1. SQL query
2. Screenshot of top 10 rows.
3. Insights
4. Recommendations
5. Assumptions

Answers of the Business Case Study : Target-SQL

```
#1.1  Data type of all columns in the "customers" table.

SELECT column_name, DATA_TYPE
from target_sql_bcs.INFORMATION_SCHEMA.COLUMNS
where table_schema = 'target_sql_bcs' and table_name = 'customers'
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	column_name	DATA_TYPE
1	customer_id	STRING
2	customer_unique_id	STRING
3	customer_zip_code_prefix	INT64
4	customer_city	STRING
5	customer_state	STRING

```
#1.2 Get the time range between which the orders were placed.

select
min(order_purchase_timestamp) as first_order,
max(order_purchase_timestamp) as last_order
from `target_sql_bcs.customers` join `target_sql_bcs.orders`
using (customer_id)
limit 1
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | CHA

w	first_order	last_order
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

Insights - Given is the time range of the first order and the last order.

```
#1.3 Count the Cities & States of customers who ordered during the given period.

select count(distinct customer_city) as count_of_cities,
count(distinct customer_state) as count_of_states
from `target_sql_bcs.customers` join `target_sql_bcs.orders`
using (customer_id)
```

JOB INFORMATION | RESULTS | JSO

Row	count_of_cities	count_of_states
1	4119	27

Insights - customers who ordered have a unique count of cities to be 4117 and unique count of states to be 27.

```
#2.1 Is there a growing trend in the no. of orders placed over the past years?
select month, count(*) total_orders
from
(select extract(month from order_purchase_timestamp) month
from `target_sql_bcs.orders` ) tbl
group by tbl.month
order by month
```

Row	month	total_orders
1	1	8026
2	2	8455
3	3	9825
4	4	9297
5	5	10516
6	6	9374
7	7	10256
8	8	10780
9	9	4284
10	10	4943
11	11	7475
12	12	5644

Insights - These are the number of orders per month over the past years which have shown some increase till 8th month, but after that there was a heavy decrease of orders.

Recommendations - maybe advertise more or set up the stores more where there are less stores so that no of orders can be increased.

#2.2 Can we see some kind of monthly seasonality in terms of the no. of orders being placed?

```
select month,count(*) total_orders
from
(select extract(month from order_purchase_timestamp) month, count(*) as no_of_orders
from `target_sql_bcs.orders`
group by order_purchase_timestamp ) tbl
group by tbl.month
order by month
```

Row	month	total_orders
1	1	8026
2	2	8455
3	3	9825
4	4	9297
5	5	10516
6	6	9374
7	7	10256
8	8	10780
9	9	4284
10	10	4943
11	11	7475
12	12	5644

Insights - by looking at the data, we can say that there were months where the orders were at peak namely 5th, 7th and 8th, where 8th month recorded the peak.

Recommendations - maybe advertise more or set up the stores more where there are less stores so that no of orders can be increased.

#2.3 During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)

```
select
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_of_the_day,
```

```
count(*) as count_of_the_orders
from `target_sql_bcs.orders`
group by time_of_the_day
```

time_of_the_day	count_of_the_orders
Morning	27733
Dawn	5242
Afternoon	38135
Night	28331

Insights - by looking at the results, we can say that the most orders were placed during the afternoon time.

```
#3.1 Get the month on month no. of orders placed in each state.

select customer_state state, extract(month from order_purchase_timestamp) month,
count(*) order_count
from `target_sql_bcs.customers` join `target_sql_bcs.orders`
using (customer_id)
group by state, month
order by state, month
```

state	month	order_count
AC	1	8
AC	2	6
AC	3	4
AC	4	9
AC	5	10
AC	6	7
AC	7	9
AC	8	7
AC	9	5
AC	10	6

Insights - This result shows the number of orders per month per state.

#3.2 How are the customers distributed across all the states?

```
select customer_state, count(distinct customer_id) customer_count
from `target_sql_bcs.customers`
group by customer_state
order by customer_state
```

customer_state	customer_count
AC	81
AL	413
AM	148
AP	68
BA	3380
CE	1336
DF	2140
ES	2033
GO	2020
MA	747
MG	11635

Insights - This result shows the number of customers per state.

#4.1 Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).
--You can use the "payment_value" column in the payments table to get the cost of orders.

```
with cte as
(select year, round(sum(monthly_revenue),2) as revenue
from
(select  round(sum(payment_value),2) as monthly_revenue,
extract(year from order_purchase_timestamp) year ,
extract(month from order_purchase_timestamp) as month
from `target_sql_bcs.payments` p join `target_sql_bcs.orders` o
using(order_id)
where extract(year from order_purchase_timestamp) in(2017, 2018)
and extract(month from order_purchase_timestamp) in (1,2,3,4,5,6,7,8)
group by year,month) tbl
group by year
order by year)
```

```sql
select year,
round((revenue - LAG (revenue) OVER (ORDER BY year))/LAG (revenue) OVER (ORDER BY
cte.year)*100,2) as percentage_growth
from cte
order by year
```

year	percentage_growth
2017	null
2018	136.98

Insights - So, in 2017 and 2018,from jan to aug month, 137 is the percentage growth in the cost of orders.

```sql
#4.2 Calculate the Total & Average value of order price for each state.

select customer_state, round(sum(price),2) as total_price, round(avg(price),2) as
avg_price
from `target_sql_bcs.customers` c join `target_sql_bcs.orders` o on c.customer_id =
o.customer_id
join `target_sql_bcs.order_items` oi on o.order_id = oi.order_id
group by customer_state
order by customer_state
```

customer_state	total_price	avg_price
AC	15982.95	173.73
AL	80314.81	180.89
AM	22356.84	135.5
AP	13474.3	164.32
BA	511349.99	134.6
CE	227254.71	153.76
DF	302603.94	125.77
ES	275037.31	121.91
GO	294591.95	126.27

Insight - Average price will be less than the total price, like it should be.
Recommendation - Marketing should be more in states where the total price is less as compared to other states.

```
#4.3 Calculate the Total & Average value of order freight for each state.

select customer_state, round(sum(freight_value),2) as total_freight,
round(avg(freight_value),2) as avg_freight

from `target_sql_bcs.customers` c join `target_sql_bcs.orders` o on c.customer_id =
o.customer_id
join `target_sql_bcs.order_items` oi on o.order_id = oi.order_id

group by customer_state
order by customer_state
```

customer_state	total_freight	avg_freight
AC	3686.75	40.07
AL	15914.59	35.84
AM	5478.89	33.21
AP	2788.5	34.01
BA	100156.68	26.36
CE	48351.59	32.71
DF	50625.5	21.04
ES	49764.6	22.06
GO	53114.98	22.77

Insights - Average will be less than the total freight value, like it should be.

```
#5.1 Find the no. of days taken to deliver each order from the order's purchase date
as delivery time.

select order_id, timestamp_diff(order_delivered_customer_date,
order_purchase_timestamp, day) time_to_deliver,
abs(timestamp_diff(order_estimated_delivery_date, order_delivered_customer_date, day))
diff_estimated_delivery

from `target_sql_bcs.orders`
where order_status = 'delivered'
order by order_id
```

Row	order_id	time_to_deliver	diff_estimated_delive
1	00010242fe8c5a6d1ba2dd792...	7	8
2	00018f77f2f0320c557190d7a1...	16	2
3	000229ec398224ef6ca0657da...	7	13
4	00024acbcdf0a6daa1e931b03...	6	5
5	00042b26cf59d7ce69dfabb4e...	25	15
6	00048cc3ae777c65dbb7d2a06...	6	14
7	00054e8431b9d7675808bcb8...	8	16
8	000576fe39319847cbb9d288c...	5	15
9	0005a1a1728c9d785b8e2b08...	9	0
10	0005f50442cb953dcd1d21e1f...	2	18
11	00061f2a7bc09da83e415a52d...	4	10

Insights - This result set shows the comparison between difference in estimated delivery time and actual time taken to be delivered for every order which is delivered.

Assumption - Only considered orders where status is delivered.

```
#5.2 Find out the top 5 states with the highest & lowest average freight value.

(select customer_state, round(avg(freight_value),2) as avg_value
from `target_sql_bcs.customers` c join `target_sql_bcs.orders` o on c.customer_id =
o.customer_id
join `target_sql_bcs.order_items` oi on o.order_id = oi.order_id
group by customer_state
order by avg_value desc
limit 5 )

UNION ALL

(select customer_state, round(avg(freight_value),2) as avg_value
from `target_sql_bcs.customers` c join `target_sql_bcs.orders` o on c.customer_id =
o.customer_id
join `target_sql_bcs.order_items` oi on o.order_id = oi.order_id
group by customer_state
order by avg_value
limit 5 )
```

Row	customer_state	avg_value
1	RR	42.98
2	PB	42.72
3	RO	41.07
4	AC	40.07
5	PI	39.15
6	SP	15.15
7	PR	20.53
8	MG	20.63
9	RJ	20.96
10	DF	21.04

Insights - This query gives the top 5 highest and top 5 lowest customer states based on their average freight value.

```
# 5.3 Find out the top 5 states with the highest & lowest average delivery time.

(select customer_state, round(avg(timestamp_diff(order_delivered_customer_date,
order_purchase_timestamp,day))) as avg_time
from `target_sql_bcs.customers` c join `target_sql_bcs.orders` o on c.customer_id =
o.customer_id
join `target_sql_bcs.order_items` oi on o.order_id = oi.order_id
group by customer_state
order by avg_time desc
limit 5 )

UNION ALL

(select customer_state, round(avg(timestamp_diff(order_delivered_customer_date,
order_purchase_timestamp,day))) as avg_time
from `target_sql_bcs.customers` c join `target_sql_bcs.orders` o on c.customer_id =
o.customer_id
join `target_sql_bcs.order_items` oi on o.order_id = oi.order_id
group by customer_state
order by avg_time
limit 5 )
```

Row	customer_state	avg_time
1	AP	28.0
2	RR	28.0
3	AM	26.0
4	AL	24.0
5	PA	23.0
6	SP	8.0
7	PR	11.0
8	MG	12.0
9	DF	13.0
10	RS	15.0

Insights - This query gives the top 5 highest and top 5 lowest customer states based on their average delivery time in days. In some states, it is reaching around 1 month which is a lot of a delivery time.
Recommendations - Recruit more delivery people so that delivery time can be less.

#5.4 Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.

```
with cte as
(select customer_state,
round(avg(timestamp_diff(order_delivered_customer_date, order_purchase_timestamp,
day))) order_delivery,
round(abs(avg((timestamp_diff(order_estimated_delivery_date,
order_delivered_customer_date, day))))) estimated_date_of_delivery

from `target_sql_bcs.orders` join `target_sql_bcs.customers`
using(customer_id)

where order_status = 'delivered'
group by customer_state )


select customer_state,
abs(estimated_date_of_delivery - order_delivery) as fast_delivery
from cte
order by fast_delivery desc
limit 5
```

Row	customer_state	fast_delivery
1	AL	16.0
2	RR	13.0
3	SE	12.0
4	MA	12.0
5	CE	11.0

Insights- this result shows top 5 states where order delivery is really fast as compared to the estimated date of delivery.

#6.1 Find the month on month no. of orders placed using different payment types.

```
select extract(year from order_purchase_timestamp) year, extract(month from
order_purchase_timestamp) month,
payment_type,
count(*) as no_of_orders

from `target_sql_bcs.orders` join `target_sql_bcs.payments`
using(order_id)

group by year, month, payment_type
order by year, month, payment_type
```

Row	year	month	payment_type	no_of_orders
1	2016	9	credit_card	3
2	2016	10	UPI	63
3	2016	10	credit_card	254
4	2016	10	debit_card	2
5	2016	10	voucher	23
6	2016	12	credit_card	1
7	2017	1	UPI	197
8	2017	1	credit_card	583
9	2017	1	debit_card	9
10	2017	1	voucher	61

Insights- it gives the count of the no. of orders placed using different payment methods in each month per year.

#6.2 Find the no. of orders placed on the basis of the payment installments that have been paid.

```
select payment_installments,
count(*) as no_of_orders

from `target_sql_bcs.orders` join `target_sql_bcs.payments`
using(order_id)
where payment_installments >= 1
group by payment_installments
order by payment_installments
```

Row	payment_installment	no_of_orders
1	1	52546
2	2	12413
3	3	10461
4	4	7098
5	5	5239
6	6	3920
7	7	1626
8	8	4268
9	9	644
10	10	5328
11	11	23
12	12	133
13	13	16

Insights- this shows the count of no. of orders placed based on the no. of payment installments where at least one installment has been successfully paid which indicates that most of the orders were paid in one installment only.