Data Analysis of NFT Crptopunks using SQL

```
use cryptopunk;
```

```
select * from pricedata;
```

1. How many sales occurred during this time period?

```
select count(*) as sales from pricedata;
```

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 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 desc) as Moving_Average
from pricedata
limit 50;
```

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) over(order by usd_price desc) as Average_Price
from pricedata
order by usd_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 event_date, count(transaction_hash) as Sales, avg(eth_price) from pricedata
group by event_date
order by 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"

```
alter table pricedata
add column summary varchar(255);

update pricedata set
summary = concat(
                    name,
                    ' was sold for $',
                    floor((usd_price + 99) / 1000) * 1000,
                    ' to ',
                    buyer_address,
                    ' from ',
                    seller_address,
                    ' on ',
                    date_format(event_date, '%Y-%m-%d')
               );

select 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';
select * from 1919_purchases;
```

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

```
select round(eth_price  / 100) * 100 as Eth_price_range, count(*) as frequency ,
Rpad('',count(*), '*') AS Bar_Graph
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.

```
Select name, max(eth_price) as price, 'Highest' as status
from pricedata
group by name
union all
select name, min(eth_price) as price, 'Lowest' as status
from pricedata
group by name
order by name asc, status asc;
```

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
        year(event_date) as Sale_Year,
        month(event_date) as Sale_Month,
        name,
        max(usd_price)
from pricedata
group by Sale_Year, Sale_Month, name;
```

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

```
select
        year(event_date) as year,
        month(event_date) as month,
        round(sum(usd_price)  / 100) * 100 as sum_of_sales
from pricedata
group by month, year
order by year asc , month asc;
```

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

```
select 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.

A.

```
create temporary table temp_table as
select event_date,
        usd_price,
        avg(usd_price) over(partition by event_date ) as avg_usd_price
from pricedata;
```

B.

```
select * from temp_table
where usd_price >= 0.1 * avg_usd_price
```