

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

A. Get number of rows in the data

- Customers

The screenshot shows a data exploration interface. On the left, the 'Explorer' panel lists a project named 'targetsql-354415' with a 'Target' folder containing several tables: 'customers', 'geolocation', 'order_items', 'order_reviews', 'orders', 'payments', 'products', and 'sellers'. The 'customers' table is selected. The main editor area shows a SQL query: `select count(*) FROM 'targetsql-354415.Target.customers';`. Below the query editor, the 'Query results' section displays a table with one row and one column, 'f0_', containing the value 99441.

Row	f0_
1	99441

- Geolocations

The screenshot shows the same data exploration interface as before, but the 'geolocation' table is selected in the Explorer panel. The SQL query is updated to: `select count(*) FROM 'targetsql-354415.Target.geolocation';`. The 'Query results' section displays a table with one row and one column, 'f0_', containing the value 1000163.

Row	f0_
1	1000163

- Order_items

The screenshot shows the same data exploration interface, but the 'order_items' table is selected in the Explorer panel. The SQL query is updated to: `select count(*) FROM 'targetsql-354415.Target.order_items';`. The 'Query results' section displays a table with one row and one column, 'f0_', containing the value 112650.

Row	f0_
1	112650

- Order_reviews

The screenshot shows a SQL IDE interface. On the left, the 'Explorer' pane displays a tree view of a database named 'targetsql-354415'. Under the 'Target' schema, the 'customers' table is selected. The main editor pane contains a SQL query: `select count(*) FROM 'targetsql-354415.Target.order_reviews'`. Below the query, the 'Query results' section is visible, showing a single row with the count 99224. The interface includes buttons for 'RUN', 'SAVE', 'SHARE', 'SCHEDULE', and 'MORE'.

Row	count(*)
1	99224

- orders

The screenshot shows a SQL IDE interface. On the left, the 'Explorer' pane displays a tree view of a database named 'targetsql-354415'. Under the 'Target' schema, the 'customers' table is selected. The main editor pane contains a SQL query: `select count(*) FROM 'targetsql-354415.Target.orders'`. Below the query, the 'Query results' section is visible, showing a single row with the count 99441. The interface includes buttons for 'RUN', 'SAVE', 'SHARE', 'SCHEDULE', and 'MORE'.

Row	count(*)
1	99441

- payments

The screenshot shows a SQL IDE interface. On the left, the 'Explorer' pane displays a tree view of a database named 'targetsql-354415'. Under the 'Target' schema, the 'customers' table is selected. The main editor pane contains a SQL query: `select count(*) FROM 'targetsql-354415.Target.payments'`. Below the query, the 'Query results' section is visible, showing a single row with the count 103886. The interface includes buttons for 'RUN', 'SAVE', 'SHARE', 'SCHEDULE', and 'MORE'.

Row	count(*)
1	103886

- products

The screenshot shows a SQL IDE interface. On the left, the 'Explorer' pane displays a tree view of a database named 'targetsql-354415'. Under the 'Target' schema, the 'customers' table is selected. The main editor pane contains a SQL query: `select count(*) FROM 'targetsql-354415.Target.products'`. Below the query, the 'Query results' section is visible, showing a single row with the count 32951. The interface includes buttons for 'RUN', 'SAVE', 'SHARE', 'SCHEDULE', and 'MORE'.

Row	count(*)
1	32951

- sellers

The screenshot shows a data explorer interface. On the left, the 'Explorer' pane lists a project named 'targetsql-354415' with a 'Target' folder containing tables: 'customers', 'geolocation', 'order_items', 'order_reviews', 'orders', 'payments', 'products', and 'sellers'. The 'customers' table is selected. The main editor shows a SQL query: `1 select count(*) FROM 'targetsql-354415.Target.sellers'`. The 'Query results' pane shows a single row with the count 3095.

Row	count
1	3095

B. Number of Null or Missing values in columns.

- customers

The screenshot shows the same data explorer interface. The SQL query is: `1 select count(*) FROM 'targetsql-354415.Target.customers'`
`2 WHERE customer_city is NULL`
`3 or customer_id is NULL`
`4 or customer_state is NULL`
`5 or customer_unique_id is Null`
`6 or customer_zip_code.prefix is Null`. The 'Query results' pane shows a single row with the count 0.

Row	count
1	0

- geolocations

The screenshot shows the same data explorer interface. The SQL query is: `1 select count(*) FROM 'targetsql-354415.Target.geolocation'`
`2 WHERE geolocation_city is NULL`
`3 or geolocation_lat is NULL`
`4 or geolocation_lng is NULL`
`5 or geolocation_state is Null`
`6 or geolocation_zip_code.prefix is Null`. The 'Query results' pane shows a single row with the count 0.

Row	count
1	0

- order_items

The screenshot shows the same data explorer interface. The SQL query is: `1 select count(*) FROM 'targetsql-354415.Target.order_items'`
`2 WHERE order_id is NULL`
`3 or order_item_id is NULL`. The 'Query results' pane shows a single row with the count 0.

Row	count
1	0

- order_reviews

The screenshot shows the BigQuery console interface. On the left, the Explorer pane displays a project named 'targetsql-354415' with a 'Target' folder containing tables like 'customers', 'geolocation', 'order_items', 'order_reviews', 'orders', 'payments', 'products', and 'sellers'. The 'order_reviews' table is selected. The main editor shows a SQL query:

```
1 select count(*) FROM `targetsql-354415.Target.order_reviews`
2 WHERE review_id is null
3 or order_id is null
4 or review_score is null
5 or review_comment_title is null
6 or review_creation_date is null
7 or review_answer_timestamp is null
```

 The 'Query results' pane at the bottom shows a single row with 'Row' 1 and 'R0_' 87675. The top bar indicates the query will process 8.86 MB.

- orders

The screenshot shows the BigQuery console interface. The Explorer pane on the left has the 'orders' table selected under the 'Target' folder. The main editor displays a SQL query:

```
1 select count(*) FROM `targetsql-354415.Target.orders`
2 WHERE order_id is null
3 or customer_id is null
4 or order_approved_at is null
5 or order_delivered_carrier_date is null
6 or order_delivered_customer_date is null
7 or order_estimated_delivery_date is null
8 or order_purchase_timestamp is null
9 or order_status is null
10
```

 The 'Query results' pane shows a single row with 'Row' 1 and 'R0_' 2980. The top bar indicates the query will process 11.25 MB.

- payment

The screenshot shows the BigQuery console interface. The Explorer pane on the left has the 'payments' table selected under the 'Target' folder. The main editor displays a SQL query:

```
1 select count(*) FROM `targetsql-354415.Target.payments`
2 WHERE order_id is null
3 or payment_installments is null
4 or payment_sequential is null
5 or payment_type is null
6 or payment_value is null
```

 The 'Query results' pane shows a single row with 'Row' 1 and 'R0_' 0. The top bar indicates the query will process 6.86 MB.

- product

The screenshot shows the BigQuery console interface. The Explorer pane on the left has the 'products' table selected under the 'Target' folder. The main editor displays a SQL query:

```
1 select count(*) FROM `targetsql-354415.Target.products`
2 WHERE product_category is null
3 or product_description_length is null
4 or product_height_cm is null
5 or product_id is null
6 or product_length_cm is null
7 or product_name_length is null
8 or product_photos_qty is null
9 or product_weight_g is null
10 or product_width_cm is null
```

 The 'Query results' pane shows a single row with 'Row' 1 and 'R0_' 611. The top bar indicates the query will process 3.3 MB.

- seller

The screenshot shows a database interface with a sidebar on the left labeled 'Explorer' containing a tree view of projects and tables. The main area displays a SQL query in a text editor:

```
1 select count(*) FROM 'targetsql-354415.Target.sellers'
2 WHERE seller_city is null
3 or seller_id is null
4 or seller_state is null
5 or seller_zip_code_prefix is null
```

Below the query editor, the 'Query results' section is visible, showing a table with one row and one column:

Row	fo_
1	0

C. Datatypes of columns in the table

- customers

The screenshot shows the 'customers' table schema in a database interface. The table has the following columns:

Field name	Type	Mode	Collation	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			

- geolocations

The screenshot shows the 'geolocation' table schema in a database interface. The table has the following columns:

Field name	Type	Mode	Collation	Policy Tags	Description
geolocation_zip_code_prefix	INTEGER	NULLABLE			
geolocation_lat	FLOAT	NULLABLE			
geolocation_lng	FLOAT	NULLABLE			
geolocation_city	STRING	NULLABLE			
geolocation_state	STRING	NULLABLE			

- order_items

The screenshot shows the 'order_items' table schema in a database interface. The table has the following columns:

Field name	Type	Mode	Collation	Policy Tags	Description
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

Explorer + ADD DATA <

Type to search

Viewing pinned projects.

targetsql-354415

Target

- customers
- geolocation
- order_items
- order_reviews
- orders
- payments
- products
- sellers

order_reviews

QUERY SHARE COPY SNAPSHOT DELETE EXPORT

SCHEMA DETAILS PREVIEW

Filter Enter property name or value

Field name	Type	Mode	Collation	Policy Tags	Description
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			

EDIT SCHEMA VIEW ROW ACCESS POLICIES

- orders

Explorer + ADD DATA <

Type to search

Viewing pinned projects.

targetsql-354415

Target

- customers
- geolocation
- order_items
- order_reviews
- orders
- payments
- products
- sellers

orders

QUERY SHARE COPY SNAPSHOT DELETE EXPORT

SCHEMA DETAILS PREVIEW

Filter Enter property name or value

Field name	Type	Mode	Collation	Policy Tags	Description
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			

EDIT SCHEMA VIEW ROW ACCESS POLICIES

- payment

Explorer + ADD DATA <

Type to search

Viewing pinned projects.

targetsql-354415

Target

- customers
- geolocation
- order_items
- order_reviews
- orders
- payments
- products
- sellers

payments

QUERY SHARE COPY SNAPSHOT DELETE EXPORT

SCHEMA DETAILS PREVIEW

Filter Enter property name or value

Field name	Type	Mode	Collation	Policy Tags	Description
order_id	STRING	NULLABLE			
payment_sequential	INTEGER	NULLABLE			
payment_type	STRING	NULLABLE			
payment_installments	INTEGER	NULLABLE			
payment_value	FLOAT	NULLABLE			

EDIT SCHEMA VIEW ROW ACCESS POLICIES

- product

Explorer + ADD DATA <

Type to search

Viewing pinned projects.

targetsql-354415

Target

- customers
- geolocation
- order_items
- order_reviews
- orders
- payments
- products
- sellers

products

QUERY SHARE COPY SNAPSHOT DELETE EXPORT

SCHEMA DETAILS PREVIEW

Filter Enter property name or value

Field name	Type	Mode	Collation	Policy Tags	Description
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			

EDIT SCHEMA VIEW ROW ACCESS POLICIES

- seller

The screenshot shows the Google Cloud BigQuery Explorer interface. On the left, the 'Explorers' pane shows a project named 'targetsql-354415' with a table named 'sellers' selected. The main pane displays the 'sellers' table schema with the following fields:

Field name	Type	Mode	Collation	Policy Tags	Description
seller_id	STRING	NULLABLE			
seller_zip_code_prefix	INTEGER	NULLABLE			
seller_city	STRING	NULLABLE			
seller_state	STRING	NULLABLE			

Buttons for 'EDIT SCHEMA' and 'VIEW ROW ACCESS POLICIES' are visible at the bottom of the schema view.

D. Get the time period for the which the data is given

The screenshot shows the Google Cloud BigQuery Explorer interface. The 'Explorers' pane shows the 'order_items' table selected. The main pane displays a query to find the time period of the data:

```
1 select min(shipping_limit_date) AS First_date,max(shipping_limit_date) AS Last_date
2 FROM `targetsql-354415.Target.order_items`
```

The query results are shown below:

Row	First_date	Last_date
1	2016-09-19 00:15:34 UTC	2020-04-09 22:35:08 UTC

E. Number of cities in our dataset

The screenshot shows the Google Cloud BigQuery Explorer interface. The 'Explorers' pane shows the 'customers' table selected. The main pane displays a query to count the number of cities:

```
1 Select count(distinct(customer_city)) from `targetsql-354415.Target.customers`
```

The query results are shown below:

Row	fo_
1	4119

F. Number of states in out dataset

The screenshot shows a data explorer interface. On the left, a tree view lists tables under 'targetsql-354415', including 'customers', 'geolocation', 'order_items', 'order_reviews', 'orders', 'payments', 'products', and 'sellers'. The 'customers' table is selected. The main query editor shows the following SQL query:

```
1 Select count(distinct(customer_state)) from 'targetsql-354415.Target.customers'
```

The 'Query results' section displays the following data:

Row	f0_
1	27

2. In-Depth Exploration

A. How many order do we have for each order status?

The screenshot shows a data explorer interface. On the left, the 'orders' table is selected. The main query editor shows the following SQL query:

```
1 SELECT order_status, count(order_status) FROM 'targetsql-354415.Target.orders'
2 GROUP BY order_status
```

The 'Query results' section displays the following data:

Row	order_status	f0_
1	created	5
2	shipped	1107
3	approved	2
4	canceled	625
5	invoiced	314
6	delivered	96478
7	processing	301
8	unavailable	609

B. Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario?

Code :-

```
SELECT OrderYear, OrderMonth, Count(*) TotalOrdersInMonth
FROM
```

```
(SELECT EXTRACT(YEAR from order_purchase_timestamp) OrderYear, EXTRACT(MONTH from order_purchase_timestamp) OrderMonth
```

```
FROM `targetsql-354415.Target.orders`)
SUB
GROUP BY OrderYear, OrderMonth
ORDER BY OrderYear, OrderMonth
```

Row	OrderYear	OrderMonth	TotalOrdersInMonth
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800
5	2017	2	1780
6	2017	3	2682
7	2017	4	2404
8	2017	5	3700
9	2017	6	3245

Row	OrderYear	OrderMonth	TotalOrdersInMonth	
10	2017	7	4026	
11	2017	8	4331	
12	2017	9	4285	
13	2017	10	4631	
14	2017	11	7544	
15	2017	12	5673	
16	2018	1	7269	
17	2018	2	6728	
18	2018	3	7211	

Row	OrderYear	OrderMonth	TotalOrdersInMonth
17	2018	2	6728
18	2018	3	7211
19	2018	4	6939
20	2018	5	6873
21	2018	6	6167
22	2018	7	6292
23	2018	8	6512
24	2018	9	16
25	2018	10	4

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

Explorer
+ ADD DATA
<

Viewing pinned projects.

- targetsql-354415
 - Target
 - customers
 - geolocation
 - order_items
 - order_reviews
 - orders**
 - payments
 - products
 - sellers

*Editor > x orders > x *Unsaved ...y 2 > x Editor 3 > x Q *Unsaved ...y 4 > x

RUN SAVE SHARE SCHEDULE MORE

✓ This query will process 3.98 MB when run.

```

1 SELECT
2 case
3 when extract(hour from order_purchase_timestamp) between 0 and 6 then 'dawn'
4 when extract(hour from order_purchase_timestamp) between 7 and 12 then 'morning'
5 when extract(hour from order_purchase_timestamp) between 13 and 18 then 'afternoon'
6 when extract(hour from order_purchase_timestamp) between 19 and 23 then 'night'
7 end as time_of_day,count(distinct(order_id))
8 FROM `targetsql-354415.Target.orders`
9 group by 1
10 order by 2
        
```

Press Alt+F1 for Accessibility Options.

Query results

SAVE RESULTS EXPLORE DATA

JOB INFORMATION	RESULTS	JSON	EXECUTION DETAILS
Row	time_of_day	f0_	
1	dawn	5242	
2	morning	27733	
3	night	28331	
4	afternoon	38135	

D. Feature Extraction: Through order_purchase_timestamp in "orders" dataset extract

A. order_purchase_year

Explorer

+ ADD DATA

⏮

🔍 Type to search

?

Viewing pinned projects.

▼ targetsql-354415

▼ Target

customers

geolocation

order_items

order_reviews

orders

payments

products

sellers

*Editor

✕

🔍 *Unsaved ...y 2

✕

📄 sellers

✕

+

ⓘ

📄

🔍

🔗 DISABLE EDITOR TABS

▶ RUN

💾 SAVE

👤 SHARE

🕒 SCHEDULE

⚙️ MORE

✅ This query will process 776.88 KB when run.

```

1 SELECT DISTINCT(EXTRACT(YEAR FROM order_purchase_timestamp)) as Year
2 FROM `targetsql-354415.Target.orders`

```

Press Alt+F1 for Accessibility Options

Query results

📄 SAVE RESULTS

📊 EXPLORE DATA

↕

JOB INFORMATION

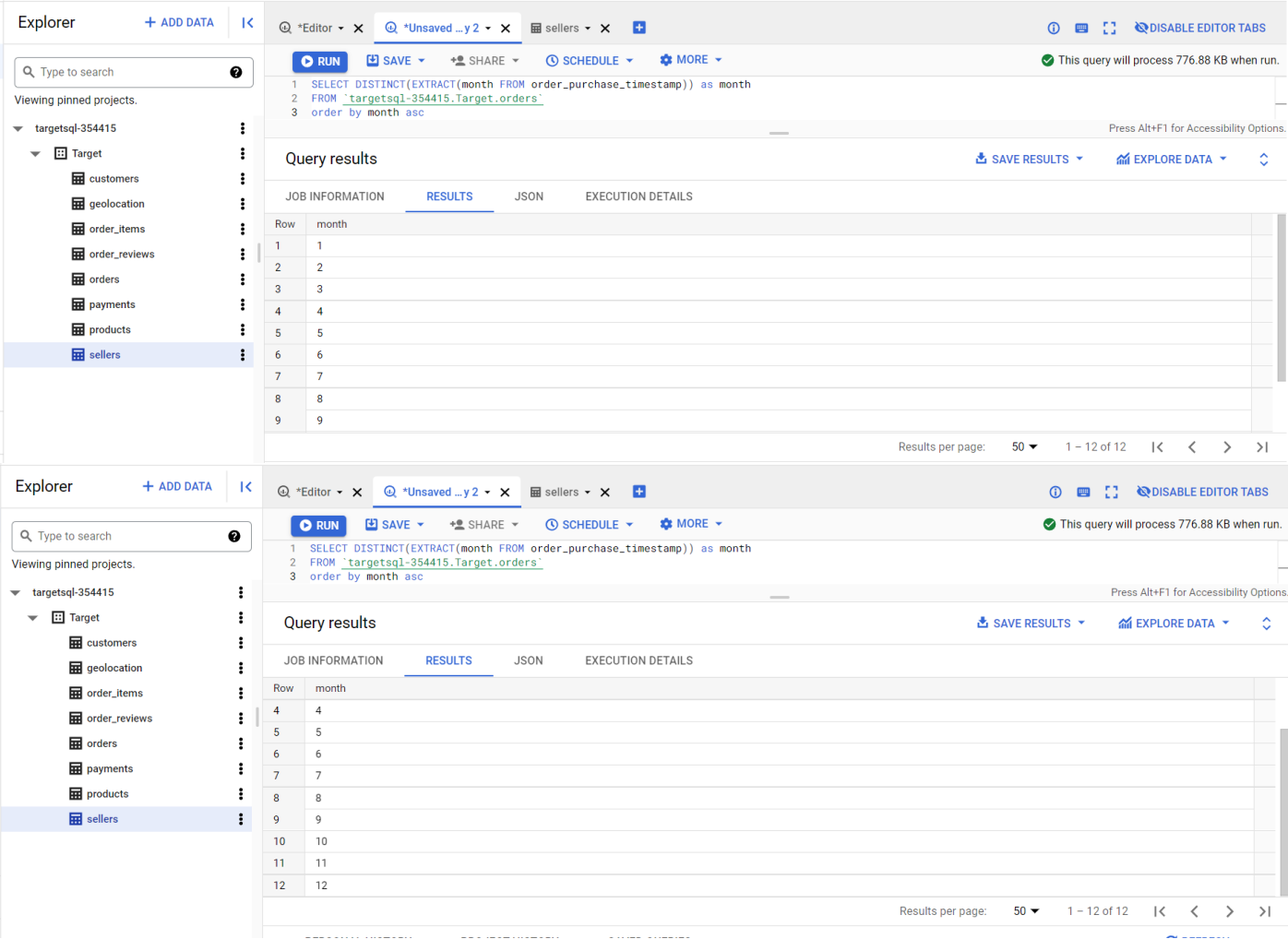
RESULTS

JSON

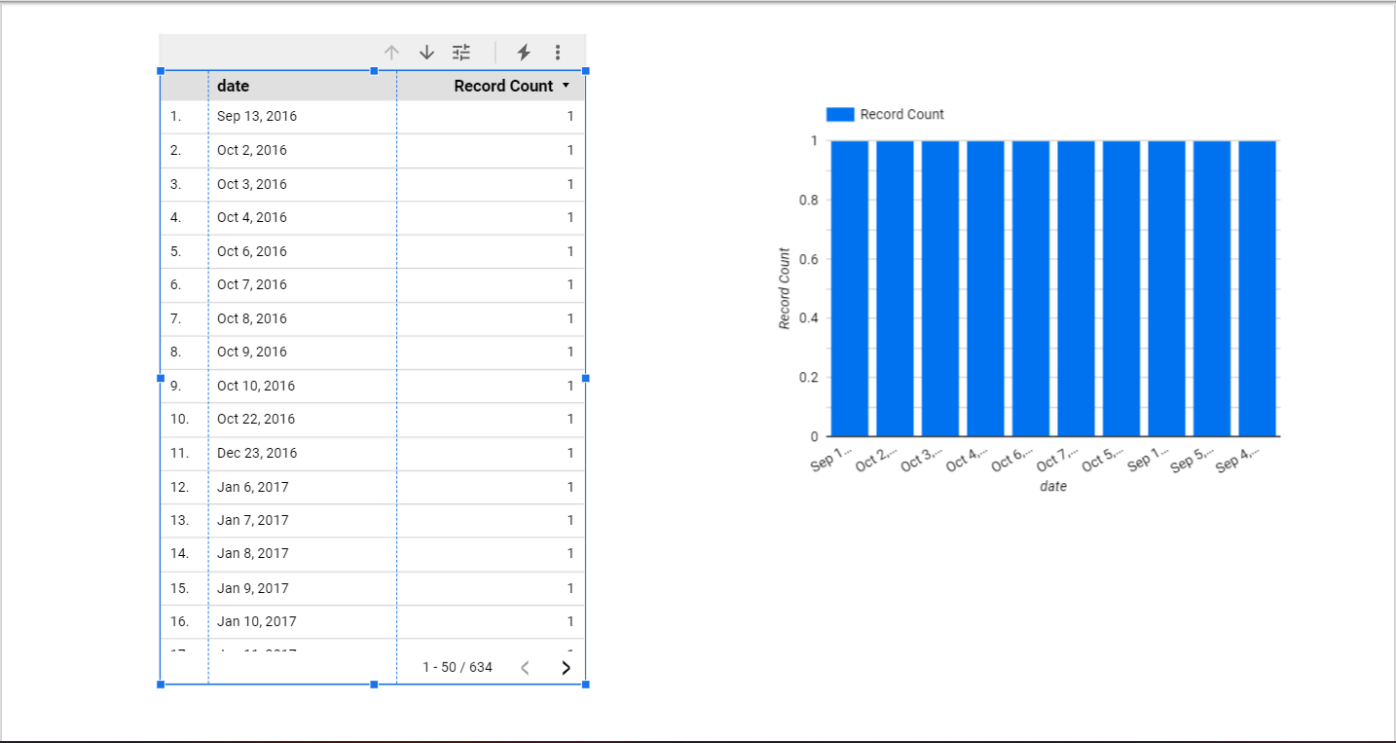
EXECUTION DETAILS

Row	Year
1	2017
2	2018
3	2016

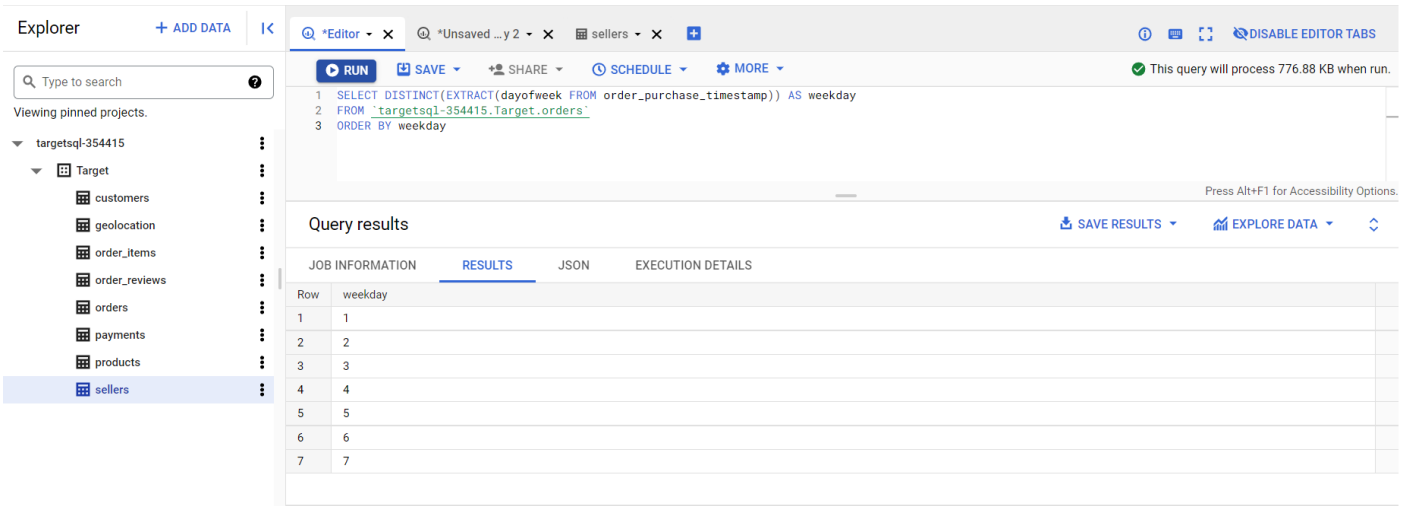
B. order_purchase_month



C. order_purchase_date



D. order_purchase_day



E. order_purchase_hour



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

1. Get month on month orders by region

2. Total of customer orders by state

Explorer

+ ADD DATA

IK

Type to search

?

Viewing pinned projects.

targetsql-354415

Target

customers

geolocation

order_items

order_reviews

orders

payments

products

sellers

orders

customers

*Unsaved ...y 3

Editor 4

+

DISABLE EDITOR TABS

RUN

SAVE

SHARE

SCHEDULE

MORE

This query will process 10.05 MB when run.

```
1 SELECT c.customer_state,COUNT(DISTINCT(o.order_id)) AS orders
2 FROM `targetsql-354415.Target.customers` c
3 INNER JOIN `targetsql-354415.Target.orders` o
4 ON o.customer_id=c.customer_id
5 group by c.customer_state
```

Query results

SAVE RESULTS

EXPLORE DATA

JOB INFORMATION

RESULTS

JSON

EXECUTION DETAILS

Row	customer_state	orders
1	RN	485
2	CE	1336
3	RS	5466
4	SC	3637
5	SP	41746
6	MG	11635
7	BA	3380
8	RJ	12852

Results per page: 50 1 - 27 of 27 |< < > >|

Explorer

+ ADD DATA

IK

Type to search

?

Viewing pinned projects.

targetsql-354415

Target

customers

geolocation

order_items

order_reviews

orders

payments

products

sellers

orders

customers

*Unsaved ...y 3

Editor 4

+

DISABLE EDITOR TABS

RUN

SAVE

SHARE

SCHEDULE

MORE

This query will process 10.05 MB when run.

```
1 SELECT c.customer_state,COUNT(DISTINCT(o.order_id)) AS orders
2 FROM `targetsql-354415.Target.customers` c
3 INNER JOIN `targetsql-354415.Target.orders` o
4 ON o.customer_id=c.customer_id
5 group by c.customer_state
```

Query results

SAVE RESULTS

EXPLORE DATA

JOB INFORMATION

RESULTS

JSON

EXECUTION DETAILS

Row	customer_state	orders
9	GO	2020
10	MA	747
11	PE	1652
12	PB	536
13	ES	2033
14	PR	5045
15	RO	253
16	MS	715

Results per page: 50 1 - 27 of 27 |< < > >|

Explorer

+ ADD DATA

IK

Type to search

?

Viewing pinned projects.

targetsql-354415

Target

customers

geolocation

order_items

order_reviews

orders

payments

products

sellers

orders

customers

*Unsaved ...y 3

Editor 4

+

DISABLE EDITOR TABS

RUN

SAVE

SHARE

SCHEDULE

MORE

This query will process 10.05 MB when run.

```
1 SELECT c.customer_state,COUNT(DISTINCT(o.order_id)) AS orders
2 FROM `targetsql-354415.Target.customers` c
3 INNER JOIN `targetsql-354415.Target.orders` o
4 ON o.customer_id=c.customer_id
5 group by c.customer_state
```

Query results

SAVE RESULTS

EXPLORE DATA

JOB INFORMATION

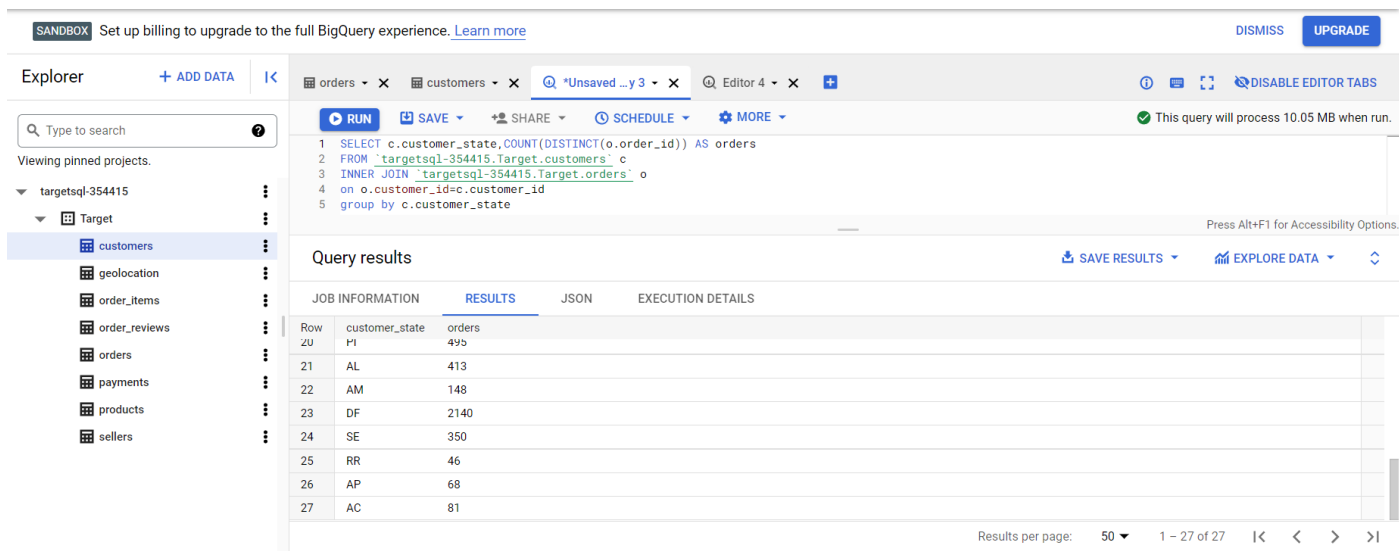
RESULTS

JSON

EXECUTION DETAILS

Row	customer_state	orders
17	PA	975
18	TO	280
19	MT	907
20	PI	495
21	AL	413
22	AM	148
23	DF	2140
24	SE	350

Results per page: 50 1 - 27 of 27 |< < > >|



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

Step 1: Using CTE

1. Group data by year and month, aggregation count(order_id), sum(price), sum(freight_value)

count(order_id)

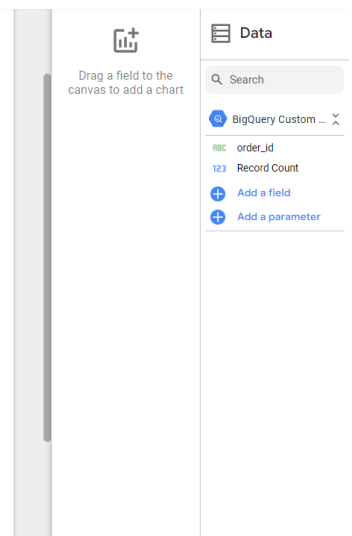
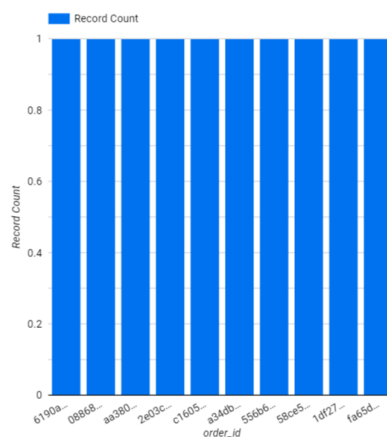
```

select orders.order_id
FROM Target.orders
INNER JOIN Target.order_items
ON orders.order_id = order_items.order_id
GROUP BY orders.order_id
order by count(orders.order_id);

```

	order_id	Record Count
1.	6190a94657e1012983a274b831d...	1
2.	088683f795a3d30bfd61152c4fab...	1
3.	aa380313c19905dd1651bd21e75...	1
4.	2e03cb2541b48c78aebca2dbfb2f...	1
5.	c160599d4ea4eefa0e420db0a0cc...	1
6.	a34db9935aad747acfecadbba0bf...	1
7.	2d1cfcc5ed3232215b908e86ff33...	1
8.	7b9906348a4044ff73ce3034e9c8...	1
9.	f247d8bea2a3d9e88b688d08f946...	1
10.	cb7911a7f5f016586dcdf66a52c1...	1
11.	a0123fbd74e1c68d8f3f46d14a07...	1
12.	709cb0731456cbfb2ca8d299b66...	1
13.	3385c99ff53af3e0f1115d79f6165...	1
14.	52cb9b4d5ee3ce7d1e2a8d9c237...	1
15.	36e16c2da456685fc8c12da624...	1
16.	0927a99c4c0b6cb24874a4e869f...	1
17.	4a010d4a1183a019a0f01697b...	1

1 - 50 / 98666 < >

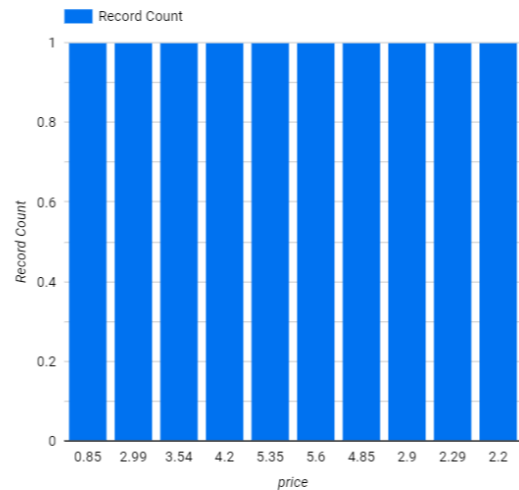


Sum(price)

```
select order_items.price
FROM Target.orders
INNER JOIN Target.order_items
ON orders.order_id =order_items.order_id
GROUP BY order_items.price
order by sum(order_items.price);
```

	price	Record Count ▾
3.	3.54	1
4.	4.2	1
5.	5.35	1
6.	5.6	1
7.	5.7	1
8.	5.73	1
9.	5.95	1
10.	3	1
11.	6.27	1
12.	6.37	1
13.	7.09	1
14.	7.12	1
15.	7.48	1
16.	7.49	1

1 - 50 / 5968 < >

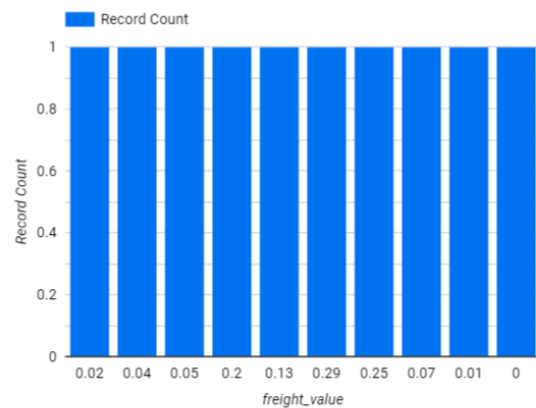


sum(freight_value)

```
select order_items.freight_value
FROM Target.orders
INNER JOIN Target.order_items
ON orders.order_id =order_items.order_id
GROUP BY order_items.freight_value
order by sum(order_items.freight_value);
```

	freight_value	Record Count ▾
1.	0.02	1
2.	0.04	1
3.	0.05	1
4.	0.2	1
5.	0.13	1
6.	0.29	1
7.	0.1	1
8.	0.31	1
9.	0.11	1
10.	0.34	1
11.	0.12	1

1 - 50 / 6999 < >



3. Create new columns:

$\text{price_per_order} = \text{sum}(\text{price}) / \text{count}(\text{order_id})$

The screenshot shows a SQL query editor with the following query:

```
1 SELECT SUM(price)/COUNT(order_id) as Price_per_order
2 FROM `targetsql-354415.Target.order_items`
```

The query results are displayed in a table:

Row	Price_per_order
1	120.65373901482961

$\text{freight_per_order} = \text{sum}(\text{freight_value}) / \text{count}(\text{order_id})$

The screenshot shows a SQL query editor with the following query:

```
1 SELECT SUM(freight_value)/COUNT(order_id) as freight_per_order
2 FROM `targetsql-354415.Target.order_items`
```

The query results are displayed in a table:

Row	freight_per_order
1	19.990319928986111

Step 2: Answer the following questions:

1. Total amount sold in 2017 between Jan to August

The screenshot shows a SQL query editor with the following query:

```
1 SELECT COUNT(order_id) AS AmountSold,
2 FROM `targetsql-354415.Target.orders`
3 WHERE order_purchase_timestamp between '2017-01-01' AND '2017-08-01'
```

The query results are displayed in a table:

Row	AmountSold
1	18637

2. Total amount sold in 2018 between Jan to august

The screenshot shows a data platform interface with an Explorer on the left and a query editor on the right. The Explorer lists a project named 'targetsql-354415' with a 'Target' folder containing tables like 'customers', 'geolocation', 'order_items', 'order_reviews', 'orders', 'payments', 'products', and 'sellers'. The 'orders' table is selected. The query editor shows a SQL query to calculate the total amount sold in 2018 between January and August. The query results are displayed in a table with one row and one column.

```
1 SELECT COUNT(order_id) AS AmountSold,
2 FROM `targetsql-354415.Target.orders`
3 WHERE order_purchase_timestamp between '2018-01-01' AND '2018-08-01'
```

Row	AmountSold
1	47479

3. % increase from 2017 to 2018

The screenshot shows the same data platform interface. The Explorer now shows the 'customers' table selected. The query editor contains a complex SQL query to calculate the percentage increase in orders from 2017 to 2018. The query results are displayed in a table with three rows and six columns.

```
1 select *, (orders-coalesce(lagger_orders,0))/coalesce(orders,1)*100 as difference from ( select *, lag (orders,1) over (order by year asc) as
2 lagger_orders from (
3 select extract(year from a.order_purchase_timestamp) as year,
4 count(distinct a.order_id) as orders,
5 count(distinct b.customer_unique_id) as customers
6 from `targetsql-354415.Target.orders` a
7 left join `targetsql-354415.Target.customers` b
8 on a.customer_id=b.customer_id
9 group by 1
10 )base) base_2
11 order by year asc
```

Row	year	orders	customers	lagger_orders	difference
1	2016	329	326	null	100.0
2	2017	45101	43713	329	99.270526152413481
3	2018	54011	52749	45101	16.496639573420229

Analysis on sales, freight and delivery time

A. Calculating days between purchasing, delivering and estimated delivery

B. Create columns:

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

The screenshot shows the data platform interface. The Explorer lists the 'orders' table. The query editor shows a SQL query to calculate the time to delivery. The query results are displayed in a table with one row and one column.

```
1 SELECT (EXTRACT(day FROM order_purchase_timestamp)-EXTRACT(day FROM order_estimated_delivery_date)) AS Time_To_Delivery
2 FROM `Target.orders`
```

Row	Time_To_Delivery
8	22
9	23
10	21
11	27
12	21
13	21

**diff_estimated_delivery = order_estimated_delivery_date -
order_delivered_customer_date**

Explorer + ADD DATA | K

Q Type to search

Viewing pinned projects.

- targetsql-354415
 - Target
 - customers
 - geolocation
 - order_items
 - order_reviews
 - orders**
 - payments
 - products
 - sellers

orders x Q *Unsaved ...y 2 x

RUN MORE SAVE SHARE SCHEDULE

✓ This query will process 1.49 MB when run.

```
1 SELECT (EXTRACT(day FROM order_delivered_customer_date)-EXTRACT(day FROM order_estimated_delivery_date)) AS Diff_estimated_delivery
2 FROM `Target.orders`
```

Press Alt+F1 for Accessibility Options.

Query results SAVE RESULTS EXPLORE DATA

JOB INFORMATION RESULTS JSON EXECUTION DETAILS

Row	Diff_estimated_delivery
1	26
2	26
3	26
4	27
5	30
6	29

Results per page: 50 1 - 50 of 99441 |< < > >|

6. Payment type analysis: Join “payments” dataset with the existing data on order_id

a. Count of orders for different payment types

Explorer + ADD DATA | K

Q Type to search

Viewing pinned projects.

- targetsql-354415
 - Target
 - customers
 - geolocation
 - order_items
 - order_reviews
 - orders**
 - payments
 - products
 - sellers

orders x payments x Q *Unsaved ...y 3 x

RUN MORE SAVE SHARE SCHEDULE

✓ This query will process 1.11 MB when run.

```
1 SELECT payment_type, count(payment_type) AS CountOfOrders
2 FROM `targetsql-354415.Target.payments`
3 GROUP BY payment_type
```

Press Alt+F1 for Accessibility Options.

Query results SAVE RESULTS EXPLORE DATA

JOB INFORMATION RESULTS JSON EXECUTION DETAILS

Row	payment_type	CountOfOrders
1	credit_card	76795
2	voucher	5775
3	not_defined	3
4	debit_card	1529
5	UPI	19784