

TARGET BRAZIL E-COMMERCE ANALYSIS

1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:
 - 1.1. Data type of all columns in the "customers" table.

The screenshot shows the BigQuery schema view for the 'customers' table. At the top, there are navigation links: 'chrome-entropy-386815 / Datasets / Target / Tables / customers'. Below this are buttons for 'customers' (star icon), 'Query' (magnifying glass icon), 'Open in' (dropdown arrow), 'Share' (person icon), and 'Copy' (copy icon). A horizontal menu bar includes 'Schema' (selected), 'Details', 'Preview', 'Table explorer', 'Preview' (which is bolded), and 'Insights'. A 'Filter' input field is present. The main area displays a table of columns:

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

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

The screenshot shows the BigQuery query editor. The query is:

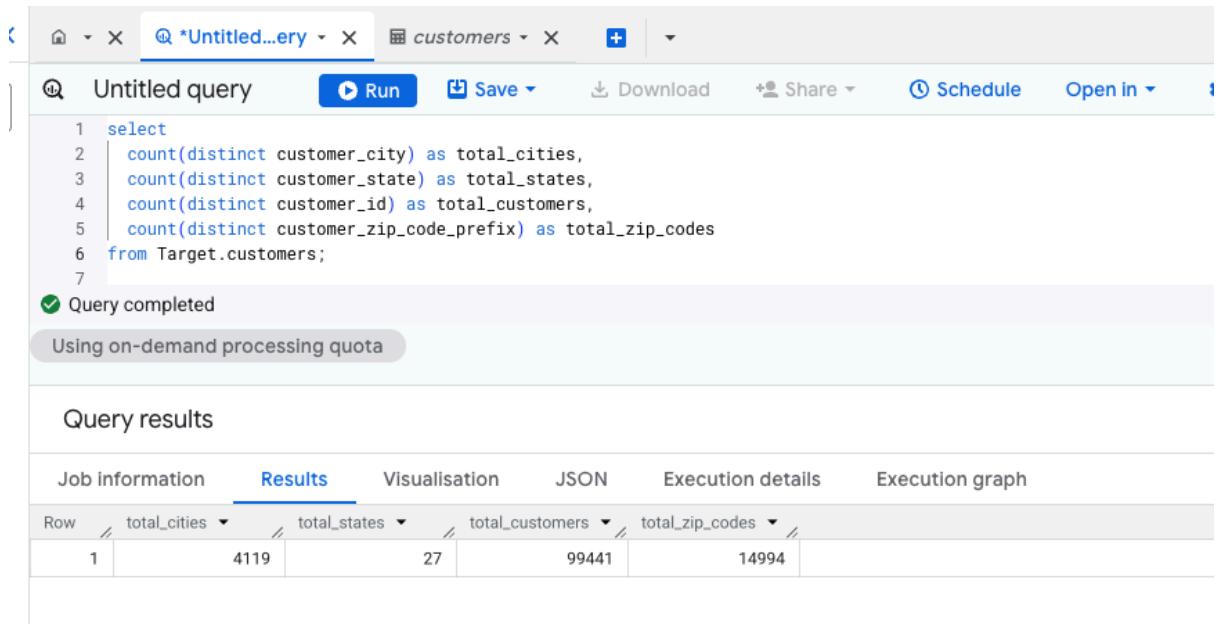
```
1 select
2   min(order_purchase_timestamp) as earliest_order,
3   max(order_purchase_timestamp) as latest_order,
4   timestamp_diff(max(order_purchase_timestamp), min(order_purchase_timestamp), day) as days_between,
5   extract(year from min(order_purchase_timestamp)) as start_year,
6   extract(year from max(order_purchase_timestamp)) as end_year
7 from Target.orders;
8
```

The status bar indicates 'Query completed' and 'Using on-demand processing quota'. Below the editor is a 'Query results' section with tabs for 'Job information', 'Results' (selected), 'Visualisation', 'JSON', 'Execution details', and 'Execution graph'. The results table has columns: Row, earliest_order, latest_order, days_between, start_year, and end_year. One row is shown:

Row	earliest_order	latest_order	days_between	start_year	end_year
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC	772	2016	2018

- Orders span approx. 3 years (2016-18) of data

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



The screenshot shows a data processing interface with the following details:

- Query Editor:** The title bar says "Untitled query". The code is:

```
1 select
2   count(distinct customer_city) as total_cities,
3   count(distinct customer_state) as total_states,
4   count(distinct customer_id) as total_customers,
5   count(distinct customer_zip_code_prefix) as total_zip_codes
6 from Target.customers;
7
```

A green checkmark indicates "Query completed". A message "Using on-demand processing quota" is shown.
- Query Results:** The results are displayed in a table with the following columns: Row, total_cities, total_states, total_customers, and total_zip_codes. The data is:

Row	total_cities	total_states	total_customers	total_zip_codes
1	4119	27	99441	14994

- Market coverage spans 27 states and 4119 cities across Brazil

2. In-depth Exploration:

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

The screenshot shows a data exploration interface with the following components:

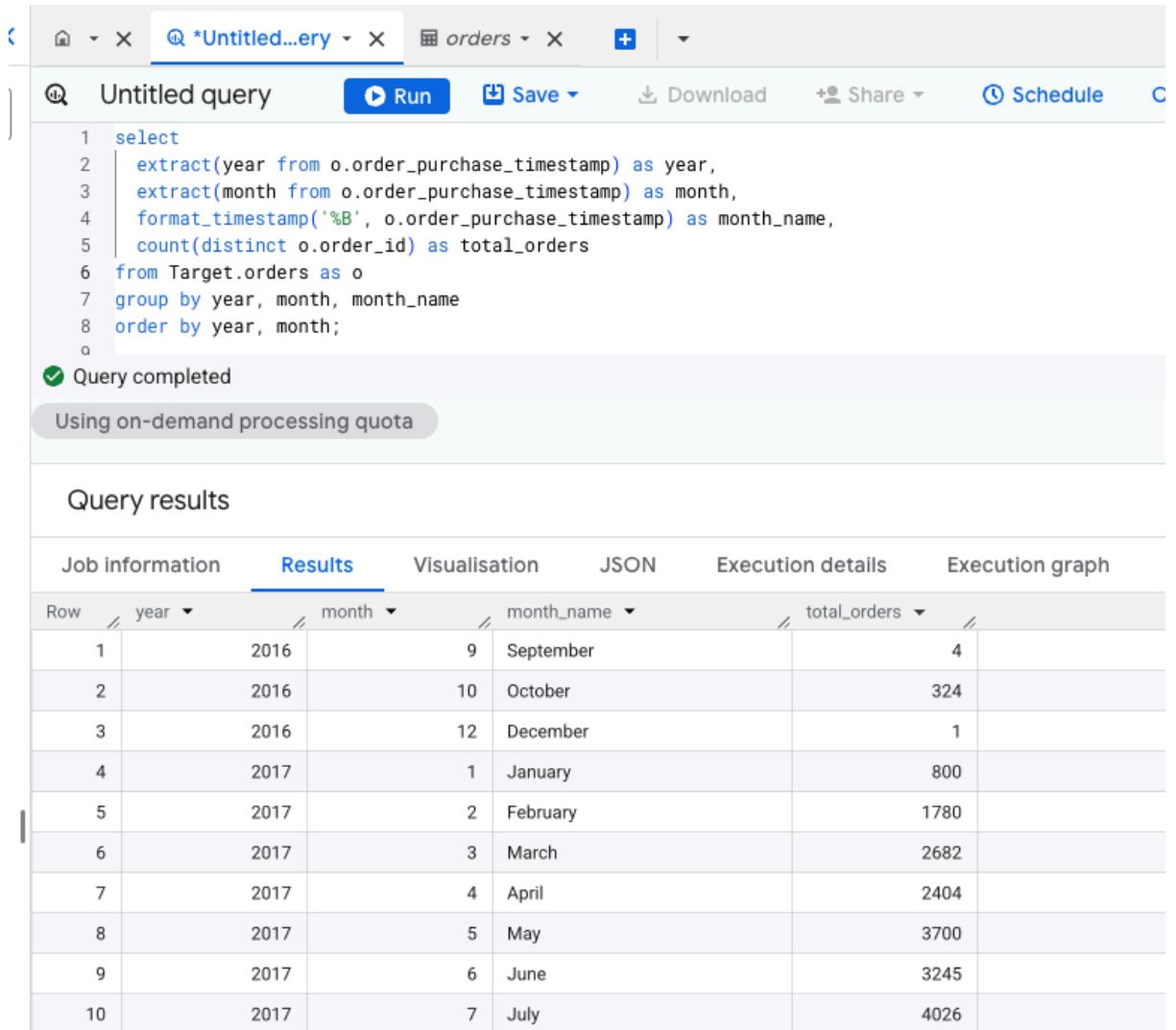
- Query Editor:** A code editor titled "Untitled query" containing the following SQL-like query:

```
1 with yearly_orders as (
2     select
3         extract(year from order_purchase_timestamp) as year,
4         count(distinct order_id) as total_orders
5     from Target.orders
6     group by year
7 )
8
9 select
10    year,
11    total_orders,
12    round(
13        (total_orders - lag(total_orders) over (order by year))
14        / lag(total_orders) over (order by year) * 100
15    , 2) as yoy_growth_pct
16 from yearly_orders
17 order by year;
```
- Status Bar:** A message "Query completed" and a note "Using on-demand processing quota".
- Results View:** A table titled "Query results" with tabs for "Job information", "Results", "Visualisation", "JSON", and "Execution details". The "Results" tab is selected, showing the following data:

Row	year	total_orders	yoy_growth_pct
1	2016	329	null
2	2017	45101	13608.51
3	2018	54011	19.76

- 2016 has very few orders (329)
- 2017 is the first full year of scale: orders explode from 329 to 45,101
- 2018 keeps growing from 45,101 to 54,011, about 20% growth, showing solid but more normal expansion.

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



The screenshot shows a data analysis interface with a query editor and a results table.

Query Editor:

```

1 select
2   extract(year from o.order_purchase_timestamp) as year,
3   extract(month from o.order_purchase_timestamp) as month,
4   format_timestamp('%B', o.order_purchase_timestamp) as month_name,
5   count(distinct o.order_id) as total_orders
6 from Target.orders as o
7 group by year, month, month_name
8 order by year, month;
9

```

Message: Query completed

Using on-demand processing quota

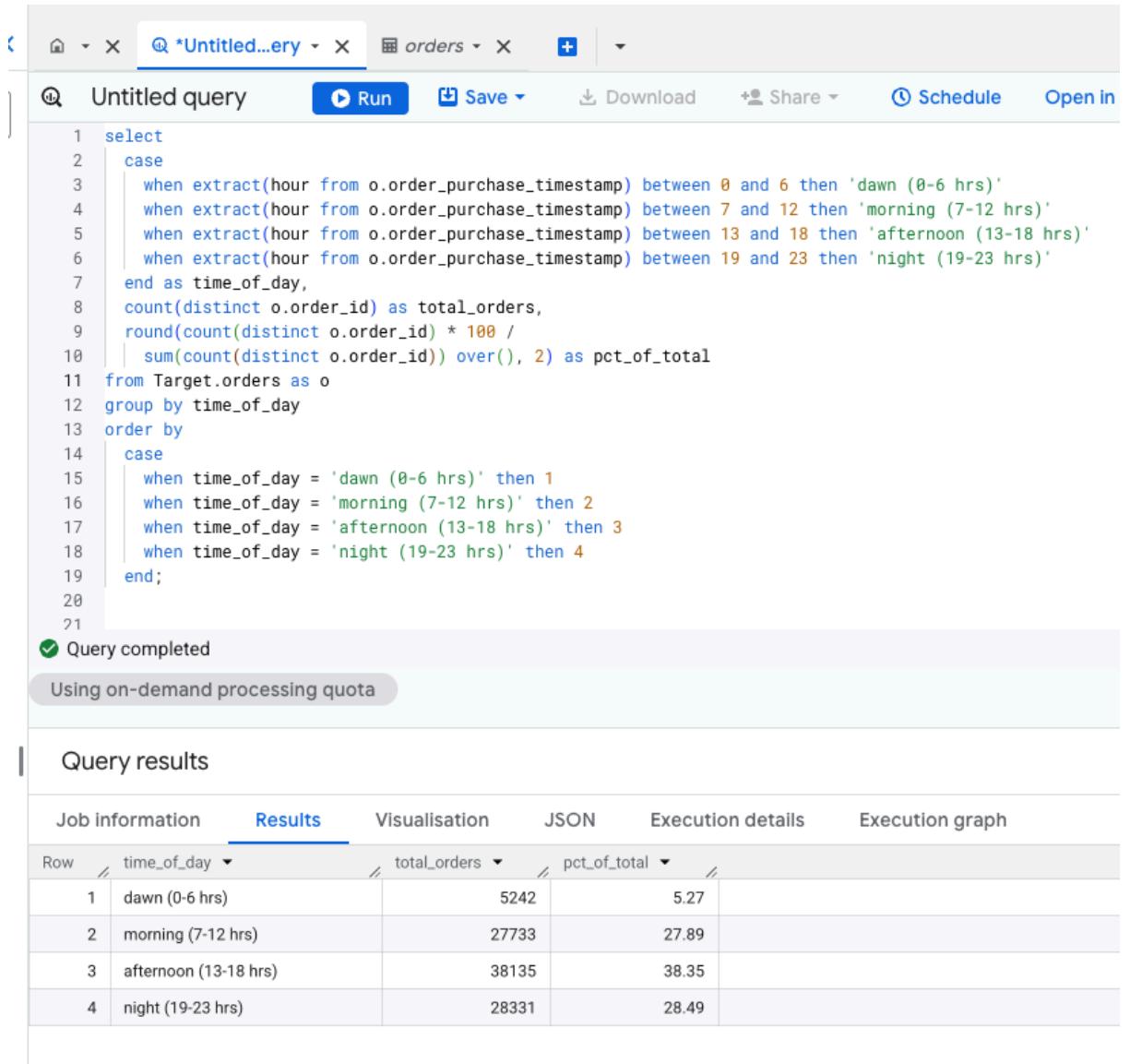
Query results:

Job information	Results	Visualisation	JSON	Execution details	Execution graph
Row	year	month	month_name	total_orders	
1	2016	9	September	4	
2	2016	10	October	324	
3	2016	12	December	1	
4	2017	1	January	800	
5	2017	2	February	1780	
6	2017	3	March	2682	
7	2017	4	April	2404	
8	2017	5	May	3700	
9	2017	6	June	3245	
10	2017	7	July	4026	

- Orders only start from Sept 2016, so 2016 is a partial
- In 2017, volumes steadily climbed from January and peaked in November, with Oct-Dec all very strong, indicating a pronounced year end/ holiday season spike.
- In 2018, monthly orders stayed high from Jan through August, reinforcing that the business has matured with consistently strong demand.

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



The screenshot shows a data processing interface with a query editor and a results table.

Query Editor:

```

1 select
2   case
3     when extract(hour from o.order_purchase_timestamp) between 0 and 6 then 'dawn (0-6 hrs)'
4     when extract(hour from o.order_purchase_timestamp) between 7 and 12 then 'morning (7-12 hrs)'
5     when extract(hour from o.order_purchase_timestamp) between 13 and 18 then 'afternoon (13-18 hrs)'
6     when extract(hour from o.order_purchase_timestamp) between 19 and 23 then 'night (19-23 hrs)'
7   end as time_of_day,
8   count(distinct o.order_id) as total_orders,
9   round(count(distinct o.order_id) * 100 /
10    sum(count(distinct o.order_id)) over(), 2) as pct_of_total
11  from Target.orders as o
12 group by time_of_day
13 order by
14   case
15     when time_of_day = 'dawn (0-6 hrs)' then 1
16     when time_of_day = 'morning (7-12 hrs)' then 2
17     when time_of_day = 'afternoon (13-18 hrs)' then 3
18     when time_of_day = 'night (19-23 hrs)' then 4
19   end;
20
21

```

Message Bar:

Query completed
Using on-demand processing quota

Query results:

Job information	Results	Visualisation	JSON	Execution details	Execution graph
Row	time_of_day	total_orders	pct_of_total		
1	dawn (0-6 hrs)	5242	5.27		
2	morning (7-12 hrs)	27733	27.89		
3	afternoon (13-18 hrs)	38135	38.35		
4	night (19-23 hrs)	28331	28.49		

- Order activity is heavily concentrated in afternoons (13-18h), which account for about 38% of all orders.
- Evenings/ nights (19-23h) and mornings (7-12h) are also strong, with roughly 28% of orders each, showing customers shop throughout the day.
- Dawn (0-6h) is marginal, contributing only about 5% of orders, so operational focus should prioritise afternoon and evening peaks.

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

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

The screenshot shows a data processing interface with a query editor at the top and a results table below it.

Query Editor:

```

1 with state_month_orders as (
2     select
3         c.customer_state,
4             extract(year from o.order_purchase_timestamp) as year,
5             extract(month from o.order_purchase_timestamp) as month,
6             format_timestamp('%B', o.order_purchase_timestamp) as month_name,
7             count(distinct o.order_id) as total_orders
8     from Target.orders as o
9     join Target.customers as c
10    | on o.customer_id = c.customer_id
11    group by
12        c.customer_state, year, month, month_name
13 )
14
15 select
16     customer_state,
17     year,
18     month,
19     month_name,
20     total_orders,
21     lag(total_orders) over (
22         partition by customer_state
23         order by year, month
24     ) as prev_month_orders,
25     round(
26         (total_orders - lag(total_orders) over (
27             partition by customer_state
28             order by year, month
29         ))
30         / lag(total_orders) over (
31             partition by customer_state
32             order by year, month
33         ) * 100
34     , 2) as mom_growth_pct
35 from state_month_orders
36 order by customer_state, year, month;

```

Results Table:

Query completed

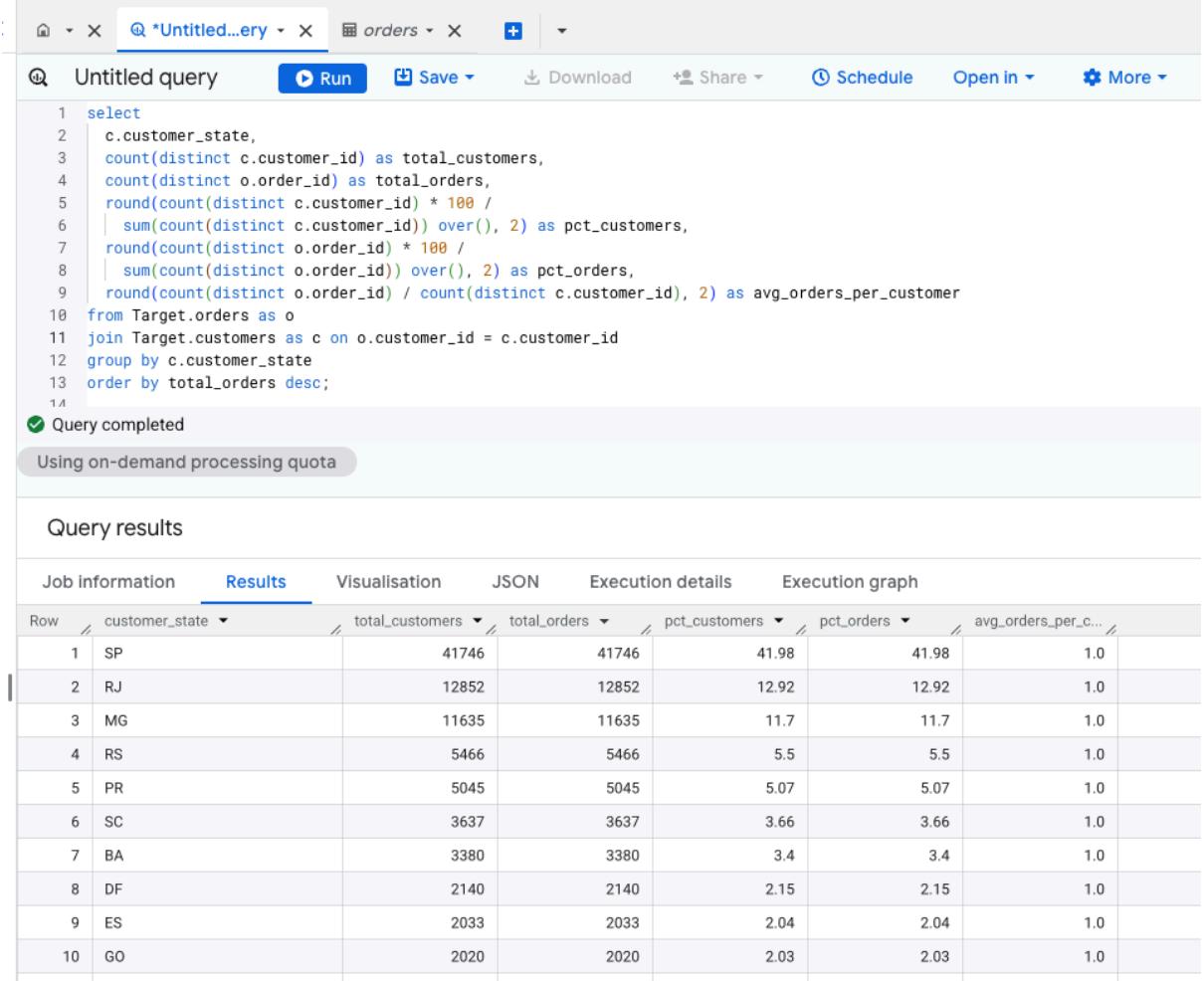
Using on-demand processing quota

Query results

Job information	Results	Visualisation	JSON	Execution details	Execution graph		
Row	customer_state	year	month	month_name	total_orders	prev_month_orders	mom_growth_pct
1	AC	2017	1	January	2	null	null
2	AC	2017	2	February	3	2	50.0
3	AC	2017	3	March	2	3	-33.33
4	AC	2017	4	April	5	2	150.0

- Most states show strong month on month (MoM) growth through 2017, especially early in the year e.g. BA, CE, PR all post frequent 40 to 80%+ MoM jumps in 2017
- Growth becomes more stable in 2018: many states still grow but MoM swings are generally smaller and alternate between moderate increases and corrections
- Several large states (BA, CE, PR, PA, PE) consistently maintain high order volumes across most months, confirming them as Target's core demand hubs, while smaller states (AC, AP, AM, AL, PI, PB) show more volatile MoM % reflecting smaller bases and more sensitivity to campaigns or local conditions

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



The screenshot shows a data analysis interface with a query editor and a results table.

Query Editor:

```

1 select
2   c.customer_state,
3   count(distinct c.customer_id) as total_customers,
4   count(distinct o.order_id) as total_orders,
5   round(count(distinct c.customer_id) * 100 /
6   | sum(count(distinct c.customer_id)) over(), 2) as pct_customers,
7   round(count(distinct o.order_id) * 100 /
8   | sum(count(distinct o.order_id)) over(), 2) as pct_orders,
9   round(count(distinct o.order_id) / count(distinct c.customer_id), 2) as avg_orders_per_customer
10  from Target.orders as o
11  join Target.customers as c on o.customer_id = c.customer_id
12  group by c.customer_state
13  order by total_orders desc;
14

```

Message: Query completed

Using on-demand processing quota

Query results:

Job information	Results	Visualisation	JSON	Execution details	Execution graph	
Row	customer_state	total_customers	total_orders	pct_customers	pct_orders	avg_orders_per_c...
1	SP	41746	41746	41.98	41.98	1.0
2	RJ	12852	12852	12.92	12.92	1.0
3	MG	11635	11635	11.7	11.7	1.0
4	RS	5466	5466	5.5	5.5	1.0
5	PR	5045	5045	5.07	5.07	1.0
6	SC	3637	3637	3.66	3.66	1.0
7	BA	3380	3380	3.4	3.4	1.0
8	DF	2140	2140	2.15	2.15	1.0
9	ES	2033	2033	2.04	2.04	1.0
10	GO	2020	2020	2.03	2.03	1.0

- The customer base is highly concentrated in SP, which alone contributes about 42% of customers and 42% of all orders, making it by far the dominant state.
- The next three states: RJ, MG and RS add roughly another 30%+ of customers and orders, so the top 4 states together account for over 70% of Target Brazil's demand.
- A long tail of smaller states (e.g. AC, AP, RR, AM) each contribute less than 1% of customers and orders, highlighting significant headroom for regional expansion outside the core regions if Target wants to reduce geographic concentration risk.

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

The screenshot shows a data processing interface with a query editor and a results table.

Query Editor:

```

1 with yearly_costs as (
2   select
3     extract(year from o.order_purchase_timestamp) as year,
4     sum(p.payment_value) as total_payment_value
5   from Target.orders as o
6   join Target.payments as p on o.order_id = p.order_id
7   where extract(month from o.order_purchase_timestamp) <= 8
8   and extract(year from o.order_purchase_timestamp) in (2017, 2018)
9   group by year
10 )
11 select
12   max(case when year = 2017 then total_payment_value end) as cost_2017_jan_to_aug,
13   max(case when year = 2018 then total_payment_value end) as cost_2018_jan_to_aug,
14   round(
15     (max(case when year = 2018 then total_payment_value end) -
16      max(case when year = 2017 then total_payment_value end)) /
17      max(case when year = 2017 then total_payment_value end) * 100, 2
18   as pct_increase
19   from yearly_costs;

```

Message Bar:

This query will process 8.14 MB when run.

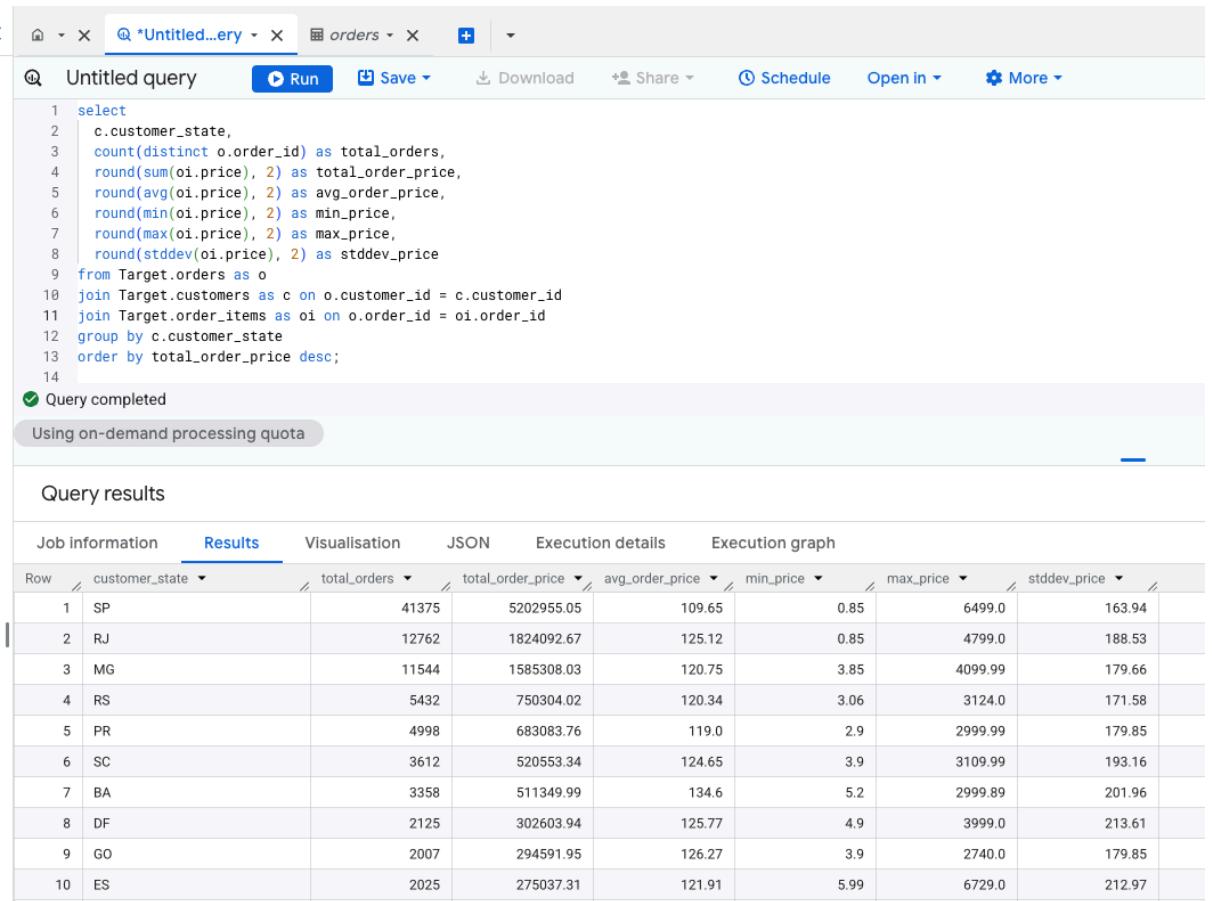
Using on-demand processing quota

Query results:

Job information	Results	Visualisation	JSON	Execution details	Execution graph
Row	cost_2017_jan_to...	cost_2018_jan_to...	pct_increase		
1	3669022.11999992	8694733.84000018	136.98		

- Total order value for Jan to Aug 2017 is about 3.67M, rising to roughly 8.69M for Jan to Aug 2018.
- This is a +136.98% year on year increase, meaning Target more than doubled its Brazil e-commerce revenue in just one year for the same period.
- Such rapid growth suggests very strong customer acquisition and/ or higher spend per order

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



The screenshot shows a data processing interface with a query editor and a results table. The query editor contains the following SQL code:

```

1 select
2     c.customer_state,
3     count(distinct o.order_id) as total_orders,
4     round(sum(oi.price), 2) as total_order_price,
5     round(avg(oi.price), 2) as avg_order_price,
6     round(min(oi.price), 2) as min_price,
7     round(max(oi.price), 2) as max_price,
8     round(stdev(oi.price), 2) as stddev_price
9 from Target.orders as o
10 join Target.customers as c on o.customer_id = c.customer_id
11 join Target.order_items as oi on o.order_id = oi.order_id
12 group by c.customer_state
13 order by total_order_price desc;
14

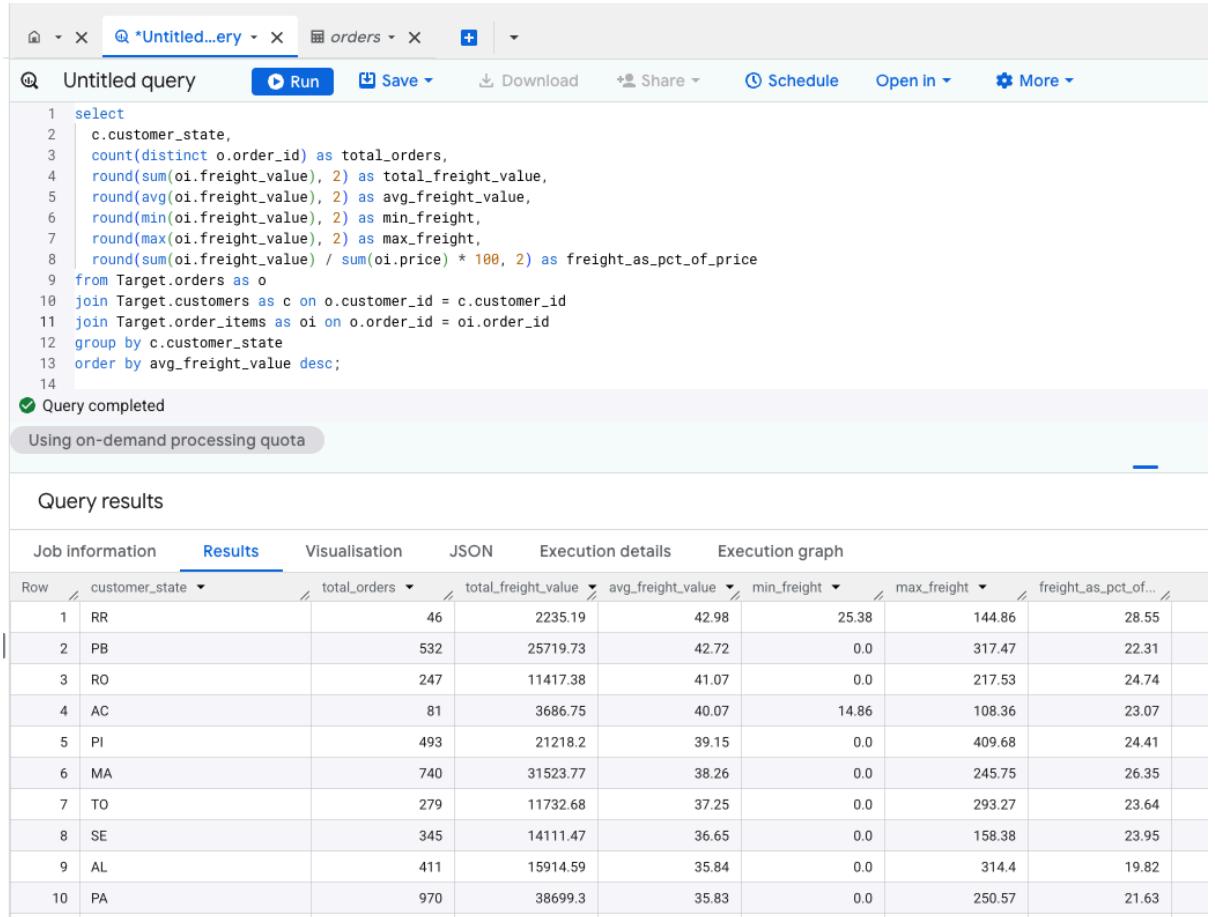
```

The results table has the following columns: Row, customer_state, total_orders, total_order_price, avg_order_price, min_price, max_price, stddev_price. The data is as follows:

Row	customer_state	total_orders	total_order_price	avg_order_price	min_price	max_price	stddev_price
1	SP	41375	5202955.05	109.65	0.85	6499.0	163.94
2	RJ	12762	1824092.67	125.12	0.85	4799.0	188.53
3	MG	11544	1585308.03	120.75	3.85	4099.99	179.66
4	RS	5432	750304.02	120.34	3.06	3124.0	171.58
5	PR	4998	683083.76	119.0	2.9	2999.99	179.85
6	SC	3612	520553.34	124.65	3.9	3109.99	193.16
7	BA	3358	511349.99	134.6	5.2	2999.89	201.96
8	DF	2125	302603.94	125.77	4.9	3999.0	213.61
9	GO	2007	294591.95	126.27	3.9	2740.0	179.85
10	ES	2025	275037.31	121.91	5.99	6729.0	212.97

- Revenue is heavily concentrated in SP alone and generates about 5.2M in order value from ~41k orders, far ahead of any other state
- RJ, MG and RS each contribute between ~0.75M and 1.8M, so the top four states together account for the clear majority of national order value.
- Average order prices are fairly similar across the large states (roughly between 110 to 135), but smaller states like PA, MT, MA show higher average ticket values (often between 145 to 180), suggesting fewer but more expensive purchases in these emerging markets.

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



The screenshot shows a data analysis interface with a query editor and a results table. The query editor contains the following SQL code:

```

1 select
2   c.customer_state,
3   count(distinct o.order_id) as total_orders,
4   round(sum(oi.freight_value), 2) as total_freight_value,
5   round(avg(oi.freight_value), 2) as avg_freight_value,
6   round(min(oi.freight_value), 2) as min_freight,
7   round(max(oi.freight_value), 2) as max_freight,
8   round(sum(oi.freight_value) / sum(oi.price) * 100, 2) as freight_as_pct_of_price
9 from Target.orders as o
10 join Target.customers as c on o.customer_id = c.customer_id
11 join Target.order_items as oi on o.order_id = oi.order_id
12 group by c.customer_state
13 order by avg_freight_value desc;
14

```

The results table has columns: Row, customer_state, total_orders, total_freight_value, avg_freight_value, min_freight, max_freight, and freight_as_pct_of_price. The data is as follows:

Row	customer_state	total_orders	total_freight_value	avg_freight_value	min_freight	max_freight	freight_as_pct_of...
1	RR	46	2235.19	42.98	25.38	144.86	28.55
2	PB	532	25719.73	42.72	0.0	317.47	22.31
3	RO	247	11417.38	41.07	0.0	217.53	24.74
4	AC	81	3686.75	40.07	14.86	108.36	23.07
5	PI	493	21218.2	39.15	0.0	409.68	24.41
6	MA	740	31523.77	38.26	0.0	245.75	26.35
7	TO	279	11732.68	37.25	0.0	293.27	23.64
8	SE	345	14111.47	36.65	0.0	158.38	23.95
9	AL	411	15914.59	35.84	0.0	314.4	19.82
10	PA	970	38699.3	35.83	0.0	250.57	21.63

- Average freight is highest in low volume states such as RR, PB, RO, AC and PI, where average freight per order is around 38 to 43 and freight often represents more than 23 to 28% of the product price.
- In contrast, large, dense markets like SP, RJ, MG, PR and RS have lower average freight per order (about 15 to 21) and freight typically accounts for 13 to 18% of item price, reflecting better logistics efficiency and economies of scale.
- This pattern suggests that deliveries to remote or less served regions are significantly more expensive on a per order basis, so Target may need differentiated shipping strategies to grow profitably outside the main urban centres.

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

- 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:

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

The screenshot shows a database query interface with the following details:

- Query:** Untitled query
- Run Status:** Query completed
- Results:** Using on-demand processing quota
- Job Information:** Job ID: 0005f50442cb953dc1d21e1fb...
- Results:** This tab is selected, showing a table with the following columns:

Row	order_id	customer_state	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	time_to_deliver_d...	diff_estimated_de...	delivery_status
1	00010242fe8c5a6d1ba2dd792c...	RJ	2017-09-13 08:59:02 UTC	2017-09-20 23:43:48 UTC	2017-09-29 00:00:00 UTC	7	-8	early
2	00018f77f2f0320c557190d7a1...	SP	2017-04-26 10:53:06 UTC	2017-05-12 16:04:24 UTC	2017-05-15 00:00:00 UTC	16	-2	early
3	000229ec398224ef6ca0657da4...	MG	2018-01-14 14:33:31 UTC	2018-01-22 13:19:16 UTC	2018-02-05 00:00:00 UTC	7	-13	early
4	00024acbcdf0a6daa1e931b038...	SP	2018-08-08 10:00:35 UTC	2018-08-14 13:32:39 UTC	2018-08-20 00:00:00 UTC	6	-5	early
5	00042b26cf59d7ce9dfab4e5...	SP	2017-02-04 13:57:51 UTC	2017-03-01 16:42:31 UTC	2017-03-17 00:00:00 UTC	25	-15	early
6	00048cc3ae777c65dbb7d2a06...	MG	2017-05-15 21:42:34 UTC	2017-05-22 13:44:35 UTC	2017-06-06 00:00:00 UTC	6	-14	early
7	00054e8431b9d7675808bb81...	SP	2017-12-10 11:53:48 UTC	2017-12-18 22:03:38 UTC	2018-01-04 00:00:00 UTC	8	-16	early
8	000576e39319847ccb9d288c5...	SP	2018-07-04 12:08:27 UTC	2018-07-09 14:04:07 UTC	2018-07-25 00:00:00 UTC	5	-15	early
9	0005a1a1728c9d785b8e2b08b...	SP	2018-03-19 18:40:33 UTC	2018-03-29 18:17:31 UTC	2018-03-29 00:00:00 UTC	9	0	on time
10	0005f50442cb953dc1d21e1fb...	SP	2018-07-02 13:59:39 UTC	2018-07-04 17:28:31 UTC	2018-07-23 00:00:00 UTC	2	-18	early
- Visualisation:** Not selected
- JSON:** Not selected
- Execution details:** Not selected
- Execution graph:** Not selected

- Most delivered orders arrive well before the promised date: many shipments are between 5 to 10 days faster than the estimate, and negative `diff_estimated_delivery_days` values dominate, labelled as “early” deliveries.
- Typical time to deliver for these orders is roughly 1 to 3 weeks from purchase, with many examples around 5 to 10 days, which is competitive for a nationwide operation with long distances.
- Only a small minority of orders are “late” (positive difference vs the estimated date), suggesting that Target’s service levels are generally conservative and often exceeded, which should support strong customer satisfaction.

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

Highest

```

1 select
2   c.customer_state,
3   count(distinct o.order_id) as total_orders,
4   round(avg(oi.freight_value), 2) as avg_freight_value
5 from Target.orders as o
6 join Target.customers as c on o.customer_id = c.customer_id
7 join Target.order_items as oi on o.order_id = oi.order_id
8 group by c.customer_state
9 order by avg_freight_value desc
10 limit 5;
11

```

✓ Query completed

Using on-demand processing quota

Query results

Job information	Results	Visualisation	JSON	Execution details
Row	customer_state	total_orders	avg_freight_value	
1	RR	46	42.98	
2	PB	532	42.72	
3	RO	247	41.07	
4	AC	81	40.07	
5	PI	493	39.15	

- The highest average freight costs are in RR, PB, RO, AC and PI, all between 39 to 43 per order, which is roughly double the freight levels seen in major states like SP or RJ.
- These are low volume, geographically remote states, so fixed logistics costs are spread over fewer orders, making each delivery significantly more expensive and suggesting a need for regional hubs or differentiated shipping policies there.

Lowest

The screenshot shows a data analysis interface with the following components:

- Top Bar:** Includes a back button, a search bar containing "Untitled...ery", a refresh button, a tab labeled "orders", a plus sign for new queries, and a dropdown menu.
- Query Editor:** A text area titled "Untitled query" containing the following SQL code:

```
1 select
2   c.customer_state,
3   count(distinct o.order_id) as total_orders,
4   round(avg(oi.freight_value), 2) as avg_freight_value
5 from Target.orders as o
6 join Target.customers as c on o.customer_id = c.customer_id
7 join Target.order_items as oi on o.order_id = oi.order_id
8 group by c.customer_state
9 order by avg_freight_value asc
10 limit 5;
11
```
- Status Bar:** Displays a green checkmark icon and the text "Query completed".
- Results Section:** A table titled "Query results" with the following columns: Job information, Results, Visualisation, JSON, and Execution details. The "Results" tab is selected.
- Data Table:** A table showing the top 5 states with the lowest average freight costs. The columns are Row, customer_state, total_orders, and avg_freight_value.

Row	customer_state	total_orders	avg_freight_value
1	SP	41375	15.15
2	PR	4998	20.53
3	MG	11544	20.63
4	RJ	12762	20.96
5	DF	2125	21.04

- The top 5 states with the lowest average freight costs are SP, PR, MG, RJ and DF, with average freight per order ranging from about 15 to 21.
- These are the largest, most densely populated and urbanised states, where high order volumes and shorter delivery distances translate into significant economies of scale in logistics

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

Highest

```

1 select
2   c.customer_state,
3   count(distinct o.order_id) as total_orders,
4   round(avg(timestamp_diff(o.order_delivered_customer_date, o.order_purc
5   | as avg_delivery_time_days
6   from Target.orders as o
7   join Target.customers as c on o.customer_id = c.customer_id
8   where o.order_delivered_customer_date is not null
9   group by c.customer_state
10  order by avg_delivery_time_days desc
11  limit 5;
12

```

Query completed

Using on-demand processing quota

Row	customer_state	total_orders	avg_delivery_time...
1	RR	41	28.98
2	AP	67	26.73
3	AM	145	25.99
4	AL	397	24.04
5	PA	946	23.32

- The slowest delivery states are RR, AP, AM, AL and PA, averaging between 23 to 29 days from purchase to delivery, which is significantly longer than the major states and reflects geographic remoteness and limited logistics infrastructure
- Target should consider establishing regional distribution hubs or partnering with local carriers in these states to reduce delivery times and improve competitiveness, since long waits damage customer satisfaction.

Lowest

The screenshot shows a data processing interface with the following details:

- Query Title:** Untitled query
- Run Status:** Query completed
- Processing Quota:** Using on-demand processing quota
- Job Information:** Row, customer_state, total_orders, avg_delivery_time...
- Results:** A table with 5 rows of data:

Row	customer_state	total_orders	avg_delivery_time...
1	SP	40495	8.3
2	PR	4923	11.53
3	MG	11355	11.54
4	DF	2080	12.51
5	SC	3547	14.48

- The fastest delivery states are SP, PR, MG, DF and SC, with average delivery times between roughly 8 to 15 days from purchase to customer receipt.
- These are all large, well-connected regions with high order volumes and good transport infrastructure, so they set the benchmark service level that Target should aim to replicate in slower regions where feasible.

- 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.

The screenshot shows a data analysis interface with the following components:

- Header:** Untitled query, Run, Save, Download, Share, Schedule, Open in.
- Query Editor:** A code editor containing the following SQL query:


```

1 select
2   c.customer_state,
3   count(distinct o.order_id) as total_orders,
4   round(avg(timestamp_diff(o.order_estimated_delivery_date, o.order_delivered_customer_date, day)), 2)
5   | as avg_days_ahead_of_estimate
6   from Target.orders as o
7   join Target.customers as c on o.customer_id = c.customer_id
8   where o.order_delivered_customer_date is not null
9   | and o.order_estimated_delivery_date is not null
10  group by c.customer_state
11  having avg(timestamp_diff(o.order_estimated_delivery_date, o.order_delivered_customer_date, day)) > 0
12  order by avg_days_ahead_of_estimate desc
13  limit 5;
14
      
```
- Status:** Query completed, Using on-demand processing quota.
- Results:** A table titled "Query results" showing the following data:

Job information	Results	Visualisation	JSON	Execution details	Execution graph
Row	customer_state	total_orders	avg_days_ahead_of_estimate		
1	AC	80	19.76		
2	RO	243	19.13		
3	AP	67	18.73		
4	AM	145	18.61		
5	RR	41	16.41		

- The states that most exceed delivery promises are AC, RO, AP, AM and RR, where orders arrive on average 16 to 20 days earlier than the estimated delivery date.
- These regions are typically remote and slow on paper, so Target's estimates are very conservative; consistently beating them by such a wide margin is great for customer satisfaction but also suggests the SLA dates could be tightened to set more realistic expectations.

6. Analysis based on the payments:

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

Untitled query

Run Save Download Share Schedule Open in More

```

1 with payment_month_data as (
2     select
3         extract(year from o.order_purchase_timestamp) as year,
4         extract(month from o.order_purchase_timestamp) as month,
5         format_timestamp('%B', o.order_purchase_timestamp) as month_name,
6         p.payment_type,
7         count(distinct o.order_id) as total_orders,
8         round(sum(p.payment_value), 2) as total_payment_value
9     from Target.orders as o
10    join Target.payments as p
11      | on o.order_id = p.order_id
12    group by year, month, month_name, p.payment_type
13 )
14
15 select
16     year,
17     month,
18     month_name,
19     payment_type,
20     total_orders,
21     total_payment_value,
22     lag(total_orders) over (
23         partition by payment_type
24         order by year, month
25     ) as prev_month_orders
26   from payment_month_data
27 order by year, month, payment_type;
28

```

Query completed

Using on-demand processing quota

Query results

Job information	Results	Visualisation	JSON	Execution details	Execution graph		
Row	year	month	month_name	payment_type	total_orders	total_payment_va...	prev_month_orders
1	2016	9	September	credit_card	3	252.24	null
2	2016	10	October	UPI	63	9679.06	null
3	2016	10	October	credit_card	253	48290.62	3
4	2016	10	October	debit_card	2	241.73	null
5	2016	10	October	voucher	11	879.07	null
6	2016	12	December	credit_card	1	19.62	253
7	2017	1	January	UPI	197	24074.43	63
8	2017	1	January	credit_card	582	109615.68	1
9	2017	1	January	debit_card	9	743.53	2
10	2017	1	January	voucher	33	4054.4	11

- Credit card shows strong consistent growth through 2017, ramping from ~600 orders in Jan 2017 to ~5,900 in Nov 2017, then stabilising around 4,700 to 5,700 through mid 2018, maintaining its dominant share without much volatility.
- UPI grows even faster percentage wise, nearly tripling from ~200 orders in Jan 2017 to ~1,500 by late 2017, then stabilising around 1,100 to 1,500 in 2018, showing strong adoption of this alternative payment method as a strategic second channel.
- Vouchers and debit cards remain flat and tiny (<250 and <100 orders/ month respectively), so efforts to drive payment diversity should focus on UPI incentives rather than voucher campaigns.

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

```

1 select
2   p.payment_installments,
3   count(distinct o.order_id) as total_orders,
4   round(sum(p.payment_value), 2) as total_payment_value,
5   round(avg(p.payment_value), 2) as avg_payment_value,
6   round(count(distinct o.order_id) * 100 /
7   | sum(count(distinct o.order_id)) over(), 2) as pct_of_total_orders
8   from Target.orders as o
9   join Target.payments as p on o.order_id = p.order_id
10  group by p.payment_installments
11  order by p.payment_installments;
12

```

Query completed

Using on-demand processing quota

Query results

Job information	Results	Visualisation	JSON	Execution details	Execution graph
Row	payment_installments	total_orders	total_payment_value	avg_payment_value	pct_of_total_orders
1	0	2	188.63	94.31	0.0
2	1	49060	5907233.36	112.42	48.91
3	2	12389	1579283.03	127.23	12.35
4	3	10443	1491103.8	142.54	10.41
5	4	7088	1163907.61	163.98	7.07
6	5	5234	961174.3	183.47	5.22
7	6	3916	822611.81	209.85	3.9
8	7	1623	305157.39	187.67	1.62
9	8	4253	1313423.34	307.74	4.24
10	9	644	131015.92	203.44	0.64

- Single payment is the norm: nearly 49% of all orders are paid upfront, with an average ticket of ~112, reflecting the price conscious segment.
- 2 to 3 installments are sweet spots, capturing ~22% of orders combined and much higher average order values (127 to 143 per order), suggesting that offering flexible 2 to 3 month payment plans drives higher-value sales without materially increasing default risk.
- Installments beyond 10 are niche and problematic: only ~0.1% of orders use >10 installments, and many show zero payment value or only partial captures, indicating these edge cases are likely failed/ disputed transactions or data anomalies that Target should investigate

7. Additional insights

7.1. Order status distribution

The screenshot shows a data analysis interface with the following components:

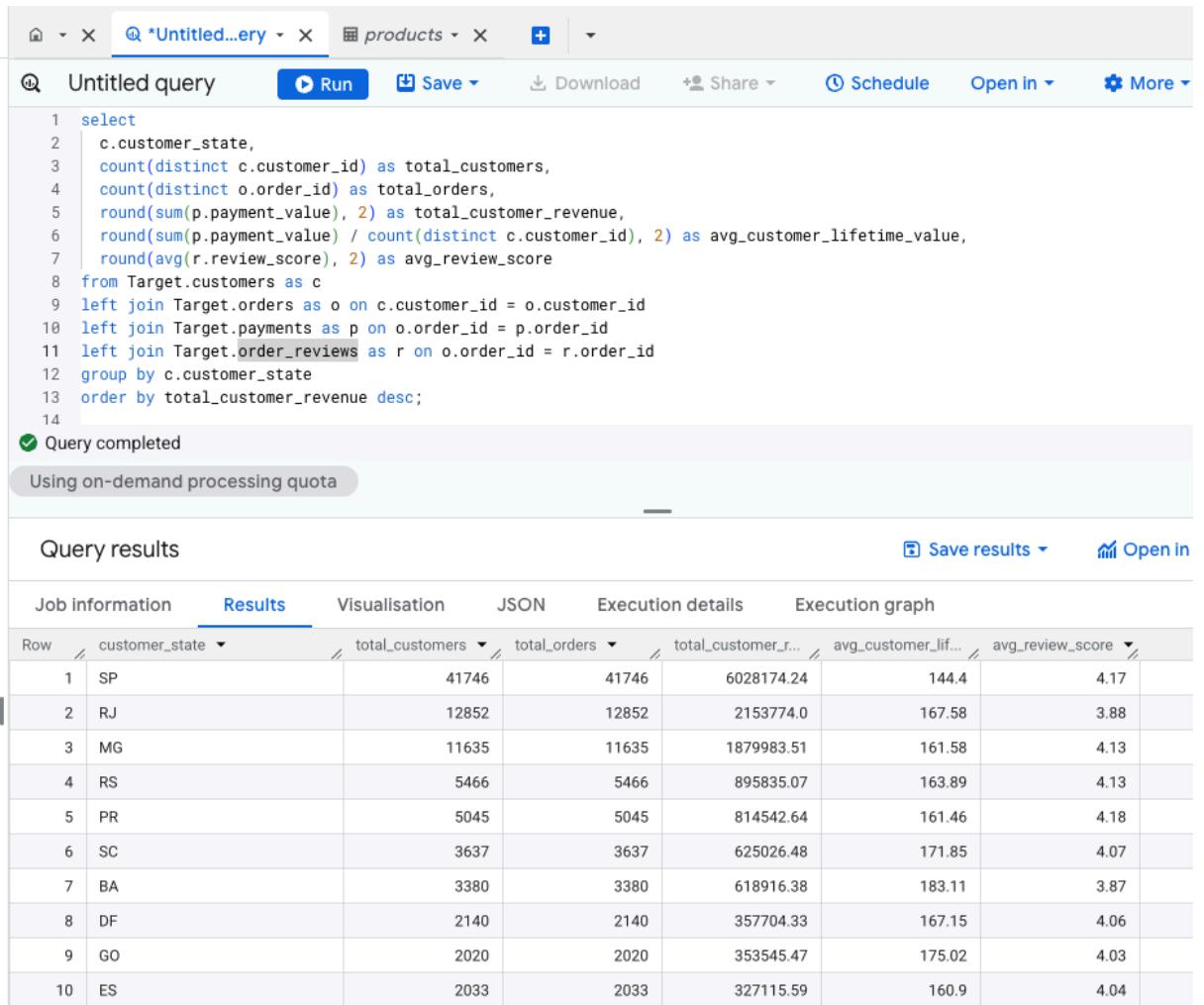
- Top Bar:** Includes tabs for "payments" and "Untitled...ery".
- Query Editor:** Titled "Untitled query". Contains the following SQL code:

```
1 select
2   order_status,
3   count(distinct order_id) as total_orders,
4   round(count(distinct order_id) * 100 /
5   | sum(count(distinct order_id)) over(), 2) as pct_of_total
6 from Target.orders
7 group by order_status
8 order by total_orders desc;
```
- Status Bar:** Shows a green checkmark icon and the text "Query completed".
- Message Bar:** Shows a grey button with the text "Using on-demand processing quota".
- Results Section:** Titled "Query results". Includes tabs for "Job information", "Results" (which is selected), "Visualisation", "JSON", and "Execution details".
- Data Table:** Displays the results of the query. The columns are "Row", "order_status", "total_orders", and "pct_of_total". The data is as follows:

Row	order_status	total_orders	pct_of_total
1	delivered	96478	97.02
2	shipped	1107	1.11
3	canceled	625	0.63
4	unavailable	609	0.61
5	invoiced	314	0.32
6	processing	301	0.3
7	created	5	0.01
8	approved	2	0.0

- Delivered orders dominate at 97% of all orders, demonstrating that Target's fulfillment operation is highly reliable and most customers receive their purchases successfully.
- Problem orders are minimal: cancellations (0.63%), unavailable (0.61%) and other non delivered statuses together account for less than 1.5% of orders, indicating very low churn and strong inventory/product availability management.
- In-transit orders are small (~1% in "shipped" status), suggesting a fast turnover from shipment to delivery, consistent with the earlier finding that most orders arrive in 8 to 15 days.

7.2. Customer Lifetime Value (CLV) by state



The screenshot shows a data analysis interface with a query editor and a results table.

Query Editor:

```

1 select
2   c.customer_state,
3   count(distinct c.customer_id) as total_customers,
4   count(distinct o.order_id) as total_orders,
5   round(sum(p.payment_value), 2) as total_customer_revenue,
6   round(sum(p.payment_value) / count(distinct c.customer_id), 2) as avg_customer_lifetime_value,
7   round(avg(r.review_score), 2) as avg_review_score
8 from Target.customers as c
9 left join Target.orders as o on c.customer_id = o.customer_id
10 left join Target.payments as p on o.order_id = p.order_id
11 left join Target.order_reviews as r on o.order_id = r.order_id
12 group by c.customer_state
13 order by total_customer_revenue desc;
14

```

Message: Query completed

Using on-demand processing quota

Query results:

Job information	Results	Visualisation	JSON	Execution details	Execution graph	
Row	customer_state	total_customers	total_orders	total_customer_r...	avg_customer_lif...	avg_review_score
1	SP	41746	41746	6028174.24	144.4	4.17
2	RJ	12852	12852	2153774.0	167.58	3.88
3	MG	11635	11635	1879983.51	161.58	4.13
4	RS	5466	5466	895835.07	163.89	4.13
5	PR	5045	5045	814542.64	161.46	4.18
6	SC	3637	3637	625026.48	171.85	4.07
7	BA	3380	3380	618916.38	183.11	3.87
8	DF	2140	2140	357704.33	167.15	4.06
9	GO	2020	2020	353545.47	175.02	4.03
10	ES	2033	2033	327115.59	160.9	4.04

- SP dominates in absolute CLV: with ~6.03M total customer lifetime value, it represents ~55% of all revenue from 41,746 customers, confirming it is the revenue powerhouse and must remain the primary focus for retention and upsell.
- RJ, MG and RS form the secondary tier: contributing 2.15M, 1.88M and 0.90M respectively, together making up another ~30% of CLV; these should be treated as strategic second-priority markets for growth and profitability initiatives.
- Regional variation in avg CLV per customer is modest: most states cluster between 144 to 224 per customer, suggesting fairly consistent purchase behaviour across the country, though smaller/ remote states like PA, CE and PB have slightly higher per-capita values, likely due to higher average order prices offsetting lower order frequency.

7.3. Seller performance by state

Untitled query Run Save Download Share Schedule Open in More

```

1 select
2   s.seller_state,
3   count(distinct s.seller_id) as total_sellers,
4   count(distinct oi.order_id) as total_orders,
5   round(sum(oi.price), 2) as total_sales,
6   round(avg(oi.price), 2) as avg_item_price,
7   round(avg(oi.freight_value), 2) as avg_freight,
8   round(avg(r.review_score), 2) as avg_review_score,
9   count(distinct case when r.review_score >= 4 then oi.order_id end) as high_rated_orders
10  from Target.sellers as s
11  join Target.order_items as oi on s.seller_id = oi.seller_id
12  left join Target.orders as o on oi.order_id = o.order_id
13  left join Target.order_reviews as r on o.order_id = r.order_id
14  group by s.seller_state
15  order by total_sales desc;
16

```

Query completed
Using on-demand processing quota

Query results

Job information	Results	Visualisation	JSON	Execution details	Execution graph			
Row	seller_state	total_sellers	total_orders	total_sales	avg_item_price	avg_freight	avg_review_score	high_rated_orders
1	SP	1849	70188	8794512.46	108.8	18.44	4.01	53390
2	PR	349	7673	1270406.16	145.22	22.68	4.07	5990
3	MG	244	7930	1015142.45	114.4	24.06	4.11	6257
4	RJ	171	4353	846763.01	175.13	19.49	4.1	3425
5	SC	190	3667	633864.61	155.02	26.13	4.09	2853
6	RS	129	1989	381013.3	172.4	26.03	4.21	1640
7	BA	19	569	285682.56	442.92	30.65	4.09	446
8	DF	30	824	97821.36	108.57	20.57	4.03	625
9	PE	9	406	91493.85	204.23	27.66	4.13	341
10	GO	40	463	66479.11	127.6	24.17	4.25	388

- SP seller ecosystem is by far the largest: 1,849 sellers generating 8.79M in sales across 70,188 orders, with an average review of 4.01 and high concentration of 5 star orders (53,390), confirming that SP is both a supply and demand powerhouse.
- Quality vs quantity trade off is evident: PR has only 349 sellers but delivers higher average product prices (145) and strong 4.07 rating, while MG (244 sellers) shows premium positioning with 4.11 rating; in contrast, smaller seller bases in CE (13 sellers) and PB (6 sellers) have unusually high avg item prices (215 to 450), suggesting niche/speciality offerings rather than mass-market goods.
- Freight costs vary by state, ranging from 18 to 51 per item: SP/ DF have lowest, while remote states (PB, RO, CE, PI) have the highest, underlining that seller incentive structures and logistics partnerships must be regionally differentiated to maintain competitive margins.

Actionable Insights & Recommendations:

Immediate Actions (0 to 3 months):

Implement tiered freight pricing by region - apply minimum order thresholds or freight surcharges in states where delivery costs consume 23 to 28% of order value, while maintaining free/ discounted shipping for orders >200 to incentivise larger baskets.

Also, optimize checkout for credit card and UPI only; remove voucher promotion and redirect that budget to 2 to 3 installment plan incentives, which data shows drive 35 to 50% higher average order values. Launch Q4 inventory surge planning immediately, as Nov to Dec orders peak at 5,800 to 7,500/ month versus baseline of 3,000 to 4,000, requiring 50%+ additional warehouse and last-mile capacity.

Strategic Growth (3 to 12 months):

Establish regional distribution hubs in PE, BA, and CE to reduce delivery times from 23+ days to <15 days, unlocking margin recovery and customer satisfaction.

Also, launch a tier 2 city seller recruitment programme targeting PR, MG, and BA to diversify the seller ecosystem beyond SP dominance (currently 1,849 of 3,300 total sellers).

Implement seller rating SLAs (minimum 3.9 avg, <0.5% cancellation rate) with monthly bonuses for high performers, mimicking SP 4.01 average.

Finally, reset delivery SLA estimates downward in major states (SP: 8 days, RJ/ MG: 12 days) to consistently beat promises and build competitive differentiation, converting delivery reliability into a marketing advantage as the market matures beyond pure growth.