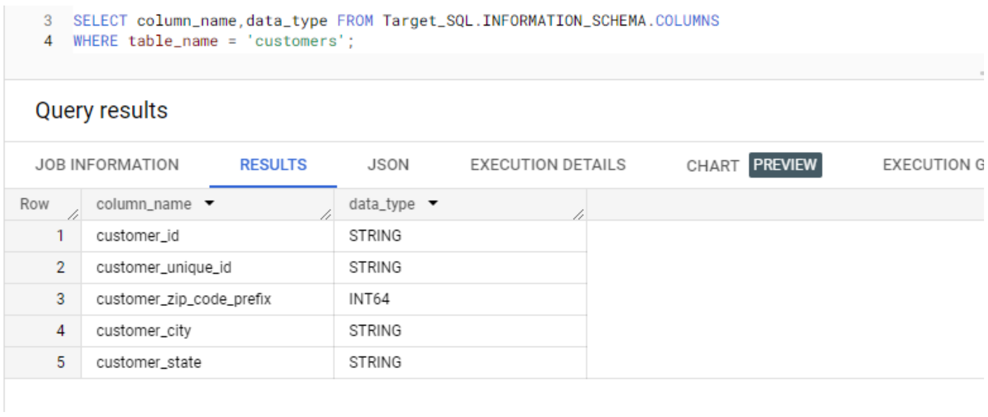
Business Case: Target SQL

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

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

   Query: SELECT column_name,data_type FROM Target_SQL.INFORMATION_SCHEMA.COLUMNS WHERE table_name = 'customers' ;



```
SELECT column_name,data_type FROM Target_SQL.INFORMATION_SCHEMA.COLUMNS
WHERE table_name = 'customers';
```

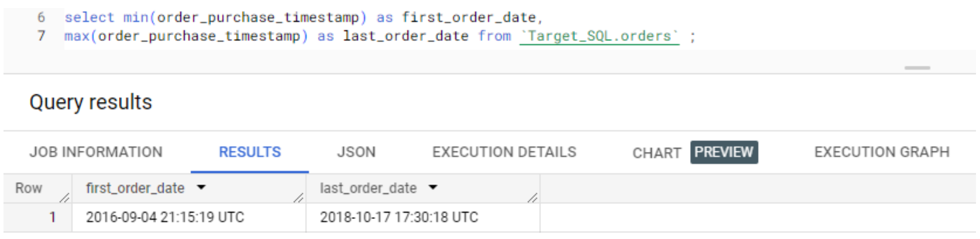
Query results

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

Inference: It is identified that zip_code is the only numeric column,Rest all including customer_id and customer_unique_id is of type string.

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

Query: select min(order_purchase_timestamp) as first_order_date,max(order_purchase_timestamp) as last_order_date from `Target_SQL.orders` ;



```
select min(order_purchase_timestamp) as first_order_date,
max(order_purchase_timestamp) as last_order_date from `Target_SQL.orders` ;
```

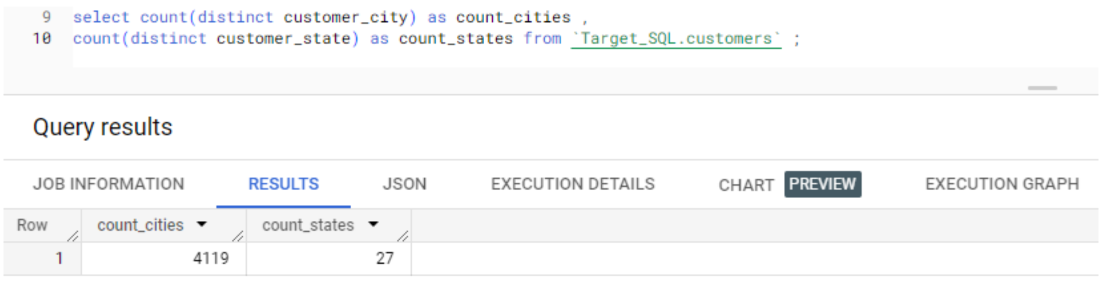
Query results

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

Inference: It is clear from the output that the data provided has the orders placed by the customers between 4th Sep 2016 and 17th Oct 2018.

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

Query: select count(distinct customer_city) as count_cities , count(distinct customer_state) as count_states from `Target_SQL.customers` ;



```
select count(distinct customer_city) as count_cities ,
count(distinct customer_state) as count_states from `Target_SQL.customers` ;
```

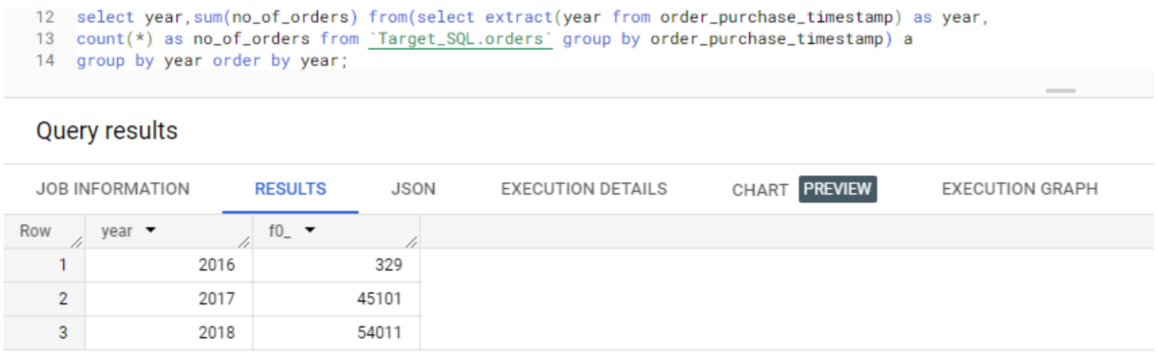
Row	count_cities	count_states
1	4119	27

Inference : It is observed that the customers who placed the orders span across 4119 cities within 27 states from Brazil.

2. **In-depth Exploration:**

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

Query: select year,sum(no_of_orders) from(select extract(year from order_purchase_timestamp) as year, count(*) as no_of_orders from `Target_SQL.orders` group by order_purchase_timestamp) a
group by year order by year;



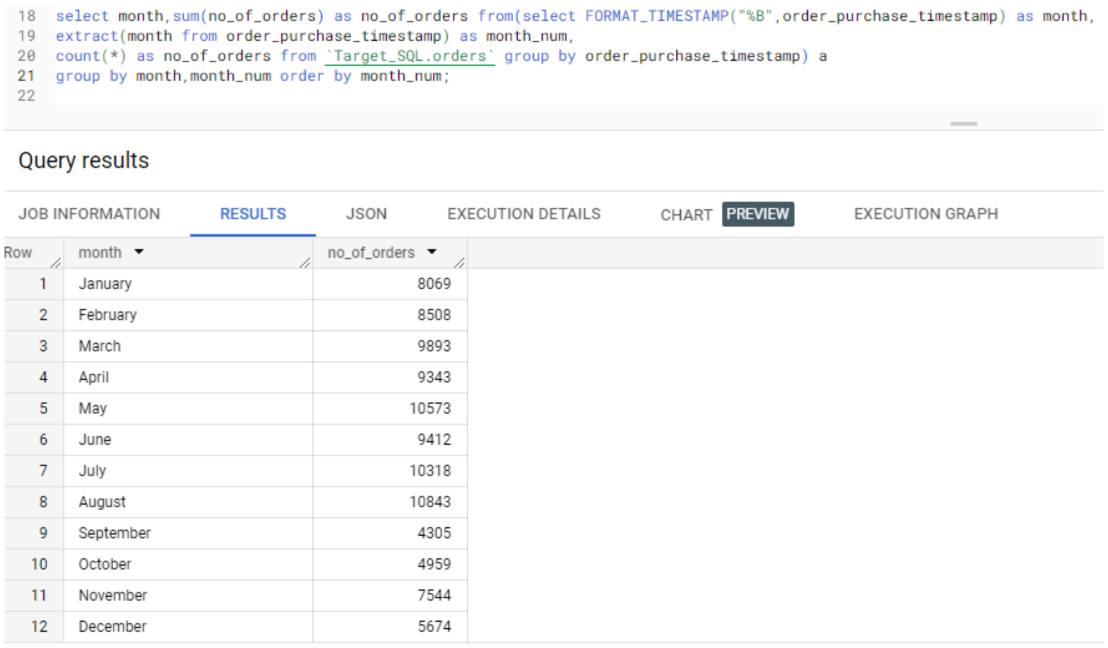
```
select year,sum(no_of_orders) from(select extract(year from order_purchase_timestamp) as year,
count(*) as no_of_orders from `Target_SQL.orders` group by order_purchase_timestamp) a
group by year order by year;
```

Row	year	f0_
1	2016	329
2	2017	45101
3	2018	54011

Inference : Here, for almost the last 4 months of 2016, a total of 329 orders were placed. For the next full year of 2017, 45101 orders were placed and in 2018, for the first 10 months, 54011 orders were placed. So it is quite obvious from the data that the no.of orders being placed is being increased year by year. From 2016 to 2017 there was a huge increase, i.e, almost 83 orders per month to 3759 orders per month on an average. And from 2017 to 2018 the no. of orders placed goes from 3759 to 5686 per month on an average, which again is an increase but not as huge as that of 2016 to 2017.

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

Query: select month,sum(no_of_orders) as no_of_orders from(select FORMAT_TIMESTAMP("%B",order_purchase_timestamp) as month, extract(month from order_purchase_timestamp) as month_num,
count(*) as no_of_orders from `Target_SQL.orders` group by order_purchase_timestamp) a
group by month,month_num order by month_num;



```
select month,sum(no_of_orders) as no_of_orders from(select FORMAT_TIMESTAMP("%B",order_purchase_timestamp) as month,
extract(month from order_purchase_timestamp) as month_num,
count(*) as no_of_orders from `Target_SQL.orders` group by order_purchase_timestamp) a
group by month,month_num order by month_num;
```

Query results

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

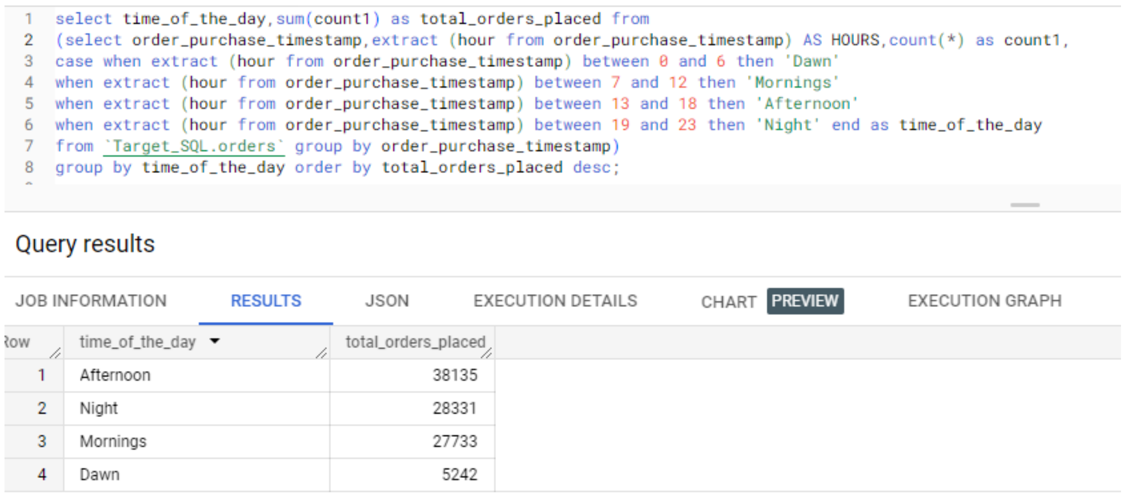
Row	month	no_of_orders
1	January	8069
2	February	8508
3	March	9893
4	April	9343
5	May	10573
6	June	9412
7	July	10318
8	August	10843
9	September	4305
10	October	4959
11	November	7544
12	December	5674

Inference: Actually we are considering almost 2.5 months of data for September and October : half-month data from 2016 and full-month data from 2017 and 2018 for September, full-month data from 2016 and 2017 and half-month data from 2018 for October. For other months across the entire dataset we are only considering only 2 months' worth of data. Even after considering 2.5 months of data for September and October, those are the months with very few orders being placed. August, May and July are the months with exceptionally high number of orders being placed. Following in rank for the no. of orders being placed are March, June, April,February, January, November, December in that respective order. The 1st, 2nd and third quarters are doing good compared to the last quarter.

Ques 3: 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**

Query: select time_of_the_day, sum(count1) as total_orders_placed from
(select order_purchase_timestamp, extract (hour from order_purchase_timestamp)
AS HOURS, count(*) as count1,
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'
when extract (hour from order_purchase_timestamp) between 19 and 23 then 'Night'
end as time_of_the_day from `Target_SQL.orders` group by order_purchase_timestamp)
group by time_of_the_day order by total_orders_placed desc;



```
select time_of_the_day,sum(count1) as total_orders_placed from
(select order_purchase_timestamp,extract (hour from order_purchase_timestamp) AS HOURS,count(*) as count1,
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'
when extract (hour from order_purchase_timestamp) between 19 and 23 then 'Night' end as time_of_the_day
from `Target_SQL.orders` group by order_purchase_timestamp)
group by time_of_the_day order by total_orders_placed desc;
```

Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS CHART PREVIEW EXECUTION GRAPH

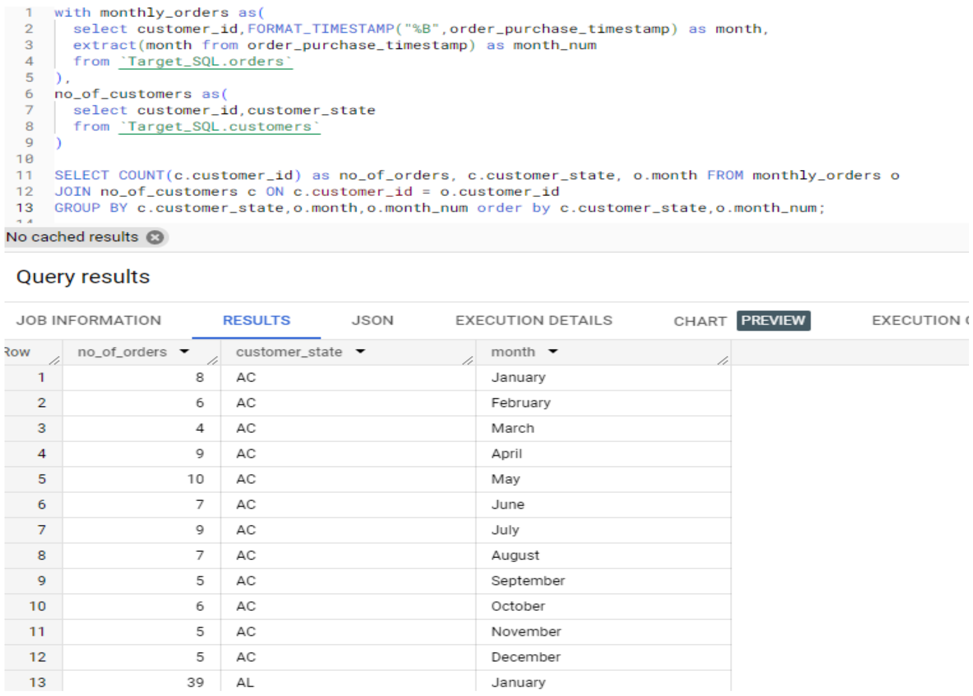
Row	time_of_the_day	total_orders_placed
1	Afternoon	38135
2	Night	28331
3	Mornings	27733
4	Dawn	5242

Inference: Due to Brazil's division into 4 different time zones within its borders, we are considering the UTC time zone for the purpose of standardization. After analyzing our data, it is evident that people in Brazil tend to place their orders primarily during the afternoon. Conversely, the fewest orders are placed during the Dawn hours.In the night time and morning, a comparable number of orders are being placed.

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

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

Query: with monthly_orders as(select customer_id,
FORMAT_TIMESTAMP("%B",order_purchase_timestamp) as month,
extract(month from order_purchase_timestamp) as month_num from `Target_SQL.orders`) ,
no_of_customers as(select customer_id,customer_state from `Target_SQL.customers`)
SELECT COUNT(c.customer_id) as no_of_orders, c.customer_state, o.month FROM
monthly_orders JOIN no_of_customers c ON c.customer_id = o.customer_id
GROUP BY c.customer_state,o.month,o.month_num order by c.customer_state,
o.month_num;



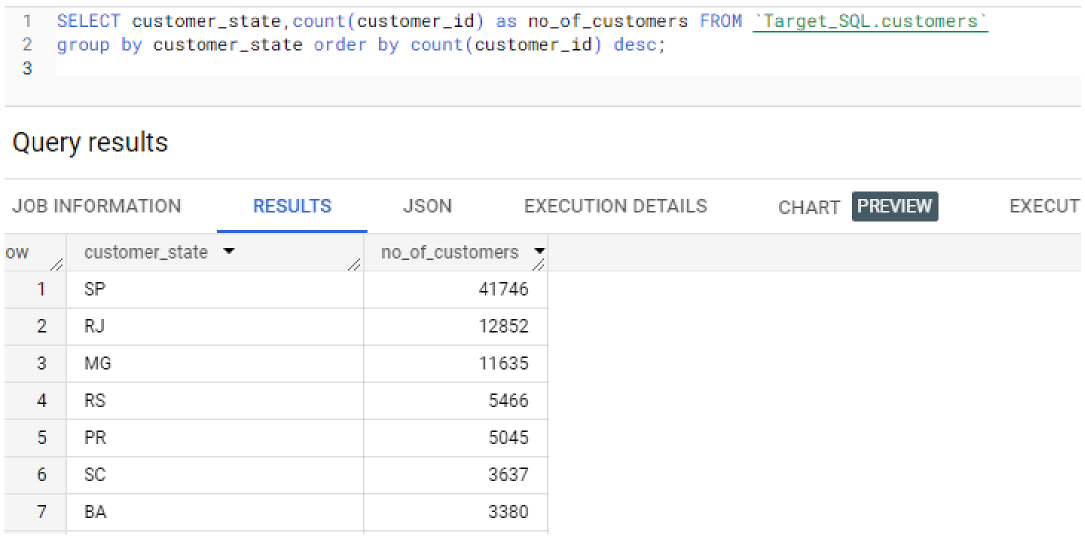
```
with monthly_orders as(
  select customer_id,FORMAT_TIMESTAMP("%B",order_purchase_timestamp) as month,
  extract(month from order_purchase_timestamp) as month_num
  from `Target_SQL.orders`
),
no_of_customers as(
  select customer_id,customer_state
  from `Target_SQL.customers`
)

SELECT COUNT(c.customer_id) as no_of_orders, c.customer_state, o.month FROM monthly_orders o
JOIN no_of_customers c ON c.customer_id = o.customer_id
GROUP BY c.customer_state,o.month,o.month_num order by c.customer_state,o.month_num;
```

Row	no_of_orders	customer_state	month
1	8	AC	January
2	6	AC	February
3	4	AC	March
4	9	AC	April
5	10	AC	May
6	7	AC	June
7	9	AC	July
8	7	AC	August
9	5	AC	September
10	6	AC	October
11	5	AC	November
12	5	AC	December
13	39	AL	January

Inference: The observation from the data indicates that, across almost all the states, the second quadrimester consistently exhibits higher number of placed orders in comparison to both the first and third quadrimesters, while quadrimester the third has the least placed orders. Another interesting finding was that, for the month of May, the majority of states had the highest number of placed orders. On the flip side, for most states, September recorded the lowest number of orders placed.

Ques 2: How are the customers distributed across all the states?

Query: SELECT customer_state,count(customer_id) as no_of_customers FROM `Target_SQL.customers` group by customer_state order by count(customer_id) desc;



```
SELECT customer_state,count(customer_id) as no_of_customers FROM `Target_SQL.customers`
group by customer_state order by count(customer_id) desc;
```

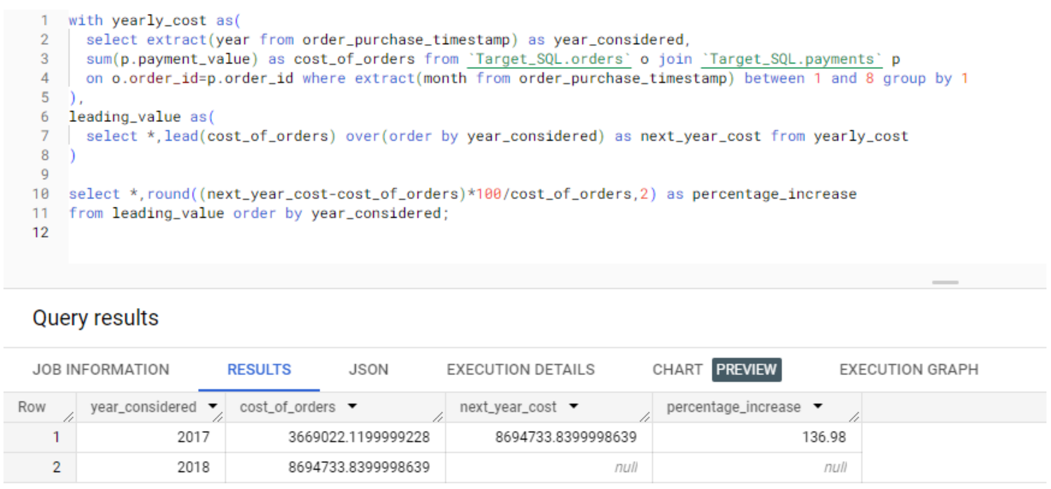
ow	customer_state	no_of_customers
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637
7	BA	3380

Inference: The data reveals a clear trend : nearly half (41,746) of the customers are from the state of SP, while only a handful are from AC (81), AP (68), and RR (46). Nearly two-thirds of the customers come from just three states: SP, RJ, and MG. The remaining 24 states collectively account for one-third of the total customers.

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

Ques 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.

Query: with yearly_cost as(select extract(year from order_purchase_timestamp) as year_considered,
sum(p.payment_value) as cost_of_orders from `Target_SQL.orders` o join
`Target_SQL.payments` p on o.order_id=p.order_id where extract(month from
order_purchase_timestamp) between 1 and 8
group by 1),
leading_value as(select *,lead(cost_of_orders) over(order by year_considered) as
next_year_cost
from yearly_cost)
select *,round((next_year_cost-cost_of_orders)*100/cost_of_orders,2) as
percentage_increase from leading_value order by year_considered;



```
with yearly_cost as(
  select extract(year from order_purchase_timestamp) as year_considered,
  sum(p.payment_value) as cost_of_orders from `Target_SQL.orders` o join `Target_SQL.payments` p
  on o.order_id=p.order_id where extract(month from order_purchase_timestamp) between 1 and 8 group by 1
),
leading_value as(
  select *,lead(cost_of_orders) over(order by year_considered) as next_year_cost from yearly_cost
)

select *,round((next_year_cost-cost_of_orders)*100/cost_of_orders,2) as percentage_increase
from leading_value order by year_considered;
```

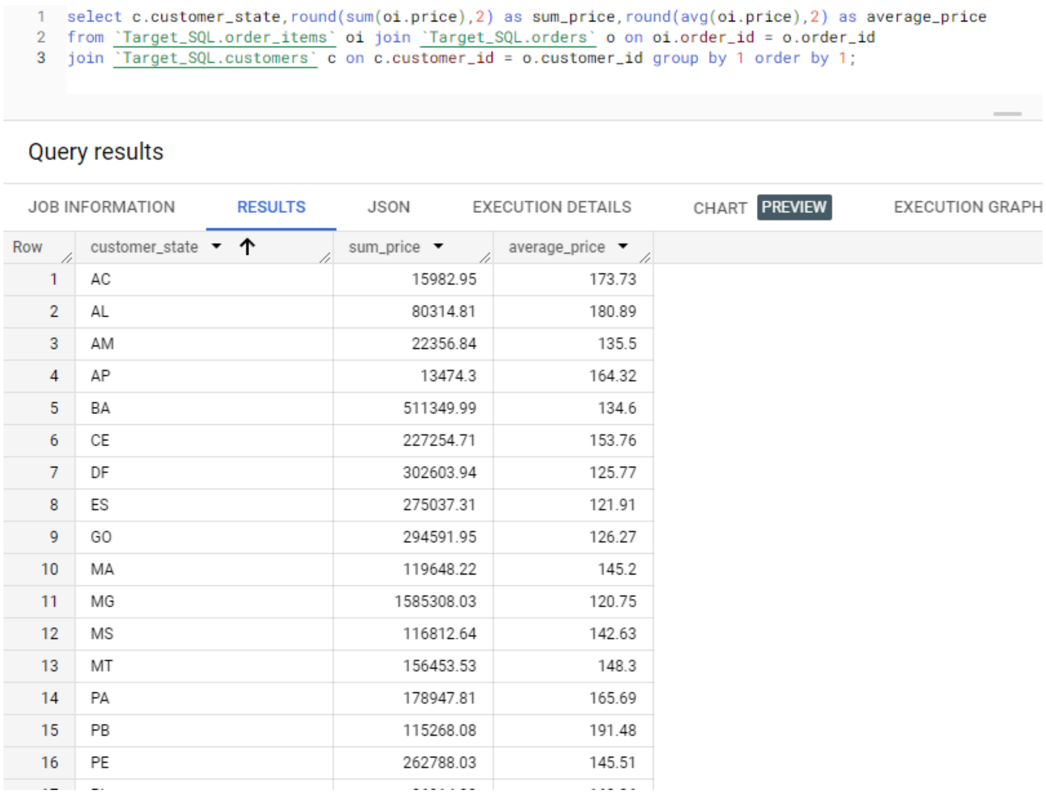
Query results

Row	year_considered	cost_of_orders	next_year_cost	percentage_increase
1	2017	3669022.1199999228	8694733.8399998639	136.98
2	2018	8694733.8399998639	null	null

Inference: After reviewing the data from January to August in both 2017 and 2018, it is evident that the total cost of orders has surged by a substantial 136.98%. This sharp increase suggests notable changes in spending patterns during these periods.

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

Query: select c.customer_state,round(sum(oi.price),2) as sum_price,round(avg(oi.price),2) as average_price from `Target_SQL.order_items` oi join `Target_SQL.orders` o on oi.order_id = o.order_id join `Target_SQL.customers` c on c.customer_id = o.customer_id group by 1 ;



```
select c.customer_state,round(sum(oi.price),2) as sum_price,round(avg(oi.price),2) as average_price
from `Target_SQL.order_items` oi join `Target_SQL.orders` o on oi.order_id = o.order_id
join `Target_SQL.customers` c on c.customer_id = o.customer_id group by 1 order by 1;
```

Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS CHART PREVIEW EXECUTION GRAPH

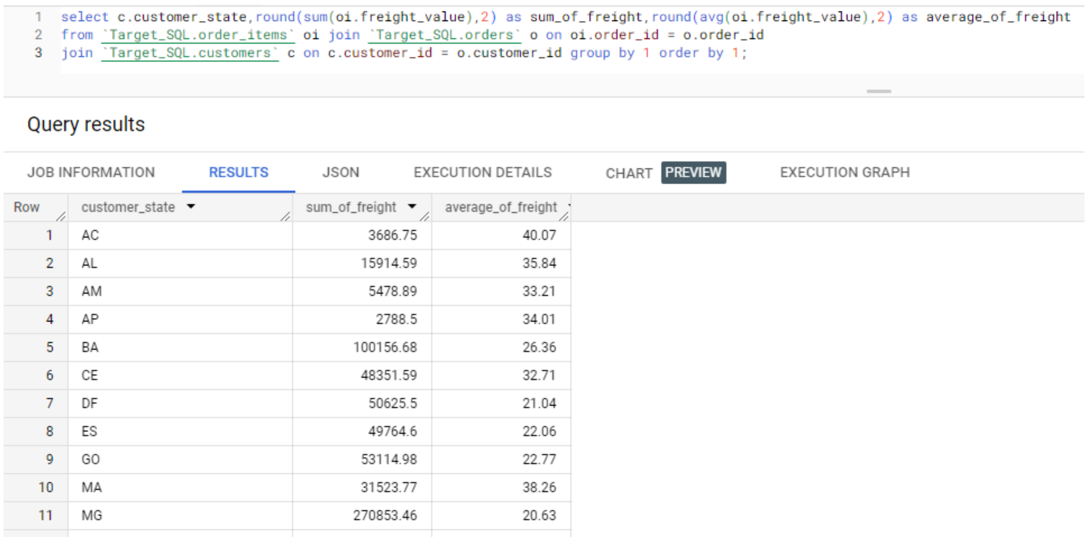
Row	customer_state	sum_price	average_price
1	AC	15982.95	173.73
2	AL	80314.81	180.89
3	AM	22356.84	135.5
4	AP	13474.3	164.32
5	BA	511349.99	134.6
6	CE	227254.71	153.76
7	DF	302603.94	125.77
8	ES	275037.31	121.91
9	GO	294591.95	126.27
10	MA	119648.22	145.2
11	MG	1585308.03	120.75
12	MS	116812.64	142.63
13	MT	156453.53	148.3
14	PA	178947.81	165.69
15	PB	115268.08	191.48
16	PE	262788.03	145.51

Inference: Considering the total value of order price in various states, it is observed that the state of SP stands out with the highest value of order price of 5202955.05, followed by MG at 1585308.03. Even though, SP generates the highest revenue, it has one of the lower average order values ,i.e, 109.65.

This might be due to a higher number of orders but with relatively lower price values. On the other hand, states like PB and RN have higher average order values of 191.48 and 156.97, respectively.

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

Query: select c.customer_state,round(sum(oi.freight_value),2) as sum_of_freight, round(avg(oi.freight_value),2) as average_of_freight from `Target_SQL.order_items` oi join `Target_SQL.orders` o on oi.order_id = o.order_id join `Target_SQL.customers` c on c.customer_id = o.customer_id group by 1 order by 1;



```
select c.customer_state,round(sum(oi.freight_value),2) as sum_of_freight,round(avg(oi.freight_value),2) as average_of_freight
from `Target_SQL.order_items` oi join `Target_SQL.orders` o on oi.order_id = o.order_id
join `Target_SQL.customers` c on c.customer_id = o.customer_id group by 1 order by 1;
```

Query results

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

Row	customer_state	sum_of_freight	average_of_freight
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
11	MG	270853.46	20.63

Inference: States like RS, PR, and BA have relatively higher sum of freight costs, whereas states like AM, AC, and AP have lower freight costs. Coming on to average freight costs, states like PE, CE, and PA have higher average freight costs, whereas SP, PR and MG have lower freight costs. Observing the situation, it becomes evident that PR has a higher sum of freight costs, accompanied by a notably low average freight cost. This pattern indicates the potential for a higher number of freights with reduced freight costs.

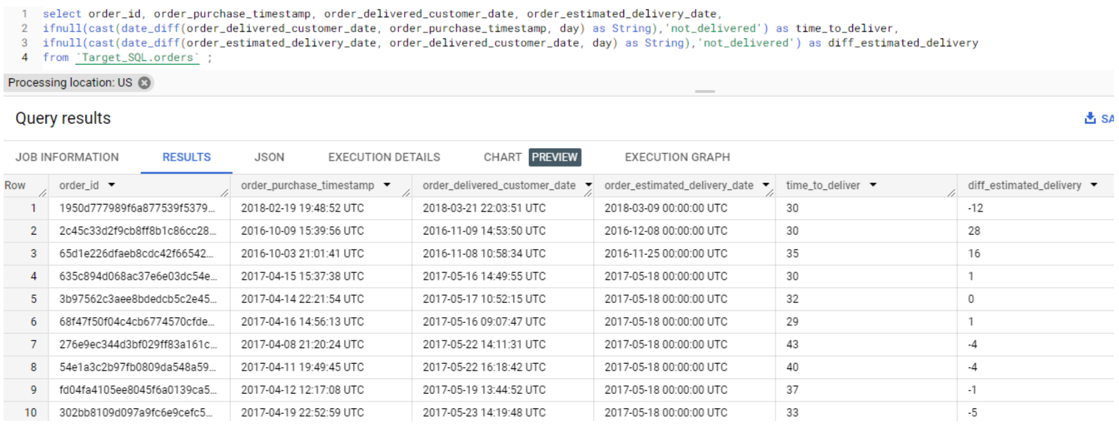
5. **Analysis based on sales, freight and delivery time:**

Ques 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**

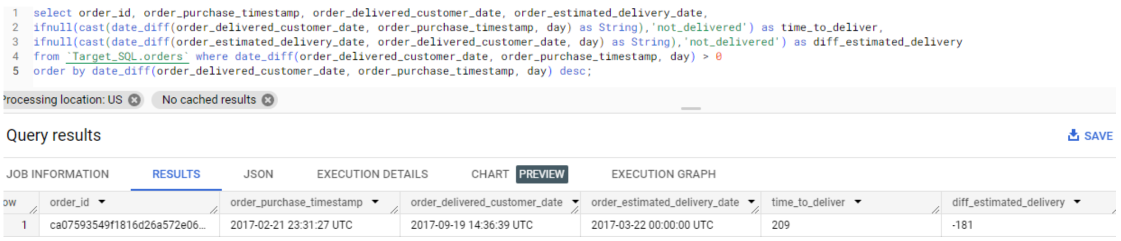
Query: select order_id, order_purchase_timestamp, order_delivered_customer_date, order_estimated_delivery_date,
ifnull(cast(date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as String),'not_delivered') as time_to_deliver,
ifnull(cast(date_diff(order_estimated_delivery_date, order_delivered_customer_date, day) as String),'not_delivered') as diff_estimated_delivery from `Target_SQL.orders` ;



```sql
select order_id, order_purchase_timestamp, order_delivered_customer_date, order_estimated_delivery_date,
ifnull(cast(date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as String),'not_delivered') as time_to_deliver,
ifnull(cast(date_diff(order_estimated_delivery_date, order_delivered_customer_date, day) as String),'not_delivered') as diff_estimated_delivery
from `Target_SQL.orders` ;
```

Row	order_id	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	time_to_deliver	diff_estimated_delivery
1	1950d777989f6a877539f5379...	2018-02-19 19:48:52 UTC	2018-03-21 22:03:51 UTC	2018-03-09 00:00:00 UTC	30	-12
2	2c45c33d2f9cb8ff8b1c86cc28...	2016-10-09 15:39:56 UTC	2016-11-09 14:53:50 UTC	2016-12-08 00:00:00 UTC	30	28
3	65d1e226dfaeb8cdc42f66542...	2016-10-03 21:01:41 UTC	2016-11-08 10:58:34 UTC	2016-11-25 00:00:00 UTC	35	16
4	635c894d068ac37e6e03dc54e...	2017-04-15 15:37:38 UTC	2017-05-16 14:49:55 UTC	2017-05-18 00:00:00 UTC	30	1
5	3b97562c3aee8bdedcb5c2e45...	2017-04-14 22:21:54 UTC	2017-05-17 10:52:15 UTC	2017-05-18 00:00:00 UTC	32	0
6	68f47f50f04c4cb6774570cfde...	2017-04-16 14:56:13 UTC	2017-05-16 09:07:47 UTC	2017-05-18 00:00:00 UTC	29	1
7	276e9ec344d3bf029ff83a161c...	2017-04-08 21:20:24 UTC	2017-05-22 14:11:31 UTC	2017-05-18 00:00:00 UTC	43	-4
8	54e1a3c2b97fb0809da548a59...	2017-04-11 19:49:45 UTC	2017-05-22 16:18:42 UTC	2017-05-18 00:00:00 UTC	40	-4
9	fd04fa4105ee8045f6a0139ca5...	2017-04-12 12:17:08 UTC	2017-05-19 13:44:52 UTC	2017-05-18 00:00:00 UTC	37	-1
10	302bb8109d097a9fc6e9cefc5...	2017-04-19 22:52:59 UTC	2017-05-23 14:19:48 UTC	2017-05-18 00:00:00 UTC	33	-5

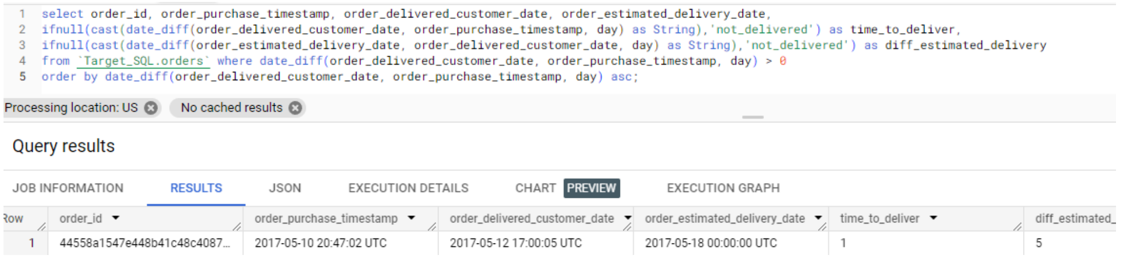
Max delivery time(209 days):



```sql
select order_id, order_purchase_timestamp, order_delivered_customer_date, order_estimated_delivery_date,
ifnull(cast(date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as String),'not_delivered') as time_to_deliver,
ifnull(cast(date_diff(order_estimated_delivery_date, order_delivered_customer_date, day) as String),'not_delivered') as diff_estimated_delivery
from `Target_SQL.orders` where date_diff(order_delivered_customer_date, order_purchase_timestamp, day) > 0
order by date_diff(order_delivered_customer_date, order_purchase_timestamp, day) desc;
```

ow	order_id	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	time_to_deliver	diff_estimated_delivery
1	ca07593549f1816d26a572e06...	2017-02-21 23:31:27 UTC	2017-09-19 14:36:39 UTC	2017-03-22 00:00:00 UTC	209	-181

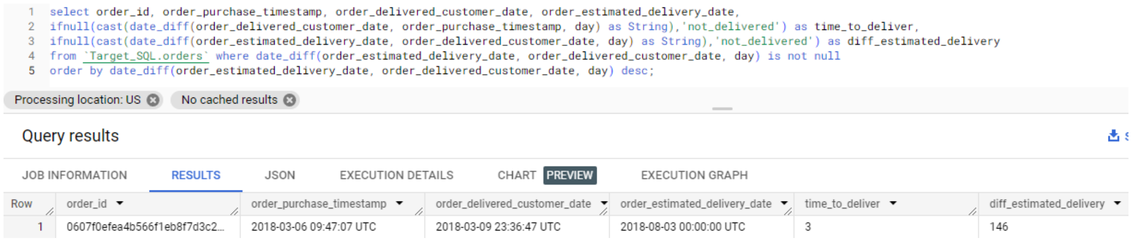
Min delivery time(1 day):



```sql
select order_id, order_purchase_timestamp, order_delivered_customer_date, order_estimated_delivery_date,
ifnull(cast(date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as String),'not_delivered') as time_to_deliver,
ifnull(cast(date_diff(order_estimated_delivery_date, order_delivered_customer_date, day) as String),'not_delivered') as diff_estimated_delivery
from `Target_SQL.orders` where date_diff(order_delivered_customer_date, order_purchase_timestamp, day) > 0
order by date_diff(order_delivered_customer_date, order_purchase_timestamp, day) asc;
```

Row	order_id	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	time_to_deliver	diff_estimated_
1	44558a1547e448b41c48c4087...	2017-05-10 20:47:02 UTC	2017-05-12 17:00:05 UTC	2017-05-18 00:00:00 UTC	1	5

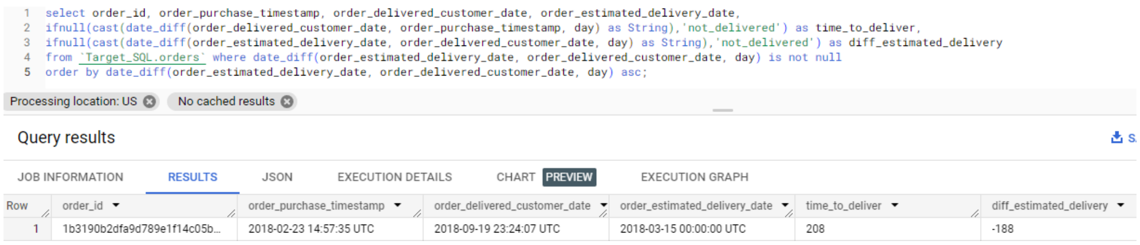
Maximum Delayed Delivery(146 days):



```sql
select order_id, order_purchase_timestamp, order_delivered_customer_date, order_estimated_delivery_date,
ifnull(cast(date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as String),'not_delivered') as time_to_deliver,
ifnull(cast(date_diff(order_estimated_delivery_date, order_delivered_customer_date, day) as String),'not_delivered') as diff_estimated_delivery
from `Target_SQL.orders` where date_diff(order_estimated_delivery_date, order_delivered_customer_date, day) is not null
order by date_diff(order_estimated_delivery_date, order_delivered_customer_date, day) desc;
```

Row	order_id	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	time_to_deliver	diff_estimated_delivery
1	0607f0efea4b566f1eb8f7d3c2...	2018-03-06 09:47:07 UTC	2018-03-09 23:36:47 UTC	2018-08-03 00:00:00 UTC	3	146

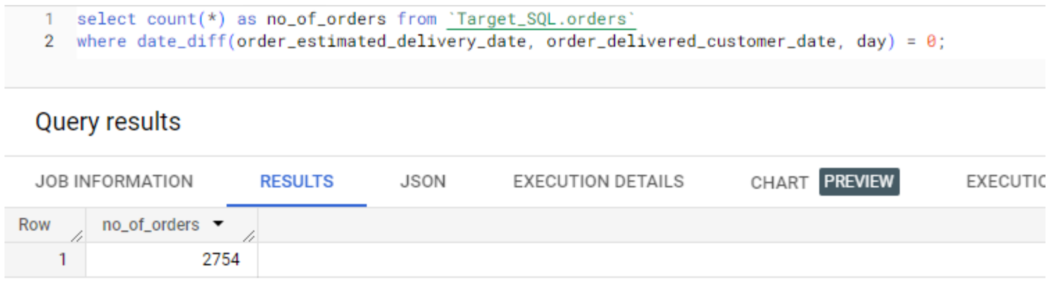
Maximum Early Delivery(188 days):



```sql
select order_id, order_purchase_timestamp, order_delivered_customer_date, order_estimated_delivery_date,
ifnull(cast(date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as String),'not_delivered') as time_to_deliver,
ifnull(cast(date_diff(order_estimated_delivery_date, order_delivered_customer_date, day) as String),'not_delivered') as diff_estimated_delivery
from `Target_SQL.orders` where date_diff(order_estimated_delivery_date, order_delivered_customer_date, day) is not null
order by date_diff(order_estimated_delivery_date, order_delivered_customer_date, day) asc;
```

Row	order_id	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	time_to_deliver	diff_estimated_delivery
1	1b3190b2dfa9d789e1f14c05b...	2018-02-23 14:57:35 UTC	2018-09-19 23:24:07 UTC	2018-03-15 00:00:00 UTC	208	-188

On-time delivery:

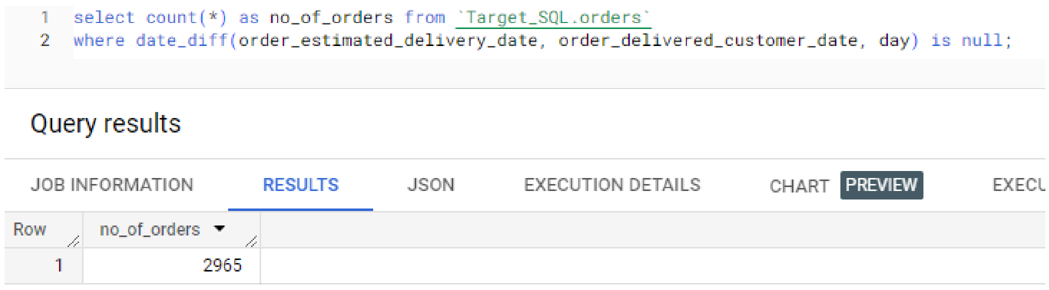


```sql
select count(*) as no_of_orders from `Target_SQL.orders`
where date_diff(order_estimated_delivery_date, order_delivered_customer_date, day) = 0;
```

Query results

Row	no_of_orders
1	2754

Not delivered orders:

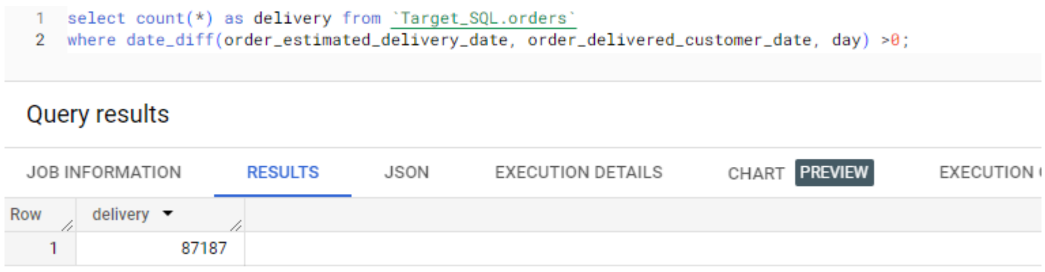


```sql
select count(*) as no_of_orders from `Target_SQL.orders`
where date_diff(order_estimated_delivery_date, order_delivered_customer_date, day) is null;
```

Query results

Row	no_of_orders
1	2965

No. of Orders delivered after estimated delivery dates:

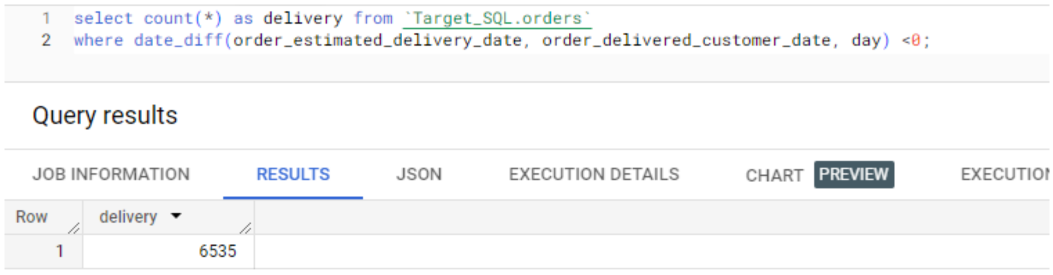


```sql
select count(*) as delivery from `Target_SQL.orders`
where date_diff(order_estimated_delivery_date, order_delivered_customer_date, day) >0;
```

Query results

Row	delivery
1	87187

No. of Orders delivered before estimated delivery dates:



```sql
select count(*) as delivery from `Target_SQL.orders`
where date_diff(order_estimated_delivery_date, order_delivered_customer_date, day) <0;
```

Query results

Row	delivery
1	6535

Inference: Out of a total of 99441 orders, some interesting patterns emerged in how they were delivered.
Unfortunately, 2965 orders didn't get delivered at all. On a positive note, 2754 orders reached their destination right on time, which is great. However, a larger group of 87187 orders faced delays and arrived later than expected. On the bright side again, 6535 orders got to their recipients earlier than predicted. Looking at the time it took, the longest delivery journey

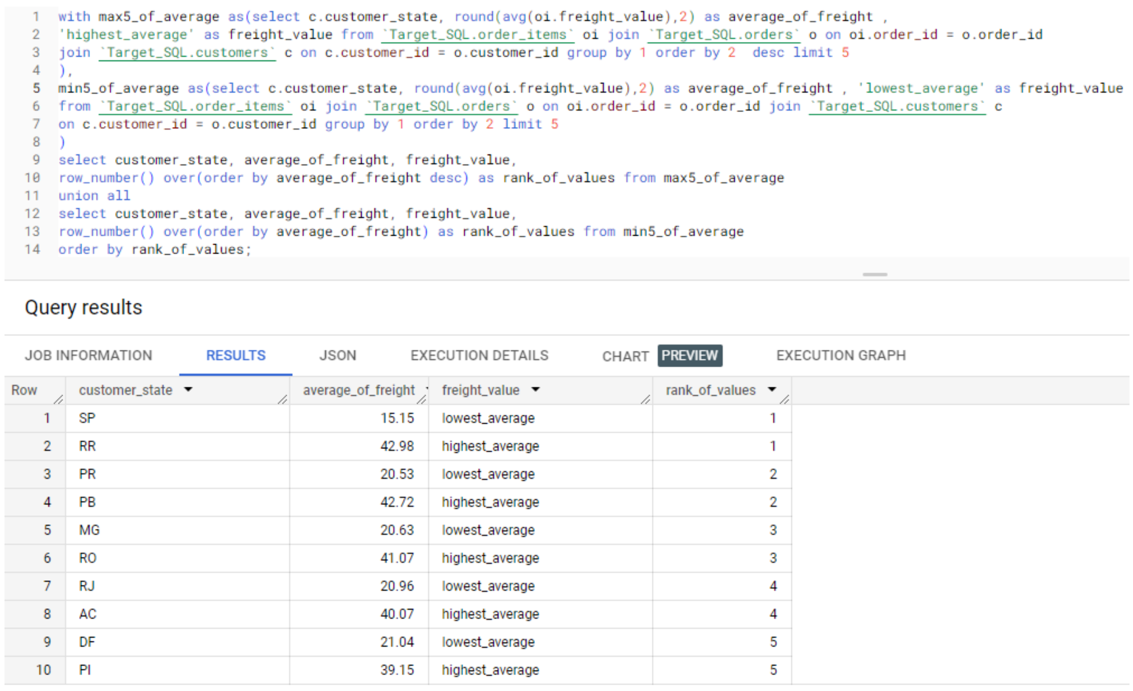
spanned 209 days, while the quickest was just 1 day. The most significant delay was 146 days, but surprisingly, there was a case where an order arrived 188 days ahead of schedule. This mix of delivery outcomes and timing paints an interesting picture of how orders were handled.

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

Query: with max5_of_average as(select c.customer_state, round(avg(oi.freight_value),2) as average_of_freight , 'highest_average' as freight_value from `Target_SQL.order_items` oi join `Target_SQL.orders` o on oi.order_id = o.order_id join `Target_SQL.customers` c on c.customer_id = o.customer_id group by 1 order by 2 desc limit 5),

min5_of_average as(
select c.customer_state, round(avg(oi.freight_value),2) as average_of_freight ,
'lowest_average' as freight_value from `Target_SQL.order_items` oi join `Target_SQL.orders` o on oi.order_id = o.order_id join `Target_SQL.customers` c on c.customer_id = o.customer_id group by 1 order by 2 limit 5)

select customer_state, average_of_freight, freight_value,
row_number() over(order by average_of_freight desc) as rank_of_values from max5_of_average
union all
select customer_state, average_of_freight, freight_value,
row_number() over(order by average_of_freight) as rank_of_values from min5_of_average
order by rank_of_values;



```
with max5_of_average as(select c.customer_state, round(avg(oi.freight_value),2) as average_of_freight ,
'highest_average' as freight_value from `Target_SQL.order_items` oi join `Target_SQL.orders` o on oi.order_id = o.order_id
join `Target_SQL.customers` c on c.customer_id = o.customer_id group by 1 order by 2  desc limit 5
),
min5_of_average as(select c.customer_state, round(avg(oi.freight_value),2) as average_of_freight , 'lowest_average' as freight_value
from `Target_SQL.order_items` oi join `Target_SQL.orders` o on oi.order_id = o.order_id join `Target_SQL.customers` c
on c.customer_id = o.customer_id group by 1 order by 2 limit 5
)
select customer_state, average_of_freight, freight_value,
row_number() over(order by average_of_freight desc) as rank_of_values from max5_of_average
union all
select customer_state, average_of_freight, freight_value,
row_number() over(order by average_of_freight) as rank_of_values from min5_of_average
order by rank_of_values;
```

Query results

Row	customer_state	average_of_freight	freight_value	rank_of_values
1	SP	15.15	lowest_average	1
2	RR	42.98	highest_average	1
3	PR	20.53	lowest_average	2
4	PB	42.72	highest_average	2
5	MG	20.63	lowest_average	3
6	RO	41.07	highest_average	3
7	RJ	20.96	lowest_average	4
8	AC	40.07	highest_average	4
9	DF	21.04	lowest_average	5
10	PI	39.15	highest_average	5

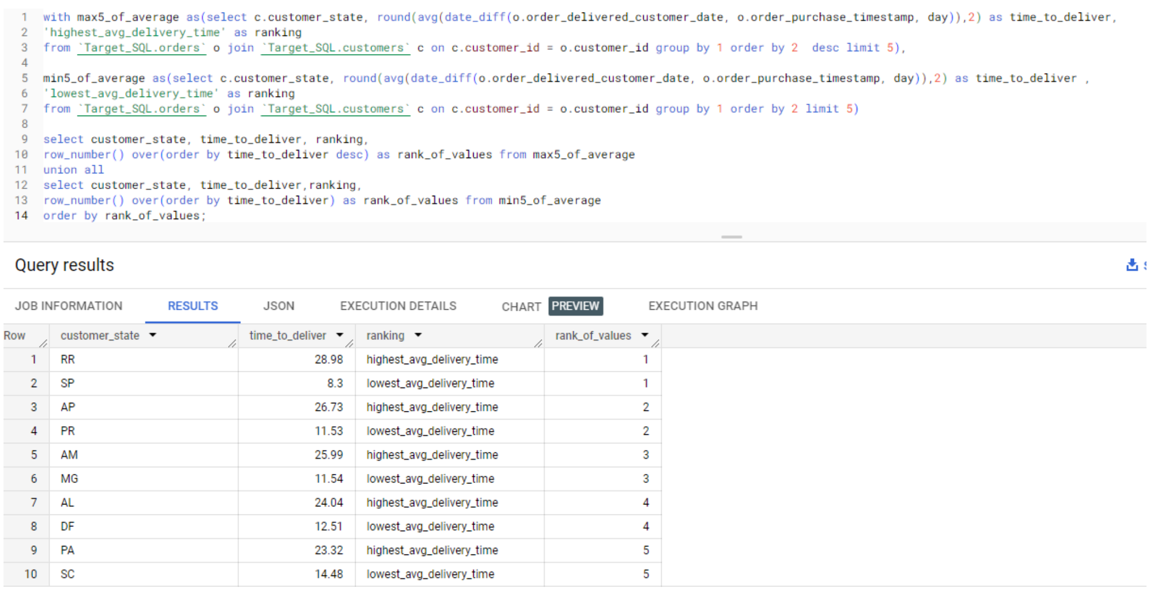
Inference: Among the customer states, the top five with the highest average freight values are RR leading with 42.98, followed by PB at 42.72, RO at 41.07, AC at 40.07, PI at 39.15. On the other end, the states with the lowest average freight values are SP with 15.15, PR with 20.53, MG with 20.63, RJ with 20.96, and DF with 21.04. This distribution highlights the considerable differences in average freight costs among the states and offers information that could be useful for making decisions about shipping and logistics.

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

Query: with max5_of_average as(select c.customer_state, round(avg(date_diff(o.order_delivered_customer_date, o.order_purchase_timestamp, day)),2) as time_to_deliver,'highest_avg_delivery_time' as ranking from `Target_SQL.orders` o join `Target_SQL.customers` c on c.customer_id = o.customer_id group by 1 order by 2 desc limit 5),

min5_of_average as(select c.customer_state,
round(avg(date_diff(o.order_delivered_customer_date, o.order_purchase_timestamp, day)),2)
as time_to_deliver ,'lowest_avg_delivery_time' as ranking
from `Target_SQL.orders` o join `Target_SQL.customers` c on c.customer_id = o.customer_id
group by 1 order by 2 limit 5)

select customer_state, time_to_deliver, ranking,
row_number() over(order by time_to_deliver desc) as rank_of_values from max5_of_average
union all
select customer_state, time_to_deliver,ranking,
row_number() over(order by time_to_deliver) as rank_of_values from min5_of_average
order by rank_of_values;



```
with max5_of_average as(select c.customer_state, round(avg(date_diff(o.order_delivered_customer_date, o.order_purchase_timestamp, day)),2) as time_to_deliver,
'highest_avg_delivery_time' as ranking
from `Target_SQL.orders` o join `Target_SQL.customers` c on c.customer_id = o.customer_id group by 1 order by 2  desc limit 5),

min5_of_average as(select c.customer_state, round(avg(date_diff(o.order_delivered_customer_date, o.order_purchase_timestamp, day)),2) as time_to_deliver ,
'lowest_avg_delivery_time' as ranking
from `Target_SQL.orders` o join `Target_SQL.customers` c on c.customer_id = o.customer_id group by 1 order by 2 limit 5)

select customer_state, time_to_deliver, ranking,
row_number() over(order by time_to_deliver desc) as rank_of_values from max5_of_average
union all
select customer_state, time_to_deliver,ranking,
row_number() over(order by time_to_deliver) as rank_of_values from min5_of_average
order by rank_of_values;
```

Query results

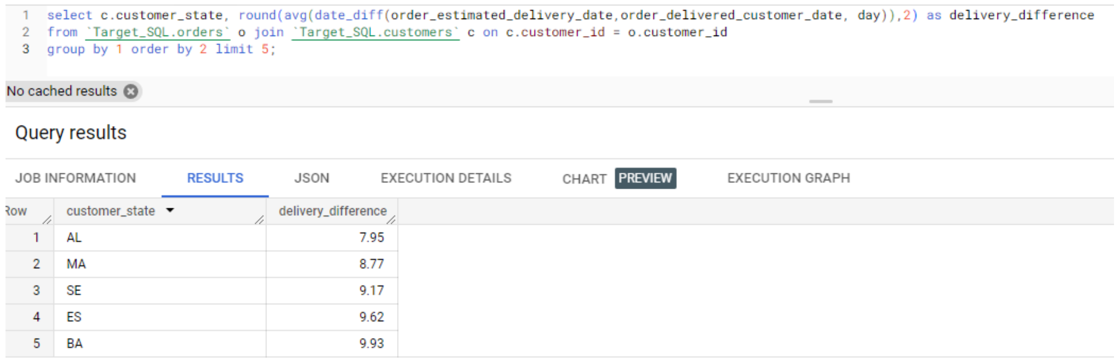
JOB INFORMATION RESULTS JSON EXECUTION DETAILS CHART PREVIEW EXECUTION GRAPH

Row	customer_state	time_to_deliver	ranking	rank_of_values
1	RR	28.98	highest_avg_delivery_time	1
2	SP	8.3	lowest_avg_delivery_time	1
3	AP	26.73	highest_avg_delivery_time	2
4	PR	11.53	lowest_avg_delivery_time	2
5	AM	25.99	highest_avg_delivery_time	3
6	MG	11.54	lowest_avg_delivery_time	3
7	AL	24.04	highest_avg_delivery_time	4
8	DF	12.51	lowest_avg_delivery_time	4
9	PA	23.32	highest_avg_delivery_time	5
10	SC	14.48	lowest_avg_delivery_time	5

Inference: In terms of the highest average delivery times, RR holds the top rank with an average delivery time of 28.98, while AP follows with a rank of 2 and an average delivery time of 26.73. AM, AL and PA also have relatively high average delivery times, securing ranks 3,4 and 5 with an average delivery time of 25.99,24.04 and 23.32 respectively. On the other hand, for the lowest average delivery times, SP boasts the most efficient performance with an average delivery time of 8.3, earning it the top rank in this category. PR takes the second rank with an average delivery time of 11.53, closely followed by MG at rank 3 with a time of 11.54. DF and SC complete the list with ranks 4 and 5 respectively with an average delivery time of 12.51 and 14.48 respectively. It is evident that SP stands out for its notably low average delivery time, while RR exhibits the highest average delivery time among the states included in the dataset.

Ques 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.

Query: select c.customer_state, round(avg(date_diff(order_estimated_delivery_date,order_delivered_customer_date, day)),2) as delivery_difference from `Target_SQL.orders` o join `Target_SQL.customers` c on c.customer_id = o.customer_id group by 1 order by 2 limit 5;



```
select c.customer_state, round(avg(date_diff(order_estimated_delivery_date,order_delivered_customer_date, day)),2) as delivery_difference
from `Target_SQL.orders` o join `Target_SQL.customers` c on c.customer_id = o.customer_id
group by 1 order by 2 limit 5;
```

No cached results

Query results

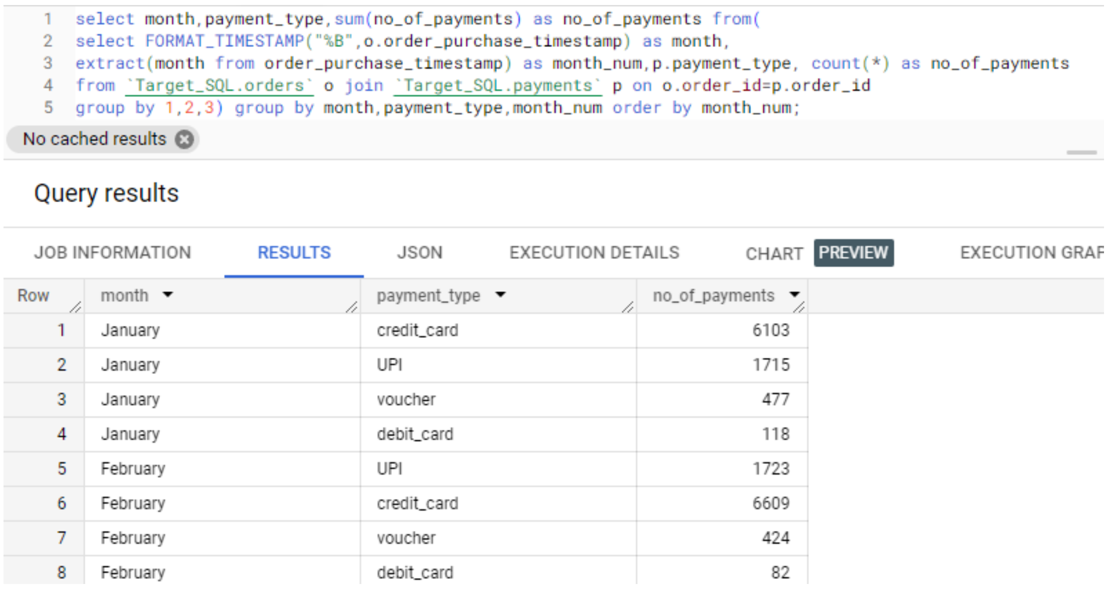
Row	customer_state	delivery_difference
1	AL	7.95
2	MA	8.77
3	SE	9.17
4	ES	9.62
5	BA	9.93

Inference: From the sorted results, it is apparent that the state of AL has the lowest average delivery time difference of 7.95 days. Following AL, the states of MA, SE, ES, and BA also exhibit relatively fast delivery speeds compared to the estimated dates, with average delivery time differences of 8.77, 9.17, 9.62, and 9.93 days respectively.The data suggests that customers residing in these states tend to experience faster order delivery times when compared to customers in other states.

6. **Analysis based on the payments:**

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

Query: select month,payment_type,sum(no_of_payments) as no_of_payments from(select FORMAT_TIMESTAMP("%B",o.order_purchase_timestamp) as month, extract(month from order_purchase_timestamp) as month_num,p.payment_type, count(*) as no_of_payments from `Target_SQL.orders` o join `Target_SQL.payments` p on o.order_id=p.order_id group by 1,2,3) group by month,payment_type,month_num order by month_num;



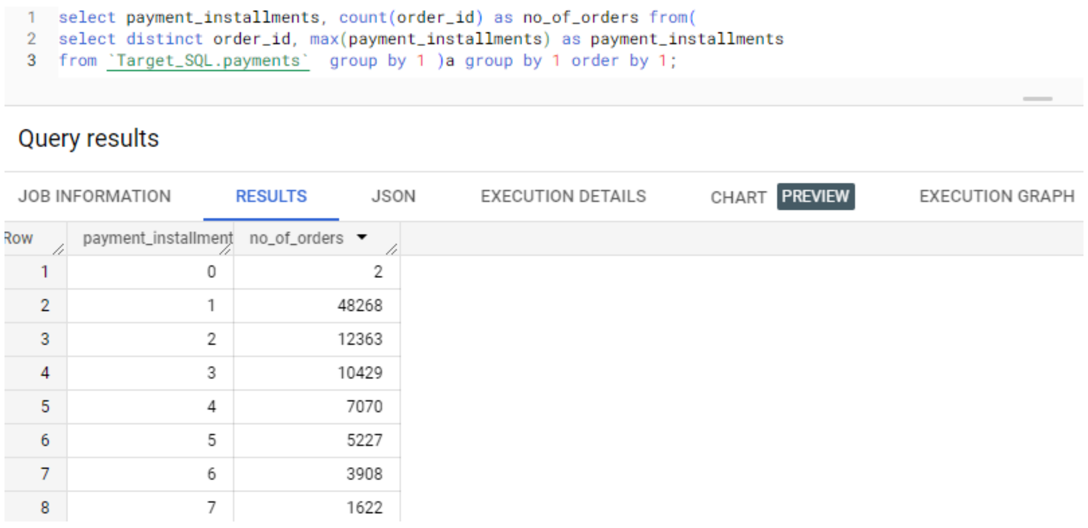
```
select month,payment_type,sum(no_of_payments) as no_of_payments from(
select FORMAT_TIMESTAMP("%B",o.order_purchase_timestamp) as month,
extract(month from order_purchase_timestamp) as month_num,p.payment_type, count(*) as no_of_payments
from `Target_SQL.orders` o join `Target_SQL.payments` p on o.order_id=p.order_id
group by 1,2,3) group by month,payment_type,month_num order by month_num;
```

Row	month	payment_type	no_of_payments
1	January	credit_card	6103
2	January	UPI	1715
3	January	voucher	477
4	January	debit_card	118
5	February	UPI	1723
6	February	credit_card	6609
7	February	voucher	424
8	February	debit_card	82

Inference: Credit cards and UPI payments consistently appear as the dominant payment methods throughout the entire year. Credit card usage remains relatively high, peaking in May with 8350 transactions.UPI transactions show a consistent pattern of usage, with notable peaks in May (2035) and August (2077). Voucher and debit card payments are considerably less frequent compared to credit cards and UPI. Credit card usage tends to peak in April, May, July, and December. UPI payments also show peaks in May, July, and December. Voucher payments appear more evenly distributed but remain relatively low in comparison to other methods. Debit card payments remain relatively consistent throughout the year, with no significant spikes. There are a few instances where payments are categorised as "Not Defined" in August and September. Further clarification is needed to understand the nature of these payments. Businesses may want to focus on credit card and UPI payment methods, as they consistently make up the majority of transactions.

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

Query: select payment_installments, count(order_id) as no_of_orders from(select distinct order_id, max(payment_installments) as payment_installments from `Target_SQL.payments` group by 1) a
group by 1 order by 1;



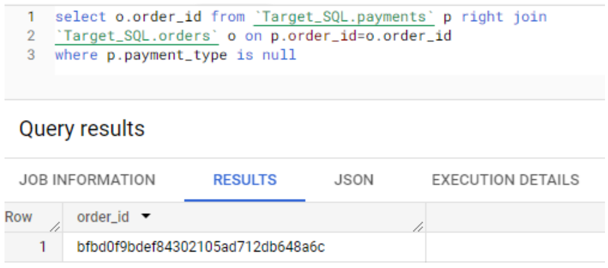
```sql
select payment_installments, count(order_id) as no_of_orders from(
select distinct order_id, max(payment_installments) as payment_installments
from `Target_SQL.payments`  group by 1 )a group by 1 order by 1;
```

Query results

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

Row	payment_installment	no_of_orders
1	0	2
2	1	48268
3	2	12363
4	3	10429
5	4	7070
6	5	5227
7	6	3908
8	7	1622

Inference: It is observed that the order_id ‘bfbd0f9bdef84302105ad712db648a6c‘ is not available in the payments table. So we are looking into only 99440 order_ids and not 99441 orders.



```sql
select o.order_id from `Target_SQL.payments` p right join
`Target_SQL.orders` o on p.order_id=o.order_id
where p.payment_type is null
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	order_id
1	bfbd0f9bdef84302105ad712db648a6c

Also, there are 2 orders with payment_installement as 0. Keeping that aside, nearly half of the orders are paid in a single installment, indicating a preference among a significant portion of customers for immediate full payments. As the number of payment installments increases, there is a gradual decline in the number of orders, highlighting that customers are more inclined to choose shorter-term payment plans. Clusters of installments at 2, 3, 4, and 5 suggest that some customers prefer to split payments into a smaller number of installments. The number of orders decreases significantly after the 5th installment, with a gradual decline in orders as the number of installments increases. Promoting single-installment payments could capture a larger customer base and enhance overall customer satisfaction is what we could deduce from the data.