Data Analysis using SQL - Capstone Project

**Instruction**

**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, 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 blockchain, meaning anytime something happens to an NFT, it is logged in this database.**

**In this project, you’ll be tasked to analyze 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.**

**You might not understand all the jargon around the NFT space, but you should be able to infer enough to answer the following prompts.**

**1.How many sales occurred during this time period?**

**Query**:

SELECT COUNT(\*) AS TotalSales

FROM pricedata

WHERE event\_date >= '2018-01-01' AND event\_date <= '2021-12-31';

**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.**

**Query:**

SELECT eth\_price, name, usd\_price, event\_date from pricedata

ORDER BY usd\_price DESC LIMIT 5;

**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.**

**Query:**

SELECT event\_date, usd\_price, AVG(usd\_price)

OVER (ORDER BY event\_date ROWS BETWEEN 49 PRECEDING

AND CURRENT ROW) AS moving\_average

FROM pricedata;

**4.Return all the NFT names and their average sale price in USD.**

**Sort descending. Name the average column as average\_price.**

**Query:**

SELECT name, AVG(usd\_price) AS average\_price

FROM pricedata GROUP BY name

ORDER BY average\_price DESC;

**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:**

SELECT DAYNAME(event\_date) AS DayOfWeek,

COUNT(\*) AS NumberOfSales,

AVG(ETH\_price) AS AvgPriceInETH FROM pricedata

GROUP BY DayOfWeek

ORDER BY NumberOfSales;

**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:**

SELECT CONCAT(name, ' was sold for $', ROUND(usd\_price, 3),

' to ', buyer\_address, ' from ', seller\_address, ' on ', event\_date)

AS summary FROM pricedata;

**7.Create a view called “1919\_purchases” and contains any sales**

**where “0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685”**

**was the buyer.**

**Query:**

CREATE VIEW 1919\_purchases AS

SELECT \* FROM pricedata

WHERE buyer\_address ='0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685';

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

**hundred value.**

**Query:**

SELECT FLOOR(eth\_price / 100) \* 100 AS ETH\_price\_range,

COUNT(\*) AS frequency

FROM pricedata

GROUP BY ETH\_price\_range

ORDER BY ETH\_price\_range;

**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:**

SELECT name, MAX(usd\_price) AS price, 'highest' AS status

FROM pricedata

GROUP BY name

UNION ALL

SELECT name, MIN(usd\_price) AS price, 'lowest' AS status

FROM pricedata

GROUP BY name

ORDER BY name, status;

**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:**

SELECT YEAR(event\_date) AS year,

MONTH(event\_date) AS month,

name, COUNT(\*) AS sales\_count,

MAX(usd\_price) AS max\_price\_usd

FROM pricedata

GROUP BY year, month, name

ORDER BY year, month;

**11.Return the total volume (sum of all sales), round to the**

**nearest hundred on a monthly basis (month/year).**

**Query:**

SELECT YEAR(event\_date) AS year,

MONTH(event\_date) AS month,

ROUND(SUM(usd\_price), 2) AS total\_volume

FROM pricedata

GROUP BY year, month;

**12.Count how many transactions the wallet**

**"0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685"had over**

**this time period.**

**Query:**

SELECT COUNT(\*) AS Tranasaction\_count FROM pricedata

WHERE buyer\_address ='0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685' OR seller\_address ='0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685'

AND event\_date >= '2018-01-01' AND event\_date <= '2021-12-31’;

**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.**

**Query:**

CREATE TEMPORARY TABLE IF NOT EXISTS daily\_avg\_price AS

SELECT event\_date, usd\_price, Avg(usd\_price) Over(partition by event\_date) As daily\_avg\_price

FROM pricedata;

**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:**

SELECT event\_date, AVG(usd\_price) AS est\_avg\_price FROM daily\_avg\_price

WHERE usd\_price >= 0.1 \* daily\_avg\_price

GROUP BY event\_date;