SQL Capstone Project

1.How many sales occurred during this time period?

SELECT COUNT(\*) AS total\_sales

FROM pricedata

WHERE event\_date BETWEEN '2018-01-01' AND '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.

SELECT name, eth\_price, 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.

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.

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.

SELECT

DAYNAME(event\_date) AS day\_of\_week,

COUNT(\*) AS num\_sales,

AVG(eth\_price) AS average\_price\_eth

FROM pricedata

GROUP BY day\_of\_week

ORDER BY num\_sales ASC;

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”

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.

CREATE VIEW 1919\_purchases AS

SELECT \* FROM pricedata

WHERE buyer\_address = '0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685';

-- To View

SELECT \* FROM 1919\_purchases;

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

SELECT

ROUND(eth\_price, -2) AS price\_range,

COUNT(\*) AS frequency,

RPAD('', COUNT(\*), '\*') AS bar

FROM pricedata

GROUP BY price\_range

ORDER BY 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.

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

FROM pricedata

GROUP BY name

UNION

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.

SELECT

name,

CONCAT(MONTH(event\_date), '/', YEAR(event\_date)) AS date,

MAX(usd\_price) AS price,

COUNT(\*) AS sales\_count

FROM pricedata

GROUP BY name, date

ORDER BY date, sales\_count DESC;

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

SELECT

CONCAT(MONTH(event\_date), '/', YEAR(event\_date)) AS month\_year,

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

FROM pricedata

GROUP BY month\_year

ORDER BY month\_year;

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

SELECT

SUM(CASE WHEN buyer\_address = '0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685' THEN 1 ELSE 0 END) AS buyer\_transaction\_count,

SUM(CASE WHEN seller\_address = '0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685' THEN 1 ELSE 0 END) AS seller\_transaction\_count,

SUM(CASE WHEN buyer\_address = '0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685' OR seller\_address = '0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685' THEN 1 ELSE 0 END) AS total\_count

FROM pricedata

WHERE buyer\_address = '0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685'

OR seller\_address = '0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685';

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

-- Part A:

CREATE TEMPORARY TABLE temp\_daily\_average AS

SELECT

event\_date,

usd\_price,

AVG(usd\_price) OVER (PARTITION BY event\_date) AS daily\_average\_price

FROM pricedata;

-- Part B:

SELECT

event\_date,

AVG(usd\_price) AS estimated\_average\_value

FROM temp\_daily\_average

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

GROUP BY event\_date

ORDER BY event\_date;

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

SELECT

COALESCE(buyer\_address, seller\_address) AS wallet\_address,

SUM(CASE WHEN buyer\_address IS NOT NULL THEN usd\_price ELSE 0 END) AS total\_purchases,

SUM(CASE WHEN seller\_address IS NOT NULL THEN usd\_price ELSE 0 END) AS total\_sales,

SUM(CASE WHEN seller\_address IS NOT NULL THEN usd\_price ELSE 0 END) -

SUM(CASE WHEN buyer\_address IS NOT NULL THEN usd\_price ELSE 0 END) AS profitability

FROM pricedata

GROUP BY COALESCE(buyer\_address, seller\_address)

ORDER BY profitability DESC;