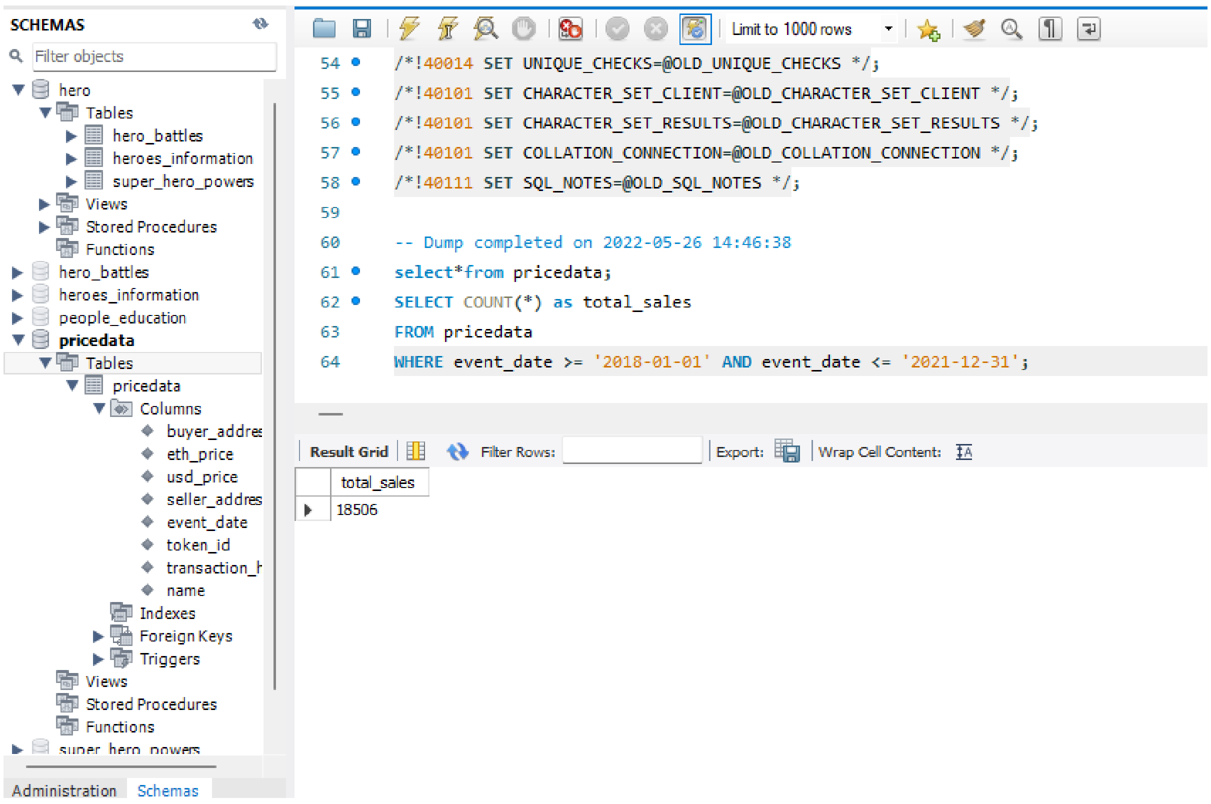
CAPSTONE PROJECT MYSQL

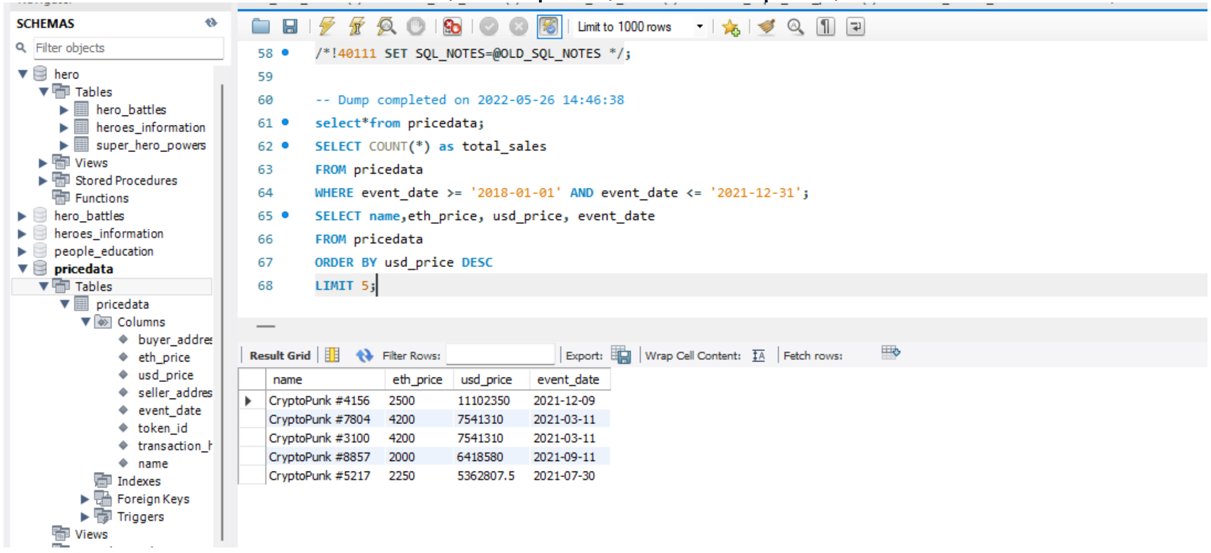
1How many sales occurred during this time period?



```
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2022-05-26 14:46:38
select*from pricedata;
SELECT COUNT(*) as total_sales
FROM pricedata
WHERE event_date >= '2018-01-01' AND event_date <= '2021-12-31';
```

total_sales
18506

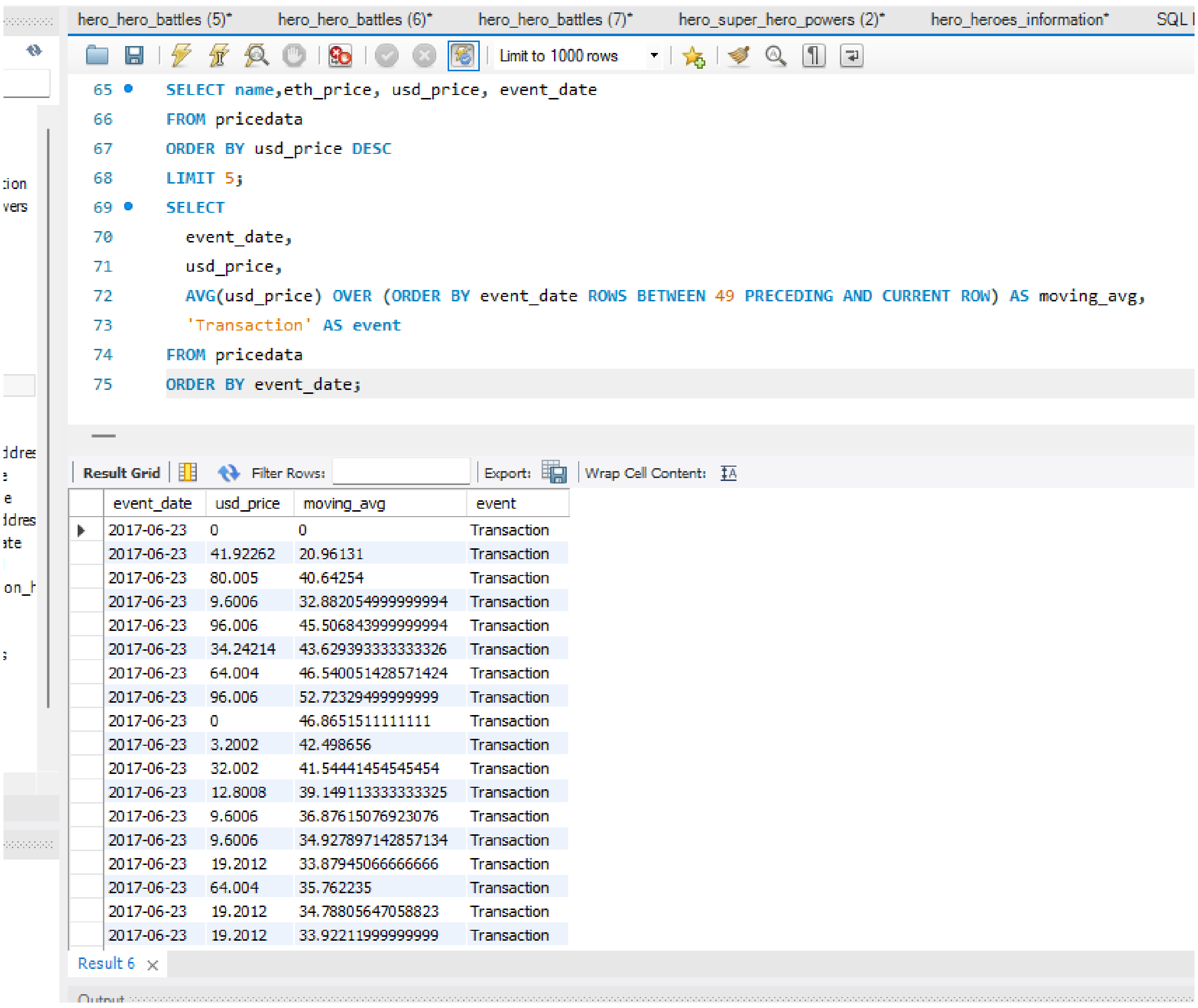
2 Return the top 5 most expensive transactions (by USD price) for this data set. Return the name, ETH price, and USD price, as well as the date.



```
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2022-05-26 14:46:38
select*from pricedata;
SELECT COUNT(*) as total_sales
FROM pricedata
WHERE event_date >= '2018-01-01' AND event_date <= '2021-12-31';
SELECT name,eth_price, usd_price, event_date
FROM pricedata
ORDER BY usd_price DESC
LIMIT 5;
```

name	eth_price	usd_price	event_date
CryptoPunk #4156	2500	11102350	2021-12-09
CryptoPunk #7804	4200	7541310	2021-03-11
CryptoPunk #3100	4200	7541310	2021-03-11
CryptoPunk #8857	2000	6418580	2021-09-11
CryptoPunk #5217	2250	5362807.5	2021-07-30

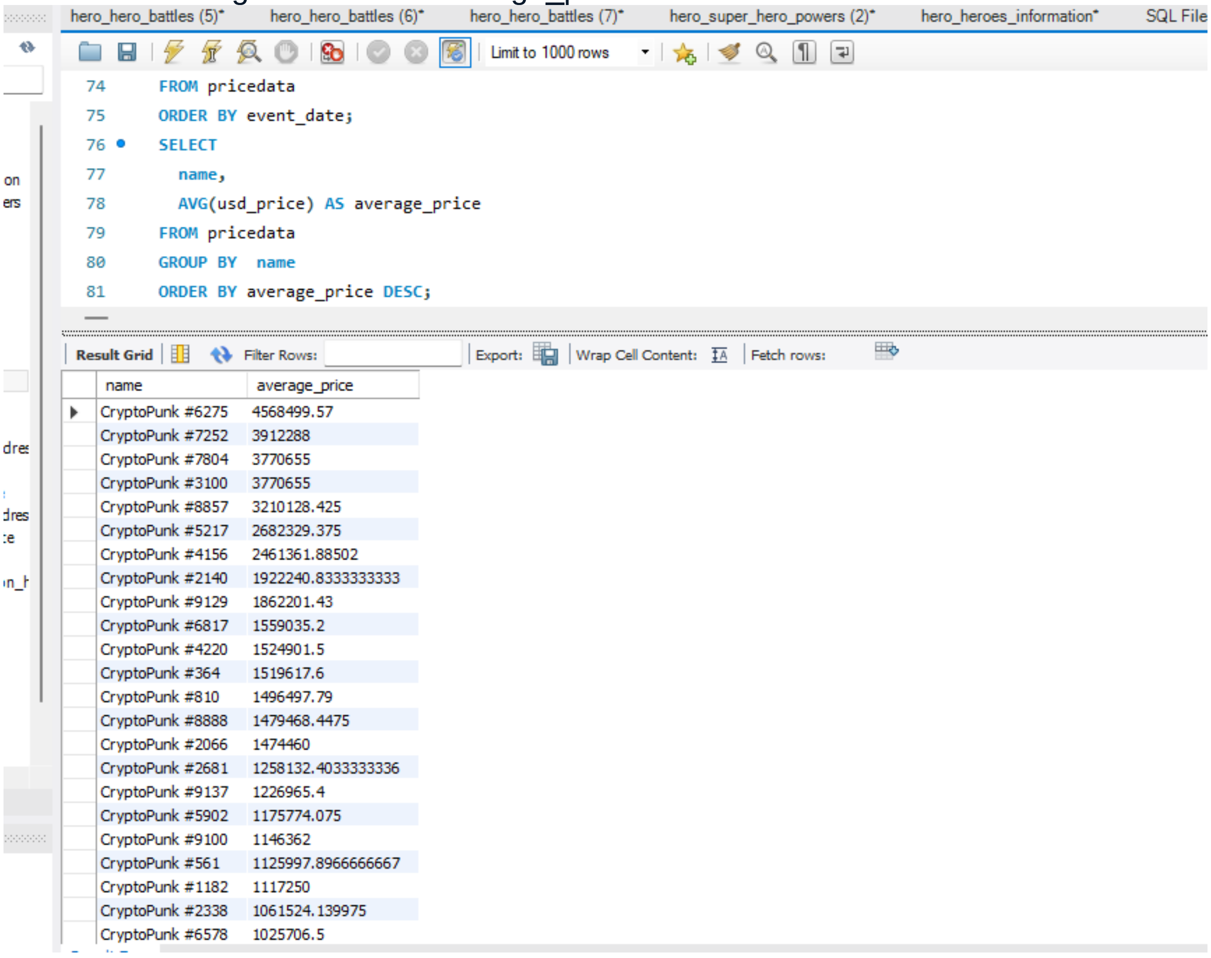
3 Return a table with a row for each transaction with an event column, a USD price column, and a moving average of USD price that averages the last 50 transactions.



```sql
SELECT name,eth_price, usd_price, event_date
FROM pricedata
ORDER BY usd_price DESC
LIMIT 5;
SELECT
  event_date,
  usd_price,
  AVG(usd_price) OVER (ORDER BY event_date ROWS BETWEEN 49 PRECEDING AND CURRENT ROW) AS moving_avg,
  'Transaction' AS event
FROM pricedata
ORDER BY event_date;
```

event_date	usd_price	moving_avg	event
2017-06-23	0	0	Transaction
2017-06-23	41.92262	20.96131	Transaction
2017-06-23	80.005	40.64254	Transaction
2017-06-23	9.6006	32.882054999999994	Transaction
2017-06-23	96.006	45.506843999999994	Transaction
2017-06-23	34.24214	43.629393333333326	Transaction
2017-06-23	64.004	46.540051428571424	Transaction
2017-06-23	96.006	52.72329499999999	Transaction
2017-06-23	0	46.86515111111111	Transaction
2017-06-23	3.2002	42.498656	Transaction
2017-06-23	32.002	41.54441454545454	Transaction
2017-06-23	12.8008	39.149113333333325	Transaction
2017-06-23	9.6006	36.87615076923076	Transaction
2017-06-23	9.6006	34.927897142857134	Transaction
2017-06-23	19.2012	33.87945066666666	Transaction
2017-06-23	64.004	35.762235	Transaction
2017-06-23	19.2012	34.78805647058823	Transaction
2017-06-23	19.2012	33.92211999999999	Transaction

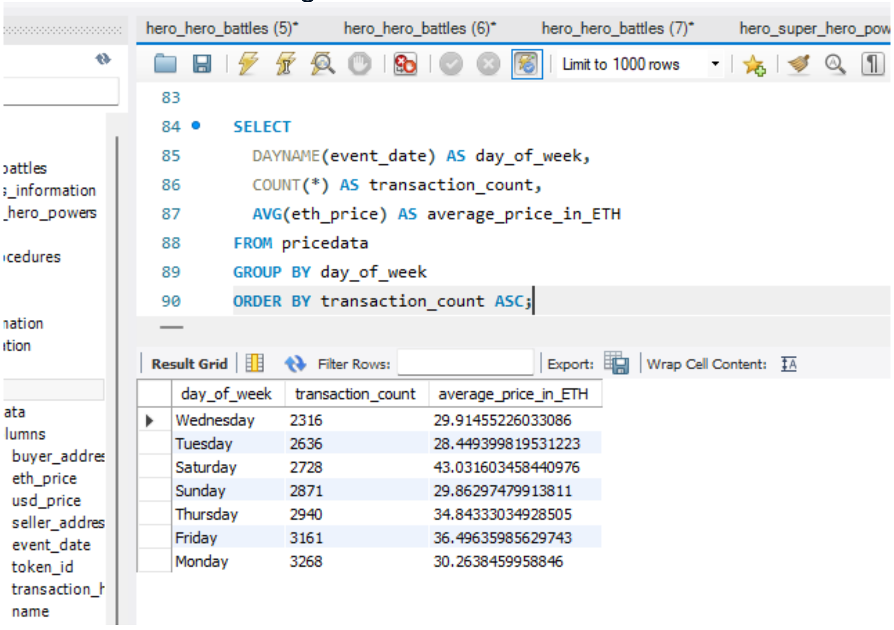
4 Return all the NFT names and their average sale price in USD. Sort descending. Name the average column as average_price.



```sql
FROM pricedata
ORDER BY event_date;
SELECT
  name,
  AVG(usd_price) AS average_price
FROM pricedata
GROUP BY  name
ORDER BY average_price DESC;
```

name	average_price
CryptoPunk #6275	4568499.57
CryptoPunk #7252	3912288
CryptoPunk #7804	3770655
CryptoPunk #3100	3770655
CryptoPunk #8857	3210128.425
CryptoPunk #5217	2682329.375
CryptoPunk #4156	2461361.88502
CryptoPunk #2140	1922240.8333333333
CryptoPunk #9129	1862201.43
CryptoPunk #6817	1559035.2
CryptoPunk #4220	1524901.5
CryptoPunk #364	1519617.6
CryptoPunk #810	1496497.79
CryptoPunk #8888	1479468.4475
CryptoPunk #2066	1474460
CryptoPunk #2681	1258132.4033333336
CryptoPunk #9137	1226965.4
CryptoPunk #5902	1175774.075
CryptoPunk #9100	1146362
CryptoPunk #561	1125997.8966666667
CryptoPunk #1182	1117250
CryptoPunk #2338	1061524.139975
CryptoPunk #6578	1025706.5

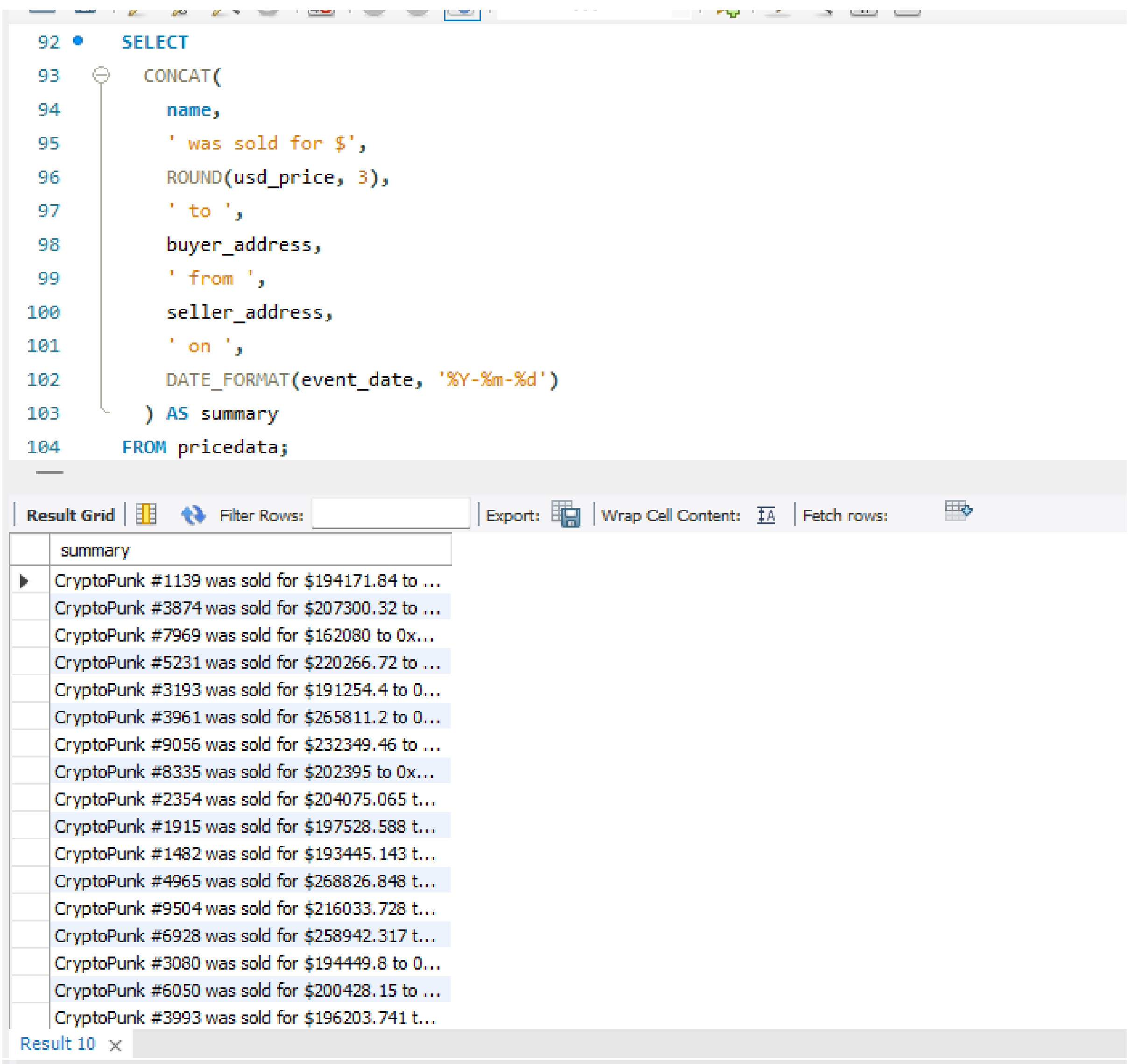
5 Return each day of the week and the number of sales that occurred on that day of the week, as well as the average price in ETH. Order by the count of transactions in ascending order.



```sql

SELECT
  DAYNAME(event_date) AS day_of_week,
  COUNT(*) AS transaction_count,
  AVG(eth_price) AS average_price_in_ETH
FROM pricedata
GROUP BY day_of_week
ORDER BY transaction_count ASC;
```

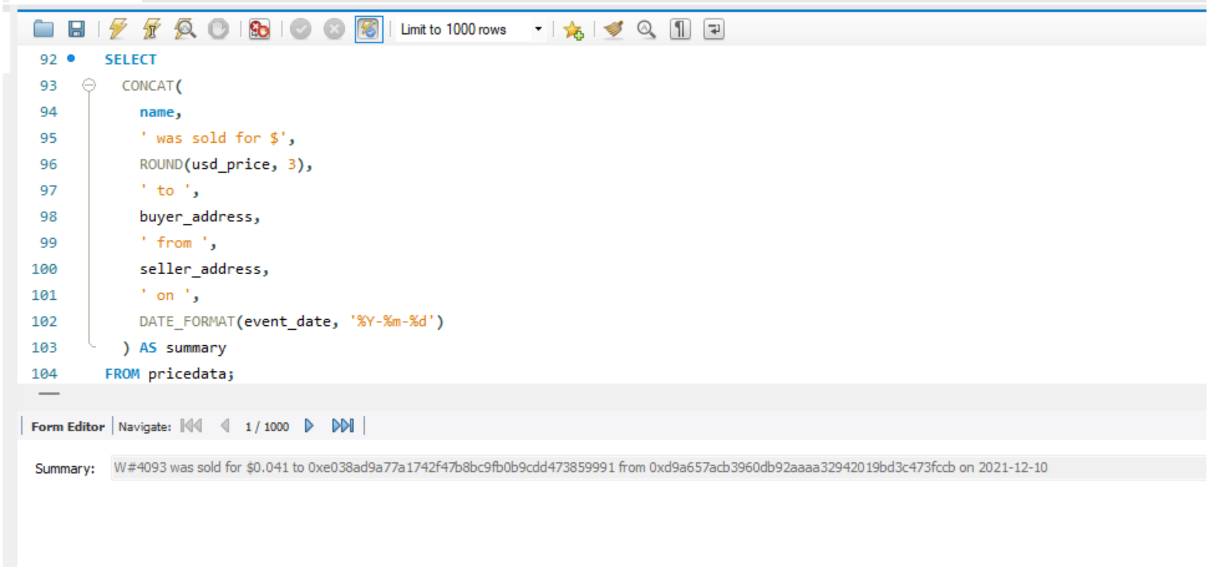
day_of_week	transaction_count	average_price_in_ETH
Wednesday	2316	29.91455226033086
Tuesday	2636	28.449399819531223
Saturday	2728	43.031603458440976
Sunday	2871	29.86297479913811
Thursday	2940	34.84333034928505
Friday	3161	36.49635985629743
Monday	3268	30.2638459958846

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”



```sql
SELECT
  CONCAT(
    name,
    ' was sold for $',
    ROUND(usd_price, 3),
    ' to ',
    buyer_address,
    ' from ',
    seller_address,
    ' on ',
    DATE_FORMAT(event_date, '%Y-%m-%d')
  ) AS summary
FROM pricedata;
```

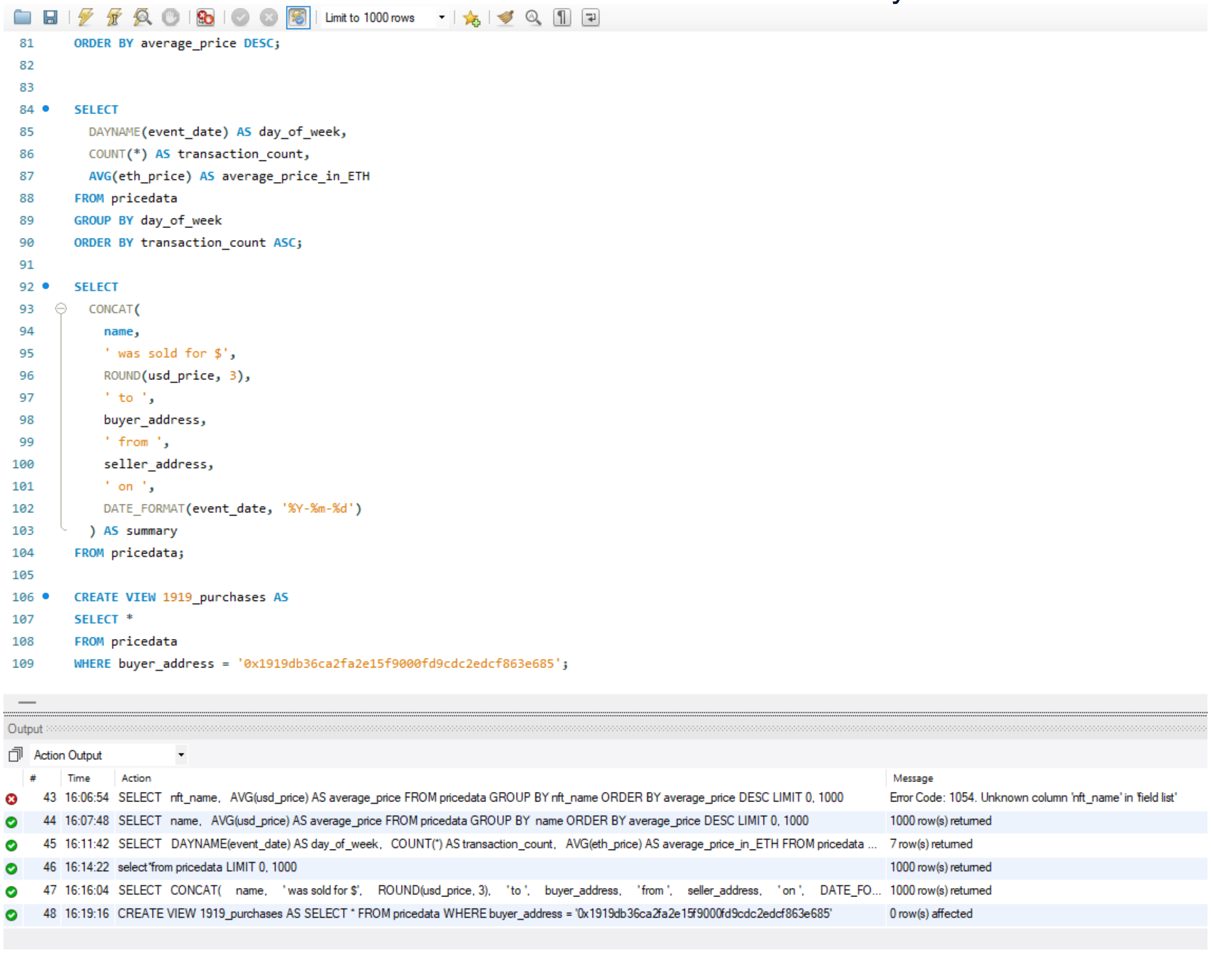
```
summary
CryptoPunk #1139 was sold for $194171.84 to ...
CryptoPunk #3874 was sold for $207300.32 to ...
CryptoPunk #7969 was sold for $162080 to 0x...
CryptoPunk #5231 was sold for $220266.72 to ...
CryptoPunk #3193 was sold for $191254.4 to 0...
CryptoPunk #3961 was sold for $265811.2 to 0...
CryptoPunk #9056 was sold for $232349.46 to ...
CryptoPunk #8335 was sold for $202395 to 0x...
CryptoPunk #2354 was sold for $204075.065 t...
CryptoPunk #1915 was sold for $197528.588 t...
CryptoPunk #1482 was sold for $193445.143 t...
CryptoPunk #4965 was sold for $268826.848 t...
CryptoPunk #9504 was sold for $216033.728 t...
CryptoPunk #6928 was sold for $258942.317 t...
CryptoPunk #3080 was sold for $194449.8 to 0...
CryptoPunk #6050 was sold for $200428.15 to ...
CryptoPunk #3993 was sold for $196203.741 t...
```



```sql
SELECT
  CONCAT(
    name,
    ' was sold for $',
    ROUND(usd_price, 3),
    ' to ',
    buyer_address,
    ' from ',
    seller_address,
    ' on ',
    DATE_FORMAT(event_date, '%Y-%m-%d')
  ) AS summary
FROM pricedata;
```

```
Summary: W#4093 was sold for $0.041 to 0xe038ad9a77a1742f47b8bc9fb0b9cdd473859991 from 0xd9a657acb3960db92aaaa32942019bd3c473fccb on 2021-12-10
```

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



Limit to 1000 rows

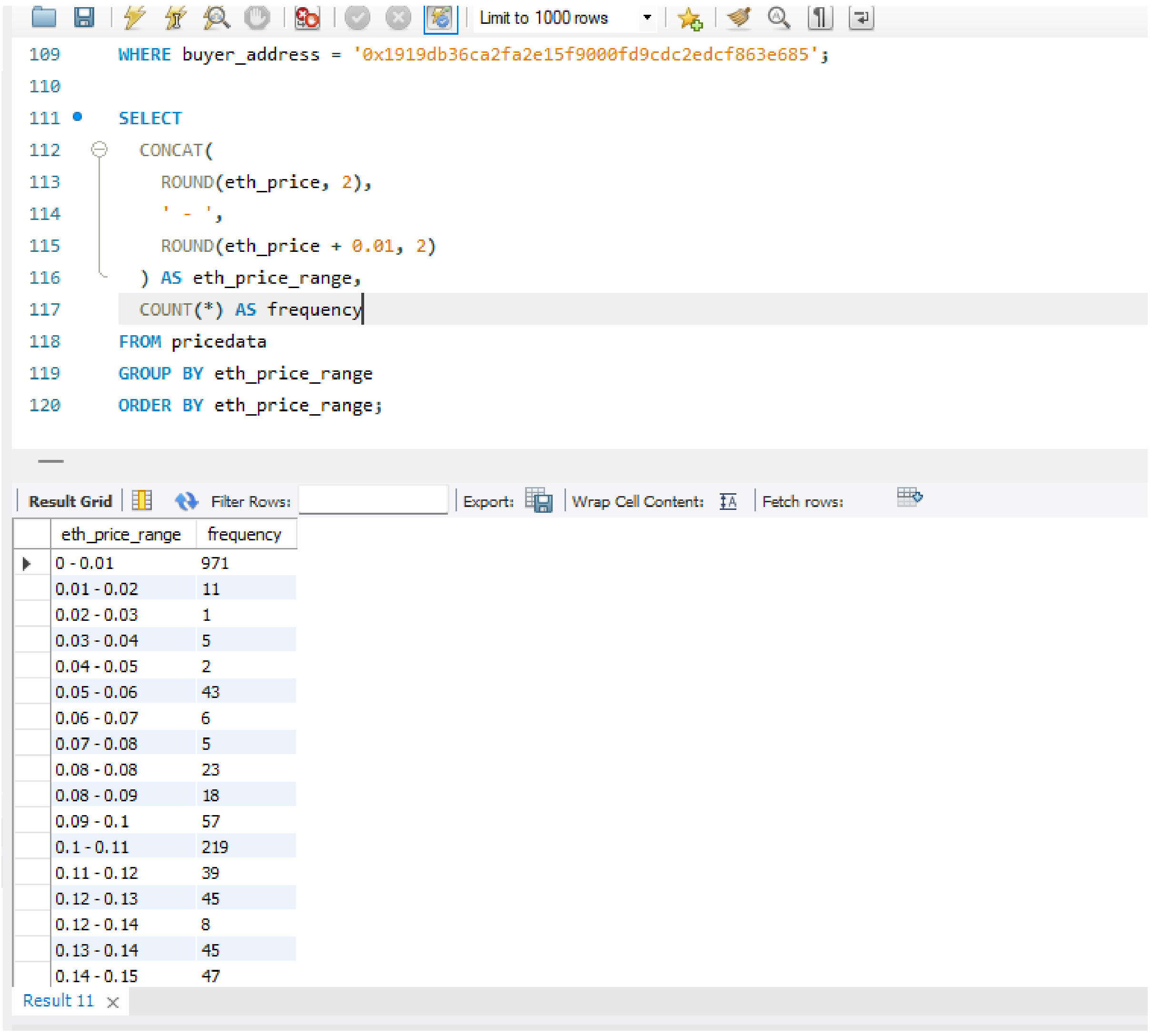
```
ORDER BY average_price DESC;


SELECT
  DAYNAME(event_date) AS day_of_week,
  COUNT(*) AS transaction_count,
  AVG(eth_price) AS average_price_in_ETH
FROM pricedata
GROUP BY day_of_week
ORDER BY transaction_count ASC;

SELECT
  CONCAT(
    name,
    ' was sold for $',
    ROUND(usd_price, 3),
    ' to ',
    buyer_address,
    ' from ',
    seller_address,
    ' on ',
    DATE_FORMAT(event_date, '%Y-%m-%d')
  ) AS summary
FROM pricedata;

CREATE VIEW 1919_purchases AS
SELECT *
FROM pricedata
WHERE buyer_address = '0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685';
```

Output

Action Output

	#	Time	Action	Message
✗	43	16:06:54	SELECT nft_name, AVG(usd_price) AS average_price FROM pricedata GROUP BY nft_name ORDER BY average_price DESC LIMIT 0, 1000	Error Code: 1054. Unknown column 'nft_name' in 'field list'
✓	44	16:07:48	SELECT name, AVG(usd_price) AS average_price FROM pricedata GROUP BY name ORDER BY average_price DESC LIMIT 0, 1000	1000 row(s) returned
✓	45	16:11:42	SELECT DAYNAME(event_date) AS day_of_week, COUNT(*) AS transaction_count, AVG(eth_price) AS average_price_in_ETH FROM pricedata ...	7 row(s) returned
✓	46	16:14:22	select *from pricedata LIMIT 0, 1000	1000 row(s) returned
✓	47	16:16:04	SELECT CONCAT(name, ' was sold for $', ROUND(usd_price, 3), ' to ', buyer_address, ' from ', seller_address, ' on ', DATE_FO...	1000 row(s) returned
✓	48	16:19:16	CREATE VIEW 1919_purchases AS SELECT * FROM pricedata WHERE buyer_address = '0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685'	0 row(s) affected

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

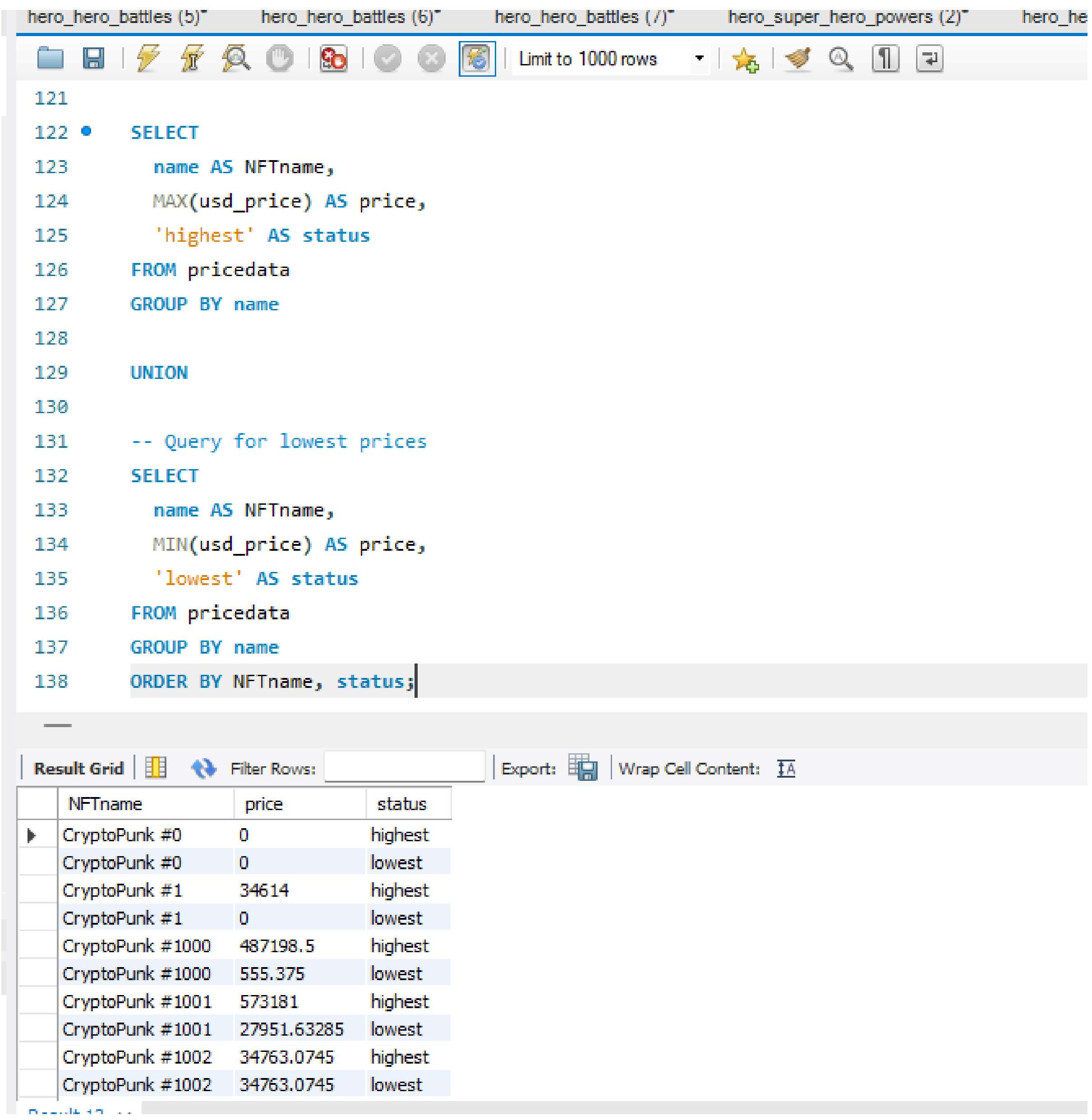


```sql
WHERE buyer_address = '0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685';

SELECT
  CONCAT(
    ROUND(eth_price, 2),
    ' - ',
    ROUND(eth_price + 0.01, 2)
  ) AS eth_price_range,
  COUNT(*) AS frequency
FROM pricedata
GROUP BY eth_price_range
ORDER BY eth_price_range;
```

eth_price_range	frequency
0 - 0.01	971
0.01 - 0.02	11
0.02 - 0.03	1
0.03 - 0.04	5
0.04 - 0.05	2
0.05 - 0.06	43
0.06 - 0.07	6
0.07 - 0.08	5
0.08 - 0.08	23
0.08 - 0.09	18
0.09 - 0.1	57
0.1 - 0.11	219
0.11 - 0.12	39
0.12 - 0.13	45
0.12 - 0.14	8
0.13 - 0.14	45
0.14 - 0.15	47

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

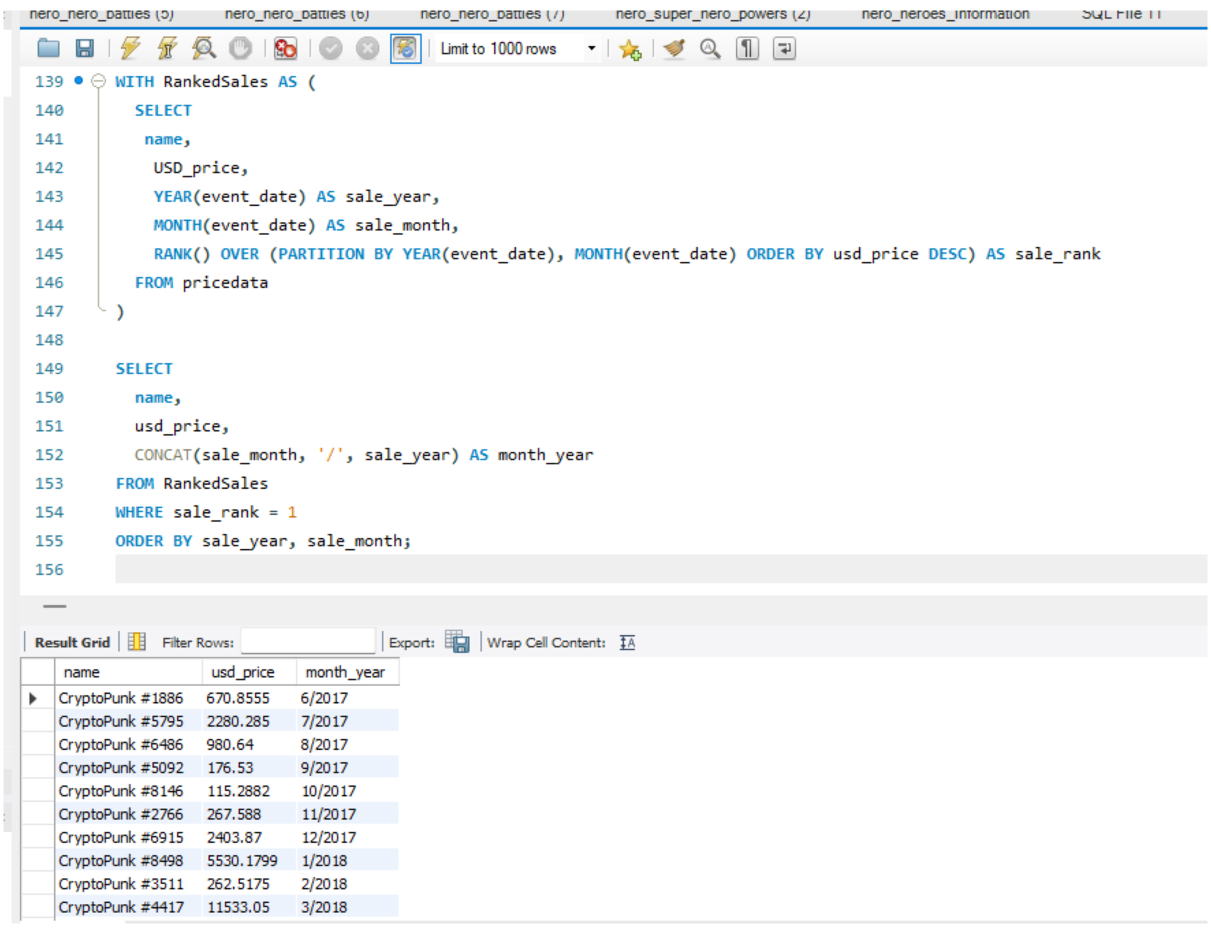
lowest price each NFT was bought for and the status column saying “lowest”. The table should have a name column, a price column called price, and a status column. Order the result set by the name of the NFT, and the status, in ascending order.



```
SELECT
  name AS NFTname,
  MAX(usd_price) AS price,
  'highest' AS status
FROM pricedata
GROUP BY name

UNION

-- Query for lowest prices
SELECT
  name AS NFTname,
  MIN(usd_price) AS price,
  'lowest' AS status
FROM pricedata
GROUP BY name
ORDER BY NFTname, status;
```

NFTname	price	status
CryptoPunk #0	0	highest
CryptoPunk #0	0	lowest
CryptoPunk #1	34614	highest
CryptoPunk #1	0	lowest
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

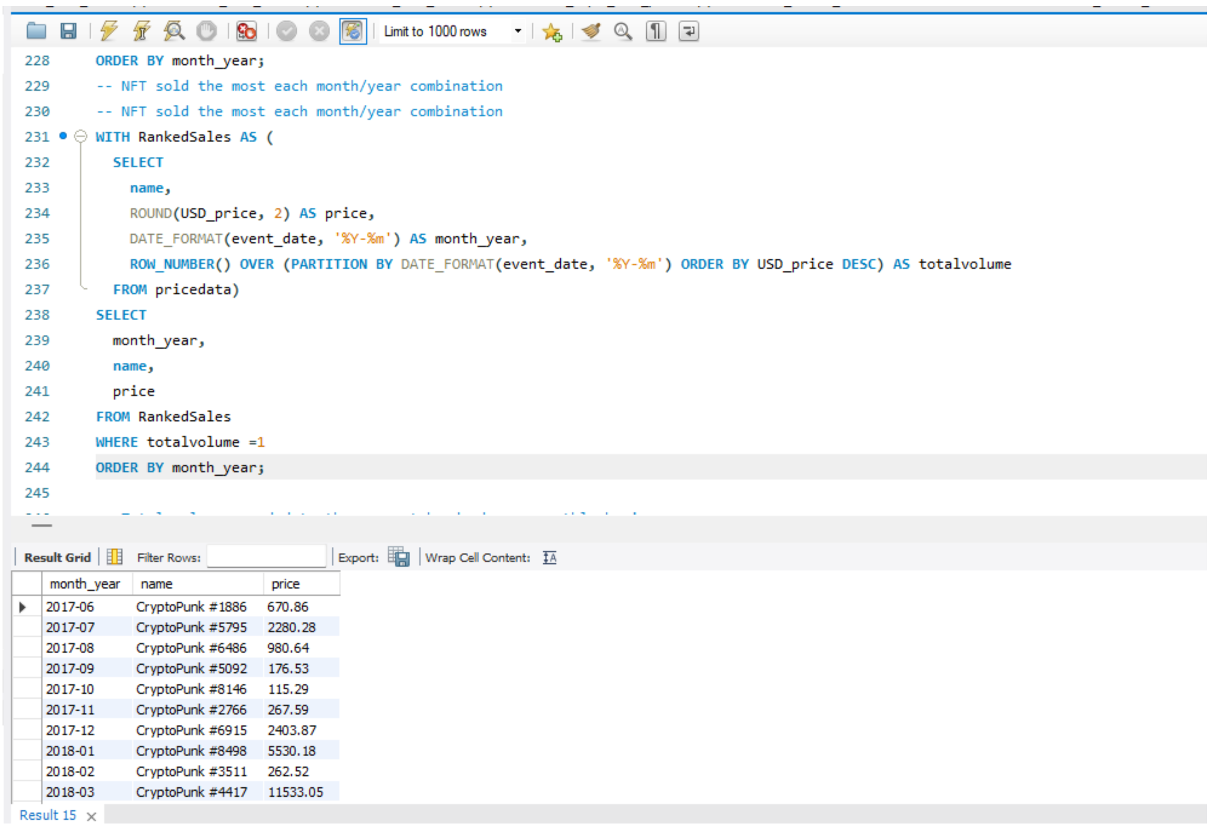
10 What NFT sold the most each month / year combination? Also, what was the name and the price in USD? Order in chronological format.



```
WITH RankedSales AS (
  SELECT
   name,
    USD_price,
    YEAR(event_date) AS sale_year,
    MONTH(event_date) AS sale_month,
    RANK() OVER (PARTITION BY YEAR(event_date), MONTH(event_date) ORDER BY usd_price DESC) AS sale_rank
  FROM pricedata
)

SELECT
  name,
  usd_price,
  CONCAT(sale_month, '/', sale_year) AS month_year
FROM RankedSales
WHERE sale_rank = 1
ORDER BY sale_year, sale_month;
```

name	usd_price	month_year
CryptoPunk #1886	670.8555	6/2017
CryptoPunk #5795	2280.285	7/2017
CryptoPunk #6486	980.64	8/2017
CryptoPunk #5092	176.53	9/2017
CryptoPunk #8146	115.2882	10/2017
CryptoPunk #2766	267.588	11/2017
CryptoPunk #6915	2403.87	12/2017
CryptoPunk #8498	5530.1799	1/2018
CryptoPunk #3511	262.5175	2/2018
CryptoPunk #4417	11533.05	3/2018

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

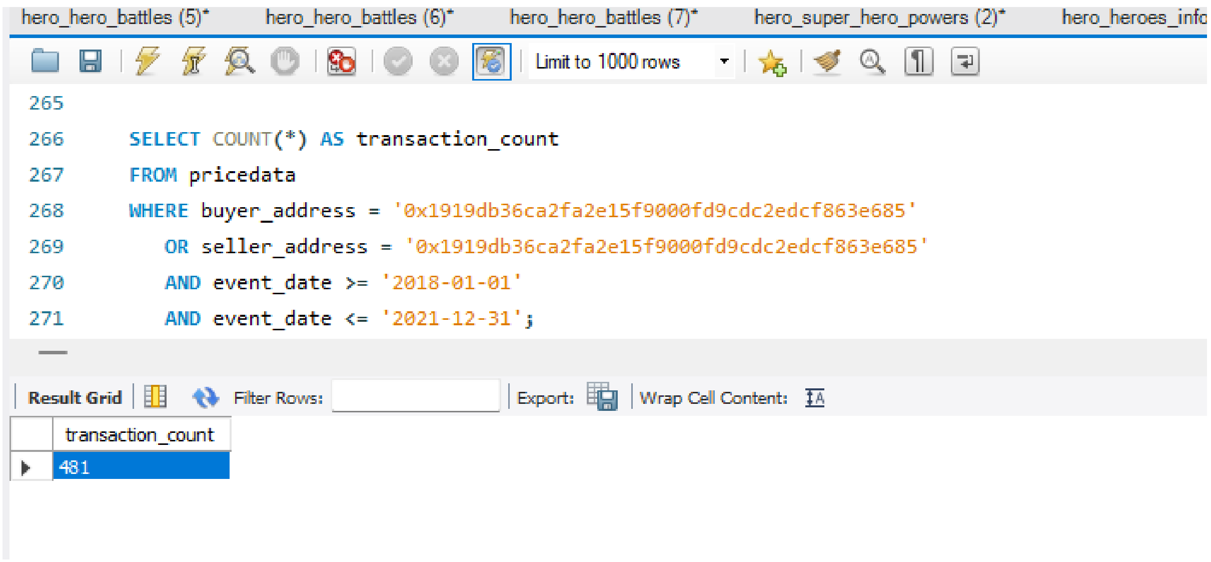


```sql
ORDER BY month_year;
-- NFT sold the most each month/year combination
-- NFT sold the most each month/year combination
WITH RankedSales AS (
  SELECT
    name,
    ROUND(USD_price, 2) AS price,
    DATE_FORMAT(event_date, '%Y-%m') AS month_year,
    ROW_NUMBER() OVER (PARTITION BY DATE_FORMAT(event_date, '%Y-%m') ORDER BY USD_price DESC) AS totalvolume
  FROM pricedata)
SELECT
  month_year,
  name,
  price
FROM RankedSales
WHERE totalvolume =1
ORDER BY month_year;
```

month_year	name	price
2017-06	CryptoPunk #1886	670.86
2017-07	CryptoPunk #5795	2280.28
2017-08	CryptoPunk #6486	980.64
2017-09	CryptoPunk #5092	176.53
2017-10	CryptoPunk #8146	115.29
2017-11	CryptoPunk #2766	267.59
2017-12	CryptoPunk #6915	2403.87
2018-01	CryptoPunk #8498	5530.18
2018-02	CryptoPunk #3511	262.52
2018-03	CryptoPunk #4417	11533.05

Result 15

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



```sql
SELECT COUNT(*) AS transaction_count
FROM pricedata
WHERE buyer_address = '0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685'
   OR seller_address = '0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685'
   AND event_date >= '2018-01-01'
   AND event_date <= '2021-12-31';
```

transaction_count
481

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.

A) CREATE TEMPORARY TABLE temp_daily_averages AS

SELECT

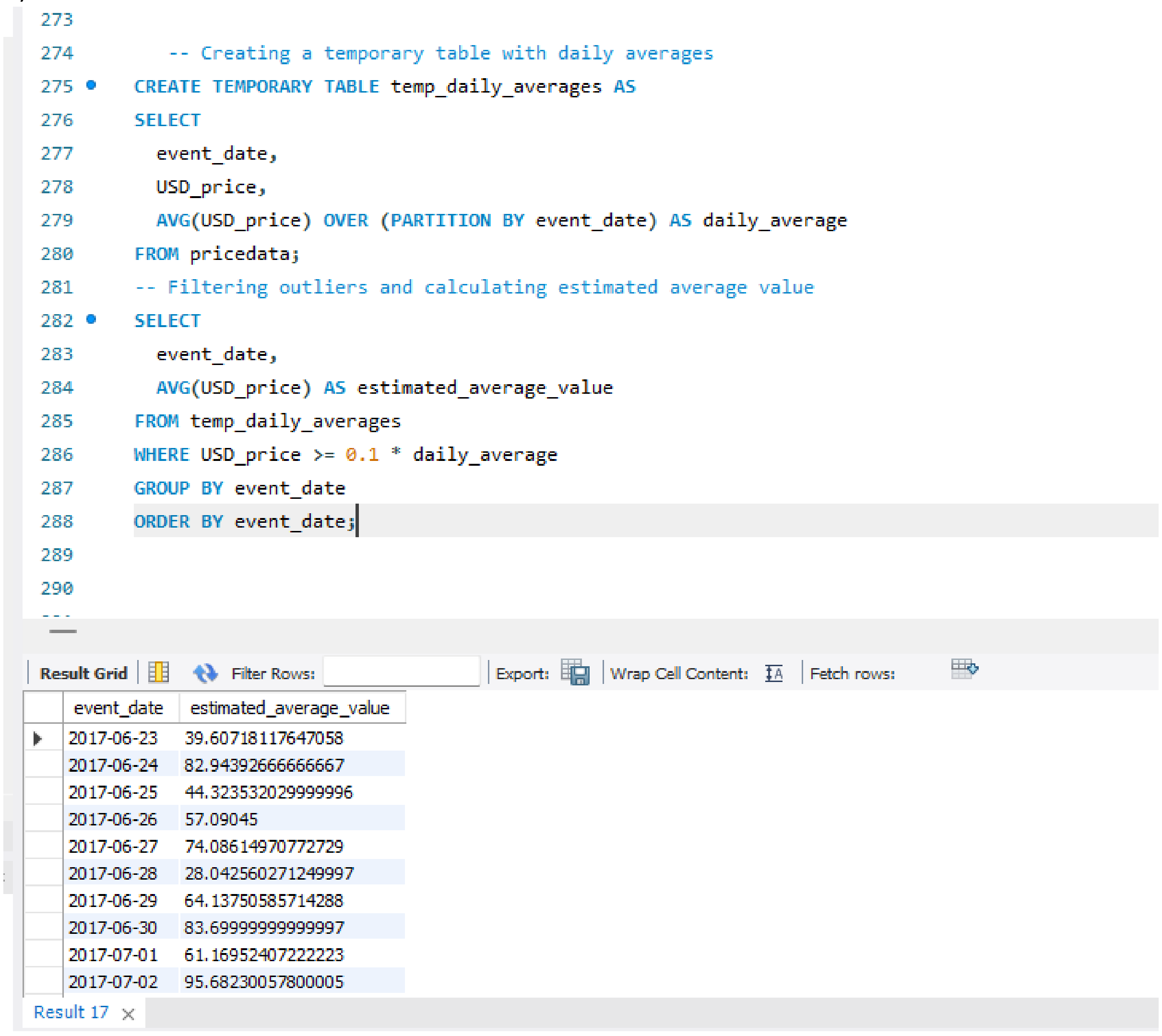
event_date,

USD_price,

AVG(USD_price) OVER (PARTITION BY event_date) AS daily_average

FROM pricedata;

B)



```
   -- Creating a temporary table with daily averages
CREATE TEMPORARY TABLE temp_daily_averages AS
SELECT
  event_date,
  USD_price,
  AVG(USD_price) OVER (PARTITION BY event_date) AS daily_average
FROM pricedata;
-- Filtering outliers and calculating estimated average value
SELECT
  event_date,
  AVG(USD_price) AS estimated_average_value
FROM temp_daily_averages
WHERE USD_price >= 0.1 * daily_average
GROUP BY event_date
ORDER BY event_date;
```

```
event_date   estimated_average_value
2017-06-23   39.60718117647058
2017-06-24   82.94392666666667
2017-06-25   44.323532029999996
2017-06-26   57.09045
2017-06-27   74.08614970772729
2017-06-28   28.042560271249997
2017-06-29   64.13750585714288
2017-06-30   83.69999999999997
2017-07-01   61.16952407222223
2017-07-02   95.68230057800005
```