

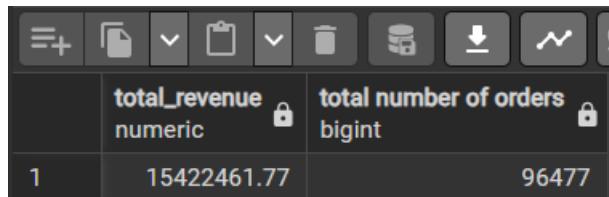
Brazilian E-Commerce SQL Queries

A . Q and A

Overall Business Performance

Q1. What is total revenue and total number of orders?

```
SELECT ROUND(SUM(olist_order_payments_dataset.payment_value::NUMERIC),2)
AS total_revenue,
COUNT(DISTINCT(olist_orders_dataset.order_id)) AS "total number of orders"
FROM olist_order_payments_dataset
JOIN olist_orders_dataset ON olist_order_payments_dataset.order_id =
olist_orders_dataset.order_id
WHERE olist_orders_dataset.order_status = 'delivered';
```



| | total_revenue numeric | total number of orders bigint |
|---|--------------------------|----------------------------------|
| 1 | 15422461.77 | 96477 |

2. Q2. Monthly revenue trend.

```
SELECT DATE_TRUNC('month', o.order_purchase_timestamp) AS month,
ROUND(SUM(p.payment_value)::NUMERIC, 2) AS monthly_revenue
FROM olist_orders_dataset o
JOIN olist_order_payments_dataset p ON o.order_id = p.order_id
WHERE o.order_status = 'delivered'
GROUP BY 1
ORDER BY 1;
```

The screenshot shows a data visualization interface with a toolbar at the top containing various icons for file operations, search, and analysis. Below the toolbar is a table with two columns. The first column is labeled 'month' and 'timestamp without time zone', and the second column is labeled 'monthly_revenue' and 'numeric'. The table contains six rows of data.

| | month timestamp without time zone | monthly_revenue numeric |
|---|--------------------------------------|----------------------------|
| 1 | 2016-10-01 00:00:00 | 46566.71 |
| 2 | 2016-12-01 00:00:00 | 19.62 |
| 3 | 2017-01-01 00:00:00 | 127545.67 |
| 4 | 2017-02-01 00:00:00 | 271298.65 |
| 5 | 2017-03-01 00:00:00 | 414369.39 |
| 6 | 2017-04-01 00:00:00 | 390952.18 |

Customer Analytics

Q3. How many repeat vs one-time customers?

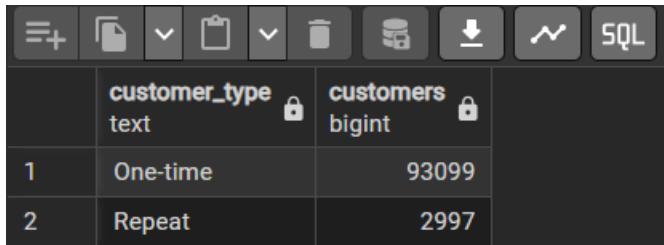
```

WITH customer_orders AS (
    SELECT c.customer_unique_id, COUNT(o.order_id) AS total_orders
    FROM olist_customers_dataset c
    JOIN olist_orders_dataset o ON c.customer_id = o.customer_id
    GROUP BY c.customer_unique_id
)
SELECT
CASE
WHEN total_orders = 1 THEN 'One-time'
ELSE 'Repeat'
END AS customer_type,
COUNT(*) AS customers

```

```
FROM customer_orders
```

```
GROUP BY customer_type;
```



| | customer_type | customers |
|---|---------------|-----------|
| 1 | text | bigint |
| 1 | One-time | 93099 |
| 2 | Repeat | 2997 |

Q4. Top 10 customers by lifetime value.

```
SELECT c.customer_unique_id, ROUND(SUM(p.payment_value::NUMERIC),2) AS  
"lifetime value"
```

```
FROM olist_customers_dataset c
```

```
JOIN olist_orders_dataset o ON c.customer_id = o.customer_id
```

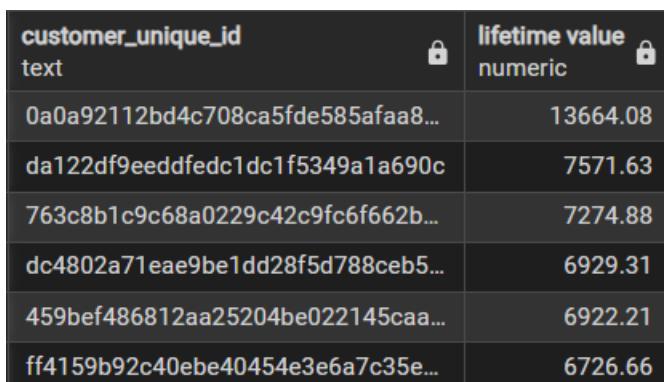
```
JOIN olist_order_payments_dataset p ON o.order_id = p.order_id
```

```
WHERE o.order_status = 'delivered'
```

```
GROUP BY c.customer_unique_id
```

```
ORDER BY "lifetime value" DESC
```

```
LIMIT 10;
```



| customer_unique_id | lifetime value |
|-----------------------------------|----------------|
| 0a0a92112bd4c708ca5fde585afaa8... | 13664.08 |
| da122df9eeddfedc1dc1f5349a1a690c | 7571.63 |
| 763c8b1c9c68a0229c42c9fc6f662b... | 7274.88 |
| dc4802a71eae9be1dd28f5d788ceb5... | 6929.31 |
| 459bef486812aa25204be022145caa... | 6922.21 |
| ff4159b92c40ebe40454e3e6a7c35e... | 6726.66 |

Q5. Revenue by customer state.

```
SELECT c.customer_state, ROUND(SUM(p.payment_value::NUMERIC),2) AS "Revenue by state"

FROM olist_customers_dataset c

JOIN olist_orders_dataset o ON c.customer_id = o.customer_id

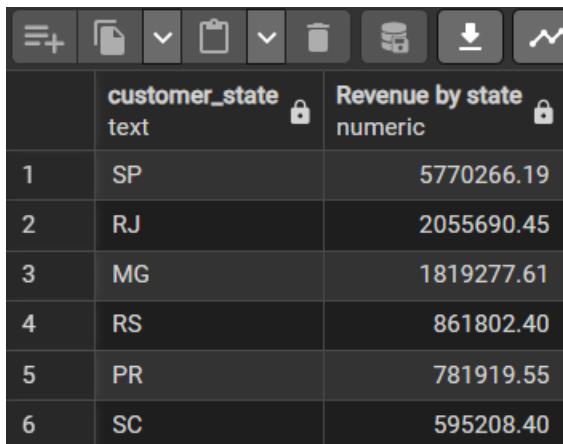
JOIN olist_order_payments_dataset p ON o.order_id = p.order_id

WHERE o.order_status = 'delivered'

GROUP BY c.customer_state

ORDER BY "Revenue by state" DESC

LIMIT 10;
```



The screenshot shows a data visualization interface with a toolbar at the top containing various icons for file operations and data manipulation. Below the toolbar is a table with two columns: 'customer_state' and 'Revenue by state'. The data is sorted by revenue in descending order, with the top 10 states listed. The table has 6 rows, indexed from 1 to 6.

| | customer_state | Revenue by state |
|---|----------------|------------------|
| 1 | SP | 5770266.19 |
| 2 | RJ | 2055690.45 |
| 3 | MG | 1819277.61 |
| 4 | RS | 861802.40 |
| 5 | PR | 781919.55 |
| 6 | SC | 595208.40 |

Product Analytics

Q6. Top 10 product categories by revenue

```
SELECT n.product_category_name_english, ROUND(SUM(i.price::NUMERIC + i.freight_value::NUMERIC),2) AS gross_revenue

FROM product_category_name_translation n
```

```

JOIN olist_products_dataset o ON n.product_category_name =
o.product_category_name

JOIN olist_order_items_dataset i ON o.product_id = i.product_id

JOIN olist_orders_dataset ord ON i.order_id = ord.order_id

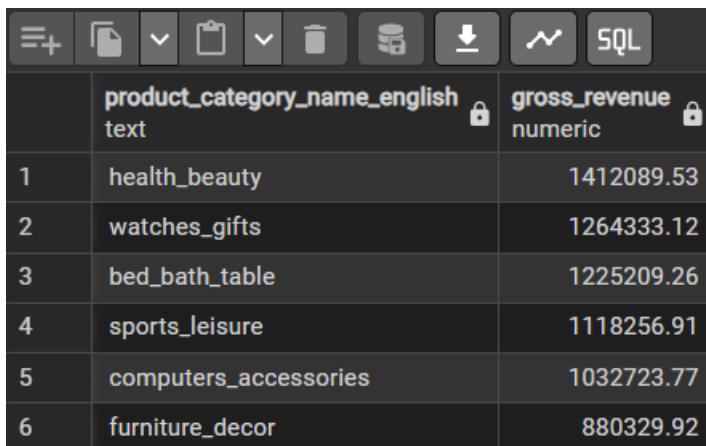
WHERE ord.order_status = 'delivered'

GROUP BY n.product_category_name_english

ORDER BY gross_revenue DESC

LIMIT 10;

```



The screenshot shows a database interface with a toolbar at the top containing various icons for file operations and SQL execution. Below the toolbar is a table with the following data:

| | product_category_name_english | gross_revenue |
|---|-------------------------------|---------------|
| | text | numeric |
| 1 | health_beauty | 1412089.53 |
| 2 | watches_gifts | 1264333.12 |
| 3 | bed_bath_table | 1225209.26 |
| 4 | sports_leisure | 1118256.91 |
| 5 | computers_accessories | 1032723.77 |
| 6 | furniture_decor | 880329.92 |

Q7. Products with highest average price.

```

SELECT n.product_category_name_english, ROUND(AVG(i.price::NUMERIC),2) AS
avg_product_price

FROM product_category_name_translation n

JOIN olist_products_dataset p ON n.product_category_name =
p.product_category_name

JOIN olist_order_items_dataset i ON p.product_id = i.product_id

JOIN olist_orders_dataset ord ON i.order_id = ord.order_id

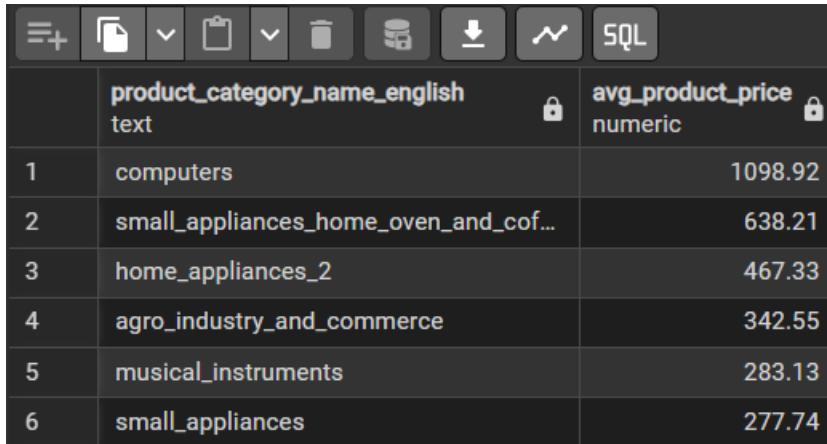
WHERE ord.order_status = 'delivered'

GROUP BY n.product_category_name_english

```

```
ORDER BY avg_product_price DESC
```

```
LIMIT 10;
```



The screenshot shows a database interface with a toolbar at the top containing various icons for file operations and SQL execution. Below the toolbar is a table with two columns: 'product_category_name_english' (text) and 'avg_product_price' (numeric). The table has 6 rows, each representing a category and its average price. The data is as follows:

| | product_category_name_english | avg_product_price |
|---|---------------------------------------|-------------------|
| 1 | computers | 1098.92 |
| 2 | small_appliances_home_oven_and_cof... | 638.21 |
| 3 | home_appliances_2 | 467.33 |
| 4 | agro_industry_and_commerce | 342.55 |
| 5 | musical_instruments | 283.13 |
| 6 | small_appliances | 277.74 |

Seller Performance

Q8. Top 10 sellers by revenue.

```
SELECT s.seller_id, ROUND(SUM(i.price::NUMERIC + freight_value::NUMERIC),2) AS revenue_by_sellers
```

```
FROM olist_sellers_dataset s
```

```
JOIN olist_order_items_dataset i ON s.seller_id = i.seller_id
```

```
JOIN olist_orders_dataset ord ON i.order_id = ord.order_id
```

```
WHERE ord.order_status = 'delivered'
```

```
GROUP BY s.seller_id
```

```
ORDER BY revenue_by_sellers DESC
```

```
LIMIT 10;
```

The screenshot shows a database interface with a toolbar at the top containing various icons for operations like insert, delete, and export. Below the toolbar is a table with two columns: 'seller_id' (text) and 'revenue_by_sellers' (numeric). The table contains six rows of data.

| | seller_id | revenue_by_sellers |
|---|-----------------------------------|--------------------|
| 1 | 4869f7a5dfa277a7dca6462dcf3b52... | 247007.06 |
| 2 | 7c67e1448b00f6e969d365cea6b010... | 237806.69 |
| 3 | 4a3ca9315b744ce9f8e93743614938... | 231220.43 |
| 4 | 53243585a1d6dc2643021fd1853d8... | 230797.02 |
| 5 | fa1c13f2614d7b5c4749cbc52fecda94 | 200833.50 |
| 6 | da8622b14eb17ae2831f4ac5b9dab8... | 184706.78 |

Q9. Seller delivery performance (average delivery time)

WITH seller_orders AS (

```

SELECT DISTINCT i.seller_id, ord.order_id, ord.order_purchase_timestamp,
ord.order_delivered_customer_date

FROM olist_order_items_dataset i

JOIN olist_orders_dataset ord ON i.order_id = ord.order_id

WHERE ord.order_status = 'delivered' AND ord.order_delivered_customer_date IS NOT
NULL)

SELECT seller_id, ROUND( AVG( EXTRACT(EPOCH
FROM( order_delivered_customer_date - order_purchase_timestamp ) / 86400)),2) AS
average_delivery_days

FROM seller_orders

GROUP BY seller_id

ORDER BY average_delivery_days

LIMIT 10;

```

A screenshot of a database interface showing a table with two columns: 'seller_id' (text) and 'average_delivery_days' (numeric). The table has 6 rows of data.

| | seller_id | average_delivery_days |
|---|-----------------------------------|-----------------------|
| 1 | 139157dd4daa45c25b0807ffff348363 | 1.21 |
| 2 | 5e063e85d44b0f5c3e6ec3131103a5... | 1.29 |
| 3 | 6561d6bf844e464b4019442692b40e... | 1.43 |
| 4 | 702835e4b785b67a084280efca3557... | 1.80 |
| 5 | 674207551483fec113276b67b0d871... | 1.87 |
| 6 | 2c00c85d30361cd2ced2969cffbbffa3 | 1.87 |

Payment Analytics

Q10. Revenue by payment type.

```

SELECT p.payment_type, ROUND(SUM(p.payment_value::NUMERIC),2) AS "Revenue
by payment type",

ROUND(100.0 * SUM(p.payment_value::NUMERIC) /
SUM(SUM(p.payment_value::NUMERIC))OVER(),2) AS revenue_percent

FROM olist_order_payments_dataset p

JOIN olist_orders_dataset ord ON p.order_id = ord.order_id

WHERE ord.order_status = 'delivered'

GROUP BY p.payment_type

ORDER BY "Revenue by payment type" DESC;

```

A screenshot of a database interface showing a table with three columns: 'payment_type' (text), 'Revenue by payment type' (numeric), and 'revenue_percent' (numeric). The table has 4 rows of data.

| | payment_type | Revenue by payment type | revenue_percent |
|---|--------------|-------------------------|-----------------|
| 1 | credit_card | 12101094.88 | 78.46 |
| 2 | boleto | 2769932.58 | 17.96 |
| 3 | voucher | 343013.19 | 2.22 |
| 4 | debit_card | 208421.12 | 1.35 |

Q11. Average payment installments by payment type

```

SELECT p.payment_type, ROUND(AVG(p.payment_installments)::NUMERIC),2) AS
"Average payment installments"

FROM olist_order_payments_dataset p

JOIN olist_orders_dataset ord ON p.order_id = ord.order_id

WHERE ord.order_status = 'delivered'

GROUP BY p.payment_type

ORDER BY "Average payment installments" DESC;

```

| | payment_type | Average payment installments |
|---|--------------|------------------------------|
| | text | numeric |
| 1 | credit_card | 3.50 |
| 2 | boleto | 1.00 |
| 3 | debit_card | 1.00 |
| 4 | voucher | 1.00 |

Delivery & Logistics

Q12. Late delivery rate by month.

```

SELECT DATE_TRUNC('month', order_purchase_timestamp) AS month, COUNT() AS
total_orders, COUNT() FILTER (
WHERE order_delivered_customer_date > order_estimated_delivery_date) AS
late_orders, ROUND(100.0 * COUNT() FILTER (
WHERE order_delivered_customer_date > order_estimated_delivery_date) /
NULLIF(COUNT(), 0),2) AS late_delivery_rate

FROM olist_orders_dataset

WHERE order_status = 'delivered'

GROUP BY month ORDER BY month;

```

| | month timestamp without time zone | total_orders bigint | late_orders bigint | late_delivery_rate numeric |
|---|--------------------------------------|------------------------|-----------------------|-------------------------------|
| 1 | 2016-09-01 00:00:00 | 1 | 1 | 100.00 |
| 2 | 2016-10-01 00:00:00 | 265 | 3 | 1.13 |
| 3 | 2016-12-01 00:00:00 | 1 | 0 | 0.00 |
| 4 | 2017-01-01 00:00:00 | 750 | 23 | 3.07 |
| 5 | 2017-02-01 00:00:00 | 1653 | 53 | 3.21 |
| 6 | 2017-03-01 00:00:00 | 2546 | 142 | 5.58 |

Reviews & Customer Satisfaction

Q13. Average review score by product category.

```

SELECT n.product_category_name_english, ROUND(AVG(r.review_score),2) AS
"Average review score",

COUNT(DISTINCT r.review_id) AS total_reviews

FROM product_category_name_translation n

JOIN olist_products_dataset p ON n.product_category_name =
p.product_category_name

JOIN olist_order_items_dataset i ON p.product_id = i.product_id

JOIN olist_order_reviews_dataset r ON i.order_id = r.order_id

JOIN olist_orders_dataset ord ON r.order_id = ord.order_id

WHERE ord.order_status = 'delivered'

GROUP BY n.product_category_name_english

HAVING COUNT(DISTINCT r.review_id) >= 50

ORDER BY "Average review score" DESC;

```

| | product_category_name_english | Average review score | total_reviews |
|---|-------------------------------------|----------------------|---------------|
| 1 | books_imported | 4.51 | 50 |
| 2 | books_general_interest | 4.51 | 492 |
| 3 | construction_tools_tools | 4.44 | 94 |
| 4 | small_appliances_home_oven_and_c... | 4.44 | 72 |
| 5 | books_technical | 4.39 | 257 |
| 6 | food_drink | 4.37 | 221 |

Q15. Does faster delivery improve ratings?

SELECT CASE

```

WHEN ord.order_delivered_customer_date <= ord.order_estimated_delivery_date
THEN 'On Time' ELSE 'Late' END AS delivery_status, ROUND(AVG(r.review_score),2)
AS avg_review,
COUNT(*) AS total_orders
FROM olist_orders_dataset ord
JOIN olist_order_reviews_dataset r ON ord.order_id = r.order_id
WHERE ord.order_status = 'delivered'
AND ord.order_delivered_customer_date IS NOT NULL
AND ord.order_estimated_delivery_date IS NOT NULL
GROUP BY delivery_status
ORDER BY avg_review DESC;
```

A screenshot of a database interface showing a table with three columns: delivery_status, avg_review, and total_orders. The table has two rows. Row 1: delivery_status = On Time, avg_review = 4.29, total_orders = 88653. Row 2: delivery_status = Late, avg_review = 2.57, total_orders = 7700.

| | delivery_status | avg_review | total_orders |
|---|-----------------|------------|--------------|
| 1 | On Time | 4.29 | 88653 |
| 2 | Late | 2.57 | 7700 |