Business Case: Target SQL

- Bandi Saideva
- saidevabandi@gmail.com
- Setup: Big Query

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

Here I am returning all the columns of the **orders** table as it is more important table in our dataset.

Query:

```
SELECT column_name, data_type

FROM `divine-bonbon-381109`.Project_1.INFORMATION_SCHEMA.COLUMNS

WHERE TABLE_NAME = "orders"
```

Output:

Row	column_name	data_type
1	order_id	STRING
2	customer_id	STRING
3	order_status	STRING
4	order_purchase_timestamp	TIMESTAMP
5	order_approved_at	TIMESTAMP
6	order_delivered_carrier_date	TIMESTAMP
7	order_delivered_customer_date	TIMESTAMP
8	order_estimated_delivery_date	TIMESTAMP

Observations:

1. There are only 2 types of datatypes in orders table they are **STRING** and **TIMESTAMP**.

2 . Time period for which the data is given

We can get the time period of the data by using **orders** table and column **order_purchase_timestamp, order_delivered_carrier_date, order_delivered_customer_date**.

Query:

```
SELECT MIN(o.order_purchase_timestamp)
min_order_purchase_timestamp,
MAX(o.order_purchase_timestamp)
max_order_purchase_timestamp,

MIN(o.order_delivered_carrier_date)
min_delivered_carrier_date,
MAX(o.order_delivered_carrier_date)
max_delivered_carrier_date,

MIN(o.order_delivered_customer_date)
min_delivered_customer_purchase_date,
MAX(o.order_delivered_customer_date)
max_delivered_customer_purchase_date

FROM `Project_1.orders` o
```

Output:

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

Row	min_delivered_carrier_date	max_delivered_carrier_date
1	2016-10-08 10:34:01 UTC	2018-09-11 19:48:28 UTC

Row	min_delivered_customer_purchase	max_delivered_customer_purchase
1	2016-10-11 13:46:32 UTC	2018-10-17 13:22:46 UTC

Observations:

1. We can find min date in the dataset is **2016-09-04.**
2. We can find max date in the dataset is **2018-10-17**
3. The date difference between two dates is **774** days, which is almost equal to two years.

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

From the schema I can see that orders table, geolocation is connected by customers table. So we need three tables in the query.

Query:

```
SELECT count(distinct geo.geolocation_city) as count_of_cities,
count(distinct geo.geolocation_state) as count_of_states

FROM `Project_1.orders` ord

JOIN `Project_1.customers` cus ON ord.customer_id =
cus.customer_id

JOIN `Project_1.geolocation` geo ON
geo.geolocation_zip_code_prefix = cus.customer_zip_code_prefix
```

Output:

Row	count_of_cities	count_of_states
1	5812	27

Observations:

1. Total cities = 5812
2. Total states = 27

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 month

We have to find number of years for each month and finds insights from the data.

Query:

```
SELECT order_final.year_num year, order_final.month_num month,
count(*) no_of_orders

FROM

(SELECT

ord.order_purchase_timestamp, EXTRACT(MONTH FROM
(CAST(ord.order_purchase_timestamp AS DATE))) month_num,
EXTRACT(YEAR FROM (CAST(ord.order_purchase_timestamp AS DATE)))
year_num

FROM Project_1.orders ord) AS order_final

group by order_final.year_num, order_final.month_num

order by order_final.year_num, order_final.month_num
```

Output:

Row	year	month	no_of_orders
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800
5	2017	2	1780
6	2017	3	2682
7	2017	4	2404
8	2017	5	3700
9	2017	6	3245
10	2017	7	4026
11	2017	8	4331
12	2017	9	4285
13	2017	10	4631
14	2017	11	7544
15	2017	12	5673
16	2018	1	7269
17	2018	2	6728
18	2018	3	7211
19	2018	4	6939
20	2018	5	6873

Observations:

1. There is no trend in the orders sometimes they are increasing sometimes they are decreasing.

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

We have to divide hours into Dawn, Morning, Afternoon and Night, based on the hour they purchased time. Then we should count number of customers bought in Dawn, Morning, Afternoon and Night.

Query:

```
    SELECT orders_final.moment,
count(orders_final.order_purchase_timestamp) count_of_orders
```

```
FROM
(SELECT
order_sub.order_purchase_timestamp, order_sub.hour,
CASE
    WHEN hour >= 0  AND hour <= 7  THEN 'dawn'
    WHEN hour >  7  AND hour <= 12 THEN 'morning'
    WHEN hour > 12  AND hour <= 17 THEN 'after'
    WHEN hour > 17                 THEN 'night'
END  moment
FROM
(SELECT
ord.order_purchase_timestamp, EXTRACT(HOUR FROM
(CAST(ord.order_purchase_timestamp AS DATETIME))) hour
FROM Project_1.orders ord) AS order_sub) AS orders_final
GROUP BY orders_final.moment
```

Output:

Row	moment	count_of_orders
1	morning	26502
2	dawn	6473
3	after	32366
4	night	34100

Observations:

1. Most of the orders are in the Night.
2. I think we can decrease the employee force at Dawn.

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

1. Get month on month orders by states

First we have to extract month from timestamp given, then we should combine tables orders, customers, geolocation and we should count the order by state and monthe.

Query:

```
SELECT geolocation_state, month_num, count(order_id) orders

FROM

(SELECT ord.order_id, EXTRACT(MONTH FROM
(CAST(ord.order_purchase_timestamp AS DATE))) month_num,
geo.geolocation_state

FROM `Project_1.orders` ord

JOIN `Project_1.customers` cus ON ord.customer_id =
cus.customer_id

JOIN `Project_1.geolocation` geo ON
geo.geolocation_zip_code_prefix = cus.customer_zip_code_prefix)
as final_table

group by 1, 2

order by orders DESC
```

Output:

Row	geolocation_state	month_num	orders
1	SP	8	660764
2	SP	5	617152
3	SP	7	596542
4	SP	6	550147
5	SP	3	540963
6	SP	4	522895
7	SP	2	460265
8	SP	1	453515
9	SP	11	412132
10	SP	12	325198
11	RJ	5	310453
12	RJ	8	308365
13	MG	3	307366
14	RJ	3	301419
15	RJ	7	298006
16	MG	8	293873
17	MG	5	292339

Observations:

1. Most of the orders are from SP state.
2. So we should not lose customers from that state, we should keep running customer retention process for this state.

2. Distribution of customers across the states in Brazil

To get this, first we should combine tables customers and geolocation tables ang get count of customers for each state.

Query:

```
SELECT

geo.geolocation_state, count(cus.customer_id) count_of_customers

FROM `Project_1.customers` cus
```

```
JOIN Project_1.geolocation geo ON cus.customer_zip_code_prefix =
geo.geolocation_zip_code_prefix

GROUP BY geo.geolocation_state

ORDER BY count_of_customers DESC
```

Output:

Row	geolocation_state	count_of_customers
1	SP	5620430
2	RJ	3015690
3	MG	2878728
4	RS	805370
5	PR	626021
6	SC	538638
7	BA	365875
8	ES	316654
9	GO	133146
10	MT	122395
11	PE	114588
12	DF	93309
13	PA	83554
14	CE	63507
15	MS	61473
16	MA	53383
17	AL	34861
18	PB	27714

Observations:

1. Most of the customers are also from SP state, so we should consider this state as important state for our business.

4. Impact on Economy: Analyze 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) - You can use “payment_value” column in payments table

First get sum of payment_value in 2017 → 1 and get sum of payment_value in 2018 → 2. Divide 2 by 1 and multiply by 100 to get percentage increase in cost of orders from 2017 to 2018.

Query:

```
SELECT ROUND((SUM(payments_2018) / SUM(payments_2017)) * 100, 2)
percentage_increase_from_2017_to_2018

FROM

(SELECT

CASE

  WHEN EXTRACT(YEAR FROM (CAST(ord.order_purchase_timestamp AS
DATETIME))) = 2017

       AND EXTRACT(MONTH FROM (CAST(ord.order_purchase_timestamp
AS DATETIME))) <= 7

  THEN pay.payment_value

  ELSE 0

END AS payments_2017,

CASE

  WHEN EXTRACT(YEAR FROM (CAST(ord.order_purchase_timestamp AS
DATETIME))) = 2018

       AND EXTRACT(MONTH FROM (CAST(ord.order_purchase_timestamp
AS DATETIME))) <= 7

  THEN pay.payment_value
```

```
  ELSE 0

END AS payments_2018

FROM

`Project_1.orders` ord

JOIN Project_1.payments pay ON pay.order_id = ord.order_id) AS
final_table
```

Output:

Row	percentage_increase_from_2017_to_2018
1	256.2

Observations:

1. There is 256% increase in sales compared to 2017, this is a good sign for business, so we should continue it.

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

Combine tables customers, geolocation, order_items get price, freight from customer state. Mean and sum them group by state.

Query:

```
SELECT

geo.geolocation_state state, ROUND(SUM(ord_ite.price), 2)
sum_price, ROUND(SUM(ord_ite.freight_value), 2) sum_freight,

ROUND(SUM(ord_ite.price) / count(cus.customer_id), 2)
average_price, ROUND(SUM(ord_ite.freight_value) /
count(cus.customer_id), 2) average_freight

FROM  `Project_1.customers` cus

JOIN `Project_1.geolocation` geo ON
geo.geolocation_zip_code_prefix = customer_zip_code_prefix

JOIN Project_1.orders ord ON  ord.customer_id = cus.customer_id
```

```
JOIN `Project_1.order_items` ord_ite ON ord.order_id =
ord_ite.order_id

GROUP BY geo.geolocation_state

ORDER BY sum_price DESC, sum_freight DESC, average_price DESC,
average_freight DESC
```

Output:

Row	state	sum_price	sum_freight	average_price	average_freight
1	SP	711838740....	98574572.43	111.28	15.41
2	RJ	440142503....	71966793.75	127.81	20.9
3	MG	397190155....	67058347.09	121.18	20.46
4	RS	111183139....	19910834.35	120.18	21.52
5	PR	85392469.28	14432159.77	119.21	20.15
6	SC	79666423.29	13472314.62	127.41	21.55
7	BA	62377311.67	11345094.0	149.64	27.22
8	ES	43634878.56	7799979.09	123.36	22.05
9	MT	22777072.82	4177068.03	156.63	28.72
10	GO	20860945.92	3590268.56	134.62	23.17
11	PE	17545068.94	4195977.72	137.42	32.87
12	PA	15586180.17	3409472.09	166.98	36.53
13	DF	13141649.62	2214955.55	124.66	21.01
14	CE	10819201.81	2306600.06	151.32	32.26
15	MS	9891112.52	1698977.52	139.1	23.89
16	MA	9020091.01	2275191.86	150.95	38.08
17	AL	7191886.1	1237356.22	196.64	33.83
18	PB	6278650.25	1350462.24	198.86	42.77

Observations:

1. We should decrease the price and freight from these values, we should come up with some logic, so that this should get decrease.

5. Analysis on sales, freight and delivery time

1. Calculate days between purchasing, delivering and estimated delivery

Here I am going to calculate average purchasing, delivery and estimated delivery time as question is not making a lot of sense, <– This I am going to do using orders table.

Note: Here I am considering order_delivered_customer_date as delivering date.

Query:

```
SELECT ROUND(SUM(date_diff_delivery) / COUNT(order_id), 2) as
avg_delivery_time, ROUND(SUM(date_diff_estd_delivery) /
COUNT(order_id), 2) avg_estd_delivery_time

FROM

(SELECT ord.order_id, ord.order_purchase_timestamp,
ord.order_delivered_customer_date,
ord.order_estimated_delivery_date,

DATE_DIFF(CAST(ord.order_delivered_customer_date as DATETIME),
CAST(ord.order_purchase_timestamp as DATETIME), DAY)
date_diff_delivery,

DATE_DIFF(CAST(ord.order_estimated_delivery_date as DATETIME),
CAST(ord.order_delivered_customer_date as DATETIME), DAY)
date_diff_estd_delivery

FROM Project_1.orders ord) AS final_table
```

Output:

Row	avg_delivery_time	avg_estd_delivery_time
1	12.12	11.52

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

Query:

```
SELECT ord.order_id, ord.order_purchase_timestamp,
ord.order_delivered_customer_date,
ord.order_estimated_delivery_date,

DATE_DIFF(CAST(ord.order_delivered_customer_date as DATETIME),
CAST(ord.order_purchase_timestamp as DATETIME), DAY)
time_to_delivery ,

DATE_DIFF(CAST(ord.order_estimated_delivery_date as DATETIME),
CAST(ord.order_delivered_customer_date as DATETIME), DAY)
diff_estimated_delivery

FROM Project_1.orders ord

ORDER BY time_to_delivery DESC, diff_estimated_delivery DESC
```

Output:

Row	order_id	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	time_to_delivery	diff_estimated_delivery
1	ca07593549f1816d26a572e06dc1eab6	2017-02-21 23:31:27 UTC	2017-09-19 14:36:39 UTC	2017-03-22 00:00:00 UTC	210	-181
2	1b3190b2dfa9d789e1f14c05b647a14a	2018-02-23 14:57:35 UTC	2018-09-19 23:24:07 UTC	2018-03-15 00:00:00 UTC	208	-188
3	440d0d17af552815d15a9e41abe49359	2017-03-07 23:59:51 UTC	2017-09-19 15:12:50 UTC	2017-04-07 00:00:00 UTC	196	-165
4	2fb597c2f772eca01b1f5c561bf6cc7b	2017-03-08 18:09:02 UTC	2017-09-19 14:33:17 UTC	2017-04-17 00:00:00 UTC	195	-155
5	285ab9426d6982034523a855f55a885e	2017-03-08 22:47:40 UTC	2017-09-19 14:00:04 UTC	2017-04-06 00:00:00 UTC	195	-166
6	0f4519c5f1c541ddec9f21b3bddd533a	2017-03-09 13:26:57 UTC	2017-09-19 14:38:21 UTC	2017-04-11 00:00:00 UTC	194	-161
7	47b40429ed8cce3aee9199792275433f	2018-01-03 09:44:01 UTC	2018-07-13 20:51:31 UTC	2018-01-19 00:00:00 UTC	191	-175
8	2fe324febf907e3ea3f2aa9650869fa5	2017-03-13 20:17:10 UTC	2017-09-19 17:00:07 UTC	2017-04-05 00:00:00 UTC	190	-167
9	2d7561026d542c8dbd8f0daeadf67a43	2017-03-15 11:24:27 UTC	2017-09-19 14:38:18 UTC	2017-04-13 00:00:00 UTC	188	-159

3 Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery

We can do this easily this if we have 2nd query, we just have to take mean of freight value, time_to_delivery, diff_estimated_delivery by group by of state.

Query:

```
SELECT

geo.geolocation_state state, ROUND(SUM(ord_ite.freight_value)/
COUNT(ord.order_id), 2) mean_freight,

ROUND(SUM(ord.time_to_delivery)/ COUNT(ord.order_id), 2)
mean_time_to_delivery,
```

```
ROUND(SUM(ord.diff_estimated_delivery)/ COUNT(ord.order_id), 2)
mean_diff_estimated_delivery

FROM

(SELECT ord.order_id, ord.customer_id,

DATE_DIFF(CAST(ord.order_delivered_customer_date as DATETIME),
CAST(ord.order_purchase_timestamp as DATETIME), DAY)
time_to_delivery ,

DATE_DIFF(CAST(ord.order_estimated_delivery_date as DATETIME),
CAST(ord.order_delivered_customer_date as DATETIME), DAY)
diff_estimated_delivery

FROM Project_1.orders ord) as ord

JOIN Project_1.order_items ord_ite ON ord_ite.order_id =
ord.order_id

JOIN `Project_1.customers` cus ON cus.customer_id =
ord.customer_id

JOIN Project_1.geolocation geo ON cus.customer_zip_code_prefix =
geo.geolocation_zip_code_prefix

GROUP BY geo.geolocation_state

ORDER BY mean_freight DESC, mean_time_to_delivery DESC,
mean_diff_estimated_delivery DESC
```

Output:

Row	state	mean_freight	mean_time_to_delivery	mean_diff_estimated_delivery
1	PB	42.77	19.83	13.1
2	RR	42.47	20.77	18.59
3	PI	39.48	17.7	12.0
4	AC	39.1	19.98	19.14
5	MA	38.08	20.7	9.53
6	RO	37.43	18.52	19.56
7	TO	37.36	16.3	12.13
8	PA	36.53	22.58	14.12
9	AP	35.66	30.11	16.23
10	SE	34.67	21.12	9.37
11	AM	34.62	24.62	21.41
12	RN	34.07	18.56	13.89

4 Sort the data to get the following:

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

If we have solution 3, just order by average freight value and use desc/asc

DESC/ Highest average freight values top 5:

Query:

```
SELECT

geo.geolocation_state state, ROUND(SUM(ord_ite.freight_value)/
COUNT(ord.order_id), 2) mean_freight,

ROUND(SUM(ord.time_to_delivery)/ COUNT(ord.order_id), 2)
mean_time_to_delivery,

ROUND(SUM(ord.diff_estimated_delivery)/ COUNT(ord.order_id), 2)
mean_diff_estimated_delivery

FROM

(SELECT ord.order_id, ord.customer_id,
```

```
DATE_DIFF(CAST(ord.order_delivered_customer_date as DATETIME),
CAST(ord.order_purchase_timestamp as DATETIME), DAY)
time_to_delivery ,

DATE_DIFF(CAST(ord.order_estimated_delivery_date as DATETIME),
CAST(ord.order_delivered_customer_date as DATETIME), DAY)
diff_estimated_delivery

FROM Project_1.orders ord) as ord

JOIN Project_1.order_items ord_ite ON ord_ite.order_id =
ord.order_id

JOIN `Project_1.customers` cus ON cus.customer_id =
ord.customer_id

JOIN Project_1.geolocation geo ON cus.customer_zip_code_prefix =
geo.geolocation_zip_code_prefix

GROUP BY geo.geolocation_state

ORDER BY mean_freight DESC

LIMIT 5
```

Output:

Row	state	mean_freight	mean_time_to_delivery	mean_diff_estimated_delivery
1	PB	42.77	19.83	13.1
2	RR	42.47	20.77	18.59
3	PI	39.48	17.7	12.0
4	AC	39.1	19.98	19.14
5	MA	38.08	20.7	9.53

ASC/ Lowest average freight values top 5:

Query:

```
SELECT

geo.geolocation_state state, ROUND(SUM(ord_ite.freight_value)/
COUNT(ord.order_id), 2) mean_freight,

ROUND(SUM(ord.time_to_delivery)/ COUNT(ord.order_id), 2)
mean_time_to_delivery,
```

```
ROUND(SUM(ord.diff_estimated_delivery)/ COUNT(ord.order_id), 2)
mean_diff_estimated_delivery

FROM

(SELECT ord.order_id, ord.customer_id,

DATE_DIFF(CAST(ord.order_delivered_customer_date as DATETIME),
CAST(ord.order_purchase_timestamp as DATETIME), DAY)
time_to_delivery ,

DATE_DIFF(CAST(ord.order_estimated_delivery_date as DATETIME),
CAST(ord.order_delivered_customer_date as DATETIME), DAY)
diff_estimated_delivery

FROM Project_1.orders ord) as ord

JOIN Project_1.order_items ord_ite ON ord_ite.order_id =
ord.order_id

JOIN `Project_1.customers` cus ON cus.customer_id =
ord.customer_id

JOIN Project_1.geolocation geo ON cus.customer_zip_code_prefix =
geo.geolocation_zip_code_prefix

GROUP BY geo.geolocation_state

ORDER BY mean_freight ASC

LIMIT 5
```

Output:

Row	state	mean_freight	mean_time_to_delivery	mean_diff_estimated_delivery
1	SP	15.41	8.66	11.07
2	PR	20.15	11.21	13.42
3	MG	20.46	11.57	13.21
4	RJ	20.9	14.39	12.06
5	DF	21.01	12.65	12.25

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

If we have solution 3, just order by average time to delivery and use desc/asc

Lowest AVG delivery time:

Query:

```
SELECT

geo.geolocation_state state, ROUND(SUM(ord_ite.freight_value)/
COUNT(ord.order_id), 2) mean_freight,

ROUND(SUM(ord.time_to_delivery)/ COUNT(ord.order_id), 2)
mean_time_to_delivery,

ROUND(SUM(ord.diff_estimated_delivery)/ COUNT(ord.order_id), 2)
mean_diff_estimated_delivery

FROM

(SELECT ord.order_id, ord.customer_id,

DATE_DIFF(CAST(ord.order_delivered_customer_date as DATETIME),
CAST(ord.order_purchase_timestamp as DATETIME), DAY)
time_to_delivery ,

DATE_DIFF(CAST(ord.order_estimated_delivery_date as DATETIME),
CAST(ord.order_delivered_customer_date as DATETIME), DAY)
diff_estimated_delivery

FROM Project_1.orders ord) as ord

JOIN Project_1.order_items ord_ite ON ord_ite.order_id =
ord.order_id

JOIN `Project_1.customers` cus ON cus.customer_id =
ord.customer_id

JOIN Project_1.geolocation geo ON cus.customer_zip_code_prefix =
geo.geolocation_zip_code_prefix

GROUP BY geo.geolocation_state

ORDER BY mean_time_to_delivery ASC

LIMIT 5
```

Output:

Row	state	mean_freight	mean_time_to_delivery	mean_diff_estimated_delivery
1	SP	15.41	8.66	11.07
2	PR	20.15	11.21	13.42
3	MG	20.46	11.57	13.21
4	DF	21.01	12.65	12.25
5	RJ	20.9	14.39	12.06

DESC OR Highest Avg delivery time:

Query:

```
SELECT

geo.geolocation_state state, ROUND(SUM(ord_ite.freight_value)/
COUNT(ord.order_id), 2) mean_freight,

ROUND(SUM(ord.time_to_delivery)/ COUNT(ord.order_id), 2)
mean_time_to_delivery,

ROUND(SUM(ord.diff_estimated_delivery)/ COUNT(ord.order_id), 2)
mean_diff_estimated_delivery

FROM

(SELECT ord.order_id, ord.customer_id,

DATE_DIFF(CAST(ord.order_delivered_customer_date as DATETIME),
CAST(ord.order_purchase_timestamp as DATETIME), DAY)
time_to_delivery ,

DATE_DIFF(CAST(ord.order_estimated_delivery_date as DATETIME),
CAST(ord.order_delivered_customer_date as DATETIME), DAY)
diff_estimated_delivery

FROM Project_1.orders ord) as ord

JOIN Project_1.order_items ord_ite ON ord_ite.order_id =
ord.order_id

JOIN `Project_1.customers` cus ON cus.customer_id =
ord.customer_id

JOIN Project_1.geolocation geo ON cus.customer_zip_code_prefix =
geo.geolocation_zip_code_prefix
```

```
GROUP BY geo.geolocation_state

ORDER BY mean_time_to_delivery DESC

LIMIT 5
```

Output:

Row	state	mean_freight	mean_time_to_delivery	mean_diff_estimated_delivery
1	AP	35.66	30.11	16.23
2	AM	34.62	24.62	21.41
3	AL	33.83	22.74	9.05
4	PA	36.53	22.58	14.12
5	SE	34.67	21.12	9.37

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

If we have solution 3, just order by `mean_diff_estimated_delivery` and use desc/asc

Fast delivery compared to estimated Date:

Query:

```
SELECT

geo.geolocation_state state, ROUND(SUM(ord_ite.freight_value)/
COUNT(ord.order_id), 2) mean_freight,

ROUND(SUM(ord.time_to_delivery)/ COUNT(ord.order_id), 2)
mean_time_to_delivery,

ROUND(SUM(ord.diff_estimated_delivery)/ COUNT(ord.order_id), 2)
mean_diff_estimated_delivery

FROM

(SELECT ord.order_id, ord.customer_id,

DATE_DIFF(CAST(ord.order_delivered_customer_date as DATETIME),
CAST(ord.order_purchase_timestamp as DATETIME), DAY)
time_to_delivery ,

DATE_DIFF(CAST(ord.order_estimated_delivery_date as DATETIME),
CAST(ord.order_delivered_customer_date as DATETIME), DAY)
diff_estimated_delivery
```

```
FROM Project_1.orders ord) as ord

JOIN Project_1.order_items ord_ite ON ord_ite.order_id =
ord.order_id

JOIN `Project_1.customers` cus ON cus.customer_id =
ord.customer_id

JOIN Project_1.geolocation geo ON cus.customer_zip_code_prefix =
geo.geolocation_zip_code_prefix

GROUP BY geo.geolocation_state

ORDER BY mean_diff_estimated_delivery ASC

LIMIT 5
```

Output:

Row	state	mean_freight	mean_time_to_delivery	mean_diff_estimated_delivery
1	AL	33.83	22.74	9.05
2	SE	34.67	21.12	9.37
3	MA	38.08	20.7	9.53
4	CE	32.26	20.54	10.51
5	ES	22.05	14.87	10.83

Slow delivery compared to estimated Date:

Query:

```
SELECT

geo.geolocation_state state, ROUND(SUM(ord_ite.freight_value)/
COUNT(ord.order_id), 2) mean_freight,

ROUND(SUM(ord.time_to_delivery)/ COUNT(ord.order_id), 2)
mean_time_to_delivery,

ROUND(SUM(ord.diff_estimated_delivery)/ COUNT(ord.order_id), 2)
mean_diff_estimated_delivery

FROM
```

```
(SELECT ord.order_id, ord.customer_id,

DATE_DIFF(CAST(ord.order_delivered_customer_date as DATETIME),
CAST(ord.order_purchase_timestamp as DATETIME), DAY)
time_to_delivery ,

DATE_DIFF(CAST(ord.order_estimated_delivery_date as DATETIME),
CAST(ord.order_delivered_customer_date as DATETIME), DAY)
diff_estimated_delivery

FROM Project_1.orders ord) as ord

JOIN Project_1.order_items ord_ite ON ord_ite.order_id =
ord.order_id

JOIN `Project_1.customers` cus ON cus.customer_id =
ord.customer_id

JOIN Project_1.geolocation geo ON cus.customer_zip_code_prefix =
geo.geolocation_zip_code_prefix

GROUP BY geo.geolocation_state

ORDER BY mean_diff_estimated_delivery DESC

LIMIT 5
```

Output:

Row	state	mean_freight	mean_time_to_delivery	mean_diff_estimated_delivery
1	AM	34.62	24.62	21.41
2	RO	37.43	18.52	19.56
3	AC	39.1	19.98	19.14
4	RR	42.47	20.77	18.59
5	AP	35.66	30.11	16.23

6. Payment Analysis

1 Month over Month count of orders for different payment types

We have to combine tables orders, payments and group by month, payment_type and count the orders for that month, payment_type using group by.

Query:

```
SELECT ord.month, pay.payment_type, count(ord.order_id)
count_of_orders

FROM

(SELECT ord.order_id, ord.customer_id,

EXTRACT(MONTH FROM CAST(ord.order_delivered_customer_date as
DATETIME)) month

FROM Project_1.orders ord) as ord

JOIN Project_1.payments pay ON pay.order_id = ord.order_id

WHERE ord.month is not null

GROUP BY ord.month, pay.payment_type

ORDER BY ord.month, pay.payment_type, count_of_orders DESC
```

Output:

Row	month	payment_type	count_of_orders
1	1	UPI	1454
2	1	credit_card	5211
3	1	debit_card	107
4	1	voucher	385
5	2	UPI	1425
6	2	credit_card	5609
7	2	debit_card	79
8	2	voucher	412
9	3	UPI	1899
10	3	credit_card	7086

Observations:

1. From the top rows I can observe most of the orders are using UPI and credit card.

2 Count of orders based on the no. of payment installments

We should join tables ord and payments and group by payment_installments and count the orders.

Query:

```
SELECT

pay.payment_installments, COUNT(ord.order_id) count_of_orders

FROM

`Project_1.orders` ord

JOIN Project_1.payments pay ON ord.order_id = pay.order_id

GROUP BY pay.payment_installments

ORDER BY count_of_orders DESC
```

Output:

Row	payment_installments	count_of_orders
1	1	52546
2	2	12413
3	3	10461
4	4	7098
5	10	5328
6	5	5239
7	8	4268
8	6	3920
9	7	1626
10	9	644
11	12	133

Observations:

1. Most of the orders are with low payment_installments. → means people are not willing to buy things with installments.