DataTypes of Tabel Columns in a given dataset

1. customers table

i) customer_id = string
ii) customer_unique_id = string
iii) customer_zip_code_prefix = integer
iv) customer_city = string
v) customer_state = string

2. geolocation table

i) geolocation_zip_code_prefix = integer
ii) geolocation_lat = Float
iii) geolocation_lng = Float
iv) geolocation_city = String
v) geolocation_state = String

3. order_items table

i) order_id = String
ii) order_item_id = Integer
iii) product_id = String
iv) seller_id = String
v) shipping_limit_date = TIMESTAMP
vi) price = Float
vii) freight_value = Float

4. order_reviews

i) review_id = String
ii) order_id = String
iii) review_score = Integer
iv) review_comment_title=String
v) reveiw_creation_date=Timestamp
vi) review_answer_timestamp = Timestamp

5. order table

i) order_id = String
ii) customer_id = String
iii) order_status = String
iv) order_purchase_timestamp = TIMESTAMP

v) order_approved_at = TIMESTAMP
vi) order_delivered_carrier_date = TIMESTAMP
vii) order_delivered_customer_date = TIMESTAMP
viii) order_estimated_delivery_date = TIMESTAMP

6. payments table

i) order_id = String
ii) payment_sequential = Integer
iii) payment_type = String
iv) payment_installments= Integer
v) payment_value = Float

7. products_table

i) product_id= String
ii) product_category = String
iii) product_name_length = INTEGER
iv) product_description_length = INTEGER
v) product_photos_qty = INTEGER
vi) product_weight_g = INTEGER
vii) product_length_cm = INTEGER
viii) product_height_cm = INTEGER
ix) product_width_cm = INTEGER

8. sellers table

i) seller_id = String
ii) seller_zip_code_prefix = Integer
iii) seller_city = String
iv) seller_state = String

The time period for which the data is given.

As observed from the below query, the first order was placed on 2016-09-04 at 21:15:19 UTC, and the last order was placed on 2018-10-17 at 17:30:18 UTC, so the dataset given is of order placed from 2016-09-04 to 2018-10-17.

Untitled 3 RUN SAVE SHARE SCHEDULE MORE

```
select min(order_purchase_timestamp) as first_order_date,
max(order_purchase_timestamp) as last_order_date
from
`target.orders`
```

Query results SAVE RESULTS

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

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

1. Below query results in Cities and States of customers ordered during the given period.

customers *Untitled 2

Untitled 2 RUN SAVE SHARE SCHEDULE MORE

```
select customer_id,
customer_city,customer_state
from
`target.customers`
group by
customer_id,customer_city,customer_state
limit 10
```

Query results

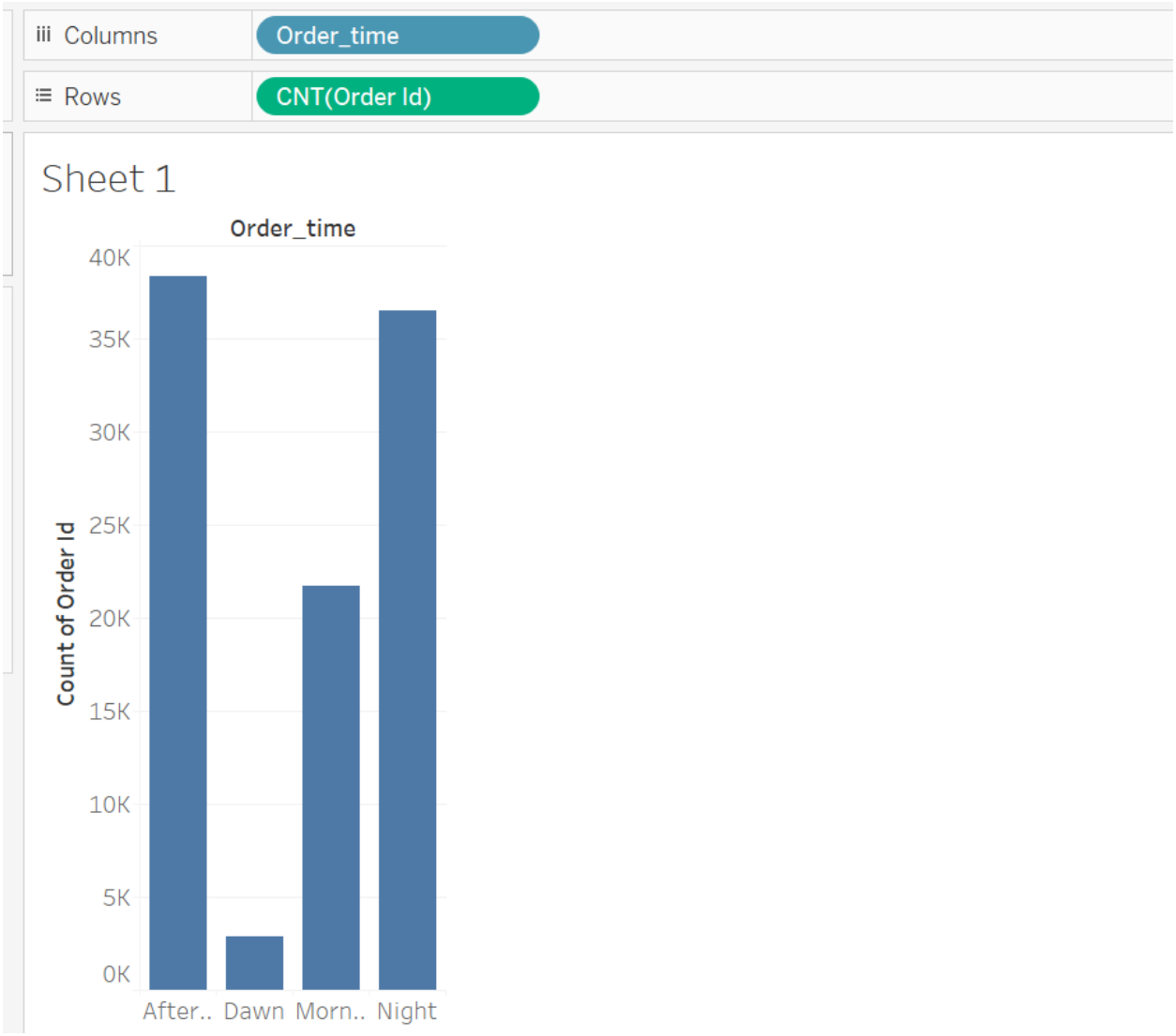
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH PREVIEW

Row	customer_id	customer_city	customer_state
1	0735e7e4298a2ebbb4664934...	acu	RN
2	903b3d86e3990db01619a4eb...	acu	RN
3	38c97666e962d4fea7fd6a83e...	acu	RN
4	77c2f46cf580f4874c9a5751c2...	ico	CE
5	4d3ef4cfffb8ad4767c199c36a...	ico	CE
6	3000841b86e1fbe9493b52324...	ico	CE
7	3c325415ccc7e622c66dec4bc...	ico	CE
8	04f3a7b250e3be964f01bf22bc...	ico	CE
9	894202b8ef01f4719a4691e79...	ico	CE
10	9d715b9fb75a9d081c14126c0...	ico	CE

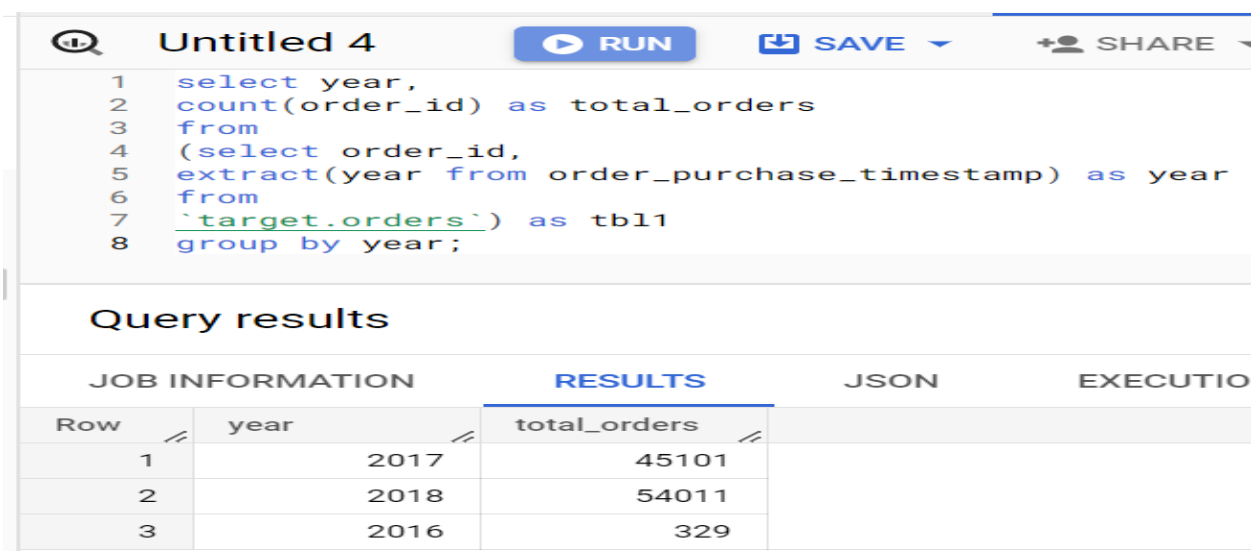
2. Below query shows that Brazil's people tend to shop more during the afternoon time i.e. from 12:00 to 18:00

```sql
select part_of_day,count(order_id) as total_orders from (select
order_id,
CAST(order_purchase_timestamp AS TIME),
Case when
Cast(order_purchase_timestamp as TIME) > '06:00:00' and Cast(order_purchase_timestamp as TIME) < '12:00:00'
then 'Morning'
when
Cast(order_purchase_timestamp as TIME) >= '12:00:00' and Cast(order_purchase_timestamp as TIME) < '18:00:00'
Then 'Afternoon'
when
Cast(order_purchase_timestamp as TIME) >= '00:00:00' and Cast(order_purchase_timestamp as TIME) < '06:00:00'
then 'Dawn'
else 'Night'
end as part_of_day,
order_purchase_timestamp
from `target.orders`) as tbl1
group by part_of_day
```

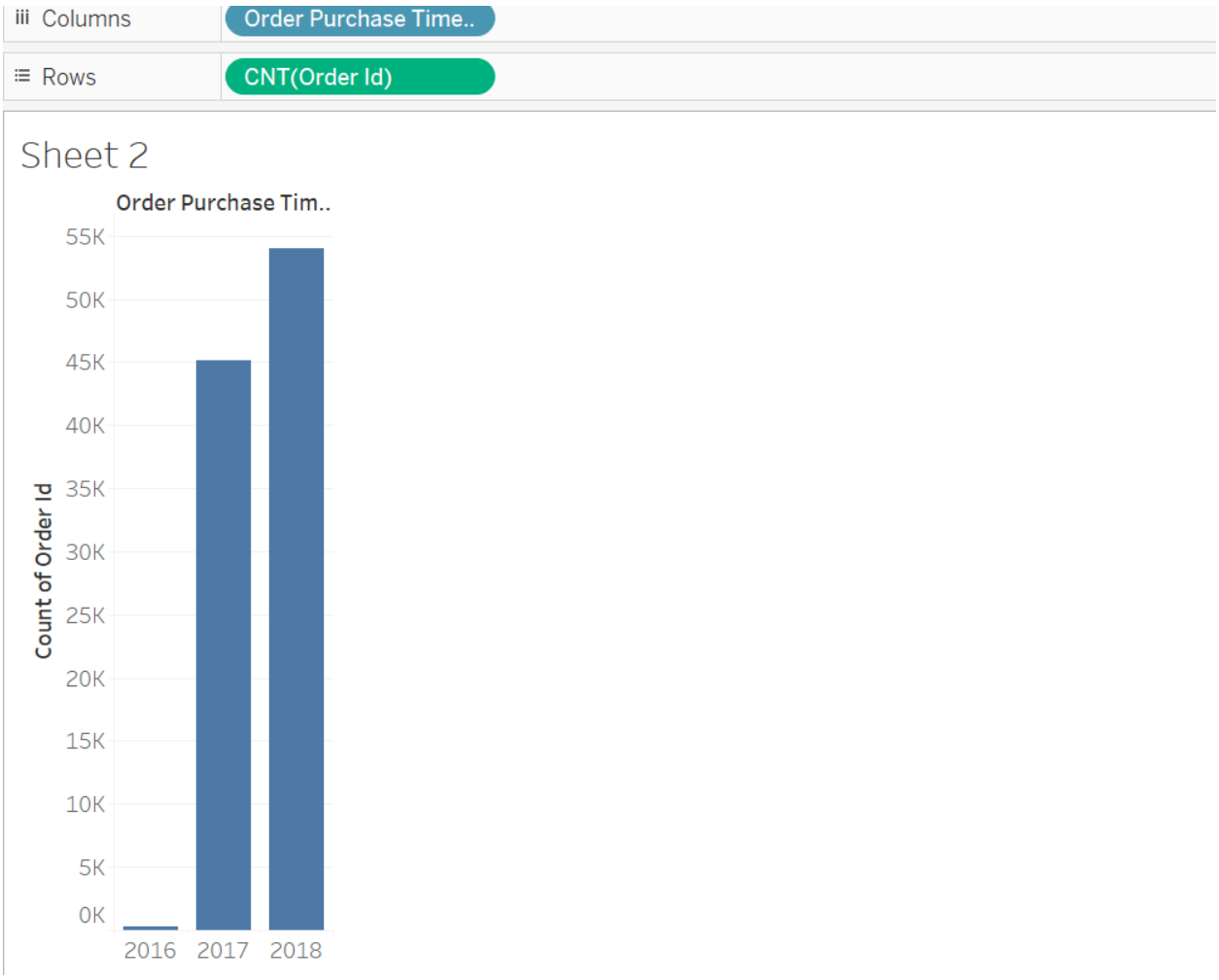
Row	part_of_day	total_orders
1	Morning	22240
2	Dawn	4740
3	Afternoon	38361
4	Night	34100



3. The below query shows the increase in orders from year to year.



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

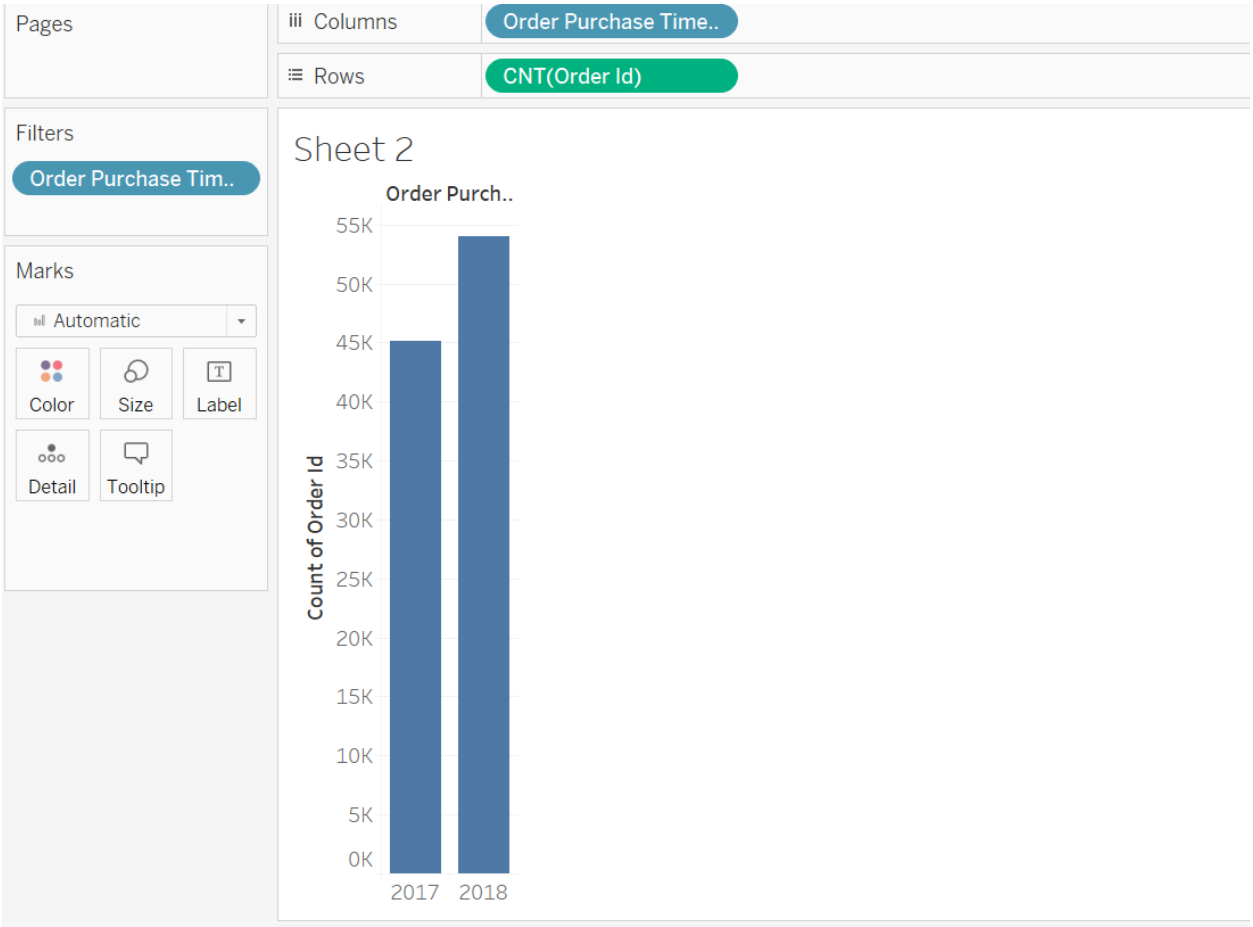


4. Below query shows that there is around 20% increase in orders from 2017 to 2018.

```
select tbl2.year,tbl2.total_order_value from (select tbl1.year,
round(sum(tbl1.payment_value),2) as total_order_value
from
(select o.order_id,
extract(month from o.order_purchase_timestamp) as month,
extract(year from o.order_purchase_timestamp) as year,
p.payment_value
from
`target.orders` as o inner join `target.payments` as p
on o.order_id = p.order_id) as tbl1
group by tbl1.year) as tbl2
where tbl2.year in(2017,2018)

```

JOB INFORMATION | RESULTS | JSON

Row	year	total_order_valu
1	2017	7249746.73
2	2018	8699763.05



5. Below query shows Mean & Sum of price and freight value by customer state.

Untitled 4 RUN SAVE SHARE SCHEDULE

```
select c.customer_state,
round(sum(it.price),2) as total_price_by_state,
round(sum(it.freight_value),2) as total_freight_by_state,
round(avg(it.price),2) as avg_price_by_state,
round(avg(it.freight_value),2) as avg_freight_by_state
from
`target.orders` as o inner join `target.order_items` as it
on o.order_id = it.order_id
inner join `target.customers` as c
on o.customer_id = c.customer_id
group by c.customer_state
order by total_price_by_state,total_freight_by_state
```

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

ow	customer_state	total_price_by_st	total_freight_by_	avg_price_by_sta	avg_freight_by_s
1	RR	7829.43	2235.19	150.57	42.98
2	AP	13474.3	2788.5	164.32	34.01
3	AC	15982.95	3686.75	173.73	40.07
4	AM	22356.84	5478.89	135.5	33.21
5	RO	46140.64	11417.38	165.97	41.07

6. Below query shows actual delivery days vs estimated delivery days and difference between them.

orders | *Untitled 4 | *Untitled 5

Untitled 5 RUN SAVE SHARE SCHEDULE MORE

```
select order_id,
date_diff(order_delivered_customer_date,order_purchase_timestamp,day) as actual_delivery_Days,
date_diff(order_estimated_delivery_date,order_purchase_timestamp,day) as estimated_delivery_Days,
date_diff(order_estimated_delivery_date,order_delivered_customer_date,day) as diff_delivery_Days
from
`target.orders`
where order_delivered_customer_date is not null
order by actual_delivery_Days,estimated_delivery_Days
```

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

Row	order_id	actual_delivery_l	estimated_delive	diff_delivery_Day
1	d5fbeedc85190ba88580d6f82...	0	8	7
2	79e324907160caea526fd8b94...	0	9	8
3	e65f1eeee1f52024ad1dcd034...	0	10	9
4	1d893dd7ca5f77ebf5f59f0d20...	0	10	10
5	b70a8d75313560b4acf607739...	0	10	9

7. Below query shows actual time taken to deliver an order.

Untitled 5 RUN SAVE SHARE SCHEDULE MORE

```
select order_id,
date_diff(order_delivered_customer_date,order_purchase_timestamp,day) as time_to_delivery
from
`target.orders`
where order_delivered_customer_date is not null
order by time_to_delivery
```

Row	order_id	time_to_delivery
9	8339b608be0d84fca9d8da68b...	0
10	f349cdb62f69c3fae5c4d7d3f3...	0
11	f3c6775ba3d2d9fe2826f93b71...	0
12	b70a8d75313560b4acf607739...	0
13	21a8ffca665bc7a1087d31751...	0
14	44558a1547e448b41c48c4087...	1
15	3bfd703ce884b8a0a65e63f2d...	1
16	68fa625f02107978e969340da...	1
17	0d8f485ffe96c81fe3e282095e...	1

8. Below query shows the avg freight value, avg time to delivery and avg diff estimated delivery

Untitled 4 RUN SAVE SHARE SCHEDULE MORE Query comple

```
SELECT c.customer_state,
round(avg(freight_value),2) as avg_freight_value,
round(avg(date_diff(order_delivered_customer_date,order_purchase_timestamp,day)),2) as avg_time_to_delivery,
round(avg(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day)),2) as
avg_diff_estimated_delivery
FROM
`target.orders` as o inner join `target.order_items` as it
on o.order_id = it.order_id
inner join `target.customers` as c
on o.customer_id = c.customer_id
group by c.customer_state
order by avg_freight_value
```

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

Row	customer_state	avg_freight_valu	avg_time_to_delivery	avg_diff_estimated_delivery
1	SP	15.15	8.26	10.27
2	PR	20.53	11.48	12.53
3	MG	20.63	11.52	12.4
4	RJ	20.96	14.69	11.14
5	DF	21.04	12.5	11.27
6	SC	21.47	14.52	10.67
7	RS	21.74	14.71	13.2
8	ES	22.06	15.19	9.77
9	GO	22.77	14.95	11.37

9. Below query shows the top 5 states with highest avg_freight_value.

Untitled 4 RUN SAVE SHARE SCHEDULE MORE Query comple

```
# Top 5 states with highest avg_freight_value

SELECT c.customer_state,
round(avg(freight_value),2) as avg_freight_value,
#round(avg(date_diff(order_delivered_customer_date,order_purchase_timestamp,day)),2) as avg_time_to_delivery,
#round(avg(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day)),2) as
avg_diff_estimated_delivery
FROM
`target.orders` as o inner join `target.order_items` as it
on o.order_id = it.order_id
inner join `target.customers` as c
on o.customer_id = c.customer_id
group by c.customer_state
order by avg_freight_value desc
limit 5
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	avg_freight_valu
1	RR	42.98
2	PB	42.72
3	RO	41.07
4	AC	40.07
5	PI	39.15

10. Below query shows the top 5 lowest_avg_freight_value.

*Untitled 4 | orders

Untitled 4 RUN SAVE SHARE SCHEDULE MORE Query c

```
# Top 5 states with lowest_avg_freight_value

SELECT c.customer_state,
round(avg(freight_value),2) as avg_freight_value,
#round(avg(date_diff(order_delivered_customer_date,order_purchase_timestamp,day)),2) as avg_time_to_delivery,
#round(avg(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day)),2) as avg_diff_estimated_delivery
FROM
`target.orders` as o inner join `target.order_items` as it
on o.order_id = it.order_id
inner join `target.customers` as c
on o.customer_id = c.customer_id
group by c.customer_state
order by avg_freight_value asc
limit 5
```

Query results

SAVE RESULTS

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

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

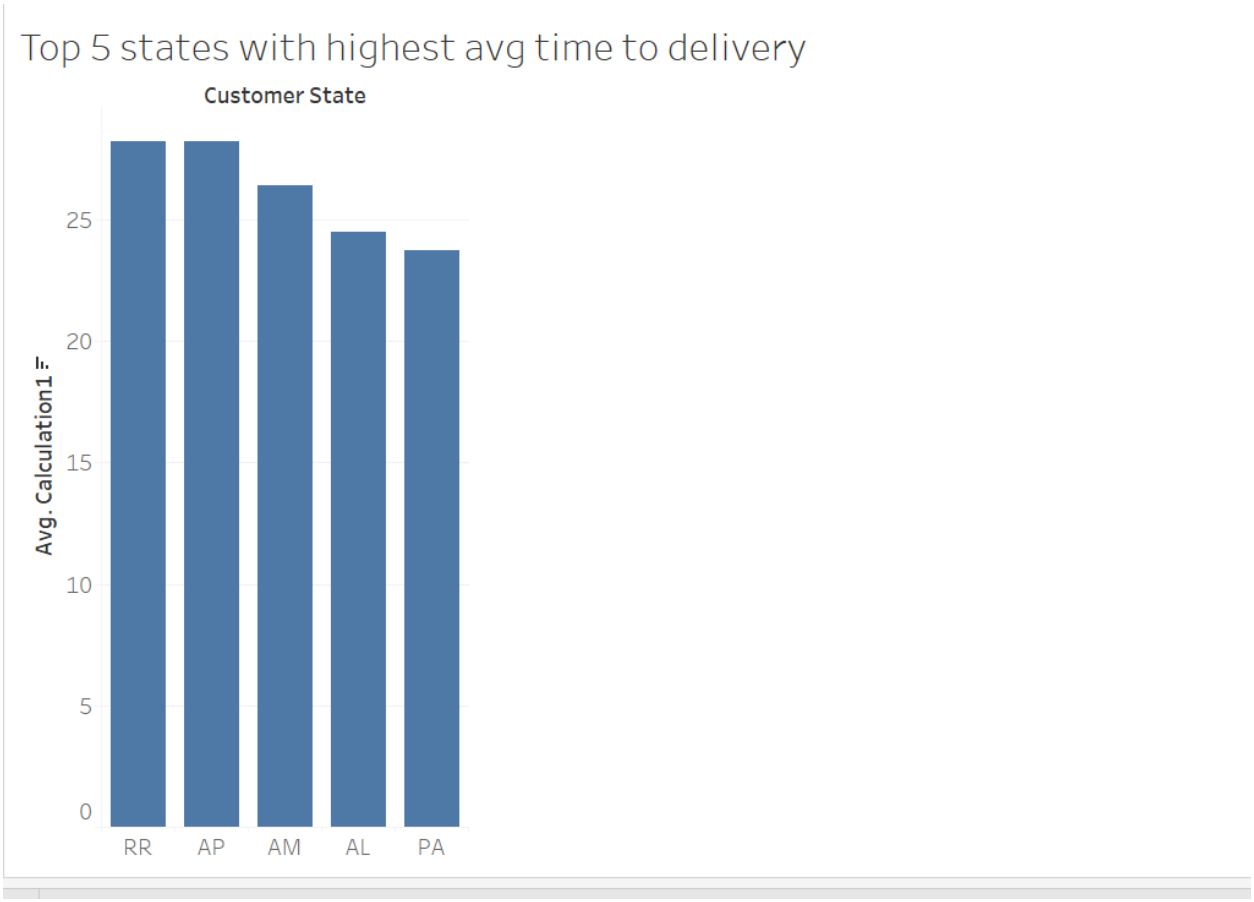
11. Below query shows the top 5 highest_avg_time_to_delivery.

Untitled 4 RUN SAVE SHARE SCHEDULE MORE Query comp

```
# Top 5 states with highest_avg_time_to_delivery

SELECT c.customer_state,
#round(avg(freight_value),2) as avg_freight_value,
round(avg(date_diff(order_delivered_customer_date,order_purchase_timestamp,day)),2) as avg_time_to_delivery,
#round(avg(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day)),2) as avg_diff_estimated_delivery
FROM
`target.orders` as o inner join `target.order_items` as it
on o.order_id = it.order_id
inner join `target.customers` as c
on o.customer_id = c.customer_id
group by c.customer_state
order by avg_time_to_delivery desc
limit 5
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAP

Row	customer_state	avg_time_to_deli
1	RR	27.83
2	AP	27.75
3	AM	25.96
4	AL	23.99
5	PA	23.3



12. Below query shows the top 5 lowest_Avg_time_to_delivery.

*Untitled 4 | orders

Untitled 4 RUN SAVE SHARE SCHEDULE MORE Query

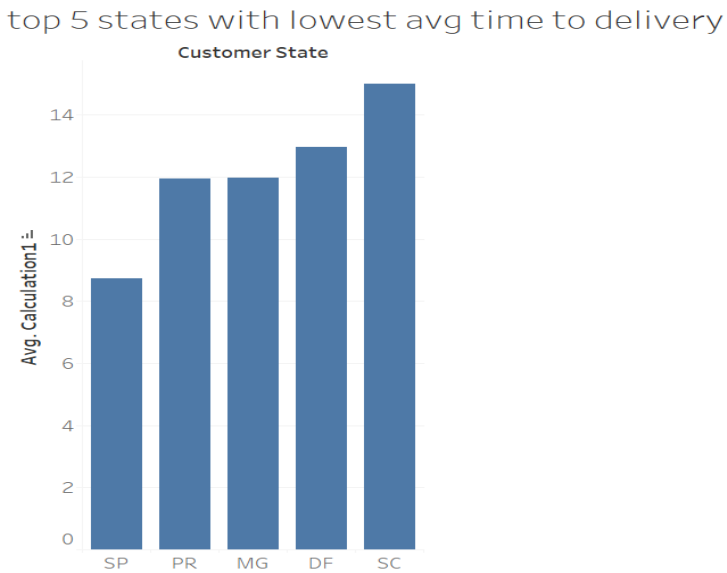
```
# Top 5 states with lowest_avg_time_to_delivery

SELECT c.customer_state,
#round(avg(freight_value),2) as avg_freight_value,
round(avg(date_diff(order_delivered_customer_date,order_purchase_timestamp,day)),2) as avg_time_to_delivery,
#round(avg(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day)),2) as avg_diff_estimated_delivery
FROM
`target.orders` as o inner join `target.order_items` as it
on o.order_id = it.order_id
inner join `target.customers` as c
on o.customer_id = c.customer_id
group by c.customer_state
order by avg_time_to_delivery
limit 5
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GI

Row	customer_state	avg_time_to_delivery
1	SP	8.26
2	PR	11.48
3	MG	11.52
4	DF	12.5
5	SC	14.52

Columns: Customer State

Rows: AVG(Calculation1)



13. From below query the findings are that the product always reached before estimated delivery date.

Untitled RUN SAVE SHARE SCHEDULE MORE This query will process 15.96 MB wl

```
#Top 5 states where delivery is really fast/ not so fast compared to estimated date
select * from (select c.customer_state,
round(avg(date_diff(order_delivered_customer_date,order_purchase_timestamp,day)),2) as avg_time_to_delivery,
round(avg(date_diff(order_estimated_delivery_date,order_purchase_timestamp,day)),2) as estimated_delivery_Date
from
`target.orders` as o inner join `target.order_items` as it
on o.order_id =it.order_id
inner join `target.customers` as c
on o.customer_id = c.customer_id
group by c.customer_state) as tbl1
where tbl1.estimated_delivery_Date < tbl1.avg_time_to_delivery
```

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

There is no data to display.

14. Below query shows the Month over Month count of orders for different payment types

Untitled 2 RUN SAVE SHARE SCHEDULE MORE Query co

```
select tbl1.payment_type, tbl1.month, count(tbl1.order_id) as num_of_orders from (select p.payment_type,o.order_id,
extract(month from o.order_purchase_timestamp) as month
from
`target.payments` as p inner join `target.orders` as o
on p.order_id = o.order_id) tbl1
group by tbl1.month,tbl1.payment_type
order by tbl1.month,tbl1.payment_type
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION

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

15 . Below query shows the Count of orders based on the no. of payment installments

*Untitled | *Untitled 2

Untitled 2 RUN SAVE SHARE SCHEDULE MORE

```
select tbl1.payment_installments,
count(tbl1.order_id) as num_of_orders from (select p.payment_type,o.order_id,payment_installments,
extract(month from o.order_purchase_timestamp) as month
from
`target.payments` as p inner join `target.orders` as o
on p.order_id = o.order_id) tbl1
group by tbl1.payment_installments
order by tbl1.payment_installments
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRA

Row	payment_installr	num_of_orders
1	0	2
2	1	52546
3	2	12413
4	3	10461
5	4	7098
6	5	5239
7	6	3920
8	7	1626

Load more