SQL BUSINESS CASE:

A globally renowned brand and a prominent retailer in the United States. The company makes itself a preferred shopping destination by offering outstanding value, inspiration, innovation and an exceptional guest experience that no other retailer can deliver.

This particular business case focuses on the operations of retailer in Brazil and provides insightful information about 100,000 orders placed between 2016 and 2018. The dataset offers a comprehensive view of various dimensions including the order status, price, payment and freight performance, customer location, product attributes, and customer reviews.

By analyzing this extensive dataset, it becomes possible to gain valuable insights into retailer's operations in Brazil. The information can shed light on various aspects of the business, such as order processing, pricing strategies, payment and shipping efficiency, customer demographics, product characteristics, and customer satisfaction levels.

Business problem:

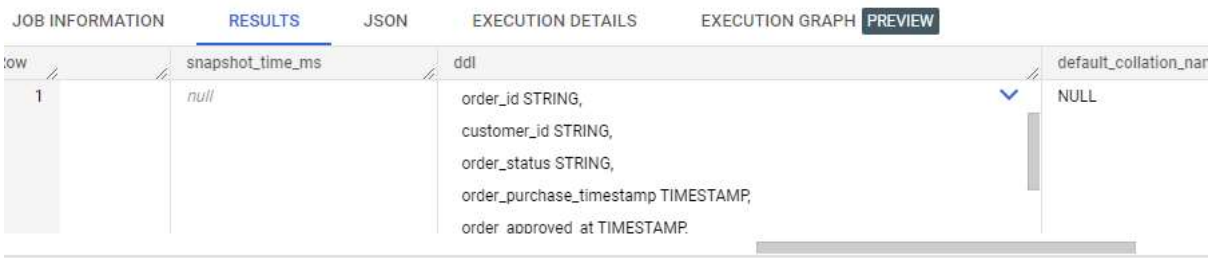
Analyse the given dataset to extract valuable insights and provide actionable recommendations.

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

1.1: **Data type of columns in a table:**

Query:

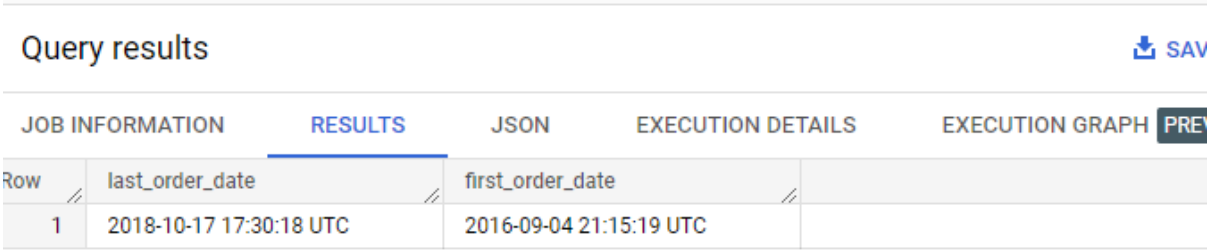
```
select *

from `t1-sql-casestudy-387120.t1.INFORMATION_SCHEMA.TABLES`

WHERE TABLE_NAME='orders'
```



1.2: **Time period for which the data is given:** I ran below query to obtain time period as 2016-09-04 to 2018-10-17

Query:

```
select
max(order_purchase_timestamp) as last_order_date,
min(order_purchase_timestamp) as first_order_date
from `t1-sql-casestudy-387120.t1.orders`
```



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

1.3 **Cities and States of customers ordered during the given period:** Performed join on customers table and orders table through customer_id to find customers who have ordered during the entire period and then grouped by city and states and displayed counts of customers who ordered in that city and state for a cleaner look, as customer_id is long and in string format

Query:

```
select count(ord.customer_id) as count_of_customers,cust.customer_city,cust.customer_state
from `t1-sql-casestudy-387120.t1.customers` as cust
inner join `t1-sql-casestudy-387120.t1.orders` as ord
on cust.customer_id=ord.customer_id
group by 2,3
order by 3,2 asc
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTIC

Row	count_of_custor	customer_city	customer_state
1	1	brasileia	AC
2	3	cruzeiro do sul	AC
3	1	epitaciolandia	AC
4	1	manoel urbano	AC
5	1	porto acre	AC
6	70	rio branco	AC
7	2	senador guiomard	AC
8	2	xapuri	AC
9	1	agua branca	AL
10	2	anadia	AL
11	29	arapiraca	AL
12	1	atalaia	AL
13	2	barra de santo antonio	AL

Insights: Cities like Sao Paulo, Rio de janeiro have highest customer base which suggests to increase the number of stores in these cities to further improve customer base. There are many states where the customers are <500 and in these states the company should focus on advertising, addressing any delivery issues.

ow	count_of_customers	customer_city	customer_state
1	15540	sao paulo	SP
2	6882	rio de janeiro	RJ
3	2773	belo horizonte	MG
4	2131	brasilia	DF
5	1521	curitiba	PR

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION

Row	count_of_customers	customer_city	customer_state
4201	128	sete lagoas	MG
4202	130	paulinia	SP
4203	131	pouso alegre	MG
4204	133	sao jose dos pinhais	PR
4205	134	sao joao de meriti	RJ
4206	135	jaboatao dos guararapes	PE
4207	136	itu	SP
4208	137	divinopolis	MG
4209	137	rio das ostras	RJ
4210	139	marica	RJ

2. In-depth Exploration:

2.1 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?

Exploration: there was a growing trend in e-commerce from January, 2017 to November, 2017. Later there were minor fluctuations during H1 of 2018, however, there was a drastic drop in orders after August, 2018 which further reduced by October, 2018.

Peak sale was observed during November 2017 and January and March 2018 with next lower sales in December 2017 and February-August 2018

Complete scenario: Overall 2017 and H1 2018 were successful years in terms of orders

Query:

```
select count(order_id) as count_of_orders,
extract(month from order_purchase_timestamp) as month,
extract(year from order_purchase_timestamp) as year
from `t1-sql-casestudy-387120.t1.orders`
group by 3,2
order by 3,2 asc
```

Row	count_of_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
10	4026	7	2017
11	4331	8	2017
12	4285	9	2017
13	4631	10	2017

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

Insight: Brazilian customers tend to buy their products mostly during Afternoon followed by Night. Few customers prefer to order during Dawn

Recommendation: Offers and discount periods can be applied during Afternoons more to improve sales.

Query:

```
select count(order_id) as count_of_orders,a.timings
from
(select *,
case when extract(time from order_purchase_timestamp) between "4:00:00" and "6:00:00"then
'Dawn'
     when extract(time from order_purchase_timestamp) between "6:00:01" and "12:00:00" then
'Morning'
     when extract(time from order_purchase_timestamp) between "12:00:01" and "19:00:00"
then 'Afternoon'
     else 'Night' end as timings
from `t1-sql-casestudy-387120.t1.orders`) as a
group by 2
order by 1
```

JOB INFORMATION | RESULTS | JSON

Row	count_of_orders	timings
1	394	Dawn
2	22240	Morning
3	32677	Night
4	44130	Afternoon

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

3.1: Get month on month orders by states

Query:

```
select
extract(month from ord.order_purchase_timestamp) as month,
extract(year from ord.order_purchase_timestamp) as year,
count(ord.order_id) as count_of_orders,
cust.customer_state
from `t1-sql-casestudy-387120.t1.orders` as ord
join `t1-sql-casestudy-387120.t1.customers` as cust
on cust.customer_id=ord.customer_id
group by 4,2,1
```

```
order by 2,1
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECU

Row	month	year	count_of_orders	customer_state
1	9	2016	1	RR
2	9	2016	1	RS
3	9	2016	2	SP
4	10	2016	113	SP
5	10	2016	24	RS
6	10	2016	56	RJ
7	10	2016	3	MT
8	10	2016	9	GO
9	10	2016	40	MG
10	10	2016	8	CE
11	10	2016	11	SC

3.2 Distribution of customers across the states in Brazil

Insights: Highest customers are in state SP followed by RJ and MG. Less than 500 customers are present in states like PI,RN,AL,SE etc., where the company can focus on advertising and campaigning about its stores products in order to increase customers and proportionally sales

Query:

```
select count(ord.customer_id) as count_of_customers,cust.customer_state
from `t1-sql-casestudy-387120.t1.customers` as cust
inner join `t1-sql-casestudy-387120.t1.orders` as ord
on cust.customer_id=ord.customer_id
group by 2
order by 1 desc
```

Row	count_of_customers	customer_state
1	41746	SP
2	12852	RJ
3	11635	MG
4	5466	RS
5	5045	PR
6	3637	SC
7	3380	BA
8	2140	DF
9	2033	ES
10	2020	GO
11	1652	PE

Row	count_of_customers	customer_state
17	536	PB
18	495	PI
19	485	RN
20	413	AL
21	350	SE
22	280	TO
23	253	RO
24	148	AM
25	81	AC
26	68	AP
27	46	RR

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

4.1 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

Insights: 137% increase in cost of orders was observed from 2017 to 2018 during January to August months.

Query:

```
select ((cost2018-cost2017)/cost2017)*100 as percent_inc
  from
  (select sum(case when month in (1,2,3,4,5,6,7,8) and year=2017 then payment_value end) as
cost2017,
        sum(case when month in (1,2,3,4,5,6,7,8) and year=2018 then payment_value end) as
cost2018
```

```
from
(select *,
extract(month from ord.order_purchase_timestamp) as month,
extract(year from ord.order_purchase_timestamp) as year
from `t1-sql-casestudy-387120.t1.orders` as ord
join `t1-sql-casestudy-387120.t1.payments` as pay
on ord.order_id=pay.order_id) as A) as B
```

Row	percent_inc
1	136.9768716466...

4.2 **Mean & Sum of price and freight value by customer state**

Query:

```
select cust.customer_state as state,

avg(ordit.price) as mean_price,
sum(ordit.price) as sum_price,
avg(ordit.freight_value) as mean_frieght,
sum(ordit.freight_value) as sum_frieght
from `t1-sql-casestudy-387120.t1.orders` as ord
join t1-sql-casestudy-387120.t1.order_items as ordit
on ord.order_id=ordit.order_id
join t1-sql-casestudy-387120.t1.customers as cust
on cust.customer_id=ord.customer_id
group by 1
order by 1
```

Row	state	mean_price	sum_price	mean_frieght	sum_frieght
1	AC	173.7277173913...	15982.94999999...	40.07336956521...	3686.749999999...
2	AL	180.8892117117...	80314.81	35.84367117117...	15914.58999999...
3	AM	135.4959999999...	22356.84000000...	33.20539393939...	5478.889999999...
4	AP	164.3207317073...	13474.29999999...	34.00609756097...	2788.500000000...
5	BA	134.6012082126...	511349.9900000...	26.36395893656...	100156.6799999...
6	CE	153.7582611637...	227254.7099999...	32.71420162381...	48351.58999999...
7	DF	125.7705486284...	302603.9399999...	21.04135494596...	50625.49999999...
8	ES	121.9137012411...	275037.3099999...	22.05877659574...	49764.59999999...
9	GO	126.2717316759...	294591.9499999...	22.76681525932...	53114.97999999...
10	MA	145.2041504854...	119648.2199999...	38.25700242718...	31523.77000000...

5. Analysis on sales, freight and delivery time

5.1 Calculate days between purchasing, delivering and estimated delivery

Query:

```
select order_purchase_timestamp,
  order_delivered_customer_date,
  order_estimated_delivery_date,
  date_diff(order_delivered_customer_date,order_purchase_timestamp,day) as delivery_days,
  date_diff(order_estimated_delivery_date,order_purchase_timestamp,day) as
estimated_delivery_days,
  date_diff(order_delivered_customer_date,order_estimated_delivery_date,day) as
diff_estimated_delivery
from `t1-sql-casestudy-387120.t1.orders`
where order_delivered_customer_date is not null
order by 4,5,6
```

First 10 rows result

Row	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	delivery_days	estimated_delivery_days	diff_estimated_delivery
1	2017-05-15 11:50:53 UTC	2017-05-16 10:21:52 UTC	2017-05-24 00:00:00 UTC	0	8	-7
2	2018-06-18 12:59:42 UTC	2018-06-19 12:43:27 UTC	2018-06-28 00:00:00 UTC	0	9	-8
3	2017-06-19 08:19:45 UTC	2017-06-19 21:07:52 UTC	2017-06-30 00:00:00 UTC	0	10	-10
4	2018-05-18 15:03:19 UTC	2018-05-19 12:28:30 UTC	2018-05-29 00:00:00 UTC	0	10	-9
5	2018-05-14 12:20:06 UTC	2018-05-15 12:17:46 UTC	2018-05-25 00:00:00 UTC	0	10	-9
6	2017-11-16 13:54:08 UTC	2017-11-17 13:49:40 UTC	2017-11-29 00:00:00 UTC	0	12	-11
7	2017-07-04 11:37:47 UTC	2017-07-05 08:09:26 UTC	2017-07-17 00:00:00 UTC	0	12	-11
8	2017-05-31 12:00:35 UTC	2017-06-01 10:28:24 UTC	2017-06-13 00:00:00 UTC	0	12	-11
9	2018-06-28 14:34:48 UTC	2018-06-29 14:12:18 UTC	2018-07-12 00:00:00 UTC	0	13	-12
10	2018-02-02 15:26:38 UTC	2018-02-03 15:05:56 UTC	2018-02-20 00:00:00 UTC	0	17	-16

Middle rows result for clear picture

Row	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	delivery_days	estimated_delivery_days	diff_estimated_delivery
96446	2017-04-30 17:28:58 UTC	2017-09-19 16:49:30 UTC	2017-05-31 00:00:00 UTC	141	30	111
96447	2017-11-25 12:14:38 UTC	2018-04-16 23:26:51 UTC	2017-12-27 00:00:00 UTC	142	31	110
96448	2017-04-28 16:28:03 UTC	2017-09-19 13:54:18 UTC	2017-05-30 00:00:00 UTC	143	31	112
96449	2018-05-21 06:48:46 UTC	2018-10-11 16:41:14 UTC	2018-06-27 00:00:00 UTC	143	36	106
96450	2017-06-03 17:53:31 UTC	2017-10-26 20:47:58 UTC	2017-06-27 00:00:00 UTC	145	23	121
96451	2017-04-26 10:41:39 UTC	2017-09-19 18:12:28 UTC	2017-05-23 00:00:00 UTC	146	28	119
96452	2018-02-26 12:30:39 UTC	2018-07-26 16:41:56 UTC	2018-03-22 00:00:00 UTC	148	21	126
96453	2017-11-25 23:51:54 UTC	2018-05-10 00:06:20 UTC	2017-12-18 00:00:00 UTC	165	22	143
96454	2017-04-06 12:59:46 UTC	2017-09-19 16:23:52 UTC	2017-05-19 00:00:00 UTC	166	42	123
96455	2017-04-04 23:21:02 UTC	2017-09-19 13:47:09 UTC	2017-05-04 00:00:00 UTC	167	29	138

Insights: The average delivery dates reduced gradually during 2017 and further during H1 of 2018 which sometimes affects the customer purchase. The less the delivery time, more will be orders placed in few instances.

Row	month	year	f0_	days
1	9	2016	1	54.0
2	10	2016	270	19.11111111111...
3	12	2016	1	4.0
4	1	2017	750	12.09200000000...
5	2	2017	1653	12.60617059891...
6	3	2017	2546	12.39552238805...
7	4	2017	2303	14.35258358662...
8	5	2017	3545	10.76050775740...
9	6	2017	3135	11.50622009569...
10	7	2017	3872	11.13119834710...
11	8	2017	4193	10.70069162890...
12	9	2017	4150	11.40072289156...

5.2 Find time_to_delivery & diff_estimated_delivery.

Insights: A negative diff_estimated_delivery indicates the order was delivered before estimated delivery date. A positive diff_estimated_delivery indicates the order was delivered for #days after the estimated delivery date has passed. This kind of data will allow the company to check for any loopholes and delays during delivery, issues with carrier delivery, number of delivery executives present in that city, availability/unavailability of product at nearest ware-house to the customer_city

Most of the orders were delivered within estimated delivery date during 2017 and 2018

Query:

```
select order_purchase_timestamp,
  order_delivered_customer_date,
  order_estimated_delivery_date,
  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 `t1-sql-casestudy-387120.t1.orders`
where order_delivered_customer_date is not null
order by 3,4
```

First 10 rows result

low	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	time_to_delivery	diff_estimated_deliv
1	2016-09-15 12:16:38 UTC	2016-11-09 07:47:38 UTC	2016-10-04 00:00:00 UTC	54	36
2	2016-10-03 09:44:50 UTC	2016-10-26 14:02:13 UTC	2016-10-27 00:00:00 UTC	23	0
3	2016-10-03 16:56:50 UTC	2016-10-27 18:19:38 UTC	2016-11-07 00:00:00 UTC	24	-10
4	2016-10-05 12:32:55 UTC	2016-10-13 16:03:46 UTC	2016-11-23 00:00:00 UTC	8	-40
5	2016-10-03 22:31:31 UTC	2016-10-14 16:08:00 UTC	2016-11-23 00:00:00 UTC	10	-39
6	2016-10-03 22:06:03 UTC	2016-10-31 11:07:42 UTC	2016-11-23 00:00:00 UTC	27	-22
7	2016-10-04 16:40:07 UTC	2016-10-13 15:56:11 UTC	2016-11-24 00:00:00 UTC	8	-41
8	2016-10-04 22:15:11 UTC	2016-10-13 16:00:43 UTC	2016-11-24 00:00:00 UTC	8	-41
9	2016-10-04 15:10:15 UTC	2016-10-13 15:56:28 UTC	2016-11-24 00:00:00 UTC	9	-41

5.3: Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery

I joined orders, order-items and customers table and calculated average freight value, time to delivery, diff estimated delivery

Insights: Lower the freight value, less is the time taken for delivery, higher the freight value longer is the time taken for delivery. This provides the company to investigate into freight price section, delivery carriers and find loopholes and re-organise such process to improve efficiency

Query:

```
select count(ord.order_id) as count_orders,
  cust.customer_state as state,
  avg(ordit.freight_value) as avg_frieght,
  avg(date_diff(ord.order_delivered_customer_date,ord.order_purchase_timestamp,DAY)) as
time_to_delivery ,
  avg(date_diff(ord.order_estimated_delivery_date,ord.order_delivered_customer_date,DAY))
as diff_estimated_delivery
  from `t1-sql-casestudy-387120.t1.orders` as ord
  join t1-sql-casestudy-387120.t1.order_items as ordit
  on ord.order_id=ordit.order_id
  join t1-sql-casestudy-387120.t1.customers as cust
  on cust.customer_id=ord.customer_id
  where ord.order_delivered_customer_date is not null
  group by 2
  order by 2
```

Row	count_orders	state	avg_frieght	time_to_delivery	diff_estimated_delivery
1	91	AC	40.047912087912081	20.32967032967...	20.010989010989018
2	427	AL	35.870655737704922	23.99297423887...	7.9765807962529349
3	163	AM	33.310613496932525	25.96319018404...	18.975460122699381
4	81	AP	34.1604938271605	27.75308641975...	17.444444444444443
5	3683	BA	26.487556339940287	18.77464023893...	10.119467825142538
6	1426	CE	32.734495091164128	20.53716690042...	10.256661991584851
7	2355	DF	21.072161358811066	12.50148619957...	11.274734607218704
8	2225	ES	22.02897977528092	15.19280898876...	9.7685393258427116
9	2277	GO	22.562867808519979	14.94817742643...	11.372859025032927
10	800	MA	38.492712500000032	21.20375000000...	9.1099999999999923
11	12917	MG	20.6258372687155	11.51552218007...	12.397151041263502

Row	count_orders	state	avg_frieght	time_to_delivery	diff_estimated_delivery
17	523	PI	39.115086042064924	18.93116634799...	10.682600382409184
18	5649	PR	20.471816250663817	11.48079306071...	12.533899805275263
19	14146	RJ	20.909784391347358	14.68938215750...	11.14449314293797
20	521	RN	35.71808061420348	18.87332053742...	13.055662188099804
21	273	RO	41.330549450549434	19.28205128205...	19.080586080586084
22	46	RR	43.088043478260865	27.82608695652...	17.434782608695652
23	6133	RS	21.61427034077937	14.70829936409...	13.203000163052323
24	4098	SC	21.506627623230841	14.52098584675...	10.6688628599317
25	375	SE	36.573173333333358	20.97866666666...	9.1653333333333276
26	46443	SP	15.114994078763218	8.259608552419...	10.26559438451439
27	310	TO	37.435032258064496	17.00322580645...	11.461290322580641

5.4 Sort the data to get the following:

5.5 Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5

Performed CTE to obtain highest and lowest

Query: for TOP 5 states with highest average freight value

```
with combined_data as (
    select count(ord.order_id) as count_orders,
  cust.customer_state as state,
  avg(ordit.freight_value) as avg_frieght,
  avg(date_diff(ord.order_delivered_customer_date,ord.order_purchase_timestamp,DAY)) as
time_to_delivery ,
  avg(date_diff(ord.order_estimated_delivery_date,ord.order_delivered_customer_date,DAY))
as diff_estimated_delivery
  from `t1-sql-casestudy-387120.t1.orders` as ord
  join t1-sql-casestudy-387120.t1.order_items as ordit
  on ord.order_id=ordit.order_id
  join t1-sql-casestudy-387120.t1.customers as cust
  on cust.customer_id=ord.customer_id
  where ord.order_delivered_customer_date is not null
  group by 2
  order by 2
  )
  select state,avg_frieght
  from combined_data
  order by 2 desc
  limit 5
```

Row	state	avg_frieght
1	PB	43.09168941979...
2	RR	43.08804347826...
3	RO	41.33054945054...
4	AC	40.04791208791...
5	PI	39.11508604206...

Query: for TOP 5 states with lowest average freight value

```
 with combined_data as (
    select count(ord.order_id) as count_orders,
  cust.customer_state as state,
  avg(ordit.freight_value) as avg_frieght,
  avg(date_diff(ord.order_delivered_customer_date,ord.order_purchase_timestamp,DAY)) as
time_to_delivery ,
  avg(date_diff(ord.order_estimated_delivery_date,ord.order_delivered_customer_date,DAY))
as diff_estimated_delivery
  from `t1-sql-casestudy-387120.t1.orders` as ord
  join t1-sql-casestudy-387120.t1.order_items as ordit
  on ord.order_id=ordit.order_id
  join t1-sql-casestudy-387120.t1.customers as cust
  on cust.customer_id=ord.customer_id
  where ord.order_delivered_customer_date is not null
  group by 2
```

```
  order by 2
  )
  select state,avg_frieght
  from combined_data
  order by 2 asc
  limit 5
```

Row	state	avg_frieght
1	PB	43.09168941979...
2	RR	43.08804347826...
3	RO	41.33054945054...
4	AC	40.04791208791...
5	PI	39.11508604206...

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

Query: Top 5 states with highest avg time to delivery

```
    with combined_data as (
    select count(ord.order_id) as count_orders,
  cust.customer_state as state,
  avg(ordit.freight_value) as avg_frieght,
  avg(date_diff(ord.order_delivered_customer_date,ord.order_purchase_timestamp,DAY)) as
time_to_delivery ,
  avg(date_diff(ord.order_estimated_delivery_date,ord.order_delivered_customer_date,DAY))
as diff_estimated_delivery
  from `t1-sql-casestudy-387120.t1.orders` as ord
  join t1-sql-casestudy-387120.t1.order_items as ordit
  on ord.order_id=ordit.order_id
  join t1-sql-casestudy-387120.t1.customers as cust
  on cust.customer_id=ord.customer_id
  where ord.order_delivered_customer_date is not null
  group by 2
  order by 2
  )
  select state,time_to_delivery
  from combined_data
  order by 2 desc
  limit 5
```

Row	state	time_to_delivery
1	RR	27.82608695652...
2	AP	27.75308641975...
3	AM	25.96319018404...
4	AL	23.99297423887...
5	PA	23.30170777988...

Query: Top 5 states with lowest avg time to delivery

```
    with combined_data as (
    select count(ord.order_id) as count_orders,
  cust.customer_state as state,
  avg(ordit.freight_value) as avg_frieght,
  avg(date_diff(ord.order_delivered_customer_date,ord.order_purchase_timestamp,DAY)) as
time_to_delivery ,
  avg(date_diff(ord.order_estimated_delivery_date,ord.order_delivered_customer_date,DAY))
as diff_estimated_delivery
  from `t1-sql-casestudy-387120.t1.orders` as ord
  join t1-sql-casestudy-387120.t1.order_items as ordit
  on ord.order_id=ordit.order_id
  join t1-sql-casestudy-387120.t1.customers as cust
  on cust.customer_id=ord.customer_id
  where ord.order_delivered_customer_date is not null
  group by 2
  order by 2
  )
  select state,time_to_delivery
  from combined_data
  order by 2 asc
  limit 5
```

Row	state	time_to_delivery
1	SP	8.259608552419...
2	PR	11.48079306071...
3	MG	11.51552218007...
4	DF	12.50148619957...
5	SC	14.52098584675...

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

Query: Top 5 states where delivery is really fast

```
   with combined_data as (
    select count(ord.order_id) as count_orders,
  cust.customer_state as state,
  avg(ordit.freight_value) as avg_frieght,
  avg(date_diff(ord.order_delivered_customer_date,ord.order_purchase_timestamp,DAY)) as
time_to_delivery ,
  avg(date_diff(ord.order_estimated_delivery_date,ord.order_delivered_customer_date,DAY))
as diff_estimated_delivery
  from `t1-sql-casestudy-387120.t1.orders` as ord
  join t1-sql-casestudy-387120.t1.order_items as ordit
  on ord.order_id=ordit.order_id
  join t1-sql-casestudy-387120.t1.customers as cust
  on cust.customer_id=ord.customer_id
  where ord.order_delivered_customer_date is not null
  group by 2
  order by 2
  )


  select state,diff_estimated_delivery
```

```
  from combined_data
  order by 2 asc
  limit 5
```

Row	state	diff_estimated_delivery
1	AL	7.9765807962529349
2	MA	9.1099999999999923
3	SE	9.1653333333333276
4	ES	9.7685393258427116
5	BA	10.119467825142538

Query: Top 5 states where delivery is not so fast

```
 with combined_data as (
    select count(ord.order_id) as count_orders,
  cust.customer_state as state,
  avg(ordit.freight_value) as avg_frieght,
  avg(date_diff(ord.order_delivered_customer_date,ord.order_purchase_timestamp,DAY)) as
time_to_delivery ,
  avg(date_diff(ord.order_estimated_delivery_date,ord.order_delivered_customer_date,DAY))
as diff_estimated_delivery
  from `t1-sql-casestudy-387120.t1.orders` as ord
  join t1-sql-casestudy-387120.t1.order_items as ordit
  on ord.order_id=ordit.order_id
  join t1-sql-casestudy-387120.t1.customers as cust
  on cust.customer_id=ord.customer_id
  where ord.order_delivered_customer_date is not null
  group by 2
  order by 2
  )
  select state,diff_estimated_delivery
  from combined_data
  order by 2 desc
  limit 5
```

Row	state	diff_estimated_delivery
1	AC	20.010989010989018
2	RO	19.080586080586084
3	AM	18.975460122699381
4	AP	17.444444444444443
5	RR	17.434782608695652

6. Payment type analysis:

6.1 Month over Month count of orders for different payment types

Query:

Insights: Most of the payments made were through credit card

```
select pay.payment_type,
        extract (month from ord.order_purchase_timestamp) as month,
        extract (year from ord.order_purchase_timestamp) as year,
        count(ord.order_id) as count_of_orders
from `t1-sql-casestudy-387120.t1.payments` as pay
inner join `t1-sql-casestudy-387120.t1.orders` as ord
on pay.order_id=ord.order_id
group by 2,3,1
order by 3,2,1
```

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

Row	payment_type	month	year	count_of_orders
1	credit_card	9	2016	3
2	UPI	10	2016	63
3	credit_card	10	2016	254
4	debit_card	10	2016	2
5	voucher	10	2016	23
6	credit_card	12	2016	1
7	UPI	1	2017	197
8	credit_card	1	2017	583
9	debit_card	1	2017	9
10	voucher	1	2017	61
11	UPI	2	2017	398

6.2: Count of orders based on the no. of payment instalments

Query:

```
select payment_installments, count(order_id) as count_of_orders
from `t1-sql-casestudy-387120.t1.payments`
group by payment_installments
```

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

Row	payment_installr	count_of_orders
1	0	2
2	1	49060
3	2	12389
4	3	10443
5	4	7088
6	5	5234
7	6	3916
8	7	1623

Results per page: 50 ▼ 1 – 24 o