Business Case: Target SQL

1 Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset

1.1. Get number of rows in the data

RUN SAVE SHARE SCHEDULE

```
select count(*) from target_dataset.customers;
select count(*) from target_dataset.geolocation;
select count(*) from target_dataset.order_items;
select count(*) from target_dataset.order_reviews;
select count(*) from target_dataset.orders;
select count(*) from target_dataset.payments;
select count(*) from target_dataset.products;
select count(*) from target_dataset.sellers;
```

Query results

JOB INFORMATION RESULTS JSON EXECUTION

	f0_
	99441

1.2. Number of null or missing values in a column

RUN SAVE SHARE SCHEDULE MORE

```
select count(*) from target_dataset.order_reviews where review_comment_title is null;
```

Query results SAVE

JOB INFORMATION RESULTS JSON EXECUTION DETAILS

ow	f0_
	87675

1.3. Data type of columns in a table

RUN SAVE SHARE SCHEDULE MORE This query w

```
1 SELECT *
2 FROM target_dataset.INFORMATION_SCHEMA.COLUMNS
```

Press

Query results SAVE RESULTS E

JOB INFORMATION RESULTS JSON EXECUTION DETAILS

Row	table_catalog	table_schema	table_name	column_name	ordinal_position	is_nullable	data_type	is
4	sql-trail-1	target_dataset	order_items	seller_id	4	YES	STRING	N
5	sql-trail-1	target_dataset	order_items	shipping_limit_date	5	YES	TIMESTAMP	N
6	sql-trail-1	target_dataset	order_items	price	6	YES	FLOAT64	N

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

RUN SAVE SHARE SCHEDULE MORE

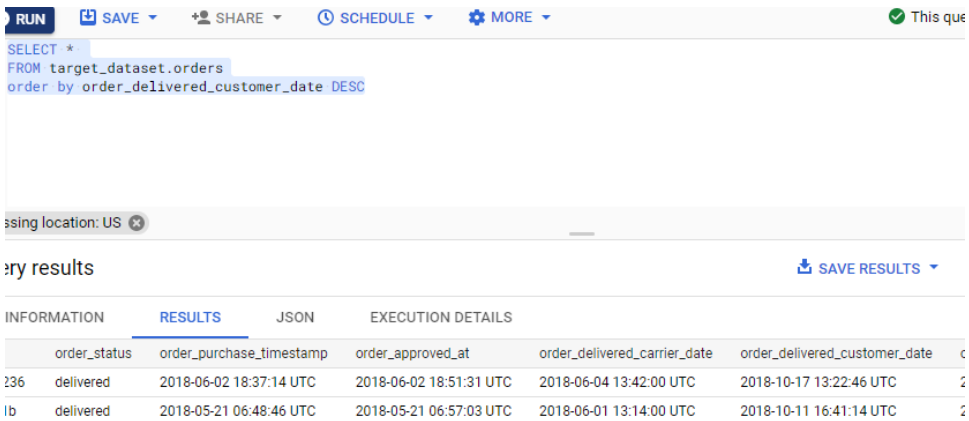
```
1 SELECT *
2 FROM target_dataset.orders
3 order by order_purchase_timestamp
```

Processing location: US

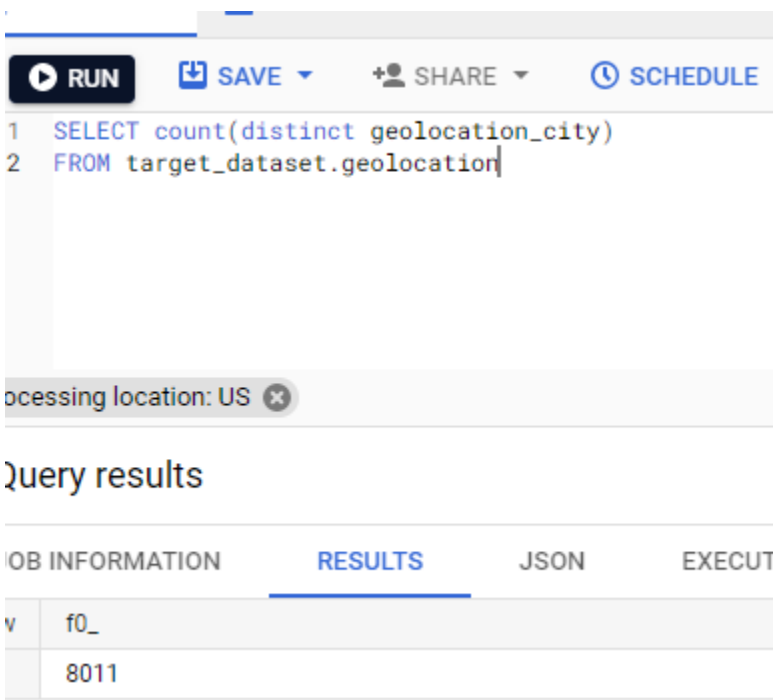
Query results SA

JOB INFORMATION RESULTS JSON EXECUTION DETAILS

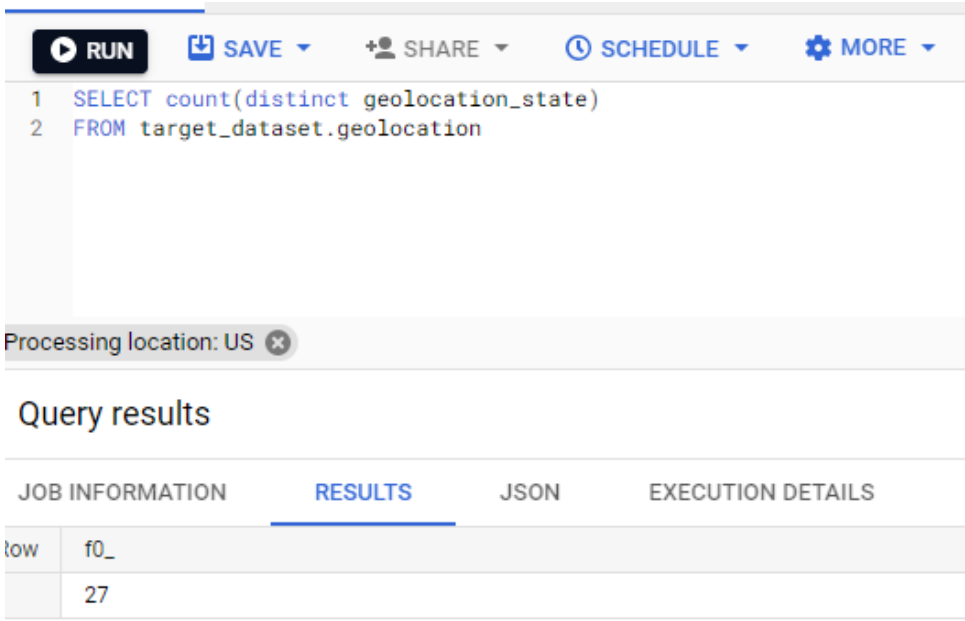
Row	order_id	customer_id	order_status	order_purchase_timestamp
1	2e7a8482f6fb09756ca50c10d7bfc047	08c5351a6aca1c1589a38f244edeee9d	shipped	2016-09-04 21:15:19 UTC
2	e5fa5a7210941f7d56d0208e4e071d35	683c54fc24d40ee9f8a6fc179fd9856c	canceled	2016-09-05 00:15:34 UTC



1.5. Number of cities in our dataset



1.6. Number of states in our dataset



2. In-depth Exploration:

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

RUN SAVE SHARE SCHEDULE

```
SELECT order_status,count(*)
FROM target_dataset.orders
group by order_status
```

Processing location: US

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAI

Row	order_status	f0_
1	created	5
2	shipped	1107
3	approved	2
4	canceled	625
5	invoiced	314
6	delivered	96478
7	processing	301
8	unavailable	609

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

```
select order_purchase_year,
 sum(price) from
(
  select
  EXTRACT (YEAR FROM o.order_purchase_timestamp) as order_purchase_year,
  oi.price as price
  from `sql-trail-1.target_dataset.order_items` oi
  inner join `sql-trail-1.target_dataset.orders` o on oi.order_id=o.order_id
)
group by order_purchase_year
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

ow	order_purchase_year	f0_
	2018	7386050.8000076534
	2017	6155806.9800049355
	2016	49785.920000000326

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

RUN | SAVE | SHARE | SCHEDULE | MORE

```
select day_of_week,count(day_of_week) as count_of_days
from
(select order_purchase_timestamp,
case when extract (DAYOFWEEK from order_purchase_timestamp)=1 then "Sunday"
when extract (DAYOFWEEK from order_purchase_timestamp)=2 then "Monday"
when extract (DAYOFWEEK from order_purchase_timestamp)=3 then "Tuesday"
when extract (DAYOFWEEK from order_purchase_timestamp)=4 then "Wednesday"
when extract (DAYOFWEEK from order_purchase_timestamp)=5 then "Thursday"
when extract (DAYOFWEEK from order_purchase_timestamp)=6 then "Friday"
when extract (DAYOFWEEK from order_purchase_timestamp)=7 then "Saturday"
end as day_of_week
from `sql-trail-1.target_dataset.orders`)
group by day_of_week
order by count(day_of_week) desc
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	day_of_week	count_of_days
1	Monday	16196
2	Tuesday	15963
3	Wednesday	15552
4	Thursday	14761
5	Friday	14122
6	Sunday	11960
7	Saturday	10887

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

Dawn-4-6

Morning- 6-12

Afternoon - 12-19

Night-19-4

```
select time_of_day,count(time_of_day) from (
SELECT order_purchase_timestamp,
case
      when EXTRACT(HOUR FROM order_purchase_timestamp)>4 and EXTRACT(HOUR FROM order_purchase_timestamp)<=6 then 'Dawn'
      when EXTRACT(HOUR FROM order_purchase_timestamp)>6 and EXTRACT(HOUR FROM order_purchase_timestamp)<=12 then 'Morning'
      when EXTRACT(HOUR FROM order_purchase_timestamp)>12 and EXTRACT(HOUR FROM order_purchase_timestamp)<=19 then 'Afternoon'
      when EXTRACT(HOUR FROM order_purchase_timestamp)>19 then 'Night'
      when EXTRACT(HOUR FROM order_purchase_timestamp)<=4 then 'Night'
end as time_of_day
FROM (
      SELECT DATETIME(order_purchase_timestamp, "America/Fortaleza") as order_purchase_timestamp from `sql-trail-1.target_dataset.orders`
)
)
group by time_of_day
order by count(time_of_day) desc
```

Press Alt+F1 for Acc

Query results

SAVE RESULTS EXPLORE DAT

JOB INFORMATION RESULTS JSON EXECUTION DETAILS

time_of_day	f0_
Afternoon	42802
Morning	38291
Night	10596
Dawn	7752

2.5.1 Through order_purchase_timestamp in “orders” dataset extract order_purchase_year

```
select extract (year from order_purchase_timestamp) from `sql-trail-1.target_dataset.orders`
```

Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS

f0_
2017
2017
2017
2018
2017
2017

Results per page:

2.5.2 order_purchase_month

```
1  select order_purchase_timestamp,extract (month from order_purchase_timestamp) from `sql-trail-1.target_dataset.orders`
```

Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

ow	order_purchase_timestamp	f0_
	2018-01-05 15:30:15 UTC	1
	2018-01-21 21:52:15 UTC	1
	2018-01-08 21:43:10 UTC	1
	2018-01-13 15:41:34 UTC	1
	2018-01-16 01:25:39 UTC	1
	2018-01-16 17:11:07 UTC	1
	2018-01-05 22:50:53 UTC	1
	2018-01-25 16:50:50 UTC	1
	2018-01-28 18:08:56 UTC	1

2.5.3 order_purchase_date

RUN | SAVE | SHARE | SCHEDULE | MORE | This que

```
1  select order_purchase_timestamp,extract (date from order_purchase_timestamp) from `sql-trail-1.target_dataset.orders`
```

Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	order_purchase_timestamp	f0_
1	2017-12-05 01:07:58 UTC	2017-12-05
2	2017-12-05 01:07:52 UTC	2017-12-05
3	2018-02-09 17:21:04 UTC	2018-02-09
4	2017-11-06 13:12:34 UTC	2017-11-06
5	2017-04-20 12:45:34 UTC	2017-04-20
6	2017-07-13 11:03:05 UTC	2017-07-13
7	2017-07-11 13:36:30 UTC	2017-07-11
8	2017-07-29 18:05:07 UTC	2017-07-29
9	2017-07-13 10:02:47 UTC	2017-07-13

Results per page: 50 1 – 50 of 9

2.5.4 order_purchase_day

```
1 select order_purchase_timestamp,extract (day from order_purchase_timestamp) from `sql-trail-1.target_dataset.orders`
```

Query results

SAVE RESULT

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	order_purchase_timestamp	f0_
42	2018-03-01 11:42:23 UTC	1
43	2018-02-01 16:49:17 UTC	1
44	2018-02-01 12:21:42 UTC	1
45	2018-05-01 11:45:29 UTC	1
46	2018-03-01 15:20:13 UTC	1
47	2017-03-01 11:26:37 UTC	1
48	2018-08-01 16:39:52 UTC	1
49	2018-03-01 22:40:51 UTC	1
50	2018-03-01 18:03:33 UTC	1

Results per page: 50 1 -

2.5.5 order_purchase_dayofweek

```
1 select order_purchase_timestamp,extract (DAYOFWEEK from order_purchase_timestamp) from `sql-trail-1.target_dataset.orders`
```

Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

ow	order_purchase_timestamp	f0_
	2018-05-20 18:58:04 UTC	1
	2018-01-21 21:52:15 UTC	1
	2017-12-17 13:34:57 UTC	1
	2017-04-09 23:15:16 UTC	1
	2018-01-28 18:08:56 UTC	1
	2018-04-29 13:02:12 UTC	1
	2018-05-06 07:09:28 UTC	1
	2017-06-11 19:08:12 UTC	1
	2017-06-18 00:27:16 UTC	1

Results per page: 50 1 – 50 of 994

2.5.6 order_purchase_dayofweek_name

```sql
select order_purchase_timestamp,
case when extract (DAYOFWEEK from order_purchase_timestamp)=1 then "Sunday"
when extract (DAYOFWEEK from order_purchase_timestamp)=2 then "Monday"
when extract (DAYOFWEEK from order_purchase_timestamp)=3 then "Tuesday"
when extract (DAYOFWEEK from order_purchase_timestamp)=4 then "Wednesday"
when extract (DAYOFWEEK from order_purchase_timestamp)=5 then "Thursday"
when extract (DAYOFWEEK from order_purchase_timestamp)=6 then "Friday"
when extract (DAYOFWEEK from order_purchase_timestamp)=7 then "Saturday"
end as day_of_week
from `sql-trail-1.target_dataset.orders`
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	order_purchase_timestamp	day_of_week
99401	2017-07-05 13:55:19 UTC	Wednesday
99402	2017-07-19 21:53:57 UTC	Wednesday
99403	2017-03-15 13:20:25 UTC	Wednesday
99404	2018-08-01 03:54:30 UTC	Wednesday
99405	2018-03-28 10:43:54 UTC	Wednesday
99406	2017-10-18 19:11:27 UTC	Wednesday

2.5.7 order_purchase_hour

RUN | SAVE | SHARE | SCHEDULE | MORE | This q

```sql
select order_purchase_timestamp,extract (hour from order_purchase_timestamp) from `sql-trail-1.target_dataset.orders`
```

Query results | SAVE RESULTS

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	order_purchase_timestamp	f0_
1	2018-03-05 03:47:11 UTC	3
2	2018-01-25 03:24:05 UTC	3
3	2018-01-18 04:37:44 UTC	4
4	2017-02-08 05:56:31 UTC	5
5	2018-04-21 03:25:21 UTC	3
6	2017-04-03 03:01:46 UTC	3
7	2017-12-01 03:16:01 UTC	3
8	2018-08-12 03:44:50 UTC	3
9	2017-03-25 05:12:19 UTC	5

2.5.8 order_purchase_time_day

RUN SAVE SHARE SCHEDULE MORE

```
select order_purchase_timestamp,extract (time from order_purchase_timestamp) from `sql-trail-1.target_dataset.orders`
```

Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	order_purchase_timestamp	f0_
1	2017-11-25 11:10:33 UTC	11:10:33
2	2017-12-05 01:07:58 UTC	01:07:58
3	2017-12-05 01:07:52 UTC	01:07:52
4	2018-02-09 17:21:04 UTC	17:21:04
5	2017-11-06 13:12:34 UTC	13:12:34
6	2017-04-20 12:45:34 UTC	12:45:34
7	2017-07-13 11:03:05 UTC	11:03:05
8	2017-07-11 13:36:30 UTC	13:36:30
9	2017-07-29 18:05:07 UTC	18:05:07

Results per page: 50 1 –

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

3.1. Get month on month orders by region

```
select extract(month FROM o.order_purchase_timestamp) as month_purchase,
extract(year FROM o.order_purchase_timestamp) as year_purchase,
c.customer_zip_code_prefix as customer_zip_code_prefix,
count(distinct o.order_id) as disc_orderid
from target_dataset.orders o inner join target_dataset.customers c
on o.customer_id=c.customer_id
group by c.customer_zip_code_prefix,month_purchase,year_purchase
order by month_purchase,year_purchase
```

ıery results

B INFORMATION | RESULTS | JSON | EXECUTION DETAILS

month_purchase	year_purchase	customer_zip_code_prefix	disc_orderid
1	2017	85660	2
1	2017	35930	1
1	2017	18550	1
1	2017	20250	2
1	2017	4679	1
1	2017	68560	1
1	2017	33913	1

3.2.Total of customer orders by state

```sql
select c.customer_state,count(o.customer_id)
from `sql-trail-1.target_dataset.orders` o
inner join `sql-trail-1.target_dataset.customers` c
on o.customer_id=c.customer_id
group by c.customer_state
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

ow	customer_state	f0_
	RJ	12852
	RS	5466
	SP	41746
	DF	2140
	PR	5045
	MT	907
	MA	747
	AL	413
	MG	11635
0	PE	1652

3.3. Top 10 brazilian cities most no. of orders

```
select c.customer_city,count(c.customer_id)
from `sql-trail-1.target_dataset.orders` o
inner join `sql-trail-1.target_dataset.customers` c
on o.customer_id=c.customer_id
group by c.customer_city
order by count(distinct c.customer_id) desc
limit 10
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_city	f0_
1	sao paulo	15540
2	rio de janeiro	6882
3	belo horizonte	2773
4	brasilia	2131
5	curitiba	1521
6	campinas	1444
7	porto alegre	1379
8	salvador	1245
9	guarulhos	1189
10	sao bernardo do campo	938

3.4. How are customers distributed in Brazil

```
select customer_state,customer_city,count(distinct customer_unique_id) as customers_count
from target_dataset.customers
group by customer_city,customer_state
order by customer_state
```

ıery results SAVE RES

B INFORMATION | RESULTS | JSON | EXECUTION DETAILS

customer_state	customer_city	customers_count
AC	xapuri	2
AC	brasileia	1
AC	porto acre	1
AC	rio branco	66
AC	manoel urbano	1

3.5 City wise number of unique customers

```
select c.customer_city,count(distinct c.customer_id)
from `sql-trail-1.target_dataset.orders` o
inner join `sql-trail-1.target_dataset.customers` c
on o.customer_id=c.customer_id
group by c.customer_city
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_city	f0_
1	rio de janeiro	6882
2	sao leopoldo	105
3	general salgado	7
4	brasilia	2131
5	paranavai	47
6	cuiaba	248
7	sao luis	353
8	maceio	247
9	hortolandia	145

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

4.1 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)

```
with order_item_plus_orders AS
(
  select o.order_id as order_id,oi.price as price,oi.freight_value as freight_value,
  extract (MONTH from o.order_purchase_timestamp) as purchase_month,
  extract (YEAR from o.order_purchase_timestamp) as purchase_year
  from `sql-trail-1.target_dataset.order_items` oi
  inner join `sql-trail-1.target_dataset.orders` o on oi.order_id=o.order_id
)

select sum(price)/COUNT(order_id) as price_per_order,
sum(freight_value)/COUNT(order_id) as freight_per_order
from order_item_plus_orders
group by purchase_month,purchase_year
```

Query results SAVE

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

price_per_order	freight_per_order
126.27005358150556	23.014778623801426
117.92029939294153	20.505262141280323
122.35762572534202	19.371327369438863
122.22722661769379	22.259508335687997

4.2 Answer the following questions:

4.2.1. Total amount sold in 2017 between Jan to August

```
with order_item_plus_orders AS
(
  select o.order_id as order_id,oi.price as price,oi.freight_value as freight_value,o.customer_id as customer_id,
  extract (MONTH from o.order_purchase_timestamp) as purchase_month,
  extract (YEAR from o.order_purchase_timestamp) as purchase_year
  from `sql-trail-1.target_dataset.order_items` oi
  inner join `sql-trail-1.target_dataset.orders` o on oi.order_id=o.order_id
)

select SUM(price) as amount_sold from order_item_plus_orders where purchase_month<9 and purchase_year=2017

-- select sum(price)/COUNT(order_id) as price_per_order
```

Press Alt+F1

Query results SAVE RESULTS EXPLOR

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

amount_sold
3113000.3199994233

4.2.2. Total amount sold in 2018 between Jan to august

RUN SAVE SHARE SCHEDULE MORE This query will process 8

```google_sql
with order_item_plus_orders AS
(
  select o.order_id as order_id,oi.price as price,oi.freight_value as freight_value,o.customer_id as customer_id,
  extract (MONTH from o.order_purchase_timestamp) as purchase_month,
  extract (YEAR from o.order_purchase_timestamp) as purchase_year
  from `sql-trail-1.target_dataset.order_items` oi
  inner join `sql-trail-1.target_dataset.orders` o on oi.order_id=o.order_id
)

select SUM(price) as amount_sold from order_item_plus_orders where purchase_month<9 and purchase_year=2018


-- select sum(price)/COUNT(order_id) as price_per_order
```

Press Alt+F1 for Ac

Query results SAVE RESULTS EXPLORE DA

JOB INFORMATION RESULTS JSON EXECUTION DETAILS

Row	amount_sold
1	7385905.8000076516

4.2.3. % increase from 2017 to 2018

```google_sql
with order_item_plus_orders AS
(
  select o.order_id as order_id,oi.price as price,oi.freight_value as freight_value,o.customer_id as customer_id,
  extract (MONTH from o.order_purchase_timestamp) as purchase_month,
  extract (YEAR from o.order_purchase_timestamp) as purchase_year
  from `sql-trail-1.target_dataset.order_items` oi
  inner join `sql-trail-1.target_dataset.orders` o on oi.order_id=o.order_id
)

select (round((amount_sold-lg)/lg,2))*100 as year_on_year from(
  select purchase_year,
  SUM(price) as amount_sold,
  LAG(SUM(price),1,0) over(order by purchase_year) as lg
  from order_item_plus_orders where purchase_year=2017 or purchase_year=2018
  group by purchase_year
) where purchase_year=2018
```

Press Alt+F1 for Ac

Query results SAVE RESULTS EXPLORE DA

JOB INFORMATION RESULTS JSON EXECUTION DETAILS

ow	year_on_year
	20.0

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

4.3.1. Mean & Sum of price by customer state

```
with order_item_plus_orders AS
(
  select o.order_id as order_id,oi.price as price,oi.freight_value as freight_value,o.customer_id as customer_id,
  extract (MONTH from o.order_purchase_timestamp) as purchase_month,
  extract (YEAR from o.order_purchase_timestamp) as purchase_year
  from `sql-trail-1.target_dataset.order_items` oi
  inner join `sql-trail-1.target_dataset.orders` o on oi.order_id=o.order_id
)

select c.customer_state,SUM(oip.price) as sum_price,AVG(oip.price) from order_item_plus_orders oip inner join `sql-trail-1.
target_dataset.customers` c on oip.customer_id=c.customer_id
group by c.customer_state
```

Press Alt+F1 for Accessibilit

ery results

SAVE RESULTS EXPLORE DATA

INFORMATION RESULTS JSON EXECUTION DETAILS

customer_state	sum_price	f0_
SP	5202955.0500027407	109.65362915972931
RJ	1824092.6699996467	125.11781809451907
PR	683083.76000003726	119.00413937282218
SC	520553.34000002244	124.65357758620696

4.3.2. Mean & Sum of freight value by customer state

```
with order_item_plus_orders AS
(
  select o.order_id as order_id,oi.price as price,oi.freight_value as freight_value,o.customer_id as customer_id,
  extract (MONTH from o.order_purchase_timestamp) as purchase_month,
  extract (YEAR from o.order_purchase_timestamp) as purchase_year
  from `sql-trail-1.target_dataset.order_items` oi
  inner join `sql-trail-1.target_dataset.orders` o on oi.order_id=o.order_id
)

select c.customer_state,SUM(oip.freight_value) as freight_value_sum,AVG(oip.freight_value) as freight_value_avg from
order_item_plus_orders oip inner join `sql-trail-1.target_dataset.customers` c on oip.customer_id=c.customer_id
group by c.customer_state
```

Press Alt+F1 for Access

Query results

SAVE RESULTS EXPLORE DATA

JOB INFORMATION RESULTS JSON EXECUTION DETAILS

customer_state	freight_value_sum	freight_value_avg
MT	29715.430000000102	28.1662843601896
MA	31523.770000000033	38.25700242718446
AL	15914.589999999991	35.843671171171152

Results per page: 50 1 – 27 of 27

5. Analysis on sales, freight and delivery time

5.1. Calculating days between purchasing, delivering and estimated delivery

```
with order_item_plus_orders AS
(
  select
  EXTRACT (DATE FROM o.order_estimated_delivery_date) as estimate_date,
  EXTRACT (DATE FROM o.order_delivered_carrier_date) as delivery_date,
  EXTRACT (DATE FROM o.order_purchase_timestamp) as purchase_date,
  o.order_id as order_id,oi.price as price,oi.freight_value as freight_value,o.customer_id as customer_id,
  from `sql-trail-1.target_dataset.order_items` oi
  inner join `sql-trail-1.target_dataset.orders` o on oi.order_id=o.order_id
)
select
    delivery_date,purchase_date,estimate_date,
    DATE_DIFF(delivery_date,purchase_date,DAY) AS delivery_time,
    DATE_DIFF(estimate_date,delivery_date,DAY) as estimate_actual_mismatch
  from order_item_plus_orders
```

Press Alt+F

ery results

SAVE RESULTS EXPL

3 INFORMATION RESULTS JSON EXECUTION DETAILS

delivery_date	purchase_date	estimate_date	delivery_time	estimate_actual_mismatch
2018-07-31	2018-07-11	2018-08-01	20	1
2017-12-18	2017-12-09	2018-01-29	9	42

Results per page: 50 1 – 50 of 112650

5.2. Create columns:

time_to_delivery = order_purchase_timestamp-order_delivered_customer_date

diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date

```
with order_item_plus_orders AS
(
  select
  EXTRACT (DATE FROM o.order_purchase_timestamp) as order_purchase_timestamp,
  EXTRACT (DATE FROM o.order_delivered_customer_date) as order_delivered_customer_date,
  EXTRACT (DATE FROM o.order_estimated_delivery_date) as order_estimated_delivery_date,
  EXTRACT (DATE FROM o.order_purchase_timestamp) as purchase_date,
  o.order_id as order_id,oi.price as price,oi.freight_value as freight_value,o.customer_id as customer_id,
  from `sql-trail-1.target_dataset.order_items` oi
  inner join `sql-trail-1.target_dataset.orders` o on oi.order_id=o.order_id
)
select order_purchase_timestamp,order_delivered_customer_date,order_estimated_delivery_date,
    DATE_DIFF(order_purchase_timestamp,order_delivered_customer_date,DAY) AS time_to_delivery,
    DATE_DIFF(order_estimated_delivery_date,order_delivered_customer_date,DAY) as diff_estimated_delivery
  from order_item_plus_orders
```

Press Alt+F1 for Acc

Query results

SAVE RESULTS EXPLORE DAT

JOB INFORMATION RESULTS JSON EXECUTION DETAILS

ow	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	time_to_delivery	diff_estimated_delivery
	2016-10-10	2016-11-24	2016-12-02	-45	8
	2018-02-09	2018-05-04	2018-03-09	-84	-56

Results per page: 50 1 – 50 of 112650

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

```
with order_item_plus_orders AS
(
  select
  o.customer_id as c_id,
  EXTRACT (DATE FROM o.order_purchase_timestamp) as order_purchase_timestamp,
  EXTRACT (DATE FROM o.order_delivered_customer_date) as order_delivered_customer_date,
  EXTRACT (DATE FROM o.order_estimated_delivery_date) as order_estimated_delivery_date,
  EXTRACT (DATE FROM o.order_purchase_timestamp) as purchase_date,
  o.order_id as order_id,oi.price as price,oi.freight_value as freight_value,o.customer_id as customer_id,
  from `sql-trail-1.target_dataset.order_items` oi
  inner join `sql-trail-1.target_dataset.orders` o on oi.order_id=o.order_id
)
select
AVG(freight_value) as mean_freight_value,
AVG(time_to_delivery) as mean_tod,
AVG(diff_estimated_delivery) as mean_ded
 from (
  select c.customer_state as customer_state,op.freight_value as freight_value,
    DATE_DIFF(op.order_purchase_timestamp,op.order_delivered_customer_date,DAY) AS time_to_delivery,
    DATE_DIFF(op.order_estimated_delivery_date,op.order_delivered_customer_date,DAY) as diff_estimated_delivery,
   from order_item_plus_orders op
   inner join target_dataset.customers c on op.c_id=c.customer_id
) group by customer_state
```

ery results

INFORMATION | RESULTS | JSON | EXECUTION DETAILS

mean_freight_value	mean_tod	mean_ded
28.1662843601896	-17.907425265188039	14.571841851494709
38.25700242718446	-21.589999999999982	9.9062499999999929
35.843671171171152	-24.447306791569098	8.73536299765808
15.147275390419248	-8.66225265379071	11.207910772344571
20.630166806306541	-11.920724626461224	13.342649221955588

5. 4. Sort the data to get the following:

5. 4 .a. Top 5 states with highest/lowest average freight value

```
AVG(freight_value) as mean_freight_value,
AVG(time_to_delivery) as mean_tod,
AVG(diff_estimated_delivery) as mean_ded
 from (
  select c.customer_state as customer_state,op.freight_value as freight_value,
    DATE_DIFF(op.order_purchase_timestamp,op.order_delivered_customer_date,DAY) AS time_to_delivery,
    DATE_DIFF(op.order_estimated_delivery_date,op.order_delivered_customer_date,DAY) as diff_estimated_delivery,
   from order_item_plus_orders op
   inner join target_dataset.customers c on op.c_id=c.customer_id
) group by customer_state
order by AVG(freight_value)
limit 5
```

Press Alt+F1 for

Query results

SAVE RESULTS | EXPLORE

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	mean_freight_value	mean_tod	mean_ded
1	15.147275390419132	-8.6622526537905529	11.207910772344583
2	20.531651567944269	-11.893078420959434	13.48610373517436
3	20.630166806306651	-11.920724626461217	13.342649221955513
4	20.960923931682483	-15.074791460483519	12.014774494556724
5	21.041354945968422	-12.893842887473463	12.200424628450106

```
select
AVG(freight_value) as mean_freight_value,
AVG(time_to_delivery) as mean_tod,
AVG(diff_estimated_delivery) as mean_ded
 from (
  select c.customer_state as customer_state,op.freight_value as freight_value,
    DATE_DIFF(op.order_purchase_timestamp,op.order_delivered_customer_date,DAY) AS time_to_delivery,
    DATE_DIFF(op.order_estimated_delivery_date,op.order_delivered_customer_date,DAY) as diff_estimated_del
   from order_item_plus_orders op
   inner join target_dataset.customers c on op.c_id=c.customer_id
) group by customer_state
order by AVG(freight_value) desc
limit 5
```

Pre

Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	mean_freight_value	mean_tod	mean_ded
	42.984423076923093	-28.173913043478258	18.326086956521742
2	42.723803986710941	-20.546075085324258	13.037542662116042
3	41.069712230215842	-19.655677655677675	20.040293040293058
4	40.073369565217405	-20.681318681318679	20.978021978021971

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

```
select customer_state,
AVG(freight_value) as mean_freight_value,
AVG(time_to_delivery) as mean_tod,
AVG(diff_estimated_delivery) as mean_ded
 from (
  select c.customer_state as customer_state,op.freight_value as freight_value,
    DATE_DIFF(op.order_purchase_timestamp,op.order_delivered_customer_date,DAY) AS time_to_delivery,
    DATE_DIFF(op.order_estimated_delivery_date,op.order_delivered_customer_date,DAY) as diff_estimated_delivery,
   from order_item_plus_orders op
   inner join target_dataset.customers c on op.c_id=c.customer_id
) group by customer_state
order by AVG(time_to_delivery)
limit 5
```

Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	mean_freight_value	mean_tod	mean_ded
	AP	34.006097560975626	-28.222222222222221	18.395061728395063
	RR	42.984423076923072	-28.173913043478262	18.326086956521738
	AM	33.205393939393922	-26.337423312883434	19.932515337423304
	AL	35.843671171171167	-24.447306791569076	8.735362997658088
	PA	35.832685185185213	-23.702087286527529	14.250474383301707

```
select customer_state,
AVG(freight_value) as mean_freight_value,
AVG(time_to_delivery) as mean_tod,
AVG(diff_estimated_delivery) as mean_ded
 from (
  select c.customer_state as customer_state,op.freight_value as freight_value,
    DATE_DIFF(op.order_purchase_timestamp,op.order_delivered_customer_date,DAY) AS time_to_delivery,
    DATE_DIFF(op.order_estimated_delivery_date,op.order_delivered_customer_date,DAY) as diff_estimated_delivery,
   from order_item_plus_orders op
   inner join target_dataset.customers c on op.c_id=c.customer_id
) group by customer_state
order by AVG(time_to_delivery) desc
limit 5
```

ıery results

SAVE RESUL

3 INFORMATION | RESULTS | JSON | EXECUTION DETAILS

customer_state	mean_freight_value	mean_tod	mean_ded
SP	15.147275390419132	-8.6622526537905529	11.207910772344583
PR	20.531651567944269	-11.893078420959434	13.48610373517436
MG	20.630166806306651	-11.920724626461217	13.342649221955513
DF	21.041354945968422	-12.893842887473463	12.200424628450106
SC	21.470368773946323	-14.95021961932652	11.572718399219173

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

```
select customer_state,
AVG(freight_value) as mean_freight_value,
AVG(time_to_delivery) as mean_tod,
AVG(diff_estimated_delivery) as mean_ded
 from (
  select c.customer_state as customer_state,op.freight_value as freight_value,
    DATE_DIFF(op.order_purchase_timestamp,op.order_delivered_customer_date,DAY) AS time_to_delivery,
    DATE_DIFF(op.order_estimated_delivery_date,op.order_delivered_customer_date,DAY) as diff_estimated_delivery,
   from order_item_plus_orders op
   inner join target_dataset.customers c on op.c_id=c.customer_id
) group by customer_state
order by AVG(diff_estimated_delivery)
limit 5
```

Query results

SAVE RE

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

ow	customer_state	mean_freight_value	mean_tod	mean_ded
	AL	35.843671171171167	-24.447306791569076	8.735362997658088
	MA	38.257002427184474	-21.589999999999982	9.9062500000000053
	SE	36.653168831168855	-21.418666666666663	10.00266666666667
	ES	22.058776595744732	-15.587415730337076	10.646292134831453
	BA	26.36395893656228	-19.192506109150163	10.982622861797456

```
select customer_state,
AVG(freight_value) as mean_freight_value,
AVG(time_to_delivery) as mean_tod,
AVG(diff_estimated_delivery) as mean_ded
 from (
  select c.customer_state as customer_state,op.freight_value as freight_value,
    DATE_DIFF(op.order_purchase_timestamp,op.order_delivered_customer_date,DAY) AS time_to_delivery,
    DATE_DIFF(op.order_estimated_delivery_date,op.order_delivered_customer_date,DAY) as diff_estimated_delivery,
   from order_item_plus_orders op
   inner join target_dataset.customers c on op.c_id=c.customer_id
) group by customer_state
order by AVG(diff_estimated_delivery) desc
limit 5
```

Query results

SAVE RES

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

ow	customer_state	mean_freight_value	mean_tod	mean_ded
	AC	40.073369565217405	-20.681318681318679	20.978021978021971
	RO	41.069712230215842	-19.655677655677675	20.040293040293058
	AM	33.205393939393936	-26.337423312883427	19.932515337423315
	AP	34.006097560975618	-28.222222222222218	18.395061728395063
	RR	42.984423076923093	-28.173913043478258	18.326086956521742

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

6.1. Count of orders for different payment types

```
select p.payment_type,count(o.order_id)
from `sql-trail-1.target_dataset.payments` p
inner join `sql-trail-1.target_dataset.orders` o
on p.order_id=o.order_id
group by p.payment_type
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	payment_type	f0_
1	credit_card	76795
2	voucher	5775
3	not_defined	3
4	debit_card	1529
5	UPI	19784

6.2. Distribution of payment installments and count of orders

```
select pur_year,
count(distinct order_id) as order_id_count,
count(payment_installments) as inst_dist
from
(
  select
  extract (year from o.order_purchase_timestamp) as pur_year,
  o.order_id as order_id,p.payment_type as payment_type,p.payment_value as payment_value,p.payment_installments as payment_installments
  from `sql-trail-1.target_dataset.orders` o
  inner join `sql-trail-1.target_dataset.payments` p
  on o.order_id=p.order_id
)
group by pur_year
```

Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

ow	pur_year	order_id_count	inst_dist
	2018	54011	56015
	2017	45101	47525
	2016	328	346

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

```
select pur_month,pur_year,payment_type,
count(distinct order_id) as order_count
from
(
  select extract (month from o.order_purchase_timestamp) as pur_month,
  extract (year from o.order_purchase_timestamp) as pur_year,
  o.order_id as order_id,p.payment_type as payment_type,p.payment_value as payment_value
  from `sql-trail-1.target_dataset.orders` o
  inner join `sql-trail-1.target_dataset.payments` p
  on o.order_id=p.order_id
)
group by pur_month,pur_year,payment_type
order by pur_month,pur_year
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

ow	pur_month	pur_year	payment_type	order_count
	1	2017	credit_card	582
	1	2017	UPI	197
	1	2017	voucher	33
	1	2017	debit_card	9
	1	2018	credit_card	5511
	1	2018	UPI	1518
	1	2018	voucher	304