Business Case: Target SQL

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

```
SELECT
 * EXCEPT(is_generated,
          generation_expression,
          is_stored,
          is_updatable,
          is_hidden,
          is_system_defined,
          is_partitioning_column,
          clustering_ordinal_position,
          collation_name,
          column_default,
          ordinal_position,
          table_catalog
         )
 FROM
     `target-sql-362404.Target.INFORMATION_SCHEMA.COLUMNS`
ORDER BY table_name;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	table_schema	table_name	column_name	is_nullable	data_type
1	Target	customers	customer_id	YES	STRING
2	Target	customers	customer_unique_id	YES	STRING
3	Target	customers	customer_zip_code_prefix	YES	INT64
4	Target	customers	customer_city	YES	STRING
5	Target	customers	customer_state	YES	STRING
6	Target	geolocation	geolocation_zip_code_prefix	YES	INT64
7	Target	geolocation	geolocation_lat	YES	FLOAT64
8	Target	geolocation	geolocation_lng	YES	FLOAT64
9	Target	geolocation	geolocation_city	YES	STRING
10	Target	geolocation	geolocation_state	YES	STRING
11	Target	order_items	order_id	YES	STRING
12	Target	order_items	order_item_id	YES	INT64
13	Target	order_items	product_id	YES	STRING
14	Target	order_items	seller_id	YES	STRING
15	Target	order_items	shipping_limit_date	YES	TIMESTAMP
16	Target	order_items	price	YES	FLOAT64
17	Target	order_items	freight_value	YES	FLOAT64
18	Target	order_reviews	review_id	YES	STRING

Count of data types in each table

```
SELECT a.table_name,
       a.data_type,
       COUNT(a.data_type) AS count_of_datatypes
 FROM  (
         SELECT
               * EXCEPT(is_generated,
                        generation_expression,
                        is_stored,
                        is_updatable,
```

```
                    is_hidden,
                    is_system_defined,
                    is_partitioning_column,
                    clustering_ordinal_position,
                    collation_name,
                    column_default,
                    ordinal_position,
                    table_catalog
                  )
        FROM
              `target-sql-362404.Target.INFORMATION_SCHEMA.COLUMNS`
    ) a
GROUP BY
       a.table_name, a.data_type
ORDER BY
       a.table_name
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	table_name	data_type	count_of_datatypes
1	customers	STRING	4
2	customers	INT64	1
3	geolocation	INT64	1
4	geolocation	FLOAT64	2
5	geolocation	STRING	2
6	order_items	STRING	3
7	order_items	INT64	1
8	order_items	TIMESTAMP	1
9	order_items	FLOAT64	2
10	order_reviews	STRING	3
11	order_reviews	INT64	1
12	order_reviews	TIMESTAMP	2
13	orders	STRING	3
14	orders	TIMESTAMP	5
15	payments	STRING	2
16	payments	INT64	2
17	payments	FLOAT64	1

2. Time period for which the data is given

```
SELECT
      MIN(DATE(order_purchase_timestamp)) AS Order_from_date,
      MAX(DATE(order_delivered_customer_date)) AS Order_till_date,
      DATE_DIFF(MAX(DATE(order_delivered_customer_date)),MIN(DATE(order_purchase_ti
mestamp)),month) AS Time_period_in_months
  FROM
      target-sql-362404.Target.orders ;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	Order_from_date	Order_till_date	Time_period_in_months
1	2016-09-04	2018-10-17	25

Insights:

1) The order data surveyed in the dataset started from **September 2016** till **October 2018**, ie for a time period of 25 months

3. Cities and States covered in the dataset

```
SELECT COUNT(DISTINCT customer_id) as Customers,
       COUNT(DISTINCT customer_unique_id) AS Unique_Customer_ID,
       COUNT(DISTINCT customer_zip_code_prefix) as Zipcodes,
       COUNT(DISTINCT customer_city) as Cities,
       COUNT(DISTINCT customer_state) as States
  FROM
       target-sql-362404.Target.customers;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	Customers	Unique_Customer_ID	Zipcodes	Cities	States
1	99441	96096	14994	4119	27

Insights:

1) The dataset consists of **99,441 unique customer identifier** who resides in **14,994 Zip location** of Brazil.
2) **There are 4,119 total unique cites within 27 states from which orders have been placed**

 # Customers per state Analysis:

```
SELECT
      customer_state,
      COUNT(customer_id) as number_of_customers
FROM
    `target-sql-362404.Target.customers`
GROUP BY
      customer_state
ORDER BY
      number_of_customers DESC;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	number_of_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
10	GO	2020

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?**

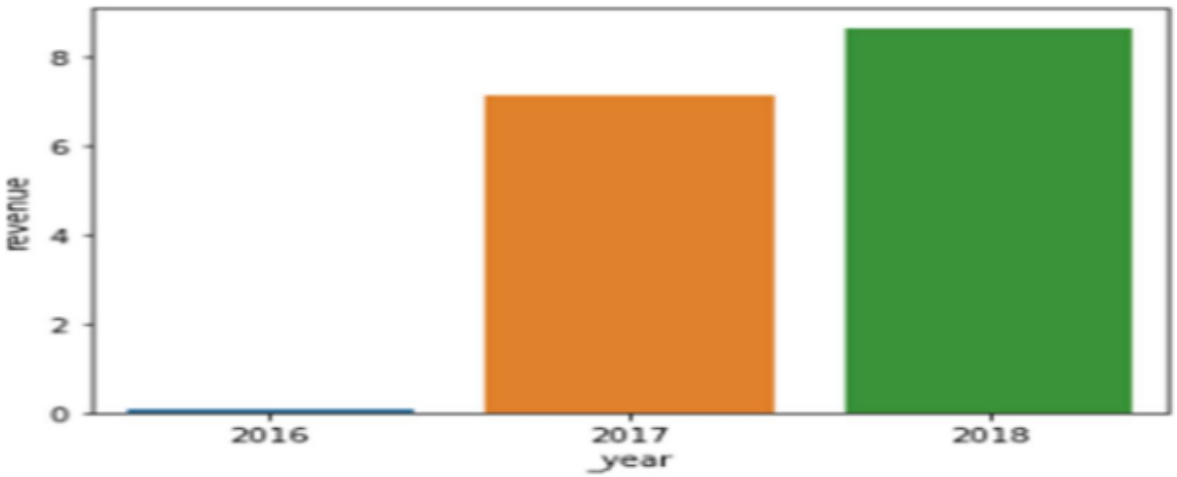
a) Revenue per year

```
SELECT
      EXTRACT(YEAR FROM order_purchase_timestamp) as year_of_purchase,
      ROUND(SUM(price*freight_value),2) AS revenue,
      COUNT(o.order_id) as number_of_orders
FROM
    `target-sql-362404.Target.orders` as o
JOIN
    `target-sql-362404.Target.order_items` as oi
ON
    o.order_id = oi.order_id
GROUP BY
    year_of_purchase
ORDER BY
    year_of_purchase ;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	year_of_pur...	revenue	number_of_orders
1	2016	1461314.76	370
2	2017	175737923.59	50864
3	2018	229936158.12	61416



Insights:

1) In comparison to **2017, sales revenue** has **increased** in **2018 by 21%**

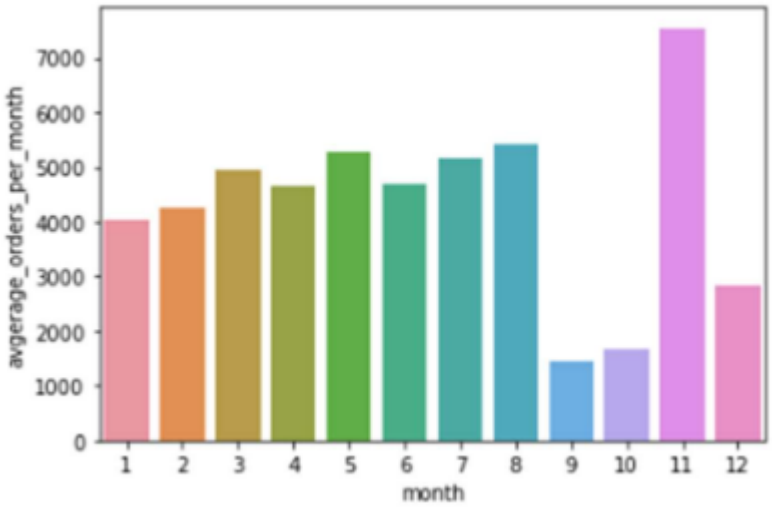
b) Average order per Month

```
SELECT
      x.month,
      AVG(x.num_of_orders) AS avgerage_orders_per_month
FROM
(
  SELECT
    EXTRACT( YEAR FROM order_purchase_timestamp) as year,
    EXTRACT( MONTH FROM order_purchase_timestamp) as month,
    COUNT(order_id) as num_of_orders
  FROM
      `target-sql-362404.Target.orders`
  GROUP BY
      year,
      month
  ORDER BY
      year,
      month
) as x
GROUP BY x.month
ORDER BY x.month ;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	month	avgerage_orders_per_month
1	1	4034.5
2	2	4254.0
3	3	4946.5
4	4	4671.5
5	5	5286.5
6	6	4706.0
7	7	5159.0
8	8	5421.5
9	9	1435.0
10	10	1653.0
11	11	7544.0
12	12	2837.0



Insights:

1) In the month of **November** the average number of order placed are **highest**
2) **September and October** month average orders are **comparatively low**.
3) In **May, July and August** order placed are **higher compared to other months**

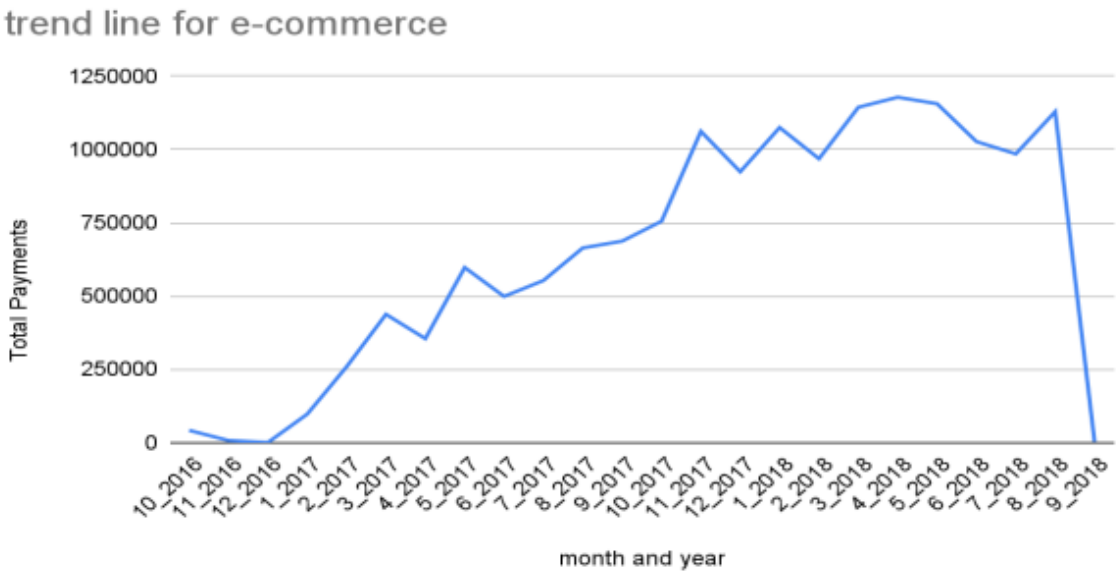
```
SELECT DISTINCT
              EXTRACT(MONTH FROM o.order_delivered_carrier_date) as purchase_month,
              EXTRACT(YEAR FROM o.order_delivered_carrier_date) as purchase_year,
              SUM(payment_value) OVER(PARTITION BY
                                        EXTRACT(YEAR FROM o.order_delivered_carrier_date),
                                        EXTRACT(MONTH FROM o.order_delivered_carrier_date)
                                      ORDER BY
                                        EXTRACT(YEAR FROM o.order_delivered_carrier_date),
```

```
                                        EXTRACT(MONTH FROM o.order_delivered_carrier_date)
                                    RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING
                                        ) AS total_payment
FROM
     `target-sql-362404.Target.orders` o
LEFT JOIN
     `target-sql-362404.Target.payments` p
ON
     o.order_id = p.order_id;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	purchase_month	purchase_year	total_payment
1	12	2016	102.30999999999999
2	4	2018	1178655.68
3	7	2017	553109.99999999988
4	5	2018	1156487.18
5	2	2017	258761.05
6	12	2017	924316.49
7	6	2017	498790.72
8	1	2018	1075304.39
9	6	2018	1027328.95
10	5	2017	597431.92999999993



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

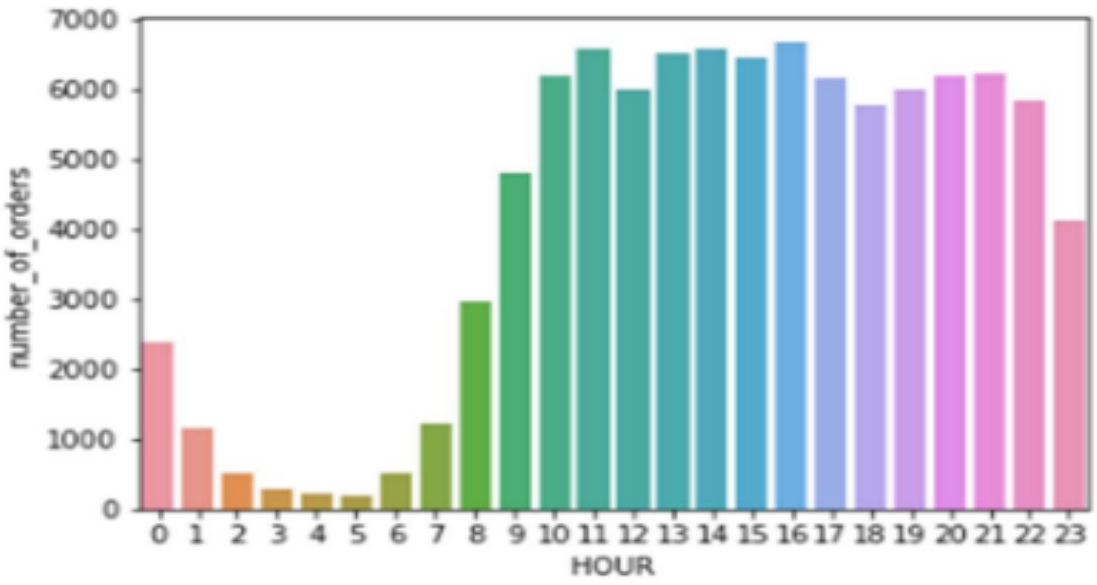
a) Number of Orders per Hour

```sql
SELECT
      X.HOUR,
      COUNT(X.order_id) AS number_of_orders
FROM
 (
 SELECT
       order_id,
       EXTRACT (HOUR FROM order_purchase_timestamp) AS HOUR
 FROM
     `target-sql-362404.Target.orders`
 ) AS X
GROUP BY
      X.HOUR;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	HOUR	number_of_orders
1	11	6578
2	1	1170
3	17	6150
4	13	6518
5	12	5995
6	18	5769
7	10	6177
8	21	6217
9	15	6454
10	22	5816



b) Number of orders per hour during daytime

```
SELECT
      X.HOUR,
      COUNT(X.order_id) AS number_of_orders,
      CASE
          WHEN X.HOUR BETWEEN 5 AND 7 THEN 'Dawn'
          WHEN X.HOUR BETWEEN 8 AND 11 THEN 'Morning'
          WHEN X.hour BETWEEN 12 AND 17 THEN 'Afternoon'
          WHEN X.hour BETWEEN 18 and 21 THEN 'Evening'
          ELSE 'night'
      END AS time
FROM
 (
 SELECT
       order_id,
       EXTRACT (HOUR FROM order_purchase_timestamp) AS HOUR
 FROM
     `target-sql-362404.Target.orders`
 ) AS X
GROUP BY
      X.HOUR;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	HOUR	number_of_orders	time
1	11	6578	Morning
2	1	1170	night
3	17	6150	Afternoon
4	13	6518	Afternoon
5	12	5995	Afternoon
6	18	5769	Evening
7	10	6177	Morning
8	21	6217	Evening
9	15	6454	Afternoon
10	22	5816	night

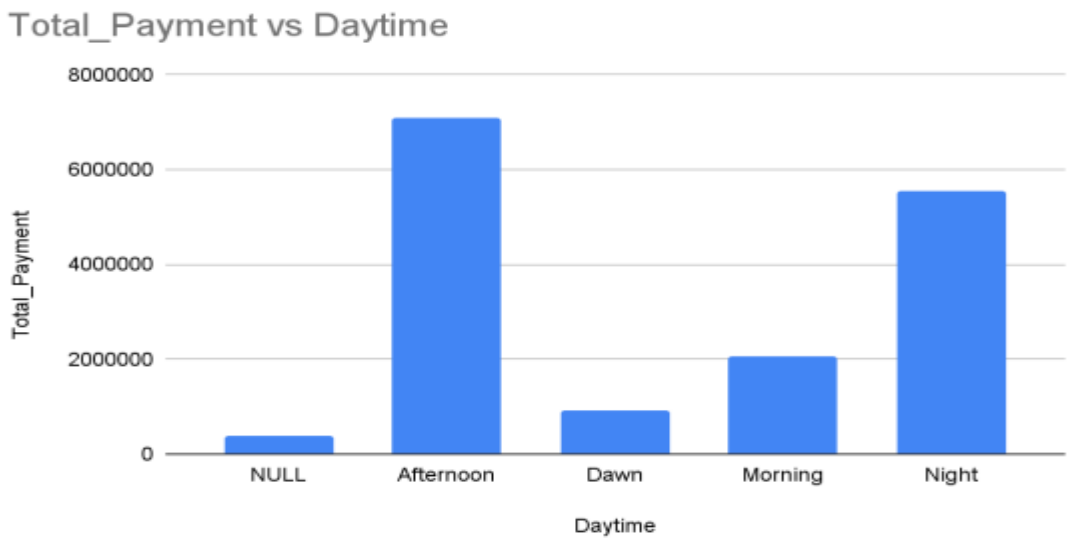
```
      SELECT DISTINCT
             daytime_info.daytime AS Daytime,

                    SUM(payment_value) OVER (PARTITION BY daytime
          ORDER BY daytime
    ) As Total_Payment

    FROM
    (SELECT
        p.payment_value,
        CASE
                WHEN EXTRACT (HOUR FROM o.order_delivered_carrier_date) BETWEEN 0 AND 6
             THEN "Dawn"
```

```
        WHEN EXTRACT (HOUR FROM o.order_delivered_carrier_date) BETWEEN 7 AND 11
      THEN "Morning"
        WHEN EXTRACT (HOUR FROM o.order_delivered_carrier_date) BETWEEN 12 AND 17
      THEN "Afternoon"
        WHEN EXTRACT (HOUR FROM o.order_delivered_carrier_date) BETWEEN 18 AND 23
      THEN "Night"
    END AS Daytime
FROM
    `target-sql-362404.Target.orders` o
LEFT JOIN
    `target-sql-362404.Target.payments` p
ON
    o.order_id = p.order_id) As daytime_info;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	Daytime	Total_Payment
1	*null*	400716.70999999996
2	Afternoon	7082950.09
3	Dawn	908729.1
4	Morning	2066651.8299999998
5	Night	5549824.39



Insights:

1) Customers are generally purchasing during **Moring at 8:00 AM till late Evening upto 11:00 PM**
2) **Afternoon orders** have a **huge spike** and also **Evening** , compared to morning , and night hour

---

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

1. Get month on month orders by region, states

```
SELECT DISTINCT
```

```
            EXTRACT (YEAR FROM o.order_delivered_carrier_date) AS purchase_year,
           EXTRACT (MONTH FROM o.order_delivered_carrier_date) AS purchase_month,
            COUNT(o.order_id) OVER( PARTITION BY
                                EXTRACT (YEAR FROM o.order_delivered_carrier_date),
                                EXTRACT (MONTH FROM o.order_delivered_carrier_date)
                                    ORDER BY
                               EXTRACT (YEAR FROM o.order_delivered_carrier_date) ,
                               EXTRACT (MONTH FROM o.order_delivered_carrier_date)
                                   ) As No_of_orders
FROM
    `target-sql-362404.Target.orders` o
LEFT jOIN
    `target-sql-362404.Target.customers` c
ON
  o.customer_id= c.customer_id;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	purchase_year	purchase_month	No_of_orders
1	*null*	*null*	1783
2	2016	10	247
3	2016	11	32
4	2016	12	2
5	2017	1	612
6	2017	2	1517
7	2017	3	2717
8	2017	4	2141
9	2017	5	3709
10	2017	6	3263

2. How are customers distributed in Brazil

```
SELECT DISTINCT customer_state,
       COUNT(customer_id) OVER(PARTITION BY customer_state ORDER BY customer_state
                           RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING
                               ) as customer_count
FROM
    `target-sql-362404.Target.customers`
ORDER BY customer_state;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	customer_count
1	AC	81
2	AL	413
3	AM	148
4	AP	68
5	BA	3380
6	CE	1336
7	DF	2140
8	ES	2033
9	GO	2020
10	MA	747

Insights:

1) 99441 customers are there in given data:

a) **68% customers** are from **southeast Brazil**

b) **14 Total %** are from **south Brazil.**

c) Rest are from **other regions of Brazil**

---

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

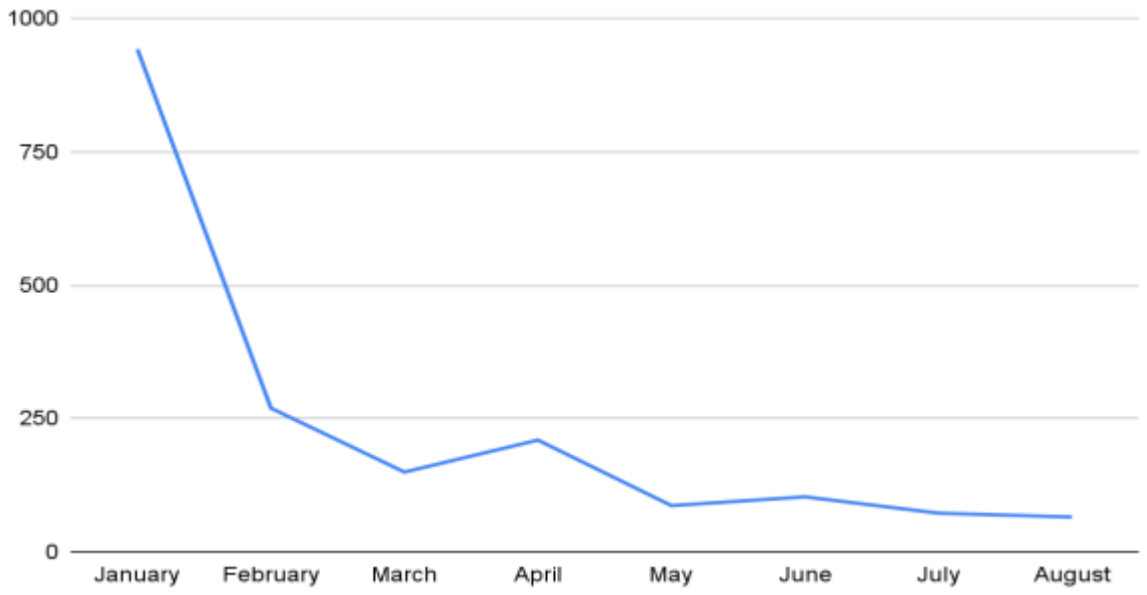
1. Get % increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only)

```sql
SELECT
      *,
      (Total_Cost_2018-
Total_Cost_2017)/Total_Cost_2017 * 100 As Precentage_Increase
FROM
  (SELECT
       Purchase_Month,
       Max(Total_Cost_2017) As Total_Cost_2017,
       Max(Total_Cost_2018) As Total_Cost_2018
   FROM
     (SELECT
          Purchase_Month,
          CASE
              WHEN purchase_year=2017
                   THEN TotalCost
              END AS Total_Cost_2017,
          CASE
              WHEN purchase_year=2018
                   THEN TotalCost
              END AS Total_Cost_2018

      FROM
        (SELECT
             DISTINCT
                 FORMAT_DATE('%B',order_delivered_carrier_date) AS Purchase_Month,
             EXTRACT (YEAR FROM order_delivered_carrier_date) AS Purchase_Year,
             SUM(oi.price) OVER ( PARTITION BY
                                  EXTRACT(MONTH FROM order_delivered_carrier_date),
                                    EXTRACT(YEAR FROM order_delivered_carrier_date)
                                   ORDER BY
                                  EXTRACT(MONTH FROM order_delivered_carrier_date),
                                 EXTRACT(YEAR FROM order_delivered_carrier_date)
                           RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING
                                  )AS TotalCost
          FROM target-sql-362404.Target.orders o
        LEFT JOIN
              target-sql-362404.Target.order_items oi
            ON o.order_id=oi.order_id
        LEFT JOIN target-sql-362404.Target.payments p
            ON p.order_id=o.order_id
    WHERE
          EXTRACT(MONTH FROM order_delivered_carrier_date) BETWEEN 1 AND 8
       AND EXTRACT(YEAR FROM order_delivered_carrier_date) IN (2017,2018)
) as month_wise_data) as year_wise_data
GROUP BY
    Purchase_Month
) AS Year_Month_Data;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	Purchase_Month	Total_Cost_2017	Total_Cost_2018	Precentage_Increase
1	August	601195.7699999999	994737.3899999999	65.4598118679378
2	February	233201.11	861780.62999999989	269.54396572126092
3	March	404459.93999999994	1009827.7899999999	149.6731295564154
4	January	92085.62	959734.78999999992	942.22004478006443
5	July	500103.04	863480.83	72.660584106827258
6	June	447699.68	910074.5199999999	103.27790272264656
7	April	339875.13999999996	1052332.5	209.62326341373489
8	May	555088.97	1036798.0199999999	86.780511960091715



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

```
SELECT
      customer_state,
      AVG(oi.price) AS mean_price,
      AVG(oi.freight_value) AS mean_freight
  FROM
    target-sql-362404.Target.order_items oi
  JOIN
    target-sql-362404.Target.orders o
  ON o.order_id=oi.order_id
  JOIN
    target-sql-362404.Target.customers c
  ON c.customer_id=o.customer_id
GROUP BY
     c.customer_state;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	mean_price	mean_freight
1	MT	148.2971848341233	28.1662843601896
2	MA	145.20415048543691	38.25700242718446
3	AL	180.88921171171171	35.843671171171152
4	SP	109.65362915972931	15.147275390419248
5	MG	120.74857414883068	20.630166806306541
6	PE	145.50832225913598	32.917862679955796
7	RJ	125.11781809451955	20.96092393168248
8	DF	125.77054862842893	21.041354945968383
9	RS	120.33745308740988	21.735804330392945
10	SE	153.04116883116873	36.653168831168855

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

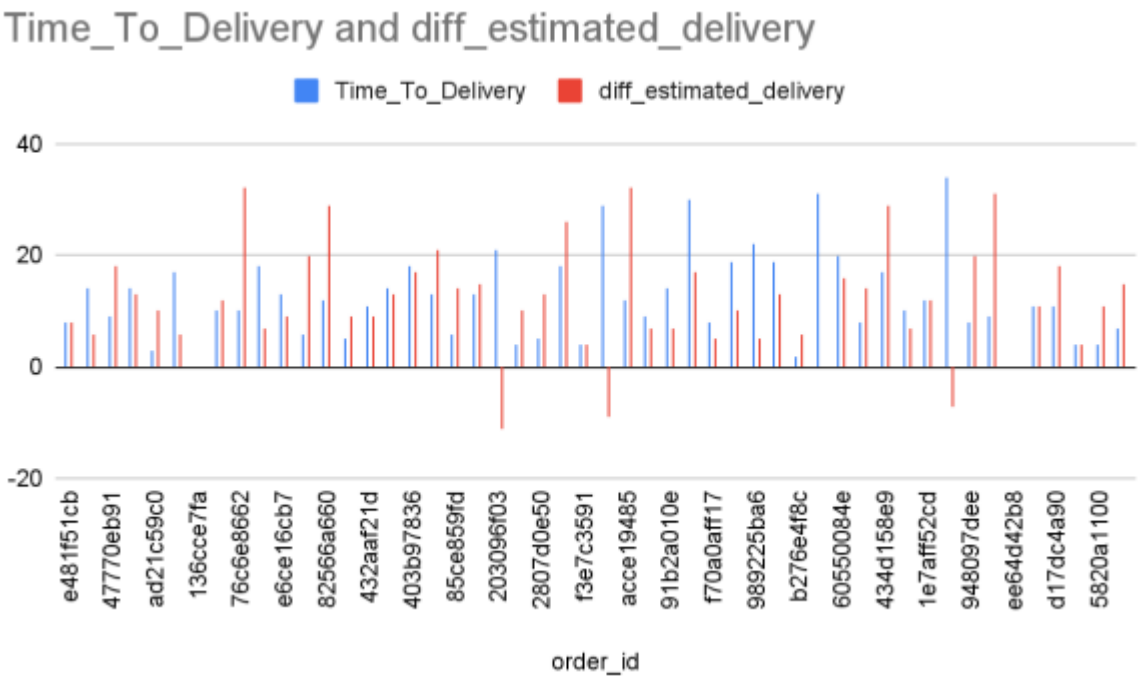
```
SELECT
      order_id,
      DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,day) AS Time
_To_Delivery,
```

```
      DATE_DIFF(order_estimated_delivery_date,order_delivered_customer_date,day) AS
 diff_estimated_delivery
FROM
      `target-sql-362404.Target.orders`;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	order_id	Time_To_Delivery	diff_estimated_delivery
1	1950d777989f6a877539f53795b4c3c3	30	-12
2	2c45c33d2f9cb8ff8b1c86cc28c11c30	30	28
3	65d1e226dfaeb8cdc42f665422522d14	35	16
4	635c894d068ac37e6e03dc54eccb6189	30	1
5	3b97562c3aee8bdedcb5c2e45a50d5e1	32	0
6	68f47f50f04c4cb6774570cfde3a9aa7	29	1
7	276e9ec344d3bf029ff83a161c6b3ce9	43	-4
8	54e1a3c2b97fb0809da548a59f64c813	40	-4
9	fd04fa4105ee8045f6a0139ca5b49f27	37	-1
10	302bb8109d097a9fc6e9cefc5917d1f3	33	-5



1. **Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery**

 # Analysis on Delivery time

- What is the expected time for delivery ?
- By how many days delivery is exceeded from expected delivery time ?
- Was the delivery made on time ?

```
SELECT
```

```
 order_id,
 date_diff(order_delivered_customer_date,order_estimated_delivery_date,day) as days
_exceeded_from_expected_delivery,
 date_diff(order_delivered_customer_date,order_purchase_timestamp,day) as days_take
n_to_delivery,
 date_diff(order_estimated_delivery_date,order_purchase_timestamp,day) as estimated
_time_to_delivery
FROM
  target-sql-362404.Target.orders
WHERE
 order_status = 'delivered'
)
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	order_id	days_exceeded_from_expected_delivery	days_taken_to_delivery	estimated_time_to_delivery	is_on_time
1	635c894d068ac37e6e03dc54eccb6189	-1	30	32	on_time_delivery
2	3b97562c3aee8bdedcb5c2e45a50d5e1	0	32	33	on_time_delivery
3	68f47f50f04c4cb6774570cfde3a9aa7	-1	29	31	on_time_delivery
4	276e9ec344d3bf029ff83a161c6b3ce9	4	43	39	delayed
5	54e1a3c2b97fb0809da548a59f64c813	4	40	36	delayed
6	fd04fa4105ee8045f6a0139ca5b49f27	1	37	35	delayed
7	302bb8109d097a9fc6e9cefc5917d1f3	5	33	28	delayed
8	66057d37308e787052a32828cd007e58	6	38	32	delayed
9	19135c945c554eebfd7576c733d5ebdd	2	36	33	delayed
10	4493e45e7ca1084efcd38ddebf174dda	0	34	33	on_time_delivery

```
SELECT
     DISTINCT
     x.delivery_on_time,
     COUNT(*) OVER (PARTITION BY x.delivery_on_time) AS count_of_records
FROM
   (
    SELECT
          *,
          CASE
              WHEN order_delivered_customer_date IS NULL THEN 'not_yet_delivered'
              WHEN (order_delivered_customer_date < order_estimated_delivery_date) THEN 'on
_time_delivery'
                  ELSE 'delayed'
          END AS delivery_on_time
      FROM
          `target-sql-362404.Target.orders`
   ) AS x ;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	delivery_on_time	count_of_records
1	on_time_delivery	88649
2	not_yet_delivered	2965
3	delayed	7827

```
SELECT
      order_id,
      order_status,
```

```
      date_diff(order_approved_at,order_purchase_timestamp,day) as approval_time,
      date_diff(order_delivered_carrier_date,order_approved_at,day) as time_taken_to_start_
delivery_by_carrier,
      date_diff(order_delivered_customer_date,order_purchase_timestamp,day) as time_taken_f
or_delivery,
      date_diff(order_estimated_delivery_date,order_purchase_timestamp,day) as estimated_ti
me_for_delivery,
      date_diff(order_estimated_delivery_date,order_delivered_customer_date,day) as diff_es
timated_delivery
FROM
        `target-sql-362404.Target.orders`;
```

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	order_id	order_status	approval_time	time_taken_to_start_delivery_by_carrier	time_taken_for_delivery	estimated_time_for_delivery	diff_estimated_delivery
1	f88aac7ebccb37f19725a075331ade3e	shipped	0	9	null	50	null
2	790cd37689193dca0d00d2feb6459164	shipped	0	2	null	6	null
3	49db7943d60b6805c3a41f5474772a09	shipped	0	6	null	44	null
4	063b573b88fc80e516aba87df524f809	shipped	0	22	null	54	null
5	a68ce1686d536ca72bd2dadc4b8671e5	shipped	2	31	null	56	null
6	45973912e490866800c0aea8f63099c8	shipped	0	18	null	54	null
7	cda873529ca7ab71f677d5ec11a40304	shipped	0	38	null	56	null
8	ead20687129da8f5d89d831bb0772867	shipped	0	0	null	41	null
9	6f028ccb7d612af251aa442a1fb8b5d0	shipped	0	1	null	3	null
10	8733c8d440c173e524d2fab8025063f4	shipped	0	0	null	3	null

```
SELECT
 c.customer_state,
 avg(date_diff(order_approved_at,order_purchase_timestamp,day)) as mean_of_approval_time,
 avg(date_diff(order_delivered_carrier_date,order_approved_at,day)) as mean_of_time_taken_t
o_start_delivery_by_carrier,
 avg(date_diff(order_delivered_customer_date,order_purchase_timestamp,day)) as mean_of_time
_taken_for_delivery,
 avg(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day)) as mean_of
_diff_estimated_delivery,
 avg(ois.freight_value) avg_fright_value
FROM
`target-sql-362404.Target.orders` as o
JOIN
`target-sql-362404.Target.customers` as c
on o.customer_id = c.customer_id
JOIN
`target-sql-362404.Target.order_items` as ois
on ois.order_id = o.order_id
group by
c.customer_state
order by
mean_of_time_taken_for_delivery ;
```

Query results

SAVE RESULTS | EXPLORE

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	mean_of_approval_time	mean_of_time_taken_to_start_delivery_by_carrier	mean_of_time_taken_for_delivery	mean_of_diff_estimated_delivery	avg_fright_value
1	SP	0.25196020571621425	2.3062200956937735	8.25960855241909	10.26559438451439	15.147275390419248
2	PR	0.29547038327526171	2.3781645569620209	11.480793060718735	12.533899805275263	20.531651567944248
3	MG	0.27510284930671852	2.3562139284340162	11.515522180072811	12.397151041263502	20.630166806306541
4	DF	0.28512053200332443	2.3922883487007613	12.501486199575384	11.274734607218704	21.041354945968383
5	SC	0.29693486590038271	2.4340672634890033	14.520985846754517	10.6688628599317	21.470368773946436
6	RJ	0.24554061470911126	2.5021085378499763	14.689382157500321	11.14449314293797	20.96092393168248
7	RS	0.3236001925236639	2.2996272889321028	14.708299364095817	13.203000163052323	21.735804330392945
8	GO	0.33647663951993179	2.0869377162629745	14.948177426438281	11.372859025032927	22.766815259322794
9	MS	0.26251526251526286	2.2739557739557759	15.107274969173847	10.337854500616523	23.374884004884006
10	ES	0.28501773049645324	2.5066844919786111	15.192808988764023	9.7685393258427116	22.058776595744682

```
SELECT
```

```
 c.customer_state,
 avg(ois.freight_value) avg_fright_value,
 avg(date_diff(order_delivered_customer_date,order_purchase_timestamp,day)) as time_taken_f
or_delivery,
 avg(date_diff(order_estimated_delivery_date,order_purchase_timestamp,day)) as estimated_ti
me_for_delivery,
 avg(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day)) as diff_es
timated_delivery,
 avg(date_diff(order_delivered_carrier_date,order_approved_at,day)) as time_taken_to_start_
delivery_by_carrier,
 avg(date_diff(order_approved_at,order_purchase_timestamp,day)) as approval_time
FROM
`target-sql-362404.Target.orders` as o
JOIN
`target-sql-362404.Target.customers` as c
on o.customer_id = c.customer_id
JOIN
`target-sql-362404.Target.order_items` as ois
on ois.order_id = o.order_id
WHERE
 o.order_delivered_customer_date is not null and
 o.order_delivered_carrier_date is not null and
 o.order_approved_at is not null
GROUP BY
 c.customer_state
ORDER BY
 time_taken_for_delivery;
```

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	avg_fright_value	time_taken_for_delivery	estimated_time_for_delivery	diff_estimated_delivery	time_taken_to_start_delivery_by_carrier	approval_time
1	SP	15.114605594676664	8.2583930917156252	18.868230075155527	10.264853457372388	2.3037017895213019	0.25169584598488143
2	PR	20.471816250663817	11.480793060718735	24.379359178615747	12.533899805275263	2.3669676048858146	0.29244114002478416
3	MG	20.626686541737637	11.514635279541569	24.259485829332604	12.394300758866368	2.3507046616075504	0.26490630323679942
4	DF	21.072161358811066	12.501486199575384	24.09426751592358	11.274734607218704	2.3719745222929953	0.28195329087048926
5	SC	21.506627623230841	14.520985846754517	25.522938018545617	10.6688628599317	2.4304538799414277	0.29502196193265112
6	RJ	20.911051403521135	14.690094039454156	26.088736477409391	11.1431096655589	2.495510146362149	0.24676518419005902
7	RS	21.614624041755043	14.709182841298317	28.267656173544296	13.201272223128383	2.2896754199967386	0.32066547055945205
8	GO	22.562867808519979	14.948177426438281	26.629776021080428	11.372859025032927	2.0790513833992126	0.33201581027667953
9	MS	23.35090012330458	15.107274969173847	25.689272503082627	10.337854500616523	2.2737361282367421	0.25770653514180009
10	ES	22.02897977528092	15.192808988764023	25.233707865168512	9.7685393258427116	2.5119101123595562	0.28584269662921313

```
SELECT
      c.customer_state,
      AVG(oi.freight_value) AS mean_freight_value,
      AVG(DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,day)) as Mean_Ti
me_To_Delivery,
      AVG(DATE_DIFF(order_estimated_delivery_date,order_delivered_customer_date,day)) as AV
G_diff_estimated_delivery
FROM
     `target-sql-362404.Target.order_items` oi
JOIN
     `target-sql-362404.Target.orders` o
    ON o.order_id=oi.order_id
JOIN
     `target-sql-362404.Target.customers` c
    ON c.customer_id=o.customer_id
GROUP BY
    c.customer_state;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	mean_freight_value	Mean_Time_To_Delivery	AVG_diff_estimated_delivery
1	MT	28.1662843601896	17.508196721311482	13.639344262295094
2	MA	38.25700242718446	21.203750000000017	9.1099999999999923
3	AL	35.843671171171152	23.992974238875881	7.9765807962529349
4	SP	15.147275390419248	8.25960855241909	10.26559438451439
5	MG	20.630166806306541	11.515522180072811	12.397151041263502
6	PE	32.917862679955796	17.792096219931292	12.552119129438733
7	RJ	20.96092393168248	14.689382157500321	11.14449314293797
8	DF	21.041354945968383	12.501486199575384	11.274734607218704
9	RS	21.735804330392945	14.708299364095817	13.203000163052323
10	SE	36.653168831168855	20.978666666666651	9.1653333333333276

Insights:

1) **After purchase being made,** the **average time for approving** the order **by seller** is **0.26** days and median time is 0 , **means with in a day.**
2) **Average time** taken for a **carrier to start the delivery is 2 and a half day.**
3) **Average time** taken to **complete delivery is 12 days** and **median of delivery time is 10 days.**
4) Estimated **time delivery average** is **23 days.**
5) There is a **positive correlation** between **freight value and delivery time.**
6) **Long distance deliveries** are having **higher freight values** and also takes **more time for delivery**
7) **States São Paulo ,Paraná,Minas Gerais, Distrito Federal ,Santa Catarina and Rio de Janeiro** are some of the states having relatively **faster delivery time.**
8) **Alagoas, Amazonas, Amapá ,Pará and Roraima** are some states have relatively **very slow delivery time.**

2. Sort the data to get the following:

1. Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5

```
SELECT
          c.customer_state,
          AVG(oi.freight_value) AS mean_freight_value
    FROM
         `target-sql-362404.Target.order_items` oi
    JOIN
         `target-sql-362404.Target.orders` o
        ON o.order_id=oi.order_id
    JOIN `target-sql-362404.Target.customers` c
        ON c.customer_id=o.customer_id
    GROUP BY
        c.customer_state
    ORDER BY
        mean_freight_value DESC
    LIMIT 5;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	mean_freight_value
1	RR	42.984423076923093
2	PB	42.723803986710941
3	RO	41.069712230215842
4	AC	40.073369565217405
5	PI	39.147970479704767

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

```
SELECT
      c.customer_state,
      AVG(DATE_DIFF(order_delivered_customer_date,order_delivered_carrier_date,day)
 as Avg_Time_To_Delivery
FROM
     `target-sql-362404.Target.order_items` oi
JOIN
     `target-sql-362404.Target.orders` o
    ON o.order_id=oi.order_id
JOIN `target-sql-362404.Target.customers` c
    ON c.customer_id=o.customer_id
GROUP BY
    c.customer_state
ORDER BY
    Avg_Time_To_Delivery DESC
LIMIT 5;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	Avg_Time_To_Delivery
1	RR	24.15217391304348
2	AP	24.135802469135808
3	AM	23.073619631901835
4	AL	20.362997658079646
5	PA	19.840607210626143

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

```
SELECT
```

```
 c.customer_state,
     AVG(DATE_DIFF(order_estimated_delivery_date,order_delivered_carrier_date,day)
as Mean_Expected_Time_To_Delivery
FROM
     `target-sql-362404.Target.order_items` oi
JOIN
     `target-sql-362404.Target.orders` o
    ON o.order_id=oi.order_id
JOIN `target-sql-362404.Target.customers` c
    ON c.customer_id=o.customer_id
GROUP BY
    c.customer_state
ORDER BY
    Mean_Expected_Time_To_Delivery DESC
LIMIT 5;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	Mean_Expected_Time_To_Delivery
1	AM	42.290909090909061
2	AP	42.000000000000007
3	RR	40.745098039215677
4	AC	37.304347826086961
5	RO	35.864468864468805

---

6. Payment type analysis:

Analysing Payment details:

```
SELECT
 * EXCEPT(is_generated, generation_expression, is_stored, is_updatable,is_hidden
        ,is_system_defined,is_partitioning_column,
         clustering_ordinal_position,collation_name,column_default,ordinal_position,table_c
atalog)
 FROM
 `target-sql-362404.Target.INFORMATION_SCHEMA.COLUMNS`
 WHERE
 table_name = 'payments';
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	table_schema	table_name	column_name	is_nullable	data_type
1	Target	payments	order_id	YES	STRING
2	Target	payments	payment_sequential	YES	INT64
3	Target	payments	payment_type	YES	STRING
4	Target	payments	payment_installments	YES	INT64
5	Target	payments	payment_value	YES	FLOAT64

```sql
SELECT
      payment_type,
      COUNT(DISTINCT(order_id)) AS Number_of_sales_per_payment_type,
      SUM(payment_value) AS Total_Payment_per_payment_type
FROM
     `target-sql-362404.Target.payments`
GROUP BY
        payment_type;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	payment_type	Number_of_sales_per_payment_type	Total_Payment_per_payment_type
1	credit_card	76505	12542084.189999517
2	voucher	3866	379436.87000000343
3	not_defined	3	0.0
4	debit_card	1528	217989.78999999937
5	UPI	19784	2869361.2699999218

Insights:

1) **78% payments** are done using **credit card** and **17.92%** are done with **UPI.**

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

```sql
SELECT
          EXTRACT(YEAR FROM order_purchase_timestamp) AS Year,
          FORMAT_TIMESTAMP("%b %Y", order_purchase_timestamp) AS Month_year_purchase_da
    te,
          p.Payment_Type,
          COUNT(o.order_id) AS Number_of_Orders
    FROM
         `target-sql-362404.Target.payments` as p
    JOIN
         `target-sql-362404.Target.orders` as o
        ON o.order_id = p.order_id
    GROUP BY
        Year,
        Month_year_purchase_date,
        p.payment_type
    ORDER BY
        Year,
        Month_year_purchase_date,
        p.payment_type;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	Year	Month_year_purchase_date	Payment_Type	Number_of_Orders
1	2016	Dec 2016	credit_card	1
2	2016	Oct 2016	UPI	63
3	2016	Oct 2016	credit_card	254
4	2016	Oct 2016	debit_card	2
5	2016	Oct 2016	voucher	23
6	2016	Sep 2016	credit_card	3
7	2017	Apr 2017	UPI	496
8	2017	Apr 2017	credit_card	1846
9	2017	Apr 2017	debit_card	27
10	2017	Apr 2017	voucher	202
11	2017	Aug 2017	UPI	938

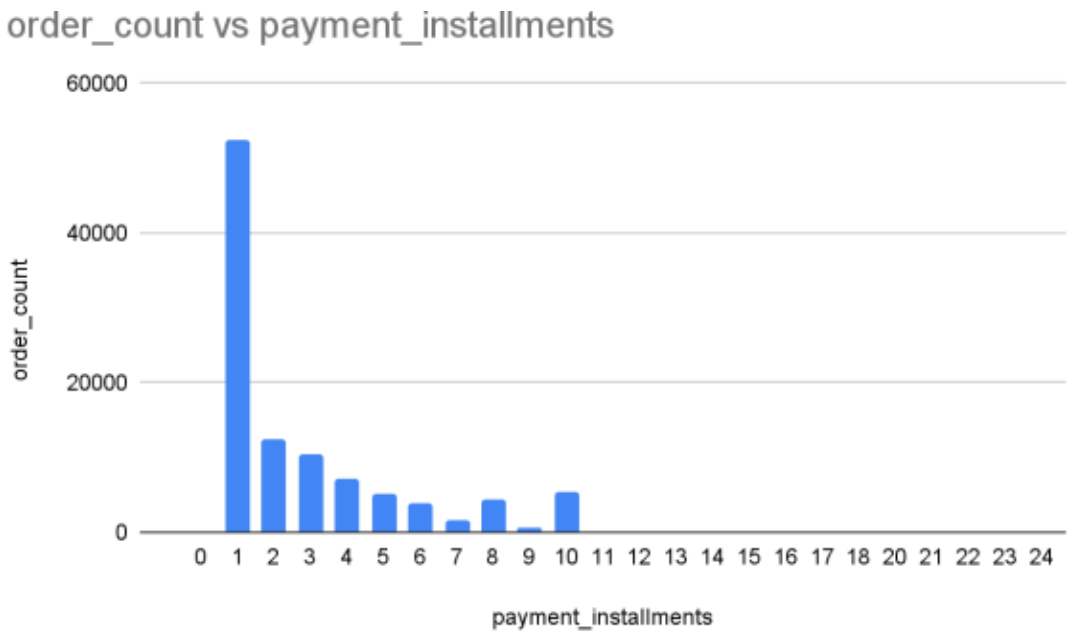
2. Distribution of payment installments and count of orders

```
SELECT
      payment_installments,
      COUNT(DISTINCT(order_id)) AS number_of_orders
FROM
    `target-sql-362404.Target.payments`
GROUP BY payment_installments ;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	payment_installments	number_of_orders
1	0	2
2	1	49060
3	2	12389
4	3	10443
5	4	7088
6	5	5234
7	6	3916
8	7	1623
9	8	4253
10	9	644
11	10	5315



Insights:

1) **Majority** of the orders are purchased with **1 payment instalment.**
2) Also **more than 5 instalments purchases are relatively very low.**

Insights & Recommendation:

All Insights :

1) Customers' **information's:**

- We have 99,441 customers of data available, out of which 96096 Unique Customers IDs with 14994 different locations of customers
- Customers are from different 4119 cities and 27 states in Brazil
- From a total of 99441 orders, 1107 are shipped, 625 were cancelled, and 96478 are delivered
- 68% of customers are from southeast Brazil, 14% are from south Brazil and the rest are from other regions of Brazil

2) **Analysis of sales and revenue as per time:**

- Time period for which the data is given is 25 months
- As compared to 2017 the revenue has increased in 2018 by 21%
- The Average number of orders is higher during the month of November. September and October have comparatively low orders on average whereas May, July, and August have a higher number of average orders compared to other months.
- Tuesday, Monday and Wednesdays have a relatively higher number of orders

3) **Increasing trend:**

- There is an increasing trend in orders, trend sustains during 2018. There is a slight fall we can observe during October 2017 followed by a great hike in November month and again a fall at end of December 2017 and January 2018.
- We can observe the trend of increasing orders with time and also for revenue.

- We can observe there's an 81.5% growth increase in terms of orders and a 70.7% growth increment in terms of revenue in January from 2017 to 2018.
- Growth rate for July and August from 2017 to 2018 is relatively very low whereas 2017-February, march, and November were the highest growing sale month in comparison to the previous month.

4) **Customer_purchasing Behavior:**

- Customers are purchasing orders from morning 8 am till late evening 11 pm.
- Afternoon and evening orders are very high as compared to the morning, and night time.

5) **Delivery time:**

- After the purchase is made, the average time for approving the order by the seller is 0.26 days and the median time is 0, which means within a day.
- Average time taken for a carrier to start the delivery is 2 and a half days.
- Average time taken to complete delivery is 12 days and the median delivery time is 10 days.
- Estimated time delivery average is 23 days.
- There is a positive correlation between freight value and delivery time.
- Long-distance deliveries are having higher freight values and also take more time for delivery
- States São Paulo, Paraná, Minas Gerais, Distrito Federal, Santa Catarina, and Rio de Janeiro are some of the states having relatively faster delivery times.
- Alagoas, Amazonas, Amapá, Pará, and Roraima are some states that have relatively very slow delivery times.

6) **Region and State-wise Analysis :**

- São Paulo, Rio de Janeiro, Minas Gerais, Rio Grande do Sul, and Paraná are the top 5 highest orders states and also generate the highest revenue.
- More than 80% of orders are coming from south, southeast, and northeast Brazil. 90% of the revenue is coming from south, southeast, and northeast Brazil

7) **Payment type-related info :**

- 78% of payments are done using a credit card and 17.92% are done with UPI.
- Majority of the orders are purchased at 1 payment installment.
- More than 5 installments purchases are relatively very low.

Recommendations:

1. From the distribution and statistical analysis we can observe the average time to complete the delivery is 12 days. which should be reduced to at least half, as due to high competition in the e-commerce market, it is vital to do so
2. In order to reduce the delivery time, if we look at the average time for the carrier to start the delivery itself takes at least 2 and a half days and the order approval time is 0.26 days. These two should be optimized as low as possible, which can result in delivery faster.
3. If we look at the Top states where delivery is really slow compared to the estimated date, they are all from the north Brazil region. Delivering faster in the northern states may create and increase new customers and revenue from the north.

4. As per Analysis products belonging to the “bed table bath” category are being sold max among all available categories, we could produce more items related to this category.

5. Increasing the network in north Brazil, having small towns can help increase the customer base. As north Brazil has the world’s largest river and most extensive rain forest, must be a good travel destination, introducing necessary survival/ camping/adventure products can help increase revenue and order from the northern region

6. As per the analysis Credit card payments are more. So target can give offers to customers paying with a credit card and can also give them an Interest-free EMI scheme.

7. It was observed that an increasing trend in revenue and orders over time, yet during October and January sales are decreasing probably after Festival Sales. Introducing possible discounts on the not-so-running products can help sell more products during those low-going months.