```sql
DROP DATABASE IF EXISTS target;
CREATE DATABASE target;


-- Table: customers
CREATE TABLE customers (
    customer_id text,
    customer_unique_id text,
    customer_zip_code_prefix integer,
    customer_city text,
    customer_state text
);

-- Table: orders
CREATE TABLE orders (
    order_id text,
    customer_id text,
    order_status text,
    order_purchase_timestamp timestamp,
    order_approved_at timestamp,
    order_delivered_carrier_date timestamp,
    order_delivered_customer_date timestamp,
    order_estimated_delivery_date timestamp
);

-- Table: order_items
CREATE TABLE order_items (
    order_id text,
    order_item_id integer,
    product_id text,
    seller_id text,
    shipping_limit_date timestamp,
    price double precision,
    freight_value double precision
);

-- Table: payments
CREATE TABLE payments (
    order_id text,
    payment_sequential integer,
    payment_type text,
    payment_installments integer,
    payment_value double precision
);

-- Table: products
CREATE TABLE products (
    product_id text,
    product_category text,
    product_name_length double precision,
    product_description_length double precision,
    product_photos_qty double precision,
    product_weight_g double precision,
```

```sql
    product_length_cm double precision,
    product_height_cm double precision,
    product_width_cm double precision
);


-- Table: sellers
CREATE TABLE sellers (
    seller_id text,
    seller_zip_code_prefix integer,
    seller_city text,
    seller_state text
);


-- Table: order_reviews
CREATE TABLE order_reviews (
    review_id text,
    order_id text,
    review_score integer,
    review_comment_title text,
    review_creation_date text,
    review_answer_timestamp text
);


-- Table: geolocation
CREATE TABLE geolocation (
    geolocation_zip_code_prefix integer,
    geolocation_lat double precision,
    geolocation_lng double precision,
    geolocation_city text,
    geolocation_state text
);
```

1) Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset

1.1) Data type of all columns in the "customers" table.

```sql
SELECT
  column_name,
  data_type,
  is_nullable,
  character_maximum_length
FROM information_schema.columns
WHERE table_schema = 'public' AND table_name = 'customers'
ORDER BY ordinal_position;
```

column_name name	data_type character varying	is_nullable character varying (3)	character_maximum_length integer
1 customer_id	text	YES	[null]
2 customer_unique_id	text	YES	[null]
3 customer_zip_code_prefix	integer	YES	[null]
4 customer_city	text	YES	[null]
5 customer_state	text	YES	[null]

1.2) Get the time range between which the orders were placed.
SELECT
  MIN(order_purchase_timestamp) AS first_order_ts,
  MAX(order_purchase_timestamp) AS last_order_ts,
  COUNT(*) AS total_orders
FROM public.orders;

first_order_ts timestamp without time zone	last_order_ts timestamp without time zone	total_orders bigint
1 2016-09-04 21:15:19	2018-10-17 17:30:18	99441

1.3) Count the Cities & States of customers who ordered during the given period.

SELECT
  COUNT(DISTINCT c.customer_city)   AS distinct_cities,
  COUNT(DISTINCT c.customer_state) AS distinct_states
FROM public.customers c
JOIN public.orders o
  ON c.customer_id = o.customer_id;

distinct_cities bigint	distinct_states bigint
1 2994	27

2) In-depth Exploration

2.1) Is there a growing trend in the no. of orders placed over the past years?

```
select
        extract(year from order_purchase_timestamp) as year,
        count(order_id) as total_orders
from orders
group by extract(year from order_purchase_timestamp)
order by extract(year from order_purchase_timestamp)
```

	year numeric 🔒	total_orders bigint 🔒
1	2016	329
2	2017	45101
3	2018	54011

2.2) Can we see some kind of monthly seasonality in terms of the no. of orders being placed?

select
        extract(month from order_purchase_timestamp) as year,
        count(order_id) as total_orders
from orders
group by extract(month from order_purchase_timestamp)
order by extract(month from order_purchase_timestamp)

	year numeric 🔒	total_orders bigint 🔒
1	1	8069
2	2	8508
3	3	9893
4	4	9343
5	5	10573
6	6	9412
7	7	10318
8	8	10843
9	9	4305
10	10	4959

2.3)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

select
        count(case when extract (hour from order_purchase_timestamp) between 0 and 6 then 1 end) as dawn,
        count(case when extract (hour from order_purchase_timestamp) between 7 and 12 then 1 end) as mornings,
        count(case when extract (hour from order_purchase_timestamp) between 13 and 18 then 1 end) as
afternoon,

count(case when extract (hour from order_purchase_timestamp) between 19 and 23 then 1 end) as night
from orders

	dawn bigint	mornings bigint	afternoon bigint	night bigint
1	5242	27733	38135	28331

3) Evolution of E-commerce orders in the Brazil region.

3.1) Get the month on month no. of orders placed in each state.

select
      c.customer_state,
      extract(month from o.order_purchase_timestamp) as month,
      extract(year from o.order_purchase_timestamp) as year,
      count(o.order_id) as no_of_orders
from customers as c
join orders as o
on c.customer_id = o.customer_id
group by c.customer_state, extract(month from o.order_purchase_timestamp), extract(year from o.order_purchase_timestamp)
order by c.customer_state, extract(month from o.order_purchase_timestamp), extract(year from o.order_purchase_timestamp)

	customer_state text	month numeric	year numeric	no_of_orders bigint
1	AC	1	2018	2
2	AC	2	2017	2
3	AC	2	2018	2
4	AC	3	2017	1
5	AC	3	2018	1
6	AC	4	2017	2
7	AC	5	2017	1
8	AC	5	2018	1
9	AC	6	2017	1
10	AC	7	2017	1

3.2) How are the customers distributed across all the states?

select
      customer_state,
      count(customer_id) as total_customers
from customers
group by customer_state

order by count(customer_id) desc

	customer_state 🔒 text	total_customers 🔒 bigint
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637
7	BA	3380
8	DF	2140
9	ES	2033
10	GO	2020

4) Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.

4.1) 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.

```
SELECT
  SUM(CASE WHEN EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2017
       AND EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8
      THEN p.payment_value ELSE 0 END) AS total_2017,
  SUM(CASE WHEN EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2018
       AND EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8
      THEN p.payment_value ELSE 0 END) AS total_2018,
  (
    SUM(CASE WHEN EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2018
         AND EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8
        THEN p.payment_value ELSE 0 END)
    -
    SUM(CASE WHEN EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2017
         AND EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8
        THEN p.payment_value ELSE 0 END)
  ) * 100.0
  /
  NULLIF(
    SUM(CASE WHEN EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2017
         AND EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8
        THEN p.payment_value ELSE 0 END),
    0
  ) AS pct_increase
FROM orders o
```

JOIN payments p ON o.order_id = p.order_id;

	total_2017 double precision 🔒	total_2018 double precision 🔒	pct_increase double precision 🔒
1	1407425.8300000043	3259796.6299999873	131.61409720610126

4.2) Calculate the Total & Average value of order price for each state.

select
      distinct c.customer_state,
      round(sum(payment_value)::numeric,2) as total,
      round(avg(payment_value)::numeric,2) as avg
from orders as o
join customers as c
on o.customer_id = c.customer_id
join payments as p
on o.order_id=p.order_id
group by c.customer_state

	customer_state text 🔒	total numeric 🔒	avg numeric 🔒
1	AC	1305.19	186.46
2	AL	14119.53	247.71
3	AM	1874.41	144.19
4	AP	2424.75	242.48
5	BA	83089.00	164.86
6	CE	37260.09	187.24
7	DF	45494.59	147.71
8	ES	55902.72	176.35
9	GO	62446.07	185.85
10	MA	18446.11	196.24

4.3) Calculate the Total & Average value of order freight for each state.

select
      c.customer_state,
      round(sum(i.freight_value)::numeric,2) as total,
      round(avg(i.freight_value)::numeric,2) as avg
from customers as c
join orders as o
on c.customer_id=o.customer_id
join order_items as i
on o.order_id=i.order_id

group by c.customer_state

	customer_state text	total numeric	avg numeric
1	AC	224.35	32.05
2	AL	1803.73	31.64
3	AM	384.44	29.57
4	AP	339.32	33.93
5	BA	12935.75	24.88
6	CE	6741.46	31.65
7	DF	6630.31	20.15
8	ES	7794.87	22.59
9	GO	9041.06	24.11
10	MA	3679.59	36.43

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

5.1) 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.

```
select
        order_id,
        extract (days from (order_delivered_customer_date - order_purchase_timestamp)) as delivery_days,
        extract (days from (order_delivered_customer_date - order_estimated_delivery_date)) as delivery_days
from orders
where order_status = 'delivered'
```

	order_id text	delivery_days numeric	delivery_days numeric
1	e481f51cbdc54678b7cc49136f2d6af7	8	-7
2	"53cdb2fc8bc7dce0b6741e2150273451"	13	-5
3	"47770eb9100c2d0c44946d9cf07ec65d"	9	-17
4	"949d5b44dbf5de918fe9c16f97b45f8a"	13	-12
5	ad21c59c0840e6cb83a9ceb5573f8159	2	-9
6	a4591c265e18cb1dcee52889e2d8acc3	16	-5
7	"6514b8ad8028c9f2cc2374ded245783f"	9	-11
8	"76c6e866289321a7c93b82b54852dc3...	9	-31
9	e69bfb5eb88e0ed6a785585b27e16dbf	18	-6
10	e6ce16cb79ec1d90b1da9085a6118aeb	12	-8

5.2) Find out the top 5 states with the highest & lowest average freight value.

```
WITH cte1 AS (
    SELECT
        c.customer_state,
        AVG(i.freight_value) AS avg_freight,
        'low'::text AS type
    FROM customers c
    JOIN orders o ON c.customer_id = o.customer_id
    JOIN order_items i ON o.order_id = i.order_id
    GROUP BY c.customer_state
    ORDER BY AVG(i.freight_value)
    LIMIT 5
),
cte2 AS (
    SELECT
        c.customer_state,
        AVG(i.freight_value) AS avg_freight,
        'high'::text AS type
    FROM customers c
    JOIN orders o ON c.customer_id = o.customer_id
    JOIN order_items i ON o.order_id = i.order_id
    GROUP BY c.customer_state
    ORDER BY AVG(i.freight_value) DESC
    LIMIT 5
)
SELECT *
FROM cte1
UNION ALL
SELECT *
```

```
FROM cte2
ORDER BY type, avg_freight;
```

	customer_state 🔒 text	avg_freight 🔒 double precision	type 🔒 text
1	PI	38.61958333333333	high
2	SE	42.27521739130435	high
3	RO	42.940769230769234	high
4	RR	57.748333333333335	high
5	PB	57.85393258426965	high
6	SP	15.02084179104475	low
7	DF	20.152917933130702	low
8	PR	20.357605294825536	low
9	MG	20.97614736842105	low
10	RJ	20.978513064132994	low

5.3) Find out the top 5 states with the highest & lowest average delivery time.

```
with cte1 as(
select
        c.customer_state,
        round(avg(extract(day from (order_delivered_customer_date - order_purchase_timestamp))),2) as
avg_delivery_time,
        'High'::text as type
from customers as c
join orders as o
on c.customer_id=o.customer_id
group by c.customer_state
order by avg(extract(day from (order_delivered_customer_date - order_purchase_timestamp))) desc
limit 5
),

cte2 as(
select
        c.customer_state,
        round(avg(extract(day from (order_delivered_customer_date - order_purchase_timestamp))),2) as
avg_delivery_time,
        'Low'::text as type
from customers as c
join orders as o
on c.customer_id=o.customer_id
group by c.customer_state
order by avg(extract(day from (order_delivered_customer_date - order_purchase_timestamp)))
limit 5
```

```
)
select *
from cte1
union all
select *
from cte2
```

	customer_state 🔒 text	avg_delivery_time 🔒 numeric	type 🔒 text
1	RR	24.63	High
2	AL	24.61	High
3	AP	23.71	High
4	PA	23.29	High
5	AM	23.21	High
6	SP	8.29	Low
7	MG	11.47	Low
8	PR	11.58	Low
9	DF	12.62	Low
10	SC	14.48	Low

5.4) 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.

```
select
        c.customer_state,
        round(avg(extract(day from (o.order_estimated_delivery_date - o.order_purchase_timestamp))),2)
        -(select round(avg(extract(day from (order_estimated_delivery_date - order_purchase_timestamp))),2) from
orders)
        as avg_days_earlier
from customers as c
join orders as o
on c.customer_id = o.customer_id
group by c.customer_state
order by round(avg(extract(day from (o.order_estimated_delivery_date - o.order_purchase_timestamp))),2)
        -(select round(avg(extract(day from (order_estimated_delivery_date - order_purchase_timestamp))),2) from
orders) desc
limit 5
```

	customer_state text	avg_days_earlier numeric
1	AM	22.41
2	RR	22.19
3	AP	21.42
4	AC	19.20
5	RO	14.20

6) Analysis based on the payments.

6.1) Find the month on month no. of orders placed using different payment types.

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

	year numeric	month numeric	payment_type text	total bigint
1	2016	9	credit_card	1
2	2016	10	credit_card	88
3	2016	10	debit_card	1
4	2016	10	UPI	28
5	2016	10	voucher	3
6	2016	12	credit_card	1
7	2017	1	credit_card	208
8	2017	1	debit_card	6
9	2017	1	UPI	84
10	2017	1	voucher	12

6.2) Find the no. of orders placed on the basis of the payment installments that have been paid.

select

```
        payment_installments,
        count(distinct order_id) as total_orders
from payments
where payment_installments>1
group by payment_installments
```

	payment_installments integer 🔒	total_orders bigint 🔒
1	2	12389
2	3	10443
3	4	7088
4	5	5234
5	6	3916
6	7	1623
7	8	4253
8	9	644
9	10	5315
10	11	23

Actionable insights

1. Customer & Order Trends
- **Growing orders year-over-year** → Brazilian e-commerce is expanding, showing demand growth.
  Actionable: Invest in scaling logistics and infrastructure; strengthen partnerships with delivery carriers.
- **Seasonality (monthly variation)** → If certain months show spikes, there may be seasonality (e.g. Black Friday, holidays).
  Actionable: Plan **marketing campaigns, inventory stocking, and staffing** around those high-demand months.
- **Time-of-day order behavior** → Customers place most orders in **afternoon/evening**.
  Actionable: Schedule **targeted ads / push notifications** during these hours for maximum conversion.

2. Regional Distribution
- **Top customer states** → A few states contribute the majority of orders (likely SP, RJ, MG, RS).
  Actionable: Focus marketing, localized offers, and faster logistics in those high-volume regions.
- **States with fewer customers** → Opportunity to expand awareness and delivery reliability.
  Actionable: Offer **shipping discounts** or partner with local carriers in underserved states.

3. Economy / Money Movement
- **Payment value increased ~X% (2017 → 2018, Jan–Aug)** → Strong revenue growth.
  Actionable: Allocate more budget to customer acquisition and product diversification.
- **Freight costs significant in some states** → Certain regions have very high average freight.
  Actionable: Optimize **distribution centers** or use **regional warehouses** to reduce shipping costs.
- **States with low average order value (AOV)** → Some markets spend less per order.
  Actionable: Encourage higher basket size with **bundled offers, free shipping thresholds**.

4. Delivery Performance

- **Average delivery times vary widely by state** → Some states have much slower deliveries.
  Actionable: Improve **last-mile delivery partnerships** in high-delay regions.
- **Some states deliver much faster than estimated** → Customer expectations are being exceeded there.
  Actionable: Use this **positive gap** in marketing campaigns ("Delivered faster than expected!").
- **Top 5 slowest states** → Risk of poor customer satisfaction.
  Actionable: Monitor closely, adjust estimated delivery dates, or offer **compensation/discounts** for late deliveries.

5. Payment Behavior

- **Majority of customers use credit cards** (likely) with **installments** being popular.
  Actionable: Promote installment options (0% EMI) in ads to attract more buyers.
- **Alternative payment methods (boleto, vouchers) also present**.
  Actionable: Continue supporting multiple payment types for financial inclusion.
- **Installments >1 common in large purchases**.
  Actionable: Bundle products or offer financing to increase high-value purchases.

6. Reviews & Customer Feedback *(if analyzed further)*

- **States or products with poor reviews** → May overlap with long delivery times.
  Actionable: Use review insights to **improve seller quality and delivery SLAs**.