ELEVATE LABS DATA ANALYST INTERN SQL PROJECT DAY 3 ON 10th APRIL 2025

Introduction to SQL Customer Order Analysis

The following set of SQL queries is designed to analyze customer behavior and order trends over time using data from a retail or e-commerce platform. These queries leverage MS SQL Server to extract meaningful insights from customer and order datasets.

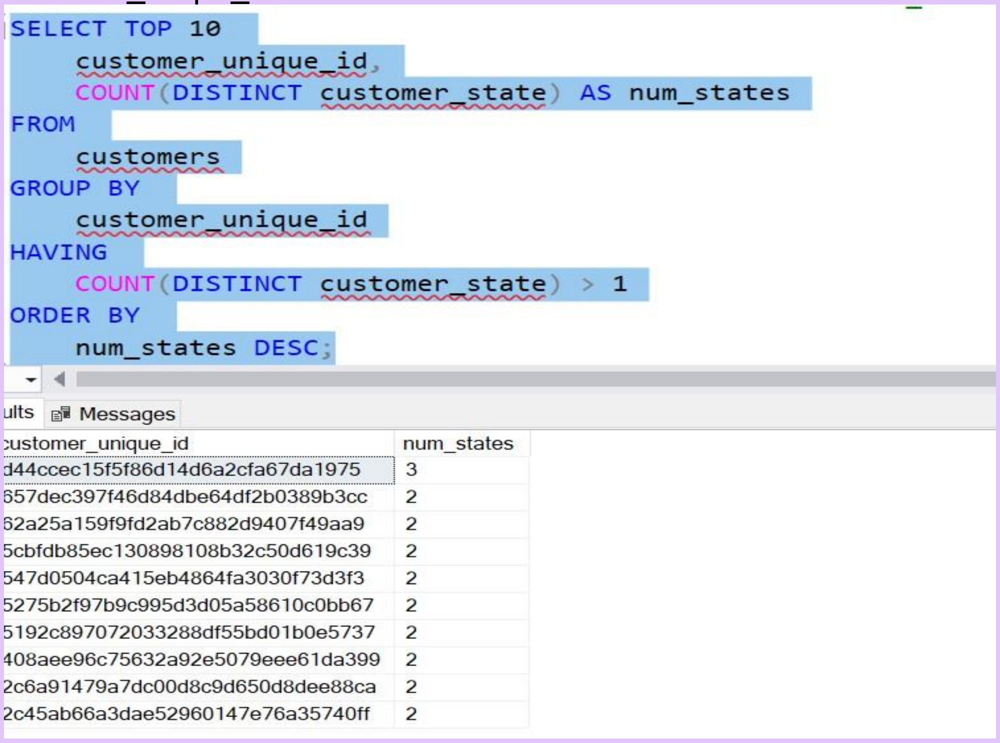
The key objectives of the analysis are:

- Identify Repeat Customers: Determine how many customers return to place more than one order in a given year, and calculate the repeat customer rate.
- Track New Customer Growth: Identify the first purchase year of each customer and evaluate year-over-year growth in new customer acquisition.
- Order Volume Trends: Analyze the total number of unique orders per year, helping to visualize overall business growth.
- Customer Order Behavior: Calculate the median number of orders per customer per year to understand typical customer activity while reducing the impact of outliers.

Customer growth rate and repeating rate

SQL queries

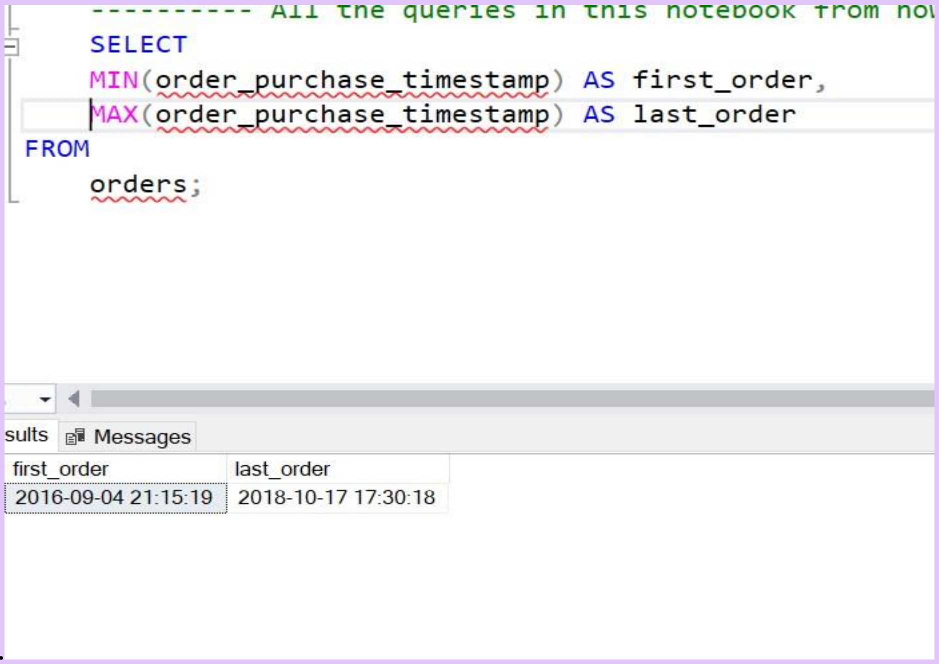
I would like to know if values in the customer_state column change within a certain customer_unique_id.



```sql
SELECT TOP 10
    customer_unique_id,
    COUNT(DISTINCT customer_state) AS num_states
FROM
    customers
GROUP BY
    customer_unique_id
HAVING
    COUNT(DISTINCT customer_state) > 1
ORDER BY
    num_states DESC;
```

customer_unique_id	num_states
d44ccec15f5f86d14d6a2cfa67da1975	3
657dec397f46d84dbe64df2b0389b3cc	2
62a25a159f9fd2ab7c882d9407f49aa9	2
5cbfdb85ec130898108b32c50d619c39	2
547d0504ca415eb4864fa3030f73d3f3	2
5275b2f97b9c995d3d05a58610c0bb67	2
5192c897072033288df55bd01b0e5737	2
408aee96c75632a92e5079eee61da399	2
2c6a91479a7dc00d8c9d650d8dee88ca	2
2c45ab66a3dae52960147e76a35740ff	2

There are customers who have more than one state addresses, which means we cannot use customer_state values to analyze the number of customers by state

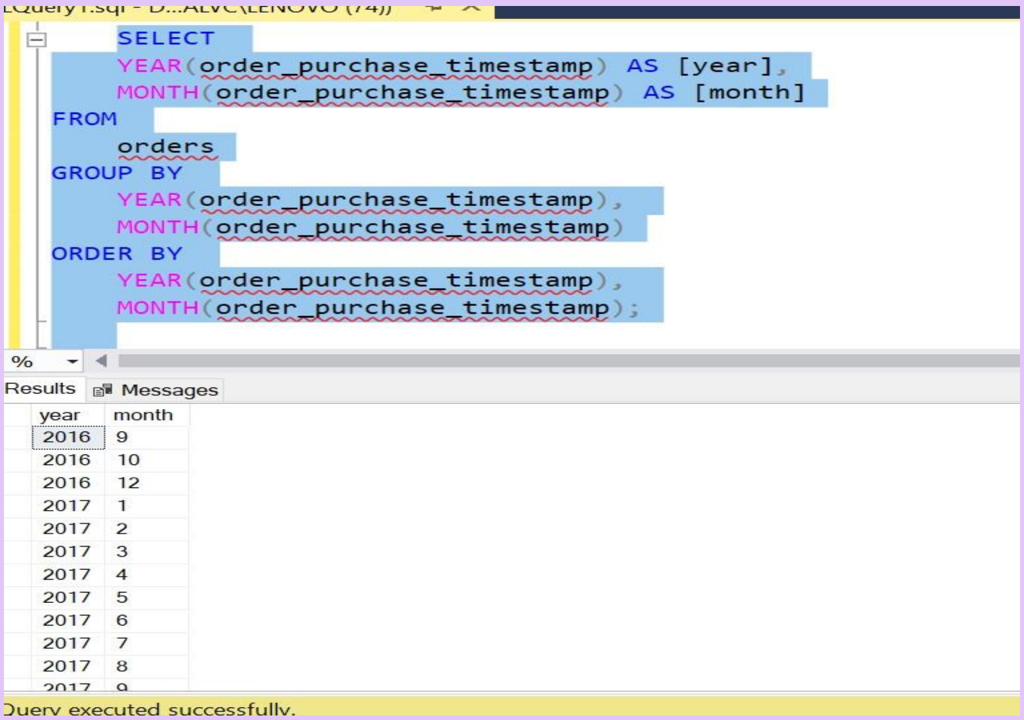


```
---------- All the queries in this notebook from no
    SELECT
    MIN(order_purchase_timestamp) AS first_order,
    MAX(order_purchase_timestamp) AS last_order
FROM
    orders;
```

```
first_order           last_order
2016-09-04 21:15:19   2018-10-17 17:30:18
```

.

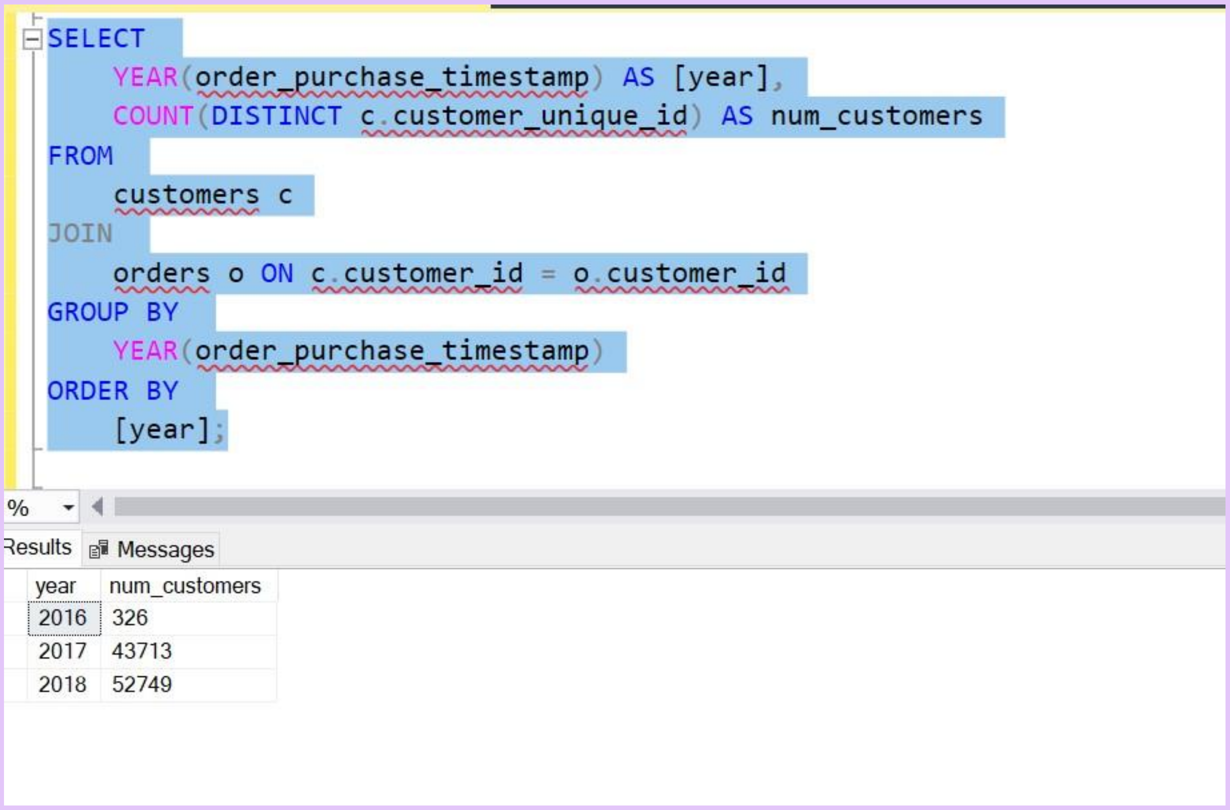
The first order was placed in the beginning of September 2019, and the last order was placed in the middle of October 2018.



```
    SELECT
    YEAR(order_purchase_timestamp) AS [year],
    MONTH(order_purchase_timestamp) AS [month]
FROM
    orders
GROUP BY
    YEAR(order_purchase_timestamp),
    MONTH(order_purchase_timestamp)
ORDER BY
    YEAR(order_purchase_timestamp),
    MONTH(order_purchase_timestamp);
```

```
year   month
2016   9
2016   10
2016   12
2017   1
2017   2
2017   3
2017   4
2017   5
2017   6
2017   7
2017   8
2017   9
Query executed successfully.
```

Number of customers and customers growth rate
Assuming that started in 2016, I would like to know the number of new customers increasing each year.

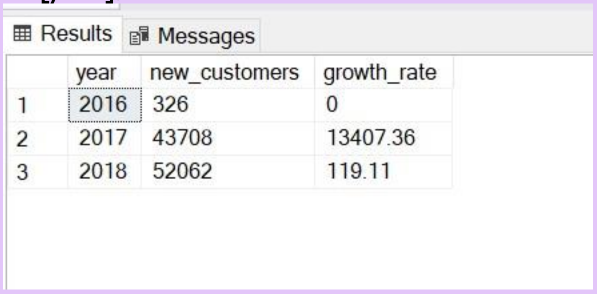


```
SELECT
    YEAR(order_purchase_timestamp) AS [year],
    COUNT(DISTINCT c.customer_unique_id) AS num_customers
FROM
    customers c
JOIN
    orders o ON c.customer_id = o.customer_id
GROUP BY
    YEAR(order_purchase_timestamp)
ORDER BY
    [year];
```

```
%
Results   Messages
   year   num_customers
   2016   326
   2017   43713
   2018   52749
```


```
WITH first_purchase_year AS (
    SELECT
        c.customer_unique_id,
        YEAR(MIN(o.order_purchase_timestamp)) AS first_year
    FROM
        customers c
    JOIN
        orders o ON c.customer_id = o.customer_id
    GROUP BY
        c.customer_unique_id
),
new_customers_per_year AS (
    SELECT
        first_year AS [year],
        COUNT(DISTINCT customer_unique_id) AS new_customers
    FROM
        first_purchase_year
    GROUP BY
        first_year
),
yearly_with_growth AS (
    SELECT
```

```
        nc1.[year],
        nc1.new_customers,
        LAG(nc1.new_customers) OVER (ORDER BY nc1.[year]) AS prev_year_customers
    FROM
        new_customers_per_year nc1
)
SELECT
    [year],
    new_customers,
    CASE
        WHEN prev_year_customers IS NULL THEN 0
        ELSE ROUND(CAST(new_customers AS FLOAT) / prev_year_customers * 100, 2)
    END AS growth_rate
FROM
    yearly_with_growth
ORDER BY
    [year];
```

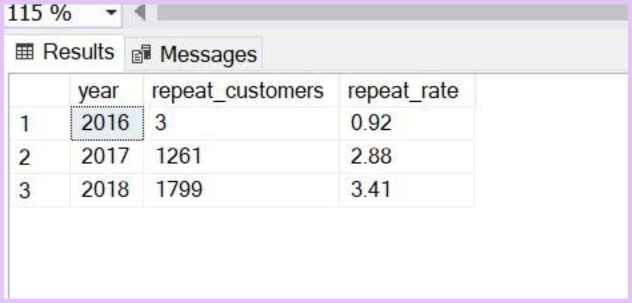


	year	new_customers	growth_rate
1	2016	326	0
2	2017	43708	13407.36
3	2018	52062	119.11

Customers repeating rate
customers repeating rate.

```
WITH customer_orders AS (
    SELECT
        c.customer_unique_id,
        YEAR(o.order_purchase_timestamp) AS [year],
        ROW_NUMBER() OVER (
            PARTITION BY c.customer_unique_id
            ORDER BY o.order_purchase_timestamp
        ) AS rn
    FROM
        customers c
    JOIN
        orders o ON c.customer_id = o.customer_id
),
repeat_customers_by_year AS (
    SELECT
        [year],
        COUNT(DISTINCT customer_unique_id) AS repeat_customers
    FROM
```

```
        customer_orders
    WHERE
        rn > 1  -- return customers (not their first purchase)
    GROUP BY
        [year]
),
total_customers_by_year AS (
    SELECT
        YEAR(o.order_purchase_timestamp) AS [year],
        COUNT(DISTINCT c.customer_unique_id) AS num_customers
    FROM
        customers c
    JOIN
        orders o ON c.customer_id = o.customer_id
    GROUP BY
        YEAR(o.order_purchase_timestamp)
)
SELECT
    t.[year],
    r.repeat_customers,
    ROUND(CAST(r.repeat_customers AS FLOAT) / t.num_customers * 100, 2) AS
repeat_rate
FROM
    total_customers_by_year t
JOIN
    repeat_customers_by_year r ON t.[year] = r.[year]
ORDER BY
    t.[year];
```

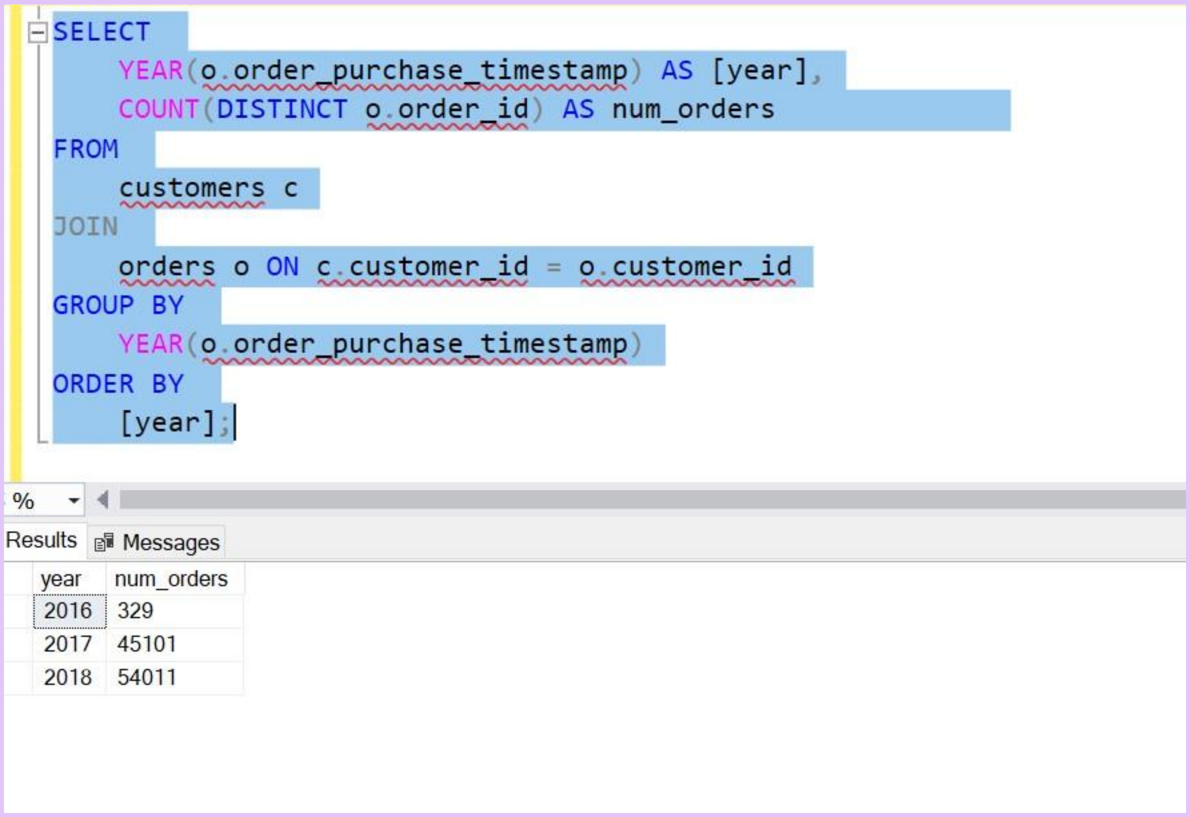


115 %

Results | Messages

	year	repeat_customers	repeat_rate
1	2016	3	0.92
2	2017	1261	2.88
3	2018	1799	3.41

Number of orders and median number of orders
I would like to know the total number of orders placed each year and the median number of orders placed by unique customers each year.

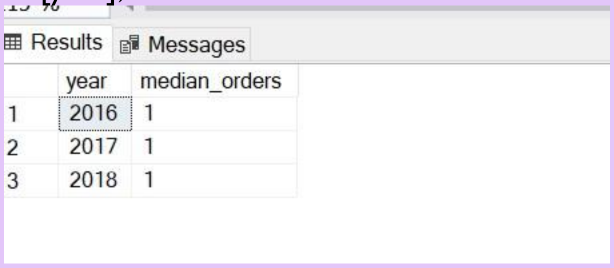


```
SELECT
    YEAR(o.order_purchase_timestamp) AS [year],
    COUNT(DISTINCT o.order_id) AS num_orders
FROM
    customers c
JOIN
    orders o ON c.customer_id = o.customer_id
GROUP BY
    YEAR(o.order_purchase_timestamp)
ORDER BY
    [year];
```

Results Messages

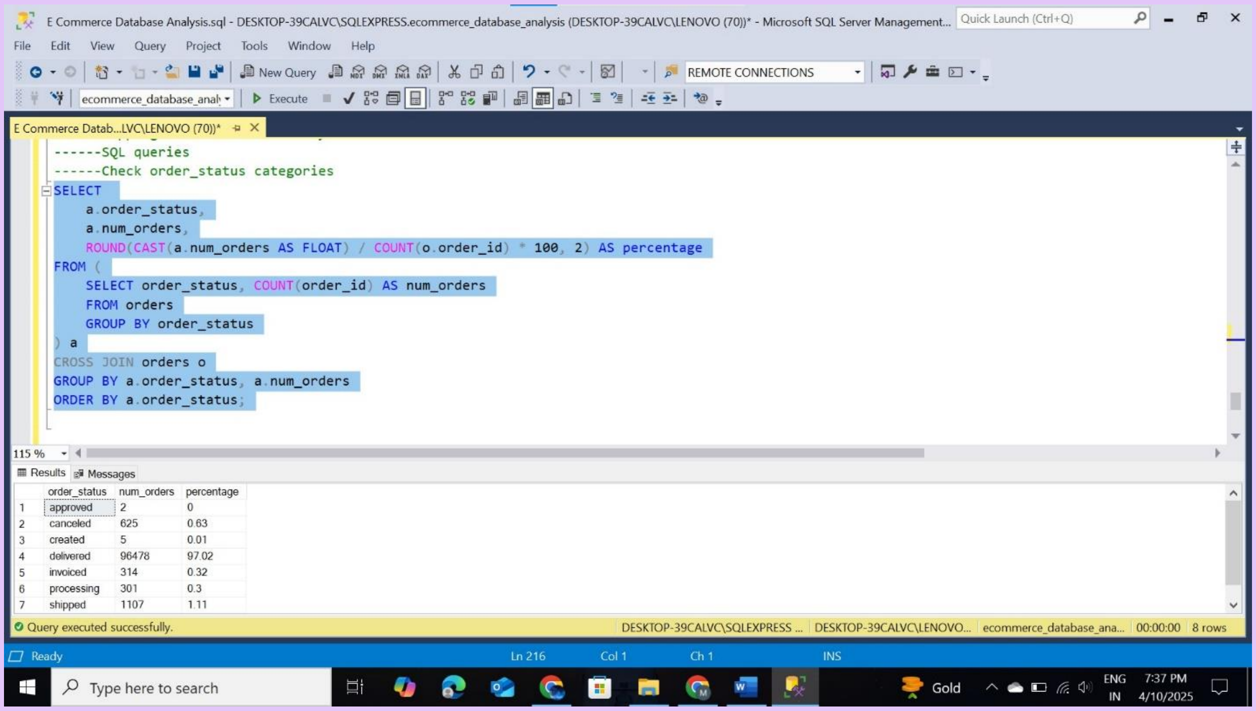
```
year  num_orders
2016  329
2017  45101
2018  54011
```

```
--------------------------------------------------------------------------------------------------------------------------
-------
WITH customer_orders_per_year AS (
    SELECT
        YEAR(o.order_purchase_timestamp) AS [year],
        c.customer_unique_id,
        COUNT(o.order_id) AS num_orders
    FROM
        customers c
    JOIN
        orders o ON c.customer_id = o.customer_id
    GROUP BY
        YEAR(o.order_purchase_timestamp), c.customer_unique_id
),
ranked_orders AS (
    SELECT
        [year],
        num_orders,
        ROW_NUMBER() OVER (PARTITION BY [year] ORDER BY num_orders) AS rn,
        COUNT(*) OVER (PARTITION BY [year]) AS total_customers
    FROM
        customer_orders_per_year
),
median_orders AS (
    SELECT
        [year],
        AVG(CAST(num_orders AS FLOAT)) AS median_orders
    FROM
        ranked_orders
    WHERE
        rn = (total_customers + 1) / 2 OR
        rn = (total_customers + 2) / 2
    GROUP BY
        [year]
)
SELECT
    [year],
    ROUND(median_orders, 0) AS median_orders
FROM
    median_orders
ORDER BY
    [year];
```



```
   year  median_orders
1  2016  1
2  2017  1
3  2018  1
```

Shipping and on-time delivery rate
SQL queries
Check order_status categories

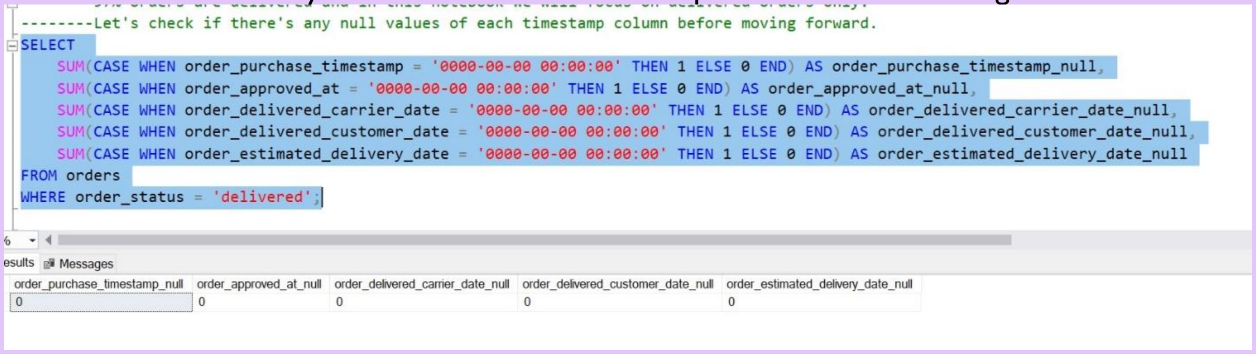


```
------SQL queries
------Check order_status categories
SELECT
    a.order_status,
    a.num_orders,
    ROUND(CAST(a.num_orders AS FLOAT) / COUNT(o.order_id) * 100, 2) AS percentage
FROM (
    SELECT order_status, COUNT(order_id) AS num_orders
    FROM orders
    GROUP BY order_status
) a
CROSS JOIN orders o
GROUP BY a.order_status, a.num_orders
ORDER BY a.order_status;
```

	order_status	num_orders	percentage
1	approved	2	0
2	canceled	625	0.63
3	created	5	0.01
4	delivered	96478	97.02
5	invoiced	314	0.32
6	processing	301	0.3
7	shipped	1107	1.11

Query executed successfully.

97% orders are delivered and in this notebook we will focus on delivered orders only.
Let's check if there's any null values of each timestamp column before moving forward.

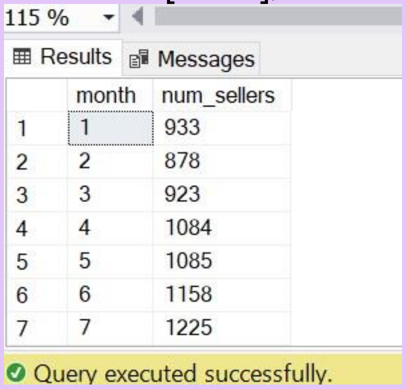


```
--------Let's check if there's any null values of each timestamp column before moving forward.
SELECT
    SUM(CASE WHEN order_purchase_timestamp = '0000-00-00 00:00:00' THEN 1 ELSE 0 END) AS order_purchase_timestamp_null,
    SUM(CASE WHEN order_approved_at = '0000-00-00 00:00:00' THEN 1 ELSE 0 END) AS order_approved_at_null,
    SUM(CASE WHEN order_delivered_carrier_date = '0000-00-00 00:00:00' THEN 1 ELSE 0 END) AS order_delivered_carrier_date_null,
    SUM(CASE WHEN order_delivered_customer_date = '0000-00-00 00:00:00' THEN 1 ELSE 0 END) AS order_delivered_customer_date_null,
    SUM(CASE WHEN order_estimated_delivery_date = '0000-00-00 00:00:00' THEN 1 ELSE 0 END) AS order_estimated_delivery_date_null
FROM orders
WHERE order_status = 'delivered';
```

order_purchase_timestamp_null	order_approved_at_null	order_delivered_carrier_date_null	order_delivered_customer_date_null	order_estimated_delivery_date_null
0	0	0	0	0

On-time delivery rate
requires merchants to maintain their on-time delivery rate at least 96% for the last 30 days to guarantee their performance.
I would like to know the number of sellers who have 96% or higher on-time delivery rate lately.
First,
let's check if there are any sellers who delivered orders on time each year in 2018.

```
SELECT
    MONTH(order_purchase_timestamp) AS [month],
    COUNT(DISTINCT seller_id) AS num_sellers
FROM (
    SELECT
        oi.seller_id,
        o.order_purchase_timestamp,
        o.order_delivered_customer_date,
        o.order_estimated_delivery_date
    FROM orders o
    JOIN order_items oi ON o.order_id = oi.order_id
    WHERE o.order_status = 'delivered'
      AND YEAR(o.order_purchase_timestamp) = 2018
      AND o.order_delivered_customer_date IS NOT NULL
      AND DATEDIFF(DAY, o.order_delivered_customer_date,
o.order_estimated_delivery_date) >= 0
) a
GROUP BY MONTH(order_purchase_timestamp)
ORDER BY [month];
```



115 %

Results | Messages

	month	num_sellers
1	1	933
2	2	878
3	3	923
4	4	1084
5	5	1085
6	6	1158
7	7	1225

Query executed successfully.

How fast is it for a customer to receive an order?

This query calculates the percentage of orders arriving within 2 days, 1 week, 2 weeks, or more than 2 weeks after they are placed

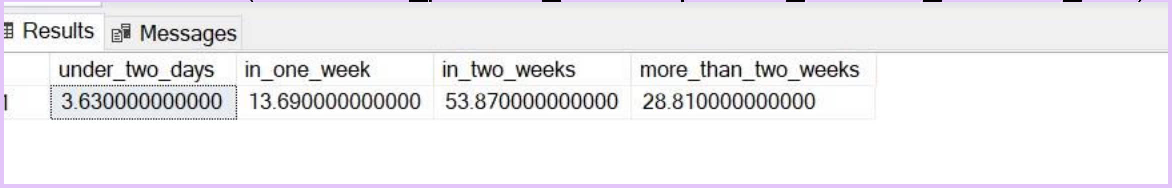
```
SELECT
    ROUND(SUM(CASE
               WHEN DATEDIFF(DAY, order_purchase_timestamp,
order_delivered_customer_date) <= 2
                     THEN 1 ELSE 0
             END) * 100.0 / COUNT(order_id), 2) AS under_two_days,

    ROUND(SUM(CASE
               WHEN DATEDIFF(DAY, order_purchase_timestamp,
order_delivered_customer_date) BETWEEN 3 AND 5
                     THEN 1 ELSE 0
             END) * 100.0 / COUNT(order_id), 2) AS in_one_week,

    ROUND(SUM(CASE
               WHEN DATEDIFF(DAY, order_purchase_timestamp,
order_delivered_customer_date) BETWEEN 6 AND 14
                     THEN 1 ELSE 0
             END) * 100.0 / COUNT(order_id), 2) AS in_two_weeks,

    ROUND(SUM(CASE
               WHEN DATEDIFF(DAY, order_purchase_timestamp,
order_delivered_customer_date) > 14
                     THEN 1 ELSE 0
```

```
            END) * 100.0 / COUNT(order_id), 2) AS more_than_two_weeks
FROM orders
WHERE order_status = 'delivered'
  AND order_delivered_customer_date IS NOT NULL
  AND DATEDIFF(DAY, order_purchase_timestamp, order_delivered_customer_date) >= 0;
```



Results | Messages

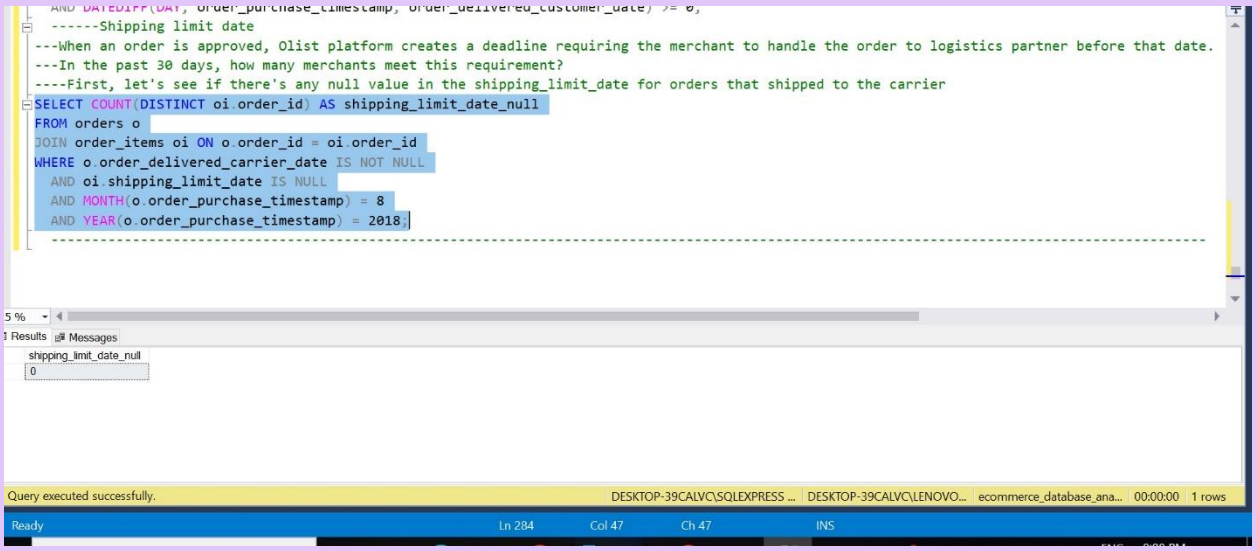
	under_two_days	in_one_week	in_two_weeks	more_than_two_weeks
1	3.630000000000	13.690000000000	53.870000000000	28.810000000000

Shipping limit date

When an order is approved, Olist platform creates a deadline requiring the merchant to handle the order to logistics partner before that date.

In the past 30 days, how many merchants meet this requirement?

First, let's see if there's any null value in the shipping_limit_date for orders that shipped to the carrier



```
  ------Shipping limit date
---When an order is approved, Olist platform creates a deadline requiring the merchant to handle the order to logistics partner before that date.
---In the past 30 days, how many merchants meet this requirement?
----First, let's see if there's any null value in the shipping_limit_date for orders that shipped to the carrier
SELECT COUNT(DISTINCT oi.order_id) AS shipping_limit_date_null
FROM orders o
JOIN order_items oi ON o.order_id = oi.order_id
WHERE o.order_delivered_carrier_date IS NOT NULL
  AND oi.shipping_limit_date IS NULL
  AND MONTH(o.order_purchase_timestamp) = 8
  AND YEAR(o.order_purchase_timestamp) = 2018;
  -------------------------------------------------------------------------------------------------------------------------------------------------------
```

Results | Messages

	shipping_limit_date_null
	0

```
Query executed successfully.    DESKTOP-39CALVC\SQLEXPRESS ...  DESKTOP-39CALVC\LENOVO...  ecommerce_database_ana...  00:00:00  1 rows
Ready    Ln 284    Col 47    Ch 47    INS
```

THANK YOU BY DURGAM MANOHAR

GIT HUB: https://github.com/manohar1054/ELEVATE-LABS-DATA-ANALYST-INTERN-

LINKED IN: www.linkedin.com/in/manohar-138091255

PORTFOLIO: https://manohardurgam10543.wixsite.com/my-site