

1.1. Data type of all columns in the "customers" table.

select

TABLE_NAME,

COLUMN_NAME,

DATA_TYPE

from `target`.INFORMATION_SCHEMA.COLUMNS

where TABLE_NAME = 'customers';

Query results

[SAVE R](#)

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	TABLE_NAME	COLUMN_NAME	DATA_TYPE				
1	customers	customer_id	STRING				
2	customers	customer_unique_id	STRING				
3	customers	customer_zip_code_prefix	INT64				
4	customers	customer_city	STRING				
5	customers	customer_state	STRING				

1.2. Get the time range between which the orders were placed.

select

min(order_purchase_timestamp) as `start_date`,

max(order_purchase_timestamp) as `end_date`

from `target.orders`;

Query results

[SAVE R](#)

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	start_date	end_date					
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC					

Insight: Time range for order is placed between 4th September 2016 to 17th October 2018

Recommendation: N/A

1.3. Count the Cities & States of customers who ordered during the given period.

select

count(distinct c.customer_city) as `city`,

count(distinct c.customer_state) as `state`

from `target.customers` c

inner join

`target.orders` o

on c.customer_id = o.customer_id

where o.order_purchase_timestamp between (select
min(order_purchase_timestamp) from `target.orders`) and (select
max(order_purchase_timestamp) from `target.orders`);

Query results

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	EXECUTION GRAP
Row	city	state					
1	4119	27					

Insight: Total 4119 cities and 27 states of customers who order the products during 4th September 2016 to 17th October 2018.

Recommendation: N/A

2.1. Is there a growing trend in the no. of orders placed over the past years?

select

t.year,

t.month,

t.orders_count,

sum(t.orders_count) OVER(partition by t.year) as `sum_of_orders`

from

(select

extract(year from order_purchase_timestamp) as `year`,

extract(month from order_purchase_timestamp) as `month`,

count(order_id) as `orders_count`

from `target.orders`

group by year,month

order by year,month) as t

order by t.year

Query results

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	EXECU
Row	year ▼	month ▼	orders_count ▼	sum_of_orders ▼			
1	2016	10	324	329			
2	2016	9	4	329			
3	2016	12	1	329			
4	2017	11	7544	45101			
5	2017	12	5673	45101			
6	2017	4	2404	45101			
7	2017	7	4026	45101			
8	2017	10	4631	45101			
9	2017	6	3245	45101			
10	2017	9	4285	45101			

Query results

JOB INFORMATION

RESULTS

CHART

PREVIEW

JSON

EXECUTION DETAILS

EXECUTION GRAPH

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

Insight: There is a growing trend over the period 2016 - 2018

Recommendation: N/A

2.2 Can we see some kind of monthly seasonality in terms of the no. of orders being placed?

select

extract(month from order_purchase_timestamp) as `month`,

extract(year from order_purchase_timestamp) as `year`,

count(order_id) as `total_no_of_orders`

from `target.orders`

group by year,month

order by year,month

Row	month	year	total_no_of_orders	
9	6	2017	3245	
10	7	2017	4026	
11	8	2017	4331	
12	9	2017	4285	
13	10	2017	4631	
14	11	2017	7544	

Row	month	year	total_no_of_orders
18	3	2018	7211
19	4	2018	6939
20	5	2018	6873
21	6	2018	6167
22	7	2018	6292
23	8	2018	6512

Insight: Number of orders are having monthly seasonality during the period July to October month of 2017 and April to July month Of 2018.

Recommendation: Provide Product promotion like

discount,

combo,

advertisement,

EMI options,

Regular Inventory check of high demand products,

Avoid Delivery delays

2.3. During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)

0-6 hrs : Dawn

7-12 hrs : Mornings

13-18 hrs : Afternoon

19-23 hrs : Night

```

select
CASE
WHEN EXTRACT(HOUR FROM order_purchase_timestamp) > 0 and
      EXTRACT(HOUR FROM order_purchase_timestamp) <
      6 THEN 'Dawn'

WHEN EXTRACT(HOUR FROM order_purchase_timestamp) > 7 and
      EXTRACT(HOUR FROM order_purchase_timestamp) <
      12 THEN 'Mornings'

WHEN EXTRACT(HOUR FROM order_purchase_timestamp) > 13 and
      EXTRACT(HOUR FROM order_purchase_timestamp) <
      18 THEN 'Afternoon'

ELSE
'NIGHT'

END as `Time_of_the_day`,
count(order_id) as `total_no_of_orders`
from target.orders
group by `Time_of_the_day`
order by `total_no_of_orders`

```

Query results

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON
Row	Time_of_the_day ▼	total_no_of_orders			
1	Dawn	2346			
2	Mornings	20507			
3	Afternoon	25848			
4	NIGHT	50740			

Insight: Most of the orders were placed during Night time and order total is “50740”.

Recommendation: N/A

3.1. Get the month on month no. of orders placed in each state.

select

```
c.customer_state as `state`,
extract(year from o.order_purchase_timestamp) as `year`,
extract(month from o.order_purchase_timestamp) as `month`,
count(o.order_id) as `no_of_orders`
```

from `target.customers` c

inner join

`target.orders` o

on c.customer_id = o.customer_id

group by `state`,`year`,`month`

order by `year`,`month`;

Query results

[SAVE RESU](#)

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	state	year	month	no_of_orders			
1	RR	2016	9	1			
2	RS	2016	9	1			
3	SP	2016	9	2			
4	SP	2016	10	113			
5	RS	2016	10	24			
6	RJ	2016	10	56			
7	MT	2016	10	3			
8	GO	2016	10	9			
9	MG	2016	10	40			
10	CE	2016	10	8			
11	SC	2016	10	11			
12	AL	2016	10	2			

Insight: SP, MG, RJ states have consistent number of orders during January, 2017 – August,2018.

There are no repeated order from the same customer.

Recommendation: N/A

3.2. How are the customers distributed across all the states?

select

```
customer_state as `state`,
```

```
count(customer_id) as `no_of_customers`
```

```
from `target.customers`
```

```
group by `state`
```

```
order by `no_of_customers` desc
```

Query results

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	EXECUTION
Row	state	no_of_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					

Insight: SP, MG, RJ have population more than 10000 so order is placed high among this region.

Recommendation: AC, AP, RR states have least customers. Hence perform additional promotion activities to increase sales.

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

You can use the "payment_value" column in the payments table to get the cost of orders.


```

WITH `A` as(
select *,
    round((((t.`cost_of_orders` - lag(t.`cost_of_orders`,1) over(order by
t.`month`)))/lag(t.`cost_of_orders`,1) over(order by t.`month`)) * 100,2) as
`percent_increase`
from
(
select
    extract(year from order_purchase_timestamp) as `year`,
    FORMAT_DATE('%m', order_purchase_timestamp) as `month`,
    round(sum(p.payment_value),2) as `cost_of_orders`,
from `target.orders` o
inner join
`target.payments` p
on o.order_id = p.order_id
where order_purchase_timestamp between '2017-01-01' and '2017-08-31'
group by `month`,`year`
) as t
order by t.`month`),

```

```

`B` as (select *,
    round((((t.`cost_of_orders` - lag(t.`cost_of_orders`,1) over(order by
t.`month`)))/lag(t.`cost_of_orders`,1) over(order by t.`month`)) * 100,2) as
`percent_increase`
from
(
select

```

```
extract(year from order_purchase_timestamp) as `year`,
FORMAT_DATE('%m', order_purchase_timestamp) as `month`,
round(sum(p.payment_value),2) as `cost_of_orders`,
from `target.orders` o
inner join
`target.payments` p
on o.order_id = p.order_id
where order_purchase_timestamp between '2018-01-01' and '2018-08-31'
group by `month`,`year`
) as t
order by t.`month`)
```

```
select
  A.year as `YEAR`,
  A.month as `MONTH`,
  A.cost_of_orders,
  A.percent_increase
from A
UNION ALL
select
  B.year,
  B.month,
  B.cost_of_orders,
  B.percent_increase
from B
order by `YEAR`,`MONTH`
```

Query results

[SAV](#)

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	YEAR	MONTH	cost_of_orders	percent_increase	Total_orders		
1	2017	01	138488.04	null	850		
2	2017	02	291908.01	110.78	1886		
3	2017	03	449863.6	54.11	2837		
4	2017	04	417788.03	-7.13	2571		
5	2017	05	592918.82	41.92	3944		
6	2017	06	511276.38	-13.77	3436		
7	2017	07	592382.92	15.86	4317		
8	2017	08	650481.47	9.81	4390		
9	2018	01	1115004.18	null	7563		
10	2018	02	992463.34	-10.99	6952		
11	2018	03	1159652.12	16.85	7512		
12	2018	04	1160785.48	0.1	7209		
13	2018	05	1153982.15	-0.59	7135		

Insight: In 2017, the % increase in the cost of order is on a growing trend. Whereas 2018 shows a declining trend because of reduction in number of orders

In 2018, 4 months (February, May, June, August) having negative percentage value and the same thing in 2017 has (April & June).

Recommendation: The reason for the reduction of the orders has to be analysed and needs improvement.

4.2 Calculate the Total & Average value of order price for each state.

select

c.customer_state as `state`,

round(sum(oi.price),2) as `Total_price`,

round(sum(oi.price)/count(o.order_id),2) as `Average_price`

from `target.customers` c

inner join

`target.orders` o

on c.customer_id = o.customer_id

inner join `target.order_items` oi

on o.order_id = oi.order_id

group by `state`

order by `state`

Query results

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	state	Total_price	no_of_orders	Average_price			
1	PB	115268.08	602	191.48			
2	AL	80314.81	444	180.89			
3	AC	15982.95	92	173.73			
4	RO	46140.64	278	165.97			
5	PA	178947.81	1080	165.69			
6	AP	13474.3	82	164.32			
7	PI	86914.08	542	160.36			
8	TO	49621.74	315	157.53			
9	RN	83034.98	529	156.97			
10	CF	227254.71	1478	153.76			

Insight: Average price is high in top three states such as PB, AL, AC.

Recommendation: Can focus to increase in the number of orders.

4.3 Calculate the Total & Average value of order freight for each state.

select

c.customer_state as `state`,

round(sum(oi.freight_value),2) as `Freight_value`,

count(o.order_id) as `no_of_orders`,

round(sum(oi.freight_value)/count(o.order_id),2) as
Average_freight_value`

from `target.customers` c

inner join

`target.orders` o

on c.customer_id = o.customer_id

inner join `target.order_items` oi

on o.order_id = oi.order_id

group by `state`

order by `freight_value`

Query results

[SAV](#)

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	state	Freight_value	no_of_orders	Average_freight_value			
1	RR	2235.19	52	42.98			
2	AP	2788.5	82	34.01			
3	AC	3686.75	92	40.07			
4	AM	5478.89	165	33.21			
5	RO	11417.38	278	41.07			
6	TO	11732.68	315	37.25			
7	SE	14111.47	385	36.65			
8	AL	15914.59	444	35.84			
9	RN	18860.1	529	35.65			
10	MS	19144.03	819	23.37			

Insight: Less freight value in top three states such as RR, AP, AC.

Recommendation: States like SP, RJ, MG, RS has high freight value. Hence alternative freight vendors can be approached for less amount.

5.1 Find the no. of days taken to deliver each order from the order's purchase date as delivery time.

Also, calculate the difference (in days) between the estimated & actual delivery date of an order.

Do this in a single query.

You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula:

$$\text{time_to_deliver} = \text{order_delivered_customer_date} - \text{order_purchase_timestamp}$$

$$\text{diff_estimated_delivery} = \text{order_estimated_delivery_date} - \text{order_delivered_customer_date}$$

select

order_id,

DATE_DIFF(order_delivered_customer_date,
order_purchase_timestamp,day) as `time_to_deliver`,

DATE_DIFF(order_estimated_delivery_date,order_delivered_customer_d
ate,day) as `diff_estimated_delivery`

from `target.orders`

where order_status = 'delivered' and

order_delivered_customer_date is not null and

order_estimated_delivery_date is not null

order by `diff_estimated_delivery`;

Query results

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	order_id	time_to_deliver	diff_estimated_delivery				
1	1b3190b2dfa9d789e1f14c05b647a14a	208	-188				
2	ca07593549f1816d26a572e06dc1eab6	209	-181				
3	47b40429ed8cce3aee9199792275433f	191	-175				
4	2fe324feb907e3ea3f2aa9650869fa5	189	-167				
5	285ab9426d6982034523a855f55a885e	194	-166				
6	440d0d17af552815d15a9e41abe49359	195	-165				
7	c27815f7e3dd0b926b58552628481575	187	-162				
8	0f4519c5f1c541ddec9f21b3bdd533a	194	-161				
9	d24e8541128cea179a11a65176e0a96f	175	-161				
10	2d7561026d542c8dbd8f0daeadf67a43	188	-159				

Insight: Maximum number of orders exceeded the estimated delivery date.

Recommendation: can focus to reduce the exceeded the estimated delivery date.

5.2 Find out the top 5 states with the highest & lowest average freight value.

WITH `A` as (select

c.customer_state as `state`,

round(AVG(oi.freight_value),2) as `average_freight_value`,

```

RANK() OVER(order by round(AVG(oi.freight_value),2)) as `lowest_value`,
RANK() OVER(order by round(AVG(oi.freight_value),2) desc) as `highest_value`
from `target.customers` c
inner join
`target.orders` o
on c.customer_id = o.customer_id
inner join `target.order_items` oi
on o.order_id = oi.order_id
group by `state`
order by `average_freight_value`)

select `A`.`state`,`A`.`average_freight_value`
from A
where `A`.`lowest_value` <= 5 or `A`.`highest_value` <= 5
order by `A`.`average_freight_value`

```

Query results

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	E
Row	state	average_freight_valu					
1	SP	15.15					
2	PR	20.53					
3	MG	20.63					
4	RJ	20.96					
5	DF	21.04					
6	PI	39.15					
7	AC	40.07					
8	RO	41.07					
9	PB	42.72					
10	RR	42.98					

Insight: The states of SP, PR, MG, RJ, and DF have the top 5 lowest average freight values and the states of RR, PB, RO, AC, PI have the top 5 highest average freight values.

Recommendation: N/A

5.3 Find out the top 5 states with the highest & lowest average delivery time.

select

t.state,

round(t.average_delivery_time,2) as `average_delivery_time`

from

(select

c.customer_state as `state`,

AVG(DATE_DIFF(o.order_delivered_customer_date,o.order_purchase_timestamp,DAY)) as `average_delivery_time`,

RANK() OVER(order by
AVG(DATE_DIFF(o.order_delivered_customer_date,o.order_purchase_timestamp,DAY))) as `lowest_value`,

RANK() OVER(order by
AVG(DATE_DIFF(o.order_delivered_customer_date,o.order_purchase_timestamp,DAY)) desc) as `highest_value`

from `target.customers` c

inner join `target.orders` o

on c.customer_id = o.customer_id

group by `state`) as t

where t.lowest_value <= 5 or t.highest_value <= 5

order by `average_delivery_time`

Query results

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS
Row	state ▼	average_delivery_time				
1	SP	8.3				
2	PR	11.53				
3	MG	11.54				
4	DF	12.51				
5	SC	14.48				
6	PA	23.32				
7	AL	24.04				
8	AM	25.99				
9	AP	26.73				
10	RR	28.98				

Insight: The states of SP, PR, MG, DF, and SC have the top 5 lowest average freight values and the states of PA, AL, AM, AP, RR have the top 5 highest average freight values.

Recommendation: N/A

5.4 Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.

You can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.

select

c.customer_state as `state`,

CEIL(AVG(DATE_DIFF(o.order_estimated_delivery_date,o.order_delivered_customer_date,day))) as `fastest_delivery`

from `target.customers` c

inner join `target.orders` o

on c.customer_id = o.customer_id

where o.order_status = 'delivered'

group by `state`

order by `fastest_delivery`

limit 5

Query results

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS
Row	state	fastest_delivery				
1	AL	8.0				
2	MA	9.0				
3	BA	10.0				
4	CE	10.0				
5	ES	10.0				

Insight: Fastest delivery state is below,

AL, MA, BA, CE, ES

Recommendation: N/A

6.1 Find the month on month no. of orders placed using different payment types.

select

extract(year from o.order_purchase_timestamp) as `year`,

FORMAT_TIMESTAMP('%B', o.order_purchase_timestamp) as `month`,

count(o.order_id) as `No_of_orders`,

p.payment_type as `mode_of_payment`

from `target.orders` o

inner join `target.payments` p

on o.order_id = p.order_id

group by `year`,`month`,`mode_of_payment`

order by `year`,`month`;

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	year	month	No_of_orders	mode_of_payment			
1	2016	December	1	credit_card			
2	2016	October	254	credit_card			
3	2016	October	23	voucher			
4	2016	October	2	debit_card			
5	2016	October	63	UPI			
6	2016	September	3	credit_card			
7	2017	April	202	voucher			
8	2017	April	1846	credit_card			
9	2017	April	496	UPI			
10	2017	April	27	debit_card			

Insight: Most of the orders were placed using “ credit card ” mode of payment.

It also shows growing trend over the months.

Recommendation: N/A

6.2 Find the no. of orders placed on the basis of the payment installments that have been paid.

select

payment_installments `Payment_Installments`,

count(o.order_id) as `No_of_orders`

from `target.orders` o

inner join `target.payments` p

on o.order_id = p.order_id

where p.payment_installments >= 1

group by payment_installments

order by `Payment_Installments`

Query results

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	E
Row	Payment_Installment	No_of_orders					
1	1	52546					
2	2	12413					
3	3	10461					
4	4	7098					
5	5	5239					
6	6	3920					
7	7	1626					
8	8	4268					
9	9	644					
10	10	5328					

Insight: Company received only one instalment for 55% of the order.

Recommendation: Collection follow up should be improved for quick customer repayment