1. Dataset Introduction

- Short background of Olist dataset (Brazilian e-commerce).
- Purpose: customer, sales, and delivery analysis.

2. Table Details

- **Customers Table** – columns explained (ID, unique ID, zip, city, state).
- **Geolocation Table** – latitude, longitude, city, state.
- **Orders Table** – purchase, approval, delivery, estimated delivery.
- **Order Items Table** – product, seller, shipping date, price, freight.
- **Order Reviews Table** – scores, comments, review timeline.
- **Payments Table** – payment method, installments, value.
- **Products Table** – category, dimensions, weight.
- **Sellers Table** – seller location details.

3. Relationships (ERD Style)

- Customers ↔ Orders
- Orders ↔ Order Items
- Orders ↔ Reviews
- Orders ↔ Payments
- Order Items ↔ Products
- Order Items ↔ Sellers
- Customers ↔ Geolocation

4. Example SQL Queries (with explanations)

- Revenue by product category.
- Average review score per seller.
- Late vs on-time delivery.
- Top 10 states by order volume.
- Payment type distribution.

Key Joins Between Tables

- **customers ↔ orders**

  customers.customer_id = orders.customer_id

- **orders ↔ order_items**

  orders.order_id = order_items.order_id

- **orders ↔ order_reviews**

  orders.order_id = order_reviews.order_id

- **orders ↔ payments**

  orders.order_id = payments.order_id

- **order_items ↔ products**

  order_items.product_id = products.product_id

- **order_items ↔ sellers**

  order_items.seller_id = sellers.seller_id

- **customers ↔ geolocation** (via ZIP prefix)

  customers.customer_zip_code_prefix = geolocation.geolocation_zip_code_prefix

Table: customers.csv

Column Name	Description	Example Value
customer_id	System-generated unique ID for each customer (used for joining with orders).	06b8999e2fba1a1fbc88172c00ba8bc7
customer_unique_id	Permanent unique identifier for the same customer (can place multiple orders).	861eff4711a542e4b93843c6dd7febb0
customer_zip_code_prefix	First 5 digits of customer's ZIP code.	14409
customer_city	City where the customer is located.	são paulo
customer_state	State code of the customer (2-letter).	SP

Table: geolocation.csv

Column Name	Description	Example Value
geolocation_zip_code_prefix	ZIP code prefix (used to map customers/sellers).	14409
geolocation_lat	Latitude coordinate.	-23.545621
geolocation_lng	Longitude coordinate.	-46.639292
geolocation_city	City name.	são paulo
geolocation_state	State code (2-letter).	SP

Table: orders.csv

Column Name	Description	Example Value
order_id	Unique identifier for each order.	e481f51cbdc54678b7cc49136f2d6af7
customer_id	Foreign key linking to customers table.	06b8999e2fba1a1fbc88172c00ba8bc7
order_status	Current status of the order (delivered, shipped, etc.).	delivered
order_purchase_timestamp	Date and time when the order was placed.	2017-08-15 10:10:10
order_approved_at	Timestamp when payment was approved.	2017-08-15 11:00:00
order_delivered_carrier_date	Date when order was handed to carrier.	2017-08-16 15:00:00
order_delivered_customer_date	Date when order was delivered to customer.	2017-08-18 14:00:00
order_estimated_delivery_date	Estimated delivery date promised to customer.	2017-08-20

Table: order_items.csv

Column Name	Description	Example Value
order_id	Foreign key linking to orders table.	e481f51cbdc54678b7cc49136f2d6af7
order_item_id	Item number within the same order.	1

Column Name	Description	Example Value
product_id	Foreign key linking to products table.	1e9e8ef04dbcff4541ed26657ea517e5
seller_id	Foreign key linking to sellers table.	3442f8959a84dea7ee197c632cb2df15
shipping_limit_date	Latest date seller should ship the item.	2017-08-17 00:00:00
price	Price of the product item.	99.90
freight_value	Shipping cost for the item.	20.00

Table: order_review.csv

Column Name	Description	Example Value
review_id	Unique identifier for each review.	a5489108a7ea0a35d82f0e5e4d8afaa3
order_id	Foreign key linking to orders table.	e481f51cbdc54678b7cc49136f2d6af7
review_score	Rating given by customer (1–5).	5
review_comment_title	Short title for the review.	Excellent
review_creation_date	Date when review was created.	2017-08-21
review_answer_timestamp	Date when review was answered.	2017-08-22 14:00:00

Table: payments.csv

Column Name	Description	Example Value
order_id	Foreign key linking to orders table.	e481f51cbdc54678b7cc49136f2d6af7
payment_sequential	Payment attempt sequence (1st, 2nd, etc.).	1
payment_type	Mode of payment (credit card, boleto, etc.).	credit card
payment_installments	Number of installments if applicable.	3

Column Name	Description	Example Value
payment_value	Total amount paid.	150.00

Table: products.csv

Column Name	Description	Example Value
product_id	Unique identifier for each product.	1e9e8ef04dbcff4541ed26657ea517e5
product_category	Category of the product.	furniture_bedroom
product_name_length	Number of characters in product name.	50
product_description_length	Number of characters in product description.	500
product_photos_qty	Number of product images uploaded.	2
product_weight_g	Weight of product in grams.	1500
product_length_cm	Length of product in centimeters.	40
product_height_cm	Height of product in centimeters.	10
product_width_cm	Width of product in centimeters.	30

Table: sellers.csv

Column Name	Description	Example Value
seller_id	Unique identifier for each seller.	3442f8959a84dea7ee197c632cb2df15
seller_zip_code_prefix	ZIP code prefix for seller location.	13023
seller_city	City of the seller.	campinas
seller_state	State code of the seller (2-letter).	SP

Problem Statement:

Assuming you are a data analyst/ scientist at Target, you have been assigned the task of analysing the given dataset to extract valuable insights and provide

actionable recommendations. What does 'good' look like?

1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:
    1. Data type of all columns in the "customers" table.
    2. Get the time range between which the orders were placed.
    3. Count the Cities & States of customers who ordered during the given period.
2. In-depth Exploration:
    1. Is there a growing trend in the no. of orders placed over the past years?
    2. Can we see some kind of monthly seasonality in terms of the no. of orders being placed?
    3. During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)
        - 0-6 hrs: Dawn
        - 7-12 hrs: Mornings
        - 13-18 hrs: Afternoon
        - 19-23 hrs: Night
3. Evolution of E-commerce orders in the Brazil region:
    1. Get the month on month no. of orders placed in each state.
    2. How are the customers distributed across all the states?
4. Impact on Economy: Analyse the money movement by e-commerce by looking at order prices, freight and others.
    1. Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).

You can use the "payment value" column in the payments table to get the cost of orders.

    1. Calculate the Total & Average value of order price for each state.
    2. Calculate the Total & Average value of order freight for each state.

5. Analysis based on sales, freight and delivery time.

1. Find the no. of days taken to deliver each order from the order’s purchase date as delivery time.

Also, calculate the difference (in days) between the estimated & actual delivery date of an order.

Do this in a single query.

You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula:

- time_to_deliver = order_delivered_customer_date - order_purchase_timestamp
- diff_estimated_delivery = order_delivered_customer_date - order_estimated_delivery_date

2. Find out the top 5 states with the highest & lowest average freight value.
3. Find out the top 5 states with the highest & lowest average delivery time.
4. Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.

You can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.

6. Analysis based on the payments:
    1. Find the month on month no. of orders placed using different payment types.
    2. Find the no. of orders placed on the basis of the payment installments that have been paid.

```sql
#Import the dataset and do usual exploratory analysis steps like checking the structure &
characteristics of the dataset:
#1. Data type of all columns in the customers table.
#2. Get the time range between which the orders were placed.
select *
from `SQL_TARGET.customers`
limit 10;

select *
from `SQL_TARGET.geolocation`
limit 5;

#  Get the time range between which the orders were placed.

select
min(order_purchase_timestamp) as start_time,
max(order_purchase_timestamp) as end_time
from `SQL_TARGET.orders`;

# Display count the cities and states of customer who ordered during the given time

select
c.customer_city , c.customer_state
from `SQL_TARGET.orders` as o
join `SQL_TARGET.customers` as c
on o.customer_id = c.customer_id
where extract(YEAR FROM o.order_purchase_timestamp) = 2018
and extract(month from o.order_purchase_timestamp) between 1 and 3;

# is there a growing trend in the no. of orders placed over the past years?

 select
 extract(month from order_purchase_timestamp) as month,
 count(order_id) as order_num
 from `SQL_TARGET.orders`
 group by extract(month from order_purchase_timestamp)
 order by order_num desc;

 # during what time of the day , do the brazilian customers mostaly place their orders?
(Dawn, Morning,Afternoon or Night)
 # 0-6 hrs: Dawn
 # 7-12 hrs: Morning
 # 13-18 hrs: Afternoon
 # 19-23 hrs: Night

 select
 extract(hour from order_purchase_timestamp) as time,
 count(order_id) as order_num
 from `SQL_TARGET.orders`
 group by extract(hour from order_purchase_timestamp)
 order by order_num desc;
```

```sql
 # get month on month number of orders

 select
 extract(month from order_purchase_timestamp) as month,
 extract(year from order_purchase_timestamp) as year,
 count(*) as num_orders
 from `SQL_TARGET.orders`
 group by year,month
 order by year,month;

 SELECT
    EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
    EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
    COUNT(*) OVER(PARTITION BY EXTRACT(MONTH FROM order_purchase_timestamp)) AS
orders_per_month,
    COUNT(*) OVER(PARTITION BY EXTRACT(YEAR FROM order_purchase_timestamp)) AS
orders_per_year
FROM `SQL_TARGET.orders`
ORDER BY year, month;

# how are the customers distributed across all the states?

select customer_state,customer_city,
count(distinct customer_id) as customer_count
from `SQL_TARGET.customers`
group by customer_state,customer_city
order by customer_count;

SELECT
    customer_state,
    COUNT(DISTINCT customer_id) OVER(PARTITION BY customer_state) AS customer_count
FROM `SQL_TARGET.customers`
ORDER BY customer_count;

SELECT DISTINCT
    customer_state,
    customer_city,
    COUNT(DISTINCT customer_id) OVER(PARTITION BY customer_state, customer_city) AS
customer_count
FROM `SQL_TARGET.customers`
ORDER BY customer_count;

# get the % increase in the cost of orders from year 2017 to 2018 (include months between
jan to aug only)
# you can use the "pyment_value" column n the payments table to get the cost of orders.


# STEP 1: CALCULATE Total payments per year

SELECT DISTINCT
  EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
  SUM(p.payment_value) OVER (PARTITION BY EXTRACT(YEAR FROM o.order_purchase_timestamp)) AS
total_payment
FROM `SQL_TARGET.payments` AS p
JOIN `SQL_TARGET.orders` AS o
```

```
  ON p.order_id = o.order_id;



with yearly_totals as (
select
extract (year from o.order_purchase_timestamp) as year,
sum(p.payment_value) as total_payment
from `SQL_TARGET.payments`as p
join SQL_TARGET.orders as o
on p.order_id = o.order_id
where extract (year from o.order_purchase_timestamp) in (2017,2018)
and extract (month from o.order_purchase_timestamp) between 1 and 8
group by extract (year from o.order_purchase_timestamp)
),

# STEP 2: Use LEAD window function to compare each year's payments with the previous year

yearly_comparisons as (
select
year,total_payment,
lead(total_payment) over (order by year desc) as prev_year_payment
from yearly_totals
)

# STEP 3: Calculate % increase

select
((total_payment - prev_year_payment) / prev_year_payment)*100
from yearly_comparisons;
# mean & sum of price and fright value by customer

select
c.customer_state,
avg(price) as avg_price,
sum(price) as sum_price,
avg(freight_value) as avg_freight,
sum(freight_value) as sum_freight
from `SQL_TARGET.orders` as o
join `SQL_TARGET.order_items` as oi
on o.order_id = oi.order_id
join `SQL_TARGET.customers` as c
on o.customer_id = c.customer_id
group by c.customer_state;

# calculate days between purchasing,delivering and estimated delivery.

select
order_id,
date_diff(date(order_delivered_customer_date), date(order_purchase_timestamp), day) as
days_to_delivery,
date_diff(date(order_delivered_customer_date), date(order_estimated_delivery_date), day) as
diff_estimated_delivery
from `SQL_TARGET.orders`;
```

```sql
# find out the top 5 states with the highest and lowest average freight value.

select
c.customer_state,
avg(freight_value) as avg_freight_value
from `SQL_TARGET.orders` as o
join `SQL_TARGET.order_items` as oi
on o.order_id = oi.order_id
join `SQL_TARGET.customers` as c
on o.customer_id = c.customer_id
group by customer_state
order by avg_freight_value desc
limit 5;

# find out the top 5 states with the highest & lowest average delivery time.

select
c.customer_state,
avg(extract(date from o.order_delivered_customer_date) - extract(date from
o.order_purchase_timestamp)) as avg_time_to_delivery
from `SQL_TARGET.orders` as o
join `SQL_TARGET.order_items` as oi
on o.order_id = oi.order_id
join `SQL_TARGET.customers` as c
on o.customer_id = c.customer_id
group by customer_state
order by avg_time_to_delivery desc # this is lowest and for highest desc change to asc
limit 5;


# find the month on month no. of orders placed using different payment types.

select
payment_type,
extract(year from order_purchase_timestamp) as year,
extract(month from order_purchase_timestamp) as month,
count(distinct o.order_id) as order_count
from `SQL_TARGET.orders` as o
inner join `SQL_TARGET.payments` as p
on o.order_id = p.order_id
group by payment_type,year,month
order by payment_type,year,month;

# count of order based on the number of payment installments

select
payment_installments,
count(distinct order_id) as num_orders
from `SQL_TARGET.payments`
group by payment_installments;
```