# TARGET BUSINESS CASE STUDY

1. Data type of all columns in the "customers" table.

| Field name | Type | Mode |
|---|---|---|
| customer_id | STRING | NULLABLE |
| customer_unique_id | STRING | NULLABLE |
| customer_zip_code_prefix | INTEGER | NULLABLE |
| customer_city | STRING | NULLABLE |
| customer_state | STRING | NULLABLE |

Inference: “customers” table consists of data with string and integer data types

## 2. Get the time range between which the orders were placed.

```
1  select
2    min(order_purchase_timestamp) as earliest_order,
3    max(order_purchase_timestamp) as latest_order
4  from sql-dsml-scaler-449919.Target_Business_Case_Study.orders
```

Press Alt+F1 for Accessibility Options.

Query results

SAVE RESULTS | OPEN IN

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

| Row | earliest_order | latest_order |
|---|---|---|
| 1 | 2016-09-04 21:15:19 UTC | 2018-10-17 17:30:18 UTC |

**Inference: The time range aligns with the time specification mentioned in the problem statement (2016-18)**

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

```
1  select
2    count(distinct customer_city) as city_count,
3    count(distinct customer_state) as state_count
4  from sql-dsml-scaler-449919.Target_Business_Case_Study.customers c join
   sql-dsml-scaler-449919.Target_Business_Case_Study.orders o on c.customer_id=o.
   customer_id
5  where
6    lower(o.order_status)<>'canceled' or
7    lower(o.order_status)<>'unavailable'
```

Press Alt+F1 for Accessibil

Query results

SAVE RESULTS | OPEN IN

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

| Row | city_count | state_count |
|---|---|---|
| 1 | 4119 | 27 |

**Inference: The customer base spans across 4119 cities in/and 27 states**

## 4. Is there a growing trend in the no. of orders placed over the past years?

```sql
select
  extract(year from order_purchase_timestamp) as year,
  count(order_id) as order_count
from sql-dsml-scaler-449919.Target_Business_Case_Study.orders
where
  lower(order_status)<>'canceled' or
  lower(order_status)<>'unavailable'
group by 1
order by 1
```

Press Alt+F1 for Accessibility Options.

Query results

SAVE RESULTS | OPEN IN

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

| Row | year | order_count |
|---|---|---|
| 1 | 2016 | 329 |
| 2 | 2017 | 45101 |
| 3 | 2018 | 54011 |

**Inference: Yes, there has been a growing trend in the number of orders placed between 2016 and 2018**

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

```sql
select
  extract(year from order_purchase_timestamp) as year,
  extract(month from order_purchase_timestamp) as month,
  count(order_id) as order_count
from sql-dsml-scaler-449919.Target_Business_Case_Study.orders
where
  lower(order_status)<>'canceled' or
  lower(order_status)<>'unavailable'
group by 1,2
order by 1,2
```

| Row | year | month | order_count |
|---|---|---|---|
| 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 |

**Inference: In terms of the order count, 'Target' in Brazil witnesses a cyclical growth (expansion, peak, contraction, and trough)**

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

a. 0-6 hrs : Dawn
b. 7-12 hrs : Mornings
c. 13-18 hrs : Afternoon
d. 19-23 hrs : Night

```sql
select
  count(a.order_id) as order_count,
  a.daytime
from
(
  select
    order_id,
    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'
      when extract(hour from order_purchase_timestamp) between 19 and 23 then 'NIGHT'
    end as daytime
  from sql-dsml-scaler-449919.Target_Business_Case_Study.orders
  where lower(order_status) = 'approved'
) a
group by a.daytime
```

Query results

SAVE RESULTS | OPEN IN

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

| Row | order_count | daytime |
|---|---|---|
| 1 | 1 | NIGHT |
| 2 | 1 | DAWN |

Inference: Maximum orders placed by Brazilian customers were during night and dawn

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

```
select
  distinct extract(month from o.order_purchase_timestamp) as MONTH,
  c.customer_state,
  extract(year from o.order_purchase_timestamp) as YEAR,
  count(order_id) as order_count
from sql-dsml-scaler-449919.Target_Business_Case_Study.orders o join sql-dsml-scaler-449919.Target_Business_Case_Study.
customers c on o.customer_id=c.customer_id
where
  lower(order_status)<>'canceled' or
  lower(order_status)<>'unavailable'
group by 1,customer_state,order_purchase_timestamp
order by 1,2
```

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

| Row | MONTH | customer_state | YEAR | order_count |
|---|---|---|---|---|
| 1 | 1 | AC | 2017 | 1 |
| 2 | 1 | AC | 2018 | 1 |
| 3 | 1 | AL | 2017 | 1 |
| 4 | 1 | AL | 2018 | 1 |
| 5 | 1 | AL | 2018 | 2 |
| 6 | 1 | AM | 2018 | 1 |
| 7 | 1 | AP | 2018 | 1 |
| 8 | 1 | BA | 2018 | 1 |
| 9 | 1 | BA | 2017 | 1 |
| 10 | 1 | CE | 2018 | 1 |
| 11 | 1 | CE | 2017 | 1 |
| 12 | 1 | DF | 2017 | 1 |
| 13 | 1 | DF | 2018 | 1 |
| 14 | 1 | ES | 2018 | 1 |
| 15 | 1 | ES | 2017 | 1 |
| 16 | 1 | ES | 2018 | 2 |

**Inference: The month-on-month track of orders' count remains fairly constant across majority of states**

## 8. How are the customers distributed across all the states?

```
select distinct
  customer_state,
  count(customer_id) as no_of_customers
from sql-dsml-scaler-449919.Target_Business_Case_Study.customers
group by 1
order by 2 desc
```

| Row | customer_state | no_of_customers |
|---|---|---|
| 1 | SP | 41746 |
| 2 | RJ | 12852 |
| 3 | MG | 11635 |
| 4 | RS | 5466 |
| 5 | PR | 5045 |
| 23 | RO | 253 |
| 24 | AM | 148 |
| 25 | AC | 81 |
| 26 | AP | 68 |
| 27 | RR | 46 |

**Inference: Of the 27 states, states of SP & RR have the maximum and the minimum customer bases respectively**

**9. 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
    extract(year from order_purchase_timestamp) as YEAR,
    sum(payment_value) as total_value
  from
    sql-dsml-scaler-449919.Target_Business_Case_Study.payments p
    join
    sql-dsml-scaler-449919.Target_Business_Case_Study.orders o
    on p.order_id=o.order_id
  where
    extract(year from order_purchase_timestamp) in (2017,2018)
    and
    extract(month from order_purchase_timestamp) in (1,8)
  group by extract(year from order_purchase_timestamp)
)
select
  max(case when YEAR=2018 then total_value end) as net_2018_cost,
  max(case when YEAR=2017 then total_value end) as net_2017_cost,
  round((max(case when YEAR=2018 then total_value end) - max(case when YEAR=2017 then total_value end))/max(case when YEAR=2017 then
total_value end)*100,2) as percent_change
from cte
```

| Row | net_2018_cost | net_2017_cost | percent_change |
|---|---|---|---|
| 1 | 2137429.500000... | 812884.3600000... | 162.94 |

**Inference: 2018 registered a 162.94% increase in sales value vis-à-vis 2017**

## 10. Calculate the Total & Average value of order price for each state.

```
select distinct
  customer_state,
  round(sum(payment_value),1) as TOTAL_ORDER_PRICE,
  round(avg(payment_value),1) as AVG_ORDER_PRICE,
from
  sql-dsml-scaler-449919.Target_Business_Case_Study.payments p
  join
  sql-dsml-scaler-449919.Target_Business_Case_Study.orders o
  on p.order_id=o.order_id
  join
  sql-dsml-scaler-449919.Target_Business_Case_Study.customers c
  on o.customer_id=c.customer_id
group by customer_state
order by 2 desc
```

| Row | customer_state | TOTAL_ORDER_PRICE | AVG_ORDER_PRICE |
|---|---|---|---|
| 1 | SP | 5998227.0 | 137.5 |
| 2 | RJ | 2144379.7 | 158.5 |
| 3 | MG | 1872257.3 | 154.7 |
| 25 | AC | 19680.6 | 234.3 |
| 26 | AP | 16262.8 | 232.3 |
| 27 | RR | 10064.6 | 218.8 |

**Inference: Maximum and minimum value of total order price is registered in the states of SP & RR respectively**

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

```
select
  customer_state,
  round(sum(freight_value),1) as TOTAL_FREIGHT_VALUE,
  round(avg(freight_value),1) as AVG_FREIGHT_VALUE
from
  sql-dsml-scaler-449919.Target_Business_Case_Study.customers c
  join
  sql-dsml-scaler-449919.Target_Business_Case_Study.orders o
  on c.customer_id=o.customer_id
  join
  sql-dsml-scaler-449919.Target_Business_Case_Study.order_items i
  on o.order_id=i.order_id
group by customer_state
order by 3 asc
```

| Row | customer_state | TOTAL_FREIGHT_VALUE | AVG_FREIGHT_VALUE |
|---|---|---|---|
| 1 | SP | 718723.1 | 15.1 |
| 2 | PR | 117851.7 | 20.5 |
| 3 | MG | 270853.5 | 20.6 |

**Inference: With the lowest average freight cost, SP is the most cost-effective state**

**12. Find the no. of days taken to deliver each order from the order's purchase date as delivery time. Also, calculate the difference (in days) between the estimated & actual delivery date of an order. Do this in a single query.**

**You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formulae:**

**A. time_to_deliver = order_delivered_customer_date - order_purchase_timestamp**

**B. diff_estimated_delivery = order_delivered_customer_date - order_estimated_delivery_date**

```
select
  order_id,
  order_purchase_timestamp,
  order_estimated_delivery_date,
  order_delivered_customer_date,
  date_diff(order_delivered_customer_date,order_purchase_timestamp,day) as time_to_deliver,
  date_diff(order_delivered_customer_date,order_estimated_delivery_date,day) as actual_and_estimated_diff
from sql-dsml-scaler-449919.Target_Business_Case_Study.orders
where order_delivered_customer_date is not null
order by order_purchase_timestamp
```

Press

Query results

SAVE RESULTS

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

| Row | order_id | order_purchase_timestamp | order_estimated_delivery_date | order_delivered_customer_date | time_to_deliver | actual_and_estimate |
|---|---|---|---|---|---|---|
| 1 | bfbd0f9bdef84302105ad712db... | 2016-09-15 12:16:38 UTC | 2016-10-04 00:00:00 UTC | 2016-11-09 07:47:38 UTC | 54 | 36 |
| 2 | 3b697a20d9e427646d925679... | 2016-10-03 09:44:50 UTC | 2016-10-27 00:00:00 UTC | 2016-10-26 14:02:13 UTC | 23 | 0 |
| 3 | be5bc2f0da14d8071e2d45451... | 2016-10-03 16:56:50 UTC | 2016-11-07 00:00:00 UTC | 2016-10-27 18:19:38 UTC | 24 | -10 |
| 4 | 65d1e226dfaeb8cdc42f66542... | 2016-10-03 21:01:41 UTC | 2016-11-25 00:00:00 UTC | 2016-11-08 10:58:34 UTC | 35 | -16 |
| 5 | a41c8759fbe7aab36ea07e038... | 2016-10-03 21:13:36 UTC | 2016-11-29 00:00:00 UTC | 2016-11-03 10:58:07 UTC | 30 | -25 |
| 6 | d207cc272675637bfed0062ed... | 2016-10-03 22:06:03 UTC | 2016-11-23 00:00:00 UTC | 2016-10-31 11:07:42 UTC | 27 | -22 |
| 7 | cd3b8574c82b42fc8129f6d50... | 2016-10-03 22:31:31 UTC | 2016-11-23 00:00:00 UTC | 2016-10-14 16:08:00 UTC | 10 | -39 |

**Inference: In majority cases, the orders were delivered before the expected delivery date**

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

```
(
select
  distinct customer_state as STATE, round(avg(freight_value),1) as AVG_FREIGHT_VALUE
from
    sql-dsml-scaler-449919.Target_Business_Case_Study.customers c
    join
    sql-dsml-scaler-449919.Target_Business_Case_Study.orders o
    on c.customer_id=o.customer_id
    join
    sql-dsml-scaler-449919.Target_Business_Case_Study.order_items i
    on o.order_id=i.order_id
group by customer_state
order by 2 desc
limit 5
)
union all
(
  select
  distinct customer_state as STATE, round(avg(freight_value),1) as AVG_FREIGHT_VALUE
from
    sql-dsml-scaler-449919.Target_Business_Case_Study.customers c
    join
    sql-dsml-scaler-449919.Target_Business_Case_Study.orders o
    on c.customer_id=o.customer_id
    join
    sql-dsml-scaler-449919.Target_Business_Case_Study.order_items i
    on o.order_id=i.order_id
group by customer_state
order by 2 asc
limit 5
)
```

JOB INFORMATION | RESULTS | CHART | JSON

| Row | STATE | AVG_FREIGHT_VALUE |
|---|---|---|
| 1 | SP | 15.1 |
| 2 | PR | 20.5 |
| 3 | MG | 20.6 |
| 4 | RJ | 21.0 |
| 5 | DF | 21.0 |
| 6 | RR | 43.0 |
| 7 | PB | 42.7 |
| 8 | RO | 41.1 |
| 9 | AC | 40.1 |
| 10 | PI | 39.1 |

**Inference:**

a. **Top 5 cost-effective states (with lowest values of average freight cost): SP>PR>MG>RJ>DF**
b. **Bottom 5 cost-effective states (with highest values of average freight cost): PI>AC>RO>PB>RR**

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

```
( select customer_state, round(avg(date_diff(order_delivered_customer_date,order_purchase_timestamp,day)),2) as AVG_DLVRY_TIME
from
  sql-dsml-scaler-449919.Target_Business_Case_Study.customers c
    join
    sql-dsml-scaler-449919.Target_Business_Case_Study.orders o
    on c.customer_id=o.customer_id
    join
    sql-dsml-scaler-449919.Target_Business_Case_Study.order_items i
    on o.order_id=i.order_id
where order_delivered_customer_date is not null
group by customer_state
order by 2 desc
limit 5 )
union all
( select customer_state, round(avg(date_diff(order_delivered_customer_date,order_purchase_timestamp,day)),2) as AVG_DLVRY_TIME
from
  sql-dsml-scaler-449919.Target_Business_Case_Study.customers c
    join
    sql-dsml-scaler-449919.Target_Business_Case_Study.orders o
    on c.customer_id=o.customer_id
    join
    sql-dsml-scaler-449919.Target_Business_Case_Study.order_items i
    on o.order_id=i.order_id
where order_delivered_customer_date is not null
group by customer_state
order by 2 asc
limit 5)
```

| Row | customer_state | AVG_DLVRY_TIME |
|---|---|---|
| 1 | RR | 27.83 |
| 2 | AP | 27.75 |
| 3 | AM | 25.96 |
| 4 | AL | 23.99 |
| 5 | PA | 23.3 |
| 6 | SP | 8.26 |
| 7 | PR | 11.48 |
| 8 | MG | 11.52 |
| 9 | DF | 12.5 |
| 10 | SC | 14.52 |

**Inference:**

a. **Top 5 states (with lowest values of average delivery time): SP>PR>MG>DF>SC**
b. **Bottom 5 states (with highest values of average delivery time): RR>AP>AM>AL>PA**

**15. Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery. You can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.**

```sql
select
  customer_state,
  round(avg(date_diff(order_delivered_customer_date,order_estimated_delivery_date,day)),2) as diff
from
  sql-dsml-scaler-449919.Target_Business_Case_Study.customers c
  join
  sql-dsml-scaler-449919.Target_Business_Case_Study.orders o
  on c.customer_id=o.customer_id
where order_delivered_customer_date is not null
group by customer_state
order by diff asc
limit 5
```

Query results

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

| Row | customer_state | diff |
|---|---|---|
| 1 | AC | -19.76 |
| 2 | RO | -19.13 |
| 3 | AP | -18.73 |
| 4 | AM | -18.61 |
| 5 | RR | -16.41 |

**Inference: The top 5 states where the order delivery faster as compared to the estimated delivery date are AC>RO>AP>AM>RR**

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

```sql
select distinct
  extract(month from order_purchase_timestamp) as MONTH,
  extract(year from order_purchase_timestamp) as YEAR,
  payment_type,
  count(o.order_id) order_count
from
  sql-dsml-scaler-449919.Target_Business_Case_Study.orders o
  join
  sql-dsml-scaler-449919.Target_Business_Case_Study.payments p
  on o.order_id=p.order_id
where order_delivered_customer_date is not null
group by
  payment_type,
  extract(month from order_purchase_timestamp),
  extract(year from order_purchase_timestamp)
order by 1,2,4 desc
```

| Row | MONTH | YEAR | payment_type | order_count |
|---|---|---|---|---|
| 1 | 1 | 2017 | credit_card | 542 |
| 2 | 1 | 2017 | UPI | 188 |
| 3 | 1 | 2017 | voucher | 60 |
| 4 | 1 | 2017 | debit_card | 9 |
| 5 | 1 | 2018 | credit_card | 5368 |
| 6 | 1 | 2018 | UPI | 1473 |
| 7 | 1 | 2018 | voucher | 401 |
| 8 | 1 | 2018 | debit_card | 109 |
| 9 | 2 | 2017 | credit_card | 1257 |
| 10 | 2 | 2017 | UPI | 371 |

**Inference: Majority of the payment modes register a steady month-on-month increase from 2016-18**

**17. Find the no. of orders placed on the basis of the payment instalments that have been paid.**

```
1  select
2    payment_installments,
3    count(order_id) as orders_count
4  from sql-dsml-scaler-449919.Target_Business_Case_Study.payments
5  group by payment_installments
6  order by 2 desc
```

Query results

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

| Row | payment_installment | orders_count |
|---|---|---|
| 1 | 1 | 52546 |
| 2 | 2 | 12413 |
| 3 | 3 | 10461 |
| 4 | 4 | 7098 |
| 5 | 10 | 5328 |
| 6 | 5 | 5239 |
| 7 | 8 | 4268 |
| 8 | 6 | 3920 |

**Inference: There's often a negative correlation between the number of payment instalments and the corresponding number of orders placed**