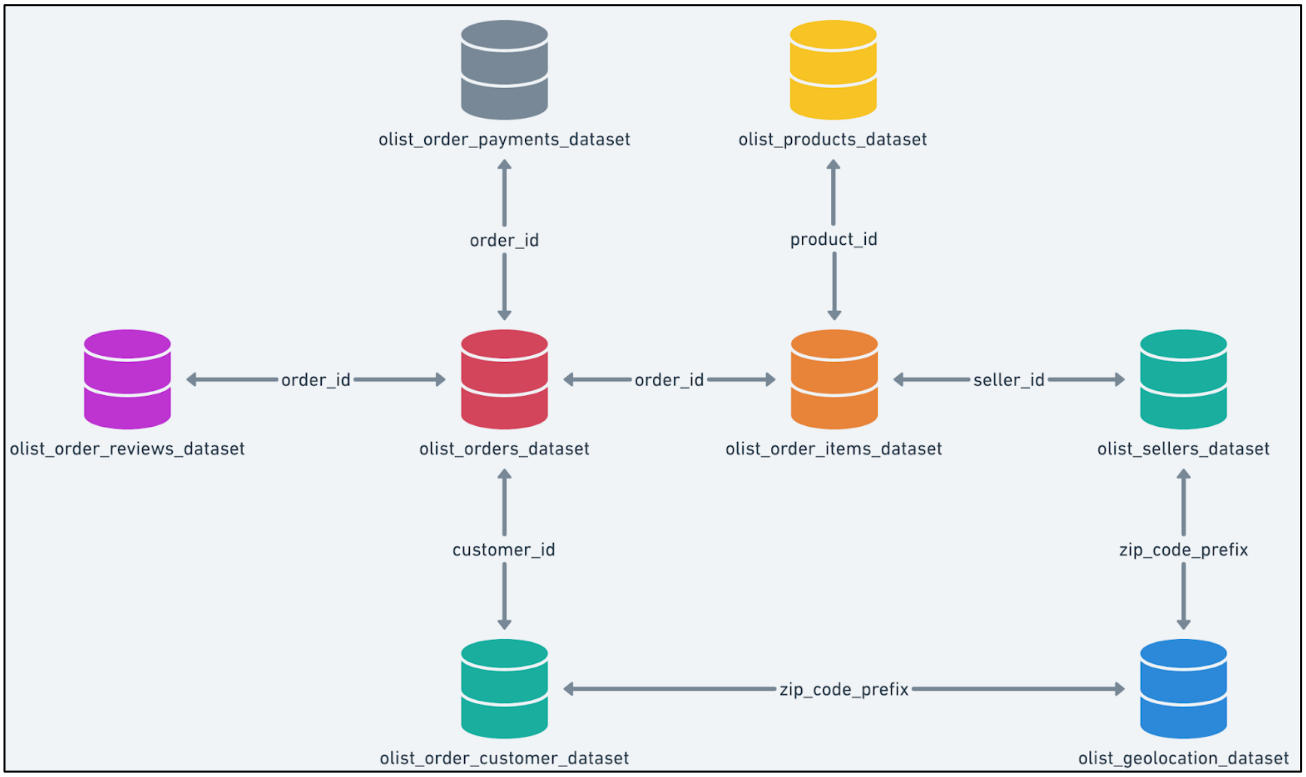
Business Case: Target SQL

Context

Target is one of the world's most recognized brands and one of America's leading retailers. 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 business case has information of 100k orders from 2016 to 2018 made at Target in Brazil. Its features allow viewing an order from multiple dimensions: from order status, price, payment and freight performance to customer location, product attributes and finally reviews written by customers.

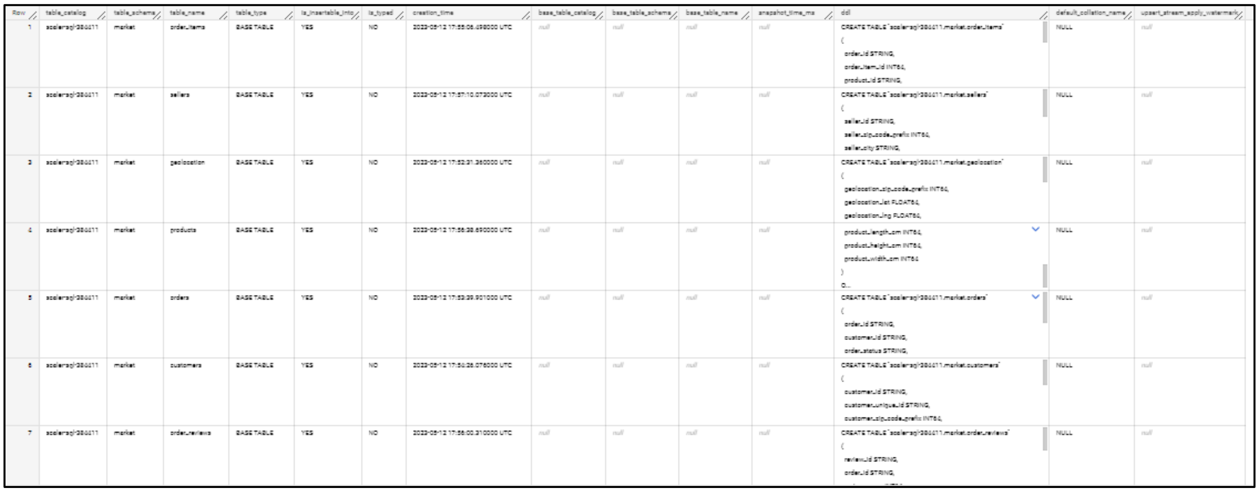
High level overview of relationship between datasets:



Let's start with some basic analysis on the dataset we have. We will study the data and draw some preliminary conclusions and recommendations from it to assist us reach additional conclusions.

a) By Importing the dataset and doing usual exploratory analysis steps like checking the structure & characteristics of the dataset we will get.

```
-- Returns metadata for tables in a single dataset.
SELECT * FROM `market`.INFORMATION_SCHEMA.TABLES;
```



Row	table_catalog	table_schema	table_name	table_type	is_insertable_into	is_typed	creation_time	base_table_catalog	base_table_schema	base_table_name	snapshot_time_ms	ddl	default_collation_name	upsert_stream_apply_watermark
1	scaler-sql-384411	market	order_items	BASE TABLE	YES	NO	2023-05-12 17:55:06.498000 UTC	null	null	null	null	CREATE TABLE `scaler-sql-384411.market.order_items` (order_id STRING, order_item_id INT64, product_id STRING,	NULL	null
2	scaler-sql-384411	market	sellers	BASE TABLE	YES	NO	2023-05-12 17:57:10.073000 UTC	null	null	null	null	CREATE TABLE `scaler-sql-384411.market.sellers` (seller_id STRING, seller_zip_code_prefix INT64, seller_city STRING,	NULL	null
3	scaler-sql-384411	market	geolocation	BASE TABLE	YES	NO	2023-05-12 17:52:31.360000 UTC	null	null	null	null	CREATE TABLE `scaler-sql-384411.market.geolocation` (geolocation_zip_code_prefix INT64, geolocation_lat FLOAT64, geolocation_lng FLOAT64,	NULL	null
4	scaler-sql-384411	market	products	BASE TABLE	YES	NO	2023-05-12 17:56:38.690000 UTC	null	null	null	null	product_length_cm INT64, product_height_cm INT64, product_width_cm INT64) O...	NULL	null
5	scaler-sql-384411	market	orders	BASE TABLE	YES	NO	2023-05-12 17:53:39.901000 UTC	null	null	null	null	CREATE TABLE `scaler-sql-384411.market.orders` (order_id STRING, customer_id STRING, order_status STRING,	NULL	null
6	scaler-sql-384411	market	customers	BASE TABLE	YES	NO	2023-05-12 17:54:26.076000 UTC	null	null	null	null	CREATE TABLE `scaler-sql-384411.market.customers` (customer_id STRING, customer_unique_id STRING, customer_zip_code_prefix INT64,	NULL	null
7	scaler-sql-384411	market	order_reviews	BASE TABLE	YES	NO	2023-05-12 17:56:00.310000 UTC	null	null	null	null	CREATE TABLE `scaler-sql-384411.market.order_reviews` (review_id STRING, order_id STRING,	NULL	null

```
-- Returns metadata for first record.
SELECT table_name, table_type, is_insertable_into, is_typed, creation_time,
ddl,default_collation_name
FROM `market`.INFORMATION_SCHEMA.TABLES
LIMIT 1;
```

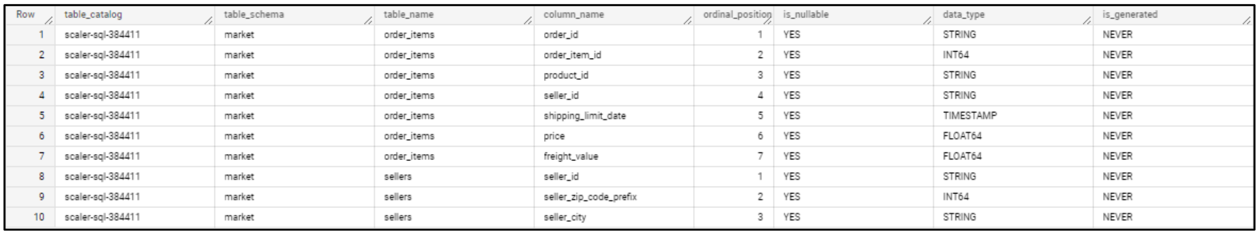
Row	1
table_name	order_items
table_type	BASE TABLE
is_insertable_into	YES
is_typed	NO
creation_time	2023-05-12 17:55:06.498000 UTC

Ddl	`CREATE TABLE `scaler-sql-384411.market.order_items`(`
	`order_id STRING,`
	`order_item_id INT64,`
	`product_id STRING,`
	`seller_id STRING,`
	`shipping_limit_date TIMESTAMP,`
	`price FLOAT64,`
	`freight_value FLOAT64)`
	`OPTIONS(expiration_timestamp=TIMESTAMP "2023-07-11T17:55:06.498Z");`

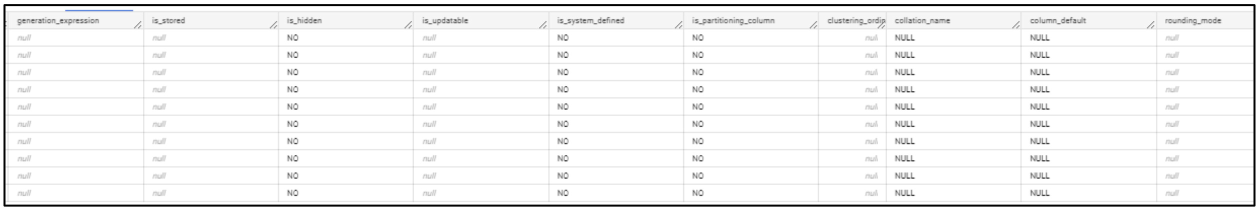
default_collation_name	NULL

b) Data type of columns in a table

```
-- Returns metadata for one row for each column (field) in a table.
SELECT * FROM `market`.INFORMATION_SCHEMA.COLUMNS limit 10;
```



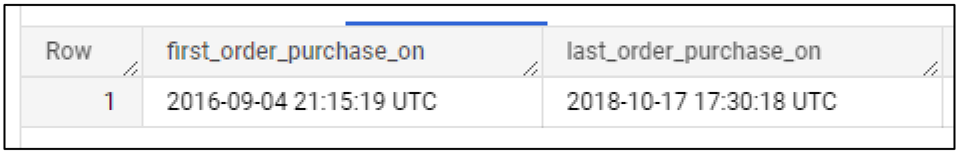
Row	table_catalog	table_schema	table_name	column_name	ordinal_position	is_nullable	data_type	is_generated
1	scaler-sql-384411	market	order_items	order_id	1	YES	STRING	NEVER
2	scaler-sql-384411	market	order_items	order_item_id	2	YES	INT64	NEVER
3	scaler-sql-384411	market	order_items	product_id	3	YES	STRING	NEVER
4	scaler-sql-384411	market	order_items	seller_id	4	YES	STRING	NEVER
5	scaler-sql-384411	market	order_items	shipping_limit_date	5	YES	TIMESTAMP	NEVER
6	scaler-sql-384411	market	order_items	price	6	YES	FLOAT64	NEVER
7	scaler-sql-384411	market	order_items	freight_value	7	YES	FLOAT64	NEVER
8	scaler-sql-384411	market	sellers	seller_id	1	YES	STRING	NEVER
9	scaler-sql-384411	market	sellers	seller_zip_code_prefix	2	YES	INT64	NEVER
10	scaler-sql-384411	market	sellers	seller_city	3	YES	STRING	NEVER



generation_expression	is_stored	is_hidden	is_updatable	is_system_defined	is_partitioning_column	clustering_ordip	collation_name	column_default	rounding_mode
null	null	NO	null	NO	NO	nul	NULL	NULL	null
null	null	NO	null	NO	NO	nul	NULL	NULL	null
null	null	NO	null	NO	NO	nul	NULL	NULL	null
null	null	NO	null	NO	NO	nul	NULL	NULL	null
null	null	NO	null	NO	NO	nul	NULL	NULL	null
null	null	NO	null	NO	NO	nul	NULL	NULL	null
null	null	NO	null	NO	NO	nul	NULL	NULL	null
null	null	NO	null	NO	NO	nul	NULL	NULL	null
null	null	NO	null	NO	NO	nul	NULL	NULL	null
null	null	NO	null	NO	NO	nul	NULL	NULL	null

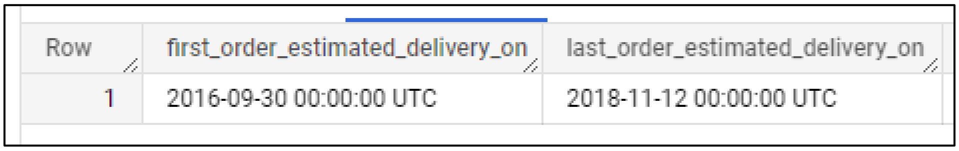
c) Time period for which the data is given

```
-- Time period for which the data is given for orders of purchase dates
SELECT
  MIN(order_purchase_timestamp) AS first_order_purchase_on,
  MAX(order_purchase_timestamp) AS last_order_purchase_on
FROM
  `market.orders`;
```



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

```
-- Time period for which the data is given for estimated delivery dates
SELECT
  MIN(order_estimated_delivery_date) AS first_order_estimated_delivery_on,
  MAX(order_estimated_delivery_date) AS last_order_estimated_delivery_on
FROM
  `market.orders`;
```



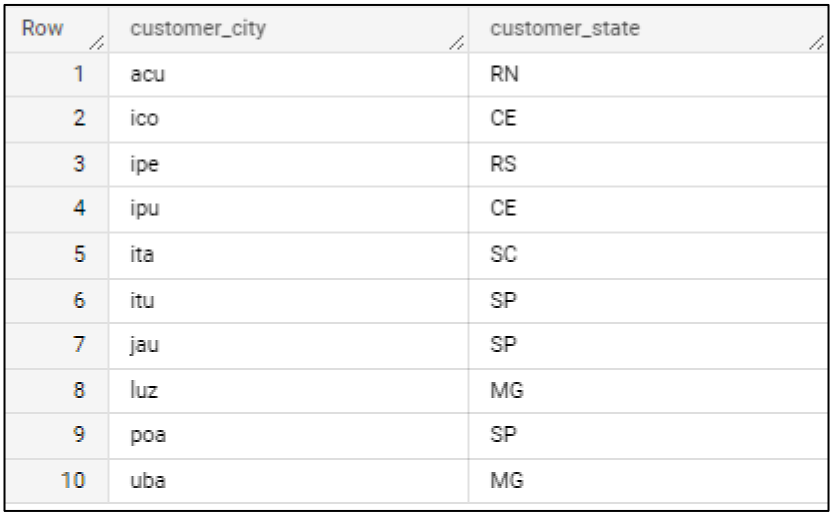
Row	first_order_estimated_delivery_on	last_order_estimated_delivery_on
1	2016-09-30 00:00:00 UTC	2018-11-12 00:00:00 UTC

Conclusion: The data is from September 2016 to November 2018.

d) Cities and States of customers ordered during the given period

Note : I have used the information which I concluded from Time period for which the data is given for orders of purchase dates and I have used order table only to get the city and sate as its already there in customer table.

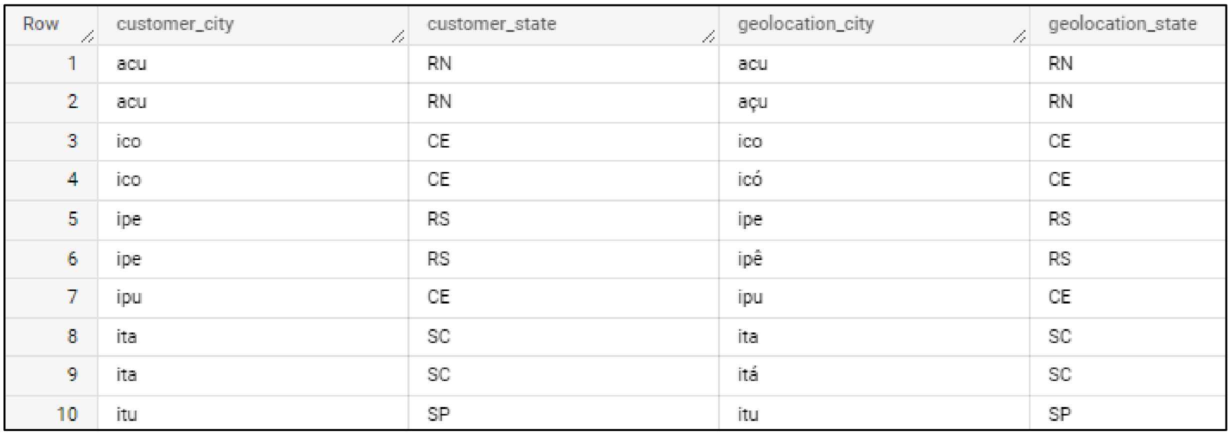
```
-- Cities and States of customers ordered during the given period of order purchased by the customers

SELECT
  c.customer_city,
  c.customer_state
FROM
  `market.customers` AS c
JOIN
  `market.orders` AS o
ON
  c.customer_id = o.customer_id
WHERE
  o.order_purchase_timestamp >= '2016-09-04' AND o.order_purchase_timestamp < '2018-
10-17'
GROUP BY
  c.customer_city,
  c.customer_state
LIMIT 10;
```



Row	customer_city	customer_state
1	acu	RN
2	ico	CE
3	ipe	RS
4	ipu	CE
5	ita	SC
6	itu	SP
7	jau	SP
8	luz	MG
9	poa	SP
10	uba	MG

We can get the same result by using geolocation table also by joining it on the above relation.

```
-- Cities and States of customers ordered during the given period of order purchasded by the customer
SELECT
  c.customer_city as customer_city,
  c.customer_state as customer_state,
  g.geolocation_city as geolocation_city,
  g.geolocation_state as geolocation_state
FROM
  `market.customers` AS c
JOIN
  `market.orders` AS o
ON
  c.customer_id = o.customer_id
JOIN
  `market.geolocation` AS g
ON
  c.customer_zip_code_prefix = g.geolocation_zip_code_prefix
WHERE
  o.order_purchase_timestamp >= '2016-09-04' AND o.order_purchase_timestamp < '2018-10-17'
GROUP BY
  c.customer_city,
  c.customer_state,
  g.geolocation_city,
  g.geolocation_state
LIMIT 10;
```



Row	customer_city	customer_state	geolocation_city	geolocation_state
1	acu	RN	acu	RN
2	acu	RN	açu	RN
3	ico	CE	ico	CE
4	ico	CE	icó	CE
5	ipe	RS	ipe	RS
6	ipe	RS	ipê	RS
7	ipu	CE	ipu	CE
8	ita	SC	ita	SC
9	ita	SC	itá	SC
10	itu	SP	itu	SP

In-depth Exploration:

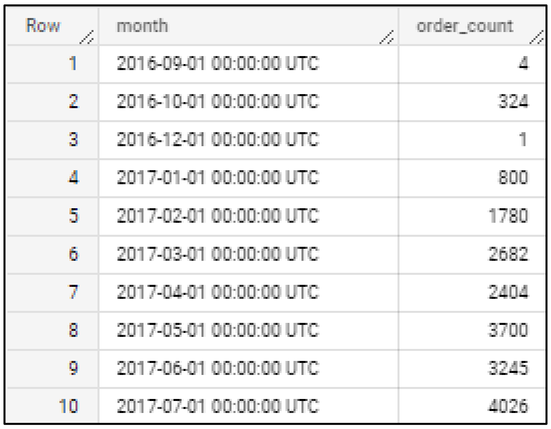
a) Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario? Can we see some seasonality with peaks at specific months?

If we on the trends on e commerce then we have to discuss on the orders as it will very helpful enlighten us to see some market on going trends. Let's try to extract some information with below relationships.

1. Monthly Order Volume

```
-- Monthly order volume trend
SELECT DATE_TRUNC(order_purchase_timestamp, MONTH) AS month,
       COUNT(*) AS order_count
FROM `market.orders`
GROUP BY month
ORDER BY month
LIMIT 10;
```

Conclusion: The sales increased month by month from 2016 until the middle of 2017, with a modest drop in the business in April 2017.

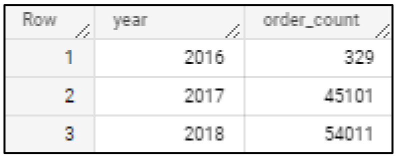


Row	month	order_count
1	2016-09-01 00:00:00 UTC	4
2	2016-10-01 00:00:00 UTC	324
3	2016-12-01 00:00:00 UTC	1
4	2017-01-01 00:00:00 UTC	800
5	2017-02-01 00:00:00 UTC	1780
6	2017-03-01 00:00:00 UTC	2682
7	2017-04-01 00:00:00 UTC	2404
8	2017-05-01 00:00:00 UTC	3700
9	2017-06-01 00:00:00 UTC	3245
10	2017-07-01 00:00:00 UTC	4026

2. Yearly Order Growth

```
-- Yearly order volume trend
SELECT EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
       COUNT(*) AS order_count
FROM `market.orders`
GROUP BY year
ORDER BY year;
```

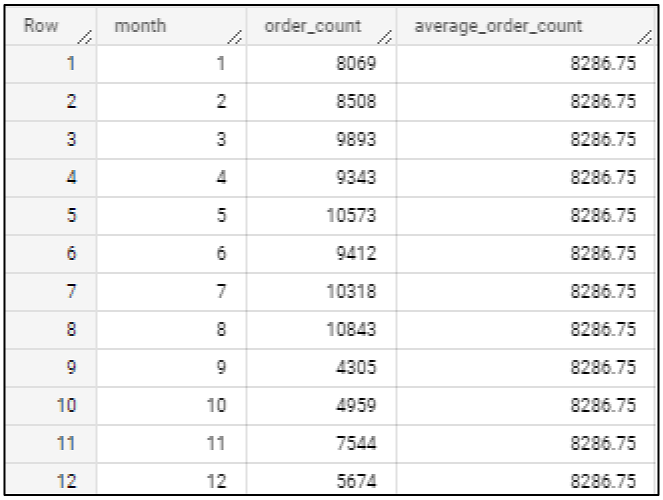
Conclusion: From 2016 to 2018, sales increased year after year.



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

3. Seasonality Analysis

```
WITH monthly_order_counts AS (
  SELECT EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
         COUNT(*) AS order_count
  FROM `market.orders`
  GROUP BY month
)
SELECT month, order_count, AVG(order_count) OVER () AS average_order_count
FROM monthly_order_counts
ORDER BY month;
```



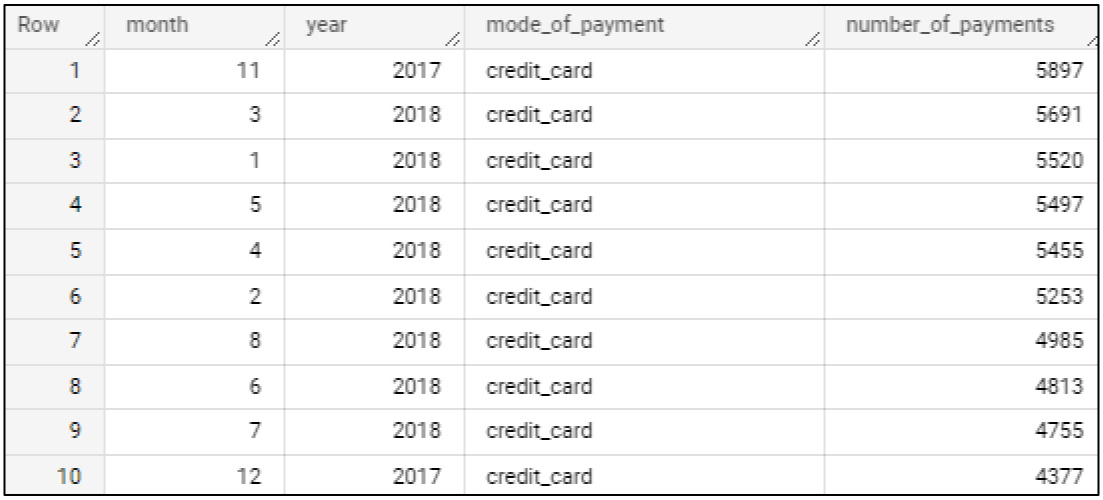
Row	month	order_count	average_order_count
1	1	8069	8286.75
2	2	8508	8286.75
3	3	9893	8286.75
4	4	9343	8286.75
5	5	10573	8286.75
6	6	9412	8286.75
7	7	10318	8286.75
8	8	10843	8286.75
9	9	4305	8286.75
10	10	4959	8286.75
11	11	7544	8286.75
12	12	5674	8286.75

Conclusion: Peak Months: Order numbers in months 3, 5, 7, 8, and 11 are greater than the average order count of 8286.75. These months are associated with increased e-commerce activity and larger order volumes. It predicts possible peak seasons or periods of strong demand for Brazilian online shopping.

Months with Low Activity: Months 9 and 10 have lower order counts than the norm. These months suggest lesser e-commerce activity and maybe lower order quantities. It indicates slowing sales or decreasing demand for online shopping in Brazil.

4. Relationship of orders and products

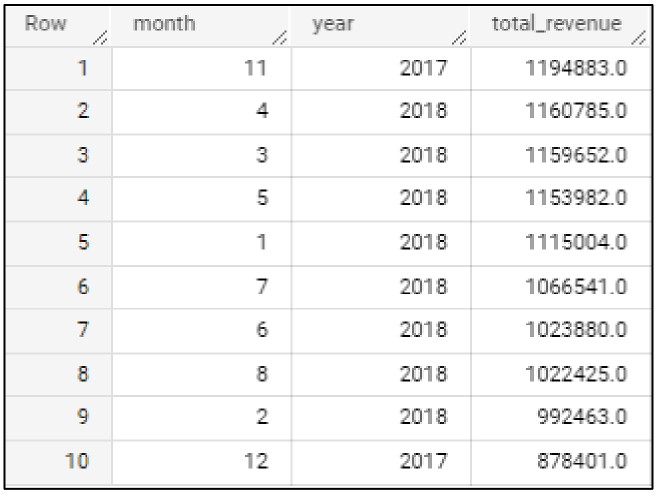
```
-- Top 10 number of payments and group by month with year
SELECT
  EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
  EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
  p.payment_type AS mode_of_payment,
  COUNT(*) AS number_of_payments
FROM
  `market.orders` AS o
JOIN
  `market.payments` AS p
ON
  o.order_id = p.order_id
GROUP BY
  month,
  year,
  mode_of_payment
ORDER BY
  number_of_payments DESC
LIMIT 10;
```



Row	month	year	mode_of_payment	number_of_payments
1	11	2017	credit_card	5897
2	3	2018	credit_card	5691
3	1	2018	credit_card	5520
4	5	2018	credit_card	5497
5	4	2018	credit_card	5455
6	2	2018	credit_card	5253
7	8	2018	credit_card	4985
8	6	2018	credit_card	4813
9	7	2018	credit_card	4755
10	12	2017	credit_card	4377

Conclusion: We can clearly see that the majority of shopping payments were made using a credit card, and the months of November 2017, March 2018, and January 2019 had the largest number of credit card transactions.

```
-- Top 10 number of generated revenues by the seller
SELECT
  EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
  EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
  ROUND(SUM(p.payment_value),0) AS total_revenue,
FROM
  `market.orders` AS o
JOIN
  `market.payments` AS p
ON
  o.order_id = p.order_id
GROUP BY
  month,
  year
ORDER BY
  total_revenue DESC
LIMIT 10;
```



Row	month	year	total_revenue
1	11	2017	1194883.0
2	4	2018	1160785.0
3	3	2018	1159652.0
4	5	2018	1153982.0
5	1	2018	1115004.0
6	7	2018	1066541.0
7	6	2018	1023880.0
8	8	2018	1022425.0
9	2	2018	992463.0
10	12	2017	878401.0

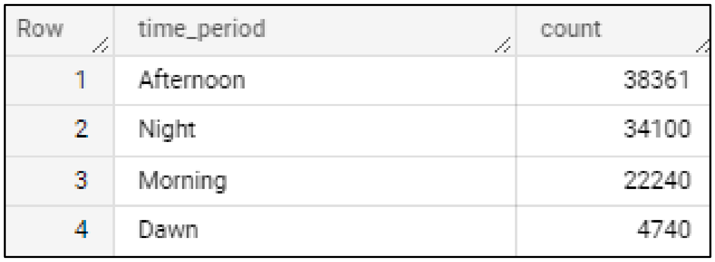
Conclusion: With this information, we can conclude that the seller generated the most revenue by selling its products in November 2017, which was the top month for the number of transactions, and that the following two months, April 2018 and March 2018, were among the top three for the seller's most revenue generated months.

Recommendations : We can also derive information from the payment table using the “payment_installments” and “payment_sequential” fields, such as when the largest number of payments occurred in installments and when the largest manner of payments was more than one, because we have other directly dependent tables on the order table. There are also some possibilities to find when the seller spent the most on shipping the products and when the greatest number of shipped products shrank in number of weights.

5. What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)?

```
-- count of records for each time period (Dawn, Morning, and Rest) based on the specified timestamp column

SELECT
  CASE
    WHEN EXTRACT(HOUR FROM o.order_purchase_timestamp) >= 0 AND EXTRACT(HOUR FROM o.order_purchase_timestamp) < 6 THEN 'Dawn'
    WHEN EXTRACT(HOUR FROM o.order_purchase_timestamp) >= 6 AND EXTRACT(HOUR FROM o.order_purchase_timestamp) < 12 THEN 'Morning'
    WHEN EXTRACT(HOUR FROM o.order_purchase_timestamp) >= 12 AND EXTRACT(HOUR FROM o.order_purchase_timestamp) < 18 THEN 'Afternoon'
    ELSE 'Night'
  END AS time_period,
  COUNT(*) AS count
FROM
  `market.orders` AS o
GROUP BY
  time_period
ORDER BY
  count DESC;
```

Conclusion: With this information, we can conclude that the Brazilian customers tend to buy mostly at Afternoon time followed by Night and very least in Dawn time.

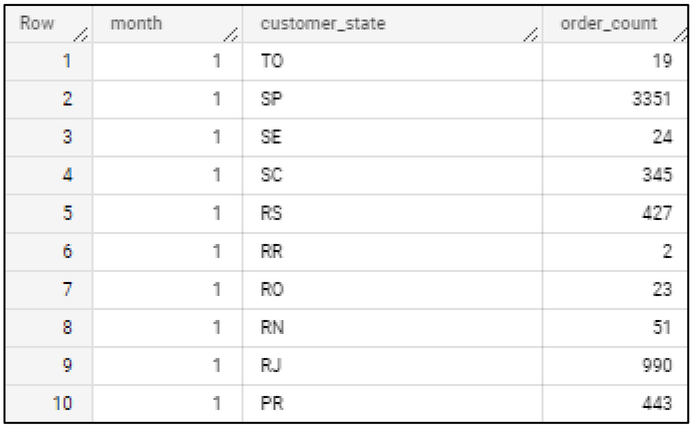


Row	time_period	count
1	Afternoon	38361
2	Night	34100
3	Morning	22240
4	Dawn	4740

Evolution of E-commerce orders in the Brazil region:

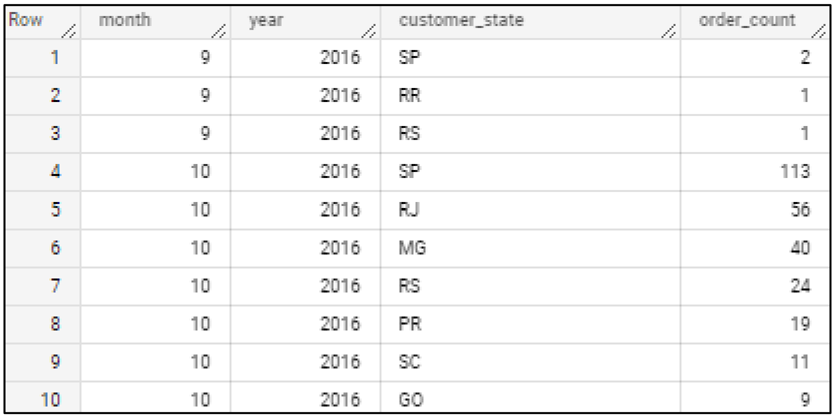
a) Get month on month orders by states

```
-- Month on Month number of orders by induvial states by merging every year together
SELECT
  EXTRACT(MONTH FROM
o.order_purchase_timestamp) AS month,
  c.customer_state,
  COUNT(*) AS order_count
FROM
  `market.orders` AS o
JOIN
  `market.customers` AS c
ON
  o.customer_id = c.customer_id
GROUP BY
  month, customer_state
ORDER BY
  month ASC,customer_state DESC
LIMIT 10;
```



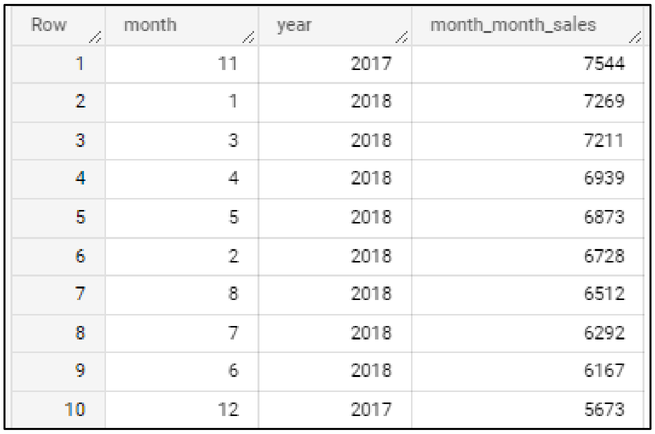
Row	month	customer_state	order_count
1	1	TO	19
2	1	SP	3351
3	1	SE	24
4	1	SC	345
5	1	RS	427
6	1	RR	2
7	1	RO	23
8	1	RN	51
9	1	RJ	990
10	1	PR	443

```
--------------------------------------------------------------------------------------------
-- Month on Month number of orders by induvial states without merging every year together
SELECT
  EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
  EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
  c.customer_state,
  COUNT(*) AS order_count
FROM
  `market.orders` AS o
JOIN
  `market.customers` AS c
ON
  o.customer_id =
c.customer_id
GROUP BY
  year,month, customer_state
ORDER BY
  year,month,order_count DESC
LIMIT 10;
```



Row	month	year	customer_state	order_count
1	9	2016	SP	2
2	9	2016	RR	1
3	9	2016	RS	1
4	10	2016	SP	113
5	10	2016	RJ	56
6	10	2016	MG	40
7	10	2016	RS	24
8	10	2016	PR	19
9	10	2016	SC	11
10	10	2016	GO	9

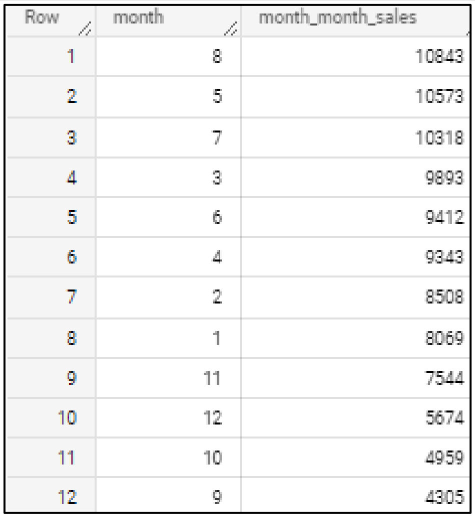
```
-- Month on Month number of orders by all state without merging every year into one
SELECT
  EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
  EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
  COUNT(*) AS month_month_sales
FROM
  `market.orders` AS o
JOIN
  `market.customers` AS c
ON
  o.customer_id = c.customer_id
GROUP BY
  month,
  year
ORDER BY
  month_month_sales DESC
LIMIT 10;
```



Row	month	year	month_month_sales
1	11	2017	7544
2	1	2018	7269
3	3	2018	7211
4	4	2018	6939
5	5	2018	6873
6	2	2018	6728
7	8	2018	6512
8	7	2018	6292
9	6	2018	6167
10	12	2017	5673

Conclusion: Highest sales occurred in month of Nov 2017 followed by January 2018.

```
-- Month on Month number of orders by all state merging every year into one
SELECT
  EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
  COUNT(*) AS month_month_sales
FROM
  `market.orders` AS o
JOIN
  `market.customers` AS c
ON
  o.customer_id = c.customer_id
GROUP BY
  month
ORDER BY
  month_month_sales DESC
```



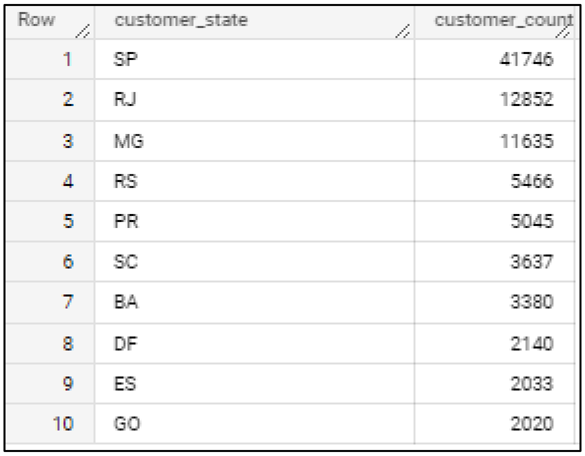
Row	month	month_month_sales
1	8	10843
2	5	10573
3	7	10318
4	3	9893
5	6	9412
6	4	9343
7	2	8508
8	1	8069
9	11	7544
10	12	5674
11	10	4959
12	9	4305

Conclusion: Highest sales occurred in month of August followed by May and July respectively.

b) Distribution of customers across the states in Brazil

```
-- Distribution of customers across the states in Brazil
SELECT
  customer_state,
  COUNT(*) AS customer_count
FROM
  `market.customers`
GROUP BY
  customer_state
ORDER BY
  customer_count DESC
LIMIT 10;
```

Conclusion: Largest populated state is SP followed by RJ and MG.

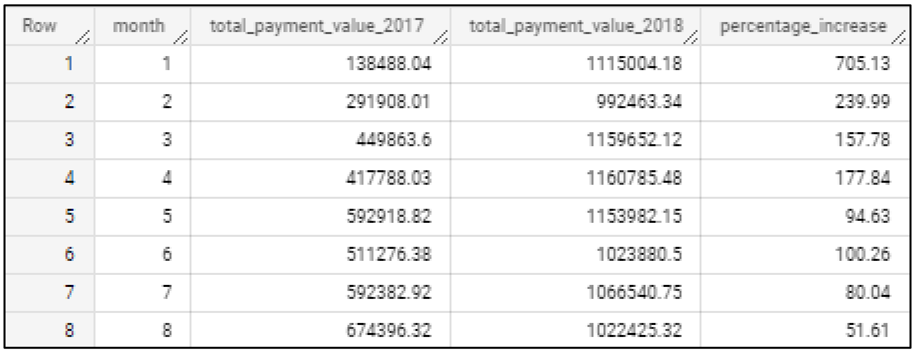


Row	customer_state	customer_count
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637
7	BA	3380
8	DF	2140
9	ES	2033
10	GO	2020

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

c) Get % increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only) - You can use “payment_value” column in payments table

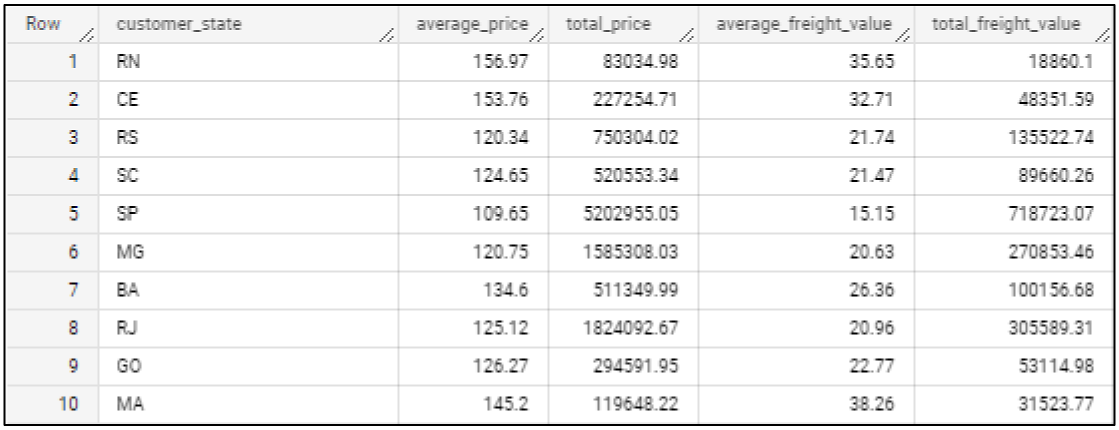
```
WITH orders_2017 AS (
  SELECT
    EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
    SUM(p.payment_value) AS total_payment_value_2017
  FROM
    `market.orders` AS o
  JOIN
    `market.payments` AS p
  ON
    o.order_id = p.order_id
  WHERE
    EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2017
    AND EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8
  GROUP BY
    month
),
orders_2018 AS (
  SELECT
    EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
    SUM(p.payment_value) AS total_payment_value_2018
  FROM
    `market.orders` AS o
  JOIN
    `market.payments` AS p
  ON
    o.order_id = p.order_id
  WHERE
    EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2018
    AND EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8
  GROUP BY
    month
)
SELECT
  orders_2018.month,
  ROUND((orders_2017.total_payment_value_2017),2) AS total_payment_value_2017,
  ROUND((orders_2018.total_payment_value_2018),2) AS total_payment_value_2018,
  ROUND(((orders_2018.total_payment_value_2018 - orders_2017.total_payment_value_2017) /
orders_2017.total_payment_value_2017 * 100 ),2)AS percentage_increase
FROM
  orders_2017
JOIN
  orders_2018
ON
  orders_2017.month = orders_2018.month
ORDER BY
  orders_2017.month;
```



Row	month	total_payment_value_2017	total_payment_value_2018	percentage_increase
1	1	138488.04	1115004.18	705.13
2	2	291908.01	992463.34	239.99
3	3	449863.6	1159652.12	157.78
4	4	417788.03	1160785.48	177.84
5	5	592918.82	1153982.15	94.63
6	6	511276.38	1023880.5	100.26
7	7	592382.92	1066540.75	80.04
8	8	674396.32	1022425.32	51.61

d) Mean & Sum of price and freight value by customer state

```
SELECT
  customer_state,
  ROUND(AVG(price),2) AS average_price,
  ROUND(SUM(price),2) AS total_price,
  ROUND(AVG(freight_value),2) AS average_freight_value,
  ROUND(SUM(freight_value),2) AS total_freight_value
FROM
  `market.orders` AS o
JOIN
  `market.customers` AS c
ON
  o.customer_id = c.customer_id
JOIN
  `market.order_items` AS oi
ON
  o.order_id = oi.order_id
GROUP BY
  customer_state
LIMIT 10;
```

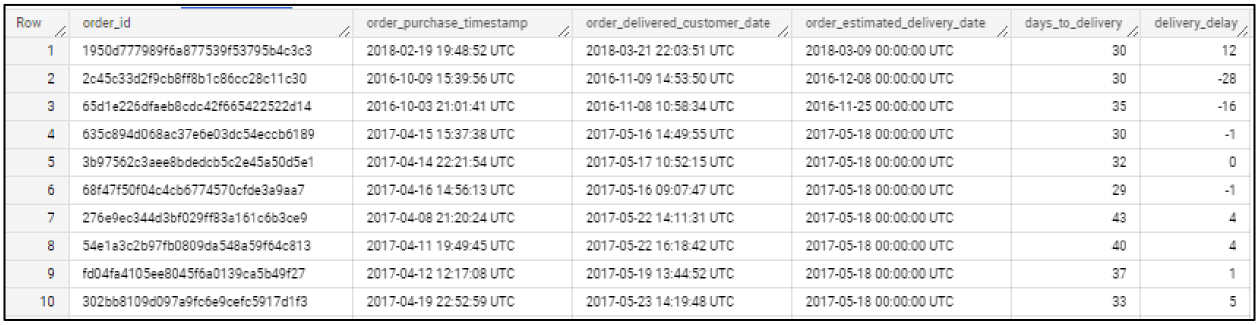


Row	customer_state	average_price	total_price	average_freight_value	total_freight_value
1	RN	156.97	83034.98	35.65	18860.1
2	CE	153.76	227254.71	32.71	48351.59
3	RS	120.34	750304.02	21.74	135522.74
4	SC	124.65	520553.34	21.47	89660.26
5	SP	109.65	5202955.05	15.15	718723.07
6	MG	120.75	1585308.03	20.63	270853.46
7	BA	134.6	511349.99	26.36	100156.68
8	RJ	125.12	1824092.67	20.96	305589.31
9	GO	126.27	294591.95	22.77	53114.98
10	MA	145.2	119648.22	38.26	31523.77

Analysis on sales, freight and delivery time

- **Calculate days between purchasing, delivering and estimated delivery**

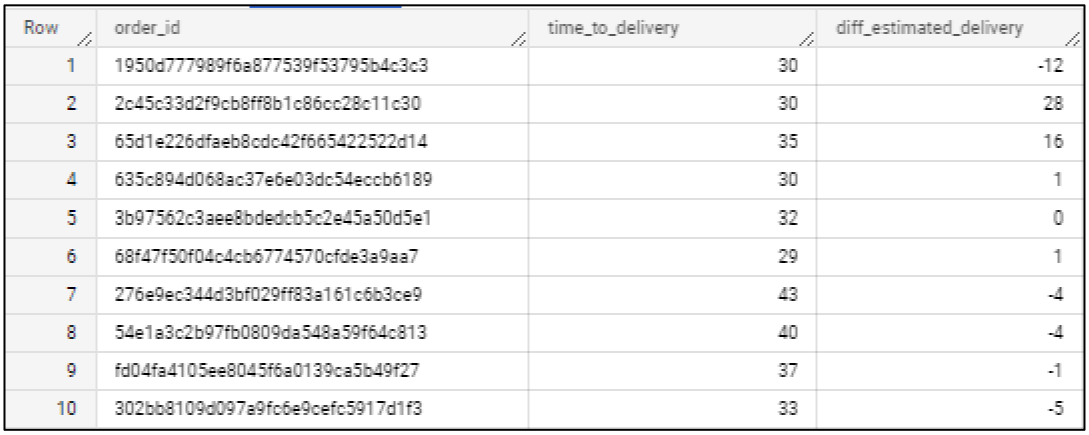
```
SELECT
  order_id,
  order_purchase_timestamp,
  order_delivered_customer_date,
  order_estimated_delivery_date,
  DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) AS days_to_delivery,
  DATE_DIFF(order_delivered_customer_date, order_estimated_delivery_date, DAY) AS
delivery_delay
FROM
  `market.orders`
LIMIT 10;
```



Row	order_id	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	days_to_delivery	delivery_delay
1	1950d777989f6a877539f53795b4c3c3	2018-02-19 19:48:52 UTC	2018-03-21 22:03:51 UTC	2018-03-09 00:00:00 UTC	30	12
2	2c45c33d2f9cb8ff8b1c86cc28c11c30	2016-10-09 15:39:56 UTC	2016-11-09 14:53:50 UTC	2016-12-08 00:00:00 UTC	30	-28
3	65d1e226dfaeb8cdc42f665422522d14	2016-10-03 21:01:41 UTC	2016-11-08 10:58:34 UTC	2016-11-25 00:00:00 UTC	35	-16
4	635c894d068ac37e6e03dc54eccb6189	2017-04-15 15:37:38 UTC	2017-05-16 14:49:55 UTC	2017-05-18 00:00:00 UTC	30	-1
5	3b97562c3aee8bdedcb5c2e45a50d5e1	2017-04-14 22:21:54 UTC	2017-05-17 10:52:15 UTC	2017-05-18 00:00:00 UTC	32	0
6	68f47f50f04c4cb6774570cfde3a9aa7	2017-04-16 14:56:13 UTC	2017-05-16 09:07:47 UTC	2017-05-18 00:00:00 UTC	29	-1
7	276e9ec344d3bf029ff83a161c6b3ce9	2017-04-08 21:20:24 UTC	2017-05-22 14:11:31 UTC	2017-05-18 00:00:00 UTC	43	4
8	54e1a3c2b97fb0809da548a59f64c813	2017-04-11 19:49:45 UTC	2017-05-22 16:18:42 UTC	2017-05-18 00:00:00 UTC	40	4
9	fd04fa4105ee8045f6a0139ca5b49f27	2017-04-12 12:17:08 UTC	2017-05-19 13:44:52 UTC	2017-05-18 00:00:00 UTC	37	1
10	302bb8109d097a9fc6e9cefc5917d1f3	2017-04-19 22:52:59 UTC	2017-05-23 14:19:48 UTC	2017-05-18 00:00:00 UTC	33	5

- **Find time_to_delivery & diff_estimated_delivery. Formula for the same given below:**
  1. **time_to_delivery = order_delivered_customer_date-order_purchase_timestamp**
  2. **diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date**

```
SELECT
  order_id,
  TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) AS
time_to_delivery,
  TIMESTAMP_DIFF(order_estimated_delivery_date, order_delivered_customer_date, DAY) AS
diff_estimated_delivery
FROM
  `market.orders`
LIMIT 10;
```

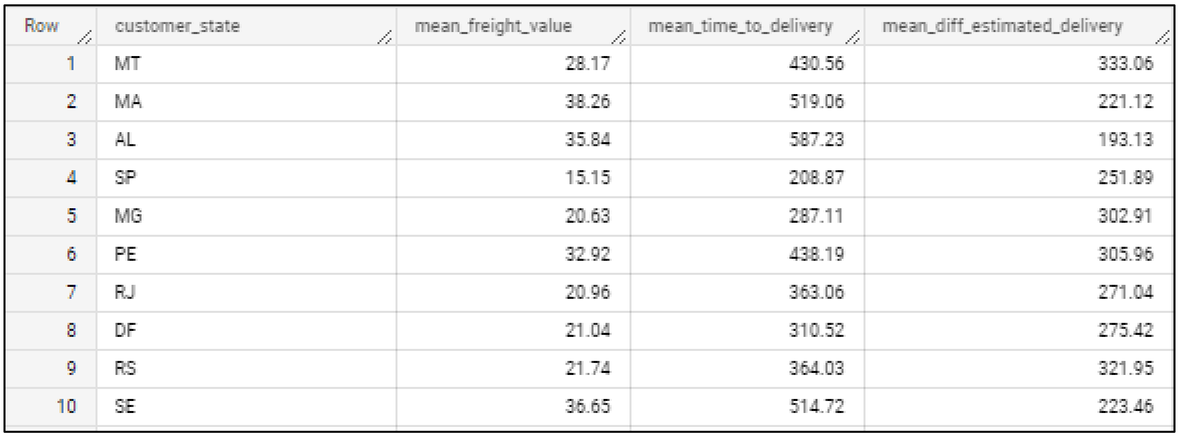


Row	order_id	time_to_delivery	diff_estimated_delivery
1	1950d777989f6a877539f53795b4c3c3	30	-12
2	2c45c33d2f9cb8ff8b1c86cc28c11c30	30	28
3	65d1e226dfaeb8cdc42f665422522d14	35	16
4	635c894d068ac37e6e03dc54eccb6189	30	1
5	3b97562c3aee8bdedcb5c2e45a50d5e1	32	0
6	68f47f50f04c4cb6774570cfde3a9aa7	29	1
7	276e9ec344d3bf029ff83a161c6b3ce9	43	-4
8	54e1a3c2b97fb0809da548a59f64c813	40	-4
9	fd04fa4105ee8045f6a0139ca5b49f27	37	-1
10	302bb8109d097a9fc6e9cefc5917d1f3	33	-5

If some diff_estimated_delivery values are coming out as negative, it means that the order_delivered_customer_date is later than the order_estimated_delivery_date. This can happen if the order was delivered earlier than expected.

- **Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery**

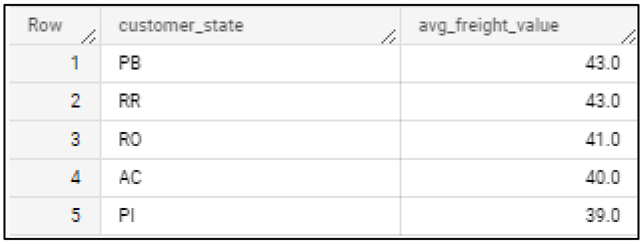
```
SELECT
  c.customer_state,
  ROUND(AVG(oi.freight_value),2) AS mean_freight_value,
  ROUND(AVG(TIMESTAMP_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp,
HOUR)),2) AS mean_time_to_delivery,
  ROUND(AVG(TIMESTAMP_DIFF(o.order_estimated_delivery_date, o.order_delivered_customer_date,
HOUR)),2) AS mean_diff_estimated_delivery
FROM
  `market.customers` AS c
JOIN
  `market.orders` AS o
ON
  c.customer_id = o.customer_id
JOIN
  `market.order_items` AS oi
ON
  o.order_id = oi.order_id
GROUP BY
  c.customer_state
LIMIT 10;
```



Row	customer_state	mean_freight_value	mean_time_to_delivery	mean_diff_estimated_delivery
1	MT	28.17	430.56	333.06
2	MA	38.26	519.06	221.12
3	AL	35.84	587.23	193.13
4	SP	15.15	208.87	251.89
5	MG	20.63	287.11	302.91
6	PE	32.92	438.19	305.96
7	RJ	20.96	363.06	271.04
8	DF	21.04	310.52	275.42
9	RS	21.74	364.03	321.95
10	SE	36.65	514.72	223.46

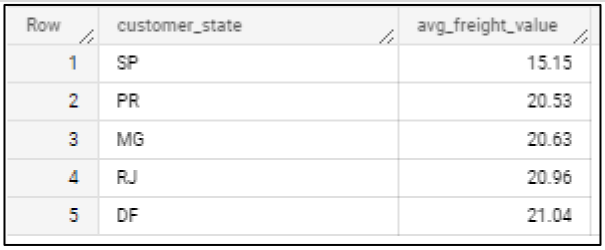
- **Sort the data to get the following:**
- **Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5**

```
--Top 5 states with highest average freight value
SELECT customer_state, ROUND(AVG(freight_value)) as avg_freight_value
FROM `market.customers` c
JOIN `market.orders` o
ON c.customer_id = o.customer_id
JOIN `market.order_items` oi
ON o.order_id = oi.order_id
GROUP BY customer_state
ORDER BY avg_freight_value DESC
LIMIT 5
```



Row	customer_state	avg_freight_value
1	PB	43.0
2	RR	43.0
3	RO	41.0
4	AC	40.0
5	PI	39.0

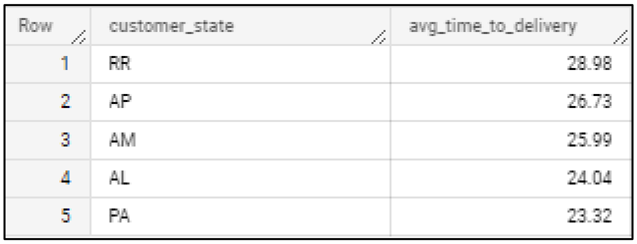
```
--Top 5 states with lowest average freight value
SELECT customer_state, AVG(freight_value) as avg_freight_value
FROM `market.customers` c
JOIN `market.orders` o
ON c.customer_id = o.customer_id
JOIN `market.order_items` oi
ON o.order_id = oi.order_id
GROUP BY customer_state
ORDER BY avg_freight_value ASC
LIMIT 5
```



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

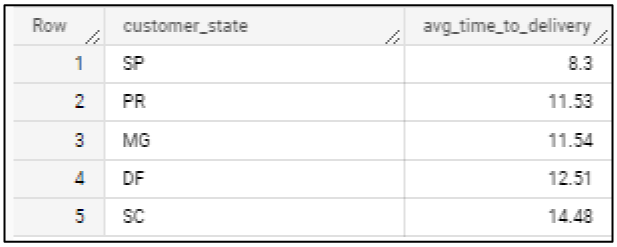
- **Top 5 states with highest/lowest average time to delivery**

```
SELECT customer_state, ROUND(AVG(time_to_delivery),2) AS avg_time_to_delivery
FROM (
  SELECT
    c.customer_state,
    TIMESTAMP_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, DAY) AS
time_to_delivery
  FROM `market.customers` AS c
  JOIN `market.orders` AS o
  ON c.customer_id = o.customer_id
) AS subquery
GROUP BY customer_state
ORDER BY avg_time_to_delivery DESC
LIMIT 5
```



Row	customer_state	avg_time_to_delivery
1	RR	28.98
2	AP	26.73
3	AM	25.99
4	AL	24.04
5	PA	23.32

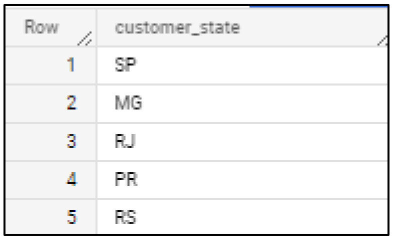
```
SELECT customer_state, ROUND(AVG(time_to_delivery),2) AS avg_time_to_delivery
FROM (
  SELECT
    c.customer_state,
    TIMESTAMP_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, DAY) AS
time_to_delivery
  FROM `market.customers` AS c
  JOIN `market.orders` AS o
  ON c.customer_id = o.customer_id
) AS subquery
GROUP BY customer_state
ORDER BY avg_time_to_delivery ASC
LIMIT 5
```



Row	customer_state	avg_time_to_delivery
1	SP	8.3
2	PR	11.53
3	MG	11.54
4	DF	12.51
5	SC	14.48

- **Top 5 states where delivery is really fast/ not so fast compared to estimated date**

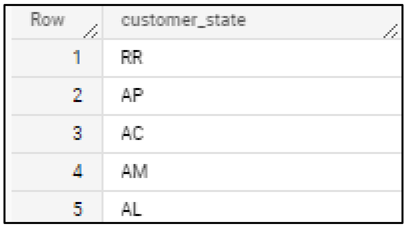
```
WITH delivery_duration AS (
  SELECT
    c.customer_state,
    TIMESTAMP_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, DAY) AS
time_to_delivery,
    TIMESTAMP_DIFF(o.order_estimated_delivery_date, o.order_delivered_customer_date, DAY) AS
diff_estimated_delivery
  FROM
    `market.customers` AS c
  JOIN
    `market.orders` AS o
  ON
    c.customer_id = o.customer_id
)
SELECT
  customer_state
FROM
  delivery_duration
WHERE
  time_to_delivery <= diff_estimated_delivery
GROUP BY
  customer_state
ORDER BY
  COUNT(*) DESC
LIMIT 5
```



Row	customer_state
1	SP
2	MG
3	RJ
4	PR
5	RS

```
WITH delivery_duration AS (
  SELECT
    c.customer_state,
    TIMESTAMP_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, DAY) AS
time_to_delivery,
    TIMESTAMP_DIFF(o.order_estimated_delivery_date, o.order_delivered_customer_date, DAY) AS
diff_estimated_delivery
  FROM
    `market.customers` AS c
  JOIN
    `market.orders` AS o
  ON
    c.customer_id = o.customer_id
)
SELECT
  customer_state
FROM
  delivery_duration
WHERE
  time_to_delivery <= diff_estimated_delivery
GROUP BY
  customer_state
```

```
ORDER BY
  COUNT(*) DESC
LIMIT 5
```

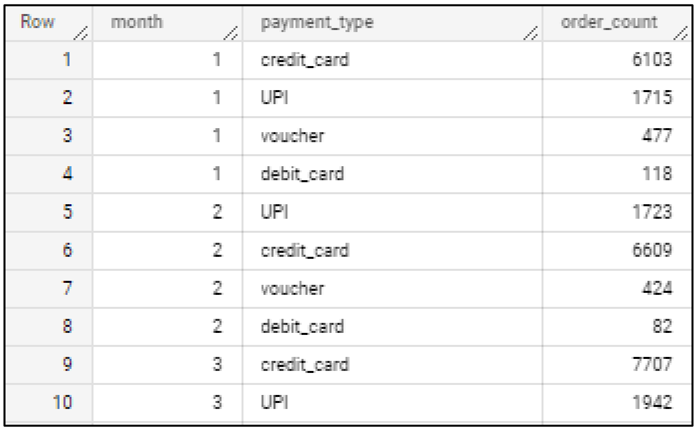


Row	customer_state
1	RR
2	AP
3	AC
4	AM
5	AL

Payment type analysis:

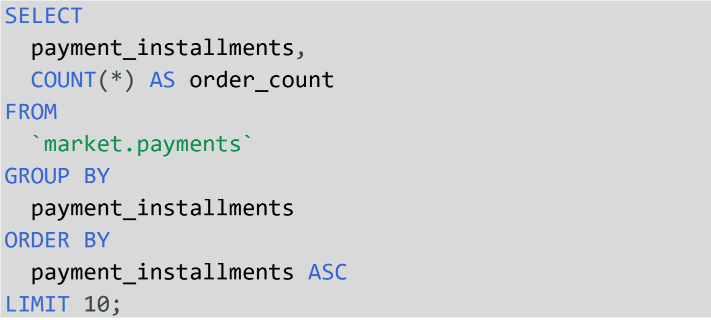
- **Month over Month count of orders for different payment types**

```
SELECT
  EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
  p.payment_type,
  COUNT(*) AS order_count
FROM
  `market.orders` AS o
JOIN
  `market.payments` AS p
ON
  o.order_id = p.order_id
GROUP BY
  month,
  payment_type
ORDER BY
  month ASC
LIMIT 10;
```

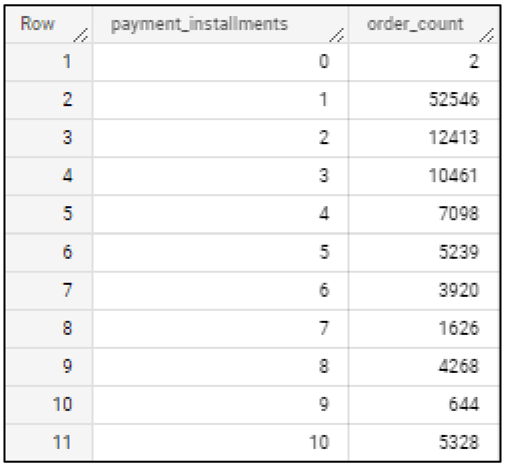


Row	month	payment_type	order_count
1	1	credit_card	6103
2	1	UPI	1715
3	1	voucher	477
4	1	debit_card	118
5	2	UPI	1723
6	2	credit_card	6609
7	2	voucher	424
8	2	debit_card	82
9	3	credit_card	7707
10	3	UPI	1942

- **Count of orders based on the no. of payment installments**



```
SELECT
  payment_installments,
  COUNT(*) AS order_count
FROM
  `market.payments`
GROUP BY
  payment_installments
ORDER BY
  payment_installments ASC
LIMIT 10;
```



Row	payment_installments	order_count
1	0	2
2	1	52546
3	2	12413
4	3	10461
5	4	7098
6	5	5239
7	6	3920
8	7	1626
9	8	4268
10	9	644
11	10	5328

Actionable Insights

1. **Sales Forecasting and Planning:** Use the orders table's month-to-month order data to anticipate future sales and manage inventories appropriately. Determine peak and low activity months to properly deploy resources and optimize stock levels.
2. **Marketing Campaigns:** Create targeted marketing campaigns using the customer data from the customers database. Customers may be segmented based on their location, purchasing history, and preferences to give personalized offers and promotions that are relevant to their requirements.
3. **Customer happiness Enhancement:** Analyze the order_reviews table's review ratings and feedback to discover areas for improvement in customer happiness. Priority should be given to answering consumer problems, increasing product quality, and improving the entire purchasing experience.
4. **Geographic Expansion Strategy:** Identify locations with high consumer concentration and low market saturation using geolocation data from the geolocation table. This data may be used to plan growth initiatives, such as building more stores or focusing marketing efforts on certain regions.
5. **Payment Optimization:** Analyze payment data from the payments table to learn about clients' preferred payment methods. Optimize the checkout process by including popular payment methods and providing a smooth payment experience, which may help lower cart abandonment rates.
6. **Product Performance Analysis:** Use information from the products table to assess the performance of various product categories and identify top-selling goods. This data may help with inventory management, product assortment planning, and promotional methods.

7. **Seller Management:** Assess seller performance using data from the sellers, order items, and order reviews tables. Identify top-performing merchants and cultivate strong connections with them while responding to any issues or concerns voiced by consumers about individual sellers.

8. **Seasonal Demand Management:** Use the orders table's order data to determine peak seasons and periods of high demand. Plan promotional events, personnel, and inventory management tactics to successfully fulfil client demand during these times.

Recommendations

1. **Sales Forecasting and Planning:** Use the year-over-year increase in sales to produce accurate sales predictions and thorough sales strategies. To capitalize on the expanding market demand, allocate resources, create sales objectives, and coordinate marketing tactics accordingly.

2. **Marketing Strategies:** Implement focused marketing efforts during the peak months highlighted in the study. Make marketing funds and resources available at these times to maximize consumer reach and engagement. To efficiently advertise items and generate sales, use multiple channels such as digital marketing, social media, and email marketing.

3. **Customer involvement and Retention:** Concentrate on increasing customer involvement and cultivating loyalty. To stimulate repeat purchases and improve client loyalty, implement customer retention programs, personalized offers, and incentive programs. To ensure client happiness and favorable word-of-mouth, provide great customer service.

4. **Inventory Management:** Optimize inventory management by using sales data and trends. To avoid stockouts or surplus inventory, analyze the best-selling goods and manage inventory levels appropriately. Implement effective supply chain and logistics management to guarantee that products are available and delivered on time.

5. **Geographic Expansion:** Consider extending operations to areas with significant client demand, such as the states with a big customer base. Conduct market research to find untapped prospects, form connections with local suppliers, and tailor marketing techniques to the unique requirements and tastes of those clients.

6. **Payment and Checkout Optimization:** Continuously analyze payment options and optimize the checkout process to give clients with a seamless and secure payment experience. To reduce cart abandonment and enhance conversion rates, consider including popular payment alternatives and ensuring a user-friendly interface.

7. **Seller Collaboration and Performance:** Strengthen connections with sellers by giving them with the tools and resources they need to succeed. Implement methods to track seller ratings, feedback, and delivery timeframes in order to maintain a high level of service and consumer satisfaction.

8. Analyze the competition and remain current on market trends, pricing tactics, and product offers. Determine unique selling features and value propositions that distinguish the company's products and services from rivals' offerings.

9. Establish a culture of continual improvement by analyzing sales data, customer feedback, and market trends on a regular basis. To stay ahead in the volatile e-commerce business, adapt strategy, optimize procedures, and innovate.