

**SQL questions all** 1245 minutes

Question - 1

**SQL: Detecting Potential Payment Fraud in an Online Marketplace**

The company needs a report identifying users who have failed transactions using different payment methods. Failed transactions have "Failed" in the *status* field.

The result should have the following columns: *user\_id | failed\_transactions | distinct\_payment\_methods*. *user\_id* – User attempting multiple failed transactions.

*failed\_transactions* – Total number of failed transactions. *distinct\_payment\_methods* – Total number of unique payment methods used.

# Note:

Only users who have made more than 5 failed transactions in the entire dataset should be included in the report. Row order does not matter.

**Schema**

transactions

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraint** | **Description** |
| transaction\_id | INT | PRIMARY KEY | Unique identifier for a transaction |
| user\_id | INT |  | User attempting the payment |
| payment\_method | VARCHAR(255) |  | Payment method used |
| amount | DECIMAL(10,2) |  | Transaction amount |
| transaction\_date | DATE |  | Date of the transaction |
| status | VARCHAR(255) |  | Status of transaction |

**Sample Data Tables**

transactions

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **transaction\_id** | **user\_id** | **payment\_method** | **amount** | **transaction\_date** | **status** |
| 101 | 202 | Credit Card | 200.43 | 2025-02-16 | Completed |
| 102 | 203 | Netbanking | 3233.10 | 2025-03-11 | Failed |
| 103 | 203 | Netbanking | 1195.35 | 2025-02-24 | Failed |
| 104 | 203 | Debit Card | 376.11 | 2025-03-10 | Failed |
| 105 | 203 | Netbanking | 112.01 | 2025-04-04 | Failed |
| 106 | 203 | Credit Card | 111.1 | 2025-09-12 | Failed |
| 107 | 203 | Debit Card | 2344.5 | 2025-10-03 | Failed |

# Sample Output

*user\_id failed\_transactions distinct\_payment\_methods*

203 6 3

**Explanation**

The user with *user\_id* 203 attempted 6 distinct transactions, which have the status as "Failed", thus it is a potential case of fraud.

Question - 2

**SQL: Average Response Time**

A customer support team wants to analyze response times for resolving tickets to identify performance metrics and improve service quality. The goal is to generate a report calculating the average response time for successfully resolved customer support tickets. Resolved tickets have a value in

the *resolved\_at* field.

The result should have the following columns: *average\_response\_time*.

*average\_response\_time* - The average time between c*reated\_at* and r*esolved\_at*, calculated in hours and set to two decimal places, including trailing zeros if necessary (e.g., 5.00).

Schema

support\_tickets

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraint** | **Description** |
| id | INT | PRIMARY KEY | Unique identifier for each support ticket |
| customer\_Id | INT |  | Reference to the customer who created the ticket |
| created\_at | VARCHAR(19) |  | Date and time when the ticket was created |
| resolved\_at | VARCHAR(19) |  | Date and time when the ticket was resolved |

Sample Data Tables

support\_tickets

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | **customer\_id** | **created\_at** | **resolved\_at** |
| 1 | 1 | 2023-12-21 05:42:00 | 2024-01-01 05:42:00 |
| 2 | 2 | 2023-07-08 14:22:00 | NULL |
| 3 | 3 | 2023-05-22 08:54:00 | 2023-06-17 08:54:00 |

# Sample Output

*average\_response\_time*

444.00

**Explanation**

The sample output shows the average response time for resolved tickets to be 444.00, excluding Ticket *id* 2 since it has not been resolved.

Question - 3

**SQL: Highest-Spending Customers per City**

A retail company wants to identify the highest-spending customer in each city to target them for personalized marketing campaigns and loyalty programs. The goal is to generate a report highlighting the top customer by total spending in each city location. Row order does not matter.

The result should have the following columns: *customer\_id | name | city | total\_spending*. *customer\_id* - Unique identifier for the customer.

*name* - Name of the customer.

*city* - The city where the customer is located.

*total\_spending* - The total spending of the customer, calculated by summing all order amounts for each customer, and should be converted to an integer by rounding down using an appropriate function, e.g., 1.99 rounds to 1.

# Note:

Only include the highest-spending customer(s) from each city based on their total spending. That is, if the maximum highest amount spent in a city is 100, include all customers in that city that spent 100.

Schema

customers

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraint** | **Description** |
| id | INT | PRIMARY KEY | Unique identifier for a customer |
| name | VARCHAR(255) |  | Name of the customer |
| city | VARCHAR(255) |  | City where the customer is located |

orders

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraint** | **Description** |
| id | INT | PRIMARY KEY | Unique identifier for an order |
| customer\_id | INT | FOREIGN KEY(customer\_id => customers.id) | Reference to the customer |
| amount | DECIMAL(10,2) |  | Total amount of the order |

Sample Data Tables

customers

|  |  |  |
| --- | --- | --- |
| **id** | **name** | **city** |
| 1 | Customer 1 | Los Angeles |
| 2 | Customer 2 | Chicago |
| 3 | Customer 3 | Chicago |

orders

|  |  |  |
| --- | --- | --- |
| **id** | **customer\_id** | **amount** |
| 1 | 1 | 150.75 |
| 2 | 2 | 230.50 |
| 3 | 3 | 345.25 |

# Sample Output

*customer\_id* n*ame*

1 Customer 1

3 Customer 3

*city*

Los Angeles Chicago

*total\_spending*

150

345

**Explanation**

The sample output shows the highest-spending customers in each city with their customer ID, name, city, and total spending. For "Los Angeles", "Customer 1" has the highest spending of 150. In "Chicago", "Customer 2" has a total spending of 230, but "Chicago" top spender is "Customer 3", with a total of 345.

Question - 4

**SQL: E-commerce Product Request Report**

An e-commerce platform maintains a database to track its products and customer requests for these products. The task is to generate a report that lists each available product's name and the total number of requests received for it.

The result should have the following columns: *product\_name | total\_requests*. *product\_name* - the name of the available product.

*total\_requests* - the total number of requests received for the product.

The result should be sorted in descending order based on *total\_requests*, and in case of a tie, by *product\_name* in ascending order. **Note:**

Only products that are currently available should be included in the report.

The *is\_available* field in the products table indicates whether a product is available (1 for available, 0 for not available).

Schema

products

|  |  |  |  |
| --- | --- | --- | --- |
| name | type | constraints | description |
| id | INT | NOT NULL PRIMARY KEY | The identifier of the product |
| name | VARCHAR(255) |  | The name of the product |
| category | VARCHAR(255) |  | The category of the product |
| is\_available | SMALLINT |  | The flag indicating if the product is available |

requests

|  |  |  |  |
| --- | --- | --- | --- |
| name | type | constraints | description |
| product\_id | INT | FOREIGN KEY REFERENCES products(id) | The reference to the product |
| client\_email | VARCHAR(255) |  | The email address of the client |

Sample Data Tables

products

|  |  |  |  |
| --- | --- | --- | --- |
| id | name | category | is\_available |
| 1 | PromoPro | beauty products | 1 |
| 2 | AdVantage | outdoor gear | 1 |
| 3 | MarketMagnet | sports equipment | 1 |
| 5 | AdBlitz | beauty products | 0 |

requests

|  |  |
| --- | --- |
| product\_id | client\_email |
| 1 | [salgate1@fc2.com](mailto:salgate1@fc2.com) |
| 1 | [lwycliff6@list-manage.com](mailto:lwycliff6@list-manage.com) |
| 1 | [ekimbleyf@scientificamerican.com](mailto:ekimbleyf@scientificamerican.com) |
| 2 | [bgooro@spotify.com](mailto:bgooro@spotify.com) |
| 2 | [vsamwayest@bbb.org](mailto:vsamwayest@bbb.org) |
| 3 | [apappin0@yellowbook.com](mailto:apappin0@yellowbook.com) |
| 3 | [ringreyb@businessinsider.com](mailto:ringreyb@businessinsider.com) |
| 3 | [mrysonm@istockphoto.com](mailto:mrysonm@istockphoto.com) |
| 5 | [ayushin1c@opera.com](mailto:ayushin1c@opera.com) |
| 5 | [bcoulston1q@hubpages.com](mailto:bcoulston1q@hubpages.com) |

# Sample Output

MarketMagnet 3

PromoPro 3

AdVantage 2

**Explanation**

The sample output lists the available products with their total request counts. 'PromoPro' and 'MarketMagnet' both have 3 requests, but 'MarketMagnet' appears first due to alphabetical ordering. 'AdVantage' has 2 requests and is listed after the others.

Question - 5

**SQL: Active Campaign Engagement Report**

A marketing company maintains a database to track its advertising campaigns and engagements. The task is to generate a report that includes the name of each active campaign, the total number of engagements, and the sum of views and clicks for those engagements.

The result should have the following columns: *campaign\_name | total\_engagements | total\_views\_and\_clicks*. *campaign\_name* - the name of the active campaign

*total\_engagements* - the number of engagements *total\_views\_and\_clicks* - the combined number of views and clicks

The result should be sorted in ascending order by *campaign\_name*.

# Note:

Only active campaigns should be included in the report.

The *is\_active* field in the campaigns table indicates whether a campaign is active (1 for active, 0 for inactive).

Schema

campaigns

|  |  |  |  |
| --- | --- | --- | --- |
| name | type | constraints | description |
| id | INT | NOT NULL PRIMARY KEY | The identifier of the campaign |
| name | VARCHAR(255) |  | The name of the campaign |
| is\_active | SMALLINT |  | The flag indicates if the campaign is active |

engagements

|  |  |  |  |
| --- | --- | --- | --- |
| name | type | constraints | description |
| campaign\_id | INT | FOREIGN KEY REFERENCES campaigns(id) | The reference to the campaign |
| views | INT |  | The total number of views of the engagement |
| clicks | INT |  | The total number of clicks of the engagement |

Sample Data Tables

campaigns

|  |  |  |
| --- | --- | --- |
| id | name | is\_active |
| 1 | SummerSavings | 1 |
| 2 | FallFrenzy | 1 |
| 3 | WinterWonderland | 0 |

engagements

|  |  |  |
| --- | --- | --- |
| campaign\_id | views | clicks |
| 1 | 100 | 10 |
| 1 | 150 | 20 |
| 2 | 200 | 30 |
| 2 | 250 | 40 |
| 3 | 300 | 50 |
| 1 | 120 | 15 |
| 2 | 180 | 25 |
| 3 | 220 | 35 |
| 1 | 130 | 18 |
| 2 | 210 | 28 |

# Sample Output

FallFrenzy

3 963

SummerSavings 4 443

**Explanation**

The output includes only the active campaigns. For each active campaign, the total number of engagements is calculated by counting the number of records in the engagements table for that campaign. The number of views and clicks is calculated by summing (views + clicks) for each engagement related

to the campaign. The campaigns are then sorted by their names in ascending order.

Question - 6

**SQL: Tax Report Summary**

An online tax reporting application needs a summary report of account activity. It should list each account's email and the sum of reported amounts during 2023.

The result should have the following columns: *email | total\_report\_amount*. *email* - the email address of the account.

*total\_report\_amount* - the sum of reported amounts submitted in 2023, rounded to two decimal places.

The results should be sorted in ascending order by *email*. **Note:**

Only reports submitted in the year 2023 should be included in the report.

Ensure all decimal values are formatted to include trailing zeros if necessary (e.g., 5.00).

Schema

accounts

|  |  |  |  |
| --- | --- | --- | --- |
| name | type | constraints | description |
| id | INT | NOT NULL PRIMARY KEY | The identifier of the account |
| email | VARCHAR(255) |  | The email address of the account |

reports

|  |  |  |  |
| --- | --- | --- | --- |
| name | type | constraints | description |
| account\_id | INT | FOREIGN KEY REFERENCES accounts(id) | The reference to the account |
| dt | VARCHAR(19) |  | The date and time of report |
| amount | DECIMAL(6, 2) |  | The reported amount |

Sample Data Tables

accounts

|  |  |
| --- | --- |
| id | email |
| 1 | [hratke0@disqus.com](mailto:hratke0@disqus.com) |
| 2 | [lcaiger1@si.edu](mailto:lcaiger1@si.edu) |
| 3 | [gburkett2@vinaora.com](mailto:gburkett2@vinaora.com) |

reports

|  |  |  |
| --- | --- | --- |
| account\_id | dt | amount |
| 1 | 2023-05-27 01:46:19 | 830.45 |
| 2 | 2023-01-15 09:23:21 | 2518.18 |
| 3 | 2023-05-08 01:44:41 | 4637.39 |
| 1 | 2023-06-30 15:02:03 | 3953.69 |

|  |  |  |
| --- | --- | --- |
| 2 | 2023-12-05 04:39:31 | 3357.99 |
| 3 | 2023-02-03 09:41:00 | 1907.38 |
| 1 | 2022-12-30 04:05:57 | 1217.29 |
| 2 | 2024-01-24 14:18:07 | 2441.66 |
| 3 | 2024-01-05 23:19:31 | 3055.2 |
| 1 | 2023-05-26 01:54:24 | 2077.36 |

# Sample Output

|  |  |
| --- | --- |
| [gburkett2@vinaora.com](mailto:gburkett2@vinaora.com) | 6544.77 |
| [hratke0@disqus.com](mailto:hratke0@disqus.com) | 6861.50 |
| [lcaiger1@si.edu](mailto:lcaiger1@si.edu) | 5876.17 |

**Explanation**

The sample output shows the total amount of reports for each account email in 2023. The amounts are summed and rounded to two decimal places. The output is sorted by the account email in ascending order.

Question - 7

**SQL: Antivirus Device Scan Report**

An antivirus software company maintains a database to track devices and the files scanned on each device. The task is to generate a report that lists each device's MAC address along with the total number of files scanned and the total number of infected files for that device.

The result should have the following columns: *mac\_address | total\_files\_scanned | total\_infected\_files*. *mac\_address* - the MAC address of the device.

*total\_files\_scanned* - the total number of files scanned on the device. *total\_infected\_files* - the total number of infected files on the device.

The result should be sorted in ascending order by *mac\_address*. **Note:**

The *is\_infected* field in the scanned files table indicates whether a file is infected (1 for infected, 0 for not infected).

Schema

devices

|  |  |  |  |
| --- | --- | --- | --- |
| name | type | constraints | description |
| id | INT | NOT NULL PRIMARY KEY | The identifier of the device |
| mac\_address | VARCHAR(255) |  | The MAC address of the device |

scanned\_files

|  |  |  |  |
| --- | --- | --- | --- |
| name | type | constraints | description |
| device\_id | INT | FOREIGN KEY REFERENCES devices(id) | The reference to the device |
| filename | VARCHAR(255) |  | The name of the file |
| is\_infected | SMALLINT |  | The flag indicating if the file is infected |

Sample Data Tables

devices

|  |  |
| --- | --- |
| id | mac\_address |
| 1 | 66-0F-84-41-B8-8E |
| 2 | A6-1A-2F-3A-7B-83 |
| 3 | 76-CD-24-48-F0-DD |

scanned\_files

|  |  |  |
| --- | --- | --- |
| device\_id | filename | is\_infected |
| 1 | File1.mp3 | 0 |
| 1 | File2.xls | 1 |
| 2 | File3.doc | 0 |
| 2 | File4.ppt | 1 |
| 2 | File5.mp3 | 1 |
| 3 | File6.xls | 0 |
| 3 | File7.doc | 1 |
| 3 | File8.ppt | 0 |
| 3 | File9.mp3 | 1 |
| 3 | File10.xls | 0 |

# Sample Output

|  |  |  |
| --- | --- | --- |
| 66-0F-84-41-B8-8E | 2 | 1 |
| A6-1A-2F-3A-7B-83 | 3 | 2 |
| 76-CD-24-48-F0-DD | 5 | 2 |

**Explanation**

The report shows the MAC address of each device along with the total number of files scanned and the total number of infected files. For example, the device with MAC address '66-0F-84-41-B8-8E' has 2 files scanned, out of which 1 is infected.

Question - 8

**SQL: Cryptocurrency Transactions Report**

In the cryptocurrency market, a database engineer is tasked with generating a report for all cryptocurrency coins and their associated transactions. The report should include the name of each coin, the total amount of transactions, and the total number of transactions for each coin in the year 2023.

The result should have the following columns: *coin\_name | total\_transaction\_amount | total\_transactions*. *coin\_name* - the name of the cryptocurrency coin.

*total\_transaction\_amount* - the sum of transaction amounts in 2023, rounded to two decimal places. *total\_transactions* - the total number of transactions in 2023

The result should be sorted in ascending order by *coin\_name*.

# Note:

Only transactions that occurred in 2023 should be included in the report.

Ensure decimal values are formatted to include trailing zeros if necessary, e.g., 5.00.

Schema

coins

|  |  |  |  |
| --- | --- | --- | --- |
| name | type | constraints | description |
| id | INT | NOT NULL PRIMARY KEY | The identifier of the cryptocurrency coin |
| name | VARCHAR(255) |  | The name of the cryptocurrency coin |

transactions

|  |  |  |  |
| --- | --- | --- | --- |
| name | type | constraints | description |
| coin\_id | INT | FOREIGN KEY REFERENCES coins(id) | The reference to the cryptocurrency coin |
| dt | VARCHAR(19) |  | The date and time of the transaction |
| amount | DECIMAL(5, 2) |  | The amount of the transaction |

Sample Data Tables

coins

|  |  |
| --- | --- |
| id | name |
| 1 | BitCash |
| 2 | Etherium |
| 3 | Litecoin |

transactions

|  |  |  |
| --- | --- | --- |
| coin\_id | dt | amount |
| 1 | 2023-07-03 12:16:53 | 34.32 |
| 1 | 2023-12-08 12:14:58 | 47.59 |
| 2 | 2022-12-16 20:42:10 | 45.54 |
| 2 | 2023-11-05 09:27:11 | 53.3 |
| 3 | 2023-12-05 06:45:23 | 71.51 |
| 3 | 2023-01-19 01:43:25 | 97.18 |
| 3 | 2024-01-24 13:34:00 | 86.68 |
| 1 | 2023-05-07 05:30:06 | 25.6 |
| 2 | 2023-03-08 08:07:20 | 40.11 |
| 3 | 2023-08-13 10:44:54 | 87.54 |

# Sample Output

|  |  |  |
| --- | --- | --- |
| BitCash | 107.51 | 3 |
| Etherium | 93.41 | 2 |
| Litecoin | 256.23 | 3 |

**Explanation**

The sample output shows the total transaction amount and count for each coin in the year 2023. For 'BitCash', there are three transactions totaling 107.51. 'Etherium' has two transactions totaling 93.41. 'Litecoin' has three transactions totaling 256.23. All amounts are rounded to two decimal places.

A domain hosting company maintains a database to manage its customers and the domains they own. The task is to generate a report that lists each customer's email address along with the total number of domains they own.

Question - 9

**SQL: Customer Domain Ownership Report**

The result should have the following columns: *email | total\_domains*. *email* - the email address of the customer.

*total\_domains* - the total number of domains owned by the customer.

The result should be sorted in ascending order by *email*.

Schema

customers

|  |  |  |  |
| --- | --- | --- | --- |
| name | type | constraints | description |
| id | INT | NOT NULL PRIMARY KEY | The identifier of the customer |
| email | VARCHAR(255) |  | The email address of the customer |

domains

|  |  |  |  |
| --- | --- | --- | --- |
| name | type | constraints | description |
| customer\_id | INT | FOREIGN KEY REFERENCES customers(id) | The reference to the customer |
| name | VARCHAR(255) |  | The name of the domain |

Sample Data Tables

customers

|  |  |
| --- | --- |
| id | email |
| 1 | [ebayldon0@washingtonpost.com](mailto:ebayldon0@washingtonpost.com) |
| 2 | [agammade1@comcast.net](mailto:agammade1@comcast.net) |
| 3 | [goloshkin2@reference.com](mailto:goloshkin2@reference.com) |
| 4 | [cantonescu3@earthlink.net](mailto:cantonescu3@earthlink.net) |
| 5 | [fparzis4@ow.ly](mailto:fparzis4@ow.ly) |
| 6 | [cpetroulis5@shutterfly.com](mailto:cpetroulis5@shutterfly.com) |
| 7 | [tbeels6@bbb.org](mailto:tbeels6@bbb.org) |
| 8 | [zmacturlough7@4shared.com](mailto:zmacturlough7@4shared.com) |
| 9 | [eshury8@skype.com](mailto:eshury8@skype.com) |
| 10 | [jfehners9@github.io](mailto:jfehners9@github.io) |

domains

|  |  |
| --- | --- |
| customer\_id | name |
| 1 | bfilipa.net |

|  |  |
| --- | --- |
| 1 | gsparsholti.net |
| 1 | jhughsr.org |
| 2 | scopas8.net |
| 2 | cglison1u.org |
| 3 | tginiz.com |
| 3 | arubinowitsch2l.net |
| 3 | clockyear2m.org |
| 4 | sfinnigand.com |
| 4 | vborrelt.net |

# Sample Output

[agammade1@comcast.net](mailto:agammade1@comcast.net) 2

[cantonescu3@earthlink.net](mailto:cantonescu3@earthlink.net) 2

[ebayldon0@washingtonpost.com](mailto:ebayldon0@washingtonpost.com) 3

[goloshkin2@reference.com](mailto:goloshkin2@reference.com) 3

**Explanation**

The output lists the email addresses of customers along with the total number of domains they own. For instance, the customer with the email ['ebayldon0@washingtonpost.com'](mailto:%27ebayldon0@washingtonpost.com) owns 3 domains, while ['agammade1@comcast.net'](mailto:%27agammade1@comcast.net) owns 2 domains. The result is sorted by email addresses in ascending order.

Question - 10

**SQL: E-commerce Wishlist Report**

Generate a report from an e-commerce database that lists the product names and prices, along with the total number of times each is on a wishlist. The result should have the following columns: *product\_name | price | total\_wishlist\_count*.

*product\_name* - the name of the product.

*price* - its price

*total\_wishlist\_count* - the total number of times it appears in wishlists

The result should be sorted in ascending order by *product\_name*. **Note:**

Only include products that are currently in stock.

Ensure all decimal values are formatted to include trailing zeros if necessary, e.g., 5.00.

Schema

products

|  |  |  |  |
| --- | --- | --- | --- |
| name | type | constraints | description |
| id | INT | NOT NULL PRIMARY KEY | The identifier of the product |
| name | VARCHAR(255) |  | The name of the product |
| price | DECIMAL(6, 2) |  | The price of the product |
| in\_stock | SMALLINT |  | 1 indicates 'in stock', 0 indicates 'out of stock' |

wishlists

|  |  |  |  |
| --- | --- | --- | --- |
| name | type | constraints | description |
| product\_id | INT | FOREIGN KEY REFERENCES products(id) | The reference to the product |
| customer\_email | VARCHAR(255) |  | The email address of the customer |

Sample Data Tables

products

|  |  |  |  |
| --- | --- | --- | --- |
| id | name | price | in\_stock |
| 1 | TechGadget Pro X | 324.24 | 1 |
| 2 | LuxuryHome Decor Set | 884.9 | 1 |
| 3 | FitnessTracker Elite | 698.59 | 0 |

wishlists

|  |  |
| --- | --- |
| product\_id | customer\_email |
| 1 | [user1@example.com](mailto:user1@example.com) |
| 1 | [user2@example.com](mailto:user2@example.com) |
| 2 | [user3@example.com](mailto:user3@example.com) |
| 2 | [user4@example.com](mailto:user4@example.com) |
| 2 | [user5@example.com](mailto:user5@example.com) |
| 3 | [user6@example.com](mailto:user6@example.com) |
| 1 | [user7@example.com](mailto:user7@example.com) |
| 2 | [user8@example.com](mailto:user8@example.com) |
| 1 | [user9@example.com](mailto:user9@example.com) |
| 3 | [user10@example.com](mailto:user10@example.com) |

# Sample Output

LuxuryHome Decor Set 884.9 3

TechGadget Pro X 324.24 4

**Explanation**

The report includes only products that are in stock. 'TechGadget Pro X' is in stock and appears 4 times in wishlists, while 'LuxuryHome Decor Set' is also in stock and appears 3 times. 'FitnessTracker Elite' is not included in the report as it is not in stock.

Question - 11

**SQL: Email Campaign Report**

An email campaign tracking platform maintains data on various campaigns and their email statistics. The task is to generate a report that lists each campaign's name along with the total number of emails sent, emails opened, and emails not opened.

The result should have the following columns: *campaign\_name | total\_emails\_sent | total\_emails\_opened | total\_emails\_not\_opened*. *campaign\_name* - the name of the email campaign.

*total\_emails\_sent* - the total number of emails sent in the campaign.

*total\_emails\_opened* - the total number of emails opened in the campaign.

*total\_emails\_not\_opened* - the total number of emails not opened.

The result should be sorted in ascending order by *campaign\_name*.

# Note:

The number of emails not opened is calculated as the difference between the total emails sent and the emails opened.

Schema

campaigns

|  |  |  |  |
| --- | --- | --- | --- |
| name | type | constraints | description |
| id | INT | NOT NULL PRIMARY KEY | The identifier of the email campaign |
| name | VARCHAR(255) |  | The name of the email campaign |

email\_stats

|  |  |  |  |
| --- | --- | --- | --- |
| name | type | constraints | description |
| campaign\_id | INT | FOREIGN KEY REFERENCES campaigns(id) | The reference to the email campaign |
| emails\_sent | INT |  | The number of emails sent in the email campaign |
| emails\_opened | INT |  | The number of emails opened in the email campaign |

Sample Data Tables

campaigns

|  |  |
| --- | --- |
| id | name |
| 1 | SummerSale2021 |
| 2 | FallPromo |
| 3 | WinterWonderland |

email\_stats

|  |  |  |
| --- | --- | --- |
| campaign\_id | emails\_sent | emails\_opened |
| 1 | 1000 | 800 |
| 2 | 1500 | 1200 |
| 3 | 2000 | 1800 |
| 1 | 500 | 300 |
| 2 | 700 | 500 |
| 3 | 800 | 600 |
| 1 | 300 | 200 |
| 2 | 400 | 300 |
| 3 | 600 | 500 |
| 3 | 400 | 300 |

# Sample Output

|  |  |  |  |
| --- | --- | --- | --- |
| FallPromo | 2600 | 2000 | 600 |
| SummerSale2021 | 1800 | 1300 | 500 |
| WinterWonderland | 3800 | 3200 | 600 |

**Explanation**

The report shows each campaign's name along with the total emails sent, opened, and not opened. For example, the 'FallPromo' campaign had a total of 2600 emails sent, 2000 emails opened, and 600 emails not opened, calculated as the difference between sent and opened.

Question - 12

**SQL: Auction Lot Offers Report**

In an e-commerce auction platform, the task is to generate a report that provides insights into the bidding activities on various lots. The report should list each lot's name, the highest offer made for that lot, and the total number of offers received.

The result should have the following columns: *lot\_name | highest\_offer | total\_offers*. *lot\_name* - the name of the lot.

*highest\_offer* - the highest offer made for the lot.

*total\_offers* - the total number of offers received for the lot.

The result should be sorted in ascending order by *lot\_name*. **Note:**

Ensure all decimal values are formatted to include trailing zeros if necessary (e.g., 5.00).

Schema

lots

|  |  |  |  |
| --- | --- | --- | --- |
| name | type | constraints | description |
| id | INT | NOT NULL PRIMARY KEY | The identifier of the lot |
| name | VARCHAR(255) |  | The name of the lot |

offers

|  |  |  |  |
| --- | --- | --- | --- |
| name | type | constraints | description |
| lot\_id | INT | FOREIGN KEY REFERENCES lots(id) | The identifier of the lot for which the offer is made |
| amount | DECIMAL(6, 2) |  | The amount of the offer |

Sample Data Tables

lots

|  |  |
| --- | --- |
| id | name |
| 1 | Acacia parramattensis Tindale |
| 2 | Poa arctica R. Br. ssp. aperta (Scribn. & Merr.) Soreng |
| 3 | Calophyllum inophyllum L. |

offers

|  |  |
| --- | --- |
| lot\_id | amount |
| 1 | 260.91 |
| 1 | 802.83 |

|  |  |
| --- | --- |
| 1 | 986.78 |
| 2 | 814.57 |
| 2 | 999.06 |
| 2 | 414.67 |
| 3 | 200.41 |
| 3 | 593.07 |
| 3 | 701.88 |
| 3 | 972.87 |

# Sample Output

|  |  |  |
| --- | --- | --- |
| Acacia parramattensis Tindale | 986.78 | 3 |
| Calophyllum inophyllum L. | 972.87 | 4 |
| Poa arctica R. Br. ssp. aperta | (Scribn. & Merr.) Soreng 999.06 | 3 |

**Explanation**

The sample output shows the highest offer and the total number of offers for each lot. For example, the lot 'Acacia parramattensis Tindale' received a maximum offer of 986.78 and a total of 3 offers. The decimal values are rounded to two decimal places.

Question - 13

**SQL: Online Banking Transactions Report**

A financial services company needs a report of account transactions for September 2022.

For each account, the report should have the following columns: *IBAN | min\_transaction | max\_transaction | avg\_transaction | total\_transactions*.

*IBAN* - the account's International Bank Account Number *min\_transaction* - the minimum transaction amount *max\_transaction* - the maximum transaction amount *avg\_transaction* - the average transaction amount *total\_transactions* - the total number of transactions

The results should be sorted in ascending order by *IBAN* to facilitate easy reference.

# Note:

Only transactions from September 2022 should be included in the report.

Ensure all decimal values are formatted to include trailing zeros if necessary, e.g., 5.00.

Schema

accounts

|  |  |  |  |
| --- | --- | --- | --- |
| name | type | constraints | description |
| id | INT | NOT NULL PRIMARY KEY | The identifier of the account |
| iban | VARCHAR(255) |  | The IBAN of the account |

transactions

|  |  |  |  |
| --- | --- | --- | --- |
| name | type | constraints | description |
| account\_id | INT | FOREIGN KEY REFERENCES accounts(id) | The identifier of the account related to the transaction |

|  |  |  |  |
| --- | --- | --- | --- |
| dt | CHAR(19) |  | The date and time of the transaction |
| amount | DECIMAL(5, 2) |  | The amount of the transaction |

Sample Data Tables

accounts

|  |  |
| --- | --- |
| id | iban |
| 1 | BG40 RFFX 4898 53DD CZD6 KQ |
| 2 | PT42 5267 0592 8451 8001 2180 3 |
| 3 | FR96 8758 8909 81LR DJ71 ERKN D56 |

transactions

|  |  |  |
| --- | --- | --- |
| account\_id | dt | amount |
| 1 | 2022-09-02 06:33:39 | 33.31 |
| 1 | 2022-09-20 08:14:39 | 31.77 |
| 1 | 2022-09-25 06:41:45 | 72.84 |
| 2 | 2022-09-04 22:28:12 | 35.26 |
| 2 | 2022-09-17 07:57:29 | 33.27 |
| 2 | 2022-09-27 22:30:36 | 70.78 |
| 3 | 2022-09-16 21:54:12 | 75.04 |
| 3 | 2022-09-19 18:27:39 | 71.19 |
| 3 | 2022-09-28 01:38:56 | 14.34 |
| 3 | 2022-08-30 01:35:31 | 69.19 |

# Sample Output

BG40 RFFX 4898 53DD CZD6 KQ 31.77 72.84 45.97 3

PT42 5267 0592 8451 8001 2180 3 33.27 70.78 46.44 3

FR96 8758 8909 81LR DJ71 ERKN D56 14.34 75.04 53.52 3

**Explanation**

The sample output shows the IBAN for each account along with the minimum, maximum, and average transaction amounts for September 2022, rounded to two decimal places. It also includes the total number of transactions for each account during that month.

Question - 14

**SQL: Top Wishlist Products Summary**

In the competitive e-commerce landscape, understanding customer preferences and product popularity is crucial for inventory management and marketing strategies. A development team is creating a report that identifies the top 3 products in customers' wishlists. This report will highlight in-stock products, allowing immediate action to capitalize on customer interest.

The result should have the following columns: *name | price | total\_wishes*. *name* - the name of the product

*price* - the price of the product

*total\_wishes* - the total number of times the product appears in customers' wishlists

The results should be sorted in descending order by *total\_wishes* to easily identify the most desired products, then in ascending order by *name*. The result should be limited to the top 3 products.

# Note:

Only products that are in stock should be included.

**Schema**

products

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| id | INT | PRIMARY KEY | The identifier of the product |
| name | VARCHAR(255) |  | The name of the product |
| price | DECIMAL(6,2) |  | The price of the product |
| in\_stock | SMALLINT |  | The flag indicating if the product in stock |

wishlists

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| product\_id | INT | FOREIGN KEY(product\_id => products.id) | The reference to the product |
| customer\_email | VARCHAR(255) |  | The email address of the customer |

**Sample Data Tables**

products

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | **name** | **price** | **in\_stock** |
| 1 | TechGadget Pro X | 274.80 | 1 |
| 2 | LuxuryHome Decor Set | 262.84 | 1 |
| 3 | FitnessTracker Elite | 637.92 | 0 |
| 4 | GourmetCookware Set | 535.34 | 1 |
| 5 | Fashionista Wardrobe Collection | 525.44 | 1 |

wishlists

|  |  |
| --- | --- |
| **product\_id** | **customer\_email** |
| 1 | [crabbec@redcross.org](mailto:crabbec@redcross.org) |
| 1 | [efindlow2@tinypic.com](mailto:efindlow2@tinypic.com) |
| 1 | [jmachoste5@issuu.com](mailto:jmachoste5@issuu.com) |
| 1 | [nselle@simplemachines.org](mailto:nselle@simplemachines.org) |
| 2 | [aonn1@ebay.co.uk](mailto:aonn1@ebay.co.uk) |
| 2 | [bbolton0@google.cn](mailto:bbolton0@google.cn) |
| 2 | [ebockett3@storify.com](mailto:ebockett3@storify.com) |

|  |  |
| --- | --- |
| 2 | [fdunguyg@symantec.com](mailto:fdunguyg@symantec.com) |
| 2 | [slowried@cbsnews.com](mailto:slowried@cbsnews.com) |
| 3 | [jgately7@goo.ne.jp](mailto:jgately7@goo.ne.jp) |
| 3 | [ospearettj@bandcamp.com](mailto:ospearettj@bandcamp.com) |
| 3 | [rpanonsb@paypal.com](mailto:rpanonsb@paypal.com) |
| 3 | [ydevauxh@toplist.cz](mailto:ydevauxh@toplist.cz) |
| 3 | [zbabbage9@imageshack.us](mailto:zbabbage9@imageshack.us) |
| 4 | [dpauleya@cnbc.com](mailto:dpauleya@cnbc.com) |
| 4 | [jletterick4@dailymotion.com](mailto:jletterick4@dailymotion.com) |
| 4 | [khunnisett6@princeton.edu](mailto:khunnisett6@princeton.edu) |
| 4 | [rkernellf@uiuc.edu](mailto:rkernellf@uiuc.edu) |
| 5 | [blodin8@wikimedia.org](mailto:blodin8@wikimedia.org) |
| 5 | [lyusupovi@nps.gov](mailto:lyusupovi@nps.gov) |

# Sample Output

+--------------------+------+ +

|name |price |total\_wishes|

+--------------------+------+ +

|LuxuryHome Decor Set|262.84|5

|GourmetCookware Set |535.34|4

|TechGadget Pro X |274.80|4

|

|

|

+--------------------+------+ +

Question - 15

**SQL: E-commerce Customer Purchases Report**

In e-commerce, understanding customer purchasing behavior is essential for tailoring marketing strategies and improving customer service. A development team is creating a query for a report that details customer activity in March 2024. This report will provide insights into the total number and sum of purchases made by each customer, facilitating targeted engagement and promotional efforts.

The result should have the following columns: *email | total\_purchases | total\_purchase\_amount*. *email* - the email address of the customer

*total\_purchases* - the total number of purchases made by the customer

*total\_purchase\_amount* - the total sum of the purchases made by the customer, with two decimal places, including trailing zeros if necessary, e.g., 500.00 The results should be sorted in ascending order by *email*.

# Note:

Only purchases made in March 2024 should be included.

**Schema**

customers

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |

|  |  |  |  |
| --- | --- | --- | --- |
| id | INT | PRIMARY KEY | The identifier of the customer |
| email | VARCHAR(255) |  | The email address of the customer |

purchases

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| customer\_id | INT | FOREIGN KEY(customer\_id => customers.id) | The reference to the customer |
| dt | VARCHAR(19) |  | The date and time of the purchase |
| amount | DECIMAL(6,2) |  | The amount of the purchase |

**Sample Data Tables**

customers

|  |  |
| --- | --- |
| **id** | **email** |
| 1 | [floggie0@newsvine.com](mailto:floggie0@newsvine.com) |
| 2 | [sgillbe1@ca.gov](mailto:sgillbe1@ca.gov) |
| 3 | [jgohn2@elegantthemes.com](mailto:jgohn2@elegantthemes.com) |

purchases

|  |  |  |
| --- | --- | --- |
| **customer\_id** | **dt** | **amount** |
| 2 | 2024-02-21 02:56:12 | 228.58 |
| 2 | 2024-02-23 09:32:47 | 972.41 |
| 1 | 2024-03-14 15:50:13 | 109.16 |
| 1 | 2024-03-17 00:31:44 | 11.49 |
| 1 | 2024-03-17 04:15:42 | 692.64 |
| 2 | 2024-03-01 04:35:09 | 589.74 |
| 2 | 2024-03-13 14:42:23 | 508.75 |
| 2 | 2024-03-17 07:57:36 | 933.91 |
| 2 | 2024-03-19 08:24:38 | 488.26 |
| 2 | 2024-03-31 23:30:54 | 55.07 |
| 3 | 2024-03-03 11:34:30 | 816.67 |
| 3 | 2024-03-08 23:46:07 | 672.93 |
| 3 | 2024-03-15 18:09:56 | 260.66 |
| 3 | 2024-03-20 15:18:11 | 321.07 |
| 3 | 2024-03-20 17:40:35 | 29.06 |
| 3 | 2024-03-20 23:41:39 | 314.85 |
| 3 | 2024-03-25 11:41:07 | 67.12 |
| 1 | 2024-04-05 03:05:10 | 417.78 |

|  |  |  |
| --- | --- | --- |
| 2 | 2024-04-09 08:16:17 | 697.53 |
| 3 | 2024-04-02 07:56:48 | 156.27 |

# Sample Output

+------------------------+---------------+ +

|email |total\_purchases|total\_purchase\_amount|

+------------------------+---------------+ +

[|floggie0@newsvine.com](mailto:floggie0@newsvine.com) |3

[|jgohn2@elegantthemes.com|7](mailto:jgohn2@elegantthemes.com)

[|sgillbe1@ca.gov](mailto:sgillbe1@ca.gov) |5

|813.29

|2482.36

|2575.73

|

|

|

+------------------------+---------------+ +

Question - 16

**SQL: Report on Applicants Pending Consular Service**

In managing consular services, addressing delays caused by system issues is critical. A development team is creating a query for the system's dashboard to generate a report that identifies applicants who have not received service due to delays, emphasizing the extent of these delays. This report is essential for prioritizing these applicants and efficiently managing the backlog.

The result should have the following columns: *email | scheduled\_appointment | days\_of\_delay*. *email* - the email address of the applicant

*scheduled\_appointment* - the date when the appointment was scheduled

*days\_of\_delay* - the number of full days between the scheduled appointment date and the current date, April 10, 2024 The result should be sorted in ascending order by *scheduled\_appointment*, then in ascending order by *email*.

# Note:

Only applicants who have not received service before the current date should be included in the result. The current date is fixed as April 10, 2024.

**Schema**

applicants

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| id | INT | PRIMARY KEY | The identifier of the applicant |
| email | VARCHAR(255) |  | The email address of the applicant |

appointments

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| applicant\_id | INT | FOREIGN KEY(applicant\_id => applicants.id) | The reference to the applicant |
| dt | VARCHAR(19) |  | The date of the scheduled appointment |
| is\_received | BOOLEAN |  | The flag indicating if the service was received |

**Sample Data Tables**

applicants

|  |  |
| --- | --- |
| **id** | **email** |

|  |  |
| --- | --- |
| 1 | [nkienzle0@spiegel.de](mailto:nkienzle0@spiegel.de) |
| 2 | [alaste1@bbc.co.uk](mailto:alaste1@bbc.co.uk) |
| 3 | [jjochanany2@ow.ly](mailto:jjochanany2@ow.ly) |
| 4 | [bsenn3@salon.com](mailto:bsenn3@salon.com) |
| 5 | [bwhittall4@nhs.uk](mailto:bwhittall4@nhs.uk) |

appointments

|  |  |  |
| --- | --- | --- |
| **applicant\_id** | **dt** | **is\_received** |
| 1 | 2024-04-27 | 0 |
| 2 | 2024-04-01 | 0 |
| 3 | 2024-04-15 | 0 |
| 4 | 2024-03-27 | 0 |
| 5 | 2024-03-26 | 1 |

# Sample Output

+----------------------+-----------------------+ +

| email | scheduled\_appointment | days\_of\_delay |

+----------------------+-----------------------+ +

| [bsenn3@salon.com](mailto:bsenn3@salon.com) | 2024-03-27 | 14 |

| [alaste1@bbc.co.uk](mailto:alaste1@bbc.co.uk) | 2024-04-01 | 9 |

+----------------------+-----------------------+ +

Question - 17

**SQL: Weekend Appointments for Consular Services Rescheduling**

In consular services, efficiently managing appointment schedules is key to effective service delivery and resource utilization. A development team is creating a query for the system's dashboard to list all appointments scheduled on weekends (Saturday or Sunday). This initiative aims to identify appointments that need to be rescheduled to a working day.

The result should have the following columns: *email | scheduled\_appointment*. *email* - the email address of the user who made the appointment

*scheduled\_appointment* - the name of the day when the appointment is scheduled

The results should be sorted in ascending order by *email*.

# Note:

Only appointments that are scheduled on weekends (Saturday or Sunday) should be included.

**Schema**

applicants

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| id | INT | PRIMARY KEY | The identifier of the applicant |
| email | VARCHAR(255) |  | The email address of the applicant |

appointments

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| applicant\_id | INT | FOREIGN KEY(applicant\_id => applicants.id) | The reference to the applicant |
| dt | VARCHAR(19) |  | The date and time of the scheduled appointment |

**Sample Data Tables**

applicants

|  |  |
| --- | --- |
| **id** | **email** |
| 1 | [rastlatt0@instagram.com](mailto:rastlatt0@instagram.com) |
| 2 | [gcarmody1@stanford.edu](mailto:gcarmody1@stanford.edu) |
| 3 | [mgreenset2@state.tx.us](mailto:mgreenset2@state.tx.us) |

appointments

|  |  |
| --- | --- |
| **applicant\_id** | **dt** |
| 1 | 2024-05-26 01:36:43 |
| 2 | 2024-05-27 16:30:28 |
| 3 | 2024-05-18 19:28:52 |

# Sample Output

+-----------------------+ +

|email |scheduled\_appointment|

+-----------------------+ +

[|mgreenset2@state.tx.us](mailto:mgreenset2@state.tx.us) |Saturday |

[|rastlatt0@instagram.com|Sunday](mailto:rastlatt0@instagram.com) |

+-----------------------+ +

Question - 18

**SQL: Active Domains Registration by Country with Totals**

In managing a domain hosting service, understanding the distribution of active domain registrations across different countries is crucial for strategic planning and market analysis. A development team is creating a query for the system's dashboard to generate a report that lists countries with active domain registrations and the total number of domains per country.

The results should have the following columns: *country\_name | total\_domains*. *country\_name* - the name of the country from where the domain was registered *total\_domains* - the total number of active domains registered from that country

The results should be sorted in ascending order by *country\_name*.

# Note:

Only active domains should be included in the result.

**Schema**

countries

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| id | INT | PRIMARY KEY | The identifier of the country |
| name | VARCHAR(255) |  | The name of the country |

domains

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| country\_id | INT | FOREIGN KEY(country\_id => countries.id) | The reference to the country |
| name | VARCHAR(255) |  | The name of the domain |
| is\_active | BOOLEAN |  | The status of the domain |

**Sample Data Tables**

countries

|  |  |
| --- | --- |
| **id** | **name** |
| 1 | Azerbaijan |
| 2 | Colombia |
| 3 | China |

domains

|  |  |  |
| --- | --- | --- |
| **country\_id** | **name** | **is\_active** |
| 1 | angelfire.com | 1 |
| 1 | free.fr | 1 |
| 1 | google.cn | 1 |
| 1 | nationalgeographic.com | 1 |
| 1 | ovh.net | 1 |
| 1 | surveymonkey.com | 1 |
| 1 | twitpic.com | 1 |
| 2 | ameblo.jp | 1 |
| 2 | berkeley.edu | 1 |
| 2 | multiply.com | 1 |
| 2 | redcross.org | 1 |
| 2 | sourceforge.net | 1 |
| 3 | hc360.com | 1 |
| 3 | liveinternet.ru | 1 |
| 3 | squidoo.com | 1 |
| 3 | technorati.com | 1 |
| 3 | webnode.com | 1 |

|  |  |  |
| --- | --- | --- |
| 3 | yahoo.co.jp | 1 |
| 1 | 1und1.de | 0 |
| 1 | qq.com | 0 |

# Sample Output

+------------+ +

|country\_name|total\_domains|

+------------+ +

|Azerbaijan |7 |

|China |6 |

|Colombia |5 |

+------------+ +

Question - 19

**SQL: Domain Renewal Overview**

In managing a domain hosting panel, tracking domain renewals is crucial for service continuity and customer satisfaction. A development team is creating a query to generate a report of all domains, including the current date, next renewal date, and days until renewal. This report helps domain owners plan their renewals, ensuring timely actions to maintain their online presence.

The result should have the following columns: *name | today\_date | next\_renewal\_date | days\_until\_renewal*. *name* - the name of the domain

*today\_date* - the current date, April 10, 2024, in the format YYYY-MM-DD

*next\_renewal\_date* - the date when the domain is scheduled for its next renewal

*days\_until\_renewal* - the number of full days from the current date until the domain’s next renewal date The results should be sorted in ascending order by *days\_until\_renewal*, then in ascending order by *name*.

# Note:

The current date is fixed as April 10, 2024.

**Schema**

domains

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| name | VARCHAR(255) |  | The name of the domain |
| next\_renewal\_date | VARCHAR(19) |  | The date when the domain is scheduled for next renewal |

**Sample Data Tables**

domains

|  |  |
| --- | --- |
| **name** | **next\_renewal\_date** |
| wired.com | 2024-06-14 00:10:12 |
| blogger.com | 2024-07-18 05:54:57 |
| com.com | 2024-07-21 02:57:25 |

# Sample Output

+-----------+----------+-----------------+ +

|name |today\_date|next\_renewal\_date|days\_until\_renewal|

+-----------+----------+-----------------+ +

|wired.com |2024 04-10|2024-06-14

|blogger.com|2024 04-10|2024-07-18

|com.com |2024-04-10|2024-07-21

|65

|99

|102

|

|

|

+-----------+----------+-----------------+ +

Question - 20

**SQL: User Transaction Details**

In payment systems, analyzing transaction data provides insights into user behavior and system performance. A development team is creating a query for the system's dashboard to generate a report detailing user transactions for March 2024. This report will show the total number, minimum, maximum, and total amount of transactions for each user. These insights help tailor user engagement strategies and improve the user experience.

The results should have the following columns: *email | total\_transactions | min\_amount | max\_amount | total\_amount*. *email* - the email address of the user

*total\_transactions* - the total number of transactions made by the user during the specified period

*min\_amount* - the minimum transaction amount by the user during the specified period, with two decimal places, including trailing zeros if necessary, e.g., 500.00

*max\_amount* - the maximum transaction amount by the user during the specified period, with two decimal places, including trailing zeros if necessary, e.g., 500.00

*total\_amount* - the total amount of all transactions made by the user during the specified period, with two decimal places, including trailing zeros if necessary, e.g., 500.00

The results should be sorted in ascending order by *email*.

# Note:

Only transactions that occurred in March 2024 should be included.

**Schema**

users

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| id | INT | PRIMARY KEY | The identifier of the user |
| email | VARCHAR(255) |  | The email address of the user |

transactions

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| user\_id | INT | FOREIGN KEY(user\_id => users.id) | The reference to the user |
| dt | VARCHAR(19) |  | The date and time of transaction |
| amount | DECIMAL(5,2) |  | The amount of transaction |

**Sample Data Tables**

users

|  |  |
| --- | --- |
| **id** | **email** |
| 1 | [bblaszczynski0@devhub.com](mailto:bblaszczynski0@devhub.com) |

|  |  |
| --- | --- |
| 2 | [dwookey1@chronoengine.com](mailto:dwookey1@chronoengine.com) |
| 3 | [flerway2@wikipedia.org](mailto:flerway2@wikipedia.org) |

transactions

|  |  |  |
| --- | --- | --- |
| **user\_id** | **dt** | **amount** |
| 1 | 2024-02-23 19:30:03 | 942.50 |
| 1 | 2024-03-07 09:01:15 | 855.22 |
| 1 | 2024-04-01 04:18:41 | 253.35 |
| 1 | 2024-04-07 02:40:58 | 886.88 |
| 2 | 2024-02-25 05:11:39 | 957.77 |
| 2 | 2024-03-06 03:00:40 | 413.39 |
| 2 | 2024-03-07 14:41:03 | 906.16 |
| 2 | 2024-03-10 00:58:13 | 116.59 |
| 2 | 2024-03-13 23:38:29 | 550.31 |
| 2 | 2024-03-22 03:07:46 | 196.23 |
| 2 | 2024-03-24 00:23:14 | 399.76 |
| 2 | 2024-03-25 12:28:18 | 398.07 |
| 2 | 2024-03-27 09:11:15 | 212.33 |
| 2 | 2024-04-09 06:33:26 | 97.85 |
| 3 | 2024-03-01 17:24:48 | 323.11 |
| 3 | 2024-03-05 10:16:06 | 673.23 |
| 3 | 2024-03-08 14:19:46 | 236.74 |
| 3 | 2024-03-23 15:37:47 | 234.87 |
| 3 | 2024-04-05 20:55:45 | 989.35 |
| 3 | 2024-04-07 05:26:35 | 369.20 |

# Sample Output

+-------------------------+------------------+----------+----------+ +

|email |total\_transactions|min\_amount|max\_amount|total\_amount|

+-------------------------+------------------+----------+----------+ +

[|bblaszczynski0@devhub.com|1](mailto:bblaszczynski0@devhub.com)

[|dwookey1@chronoengine.com|8](mailto:dwookey1@chronoengine.com)

[|flerway2@wikipedia.org](mailto:flerway2@wikipedia.org) |4

|855.22

|116.59

|234.87

|855.22

|906.16

|673.23

|855.22

|3192.84

|1467.95

|

|

|

+-------------------------+------------------+----------+----------+ +

Question - 21

**SQL: Total Transactions and Sum for Each User**

In the digital age, payment systems are crucial for online transactions. For businesses operating these systems, understanding user activity helps optimize services and enhance engagement. A development team is creating a report that details the total number and sum of transactions for each user. User transaction patterns will help identify highly active users and those who need additional engagement.

The result should have the following columns: *email | total\_transactions | total\_amount*. *email* - the email address of the user

*total\_transactions* - the total number of transactions made by the user

*total\_amount* - the total sum of the transactions made by the user, with two decimal places, including trailing zeros if necessary, e.g., 500.00 The results should be sorted in ascending order by *email*.

# Note:

Only transactions that occurred in the year 2023 should be included.

**Schema**

users

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| id | INT | PRIMARY KEY | The identifier of the user |
| email | VARCHAR(255) |  | The email address of the user |

transactions

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| user\_id | INT | FOREIGN KEY(user\_id => users.id) | The reference to the user |
| dt | VARCHAR(19) |  | The date and time of transaction |
| amount | DECIMAL(5,2) |  | The amount of transaction |

**Sample Data Tables**

users

|  |  |
| --- | --- |
| **id** | **email** |
| 1 | [lvasilevich0@google.co.uk](mailto:lvasilevich0@google.co.uk) |
| 2 | [hscholey1@sina.com.cn](mailto:hscholey1@sina.com.cn) |
| 3 | [mmcjury2@hibu.com](mailto:mmcjury2@hibu.com) |

transactions

|  |  |  |
| --- | --- | --- |
| **user\_id** | **dt** | **amount** |
| 3 | 2022-12-05 00:16:56 | 162.11 |
| 1 | 2023-05-20 03:20:58 | 81.58 |
| 1 | 2023-06-08 19:24:02 | 52.46 |
| 1 | 2023-06-27 21:16:07 | 447.59 |
| 1 | 2023-07-20 08:19:32 | 136.68 |
| 1 | 2023-12-11 17:08:05 | 852.55 |

|  |  |  |
| --- | --- | --- |
| 1 | 2023-12-15 04:45:54 | 77.11 |
| 1 | 2023-12-22 00:46:34 | 670.71 |
| 1 | 2023-12-29 12:43:23 | 948.46 |
| 2 | 2023-01-04 00:51:46 | 793.50 |
| 2 | 2023-04-07 16:29:14 | 762.52 |
| 2 | 2023-06-17 17:42:50 | 527.18 |
| 2 | 2023-10-10 11:16:51 | 733.47 |
| 2 | 2023-10-18 23:32:00 | 920.14 |
| 3 | 2023-03-27 18:31:41 | 408.13 |
| 3 | 2023-04-08 09:57:55 | 817.88 |
| 3 | 2023-05-18 09:47:14 | 916.98 |
| 3 | 2023-09-14 14:00:54 | 53.30 |
| 3 | 2023-09-30 01:34:01 | 589.37 |
| 3 | 2024-01-27 15:13:58 | 666.37 |

# Sample Output

+-------------------------+------------------+ +

|email |total\_transactions|total\_amount|

+-------------------------+------------------+ +

[|hscholey1@sina.com.cn](mailto:hscholey1@sina.com.cn) |5

[|lvasilevich0@google.co.uk|8](mailto:lvasilevich0@google.co.uk)

[|mmcjury2@hibu.com](mailto:mmcjury2@hibu.com) |5

|3736.81

|3267.14

|2785.66

|

|

|

+-------------------------+------------------+ +

Question - 22

**SQL: Top Cryptocurrencies by Average Transaction Amount**

In the rapidly changing cryptocurrency market, understanding average transaction amounts provides valuable insights into market behavior and investor sentiment. A development team is creating a report for a financial dashboard that highlights the top 3 cryptocurrencies based on the average transaction amount for all transactions in 2023. Cryptocurrencies with higher average transaction values indicate higher investor confidence or larger institutional involvement.

The results should include the following columns: *name, avg\_transaction\_amount*. *name* - the name of the cryptocurrency

*avg\_transaction\_amount* - the average amount of transactions for the cryptocurrency during the specified period, rounded to two decimal places, including trailing zeros if necessary, e.g., 5.00

The result should be sorted in ascending order by *avg\_transaction\_amount*. The results should be limited to the top 3 coins.

# Note:

Only transactions that occurred in 2023 should be included.

**Schema**

coins

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| id | INT | PRIMARY KEY | The identifier of cryptocurrency |
| name | VARCHAR(255) |  | The name of the cryptocurrency |

transactions

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| coin\_id | INT | FOREIGN KEY(coin\_id => coins.id) | The reference to the cryptocurrency |
| dt | VARCHAR(19) |  | The date and time of transaction |
| amount | DECIMAL(5,2) |  | The amount of transaction |

**Sample Data Tables**

coins

|  |  |
| --- | --- |
| **id** | **name** |
| 1 | BitCash |
| 2 | Etherium |
| 3 | Litecoin |
| 4 | Ripple |
| 5 | Dogecoin |

transactions

|  |  |  |
| --- | --- | --- |
| **coin\_id** | **dt** | **amount** |
| 1 | 2022-12-09 19:40:17 | 60.91 |
| 1 | 2023-01-02 09:35:37 | 76.35 |
| 1 | 2023-03-21 09:34:39 | 23.11 |
| 1 | 2023-08-11 03:43:27 | 80.20 |
| 1 | 2023-10-21 19:42:46 | 29.59 |
| 2 | 2023-07-08 19:47:20 | 69.49 |
| 2 | 2023-09-22 14:23:40 | 23.13 |
| 3 | 2023-01-08 10:22:10 | 72.45 |
| 3 | 2023-01-28 00:54:51 | 98.72 |
| 3 | 2023-02-24 00:13:32 | 70.36 |
| 3 | 2023-05-16 15:13:19 | 93.59 |
| 4 | 2023-05-24 13:43:44 | 9.34 |
| 4 | 2023-07-25 14:59:09 | 78.52 |

|  |  |  |
| --- | --- | --- |
| 5 | 2023-01-20 15:49:38 | 81.66 |
| 5 | 2023-08-21 17:19:45 | 94.89 |
| 5 | 2023-10-25 00:44:42 | 64.40 |
| 5 | 2023-11-30 02:38:47 | 86.84 |
| 5 | 2023-12-31 03:26:39 | 58.99 |
| 2 | 2024-01-21 10:25:26 | 29.36 |
| 5 | 2024-01-08 03:09:00 | 95.25 |

# Sample Output

+--------+ +

|name |avg\_transaction\_amount|

+--------+ +

|Ripple |43.93 |

|Etherium|46.31 |

|BitCash |52.31 |

+--------+ +

Question - 23

**SQL: Cryptocurrency Transactions Summary Report**

In the fast-changing world of cryptocurrency, tracking transaction metrics is crucial for investors and financial analysts to understand market dynamics. A development team is creating a report that summarizes transactions in March 2024 for each cryptocurrency, providing users with critical insights into transaction activity to help them make informed investment decisions.

The results should include the following columns: *name, total\_transactions, min\_amount, max\_amount, avg\_amount*. *name* - the name of the cryptocurrency

*total\_transactions* - the total number of transactions for the cryptocurrency during the specified period

*min\_amount* - the minimum transaction amount for the cryptocurrency during the specified period, with two decimal places, including trailing zeros if necessary, e.g., 500.00

*max\_amount* - the maximum transaction amount for the cryptocurrency during the specified period, with two decimal places, including trailing zeros if necessary, e.g., 500.00

*avg\_amount* - the average transaction amount for the cryptocurrency during the specified period, rounded to two decimal places, including trailing zeros if necessary, e.g., 5.00

The results should be sorted in descending order by *total\_transactions* to easily identify the most actively traded coins, then in ascending order by *name*.

# Note:

Only transactions that occurred in March 2024 should be included.

**Schema**

coins

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| id | INT | PRIMARY KEY | The identifier of cryptocurrency coin |
| name | VARCHAR(255) |  | The name of the cryptocurrency coin |

transactions

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |

|  |  |  |  |
| --- | --- | --- | --- |
| coin\_id | INT | FOREIGN KEY(coin\_id => coins.id) | The reference to the cryptocurrency coin |
| dt | VARCHAR(19) |  | The date and time of transaction |
| amount | DECIMAL(5,2) |  | The amount of transaction |

**Sample Data Tables**

coins

|  |  |
| --- | --- |
| **id** | **name** |
| 1 | BitCash |
| 2 | Etherium |
| 3 | Litecoin |

transactions

|  |  |  |
| --- | --- | --- |
| **coin\_id** | **dt** | **amount** |
| 1 | 2024-02-24 14:56:31 | 69.38 |
| 2 | 2024-02-24 17:23:54 | 46.79 |
| 1 | 2024-03-07 15:07:57 | 73.45 |
| 1 | 2024-03-13 00:47:18 | 2.10 |
| 1 | 2024-03-13 06:21:06 | 5.68 |
| 1 | 2024-03-14 15:06:59 | 25.32 |
| 1 | 2024-03-20 02:10:37 | 72.85 |
| 2 | 2024-03-09 12:06:47 | 67.79 |
| 2 | 2024-03-16 06:17:14 | 82.87 |
| 2 | 2024-03-24 11:11:23 | 5.96 |
| 2 | 2024-03-26 14:36:34 | 21.80 |
| 3 | 2024-03-20 08:28:56 | 5.07 |
| 3 | 2024-03-23 06:45:13 | 11.85 |
| 3 | 2024-03-27 02:40:23 | 34.25 |
| 3 | 2024-03-28 07:37:50 | 81.12 |
| 3 | 2024-03-29 07:34:32 | 19.06 |
| 1 | 2024-04-02 08:49:17 | 13.04 |
| 2 | 2024-04-04 17:29:13 | 4.74 |
| 2 | 2024-04-04 20:17:36 | 94.42 |
| 2 | 2024-04-07 01:47:46 | 64.76 |

# Sample Output

+--------+------------------+----------+----------+ +

|name |total\_transactions|min\_amount|max\_amount|avg\_amount|

+--------+------------------+----------+----------+ +

|BitCash |5 |2.10 |73.45 |35.88 |

|Litecoin|5 |5.07 |81.12 |30.27 |

|Etherium|4 |5.96 |82.87 |44.61 |

+--------+------------------+----------+----------+ +

Question - 24

**SQL: Antivirus Suspicious File Extensions Report**

In cybersecurity, knowing the types of file extensions of suspicious files scanned by antivirus software is essential to understanding digital threats. A development team is creating a report for an antivirus dashboard that shows the top 5 suspicious file extensions for March 2024. It aims to give clients insights into common threats so they can protect their systems effectively.

The results should include the following columns: *extension, total\_suspicious\_files*. *extension* - the extension of the suspicious file

*total\_suspicious\_files* - the number of times files with this extension were flagged as suspicious

The results should be sorted in descending order by *total\_suspicious\_files* and then in ascending order by *extension*. The results should be limited to the top 5 extensions.

# Note:

Only files that were scanned and flagged as suspicious in March 2024 should be included.

**Schema**

suspicious\_files

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| filename | VARCHAR(255) |  | The name of the file |
| extension | VARCHAR(255) |  | The extension of the file |
| scan\_dt | VARCHAR(19) |  | The date and time when the file was scanned |
| is\_suspicious | BOOLEAN |  | The flag indicating whether the file was flagged as suspicious |

**Sample Data Tables**

suspicious\_files

|  |  |  |  |
| --- | --- | --- | --- |
| **filename** | **extension** | **scan\_dt** | **is\_suspicious** |
| Mauris.pdf | .pdf | 2024-04-05 23:55:27 | 1 |
| Augue.xls | .xls | 2024-02-28 18:11:28 | 1 |
| Sapien.avi | .avi | 2024-03-30 12:24:10 | 1 |
| Pulvinar.doc | .doc | 2024-03-08 22:00:41 | 1 |
| TemporConvallisNulla.gif | .gif | 2024-03-29 21:32:41 | 1 |
| InFaucibus.mp3 | .mp3 | 2024-03-20 14:18:32 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| EleifendPedeLibero.ppt | .ppt | 2024-03-05 04:47:56 | 1 |
| VestibulumAnteIpsum.ppt | .ppt | 2024-03-05 17:34:34 | 1 |
| IntegerPede.ppt | .ppt | 2024-03-12 17:11:28 | 1 |
| VenenatisNon.tiff | .tiff | 2024-03-20 18:04:47 | 1 |
| IaculisDiam.xls | .xls | 2024-03-01 05:18:03 | 1 |
| QuisqueArcuLibero.xls | .xls | 2024-03-09 09:00:32 | 1 |
| MaurisSit.png | .png | 2024-04-03 23:20:03 | 0 |
| SitAmetSem.mp3 | .mp3 | 2024-02-23 22:06:43 | 0 |
| Nisi.mp3 | .mp3 | 2024-02-29 09:40:45 | 0 |
| Magna.tiff | .tiff | 2024-02-27 00:25:16 | 0 |
| EratVestibulum.gif | .gif | 2024-03-30 04:19:52 | 0 |
| Neque.jpeg | .jpeg | 2024-03-07 07:11:26 | 0 |
| VolutpatQuam.ppt | .ppt | 2024-03-23 04:33:43 | 0 |
| NonQuam.xls | .xls | 2024-03-10 19:12:29 | 0 |

# Sample Output

+---------+ +

|extension|total\_suspicious\_files|

+---------+ +

+---------+ +

|  |  |  |
| --- | --- | --- |
| |.ppt | |3 | | |
| |.xls | |2 | | |
| |.avi | |1 | | |
| |.doc | |1 | | |
| |.gif | |1 | | |

Question - 25

**SQL: Antivirus Scanned Devices Report**

In cybersecurity, monitoring the number of devices scanned by antivirus software is essential for digital security. A development team is creating a feature for an antivirus dashboard to provide clients with a report detailing the total number of devices scanned. This report will focus on devices actively scanned in March 2024, giving clients insights into their antivirus activity.

The results should include the following columns: *email, total\_scanned\_devices*. *email* - the email address of the client

*total\_scanned\_devices* - the total number of devices that have been scanned for each client during the specified period

The results should be sorted in ascending order by *email*.

# Note:

Only devices that were scanned in March 2024 should be included.

**Schema**

clients

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| id | INT | PRIMARY KEY | The identifier of the client |
| email | VARCHAR(255) |  | The email address of the client |

devices

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| client\_id | INT | FOREIGN KEY(client\_id => clients.id) | The reference to the client |
| mac\_address | VARCHAR(255) |  | The MAC address of the device |
| is\_scanned | BOOLEAN |  | The flag indicating whether the device was scanned |
| scheduled\_scan\_dt | VARCHAR(19) |  | The date and time the device is scheduled to be scanned |

**Sample Data Tables**

clients

|  |  |
| --- | --- |
| **id** | **email** |
| 1 | [cbracegirdle0@irs.gov](mailto:cbracegirdle0@irs.gov) |
| 2 | [gwickardt1@msu.edu](mailto:gwickardt1@msu.edu) |
| 3 | [mpaulon2@edublogs.org](mailto:mpaulon2@edublogs.org) |

devices

|  |  |  |  |
| --- | --- | --- | --- |
| **client\_id** | **mac\_address** | **is\_scanned** | **scheduled\_scan\_dt** |
| 1 | 87-EF-C7-BD-DF-A2 | 1 | 2024-02-28 08:13:55 |
| 1 | 37-FE-45-2B-9D-2A | 1 | 2024-03-07 00:00:05 |
| 1 | 13-82-F2-48-88-FD | 1 | 2024-03-11 06:53:47 |
| 1 | 0D-56-2A-B2-33-EF | 1 | 2024-03-30 19:41:31 |
| 1 | 17-C3-E3-2E-37-7E | 1 | 2024-04-06 13:14:06 |
| 1 | 93-25-74-C5-07-32 | 0 | 2024-02-21 16:23:31 |
| 2 | 64-E1-5B-12-AC-F9 | 0 | 2024-02-22 10:40:23 |
| 2 | 0F-66-56-E2-B0-3A | 0 | 2024-02-23 11:03:58 |
| 2 | 40-F4-40-12-C8-A5 | 0 | 2024-02-24 18:01:50 |
| 2 | B0-2B-99-84-68-7C | 0 | 2024-02-25 21:31:06 |
| 2 | 07-1F-BD-16-AC-23 | 0 | 2024-02-29 21:45:22 |
| 2 | CA-79-F4-B4-9E-69 | 0 | 2024-03-29 00:05:10 |
| 3 | 71-EB-63-A2-3C-AF | 1 | 2024-03-03 07:50:20 |
| 3 | 0B-40-DF-14-53-0F | 1 | 2024-03-21 11:10:52 |
| 3 | 0A-77-ED-ED-50-28 | 1 | 2024-04-03 04:11:25 |

|  |  |  |  |
| --- | --- | --- | --- |
| 3 | A4-79-0C-6D-B8-4C | 1 | 2024-04-04 13:50:43 |
| 3 | 44-A5-56-27-C8-70 | 0 | 2024-03-28 21:15:26 |
| 3 | 93-64-42-51-62-6F | 0 | 2024-03-31 20:26:01 |
| 3 | 87-5E-B3-51-38-2D | 0 | 2024-04-05 10:51:00 |
| 3 | DE-F2-F6-AD-76-4A | 0 | 2024-04-08 23:15:03 |

# Sample Output

+---------------------+ +

|email |total\_scanned\_devices|

+---------------------+ +

[|cbracegirdle0@irs.gov|3](mailto:cbracegirdle0@irs.gov) |

[|mpaulon2@edublogs.org|2](mailto:mpaulon2@edublogs.org) |

+---------------------+ +

Question - 26

**SQL: Resource Usage Report for Online Hosting Panel**

In web hosting, monitoring resource utilization is essential for providers and customers. A development team is adding a new report that shows customers whose websites use more than 50% of any resource (CPU, memory, or disk). It will provide a detailed overview of their average resource consumption to enable more effective hosting management.

The results should include the following columns: *email, average\_cpu\_usage, average\_memory\_usage, average\_disk\_usage*. *email* - the email address of the customer

*average\_cpu\_usage* - the average CPU usage across all sites for each customer, rounded to two decimal places, including trailing zeros if necessary, e.g., 500.00

*average\_memory\_usage* - the average memory usage across all sites for each customer, rounded to two decimal places, including trailing zeros if necessary, e.g., 500.00

*average\_disk\_usage* - the average disk usage across all sites for each customer, rounded to two decimal places, including trailing zeros if necessary, e.g., 500.00

The results should be sorted in ascending order by *email*.

# Note:

Only customers with at least one average value of a metric (CPU, memory, or disk usage) greater than 50% should be included.

**Schema**

customers

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| id | INT | PRIMARY KEY | The identifier of the customer |
| email | VARCHAR(255) |  | The email address of the customer |

site\_metrics

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| customer\_id | INT | FOREIGN KEY(customer\_id => customers.id) | The reference to the customer |
| cpu\_usage | DECIMAL(5,2) |  | The CPU usage percentage |
| memory\_usage | DECIMAL(5,2) |  | The memory usage percentage |

|  |  |  |  |
| --- | --- | --- | --- |
| disk\_usage | DECIMAL(5,2) |  | The disk usage percentage |

**Sample Data Tables**

customers

|  |  |
| --- | --- |
| **id** | **email** |
| 1 | [lrathke0@usa.gov](mailto:lrathke0@usa.gov) |
| 2 | [epearsall1@fema.gov](mailto:epearsall1@fema.gov) |
| 3 | [sivasechko2@cisco.com](mailto:sivasechko2@cisco.com) |

site\_metrics

|  |  |  |  |
| --- | --- | --- | --- |
| **customer\_id** | **cpu\_usage** | **memory\_usage** | **disk\_usage** |
| 1 | 31.53 | 80.84 | 1.51 |
| 1 | 12.54 | 26.47 | 47.74 |
| 1 | 12.34 | 46.24 | 34.43 |
| 1 | 26.64 | 84.98 | 17.56 |
| 2 | 80.45 | 50.05 | 10.63 |
| 2 | 40.14 | 86.67 | 15.98 |
| 2 | 30.14 | 34.38 | 17.67 |
| 2 | 1.11 | 83.44 | 2.95 |
| 3 | 30.60 | 18.60 | 28.02 |
| 3 | 41.64 | 33.64 | 5.20 |
| 3 | 31.88 | 7.37 | 91.14 |
| 3 | 43.20 | 9.56 | 40.40 |
| 3 | 2.33 | 34.29 | 18.65 |
| 3 | 11.50 | 32.89 | 71.39 |
| 3 | 39.57 | 4.49 | 48.05 |
| 3 | 25.06 | 23.77 | 33.00 |
| 3 | 32.81 | 1.59 | 25.85 |
| 3 | 48.38 | 79.21 | 8.31 |
| 3 | 11.62 | 26.75 | 71.71 |
| 3 | 54.43 | 6.48 | 4.86 |

# Sample Output

+-------------------+-----------------+--------------------+ +

|email |average\_cpu\_usage|average\_memory\_usage|average\_disk\_usage|

+-------------------+-----------------+--------------------+ +

[|epearsall1@fema.gov|37.96](mailto:epearsall1@fema.gov) |63.64 |11.81 |

[|lrathke0@usa.gov](mailto:lrathke0@usa.gov) |20.76

|59.63

|25.31

|

+-------------------+-----------------+--------------------+ +

Question - 27

**SQL: Dashboard Report for Online Hosting Customers Panel**

In the dynamic world of web hosting, providing customers with a clear overview of their assets is key to enhancing user experience. A development team must create a feature for an online hosting customers panel that generates a dashboard report of all customers and the number of active websites they own.

The results should include the following columns: *email, total\_active\_sites*. *email* - the email address of the customer

*total\_active\_sites* - the total number of active websites for each customer

The results should be sorted in ascending order by *email*.

# Note:

Only active websites should be included.

**Schema**

customers

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| id | INT | PRIMARY KEY | The identifier of the customer |
| email | VARCHAR(255) |  | The email address of the customer |

sites

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| customer\_id | INT | FOREIGN KEY(customer\_id => customers.id) | The reference to the customer |
| url | VARCHAR(255) |  | The URL of the website |
| is\_active | BOOLEAN |  | The status of the website |

**Sample Data Tables**

customers

|  |  |
| --- | --- |
| **id** | **email** |
| 1 | [dcristofol0@slashdot.org](mailto:dcristofol0@slashdot.org) |
| 2 | [mbillanie1@japanpost.jp](mailto:mbillanie1@japanpost.jp) |
| 3 | [hmainz2@utexas.edu](mailto:hmainz2@utexas.edu) |

sites

|  |  |  |
| --- | --- | --- |
| **customer\_id** | **url** | **is\_active** |
| 1 | [https://trellian.com](https://trellian.com/) | 1 |
| 1 | [https://www.google.de](https://www.google.de/) | 1 |

|  |  |  |
| --- | --- | --- |
| 1 | [https://merriam-webster.com](https://merriam-webster.com/) | 1 |
| 1 | [https://wordpress.com](https://wordpress.com/) | 1 |
| 1 | [https://nsw.gov.au](https://nsw.gov.au/) | 1 |
| 1 | [https://www.barnesandnoble.com](https://www.barnesandnoble.com/) | 1 |
| 1 | [https://www.yahoo.com](https://www.yahoo.com/) | 1 |
| 2 | [https://cloudflare.com](https://cloudflare.com/) | 0 |
| 2 | [https://www.is.gd](https://www.is.gd/) | 1 |
| 2 | [https://www.unesco.org](https://www.unesco.org/) | 1 |
| 3 | [https://www.sina.com.cn](https://www.sina.com.cn/) | 0 |
| 3 | [https://xinhuanet.com](https://xinhuanet.com/) | 1 |
| 3 | [https://cyberchimps.com](https://cyberchimps.com/) | 1 |
| 3 | [https://ask.com](https://ask.com/) | 1 |
| 3 | [https://businessinsider.com](https://businessinsider.com/) | 1 |
| 3 | [https://www.dailymail.co.uk](https://www.dailymail.co.uk/) | 1 |
| 3 | [https://www.guardian.co.uk](https://www.guardian.co.uk/) | 1 |
| 3 | [https://www.microsoft.com](https://www.microsoft.com/) | 1 |
| 3 | [https://www.gizmodo.com](https://www.gizmodo.com/) | 1 |
| 3 | [https://www.163.com](https://www.163.com/) | 1 |

# Sample Output

+------------------------+ +

|email |total\_active\_sites|

+------------------------+ +

[|dcristofol0@slashdot.org|7](mailto:dcristofol0@slashdot.org)

[|hmainz2@utexas.edu](mailto:hmainz2@utexas.edu) |9

[|mbillanie1@japanpost.jp](mailto:mbillanie1@japanpost.jp) |2

|

|

|

+------------------------+ +

Question - 28

**SQL: Average Income Report in Online Tax Application**

A development team is enhancing an online tax application to provide users with actionable financial insights. A new feature under development aims to generate a report showcasing the accounts with the highest average income based on income received within a specific timeframe. The immediate focus is on income recorded in the first quarter of 2024.

The result should include the following columns: *iban, average\_income, total\_income*. *iban* - the IBAN of the account

*average\_income* - the average income for each account during the specified period, rounded to two decimal places, including trailing zeros if necessary, e.g., 500.00

*total\_income* - the total amount of income for each account during the specified period, with two decimal places, including trailing zeros if necessary, e.g., 500.00

The results should first be sorted in descending order by *average\_income*, then in ascending order by *iban*.

The results should identify the top 3 accounts by their average income during this period.

# Note:

Only income recorded in the first quarter of 2024 (from January 1, 2024, to March 31, 2024) should be included.

**Schema**

accounts

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| id | INT | PRIMARY KEY | The identifier of the account |
| iban | VARCHAR(255) |  | The IBAN of the account |

income

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| account\_id | INT | FOREIGN KEY(account\_id => accounts.id) | The reference to the account |
| dt | VARCHAR(19) |  | The date and time of income |
| amount | DECIMAL(6,2) |  | The income amount |

**Sample Data Tables**

accounts

|  |  |
| --- | --- |
| **id** | **iban** |
| 1 | SK39 8924 2092 2997 1101 4161 |
| 2 | PL28 9141 8610 8442 2367 7521 0000 |
| 3 | CH93 8418 0F7G KQK4 NEHF Q |
| 4 | GT41 TBM8 DPFH MTNS BVW5 D4CX VIRR |
| 5 | IT27 Y015 0159 036T W7E5 I6ZD EQZ |

income

|  |  |  |
| --- | --- | --- |
| **account\_id** | **dt** | **amount** |
| 1 | 2024-01-17 16:43:20 | 4061.53 |
| 1 | 2024-02-28 05:30:15 | 4488.11 |
| 1 | 2024-04-07 05:41:27 | 4001.91 |
| 2 | 2023-12-21 07:38:45 | 4313.69 |
| 2 | 2024-01-08 04:48:45 | 3640.82 |
| 2 | 2024-01-20 17:31:20 | 3385.15 |
| 3 | 2024-01-06 23:18:30 | 2347.15 |
| 3 | 2024-03-08 12:53:20 | 3814.86 |

|  |  |  |
| --- | --- | --- |
| 3 | 2024-04-01 21:18:16 | 2764.27 |
| 4 | 2024-01-02 23:52:06 | 3526.08 |
| 4 | 2024-02-04 12:32:28 | 2221.91 |
| 4 | 2024-02-11 19:44:53 | 4197.07 |
| 4 | 2024-03-06 06:28:34 | 1357.44 |
| 4 | 2024-03-16 16:13:49 | 1854.52 |
| 5 | 2023-12-31 22:08:57 | 2819.54 |
| 5 | 2024-01-14 18:03:47 | 2641.20 |
| 5 | 2024-01-23 07:50:22 | 3692.56 |
| 5 | 2024-02-28 23:43:28 | 1999.09 |
| 5 | 2024-03-20 10:29:44 | 1670.18 |
| 5 | 2024-03-27 11:12:04 | 1193.15 |

# Sample Output

+----------------------------------+--------------+ +

|iban |average\_income|total\_income |

+----------------------------------+--------------+ +

+----------------------------------+--------------+ +

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |SK39 | 8924 | 2092 | 2997 | 1101 | 4161 | |4274.82 | |8549.64 | | |
| |PL28 | 9141 | 8610 | 8442 | 2367 | 7521 | 0000|3512.99 | |7025.97 | | |
| |CH93 | 8418 | 0F7G | KQK4 | NEHF | Q | |3081.01 | |6162.01 | | |

Question - 29

**SQL: Tax Calculation for Online Tax Application**

In the world of finance, tax calculation is a fundamental operation. A development team is working on an online tax application designed to simplify tax calculations for users. One of the key features they aim to implement is a tax calculation mechanism. This mechanism will calculate a fixed 20% tax on the total income for each account in the system.

The results should have the following columns: *iban | total\_income | tax\_rate | calculated\_tax*. *iban* - the IBAN of the account

*total\_income* - the total income for the account, with two decimal places, including trailing zeros if necessary, e.g., 500.00 *tax\_rate* - the tax rate applied, which is a fixed text 20%

*calculated\_tax* - the tax calculated, rounded to two decimal places, including trailing zeros if necessary, e.g., 500.00 The results should be sorted in ascending order by *iban*.

# Note:

Only income in 2023 should be included.

**Schema**

accounts

|  |  |  |  |
| --- | --- | --- | --- |
| id | INT | PRIMARY KEY | The identifier of the account |
| iban | VARCHAR(255) |  | The IBAN of the account |

income

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| account\_id | INT | FOREIGN KEY(account\_id => accounts.id) | The reference to the account |
| dt | VARCHAR(19) |  | The date and time of income |
| amount | DECIMAL(6,2) |  | The income amount |

**Sample Data Tables**

accounts

|  |  |
| --- | --- |
| **id** | **iban** |
| 1 | FR55 4477 6154 73ND TN3F HMOU T36 |
| 2 | DK46 1272 1831 2573 01 |
| 3 | RS53 5237 5794 6016 5411 43 |

income

|  |  |  |
| --- | --- | --- |
| **account\_id** | **dt** | **amount** |
| 1 | 2022-12-31 10:03:42 | 2779.19 |
| 1 | 2023-02-04 08:50:14 | 1777.68 |
| 1 | 2023-02-13 04:22:07 | 1954.81 |
| 1 | 2023-03-04 14:46:04 | 1547.79 |
| 1 | 2023-05-23 15:42:13 | 1208.49 |
| 1 | 2023-05-24 23:24:07 | 1521.72 |
| 1 | 2023-07-28 11:01:46 | 1792.75 |
| 1 | 2023-12-07 14:19:09 | 2374.25 |
| 1 | 2024-01-27 05:55:36 | 2803.39 |
| 2 | 2022-12-03 18:04:34 | 1826.65 |
| 2 | 2023-02-17 00:59:57 | 3074.11 |
| 2 | 2023-03-01 08:17:15 | 1007.30 |
| 2 | 2023-08-19 09:16:41 | 4515.04 |
| 2 | 2024-01-08 04:14:22 | 3321.78 |
| 2 | 2024-01-10 15:16:28 | 2033.87 |
| 3 | 2023-05-09 07:28:27 | 3158.66 |
| 3 | 2023-05-22 04:39:34 | 3851.20 |
| 3 | 2023-07-21 19:51:14 | 4152.29 |

|  |  |  |
| --- | --- | --- |
| 3 | 2023-10-05 05:42:49 | 4722.20 |
| 3 | 2023-11-11 02:42:59 | 1592.16 |

# Sample Output

+---------------------------------+------------+--------+ +

|iban |total\_income|tax\_rate|calculated\_tax|

+---------------------------------+------------+--------+ +

+---------------------------------+------------+--------+ +

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |DK46 | 1272 | 1831 | 2573 | 01 | |8596.45 | |20% | |1719.29 | | |
| |FR55 | 4477 | 6154 | 73ND | TN3F | HMOU T36|12177.49 | |20% | |2435.50 | | |
| |RS53 | 5237 | 5794 | 6016 | 5411 | 43 |17476.51 | |20% | |3495.30 | | |

Question - 30

**SQL: Monthly Budget Report for Online Budgeting Application**

A development team is working on an online budgeting application and needs to implement a feature that generates a monthly report for each customer. The report should provide insights into the customers’ total expenses and total income, allowing them to better manage their finances.

The results should have the following columns: *email | total\_expenses | total\_income*. *email* - the email address of the customer

*total\_expenses* - the sum of all expenses recorded for the customer, showing two decimal places, for example, 500.00 *total\_income* - the sum of income recorded for the customer, showing two decimal places, for example, 500.00

The results should be sorted in ascending order by *email*.

# Note:

Only expenses and incomes that were recorded in March 2024 should be included in the results.

**Schema**

customers

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraint** | **Description** |
| id | INT | PRIMARY KEY | The identifier of the customer |
| email | VARCHAR(255) |  | The email address of the customer |

expenses

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraint** | **Description** |
| customer\_id | INT | FOREIGN KEY(customer\_id => customers.id) | The reference to the customer |
| dt | VARCHAR(19) |  | The date and time of expense |
| amount | DECIMAL(6,2) |  | The expense amount |

income

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraint** | **Description** |
| customer\_id | INT | FOREIGN KEY(customer\_id => customers.id) | The reference to the customer |
| dt | VARCHAR(19) |  | The date and time of income |
| amount | DECIMAL(6,2) |  | The income amount |

**Sample Data Tables**

customers

|  |  |
| --- | --- |
| **id** | **email** |
| 1 | [otoohey0@elpais.com](mailto:otoohey0@elpais.com) |
| 2 | [egrebbin1@state.gov](mailto:egrebbin1@state.gov) |
| 3 | [arides2@sohu.com](mailto:arides2@sohu.com) |

expenses

|  |  |  |
| --- | --- | --- |
| **customer\_id** | **dt** | **amount** |
| 1 | 2024-02-21 22:12:12 | 90.41 |
| 1 | 2024-02-27 06:48:37 | 792.88 |
| 1 | 2024-03-10 05:19:43 | 442.01 |
| 1 | 2024-03-11 19:48:25 | 327.35 |
| 1 | 2024-03-24 22:03:06 | 639.62 |
| 1 | 2024-03-29 00:37:46 | 150.12 |
| 1 | 2024-04-02 03:36:50 | 257.67 |
| 2 | 2024-02-21 06:11:26 | 400.22 |
| 2 | 2024-03-11 15:34:19 | 298.41 |
| 2 | 2024-03-25 04:36:27 | 376.87 |
| 2 | 2024-03-29 19:05:51 | 530.07 |
| 2 | 2024-03-30 07:07:28 | 287.84 |
| 2 | 2024-04-02 15:44:22 | 868.03 |
| 3 | 2024-03-01 16:02:47 | 33.30 |
| 3 | 2024-03-06 11:53:42 | 838.51 |
| 3 | 2024-03-20 23:34:48 | 968.08 |
| 3 | 2024-03-21 21:18:08 | 35.36 |
| 3 | 2024-03-30 06:51:13 | 956.12 |
| 3 | 2024-03-31 10:11:56 | 896.32 |
| 3 | 2024-03-31 22:36:57 | 740.94 |

income

|  |  |  |
| --- | --- | --- |
| **customer\_id** | **dt** | **amount** |
| 1 | 2024-02-20 21:00:55 | 366.66 |
| 1 | 2024-03-11 03:25:04 | 769.38 |

|  |  |  |
| --- | --- | --- |
| 1 | 2024-03-15 00:49:53 | 84.10 |
| 1 | 2024-03-21 18:32:51 | 839.48 |
| 1 | 2024-03-29 15:34:13 | 333.97 |
| 1 | 2024-04-01 00:34:24 | 253.13 |
| 1 | 2024-04-02 11:13:49 | 263.56 |
| 2 | 2024-02-20 15:03:26 | 822.75 |
| 2 | 2024-02-26 14:57:39 | 277.23 |
| 2 | 2024-03-19 09:24:47 | 24.08 |
| 2 | 2024-03-20 15:54:24 | 988.34 |
| 2 | 2024-04-02 08:28:38 | 990.54 |
| 3 | 2024-02-21 10:23:33 | 430.82 |
| 3 | 2024-02-29 08:25:32 | 482.85 |
| 3 | 2024-03-01 05:10:42 | 962.60 |
| 3 | 2024-03-04 08:27:34 | 30.21 |
| 3 | 2024-03-19 12:12:01 | 80.00 |
| 3 | 2024-03-21 00:32:10 | 674.76 |
| 3 | 2024-03-23 14:14:32 | 863.79 |
| 3 | 2024-04-09 13:37:07 | 51.42 |

# Sample Output

+-------------------+--------------+ +

|email |total\_expenses|total\_income|

+-------------------+--------------+ +

[|arides2@sohu.com](mailto:arides2@sohu.com) |4468.63

[|egrebbin1@state.gov|1493.19](mailto:egrebbin1@state.gov)

[|otoohey0@elpais.com|1559.10](mailto:otoohey0@elpais.com)

|2611.36

|1012.42

|2026.93

|

|

|

+-------------------+--------------+ +

Question - 31

**SQL: Balance Report for Online Budgeting Application**

In personal finance, knowing one’s income versus expenses is crucial. A development team is creating an online budgeting application to provide insights into financial health. One key feature is a balance report, showing the difference between each customer’s total income and total expenses.

The results should have the following columns: *email | balance*. *email* - the email address of the customer

*balance* - the difference between the total income and total expenses for the customer, showing two decimal places, for example, 500.00

The results should be sorted in ascending order by *email*.

# Note:

Only customers that have a negative balance should be included.

**Schema**

customers

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraint** | **Description** |
| id | INT | PRIMARY KEY | The identifier of the customer |
| email | VARCHAR(255) |  | The email address of the customer |

expenses

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraint** | **Description** |
| customer\_id | INT | FOREIGN KEY(customer\_id => customers.id) | The reference to the customer |
| amount | DECIMAL(6,2) |  | The expense amount |

income

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraint** | **Description** |
| customer\_id | INT | FOREIGN KEY(customer\_id => customers.id) | The reference to the customer |
| amount | DECIMAL(6,2) |  | The income amount |

**Sample Data Tables**

customers

|  |  |
| --- | --- |
| **id** | **email** |
| 1 | [dtollmache0@typepad.com](mailto:dtollmache0@typepad.com) |
| 2 | [eclutterbuck1@baidu.com](mailto:eclutterbuck1@baidu.com) |
| 3 | [mdensun2@ustream.tv](mailto:mdensun2@ustream.tv) |

expenses

|  |  |
| --- | --- |
| **customer\_id** | **amount** |
| 1 | 136.18 |
| 1 | 323.28 |
| 1 | 383.37 |
| 1 | 505.41 |
| 1 | 841.21 |
| 2 | 5.23 |
| 2 | 408.33 |
| 2 | 489.45 |
| 2 | 545.40 |
| 2 | 591.43 |
| 2 | 706.13 |

|  |  |
| --- | --- |
| 2 | 716.82 |
| 2 | 761.75 |
| 2 | 796.30 |
| 3 | 152.26 |
| 3 | 211.30 |
| 3 | 447.57 |
| 3 | 685.03 |
| 3 | 966.89 |
| 3 | 967.30 |

income

|  |  |
| --- | --- |
| **customer\_id** | **amount** |
| 1 | 39.44 |
| 1 | 49.49 |
| 1 | 292.19 |
| 1 | 419.36 |
| 1 | 529.26 |
| 1 | 695.43 |
| 1 | 763.72 |
| 1 | 797.92 |
| 1 | 833.34 |
| 2 | 139.42 |
| 2 | 422.18 |
| 2 | 506.59 |
| 2 | 566.00 |
| 2 | 697.92 |
| 2 | 938.51 |
| 3 | 304.66 |
| 3 | 345.03 |
| 3 | 371.86 |
| 3 | 371.88 |
| 3 | 552.08 |

# Sample Output

+-----------------------+ +

|email |balance |

+-----------------------+ +

[|eclutterbuck1@baidu.com|-1750.22|](mailto:eclutterbuck1@baidu.com)

[|mdensun2@ustream.tv](mailto:mdensun2@ustream.tv) |-1484.84|

+-----------------------+ +

Question - 32

**SQL: Monthly Sales Report**

The sales department of a company wants to generate a monthly sales report for the first quarter of the year to track the performance of each product. The goal is to identify the top-selling products based on the total sales amount.

The result should have the following columns: *name | month | total\_sales*. *name* - the name of the product.

*month* - the month name of the sale (e.g., "January", "February", "March").

*total\_sales* - the total sales amount for the product in a specific month, rounded to two decimal places, including trailing zeros if necessary (e.g., 5.00).

The result should be sorted first by *month* in ascending calendar order, then by *total\_sales* in descending order within each month.

# Note:

Consider only transactions that occurred in the first quarter of 2024.

**Schema**

products

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| id | INT | PRIMARY KEY | The identifier of the product |
| name | VARCHAR(255) |  | The name of the product |

sales

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| product\_id | INT | FOREIGN KEY(product\_id => products.id) | The reference to the product |
| dt | VARCHAR(19) |  | The date and time of sale |
| amount | DECIMAL(7,2) |  | The amount of the sale |

**Sample Data Tables**

products

|  |  |
| --- | --- |
| **id** | **name** |
| 1 | Luxury Gold Watch |
| 2 | Smartphone Holder Stand |
| 3 | Stainless Steel Water Bottle |

sales

|  |  |  |
| --- | --- | --- |
| **product\_id** | **dt** | **amount** |
| 1 | 2024-01-13 17:12:22 | 7008.16 |
| 1 | 2024-01-03 03:15:27 | 6191.64 |
| 1 | 2024-01-22 18:29:09 | 4527.86 |
| 1 | 2024-01-26 19:38:53 | 7828.36 |
| 1 | 2024-02-17 09:27:13 | 5273.16 |
| 1 | 2024-02-11 09:51:24 | 3364.73 |
| 1 | 2024-02-22 23:53:15 | 8584.33 |
| 2 | 2024-01-28 11:33:58 | 3710.06 |
| 2 | 2024-01-25 14:47:25 | 5221.02 |
| 2 | 2024-01-21 07:58:53 | 2525.72 |
| 2 | 2024-03-15 14:16:18 | 8158.08 |
| 2 | 2024-03-12 17:02:01 | 6760.77 |
| 3 | 2024-01-13 19:27:51 | 1942.79 |
| 3 | 2024-02-15 08:04:40 | 9186.38 |
| 3 | 2024-03-06 08:02:37 | 5821.97 |
| 3 | 2024-03-03 15:39:18 | 8676.24 |
| 2 | 2024-04-08 09:53:01 | 6327.20 |
| 1 | 2023-12-26 05:48:22 | 8360.43 |
| 1 | 2023-12-15 14:52:51 | 9101.30 |
| 3 | 2023-12-21 15:57:50 | 3857.98 |

# Sample Output

+----------------------------+--------+ +

|name |month |total\_sales|

+----------------------------+--------+ +

|Luxury Gold Watch |January |25556.02

|Smartphone Holder Stand |January |11456.80

|Stainless Steel Water Bottle|January |1942.79

|Luxury Gold Watch |February|17222.22

|Stainless Steel Water Bottle|February|9186.38

|Smartphone Holder Stand |March

|Stainless Steel Water Bottle|March

|

|

|

|

|

|14918.85 |

|14498.21 |

+----------------------------+--------+ +

Question - 33

**SQL: IT Project Resource Analysis**

An IT company is looking to analyze the allocation of resources across various projects. The goal is to understand the distribution of employees among projects, focusing on projects that require more resources due to their complexity or scale. The analysis should provide insights into the number of employees assigned to each project, the average experience level of the team, and identify projects that are potentially understaffed or overstaffed.

The result should have the following columns: *project\_name | employee\_count | avg\_experience\_years | is\_understaffed*. *project\_name* - the name of the project.

*employee\_count* - the total number of employees assigned to the project.

*avg\_experience\_years* - the average years of experience per employee on the project, rounded up to the nearest integer (the ceiling) *is\_understaffed* - the derived column that shows Yes if the employee count is less than 5, otherwise No .

The result should be sorted in descending order by *employee\_count*, and then in ascending order by *project\_name*.

# Note:

The same employee can be assigned to two or more projects.

The same employee can have two or more assignments to the same project, so the number of employees on a project should be counted as the number of assignments on that project.

Only include projects with more than 2 years of average experience per employee.

**Schema**

projects

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| id | INT | PRIMARY KEY | The identifier of the project |
| name | VARCHAR(255) |  | The name of the project |

employees

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| id | INT | PRIMARY KEY | The identifier of the employee |
| ein | VARCHAR(255) |  | The employee identification number |
| experience\_years | INT |  | Years of experience |

projects\_employees

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| project\_id | INT | FOREIGN KEY(project\_id => projects.id) | The reference to the project |
| employee\_id | INT | FOREIGN KEY(employee\_id => employees.id) | The reference to the employee |

**Sample Data Tables**

projects

|  |  |
| --- | --- |
| **id** | **name** |
| 1 | Project X |
| 2 | Sunshine Project |
| 3 | Blue Sky Initiative |

employees

|  |  |  |
| --- | --- | --- |
| **id** | **ein** | **experience\_years** |
| 1 | 62-0524667 | 4 |

|  |  |  |
| --- | --- | --- |
| 2 | 62-1435366 | 1 |
| 3 | 29-3144922 | 1 |
| 4 | 80-9606443 | 1 |
| 5 | 63-6630813 | 1 |

projects\_employees

|  |  |
| --- | --- |
| **project\_id** | **employee\_id** |
| 1 | 1 |
| 1 | 1 |
| 1 | 2 |
| 1 | 3 |
| 1 | 5 |
| 2 | 1 |
| 2 | 1 |
| 2 | 2 |
| 2 | 5 |
| 3 | 1 |
| 3 | 1 |
| 3 | 2 |
| 3 | 3 |
| 3 | 3 |
| 3 | 4 |
| 3 | 4 |
| 3 | 5 |
| 3 | 5 |
| 3 | 5 |
| 3 | 5 |

# Sample Output

+----------------+--------------+--------------------+ +

|project\_name |employee\_count|avg\_experience\_years|is\_understaffed|

+----------------+--------------+--------------------+ +

|Project X |5

|Sunshine Project|4

|3

|3

|No

|Yes

|

|

+----------------+--------------+--------------------+ +

Question - 34

**SQL: Ethereum Market Dashboard Analysis**

A cryptocurrency trading platform is interested in analyzing the Ethereum transactions on its platform. Specifically, the platform wants to understand the buying and selling activities of Ethereum for each wallet address.

The result should have the following columns: *wallet | total\_transactions | total\_bought | total\_sold*. *wallet* - the unique identifier of a wallet on the platform.

*total\_transactions* - the total number of transactions for the specific wallet address.

*total\_bought* - the total amount of Ethereum bought by the wallet, rounded to two decimal places, including trailing zeros if necessary (e.g., 5.00). *total\_sold* - the total amount of Ethereum sold by the wallet, rounded to two decimal places, including trailing zeros if necessary (e.g., 5.00).

The result should be sorted in ascending order by *wallet*.

# Note:

Positive transaction values represent purchases, negative values represent sales. Only include transactions that occurred in February 2024.

**Schema**

transactions

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| dt | VARCHAR(19) |  | The date and time when the transaction occurred |
| wallet | VARCHAR(255) |  | The wallet address involved in the transaction |
| amount | DECIMAL(4,2) |  | The amount of Ethereum transacted |

**Sample Data Tables**

transactions

|  |  |  |
| --- | --- | --- |
| **dt** | **wallet** | **amount** |
| 2024-01-31 13:42:19 | 0x1a2b3c4d5e6f7a8b9c0d1e2f3a4b5c | -5.78 |
| 2024-01-24 06:07:14 | 0x1a2b3c4d5e6f7a8b9c0d1e2f3a4b5c | -2.79 |
| 2024-01-25 06:39:19 | 0x9B8aDc2eFf4cC3DdEe5f6a7B8dE9aC1F | -2.73 |
| 2024-01-29 04:37:45 | 0x9B8aDc2eFf4cC3DdEe5f6a7B8dE9aC1F | 4.68 |
| 2024-02-28 06:20:04 | 0x1a2b3c4d5e6f7a8b9c0d1e2f3a4b5c | -7.36 |
| 2024-02-12 07:45:28 | 0x1a2b3c4d5e6f7a8b9c0d1e2f3a4b5c | -3.71 |
| 2024-02-25 10:49:54 | 0x1a2b3c4d5e6f7a8b9c0d1e2f3a4b5c | -3.53 |
| 2024-02-03 19:43:00 | 0x1a2b3c4d5e6f7a8b9c0d1e2f3a4b5c | 4.01 |
| 2024-02-14 08:55:30 | 0x1a2b3c4d5e6f7a8b9c0d1e2f3a4b5c | 8.20 |
| 2024-02-16 04:31:26 | 0x3a4FbC5Df2E1bBfDdE5c4fA7bF8dE7aC1F | -8.96 |
| 2024-02-06 23:45:31 | 0x3a4FbC5Df2E1bBfDdE5c4fA7bF8dE7aC1F | -7.88 |
| 2024-02-11 01:00:35 | 0x3a4FbC5Df2E1bBfDdE5c4fA7bF8dE7aC1F | -7.66 |
| 2024-02-25 09:39:01 | 0x3a4FbC5Df2E1bBfDdE5c4fA7bF8dE7aC1F | -7.45 |
| 2024-02-14 04:04:15 | 0x3a4FbC5Df2E1bBfDdE5c4fA7bF8dE7aC1F | 4.17 |

|  |  |  |
| --- | --- | --- |
| 2024-02-15 11:47:23 | 0x3a4FbC5Df2E1bBfDdE5c4fA7bF8dE7aC1F | 7.56 |
| 2024-02-24 14:58:54 | 0x9B8aDc2eFf4cC3DdEe5f6a7B8dE9aC1F | -1.45 |
| 2024-02-18 21:17:24 | 0x9B8aDc2eFf4cC3DdEe5f6a7B8dE9aC1F | 1.05 |
| 2024-02-19 11:12:32 | 0x9B8aDc2eFf4cC3DdEe5f6a7B8dE9aC1F | 3.67 |
| 2024-03-09 16:52:14 | 0x1a2b3c4d5e6f7a8b9c0d1e2f3a4b5c | 1.07 |
| 2024-03-07 10:29:46 | 0x9B8aDc2eFf4cC3DdEe5f6a7B8dE9aC1F | 7.26 |

# Sample Output

+------------------------------------+------------------+------------+ +

|wallet |total\_transactions|total\_bought|total\_sold|

+------------------------------------+------------------+------------+ +

|0x1a2b3c4d5e6f7a8b9c0d1e2f3a4b5c |5

|0x3a4FbC5Df2E1bBfDdE5c4fA7bF8dE7aC1F|6

|0x9B8aDc2eFf4cC3DdEe5f6a7B8dE9aC1F |3

|12.21

|11.73

|4.72

|14.60

|31.95

|1.45

|

|

|

+------------------------------------+------------------+------------+ +

Question - 35

**SQL: Employee Leave Tracker**

A human resources department needs to track the number of leave days taken by employees in a specific year. The aim is to identify employees who have exceeded their annual leave allowance.

The result should have the following columns: *email | leave\_days\_taken | leave\_status*. *email* - the email address of the employee.

*leave\_days\_taken* - the total number of leave days taken by the employee.

*leave\_status* - a derived column that shows Within Limit if the total number of leave days taken by the employee is less than or equal to 20, otherwise Exceeded .

The result should be sorted in ascending order by *email*.

# Note:

Only consider leave records that created in 2023.

**Schema**

employees

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| id | INT | PRIMARY KEY | The identifier of the employee |
| email | VARCHAR(255) |  | The email address of the employee |

leave\_records

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| employee\_id | INT | FOREIGN KEY(employee\_id => employees.id) | The reference to the employee |
| leave\_dt | VARCHAR(19) |  | The date and time when the leave record was created |
| days\_taken | INT |  | The number of leave days taken |

**Sample Data Tables**

employees

|  |  |
| --- | --- |
| **id** | **email** |
| 1 | [jquartly0@macromedia.com](mailto:jquartly0@macromedia.com) |
| 2 | [cchastand1@stanford.edu](mailto:cchastand1@stanford.edu) |
| 3 | [lpuckrin2@creativecommons.org](mailto:lpuckrin2@creativecommons.org) |

leave\_records

|  |  |  |
| --- | --- | --- |
| **employee\_id** | **leave\_dt** | **days\_taken** |
| 1 | 2022-11-10 11:52:14 | 4 |
| 1 | 2022-09-07 23:22:46 | 1 |
| 2 | 2022-11-11 01:47:50 | 7 |
| 3 | 2022-11-06 23:12:27 | 7 |
| 3 | 2022-11-17 07:43:18 | 7 |
| 1 | 2023-05-19 04:40:25 | 2 |
| 1 | 2023-12-25 16:29:51 | 7 |
| 1 | 2023-03-12 18:54:29 | 1 |
| 1 | 2023-08-23 12:33:56 | 6 |
| 2 | 2023-04-20 04:19:10 | 5 |
| 2 | 2023-04-28 00:41:50 | 7 |
| 3 | 2023-06-11 18:49:25 | 2 |
| 3 | 2023-12-23 15:53:10 | 7 |
| 3 | 2023-03-13 13:46:16 | 2 |
| 3 | 2023-10-08 11:57:43 | 2 |
| 3 | 2023-04-12 07:49:02 | 4 |
| 3 | 2023-01-17 06:05:35 | 6 |
| 1 | 2024-02-05 16:01:59 | 1 |
| 1 | 2024-01-05 22:15:30 | 7 |
| 2 | 2024-02-21 00:50:11 | 4 |

# Sample Output

+-----------------------------+----------------+ +

|email |leave\_days\_taken|leave\_status|

+-----------------------------+----------------+ +

[|cchastand1@stanford.edu](mailto:cchastand1@stanford.edu) |12 |Within Limit|

[|jquartly0@macromedia.com](mailto:jquartly0@macromedia.com) |16 |Within Limit|

[|lpuckrin2@creativecommons.org|23](mailto:lpuckrin2@creativecommons.org) |Exceeded |

+-----------------------------+----------------+ +

An email marketing platform wants to analyze user engagement by generating a report that shows the total number of emails sent, the total number of emails opened, and the open rate for each campaign.

Question - 36

**SQL: Email Platform Engagement Stats**

The result should have the following columns: *name | total\_emails\_sent | total\_emails\_opened | open\_rate*. *name* - the name of the email campaign.

*total\_emails\_sent* - the total number of emails sent in the campaign.

*total\_emails\_opened* - the total number of emails that were opened in the campaign.

*open\_rate* - percentage of emails opened out of the total emails sent, calculated as (emails\_opened / emails\_sent) \* 100 , rounded to two decimal places, including trailing zeros if necessary (e.g., 5.00).

The result should be sorted in descending order by *open\_rate*, and then alphabetically by *name*.

# Note:

Only include email campaigns with an open rate greater than 50%.

**Schema**

campaigns

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| id | INT | PRIMARY KEY | The identifier of the email campaign |
| name | VARCHAR(255) |  | The name of the email campaign |

email\_stats

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| campaign\_id | INT | FOREIGN KEY(campaign\_id => campaigns.id) | The reference to the email campaign |
| emails\_sent | INT |  | The number of emails sent in the email campaign |
| emails\_opened | INT |  | The number of emails opened in the email campaign |

**Sample Data Tables**

campaigns

|  |  |
| --- | --- |
| **id** | **name** |
| 1 | SummerSale2021 |
| 2 | FallPromo |
| 3 | WinterWonderland |

email\_stats

|  |  |  |
| --- | --- | --- |
| **campaign\_id** | **emails\_sent** | **emails\_opened** |
| 1 | 1749 | 775 |
| 1 | 641 | 423 |

|  |  |  |
| --- | --- | --- |
| 1 | 976 | 598 |
| 1 | 756 | 121 |
| 1 | 975 | 107 |
| 1 | 752 | 367 |
| 1 | 1068 | 809 |
| 1 | 1046 | 589 |
| 1 | 1212 | 939 |
| 1 | 567 | 214 |
| 2 | 1084 | 283 |
| 2 | 992 | 478 |
| 2 | 1505 | 604 |
| 3 | 899 | 315 |
| 3 | 742 | 554 |
| 3 | 1744 | 917 |
| 3 | 1163 | 423 |
| 3 | 1501 | 948 |
| 3 | 736 | 451 |
| 3 | 537 | 434 |

# Sample Output

+----------------+-----------------+-------------------+ +

|name |total\_emails\_sent|total\_emails\_opened|open\_rate|

+----------------+-----------------+-------------------+ +

|WinterWonderland|7322 |4042 |55.20 |

|SummerSale2021 |9742 |4942 |50.73 |

+----------------+-----------------+-------------------+ +

Question - 37

**SQL: Bond Maturity Analysis**

A team of financial analysts wants a comprehensive assessment of the maturity dates associated with mortgages within various mortgage-backed bonds held by their company. Draft a query that delivers insights for each bond.

The insights required are a count of the mortgage maturity dates per bond, the earliest maturity date, the latest maturity date, and the average days to maturity.

The result should have the following columns: *name | maturity\_dates | earliest\_maturity | latest\_maturity | avg\_days\_to\_maturity*. *name* - name of the bond

*maturity\_dates* - total number of maturity dates for a specific bond *earliest\_maturity* - earliest maturity date for a specific bond

*latest\_maturity* - latest maturity date for a specific bond

*avg\_days\_to\_maturity* - average days to maturity for a specific bond, rounded up to the nearest whole number, e.g., 4.01 rounds up to 5.

The result should be sorted in ascending order by *name*.

# Note:

Only bonds with an average days to maturity greater than 365 days should be included in the result. Today is September 13, 2023.

Use actual days rather than 30-day months.

**Schema**

|  |  |  |  |
| --- | --- | --- | --- |
| **bonds** | | | |
| **name** | **type** | **constraint** | **description** |
| id | INT | PRIMARY KEY | Bond ID |
| name | VARCHAR(255) |  | Name of the bond |

|  |  |  |  |
| --- | --- | --- | --- |
| **maturities** | | | |
| **name** | **type** | **constraint** | **description** |
| bond\_id | INT | FOREIGN KEY(bond\_id => bonds.id) | Reference to the bond |
| maturity | DATE |  | Maturity date of the bond |

**Sample Data Tables**

|  |  |
| --- | --- |
| **bonds** | |
| **id** | **name** |
| 1 | Alpha Mortgage Bond |
| 2 | Beta Mortgage Bond |
| 3 | Gamma Mortgage Bond |

|  |  |
| --- | --- |
| **maturities** | |
| **bond\_id** | **maturity** |
| 1 | 2024-01-26 |
| 1 | 2024-02-22 |
| 1 | 2024-03-26 |
| 1 | 2024-05-13 |
| 1 | 2024-07-06 |

|  |  |
| --- | --- |
| 1 | 2024-08-23 |
| 1 | 2024-09-06 |
| 1 | 2024-11-30 |
| 1 | 2024-12-30 |
| 1 | 2025-04-30 |
| 1 | 2025-05-03 |
| 2 | 2024-07-25 |
| 2 | 2024-12-07 |
| 3 | 2023-12-16 |
| 3 | 2024-01-25 |
| 3 | 2024-01-26 |
| 3 | 2024-05-04 |
| 3 | 2024-10-02 |
| 3 | 2024-12-14 |
| 3 | 2025-01-15 |

**Expected Output**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **name** | **maturity\_dates** | **earliest\_maturity** | **latest\_maturity** | **avg\_days\_to\_maturity** |
| Beta Mortgage Bond | 2 | 2024-07-25 | 2024-12-07 | 384 |

Question - 38

**SQL: Bond Interest Rate Analysis**

A finance team is keen to analyze the coupon rates associated with their variable-rate mortgage-backed bonds. The insights required for each bond are a count of the interest rates seen, the lowest interest rate, the highest interest rate, and the average interest rate.

The result should have the following columns: *name | interest\_rates | lowest\_rate | highest\_rate | avg\_rate*. *name* - name of the bond

*interest\_rates* - total number of interest rates for a specific bond

*lowest\_rate* - lowest interest rate for a specific bond with one place after the decimal *highest\_rate* - highest interest rate for a specific bond with one place after the decimal *avg\_rate* - average interest rate for a specific bond, rounded to two decimal places

The results should be sorted in ascending order by *name*. **Note:**

Only bonds with an average interest rate greater than 3% should be included in the result.

Retain trailing zeros after the decimal as required, e.g., 5.0 or 3.10.

**Schema**

|  |  |  |  |
| --- | --- | --- | --- |
| **bonds** | | | |
| **name** | **type** | **constraint** | **description** |
| id | INT | PRIMARY KEY | Bond ID |
| name | VARCHAR(255) |  | Name of the bond |

|  |  |  |  |
| --- | --- | --- | --- |
| **interest\_rates** | | | |
| **name** | **type** | **constraint** | **description** |
| bond\_id | INT | FOREIGN KEY(bond\_id => bonds.id) | Reference to the bond |
| rate | DECIMAL(2,1) |  | Interest rate of the bond (in percent) |

**Sample Data Tables**

|  |  |
| --- | --- |
| **bonds** | |
| **id** | **name** |
| 1 | Alpha Mortgage Bond |
| 2 | Beta Mortgage Bond |
| 3 | Gamma Mortgage Bond |

|  |  |
| --- | --- |
| **interest\_rates** | |
| **bond\_id** | **rate** |
| 1 | 1.4 |
| 1 | 1.8 |
| 1 | 2.0 |
| 1 | 2.4 |
| 1 | 3.4 |
| 1 | 4.6 |
| 1 | 4.7 |
| 1 | 4.9 |

|  |  |
| --- | --- |
| 2 | 2.0 |
| 2 | 2.1 |
| 2 | 3.0 |
| 2 | 3.2 |
| 2 | 4.0 |
| 3 | 1.2 |
| 3 | 1.3 |
| 3 | 1.4 |
| 3 | 2.1 |
| 3 | 2.5 |
| 3 | 3.5 |
| 3 | 4.0 |

**Expected Output**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **name** | **interest\_rates** | **lowest\_rate** | **highest\_rate** | **avg\_rate** |
| Alpha Mortgage Bond | 8 | 1.4 | 4.9 | 3.15 |

Question - 39

**SQL: Bond Cash Flow Analysis for Bondholders**

A company's finance department wishes to understand the total cash flow expected from bond investments for each bondholder. Create a query to calculate this based on coupon payments of the bonds each bondholder possesses.

The result should have the following columns: *name | total\_cash\_flow*. *name* - name of the bondholder

*total\_cash\_flow* - total expected cash flow from all bonds the bondholder possesses with 2 places after the decimal, e.g., 10 shows as 10.00.

The result should be sorted in descending order by *total\_cash\_flow*.

# Note:

Only bondholders with a total expected cash flow greater than 10,000.00 should be included in the result. The total\_cash\_flow = annual\_coupon x coupons\_remaining.

**Schema**

**bondholders**

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **constraint** | **description** |
| id | INT | PRIMARY KEY | Bondholder ID |
| name | VARCHAR(255) |  | Name of the bondholder |

|  |  |  |  |
| --- | --- | --- | --- |
| **bonds** | | | |
| **name** | **type** | **constraint** | **description** |
| id | INT | PRIMARY KEY | Bond ID |
| name | VARCHAR(255) |  | Name of the bond |
| annual\_coupon | DECIMAL(5,2) |  | Annual coupon payment |
| coupons\_remaining | INT |  | Number of remaining coupon payments |

|  |  |  |  |
| --- | --- | --- | --- |
| **bondholders\_bonds** | | | |
| **name** | **type** | **constraint** | **description** |
| bondholder\_id | INT | FOREIGN KEY(bondholder\_id => bondholders.id) | Reference to the bondholder |
| bond\_id | INT | FOREIGN KEY(bond\_id => bonds.id) | Reference to the bond |

**Sample Data Tables**

|  |  |
| --- | --- |
| **bondholders** | |
| **id** | **name** |
| 1 | Alex Smith |
| 2 | Taylor Johnson |
| 3 | Jordan Davis |

|  |  |  |  |
| --- | --- | --- | --- |
| **bonds** | | | |
| **id** | **name** | **annual\_coupon** | **coupons\_remaining** |
| 1 | Golden Bonds | 150.00 | 4 |
| 2 | Silver Lining | 200.00 | 2 |
| 3 | Diamond Trust | 100.00 | 4 |
| 4 | Emerald Wealth | 350.00 | 5 |
| 5 | Ruby Returns | 150.00 | 8 |
| 6 | Sapphire Security | 450.00 | 5 |

|  |  |  |  |
| --- | --- | --- | --- |
| 7 | Amber Assurance | 100.00 | 8 |
| 8 | Topaz Treasury | 100.00 | 2 |
| 9 | Opal Opportunities | 150.00 | 5 |
| 10 | Pearl Prosperity | 450.00 | 5 |
| 11 | Platinum Promise | 450.00 | 9 |
| 12 | Jade Investments | 350.00 | 1 |
| 13 | Garnet Growth | 150.00 | 4 |
| 14 | Onyx Returns | 350.00 | 2 |
| 15 | Quartz Capital | 100.00 | 2 |
| 16 | Citrine Securities | 250.00 | 2 |
| 17 | Aquamarine Assets | 250.00 | 2 |
| 18 | Peridot Portfolio | 300.00 | 8 |
| 19 | Tourmaline Trust | 100.00 | 6 |
| 20 | Moonstone Money | 150.00 | 9 |

|  |  |
| --- | --- |
| **bondholders\_bonds** | |
| **bondholder\_id** | **bond\_id** |
| 1 | 1 |
| 1 | 2 |
| 1 | 6 |
| 1 | 8 |
| 1 | 9 |
| 1 | 13 |
| 1 | 14 |
| 1 | 16 |
| 1 | 17 |
| 2 | 4 |
| 2 | 5 |
| 2 | 7 |
| 2 | 11 |
| 2 | 15 |

|  |  |
| --- | --- |
| 2 | 18 |
| 3 | 3 |
| 3 | 10 |
| 3 | 12 |
| 3 | 19 |
| 3 | 20 |

**Expected Output**

|  |  |
| --- | --- |
| **name** | **total\_cash\_flow** |
| Taylor Johnson | 10400.00 |

Question - 40

**SQL: Sum of the Cash Flows Analysis**

A financial analysis firm is building a tool to analyze the sum of the cash flows from bond investments. Create a query to extract insights on the cash flows per investor.

The required statistics are a list of all investors and their total, minimum, maximum, and average cash flows from investments.

The result should have the following columns: *email | investments | min\_cash\_flow | max\_cash\_flow | avg\_cash\_flow*. *email* - investor email

*investments* - total number of investments by a specific investor

*min\_cash\_flow* - minimum cash flow from investments for a specific investor *max\_cash\_flow* - maximum cash flow from investments for a specific investor

*avg\_cash\_flow* - average cash flow from investments for a specific investor, rounded to two decimal places The result should be sorted in ascending order by *email*.

# Note:

Only investors who have a total cash flow greater than 1,000,000 should be included in the results.

**Schema**

|  |  |  |  |
| --- | --- | --- | --- |
| **investors** | | | |
| **name** | **type** | **constraint** | **description** |
| id | INT | PRIMARY KEY | Investor ID |
| email | VARCHAR(255) | UNIQUE | Email address |

|  |  |  |  |
| --- | --- | --- | --- |
| **cash\_flows** | | | |
| **name** | **type** | **constraint** | **description** |
| investor\_id | INT | FOREIGN KEY(investor\_id => investors.id) | Reference to the investor |
| cash\_flow | DECIMAL(8,2) |  | Cash flow amount |

**Sample Data Tables**

|  |  |
| --- | --- |
| **investors** | |
| **id** | **email** |
| 1 | [ematson0@ebay.co.uk](mailto:ematson0@ebay.co.uk) |
| 2 | [lsalvadore1@msn.com](mailto:lsalvadore1@msn.com) |
| 3 | [aclowser2@patch.com](mailto:aclowser2@patch.com) |

|  |  |
| --- | --- |
| **cash\_flows** | |
| **investor\_id** | **cash\_flow** |
| 1 | 184040.12 |
| 1 | 179280.08 |
| 1 | 179374.42 |
| 1 | 79302.21 |
| 1 | 87466.20 |
| 1 | 194588.36 |
| 1 | 153563.92 |
| 1 | 56377.92 |
| 2 | 59039.14 |
| 2 | 167247.23 |
| 2 | 59311.03 |
| 2 | 183883.00 |
| 2 | 118851.21 |
| 3 | 58868.62 |
| 3 | 96909.26 |
| 3 | 103735.73 |

|  |  |
| --- | --- |
| 3 | 171261.97 |
| 3 | 86463.11 |
| 3 | 56931.73 |
| 3 | 194699.58 |

**Expected Output**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **email** | **investments** | **min\_cash\_flow** | **max\_cash\_flow** | **avg\_cash\_flow** |
| [ematson0@ebay.co.uk](mailto:ematson0@ebay.co.uk) | 8 | 56377.92 | 194588.36 | 139249.15 |

Question - 41

**SQL: Expected Cash Flow Analysis**

A financial analytics platform is developing a feature to help investors understand the expected cash flows from their bond investments. Create a query that provides insights into each investor's expected cash flows.

The output should list all investors with their total number of investments, the sum of expected cash flows, and the range of expected cash flows (the difference between the highest and the lowest expected cash flow).

The result should have the following columns: *email | investment\_count | total\_expected\_flow | range\_expected\_flow*. *email* - investor email

*investment\_count* - total number of investments held by a specific investor

*total\_expected\_flow* - sum of expected cash flows for a specific investor

*range\_expected\_flow* - the range of expected cash flows for a specific investor ( maximum - minimum ) Sort the result in ascending order by *email*.

# Note:

Only investors who have a sum of expected cash flows greater than 100,000 should be included in the result.

**Schema**

|  |  |  |  |
| --- | --- | --- | --- |
| **investors** | | | |
| **name** | **type** | **constraint** | **description** |
| id | INT | PRIMARY KEY | Investor ID |
| email | VARCHAR(255) | UNIQUE | Email address |

|  |  |  |  |
| --- | --- | --- | --- |
| **cash\_flows** | | | |
| **name** | **type** | **constraint** | **description** |

|  |  |  |  |
| --- | --- | --- | --- |
| investor\_id | INT | FOREIGN KEY(investor\_id => investors.id) | Reference to the investor |
| expected\_flow | DECIMAL(8,2) |  | Expected cash flow amount |

**Sample Data Tables**

|  |  |
| --- | --- |
| **investors** | |
| **id** | **email** |
| 1 | [tdowner0@timesonline.co.uk](mailto:tdowner0@timesonline.co.uk) |
| 2 | [cgarza1@opera.com](mailto:cgarza1@opera.com) |
| 3 | [nbarwise2@si.edu](mailto:nbarwise2@si.edu) |

|  |  |
| --- | --- |
| **cash\_flows** | |
| **investor\_id** | **expected\_flow** |
| 1 | 24923.83 |
| 1 | 30212.10 |
| 1 | 87126.50 |
| 1 | 56018.65 |
| 1 | 93357.47 |
| 1 | 55073.54 |
| 1 | 27095.07 |
| 2 | 44165.12 |
| 2 | 43658.84 |
| 2 | 35835.34 |
| 2 | 12660.46 |
| 2 | 58676.60 |
| 2 | 95929.25 |
| 2 | 47161.23 |
| 2 | 80283.91 |
| 2 | 54427.20 |
| 2 | 93223.98 |
| 3 | 19741.35 |

|  |  |
| --- | --- |
| 3 | 12243.25 |
| 3 | 50470.06 |

**Expected Output**

|  |  |  |  |
| --- | --- | --- | --- |
| **email** | **investment\_count** | **total\_expected\_flow** | **range\_expected\_flow** |
| [cgarza1@opera.com](mailto:cgarza1@opera.com) | 10 | 566021.93 | 83268.79 |
| [tdowner0@timesonline.co.uk](mailto:tdowner0@timesonline.co.uk) | 7 | 373807.16 | 68433.64 |

Question - 42

**SQL: Online Store Coupon Codes Report**

An online store needs a report on the usage of its coupon codes.

The result should have the following columns: *coupon\_code | description | total\_uses | min\_discount | max\_discount | avg\_discount*. *coupon\_code* - the unique code of the coupon

*description* - a description of the coupon code

*total\_uses* - the total number of times the coupon code has been redeemed *min\_discount* - the smallest discount amount given using this coupon code *max\_discount* - the largest discount amount given using this coupon code

*avg\_discount* - the average discount amount given using this coupon code, rounded to two places after the decimal The result should be sorted in ascending order by *coupon\_code*.

# Note:

Only active coupons should be included in the report.

**Schema**

|  |  |  |  |
| --- | --- | --- | --- |
| **coupons** | | | |
| **name** | **type** | **constraint** | **description** |
| id | INT | PRIMARY KEY | Coupon Code ID |
| coupon\_code | VARCHAR(255) | UNIQUE | Unique code of the coupon |
| description | VARCHAR(255) |  | Description of the coupon code |
| is\_enabled | SMALLINT |  | Coupon enable flag |

|  |  |  |  |
| --- | --- | --- | --- |
| **coupon\_uses** | | | |
| **name** | **type** | **constraint** | **description** |
| coupon\_id | INT | FOREIGN KEY(coupon\_id => coupons.id) | Reference to the coupon |

|  |  |  |  |
| --- | --- | --- | --- |
| amount | DECIMAL(4,2) |  | Discount amount given using the coupon |

**Sample Data Tables**

|  |  |  |  |
| --- | --- | --- | --- |
| **coupons** | | | |
| **id** | **coupon\_code** | **description** | **is\_enabled** |
| 1 | COUPON123 | nisi nam ultrices libero non | 0 |
| 2 | SAVE20 | ac est lacinia | 1 |
| 3 | DISCOUNT50 | quis odio consequat | 1 |

|  |  |
| --- | --- |
| **coupon\_uses** | |
| **coupon\_id** | **amount** |
| 1 | 36.68 |
| 1 | 3.56 |
| 1 | 2.10 |
| 1 | 39.58 |
| 2 | 39.81 |
| 2 | 24.07 |
| 2 | 28.42 |
| 2 | 31.03 |
| 2 | 3.24 |
| 2 | 36.33 |
| 3 | 8.89 |
| 3 | 30.44 |
| 3 | 36.94 |
| 3 | 42.65 |
| 3 | 33.61 |
| 3 | 41.92 |
| 3 | 1.78 |
| 3 | 20.26 |
| 3 | 27.92 |

|  |  |
| --- | --- |
| 3 | 0.23 |

**Expected Output**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **coupon\_code** | **description** | **total\_uses** | **min\_discount** | **max\_discount** | **avg\_discount** |
| DISCOUNT50 | quis odio consequat | 10 | 0.23 | 42.65 | 24.46 |
| SAVE20 | ac est lacinia | 6 | 3.24 | 39.81 | 27.15 |

Question - 43

**SQL: Freelancer Platform Yearly Income Report**

A freelancer platform wants to generate a yearly report detailing the income they have earned from their top professions.

The result should have the following columns: *title | total\_projects | total\_income | total\_freelancers | average\_income\_per\_freelancer*. *title* - the profession title (e.g., Web Development, Graphic Design)

*total\_projects* - the total number of projects in each profession

*total\_income* - the total income the platform earned from each profession *total\_freelancers* - the total number of freelancers in each profession

*average\_income\_per\_freelancer* - the average income earned from each project in the profession, rounded to two decimal places The result should be sorted in descending order by *total\_income*.

# Note:

Only completed projects should be included in the report.

**Schema**

|  |  |  |  |
| --- | --- | --- | --- |
| **professions** | | | |
| **name** | **type** | **constraint** | **description** |
| id | INT | PRIMARY KEY | Profession ID |
| title | VARCHAR(255) | UNIQUE | Title of profession (e.g., Web Development) |

|  |  |  |  |
| --- | --- | --- | --- |
| **freelancers** | | | |
| **name** | **type** | **constraint** | **description** |
| id | INT | PRIMARY KEY | Freelancer ID |
| profession\_id | INT | FOREIGN KEY(profession\_id => professions.id) | Reference to the profession |
| email | VARCHAR(255) | UNIQUE | Email of the freelancer |

|  |  |  |  |
| --- | --- | --- | --- |
| **projects** | | | |
| **name** | **type** | **constraint** | **description** |
| id | INT | PRIMARY KEY | Project ID |
| freelancer\_id | INT | FOREIGN KEY(freelancer\_id => freelancers.id) | Reference to the freelancer |
| status | ENUM('Completed','Ongoing','Cancelled') |  | Status of the project |
| income | DECIMAL(6,2) |  | Income earned by the platform for the project |

**Sample Data Tables**

|  |  |
| --- | --- |
| **professions** | |
| **id** | **title** |
| 1 | Artificial Intelligence Engineer |
| 3 | Game Developer |
| 2 | Network Administrator |

|  |  |  |
| --- | --- | --- |
| **freelancers** | | |
| **id** | **profession\_id** | **email** |
| 1 | 1 | [lfernez0@microsoft.com](mailto:lfernez0@microsoft.com) |
| 3 | 2 | [mbrydone2@delicious.com](mailto:mbrydone2@delicious.com) |
| 4 | 2 | [jhamp3@4shared.com](mailto:jhamp3@4shared.com) |
| 5 | 3 | [cparfett4@twitter.com](mailto:cparfett4@twitter.com) |

|  |  |  |  |
| --- | --- | --- | --- |
| **projects** | | | |
| **id** | **freelancer\_id** | **status** | **income** |
| 5 | 1 | Completed | 8562.13 |
| 11 | 1 | Completed | 6727.56 |
| 10 | 3 | Completed | 3753.46 |
| 20 | 3 | Completed | 6659.39 |
| 6 | 4 | Completed | 8459.28 |
| 13 | 4 | Completed | 5899.31 |
| 16 | 4 | Completed | 2709.63 |

|  |  |  |  |
| --- | --- | --- | --- |
| 4 | 5 | Completed | 5029.44 |
| 7 | 5 | Completed | 1763.94 |
| 9 | 5 | Completed | 6988.36 |
| 8 | 3 | Cancelled | 8699.67 |
| 1 | 5 | Cancelled | 5403.21 |
| 19 | 3 | Ongoing | 72.51 |
| 3 | 4 | Ongoing | 8561.14 |
| 15 | 4 | Ongoing | 9235.78 |
| 17 | 4 | Ongoing | 4307.76 |

**Expected Output**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **title** | **total\_projects** | **total\_income** | **total\_freelancers** | **average\_income\_per\_freelancer** |
| Network Administrator | 5 | 27481.07 | 2 | 5496.21 |
| Artificial Intelligence Engineer | 2 | 15289.69 | 1 | 7644.85 |
| Game Developer | 3 | 13781.74 | 1 | 4593.91 |

Question - 44

**SQL: Ecommerce Warehouse Stock Report**

An ecommerce website admin panel needs to generate a report to display product stock status in their warehouse. The report should display a list of all product categories and the total number of products available in each category. It should be limited to products with more than 10 units available.

The result should have the following columns: *category | title | total\_stock*. *category* - the name of the product category (e.g., Electronics, Clothing) *title* - the title of the product

*total\_stock* - total number of items available in stock for that product

The result should be sorted in ascending order by *category*, then in ascending order by *title,* and finally in descending order by *total\_stock*.

# Note:

Only products with a total stock of more than 10 items should be included in the report.

**Schema**

|  |  |  |  |
| --- | --- | --- | --- |
| **categories** | | | |
| **name** | **type** | **constraint** | **description** |
| id | INT | PRIMARY KEY | Category ID |

|  |  |  |  |
| --- | --- | --- | --- |
| title | VARCHAR(255) | UNIQUE | Title of the product category |

|  |  |  |  |
| --- | --- | --- | --- |
| **products** | | | |
| **name** | **type** | **constraint** | **description** |
| id | INT | PRIMARY KEY | Product ID |
| category\_id | INT | FOREIGN KEY(category\_id => categories.id) | Category ID reference |
| title | VARCHAR(255) |  | Title of the product |
| sku | VARCHAR(255) | UNIQUE | SKU of the product |
| stock\_number | INT |  | Number of items available in stock |

**Sample Data Tables**

|  |  |
| --- | --- |
| **categories** | |
| **id** | **title** |
| 1 | Electronics |
| 2 | Clothing |
| 3 | Home & Kitchen |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **products** | | | | |
| **id** | **category\_id** | **title** | **sku** | **stock\_number** |
| 11 | 1 | Elegant Gadget | EG-11 | 4 |
| 3 | 1 | Luxury Gizmo | LG-3 | 10 |
| 19 | 1 | Sleek Widget | SW-19 | 8 |
| 8 | 1 | Sleek Widget | SW-8 | 8 |
| 14 | 2 | Elegant Gadget | EG-14 | 2 |
| 16 | 2 | Elegant Gadget | EG-16 | 6 |
| 10 | 2 | Elegant Gadget | EG-10 | 10 |
| 7 | 2 | Luxury Gizmo | LG-7 | 3 |
| 2 | 2 | Luxury Gizmo | LG-2 | 8 |
| 18 | 2 | Luxury Gizmo | LG-18 | 9 |
| 1 | 2 | Sleek Widget | SW-1 | 3 |
| 6 | 2 | Sleek Widget | SW-6 | 7 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 20 | 3 | Elegant Gadget | EG-20 | 10 |
| 9 | 3 | Luxury Gizmo | LG-9 | 4 |
| 12 | 3 | Luxury Gizmo | LG-12 | 5 |
| 13 | 3 | Luxury Gizmo | LG-13 | 5 |
| 5 | 3 | Luxury Gizmo | LG-5 | 9 |
| 4 | 3 | Sleek Widget | SW-4 | 8 |
| 15 | 3 | Sleek Widget | SW-15 | 9 |
| 17 | 3 | Sleek Widget | SW-17 | 9 |

**Expected Output**

|  |  |  |
| --- | --- | --- |
| **category** | **title** | **total\_stock** |
| Clothing | Elegant Gadget | 18 |
| Clothing | Luxury Gizmo | 20 |
| Electronics | Sleek Widget | 16 |
| Home & Kitchen | Luxury Gizmo | 23 |
| Home & Kitchen | Sleek Widget | 26 |

Question - 45

**SQL: Antivirus Database Quarantine Report**

A prominent cyber-security company aims to generate a report detailing quarantined URLs. This report should list URL domains, their associated threat types, the total number of times they have been quarantined, and the total number of users affected by each threat from a domain.

The result should have the following columns: *domain\_name | threat\_type | total\_occurrences | total\_users\_affected*. *domain\_name* - the website's domain (e.g., example.com)

*threat\_type* - the type of threat identified (e.g., Malware, Phishing)

*total\_occurrences* - the number of times this domain was quarantined for the specific threat *total\_users\_affected* - the number of users affected by each threat from the domain

The result should be sorted in descending order by *total\_users\_affected*, then ascending by *domain\_name*.

# Note:

Only URLs with a status of "Quarantined" should be included in the report.

**Schema**

**threat\_types**

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **constraint** | **description** |
| id | INT | PRIMARY KEY | Threat type ID |
| threat\_type | VARCHAR(255) |  | Type of threat (e.g., Malware, Phishing) |

|  |  |  |  |
| --- | --- | --- | --- |
| **quarantine\_urls** | | | |
| **name** | **type** | **constraint** | **description** |
| id | INT | PRIMARY KEY | URL ID |
| threat\_id | INT | FOREIGN KEY(threat\_id => threat\_types.id) | Reference to the type of threat |
| domain\_name | VARCHAR(255) |  | Domain name of the quarantined URL |
| status | ENUM('Quarantined','Safe','Deleted') |  | URL status in the system |
| users\_affected | INT |  | Number of users affected by the quarantined URL |

**Sample Data Tables**

|  |  |
| --- | --- |
| **threat\_types** | |
| **id** | **threat\_type** |
| 1 | Phishing |
| 2 | Rootkit |
| 3 | Malware |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **quarantine\_urls** | | | | |
| **id** | **threat\_id** | **domain\_name** | **status** | **users\_affected** |
| 17 | 1 | amazon.com | Quarantined | 862 |
| 16 | 1 | google.com | Quarantined | 63 |
| 9 | 1 | amazon.com | Quarantined | 41 |
| 18 | 2 | amazon.com | Quarantined | 149 |
| 12 | 2 | yahoo.com | Quarantined | 967 |
| 4 | 3 | amazon.com | Quarantined | 377 |
| 10 | 3 | yahoo.com | Quarantined | 721 |
| 11 | 1 | yahoo.com | Deleted | 551 |
| 20 | 1 | amazon.com | Safe | 407 |
| 19 | 1 | amazon.com | Deleted | 665 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 15 | 1 | facebook.com | Safe | 52 |
| 2 | 1 | google.com | Safe | 309 |
| 1 | 2 | twitter.com | Safe | 562 |
| 13 | 2 | facebook.com | Safe | 208 |
| 14 | 2 | google.com | Deleted | 731 |
| 8 | 2 | twitter.com | Safe | 924 |
| 7 | 2 | twitter.com | Safe | 982 |
| 6 | 2 | google.com | Deleted | 864 |
| 3 | 2 | facebook.com | Safe | 136 |
| 5 | 3 | yahoo.com | Safe | 949 |

**Expected Output**

|  |  |  |  |
| --- | --- | --- | --- |
| **domain\_name** | **threat\_type** | **total\_occurrences** | **total\_users\_affected** |
| yahoo.com | Rootkit | 1 | 967 |
| amazon.com | Phishing | 2 | 903 |
| yahoo.com | Malware | 1 | 721 |
| amazon.com | Malware | 1 | 377 |
| amazon.com | Rootkit | 1 | 149 |
| google.com | Phishing | 1 | 63 |

Question - 46

**SQL: Online Streaming Service Traffic Report**

Create a query for an online streaming service. It should return a list of clients, their number of streams, and the total amount of traffic.

The result should have the following columns: *mac\_address | streams | total\_traffic*. *mac\_address* - client MAC address

*streams* - total number of streams for a specific client

*total\_traffic* - total traffic amount of all streams for a specific client The result should be sorted in descending order by *total\_traffic*.

# Note:

Only streams of "720p" quality or higher should be included in the result.

**Schema**

|  |  |  |  |
| --- | --- | --- | --- |
| **clients** | | | |
| **name** | **type** | **constraint** | **description** |
| id | INT | PRIMARY KEY | Client ID |
| mac\_address | VARCHAR(255) |  | MAC address |

|  |  |  |  |
| --- | --- | --- | --- |
| **streams** | | | |
| **name** | **type** | **constraint** | **description** |
| client\_id | INT | FOREIGN KEY (client\_id => clients.id) | Client ID |
| title | VARCHAR(255) |  | Title |
| quality | ENUM('240p','360p','480p','720p','1080p','1440p','2160p') |  | Quality |
| traffic | INT |  | Traffic |

**Sample Data Tables**

|  |  |
| --- | --- |
| **clients** | |
| **id** | **mac\_address** |
| 1 | 2F-80-8E-F2-0E-4C |
| 2 | A1-F7-D4-48-B9-E6 |
| 3 | 9F-72-DB-7C-73-FC |

|  |  |  |  |
| --- | --- | --- | --- |
| **streams** | | | |
| **client\_id** | **title** | **quality** | **traffic** |
| 1 | Monte Carlo | 360p | 71928308 |
| 1 | Separation, The (Sparation, La) | 480p | 35221785 |
| 1 | Felidae | 480p | 54617023 |
| 1 | Dirty Dancing | 1440p | 56419563 |
| 1 | Ragtime | 1440p | 12404457 |
| 1 | Oscar | 1440p | 49717246 |
| 1 | Barb Wire | 2160p | 83761463 |
| 1 | Jason and the Argonauts | 2160p | 27364051 |
| 2 | Carry on Cruising | 240p | 33226462 |

|  |  |  |  |
| --- | --- | --- | --- |
| 2 | Best of the Best | 240p | 62793858 |
| 2 | Ecstasy (xtasis) | 240p | 73079415 |
| 2 | Go Go Tales | 480p | 48836837 |
| 2 | Nights and Weekends | 1440p | 32708277 |
| 3 | Coneheads | 480p | 92308213 |
| 3 | Silences of the Palace, The (Saimt el Qusur) | 480p | 52917945 |
| 3 | Good Pick | 720p | 71890218 |
| 3 | Wuthering Heights | 720p | 19813053 |
| 3 | Big Kahuna, The | 1080p | 28786846 |
| 3 | Work of Director Michel Gondry, The | 2160p | 18789351 |
| 3 | My Best Friends | 2160p | 44347338 |

**Expected Output**

|  |  |  |
| --- | --- | --- |
| **mac\_address** | **streams** | **total\_traffic** |
| 2F-80-8E-F2-0E-4C | 5 | 229666780 |
| 9F-72-DB-7C-73-FC | 5 | 183626806 |
| A1-F7-D4-48-B9-E6 | 1 | 32708277 |

Question - 47

**SQL: Cloud Hosting Instances Performance Statistics**

A cloud hosting provider dashboard that is in development requires a query that returns a list of networks where at least one instance within them has reached its maximum CPU usage threshold.

The result should have the following columns: *cidr | instances | avg\_cpu\_usage | avg\_memory\_usage | avg\_network\_usage*. *cidr* - network CIDR

*instances* - total number of instances for a specific network

*avg\_cpu\_usage* - average CPU usage of instances for a specific network

 Record format is ##% , where the placeholders are ## in the order they appear:

1. Average CPU usage across all instances of a specific network, rounded up to the nearest integer *avg\_memory\_usage* - average memory usage of instances for a specific network

 Record format is ##% , where the placeholders are ## in the order they appear:

1. Average memory usage across all instances of a specific network, rounded up to the nearest integer *avg\_network\_usage* - average network usage of instances for a specific network

 Record format is ##% , where the placeholders are ## in the order they appear:

1. Average network usage across all instances of a specific network, rounded up to the nearest integer The result should be sorted in ascending order by *cidr*.

# Note:

Only include networks where there is at least one instance with *cpu\_usage* of 80% or greater.

**Schema**

|  |  |  |  |
| --- | --- | --- | --- |
| **networks** | | | |
| **name** | **type** | **constraint** | **description** |
| id | INT | PRIMARY KEY | Network ID |
| cidr | VARCHAR(255) |  | IP Address v4 CIDR |

|  |  |  |  |
| --- | --- | --- | --- |
| **instances** | | | |
| **name** | **type** | **constraint** | **description** |
| network\_id | INT | FOREIGN KEY (network\_id => networks.id) | Network ID |
| cpu\_usage | VARCHAR(255) |  | CPU usage |
| memory\_usage | VARCHAR(255) |  | Memory usage |
| network\_usage | VARCHAR(255) |  | Network usage |

**Sample Data Tables**

|  |  |
| --- | --- |
| **networks** | |
| **id** | **cidr** |
| 1 | 24.77.36.156/9 |
| 2 | 74.213.138.70/7 |
| 3 | 167.244.163.58/29 |

|  |  |  |  |
| --- | --- | --- | --- |
| **instances** | | | |
| **network\_id** | **cpu\_usage** | **memory\_usage** | **network\_usage** |
| 1 | 20% | 74% | 74% |
| 3 | 26% | 9% | 99% |
| 3 | 2% | 21% | 97% |
| 1 | 51% | 19% | 89% |
| 2 | 2% | 27% | 79% |
| 3 | 92% | 35% | 41% |

|  |  |  |  |
| --- | --- | --- | --- |
| 2 | 27% | 5% | 44% |
| 3 | 67% | 47% | 79% |
| 1 | 14% | 28% | 43% |
| 3 | 47% | 0% | 53% |
| 1 | 38% | 3% | 46% |
| 2 | 71% | 51% | 6% |
| 3 | 77% | 74% | 53% |
| 3 | 31% | 48% | 80% |
| 2 | 31% | 42% | 24% |
| 1 | 77% | 65% | 46% |
| 2 | 51% | 94% | 41% |
| 3 | 8% | 3% | 57% |
| 1 | 1% | 56% | 62% |
| 2 | 15% | 66% | 65% |

**Expected Output**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **cidr** | **instances** | **avg\_cpu\_usage** | **avg\_memory\_usage** | **avg\_network\_usage** |
| 167.244.163.58/29 | 8 | 44% | 30% | 70% |

Question - 48

**SQL: AI Video Processing Service Usage Time Calculation**

A video processing service in development uses AI to improve the quality of the input video. The service uses a process separation mechanism to improve performance and minimize overall execution time.

Create a query that returns a list of all tasks and the total execution time for all processes of those tasks.

The result should have the following columns: *hash | usage\_time*. *hash* - task hash

*usage\_time* - total execution time in seconds of all processes for a specific task

The result should be sorted in descending order by *usage\_time*.

**Schema**

|  |  |  |  |
| --- | --- | --- | --- |
| **tasks** | | | |
| **name** | **type** | **constraint** | **description** |
| id | INT | PRIMARY KEY | Task ID |
| hash | VARCHAR(255) |  | Hash |

|  |  |  |  |
| --- | --- | --- | --- |
| **processes** | | | |
| **name** | **type** | **constraint** | **description** |
| task\_id | INT | FOREIGN KEY (task\_id => tasks.id) | Task ID |
| start\_dt | VARCHAR(19) |  | Start timestamp |
| end\_dt | VARCHAR(19) |  | End timestamp |

**Sample Data Tables**

|  |  |
| --- | --- |
| **tasks** | |
| **id** | **hash** |
| 1 | 208f95e0fcff792f617ade3cebf33ad9 |
| 2 | 0f44a9ffead2f18a7f25425c1260fc74 |
| 3 | dbcf54e94395c32e01ec09a5db731912 |

|  |  |  |
| --- | --- | --- |
| **processes** | | |
| **task\_id** | **start\_dt** | **end\_dt** |
| 1 | 2023-04-20 02:01:16 | 2023-04-20 02:11:35 |
| 1 | 2023-04-09 15:11:10 | 2023-04-09 15:26:43 |
| 1 | 2023-04-07 23:41:49 | 2023-04-08 00:34:10 |
| 2 | 2023-04-07 23:05:47 | 2023-04-08 00:00:05 |
| 2 | 2023-04-19 18:39:33 | 2023-04-19 18:54:57 |
| 2 | 2023-04-28 13:17:11 | 2023-04-28 13:24:37 |
| 2 | 2023-04-16 00:13:06 | 2023-04-16 01:02:39 |
| 2 | 2023-04-16 15:02:26 | 2023-04-16 15:58:14 |
| 2 | 2023-04-27 02:23:07 | 2023-04-27 02:59:13 |
| 2 | 2023-04-10 23:33:47 | 2023-04-11 00:09:35 |
| 2 | 2023-04-16 17:29:51 | 2023-04-16 18:10:22 |

|  |  |  |
| --- | --- | --- |
| 2 | 2023-04-23 12:16:01 | 2023-04-23 12:48:07 |
| 3 | 2023-04-01 02:25:12 | 2023-04-01 02:49:26 |
| 3 | 2023-04-04 03:02:43 | 2023-04-04 03:42:03 |
| 3 | 2023-04-10 22:42:26 | 2023-04-10 23:14:42 |
| 3 | 2023-04-09 17:46:12 | 2023-04-09 18:10:19 |
| 3 | 2023-04-25 15:09:36 | 2023-04-25 15:19:54 |
| 3 | 2023-04-19 14:39:52 | 2023-04-19 15:21:23 |
| 3 | 2023-04-12 04:22:29 | 2023-04-12 04:25:10 |
| 3 | 2023-04-25 07:40:26 | 2023-04-25 08:01:30 |

**Expected Output**

|  |  |
| --- | --- |
| **hash** | **usage\_time** |
| 0f44a9ffead2f18a7f25425c1260fc74 | 19620 |
| dbcf54e94395c32e01ec09a5db731912 | 11731 |
| 208f95e0fcff792f617ade3cebf33ad9 | 4693 |

Question - 49

**SQL: Benchmarking Tool Report**

A benchmarking tool that is under development needs a query that returns all devices and information about the achieved performance class.

The result should have the following columns: *device*. *device* - device benchmark result:

 Record format is Device ## has class: ## , where the placeholders are ## in the order they appear:

1. Device ID
2. The benchmark performance evaluation class, which can be calculated as: "A", when score ≥ 80

"B", when 60 ≤ score < 80 "C", when 40 ≤ score < 60 "D", when 20 ≤ score < 40 "F", when score < 20

The result should be sorted in ascending order by device ID.

**Schema**

**devices**

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **constraint** | **description** |
| id | INT | PRIMARY KEY | Device ID |
| score | INT |  | Score |

**Sample Data Tables**

|  |  |
| --- | --- |
| **devices** | |
| **id** | **score** |
| 1 | 20 |
| 2 | 50 |
| 3 | 50 |
| 4 | 68 |
| 5 | 95 |

**Expected Output**

|  |
| --- |
| **device** |
| Device 1 has class: D |
| Device 2 has class: C |
| Device 3 has class: C |
| Device 4 has class: B |
| Device 5 has class: A |

Question - 50

**SQL: Smart Home Application Customer Report**

As part of a smart home application, create a query that, based on data from meter readings, calculates the total electricity consumption, the name of the most expensive tariff consumed, and the total cost for invoicing.

The result should have the following columns: *username | email | highest\_tariff | consumption | total\_cost*. *username* - account username

*email* - account email address

*highest\_tariff* - name of the most expensive tariff that is consumed *consumption* - total consumption amount

*total\_cost* - total cost of all consumed electricity, rounded to two decimal places, e.g., 8.404 shows 8.40.

The results should be sorted in ascending order by *username*.

# Note:

The total cost of all electricity consumed is the sum of the products of all meter readings and their respective consumed tariffs.

**Schema**

|  |  |  |  |
| --- | --- | --- | --- |
| **accounts** | | | |
| **name** | **type** | **constraint** | **description** |
| id | INT | PRIMARY KEY | Account ID |
| username | VARCHAR(255) |  | Account username |
| email | VARCHAR(255) |  | Account email address |

|  |  |  |  |
| --- | --- | --- | --- |
| **tariffs** | | | |
| **name** | **type** | **constraint** | **description** |
| id | INT | PRIMARY KEY | Tariff ID |
| name | ENUM('A','B','C','D','E') |  | Tariff name |
| cost | DECIMAL(4,3) |  | Tariff cost |

|  |  |  |  |
| --- | --- | --- | --- |
| **readings** | | | |
| **name** | **type** | **constraint** | **description** |
| account\_id | INT | FOREIGN KEY (account\_id => accounts.id) | Account ID |
| tariff\_id | INT | FOREIGN KEY (tariff\_id => tariffs.id) | Tariff ID |
| amount | SMALLINT |  | Readings amount |

**Sample Data Tables**

|  |  |  |
| --- | --- | --- |
| **accounts** | | |
| **id** | **username** | **email** |
| 1 | hshillabeare0 | [rcalkin0@sourceforge.net](mailto:rcalkin0@sourceforge.net) |
| 2 | sdandy1 | [agaule1@businessweek.com](mailto:agaule1@businessweek.com) |
| 3 | sgreiswood2 | [toppy2@lulu.com](mailto:toppy2@lulu.com) |

|  |  |  |
| --- | --- | --- |
| **tariffs** | | |
| **id** | **name** | **cost** |

|  |  |  |
| --- | --- | --- |
| 1 | A | 0.010 |
| 2 | B | 0.020 |
| 3 | C | 0.050 |
| 4 | D | 0.075 |
| 5 | E | 0.100 |

|  |  |  |
| --- | --- | --- |
| **readings** | | |
| **account\_id** | **tariff\_id** | **amount** |
| 1 | 2 | 54 |
| 1 | 3 | 19 |
| 1 | 3 | 37 |
| 1 | 3 | 89 |
| 1 | 3 | 119 |
| 2 | 1 | 12 |
| 2 | 1 | 44 |
| 2 | 1 | 81 |
| 2 | 2 | 60 |
| 2 | 2 | 164 |
| 2 | 2 | 199 |
| 2 | 3 | 79 |
| 2 | 5 | 186 |
| 3 | 1 | 31 |
| 3 | 1 | 59 |
| 3 | 1 | 77 |
| 3 | 1 | 95 |
| 3 | 1 | 110 |
| 3 | 1 | 125 |
| 3 | 2 | 31 |

**Expected Output**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **username** | **email** | **highest\_tariff** | **consumption** | **total\_cost** |
| hshillabeare0 | [rcalkin0@sourceforge.net](mailto:rcalkin0@sourceforge.net) | C | 318 | 14.28 |
| sdandy1 | [agaule1@businessweek.com](mailto:agaule1@businessweek.com) | E | 825 | 32.38 |
| sgreiswood2 | [toppy2@lulu.com](mailto:toppy2@lulu.com) | B | 528 | 5.59 |

Question - 51

**SQL: MMORPG Game Inventory Overload Notification**

An MMORPG game is under development. For the profile and inventory mechanics, it needs a query that calculates a list of game accounts whose inventory is overloaded with game items.

The result should have the following columns: *username | email | items | total\_weight*. *username* - account username

*email* - account email address

*items* - total number of items in inventory *total\_weight* - total weight of items in inventory

The result should be sorted in descending order by *total\_weight*, then in ascending order by *username*.

# Note:

Each item in the inventory has its own weight.

Only accounts where the total weight of all items in the inventory exceeds the overload threshold should be included in the result. The overload threshold is 20.

**Schema**

|  |  |  |  |
| --- | --- | --- | --- |
| **accounts** | | | |
| **name** | **type** | **constraint** | **description** |
| id | INT | PRIMARY KEY | Account ID |
| username | VARCHAR(255) |  | Account username |
| email | VARCHAR(255) |  | Account email address |

|  |  |  |  |
| --- | --- | --- | --- |
| **items** | | | |
| **name** | **type** | **constraint** | **description** |
| id | INT | PRIMARY KEY | Item ID |
| type | ENUM('sword','shield','armor') |  | Item type |
| name | VARCHAR(255) |  | Item name |
| weight | SMALLINT |  | Item weight |

|  |  |  |  |
| --- | --- | --- | --- |
| **accounts\_items** | | | |
| **name** | **type** | **constraint** | **description** |
| account\_id | INT | FOREIGN KEY (account\_id => accounts.id) | Account ID |
| item\_id | INT | FOREIGN KEY (item\_id => items.id) | Item ID |

**Sample Data Tables**

|  |  |  |
| --- | --- | --- |
| **accounts** | | |
| **id** | **username** | **email** |
| 1 | esoane0 | [alefwich0@nytimes.com](mailto:alefwich0@nytimes.com) |
| 2 | jrafter1 | [bmcniff1@census.gov](mailto:bmcniff1@census.gov) |
| 3 | rcawston2 | [fnickoll2@flickr.com](mailto:fnickoll2@flickr.com) |

|  |  |  |  |
| --- | --- | --- | --- |
| **items** | | | |
| **id** | **type** | **name** | **weight** |
| 1 | shield | Shield of Asteraceae | 3 |
| 2 | sword | Sword of Cyperaceae | 3 |
| 3 | shield | Shield of Apiaceae | 3 |
| 4 | sword | Sword of Onagraceae | 3 |
| 5 | sword | Sword of Campanulaceae | 3 |

|  |  |
| --- | --- |
| **accounts\_items** | |
| **account\_id** | **item\_id** |
| 1 | 2 |
| 1 | 3 |
| 1 | 3 |
| 1 | 4 |
| 1 | 4 |
| 1 | 5 |
| 1 | 5 |
| 1 | 5 |

|  |  |
| --- | --- |
| 2 | 1 |
| 2 | 1 |
| 2 | 2 |
| 2 | 2 |
| 2 | 2 |
| 2 | 2 |
| 2 | 2 |
| 2 | 3 |
| 2 | 3 |
| 2 | 5 |
| 3 | 3 |
| 3 | 4 |

**Expected Output**

|  |  |  |  |
| --- | --- | --- | --- |
| **username** | **email** | **items** | **total\_weight** |
| jrafter1 | [bmcniff1@census.gov](mailto:bmcniff1@census.gov) | 10 | 30 |
| esoane0 | [alefwich0@nytimes.com](mailto:alefwich0@nytimes.com) | 8 | 24 |

Question - 52

**SQL: Outdoor Banner Digital Marketplace Placement Report**

In developing a digital marketplace for outdoor banners, a query is needed to return information about the banners in each of the cities.

The result should have the following columns: *city | banners | min\_area | avg\_area | max\_area | total\_area*. *city* - city name

*banners* - total number of banners for a specific city

*min\_area* - minimum available banner area for a specific city

*avg\_area* - average banner area for a specific city, rounded up to the nearest integer, e.g., *ceiling(1.1) = 2 max\_area* - maximum available banner area for a specific city

*total\_area* - total available banner area for a specific city The result should be sorted in ascending order by *city*.

# Note:

Banners are rectangular.

**Schema**

|  |  |  |  |
| --- | --- | --- | --- |
| **cities** | | | |
| **name** | **type** | **constraint** | **description** |
| id | INT | PRIMARY KEY | City ID |
| name | VARCHAR(255) |  | City name |

|  |  |  |  |
| --- | --- | --- | --- |
| **banners** | | | |
| **name** | **type** | **constraint** | **description** |
| city\_id | INT | FOREIGN KEY (city\_id => cities.id) | City ID |
| width | SMALLINT |  | Banner width |
| height | SMALLINT |  | Banner height |

**Sample Data Tables**

|  |  |
| --- | --- |
| **cities** | |
| **id** | **name** |
| 1 | Kayu Agung |
| 2 | Yangkou |
| 3 | Marseille |

|  |  |  |
| --- | --- | --- |
| **banners** | | |
| **city\_id** | **width** | **height** |
| 3 | 6 | 20 |
| 1 | 20 | 14 |
| 1 | 6 | 17 |
| 1 | 15 | 6 |
| 2 | 16 | 8 |
| 2 | 6 | 7 |
| 3 | 6 | 9 |
| 1 | 20 | 16 |
| 3 | 19 | 14 |
| 2 | 9 | 17 |

|  |  |  |
| --- | --- | --- |
| 2 | 8 | 12 |
| 1 | 12 | 16 |
| 3 | 15 | 14 |
| 3 | 11 | 7 |
| 3 | 6 | 14 |
| 2 | 12 | 7 |
| 3 | 7 | 20 |
| 1 | 13 | 6 |
| 3 | 10 | 13 |
| 2 | 19 | 15 |

**Expected Output**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **city** | **banners** | **min\_area** | **avg\_area** | **max\_area** | **total\_area** |
| Kayu Agung | 6 | 78 | 177 | 320 | 1062 |
| Marseille | 8 | 54 | 136 | 266 | 1081 |
| Yangkou | 6 | 42 | 132 | 285 | 788 |

Question - 53

**SQL: Auction Web Service Lot Statistics**

An auction web service needs a query that returns a list of all available lots with offers left by buyers.

The result should have the following columns: *name | min\_offer | avg\_offer | max\_offer | offers*. *name* - lot name

*offers* - total number of offers for a specific lot

*min\_offer* - minimum offer for a specific lot

*avg\_offer* - average offer for a specific lot, rounded to two decimal places *max\_offer* - maximum offer for a specific lot

The result should be sorted in descending order by *offers*.

# Notes:

Some lots may not have offers.

Values for *min\_offer, avg\_offer,* and *max\_offer* should always show two places after the decimal, e.g., 10 is shown as 10.00.

**Schema**

|  |  |  |  |
| --- | --- | --- | --- |
| **lots** | | | |
| **name** | **type** | **constraint** | **description** |
| id | INT | PRIMARY KEY | Lot ID |
| name | VARCHAR(255) |  | Lot name |

|  |  |  |  |
| --- | --- | --- | --- |
| **offers** | | | |
| **name** | **type** | **constraint** | **description** |
| lot\_id | INT | FOREIGN KEY (lot\_id => lots.id) | Lot ID |
| amount | DECIMAL(6,2) |  | Offer amount |

**Sample Data Tables**

|  |  |
| --- | --- |
| **lots** | |
| **id** | **name** |
| 1 | Merremia quinquefolia (L.) Hallier f. |
| 2 | Plantago maritima L. |
| 3 | Hohenbergia antillana Mez |
| 4 | Penstemon eriantherus Pursh var. argillosus M.E. Jones |

|  |  |
| --- | --- |
| **offers** | |
| **lot\_id** | **amount** |
| 1 | 510.51 |
| 2 | 703.80 |
| 2 | 181.80 |
| 1 | 38.06 |
| 2 | 368.78 |
| 3 | 91.40 |
| 2 | 413.80 |
| 3 | 157.99 |
| 3 | 885.82 |
| 2 | 863.99 |
| 1 | 307.61 |

|  |  |
| --- | --- |
| 2 | 120.39 |
| 1 | 771.96 |
| 2 | 801.42 |
| 3 | 871.59 |
| 1 | 541.61 |
| 3 | 477.62 |
| 2 | 303.29 |
| 2 | 612.83 |
| 3 | 464.98 |

**Expected Output**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **name** | **offers** | **min\_offer** | **avg\_offer** | **max\_offer** |
| Plantago maritima L. | 9 | 120.39 | 485.57 | 863.99 |
| Hohenbergia antillana Mez | 6 | 91.40 | 491.57 | 885.82 |
| Merremia quinquefolia (L.) Hallier f. | 5 | 38.06 | 433.95 | 771.96 |
| Penstemon eriantherus Pursh var. argillosus M.E. Jones | 0 | NULL | NULL | NULL |

Question - 54

**SQL: Tax Calculator Web Service Simple Report**

Create a query for a tax calculator web service. It should return a list of all accounts, the total taxable income, the account's individual tax rate, and the amount of tax payable.

The result should have the following columns: *full\_name | iban | income | rate | tax*. *full\_name* - full profile name:

 Record format is ## ## , where the placeholders are ## in the order they appear:

1. Last name of the profile
2. First name of the profile *iban* - account IBAN

*income* - total taxable income

*rate* - individual tax rate of the account

*tax* - payable amount of tax, rounded to two decimal places The result should be sorted in ascending order by *full\_name*.

# Note:

The payable amount of tax is calculated as the total taxable income for the periods Q1'21-Q4'21 multiplied by the individual tax rate. The individual tax rate is 10% and applies to all accounts.

**Schema**

|  |  |  |  |
| --- | --- | --- | --- |
| **accounts** | | | |
| **name** | **type** | **constraint** | **description** |
| id | INT | PRIMARY KEY | Account ID |
| first\_name | VARCHAR(255) |  | Account first name |
| last\_name | VARCHAR(255) |  | Account last name |
| iban | VARCHAR(255) |  | Account IBAN |

|  |  |  |  |
| --- | --- | --- | --- |
| **declarations** | | | |
| **name** | **type** | **constraint** | **description** |
| account\_id | INT | FOREIGN KEY (account\_id => accounts.id) | Account ID |
| quarter | ENUM('Q1','Q2','Q3','Q4') |  | Declaration quarter |
| income | DECIMAL(7,2) |  | Declaration taxable income |

**Sample Data Tables**

|  |  |  |  |
| --- | --- | --- | --- |
| **accounts** | | | |
| **id** | **first\_name** | **last\_name** | **iban** |
| 1 | Alex | Cantua | IL29 9590 1551 0560 0553 712 |
| 2 | Chris | Lashmore | AZ54 CNUI 01DR XEXZ ASKY QM4W F8JI |
| 3 | Taylor | Blum | HR20 2041 7741 5014 9873 9 |
| 4 | Robin | Neachell | NL87 PPCD 0429 1849 92 |
| 5 | Drew | Barbier | FR72 7843 3990 42WM QC8P GVNV F78 |

|  |  |  |
| --- | --- | --- |
| **declarations** | | |
| **account\_id** | **quarter** | **income** |
| 1 | Q1 | 49235.67 |
| 1 | Q2 | 46653.11 |
| 1 | Q3 | 63739.99 |
| 1 | Q4 | 43222.54 |
| 2 | Q1 | 69743.50 |

|  |  |  |
| --- | --- | --- |
| 2 | Q2 | 29641.01 |
| 2 | Q3 | 97725.49 |
| 2 | Q4 | 91481.98 |
| 3 | Q1 | 68402.43 |
| 3 | Q2 | 12660.12 |
| 3 | Q3 | 59601.65 |
| 3 | Q4 | 54701.74 |
| 4 | Q1 | 55220.27 |
| 4 | Q2 | 87752.41 |
| 4 | Q3 | 44447.06 |
| 4 | Q4 | 45876.26 |
| 5 | Q1 | 42511.74 |
| 5 | Q2 | 22022.78 |
| 5 | Q3 | 88396.81 |
| 5 | Q4 | 67252.54 |

**Expected Output**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **full\_name** | **iban** | **income** | **rate** | **tax** |
| Barbier Drew | FR72 7843 3990 42WM QC8P GVNV F78 | 220183.87 | 10% | 22018.39 |
| Blum Taylor | HR20 2041 7741 5014 9873 9 | 195365.94 | 10% | 19536.59 |
| Cantua Alex | IL29 9590 1551 0560 0553 712 | 202851.31 | 10% | 20285.13 |
| Lashmore Chris | AZ54 CNUI 01DR XEXZ ASKY QM4W F8JI | 288591.98 | 10% | 28859.20 |
| Neachell Robin | NL87 PPCD 0429 1849 92 | 233296.00 | 10% | 23329.60 |

Question - 55

**SQL: Social Network Relationship Statistics**

A social network under development needs a query that returns all profiles and statistics about their relationships.

The result should have the following columns: *full\_name | email | total\_relations | approved\_relations | pending\_relations*. *full\_name* - full profile name:

 Record format is ## ##, where the placeholders are ## in the order they appear:

1. Last name of the profile
2. First name of the profile

*email* - email address

*total\_relations* - total number of relations for a specific profile *approved\_relations* - total number of "approved" relations for a specific profile *pending\_relations* - total number of "pending" relations for a specific profile

The result should be sorted in ascending order by *full\_name*.

# Note:

A relation is "approved" when *relations.is\_approved* is true A relation is "pending" when *relations.is\_approved* is false

**Schema**

|  |  |  |  |
| --- | --- | --- | --- |
| **profiles** | | | |
| **name** | **type** | **constraint** | **description** |
| id | INT | PRIMARY KEY | Profile ID |
| first\_name | VARCHAR(255) |  | Profile first name |
| last\_name | VARCHAR(255) |  | Profile last name |
| email | VARCHAR(255) |  | Profile email address |

|  |  |  |  |
| --- | --- | --- | --- |
| **relations** | | | |
| **name** | **type** | **constraint** | **description** |
| profile\_id | INT | FOREIGN KEY (profile\_id => profiles.id) | Profile ID |
| related\_to | VARCHAR(255) |  | Relation to |
| is\_approved | BOOLEAN |  | Relation approval flag |

**Sample Data Tables**

|  |  |  |  |
| --- | --- | --- | --- |
| **profiles** | | | |
| **id** | **first\_name** | **last\_name** | **email** |
| 1 | Shayne | Shilito | [sshilito0@ftc.gov](mailto:sshilito0@ftc.gov) |
| 2 | Shell | Shade | [sshade1@paginegialle.it](mailto:sshade1@paginegialle.it) |
| 3 | Nobie | Splain | [nsplain2@npr.org](mailto:nsplain2@npr.org) |

|  |  |  |
| --- | --- | --- |
| **relations** | | |
| **profile\_id** | **related\_to** | **is\_approved** |
| 1 | cbasinigazzii | 1 |

|  |  |  |
| --- | --- | --- |
| 1 | ldevered | 1 |
| 1 | edeniskeb | 1 |
| 1 | cstirland4 | 1 |
| 1 | ngooddiea | 1 |
| 1 | alockney7 | 1 |
| 1 | jsorrillj | 0 |
| 1 | bnodin3 | 0 |
| 1 | dwall2 | 0 |
| 1 | folivas1 | 0 |
| 2 | ksharland6 | 0 |
| 2 | pbarosch8 | 0 |
| 2 | smacieja9 | 0 |
| 2 | bbrasonf | 0 |
| 2 | dabrahartg | 0 |
| 3 | gaymer5 | 1 |
| 3 | rwoolcockse | 1 |
| 3 | egilyott0 | 1 |
| 3 | agillionc | 0 |
| 3 | fgribbinh | 0 |

**Expected Output**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **full\_name** | **email** | **total\_relations** | **approved\_relations** | **pending\_relations** |
| Shade Shell | [sshade1@paginegialle.it](mailto:sshade1@paginegialle.it) | 5 | 0 | 5 |
| Shilito Shayne | [sshilito0@ftc.gov](mailto:sshilito0@ftc.gov) | 10 | 6 | 4 |
| Splain Nobie | [nsplain2@npr.org](mailto:nsplain2@npr.org) | 5 | 3 | 2 |

Question - 56

**SQL: Online Banking Transactions**

A client requested a query for a dashboard in their online banking web service. It should return the count and the sum of transaction amounts for each customer in the current month.

The result should have the following columns: *iban | transactions | total*.

*iban* - account IBAN

*transactions* - total number of transactions *total* - the sum of the transaction amounts

The result should be sorted in descending order by *total*.

# Note:

Only transactions in the current month should be included in the result. The current month is September.

**Schema**

|  |  |  |  |
| --- | --- | --- | --- |
| **accounts** | | | |
| **name** | **type** | **constraint** | **description** |
| id | INT | PRIMARY KEY | Account ID |
| iban | VARCHAR(255) |  | Account IBAN |

|  |  |  |  |
| --- | --- | --- | --- |
| **transactions** | | | |
| **name** | **type** | **constraint** | **description** |
| account\_id | INT | FOREIGN KEY (account\_id => accounts.id) | Account ID |
| dt | DATETIME |  | Transaction date and time |
| amount | DECIMAL(5,2) |  | Transaction amount |

**Sample Data Tables**

|  |  |
| --- | --- |
| **accounts** | |
| **id** | **iban** |
| 1 | SE48 2961 2087 8112 2835 6438 |
| 2 | BE89 2286 5514 4847 |
| 3 | MU84 HRGV 2047 2584 5774 3195 856J PZ |

|  |  |  |
| --- | --- | --- |
| **transactions** | | |
| **account\_id** | **dt** | **amount** |
| 2 | 2022-09-25 19:24:50 | 75.06 |
| 2 | 2022-09-24 03:09:17 | 41.10 |

|  |  |  |
| --- | --- | --- |
| 1 | 2022-09-19 04:13:17 | 65.85 |
| 3 | 2022-09-30 07:18:29 | 44.57 |
| 1 | 2022-09-26 01:51:44 | 98.93 |
| 1 | 2022-08-28 02:51:04 | 60.42 |
| 1 | 2022-08-25 23:25:54 | 45.34 |
| 2 | 2022-09-09 11:00:48 | 11.05 |
| 3 | 2022-08-25 19:37:02 | 53.61 |
| 2 | 2022-09-23 09:44:05 | 89.18 |
| 1 | 2022-08-28 19:48:40 | 47.60 |
| 3 | 2022-09-12 10:28:10 | 96.40 |
| 3 | 2022-10-03 16:49:51 | 45.41 |
| 2 | 2022-09-05 16:20:41 | 46.78 |
| 3 | 2022-10-03 04:51:29 | 50.81 |
| 1 | 2022-09-10 17:31:44 | 78.72 |
| 2 | 2022-08-31 21:59:56 | 61.09 |
| 2 | 2022-09-14 12:52:13 | 20.36 |
| 1 | 2022-09-28 11:05:21 | 70.52 |
| 3 | 2022-09-30 09:21:12 | 48.00 |

**Expected Output**

|  |  |  |
| --- | --- | --- |
| **iban** | **transactions** | **total** |
| SE48 2961 2087 8112 2835 6438 | 4 | 314.02 |
| BE89 2286 5514 4847 | 6 | 283.53 |
| MU84 HRGV 2047 2584 5774 3195 856J PZ | 3 | 188.97 |

Question - 57

**SQL: Internet Service Provider Monthly Report**

An ISP is updating its billing back end. They need a query that returns a list of all client MAC addresses, with traffic for the current billing period and its total cost. The current billing period includes all records where *dt* is in May 2022.

Since the business model of the ISP is significantly different from its competitors, each client has its own tariff per megabyte of data.

The result should have the following columns: *mac | traffic | cost*.

*mac* - MAC address of the client

*traffic* - total traffic for a specific client in the current billing period

*cost* - total cost of traffic for a specific client in the current billing period:

 Record is the sum of all traffic for a specific client in the current billing period, multiplied by the client's tariff and rounded to two decimal places The result should be sorted in descending order by *cost*.

# Note:

Only traffic whose reporting date is in May 2022 should be included in the report

**Schema**

|  |  |  |
| --- | --- | --- |
| **clients** | | |
| **name** | **type** | **description** |
| id | SMALLINT | ID |
| mac | VARCHAR(17) | MAC address |
| tariff | DECIMAL(6,5) | Tariff per MB |

|  |  |  |
| --- | --- | --- |
| **traffic** | | |
| **name** | **type** | **description** |
| client\_id | SMALLINT | Client ID |
| dt | VARCHAR(19) | Reporting date |
| amount | INT | Amount |

**Sample Data Tables**

|  |  |  |
| --- | --- | --- |
| **clients** | | |
| **id** | **mac** | **tariff** |
| 1 | A2-53-FC-0C-3E-B4 | 0.00007 |
| 2 | DC-80-42-E9-AE-FC | 0.00003 |
| 3 | 3F-9B-A9-2A-B1-7B | 0.00007 |
| 4 | D4-6F-E4-AF-47-D5 | 0.00004 |
| 5 | B9-65-FC-8E-F0-15 | 0.00007 |

|  |  |  |
| --- | --- | --- |
| **traffic** | | |
| **client\_id** | **dt** | **amount** |

|  |  |  |
| --- | --- | --- |
| 1 | 2022-05-22 | 9127 |
| 1 | 2022-06-07 | 62203 |
| 1 | 2022-06-10 | 88227 |
| 2 | 2022-05-31 | 99874 |
| 2 | 2022-06-01 | 78400 |
| 2 | 2022-06-03 | 61106 |
| 2 | 2022-06-12 | 20963 |
| 2 | 2022-06-29 | 98304 |
| 2 | 2022-07-04 | 6626 |
| 3 | 2022-05-22 | 8386 |
| 3 | 2022-06-08 | 22959 |
| 3 | 2022-07-05 | 52096 |
| 3 | 2022-07-14 | 70777 |
| 4 | 2022-05-22 | 93743 |
| 5 | 2022-05-16 | 84660 |
| 5 | 2022-05-28 | 63267 |
| 5 | 2022-06-10 | 80681 |
| 5 | 2022-06-21 | 55460 |
| 5 | 2022-07-04 | 91365 |
| 5 | 2022-07-09 | 23296 |

**Expected Output**

|  |  |  |
| --- | --- | --- |
| **mac** | **traffic** | **cost** |
| B9-65-FC-8E-F0-15 | 147927 | 10.35 |
| D4-6F-E4-AF-47-D5 | 93743 | 3.75 |
| DC-80-42-E9-AE-FC | 99874 | 3.00 |
| A2-53-FC-0C-3E-B4 | 9127 | 0.64 |
| 3F-9B-A9-2A-B1-7B | 8386 | 0.59 |

Question - 58

**SQL: The Yellow Pages Companies Report**

A company maintains "The Yellow Pages" website. They need a query that returns a list of companies and their overall rating based on all the categories in which they are listed.

The result should have the following columns: *name | address | phone | overall\_review\_rating*. *name* - company name

*address* - company address *phone* - company phone number

*overall\_review\_rating* - overall review rating for a specific company:

 Record format is ## (## categories) , where the placeholders are ## in the order they appear:

1. The average review rating across all categories for a specific company, rounded to one decimal place
2. The total number of categories in which the company is listed

The result should be sorted in descending order of the average rating (before rounding), then in ascending order by *name*.

**Schema**

|  |  |  |
| --- | --- | --- |
| **companies** | | |
| **name** | **type** | **description** |
| id | SMALLINT | unique id, primary key |
| name | VARCHAR(255) |  |
| address | VARCHAR(255) |  |
| phone | VARCHAR(255) |  |

|  |  |  |
| --- | --- | --- |
| **categories** | | |
| **name** | **type** | **description** |
| company\_id | SMALLINT | foreign key, companies.id |
| name | VARCHAR(255) |  |
| review\_rating | SMALLINT |  |

**Sample Data Tables**

|  |  |  |  |
| --- | --- | --- | --- |
| **companies** | | | |
| **id** | **name** | **address** | **phone** |
| 1 | Casper, Oberbrunner and Williamson | 53 Di Loreto Hill | +420 (569) 566-3689 |
| 2 | Tromp, Kozey and Abbott | 84 Mcguire Plaza | +62 (145) 722-2330 |
| 3 | Gerlach, Hayes and Stamm | 80 Service Point | +86 (731) 234-4119 |
| 4 | Wolff-Fadel | 06 Fair Oaks Trail | +7 (894) 233-0976 |

|  |  |  |  |
| --- | --- | --- | --- |
| 5 | Kihn-Cronin | 483 Nobel Road | +1 (396) 693-1661 |

|  |  |  |
| --- | --- | --- |
| **categories** | | |
| **company\_id** | **name** | **review\_rating** |
| 1 | HVAC | 2 |
| 2 | HVAC | 2 |
| 2 | Retaining Wall and Brick Pavers | 1 |
| 2 | Rebar & Wire Mesh Install | 2 |
| 3 | Prefabricated Aluminum Metal Canopies | 2 |
| 3 | Prefabricated Aluminum Metal Canopies | 0 |
| 3 | RF Shielding | 2 |
| 3 | Overhead Doors | 0 |
| 3 | Rebar & Wire Mesh Install | 5 |
| 3 | Termite Control | 0 |
| 4 | Sitework & Site Utilities | 0 |
| 4 | Electrical and Fire Alarm | 2 |
| 4 | Masonry | 2 |
| 4 | Temp Fencing, Decorative Fencing and Gates | 0 |
| 4 | Elevator | 1 |
| 4 | Drywall & Acoustical (FED) | 5 |
| 5 | Asphalt Paving | 0 |
| 5 | Glass & Glazing | 1 |
| 5 | Framing (Steel) | 3 |
| 5 | Structural & Misc Steel Erection | 1 |

**Expected Output**

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **address** | **phone** | **overall\_review\_rating** |
| Casper, Oberbrunner and Williamson | 53 Di Loreto Hill | +420 (569) 566-3689 | 2.0 (1 categories) |
| Tromp, Kozey and Abbott | 84 Mcguire Plaza | +62 (145) 722-2330 | 1.7 (3 categories) |
| Wolff-Fadel | 06 Fair Oaks Trail | +7 (894) 233-0976 | 1.7 (6 categories) |

|  |  |  |  |
| --- | --- | --- | --- |
| Gerlach, Hayes and Stamm | 80 Service Point | +86 (731) 234-4119 | 1.5 (6 categories) |
| Kihn-Cronin | 483 Nobel Road | +1 (396) 693-1661 | 1.3 (4 categories) |

Question - 59

**SQL: Domain Name Registrar Accounts Report**

A domain name registration service needs new reporting functionality. Create a query that returns a list of all active accounts, the number of unexpired domain names they have, and the nearest expiration date later than July 15, 2022.

The result should have the following columns: *username | domains | nearest\_expiration*. *username* - account username

*domains* - total number of domains for a specific account

*nearest\_expiration* - nearest expiration date for domains for a specific account The result should be sorted in ascending order by *username*.

# Note:

Only active accounts should be included in the report.

Only domain names that have not expired (with an expiration date greater than today) should be included in the report. Today is July 15, 2022.

**Schema**

|  |  |  |
| --- | --- | --- |
| **accounts** | | |
| **name** | **type** | **description** |
| id | SMALLINT | unique id, primary key |
| username | VARCHAR(255) |  |
| is\_active | SMALLINT | Account status: 1 = Active, 0 = Not active |

|  |  |  |
| --- | --- | --- |
| **domains** | | |
| **name** | **type** | **description** |
| account\_id | SMALLINT | foreign key, accounts.id |
| name | VARCHAR(255) |  |
| expiration\_date | VARCHAR(19) |  |

**Sample Data Tables**

|  |  |  |
| --- | --- | --- |
| **accounts** | | |
| **id** | **username** | **is\_active** |

|  |  |  |
| --- | --- | --- |
| 1 | obeedie0 | 0 |
| 2 | stopham1 | 1 |
| 3 | ndolder2 | 1 |
| 4 | jyanshinov3 | 1 |
| 5 | ewilflinger4 | 0 |

|  |  |  |
| --- | --- | --- |
| **domains** | | |
| **account\_id** | **name** | **expiration\_date** |
| 1 | imgur.com | 2022-05-14 |
| 1 | domainmarket.com | 2022-07-02 |
| 1 | comsenz.com | 2022-07-28 |
| 1 | gizmodo.com | 2022-08-09 |
| 1 | toplist.cz | 2022-08-15 |
| 1 | scientificamerican.com | 2022-09-03 |
| 1 | examiner.com | 2022-12-18 |
| 1 | photobucket.com | 2023-01-22 |
| 2 | merriam-webster.com | 2022-02-20 |
| 2 | tripod.com | 2022-08-08 |
| 3 | ca.gov | 2022-04-24 |
| 3 | ehow.com | 2022-06-28 |
| 3 | purevolume.com | 2022-07-01 |
| 3 | squidoo.com | 2022-10-27 |
| 3 | eepurl.com | 2022-12-21 |
| 4 | digg.com | 2022-05-14 |
| 4 | jugem.jp | 2022-08-05 |
| 4 | artisteer.com | 2022-10-21 |
| 5 | behance.net | 2022-03-24 |
| 5 | cnn.com | 2022-05-11 |

**Expected Output**

|  |  |  |
| --- | --- | --- |
| **username** | **domains** | **nearest\_expiration** |
| jyanshinov3 | 2 | 2022-08-05 |
| ndolder2 | 2 | 2022-10-27 |
| stopham1 | 1 | 2022-08-08 |

Question - 60

**SQL: Advertising Network Events Report**

As part of a marketing audit of your company's advertising campaigns published in the an online advertising network, they need a list of all advertising campaigns whose average event values as of July 15, 2022 are equal to or greater than 0.7.

The result should have the following columns: *campaign | events | average\_value*. *campaign* - campaign name

*events* - total number of events for a specific campaign

*average\_value* - average event value for a specific campaign, rounded to five decimal places (format 0.##### where each # is a digit) The result should be sorted in descending order by *average\_value*.

# Note:

Only events for July 15, 2022, should be included in the report

Only campaigns with average event values equal to or greater than 0.7 should be included in the report

**Schema**

|  |  |  |
| --- | --- | --- |
| **campaigns** | | |
| **name** | **type** | **description** |
| id | SMALLINT | unique id, primary key |
| name | VARCHAR(255) |  |

|  |  |  |
| --- | --- | --- |
| **events** | | |
| **name** | **type** | **description** |
| campaign\_id | SMALLINT | foreign key campaign.id |
| dt | VARCHAR(19) | Event datetime |
| value | DECIMAL(6,5) | Event value |

**Sample Data Tables**

**campaigns**

|  |  |
| --- | --- |
| **id** | **name** |
| 1 | 11-080 - Registration Equipment |
| 2 | 12-700 - Systems Furniture |
| 3 | 9-900 - Paints and Coatings |

|  |  |  |
| --- | --- | --- |
| **events** | | |
| **campaign\_id** | **dt** | **value** |
| 1 | 2022-07-14 13:11:38 | 0.59275 |
| 1 | 2022-07-14 14:55:43 | 0.12928 |
| 1 | 2022-07-14 18:16:11 | 0.82350 |
| 1 | 2022-07-15 01:19:44 | 0.97144 |
| 1 | 2022-07-15 22:52:02 | 0.60728 |
| 1 | 2022-07-16 08:55:38 | 0.71158 |
| 1 | 2022-07-16 10:22:44 | 0.29627 |
| 2 | 2022-07-14 02:36:31 | 0.42323 |
| 2 | 2022-07-14 04:45:32 | 0.91077 |
| 2 | 2022-07-14 07:24:11 | 0.35956 |
| 2 | 2022-07-15 06:43:08 | 0.16662 |
| 2 | 2022-07-16 08:21:27 | 0.02559 |
| 2 | 2022-07-16 11:59:41 | 0.34606 |
| 2 | 2022-07-16 23:26:12 | 0.62697 |
| 3 | 2022-07-14 00:21:56 | 0.97297 |
| 3 | 2022-07-14 10:22:11 | 0.93894 |
| 3 | 2022-07-14 12:29:59 | 0.44633 |
| 3 | 2022-07-15 01:17:41 | 0.37531 |
| 3 | 2022-07-15 14:20:48 | 0.24872 |
| 3 | 2022-07-16 23:02:51 | 0.80594 |

**Expected Output**

|  |  |  |
| --- | --- | --- |
| **campaign** | **events** | **average\_value** |

|  |  |  |
| --- | --- | --- |
| 11-080 - Registration Equipment | 2 | 0.78936 |

Question - 61

**SQL: Ecommerce Deal Report**

An online shop needs a new deal history feature. Create a query that returns a list of the top three seller profiles with the highest total deals in June, 2022.

The result should have the following columns: *first\_name | last\_name | email | total*. *total* - the total amount of all deals for a specific profile

The result should be sorted in descending order by *total*. The result should be limited to the first three records.

# Note:

Only deals for June 2022 should be included in the report

**Schema**

|  |  |  |
| --- | --- | --- |
| **profiles** | | |
| **name** | **type** | **description** |
| id | SMALLINT | unique id, primary key |
| first\_name | VARCHAR(255) |  |
| last\_name | VARCHAR(255) |  |
| email | VARCHAR(255) |  |

|  |  |  |
| --- | --- | --- |
| **deals** | | |
| **name** | **type** | **description** |
| profile\_id | SMALLINT | foreign key into profile.id |
| dt | VARCHAR(19) | Deal datetime |
| amount | DECIMAL(5,2) | Deal amount |

**Sample Data Tables**

|  |  |  |  |
| --- | --- | --- | --- |
| **profiles** | | | |
| **id** | **first\_name** | **last\_name** | **email** |
| 1 | Wallis | Treadway | [wtreadway0@senate.gov](mailto:wtreadway0@senate.gov) |

|  |  |  |  |
| --- | --- | --- | --- |
| 2 | Franklin | Blackston | [fblackston1@parallels.com](mailto:fblackston1@parallels.com) |
| 3 | Honoria | Constant | [hconstant2@umich.edu](mailto:hconstant2@umich.edu) |
| 4 | Bertine | Hillaby | [bhillaby3@artisteer.com](mailto:bhillaby3@artisteer.com) |
| 5 | Constance | Knutsen | [cknutsen4@google.ca](mailto:cknutsen4@google.ca) |

|  |  |  |
| --- | --- | --- |
| **deals** | | |
| **profile\_id** | **dt** | **amount** |
| 5 | 2022-05-21 02:44:24 | 49.10 |
| 2 | 2022-05-22 23:26:59 | 46.21 |
| 1 | 2022-05-23 09:56:25 | 58.57 |
| 5 | 2022-05-28 02:38:08 | 27.81 |
| 4 | 2022-06-04 07:16:27 | 22.31 |
| 4 | 2022-06-04 14:15:03 | 36.33 |
| 5 | 2022-06-04 15:03:10 | 21.41 |
| 1 | 2022-06-07 02:58:06 | 92.84 |
| 4 | 2022-06-08 05:09:52 | 24.41 |
| 3 | 2022-06-13 03:28:52 | 61.55 |
| 4 | 2022-06-16 15:09:39 | 77.70 |
| 5 | 2022-06-18 16:51:32 | 58.79 |
| 4 | 2022-06-20 02:55:20 | 43.61 |
| 3 | 2022-06-22 06:52:10 | 10.41 |
| 1 | 2022-06-23 04:59:05 | 6.59 |
| 1 | 2022-06-30 16:11:02 | 43.07 |
| 4 | 2022-07-05 06:05:28 | 36.45 |
| 5 | 2022-07-12 07:49:51 | 14.76 |
| 4 | 2022-07-12 18:58:11 | 91.61 |
| 5 | 2022-07-14 00:50:45 | 69.61 |

**Expected Output**

|  |  |  |  |
| --- | --- | --- | --- |
| **first\_name** | **last\_name** | **email** | **total** |

|  |  |  |  |
| --- | --- | --- | --- |
| Bertine | Hillaby | [bhillaby3@artisteer.com](mailto:bhillaby3@artisteer.com) | 204.36 |
| Wallis | Treadway | [wtreadway0@senate.gov](mailto:wtreadway0@senate.gov) | 142.50 |
| Constance | Knutsen | [cknutsen4@google.ca](mailto:cknutsen4@google.ca) | 80.20 |

Question - 62

**SQL: Freelance Platform Candidate Review**

A company's HR department needs an SQL query against data from a freelance hiring platform. The result should have the following columns: *first\_name | last\_name | email | job\_success\_score*.

It should be sorted in descending order by *job\_success\_score*, then in ascending order by *first\_name* and *last\_name*. Limit the result to the first ten records.

# Note:

Only profiles that have a "JSS" (Job Success Score) greater than 90 should be included.

Only those profiles that have passed the verification process (*is\_verified* is 1*)* should be included.

**Schema**

|  |  |  |
| --- | --- | --- |
| **profiles** | | |
| **name** | **type** | **description** |
| id | SMALLINT | unique id, primary key |
| first\_name | VARCHAR(255) |  |
| last\_name | VARCHAR(255) |  |
| email | VARCHAR(255) |  |
| is\_verified | SMALLINT | 1 = True, 0 = False |

|  |  |  |
| --- | --- | --- |
| **stats** | | |
| **name** | **type** | **description** |
| profile\_id | SMALLINT | foreign key into profiles.id |
| job\_success\_score | SMALLINT |  |

**Sample Data Tables**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **profiles** | | | | |
| **id** | **first\_name** | **last\_name** | **email** | **is\_verified** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | Junia | Sehorsch | [jsehorsch0@oracle.com](mailto:jsehorsch0@oracle.com) | 0 |
| 2 | Dave | Halliburton | [dhalliburton1@pbs.org](mailto:dhalliburton1@pbs.org) | 1 |
| 3 | Agneta | Dutch | [adutch2@thetimes.co.uk](mailto:adutch2@thetimes.co.uk) | 1 |
| 4 | Kendell | Sylvester | [ksylvester3@canalblog.com](mailto:ksylvester3@canalblog.com) | 1 |
| 5 | Koralle | Ragsdale | [kragsdale4@buzzfeed.com](mailto:kragsdale4@buzzfeed.com) | 1 |
| 6 | Roma | Kenelin | [rkenelin5@prnewswire.com](mailto:rkenelin5@prnewswire.com) | 1 |
| 7 | Harold | Molloy | [hmolloy6@ycombinator.com](mailto:hmolloy6@ycombinator.com) | 1 |
| 8 | Berri | Hartzogs | [bhartzogs7@ox.ac.uk](mailto:bhartzogs7@ox.ac.uk) | 0 |
| 9 | Garrik | Preddle | [gpreddle8@topsy.com](mailto:gpreddle8@topsy.com) | 1 |
| 10 | Sophie | Messenger | [smessenger9@myspace.com](mailto:smessenger9@myspace.com) | 1 |
| 11 | Ashby | Philipsson | [aphilipssona@typepad.com](mailto:aphilipssona@typepad.com) | 1 |
| 12 | Kayle | Jesteco | [kjestecob@ocn.ne.jp](mailto:kjestecob@ocn.ne.jp) | 1 |
| 13 | Munroe | Chevolleau | [mchevolleauc@yandex.ru](mailto:mchevolleauc@yandex.ru) | 1 |
| 14 | Etheline | Choake | [echoaked@hao123.com](mailto:echoaked@hao123.com) | 1 |
| 15 | Marten | Zamboniari | [mzamboniarie@cbc.ca](mailto:mzamboniarie@cbc.ca) | 1 |
| 16 | Hersch | Blasdale | [hblasdalef@wunderground.com](mailto:hblasdalef@wunderground.com) | 1 |
| 17 | Jori | MacFaell | [jmacfaellg@va.gov](mailto:jmacfaellg@va.gov) | 1 |
| 18 | Margo | Finnemore | [mfinnemoreh@discovery.com](mailto:mfinnemoreh@discovery.com) | 1 |
| 19 | Felicle | Ramsdale | [framsdalei@devhub.com](mailto:framsdalei@devhub.com) | 1 |
| 20 | Demetris | Arnet | [darnetj@livejournal.com](mailto:darnetj@livejournal.com) | 1 |

|  |  |
| --- | --- |
| **stats** | |
| **profile\_id** | **job\_success\_score** |
| 4 | 100 |
| 1 | 100 |
| 18 | 95 |
| 5 | 95 |
| 8 | 95 |
| 10 | 95 |
| 12 | 95 |
| 13 | 95 |

|  |  |
| --- | --- |
| 14 | 95 |
| 20 | 90 |
| 17 | 90 |
| 7 | 90 |
| 15 | 90 |
| 11 | 85 |
| 16 | 85 |
| 9 | 85 |
| 6 | 85 |
| 2 | 85 |
| 3 | 75 |
| 19 | 75 |

**Expected Output**

|  |  |  |  |
| --- | --- | --- | --- |
| **first\_name** | **last\_name** | **email** | **job\_success\_score** |
| Kendell | Sylvester | [ksylvester3@canalblog.com](mailto:ksylvester3@canalblog.com) | 100 |
| Etheline | Choake | [echoaked@hao123.com](mailto:echoaked@hao123.com) | 95 |
| Kayle | Jesteco | [kjestecob@ocn.ne.jp](mailto:kjestecob@ocn.ne.jp) | 95 |
| Koralle | Ragsdale | [kragsdale4@buzzfeed.com](mailto:kragsdale4@buzzfeed.com) | 95 |
| Margo | Finnemore | [mfinnemoreh@discovery.com](mailto:mfinnemoreh@discovery.com) | 95 |
| Munroe | Chevolleau | [mchevolleauc@yandex.ru](mailto:mchevolleauc@yandex.ru) | 95 |
| Sophie | Messenger | [smessenger9@myspace.com](mailto:smessenger9@myspace.com) | 95 |
| Demetris | Arnet | [darnetj@livejournal.com](mailto:darnetj@livejournal.com) | 90 |
| Harold | Molloy | [hmolloy6@ycombinator.com](mailto:hmolloy6@ycombinator.com) | 90 |
| Jori | MacFaell | [jmacfaellg@va.gov](mailto:jmacfaellg@va.gov) | 90 |

Question - 63

**SQL: Virtual Machine Deployment Report**

While working on a new dashboard for a large cloud hosting company, an engineer needs a query that returns a list of CVM (Cloud Virtual Machine) configurations and their total number of deployments in 2021.

The result should have the following columns: *configuration | deployments*.

*configuration* - configuration name

*deployments* - the total number of all deployments for a specific configuration

# Note:

Only deployments in 2021 should be included in the report

The result should be sorted in descending order by *deployments*.

**Schema**

|  |  |  |
| --- | --- | --- |
| **configurations** | | |
| **name** | **type** | **description** |
| id | VARCHAR(64) | unique id, primary key |
| name | VARCHAR(255) |  |

|  |  |  |
| --- | --- | --- |
| **deployments** | | |
| **name** | **type** | **description** |
| configuration\_id | VARCHAR(64) | foreign key configurations.id |
| dt | VARCHAR(19) | deployment date and time |

**Sample Data Tables**

|  |  |
| --- | --- |
| **configurations** | |
| **id** | **name** |
| 1vcpu\_512mb\_10gb\_500gb | 1 CPU / 512 MB RAM / 10 GB SSD Disk / 500 GB transfer |
| 1vcpu\_1gb\_25gb\_1tb | 1 CPU / 1 GB RAM / 25 GB SSD Disk / 1000 GB transfer |
| 1vcpu\_2gb\_50gb\_2tb | 1 CPU / 2 GB RAM / 50 GB SSD Disk / 2 TB transfer |
| 2vcpu\_2gb\_60gb\_3tb | 2 CPUs / 2 GB RAM / 60 GB SSD Disk / 3 TB transfer |
| 2vcpu\_4gb\_80gb\_4tb | 2 CPUs / 4 GB RAM / 80 GB SSD Disk / 4 TB transfer |
| 4vcpu\_8gb\_160gb\_5tb | 4 CPUs / 8 GB RAM / 160 GB SSD Disk / 5 TB transfer |
| 8vcpu\_16gb\_320gb\_6tb | 8 CPUs / 16 GB RAM / 320 GB SSD Disk / 6 TB transfer |

|  |  |
| --- | --- |
| **deployments** | |
| **configuration\_id** | **dt** |
| 1vcpu\_512mb\_10gb\_500gb | 2020-10-22 05:59:47 |
| 1vcpu\_1gb\_25gb\_1tb | 2020-11-09 06:07:57 |

|  |  |
| --- | --- |
| 1vcpu\_2gb\_50gb\_2tb | 2020-12-02 14:47:24 |
| 2vcpu\_2gb\_60gb\_3tb | 2021-01-24 15:41:42 |
| 4vcpu\_8gb\_160gb\_5tb | 2021-01-25 09:31:37 |
| 2vcpu\_2gb\_60gb\_3tb | 2021-02-08 02:43:14 |
| 1vcpu\_512mb\_10gb\_500gb | 2021-03-25 06:13:36 |
| 8vcpu\_16gb\_320gb\_6tb | 2021-03-26 22:23:42 |
| 2vcpu\_2gb\_60gb\_3tb | 2021-05-24 09:48:16 |
| 1vcpu\_512mb\_10gb\_500gb | 2021-05-31 05:03:28 |
| 1vcpu\_2gb\_50gb\_2tb | 2021-08-25 22:24:10 |
| 2vcpu\_4gb\_80gb\_4tb | 2021-09-05 22:12:17 |
| 2vcpu\_4gb\_80gb\_4tb | 2021-09-23 08:31:50 |
| 2vcpu\_4gb\_80gb\_4tb | 2021-09-28 05:15:24 |
| 2vcpu\_2gb\_60gb\_3tb | 2021-10-14 08:26:20 |
| 1vcpu\_2gb\_50gb\_2tb | 2021-11-01 19:00:30 |
| 1vcpu\_512mb\_10gb\_500gb | 2021-11-26 10:53:05 |
| 1vcpu\_1gb\_25gb\_1tb | 2021-12-27 07:07:23 |
| 2vcpu\_2gb\_60gb\_3tb | 2022-02-13 06:00:29 |
| 2vcpu\_2gb\_60gb\_3tb | 2022-03-03 09:10:30 |

**Expected Output**

|  |  |
| --- | --- |
| **configuration** | **deployments** |
| 2 CPUs / 2 GB RAM / 60 GB SSD Disk / 3 TB transfer | 4 |
| 2 CPUs / 4 GB RAM / 80 GB SSD Disk / 4 TB transfer | 3 |
| 1 CPU / 512 MB RAM / 10 GB SSD Disk / 500 GB transfer | 3 |
| 1 CPU / 2 GB RAM / 50 GB SSD Disk / 2 TB transfer | 2 |
| 8 CPUs / 16 GB RAM / 320 GB SSD Disk / 6 TB transfer | 1 |
| 4 CPUs / 8 GB RAM / 160 GB SSD Disk / 5 TB transfer | 1 |
| 1 CPU / 1 GB RAM / 25 GB SSD Disk / 1000 GB transfer | 1 |

Question - 64

**SQL: Visitors Behavior Report**

You are working on developing visitor tracking software and need to get the total number of purchase events in May 2022. A purchase event has "buy" in the *type* field.

The result should have the following columns: *purchases*. *purchases* - the total number of events of the "buy" type

# Note:

Only events of the type "buy" should be included in the report.

Only events that occurred in May 2022 should be included in the report.

**Schema**

|  |  |  |
| --- | --- | --- |
| **events** | | |
| **name** | **type** | **description** |
| dt | VARCHAR(19) | Event timestamp |
| type | VARCHAR(64) | Event type |

**Sample Data Tables**

|  |  |
| --- | --- |
| **events** |  |
| **dt** | **type** |
| 2022-04-04 03:36:00 | buy |
| 2022-04-21 07:05:09 | buy |
| 2022-04-02 11:34:24 | sell |
| 2022-05-27 16:12:50 | buy |
| 2022-05-20 09:09:07 | buy |
| 2022-05-22 09:06:37 | buy |
| 2022-05-31 07:49:36 | buy |
| 2022-05-14 22:29:10 | buy |
| 2022-05-13 15:00:54 | sell |
| 2022-05-24 15:40:54 | sell |
| 2022-05-13 01:20:05 | sell |
| 2022-05-16 07:07:44 | sell |
| 2022-05-01 16:57:00 | sell |
| 2022-06-02 09:42:02 | buy |

|  |  |
| --- | --- |
| 2022-06-01 06:34:59 | buy |
| 2022-06-06 17:14:47 | buy |
| 2022-06-05 13:37:23 | buy |
| 2022-06-17 19:10:13 | buy |
| 2022-06-15 21:40:13 | sell |
| 2022-06-11 12:26:43 | sell |

**Expected Output**

|  |
| --- |
| **purchases** |
| 5 |

Question - 65

**SQL: Advertising Campaigns Report**

As part of the summary report on advertising campaigns, get a list of the top three companies whose campaigns have a positive return on their advertising budgets.

The result should have the following columns: *company\_name | profit*. *profit* is the campaign revenues less expenses

The result should be sorted in descending order by *profit*. The result should be limited to the three first records.

# Note:

Only companies whose campaigns have a positive return on their advertising budget should be included in the report

**Schema**

|  |  |  |
| --- | --- | --- |
| **companies** | | |
| **name** | **type** | **description** |
| id | SMALLINT | ID |
| name | VARCHAR(255) | Company name |

|  |  |  |
| --- | --- | --- |
| **campaigns** | | |
| **name** | **type** | **description** |
| company\_id | SMALLINT | Company ID |

|  |  |  |
| --- | --- | --- |
| expenses | DECIMAL(7,2) | Campaign expenses |
| revenue | DECIMAL(7,2) | Campaign revenue |

**Sample Data Tables**

|  |  |
| --- | --- |
| **companies** | |
| **id** | **name** |
| 1 | Lion Biotechnologies, Inc. |
| 2 | Boston Private Financial Holdings, Inc. |
| 3 | Universal Corporation |
| 4 | Arbutus Biopharma Corporation |
| 5 | Royal Bank Of Canada |
| 6 | Penn West Petroleum Ltd |
| 7 | Public Storage |
| 8 | Halcon Resources Corporation |
| 9 | TTM Technologies, Inc. |
| 10 | Atwood Oceanics, Inc. |
| 11 | ACADIA Pharmaceuticals Inc. |
| 12 | Central European Media Enterprises Ltd. |
| 13 | Oxbridge Re Holdings Limited |
| 14 | Western Refining Logistics, LP |
| 15 | Vaalco Energy Inc |
| 16 | Xilinx, Inc. |
| 17 | Liberty Global plc |
| 18 | Honda Motor Company, Ltd. |
| 19 | Great Plains Energy Inc |
| 20 | Assurant, Inc. |

|  |  |  |
| --- | --- | --- |
| **campaigns** | | |
| **company\_id** | **expenses** | **revenue** |
| 1 | 7390.24 | 8652.18 |

|  |  |  |
| --- | --- | --- |
| 2 | 5774.65 | 7955.47 |
| 3 | 2154.71 | 5920.23 |
| 4 | 9366.49 | 3397.85 |
| 5 | 2765.18 | 9158.63 |
| 6 | 7908.41 | 5018.85 |
| 7 | 2251.44 | 6654.52 |
| 8 | 3383.14 | 9354.79 |
| 9 | 8287.96 | 9522.53 |
| 10 | 4356.62 | 4658.52 |
| 11 | 9272.86 | 9161.77 |
| 12 | 4996.18 | 5903.57 |
| 13 | 8354.75 | 2259.26 |
| 14 | 6402.90 | 8146.16 |
| 15 | 1692.05 | 686.71 |
| 16 | 5988.48 | 9089.41 |
| 17 | 6192.33 | 7580.19 |
| 18 | 3016.37 | 7761.25 |
| 19 | 9838.05 | 1293.09 |
| 20 | 4386.52 | 9513.73 |

**Expected Output**

|  |  |
| --- | --- |
| **company\_name** | **profit** |
| Royal Bank Of Canada | 6393.45 |
| Halcon Resources Corporation | 5971.65 |
| Assurant, Inc. | 5127.21 |

Question - 66

**SQL: Traffic Audit Report**

As part of the traffic audit report for one of the ISPs, you need to get a list of customers whose hardware has a higher downstream rate than upstream rate, but no more than one downtime.

The result should have the following columns: *mac | upstream\_rate | downstream\_rate | downtime\_rate*.

The result should be sorted in ascending order by *mac*.

# Note:

Only clients whose downstream rate is higher than the upstream rate should be included in the report. Only clients whose downtime ratio is "never" or "once" should be included in the report.

**Schema**

|  |  |  |
| --- | --- | --- |
| **clients** | | |
| **name** | **type** | **description** |
| mac | VARCHAR(64) | MAC address |
| upstream\_rate | INT | Upstream rate |
| downstream\_rate | INT | Downstream rate |
| downtime\_rate | VARCHAR(64) | Downtime rate |

**Sample Data Tables**

|  |  |  |  |
| --- | --- | --- | --- |
| **clients** | | | |
| **mac** | **upstream\_rate** | **downstream\_rate** | **downtime\_rate** |
| 78-C1-E5-20-D5-61 | 925526 | 5195 | never |
| 78-E2-20-71-9C-30 | 582152 | 375829 | never |
| 0D-09-F7-77-03-E5 | 359529 | 710743 | never |
| 56-18-67-55-58-EA | 78626 | 562544 | once |
| C9-73-EC-1C-4C-B7 | 574927 | 669655 | yearly |
| 02-35-3F-7B-CC-76 | 430072 | 296196 | seldom |
| 3D-95-33-8A-65-F9 | 894176 | 489401 | monthly |
| 8D-33-3F-0E-04-D5 | 897666 | 297063 | weekly |
| 0E-0A-63-B9-79-3E | 69133 | 984354 | seldom |
| 15-D3-2A-DD-02-A4 | 19203 | 995983 | seldom |
| 8D-42-B1-97-AB-87 | 476648 | 177677 | monthly |
| 6B-64-60-47-16-D3 | 700056 | 374321 | monthly |
| FD-61-81-00-BF-EC | 216401 | 498229 | yearly |
| 95-46-C6-C7-6F-E0 | 236331 | 341013 | monthly |

|  |  |  |  |
| --- | --- | --- | --- |
| 11-0E-62-32-62-5E | 694746 | 451525 | daily |
| D6-5B-72-D5-FF-4F | 2931 | 992852 | monthly |
| 66-BF-AD-F2-E5-45 | 861075 | 44216 | monthly |
| E5-C9-C6-74-2E-A8 | 639487 | 968494 | daily |
| 18-56-3A-93-8E-9F | 494945 | 259910 | weekly |
| 51-EB-D7-22-45-99 | 219419 | 326479 | often |

**Expected Output**

|  |  |  |  |
| --- | --- | --- | --- |
| **mac** | **upstream\_rate** | **downstream\_rate** | **downtime\_rate** |
| 0D-09-F7-77-03-E5 | 359529 | 710743 | never |
| 56-18-67-55-58-EA | 78626 | 562544 | once |

Question - 67

**SQL: Calendar Application Events Report**

You are working on a calendar application and need to get a list of the five earliest events for all event owners who are not on vacation. The result should have the following columns: *dt | title | full\_name | email\_address*.

It should be sorted in ascending order by *dt,* and limited to the first five records.

# Note:

Only events whose owner is not on vacation should be included in the report.

**Schema**

|  |  |  |
| --- | --- | --- |
| **owners** | | |
| **name** | **type** | **description** |
| id | SMALLINT | ID |
| full\_name | VARCHAR(255) | Full name |
| email\_address | VARCHAR(255) | Email address |
| on\_vacation | SMALLINT | Vacation status, 1 = on vacation |

|  |  |  |
| --- | --- | --- |
| **events** | | |
| **name** | **type** | **description** |
| owner\_id | SMALLINT | Owner ID |

|  |  |  |
| --- | --- | --- |
| dt | VARCHAR(19) | Timestamp |
| title | VARCHAR(255) | Title |

**Sample Data Tables**

|  |  |  |  |
| --- | --- | --- | --- |
| **owners** | | | |
| **id** | **full\_name** | **email\_address** | **on\_vacation** |
| 2 | Aleksandr Fellows | [afellows1@instagram.com](mailto:afellows1@instagram.com) | 0 |
| 3 | Collete Pack | [cpack2@mit.edu](mailto:cpack2@mit.edu) | 0 |
| 4 | Lorelle Squibb | [lsquibb3@huffingtonpost.com](mailto:lsquibb3@huffingtonpost.com) | 0 |
| 1 | Benjamin Sevier | [bsevier0@discuz.net](mailto:bsevier0@discuz.net) | 1 |
| 5 | Cointon Welberry | [cwelberry4@theguardian.com](mailto:cwelberry4@theguardian.com) | 1 |

|  |  |  |
| --- | --- | --- |
| **events** | | |
| **owner\_id** | **dt** | **title** |
| 1 | 2021-08-13 09:17:41 | ac consequat metus sapien ut |
| 1 | 2021-10-26 16:14:28 | a libero nam |
| 1 | 2022-05-18 16:09:36 | vestibulum ante ipsum primis in faucibus |
| 1 | 2022-06-04 20:37:36 | nisi volutpat eleifend donec |
| 2 | 2021-01-17 23:34:59 | eu mi nulla ac |
| 2 | 2021-01-26 15:59:04 | nisl venenatis lacinia |
| 2 | 2021-04-25 05:04:29 | arcu adipiscing molestie hendrerit |
| 2 | 2022-02-22 10:24:50 | lectus vestibulum quam sapien varius |
| 2 | 2022-05-22 03:04:33 | pellentesque quisque porta volutpat erat |
| 3 | 2021-03-11 08:25:21 | felis sed interdum venenatis |
| 3 | 2022-02-15 07:29:45 | nulla sed accumsan felis ut |
| 4 | 2021-02-28 16:10:55 | cras mi pede malesuada in imperdiet et |
| 4 | 2021-05-20 03:35:11 | odio in hac habitasse platea dictumst |
| 4 | 2021-09-01 19:25:41 | sagittis nam congue risus semper |
| 4 | 2022-02-04 03:11:53 | luctus cum sociis natoque penatibus et magnis |
| 4 | 2022-06-18 08:39:41 | nec molestie sed justo pellentesque |

|  |  |  |
| --- | --- | --- |
| 5 | 2021-04-04 13:39:38 | id massa id nisl venenatis lacinia aenean |
| 5 | 2021-10-18 19:04:14 | adipiscing lorem vitae mattis |
| 5 | 2021-10-19 00:56:11 | vestibulum eget vulputate ut ultrices vel augue |
| 5 | 2022-04-26 18:55:04 | at dolor quis |

**Expected Output**

|  |  |  |  |
| --- | --- | --- | --- |
| **dt** | **title** | **full\_name** | **email\_address** |
| 2021-01-17 23:34:59 | eu mi nulla ac | Aleksandr Fellows | [afellows1@instagram.com](mailto:afellows1@instagram.com) |
| 2021-01-26 15:59:04 | nisl venenatis lacinia | Aleksandr Fellows | [afellows1@instagram.com](mailto:afellows1@instagram.com) |
| 2021-02-28 16:10:55 | cras mi pede malesuada in imperdiet et | Lorelle Squibb | [lsquibb3@huffingtonpost.com](mailto:lsquibb3@huffingtonpost.com) |
| 2021-03-11 08:25:21 | felis sed interdum venenatis | Collete Pack | [cpack2@mit.edu](mailto:cpack2@mit.edu) |
| 2021-04-25 05:04:29 | arcu adipiscing molestie hendrerit | Aleksandr Fellows | [afellows1@instagram.com](mailto:afellows1@instagram.com) |

Question - 68

**Firewall Active Clients Tracking**

As part of a firewall log analysis process, a team needs a list of all the client's MAC addresses that have had traffic.

The result should be in the following format: *mac*.

*mac* is a client MAC address, which was active and has traffic (its *clients.id* exists in the *traffic* table).

**Schema**

There are 2 tables:

|  |  |  |
| --- | --- | --- |
| **clients** | | |
| **name** | **type** | **description** |
| id | SMALLINT | Client ID |
| mac | VARCHAR(64) | MAC address |

|  |  |  |
| --- | --- | --- |
| **traffic** | | |
| **name** | **type** | **description** |
| client\_id | SMALLINT | Client ID |
| amount | INT | Traffic amount |

**Sample Data Tables**

For the sample data in tables:

|  |  |
| --- | --- |
| **clients** | |
| **id** | **mac** |
| 1 | E5-A3-AC-8A-20-F9 |
| 2 | 3B-2F-83-25-A8-81 |
| 3 | 3A-4E-A6-43-1D-B1 |
| 4 | B7-03-14-91-8F-58 |
| 5 | 63-1A-FD-9A-AF-6F |
| 6 | D6-B8-1F-1D-34-04 |
| 7 | 38-83-E5-F8-C8-DC |
| 8 | E9-F6-89-7F-8D-34 |
| 9 | 82-2E-B3-67-04-41 |
| 10 | D1-9D-DE-37-A0-49 |
| 11 | 8A-46-F4-83-29-13 |
| 12 | 4C-1F-7B-C7-08-7E |
| 13 | 72-57-E6-CA-2C-91 |
| 14 | 57-F7-E7-E7-45-36 |
| 15 | 05-8A-05-1D-2D-20 |
| 16 | 46-06-F1-B9-65-7C |
| 17 | 0A-E0-26-9D-2A-27 |
| 18 | F0-86-99-18-36-9B |
| 19 | DD-81-BF-53-BD-9B |
| 20 | 50-53-64-8E-42-BE |

|  |  |
| --- | --- |
| **traffic** | |
| **client\_id** | **amount** |
| 3 | 6385047 |
| 8 | 6490817 |
| 14 | 9109219 |

|  |  |
| --- | --- |
| 16 | 5558512 |
| 17 | 1870152 |
| 17 | 8228920 |
| 18 | 326127 |
| 18 | 5429741 |
| 18 | 4063477 |
| 19 | 7411789 |
| 20 | 5832337 |
| 20 | 1426585 |
| 23 | 1097368 |
| 23 | 6769594 |
| 23 | 802387 |
| 24 | 5959513 |
| 24 | 1300408 |
| 24 | 4631624 |
| 28 | 1629306 |
| 29 | 2814818 |

the expected output is:

|  |
| --- |
| **mac** |
| 0A-E0-26-9D-2A-27 |
| 3A-4E-A6-43-1D-B1 |
| 46-06-F1-B9-65-7C |
| 50-53-64-8E-42-BE |
| 57-F7-E7-E7-45-36 |
| DD-81-BF-53-BD-9B |
| E9-F6-89-7F-8D-34 |
| F0-86-99-18-36-9B |

# Explanation

Each of the addresses listed has at least one record in the traffic table.

As part of an Etherium transactions monitoring process, a team needs a list of all the wallets that have one or more transactions.

Question - 69

**Active Wallets**

The result should be in the following format: *address*

*address* is a wallet address which has one or more transactions (its *wallet.id* exists in the *transactions* table).

**Schema**

There are 2 tables:

|  |  |  |
| --- | --- | --- |
| **wallets** | | |
| **name** | **type** | **description** |
| id | SMALLINT | Wallet ID |
| address | VARCHAR(64) | Wallet address |

|  |  |  |
| --- | --- | --- |
| **transactions** | | |
| **name** | **type** | **description** |
| wallet\_id | SMALLINT | Wallet ID |
| credit | DECIMAL(4,2) | Transaction amount |

**Sample Data Tables**

For the sample data in tables:

|  |  |
| --- | --- |
| **wallets** | |
| **id** | **address** |
| 1 | 0x1ecc4cefde6dfb773352a2dcd8b5f518ccd24ff4 |
| 2 | 0x0f4487168610dcae7f16b6c000a7ba284bb6703c |
| 3 | 0x54c5c516b5c601a3e6fdea18db966186274879e6 |
| 4 | 0x50c404a6790d44849c3500d95f147d2102db5f77 |
| 5 | 0x41cc32037fdcc6cbf5cb04ada4e38032a84361ac |
| 6 | 0xb38ca2076b2c3a0a1c0796ccab5dc3fcbb645336 |
| 7 | 0x923c6ee4debce6c37c512eb90e9b26fc9bbfc3e1 |
| 8 | 0x76cfbf67e1765117a98fd2286bdbcb731b0d29ec |

|  |  |
| --- | --- |
| 9 | 0xf4dff43df89365453440dbeb5e53445d3400d218 |
| 10 | 0x010a4b23f985eda6b397864878cb70bf0b233aa6 |
| 11 | 0x106c4c852394d5a6580c11ca0060687a798ec78c |
| 12 | 0x541da74b9f4b2283524a65156263bcce7429ba61 |
| 13 | 0x86df8e62ad23a35ee2e64b3b4b128d8a3660116a |
| 14 | 0x6b021b12b779aec27009f18bd6ad227685df5474 |
| 15 | 0x7073e9f48b7211d9940db4c8bad8e31d5d9d0577 |
| 16 | 0x4672e132b9627a7db76e5af431bb5febc93b1b2f |
| 17 | 0x5cbc82a01d3b9df8f50ee09bf5a1c48afe0cb966 |
| 18 | 0x1c7d30f90081b45004e0d6549534ef8658795672 |
| 19 | 0xf830618ca9f027ce5f23bf270629a6a418ad355c |
| 20 | 0x1a31ac1b923fcc17a42a340091424ea18192a3e7 |

|  |  |
| --- | --- |
| **transactions** | |
| **wallet\_id** | **credit** |
| 3 | 5.21 |
| 6 | 24.04 |
| 8 | 29.66 |
| 10 | 1.67 |
| 10 | 3.32 |
| 10 | 4.27 |
| 14 | 3.32 |
| 14 | 3.35 |
| 15 | 14.34 |
| 16 | 11.94 |
| 20 | 13.86 |
| 20 | 16.54 |
| 21 | 11.64 |
| 25 | 12.86 |
| 26 | 8.77 |
| 26 | 12.36 |

|  |  |
| --- | --- |
| 27 | 29.65 |
| 28 | 3.00 |
| 28 | 13.26 |
| 28 | 2.85 |

the expected output is:

|  |
| --- |
| **address** |
| 0x010a4b23f985eda6b397864878cb70bf0b233aa6 |
| 0x1a31ac1b923fcc17a42a340091424ea18192a3e7 |
| 0x4672e132b9627a7db76e5af431bb5febc93b1b2f |
| 0x54c5c516b5c601a3e6fdea18db966186274879e6 |
| 0x6b021b12b779aec27009f18bd6ad227685df5474 |
| 0x7073e9f48b7211d9940db4c8bad8e31d5d9d0577 |
| 0x76cfbf67e1765117a98fd2286bdbcb731b0d29ec |
| 0xb38ca2076b2c3a0a1c0796ccab5dc3fcbb645336 |

# Explanation

Each of the wallets listed has at least one record in the transactions table.

Question - 70

**Animal Tracking**

As part of an animal tracking program, a team needs a list of all the animals, tracked at least once.

The result should be in the following format: *name*.

*name* is the name of an animal, which has at least one record of being tracked (its *animal.id* exists in the *tracklog* table).

**Schema**

There are 2 tables:

|  |  |  |
| --- | --- | --- |
| **animals** | | |
| **name** | **type** | **description** |
| id | SMALLINT | Animal ID |
| name | VARCHAR(64) | Animal name |

|  |  |  |
| --- | --- | --- |
| **tracklog** | | |
| **name** | **type** | **description** |
| animal\_id | SMALLINT | Animal ID |
| tracked\_at | VARCHAR(19) | Track timestamp |

**Sample Data Tables**

For the sample data in tables:

|  |  |
| --- | --- |
| **animals** | |
| **id** | **name** |
| 1 | Blue and yellow macaw |
| 2 | Jungle kangaroo |
| 3 | Stork, woolly-necked |
| 4 | North American river otter |
| 5 | Square-lipped rhinoceros |
| 6 | Black-fronted bulbul |
| 7 | American beaver |
| 8 | Capybara |
| 9 | Black-backed jackal |
| 10 | Dragon, ornate rock |
| 11 | Wombat, southern hairy-nosed |
| 12 | Snake, carpet |
| 13 | Egyptian cobra |
| 14 | Green heron |
| 15 | Indian star tortoise |
| 16 | Roan antelope |
| 17 | Rhea, common |
| 18 | Fairy penguin |
| 19 | Black-eyed bulbul |
| 20 | Starling, cape |

|  |  |
| --- | --- |
| **tracklog** | |
| **animal\_id** | **tracked\_at** |
| 2 | 2021-07-08 12:30:34 |
| 5 | 2021-09-15 03:00:04 |
| 8 | 2021-12-14 11:20:50 |
| 9 | 2021-05-15 14:54:15 |
| 11 | 2021-07-03 17:04:14 |
| 11 | 2021-02-23 09:02:11 |
| 15 | 2021-11-02 22:49:37 |
| 15 | 2021-07-31 04:51:25 |
| 15 | 2021-06-06 14:57:23 |
| 19 | 2021-03-25 02:00:31 |
| 19 | 2021-04-21 22:02:51 |
| 19 | 2021-07-17 17:12:35 |
| 21 | 2021-05-06 03:35:59 |
| 22 | 2021-09-21 22:49:36 |
| 22 | 2021-12-04 00:26:29 |
| 24 | 2021-12-27 08:19:52 |
| 26 | 2021-05-14 07:30:11 |
| 28 | 2021-06-08 00:47:00 |
| 28 | 2021-04-24 00:42:31 |
| 30 | 2021-05-06 21:03:03 |

the expected output is:

|  |
| --- |
| **name** |
| Black-backed jackal |
| Black-eyed bulbul |
| Capybara |
| Indian star tortoise |
| Jungle kangaroo |
| Square-lipped rhinoceros |

|  |
| --- |
| Wombat, southern hairy-nosed |

# Explanation

Each of the animals listed has at least one record in the tracklog table.

Question - 71

**SQL: Exchange Rates**

In exchange for a transaction fee, a stock exchange lets people trade stocks online. For every buy order they charge 0.1% of the order amount, and for every sell order they charge 0.15%. Calculate the fees paid by customers for their orders.

Fetch the name of the customer and their total fees paid. Order the result by customer names and round the fees to 2 decimal places.

To round the data:

In MySQL, Oracle, or DB2, use ROUND(val, 2). In MS SQL use FORMAT(val, 'N2').

**Schema**

You are provided 2 tables: customers , orders .

|  |  |  |
| --- | --- | --- |
| **CUSTOMERS** | | |
| Name | Type | Description |
| id | int | Unique id of the customer. |
| customer\_name | varchar(30) | Name of the customer. |

|  |  |  |
| --- | --- | --- |
| **ORDERS** | | |
| Name | Type | Description |
| order\_id | int | Unique id of the order. |
| customer\_id | int | Id of the customer referring to the customers table. |
| order\_type | varchar(5) | Type of order placed (Buy or Sell). |
| order\_amount | decimal(18,2) | Amount of the order. |

**Sample Data Tables**

|  |  |
| --- | --- |
| **CUSTOMERS** | |
| id | customer\_name |
| 401 | Hubert Keesler |
| 402 | Devin Vert |
| 403 | Lashawna Bowerman |

|  |  |
| --- | --- |
| 404 | Brigid Wellborn |
| 405 | Josefine Perl |

|  |  |  |  |
| --- | --- | --- | --- |
| **ORDERS** | | | |
| order\_id | customer\_id | order\_type | order\_amount |
| 4361 | 401 | Sell | 912.77 |
| 3478 | 405 | Sell | 741.69 |
| 7292 | 405 | Sell | 436.05 |
| 5833 | 405 | Sell | 231.30 |
| 3472 | 402 | Buy | 950.92 |
| 4472 | 401 | Sell | 367.70 |
| 2624 | 404 | Buy | 218.15 |
| 7198 | 405 | Buy | 797.29 |
| 7660 | 403 | Buy | 131.18 |
| 5192 | 401 | Buy | 362.44 |
| 5260 | 402 | Buy | 636.26 |
| 2726 | 403 | Sell | 138.15 |
| 6594 | 401 | Buy | 234.51 |
| 4657 | 404 | Buy | 427.30 |
| 9744 | 402 | Sell | 623.36 |

|  |  |
| --- | --- |
| **OUTPUT** | |
| customer\_name | total\_fees |
| Brigid Wellborn | 0.65 |
| Devin Vert | 2.52 |
| Hubert Keesler | 2.52 |
| Josefine Perl | 2.91 |
| Lashawna Bowerman | 0.34 |

# Explanation

For Devin Vert, total amount of buy orders was 1587.18 and total amount of buy orders was 623.36. Fee applied on buy and sell orders will be 1.58718 and 0.93504 with rate of 0.1% and 0.15% respectively. Total fee would be 2.52222 after rounding final fee would be 2.52.

Question - 72

**SQL: Credit Dues**

A credit card issuer sets its customers' interest rates based on their credit scores. Given details of customer transactions, calculate the amount due for every customer based on their individual interest rates.

Write a query to fetch the full name of the customer along with the amount due (i.e. total transaction amount + interest on that amount) for the customers whose interest rate is more than 12%. Round the amounts to 2 places after the decimal and order them from high to low.

**Schema**

There are 2 tables: credit\_holders , transactions .

|  |  |  |
| --- | --- | --- |
| **CREDIT\_HOLDERS** | | |
| Name | Type | Description |
| id | int | Unique id of the credit holder (customer). |
| first\_name | varchar(15) | First name of the customer. |
| last\_name | varchar(15) | Last name of the customer. |
| interest\_rate | int | Interest based on their credit score (in percentage). |

|  |  |  |
| --- | --- | --- |
| **TRANSACTIONS** | | |
| Name | Type | Description |
| transaction\_id | int | Unique id of the transaction. |
| credit\_holder\_id | int | Id of the customer referring to the credit\_holders table. |
| amount | decimal(18,2) | Amount of the transaction. |

**Sample Data Tables**

|  |  |  |  |
| --- | --- | --- | --- |
| **CREDIT\_HOLDERS** | | | |
| id | first\_name | last\_name | interest\_rate |
| 101 | Clemencia | Hutsell | 12 |
| 102 | Susannah | Ismail | 18 |
| 103 | Sixta | Hagy | 18 |
| 104 | Otto | Izquierdo | 18 |
| 105 | Anita | Degroot | 15 |

|  |  |  |
| --- | --- | --- |
| **TRANSACTIONS** | | |
| transaction\_id | credit\_holder\_id | amount |
| 4361 | 101 | 65.22 |
| 3478 | 104 | 51.85 |
| 7292 | 104 | 64.60 |

|  |  |  |
| --- | --- | --- |
| 5833 | 105 | 72.15 |
| 3472 | 102 | 96.28 |
| 4472 | 101 | 80.06 |
| 2624 | 101 | 85.27 |
| 7198 | 104 | 23.73 |
| 7660 | 103 | 81.86 |
| 5192 | 101 | 69.64 |
| 5260 | 101 | 71.72 |
| 2726 | 102 | 57.66 |
| 6594 | 103 | 23.23 |
| 4657 | 101 | 81.68 |
| 9744 | 104 | 99.57 |
| 2054 | 103 | 51.13 |
| 7156 | 105 | 12.78 |
| 3273 | 105 | 36.15 |
| 9756 | 101 | 45.41 |
| 9702 | 105 | 69.75 |

|  |  |
| --- | --- |
| **OUTPUT** | |
| full\_name | dues |
| Otto Izquierdo | 282.91 |
| Anita Degroot | 219.45 |
| Sixta Hagy | 184.34 |
| Susannah Ismail | 181.65 |

# Explanation

For Otto Izquierdo, there are 4 transactions of 51.85, 64.60, 23.73, 99.57 which sums to 239.75. After applying 18% interest, the due amount is 282.905

rounded to 282.91.

Question - 73

**SQL: Interest Earned**

A bank offers 5% interest on deposits. Given account holder names and amounts deposited, calculate the interest earned in 1 year. Report the depositor names and the interest they earn ordered ascending by name.

# Notes:

The *amount* field is a string where the first character is a currency symbol, and the remainder is the deposit amount.

The interest earned should be preceded by the same currency symbol, rounded to 2 decimals, e.g., $10.004 should show as $10.00.

**Schema**

There is 1 table: accounts.

|  |  |  |
| --- | --- | --- |
| **ACCOUNTS** | | |
| Name | Type | Description |
| id | int | Unique id of the account. |
| account\_holder | varchar(30) | Name of the account holder. |
| amount | varchar(10) | Amount deposited in the fixed deposit. |

**Sample Data Tables**

|  |  |  |
| --- | --- | --- |
| **ACCOUNTS** | | |
| id | account\_holder | amount |
| 1 | Ellis Beane | $5582.03 |
| 2 | Drew Nolf | $2470.3 |
| 3 | Jordan Chatmon | $6211.52 |
| 4 | Robin Hansard | $8133.31 |
| 5 | Spencer Days | $5273.81 |
| 6 | Morgan Criss | $3741.85 |
| 7 | Wesley Waugh | $7056.14 |
| 8 | Alex Canty | $2590.45 |
| 9 | Blake Hawbaker | $3987.27 |
| 10 | Taylor Blackston | $8351.98 |

|  |  |
| --- | --- |
| **OUTPUT** | |
| account\_holder | interest |
| Taylor Blackston | $417.60 |
| Wesley Waugh | $352.81 |
| Jordan Chatmon | $310.58 |
| Robin Hansard | $406.67 |
| Alex Canty | $129.52 |
| Drew Nolf | $123.52 |
| Blake Hawbaker | $199.36 |
| Spencer Days | $263.69 |

|  |  |
| --- | --- |
| Ellis Beane | $279.10 |
| Morgan Criss | $187.09 |

# Explanation

For Taylor Blackston, after 1 year the interest is $8351.98 \* 0.05 = $147.599 which rounds to $417.60.

Question - 74

**SQL: Monthly Revenue**

A company has been selling beauty products for the last three years. They need a report of transactions by month over the period. The transaction id contains the year and month of the transaction, e.g. 19JAN1092 was executed in January of 2019.

Write a query to fetch year, month, and total transactions in each month. Order the results ascending by year and month.

**Schema**

There is 1 table: transactions.

|  |  |  |
| --- | --- | --- |
| **TRANSACTIONS** | | |
| Name | Type | Description |
| transaction\_id | varchar(10) | Unique id of the transaction YYMMMDD[the rest] |
| amount | decimal(18,2) | Amount of the transaction. |

**Sample Data Tables**

|  |  |
| --- | --- |
| **TRANSACTIONS** | |
| transaction\_id | amount |
| 19SEP2187 | 785.72 |
| 19OCT4361 | 752.64 |
| 19APR3478 | 197.92 |
| 19SEP7292 | 910.26 |
| 21MAR5833 | 344.70 |
| 20MAY3472 | 939.61 |
| 20DEC4472 | 154.98 |
| 20DEC2624 | 935.44 |
| 21JUN7198 | 309.81 |
| 19APR7660 | 528.10 |
| 20MAR5192 | 995.22 |
| 19OCT5260 | 861.11 |

|  |  |
| --- | --- |
| 21JUN2726 | 611.94 |
| 19OCT6594 | 478.54 |
| 19APR4657 | 183.20 |

|  |  |  |
| --- | --- | --- |
| **OUTPUT** | | |
| year | month | total\_transactions |
| 19 | APR | 909.22 |
| 19 | OCT | 2092.29 |
| 19 | SEP | 1695.98 |
| 20 | DEC | 1090.42 |
| 20 | MAR | 995.22 |
| 20 | MAY | 939.61 |
| 21 | JUN | 921.75 |
| 21 | MAR | 344.70 |

# Explanation

For the month of April, 19 there are three transactions of 197.92, 528.10, and 183.20 which sum up to 909.22.

Question - 75

**SQL: Final Result**

A college offers a 4-year B.Tech degree. Each year students receive grades out of 10 (CGPA).

Write a query to fetch the full name of each student with the average GPA over the 4 years. Round the average to one decimal place. Order the result from highest to lowest average GPA.

**Schema**

There is 1 table: results .

|  |  |  |
| --- | --- | --- |
| **RESULTS** | | |
| Name | Type | Description |
| id | int | Unique id of the student. |
| first\_name | varchar(20) | First name of the student. |
| last\_name | varchar(20) | Last name of the student. |
| cgpa\_first\_year | float | CGPA year 1 |
| cgpa\_second\_year | float | CGPA year 2 |
| cgpa\_third\_year | float | CGPA year 3 |

|  |  |  |
| --- | --- | --- |
| cgpa\_fourth\_year | float | CGPA year 4 |

**Sample Data Tables**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **RESULTS** | | | | | | |
| id | first\_name | last\_name | cgpa\_first\_year | cgpa\_second\_year | cgpa\_third\_year | cgpa\_fourth\_year |
| 1 | Pearlene | Beane | 7 | 5.1 | 8.4 | 8.9 |
| 2 | Franklin | Nolf | 7.7 | 7.2 | 5.2 | 8.3 |
| 3 | Bell | Chatmon | 7.3 | 8.4 | 8.9 | 10 |
| 4 | Belva | Hansard | 6.2 | 9.2 | 5.8 | 6.7 |
| 5 | Missy | Days | 8.3 | 10 | 7.3 | 6.7 |
| 6 | Vicenta | Criss | 5.4 | 9.5 | 6.1 | 9 |
| 7 | Annelle | Waugh | 6.5 | 7.9 | 9.6 | 9.3 |
| 8 | Darby | Canty | 5.5 | 9 | 8.6 | 5.9 |
| 9 | Ka | Hawbaker | 5.7 | 6.4 | 5.2 | 6.8 |
| 10 | Alease | Blackston | 5.3 | 7.5 | 9.3 | 6 |

|  |  |
| --- | --- |
| **OUTPUT** | |
| full\_name | average\_gpa |
| Bell Chatmon | 8.6 |
| Annelle Waugh | 8.3 |
| Missy Days | 8.1 |
| Vicenta Criss | 7.5 |
| Pearlene Beane | 7.3 |
| Darby Canty | 7.3 |
| Franklin Nolf | 7.1 |
| Belva Hansard | 7.0 |
| Alease Blackston | 7.0 |
| Ka Hawbaker | 6.0 |

# Explanation

For Bell Chatmon, the cgpa sum is 7.3 + 8.4 + 8.9 + 10 = 34.6. The average is 34.6/4 = 8.65. After rounding to 1 decimal place the final value is 8.6. All values should show 1 place after the decimal, even in a case like Belva Hansard where the final value should be 7.0.

Question - 76

**SQL: Mutual Funds**

A bank offers several mutual funds to investors. They receive investments each month and would like a summary.

Write a query to fetch the month, name of the fund, and the total amount invested that month. Order the report by month and mutual fund name.

**Schema**

There is 1 table: funds .

|  |  |  |
| --- | --- | --- |
| **FUNDS** | | |
| Name | Type | Description |
| id | int | Unique id of the order |
| order\_date | date | transaction date |
| fund\_name | varchar(50) | Name of the fund |
| order\_amount | int | Amount invested |

**Sample Data Tables**

|  |  |  |  |
| --- | --- | --- | --- |
| **FUNDS** | | | |
| id | order\_date | fund\_name | order\_amount |
| 1 | 2021-09-05 | Mid-Cap | 151 |
| 2 | 2020-11-27 | Small-Cap | 784 |
| 3 | 2020-11-22 | Multi-Cap | 761 |
| 4 | 2020-02-26 | Large-Cap | 778 |
| 5 | 2020-01-04 | Mid-Cap | 949 |
| 6 | 2020-02-01 | Large-Cap | 392 |
| 7 | 2020-02-07 | Mid-Cap | 629 |
| 8 | 2020-06-01 | Mid-Cap | 529 |
| 9 | 2020-08-05 | Large-Cap | 258 |
| 10 | 2021-09-23 | Mid-Cap | 739 |
| 11 | 2020-02-14 | Large-Cap | 563 |
| 12 | 2021-09-29 | Small-Cap | 817 |
| 13 | 2020-05-11 | Large-Cap | 121 |
| 14 | 2021-09-18 | Mid-Cap | 341 |
| 15 | 2021-09-07 | Large-Cap | 260 |
| 16 | 2021-10-11 | Small-Cap | 102 |
| 17 | 2020-06-04 | Mid-Cap | 496 |
| 18 | 2021-07-26 | Multi-Cap | 321 |

|  |  |  |  |
| --- | --- | --- | --- |
| 19 | 2020-04-11 | Small-Cap | 323 |
| 20 | 2020-09-24 | Multi-Cap | 938 |
| 21 | 2020-06-16 | Small-Cap | 668 |
| 22 | 2021-03-08 | Small-Cap | 733 |
| 23 | 2021-04-27 | Small-Cap | 597 |
| 24 | 2020-01-11 | Small-Cap | 767 |
| 25 | 2021-11-14 | Small-Cap | 489 |

|  |  |  |
| --- | --- | --- |
| **OUTPUT** | | |
| month | fund\_name | total\_investments |
| 1 | Mid-Cap | 949 |
| 1 | Small-Cap | 767 |
| 2 | Large-Cap | 1733 |
| 2 | Mid-Cap | 629 |
| 3 | Small-Cap | 733 |
| 4 | Small-Cap | 920 |
| 5 | Large-Cap | 121 |
| 6 | Mid-Cap | 1025 |
| 6 | Small-Cap | 668 |
| 7 | Multi-Cap | 321 |
| 8 | Large-Cap | 258 |
| 9 | Large-Cap | 260 |
| 9 | Mid-Cap | 1231 |
| 9 | Multi-Cap | 938 |
| 9 | Small-Cap | 817 |
| 10 | Small-Cap | 102 |
| 11 | Multi-Cap | 761 |
| 11 | Small-Cap | 1273 |

# Explanation

For the 1st month the total investment for the Mid-Cap fund is 949 and for Small-Cap is 767. Since there is no Large-Cap or Multi-Cap fund investments that month, they are not shown.

Question - 77

**DPI Software Protocols Report**

As part of HackerSniff's DPI (Deep Packet Inspection) software analytics, a team needs a list of all the protocols for which incoming traffic is higher than outgoing.

The result should be in the following format: *protocol, traffic\_in, traffic\_out*.

Results should be sorted ascending by *protocol*.

**Schema**

There is 1 table:

|  |  |  |
| --- | --- | --- |
| **traffic** | | |
| **name** | **type** | **description** |
| client | VARCHAR(17) | Client MAC address |
| protocol | VARCHAR(64) | Protocol name |
| traffic\_in | INT | Traffic in |
| traffic\_out | INT | Traffic out |

**Sample Data Tables**

For the sample data in table:

|  |  |  |  |
| --- | --- | --- | --- |
| **traffic** | | | |
| **client** | **protocol** | **traffic\_in** | **traffic\_out** |
| 02-E1-80-76-EC-4B | BGP | 0 | 234737 |
| 43-15-AA-26-0F-A4 | BGP | 402860 | 606565 |
| 90-E7-B0-14-7E-8C | BGP | 840772 | 988197 |
| FB-60-23-C1-5E-D6 | DNS | 341155 | 356569 |
| 4D-6D-7F-62-F4-00 | FTP | 8346 | 413322 |
| 09-89-26-46-C4-21 | FTP | 210656 | 470568 |
| B1-6A-35-2F-1A-C2 | FTP | 897097 | 161083 |
| 0C-CA-68-2D-4B-F5 | HTTP | 918793 | 550403 |
| A4-C6-52-10-2E-9C | HTTPS | 520856 | 185387 |
| 95-B8-7D-78-06-42 | POP | 150880 | 423073 |
| B9-C1-1B-32-55-95 | POP | 862946 | 979544 |
| 14-FD-21-F6-5E-67 | SMTP | 139389 | 280646 |
| 70-E1-2D-B1-B2-9B | SMTP | 163986 | 450401 |

|  |  |  |  |
| --- | --- | --- | --- |
| C6-F1-59-FF-5D-BE | SMTP | 271295 | 878246 |
| 62-01-CF-AD-32-A7 | SMTP | 388933 | 81625 |
| 41-80-FB-86-D1-93 | SMTP | 752842 | 253981 |
| 93-3F-01-57-5F-4A | SSH | 496717 | 599280 |
| 52-F2-BF-45-84-74 | SSH | 632534 | 128765 |
| 87-66-B5-A5-2F-7B | SSH | 835441 | 354950 |
| CE-FC-80-F3-95-58 | UDP | 903443 | 120298 |

the expected output is:

|  |  |  |
| --- | --- | --- |
| **protocol ▲** | **traffic\_in** | **traffic\_out** |
| FTP | 1116099 | 1044973 |
| HTTP | 918793 | 550403 |
| HTTPS | 520856 | 185387 |
| SSH | 1964692 | 1082995 |
| UDP | 903443 | 120298 |

Question - 78

**Advertising System Failures Report**

As part of HackerAd's advertising system analytics, a team needs a list of customers who have a maximum number of failure events (*status* = "failure") in their campaigns.

For all customers with more than *3* events with *status* = 'failure', report the customer name and their number of failures.

The result should be in the following format: *customer, failures*.

*customer* is a candidate's full name, the *first\_name* and *last\_name* separated by a single space. The order of the output is not important.

**Schema**

There are 3 tables:

|  |  |  |
| --- | --- | --- |
| **customers** | | |
| **name** | **type** | **description** |
| id | SMALLINT | Customer ID |
| first\_name | VARCHAR(64) | Customer first name |
| last\_name | VARCHAR(64) | Customer last name |

|  |  |  |
| --- | --- | --- |
| **campaigns** | | |
| **name** | **type** | **description** |
| id | SMALLINT | Campaign ID |
| customer\_id | SMALLINT | Customer ID |
| name | VARCHAR(64) | Campaign name |

|  |  |  |
| --- | --- | --- |
| **events** | | |
| **name** | **type** | **description** |
| dt | VARCHAR(19) | Event timestamp |
| campaign\_id | SMALLINT | Campaign ID |
| status | VARCHAR(64) | Event status |

**Sample Data Tables**

For the sample data in tables:

|  |  |  |
| --- | --- | --- |
| **customers** | | |
| **id** | **first\_name** | **last\_name** |
| 1 | Whitney | Ferrero |
| 2 | Dickie | Romera |

|  |  |  |
| --- | --- | --- |
| **campaigns** | | |
| **id** | **customer\_id** | **name** |
| 1 | 1 | Upton Group |
| 2 | 1 | Roob, Hudson and Rippin |
| 3 | 1 | McCullough, Rempel and Larson |
| 4 | 1 | Lang and Sons |
| 5 | 2 | Ruecker, Hand and Haley |

|  |  |  |
| --- | --- | --- |
| **events** | | |
| **dt** | **campaign\_id** | **status** |
| 2021-12-02 13:52:00 | 1 | failure |

|  |  |  |
| --- | --- | --- |
| 2021-12-02 08:17:48 | 2 | failure |
| 2021-12-02 08:18:17 | 2 | failure |
| 2021-12-01 11:55:32 | 3 | failure |
| 2021-12-01 06:53:16 | 4 | failure |
| 2021-12-02 04:51:09 | 4 | failure |
| 2021-12-01 06:34:04 | 5 | failure |
| 2021-12-02 03:21:18 | 5 | failure |
| 2021-12-01 03:18:24 | 5 | failure |
| 2021-12-02 15:32:37 | 1 | success |
| 2021-12-01 04:23:20 | 1 | success |
| 2021-12-02 06:53:24 | 1 | success |
| 2021-12-02 08:01:02 | 2 | success |
| 2021-12-01 15:57:19 | 2 | success |
| 2021-12-02 16:14:34 | 3 | success |
| 2021-12-02 21:56:38 | 3 | success |
| 2021-12-01 05:54:43 | 4 | success |
| 2021-12-02 17:56:45 | 4 | success |
| 2021-12-02 11:56:50 | 4 | success |
| 2021-12-02 06:08:20 | 5 | success |

the expected output is:

|  |  |
| --- | --- |
| **customer** | **failures ▼** |
| Whitney Ferrero | 6 |

Question - 79

**Election Exit Poll Report**

As part of HackerPoll's election exit poll analytics, a team needs a list of candidates with votes they received.

The result should be in the following format: *candidate, votes*.

*candidate* is a candidate's full name, the *first\_name* and *last\_name* separated by a single space. *votes* is a total count of all *vote\_at* of the candidate.

Votes that cannot be matched to a candidate should be ignored.

Results should be sorted descending by *votes,* then ascending by *candidate*.

**Schema**

There are 2 tables:

|  |  |  |
| --- | --- | --- |
| **candidates** | | |
| **name** | **type** | **description** |
| id | SMALLINT | Candidate ID |
| first\_name | VARCHAR(64) | Candidate first name |
| last\_name | VARCHAR(64) | Candidate last name |

|  |  |  |
| --- | --- | --- |
| **results** | | |
| **name** | **type** | **description** |
| candidate\_id | SMALLINT | Candidate ID |
| vote\_at | VARCHAR(19) | Vote timestamp |

**Sample Data Tables**

For the sample data in tables:

|  |  |  |
| --- | --- | --- |
| **candidates** | | |
| **id** | **first\_name** | **last\_name** |
| 1 | Xavier | Ping |
| 2 | Westley | Drewell |
| 3 | Dominick | Scoble |

|  |  |
| --- | --- |
| **results** | |
| **candidate\_id** | **vote\_at** |
| 0 | 2021-12-01 14:15:52 |
| 1 | 2021-12-01 03:55:23 |
| 1 | 2021-12-01 21:53:26 |
| 1 | 2021-12-02 07:57:40 |
| 1 | 2021-12-02 13:56:06 |
| 2 | 2021-12-01 11:46:40 |
| 2 | 2021-12-01 14:56:05 |

|  |  |
| --- | --- |
| 2 | 2021-12-01 21:54:50 |
| 2 | 2021-12-02 00:43:18 |
| 2 | 2021-12-02 06:59:33 |
| 2 | 2021-12-02 08:36:35 |
| 2 | 2021-12-02 10:20:33 |
| 2 | 2021-12-02 14:02:38 |
| 3 | 2021-12-01 05:18:34 |
| 3 | 2021-12-02 03:55:37 |
| 3 | 2021-12-02 05:30:24 |
| 3 | 2021-12-02 08:32:06 |
| 4 | 2021-12-02 05:05:55 |
| 5 | 2021-12-02 15:50:50 |
| 5 | 2021-12-02 20:45:08 |

the expected output is:

|  |  |
| --- | --- |
| **candidate** | **votes ▼** |
| Westley Drewell | 8 |
| Xavier Ping | 4 |
| Dominick Scoble | 4 |

Question - 80

**Billing Analytics Customer Report**

As part of HackerPay's billing analytics, a team needs a list of MVCs (Most Valued Customers) and their total transactions in December. An MVC is a customer with 3 or more transactions in a month.

List all customers who are MVC in December. For each MVC, include the customer's name (*customer)*, their transactions count, and the sum of their transactions *amount* for the month.

The result should be in the following format: *customer, transactions, total*. Sort the result ascending by *customer*.

**Schema**

There is 1 table:

|  |  |  |
| --- | --- | --- |
| **events** | | |
| **name** | **type** | **description** |
| dt | VARCHAR(19) | Transaction timestamp |
| customer | VARCHAR(64) | Customer name |
| amount | DECIMAL(5,2) | Transaction amount |

**Sample Data Tables**

For the sample data in table:

|  |  |  |
| --- | --- | --- |
| **events** | | |
| **dt** | **customer** | **amount** |
| 2021-11-22 06:41:01 | Donaugh Furneaux | 0.89 |
| 2021-12-22 20:07:04 | Donaugh Furneaux | 10.51 |
| 2021-12-31 05:22:11 | Donaugh Furneaux | 55.92 |
| 2021-12-12 21:26:42 | Harley Lyddiard | 37.68 |
| 2021-11-22 21:24:30 | Kippy Jelly | 85.87 |
| 2021-11-25 07:00:29 | Kippy Jelly | 7.25 |
| 2021-12-16 16:48:32 | Kippy Jelly | 65.49 |
| 2021-11-22 23:30:55 | Latrina Jackman | 93.49 |
| 2021-11-24 19:38:52 | Latrina Jackman | 82.28 |
| 2021-11-30 22:59:33 | Latrina Jackman | 96.87 |
| 2021-12-30 13:05:34 | Latrina Jackman | 88.19 |
| 2021-11-22 02:08:02 | Maribel Braim | 20.19 |
| 2021-12-13 00:14:58 | Maribel Braim | 97.99 |
| 2021-12-26 13:22:20 | Maribel Braim | 57.06 |
| 2021-12-29 00:20:27 | Maribel Braim | 24.35 |
| 2021-11-25 14:29:29 | Orrin Curley | 6.69 |
| 2021-12-08 06:22:16 | Orrin Curley | 36.85 |
| 2021-12-09 15:32:16 | Orrin Curley | 11.04 |
| 2021-11-28 00:15:20 | Rasla Venny | 14.59 |
| 2021-12-25 09:58:23 | Rasla Venny | 6.41 |

the expected output is:

|  |  |  |
| --- | --- | --- |
| **customer ▲** | **transactions** | **total** |
| Maribel Braim | 3 | 179.40 |

Question - 81

**Aggregate Marks**

There is a database containing the marks of some students in various subjects. The data may contain any number of subjects for a student.

Retrieve the records of students who have a sum of marks greater than or equal to 500. The result should be in the following format: *STUDENT\_ID SUM\_OF\_MARKS* sorted descending by *STUDENT\_ID*.

**Schema**

There is 1 table: marks .

|  |  |  |
| --- | --- | --- |
| **marks** | | |
| Name | Type | Description |
| STUDENT\_ID | INTEGER | This is the student's unique ID. |
| MARKS | INTEGER | These are the marks obtained. |

**Sample Data Table**

|  |  |
| --- | --- |
| **marks** | |
| STUDENT\_ID | MARKS |
| 1 | 450 |
| 2 | 200 |
| 3 | 260 |
| 2 | 300 |
| 3 | 250 |

# Sample Output

**Explanation**

3 510

2 500

*3* has a sum of **510**, so it is printed in the query results.

*1* has a sum of **450**, so it does not get printed in the query results.

*2* has a sum of **500**, so it is printed in the query results.

A travel and tour company has 2 tables that relate to customers: *FAMILIES* and *COUNTRIES*. Each tour offers a discount if a minimum number of people book at the same time.

Question - 82

**Trip Query**

Write a query to print the maximum number of discounted tours any 1 family in the FAMILIES table can choose from.

**Schema**

There are 2 tables: FAMILIES , COUNTRIES.

|  |  |  |
| --- | --- | --- |
| **FAMILIES** | | |
| Name | Type | Description |
| ID | STRING | Unique ID of the family. |
| NAME | STRING | Name of the primary contact. |
| FAMILY\_SIZE | INTEGER | Size of the family. |

|  |  |  |
| --- | --- | --- |
| **COUNTRIES** | | |
| Name | Type | Description |
| ID | STRING | Unique ID of the country. |
| NAME | STRING | Name of the country. |
| MIN\_SIZE | INTEGER | Minimum size group to get a discount. |

**Sample Data Tables**

|  |  |  |
| --- | --- | --- |
| **FAMILIES** | | |
| ID | NAME | FAMILY\_SIZE |
| c00dac11bde74750b4d207b9c182a85f | Alex Thomas | 9 |
| eb6f2d3426694667ae3e79d6274114a4 | Chris Gray | 2 |

|  |  |  |
| --- | --- | --- |
| **COUNTRIES** | | |
| ID | NAME | MIN\_SIZE |
| 023fd23615bd4ff4b2ae0a13ed7efec9 | Bolivia | 2 |
| be247f73de0f4b2d810367cb26941fb9 | Cook Islands | 4 |
| 3e85ab80a6f84ef3b9068b21dbcc54b3 | Brazil | 4 |

# Sample Output

3

**Explanation**

The Thomas family can choose from any of the 3 tours and qualify for the discount. The Gray family only qualifies for 1.

Question - 83

**Activity Query**

A parent keeps track of the activities of a child and their friends in two tables: *FRIENDS* and *ACTIVITIES*. Write a query to print the names of all the activities with neither the maximum nor minimum number of participants.

**Schema**

There are 2 tables: *FRIENDS*, *ACTIVITIES*.

|  |  |  |
| --- | --- | --- |
| **FRIENDS** | | |
| Name | Type | Description |
| ID | INTEGER | The ID of a friend. This is the primary key. |
| NAME | STRING | The name of the friend. |
| ACTIVITY | STRING | Name of the activity which the friend takes part in. |

|  |  |  |
| --- | --- | --- |
| **ACTIVITIES** | | |
| Name | Type | Description |
| ID | INTEGER | ID of the activity. |
| NAME | STRING | Name of the activity. |

**Sample Data Tables**

**Sample Input**

|  |  |  |
| --- | --- | --- |
| **FRIENDS** | | |
| ID | NAME | ACTIVITY |
| 1 | James Smith | Horse Riding |
| 2 | Eric Jenkins | Eating |
| 3 | Sean Cox | Eating |
| 4 | Eric Schmidt | Horse Riding |
| 5 | Chris Evans | Eating |
| 6 | Jessica Breeds | Playing |

|  |  |
| --- | --- |
| **Activities** | |
| ID | NAME |
| 1 | Horse Riding |
| 2 | Eating |
| 3 | Playing |

Sample Output

Horse Riding

Question - 84

**Restaurant's Growth**

A restaurant is visited by various customers during a day. At the same time, the restaurant is advertising to increase customer revenue. Write an SQL query to compute the moving average of how much customers spent over a window of 7 days (current day and 6 days before) to analyze their business growth. The output should contain three columns *{visited\_on, amount, avg\_amount(avg over 7 days)}*

**Schema**

There is 1 table: customers

|  |  |  |
| --- | --- | --- |
| **CUSTOMERS** | | |
| Name | Type | Description |
| id | INTEGER | The customer's id, Primary Key |
| name | VARCHAR | Name of the customer |
| phone | VARCHAR | Phone of the customer |
| visited\_on | DATE | Date that the customer visited |
| amount | INTEGER | Amount he customer spent |

**Sample Data Tables**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CUSTOMERS** | | | | |
| id | name | phone | visited\_on | amount |
| 1 | Julia | 1234567890 | 2015-05-01 | 100 |
| 2 | Samantha | 1234567890 | 2015-05-02 | 200 |
| 3 | Julia-Samantha | 1234567890 | 2015-05-03 | 300 |

**Sample Output** 2015-05-01 100 100

2015-05-02 200 150

2015-05-03 300 200

# Explanation

Row 1 : On May 1*,* Julia visited and spent *100. avg\_amount* = *100/1*

Row 2 : On May 2, Samantha visited and spent *200. avg\_amount* = *(100+200)/2 = 150* Row 3 : On May 3, they both visited and spent 300. *avg\_amount* = *(100+200+300)/3 = 200*

Question - 85

**Examination Data Management**

A college maintains the data of its students and their respective appeared examinations in two tables: STUDENT and EXAMINATION. A student might appear for an exam in a particular subject more than once. Write a query to print the student ID, the subject name, and the total number of times the student appeared in a particular subject's examination. The order of output does not matter. The result should be in the following format: STUDENT.ID EXAMINATION.SUBJECT NUMBER\_OF\_TIMES

**Schema**

|  |  |  |
| --- | --- | --- |
| **STUDENT** | | |
| Name | Type | Description |
| ID | Integer | The student's ID number. This is the primary key. |
| NAME | String | The student's name. |

|  |  |  |
| --- | --- | --- |
| **EXAMINATION** | | |
| Name | Type | Description |
| STUDENT\_ID | Integer | The student's ID number. |
| SUBJECT | String | The subject's name. |

**Sample Data Tables**

|  |  |
| --- | --- |
| **STUDENT** | |
| ID | NAME |
| 1 | Taylor |
| 2 | Wesley |
| 3 | Jordan |
| 4 | Robin |
| 5 | Alex |

|  |  |
| --- | --- |
| **EXAMINATION** | |
| STUDENT\_ID | SUBJECT |
| 1 | Biology |
| 1 | Physics |
| 3 | History |
| 4 | Geography |
| 4 | Geography |

# Sample Output

1 Biology 1

1 Physics 1

3 History 1

4 Geography 2

**Explanation**

*Taylor* appeared in *Biology* examination one time. *Taylor* appeared in the *Physics* examination one time. *Jordan* appeared in the *History* examination one time.

*Robin* appeared in the *Geography* examination two times.

Question - 86

**The Perfect Arrangement**

Write a query to print the *id, first\_name* and *last\_name.* To filter the names, concatenate the first and last names to create a *combined name.* Return the names of customers whose combined names are less than 12 letters long. Sort the results by their combined name lengths, then alphabetically, case insensitive, by combined name, then by id. All sorts are ascending.

# Input Format CUSTOMER

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| ID | Integer | unique id, primary key. |
| FIRST\_NAME | String |  |
| LAST\_NAME | String |  |
| COUNTRY | String |  |
| CREDIT\_LIMIT | Float |  |

**Output Format**

CUSTOMER.ID CUSTOMER.FIRST\_NAME CUSTOMER.LAST\_NAME

**Sample Input** CUSTOMER

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **FIRST\_NAME** | **LAST\_NAME** | **COUNTRY** | CREDIT\_LIMIT |
| 1 | Alex | White | USA | 200350.54 |
| 2 | Tyler | Hanson | UK | 15354.23 |
| 3 | Jordan | Fernandez | France | 359200.67 |
| 4 | Drew | Bradley | Albania | 1060.57 |
| 5 | Blake | Fuller | USA | 14789.00 |
| 6 | Spencer | Johnston | China | 100243.35 |
| 7 | Ellis | Gutierrez | USA | 998999.20 |
| 8 | Morgan | Thomas | Canada | 500500.23 |
| 9 | Riley | Garza | UK | 18782.44 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10 | Peyton | Harris | USA | 158367.00 |

Sample Output

1

9

5

4

2

Alex White Riley Garza

Blake Fuller Drew Bradley

Tyler Hanson

# Explanation

AlexWhite is 9 letters, so it is included in the results. JordanFernandez is 15 letters, so it is omitted. The last 3 names are the same length, so they are sorted alphabetically ascending.

Names that are excluded and their lengths

MorganThomas 12

PeytonHarris 12

EllisGutierrez 14

JordanFernandez 15

SpencerJohnston 15

Question - 87

**Students Score**

The Math scores of students have been stored in the *STUDENT* table. Write a query to print the ids and the names of the students who have secured higher than the average score. The result should be sorted in ascending order of student ID.

# Input Format

|  |  |  |
| --- | --- | --- |
| STUDENT | | |
| Name | Type | Description |
| ID | Integer | A student ID in the inclusive range *[1, 1000]*. This field is the primary key. |
| NAME | String | A student name. This field contains between *1* and *100* characters (inclusive). |
| SCORE | Float | The Math score of the student. |

**Output Format**

The result should contain the ids and the names of the students who secured higher than the average score. The result should be sorted in ascending order of student ID.

STUDENT.ID STUDENT.NAME

**Sample Input**

|  |  |  |
| --- | --- | --- |
| STUDENT | | |
| ID | NAME | SCORE |
| 1 | Bob | 50 |
| 2 | John | 65.5 |
| 3 | Harry | 45 |
| 4 | Dick | 85 |
| 5 | Dev | 25 |
| 6 | Sid | 98 |
| 7 | Tom | 90 |
| 8 | Julia | 70.5 |
| 9 | Erica | 81 |
| 10 | Jerry | 85 |

**Sample Output**

4 Dick

1. Sid
2. Tom
3. Julia
4. Erica
5. Jerry

**Explanation**

The average score is *(50 + 65.5 + 45 + 85 + 25 + 98 + 90 + 70.5 + 81 + 85) / 10 = 69.5*.

*Bob* scored *50*, which is less than *69.5. John* scored *65.5*, which is less than *69.5. Harry* scored *45*, which is less than *69.5. Dick* scored *85*, which is more than *69.5. Dev* scored *25*, which is less than *69.5.*

*Sid* scored 98, which is more than *69.5 Tom* scored *90*, which is more than *69.5. Julia* scored *70.5*, which is more than *69.5. Erica* scored *81*, which is more than *69.5. Jerry* scored *85*, which is more than *69.5.*

*Sid, Dick, Tom, Julia, Erica*, and *Jerry* scored more than the average score.

Question - 88

**The First Orders**

A company maintains information about its orders in the *ORDERS* table. Write a query to print details of the earliest *five* orders (sorted by ORDER\_DATE, ascending) that have not been delivered (i.e., *STATUS* is not *DELIVERED*). If there are more than five orders to choose from, select the ones with the lowest order ID. Sort the output in the increasing order of order *ID*. The output should contain ID, ORDER\_DATE, STATUS, CUSTOMER\_ID.

**Schema**

**Table**: **Orders**

|  |  |
| --- | --- |
| **column name** | **column type** |
| id | int |
| order\_date | date |
| status | varchar(50) |
| customer\_id | int |

**Sample Data Tables**

Sample Input

|  |  |  |  |
| --- | --- | --- | --- |
| **ORDERS** | | | |
| **ID** | **ORDER\_DATE** | **STATUS** | **CUSTOMER\_ID** |
| 10100 | 2003-01-06 | PLACED | 363 |
| 10101 | 2003-01-06 | PLACED | 128 |
| 10102 | 2003-01-06 | IN TRANSIT | 181 |
| 10103 | 2003-01-06 | DELIVERED | 121 |
| 10104 | 2003-01-07 | DELIVERED | 114 |
| 10106 | 2003-01-07 | IN TRANSIT | 278 |
| 10120 | 2003-01-07 | PLACED | 114 |
| 10122 | 2003-05-05 | IN TRANSIT | 350 |
| 10123 | 2003-05-05 | DELIVERED | 103 |

Sample Output

|  |  |  |
| --- | --- | --- |
| 10100 | 2003-01-06 | PLACED 363 |
| 10101 | 2003-01-06 | PLACED 128 |
| 10102 | 2003-01-06 | IN TRANSIT 181 |
| 10106 | 2003-01-07 | IN TRANSIT 278 |
| 10120 | 2003-01-07 | PLACED 114 |

Explanation

The orders with order numbers *10100*, *10101*, *10102*, *10106*, and *10120* are the earliest placed orders and also have order status not equal to *DELIVERED*, so all their information in the increasing order of their order ID is printed.

Question - 89

**Customers Credit Limit**

A company maintains the data of its customers in the *CUSTOMER* table. Write a query to print the *ID*s and the *NAME*s of the customers who are from the USA and whose credit limit is greater than *100000,* ordered by increasing *ID* number.

# Input Format

|  |  |  |
| --- | --- | --- |
| CUSTOMER | | |
| Name | Type | Description |
| ID | Integer | A customer ID in the inclusive range *[1, 1000]*. This is the primary key. |
| NAME | String | A customer name. This field contains between *1* and *100* characters (inclusive). |
| COUNTRY | String | The country of the customer. |
| CREDITS | Integer | The credit limit of the customer. |

**Output Format**

The result should print the *ID*s and the *NAME*s of those customers who are from the USA and whose credit limit is greater than *100000*, in ascending *ID* order and in the following format:

CUSTOMER.ID CUSTOMER.NAME

**Sample Input**

|  |  |  |  |
| --- | --- | --- | --- |
| CUSTOMER | | | |
| ID | NAME | COUNTRY | CREDITS |
| 1 | Frances White | USA | 200350 |
| 2 | Carolyn Bradley | UK | 15354 |
| 3 | Annie Fernandez | France | 359200 |
| 4 | Ruth Hanson | Albania | 1060 |
| 5 | Paula Fuller | USA | 14789 |
| 6 | Bonnie Johnston | China | 100243 |
| 7 | Ruth Gutierrez | USA | 998999 |
| 8 | Ernest Thomas | Canada | 500500 |
| 9 | Joe Garza | UK | 18782 |
| 10 | Anne Harris | USA | 158367 |

**Sample Output**

1 Frances White

7 Ruth Gutierrez

10 Anne Harris

**Explanation**

Description of some of the customers is given below:

*Frances White* is from the *USA*, and his credit limit is *200350,* which is greater than *100000*. *Carolyn Bradley* is from the *UK*, and her credit limit is *15354*, which is less than *100000*.

*Paula Fuller* is from the *USA*, and her credit limit is *14789*, which is less than *100000*.

Remaining records are analyzed similarly.

So, *Frances White, Ruth Gutierrez,* and *Anne Harris* are from *USA* and credit is greater than *100000.*

A shopkeeper maintains the count of the different colored balls ( Red , green , and blue ) in the *COLLECTION* table. Each row of the table represents one of the following types:

Question - 90

**The Beautiful Collection**

GOOD : If the count of the red, green, and blue balls are equal.



BAD : If the count of any two colored balls are equal, i.e., only one of the following conditions is true:

 Red balls count is equal to green balls.  Red balls count is equal to blue balls.

 Green balls count is equal to blue balls.

WORSE : If all the colored balls have different counts.

Write a query to print the type which is represented by each row of the table. Note that the output is *case-sensitive*, so make sure to output only GOOD , BAD ,

or WORSE .

# Input Format

|  |  |  |
| --- | --- | --- |
| COLLECTION | | |
| Name | Type | Description |
| RED | Integer | This describes the count of red balls. The count can be between *30* units and *100* units (inclusive). |
| GREEN | Integer | This describes the count of green balls. The count can be between *30* units and *100* units (inclusive). |
| BLUE | Integer | This describes the count of blue balls. The count can be between *30* units and *100* units (inclusive). |

**Output Format**

Each row of the result should contain one of the types which are described above, in the following format. Note that the output is *case-sensitive*, so make sure



BAD

WORSE

to output only

GOOD

,

, or .

TYPE\_OF\_COLLECTION

**Sample Input**

|  |  |  |
| --- | --- | --- |
| COLLECTION | | |
| RED | GREEN | BLUE |
| 65 | 65 | 87 |
| 50 | 50 | 50 |
| 30 | 50 | 100 |
| 40 | 50 | 90 |
| 92 | 50 | 50 |

**Sample Output**

BAD GOOD

WORSE WORSE BAD

**Explanation**

The type of collection represented by each row is explained below:

The count of the red balls is equal to the count of the green balls. But the count of blue balls is not equal to the count of red or green balls. So, the type is



BAD .

The count of the red, green, and blue balls are equal. So, the type is GOOD .

The count of the red, green, and blue balls are different from each other. So, the type is WORSE . The count of the red, green, and blue balls are different from each other. So, the type is WORSE .

The count of the green balls is equal to the count of the blue balls. But the count of red balls is not equal to the count of green or blue balls. So, the type is



BAD .

Question - 91

**Big Companies**

An organization maintains employment data in three tables: *EMPLOYEE*, *COMPANY*, and *SALARY*. Write a query to print the names of every company where the average salary is greater than *40000*. Each distinct row of results in the output must contain the name of a company whose average employee salary is *> 40,000* in the COMPANY.NAME format.

**Schema**

|  |  |  |
| --- | --- | --- |
| **EMPLOYEE** | | |
| Name | Type | Description |
| ID | Integer | An employee ID in the inclusive range [1, 1000]. This is the primary key. |
| NAME | String | An employee name. This field contains between 1 and 100 characters (inclusive). |

|  |  |  |
| --- | --- | --- |
| **COMPANY** | | |
| Name | Type | Description |
| ID | Integer | A company ID in the inclusive range [1, 1000]. This is the primary key. |
| NAME | String | A company name. This field contains between 1 and 100 characters (inclusive). |

|  |  |  |
| --- | --- | --- |
| **SALARY** | | |
| Name | Type | Description |
| EMPLOYEE\_ID | Integer | An employee ID in the inclusive range [1, 1000]. |
| COMPANY\_ID | Integer | A company ID in the inclusive range [1, 1000]. |
| SALARY | Integer | The salary of the employee in the inclusive range [10000, 100000]. |

**Sample Data Tables**

|  |  |
| --- | --- |
| **EMPLOYEE** | |
| ID | NAME |
| 1 | Frances White |
| 2 | Carolyn Bradley |

|  |  |
| --- | --- |
| 3 | Annie Fernandez |
| 4 | Ruth Hanson |
| 5 | Paula Fuller |
| 6 | Bonnie Johnston |
| 7 | Ruth Gutierrez |
| 8 | Ernest Thomas |
| 9 | Joe Garza |
| 10 | Anne Harris |

|  |  |
| --- | --- |
| **COMPANY** | |
| ID | NAME |
| 1 | PeopleSoft Inc |
| 2 | Baker Hughes Incorporated |
| 3 | MDU Resources Group Inc. |
| 4 | DST Systems, Inc. |
| 5 | Williams Companies Inc |
| 6 | Fisher Scientific International Inc. |
| 7 | Emcor Group Inc. |
| 8 | Genuine Parts Company |
| 9 | MPS Group Inc. |
| 10 | Novellus Systems Inc |

|  |  |  |
| --- | --- | --- |
| **SALARY** | | |
| EMPLOYEE\_ID | COMPANY\_ID | SALARY |
| 2 | 4 | 27779 |
| 2 | 9 | 36330 |
| 3 | 9 | 71466 |
| 3 | 10 | 22804 |
| 5 | 5 | 49892 |
| 6 | 4 | 31493 |
| 6 | 10 | 26888 |
| 7 | 3 | 87118 |
| 7 | 7 | 70767 |
| 7 | 9 | 39929 |

Sample Output

MDU Resources Group Inc. Williams Companies Inc

Emcor Group Inc. MPS Group Inc.

Explanation

The following companies have average employee salaries *> 40,000*.

*MDU Resources Group Inc.* has one employee (*ID 7*) whose *SALARY* is *87118*. The company's average employee salary is *87118 ÷ 1 = 87118* and *87118 > 40000*.

*Williams Companies Inc* has one employee (*ID 5*) whose *SALARY* is *49892*. The company's average employee salary is *49892 ÷ 1 = 49892*. *Emcor Group Inc.* has one employee (*ID 7*) whose *SALARY* is *70767*. The company's average employee salary is *70767 ÷ 1 = 70767*.

*MPS Group Inc.* has three employees (*ID*s *2*, *3*, and *7*) whose *SALARY* values are *36330*, *71466*, and *39929*. The company's average employee salary *(36330 + 71466 + 39929) ÷ 3 = 49241.67*.

Question - 92

**Scheduling Errors**

Write a query to return a list of professor names and their associated courses for all courses outside of their departments. There should be no duplicate rows, but they can be in any order.

The output should contain two columns: *professor.name, course.name*.

**Schema**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **PROFESSOR** | | |  | **DEPARTMENT** | | |
| **Name** | **Type** | **Description** | **Name** | **Type** | **Description** |
| ID | Integer | unique id, primary key | ID | Integer | unique id, primary key |
| NAME | String |  | NAME | String |  |
| DEPARTMENT\_ID | Integer | foreign key, *department.id* |  | | | |
| SALARY | Integer |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **COURSE** | | |  | **SCHEDULE** | | |
| **Name** | **Type** | **Description** | **Name** | **Type** | **Description** |
| ID | Integer | unique id, primary key | PROFESSOR\_ID | Integer | foreign key, *professor.id* |
| NAME | String |  | COURSE\_ID | Integer | foreign key, *course.id* |
| DEPARTMENT\_ID | Integer | foreign key, *department.id* | SEMESTER | Integer |  |
| CREDITS | Integer |  | YEAR | Integer |  |

**Sample Data Tables**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **PROFESSOR** | | | |  | **DEPARTMENT** | |
| **ID** | **NAME** | **DEPARTMENT\_ID** | **SALARY** | **ID** | **NAME** |
| 1 | Alex Daniels | 4 | 7169 | 3 | Biological Sciences |
| 2 | Drew Knight | 1 | 9793 | 5 | Technology |
| 3 | Jordan Myers | 4 | 25194 | 6 | Humanities & Social Sciences |
| 4 | Tyler Rodriguez | 3 | 9686 | 2 | Clinical Medicine |
| 5 | Blake Gome | 2 | 30860 | 4 | Arts and Humanities |
| 6 | Spencer George | 5 | 10487 | 1 | Physical Sciences |
| 7 | Ellis Vasquez | 4 | 6353 |  | | |
| 8 | Morgan Flores | 1 | 25796 |
| 9 | Riley Gilbert | 5 | 35678 |
| 10 | Peyton Stevens | 2 | 26648 |

|  |  |  |  |
| --- | --- | --- | --- |
| **COURSE** | | | |
| **ID** | **NAME** | **DEPARTMENT\_ID** | **CREDITS** |
| 9 | Clinical Biochemistry | 2 | 3 |
| 4 | Astronomy | 1 | 6 |
| 10 | Clinical Neuroscience | 2 | 5 |
| 1 | Pure Mathematics and Mathematical Statistics | 1 | 3 |
| 6 | Geography | 1 | 7 |
| 8 | Chemistry | 1 | 1 |
| 5 | Physics | 1 | 8 |
| 3 | Earth Science | 1 | 7 |
| 7 | Materials Science and Metallurgy | 1 | 5 |
| 2 | Applied Mathematics and Theoretical Physics | 1 | 5 |

|  |  |  |  |
| --- | --- | --- | --- |
| **SCHEDULE** | | | |
| **PROFESSOR\_ID** | **COURSE\_ID** | **SEMESTER** | **YEAR** |
| 4 | 4 | 3 | 2003 |
| 3 | 3 | 1 | 2011 |
| 1 | 7 | 5 | 2011 |
| 7 | 7 | 1 | 2010 |
| 4 | 6 | 1 | 2001 |
| 9 | 3 | 1 | 2012 |
| 10 | 2 | 4 | 2009 |
| 1 | 1 | 3 | 2014 |
| 1 | 2 | 3 | 2008 |
| 1 | 7 | 5 | 2007 |

# Sample Output

Tyler Rodriguez Astronomy

Jordan Myers Earth Sciences

Alex Daniels Materials Science and Metallurgy Ellis Vasquez Materials Science and Metallurgy Tyler Rodriguez Geography

Riley Gilbert Earth Sciences

Peyton Stevens Applied Mathematics and Theoretical Physics Alex Daniels Pure Mathematics and Mathematical Statistics Alex Daniels Applied Mathematics and Theoretical Physics

Alex Daniels Materials Science and Metallurgy

**Explanation**

Example logic

1. Professor *Tyler Rodriguez's department\_id* is *3*, but the *Astronomy* course's *department\_id* is *1*.
2. Professor *Jordan Myers*'s *department\_id* is *4*, but the *Earth Sciences* course's *department\_id* is *1*

Question - 93

**List the Course Names**

Write a query to return a list of professor names and their associated courses. The results can be in any order but must not contain duplicate rows.

**Schema**

|  |  |  |
| --- | --- | --- |
| **PROFESSOR** | | |
| Name | Type | Description |
| ID | Integer | unique id, primary key |
| NAME | String |  |
| DEPARTMENT\_ID | Integer | This is a foreign key to DEPARTMENT.ID. |
| SALARY | Integer |  |

|  |  |  |
| --- | --- | --- |
| **DEPARTMENT** | | |
| Name | Type | Description |
| ID | Integer | unique id, primary key |
| NAME | String |  |

|  |  |  |
| --- | --- | --- |
| **COURSE** | | |
| Name | Type | Description |
| ID | Integer | unique id, primary key |
| NAME | String |  |
| DEPARTMENT\_ID | Integer | foreign key to DEPARTMENT.ID |
| CREDITS | Integer |  |

|  |  |  |
| --- | --- | --- |
| **SCHEDULE** | | |
| Name | Type | Description |
| PROFESSOR\_ID | Integer | foreign key to PROFESSOR.ID |
| COURSE\_ID | Integer | foreign key to COURSE.ID |
| SEMESTER | Integer |  |
| YEAR | Integer |  |

**Sample Data Tables**

|  |  |  |  |
| --- | --- | --- | --- |
| **PROFESSOR** | | | |
| ID | NAME | DEPARTMENT\_ID | SALARY |
| 1 | Alex Burton | 5 | 7340 |
| 8 | Jordan Diaz | 1 | 17221 |
| 9 | Drew Hicks | 5 | 16613 |
| 2 | Tyler Matthews | 2 | 14521 |
| 10 | Blake Foster | 4 | 28526 |
| 3 | Spencer Peters | 1 | 10487 |
| 4 | Ellis Marshall | 3 | 6353 |
| 7 | Morgan Lee | 2 | 25796 |
| 5 | Riley Peterson | 1 | 35678 |
| 6 | Peyton Fields | 5 | 26648 |

|  |  |
| --- | --- |
| **DEPARTMENT** | |
| ID | NAME |
| 3 | Biological Sciences |
| 5 | Technology |
| 6 | Humanities & Social Sciences |
| 2 | Clinical Medicine |
| 4 | Arts and Humanities |
| 1 | Physical Sciences |

|  |  |  |  |
| --- | --- | --- | --- |
| **COURSE** | | | |
| ID | NAME | DEPARTMENT\_ID | CREDITS |
| 9 | Clinical Biochemistry | 2 | 3 |
| 4 | Astronomy | 1 | 6 |
| 10 | Clinical Neuroscience | 2 | 5 |
| 1 | Pure Mathematics and Mathematical Statistics | 1 | 3 |
| 6 | Geography | 1 | 7 |
| 8 | Chemistry | 1 | 1 |
| 5 | Physics | 1 | 8 |
| 3 | Earth Science | 1 | 7 |
| 7 | Materials Science and Metallurgy | 1 | 5 |
| 2 | Applied Mathematics and Theoretical Physics | 1 | 5 |

|  |  |  |  |
| --- | --- | --- | --- |
| **SCHEDULE** | | | |
| PROFESSOR\_ID | COURSE\_ID | SEMESTER | YEAR |
| 5 | 3 | 6 | 2012 |
| 7 | 3 | 1 | 2013 |
| 5 | 7 | 6 | 2010 |
| 2 | 10 | 2 | 2004 |
| 5 | 1 | 1 | 2011 |
| 2 | 9 | 4 | 2005 |
| 7 | 10 | 6 | 2009 |
| 5 | 6 | 4 | 2007 |
| 7 | 9 | 1 | 2014 |
| 9 | 9 | 5 | 2011 |

# Sample Output

Tyler Matthews Clinical Biochemistry Tyler Matthews Clinical Neuroscience Drew Hicks Clinical Biochemistry

Morgan Lee Clinical Biochemistry Morgan Lee Clinical Neuroscience Morgan Lee Earth Science

Riley Peterson Earth Science Riley Peterson Geography

Riley Peterson Materials Science and Metallurgy

Riley Peterson Pure Mathematics and Mathematical Statistics

Question - 94

**Professor Names and Salaries**

A university maintains data on professors and departments in two tables: *PROFESSOR* and *DEPARTMENT*. Write a query to print the NAME and SALARY for each professor who satisfies the following two requirements:

The professor does not work in the *Arts and Humanities* department.

The professor's salary is *greater than* the *smallest* salary of any professor in the *Arts and Humanities* department.

The name must be printed before the salary, but row order does not matter.

**Schema**

|  |  |  |
| --- | --- | --- |
| **DEPARTMENT** | | |
| Name | Type | Description |
| ID | Integer | A department ID in the inclusive range [1, 1000]. This is a primary key. |
| NAME | String | A department name. This field contains between 1 and 100 characters. |

|  |  |  |
| --- | --- | --- |
| **PROFESSOR** | | |
| Name | Type | Description |

|  |  |  |
| --- | --- | --- |
| ID | Integer | A professor's ID in the inclusive range [1, 1000]. This is a primary key. |
| NAME | String | A professor's name. This field contains between 1 and 100 characters. |
| DEPARTMENT\_ID | Integer | A professor's department ID. This is a foreign key to DEPARTMENT.ID. |
| SALARY | Integer | A professor's salary in the inclusive range [5000, 40000]. |

**Sample Data Tables**

|  |  |
| --- | --- |
| **DEPARTMENT** | |
| ID | NAME |
| 3 | Biological Sciences |
| 5 | Technology |
| 6 | Humanities & Social Sciences |
| 2 | Clinical Medicine |
| 4 | Arts and Humanities |
| 1 | Physical Sciences |

|  |  |  |  |
| --- | --- | --- | --- |
| **PROFESSOR** | | | |
| ID | NAME | DEPARTMENT\_ID | SALARY |
| 1 | Shauna Rivera | 1 | 22606 |
| 8 | Ruth Price | 3 | 9287 |
| 9 | Julie Gonzalez | 4 | 18870 |
| 2 | Craig Elliott | 5 | 27524 |
| 10 | Scott Butler | 1 | 26200 |
| 3 | Nancy Russell | 2 | 7076 |
| 4 | Clarence Johnson | 1 | 7249 |
| 7 | Louis Schmidt | 1 | 13437 |
| 5 | Terri Thompson | 3 | 28432 |
| 6 | Keith Gilbert | 5 | 12610 |

# Sample Output

**Explanation**

Shauna Rivera 22606

Craig Elliott 27524

Terri Thompson 28432

Scott Butler 26200

Julie Gonzalez has a salary of 18870 , which is smaller than the salary of any other professor in the Arts and Humanities department. The following employees of other departments have salaries higher than Julie's:

Shauna Rivera 's salary of 22606 is higher than Julie Gonzalez 's.

Craig Elliott 's salary of 27524 is higher than Julie Gonzalez 's.

Terri Thompson 's salary of 28432 is higher than Julie Gonzalez 's.

Scott Butler 's salary of 26200 is higher than Julie Gonzalez 's.

Question - 95

**Student's Major**

A university maintains data on students and their majors in three tables: STUDENTS, MAJORS, and REGISTER. The university needs a list of STUDENT\_NAME and MAJOR\_NAME. Sort the list by STUDENT\_ID and return the first 20 records.

**Table Schema**

'

|  |  |  |
| --- | --- | --- |
|  | | |
| **Name** | **Type** | **Description** |
| STUDENT\_ID | Integer | The ID of a student. This is a *foreign key*. |
| MAJOR\_ID | Integer | The ID of a major. This is a *foreign key*. |

**Sample Case 0**

# Sample Input 0

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **STUDENTS** | | |  | **MAJORS** | |  | **REGISTER** | |
| **STUDENT\_ID** | **STUDENT\_NAME** | **STUDENT\_AGE** | **MAJOR\_ID** | **MAJOR\_NAME** | **STUDENT\_ID** | **MAJOR\_ID** |
| 1 | John | 20 | 1000 | Computer Science | 2 | 1000 |
| 2 | Masie | 21 | 2000 | Biology | 3 | 3000 |
| 3 | Harry | 21 | 3000 | Physics | 1 | 2000 |

**Sample Output**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **STUDENTS** | | |  | **MAJORS** | | |
| **Name** | **Type** | **Description** | **Name** | **Type** | **Description** |
| STUDENT\_ID | Integer | The ID of a student. This is the *primary key*. | MAJOR\_ID | Integer | The ID of a major. This is the *primary key*. |
| STUDENT\_NAME | String | The name of the student. | MAJOR\_NAME | String | The name of a major. |
| STUDENT\_AGE | Integer | The age of the student. | **REGISTER** | | | |

John Biology

Masie Computer Science Harry Physics

Question - 96

**Student Rank**

A university stores students' standardized test scores in a table named *STUDENT*. Student X placed *213th* on the test. Write a query to find Student X's test score (i.e., the *213th* highest *STUDENT.SCORE* in *STUDENT*).

**Schema**

A table is provided: STUDENT .

|  |  |  |
| --- | --- | --- |
| **STUDENT** | | |
| **Name** | **Type** | **Description** |
| ID | Integer | The student's unique ID. This is a *primary key*. |
| AGE | Integer | The student's age. |
| SCORE | Integer | The student's standardized test score. |

**Sample Data Tables**

|  |  |  |
| --- | --- | --- |
| **STUDENT** | | |
| **ID** | **AGE** | **SCORE** |
| 1 | 19 | 91 |
| 2 | 20 | 90 |
| 3 | 20 | 87 |
| 4 | 21 | 72 |
| 5 | 19 | 98 |
| 6 | 20 | 50 |

The table's scores (from highest to lowest) are *{98, 91, 90, 87, 72, 50}*. As an example, the fourth-highest score is *87*. For the real query, find the 213th highest score.

Question - 97

**Clumsy Administrator**

A company maintains the records of all employees. The company pays the database administrator too little so the work has been quite clumsy. The administrator carelessly inserted the records of many employees into the employee records table multiple times. An employee's record is considered duplicate only if all columns (fields) of the employee's record are duplicated.

Write a query to find the names of employees whose records occur more than once in the table. Each name should appear only once in the results. The order of results does not matter.

**Schema**

You are provided *1* table: EMPLOYEE .

|  |  |  |
| --- | --- | --- |
| **EMPLOYEE** | | |
| Name | Type | Description |
| NAME | STRING | The name of the employee. |
| PHONE | STRING | The telephone number of the employee. |
| AGE | INTEGER | The age of the employee. |

**Sample Data Tables**

|  |  |  |
| --- | --- | --- |
| **EMPLOYEE** | | |
| **NAME** | **PHONE** | **AGE** |
| Sam | 1000040000 | 30 |
| Alex | 1000020000 | 60 |
| Alex | 1000020012 | 65 |
| Sam | 1000040000 | 30 |
| Chris | 1000012000 | 34 |
| Chris | 1000012000 | 34 |

Here, the exact records of Sam and Chris occur more than once in the table. Hence, Sam and Chris are the employees in the resultant output.

Question - 98

**Accounting Software Balance Report**

HackerFinance is developing an accounting software and needs to generate a list of customers and the change in their account balances during December.

The output should display each customer and their corresponding balance, formatted as "customer balance". The balance should show two places after the decimal, e.g., 0.00.

The change is calculated as the sum of debit transactions minus the sum of credit transactions during December. The results should be sorted in ascending order by customer name.

**Schema**

There is 1 table:

|  |  |  |
| --- | --- | --- |
| **transactions** | | |
| **name** | **type** | **description** |
| dt | VARCHAR(19) | Transaction timestamp |
| customer | VARCHAR(64) | Customer name |
| debit | DECIMAL(5,2) | Transaction debit |
| credit | DECIMAL(5,2) | Transaction credit |

**Sample Data Tables**

For the sample data in table:

**transactions**

|  |  |  |  |
| --- | --- | --- | --- |
| **dt** | **customer** | **debit** | **credit** |
| 2021-11-30 12:48:22 | Arney Cuff | 6.43 | 16.12 |
| 2021-12-25 19:00:46 | Arney Cuff | 97.78 | 12.53 |
| 2021-11-27 21:34:24 | Donaugh Furneaux | 89.71 | 85.04 |
| 2021-11-25 07:31:37 | Ferrell Brunn | 63.58 | 28.58 |
| 2021-11-25 15:30:56 | Gibbie Jurisic | 25.81 | 13.75 |
| 2021-11-21 00:09:50 | Harley Lyddiard | 57.49 | 7.11 |
| 2021-12-01 07:37:42 | Harley Lyddiard | 48.33 | 82.35 |
| 2021-12-02 13:08:52 | Harley Lyddiard | 12.13 | 63.81 |
| 2021-11-24 03:51:13 | Kippy Jelly | 50.34 | 12.91 |
| 2021-12-04 10:11:40 | Latrina Jackman | 10.73 | 39.51 |
| 2021-12-12 13:02:50 | Latrina Jackman | 5.35 | 96.74 |
| 2021-12-20 17:31:44 | Latrina Jackman | 54.99 | 92.73 |
| 2021-12-30 13:40:43 | Maribel Braim | 57.06 | 21.37 |
| 2021-12-02 09:57:35 | Orrin Curley | 65.44 | 51.31 |
| 2021-12-14 19:57:25 | Orrin Curley | 40.04 | 96.44 |
| 2021-11-20 09:28:11 | Rasla Venny | 80.33 | 20.69 |
| 2021-12-08 03:31:31 | Rasla Venny | 55.43 | 99.04 |
| 2021-12-09 21:31:29 | Rasla Venny | 87.96 | 5.87 |
| 2021-12-11 13:02:54 | Rasla Venny | 45.42 | 55.81 |
| 2021-12-28 18:04:52 | Rasla Venny | 68.17 | 85.30 |

the expected output is:

|  |  |
| --- | --- |
| **customer ▲** | **balance** |
| Arney Cuff | 85.25 |
| Harley Lyddiard | -85.70 |
| Latrina Jackman | -157.91 |
| Maribel Braim | 35.69 |
| Orrin Curley | -42.27 |
| Rasla Venny | 10.96 |