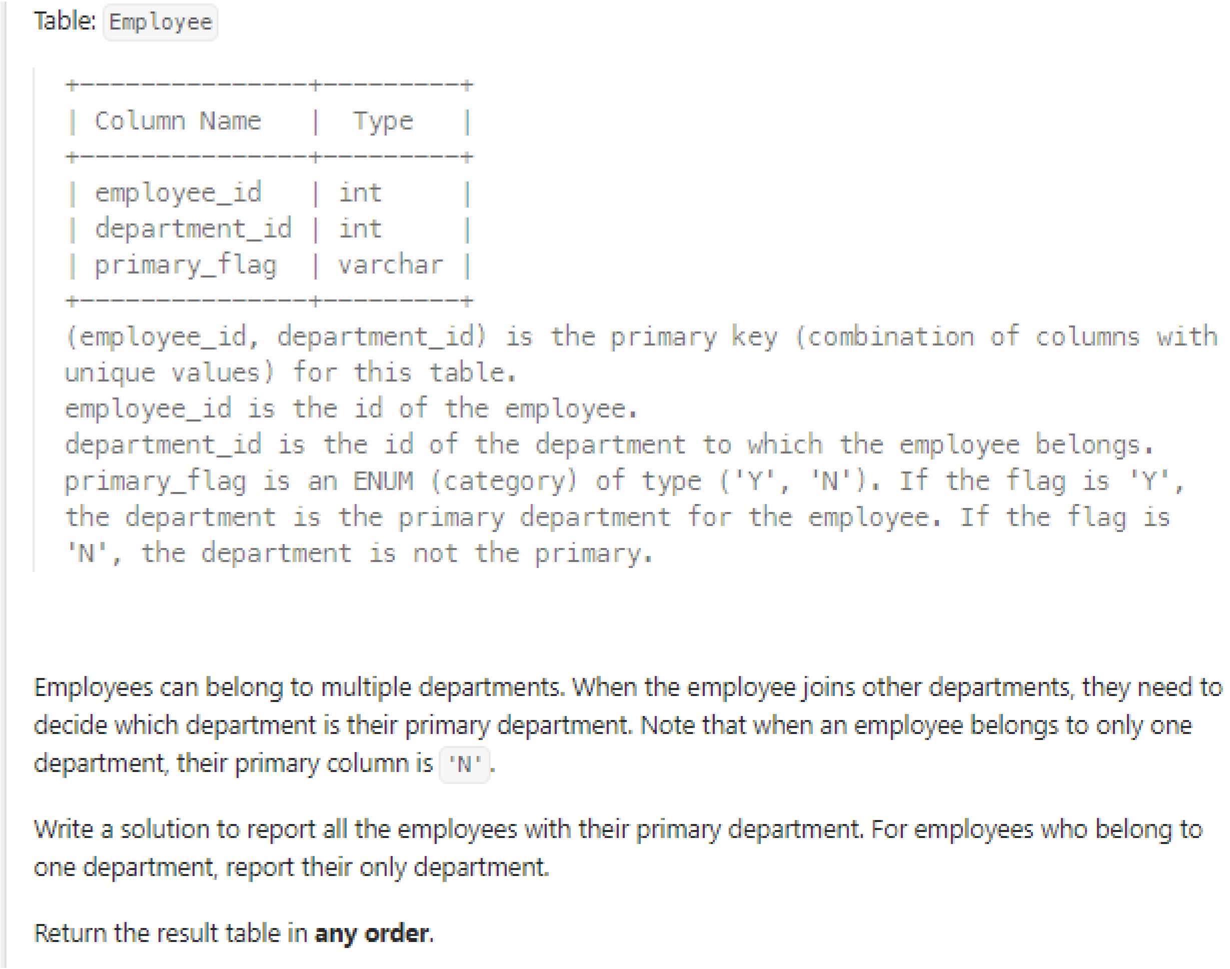


Table: `Employee`

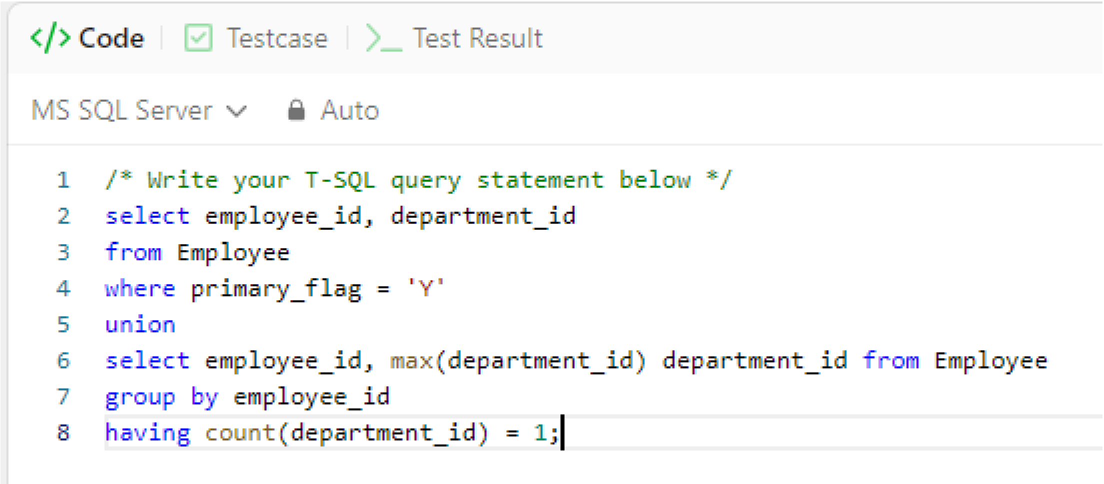
```
+-----------------+----------+
| Column Name     |  Type    |
+-----------------+----------+
| employee_id     | int      |
| department_id   | int      |
| primary_flag    | varchar  |
+-----------------+----------+
(employee_id, department_id) is the primary key (combination of columns with
unique values) for this table.
employee_id is the id of the employee.
department_id is the id of the department to which the employee belongs.
primary_flag is an ENUM (category) of type ('Y', 'N'). If the flag is 'Y',
the department is the primary department for the employee. If the flag is
'N', the department is not the primary.
```

Employees can belong to multiple departments. When the employee joins other departments, they need to decide which department is their primary department. Note that when an employee belongs to only one department, their primary column is `'N'`.

Write a solution to report all the employees with their primary department. For employees who belong to one department, report their only department.

Return the result table in **any order**.

Answer



```sql
/* Write your T-SQL query statement below */
select employee_id, department_id
from Employee
where primary_flag = 'Y'
union
select employee_id, max(department_id) department_id from Employee
group by employee_id
having count(department_id) = 1;
```

Q: 46

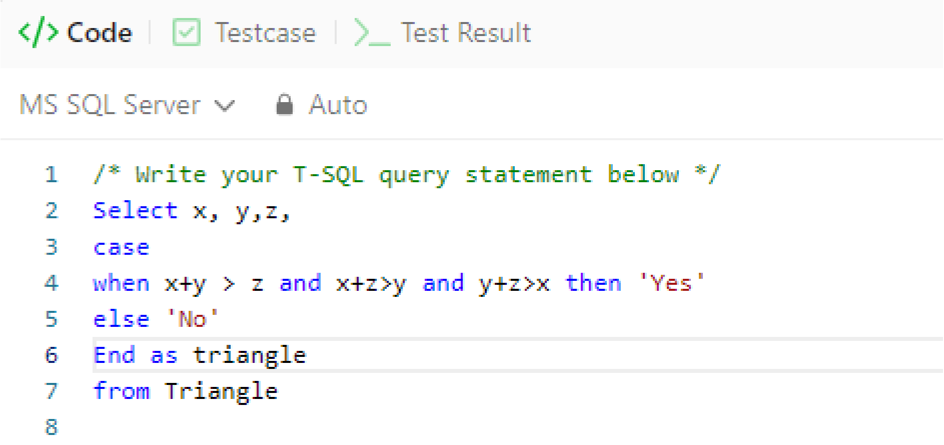
Table: `Triangle`

```
+-------------+------+
| Column Name | Type |
+-------------+------+
| x           | int  |
| y           | int  |
| z           | int  |
+-------------+------+
In SQL, (x, y, z) is the primary key column for this table.
Each row of this table contains the lengths of three line segments.
```

Report for every three line segments whether they can form a triangle.

Return the result table in **any order**.

Answer



```sql
/* Write your T-SQL query statement below */
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

```

Q: 47

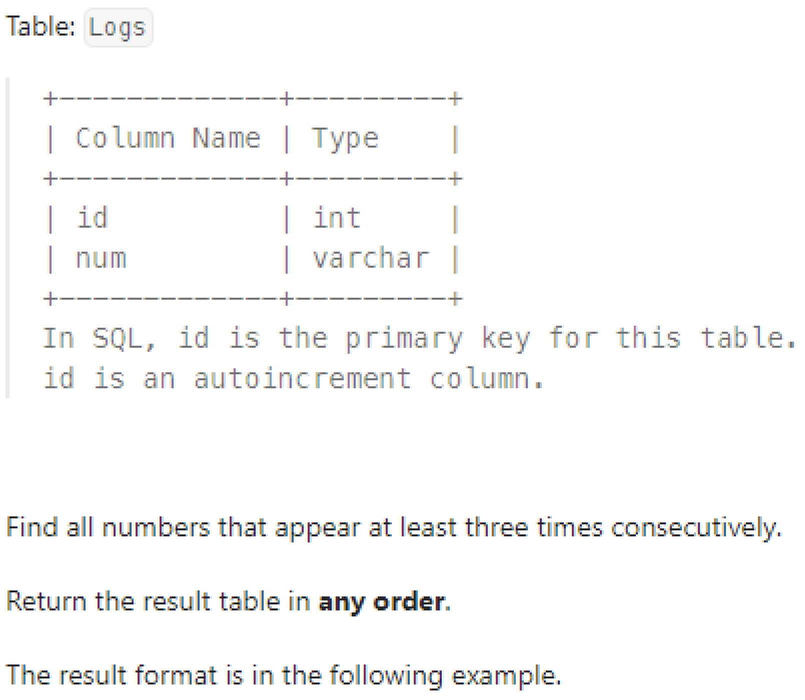


Table: `Logs`

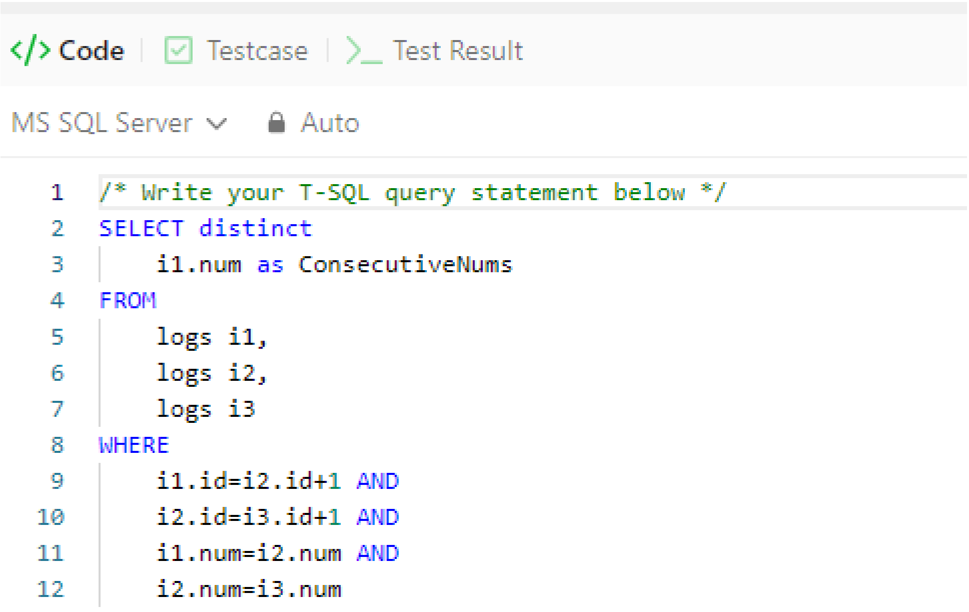
```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| id          | int     |
| num         | varchar |
+-------------+---------+
In SQL, id is the primary key for this table.
id is an autoincrement column.
```

Find all numbers that appear at least three times consecutively.

Return the result table in **any order**.

The result format is in the following example.

Answer

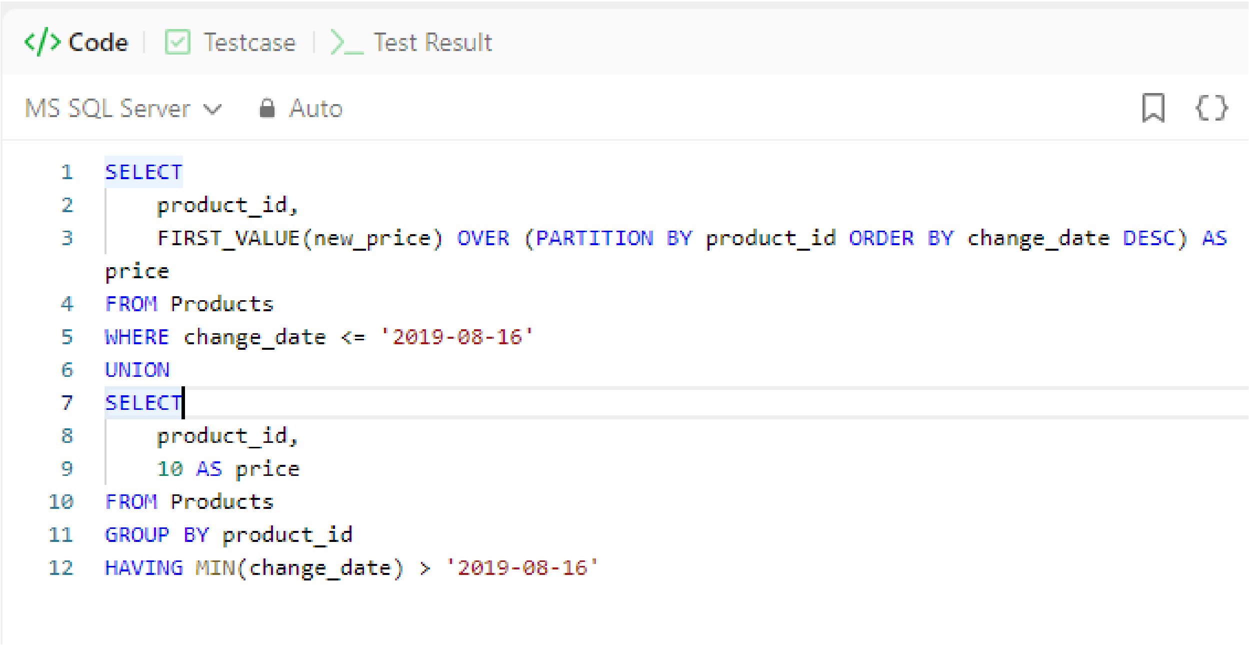


```sql
/* Write your T-SQL query statement below */
SELECT distinct
    i1.num as ConsecutiveNums
FROM
    logs i1,
    logs i2,
    logs i3
WHERE
    i1.id=i2.id+1 AND
    i2.id=i3.id+1 AND
    i1.num=i2.num AND
    i2.num=i3.num
```

Q: 48

> Table: `Products`
>
> ```
> +---------------+---------+
> | Column Name   | Type    |
> +---------------+---------+
> | product_id    | int     |
> | new_price     | int     |
> | change_date   | date    |
> +---------------+---------+
> (product_id, change_date) is the primary key (combination of
> columns with unique values) of this table.
> Each row of this table indicates that the price of some product
> was changed to a new price at some date.
> ```
>
> Write a solution to find the prices of all products on `2019-08-16`. Assume the price of all products before any change is `10`.
>
> Return the result table in **any order**.

Answer



```sql
SELECT
    product_id,
    FIRST_VALUE(new_price) OVER (PARTITION BY product_id ORDER BY change_date DESC) AS price
FROM Products
WHERE change_date <= '2019-08-16'
UNION
SELECT
    product_id,
    10 AS price
FROM Products
GROUP BY product_id
HAVING MIN(change_date) > '2019-08-16'
```

Q: 49

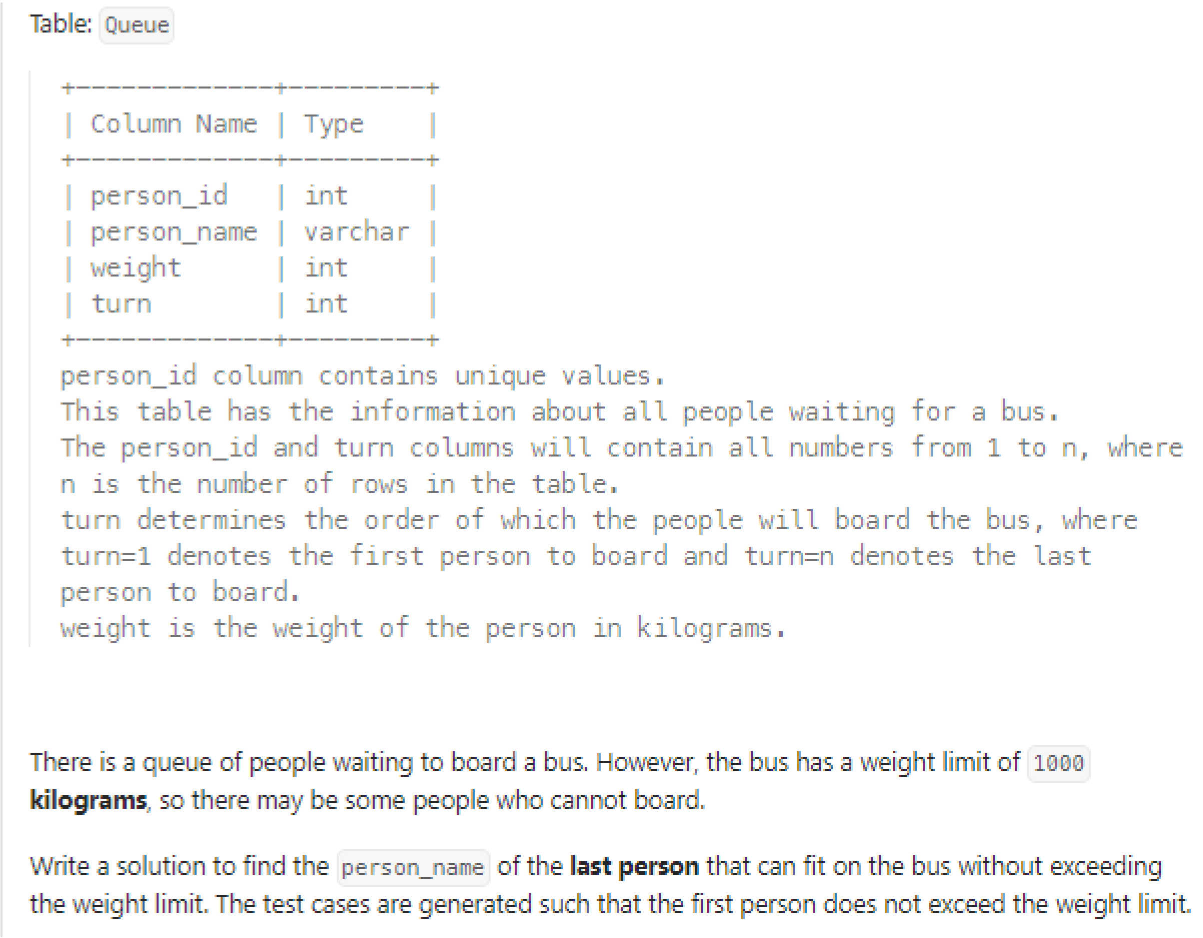


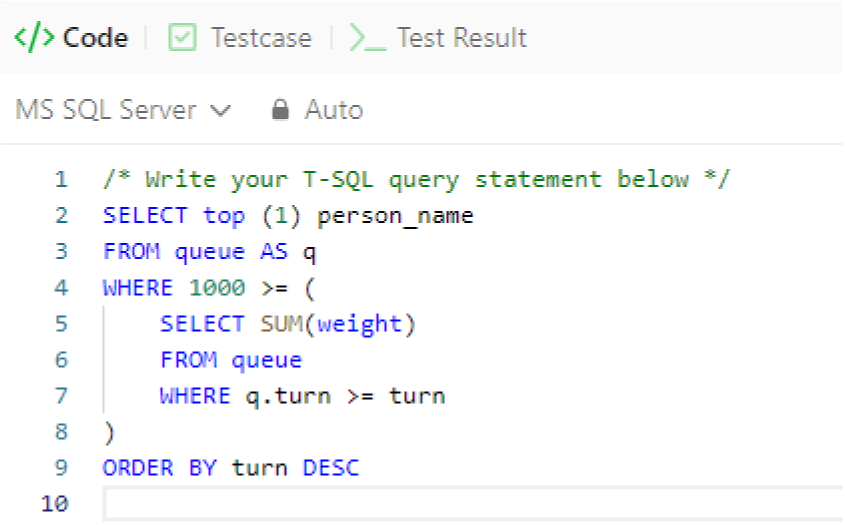
Table: `Queue`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| person_id   | int     |
| person_name | varchar |
| weight      | int     |
| turn        | int     |
+-------------+---------+
person_id column contains unique values.
This table has the information about all people waiting for a bus.
The person_id and turn columns will contain all numbers from 1 to n, where
n is the number of rows in the table.
turn determines the order of which the people will board the bus, where
turn=1 denotes the first person to board and turn=n denotes the last
person to board.
weight is the weight of the person in kilograms.
```

There is a queue of people waiting to board a bus. However, the bus has a weight limit of `1000` **kilograms**, so there may be some people who cannot board.

Write a solution to find the `person_name` of the **last person** that can fit on the bus without exceeding the weight limit. The test cases are generated such that the first person does not exceed the weight limit.

Answer



```sql
/* Write your T-SQL query statement below */
SELECT top (1) person_name
FROM queue AS q
WHERE 1000 >= (
    SELECT SUM(weight)
    FROM queue
    WHERE q.turn >= turn
)
ORDER BY turn DESC
```

Q: 50

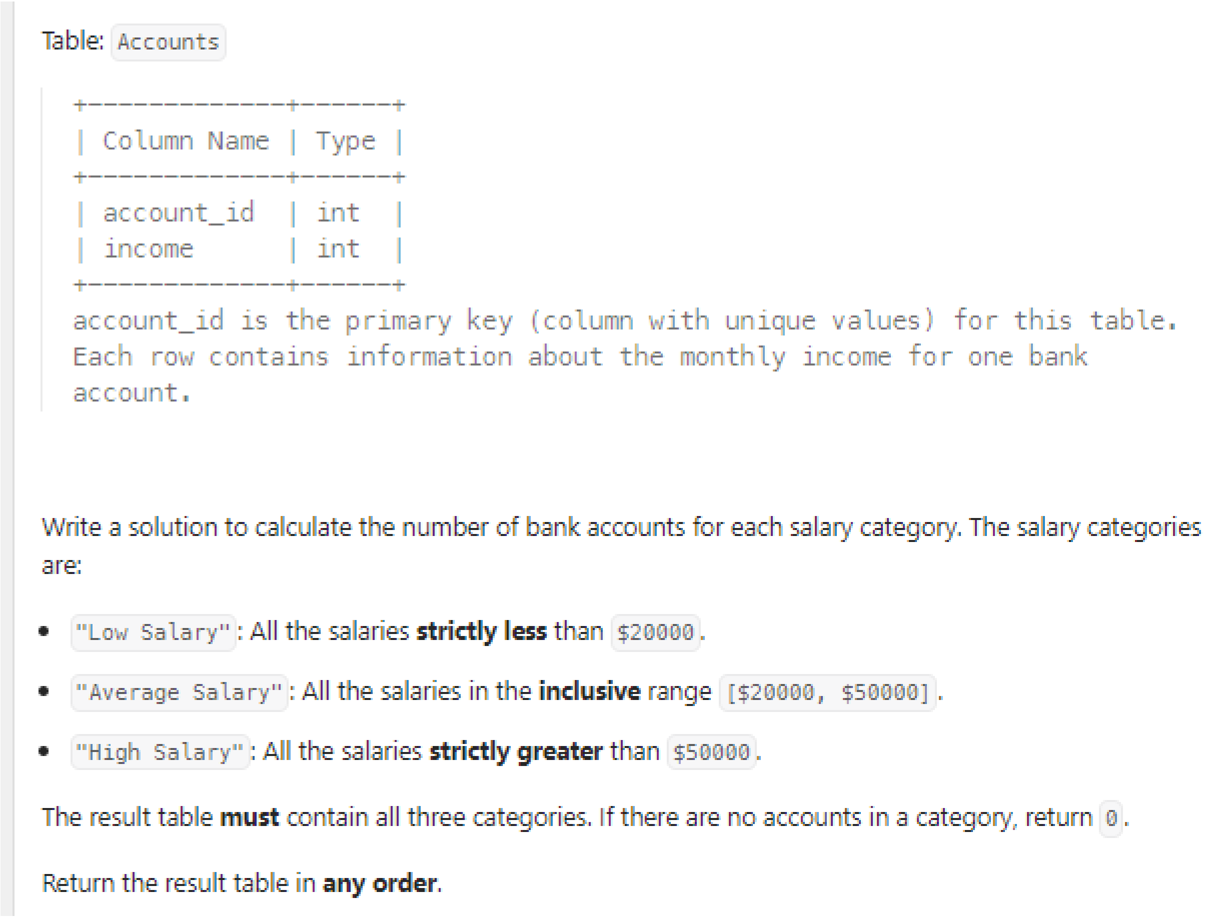


Table: `Accounts`

```
+-------------+------+
| Column Name | Type |
+-------------+------+
| account_id  | int  |
| income      | int  |
+-------------+------+
account_id is the primary key (column with unique values) for this table.
Each row contains information about the monthly income for one bank
account.
```

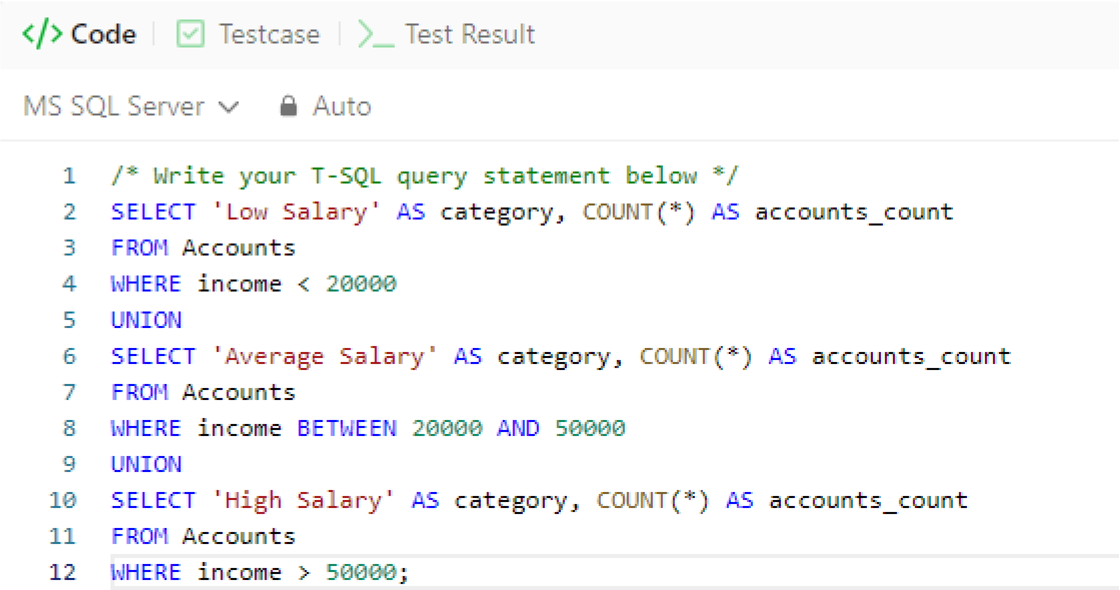
Write a solution to calculate the number of bank accounts for each salary category. The salary categories are:

- `"Low Salary"`: All the salaries **strictly less** than `$20000`.
- `"Average Salary"`: All the salaries in the **inclusive** range `[$20000, $50000]`.
- `"High Salary"`: All the salaries **strictly greater** than `$50000`.

The result table **must** contain all three categories. If there are no accounts in a category, return `0`.

Return the result table in **any order**.

Answer



```sql
/* Write your T-SQL query statement below */
SELECT 'Low Salary' AS category, COUNT(*) AS accounts_count
FROM Accounts
WHERE income < 20000
UNION
SELECT 'Average Salary' AS category, COUNT(*) AS accounts_count
FROM Accounts
WHERE income BETWEEN 20000 AND 50000
UNION
SELECT 'High Salary' AS category, COUNT(*) AS accounts_count
FROM Accounts
WHERE income > 50000;
```