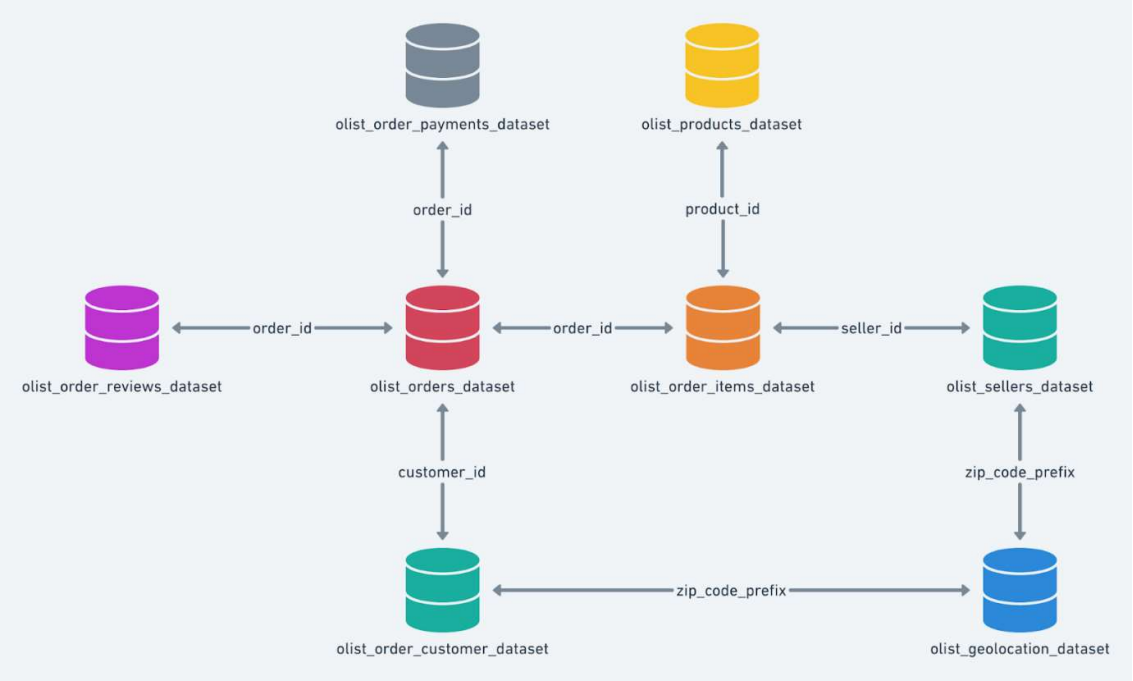
Business Case: Target SQL

Data from 100,000 orders placed at Target in Brazil between 2016 and 2018 is included in this business case. It is the top retail chain in America. Eight tables include data about orders from several aspects, including order status, payment information, order location and time, client who made the purchase, items in the order, product details, vendor information, order reviews, etc.



ANAYLISIS

1. Initial exploration of dataset

i. Data type of all columns of different tables in the "Target" dataset.

→→ *Query:*

```
SELECT
  table_schema,
  table_name,
  column_name,
  data_type
FROM
  `Target`.INFORMATION_SCHEMA.COLUMNS;
```

→→ *Result*:

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

Row	table_schema	table_name	column_name	data_type
1	Target	order_items	order_id	STRING
2	Target	order_items	order_item_id	INT64
3	Target	order_items	product_id	STRING
4	Target	order_items	seller_id	STRING
5	Target	order_items	shipping_limit_date	TIMESTAMP
6	Target	order_items	price	FLOAT64
7	Target	order_items	freight_value	FLOAT64
8	Target	sellers	seller_id	STRING
9	Target	sellers	seller_zip_code_prefix	INT64
10	Target	sellers	seller_city	STRING
11	Target	sellers	seller_state	STRING

ii. The timespan of available data.

→→ *Query:*

```
SELECT
  MIN(DATE(order_purchase_timestamp)) AS first_order_date,
  MAX(DATE(order_purchase_timestamp)) AS last_order_date
FROM
  `Target.orders`;
```

→→ *Result*:

Query results

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

Row	first_order_date	last_order_date
1	2016-09-04	2018-10-17

iii. Number of cities per State of customers who ordered during the given period.

→→ *Query:*

```
SELECT
  DISTINCT c.customer_state , count (c.customer_city) as No_of_Cities
FROM
  `Target.customers` c
RIGHT JOIN
  `Target.orders` o
ON
  c.customer_id = o.customer_id
GROUP BY c.customer_state
ORDER BY No_of_Cities DESC, c.customer_state;
```

→→ *Result*:

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

Row	customer_state	No_of_Cities
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
11	PE	1652

iv. Distribution of total orders as per their status.

→→ *Query:*

```
SELECT
  DISTINCT c.customer_state , count (c.customer_city) as No_of_Cities
FROM
```

```
  `Target.customers` c
RIGHT JOIN
  `Target.orders` o
ON
  c.customer_id = o.customer_id
GROUP BY c.customer_state
ORDER BY No_of_Cities DESC, c.customer_state;
```

→→ *Result*:

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

Row	order_status	order_count
1	delivered	96478
2	shipped	1107
3	canceled	625
4	unavailable	609
5	invoiced	314
6	processing	301
7	created	5
8	approved	2

2. In-Depth Analysis

i. Top 10 cities with largest customer base.

→→ *Query:*

```
SELECT
time_period, order_count,
ROUND((((order_count - LAG(order_count) OVER(ORDER BY year, month)) /
LAG(order_count) OVER(ORDER BY year, month))* 100), 2) AS growth_percent
FROM
(SELECT
EXTRACT (MONTH FROM order_purchase_timestamp) AS month,
EXTRACT (YEAR FROM order_purchase_timestamp) AS year,
FORMAT_DATE('%b %Y', DATE(order_purchase_timestamp)) AS time_period,COUNT(order_id)
AS order_count
FROM `Target.orders`
WHERE order_status = 'delivered'
GROUP BY month, year, time_period) as T1
ORDER BY year, month;
```

→→ *Result*:

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

Row	customer_city	No_Customers_City
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

---

ii. Top 10 cities with the greatest number of orders.

→→ *Query:*

```
SELECT
  Tempo.geolocation_city, COUNT(Tempo.geolocation_city) as Orders_per_City
FROM
  `Target.orders` as o
INNER JOIN
  (SELECT c.customer_id ,g.geolocation_city
  FROM
    `Target.customers` as c
  INNER JOIN
    `Target.geolocation` as g
  ON
    c.customer_zip_code_prefix = g.geolocation_zip_code_prefix) AS Tempo
ON
  Tempo.customer_id = o.customer_id
GROUP BY
  Tempo.geolocation_city
ORDER BY
  Orders_per_City DESC
LIMIT 10
```

→→ *Result*:

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

Row	geolocation_city	Orders_per_City
1	rio de janeiro	1913913
2	sao paulo	1164470
3	belo horizonte	737556
4	niteroi	393175
5	curitiba	255731
6	santos	238952
7	porto alegre	228803
8	são paulo	207882
9	campinas	170318
10	uberlandia	160498

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

→→ *Query:*

```
SELECT
time_period, order_count,
ROUND((((order_count - LAG(order_count) OVER(ORDER BY year, month)) /
LAG(order_count) OVER(ORDER BY year, month))* 100), 2) AS growth_percent
FROM
(SELECT
EXTRACT (MONTH FROM order_purchase_timestamp) AS month,
EXTRACT (YEAR FROM order_purchase_timestamp) AS year,
FORMAT_DATE('%b %Y', DATE(order_purchase_timestamp)) AS time_period,COUNT(order_id)
AS order_count
FROM `Target.orders`
WHERE order_status = 'delivered'
GROUP BY month, year, time_period) as T1
ORDER BY year, month;
```

→→ *Result*:

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

Row	time_period	order_count	growth_percent
1	Sep 2016	1	null
2	Oct 2016	265	26400.0
3	Dec 2016	1	-99.62
4	Jan 2017	750	74900.0
5	Feb 2017	1653	120.4
6	Mar 2017	2546	54.02
7	Apr 2017	2303	-9.54
8	May 2017	3546	53.97
9	Jun 2017	3135	-11.59
10	Jul 2017	3872	23.51

There is no concrete evidence to show any pattern that shows seasonality in orders placed. However, it is seen that there is growing trend in no. of orders placed over the past years.

iv. <u>What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)?</u>

→ → *Query:*

```
SELECT order_time, count(*) AS Total_Orders
FROM
(SELECT
    order_id,
    CASE
      WHEN TIME(order_purchase_timestamp) BETWEEN "00:00:00" AND "07:00:00" THEN
"Dawn"
      WHEN TIME(order_purchase_timestamp) BETWEEN "07:00:01" AND "12:00:00" THEN
"Morning"
      WHEN TIME(order_purchase_timestamp) BETWEEN "12:00:01" AND "18:00:00" THEN
"Afternoon"
      WHEN TIME(order_purchase_timestamp) BETWEEN "18:00:01" AND "23:59:59" THEN
"Night"
    END AS order_time
  FROM
    `Target.orders`) AS ORDER_SLOTS
GROUP BY order_time
ORDER BY Total_Orders DESC
```

→ → *Result*:

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

Row	order_time	Total_Orders
1	Afternoon	38365
2	Night	34096
3	Morning	21738
4	Dawn	5242

It is seen that the Brazilian customers tend to place most of their orders during afternoon and night.

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

i. Month on month orders by states

→→ *Query:*

```
SELECT
  state ,year, month, time_period , total_orders,
  LAG(total_orders) OVER(PARTITION BY state ORDER BY year, month ) AS
prev_month_orders_count,
  ROUND(((total_orders - LAG(total_orders) OVER(PARTITION BY state ORDER BY year,
month )) / LAG(total_orders) OVER(PARTITION BY state ORDER BY year, month))* 100,2)
AS MoM_percent_growth
FROM (
  SELECT
    state, time_period, year,month,
    COUNT(*) AS total_orders
  FROM (
    SELECT
      o.order_id, o.order_purchase_timestamp,
      EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
      EXTRACT(Month FROM order_purchase_timestamp) AS month,
      FORMAT_DATE('%b %Y', DATE(order_purchase_timestamp)) AS time_period,
      c.customer_state AS state
    FROM
      `Target.orders` o
    JOIN
      `Target.customers` c
    USING
      (customer_id)
    ORDER BY
       year, month) Tempo
  GROUP BY
    state, time_period,year, month) Tempo1
  ORDER BY state, year, month;
```

→→ *Result*:

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

Row	state	year	month	time_period	total_orders	prev_month_orders_c	MoM_percent_growt
1	AC	2017	1	Jan 2017	2	null	null
2	AC	2017	2	Feb 2017	3	2	50.0
3	AC	2017	3	Mar 2017	2	3	-33.33
4	AC	2017	4	Apr 2017	5	2	150.0
5	AC	2017	5	May 2017	8	5	60.0
6	AC	2017	6	Jun 2017	4	8	-50.0
7	AC	2017	7	Jul 2017	5	4	25.0
8	AC	2017	8	Aug 2017	4	5	-20.0
9	AC	2017	9	Sep 2017	5	4	25.0
10	AC	2017	10	Oct 2017	6	5	20.0
11	AC	2017	11	Nov 2017	5	6	-16.67

ii. Distribution of customers across states in Brazil

→ → *Query:*

```
SELECT
  customer_state AS State,
  COUNT(*) AS Total_Customers
FROM
  `Target.customers`
GROUP BY
  customer_state
ORDER BY
  total_customers DESC;
```

→ → *Result*:

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

Row	state	total_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

4. Impact on Economy: Analyzing the money movement by e-commerce by looking at order prices, freight and others.

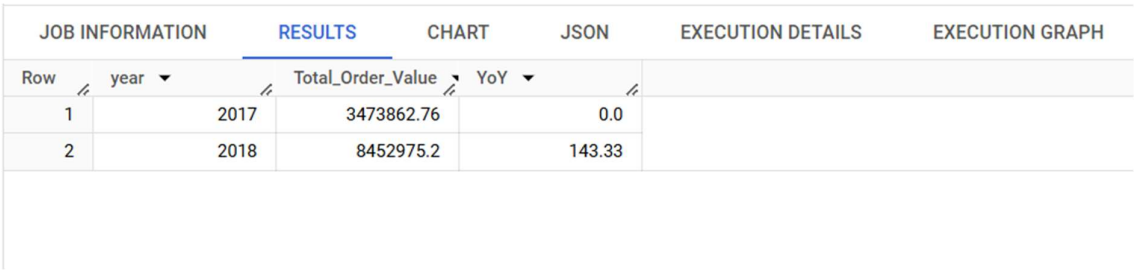
1. Percentage increase in the cost of orders from year 2017 to 2018 (including months between Jan to Aug only).

→ → *Query:*

```
SELECT
  *,
COALESCE((ROUND(((Total_Order_Value - LAG(Total_Order_Value) OVER(ORDER BY
year))/LAG(Total_Order_Value) OVER(ORDER BY year))* 100, 2)), 0) AS YoY
FROM (
  SELECT
    year,
    ROUND(SUM(payment_value), 2) AS Total_Order_Value
  FROM (
    SELECT
      o.order_id,
```

```
      o.order_purchase_timestamp,
      EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
      EXTRACT(YEAR FROM  o.order_purchase_timestamp) AS year,
      p.payment_value
    FROM
      `Target.orders` o
    INNER JOIN
      `Target.payments` p
    USING
      (order_id)
    WHERE
      o.order_status = "delivered") AS T
  WHERE
    month BETWEEN 1 AND 8
  GROUP BY year) AS T1
ORDER BY Year;
```

→ → *Result*:



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

Row	year	Total_Order_Value	YoY
1	2017	3473862.76	0.0
2	2018	8452975.2	143.33

2. Total & Average value of order price and freight value for each state.

→ → *Query:*

```
SELECT
  c.customer_state,
  ROUND(SUM(oi.price)) AS Total_Price,
  ROUND(AVG(oi.price)) AS Avg_Price,
  ROUND(SUM(oi.freight_value)) AS Total_Freight,
  ROUND(AVG(oi.freight_value)) AS Avg_Freight
FROM
  `Target.order_items` AS oi
INNER JOIN
  `Target.orders` o
ON
  oi.order_id = o.order_id
INNER JOIN
  `Target.customers` c
ON
  c.customer_id = o.customer_id
GROUP BY
  c.customer_state;
```

→ → *Result*:

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

Row	customer_state	Total_Price	Avg_Price	Total_Freight	Avg_Freight
1	SP	5202955.0	110.0	718723.0	15.0
2	RJ	1824093.0	125.0	305589.0	21.0
3	PR	683084.0	119.0	117852.0	21.0
4	SC	520553.0	125.0	89660.0	21.0
5	DF	302604.0	126.0	50625.0	21.0
6	MG	1585308.0	121.0	270853.0	21.0
7	PA	178948.0	166.0	38699.0	36.0
8	BA	511350.0	135.0	100157.0	26.0
9	GO	294592.0	126.0	53115.0	23.0
10	RS	750304.0	120.0	135523.0	22.0

5. Analysis based on sales, freight and delivery time.

1. Number of days taken to deliver each order from the order's purchase date as delivery time. Along with, the difference (in days) between the estimated & actual delivery date of an order.

→→ *Query:*

```
SELECT
    * ,T.Actual_Delivery_Time_in_Days - T.Estimated_Delivery_Time_in_Days AS
Diff_Estimated_Delivery
FROM
(SELECT
  order_id,
  TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp, day) AS
Actual_Delivery_Time_in_Days ,
  TIMESTAMP_DIFF(order_estimated_delivery_date, order_purchase_timestamp, day) AS
Estimated_Delivery_Time_in_Days,
FROM
  `Target.orders`
WHERE
  order_status = "delivered") AS T;
```

→ → *Result*:

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

Row	order_id	Actual_Delivery_Time	Estimated_Delivery_T	Diff_Estimated_Deliv
1	635c894d068ac37e6e03dc54e...	30	32	-2
2	3b97562c3aee8bdedcb5c2e45...	32	33	-1
3	68f47f50f04c4cb6774570cfde...	29	31	-2
4	276e9ec344d3bf029ff83a161c...	43	39	4
5	54e1a3c2b97fb0809da548a59...	40	36	4
6	fd04fa4105ee8045f6a0139ca5...	37	35	2
7	302bb8109d097a9fc6e9cefc5...	33	28	5
8	66057d37308e787052a32828...	38	32	6
9	19135c945c554eebfd7576c73...	36	33	3
10	4493e45e7ca1084efcd38ddeb...	34	33	1

2. Ranking states with the highest & lowest average freight value.

→ → *Query:*

```
SELECT
  State,Avg_Freight_Value,
  ROW_NUMBER() OVER (ORDER BY Avg_Freight_Value) AS Ranking_for_Lowest,
  ROW_NUMBER() OVER (ORDER BY Avg_Freight_Value DESC) AS Ranking_for_Highest
FROM
  (SELECT DISTINCT c.customer_state as State,
    AVG(oi.freight_value) OVER (PARTITION BY c.customer_state) AS Avg_Freight_Value
  FROM
    `Target.order_items` AS oi
  INNER JOIN
    `Target.orders` AS o
  USING
    (order_id)
  INNER JOIN
    `Target.customers` AS c
  ON
    o.customer_id = c.customer_id) AS T
GROUP BY
  State, Avg_Freight_Value;
```

→ → *Result*:

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

Row	State	Avg_Freight_Value	Ranking_for_Lowest	Ranking_for_Highest
1	RR	42.98442307692…	27	1
2	PB	42.72380398671…	26	2
3	RO	41.06971223021…	25	3
4	AC	40.07336956521…	24	4
5	PI	39.14797047970…	23	5
6	MA	38.25700242718…	22	6
7	TO	37.24660317460…	21	7
8	SE	36.65316883116…	20	8
9	AL	35.84367117117…	19	9
10	PA	35.83268518518…	18	10

3. Top 5 states with the lowest average delivery time.

→ → *Query:*

```
SELECT
  DISTINCT c.customer_state ,AVG (o.Actual_Delivery_Time_in_Days) OVER (PARTITION
BY c.customer_state) AS Avg_Delivery_Time
FROM
(SELECT
  customer_id, order_id,
  TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp, day) AS
Actual_Delivery_Time_in_Days ,
  TIMESTAMP_DIFF(order_estimated_delivery_date, order_purchase_timestamp, day) AS
Estimated_Delivery_Time_in_Days,
FROM
  `Target.orders`
WHERE
  order_status = "delivered") AS o
INNER JOIN
  `Target.customers` AS c
USING
  (customer_id)
ORDER BY Avg_Delivery_Time
LIMIT 5;
```

→→ *Result*

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

Row	customer_state	Avg_Delivery_Time
1	SP	8.298093544722...
2	PR	11.52671135486...
3	MG	11.54218777523...
4	DF	12.50913461538...
5	SC	14.47518330513...

The above data provides top 5 states with lowest average delivery time in days. The data for top 5 states with highest average delivery time can be retrieved by ordering the same data by "Avg_Delivery_Time" in descending order.

6. Analysis based on the payments

1. Month on month number of orders placed using different payment types.

→→ *Query:*

```
SELECT
  time_period,
  payment_type,
  COUNT(*) AS Total_Orders
FROM
  (SELECT
    p.order_id,
    p.payment_type,
    EXTRACT(YEAR FROM order_purchase_timestamp) AS Year,
    EXTRACT(Month FROM order_purchase_timestamp) AS Month,
    FORMAT_DATE('%b %Y', DATE(order_purchase_timestamp)) AS Time_Period
  FROM
    `Target.payments` AS p
  JOIN
    `Target.orders` AS o
  USING
    (order_id)) T
GROUP BY
  time_period, payment_type, T.YEAR, T.month
ORDER BY
  T.YEAR, T.month,payment_type;
```

→ → *Result*:

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

Row	time_period	payment_type	Total_Orders
1	Sep 2016	credit_card	3
2	Oct 2016	UPI	63
3	Oct 2016	credit_card	254
4	Oct 2016	debit_card	2
5	Oct 2016	voucher	23
6	Dec 2016	credit_card	1
7	Jan 2017	UPI	197
8	Jan 2017	credit_card	583
9	Jan 2017	debit_card	9
10	Jan 2017	voucher	61

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

→ → *Query:*

```
SELECT
  payment_installments,
  COUNT(*) AS Total_Orders
FROM
  `Target.payments`
GROUP BY
  payment_installments;
```

→ → *Result*:

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

Row	payment_installment	Total_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

7. Actionable Insights & Recommendations

1. The orders trajectory shows a sharp rise in the volume of orders within a short period of time. Business in Brazil is growing quickly, according to the general trend, thus companies need to be prepared with more staff. Company may think about employing contract workers to reduce excessive risk.

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	time_period	order_count	growth_percent
1	Sep 2016	1	*null*
2	Oct 2016	265	26400.0
3	Dec 2016	1	-99.62
4	Jan 2017	750	74900.0
5	Feb 2017	1653	120.4
6	Mar 2017	2546	54.02
7	Apr 2017	2303	-9.54
8	May 2017	3546	53.97
9	Jun 2017	3135	-11.59
10	Jul 2017	3872	23.51

2. Total 609 orders were unavailable and 625 orders were cancelled during the given time period, which makes it to be around 1.2 % of total orders. We can reduce this number by studying the reasons behind order cancellation and items unavailability.

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECU

Row	order_status	orders_count	percent_of_total_orders
1	delivered	96478	97.02
2	shipped	1107	1.11
3	canceled	625	0.63
4	unavailable	609	0.61
5	invoiced	314	0.32
6	processing	301	0.3
7	created	5	0.01
8	approved	2	0.0

3. The query below calculates the ratio of review score for each is state. While, the extracted data is ordered to get the states with higher proportion of unsatisfied customers.

→→ *Query:*

```
SELECT
  customer_state AS State,
  ROUND(_1/Total_Reviews*100,1) AS R1,
  ROUND(_2/Total_Reviews*100,1) AS R2,
  ROUND(_3/Total_Reviews*100,1) AS R3,
  ROUND(_4/Total_Reviews*100,1) AS R4,
  ROUND(_5/Total_Reviews*100,1) AS R5
FROM
(SELECT *,(_1 + _2 + _3 + _4 +_5) AS Total_Reviews
FROM
(SELECT
  *
FROM (
  SELECT
    c.customer_state,
    orr.review_score
  FROM
    `Target.order_reviews` orr
  JOIN
    `Target.orders` o
  USING
    (order_id)
  JOIN
    `Target.customers` c
  USING
    (customer_id))
  PIVOT(COUNT(*) FOR review_score IN (1, 2, 3, 4, 5))) AS T)
  ORDER BY R1 DESC,R5;
```

→→ *Result*:

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

Row	State	R1	R2	R3	R4	R5
1	RR	19.6	4.3	15.2	17.4	43.5
2	SE	18.1	3.7	8.3	19.2	50.7
3	MA	17.6	4.2	9.9	21.0	47.3
4	AL	17.6	6.3	7.0	21.5	47.6
5	RJ	17.1	3.6	8.2	16.7	54.3
6	PA	16.0	3.6	9.9	20.4	50.1
7	CE	15.8	4.1	9.9	19.8	50.5
8	BA	15.0	3.9	10.0	22.2	48.9
9	PI	14.7	3.7	8.6	21.2	51.9
10	PE	13.4	3.2	8.0	19.6	55.8

4. A closer look reveals that the majority of these complaints mention problems with deliveries that were delayed or that the consumer did not get. However, a lot of the negative reviews are also brought on by things that were shipped incorrectly or damaged. Therefore, the business should concentrate on improving its logistics in order to win over customers and increase profitability. The most popular review titles are displayed with the following query.

→→ *Query:*

```
SELECT
  review_comment_title, count(review_comment_title) as CNT
FROM
  `Target.order_reviews`
WHERE
  review_score IN (1,2)
GROUP BY
  review_comment_title
ORDER BY
  CNT DESC;
```

→→ *Result*:

JOB INFORMATION | RESULTS | CHART | JSON

Row	review_comment_title	CNT
1	I recommend	118
2	I didn't receive the product	57
3	Bad	52
4	Product not delivered	48
5	Wrong product	47
6	Defective product	39
7	PÃ © ssimo	32
8	I didn't receive	24
9	Delivery delay	21
10	Good	21
11	I do not recommend	18

5. Rio de Janeirio, Sao Paulo and Belo Horizonte among others cities in Brazil that contribute the major chunk of orders. The company can upscale their businesses by improving their product offerings and logistics in order to attain higher customer satisfaction to gain more trust in these markets. The results of this analysis can be referred from “In-depth Analysis”, consisting the data for number of customers and number of orders placed per city.

6. Additionally, Brazilian customers show a tendency for shopping online during afternoon and night. The company can focus on scheduling and optimizing their digital marketing campaigns during these hours of the day to increasing customer engagement.