

```
mysql> use sample;
```

```
Database changed
```

```
mysql> CREATE TABLE BooksStore (
```

```
->     BookID INT AUTO_INCREMENT PRIMARY KEY,
```

```
->     Title VARCHAR(255) NOT NULL,
```

```
->     Author VARCHAR(255),
```

```
->     Genre VARCHAR(50),
```

```
->     Price DECIMAL(10, 2),
```

```
->     Rating DECIMAL(3, 2),
```

```
->     Sales INT,
```

```
->     PublishedYear INT
```

```
-> );
```

```
Query OK, 0 rows affected (0.03 sec)
```

```
mysql> INSERT INTO BooksStore (Title, Author, Genre, Price, Rating, Sales, PublishedYear)
```

```
-> VALUES
```

```
-> ('The Alchemist', 'Paulo Coelho', 'Fiction', 15.99, 4.7, 250000, 1988),
```

```
-> ('Becoming', 'Michelle Obama', 'Biography', 20.50, 4.8, 500000, 2018),
```

```
-> ('Atomic Habits', 'James Clear', 'Self-Help', 11.99, 4.9, 300000, 2018),
```

```
-> ('1984', 'George Orwell', 'Dystopian', 9.99, 4.6, 450000, 1949),
```

```
-> ('Sapiens', 'Yuval Noah Harari', 'History', 14.99, 4.7, 400000, 2011),
```

```
-> ('The Subtle Art of Not Giving a F*ck', 'Mark Manson', 'Self-Help', 12.99, 4.3, 200000, 2016),
```

```
-> ('The Great Gatsby', 'F. Scott Fitzgerald', 'Classic', 10.99, 4.4, 350000, 1925),
```

```
-> ('Educated', 'Tara Westover', 'Memoir', 16.99, 4.7, 220000, 2018);
```

```
Query OK, 8 rows affected (0.01 sec)
```

```
Records: 8  Duplicates: 0  Warnings: 0
```

```
mysql> -- Total number of books
mysql> SELECT COUNT(*) AS TotalBooks FROM BooksStore;
```

TotalBooks
8

```
1 row in set (0.01 sec)
```

```
mysql>
mysql> -- Average price and rating
mysql> SELECT AVG(Price) AS AvgPrice, AVG(Rating) AS AvgRating FROM BooksStore;
```

AvgPrice	AvgRating
14.303750	4.637500

```
1 row in set (0.00 sec)
```

```
mysql>
mysql> -- Total and average sales
mysql> SELECT SUM(Sales) AS TotalSales, AVG(Sales) AS AvgSales FROM BooksStore;
```

TotalSales	AvgSales
2670000	333750.0000

```
1 row in set (0.00 sec)
```

```
mysql> -- Count of books by genre
```

```
mysql> SELECT Genre, COUNT(*) AS Count FROM BooksStore GROUP BY Genre;
```

Genre	Count
Fiction	1
Biography	1
Self-Help	2
Dystopian	1
History	1
Classic	1
Memoir	1

```
7 rows in set (0.00 sec)
```

```
mysql> -- Books with a rating of 4.7 or higher
```

```
mysql> SELECT Title, Author, Rating FROM BooksStore WHERE Rating >= 4.7 ORDER BY Rating DESC;
```

Title	Author	Rating
Atomic Habits	James Clear	4.90
Becoming	Michelle Obama	4.80
The Alchemist	Paulo Coelho	4.70
Sapiens	Yuval Noah Harari	4.70
Educated	Tara Westover	4.70

```
5 rows in set (0.00 sec)
```

```
mysql> -- Top-selling books
```

```
mysql> SELECT Title, Author, Sales FROM BooksStore ORDER BY Sales DESC LIMIT 5;
```

Title	Author	Sales
Becoming	Michelle Obama	500000
1984	George Orwell	450000
Sapiens	Yuval Noah Harari	400000
The Great Gatsby	F. Scott Fitzgerald	350000
Atomic Habits	James Clear	300000

```
5 rows in set (0.00 sec)
```

```
mysql>
```

```
mysql> -- Sales by genre
```

```
mysql> SELECT Genre, SUM(Sales) AS TotalSales FROM BooksStore GROUP BY Genre;
```

Genre	TotalSales
Fiction	250000
Biography	500000
Self-Help	500000
Dystopian	450000
History	400000
Classic	350000
Memoir	220000

```
7 rows in set (0.00 sec)
```

```
mysql>
mysql> -- Price distribution by genre
mysql> SELECT Genre, AVG(Price) AS AvgPrice FROM BooksStore GROUP BY Genre;
```

Genre	AvgPrice
Fiction	15.990000
Biography	20.500000
Self-Help	12.490000
Dystopian	9.990000
History	14.990000
Classic	10.990000
Memoir	16.990000

7 rows in set (0.00 sec)

```
mysql> -- Books published after 2000
mysql> SELECT Title, Author, PublishedYear FROM BooksStore WHERE PublishedYear > 2000;
```

Title	Author	PublishedYear
Becoming	Michelle Obama	2018
Atomic Habits	James Clear	2018
Sapiens	Yuval Noah Harari	2011
The Subtle Art of Not Giving a F*ck	Mark Manson	2016
Educated	Tara Westover	2018

5 rows in set (0.00 sec)

[Description](#)
[Editorial](#)
[Solutions](#)
[Submissions](#)

## 1174. Immediate Food Delivery II

Medium
Topics
Companies

[SQL Schema](#) > [Pandas Schema](#) >

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

861 179

42 Online

Code

MySQL Auto

```

1 # Write your MySQL query statement below
2 select
3 round(100*sum(case when b.min_order_date = b.min_delivery_date then 1 else 0
4 end)/count(*), 2)
5 as immediate_percentage
6 from (
7     select min(order_date) as min_order_date, min(customer_pref_delivery_date)
8     as min_delivery_date
9     from delivery
10    group by customer_id
11 ) b;
```

Saved

Ln 9, Col 5

[Testcase](#) | [Test Result](#)

Accepted Runtime: 129 ms

Case 1

Input

Delivery =

delivery_id	customer_id	order_date	customer_pref_delivery_date
_____	_____	_____	_____
_____	_____	_____	_____

## 570. Managers with at Least 5 Direct Reports

Medium

Topics

Companies

Hint

[SQL Schema](#) > [Pandas Schema](#) >

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 themselves.

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

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

1.3K | 251 | ☆ | 📄 | ⌚

48 Online

Code

MySQL Auto

```

1 # Write your MySQL query statement below
2 SELECT name
3 FROM
4     Employee
5     JOIN (
6         SELECT managerId AS id, COUNT(1) AS cnt
7         FROM Employee
8         GROUP BY 1
9         HAVING cnt >= 5
10     ) AS t
11     USING (id);

```

Saved

Ln 9, Col 24

Testcase | Test Result

Accepted Runtime: 176 ms

Case 1

Input

Employee =

id	name	department	managerId
101	John	A	null
102	Dan	A	101

[Description](#) | [Editorial](#) | [Solutions](#) | [Submissions](#)[SQL Schema](#) > [Pandas Schema](#) >

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.

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

The query result format is in the following example.

Example 1:



972



140



42 Online

&lt;/&gt; Code

MySQL Auto



```
1 # Write your MySQL query statement below
2 SELECT
3     DATE_FORMAT(trans_date, '%Y-%m') AS month,
4     country,
5     COUNT(1) AS trans_count,
6     SUM(state = 'approved') AS approved_count,
7     SUM(amount) AS trans_total_amount,
8     SUM(IF(state = 'approved', amount, 0)) AS approved_total_amount
9 FROM Transactions
10 GROUP BY 1, 2;
```

Saved

Ln 10, Col 15

☒ Testcase | [Test Result](#)

Accepted Runtime: 136 ms

• Case 1

Input

Transactions =

id	country	state	amount	trans_date
121	US	approved	1000	2018-12-18
122	US	declined	2000	2018-12-19



# 1934. Confirmation Rate

Medium

Topics

Companies

[SQL Schema](#) > [Pandas Schema](#) >

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

Code

MySQL Auto

```

1 # Write your MySQL query statement below
2 select Signups.user_id, ifnull(round(sum(action = 'confirmed') / count(*),
3     2), 0.00) as confirmation_rate
4     from Signups
5     left join Confirmations
6     on Signups.user_id = Confirmations.user_id
7     group by Signups.user_id;
    
```

Saved

Ln 6, Col 30

[Testcase](#) | [Test Result](#)

Accepted Runtime: 172 ms

Case 1

Input

Signups =

user_id	time_stamp
3	2020-03-21 10:16:13
7	2020-01-04 13:57:59