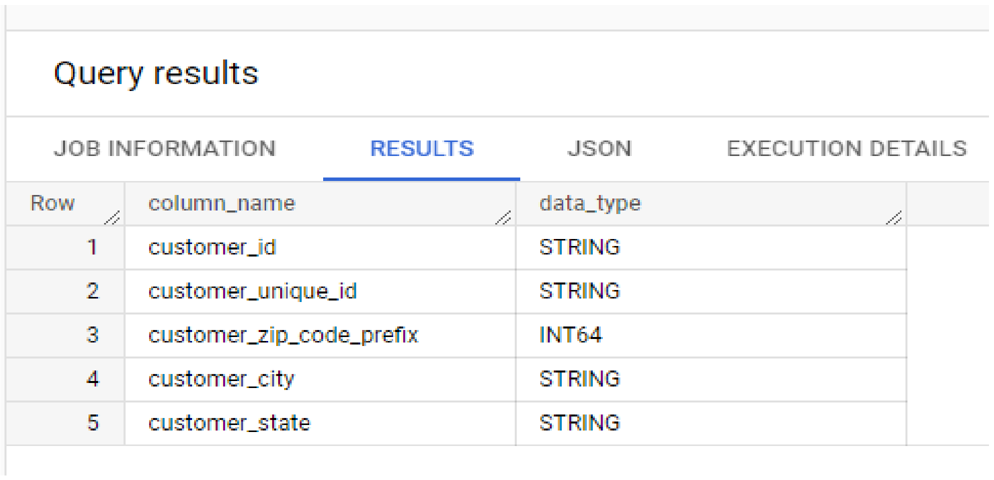
SQL Target Business Case Study

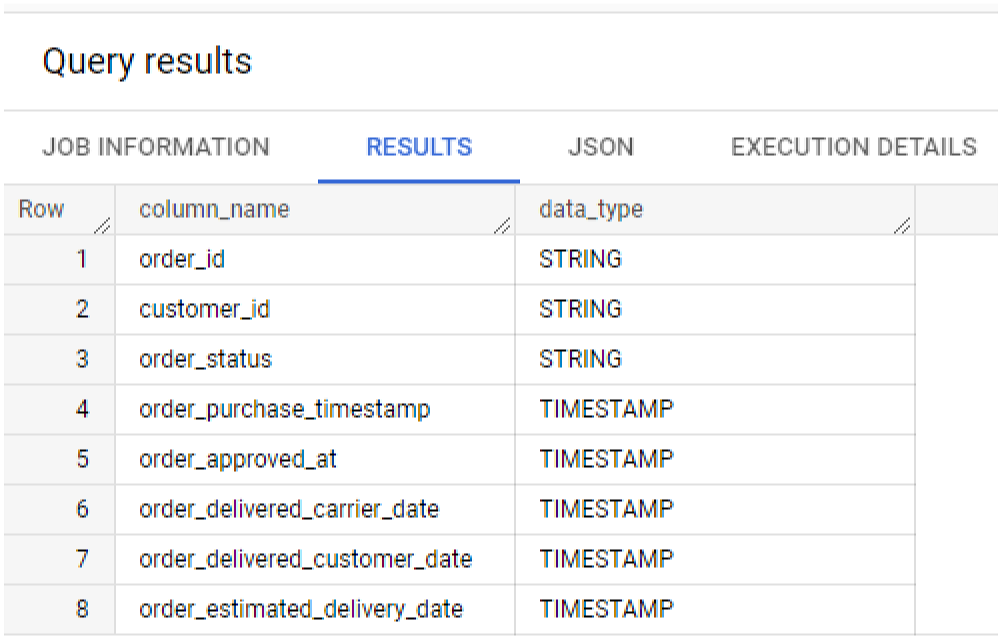
1. Import the dataset and do the usual exploratory analysis steps like checking the structure & characteristics of the dataset
    1. The data type of columns in a table

```
SELECT column_name, data_type

FROM target.INFORMATION_SCHEMA.COLUMNS

WHERE table_name = 'customers' ;
```


Query results

Row	column_name	data_type
1	customer_id	STRING
2	customer_unique_id	STRING
3	customer_zip_code_prefix	INT64
4	customer_city	STRING
5	customer_state	STRING

```
SELECT column_name, data_type

FROM target.INFORMATION_SCHEMA.COLUMNS

WHERE table_name = 'orders' ;
```

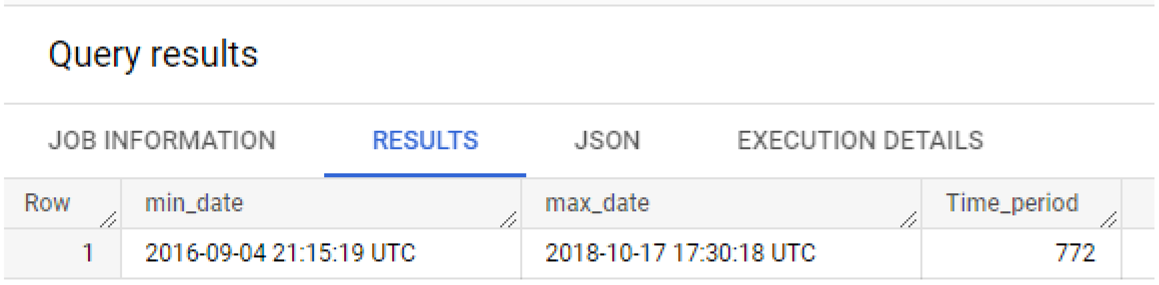

Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS

Row	column_name	data_type
1	order_id	STRING
2	customer_id	STRING
3	order_status	STRING
4	order_purchase_timestamp	TIMESTAMP
5	order_approved_at	TIMESTAMP
6	order_delivered_carrier_date	TIMESTAMP
7	order_delivered_customer_date	TIMESTAMP
8	order_estimated_delivery_date	TIMESTAMP

2. Time period for which the data is given

```
with cte as (

SELECT min(order_purchase_timestamp) as min_date ,

max(order_purchase_timestamp) as max_date

from target.orders )

SELECT

cte.min_date,

cte.max_date ,

DATE_DIFF(cte.max_date,cte.min_date,day) as Time_period   from cte
```

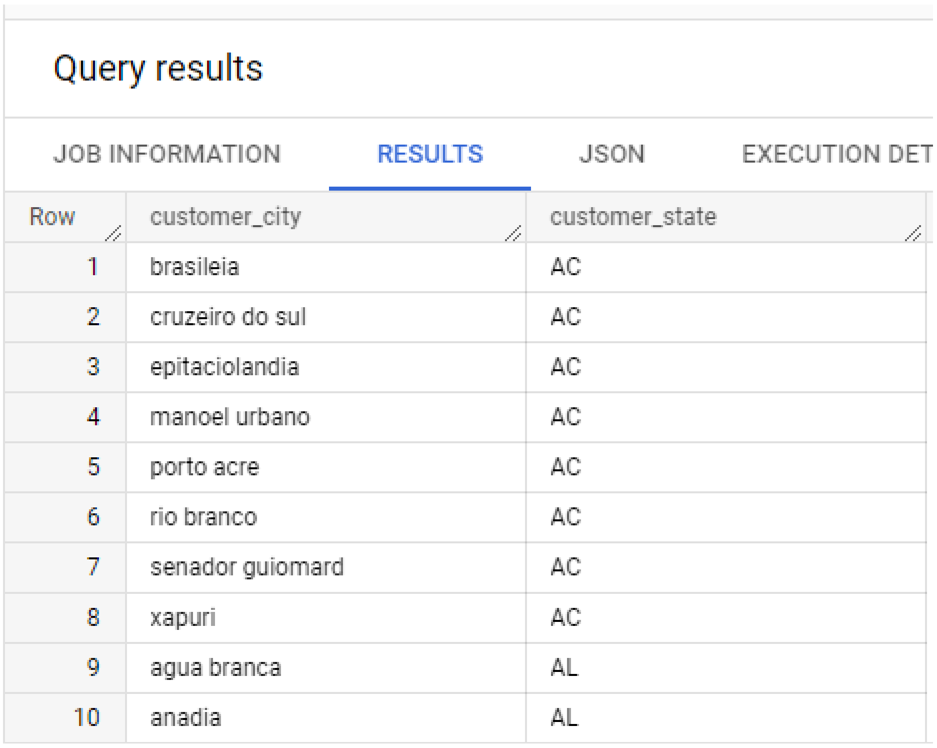

Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS

Row	min_date	max_date	Time_period
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC	772

3. Cities and States covered in the dataset

```
SELECT c.customer_city, c.customer_state

from target.customers as c

JOIN target.orders  as o

ON o.customer_id=c.customer_id

GROUP BY c.customer_state,c.customer_city

ORDER BY c.customer_state,customer_city
```

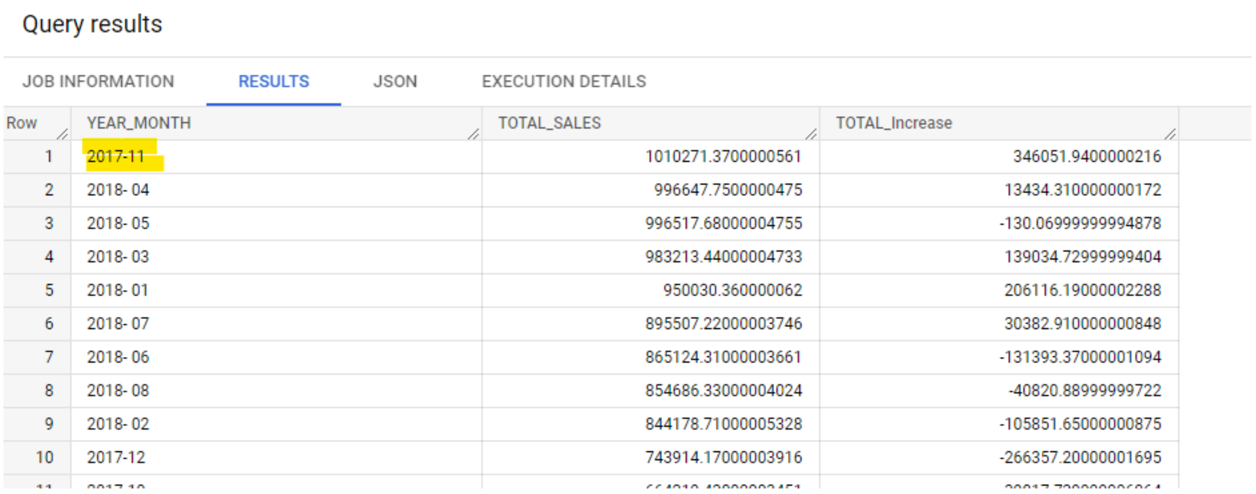


Row	customer_city	customer_state
1	brasileia	AC
2	cruzeiro do sul	AC
3	epitaciolandia	AC
4	manoel urbano	AC
5	porto acre	AC
6	rio branco	AC
7	senador guiomard	AC
8	xapuri	AC
9	agua branca	AL
10	anadia	AL

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

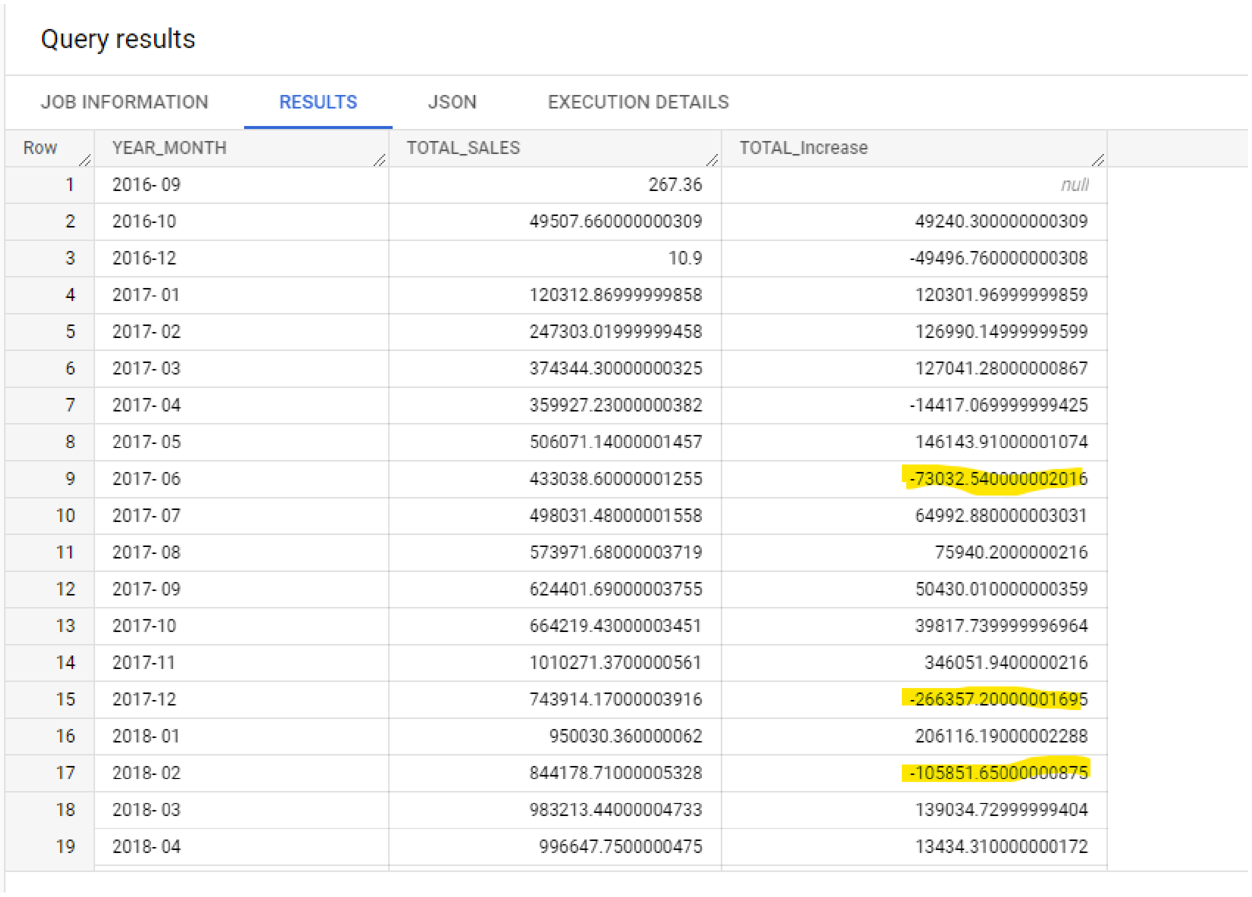
```
      with cte as (

  SELECT case when LENGTH( CAST(EXTRACT(MONTH FROM o.order_purchase_timestamp) as
STRING)) = 1 then

  CONCAT(EXTRACT(YEAR FROM o.order_purchase_timestamp),'- 0',EXTRACT(MONTH FROM
o.order_purchase_timestamp))

  else

  CONCAT(EXTRACT(YEAR FROM o.order_purchase_timestamp),'-',EXTRACT(MONTH FROM
o.order_purchase_timestamp))

  end as year_month_analysis,

  oi.price

  FROM target.orders as o

  JOIN target.order_items as oi

  ON o.order_id = oi.order_id

) ,

cte2 as (SELECT year_month_analysis AS YEAR_MONTH ,

SUM(cte.price) as TOTAL_SALES,

AVG(cte.price) as AVG_SALES,

LAG(SUM(cte.price),1) OVER(ORDER BY year_month_analysis) as diff

FROM cte

GROUP BY year_month_analysis

ORDER BY year_month_analysis)

SELECT YEAR_MONTH,TOTAL_SALES, TOTAL_SALES - diff  as TOTAL_Increase

 FROM cte2

ORDER BY cte2.TOTAL_SALES DESC
```


Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS

Row	YEAR_MONTH	TOTAL_SALES	TOTAL_Increase
1	2017-11	1010271.3700000561	346051.9400000216
2	2018- 04	996647.7500000475	13434.310000000172
3	2018- 05	996517.68000004755	-130.06999999994878
4	2018- 03	983213.44000004733	139034.72999999404
5	2018- 01	950030.360000062	206116.19000002288
6	2018- 07	895507.22000003746	30382.910000000848
7	2018- 06	865124.31000003661	-131393.37000001094
8	2018- 08	854686.33000004024	-40820.88999999722
9	2018- 02	844178.71000005328	-105851.65000000875
10	2017-12	743914.17000003916	-266357.20000001695

```
with cte as (

  SELECT case when LENGTH( CAST(EXTRACT(MONTH FROM o.order_purchase_timestamp) as
STRING)) = 1 then

  CONCAT(EXTRACT(YEAR FROM o.order_purchase_timestamp),'- 0',EXTRACT(MONTH FROM
o.order_purchase_timestamp))

  else

  CONCAT(EXTRACT(YEAR FROM o.order_purchase_timestamp),'-',EXTRACT(MONTH FROM
o.order_purchase_timestamp))

  end as year_month_analysis,

  oi.price

  FROM target.orders as o

  JOIN target.order_items as oi

  ON o.order_id = oi.order_id

) ,
cte2 as (SELECT year_month_analysis AS YEAR_MONTH ,

SUM(cte.price) as TOTAL_SALES,

AVG(cte.price) as AVG_SALES,

LAG(SUM(cte.price),1) OVER(ORDER BY year_month_analysis) as diff

FROM cte

GROUP BY year_month_analysis

ORDER BY year_month_analysis)

SELECT YEAR_MONTH,TOTAL_SALES, TOTAL_SALES - diff  as TOTAL_Increase

 FROM cte2
```



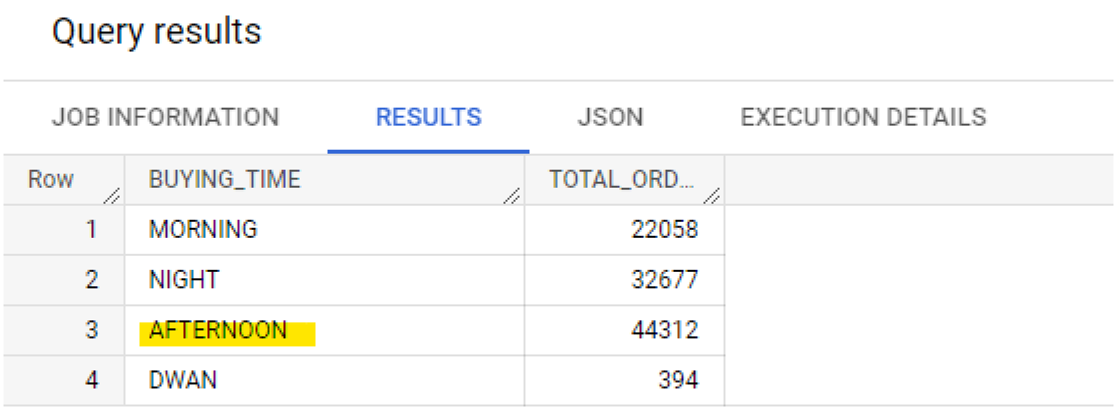
Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	YEAR_MONTH	TOTAL_SALES	TOTAL_Increase
1	2016-09	267.36	null
2	2016-10	49507.660000000309	49240.300000000309
3	2016-12	10.9	-49496.760000000308
4	2017-01	120312.86999999858	120301.96999999859
5	2017-02	247303.01999999458	126990.14999999599
6	2017-03	374344.30000000325	127041.28000000867
7	2017-04	359927.23000000382	-14417.069999999425
8	2017-05	506071.14000001457	146143.91000001074
9	2017-06	433038.60000001255	-73032.540000002016
10	2017-07	498031.48000001558	64992.880000003031
11	2017-08	573971.68000003719	75940.2000000216
12	2017-09	624401.69000003755	50430.010000000359
13	2017-10	664219.43000003451	39817.739999996964
14	2017-11	1010271.3700000561	346051.9400000216
15	2017-12	743914.17000003916	-266357.20000001695
16	2018-01	950030.360000062	206116.19000002288
17	2018-02	844178.71000005328	-105851.65000000875
18	2018-03	983213.44000004733	139034.72999999404
19	2018-04	996647.7500000475	13434.310000000172

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

```
SELECT case when CAST( EXTRACT( HOUR FROM order_purchase_timestamp) as INT64)
IN  (6,7,8,9,11,12) then 'MORNING'

when CAST( EXTRACT( HOUR FROM order_purchase_timestamp) as INT64) IN  (4,5,6)
then 'DWAN'

when CAST( EXTRACT( HOUR FROM order_purchase_timestamp) as INT64) IN
(19,20,21,22,23,0,1,2,3,4) then 'NIGHT'

ELSE 'AFTERNOON'

END AS BUYING_TIME,

COUNT(order_id) AS TOTAL_ORDERS

From target.orders

GROUP BY BUYING_TIME
```

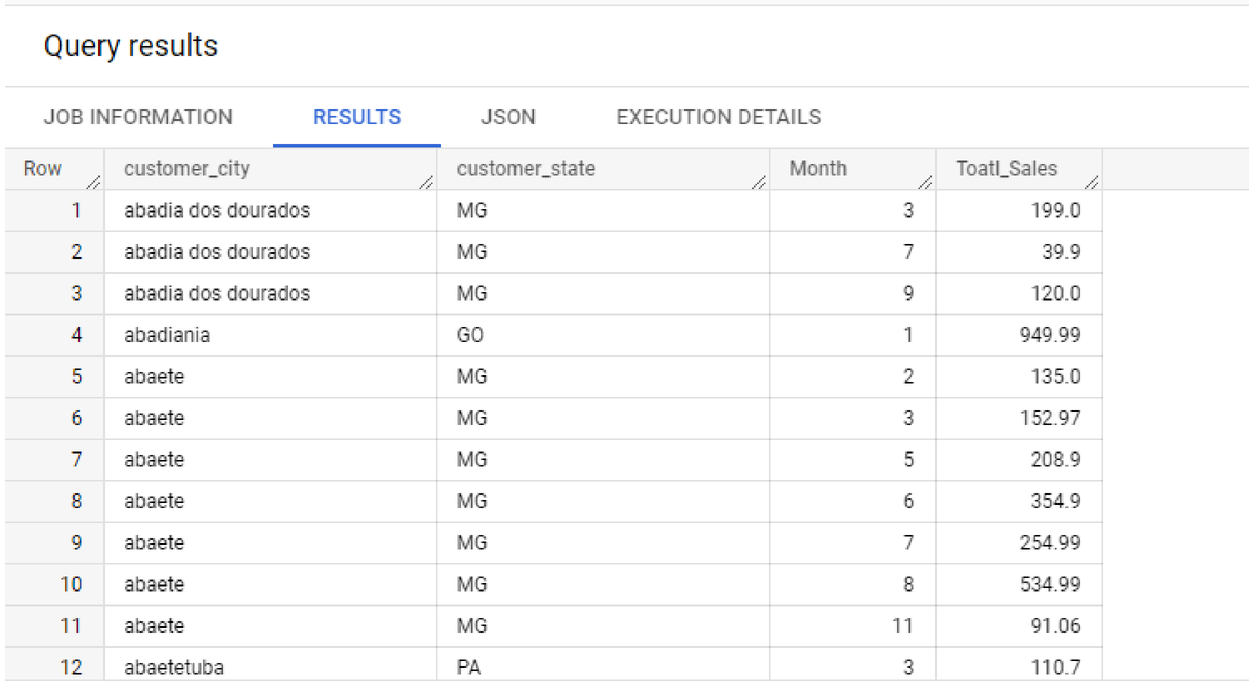


Row	BUYING_TIME	TOTAL_ORD...
1	MORNING	22058
2	NIGHT	32677
3	AFTERNOON	44312
4	DWAN	394

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

1. Get month-on-month orders by region, states

```
with cte as (

SELECT c.customer_city, c.customer_state,

EXTRACT(MONTH FROM o.order_purchase_timestamp) as Month,

oi.price

from target.customers as c

JOIN target.orders  as o

ON o.customer_id=c.customer_id

JOIN target.order_items oi

ON o.order_id = oi.order_id)

SELECT customer_city, customer_state,Month,ROUND(sum(price),2) as
Toatl_Sales

from cte

group by customer_city,customer_state,Month

order by customer_city, customer_state,Month
```



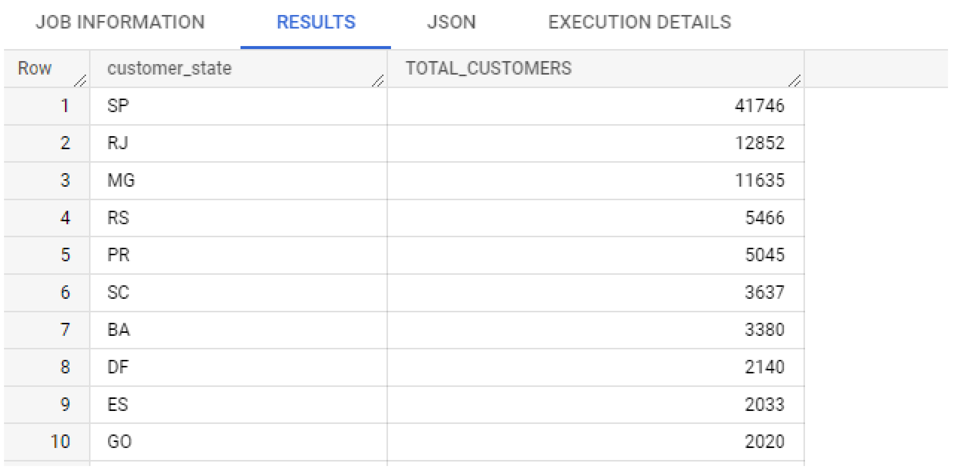
Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS

Row	customer_city	customer_state	Month	Toatl_Sales
1	abadia dos dourados	MG	3	199.0
2	abadia dos dourados	MG	7	39.9
3	abadia dos dourados	MG	9	120.0
4	abadiania	GO	1	949.99
5	abaete	MG	2	135.0
6	abaete	MG	3	152.97
7	abaete	MG	5	208.9
8	abaete	MG	6	354.9
9	abaete	MG	7	254.99
10	abaete	MG	8	534.99
11	abaete	MG	11	91.06
12	abaetetuba	PA	3	110.7

2. How are customers distributed in Brazil

```
SELECT c.customer_state,

COUNT(c.customer_id) as TOTAL_CUSTOMERS

from target.customers as c

JOIN target.orders  as o

ON o.customer_id=c.customer_id

GROUP BY c.customer_state

ORDER BY TOTAL_CUSTOMERS DESC
```



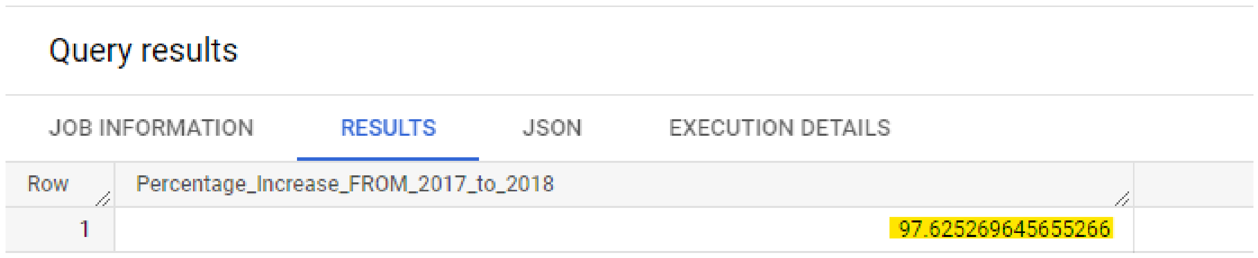
Row	customer_state	TOTAL_CUSTOMERS
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637
7	BA	3380
8	DF	2140
9	ES	2033
10	GO	2020

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

1. Get % increase in the cost of orders from 2017 to 2018 (include months between Jan to Aug only)

```
with cte as (

  SELECT case when CAST(EXTRACT(MONTH FROM o.order_purchase_timestamp) AS
INT64) IN (1,2,3,4,5,6,7,8,9) AND CAST(EXTRACT(YEAR FROM
o.order_purchase_timestamp) as INT64) =2017 THEN '2017'
```

```
  when CAST(EXTRACT(MONTH FROM o.order_purchase_timestamp) AS INT64) IN
(1,2,3,4,5,6,7,8,9) AND CAST(EXTRACT(YEAR FROM o.order_purchase_timestamp)as
INT64) = 2018 then '2018'

  END AS YEARS,

  o.order_id

  FROM target.orders as o

) ,

cte2 as (

  SELECT cte.YEARS, oi.price

  from cte

  JOIN target.order_items oi

  ON cte.order_id = oi.order_id

),

cte3 as (

SELECT cte2.YEARS, SUM(cte2.price)as TOTAL_SALES_IN_YEAR

,LAG(SUM(cte2.price),1) OVER(ORDER BY cte2.YEARS ) as prev_year_sale from cte2

where YEARS IS NOT NULL

GROUP BY YEARS )

SELECT ((cte3.TOTAL_SALES_IN_YEAR - prev_year_sale)/prev_year_sale) *100  As
Percentage_Increase_FROM_2017_to_2018 FROM cte3

where prev_year_sale IS NOT NULL
```

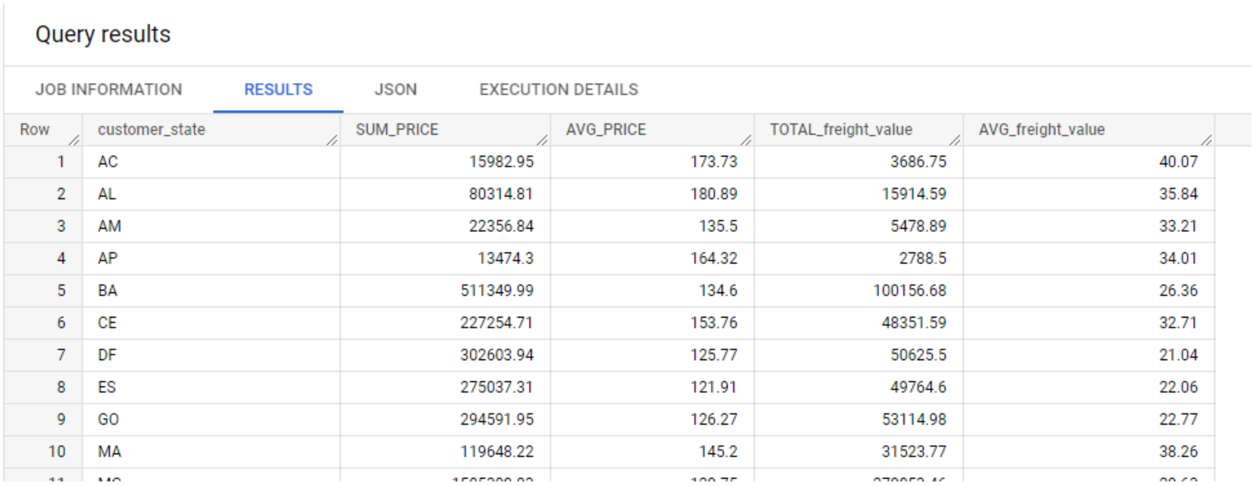

Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS

Row	Percentage_Increase_FROM_2017_to_2018
1	97.625269645655266

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

```
SELECT c.customer_state,ROUND(SUM(oi.price),2) SUM_PRICE,
ROUND(AVG(oi.price),2) AVG_PRICE,ROUND(SUM(oi.freight_value),2)
TOTAL_freight_value, ROUND(AVG(oi.freight_value),2) AVG_freight_value

from target.orders as o

JOIN target.order_items  as oi

ON o.order_id = oi.order_id

JOIN target.customers as c

ON  c.customer_id = o.customer_id

GROUP BY  c.customer_state

ORDER BY c.customer_state
```


Query results

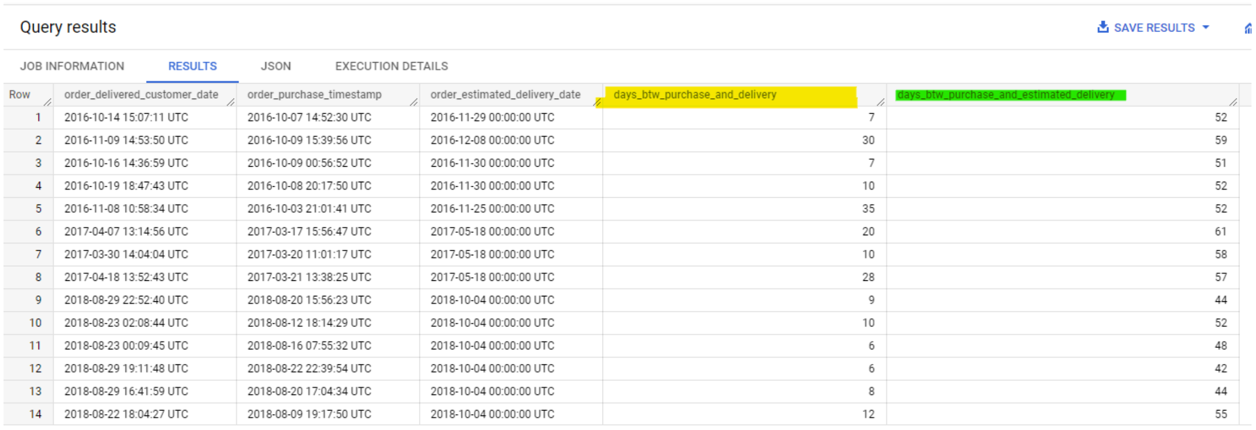
JOB INFORMATION RESULTS JSON EXECUTION DETAILS

Row	customer_state	SUM_PRICE	AVG_PRICE	TOTAL_freight_value	AVG_freight_value
1	AC	15982.95	173.73	3686.75	40.07
2	AL	80314.81	180.89	15914.59	35.84
3	AM	22356.84	135.5	5478.89	33.21
4	AP	13474.3	164.32	2788.5	34.01
5	BA	511349.99	134.6	100156.68	26.36
6	CE	227254.71	153.76	48351.59	32.71
7	DF	302603.94	125.77	50625.5	21.04
8	ES	275037.31	121.91	49764.6	22.06
9	GO	294591.95	126.27	53114.98	22.77
10	MA	119648.22	145.2	31523.77	38.26

5. Analysis on sales, freight and delivery time

1. Calculate days between purchasing, delivering and estimated delivery

```
SELECT
order_delivered_customer_date,order_purchase_timestamp,order_estimated_delivery
_date,

DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp, DAY) AS
days_btw_purchase_and_delivery,

DATE_DIFF(order_estimated_delivery_date,order_purchase_timestamp, DAY) AS
days_btw_purchase_and_estimated_delivery

FROM target.orders

where order_delivered_customer_date IS NOT NULL
```

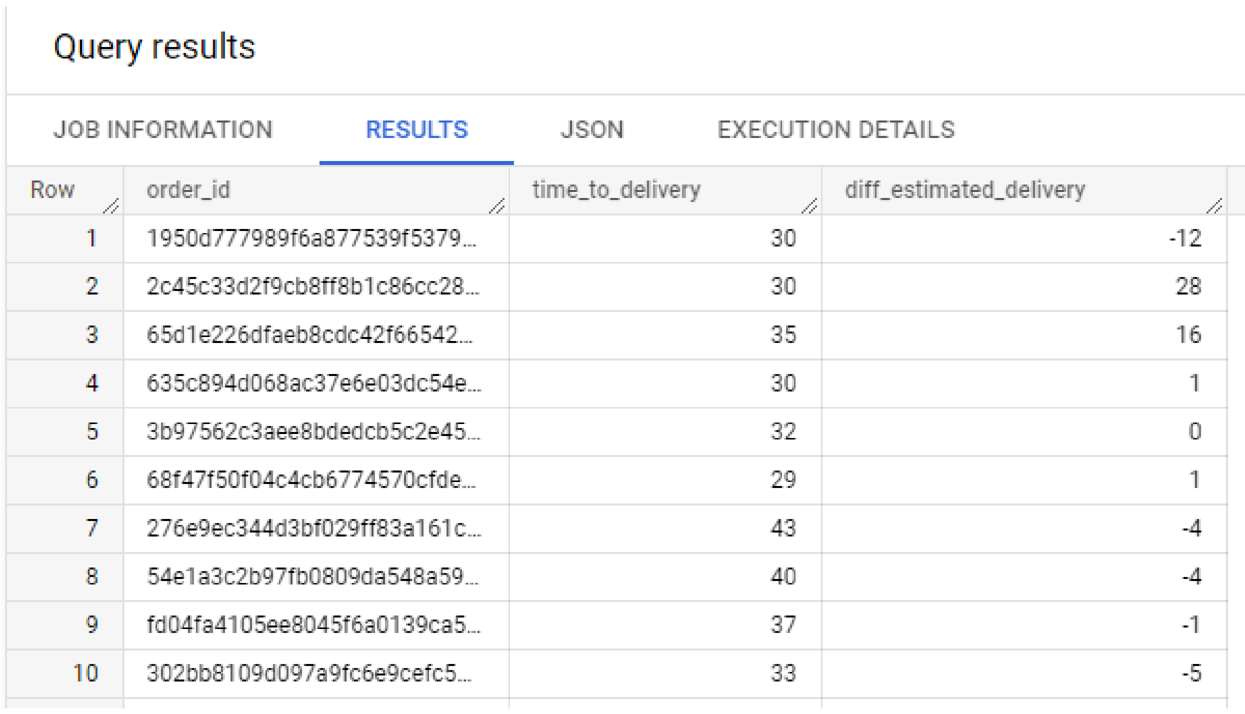


Query results

Row	order_delivered_customer_date	order_purchase_timestamp	order_estimated_delivery_date	days_btw_purchase_and_delivery	days_btw_purchase_and_estimated_delivery
1	2016-10-14 15:07:11 UTC	2016-10-07 14:52:30 UTC	2016-11-29 00:00:00 UTC	7	52
2	2016-11-09 14:53:50 UTC	2016-10-09 15:39:56 UTC	2016-12-08 00:00:00 UTC	30	59
3	2016-10-16 14:36:59 UTC	2016-10-09 00:56:52 UTC	2016-11-30 00:00:00 UTC	7	51
4	2016-10-19 18:47:43 UTC	2016-10-08 20:17:50 UTC	2016-11-30 00:00:00 UTC	10	52
5	2016-11-08 10:58:34 UTC	2016-10-03 21:01:41 UTC	2016-11-25 00:00:00 UTC	35	52
6	2017-04-07 13:14:56 UTC	2017-03-17 15:56:47 UTC	2017-05-18 00:00:00 UTC	20	61
7	2017-03-30 14:04:04 UTC	2017-03-20 11:01:17 UTC	2017-05-18 00:00:00 UTC	10	58
8	2017-04-18 13:52:43 UTC	2017-03-21 13:38:25 UTC	2017-05-18 00:00:00 UTC	28	57
9	2018-08-29 22:52:40 UTC	2018-08-20 15:56:23 UTC	2018-10-04 00:00:00 UTC	9	44
10	2018-08-23 02:08:44 UTC	2018-08-12 18:14:29 UTC	2018-10-04 00:00:00 UTC	10	52
11	2018-08-23 00:09:45 UTC	2018-08-16 07:55:32 UTC	2018-10-04 00:00:00 UTC	6	48
12	2018-08-29 19:11:48 UTC	2018-08-22 22:39:54 UTC	2018-10-04 00:00:00 UTC	6	42
13	2018-08-29 16:41:59 UTC	2018-08-20 17:04:34 UTC	2018-10-04 00:00:00 UTC	8	44
14	2018-08-22 18:04:27 UTC	2018-08-09 19:17:50 UTC	2018-10-04 00:00:00 UTC	12	55

2. Create columns:
    a. time_to_delivery = order_purchase_timestamp-order_delivered_customer_date
    b. 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 target.orders

where order_delivered_customer_date IS NOT NULL
```


Query results

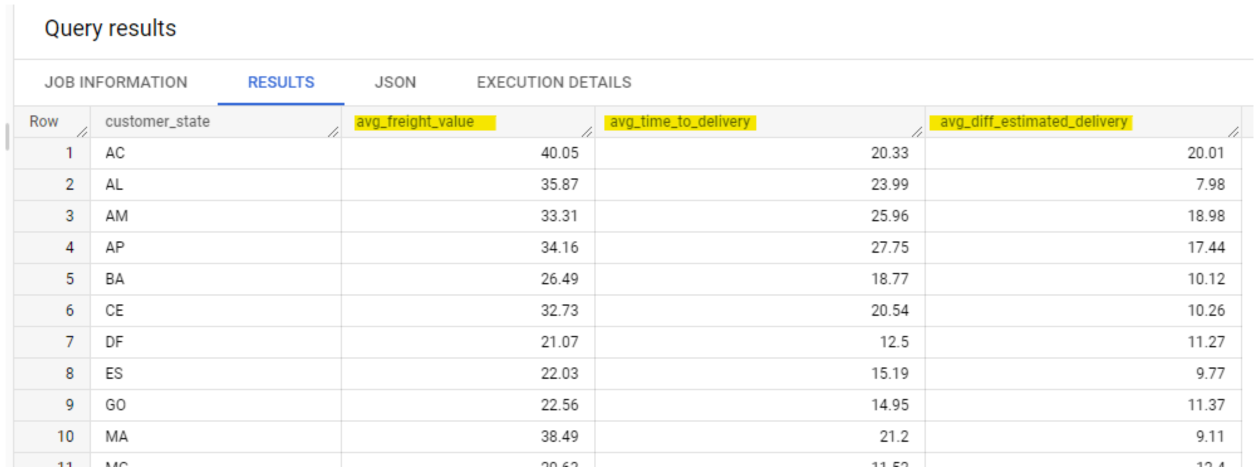
JOB INFORMATION RESULTS JSON EXECUTION DETAILS

Row	order_id	time_to_delivery	diff_estimated_delivery
1	1950d777989f6a877539f5379...	30	-12
2	2c45c33d2f9cb8ff8b1c86cc28...	30	28
3	65d1e226dfaeb8cdc42f66542...	35	16
4	635c894d068ac37e6e03dc54e...	30	1
5	3b97562c3aee8bdedcb5c2e45...	32	0
6	68f47f50f04c4cb6774570cfde...	29	1
7	276e9ec344d3bf029ff83a161c...	43	-4
8	54e1a3c2b97fb0809da548a59...	40	-4
9	fd04fa4105ee8045f6a0139ca5...	37	-1
10	302bb8109d097a9fc6e9cefc5...	33	-5

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

```
SELECT c.customer_state,ROUND(AVG(oi.freight_value),2) as avg_freight_value,

ROUND(AVG(DATE_DIFF(o.order_delivered_customer_date,o.order_purchase_timestamp,
DAY)),2) AS avg_time_to_delivery ,

ROUND(AVG(DATE_DIFF(o.order_estimated_delivery_date,o.order_delivered_customer_
date, DAY)),2) AS avg_diff_estimated_delivery

FROM target.orders o

JOIN target.order_items as oi

ON o.order_id=oi.order_id

JOIN target.customers as  c

ON o.customer_id = c.customer_id
```

```
where order_delivered_customer_date IS NOT NULL

GROUP BY c.customer_state

order by c.customer_state
```



Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

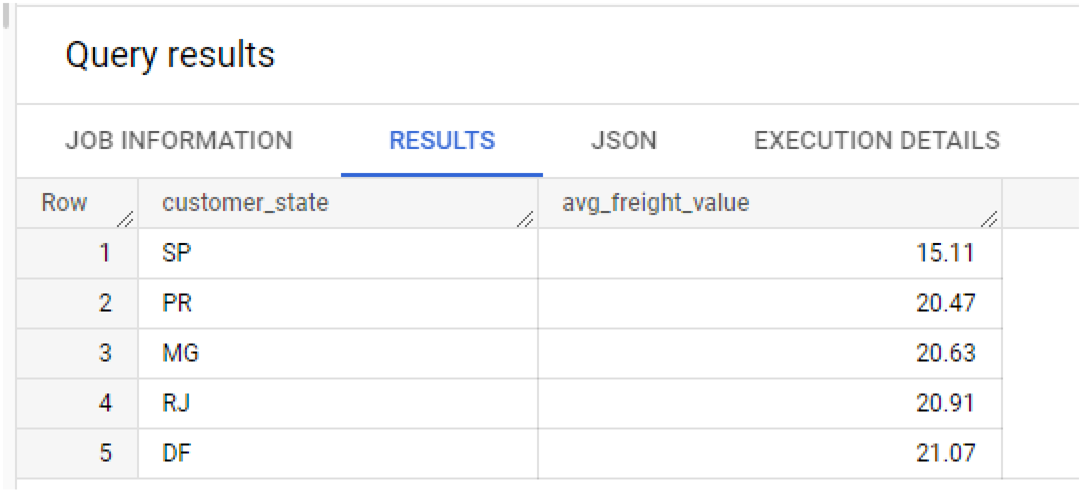
Row	customer_state	avg_freight_value	avg_time_to_delivery	avg_diff_estimated_delivery
1	AC	40.05	20.33	20.01
2	AL	35.87	23.99	7.98
3	AM	33.31	25.96	18.98
4	AP	34.16	27.75	17.44
5	BA	26.49	18.77	10.12
6	CE	32.73	20.54	10.26
7	DF	21.07	12.5	11.27
8	ES	22.03	15.19	9.77
9	GO	22.56	14.95	11.37
10	MA	38.49	21.2	9.11

4. Sort the data to get the following:

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

```
SELECT c.customer_state,ROUND(AVG(oi.freight_value),2) as
avg_freight_value,

FROM target.orders o

JOIN target.order_items as oi

ON o.order_id=oi.order_id

JOIN target.customers as  c

ON o.customer_id = c.customer_id

where order_delivered_customer_date IS NOT NULL

GROUP BY c.customer_state

order by avg_freight_value
```

```
LIMIT 5
```

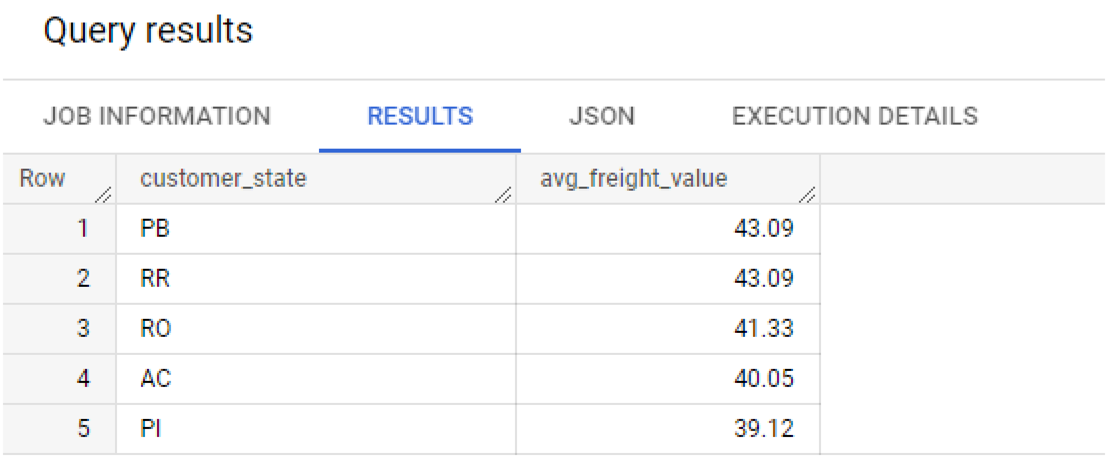


Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS

Row	customer_state	avg_freight_value
1	SP	15.11
2	PR	20.47
3	MG	20.63
4	RJ	20.91
5	DF	21.07

```
SELECT c.customer_state,ROUND(AVG(oi.freight_value),2) as
avg_freight_value,

FROM target.orders o

JOIN target.order_items as oi

ON o.order_id=oi.order_id

JOIN target.customers as  c

ON o.customer_id = c.customer_id

where order_delivered_customer_date IS NOT NULL

GROUP BY c.customer_state

order by avg_freight_value DESC

LIMIT 5
```

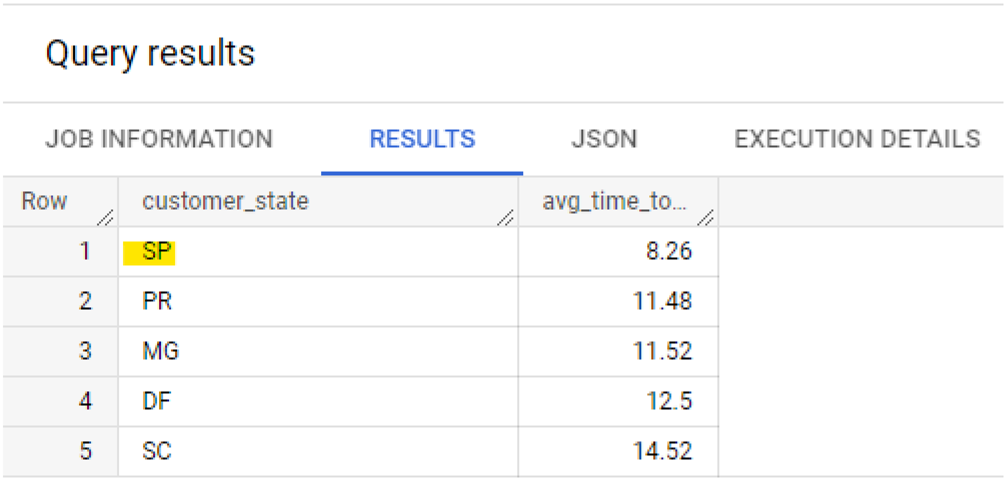

Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS

Row	customer_state	avg_freight_value
1	PB	43.09
2	RR	43.09
3	RO	41.33
4	AC	40.05
5	PI	39.12

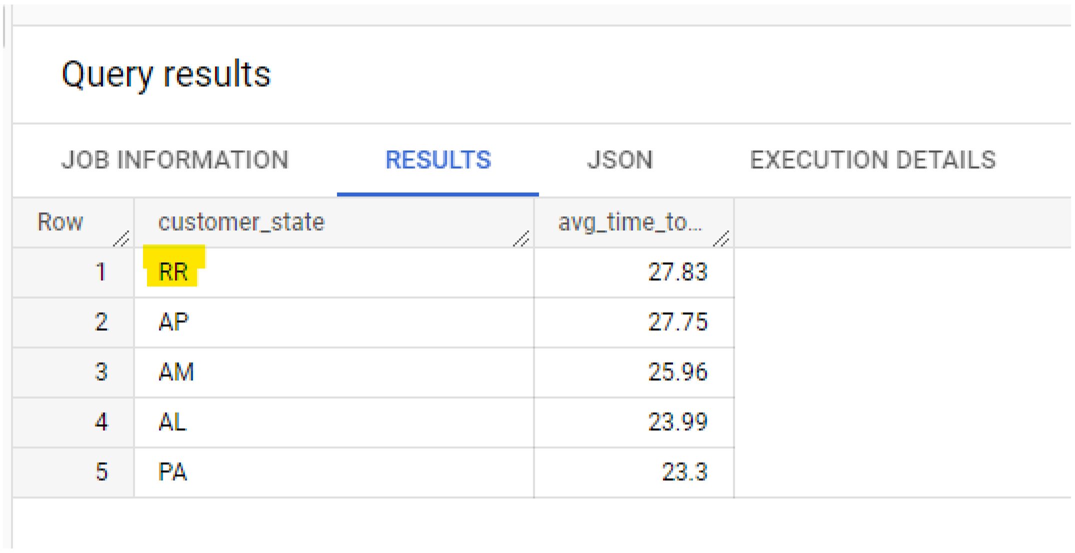
2. Top 5 states with highest/lowest average time to delivery

```
SELECT c.customer_state,

ROUND(AVG(DATE_DIFF(o.order_delivered_customer_date,o.order_purchase_time
stamp, DAY)),2) AS avg_time_to_delivery

FROM target.orders o

JOIN target.order_items as oi

ON o.order_id=oi.order_id

JOIN target.customers as  c

ON o.customer_id = c.customer_id

where order_delivered_customer_date IS NOT NULL

GROUP BY c.customer_state

order by avg_time_to_delivery

LIMIT 5
```


Query results

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

```
SELECT c.customer_state,

ROUND(AVG(DATE_DIFF(o.order_delivered_customer_date,o.order_purchase_time
stamp, DAY)),2) AS avg_time_to_delivery

FROM target.orders o

JOIN target.order_items as oi

ON o.order_id=oi.order_id

JOIN target.customers as  c

ON o.customer_id = c.customer_id

where order_delivered_customer_date IS NOT NULL

GROUP BY c.customer_state

order by avg_time_to_delivery DESC

LIMIT 5
```

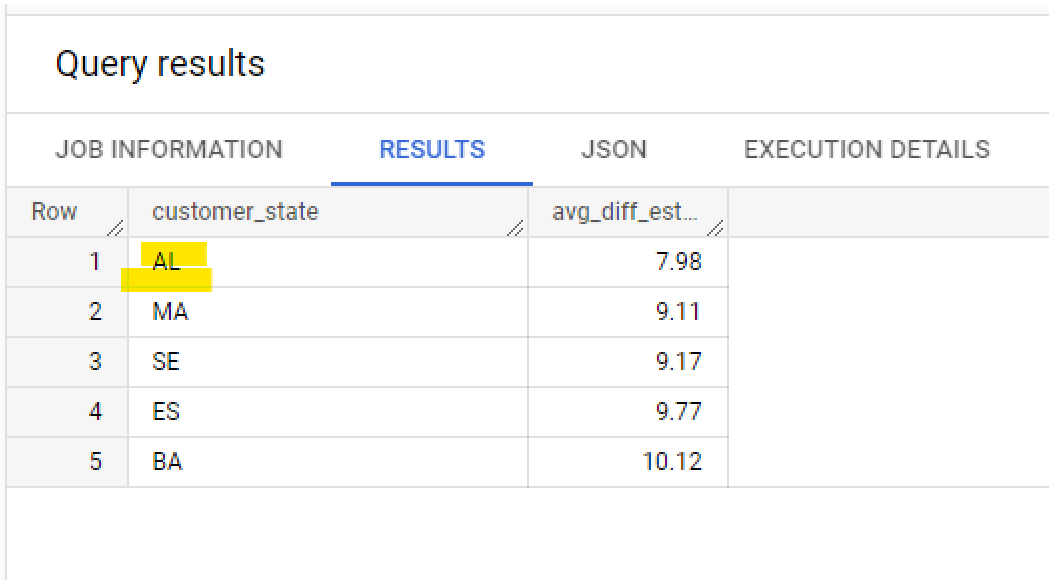

Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS

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

3. Top 5 states where delivery is speedy/ not so fast compared to estimated date

```
SELECT c.customer_state,

ROUND(AVG(DATE_DIFF(o.order_estimated_delivery_date,o.order_delivered_cus
tomer_date, DAY)),2) AS avg_diff_estimated_delivery

FROM target.orders o

JOIN target.order_items as oi

ON o.order_id=oi.order_id

JOIN target.customers as  c

ON o.customer_id = c.customer_id

where order_delivered_customer_date IS NOT NULL

GROUP BY c.customer_state

order by avg_diff_estimated_delivery LIMIT 5
```

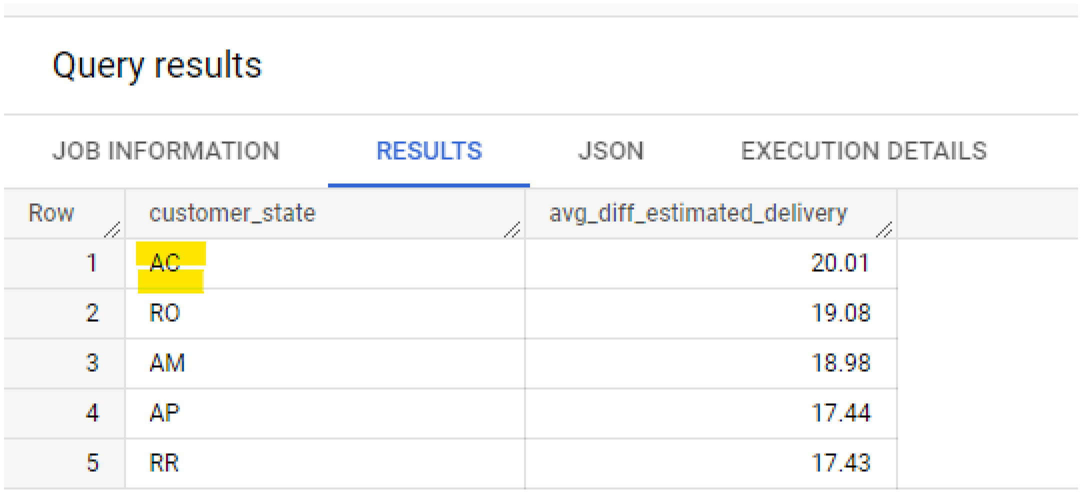


Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	avg_diff_est...
1	AL	7.98
2	MA	9.11
3	SE	9.17
4	ES	9.77
5	BA	10.12

```
SELECT c.customer_state,

ROUND(AVG(DATE_DIFF(o.order_estimated_delivery_date,o.order_delivered_customer_
date, DAY)),2) AS avg_diff_estimated_delivery

FROM target.orders o

JOIN target.order_items as oi

ON o.order_id=oi.order_id

JOIN target.customers as  c

ON o.customer_id = c.customer_id

where order_delivered_customer_date IS NOT NULL

GROUP BY c.customer_state

order by avg_diff_estimated_delivery DESC  LIMIT 5
```



Query results

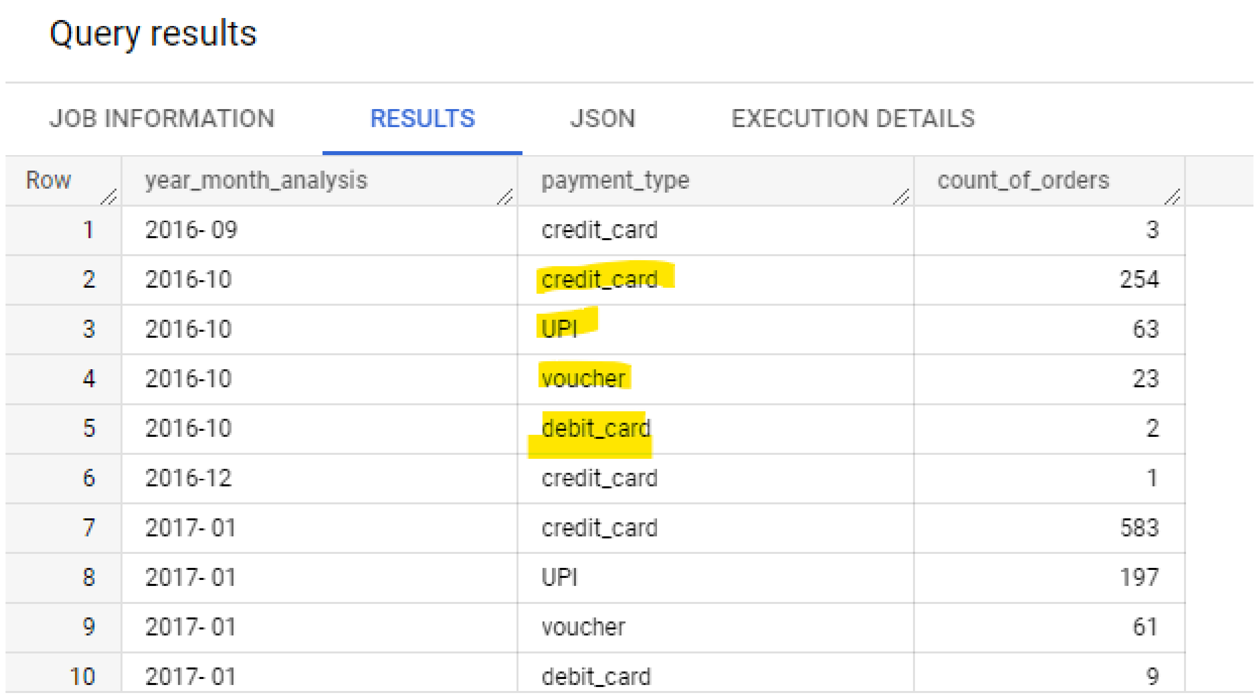
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	avg_diff_estimated_delivery
1	AC	20.01
2	RO	19.08
3	AM	18.98
4	AP	17.44
5	RR	17.43

6. Payment type analysis:

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

```
with cte as (

  SELECT case when LENGTH( CAST(EXTRACT(MONTH FROM o.order_purchase_timestamp)
as STRING)) = 1 then

  CONCAT(EXTRACT(YEAR FROM o.order_purchase_timestamp),'- 0',EXTRACT(MONTH FROM
o.order_purchase_timestamp))

  else

  CONCAT(EXTRACT(YEAR FROM o.order_purchase_timestamp),'-',EXTRACT(MONTH FROM
o.order_purchase_timestamp))

  end as year_month_analysis,

  pi.payment_type

  FROM target.orders as o

  JOIN target.payments as pi

  ON o.order_id = pi.order_id
)

SELECT  year_month_analysis ,payment_type, count(payment_type) as
count_of_orders  FROM cte

GROUP BY year_month_analysis , payment_type

ORDER BY year_month_analysis
```

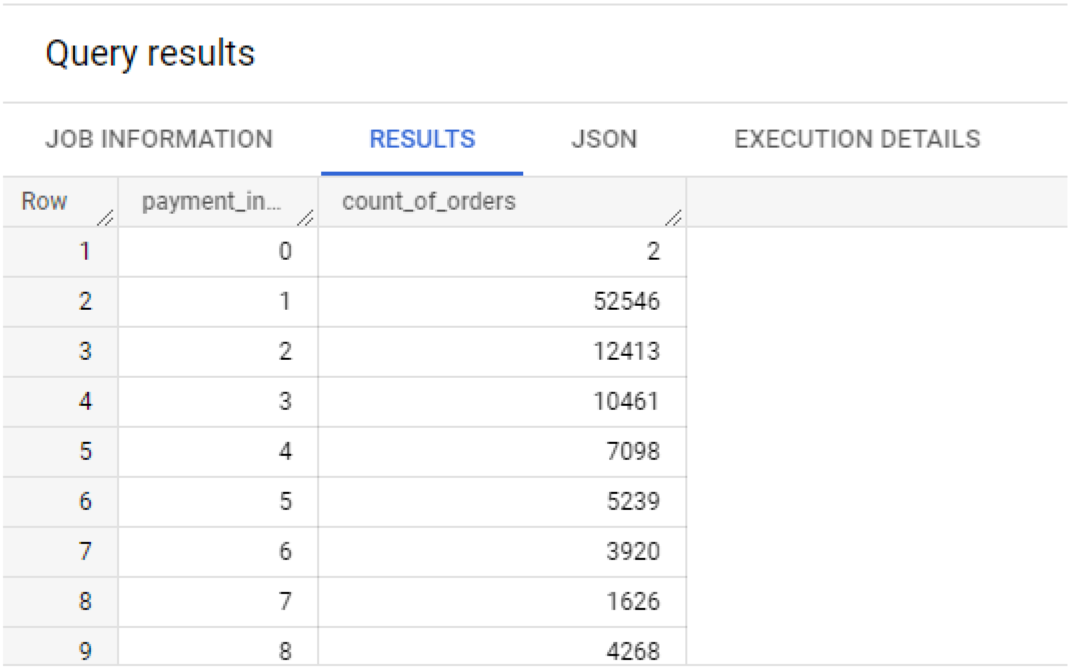


Query results

Row	year_month_analysis	payment_type	count_of_orders
1	2016- 09	credit_card	3
2	2016-10	credit_card	254
3	2016-10	UPI	63
4	2016-10	voucher	23
5	2016-10	debit_card	2
6	2016-12	credit_card	1
7	2017- 01	credit_card	583
8	2017- 01	UPI	197
9	2017- 01	voucher	61
10	2017- 01	debit_card	9

2. Distribution of payment installments and count of orders

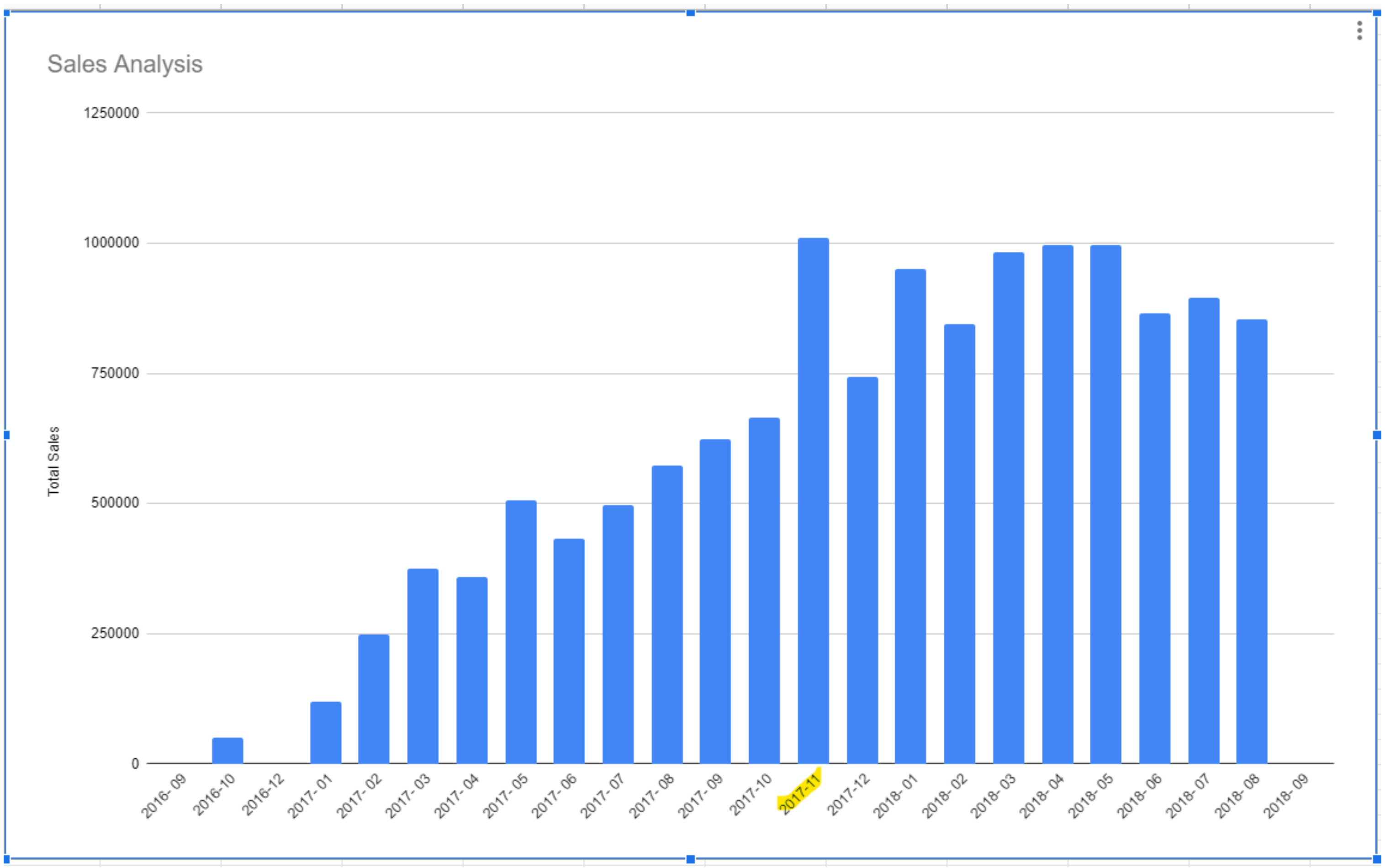
```
SELECT payment_installments, count(order_id) as count_of_orders FROm
`target.payments`

GROUP BY payment_installments
```


Query results

Row	payment_in...	count_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
9	8	4268

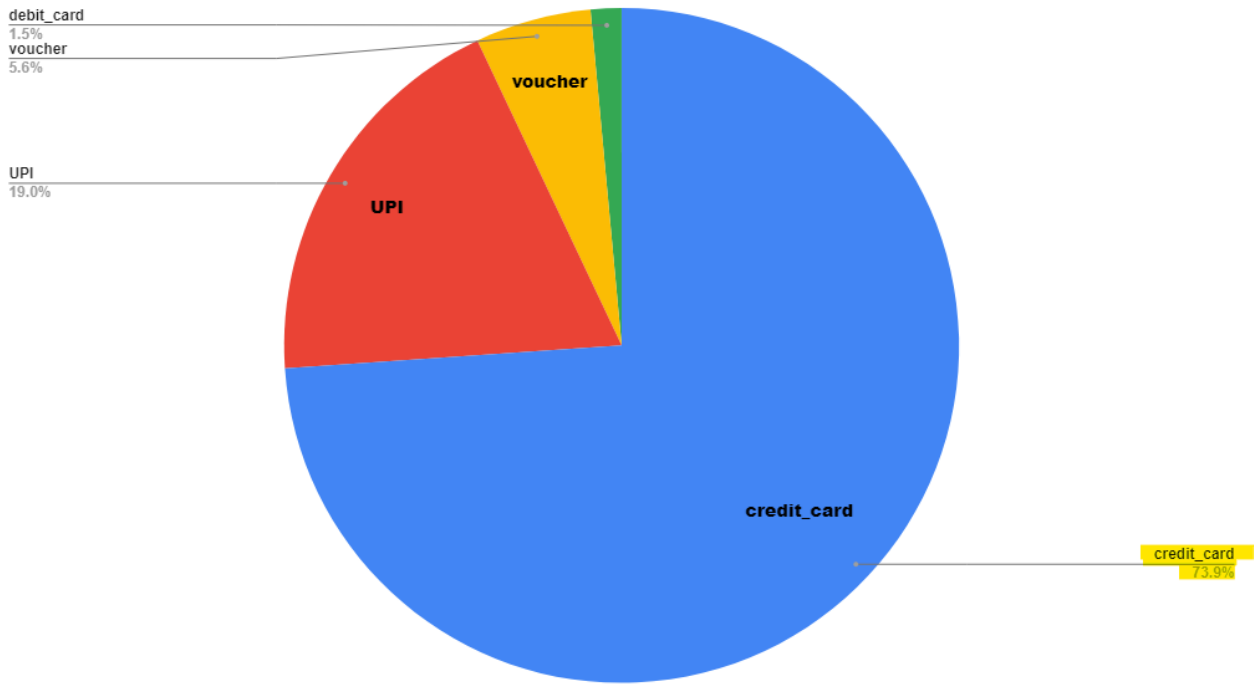
7. Actionable Insights

1. The Overall trend in e-commerce in Brazil is growing, with the maximum sales in November 2017.



2. The Overall trend in e-commerce in Brazil is growing, with the maximum sales in November 2017.
3. The maximum number of orders are placed in the afternoon between 1:00 P.M to 6:00 P.M
4. The Maximum number of customers are from Sao Paulo (SP) state in brazil with around forty thousand customers.
5. Roraima(RR) State in brazil has the least number of customers
6. The percentage Increase in total sales from 2017(Jan to Aug) to 2018(Jan to Aug) is 97%
7. Roraima(RR) State in brazil has the highest average fright value of 40

8. Sao Paulo (SP) least average fright value of 15
9. Paraíba(PB) state of brazil has the highest average order value
10. Sao Paulo (SP) has the least average order value
11. Sao Paulo (SP) has the fastest delivery in brazil, on the other hand, Roraima(RR) has the slowest average delivery time.
12. Around 74% of orders are placed using Credit Cards making it the most preferred way of payment in Brazil



8. Recommendations

1. As the maximum number of orders are placed in the afternoon time, Target can offer discounts at other times to distribute the orders.
2. Roraima(RR) has the highest Average fright value and the slowest delivery, Target should Improve this by keeping a minimum order value for Roraima(RR).
3. Acre(AC) state in brazil has the worst Delivery Estimation system, Target should work on improving this
4. As Credit Card is the most used mode of payment and Target has to pay commission to the Credit Cards companies, Target should give more offers on other modes of payment to encourage other modes of payment as well.
5. Target Should give discounts on March, as it is the least grossing month