1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset
    1. Get number of rows in the data
    2. Number of null or missing values in a column
    3. Data type of columns in a table
    4. Get the time period for which the data is given
    5. Number of cities in our dataset
    6. Number of states in our dataset

Solution:

1. Number of rows in the data:
    a. Customers.csv :
        i. Total number of rows: 99,441
        ii. Total number of unique rows (customer_id): 99,441
    b. Geolocation.csv:
        i. Total number of rows: 1,000,163
        ii. Total number of unique rows (`geolocation_zip_code_prefix`): 19,015

There are several rows for one geolocation_zip_code_prefix:

Row	geolocation_zip_code_prefix	count(distinct city)
1	49010	92
2	49047	45
3	49030	105
4	49048	62
5	49050	60

Obs: one geolocation_zip_code_prefix has several longitudes and latitudes.

```
select geolocation_zip_code_prefix
  , geolocation_city
  , *
from `sqltest-353804.ecommerce.geolocation`
where geolocation_zip_code_prefix = 49010
limit 100
```

Row	geolocation_zip_code_prefix	geolocation_city	geolocation_zip_code_prefix_1	geolocation_lat	geolocation_lng	geolocation_city_1	geolocation_state
1	49010	aracaju	49010	-10.910514518754546	-37.052400776992329	aracaju	SE
2	49010	aracaju	49010	-10.917257467684108	-37.05154183236192	aracaju	SE
3	49010	aracaju	49010	-10.915496497085812	-37.0539594928153	aracaju	SE
4	49010	aracaju	49010	-10.910924528449412	-37.050401528882091	aracaju	SE
5	49010	aracaju	49010	-10.915000478792603	-37.052360025535222	aracaju	SE

c. Order_items.csv

i. Total number of rows: 112,650

ii. Total number of unique rows (`order_id`): 98,666

There are many order_id which are repeated as number of order_items are more.

```
1  select order_id
2    , count(*) as count_order_id
3  from `sqltest-353804.ecommerce.order_items`
4  group by 1
5  order by 2 desc
6
7
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	order_id	count_order_id
1	8272b63d03f5f79c56e9e4120aec44ef	21
2	1b15974a0141d54e36626dca3fdc731a	20
3	ab14fdcfbe524636d65ee38360e22ce8	20
4	9ef13efd6949e4573a18964dd1bbe7f5	15
5	428a2f660dc84138d969ccd69a0ab6d5	15

d. Payments.csv

i. Total number of rows: 103,886

ii. Total number of unique rows (`order_id`): 99,440

```sql
select distinct order_id
  , count(*)
from `sqltest-353804.ecommerce.payments`
group by 1
order by 2 desc
limit 100
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

ow	order_id	f0_
	fa65dad1b0e818e3ccc5cb0e39231352	29
	ccf804e764ed5650cd8759557269dc13	26
	285c2e15bebd4ac83635ccc563dc71f4	22
	895ab968e7bb0d5659d16cd74cd1650c	21
	fedcd9f7ccdc8cba3a18defedd1a5547	19

```sql
select *
from `sqltest-353804.ecommerce.payments`
where order_id = 'fa65dad1b0e818e3ccc5cb0e39231352'
limit 100
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	order_id	payment_sequential	payment_type	payment_installments	payment_value
1	fa65dad1b0e818e3ccc5cb0e39231352	14	voucher	1	0.0
2	fa65dad1b0e818e3ccc5cb0e39231352	13	voucher	1	0.0
3	fa65dad1b0e818e3ccc5cb0e39231352	20	voucher	1	150.0
4	fa65dad1b0e818e3ccc5cb0e39231352	6	voucher	1	5.02
5	fa65dad1b0e818e3ccc5cb0e39231352	19	voucher	1	5.02
6	fa65dad1b0e818e3ccc5cb0e39231352	15	voucher	1	14.04

Payment_sequential: sequences of the payments made in case of EMI for a given order_id

e. Reviews.csv

i. Total number of rows: 99,224

ii. Total number of unique rows (`review_id`): 98,410

```
1  select distinct review_id
2    , count(*)
3  from `sqltest-353804.ecommerce.order_reviews`
4  group by 1
5  order by 2 desc
6  limit 100
7
8
```

Query results

JOB INFORMATION | RESULTS | JSON | E

Row	review_id	f0_
1	1fb4ddc969e6bea80e38deec00393a6f	3
2	f4bb9d6dd4fb6dcc2298f0e7b17b8e1e	3
3	3415c9f764e478409e8e0660ae816dd2	3
4	4d0e6dd087008d1f992d25ef6e1f619f	3
5	2d6ac45f859465b5c185274a1c929637	3
6	38821b5c496b678cf91acc34892805ad	3

```
1  select *
2  from `sqltest-353804.ecommerce.order_reviews`
3  where review_id = '1fb4ddc969e6bea80e38deec00393a6f'
4
5
```

Press Alt+F1 for Accessibil

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

tow	review_id	order_id	review_score	review_comment_title	review_creation_date	review_answer_timestamp
	1fb4ddc969e6bea80e38deec00393a6f	d6dde74bdeb424af6b660214881b4845	5	*null*	0011-08-17 00:00:00 UTC	0012-08-17 14:35:00 UTC
	1fb4ddc969e6bea80e38deec00393a6f	afed4265a8b956d840bc032e54dfccd1	5	*null*	0011-08-17 00:00:00 UTC	0012-08-17 14:35:00 UTC
	1fb4ddc969e6bea80e38deec00393a6f	3c1098cb17277b62cfc709c7a9b500f5	5	*null*	0011-08-17 00:00:00 UTC	0012-08-17 14:35:00 UTC

5 is good review and 1 is bad review

f. Orders.csv

i. Total number of rows: 99,441

ii. Total number of unique rows (`order_id`): 99,441

```sql
select count(customer_id)
  , count(order_id)
from `sqltest-353804.ecommerce.orders`
limit 10
```

Query results

JOB INFORMATION | RESULTS | JSON

Row	f0_	f1_
	99441	99441

g. Products.csv
   i. Total number of rows: 32,951
   ii. Total number of unique rows (`product_id`): 32,951
h. Sellers.csv
   i. Total number of rows: 3,095
   ii. Total number of unique rows (`seller_id`): 3,095

—---------- —----------

2. Number of null or missing values in a column

Solution:

```sql
select sum(case when customer_id is null then 1 else 0 end) as missing_customer_id
  , sum(case when customer_unique_id is null then 1 else 0 end) as missing_customer_unique_id
  , sum(case when customer_zip_code_prefix is null then 1 else 0 end) as missing_customer_zip_code_prefix
  , sum(case when customer_city is null then 1 else 0 end) as missing_customer_city
  , sum(case when customer_state is null then 1 else 0 end) as missing_customer_customer_state
from `sqltest-353804.ecommerce.customers`
limit 10
```

Query results SAVE RESULTS

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	missing_customer_id	missing_customer_unique_id	missing_customer_zip_code_prefix	missing_customer_city	missing_customer_customer_state
	0	0	0	0	0

```sql
select sum(case when geolocation_zip_code_prefix is null then 1 else 0 end) as missing_geolocation_zip_code_prefix
  , sum(case when geolocation_lat is null then 1 else 0 end) as missing_geolocation_lat
  , sum(case when geolocation_lng is null then 1 else 0 end) as missing_geolocation_lng
  , sum(case when geolocation_city is null then 1 else 0 end) as missing_geolocation_city
  , sum(case when geolocation_state is null then 1 else 0 end) as missing_geolocation_state
from `sqltest-353804.ecommerce.geolocation`
limit 10
```

Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	missing_geolocation_zip_code_prefix	missing_geolocation_lat	missing_geolocation_lng	missing_geolocation_city	missing_geolocation_state
1	0	0	0	0	0

```sql
select sum(case when order_id is null then 1 else 0 end) as missing_order_id
  , sum(case when order_item_id is null then 1 else 0 end) as missing_order_item_id
  , sum(case when product_id is null then 1 else 0 end) as missing_product_id
  , sum(case when seller_id is null then 1 else 0 end) as missing_seller_id
  , sum(case when shipping_limit_date is null then 1 else 0 end) as missing_shipping_limit_date
  , sum(case when price is null then 1 else 0 end) as price
  , sum(case when freight_value is null then 1 else 0 end) as freight_value
from `sqltest-353804.ecommerce.order_items`
```

Query results

SAVE RES

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	missing_order_id	missing_order_item_id	missing_product_id	missing_seller_id	missing_shipping_limit_date	price	freight_value
1	0	0	0	0	0	0	0

```sql
select sum(case when review_id is null then 1 else 0 end) as missing_review_id
  , sum(case when order_id is null then 1 else 0 end) as missing_order_id
  , sum(case when review_score is null then 1 else 0 end) as missing_review_score
  , sum(case when review_comment_title is null then 1 else 0 end) as missing_review_comment_title
  , sum(case when review_creation_date is null then 1 else 0 end) as missing_review_creation_date
  , sum(case when review_answer_timestamp is null then 1 else 0 end) as missing_review_answer_timestamp
from `sqltest-353804.ecommerce.order_reviews`
limit 10;
```

Press Alt+F

Query results

SAVE RESULTS | EXPL

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

ow	missing_review_id	missing_order_id	missing_review_score	missing_review_comment_title	missing_review_creation_date	missing_review_answer_timestamp
	0	0	0	87675	0	0

```sql
select sum(case when order_id is null then 1 else 0 end) as missing_order_id
  , sum(case when customer_id is null then 1 else 0 end) as missing_customer_id
  , sum(case when order_status is null then 1 else 0 end) as missing_order_status
  , sum(case when order_purchase_timestamp is null then 1 else 0 end) as missing_order_purchase_timestamp
  , sum(case when order_approved_at is null then 1 else 0 end) as missing_order_approved_at
  , sum(case when order_delivered_carrier_date is null then 1 else 0 end) as missing_order_delivered_carrier_date
  , sum(case when order_estimated_delivery_date is null then 1 else 0 end) as missing_order_estimated_delivery_date
from `sqltest-353804.ecommerce.orders`
limit 10;
```

Press Alt+F1 for Accessibility Options

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	missing_order_id	missing_customer_id	missing_order_status	missing_order_purchase_timestamp	missing_order_approved_at	missing_order_delivered_carrier_date	missing_order_estimated_delivery_da
1	0	0	0	0	160	1783	0

```sql
select sum(case when order_id is null then 1 else 0 end) as missing_order_id
  , sum(case when payment_sequential is null then 1 else 0 end) as missing_payment_sequential
  , sum(case when payment_type is null then 1 else 0 end) as missing_payment_type
  , sum(case when payment_installments is null then 1 else 0 end) as missing_payment_installments
  , sum(case when payment_value is null then 1 else 0 end) as missing_payment_value
from `sqltest-353804.ecommerce.payments`
limit 10;
```

Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	missing_order_id	missing_payment_sequential	missing_payment_type	missing_payment_installments	missing_payment_value
1	0	0	0	0	0

```sql
select sum(case when product_id is null then 1 else 0 end) as missing_product_id
  , sum(case when product_category is null then 1 else 0 end) as missing_product_category
  , sum(case when product_name_length is null then 1 else 0 end) as missing_product_name_length
  , sum(case when product_description_length is null then 1 else 0 end) as missing_product_description_length
  , sum(case when product_photos_qty is null then 1 else 0 end) as missing_product_photos_qty
  , sum(case when product_weight_g is null then 1 else 0 end) as missing_product_weight_g
  , sum(case when product_length_cm is null then 1 else 0 end) as missing_product_length_cm
  , sum(case when product_height_cm is null then 1 else 0 end) as missing_product_height_cm
  , sum(case when product_width_cm is null then 1 else 0 end) as missing_product_width_cm
from `sqltest-353804.ecommerce.products`
limit 10;
```

Press Alt+F1 for Accessibility Option

ery results

SAVE RESULTS | EXPLORE DATA

INFORMATION | RESULTS | JSON | EXECUTION DETAILS

missing_product_id	missing_product_category	missing_product_name_length	missing_product_description_length	missing_product_photos_qty	missing_product_weight_g	missing_product_length_cm	miss
0	610	610	610	610	2	2	2

```sql
select sum(case when seller_id is null then 1 else 0 end) as missing_seller_id
  , sum(case when seller_zip_code_prefix is null then 1 else 0 end) as missing_seller_zip_code_prefix
  , sum(case when seller_city is null then 1 else 0 end) as missing_seller_city
  , sum(case when seller_state is null then 1 else 0 end) as missing_seller_state
from `sqltest-353804.ecommerce.sellers`
limit 10;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	missing_seller_id	missing_seller_zip_code_prefix	missing_seller_city	missing_seller_state
1	0	0	0	0

—---------- —----------

3. Data type of columns in a table

Solution:

```sql
select column_name, data_type
from `sqltest-353804.ecommerce.INFORMATION_SCHEMA.COLUMNS`
where table_name = 'customers'
order by ordinal_position
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAIL

ow	column_name	data_type
	customer_id	STRING
	customer_unique_id	STRING
	customer_zip_code_prefix	INT64
	customer_city	STRING
	customer_state	STRING

```sql
select column_name, data_type
from `sqltest-353804.ecommerce.INFORMATION_SCHEMA.COLUMNS`
where table_name = 'geolocation'
order by ordinal_position
```

ıery results

B INFORMATION | RESULTS | JSON | EXECUTION DETAI

column_name	data_type
geolocation_zip_code_prefix	INT64
geolocation_lat	FLOAT64
geolocation_lng	FLOAT64
geolocation_city	STRING
geolocation_state	STRING

```
select column_name, data_type
from `sqltest-353804.ecommerce.INFORMATION_SCHEMA.COLUMNS`
where table_name = 'order_items'
order by ordinal_position
```

ery results

INFORMATION | RESULTS | JSON | EXECUTION DETAILS

column_name	data_type
order_id	STRING
order_item_id	INT64
product_id	STRING
seller_id	STRING
shipping_limit_date	TIMESTAMP
price	FLOAT64
freight_value	FLOAT64

```
select column_name, data_type
from `sqltest-353804.ecommerce.INFORMATION_SCHEMA.COLUMNS`
where table_name = 'order_reviews'
order by ordinal_position
```

ery results

INFORMATION | RESULTS | JSON | EXECUTION DETAIL

column_name	data_type
review_id	STRING
order_id	STRING
review_score	INT64
review_comment_title	STRING
review_creation_date	TIMESTAMP
review_answer_timestamp	TIMESTAMP

```
select column_name, data_type
from `sqltest-353804.ecommerce.INFORMATION_SCHEMA.COLUMNS`
where table_name = 'orders'
order by ordinal_position
```

ery results

INFORMATION | RESULTS | JSON | EXECUTION DETAIL

column_name	data_type
order_id	STRING
customer_id	STRING
order_status	STRING
order_purchase_timestamp	TIMESTAMP
order_approved_at	TIMESTAMP
order_delivered_carrier_date	TIMESTAMP
order_delivered_customer_date	TIMESTAMP
order_estimated_delivery_date	TIMESTAMP

```
select column_name, data_type
from `sqltest-353804.ecommerce.INFORMATION_SCHEMA.COLUMNS`
where table_name = 'payments'
order by ordinal_position
```

ery results

INFORMATION | RESULTS | JSON | EXECUTION DETAIL

column_name	data_type
order_id	STRING
payment_sequential	INT64
payment_type	STRING
payment_installments	INT64
payment_value	FLOAT64

```
select column_name, data_type
from `sqltest-353804.ecommerce.INFORMATION_SCHEMA.COLUMNS`
where table_name = 'products'
order by ordinal_position
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAIL

ow	column_name	data_type
	product_id	STRING
	product_category	STRING
	product_name_length	INT64
	product_description_length	INT64
	product_photos_qty	INT64
	product_weight_g	INT64
	product_length_cm	INT64
	product_height_cm	INT64
	product_width_cm	INT64

```
select column_name, data_type
from `sqltest-353804.ecommerce.INFORMATION_SCHEMA.COLUMNS`
where table_name = 'sellers'
order by ordinal_position
```

iery results

3 INFORMATION | RESULTS | JSON | EXECUTION DETAI

column_name	data_type
seller_id	STRING
seller_zip_code_prefix	INT64
seller_city	STRING
seller_state	STRING

—---------- —----------

4. Get the time period for which the data is given

Solution:

```
select min(order_purchase_timestamp) as min_timestamp
  , max(order_purchase_timestamp) as max_timestamp
  , DATE_diff(max(order_purchase_timestamp), min(order_purchase_timestamp), DAY)/365 AS num_year
from `sqltest-353804.ecommerce.orders`
limit 10;
```

Query results

SAVE RESULT

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	min_timestamp	max_timestamp	num_year
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC	2.1150684931506851

—---------- —----------

5. Number of cities in our dataset

Solution:

```
1  select count(distinct geolocation_city)
2  from `sqltest-353804.ecommerce.geolocation`
3  limit 10;
4
```

Query results

JOB INFORMATION | RESULTS | JSON | EXEC

Row	f0_
1	8011

—---------- —----------

6. Number of states in our dataset

Solution:

```
1  select count(distinct geolocation_state)
2  from `sqltest-353804.ecommerce.geolocation`
3  limit 10;
4
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DE

Row	f0_
1	27

—---------- —----------

########################## END SOLUTION 1############################

2. In-depth Exploration:

1. How many orders do we have for each order status?

Solution:

```
select order_status
  , count(distinct order_id) as num_order_status
from `sqltest-353804.ecommerce.orders`
group by 1
order by 2 desc;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAIL

Row	order_status	num_order_status
1	delivered	96478
2	shipped	1107
3	canceled	625
4	unavailable	609
5	invoiced	314
6	processing	301
7	created	5
8	approved	2

—---------- —----------

2. Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario?

Solution:

We can describe in terms of number of orders, total_purchase_amount year-on-year and sellers_added year-on-year.

```
with num_order as (

 select extract(year from eo.order_purchase_timestamp) as purchase_year

       , count(distinct eo.customer_id) as num_customers_per_year

       , count(distinct eo.order_id) as num_orders_per_year
```

```sql
       , round(sum(ep.payment_value),2) as total_purchase_value_per_year

       , count(distinct es.seller_id) as num_sellers_per_year

 from `sqltest-353804.ecommerce.orders` eo

 left join `sqltest-353804.ecommerce.payments` ep

 ON eo.order_id = ep.order_id

 left join `sqltest-353804.ecommerce.order_items` eoi

 on eo.order_id = eoi.order_id

 left join `sqltest-353804.ecommerce.sellers` es

 on eoi.seller_id = es.seller_id

 group by 1

)

select *

  , round(((num_customers_per_year - lag(num_customers_per_year,1) OVER (ORDER BY
purchase_year))/num_customers_per_year) *100,4) as
year_on_year_num_customers_change_pct

     , round(((num_sellers_per_year - lag(num_sellers_per_year,1) OVER (ORDER BY
purchase_year))/num_sellers_per_year) *100,4) as year_on_year_num_sellers_change_pct

  , round(((num_orders_per_year - lag(num_orders_per_year,1) OVER (ORDER BY
purchase_year))/num_orders_per_year) *100,4) as year_on_year_num_orders_change_pct

  , round(((total_purchase_value_per_year - lag(total_purchase_value_per_year,1) OVER
(ORDER BY purchase_year))/total_purchase_value_per_year) *100,4) as
year_on_year_purchase_value_change_pct

from num_order

order by 1;
```

purchase_year	num_customers_per_year	num_orders_per_year	total_purchase_value_per_year	num_sellers_per_year	year_on_year_num_customers_change_pct	year_on_year_num_sellers_change_pct	year_on_year_num_orders_change_pct	year_on_year_purchase_value_change_pct
2016	329	329	76528.26	145				
2017	45101	45101	9266612.28	1784	99.2705	91.8722	99.2705	99.1742
2018	54011	54011	11127586.12	2383	16.4966	25.1364	16.4966	16.724

3. On what day of week brazilians customers tend to do online purchasing?

Solution:

```
1  select FORMAT_DATE('%A', order_purchase_timestamp) as purchase_dayofweekname
2  , count(distinct order_id) as num_orders
3  from `sqltest-353804.ecommerce.orders`
4  group by 1
5  order by 2 desc;
```

uery results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

purchase_dayofweekname	num_orders
Monday	16196
Tuesday	15963
Wednesday	15552
Thursday	14761
Friday	14122
Sunday	11960
Saturday	10887

```
select concat(extract(year from order_purchase_timestamp) , '_',extract(week from order_purchase_timestamp)) as purchase_year_week
  , extract(day from order_purchase_timestamp) as purchase_day_of_week
  , extract(month from order_purchase_timestamp) as purchase_month
  , count(distinct order_id) as num_orders
from `sqltest-353804.ecommerce.orders`
group by 1, 2, 3
```

Press Alt+F1 for Accessibility

Query results SAVE RESULTS EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	purchase_year_week	purchase_day_of_week	purchase_month	num_orders
1	2017_47	25	11	499
2	2017_49	5	12	282
3	2018_5	9	2	216
4	2017_45	6	11	193
5	2017_16	20	4	98

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

Solution:

```
with hr_extract as (

 select extract(hour from order_purchase_timestamp) as purchase_hour

 , count(distinct order_id) as num_orders

from `sqltest-353804.ecommerce.orders`

group by 1

)

select case

       when purchase_hour>= 0 and purchase_hour < 6 then 'dawn'

       when purchase_hour>= 6 and purchase_hour < 12 then 'morning'

       when purchase_hour>= 12 and purchase_hour < 18 then 'afternoon'

       when purchase_hour>= 18 and purchase_hour < 24 then 'night'

     end as purchase_time
```

```
    , sum(num_orders) as Total_purchases
from hr_extract
group by 1
order by 2 desc
```

purchase_time	Total_purchases
afternoon	38361
night	34100
morning	22240
dawn	4740

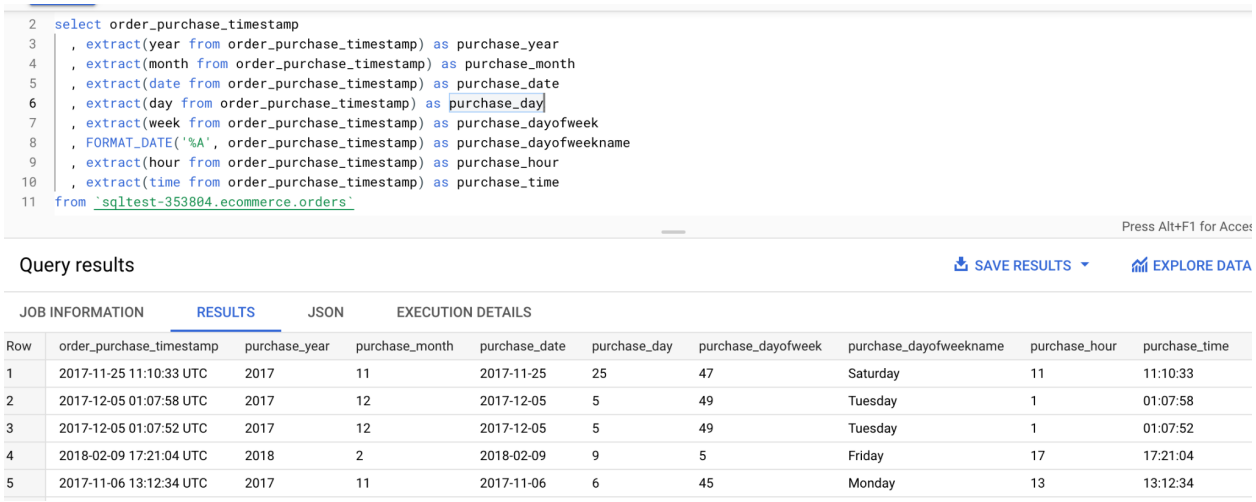
—---------- —----------

5. Feature Extraction: Through order_purchase_timestamp in “orders” dataset extract
    1. order_purchase_year
    2. order_purchase_month
    3. order_purchase_date
    4. order_purchase_day
    5. order_purchase_dayofweek
    6. order_purchase_dayofweek_name
    7. order_purchase_hour
    8. order_purchase_time_day

Solution:

```
select order_purchase_timestamp
 , extract(year from order_purchase_timestamp) as purchase_year
 , extract(month from order_purchase_timestamp) as purchase_month
 , extract(date from order_purchase_timestamp) as purchase_date
```

```
 , extract(day from order_purchase_timestamp) as purchase_day
 , extract(week from order_purchase_timestamp) as purchase_dayofweek
 , FORMAT_DATE('%A', order_purchase_timestamp) as purchase_dayofweekname
 , extract(hour from order_purchase_timestamp) as purchase_hour
 , extract(time from order_purchase_timestamp) as purchase_time
from `sqltest-353804.ecommerce.orders`
limit 10
```



```
select order_purchase_timestamp
  , extract(year from order_purchase_timestamp) as purchase_year
  , extract(month from order_purchase_timestamp) as purchase_month
  , extract(date from order_purchase_timestamp) as purchase_date
  , extract(day from order_purchase_timestamp) as purchase_day
  , extract(week from order_purchase_timestamp) as purchase_dayofweek
  , FORMAT_DATE('%A', order_purchase_timestamp) as purchase_dayofweekname
  , extract(hour from order_purchase_timestamp) as purchase_hour
  , extract(time from order_purchase_timestamp) as purchase_time
from `sqltest-353804.ecommerce.orders`
```

Press Alt+F1 for Acces

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	order_purchase_timestamp	purchase_year	purchase_month	purchase_date	purchase_day	purchase_dayofweek	purchase_dayofweekname	purchase_hour	purchase_time
1	2017-11-25 11:10:33 UTC	2017	11	2017-11-25	25	47	Saturday	11	11:10:33
2	2017-12-05 01:07:58 UTC	2017	12	2017-12-05	5	49	Tuesday	1	01:07:58
3	2017-12-05 01:07:52 UTC	2017	12	2017-12-05	5	49	Tuesday	1	01:07:52
4	2018-02-09 17:21:04 UTC	2018	2	2018-02-09	9	5	Friday	17	17:21:04
5	2017-11-06 13:12:34 UTC	2017	11	2017-11-06	6	45	Monday	13	13:12:34

—---------- —----------

################## END SOLUTION 2#########################

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

1. Get month on month orders by region

Solution:

```
with order_count_year_month as

(

select ec.customer_state

 , extract(year from order_purchase_timestamp) as purchase_year

 , extract (month from order_purchase_timestamp) as purchase_month

 , count(distinct order_id) as num_order

from `sqltest-353804.ecommerce.orders` eo

left join `sqltest-353804.ecommerce.customers` ec

on eo.customer_id = ec.customer_id

group by 1,2,3

),

lagged_calc1 as (

 select *
```

```
 , lag(om.num_order) OVER(partition by om.customer_state, om.purchase_year order by om.purchase_month) as lag_order

 from order_count_year_month om

)

select *

 , round(((lc.num_order- lc.lag_order)/lc.lag_order)*100 ,2) as month_over_month_change_in_num_order_by_state_pct

from lagged_calc1 lc

order by 1,2,3

limit 100;
```

customer_state	purchase_year	purchase_month	num_order	lag_order	month_over_month_change_in_num_order_by_state_pct
AC	2017	1	2		
AC	2017	2	3	2	50
AC	2017	3	2	3	-33.33
AC	2017	4	5	2	150
AC	2017	5	8	5	60
AC	2017	6	4	8	-50
AC	2017	7	5	4	25
AC	2017	8	4	5	-20
AC	2017	9	5	4	25
AC	2017	10	6	5	20
AC	2017	11	5	6	-16.67
AC	2017	12	5	5	0
AC	2018	1	6		
AC	2018	2	3	6	-50
AC	2018	3	2	3	-33.33
AC	2018	4	4	2	100
AC	2018	5	2	4	-50
AC	2018	6	3	2	50
AC	2018	7	4	3	33.33
AC	2018	8	3	4	-25
AL	2016	10	2		
AL	2017	1	2		
AL	2017	2	12	2	500
AL	2017	3	10	12	-16.67
AL	2017	4	23	10	130
AL	2017	5	27	23	17.39
AL	2017	6	10	27	-62.96
AL	2017	7	17	10	70

—---------- —----------

2. Total of customer orders by state

<u>Solution:</u>

```sql
-- Total of customer orders by state
select ec.customer_state
  , count(distinct eo.order_id) as total_customer_order_by_state
from `sqltest-353804.ecommerce.customers` ec
left join `sqltest-353804.ecommerce.orders` eo
on ec.customer_id = eo.customer_id
group by 1
order by 2 desc
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	total_customer_order_by_state
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045

---

3. Top 10 brazilian cities most no. of orders

Solution:

```sql
select ec.customer_city

 , count(distinct eo.order_id) as total_customer_order_by_city

from `sqltest-353804.ecommerce.customers` ec

left join `sqltest-353804.ecommerce.orders` eo

on ec.customer_id = eo.customer_id

group by 1

order by 2 desc

limit 10;
```

customer_city	total_customer_order_by_city
sao paulo	15540
rio de janeiro	6882
belo horizonte	2773
brasilia	2131
curitiba	1521
campinas	1444
porto alegre	1379
salvador	1245
guarulhos	1189
sao bernardo do campo	938

—---------- —----------

4. How are customers distributed in Brazil

Solution:

```sql
-- How are customers distributed in Brazil
select customer_state
 , count(distinct customer_id) as total_customer_by_state
from `sqltest-353804.ecommerce.customers`
group by 1
order by 2 desc
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	total_customer_by_state
	SP	41746
	RJ	12852
	MG	11635
	RS	5466
	PR	5045
	SC	3637
	BA	3380
	DF	2140

—---------- —----------

5. City wise number of unique customers

Solution:

```sql
select customer_city

 , count(customer_id) as total_customer_by_city

from `sqltest-353804.ecommerce.customers`

group by 1

order by 2 desc

limit 10;
```

customer_city	total_customer_by_city
sao paulo	15540
rio de janeiro	6882
belo horizonte	2773
brasilia	2131
curitiba	1521
campinas	1444
porto alegre	1379
salvador	1245
guarulhos	1189
sao bernardo do campo	938

—---------- —----------

################## END SOLUTION 3 ########################

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

Step 1: Using CTE

1. “order_items” + “order” joined on order id where order_purchase timestamp is already divided into month & year

2. Group data by year and month, aggregation count(order_id), sum(price), sum(freight_value)

3. Create new columns:

price_per_order = sum(price) / count(order_id)

freight_per_order= sum(freight_value) / count(order_id)

Step 2: Answer the following questions:

1. Total amount sold in 2017 between Jan to August - *Answer should be ~3.11Mn*

2. Total amount sold in 2018 between Jan to august - *Answer should be ~7.39 Mn*

3. % increase from 2017 to 2018 (+142.15%)

Solution:

```
with price_value_year_month as (

select extract(year from eo.order_purchase_timestamp) as purchase_year

 , extract(month from eo.order_purchase_timestamp) as purchase_month

 , count(eo.order_id) as num_orders

 , sum(eoi.price) as total_price_by_year_month

 , sum(eoi.freight_value) as total_freight_value_by_year_month

from `sqltest-353804.ecommerce.order_items` eoi

inner join `sqltest-353804.ecommerce.orders` eo

on eoi.order_id = eo.order_id
```

```sql
group by 1, 2

),

per_order_calc as

(

select *

 , total_price_by_year_month/num_orders as price_per_order

 , total_freight_value_by_year_month/num_orders as freight_per_order

from price_value_year_month

),

total_amt_sold as

(

select

  sum(CASE WHEN purchase_year = 2017 and purchase_month BETWEEN 1 and 8 THEN
total_price_by_year_month ELSE 0 end) as total_amount_sold_2017_Jan_Aug

 , SUM(CASE WHEN purchase_year = 2018 and purchase_month BETWEEN 1 and 8 THEN
total_price_by_year_month ELSE 0 end) as total_amount_sold_2018_Jan_Aug

from per_order_calc

)

select round(total_amount_sold_2017_Jan_Aug,2)/1000000 as
total_amount_sold_2017_Jan_Aug_in_Millions

 , round(total_amount_sold_2018_Jan_Aug, 2)/1000000 as
total_amount_sold_2018_Jan_Aug_in_Millions
```

```
 ,
round(((total_amount_sold_2018_Jan_Aug-total_amount_sold_2017_Jan_Aug)/total_amount_so
ld_2017_Jan_Aug) , 4)*100 as pct_increase_2017_2018

from total_amt_sold;
```

total_amount_sold_2017_Jan_Aug_in_Millions	total_amount_sold_2018_Jan_Aug_in_Millions	pct_increase_2017_2018
3.11300032	7.3859058	137.26

Step 3: Join (orders+order_items) table from previous step with “customers” table on Customer_id and find:

1. Mean & Sum of price by customer state

2. Mean & Sum of freight value by customer state

Solution:

```
select ec.customer_state

 , round(avg(eoi.price),2) as mean_price

 , round(sum(eoi.price),2) as sum_price

 , round(avg(eoi.freight_value),2) as avg_freight_value

 , round(sum(eoi.freight_value),2) as sum_freight_value

from `sqltest-353804.ecommerce.order_items` eoi

inner join `sqltest-353804.ecommerce.orders` eo

on eoi.order_id = eo.order_id

right join `sqltest-353804.ecommerce.customers` ec

on eo.customer_id = ec.customer_id

group by 1

order by 1;
```

customer_state	mean_price	sum_price	avg_freight_value	sum_freight_value
AC	173.73	15982.95	40.07	3686.75
AL	180.89	80314.81	35.84	15914.59
AM	135.5	22356.84	33.21	5478.89
AP	164.32	13474.3	34.01	2788.5
BA	134.6	511349.99	26.36	100156.68
CE	153.76	227254.71	32.71	48351.59
DF	125.77	302603.94	21.04	50625.5
ES	121.91	275037.31	22.06	49764.6
GO	126.27	294591.95	22.77	53114.98
MA	145.2	119648.22	38.26	31523.77

—---------- —----------

################## END SOLUTION 4 ########################

5. Analysis on sales, freight and delivery time

1. Calculating days between purchasing, delivering and estimated delivery

Solution:

```
select order_id

   , order_delivered_carrier_date

   , order_delivered_customer_date

   , order_estimated_delivery_date

   , order_purchase_timestamp

   , date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as
days_between_purchaseAndDelivery

   , date_diff(order_estimated_delivery_date, order_purchase_timestamp,  day) as
days_between_purchaseAndEstimatedDelivery

   , date_diff(order_delivered_carrier_date, order_purchase_timestamp,  day) as
days_between_purchaseAndDeliveredAtCarrier



   , date_diff(order_delivered_customer_date, order_delivered_carrier_date,  day) as
days_between_DeliveredAtCarrierAndCustomerDelivery

   , date_diff(order_estimated_delivery_date, order_delivered_customer_date,  day) as
days_between_customerDeliveryrAndEstimatedDelivery

   , date_diff(order_estimated_delivery_date, order_delivered_carrier_date,  day) as
days_between_DeliveredAtCarrierAndEstimatedDelivery

 from `sqltest-353804.ecommerce.orders`
```

order_id	order_delivered_carrier_date	order_delivered_customer_date	order_estimated_delivery_date	order_purchase_timestamp
770d331c84e5b214bd9dc70a10b829d0	2016-10-11T15:07:11Z	2016-10-14T15:07:11Z	2016-11-29T00:00:00Z	2016-10-07T14:52:30Z
1950d777989f6a877539f53795b4c3c3	2018-02-20T19:57:13Z	2018-03-21T22:03:51Z	2018-03-09T00:00:00Z	2018-02-19T19:48:52Z
2c45c33d2f9cb8ff8b1c86cc28c11c30	2016-10-14T10:40:50Z	2016-11-09T14:53:50Z	2016-12-08T00:00:00Z	2016-10-09T15:39:56Z
dabf2b0e35b423f94618bf965fcb7514	2016-10-13T13:36:59Z	2016-10-16T14:36:59Z	2016-11-30T00:00:00Z	2016-10-09T00:56:52Z
8beb59392e21af5eb9547ae1a9938d06	2016-10-14T22:45:26Z	2016-10-19T18:47:43Z	2016-11-30T00:00:00Z	2016-10-08T20:17:50Z
65d1e226dfaeb8cdc42f665422522d14	2016-10-25T12:14:28Z	2016-11-08T10:58:34Z	2016-11-25T00:00:00Z	2016-10-03T21:01:41Z
c158e9806f85a33877bdfd4f607b72e7	2017-04-17T14:52:32Z	2017-05-08T11:10:26Z	2017-05-18T00:00:00Z	2017-04-14T22:06:32Z
b60b53ad0bb7dacacf2989fe27ad567a	2017-05-11T16:25:53Z	2017-05-23T13:12:27Z	2017-05-18T00:00:00Z	2017-05-10T14:03:27Z
c830f223aae08493ebecb52f29aa48ca	2017-04-25T08:23:19Z	2017-05-05T13:27:50Z	2017-05-18T00:00:00Z	2017-04-22T15:50:30Z
a8aa2cd070eeac7e4368cae3d8222e2b	2017-05-10T15:30:08Z	2017-05-16T23:22:20Z	2017-05-18T00:00:00Z	2017-05-09T17:42:45Z

days_between_purchaseAndDelivery	days_between_purchaseAndEstimatedDelivery	days_between_purchaseAndDeliveredAtCarrier	days_between_DeliveredAtCarrierAndCustomerDelivery	days_between_customerDeliveryrAndEstimatedDelivery	days_between_DeliveredAtCarrierAndEstimatedDelivery
7	52	4	3	45	48
30	17	1	29	-12	16
30	59	4	26	28	54
7	51	4	3	44	47
10	52	6	4	41	46
35	52	21	13	16	30
23	33	2	20	9	30
12	7	1	11	-5	6
12	25	2	10	12	22
7	8	0	6	1	7

—---------- —----------

2. Create columns:

Solution:

time_to_delivery = order_purchase_timestamp-order_delivered_customer_date

```
select order_id
  -- , order_delivered_carrier_date
  , order_delivered_customer_date
  -- , order_estimated_delivery_date
  , order_purchase_timestamp
  , date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as days_to_delivery
  , date_diff(order_delivered_customer_date, order_purchase_timestamp, day) * 24 as time_to_delivery
from `sqltest-353804.ecommerce.orders`
where order_delivered_customer_date is not null
```

order_id	order_delivered_customer_date	order_purchase_timestamp	days_to_delivery	time_to_delivery
770d331c84e5b214bd9dc70a10b829d0	2016-10-14 15:07:11 UTC	2016-10-07 14:52:30 UTC	7	168
1950d777989f6a877539f53795b4c3c3	2018-03-21 22:03:51 UTC	2018-02-19 19:48:52 UTC	30	720
2c45c33d2f9cb8ff8b1c86cc28c11c30	2016-11-09 14:53:50 UTC	2016-10-09 15:39:56 UTC	30	720
dabf2b0e35b423f94618bf965fcb7514	2016-10-16 14:36:59 UTC	2016-10-09 00:56:52 UTC	7	168
8beb59392e21af5eb9547ae1a9938d06	2016-10-19 18:47:43 UTC	2016-10-08 20:17:50 UTC	10	240
65d1e226dfaeb8cdc42f665422522d14	2016-11-08 10:58:34 UTC	2016-10-03 21:01:41 UTC	35	840
c158e9806f85a33877bdfd4f607b72e7	2017-05-08 11:10:26 UTC	2017-04-14 22:06:32 UTC	23	552
b60b53ad0bb7dacacf2989fe27ad567a	2017-05-23 13:12:27 UTC	2017-05-10 14:03:27 UTC	12	288
c830f223aae08493ebecb52f29aa48ca	2017-05-05 13:27:50 UTC	2017-04-22 15:50:30 UTC	12	288

diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date

```sql
select order_id
  -- , order_delivered_carrier_date
  , order_delivered_customer_date
  , order_estimated_delivery_date
  , order_purchase_timestamp
  , date_diff(order_estimated_delivery_date, order_delivered_customer_date, day) as diff_estimated_delivery
  -- , date_diff(order_delivered_customer_date, order_purchase_timestamp, day) * 24 as time_to_delivery
from `sqltest-353804.ecommerce.orders`
where order_delivered_customer_date is not null
limit 10
```

order_id	order_delivered_customer_date	order_estimated_delivery_date	order_purchase_timestamp	diff_estimated_delivery
770d331c84e5b214bd9dc70a10b829d0	2016-10-14 15:07:11 UTC	2016-11-29 00:00:00 UTC	2016-10-07 14:52:30 UTC	45
1950d777989f6a877539f53795b4c3c3	2018-03-21 22:03:51 UTC	2018-03-09 00:00:00 UTC	2018-02-19 19:48:52 UTC	-12
2c45c33d2f9cb8ff8b1c86cc28c11c30	2016-11-09 14:53:50 UTC	2016-12-08 00:00:00 UTC	2016-10-09 15:39:56 UTC	28
dabf2b0e35b423f94618bf965fcb7514	2016-10-16 14:36:59 UTC	2016-11-30 00:00:00 UTC	2016-10-09 00:56:52 UTC	44
8beb59392e21af5eb9547ae1a9938d06	2016-10-19 18:47:43 UTC	2016-11-30 00:00:00 UTC	2016-10-08 20:17:50 UTC	41
65d1e226dfaeb8cdc42f665422522d14	2016-11-08 10:58:34 UTC	2016-11-25 00:00:00 UTC	2016-10-03 21:01:41 UTC	16
c158e9806f85a33877bdfd4f607b72e7	2017-05-08 11:10:26 UTC	2017-05-18 00:00:00 UTC	2017-04-14 22:06:32 UTC	9
b60b53ad0bb7dacacf2989fe27ad567a	2017-05-23 13:12:27 UTC	2017-05-18 00:00:00 UTC	2017-05-10 14:03:27 UTC	-5

—---------- —----------

3. Grouping data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery

Solution:

```sql
with days_calc as (

 select customer_id

   , order_id

   , date_diff(order_estimated_delivery_date, order_delivered_customer_date, day) as diff_estimated_delivery

   , date_diff(order_delivered_customer_date, order_purchase_timestamp, day) * 24 as time_to_delivery

 from `sqltest-353804.ecommerce.orders`

)

select ec.customer_state
```

```
 , round(avg(eoi.freight_value),2) as avg_freight_value
 , round(avg(dc.diff_estimated_delivery),2) as avg_diff_estimated_delivery
 , round(avg(dc.time_to_delivery),2) as avg_time_to_delivery
from days_calc dc
right join `sqltest-353804.ecommerce.customers` ec
on dc.customer_id = ec.customer_id
inner join `sqltest-353804.ecommerce.order_items` eoi
on dc.order_id = eoi.order_id
group by 1
limit 10
```

customer_state	avg_freight_value	avg_diff_estimated_delivery	avg_time_to_delivery
RN	35.65	13.06	452.96
CE	32.71	10.26	492.89
RS	21.74	13.2	353.0
SC	21.47	10.67	348.5
SP	15.15	10.27	198.23
MG	20.63	12.4	276.37
BA	26.36	10.12	450.59
RJ	20.96	11.14	352.55

—---------- —----------

4. Sort the data to get the following:

a. Top 5 states with highest/lowest average freight value

Solution:

Top 5 states with Highest average freight value:

```sql
with days_calc as (
  select customer_id
    , order_id
    , date_diff(order_estimated_delivery_date, order_delivered_customer_date, day) as diff_estimated_delivery
    , date_diff(order_delivered_customer_date, order_purchase_timestamp, day) * 24 as time_to_delivery
  from `sqltest-353804.ecommerce.orders`
)
select ec.customer_state
  , round(avg(eoi.freight_value),2) as avg_freight_value
  , round(avg(dc.diff_estimated_delivery),2) as avg_diff_estimated_delivery
  , round(avg(dc.time_to_delivery),2) as avg_time_to_delivery
from days_calc dc
right join `sqltest-353804.ecommerce.customers` ec
on dc.customer_id = ec.customer_id
inner join `sqltest-353804.ecommerce.order_items` eoi
on dc.order_id = eoi.order_id
group by 1
order by 1 desc
limit 10
```

customer_state	avg_freight_value	avg_diff_estimated_delivery	avg_time_to_delivery
RR	42.98	17.43	667.83
PB	42.72	12.15	482.87
RO	41.07	19.08	462.77
AC	40.07	20.01	487.91
PI	39.15	10.68	454.35
MA	38.26	9.11	508.89

—---------- —----------

Top 5 states with Lowest average freight value:

```sql
with days_calc as (
  select customer_id
    , order_id
    , date_diff(order_estimated_delivery_date, order_delivered_customer_date, day) as diff_estimated_delivery
    , date_diff(order_delivered_customer_date, order_purchase_timestamp, day) * 24 as time_to_delivery
  from `sqltest-353804.ecommerce.orders`
)
select ec.customer_state
  , round(avg(eoi.freight_value),2) as avg_freight_value
  , round(avg(dc.diff_estimated_delivery),2) as avg_diff_estimated_delivery
  , round(avg(dc.time_to_delivery),2) as avg_time_to_delivery
from days_calc dc
right join `sqltest-353804.ecommerce.customers` ec
on dc.customer_id = ec.customer_id
inner join `sqltest-353804.ecommerce.order_items` eoi
on dc.order_id = eoi.order_id
group by 1
order by 2
limit 10
```

customer_state	avg_freight_value	avg_diff_estimated_delivery	avg_time_to_delivery
SP	15.15	10.27	198.23
PR	20.53	12.53	275.54
MG	20.63	12.4	276.37
RJ	20.96	11.14	352.55
DF	21.04	11.27	300.04

—---------- —----------

b. Top 5 states with highest/lowest average time to delivery

Solution:

Top 5 states with Highest average time to delivery:

Similar query as above (only change as:

Order by 4 desc)

customer_state	avg_freight_value	avg_diff_estimated_delivery	avg_time_to_delivery
RR	42.98	17.43	667.83
AP	34.01	17.44	666.07
AM	33.21	18.98	623.12
AL	35.84	7.98	575.83
PA	35.83	13.37	559.24

—---------- —----------

Top 5 states with lowest average time to delivery:

customer_state	avg_freight_value	avg_diff_estimated_delivery	avg_time_to_delivery
SP	15.15	10.27	198.23
PR	20.53	12.53	275.54
MG	20.63	12.4	276.37
DF	21.04	11.27	300.04
SC	21.47	10.67	348.5

Similar query as above (only change as:

Order by 4)

—---------- —----------

c. Top 5 states where delivery is really fast/ not so fast compared to estimated date

Solution:

```
with days_calc as (
  select customer_id
    , order_id
    , date_diff(order_estimated_delivery_date, order_delivered_customer_date, day) as diff_estimated_delivery
    , date_diff(order_delivered_customer_date, order_purchase_timestamp, day) * 24 as time_to_delivery
  from `sqltest-353804.ecommerce.orders`
)
select ec.customer_state
  , round(avg(eoi.freight_value),2) as avg_freight_value
  , round(avg(dc.diff_estimated_delivery),2) as avg_diff_estimated_delivery
  , round(avg(dc.time_to_delivery)/24,2) as avg_time_to_delivery
from days_calc dc
right join `sqltest-353804.ecommerce.customers` ec
on dc.customer_id = ec.customer_id
inner join `sqltest-353804.ecommerce.order_items` eoi
on dc.order_id = eoi.order_id
group by 1
having avg_time_to_delivery < avg_diff_estimated_delivery
order by 4
limit 10
```

customer_state	avg_freight_value	avg_diff_estimated_delivery	avg_time_to_delivery
SP	15.15	10.27	8.26
PR	20.53	12.53	11.48
MG	20.63	12.4	11.52

—---------- —----------

################## END SOLUTION 5 ########################

6. Payment type analysis: Join “payments” dataset with the existing data on order_id

a. Count of orders for different payment types

Solution:

```
with days_calc as (

 select customer_id

   , order_id

   , date_diff(order_estimated_delivery_date, order_delivered_customer_date, day) as diff_estimated_delivery

   , date_diff(order_delivered_customer_date, order_purchase_timestamp, day) * 24 as time_to_delivery

 from `sqltest-353804.ecommerce.orders`

)

select ep.payment_type

 , count(distinct dc.order_id) as num_order

from days_calc dc

right join `sqltest-353804.ecommerce.customers` ec

on dc.customer_id = ec.customer_id

inner join `sqltest-353804.ecommerce.order_items` eoi

on dc.order_id = eoi.order_id

left join `sqltest-353804.ecommerce.payments` ep

on dc.order_id = ep.order_id
```

```
group by 1
```

payment_type	num_order
credit_card	75991
UPI	19614
voucher	3766
debit_card	1521
null	1

—---------- —----------

b. Distribution of payment installments and count of orders

Solution:

```
with days_calc as (
  select customer_id
    , order_id
    , date_diff(order_estimated_delivery_date, order_delivered_customer_date, day) as diff_estimated_delivery
    , date_diff(order_delivered_customer_date, order_purchase_timestamp, day) * 24 as time_to_delivery
  from `sqltest-353804.ecommerce.orders`
)
select ep.payment_installments
  , count(distinct dc.order_id) as num_order
from days_calc dc
right join `sqltest-353804.ecommerce.customers` ec
on dc.customer_id = ec.customer_id
inner join `sqltest-353804.ecommerce.order_items` eoi
on dc.order_id = eoi.order_id
left join `sqltest-353804.ecommerce.payments` ep
on dc.order_id = ep.order_id
group by 1
order by 1
```

payment_installments	num_order
	1
0	2
1	48609
2	12317
3	10385
4	7040
5	5195
6	3895
7	1612
8	4239
9	635
10	5261
11	22
12	131
13	16
14	15
15	74
16	5
17	7
18	27
20	17
21	3
22	1
23	1
24	18

—---------- —----------

c. Count of orders for different payment types Month over Month

Solution:

```
with days_calc as (

 select customer_id

   , order_id

   , extract(year from order_purchase_timestamp) as purchase_year
```

```
   , extract(month from order_purchase_timestamp)  as purchase_month

   , date_diff(order_estimated_delivery_date, order_delivered_customer_date, day) as
diff_estimated_delivery

   , date_diff(order_delivered_customer_date, order_purchase_timestamp, day) * 24 as
time_to_delivery

 from `sqltest-353804.ecommerce.orders`

)

select ep.payment_type

 , dc.purchase_year

 , dc.purchase_month

 , count(distinct dc.order_id) as num_order

 , lag(count(distinct dc.order_id)) over(partition by ep.payment_type order by
purchase_year, purchase_month ) as num_order_last_period

 , round(((count(distinct dc.order_id) - lag(count(distinct dc.order_id))
over(partition by ep.payment_type order by purchase_year, purchase_month
))/(lag(count(distinct dc.order_id)) over(partition by ep.payment_type order by
purchase_year, purchase_month ))),4)*100 as month_over_month_increase

from days_calc dc

right join `sqltest-353804.ecommerce.customers` ec

on dc.customer_id = ec.customer_id

inner join `sqltest-353804.ecommerce.order_items` eoi

on dc.order_id = eoi.order_id

left join `sqltest-353804.ecommerce.payments` ep

on dc.order_id = ep.order_id
```

```
group by 1,2,3

limit 100
```

payment_type	purchase_year	purchase_month	num_order	num_order_last_	month_over_month_increase
debit_card	2018	3	78	69	13.04
debit_card	2018	4	97	78	24.36
debit_card	2018	5	50	97	-48.45
debit_card	2018	6	180	50	260
debit_card	2018	7	241	180	33.89
debit_card	2018	8	276	241	14.52
voucher	2016	10	10		
voucher	2017	1	32	10	220
voucher	2017	2	67	32	109.38
voucher	2017	3	123	67	83.58
voucher	2017	4	114	123	-7.32
voucher	2017	5	171	114	50
voucher	2017	6	141	171	-17.54
voucher	2017	7	200	141	41.84
voucher	2017	8	195	200	-2.5
voucher	2017	9	173	195	-11.28
voucher	2017	10	203	173	17.34
voucher	2017	11	263	203	29.56
voucher	2017	12	217	263	-17.49
voucher	2018	1	301	217	38.71
voucher	2018	2	219	301	-27.24

—---------- —----------

################## END SOLUTION 6 ########################

CONCLUSION

1. Initial exploration of dataset like checking the characteristics of data

- There are (customer_id): 99,441 unique customers and number of order_items as: 112,650 . And the number of order_id is 99,441. So, it seems one customer has multiple orders.
- There are 32,951 products listed and 3,095 sellers selling these products
- There are not many missing data apart from missing data in product descriptions and some delivery-date_at_carrier.
- Data from 4th-Sept-2016 to 17th-Oct-2018 has been provided
- Data spans 27 states covering 8011 cities.

2. In-depth Exploration

How many orders do we have for each order status?

- Most of data provided shows “delivered” status (around 97% of data)

Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario?

purchase_year	num_customers_per_year	num_orders_per_year	total_purchase_value_per_year	num_sellers_per_year	year_on_year_num_customers_change_pct	year_on_year_num_sellers_change_pct	year_on_year_num_orders_change_pct	year_on_year_purchase_value_change_pct
2016	329	329	76528.26	145				
2017	45101	45101	9266612.28	1784	99.2705	91.8722	99.2705	99.1742
2018	54011	54011	11127586.12	2383	16.4966	25.1364	16.4966	16.724

- We can describe in terms of number of orders, total_purchase_amount year-on-year and sellers_added year-on-year.
- From year 2016 to 2017, there has been tremendous increase. Perhaps Target is starting to pick in Brazil in year 2016. 2017-2018 saw around 16% growth in terms of number of orders/ number of sellers added on the platform/ number of orders people is placing on Target website.

On what day of week brazilians customers tend to do online purchasing?

- People tend to shop more on Monday/Tuesdays and least on Saturday/Sunday

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

- Normally people shop more in the afternoon followed by night.
- Feature Extraction: Through order_purchase_timestamp in “orders” dataset extract
    - order_purchase_year
    - order_purchase_month
    - order_purchase_date
    - order_purchase_day
    - order_purchase_dayofweek
    - order_purchase_dayofweek_name
    - order_purchase_hour
    - order_purchase_time_day

Result for above:

```
2  select order_purchase_timestamp
3    , extract(year from order_purchase_timestamp) as purchase_year
4    , extract(month from order_purchase_timestamp) as purchase_month
5    , extract(date from order_purchase_timestamp) as purchase_date
6    , extract(day from order_purchase_timestamp) as purchase_day
7    , extract(week from order_purchase_timestamp) as purchase_dayofweek
8    , FORMAT_DATE('%A', order_purchase_timestamp) as purchase_dayofweekname
9    , extract(hour from order_purchase_timestamp) as purchase_hour
10   , extract(time from order_purchase_timestamp) as purchase_time
11 from `sqltest-353804.ecommerce.orders`
```

Press Alt+F1 for Acces

Query results SAVE RESULTS EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	order_purchase_timestamp	purchase_year	purchase_month	purchase_date	purchase_day	purchase_dayofweek	purchase_dayofweekname	purchase_hour	purchase_time
1	2017-11-25 11:10:33 UTC	2017	11	2017-11-25	25	47	Saturday	11	11:10:33
2	2017-12-05 01:07:58 UTC	2017	12	2017-12-05	5	49	Tuesday	1	01:07:58
3	2017-12-05 01:07:52 UTC	2017	12	2017-12-05	5	49	Tuesday	1	01:07:52
4	2018-02-09 17:21:04 UTC	2018	2	2018-02-09	9	5	Friday	17	17:21:04
5	2017-11-06 13:12:34 UTC	2017	11	2017-11-06	6	45	Monday	13	13:12:34

3. Evolution of E-commerce orders in the Brazil region
    a. Get month on month orders by region

For a given region, growth in number of orders remain more or less constant

customer_state	purchase_year	purchase_month	num_order	lag_order	month_over_month_change_in_num_order_by_state_pct
AC	2017	1	2		
AC	2017	2	3	2	50
AC	2017	3	2	3	-33.33
AC	2017	4	5	2	150
AC	2017	5	8	5	60
AC	2017	6	4	8	-50
AC	2017	7	5	4	25
AC	2017	8	4	5	-20
AC	2017	9	5	4	25
AC	2017	10	6	5	20
AC	2017	11	5	6	-16.67
AC	2017	12	5	5	0
AC	2018	1	6		
AC	2018	2	3	6	-50
AC	2018	3	2	3	-33.33
AC	2018	4	4	2	100
AC	2018	5	2	4	-50
AC	2018	6	3	2	50
AC	2018	7	4	3	33.33
AC	2018	8	3	4	-25
AL	2016	10	2		
AL	2017	1	2		
AL	2017	2	12	2	500
AL	2017	3	10	12	-16.67
AL	2017	4	23	10	130
AL	2017	5	27	23	17.39
AL	2017	6	10	27	-62.96
AL	2017	7	17	10	70

b. Total of customer orders by state

We get more orders from SP and then RJ.

c. Top 10 brazilian cities most no. of orders

We get more orders from Sao-Paolo and Rio-de-Janerio.

d. How are customers distributed in Brazil

Mostly customers are based out of Sao-Paolo and Rio-de-Janerio.

e. City wise number of unique customers

Mostly customers are based out of Sao-Paolo and Rio-de-Janerio.

Conclusion: Most of orders are coming from people based out of Sao-Paolo and Rio-de-Janerio.

4. Impact on Economy

total_amount_sold_2017_Jan_Aug_in_Millions	total_amount_sold_2018_Jan_Aug_in_Millions	pct_increase_2017_2018
3.11300032	7.3859058	137.26

1. Total amount sold in 2017 between Jan to August: 3.11M

2. Total amount sold in 2018 between Jan to august: 7.38M

3. % increase from 2017 to 2018: 137.26

1. Mean & Sum of price by customer state

2. Mean & Sum of freight value by customer state

customer_state	mean_price	sum_price	avg_freight_value	sum_freight_value
AC	173.73	15982.95	40.07	3686.75
AL	180.89	80314.81	35.84	15914.59
AM	135.5	22356.84	33.21	5478.89
AP	164.32	13474.3	34.01	2788.5
BA	134.6	511349.99	26.36	100156.68
CE	153.76	227254.71	32.71	48351.59
DF	125.77	302603.94	21.04	50625.5
ES	121.91	275037.31	22.06	49764.6
GO	126.27	294591.95	22.77	53114.98
MA	145.2	119648.22	38.26	31523.77

5. Analysis on sales, freight and delivery time

a. Top 5 states with highest/lowest average freight value

Highest freight value: PR, PB,RO

Lowest freight value: SP, PR, MG

b. Top 5 states with highest/lowest average time to delivery

Highest time to delivery: RR, AP, AM

Lowest time to delivery: SP, PR, MG

c. Top 5 states where delivery is really fast/ not so fast compared to estimated date

Fast deliver states: SP, PR, MG

6. Payment type analysis

Around 75% of people have done Credit-card purchasing with >90% of people opting <10 number of installments.

7. Actionable Insights (10 points)

- Currently there are only 3-4 states where people are doing more shopping. We can increase penetration to other states as well
- We can have more distribution-centers to reduce delivery time from carrier to customer sites.
- We can offer good discount over weekends/festivals to increase traffic/footprints over Saturdays/Sundays
- We can also see reviews with ratings 1 and try to mitigate those issues like-wrong delivery/delay in delivery.

review_id	order_id	review_score	review_comment_title	review_creation_date	review_answer_timestamp
4cc9dc519327a019f0193e75e34c851d	d1aa2409cda0cbe698a489a0f9364c2a	1	Delay	0003-06-18 00:00:00 UTC	0005-06-18 14:38:00 UTC
4a93de6c66dee67442f7cbd44428f399	37cd892c7a6275aecbe6dfdcee8de6b7	1	Wrong product	0003-06-18 00:00:00 UTC	0006-06-18 22:46:00 UTC
d1612f2df440afddd343d6788628fb28	d34b43bca6cc1cb6d49c06d6090770eb	1	I don't want to evaluate	0003-06-18 00:00:00 UTC	0007-06-18 12:23:00 UTC
9f2b2dee044af4983b7728ff87f43564	0a79d1506c2e0249c9aff8723bcf51e9	1	Delivery delay	0003-06-18 00:00:00 UTC	0006-06-18 15:55:00 UTC
b5e71e87bfbbcd2a826b9610bac99680	058e0a8627dc25b49eae4ce6ca658cb8	1	delivery delay	0003-06-18 00:00:00 UTC	0004-06-18 15:31:00 UTC
6c49563f8fa34a5c1b33919bf9748bed	f119f2d71c96714540bb893bff51e122	1	Dissatisfied	0003-06-18 00:00:00 UTC	0004-06-18 03:13:00 UTC
a3f5d875527ec65adefae2385c9f89c9	e3ba5747e5730389e76a7d30694beeff	1	Bad product	0003-06-18 00:00:00 UTC	0003-06-18 21:48:00 UTC

- We can also tie these order_id to sellers and check for pattern, if any.

8. Recommendations

- Increase penetration to other states as well by doing more marketing (ads etc.)
- Increase the number of distribution-centers by lease/purchase etc.
- Capture “offers and discounts” period in the data so as to investigate high traffic/change in purchase behavior etc.
- We can also see reviews with ratings 1 and try to mitigate those issues like-wrong delivery/delay in delivery.

##

Additional analysis on “Sellers” data:

We want to see who are good sellers and who are getting bad review_scores. This can be defined by mapping sellers to review_score through order_id.

```
with seller_info as

(

 select eo.order_id

   , esi.seller_id

   , date_diff(eo.order_delivered_carrier_date, eo.order_purchase_timestamp,  day) as days_to_deliver

   , eor.review_score

 from `sqltest-353804.ecommerce.order_items` eoi

 inner join `sqltest-353804.ecommerce.sellers` esi

 on eoi.seller_id = esi.seller_id

 inner join `sqltest-353804.ecommerce.order_reviews` eor

 on eoi.order_id = eor.order_id

 inner join `sqltest-353804.ecommerce.orders` eo

 on eoi.order_id = eo.order_id

)

select seller_id

 , avg(days_to_deliver) as avg_days_to_deliver_by_seller

 , avg(review_score) as avg_review_score_by_seller

 , count(order_id) as number_of_orders_by_seller
```

```
from seller_info

group by 1

order by 4 desc

limit 100
```

These are “good” sellers who deliver high volume of orders and have got >average ratings.

seller_id	avg_days_to_deliver_by_seller	avg_review_score_by_seller	number_of_orders_by_seller
6560211a19b47992c3666cc44a7e94c0	1.1063089915548949	3.9094059405940578	2020
4a3ca9315b744ce9f8e9374361493884	2.2702975289964669	3.8039314516129057	1984
1f50f920176fa81dab994f9023523100	3.4772256728778479	3.9824016563147007	1932
cc419e0650a3c5ba77189a1882b7556a	2.3756967670011169	4.0695748205411366	1811
da8622b14eb17ae2831f4ac5b9dab84a	2.2774234693877569	4.0714285714285712	1568
955fee9216a65b617aa5c0531780ce60	1.7538668459986559	4.0517125587642733	1489
1025f0e2d44d7041d6cf58b6550e0bfa	3.72117400419287	3.8497554157931462	1431
7c67e1448b00f6e969d365cea6b010ab	11.662271062271067	3.3482077542062934	1367

However, we want to see those sellers who got very low average review_score:

seller_id	avg_days_to_deliver_by_seller	avg_review_score_by_seller	number_of_orders_by_seller
8d92f3ea807b89465643c219455e7369	33.75	1.0	8
a0e19590a0923cdd0614ea9427713ced	5.0	1.0	7
010da0602d7774602cd1b3f5fb7b709e	21.0	1.0	5
4e42581f08e8cfc7c090f930bac4552a	33.0	1.0	4
fc6295add6f51a0936407ead70c1001d	0.0	1.0	4

We can investigate sellers who got high avg_days_to_deliver.