Target SQL Business Case

- M Santosh Kumar (2023 batch)

1. Exploratory Data Analysis

Overall Structure of Dataset

- Query: select dataset_id, table_id, row_count, ROUND(size_bytes/POW(10,6),2) as size_MB
  - from `s-business-case-target-sql.TargetSQL.__TABLES__`
- Above query gives overall structure of all 8 tables as below (including count of rows & size of table)

Row	dataset_id	table_id	row_count	size_mb
1	TargetSQL	customers	99441	9.18
2	TargetSQL	geolocation	1000163	40.56
3	TargetSQL	order_items	112650	15.1
4	TargetSQL	order_reviews	99224	9.29
5	TargetSQL	orders	99441	11.79
6	TargetSQL	payments	103886	7.19
7	TargetSQL	products	32951	3.47
8	TargetSQL	sellers	3095	0.18

1.1. Data type of columns in a table

- Query: select * from `s-business-case-target-sql.TargetSQL.INFORMATION_SCHEMA.COLUMNS`
- Using INFORMATION_SCHEMA.COLUMNS in BiqQuery, we can see data types of all columns of all tables in given dataset (sample results below)
- There are totally 49 columns in the 8 tables. All columns are of one of the 4 data types: INT64, FLOAT64, STRING, TIMESTAMP

w	table_name	column_name	ordinal_position	is_nullable	data_type
1	order_items	order_id	1	YES	STRING
2	order_items	order_item_id	2	YES	INT64
3	order_items	product_id	3	YES	STRING
4	order_items	seller_id	4	YES	STRING
5	order_items	shipping_limit_date	5	YES	TIMESTAMP
6	order_items	price	6	YES	FLOAT64
7	order_items	freight_value	7	YES	FLOAT64
8	sellers	seller_id	1	YES	STRING
9	sellers	seller_zip_code_prefix	2	YES	INT64
10	sellers	seller_city	3	YES	STRING

1.2. Time period for which the data is given

- Query:
```
select min(order_purchase_timestamp) as Data_From, max(order_purchase_timestamp)
as Data_Till from `s-business-case-target-sql.TargetSQL.orders`;
```

- Orders data is available from: ***2016-09-04 till 2018-10-17*** (As shown below)

Query results

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

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

1.3. Cities and States covered in the dataset

- Query: select distinct cust.customer_city, cust.customer_state
  - from `s-business-case-target-sql.TargetSQL.orders` ord
    join `s-business-case-target-sql.TargetSQL.customers` cust
    on ord.customer_id = cust.customer_id;

- By joining orders & customers tables, we can get cities & states from which the orders were made
- As we see below, there are total 4310 city-state combinations from where we received orders

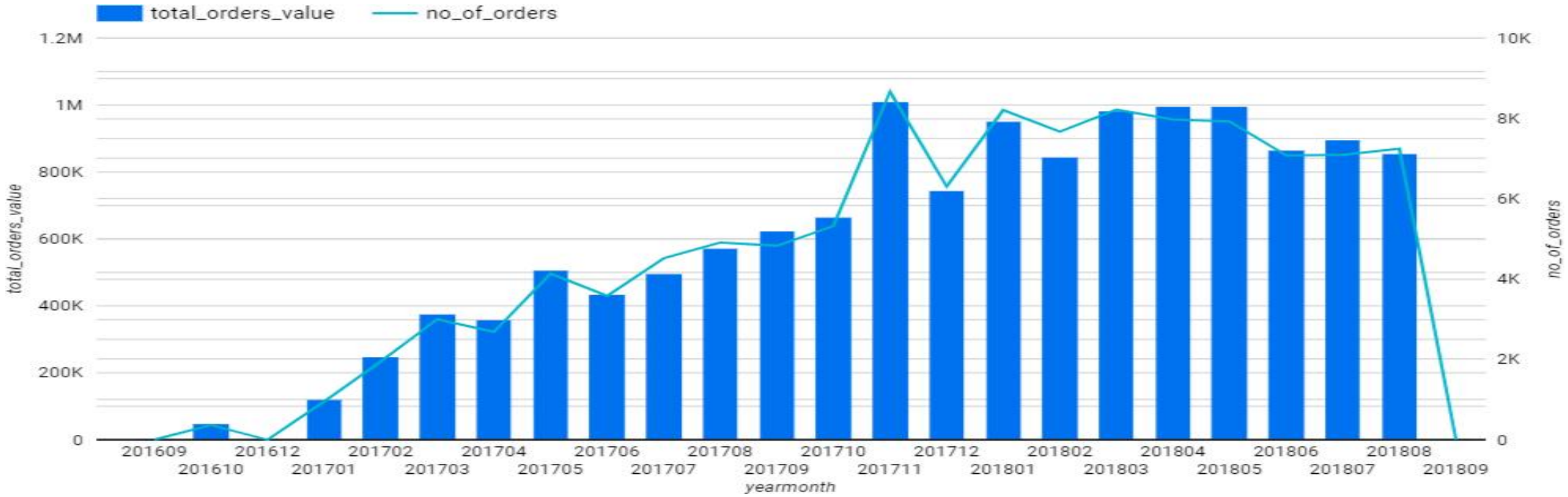
Row	customer_city	customer_state
1	rio de janeiro	RJ
2	sao leopoldo	RS
3	general salgado	SP
4	brasilia	DF
5	paranavai	PR
6	cuiaba	MT
7	sao luis	MA
8	maceio	AL
9	hortolandia	SP
10	varzea grande	MT

Results per page: 50 1 – 50 of 4310

2. In-depth Exploration

2.1. Growing Trend, Seasonality & Peaks

```
Query: select CONCAT(CAST(EXTRACT(YEAR from ord.order_purchase_timestamp) as string),
    LPAD(CAST(EXTRACT(MONTH from ord.order_purchase_timestamp) as string),2,'0') ) as yearmonth,
    round(sum(ordItms.price),2) as total_orders_value,
    count(ord.order_id) as no_of_orders
            from `s-business-case-target-sql.TargetSQL.orders` ord
            join `s-business-case-target-sql.TargetSQL.order_items` ordItms on ord.order_id = ordItms.order_id
            group by yearmonth
            order by yearmonth;
```



2.1. Growing Trend, Seasonality & Peaks

Observations/Insights:

- We can see an absolute Growth trend (in both no. of orders & total order value) from 2016 to 2018 in the Brazilian e-commerce business.
- However, 2017 year clearly is a growth year. Whereas, in 2018, business almost stagnated at similar levels.
- Full 12-month data (Jan to Dec) is available only in 2017 year. So, we can't clearly comment on monthly trends. But, Nov apparently is peak month, followed by a decline in Dec (holiday season)

Recommendation:

- Stock up & optimise resources before November month

2.2. Specific time of purchase?

```
Query: select CASE when EXTRACT(hour from ord.order_purchase_timestamp) between 0 and 6 then 'Before Dawn'
          when EXTRACT(hour from ord.order_purchase_timestamp) between 6 and 12 then 'Morning'
          when EXTRACT(hour from ord.order_purchase_timestamp) between 12 and 18 then 'Afternoon'
          else 'Night'
        END as time_of_day,
       count(ord.order_id) as no_of_orders
  from `s-business-case-target-sql.TargetSQL.orders` ord
  group by time_of_day
  order by no_of_orders;
```

Insights:

- Brazilians order very less Before Dawn (12am to 6am), and then as time passes on, the no. of orders increase from Morning (6am to 12noon) to Afternoon (12noon to 6pm). Again, the orders will drop in the Night (6pm to 12am)
- Thus, Afternoons are peak hours. Any maintenance activities can be planned before dawn

Query results

JOB INFORMATION | RESULTS | JSON | EXI

Row	time_of_day	no_of_orders
1	Before Dawn	5242
2	Morning	27733
3	Night	28331
4	Afternoon	38135

3. Evolution of Brazilian E-commerce

3.1. Month-on-month orders

Month-on-month trend across Brazil:

Query:

```
select CONCAT(CAST(EXTRACT(YEAR from ord.order_purchase_timestamp) as string),
      LPAD(CAST(EXTRACT(MONTH from ord.order_purchase_timestamp) as string),2,'0') )  as yearmonth,
 round(sum(ordItms.price),2) as total_order_value
 from `s-business-case-target-sql.TargetSQL.orders` ord
  join `s-business-case-target-sql.TargetSQL.order_items` ordItms on ord.order_id = ordItms.order_id
 group by yearmonth
 order by yearmonth;
```

Row	yearmonth	total_order_valu
1	201609	267.36
2	201610	49507.66
3	201612	10.9
4	201701	120312.87
5	201702	247303.02
6	201703	374344.3
7	201704	359927.23
8	201705	506071.14
9	201706	433038.6
10	201707	498031.48
11	201708	573971.68

3.1. Month-on-month orders by region, states

Month-on-month orders by region, states:

Query:

```
select CONCAT(CAST(EXTRACT(YEAR from ord.order_purchase_timestamp) as string),
    LPAD(CAST(EXTRACT(MONTH from ord.order_purchase_timestamp) as string),2,'0') )  as yearmonth,
cust.customer_city, cust.customer_state,
round(sum(ordItms.price),2) as total_order_value
from `s-business-case-target-sql.TargetSQL.orders` ord
  join `s-business-case-target-sql.TargetSQL.order_items` ordItms on ord.order_id = ordItms.order_id
join `s-business-case-target-sql.TargetSQL.customers` cust on ord.customer_id = cust.customer_id
group by yearmonth, cust.customer_city, cust.customer_state
order by yearmonth, cust.customer_city, cust.customer_state;
```

Row	yearmonth	customer_city	customer_state	total_order_value
1	201609	boa vista	RR	72.89
2	201609	passo fundo	RS	59.5
3	201609	sao joaquim da barra	SP	134.97
4	201610	alem paraiba	MG	69.9
5	201610	ananindeua	PA	189.0
6	201610	aparecida de goiania	GO	49.0
7	201610	apuarema	BA	169.99
8	201610	aracaju	SE	58.0
9	201610	aracariguama	SP	64.9
10	201610	bacaxa	RJ	57.9
11	201610	bage	RS	129.99

3.2. How are customers distributed in Brazil

City-wise distribution:

- Query:
```
select customer_city, count(customer_id) as no_of_customers
from `s-business-case-target-sql.TargetSQL.customers`
group by customer_city order by no_of_customers desc
```

- Top 10 city results are in screenshot:

Row	customer_city	no_of_customer
1	sao paulo	15540
2	rio de janeiro	6882
3	belo horizonte	2773
4	brasilia	2131
5	curitiba	1521
6	campinas	1444
7	porto alegre	1379
8	salvador	1245
9	guarulhos	1189
10	sao bernardo do campo	938

3.2. How are customers distributed in Brazil

State-wise distribution:

- Query:
```
select customer_state, count(customer_id) as no_of_customers
from `s-business-case-target-sql.TargetSQL.customers`
group by customer_state order by no_of_customers desc
```

- Top 10 state results are in screenshot:

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
9	ES	2033
10	GO	2020

4. Impact on Economy: Money movement & Inflation

4.1. % increase in cost of orders from 2017 to 2018

- Query:
```
select (temp_table.year_total_order_price/(LAG(temp_table.year_total_order_price) over (o
rder by temp_table.year)) - 1)*100 as perc_growth
 from (
        select extract(year from ord.order_purchase_timestamp) as year,
            round(sum(ordItms.price),2) as year_total_order_price
            from `s-business-case-target-sql.TargetSQL.orders` ord
            join `s-business-case-target-
sql.TargetSQL.order_items` ordItms on ord.order_id = ordItms.order_id
            where extract(month from ord.order_purchase_timestamp) between 1 and 8
            group by year
        ) as temp_table;
```

Row	perc_growth
1	null
2	137.260039...

Observations: There is a huge growth of 137.26% in total ordered value from 2017 to 2018. As Brazilians are ordering more online, the Target business upscale its infrastructure, invest in marketing and increase its customer base further.

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

Query:

```
select cust.customer_state, round(avg(ordItms.price),2) as avg_price, round(sum(ordItms.price),2) as sum_price,
  round(avg(ordItms.freight_value),2) as avg_freight_value, round(sum(ordItms.freight_value),2) as sum_freight_value
        from `s-business-case-target-sql.TargetSQL.order_items` ordItms
        join `s-business-case-target-sql.TargetSQL.orders` ord on ordItms.order_id = ord.order_id
        join `s-business-case-target-sql.TargetSQL.customers` cust on ord.customer_id = cust.customer_id
        group by cust.customer_state order by cust.customer_state;
```

Row	customer_state	avg_price	sum_price	avg_freight_value	sum_freight_value
1	AC	173.73	15982.95	40.07	3686.75
2	AL	180.89	80314.81	35.84	15914.59
3	AM	135.5	22356.84	33.21	5478.89
4	AP	164.32	13474.3	34.01	2788.5
5	BA	134.6	511349.99	26.36	100156.68
6	CE	153.76	227254.71	32.71	48351.59
7	DF	125.77	302603.94	21.04	50625.5
8	ES	121.91	275037.31	22.06	49764.6
9	GO	126.27	294591.95	22.77	53114.98
10	MA	145.2	119648.22	38.26	31523.77
11	MG	120.75	1585308.03	20.63	270853.46

5. Analysis on sales, freight and delivery time

5.1. Days between purchasing, delivering and estimated delivery

```
Query: select order_purchase_timestamp, order_delivered_customer_date, order_estimated_delivery_date,
  date_diff(order_estimated_delivery_date, order_purchase_timestamp, DAY) as est_days_for_delivery,
  date_diff(order_delivered_customer_date, order_purchase_timestamp, DAY) as actual_days_for_delivery,
  date_diff(order_delivered_customer_date, order_estimated_delivery_date, DAY) as delay_beyond_estimation
        from `s-business-case-target-sql.TargetSQL.orders`
        where order_delivered_customer_date is not null
        order by order_purchase_timestamp;
```

Row	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	est_days_for_delivery	actual_days_for_delivery	delay_beyond_estimation
1	2016-09-15 12:16:38 UTC	2016-11-09 07:47:38 UTC	2016-10-04 00:00:00 UTC	18	54	36
2	2016-10-03 09:44:50 UTC	2016-10-26 14:02:13 UTC	2016-10-27 00:00:00 UTC	23	23	0
3	2016-10-03 16:56:50 UTC	2016-10-27 18:19:38 UTC	2016-11-07 00:00:00 UTC	34	24	-10
4	2016-10-03 21:01:41 UTC	2016-11-08 10:58:34 UTC	2016-11-25 00:00:00 UTC	52	35	-16
5	2016-10-03 21:13:36 UTC	2016-11-03 10:58:07 UTC	2016-11-29 00:00:00 UTC	56	30	-25
6	2016-10-03 22:06:03 UTC	2016-10-31 11:07:42 UTC	2016-11-23 00:00:00 UTC	50	27	-22
7	2016-10-03 22:31:31 UTC	2016-10-14 16:08:00 UTC	2016-11-23 00:00:00 UTC	50	10	-39
8	2016-10-03 22:44:10 UTC	2016-11-03 14:04:50 UTC	2016-12-01 00:00:00 UTC	58	30	-27
9	2016-10-03 22:51:30 UTC	2016-11-01 15:14:45 UTC	2016-11-25 00:00:00 UTC	52	28	-23
10	2016-10-04 09:06:10 UTC	2016-10-22 14:51:18 UTC	2016-11-24 00:00:00 UTC	50	18	-32
11	2016-10-04 09:16:33 UTC	2016-10-24 16:33:45 UTC	2016-11-24 00:00:00 UTC	50	20	-30

5.2. Time to delivery & Diff from estimated delivery

Query:
```
select order_purchase_timestamp, order_delivered_customer_date, order_estimated_delivery_date,
  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 `s-business-case-target-sql.TargetSQL.orders`
        where order_delivered_customer_date is not null
        order by order_purchase_timestamp;
```

Row	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	time_to_delivery	diff_estimated_
1	2016-09-15 12:16:38 UTC	2016-11-09 07:47:38 UTC	2016-10-04 00:00:00 UTC	54	-36
2	2016-10-03 09:44:50 UTC	2016-10-26 14:02:13 UTC	2016-10-27 00:00:00 UTC	23	0
3	2016-10-03 16:56:50 UTC	2016-10-27 18:19:38 UTC	2016-11-07 00:00:00 UTC	24	10
4	2016-10-03 21:01:41 UTC	2016-11-08 10:58:34 UTC	2016-11-25 00:00:00 UTC	35	16
5	2016-10-03 21:13:36 UTC	2016-11-03 10:58:07 UTC	2016-11-29 00:00:00 UTC	30	25
6	2016-10-03 22:06:03 UTC	2016-10-31 11:07:42 UTC	2016-11-23 00:00:00 UTC	27	22
7	2016-10-03 22:31:31 UTC	2016-10-14 16:08:00 UTC	2016-11-23 00:00:00 UTC	10	39
8	2016-10-03 22:44:10 UTC	2016-11-03 14:04:50 UTC	2016-12-01 00:00:00 UTC	30	27
9	2016-10-03 22:51:30 UTC	2016-11-01 15:14:45 UTC	2016-11-25 00:00:00 UTC	28	23
10	2016-10-04 09:06:10 UTC	2016-10-22 14:51:18 UTC	2016-11-24 00:00:00 UTC	18	32

5.3. Summarising Freight & Delivery info by States

```
Query: select cust.customer_state, round(avg(ordItms.freight_value),2) as avg_freight_value,
  round(avg(date_diff(ord.order_delivered_customer_date, ord.order_purchase_timestamp, DAY)),2) as mean_time_to
_delivery,
  round(avg(date_diff(ord.order_estimated_delivery_date, ord.order_delivered_customer_date, DAY)),2) as mean_diff_
estimated_delivery
  from `s-business-case-target-sql.TargetSQL.orders` ord
    join `s-business-case-target-sql.TargetSQL.order_items` ordItms on ord.order_id = ordItms.order_id
  join `s-business-case-target-sql.TargetSQL.customers` cust on ord.customer_id = cust.customer_id
  group by cust.customer_state order by cust.customer_state;
```

Row	customer_state	avg_freight_value	mean_time_to_delivery	mean_diff_estimated_delivery
1	AC	40.07	20.33	20.01
2	AL	35.84	23.99	7.98
3	AM	33.21	25.96	18.98
4	AP	34.01	27.75	17.44
5	BA	26.36	18.77	10.12
6	CE	32.71	20.54	10.26
7	DF	21.04	12.5	11.27
8	ES	22.06	15.19	9.77
9	GO	22.77	14.95	11.37
10	MA	38.26	21.2	9.11
11	MG	20.63	11.52	12.4

5.4.1 Top 5 states by highest/lowest average freight value

Top 5 states by freight value:

```
select cust.customer_state as Top5_states, avg(ordItms.freight_value) as avg_freight_value,
  from `s-business-case-target-sql.TargetSQL.orders` ord
    join `s-business-case-target-sql.TargetSQL.order_items` ordItms on ord.order_id = ordItms.order_id
    join `s-business-case-target-sql.TargetSQL.customers` cust on ord.customer_id = cust.customer_id
  group by cust.customer_state order by avg_freight_value desc limit 5;
```

Row	Top5_states	avg_freight_valu
1	RR	42.9844230...
2	PB	42.7238039...
3	RO	41.0697122...
4	AC	40.0733695...
5	PI	39.1479704...

Bottom 5 states by freight value :

```
select cust.customer_state as Bottom5_states, avg(ordItms.freight_value) as avg_freight_value,
  from `s-business-case-target-sql.TargetSQL.orders` ord
    join `s-business-case-target-sql.TargetSQL.order_items` ordItms on ord.order_id = ordItms.order_id
    join `s-business-case-target-sql.TargetSQL.customers` cust on ord.customer_id = cust.customer_id
  group by cust.customer_state order by avg_freight_value asc limit 5;
```

Row	Bottom5_states	avg_freight_valu
1	SP	15.1472753...
2	PR	20.5316515...
3	MG	20.6301668...
4	RJ	20.9609239...
5	DF	21.0413549...

5.4.2 Top 5 states by highest/lowest average time to delivery

Top 5 states:

```
select cust.customer_state as Top5_states,
round(avg(date_diff(ord.order_delivered_customer_date, or
d.order_purchase_timestamp, DAY)),2) as mean_time_to_de
livery
  from `s-business-case-target-sql.TargetSQL.orders` ord
    join `s-business-case-target-
sql.TargetSQL.order_items` ordItms on ord.order_id = ordItm
s.order_id
    join `s-business-case-target-
sql.TargetSQL.customers` cust on ord.customer_id = cust.cu
stomer_id
  group by cust.customer_state order by mean_time_to_deliv
ery desc limit 5;
```

Row	Top5_states	mean_time_to_delivery
1	RR	27.83
2	AP	27.75
3	AM	25.96
4	AL	23.99
5	PA	23.3

Bottom 5 states:

```
select cust.customer_state as Bottom5_states,
round(avg(date_diff(ord.order_delivered_customer_date, or
d.order_purchase_timestamp, DAY)),2) as mean_time_to_de
livery
  from `s-business-case-target-sql.TargetSQL.orders` ord
    join `s-business-case-target-sql.TargetSQL.order_
items` ordItms on ord.order_id = ordItms.order_id
    join `s-business-case-target-
sql.TargetSQL.customers` cust on ord.customer_id = cust.cu
stomer_id
  group by cust.customer_state order by mean_time_to_deliv
ery asc limit 5;
```

Row	Bottom5_states	mean_time_to_d
1	SP	8.26
2	PR	11.48
3	MG	11.52
4	DF	12.5
5	SC	14.52

5.4.2 Top 5 states by fastest/slowest delivery w.r.t. estimated date

Top 5 states:

```
select cust.customer_state as Fastest_delivery_states,
  round(avg(date_diff(ord.order_estimated_delivery_date, ord
.order_delivered_customer_date, DAY)),2) as mean_diff_esti
mated_delivery
  from `s-business-case-target-sql.TargetSQL.orders` ord
    join `s-business-case-target-
sql.TargetSQL.order_items` ordItms on ord.order_id = ordItm
s.order_id
    join `s-business-case-target-
sql.TargetSQL.customers` cust on ord.customer_id = cust.cu
stomer_id
  group by cust.customer_state order by mean_diff_estimated
_delivery desc limit 5;
```

Row	Fastest_delivery_states	mean_diff_estimated_delivery
1	AC	20.01
2	RO	19.08
3	AM	18.98
4	AP	17.44
5	RR	17.43

Bottom 5 states:

```
select cust.customer_state as Slowest_delivery_states,
  round(avg(date_diff(ord.order_estimated_delivery_date, ord
.order_delivered_customer_date, DAY)),2) as mean_diff_esti
mated_delivery
  from `s-business-case-target-sql.TargetSQL.orders` ord
    join `s-business-case-target-
sql.TargetSQL.order_items` ordItms on ord.order_id = ordItm
s.order_id
    join `s-business-case-target-
sql.TargetSQL.customers` cust on ord.customer_id = cust.cu
stomer_id
  group by cust.customer_state order by mean_diff_estimated
_delivery asc limit 5;
```

Row	Slowest_delivery_states	mean_diff_estimated_delivery
1	AL	7.98
2	MA	9.11
3	SE	9.17
4	ES	9.77
5	BA	10.12

mean_diff_estimated_delivery

6. Payment Type Analysis

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

Query:

```
select concat(cast(extract(year from ord.order_purchase_timestamp) as string),
      LPAD(cast(extract(month from ord.order_purchase_timestamp) as string), 2,'0') ) as yearmonth,
      pymts.payment_type, count(ord.order_id) as count_of_orders
  from `s-business-case-target-sql.TargetSQL.orders` ord
   join `s-business-case-target-sql.TargetSQL.payments` pymts on ord.order_id = pymts.order_id
  group by yearmonth, pymts.payment_type
  order by yearmonth, pymts.payment_type;
```

Observations: From 2016 to 2018, payments are increasingly made using UPI & Credit Card. Thus, clearly customers preferences are increasing for instant payments (UPI) and delayed payments (Credit card). Thus, Target business should tie-up with banks to offer cashbacks on such payments, to increase sales further

Row	yearmonth	payment_type	count_of_orders
1	201609	credit_card	3
2	201610	UPI	63
3	201610	credit_card	254
4	201610	debit_card	2
5	201610	voucher	23
6	201612	credit_card	1
7	201701	UPI	197
8	201701	credit_card	583
9	201701	debit_card	9
10	201701	voucher	61

6.2. Distribution of payment instalments and count of orders

Query:

```
select payment_installments, count(order_id) as count_of_orders
  from `s-business-case-target-sql.TargetSQL.payments`
  group by payment_installments
```

Observations:

- 2 suspicious records where payment_installments is recorded as 0. Need to investigate this.
- Most payments are of single-instalment payment.
- There is special preference for 10-instalments option too (used in 5328 orders)

Row	payment_install	count_of_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
11	10	5328

Thank you!