1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset
    1. Data type of columns in a table

```
SELECT table_name, column_name,data_type FROM scaler-case1.Target.INFORMATION_SCHEMA.COLUMNS
```

Query results

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

Row	table_name	column_name	data_type
1	order_items	order_id	STRING
2	order_items	order_item_id	INT64
3	order_items	product_id	STRING
4	order_items	seller_id	STRING
5	order_items	shipping_limit_date	TIMESTAMP
6	order_items	price	FLOAT64
7	order_items	freight_value	FLOAT64
8	sellers	seller_id	STRING
9	sellers	seller_zip_code_prefix	INT64
10	sellers	seller_city	STRING
11	sellers	seller_state	STRING

    2. Time period for which the data is given

RUN | SAVE | SHARE | SCHEDULE | MORE

```
# Time Period for which the date is given
#Get max order_estimated_delivery_date and min order_purchase_timestamp
#Taken in DAYS,WEEK,MONTH,YEAR
with limits as (
SELECT max(date(order_estimated_delivery_date)) start_date, min(date(order_purchase_timestamp)) end_date FROM `scaler-case1.Target.orders`)

select *,date_diff(start_date, end_date, day) DAYS,date_diff(start_date, end_date, week) WEEK,
date_diff(start_date, end_date, month) MONTH,date_diff(start_date, end_date, year) YEAR from limits
```

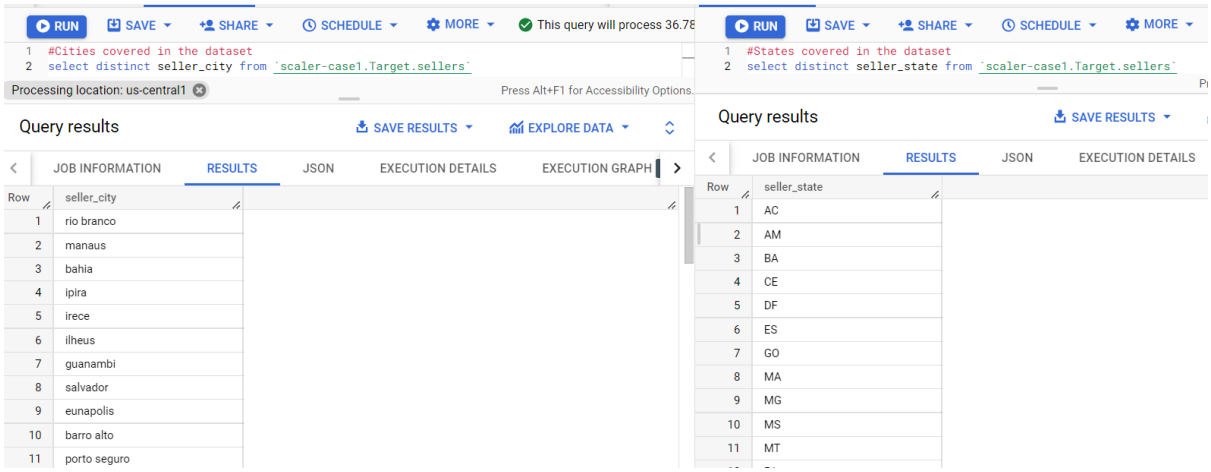
Query results

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

Row	start_date	end_date	DAYS	WEEK	MONTH	YEAR
1	2018-11-12	2016-09-04	799	114	26	2

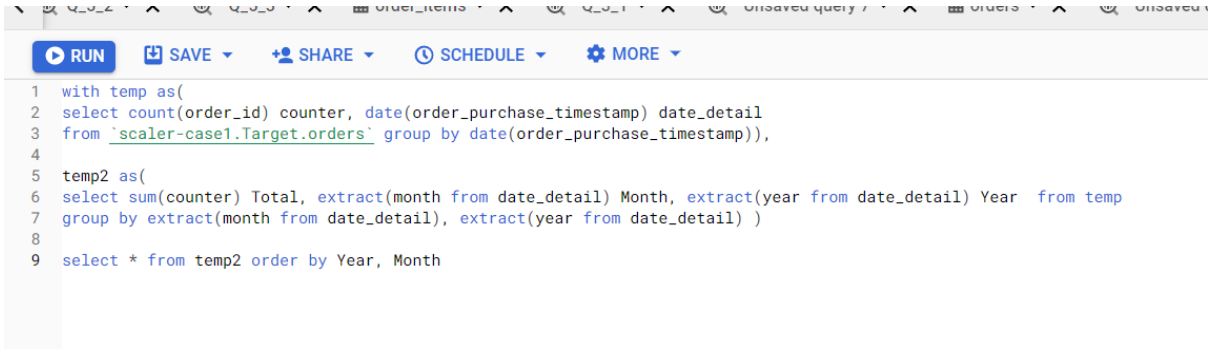
    3. Cities and States covered in the dataset

    Ans:



2. In-depth Exploration:
    1. 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?

Yes, We can see a growing trend on e-commerce in Brazil.



Query results

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

Row	Total	Month	Year
1	4	9	2016
2	324	10	2016
3	1	12	2016
4	800	1	2017
5	1780	2	2017
6	2682	3	2017
7	2404	4	2017
8	3700	5	2017
9	3245	6	2017
10	4026	7	2017
11	4331	8	2017
12	4285	9	2017
13	4631	10	2017
14	7544	11	2017
15	5673	12	2017
16	7269	1	2018
17	6728	2	2018

RUN SAVE SHARE SCHEDULE MORE This query w

```sql
with temp as(
select count(order_id) counter, date(order_purchase_timestamp) date_detail
from `scaler-case1.Target.orders` group by date(order_purchase_timestamp)),

temp3 as(
select sum(counter) Total, extract(quarter from date_detail) Quarter from temp
group by extract(quarter from date_detail))


select *, sum(Total) over (partition by Quarter) Seasonal from temp3 order by Quarter
```

Pres

Query results SAVE RESULTS

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

Row	Total	Quarter	Seasonal
1	26470	1	26470
2	29328	2	29328
3	25466	3	25466
4	18177	4	18177

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

```sql
select TIME_PERIOD, count(TIME_PERIOD) Order_Count from
  (select
  case when extract(hour from order_purchase_timestamp) >= 4 and extract(hour from order_purchase_timestamp) < 7 then "DAWN"
       when extract(hour from order_purchase_timestamp) >= 7 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"
        else "NIGHT"
        end as TIME_PERIOD
        from `scaler-case1.Target.orders`
  )
  group by TIME_PERIOD
```

Press Alt+F1 for Ac

Query results SAVE RESULTS EXPLORE DA

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

Row	TIME_PERIOD	Order_Count
1	MORNING	21738
2	NIGHT	32677
3	AFTERNOON	44130
4	DAWN	896

3. Evolution of E-commerce orders in the Brazil region:
    1. Get month on month orders by region, states

RUN SAVE SHARE SCHEDULE MORE

```sql
with temp as(
select customer_id, order_id,date(order_purchase_timestamp) date_detail from `scaler-case1.Target.orders`
),

temp2 as(
select  t.order_id, c.customer_state, date_detail from temp t inner join `scaler-case1.Target.customers` c
on t.customer_id = c.customer_id
),

temp3 as(
select count(order_id) counter, extract(year from date_detail) Year, extract(month from date_detail) Month, customer_state from temp2
group by extract(month from date_detail),extract(year from date_detail),customer_state
)

select * from temp3 order by Year, Month
```

Query results

SAVE RESULTS | EX

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

Row	counter	Year	Month	customer_state
1	1	2016	9	RR
2	1	2016	9	RS
3	2	2016	9	SP
4	113	2016	10	SP
5	24	2016	10	RS
6	56	2016	10	RJ
7	3	2016	10	MT
8	9	2016	10	GO
9	40	2016	10	MG
10	8	2016	10	CE
11	11	2016	10	SC
12	2	2016	10	AL
13	4	2016	10	BA
14	7	2016	10	PE
15	4	2016	10	ES

Load more

Results per page: 50 | 1 – 50 of 565

2. How are customers distributed in Brazil

RUN | SAVE | SHARE | SCHEDULE | MORE

```
1 #How are customers distributed in Brazil
2 select distinct count(customer_id) customers_count, customer_city from `scaler-case1.Target.customers` group by customer_city
3 order by customer_city
```

Press Alt+F

Query results

SAVE RESULTS | EXPL

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

Row	customers_count	customer_city
1	3	abadia dos dourados
2	1	abadiania
3	12	abaete
4	11	abaetetuba
5	2	abaiara
6	2	abaira
7	2	abare
8	3	abatia
9	1	abdon batista
10	6	abelardo luz
11	2	abrantes
12	6	abre campo
13	11	abreu e lima
14	2	acaiaca
15	7	acailandia
16	1	acajutiba
17	8	acarau
18	1	acari
19	2	acegua

4. Impact on Economy: Analyze the money movemented by e-commerce by looking at order prices, freight and others.
    1. Get % increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only)

RUN SAVE SHARE SCHEDULE MORE Query com

```sql
with temp1 as(
select o.order_id, o_i.price, date(order_purchase_timestamp) date_detail from `scaler-case1.Target.orders` as o inner join `scaler-case1.Target.order_items` as o_i
on o.order_id = o_i.order_id
),

temp2 as(
select *, extract(month from date_detail) Month, extract(year from date_detail) Year from temp1
),

temp3 as(
select * from temp2 where Month in (1,2,3,4,5,6,7,8) and Year in (2017, 2018) order by Year, Month
),

temp4 as (
select sum(price) Cost_price, Month, Year from temp3 group by Month, Year order by Year, Month
)

select *,
  round(100 * (cost_price/lag(cost_price) over(order by Year,Month) - 1), 2)  || '%' percentage
from temp4 order by Year,Month
```

Query results SAVE RESULTS EXPLORE DA

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

Row	Cost_price	Month	Year	percentage
1	120312.869...	1	2017	null
2	247303.019...	2	2017	105.55%
3	374344.300...	3	2017	51.37%
4	359927.230...	4	2017	-3.85%
5	506071.140...	5	2017	40.6%
6	433038.600...	6	2017	-14.43%
7	498031.480...	7	2017	15.01%
8	573971.680...	8	2017	15.25%
9	950030.360...	1	2018	65.52%
10	844178.710...	2	2018	-11.14%
11	983213.440...	3	2018	16.47%
12	996647.750...	4	2018	1.37%
13	996517.680...	5	2018	-0.01%
14	865124.310...	6	2018	-13.19%

2. Mean & Sum of price and freight value by customer state

RUN Close tab SHARE SCHEDULE MORE This query will process 15.42 MB whe

```sql
with val as(
select distinct o.order_id, o_i.price, o_i.freight_value, c.customer_id, c.customer_state  from `scaler-case1.Target.orders` as o inner join `scaler-case1.Target.customers` as c on o.customer_id = c.customer_id inner join `scaler-case1.Target..order_items` as o_i on o_i.order_id = o.order_id
)
select sum(price) price_sum, avg(price) price_mean, sum(freight_value) freight_value_sum, avg(freight_value) freight_value_mean, customer_state from val
group by customer_state
```

Press Alt+F1 for Accessibility

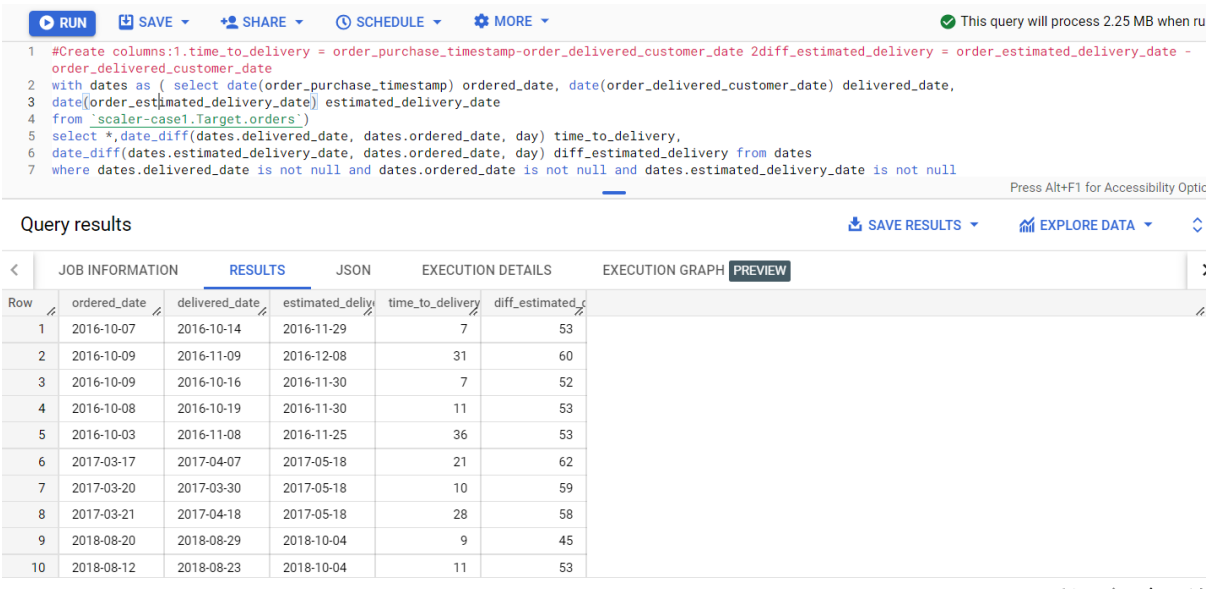
Query results SAVE RESULTS EXPLORE DATA

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

Row	price_sum	price_mean	freight_value_sum	freight_value_mean	customer_state
1	142587.88999999958	151.36718683651802	26529.040000000059	28.162462845010587	MT
2	113895.72999999995	149.86280263157897	23805.600000000013	37.902105263157871	MA
3	77010.96	183.79704057279253	15150.629999999992	36.15902147971358	AL
4	4857235.88000085	113.79790267788083	651453.41999999247	15.262596818405378	SP
5	1485174.4399999098	124.97260518343998	246902.87000000529	20.776074554022149	MG
6	250482.14999999688	148.91923305588591	55281.809999999896	32.866712247324671	PE
7	1696444.9999998615	129.18405421870185	276930.56000000483	21.088224185196353	RJ
8	286146.1899999975	130.95935469107525	46554.499999999847	21.306407322654486	DF
9	698873.66000002087	124.97740701001406	122825.03000000137	21.964418812589432	RS

5. Analysis on sales, freight and delivery time
    1. Calculate days between purchasing, delivering and estimated delivery
    2. Create columns:
        - time_to_delivery = order_purchase_timestamp-order_delivered_customer_date

- diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date

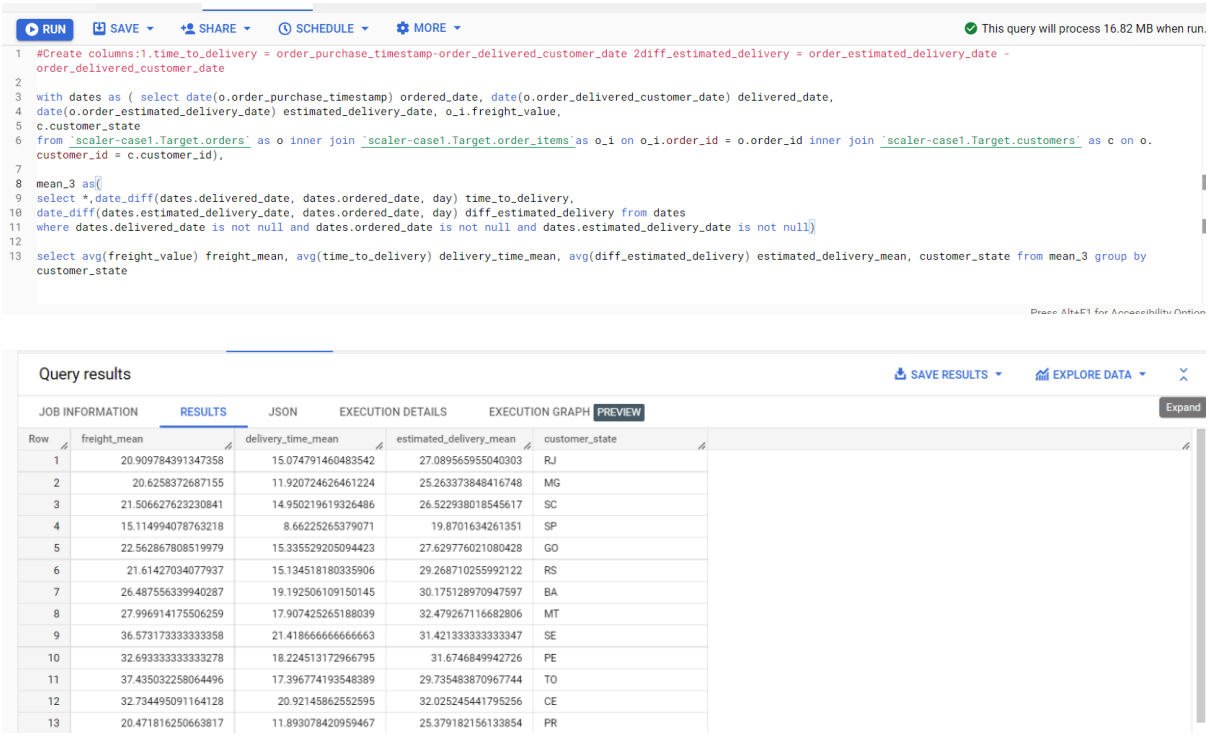


```
#Create columns:1.time_to_delivery = order_purchase_timestamp-order_delivered_customer_date 2diff_estimated_delivery = order_estimated_delivery_date -
order_delivered_customer_date
with dates as ( select date(order_purchase_timestamp) ordered_date, date(order_delivered_customer_date) delivered_date,
date(order_estimated_delivery_date) estimated_delivery_date
from `scaler-case1.Target.orders`)
select *,date_diff(dates.delivered_date, dates.ordered_date, day) time_to_delivery,
date_diff(dates.estimated_delivery_date, dates.ordered_date, day) diff_estimated_delivery from dates
where dates.delivered_date is not null and dates.ordered_date is not null and dates.estimated_delivery_date is not null
```

Query results

Row	ordered_date	delivered_date	estimated_deliv	time_to_delivery	diff_estimated_c
1	2016-10-07	2016-10-14	2016-11-29	7	53
2	2016-10-09	2016-11-09	2016-12-08	31	60
3	2016-10-09	2016-10-16	2016-11-30	7	52
4	2016-10-08	2016-10-19	2016-11-30	11	53
5	2016-10-03	2016-11-08	2016-11-25	36	53
6	2017-03-17	2017-04-07	2017-05-18	21	62
7	2017-03-20	2017-03-30	2017-05-18	10	59
8	2017-03-21	2017-04-18	2017-05-18	28	58
9	2018-08-20	2018-08-29	2018-10-04	9	45
10	2018-08-12	2018-08-23	2018-10-04	11	53

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



```
#Create columns:1.time_to_delivery = order_purchase_timestamp-order_delivered_customer_date 2diff_estimated_delivery = order_estimated_delivery_date -
order_delivered_customer_date

with dates as ( select date(o.order_purchase_timestamp) ordered_date, date(o.order_delivered_customer_date) delivered_date,
date(o.order_estimated_delivery_date) estimated_delivery_date, o_i.freight_value,
c.customer_state
from `scaler-case1.Target.orders` as o inner join `scaler-case1.Target.order_items`as o_i on o_i.order_id = o.order_id inner join `scaler-case1.Target.customers` as c on o.
customer_id = c.customer_id),

mean_3 as(
select *,date_diff(dates.delivered_date, dates.ordered_date, day) time_to_delivery,
date_diff(dates.estimated_delivery_date, dates.ordered_date, day) diff_estimated_delivery from dates
where dates.delivered_date is not null and dates.ordered_date is not null and dates.estimated_delivery_date is not null)

select avg(freight_value) freight_mean, avg(time_to_delivery) delivery_time_mean, avg(diff_estimated_delivery) estimated_delivery_mean, customer_state from mean_3 group by
customer_state
```

Query results

Row	freight_mean	delivery_time_mean	estimated_delivery_mean	customer_state
1	20.909784391347358	15.074791460483542	27.089565955040303	RJ
2	20.6258372687155	11.920724626461224	25.263373848416748	MG
3	21.506627623230841	14.950219619326486	26.522938018545617	SC
4	15.114994078763218	8.66225265379071	19.8701634261351	SP
5	22.562867808519979	15.335529205094423	27.629776021080428	GO
6	21.61427034077937	15.134518180335906	29.268710255992122	RS
7	26.487556339940287	19.192506109150145	30.175128970947597	BA
8	27.996914175506259	17.907425265188039	32.479267116682806	MT
9	36.573173333333358	21.418666666666663	31.421333333333347	SE
10	32.693333333333278	18.224513172966795	31.6746849942726	PE
11	37.435032258064496	17.396774193548389	29.735483870967744	TO
12	32.734495091164128	20.92145862552595	32.025245441795256	CE
13	20.471816250663817	11.893078420959467	25.379182156133854	PR

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

RUN SAVE SHARE SCHEDULE MORE Query completed.

```
with val as(
select distinct o.order_id, o_i.price, o_i.freight_value, c.customer_id, c.
customer_state  from `scaler-case1.Target.orders` as o inner join `scaler-case1.
Target.customers` as c on o.customer_id = c.customer_id inner join `scaler-case1.
Target..order_items` as o_i on o_i.order_id = o.order_id)
,
val1 as(
select sum(price) price_sum, avg(price) price_mean, sum(freight_value)
freight_value_sum, avg(freight_value) freight_value_mean, customer_state from val
group by customer_state
),
val2 as(
select *,rank() over (order by freight_value_mean desc) ranker from val1
)
select customer_state, freight_value_mean from val2  where ranker <= 5 order by ranker
```

Press Alt+F1 for Accessibility Options.

Query results SAVE RESULTS EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION G

Row	customer_state	freight_valu...
1	RR	42.0302127...
2	RO	41.9667187...
3	PB	41.5168876...
4	AC	41.2995121...
5	PI	38.8619284...

RUN SAVE SHARE SCHEDULE MORE

```
with val as(
select distinct o.order_id, o_i.price, o_i.freight_value, c.customer_id, c.
customer_state  from `scaler-case1.Target.orders` as o inner join `scaler-case1.
Target.customers` as c on o.customer_id = c.customer_id inner join `scaler-case1.
Target..order_items` as o_i on o_i.order_id = o.order_id)
,
val1 as(
select sum(price) price_sum, avg(price) price_mean, sum(freight_value)
freight_value_sum, avg(freight_value) freight_value_mean, customer_state from val
group by customer_state
),
val2 as(
select *,rank() over (order by freight_value_mean asc) ranker from val1
)
select customer_state, freight_value_mean from val2  where ranker <= 5 order by ranker
```

Press Alt+F1 for Accessibility Optio

Query results SAVE RESULTS EXPLORE DATA

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

Row	customer_state	freight_value_mean
1	SP	15.262596818405378
2	PR	20.448664972634887
3	MG	20.776074554022149
4	RJ	21.088224185196353
5	DF	21.306407322654486

- Top 5 states with highest average time to delivery

```
with dates as ( select date(order_purchase_timestamp) ordered_date, date(order_delivered_customer_date) delivered_date, customer_id
from `scaler-case1.Target.orders`),

cal_time_diff as(select *,date_diff(dates.delivered_date, dates.ordered_date, day) time_to_delivery from dates
where dates.delivered_date is not null and dates.ordered_date is not null
),

val2 as(
  SELECT avg(ca.time_to_delivery) time_diff_calc , c.customer_state from cal_time_diff ca inner join `scaler-case1.Target.customers` c on
  ca.customer_id = c.customer_id group by c.customer_state
),
val3 as(
select *,rank() over (order by val2.time_diff_calc asc) ranker_lowest from val2
)
select * from val3
where ranker_lowest <= 5 order by ranker_lowest
```

Press Alt+F1 for Accessib

Query results SAVE RESULTS EXPLORE DATA

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

Row	time_diff_calc	customer_state	ranker_lowest
1	8.70053092...	SP	1
2	11.9380459...	PR	2
3	11.9465433...	MG	3
4	12.8990384...	DF	4
5	14.9075274...	SC	5

- Top 5 states with lowest average time to delivery

```sql
with dates as ( select date(order_purchase_timestamp) ordered_date, date(order_delivered_customer_date) delivered_date, customer_id
from `scaler-case1.Target.orders`),

cal_time_diff as(select *,date_diff(dates.delivered_date, dates.ordered_date, day) time_to_delivery from dates
where dates.delivered_date is not null and dates.ordered_date is not null
),

val2 as(
  SELECT avg(ca.time_to_delivery) time_diff_calc , c.customer_state from cal_time_diff ca inner join `scaler-case1.Target.customers` c on
  ca.customer_id = c.customer_id group by c.customer_state
),
val3 as(
select *,rank() over (order by val2.time_diff_calc desc) ranker_highest from val2
)
select * from val3
where ranker_highest <= 5 order by ranker_highest
```

Press Alt+F1 for Accessibili

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	time_diff_calc	customer_state	ranker_highest
1	29.3414634...	RR	1
2	27.1791044...	AP	2
3	26.3586206...	AM	3
4	24.5012594...	AL	4
5	23.7251585...	PA	5

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

RUN | SAVE | SHARE | SCHEDULE | MORE | This query will process 8.32 MB when run

```sql
with dates as ( select date(order_delivered_customer_date) delivered_date, date(order_estimated_delivery_date) estimated_delivery, customer_id
from `scaler-case1.Target.orders`),

cal_time_diff as(select *,date_diff(dates.estimated_delivery, dates.delivered_date, day) time_to_delivery from dates
where dates.delivered_date is not null and dates.estimated_delivery is not null
),

val2 as(
  SELECT avg(ca.time_to_delivery) time_diff_calc , c.customer_state from cal_time_diff ca inner join `scaler-case1.Target.customers` c on
  ca.customer_id = c.customer_id group by c.customer_state
),
val3 as(
select *,rank() over (order by val2.time_diff_calc desc) ranker_highest from val2
)
select * from val3
where ranker_highest <= 5 order by ranker_highest
```

Press Alt+F1 for Accessibility Option

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	time_diff_calc	customer_state	ranker_highest
1	20.7249999...	AC	1
2	20.1028806...	RO	2
3	19.6865671...	AP	3
4	19.5655172...	AM	4
5	17.2926829...	RR	5

- Top 5 states where delivery is really very fast compared to estimated date

```sql
with dates as ( select date(order_delivered_customer_date) delivered_date, date(order_estimated_delivery_date) estimated_delivery, customer_id
from `scaler-case1.Target.orders`),

cal_time_diff as(select *,date_diff(dates.estimated_delivery, dates.delivered_date, day) time_to_delivery from dates
where dates.delivered_date is not null and dates.estimated_delivery is not null
),

val2 as(
  SELECT avg(ca.time_to_delivery) time_diff_calc , c.customer_state from cal_time_diff ca inner join `scaler-case1.Target.customers` c on
  ca.customer_id = c.customer_id group by c.customer_state
),
val3 as(
select *,rank() over (order by val2.time_diff_calc asc) ranker_lowest from val2
)
select * from val3
where ranker_lowest <= 5 order by ranker_lowest
```

Press Alt+F1 for Accessibili

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	time_diff_calc	customer_state	ranker_highest
1	20.7249999...	AC	1
2	20.1028806...	RO	2
3	19.6865671...	AP	3
4	19.5655172...	AM	4
5	17.2926829...	RR	5

6. Payment type analysis:
    1. Month over Month count of orders for different payment types

RUN | SAVE | SHARE | SCHEDULE | MORE

```sql
with temp1 as(
  select order_id, date(order_purchase_timestamp) date_detail from `scaler-case1.Target.orders`
),

temp2 as(
  select t.order_id, p.payment_type, date_detail from temp1 as t
  inner join `scaler-case1.Target.payments` as p on t.order_id = p.order_id
  ),

temp3 as(
select count(order_id) count_orders, payment_type,
extract(year from date_detail) Year, extract(month from date_detail) Month from temp2
group by extract(month from date_detail),extract(year from date_detail),payment_type
)

select * from temp3 order by Year, Month
```

Query results

SAVE

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

Row	count_orders	payment_type	Year	Month
1	3	credit_card	2016	9
2	254	credit_card	2016	10
3	63	UPI	2016	10
4	23	voucher	2016	10
5	2	debit_card	2016	10
6	1	credit_card	2016	12
7	583	credit_card	2017	1
8	197	UPI	2017	1
9	61	voucher	2017	1
10	9	debit_card	2017	1

2. Distribution of payment installments and count of orders

RUN SAVE SHARE SCHEDULE MORE

```
1  with temp1 as(
2  select o.order_id, p.payment_installments from `scaler-case1.Target.orders` as o
3  inner join `scaler-case1.Target.payments` as p on p.order_id = o.order_id
4  )
5
6  select count(order_id) count_orders, payment_installments from temp1 group by payment_installments order by payment_installments
```

Press Alt+F1 for A

Query results

SAVE RESULTS

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

Row	count_orders	payment_installments
1	2	0
2	52546	1
3	12413	2
4	10461	3
5	7098	4
6	5239	5
7	3920	6
8	1626	7
9	4268	8
10	644	9
11	5328	10
12	23	11
13	133	12
14	16	13
15	15	14
16	74	15
17	5	16
18	8	17
19	27	18
20	17	20
21	3	21
22	1	22
23	1	23

```sql
with temp1 as(
select o.order_id, p.payment_installments from `scaler-case1.Target.orders` as o
inner join `scaler-case1.Target.payments` as p on p.order_id = o.order_id
)

select count(order_id) count_orders, payment_installments from temp1 group by payment_installments order by payment_installments
```

Press Alt+F1 f

Query results

SAVE RESULTS | EXPLOR

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

Row	count_orders	payment_installments
1	2	0
2	52546	1
3	12413	2
4	10461	3
5	7098	4
6	5239	5
7	3920	6
8	1626	7
9	4268	8
10	644	9
11	5328	10
12	23	11
13	133	12
14	16	13
15	15	14
16	74	15
17	5	16

Results per page: 50 1 – 24 of 24