1. How many sales occurred during this time period?

CREATE TEMPORARY TABLE transactions

SELECT \*,COUNT(event\_date) OVER() AS total\_transaction

FROM pricedata

WHERE event\_date BETWEEN "2018-01-01" AND "2021-12-31"

ORDER BY event\_date asc;

1. 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 sales

ORDER BY usd\_price desc

LIMIT 5;

1. 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 50 PRECEDING AND CURRENT ROW) AS moving\_average

FROM pricedata;

1. 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) OVER() AS average\_price

FROM pricedata

ORDER BY average\_price desc;

1. 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.

CREATE TEMPORARY TABLE temp2

SELECT DATE\_FORMAT(event\_date,"%a") AS day\_,COUNT(event\_date) OVER(PARTITION BY DATE\_FORMAT(event\_date,"%a")) AS sale\_count,AVG(usd\_price) OVER(PARTITION BY DATE\_FORMAT(event\_date,"%a")) AS avg\_sale FROM pricedata;

SELECT DISTINCT(day\_),sale\_count,avg\_sale FROM temp2

ORDER BY sale\_count asc;

1. 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 ',usd\_price,' to ',buyer\_address,' from ',seller\_address,' on ',event\_date) AS summary

FROM pricedata;

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

CREATE VIEW view1 AS(

SELECT \* FROM pricedata

WHERE LEFT(buyer\_address,6) = "0x1919");

SELECT \* FROM view1;

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

SELECT ROUND(eth\_price,-2) AS eth,

COUNT(\*) AS count,

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

FROM pricedata

GROUP BY eth

ORDER BY eth;

1. 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,MIN(usd\_price) as price,

CASE

WHEN MIN(usd\_price) != -1 THEN "Low"

ELSE "High"

END AS status

FROM pricedata

GROUP BY name

UNION

SELECT name,MAX(usd\_price) as price,

CASE

WHEN MAX(usd\_price) = 0 THEN "Low"

WHEN MAX(usd\_price) != -1 THEN "High"

ELSE "Low"

END AS status

FROM pricedata

GROUP BY name

ORDER BY price desc;

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

CREATE TEMPORARY TABLE temp3

SELECT DATE\_FORMAT(event\_date,"%Y") as year,DATE\_FORMAT(event\_date,'%b') as month,name,usd\_price,COUNT(name) OVER(PARTITION BY DATE\_FORMAT(event\_date,"%Y"),DATE\_FORMAT(event\_date,'%b'))

AS "month\_total" FROM pricedata;

CREATE TEMPORARY TABLE temp4

SELECT year,month,name,MAX(usd\_price) OVER(PARTITION BY name) as month\_max\_price,count(name) AS montly\_count

FROM temp3

GROUP BY year,month,name,usd\_price

ORDER BY count(name) DESC;

CREATE TEMPORARY TABLE temp5

SELECT year,month,name,month\_max\_price,SUM(montly\_count)

as monthly\_num

FROM temp4

GROUP BY year,month,name,month\_max\_price

ORDER BY count(name) DESC;

CREATE TABLE temp6

SELECT year,month,name,month\_max\_price,monthly\_num,

DENSE\_RANK() OVER(PARTITION BY year,month ORDER BY monthly\_num DESC) AS Rank\_

FROM temp5

ORDER BY monthly\_num DESC;

SELECT \* FROM temp6

WHERE Rank\_ = 1

ORDER BY year,month;

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

WITH volume AS (

SELECT year,month,SUM(usd\_price) OVER(PARTITION BY year,month) AS total\_volume

FROM temp3)

SELECT year,month,ROUND(total\_volume,-2) AS total\_volume FROM volume

GROUP BY year,month,total\_volume;

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

SELECT count(\*) FROM transactions

WHERE buyer\_address ="0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685" or seller\_address = "0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685";

1. 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 average\_

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

FROM pricedata;

WITH new\_avg AS(

SELECT \* FROM average\_

WHERE NOT usd\_price <= 0.1\*daily\_average)

SELECT event\_date,usd\_price, AVG(usd\_price) OVER(PARTITION BY event\_date) AS new\_daily\_average

FROM new\_avg;

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

CREATE TEMPORARY TABLE this

SELECT event\_date,usd\_price,new\_daily\_average,

CASE

WHEN usd\_price>new\_daily\_average THEN "Profit"

ELSE "Loss"

END AS "Profitability"

FROM average\_new;

CREATE TEMPORARY TABLE test

WITH something AS (

SELECT t.event\_date,p.buyer\_address,t.usd\_price,t.new\_daily\_average

FROM this t

JOIN pricedata p

ON t.event\_date=p.event\_date AND t.usd\_price=p.usd\_price)

SELECT DISTINCT(buyer\_address),SUM(usd\_price) OVER(PARTITION BY buyer\_address) AS total\_made,

SUM(new\_daily\_average) OVER(PARTITION BY buyer\_address) AS min\_to\_make

FROM something;

SELECT \*,

CASE

WHEN total\_made>min\_to\_make THEN "Wallet In Profit"

ELSE "Wallet In Loss"

END AS "Profitability"

FROM test;