E- COMMERCE SALES BUSINESS CASE STUDY

INDEX

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

1. Data type of columns in a table.

Query:

```
SELECT
      *
FROM
      `target-business-case-study-1.olist_dataset.INFORMATION_SCHEMA.COLUMNS`
WHERE
        table_name ='customers'
```

Result:

Row	table_name	column_name	ordinal_position	is_nullable	data_type
1	customers	customer_id	1	YES	STRING
2	customers	customer_unique_id	2	YES	STRING
3	customers	customer_zip_code_prefix	3	YES	INT64
4	customers	customer_city	4	YES	STRING
5	customers	customer_state	5	YES	STRING

2. Time period for which the data is given

Query:

```
SELECT
      EXTRACT (DATE FROM MIN (order_purchase_timestamp)) AS first_order_date,
      EXTRACT (DATE FROM MAX (order_purchase_timestamp)) AS last_order_date
FROM
      olist_dataset.orders;
```

Result:

first_order_date	last_order_date
2016-09-04	2018-10-17

3. Cities and States of customers ordered during the given period

Query:

```
SELECT DISTINCT
        c.customer_state,
       c.customer_city,
       o.customer_id
FROM
       olist_dataset.orders AS o
                        JOIN
       olist_dataset.customers AS c ON o.customer_id = c.customer_id
ORDER BY c.customer_state,
                c.customer_city;
```

Result:

Row	customer_state	customer_city	customer_id
1	AC	brasileia	b1161707c5b3711b7cf6213c1...
2	AC	cruzeiro do sul	757bbd8c61a5fd67d5b8c18ef...
3	AC	cruzeiro do sul	f23c4b530f6d7d421de1e38d3...
4	AC	cruzeiro do sul	ee0ab5a9747a11f916fff4b4fc...
5	AC	epitaciolandia	8a45f7d2b87f16a273fd86e9d5...
6	AC	manoel urbano	013bdb994a9c8f09fde3f5f543...
7	AC	porto acre	d8e3846d82e712608dfda713b...
8	AC	rio branco	2201362e68992f654942dc006...
9	AC	rio branco	31dbc13addc753e210692eaca...

Results per page: 50 ▾ 1 – 50 of 99441

2. In-depth Exploration:

1. Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario? Can we see some seasonality with peaks at specific months?

Explanation: As we plot the monthly orders vs month graph from the data, we find out that there is a lot of noise in it (wriggles). So, what we did was plotting 3 months rolling average of order vs time curve to smoothen out the curve a little bit so that we can better understand the trend. When we did exploratory data analysis on a deeper level, we found out that most of the data points of orders in Sept 2018 are missing because number of orders recorded in this month is 1 which is way less than that just the previous month (Aug 2018) which is 6421. So, we can ignore this decline as this is due to some missing data.

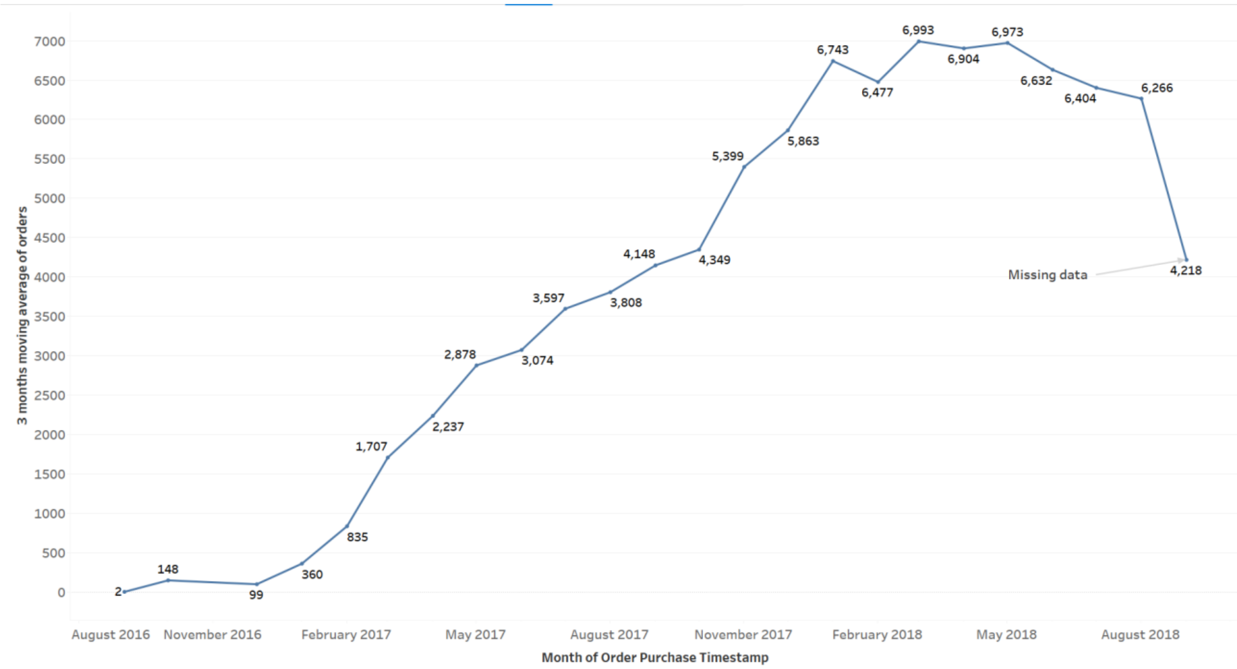
Query: (For overall trend)

```
CREATE OR REPLACE VIEW olist_dataset.monthly_orders AS
        (SELECT
                EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
                EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
                COUNT(DISTINCT order_id) AS orders
            FROM
                olist_dataset.orders
        WHERE
                order_status NOT IN ('canceled', 'unavailable')
        GROUP BY year, month
        ORDER BY year, month);


SELECT
        year,
        month,
        orders,
        CAST(AVG(orders) OVER(ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) AS
INT)
         AS three_month_orders_rolling_avg
FROM
         olist_dataset.monthly_orders
ORDER BY year, month;
```

Result :

Row	year	month	orders	three_month_orders_rolling_avg
1	2016	9	2	2
2	2016	10	293	148
3	2016	12	1	99
4	2017	1	787	360
5	2017	2	1718	835
6	2017	3	2617	1707
7	2017	4	2377	2237
8	2017	5	3640	2878
9	2017	6	3205	3074

Results per page: 50 1 – 24 of 24



As we can clearly see the overall trend from the curve itself, 3 months moving average of orders is growing at a good rate.

Query: (For seasonality with peaks at specific months)

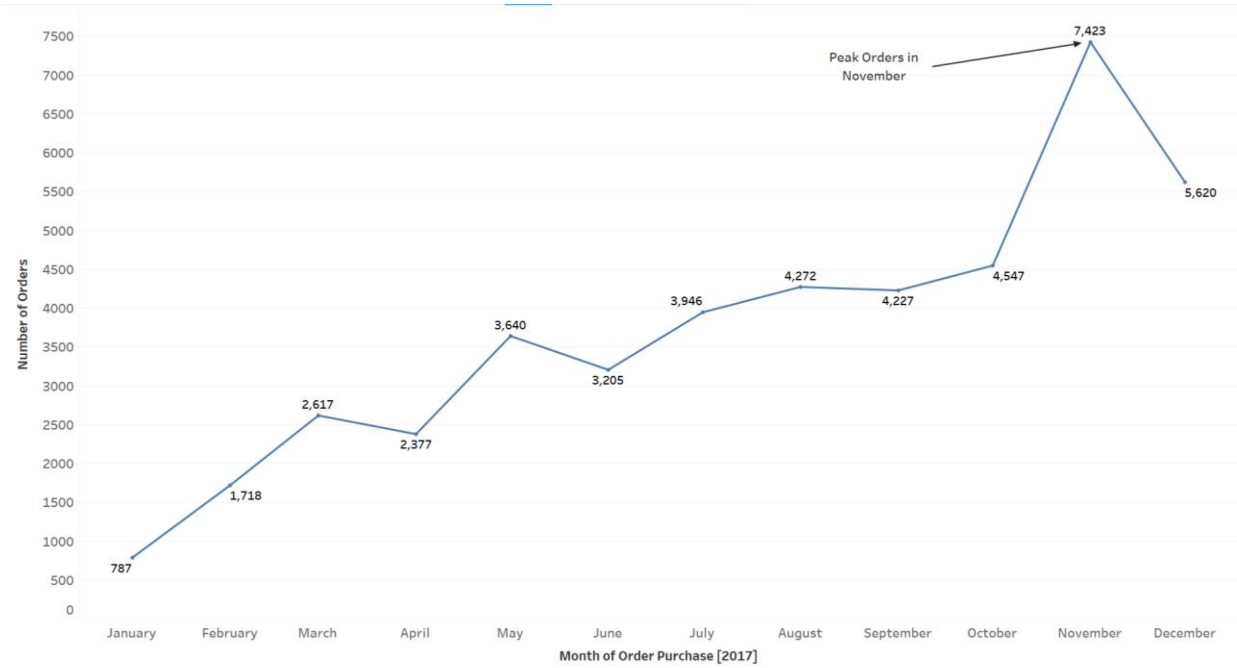
Explanation: Our data ranges from Sep 2016 to Oct 2018. So, to check for seasonality with peaks at specific months, we can only utilize the data for year 2017 as the data for few months is missing for other two years i.e. 2016 & 2018.

```
CREATE OR REPLACE VIEW olist_dataset.monthly_orders AS
        (SELECT
                EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
                EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
                COUNT(DISTINCT order_id) AS orders
         FROM
                olist_dataset.orders
         WHERE
                order_status NOT IN ('canceled', 'unavailable')
         GROUP BY year, month
         ORDER BY year, month);


SELECT
       year,
       month,
       orders
FROM
       olist_dataset.monthly_orders
WHERE
        year = 2017
ORDER BY month;
```

Result :

Row	year	month	orders
1	2017	1	787
2	2017	2	1718
3	2017	3	2617
4	2017	4	2377
5	2017	5	3640
6	2017	6	3205
7	2017	7	3946

Results per page: 50 ▾ 1 – 12 of 12



From the line curve above (for monthly orders in the year 2017), we can clearly see that there is a sharp peak in orders in November. We can guess that it may be due to shopping for Thanksgiving or may be for Christmas.

2. What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)?

Explanation: In this problem statement, we assumed 05:00 am to 06:00 am as dawn, 06:00 am to 12:00m pm as morning, 12:00 pm to 6:00 pm as afternoon and 6:00 pm to 5:00 am next day as night and we recorded the number of orders in these intervals.

Query:

```
CREATE OR REPLACE VIEW olist_dataset.timeslot_orders AS

SELECT
    CASE
            WHEN EXTRACT
(TIME FROM order_purchase_timestamp) BETWEEN '05:00:01' AND '06:00:00'
           THEN 'Dawn (5:00 am – 6:00 am)'
            WHEN EXTRACT
(TIME FROM order_purchase_timestamp) BETWEEN '06:00:01' AND '12:00:00'
           THEN 'Morning (6:00 am – 12:00 pm)'
            WHEN EXTRACT
(TIME FROM order_purchase_timestamp) BETWEEN '12:00:01' AND '18:00:00'
           THEN 'Afternoon (12:00 pm – 6:00 pm)'
           ELSE 'Night (6:00 pm to 00:00 am and 00:00 to 5:00 am)'
     END AS timeslot,
     COUNT (DISTINCT order_id) AS orders
FROM
     olist_dataset.orders
GROUP BY timeslot
ORDER BY orders DESC;
```

Result :

Row	timeslot	orders	percent_of_total_orders
1	Night (6:00 pm to 00:00 am and 00:00 am to 5:00 am)	38648	38.9
2	Afternoon (12:00 pm - 6:00 pm	38365	38.6
3	Morning (6:00 am - 12:00 pm)	22240	22.4
4	Dawn (5:00 am - 6:00 am)	188	0.2

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

1. Get month on month orders by states

Query :

```
WITH state_wise_monthly_orders AS
        (SELECT
                c.customer_state AS state,
                EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
                EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
                COUNT(DISTINCT o.order_id) AS total_orders
        FROM
                olist_dataset.customers AS c
                      LEFT JOIN
                olist_dataset.orders AS o ON c.customer_id = o.customer_id
        GROUP BY state, year, month)


SELECT
        state,
        year,
        month,
        total_orders,
        ROUND(
          (total_orders - LAG(total_orders) OVER(PARTITION BY state ORDER BY year, month))
                * 100  / LAG(total_orders) OVER(PARTITION BY state ORDER BY year, month))
        AS month_on_month_growth_percent
FROM
        state_wise_monthly_orders
ORDER BY state, year, month;
```

Result :

Row	state	year	month	total_orders	month_on_month_growth_percent
1	AC	2017	1	2	null
2	AC	2017	2	3	50.0
3	AC	2017	3	2	-33.0
4	AC	2017	4	5	150.0
5	AC	2017	5	8	60.0
6	AC	2017	6	4	-50.0
7	AC	2017	7	5	25.0
8	AC	2017	8	4	-20.0
9	AC	2017	9	5	25.0

Results per page: 50 1 – 50 of 565

2. Distribution of customers across the states in Brazil

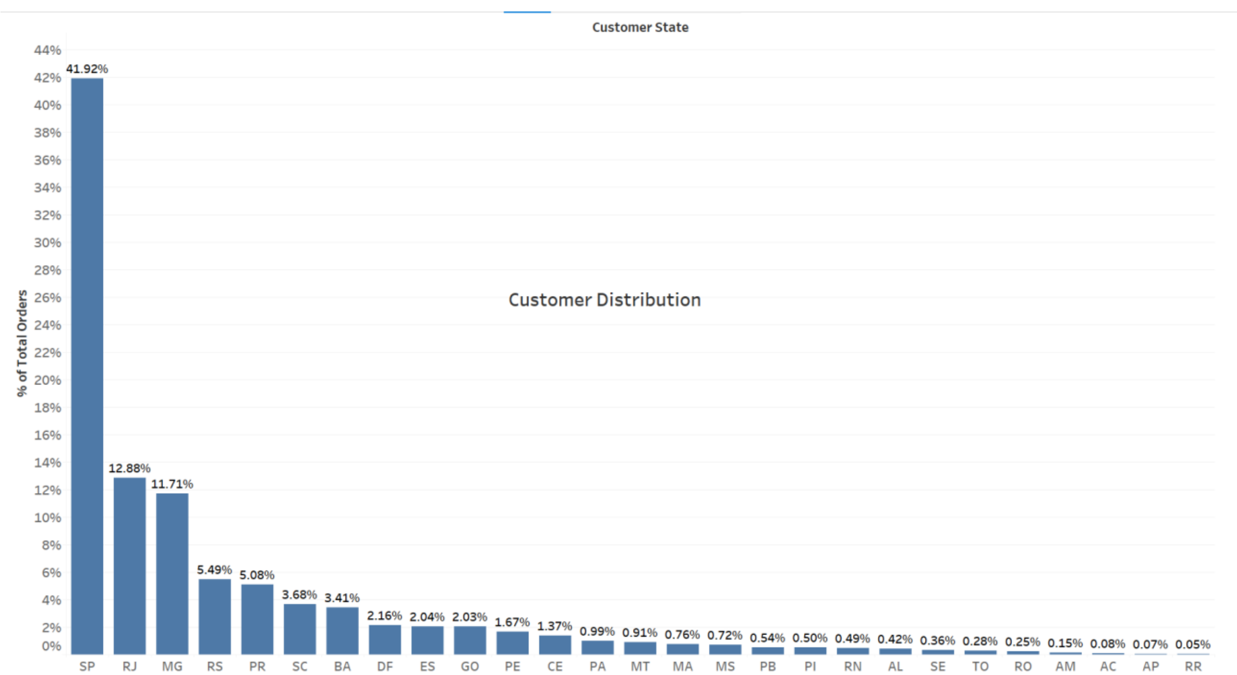
Query :

```
WITH customers_who_ordered AS
          (SELECT DISTINCT
                    customer_id
           FROM
                    olist_dataset.orders),

state_customers_count AS
          (SELECT
                    c2.customer_state AS state,
                    COUNT(DISTINCT c2.customer_unique_id) AS customers
           FROM
                    customers_who_ordered AS c1
                         JOIN
                    olist_dataset.customers AS c2 ON c1.customer_id = c2.customer_id
          GROUP BY state)

SELECT
          state,
          customers,
          ROUND(customers * 100 / (SELECT SUM(customers) FROM state_customers_count)
                    , 2) AS percent_of_total_customers
FROM
          state_customers_count
ORDER BY percent_of_total_customers DESC;
```

Result :

Row	state	customers	percent_of_total_customers
1	SP	40302	41.92
2	RJ	12384	12.88
3	MG	11259	11.71
4	RS	5277	5.49
5	PR	4882	5.08
6	SC	3534	3.68
7	BA	3277	3.41
8	DF	2075	2.16
9	ES	1964	2.04
10	GO	1952	2.03
11	PE	1609	1.67

Results per page: 50 ▾ 1 – 27 of 27



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

1. Get % increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only) - You can use "payment_value" column in payments table.

Query :

```
WITH order_payments AS
        (SELECT
                order_id,
                SUM(payment_value) AS total_payment
         FROM
                 olist_dataset.payments
         GROUP BY order_id),

yearly_revenue AS
        (SELECT
                EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
                    SUM(p.total_payment) AS cost_of_orders
            FROM
                    order_payments AS p
                          JOIN
                    olist_dataset.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 year
         ORDER BY year)

SELECT
        year,
        ROUND(cost_of_orders, 2) AS cost_of_orders,
        ROUND((cost_of_orders - LAG(cost_of_orders) OVER(ORDER BY year)) * 100

          / LAG(cost_of_orders) OVER(ORDER BY year), 2) AS percent_increase_from_last_year
FROM
        yearly_revenue;
```

Result :

Row	year	cost_of_orders	percent_increase_from_last_year
1	2017	3574992.0	null
2	2018	8593138.0	140.4

2. Mean & Sum of price and freight value by customer state.

Query :

```
SELECT
        c.customer_state,
        ROUND(AVG(outpriced), 1) AS mean_price,
        ROUND(AVG(oi.freight_value), 1) AS mean_freight,
        ROUND(SUM(oi.price), 1) AS total_price,
        ROUND(SUM(oi.freight_value), 1) AS total_freight
FROM
        olist_dataset.orders AS o
              JOIN
        olist_dataset.order_items AS oi ON o.order_id = oi.order_id
          JOIN
         olist_dataset.customers AS c ON c.customer_id = o.customer_id
WHERE o.order_status NOT IN ('canceled', 'unavailable')
GROUP BY c.customer_state
ORDER BY total_price DESC, mean_price DESC,
          total_freight DESC, mean_freight DESC;
```

Result :

Row	customer_state	mean_price	mean_freight	total_price	total_freight
1	SP	109.5	15.1	5163867.2	714264.8
2	RJ	124.9	21.0	1811623.4	304044.1
3	MG	120.4	20.6	1573508.2	269566.2
4	RS	119.6	21.7	742559.8	134730.8
5	PR	118.4	20.5	676883.1	117313.6
6	SC	124.6	21.5	518578.3	89445.4
7	BA	134.0	26.4	507108.8	99799.8
8	DF	125.5	21.0	300886.4	50440.8
9	GO	123.9	22.7	287870.5	52673.9
10	ES	121.7	22.0	273532.1	49548.9

Load more

Results per page: 50 1 – 27 of 27

5. Analysis on sales, freight and delivery time

1. **Calculate days between purchasing, delivering and estimated delivery**
2. **Create columns:**
    - **time_to_delivery = order_purchase_timestamp-order_delivered_customer_date**
    - **diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date**
3. **Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery**

Query :

```
CREATE OR REPLACE VIEW olist_dataset.order_level_data AS
        (SELECT
              o.order_id,
              c.customer_state,
              MAX(TIMESTAMP_DIFF(o.order_delivered_customer_date, o.order_purchase_timest
amp, DAY))                AS time_to_delivery,
            MAX(TIMESTAMP_DIFF(order_estimated_delivery_date, order_delivered_customer_dat
            e, DAY)) AS diff_estimated_delivery,
            SUM(oi.freight_value) AS order_freight_value
FROM
            olist_dataset.orders AS o
                      JOIN
            olist_dataset.order_items AS oi ON o.order_id = oi.order_id
                      JOIN
             olist_dataset.customers AS c ON o.customer_id = c.customer_id
GROUP BY o.order_id, c.customer_state);

CREATE OR REPLACE VIEW olist_dataset.state_level_data AS
        (SELECT
              customer_state,
              AVG(order_freight_value) AS avg_freight_value,
              AVG(time_to_delivery) AS avg_time_to_delivery,
              AVG(diff_estimated_delivery) AS avg_diff_estimated_delivery
         FROM
              olist_dataset.order_level_data
         GROUP BY
              customer_state);
```

4. Sort the data to get the following:

1. Top 5 states with highest/lowest freight value – sort in dec/asc limit 5

Query : (For top 5 states with highest average freight value)

(Base table **olist_dataset.order_level_data** on Page 15)

```
SELECT
    customer_state AS top_5_states_by_highest_avg_freight,
    ROUND(avg_freight_value, 1) AS avg_freight_value
FROM
    olist_dataset.state_level_data
ORDER BY avg_freight_value DESC
LIMIT 5;
```

Result :

Row	top_5_states_by_highest_avg_freight	avg_freight_value
1	RR	49.1
2	PB	48.3
3	RO	46.3
4	AC	45.5
5	PI	43.1

Query : (For top 5 states with lowest average freight value)

(Base table **olist_dataset.order_level_data** on Page 15)

```
SELECT
    customer_state AS top_5_states_by_lowest_avg_freight,
    ROUND(avg_freight_value, 1) AS avg_freight_value
FROM
    olist_dataset.state_level_data
ORDER BY avg_freight_value
LIMIT 5;
```

Result :

Row	top_5_states_by_lowest_avg_freight	avg_freight_value
1	SP	17.4
2	MG	23.4
3	PR	23.5
4	DF	23.8
5	RJ	23.9

2. Top 5 states with highest/lowest average time to delivery.

Query : (For Top 5 states with highest average time to delivery)

(Base table **olist_dataset.order_level_data** on Page 15)

```
SELECT
    customer_state AS top_5_states_by_highest_avg_time_to_delivery,
    CAST(avg_time_to_delivery AS INT) AS avg_time_to_delivery
FROM
    olist_dataset.state_level_data
ORDER BY avg_time_to_delivery DESC
LIMIT 5;
```

Result :

Row	top_5_states_by_highest_avg_time_to_delivery	avg_time_to_delivery
1	RR	29
2	AP	27
3	AM	26
4	AL	24
5	PA	23

Query : (For Top 5 states with lowest average time to delivery)

(Base table **olist_dataset.order_level_data** on Page 15)

```
SELECT
    customer_state AS top_5_states_by_lowest_avg_time_to_delivery,
    CAST(avg_time_to_delivery AS INT) AS avg_time_to_delivery
FROM
    olist_dataset.state_level_data
ORDER BY avg_time_to_delivery
LIMIT 5;
```

Result :

Row	top_5_states_by_lowest_avg_time_to_delivery	avg_time_to_delivery
1	SP	8
2	PR	12
3	MG	12
4	DF	13
5	SC	14

3. Top 5 states where delivery is really fast/ not so fast compared to estimated date.

Query : (For top 5 states where is really fast compared to estimated date)

(Base table **olist_dataset.order_level_data** on Page 15)

```
SELECT
    customer_state AS top_5_states_by_fastest_delivery,
    CAST(avg_diff_estimated_delivery AS INT) AS avg_diff_estimated_delivery
FROM
    olist_dataset.state_level_data
ORDER BY avg_diff_estimated_delivery DESC
LIMIT 5;
```

Result :

Row	top_5_states_by_fastest_delivery	avg_diff_estimated_delivery
1	AC	20
2	RO	19
3	AM	19
4	AP	19
5	RR	16

Query : (For top 5 states where is not so fast compared to estimated date)

(Base table **olist_dataset.order_level_data** on Page 15)

```
SELECT
    customer_state AS top_5_states_by_slowest_delivery,
    CAST(avg_diff_estimated_delivery AS INT) AS avg_diff_estimated_delivery
FROM
    olist_dataset.state_level_data
ORDER BY avg_diff_estimated_delivery
LIMIT 5;
```

Result :

Row	top_5_states_by_slowest_delivery	avg_diff_estimated_delivery
1	AL	8
2	SE	9
3	MA	9
4	SP	10
5	BA	10

6. Payment type analysis:

1. Month over Month count of orders for different payment types.

Query :

```
SELECT
        p.payment_type,
        EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
        EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
        COUNT(DISTINCT o.order_id) AS orders
FROM
        olist_dataset.orders AS o
                JOIN
        olist_dataset.payments AS p ON o.order_id = p.order_id
WHERE o.order_status NOT IN ('canceled', 'unavailable')
GROUP BY year, month, p.payment_type
ORDER BY p.payment_type, year, month
```

Result :

Row	payment_type	year	month	orders
1	UPI	2016	10	60
2	UPI	2017	1	193
3	UPI	2017	2	383
4	UPI	2017	3	582
5	UPI	2017	4	488
6	UPI	2017	5	758
7	UPI	2017	6	702
8	UPI	2017	7	829
9	UPI	2017	8	920
10	UPI	2017	9	885
11	UPI	2017	10	971

Results per page: 50 1 – 50 of 87

2. Distribution of payment installments and count of orders.

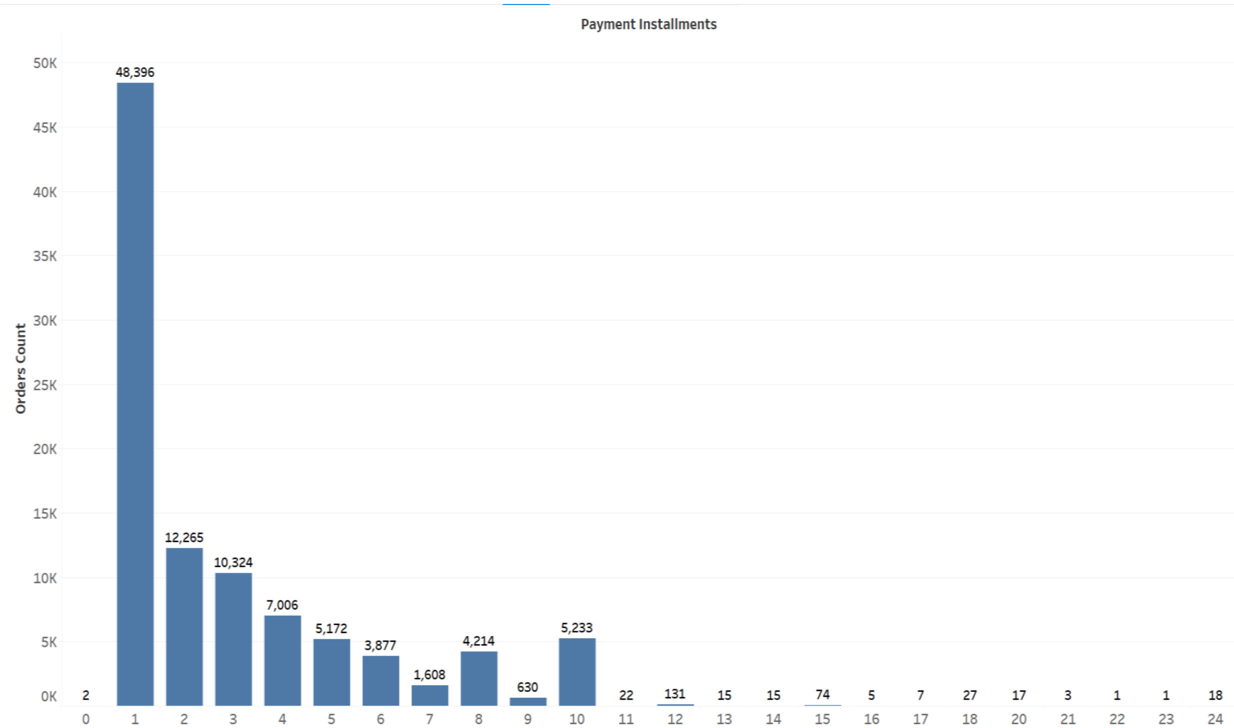
Query :

```
SELECT
        p.payment_installments,
        COUNT(DISTINCT o.order_id) AS orders_count
FROM
        olist_dataset.payments AS p
                JOIN
        olist_dataset.orders AS o ON o.order_id = p.order_id
WHERE
        o.order_status NOT IN ('canceled', 'unavailable')
GROUP BY payment_installments
ORDER BY payment_installments;
```

Result :

Row	payment_installments	orders_count
1	0	2
2	1	48396
3	2	12265
4	3	10324
5	4	7006
6	5	5172
7	6	3877
8	7	1608
9	8	4214
10	9	630
11	10	5233

Results per page: 50 ▾ 1 – 24 of 24


Payment Installments
Orders Count
50K
45K
40K
35K
30K
25K
20K
15K
10K
5K
0K
2
48,396
12,265
10,324
7,006
5,172
3,877
1,608
4,214
630
5,233
22
131
15
15
74
5
7
27
17
3
1
1
18
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
20
21
22
23
24

7. Frequently Bought Together

1. Which are those products which customers tend to buy together most frequently?

Query :

```
WITH freq_bought_together_products AS
 (SELECT
       o1.product_id AS product_1,
       o2.product_id AS product_2,
       COUNT(DISTINCT o1.order_id) AS bought_together_freq
  FROM
       olist_dataset.order_items AS o1
          JOIN
       olist_dataset.order_items AS o2 ON o1.order_id = o2.order_id
                                            AND o1.product_id < o2.product_id
  GROUP BY product_1,
                  product_2)
```

Result :

```
SELECT
  *
FROM
  freq_bought_together_products
ORDER BY bought_together_freq DESC;
```

Row	product_1	product_2	bought_together_freq
1	36f60d45225e60c7da4558b07...	e53e557d5a159f5aa2c5e995d...	34
2	35afc973633aaeb6b877ff57b2...	99a4788cb24856965c36a24e3...	29
3	4fcb3d9a5f4871e8362dfedbdb...	f4f67ccaece962d013a4e1d7dc...	17
4	36f60d45225e60c7da4558b07...	3f14d740544f37ece8a9e7bc8...	12
5	389d119b48cf3043d311335e4...	422879e10f46682990de24d77...	11
6	389d119b48cf3043d311335e4...	53759a2ecddad2bb87a079a1f...	9
7	368c6c730842d78016ad8238...	53759a2ecddad2bb87a079a1f...	8
8	422879e10f46682990de24d77...	53759a2ecddad2bb87a079a1f...	7
9	18486698933fbb64af6c0a255f...	dbb67791e405873b259e4656...	7

Results per page: 50 ▾ 1 – 50 of 4058

2. Which are those product categories which customers tend to buy together most frequently?

Query :

```
WITH freq_bought_together_products AS
    (SELECT
                o1.product_id AS product_1,
                o2.product_id AS product_2,
          COUNT(DISTINCT o1.order_id) AS bought_together_freq
     FROM
          olist_dataset.order_items AS o1
            JOIN
          olist_dataset.order_items AS o2 ON o1.order_id = o2.order_id
                                     AND o1.product_id < o2.product_id
     GROUP BY product_1,
              product_2),

categories_with_redundancy AS
    (SELECT
         p1.product_category AS category_1,
         p2.product_category AS category_2,
         SUM(bought_together_freq) AS bought_together_freq
    FROM
         freq_bought_together_products AS f
                JOIN
         olist_dataset.products AS p1 ON p1.product_id = f.product_1
                JOIN
         olist_dataset.products AS p2 ON p2.product_id = f.product_2
    GROUP BY category_1,
          category_2
    HAVING category_1 != category_2)

SELECT
     c1.category_1,
     c1.category_2,
     c1.bought_together_freq + COALESCE(c2.bought_together_freq, 0) AS bought_together_freq
FROM
     categories_with_redundancy AS c1
          LEFT JOIN
    categories_with_redundancy AS c2 ON c1.category_1 = c2.category_2
                                  AND c2.category_1 = c1.category_2
WHERE
     c1.category_1 < c1.category_2
ORDER BY bought_together_freq DESC;
```

Result:

Row	category_1	category_2	bought_together_freq
1	Furniture Decoration	bed table bath	91
2	House comfort	bed table bath	48
3	Furniture Decoration	housewares	25
4	bed table bath	housewares	23
5	Cool Stuff	babies	22
6	babies	toys	21
7	babies	bed table bath	21
8	Furniture Decoration	Garden tools	20
9	HEALTH BEAUTY	sport leisure	14

Results per page: 50 1 – 50 of 165

Actionable Insights & Recommendations

1. Peak month seasonality :

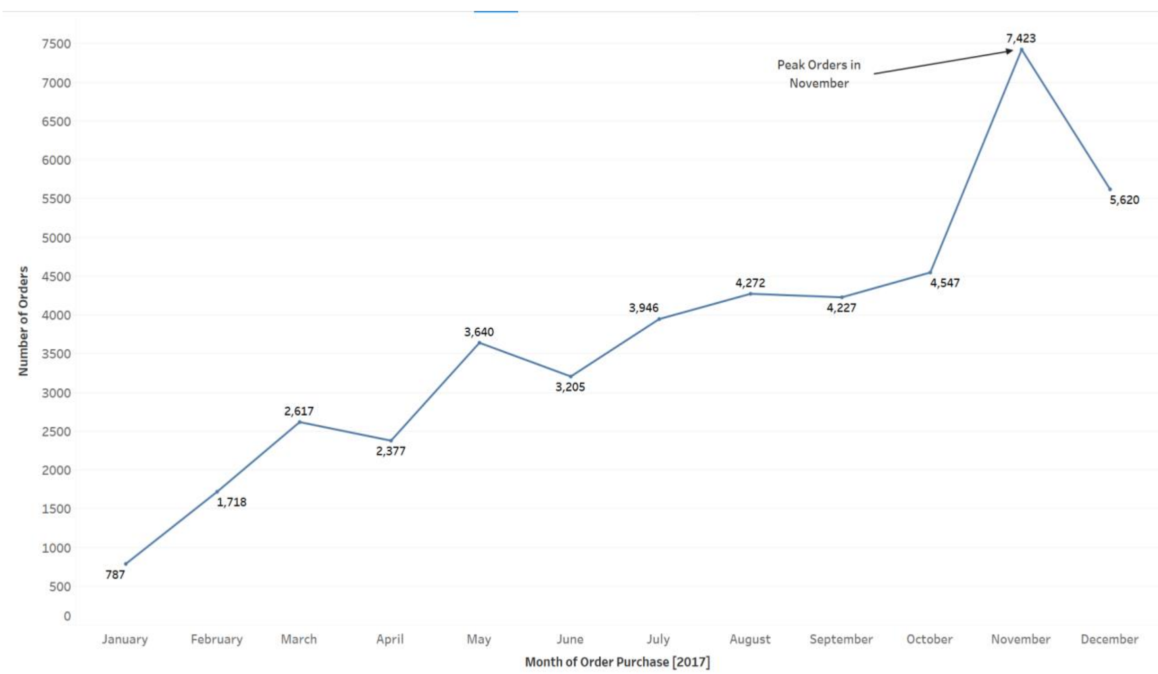
Insights:

We can clearly see from the trend line (for year 2017) that there is a peak seasonality in the month of November.

Recommendations:

Some key points to be ensured before the peak season:

1. Re-stock the inventory before the peak month.
2. Running advertisements and giving out some offers to enhance customer engagement.
3. Introducing some new product line (if there is any, in the pipeline) so that maximum customer response can be recorded for that without much marketing expenditure.



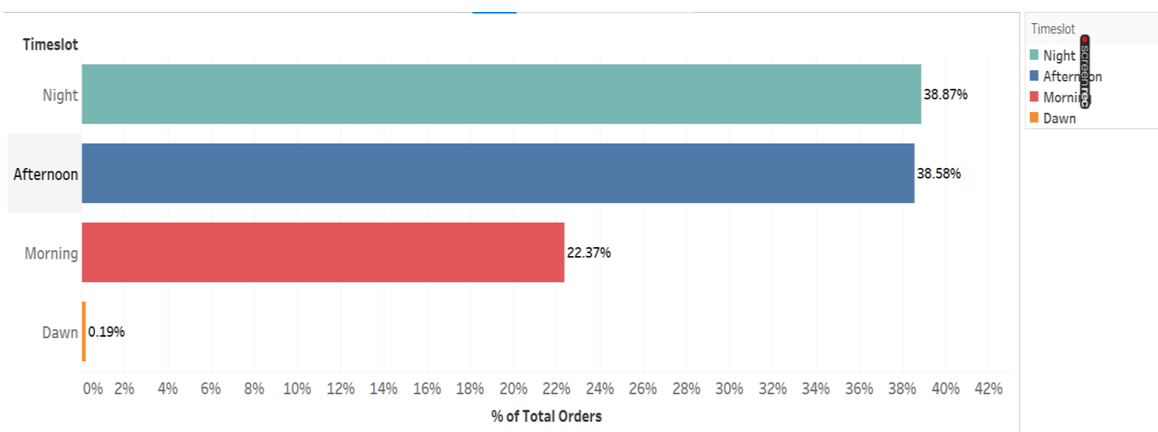
2. Peak hours:

Insights:

We can clearly see that more than 75 % of our orders are booked at night or afternoon. And most importantly, the afternoon slot despite being just 6 hours long as compared to night slot being 11 hours, it is showing an amazing customer engagement contributing a whopping 38.9 % of the total orders.

Recommendations:

1. Our recommendation would be to running ad campaign or suggest some newly launched products during the afternoon time as it can result in really good results in regard to revenue figures.
2. During dawn, only 0.2 % of the total orders are booked. We can utilize this slot in maintenance of the website so that at the time of peak hours, our website doesn't crash.



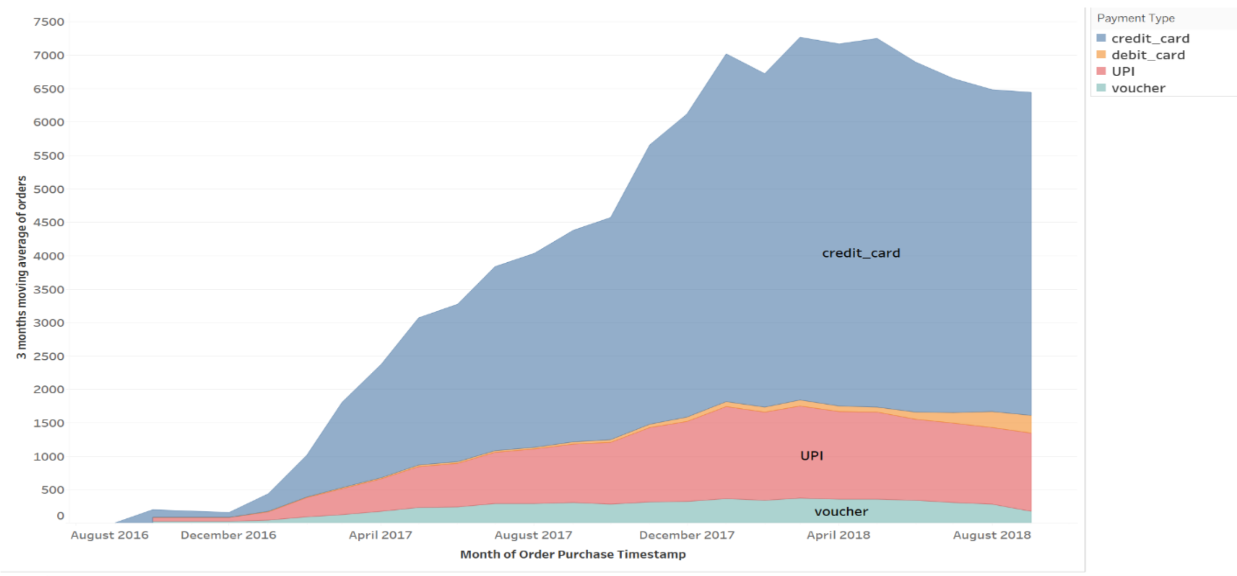
3. Payment modes :

Insights:

We can clearly see the area chart that **credit card** is the most popular payment mode on our platform and it is still growing at a much higher rate than the others.

Recommendation :

Our recommendation would be to onboard some credit card companies or banks which can give some offers to our customers so that the user engagement can be enhanced even more. We can even roll out our own credit card.



4. Logistics:

1. Setting up of new logistic hubs.

Insights:

As we can see from the data itself, the delivery in the following states is really slow which can really ruin the customer experience.

Recommendation:

We can try to open up more logistic hubs or identify few more suppliers (if needed) in these regions so that average time to delivery can be reduced and customer experience can be further enhanced.

Row	top_5_states_by_highest_avg_time_to_delivery	avg_time_to_delivery
1	RR	29
2	AP	27
3	AM	26
4	AL	24
5	PA	23

2. Re-tuning of ML model predicting estimated delivery date.

Insights:

We can clearly notice by looking at the data that, there is a significant difference between the estimated delivery date and actual delivery to customer date.

Recommendation:

Our recommendation is to re-tune the Machine Learning Model predicting the estimated delivery date so that it can come close to the real world situation. This will add up to the credibility of our brand.

Row	top_5_states_by_fastest_delivery	avg_diff_estimated_delivery
1	AC	20
2	RO	19
3	AM	19
4	AP	19
5	RR	16

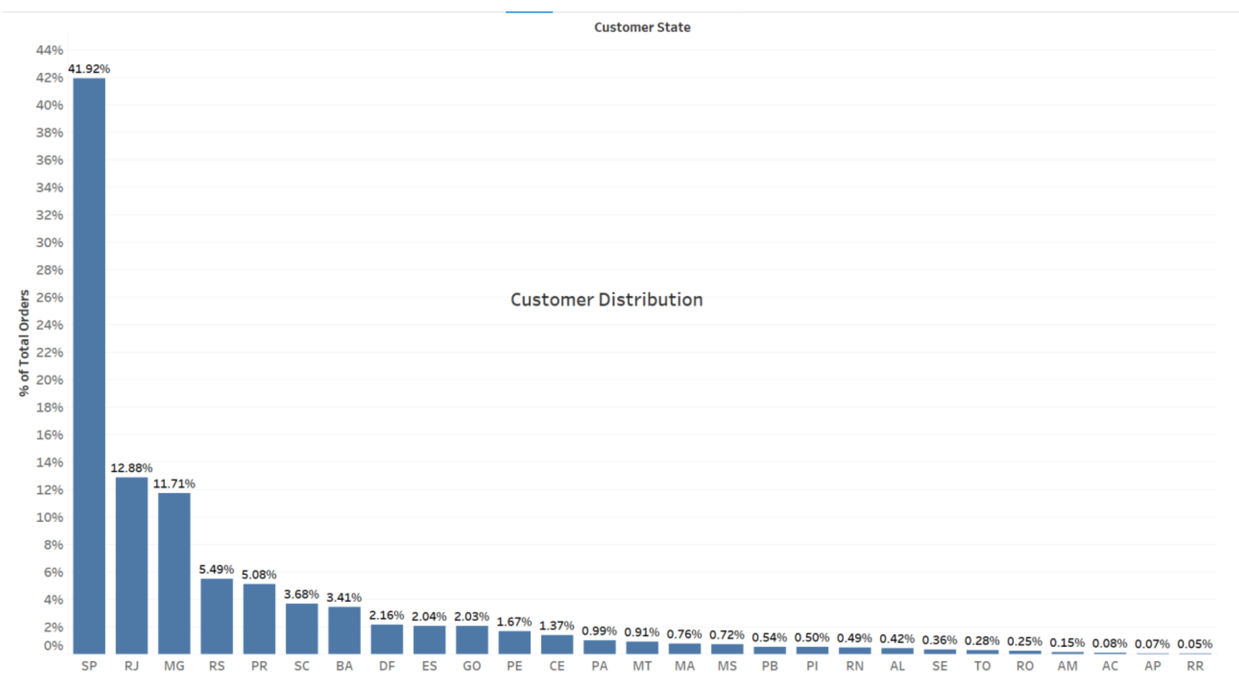
5. Launching new product line:

Insights:

More than **40%** of the total customers (who have ever ordered from us) come from state **SP** alone.

Recommendations:

As we know that Sao Paulo is the top performing state, we can leverage this to introduce some new products (which are in the pipeline) in **SP** and see the customer feedback. If there is a positive response, we can roll out the same in other states too.



6. Total Revenue vs Average Order Value:

Insights:

We can clearly see from the dual axis chart below that, **SP** despite being the top performing state in terms of revenue, it is still lagging behind in terms of average order value (which is only **$125.75**).

Recommendations:

The low average order value in state SP should be a slight concern to us despite it being the top performing state in terms of revenue. If we are able to increase the average order value by just a bit, we can hugely impact the total revenue from the state. Our recommendation would be to make a recommender system which can recommend correlated products or product categories (frequently bought together items) while the customer is buying. For instance, if the customer is buying a mobile phone from us, we should recommend him mobile accessories. This will impact the average order value and in turn, boost the total revenue too.