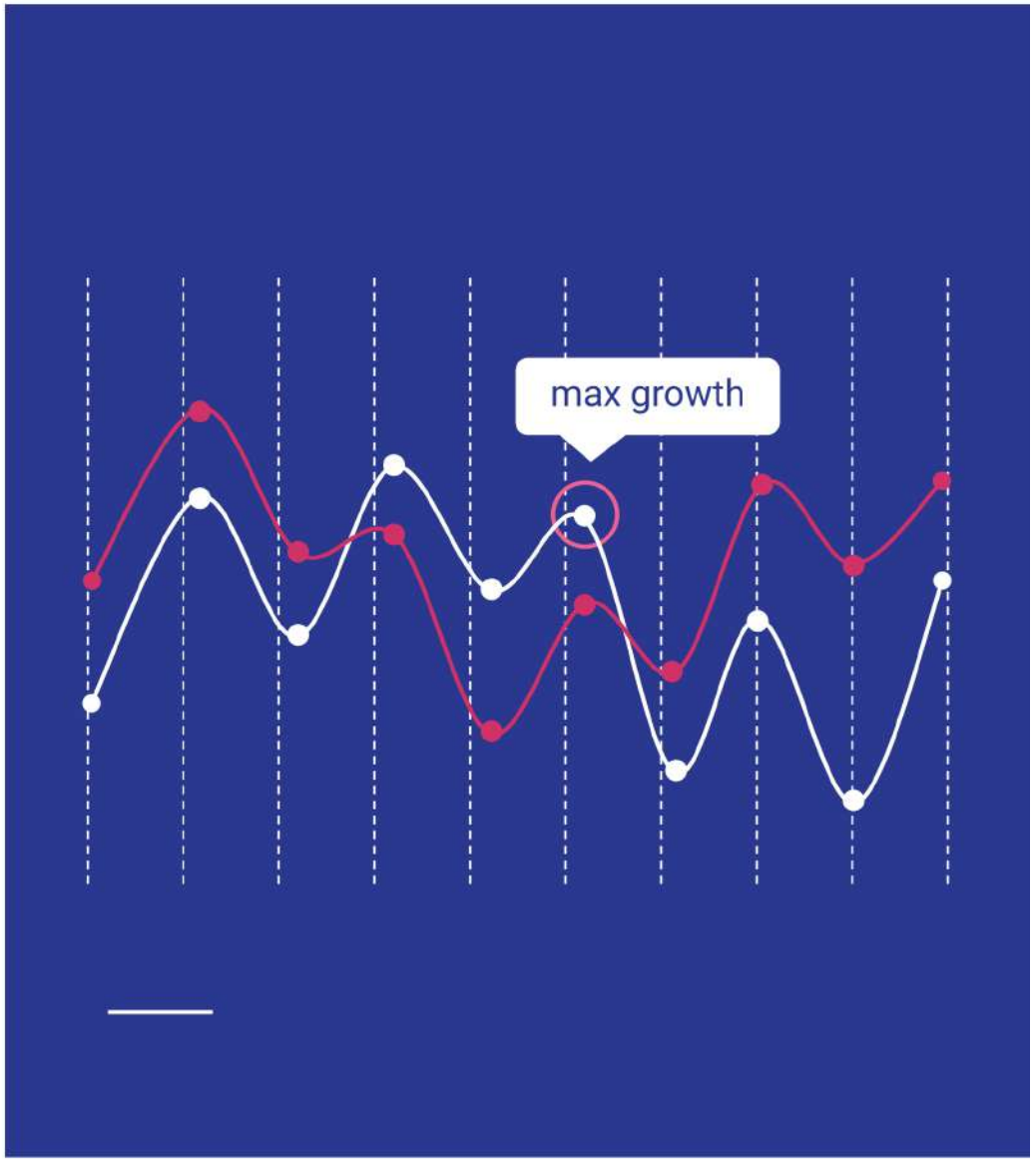
Project by

Md Rafi Islam

Contact

Email: rakars23@gmail.com

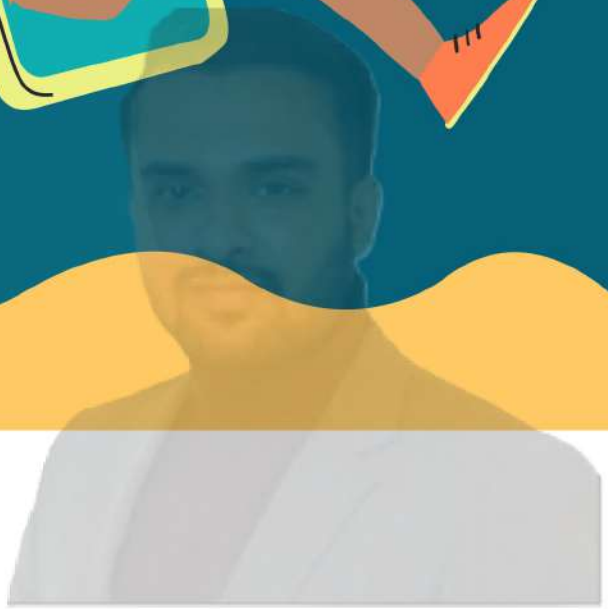
Github



mercado livre

E-COMMERCE GROWTH ANALYSIS

Prepared by - Md Rafi Islam



ABOUT MERCADO LIVRE

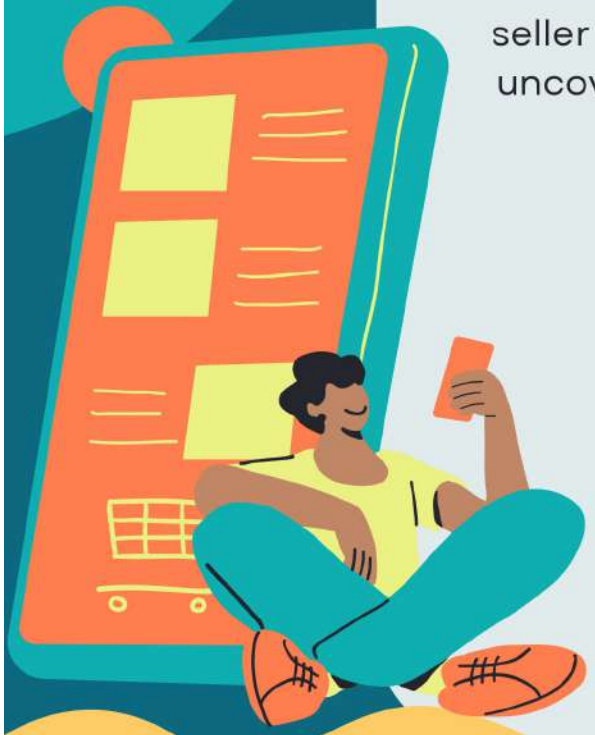
MercadoLibre, Inc in Spanish, and known as Mercado Livre in Portuguese.

Mercado Livre is an Argentine company headquartered in Montevideo, Uruguay and incorporated in Delaware in the United States that operates online marketplaces dedicated to e-commerce and online auctions.

Mercado Libre also runs a real estate and motors division under the name "*Mercado Libre Classificados*". Realtors pay a monthly fee to list properties and automobiles on the Mercado Libre platform.

PROBLEM STATEMENT

Mercado Livre, a leading Brazilian e-commerce platform, aims to optimize its marketplace operations by gaining deeper insights into customer behavior, seller performance, product trends, and fulfillment efficiency. The goal is to uncover growth opportunities, improve customer experience, and forecast future sales trends to support data-driven decision-making.



STEPS INVOLVED:

PART 1: SQL ANALYSIS

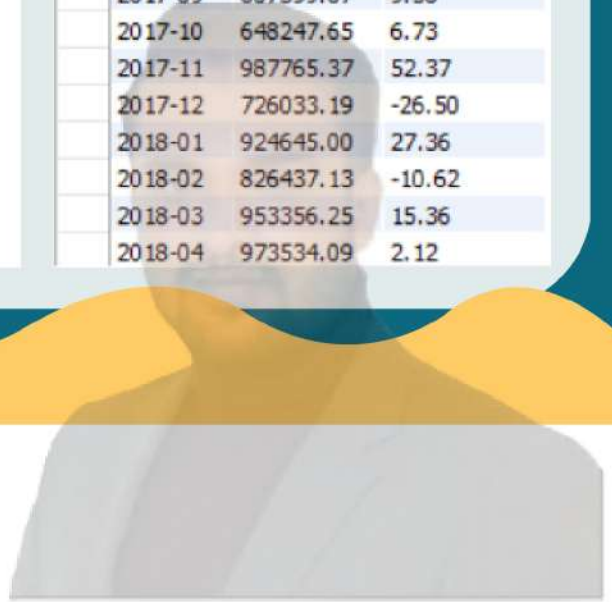
PART 2: DASHBOARD CREATION - POWER BI

PART 3: PYTHON – EDA & TREND ANALYSIS

FOR EACH MONTH, CALCULATE TOTAL REVENUE AND MONTH-OVER-MONTH GROWTH PERCENTAGE.

```sql
SELECT
  DATE_FORMAT(o.order_purchase_timestamp, '%Y-%m') AS month,
  SUM(oi.price) AS revenue,
  ROUND(
    (SUM(oi.price) - LAG(SUM(oi.price))
    OVER (ORDER BY DATE_FORMAT(o.order_purchase_timestamp, '%Y-%m')))
    / NULLIF(LAG(SUM(oi.price))
    OVER (ORDER BY DATE_FORMAT(o.order_purchase_timestamp, '%Y-%m')), 0) * 100, 2
  ) AS growth
FROM
  order_items_dataset oi
JOIN
  orders_dataset o ON o.order_id = oi.order_id
WHERE
  o.order_status = 'delivered'
GROUP BY
  month
ORDER BY
  month;
```

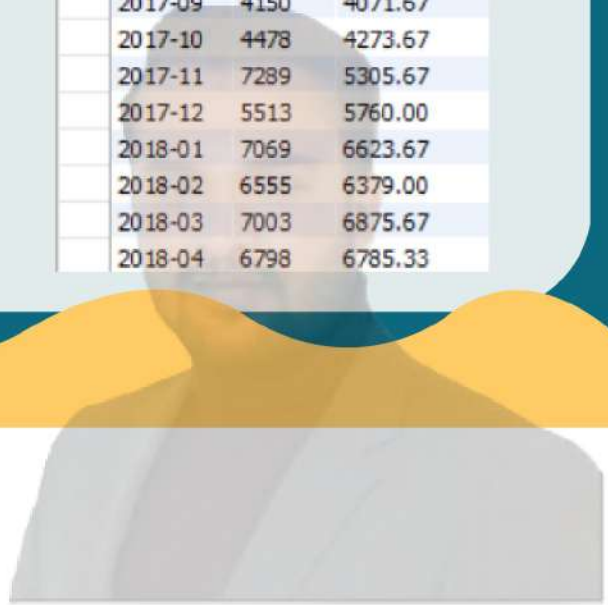

month	revenue	growth
2016-09	134.97	NULL
2016-10	40325.11	29777.09
2016-12	10.90	-99.97
2017-01	111798.36	1025573.03
2017-02	234223.40	109.51
2017-03	359198.85	53.36
2017-04	340669.68	-5.16
2017-05	489338.25	43.64
2017-06	421923.37	-13.78
2017-07	481604.52	14.15
2017-08	554699.70	15.18
2017-09	607399.67	9.50
2017-10	648247.65	6.73
2017-11	987765.37	52.37
2017-12	726033.19	-26.50
2018-01	924645.00	27.36
2018-02	826437.13	-10.62
2018-03	953356.25	15.36
2018-04	973534.09	2.12



FOR EACH MONTH, CALCULATE THE 3-MONTH MOVING AVERAGE OF THE NUMBER OF ORDERS.

```sql
SELECT
  month,
  orders,
  ROUND(AVG(orders) OVER (ORDER BY month ROWS 2 PRECEDING), 2) AS moving_avg
FROM (
  SELECT
    DATE_FORMAT(order_purchase_timestamp, '%Y-%m') AS month,
    COUNT(*) AS orders
  FROM
    orders_dataset
  WHERE
    order_status = 'delivered'
  GROUP BY
    month
) AS monthly_orders;
```

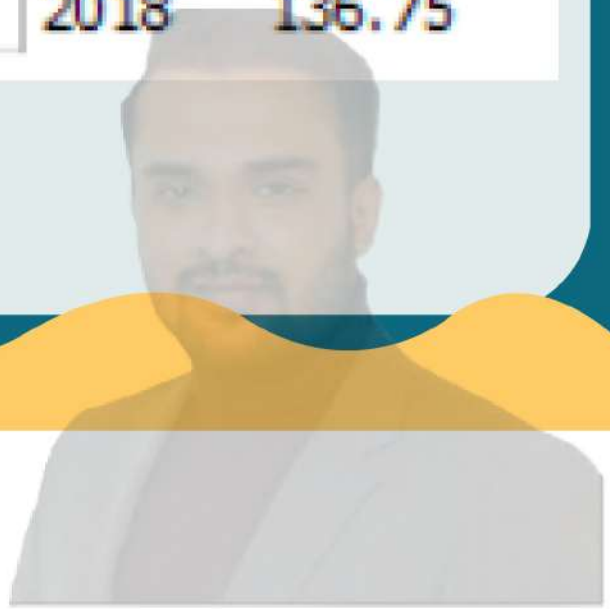

month	orders	moving_avg
2016-09	1	1.00
2016-10	265	133.00
2016-12	1	89.00
2017-01	750	338.67
2017-02	1653	801.33
2017-03	2546	1649.67
2017-04	2303	2167.33
2017-05	3546	2798.33
2017-06	3135	2994.67
2017-07	3872	3517.67
2017-08	4193	3733.33
2017-09	4150	4071.67
2017-10	4478	4273.67
2017-11	7289	5305.67
2017-12	5513	5760.00
2018-01	7069	6623.67
2018-02	6555	6379.00
2018-03	7003	6875.67
2018-04	6798	6785.33



YEARLY AVERAGE ORDER VALUE (AOV): TREND COMPUTE AVERAGE ORDER VALUE PER YEAR.

```sql
SELECT
  YEAR(o.order_purchase_timestamp) AS year,
  ROUND(SUM(oi.price) / COUNT(DISTINCT o.order_id), 2) AS aov
FROM
  order_items_dataset oi
JOIN
  orders_dataset o ON o.order_id = oi.order_id
WHERE
  o.order_status = 'delivered'
GROUP BY
  year
ORDER BY
  year;
```

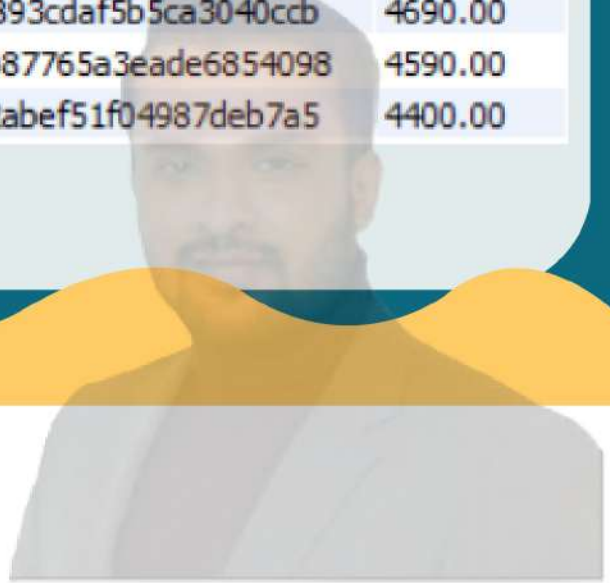

	year	aov
▶	2016	151.58
	2017	137.31
	2018	136.75



CALCULATE TOTAL REVENUE GENERATED BY EACH CUSTOMER. SORT TOP 10 HIGHEST LIFETIME VALUES.

```sql
SELECT
  c.customer_unique_id,
  ROUND(SUM(oi.price), 2) AS cltv
FROM
  customers_dataset c
JOIN
  orders_dataset o ON o.customer_id = c.customer_id
JOIN
  order_items_dataset oi ON oi.order_id = o.order_id
WHERE
  o.order_status = 'delivered'
GROUP BY
  c.customer_unique_id
ORDER BY
  cltv DESC
LIMIT 10;
```

customer_unique_id	cltv
0a0a92112bd4c708ca5fde585afaa872	13440.00
da122df9eeddfedc1dc1f5349a1a690c	7388.00
763c8b1c9c68a0229c42c9fc6f662b93	7160.00
dc4802a71eae9be1dd28f5d788ceb526	6735.00
459bef486812aa25204be022145caa62	6729.00
ff4159b92c40ebe40454e3e6a7c35ed6	6499.00
4007669dec559734d6f53e029e360987	5934.60
eebb5dda148d3893cdaf5b5ca3040ccb	4690.00
48e1ac109decbb87765a3eade6854098	4590.00
a229eba70ec1c2abef51f04987deb7a5	4400.00



IDENTIFY TOP 5 CATEGORIES WITH THE HIGHEST YEAR-OVER-YEAR REVENUE GROWTH.

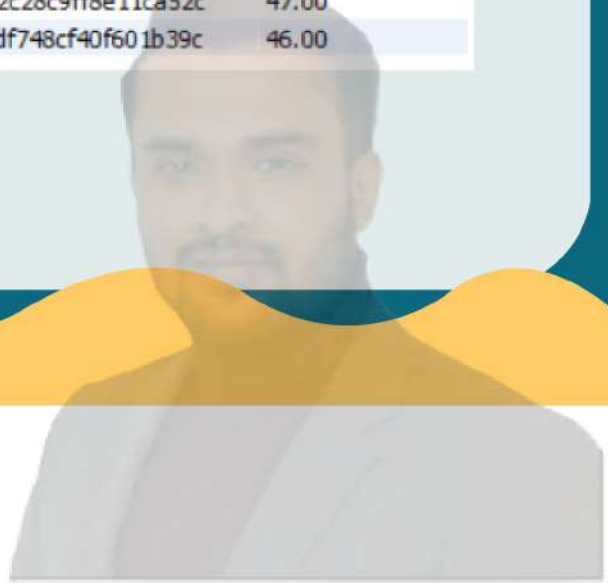
```sql
SELECT category, year,
ROUND(((revenue - prev_revenue) / NULLIF(prev_revenue, 0)) * 100, 2) AS growth
FROM (
  SELECT
    pc.product_category_name_english AS category,
    YEAR(o.order_purchase_timestamp) AS year,
    SUM(oi.price) AS revenue, LAG(SUM(oi.price))
    OVER (PARTITION BY pc.product_category_name_english ORDER BY YEAR(o.order_purchase_timestamp))
    AS prev_revenue
  FROM
    order_items_dataset oi
  JOIN
    orders_dataset o ON o.order_id = oi.order_id
  JOIN
    products_dataset p ON p.product_id = oi.product_id
  LEFT JOIN
    product_category_name_translati pc ON pc.product_category_name = p.product_category_name
  WHERE
    o.order_status = 'delivered'
  GROUP BY category, year
) AS t
WHERE prev_revenue IS NOT NULL
ORDER BY growth DESC LIMIT 5;
```


category	year	growth
NULL	2017	152558.61
bed_bath_table	2017	102323.21
telephony	2017	59257.95
computers_accessories	2017	58461.22
electronics	2017	51016.16

FOR EACH SELLER, CALCULATE THE AVERAGE DIFFERENCE BETWEEN ESTIMATED AND ACTUAL DELIVERY DATE. HIGHLIGHT THE MOST EFFICIENT ONES.

```sql
SELECT
  oi.seller_id,
  ROUND(AVG(DATEDIFF(o.order_estimated_delivery_date,
   o.order_delivered_customer_date)),2)
  AS avg_days_early
FROM
  order_items_dataset oi
JOIN
  orders_dataset o ON o.order_id = oi.order_id
WHERE
  o.order_status = 'delivered'
  AND o.order_delivered_customer_date IS NOT NULL
  AND o.order_estimated_delivery_date IS NOT NULL
GROUP BY
  oi.seller_id
ORDER BY
  avg_days_early DESC
LIMIT 10;
```

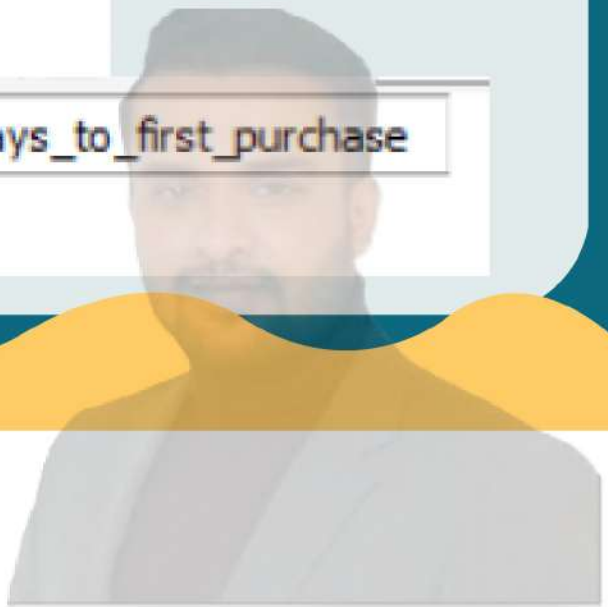
seller_id	avg_days_early
933446e9a59dece7ae9175103820ca8f	66.00
0b09101900100c0e9d312861fad5a1b9	61.00
fa5fdc4e4bb6bd1009ad0e4ac4096562	58.00
58e4b302b54937e55a678c4d15111da4	48.00
432c67955c0acd1fd6b0b5d678766a71	48.00
939f6e231201f26803cb5c3a3d2940b3	48.00
ffff564a4f9085cd26170f4732393726	48.00
4bde6149c15cf7e177b36fa060dd6de8	47.50
ae9690c6e8fee182c28c9ff8e11ca52c	47.00
f5b84683a9bf9e1df748cf40f601b39c	46.00



FOR CUSTOMERS WITH MULTIPLE ORDERS, CALCULATE THE AVERAGE NUMBER OF DAYS BETWEEN ACCOUNT CREATION AND FIRST PURCHASE.

```sql
WITH ranked_orders AS (
SELECT c.customer_unique_id, o.order_purchase_timestamp,
ROW_NUMBER() OVER (PARTITION BY c.customer_unique_id ORDER BY o.order_purchase_timestamp) AS rn
  FROM customers_dataset c
  JOIN orders_dataset o ON o.customer_id = c.customer_id
  WHERE o.order_status = 'delivered'),
first_second_orders AS (
  SELECT
    customer_unique_id,
    MAX(CASE WHEN rn = 1 THEN order_purchase_timestamp END) AS first_seen,
    MAX(CASE WHEN rn = 2 THEN order_purchase_timestamp END) AS first_purchase
  FROM ranked_orders GROUP BY customer_unique_id HAVING COUNT(*) >= 2)
SELECT
  ROUND(AVG(DATEDIFF(first_purchase, first_seen)), 2) AS avg_days_to_first_purchase
FROM first_second_orders;
```

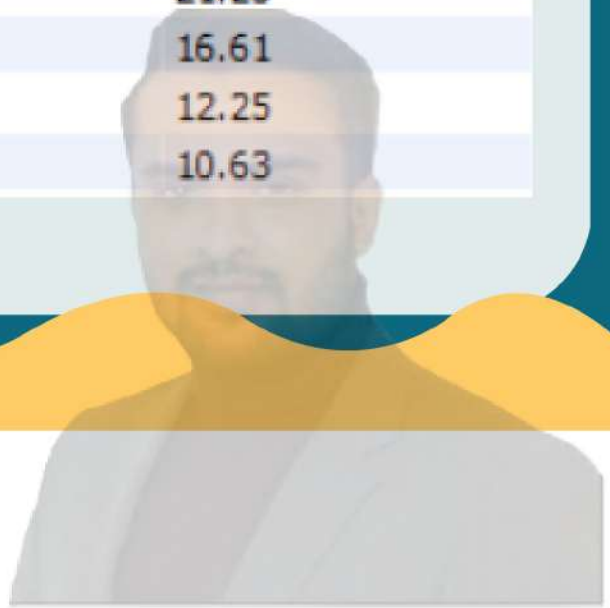

	avg_days_to_first_purchase
▶	81.20



COMPARE AVERAGE DELIVERY DAYS FOR ORDERS WITH 1-2 STARS VS 4-5 STARS.

```sql
SELECT
  r.review_score,
  ROUND(AVG(DATEDIFF(o.order_delivered_customer_date, o.order_purchase_timestamp)), 2) AS avg_delivery_days
FROM
  order_reviews_dataset r
JOIN
  orders_dataset o ON o.order_id = r.order_id
WHERE
  r.review_score IN (1, 2, 4, 5)
  AND o.order_status = 'delivered'
  AND o.order_delivered_customer_date IS NOT NULL
GROUP BY
  r.review_score
ORDER BY
  r.review_score;
```

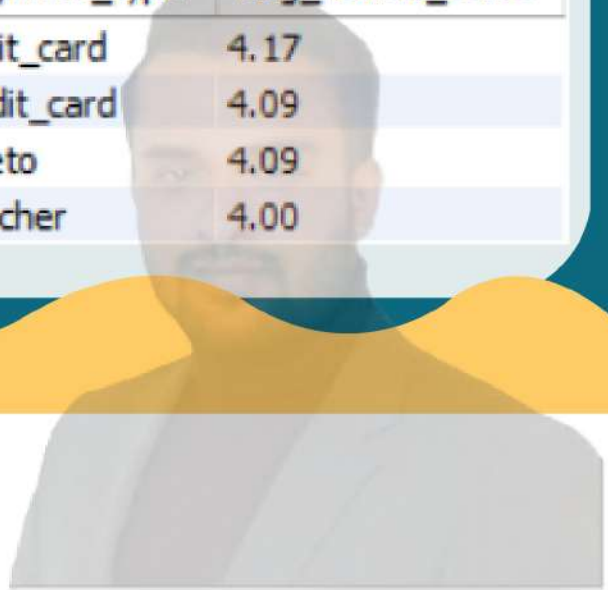

review_score	avg_delivery_days
1	21.25
2	16.61
4	12.25
5	10.63



ANALYZE WHICH PAYMENT METHODS LEAD TO THE HIGHEST AVERAGE REVIEW SCORES.

```sql
SELECT
  p.payment_type,
  ROUND(AVG(r.review_score), 2) AS avg_review_score
FROM
  order_payments_dataset p
JOIN
  order_reviews_dataset r ON p.order_id = r.order_id
GROUP BY
  p.payment_type
ORDER BY
  avg_review_score DESC;
```

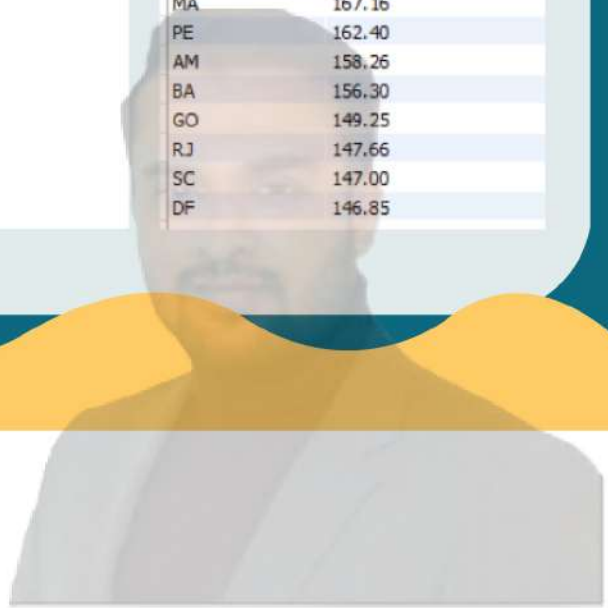

payment_type	avg_review_score
debit_card	4.17
credit_card	4.09
boleto	4.09
voucher	4.00



FOR EACH STATE, COMPUTE AVERAGE REVENUE PER CUSTOMER.

```sql
SELECT
  c.customer_state,
  ROUND(SUM(oi.price) / COUNT(DISTINCT c.customer_unique_id), 2) AS revenue_per_customer
FROM
  customers_dataset c
JOIN
  orders_dataset o ON o.customer_id = c.customer_id
JOIN
  order_items_dataset oi ON oi.order_id = o.order_id
WHERE
  o.order_status = 'delivered'
GROUP BY
  c.customer_state
ORDER BY
  revenue_per_customer DESC;
```

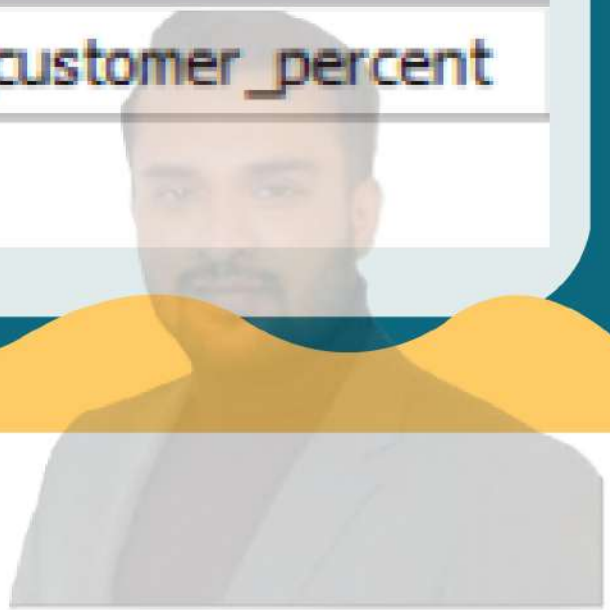

customer_state	revenue_per_customer
PB	223.39
AC	209.62
AL	203.76
AP	202.65
RO	197.76
PA	189.23
PI	182.59
TO	181.28
MT	177.79
RN	176.95
RR	176.44
CE	174.69
SE	171.96
MS	169.50
MA	167.16
PE	162.40
AM	158.26
BA	156.30
GO	149.25
RJ	147.66
SC	147.00
DF	146.85



WHAT PERCENTAGE OF CUSTOMERS PLACED MORE THAN ONE ORDER?

```sql
SELECT
  ROUND(COUNT(*) * 100.0 / (SELECT COUNT(DISTINCT customer_unique_id)
  FROM customers_dataset), 2) AS repeat_customer_percent
FROM (
  SELECT
    c.customer_unique_id
  FROM
    customers_dataset c
  JOIN
    orders_dataset o ON o.customer_id = c.customer_id
  WHERE
    o.order_status = 'delivered'
  GROUP BY
    c.customer_unique_id
  HAVING
    COUNT(DISTINCT o.order_id) > 1
) AS repeat_customers;
```

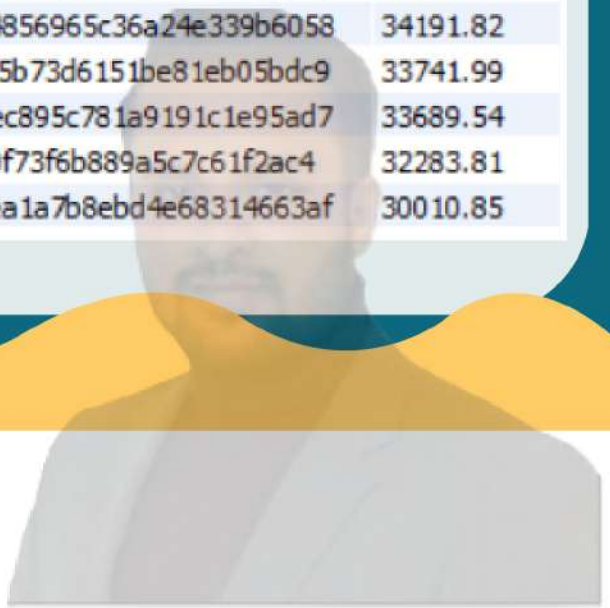

repeat_customer_percent
2.91



CALCULATE PROFIT PER SKU, THEN LIST TOP 10 CONTRIBUTORS.

```sql
SELECT
  oi.product_id,
  ROUND(SUM(oi.price - oi.freight_value), 2) AS total_profit
FROM
  order_items_dataset oi
JOIN
  orders_dataset o ON o.order_id = oi.order_id
WHERE
  o.order_status = 'delivered'
GROUP BY
  oi.product_id
ORDER BY
  total_profit DESC
LIMIT 10;
```


product_id	total_profit
bb50f2e236e5eea0100680137654686c	59861.97
6cdd53843498f92890544667809f1595	49370.87
d6160fb7873f184099d9bc95e30376af	44584.52
25c38557cf793876c5abdd5931f922db	37502.69
53b36df67ebb7c41585e8d54d6772e08	35195.77
99a4788cb24856965c36a24e339b6058	34191.82
5f504b3a1c75b73d6151be81eb05bdc9	33741.99
3dd2a17168ec895c781a9191c1e95ad7	33689.54
d1c427060a0f73f6b889a5c7c61f2ac4	32283.81
aca2eb7d00ea1a7b8ebd4e68314663af	30010.85



INSIGHTS:

1. . Top Performing Customers & Revenue Generation

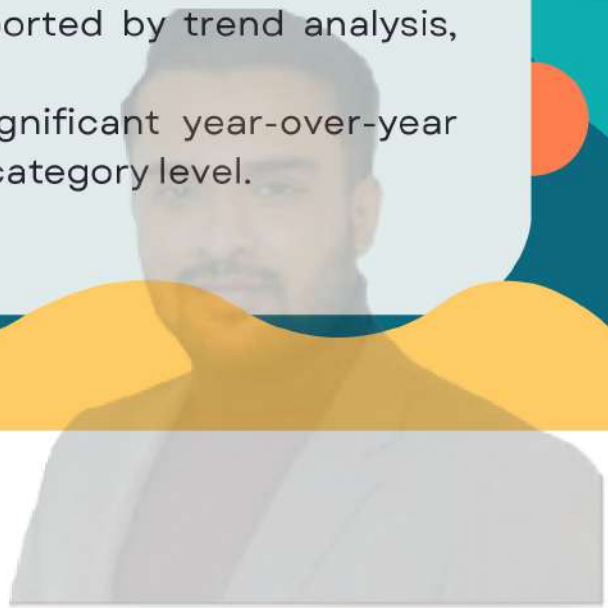
- The top 10 customers by lifetime revenue are identified – they form the core contributors to business income. Prioritizing their retention and loyalty is crucial.
- Customers who placed more than one order reflect positive user experience and product satisfaction.

2. Customer Behavior Patterns

- On average, repeat customers took a few days to make their first purchase after account creation. This insight reveals a short activation lag, which can be optimized through onboarding nudges or welcome offers.

3. Sales & Revenue Trends

- Revenue has shown positive month-over-month growth, supported by trend analysis, confirming the business is scaling steadily.
- Top 5 product categories (by revenue) also experienced significant year-over-year growth, showing strong demand evolution and performance at category level.



INSIGHTS:

4. Moving Averages & Seasonal Patterns

- A 3-month moving average of order volume reveals seasonal peaks and dips, notably during major sales periods and year-end – ideal for campaign planning.
- Monthly Average Order Value (AOV) shows a consistent upward trend, indicating increasing customer willingness to spend per transaction.

5. Delivery Performance Impact

- Orders with lower review scores (1–2 stars) took longer to deliver, clearly linking delivery delays to customer dissatisfaction.
- Sellers with shortest estimated vs actual delivery gaps stand out – these sellers should be recognized and emulated for operational efficiency.

6. Review Score Analysis by Payment Method

- Certain payment methods are consistently associated with higher review scores, suggesting they offer a smoother and more trusted checkout experience. Promoting these methods can enhance buyer satisfaction.