TARGET CASE STUDY

DESCRIPTION:

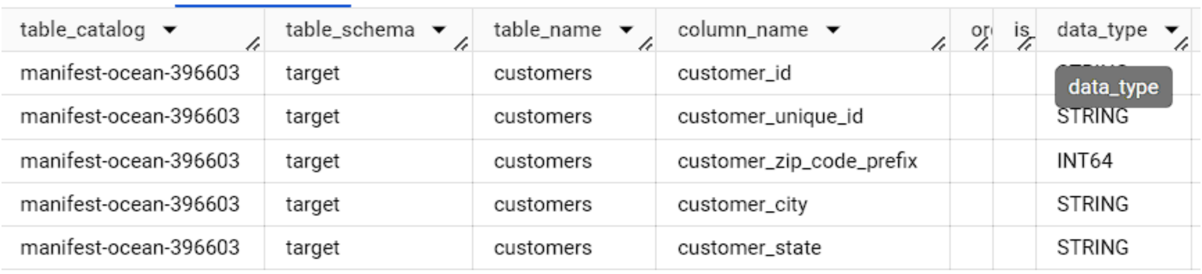
This case study uses SQL to analyze retail orders in Target, Brazil, focusing on trends over time, customer geography, delivery performance, and payment behavior. Results are reported with clear metric definitions and reproducible queries.

1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:

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

→ SELECT *

```
FROM manifest-ocean-396603-e5.target. INFORMATION_SCHEMA.COLUMNS
WHERE table_name = 'customers';
```

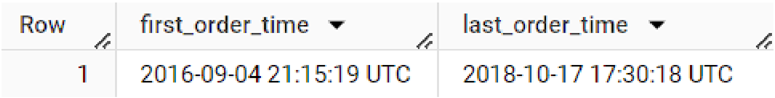


table_catalog	table_schema	table_name	column_name	or	is_	data_type
manifest-ocean-396603	target	customers	customer_id			STRING
manifest-ocean-396603	target	customers	customer_unique_id			STRING
manifest-ocean-396603	target	customers	customer_zip_code_prefix			INT64
manifest-ocean-396603	target	customers	customer_city			STRING
manifest-ocean-396603	target	customers	customer_state			STRING

1. Get the time range between which the orders were placed.

→ SELECT

```
MIN (order_purchase_timestamp) first_order_time,
MAX (order_purchase_timestamp) last_order_time
FROM `target.orders`;
```



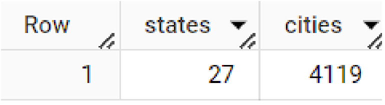
Row	first_order_time	last_order_time
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

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

→ SELECT

```
COUNT (DISTINCT c.customer_state) states,
COUNT (DISTINCT c.customer_city) cities
 FROM `target.customers` c
```

```
JOIN `target.orders` o
ON c.customer_id=o.customer_id;
```



Row	states	cities
1	27	4119

Observation :

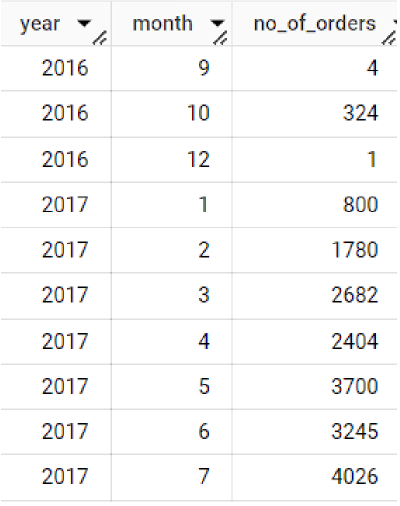
- Orders come from lots of cities across many states. This tells us the data covers most regions and sets us up for state-by-state views later

2. In-depth Exploration:

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

→

```
SELECT
EXTRACT (YEAR FROM order_purchase_timestamp) year,
EXTRACT (MONTH FROM order_purchase_timestamp) month,
COUNT (order_id) AS no_of_orders
FROM `target. orders`
GROUP BY year, month
ORDER BY year, month;
```



year	month	no_of_orders
2016	9	4
2016	10	324
2016	12	1
2017	1	800
2017	2	1780
2017	3	2682
2017	4	2404
2017	5	3700
2017	6	3245
2017	7	4026

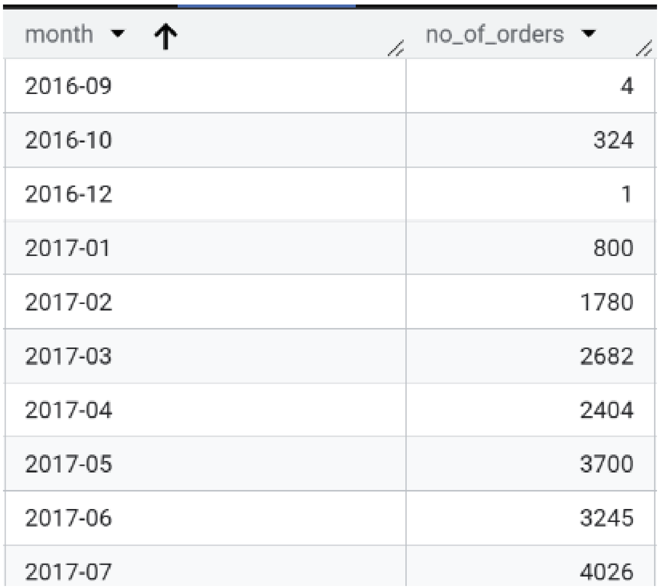
Observation :

- **Orders climb steadily from January 2017 onward. Missing entries in late-2016 look like gaps in the data, not a real dip in sales.**

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

→ SELECT

```
FORMAT_DATETIME('%Y-%m, order_purchase_timestamp) AS month,
COUNT(order_id) AS no_of_orders
FROM `target.orders`
GROUP BY month
ORDER BY  no_of_orders DESC, month;
```



month	no_of_orders
2016-09	4
2016-10	324
2016-12	1
2017-01	800
2017-02	1780
2017-03	2682
2017-04	2404
2017-05	3700
2017-06	3245
2017-07	4026

Observation :

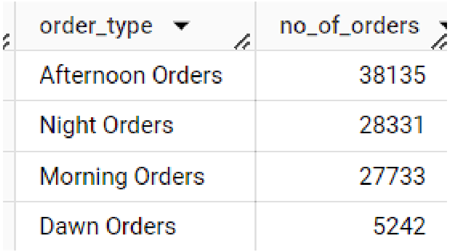
- **Across years, we see higher volumes in May, July, and August. This likely lines up with campaigns or seasonal demand.**

1. **During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night) time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)**
    1. **0-6 hrs: Dawn**
    2. **7-12 hrs: Mornings**
    3. **13-18 hrs: Afternoon**
    4. **19-23 hrs: Night**

→ SELECT

```
order_type,
COUNT(order_type) AS no_of_orders
```

```
FROM
 (SELECT
  CASE
        WHEN TIME(order_purchase_timestamp) BETWEEN '00:00:00' AND '06:59:59'
        THEN 'Dawn Orders'
        WHEN TIME(order_purchase_timestamp) BETWEEN '07:00:00' AND '12:59:59'
        THEN 'Morning Orders'
        WHEN TIME(order_purchase_timestamp) BETWEEN '13:00:00' AND '18:59:59'
        THEN 'Afternoon Orders'
        WHEN TIME(order_purchase_timestamp) BETWEEN '19:00:00' AND '23:59:59'
         THEN 'Night Orders'
   END order_type
   FROM `target.orders`)
   GROUP BY order_type
   ORDER BY no_of_orders DESC;
```



order_type	no_of_orders
Afternoon Orders	38135
Night Orders	28331
Morning Orders	27733
Dawn Orders	5242

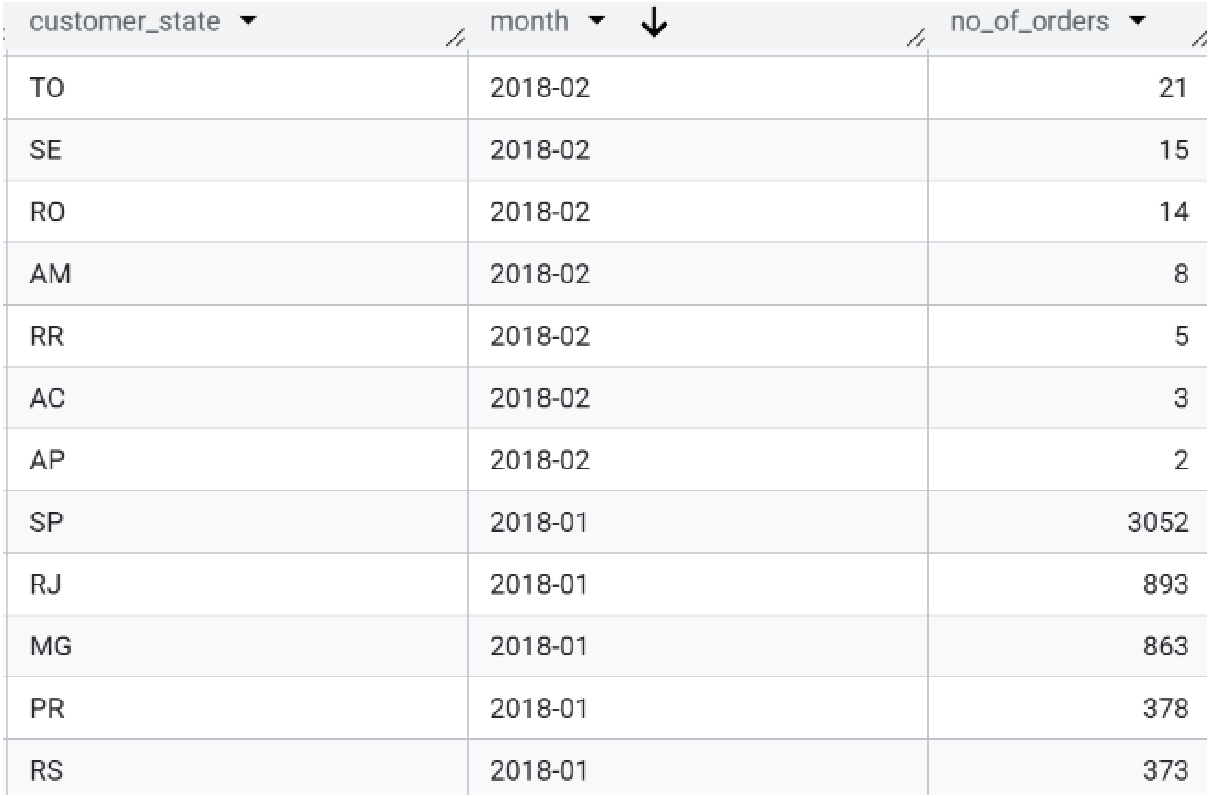
Observation :

- **Most orders land in the afternoon and evening. That’s when people are most active and these windows are useful for send-time optimization and capacity planning**

1. Evolution of E-commerce orders in the Brazil region:

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

```
→ SELECT

  DISTINCT c.customer_state,
  FORMAT_DATETIME(‘%Y-%m’, o.order_purchase_timestamp) month,
  COUNT(o.order_id) AS no_of_orders
   FROM `target.customers` c
   RIGHT JOIN `target.orders` o
   ON c.customer_id=o.customer_id
   GROUP BY c.customer_state, month
   ORDER BY no_of_orders DESC, c.customer_state;
```



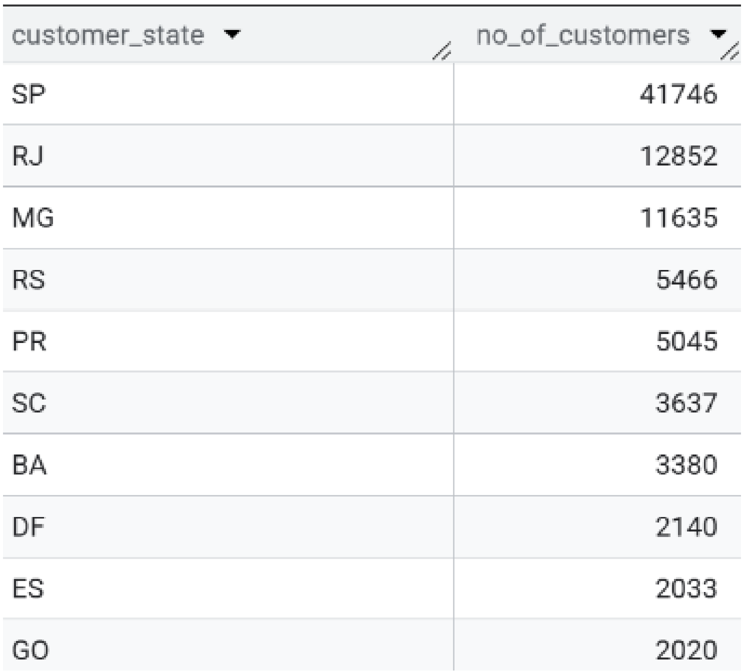
customer_state	month	no_of_orders
TO	2018-02	21
SE	2018-02	15
RO	2018-02	14
AM	2018-02	8
RR	2018-02	5
AC	2018-02	3
AP	2018-02	2
SP	2018-01	3052
RJ	2018-01	893
MG	2018-01	863
PR	2018-01	378
RS	2018-01	373

Observation :

- **Some states pick up faster than others month to month. This view helps decide where to place stock and delivery capacity.**

2. How are the customers distributed across all the states?

→
```
SELECT

customer_state,

COUNT(DISTINCT customer_id) AS no_of_customers
FROM `target.customers`
GROUP BY customer_state
ORDER BY no_of_customers DESC;
```



customer_state	no_of_customers
SP	41746
RJ	12852
MG	11635
RS	5466
PR	5045
SC	3637
BA	3380
DF	2140
ES	2033
GO	2020

<u>Observation</u> :

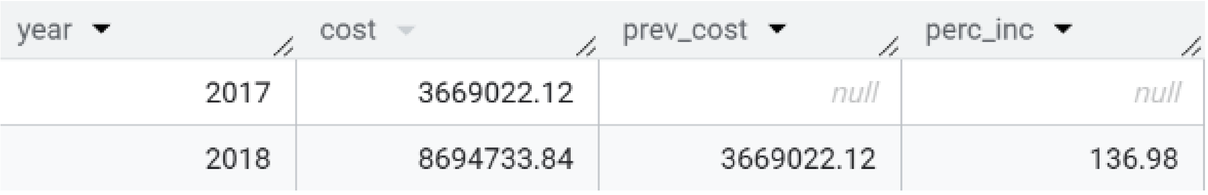
- **A few states hold most of the customers. This helps to target market development and service coverage.**

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

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 cte1 AS(SELECT *
  FROM `target.orders` o
  JOIN `target.payments` 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),
   cte2 AS (SELECT
   EXTRACT(YEAR FROM order_purchase_timestamp) AS year
   ROUND(SUM(payment_value),2) AS cost
```

```
    FROM cte1
    GROUP BY year)
    SELECT *, LAG(cost,1) OVER (ORDER BY year) AS prev_cost,
  ROUND(((cost - LAG(cost,1) OVER (ORDER BY year))*100 /LAG(cost,1) OVER
(ORDER BY year)),2)     AS perc_inc
  FROM cte2
  ORDER BY year;
```



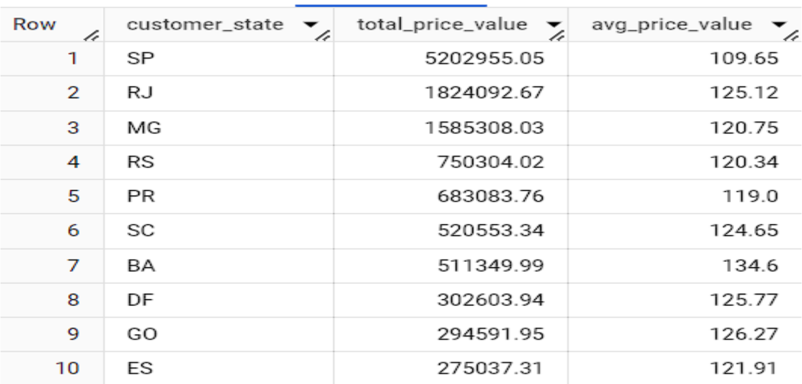
year	cost	prev_cost	perc_inc
2017	3669022.12	null	null
2018	8694733.84	3669022.12	136.98

Observation :

- **From January to August, total order cost is up year over year. Part of this is more orders; part could be bigger baskets**

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

```
→ SELECT
  c.customer_state,
  ROUND(SUM(ot.price),2) AS total_price_value,
  ROUND(SUM(ot.price)/COUNT(DISTINCT o.order_id),2) AS avg_price_value
  FROM `target.order_items` ot
  JOIN `target.orders` o
  ON ot.order_id=o.order_id
  JOIN `target.customers` c
  ON o.customer_id=c.customer_id
  GROUP BY c.customer_state
  ORDER BY total_price_value DESC;
```



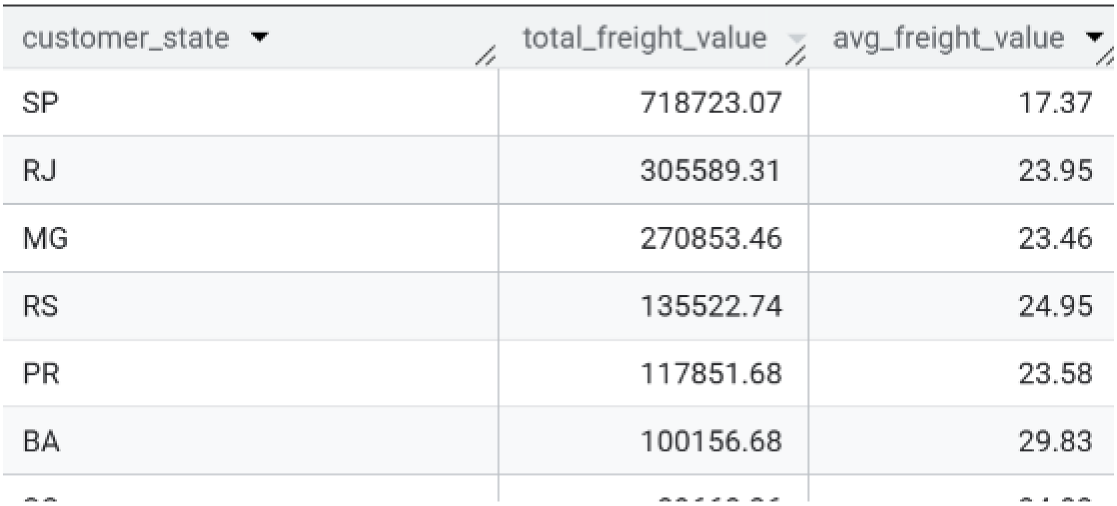
Row	customer_state	total_price_value	avg_price_value
1	SP	5202955.05	109.65
2	RJ	1824092.67	125.12
3	MG	1585308.03	120.75
4	RS	750304.02	120.34
5	PR	683083.76	119.0
6	SC	520553.34	124.65
7	BA	511349.99	134.6
8	DF	302603.94	125.77
9	GO	294591.95	126.27
10	ES	275037.31	121.91

Observation :

If a state has big totals but a middling average order value, it’s mainly volume-driven. High AOV(Average Over Value) with lower totals points to pricier baskets but fewer shoppers.

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

```
→ SELECT

    c.customer_state,
    ROUND(SUM(ot.freight_value),2) AS total_freight_value,
    ROUND(SUM(ot.freight_value)/COUNT(DISTINCT o.order_id),2) AS
avg_freight_value
    FROM `target.order_items` ot
    JOIN `target.orders` o
    ON ot.order_id=o.order_id
    JOIN `target.customers` c
    ON o.customer_id=c.customer_id
    GROUP BY c.customer_state
    ORDER BY total_freight_value DESC;
```



customer_state	total_freight_value	avg_freight_value
SP	718723.07	17.37
RJ	305589.31	23.95
MG	270853.46	23.46
RS	135522.74	24.95
PR	117851.68	23.58
BA	100156.68	29.83
[illegible]	[illegible]	[illegible]

Observation :

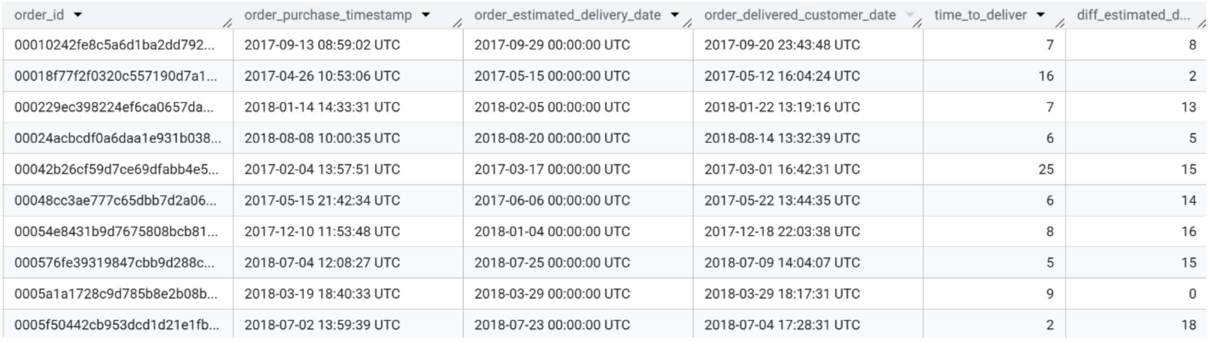
Average freight per order changes a lot by state. We can use this to pair it with delivery time to spot the routes that are both slow and expensive.

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

1. 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 formula:

time_to_deliver = order_delivered_customer_date - order_purchase_timestamp,
diff_estimated_delivery = order_estimated_delivery_date - order_delivered_customer_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_estimated_delivery_date,order_delivered_customer_date,DAY) AS diff_estimated_delivery
FROM `target.orders`
WHERE order_status = 'delivered'
ORDER BY order_id
```



order_id	order_purchase_timestamp	order_estimated_delivery_date	order_delivered_customer_date	time_to_deliver	diff_estimated_d...
00010242fe8c5a6d1ba2dd792...	2017-09-13 08:59:02 UTC	2017-09-29 00:00:00 UTC	2017-09-20 23:43:48 UTC	7	8
00018f77f2f0320c557190d7a1...	2017-04-26 10:53:06 UTC	2017-05-15 00:00:00 UTC	2017-05-12 16:04:24 UTC	16	2
000229ec398224ef6ca0657da...	2018-01-14 14:33:31 UTC	2018-02-05 00:00:00 UTC	2018-01-22 13:19:16 UTC	7	13
00024acbcdf0a6daa1e931b038...	2018-08-08 10:00:35 UTC	2018-08-20 00:00:00 UTC	2018-08-14 13:32:39 UTC	6	5
00042b26cf59d7ce69dfabb4e5...	2017-02-04 13:57:51 UTC	2017-03-17 00:00:00 UTC	2017-03-01 16:42:31 UTC	25	15
00048cc3ae777c65dbb7d2a06...	2017-05-15 21:42:34 UTC	2017-06-06 00:00:00 UTC	2017-05-22 13:44:35 UTC	6	14
00054e8431b9d7675808bcb81...	2017-12-10 11:53:48 UTC	2018-01-04 00:00:00 UTC	2017-12-18 22:03:38 UTC	8	16
000576fe39319847cbb9d288c...	2018-07-04 12:08:27 UTC	2018-07-25 00:00:00 UTC	2018-07-09 14:04:07 UTC	5	15
0005a1a1728c9d785b8e2b08b...	2018-03-19 18:40:33 UTC	2018-03-29 00:00:00 UTC	2018-03-29 18:17:31 UTC	9	0
0005f50442cb953dcd1d21e1fb...	2018-07-02 13:59:39 UTC	2018-07-23 00:00:00 UTC	2018-07-04 17:28:31 UTC	2	18

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

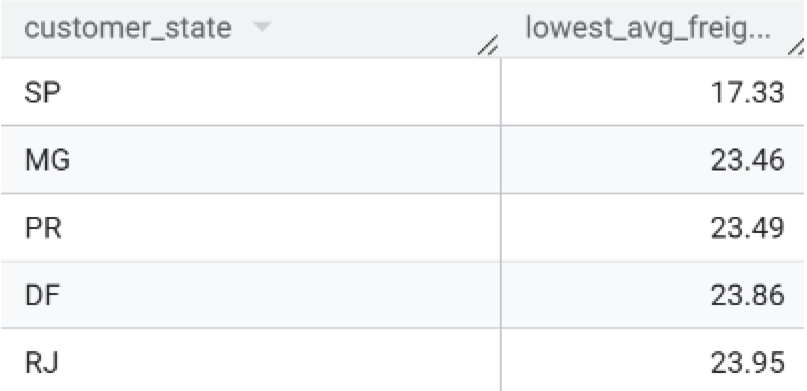
➔ **-- Lowest 5**

```
SELECT

c.customer_state,

ROUND((SUM(ot.freight_value)/COUNT(DISTINCT o.order_id)),2) AS lowest_avg_freight_val

FROM `target.order_items` AS ot

JOIN `target.orders` AS o

ON ot.order_id = o.order_id

JOIN `target.customers` c
```

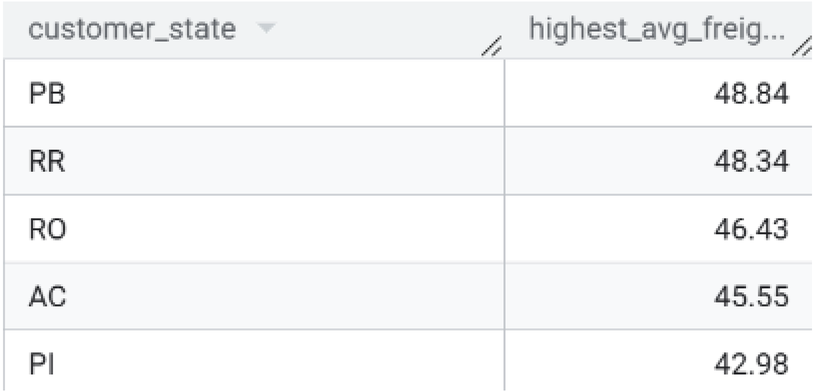
```
      ON o.customer_id = c.customer_id
      WHERE o.order_status = 'delivered'
    GROUP BY c.customer_state
    ORDER BY lowest_avg_freight_val
    LIMIT 5
```

-- Highest 5

```
  SELECT
  c.customer_state,
  ROUND(SUM(ot.freight_value)/COUNT(DISTINCT o.order_id),2) AS
highest_avg_freight_val
  FROM `target.order_items` AS ot
  JOIN `target.orders` AS o
  ON ot.order_id=o.order_id
  JOIN `target.customers` c
 ON o.customer_id=c.customer_id
 WHERE o.order_status = 'delivered'
 GROUP BY c.customer_state
 ORDER BY highest_avg_freight_val DESC
 LIMIT 5;
```



customer_state	lowest_avg_freig...
SP	17.33
MG	23.46
PR	23.49
DF	23.86
RJ	23.95

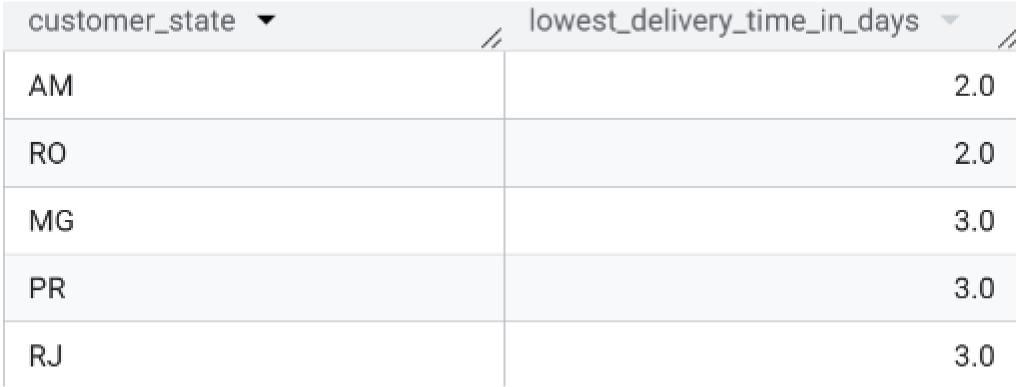


customer_state	highest_avg_freig...
PB	48.84
RR	48.34
RO	46.43
AC	45.55
PI	42.98

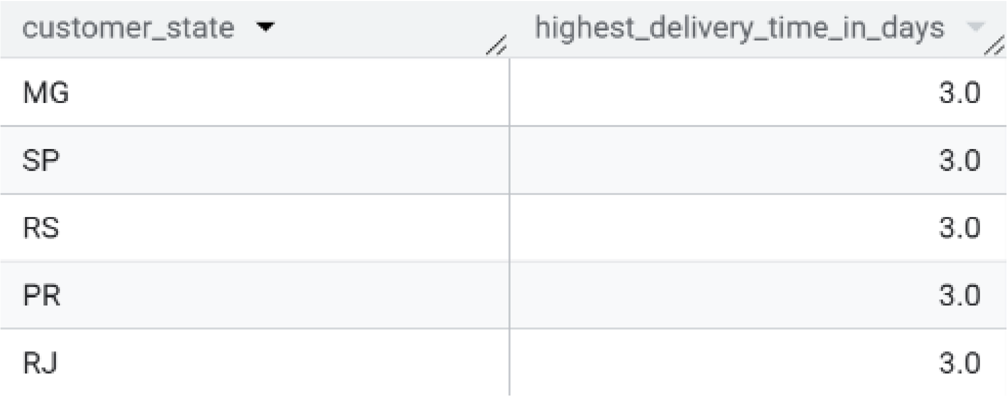
3. Find out the top 5 states with the highest & lowest average delivery time.

→ SELECT

```
    c.customer_state,
ROUND((SUM(DATE_DIFF(o.order_delivered_carrier_date,o.order_purchase_timestamp,DAY))/COUNT(DISTINCT o.order_id)),0) AS lowest_delivery_time_in_days,

FROM `target.orders` o

JOIN `target.customers` c

ON o.customer_id = c.customer_id

WHERE o.order_status = 'delivered'

GROUP BY c.customer_state

ORDER BY lowest_delivery_time_in_days

LIMIT 5;

-- Top 5 States with highest delivery time

SELECT

c.customer_state,

ROUND((SUM(DATE_DIFF(o.order_delivered_carrier_date,o.order_purchase_timestamp,DAY))/COUNT(DISTINCT o.order_id)),0) AS highest_delivery_time_in_days,

FROM `target.orders` o

JOIN `target.customers` c

ON o.customer_id = c.customer_id

WHERE o.order_status = 'delivered'

GROUP BY c.customer_state

ORDER BY highest_delivery_time_in_days DESC

LIMIT 5;
```



customer_state	lowest_delivery_time_in_days
AM	2.0
RO	2.0
MG	3.0
PR	3.0
RJ	3.0

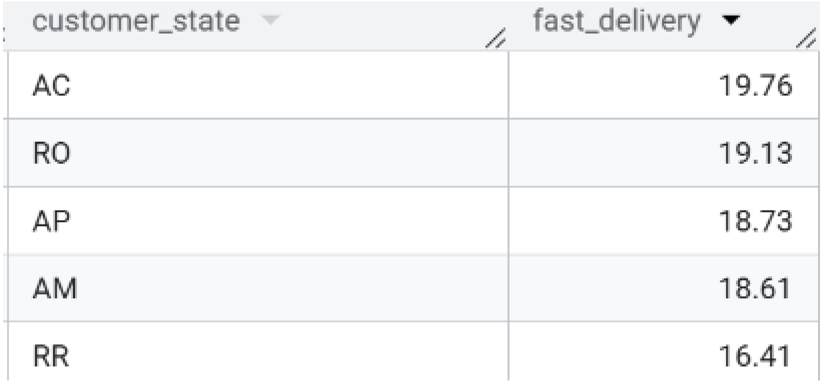


customer_state	highest_delivery_time_in_days
MG	3.0
SP	3.0
RS	3.0
PR	3.0
RJ	3.0

3. **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.**

→

```
SELECT
c.customer_state,
ROUND(SUM(DATE_DIFF(o.order_estimated_delivery_date,o.order_delivered_customer_date,DAY)/COUNT(DISTINCt o.order_id)),2) AS fast_delivery
FROM `target.orders`o
JOIN `target.customers` c
ON o.customer_id=c.customer_id
WHERE o.order_status='delivered'
GROUP BY c.customer_state
ORDER BY fast_delivery DESC
LIMIT 5;
```



customer_state	fast_delivery
AC	19.76
RO	19.13
AP	18.73
AM	18.61
RR	16.41

Observation :

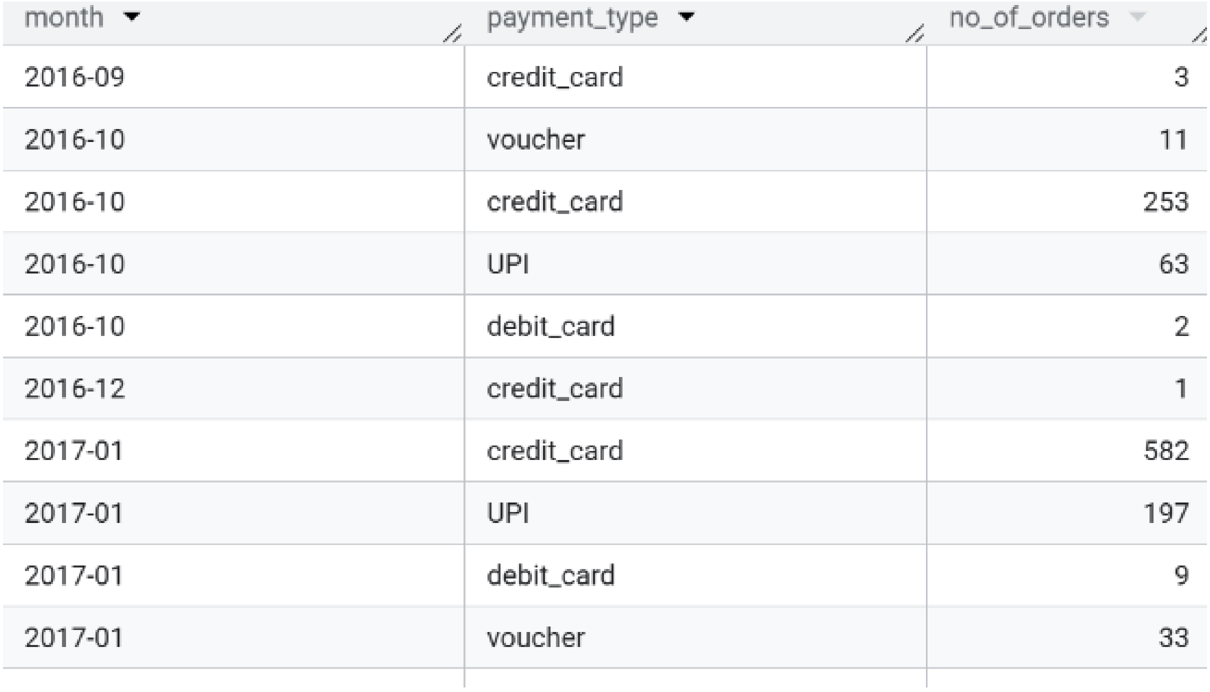
Most deliveries arrive within a similar number of days. If “estimate minus actual” is positive, we beat the estimate; if it’s negative, we were late

6. Analysis based on the payments:

1. Find the month-on-month no. of orders placed using different payment types.

→

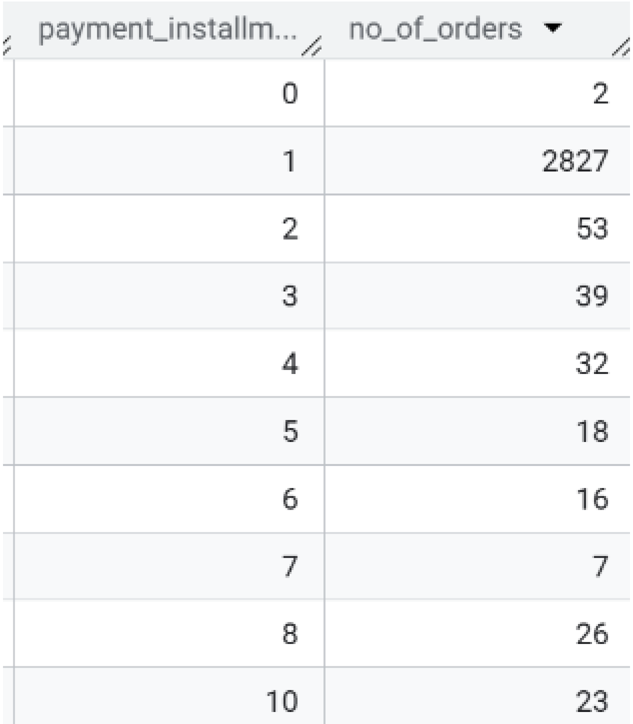
```
SELECT
FORMAT_DATETIME('%Y-%m', o.order_purchase_timestamp) AS month,
p.payment_type,
COUNT(DISTINCT o.order_id) AS no_of_orders
FROM `target.orders` o
JOIN `target.payments`p
ON o.order_id = p.order_id
GROUP BY month,p.payment_type
ORDER BY month
```



month	payment_type	no_of_orders
2016-09	credit_card	3
2016-10	voucher	11
2016-10	credit_card	253
2016-10	UPI	63
2016-10	debit_card	2
2016-12	credit_card	1
2017-01	credit_card	582
2017-01	UPI	197
2017-01	debit_card	9
2017-01	voucher	33

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

```
→ SELECT

    payment_installments,
    COUNT(DISTINCT order_id) AS no_of_orders,
    FROM `target.payments`
    WHERE payment_sequential>=1
    GROUP BY payment_installments;
```



payment_installm...	no_of_orders ▾
0	2
1	2827
2	53
3	39
4	32
5	18
6	16
7	7
8	26
10	23

Observation :

Payment mix shifts a bit over time. Installments can lift conversions but they also delay cash; so we need to watch the number of installments against basket size.

Overall Observation and Insights :

- Orders trend upward overall, with noticeable mid-year peaks.
- A handful of states account for most activity.
- Afternoon and evening are the busiest ordering windows.
- Delivery is generally consistent; a few routes run slower than the rest.
- Payment mix shifts over time; installments help conversion but spread receipts out.