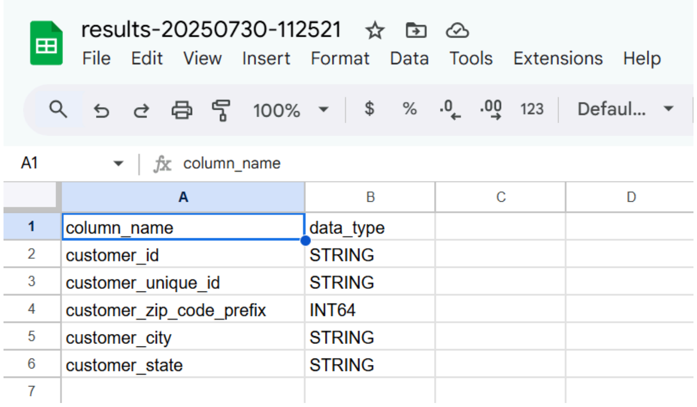
1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:

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

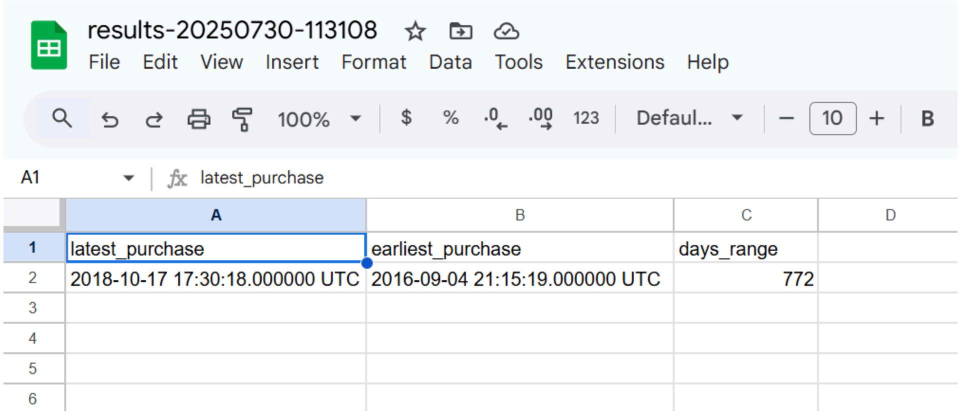
```
          SELECT column_name, data_type FROM
dsml-sql-462709.Target.INFORMATION_SCHEMA.COLUMNS
          where table_name='customers'
```


results-20250730-112521
File Edit View Insert Format Data Tools Extensions Help
100% $ % .0 .00 123 Defaul...
A1
column_name

column_name	data_type
customer_id	STRING
customer_unique_id	STRING
customer_zip_code_prefix	INT64
customer_city	STRING
customer_state	STRING

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

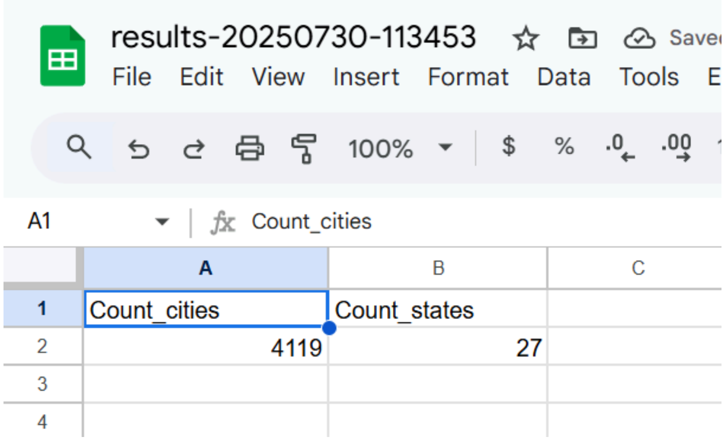
```
SELECT max(order_purchase_timestamp) as latest_purchase, min(order_purchase_timestamp)
as earliest_purchase,
date_diff(max(order_purchase_timestamp), min(order_purchase_timestamp), day) as
days_range FROM `Target.orders`
```


results-20250730-113108
File Edit View Insert Format Data Tools Extensions Help
100% $ % .0 .00 123 Defaul... − 10 + B
A1
latest_purchase

latest_purchase	earliest_purchase	days_range
2018-10-17 17:30:18.000000 UTC	2016-09-04 21:15:19.000000 UTC	772

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

```
SELECT count(distinct(customer_city)) as Count_cities,
count(distinct(customer_state)) as Count_states FROM Target.customers
```

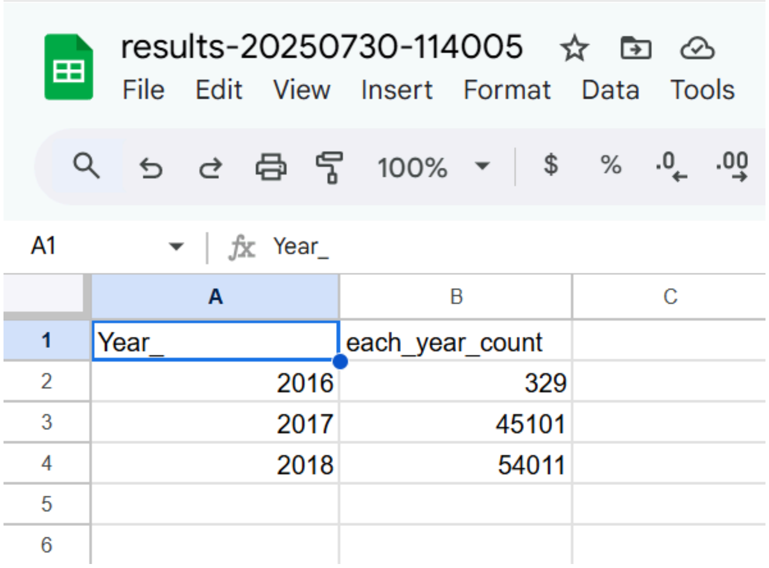


	A	B	C
1	Count_cities	Count_states	
2	4119	27	
3			
4			

2. In-depth Exploration:

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

```
SELECT extract(year from order_purchase_timestamp) as Year_, count(*)
each_year_count FROM `Target.orders` group by Year_ order by Year_ asc
```

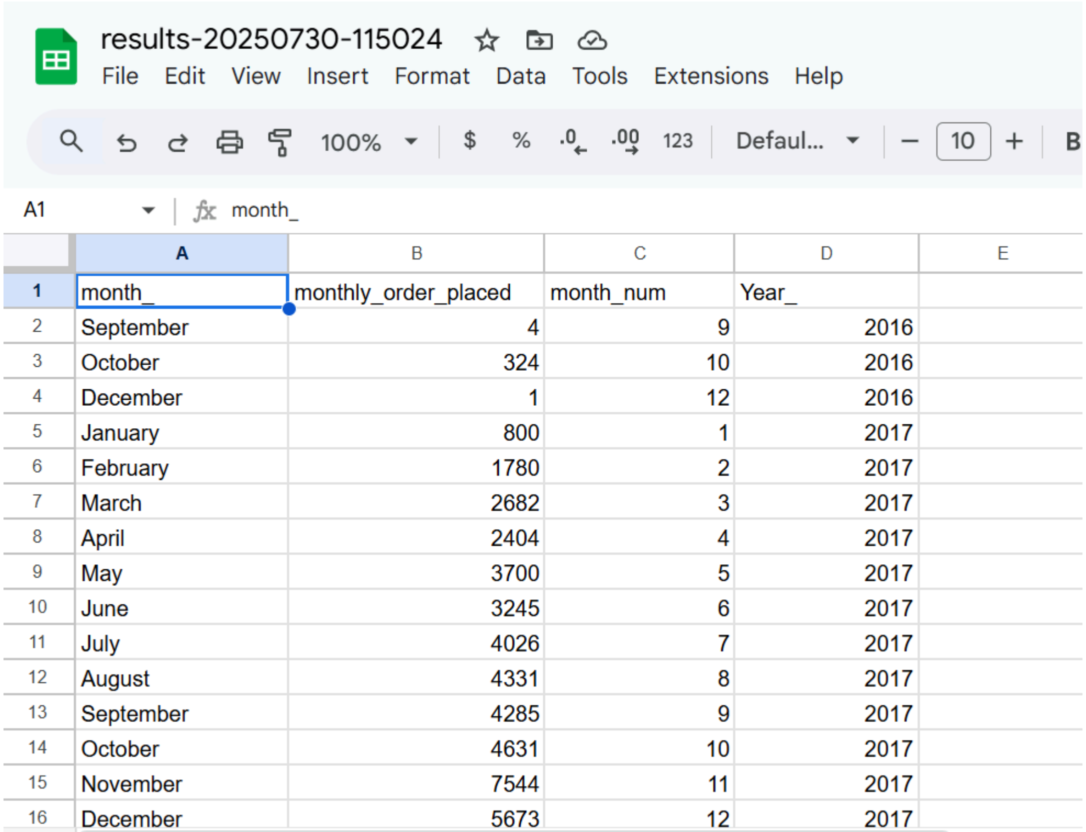


	A	B	C
1	Year_	each_year_count	
2	2016	329	
3	2017	45101	
4	2018	54011	
5			
6			

The output reveals that, post-2016, the number of orders placed increases exponentially, and after 2017, the orders placed continue to rise, although the difference between 2016 and 2017 is significantly larger than the difference between 2017 and 2018.

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

```
SELECT format_datetime("%B", order_purchase_timestamp) as month_, count(*) as
monthly_order_placed, extract(month from order_purchase_timestamp) as
month_num,
 extract(year from order_purchase_timestamp) as Year_ FROM Target.orders group
by month_, month_num, year_ order by year_ asc, month_num asc
```



	month_	monthly_order_placed	month_num	Year_
2	September	4	9	2016
3	October	324	10	2016
4	December	1	12	2016
5	January	800	1	2017
6	February	1780	2	2017
7	March	2682	3	2017
8	April	2404	4	2017
9	May	3700	5	2017
10	June	3245	6	2017
11	July	4026	7	2017
12	August	4331	8	2017
13	September	4285	9	2017
14	October	4631	10	2017
15	November	7544	11	2017
16	December	5673	12	2017

The output shows that 2016 had significantly fewer orders, whereas in 2017 the order count steadily increased each month. Some unusual trends were also observed in August and September, as well as in November and December. In December, the order count can be revived by using the “New Year event” and strategically linking it with new attractive offers.

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

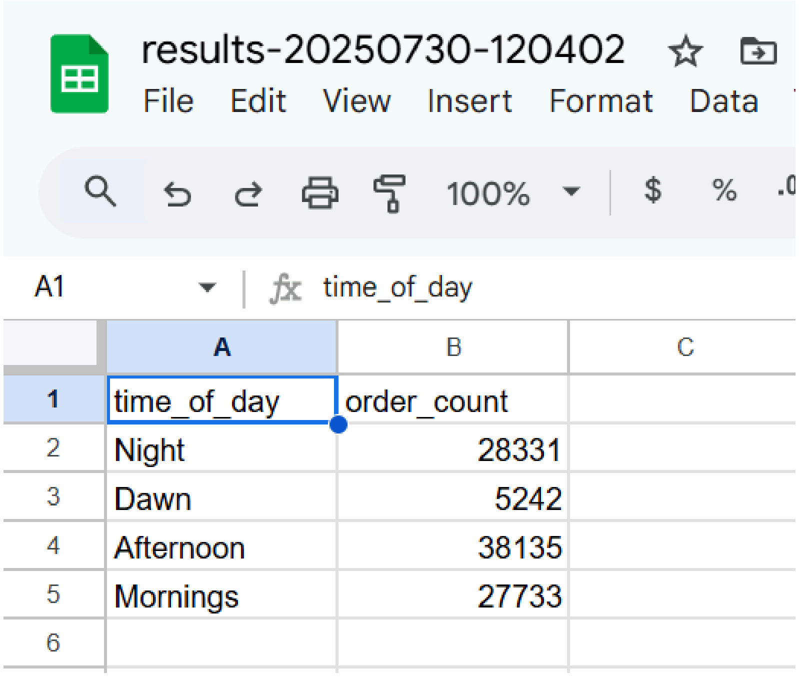
a. 0-6 hrs : Dawn

b. 7-12 hrs : Mornings

c. 13-18 hrs : Afternoon

d. 19-23 hrs : Night

```
SELECT (CASE WHEN hour_order>=0 and hour_order<=6 then "Dawn"
WHEN hour_order>=7 and hour_order <=12 then "Mornings"
WHEN hour_order>=13 and hour_order<=18 then "Afternoon"
WHEN hour_order>=19 and hour_order<=23 then "Night"
END) time_of_day, count(*) order_count FROM
(SELECT extract(hour from order_purchase_timestamp) hour_order FROM
Target.orders)
group by time_of_day
```



A1 time_of_day

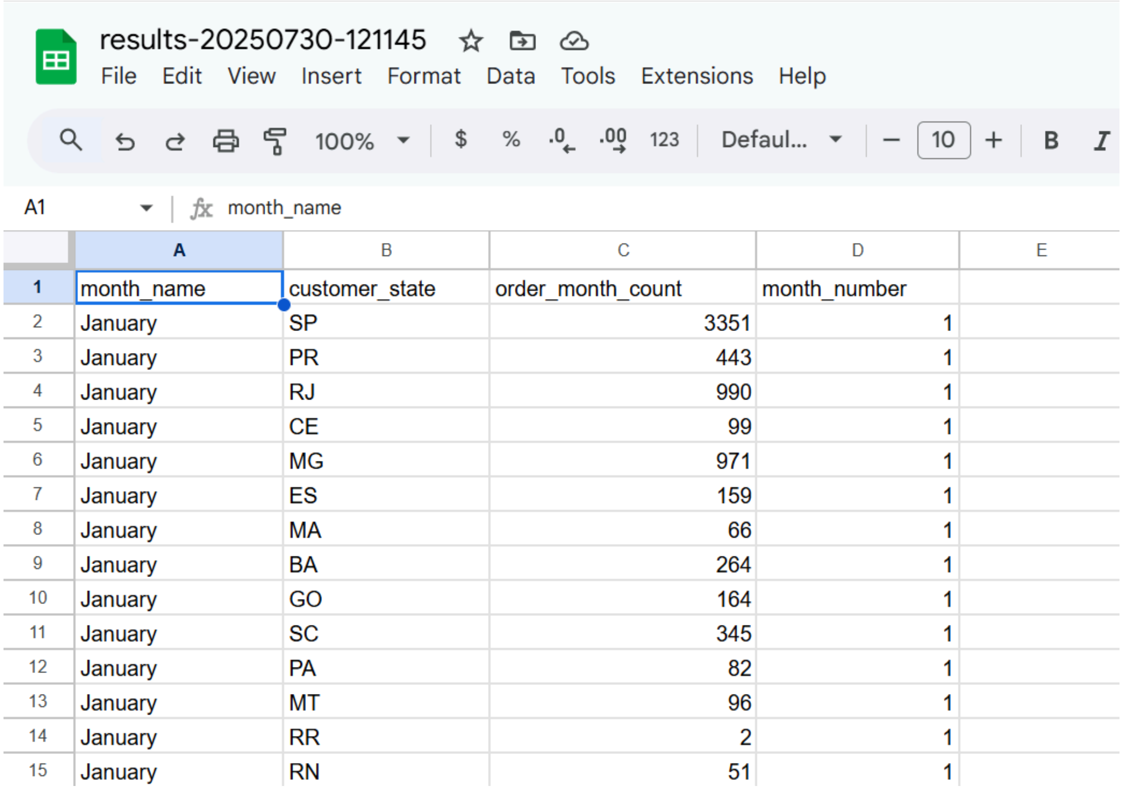
time_of_day	order_count
Night	28331
Dawn	5242
Afternoon	38135
Mornings	27733

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

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

```
SELECT format_datetime("%B", T_o.order_purchase_timestamp) as month_name,
T_c.customer_state, count(*)
```

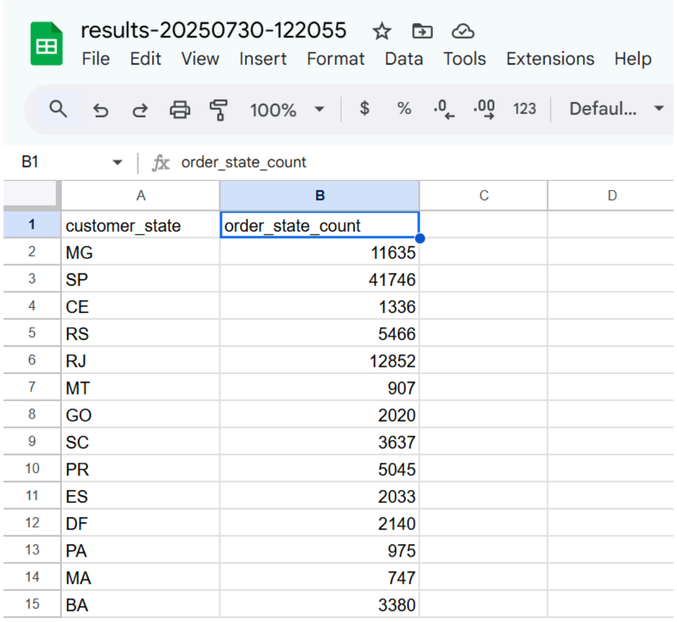
```
as order_month_count, extract(month from T_o.order_purchase_timestamp) as
month_number FROM Target.orders as T_o join `Target.customers` as T_c on
T_o.customer_id=T_c.customer_id
group by month_name, month_number, T_c.customer_state
order by month_number asc
```



	A	B	C	D	E
1	month_name	customer_state	order_month_count	month_number	
2	January	SP	3351	1	
3	January	PR	443	1	
4	January	RJ	990	1	
5	January	CE	99	1	
6	January	MG	971	1	
7	January	ES	159	1	
8	January	MA	66	1	
9	January	BA	264	1	
10	January	GO	164	1	
11	January	SC	345	1	
12	January	PA	82	1	
13	January	MT	96	1	
14	January	RR	2	1	
15	January	RN	51	1	

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

```
SELECT T_c.customer_state, count(*)
as order_state_count FROM Target.orders as T_o join `Target.customers` as T_c
on
T_o.customer_id=T_c.customer_id
group by T_c.customer_state
```



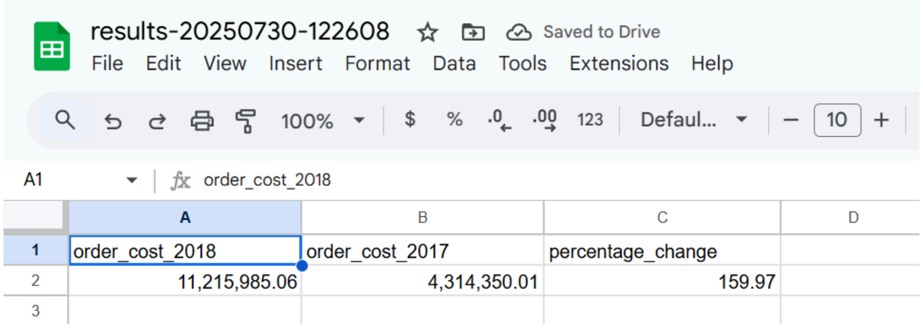
order_state_count

customer_state	order_state_count
MG	11635
SP	41746
CE	1336
RS	5466
RJ	12852
MT	907
GO	2020
SC	3637
PR	5045
ES	2033
DF	2140
PA	975
MA	747
BA	3380

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.

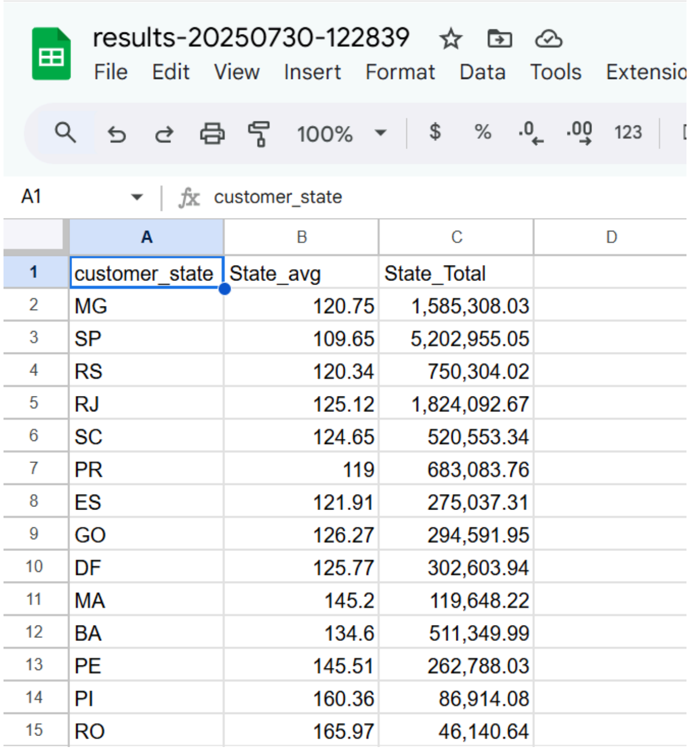
```
SELECT Round(max(Order_cost), 2) order_cost_2018 , Round(min(Order_cost), 2)
order_cost_2017, Round
(((max(Order_cost)-min(Order_cost))/min(Order_cost)*100), 2) as
percentage_change FROM
(
SELECT extract(year from T_oi.shipping_limit_date) as Year_, SUM(payment_value)
Order_cost FROM Target.order_items as T_oi  join `Target.payments` as T_py on
 T_oi.order_id=T_py.order_id
 WHERE (extract(month from T_oi.shipping_limit_date)) between 1 and 8 AND
extract(year from T_oi.shipping_limit_date) between 2017 and 2018
 group by Year_
)
```



	A	B	C	D
1	order_cost_2018	order_cost_2017	percentage_change	
2	11,215,985.06	4,314,350.01	159.97	
3				

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

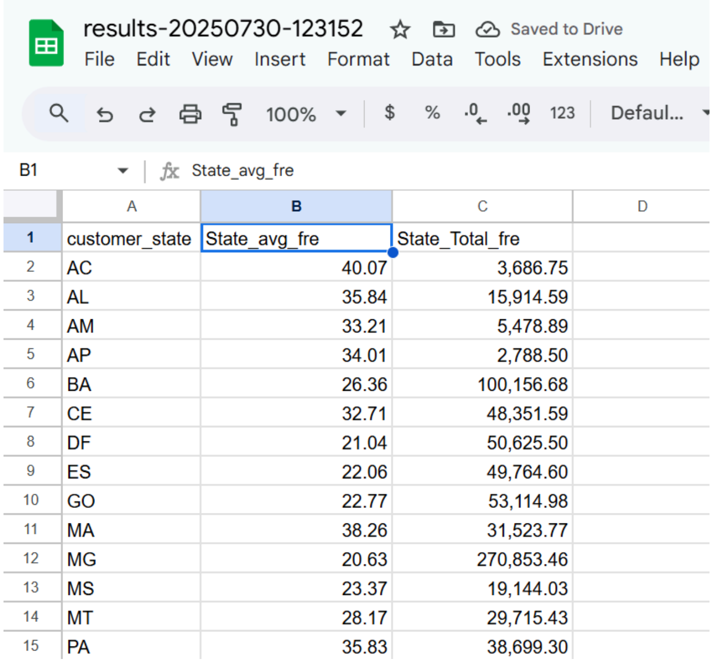
```
SELECT customer_state, Round(avg(price), 2) as State_avg, Round(sum(price),2)
as State_Total
 FROM `Target.customers` as T_cu  join `Target.orders` as T_or on
T_cu.customer_id=T_or.customer_id join `Target.order_items` as T_oi on
T_or.order_id=T_oi.order_id
Group by customer_state
```



	A	B	C	D
1	customer_state	State_avg	State_Total	
2	MG	120.75	1,585,308.03	
3	SP	109.65	5,202,955.05	
4	RS	120.34	750,304.02	
5	RJ	125.12	1,824,092.67	
6	SC	124.65	520,553.34	
7	PR	119	683,083.76	
8	ES	121.91	275,037.31	
9	GO	126.27	294,591.95	
10	DF	125.77	302,603.94	
11	MA	145.2	119,648.22	
12	BA	134.6	511,349.99	
13	PE	145.51	262,788.03	
14	PI	160.36	86,914.08	
15	RO	165.97	46,140.64	

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

```
SELECT customer_state, Round(avg(freight_value), 2) as State_avg_fre,
Round(sum(freight_value),2) as State_Total_fre
 FROM `Target.customers` as T_cu  join `Target.orders` as T_or on
T_cu.customer_id=T_or.customer_id join `Target.order_items` as T_oi on
T_or.order_id=T_oi.order_id
Group by customer_state
```



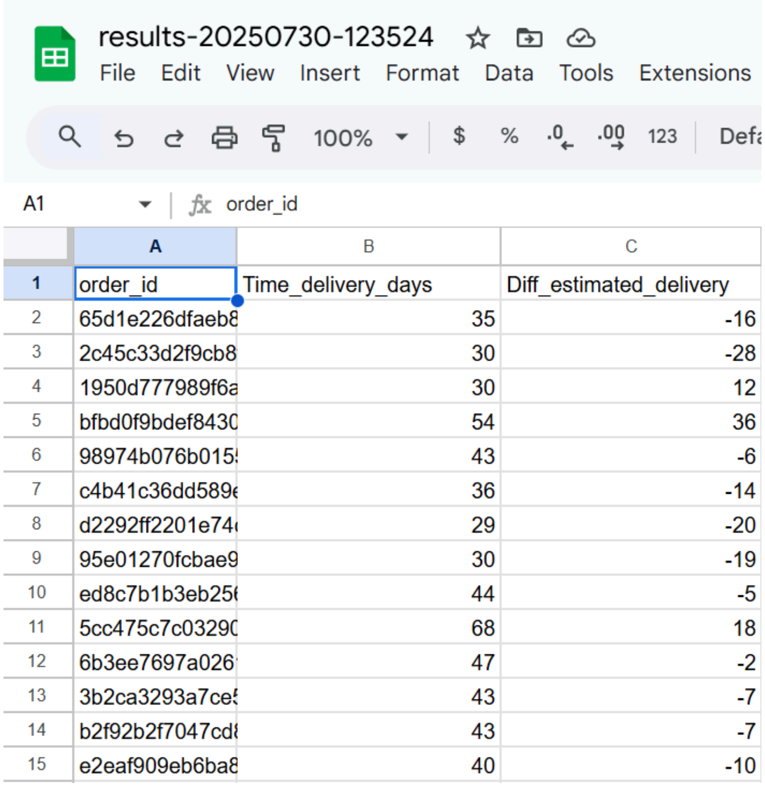
customer_state	State_avg_fre	State_Total_fre
AC	40.07	3,686.75
AL	35.84	15,914.59
AM	33.21	5,478.89
AP	34.01	2,788.50
BA	26.36	100,156.68
CE	32.71	48,351.59
DF	21.04	50,625.50
ES	22.06	49,764.60
GO	22.77	53,114.98
MA	38.26	31,523.77
MG	20.63	270,853.46
MS	23.37	19,144.03
MT	28.17	29,715.43
PA	35.83	38,699.30

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:
   a. time_to_deliver = order_delivered_customer_date - order_purchase_timestamp
   b. diff_estimated_delivery = order_delivered_customer_date - order_estimated_delivery_date

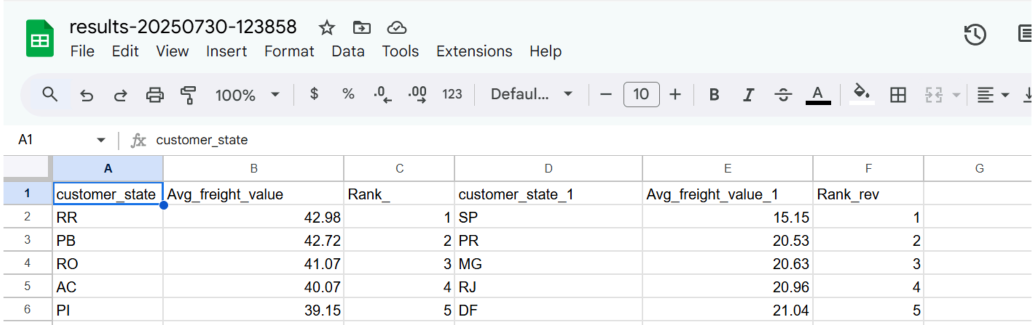
```
SELECT order_id, date_diff(order_delivered_customer_date,
order_purchase_timestamp, day) as Time_delivery_days,
date_diff(order_delivered_customer_date, order_estimated_delivery_date,day) as
Diff_estimated_delivery
FROM `Target.orders`
group by order_id, Time_delivery_days,Diff_estimated_delivery
```



order_id	Time_delivery_days	Diff_estimated_delivery
65d1e226dfaeb8	35	-16
2c45c33d2f9cb8	30	-28
1950d777989f6a	30	12
bfbd0f9bdef8430	54	36
98974b076b015	43	-6
c4b41c36dd589	36	-14
d2292ff2201e74	29	-20
95e01270fcbae9	30	-19
ed8c7b1b3eb25	44	-5
5cc475c7c0329	68	18
6b3ee7697a026	47	-2
3b2ca3293a7ce	43	-7
b2f92b2f7047cd	43	-7
e2eaf909eb6ba8	40	-10

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

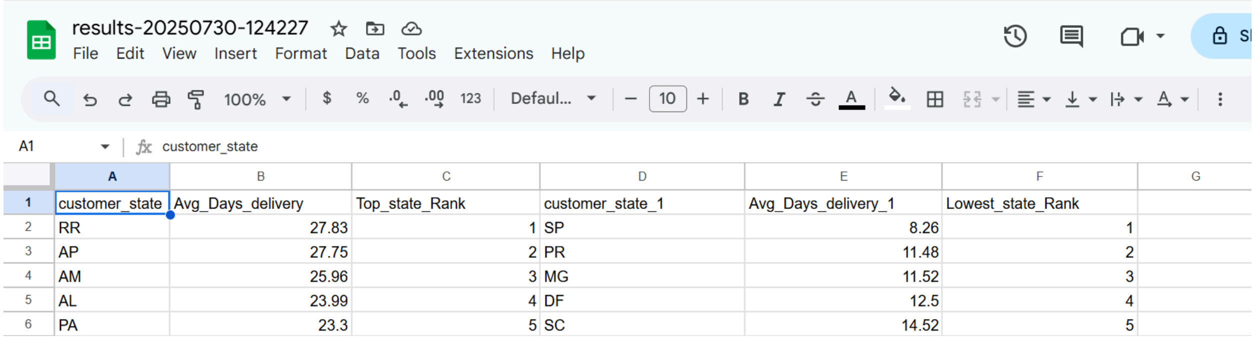
```
SELECT * FROM

(SELECT T_c.customer_state, Round(avg(T_oi.freight_value), 2) as
Avg_freight_value, (dense_rank() over(order by avg(T_oi.freight_value) desc) )
as Rank_
FROM `Target.customers` as T_c join `Target.orders` as T_o on
T_c.customer_id=T_o.customer_id join `Target.order_items` as T_oi on
T_o.order_id=T_oi.order_id

group by T_c.customer_state
order by Avg_freight_value desc
limit 5) as X

JOIN

(SELECT T_c.customer_state, Round(avg(T_oi.freight_value), 2) as
Avg_freight_value, (dense_rank() over(order by avg(T_oi.freight_value) asc) )
as Rank_rev
FROM `Target.customers` as T_c join `Target.orders` as T_o on
T_c.customer_id=T_o.customer_id join `Target.order_items` as T_oi on
T_o.order_id=T_oi.order_id

group by T_c.customer_state

order by Avg_freight_value asc
limit 5 ) as Y

on X.Rank_=Y.Rank_rev
```



customer_state	Avg_freight_value	Rank_	customer_state_1	Avg_freight_value_1	Rank_rev
RR	42.98	1	SP	15.15	1
PB	42.72	2	PR	20.53	2
RO	41.07	3	MG	20.63	3
AC	40.07	4	RJ	20.96	4
PI	39.15	5	DF	21.04	5

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

```
SELECT * FROM

(SELECT T_c.customer_state,
Round(avg(date_diff(T_o.order_delivered_customer_date,
T_o.order_purchase_timestamp, day)), 2) as Avg_Days_delivery,(dense_rank()
over(order by (Round(avg(date_diff(T_o.order_delivered_customer_date,
T_o.order_purchase_timestamp, day)), 2)) desc))as Top_state_Rank FROM
`Target.customers` as T_c join `Target.orders` as T_o on
T_c.customer_id=T_o.customer_id join `Target.order_items` as T_oi
on T_o.order_id=T_oi.order_id
group by T_c.customer_state
order by Avg_Days_delivery desc
limit 5
 ) as X

Join

(SELECT T_c.customer_state,
Round(avg(date_diff(T_o.order_delivered_customer_date,
T_o.order_purchase_timestamp, day)), 2) as Avg_Days_delivery,
(dense_rank() over(order by
(Round(avg(date_diff(T_o.order_delivered_customer_date,
T_o.order_purchase_timestamp, day)), 2)) asc))as Lowest_state_Rank
 FROM `Target.customers` as T_c join `Target.orders` as T_o on
T_c.customer_id=T_o.customer_id join `Target.order_items` as T_oi
on T_o.order_id=T_oi.order_id
group by T_c.customer_state
order by Avg_Days_delivery asc
limit 5) as Y


on X.Top_state_Rank=Y.Lowest_state_Rank
```

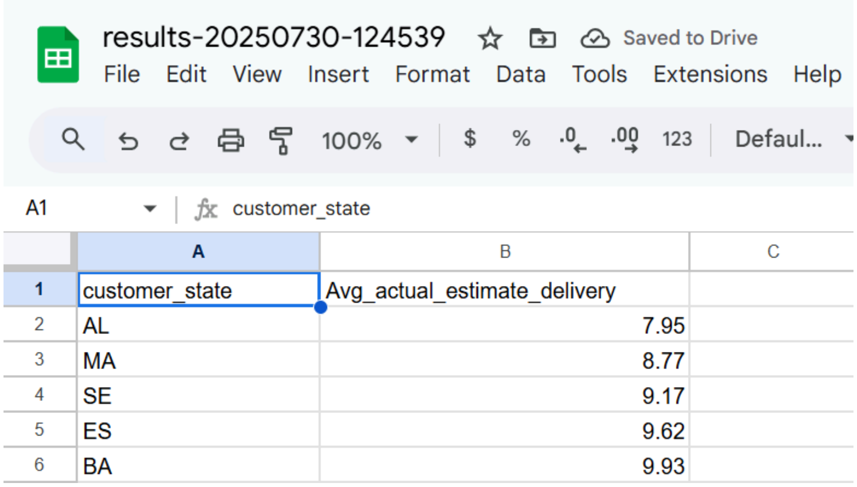


	A	B	C	D	E	F	G
1	customer_state	Avg_Days_delivery	Top_state_Rank	customer_state_1	Avg_Days_delivery_1	Lowest_state_Rank	
2	RR	27.83	1	SP	8.26	1	
3	AP	27.75	2	PR	11.48	2	
4	AM	25.96	3	MG	11.52	3	
5	AL	23.99	4	DF	12.5	4	
6	PA	23.3	5	SC	14.52	5	

4. 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 avg_actual_estimate as (
  SELECT T_cu.customer_state , Round(avg(date_diff(T_or.order_estimated_delivery_date,
T_or.order_delivered_customer_date, day)),2) as Avg_actual_estimate_delivery
```

```
    FROM `Target.customers` as T_cu join Target.orders as T_or on
T_cu.customer_id=T_or.customer_id
    GROUP BY T_cu.customer_state
    order by Avg_actual_estimate_delivery asc
    limit 5
)

SELECT * FROM avg_actual_estimate
```

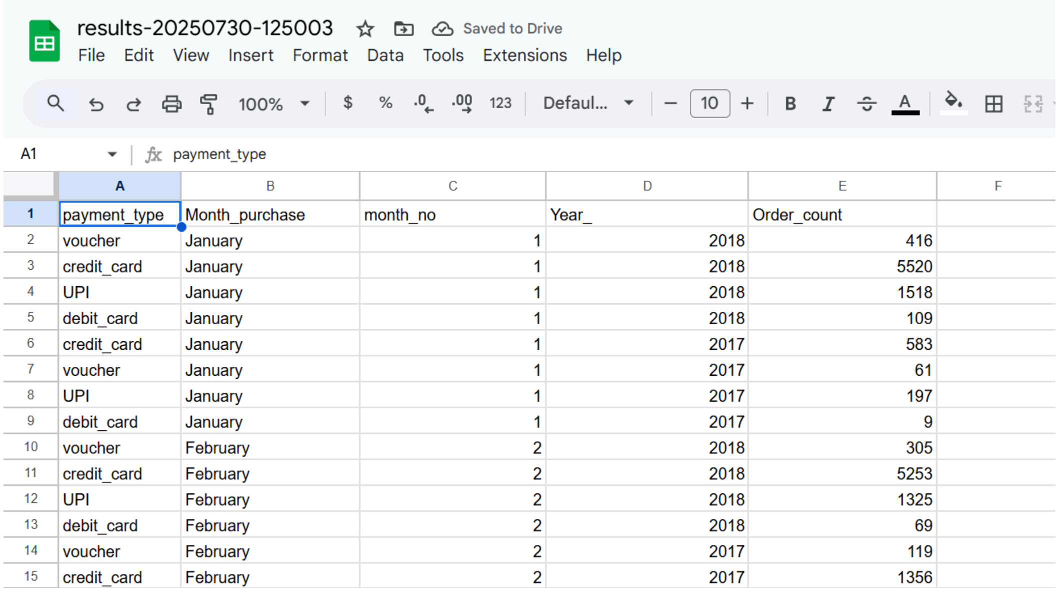


	A	B	C
1	customer_state	Avg_actual_estimate_delivery	
2	AL	7.95	
3	MA	8.77	
4	SE	9.17	
5	ES	9.62	
6	BA	9.93	

6. Analysis based on the payments:

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

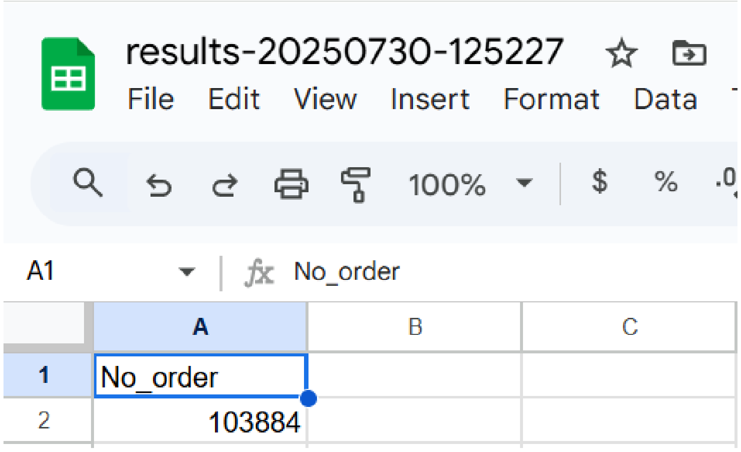
```
SELECT T_py.payment_type, format_datetime("%B", T_or.order_purchase_timestamp) as
Month_purchase,
  extract(month FROM T_or.order_purchase_timestamp) as month_no,
  extract(year FROM T_or.order_purchase_timestamp) as Year_,
    count(*) as Order_count
 FROM `Target.payments` as T_py join Target.orders as T_or
  on T_py.order_id=T_or.order_id
 group by Month_purchase,month_no,T_py.payment_type, Year_
 order by month_no asc, Year_ desc
```



	A	B	C	D	E	F
1	payment_type	Month_purchase	month_no	Year_	Order_count	
2	voucher	January	1	2018	416	
3	credit_card	January	1	2018	5520	
4	UPI	January	1	2018	1518	
5	debit_card	January	1	2018	109	
6	credit_card	January	1	2017	583	
7	voucher	January	1	2017	61	
8	UPI	January	1	2017	197	
9	debit_card	January	1	2017	9	
10	voucher	February	2	2018	305	
11	credit_card	February	2	2018	5253	
12	UPI	February	2	2018	1325	
13	debit_card	February	2	2018	69	
14	voucher	February	2	2017	119	
15	credit_card	February	2	2017	1356	

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

```
SELECT Count(*) as No_order FROM `Target.payments` where payment_installments>=1
```



	A	B	C
1	No_order		
2	103884		