Analyzing NFT Data: Cryptopunks Sales Analysis

Exploring Trends, Insights, and Patterns in NFT Sales Data

What is this project about?

- Cryptopunks is one of the most famous NFT projects.
- Each row in the dataset represents a sale of a Cryptopunk NFT.
- Dataset includes:
  - **Buyer/Seller Information**
  - **ETH Price and USD Price**
  - **Date, Time, NFT ID, and Name**
  - **Transaction Hash**
- Focus: Glean insights about Cryptopunk sales trends and behaviours.

Goals of the Analysis

The dataset contains:

- **Buyer Address**: Wallet of the purchaser.
- **Seller Address**: Wallet of the seller.
- **ETH Price**: Sale price in Ethereum.
- **USD Price**: Sale price in US dollars.
- **Date and Time**: Timestamp of sale.
- **NFT Name**: The specific Cryptopunk sold.
- **Transaction Hash**: Unique ID of the sale.

Dataset Features

- We aim to:

1. Determine sales volume and top transactions.
2. Understand price trends and averages.
3. Analyze buyer/seller activity and patterns.

DATA PREPARATION

```
# creating schema
CREATE SCHEMA cryptopunk;

USE cryptopunk;
```

```
-- As Per Problem Statement fetching data from Jan 1st, 2018 to Dec 31st, 2021
CREATE TABLE new_pricedata_tbl AS
        SELECT *
        FROM pricedata
        WHERE event_date>'2018-01-01' AND event_date<'2021-12-31';

SELECT * FROM new_pricedata_tbl;
```

Q1.
How many sales occurred during this time period?

```
SELECT
    ROUND(SUM(usd_price)/POWER(10,6),2) AS `Sales (million)`
FROM new_pricedata_tbl;
```

	Sales (million)
▶	1869.21

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, event_date, eth_price, usd_price
FROM new_pricedata_tbl
ORDER BY usd_price DESC
LIMIT 5;
```

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

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.

```sql
SELECT
    event_date,
    ROUND(usd_price, 1) AS usd_price,
    ROUND(
        AVG(usd_price) OVER (
            ORDER BY event_date
            ROWS BETWEEN 49 PRECEDING AND CURRENT ROW
        ), 1
    ) AS moving_avg
FROM new_pricedata_tbl
ORDER BY event_date;
```

event_date	usd_price	moving_avg
2018-01-02	98.7	98.7
2018-01-02	121.4	110.1
2018-01-02	91.1	103.7
2018-01-02	0	77.8
2018-01-02	0	62.2
2018-01-03	86.5	66.3
2018-01-03	85.6	69

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

```
SELECT
    name, Round(AVG(eth_price),1) AS average_price
FROM new_pricedata_tbl
GROUP BY name
ORDER BY average_price DESC;
```

name	average_price
CryptoPunk #3100	4200
CryptoPunk #5217	2250
CryptoPunk #7804	2100
CryptoPunk #7252	1300
CryptoPunk #6275	1159.5
CryptoPunk #8857	1001.2
CryptoPunk #2140	916.7
CryptoPunk #6965	800

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
    WEEKDAY(event_date) AS  number_,
    dayname(event_date) AS  week_name,
    ROUND(AVG(eth_price),1) AS average_price,
    ROUND(SUM(usd_price)/power(10,6),1) AS `Sales (mn)`
FROM new_pricedata_tbl
GROUP BY weekday(event_date),dayname(event_date)
ORDER BY WEEKDAY(event_date);
```

number_	week_name	average_price	Sales (mn)
0	Monday	32	282.8
1	Tuesday	31.2	211.4
2	Wednesday	31.9	195.6
3	Thursday	37	283.7
4	Friday	38.2	320
5	Saturday	45.2	340
6	Sunday	31.3	235.7

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 $", usd_price, " to ",buyer_address," from " ,
        seller_address, " on ", event_date)
    AS summary
FROM new_pricedata_tbl;
```

summary
CryptoPunk #8808 was sold for $388429.26 to 0xfe2f279d3679bac2d07cf46c93503410ef9ca448 from 0x6639c089adfba8bb9968da643c6be208a70d6daa on ...
CryptoPunk #6307 was sold for $373908.54 to 0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685 from 0xfe2f279d3679bac2d07cf46c93503410ef9ca448 on 2...
CryptoPunk #3994 was sold for $383673.7242 to 0x59d22c0cd4b1ba17f364cbab4d3a1489e780b036 from 0x6ec30fd91a504aad948839b985c7263888b2ad68...

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

```
CREATE VIEW 1919_purchases AS
SELECT * FROM new_pricedata_tbl
WHERE buyer_address="0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685";

SELECT * FROM 1919_purchases;
```

buyer_address	eth_price	usd_price	seller_address	event_date	token_id	transa
0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	103	373908.54	0xfe2f279d3679bac2d07cf46c93503410ef9ca448	2021-12-30	6307	0xacdd
0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	87	315825.66	0x794316614b210acd02f7c88085f2872a8d657...	2021-12-30	6138	0xca51
0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	65	235961.7	0xa438ddd77fdbeca2496f0c5a39b317a67272e...	2021-12-30	7564	0x60a8
0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	64	232331.52	0x4c6b83ca1c59781fab57790a46281bbd93e53...	2021-12-30	197	0xd302
0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	81	307286.46	0x0c358da7b66fbf19d1114e9ac810106c8d1bd...	2021-12-29	1556	0x2204
0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685	67	254175.22	0x9a70d89db9a7ae5c04d2bea3049d136098e3	2021-12-29	2376	0xe1e2

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 new_pricedata_tbl
GROUP BY Bucket
ORDER BY COUNT(*) DESC;
```

Bucket	count	Bar
0	14348	**********
100	3729	**********
200	287	**********
300	45	**********
400	31	**********
500	10	**********
800	4	****
600	3	***

Q9. Return a union 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 new_pricedata_tbl
GROUP BY name)
UNION
(SELECT
    name, MIN(eth_price) AS price,'Lowest' AS status
FROM new_pricedata_tbl
GROUP BY name);
```

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

```
WITH cte AS(
        SELECT
            name,
            YEAR(event_date) AS year, MONTH(event_date) AS month,
            eth_price, usd_price,  ROUND(usd_price/eth_price,0) AS quantity
            ,RANK() OVER(PARTITION BY YEAR(event_date), MONTH(event_date) order by
                usd_price/eth_price DESC) AS rnk
        FROM new_pricedata_tbl)
SELECT
    year, month, name, eth_price, quantity
FROM cte
WHERE rnk=1;
```

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

```sql
SELECT
    YEAR(event_date) AS year,
    MONTH(event_date) AS month,
    ROUND(SUM(usd_price)/POWER(10,3),1) AS `total (thousands)`
FROM new_pricedata_tbl
GROUP BY year,month
ORDER BY year,month;
```

Q12.

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

```
SELECT COUNT(*) AS count
FROM new_pricedata_tbl
WHERE buyer_address="0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685"
    OR seller_address="0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685";
```

	count
▶	466

Q13.

Create an “estimated average value calculator” that has a representative price of the collection every day based 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.

```sql
CREATE TEMPORARY TABLE average_calculator AS
SELECT
    event_date,
    usd_price,
    AVG(usd_price) OVER(PARTITION BY event_date) AS daily_avg_price
FROM new_pricedata_tbl;

SELECT
    event_date,
    ROUND(AVG(usd_price), 2) AS estimated_daily_value
FROM average_calculator
WHERE usd_price>0.1*daily_avg_price
GROUP BY event_date
ORDER BY event_date;
```

What is the total USD price (in thousands) for each year and month, including yearly totals and a grand total?

```sql
WITH CTE AS (
SELECT
    event_date,
    usd_price,
    YEAR(event_date) AS year,
    MONTH(event_date) AS month
FROM new_pricedata_tbl)
SELECT
    COALESCE(year, 'Total') AS year,
    COALESCE(month,'') AS month,
    ROUND(SUM(usd_price)/POWER(10,3), 1) AS `total (thousands)`
FROM CTE
GROUP BY year,month WITH ROLLUP
ORDER BY year, month;
```

What Did We Learn?

- **Sales Volume and Trends**:
Total sales and distribution over time highlight the popularity and peak periods of Cryptopunks.

- **High-Value Transactions**:
Identified the most expensive NFTs, showcasing the significant monetary value in the NFT space.

- **Buyer and Seller Behavior**:
Transaction summaries reveal key wallets driving activity.

- **Average Prices & Daily Patterns**:
Moving averages and daily averages show price fluctuations.