SQL (Structured Query Language) is a foundational tool for data analysts, empowering them to extract valuable insights from vast datasets. As a specialized programming language, SQL facilitates seamless interaction with relational databases, allowing analysts to craft queries that filter, aggregate, and manipulate data.

Its efficiency in handling large datasets ensures quick and precise analysis, making it an indispensable skill for data professionals. SQL enables data analysts to retrieve specific information, perform complex calculations, and uncover patterns, empowering informed decision-making.

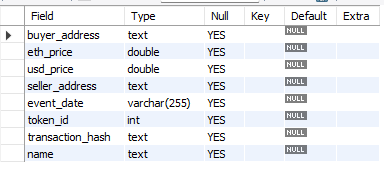
Its widespread use across industries underscores its significance in transforming raw data into actionable intelligence, cementing SQL as an essential tool in the data analyst’s toolkit.

**Understanding NFTs data**

In this blog, we’ll delve into an insightful project that uses SQL to analyze NFT sales data. NFTs, or Non-Fungible Tokens, have taken the digital art and collectibles world by storm, offering a new way to buy, sell, and trade digital assets. Our project focuses on extracting valuable insights from NFT sales data, using SQL queries to uncover trends and patterns.

Let’s explore the structural details of the dataset by examining the table configuration.

DESCRIBE pricedata;



Data is a sales dataset 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.

Let’s kick off our data exploration with a set of 14 questions, starting from easy ones and gradually moving towards more intricate queries. These questions are tailored to make our analysis both engaging and insightful, turning our data examination into an enjoyable journey. Join me as we unravel the nuances of the dataset together!

**Question 01:**

*How many sales occurred during this time period?*

**Understanding the Volume of Sales**

The first step in our analysis is to determine the total number of sales that occurred during a whole time period. This is achieved with a straightforward SQL query:

SELECT   
 COUNT(\*)  
FROM  
 pricedata;

This query counts the number of rows in the *pricedata* table, effectively giving us the total number of sales. That is 19,920.

**Question 02:**

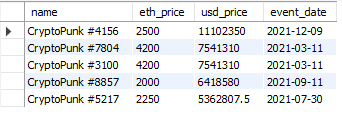
*Return the top 5 most expensive transactions (by USD price) for this data set. Return the name, ETH price, and USD price, as well as the date.*

**Identifying Top Transactions**

Next, we aim to identify the top five most expensive transactions. This information is critical for understanding the market’s high-end segment and identifying the most valued NFTs. The query below does the job:

SELECT   
 name, eth\_price, usd\_price, event\_date  
FROM  
 pricedata  
ORDER BY usd\_price DESC  
LIMIT 5;

By ordering the transactions in descending order of their USD price and limiting the results to the top five, we can quickly see the most expensive NFT sales. That is



**Question 03:**

*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*

**Analyzing Price Trends with Moving Averages**

Understanding the price dynamics over time is crucial in any market analysis. For this, we use a moving average of the USD price. A moving average helps smooth out short-term fluctuations and highlight longer-term trends. The following query demonstrates this approach:

SELECT   
 transaction\_hash,   
 usd\_price,   
 AVG(usd\_price) OVER(ORDER BY event\_date ROWS BETWEEN 49 PRECEDING AND CURRENT ROW) as usd\_mv\_avg  
FROM  
 pricedata;

This query provides insights into the market trend by averaging the prices over a set number of preceding transactions, offering a clearer picture of the price dynamics in the NFT market. By applying this analysis, investors and enthusiasts can gain a deeper understanding of market movements and make more informed decisions in the NFT space.

**Question 04:**

*Return all the NFT names and their average sale price in USD. Sort descending. Name the average column as average\_price.*

**Analyzing Average Sale Prices of NFTs**

A crucial aspect of understanding the NFT market is knowing the average sale prices of different NFTs. This helps in identifying which NFTs are more valuable on average and understanding the market’s preference. The SQL query below calculates the average sale price for each NFT and sorts the results in descending order:

SELECT   
 name, AVG(usd\_price) as average\_price  
FROM  
 pricedata  
GROUP BY name  
ORDER BY average\_price DESC;

This query groups the sales data by the NFT name and calculates the average USD price for each group. Sorting the results by *average\_price*in descending order highlights the most valuable NFTs based on their average sale price.

This approach offers an aggregated view of the NFT market, allowing stakeholders to quickly identify which NFTs command higher average prices and may therefore be in higher demand or more limited supply.

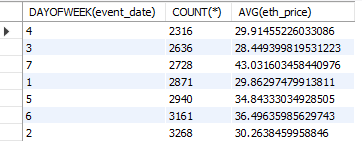
**Question 05:**

*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.*

**Weekly Trends in NFT Sales and Prices**

Understanding the NFT market also involves analyzing sales patterns across different days of the week. This can reveal insights into buyer behavior and market dynamics on specific days. The following SQL query provides a breakdown of NFT sales and average prices in Ethereum (ETH) by each day of the week, ordered by the number of transactions:

SELECT   
 DAYOFWEEK(event\_date), COUNT(\*), AVG(eth\_price)  
FROM  
 pricedata  
GROUP BY DAYOFWEEK(event\_date)  
ORDER BY COUNT(\*);



This query:

* Utilizes *DAYOFWEEK(event\_date) to*categorize transactions by the day of the week.
* Counts the number of transactions and calculates the average ETH price for each day.
* Orders the results by the number of transactions, showing the days with the least activity first.

Such an analysis is useful for identifying trends and patterns in purchasing behavior, potentially informing marketing and sales strategies for NFT creators and platforms.

**Question 06:**

*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”***

**Generating Summaries of NFT Transactions**

For a more personalized and narrative approach to NFT data analysis, generating transaction summaries can be insightful. These summaries provide a quick, readable overview of each transaction, including key details like the NFT name, sale price, seller, buyer, and transaction date. The following SQL query creates such summaries:

SELECT   
 (CONCAT(name,  
 ' was sold for $',  
 ROUND(usd\_price, -3),  
 ' to ',  
 seller\_address,  
 ' from ',  
 buyer\_address,  
 ' on ',  
 event\_date)) as summary  
FROM  
 pricedata;

This query:

* Concatenates various fields (*name, usd\_price, seller\_address, buyer\_address,*and*event\_date*) into a single string.
* Rounds the *usd\_price* to the nearest thousand for a cleaner look.
* Labels the result as *summary*for each transaction.

These summaries offer a straightforward, human-readable format, making it easier to quickly understand the specifics of each transaction. This can be particularly useful for presentations, reports, or simply for a more engaging way to explore the data.

**Question 07:**

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

**Creating a View for Specific Buyer Transactions in NFT Sales**

In some cases, we may want to focus on the transactions of a specific buyer within the NFT market. This is particularly useful for tracking the activity of prominent buyers or for conducting detailed analysis on specific market participants. The following SQL query creates a view called *1919\_purchases*, which includes all sales where a particular buyer, identified by their address *0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685*, was involved:

CREATE VIEW 1919\_purchases AS  
 SELECT   
 \*  
 FROM  
 pricedata  
 WHERE  
 buyer\_address = '0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685';

This query:

* Uses *CREATE VIEW* to establish a new view named *1919\_purchases*.
* Selects all columns from the *pricedata* table.
* Filters the data to include only those transactions where the specified address is the buyer.

Creating a view like *1919\_purchases*is an efficient way to repeatedly access a subset of data without the need to run complex queries each time. This can be particularly valuable for ongoing analysis or monitoring of specific entities in the NFT marketplace.

**Question 08:**

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

**Creating a Histogram of ETH Price Ranges for NFT Sales**

Visualizing the distribution of NFT prices can provide valuable insights into the market. A histogram is an excellent tool for this purpose, as it allows us to see how many sales fall into different price ranges. The following SQL query creates a histogram of ETH prices for NFT sales, rounding prices to the nearest hundred and displaying the distribution:

SELECT   
 ROUND(eth\_price, -2) AS bucket,  
 COUNT(\*) AS count,  
 RPAD('', COUNT(\*), '\*') AS bar  
FROM  
 pricedata  
GROUP BY bucket  
ORDER BY bucket;



This query:

* Rounds the *eth\_price*to the nearest hundred (indicated by -2) to create price buckets.
* Counts the number of sales (*COUNT(\*)*) in each bucket.
* Uses *RPAD* to create a text-based bar for each bucket, where the length of the bar (\*) corresponds to the count of sales in that bucket.

This histogram provides a quick, visual representation of the ETH price distribution in the NFT market, making it easier to understand which price ranges are most common. It’s a straightforward yet powerful way to analyze the market’s pricing structure.

**Question 09:**

*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.*

**Comparing Highest and Lowest Prices for Each NFT**

In the NFT market, understanding the range of prices for which a particular NFT has been sold can be quite enlightening. It provides insights into the variability of the market and the perceived value of each NFT. The following SQL query uses a union of two subqueries to list the highest and lowest sale prices for each NFT, along with a status indicator:

SELECT   
 name, MAX(eth\_price) AS price, 'Highest' AS status  
FROM  
 pricedata  
GROUP BY name   
UNION   
SELECT   
 name, MIN(eth\_price) AS price, 'Lowest' AS status  
FROM  
 pricedata  
GROUP BY name  
ORDER BY name;

This query:

* First, selects the name of the NFT, the maximum price it was sold for (*MAX(eth\_price)*), and labels this as 'Highest'.
* Then, it uses *UNION* to combine this with another query selecting the name of the NFT, the minimum price it was sold for (*MIN(eth\_price)*), and labels this as 'Lowest'.
* The results are grouped by the name of the NFT and ordered by the name.

This combined query provides a comprehensive view of the price range for each NFT, allowing for a comparative analysis of their highest and lowest sale values. Such insights are particularly useful for buyers and sellers in understanding the price volatility and market trends of specific NFTs.

**Question 10:**

*What NFT sold the most each month / year combination? Also, what was the name and the price in USD? Order in chronological format. I have consider the all top NFTs with same number of sale\_counts*

**Identifying Top-Selling NFTs Each Month and Year**

To gain a deeper understanding of the NFT market trends over time, it’s valuable to identify which NFTs had the highest number of sales in each month-year combination, along with their names and prices in USD. The following SQL query achieves this by calculating the top-selling NFTs for each month and year, considering only those with the highest sale counts:

SELECT   
 name,  
 usd\_price,  
 sale\_year,  
 sale\_month,  
 sale\_count,  
 ranked\_in\_month  
FROM (  
 SELECT   
 name,  
 MAX(usd\_price) as usd\_price,  
 YEAR(event\_date) AS sale\_year,  
 MONTH(event\_date) AS sale\_month,  
 COUNT(\*) AS sale\_count,  
 DENSE\_RANK() OVER (PARTITION BY YEAR(event\_date), MONTH(event\_date) ORDER BY COUNT(\*) DESC) as ranked\_in\_month  
 FROM  
 pricedata  
 GROUP BY name, YEAR(event\_date), MONTH(event\_date)  
) as dt  
WHERE ranked\_in\_month = 1;

This query:

* First, groups the sales data by NFT name and each month-year combination.
* Calculates the maximum USD price, total sales count (*COUNT(\*)*), and assigns a rank based on the sales count for each month-year (*DENSE\_RANK()*).
* Filters to include only those NFTs that are ranked first in their respective month-year, indicating they are the top-selling for that period.

This approach provides a clear view of which NFTs dominated the market in terms of sales volume at different times, offering valuable insights into market preferences and trends. Such data can be pivotal for market analysts, investors, and creators in the NFT space.

**Question 11:**

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

**Monthly Sales Volume Analysis in NFT Market**

Understanding the total volume of sales in the NFT market on a monthly basis is crucial for grasping market dynamics and trends. The following SQL query calculates the total sales volume, rounded to the nearest hundred, for each month and year, providing a clear picture of how the market has evolved over time:

SELECT   
 YEAR(event\_date) AS sale\_year,  
 MONTH(event\_date) AS sale\_month,  
 ROUND(SUM(usd\_price), -2) AS sum\_of\_sales\_volume  
FROM  
 pricedata  
GROUP BY YEAR(event\_date), MONTH(event\_date)  
ORDER BY YEAR(event\_date), MONTH(event\_date);

This query:

* Extracts the year and month from *event\_date* and groups the data accordingly.
* Sums up the *usd\_price* for each group, rounding the total to the nearest hundred for a cleaner representation.
* Orders the results chronologically by year and month.

This analysis is particularly helpful for identifying seasonal trends, growth patterns, or declines in the NFT market, offering valuable insights for market participants, analysts, and investors.

**Question 12:**

*Count how many transactions the wallet “0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685”had over this time period.*

**Transaction Count for a Specific Wallet in the NFT Market**

Analyzing the activity of a specific wallet in the NFT market can provide insights into the behavior of individual market participants. The following SQL query counts the number of transactions associated with the wallet address “0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685”, considering both buying and selling activities:

SELECT   
 COUNT(\*)  
FROM  
 pricedata  
WHERE  
 buyer\_address = '0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685'  
 OR seller\_address = '0x1919db36ca2fa2e15f9000fd9cdc2edcf863e685';

This query:

* Checks both *buyer\_address* and *seller\_address* fields in the *pricedata* table.
* Uses *COUNT(\*)* to tally the total number of transactions where the specified wallet address appears as either the buyer or the seller.

Such an analysis is valuable for understanding the level of activity of a specific participant in the NFT market, which can be indicative of market influence, investment strategy, or other behavioral patterns.

**Question 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*

**Creating an Estimated Average Value Calculator for NFT Collections**

To accurately gauge the value of NFT collections, it’s essential to consider daily transaction data while filtering out anomalies. This approach involves creating an “estimated average value calculator” that excludes outlier sales and focuses on the average transaction value each day. The SQL query for this involves two main steps:

**Part A: Creating a Temporary Table**

First, we create a temporary table to hold the daily average USD price for each transaction. This table also includes the event date and the individual transaction price:

CREATE TEMPORARY TABLE avg\_usd\_price\_per\_day AS  
SELECT   
 event\_date,   
 usd\_price,   
 AVG(usd\_price) OVER (PARTITION BY DATE(event\_date)) AS daily\_avg  
FROM  
 pricedata;

This query:

* Selects *event\_date*and *usd\_price*from the *pricedata*table.
* Uses a window function to calculate the daily average (*daily\_avg*) of *usd\_price*.

**Part B: Calculating the New Estimated Value**

Next, we use the created temporary table to filter out transactions that are below 10% of the daily average price, then calculate a new daily average from the remaining data:

SELECT   
 \*,   
 AVG(usd\_price) OVER (PARTITION BY DATE(event\_date)) AS new\_estimated\_value  
FROM  
 avg\_usd\_price\_per\_day  
WHERE  
 usd\_price > (0.9 \* daily\_avg);

This query:

* Utilizes the *avg\_usd\_price\_per\_day*temporary table.
* Applies a filter to exclude transactions where *usd\_price*is less than 10% of the *daily\_avg*.
* Recalculates the average (*new\_estimated\_value*) based on the filtered data.

This two-step process results in a more accurate and representative average value of NFT collections by excluding significant outliers. It’s an effective way to assess the daily value trend of NFTs, providing a clearer picture of the market for analysts and investors.

**Question 14:**

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

**Analyzing Wallet Profitability in the NFT Market**

In the dynamic world of NFT trading, understanding the profitability of individual wallets is key to gaining insights into trading strategies and market behavior. The following SQL query provides a complete list of wallets ordered by their profitability, indicating whether the wallet owners have made or lost money through their transactions:

SELECT   
 walletID, SUM(cost\_of\_trade) as profitability  
FROM  
 (SELECT   
 buyer\_address AS walletID, (usd\_price \* -1) AS cost\_of\_trade   
 FROM  
 pricedata   
 UNION ALL  
 SELECT   
 seller\_address AS walletID, usd\_price AS cost\_of\_trade   
 FROM  
 pricedata) AS total\_transactions  
GROUP BY walletID  
ORDER BY SUM(cost\_of\_trade);

This query:

* Creates a subquery that consolidates all transactions, treating buys as negative costs and sales as positive.
* Each transaction is associated with a *walletID*, which represents either a buyer or a seller.
* The outer query then sums up the *cost\_of\_trade*for each wallet, treating sales as positive values and purchases as negative.
* The final result is grouped by *walletID*and ordered by the sum of *cost\_of\_trade*, showing the overall profitability for each wallet.

This profitability analysis is crucial for understanding who the key players are in the NFT market and how successful their trading strategies have been. It can reveal patterns in buying and selling behavior and highlight the most effective traders.

**Conclusion**

This project showcases the power of SQL in extracting meaningful insights from complex datasets like NFT sales. By using various queries, we can understand the overall sales volume, identify the highest-value transactions, and analyze price trends over time. These insights are invaluable for investors, collectors, and anyone interested in the burgeoning world of NFTs.

This blog provides a glimpse into the potential of data analysis in understanding emerging digital markets. SQL, with its powerful querying capabilities, proves to be an essential tool in the data analyst’s toolkit, especially in the dynamic and rapidly evolving world of NFTs.