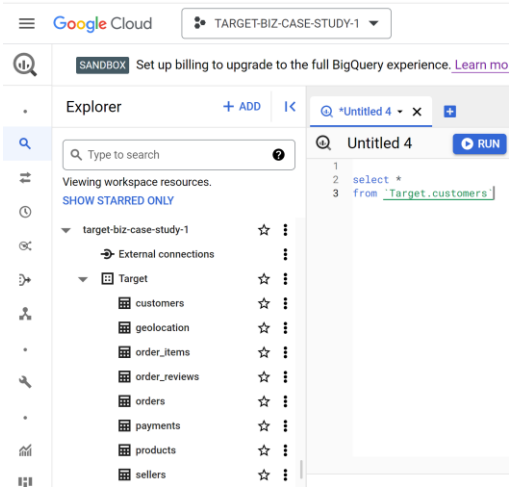
TARGET SQL CASE STUDY

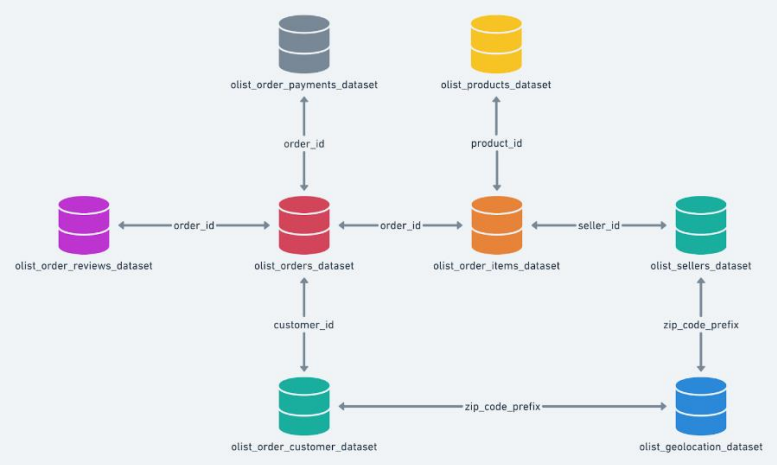
Name: Sai Vivekanand Kuchimanchi (Vivek)
Date : 17/4/2023

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
    2. Time period for which the data is given
    3. Cities and States of customers ordered during the given period

~ Import the dataset

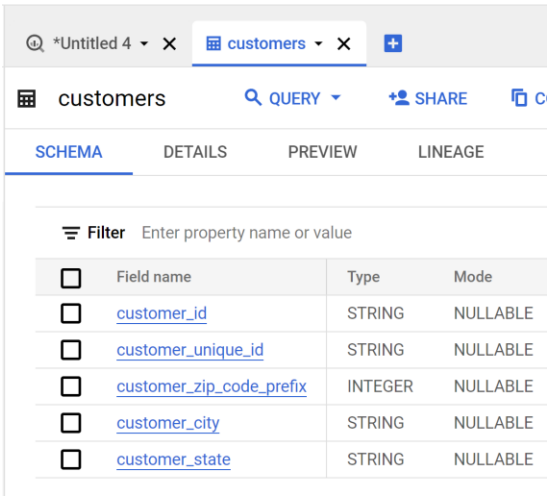


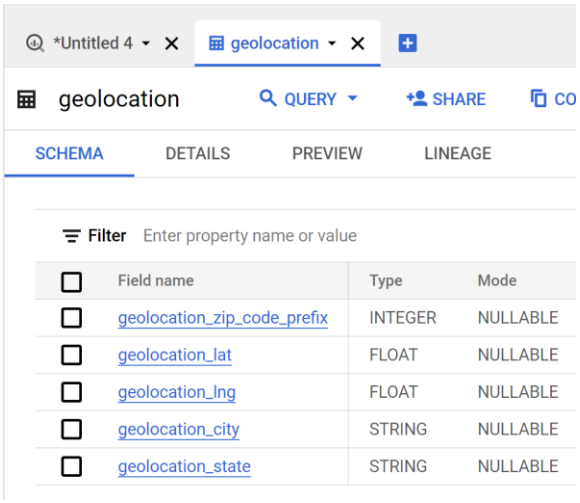
High level overview of relationship between datasets:

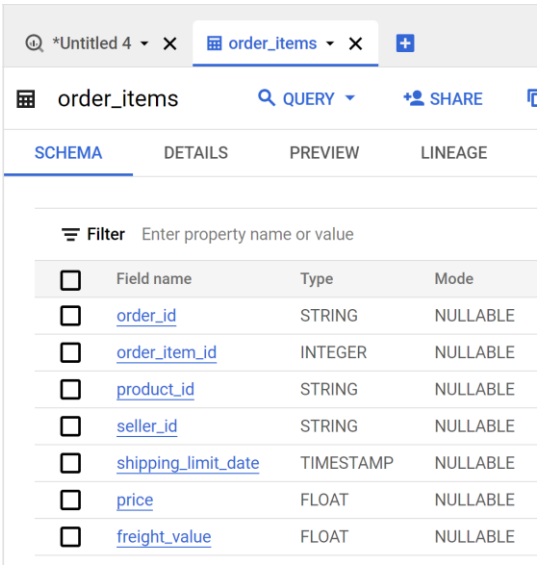


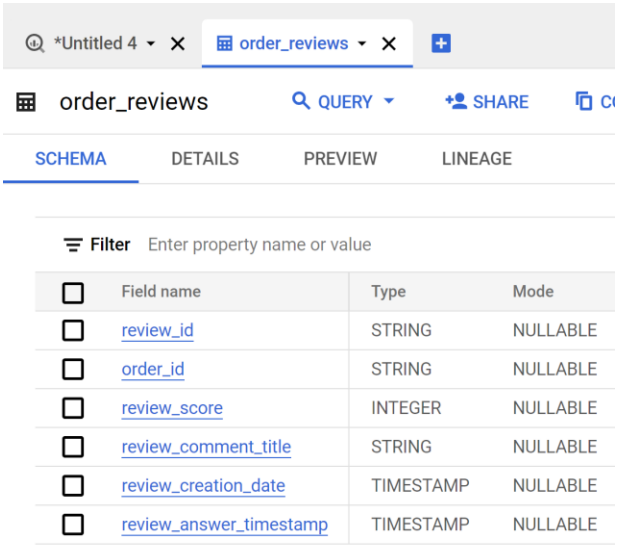
~ do usual exploratory analysis steps like checking the structure & characteristics of the dataset

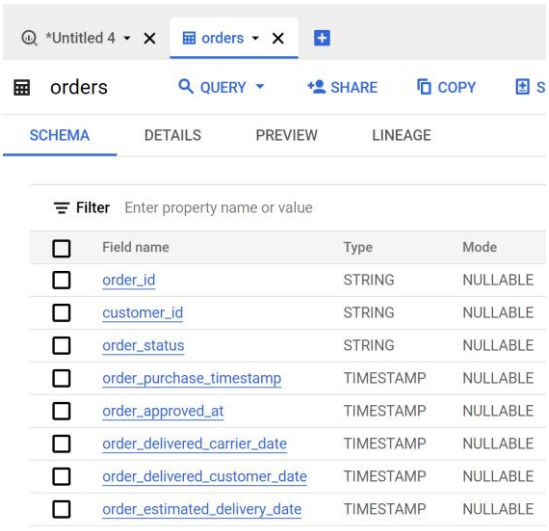
1. Data type of columns in a table

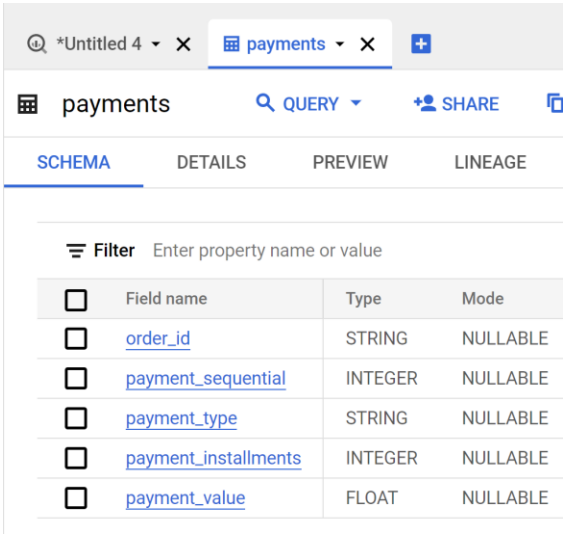


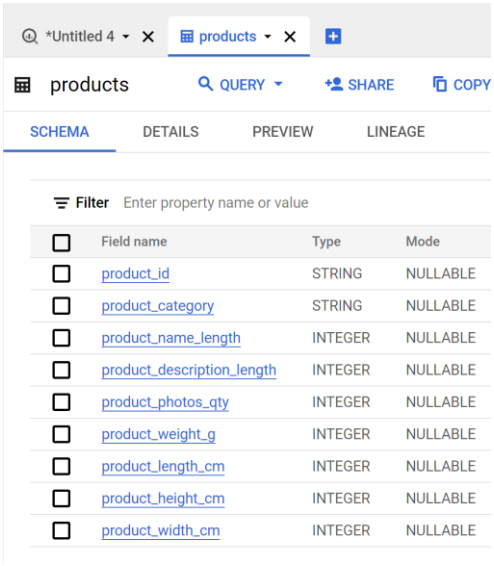


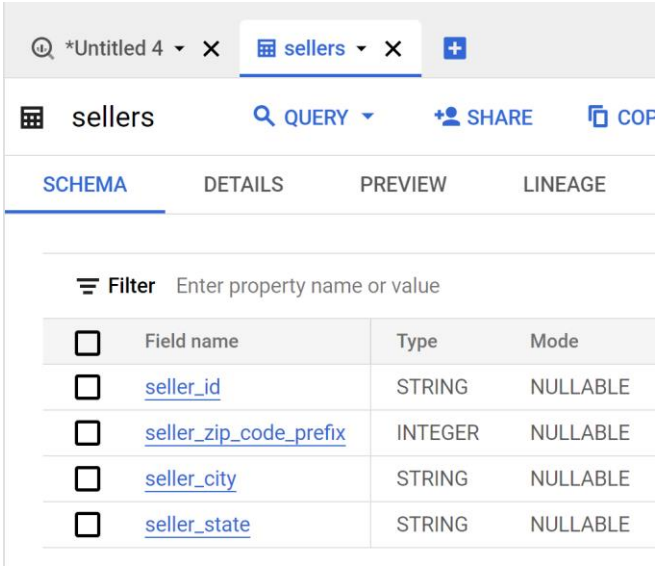





*Untitled 4
orders
orders
QUERY
SHARE
COPY
SCHEMA
DETAILS
PREVIEW
LINEAGE
Filter
Enter property name or value
Field name
Type
Mode
order_id
STRING
NULLABLE
customer_id
STRING
NULLABLE
order_status
STRING
NULLABLE
order_purchase_timestamp
TIMESTAMP
NULLABLE
order_approved_at
TIMESTAMP
NULLABLE
order_delivered_carrier_date
TIMESTAMP
NULLABLE
order_delivered_customer_date
TIMESTAMP
NULLABLE
order_estimated_delivery_date
TIMESTAMP
NULLABLE


*Untitled 4
payments
payments
QUERY
SHARE
SCHEMA
DETAILS
PREVIEW
LINEAGE
Filter
Enter property name or value
Field name
Type
Mode
order_id
STRING
NULLABLE
payment_sequential
INTEGER
NULLABLE
payment_type
STRING
NULLABLE
payment_installments
INTEGER
NULLABLE
payment_value
FLOAT
NULLABLE


*Untitled 4
products
products
QUERY
SHARE
COPY
SCHEMA
DETAILS
PREVIEW
LINEAGE
Filter
Enter property name or value
Field name
Type
Mode
product_id
STRING
NULLABLE
product_category
STRING
NULLABLE
product_name_length
INTEGER
NULLABLE
product_description_length
INTEGER
NULLABLE
product_photos_qty
INTEGER
NULLABLE
product_weight_g
INTEGER
NULLABLE
product_length_cm
INTEGER
NULLABLE
product_height_cm
INTEGER
NULLABLE
product_width_cm
INTEGER
NULLABLE


*Untitled 4
sellers
sellers
QUERY
SHARE
SCHEMA
DETAILS
PREVIEW
LINEAGE
Filter
Enter property name or value
Field name
Type
Mode
seller_id
STRING
NULLABLE
seller_zip_code_prefix
INTEGER
NULLABLE
seller_city
STRING
NULLABLE
seller_state
STRING
NULLABLE

*TARGET SQL CASE STUDY

TARGET SQL CASE STUDY

RUN SAVE

```sql
# Q2. Time period for which the data is given

select *
from `Target.orders` ;

select
min(order_purchase_timestamp) as start_time,
max(order_purchase_timestamp) as end_time
from `Target.orders` ;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETA

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

*TARGET SQL CASE STUDY

TARGET SQL CASE STUDY

RUN SAVE SHARE

```sql
# Q3. Cities and States of customers ordered during the given period
select *
from `Target.geolocation` ;

select
geolocation_state,
geolocation_city
from `Target.geolocation`
group by geolocation_state, geolocation_city
limit 10 ;
```

Query results

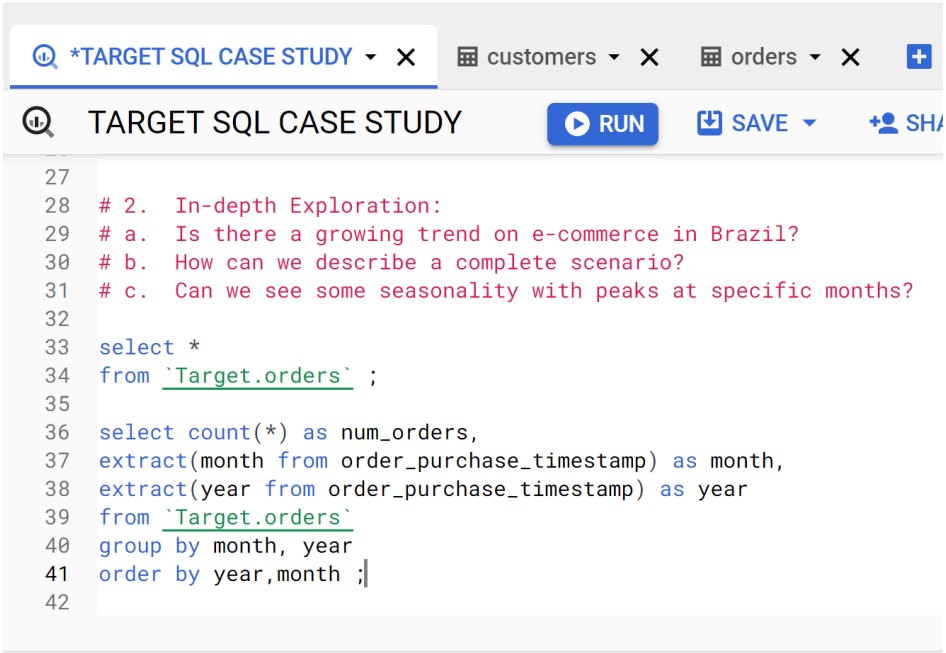
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECI

Row	geolocation_state	geolocation_city
1	SE	aracaju
2	SE	riachuelo
3	SE	nossa senhora do socorro
4	SE	barra dos coqueiros
5	SE	itaporanga d'ajuda
6	SE	sao cristovao
7	SE	são cristóvão
8	SE	santo amaro das brotas
9	SE	pirambu
10	SE	laranjeiras

Results per page: 50 1 – 50 of 8463

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?

Query :



Query results

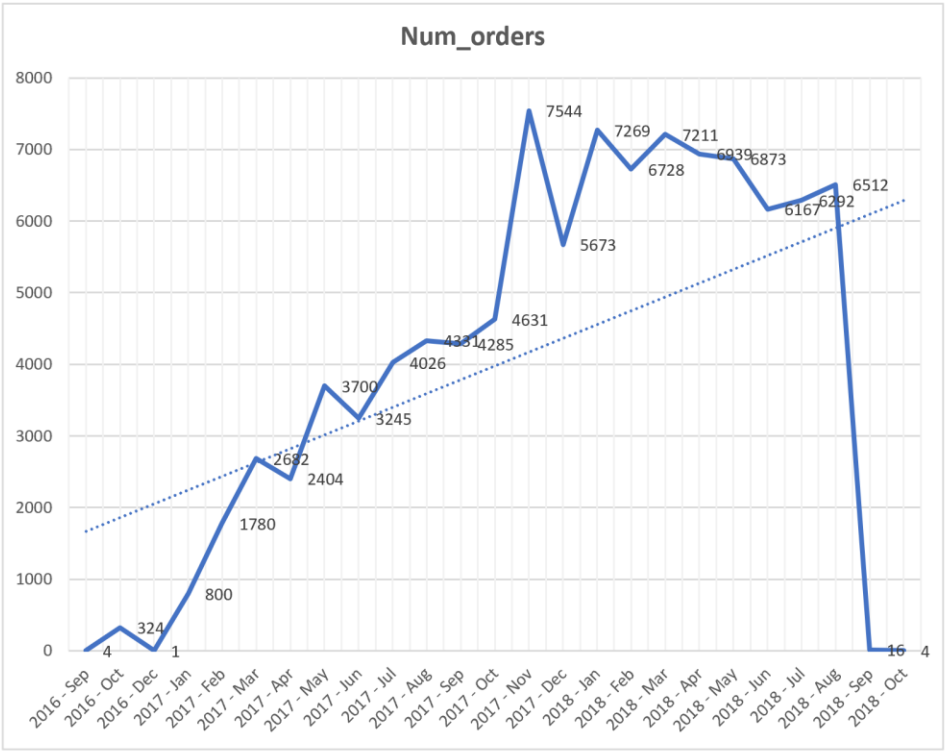
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	num_orders	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

Results per page: 50 1 – 25 of 25

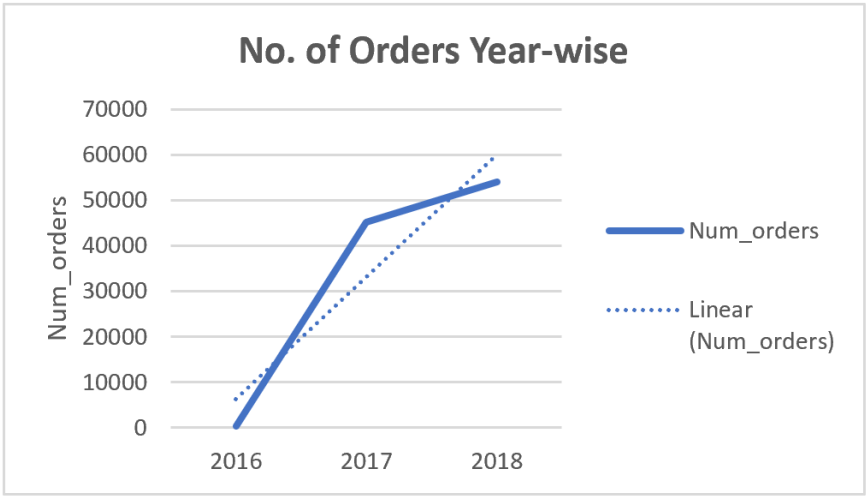
If we take a graph of the Month and Year orders, the graph has up and down trends. But the trend line is showing an upward trend.

Year_Month	Num_orders
2016 - Sep	4
2016 - Oct	324
2016 - Dec	1
2017 - Jan	800
2017 - Feb	1780
2017 - Mar	2682
2017 - Apr	2404
2017 - May	3700
2017 - Jun	3245
2017 - Jul	4026
2017 - Aug	4331
2017 - Sep	4285
2017 - Oct	4631
2017 - Nov	7544
2017 - Dec	5673
2018 - Jan	7269
2018 - Feb	6728
2018 - Mar	7211
2018 - Apr	6939
2018 - May	6873
2018 - Jun	6167
2018 - Jul	6292
2018 - Aug	6512
2018 - Sep	16
2018 - Oct	4



So I aggregated it to only years, so the trend is strikingly apparent

Year	Num_orders
2016	329
2017	45101
2018	54011

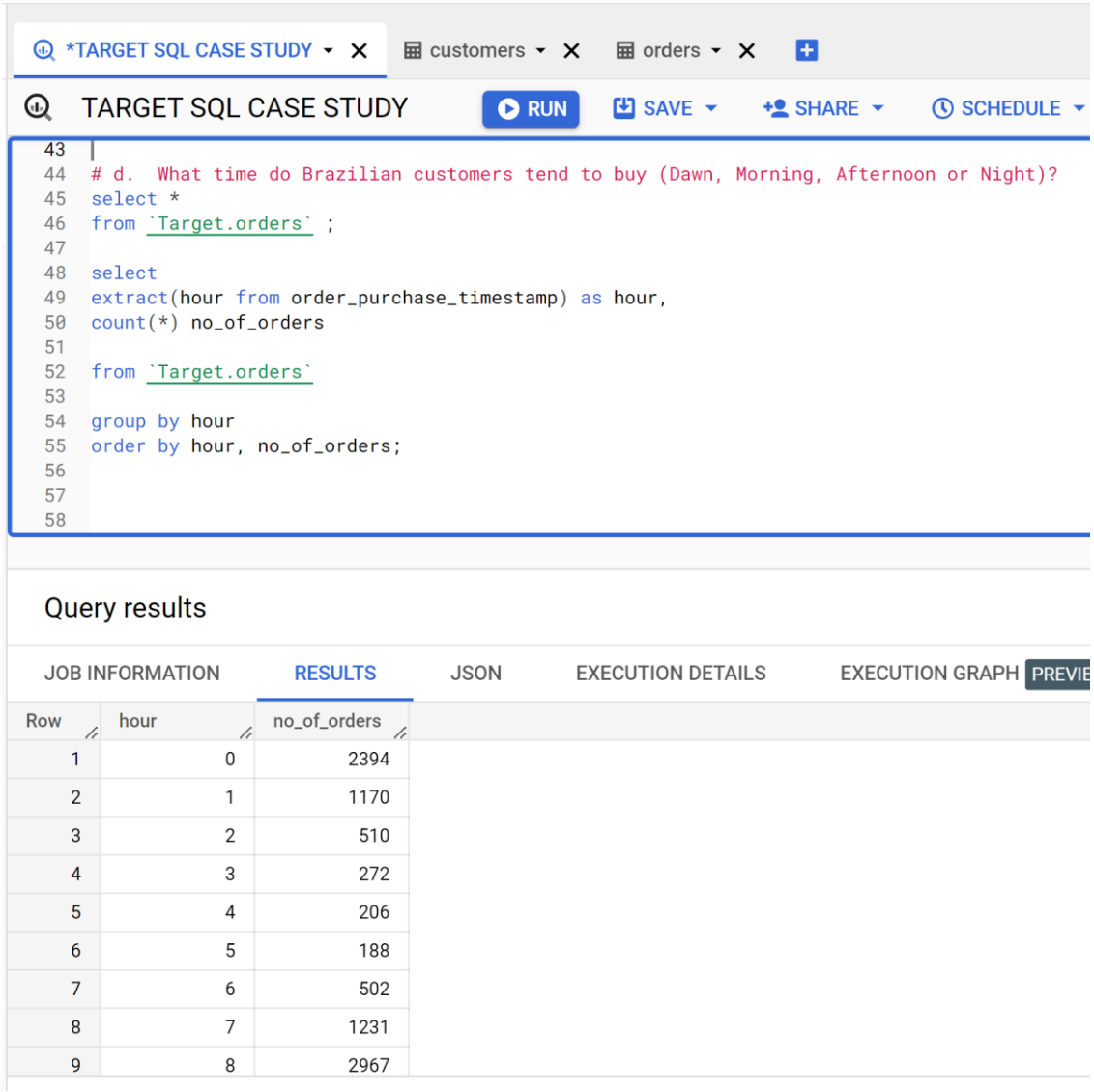


Points:

- Orders trend is growing overall.
- Although some months are very dull. 2016 (Sep, Dec) and 2018 (Sep, Dec) there is a downward trend
- 2017 (Sep, Dec) orders were quite high(2017 Sep – 4285, 2017 Dec – 5673).
- 2017 Nov – 7544 Orders (highest in the dataset)

---

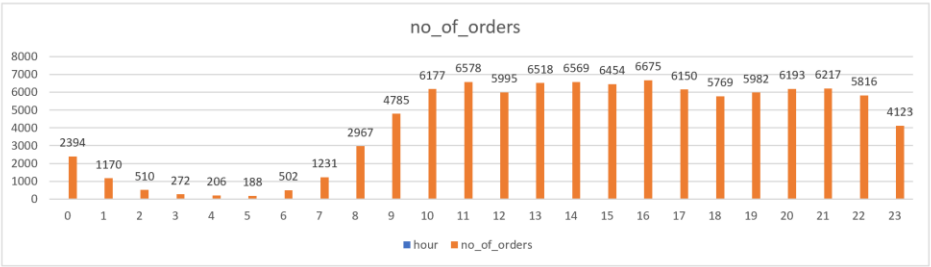
2. What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)?

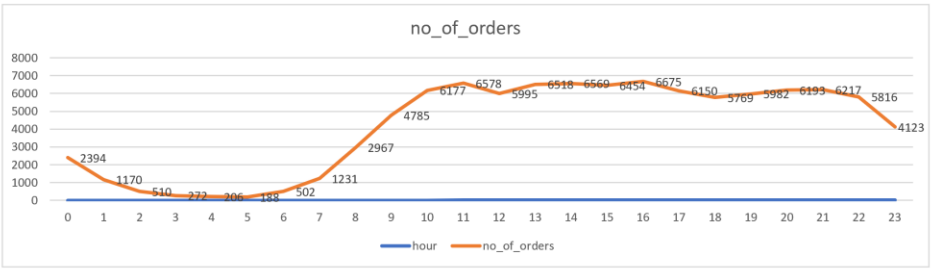


Row	hour	no_of_orders
1	0	2394
2	1	1170
3	2	510
4	3	272
5	4	206
6	5	188
7	6	502
8	7	1231
9	8	2967

99441

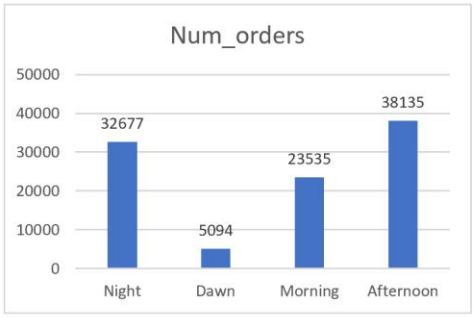
Category	hour	no_of_orders
Night	0	2394
Night	1	1170
Night	2	510
Night	3	272
Dawn	4	206
Dawn	5	188
Dawn	6	502
Dawn	7	1231
Dawn	8	2967
Morning	9	4785
Morning	10	6177
Morning	11	6578
Morning	12	5995
Afternoon	13	6518
Afternoon	14	6569
Afternoon	15	6454
Afternoon	16	6675
Afternoon	17	6150
Afternoon	18	5769
Night	19	5982
Night	20	6193
Night	21	6217
Night	22	5816
Night	23	4123

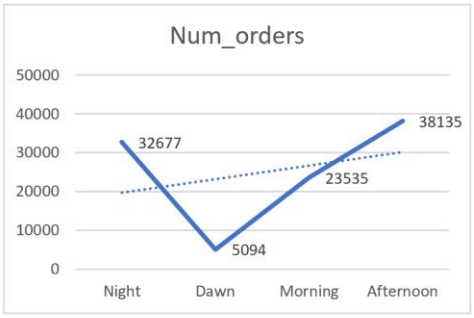


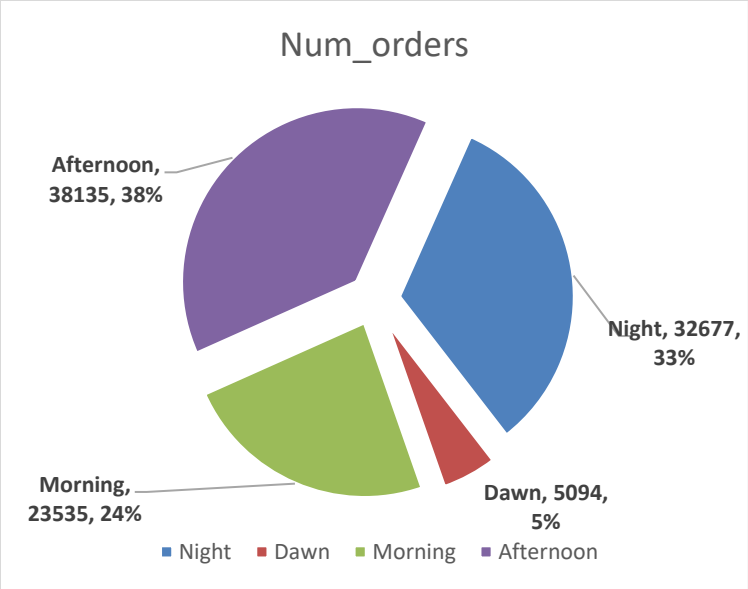


99441

Category	Num_orders
Night	32677
Dawn	5094
Morning	23535
Afternoon	38135







*TARGET SQL CASE STUDY | customers | orders

TARGET SQL CASE STUDY — RUN | SAVE | SHARE | SCHEDULE

```
43
44  # d.  What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)?
45  select *
46  from `Target.orders` ;
47
48  with hr as
49  (select
50  extract(hour from order_purchase_timestamp) as hour,
51  count(*) as no_of_orders
52
53  from `Target.orders`
54
55  group by hour )
56
57
58  select
59  sum(no_of_orders) as Tot_orders,
60
61  case
62    when hour between 4 and 8 then 'Dawn'
63    when hour between 9 and 12 then 'Afternoon'
64    when hour between 13 and 18 then 'Evening'
65    else 'Night'
66  end as time_category
67
68  from hr
69  group by time_category
70  order by Tot_orders ;
71
72
```

Query results

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

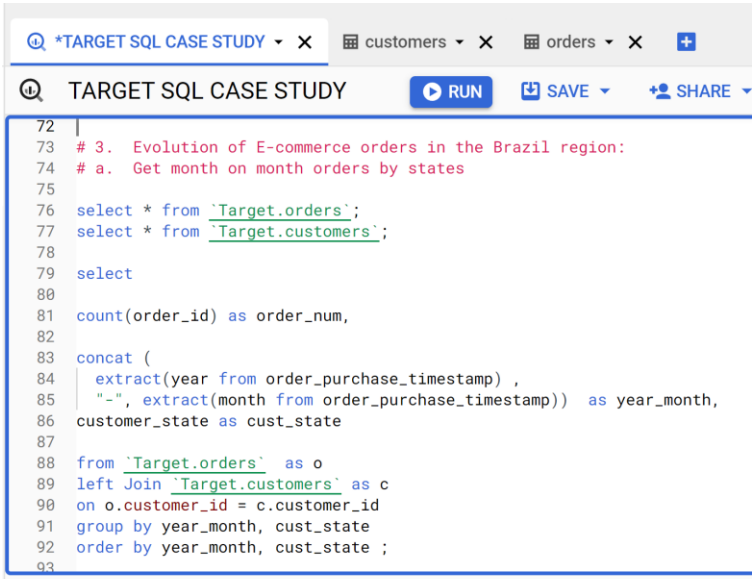
Row	Tot_orders	time_category
1	5094	Dawn
2	23535	Afternoon
3	32677	Night
4	38135	Evening

Points:

- Its obvious that in the ‘Night’ the orders are quite high and there is a dip during ‘Dawn’. Sales pick up during ‘Morning’ and in ‘Afternoon’, where its highest.

---

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



Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXEC

Row	order_num	year_month	cust_state
1	2	2016-10	AL
2	4	2016-10	BA
3	8	2016-10	CE
4	6	2016-10	DF
5	4	2016-10	ES
6	9	2016-10	GO
7	4	2016-10	MA
8	40	2016-10	MG
9	3	2016-10	MT
10	4	2016-10	PA

Results per page: 50 1 – 50 of 565

Its not possible to have all 565 state-wise, year-month wise, count of orders.

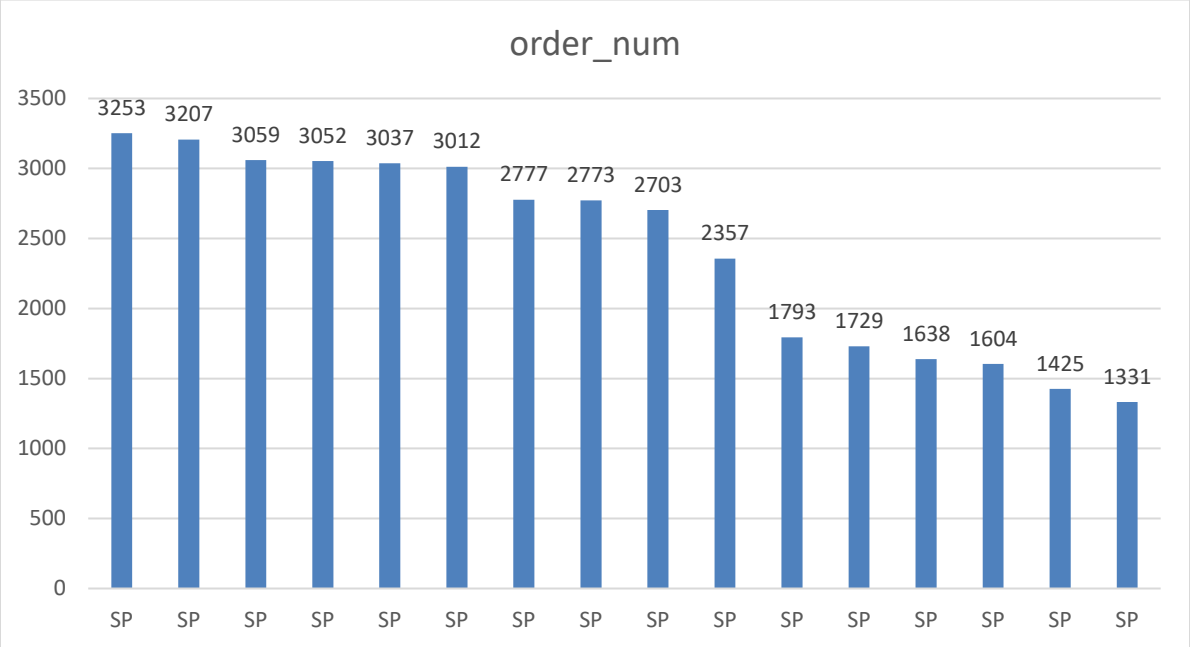
Minimum orders were from the following months

order_num	year_month	cust_state
1	2016-9	RR
1	2016-9	RS
1	2016-10	PB
1	2016-10	PI
1	2016-10	RR
1	2016-12	PR

Maximum orders were from the following months

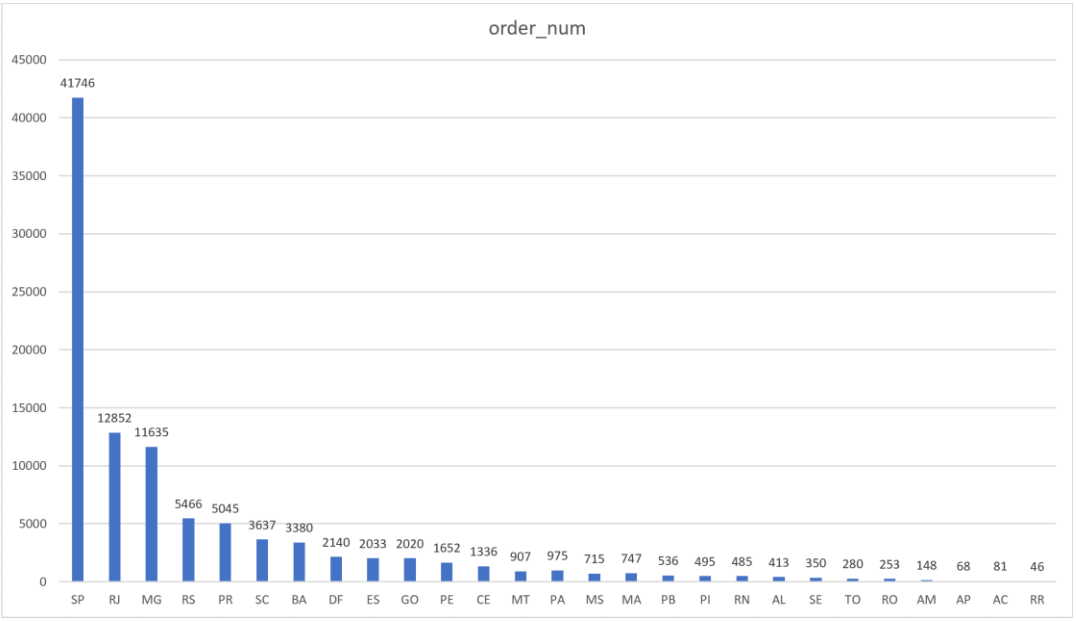
order_num	year_month	cust_state
3253	2018-8	SP
3207	2018-5	SP
3059	2018-4	SP
3052	2018-1	SP
3037	2018-3	SP
3012	2017-11	SP
2777	2018-7	SP
2773	2018-6	SP
2703	2018-2	SP
2357	2017-12	SP

Highest 16 orders are from the state ‘SP’

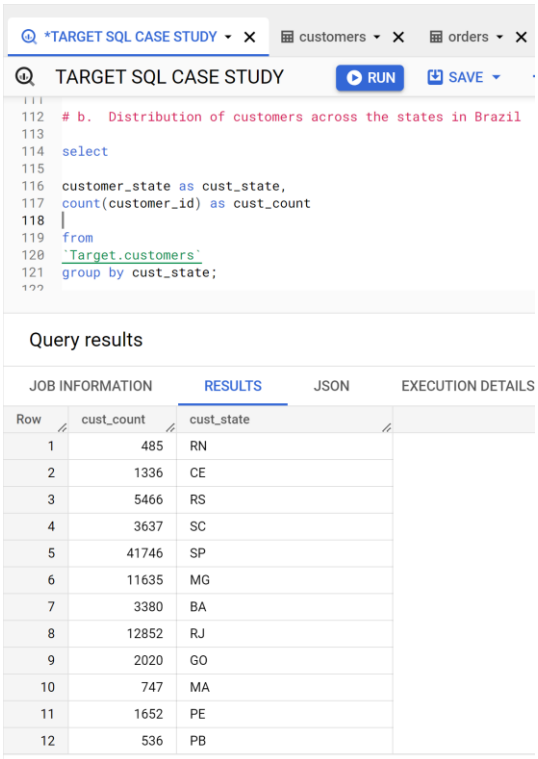


If we remove the year-month from the equation and aggregate the num of orders per state

cust_state	order_num
SP	41746
RJ	12852
MG	11635
RS	5466
PR	5045
SC	3637
BA	3380
DF	2140
ES	2033
GO	2020
PE	1652
CE	1336
MT	907
PA	975
MS	715
MA	747
PB	536
PI	495
RN	485
AL	413
SE	350
TO	280
RO	253
AM	148
AP	68
AC	81
RR	46



2. Distribution of customers across the states in Brazil

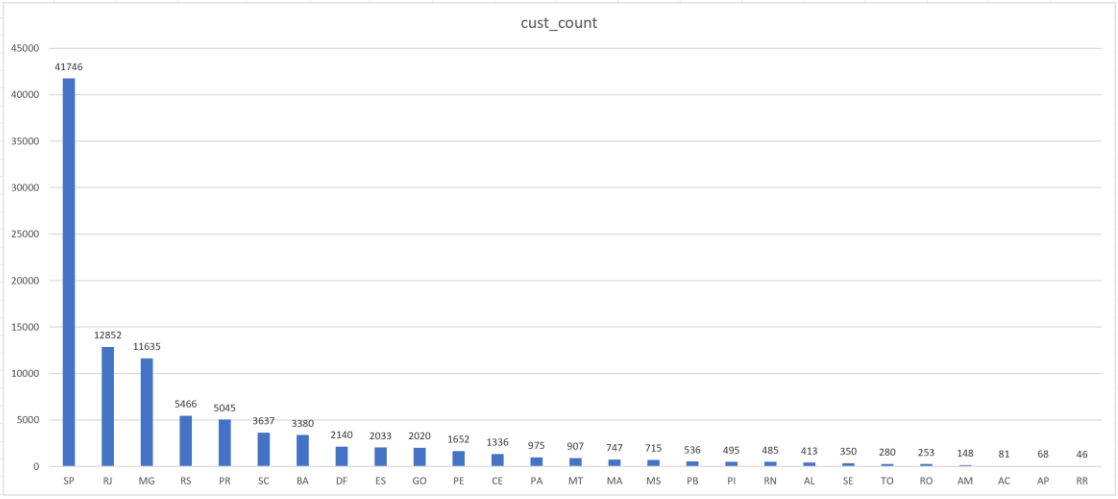


Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	cust_count	cust_state
1	485	RN
2	1336	CE
3	5466	RS
4	3637	SC
5	41746	SP
6	11635	MG
7	3380	BA
8	12852	RJ
9	2020	GO
10	747	MA
11	1652	PE
12	536	PB

cust_state	cust_count
SP	41746
RJ	12852
MG	11635
RS	5466
PR	5045
SC	3637
BA	3380
DF	2140
ES	2033
GO	2020
PE	1652
CE	1336
PA	975
MT	907
MA	747
MS	715
PB	536
PI	495
RN	485
AL	413
SE	350
TO	280
RO	253
AM	148
AC	81
AP	68
RR	46



Whether we use Order numbers per state or cust_id per state , the data points are same. 27 data points (states).

4. Impact on Economy: Analyze the money movement 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) - You can use “payment_value” column in payments table

Solution:

Prepare 2017 data, prepare 2018 data, compare both to find the % increase in cost of orders.

Query :

```
with

x as (

select
tot_price, month
from
(
select (price + freight_value) as tot_price,
extract(year from order_purchase_timestamp) as year,
extract(month from order_purchase_timestamp) as month

from `Target.orders` o1
join
`Target.order_items` o2

on o1.order_id = o2.order_id
) as a
```

```sql
where year = 2017 and month between 1 and 8 ) ,

y as (

select
tot_price, month
from
(
select (price + freight_value) as tot_price,
extract(year from order_purchase_timestamp) as year,
extract(month from order_purchase_timestamp) as month

from `Target.orders` o1
join
`Target.order_items` o2

on o1.order_id = o2.order_id
) as a
where year = 2018 and month between 1 and 8 )

# Formula
select round(((sum(y.tot_price) - sum(x.tot_price)) / sum(x.tot_price)) * 100,3) as percentage
_cost_increase
from x join y on x.month = y.month ;
```

Row	percentage_cost_increase
1	1.215

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

Query :

```
select

avg(price) as mean_price, sum(price) as tot_price,
avg(freight_value) as mean_frieght_value, sum(freight_value) as tot_frieght_value,
avg(price + freight_value) as mean_order_cost, sum(price + freight_value) as tot_order_cost,
customer_state

from `Target.order_items` x
join
`Target.orders` y
on x.order_id = y.order_id
join `Target.customers` c
on y.customer_id = c.customer_id

group by customer_state ;
```

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

Row	mean_price	tot_price	mean_frieght_va	tot_frieght_value	mean_order_cos	tot_order_cost	customer_state
1	148.297184...	156453.529...	28.1662843...	29715.4300...	176.463469...	186168.960...	MT
2	145.204150...	119648.219...	38.2570024...	31523.7700...	183.461152...	151171.990...	MA
3	180.889211...	80314.81	35.8436711...	15914.5899...	216.732882...	96229.3999...	AL
4	109.653629...	5202955.05...	15.1472753...	718723.069...	124.800904...	5921678.11...	SP
5	120.748574...	1585308.02...	20.6301668...	270853.460...	141.378740...	1856161.49...	MG
6	145.508322...	262788.029...	32.9178626...	59449.6599...	178.426184...	322237.690...	PE
7	125.117818...	1824092.66...	20.9609239...	305589.310...	146.078742...	2129681.98	RJ
8	125.770548...	302603.939...	21.0413549...	50625.4999...	146.811903...	353229.440...	DF
9	120.337453...	750304.020...	21.7358043...	135522.740...	142.073257...	885826.759...	RS
10	153.041168...	58920.8500...	36.6531688...	14111.4699...	189.694337...	73032.3199...	SE
11	119.004139...	683083.760...	20.5316515...	117851.680...	139.535790...	800935.439...	PR
12	165.692416...	178947.809...	35.8326851...	38699.3000...	201.525101...	217647.109...	PA

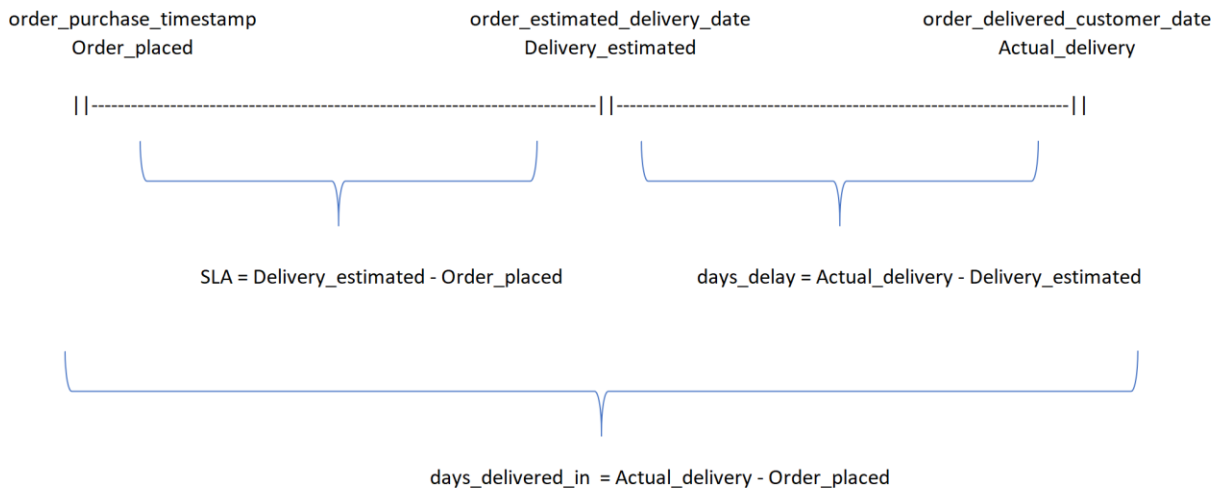
5. Analysis on sales, freight and delivery time

1. Calculate days between purchasing, delivering and estimated delivery

```
select
order_id,
date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as days_delivered_in,
date_diff(order_estimated_delivery_date, order_purchase_timestamp, day) as SLA,
date_diff(order_delivered_customer_date, order_estimated_delivery_date, day) as days_delay

from `Target.orders`
order by days_delivered_in desc, SLA, days_delay ;
```

Row	order_id	days_delivered_in	SLA	days_delay
1	ca07593549f1816d26a572e06...	209	28	181
2	1b3190b2dfa9d789e1f14c05b...	208	19	188
3	440d0d17af552815d15a9e41a...	195	30	165
4	285ab9426d6982034523a855f...	194	28	166
5	0f4519c5f1c541ddec9f21b3bd...	194	32	161
6	2fb597c2f772eca01b1f5c561b...	194	39	155
7	47b40429ed8cce3aee9199792...	191	15	175
8	2fe324febf907e3ea3f2aa9650...	189	22	167
9	2d7561026d542c8dbd8f0daea...	188	28	159
10	c27815f7e3dd0b926b5855262...	187	25	162
11	437222e3fd1b07396f1d9ba8c...	187	42	144
12	dfe5f68118c2576143240b8d7...	186	32	153
13	6e82dcfb5eada6283dba34f16...	182	27	155
14	2ba1366baecad3c3536f27546...	181	28	152



Out of 17876 rows of data fetched - Fastest delivered order :

S #	order_id	days_delivered_in	SLA	days_delay
14226	eec7f369423b033e549c02f3c5381205	20	155	-134

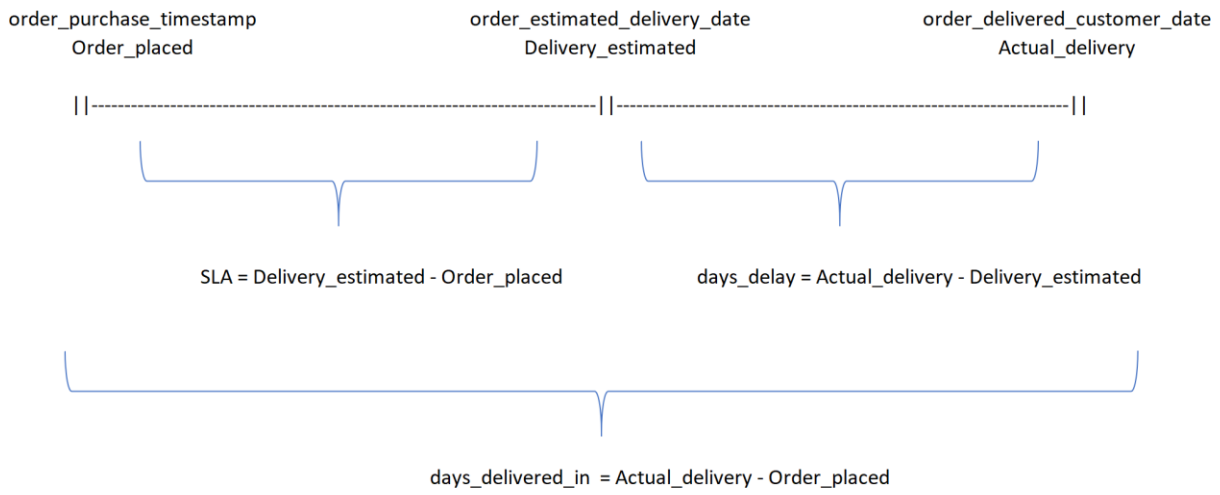
Order should have taken 155 days, but delivered in 20 days, saved 134 days.

Out of 17876 rows of data fetched – Most delayed order :

S #	order_id	days_delivered_in	SLA	days_delay
2	1b3190b2dfa9d789e1f14c05b647a14a	208	19	188

Order should have taken 19 days , but delivered in 208 days, delay of 188 days.

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



Formula seems to be wrong.

S #	order_id	days_delivered_in	SLA	days_delay	Minus Values
1195	ed638b613c111b199c3550cb29fd6bfa	44	45	-1	TRUE
1196	e282dabaac0b31c7da8a0896fd56e7f3	44	45	-1	TRUE
1605	0d40f8fd78977af4c4fa84293ae7fa21	41	43	-1	TRUE
1908	cb667db7bf699e2940c1ccdc16ef7ec7	39	40	-1	TRUE
1909	c6d26341bec0769b3ccfa232c63b3675	39	40	-1	TRUE

Formula should be :

time_to_delivery(`days_delivered_in`) =
order_delivered_customer_date - order_purchase_timestamp

diff_estimated_delivery(`days_delay`) =
order_delivered_customer_date - order_estimated_delivery_date
if days delay is in negative then order has arrived so many days early.

Query :

```
select
order_id,
date_diff(order_delivered_customer_date , order_purchase_timestamp, day) as time_to_delivery,
date_diff(order_delivered_customer_date, order_estimated_delivery_date, day) as diff_estimated
_delivery

from `Target.orders`
order by time_to_delivery desc, diff_estimated_delivery ;
```

Row	order_id	time_to_delivery	diff_estimated_delivery
1	ca07593549f1816d26a572e06...	209	181
2	1b3190b2dfa9d789e1f14c05b...	208	188
3	440d0d17af552815d15a9e41a...	195	165
4	2fb597c2f772eca01b1f5c561b...	194	155
5	0f4519c5f1c541ddec9f21b3bd...	194	161
6	285ab9426d6982034523a855f...	194	166
7	47b40429ed8cce3aee9199792...	191	175
8	2fe324febf907e3ea3f2aa9650...	189	167
9	2d7561026d542c8dbd8f0daea...	188	159
10	437222e3fd1b07396f1d9ba8c...	187	144
11	c27815f7e3dd0b926b5855262...	187	162
12	dfe5f68118c2576143240b8d7...	186	153
13	6e82dcfb5eada6283dba34f16...	182	155
14	2ba1366baecad3c3536f27546...	181	152
15	d24e8541128cea179a11a6517...	175	161
16	3566eabb132f8d64741ae7b92...	174	137

Points:

1. **time_to_delivery** – days in which the order is delivered from purchase timestamp to delivery date.
2. **diff_estimated_delivery** – is the delay of the order. After how many days past the estimate, the order is delivered.

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

Query :

```
select
c.customer_state,
round(avg(o1.freight_value),3) as mean_freight_value,
round(avg(date_diff(order_delivered_customer_date , order_purchase_timestamp, day)),3) as mean
_time_to_delivery,
round(avg(date_diff(order_delivered_customer_date, order_estimated_delivery_date, day)),3) as
mean_diff_estimated_delivery

from `Target.order_items` o1
join `Target.orders` o2 on o1.order_id = o2.order_id
join `Target.customers` c on o2.customer_id = c.customer_id

group by customer_state
order by mean_freight_value desc;
```

Mean means average, so take average of :

1. freight_value
2. days delivered in
3. days delay

Row	customer_state	mean_freight_value	mean_time_to_delivery	mean_diff_estimated_delivery
1	RR	42.984	27.826	-17.435
2	PB	42.724	20.119	-12.15
3	RO	41.07	19.282	-19.081
4	AC	40.073	20.33	-20.011
5	PI	39.148	18.931	-10.683
6	MA	38.257	21.204	-9.11
7	TO	37.247	17.003	-11.461
8	SE	36.653	20.979	-9.165
9	AL	35.844	23.993	-7.977
10	PA	35.833	23.302	-13.375
11	RN	35.652	18.873	-13.056
12	AP	34.006	27.753	-17.444

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

Query :

```
# Highest average freight value
select
c.customer_state,
round(avg(o1.freight_value),3) as mean_freight_value

from `Target.order_items` o1
join `Target.orders` o2 on o1.order_id = o2.order_id
join `Target.customers` c on o2.customer_id = c.customer_id

group by customer_state
order by mean_freight_value desc

limit 5;
```

Row	customer_state	mean_freight_value
1	RR	42.984
2	PB	42.724
3	RO	41.07
4	AC	40.073
5	PI	39.148

```
# Lowest average freight value

select
c.customer_state,
round(avg(o1.freight_value),3) as mean_freight_value

from `Target.order_items` o1
join `Target.orders` o2 on o1.order_id = o2.order_id
join `Target.customers` c on o2.customer_id = c.customer_id

group by customer_state
order by mean_freight_value

limit 5;
```

Row	customer_state	mean_freight_value
1	SP	15.147
2	PR	20.532
3	MG	20.63
4	RJ	20.961
5	DF	21.041

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

Query :

```
# Highest average time to delivery(days_delivered_in)
select
c.customer_state,
round(avg(date_diff(order_delivered_customer_date , order_purchase_timestamp, day)),3) as mean
_time_to_delivery,

from `Target.order_items` o1
join `Target.orders` o2 on o1.order_id = o2.order_id
join `Target.customers` c on o2.customer_id = c.customer_id

group by customer_state
order by mean_time_to_delivery desc
limit 5;
```

Row	customer_state	mean_time_to_delivery
1	RR	27.826
2	AP	27.753
3	AM	25.963
4	AL	23.993
5	PA	23.302

```
# Lowest average time to delivery(days_delivered_in)
select
c.customer_state,
round(avg(date_diff(order_delivered_customer_date , order_purchase_timestamp, day)),3) as mean
_time_to_delivery,

from `Target.order_items` o1
join `Target.orders` o2 on o1.order_id = o2.order_id
join `Target.customers` c on o2.customer_id = c.customer_id

group by customer_state
order by mean_time_to_delivery
limit 5;
```

Row	customer_state	mean_time_to_delivery
1	SP	8.26
2	PR	11.481
3	MG	11.516
4	DF	12.501
5	SC	14.521

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

Query :

```
# Highest average delay
select
c.customer_state,
round(avg(date_diff(order_delivered_customer_date, order_estimated_delivery_date, day)),3) as
mean_diff_estimated_delivery

from `Target.order_items` o1
join `Target.orders` o2 on o1.order_id = o2.order_id
join `Target.customers` c on o2.customer_id = c.customer_id

group by customer_state
order by mean_diff_estimated_delivery

limit 5;
```

Row	customer_state	mean_diff_estimated_delivery
1	AC	-20.011
2	RO	-19.081
3	AM	-18.975
4	AP	-17.444
5	RR	-17.435

```
# Lowest average delay
select
c.customer_state,
round(avg(date_diff(order_delivered_customer_date, order_estimated_delivery_date, day)),3) as
mean_diff_estimated_delivery

from `Target.order_items` o1
join `Target.orders` o2 on o1.order_id = o2.order_id
join `Target.customers` c on o2.customer_id = c.customer_id

group by customer_state
order by mean_diff_estimated_delivery desc

limit 5;
```

Row	customer_state	mean_diff_estim
1	AL	-7.977
2	MA	-9.11
3	SE	-9.165
4	ES	-9.769
5	BA	-10.119

6. Payment type analysis:

1. Month over Month count of orders for different payment types

Query :

```
# Full Query

select
extract(month from o.order_purchase_timestamp) as month,
extract(year from o.order_purchase_timestamp) as year,
concat(extract(month from o.order_purchase_timestamp),' - ', extract(year from o.order_purchas
e_timestamp)) as month_year,
count(o.order_id) as ord_count,
p.payment_type
from
`Target.orders` o
join
`Target.payments` p
on o.order_id = p.order_id

group by month, year, month_year, p.payment_type
order by year, month ;
```

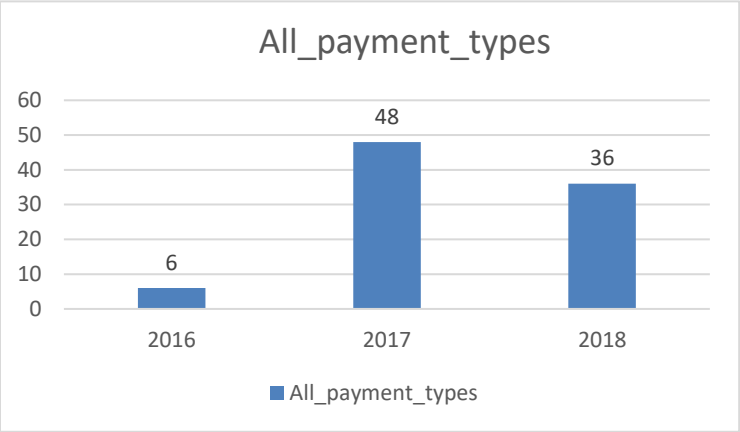
Note: using the year and month in select clause just to do the sorting in order by.

Row	month	year	month_year	ord_count	payment_type
1	9	2016	9 - 2016	3	credit_card
2	10	2016	10 - 2016	254	credit_card
3	10	2016	10 - 2016	63	UPI
4	10	2016	10 - 2016	23	voucher
5	10	2016	10 - 2016	2	debit_card
6	12	2016	12 - 2016	1	credit_card
7	1	2017	1 - 2017	583	credit_card

Extra Analysis:

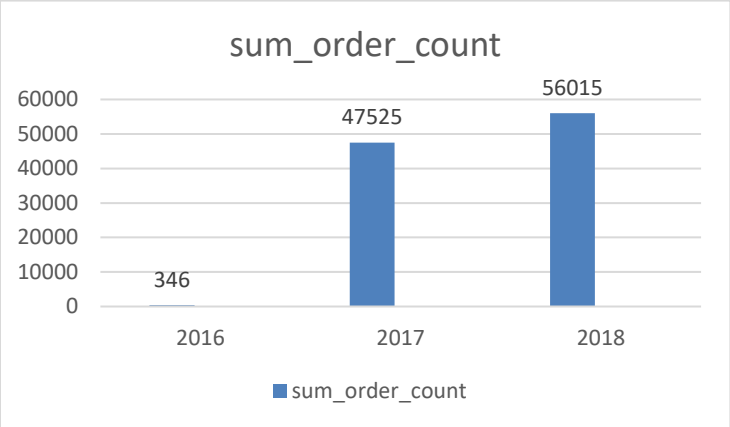
- Year-wise split of count of all payments types

year	***All_payment_types***
2016	6
2017	48
2018	36



- Year-wise split of sum of orders

year	***sum_order_count***
2016	346
2017	47525
2018	56015



2. Count of orders based on the no. of payment installments

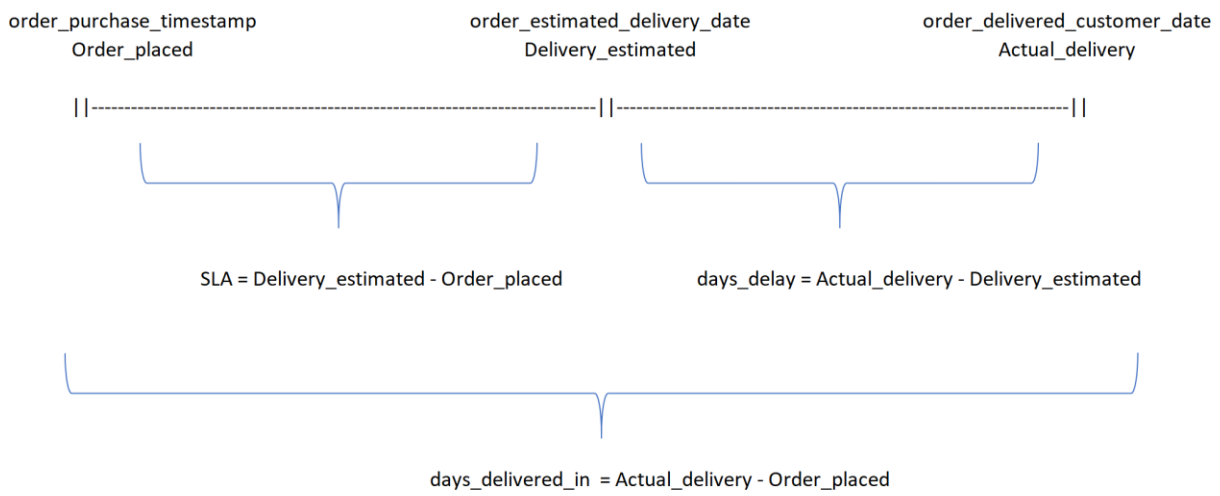
Query :

```
select *
from `Target.payments` ;

select
count(order_id) as num_of_orders,
payment_type,
payment_installments
from `Target.payments`
group by payment_installments, payment_type ;
```

Row	num_of_orders	payment_type	payment_installments
1	2	credit_card	0
2	5775	voucher	1
3	3	not_defined	1
4	25455	credit_card	1
5	1529	debit_card	1
6	19784	UPI	1
7	12413	credit_card	2
8	10461	credit_card	3
9	7098	credit_card	4
10	5239	credit_card	5
11	3920	credit_card	6
12	1626	credit_card	7

ACTIONABLE INSIGHTS :



1. Actual delivery – delivery estimated gives you delay / faster delivery in days
2. Actual delivery – order purchase timestamp gives you days delivered in.
3. Based on the above points, target can ramp up staff during peak months and ask employees to take holidays during Sep, Dec months.
4. Reduce delays, by planning logistic pipelines. Based on the trend, keep a par stock in the local warehouse, all those items that are sure to be sold / necessary so as to reduce delay and increase customer satisfaction.
5. Give away some gifts during festivals and make target a household brand.

RECOMMENDATIONS :

1. They should start 24 outlets at-least during Christmas to make lot of sales
2. Open an app / loyalty program and give discounts, coupons to retain customers.
3. Train staff to be more friendly and help elders in their shopping.
4. Organize events in some parts of the cities. Go to the customers, rather than customers coming to store to buy.