

Context:

Leading Brazilian Retailer is a globally renowned brand and a prominent retailer in the United States. Leading Brazilian Retailer 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 Leading Brazilian 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 Leading Brazilian Retailer 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.

Dataset: <https://drive.google.com/drive/folders/1TGEc66YKbD443nslRi1bWgVd238gJCnb>

The data is available in 8 csv files:

1. customers.csv
2. sellers.csv
3. order_items.csv
4. geolocation.csv
5. payments.csv
6. reviews.csv
7. orders.csv
8. products.csv

The column description for these csv files is given below.

The **customers.csv** contain following features:

Features	Description
customer_id	ID of the consumer who made the purchase
customer_unique_id	Unique ID of the consumer
customer_zip_code_prefix	Zip Code of consumer's location
customer_city	Name of the City from where order is made
customer_state	State Code from where order is made (Eg. são paulo - SP)

The **sellers.csv** contains following features:

Features	Description
----------	-------------

seller_id	Unique ID of the seller registered
seller_zip_code_prefix	Zip Code of the seller's location
seller_city	Name of the City of the seller
seller_state	State Code (Eg. são paulo - SP)

The **order_items.csv** contain following features:

Features	Description
order_id	A Unique ID of order made by the consumers
order_item_id	A Unique ID given to each item ordered in the order
product_id	A Unique ID given to each product available on the site
seller_id	Unique ID of the seller registered in Leading Brazilian Retailer
shipping_limit_date	The date before which the ordered product must be shipped
price	Actual price of the products ordered
freight_value	Price rate at which a product is delivered from one point to another

The **geolocations.csv** contain following features:

Features	Description
geolocation_zip_code_prefix	First 5 digits of Zip Code
geolocation_lat	Latitude
geolocation_lng	Longitude
geolocation_city	City
geolocation_state	State

The **payments.csv** contain following features:

Features	Description
order_id	A Unique ID of order made by the consumers
payment_sequential	Sequences of the payments made in case of EMI
payment_type	Mode of payment used (Eg. Credit Card)
payment_installments	Number of installments in case of EMI purchase
payment_value	Total amount paid for the purchase order

The **orders.csv** contain following features:

Features	Description
order_id	A Unique ID of order made by the consumers

customer_id	ID of the consumer who made the purchase
order_status	Status of the order made i.e. delivered, shipped, etc.
order_purchase_timestamp	Timestamp of the purchase
order_delivered_carrier_date	Delivery date at which carrier made the delivery
order_delivered_customer_date	Date at which customer got the product
order_estimated_delivery_date	Estimated delivery date of the products

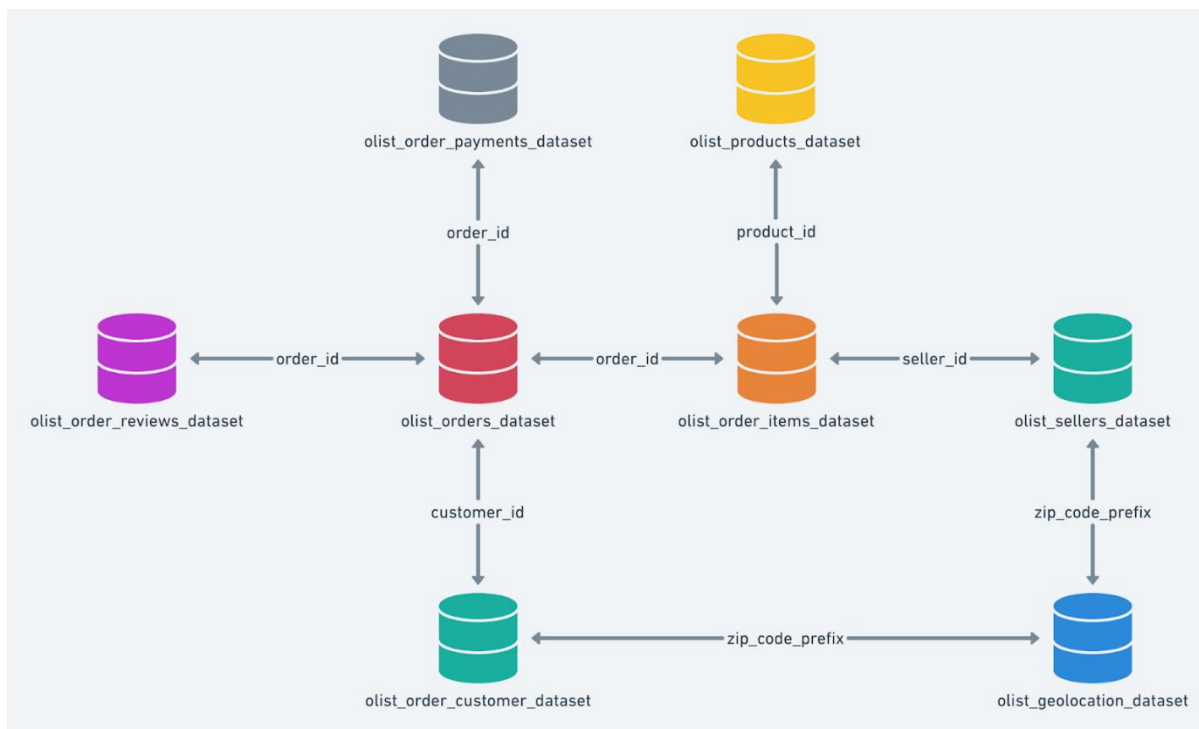
The **reviews.csv** contain following features:

Features	Description
review_id	ID of the review given on the product ordered by the order id
order_id	A Unique ID of order made by the consumers
review_score	Review score given by the customer for each order on a scale of 1-5
review_comment_title	Title of the review
review_comment_message	Review comments posted by the consumer for each order
review_creation_date	Timestamp of the review when it is created
review_answer_timestamp	Timestamp of the review answered

The **products.csv** contain following features:

Features	Description
product_id	A Unique identifier for the proposed project.
product_category_name	Name of the product category
product_name_lenght	Length of the string which specifies the name given to the products ordered
product_description_lenght	Length of the description written for each product ordered on the site
product_photos_qty	Number of photos of each product ordered available on the shopping portal
product_weight_g	Weight of the products ordered in grams
product_length_cm	Length of the products ordered in centimeters
product_height_cm	Height of the products ordered in centimeters
product_width_cm	Width of the product ordered in centimeters

Dataset schema:



Problem Statement:

Assuming you are a data analyst/ scientist at Leading Brazilian Retailer, you have been assigned the task of analyzing the given dataset to extract valuable insights and provide actionable recommendations.

What does 'good' look like?

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

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

customers								
QUERY	SHARE	COPY	SNAPSHOT	DELETE	EXPORT	REFRESH		
SCHEMA	DETAILS	PREVIEW	LINEAGE	DATA PROFILE	DATA QUALITY			
Filter Enter property name or value								
Field name	Type	Mode	Key	Collation	Default value	Policy tags	Description	
customer_id	STRING	NULLABLE						
customer_unique_id	STRING	NULLABLE						
customer_zip_code_prefix	INTEGER	NULLABLE						
customer_city	STRING	NULLABLE						
customer_state	STRING	NULLABLE						
EDIT SCHEMA VIEW ROW ACCESS POLICIES								

Insights: It will display the datatypes that each column of table is having.

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

```
select min(order_purchase_timestamp) start_order_purchase_timestamp,
max(order_purchase_timestamp) end_order_purchase_timestamp
```

```
from `EnterpriceBC_01.orders`;
```

Query results

SAVE RESULTS


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

Insights: The query will display the output of starting timestamp of order placed and end timestamp of order placed.

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

```
select count(distinct c.customer_state) as state,
count(distinct c.customer_city) as city
from `EnterpriceBC_01.orders` o
join `EnterpriceBC_01.customers` c
on o.customer_id = c.customer_id;
```

Query results

 SAVE RESULTS

JOB INFORMATIONRESULTSCHARTPREVIEWJSONEXECUTION DETAILSEXECUTION GRAPH

Row	state	city	
1	27	4119	

Insights: The query is displaying the count of state and city of customers who have ordered during given period

2. In-depth Exploration:

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

```
select order_status, ord_placed_year, count(ord_placed_year) y_on_y
from
```

```
(select order_id, order_status, format_date('%Y', order_purchase_timestamp)
ord_placed_year from `EnterpriseBC_01.orders`) a
where order_status not in ('unavailable','canceled')
group by 1,2
order by 2;
```

Query results					SAVE RESULTS	E
JOB INFORMATION	RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	order_status	ord_placed_year	y_on_y			
1	shipped	2016	9			
2	invoiced	2016	18			
3	delivered	2016	267			
4	processing	2016	2			
5	created	2017	4			
6	shipped	2017	530			
7	approved	2017	2			
8	invoiced	2017	175			
9	delivered	2017	43428			
10	processing	2017	240			

Results per page: 50 1 – 15 of 15

Insights: The query is displaying the count order placed, shipped, invoiced, delivered etc. year on year.

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

```
select order_status, ord_placed_month, count(ord_placed_month)
count_of_ord_month_wise
from
(select order_id, order_status, format_date('%Y-%m', order_purchase_timestamp)
ord_placed_month
from `EnterpriseBC_01.orders`) a
where order_status not in ('canceled','unavailable')
```

group by 1,2
order by 2;

Query results

SAVE RESULTS

E

JOB INFORMATION

RESULTS

CHART

PREVIEW

JSON

EXECUTION DETAILS

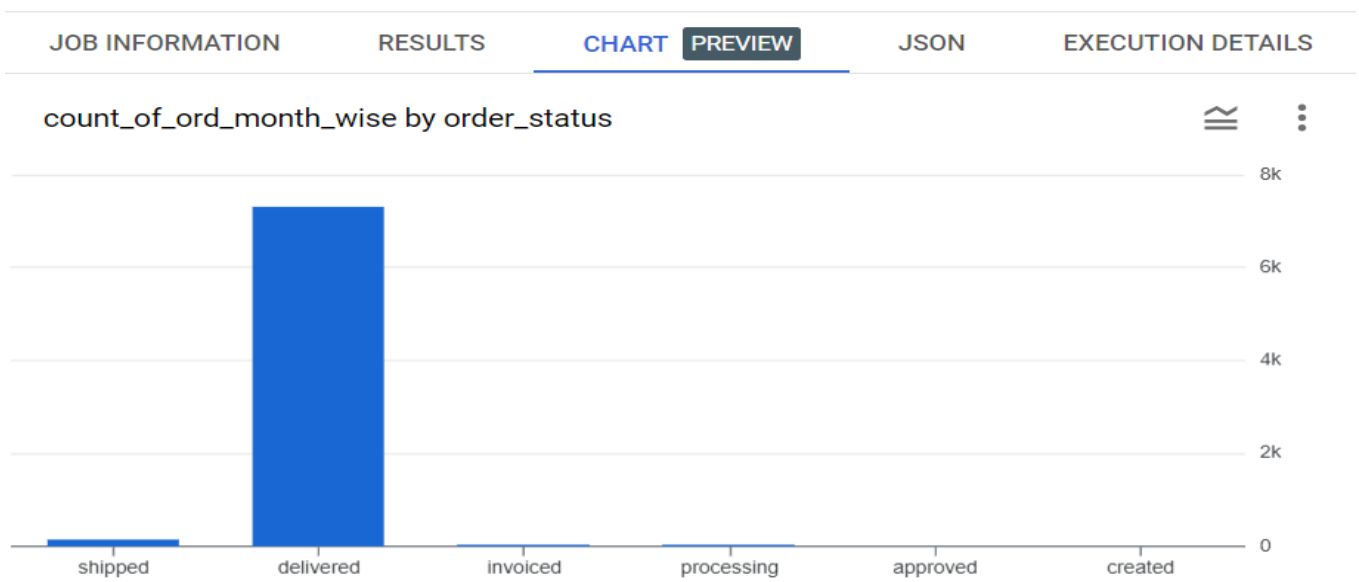
EXECUTION GRAPH

Row	order_status	ord_placed_month	count_of_ord_month
1	shipped	2016-09	1
2	delivered	2016-09	1
3	shipped	2016-10	8
4	invoiced	2016-10	18
5	delivered	2016-10	265
6	processing	2016-10	2
7	delivered	2016-12	1
8	shipped	2017-01	16
9	invoiced	2017-01	12
10	delivered	2017-01	750

Results per page:

50

1 – 50 of 91



Insights: The query is displaying the count order placed, shipped, invoiced, delivered etc. month wise.

- 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(DAY FROM order_purchase_timestamp) between 0 and 6 then
'Dawn'

```

when EXTRACT(DAY FROM order_purchase_timestamp) between 7 and 12 then
'Mornings'
when EXTRACT(DAY FROM order_purchase_timestamp) between 13 and 18 then
'Afternoon'
when EXTRACT(DAY FROM order_purchase_timestamp) between 19 and 23 then
'Night'
else 'unknown' end as time_of_day,
count(*) orders_count
from `EnterpriseBC_01.orders` o
join `EnterpriseBC_01.customers` c
on o.customer_id = c.customer_id
group by 1
order by 2 desc;

```

Query results			SAVE RESULTS ▼
JOB INFORMATION	RESULTS	CHART	PREVIEW
JSON	EXECUTION DETAILS	EXECUTION GRAPH	
Row	time_of_day ▼	orders_count ▼	
1	unknown	23361	
2	Afternoon	20399	
3	Dawn	19993	
4	Mornings	19638	
5	Night	16050	

Insights: The query is displaying that during what time of day users place most of the orders.

Recommendation: As we can see there is one row with unknown day time having most of the orders placed. If we will filter it and name it as time_of_day condition, so it will be easy for business and stakeholders to understand and do the analysis on it.

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

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

```

select distinct customer_state, format_date('%Y-%m',order_purchase_timestamp)
m_on_m
,count(order_status) as count_of_ord_by_state
from `EnterpriseBC_01.orders` o
join `EnterpriseBC_01.customers` c
on o.customer_id = c.customer_id
where order_status not in ('unavailable','canceled')
group by 1,2

```


order by 1,2, 3 desc;

Query results

SAVE RESULTS

E

JOB INFORMATION

RESULTS

CHART

PREVIEW

JSON

EXECUTION DETAILS

EXECUTION GRAPH

Row	customer_state	m_on_m	count_of_ord_by_sta
1	AC	2017-01	2
2	AC	2017-02	3
3	AC	2017-03	2
4	AC	2017-04	5
5	AC	2017-05	8
6	AC	2017-06	4
7	AC	2017-07	5
8	AC	2017-08	4
9	AC	2017-09	5
10	AC	2017-10	6

Results per page: 501 – 50 of 558

PERSONAL HISTORY

PROJECT HISTORY

Insights: The query is displaying the month on month orders placed, shipped, invoiced etc.

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

```
select distinct customer_state, count(customer_state) as state_wise_cust
from `EnterpriceBC_01.customers`
where customer_id is not null
group by 1
order by 2 desc;
```

Query results

SAVE RESULTS

JOB INFORMATION

RESULTS

CHART

PREVIEW

JSON

EXECUTION DETAILS

EXECUTION GRAPH

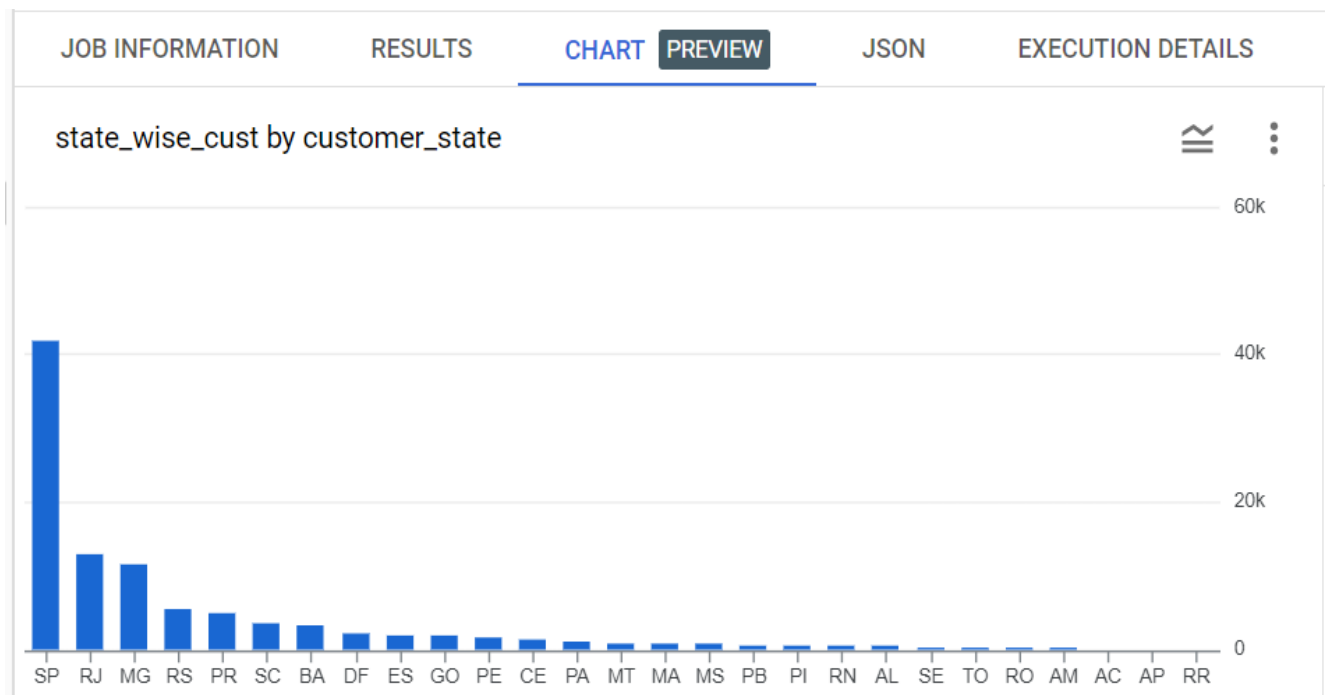
Row	customer_state	state_wise_cust
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637
7	BA	3380
8	DF	2140
9	ES	2033
10	GO	2020

Results per page: 50

1 – 27 of 27

PERSONAL HISTORY

PROJECT HISTORY



Insights: The query is displaying that how customers are distributed across all the states in descending order.

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

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.

```
select EXTRACT(MONTH from order_purchase_timestamp) as Month,
(
```

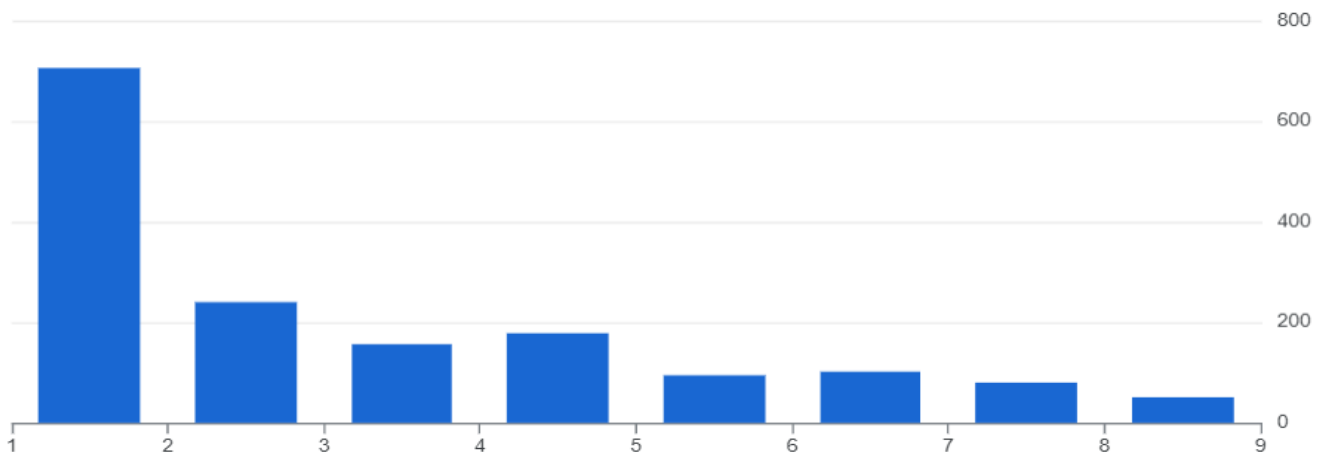
```

(
    sum(case when extract(YEAR FROM a.order_purchase_timestamp) = 2018 and
        Extract(MONTH FROM a.order_purchase_timestamp) between 1 and 8 then
b.payment_value end) -
    sum(case when extract(YEAR FROM a.order_purchase_timestamp) = 2017 and
        Extract(MONTH FROM a.order_purchase_timestamp) between 1 and 8 then
b.payment_value end))
    / sum(case when extract(YEAR FROM a.order_purchase_timestamp) = 2017 and
        Extract(MONTH FROM a.order_purchase_timestamp) between 1 and 8 then
b.payment_value end)) * 100
    as percentage_increased
from `EnterpriceBC_01.orders` a
join `EnterpriceBC_01.payments` b
ON a.order_id = b.order_id
WHERE
    Extract(year FROM a.order_purchase_timestamp) IN (2017, 2018) AND
    Extract(month FROM a.order_purchase_timestamp) BETWEEN 1 AND 8
GROUP BY 1
ORDER BY 1;

```

Query results				SAVE RESULTS ▼		
JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS
EXECUTION GRAPH						
Row	Month ▼	percentage_increased				
1	1	705.1266954171...				
2	2	239.9918145445...				
3	3	157.7786066709...				
4	4	177.8407701149...				
5	5	94.62734375677...				
6	6	100.2596912456...				
7	7	80.04245463390...				
8	8	51.60600520477...				

percentage_increased by Month



Insights: The query is delaying the percentage increased in the cost of orders placed from 2017 – 2018 between January to August.

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

```
select customer_state ,sum(price) total_order_price, avg(price) avg_order_price
from `EnterpriseBC_01.customers` c
join `EnterpriseBC_01.orders` o
on c.customer_id =o.customer_id
join `EnterpriseBC_01.order_items` oi
on o.order_id = oi.order_id
group by 1
order by 1,2 desc, 3 desc;
```

Query results

[SAVE RESULTS](#)

[EXP](#)

JOB INFORMATION	RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	customer_state	total_order_price	avg_order_price			
1	AC	15982.94999999...	173.7277173913...			
2	AL	80314.81	180.8892117117...			
3	AM	22356.84000000...	135.4959999999...			
4	AP	13474.29999999...	164.3207317073...			
5	BA	511349.99000000...	134.6012082126...			
6	CE	227254.7099999...	153.7582611637...			
7	DF	302603.9399999...	125.7705486284...			
8	ES	275037.3099999...	121.9137012411...			
9	GO	294591.9499999...	126.2717316759...			
10	MA	119648.2199999...	145.2041504854...			



Results per page: 50 1 – 27 of 27

Insights: The query is displaying total & average value of order price by each state.

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

```
select customer_state ,sum(freight_value) total_ord_fre_price, avg(freight_value)
avg_ord_fre_price
from `EnterpriceBC_01.customers` c
join `EnterpriceBC_01.orders` o
on c.customer_id =o.customer_id
join `EnterpriceBC_01.order_items` oi
on o.order_id = oi.order_id
group by 1
order by 1,2 desc, 3 desc;
```

Query results

 **SAVE RESULTS** 

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	customer_state	total_ord_fre_price	avg_ord_fre_price				
1	AC	3686.749999999...	40.07336956521...				
2	AL	15914.589999999...	35.84367117117...				
3	AM	5478.889999999...	33.20539393939...				
4	AP	2788.500000000...	34.00609756097...				
5	BA	100156.6799999...	26.36395893656...				
6	CE	48351.589999999...	32.71420162381...				
7	DF	50625.499999999...	21.04135494596...				
8	ES	49764.599999999...	22.05877659574...				
9	GO	53114.979999999...	22.76681525932...				
10	MA	31523.770000000...	38.25700242718...				

Results per page: 50 1 – 27 of 27

PERSONAL HISTORYPROJECT HISTORY

Insights: The query is displaying total & average value of order freight price by each state.

5. Analysis based on sales, freight and delivery time.

- 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:

- time_to_deliver** = order_delivered_customer_date - order_purchase_timestamp
- diff_estimated_delivery** = order_estimated_delivery_date - order_delivered_customer_date

```
select order_delivered_customer_date,order_purchase_timestamp,
       (date_diff(order_delivered_customer_date, order_purchase_timestamp, day))
       time_to_deliver,
       order_estimated_delivery_date,order_delivered_customer_date,
       (date_diff(order_estimated_delivery_date, order_delivered_customer_date, day))
       diff_estimated_delivery
from `EnterpriseBC_01.orders`;
```

Query results							
JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	order_delivered_customer_date	order_purchase_timestamp	time_to_deliver	order_estimated_delivery_date	order_delivered_customer_date_1	diff_estimated_delivery	
1	2018-03-21 22:03:51 UTC	2018-02-19 19:48:52 UTC	30	2018-03-09 00:00:00 UTC	2018-03-21 22:03:51 UTC	-12	
2	2016-11-09 14:53:50 UTC	2016-10-09 15:39:56 UTC	30	2016-12-08 00:00:00 UTC	2016-11-09 14:53:50 UTC	28	
3	2016-11-08 10:58:34 UTC	2016-10-03 21:01:41 UTC	35	2016-11-25 00:00:00 UTC	2016-11-08 10:58:34 UTC	16	
4	2017-05-16 14:49:55 UTC	2017-04-15 15:37:38 UTC	30	2017-05-18 00:00:00 UTC	2017-05-16 14:49:55 UTC	1	
5	2017-05-17 10:52:15 UTC	2017-04-14 22:21:54 UTC	32	2017-05-18 00:00:00 UTC	2017-05-17 10:52:15 UTC	0	
6	2017-05-16 09:07:47 UTC	2017-04-16 14:56:13 UTC	29	2017-05-18 00:00:00 UTC	2017-05-16 09:07:47 UTC	1	
7	2017-05-22 14:11:31 UTC	2017-04-08 21:20:24 UTC	43	2017-05-18 00:00:00 UTC	2017-05-22 14:11:31 UTC	-4	
8	2017-05-22 16:18:42 UTC	2017-04-11 19:49:45 UTC	40	2017-05-18 00:00:00 UTC	2017-05-22 16:18:42 UTC	-4	
9	2017-05-19 13:44:52 UTC	2017-04-12 12:17:08 UTC	37	2017-05-18 00:00:00 UTC	2017-05-19 13:44:52 UTC	-1	
10	2017-05-23 14:19:48 UTC	2017-04-19 22:52:59 UTC	33	2017-05-18 00:00:00 UTC	2017-05-23 14:19:48 UTC	-5	
Results per page: 50 1 - 50 of 99441 < >							

Insights: The is displaying the numbers days taken to deliver any orders as **time_to_deliver** and difference in days between estimated & actual delivery date as **diff_estimated_delivery**.

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

```
WITH T1 AS (SELECT c.customer_state,
                ROUND(AVG(b.freight_value), 2) AS avg_freight_value,
                DENSE_RANK() OVER(ORDER BY ROUND(AVG(b.freight_value), 2) DESC) Rank
FROM
    `EnterpriceBC_01.orders` a
JOIN
    `EnterpriceBC_01.order_items` b
ON a.order_id = b.order_id
JOIN
    `EnterpriceBC_01.customers` c
ON a.customer_id = c.customer_id
GROUP BY 1
ORDER BY 3 DESC)

(SELECT *, "Higest" AS High FROM T1
ORDER BY 3
LIMIT 5)

UNION ALL

(SELECT *, "Lowest" AS Low FROM T1
ORDER BY 3 DESC
LIMIT 5);
```

Query results

 SAV

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	customer_state	avg_freight_value	Rank	High			
1	RR	42.98	1	Higest			
2	PB	42.72	2	Higest			
3	RO	41.07	3	Higest			
4	AC	40.07	4	Higest			
5	PI	39.15	5	Higest			
6	SP	15.15	27	Lowest			
7	PR	20.53	26	Lowest			
8	MG	20.63	25	Lowest			
9	RJ	20.96	24	Lowest			
10	DF	21.04	23	Lowest			

avg_freight_value, Rank by customer_state

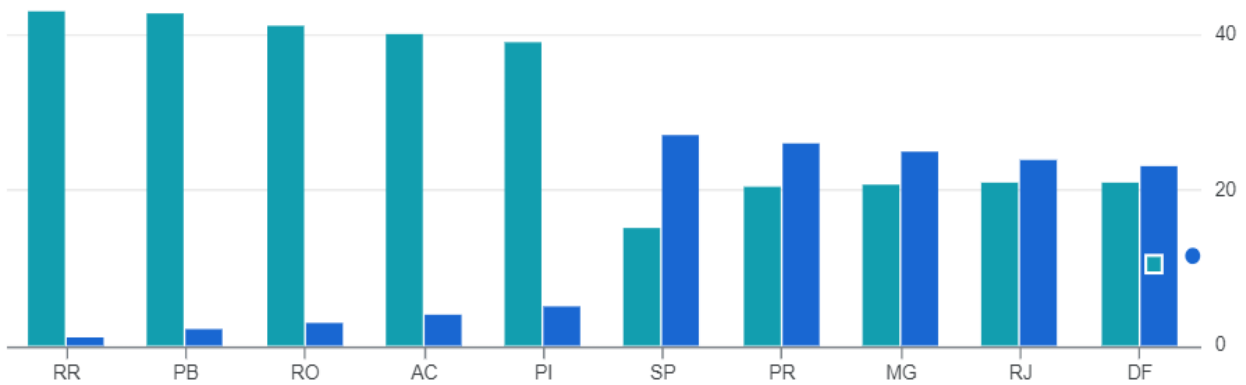


60

40

20

0



PERSONAL HISTORY

PROJECT HISTORY

Insights: The output of the query is displaying the top 5 states with the highest & lowest average freight value

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

```
WITH T1 AS
(select distinct c.customer_state,
    round(avg(date_diff(a.order_delivered_customer_date,
a.order_purchase_timestamp, day)), 2)
    as average_delivery_time,
    dense_rank() over(order by
(round(avg(date_diff(a.order_delivered_customer_date,
a.order_purchase_timestamp,
    day)),2) )) as rnk_delivery_time
FROM
    `EnterpriseBC_01.orders` a
left JOIN
    `EnterpriseBC_01.order_items` b
ON a.order_id = b.order_id
left JOIN
    `EnterpriseBC_01.customers` c
ON a.customer_id = c.customer_id
GROUP BY 1
ORDER BY 3 DESC)
```


(SELECT *, "Higest" AS High FROM T1
ORDER BY 2
limit 5)

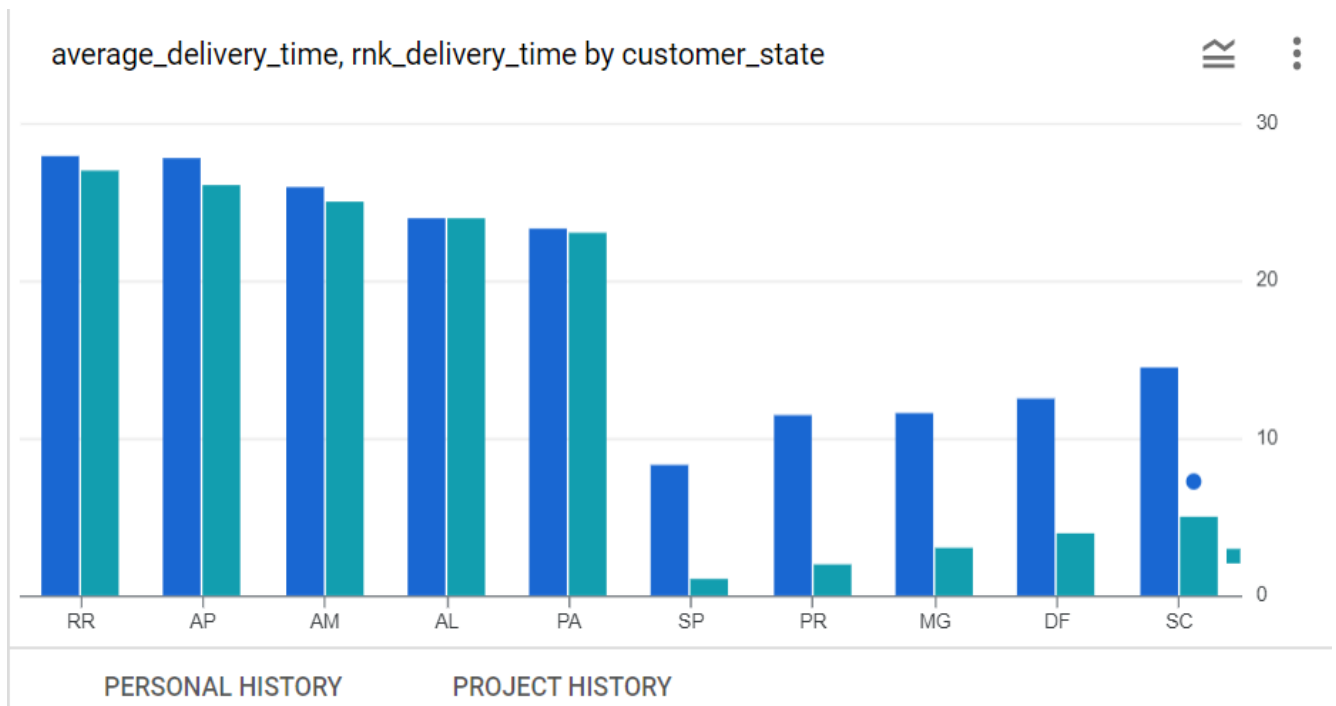
UNION ALL

(SELECT *, "Lowest" AS Low FROM T1
ORDER BY 2 desc
limit 5);

Query results

SAVE RESULTS

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	customer_state	average_delivery_tim	rnk_delivery_time	High			
1	SP	8.26	1	Higest			
2	PR	11.48	2	Higest			
3	MG	11.52	3	Higest			
4	DF	12.5	4	Higest			
5	SC	14.52	5	Higest			
6	RR	27.83	27	Lowest			
7	AP	27.75	26	Lowest			
8	AM	25.96	25	Lowest			
9	AL	23.99	24	Lowest			
10	PA	23.3	23	Lowest			



Insights: The output of the query is displaying top 5 states with the highest & lowest average delivery time.

- 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.

```
WITH T1 AS
(select distinct c.customer_state,
  ((round(avg(date_diff(a.order_delivered_customer_date,
a.order_purchase_timestamp, day)), 2))
-
  (round(avg(date_diff(a.order_delivered_customer_date,
a.order_estimated_delivery_date, day)), 2))) as diff_act_esti,
  dense_rank() over( order by
  ((round(avg(date_diff(a.order_delivered_customer_date, a.order_purchase_timestamp,
day)), 2))
-
  (round(avg(date_diff(a.order_delivered_customer_date,
a.order_estimated_delivery_date, day)), 2)))) as rnk_diff_act_esti
FROM
`EnterpriseBC_01.orders` a
```

```
left JOIN
  `EnterpriseBC_01.order_items` b
ON a.order_id = b.order_id
left JOIN
  `EnterpriseBC_01.customers` c
ON a.customer_id = c.customer_id
GROUP BY 1
ORDER BY 3)

(SELECT *, "Higest" AS High FROM T1
ORDER BY 2
limit 5)

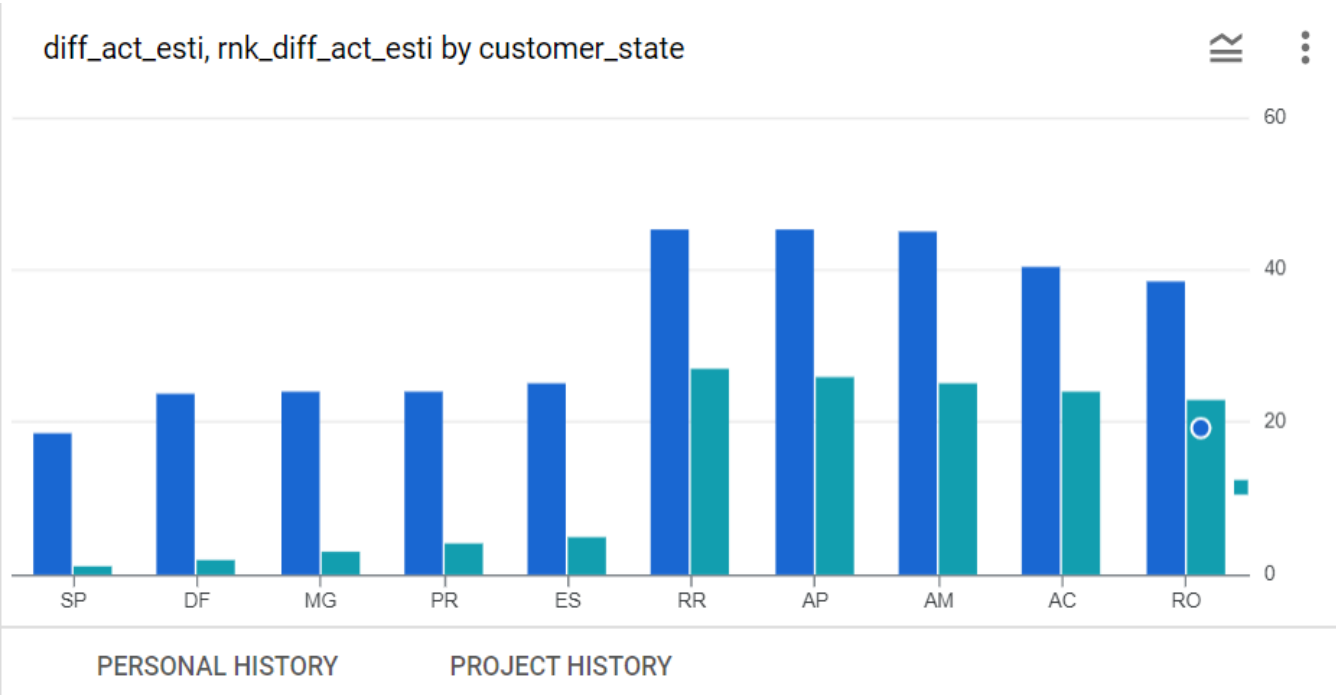
UNION ALL

(SELECT *, "Lowest" AS Low FROM T1
ORDER BY 2 desc
limit 5);
```

Query results

SAVE RESULTS

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	customer_state	diff_act_esti	rnk_diff_act_esti	High			
1	SP	18.53	1	Higest			
2	DF	23.77	2	Higest			
3	MG	23.92	3	Higest			
4	PR	24.009999999999...	4	Higest			
5	ES	24.96	5	Higest			
6	RR	45.26	27	Lowest			
7	AP	45.19	26	Lowest			
8	AM	44.94	25	Lowest			
9	AC	40.34	24	Lowest			
10	RO	38.36	23	Lowest			



Insights: The output of the query is displaying top 5 states where the order delivery is really fast as compared to the estimated date of delivery.

6. Analysis based on the payments:

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

```
select distinct payment_type, format_date('%Y-%m',order_purchase_timestamp)
m_on_m
,count(order_status) as count_ord
from `EnterpriceBC_01.orders` o
join `EnterpriceBC_01.payments` p
on o.order_id = p.order_id
where order_status not in ('unavailable','canceled')
group by 1,2
order by 2;
```

Query results

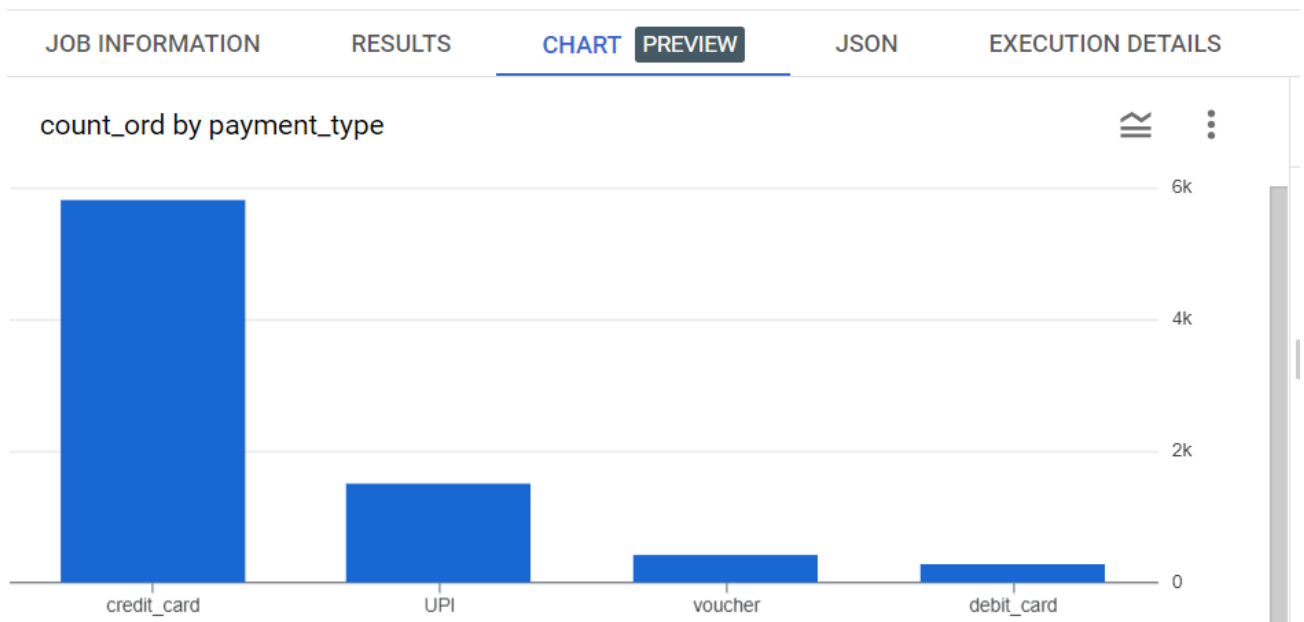
[SAVE RESULTS](#)  E

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	payment_type ▼	m_on_m ▼	count_ord ▼				
1	credit_card	2016-09	1				
2	credit_card	2016-10	227				
3	UPI	2016-10	60				
4	voucher	2016-10	22				
5	debit_card	2016-10	2				
6	credit_card	2016-12	1				
7	credit_card	2017-01	574				
8	UPI	2017-01	193				
9	voucher	2017-01	60				
10	debit_card	2017-01	9				

Results per page: 50 ▼ 1 – 50 of 87

PERSONAL HISTORY

PROJECT HISTORY



Insights: The query is displaying the output month on month no. of orders placed using different payment types.

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

```
select payment_installments, count(order_status) as number_of_orders
from `EnterpriceBC_01.orders` o
join `EnterpriceBC_01.payments` p
on o.order_id = p.order_id
where order_status not in ('unavailable','canceled')
and payment_installments != 0 and payment_installments is not null
group by 1;
```

Query results [SAVE RESULTS](#)

JOB INFORMATION	RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	payment_installment	number_of_orders				
1	1	51814				
2	2	12289				
3	3	10342				
4	4	7016				
5	5	5177				
6	6	3881				
7	7	1611				
8	8	4229				
9	9	630				
10	10	5246				

Results per page: 50 1 – 23 of 23



Insights: The query is displaying the output no. of orders placed on the basis of the payment instalments that have been paid.