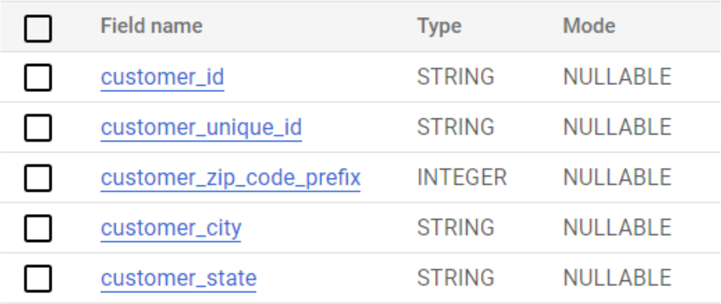
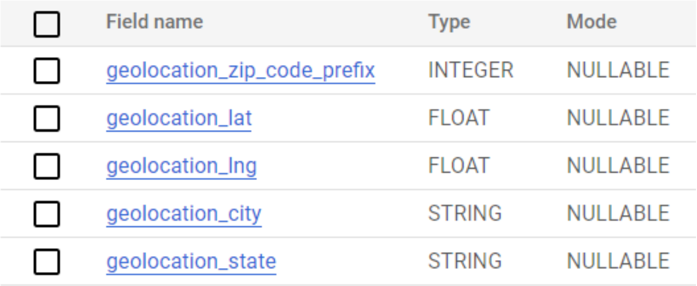
/* (1.1) Data type of columns in a table
*/

Customers table



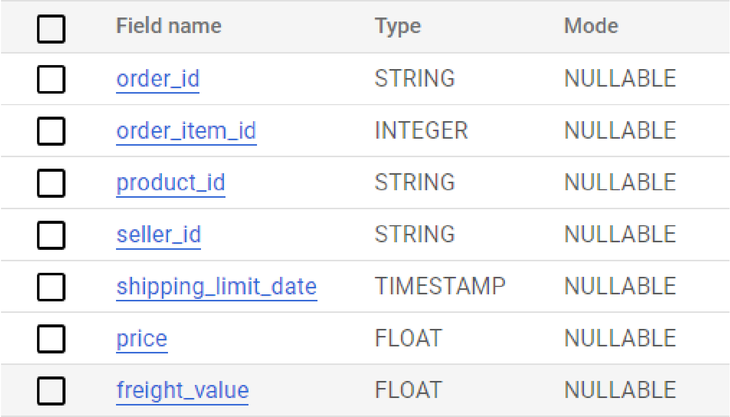
□	Field name	Type	Mode
□	customer_id	STRING	NULLABLE
□	customer_unique_id	STRING	NULLABLE
□	customer_zip_code_prefix	INTEGER	NULLABLE
□	customer_city	STRING	NULLABLE
□	customer_state	STRING	NULLABLE

geolocation table



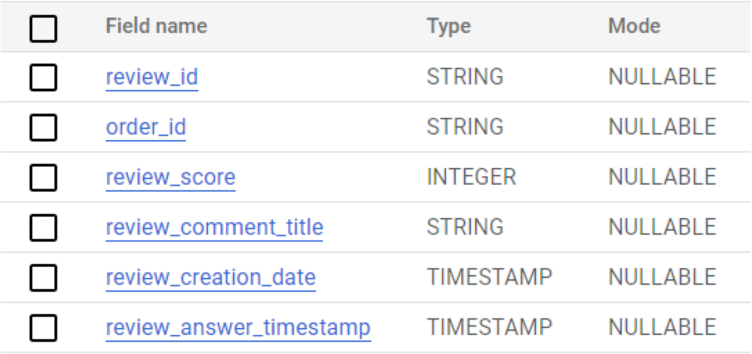
□	Field name	Type	Mode
□	geolocation_zip_code_prefix	INTEGER	NULLABLE
□	geolocation_lat	FLOAT	NULLABLE
□	geolocation_lng	FLOAT	NULLABLE
□	geolocation_city	STRING	NULLABLE
□	geolocation_state	STRING	NULLABLE

Order_items table



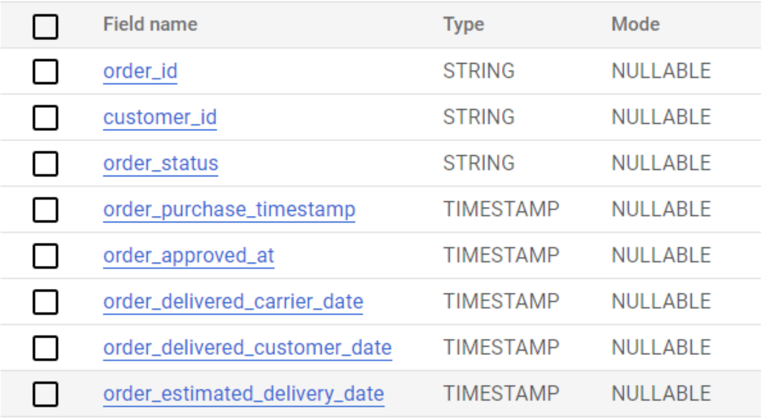
□	Field name	Type	Mode
□	order_id	STRING	NULLABLE
□	order_item_id	INTEGER	NULLABLE
□	product_id	STRING	NULLABLE
□	seller_id	STRING	NULLABLE
□	shipping_limit_date	TIMESTAMP	NULLABLE
□	price	FLOAT	NULLABLE
□	freight_value	FLOAT	NULLABLE

Order_reviews table



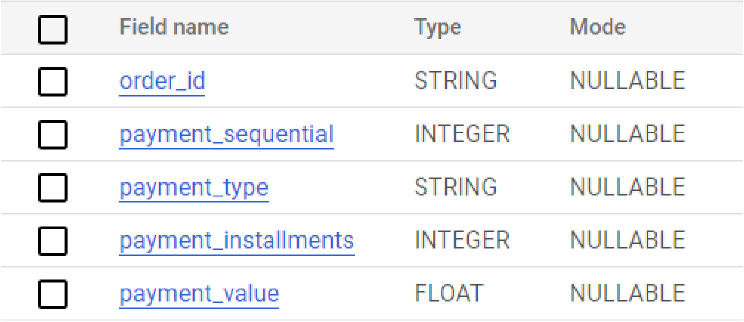
□	Field name	Type	Mode
□	review_id	STRING	NULLABLE
□	order_id	STRING	NULLABLE
□	review_score	INTEGER	NULLABLE
□	review_comment_title	STRING	NULLABLE
□	review_creation_date	TIMESTAMP	NULLABLE
□	review_answer_timestamp	TIMESTAMP	NULLABLE

Orders table:



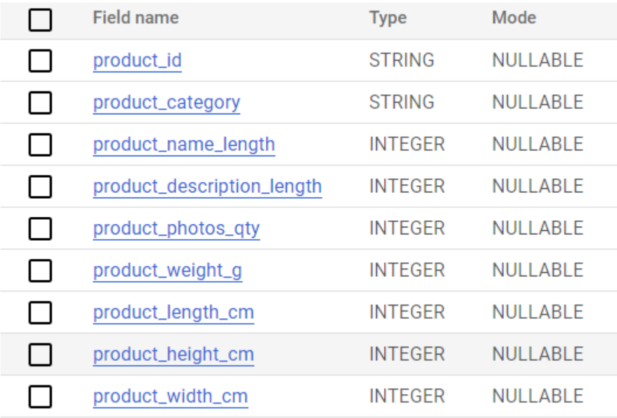
□	Field name	Type	Mode
□	order_id	STRING	NULLABLE
□	customer_id	STRING	NULLABLE
□	order_status	STRING	NULLABLE
□	order_purchase_timestamp	TIMESTAMP	NULLABLE
□	order_approved_at	TIMESTAMP	NULLABLE
□	order_delivered_carrier_date	TIMESTAMP	NULLABLE
□	order_delivered_customer_date	TIMESTAMP	NULLABLE
□	order_estimated_delivery_date	TIMESTAMP	NULLABLE

Payments table:



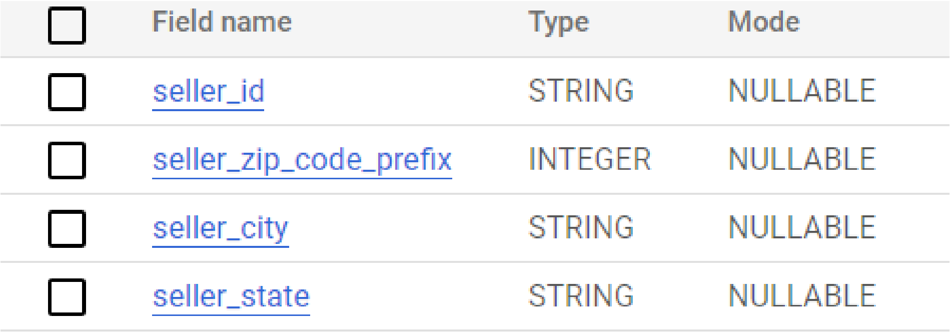
□	Field name	Type	Mode
□	order_id	STRING	NULLABLE
□	payment_sequential	INTEGER	NULLABLE
□	payment_type	STRING	NULLABLE
□	payment_installments	INTEGER	NULLABLE
□	payment_value	FLOAT	NULLABLE

Products table:

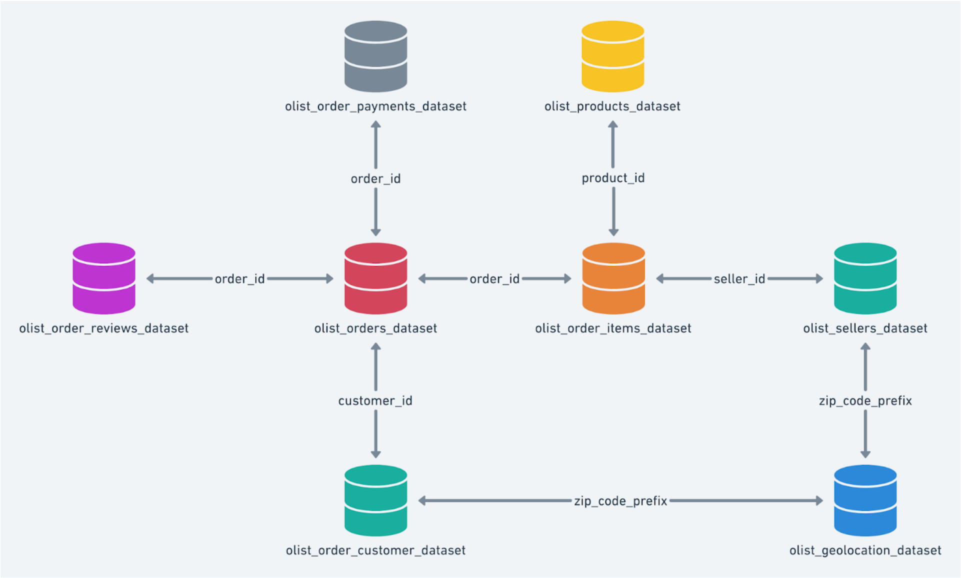


□	Field name	Type	Mode
□	product_id	STRING	NULLABLE
□	product_category	STRING	NULLABLE
□	product_name_length	INTEGER	NULLABLE
□	product_description_length	INTEGER	NULLABLE
□	product_photos_qty	INTEGER	NULLABLE
□	product_weight_g	INTEGER	NULLABLE
□	product_length_cm	INTEGER	NULLABLE
□	product_height_cm	INTEGER	NULLABLE
□	product_width_cm	INTEGER	NULLABLE

Sellers table:



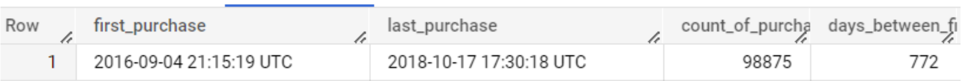
□	Field name	Type	Mode
□	seller_id	STRING	NULLABLE
□	seller_zip_code_prefix	INTEGER	NULLABLE
□	seller_city	STRING	NULLABLE
□	seller_state	STRING	NULLABLE



This is the relationship between all the tables which helps in joining the table if needed.

---(1.2)time period of the order purchases for the dataset alog with total orders between the period and total time period

```
SELECT
 MIN(order_purchase_timestamp) AS first_purchase,
 MAX(order_purchase_timestamp) AS last_purchase,
 COUNT(DISTINCT (order_purchase_timestamp)) AS count_of_purchase_dates,
 DATE_DIFF(MAX(order_purchase_timestamp), MIN(order_purchase_timestamp),DAY) AS
days_between_first_last_purchase
 FROM `business_data.orders`;
```

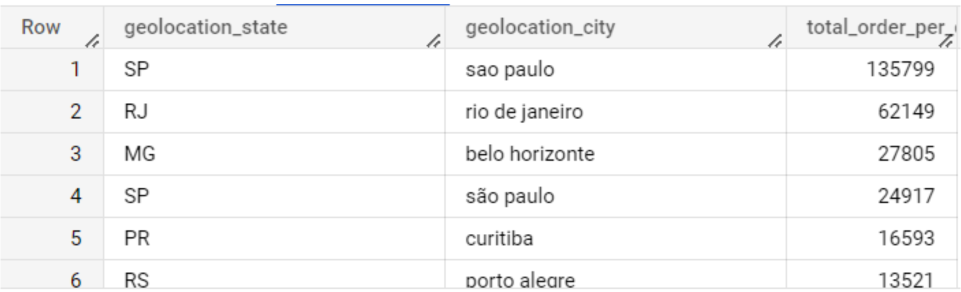


Row	first_purchase	last_purchase	count_of_purcha	days_between_fi
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC	98875	772

```
result=The orders happened between 2016-09-04 to 2018-10-17 for our data set
```

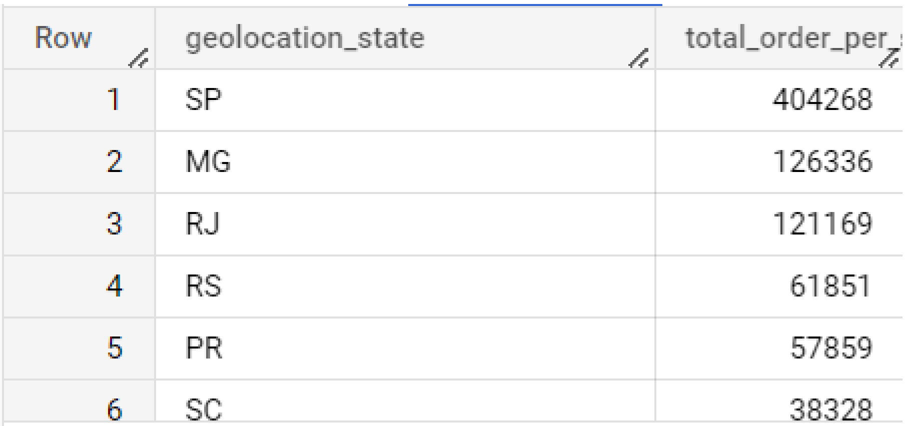
---(1.3)Cities and States of customers ordered during the given period

```
select geolocation_state,geolocation_city,count(geolocation_city) as
total_order_per_city
from `business_data.geolocation`
group by geolocation_state,geolocation_city
order by total_order_per_city desc;
```



Row	geolocation_state	geolocation_city	total_order_per_
1	SP	sao paulo	135799
2	RJ	rio de janeiro	62149
3	MG	belo horizonte	27805
4	SP	são paulo	24917
5	PR	curitiba	16593
6	RS	porto alegre	13521

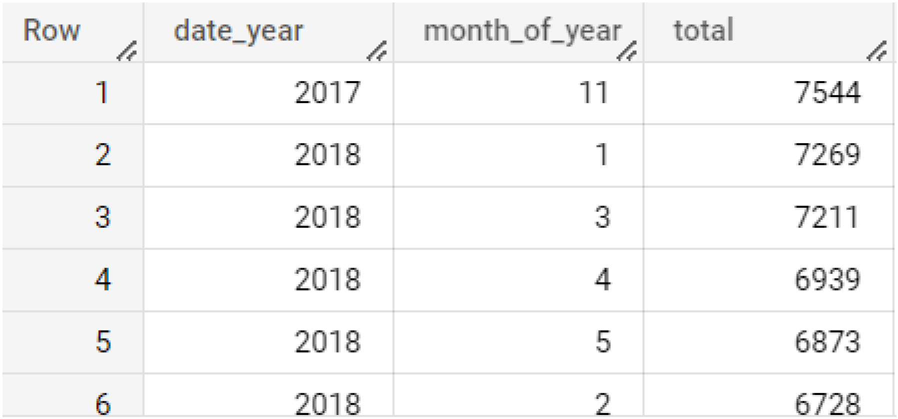
```
select geolocation_state,count(geolocation_state) as total_order_per_state
from `business_data.geolocation`
group by geolocation_state
order by total_order_per_state desc;
```



Row	geolocation_state	total_order_per_
1	SP	404268
2	MG	126336
3	RJ	121169
4	RS	61851
5	PR	57859
6	SC	38328

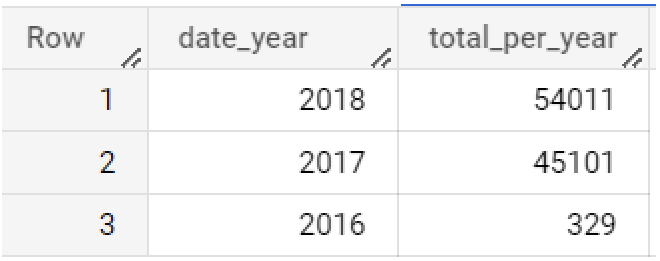
```
---result=Orders are placed more on the south region i.e. countries near the south
Atlantic region while least on north region of country which is land locked
```

```
---(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?
select
EXTRACT(YEAR FROM order_purchase_timestamp) AS date_year,
EXTRACT(MONTH FROM order_purchase_timestamp) AS month_of_year,
count(*) as total
from `business_data.orders`
group by date_year,month_of_year
order by total desc;
```



Row	date_year	month_of_year	total
1	2017	11	7544
2	2018	1	7269
3	2018	3	7211
4	2018	4	6939
5	2018	5	6873
6	2018	2	6728

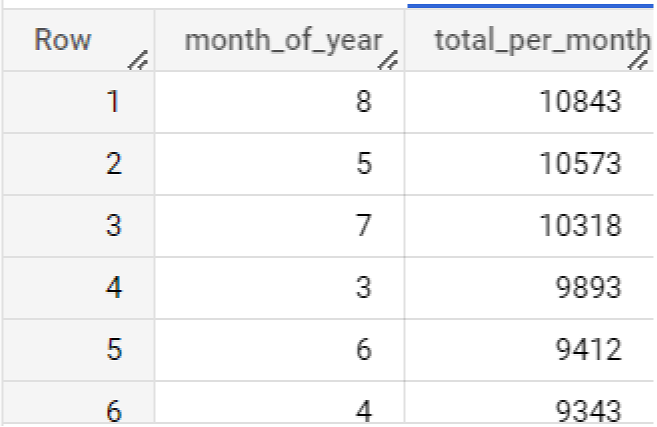
```
select
EXTRACT(YEAR FROM order_purchase_timestamp) AS date_year,
count(*) as total_per_year
from `business_data.orders`
group by date_year
order by total_per_year desc;
```



Row	date_year	total_per_year
1	2018	54011
2	2017	45101
3	2016	329

```
---result= the business grow at a very high rate in 2016 and it keeps growing at good
rate till the end of our data
```

```
select
EXTRACT(MONTH FROM order_purchase_timestamp) AS month_of_year,
count(*) as total_per_month
from `business_data.orders`
group by month_of_year
order by total_per_month desc;
```

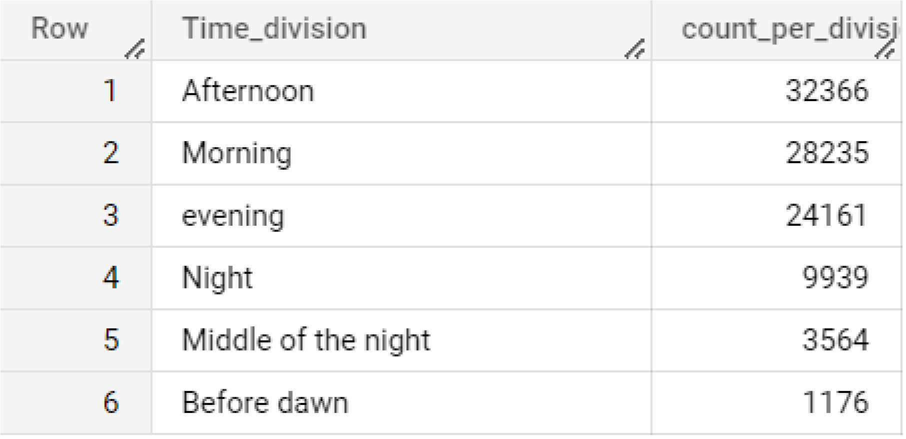


Row	month_of_year	total_per_month
1	8	10843
2	5	10573
3	7	10318
4	3	9893
5	6	9412
6	4	9343

---result=people prefer buying around mid of the year (having highest orders for the month of august) and they tend to buy less at the end of the month

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

```
select temp.Time_division,count(*) as count_per_division
from (
  select
  EXTRACT(HOUR FROM order_purchase_timestamp) AS hour_of_day,
  case
       when EXTRACT(HOUR FROM order_purchase_timestamp)>=2 and EXTRACT(HOUR FROM
order_purchase_timestamp)<=5 then 'Before dawn'
       when EXTRACT(HOUR FROM order_purchase_timestamp)>=6 and EXTRACT(HOUR FROM
order_purchase_timestamp)<=12 then 'Morning'
       when EXTRACT(HOUR FROM order_purchase_timestamp)>=13 and EXTRACT(HOUR FROM
order_purchase_timestamp)<=17 then 'Afternoon'
       when EXTRACT(HOUR FROM order_purchase_timestamp)>=18 and EXTRACT(HOUR FROM
order_purchase_timestamp)<=21 then 'evening'
       when EXTRACT(HOUR FROM order_purchase_timestamp)>=22 and EXTRACT(HOUR FROM
order_purchase_timestamp)<=23 then 'Night'
       else 'Middle of the night'
  END as Time_division
  from `business_data.orders`) as temp
  group by temp.Time_division
  order by count_per_division desc
```

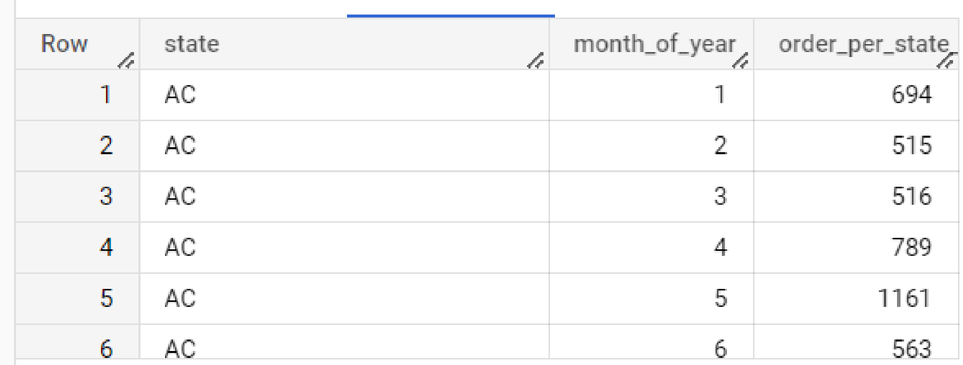


Row	Time_division	count_per_divisi
1	Afternoon	32366
2	Morning	28235
3	evening	24161
4	Night	9939
5	Middle of the night	3564
6	Before dawn	1176

---result= People are more likely to order between morning and evening. And the peak time of order is afternoon i.e between 12 pm to 5 pm.

---(3.1) Get month on month orders by states

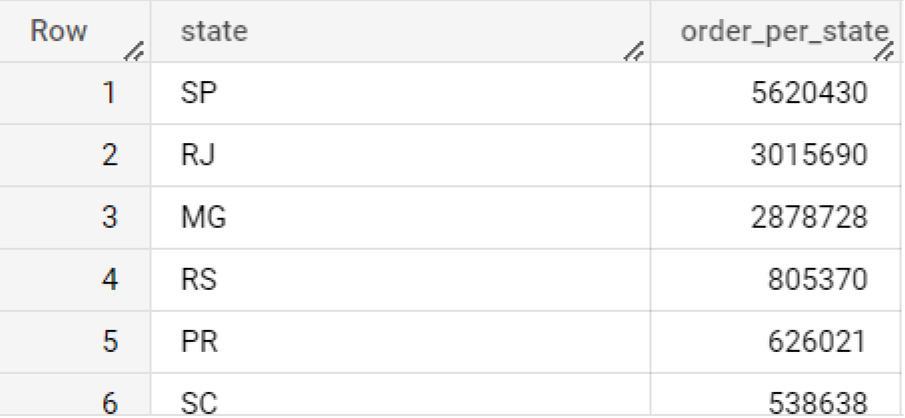
```
select g.geolocation_state as state,
EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month_of_year,
count(*) as order_per_state_permonth
from `business_data.orders` as o
join `business_data.customers`as c
on o.customer_id=c.customer_id
join `business_data.geolocation` as g
on c.customer_zip_code_prefix=g.geolocation_zip_code_prefix
group by month_of_year,state
order by state,month_of_year
```



Row	state	month_of_year	order_per_state
1	AC	1	694
2	AC	2	515
3	AC	3	516
4	AC	4	789
5	AC	5	1161
6	AC	6	563

---(3.2)Distribution of customers across the states in Brazil

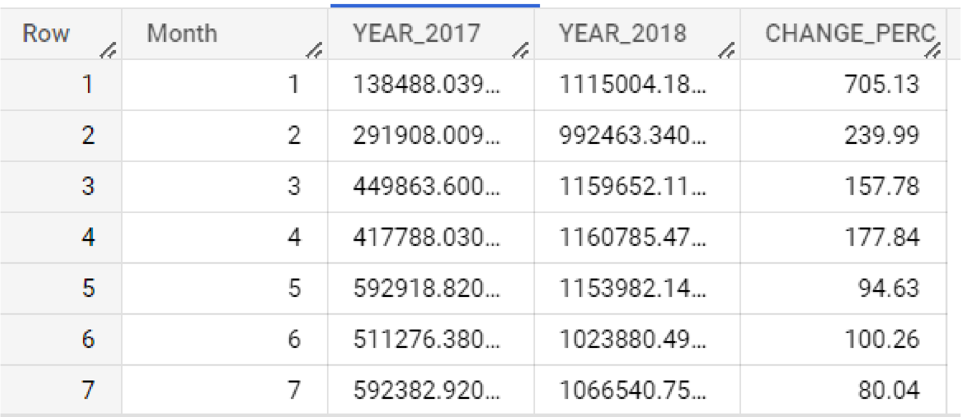
```
select g.geolocation_state as state,
count(*) as order_per_state
from `business_data.orders` as o
join `business_data.customers`as c
on o.customer_id=c.customer_id
join `business_data.geolocation` as g
on c.customer_zip_code_prefix=g.geolocation_zip_code_prefix
group by state
order by order_per_state desc
```



Row	state	order_per_state
1	SP	5620430
2	RJ	3015690
3	MG	2878728
4	RS	805370
5	PR	626021
6	SC	538638

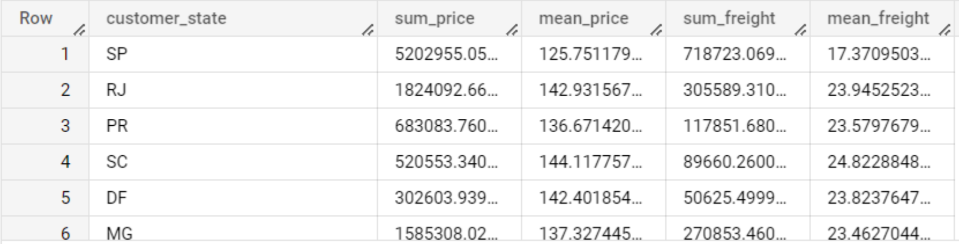
---result= Bulk of the order came from the southeastern border, the one with the open sea border

```
---(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

WITH CTE_1 AS(
  select EXTRACT(YEAR FROM o.order_purchase_timestamp) as Year,
  EXTRACT(MONTH FROM o.order_purchase_timestamp) as Month,
  sum(p.payment_value) as total_ordercost,
  from `business_data.payments` p
  join `business_data.orders` o
  on o.order_id=p.order_id
  where EXTRACT(MONTH FROM o.order_purchase_timestamp)<=8
  group by Year,Month
  order by Year,Month
)
SELECT Month,
YEAR_2017,
YEAR_2018,
ROUND((YEAR_2018-YEAR_2017)/YEAR_2017*100,2) AS CHANGE_PERC
FROM (
  SELECT Month,
  SUM(CASE WHEN Year=2017 THEN total_ordercost ELSE 0 END )AS YEAR_2017,
  SUM(CASE WHEN Year=2018 THEN total_ordercost ELSE 0 END )AS YEAR_2018
  FROM CTE_1
  WHERE (Year=2017 or year=2018) and Month<=8
  group by Month
  order by Month
)
```



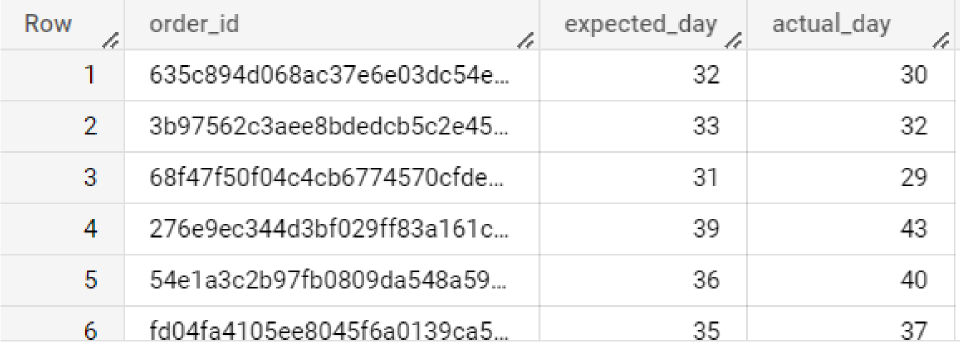
Row	Month	YEAR_2017	YEAR_2018	CHANGE_PERC
1	1	138488.039...	1115004.18...	705.13
2	2	291908.009...	992463.340...	239.99
3	3	449863.600...	1159652.11...	157.78
4	4	417788.030...	1160785.47...	177.84
5	5	592918.820...	1153982.14...	94.63
6	6	511276.380...	1023880.49...	100.26
7	7	592382.920...	1066540.75...	80.04

```
---RESULT= there is a tremendous increase for the first month and for the rest of
month its a positive change.


---(4.2) Mean & Sum of price and freight value by customer state
select c.customer_state,sum(ot.price) as
sum_price,sum(ot.price)/count(distinct(o.order_id)) as mean_price,
sum(ot.freight_value) as sum_freight,sum(ot.freight_value)/count(distinct(o.order_id))
as mean_freight
from `business_data.order_items` ot
join `business_data.orders` o
on o.order_id=ot.order_id
join `business_data.customers` c
on c.customer_id=o.customer_id
group by c.customer_state
```



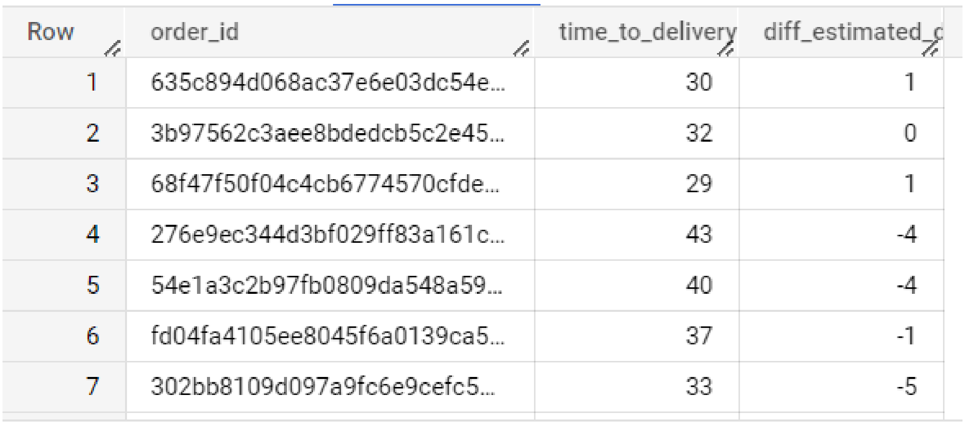
Row	customer_state	sum_price	mean_price	sum_freight	mean_freight
1	SP	5202955.05...	125.751179...	718723.069...	17.3709503...
2	RJ	1824092.66...	142.931567...	305589.310...	23.9452523...
3	PR	683083.760...	136.671420...	117851.680...	23.5797679...
4	SC	520553.340...	144.117757...	89660.2600...	24.8228848...
5	DF	302603.939...	142.401854...	50625.4999...	23.8237647...
6	MG	1585308.02...	137.327445...	270853.460...	23.4627044...

```
---(5.1)Calculate days between purchasing, delivering and estimated delivery


select order_id,
date_diff(order_estimated_delivery_date,order_purchase_timestamp,DAY) as expected_day,
date_diff(order_delivered_customer_date,order_purchase_timestamp,DAY) as actual_day
from `business_data.orders`
where order_status='delivered';
```



Row	order_id	expected_day	actual_day
1	635c894d068ac37e6e03dc54e...	32	30
2	3b97562c3aee8bdedcb5c2e45...	33	32
3	68f47f50f04c4cb6774570cfde...	31	29
4	276e9ec344d3bf029ff83a161c...	39	43
5	54e1a3c2b97fb0809da548a59...	36	40
6	fd04fa4105ee8045f6a0139ca5...	35	37

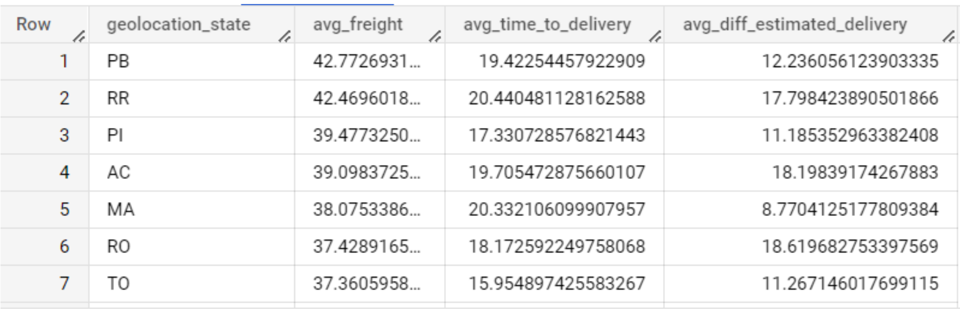
```
/* (5.2) Find time_to_delivery & diff_estimated_delivery. Formula for the same given
below:
time_to_delivery = order_purchase_timestamp-order_delivered_customer_date
diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date
*/
select order_id,
date_diff(order_delivered_customer_date,order_purchase_timestamp,DAY) as
time_to_delivery,
date_diff(order_estimated_delivery_date, order_delivered_customer_date,DAY) as
diff_estimated_delivery
from `business_data.orders`
where order_status='delivered';
```



Row	order_id	time_to_delivery	diff_estimated_c
1	635c894d068ac37e6e03dc54e...	30	1
2	3b97562c3aee8bdedcb5c2e45...	32	0
3	68f47f50f04c4cb6774570cfde...	29	1
4	276e9ec344d3bf029ff83a161c...	43	-4
5	54e1a3c2b97fb0809da548a59...	40	-4
6	fd04fa4105ee8045f6a0139ca5...	37	-1
7	302bb8109d097a9fc6e9cefc5...	33	-5

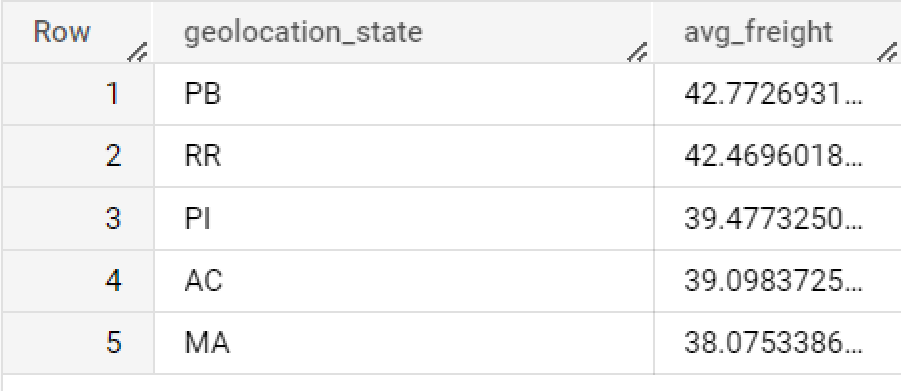
Result= the negative diff_estimated_delivery indicates the late delivery while postive shows that it reaches earlier than expected.

```
/* (5.3) Group data by state, take mean of freight_value, time_to_delivery,
diff_estimated_delivery */

select g.geolocation_state,
sum(ot.freight_value)/count(o.order_id) as avg_freight,
sum(date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp,DAY))/count(o
.order_id) as avg_time_to_delivery,
sum(date_diff(o.order_estimated_delivery_date,o.order_delivered_customer_date,DAY))/co
unt(o.order_id) as avg_diff_estimated_delivery
from `business_data.orders` o
join `business_data.customers` c
on o.customer_id=c.customer_id
join `business_data.order_items` ot
on o.order_id=ot.order_id
join `business_data.geolocation` g
on c.customer_zip_code_prefix=g.geolocation_zip_code_prefix
group by g.geolocation_state
order by avg_freight desc
```



Row	geolocation_state	avg_freight	avg_time_to_delivery	avg_diff_estimated_delivery
1	PB	42.7726931...	19.42254457922909	12.236056123903335
2	RR	42.4696018...	20.440481128162588	17.798423890501866
3	PI	39.4773250...	17.330728576821443	11.185352963382408
4	AC	39.0983725...	19.705472875660107	18.19839174267883
5	MA	38.0753386...	20.332106099907957	8.7704125177809384
6	RO	37.4289165...	18.172592249758068	18.619682753397569
7	TO	37.3605958...	15.954897425583267	11.267146017699115

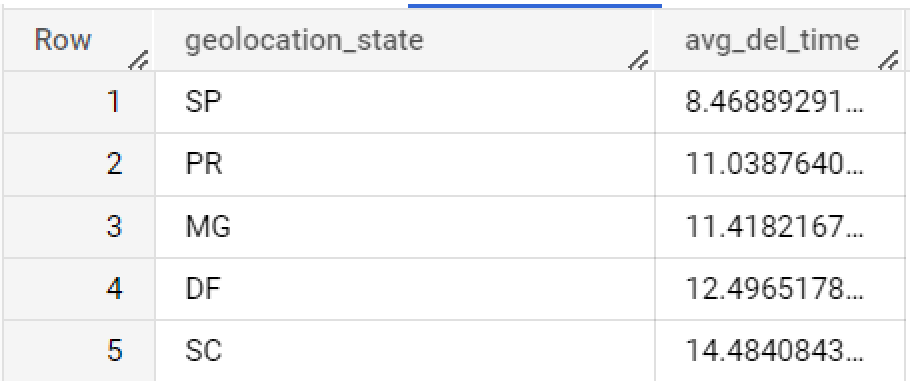
```
/* ( 5.4 and 5.5)Top 5 states with highest/lowest average freight value - sort in
desc/asc limit 5
*/
---top 5 states with highest average freight value


select g.geolocation_state,
sum(ot.freight_value)/count(o.order_id) as avg_freight
from `business_data.orders` o
join `business_data.customers` c
on o.customer_id=c.customer_id
join `business_data.order_items` ot
on o.order_id=ot.order_id
join `business_data.geolocation` g
on c.customer_zip_code_prefix=g.geolocation_zip_code_prefix
group by g.geolocation_state
order by avg_freight desc
limit 5
```



Row	geolocation_state	avg_freight
1	PB	42.7726931...
2	RR	42.4696018...
3	PI	39.4773250...
4	AC	39.0983725...
5	MA	38.0753386...

```
/*(5.6)Top 5 states with highest/lowest average time to delivery
*/
---top 5 dtsted which hss least average time of delivery
```

```
select g.geolocation_state,
sum(date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp,DAY))/count(o
.order_id) as avg_del_time
from `business_data.orders` o
join `business_data.customers` as c
on o.customer_id=c.customer_id
join `business_data.geolocation` as g
on c.customer_zip_code_prefix=g.geolocation_zip_code_prefix
where o.order_status='delivered'
group by g.geolocation_state
order by avg_del_time
limit 5;
```

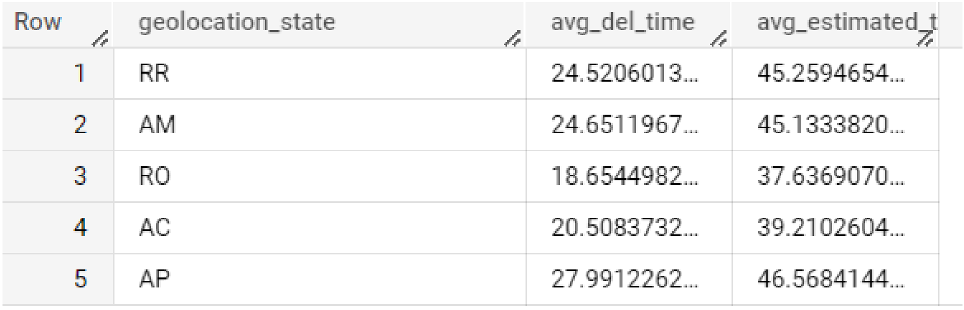


Row	geolocation_state	avg_del_time
1	SP	8.46889291...
2	PR	11.0387640...
3	MG	11.4182167...
4	DF	12.4965178...
5	SC	14.4840843...

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

```
select g.geolocation_state,
sum(date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp,DAY))/count(o
.order_id) as avg_del_time,
sum(date_diff(o.order_estimated_delivery_date,o.order_purchase_timestamp,DAY))/count(o
.order_id) as avg_estimated_time
from `business_data.orders` o
join `business_data.customers` as c
on o.customer_id=c.customer_id
join `business_data.geolocation` as g
on c.customer_zip_code_prefix=g.geolocation_zip_code_prefix
where o.order_status='delivered'
group by g.geolocation_state
order by (avg_estimated_time-avg_del_time) desc
```

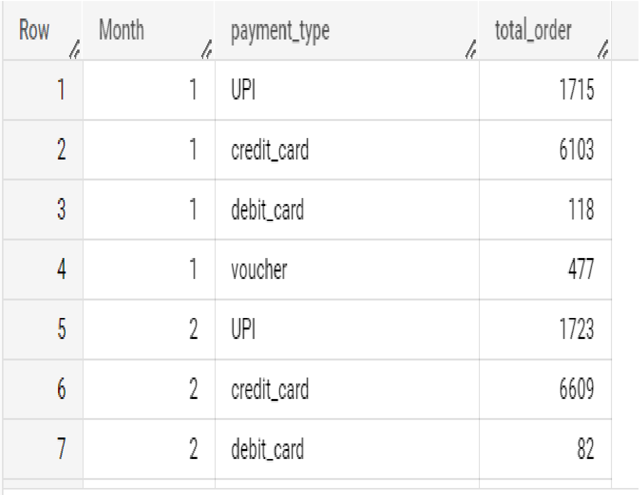
```
limit 5;
```



Row	geolocation_state	avg_del_time	avg_estimated_t
1	RR	24.5206013...	45.2594654...
2	AM	24.6511967...	45.1333820...
3	RO	18.6544982...	37.6369070...
4	AC	20.5083732...	39.2102604...
5	AP	27.9912262...	46.5684144...

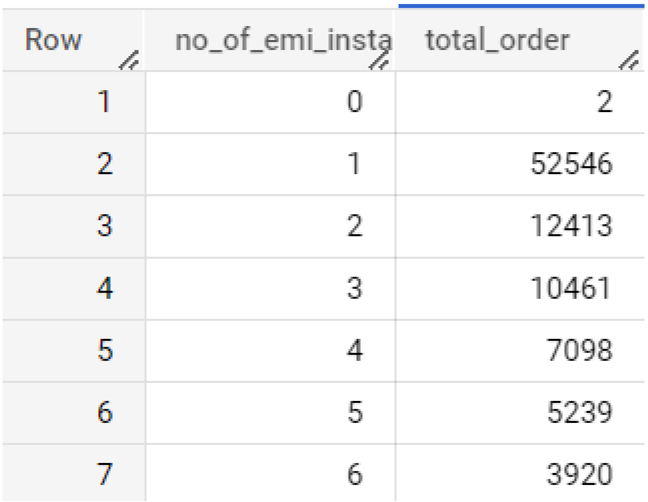
Result= here are the top states where delivery is very fast as the it always reaches a lot faster than the expected time of delivery.

```
----------------------------------------------------------
/*(6.1)Month over Month count of orders for different payment types
*/


select EXTRACT(MONTH FROM o.order_purchase_timestamp) as Month,
p.payment_type,
count(o.order_id) as total_order
from `business_data.orders` as o
join `business_data.payments` as p
on o.order_id=p.order_id
GROUP BY Month,p.payment_type
order by Month,p.payment_type
```



Row	Month	payment_type	total_order
1	1	UPI	1715
2	1	credit_card	6103
3	1	debit_card	118
4	1	voucher	477
5	2	UPI	1723
6	2	credit_card	6609
7	2	debit_card	82

```
/*(6.2)Count of orders based on the no. of payment installments
*/
select
p.payment_installments as no_of_emi_installments,
count(o.order_id) as total_order
from `business_data.orders` as o
join `business_data.payments` as p
on o.order_id=p.order_id
GROUP BY no_of_emi_installments
order by no_of_emi_installments
```



Row	no_of_emi_insta	total_order
1	0	2
2	1	52546
3	2	12413
4	3	10461
5	4	7098
6	5	5239
7	6	3920

Recommendation :
The bulk of the order scome from the south eastern border , the norder with the sea border. While target needs to expand to other areas by reducing the time of delivery and having a warehouse there.