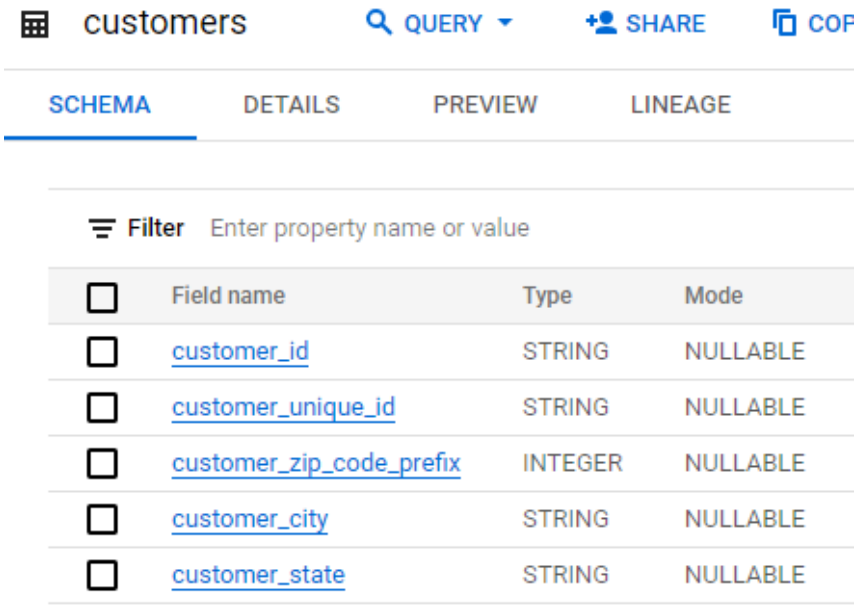
Solution by Savio Dias

Problem Statement:

Assuming you are a data analyst/ scientist at Target, you have been assigned the task of analyzing the given dataset to extract valuable insights and provide actionable recommendations.

What does 'good' look like?

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.

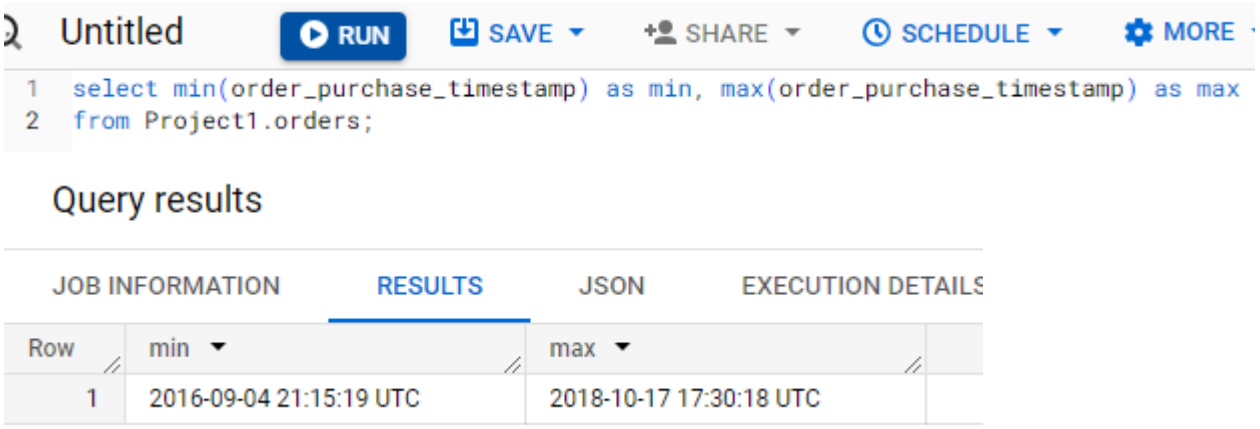


The customers table consist of the data types mentioned above.

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

Query:

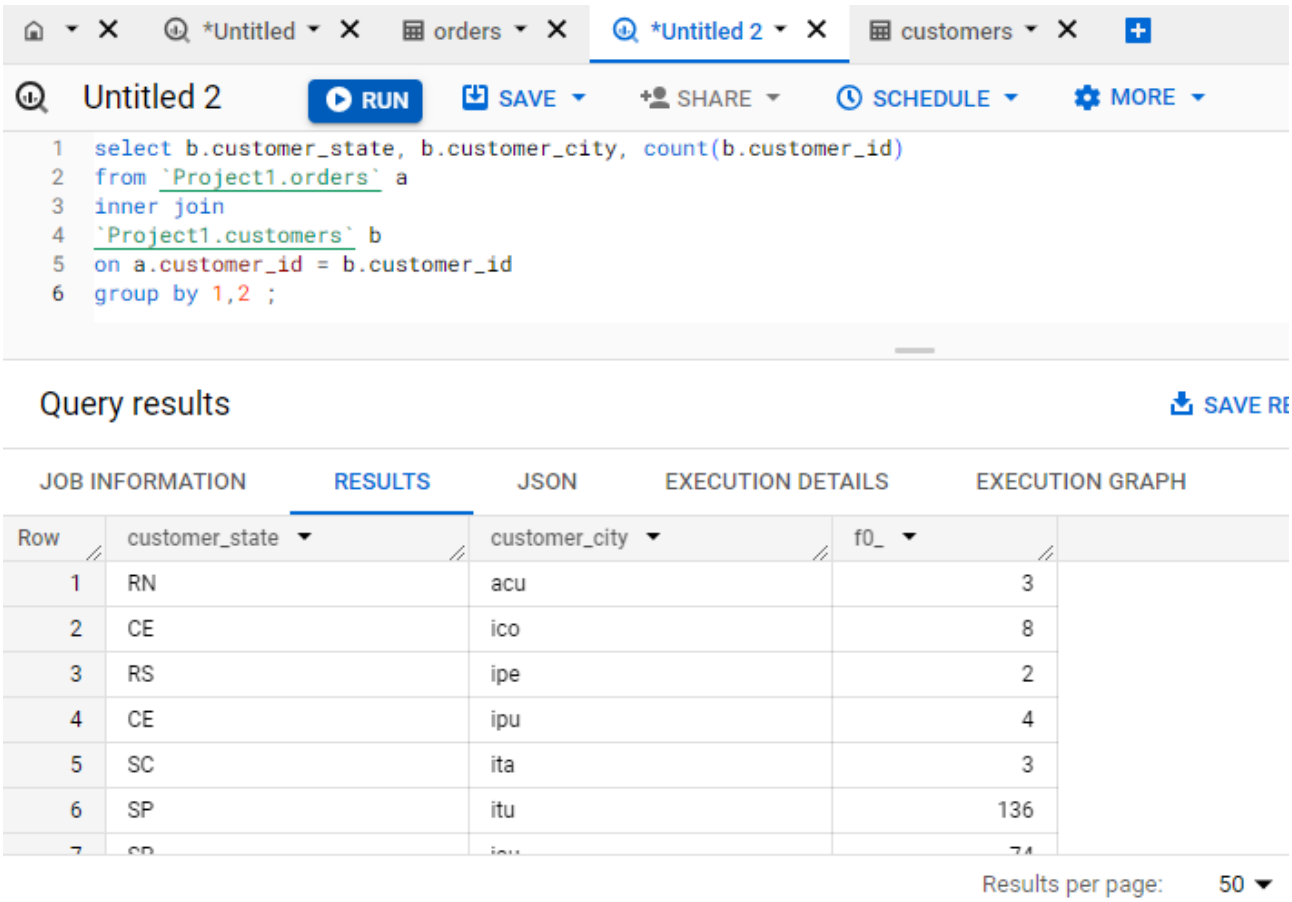
```
select min(order_purchase_timestamp) as min, max(order_purchase_timestamp) as max
from Project1.orders;
```



As observed by running the above query to get the date range between the orders placed, I found that the date-range is not starting from 1st January 2016 i.e. not the complete year as the 1st order was placed on 4th September 2016 and last order was placed on 17th October 2018

3. Count the number of Cities and States in our dataset.

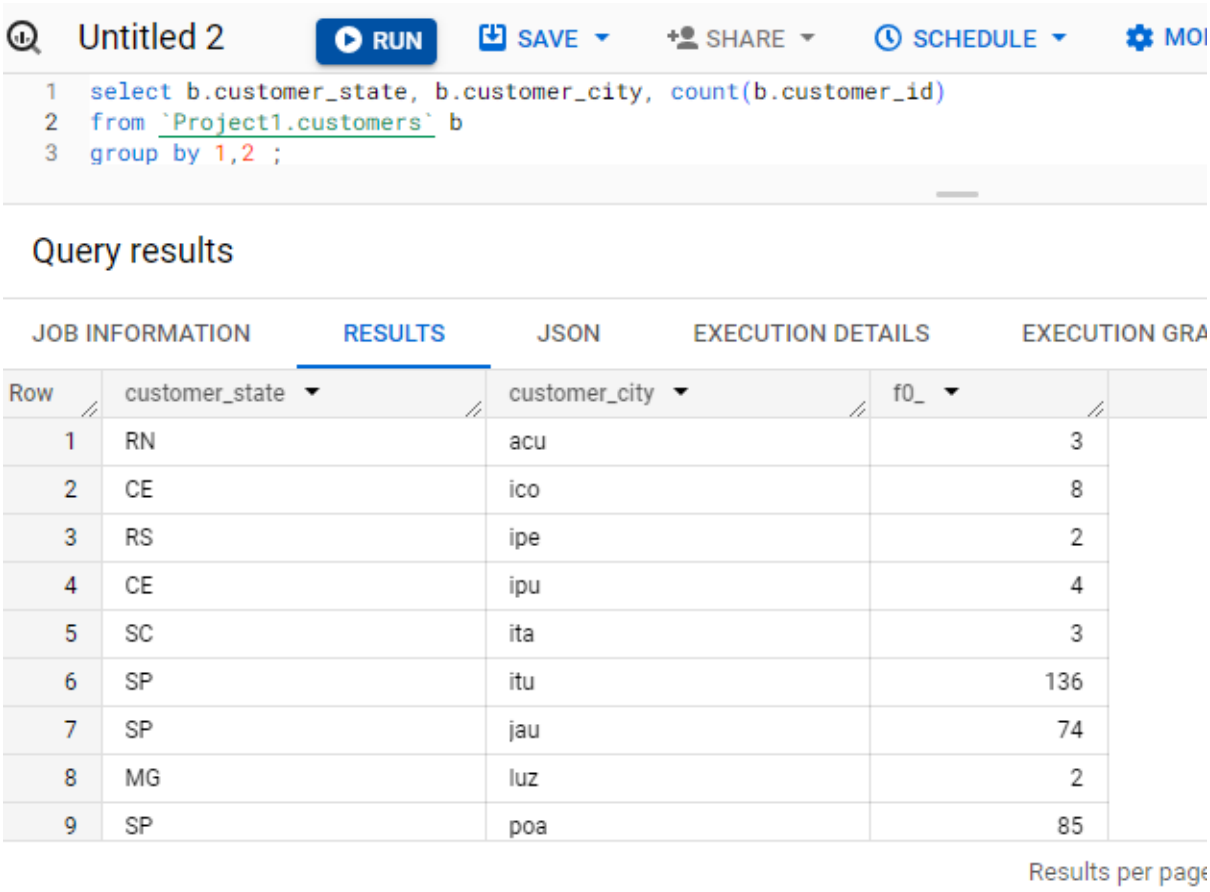
```
select b.customer_state, b.customer_city, count(b.customer_id)
from `Project1.orders` a
inner join
`Project1.customers` b
on a.customer_id = b.customer_id
group by 1,2;
```



The count can be obtained by using 2 tables- orders and customers by using inner join between them, so as to get only the customers ordered during the given period.

However, to get all the count in the dataset with no time-period, we can use only the customers table as shown below,

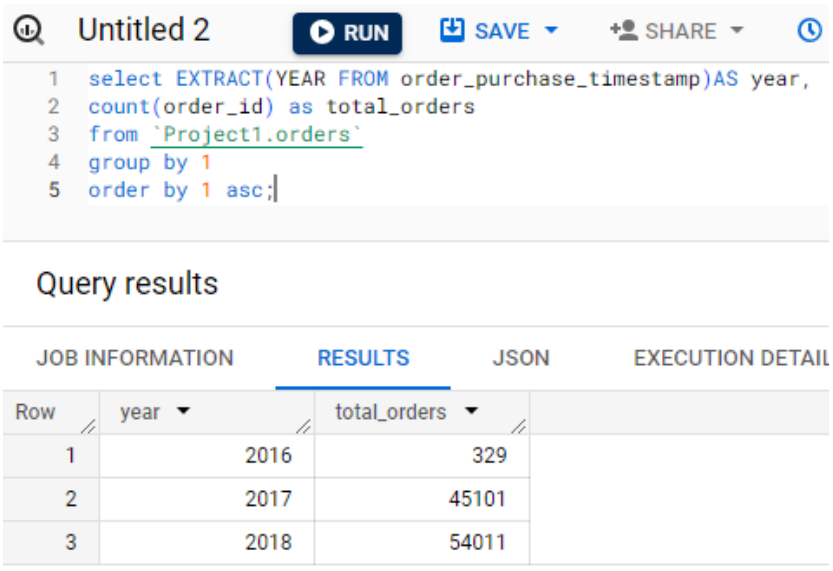
```
select b.customer_state, b.customer_city, count(b.customer_id)
from `Project1.customers` b
group by 1,2;
```



2. **In-depth Exploration:**

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

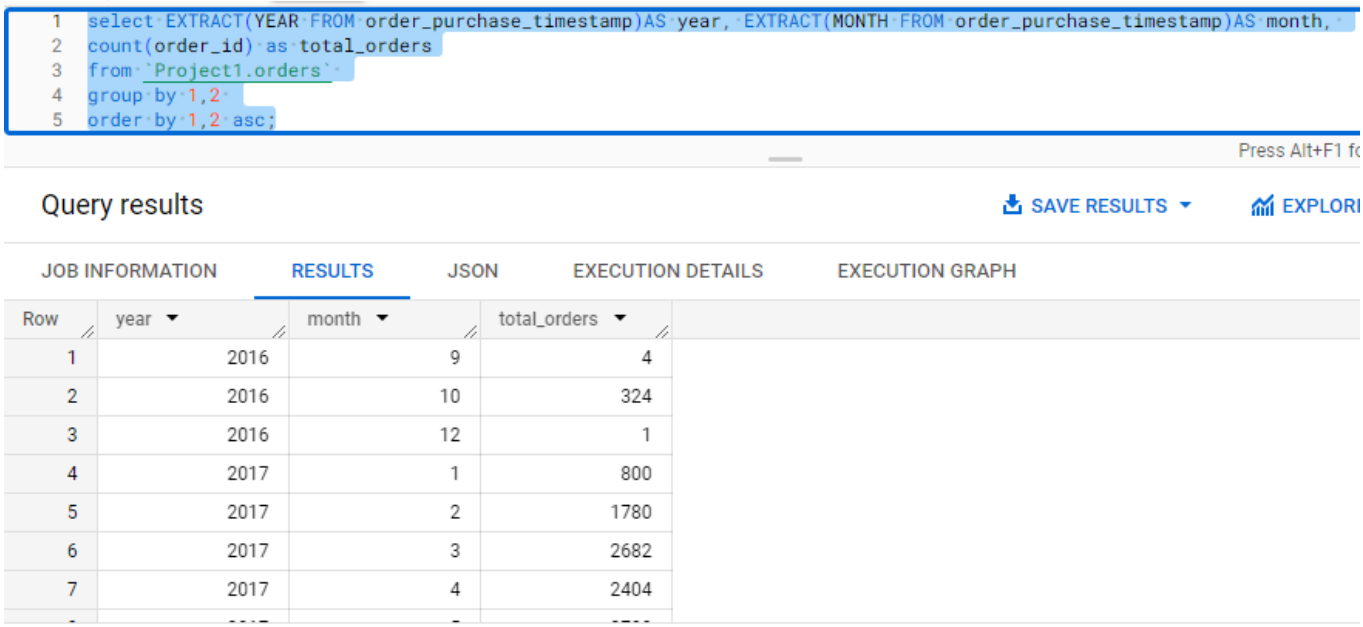
```
select EXTRACT(YEAR FROM order_purchase_timestamp)AS year,
count(order_id) as total_orders
from `Project1.orders`
group by 1
order by 1 asc;
```



Yes, there's a growing trend i.e. an increase in orders when grouping and checking them by year starting from 2016 to 2018 as shown above in the results

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

```
select EXTRACT(YEAR FROM order_purchase_timestamp)AS year, EXTRACT(MONTH FROM
order_purchase_timestamp)AS month,
count(order_id) as total_orders
from `Project1.orders`
group by 1,2
order by 1,2 asc;
```



Query results

SAVE RESULTS EXPLORI

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

Row	year	month	total_orders
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800
5	2017	2	1780
6	2017	3	2682
7	2017	4	2404

Yes, there's a monthly seasonality in terms of the no. of orders being placed there's a trend by month as shown below. The count of orders generally increases from March to August with fluctuations in between. Notably, there is an increase in orders during February and March. Additionally, the month of August shows a peak in order count.

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

```
select x.Time_of_Day, count(distinct x.order_id) as Total_Orders from
(select *,
CASE
WHEN EXTRACT(HOUR FROM order_purchase_timestamp) between 00 and 06  THEN 'Dawn'
WHEN EXTRACT(HOUR FROM order_purchase_timestamp) between 07 and 12  THEN 'Morning'
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_Day
from `Project1.orders`)x
group by x.Time_of_Day
order by count(distinct x.order_id) desc;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXE

Row	Time_of_Day	Total_Orders
1	Afternoon	38135
2	Night	28331
3	Morning	27733
4	Dawn	5242

From above as obtained the results based on counting the total orders and grouping them by time of day, we can conclude that the Brazilian customers mostly place their orders during **Afternoon** (13-18 hrs).

3. **Evolution of E-commerce orders in the Brazil region:**
    1. Get the month on month no. of orders placed in each state.

```
SELECT  b.customer_state,
  EXTRACT(month FROM a.order_purchase_timestamp) AS month,
  COUNT(a.order_purchase_timestamp) AS order_count
FROM `Project1.orders` a
JOIN `Project1.customers` b
ON a.customer_id = b.customer_id
GROUP BY b.customer_state, month
ORDER BY b.customer_state, month;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	month	order_count
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

Analyzed the month-on-month order counts for each state and it is evident that SP consistently has the highest number of orders in almost any given month, followed by RJ and MG.

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

```
select c.customer_state as state, count(c.customer_id) AS no_of_customers,
from `Project1.customers` c
group by 1
order by 2 desc;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXE

Row	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
8	DF	2140

The results from the SQL query gives that the state of SP has the highest number of customers, followed by RJ and then MG. SP can be said as the most populous state in Brazil

4. **Impact on Economy: Analyze 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
  EXTRACT(MONTH from a.order_purchase_timestamp) AS month,
  (( sum(CASE WHEN EXTRACT(YEAR from a.order_purchase_timestamp) = 2018 AND
      EXTRACT(MONTH from a.order_purchase_timestamp) BETWEEN 1 AND 8 THEN
      p.payment_value END) - sum(CASE WHEN EXTRACT(YEAR from a.order_purchase_timestamp) = 2017
AND
      EXTRACT(MONTH from a.order_purchase_timestamp) BETWEEN 1 AND 8 THEN p.payment_value END)
    )
    / sum(CASE WHEN EXTRACT(YEAR from a.order_purchase_timestamp) = 2017 AND
    EXTRACT(MONTH from a.order_purchase_timestamp) BETWEEN 1 AND 8 THEN p.payment_value END)
  )*100 AS Percentage_Increase
From `Project1.orders` a
JOIN `Project1.payments` p ON a.order_id = p.order_id
WHERE EXTRACT(YEAR from a.order_purchase_timestamp) IN (2017, 2018) AND
EXTRACT(MONTH from a.order_purchase_timestamp) BETWEEN 1 AND 8
group by 1
Order by 1;
```

Query results

JOB INFORMATION | RESULTS | JSON

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

To solve this, I calculated the percentage increase in the cost of orders from 2017 to 2018, only the months from January to August. Upon viewing the month-wise Percentage increase, January shows the highest percentage increase, followed by February and April.

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

```
SELECT c.customer_state,
ROUND(sum(b.price), 2) AS total_price,
ROUND(avg(b.price), 2) AS avg_price,
FROM `Project1.orders` a
INNER JOIN `Project1.order_items` b ON a.order_id = b.order_id
INNER JOIN `Project1.customers` c ON a.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY sum(b.freight_value) DESC;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	total_price	avg_price
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

While the state SP has the highest 'total price value' and, it surprisingly has the lowest 'average price' value among all states. On the other hand, the state – PB has the highest 'average price 'value.

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

```
SELECT c.customer_state,
sum(b.freight_value) AS total_freight_value,
ROUND(avg(b.freight_value), 2) AS avg_freight_value
FROM `Project1.orders` a
JOIN `Project1.order_items` b ON a.order_id = b.order_id
JOIN `Project1.customers` c ON a.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY sum(b.freight_value) DESC;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	total_freight_value	avg_freight_value
1	SP	718723.0699999...	15.15
2	RJ	305589.3100000...	20.96
3	MG	270853.4600000...	20.63
4	RS	135522.7400000...	21.74
5	PR	117851.6800000...	20.53
6	BA	100156.6799999...	26.36
7	SC	89660.26000000...	21.47

The analysis reveals the state SP has the highest 'total freight' value, it surprisingly has the lowest average 'freight value' among all states. On the other hand, the state - RR has the highest 'average freight value'.

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,
DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) AS time_to_deliver,
--(order_delivered_customer_date - order_purchase_timestamp) AS time_to_deliver,
DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date, DAY) AS
diff_estimated_delivery
--(order_estimated_delivery_date - order_delivered_customer_date) AS diff_estimated_delivery
FROM `Project1.orders`
WHERE DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) IS NOT NULL
ORDER BY time_to_deliver;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXE

Row	order_id	time_to_deliver	diff_estimated_delivery
13	21a8ffca665bc7a1087d31751...	0	11
14	44558a1547e448b41c48c4087...	1	5
15	3bfd703ce884b8a0a65e63f2d...	1	5
16	68fa625f02107978e969340da...	1	5
17	0d8f485ffe96c81fe3e282095e...	1	11
18	0922ee1619de7b995648e5a84...	1	32
19	1a0f86a669f41850ec641339c...	1	25
20	04e452498b75f049f617f5ed7f...	1	12

To understand the time duration between purchasing an order, its delivery, and the estimated delivery, we calculated the number of days using the given formulas:-

- **time_to_deliver** = order_delivered_customer_date - order_purchase_timestamp
- **diff_estimated_delivery** = order_estimated_delivery_date - order_delivered_customer_date

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

```
SELECT c.customer_state as state,
ROUND(avg(b.freight_value), 2) AS avg_freight_value
FROM `Project1.orders` a
JOIN `Project1.order_items` b ON a.order_id = b.order_id
JOIN `Project1.customers` c ON a.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY avg(b.freight_value) DESC
limit 5;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXE

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

The state - RR has the **highest** Average Freight Value with 42.98, followed by PB and then RO.

```sql
SELECT c.customer_state as state,
ROUND(avg(b.freight_value), 2) AS avg_freight_value
FROM `Project1.orders` a
JOIN `Project1.order_items` b ON a.order_id = b.order_id
JOIN `Project1.customers` c ON a.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY avg(b.freight_value) ASC
limit 5;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXE

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

The state - SP has the **Lowest** Average Freight Value with 15.15, followed by PR and then MG.

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

```sql
SELECT c.customer_state as state,
avg(DATE_DIFF(a.order_delivered_customer_date, a.order_purchase_timestamp, DAY)) AS
time_to_deliver
FROM `Project1.orders` a
JOIN `Project1.order_items` b ON a.order_id = b.order_id
JOIN `Project1.customers` c ON a.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY (time_to_deliver) desc
limit 5;
```

Query results

JOB INFORMATION | RESULTS | JSON | EX

Row	state	time_to_deliver
1	RR	27.82608695652...
2	AP	27.75308641975...
3	AM	25.96319018404...
4	AL	23.99297423887...
5	PA	23.30170777988...

The state - RR has the **highest** Average delivery time(in days), followed by AP and then AM. The top 5 states are displayed above.

```sql
SELECT c.customer_state as state,
avg(DATE_DIFF(a.order_delivered_customer_date, a.order_purchase_timestamp, DAY)) AS
time_to_deliver
FROM `Project1.orders` a
JOIN `Project1.order_items` b ON a.order_id = b.order_id
JOIN `Project1.customers` c ON a.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY (time_to_deliver) asc
limit 5;
```

Query results

JOB INFORMATION | RESULTS | JSON | EX

Row	state	time_to_deliver
1	SP	8.259608552419...
2	PR	11.48079306071...
3	MG	11.51552218007...
4	DF	12.50148619957...
5	SC	14.52098584675...

The top 5 states are displayed above. The state -SP has the **lowest** Average delivery time(in days), followed by PR and then MG.

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.

```sql
SELECT c.customer_state,
ROUND(AVG(DATE_DIFF(a.order_delivered_customer_date, a.order_purchase_timestamp, DAY)), 2) AS
avg_time_to_delivery,
ROUND(AVG(DATE_DIFF(a.order_estimated_delivery_date, a.order_delivered_customer_date, DAY)), 2)
AS avg_diff_estimated_delivery,
ROUND(AVG(DATE_DIFF(a.order_delivered_customer_date, a.order_purchase_timestamp, DAY)) -
AVG(DATE_DIFF(a.order_estimated_delivery_date, a.order_delivered_customer_date, DAY)), 2) AS
day_difference
FROM `Project1.orders` a
JOIN `Project1.customers` c ON a.customer_id = c.customer_id
WHERE DATE_DIFF(a.order_purchase_timestamp, a.order_delivered_customer_date, DAY) IS NOT NULL
AND DATE_DIFF(a.order_estimated_delivery_date, a.order_delivered_customer_date, DAY) IS NOT NULL
GROUP BY c.customer_state
ORDER BY day_difference
limit 5;
```

Query results

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

Row	customer_state	avg_time_to_delivery	avg_diff_estimated_c	day_difference
1	SP	8.3	10.14	-1.84
2	PR	11.53	12.36	-0.84
3	MG	11.54	12.3	-0.75
4	RO	18.91	19.13	-0.22
5	AC	20.64	19.76	0.87

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

```
SELECT p.payment_type,
EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
COUNT(DISTINCT o.order_id) AS order_count
FROM `Project1.orders` o
JOIN `Project1.payments` p
ON o.order_id = p.order_id
GROUP BY 1, 2
ORDER BY 1, 2;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	payment_type	month	order_count
1	UPI	1	1715
2	UPI	2	1723
3	UPI	3	1942
4	UPI	4	1783
5	UPI	5	2035
6	UPI	6	1807

I have observed that Credit card transactions are the most popular payment method, followed by UPI. Debit card transactions are the least preferred option.

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

```
SELECT p.payment_installments, COUNT(o.order_id) AS order_count
FROM `Project1.orders` o
JOIN `Project1.payments` p
```

```
ON o.order_id = p.order_id
WHERE o.order_status != 'canceled'
GROUP BY 1
ORDER BY 2 DESC;
```

Query results

JOB INFORMATION | RESULTS | JSON

Row	payment_installment	order_count
1	1	52184
2	2	12353
3	3	10392
4	4	7056
5	10	5292
6	5	5209
7	8	4239
8	6	3898

The above result reveals that most orders have only **one** payment installment. The highest number of installments is 24, which is associated with only 18 orders.