Business Case Study: Target SQL

This Business case study has the information of 100K orders from 2016 to 2018 made at Target in Brazil.

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

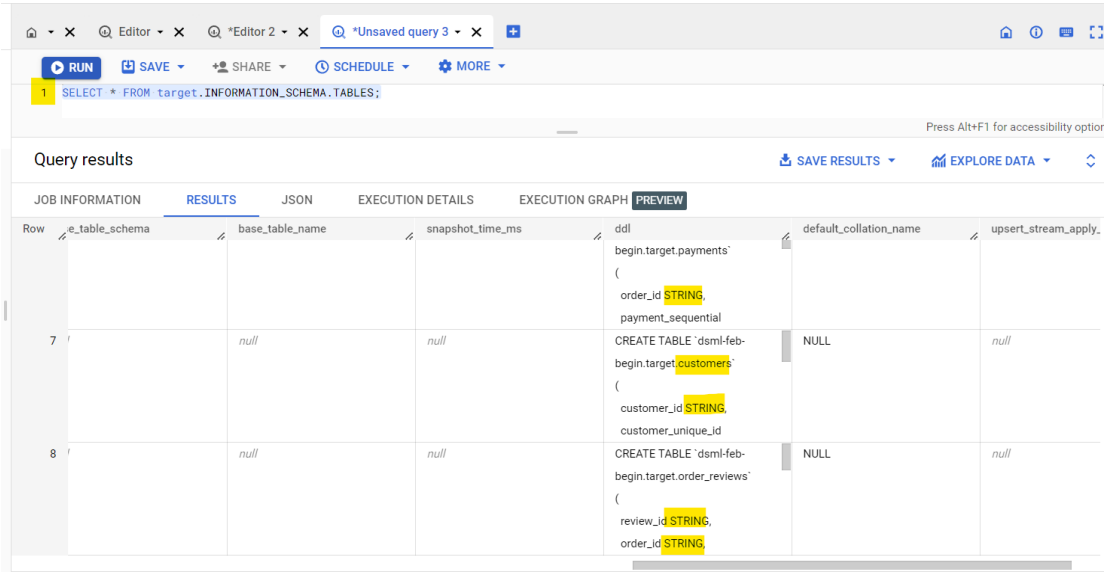
1.1 Data type of columns in the table.

There are 8 tables in the dataset. The dataset name is defined here as "**target**".

SQL Query to get the column data types for each of the table in the dataset:

```
SELECT * FROM target.INFORMATION_SCHEMA.TABLES;
```

Result:
Customers:



Features	Description	Data type
customer_id	Id of the consumer who made the purchase.	String
customer_unique_id	Unique Id of the consumer.	String
customer_zip_code_prefix	Zip Code of the location of the consumer.	Int64
customer_city	Name of the City from where order is made.	String
customer_state	State Code from where order is	String

	made(Ex- sao paulo-SP).	

GeoLocation:

Features	Description	Data Type
geolocation_zip_code_prefix	first 5 digits of zip code	Int64
geolocation_lat	latitude	Float64
geolocation_lng	longitude	Float64
geolocation_city	city name	String
geolocation_state	state	String

Order_items:

Features	Description	Data Type
order_id	A unique id of order made by the consumers.	String
order_item_id	A Unique id given to each item ordered in the order.	Int64
product_id	A unique id given to each product available on the site.	String
seller_id	Unique Id of the seller registered in Target.	String
shipping_limit_date	The date before which shipping of the ordered product must be completed.	Timestamp
price	Actual price of the products ordered .	Float64
freight_value	Price rate at which a product is delivered from one point to another.	Float64

Order_reviews:

Features	Description	Data Type
review_id	Id of the review given on the product ordered by the order id.	String
order_id	A unique id of order made by the consumers.	String
review_score	review score given by the customer for each order on the scale of 1–5.	Int64

review_comment_title	Title of the review	String
review_creation_date	Timestamp of the review when it is created.	Timestamp
review_answer_timestamp	Timestamp of the review answered.	Timestamp

Orders:

Features	Description	Data Type
order_id	A unique id of order made by the consumers.	String
customer_id	Id of the consumer who made the purchase.	String
order_status	status of the order made i.e delivered, shipped etc.	String
order_purchase_timestamp	Timestamp of the purchase.	Timestamp
order_delivered_carrier_date	delivery date at which carrier made the delivery.	Timestamp
order_delivered_customer_date	date at which customer got the product.	Timestamp
order_estimated_delivery_date	estimated delivery date of the products.	Timestamp

Payments:

Features	Description	Data Type
order_id	A unique id of order made by the consumers.	String
payment_sequential	sequences of the payments made in case of EMI.	Int64
payment_type	mode of payment used.(Ex-Credit Card)	String
payment_installments	number of installments in case of EMI purchase.	Int64
payment_value	Total amount paid for the purchase order.	Float64

Products:

Features	Description	Data Type
product_id	A unique identifier for the proposed project.	String
product_category_name	Name of the product category	String
product_name_length	length of the string which specifies the name given to the products ordered.	Int64
product_description_length	length of the description written for each product ordered on the site.	Int64
product_photos_qty	Number of photos of each product ordered available on the shopping portal.	Int64
product_weight_g	Weight of the products ordered in grams.	Int64
product_length_cm	Length of the products ordered in centimeters.	Int64
product_height_cm	Height of the products ordered in centimeters.	Int64
product_width_cm	width of the product ordered in centimeters.	Int64

Sellers:

Features	Description	Data Type
seller_id	Unique Id of the seller registered	String
seller_zip_code_prefix	Zip Code of the location of the seller.	Int64
seller_city	Name of the City of the seller.	String
seller_state	State Code (Ex- sao paulo-SP)	String

1.2 : The Time period for which the data is given

```sql
SELECT
  DISTINCT EXTRACT(YEAR FROM DATE(order_purchase_timestamp)) AS Years
FROM `target.orders`;
```

Query results

JOB INFORMATION | RESULTS | JSON | E

Row	Years
1	2017
2	2018
3	2016

The time period of orders placed by the customer is over the years 2016,2017 and 2018.

1.3: Cities and States of customers ordered during the given period

```
SELECT
  DISTINCT customer_city,
  customer_state
FROM `target.customers`
```

```
4 SELECT
5   DISTINCT customer_city,
6   customer_state
7 FROM `target.customers`
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_city	customer_state
1	acu	RN
2	ico	CE
3	ipe	RS
4	ipu	CE
5	ita	SC
6	itu	SP
7	jau	SP
8	luz	MG
9	poa	SP
10	uba	MG

2. In-depth exploration:

2.1 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?

From the given dataset we can clearly say that over the years, there is a tremendous increase in the number of orders made online.

```
WITH orderDate_table AS
(SELECT
  customer_id,
  order_id,
  order_status,
  DATE(order_purchase_timestamp) AS order_date
FROM `target.orders`)

SELECT
  order_date,
  COUNT(DISTINCT order_id) AS total_orders_per_date
FROM orderDate_table
GROUP BY order_date
ORDER BY order_date;
```

Result: The total number of orders is increasing from barely 1 per date in 2016 to 300 per date in the year 2018.

Query results

JOB INFORMATION | RESULTS | JSON | EXECUT

Row	order_date	total_orders_per_date
1	2016-09-04	1
2	2016-09-05	1
3	2016-09-13	1
4	2016-09-15	1
5	2016-10-02	1
6	2016-10-03	8
7	2016-10-04	63
8	2016-10-05	47
9	2016-10-06	51
10	2016-10-07	46

Query results

JOB INFORMATION | RESULTS | JSON | EXE

Row	order_date	total_orders_per_date
601	2018-08-13	292
602	2018-08-14	316
603	2018-08-15	288
604	2018-08-16	320
605	2018-08-17	257
606	2018-08-18	198
607	2018-08-19	204
608	2018-08-20	256
609	2018-08-21	243
610	2018-08-22	187

Total Orders per Year: Over the years we can see the increasing trend in total number of orders.

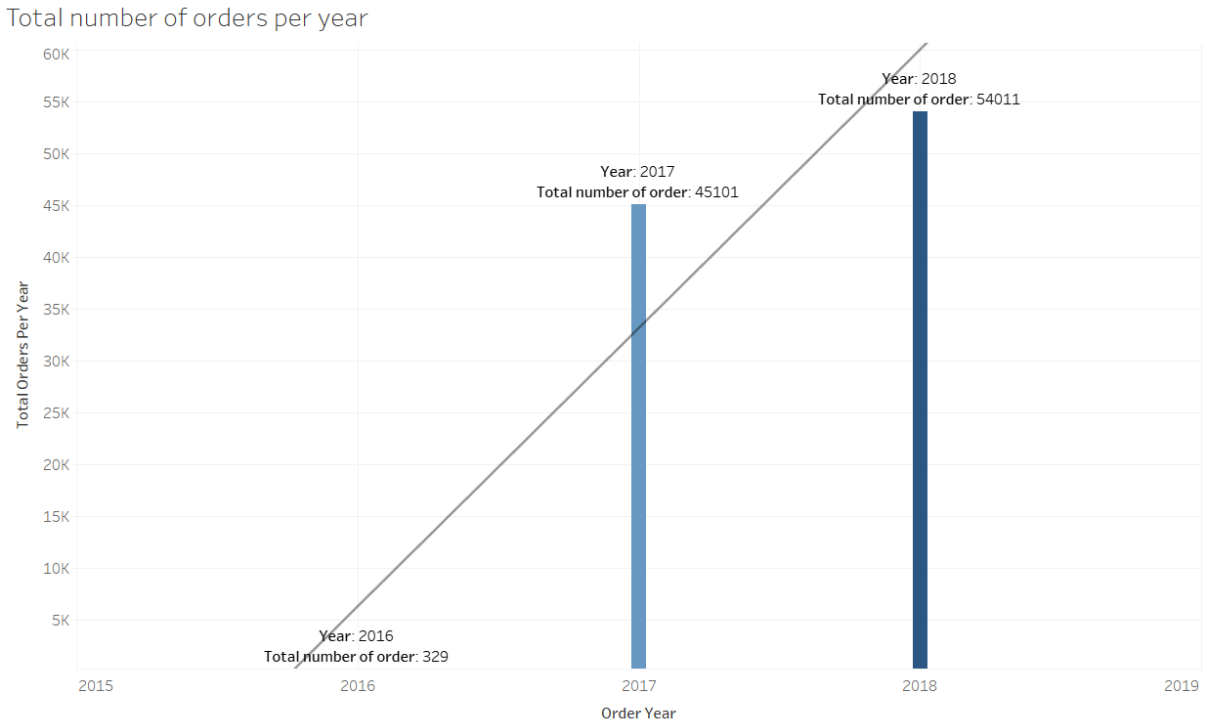
```
WITH orderDate_table AS
(SELECT
  customer_id,
  order_id,
  order_status,
  DATE(order_purchase_timestamp) AS order_date,
  EXTRACT(YEAR FROM DATE(order_purchase_timestamp)) AS order_year
FROM `target.orders`)

SELECT
  order_year,
  COUNT(DISTINCT order_id) AS total_orders_per_year
FROM orderDate_table
GROUP BY order_year
ORDER BY order_year;
```

Query results

JOB INFORMATION | RESULTS | JSON

Row	order_year	total_orders_per_year
1	2016	329
2	2017	45101
3	2018	54011

Tableau visualisation for the increasing trend of e-commerce in Brazil.



We can see some seasonality as well in the increasing trends of e-commerce.
The number of orders are the highest in the months of **October-November** as it is the festive months of the year.

```
WITH orderDate_table AS
(SELECT
  customer_id,
  order_id,
  order_status,
  DATE(order_purchase_timestamp) AS order_date,
  EXTRACT(YEAR FROM DATE(order_purchase_timestamp)) AS order_year,
  EXTRACT(MONTH FROM DATE(order_purchase_timestamp)) AS order_month
FROM `target.orders`)

SELECT
  order_year,
  order_month,
  COUNT(DISTINCT order_id) AS total_orders
FROM orderDate_table
GROUP BY order_year,order_month
ORDER BY order_year,order_month;
```

Result:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION

Row	order_year	order_month	total_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

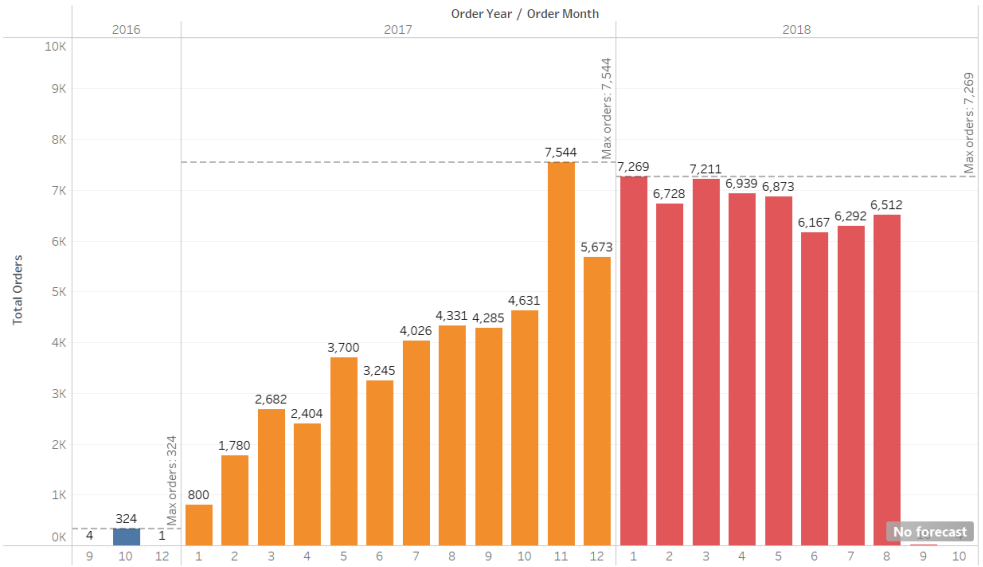
Query results

JOB INFORMATION | RESULTS | JSON | E

Row	order_year	order_month	total_orders
9	2017	6	3245
10	2017	7	4026
11	2017	8	4331
12	2017	9	4285
13	2017	10	4631
14	2017	11	7544
15	2017	12	5673

Tableau visualisation of the seasonality.

Months with Peak number of orders in a year



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

The given timestamp in the orders dataset is the timestamp with UTC timezone. We have to convert this time zone to Brazil time zone (ie: Brasilia time zone). Brasilia time zone is 3 hours behind that of the UTC timezone.

So we will be subtracting 3 hours from the UTC timestamp to get the timestamp in Brasilia time zone.

After conversion of timezone, I have considered

1. Time between **00** hours and **06** hours as "**Dawn**"
2. Time between **06** hours and **12** hours as "**Morning**"
3. Time between **12** hours to **16** hours as "**Afternoon**"
4. Time between **16** hours and **23** hours as "**Evening/ Afternoon**"

According to these considerations the common time zone when Brazilians tend to buy/make an order is "**Morning**" ie: Between **6 AM** and **12 PM**. The number of orders made in the morning over the years 2016, 2017 and 2018 is around **38300**.

SQL Query:

```
WITH converted_purchase_time AS
(SELECT
  order_purchase_timestamp,
  TIME(order_purchase_timestamp) AS time_of_purchase_UTC,
  TIME_SUB(TIME(order_purchase_timestamp),INTERVAL 3 HOUR) AS time_of_purchase_BRT
FROM `target.orders`),
part_of_day AS
(SELECT
  order_purchase_timestamp,
  time_of_purchase_UTC,
  time_of_purchase_BRT,
  EXTRACT(HOUR FROM time_of_purchase_BRT) AS Hour_of_purchase,
  EXTRACT(MINUTE FROM time_of_purchase_BRT) AS Minute_of_purchase,
  CASE
    WHEN EXTRACT(HOUR FROM time_of_purchase_BRT) BETWEEN 0 AND 6 THEN "Dawn"
    WHEN EXTRACT(HOUR FROM time_of_purchase_BRT) BETWEEN 6 AND 12 THEN "Morning"
    WHEN EXTRACT(HOUR FROM time_of_purchase_BRT) BETWEEN 12 AND 16 THEN "Afternoon"
    WHEN EXTRACT(HOUR FROM time_of_purchase_BRT) BETWEEN 16 AND 23 THEN
"Evening/Night"
  END AS timezone_day
FROM converted_purchase_time)
```

```sql
SELECT
  timezone_day,
  COUNT(time_of_purchase_BRT) AS No_of_orders_per_timezone
FROM part_of_day
GROUP BY timezone_day;
```

Result:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTI

Row	timezone_day	No_of_orders_pe
1	Morning	38291
2	Evening/Night	26423
3	Afternoon	24576
4	Dawn	10151

3. Evolution of e-commerce orders in the Brazilian region:

3.1 Get month on month orders by state:

SQL Query:

```sql
WITH orderDate_table AS
(SELECT
  customer_id,
  order_id,
  order_status,
  DATE(order_purchase_timestamp) AS order_date,
  EXTRACT(YEAR FROM DATE(order_purchase_timestamp)) AS order_year,
  EXTRACT(MONTH FROM DATE(order_purchase_timestamp)) AS order_month
FROM `target.orders`)

SELECT
  cust.customer_state,
  ord.order_month,
  COUNT(ord.order_id) AS total_orders
```

```
FROM orderDate_table AS ord JOIN `target.customers` AS cust ON ord.customer_id =
cust.customer_id
GROUP BY cust.customer_state,ord.order_month
ORDER BY order_month ASC;
```

From this Query we will be able to get the total number of orders that were placed in a particular month from a particular state.

Result: Here customer_state represents the state code of the customer's state from where he placed the order.

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

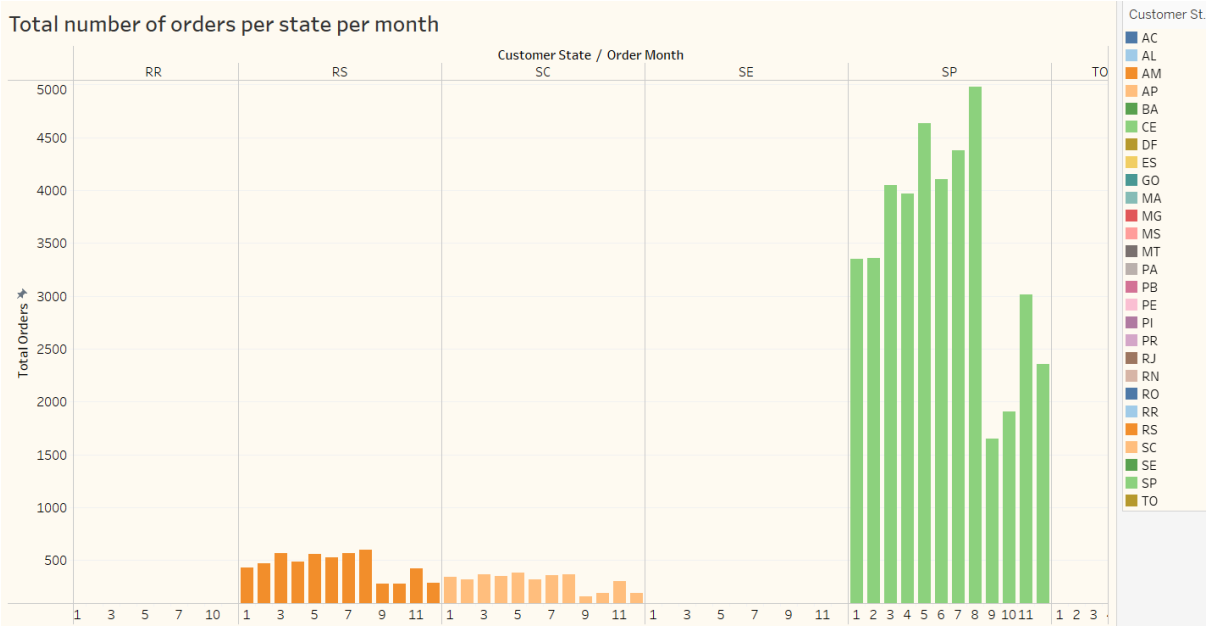
Row	customer_state	order_month	total_orders
1	RJ	1	990
2	SP	1	3351
3	DF	1	151
4	RS	1	427
5	CE	1	99
6	PE	1	113
7	PR	1	443
8	BA	1	264
9	MG	1	971
10	RN	1	51

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	order_month	total_orders
297	RS	12	283
298	SP	12	2357
299	RJ	12	783
300	PR	12	271
301	MG	12	691
302	ES	12	113
303	BA	12	192
304	MS	12	36
305	MA	12	41
306	DF	12	131

Tableau Visualisation:



3.2. Distribution of customers across the states in Brazil:

There are about 27 states in Brazil and the customers are spread across these states in brazil. We can get the COUNT of customers present in a particular state by using the below query.

SQL Query:

```
SELECT
  customer_state,
  COUNT(DISTINCT customer_id) AS total_no_of_customers
FROM `target.customers`
GROUP BY customer_state;
```

Result:

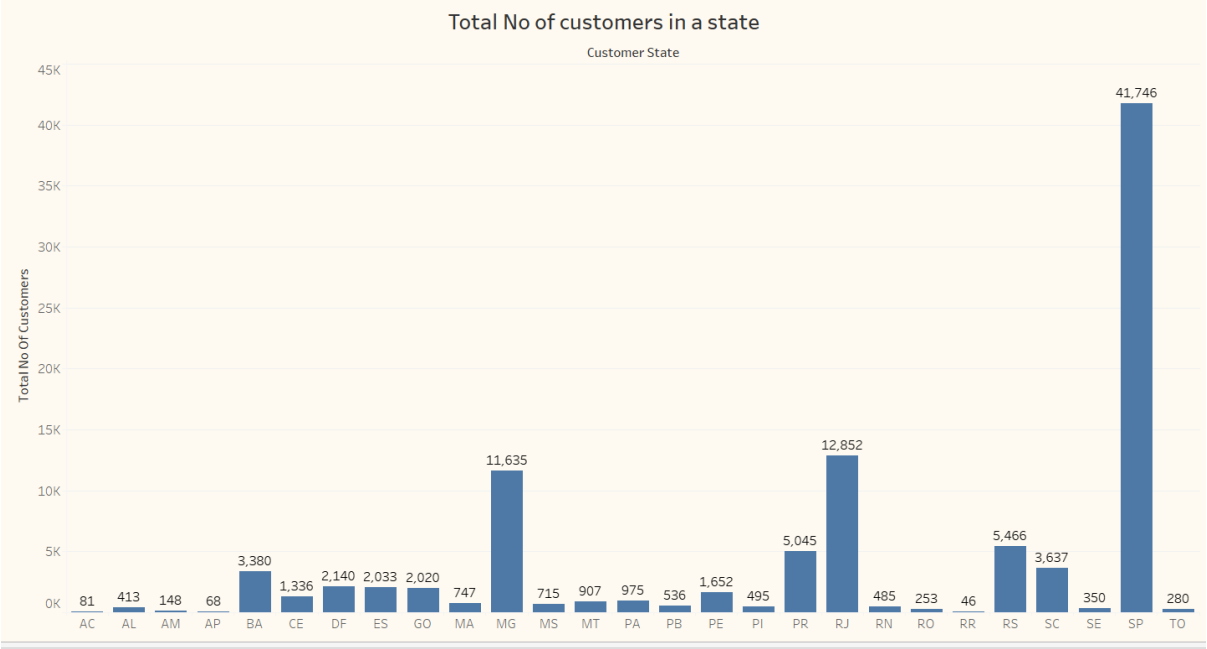
Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAI

Row	customer_state	total_no_of_customers
1	RN	485
2	CE	1336
3	RS	5466
4	SC	3637
5	SP	41746
6	MG	11635
7	BA	3380
8	RJ	12852
9	GO	2020
10	MA	747

The State with state code “**SP**” is having the maximum number of customers from where they are using the e-commerce application.

Tableau Visualisation:



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

4.1 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.

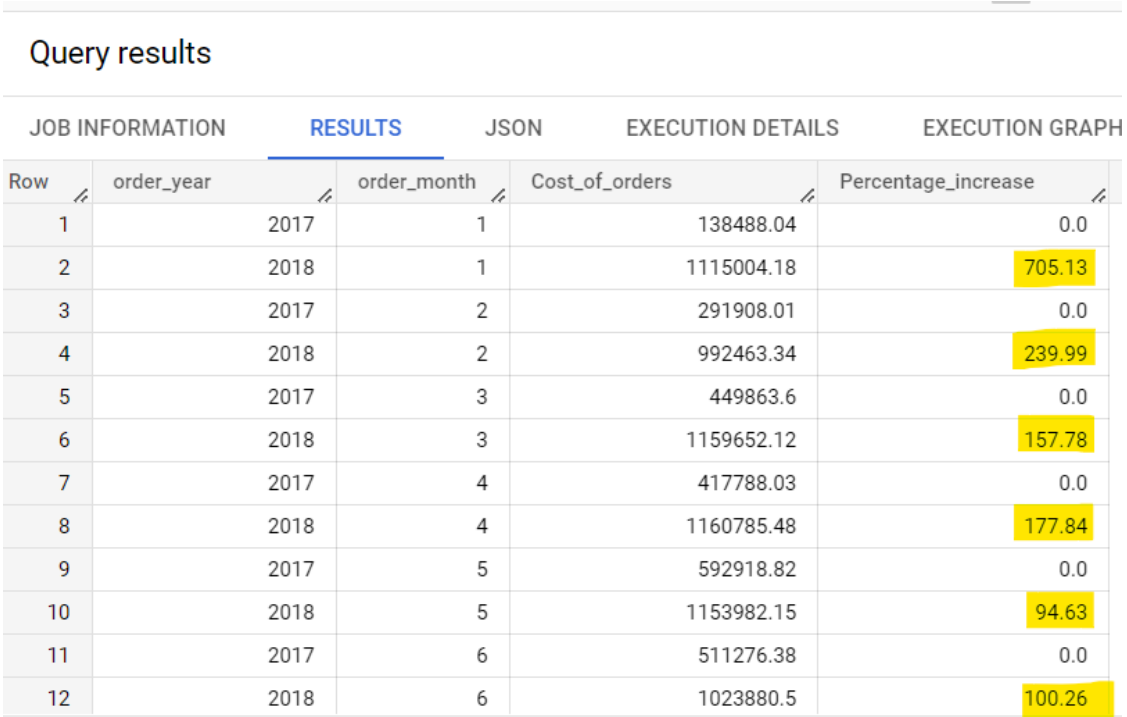
There has been a significant percentage increase in the cost of orders from the year 2017 to 2018 for each month

SQL Query:

```
WITH orderDate_table AS
(SELECT
  customer_id,
  order_id,
  order_status,
  DATE(order_purchase_timestamp) AS order_date,
  EXTRACT(YEAR FROM DATE(order_purchase_timestamp)) AS order_year,
  EXTRACT(MONTH FROM DATE(order_purchase_timestamp)) AS order_month
FROM `target.orders`
WHERE EXTRACT(YEAR FROM DATE(order_purchase_timestamp)) BETWEEN 2017 AND 2018
```

```
AND EXTRACT(MONTH FROM DATE(order_purchase_timestamp)) BETWEEN 1 AND 8),
OrdersCost AS
(SELECT
  ord.order_year,
  ord.order_month,
  SUM(pay.payment_value) AS Cost_of_orders
FROM orderDate_table AS ord JOIN `target.payments` AS pay ON ord.order_id =
pay.order_id
GROUP BY ord.order_year,ord.order_month
ORDER BY ord.order_year, ord.order_month)

SELECT
  order_year,
  order_month,
  ROUND(Cost_of_orders,2) AS Cost_of_orders,
  ROUND((ABS(FIRST_VALUE(Cost_of_orders) OVER(PARTITION BY order_month ORDER BY
order_year) - LAST_VALUE(Cost_of_orders) OVER(PARTITION BY order_month ORDER BY
order_year))/(FIRST_VALUE(Cost_of_orders) OVER(PARTITION BY order_month ORDER BY
order_year)))*100,2) AS Percentage_increase
FROM OrdersCost;
```

Result: Here in the results we can see that for each month between the years 2017 and 2018, we can see the significant % increase in the cost of orders.



Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	order_year	order_month	Cost_of_orders	Percentage_increase
1	2017	1	138488.04	0.0
2	2018	1	1115004.18	705.13
3	2017	2	291908.01	0.0
4	2018	2	992463.34	239.99
5	2017	3	449863.6	0.0
6	2018	3	1159652.12	157.78
7	2017	4	417788.03	0.0
8	2018	4	1160785.48	177.84
9	2017	5	592918.82	0.0
10	2018	5	1153982.15	94.63
11	2017	6	511276.38	0.0
12	2018	6	1023880.5	100.26

4.2 Mean & Sum of price and freight value by customer state

SQL Query:

```
SELECT

  cust.customer_state,

  ROUND(AVG(ord_items.price),2) AS Mean_price,

  ROUND(SUM(ord_items.freight_value),2) AS Sum_freightVal

FROM `target.customers` AS cust JOIN `target.orders` AS ord ON cust.customer_id = ord.customer_id

JOIN `target.order_items` AS ord_items ON ord.order_id = ord_items.order_id

GROUP BY cust.customer_state;
```

Result: For each of the States we can aggregate the Mean price and the sum of freight value.

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	Mean_price	Sum_freightVal
1	RN	156.97	18860.1
2	CE	153.76	48351.59
3	RS	120.34	135522.74
4	SC	124.65	89660.26
5	SP	109.65	718723.07
6	MG	120.75	270853.46
7	BA	134.6	100156.68
8	RJ	125.12	305589.31
9	GO	126.27	53114.98
10	MA	145.2	31523.77

5. Analysis on sales, freight and delivery time

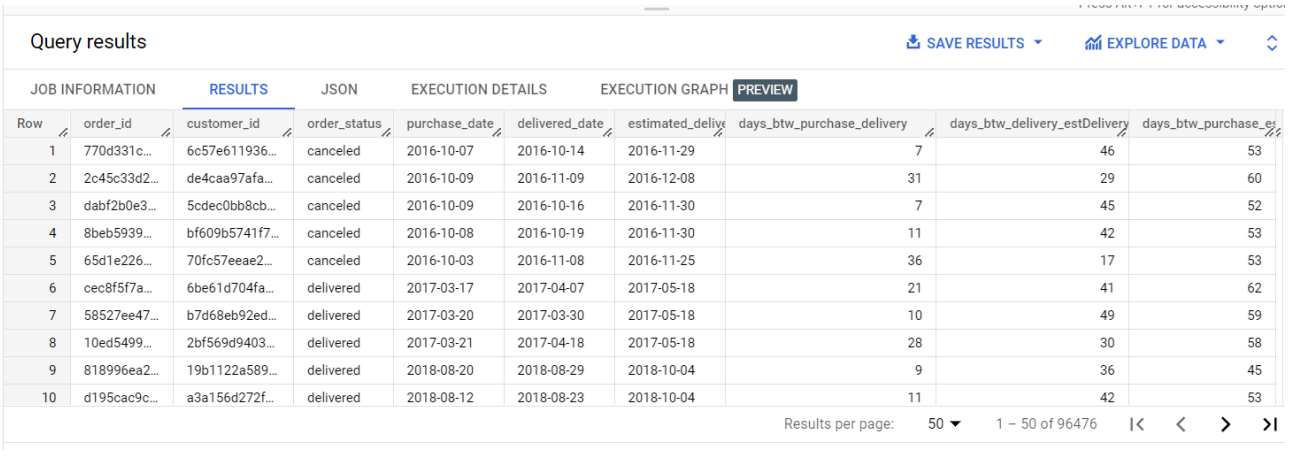
5.1 Calculate the days between purchasing, delivering and estimated delivery.

SQL Query:

```
WITH ordersData AS
```

```
(SELECT

  order_id,

  customer_id,

  order_status,

  DATE(order_purchase_timestamp) AS purchase_date,

  DATE(order_delivered_customer_date) AS delivered_date,

  DATE(order_estimated_delivery_date) AS 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)

SELECT

  order_id,

  customer_id,

  order_status,

  purchase_date,

  delivered_date,

  estimated_delivery,

  ABS(DATE_DIFF(purchase_date,delivered_date,DAY)) AS days_btw_purchase_delivery,

  ABS(DATE_DIFF(delivered_date, estimated_delivery, DAY)) AS
days_btw_delivery_estDelivery,

  ABS(DATE_DIFF(purchase_date,estimated_delivery, DAY)) AS
days_btw_purchase_estDelivery

FROM ordersData;
```

Result:



Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH PREVIEW

Row	order_id	customer_id	order_status	purchase_date	delivered_date	estimated_delive	days_btw_purchase_delivery	days_btw_delivery_estDelivery	days_btw_purchase_es
1	770d331c...	6c57e611936...	canceled	2016-10-07	2016-10-14	2016-11-29	7	46	53
2	2c45c33d2...	de4caa97afa...	canceled	2016-10-09	2016-11-09	2016-12-08	31	29	60
3	dabf2b0e3...	5cdec0bb8cb...	canceled	2016-10-09	2016-10-16	2016-11-30	7	45	52
4	8beb5939...	bf609b5741f7...	canceled	2016-10-08	2016-10-19	2016-11-30	11	42	53
5	65d1e226...	70fc57eeae2...	canceled	2016-10-03	2016-11-08	2016-11-25	36	17	53
6	cec8f5f7a...	6be61d704fa...	delivered	2017-03-17	2017-04-07	2017-05-18	21	41	62
7	58527ee47...	b7d68eb92ed...	delivered	2017-03-20	2017-03-30	2017-05-18	10	49	59
8	10ed5499...	2bf569d9403...	delivered	2017-03-21	2017-04-18	2017-05-18	28	30	58
9	818996ea2...	19b1122a589...	delivered	2018-08-20	2018-08-29	2018-10-04	9	36	45
10	d195cac9c...	a3a156d272f...	delivered	2018-08-12	2018-08-23	2018-10-04	11	42	53

Results per page: 50 | 1 – 50 of 96476

5.2 Find time_to_delivery & diff_estimated_delivery. Formula for the same given below:

time_to_delivery = order_purchase_timestamp-order_delivered_customer_date

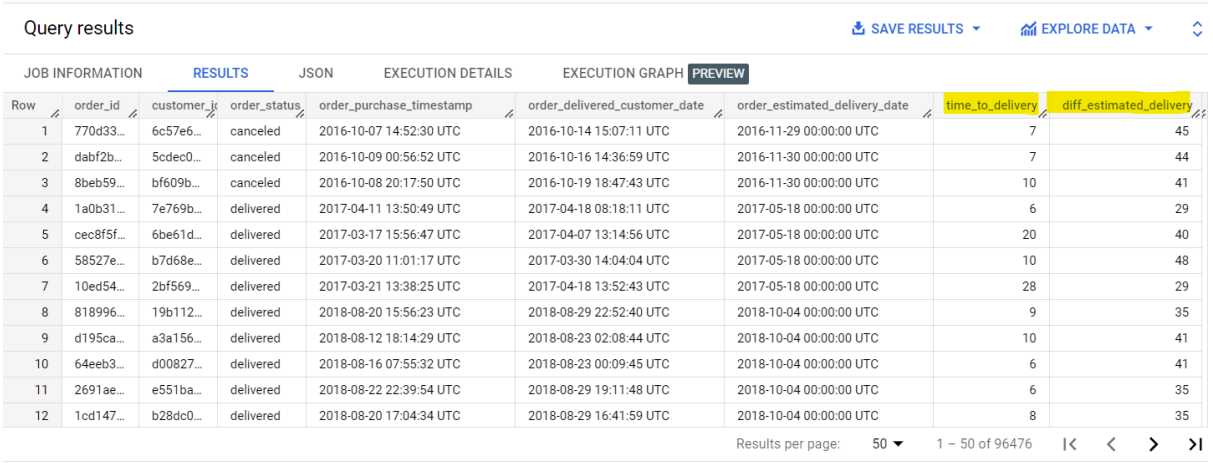
diff_estimated_delivery = order_estimated_delivery_date - order_delivered_customer_date.

SQL Query:

```
WITH ordersData AS

(SELECT

  order_id,

  customer_id,

  order_status,

  order_purchase_timestamp,

  order_delivered_customer_date,

  order_estimated_delivery_date

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)



SELECT

  order_id,

  customer_id,

  order_status,

  order_purchase_timestamp,

  order_delivered_customer_date,

  order_estimated_delivery_date,

  ABS(DATETIME_DIFF(order_purchase_timestamp,order_delivered_customer_date,DAY)) AS time_to_delivery,
```

```
  ABS(DATETIME_DIFF(order_estimated_delivery_date, order_delivered_customer_date,
DAY)) AS diff_estimated_delivery

FROM ordersData;
```

Result:



Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH PREVIEW

Row	order_id	customer_id	order_status	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	time_to_delivery	diff_estimated_delivery
1	770d33...	6c57e6...	canceled	2016-10-07 14:52:30 UTC	2016-10-14 15:07:11 UTC	2016-11-29 00:00:00 UTC	7	45
2	dabf2b...	5cdec0...	canceled	2016-10-09 00:56:52 UTC	2016-10-16 14:36:59 UTC	2016-11-30 00:00:00 UTC	7	44
3	8beb59...	bf609b...	canceled	2016-10-08 20:17:50 UTC	2016-10-19 18:47:43 UTC	2016-11-30 00:00:00 UTC	10	41
4	1a0b31...	7e769b...	delivered	2017-04-11 13:50:49 UTC	2017-04-18 08:18:11 UTC	2017-05-18 00:00:00 UTC	6	29
5	cec8f5f...	6be61d...	delivered	2017-03-17 15:56:47 UTC	2017-04-07 13:14:56 UTC	2017-05-18 00:00:00 UTC	20	40
6	58527e...	b7d68e...	delivered	2017-03-20 11:01:17 UTC	2017-03-30 14:04:04 UTC	2017-05-18 00:00:00 UTC	10	48
7	10ed54...	2bf569...	delivered	2017-03-21 13:38:25 UTC	2017-04-18 13:52:43 UTC	2017-05-18 00:00:00 UTC	28	29
8	818996...	19b112...	delivered	2018-08-20 15:56:23 UTC	2018-08-29 22:52:40 UTC	2018-10-04 00:00:00 UTC	9	35
9	d195ca...	a3a156...	delivered	2018-08-12 18:14:29 UTC	2018-08-23 02:08:44 UTC	2018-10-04 00:00:00 UTC	10	41
10	64eeb3...	d00827...	delivered	2018-08-16 07:55:32 UTC	2018-08-23 00:09:45 UTC	2018-10-04 00:00:00 UTC	6	41
11	2691ae...	e551ba...	delivered	2018-08-22 22:39:54 UTC	2018-08-29 19:11:48 UTC	2018-10-04 00:00:00 UTC	6	35
12	1cd147...	b28dc0...	delivered	2018-08-20 17:04:34 UTC	2018-08-29 16:41:59 UTC	2018-10-04 00:00:00 UTC	8	35

Results per page: 50 | 1 – 50 of 96476

5.3 Group the data by State and get the Mean freight value, time_to_delivery and estimated_diff_delivery

1. **Top 5 states with Highest average freight value - sort in DESC limit 5**

SQL Query:

```
WITH ordersData AS

(SELECT

  order_id,

  customer_id,

  order_status,

  order_purchase_timestamp,

  order_delivered_customer_date,

  order_estimated_delivery_date

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),

ordersDiffData AS

(SELECT
```

```
  order_id,

  customer_id,

  order_status,

  order_purchase_timestamp,

  order_delivered_customer_date,

  order_estimated_delivery_date,

  ABS(DATETIME_DIFF(order_purchase_timestamp,order_delivered_customer_date,DAY)) AS
time_to_delivery,

  ABS(DATETIME_DIFF(order_estimated_delivery_date, order_delivered_customer_date,
DAY)) AS diff_estimated_delivery

FROM ordersData)

SELECT

  cust.customer_state,

  ROUND(AVG(ord_items.freight_value),2) AS avg_freightVal

FROM `target.customers` AS cust JOIN ordersDiffData AS ord ON cust.customer_id =
ord.customer_id

JOIN `target.order_items` AS ord_items ON ord.order_id = ord_items.order_id

GROUP BY cust.customer_state

ORDER BY avg_freightVal DESC LIMIT 5;
```

Result:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUT

Row	customer_state	avg_freightVal
1	PB	43.09
2	RR	43.09
3	RO	41.33
4	AC	40.05
5	PI	39.12

2. Top 5 states with Lowest average freight value - sort in ASC limit 5

SQL Query:

```
WITH ordersData AS

(SELECT

  order_id,

  customer_id,

  order_status,

  order_purchase_timestamp,

  order_delivered_customer_date,

  order_estimated_delivery_date

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),

ordersDiffData AS

(SELECT

  order_id,

  customer_id,

  order_status,

  order_purchase_timestamp,

  order_delivered_customer_date,

  order_estimated_delivery_date,

  ABS(DATETIME_DIFF(order_purchase_timestamp,order_delivered_customer_date,DAY)) AS
time_to_delivery,

  ABS(DATETIME_DIFF(order_estimated_delivery_date, order_delivered_customer_date,
DAY)) AS diff_estimated_delivery

FROM ordersData)


SELECT

  cust.customer_state,

  ROUND(AVG(ord_items.freight_value),2) AS avg_freightVal
```

```
FROM `target.customers` AS cust JOIN ordersDiffData AS ord ON cust.customer_id =
ord.customer_id

JOIN `target.order_items` AS ord_items ON ord.order_id = ord_items.order_id

GROUP BY cust.customer_state

ORDER BY avg_freightVal ASC LIMIT 5;
```

Results:

Query results

JOB INFORMATION | RESULTS | JSON

Row	customer_state	avg_freightVal
1	SP	15.11
2	PR	20.47
3	MG	20.63
4	RJ	20.91
5	DF	21.07

3. Top 5 states with Highest average time to delivery

SQL Query:

```
WITH ordersData AS

(SELECT

  order_id,

  customer_id,

  order_status,

  order_purchase_timestamp,

  order_delivered_customer_date,

  order_estimated_delivery_date

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),

ordersDiffData AS

(SELECT
```

```
  order_id,

  customer_id,

  order_status,

  order_purchase_timestamp,

  order_delivered_customer_date,

  order_estimated_delivery_date,

  ABS(DATETIME_DIFF(order_purchase_timestamp,order_delivered_customer_date,DAY)) AS
time_to_delivery,

  ABS(DATETIME_DIFF(order_estimated_delivery_date, order_delivered_customer_date,
DAY)) AS diff_estimated_delivery

FROM ordersData)



SELECT

  cust.customer_state,

  ROUND(AVG(ord.time_to_delivery),2) AS avg_time_to_delivery

FROM `target.customers` AS cust JOIN ordersDiffData AS ord ON cust.customer_id =
ord.customer_id

JOIN `target.order_items` AS ord_items ON ord.order_id = ord_items.order_id

GROUP BY cust.customer_state

ORDER BY avg_time_to_delivery DESC LIMIT 5;
```

Result:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION D

Row	customer_state	avg_time_to_delivery
1	RR	27.83
2	AP	27.75
3	AM	25.96
4	AL	23.99
5	PA	23.3

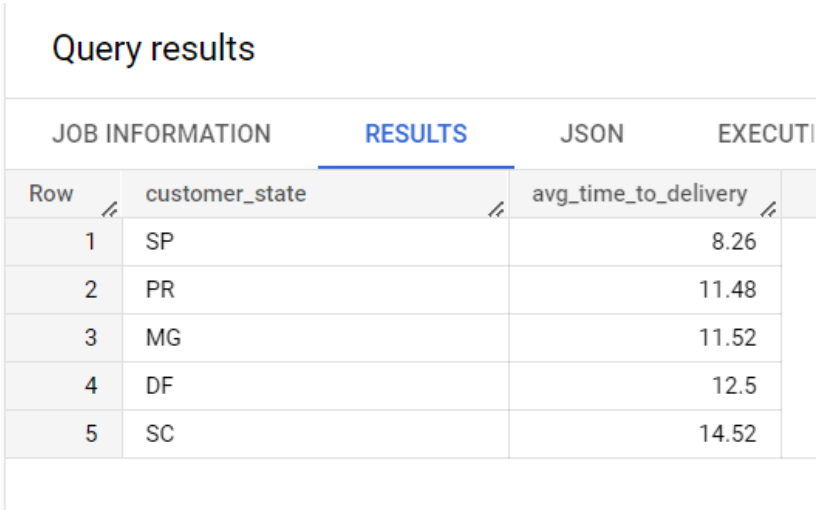
4. Top 5 states with Lowest average time to delivery

SQL Query:

```
WITH ordersData AS

(SELECT

  order_id,

  customer_id,

  order_status,

  order_purchase_timestamp,

  order_delivered_customer_date,

  order_estimated_delivery_date

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),

ordersDiffData AS

(SELECT

  order_id,

  customer_id,

  order_status,

  order_purchase_timestamp,

  order_delivered_customer_date,

  order_estimated_delivery_date,

  ABS(DATETIME_DIFF(order_purchase_timestamp,order_delivered_customer_date,DAY)) AS
time_to_delivery,

  ABS(DATETIME_DIFF(order_estimated_delivery_date, order_delivered_customer_date,
DAY)) AS diff_estimated_delivery

FROM ordersData)


SELECT

  cust.customer_state,

  ROUND(AVG(ord.time_to_delivery),2) AS avg_time_to_delivery
```

```
FROM `target.customers` AS cust JOIN ordersDiffData AS ord ON cust.customer_id =
ord.customer_id

JOIN `target.order_items` AS ord_items ON ord.order_id = ord_items.order_id

GROUP BY cust.customer_state

ORDER BY avg_time_to_delivery ASC LIMIT 5;
```

Result:



Row	customer_state	avg_time_to_delivery
1	SP	8.26
2	PR	11.48
3	MG	11.52
4	DF	12.5
5	SC	14.52

5. Top 5 states where delivery is really fast compared to estimated date

SQl Query:

```
WITH ordersData AS

(SELECT

  order_id,

  customer_id,

  order_status,

  order_purchase_timestamp,

  order_delivered_customer_date,

  order_estimated_delivery_date

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),

ordersDiffData AS

(SELECT

  order_id,
```

```
  customer_id,

  order_status,

  order_purchase_timestamp,

  order_delivered_customer_date,

  order_estimated_delivery_date,

  ABS(DATETIME_DIFF(order_purchase_timestamp,order_delivered_customer_date,DAY)) AS
time_to_delivery,

  ABS(DATETIME_DIFF(order_estimated_delivery_date, order_delivered_customer_date,
DAY)) AS diff_estimated_delivery

FROM ordersData)



SELECT

  cust.customer_state,

  ROUND(AVG(ord.time_to_delivery),2) AS avg_time_to_delivery,

  ROUND(AVG(ord.diff_estimated_delivery),2) AS avg_diff_estDelivery

FROM `target.customers` AS cust JOIN ordersDiffData AS ord ON cust.customer_id =
ord.customer_id

JOIN `target.order_items` AS ord_items ON ord.order_id = ord_items.order_id

GROUP BY cust.customer_state

HAVING ROUND(AVG(ord.time_to_delivery),2) <
ROUND(AVG(ord.diff_estimated_delivery),2)

ORDER BY avg_time_to_delivery ASC LIMIT 5;
```

Result:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXEC

Row	customer_state	avg_time_to_delivery	avg_diff_estDelivery
1	SP	8.26	10.99
2	PR	11.48	13.16
3	MG	11.52	13.13
4	RO	19.28	19.6
5	AC	20.33	21.24

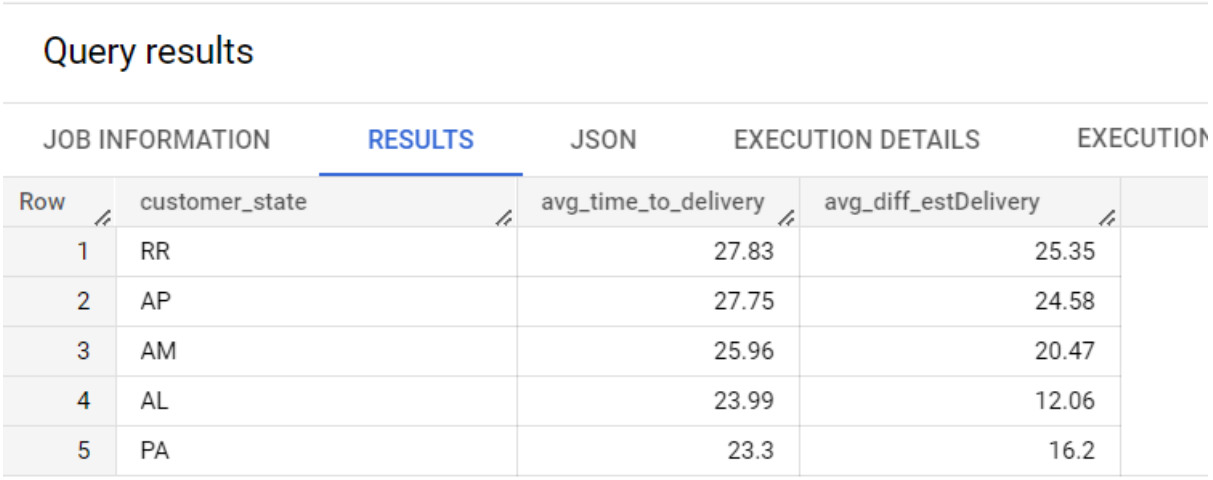
6. Top 5 states where delivery is not so fast compared to estimated date

SQL Query:

```
WITH ordersData AS

(SELECT

  order_id,

  customer_id,

  order_status,

  order_purchase_timestamp,

  order_delivered_customer_date,

  order_estimated_delivery_date

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),

ordersDiffData AS

(SELECT

  order_id,

  customer_id,

  order_status,

  order_purchase_timestamp,

  order_delivered_customer_date,

  order_estimated_delivery_date,

  ABS(DATETIME_DIFF(order_purchase_timestamp,order_delivered_customer_date,DAY)) AS
time_to_delivery,

  ABS(DATETIME_DIFF(order_estimated_delivery_date, order_delivered_customer_date,
DAY)) AS diff_estimated_delivery

FROM ordersData)



SELECT

  cust.customer_state,

  ROUND(AVG(ord.time_to_delivery),2) AS avg_time_to_delivery,

  ROUND(AVG(ord.diff_estimated_delivery),2) AS avg_diff_estDelivery
```

```
FROM `target.customers` AS cust JOIN ordersDiffData AS ord ON cust.customer_id =
ord.customer_id

JOIN `target.order_items` AS ord_items ON ord.order_id = ord_items.order_id

GROUP BY cust.customer_state

HAVING ROUND(AVG(ord.time_to_delivery),2) >
ROUND(AVG(ord.diff_estimated_delivery),2)

ORDER BY avg_time_to_delivery DESC LIMIT 5;
```

Result:



Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION

Row	customer_state	avg_time_to_delivery	avg_diff_estDelivery
1	RR	27.83	25.35
2	AP	27.75	24.58
3	AM	25.96	20.47
4	AL	23.99	12.06
5	PA	23.3	16.2

Payment type analysis:

6.1 Month over Month count of orders for different payment types

SQL Query:

```
WITH joint_payments AS

(SELECT

  ord.order_id,

  EXTRACT(MONTH FROM DATE(ord.order_purchase_timestamp)) AS Month,

  pays.payment_type AS Payments_type

FROM `target.orders` AS ord JOIN `target.payments` AS pays ON ord.order_id =
pays.order_id

WHERE ord.order_approved_at IS NOT NULL AND ord.order_delivered_carrier_date IS NOT
NULL AND ord.order_delivered_customer_date IS NOT NULL)
```

```sql
SELECT
  Month,
  Payments_type,
  COUNT(order_id) AS Total_no_of_orders
FROM joint_payments
GROUP BY Month,Payments_type
ORDER BY Month,Payments_type;
```

Result:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	Month	Payments_type	Total_no_of_orders
1	1	UPI	1659
2	1	credit_card	5910
3	1	debit_card	118
4	1	voucher	461
5	2	UPI	1653
6	2	credit_card	6372
7	2	debit_card	81
8	2	voucher	408
9	3	UPI	1881
10	3	credit_card	7434

6.2 Count of orders based on the no. of payment installments:

SQL Query:

```sql
SELECT
  payment_installments,
  COUNT(order_id) AS total_orders
FROM `target.payments`
GROUP BY payment_installments
ORDER BY payment_installments;
```

Result:

Query results

JOB INFORMATION | RESULTS | JSON

Row	payment_installments	total_orders
1	0	2
2	1	52546
3	2	12413
4	3	10461
5	4	7098
6	5	5239
7	6	3920
8	7	1626
9	8	4268
10	9	644

Tableau Visualisation: