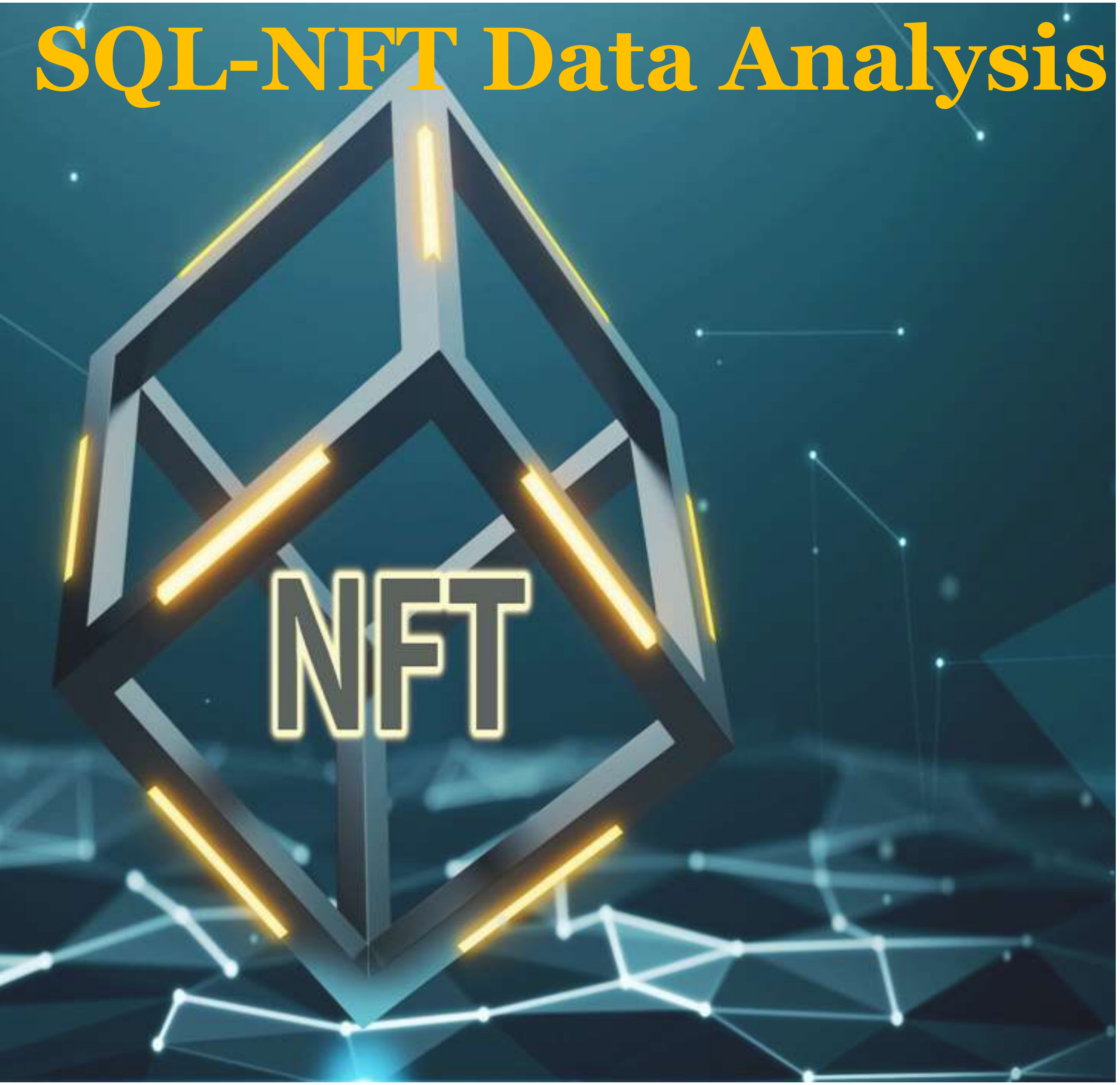
SQL-NFT Data Analysis



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 block chain and over $22 billion was spent last year on purchasing NFTs. Why? People enjoyed the art, the 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 block chain, meaning anytime something happens to an NFT, it is logged in this database.

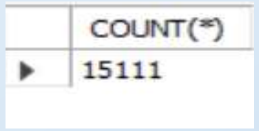
Dataset Overview:

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.

I've to answer the following prompts.

Q1. How many sales occurred during this time period?

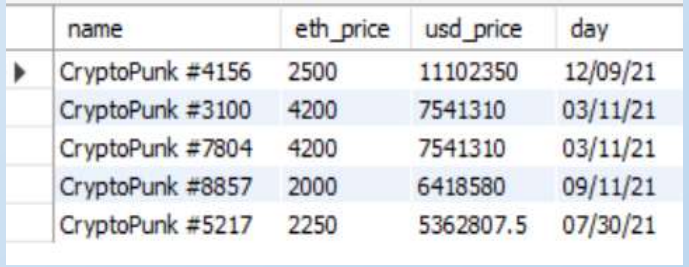
- SELECT COUNT(*) FROM cryptopunkdata;



COUNT(*)
15111

Q2. 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.

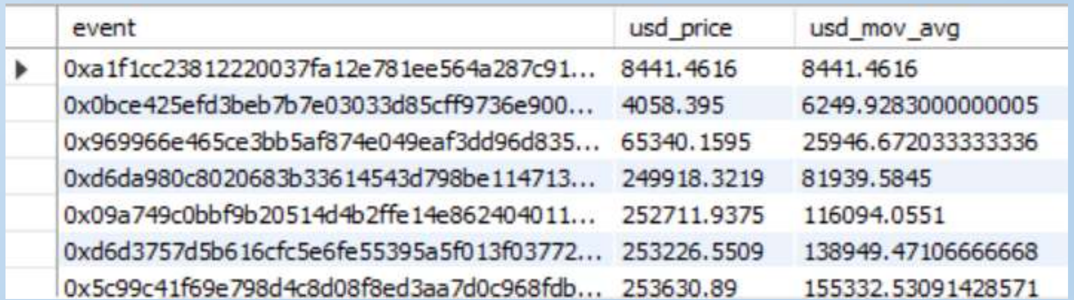
- SELECT name, eth_price, usd_price, day FROM cryptopunkdata ORDER BY usd_price DESC LIMIT 5;



name	eth_price	usd_price	day
CryptoPunk #4156	2500	11102350	12/09/21
CryptoPunk #3100	4200	7541310	03/11/21
CryptoPunk #7804	4200	7541310	03/11/21
CryptoPunk #8857	2000	6418580	09/11/21
CryptoPunk #5217	2250	5362807.5	07/30/21

Q3. 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.

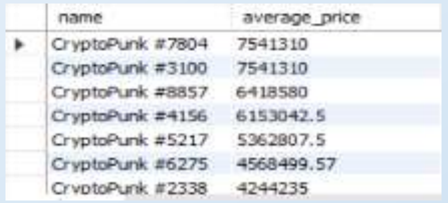
- SELECT transaction_hash AS event, usd_price, AVG(usd_price)OVER(ORDER BY day ROWS BETWEEN 49 preceding AND CURRENT ROW) AS usd_mov_avg FROM cryptopunkdata;



event	usd_price	usd_mov_avg
0xa1f1cc23812220037fa12e781ee564a287c91...	8441.4616	8441.4616
0x0bce425efd3beb7b7e03033d85cff9736e900...	4058.395	6249.928300000005
0x969966e465ce3bb5af874e049eaf3dd96d835...	65340.1595	25946.672033333336
0xd6da980c8020683b33614543d798be114713...	249918.3219	81939.5845
0x09a749c0bbf9b20514d4b2ffe14e862404011...	252711.9375	116094.0551
0xd6d3757d5b616cfc5e6fe55395a5f013f03772...	253226.5509	138949.4710666668
0x5c99c41f69e798d4c8d08f8ed3aa7d0c968fdb...	253630.89	155332.5309142857

Q4. Return all the NFT names and their average sale price in USD. Sort descending. Name the average column as average_price.

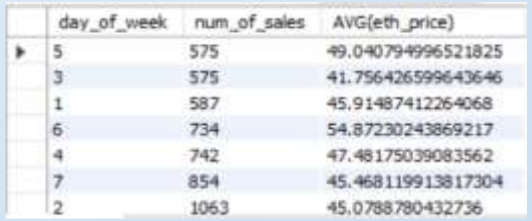
➢ SELECT name, AVG(usd_price) AS average_price FROM cryptopunkdata GROUP BY name ORDER BY average_price DESC;



name	average_price
CryptoPunk #7804	7541310
CryptoPunk #3100	7541310
CryptoPunk #8857	6418580
CryptoPunk #4156	6153042.5
CryptoPunk #5217	5362807.5
CryptoPunk #6275	4568499.57
CryptoPunk #2338	4244235

Q5. 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.

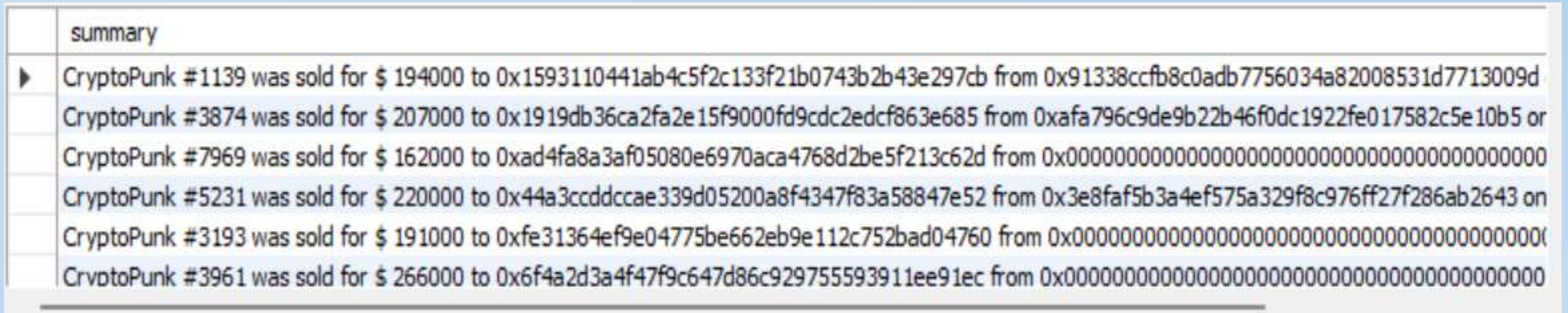
➢ SELECT dayofweek (day) AS day_of_week, COUNT(*) AS num_of_sales, AVG(eth_price)FROM cryptopunkdata GROUP BY day_of_week ORDER BY num_of_sales;



day_of_week	num_of_sales	AVG(eth_price)
5	575	49.040794996521825
3	575	41.756426599643646
1	587	45.91487412264068
6	734	54.87230243869217
4	742	47.48175039083562
7	854	45.468119913817304
2	1063	45.0788780432736

Q6. 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.

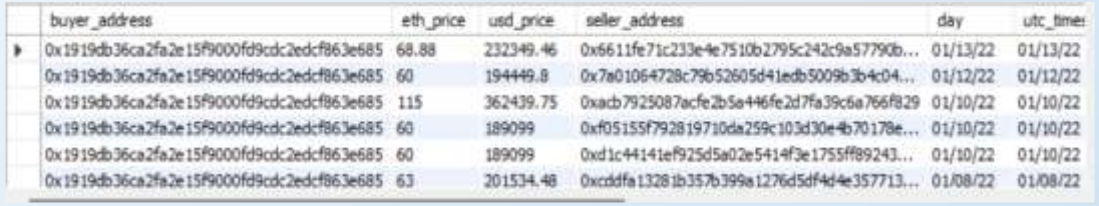
➢ SELECT CONCAT(name," ", 'was sold for $'," ", ROUND(usd_price,-3)," ",'to'," ", seller_address," ",'from'," ", buyer_address," ", 'on'," ", day) AS summary FROM cryptopunkdata;



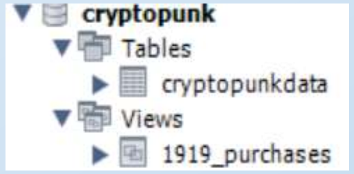
summary
CryptoPunk #1139 was sold for $ 194000 to 0x1593110441ab4c5f2c133f21b0743b2b43e297cb from 0x91338ccfb8c0adb7756034a82008531d7713009d
CryptoPunk #3874 was sold for $ 207000 to 0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685 from 0xafa796c9de9b22b46f0dc1922fe017582c5e10b5 or
CryptoPunk #7969 was sold for $ 162000 to 0xad4fa8a3af05080e6970aca4768d2be5f213c62d from 0x0000000000000000000000000000000000000000
CryptoPunk #5231 was sold for $ 220000 to 0x44a3ccddccae339d05200a8f4347f83a58847e52 from 0x3e8faf5b3a4ef575a329f8c976ff27f286ab2643 on
CryptoPunk #3193 was sold for $ 191000 to 0xfe31364ef9e04775be662eb9e112c752bad04760 from 0x0000000000000000000000000000000000000000
CryptoPunk #3961 was sold for $ 266000 to 0x6f4a2d3a4f47f9c647d86c929755593911ee91ec from 0x0000000000000000000000000000000000000000

Q7. Create a view called “1919_purchases” and contains any sales where “0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685” was the buyer.

- CREATE VIEW 1919_purchases AS SELECT * FROM cryptopunkdata WHERE buyer_address="0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685";

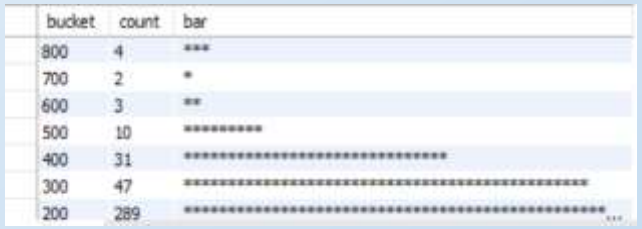


	buyer_address	eth_price	usd_price	seller_address	day	utc_times
▶	0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	68.88	232349.46	0x6611fe71c233e4e7510b2795c242c9a57790b...	01/13/22	01/13/22
	0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	60	194449.8	0x7a01064728c79b52605d41edb5009b3b4c04...	01/12/22	01/12/22
	0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	115	362439.75	0xacb7925087acfe2b5a446fe2d7fa39c6a766f829	01/10/22	01/10/22
	0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	60	189099	0xf05155f792819710da259c103d30e4b70178e...	01/10/22	01/10/22
	0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	60	189099	0xd1c44141ef925d5a02e5414f3e1755ff89243...	01/10/22	01/10/22
	0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	63	201534.48	0xcddfa13281b357b399a1276d5df4d4e357713...	01/08/22	01/08/22



Q8. Create a histogram of ETH price ranges. Round to the nearest hundred value.

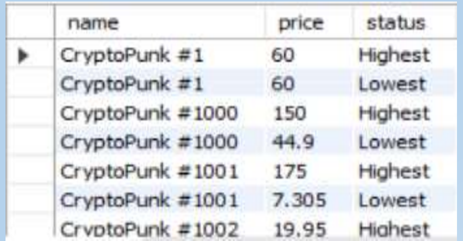
- SELECT ROUND(eth_price,-2) AS bucket, COUNT(*) AS count, RPAD(' ', COUNT(*),'*') AS bar FROM cryptopunkdata GROUP BY bucket ORDER BY bucket DESC;



bucket	count	bar
800	4	***
700	2	*
600	3	**
500	10	*********
400	31	******************************
300	47	**
200	289	**...

Q9. 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.

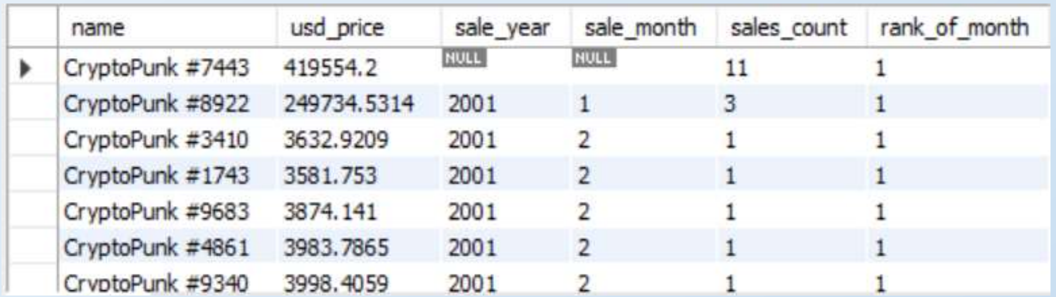
- SELECT name, MAX(eth_price) AS price, 'Highest' AS status FROM cryptopunkdata GROUP BY name UNION SELECT name, MIN(eth_price) AS price, 'Lowest' AS status FROM cryptopunkdata GROUP BY name ORDER BY name, status;



	name	price	status
▶	CryptoPunk #1	60	Highest
	CryptoPunk #1	60	Lowest
	CryptoPunk #1000	150	Highest
	CryptoPunk #1000	44.9	Lowest
	CryptoPunk #1001	175	Highest
	CryptoPunk #1001	7.305	Lowest
	CryptoPunk #1002	19.95	Highest

Q10. What NFT sold the most each month / year combination? Also, what was the name and the price in USD? Order in chronological format.

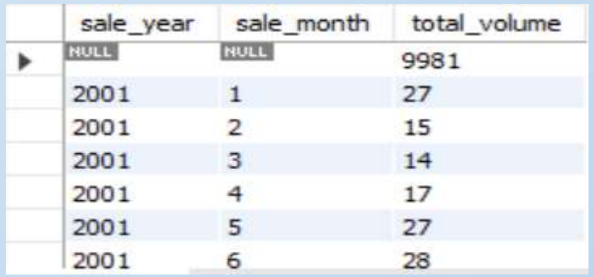
- SELECT name, usd_price, sale_year,sale_month,sales_count,rank_of_month FROM(SELECT name, MAX(usd_price) AS usd_price, YEAR(day) AS sale_year, MONTH(day) AS sale_month, COUNT(*) AS sales_count, DENSE_RANK() OVER(PARTITION BY YEAR(day), MONTH(day) ORDER BY COUNT(*)DESC) AS rank_of_month FROM cryptopunkdata GROUP BY name, YEAR(day),MONTH(day)) AS most_sold WHERE rank_of_month=1;



name	usd_price	sale_year	sale_month	sales_count	rank_of_month
CryptoPunk #7443	419554.2	NULL	NULL	11	1
CryptoPunk #8922	249734.5314	2001	1	3	1
CryptoPunk #3410	3632.9209	2001	2	1	1
CryptoPunk #1743	3581.753	2001	2	1	1
CryptoPunk #9683	3874.141	2001	2	1	1
CryptoPunk #4861	3983.7865	2001	2	1	1
CryptoPunk #9340	3998.4059	2001	2	1	1

Q11. Return the total volume (sum of all sales), round to the nearest hundred on a monthly basis (month/year).

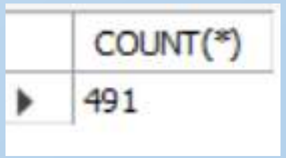
- SELECT YEAR(day) AS sale_year, MONTH(day) AS sale_month, COUNT(*) AS total_volume FROM cryptopunkdata GROUP BY YEAR(day), MONTH(day)ORDER BY YEAR(day), MONTH(day);



sale_year	sale_month	total_volume
NULL	NULL	9981
2001	1	27
2001	2	15
2001	3	14
2001	4	17
2001	5	27
2001	6	28

Q12. Count how many transactions the wallet "0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685"had over this time period

- SELECT COUNT(*) FROM cryptopunkdata WHERE buyer_address='0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685‘ OR seller_address='0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685';

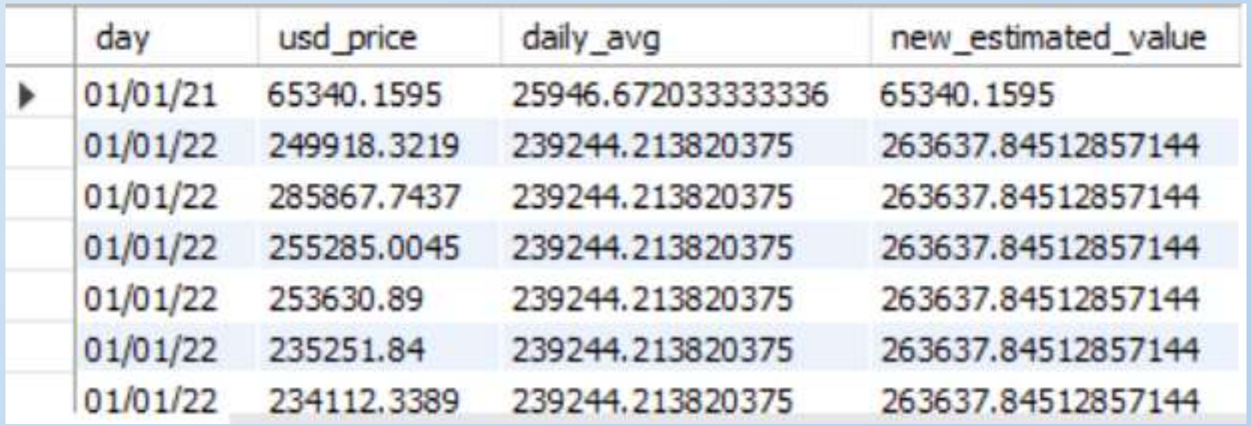


COUNT(*)
491

Q13. 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.

- CREATE TEMPORARY TABLE avg_usd_price_per_day AS SELECT day, usd_price, AVG(usd_price)OVER(PARTITION BY day) AS daily_avg FROM cryptopunkdata;

SELECT*, AVG(usd_price)OVER(PARTITION BY day) AS new_estimated_value FROM avg_usd_price_per_day WHERE usd_price>(0.9*daily_avg);



day	usd_price	daily_avg	new_estimated_value
01/01/21	65340.1595	25946.672033333336	65340.1595
01/01/22	249918.3219	239244.213820375	263637.84512857144
01/01/22	285867.7437	239244.213820375	263637.84512857144
01/01/22	255285.0045	239244.213820375	263637.84512857144
01/01/22	253630.89	239244.213820375	263637.84512857144
01/01/22	235251.84	239244.213820375	263637.84512857144
01/01/22	234112.3389	239244.213820375	263637.84512857144