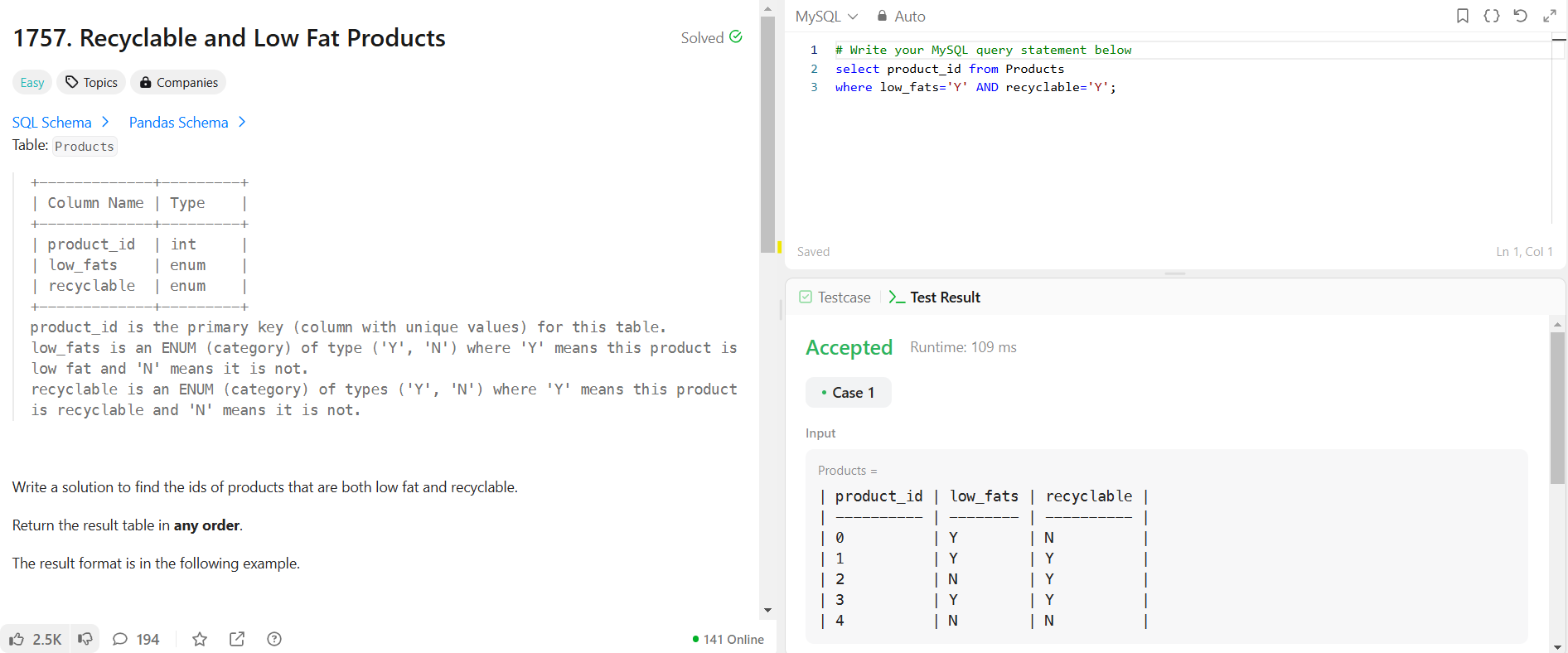
**SQL50**

[**1757. Recyclable and Low Fat Products**](https://leetcode.com/problems/recyclable-and-low-fat-products/)

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



*select product\_id from Products*

*where low\_fats='Y' AND recyclable='Y';*

[**584. Find Customer Referee**](https://leetcode.com/problems/find-customer-referee/)

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

A screenshot of a computer

Description automatically generated

*select name from Customer*

*where referee\_id <> 2 OR referee\_id IS NULL;*

[**595. Big Countries**](https://leetcode.com/problems/big-countries/)

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

A screenshot of a computer

Description automatically generated

*SELECT name, population, area FROM World*

*where area >= 3000000 OR population >= 25000000;*

[**1148. Article Views I**](https://leetcode.com/problems/article-views-i/)

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.

A screenshot of a computer

Description automatically generated

SELECT DISTINCT author\_id AS id FROM Views

WHERE author\_id = viewer\_id

ORDER BY author\_id ASC;

[**1683. Invalid Tweets**](https://leetcode.com/problems/invalid-tweets/)

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.

A screenshot of a computer

Description automatically generated

*SELECT tweet\_id FROM Tweets*

*WHERE length(content)>15 ;*

[**1378. Replace Employee ID With The Unique Identifier**](https://leetcode.com/problems/replace-employee-id-with-the-unique-identifier/)

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

A screenshot of a computer

Description automatically generated

*SELECT EmployeeUNI.unique\_id,Employees.name*

*FROM Employees*

*LEFT JOIN EmployeeUNI ON Employees.id=EmployeeUNI.id;*

[**1068. Product Sales Analysis I**](https://leetcode.com/problems/product-sales-analysis-i/)

Write a solution to report the product\_name, year, and price for each sale\_id in the Sales table.

A screenshot of a computer

Description automatically generated

*SELECT Product.product\_name, Sales.year, Sales.price from Sales*

*INNER JOIN Product ON Sales.product\_id=Product.product\_id;*

[**1581. Customer Who Visited but Did Not Make Any Transactions**](https://leetcode.com/problems/customer-who-visited-but-did-not-make-any-transactions/)

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.

A screenshot of a computer

Description automatically generated

*SELECT Visits.customer\_id, count(Visits.customer\_id ) AS count\_no\_trans  from Visits*

*LEFT JOIN Transactions ON Visits.visit\_id= Transactions.visit\_id*

*where Transactions.visit\_id IS NULL*

*GROUP BY Visits.customer\_id;*

[**197. Rising Temperature**](https://leetcode.com/problems/rising-temperature/)

Write a solution to find all dates' id with higher temperatures compared to its previous dates (yesterday).A screenshot of a computer

Description automatically generated

*SELECT w.id AS Id from Weather w*

*JOIN Weather t ON DATEDIFF(w.recordDate,t.recordDate)=1*

*AND w.temperature > t.temperature;*

[**1661. Average Time of Process per Machine**](https://leetcode.com/problems/average-time-of-process-per-machine/)

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**.

*A screenshot of a computer

Description automatically generated*

*SELECT a.machine\_id, ROUND(AVG(b.timestamp - a.timestamp),3) AS processing\_time from Activity a*

*JOIN Activity b ON a.machine\_id = b.machine\_id*

*AND a.process\_id = b.process\_id*

*AND a.activity\_type = "start"*

*AND b.activity\_type = "end"*

*GROUP BY a.machine\_id*

[**577. Employee Bonus**](https://leetcode.com/problems/employee-bonus/)

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

A screenshot of a computer

Description automatically generated

*SELECT e.name, b.bonus from Employee e*

*LEFT JOIN bONUS b ON e.empId=b.empId*

*WHERE b.bonus IS NULL OR b.bonus<1000 ;*

[**1280. Students and Examinations**](https://leetcode.com/problems/students-and-examinations/)

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

A screenshot of a computer

Description automatically generated

*SELECT Students.student\_id, Students.student\_name, Subjects.subject\_name, COUNT(Examinations.subject\_name) AS attended\_exams from Students*

*CROSS JOIN Subjects*

*LEFT JOIN Examinations ON Students.student\_id = Examinations.student\_id*

*AND Subjects.subject\_name = Examinations.subject\_name*

*GROUP BY Students.student\_id,Subjects.subject\_name*

*ORDER BY Students.student\_id;*

[**570. Managers with at Least 5 Direct Reports**](https://leetcode.com/problems/managers-with-at-least-5-direct-reports/)

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

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

The result format is in the following example.

A screenshot of a computer

Description automatically generated

*select E1.name FROM Employee E1*

*JOIN*

*(SELECT managerId, count(\*) AS Dr FROM Employee*

*GROUP BY managerId*

*HAVING count(\*)>=5) E2*

*ON E1.id = E2.managerId*

[**1934. Confirmation Rate**](https://leetcode.com/problems/confirmation-rate/)

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.

A screenshot of a computer

Description automatically generated

*SELECT s.user\_id, ROUND(AVG(IF(action='confirmed',1,0)),2) AS confirmation\_rate FROM Signups s*

*LEFT JOIN Confirmations c ON s.user\_id=c.user\_id*

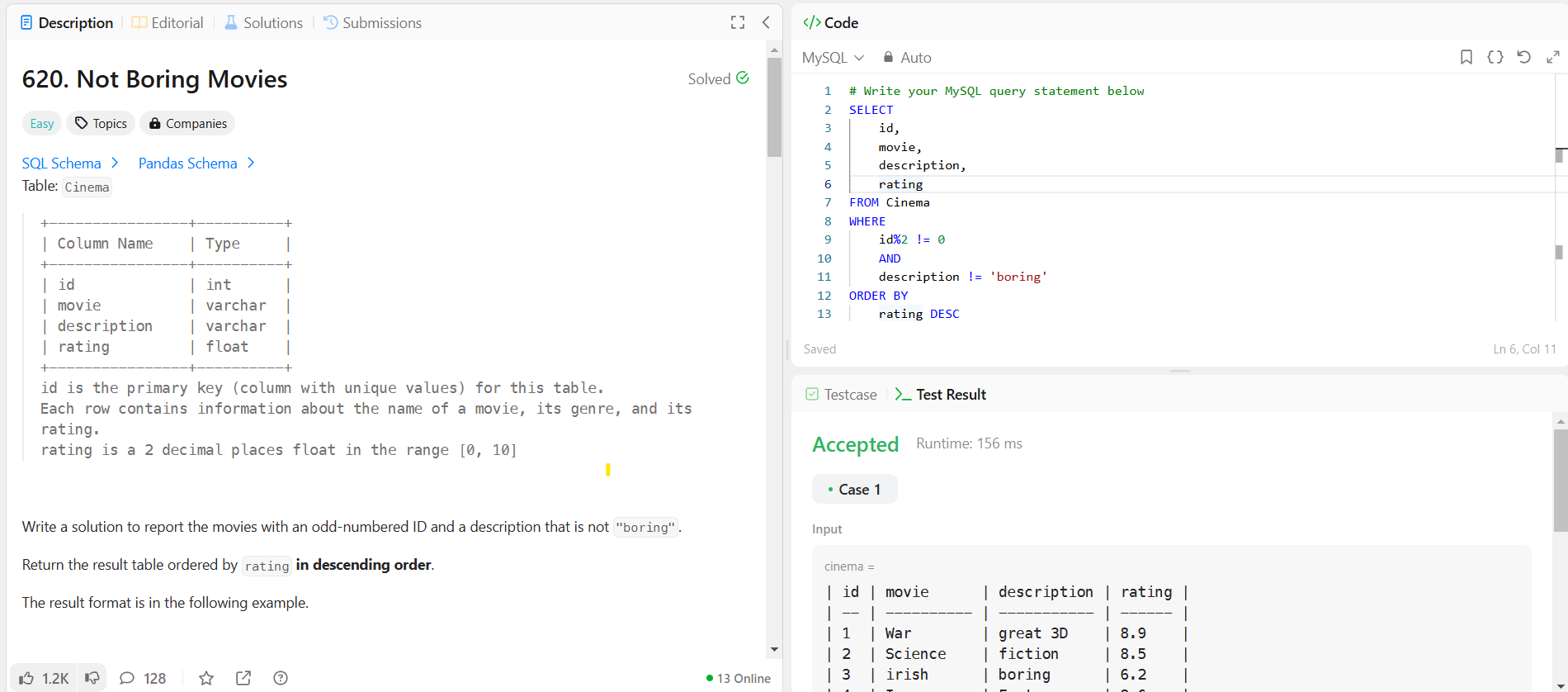
*GROUP BY s.user\_id*

[**620. Not Boring Movies**](https://leetcode.com/problems/not-boring-movies/)

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**.

The result format is in the following example.



*SELECT id, movie, description, rating FROM Cinema*

*WHERE*

*id%2 != 0*

*AND*

*description != 'boring'*

*ORDER BY*

*rating DESC*

[**1251. Average Selling Price**](https://leetcode.com/problems/average-selling-price/)

Write a solution to find the average selling price for each product. average\_price should be **rounded to 2 decimal places**. If a product does not have any sold units, its average selling price is assumed to be 0.

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

A screenshot of a computer

Description automatically generated

*SELECT*

*p.product\_id,*

*IFNULL(ROUND(SUM(units\*price) / SUM(units),2), 0) AS average\_price*

*FROM Prices p*

*LEFT JOIN UnitsSold u*

*ON (p.product\_id = u.product\_id) AND (u.purchase\_date BETWEEN p.start\_date AND p.end\_date)*

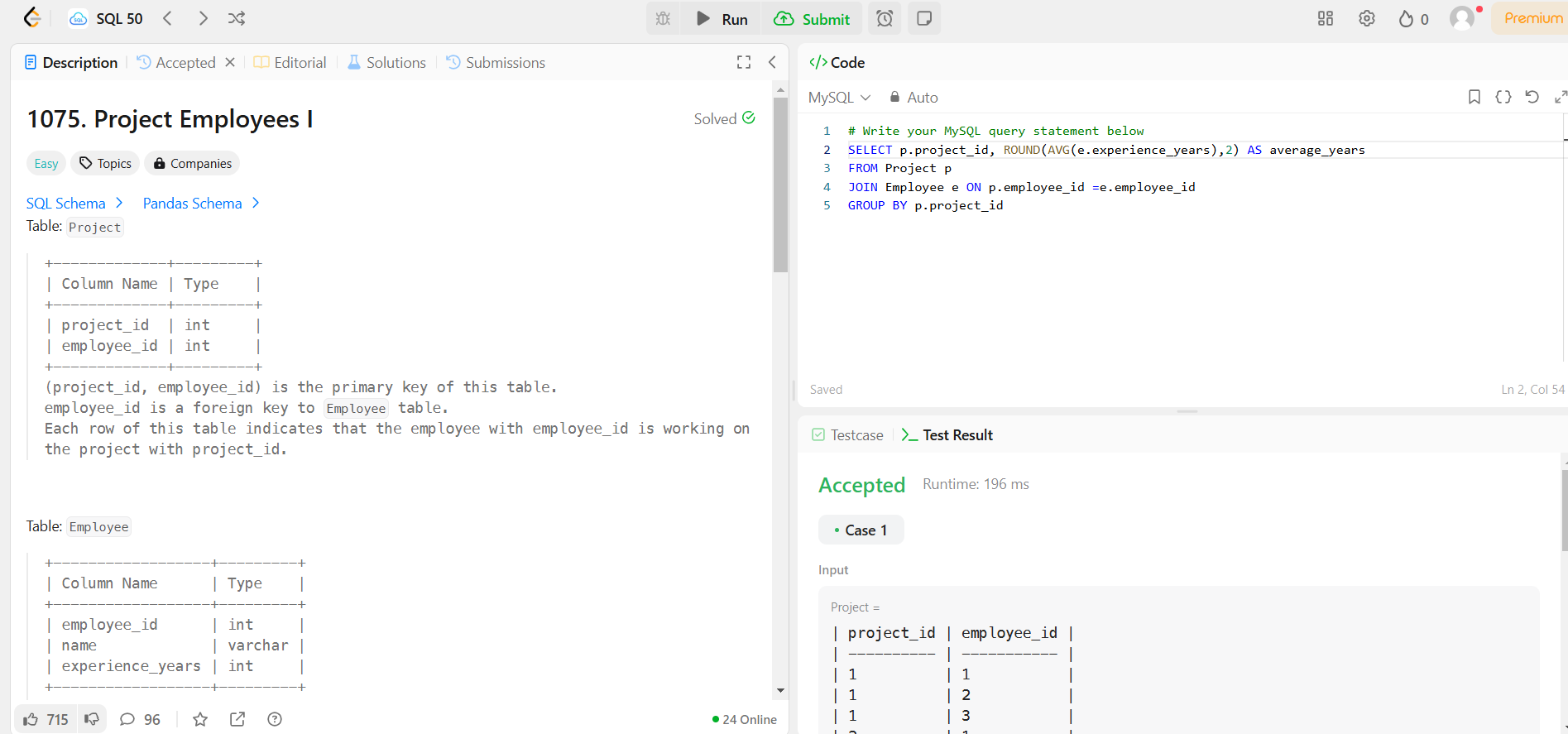
*GROUP BY p.product\_id*

[**1075. Project Employees I**](https://leetcode.com/problems/project-employees-i/)

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

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

The query result format is in the following example.



*SELECT p.project\_id, ROUND(AVG(e.experience\_years),2) AS average\_years*

*FROM Project p*

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

*GROUP BY p.project\_id*

[**1633. Percentage of Users Attended a Contest**](https://leetcode.com/problems/percentage-of-users-attended-a-contest/)

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.

*A screenshot of a computer

Description automatically generated*

*SELECT contest\_id, ROUND(((count(DISTINCT user\_id)/(SELECT COUNT(user\_id) FROM Users))\*100),2) AS percentage  FROM Register*

*GROUP BY contest\_id*

*ORDER BY percentage DESC, contest\_id*

[**1211. Queries Quality and Percentage**](https://leetcode.com/problems/queries-quality-and-percentage/)

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.

A screenshot of a computer

Description automatically generated

*SELECT query\_name,*

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

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

*FROM Queries*

*GROUP BY query\_name*

[**1193. Monthly Transactions I**](https://leetcode.com/problems/monthly-transactions-i/)

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.

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

*A screenshot of a computer

Description automatically generated*

*SELECT DATE\_FORMAT(trans\_date,'%Y-%m') AS month,*

*country,*

*COUNT(trans\_date) AS trans\_count,*

*SUM(IF(state='approved',1,0)) AS approved\_count,*

*SUM(amount) AS trans\_total\_amount,*

*SUM(IF(state='approved',amount,0)) AS approved\_total\_amount*

*FROM Transactions*

*GROUP BY month, country*