Business Case: Target SQL

NAME - ANSH JOSHI

Context:

Target is a globally renowned brand and a prominent retailer in the United States. Target makes itself a preferred shopping destination by offering outstanding value, inspiration, innovation and an exceptional guest experience that no other retailer can deliver.

This particular business case focuses on the operations of Target in Brazil and provides insightful information about 100,000 orders placed between 2016 and 2018. The dataset offers a comprehensive view of various dimensions including the order status, price, payment and freight performance, customer location, product attributes, and customer reviews.

By analyzing this extensive dataset, it becomes possible to gain valuable insights into Target's operations in Brazil. The information can shed light on various aspects of the business, such as order processing, pricing strategies, payment and shipping efficiency, customer demographics, product characteristics, and customer satisfaction levels.

Dataset: https://drive.google.com/drive/folders/1TGEc66YKbD443nslRi1bWgVd238gJCnb

The data is available in 8 csv files:

1. customers.csv
2. sellers.csv
3. order_items.csv
4. geolocation.csv
5. payments.csv
6. reviews.csv
7. orders.csv
8. products.csv

The column description for these csv files is given below.

The **customers.csv** contain following features:

Features	Description
customer_id	ID of the consumer who made the purchase
customer_unique_id	Unique ID of the consumer
customer_zip_code_prefix	Zip Code of consumer's location
customer_city	Name of the City from where order is made
customer_state	State Code from where order is made (Eg. são paulo - SP)

The **sellers.csv** contains following features:

Features	Description
seller_id	Unique ID of the seller registered
seller_zip_code_prefix	Zip Code of the seller's location
seller_city	Name of the City of the seller
seller_state	State Code (Eg. são paulo - SP)

The **order_items.csv** contain following features:

Features	Description
order_id	A Unique ID of order made by the consumers
order_item_id	A Unique ID given to each item ordered in the order
product_id	A Unique ID given to each product available on the site
seller_id	Unique ID of the seller registered in Target
shipping_limit_date	The date before which the ordered product must be shipped
price	Actual price of the products ordered
freight_value	Price rate at which a product is delivered from one point to another

The **geolocations.csv** contain following features:

Features	Description
geolocation_zip_code_prefix	First 5 digits of Zip Code
geolocation_lat	Latitude
geolocation_lng	Longitude
geolocation_city	City
geolocation_state	State

The **payments.csv** contain following features:

Features	Description
order_id	A Unique ID of order made by the consumers
payment_sequential	Sequences of the payments made in case of EMI
payment_type	Mode of payment used (Eg. Credit Card)
payment_installments	Number of installments in case of EMI purchase
payment_value	Total amount paid for the purchase order

The **orders.csv** contain following features:

Features	Description
order_id	A Unique ID of order made by the consumers
customer_id	ID of the consumer who made the purchase
order_status	Status of the order made i.e. delivered, shipped, etc.
order_purchase_timestamp	Timestamp of the purchase
order_delivered_carrier_date	Delivery date at which carrier made the delivery
order_delivered_customer_date	Date at which customer got the product
order_estimated_delivery_date	Estimated delivery date of the products

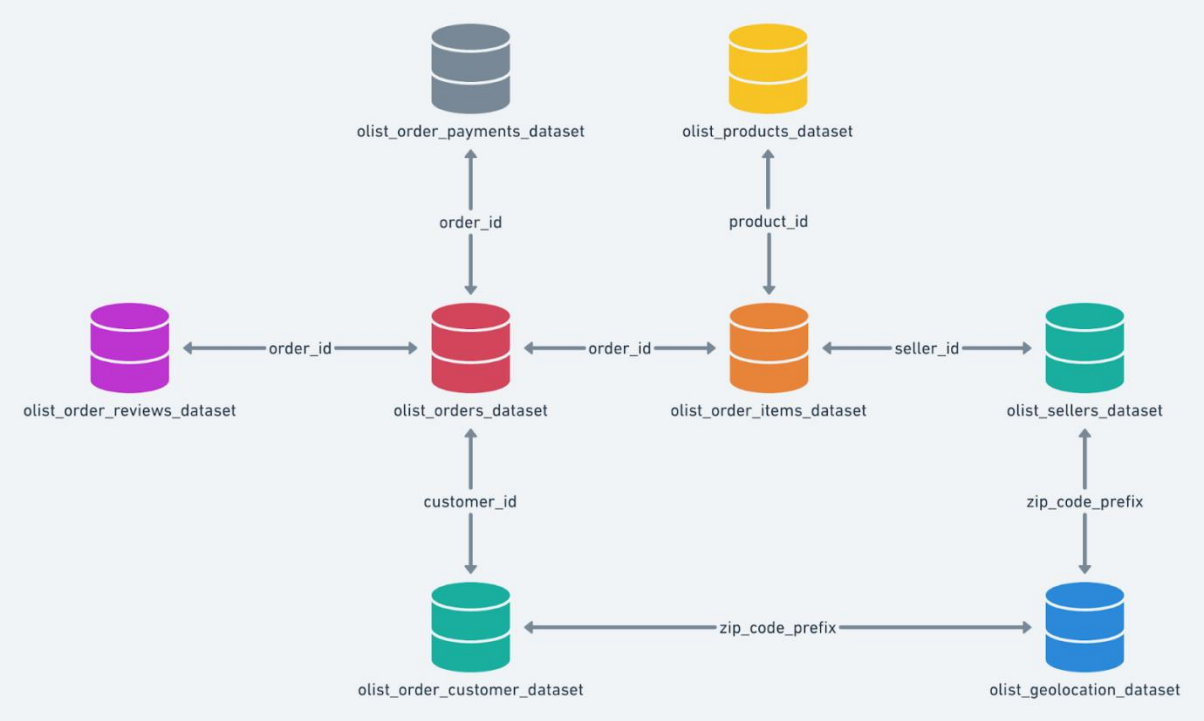
The **reviews.csv** contain following features:

Features	Description
review_id	ID of the review given on the product ordered by the order id
order_id	A Unique ID of order made by the consumers
review_score	Review score given by the customer for each order on a scale of 1-5
review_comment_title	Title of the review
review_comment_message	Review comments posted by the consumer for each order
review_creation_date	Timestamp of the review when it is created
review_answer_timestamp	Timestamp of the review answered

The **products.csv** contain following features:

Features	Description
product_id	A Unique identifier for the proposed project.
product_category_name	Name of the product category
product_name_lenght	Length of the string which specifies the name given to the products or
product_description_lenght	Length of the description written for each product ordered on the site
product_photos_qty	Number of photos of each product ordered available on the shopping
product_weight_g	Weight of the products ordered in grams
product_length_cm	Length of the products ordered in centimeters
product_height_cm	Height of the products ordered in centimeters
product_width_cm	Width of the product ordered in centimeters

Dataset schema:



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:

#1.1 - Data type of all columns in the "customers" table.
```

QUERY-

```
select * from BUSSINESS_CASE_STUDY.customers;
```

OUTPUT –

Row	customer_id	customer_unique_id	customer_zip_code	customer_city	customer_state
1	0735e7e4298a2ebbb4664934...	fcb003b1bdc0df64b4d065d9b...	59650	acu	RN
2	903b3d86e3990db01619a4eb...	46824822b15da44e983b021d...	59650	acu	RN
3	38c97666e962d4fea7fd6a83e...	b6108acc674ae5c99e29adc10...	59650	acu	RN
4	77c2f46cf580f4874c9a5751c2...	402cce5c0509000eed9e77fec...	63430	ico	CE
5	4d3ef4cfffb8ad4767c199c36a...	6ba00666ab7eada5ceec279b2...	63430	ico	CE
6	3000841b86e1fbe9493b52324...	796a0b1a21f597704057184a1...	63430	ico	CE
7	3c325415ccc7e622c66dec4bc...	05d1d2d9f0161c5f397ce7fc77...	63430	ico	CE
8	04f3a7b250e3be964f01bf22bc...	c34585a0276ecc5e4fb03de75...	63430	ico	CE
9	894202b8ef01f4719a4691e79...	01a4fe5fc00bbdb0b0a4af5a53...	63430	ico	CE
10	9d715b9fb75a9d081c14126c0...	8f399f3b7ace8e6245422c9e1f...	63430	ico	CE
11	018184ac5f52a821bb00f3ef21...	54fc4ff419d5e05db5fe42906b...	63430	ico	CE
12	1b079952d7f8ea0edc2babd69...	587482ee4b3da3583df4057f5...	95240	ipe	RS

```
#1.2 - Get the time range between which the orders were placed.
```

QUERY-

```
SELECT
MIN(order_purchase_timestamp) AS start_time,
MAX(order_purchase_timestamp) AS end_time
FROM BUSSINESS_CASE_STUDY.orders;
```

OUTPUT-

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

#1.3 -Count the Cities & States of customers who ordered during the given period.

QUERY-

```
select
count( distinct customer_city) as Cities ,
count( distinct customer_state) as States
from BUSSINESS_CASE_STUDY.customers;
```

OUTPUT-

Row	Cities	States
1	4119	27

2 - In-depth Exploration:

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

QUERY-

```
select *
from (SELECT
EXTRACT(YEAR FROM order_purchase_timestamp) AS yr,
COUNT(*) AS count_per_yr
FROM BUSSINESS_CASE_STUDY.orders
GROUP BY EXTRACT(YEAR FROM order_purchase_timestamp)) k
order by k.yr;

#OR

WITH YearlyOrderCounts AS (
    SELECT
    EXTRACT(YEAR FROM order_purchase_timestamp) AS yr,
    COUNT(*) AS count_per_yr
    FROM BUSSINESS_CASE_STUDY.orders
    GROUP BY EXTRACT(YEAR FROM order_purchase_timestamp)
)
SELECT *
FROM YearlyOrderCounts
ORDER BY yr;
```

OUTPUT-

Row	yr	count_per_yr
1	2016	329
2	2017	45101
3	2018	54011

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

QUERY-

```
SELECT k.month_name , k.count_per_month
FROM(select
extract(month from order_purchase_timestamp ) as month_id,
FORMAT_TIMESTAMP('%B', order_purchase_timestamp) as
month_name ,
COUNT(*) AS count_per_month
FROM BUSSINESS_CASE_STUDY.orders
group by  FORMAT_TIMESTAMP('%B', order_purchase_timestamp)
, extract(month from order_purchase_timestamp )
) k
ORDER BY k.month_id ;
```

OUTPUT-

Row	month_name	count_per_month
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

```
/*question 2.3
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
min(k.Purchase_time) as start_period_time ,
max(k.Purchase_time) as end_period_time ,
CASE
WHEN k.Purchase_time BETWEEN '00:00:00' and '06:59:59'
THEN 'Dawn'
WHEN k.Purchase_time BETWEEN '07:00:00' and '12:59:59'
THEN 'Mornings'
WHEN k.Purchase_time BETWEEN '13:00:00' and '18:59:59'
THEN 'Afternoon'

WHEN k.Purchase_time BETWEEN '19:00:00' and '23:59:59'
THEN 'Night'
END as purchase_period ,
count(*) as total_orders_in_period
from (select
customer_id ,
extract (time from order_purchase_timestamp ) as
Purchase_time
FROM BUSSINESS_CASE_STUDY.orders
) as k
group by 3
order by 1 , 2 ;
```

OUTPUT-

Row	start_period_time	end_period_time	purchase_period	total_orders_in_period
1	00:00:00	06:59:58	Dawn	5242
2	07:00:26	12:59:59	Mornings	27733
3	13:00:00	18:59:59	Afternoon	38135
4	19:00:01	23:59:59	Night	28331

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

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

QUERY-

```
SELECT
  k.customer_state,
  CASE
    WHEN k.month_of_order = 1 THEN 'January'
    WHEN k.month_of_order = 2 THEN 'February'
    WHEN k.month_of_order = 3 THEN 'March'
    WHEN k.month_of_order = 4 THEN 'April'
    WHEN k.month_of_order = 5 THEN 'May'
    WHEN k.month_of_order = 6 THEN 'June'
    WHEN k.month_of_order = 7 THEN 'July'
    WHEN k.month_of_order = 8 THEN 'August'
    WHEN k.month_of_order = 9 THEN 'September'
    WHEN k.month_of_order = 10 THEN 'October'
    WHEN k.month_of_order = 11 THEN 'November'
    WHEN k.month_of_order = 12 THEN 'December'
  END AS month_name,
  k.month_of_order,
  k.number_of_order,
  LAG(k.number_of_order) OVER (PARTITION BY
k.customer_state ORDER BY k.month_of_order) as
prev_month_orders,
  LAG(k.number_of_order) OVER (PARTITION BY
k.customer_state ORDER BY k.month_of_order) -
k.number_of_order as MoM_diff
FROM (
  SELECT
    c.customer_state,
    EXTRACT(MONTH FROM o.order_purchase_timestamp) AS
month_of_order,
    COUNT(*) AS number_of_order
  FROM BUSSINESS_CASE_STUDY.customers c
  JOIN BUSSINESS_CASE_STUDY.orders o ON c.customer_id =
o.customer_id
  GROUP BY 1, 2
  ORDER BY 1, 2) k
ORDER BY k.customer_state, k.month_of_order;
```

OUTPUT-

Row	customer_state	month_name	month_of_order	number_of_order	prev_month_orders	MoM_diff
1	AC	January	1	8	null	null
2	AC	February	2	6	8	2
3	AC	March	3	4	6	2
4	AC	April	4	9	4	-5
5	AC	May	5	10	9	-1
6	AC	June	6	7	10	3
7	AC	July	7	9	7	-2
8	AC	August	8	7	9	2
9	AC	September	9	5	7	2
10	AC	October	10	6	5	-1
11	AC	November	11	5	6	1
12	AC	December	12	5	5	0
13	AL	January	1	39	null	null
14	AL	February	2	39	39	0

#3.2 How are the customers distributed across all the states?

QUERY-

```
SELECT
customer_city ,
customer_state ,
count(*) as customer_cnt
FROM BUSSINESS_CASE_STUDY.customers
group by 1 ,  2
order by 2 , 1 ;
```

OUTPUT-

Row	customer_city	customer_state	customer_cnt
1	brasileia	AC	1
2	cruzeiro do sul	AC	3
3	epitaciolandia	AC	1
4	manoel urbano	AC	1
5	porto acre	AC	1
6	rio branco	AC	70
7	senador guiomard	AC	2
8	xapuri	AC	2
9	agua branca	AL	1
10	anadia	AL	2
11	arapiraca	AL	29
12	atalaia	AL	1
13	barra de santo antonio	AL	2

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

#4.1 -

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 PAYMENT_CTE AS (

  SELECT
      EXTRACT(YEAR FROM o.order_purchase_timestamp) AS
ORDER_YEAR,
      cast(SUM(p.payment_value) AS INT64 ) TOT_S
  FROM BUSSINESS_CASE_STUDY.orders o
  JOIN BUSSINESS_CASE_STUDY.payments p ON o.order_id =
p.order_id
  WHERE EXTRACT(MONTH FROM o.order_purchase_timestamp)
BETWEEN 1 AND 8
  GROUP BY ORDER_YEAR
  order by 1
  )
 select *
  from (SELECT
  ORDER_YEAR,
  TOT_S ,
  Lead(TOT_S) OVER(order by TOT_S) as tot_s_next_year ,
  CONCAT(ROUND(((LEAD(TOT_S) OVER (ORDER BY TOT_S) -
TOT_S) / TOT_S) * 100, 2), '%') AS percent_increase
  FROM PAYMENT_CTE
  GROUP BY ORDER_YEAR , TOT_S
  ORDER BY ORDER_YEAR) k
 where k.tot_s_next_year is not null;
```

OUTPUT -

Row	ORDER_YEAR	TOT_S	tot_s_next_year	percent_increase
1	2017	3669022	8694734	136.98%

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

QUERY-

```
SELECT
DISTINCT c.customer_state ,
ROUND(SUM(p.payment_value),2)  as TOT_value ,
round(avg(p.payment_value),2)  as AVG_VALUE
FROM BUSSINESS_CASE_STUDY.customers c
join BUSSINESS_CASE_STUDY.orders o on c.customer_id =
o.customer_id
join BUSSINESS_CASE_STUDY.payments p on p.order_id =
o.order_id
group by 1
order by 1;
```

output-

Row	customer_state	TOT_value	AVG_VALUE
1	AC	19680.62	234.29
2	AL	96962.06	227.08
3	AM	27966.93	181.6
4	AP	16262.8	232.33
5	BA	616645.82	170.82
6	CE	279464.03	199.9
7	DF	355141.08	161.13
8	ES	325967.55	154.71
9	GO	350092.31	165.76
10	MA	152523.02	198.86
11	MG	1872257.26	154.71
12	MS	137534.84	186.87
13	MT	187029.29	195.23

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

QUERY

```
SELECT
DISTINCT c.customer_state ,
ROUND(SUM(oi.freight_value),2)  as TOT_freight_value ,
round(avg(oi.freight_value),2)  as AVG_freight_VALUE
from BUSSINESS_CASE_STUDY.customers c
join BUSSINESS_CASE_STUDY.orders o
on c.customer_id = o.customer_id
join BUSSINESS_CASE_STUDY.order_items oi
```

```
on o.order_id = oi.order_id
group by 1
order by 1;
```

Row	customer_state	TOT_freight_value	AVG_freight_VALUE
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
12	MS	19144.03	23.37
13	MT	29715.43	28.17
14	PA	38699.3	35.83

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

```
#5.1
/*
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-

```sql
SELECT
order_id ,
customer_id ,
order_purchase_timestamp ,
order_delivered_carrier_date ,
order_estimated_delivery_date ,
DATE_DIFF(order_delivered_carrier_date ,
order_purchase_timestamp, DAY) AS DAYS_TOOK_IN_DELIVERY ,
DATE_DIFF(order_delivered_carrier_date ,
order_estimated_delivery_date, DAY) AS
estimated_actual_delivery_difference_in_days ,
CASE
WHEN DATE_DIFF(order_delivered_carrier_date ,
order_estimated_delivery_date, DAY) < 0
THEN CONCAT(DATE_DIFF(order_delivered_carrier_date ,
order_estimated_delivery_date, DAY)*-1 , ' DAYS EARLY ')
WHEN DATE_DIFF(order_delivered_carrier_date ,
order_estimated_delivery_date, DAY) > 0
THEN CONCAT(DATE_DIFF(order_delivered_carrier_date ,
order_estimated_delivery_date, DAY)*1 , ' DAYS LATER ')
ELSE 'ON TIME '
END AS ORDER_DELIVERY_STATUS
FROM BUSSINESS_CASE_STUDY.orders
where order_status = 'delivered' and
order_delivered_carrier_date is not null;
```

OUTPUT-

Row	order_id	customer_id	order_purchase_timestamp	order_delivered_carrier_date	order_estimated_delivery_date	DAYS_TOOK_IN_DELIVERY	estimated_actual_	ORDER_DELIVERY_STATUS
1	0312ecf907...	45cce495588388...	2017-05-10 22:02:40 UTC	2017-05-17 10:06:53 UTC	2017-05-18 00:00:00 UTC	6	0	ON TIME
2	cec8f5f7a1...	6be61d704faaff9...	2017-03-17 15:56:47 UTC	2017-04-04 10:53:37 UTC	2017-05-18 00:00:00 UTC	17	-43	43 DAYS EARLY
3	2d846c030...	5a3473648da10d...	2017-05-10 15:18:16 UTC	2017-05-15 09:14:35 UTC	2017-05-18 00:00:00 UTC	4	-2	2 DAYS EARLY
4	58527ee47...	b7d68eb92ede54...	2017-03-20 11:01:17 UTC	2017-03-22 08:50:27 UTC	2017-05-18 00:00:00 UTC	1	-56	56 DAYS EARLY
5	7752cfbc93...	f948127c1046ac...	2017-04-26 13:31:30 UTC	2017-05-16 12:18:15 UTC	2017-05-18 00:00:00 UTC	19	-1	1 DAYS EARLY
6	10ed5499d...	2bf569d940353f...	2017-03-21 13:38:25 UTC	2017-04-04 16:30:16 UTC	2017-05-18 00:00:00 UTC	14	-43	43 DAYS EARLY
7	cb837ba27...	e8d5a2b4873087...	2017-05-10 04:12:20 UTC	2017-05-17 12:56:43 UTC	2017-05-18 00:00:00 UTC	7	0	ON TIME
8	818996ea2...	19b1122a589ca4...	2018-08-20 15:56:23 UTC	2018-08-22 14:28:00 UTC	2018-10-04 00:00:00 UTC	1	-42	42 DAYS EARLY
9	d195cac9cc...	a3a156d272fd0b...	2018-08-12 18:14:29 UTC	2018-08-14 13:41:00 UTC	2018-10-04 00:00:00 UTC	1	-50	50 DAYS EARLY
10	64eeb35d3...	d00827c5fac201...	2018-08-16 07:55:32 UTC	2018-08-17 10:40:00 UTC	2018-10-04 00:00:00 UTC	1	-47	47 DAYS EARLY
11	2691ae869f...	e551bab5d422fd...	2018-08-22 22:39:54 UTC	2018-08-23 12:28:00 UTC	2018-10-04 00:00:00 UTC	0	-41	41 DAYS EARLY
12	1cd147d1c...	b28dc057a0489d...	2018-08-20 17:04:34 UTC	2018-08-21 15:33:00 UTC	2018-10-04 00:00:00 UTC	0	-43	43 DAYS EARLY
13	b36d2e6b1...	56ffad14ea1b4d2...	2018-08-09 19:17:50 UTC	2018-08-10 12:34:00 UTC	2018-10-04 00:00:00 UTC	0	-54	54 DAYS EARLY

#5.2
/*Find out the top 5 states with the highest & lowest average freight value.*/

```
WITH FreightStats AS (
  SELECT
    c.customer_state,
    ROUND(AVG(oi.freight_value), 2) AS avg_freight_value
  FROM BUSSINESS_CASE_STUDY.customers c
  JOIN BUSSINESS_CASE_STUDY.orders o ON c.customer_id =
o.customer_id
  JOIN BUSSINESS_CASE_STUDY.order_items oi ON o.order_id =
oi.order_id
  GROUP BY 1
)
select * from (
SELECT
  customer_state,
  avg_freight_value,
concat('top ' , DENSE_RANK() OVER (ORDER BY
avg_freight_value DESC) ,'th rank'  ) as rank_status
FROM FreightStats
ORDER BY avg_freight_value DESC
LIMIT 5)

UNION all

select * from (SELECT
  customer_state,
  avg_freight_value,
  concat('bottom ' , DENSE_RANK() OVER (ORDER BY
avg_freight_value asc) ,'th rank'  )
FROM FreightStats
ORDER BY avg_freight_value
LIMIT 5);
```

Output-

Row	customer_state	avg_freight_value	rank_status
1	RR	42.98	top 1th rank
2	PB	42.72	top 2th rank
3	RO	41.07	top 3th rank
4	AC	40.07	top 4th rank
5	PI	39.15	top 5th rank
6	SP	15.15	bottom 1th rank
7	PR	20.53	bottom 2th rank
8	MG	20.63	bottom 3th rank
9	RJ	20.96	bottom 4th rank
10	DF	21.04	bottom 5th rank

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

QUERY-

```
with customers_and_orders as
(
select
o.order_id ,
o.customer_id ,
o.order_purchase_timestamp ,
o.order_delivered_customer_date ,
date_diff(o.order_delivered_customer_date ,
o.order_purchase_timestamp , day) as
time_taken_in_delivery ,
c.customer_state
from BUSSINESS_CASE_STUDY.orders o join
BUSSINESS_CASE_STUDY.customers c on o.customer_id =
c.customer_id
where order_status = 'delivered'and
o.order_delivered_carrier_date is not null
)

select * from (select
customer_state ,
round(avg(time_taken_in_delivery) , 2) as
avg_delivery_time ,
```

```
concat('top ' , DENSE_RANK() OVER (ORDER BY
round(avg(time_taken_in_delivery) , 2) desc) ,'th average
value'  ) as rank_status
from customers_and_orders
group by customer_state
order by 2 desc
LIMIT 5)
union all
select * from (select
customer_state ,
round(avg(time_taken_in_delivery) , 2) as
avg_delivery_time ,
concat('bottom ' , DENSE_RANK() OVER (ORDER BY
round(avg(time_taken_in_delivery) , 2) asc) ,'th average
value'  ) as rank_status
from customers_and_orders
group by customer_state
order by 2 asc
LIMIT 5);
```

Output-

Row	customer_state	avg_delivery_time	rank_status
1	RR	28.98	top 1th average value
2	AP	26.73	top 2th average value
3	AM	25.99	top 3th average value
4	AL	24.04	top 4th average value
5	PA	23.32	top 5th average value
6	SP	8.3	bottom 1th average value
7	PR	11.53	bottom 2th average value
8	MG	11.54	bottom 3th average value
9	DF	12.51	bottom 4th average value
10	SC	14.48	bottom 5th average value

INSIGHTS -

• The query analyzes order delivery times in different states.

• It calculates the average time it takes for orders to be delivered in each state.

• States with the fastest delivery times are ranked at the top, and those with the slowest times are ranked at the bottom.

• The top 5 states with the fastest delivery times are shown, along with their average delivery times and ranks.

• Similarly, the bottom 5 states with the slowest delivery times are displayed with their average times and ranks.

```
#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.
```

```
QUERY-

WITH order_cte AS (
  SELECT
    o.order_id,
    c.customer_id,
    o.order_purchase_timestamp,
    o.order_status,
    o.order_delivered_carrier_date,
    o.order_estimated_delivery_date,
    DATE_DIFF(o.order_delivered_carrier_date,
o.order_purchase_timestamp, DAY) AS days_took_in_delivery,
    DATE_DIFF(o.order_delivered_carrier_date,
o.order_estimated_delivery_date, DAY) AS
estimated_actual_delivery_difference_in_days,
    c.customer_state
  FROM BUSSINESS_CASE_STUDY.orders o
  JOIN BUSSINESS_CASE_STUDY.customers c ON o.customer_id =
c.customer_id
  WHERE order_status = 'delivered' AND
order_delivered_carrier_date IS NOT NULL
)

SELECT
  customer_state,
```

```
  cast(AVG(estimated_actual_delivery_difference_in_days)
as int64) AS avg_delivery_diff,




CASE
    WHEN AVG(estimated_actual_delivery_difference_in_days)
< 0
    THEN CONCAT('Delivery average in state= ',
CAST(AVG(estimated_actual_delivery_difference_in_days)* -
1  AS INT64 ), ' days early.')
    ELSE 'On time'
  END AS status
FROM order_cte
GROUP BY customer_state
ORDER BY 2
LIMIT 5;
```

Output-

Row	customer_state	avg_delivery_diff	status
1	AP	-42	Delivery average in state= 42 days early.
2	RR	-42	Delivery average in state= 42 days early.
3	AM	-42	Delivery average in state= 42 days early.
4	AC	-37	Delivery average in state= 37 days early.
5	RO	-36	Delivery average in state= 36 days early.

```
INSIGHTS-
    1-  This query calculates the average difference in days for each state
    2-  CASE statement correctly determines whether the delivery is early or on time
```

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

QUERY-

```sql
with cte as (
SELECT
  extract(month from order_purchase_timestamp) as
order_month ,
  FORMAT_TIMESTAMP('%B', order_purchase_timestamp) as
month_name,
  p.payment_type,
  count(p.order_id) as no_of_orders
FROM BUSSINESS_CASE_STUDY.payments p
JOIN BUSSINESS_CASE_STUDY.orders o ON o.order_id =
p.order_id
GROUP BY order_month, month_name, p.payment_type
order by 3 , 1)
select
month_name ,
payment_type ,
no_of_orders
from cte;
```

OUTPUT-

Row	month_name	payment_type	no_of_orders
1	January	UPI	1715
2	February	UPI	1723
3	March	UPI	1942
4	April	UPI	1783
5	May	UPI	2035
6	June	UPI	1807
7	July	UPI	2074
8	August	UPI	2077
9	September	UPI	903
10	October	UPI	1056
11	November	UPI	1509
12	December	UPI	1160
13	January	credit_card	6103
14	February	credit_card	6609
15	March	credit_card	7707
16	April	credit_card	7301
17	May	credit_card	8350
18	June	credit_card	7276
19	July	credit_card	7841
20	August	credit_card	8269
21	September	credit_card	3286
22	October	credit_card	3778
23	November	credit_card	5897
24	December	credit_card	4378
25	January	debit_card	118

#6.2 Find the no. of orders placed on the basis of the payment installments that have been paid.--For total number of orders placed on the basis of the payment installments

QUERY-

```
select count(*) as total_order_based_on_installments
from BUSSINESS_CASE_STUDY.payments p
join BUSSINESS_CASE_STUDY.orders o
on o.order_id = p.order_id
where p.payment_installments > 0 AND p.payment_value > 0;
```

output-

Row	total_order_based_on_installments
1	103875

INSIGHTS-

- Counts the total number of orders.
- Filters orders where both payment installments and payment values are greater than 0.
- Provides a single count of orders meeting the criteria.

```
SELECT
COUNT(o.order_id) AS num_of_orders,
payment_installments AS num_of_installments
FROM BUSSINESS_CASE_STUDY.orders o
JOIN BUSSINESS_CASE_STUDY.payments p
ON o.order_id = p.order_id
WHERE p.payment_installments > 0 AND p.payment_value > 0
GROUP BY payment_installments
ORDER BY num_of_installments;
```

Output:-

Row	num_of_orders	num_of_installments
1	52537	1
2	12413	2
3	10461	3
4	7098	4
5	5239	5
6	3920	6
7	1626	7
8	4268	8
9	644	9
10	5328	10
11	23	11
12	133	12
13	16	13
14	15	14

INSIGHTS-

- Counts the number of orders.
- Filters orders where both payment installments and payment values are greater than 0.
- Groups the results by the number of payment installments.
- Provides a count for each installment count (e.g., how many orders have 1 installment, how many have 2, etc.).
- Orders the results by the number of installments.

THANK YOU