BUSINESS CASE: TARGET SQL

1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset
    1. Data type of columns in a table : Available data types are written next to each column

- customers (String, Integer)

Field name	Type	Mode	Collation	Default Value	Policy Tags	Description
customer_id	STRING	NULLABLE				
customer_unique_id	STRING	NULLABLE				
customer_zip_code_prefix	INTEGER	NULLABLE				
customer_city	STRING	NULLABLE				
customer_state	STRING	NULLABLE				

- geolocation(String, Integer, Float)

SCHEMA DETAILS PREVIEW LINEAGE

Filter Enter property name or value

Field name	Type	Mode	Collation	Default Value	Policy Tags	Description
geolocation_zip_code_prefix	INTEGER	NULLABLE				
geolocation_lat	FLOAT	NULLABLE				
geolocation_lng	FLOAT	NULLABLE				
geolocation_city	STRING	NULLABLE				
geolocation_state	STRING	NULLABLE				

- order_items(String, Integer, Float, Timestamp)

Filter Enter property name or value

Field name	Type	Mode	Collation	Default Value	Policy Tags	Description
order_id	STRING	NULLABLE				
order_item_id	INTEGER	NULLABLE				
product_id	STRING	NULLABLE				
seller_id	STRING	NULLABLE				
shipping_limit_date	TIMESTAMP	NULLABLE				
price	FLOAT	NULLABLE				
freight_value	FLOAT	NULLABLE				

- order_reviews(String, Integer, Timestamp)

SCHEMA DETAILS PREVIEW LINEAGE

Filter Enter property name or value

Field name	Type	Mode	Collation	Default Value	Policy Tags	Description
review_id	STRING	NULLABLE				
order_id	STRING	NULLABLE				
review_score	INTEGER	NULLABLE				
review_comment_title	STRING	NULLABLE				
review_creation_date	TIMESTAMP	NULLABLE				
review_answer_timestamp	TIMESTAMP	NULLABLE				

BUSINESS CASE: TARGET SQL

- orders(String, Timestamp)

Filter Enter property name or value

Field name	Type	Mode	Collation	Default Value	Policy Tags
order_id	STRING	NULLABLE			
customer_id	STRING	NULLABLE			
order_status	STRING	NULLABLE			
order_purchase_timestamp	TIMESTAMP	NULLABLE			
order_approved_at	TIMESTAMP	NULLABLE			
order_delivered_carrier_date	TIMESTAMP	NULLABLE			
order_delivered_customer_date	TIMESTAMP	NULLABLE			
order_estimated_delivery_date	TIMESTAMP	NULLABLE			

- payments(String, Integer, Float)

Filter Enter property name or value

Field name	Type	Mode	Collation	Default Value	Policy Tags
order_id	STRING	NULLABLE			
payment_sequential	INTEGER	NULLABLE			
payment_type	STRING	NULLABLE			
payment_installments	INTEGER	NULLABLE			
payment_value	FLOAT	NULLABLE			

- products(String, Integer)

Filter Enter property name or value

Field name	Type	Mode	Collation	Default Value	Policy Tags
product_id	STRING	NULLABLE			
product_category	STRING	NULLABLE			
product_name_length	INTEGER	NULLABLE			
product_description_length	INTEGER	NULLABLE			
product_photos_qty	INTEGER	NULLABLE			
product_weight_g	INTEGER	NULLABLE			
product_length_cm	INTEGER	NULLABLE			
product_height_cm	INTEGER	NULLABLE			
product_width_cm	INTEGER	NULLABLE			

- sellers(String, Integer)

Filter Enter property name or value

Field name	Type	Mode	Collation	Default Value	Policy Tags
seller_id	STRING	NULLABLE			
seller_zip_code_prefix	INTEGER	NULLABLE			
seller_city	STRING	NULLABLE			
seller_state	STRING	NULLABLE			

BUSINESS CASE: TARGET SQL

2. Time period for which the data is given (2016-09-04 – 2018-10-17)

```
SELECT min(order_purchase_timestamp) FROM `my-project-
384902.Target_SQL_Business_Case.orders`;
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRA

Row	f0_
1	2016-09-04 21:15:19 UTC

```
SELECT max(order_purchase_timestamp) FROM `my-project-
384902.Target_SQL_Business_Case.orders`;
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION G

Row	f0_
1	2018-10-17 17:30:18 UTC

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

```
select c.customer_id, c.customer_city, c.customer_state
from `Target_SQL_Business_Case.customers` c
join `Target_SQL_Business_Case.orders` o on c.customer_id = o.customer_id
where  order_purchase_timestamp between "2016-09-04" and "2018-10-17";
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUT

Row	customer_id	customer_city	customer_state
1	5fc4c97dcb63903f996714524...	maceio	AL
2	a5c8228ef32a5a250903b18c0...	aracaju	SE
3	670af30ca5b8c20878fecdafa5...	aracaju	SE
4	5351c1e4ae199735063d6406c...	maceio	AL
5	5b54155ba8103b1bb1e157edc...	teresina	PI
6	1318775058e4321f5018e2fe4...	pau d'arco	AL
7	9c4efecd1866c2177998d461b...	natal	RN
8	84cb4824ee3f6d0c24b60d12a...	teresina	PI
9	6143e5df1b61e9568a5f02adb...	sao joao do piaui	PI
10	de270dbea5d94e6436d84456...	boquim	SE

BUSINESS CASE: TARGET SQL

2. In-depth Exploration:
    1. Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario? Can we see some seasonality with peaks at specific months?

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

```
select order_id,
customer_id,
order_status,
order_purchase_timestamp,
case
    when extract(hour from order_purchase_timestamp) between 0 and 6 then "Dawn"
    when extract(hour from order_purchase_timestamp) between 7 and 12 then "Morning"
    when extract(hour from order_purchase_timestamp) between 13 and 18 then "Afternoon"
    else "Night"
end as time
from `my-project-384902.Target_SQL_Business_Case.orders`;
```

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

Row	order_id	customer_id	order_status	order_purchase_timestamp	time
1	35de4050331c6c644cddc86f4...	4ee64f4bfc542546f422da0aeb...	created	2017-12-05 01:07:58 UTC	Dawn
2	b5359909123fa03c50bdb0cfe...	438449d4af8980d107bf04571...	created	2017-12-05 01:07:52 UTC	Dawn
3	f247ddbea2a3d9e88b688d08f...	ba9aef23b79c16ffd75994ead6...	shipped	2018-01-16 01:25:39 UTC	Dawn
4	0927a99c4c0b6c6b2487a4e86...	4635c796ac0535050e1eac3e0...	shipped	2017-12-07 00:02:16 UTC	Dawn
5	4041bc9d1b1fce0b58abecb80...	58803ced08e52f5e3b028ef81...	shipped	2018-03-05 03:47:11 UTC	Dawn
6	553379bf1f6c95489ae135f973...	aeb0f0b842c6516b8ec3fd2d1...	shipped	2017-04-17 00:10:48 UTC	Dawn
7	571a8f73010bc659b5a692e9d...	06e3ab9b654241d5b8d94c4b...	shipped	2018-05-07 01:24:47 UTC	Dawn
8	39841bda2ced6b58fb455964b...	085ee5f8069c949eb5e31d565...	shipped	2017-06-18 00:27:16 UTC	Dawn
9	07b9210fa8f57b996c79cb931...	fd53366df6136213ab393be0c...	shipped	2018-01-25 03:24:05 UTC	Dawn
10	a8096d0b94faa1314dbd562ef...	45ead47bd4ee6bb5b5108506c...	shipped	2018-01-18 04:37:44 UTC	Dawn

```
select count(time) from
(select order_id,
customer_id,
order_status,
order_purchase_timestamp,
case
    when extract(hour from order_purchase_timestamp) between 0 and 6 then "Dawn"
    when extract(hour from order_purchase_timestamp) between 7 and 12 then "Morning"
    when extract(hour from order_purchase_timestamp) between 13 and 18 then "Afternoon"
    else "Night"
end as time
from `my-project-384902.Target_SQL_Business_Case.orders`) t
where time = "Afternoon";
```

It seems Brazilian customers tend to buy more during Afternoon

time	Count of purchases
Dawn	5242
Morning	27733
Afternoon	38135
Night	28331

BUSINESS CASE: TARGET SQL

3. Evolution of E-commerce orders in the Brazil region:
    1. Get month on month orders by states

```
select c.customer_state,
count(order_purchase_timestamp) order_count, extract(month from order_purchase_timesta
mp) month,extract(year from order_purchase_timestamp) year
from `Target_SQL_Business_Case.customers` c
join `Target_SQL_Business_Case.orders` o on c.customer_id = o.customer_id
group by year,month, c.customer_state;

# It represents no of orders for each month over the years and for each state
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUT

Row	customer_state	order_count	month	year
1	RN	46	1	2018
2	RN	30	12	2017
3	RN	17	5	2017
4	CE	88	2	2018
5	CE	98	3	2018
6	CE	62	5	2017
7	CE	43	4	2017
8	CE	74	5	2018
9	RS	418	3	2018
10	RS	305	6	2018
11	SC	159	8	2017

```
select c.customer_state,
count(order_purchase_timestamp) order_count, extract(month from order_purchase_timesta
mp) month
from `Target_SQL_Business_Case.customers` c
join `Target_SQL_Business_Case.orders` o on c.customer_id = o.customer_id
group by month, c.customer_state
order by c.customer_state;

# It represents no of orders for each month(irrespective of years) and for each state
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

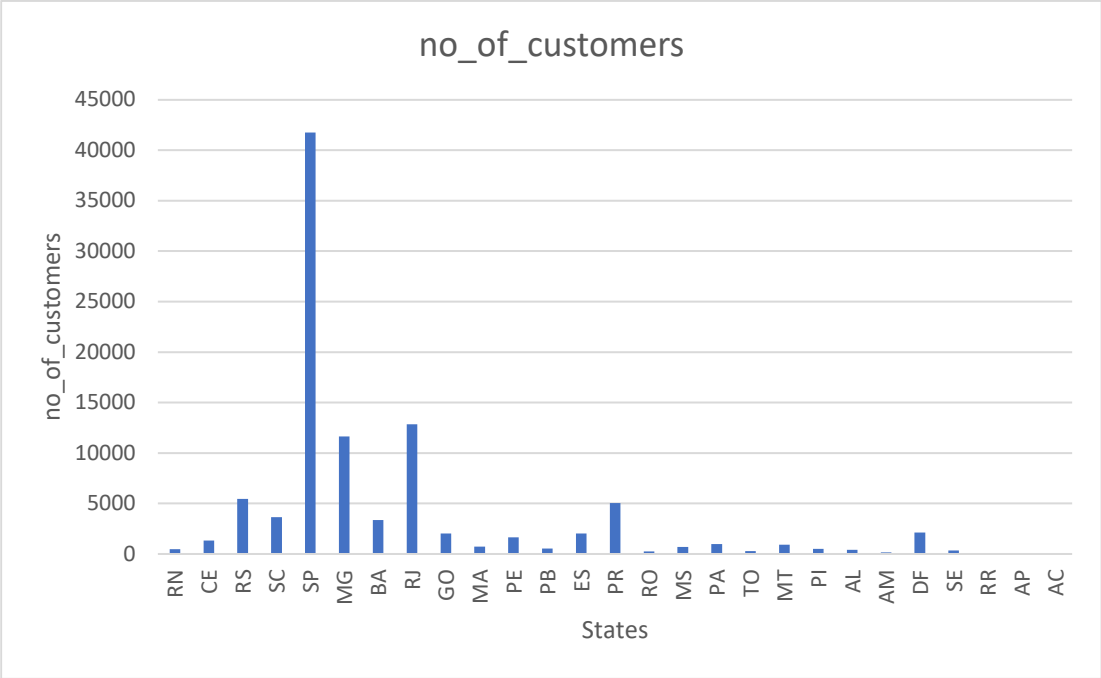
Row	customer_state	order_count	month
1	AC	5	11
2	AC	9	4
3	AC	6	2
4	AC	7	6
5	AC	7	8
6	AC	10	5
7	AC	4	3
8	AC	8	1
9	AC	9	7
10	AC	6	10

2. Distribution of customers across the states in Brazil

```
select c.customer_state,
count(c.customer_id) as no_of_customers
from `Target_SQL_Business_Case.customers` c
group by c.customer_state;
```

JOB INFORMATION | RESULTS | JSON | E

Row	customer_state	no_of_customer
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
11	PE	1652



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

```
select *
from
(select sum(p.payment_value) payment_value,extract(year from o.order_purchase_time
stamp) year, extract(month from o.order_purchase_timestamp) month
from `Target_SQL_Business_Case.orders` o
join `Target_SQL_Business_Case.payments` p on o.order_id = p.order_id
group by year, month) t
where month in (1,2,3,4,5,6,7,8) and year in (2018)
order by month, year;

select *
from
(select sum(p.payment_value) payment_value,extract(year from o.order_purchase_time
stamp) year, extract(month from o.order_purchase_timestamp) month
from `Target_SQL_Business_Case.orders` o
join `Target_SQL_Business_Case.payments` p on o.order_id = p.order_id
group by year, month) t
where month in (1,2,3,4,5,6,7,8) and year in (2017)
order by month, year;
```

payment_v	year	month	payment_v	year	month	%increase
1115004	2018	1	138488	2017	1	7.051267
992463.3	2018	2	291908	2017	2	2.399918
1159652	2018	3	449863.6	2017	3	1.577786
1160785	2018	4	417788	2017	4	1.778408
1153982	2018	5	592918.8	2017	5	0.946273
1023880	2018	6	511276.4	2017	6	1.002597
1066541	2018	7	592382.9	2017	7	0.800425
1022425	2018	8	674396.3	2017	8	0.51606

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

```
select round(avg(oi.price)) avg_price,round(sum(oi.price)) sum_price, round(avg(fr
eight_value)) avg_freight_value, round(sum(freight_value)) sum_freight_value
from `Target_SQL_Business_Case.order_items` oi
join `Target_SQL_Business_Case.orders` o on o.order_id = oi.order_id
join `Target_SQL_Business_Case.customers` c on c.customer_id = o.customer_id
group by c.customer_state
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	avg_price	sum_price	avg_freight_value	sum_freight_value
1	148.0	156454.0	28.0	29715.0
2	145.0	119648.0	38.0	31524.0
3	181.0	80315.0	36.0	15915.0
4	110.0	5202955.0	15.0	718723.0
5	121.0	1585308.0	21.0	270853.0
6	146.0	262788.0	33.0	59450.0
7	125.0	1824093.0	21.0	305589.0
8	126.0	302604.0	21.0	50625.0
9	120.0	750304.0	22.0	135523.0
10	153.0	58921.0	37.0	14111.0

5. Analysis on sales, freight and delivery time

1. Calculate days between purchasing, delivering and estimated delivery

```
select order_id, date_diff(order_delivered_customer_date,order_purchase_timestamp,d
ay) days_between_purchase_delivery,
date_diff(order_delivered_customer_date, order_estimated_delivery_date,day) days_be
tween_delivery_estimated_delivery,
from `Target_SQL_Business_Case.orders` o;
# some orders are after and before estimated dates, only few have reached on time
```

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

Row	order_id	days_between_purchase_delivery	days_between_delivery_estimated_delivery
1	1950d777989f6a877539f5379...	30	12
2	2c45c33d2f9cb8ff8b1c86cc28...	30	-28
3	65d1e226dfaeb8cdc42f66542...	35	-16
4	635c894d068ac37e6e03dc54e...	30	-1
5	3b97562c3aee8bdedcb5c2e45...	32	0
6	68f47f50f04c4cb6774570cfde...	29	-1
7	276e9ec344d3bf029ff83a161c...	43	4
8	54e1a3c2b97fb0809da548a59...	40	4

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

```
select order_id, date_diff(order_delivered_customer_date,order_purchase_timestam
p,day) time_to_delivery,
date_diff(order_delivered_customer_date, order_estimated_delivery_date,day) diff
_estimated_delivery
from `Target_SQL_Business_Case.orders` o;
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	order_id	time_to_delivery	diff_estimated_delivery
1	1950d777989f6a877539f5379...	30	12
2	2c45c33d2f9cb8ff8b1c86cc28...	30	-28
3	65d1e226dfaeb8cdc42f66542...	35	-16
4	635c894d068ac37e6e03dc54e...	30	-1
5	3b97562c3aee8bdedcb5c2e45...	32	0
6	68f47f50f04c4cb6774570cfde...	29	-1
7	276e9ec344d3bf029ff83a161c...	43	4
8	54e1a3c2b97fb0809da548a59...	40	4
9	fd04fa4105ee8045f6a0139ca5...	37	1
10	302bb8109d097a9fc6e9cefc5...	33	5

BUSINESS CASE: TARGET SQL

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

```
select c.customer_state, round(avg(freight_value)) avg_freight_value, round(avg(da
te_diff(order_delivered_customer_date,order_purchase_timestamp,day))) avg_time_to_
delivery,
round(avg(date_diff(order_delivered_customer_date, order_estimated_delivery_date,d
ay))) avg_diff_estimated_delivery
from `Target_SQL_Business_Case.orders` o
join `Target_SQL_Business_Case.order_items` oi on o.order_id = oi.order_id
join `Target_SQL_Business_Case.customers` c on c.customer_id = o.customer_id
group by c.customer_state;
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION

Row	customer_state	avg_freight_value	avg_time_to_delivery	avg_diff_estimated_delivery
1	MT	28.0	18.0	-14.0
2	MA	38.0	21.0	-9.0
3	AL	36.0	24.0	-8.0
4	SP	15.0	8.0	-10.0
5	MG	21.0	12.0	-12.0
6	PE	33.0	18.0	-13.0
7	RJ	21.0	15.0	-11.0
8	DF	21.0	13.0	-11.0
9	RS	22.0	15.0	-13.0
10	SE	37.0	21.0	-9.0

```
select * from
(select c.customer_state, round(avg(freight_value)) avg_freight_value, round(avg(da
te_diff(order_delivered_customer_date,order_purchase_timestamp,day))) avg_time_to_d
elivery,
round(avg(date_diff(order_delivered_customer_date, order_estimated_delivery_date,da
y))) avg_diff_estimated_delivery
from `Target_SQL_Business_Case.orders` o
join `Target_SQL_Business_Case.order_items` oi on o.order_id = oi.order_id
join `Target_SQL_Business_Case.customers` c on c.customer_id = o.customer_id
group by c.customer_state) temp
order by temp.avg_freight_value desc
limit 5;
```

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

Row	customer_state	avg_freight_value	avg_time_to_delivery	avg_diff_estimated_delivery
1	PB	43.0	20.0	-12.0
2	RR	43.0	28.0	-17.0
3	RO	41.0	19.0	-19.0
4	AC	40.0	20.0	-20.0
5	PI	39.0	19.0	-11.0

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

```
select * from
(select c.customer_state, round(avg(freight_value)) avg_freight_value, round(avg(da
te_diff(order_delivered_customer_date,order_purchase_timestamp,day))) avg_time_to_d
elivery,
round(avg(date_diff(order_delivered_customer_date, order_estimated_delivery_date,da
y))) avg_diff_estimated_delivery
from `Target_SQL_Business_Case.orders` o
join `Target_SQL_Business_Case.order_items` oi on o.order_id = oi.order_id
join `Target_SQL_Business_Case.customers` c on c.customer_id = o.customer_id
group by c.customer_state) temp
order by temp.avg_time_to_delivery desc
limit 5;
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION G

Row	customer_state	avg_freight_value	avg_time_to_delivery	avg_diff_estimated_delivery
1	RR	43.0	28.0	-17.0
2	AP	34.0	28.0	-17.0
3	AM	33.0	26.0	-19.0
4	AL	36.0	24.0	-8.0
5	PA	36.0	23.0	-13.0

```
select * from
(select c.customer_state,
date_diff(order_delivered_customer_date, order_estimated_delivery_date,day) diff_es
timated_delivery
from `Target_SQL_Business_Case.orders` o
join `Target_SQL_Business_Case.order_items` oi on o.order_id = oi.order_id
join `Target_SQL_Business_Case.customers` c on c.customer_id = o.customer_id) temp
order by temp.diff_estimated_delivery desc
limit 5;
```

JOB INFORMATION | RESULTS | JSON

Row	customer_state	diff_estimated_delivery
1	RJ	188
2	ES	181
3	SP	175
4	SP	167
5	SE	166

6. Payment type analysis:

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

```
select p.payment_type, count(o.order_purchase_timestamp) order_count, extract(mont
h from o.order_estimated_delivery_date) month,
extract(year from o.order_estimated_delivery_date) year
from `Target_SQL_Business_Case.orders` o
join `Target_SQL_Business_Case.payments` p on p.order_id = o.order_id
group by month,year, payment_type
order by year, month, payment_type
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXE

Row	payment_type	order_count	month	year
1	credit_card	1	9	2016
2	UPI	1	10	2016
3	credit_card	3	10	2016
4	voucher	1	10	2016
5	UPI	25	11	2016
6	credit_card	142	11	2016
7	debit_card	2	11	2016
8	voucher	16	11	2016
9	UPI	36	12	2016

```
select p.payment_installments, count(o.order_purchase_timestamp) order_count, extr
act(month from o.order_estimated_delivery_date) month,
extract(year from o.order_estimated_delivery_date) year
from `Target_SQL_Business_Case.orders` o
join `Target_SQL_Business_Case.payments` p on p.order_id = o.order_id
group by month,year, payment_installments
order by year, month, payment_installments
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	payment_installments	order_count	month	year
1	2	1	9	2016
2	1	4	10	2016
3	3	1	10	2016
4	1	76	11	2016
5	2	17	11	2016
6	3	25	11	2016
7	4	12	11	2016
8	5	11	11	2016
9	6	11	11	2016
10	7	9	11	2016