

# BUSINESS CASE USING SQL

This case study is about one of the world's most recognized brands and one of America's leading retailers. This brand makes itself a preferred shopping destination by offering outstanding value, inspiration, innovation, and an exceptional guest experience that no other retailer can deliver.

This business case has information of 100k orders from 2016 to 2018 made by the brand in Brazil. Its features allow viewing an order from multiple dimensions: from order status, price, payment and freight performance to customer location, product attributes and finally reviews written by customers.

1. Import the dataset and do the usual exploratory analysis steps like checking the structure & characteristics of the dataset.
  - 1.1 Data type of columns in a table
  - 1.2 Time period for which the data is given.
  - 1.3 Cities and States of customers ordered during the given period.

## Solution:

We can know about the Data Type of the columns from Table SCHEMA. The following table SCHEMA shows that the whole dataset contains data types such as **STRING, INTEGER, FLOAT, TIMESTAMP**, etc.

<input type="checkbox"/> Field name	Type	<input type="checkbox"/> Field name	Type	<input type="checkbox"/> Field name	Type
<input type="checkbox"/> <a href="#">customer_id</a>	STRING	<input type="checkbox"/> <a href="#">geolocation_zip_code_prefix</a>	INTEGER	<input type="checkbox"/> <a href="#">order_id</a>	STRING
<input type="checkbox"/> <a href="#">customer_unique_id</a>	STRING	<input type="checkbox"/> <a href="#">geolocation_lat</a>	FLOAT	<input type="checkbox"/> <a href="#">order_item_id</a>	INTEGER
<input type="checkbox"/> <a href="#">customer_zip_code_prefix</a>	INTEGER	<input type="checkbox"/> <a href="#">geolocation_lng</a>	FLOAT	<input type="checkbox"/> <a href="#">product_id</a>	STRING
<input type="checkbox"/> <a href="#">customer_city</a>	STRING	<input type="checkbox"/> <a href="#">geolocation_city</a>	STRING	<input type="checkbox"/> <a href="#">seller_id</a>	STRING
<input type="checkbox"/> <a href="#">customer_state</a>	STRING	<input type="checkbox"/> <a href="#">geolocation_state</a>	STRING	<input type="checkbox"/> <a href="#">shipping_limit_date</a>	TIMESTAMP
				<input type="checkbox"/> <a href="#">price</a>	FLOAT
				<input type="checkbox"/> <a href="#">freight_value</a>	FLOAT

  

<input type="checkbox"/> Field name	Type	<input type="checkbox"/> Field name	Type	<input type="checkbox"/> Field name	Type
<input type="checkbox"/> <a href="#">order_id</a>	STRING	<input type="checkbox"/> <a href="#">order_id</a>	STRING	<input type="checkbox"/> <a href="#">product_id</a>	STRING
<input type="checkbox"/> <a href="#">customer_id</a>	STRING	<input type="checkbox"/> <a href="#">payment_sequential</a>	INTEGER	<input type="checkbox"/> <a href="#">product_category</a>	STRING
<input type="checkbox"/> <a href="#">order_status</a>	STRING	<input type="checkbox"/> <a href="#">payment_type</a>	STRING	<input type="checkbox"/> <a href="#">product_name_length</a>	INTEGER
<input type="checkbox"/> <a href="#">order_purchase_timestamp</a>	TIMESTAMP	<input type="checkbox"/> <a href="#">payment_installments</a>	INTEGER	<input type="checkbox"/> <a href="#">product_description_length</a>	INTEGER
<input type="checkbox"/> <a href="#">order_approved_at</a>	TIMESTAMP	<input type="checkbox"/> <a href="#">payment_value</a>	FLOAT	<input type="checkbox"/> <a href="#">product_photos_qty</a>	INTEGER
<input type="checkbox"/> <a href="#">order_delivered_carrier_date</a>	TIMESTAMP			<input type="checkbox"/> <a href="#">product_weight_g</a>	INTEGER
<input type="checkbox"/> <a href="#">order_delivered_customer_date</a>	TIMESTAMP			<input type="checkbox"/> <a href="#">product_length_cm</a>	INTEGER
<input type="checkbox"/> <a href="#">order_estimated_delivery_date</a>	TIMESTAMP			<input type="checkbox"/> <a href="#">product_height_cm</a>	INTEGER
				<input type="checkbox"/> <a href="#">product_width_cm</a>	INTEGER

  

<input type="checkbox"/> Field name	Type	<input type="checkbox"/> Field name	Type
<input type="checkbox"/> <a href="#">seller_id</a>	STRING	<input type="checkbox"/> <a href="#">review_id</a>	STRING
<input type="checkbox"/> <a href="#">seller_zip_code_prefix</a>	INTEGER	<input type="checkbox"/> <a href="#">order_id</a>	STRING
<input type="checkbox"/> <a href="#">seller_city</a>	STRING	<input type="checkbox"/> <a href="#">review_score</a>	INTEGER
<input type="checkbox"/> <a href="#">seller_state</a>	STRING	<input type="checkbox"/> <a href="#">review_comment_title</a>	STRING
		<input type="checkbox"/> <a href="#">review_creation_date</a>	TIMESTAMP
		<input type="checkbox"/> <a href="#">review_answer_timestamp</a>	TIMESTAMP

The time period for which the data is given -

From `min(order_purchase_timestamp)` and `max(order_purchase_timestamp)`, we can get the total period for which the data is given.

The data is given from **4<sup>th</sup> September 2016 to 17<sup>th</sup> October 2018**.

The screenshot shows the Google Cloud BigQuery interface. On the left, the Explorer pane displays the 'target' dataset with tables like customers, geolocation, order\_items, order\_reviews, orders, payments, products, and sellers. The main editor shows a query titled 'Untitled' with the following SQL:

```
1 SELECT min(order_purchase_timestamp) as start_timestamp,
2 max(order_purchase_timestamp) as end_timestamp
3 FROM `target.orders`
```

The query has been executed successfully, as indicated by the 'Query completed.' status. Below the query editor, the 'Query results' section is visible, showing a table with two columns: 'start\_timestamp' and 'end\_timestamp'.

Row	start_timestamp	end_timestamp
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

Cities and States of customers ordered during the given period –

To get the cities and states of the customers who were active during the time period, let's join the customer and order table over the customer\_id and get the distinct customer\_city and customer\_state from which we got the orders.

The screenshot shows the Google Cloud BigQuery interface with a new query titled 'Untitled 3'. The query joins the 'target.customers' and 'target.orders' tables on 'customer\_id' to retrieve distinct cities and states. The SQL is as follows:

```
1 SELECT DISTINCT c.customer_city, c.customer_state
2 FROM `target.customers` c
3 INNER JOIN `target.orders` o ON c.customer_id = o.customer_id
```

The query results are displayed in a table with two columns: 'customer\_city' and 'customer\_state'.

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

At the bottom right of the results section, it indicates 'Results per page: 50'.

## INSIGHT:

- The dataset consists of the data types such as **STRING, INTEGER, FLOAT, and TIMESTAMP**.
- We received the dataset from the period **September 2016 to October 2018**.

## 2. In-depth Exploration:

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

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

0-7: Dawn

7-12: Morning

12-18: Afternoon

18-23: Night

Solution:

To see if there is a growing trend or not, we need to see if there is any increase in the number of orders over the years.

orders

\*Business case: 2: 1

+

Business case: 2: 1

RUN

SAVE

SHARE

```
1 -- Total number of orders per year
2 SELECT EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
3 COUNT(order_id) AS no_of_orders
4 FROM `target.orders`
5 GROUP BY 1
6 ORDER BY 1
7
```

Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS
Row	year	no_of_orders		
1	2016	329		
2	2017	45101		
3	2018	54011		

orders

\*Business case: 2: 1

+

Business case: 2: 1

RUN

SAVE

SHARE

```

1  -- Total number of orders in a month
2  SELECT a.year, a.month, a.no_of_orders,
3  SUM(a.no_of_orders) OVER (partition by a.year) AS orders_per_year
4  FROM
5  (SELECT EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
6  EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
7  COUNT(order_id) AS no_of_orders
8  FROM `target.orders`
9  GROUP BY 1, 2) AS a
10 ORDER BY 1 ASC, 3 DESC

```

### Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS
Row	year	month	no_of_orders	orders_per_year
14	2017	2	1780	45101
15	2017	1	800	45101
16	2018	1	7269	54011

orders

\*Business case: 2: 1

+

### Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS
Row	year	month	no_of_orders	orders_per_year
1	2016	10	324	329
2	2016	9	4	329
3	2016	12	1	329
4	2017	11	7544	45101
5	2017	12	5673	45101
6	2017	10	4631	45101
7	2017	8	4331	45101
8	2017	9	4285	45101
9	2017	7	4026	45101
10	2017	5	3700	45101
11	2017	6	3245	45101
12	2017	3	2682	45101

### Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION DET
Row	year	month	no_of_orders	orders_per_year
14	2017	2	1780	45101
15	2017	1	800	45101
16	2018	1	7269	54011
17	2018	3	7211	54011
18	2018	4	6939	54011
19	2018	5	6873	54011
20	2018	2	6728	54011
21	2018	8	6512	54011
22	2018	7	6292	54011
23	2018	6	6167	54011
24	2018	9	16	54011
25	2018	10	4	54011

### Insight:

- As we do not have sufficient data for the year 2016 (data of only 3 months have been given), we cannot infer that there is growth in e-commerce, but from the overall growth of 2017 to 2018, we can say that there is growth in e-commerce in Brazil.
- We can see from the 2<sup>nd</sup> query that the highest number of orders were towards the end of the year 2017 and the start of the year 2018, i.e., during the winter season.
- But in 2018, the number of orders was decreasing over the year.

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

0-7: Dawn

7-12: Morning

12-18: Afternoon

18-23: Night

Business case: 2: 1

RUN

SAVE

SHARE

SCHEDULE

MORE

```
14 SELECT
15   CASE
16   WHEN EXTRACT(TIME FROM order_purchase_timestamp) between "00:00:00" AND "07:00:00"
17   THEN "Dawn"
18   WHEN EXTRACT(TIME FROM order_purchase_timestamp) between "07:00:01" AND "12:00:00"
19   THEN "Morning"
20   WHEN EXTRACT(TIME FROM order_purchase_timestamp) between "12:00:01" AND "18:00:00"
21   THEN "Afternoon"
22   ELSE "Night"
23   END AS time_of_day,
24   count(DISTINCT order_id) AS no_of_orders
25 FROM `target.orders`
26 GROUP BY 1
27 ORDER BY 2 DESC
```

Pre:

Query results

SAVE RESULTS

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Row	time_of_day	no_of_orders				
1	Afternoon	38365				
2	Night	34096				

## Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION
Row	time_of_day	no_of_orders		
1	Afternoon	38365		
2	Night	34096		
3	Morning	21738		
4	Dawn	5242		

PERSONAL HISTORY

PROJECT HISTORY

### Insight:

We can clearly see from the result that the customers tend to buy more during the afternoon time.

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

3.1 Get month-on-month orders by state.

3.2 Distribution of customers across the states in Brazil.

Business case: 2: 1

RUN

SAVE

SHARE

SCHEDULE

```
52 -- Distribution of customers across states in Brazil
53 SELECT customer_state, COUNT(DISTINCT customer_id) AS No_of_customers
54 FROM target.customers
55 GROUP BY 1
56 ORDER BY 2 DESC
57
```

### Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	EXECUTION
Row	customer_state	No_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			



Query results		
JOB INFORMATION		
RESULTS		
JSON		
EXECUTION DETAILS		
Row	customer_state	No_of_customers
14	MT	907
15	MA	747
16	MS	715
17	PB	536
18	PI	495
19	RN	485
20	AL	413
21	SE	350
22	TO	280
23	RO	253
24	AM	148
25	AC	81
26	AP	68
27	RR	46

- 4 Impact on the Economy: Analyze the money movement by e-commerce by looking at order prices, freight, and others.
  - 4.1 Get a % increase in the cost of orders from 2017 to 2018 (include months between Jan to Aug only) - You can use the "payment\_value" column in the payments table.
  - 4.2 Mean & Sum of price and freight value by customer states.

```

Business case: 2: 1  [RUN] [SAVE] [SHARE] [SCHEDULE] [MORE]
58 -- 4.1 Get a % increase in the cost of orders from 2017 to 2018 (include months between Jan to Aug only) - Y
   payments table.
59 SELECT
60 b2.month, ROUND(b2.payment_value_2018, 2) cost_of_order_2018,
61 ROUND(b2.payment_value_2017, 2) cost_of_order_2017,
62 ROUND(((b2.payment_value_2018 - b2.payment_value_2017) / payment_value_2017) * 100, 2) AS growth_in_cost
63 FROM
64 (SELECT
65 b1.month, b1.payment_value AS payment_value_2018,
66 LAG(b1.payment_value) OVER(partition by month ORDER BY year) AS payment_value_2017
67 FROM
68 (SELECT
69 EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
70 EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
71 AVG(p.payment_value) payment_value
72 FROM target.orders o
73 INNER JOIN target.payments p
74 ON o.order_id = p.order_id
75 GROUP BY 1, 2
76 ORDER BY 1, 2) AS b1
77 WHERE b1.year IN (2017, 2018) AND b1.month IN (1, 2, 3, 4, 5, 6, 7, 8)
78 ORDER BY 2) AS b2
79 WHERE NOT b2.payment_value_2017 IS null
80 ORDER BY 1

```



Query results						
<div> JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH PREVIEW </div>						
Row	month	cost_of_order_2018	cost_of_order_2017	growth_in_cost		
1	1	147.43	162.93	-9.51		
2	2	142.76	154.78	-7.76		
3	3	154.37	158.57	-2.65		
4	4	161.02	162.5	-0.91		
5	5	161.74	150.33	7.58		
6	6	159.51	148.8	7.2		
7	7	163.91	137.22	19.45		
8	8	152.65	148.22	2.99		

#### Insight:

- For the first four months of 2018, the cost of orders decreased, but after that, it increased.

Mean & Sum of price and freight value by customer states -

<div> Business case: 2: 1 <div> RUN SAVE SHARE SCHEDULE MORE </div> </div>						
<pre> 83 -- 4.2 Mean &amp; Sum of price and freight value by a customer state. 84 85 SELECT c.customer_state, 86        ROUND(SUM(oi.price), 2) AS total_price, 87        ROUND(AVG(oi.price), 2) AS avg_price, 88        ROUND(SUM(oi.freight_value), 2) AS total_freight_value, 89        ROUND(AVG(oi.freight_value), 2) AS avg_freight_value 90 FROM `target.customers` c 91 JOIN `target.orders` o 92 ON c.customer_id = o.customer_id 93 JOIN `target.order_items` oi 94 ON o.order_id = oi.order_id 95 GROUP BY 1 </pre>						
Query results						
<div> JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH PREVIEW </div>						
Row	customer_state	total_price	avg_price	total_freight_value	avg_freight_value	
1	RN	83034.98	156.97	18860.1	35.65	
2	CE	227254.71	153.76	48351.59	32.71	
3	RS	750304.02	120.34	135522.74	21.74	

## Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Row	customer_state ▼	total_price ▼	avg_price ▼	total_freight_value	avg_freight_value ▼	
4	SC	520553.34	124.65	89660.26	21.47	
5	SP	5202955.05	109.65	718723.07	15.15	
6	MG	1585308.03	120.75	270853.46	20.63	
7	BA	511349.99	134.6	100156.68	26.36	
8	RJ	1824092.67	125.12	305589.31	20.96	
9	GO	294591.95	126.27	53114.98	22.77	
10	MA	119648.22	145.2	31523.77	38.26	
11	PE	262788.03	145.51	59449.66	32.92	
12	PB	115268.08	191.48	25719.73	42.72	
13	ES	275037.31	121.91	49764.6	22.06	
14	PR	683083.76	119.0	117851.68	20.53	
15	RO	46140.64	165.97	11417.38	41.07	
16	MS	116812.64	142.63	19144.03	23.37	
17	PA	178947.81	165.69	38699.3	35.83	

### 5. Analysis of sales, freight, and delivery time

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

Business case: 2: 1 RUN SAVE SHARE SCHEDULE MORE

```

98 -- 5.1 Calculate the days between purchasing, delivery, and estimated delivery
99 SELECT
100 order_id,
101 DATE_DIFF(DATE(order_estimated_delivery_date), DATE(order_purchase_timestamp), DAY) AS estimated_delivery_days,
102 DATE_DIFF(DATE(order_delivered_customer_date), DATE(order_purchase_timestamp), DAY) AS actual_delivery_days
103 FROM `target.orders`
104 ORDER BY 3 DESC

```

## Query results

[SAVE](#)

JOB INFORMATION					RESULTS	JSON	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Row	order_id	estimated_delivery_c	actual_delivery_days						
1	ca07593549f1816d26a572e06...	29	210						
2	1b3190b2dfa9d789e1f14c05b...	20	208						
3	440d0d17af552815d15a9e41a...	31	196						
4	285ab9426d6982034523a855f...	29	195						
5	2fb597c2f772eca01b1f5c561b...	40	195						
6	0f4519c5f1c541ddec9f21b3bd...	33	194						
7	47b40429ed8cce3aee9199792...	16	191						

## INSIGHT:

- For some orders there is actually delivery time was much more than the estimated delivery time

## 5.2 Find time\_to\_delivery & diff\_estimated\_delivery.

The formula for the same is given below:

time\_to\_delivery = order\_delivered\_customer\_date-order\_purchase\_timestamp

diff\_estimated\_delivery = order\_estimated\_delivery\_date-order\_delivered\_customer\_date

Business case: 2: 1 RUN SAVE SHARE SCHEDULE MORE

```

110
111 SELECT order_id,
112 DATE_DIFF(DATE(order_delivered_customer_date), DATE(order_purchase_timestamp), DAY) AS time_to_delivery,
113 DATE_DIFF(DATE(order_estimated_delivery_date), DATE(order_delivered_customer_date), DAY) AS diff_estimated_delivery
114 FROM `target.orders`

```

## Query results

[SAVE](#)

JOB INFORMATION					RESULTS	JSON	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Row	order_id	time_to_delivery	diff_estimated_delivery						
1	1950d777989f6a877539f5379...	30	-12						
2	2c45c33d2f9cb8ff8b1c86cc28...	31	29						
3	65d1e226dfaeb8cdc42f66542...	36	17						
4	635c894d068ac37e6e03dc54e...	31	2						
5	3b97562c3aee8bdedcb5c2e45...	33	1						
6	68f47f50f04c4cb6774570cfde...	30	2						
7	276e9ec344d3bf029ff83a161c...	44	-4						
8	54e1a3c2b97fb0809da548a59...	41	-4						

5.3 Group data by state, take a mean of freight\_value, time\_to\_delivery, and diff\_estimated\_delivery.

Business case: 2: 1 RUN SAVE SHARE SCHEDULE MORE

```
116 -- 5.3 Group data by state, take a mean of freight_value, time_to_delivery, and diff_estimated_delivery.
117 SELECT
118   c.customer_state,
119   ROUND(avg(oi.freight_value), 2) as avg_freight_value,
120   ROUND(avg(DATE_DIFF(DATE(o.order_delivered_customer_date), DATE(o.order_purchase_timestamp), DAY)), 2) as avg_time_to_delivery,
121   ROUND(avg(DATE_DIFF(DATE(o.order_estimated_delivery_date), DATE(o.order_delivered_customer_date), DAY)), 2) as avg_diff_estimated_delivery
122 FROM `target.customers` c
123 JOIN `target.orders` o ON c.customer_id = o.customer_id
124 JOIN `target.order_items` oi ON o.order_id = oi.order_id
125 GROUP BY 1
```

Query results SAVE RESULTS EXP

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Row	customer_state	avg_freight_value	avg_time_to_delivery	avg_diff_estimated_c		
1	MT	28.17	17.91	14.57		
2	MA	38.26	21.59	9.91		
3	AL	35.84	24.45	8.74		
4	SP	15.15	8.66	11.21		
5	MG	20.63	11.92	13.34		

Results per page: 50 1 - 27 of 27

5.4 Sort the data to get the following:

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

States with lowest average freight value -

Business case: 2: 1 RUN SAVE SHARE SCHEDULE MORE

```
128 -- Top 5 states with lowest avg freight value
129 SELECT
130   c.customer_state,
131   ROUND(avg(oi.freight_value), 2) as avg_freight_value,
132   ROUND(avg(DATE_DIFF(DATE(o.order_delivered_customer_date), DATE(o.order_purchase_timestamp), DAY)), 2) as avg_time_to_delivery,
133   ROUND(avg(DATE_DIFF(DATE(o.order_estimated_delivery_date), DATE(o.order_delivered_customer_date), DAY)), 2) as avg_diff_estimated_delivery
134 FROM `target.customers` c
135 JOIN `target.orders` o ON c.customer_id = o.customer_id
136 JOIN `target.order_items` oi ON o.order_id = oi.order_id
137 GROUP BY 1 ORDER BY 2 LIMIT 5
```

Query results SAVE RESULTS EXPL

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Row	customer_state	avg_freight_value	avg_time_to_delivery	avg_diff_estimated_c		
1	SP	15.15	8.66	11.21		
2	PR	20.53	11.89	13.49		
3	MG	20.63	11.92	13.34		
4	RJ	20.96	15.07	12.01		
5	DF	21.04	12.89	12.2		

States with highest average freight value -

Business case: 2: 1				
<pre> 139 -- Top 5 states with highest avg freight value 140 SELECT c.customer_state, 141 ROUND(avg(o1.freight_value), 2) as avg_freight_value, 142 ROUND(avg(DATE_DIFF(DATE(o.order_delivered_customer_date), DATE(o.order_purchase_timestamp), DAY)), 2) as avg_time_to_delivery, 143 ROUND(avg(DATE_DIFF(DATE(o.order_estimated_delivery_date), DATE(o.order_delivered_customer_date), DAY)), 2) as avg_diff_estimated_delivery 144 FROM `target.customers` c 145 JOIN `target.orders` o ON c.customer_id = o.customer_id 146 JOIN `target.order_items` oi ON o.order_id = oi.order_id 147 GROUP BY 1 148 ORDER BY 2 DESC 149 LIMIT 5 150 </pre>				
Query results				
<div> JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH PREVIEW </div>				
Row	customer_state	avg_freight_value	avg_time_to_delivery	avg_diff_estimated_c
1	RR	42.98	28.17	18.33
2	PB	42.72	20.55	13.04
3	RO	41.07	19.66	20.04
4	AC	40.07	20.68	20.98
5	PI	39.15	19.32	11.53

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

States with the highest average time to delivery –

Business case: 2: 1				
<pre> 152 -- Top 5 states with highest average time to delivery 153 SELECT c.customer_state, 154 ROUND(avg(o1.freight_value), 2) as avg_freight_value, 155 ROUND(avg(DATE_DIFF(DATE(o.order_delivered_customer_date), DATE(o.order_purchase_timestamp), DAY)), 2) as avg_time_to_delivery, 156 ROUND(avg(DATE_DIFF(DATE(o.order_estimated_delivery_date), DATE(o.order_delivered_customer_date), DAY)), 2) as avg_diff_estimated_delivery 157 FROM `target.customers` c 158 JOIN `target.orders` o ON c.customer_id = o.customer_id 159 JOIN `target.order_items` oi ON o.order_id = oi.order_id 160 GROUP BY 1 161 ORDER BY 3 DESC 162 LIMIT 5 </pre>				
Query results				
<div> JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH PREVIEW </div>				
Row	customer_state	avg_freight_value	avg_time_to_delivery	avg_diff_estimated_c
1	AP	34.01	28.22	18.4
2	RR	42.98	28.17	18.33
3	AM	33.21	26.34	19.93
4	AL	35.84	24.45	8.74
5	PA	35.83	23.7	14.25

States with the lowest average time to delivery –



Business case: 2: 1 RUN SAVE SHARE SCHEDULE MORE

```

164 -- -- Top 5 states with lowest average time to delivery
165 SELECT c.customer_state,
166 ROUND(avg(o1.freight_value), 2) as avg_freight_value,
167 ROUND(avg(DATE_DIFF(DATE(o.order_delivered_customer_date), DATE(o.order_purchase_timestamp), DAY)), 2) as avg_time_to_delivery,
168 ROUND(avg(DATE_DIFF(DATE(o.order_estimated_delivery_date), DATE(o.order_delivered_customer_date), DAY)), 2) as avg_diff_estimated_delivery
169 FROM `target.customers` c
170 JOIN `target.orders` o ON c.customer_id = o.customer_id
171 JOIN `target.order_items` oi ON o.order_id = oi.order_id
172 GROUP BY 1
173 ORDER BY 3
174 LIMIT 5

```

Press Alt+i

### Query results

SAVE RESULTS EXPL

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Row	customer_state	avg_freight_value	avg_time_to_delivery	avg_diff_estimated_c		
1	SP	15.15	8.66	11.21		
2	PR	20.53	11.89	13.49		
3	MG	20.63	11.92	13.34		
4	DF	21.04	12.89	12.2		
5	SC	21.47	14.95	11.57		

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

States with fast delivery –

Business case: 2: 1 RUN SAVE SHARE SCHEDULE MORE

```

176 -- 5.7 Top 5 states where delivery is really fast/ not so fast compared to the estimated date.
177 -- FAST
178 SELECT c.customer_state,
179 ROUND(avg(o1.freight_value), 2) as avg_freight_value,
180 ROUND(avg(DATE_DIFF(DATE(o.order_delivered_customer_date), DATE(o.order_purchase_timestamp), DAY)), 2) as avg_time_to_delivery,
181 ROUND(avg(DATE_DIFF(DATE(o.order_estimated_delivery_date), DATE(o.order_delivered_customer_date), DAY)), 2) as avg_diff_estimated_delivery
182 FROM `target.customers` c
183 JOIN `target.orders` o ON c.customer_id = o.customer_id
184 JOIN `target.order_items` oi ON o.order_id = oi.order_id
185 GROUP BY 1
186 ORDER BY 4
187 LIMIT 5

```

Press Alt+i

### Query results

SAVE RESULTS EXPL

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Row	customer_state	avg_freight_value	avg_time_to_delivery	avg_diff_estimated_c		
1	AL	35.84	24.45	8.74		
2	MA	38.26	21.59	9.91		
3	SE	36.65	21.42	10.0		
4	ES	22.06	15.59	10.65		
5	BA	26.36	19.19	10.98		

States with slow delivery –

Business case: 2: 1 RUN SAVE SHARE SCHEDULE MORE

```

189 -- SLOW
190 SELECT c.customer_state,
191 ROUND(avg(oi.freight_value), 2) as avg_freight_value,
192 ROUND(avg(DATE_DIFF(DATE(o.order_delivered_customer_date), DATE(o.order_purchase_timestamp), DAY)), 2) as avg_time_to_delivery,
193 ROUND(avg(DATE_DIFF(DATE(o.order_estimated_delivery_date), DATE(o.order_delivered_customer_date), DAY)), 2) as avg_diff_estimated_delivery
194 FROM target.customers c
195 JOIN target.orders o ON c.customer_id = o.customer_id
196 JOIN target.order_items oi ON o.order_id = oi.order_id
197 GROUP BY 1
198 ORDER BY 4 DESC
199 LIMIT 5

```

Query results SAVE RESULTS EXP

Row	customer_state	avg_freight_value	avg_time_to_delivery	avg_diff_estimated_c
1	AC	40.07	20.68	20.98
2	RO	41.07	19.66	20.04
3	AM	33.21	26.34	19.93
4	AP	34.01	28.22	18.4
5	RR	42.98	28.17	18.33

## 6. Payment type analysis:

### 6.1 Month over Month count of orders for different payment types

Business case: 2: 1 RUN SAVE SHARE SCHEDULE

```

201 -- 6.1 Month over Month count of orders for different payment types
202 SELECT p.payment_type, EXTRACT(MONTH FROM o.order_purchase_timestamp) as month,
203 COUNT(DISTINCT o.order_id) as No_of_orders
204 FROM target.orders o
205 INNER JOIN target.payments p
206 ON o.order_id = p.order_id
207 group by 1, 2 order by 2

```

Query results

Row	payment_type	month	No_of_orders
1	credit_card	1	6093
2	UPI	1	1715
3	voucher	1	337
4	debit_card	1	118
5	UPI	2	1723
6	credit_card	2	6582
7	voucher	2	288



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

Business case: 2: 1

RUN

SAVE

SHARE

S

```
209 -- 6.2 Count of orders based on the no. of payment installments.
210 SELECT payment_installments, COUNT(DISTINCT order_id) as No_of_orders
211 FROM `target.payments`
212 GROUP BY 1
213 ORDER BY 1
```

### Query results

JOB INFORMATION	RESULTS	JSON	EXECUTION DETAILS	EXEC
Row	payment_installment	No_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		

### Insight:

- Most people tend to buy from credit cards.
- Most of the orders are with payment installment - 1.