Data Analysis on NFTs

DATA ANALYSIS USING SQL – CAPSTONE PROJECT

WEEK 4 – DATA ANALYSIS USING SQL

Sanam Bodake

Description:

Over the past 18 months, an emerging technology has caught the attention of the world; the NFT. What is an NFT? They are digital assets stored on the blockchain. And over $22 billion was spent last year on purchasing NFTs. Why? People enjoyed the art, they speculated on what they might be worth in the future, and people didn't want to miss out.

The future of NFT's is unclear as much of the NFT's turned out to be scams of sorts since the field is wildly unregulated. They're also contested heavily for their impact on the environment.

Regardless of these controversies, it is clear that there is money to be made in NFT's. And one cool part about NFT's is that all of the data is recorded on the blockchain, meaning anytime something happens to an NFT, it is logged in this database.

In this project, analysis is to be done on real-world NFT data.

That data set is a sales data set of one of the most famous NFT projects, Cryptopunks. Meaning each row of the data set represents a sale of an NFT. The data includes sales from January 1st, 2018, to December 31st, 2021. The table has several columns including the buyer address, the ETH price, the price in U.S. dollars, the seller's address, the date, the time, the NFT ID, the transaction hash, and the NFT name.

What to find to proceed further-

1. How many sales occurred during this time period?
2. Return the top 5 most expensive transactions (by USD price) for this data set. Return the name, ETH price, and USD price, as well as the date.
3. Return a table with a row for each transaction with an event column, a USD price column, and a moving average of USD price that averages the last 50 transactions.
4. Return all the NFT names and their average sale price in USD. Sort descending. Name the average column as average_price.
5. Return each day of the week and the number of sales that occurred on that day of the week, as well as the average price in ETH. Order by the count of transactions in ascending order.
6. Construct a column that describes each sale and is called summary. The sentence should include who sold the NFT name, who bought the NFT, who sold the NFT, the date, and what price it was sold for in USD rounded to the nearest thousandth. Here's an example summary:
   "CryptoPunk #1139 was sold for $194000 to 0x91338ccfb8c0adb7756034a8200853 1d7713009d from 0x1593110441ab4c5f2c133f21b0743b2b43e297cb on 2022-01-14"
7. Create a view called "1919_purchases" and contains any sales where "0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685" was the buyer.
8. Create a histogram of ETH price ranges. Round to the nearest hundred value.

9. Return a unioned query that contains the highest price each NFT was bought for, and a new column called status saying “highest” with a query that has the lowest price each NFT was bought for and the status column saying “lowest”. The table should have a name column, a price column called price, and a status column. Order the result set by the name of the NFT, and the status, in ascending order.
10. What NFT sold the most each month / year combination? Also, what was the name and the price in USD? Order in chronological format.
11. Return the total volume (sum of all sales), round to the nearest hundred on a monthly basis (month/year).
12. Count how many transactions the wallet "0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685"had over this time period.
13. Create an “estimated average value calculator” that has a representative price of the collection every day based off of these criteria:
    - Exclude all daily outlier sales where the purchase price is below 10% of the daily average price.
    - Take the daily average of remaining transactions
    a) First create a query that will be used as a subquery. Select the event date, the USD price, and the average USD price for each day using a window function. Save it as a temporary table.
    b) Use the table you created in Part A to filter out rows where the USD prices is below 10% of the daily average and return a new estimated value which is just the daily average of the filtered data.
14. Give a complete list ordered by wallet profitability (whether people have made or lost money).

Results:

Using MySQL, analysis is done on the provided dataset for the NFTs. Below are the detailed results given for each of the findings mentioned in the description.

Datasets actually contains the data from June 23rd, 2017, to January 14th, 2022, instead of January 1st, 2018, to December 31st, 2021.

```
USE cryptopunk;
-- 23/06/2017 to 14/01/2022
SELECT min(event_date), max(event_date) FROM pricedata;
```

min(event_date)	max(event_date)
2017-06-23	2022-01-14

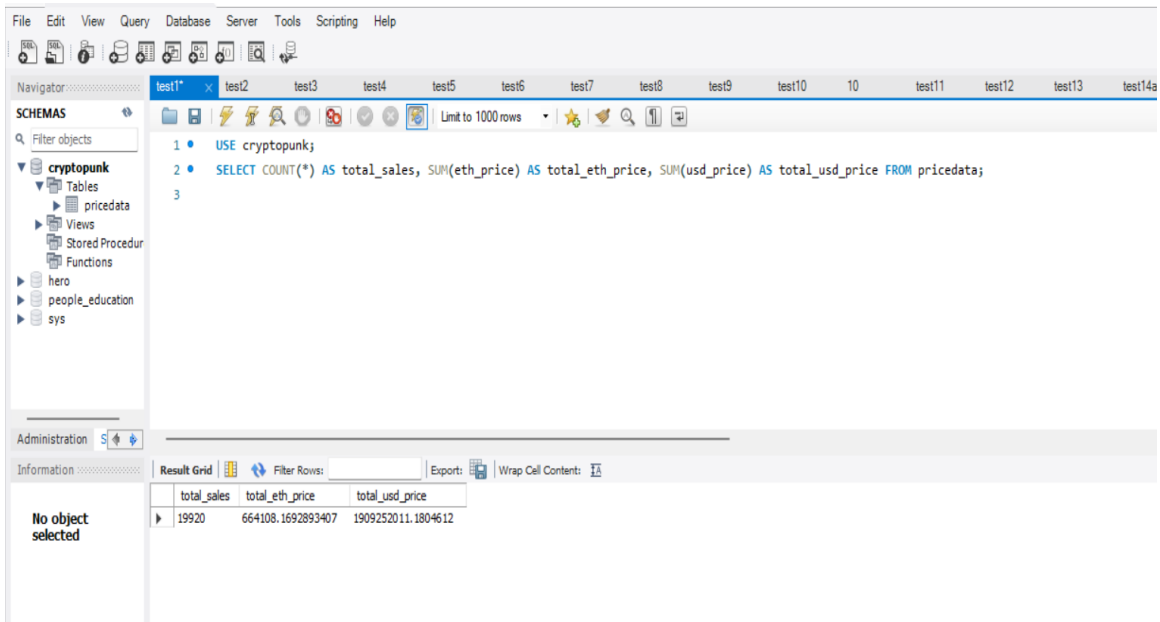
We will proceed in this analysis for this time period.(June 23rd, 2017, to January 14th, 2022,).

1) How many sales occurred during this time period?

Using the query below obtained results are shown the following screenshot.

Query-

```
USE cryptopunk;

SELECT COUNT(*) AS total_sales, SUM(eth_price) AS total_eth_price,

SUM(usd_price) AS total_usd_price FROM pricedata;
```



For the given time period total sales transactions occurred are **19920** which is of **664,108.17** eth price worth of **1,909,252,011.18** US dollars.

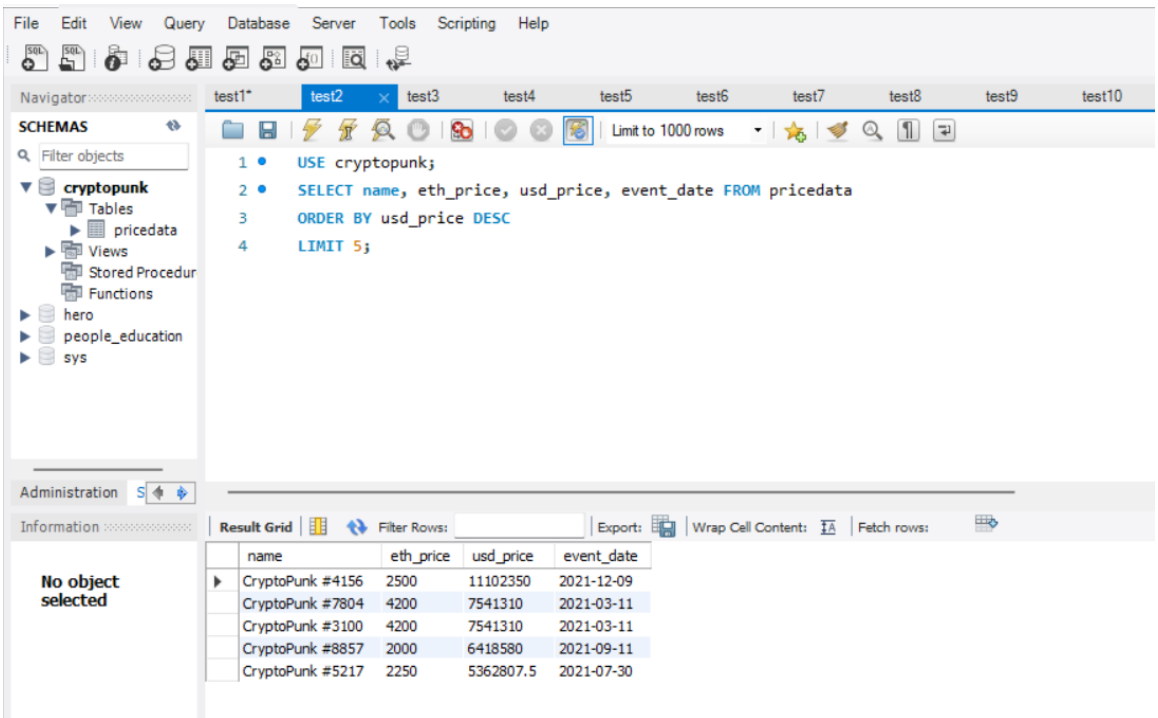
2) Return the top 5 most expensive transactions (by USD price) for this data set. Return the name, ETH price, and USD price, as well as the date.

Using queries below result is returned in the following screenshot by sorting usd_price in descending order to obtain the topmost expensive transactions.

Query-

```
USE cryptopunk;

SELECT name, eth_price, usd_price, event_date FROM pricedata

ORDER BY usd_price DESC

LIMIT 5;
```

Below are the top 5 most expensive transactions (by USD price) for this data set.

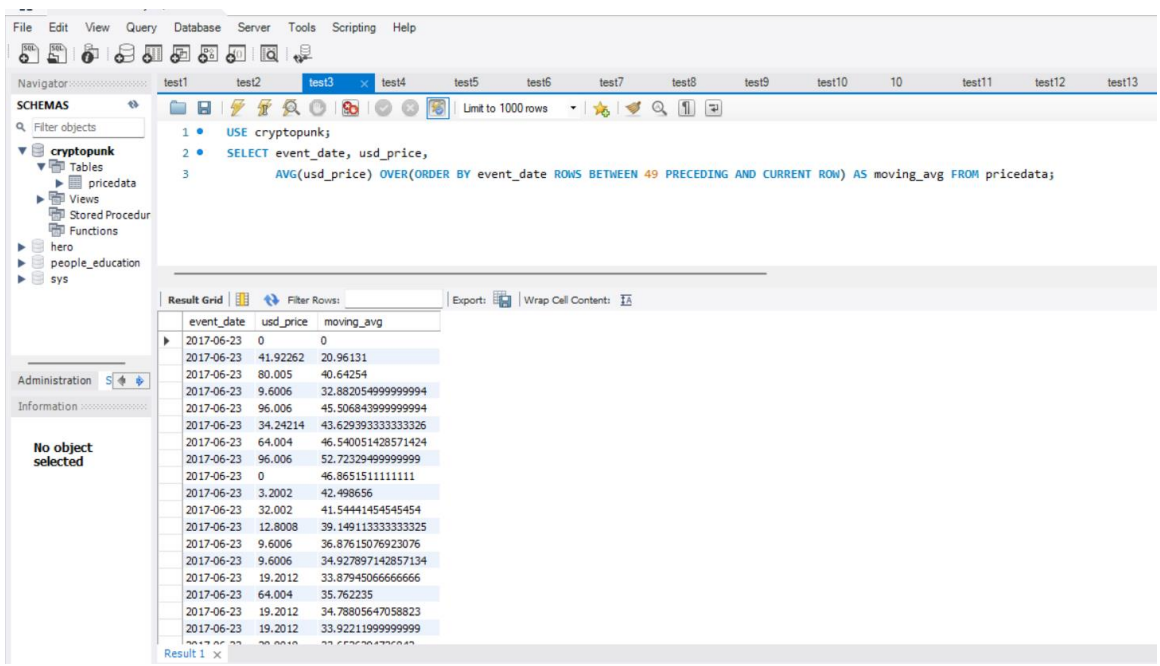


3) Return a table with a row for each transaction with an event column, a USD price column, and a moving average of USD price that averages the last 50 transactions.

Using queries below result is returned in the following screenshot by taking moving average of USD price for the last 50 transactions.

Query-

```
USE cryptopunk;

SELECT event_date, usd_price,

        AVG(usd_price) OVER(ORDER BY event_date ROWS BETWEEN 49

                                PRECEDING AND CURRENT ROW) AS moving_avg

FROM pricedata;
```



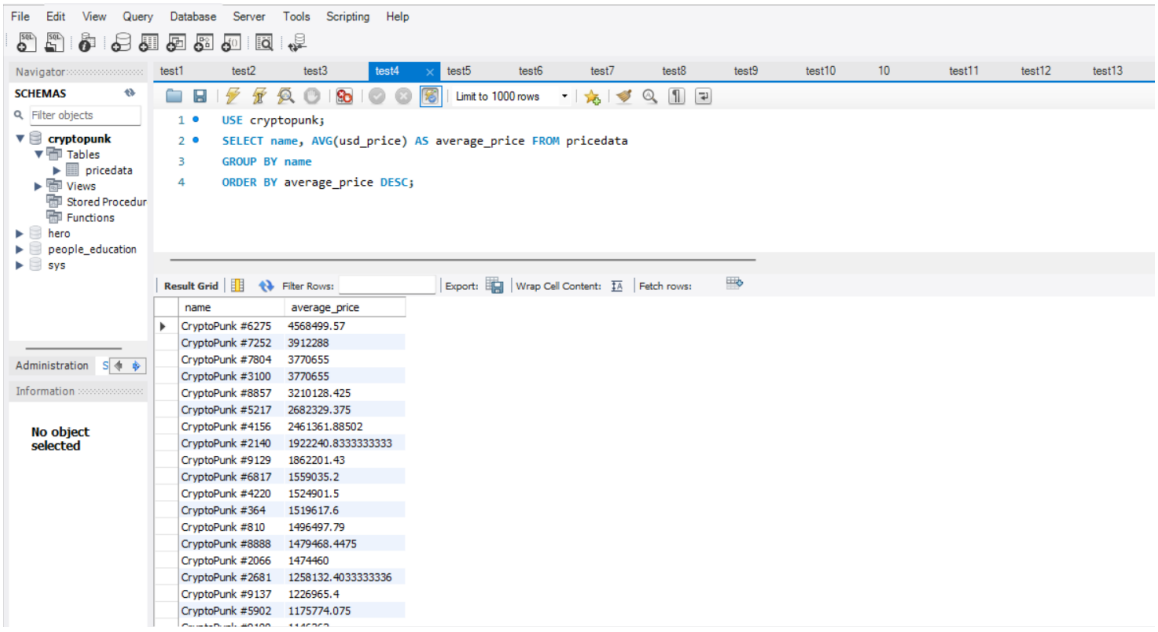
event_date	usd_price	moving_avg
2017-06-23	0	0
2017-06-23	41.92262	20.96131
2017-06-23	80.005	40.64254
2017-06-23	9.6006	32.882054999999994
2017-06-23	96.006	45.506843999999994
2017-06-23	34.24214	43.629393333333326
2017-06-23	64.004	46.540051428571424
2017-06-23	96.006	52.72329499999999
2017-06-23	0	46.8651511111111
2017-06-23	3.2002	42.498656
2017-06-23	32.002	41.54441454545454
2017-06-23	12.8008	39.149113333333325
2017-06-23	9.6006	36.87615076923076
2017-06-23	9.6006	34.927897142857134
2017-06-23	19.2012	33.87945066666666
2017-06-23	64.004	35.762235
2017-06-23	19.2012	34.78805647058823
2017-06-23	19.2012	33.92211999999999

4) Return all the NFT names and their average sale price in USD. Sort descending. Name the average column as average_price.

Query-

```
USE cryptopunk;

SELECT name, AVG(usd_price) AS average_price FROM pricedata

GROUP BY name

ORDER BY average_price DESC;
```

Using above queries obtained results are shown in the below screenshot-

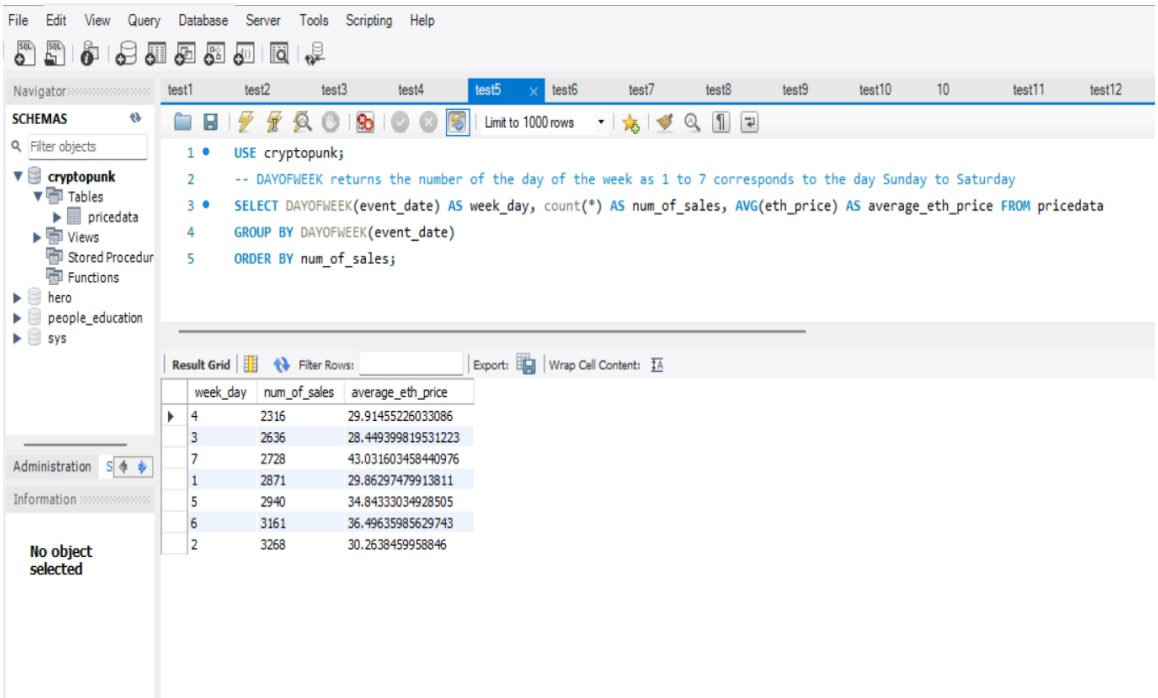


5) Return each day of the week and the number of sales that occurred on that day of the week, as well as the average price in ETH. Order by the count of transactions in ascending order.

Query-

```
USE cryptopunk;

-- DAYOFWEEK returns the number of the day of the week as 1 to 7 corresponds to the day Sunday to Saturday

SELECT DAYOFWEEK(event_date) AS week_day, count(*) AS num_of_sales,

        AVG(eth_price) AS average_eth_price FROM pricedata

GROUP BY DAYOFWEEK(event_date)

ORDER BY num_of_sales;
```

Using above queries obtained results are shown in the below screenshot-

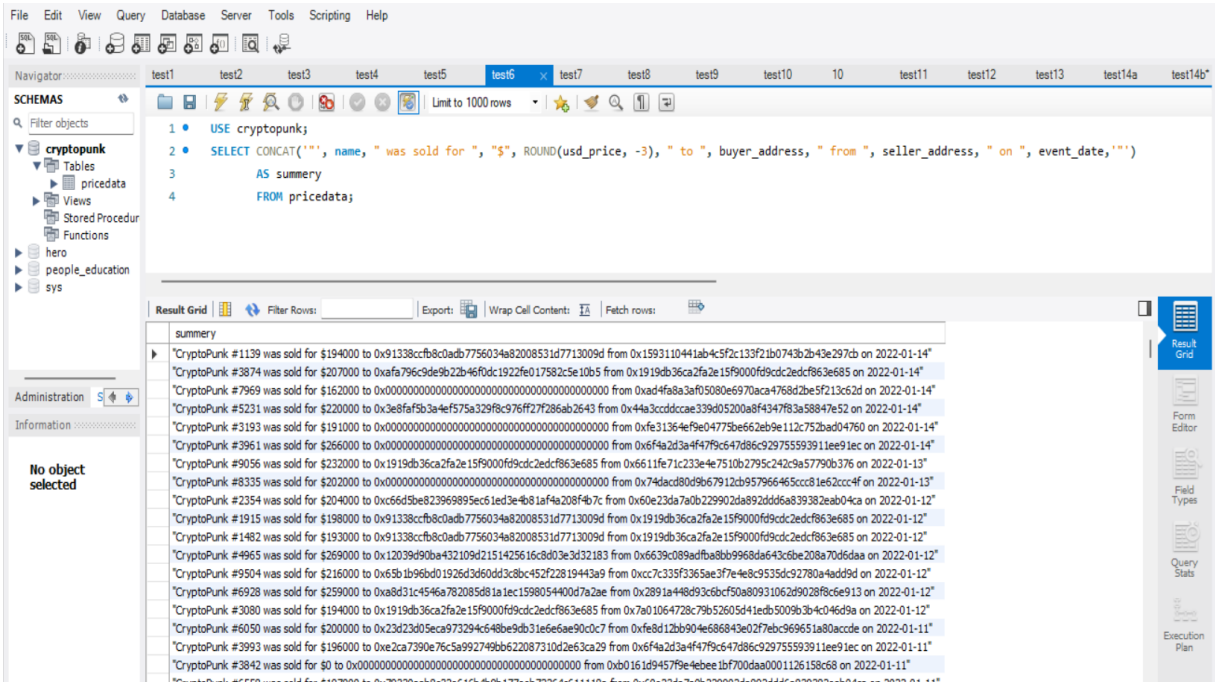


6) Construct a column that describes each sale and is called summary. The sentence should include who sold the NFT name, who bought the NFT, who sold the NFT, the date, and what price it was sold for in USD rounded to the nearest thousandth.
Here's an example summary:
"CryptoPunk #1139 was sold for $194000 to 0x91338ccfb8c0adb7756034a82008531d7713009d from 0x1593110441ab4c5f2c133f21b0743b2b43e297cb on 2022-01-14"

Query-

```
USE cryptopunk;

SELECT CONCAT('"', name, " was sold for ", "$", ROUND(usd_price, -3), " to ",

                buyer_address, " from ", seller_address, " on ", event_date,'"')

                AS summery

FROM pricedata;
```

Using above queries obtained results are shown in the below screenshot-

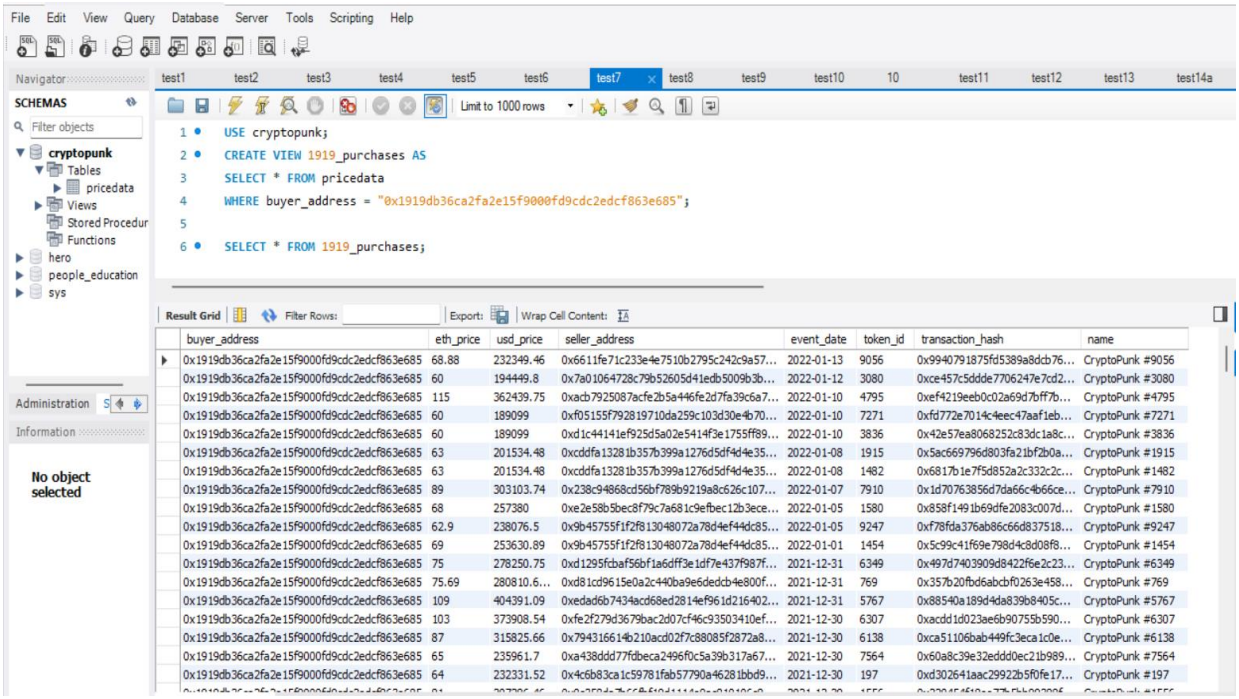


7) Create a view called “1919_purchases” and contains any sales where “0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685” was the buyer.

Query-

```
USE cryptopunk;

CREATE VIEW 1919_purchases AS

SELECT * FROM pricedata

WHERE buyer_address = "0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685";


SELECT * FROM 1919_purchases;
```

Using above queries view called “1919_purchases” has been created and using that view data is returned in the output as shown in the below screenshot-



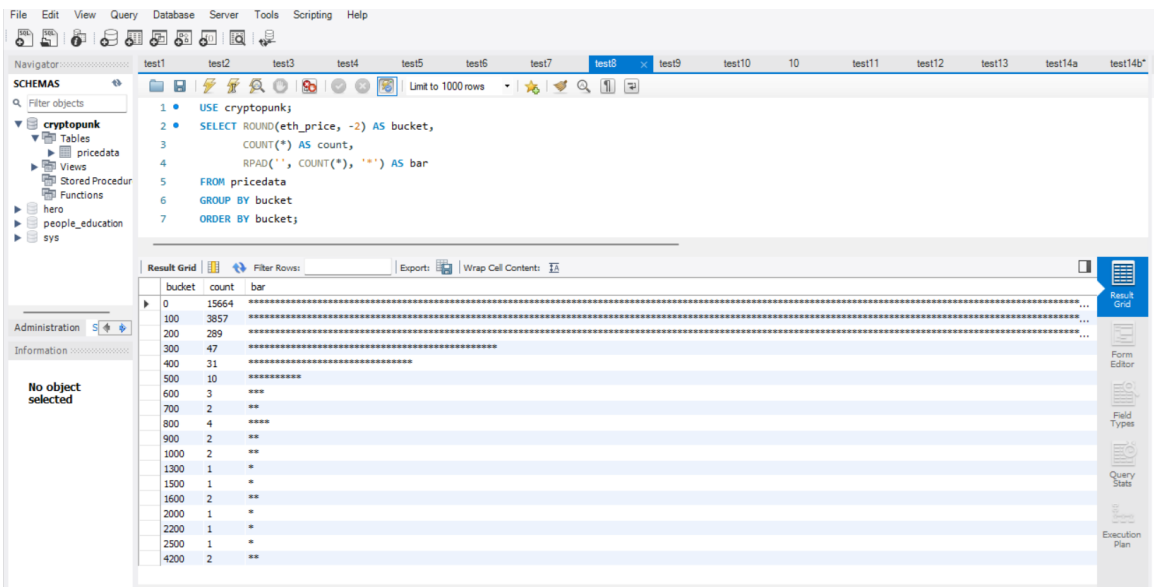
buyer_address	eth_price	usd_price	seller_address	event_date	token_id	transaction_hash	name
0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	68.88	232349.46	0x6611fe71c233e4e7510b2795c242c9a57...	2022-01-13	9056	0x9940791875fd5389a8dcb76...	CryptoPunk #9056
0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	60	194449.8	0x7a01064728c79b52605d41edb5009b3b...	2022-01-12	3080	0xce457c5ddde7706247e7cd2...	CryptoPunk #3080
0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	115	362439.75	0xacb7925087acfe2b5a446fe2d7fa39c6a7...	2022-01-10	4795	0xef4219eeb0c02a69d7bff7b...	CryptoPunk #4795
0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	60	189099	0xf05155f792819710da259c103d30e4b70...	2022-01-10	7271	0xfd772e7014c4eec47aaf1eb...	CryptoPunk #7271
0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	60	189099	0xd1c44141ef925d5a02e5414f3e1755ff89...	2022-01-10	3836	0x42e57ea8068252c83dc1a8c...	CryptoPunk #3836
0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	63	201534.48	0xcddfa13281b357b399a1276d5df4d4e35...	2022-01-08	1915	0x5ac669796d803fa21bf2b0a...	CryptoPunk #1915
0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	63	201534.48	0xcddfa13281b357b399a1276d5df4d4e35...	2022-01-08	1482	0x6817b1e7f5d852a2c332c2c...	CryptoPunk #1482
0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	89	303103.74	0x238c94868cd56bf789b9219a8c626c107...	2022-01-07	7910	0x1d70763856d7da66c4b66ce...	CryptoPunk #7910
0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	68	257380	0xe2e58b5bec8f79c7a681c9efbec12b3ece...	2022-01-05	1580	0x858f1491b69dfe2083c007d...	CryptoPunk #1580
0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	62.9	238076.5	0x9b45755f1f2f813048072a78d4ef44dc85...	2022-01-05	9247	0xf78fda376ab86c66d837518...	CryptoPunk #9247
0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	69	253630.89	0x9b45755f1f2f813048072a78d4ef44dc85...	2022-01-01	1454	0x5c99c41f69e798d4c8d08f8...	CryptoPunk #1454
0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	75	278250.75	0xd1295fcbaf56bf1a6dff3e1df7e437f987f...	2021-12-31	6349	0x497d7403909d8422f6e2c23...	CryptoPunk #6349
0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	75.69	280810.6...	0xd81cd9615e0a2c440ba9e6dedcb4e800f...	2021-12-31	769	0x357b20fbd6abcbf0263e458...	CryptoPunk #769
0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	109	404391.09	0xedad6b7434acd68ed2814ef961d216402...	2021-12-31	5767	0x88540a189d4da839b8405c...	CryptoPunk #5767
0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	103	373908.54	0xfe2f279d3679bac2d07cf46c93503410ef...	2021-12-30	6307	0xacdd1d023ae6b90755b590...	CryptoPunk #6307
0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	87	315825.66	0x794316614b210acd02f7c88085f2872a8...	2021-12-30	6138	0xca51106bab449fc3eca1c0e...	CryptoPunk #6138
0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	65	235961.7	0xa438ddd77fdbeca2496f0c5a39b317a67...	2021-12-30	7564	0x60a8c39e32eddd0ec21b989...	CryptoPunk #7564
0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	64	232331.52	0x4c6b83ca1c59781fab57790a46281bbd9...	2021-12-30	197	0xd302641aac29922b5f0fe17...	CryptoPunk #197

8) Create a histogram of ETH price ranges. Round to the nearest hundred value.

Query-

```
SELECT ROUND(eth_price, -2) AS bucket,
       COUNT(*) AS count,
       RPAD('', COUNT(*), '*') AS bar

FROM pricedata

GROUP BY bucket

ORDER BY bucket;
```

Below screenshot shows the histogram of ETH price ranges, we can see the positive skewness in the distribution (Right skewed distribution) of ETH price.

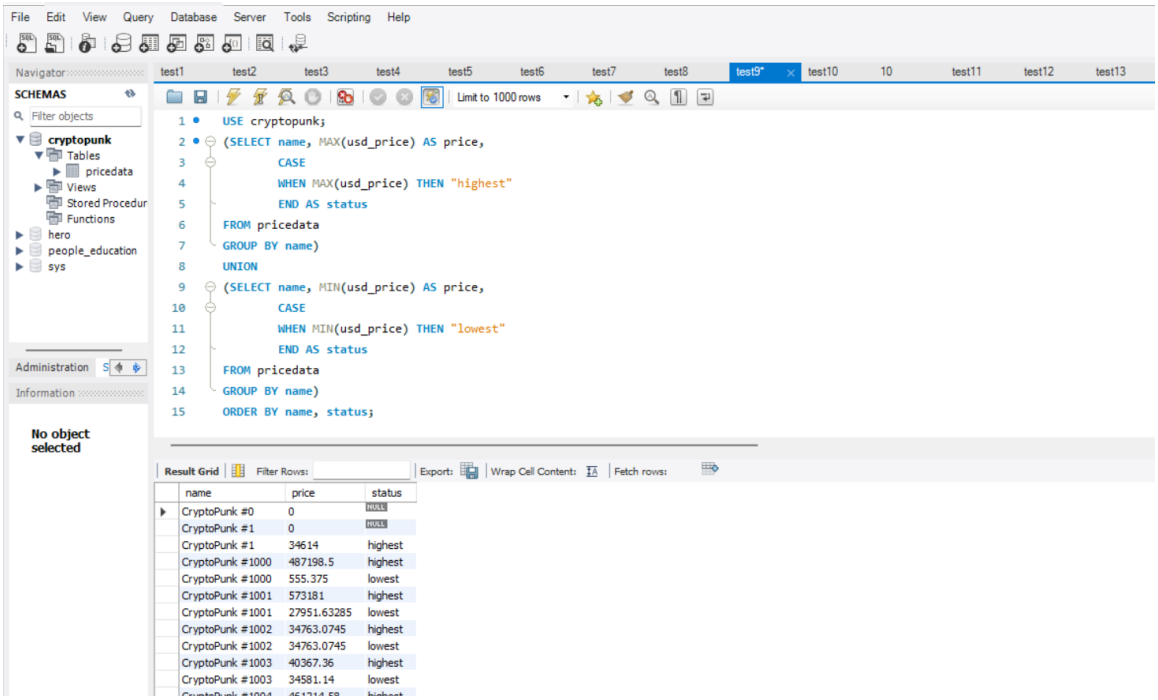


9) Return a unioned query that contains the highest price each NFT was bought for, and a new column called status saying “highest” with a query that has the lowest price each NFT was bought for and the status column saying “lowest”. The table should have a name column, a price column called price, and a status column.
Order the result set by the name of the NFT, and the status, in ascending order.

Query-

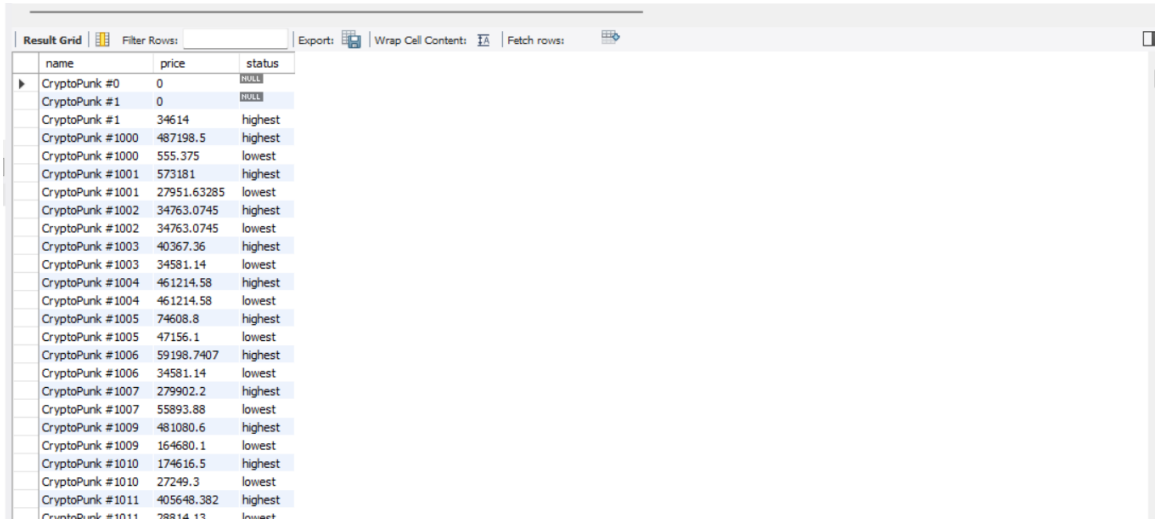
```
USE cryptopunk;

(SELECT name, MAX(usd_price) AS price,
        CASE
        WHEN MAX(usd_price) THEN "highest"
        END AS status
FROM pricedata
GROUP BY name)

UNION

(SELECT name, MIN(usd_price) AS price,
     CASE
     WHEN MIN(usd_price) THEN "lowest"
     END AS status
FROM pricedata
GROUP BY name)

ORDER BY name, status;
```

Below screenshots shows a unioned result for the highest and lowest usd_price for the each NFT ordered in the ascending manner by name and status.



```
USE cryptopunk;
(SELECT name, MAX(usd_price) AS price,
        CASE
        WHEN MAX(usd_price) THEN "highest"
        END AS status
FROM pricedata
GROUP BY name)
UNION
(SELECT name, MIN(usd_price) AS price,
        CASE
        WHEN MIN(usd_price) THEN "lowest"
        END AS status
FROM pricedata
GROUP BY name)
ORDER BY name, status;
```

name	price	status
CryptoPunk #0	0	NULL
CryptoPunk #1	0	NULL
CryptoPunk #1	34614	highest
CryptoPunk #1000	487198.5	highest
CryptoPunk #1000	555.375	lowest
CryptoPunk #1001	573181	highest
CryptoPunk #1001	27951.63285	lowest
CryptoPunk #1002	34763.0745	highest
CryptoPunk #1002	34763.0745	lowest
CryptoPunk #1003	40367.36	highest
CryptoPunk #1003	34581.14	lowest



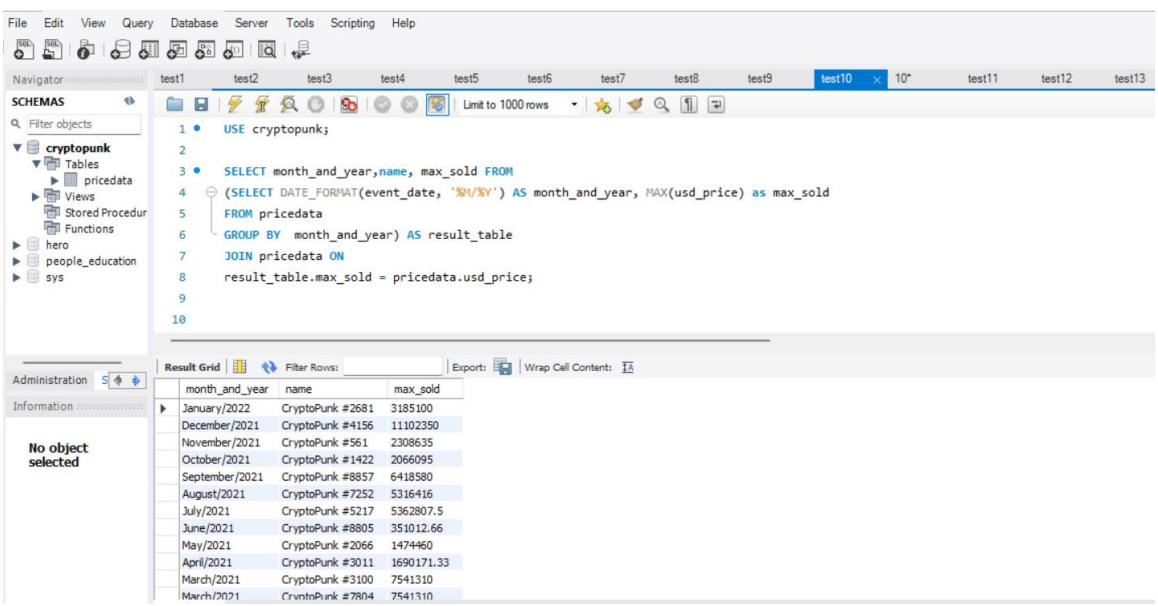
name	price	status
CryptoPunk #0	0	NULL
CryptoPunk #1	0	NULL
CryptoPunk #1	34614	highest
CryptoPunk #1000	487198.5	highest
CryptoPunk #1000	555.375	lowest
CryptoPunk #1001	573181	highest
CryptoPunk #1001	27951.63285	lowest
CryptoPunk #1002	34763.0745	highest
CryptoPunk #1002	34763.0745	lowest
CryptoPunk #1003	40367.36	highest
CryptoPunk #1003	34581.14	lowest
CryptoPunk #1004	461214.58	highest
CryptoPunk #1004	461214.58	lowest
CryptoPunk #1005	74608.8	highest
CryptoPunk #1005	47156.1	lowest
CryptoPunk #1006	59198.7407	highest
CryptoPunk #1006	34581.14	lowest
CryptoPunk #1007	279902.2	highest
CryptoPunk #1007	55893.88	lowest
CryptoPunk #1009	481080.6	highest
CryptoPunk #1009	164680.1	lowest
CryptoPunk #1010	174616.5	highest
CryptoPunk #1010	27249.3	lowest
CryptoPunk #1011	405648.382	highest

10) What NFT sold the most each month / year combination? Also, what was the name and the price in USD? Order in chronological format.

Query-

```
USE cryptopunk;

SELECT month_and_year, name, max_sold FROM

(SELECT DATE_FORMAT(event_date, '%M/%Y') AS month_and_year,

        MAX(usd_price) as max_sold

FROM pricedata

GROUP BY  month_and_year) AS result_table

JOIN pricedata ON

result_table.max_sold = pricedata.usd_price;
```

Below screenshot shows the NFT name with highest sold price for month and year combination.

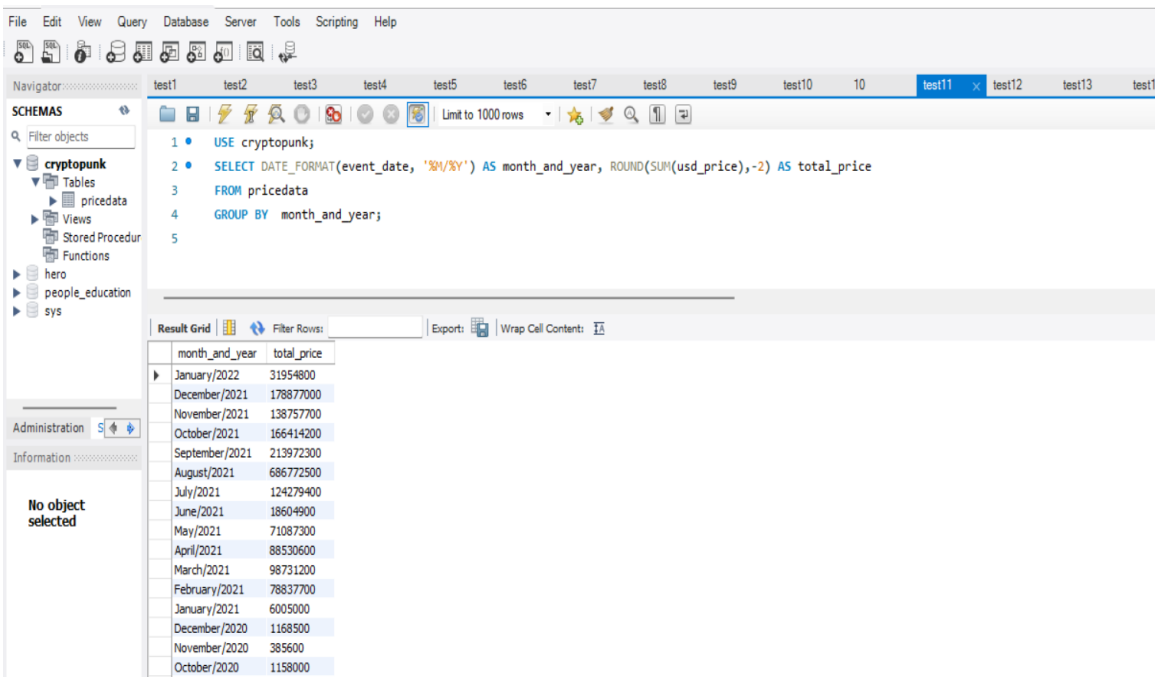


11) Return the total volume (sum of all sales), round to the nearest hundred on a monthly basis (month/year).

Query-

```
USE cryptopunk;

SELECT DATE_FORMAT(event_date, '%M/%Y') AS month_and_year,

        ROUND(SUM(usd_price),-2) AS total_price

FROM pricedata

GROUP BY  month_and_year;
```

Below are the monthly basis (month/year) total sales-

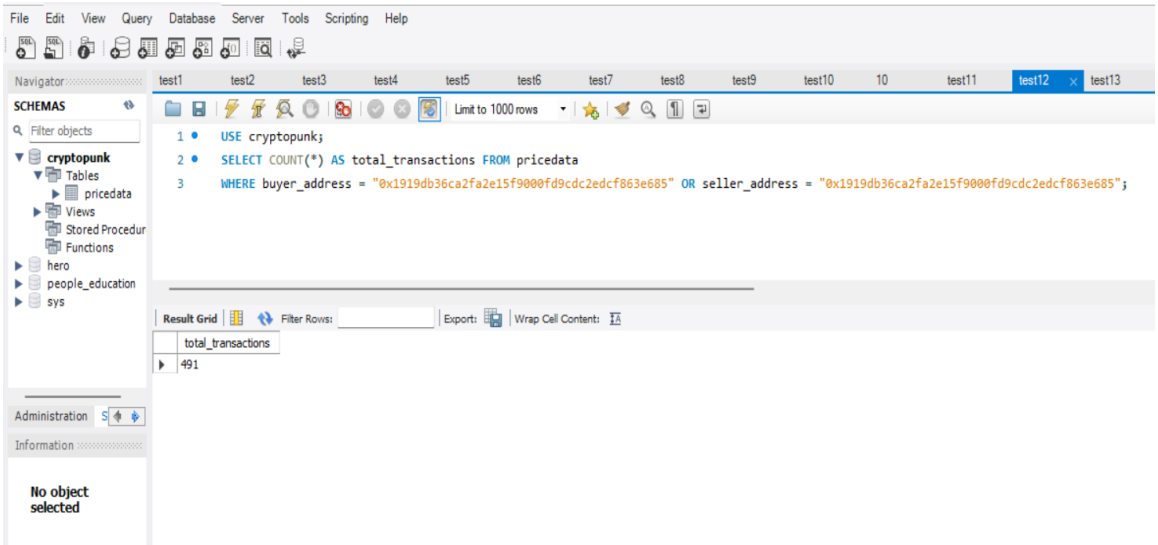


12) Count how many transactions the wallet "0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685"had over this time period.

Query-

```
USE cryptopunk;

SELECT COUNT(*) AS total_transactions FROM pricedata

WHERE buyer_address = "0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685" OR

        seller_address = "0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685";
```

Below screenshot shows the result obtained for the above queries, there are total **491** transactions the wallet "0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685"had over this time period.



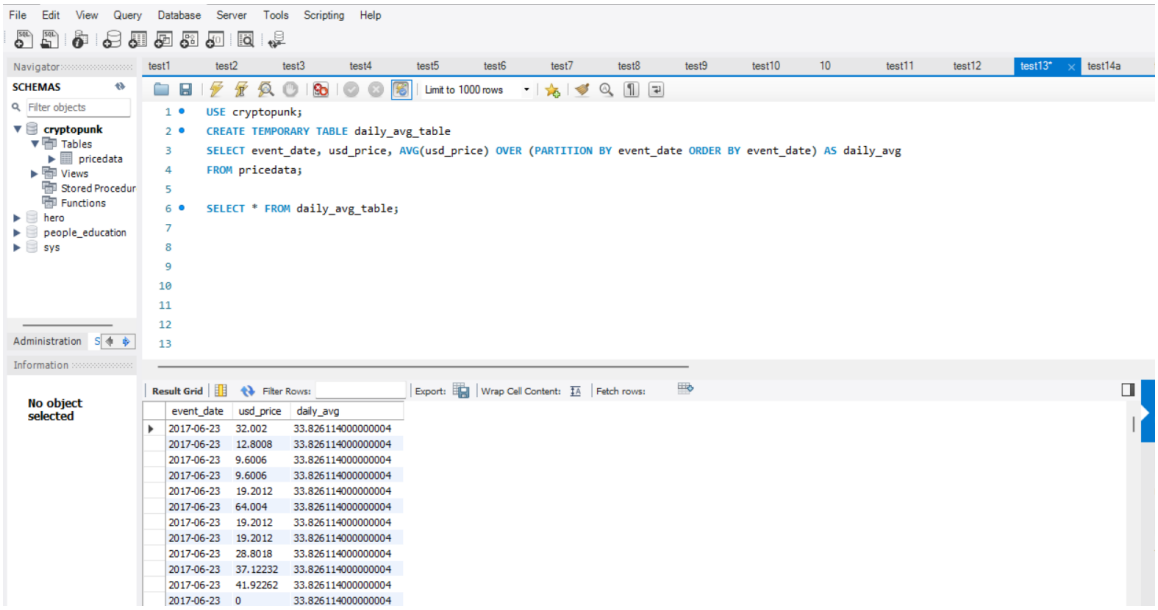
13) Create an "estimated average value calculator" that has a representative price of the collection every day based off of these criteria:
 - Exclude all daily outlier sales where the purchase price is below 10% of the daily average price
 - Take the daily average of remaining transactions
 a) First create a query that will be used as a subquery. Select the event date, the USD price, and the average USD price for each day using a window function. Save it as a temporary table.
 b) Use the table you created in Part A to filter out rows where the USD prices is below 10% of the daily average and return a new estimated value which is just the daily average of the filtered data

Query-

a)

```
USE cryptopunk;

CREATE TEMPORARY TABLE daily_avg_table

SELECT event_date, usd_price, AVG(usd_price) OVER (PARTITION BY event_date

ORDER BY event_date) AS daily_avg

FROM pricedata;

SELECT * FROM daily_avg_table;
```

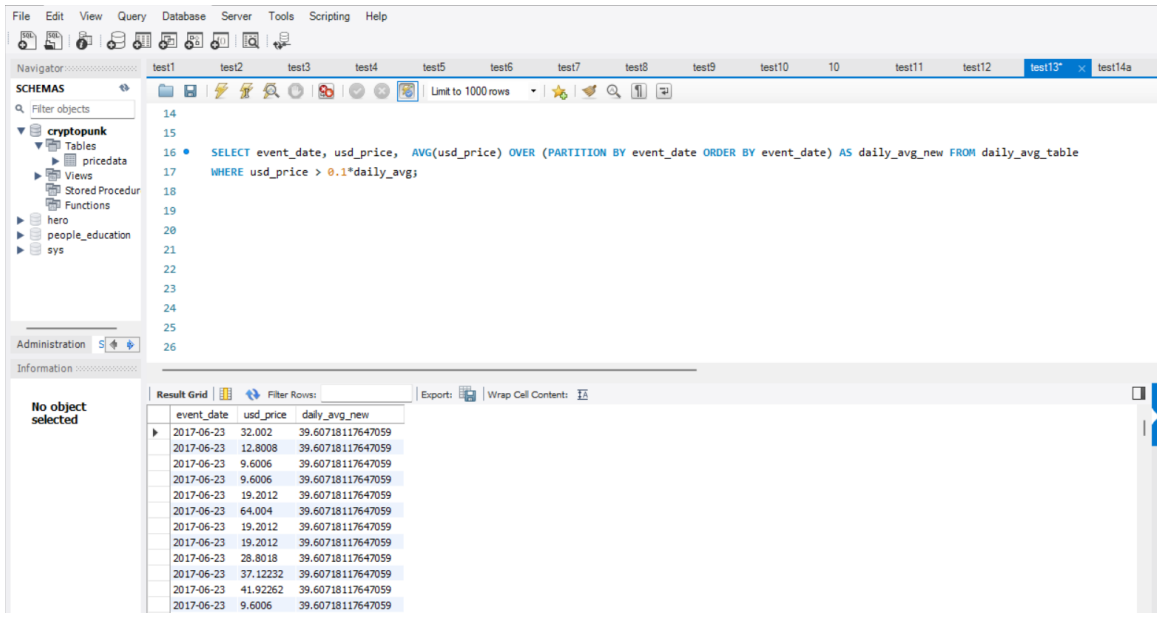
Below screenshot shows the result obtained from the temporary table daily_avg_table. Initial daily average USD price is **33.826**.



b)

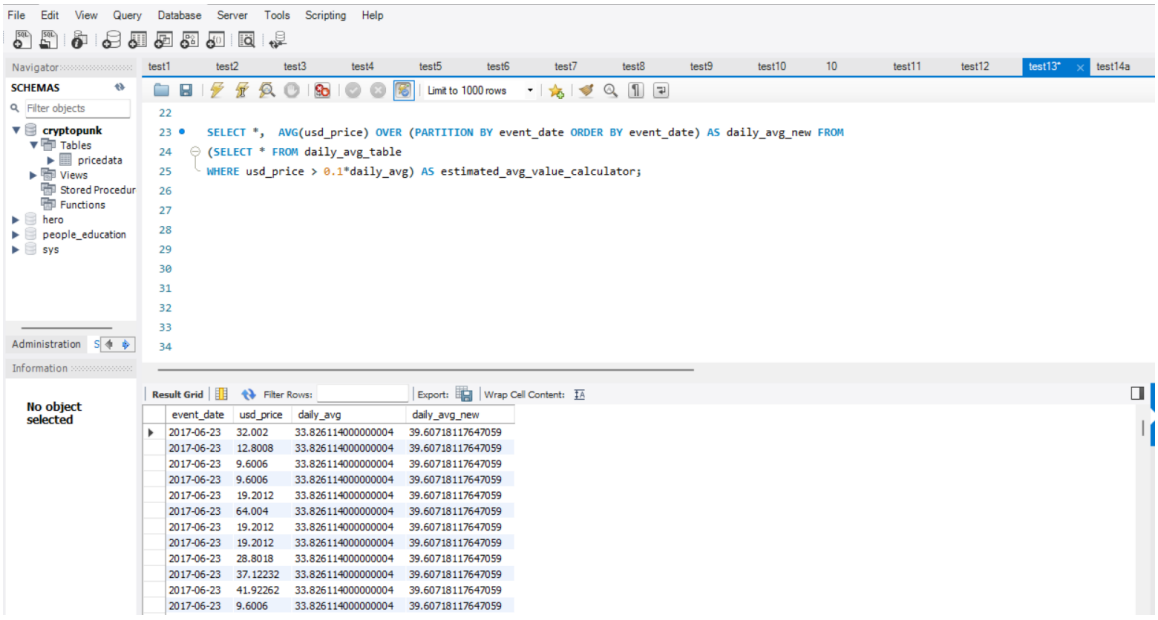
```
SELECT event_date, usd_price,  AVG(usd_price) OVER (PARTITION BY event_date
        ORDER BY event_date) AS daily_avg_new
FROM daily_avg_table
WHERE usd_price > 0.1*daily_avg;
```

Below screenshot shows new daily average for the filtered usd_price (excluded usd_price which is below the 10% of daily average). New Daily average USD price is now **39.607.**



Combined results are obtained using queries below and are shown in the following screenshot-

```
SELECT *,  AVG(usd_price) OVER (PARTITION BY event_date ORDER BY

event_date) AS daily_avg_new FROM

(SELECT * FROM daily_avg_table

WHERE usd_price > 0.1*daily_avg) AS estimated_avg_value_calculator;
```

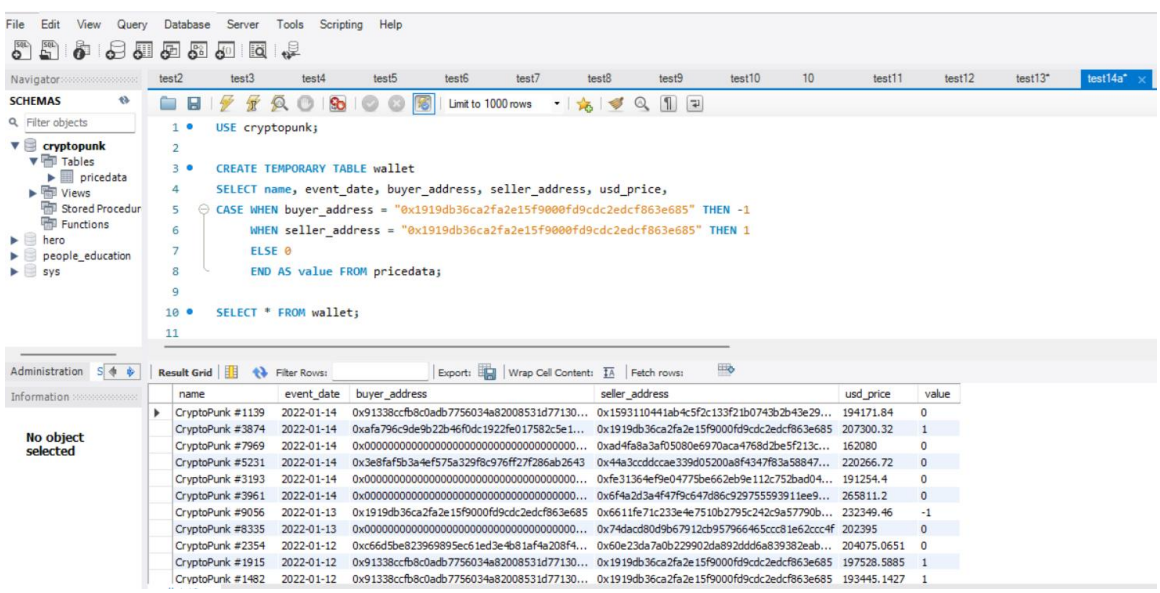


14) Give a complete list ordered by wallet profitability (whether people have made or lost money).

Query-

```
USE cryptopunk;

CREATE TEMPORARY TABLE wallet

SELECT name, event_date, buyer_address, seller_address, usd_price,

CASE

WHEN buyer_address = "0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685" THEN -1

WHEN seller_address = "0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685" THEN 1

ELSE 0

END AS value FROM pricedata;

SELECT * FROM wallet;
```

A temporary table called 'wallet' created for "0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685" for saving the results with additional column 'value' consisting of values for the wallet if they had bought or sold the NFTs as shown below-



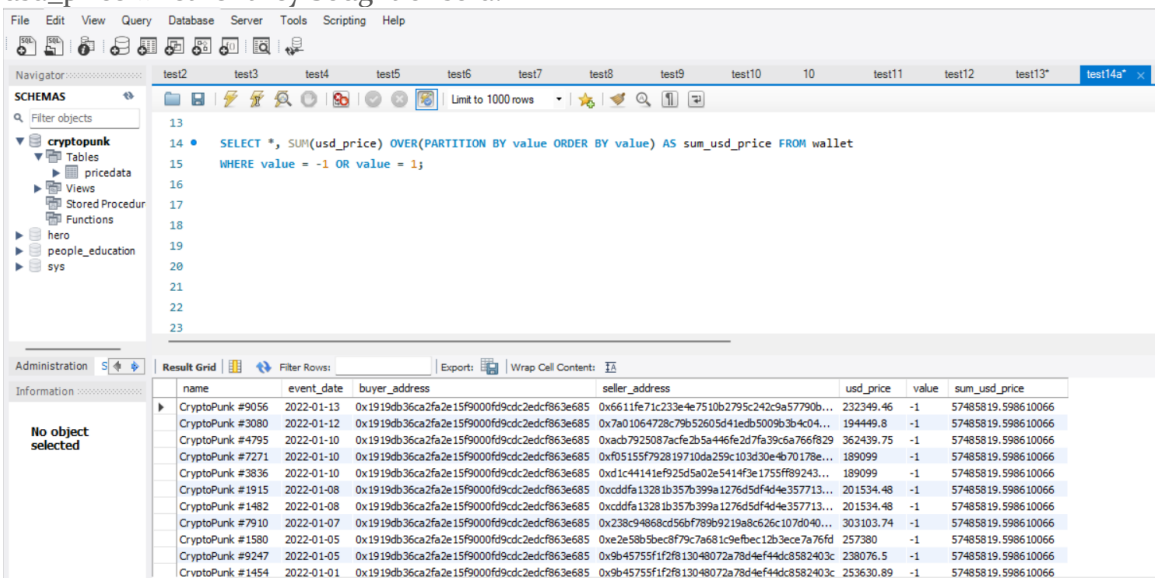
name	event_date	buyer_address	seller_address	usd_price	value
CryptoPunk #1139	2022-01-14	0x91338ccfb8c0adb7756034a82008531d77130...	0x1593110441ab4c5f2c133f21b0743b2b43e29...	194171.84	0
CryptoPunk #3874	2022-01-14	0xafa796c9de9b22b46f0dc1922fe017582c5e1...	0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	207300.32	1
CryptoPunk #7969	2022-01-14	0x0000000000000000000000000000000000...	0xad4fa8a3af05080e6970aca4768d2be5f213c...	162080	0
CryptoPunk #5231	2022-01-14	0x3e8faf5b3a4ef575a329f8c976ff27f286ab2643	0x44a3ccddccae339d05200a8f4347f83a58847...	220266.72	0
CryptoPunk #3193	2022-01-14	0x0000000000000000000000000000000000...	0xfe31364ef9e04775be662eb9e112c752bad04...	191254.4	0
CryptoPunk #3961	2022-01-14	0x0000000000000000000000000000000000...	0x6f4a2d3a4f47f9c647d86c929755593911ee9...	265811.2	0
CryptoPunk #9056	2022-01-13	0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	0x6611fe71c233e4e7510b2795c242c9a57790b...	232349.46	-1
CryptoPunk #8335	2022-01-13	0x0000000000000000000000000000000000...	0x74dacd80d9b67912cb957966465ccc81e62ccc4f	202395	0
CryptoPunk #2354	2022-01-12	0xc66d5be823969895ec61ed3e4b81af4a208f4...	0x60e23da7a0b229902da892ddd6a839382eab...	204075.0651	0
CryptoPunk #1915	2022-01-12	0x91338ccfb8c0adb7756034a82008531d77130...	0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	197528.5885	1
CryptoPunk #1482	2022-01-12	0x91338ccfb8c0adb7756034a82008531d77130...	0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	193445.1427	1

Using window function we can create a list for the wallet "0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685" in transaction and can calculate sum of USD price for it as shown below-

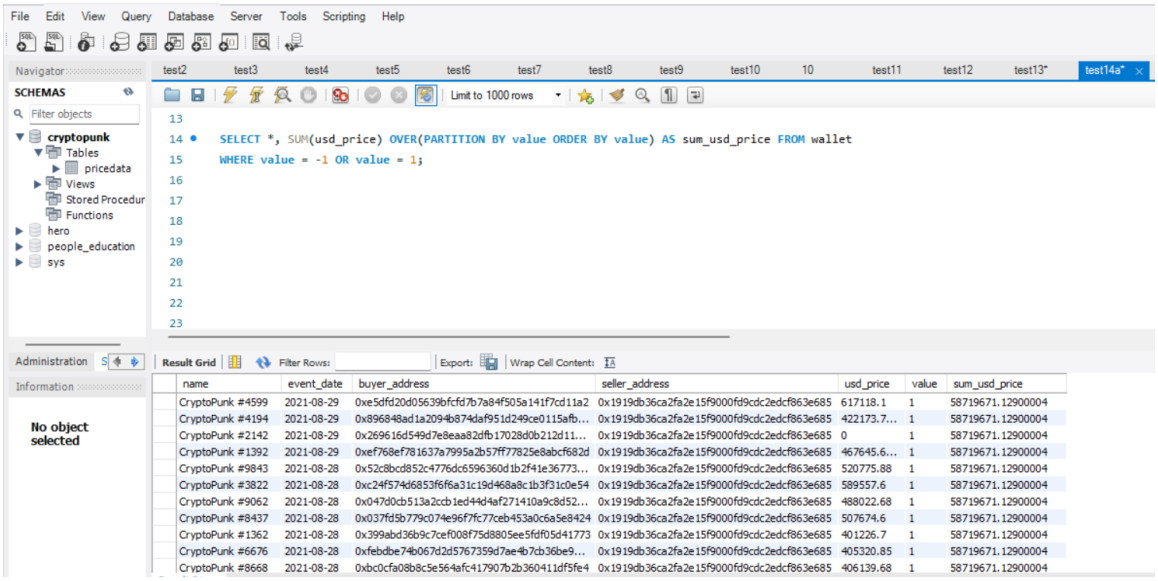
Query-

```
SELECT *, SUM(usd_price) OVER(PARTITION BY value ORDER BY value) AS

sum_usd_price FROM wallet

WHERE value = -1 OR value = 1;
```

List for the transaction "0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685" with total usd_price whether they bought or sold.



name	event_date	buyer_address	seller_address	usd_price	value	sum_usd_price
CryptoPunk #9056	2022-01-13	0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	0x6611fe71c233e4e7510b2795c242c9a57790b...	232349.46	-1	57485819.598610066
CryptoPunk #3080	2022-01-12	0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	0x7a01064728c79b52605d41edb5009b3b4c04...	194449.8	-1	57485819.598610066
CryptoPunk #4795	2022-01-10	0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	0xacb7925087acfe2b5a446fe2d7fa39c6a766f829	362439.75	-1	57485819.598610066
CryptoPunk #7271	2022-01-10	0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	0xf05155f792819710da259c103d30e4b70178e...	189099	-1	57485819.598610066
CryptoPunk #3836	2022-01-10	0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	0xd1c44141ef925d5a02e5414f3e1755ff89243...	189099	-1	57485819.598610066
CryptoPunk #1915	2022-01-08	0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	0xcddfa13281b357b399a1276d5df4d4e357713...	201534.48	-1	57485819.598610066
CryptoPunk #1482	2022-01-08	0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	0xcddfa13281b357b399a1276d5df4d4e357713...	201534.48	-1	57485819.598610066
CryptoPunk #7910	2022-01-07	0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	0x238c94868cd56bf789b9219a8c626c107d040...	303103.74	-1	57485819.598610066
CryptoPunk #1580	2022-01-05	0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	0xe2e58b5bec8f79c7a681c9efbec12b3ece7a76fd	257380	-1	57485819.598610066
CryptoPunk #9247	2022-01-05	0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	0x9b45755f1f2f813048072a78d4ef44dc8582403c	238076.5	-1	57485819.598610066
CryptoPunk #1454	2022-01-01	0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	0x9b45755f1f2f813048072a78d4ef44dc8582403c	253630.89	-1	57485819.598610066

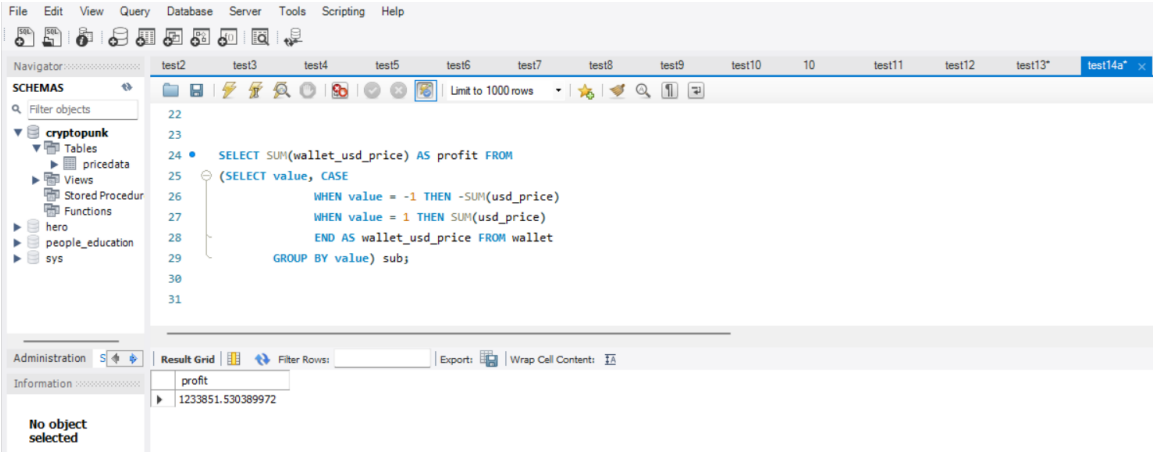


name	event_date	buyer_address	seller_address	usd_price	value	sum_usd_price
CryptoPunk #4599	2021-08-29	0xe5dfd20d05639bfcfd7b7a84f505a141f7cd11a2	0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	617118.1	1	58719671.12900004
CryptoPunk #4194	2021-08-29	0x896848ad1a2094b874daf951d249ce0115afb...	0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	422173.7...	1	58719671.12900004
CryptoPunk #2142	2021-08-29	0x269616d549d7e8eaa82dfb17028d0b212d11...	0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	0	1	58719671.12900004
CryptoPunk #1392	2021-08-29	0xef768ef781637a7995a2b57ff77825e8abcf682d	0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	467645.6...	1	58719671.12900004
CryptoPunk #9843	2021-08-28	0x52c8bcd852c4776dc6596360d1b2f41e36773...	0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	520775.88	1	58719671.12900004
CryptoPunk #3822	2021-08-28	0xc24f574d6853f6f6a31c19d468a8c1b3f31c0e54	0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	589557.6	1	58719671.12900004
CryptoPunk #9062	2021-08-28	0x047d0cb513a2ccb1ed44d4af271410a9c8d52...	0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	488022.68	1	58719671.12900004
CryptoPunk #8437	2021-08-28	0x037fd5b779c074e96f7fc77ceb453a0c6a5e8424	0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	507674.6	1	58719671.12900004
CryptoPunk #1362	2021-08-28	0x399abd36b9c7cef008f75d8805ee5fdf05d41773	0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	401226.7	1	58719671.12900004
CryptoPunk #6676	2021-08-28	0xfebdbe74b067d2d5767359d7ae4b7cb36be9...	0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	405320.85	1	58719671.12900004
CryptoPunk #8668	2021-08-28	0xbc0cfa08b8c5e564afc417907b2b360411df5fe4	0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	406139.68	1	58719671.12900004

Using the below queries profitability is calculated for this wallet as shown in the following screenshot-

Query-

```
SELECT SUM(wallet_usd_price) AS profit FROM

(SELECT value, CASE
        WHEN value = -1 THEN -SUM(usd_price)
        WHEN value = 1 THEN SUM(usd_price)
        END AS wallet_usd_price FROM wallet
GROUP BY value) sub;
```



In this wallet for the transaction "0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685" the total **1233851.53** USD price has been made.