

Question 1

Display the highest, lowest, sum, and average salary of all employees. Label the columns Maximum, Minimum , Sum , and Average , respectively. Round your results to the nearest whole number. in sql

Query

```
SELECT
```

```
    ROUND(MAX(salary)) AS Maximum,
```

```
    ROUND(MIN(salary)) AS Minimum,
```

```
    ROUND(SUM(salary)) AS Sum,
```

```
    ROUND(AVG(salary)) AS Average
```

```
FROM hr.employees;
```

Output

MAXIMUM	MINIMUM	SUM	AVERAGE
24000	2100	692016	6467

[Download CSV](#)

Question 2

Modify the query in Q1 to display the minimum, maximum, sum, and average salary for each job type.

Query

```
SELECT
```

```
    job_id,
```

```
    ROUND(MIN(salary)) AS Minimum,
```

```
    ROUND(MAX(salary)) AS Maximum,
```

```
    ROUND(SUM(salary)) AS Sum,
```

```

ROUND(AVG(salary)) AS Average
FROM hr.employees
GROUP BY job_id;

```

Output

JOB_ID	MINIMUM	MAXIMUM	SUM	AVERAGE
AD_VP	17000	17000	34000	17000
FI_ACCOUNT	6900	9000	39600	7920
PU_CLERK	2500	3100	13900	2780
SH_CLERK	2500	4200	64300	3215
HR_REP	6500	6500	6500	6500
PU_MAN	11000	11000	11000	11000
AC_MGR	12008	12008	12008	12008
ST_CLERK	2100	3600	55700	2785
AD_ASST	4400	4400	4400	4400
IT_PROG	4200	9000	28800	5760
SA_MAN	10500	14000	61000	12200
AC_ACCOUNT	8300	8300	8300	8300
FI_MGR	12008	12008	12008	12008
ST_MAN	5800	8200	36400	7280
AD_PRES	24000	24000	24000	24000
MK_MAN	13000	13000	13000	13000

HR_REP	6500	6500	6500	6500
PU_MAN	11000	11000	11000	11000
AC_MGR	12008	12008	12008	12008
ST_CLERK	2100	3600	55700	2785
AD_ASST	4400	4400	4400	4400
IT_PROG	4200	9000	28800	5760
SA_MAN	10500	14000	61000	12200
AC_ACCOUNT	8300	8300	8300	8300
FI_MGR	12008	12008	12008	12008
ST_MAN	5800	8200	36400	7280
AD_PRES	24000	24000	24000	24000
MK_MAN	13000	13000	13000	13000
SA_REP	6100	11500	251100	8370
MK_REP	6000	6000	6000	6000
PR_REP	10000	10000	10000	10000

[Download CSV](#)

19 rows selected.

Question 3

Write a query to display the number of people with the same job.

Query

```

SELECT
    job_id,
    COUNT(*) AS employee_count
FROM hr.employees
GROUP BY job_id;

```

Output

JOB_ID	EMPLOYEE_COUNT	FI_ACCOUNT	5
AC_ACCOUNT	1	FI_MGR	1
AC_MGR	1	HR_REP	1
AD_ASST	1	IT_PROG	5
AD_PRES	1	MK_MAN	1
AD_VP	2	MK_REP	1
FI_ACCOUNT	5	PR_REP	1
FI_MGR	1	PU_CLERK	5
HR_REP	1	PU_MAN	1
IT_PROG	5	SA_MAN	5
MK_MAN	1	SA_REP	30
MK_REP	1	SH_CLERK	20
PR_REP	1	ST_CLERK	20
PU_CLERK	5	ST_MAN	5
PU_MAN	1		
SA_MAN	5		
SA_REP	30		

Download CSV

19 rows selected.

Question 4

4Write a query to display each department's name, location, number of employees, and the average salary for all employees in that department. Label the columns Name , Location , Number of People , and Salary , respectively. Round the average salary to two decimal places.

SELECT

d.department_name AS "Name",
 l.city AS "Location",
 COUNT(e.employee_id) AS "Number of People",

```

ROUND(AVG(e.salary), 2) AS "Salary"

FROM

  hr.employees e

JOIN

  hr.departments d ON e.department_id = d.department_id

JOIN

  hr.locations l ON d.location_id = l.location_id

GROUP BY

  d.department_name, l.city

ORDER BY

  d.department_name;

```

Output

Name	Location	Number of People	Salary
Accounting	Seattle	2	10154
Administration	Seattle	1	4400
Executive	Seattle	3	19333.33
Finance	Seattle	6	8601.33
Human Resources	London	1	6500
IT	Southlake	5	5760
Marketing	Toronto	2	9500
Public Relations	Munich	1	10000
Purchasing	Seattle	6	4150
Sales	Oxford	34	8973.53
Shipping	South San Francisco	45	3475.56

Download CSV

11 rows selected.

Question 5

Display the manager number and the salary of the lowest paid employee for that manager.

Exclude anyone whose manager is not known. Exclude any groups where the minimum

salary is less than \$6,000. Sort the output in descending order of salary.

Query

SELECT

e.manager_id AS "Manager Number",

MIN(e.salary) AS "Lowest Salary"

FROM

hr.employees e

WHERE

e.manager_id IS NOT NULL

GROUP BY

e.manager_id

HAVING

MIN(e.salary) >= 6000

ORDER BY

"Lowest Salary" DESC;

Output

Manager Number	Lowest Salary
102	9000
205	8300
145	7000
146	7000
108	6900
149	6200
147	6200
148	6100
201	6000

[Download CSV](#)

9 rows selected.

Question 6

Create a query that will display the total number of employees and, of that total, the number of employees hired in 2005, 2006, and 2007. Create appropriate column headings.

Query

SELECT

COUNT(*) AS "Total Employees",

SUM(CASE WHEN EXTRACT(YEAR FROM hire_date) = 2005 THEN 1 ELSE 0 END) AS "Hired in 2005",

SUM(CASE WHEN EXTRACT(YEAR FROM hire_date) = 2006 THEN 1 ELSE 0 END) AS "Hired in 2006",

SUM(CASE WHEN EXTRACT(YEAR FROM hire_date) = 2007 THEN 1 ELSE 0 END) AS "Hired in 2007"

FROM

hr.employees;

Output

Total Employees	Hired in 2005	Hired in 2006	Hired in 2007
107	29	24	19

Download CSV

Question 7

Complete the section “Basic Aggregate Functions” of SQL 50 Badge on Leetcode.

[550. Game Play Analysis IV](#)

550. Game Play Analysis IV

Medium Topics Companies

SQL Schema > Pandas Schema >

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.

The result format is in the following example.

Example 1:

Input:

Activity table:

player_id	device_id	event_date	games_played
1	2	2016-03-01	5
1	2	2016-03-02	6
2	3	2017-06-25	1
3	1	2016-03-02	0
3	4	2016-07-03	5

Code

```

1 WITH
2   Players AS (
3     SELECT player_id, MIN(event_date) AS first_login
4     FROM Activity
5     GROUP BY 1
6   )
7   SELECT ROUND(
8     COUNT(Players.player_id) / (
9       SELECT COUNT(DISTINCT Activity.player_id)
10      FROM Activity
11    ),
12    2) AS fraction
13 FROM Players
14 INNER JOIN Activity
15   ON (
16     players.player_id = Activity.player_id
17    AND DATEDIFF(Players.first_login, Activity.event_date) = -1)

```

Testcase Test Result

Accepted Runtime: 94 ms

Case 1

Input

Activity =

player_id	device_id	event_date	games_played
1	2	2016-03-01	5
1	2	2016-03-02	6
2	3	2017-06-25	1
3	1	2016-03-02	0
3	4	2016-07-03	5

1174. Immediate Food Delivery II

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 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**.

The result format is in the following example.

Example 1:

Input:

Delivery table:

delivery_id	customer_id	order_date	customer_pref_delivery_date
1	1	2019-08-01	2019-08-02
2	2	2019-08-02	2019-08-02
3	1	2019-08-11	2019-08-12
4	3	2019-08-24	2019-08-24
5	3	2019-08-21	2019-08-22
6	2	2019-08-11	2019-08-13

Code

```

1 WITH
2   CustomerToIsImmediate AS(
3     SELECT
4       DISTINCT customer_id,
5       FIRST_VALUE(order_date = customer_pref_delivery_date) OVER(
6         PARTITION BY customer_id
7         ORDER BY order_date
8       ) IS immediate
9     FROM Delivery
10  )
11   SELECT ROUND(AVG(is_immediate) * 100, 2) immediate_percentage
12 FROM CustomerToIsImmediate;

```

Testcase Test Result

Accepted Runtime: 158 ms

Case 1

Input

Delivery =

delivery_id	customer_id	order_date	customer_pref_delivery_date
1	1	2019-08-01	2019-08-02
2	2	2019-08-02	2019-08-02
3	1	2019-08-11	2019-08-12
4	3	2019-08-24	2019-08-24
5	3	2019-08-21	2019-08-22
6	2	2019-08-11	2019-08-13

1193. Monthly Transactions I

1193. Monthly Transactions I

Medium Topics Companies

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:

Input:
Transactions table:

id	country	state	amount	trans_date
121	US	approved	1000	2018-12-18

68 Online

Code

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

Saved Ln 9, Col 15

Testcase 1 Test Result

Accepted Runtime: 115 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
123	US	approved	2000	2019-01-01
124	DE	approved	2000	2019-01-07

Output

month	country	trans_count	approved_count	trans_total_amount	approved_total_amount
2018-12	US	2	1	3000	1000

1211. Queries Quality and Percentage

1211. Queries Quality and Percentage

Easy Topics Companies

SQL Schema > Pandas Schema >
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.

795 Online

Code

```
MySQL
1 SELECT
2   query_name,
3   ROUND(AVG(rating / position), 2) AS quality,
4   ROUND(AVG(rating < 3), 2) AS poor_query_percentage
5 FROM Queries
6 GROUP BY 1;
```

Saved Ln 6, Col 12

Testcase 1 Test Result

Accepted Runtime: 190 ms

Case 1

Input

Queries =

query_name	result	position	rating
Dog	Golden Retriever	1	5
Dog	German Shepherd	2	5
Dog	Mule	200	1
Cat	Shirazi	5	2
Cat	Siamese	3	3
Cat	Sphinx	7	4

Output

query_name	quality	poor_query_percentage
------------	---------	-----------------------

1633. Percentage of Users Attended a Contest

1633. Percentage of Users Attended a Contest

Easy Topics Companies

SQL Schema > Pandas Schema >

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.

850 133 72 Online

```

1 SELECT
2   contest_id,
3   ROUND(
4     COUNT(user_id) * 100 / (
5       SELECT COUNT(*)
6     FROM Users
7   ),
8   2) AS percentage
9 FROM Register
10 GROUP BY 1
11 ORDER BY percentage DESC, contest_id;

```

Testcase Test Result

Accepted Runtime: 251 ms

Case 1

Input

Users =

user_id	user_name
6	Alice
2	Bob
7	Alex

Register =

contest_id	user_id
215	6
289	2

1075. Project Employees I

1075. Project Employees I

Easy Topics Companies

SQL Schema > Pandas Schema >

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**. Return the result table in **any order**. The query result format is in the following example.

749 99 51 Online

```

1 SELECT
2   Project.project_id,
3   ROUND(AVG(Employee.experience_years), 2) AS average_years
4 FROM Project
5 INNER JOIN Employee
6   USING (employee_id)
7 GROUP BY 1;

```

Testcase Test Result

Accepted Runtime: 105 ms

Case 1

Input

Project =

project_id	employee_id
1	1
1	2
1	3
2	1
2	4

Employee =

employee_id	name	experience_years
1		
2		
3		
4		

1251. Average Selling Price

1251. Average Selling Price

SQL Schema > Pandas Schema > 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. If a product does not have any sales, the average price should be null.

1.4K 294 95 Online

```

1 SELECT
2   Prices.product_id,
3   IFNULL(
4     ROUND(
5       SUM(Prices.price * UnitsSold.units) / SUM(UnitsSold.units),
6       2
7     ),
8     0
9   ) As average_price
10 FROM Prices
11 LEFT JOIN UnitsSold
12   ON (
13     Prices.product_id = UnitsSold.product_id
14     AND UnitsSold.purchase_date BETWEEN Prices.start_date AND Prices.end_date)
15 GROUP BY 1;

```

Testcase: Test Result

Accepted Runtime: 167 ms

Case 1

Input

Prices =

product_id	start_date	end_date	price
1	2019-02-17	2019-02-28	5
1	2019-03-01	2019-03-22	20
2	2019-02-01	2019-02-20	15
2	2019-02-21	2019-03-31	30

UnitsSold =

product_id	purchase_date	units
1	2019-02-25	100
1	2019-03-01	10

620. Not Boring Movies

620. Not Boring Movies

SQL Schema > Pandas Schema > 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.
The result format is in the following example.

Example 1:

Input:

Cinema table:

id	movie	description	rating
1	War	great 3D	8.9
2	Science	fiction	8.5
3	Irish	boring	6.2

Output:

id	movie	description	rating
1	War	great 3D	8.9
3	Irish	boring	6.2

1.3K 139 66 Online

```

1 SELECT *
2 FROM Cinema
3 WHERE
4   MOD(id, 2) = 1
5   AND description != 'boring'
6 ORDER BY rating DESC;

```

Testcase: Test Result

Accepted Runtime: 168 ms

Case 1

Input

cinema =

id	movie	description	rating
1	War	great 3D	8.9
2	Science	fiction	8.5
3	Irish	boring	6.2
4	Ice song	Fantasy	8.6
5	House card	Interesting	9.1

Output

id	movie	description	rating
1	War	great 3D	8.9
3	Irish	boring	6.2