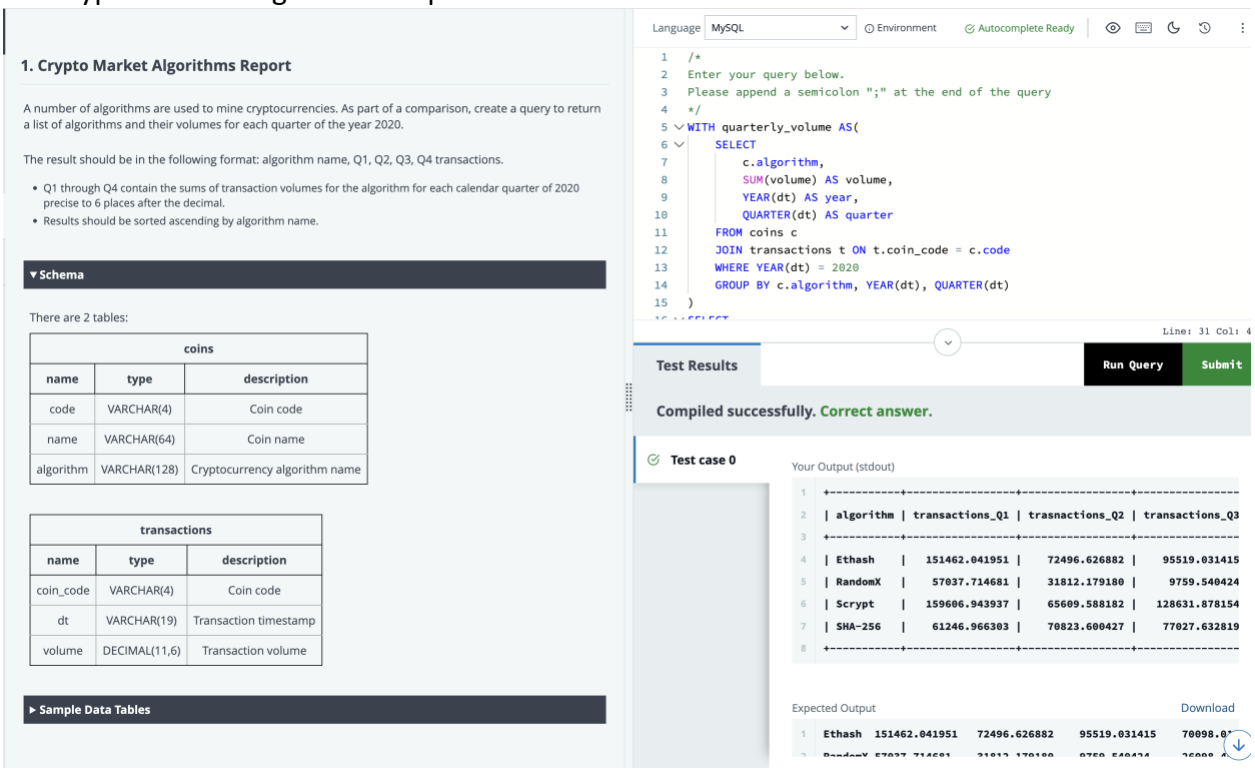
SQL (Advanced)

1. Crypto Market Algorithms Report



1. Crypto Market Algorithms Report

A number of algorithms are used to mine cryptocurrencies. As part of a comparison, create a query to return a list of algorithms and their volumes for each quarter of the year 2020.

The result should be in the following format: algorithm name, Q1, Q2, Q3, Q4 transactions.

- Q1 through Q4 contain the sums of transaction volumes for the algorithm for each calendar quarter of 2020 precise to 6 places after the decimal.
- Results should be sorted ascending by algorithm name.

▾ Schema

There are 2 tables:

coins		
name	**type**	**description**
code	VARCHAR(4)	Coin code
name	VARCHAR(64)	Coin name
algorithm	VARCHAR(128)	Cryptocurrency algorithm name

transactions		
name	**type**	**description**
coin_code	VARCHAR(4)	Coin code
dt	VARCHAR(19)	Transaction timestamp
volume	DECIMAL(11,6)	Transaction volume

▸ Sample Data Tables

Language: MySQL | Environment | Autocomplete Ready

```
/*
Enter your query below.
Please append a semicolon ";" at the end of the query
*/
WITH quarterly_volume AS(
    SELECT
        c.algorithm,
        SUM(volume) AS volume,
        YEAR(dt) AS year,
        QUARTER(dt) AS quarter
    FROM coins c
    JOIN transactions t ON t.coin_code = c.code
    WHERE YEAR(dt) = 2020
    GROUP BY c.algorithm, YEAR(dt), QUARTER(dt)
)
```

Line: 31 Col: 4

Test Results | Run Query | Submit

Compiled successfully. Correct answer.

Test case 0

Your Output (stdout)

```
+-----------+-----------------+-----------------+-----------------
| algorithm | transactions_Q1 | trasnactions_Q2 | transactions_Q3
+-----------+-----------------+-----------------+-----------------
| Ethash    |   151462.041951 |    72496.626882 |    95519.031415
| RandomX   |    57037.714681 |    31812.179180 |     9759.540424
| Scrypt    |   159606.943937 |    65609.588182 |   128631.878154
| SHA-256   |    61246.966303 |    70823.600427 |    77027.632819
+-----------+-----------------+-----------------+-----------------
```

Expected Output | Download

```
Ethash  151462.041951   72496.626882    95519.031415    70098.0
```

Language MySQL | Environment | Autocomplete Ready

```sql
/*
Enter your query below.
Please append a semicolon ";" at the end of the query
*/
WITH quarterly_volume AS(
    SELECT
        c.algorithm,
        SUM(volume) AS volume,
        YEAR(dt) AS year,
        QUARTER(dt) AS quarter
    FROM coins c
    JOIN transactions t ON t.coin_code = c.code
    WHERE YEAR(dt) = 2020
    GROUP BY c.algorithm, YEAR(dt), QUARTER(dt)
)
SELECT
    c.algorithm,
    qv1.volume AS transactions_Q1,
    qv2.volume AS trasnactions_Q2,
    qv3.volume AS transactions_Q3,
    qv4.volume AS transactions_Q4
FROM coins c
LEFT JOIN quarterly_volume qv1
    ON c.algorithm = qv1.algorithm AND qv1.year = 2020 AND qv1.quarter = 1
LEFT JOIN quarterly_volume qv2
    ON c.algorithm = qv2.algorithm AND qv2.year = 2020 AND qv2.quarter = 2
LEFT JOIN quarterly_volume qv3
    ON c.algorithm = qv3.algorithm AND qv3.year = 2020 AND qv3.quarter = 3
LEFT JOIN quarterly_volume qv4
    ON c.algorithm = qv4.algorithm AND qv4.year = 2020 AND qv4.quarter = 4
WHERE c.code <> "DOGE" -- NOT LIKE == <>
ORDER BY c.algorithm
```

Line: 36 Col: 1

Test Results | Run Query | Submit

▼ Sample Data Tables

For the sample data in tables:

coins		
code	name	algorithm
BTC	Bitcoin	SHA-256
DOGE	Dogecoin	Scrypt
ETH	Etherium	Ethash
LTC	Litecoin	Scrypt
XMR	Monero	RandomX

transactions		
coin_code	dt	volume
LTC	2021-04-05 04:30:30	422.830473
DOGE	2020-01-22 07:44:35	55.225905
BTC	2019-10-19 06:32:46	6906.520151
DOGE	2020-01-29 12:21:19	9370.519560
BTC	2020-04-05 12:38:54	4775.700964
DOGE	2019-12-16 13:41:53	8865.966189
BTC	2019-12-07 02:24:27	9086.928039
ETH	2020-04-21 21:27:43	7273.565939
DOGE	2020-08-25 19:09:10	8027.924009
ETH	2020-06-04 03:53:39	6736.423699
DOGE	2019-12-23 00:13:13	6704.032765

Language MySQL | Environment | Autocomplete Ready

```sql
    qv3.volume AS transactions_Q3,
    qv4.volume AS transactions_Q4
FROM coins c
LEFT JOIN quarterly_volume qv1
    ON c.algorithm = qv1.algorithm AND qv1.year = 2020 AND qv1.quarter = 1
LEFT JOIN quarterly_volume qv2
    ON c.algorithm = qv2.algorithm AND qv2.year = 2020 AND qv2.quarter = 2
LEFT JOIN quarterly_volume qv3
    ON c.algorithm = qv3.algorithm AND qv3.year = 2020 AND qv3.quarter = 3
LEFT JOIN quarterly_volume qv4
    ON c.algorithm = qv4.algorithm AND qv4.year = 2020 AND qv4.quarter = 4
WHERE c.code <> "DOGE" -- NOT LIKE == <>
ORDER BY c.algorithm
```

Line: 36 Col: 1

Test Results | Run Query | Submit

Compiled successfully. Correct answer.

Test case 0

Your Output (stdout)

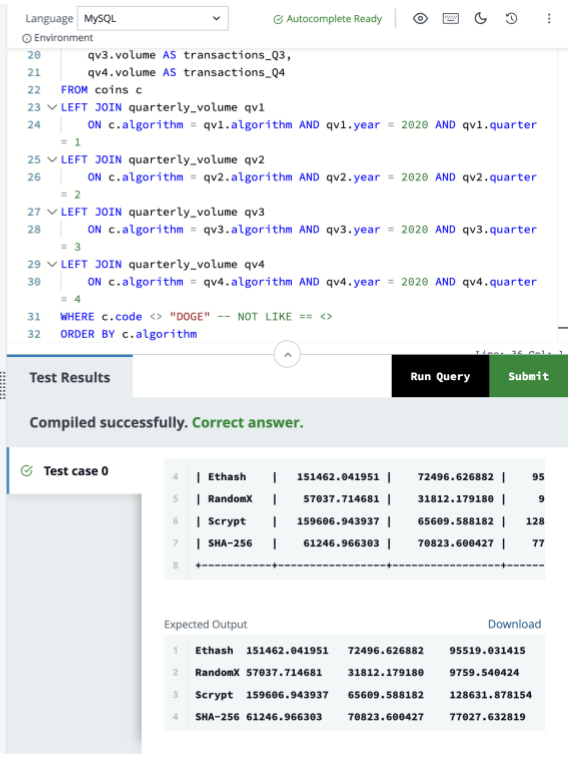
```
+-----------+-----------------+-----------------+-----------------+-----------------+
| algorithm | transactions_Q1 | trasnactions_Q2 | transactions_Q3 | transactions_Q4 |
+-----------+-----------------+-----------------+-----------------+-----------------+
| Ethash    |   151462.041951 |    72496.626882 |    95519.031415 |    70098.019852 |
| RandomX   |    57037.714681 |    31812.179180 |     9759.540424 |    26098.469680 |
| Scrypt    |   159606.943937 |    65609.588182 |   128631.878154 |    78124.251634 |
| SHA-256   |    61246.966303 |    70823.600427 |    77027.632819 |    99171.741237 |
+-----------+-----------------+-----------------+-----------------+-----------------+
```

Expected Output | Download

DOGE	2020-01-29 12:21:19	9370.519560
BTC	2020-04-05 12:38:54	4775.700964
DOGE	2019-12-16 13:41:53	8865.966189
BTC	2019-12-07 02:24:27	9086.928039
ETH	2020-04-21 21:27:43	7273.565939
DOGE	2020-08-25 19:09:10	8027.924009
ETH	2020-06-04 03:53:39	6736.423699
DOGE	2019-12-23 00:13:13	6704.032765
LTC	2021-04-01 23:13:14	3117.003666
XMR	2020-10-10 17:50:52	1989.514313
BTC	2020-10-09 13:56:39	9776.150048
DOGE	2021-03-11 18:57:57	4346.517803
BTC	2019-12-10 11:28:37	3287.686066
DOGE	2019-11-28 01:05:59	7108.628739
BTC	2020-07-16 18:26:49	971.325293
BTC	2020-04-22 03:32:11	2389.737999
LTC	2020-08-22 22:13:59	1079.398703

the expected output is:

algorithm ▲	transactions_Q1	transactions_Q2	transactions_Q3	transactions_Q4
Ethash	0.000000	14009.989638	0.000000	0.000000
RandomX	0.000000	0.000000	0.000000	1989.514313
Scrypt	9425.745465	0.000000	9107.322712	0.000000
SHA-256	0.000000	7165.438963	971.325293	9776.150048



2. Winners Chart

2. Winners Chart

There were a number of contests where participants each made multiple attempts. The attempt with the highest score is the only one considered. Write a query to list the contestants ranked in the top 3 for each contest. If multiple contestants have the same score in a contest, they are at the same rank.

Report *event_id, rank 1 name(s), rank 2 name(s), rank 3 name(s).* Order the contests by event_id. Names that share a rank should be ordered alphabetically and separated by a comma.

Order the report by *event_id.*

▾ Schema

There is one table: `scoretable`.

scoretable		
Name	Type	Description
event_id	int	id of the event.
participant_name	varchar(25)	name of the participant.
score	float	the score

▾ Sample Data Tables

scoretable		
event_id	participant_name	score
2187	Clemencia	9
2187	Clemencia	6.6
2187	Clemencia	8.6
2187	Susannah	8.8
2187	Susannah	10
2187	Susannah	8.2
2187	Sixta	7.4
3478	Hiram	7.2
3478	Hiram	8.8
3478	Treasa	9.4
3478	Treasa	6.6
3478	Treasa	6.2
3478	Pa	6.8
3478	Pa	8.6
3478	Pa	7

OUTPUT			
event_id	first	second	third
2187	Susannah	Nakita,Otto,Sixta	Clemencia
3478	Charmaine,Hiram	Karole,Treasa	Amal,Pa
4361	Pura	Elissa,Joya	Rachele

Explanation:

event_id	participant_name	best	rank
2187	Susannah	10	1
2187	Sixta	9.8	2
2187	Otto	9.8	2
2187	Nakita	9.8	2
2187	Clemencia	9	3

For *event_id* 2187, Susannah has the highest score. Sixta, Otto, and Nakita are tied for the second position. Clemencia is ranked third.

Language: MySQL | Environment | Autocomplete Ready

```mysql
/*
Enter your query below.
Please append a semicolon ";" at the end of the query
*/
-- increases the max length for group concat
SET SESSION group_concat_max_len = 1000000;
WITH
ranked_scores AS(
    SELECT
        event_id,
        participant_name,
        score,
        DENSE_RANK() OVER(
            PARTITION BY event_id
            ORDER BY score DESC
        ) AS ranking
    FROM (
        SELECT event_id, participant_name, MAX(score) AS score
        FROM scoretable
        GROUP BY event_id, participant_name
    ) t
)
SELECT
    event_id,
    GROUP_CONCAT(CASE WHEN ranking = 1 THEN participant_name END ORDER BY participant_name) AS first,
    GROUP_CONCAT(CASE WHEN ranking = 2 THEN participant_name END ORDER BY participant_name) AS second,
    GROUP_CONCAT(CASE WHEN ranking = 3 THEN participant_name END ORDER BY participant_name) AS third
FROM ranked_scores
GROUP BY event_id
ORDER BY event_id;
```

Line: 30 Col: 19

Test Results | Run Query | Submit

Language: MySQL | Environment | Autocomplete Ready

```mysql
/*
Enter your query below.
Please append a semicolon ";" at the end of the query
*/
-- increases the max length for group concat
SET SESSION group_concat_max_len = 1000000;
WITH
ranked_scores AS(
    SELECT
        event_id,
        participant_name,
        score,
        DENSE_RANK() OVER(
            PARTITION BY event_id
            ORDER BY score DESC
        ) AS ranking
    FROM (
        SELECT event_id, participant_name, MAX(score) AS score
        FROM scoretable
        GROUP BY event_id, participant_name
    ) t
)
SELECT
    event_id,
    GROUP_CONCAT(CASE WHEN ranking = 1 THEN participant_name END ORDER BY participant_name) AS first,
    GROUP_CONCAT(CASE WHEN ranking = 2 THEN participant_name END ORDER BY participant_name) AS second,
    GROUP_CONCAT(CASE WHEN ranking = 3 THEN participant_name END ORDER BY participant_name) AS third
FROM ranked_scores
GROUP BY event_id
ORDER BY event_id;
```

Line: 30 Col: 19

Test Results | Run Query | Submit

participant_name	varchar(25)	name of the participant.
score	float	the score

▾ Sample Data Tables

scoretable

event_id	participant_name	score
2187	Clemencia	9
2187	Clemencia	6.6
2187	Clemencia	8.6
2187	Susannah	8.8
2187	Susannah	10
2187	Susannah	8.2
2187	Sixta	7.4
2187	Sixta	9.8
2187	Sixta	6.2
2187	Otto	9.8
2187	Otto	7.2
2187	Otto	8.8
2187	Anita	8.4
2187	Anita	8.6
2187	Anita	6.6
2187	Arielle	6.8
2187	Arielle	8.4
2187	Arielle	7.6
2187	Nakita	6.6
4361	Pura	9.8
4361	Pura	9.4
4361	Pura	7.6
4361	Elissa	9.6
4361	Elissa	7.8
4361	Elissa	7.4
4361	Florinda	8.6
4361	Florinda	8.8
4361	Florinda	7.4
4361	Eloise	6.6
4361	Eloise	7.6
4361	Eloise	7.6
4361	Rachele	7.4
4361	Rachele	9.2
4361	Rachele	6.8
4361	Helga	6.6
4361	Helga	9
4361	Helga	7.6
4361	Joya	9.6
4361	Joya	8.6
4361	Joya	8.8
3478	Art	8.4
3478	Art	8.2
3478	Art	8.2
3478	Charmaine	9.6

Language MySQL | Environment | Autocomplete Ready

```
        FROM scoretable
        GROUP BY event_id, participant_name
    ) t
)
SELECT
    event_id,
    GROUP_CONCAT(CASE WHEN ranking = 1 THEN participant_name END ORDER BY
participant_name) AS first,
    GROUP_CONCAT(CASE WHEN ranking = 2 THEN participant_name END ORDER BY
participant_name) AS second,
    GROUP_CONCAT(CASE WHEN ranking = 3 THEN participant_name END ORDER BY
participant_name) AS third
FROM ranked_scores
GROUP BY event_id
ORDER BY event_id;
```

Line: 30 Col: 19

Test Results — Run Query | Submit

Compiled successfully. Correct answer.

Test case 0

Your Output (stdout)

```
+----------+-----------------------------------+--------------------
| event_id | first                             | second
+----------+-----------------------------------+--------------------
|     1434 | Charmaine Bevins                  | Joya Lipton
|     2626 | Christinia Padgett                | Jesusita Toland
|     2661 | Tricia Norgard                    | Ilene Schow
|     2962 | Annita Tessier,Rosalyn Corley     | Jessika Gard
|     3024 | Ashlyn Cheatam                    | Sunday Hires,Yolan
|     3536 | Otto Izquierdo                    | Arielle Renzi,Bric
|     3994 | Aurora Leedom{-truncated-}
```

Expected Output — Download

Language MySQL | Environment | Autocomplete Ready

```
        FROM scoretable
        GROUP BY event_id, participant_name
    ) t
)
SELECT
    event_id,
    GROUP_CONCAT(CASE WHEN ranking = 1 THEN participant_name END ORDER BY
participant_name) AS first,
    GROUP_CONCAT(CASE WHEN ranking = 2 THEN participant_name END ORDER BY
participant_name) AS second,
    GROUP_CONCAT(CASE WHEN ranking = 3 THEN participant_name END ORDER BY
participant_name) AS third
FROM ranked_scores
GROUP BY event_id
ORDER BY event_id;
```

Line: 30 Col: 19

Test Results — Run Query | Submit

Compiled successfully. Correct answer.

Test case 0

```
2661    Tricia Norgard  Ilene Schow     Herbert Denis,Kirstie Sto
2962    Annita Tessier,Rosalyn Corley   Jessika Gard    Brynn Lim
3024    Ashlyn Cheatam  Sunday Hires,Yolando Stours     Edmond Su
3536    Otto Izquierdo  Arielle Renzi,Brice Rolle       Clemencia
3994    Aurora Leedom,Maye Garfinkel    Malka Pair      Quintin M
5114    Hanh Kendig,Sibyl Mcconico      Deja Divito     Annmarie
5744    German Pellegrino       Eli Plowden     Mertie Frei
6051    Shaunta Barletta        Homer Haffner   Ernestine Smithwi
6084    Treasa Twersky  Frederica Mahnke         Jammie Stogsdill
7882    Athena Burtch,Nikita Knack      Hellie Blumberg Salina Si
8870    Pia Klinge      Athena Dang     Evie Fabiani
9270    Lavone Franklin Dorathy Yzaguirre       Tianna Kaba
9782    Loura Fortino,Stepanie Huckabee Lyman Bagnall   Elease Co
```

Language MySQL | Environment | Autocomplete Ready

```
        ) AS ranking
    FROM (
        SELECT event_id, participant_name, MAX(score) AS score
        FROM scoretable
        GROUP BY event_id, participant_name
    ) t
)
SELECT
    event_id,
    GROUP_CONCAT(CASE WHEN ranking = 1 THEN participant_name END ORDER BY participant_name) AS first,
    GROUP_CONCAT(CASE WHEN ranking = 2 THEN participant_name END ORDER BY participant_name) AS second,
    GROUP_CONCAT(CASE WHEN ranking = 3 THEN participant_name END ORDER BY participant_name) AS third
FROM ranked_scores
GROUP BY event_id
ORDER BY event_id;
```

Line: 13 Col: 27

PROBLEM STATEMENT

Test Results | Run Query | Submit

Compiled successfully. Correct answer.

Test case 0

Your Output (stdout)

```
+----------+-----------------------------------+-----------------------------+-----------------------------------+
| event_id | first                             | second                      | third                             |
+----------+-----------------------------------+-----------------------------+-----------------------------------+
|     1434 | Charmaine Bevins                  | Joya Lipton                 | Eloise Nodine                     |
|     2626 | Christinia Padgett                | Jesusita Toland             | Cletus Tull                       |
|     2661 | Tricia Norgard                    | Ilene Schow                 | Herbert Denis,Kirstie Stocking    |
|     2962 | Annita Tessier,Rosalyn Corley     | Jessika Gard                | Brynn Liming,Regenia Scalf        |
|     3024 | Ashlyn Cheatam                    | Sunday Hires,Yolando Stours | Edmond Sunseri,Roxanne Truman     |
|     3536 | Otto Izquierdo                    | Arielle Renzi,Brice Rolle   | Clemencia Hutsell,Sixta Hagy      |
|     3994 | Aurora Leedom{-truncated-}
```

Expected Output | Download

Language MySQL | Environment | Autocomplete Ready

```
FROM coins c
LEFT JOIN quarterly_volume qv1
    ON c.algorithm = qv1.algorithm AND qv1.year = 2020 AND qv1.quarter = 1
LEFT JOIN quarterly_volume qv2
    ON c.algorithm = qv2.algorithm AND qv2.year = 2020 AND qv2.quarter = 2
LEFT JOIN quarterly_volume qv3
    ON c.algorithm = qv3.algorithm AND qv3.year = 2020 AND qv3.quarter = 3
LEFT JOIN quarterly_volume qv4
    ON c.algorithm = qv4.algorithm AND qv4.year = 2020 AND qv4.quarter = 4
WHERE c.code <> "DOGE" -- NOT LIKE == <>
ORDER BY c.algorithm
```

Line: 36 Col: 1

Test Results | Run Query | Submit

Compiled successfully. Correct answer.

Test case 0

Your Output (stdout)

```
+-----------+-----------------+-----------------+-----------------+-----------------+
| algorithm | transactions_Q1 | trasnactions_Q2 | transactions_Q3 | transactions_Q4 |
+-----------+-----------------+-----------------+-----------------+-----------------+
| Ethash    |   151462.041951 |    72496.626882 |    95519.031415 |    70098.019852 |
| RandomX   |    57037.714681 |    31812.179180 |     9759.540424 |    26098.469680 |
| Scrypt    |   159606.943937 |    65609.588182 |   128631.878154 |    78124.251634 |
| SHA-256   |    61246.966303 |    70823.600427 |    77027.632819 |    99171.741237 |
+-----------+-----------------+-----------------+-----------------+-----------------+
```

Expected Output | Download

```
Ethash  151462.041951   72496.626882    95519.031415    70098.019852
RandomX 57037.714681    31812.179180    9759.540424     26098.469680
```