# Business Case: SQL (Table Name: Target)

SQL based data analysis performed on Brazilian E-Commerce Public Dataset by Olist (https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce).

This particular business case focuses on the operations of Olist in Brazil and provides insightful information about 100,000 orders placed between 2016 and 2018. The dataset offers a comprehensive view of various dimensions including the order status, price, payment and freight performance, customer location, product attributes, and customer reviews.

By analyzing this extensive dataset, it becomes possible to gain valuable insights into Olist's operations in Brazil. The information can shed light on various aspects of the business, such as order processing, pricing strategies, payment and shipping efficiency, customer demographics, product characteristics, and customer satisfaction levels.

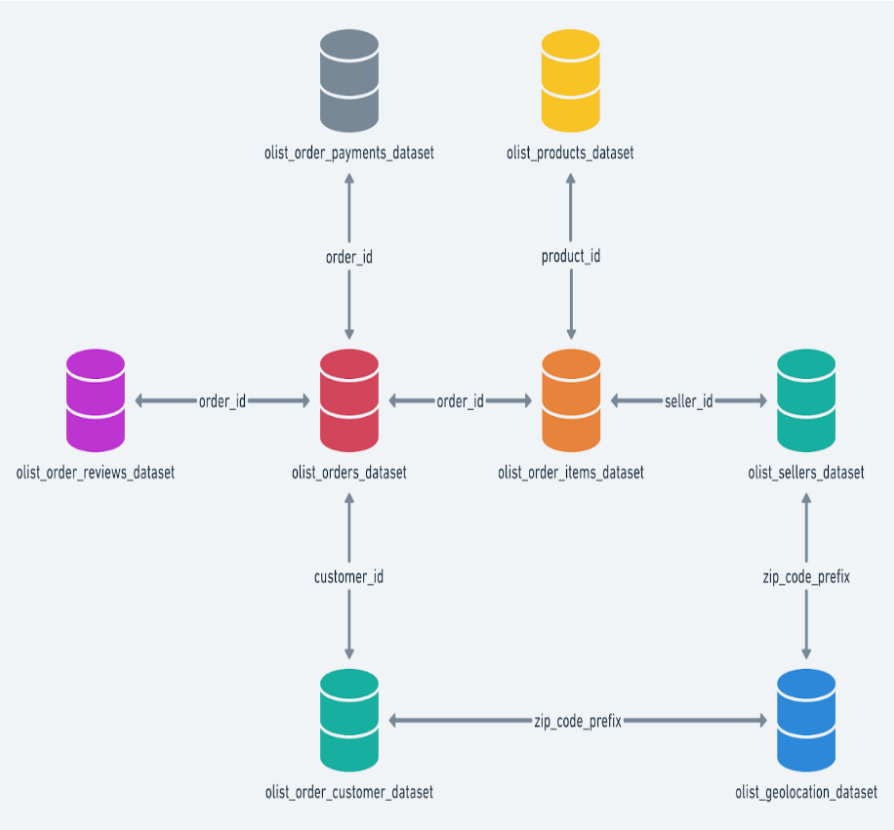


*Figure 1. Database Schema*

The data is available in 8 csv files:

- customers.csv
- sellers.csv
- order_items.csv
- geolocation.csv
- payments.csv
- reviews.csv
- orders.csv
- products.csv

The column description for these csv files is given below.

The **customers.csv** contain following features:

| **Features** | **Description** |
|---|---|
| customer_id | ID of the consumer who made the purchase |
| customer_unique_id | Unique ID of the consumer |
| customer_zip_code_prefix | Zip Code of consumer’s location |
| customer_city | Name of the City from where order is made |
| customer_state | State Code from where order is made (Eg. são paulo - SP) |

The **sellers.csv** contains following features:

| **Features** | **Description** |
|---|---|
| seller_id | Unique ID of the seller registered |
| seller_zip_code_prefix | Zip Code of the seller’s location |
| seller_city | Name of the City of the seller |
| seller_state | State Code (Eg. são  aulo – SP) |

The **order_items.csv** contain following features:

| **Features** | **Description** |
|---|---|
| order_id | A Unique ID of order made by the consumers |
| order_item_id | A Unique ID given to each item ordered in the order |
| product_id | A Unique ID given to each product available on the site |
| seller_id | Unique ID of the seller registered in Target |
| shipping_limit_date | The date before which the ordered product must be shipped |
| price | Actual price of the products ordered |
| freight_value | Price rate at which a product is delivered from one point to another |

The **geolocations.csv** contain following features:

| **Features** | **Description** |
|---|---|
| geolocation_zip_code_prefix | First 5 digits of Zip Code |
| geolocation_lat | Latitude |
| geolocation_lng | Longitude |
| geolocation_city | City |
| geolocation_state | State |

The **payments.csv** contain following features:

| Features | Description |
|---|---|
| order_id | A Unique ID of order made by the consumers |
| payment_sequential | Sequences of the payments made in case of EMI |
| payment_type | Mode of payment used (Eg. Credit Card) |
| payment_installments | Number of installments in case of EMI purchase |
| payment_value | Total amount paid for the purchase order |

The **orders.csv** contain following features:

| Features | Description |
|---|---|
| order_id | A Unique ID of order made by the consumers |
| customer_id | ID of the consumer who made the purchase |
| order_status | Status of the order made i.e. delivered, shipped, etc. |
| order_purchase_timestamp | Timestamp of the purchase |
| order_delivered_carrier_date | Delivery date at which carrier made the delivery |
| order_delivered_customer_date | Date at which customer got the product |
| order_estimated_delivery_date | Estimated delivery date of the products |

The **reviews.csv** contain following features:

| Features | Description |
|---|---|
| review_id | ID of the review given on the product ordered by the order id |
| order_id | A Unique ID of order made by the consumers |
| review_score | Review score given by the customer for each order on a scale of 1-5 |
| review_comment_title | Title of the review |
| review_comment_message | Review comments posted by the consumer for each order |
| review_creation_date | Timestamp of the review when it is created |
| review_answer_timestamp | Timestamp of the review answered |

The **products.csv** contain following features:

| Features | Description |
|---|---|
| product_id | A Unique identifier for the proposed project. |
| product_category_name | Name of the product category |
| product_name_lenght | Length of the string which specifies the name given to the products ordered |
| product_description_lenght | Length of the description written for each product ordered on the site |
| product_photos_qty | Number of photos of each product ordered available on the shopping portal |
| product_weight_g | Weight of the products ordered in grams |
| product_length_cm | Length of the products ordered in centimeters |
| product_height_cm | Height of the products ordered in centimeters |
| product_width_cm | Width of the product ordered in centimeters |

## 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.

```
select
  column_name,
  data_type
from
  target.INFORMATION_SCHEMA.COLUMNS
where
  table_name = 'customers'
```

| 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 |

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

```
select
  min(order_purchase_timestamp) as first_order,
  max(order_purchase_timestamp) as last_order
from
  `target.orders`
```

| Row | first_order | last_order |
|---|---|---|
| 1 | 2016-09-04 21:15:19 UTC | 2018-10-17 17:30:18 UTC |

### 3. Count the Cities & States of customers who ordered during the given period.

```
select
  count(distinct t1.customer_city) as number_of_cities,
  count(distinct t1.customer_state) as number_of_states
from
  `target.customers` t1 inner join `target.orders` t2
  on t1.customer_id = t2.customer_id
```

| Row | number_of_cities | number_of_states |
|---|---|---|
| 1 | 4119 | 27 |

## 2. In-depth Exploration:

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,
  extract(month from order_purchase_timestamp) as month,
  count(*) as num_orders
from
  `target.orders`
group by 1,2
order by 1,2
```

| Row | year | month | num_orders |
|---|---|---|---|
| 1 | 2016 | 9 | 4 |
| 2 | 2016 | 10 | 324 |
| 3 | 2016 | 12 | 1 |
| 4 | 2017 | 1 | 800 |
| 5 | 2017 | 2 | 1780 |
| 6 | 2017 | 3 | 2682 |
| 7 | 2017 | 4 | 2404 |
| 8 | 2017 | 5 | 3700 |
| 9 | 2017 | 6 | 3245 |

A growing trend is visible in the number of orders placed as years go by.

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 month,
  count(*) as num_orders
from
  `target.orders`
group by 1
order by 2 desc
```

| Row | month | num_orders |
|---|---|---|
| 1 | 8 | 10843 |
| 2 | 5 | 10573 |
| 3 | 7 | 10318 |
| 4 | 3 | 9893 |
| 5 | 6 | 9412 |
| 6 | 4 | 9343 |
| 7 | 2 | 8508 |
| 8 | 1 | 8069 |

As we can see throughout all the years, month of August has seen most number of orders being placed. In general, the top 6 months which saw most orders lie from march to July. Brazilian summer lies from December through march while winter lies from June through September. The increased number of orders from march to August might indicate, people were preparing for their winter season by ordering the essentials.

3. During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)
    1. 0-6 hrs : Dawn
    2. 7-12 hrs : Mornings
    3. 13-18 hrs : Afternoon
    4. 19-23 hrs : Night

```sql
with cte1 as
(select
  extract(time from order_purchase_timestamp) as order_time,
  count(*) as num_orders
from
  `target.orders`
group by 1 )

select
  case when (order_time between '00:00:00' and '06:59:59') then 'Dawn'
  when (order_time between '07:00:00' and '12:59:59') then 'Mornings'
  when (order_time between '13:00:00' and '18:59:59') then 'Afternoon'
  else 'Night'
  end as time_period,
  sum(num_orders) as num_orders
from
  cte1
group by 1
order by 2 desc
```

| Row | time_period | num_orders |
|---|---|---|
| 1 | Afternoon | 38135 |
| 2 | Night | 28331 |
| 3 | Mornings | 27733 |
| 4 | Dawn | 5242 |

As we can notice, Brazilian customers mostly placed their orders in the afternoon. (note the time of the day was not clearly specified in the question hence timing was assumed as such – e.g. 0-6 = 00:00:00 to 06:59:59 i.e. before start of 7 as so on)

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

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

```
select
  extract(month from order_purchase_timestamp) as month,
  t2.customer_state as state,
  count(*) as num_orders
from
  `target.orders` t1 inner join `target.customers` t2
  on t1.customer_id = t2.customer_id
group by 1,2
order by 3 desc
```

| Row | month | state | num_orders |
|---|---|---|---|
| 1 | 8 | SP | 4982 |
| 2 | 5 | SP | 4632 |
| 3 | 7 | SP | 4381 |
| 4 | 6 | SP | 4104 |
| 5 | 3 | SP | 4047 |
| 6 | 4 | SP | 3967 |

As we can notice, Sao Paulo state in the month of August has seen the maximum number of orders being placed throughout the years. Sao Paulo being the most populated state in Brazil over the last decade may be one of the reasons for high number of orders seen.

## 2. How are the customers distributed across all the states?

```
select
  customer_state,
  count(customer_id) as num_customers
from
  `target.customers`
group by 1
order by 2 desc
```

| Row | customer_state | num_customers |
|---|---|---|
| 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 |

customer_state

As observed in previous analysis, we can notice as Sao Paolo has maximum number of customers. We can similarly check the state with least number of customers ( State – Roraima, least populous in Brazil ) and try to aquire more people from this city, may be by providing offers or discounts etc.

## 4. Impact on Economy: Analyze 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.

```
--- orders from year 2017 to 2018 (Months between Jan - Aug)
with
cte1 as
(select
  order_id,
  extract(year from order_purchase_timestamp) as order_year,
  extract(month from order_purchase_timestamp) as order_month
from
  `target.orders`
where
  extract(month from order_purchase_timestamp) in (1,2,3,4,5,6,7,8)
  and
  extract(year from order_purchase_timestamp) in (2017,2018)),

--- total cost of all orders for 2017, 2018
cte2 as
(select
  t1.order_year as order_year,
  sum(t2.payment_value) as total_cost_of_orders
from
  cte1 t1 inner join `target.payments` t2
  on t1.order_id = t2.order_id
group by 1),

--- total cost of orders for 2017
cte3 as
(select
  total_cost_of_orders as initial
from cte2
where order_year = 2017),

--- total cost of orders for 2018
cte4 as
(select
  total_cost_of_orders as final
from cte2
where order_year = 2018)

--- % Increase
select
  round((final-initial)/initial * 100 , 2) as percentage_increase
from
  cte3, cte4
```

| Row | percentage_increase |
|---|---|
| 1 | 136.98 |

As we can see, total the cost of all the orders more than doubled for 2018 compared to 2017 for months between Jan – Aug. This may be due to increase in prices of the products as well as increase in customers. This validates our growing trend in number of orders placed.

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

```
with
cte1 as
(select
  c.customer_state as state,
  round(sum(i.price),2) as total_amount,
  round(count(distinct o.order_id),2) as total_orders
from
  `target.orders` o inner join `target.order_items` i on o.order_id = i.order_id
  inner join `target.customers` c on o.customer_id = c.customer_id
group by state)

select
  state,
  total_amount,
  total_orders,
  round(total_amount/total_orders,2) as avg_price
from cte1
order by total_amount desc
```

| Row | state | total_amount | total_orders | avg_price |
|---|---|---|---|---|
| 1 | SP | 5202955.05 | 41375.0 | 125.75 |
| 2 | RJ | 1824092.67 | 12762.0 | 142.93 |
| 3 | MG | 1585308.03 | 11544.0 | 137.33 |
| 4 | RS | 750304.02 | 5432.0 | 138.13 |
| 5 | PR | 683083.76 | 4998.0 | 136.67 |
| 6 | SC | 520553.34 | 3612.0 | 144.12 |
| 7 | BA | 511349.99 | 3358.0 | 152.28 |
| 8 | DF | 302603.94 | 2125.0 | 142.4 |
| 9 | GO | 294591.95 | 2007.0 | 146.78 |
| 10 | ES | 275037.31 | 2025.0 | 135.82 |

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

```
with
cte1 as
(select
  c.customer_state as state,
  round(sum(i.freight_value ),2) as total_freight_value,
  round(count(distinct o.order_id),2) as total_orders
from
  `target.orders` o inner join `target.order_items` i on o.order_id = i.order_id
  inner join `target.customers` c on o.customer_id = c.customer_id
group by state)

select
  state,
  total_freight_value,
  total_orders,
  round(total_freight_value/total_orders,2) as avg_freight_value
from cte1
order by total_freight_value desc
```

| Row | state | total_freight_value | total_orders | avg_freight_value |
|---|---|---|---|---|
| 1 | SP | 718723.07 | 41375.0 | 17.37 |
| 2 | RJ | 305589.31 | 12762.0 | 23.95 |
| 3 | MG | 270853.46 | 11544.0 | 23.46 |
| 4 | RS | 135522.74 | 5432.0 | 24.95 |
| 5 | PR | 117851.68 | 4998.0 | 23.58 |
| 6 | BA | 100156.68 | 3358.0 | 29.83 |
| 7 | SC | 89660.26 | 3612.0 | 24.82 |
| 8 | PE | 59449.66 | 1648.0 | 36.07 |
| 9 | GO | 53114.98 | 2007.0 | 26.46 |
| 10 | DF | 50625.5 | 2125.0 | 23.82 |

We can notice, although the total freight value for Sao Paulo is highest, its average value is very low compared to the rest. This is likely because most orders are from Sao Paulo which will make the delivery/shipment company deliver orders in bulk which will cost less individually and make up for the shipment cost with cost of total orders delivered together.

## 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_estimated_delivery_date - order_delivered_customer_date

```
select
  order_purchase_timestamp,
  order_estimated_delivery_date,
  order_delivered_customer_date,
  date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as
time_to_deliver,
  date_diff(order_estimated_delivery_date, order_delivered_customer_date, day) as
diff_estimated_delivery
from `target.orders`
where
  (order_delivered_customer_date is not Null) and
  (order_purchase_timestamp is not Null) and
  (order_estimated_delivery_date is not Null)
order by diff_estimated_delivery
```

| Row | order_purchase_timestamp | order_estimated_delivery_date | order_delivered_customer_date | time_to_deliver | diff_estimated_delivery |
|---|---|---|---|---|---|
| 1 | 2018-02-23 14:57:35 UTC | 2018-03-15 00:00:00 UTC | 2018-09-19 23:24:07 UTC | 208 | -188 |
| 2 | 2017-02-21 23:31:27 UTC | 2017-03-22 00:00:00 UTC | 2017-09-19 14:36:39 UTC | 209 | -181 |
| 3 | 2018-01-03 09:44:01 UTC | 2018-01-19 00:00:00 UTC | 2018-07-13 20:51:31 UTC | 191 | -175 |
| 4 | 2017-03-13 20:17:10 UTC | 2017-04-05 00:00:00 UTC | 2017-09-19 17:00:07 UTC | 189 | -167 |
| 5 | 2017-03-08 22:47:40 UTC | 2017-04-06 00:00:00 UTC | 2017-09-19 14:00:04 UTC | 194 | -166 |
| 6 | 2017-03-07 23:59:51 UTC | 2017-04-07 00:00:00 UTC | 2017-09-19 15:12:50 UTC | 195 | -165 |
| 7 | 2017-03-15 23:23:17 UTC | 2017-04-10 00:00:00 UTC | 2017-09-19 17:14:25 UTC | 187 | -162 |
| 8 | 2017-03-09 13:26:57 UTC | 2017-04-11 00:00:00 UTC | 2017-09-19 14:38:21 UTC | 194 | -161 |
| 9 | 2017-06-12 13:14:11 UTC | 2017-06-26 00:00:00 UTC | 2017-12-04 18:36:29 UTC | 175 | -161 |

As we can notice, some deliveries (-ve days) were delivered way beyond the estimated delivery time (COVID19 Lockdown might be a reason). It is ideal to reduce this difference as much as possible so much so than the actual delivery date is before the estimated one. This is likely to put a positive impression on the customers which will make them likely to use the service more.

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

```sql
with
cte1 as
(select
  c.customer_state as state,
  round(sum(i.freight_value ),2) as total_freight_value,
  round(count(distinct o.order_id),2) as total_orders
from
  `target.orders` o inner join `target.order_items` i on o.order_id = i.order_id
  inner join `target.customers` c on o.customer_id = c.customer_id
group by state),

cte2 as
(select
  state,
  total_freight_value,
  total_orders,
  round(total_freight_value/total_orders,2) as avg_freight_value
from cte1),

--- Top 5
cte5 as
(select distinct * from cte2
order by avg_freight_value desc
limit 5),
--- Bottom 5
cte6 as
(select distinct * from cte2
order by avg_freight_value asc
limit 5)

select * from cte5
UNION ALL
select * from cte6
```

| Row | state | total_freight_value | total_orders | avg_freight_value |
|---|---|---|---|---|
| 1 | RR | 2235.19 | 46.0 | 48.59 |
| 2 | PB | 25719.73 | 532.0 | 48.35 |
| 3 | RO | 11417.38 | 247.0 | 46.22 |
| 4 | AC | 3686.75 | 81.0 | 45.52 |
| 5 | PI | 21218.2 | 493.0 | 43.04 |
| 6 | SP | 718723.07 | 41375.0 | 17.37 |
| 7 | MG | 270853.46 | 11544.0 | 23.46 |
| 8 | PR | 117851.68 | 4998.0 | 23.58 |
| 9 | DF | 50625.5 | 2125.0 | 23.82 |
| 10 | RJ | 305589.31 | 12762.0 | 23.95 |

We can notice the top 5 states with highest average freight value followed by the top 5 states with lowest values. Reducing the average freight value will mean more customers, better shipping partner beneficial for the organisation as well as the customers.

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

```
--- getting delivery time
with
cte1 as
(select
  customer_id,
  date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as
time_to_deliver,
from `target.orders`
where
  (order_delivered_customer_date is not Null) and
  (order_purchase_timestamp is not Null)),

--- delivery time for each state
cte2 as
(select
  t1.time_to_deliver as time_to_deliver,
  t2.customer_state as state
from
  cte1 t1 inner join `target.customers` t2
  on t1.customer_id = t2.customer_id),

---average delivery time for each state
cte3 as
(select
  state,
  avg(time_to_deliver) over (partition by state) as Avg_time_to_deliver
from cte2),

--- Since delivery time is in days (rounding to whole number)
cte4 as
(select
  state,
  cast(round(Avg_time_to_deliver) as int) as Avg_time_to_deliver
from cte3),

--- Top 5
cte5 as
(select distinct * from cte4
order by Avg_time_to_deliver desc
limit 5),
--- Bottom 5
cte6 as
(select distinct * from cte4
order by Avg_time_to_deliver asc
limit 5)

select * from cte5
UNION ALL
```

```
select * from cte6
```

| Row | state | Avg_time_to_deliver |
|---|---|---|
| 1 | RR | 29 |
| 2 | AP | 27 |
| 3 | AM | 26 |
| 4 | AL | 24 |
| 5 | PA | 23 |
| 6 | SP | 8 |
| 7 | PR | 12 |
| 8 | MG | 12 |
| 9 | DF | 13 |
| 10 | SC | 14 |

We can notice the top 5 states with highest average delivery time followed by the top 5 states with lowest average delivery time. As we can see from the above as well as previous query for freight value, the top states with highest and lowest values remain same i.e. **RR and SP**.

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.

```
--- getting delivery days, estimated delivery days (removing null value rows)
with
cte1 as
(select
  customer_id,
  date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as
time_to_deliver,
  date_diff(order_estimated_delivery_date, order_purchase_timestamp, day) as
estimated_time_to_deliver,
from `target.orders`
where
  (order_delivered_customer_date is not Null) and
  (order_estimated_delivery_date is not Null) and
  (order_purchase_timestamp is not Null)),

--- delivery days, estimated delivery days for each state
cte2 as
(select
  t1.time_to_deliver as time_to_deliver,
  t1.estimated_time_to_deliver as estimated_time_to_deliver,
  t2.customer_state as state
from
  cte1 t1 inner join `target.customers` t2
  on t1.customer_id = t2.customer_id),

---average for each state
cte3 as
(select
  state,
  avg(time_to_deliver) over (partition by state) as Avg_time_to_deliver,
  avg(estimated_time_to_deliver) over (partition by state) as
Avg_estimated_time_to_deliver
from cte2),

--- Since delivery time is in days (rounding to whole number)
cte4 as
(select
  state,
  cast(round(Avg_time_to_deliver) as int) as Avg_time_to_deliver,
  cast(round(Avg_estimated_time_to_deliver) as int) as
Avg_estimated_time_to_deliver
from cte3),

--- Distinct values
cte5 as
```

```sql
(select distinct * from cte4 )

--- difference - delivery time was lowest compared to estimated time
select
  state
from cte5
order by (Avg_estimated_time_to_deliver-Avg_time_to_deliver) desc
limit 5
```

| Row | state |
|---|---|
| 1 | AC |
| 2 | RO |
| 3 | AP |
| 4 | AM |
| 5 | RR |

We can notice the top 5 states where delivery is faster as compared to estimated delivery.

## 6. Analysis based on the payments:

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

```sql
---Required columns
with
cte1 as
(select
  t1.order_id as order_id,
  t2.payment_type as payment_type,
  extract(month from t1.order_purchase_timestamp) as order_month
from
  `target.orders` t1 inner join `target.payments` t2
  on t1.order_id = t2.order_id),

--- month by month partition
cte2 as
(select
  order_month, payment_type,
  count(order_id) over (partition by order_month, payment_type) as num_orders
from cte1
)

select distinct * from cte2
order by order_month,num_orders desc
```

| Row | order_month | payment_type | num_orders |
|---|---|---|---|
| 1 | 1 | credit_card | 6103 |
| 2 | 1 | UPI | 1715 |
| 3 | 1 | voucher | 477 |
| 4 | 1 | debit_card | 118 |
| 5 | 2 | credit_card | 6609 |
| 6 | 2 | UPI | 1723 |
| 7 | 2 | voucher | 424 |
| 8 | 2 | debit_card | 82 |
| 9 | 3 | credit_card | 7707 |
| 10 | 3 | UPI | 1942 |

We notice that for all orders throughout all the years for all the months have made payments using credit card.

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

```sql
--- price for each order
with cte1 as
(select
  order_id,
  round(sum(price),2) as order_price
from `target.order_items`
group by order_id),

--- price for that order
cte2 as
(select
  order_id,
  round(sum(payment_value),2) as paid_value
from `target.payments`
group by order_id),

cte3 as
(
  select
  count(t1.order_id) as num_orders
  from cte1 t1 inner join cte2 t2
  on t1.order_id = t2.order_id
  where t1.order_price <= t2.paid_value
)

select * from cte3
```

| Row | num_orders |
|---|---|
| 1 | 98664 |