Crypto Dataset Analysis using SQL

| Datasets: | Source Code: |
|--|--------------|
| <u>Transaction.csv</u> | Code |
| Prices.csv | |
| •members.csv | |
| ◆Cryoto.sql | |

When we explore datasets, especially in a relational database context, understanding the relationships between different tables is crucial. Here's a general outline of steps might we take while exploring datasets:

- 1. **Identify Tables:** Start by identifying all the tables present in the dataset. Each table represents a different entity or concept, such as members, prices, transactions, etc.
- Understand Table Structures: Examine the structure of each table, including
 the columns it contains, their data types, and any constraints or indexes defined
 on the table.
- **3. Explore Relationships:** Determine how the tables are related to each other. This typically involves looking for foreign key constraints that establish connections between tables.
- **4. Analyze Data Distribution:** Analyze the data distribution within each table. Look for patterns, outliers, missing values, and any other anomalies that may need further investigation.
- **5. Aggregate and Summarize Data:** Use aggregation functions to summarize data and gain insights. For example, calculate average values, counts, sums, etc., to understand trends and patterns within the dataset.
- **6. Join Tables:** Combine data from multiple tables using SQL JOIN operations. This allows us to create comprehensive views of the data that incorporate information from different sources.
- 7. **Iterate and Refine:** Continue iterating through these steps as we gain more insights and refine your understanding of the dataset. Don't hesitate to explore different angles or hypotheses to uncover hidden patterns or relationships.

Question 1: Explore Dataset.

Input

SELECT * FROM members

LIMIT 3;

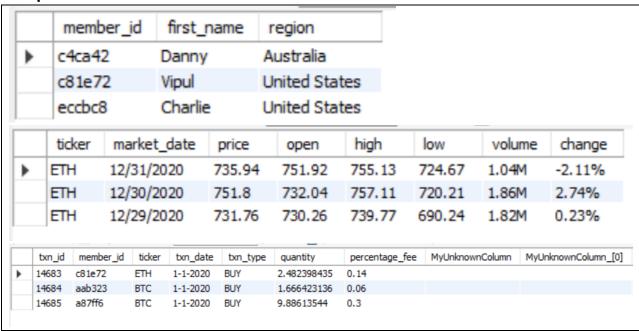
SELECT * FROM prices

LIMIT 3;

SELECT * FROM transactions

LIMIT 3;

Output



Question 2: Sort all the rows in the table by first_name in alphabetical order and show the top 3 rows.

Input:

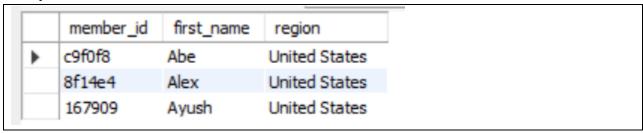
SELECT * FROM members ORDER BY first_name LIMIT 3;



Question 3: Which records from trading members are from the United States region? **Input:**

SELECT * FROM members
WHERE region = 'United States';

Output:

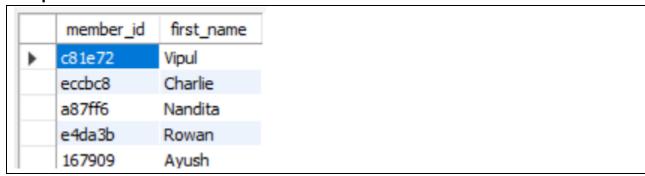


Question 4: Select only the member_id and first_name columns for members not from Australia.

Input:

SELECT
member_id,
first_name
FROM members
WHERE region != 'Australia';

Output:

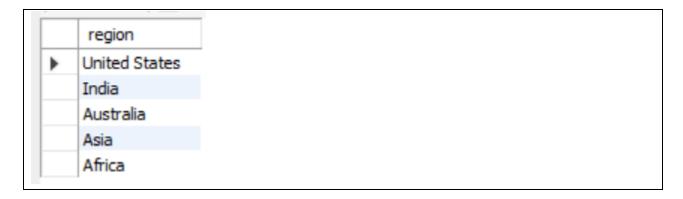


Question 5: Return the unique region values from the trading.members table and sort the output by reverse —alphabetical order

Input:

SELECT DISTINCT region FROM members

ORDER BY region DESC;

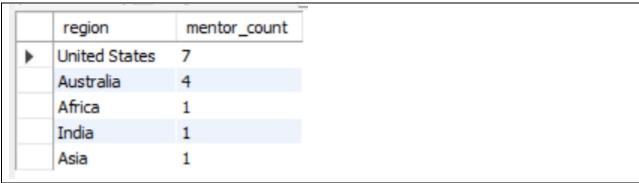


Question 6: How many mentors are there per region? Sort the output by regions with the most mentors to the least.

Input:

SELECT
region,
COUNT(*) AS mentor_count
FROM members
GROUP BY region
ORDER BY mentor_count DESC;

Output:



Question 7: How many US mentors and non-US mentors are there? **Input:**

```
SELECT
(CASE
WHEN members.region != 'United States' THEN 'Non US'
ELSE members.region
END) mentor_region,
COUNT(*) AS mentor_count
FROM members
GROUP BY (CASE
WHEN members.region != 'United States' THEN 'Non US'
```

ELSE members.region
END)
ORDER BY mentor_count DESC;

Output:



Question 8: How many total records do we have in the prices table? **Input:**

SELECT
COUNT(*) AS total_records
FROM prices;

Output:



Question 9: How many records are there per ticker value? **Input:**

SELECT
ticker,
COUNT(*) AS record_count
FROM prices
GROUP BY ticker;

Output:

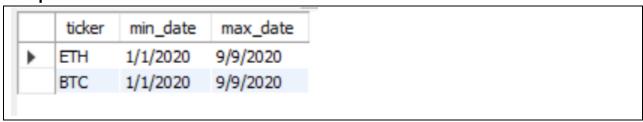


Question 10: What are the minimum and maximum market_date values? **Input:**

SELECT
ticker,
MIN(market_date) AS min_date,

MAX(market_date) AS max_date
FROM prices
GROUP BY ticker;

Output:



Question 11: Are there any duplicate market_date values for any ticker value in our table?

Input:

SELECT ticker,

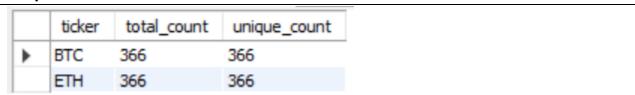
COUNT(market_date) AS total_count,

COUNT(DISTINCT market_date) AS unique_count

FROM prices

GROUP BY ticker;

Output:



Question 12: How many days from the prices table exist where the high price of Bitcoin is over \$20,000?

Input:

SELECT

COUNT(*) AS row_count

FROM prices

WHERE ticker = 'BTC'

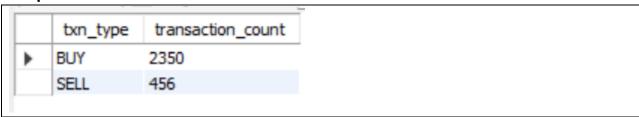
AND high > 20000;



Question 13: How many buy and sell transactions are there for Bitcoin? **Input:**

```
SELECT
txn_type,
COUNT(*) AS transaction_count
FROM transactions
WHERE ticker = 'BTC'
GROUP BY txn_type;
```

Output:



Question 14: Which members have sold less than 500 Bitcoin? Sort the output from the most BTC sold to the least.

Input:

```
WITH cte AS (
SELECT
member_id,
SUM(quantity) AS btc_sold_quantity
FROM transactions
WHERE ticker = 'BTC'
AND txn_type = 'SELL'
GROUP BY member_id
)
SELECT * FROM cte
WHERE btc_sold_quantity < 500
ORDER BY btc_sold_quantity DESC;
```

| | member_id | btc_sold_quantity |
|-------------|-----------|--------------------|
| > | c4ca42 | 331.5233657570001 |
| | c51ce4 | 248.521892159 |
| | e4da3b | 218.50541984500003 |
| | c20ad4 | 211.289496584 |
| | c81e72 | 204.49363130999998 |
| | aab323 | 181.02844811 |

Question 15: Which member_id has the highest buy to sell ratio by quantity? **Input:**

```
SELECT
member_id,
SUM(CASE WHEN txn_type = 'BUY' THEN quantity ELSE 0 END) /
SUM(CASE WHEN txn_type = 'SELL' THEN quantity ELSE 0 END) AS
buy_to_sell_ratio
FROM transactions
GROUP BY member_id
ORDER BY buy_to_sell_ratio DESC;
```

Output:

| Г | member_id | buy_to_sell_ratio |
|---|--------------------|--------------------|
| • | 45c48c | 16.067115846488733 |
| | c9f0f8 | 6.748227183245516 |
| | 8f1 4e4 | 6.74426346848583 |
| | a87ff6 | 6.578310625697089 |
| | eccbc8 | 6.136423045415139 |
| | aab323 | 5.412857603464859 |

Question 16: Which top 3 mentors have the most Bitcoin quantity as of the end of the period?

Input:

```
SELECT
members.first_name,
SUM(
CASE
WHEN transactions.txn_type = 'BUY' THEN transactions.quantity
```

WHEN transactions.txn_type = 'SELL' THEN transactions.quantity END

) AS total_quantity

FROM transactions

INNER JOIN members

ON transactions.member_id = members.member_id

WHERE ticker = 'BTC'

GROUP BY members.first_name

ORDER BY total_quantity DESC

LIMIT 3;

| | first_name | total_quantity |
|----------|------------|--------------------|
| • | Leah | 1344.1898886250003 |
| | Danny | 1281.7499713389998 |
| | Abe | 1206.236215487 |