Business Case – Target SQL

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

```
SELECT
  * EXCEPT (ordinal_position, is_generated, generation_expression, is_stored,
is_hidden, is_updatable, is_system_defined, is_partitioning_column,
clustering_ordinal_position, collation_name, column_default, rounding_mode)
FROM
  `lucid-totality-409609.retail_dataset`.INFORMATION_SCHEMA.COLUMNS
WHERE
  table_name = 'customers';
```

Query results

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

Row	table_catalog	table_schema	table_name	column_name	is_nullable	data_type
1	lucid-totality-409609	retail_dataset	customers	customer_id	YES	STRING
2	lucid-totality-409609	retail_dataset	customers	customer_unique_id	YES	STRING
3	lucid-totality-409609	retail_dataset	customers	customer_zip_code_prefix	YES	INT64
4	lucid-totality-409609	retail_dataset	customers	customer_city	YES	STRING
5	lucid-totality-409609	retail_dataset	customers	customer_state	YES	STRING

Insights:

- We could see the customers dataset has 4 columns of String Datatype and 1 column of numerical datatype.
- There are 99441 records in customers table.
- We have 99441 unique and non-null values in customer_id column.
- We have 96096 unique and non-null values in customer_unique_id column.
- There are 14994 unique groups and non-null values in customer_zip_code_prefix column.
- There are 4119 unique groups and non-null values in customer_city column.
- There are 27 unique groups and non-null values in customer_state column.

Recommendation: N/A

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

```
SELECT MIN(order_purchase_timestamp) AS `First_order`,
MAX(order_purchase_timestamp) `last_order`
FROM lucid-totality-409609.retail_dataset.orders
```

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

Insights:

- The orders range between 2016-09-04 21:15:19 UTC AND 2018-10-17 17:30:18 UTC.

Recommendation: N/A

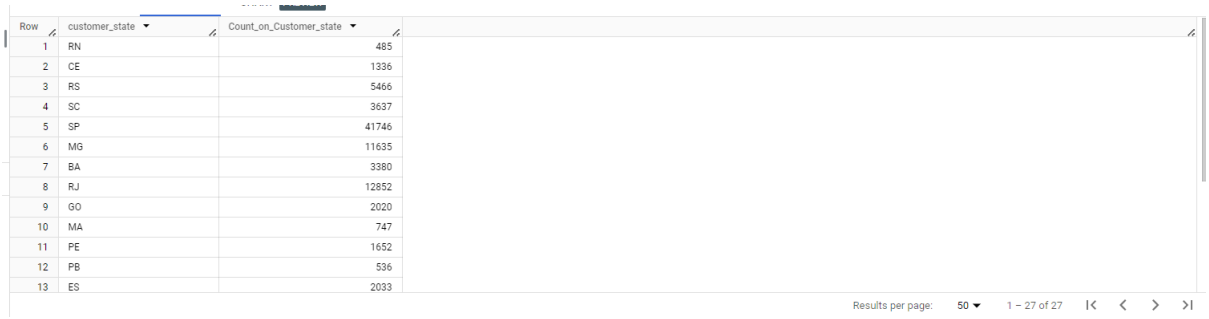
Q. Count the Cities & States of customers who ordered during the given period.

```
SELECT c.customer_city, COUNT(c.customer_city) AS `Count_on_Customer_city`
FROM lucid-totality-409609.retail_dataset.customers AS `c`
JOIN lucid-totality-409609.retail_dataset.orders AS `o`
  ON o.customer_id = c.customer_id
WHERE o.order_purchase_timestamp BETWEEN
  (SELECT MIN(order_purchase_timestamp) FROM lucid-totality-
409609.retail_dataset.orders) AND
  (SELECT MAX(order_purchase_timestamp) FROM lucid-totality-
409609.retail_dataset.orders)
  GROUP BY c.customer_city
```

Row	customer_city	Count_on_Customer_city
1	acu	3
2	ico	8
3	ipe	2
4	ipu	4
5	ita	3
6	itu	136
7	jau	74
8	luz	2
9	poa	85
10	uba	53
11	una	5
12	anta	4
13	avai	1

Results per page: 50 1 – 50 of 4119

```
SELECT c.customer_state, COUNT(c.customer_state) AS `Count_on_Customer_state`
FROM lucid-totality-409609.retail_dataset.customers AS `c`
JOIN lucid-totality-409609.retail_dataset.orders AS `o`
  ON o.customer_id = c.customer_id
WHERE o.order_purchase_timestamp BETWEEN
  (SELECT MIN(order_purchase_timestamp) FROM lucid-totality-
409609.retail_dataset.orders) AND
  (SELECT MAX(order_purchase_timestamp) FROM lucid-totality-
409609.retail_dataset.orders)
  GROUP BY c.customer_state
```



Row	customer_state	Count_on_Customer_state
1	RN	485
2	CE	1336
3	RS	5466
4	SC	3637
5	SP	41746
6	MG	11635
7	BA	3380
8	RJ	12852
9	GO	2020
10	MA	747
11	PE	1652
12	PB	536
13	ES	2033

Results per page: 50 1 – 27 of 27

Insights:

- The customers are from 4119 different cities from 27 different states.

Recommendation: N/A

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

```
SELECT DISTINCT EXTRACT(YEAR FROM order_purchase_timestamp) AS `Year_Of_Purchase`,
  COUNT(order_purchase_timestamp) OVER(PARTITION BY EXTRACT(YEAR FROM
order_purchase_timestamp)) AS `Count_of Purchase_by_Year`
FROM `lucid-totality-409609.retail_dataset.orders`
ORDER BY Year_Of_Purchase ASC
```

Row	Year_Of_Purchase	Count_of Purchase_by_Year
1	2016	329
2	2017	45101
3	2018	54011

Insights:

- Compared to 2016, the numbers of 2017 had a huge growth.
- Compared to 2017, the count of 2018 is higher
- We could see a growing trend from later years

Recommendation: N/A

```
SELECT DISTINCT EXTRACT(YEAR FROM order_purchase_timestamp) AS `Year_Of_Purchase`,
  EXTRACT(MONTH FROM order_purchase_timestamp) AS `Month_Of_Purchase`,
  COUNT(order_purchase_timestamp) OVER(PARTITION BY EXTRACT(MONTH FROM
order_purchase_timestamp)) AS `Count_of Purchase_by_Month`
FROM `lucid-totality-409609.retail_dataset.orders`
ORDER BY Year_Of_Purchase ASC, Month_Of_Purchase ASC
```

Row	Year_Of_Purchase	Month_Of_Purchase	Count_of Purchase_by_Month
1	2016	9	4305
2	2016	10	4959
3	2016	12	5674
4	2017	1	8069
5	2017	2	8508
6	2017	3	9893
7	2017	4	9343
8	2017	5	10573
9	2017	6	9412
10	2017	7	10318
11	2017	8	10843
12	2017	9	4305
13	2017	10	4959
14	2017	11	7544
15	2017	12	5674

Insights:

- In 2016, We could see upward trend over the months of October, November and December

- In 2017, There was a steady growth and only during September and December, there was a slight decrease in the orders.
- In 2018, The orders were consistently rising and dropping between 6167 and 7269 till August and the orders were not good during September and October as there was very less purchase.

Recommendation: N/A

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

```
SELECT DISTINCT EXTRACT(YEAR FROM order_purchase_timestamp) AS `Year_Of_Purchase`,
  EXTRACT(MONTH FROM order_purchase_timestamp) AS `Month_Of_Purchase`,
  COUNT(order_purchase_timestamp) OVER(PARTITION BY EXTRACT(MONTH FROM
order_purchase_timestamp)) AS `Count_of Purchase_by_Month`
FROM `lucid-totality-409609.retail_dataset.orders`
ORDER BY Year_Of_Purchase ASC, Month_Of_Purchase ASC
```

Row	Year_Of_Purchase	Month_Of_Purchase	Count_of Purchase_by_Month
1	2016	9	4305
2	2016	10	4959
3	2016	12	5674
4	2017	1	8069
5	2017	2	8508
6	2017	3	9893
7	2017	4	9343
8	2017	5	10573
9	2017	6	9412
10	2017	7	10318
11	2017	8	10843
12	2017	9	4305
13	2017	10	4959
14	2017	11	7544

Insights:

- We can only find a pattern during December month among the years 2016,2017,2018 as there is a drop in the number of orders.

Recommendation: N/A

Q. 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 DISTINCT T.TimeOfDay, COUNT(*) OVER(PARTITION BY T.TimeOfDay) AS
`Within_Period_Count_of_Orders`
FROM (
  SELECT order_purchase_timestamp,
  (
    CASE
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 0 AND 6 THEN  '0-6'
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 7 AND 12 THEN  '7-12'
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 13 AND 18 THEN  '13-
18'
    ELSE '19-24'
    END
  ) AS `TimeOfDay`
FROM `lucid-totality-409609.retail_dataset.orders`
) AS `T`
```

Row	TimeOfDay	Within_Period_Count_of_Orders
1	0-6	5242
2	19-24	28331
3	13-18	38135
4	7-12	27733

Insights:

- We can see the most number of orders were placed between 13 and 18 hours which is during Afternoon time.
- There are purchases made all the time but comparatively only during Dawn we have very less when we consider the period between 2016 – 2018.

Recommendation: N/A

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

```
/*Get the month on month no. of orders placed in each state.*/

SELECT DISTINCT c.customer_state, EXTRACT(MONTH FROM o.order_purchase_timestamp) AS
`MonthOfPurchase`,
  COUNT(o.order_id) OVER(PARTITION BY c.customer_state, EXTRACT(MONTH FROM
o.order_purchase_timestamp)) AS `OrderCount`
FROM `lucid-totality-409609.retail_dataset.customers` AS `c`
JOIN `lucid-totality-409609.retail_dataset.orders` AS `o`
  ON o.customer_id = c.customer_id
ORDER BY c.customer_state ASC, MonthOfPurchase ASC
```

Row	customer_state	MonthOfPurchase	OrderCount
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
10	AC	10	6
11	AC	11	5

Insights:

- N/A

Recommendation: N/A

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

```
  /*How are the customers distributed across all the states?*/
SELECT
  customer_state,
  COUNT(*) AS `CountOfCustomers`
FROM
  `lucid-totality-409609.retail_dataset.customers`
GROUP BY
  customer_state
```

Row	customer_state	CountOfCustomers
1	RN	485
2	CE	1336
3	RS	5466
4	SC	3637
5	SP	41746
6	MG	11635
7	BA	3380
8	RJ	12852
9	GO	2020
10	MA	747

```
/*Below table displays the result by descending order of count of customer*/

SELECT DISTINCT customer_state, COUNT(customer_unique_id) OVER(PARTITION BY
customer_state) AS `Number_Of_Customers_Per_state`,
  COUNT(customer_unique_id) OVER() AS `Total_Count`,
  ROUND(COUNT(customer_unique_id) OVER(PARTITION BY customer_state) /
COUNT(customer_unique_id) OVER(),2) AS
`Percentage_Of_Number_Of_Customers_Per_state`,
FROM `lucid-totality-409609.retail_dataset.customers`
ORDER BY Number_Of_Customers_Per_state DESC
```

Row	customer_state	CountOfCustomers
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637
7	BA	3380
8	DF	2140
9	ES	2033
10	GO	2020

Row	customer_state	Number_Of_Custom	Total_Count	Percentage_Of_Number_Of_Customers_Per_state
1	SP	41746	99441	0.42
2	RJ	12852	99441	0.13
3	MG	11635	99441	0.12
4	RS	5466	99441	0.05
5	PR	5045	99441	0.05
6	SC	3637	99441	0.04
7	BA	3380	99441	0.03
8	DF	2140	99441	0.02
9	ES	2033	99441	0.02
10	GO	2020	99441	0.02
11	PE	1652	99441	0.02

Insights:

- We can see the top 5 states where the customers are in majority are SP, RJ, MG, RS, PR.

Recommendation: N/A

Q. Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).

```
SELECT DISTINCT EXTRACT(YEAR FROM order_purchase_timestamp) AS `Year`,
  SUM(payment_value) OVER(PARTITION BY EXTRACT(YEAR FROM order_purchase_timestamp))
AS `Total_Purchase_Value_Based_on_Years`

FROM `lucid-totality-409609.retail_dataset.orders` AS `o`
JOIN `lucid-totality-409609.retail_dataset.payments` AS `p`
  ON p.order_id = o.order_id
WHERE EXTRACT(YEAR FROM order_purchase_timestamp) != 2016
  AND
  EXTRACT(MONTH FROM order_purchase_timestamp) NOT IN (9,10,11,12)
ORDER BY Year ASC
```

Row	Year	Total_Purchase_Value_Based_on_Years
1	2017	3669022.12
2	2018	8694733.84

```
SELECT DISTINCT EXTRACT(YEAR FROM order_purchase_timestamp) AS `Year`,
```

```
  SUM(payment_value) OVER(PARTITION BY EXTRACT(YEAR FROM order_purchase_timestamp))
AS `Total_Purchase_Value_Based_on_Years`,
  SUM(payment_value) OVER() AS `Total_value`,
  SUM(payment_value) OVER(PARTITION BY EXTRACT(YEAR FROM
order_purchase_timestamp))/ SUM(payment_value) OVER() AS
`Percentage_of_Payments_By_Years`
FROM `lucid-totality-409609.retail_dataset.orders` AS `o`
JOIN `lucid-totality-409609.retail_dataset.payments` AS `p`
  ON p.order_id = o.order_id
WHERE EXTRACT(YEAR FROM order_purchase_timestamp) != 2016
  AND
  EXTRACT(MONTH FROM order_purchase_timestamp) NOT IN (9,10,11,12)
ORDER BY Year ASC#, Month ASC
```

Row	Year	Total_Purchase_Value_Based_on_Years	Total_value	Percentage_of_Payments_By_Years
1	2017	3669022.12	12363755.959999995	0.29675627146558475
2	2018	8694733.84	12363755.959999995	0.70324372853441564

```
SELECT DISTINCT EXTRACT(YEAR FROM order_purchase_timestamp) AS `Year`,
  EXTRACT(MONTH FROM order_purchase_timestamp) AS `Month`,
  SUM(payment_value) OVER(PARTITION BY EXTRACT(MONTH FROM
order_purchase_timestamp)) AS `Total_Purchase_Value_Based_on_Months`,
  SUM(payment_value) OVER() AS `Total_value`,
  SUM(payment_value) OVER(PARTITION BY EXTRACT(MONTH FROM
order_purchase_timestamp))/ SUM(payment_value) OVER() AS
`Percentage_of_Payments_By_Months_2017`
FROM `lucid-totality-409609.retail_dataset.orders` AS `o`
JOIN `lucid-totality-409609.retail_dataset.payments` AS `p`
  ON p.order_id = o.order_id
WHERE EXTRACT(YEAR FROM order_purchase_timestamp) NOT IN (2016, 2018)
  AND
  EXTRACT(MONTH FROM order_purchase_timestamp) NOT IN (9,10,11,12)
ORDER BY Year ASC, Month ASC
```

Row	Year	Month	Total_Purchase_Valu	Total_value	Percentage_of_Paym
1	2017	1	138488.04	3669022.119999…	0.037745218063…
2	2017	2	291908.01	3669022.119999…	0.079560166293…
3	2017	3	449863.6	3669022.119999…	0.122611307669…
4	2017	4	417788.03	3669022.119999…	0.113869040942…
5	2017	5	592918.82	3669022.119999…	0.161601320626…
6	2017	6	511276.38	3669022.119999…	0.139349495118…
7	2017	7	592382.92	3669022.119999…	0.161455259909…
8	2017	8	674396.32	3669022.119999…	0.183808191377…

```
SELECT DISTINCT EXTRACT(YEAR FROM order_purchase_timestamp) AS `Year`,
  EXTRACT(MONTH FROM order_purchase_timestamp) AS `Month`,
  SUM(payment_value) OVER(PARTITION BY EXTRACT(MONTH FROM
order_purchase_timestamp)) AS `Total_Purchase_Value_Based_on_Months`,
  SUM(payment_value) OVER() AS `Total_value`,
```

```
  SUM(payment_value) OVER(PARTITION BY EXTRACT(MONTH FROM
order_purchase_timestamp))/ SUM(payment_value) OVER() AS
`Percentage_of_Payments_By_Months_2018`
FROM `lucid-totality-409609.retail_dataset.orders` AS `o`
JOIN `lucid-totality-409609.retail_dataset.payments` AS `p`
  ON p.order_id = o.order_id
WHERE EXTRACT(YEAR FROM order_purchase_timestamp) NOT IN (2016, 2017)
  AND
  EXTRACT(MONTH FROM order_purchase_timestamp) NOT IN (9,10,11,12)
ORDER BY Year ASC, Month ASC
```

Row	Year	Month	Total_Purchase_Valu	Total_value	Percentage_of_Payn
1	2018	1	1115004.18	8694733.84	0.128239023818...
2	2018	2	992463.34	8694733.84	0.114145338806...
3	2018	3	1159652.12	8694733.84	0.133374079223...
4	2018	4	1160785.48	8694733.84	0.133504429389...
5	2018	5	1153982.15	8694733.84	0.132721963804...
6	2018	6	1023880.5	8694733.84	0.117758693807...
7	2018	7	1066540.75	8694733.84	0.122665140719...
8	2018	8	1022425.32	8694733.84	0.117591330432...

```
WITH `2018` AS

(
  SELECT DISTINCT EXTRACT(YEAR FROM order_purchase_timestamp) AS `Year`,
  EXTRACT(MONTH FROM order_purchase_timestamp) AS `Month`,
  SUM(payment_value) OVER(PARTITION BY EXTRACT(MONTH FROM
order_purchase_timestamp)) AS `Total_Purchase_Value_Based_on_Months`,
  SUM(payment_value) OVER() AS `Total_value`,
  SUM(payment_value) OVER(PARTITION BY EXTRACT(MONTH FROM
order_purchase_timestamp))/ SUM(payment_value) OVER() AS
`Percentage_of_Payments_By_Months_2018`
FROM `lucid-totality-409609.retail_dataset.orders` AS `o`
JOIN `lucid-totality-409609.retail_dataset.payments` AS `p`
  ON p.order_id = o.order_id
WHERE EXTRACT(YEAR FROM order_purchase_timestamp) NOT IN (2016, 2017)
  AND
  EXTRACT(MONTH FROM order_purchase_timestamp) NOT IN (9,10,11,12)
ORDER BY Year ASC, Month ASC
),

`2017` AS

(
  SELECT DISTINCT EXTRACT(YEAR FROM order_purchase_timestamp) AS `Year`,
  EXTRACT(MONTH FROM order_purchase_timestamp) AS `Month`,
  SUM(payment_value) OVER(PARTITION BY EXTRACT(MONTH FROM
order_purchase_timestamp)) AS `Total_Purchase_Value_Based_on_Months`,
  SUM(payment_value) OVER() AS `Total_value`,
  SUM(payment_value) OVER(PARTITION BY EXTRACT(MONTH FROM
order_purchase_timestamp))/ SUM(payment_value) OVER() AS
`Percentage_of_Payments_By_Months_2017`
FROM `lucid-totality-409609.retail_dataset.orders` AS `o`
JOIN `lucid-totality-409609.retail_dataset.payments` AS `p`
```

```
  ON p.order_id = o.order_id
WHERE EXTRACT(YEAR FROM order_purchase_timestamp) NOT IN (2016, 2018)
  AND
  EXTRACT(MONTH FROM order_purchase_timestamp) NOT IN (9,10,11,12)
ORDER BY Year ASC, Month ASC
)

SELECT T17.Year, T17.Month, T17.Percentage_of_Payments_By_Months_2017, T18.Year,
T18.Month, T18.Percentage_of_Payments_By_Months_2018,
  IF(T17.Percentage_of_Payments_By_Months_2017 <
T18.Percentage_of_Payments_By_Months_2018, 'Downward', 'Upward') AS
`Trend_based_on_month`
FROM `2017` AS `T17`
JOIN `2018` AS `T18`
  ON T18.Month = T17.Month
ORDER BY T17.Month ASC
```

Row	Year	Month	Percentage_of_Payn	Year_1	Month_1	Percentage_of_Payn	Trend_based_on_month
1	2017	1	0.037745218063...	2018	1	0.128239023818...	Downward
2	2017	2	0.079560166293...	2018	2	0.114145338806...	Downward
3	2017	3	0.122611307669...	2018	3	0.133374079223...	Downward
4	2017	4	0.113869040942...	2018	4	0.133504429389...	Downward
5	2017	5	0.161601320626...	2018	5	0.132721963804...	Upward
6	2017	6	0.139349495118...	2018	6	0.117758693807...	Upward
7	2017	7	0.161455259909...	2018	7	0.122665140719...	Upward
8	2017	8	0.183808191377...	2018	8	0.117591330432...	Upward

Insights:

- Comparing 2017 and 2018, we could see for the first 4 months, 2017 had less purchase and 2018 had higher purchase.
- Similarly Comparing 2017 and 2018, we could see for the last 4 months, 2017 had high purchase and 2018 had lesser purchase.

Recommendation: N/A

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

```
/*Calculate the Total & Average value of order price for each state.*/

SELECT DISTINCT c.customer_state,
  SUM(payment_value) OVER(PARTITION BY c.customer_state) AS
`Total_Value_Based_On_State`,
  AVG(payment_value) OVER(PARTITION BY c.customer_state) AS
`Average_Value_Based_On_State`
FROM `lucid-totality-409609.retail_dataset.payments` AS `p`
JOIN `lucid-totality-409609.retail_dataset.orders` AS `o`
  ON o.order_id = p.order_id
JOIN `lucid-totality-409609.retail_dataset.customers` AS `c`
  ON c.customer_id = o.customer_id
#  ORDER BY `Total_Value_Based_On_State` DESC, `Average_Value_Based_On_State` DESC
```

Row	customer_state	Total_Value_Based_On_State	Average_Value_Based_On_State
1	BA	616645.82	170.8160166204986
2	AP	16262.8	232.3257142857143
3	MG	1872257.26	154.706433647331
4	RS	890898.54	157.18040578687368
5	SE	75246.25	208.43836565096953
6	RO	60866.2	233.20383141762451
7	RJ	2144379.69	158.5258882235529
8	AC	19680.62	234.29309523809525
9	RR	10064.62	218.79608695652175
10	GO	350092.31	165.76340435606062
11	AM	27966.93	181.60344155844155
12	MS	137534.84	186.86798913043478

```
/*Sorted Data based on Total Desc*/
```

Row	customer_state	Total_Value_Based_C	Average_Value_Base
1	SP	5998226.96	137.5046297739...
2	RJ	2144379.69	158.5258882235...
3	MG	1872257.26	154.7064336473...
4	RS	890898.54	157.1804057868...
5	PR	811156.38	154.1536259977...
6	SC	623086.43	165.9793367075...
7	BA	616645.82	170.8160166204...
8	DF	355141.08	161.1347912885...
9	GO	350092.31	165.7634043560...
10	ES	325967.55	154.7069530137...
11	PE	324850.44	187.9921527777...
12	CE	279464.03	199.9027396280...

Insights:

- The top 5 states contribute 73% of the total sales.
- The top 5 states are SP, RJ, MG, RS and PR.

Recommendation: N/A

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

```
SELECT DISTINCT c.customer_state,
  SUM(oi.freight_value) OVER(PARTITION BY c.customer_state) AS
`Total_Freight_Value_By_State`,
  AVG(oi.freight_value) OVER(PARTITION BY c.customer_state) AS
`Average_Freight_Value_By_State`
FROM lucid-totality-409609.retail_dataset.customers AS `c`
JOIN lucid-totality-409609.retail_dataset.orders AS `o`
  ON o.customer_id = c.customer_id
JOIN lucid-totality-409609.retail_dataset.order_items AS `oi`
  ON oi.order_id = o.order_id
```

Row	customer_state	Total_Freight_Value_By_State	Average_Freight_Value_By_State
1	AL	15914.59	35.843671171171174
2	MT	29715.43	28.166284360189572
3	GO	53114.98	22.766815259322762
4	DF	50625.5	21.041354945968411
5	BA	100156.68	26.363958936562252
6	ES	49764.6	22.058776595744682
7	PE	59449.659999999996	32.917862679955704
8	SP	718723.07	15.147275390419187
9	TO	11732.68	37.246603174603173
10	MS	19144.03	23.374884004884006
11	CE	48351.59	32.714201623815967
12	AM	5478.89	33.205393939393936
13	RS	135522.74	21.735804330392941
14	MA	31523.77	38.257002427184467

Q. 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_delivered_customer_date - order_estimated_delivery_date

```
SELECT *,
    DATETIME_DIFF (DATETIME(order_delivered_customer_date),
DATETIME(order_purchase_timestamp), DAY) AS `time_to_deliver`,
    DATETIME_DIFF (DATETIME(order_delivered_customer_date),
DATETIME(order_estimated_delivery_date), DAY) AS `diff_estimated_delivery`
FROM `lucid-totality-409609.retail_dataset.orders`
```

Row	order_id	customer_id	order_status	order_purchase_timestamp	order_approved_at	order_delivered_carrier_date	order_delivered_customer_date	order_estimated_delivery_date	time_to_deliver	diff_estimated_deliv
1	1950d777989f6a877539f5379...	1bccb206de9f0f25adc6871a1...	canceled	2018-02-19 19:48:52 UTC	2018-02-19 20:56:05 UTC	2018-02-20 19:57:13 UTC	2018-03-21 22:03:51 UTC	2018-03-09 00:00:00 UTC	30	12
2	2c45c33d2f9cb8ff8b1c86cc28...	de4caa97afa80c8eeac2f4c8d...	canceled	2016-10-09 15:39:56 UTC	2016-10-10 10:40:49 UTC	2016-10-14 10:40:50 UTC	2016-11-09 14:53:50 UTC	2016-12-08 00:00:00 UTC	31	-29
3	65d1e226dfaeb8cdc42f66542...	70fc57eeae292675927697fe0...	canceled	2016-10-03 21:01:41 UTC	2016-10-04 10:18:57 UTC	2016-10-25 12:14:28 UTC	2016-11-08 10:58:34 UTC	2016-11-25 00:00:00 UTC	36	-17
4	635c894d068ac37e6e03dc54e...	7a34a8e890765ad6f90db76d0...	delivered	2017-04-15 15:37:38 UTC	2017-04-15 15:45:14 UTC	2017-04-27 16:06:59 UTC	2017-05-16 14:49:55 UTC	2017-05-18 00:00:00 UTC	31	-2
5	3b97562c3aee8bdedcb5c2e45...	065d53860347d845788e041c...	delivered	2017-04-14 22:21:54 UTC	2017-04-15 22:30:19 UTC	2017-04-17 09:08:52 UTC	2017-05-17 10:52:15 UTC	2017-05-18 00:00:00 UTC	33	-1
6	68f47f50f04c4cb6774570cfde...	0378e1381c730d4504ebc07d...	delivered	2017-04-16 14:56:13 UTC	2017-04-16 15:05:14 UTC	2017-04-17 10:44:19 UTC	2017-05-16 09:07:47 UTC	2017-05-18 00:00:00 UTC	30	-2
7	276e9ec344d3bf029ff83a161c...	d39e520a99eb4cfc0d3ef2b6ff...	delivered	2017-04-08 21:20:24 UTC	2017-04-08 21:30:16 UTC	2017-04-25 10:53:00 UTC	2017-05-22 14:11:31 UTC	2017-05-18 00:00:00 UTC	44	4
8	54e1a3c2b97fb0809da548a59...	a0bc11375dd3d8bdd0e0bfcbc...	delivered	2017-04-11 19:49:45 UTC	2017-04-11 20:02:27 UTC	2017-04-12 14:47:39 UTC	2017-05-22 16:18:42 UTC	2017-05-18 00:00:00 UTC	41	4
9	fd04fa4105ee8045f6a0139ca5...	8fe0db7abbccaf2d788689e91...	delivered	2017-04-12 12:17:08 UTC	2017-04-13 12:22:08 UTC	2017-04-19 14:19:04 UTC	2017-05-19 13:44:52 UTC	2017-05-18 00:00:00 UTC	37	1

Insights:

- We can see there are some positive values for diff_estimated_delivery which tells that the delivery was made post the agreed time/day.
- We can see there are some negative values for diff_estimated_delivery which tells that the delivery was made within the agreed time/day.

Recommendation: N/A

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

```
WITH `Last5FreightValues` AS

(
  SELECT DISTINCT c.customer_state,
  SUM(oi.freight_value) OVER(PARTITION BY c.customer_state) AS
`Total_Freight_Value_By_State`,
  AVG(oi.freight_value) OVER(PARTITION BY c.customer_state) AS
`Average_Freight_Value_By_State`
FROM lucid-totality-409609.retail_dataset.customers AS `c`
JOIN lucid-totality-409609.retail_dataset.orders AS `o`
  ON o.customer_id = c.customer_id
JOIN lucid-totality-409609.retail_dataset.order_items AS `oi`
  ON oi.order_id = o.order_id
ORDER BY Average_Freight_Value_By_State ASC
LIMIT 5
),
```

Row	customer_state	Total_Freight_Value_By_State	Average_Freight_Value_By_State
1	SP	718723.07	15.147275390419187
2	PR	117851.68000000001	20.531651567944252
3	MG	270853.46	20.630166806306651
4	RJ	305589.31	20.960923931682558
5	DF	50625.5	21.041354945968411

```
WITH `Top5FreightValues` AS

(SELECT DISTINCT c.customer_state,
  SUM(oi.freight_value) OVER(PARTITION BY c.customer_state) AS
`Total_Freight_Value_By_State`,
  AVG(oi.freight_value) OVER(PARTITION BY c.customer_state) AS
`Average_Freight_Value_By_State`
FROM lucid-totality-409609.retail_dataset.customers AS `c`
JOIN lucid-totality-409609.retail_dataset.orders AS `o`
  ON o.customer_id = c.customer_id
JOIN lucid-totality-409609.retail_dataset.order_items AS `oi`
  ON oi.order_id = o.order_id
ORDER BY Average_Freight_Value_By_State DESC
LIMIT 5)
```

Row	customer_state	Total_Freight_Value	Average_Freight_Val
1	RR	2235.19	42.98442307692…
2	PB	25719.73	42.72380398671…
3	RO	11417.38	41.06971223021…
4	AC	3686.75	40.07336956521…
5	PI	21218.2	39.14797047970…

Insights:

- We can see the top 5 states where the average freight value is high based on customer states are RR,PB,RO,AC,PI.
- We can see the last 5 states where the average freight value is high based on customer states are DF,RJ,MG.PR,SP.

Recommendation: N/A

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

```
/*States with highest delivery time*/

SELECT
  DISTINCT customer_state,
  AVG(DATETIME_DIFF (DATETIME(order_delivered_customer_date),
DATETIME(order_purchase_timestamp), DAY) ) OVER(PARTITION BY customer_state) AS
`AverageDeliveryTime`
FROM
  `lucid-totality-409609.retail_dataset.customers` AS `cust`
JOIN
  `lucid-totality-409609.retail_dataset.orders` AS `ord`
ON
  ord.customer_id = cust.customer_id
```

```
ORDER BY AverageDeliveryTime DESC
LIMIT 5
```

Row	customer_state	AverageDeliveryTime
1	RR	29.341463414634145
2	AP	27.17910447761194
3	AM	26.358620689655172
4	AL	24.501259445843829
5	PA	23.725158562367866

```
SELECT
  DISTINCT customer_state,
  AVG(DATETIME_DIFF (DATETIME(order_delivered_customer_date),
DATETIME(order_purchase_timestamp), DAY) ) OVER(PARTITION BY customer_state) AS
`AverageDeliveryTime`
FROM
  `lucid-totality-409609.retail_dataset.customers` AS `cust`
JOIN
  `lucid-totality-409609.retail_dataset.orders` AS `ord`
ON
  ord.customer_id = cust.customer_id
ORDER BY AverageDeliveryTime ASC
LIMIT 5

/*States with lowest delivery time*/
```

Row	customer_state	AverageDeliveryTime
1	SP	8.7005309297444136
2	PR	11.938045906967297
3	MG	11.946543372963452
4	DF	12.899038461538462
5	SC	14.907527488018044

Insights:

- The 5 states were the delivery time is very high are RR,AP,AM,AL,PA.
- RR state customers get their orders very delayed.
- The 5 states were the delivery time is very low are SC,DF,MG,PR, SP.
- SP state customers get their orders delivered quickly compared to the other states.

Recommendation: N/A

Q. Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.

```
  /*Find out the top 5 states where the order delivery is really fast as compared
to the estimated date of delivery.*/
SELECT
  DISTINCT cust.customer_state,
```

```
  AVG(DATETIME_DIFF (DATETIME(order_delivered_customer_date),
DATETIME(order_estimated_delivery_date), DAY)) OVER(PARTITION BY
cust.customer_state) AS `diff_estimated_delivery_average`
FROM
  `lucid-totality-409609.retail_dataset.customers` AS `cust`
JOIN
  `lucid-totality-409609.retail_dataset.orders` AS `ord`
ON
  ord.customer_id = cust.customer_id
WHERE
  ord.order_delivered_customer_date < ord.order_estimated_delivery_date
ORDER BY diff_estimated_delivery_average DESC
LIMIT 5
```

Row	customer_state	diff_estimated_delivery_average
1	SP	-12.168153668521045
2	ES	-13.339805825242719
3	MS	-13.411290322580646
4	DF	-13.416968442834971
5	SC	-13.507966260543579

Insights:

- The 5 states were the average delivery time is very low are SC,DF,MG,PR, SP.
- SP state customers get their orders delivered quickly compared to the other states.

Recommendation: N/A

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

```
/*Find the month on month no. of orders placed using different payment types.*/


SELECT T1.* EXCEPT(NumberOfOrdersBasedOnMonthsandPaymentType),
    T1.NumberOfOrdersBasedOnMonthsandPaymentType -
IF(LAG(T1.NumberOfOrdersBasedOnMonthsandPaymentType) OVER(PARTITION BY
T1.payment_type ORDER BY T1.payment_type,/*T1.Year,*/ T1.Month ) IS NULL, 0,
LAG(T1.NumberOfOrdersBasedOnMonthsandPaymentType) OVER(PARTITION BY T1.payment_type
ORDER BY T1.payment_type,/*T1.Year,*/ T1.Month )) AS
`CountOfOrdersSeparatedByPaymentTypeMonth`

FROM
(
SELECT
  DISTINCT pay.payment_type,
/*  EXTRACT(YEAR
  FROM
    ord.order_purchase_timestamp) AS `Year`,*/
  EXTRACT(MONTH
  FROM
    ord.order_purchase_timestamp) AS `Month`,
  COUNT(ord.order_id) OVER(PARTITION BY pay.payment_type ORDER BY
```

```
  /*EXTRACT(YEAR
  FROM
    ord.order_purchase_timestamp),*/
  EXTRACT(MONTH
  FROM
    ord.order_purchase_timestamp)) AS `NumberOfOrdersBasedOnMonthsandPaymentType`
FROM
  `lucid-totality-409609.retail_dataset.payments` AS `pay`
JOIN
  `lucid-totality-409609.retail_dataset.orders` AS `ord`
ON
  ord.order_id = pay.order_id
JOIN
  `lucid-totality-409609.retail_dataset.customers` AS `cust`
ON
  ord.customer_id = cust.customer_id
ORDER BY
  payment_type ASC,
  #Year ASC,
  Month ASC
) AS `T1`

ORDER BY
  T1.payment_type ASC,
  #T1.Year ASC,
  T1.Month ASC
```

Row	payment_type	Month	CountOfOrdersSeparatedByPaymentTypeMonth
1	UPI	1	1715
2	UPI	2	1723
3	UPI	3	1942
4	UPI	4	1783
5	UPI	5	2035
6	UPI	6	1807
7	UPI	7	2074
8	UPI	8	2077
9	UPI	9	903
10	UPI	10	1056
11	UPI	11	1509

```
/*Find the month on month no. of orders placed using different payment types.
Additionally segregated by year*/

SELECT T1.* EXCEPT(NumberOfOrdersBasedOnYearMonthsandPaymentType),
    T1.NumberOfOrdersBasedOnYearMonthsandPaymentType -
IF(LAG(T1.NumberOfOrdersBasedOnYearMonthsandPaymentType) OVER(PARTITION BY
T1.payment_type ORDER BY T1.payment_type,T1.Year, T1.Month ) IS NULL, 0,
LAG(T1.NumberOfOrdersBasedOnYearMonthsandPaymentType) OVER(PARTITION BY
T1.payment_type ORDER BY T1.payment_type,T1.Year, T1.Month )) AS
`CountOfOrdersSeparatedByPaymentTypeYearMonth`

FROM
(
SELECT
  DISTINCT pay.payment_type,
```

```
  EXTRACT(YEAR
  FROM
    ord.order_purchase_timestamp) AS `Year`,
  EXTRACT(MONTH
  FROM
    ord.order_purchase_timestamp) AS `Month`,
  COUNT(ord.order_id) OVER(PARTITION BY pay.payment_type ORDER BY
  EXTRACT(YEAR
  FROM
    ord.order_purchase_timestamp),
  EXTRACT(MONTH
  FROM
    ord.order_purchase_timestamp)) AS
`NumberOfOrdersBasedOnYearMonthsandPaymentType`
FROM
  `lucid-totality-409609.retail_dataset.payments` AS `pay`
JOIN
  `lucid-totality-409609.retail_dataset.orders` AS `ord`
ON
  ord.order_id = pay.order_id
JOIN
  `lucid-totality-409609.retail_dataset.customers` AS `cust`
ON
  ord.customer_id = cust.customer_id
ORDER BY
  payment_type ASC,
  Year ASC,
  Month ASC
) AS `T1`

ORDER BY
  T1.payment_type ASC,
  T1.Year ASC,
  T1.Month ASC
```

Row	payment_type	Year	Month	CountOfOrdersSeparatedByPaymentTypeYearMonth
1	UPI	2016	10	63
2	UPI	2017	1	197
3	UPI	2017	2	398
4	UPI	2017	3	590
5	UPI	2017	4	496
6	UPI	2017	5	772
7	UPI	2017	6	707
8	UPI	2017	7	845
9	UPI	2017	8	938
10	UPI	2017	9	903

Results per page: 50 1 – 50 of 90

Insights: N/A
Recommendation: N/A

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

```
SELECT DISTINCT payment_type, COUNT(payment_sequential) OVER(PARTITION BY
payment_type) AS `Payment_by_installments_paid`
FROM `lucid-totality-409609.retail_dataset.payments`
```

Row	payment_type	Payment_by_installments_paid
1	credit_card	76795
2	not_defined	3
3	voucher	5775
4	debit_card	1529
5	UPI	19784

Insights:

- We could see the orders were mostly paid using credit card.
- Not defined payment type can be assumed as Paid by cash.

Recommendation: N/A
