Target Case Study

```sql
# Data type of all columns in the "customers" table.
# Assuming customers table already exists in the database
SELECT column_name, data_type
FROM target_sales_2016_2018.INFORMATION_SCHEMA.COLUMNS
WHERE table_name = 'customers';
```

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

```sql
# Get the time range between which the orders were placed.
# Count the Cities & States of customers who ordered during the given period.
SELECT
  MIN(order_purchase_timestamp) AS start_date,
  MAX(order_purchase_timestamp) AS end_date
FROM target_sales_2016_2018.orders;
```

JOB INFORMATION | RESULTS | CHART | JSON | E

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

```sql
SELECT
  COUNT(DISTINCT customer_city) AS num_cities,
  COUNT(DISTINCT customer_state) AS num_states
FROM target_sales_2016_2018.customers
WHERE customer_id IN
  (SELECT DISTINCT customer_id
  FROM target_sales_2016_2018.orders
  );
```

JOB INFORMATION | RESULTS | CHART

Row	num_cities	num_states
1	4119	27

```
# In-depth Exploration:
# Is there a growing trend in the no. of orders placed over the past years?

SELECT
  EXTRACT(YEAR FROM order_purchase_timestamp) AS order_year,
  COUNT(*) AS num_orders
FROM target_sales_2016_2018.orders
GROUP BY order_year
ORDER BY order_year;
```

Row	order_year	num_orders
1	2016	329
2	2017	45101
3	2018	54011

Inference: Yes, there has been major growth from 2016-17, then a small growth from 2017-18.

```
# Can we see some kind of monthly seasonality in terms of the no. of orders being placed?
SELECT EXTRACT(MONTH FROM order_purchase_timestamp) AS order_month,
       COUNT(*) AS num_orders
FROM target_sales_2016_2018.orders
GROUP BY order_month
ORDER BY order_month;
```

Row	order_month	num_orders
1	1	8069
2	2	8508
3	3	9893
4	4	9343
5	5	10573
6	6	9412
7	7	10318
8	8	10843
9	9	4305
10	10	4959

Inference: From the results we can infer that peak seasons were mid-years May-Aug, Weak seasons were end of the year Sep-Dec.

```sql
# During what time of the day, do the Brazilian customers mostly place their orders? (Dawn,
Morning, Afternoon or Night)
# 0-6 hrs : Dawn
# 7-12 hrs : Mornings
# 13-18 hrs : Afternoon
# 19-23 hrs : 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 order_time_of_day,
    COUNT(*) AS num_orders
FROM target_sales_2016_2018.orders
GROUP BY order_time_of_day
ORDER BY num_orders DESC;
```

JOB INFORMATION | RESULTS | CHART | JSC

Row	order_time_of_day	num_orders
1	Afternoon	38135
2	Night	28331
3	Morning	27733
4	Dawn	5242

Inference: Most of the orders were placed during Afternoon time.

```sql
# Evolution of E-commerce orders in the Brazil region:
# Get the month on month no. of orders placed in each state.
SELECT
  EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month,
  c.customer_state,
  COUNT(*) AS num_orders
FROM target_sales_2016_2018.orders o
JOIN target_sales_2016_2018.customers c ON o.customer_id = c.customer_id
GROUP BY order_month, c.customer_state
ORDER BY order_month, c.customer_state;
```

Row	order_month	customer_state	num_orders
1	1	AC	8
2	1	AL	39
3	1	AM	12
4	1	AP	11
5	1	BA	264
6	1	CE	99
7	1	DF	151
8	1	ES	159
9	1	GO	164
10	1	MA	66

```
# How are the customers distributed across all the states?
SELECT customer_state,
       COUNT(DISTINCT customer_id) AS num_customers
FROM target_sales_2016_2018.customers
GROUP BY customer_state
ORDER BY num_customers DESC;
```

Row	customer_state	num_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

Inference: State São Paulo (SP) has the most customers, while state Roraima (RR) has the least customer base.

```
# Impact on Economy: Analyze the money movement by e-commerce by looking at order prices,
freight and others.
# Get the % increase in the cost of orders from year 2017 to 2018 (include months between
Jan to Aug only).

WITH order_costs_2017 AS (
    SELECT EXTRACT(YEAR FROM o.order_purchase_timestamp) AS order_year,
           EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month,
           SUM(p.payment_value) AS total_payment_value
    FROM target_sales_2016_2018.orders o
    JOIN target_sales_2016_2018.payments p ON o.order_id = p.order_id
    WHERE EXTRACT(YEAR FROM o.order_purchase_timestamp)=2017
      AND EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8
    GROUP BY order_year, order_month
), order_costs_2018 AS (
    SELECT EXTRACT(YEAR FROM o.order_purchase_timestamp) AS order_year,
           EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month,
           SUM(p.payment_value) AS total_payment_value
    FROM target_sales_2016_2018.orders o
    JOIN target_sales_2016_2018.payments p ON o.order_id = p.order_id
    WHERE EXTRACT(YEAR FROM o.order_purchase_timestamp)=2018
      AND EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8
    GROUP BY order_year, order_month
)

SELECT
  order_costs_2017.order_month as month,
  (order_costs_2018.total_payment_value - order_costs_2017.total_payment_value) /
order_costs_2017.total_payment_value * 100 AS percentage_increase
FROM order_costs_2017
JOIN order_costs_2018 ON
order_costs_2017.order_month = order_costs_2018.order_month
ORDER BY month;
```

JOB INFORMATION | RESULTS | CHA

Row	month	percentage_increase
1	1	705.1266954171...
2	2	239.9918145445...
3	3	157.7786066709...
4	4	177.8407701149...
5	5	94.62734375677...
6	6	100.2596912456...
7	7	80.04245463390...
8	8	51.60600520477...

Inference: Month of January had a huge boost of 700% compared to other months

```sql
# Calculate the Total & Average value of order price for each state.
SELECT c.customer_state,
      ROUND(SUM(p.payment_value),2) AS total_order_price,
      ROUND(AVG(p.payment_value),2) AS avg_order_price
FROM target_sales_2016_2018.orders o
JOIN target_sales_2016_2018.customers c ON o.customer_id = c.customer_id
JOIN target_sales_2016_2018.payments p ON o.order_id = p.order_id
GROUP BY c.customer_state
ORDER BY c.customer_state;
```

Row	customer_state	total_order_price	avg_order_price
1	AC	19680.62	234.29
2	AL	96962.06	227.08
3	AM	27966.93	181.6
4	AP	16262.8	232.33
5	BA	616645.82	170.82
6	CE	279464.03	199.9
7	DF	355141.08	161.13
8	ES	325967.55	154.71
9	GO	350092.31	165.76
10	MA	152523.02	198.86

```sql
# Calculate the Total & Average value of order freight for each state.
SELECT c.customer_state,
       ROUND(SUM(oi.freight_value),2) AS total_freight_value,
       ROUND(AVG(oi.freight_value),2) AS avg_freight_value
FROM target_sales_2016_2018.orders o
JOIN target_sales_2016_2018.customers c ON o.customer_id = c.customer_id
JOIN target_sales_2016_2018.order_items oi ON oi.order_id = o.order_id
GROUP BY c.customer_state
ORDER BY c.customer_state;
```

Row	customer_state	total_freight_value	avg_freight_value
1	AC	3686.75	40.07
2	AL	15914.59	35.84
3	AM	5478.89	33.21
4	AP	2788.5	34.01
5	BA	100156.68	26.36
6	CE	48351.59	32.71
7	DF	50625.5	21.04
8	ES	49764.6	22.06
9	GO	53114.98	22.77
10	MA	31523.77	38.26

```
# Analysis based on sales, freight and delivery time.
# 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.

WITH delivery_times AS (
    SELECT
        order_id,
        DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) AS
delivery_time,
        DATE_DIFF(order_delivered_customer_date, order_estimated_delivery_date, DAY) AS
diff_estimated_delivery
    FROM
    target_sales_2016_2018.orders
    WHERE
        order_delivered_customer_date IS NOT NULL
        AND order_estimated_delivery_date IS NOT NULL
)

SELECT
    order_id,
    delivery_time,
    diff_estimated_delivery
FROM
    delivery_times;
```

ow	order_id	delivery_time	diff_estimated_delive
1	1950d777989f6a877539f53795…	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

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

WITH state_avg_freight AS (
    SELECT
      c.customer_state,
      AVG(oi.freight_value) AS avg_freight,
    FROM target_sales_2016_2018.orders o
    JOIN target_sales_2016_2018.customers c ON o.customer_id = c.customer_id
    JOIN target_sales_2016_2018.order_items oi ON oi.order_id = o.order_id
    GROUP BY c.customer_state
), state_ranks as (
  SELECT
    customer_state,
    avg_freight,
    ROW_NUMBER() OVER (ORDER BY avg_freight DESC) AS rank_high,
    ROW_NUMBER() OVER (ORDER BY avg_freight ASC) AS rank_low
  FROM state_avg_freight
)

SELECT
  customer_state,
  avg_freight,
  rank_high
FROM state_ranks
WHERE rank_high<=5
```

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTI

Row	customer_state	avg_freight	rank_high
1	RR	42.98442307692...	1
2	PB	42.72380398671...	2
3	RO	41.06971223021...	3
4	AC	40.07336956521...	4
5	PI	39.14797047970...	5

```sql
SELECT
  customer_state,
  avg_freight,
  rank_low
FROM state_ranks
WHERE rank_low<=5;
```

Row	customer_state	avg_freight	rank_low
1	SP	15.14727539041...	1
2	PR	20.53165156794...	2
3	MG	20.63016680630...	3
4	RJ	20.96092393168...	4
5	DF	21.04135494596...	5

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

WITH state_delivery_time AS (
    SELECT
      c.customer_state,
      AVG(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY)) AS
avg_delivery_time,
      ROW_NUMBER() OVER (ORDER BY AVG(DATE_DIFF(order_delivered_customer_date,
order_purchase_timestamp, DAY)) DESC) AS rank_high,
      ROW_NUMBER() OVER (ORDER BY AVG(DATE_DIFF(order_delivered_customer_date,
order_purchase_timestamp, DAY)) ASC) AS rank_low
    FROM target_sales_2016_2018.orders o
    JOIN target_sales_2016_2018.customers c ON o.customer_id = c.customer_id
    WHERE order_delivered_customer_date IS NOT NULL
    GROUP BY c.customer_state
)

SELECT
    customer_state,
    avg_delivery_time,
    rank_high
FROM state_delivery_time
WHERE rank_high <= 5;
```

Row	customer_state	avg_delivery_time	rank_high
1	RR	28.97560975609...	1
2	AP	26.73134328358...	2
3	AM	25.98620689655...	3
4	AL	24.04030226700...	4
5	PA	23.31606765327...	5

```
# SELECT
#    customer_state,
#    avg_delivery_time,
#    rank_low
# FROM state_delivery_time
# WHERE rank_low <= 5
```

Row	customer_state	avg_delivery_time	rank_low
1	SP	8.298061489072...	1
2	PR	11.52671135486...	2
3	MG	11.54381329810...	3
4	DF	12.50913461538...	4
5	SC	14.47956019171...	5

```
# 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.

WITH state_delivery_speed AS (
    SELECT
        c.customer_state,
        AVG(DATE_DIFF(order_delivered_customer_date, order_estimated_delivery_date, DAY))
AS avg_delivery_speed,
        ROW_NUMBER() OVER (ORDER BY AVG(DATE_DIFF(order_delivered_customer_date,
order_estimated_delivery_date, DAY)) ASC) AS rank_fastest
    FROM target_sales_2016_2018.orders o
    JOIN target_sales_2016_2018.customers c ON o.customer_id = c.customer_id
    WHERE
        order_delivered_customer_date IS NOT NULL
        AND order_estimated_delivery_date IS NOT NULL
    GROUP BY c.customer_state
)

SELECT
    customer_state,
    avg_delivery_speed,
    rank_fastest
FROM state_delivery_speed
WHERE rank_fastest <= 5;
```

Row	customer_state	avg_delivery_speed	rank_fastest
1	AC		1
2	RO	-19.13168724279836	2
3	AP	-18.731343283582088	3
4	AM	-18.60689655172413	4
5	RR	-16.414634146341463	5

avg_delivery_speed

```
# Analysis based on the payments:
# Find the month on month no. of orders placed using different payment types.

SELECT
    EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month,
    p.payment_type,
    COUNT(*) AS num_orders
FROM target_sales_2016_2018.orders o
JOIN target_sales_2016_2018.payments p ON o.order_id = p.order_id
GROUP BY order_month, p.payment_type
ORDER BY order_month, p.payment_type;
```

Row	order_month	payment_type	num_orders
1	1	UPI	1715
2	1	credit_card	6103
3	1	debit_card	118
4	1	voucher	477
5	2	UPI	1723
6	2	credit_card	6609
7	2	debit_card	82
8	2	voucher	424
9	3	UPI	1942
10	3	credit_card	7707

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

SELECT
    payment_installments,
    COUNT(*) AS num_orders
FROM target_sales_2016_2018.payments
GROUP BY payment_installments
ORDER BY payment_installments;
```

JOB INFORMATION | RESULTS | CHAR

ow	payment_installment	num_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

--

Actionable Insights and recommendations:

1. Introducing special offers, discounts, events, etc., in the time of peak season May-Aug.
2. Taking special care and measures to maintain the order servers in Afternoon period.
3. Designating some states like SP as high value due to large customer base and creating business plans to improve sales push in these regions.
4. Upgrading transport and infrastructure to improve the Delivery days taken for states like RR, AP, who has smaller customer base.
5. Introducing offers catered to the credit card users due to its popularity as a preferred payment method.