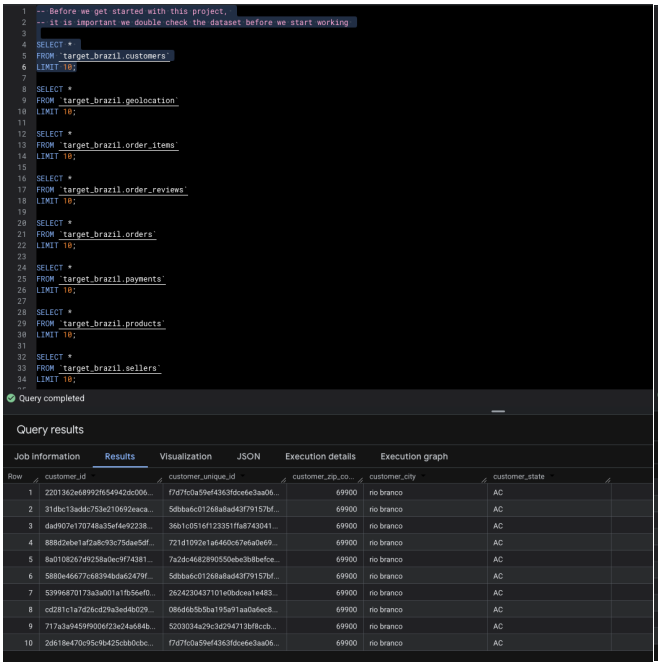
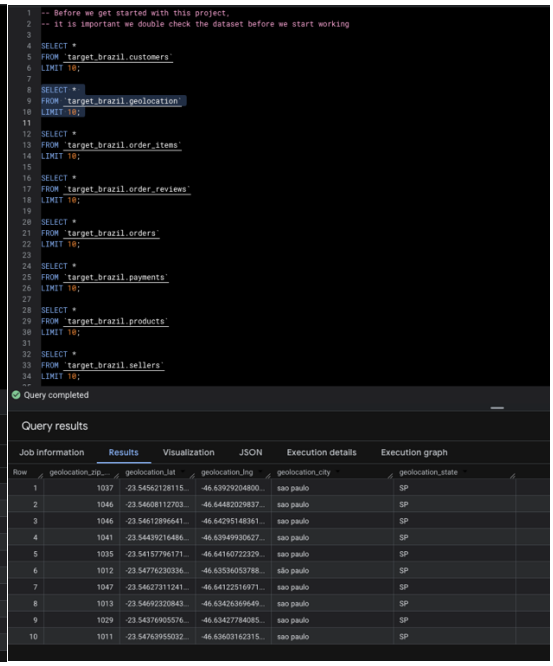
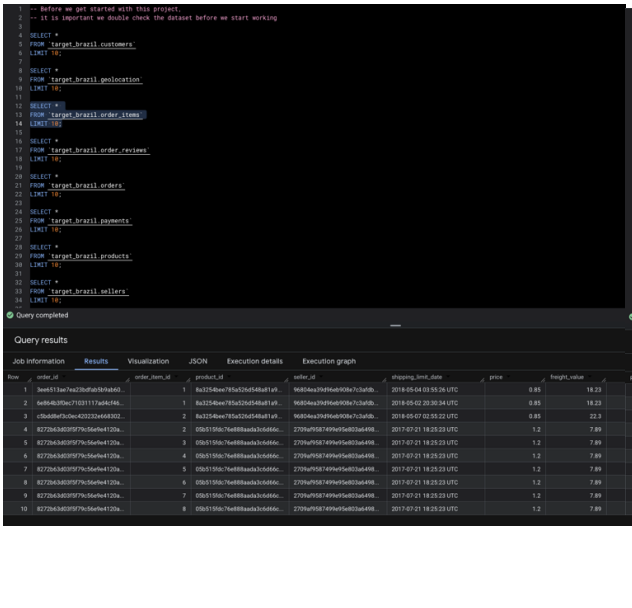
Target Brazil – SQL Case Study (BigQuery)
By: Shivesh Raj Sahu

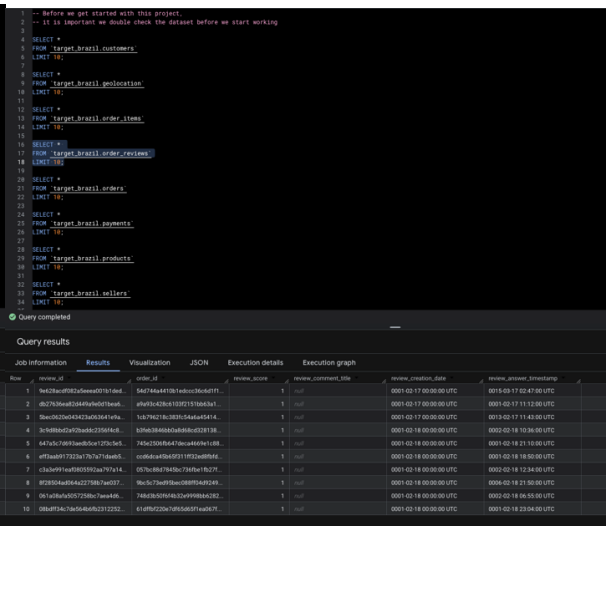
Dataset & Setup

A small sets of screenshots, of the BigQuery dataset tree showing all 8 tables.









Target Brazil – SQL Case Study (BigQuery)
By: Shivesh Raj Sahu

```sql
-- Before we get started with this project,
-- it is important we double check the dataset before we start working

SELECT *
FROM `target_brazil.customers`
LIMIT 10;

SELECT *
FROM `target_brazil.geolocation`
LIMIT 10;

SELECT *
FROM `target_brazil.order_items`
LIMIT 10;

SELECT *
FROM `target_brazil.order_reviews`
LIMIT 10;

SELECT *
FROM `target_brazil.orders`
LIMIT 10;

SELECT *
FROM `target_brazil.payments`
LIMIT 10;

SELECT *
FROM `target_brazil.products`
LIMIT 10;

SELECT *
FROM `target_brazil.sellers`
LIMIT 10;
```

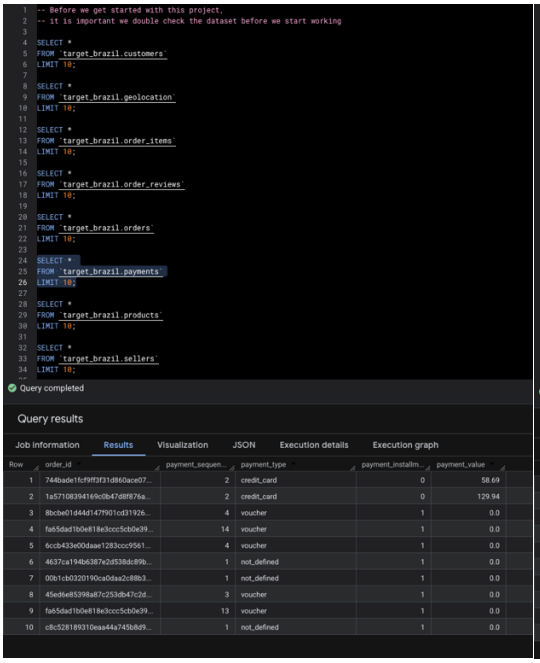
Query completed

Query results

Save results | Open in

Job information | Results | Visualization | JSON | Execution details | Execution graph

Row	order_id	customer_id	order_status	order_purchase_timestamp	order_approved_at	order_delivered_carrier_date	order_delivered_customer_date	order_estimated_delivery_date
1	a2e4c44360b4a57bdff22f3a46...	8886130db0ea6e9e70ba0b03d...	approved	2017-02-06 20:18:17 UTC	2017-02-06 20:30:19 UTC	null	null	2017-03-01 00:00:00 UTC
2	132f1e724165a07f6362532bfb...	b2191912d8ad6eac2e4dc3b6e...	approved	2017-04-25 01:25:34 UTC	2017-04-30 20:32:41 UTC	null	null	2017-05-22 00:00:00 UTC
3	809a282bbd5dbcabb6f2f724fc...	622e13439d6b5a0b486c43561...	canceled	2016-09-13 15:24:19 UTC	2016-10-07 13:16:46 UTC	null	null	2016-09-30 00:00:00 UTC
4	e5215415bb6f76fe3b7cb68103...	b6f6cbfc126f1ae6723fe2f9b37...	canceled	2016-10-22 08:25:27 UTC	null	null	null	2016-10-24 00:00:00 UTC
5	71303d7e93b399f5bcd537d12...	b106b360fe2ef8849fbbd056f7...	canceled	2016-10-02 22:07:52 UTC	2016-10-06 15:50:56 UTC	null	null	2016-10-25 00:00:00 UTC
6	e5fa5a7210941f7d56d0208e4e...	683c54fc24d40ee9f8a6fc179fd...	canceled	2016-09-05 00:15:34 UTC	2016-10-07 13:17:15 UTC	null	null	2016-10-28 00:00:00 UTC
7	92d731517f17c26f161820d549...	95c44daefa7bfb91c6bb0665a0...	canceled	2016-10-05 11:23:13 UTC	null	null	null	2016-11-14 00:00:00 UTC
8	ddaec6fff982b13e7e048b627a...	68f4ad79cc0c2ad06e19088f5c...	canceled	2016-10-04 19:41:32 UTC	null	null	null	2016-11-16 00:00:00 UTC
9	c549f0d88f33bfe43351b893f1f...	ea2695007541d4c3569a7185...	canceled	2016-10-06 13:21:05 UTC	null	null	null	2016-11-16 00:00:00 UTC
10	9bd5312d12f48b04a460f702ba...	4d145fd1d3245cbb3697dd41b...	canceled	2016-10-06 20:43:30 UTC	null	null	null	2016-11-16 00:00:00 UTC

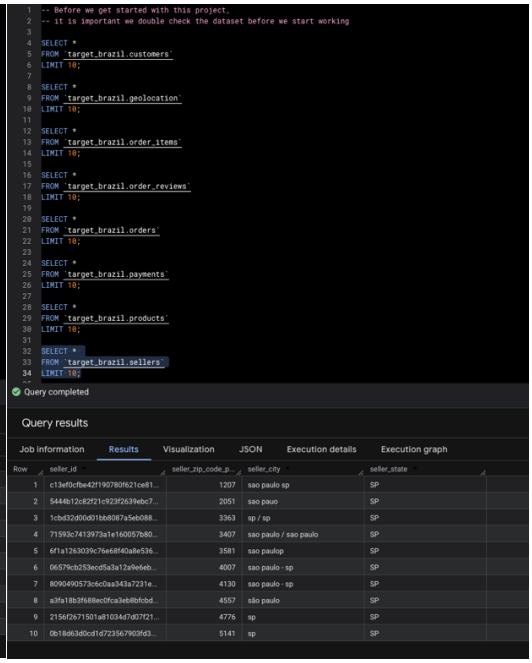


Query completed

Query results

Job information | Results | Visualization | JSON | Execution details | Execution graph

Row	order_id	payment_sequen...	payment_type	payment_installm...	payment_value
1	744bade1fcf9ff3f31d860ace07...	2	credit_card	0	58.69
2	1a5710839419c0b47d8f876a...	2	credit_card	0	129.94
3	8bcbe01d44d147f901cd3192b...	4	voucher	1	0.0
4	fa65dad1b0e818e3ccc5cb0e39...	14	voucher	1	0.0
5	6ccb433e00daae1283ccc9561...	4	voucher	1	0.0
6	4637ca194b6387e2d538dc89b...	1	not_defined	1	0.0
7	00b1cb0320190ca0daa2c88b3...	1	not_defined	1	0.0
8	45ed6e85398a87c253db47c2d...	3	voucher	1	0.0
9	fa65dad1b0e818e3ccc5cb0e39...	13	voucher	1	0.0
10	c8c528189310eaa44a745b8d9...	1	not_defined	1	0.0



Query completed

Query results

Job information | Results | Visualization | JSON | Execution details | Execution graph

Row	seller_id	seller_zip_code_p...	seller_city	seller_state
1	c13ef0cfbe42f190780f621ce81...	1207	sao paulo sp	SP
2	5444b12c82f21c923f2639ebc7...	2051	sao pauo	SP
3	1cbd32d00d01bb8087a5eb088...	3363	sp / sp	SP
4	71593c7413973a1e160057b80...	3407	sao paulo / sao paulo	SP
5	6f1a1263039c76e68f40a8e536...	3581	sao paulop	SP
6	06579cb253ecd5a3a12a9e6eb...	4007	sao paulo - sp	SP
7	8090490573c6c0aa343a7231e...	4130	sao paulo - sp	SP
8	a3fa18b3f688ec0fca3eb8bfcbd...	4557	são paulo	SP
9	2156f2671501a81034d7d07f21...	4776	sp	SP
10	0b18d63d0cd1d723567903fd3...	5141	sp	SP

Target Brazil – SQL Case Study (BigQuery)
By: Shivesh Raj Sahu

```sql
-- Before we get started with this project,
-- it is important we double check the dataset before we start working

SELECT *
FROM `target_brazil.customers`
LIMIT 10;

SELECT *
FROM `target_brazil.geolocation`
LIMIT 10;

SELECT *
FROM `target_brazil.order_items`
LIMIT 10;

SELECT *
FROM `target_brazil.order_reviews`
LIMIT 10;

SELECT *
FROM `target_brazil.orders`
LIMIT 10;

SELECT *
FROM `target_brazil.payments`
LIMIT 10;

SELECT *
FROM `target_brazil.products`
LIMIT 10;

SELECT *
FROM `target_brazil.sellers`
LIMIT 10;
```

Query completed

Query results

Job information | Results | Visualization | JSON | Execution details | Execution graph

Row	product_id	product category	product_name_le...	product_descripti...	product_photos_...	product_weight_g	product_length_cm	product_height_cm	product_width_cm
1	5eb564652db742ff8f28759cd8...	null	null	null	null	null	null	null	null
2	09ff539a621711667c43eba6a3...	babies	60	865	3	null	null	null	null
3	a69f15dfb803d485e8933e80b9...	Watches present	53	506	6	150	11	16	6
4	2f763ba79d9cd987b2034aac7...	electronics	45	1198	2	595	8	6	6
5	106392145fca363410d287a81...	bed table bath	58	309	1	2083	12	2	7
6	7e33f4a1c59f89da30a335b2dc...	electronics	51	381	3	1075	22	5	7
7	bc9cc914f974963c07be697fc9...	HEALTH BEAUTY	55	435	1	75	14	9	7
8	e1cfc87f543782b8a78b59fc85...	Garden tools	39	524	4	369	26	7	7
9	fd78812672eb8dd819fad93951...	HEALTH BEAUTY	60	630	2	187	12	7	8
10	3fddda7e92c16fa45033a4373a...	perfumery	45	650	1	75	11	11	8

Target Brazil – SQL Case Study (BigQuery)

By: Shivesh Raj Sahu

Initial Exploration

```sql
--All of the tables are correctly loaded and
--the dataset is present with correct column names

--Quick checklist
--1. Row counts per table (helps catch partial uploads)
SELECT 'customers' AS table, COUNT(*) AS row_count FROM `target_brazil.customers` UNION ALL
SELECT 'sellers' AS table, COUNT(*) AS row_count FROM `target_brazil.sellers` UNION ALL
SELECT 'orders' AS table, COUNT(*) AS row_count FROM `target_brazil.orders` UNION ALL
SELECT 'order_items' AS table, COUNT(*) AS row_count FROM `target_brazil.order_items` UNION ALL
SELECT 'payments' AS table, COUNT(*) AS row_count FROM `target_brazil.payments` UNION ALL
SELECT 'order_reviews' AS table, COUNT(*) AS row_count FROM `target_brazil.order_reviews` UNION ALL
SELECT 'geolocation' AS table, COUNT(*) AS row_count FROM `target_brazil.geolocation` UNION ALL
SELECT 'products' AS table, COUNT(*) AS row_count FROM `target_brazil.products`;


--  2.  Confirm column types ( especially about timestamps and numeric fields)
SELECT table_name, column_name, data_type
FROM `target_brazil`.INFORMATION_SCHEMA.COLUMNS
ORDER BY table_name, ordinal_position;
```

This script will process 252.67 MB when run.

Query results

Job information | Results | Visualization | JSON | Execution details | Execution graph

Row	table	row_count
1	geolocation	1000163
2	products	32951
3	order_reviews	99224
4	payments	103886
5	orders	99441
6	order_items	112650
7	customers	99441
8	sellers	3095

Target Brazil – SQL Case Study (BigQuery)
By: Shivesh Raj Sahu

```sql
-- 2. Confirm column types ( especially about timestamps and numeric fields)
SELECT table_name, column_name, data_type
FROM `target_brazil`.INFORMATION_SCHEMA.COLUMNS
ORDER BY table_name, ordinal_position;
```

This script will process 252.67 MB when run.

Query results

Job information | Results | Visualization | JSON | Execution details | Execution graph

Row	table_name	column_name	data_type
1	customers	customer_id	STRING
2	customers	customer_unique_id	STRING
3	customers	customer_zip_code_prefix	INT64
4	customers	customer_city	STRING
5	customers	customer_state	STRING
6	geolocation	geolocation_zip_code_prefix	INT64
7	geolocation	geolocation_lat	FLOAT64
8	geolocation	geolocation_lng	FLOAT64
9	geolocation	geolocation_city	STRING
10	geolocation	geolocation_state	STRING
11	order_items	order_id	STRING
12	order_items	order_item_id	INT64
13	order_items	product_id	STRING
14	order_items	seller_id	STRING
15	order_items	shipping_limit_date	TIMESTAMP
16	order_items	price	FLOAT64
17	order_items	freight_value	FLOAT64
18	order_reviews	review_id	STRING
19	order_reviews	order_id	STRING
20	order_reviews	review_score	INT64
21	order_reviews	review_comment_title	STRING
22	order_reviews	review_creation_date	TIMESTAMP
23	order_reviews	review_answer_timestamp	TIMESTAMP
24	orders	order_id	STRING
25	orders	customer_id	STRING
26	orders	order_status	STRING
27	orders	order_purchase_timestamp	TIMESTAMP
28	orders	order_approved_at	TIMESTAMP
29	orders	order_delivered_carrier_date	TIMESTAMP
30	orders	order_delivered_customer_date	TIMESTAMP
31	orders	order_estimated_delivery_date	TIMESTAMP
32	payments	order_id	STRING

Target Brazil – SQL Case Study (BigQuery)
By: Shivesh Raj Sahu

1.1 Data types of all columns in customers

Customers contains ID fields (STRING), location fields (STRING/INT64).

Types match the business meaning.

No type mismatches detected.

```sql
-- Now to start working on the Problem Statements


-- 1. Exploratory Analysis
-- 1.1 Data type of all columns in the "customers" table
-- Problem: Check the data types of all columns in the customers table.
SELECT
  table_name,
  column_name,
  data_type
FROM `target_brazil.INFORMATION_SCHEMA.COLUMNS`
WHERE table_name = 'customers';

```

Query completed

Query results — Save results — Open in

Job information | Results | Visualization | JSON | Execution details | Execution graph

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

1.2 Time range of orders

Orders span 2016-09-04 to 2018-10-17, covering two full years and partial years.

```sql
-- 1. Exploratory Analysis
-- 1.1 Data type of all columns in the "customers" table
-- Problem: Check the data types of all columns in the customers table.
SELECT
  table_name,
  column_name,
  data_type
FROM `target_brazil.INFORMATION_SCHEMA.COLUMNS`
WHERE table_name = 'customers';

-- 1.2 Get the time range between which the orders were placed
-- Problem: Find the earliest and latest purchase dates in the orders table.
SELECT
  MIN(order_purchase_timestamp) AS first_order_date,
  MAX(order_purchase_timestamp) AS last_order_date
FROM `target_brazil.orders`;

```

Query completed

Query results — Save results — Open in

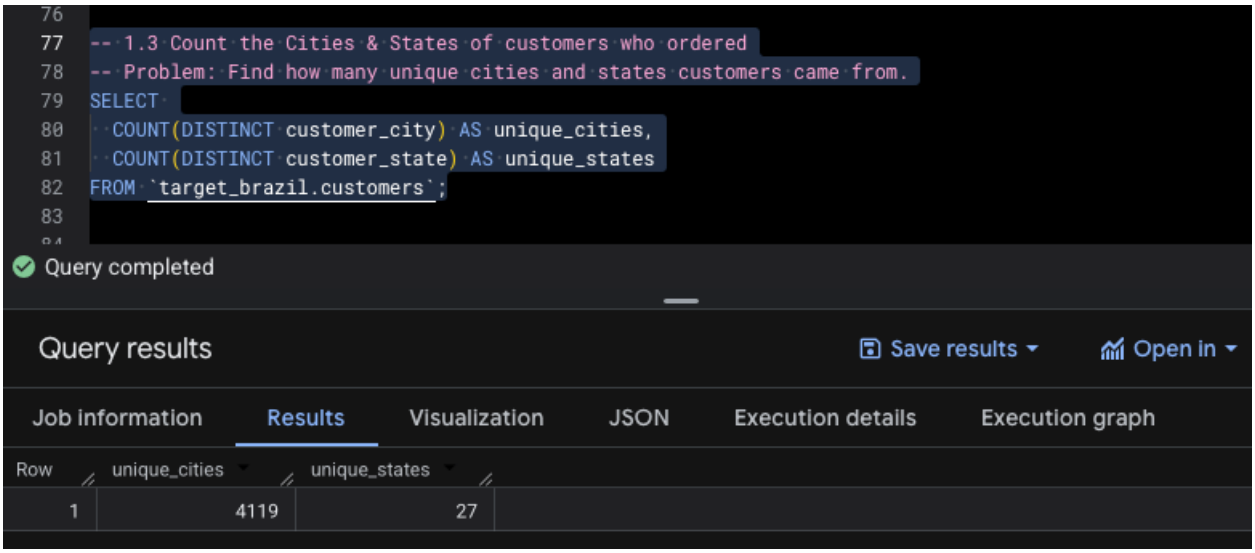
Job information | Results | Visualization | JSON | Execution details | Execution graph

Row	first_order_date	last_order_date
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

Target Brazil – SQL Case Study (BigQuery)
By: Shivesh Raj Sahu

1.3 How many cities and states did customers come from?

Orders come from 4,119 cities across 27 states, looks like a wide geographic coverage**.**



```
76
77  -- 1.3 Count the Cities & States of customers who ordered
78  -- Problem: Find how many unique cities and states customers came from.
79  SELECT
80    COUNT(DISTINCT customer_city) AS unique_cities,
81    COUNT(DISTINCT customer_state) AS unique_states
82  FROM `target_brazil.customers`;
83
```

Query completed

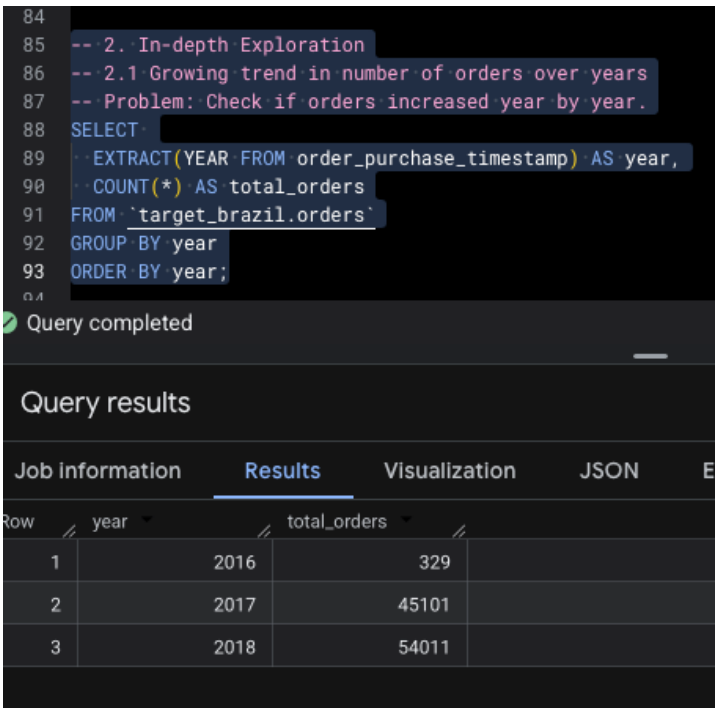
Query results — Save results — Open in

Job information | Results | Visualization | JSON | Execution details | Execution graph

Row	unique_cities	unique_states
1	4119	27

2.1 Year-over-year growth in orders

Orders grew sharply from 2017 to 2018, the numbers show ~20% increase.



```
84
85  -- 2. In-depth Exploration
86  -- 2.1 Growing trend in number of orders over years
87  -- Problem: Check if orders increased year by year.
88  SELECT
89    EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
90    COUNT(*) AS total_orders
91  FROM `target_brazil.orders`
92  GROUP BY year
93  ORDER BY year;
```

Query completed

Query results

Job information | Results | Visualization | JSON | E

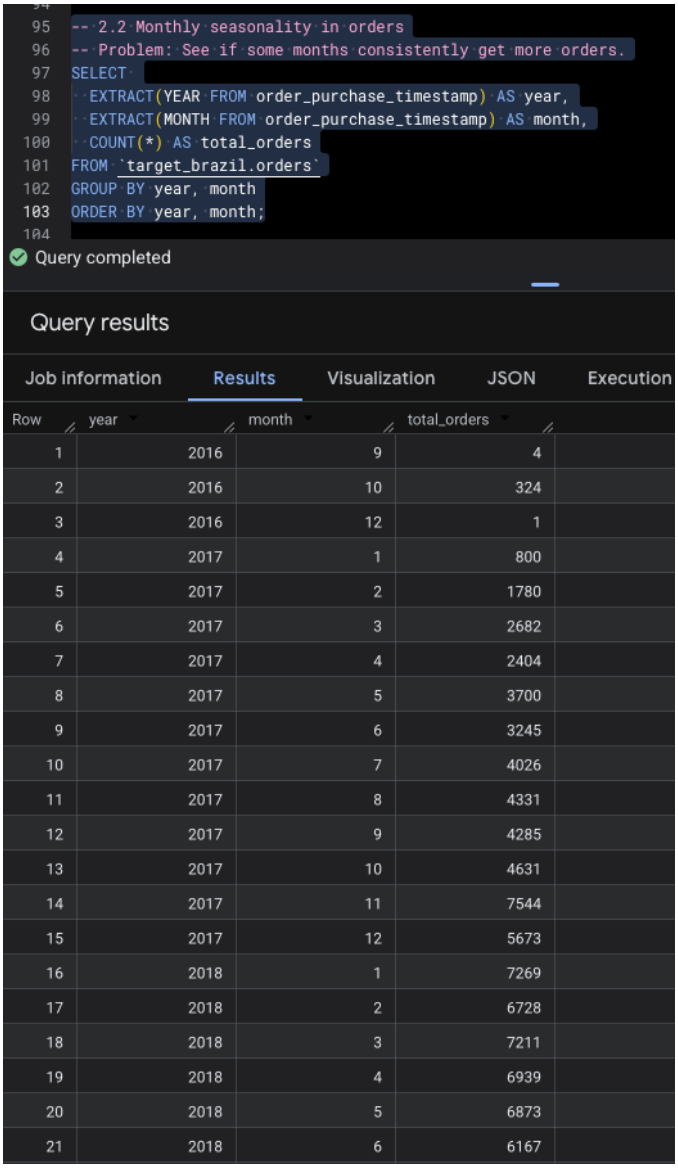
Row	year	total_orders
1	2016	329
2	2017	45101
3	2018	54011

Target Brazil – SQL Case Study (BigQuery)
By: Shivesh Raj Sahu

2.2 Monthly seasonality (year x month)

Mid-year months peak

Year - end softens slightly (at least based on monthly counts that I can see)



```sql
-- 2.2 Monthly seasonality in orders
-- Problem: See if some months consistently get more orders.
SELECT
  EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
  EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
  COUNT(*) AS total_orders
FROM `target_brazil.orders`
GROUP BY year, month
ORDER BY year, month;
```

Query completed

Query results

Job information | Results | Visualization | JSON | Execution

Row	year	month	total_orders
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
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
19	2018	4	6939
20	2018	5	6873
21	2018	6	6167

Target Brazil – SQL Case Study (BigQuery)
By: Shivesh Raj Sahu

2.3 Time of day when orders are placed

Most purchases occur in Afternoons and Night times, this will be useful for campaign timing.

```sql
-- 2.3 Time of day when orders are placed
-- Problem: Categorize orders into Dawn, Morning, Afternoon, Night.
SELECT
  CASE
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 0 AND 6 THEN 'Dawn'
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 7 AND 12 THEN 'Morning'
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 13 AND 18 THEN 'Afternoon'
    ELSE 'Night'
  END AS time_of_day,
  COUNT(*) AS total_orders
FROM `target_brazil.orders`
GROUP BY time_of_day
ORDER BY total_orders DESC;
```

Query completed

Query results

Save results | Open in

Job information | Results | Visualization | JSON | Execution details | Execution graph

Row	time_of_day	total_orders
1	Afternoon	38135
2	Night	28331
3	Morning	27733
4	Dawn	5242

Target Brazil – SQL Case Study (BigQuery)
By: Shivesh Raj Sahu

3.1 Month-on-month orders by state

SP/RJ dominate volume each month

Smaller northern states are more volatile.

```sql
-- 3) Evolution of E-commerce orders in Brazil
-- 3.1 Month-on-month number of orders placed in each state
-- Problem: For every month, how many orders came from each customer state?
SELECT
  EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
  EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
  c.customer_state AS state,
  COUNT(*) AS orders
FROM `target_brazil.orders` AS o
JOIN `target_brazil.customers` AS c
  ON o.customer_id = c.customer_id
GROUP BY year, month, state
ORDER BY year, month, state;
```

This query will process 7.59 MB when run.

Query results — Save results — Open in

Job information | Results | Visualization | JSON | Execution details | Execution graph

Row	year	month	state	orders
1	2016	9	RR	1
2	2016	9	RS	1
3	2016	9	SP	2
4	2016	10	AL	2
5	2016	10	BA	4
6	2016	10	CE	8
7	2016	10	DF	6
8	2016	10	ES	4
9	2016	10	GO	9
10	2016	10	MA	4
11	2016	10	MG	40
12	2016	10	MT	3
13	2016	10	PA	4
14	2016	10	PB	1
15	2016	10	PE	7
16	2016	10	PI	1
17	2016	10	PR	19
18	2016	10	RJ	56
19	2016	10	RN	4
20	2016	10	RR	1
21	2016	10	RS	24

Target Brazil – SQL Case Study (BigQuery)
By: Shivesh Raj Sahu

3.2 Customer distribution across states

Top states by unique customers: SP, RJ, MG … quiet a long tail beyond that

SP alone contributes ~ 40k unique customers vs less than 1k in smaller states

```sql
-- 3.2 Customer distribution across states
-- Problem: How many distinct customers are in each state?
SELECT
  customer_state AS state,
  COUNT(DISTINCT customer_unique_id) AS unique_customers
FROM `target_brazil.customers`
GROUP BY state
ORDER BY unique_customers DESC;
```

Query completed

Query results

Job information | Results | Visualization | JSON | Execut

Row	state	unique_customers
1	SP	40302
2	RJ	12384
3	MG	11259
4	RS	5277
5	PR	4882
6	SC	3534
7	BA	3277
8	DF	2075
9	ES	1964
10	GO	1952
11	PE	1609
12	CE	1313
13	PA	949
14	MT	876
15	MA	726
16	MS	694
17	PB	519
18	PI	482
19	RN	474
20	AL	401
21	SE	342
22	TO	273

Target Brazil – SQL Case Study (BigQuery)
By: Shivesh Raj Sahu

4.1 Percentage increase in total payments from 2017 -> 2018 (Jan-Aug)

Total payments rose about 136.98% from January to August 2017 to 2018, indicating strong growth

```sql
-- 4) Impact on the Economy
-- 4.1 % increase in order cost from 2017 -> 2018 (Jan–Aug only)
-- Problem: Compare total cost of orders (use payments.payment_value) across 2017 vs 2018, restricting to months 1-8.
WITH monthly AS (
  SELECT
    EXTRACT(YEAR FROM o.order_purchase_timestamp) AS yr,
    EXTRACT(MONTH FROM o.order_purchase_timestamp) AS mo,
    SUM(p.payment_value) AS total_paid
  FROM `target_brazil.orders` o
  JOIN `target_brazil.payments` p USING (order_id)
  WHERE EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8
  GROUP BY yr, mo
),
by_year AS (
  SELECT yr AS year, SUM(total_paid) AS total_paid
  FROM monthly
  GROUP BY yr
)
SELECT
  SUM(CASE WHEN year = 2017 THEN total_paid END) AS total_2017_JanAug,
  SUM(CASE WHEN year = 2018 THEN total_paid END) AS total_2018_JanAug,
  SAFE_DIVIDE(
    SUM(CASE WHEN year = 2018 THEN total_paid END) -
    SUM(CASE WHEN year = 2017 THEN total_paid END),
    SUM(CASE WHEN year = 2017 THEN total_paid END)
  ) * 100 AS pct_increase_JanAug_2018_vs_2017
FROM by_year;
```

Query completed

Query results

Save results | Open in

Job information | Results | Visualization | JSON | Execution details | Execution graph

Row	total_2017_JanAug	total_2018_JanAug	pct_increase_JanAug_2018_vs_2017
1	3669022.11999992	8694733.84000018	136.97687164667107

Target Brazil – SQL Case Study (BigQuery)
By: Shivesh Raj Sahu

4.2 Total & Average order price by state

Highest total order value in SP and RJ, average order value varies by state.

```sql
-- 4.2 Total & Average order price by state
-- Problem: For each state, compute total and average order price.
-- Order price here = sum of order_items.price for an order.
WITH order_price AS (
  SELECT
    oi.order_id,
    SUM(oi.price) AS order_price
  FROM `target_brazil.order_items` oi
  GROUP BY oi.order_id
)
SELECT
  c.customer_state AS state,
  SUM(op.order_price)      AS total_order_price,
  AVG(op.order_price)      AS avg_order_price,
  COUNT(DISTINCT op.order_id) AS orders_count
FROM order_price op
JOIN `target_brazil.orders`    o USING (order_id)
JOIN `target_brazil.customers` c ON o.customer_id = c.customer_id
GROUP BY state
ORDER BY total_order_price DESC;

```

This script will process 252.67 MB when run.

Query results — Save results

Job information | Results | Visualization | JSON | Execution details | Executi

Row	state	total_order_price	avg_order_price	orders_count
1	SP	5202955.049999...	125.7511794561...	41375
2	RJ	1824092.669999...	142.9315679360...	12762
3	MG	1585308.030000...	137.3274454261...	11544
4	RS	750304.0200000...	138.1266605301...	5432
5	PR	683083.7600000...	136.6714205682...	4998
6	SC	520553.3399999...	144.1177574750...	3612
7	BA	511349.9899999...	152.2781387730...	3358
8	DF	302603.9399999...	142.4018541176...	2125
9	GO	294591.9499999...	146.7822371699...	2007
10	ES	275037.3099999...	135.8208938271...	2025
11	PE	262788.0299999...	159.4587560679...	1648
12	CE	227254.7099999...	171.2544913338...	1327
13	PA	178947.8100000...	184.4822783505...	970
14	MT	156453.5299999...	173.2597231450...	903
15	MA	119648.2200000...	161.6867837837...	740
16	MS	116812.6400000...	164.7568970380...	709
17	PB	115268.0800000...	216.6693233082...	532
18	PI	86914.08000000...	176.2963083164...	493
19	RN	83034.98000000...	172.2717427385...	482
20	AL	80314.81000000...	195.4131630170...	411

Target Brazil – SQL Case Study (BigQuery)
By: Shivesh Raj Sahu

4.3 Total & Average freight by state

Total freight is highest where volume is highest, avg freight highlights logistics cost differences

```sql
-- 4.3 Total & Average freight by state
-- Problem: For each state, compute total and average freight cost.
WITH order_freight AS (
  SELECT
    oi.order_id,
    SUM(oi.freight_value) AS order_freight
  FROM `target_brazil.order_items` oi
  GROUP BY oi.order_id
)
SELECT
  c.customer_state AS state,
  SUM(ofr.order_freight)      AS total_freight,
  AVG(ofr.order_freight)      AS avg_freight,
  COUNT(DISTINCT ofr.order_id) AS orders_count
FROM order_freight ofr
JOIN `target_brazil.orders`    o USING (order_id)
JOIN `target_brazil.customers` c ON o.customer_id = c.customer_id
GROUP BY state
ORDER BY total_freight DESC;
```

Query completed

Query results — Save results

Job information | Results | Visualization | JSON | Execution details | Execution

Row	state	total_freight	avg_freight	orders_count
1	SP	718723.0700000...	17.37095033232...	41375
2	RJ	305589.3100000...	23.94525231154...	12762
3	MG	270853.4600000...	23.46270443520...	11544
4	RS	135522.7400000...	24.94895802650...	5432
5	PR	117851.6799999...	23.57976790716...	4998
6	BA	100156.6800000...	29.82628945801...	3358
7	SC	89660.25999999...	24.82288482834...	3612
8	PE	59449.65999999...	36.07382281553...	1648
9	GO	53114.98000000...	26.46486297957...	2007
10	DF	50625.50000000...	23.82376470588...	2125
11	ES	49764.59999999...	24.57511111111...	2025
12	CE	48351.59000000...	36.43676714393...	1327
13	PA	38699.30000000...	39.89618556701...	970
14	MA	31523.76999999...	42.59968918918...	740
15	MT	29715.42999999...	32.90745293466...	903
16	PB	25719.72999999...	48.34535714285...	532
17	PI	21218.2	43.03894523326...	493
18	MS	19144.03000000...	27.00145275035...	709
19	RN	18860.09999999...	39.12883817427...	482
20	AL	15914.59000000...	38.72163017031...	411

Target Brazil – SQL Case Study (BigQuery)
By: Shivesh Raj Sahu

5.1 Per-order delivery time & difference vs estimate (single query)

Most deliveries cluster around certain days

diff_estimate_days < 0 means earlier than promised

```sql
-- 5) Sales, Freight & Delivery Time
-- 5.1 Delivery time & difference vs estimated – per order (single query)
-- Problem: For each delivered order, compute:
--          time_to_deliver  = delivered_date - purchase_date (days)
--          diff_estimated_delivery = delivered_date - estimated_date (days)
SELECT
  o.order_id,
  c.customer_state AS state,
  DATE(o.order_purchase_timestamp)          AS purchase_date,
  DATE(o.order_delivered_customer_date)     AS delivered_date,
  DATE(o.order_estimated_delivery_date)     AS estimated_date,
  DATE_DIFF(DATE(o.order_delivered_customer_date),
            DATE(o.order_purchase_timestamp), DAY) AS time_to_deliver_days,
  DATE_DIFF(DATE(o.order_delivered_customer_date),
            DATE(o.order_estimated_delivery_date), DAY) AS diff_estimated_delivery_days
FROM `target_brazil.orders` o
JOIN `target_brazil.customers` c
  ON o.customer_id = c.customer_id
WHERE o.order_delivered_customer_date IS NOT NULL;
```

Query completed

Query results

Job information | Results | Visualization | JSON | Execution details | Execution graph

Row	order_id	state	purchase_date	delivered_date	estimated_date	time_to_deliver_d...	diff_estimated_d...
1	770d331c84e5b214bd9dc70a1...	RJ	2016-10-07	2016-10-14	2016-11-29	7	-46
2	dabf2b0e35b423f94618bf965f...	SP	2016-10-09	2016-10-16	2016-11-30	7	-45
3	8beb59392e21af5eb9547ae1a...	RJ	2016-10-08	2016-10-19	2016-11-30	11	-42
4	1950d777989f6a877539f53795...	MG	2018-02-19	2018-03-21	2018-03-09	30	12
5	bfbd0f9bdef84302105ad712db...	SP	2016-09-15	2016-11-09	2016-10-04	55	36
6	cd3b8574c82b42fc8129f6d502...	SP	2016-10-03	2016-10-14	2016-11-23	11	-40
7	31b0dd6152d2e471443debf03...	RJ	2016-10-05	2016-10-13	2016-11-23	8	-41
8	a8c3f65a43b2e956cbfca10db4...	SP	2016-10-04	2016-10-20	2016-11-24	16	-35
9	a041155864e51411164582913...	SP	2016-10-04	2016-10-13	2016-11-24	9	-42
10	50013835d7b14aefb45282586...	SP	2016-10-04	2016-10-21	2016-11-24	17	-34
11	1ff217aa612f6cd7c4255c9bfe9...	SP	2016-10-04	2016-10-24	2016-11-24	20	-31
12	6eb51c1c9c69f78ea9fd4665fb...	SP	2016-10-04	2016-10-13	2016-11-24	9	-42
13	d2d67121a09535b3a7b6a000e...	SP	2016-10-04	2016-10-13	2016-11-24	9	-42
14	5cc475c7c03290048eb2e742c...	SP	2016-10-04	2016-12-12	2016-11-24	69	18
15	0a0837a5eee9e7a9ce2b1fa831...	SP	2016-10-04	2016-10-22	2016-11-24	18	-33
16	85841af0d94e5d0cd3a9c0e42e...	SP	2016-10-05	2016-10-13	2016-11-25	8	-43
17	f716dffba1232aaef7c899fb8c1...	RJ	2016-10-07	2016-10-14	2016-11-25	7	-42
18	719628a909505f12f342015fd5...	SP	2016-10-05	2016-10-18	2016-11-25	13	-38
19	5c973d2b4652e1dec15353b59...	RJ	2016-10-07	2016-10-14	2016-11-25	7	-42
20	ca3b8093576a51bea61c9782c...	SP	2016-10-05	2016-10-24	2016-11-25	19	-32
21	92b44b87f1f7670b8911c5f0e6...	SP	2016-10-05	2016-10-13	2016-11-25	8	-43

Target Brazil – SQL Case Study (BigQuery)
By: Shivesh Raj Sahu

5.2 Top 5 states, highest & lowest average freight

Highest avg freight in RR, RO, ...

lowest in SP, MG, ... suggesting a denser networks

```sql
-- 5.2 Top 5 states – highest & lowest average freight
-- Problem: Rank states by average freight per order.
-- Query for the highest 5
WITH order_freight AS (
  SELECT order_id, SUM(freight_value) AS order_freight
  FROM `target_brazil.order_items`
  GROUP BY order_id
)
SELECT
  c.customer_state AS state,
  AVG(ofr.order_freight) AS avg_freight
FROM order_freight ofr
JOIN `target_brazil.orders`    o USING (order_id)
JOIN `target_brazil.customers` c ON o.customer_id = c.customer_id
GROUP BY state
ORDER BY avg_freight DESC
LIMIT 5;

-- Query for the lowest 5
-- same CTE as above
WITH order_freight AS (
  SELECT order_id, SUM(freight_value) AS order_freight
  FROM `target_brazil.order_items`
  GROUP BY order_id
)
SELECT
  c.customer_state AS state,
  AVG(ofr.order_freight) AS avg_freight
FROM order_freight ofr
JOIN `target_brazil.orders`    o USING (order_id)
JOIN `target_brazil.customers` c ON o.customer_id = c.customer_id
GROUP BY state
ORDER BY avg_freight ASC
LIMIT 5;
```

Query completed

Query results

Job information | Results | Visualization | JSON | Execution details | Execution graph

Row	state	avg_freight
1	RR	48.59108695652...
2	PB	48.34535714285...
3	RO	46.22421052631...
4	AC	45.51543209876...
5	PI	43.03894523326...

Target Brazil – SQL Case Study (BigQuery)

By: Shivesh Raj Sahu

```sql
-- 5.2 Top 5 states – highest & lowest average freight
-- Problem: Rank states by average freight per order.
-- Query for the highest 5
WITH order_freight AS (
  SELECT order_id, SUM(freight_value) AS order_freight
  FROM `target_brazil.order_items`
  GROUP BY order_id
)
SELECT
  c.customer_state AS state,
  AVG(ofr.order_freight) AS avg_freight
FROM order_freight ofr
JOIN `target_brazil.orders`    o USING (order_id)
JOIN `target_brazil.customers` c ON o.customer_id = c.customer_id
GROUP BY state
ORDER BY avg_freight DESC
LIMIT 5;

-- Query for the lowest 5
-- same CTE as above
WITH order_freight AS (
  SELECT order_id, SUM(freight_value) AS order_freight
  FROM `target_brazil.order_items`
  GROUP BY order_id
)
SELECT
  c.customer_state AS state,
  AVG(ofr.order_freight) AS avg_freight
FROM order_freight ofr
JOIN `target_brazil.orders`    o USING (order_id)
JOIN `target_brazil.customers` c ON o.customer_id = c.customer_id
GROUP BY state
ORDER BY avg_freight ASC
LIMIT 5;
```

Query completed

Query results

Job information | Results | Visualization | JSON | Execution details | Execution graph

Row	state	avg_freight
1	SP	17.37095033232...
2	MG	23.46270443520...
3	PR	23.57976790716...
4	DF	23.82376470588...
5	RJ	23.94525231154...

Target Brazil – SQL Case Study (BigQuery)
By: Shivesh Raj Sahu

5.3 Top 5 states, highest & lowest average delivery time

```sql
-- 5.3 Top 5 states – highest & lowest average delivery time
-- Problem: Rank states by average delivery days.
-- Query for the highest 5
WITH per_order AS (
  SELECT
    o.order_id, c.customer_state AS state,
    DATE_DIFF(DATE(o.order_delivered_customer_date),
              DATE(o.order_purchase_timestamp), DAY) AS time_to_deliver_days
  FROM `target_brazil.orders` o
  JOIN `target_brazil.customers` c ON o.customer_id = c.customer_id
  WHERE o.order_delivered_customer_date IS NOT NULL
)
SELECT
  state,
  AVG(time_to_deliver_days) AS avg_delivery_days
FROM per_order
GROUP BY state
ORDER BY avg_delivery_days DESC
LIMIT 5;

-- Query for the lowest 5
-- same CTE as above
WITH per_order AS (
  SELECT
    o.order_id, c.customer_state AS state,
    DATE_DIFF(DATE(o.order_delivered_customer_date),
              DATE(o.order_purchase_timestamp), DAY) AS time_to_deliver_days
  FROM `target_brazil.orders` o
  JOIN `target_brazil.customers` c ON o.customer_id = c.customer_id
  WHERE o.order_delivered_customer_date IS NOT NULL
)
SELECT
  state,
  AVG(time_to_deliver_days) AS avg_delivery_days
FROM per_order
GROUP BY state
ORDER BY avg_delivery_days ASC
LIMIT 5;
```

Query completed

Query results

Job information | Results | Visualization | JSON | Execution details | Execution graph

Row	state	avg_delivery_days
1	RR	29.34146341463...
2	AP	27.17910447761...
3	AM	26.35862068965...
4	AL	24.50125944584...
5	PA	23.72515856236...

Target Brazil – SQL Case Study (BigQuery)
By: Shivesh Raj Sahu

```sql
-- 5.3 Top 5 states – highest & lowest average delivery time
-- Problem: Rank states by average delivery days.
-- Query for the highest 5
WITH per_order AS (
  SELECT
    o.order_id, c.customer_state AS state,
    DATE_DIFF(DATE(o.order_delivered_customer_date),
              DATE(o.order_purchase_timestamp), DAY) AS time_to_deliver_days
  FROM `target_brazil.orders` o
  JOIN `target_brazil.customers` c ON o.customer_id = c.customer_id
  WHERE o.order_delivered_customer_date IS NOT NULL
)
SELECT
  state,
  AVG(time_to_deliver_days) AS avg_delivery_days
FROM per_order
GROUP BY state
ORDER BY avg_delivery_days DESC
LIMIT 5;

-- Query for the lowest 5
-- same CTE as above
WITH per_order AS (
  SELECT
    o.order_id, c.customer_state AS state,
    DATE_DIFF(DATE(o.order_delivered_customer_date),
              DATE(o.order_purchase_timestamp), DAY) AS time_to_deliver_days
  FROM `target_brazil.orders` o
  JOIN `target_brazil.customers` c ON o.customer_id = c.customer_id
  WHERE o.order_delivered_customer_date IS NOT NULL
)
SELECT
  state,
  AVG(time_to_deliver_days) AS avg_delivery_days
FROM per_order
GROUP BY state
ORDER BY avg_delivery_days ASC
LIMIT 5;
```

Query completed

Query results

Job information | Results | Visualization | JSON | Execution details | Execution graph

Row	state	avg_delivery_days
1	SP	8.700530929744...
2	PR	11.93804590696...
3	MG	11.94654337296...
4	DF	12.89903846153...
5	SC	14.90752748801...

Target Brazil – SQL Case Study (BigQuery)
By: Shivesh Raj Sahu

5.4 Top 5 states, earliest vs. estimate (negative = earlier)

AC/AP/… deliver earlier than promised on average, AC delivers ~ -2.3 days earlier than estimated

AL/MA/… tend to be late

```sql
-- 5.4 Top 5 states – fastest vs estimate
-- Problem: Where is delivery earliest relative to estimate?
-- Use average delivered - estimated; more negative = faster than estimate.
-- Query for the fastest relative to estimate
WITH per_order AS (
  SELECT
    c.customer_state AS state,
    DATE_DIFF(DATE(o.order_delivered_customer_date),
              DATE(o.order_estimated_delivery_date), DAY) AS diff_vs_estimate_days
  FROM `target_brazil.orders` o
  JOIN `target_brazil.customers` c ON o.customer_id = c.customer_id
  WHERE o.order_delivered_customer_date IS NOT NULL
    AND o.order_estimated_delivery_date IS NOT NULL
)
SELECT
  state,
  AVG(diff_vs_estimate_days) AS avg_diff_vs_estimate_days
FROM per_order
GROUP BY state
ORDER BY avg_diff_vs_estimate_days ASC   -- most negative first (earliest vs est.)
LIMIT 5;

-- Query for the worst relative to estimate, i.e., latest
-- same CTE as above
WITH per_order AS (
  SELECT
    c.customer_state AS state,
    DATE_DIFF(DATE(o.order_delivered_customer_date),
              DATE(o.order_estimated_delivery_date), DAY) AS diff_vs_estimate_days
  FROM `target_brazil.orders` o
  JOIN `target_brazil.customers` c ON o.customer_id = c.customer_id
  WHERE o.order_delivered_customer_date IS NOT NULL
    AND o.order_estimated_delivery_date IS NOT NULL
)
SELECT
  state,
  AVG(diff_vs_estimate_days) AS avg_diff_vs_estimate_days
FROM per_order
GROUP BY state
ORDER BY avg_diff_vs_estimate_days DESC  -- most positive (latest vs est.)
LIMIT 5;
```

Query completed

Query results

Job information | Results | Visualization | JSON | Execution details | Execution graph

Row	state	avg_diff_vs_esti...
1	AC	-20.72499999999...
2	RO	-20.10288065843...
3	AP	-19.68656716417...
4	AM	-19.56551724137...
5	RR	-17.29268292682...

Target Brazil – SQL Case Study (BigQuery)
By: Shivesh Raj Sahu

```sql

-- 5.4 Top 5 states – fastest vs estimate
-- Problem: Where is delivery earliest relative to estimate?
-- Use average delivered - estimated; more negative = faster than estimate.
-- Query for the fastest relative to estimate
WITH per_order AS (
  SELECT
    c.customer_state AS state,
    DATE_DIFF(DATE(o.order_delivered_customer_date),
              DATE(o.order_estimated_delivery_date), DAY) AS diff_vs_estimate_days
  FROM `target_brazil.orders` o
  JOIN `target_brazil.customers` c ON o.customer_id = c.customer_id
  WHERE o.order_delivered_customer_date IS NOT NULL
    AND o.order_estimated_delivery_date IS NOT NULL
)
SELECT
  state,
  AVG(diff_vs_estimate_days) AS avg_diff_vs_estimate_days
FROM per_order
GROUP BY state
ORDER BY avg_diff_vs_estimate_days ASC   -- most negative first (earliest vs est.)
LIMIT 5;

-- Query for the worst relative to estimate, i.e., latest
-- same CTE as above
WITH per_order AS (
  SELECT
    c.customer_state AS state,
    DATE_DIFF(DATE(o.order_delivered_customer_date),
              DATE(o.order_estimated_delivery_date), DAY) AS diff_vs_estimate_days
  FROM `target_brazil.orders` o
  JOIN `target_brazil.customers` c ON o.customer_id = c.customer_id
  WHERE o.order_delivered_customer_date IS NOT NULL
    AND o.order_estimated_delivery_date IS NOT NULL
)
SELECT
  state,
  AVG(diff_vs_estimate_days) AS avg_diff_vs_estimate_days
FROM per_order
GROUP BY state
ORDER BY avg_diff_vs_estimate_days DESC  -- most positive (latest vs est.)
LIMIT 5;
```

Query completed

Query results

Job information | **Results** | Visualization | JSON | Execution details | Execution graph

Row	state	avg_diff_vs_esti...
1	AL	-8.707808564231...
2	MA	-9.571827057182...
3	SE	-10.02089552238...
4	ES	-10.49624060150...
5	BA	-10.79453316953...

Target Brazil – SQL Case Study (BigQuery)
By: Shivesh Raj Sahu

6.1 Month-on-month number of orders by payment type

Payment mix is dominated by credit_card

Other types (boleto, voucher) are smaller but steady

```sql
-- 6) Payments Analysis
-- 6.1 Month-on-month number of orders by payment type
-- Problem: For every month, how many orders used each payment_type?
-- Count distinct orders per type to be safe.
SELECT
  EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
  EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
  p.payment_type,
  COUNT(DISTINCT p.order_id) AS orders
FROM `target_brazil.payments` p
JOIN `target_brazil.orders`   o USING (order_id)
GROUP BY year, month, payment_type
ORDER BY year, month, payment_type;
```

Query completed

Query results

Job information | Results | Visualization | JSON | Execution details | Execution graph

Row	year	month	payment_type	orders
1	2016	9	credit_card	3
2	2016	10	UPI	63
3	2016	10	credit_card	253
4	2016	10	debit_card	2
5	2016	10	voucher	11
6	2016	12	credit_card	1
7	2017	1	UPI	197
8	2017	1	credit_card	582
9	2017	1	debit_card	9
10	2017	1	voucher	33
11	2017	2	UPI	398
12	2017	2	credit_card	1347
13	2017	2	debit_card	13
14	2017	2	voucher	69
15	2017	3	UPI	590

6.2 Orders by payment installments

Most Brazilian e-commerce orders in this dataset are paid in full (1 installment) or split across 2 to 4 installments

```
365  -- 6.2 Orders by number of payment installments
366  -- Problem: How many orders used 1, 2, 3… installments?
367  SELECT
368    p.payment_installments,
369    COUNT(DISTINCT p.order_id) AS orders
370  FROM `target_brazil.payments` p
371  GROUP BY p.payment_installments
372  ORDER BY p.payment_installments;
373
374
375
376
```

Query completed

Query results

Job information | Results | Visualization | JSON | Execution details

Row	payment_installm...	orders
1	0	2
2	1	49060
3	2	12389
4	3	10443
5	4	7088
6	5	5234
7	6	3916
8	7	1623
9	8	4253
10	9	644
11	10	5315
12	11	23
13	12	133
14	13	16
15	14	15

Target Brazil – SQL Case Study (BigQuery)
By: Shivesh Raj Sahu

Final section

Actionable insights & recommendations

- Staff campaigns later in the day: Afternoon and Night contribute most orders, therefore prioritize push/email and support staffing in those windows.
- Regional logistics: States with highest avg freight and longest delivery times are prime targets for 3PL renegotiation or micro-fulfillment pilots.
- Promise accuracy: Where delivered is earlier than estimate (negative average), tighten ETA windows to improve conversion while keeping promise-keep high.
- Installments product: If many orders use more than 3 installments, surface installment messaging earlier in the funnel, negotiate card rates.
- State-specific growth: SP/RJ drive volume, smaller northern states show volatility, therefore test regional promos and inventory placement to smooth demand.