Olist: An E-Commerce Analysis of the Brazilian Marketplace

Olist is a **Brazilian e-commerce** platform that connects small and medium-sized businesses (SMEs) with customers, allowing merchants to list and sell their products online. The platform operates as a **marketplace**, where merchants can list their products and customers can browse & purchase them online. Olist started its operation in 2015.

This particular business case focuses on analyzing the operations of Olist from **Sept-2016 to Oct-2018** in Brazil and for that we are provided with information of **approx. 100K orders** placed during this period.

The available datasets facilitate the analysis of orders across dimensions such as order status, price, payment and freight charges, customer location, product attributes, and customer reviews.

Via this analysis, we will try to shed light on key aspects of the business, including *sales distribution, revenue generation, order basket diversification, customer demographics and preferences, shipping efficiency, and customer satisfaction levels.* Through a comprehensive examination of these business aspects, we will try to derive **valuable insights** about the company's operations and performance.

```
# approx. 100K orders
select
  count(order_id) as num_of_orders
from `olist_business_case.orders`;
```

This script will process 827.62 MB when run.

Query results

Job information | Results | Chart

Row	num_of_orders
1	99441

```
# Orders placed between Sept-2016 to Oct-2018
select
  min(order_purchase_timestamp) as start_point,
  max(order_purchase_timestamp) as end_point
from `olist_business_case.orders`;
```

This script will process 827.62 MB when run.

Query results

Job information | Results | Chart | JSON | Executic

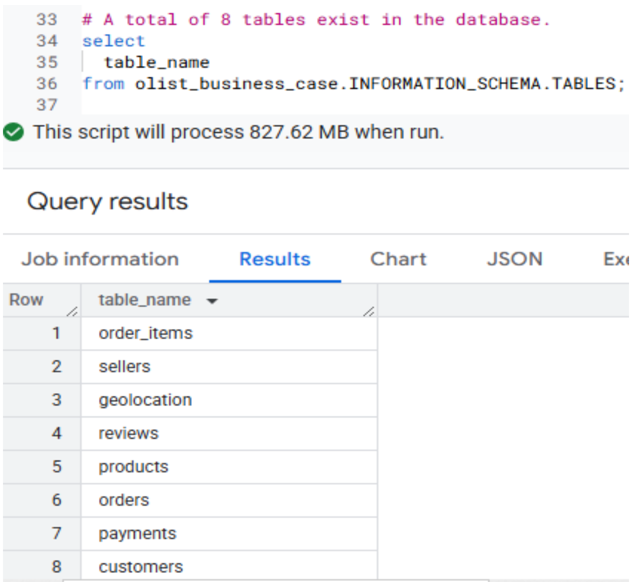
Row	start_point	end_point
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

This report is organized into **two primary sections** –

- **Section 1:** Initial Data Assessment and Familiarization *(Refer to Pages 2-14)*
- **Section 2:** Deep Dive into Metrics for Business Insights *(Refer to Pages 15-37)*

Section 1 – Initial Data Assessment and Familiarization

In this we will examine the **schemas** of all the provided tables and also figure out how these tables are related to each other i.e., the common keys. Simultaneously, we will also interpret the **business meaning** of each column. Also, we will validate the data i.e., assess the quality of data provided to us; for example, looking at **missing values**, check for **data integrity**.



```
# A total of 8 tables exist in the database.
select
  table_name
from olist_business_case.INFORMATION_SCHEMA.TABLES;
```

This script will process 827.62 MB when run.

Query results

Job information | Results | Chart | JSON | Ex

Row	table_name
1	order_items
2	sellers
3	geolocation
4	reviews
5	products
6	orders
7	payments
8	customers

As shown above, we have total **8 tables/datasets** provided to us –

orders, order_items, products, customers, sellers, geolocation, payments, reviews

Let's skim through each table one by one –

---

customers –

This table has information about customers and their locations.

```sql
select
  column_name,
  data_type
from olist_business_case.INFORMATION_SCHEMA.COLUMNS
where table_name = 'customers';
```

This script will process 827.62 MB when run.

Query results

Job information | Results | Chart | JSON | Executic

Row	column_name	data_type
1	customer_id	STRING
2	customer_unique_id	STRING
3	customer_zip_code_prefix	INT64
4	customer_city	STRING
5	customer_state	STRING

- Each *'customer_id'* represents a unique order of the *'orders'* table as the company's database is designed in such a way that whenever an order gets placed, a new *'customer_id'* gets created for that customer. It means that the same customer will get different *'customer_id'* for different orders they place.
- In that case what is the unique identifier for each customer - There is a column named *'customer_unique_id'* in the *'customers'* table that assigns one single unique id to each customer. So, if a customer has placed N orders, then they have N different *'customer_id'* but one single *'customer_unique_id'*.
- But what can be the intuition behind so i.e., assigning a new *'customer_id'* to the same customer for new every order they place – Maybe so that we can use the location information of the *'customers'* table as the delivery location for that order.

```sql
# As explained, the 'customer_id' column has all unique values whereas,
# the 'customer_unique_id' column has duplicate values in it.
select
  count(customer_id),
  count(distinct customer_id),
  count(customer_unique_id),
  count(distinct customer_unique_id)
from `olist_business_case.customers`;
```

This script will process 827.62 MB when run.

Query results

Job information | Results | Chart | JSON | Execution details | E

Row	f0_	f1_	f2_	f3_
1	99441	99441	99441	96096

- The *'customer_zip_code_prefix'* column represents the first five digits of the location zip code.
- The *'customer_city'* column represents the name of the city where the customer is located or from where the order is placed.
- The *'customer_state'* column represents the state code where the customer is located or from where the order is placed. For example, RJ represents Rio de Janeiro.

```
-- The customers are located across 27 different states of Brazil
select count(distinct customer_state) as num_of_states
from `olist_business_case.customers`;
```

This script will process 827.62 MB when run.

Query results

Job information | Results | Chart | JSON | Execution details

Row	num_of_states
1	27

- None of the columns in the *'customers'* table has missing values (NULL values) in it.

---

sellers –

This table has information about the sellers who are listed on Olist along with their locations.

```
select
  column_name,
  data_type
from olist_business_case.INFORMATION_SCHEMA.COLUMNS
where table_name = 'sellers';
```

This script will process 827.62 MB when run.

Query results

Job information | Results | Chart | JSON | Executic

Row	column_name	data_type
1	seller_id	STRING
2	seller_zip_code_prefix	INT64
3	seller_city	STRING
4	seller_state	STRING

- Each *'seller_id'* represents a unique seller listed on the platform.
- *'seller_zip_code_prefix'* column represents the first five digits of the location zip code.

- *'seller_city'* column represents the name of the city where the seller is located.
- *'seller_state'* column represents the state code where the seller is located. For example, RJ represents Rio de Janeiro.

```
-- The sellers are located across 23 different states of Brazil.
select count(distinct seller_state) as num_of_states
from `olist_business_case.sellers`;
```

This script will process 827.62 MB when run.

Query results

Job information | Results | Chart | JSON | Execution details

Row	num_of_states
1	23

- None of the columns in the *'sellers'* table has missing values (NULL values) in it.

---

geolocation –

This table has information of Brazilian zip codes and their latitude/longitude coordinates along with the city and state names.

```
# This table has information of Brazilian zip codes and their
# latitude/longitude coordinates along with the city and state names.

select
  column_name,
  data_type
from olist_business_case.INFORMATION_SCHEMA.COLUMNS
where table_name = 'geolocation';
```

This script will process 827.62 MB when run.

Query results

Job information | Results | Chart | JSON | Execution details

Row	column_name	data_type
1	geolocation_zip_code_prefix	INT64
2	geolocation_lat	FLOAT64
3	geolocation_lng	FLOAT64
4	geolocation_city	STRING
5	geolocation_state	STRING

- None of the columns in the *'geolocation'* table has missing values (NULL values) in it.

orders –

This table has information about all the orders placed at Olist from Sept-2016 to Oct-2018.

```
select
  column_name,
  data_type
from olist_business_case.INFORMATION_SCHEMA.COLUMNS
where table_name = 'orders';
```

This script will process 827.62 MB when run.

Query results

Job information | Results | Chart | JSON | Executic

Row	column_name	data_type
1	order_id	STRING
2	customer_id	STRING
3	order_status	STRING
4	order_purchase_timestamp	TIMESTAMP
5	order_approved_at	TIMESTAMP
6	order_delivered_carrier_date	TIMESTAMP
7	order_delivered_customer_date	TIMESTAMP
8	order_estimated_delivery_date	TIMESTAMP

- *'order_id'* represents the unique identifier assigned to each order.
- *'customer_id'* represents the customer who placed that order.
- *'order_status'* represents the current status of the orders – delivered, shipped, cancelled, etc.
- *'order_purchase_timestamp'* represents the timestamp when the order was placed.
- *'order_approved_at'* represents the payment approval timestamp.
- *'order_delivered_carrier_date'* represents the timestamp when the order was handled to the logistic partner.
- *'order_delivered_customer_date'* represents the timestamp when the order was actually delivered to the customer.
- *'order_estimated_delivery_date'* represents the estimated delivery date that was informed to the customers when they placed the order.

Now, let's try to assess the missing values (NULL values) present in the *'orders'* table –

- The following columns – *'order_id'*, *'customer_id'*, *'order_status'*, *'order_purchase_timestamp'*, *'order_estimated_delivery_date'* do not have any missing values.

- The *'order_approved_at'* field is missing for 160 orders out of a total of approx. 100K orders; however, corresponding payment records exist for all these orders in the 'payments' table signifying that payment has been done for all these orders. Therefore, we can safely impute these missing values (NULL values). In order to impute these missing values, we can use the average/median approval time of all the orders.

```
# 160 orders out of ~100K orders have
# a missing "payment approval timestamp"
select count(*) as missing_value_count
from `olist_business_case.orders`
where order_approved_at IS NULL;
```

This script will process 817.13 MB when run.

Query results

Job information | Results | Chart | JSO

Row	missing_value_count
1	160

```
# Get me the 'order_id' for which payment approval timestamp
# is missing and payment indeed has not been done.
select order_id
from `olist_business_case.orders`
where order_approved_at IS NULL
EXCEPT DISTINCT
select distinct order_id
from `olist_business_case.payments`;
```

This script will process 817.13 MB when run.

Query results

Job information | Results | Chart | JSON | Execution de

There is no data to display.

- Out of approx. 100K orders placed at Olist, 1783 orders (i.e., 2%) have a missing information for when it was handled to the logistics partner for delivery and 2965 orders (i.e., 3%) have a missing information for when it was delivered to the customer.

```sql
# 1783 orders out of ~100K orders i.e., ~2% of total orders
# have a missing information for when it was handled to the
# logistics partner for delivery.
select count(order_id) as num_of_orders
from `olist_business_case.orders`
where order_delivered_carrier_date IS NULL;
```

This script will process 825.8 MB when run.

Query results

Job information | Results | Chart | JSON | Execution de

Row	num_of_orders
1	1783

```sql
# 2965 orders out of ~100K orders i.e., ~3% of total orders
# have a missing information for when it was delivered to
# the customer.
select count(order_id) as num_of_orders
from `olist_business_case.orders`
where order_delivered_customer_date IS NULL;
```

This script will process 825.8 MB when run.

Query results

Job information | Results | Chart | JSON | Execution d

Row	num_of_orders
1	2965

Now, let's further inspect these missing values for the *'order_delivered_carrier_date'* and *'order_delivered_customer_date'* fields so that we can think of an imputation logic for these fields.

- For the following *'order_status'* – 'created', 'approved', 'unavailable', 'processing', and 'invoiced', both the *'order_delivered_carrier_date'* and *'order_delivered_customer_date'* fields should be NULL as these orders have not yet reached to the shipment stage. Upon checking the data, it is indeed the case as well.

```sql
select distinct
  order_delivered_carrier_date,
  order_delivered_customer_date
from `olist_business_case.orders`
where order_status IN ('created', 'approved', 'unavailable', 'processing', 'invoiced');
```

This script will process 817.87 MB when run.

Query results

Job information | Results | Chart | JSON | Execution details | Execution graph

Row	order_delivered_carrier_date	order_delivered_customer_date
1	null	null

- For the orders with *'order_status'* – 'shipped', the field *'order_delivered_carrier_date'* should not be NULL and the field *'order_delivered_customer_date'* should be NULL. Upon checking the data, it is indeed the case as well.
- For the orders with *'order_status'* – 'cancelled', the fields *'order_delivered_carrier_date'* and *'order_delivered_customer_date'* may or may not be NULL as cancellation can be done at any stage.
- For the orders with *'order_status'* – 'delivered', both the fields *'order_delivered_carrier_date'* and *'order_delivered_customer_date'* should not be NULL. Upon checking the data, it is indeed the case as well (ignoring few exceptions – 8 orders as that can be a human error).

So, from the above inspection, we can comment that NULL values appearing in both the fields are completely aligned with the business logic and so, these NULL values should not be interpreted as missing values. However, if we still need to impute these NULL values, then we can do it using the average/median shipping and delivery time respectively.

- Data Integrity validation – All the customers present in the *'orders'* table are available in the parent table *'customers'*, ensuring referential integrity check.

- Anomalies detected in Delivery timestamps – Among the approximately 100K orders, 23 records present a customer delivery date earlier than the carrier handover date, while 166 orders show a carrier handover date preceding the purchase timestamp. These nonsensical timestamps require either manual correction or omission from delivery time analysis.

Order_items –

This table contains the item-level information of each order (basically, multiple items can be purchased within the same order).

```sql
select
  column_name,
  data_type
from olist_business_case.INFORMATION_SCHEMA.COLUMNS
where table_name = 'order_items';
```

This script will process 825.8 MB when run.

Query results

Job information | Results | Chart | JSON | Executic

Row	column_name	data_type
1	order_id	STRING
2	order_item_id	INT64
3	product_id	STRING
4	seller_id	STRING
5	shipping_limit_date	TIMESTAMP
6	price	FLOAT64
7	freight_value	FLOAT64

- *'order_id'* represents the unique identifier of each order.
- *'order_item_id'* is the serial number assigned to items within an order.

It is important to note the absence of a dedicated quantity field in the table. Consequently, each unit of a product within an order is stored as a separate row in the table resulting in duplicate *'order_id'* entries.

- *'product_id'* represents the unique identifier of the products included in the order.
- *'seller_id'* represents the unique identifier of the seller who will be supplying that product.

Note that different products within an order might be fulfilled by different sellers.

- *'shipping_limit_date'* represents the seller shipping limit date i.e., the deadline for the seller to hand over the order to the logistics partner.
- *'price'* represents the actual price of the associated product
- *'freight_value'* represents the delivery associated costs

The total price for an item would be represented by *'price'* + *'freight_value'*.

- Data Integrity validation – All the orders, sellers and products present in the *'order_items'* table are available in their parent tables *'orders'*, *'sellers'* and *'products'* respectively, ensuring referential integrity.

- None of the columns in the *'order_items'* table has missing values (NULL values) in it.

products –

This table contains details of all the products listed on Olist.

```sql
select
  column_name,
  data_type
from olist_business_case.INFORMATION_SCHEMA.COLUMNS
where table_name = 'products';
```

This script will process 825.8 MB when run.

Query results

Job information | Results | Chart | JSON | Executic

Row	column_name	data_type
1	product_id	STRING
2	product_category_name	STRING
3	product_name_length	INT64
4	product_description_length	INT64
5	product_photos_qty	INT64
6	product_weight_g	INT64
7	product_length_cm	INT64
8	product_height_cm	INT64
9	product_width_cm	INT64

- *'product_id'* represents the unique identifier for each product.
- *'product_category_name'* represents the product category that the product belongs to.
- Out of total 32951 products available in the table, 610 products (i.e., approx. 2%) have *'product_category_name'* missing (i.e., NULL). These missing values are replaced with the string value 'unknown'.
- There are a total of 74 distinct product categories present in the data.

```sql
# Total 74 product categories -
select
  count(distinct product_category_name) as num_of_product_categories
from `olist_business_case.products`;
```

This script will process 839.17 MB when run.

Query results

Job information | Results | Chart | JSON | Execution details

Row	num_of_product_categories
1	74

- There is information available about some other attributes of the product as well like dimensions of the product, weight of the product, etc.

payments –

This table contains details of the payments made for the orders.

```sql
select
  column_name,
  data_type
from olist_business_case.INFORMATION_SCHEMA.COLUMNS
where table_name = 'payments';
```

Query completed

Query results

Job information | Results | Chart | JSON | Executic

Row	column_name	data_type
1	order_id	STRING
2	payment_sequential	INT64
3	payment_type	STRING
4	payment_installments	INT64
5	payment_value	FLOAT64

- *'order_id'* represents the unique identifier of the order.
- *'payment_sequential'* – A customer may pay for an order with more than one payment method (for example, credit card, voucher, debit card, etc.). If they do so, a sequence will be created to accommodate all the payments.
- *'payment_type'* represents the method of payment chosen by the customer. The different payment methods available in the data are – credit card, boleto, voucher, debit card.

  Note: "Boleto" refers to a bank slip or voucher used for payments, particularly in Brazil. It is often used to make payments for goods, services, or bills, especially for those without credit cards or online banking access.

- *'payment_value'* represents the transaction value.
- *'payment_installments'* represents the number of installments chosen by the customer to fulfill the payment.
  Note: From the data, we can observe that payment via credit card is the sole method permitting transactions to be completed in multiple installments. All other payment methods are getting processed as a single, lump-sum transaction.

```sql
# Payment via credit card is the sole method permitting transactions to be completed in multiple installments.
# All other payment methods are processed as a single, lump-sum transaction.
select
  payment_type,
  max(payment_installments) as max_num_of_installments
from `olist_business_case.payments`
group by payment_type;
```

This script will process 814.56 MB when run.

Query results

Job information | Results | Chart | JSON | Execution details | Execution graph

Row	payment_type	max_num_of_installn
1	credit_card	24
2	voucher	1
3	not_defined	1
4	boleto	1
5	debit_card	1

- Data Integrity validation – All the orders present in the *'payments'* table are available in its parent table *'orders'*, ensuring referential integrity.
- None of the columns in the *'payments'* table has missing values (NULL values) in it.

<u>reviews</u> –

This table contains information about the reviews given by customers for their overall purchasing experience. Once the customer receives the product or the estimated delivery date is due, the customer gets a satisfaction survey by email where they can give a **review score (1 to 5)** for the purchase experience and write down some comments as well.

```sql
select
  column_name,
  data_type
from olist_business_case.INFORMATION_SCHEMA.COLUMNS
where table_name = 'reviews';
```

Query completed

Query results

Job information | Results | Chart | JSON | Executic

Row	column_name	data_type
1	review_id	STRING
2	order_id	STRING
3	review_score	INT64
4	review_comment_title	STRING
5	review_creation_date	TIMESTAMP
6	review_answer_timestamp	TIMESTAMP

- *'review_id'* represents the identifier for the review.
- *'order_id'* represents the order associated with the review.

Multiple *'review_id'* values exist for the same *'order_id'* in the table. It may be because customer is giving more than one review for the same order.

- *'review_score'* represents a 1 to 5 integer score given by the customer for the overall purchase experience.
- *'review_comment_title'* represents the title of the review posted by customer.
- *'review_creation_date'* represents the date on which satisfaction survey was sent to the customer.
- *'review_answer_timestamp'* represents the timestamp at which the satisfaction survey was answered by the customer.

- None of the fields in the *'reviews'* table except *'review_comment_title'* has missing values (NULL values) in it. And the NULL values in the *'review_comment_title'* field should not be interpreted as missing data as customers may not always be interested in leaving a comment.

- Data Integrity validation – All the orders present in the *'reviews'* table are available in its parent table *'orders'*, ensuring referential integrity. Also, out of approx. 100K orders, 768 do not have corresponding customer reviews which is an expected behavior as providing feedback/review is not a mandatory requirement.

Section 2 – Deep Dive into Metrics for Business Insights

Order fulfillment rate –

```
select
  order_status,
  num_of_orders,
  CONCAT(CAST(ROUND(num_of_orders/(SUM(num_of_orders) OVER())*100, 2) AS STRING), ' %') as percent_of_orders
from (
      select
        order_status,
        count(order_id) as num_of_orders
      from `olist_business_case.orders`
      group by order_status
    ) as tbl
order by num_of_orders desc;
```

This script will process 823.42 MB when run.

Query results

Job information | Results | Chart | JSON | Execution details | Execution graph

Row	order_status	num_of_orders	percent_of_orders
1	delivered	96478	97.02 %
2	shipped	1107	1.11 %
3	canceled	625	0.63 %
4	unavailable	609	0.61 %
5	invoiced	314	0.32 %
6	processing	301	0.3 %
7	created	5	0.01 %
8	approved	2	0 %

- The overwhelming majority of orders (**>97%**) reached the *'delivered'* status, indicating a **highly successful order fulfillment process** at Olist. Achieving such a high delivery rate **builds trust and satisfaction among customers**, encouraging them to return to the platform and recommend it to others (**word-of-mouth referrals**).
- Cancelled orders represent just 0.63% of the total, indicating **low order cancellation rates**.
- Overall, the data **reflects a mature and optimized e-commerce operation** with minimal friction in order processing and fulfillment.

Order Basket analysis –

Let's analyze the diversification of products in the order basket.

```
select
  Basket_Type,
  num_of_orders,
  CONCAT(CAST(ROUND(num_of_orders/(SUM(num_of_orders) OVER())*100, 2) AS STRING), ' %') as percentage_of_orders
from (
      select
        'Single-item orders' as Basket_Type,
        count(*) as num_of_orders
      from (
            select
              order_id
            from `olist_business_case.order_items`
            group by order_id
            having count(order_item_id) = 1)
      UNION ALL
      select
        'Multiple-item orders' as Basket_Type,
        count(*) as num_of_orders
      from (
            select
              order_id
            from `olist_business_case.order_items`
            group by order_id
            having count(order_item_id) > 1)
    )
order by CASE
          WHEN Basket_Type = 'Single-item orders' THEN 1
          ELSE 2
        END;
```

Query results

Job information | Results | Chart | JSON | Execution details | Exe

Row	Basket_Type	num_of_orders	percentage_of_orders
1	Single-item orders	88863	90.06 %
2	Multiple-item orders	9803	9.94 %

- We can observe an **overwhelming majority (90%) of single-item orders** meaning that most transactions involve the purchase of just one item.
- Only 10% of orders include more than one item. This indicates **limited cross-selling or bundling activity.**
- This stark contrast between single-item orders and multiple-item orders highlights a significant opportunity to **encourage customers to purchase more items per order.**

- Olist can offer **Volume discounts** (for example, flat discounts on higher quantities or offers like Buy x Get y Free), provide **"frequently bought together" or "You may also like" product suggestions**, etc. to encourage customers to add more items to their basket.

- Also, Olist can offer **free delivery beyond a minimum order value** to incentivize customers to add more items to their basket.

- Additionally, Olist can flash "**Limited-time offers**" on related items to **create urgency** and encourage multi-unit purchases.

Let's try to investigate if price can be the reason behind these single-item orders as in are expensive products leading to single-item orders?

For this, we will examine if the prices involved in these single-item orders are significantly higher than the prices involved in the multiple-item orders.

```
select
  Basket_Type,
  num_of_orders,
  CONCAT(CAST(ROUND(num_of_orders/(SUM(num_of_orders) OVER())*100, 2) AS STRING), ' %') as percentage_of_orders,
  median_price,
  mean_price
from (
      select distinct
        'Single-item orders' as Basket_Type,
        count(order_id) over() as num_of_orders,
        ROUND(percentile_cont(price+freight_value, 0.5) OVER(), 2) as median_price,
        ROUND(avg(price+freight_value) OVER(), 2) as mean_price
      from `olist_business_case.order_items`
      where order_id IN (select
                           order_id
                         from `olist_business_case.order_items`
                         group by order_id
                         having count(order_item_id) = 1)
      UNION ALL

      select distinct
        'Multiple-item orders' as Basket_Type,
        count(distinct order_id) over() as num_of_orders,
        ROUND(percentile_cont(price+freight_value, 0.5) OVER(), 2) as median_price,
        ROUND(avg(price+freight_value) OVER(), 2) as mean_price
      from `olist_business_case.order_items`
      where order_id IN (select
                           order_id
                         from `olist_business_case.order_items`
                         group by order_id
                         having count(order_item_id) > 1)
    )
order by CASE
           WHEN Basket_Type = 'Single-item orders' THEN 1
           ELSE 2
         END;
```

Query results

Job information | Results | Chart | JSON | Execution details | Execution graph

Row	Basket_Type	num_of_orders	percentage_of_orders	median_price	mean_price
1	Single-item orders	88863	90.06 %	99.03	150.75
2	Multiple-item orders	9803	9.94 %	73.34	102.91

- Although, median prices involved in the single-item orders is higher than the multiple-item orders, the difference is not large. So, we **can't attribute prices to be the reason behind single-item orders**.

Since price does not appear to be the main factor behind the high proportion of single-item orders, this trend may be attributed to a large number of first-time or one-time customers who have not yet developed enough trust in the platform to place large orders. To examine this, let's calculate the percentage of repeat customers on the platform –

```
select
  CONCAT(CAST(ROUND(countif(num_of_orders = 1)/count(*)*100, 2) AS STRING), ' %') as percent_of_one_time_customers
from (
      select
        c.customer_unique_id,
        count(o.order_id) as num_of_orders
      from `olist_business_case.orders` as o
      INNER JOIN `olist_business_case.customers` as c
      ON o.customer_id = c.customer_id
      group by c.customer_unique_id
    ) as tbl;
```

Query completed

Query results

Job information | Results | Chart | JSON | Execution details | Execution graph

Row	percent_of_one_time_customers
1	96.88 %

- As anticipated, approximately **97% of the customers on the platform are first-time or one-time buyers**, and **only about 3% of the customers making repeat purchases**. This underscores a significant opportunity to **enhance customer retention** through targeted strategies such as **loyalty programs** to reward repeat customers and **post-purchase engagement**.

- In the highly competitive e-commerce market, **acquiring new customers** often involves substantial marketing and promotional expenses, resulting in a **high Customer Acquisition Cost (CAC)**. Therefore, **repeat customers** are crucial for **sustaining long-term business growth** and **profitability** as they are more likely to make additional purchases and require less investment to re-engage.

- Given that Olist was founded in 2015 and the available data spans from September 2016 to October 2018, it is reasonable to observe a high proportion of first-time customers during this growth phase. Therefore, to accurately assess the effectiveness of retention strategies, it is essential to analyze customer data over a longer period.

- Moreover, later in this report, we will also assess the customer review scores to evaluate the overall purchasing experience. Particularly, we will examine whether a high proportion of low review scores is present, as this could be a significant factor behind the low rate of repeat purchases.

Sales and Revenue Analysis –

Note: For analyzing the sales and revenue figures, let's not include the orders which have been *'cancelled'*.

Let's firstly calculate the total sales (number of orders) made by Olist and then analyze how it has changed over time (i.e., periodical trend) –

```
# Olist fulfilled nearly 99K orders over a time period of 2 years (i.e., Sept-2016 to Oct-2018).
select count(order_id) as num_of_orders
from `olist_business_case.orders`
where order_status != 'canceled';
```

This script will process 823.42 MB when run.

Query results

Job information | Results | Chart | JSON | Execution details | Execution graph

Row	num_of_orders
1	98816

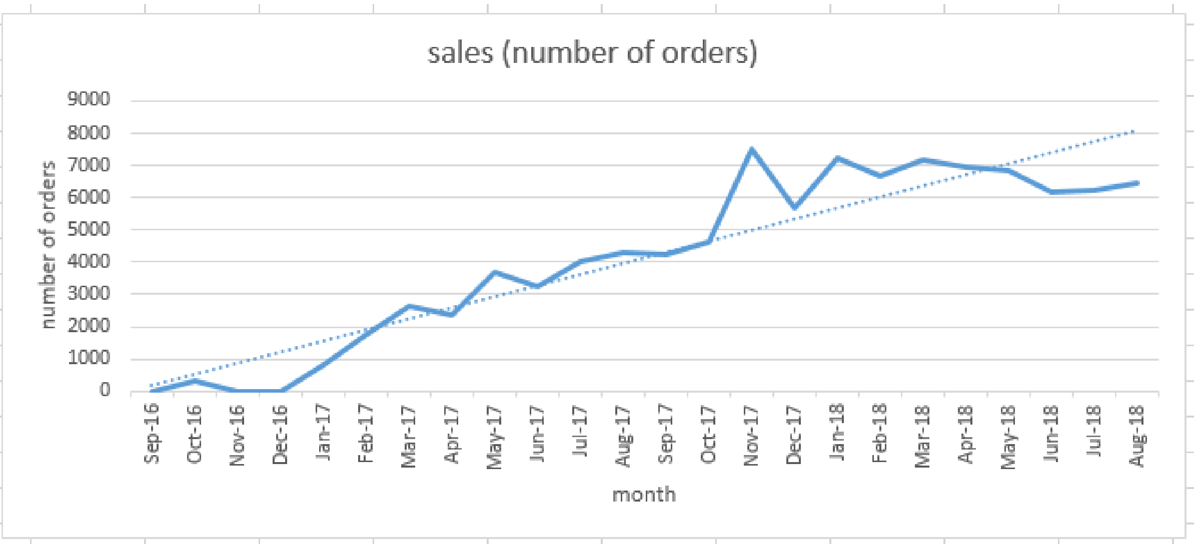
Month-on-month change in sales volume (number of orders) –

```
select
  time_period,
  current_month_sales,
  CONCAT(CAST(ROUND((current_month_sales - prev_month_sales)/prev_month_sales*100, 2) AS STRING), ' %') as percentage_change
from (
    select
      time_period,
      num_of_orders as current_month_sales,
      LAG(num_of_orders, 1) OVER(ORDER BY SUBSTR(time_period, -1, 4), SUBSTR(time_period, 1, 2)) as prev_month_sales
    from (
        select
          FORMAT_TIMESTAMP('%m-%Y', order_purchase_timestamp) as time_period,
          count(order_id) as num_of_orders
        from `olist_business_case.orders`
        where order_status != 'canceled'
        group by FORMAT_TIMESTAMP('%m-%Y', order_purchase_timestamp)
      ) as tbl
  ) as tbl2
order by SUBSTR(time_period, -1, 4), SUBSTR(time_period, 1, 2);
```

Query results

Job information | Results | Chart | JSON | Execution details | Exe

Row	time_period	current_month_sales	percentage_change
1	09-2016	2	null
2	10-2016	300	14900 %
3	12-2016	1	-99.67 %
4	01-2017	797	79600 %
5	02-2017	1763	121.2 %
6	03-2017	2649	50.26 %
7	04-2017	2386	-9.93 %
8	05-2017	3671	53.86 %
9	06-2017	3229	-12.04 %
10	07-2017	3998	23.82 %
11	08-2017	4304	7.65 %
12	09-2017	4265	-0.91 %
13	10-2017	4605	7.97 %
14	11-2017	7507	63.02 %
15	12-2017	5662	-24.58 %
16	01-2018	7235	27.78 %
17	02-2018	6655	-8.02 %
18	03-2018	7185	7.96 %
19	04-2018	6924	-3.63 %
20	05-2018	6849	-1.08 %
21	06-2018	6149	-10.22 %
22	07-2018	6251	1.66 %
23	08-2018	6428	2.83 %



- The **sales trend analysis** reveals a generally **consistent upward trajectory**, with only minor dips that are outweighed by periods of growth. Given the low percentage of repeat customers, the increasing sales figures suggest that Olist is successfully attracting a **growing number of new customers**. These trends are encouraging indicators of strong market traction and business development during the company's early years.

Now, let's confirm if the revenue growth is also following a similar trajectory as the sales (number of orders). Basically, we will calculate the total revenue generated and then analyze how it has changed over time (i.e., periodical trend) –

```sql
# Olist generated a total revenue of approx. $16M (or brazilian currency)
# over a time period of 2 years (i.e., Sept 2016 to Oct 2018).
select ROUND(sum(payment_value), 2) as total_revenue
from `olist_business_case.payments`
where order_id IN (select
                        order_id
                      from `olist_business_case.orders`
                      where order_status != 'canceled'
                      );
```

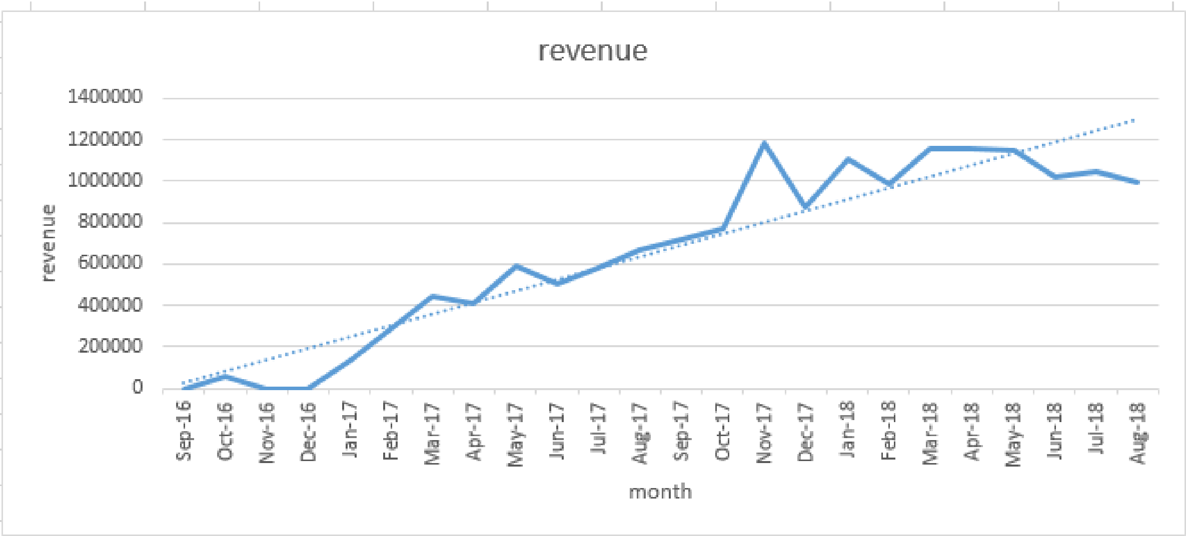
This script will process 823.42 MB when run.

Query results

Job information | Results | Chart | JSON | Execution details | Exe

Row	total_revenue
1	15865616.52

Month-on-month change in revenue –



- The **revenue trend analysis** closely mirrors the positive trajectory observed in sales.
- This **upward movement in revenue**, despite occasional minor declines, **reflects sustained business growth.**
- The **consistent rise in revenue, alongside growing sales volumes and an expanding customer base** during the early years, indicates that Olist is **successfully scaling its operations and strengthening its presence in the e-commerce sector**.

Note: It would be insightful to analyze **monthly seasonality** in sales or revenue numbers to determine if certain months experience significant peaks or dips. However, the current dataset of just **24 months is insufficient** to make any reliable conclusions about any such seasonal trends.

Next, let's examine the **distribution of sales across different states and cities** to identify the top and bottom performing regions –

```
select
  *,
  CONCAT(CAST(ROUND(num_of_orders/sum(num_of_orders) OVER()*100, 2) AS STRING), ' %') as percent_of_orders
from
    (select
      c.customer_state,
      count(o.order_id) as num_of_orders
    from `olist_business_case.orders` as o
    INNER JOIN `olist_business_case.customers` as c
    ON o.customer_id = c.customer_id
    where o.order_status != 'canceled'
    group by c.customer_state
  )
order by num_of_orders desc;
```

Top 10 States by Sales Volume (Number of Orders) –

Row	customer_state	num_of_orders	percent_of_orders
1	SP	41419	41.92 %
2	RJ	12766	12.92 %
3	MG	11571	11.71 %
4	RS	5441	5.51 %
5	PR	5023	5.08 %
6	SC	3618	3.66 %
7	BA	3364	3.4 %
8	DF	2133	2.16 %
9	ES	2024	2.05 %
10	GO	2007	2.03 %

- Nearly **42%** of all orders originate from the state of **Sao Paulo (SP)** alone, highlighting its significant contribution to Olist's overall sales. The next largest shares come from **Rio de Janeiro (RJ)** and **Minas Gerais (MG)**, accounting for **13%** and **12%** of orders, respectively. The distribution reflects the demographic and economic realities of Brazil as these three states are not only the **most populous** but also the **most economically active**, together representing a substantial portion of the country's GDP.

- The **top 3 states account for 67% of total orders**, while the top 10 states contribute 90%, indicating a **strong concentration of sales within a limited number of regions**. This high concentration suggests significant market penetration in these states, but also highlights opportunities for growth in less represented regions.

- Also, note that high sales figures in these leading states mainly **represent the concentration of a substantial customer base in these regions**, as 97% of the customers on the platform are one-time purchasers.

Bottom 10 States by Sales Volume (Number of Orders) –

18	PI	491	0.5 %
19	RN	485	0.49 %
20	AL	412	0.42 %
21	SE	349	0.35 %
22	TO	279	0.28 %
23	RO	250	0.25 %
24	AM	148	0.15 %
25	AC	81	0.08 %
26	AP	68	0.07 %
27	RR	45	0.05 %

- The **bottom 10 states collectively contribute less than 3% to the overall sales volume**, indicating a significantly lower level of market penetration and sales activity in these regions as compared to the leading states.
- The possible reasons behind such **low sales activity in these regions** could be –
    - lower adoption of online shopping habits, particularly among rural populations
    - low digital literacy
    - lower average income levels
    - limited internet access

- Further, such low sales activity makes it cost-ineffective for companies like Olist to serve these regions efficiently.
- These combined **socio-economic and infrastructural barriers** hinder consumer's participation in e-commerce and thereby the company's ability to expand its reach in these states.

Next, we identify the top cities in terms of sales volume. And as expected, the **top-performing cities are the leading states' capitals** – sao paulo (capital of Sao Paulo, SP) leads with 16% of total sales volume, followed by rio de janeiro (capital of Rio de Janeiro, RJ) with 7%, and Belo Horizonte (capital of Minas Gerais, MG) with 3%.

Now, let's examine the **revenue distribution across states** to figure out if they follow the same pattern observed in sales distribution or not –

```
select
  *,
  CONCAT(CAST(ROUND(revenue/sum(revenue) OVER()*100, 2) AS STRING), ' %') as percent_of_revenue
from (
      select
        c.customer_state,
        ROUND(sum(p.payment_value), 2) as revenue
      from `olist_business_case.orders` as o
      INNER JOIN `olist_business_case.payments` as p
      ON o.order_id = p.order_id
      INNER JOIN `olist_business_case.customers` as c
      ON o.customer_id = c.customer_id
      where o.order_status != 'canceled'
      group by c.customer_state
    ) as tbl
order by revenue desc;
```

Top 10 States by Revenue generated –

Row	customer_state	revenue	percent_of_revenue
1	SP	5942397.11	37.45 %
2	RJ	2126444.23	13.4 %
3	MG	1856375.81	11.7 %
4	RS	881680.6	5.56 %
5	PR	802319.18	5.06 %
6	SC	613707.46	3.87 %
7	BA	611796.01	3.86 %
8	DF	352718.04	2.22 %
9	GO	342124.8	2.16 %
10	ES	324038.9	2.04 %

- The analysis of revenue distribution across states reveal a pattern closely aligned with sales volume, with **Sao Paulo (SP) contributing 37.45% of total revenue**, followed by **Rio de Janeiro (RJ) at 13.4% and Minas Gerais (MG) at 11.7%.**
- Collectively, the **top 5 states account for 73%** of total revenue, while the top 10 states contribute 87%. And the **bottom 10 states collectively contribute less than 4%** of the total revenue.
- The close alignment between sales and revenue distribution across states suggests that the **average order value (AOV) is largely uniform across states.**

Next, let's examine the **distribution of sales across different product categories** to identify the top and bottom performing categories –

```
select
  *,
  CONCAT(CAST(ROUND(num_of_orders/(sum(num_of_orders) over())*100, 2) AS STRING), ' %') as percent_of_orders
from (
      select
        p.product_category_name,
        count(order_id) as num_of_orders,
      from
      (select *
      from `olist_business_case.order_items`
      where order_id IN (select order_id from `olist_business_case.orders` where order_status != 'canceled')) as oi
      INNER JOIN `olist_business_case.products` as p
      ON oi.product_id = p.product_id
      group by p.product_category_name
    )
order by num_of_orders desc;
```

Top 10 Product categories by Sales Volume (Number of Orders) –

Row	product_category_name	num_of_orders	percent_of_orders
1	bed table bath	11097	9.9 %
2	HEALTH BEAUTY	9634	8.59 %
3	sport leisure	8590	7.66 %
4	Furniture Decoration	8298	7.4 %
5	computer accessories	7781	6.94 %
6	housewares	6915	6.17 %
7	Watches present	5970	5.33 %
8	telephony	4527	4.04 %
9	Garden tools	4328	3.86 %
10	automotive	4205	3.75 %

- The **top 10 product categories account for a substantial 64% share of total sales volume**. Many of these categories, such as household goods and health and beauty products represent **everyday essentials, explaining their high demand.**
- Also, the top 20 product categories (out of a total of 74) accounts for 88% of total sales volume.
- These figures indicate that **customer demand is heavily focused on a limited selection of categories.** This insight is valuable for **inventory management**, as it enables the company to strategically prioritize stocking and replenishment for these high-demand categories.

Now that we have identified the product categories with high and low demand, let's examine the **pricing** within these categories. This will provide us valuable insights into the **purchasing capacity/behavior of the customers** –

```
select distinct
  p.product_category_name,
  ROUND(PERCENTILE_CONT(oi.price, 0.5) OVER(PARTITION BY p.product_category_name), 2) as median_price,
  ROUND(AVG(oi.price) OVER(PARTITION BY p.product_category_name), 2) as avg_price,
  COUNT(oi.order_id) OVER(PARTITION BY p.product_category_name) as num_of_orders,
  ROUND(COUNT(oi.order_id) OVER(PARTITION BY p.product_category_name)/(COUNT(oi.order_id) OVER())*100, 2)
  as percent_of_orders
from
(select *
from `olist_business_case.order_items`
where order_id IN (select order_id from `olist_business_case.orders` where order_status != 'canceled')) as oi
INNER JOIN `olist_business_case.products` as p
ON oi.product_id = p.product_id
order by num_of_orders desc;
```

Mostly, **product categories having high Sales Volume have median prices below 100 dollars** (or Brazil currency) –

Row	product_category_name	median_price	avg_price	num_of_orders	percentage_of_orders
1	bed table bath	79.9	93.36	11097	9.9 %
2	HEALTH BEAUTY	79.9	130.34	9634	8.59 %
3	sport leisure	78.0	114.06	8590	7.66 %
4	Furniture Decoration	65.49	87.67	8298	7.4 %
5	computer accessories	81.99	116.22	7781	6.94 %
6	housewares	59.8	90.65	6915	6.17 %
7	Watches present	129.0	200.7	5970	5.33 %
8	telephony	29.99	71.2	4527	4.04 %
9	Garden tools	59.9	111.14	4328	3.86 %
10	automotive	84.9	139.51	4205	3.75 %

Mostly, **product categories having higher price shows negligible sales** –

Row	product_category_name	median_price	avg_price	num_of_orders	percentage_of_orders
1	PCs	1100.0	1098.34	203	0.18 %
2	HOUSE PASTALS OVEN AND C...	587.0	624.29	76	0.07 %
3	Agro Industria e Comercio	258.65	342.12	212	0.19 %
4	ELECTRICES 2	227.99	470.85	235	0.21 %
5	Furniture	179.0	183.75	109	0.1 %
6	Furniture office	144.99	161.88	1690	1.51 %
7	insurance and services	141.64	141.65	2	0 %
8	climatization	139.99	185.5	295	0.26 %
9	La Cuisine	137.0	146.78	14	0.01 %
10	Cool Stuff	129.99	164.24	3780	3.37 %

To get further insights into the purchasing capacity/behavior of the customers, let's examine the **distribution of sales across different price segments** –

```
with master_table as (
  select
    product_id,
    count(order_id) as sales_volume,
    avg(price) as price
  from `olist_business_case.order_items`
  group by product_id
),

price_categories as (
  select
    *,
    case
      when price < 100 then '<100'
      when price between 100 and 200 then '100-200'
      when price between 200 and 500 then '200-500'
      else '>500'
    end as price_range
  from master_table
)

select distinct
  price_range,
  sum(sales_volume) over(partition by price_range) as total_sales_volume,
  CONCAT(CAST(ROUND(100*sum(sales_volume) over(partition by price_range)/(sum(sales_volume) over()), 2) AS STRING), ' %')
  as percent_of_sales_volume
from price_categories
order by case
          when price_range = '<100' then 1
          when price_range = '100-200' then 2
          when price_range = '200-500' then 3
          else 4
        end;
```

Row	price_range	total_sales_volume	percent_of_sales_volume
1	<100	70914	62.95 %
2	100-200	27958	24.82 %
3	200-500	10502	9.32 %
4	>500	3276	2.91 %

- The **vast majority of orders (nearly 63%) come from products priced under 100 dollars** (or Brazil currency). Products priced between 100 and 200 dollars account for about 25% of total sales volume. So, **overall, 88% of total sales volume is coming from products priced below 200 dollars**.

- Almost all **high-volume categories** that we identified before, such as household goods and health and beauty products, **fall into the lower price range of under 100 dollars**.

- These observations highlight **largely price-sensitive and utility-driven shopping behavior of customers**, suggesting that demand is largely driven by affordability and everyday needs, rather than premium or niche products. However, **mid-range products are also getting significant traction.**

- Additionally, the fact that **97% of customers on the platform are one-time buyers (first-time customers)** suggests **limited trust or confidence in the platform which could be discouraging the customers from purchasing premium (expensive) products**, further reinforcing the preference for lower/moderate priced items.

Furthermore, since **high-volume categories are priced similarly**, the revenue distribution will largely follow sales patterns, which means these product categories will be the **primary revenue drivers for the company.**

Mode of payments –

```
select
  *,
  CONCAT(CAST(ROUND(num_of_orders/sum(num_of_orders) OVER()*100, 2) AS STRING), ' %') as percent_of_orders
from (
      select
        payment_type,
        count(order_id) as num_of_orders
      from `olist_business_case.payments`
      group by payment_type
    )
order by num_of_orders desc;
```

Row	payment_type	num_of_orders	percent_of_orders
1	credit_card	76795	73.92 %
2	boleto	19784	19.04 %
3	voucher	5775	5.56 %
4	debit_card	1529	1.47 %

- Nearly **74% of all transactions are completed via credit cards** – meaning almost three out of every four transactions are done using credit cards, making it the most preferred payment method among customers. This also reflects a **purchasing behavior that is largely credit-driven.**
- Olist could **collaborate with leading credit card providers** to facilitate seamless card payments for its customers while offering attractive **perks such as reward points, cashback offers, special discounts**, etc. This would help in enhancing the overall **user experience**.

- The **'Boleto' payment method**, a popular alternative in Brazil, accounts for almost one in five orders (**19%**), **highlighting its continued relevance** for customers who may not have access to credit cards or prefer not to use them.

Row	payment_type	revenue	percent_of_total_revenue
1	credit_card	12542084.19	78.34 %
2	boleto	2869361.27	17.92 %
3	voucher	379436.87	2.37 %
4	debit_card	217989.79	1.36 %

- The **distribution of revenue across payment methods closely follows the pattern observed in transaction volume.** Credit cards account for 78% of total revenue, which aligns with their 74% share of total transactions. Similarly, 'Boleto' contributes 18% of revenue, similar to its 19% share of transactions.

- This close alignment between transaction volume and revenue share for each payment method suggests a largely **uniform Average Order Value (AOV) across payment methods.**

Next, we will examine the preferences for installment options to make the payment.

```
select
  distinct payment_type
from `olist_business_case.payments`
where payment_installments > 1;
```

This script will process 851.81 MB when run.

Query results

Job information | Results | Chart

Row	payment_type
1	credit_card

- Among all payment methods, only credit cards offer the option of paying in more than one installment.

```
select
  CONCAT(CAST(ROUND(100*countif(payment_installments > 1)/count(*), 2) AS STRING),' %')
  as percent_of_installment_transactions
from `olist_business_case.payments`;
```

This script will process 851.81 MB when run.

Query results

Job information | Results | Chart | JSON | Execution details | Execution graph

Row	percent_of_installment_transactions
1	49.42 %

- **Nearly half (49.4%) of all transactions are made using installment plans**, **indicating a strong customer preference** for flexible payment options.

- Given this strong preference, Olist should prioritize making **installment plans readily available and prominently displayed during the checkout process**. By doing so, the company can **enhance customer convenience**, reduce financial barriers to purchase – especially for higher-value items—and **potentially increase conversion rates** and overall sales.

Analysis of Review scores –

Review scores are a direct feedback from the customers representing their overall purchasing experience and that's why review scores can be seen as a valuable **measure of customer satisfaction**.

Let's begin by examining the **distribution of review scores** to gauge overall customer sentiment –

```
select
  *,
  CONCAT(CAST(ROUND(num_of_orders/(sum(num_of_orders) OVER())*100, 2) AS STRING), ' %') as percent_of_orders
from (
      select
        review_score,
        count(distinct order_id) as num_of_orders
      from `olist_business_case.reviews`
      group by review_score
    ) as tbl
order by num_of_orders desc;
```

Row	review_score	num_of_orders	percent_of_orders
1	5	57076	57.73 %
2	4	19098	19.32 %
3	1	11393	11.52 %
4	3	8160	8.25 %
5	2	3148	3.18 %

- Nearly **58% of all orders have received the highest rating of 5** and overall, approximately **77% of all orders have ratings of 4 or 5.** These figures **indicate a strong level of customer satisfaction and positive purchasing experience.**

- Less than **12% of all the orders received the lowest rating of 1** and overall, approximately **15% of all orders have ratings of 1 or 2**. These figures suggest that while most customers are satisfied with their purchasing experience, a **significant minority had a poor experience**.

- Also, seeing such high proportion of positive review scores, we can comment that **purchasing experience is not the reason behind such low rate of repeat customers (3%)**.

Let's further deep-dive into these review scores to identify the best and worst performing product categories –

To assess the best and worst-performing product categories based on customer reviews, we will calculate the proportion of **low (review score <= 3)** and **high (review score > 3)** rated reviews for each product category. This will help us **identify categories having large proportion of *"low rated reviews"* (priority for improvement) or *"high rated reviews"* (potential bestsellers).**

Note: To ensure statistical reliability, we will consider only those categories for our analysis that have received more than 100 reviews (median 'number of reviews' for product categories was calculated to be 244).

```
WITH master_table as (

  select distinct
    oi.order_id,
    p.product_category_name,
    r.review_score
  from `olist_business_case.order_items` as oi
  INNER JOIN `olist_business_case.products` as p
  ON oi.product_id = p.product_id
  INNER JOIN `olist_business_case.reviews` as r
  ON oi.order_id = r.order_id
)

select
  product_category_name,
  ROUND(100*countif(review_score <= 3)/count(review_score), 2) as percentage_of_low_ratings
from master_table
group by product_category_name
having count(order_id) > 100
order by percentage_of_low_ratings desc;
```

5 best performing product categories based on customer reviews –

Row	product_category_name	percentage_of_high_ratings
1	General Interest Books	88.39
2	technical books	86.38
3	Bags Accessories	83.98
4	Drink foods	83.7
5	foods	83.6

5 worst performing product categories based on customer reviews –

Row	product_category_name	percentage_of_low_ratings
1	Furniture office	37.28
2	House comfort	32.49
3	Fashion Men's Clothing	32.43
4	audio	31.61
5	CONSTRUCTION SECURITY TO...	28.31

- Among the 52 product categories with sufficient reviews (over 100 reviews), 29 categories outperformed the overall platform benchmark i.e., more than 77% of their reviews received high (review score > 3) ratings. Furthermore, among these, the following **2 categories – *'General Interest Books'* and *'technical books'* performed exceptionally well, with more than 85% of their reviews receiving high ratings.**

- Performance of the following **4 categories – *'Furniture Office'*, *'House comfort'*, *'Fashion Men's Clothing'*, and *'audio'*** signals significant customer dissatisfaction as more than 30% of their reviews received low (review score <= 3) ratings.

Furthermore, let's assess the **review scores of the key product categories** that we previously identified as the main contributors to Olist's overall sales and revenue –

Row	product_category_name	percent_of_orders	percentage_of_low_ratings
1	bed table bath	9.9 %	26.88
2	HEALTH BEAUTY	8.59 %	20.49
3	sport leisure	7.66 %	20.31
4	Furniture Decoration	7.4 %	25.62
5	computer accessories	6.94 %	24.31
6	housewares	6.17 %	21.73
7	Watches present	5.33 %	23.83
8	telephony	4.04 %	25.48
9	Garden tools	3.86 %	21.45
10	automotive	3.75 %	22.28

- There are **no significant performance concerns among these high-volume product categories**, which reflects an overall positive customer response to Olist's most in-demand product segments.

- Nevertheless, given their substantial impact on total sales, it remains crucial for the company to continually strive for service improvements and higher customer ratings. For instance, while the 'bed table bath' category leads in sales accounting for nearly 10% of total sales volume, it still receives 27% low ratings (review scores ≤ 3).

Timely delivery is often considered as a major factor influencing customer experience in online shopping. So, we will now explore whether there is a correlation between order delivery times and the review scores customers have provided.

Note: For this analysis of order delivery times, we will only consider those 97% orders that have been *'delivered'* as for the remaining 3% orders, the delivery timestamp is not available yet.

```
select
  r.review_score,
  ROUND(avg(d.delivery_time_in_days), 2) as avg_delivery_time_in_days
from
    (select
      order_id,
      TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp, SECOND)/86400
      as delivery_time_in_days
    from `olist_business_case.orders`
    where order_status = 'delivered') as d
INNER JOIN
    (select
      order_id,
      review_score
    from `olist_business_case.reviews`) as r
ON d.order_id = r.order_id
group by r.review_score
order by avg_delivery_time_in_days desc;
```

Row	review_score	avg_delivery_time_in_days
1	1	21.31
2	2	16.66
3	3	14.26
4	4	12.31
5	5	10.69

- The data suggests a **clear and strongly negative correlation between average delivery time and customer ratings** – longer delivery times correspond to lower customer ratings.

- **Orders with the lowest ratings (1 and 2) have significantly longer average delivery times** (21.3 and 16.7 days, respectively), while orders with the highest ratings (4 and 5) are delivered much faster (12.3 and 10.7 days, respectively).

- Moreover, Orders that received a 1-star rating experienced an average delivery time of 21.3 days, which is **almost double the delivery time for orders rated 4 or 5** (12.3 and 10.7 days, respectively). This stark difference underscores the **crucial role of timely delivery in shaping customer satisfaction.**

Order delivery time –

In the previous section, we saw a clear and strongly negative correlation between average delivery time and customer satisfaction – longer delivery times correspond to lower customer ratings. Therefore, let's examine the order delivery times to identify potential areas for improvement and opportunities to shorten delivery times –

To better understand and optimize the order fulfillment process, we have identified **three key stages that each order passes through**:

- the **approval stage** (from order placement to approval),
- the **waiting stage** (from approval to handover to the logistics partner),
- and, the **shipping stage** (from carrier handover to final delivery to the customer).

Let's examine the average time taken at each of these stages –

```
select distinct
  'approval_time (in hours)' as different_stages,
  ROUND(AVG(TIMESTAMP_DIFF(order_approved_at, order_purchase_timestamp, SECOND)/3600) OVER(), 2) as mean,
  ROUND(PERCENTILE_CONT(TIMESTAMP_DIFF(order_approved_at, order_purchase_timestamp, SECOND)/3600, 0.5) OVER(), 2) as median
from `olist_business_case.orders`

UNION ALL

select distinct
  'waiting_time (in days)' as different_stages,
  ROUND(AVG(TIMESTAMP_DIFF(order_delivered_carrier_date, order_approved_at, SECOND)/86400) OVER(), 2) as mean,
  ROUND(PERCENTILE_CONT(TIMESTAMP_DIFF(order_delivered_carrier_date, order_approved_at, SECOND)/86400, 0.5) OVER(), 2) as median
from `olist_business_case.orders`

UNION ALL

select distinct
  'shipping_time (in days)' as different_stages,
  ROUND(AVG(TIMESTAMP_DIFF(order_delivered_customer_date, order_delivered_carrier_date, SECOND)/86400) OVER(), 2) as mean,
  ROUND(PERCENTILE_CONT(TIMESTAMP_DIFF(order_delivered_customer_date, order_delivered_carrier_date, SECOND)/86400, 0.5) OVER(), 2) as median
from `olist_business_case.orders`
where order_status = 'delivered'

order by CASE
            WHEN different_stages = 'approval_time (in hours)' THEN 1
            WHEN different_stages = 'waiting_time (in days)' THEN 2
            ELSE 3
          END;
```

Row	different_stages	mean	median
1	approval_time (in hours)	10.42	0.34
2	waiting_time (in days)	2.81	1.82
3	shipping_time (in days)	9.33	7.1

- The **median approval time is exceptionally fast at just 20 minutes**, indicating that **approvals might be automated** and thus doesn't require any manual intervention.

- However, the large gap between the median and mean approval times (20 minutes vs. 10.4 hours) highlights that large number of orders experience substantial delays – over 27% of orders take more than 10 hours for approval and **over 17% of orders take more than 24 hours for approval**. These significant delays **may be due to requirement of manual reviews** or exceptional circumstances.

- Once approved, **sellers typically hand over orders to logistics partners within two days**, which is competitive by industry standards and **suggests efficient seller operations**.

- The **shipping stage**, with a **median duration of 7.1 days** and a mean of 9.3 days, emerges as the **primary contributor to overall delivery time**.

- To further enhance customer satisfaction and reduce delivery times, **efforts should focus on investigating and optimizing** both the outlier approval cases and, most importantly, the shipping process.

Next, we will evaluate Olist's **delivery efficiency** by **examining its ability to deliver the order on time**. For this we will analyze the proportion of orders delivered early, on time, and late based on the estimated delivery date the customer was informed during order placement –

```
select
  'Early delivery' as delivery,
  CONCAT(CAST(ROUND(100*countif(extract(date from order_delivered_customer_date) < extract(date from order_estimated_delivery_date))/count(*), 2) AS STRING), ' %')
  as percent_of_orders
from `olist_business_case.orders`
where order_status = 'delivered'

UNION ALL

select
  'On-time delivery' as delivery,
  CONCAT(CAST(ROUND(100*countif(extract(date from order_delivered_customer_date) = extract(date from order_estimated_delivery_date))/count(*), 2) AS STRING), ' %')
  as percent_of_orders
from `olist_business_case.orders`
where order_status = 'delivered'

UNION ALL

select
  'Late delivery' as delivery,
  CONCAT(CAST(ROUND(100*countif(extract(date from order_delivered_customer_date) > extract(date from order_estimated_delivery_date))/count(*), 2) AS STRING), ' %')
  as percent_of_orders
from `olist_business_case.orders`
where order_status = 'delivered'

order by case
          when delivery = 'Early delivery' then 1
          when delivery = 'On-time delivery' then 2
          else 3
        end;
```

Row	delivery	percent_of_orders
1	Early delivery	91.88 %
2	On-time delivery	1.34 %
3	Late delivery	6.77 %

- The data reveals an **outstanding delivery efficiency**, with nearly **92% of orders getting delivered earlier than the promised delivery date**. This positions Olist as a reliable and customer-centric platform in terms of order fulfillment.

- However, the **disproportionately high *"early deliveries"* suggests that Olist's estimated delivery windows may be overly cautious/conservative**, resulting in most orders arriving ahead of schedule. Let's validate it by examining the difference between *'average actual order delivery times'* and *'average estimated delivery times'* –

```
select distinct
  'actual_delivery_time' as delivery_time_in_days,
  ROUND(AVG(TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp, SECOND)/86400) OVER(), 2) as mean,
  ROUND(PERCENTILE_CONT(TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp, SECOND)/86400, 0.5) OVER(), 2)as median
from `olist_business_case.orders`
where order_status = 'delivered'

UNION ALL

select distinct
  'estimated_delivery_time' as delivery_time_in_days,
  ROUND(AVG(TIMESTAMP_DIFF(order_estimated_delivery_date, order_purchase_timestamp, SECOND)/86400) OVER(), 2) as mean,
  ROUND(PERCENTILE_CONT(TIMESTAMP_DIFF(order_estimated_delivery_date, order_purchase_timestamp, SECOND)/86400, 0.5) OVER(), 2) as median
from `olist_business_case.orders`
where order_status = 'delivered'

order by CASE
          WHEN delivery_time_in_days = 'actual_delivery_time' THEN 1
          ELSE 2
        END;
```

Row	delivery_time_in_days	mean	median
1	actual_delivery_time	12.56	10.22
2	estimated_delivery_time	23.74	23.23

- As anticipated, there is a **significantly large gap – nearly double – between Olist's estimated and actual delivery times**, confirming that the company's **delivery estimates are highly conservative**.
- Olist should consider refining its forecasting of estimated delivery times as longer estimated delivery times informed to customers may have unintended consequences. **Overly long delivery estimates at checkout could deter potential customers from completing their purchases.**