## 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   name                  city                    total_spending
1                   Customer 1       Los Angeles       150
3                   Customer 3       Chicago              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 |
| 1 | lwycliff6@list-manage.com |
| 1 | ekimbleyf@scientificamerican.com |
| 2 | bgooro@spotify.com |
| 2 | vsamwayest@bbb.org |
| 3 | apappin0@yellowbook.com |
| 3 | ringreyb@businessinsider.com |
| 3 | mrysonm@istockphoto.com |
| 5 | ayushin1c@opera.com |
| 5 | 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 |
| 2 | lcaiger1@si.edu |
| 3 | 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  6544.77
hratke0@disqus.com     6861.50
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.

Question - 9
**SQL: Customer Domain Ownership Report**

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.

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 |
| 2 | agammade1@comcast.net |
| 3 | goloshkin2@reference.com |
| 4 | cantonescu3@earthlink.net |
| 5 | fparzis4@ow.ly |
| 6 | cpetroulis5@shutterfly.com |
| 7 | tbeels6@bbb.org |
| 8 | zmacturlough7@4shared.com |
| 9 | eshury8@skype.com |
| 10 | 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 2
cantonescu3@earthlink.net 2
ebayldon0@washingtonpost.com  3
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' owns 3 domains, while 'agammade1@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 |
| 1 | user2@example.com |
| 2 | user3@example.com |
| 2 | user4@example.com |
| 2 | user5@example.com |
| 3 | user6@example.com |
| 1 | user7@example.com |
| 2 | user8@example.com |
| 1 | user9@example.com |
| 3 | 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.

| | | |
|---|---|---|
| 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              |3                    |No              |
|Sunshine Project|4              |3                    |Yes             |
+----------------+--------------+--------------------+---------------------------------+
```

## Question - 34
**SQL: Ethereum Market Dashboard Analysis**