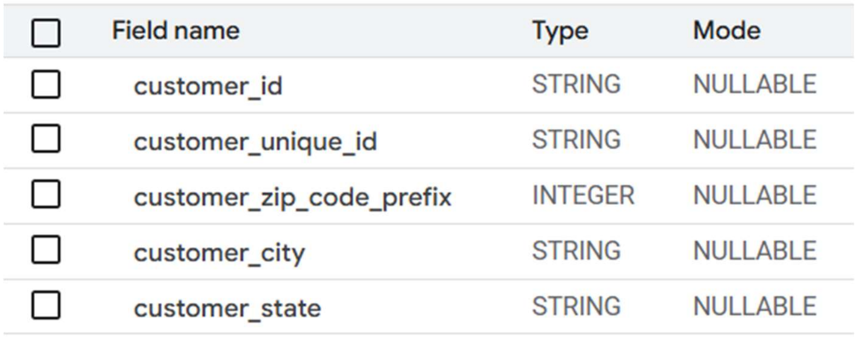
Problem Statement:

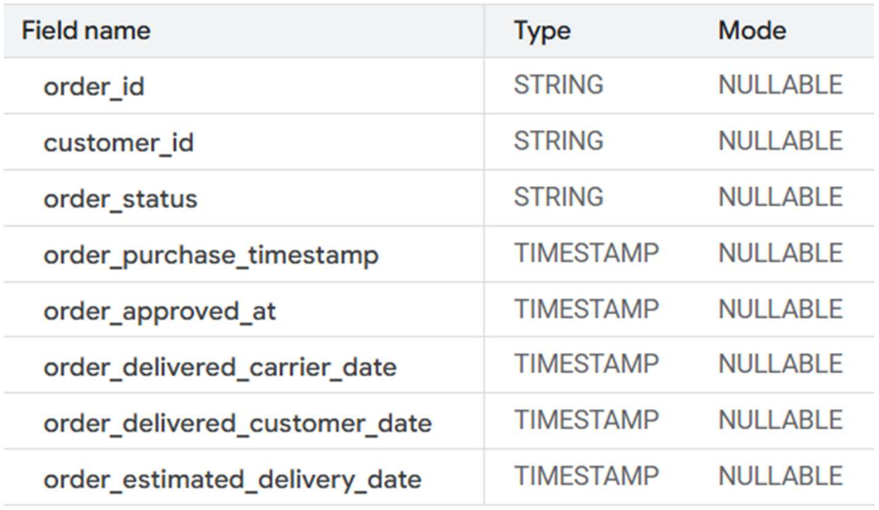
Target seeks to improve its operations and customer experience in Brazil by analysing historical order data FROM 2016 to 2018. The objective is to uncover insights related to order fulfilment, pricing, payments, shipping, customer behaviour, product performance, and reviews. These insights will help Target enhance operational efficiency, optimize pricing and delivery strategies, and improve customer satisfaction to strengthen its position in the Brazilian retail market.

- **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

- **Date type of all columns in the “orders” table.**



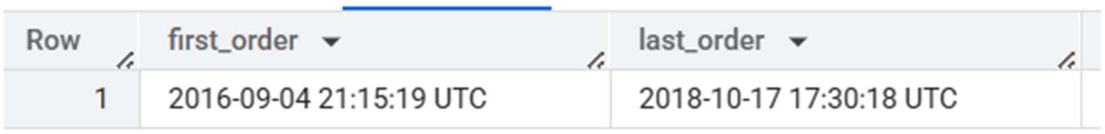
Field name	Type	Mode
order_id	STRING	NULLABLE
customer_id	STRING	NULLABLE
order_status	STRING	NULLABLE
order_purchase_timestamp	TIMESTAMP	NULLABLE
order_approved_at	TIMESTAMP	NULLABLE
order_delivered_carrier_date	TIMESTAMP	NULLABLE
order_delivered_customer_date	TIMESTAMP	NULLABLE
order_estimated_delivery_date	TIMESTAMP	NULLABLE

- **Get the time range between which the orders were placed.**

Code:

```
SELECT
min(order_purchase_timestamp) as first_order,
max(order_purchase_timestamp) as last_order
FROM
`scalerproject-454408.target_sql_project.orders`
;
```

Output:



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

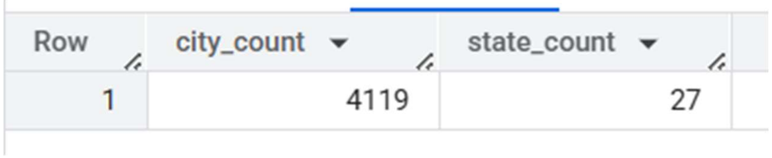
First order was at 4th September 2016 and the Last Order was at 17th October 2018

- **COUNT the Cities & States of customers who ordered during the given period.**

Code:

```
SELECT
        COUNT(distinct c.customer_city) as city_COUNT,
        COUNT(distinct c.customer_state) as state_COUNT
FROM
        `scalerproject-454408.target_sql_project.customers` as c
        JOIN
        `scalerproject-454408.target_sql_project.orders` as o
        on c.customer_id = o.customer_id  ;
```

Output:



Row	city_count	state_count
1	4119	27

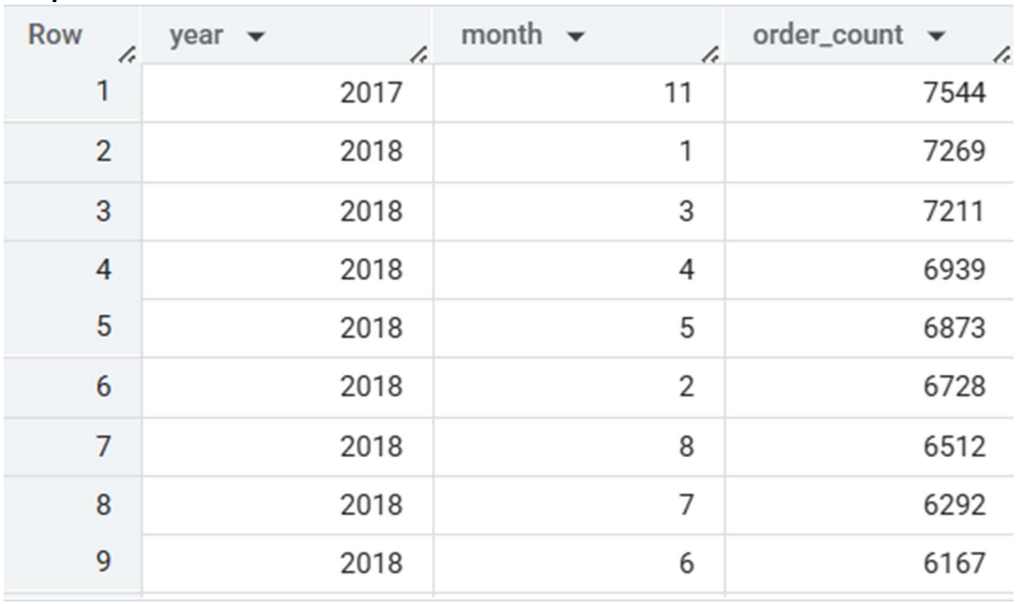
Total City COUNT is at 4119 and Total State COUNT is at 27

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

Code:

```
SELECT
        EXTRACT(year FROM order_purchase_timestamp) as year,
        EXTRACT(month FROM order_purchase_timestamp) as month,
        COUNT(order_id) as order_COUNT
FROM
        `scalerproject-454408.target_sql_project.orders`
GROUP BY year,month
ORDER BY order_COUNT DESC,year,month
;
```

Output:



Row	year	month	order_count
1	2017	11	7544
2	2018	1	7269
3	2018	3	7211
4	2018	4	6939
5	2018	5	6873
6	2018	2	6728
7	2018	8	6512
8	2018	7	6292
9	2018	6	6167

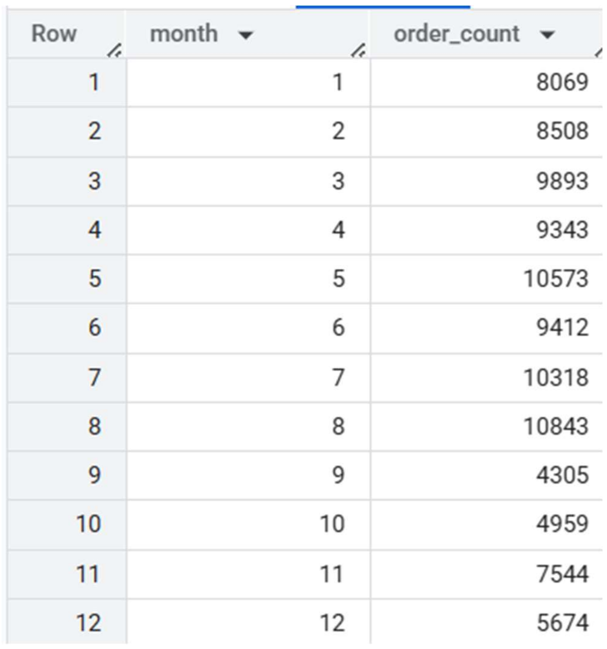
Highest order count of 7544 was at November of 2017 followed by Jan of 2018

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

Code:

```
SELECT
        EXTRACT(month FROM order_purchase_timestamp) as month,
        COUNT(order_id) as order_count
FROM
        `scalerproject-454408.target_sql_project.orders`
GROUP BY month
ORDER BY order_count DESC,month
;
```

Output:



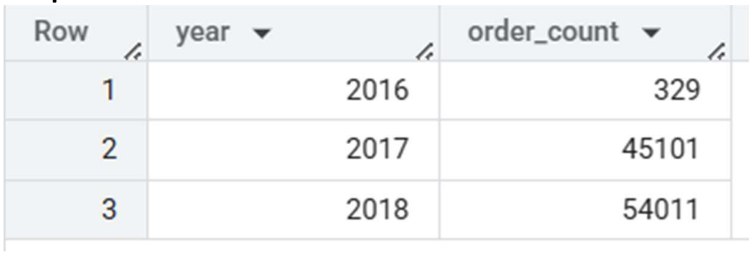
Row	month	order_count
1	1	8069
2	2	8508
3	3	9893
4	4	9343
5	5	10573
6	6	9412
7	7	10318
8	8	10843
9	9	4305
10	10	4959
11	11	7544
12	12	5674

#Year wise Order Count

Code:

```
SELECT
        EXTRACT(year FROM order_purchase_timestamp) as year,
        COUNT(order_id) as order_count
FROM
        `scalerproject-454408.target_sql_project.orders`
GROUP BY year
ORDER BY order_count DESC,year
;
```

Output:



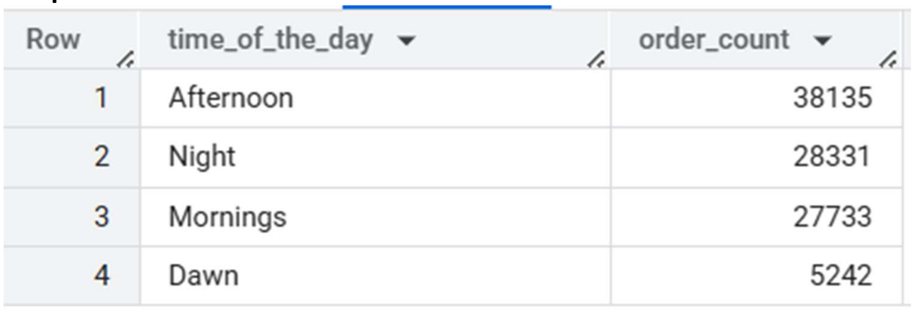
Row	year	order_count
1	2016	329
2	2017	45101
3	2018	54011

# 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

Code:

```
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
'Mornings'
        WHEN EXTRACT(hour FROM order_purchase_timestamp) between 13 and 18 then
'Afternoon'
        else 'Night' end as time_of_the_day,
        COUNT(order_id) as order_ count
FROM
        `scalerproject-454408.target_sql_project.orders`
GROUP BY time_of_the_day
ORDER BY order_ count DESC ;
```

Output:



Row	time_of_the_day	order_count
1	Afternoon	38135
2	Night	28331
3	Mornings	27733
4	Dawn	5242

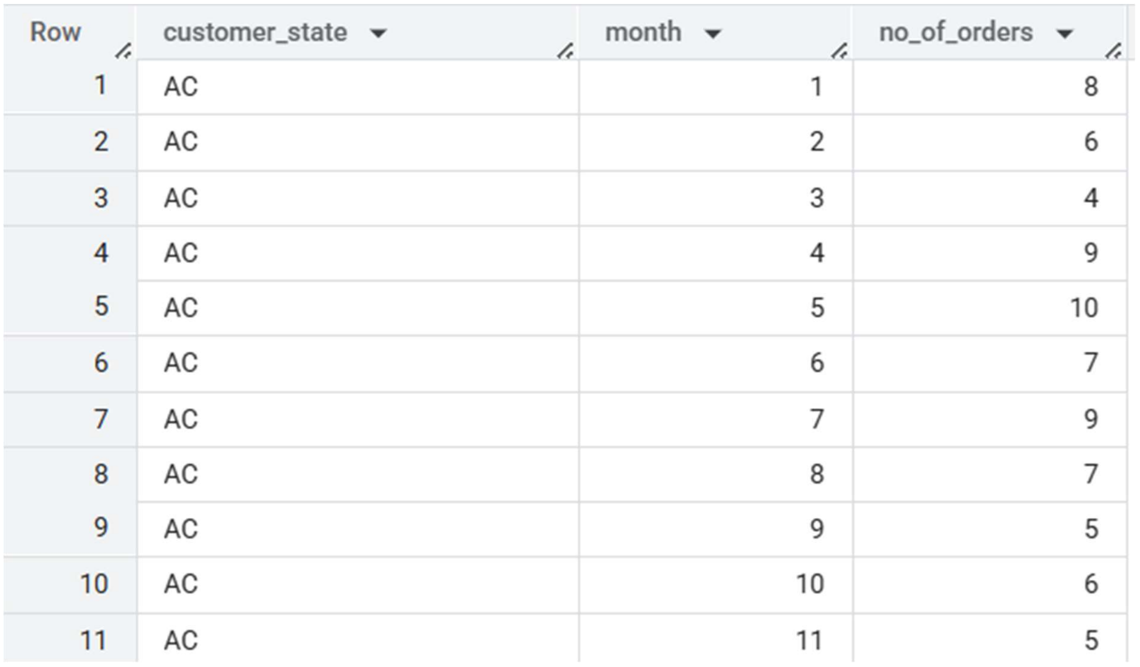
# Evolution of E-commerce orders in the Brazil region:

- **Get the month-on-month no. of orders placed in each state.**

Code:

```
SELECT
        c.customer_state,
        EXTRACT(month FROM o.order_purchase_timestamp) as month,
        COUNT(o.order_id) as no_of_orders
FROM
    `scalerproject-454408.target_sql_project.customers` as c
JOIN
    `scalerproject-454408.target_sql_project.orders` as o
    on c.customer_id = o.customer_id
GROUP BY c.customer_state, month
ORDER BY c.customer_state, month ;
```

Output:



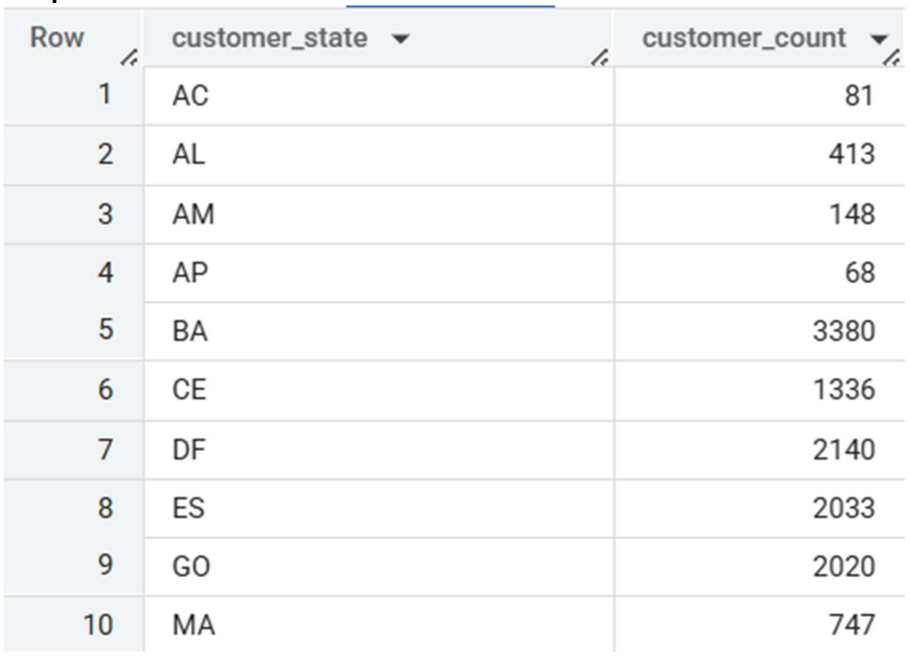
Row	customer_state	month	no_of_orders
1	AC	1	8
2	AC	2	6
3	AC	3	4
4	AC	4	9
5	AC	5	10
6	AC	6	7
7	AC	7	9
8	AC	8	7
9	AC	9	5
10	AC	10	6
11	AC	11	5

#How are the customers distributed across all the states?

Code:

```
SELECT
        customer_state,
        COUNT(customer_id) as customer_ count
FROM
        `scalerproject-454408.target_sql_project.customers`
GROUP BY customer_state
ORDER BY customer_state ;
```

Output:



Row	customer_state	customer_count
1	AC	81
2	AL	413
3	AM	148
4	AP	68
5	BA	3380
6	CE	1336
7	DF	2140
8	ES	2033
9	GO	2020
10	MA	747

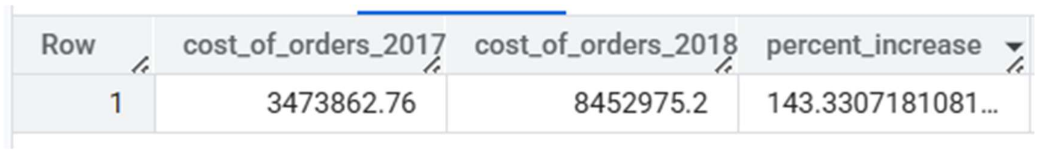
# Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.

- **Get the % increase in the cost of orders FROM year 2017 to 2018 (include months between Jan to Aug only)**

Method 1:
Code:

```
SELECT
        ROUND(sum(CASE
         WHEN EXTRACT(year FROM order_purchase_timestamp) = 2017 then
        p.payment_value
         else 0 end),2) as cost_of_orders_2017,
        ROUND(sum(CASE
        WHEN EXTRACT(year FROM o.order_purchase_timestamp) = 2018 then
p.payment_value
        else 0 end),2) as cost_of_orders_2018,
        (sum(CASE
        WHEN EXTRACT(year FROM o.order_purchase_timestamp) = 2018 then
        p.payment_value    else 0 end) *100 /sum(CASE
        WHEN EXTRACT(year FROM o.order_purchase_timestamp) = 2017 then
        p.payment_value else 0 end))-100 as percent_increase
FROM
        `scalerproject-454408.target_sql_project.orders` as o
JOIN
        `scalerproject-454408.target_sql_project.payments` as p
        on o.order_id = p.order_id
WHERE  (EXTRACT(year FROM o.order_purchase_timestamp) between 2017 and 2018) and
        (EXTRACT(month FROM o.order_purchase_timestamp) between 1 and 8) and
        o.order_status='delivered' ;
```

Output:

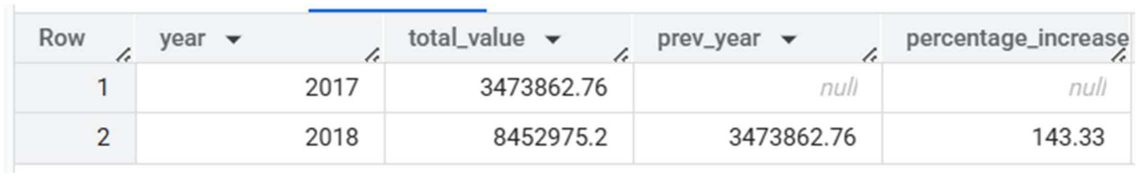


Row	cost_of_orders_2017	cost_of_orders_2018	percent_increase
1	3473862.76	8452975.2	143.3307181081...

Method 2:
Code:

```
With CLT as (
SELECT
        EXTRACT(year FROM o.order_purchase_timestamp) as year,
        ROUND(sum(p.payment_value),2) as total_value
FROM
        `scalerproject-454408.target_sql_project.orders` as o
JOIN
        `scalerproject-454408.target_sql_project.payments` as p
        on o.order_id = p.order_id
WHERE (EXTRACT(year FROM o.order_purchase_timestamp) between 2017 and 2018) and
        (EXTRACT(month FROM o.order_purchase_timestamp) between 1 and 8) and
        o.order_status='delivered'
GROUP BY year
)
SELECT
         year,
        ROUND(total_value,2) as total_value,
        Lag(total_value) over(ORDER BY year ASC ) as prev_year,
        ROUND((total_value *100/(lag(total_value) over(ORDER BY year ASC )))-100,2) as
        percentage_increase
FROM CLT
ORDER BY total_value ASC
;
```

Output:



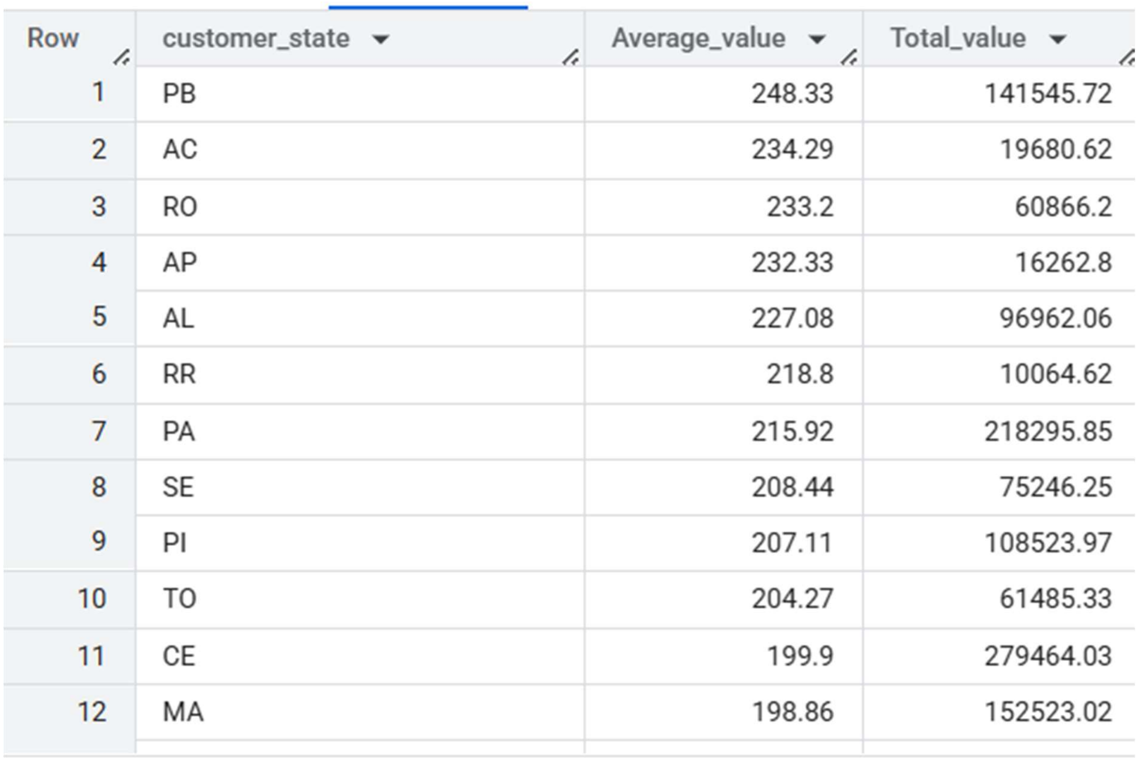
Row	year	total_value	prev_year	percentage_increase
1	2017	3473862.76	null	null
2	2018	8452975.2	3473862.76	143.33

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

Code:

```
SELECT
        c.customer_state,
        ROUND(avg(p.payment_value),2) as Average_value,
        ROUND(sum(p.payment_value),2) as Total_value
FROM
        `scalerproject-454408.target_sql_project.orders` as o
JOIN
        `scalerproject-454408.target_sql_project.payments` as p
        on p.order_id = o.order_id
JOIN
        `scalerproject-454408.target_sql_project.customers` as c
        on c.customer_id = o.customer_id
GROUP BY c.customer_state
ORDER BY Average_value DESC, Total_value DESC
;
```

Output:



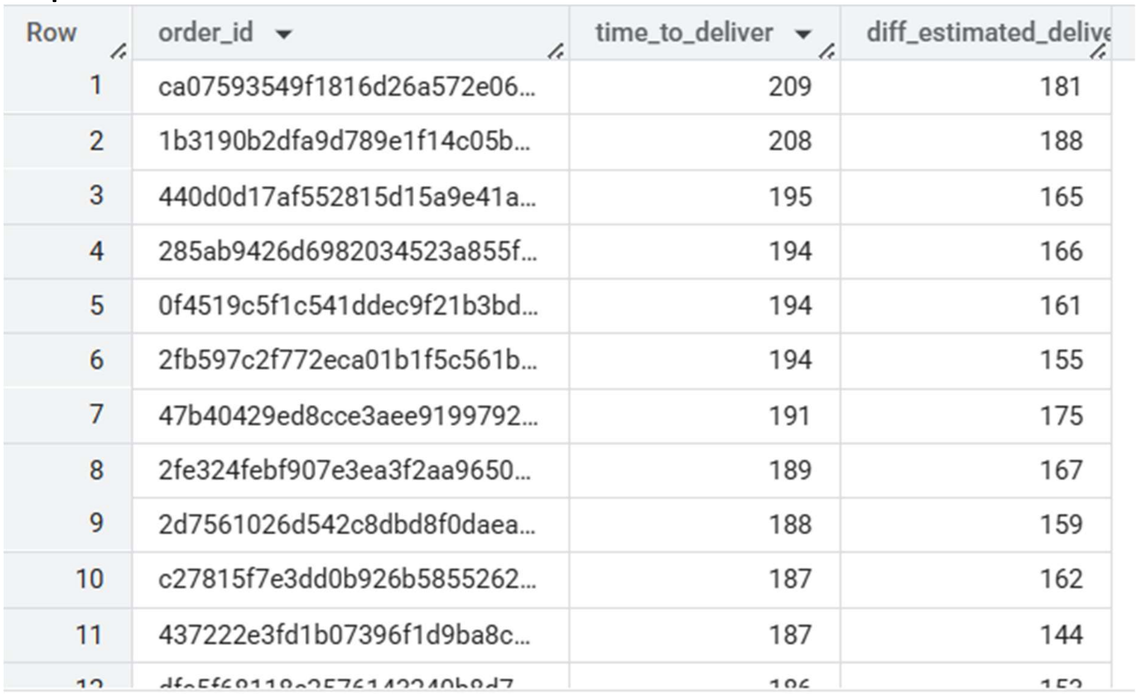
Row	customer_state	Average_value	Total_value
1	PB	248.33	141545.72
2	AC	234.29	19680.62
3	RO	233.2	60866.2
4	AP	232.33	16262.8
5	AL	227.08	96962.06
6	RR	218.8	10064.62
7	PA	215.92	218295.85
8	SE	208.44	75246.25
9	PI	207.11	108523.97
10	TO	204.27	61485.33
11	CE	199.9	279464.03
12	MA	198.86	152523.02

# 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.

Code:

```
SELECT
        order_id,
        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
        diff_estimated_delivery
FROM
        `scalerproject-454408.target_sql_project.orders`
WHERE order_status = 'delivered'
ORDER BY time_to_deliver DESC, diff_estimated_delivery DESC
;
```

Output:



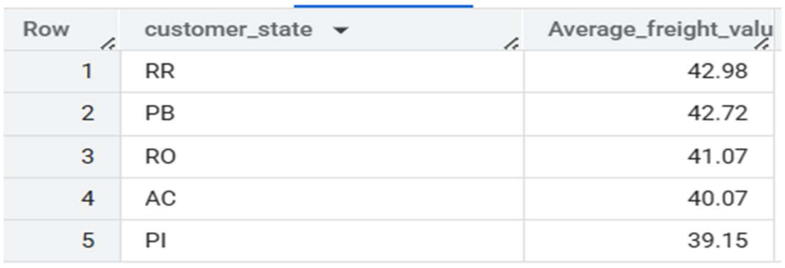
Row	order_id	time_to_deliver	diff_estimated_delive
1	ca07593549f1816d26a572e06...	209	181
2	1b3190b2dfa9d789e1f14c05b...	208	188
3	440d0d17af552815d15a9e41a...	195	165
4	285ab9426d6982034523a855f...	194	166
5	0f4519c5f1c541ddec9f21b3bd...	194	161
6	2fb597c2f772eca01b1f5c561b...	194	155
7	47b40429ed8cce3aee9199792...	191	175
8	2fe324febf907e3ea3f2aa9650...	189	167
9	2d7561026d542c8dbd8f0daea...	188	159
10	c27815f7e3dd0b926b5855262...	187	162
11	437222e3fd1b07396f1d9ba8c...	187	144
12	dfe5f68118e2576143240b8d7	186	152

# Find out the top 5 states with the highest & lowest average freight value.
# Highest Avg Freight Value:

Code:

```
SELECT
        c.customer_state,
        ROUND(avg(ot.freight_value),2) as Average_freight_value
FROM
        `scalerproject-454408.target_sql_project.customers` as c
JOIN
        `scalerproject-454408.target_sql_project.orders` as o
        on c.customer_id = o.customer_id
JOIN
        `scalerproject-454408.target_sql_project.order_items` as ot
        on ot.order_id = o.order_id
GROUP BY c.customer_state
ORDER BY Average_freight_value DESC
LIMIT 5 ;
```

Output:



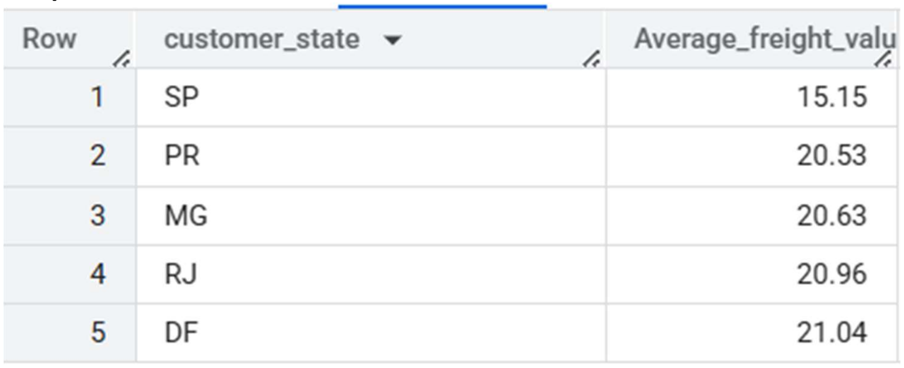
Row	customer_state	Average_freight_valu
1	RR	42.98
2	PB	42.72
3	RO	41.07
4	AC	40.07
5	PI	39.15

# Lowest Average Freight value:

Code:

```
SELECT
        c.customer_state,
        ROUND(avg(ot.freight_value),2) as Average_freight_value
FROM
        `scalerproject-454408.target_sql_project.customers` as c
JOIN
        `scalerproject-454408.target_sql_project.orders` as o
        on c.customer_id = o.customer_id
JOIN
        `scalerproject-454408.target_sql_project.order_items` as ot
        on ot.order_id = o.order_id
GROUP BY c.customer_state
ORDER BY Average_freight_value ASC
LIMIT 5 ;
```

Output:



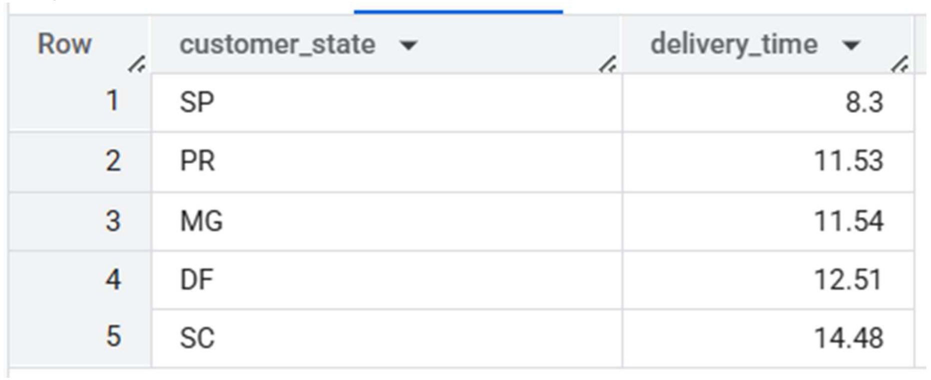
Row	customer_state	Average_freight_valu
1	SP	15.15
2	PR	20.53
3	MG	20.63
4	RJ	20.96
5	DF	21.04

# Find out the top 5 states with the highest & lowest average delivery time.
# top 5 states with Lowest Average Delivery time

Code:

```
SELECT
        c.customer_state,
        ROUND(avg(DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp
        , DAY)),2) as delivery_time
FROM
        `scalerproject-454408.target_sql_project.customers` as c
JOIN
        `scalerproject-454408.target_sql_project.orders` as o
        on c.customer_id = o.customer_id
GROUP BY c.customer_state
ORDER BY delivery_time ASC
LIMIT 5
;
```

Output:



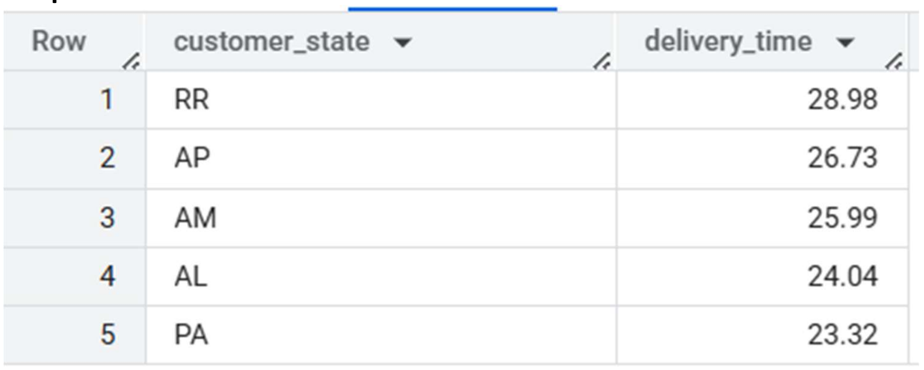
Row	customer_state	delivery_time
1	SP	8.3
2	PR	11.53
3	MG	11.54
4	DF	12.51
5	SC	14.48

# Top 5 states with Highest Average Delivery Time :

Code:

```
SELECT
        c.customer_state,
        ROUND(avg(DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,
DAY)),2) as delivery_time
FROM
         `scalerproject-454408.target_sql_project.customers` as c
JOIN
         `scalerproject-454408.target_sql_project.orders` as o
         on c.customer_id = o.customer_id
GROUP BY c.customer_state
ORDER BY delivery_time DESC
LIMIT 5 ;
```

Output:



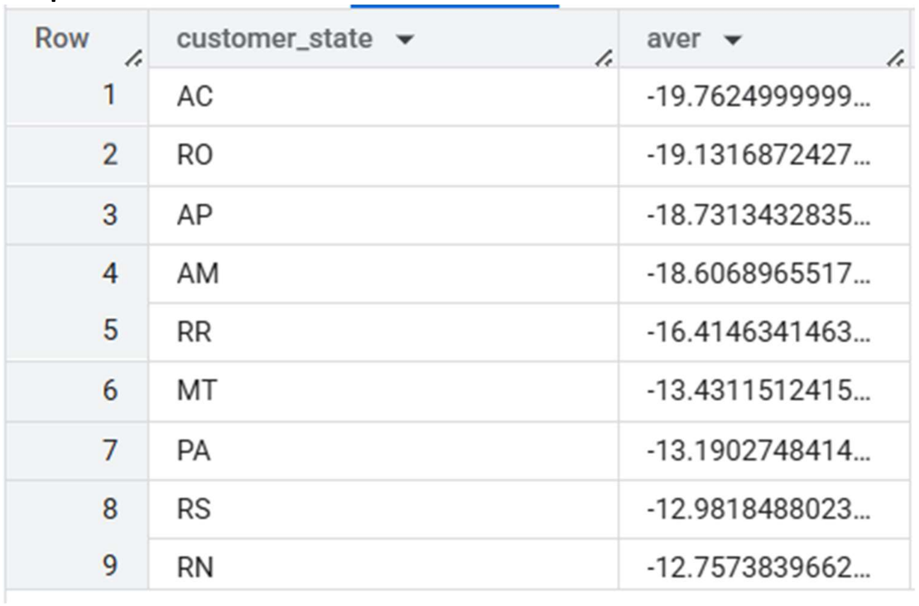
Row	customer_state	delivery_time
1	RR	28.98
2	AP	26.73
3	AM	25.99
4	AL	24.04
5	PA	23.32

# Find out the top 5 states WHERE the order delivery is really fast as compared to the estimated date of delivery. Difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.

Code:

```
with delivery_time as (
SELECT
         c.customer_state,
         c.customer_id,
         o.order_id,
	DATE_DIFF(o.order_delivered_customer_date,o.order_estimated_delivery_date,DAY)
as delivery_time_diff
FROM
	`scalerproject-454408.target_sql_project.orders` as o
JOIN
	`scalerproject-454408.target_sql_project.customers` as c
	on o.customer_id = c.customer_id
WHERE o.order_status = 'delivered'
)
SELECT
	customer_state,
	avg(delivery_time_diff) as aver
FROM
delivery_time
GROUP BY customer_state
ORDER BY aver ASC ;
```

Output:



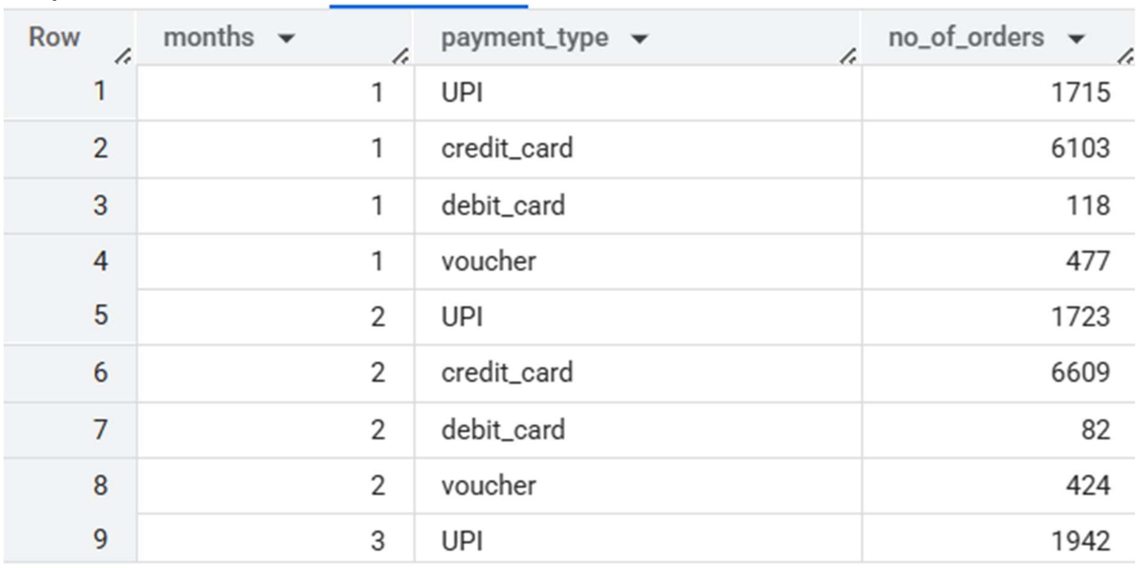
Row	customer_state	aver
1	AC	-19.7624999999...
2	RO	-19.1316872427...
3	AP	-18.7313432835...
4	AM	-18.6068965517...
5	RR	-16.4146341463...
6	MT	-13.4311512415...
7	PA	-13.1902748414...
8	RS	-12.9818488023...
9	RN	-12.7573839662...

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

Code:

```
SELECT
	EXTRACT(month FROM o.order_purchase_timestamp) as months,
	p.payment_type,
	COUNT(o.order_id) as no_of_orders
FROM
	`scalerproject-454408.target_sql_project.orders` as o
JOIN
	`scalerproject-454408.target_sql_project.payments` as p
on o.order_id = p.order_id
GROUP BY months, payment_type
ORDER BY months, payment_type, no_of_orders ASC  ;
```

Output:



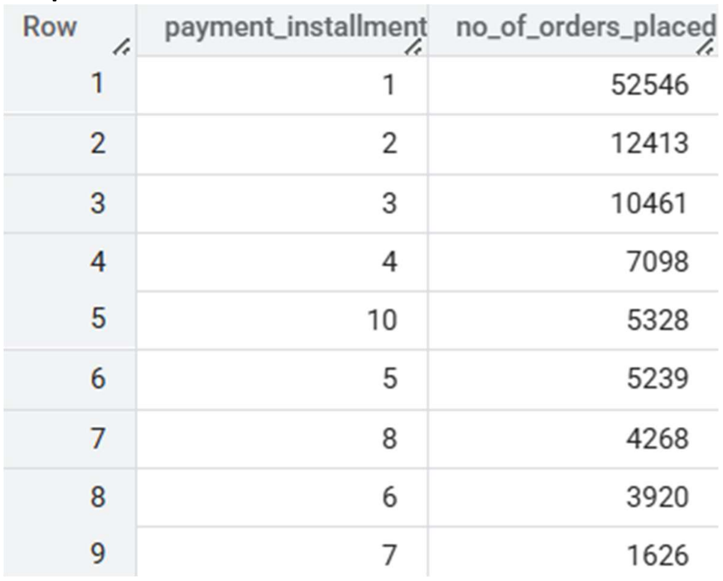
Row	months	payment_type	no_of_orders
1	1	UPI	1715
2	1	credit_card	6103
3	1	debit_card	118
4	1	voucher	477
5	2	UPI	1723
6	2	credit_card	6609
7	2	debit_card	82
8	2	voucher	424
9	3	UPI	1942

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

Code:

```
SELECT
        p.payment_installments,
        COUNT(o.order_id) as no_of_orders_placed
FROM
        `scalerproject-454408.target_sql_project.payments` as p
JOIN
        `scalerproject-454408.target_sql_project.orders` as o
        on p.order_id = o.order_id
GROUP BY p.payment_installments
ORDER BY no_of_orders_placed DESC ;
```

Output:



Row	payment_installment	no_of_orders_placed
1	1	52546
2	2	12413
3	3	10461
4	4	7098
5	10	5328
6	5	5239
7	8	4268
8	6	3920
9	7	1626

Data Analysis Insights:

- First order was at 4th September 2016 and the Last Order was at 17th October 2018
- Order details of Customers from total of 4119 city from 27 states
- Highest order count was at November 2017 followed by January to May of 2018
- Overall, August, May and July month contributed highest order count whereas September and October contributed less order count
- Though we had order details till October 2018 still 2018 contributed highest order counts of 54,011 compare to 2017 with 45,101
- when we compare day-part, a greater number of orders were placed at Afternoon (13 to 18hrs) followed by Night and Dawn (0 to 6hrs) has the lowest order count
- When we compare state wise order month on month, state='SP' contributed highest number of orders on August, May and July followed by other states
- State wise customer count, State-'SP' has highest customer count which actually matches with previous observation that 'SP' had the highest order count, RJ and MG were the second and third highest in customer count
- There is increase of 143% in Order value at 2018 compare to 2017 which matches with the last observation that 2018 has highest order count compare to 2017
- while observing Payment value it's clear that 'PB' has the highest average payment value followed by AC and RO state. In terms of Total value state-'CE' is the highest
- RP, PB, RO were the top 3 in highest Freight value
- SP, PR, MG were the top 3 in lowest Freight value
- SP has the lowest delivery time of 8.3 which in return reflects in highest sales and order count followed by PR with 11.53 and MG with 11.54
- PR, AP and AM have the highest delivery time of above 26
- In states AC, RO, AP, AM and RR actual delivery date is faster than the estimated delivery date
- Highest number of orders were done through credit card followed by UPI and then vouchers and Debit card
- Top 3 payment installments basic of order counts are 1,2,3 with total of 75k orders placed and lowest was 22 and 23 with only one order

Recommendations for Target:

Sales and Region Focus:

- Focus sales and promotions in high-performing states: SP, RJ, MG
- Expand campaigns in mid-tier but high-value states: CE, PB, RO
- Leverage customer density in SP to test new products and loyalty programs

Logistics and Delivery Optimization

- Reduce delivery times in slow regions: PR, AP, AM (over 26 days)
- Update estimated delivery dates where actual delivery is faster: AC, RO, AP, AM, RR
- Replicate SP's logistics model (8.3-day average) in other regions for improved efficiency

Payment Method Strategy:

- Promote Credit Card and UPI options — top 2 payment methods
- Focus on 1–3 installments options, which account for ~75% of orders

- Avoid complex installments options (20+), as they have minimal adoption

Time & Season-Based Marketing

- Run sales and campaigns during peak ordering times: Afternoon (13–18 hrs)
- Maximize campaigns in high-order months: August, May, July
- Plan November promotions, leveraging past success (e.g., Black Friday bump)

Order Value Optimization

- Explore high average payment states: PB, AC, RO for premium offerings
- Offer premium bundles or loyalty perks targeting high-value regions

Inventory & Demand Planning

- Align inventory forecasting with 2018's 143% YoY order value growth
- Prioritize stock placement in high-demand states: SP, RJ, MG

Localized Operational Strategy

- Develop state-specific dashboards for delivery, payments, and order trends
- Assign regional leads to tailor strategies based on local performance