An Analysis of Brazilian E-Commerce Olist Store

Team Members

Arora, Jaideep

Fernando, Jithmi

Jaspreet Singh, Jaspreet Singh

Kumaresan, Santhoss

Massoun, Alisha

Raghavan, Siddarth

Singh, Gunpreet

Contents

Overview

Olist, a leading department store within the Brazilian marketplace, serves as a seamless bridge between small businesses across Brazil and various sales channels. Through a singular contract, these merchants can effortlessly showcase their products on the Olist Store and conveniently fulfill orders using Olist's network of logistics partners.

Following a customer's purchase from the Olist Store, the seller receives a notification to proceed with order fulfillment. Once the customer either receives the product or the estimated delivery date arrives, the customer is sent an email containing a satisfaction survey. This survey enables the customer to rate their purchasing experience and provide additional comments.

Dataset

The dataset has information of 100k orders from 2016 to 2018 made at multiple marketplaces in Brazil. Its features allows viewing an order from multiple dimensions: from order status, price, payment and freight performance to customer location, product attributes and finally reviews written by customers. There also is a geolocation dataset that relates Brazilian zip codes to lat/lng coordinates.

Dataset Link

Objectives

Product Category Analysis:

- Identify the top-performing product categories in terms of sales volume and revenue.
- Determine the least performing product categories and consider strategies to improve their sales.

Revenue Enhancement:

- Examine the revenue associated with individual product categories, states, and cities.
- Increase revenue by focusing on high-demand product categories and introducing marketing campaigns or promotions.
- Analyze customer preferences within specific categories and offer complementary products or bundles to enhance cross-selling.

Delivery Time Optimization:

- Investigate delivery time data to identify any patterns of delays or bottlenecks in the supply chain.
- Implement measures to minimize delivery delays, such as improving logistics, partnering with reliable shipping providers, or optimizing inventory management.

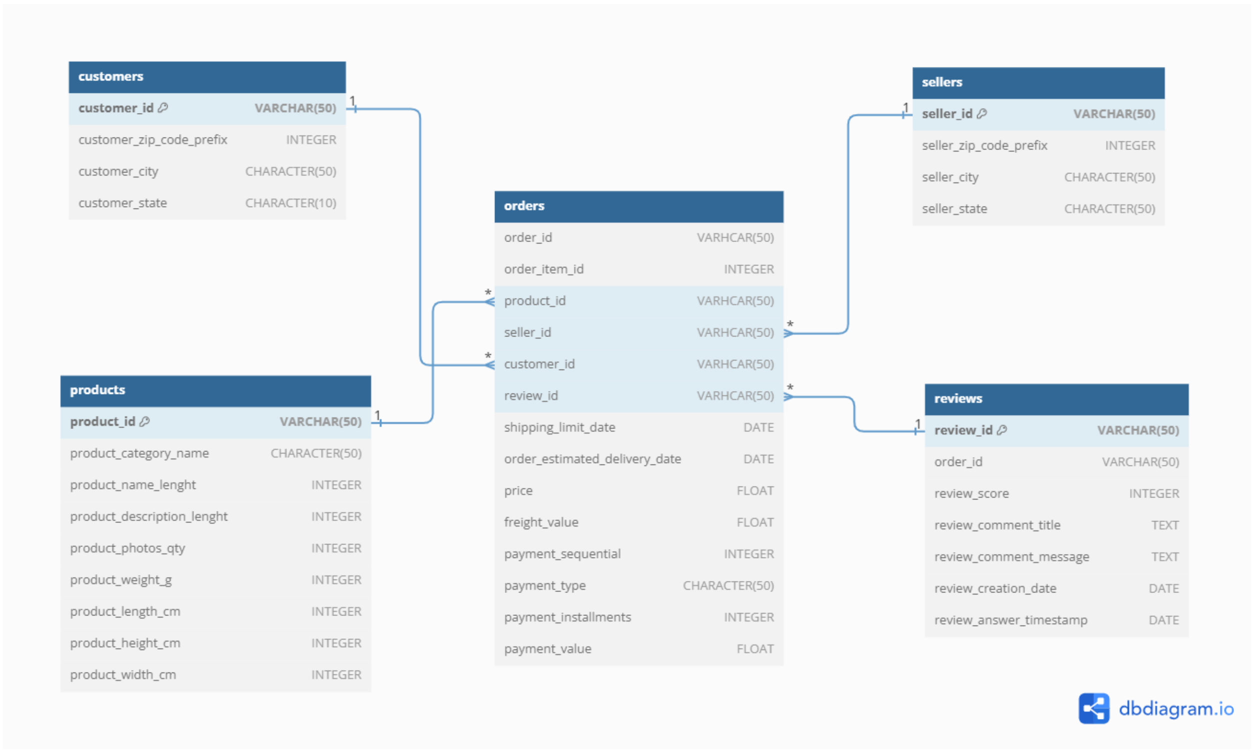
Customer Satisfaction Improvement:

- Examine customer reviews to gauge overall satisfaction levels and identify any recurring issues or concerns.

Geographical Analysis:

- Utilize the geolocation dataset to understand regional variations in product demand and delivery efficiency.
- Optimize distribution strategies based on the geographical concentration of orders and customer locations.

Relational Schema



customers	
customer_id	VARCHAR(50)
customer_zip_code_prefix	INTEGER
customer_city	CHARACTER(50)
customer_state	CHARACTER(10)

sellers	
seller_id	VARCHAR(50)
seller_zip_code_prefix	INTEGER
seller_city	CHARACTER(50)
seller_state	CHARACTER(50)

orders	
order_id	VARHCAR(50)
order_item_id	INTEGER
product_id	VARHCAR(50)
seller_id	VARHCAR(50)
customer_id	VARHCAR(50)
review_id	VARHCAR(50)
shipping_limit_date	DATE
order_estimated_delivery_date	DATE
price	FLOAT
freight_value	FLOAT
payment_sequential	INTEGER
payment_type	CHARACTER(50)
payment_installments	INTEGER
payment_value	FLOAT

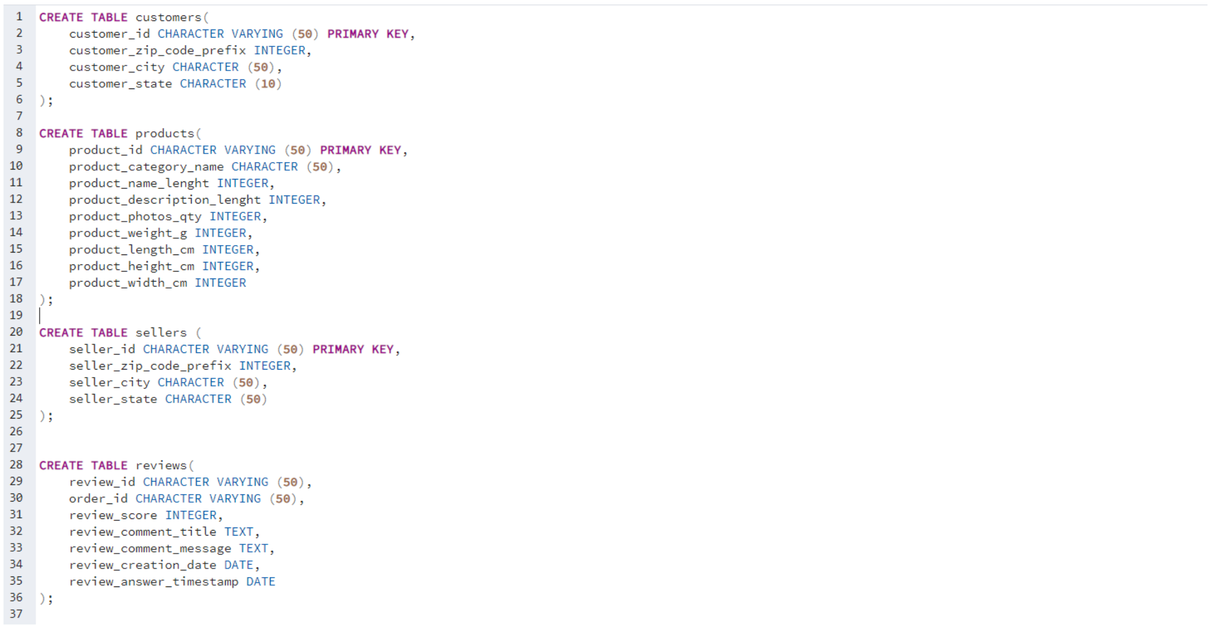
products	
product_id	VARCHAR(50)
product_category_name	CHARACTER(50)
product_name_lenght	INTEGER
product_description_lenght	INTEGER
product_photos_qty	INTEGER
product_weight_g	INTEGER
product_length_cm	INTEGER
product_height_cm	INTEGER
product_width_cm	INTEGER

reviews	
review_id	VARCHAR(50)
order_id	VARCHAR(50)
review_score	INTEGER
review_comment_title	TEXT
review_comment_message	TEXT
review_creation_date	DATE
review_answer_timestamp	DATE

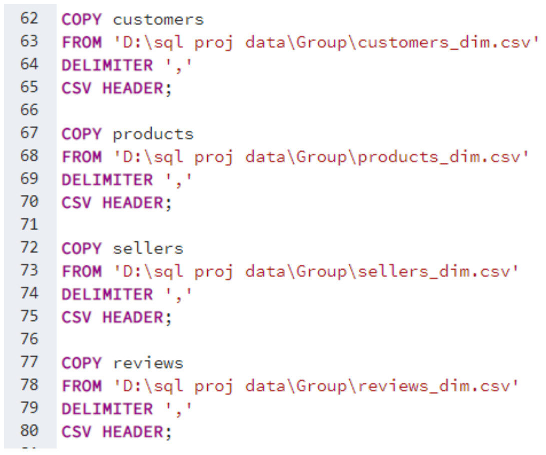
*The highlighted column in dimensional tables are **"Primary Key"** and highlighted columns in Orders table are **"Foreign Keys".***

Data Cleaning

- Creating Dimensional Tables and importing data into them. *(Appendix1)*

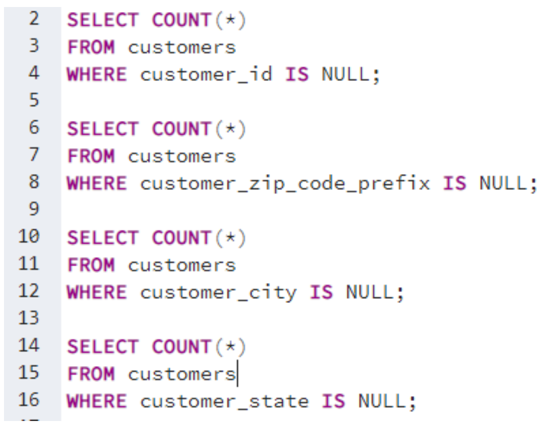


```
CREATE TABLE customers(
    customer_id CHARACTER VARYING (50) PRIMARY KEY,
    customer_zip_code_prefix INTEGER,
    customer_city CHARACTER (50),
    customer_state CHARACTER (10)
);

CREATE TABLE products(
    product_id CHARACTER VARYING (50) PRIMARY KEY,
    product_category_name CHARACTER (50),
    product_name_lenght INTEGER,
    product_description_lenght INTEGER,
    product_photos_qty INTEGER,
    product_weight_g INTEGER,
    product_length_cm INTEGER,
    product_height_cm INTEGER,
    product_width_cm INTEGER
);

CREATE TABLE sellers (
    seller_id CHARACTER VARYING (50) PRIMARY KEY,
    seller_zip_code_prefix INTEGER,
    seller_city CHARACTER (50),
    seller_state CHARACTER (50)
);


CREATE TABLE reviews(
    review_id CHARACTER VARYING (50),
    order_id CHARACTER VARYING (50),
    review_score INTEGER,
    review_comment_title TEXT,
    review_comment_message TEXT,
    review_creation_date DATE,
    review_answer_timestamp DATE
);

```

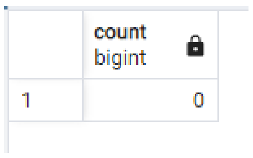


```
COPY customers
FROM 'D:\sql proj data\Group\customers_dim.csv'
DELIMITER ','
CSV HEADER;

COPY products
FROM 'D:\sql proj data\Group\products_dim.csv'
DELIMITER ','
CSV HEADER;

COPY sellers
FROM 'D:\sql proj data\Group\sellers_dim.csv'
DELIMITER ','
CSV HEADER;

COPY reviews
FROM 'D:\sql proj data\Group\reviews_dim.csv'
DELIMITER ','
CSV HEADER;
```

- Checking missing values for *Customers* Table



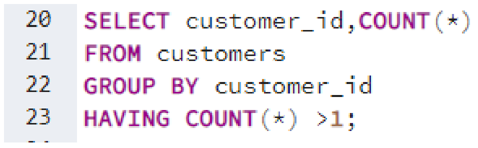
```sql
SELECT COUNT(*)
FROM customers
WHERE customer_id IS NULL;

SELECT COUNT(*)
FROM customers
WHERE customer_zip_code_prefix IS NULL;

SELECT COUNT(*)
FROM customers
WHERE customer_city IS NULL;

SELECT COUNT(*)
FROM customers
WHERE customer_state IS NULL;
```



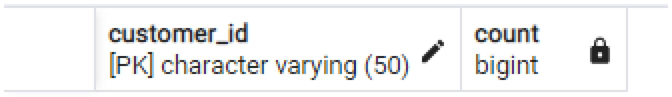
	count bigint
1	0

There are no missing values in customers table

- Checking duplicate values in customers table *(Appendix 2)*



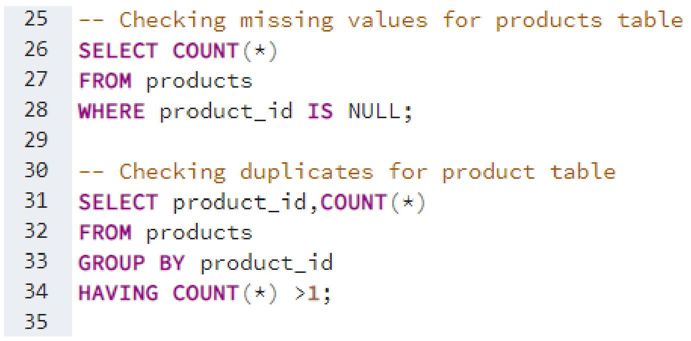
```sql
SELECT customer_id,COUNT(*)
FROM customers
GROUP BY customer_id
HAVING COUNT(*) >1;
```

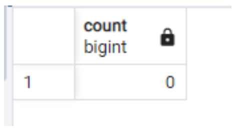


	customer_id [PK] character varying (50)	count bigint

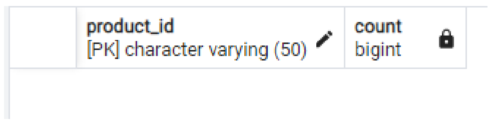
There are no duplicates in customer table

- Checking missing and duplicate values in Product table *(Appendix 3)*


```sql
-- Checking missing values for products table
SELECT COUNT(*)
FROM products
WHERE product_id IS NULL;

-- Checking duplicates for product table
SELECT product_id,COUNT(*)
FROM products
GROUP BY product_id
HAVING COUNT(*) >1;
```



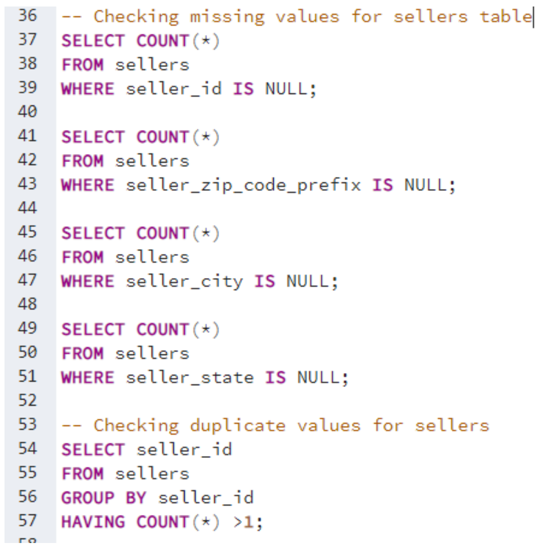
	count bigint
1	0

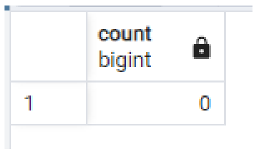


	product_id [PK] character varying (50)	count bigint

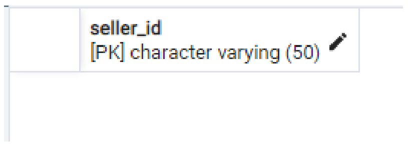
There are no missing or duplicate values in products table

- Checking missing and duplicate values in sellers table *(Appendix 4)*


```sql
-- Checking missing values for sellers table
SELECT COUNT(*)
FROM sellers
WHERE seller_id IS NULL;

SELECT COUNT(*)
FROM sellers
WHERE seller_zip_code_prefix IS NULL;

SELECT COUNT(*)
FROM sellers
WHERE seller_city IS NULL;

SELECT COUNT(*)
FROM sellers
WHERE seller_state IS NULL;

-- Checking duplicate values for sellers
SELECT seller_id
FROM sellers
GROUP BY seller_id
HAVING COUNT(*) >1;
```



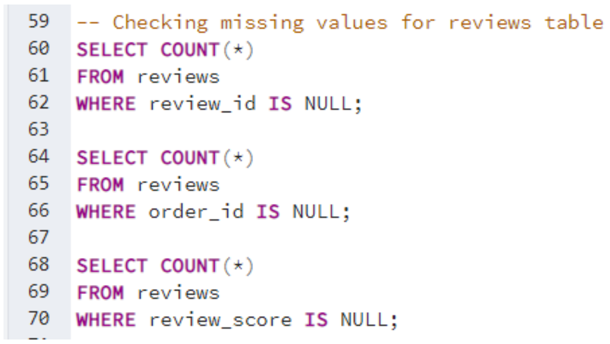
	count bigint
1	0



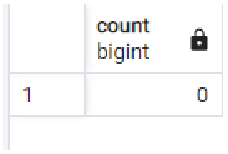
	seller_id [PK] character varying (50)

There are no missing or duplicate values in sellers table

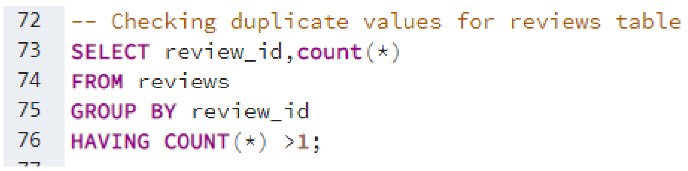
- Checking missing and duplicate values for reviews table *(Appendix 5)*



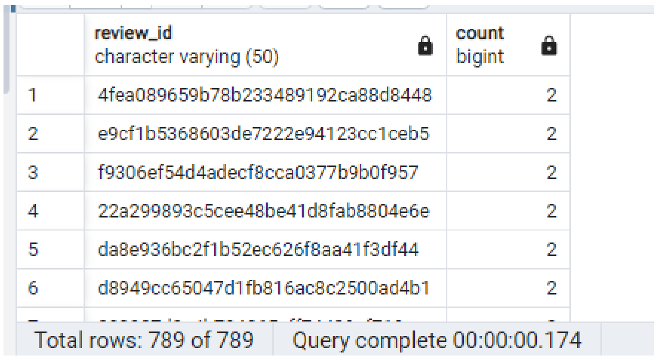
```sql
-- Checking missing values for reviews table
SELECT COUNT(*)
FROM reviews
WHERE review_id IS NULL;

SELECT COUNT(*)
FROM reviews
WHERE order_id IS NULL;

SELECT COUNT(*)
FROM reviews
WHERE review_score IS NULL;
```



	count bigint
1	0



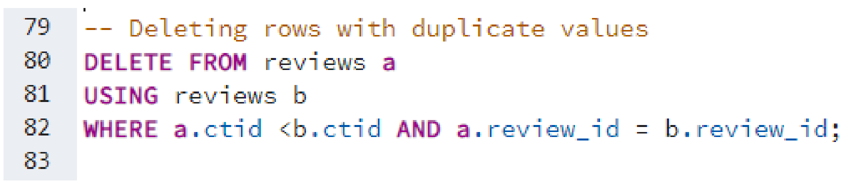
```sql
-- Checking duplicate values for reviews table
SELECT review_id,count(*)
FROM reviews
GROUP BY review_id
HAVING COUNT(*) >1;
```



	review_id character varying (50)	count bigint
1	4fea089659b78b233489192ca88d8448	2
2	e9cf1b5368603de7222e94123cc1ceb5	2
3	f9306ef54d4adecf8cca0377b9b0f957	2
4	22a299893c5cee48be41d8fab8804e6e	2
5	da8e936bc2f1b52ec626f8aa41f3df44	2
6	d8949cc65047d1fb816ac8c2500ad4b1	2

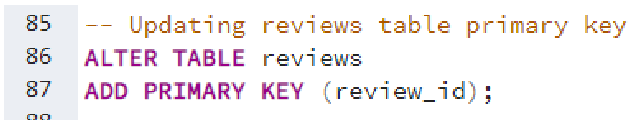
Total rows: 789 of 789 Query complete 00:00:00.174

From above results, we can see that reviews table have some duplicate “review_id” that ideally should be unique.
Now we will delete these duplicate rows

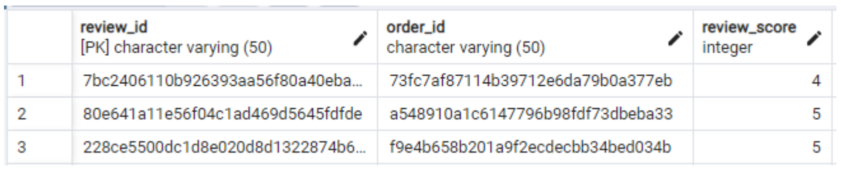


```sql
-- Deleting rows with duplicate values
DELETE FROM reviews a
USING reviews b
WHERE a.ctid <b.ctid AND a.review_id = b.review_id;
```

Now, “review_id” is a unique column, we will convert this column to primary key.

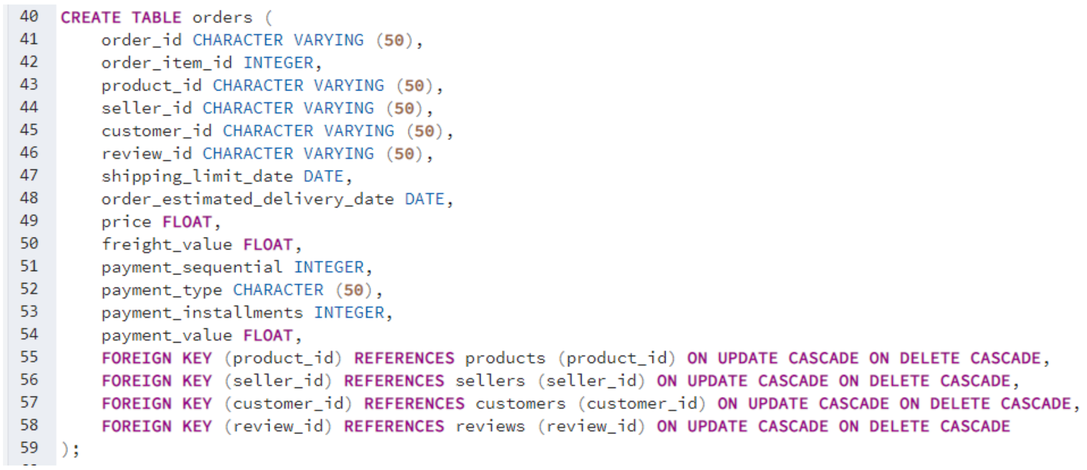


```sql
-- Updating reviews table primary key
ALTER TABLE reviews
ADD PRIMARY KEY (review_id);
```

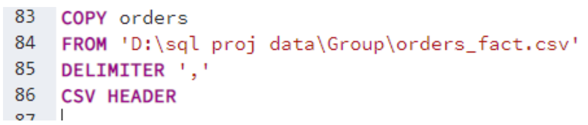


	review_id [PK] character varying (50)	order_id character varying (50)	review_score integer
1	7bc2406110b926393aa56f80a40eba...	73fc7af87114b39712e6da79b0a377eb	4
2	80e641a11e56f04c1ad469d5645fdfde	a548910a1c6147796b98fdf73dbeba33	5
3	228ce5500dc1d8e020d8d1322874b6...	f9e4b658b201a9f2ecdecbb34bed034b	5

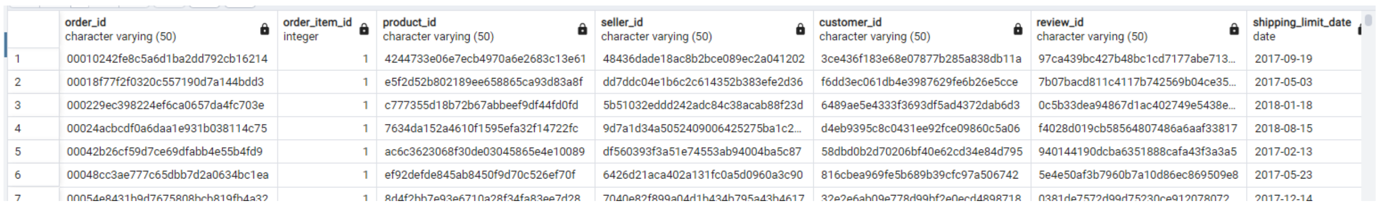
- Creating Orders fact table *(Appendix 6)*



```sql
CREATE TABLE orders (
    order_id CHARACTER VARYING (50),
    order_item_id INTEGER,
    product_id CHARACTER VARYING (50),
    seller_id CHARACTER VARYING (50),
    customer_id CHARACTER VARYING (50),
    review_id CHARACTER VARYING (50),
    shipping_limit_date DATE,
    order_estimated_delivery_date DATE,
    price FLOAT,
    freight_value FLOAT,
    payment_sequential INTEGER,
    payment_type CHARACTER (50),
    payment_installments INTEGER,
    payment_value FLOAT,
    FOREIGN KEY (product_id) REFERENCES products (product_id) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (seller_id) REFERENCES sellers (seller_id) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (customer_id) REFERENCES customers (customer_id) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (review_id) REFERENCES reviews (review_id) ON UPDATE CASCADE ON DELETE CASCADE
);
```

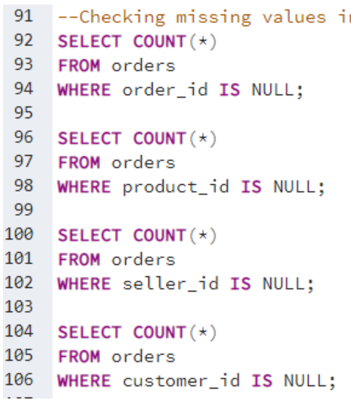


```sql
COPY orders
FROM 'D:\sql proj data\Group\orders_fact.csv'
DELIMITER ','
CSV HEADER
```



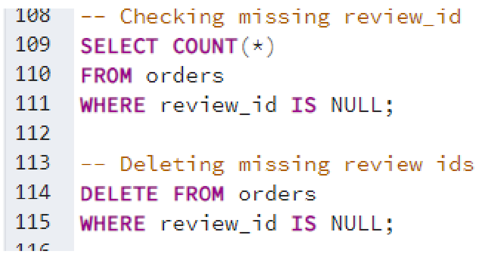
	order_id character varying (50)	order_item_id integer	product_id character varying (50)	seller_id character varying (50)	customer_id character varying (50)	review_id character varying (50)	shipping_limit_date date
1	00010242fe8c5a6d1ba2dd792cb16214	1	4244733e06e7ecb4970a6e2683c13e61	48436dade18ac8b2bce089ec2a041202	3ce436f183e68e07877b285a838db11a	97ca439bc427b48bc1cd7177abe713...	2017-09-19
2	00018f77f2f0320c557190d7a144bdd3	1	e5f2d52b802189ee658865ca93d83a8f	dd7ddc04e1b6c2c614352b383efe2d36	f6dd3ec061db4e3987629fe6b26e5cce	7b07bacd811c4117b742569b04ce35...	2017-05-03
3	000229ec398224ef6ca0657da4fc703e	1	c777355d18b72b67abbeef9df44fd0fd	5b51032eddd242adc84c38acab88f23d	6489ae5e4333f3693df5ad4372dab6d3	0c5b33dea94867d1ac402749e5438e...	2018-01-18
4	00024acbcdf0a6daa1e931b038114c75	1	7634da152a4610f1595efa32f14722fc	9d7a1d34a5052409006425275ba1c2...	d4eb9395c8c0431ee92fce09860c5a06	f4028d019cb58564807486a6aaf33817	2018-08-15
5	00042b26cf59d7ce69dfabb4e55b4fd9	1	ac6c3623068f30de03045865e4e10089	df560393f3a51e74553ab94004ba5c87	58dbd0b2d70206bf40e62cd34e84d795	940144190dcba6351888cafa43f3a3a5	2017-02-13
6	00048cc3ae777c65dbb7d2a0634bc1ea	1	ef92defde845ab8450f9d70c526ef70f	6426d21aca402a131fc0a5d0960a3c90	816cbea969fe5b689b39cfc97a506742	5e4e50af3b7960b7a10d86ec869509e8	2017-05-23
7	00054e8431b9d7675808bcb819fb4a32	1	8d4f2bb7e93e6710a28f34fa83ee7d28	7040e82f899a04d1b434b795a43b4617	32e2e6ab09e778d99bf2e0ecd4898718	0381de7572d99d75230ce912078072	2017-12-14

- Checking missing values in orders table *(Appendix 7)*

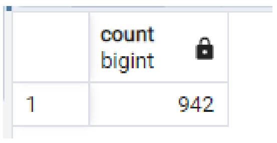


```sql
--Checking missing values in Orders Table
SELECT COUNT(*)
FROM orders
WHERE order_id IS NULL;

SELECT COUNT(*)
FROM orders
WHERE product_id IS NULL;

SELECT COUNT(*)
FROM orders
WHERE seller_id IS NULL;

SELECT COUNT(*)
FROM orders
WHERE customer_id IS NULL;
```

Columns – order_id, product_id, seller_id and customer_id have no missing values

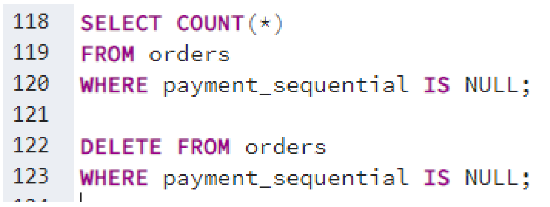


```sql
-- Checking missing review_id
SELECT COUNT(*)
FROM orders
WHERE review_id IS NULL;

-- Deleting missing review ids
DELETE FROM orders
WHERE review_id IS NULL;
```

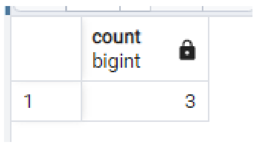


	count bigint
1	942

Review_id has 942 missing values and before moving ahead with analysis, we remove these values



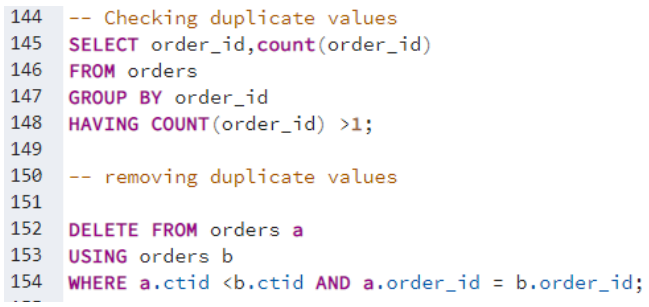
```sql
SELECT COUNT(*)
FROM orders
WHERE payment_sequential IS NULL;

DELETE FROM orders
WHERE payment_sequential IS NULL;
```



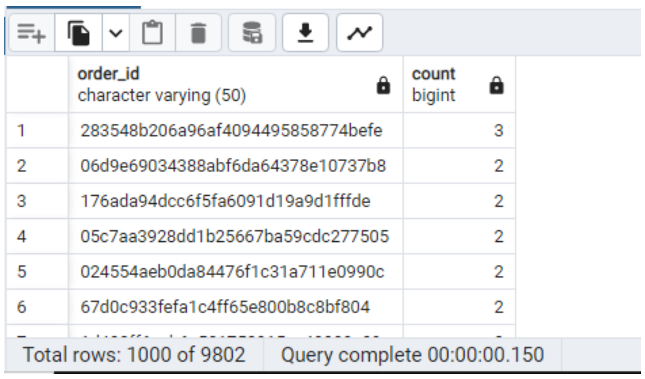
	count bigint
1	3

Similarly, we remove missing values in payment_sequential also.

- Checking duplicates in orders table *(Appendix 8)*



```sql
-- Checking duplicate values
SELECT order_id,count(order_id)
FROM orders
GROUP BY order_id
HAVING COUNT(order_id) >1;

-- removing duplicate values

DELETE FROM orders a
USING orders b
WHERE a.ctid <b.ctid AND a.order_id = b.order_id;
```



	order_id character varying (50)	count bigint
1	283548b206a96af4094495858774befe	3
2	06d9e69034388abf6da64378e10737b8	2
3	176ada94dcc6f5fa6091d19a9d1fffde	2
4	05c7aa3928dd1b25667ba59cdc277505	2
5	024554aeb0da84476f1c31a711e0990c	2
6	67d0c933fefa1c4ff65e800b8c8bf804	2

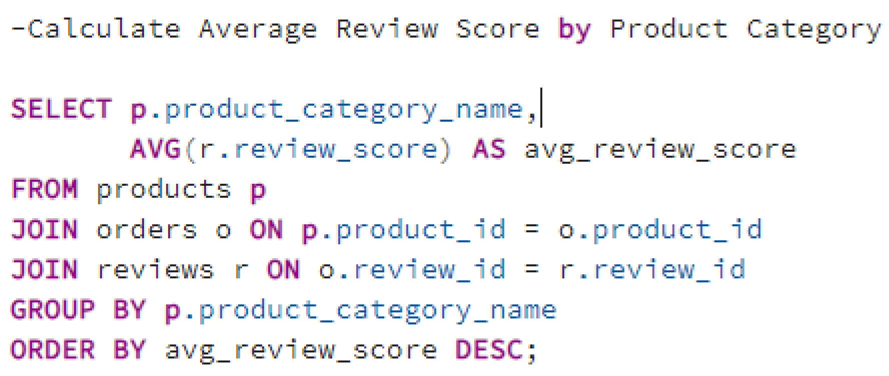
Total rows: 1000 of 9802 Query complete 00:00:00.150

There are lot of duplicates in order_id, for better analysis we will delete these.

Product Category Analysis

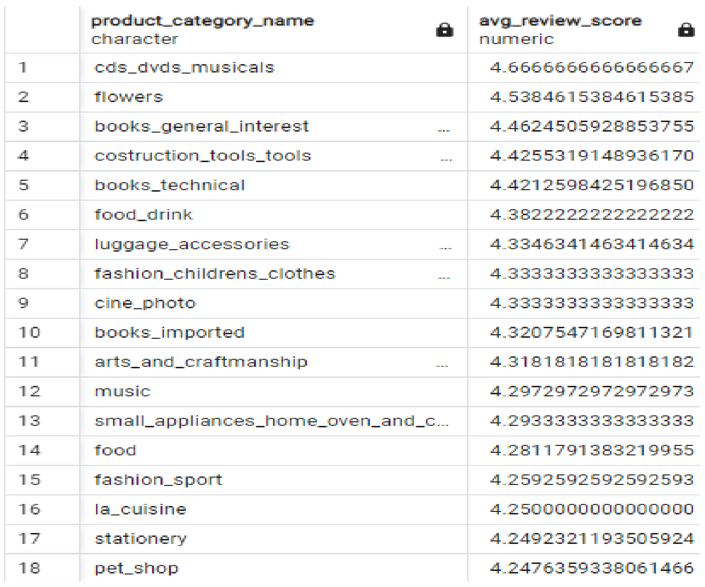
- **Average Review Score by Product Category*(Appendix 9)***

Query

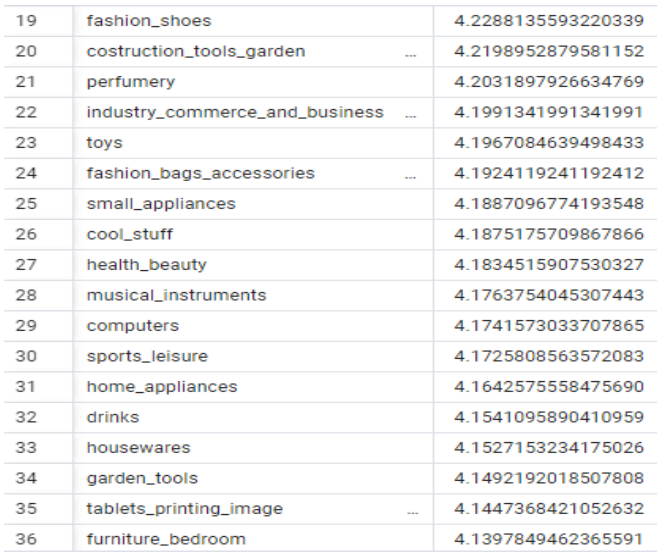


```
-Calculate Average Review Score by Product Category

SELECT p.product_category_name,
       AVG(r.review_score) AS avg_review_score
FROM products p
JOIN orders o ON p.product_id = o.product_id
JOIN reviews r ON o.review_id = r.review_id
GROUP BY p.product_category_name
ORDER BY avg_review_score DESC;
```

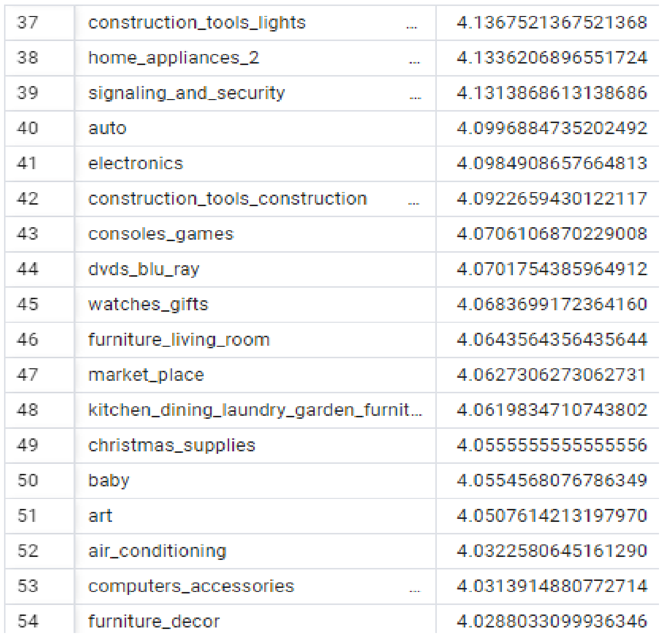
Output



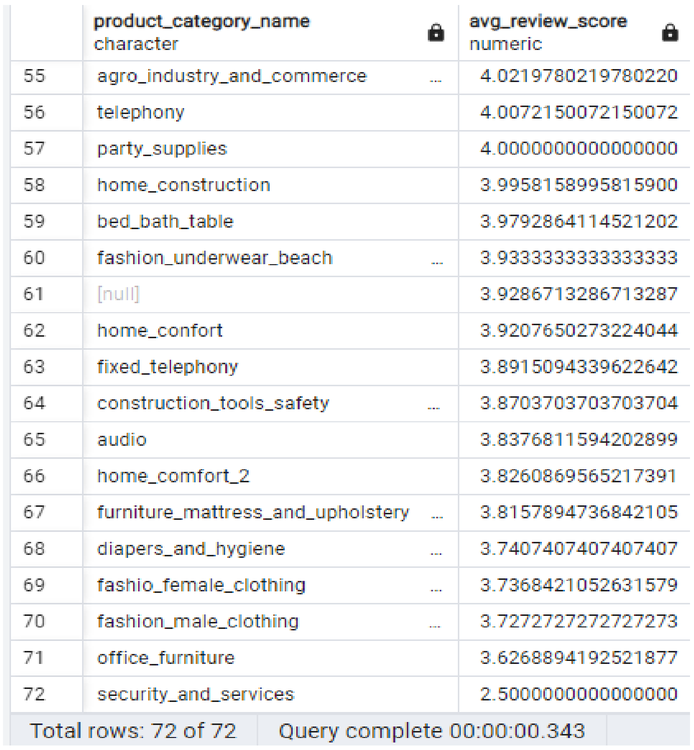
	product_category_name character	avg_review_score numeric
1	cds_dvds_musicals	4.6666666666666667
2	flowers	4.5384615384615385
3	books_general_interest	4.4624505928853755
4	costruction_tools_tools	4.4255319148936170
5	books_technical	4.4212598425196850
6	food_drink	4.3822222222222222
7	luggage_accessories	4.3346341463414634
8	fashion_childrens_clothes	4.3333333333333333
9	cine_photo	4.3333333333333333
10	books_imported	4.3207547169811321
11	arts_and_craftmanship	4.3181818181818182
12	music	4.2972972972972973
13	small_appliances_home_oven_and_c...	4.2933333333333333
14	food	4.2811791383219955
15	fashion_sport	4.2592592592592593
16	la_cuisine	4.2500000000000000
17	stationery	4.2492321193505924
18	pet_shop	4.2476359338061466



19	fashion_shoes	4.2288135593220339
20	costruction_tools_garden	4.2198952879581152
21	perfumery	4.2031897926634769
22	industry_commerce_and_business	4.1991341991341991
23	toys	4.1967084639498433
24	fashion_bags_accessories	4.1924119241192412
25	small_appliances	4.1887096774193548
26	cool_stuff	4.1875175709867866
27	health_beauty	4.1834515907530327
28	musical_instruments	4.1763754045307443
29	computers	4.1741573033707865
30	sports_leisure	4.1725808563572083
31	home_appliances	4.1642575558475690
32	drinks	4.1541095890410959
33	housewares	4.1527153234175026
34	garden_tools	4.1492192018507808
35	tablets_printing_image	4.1447368421052632
36	furniture_bedroom	4.1397849462365591



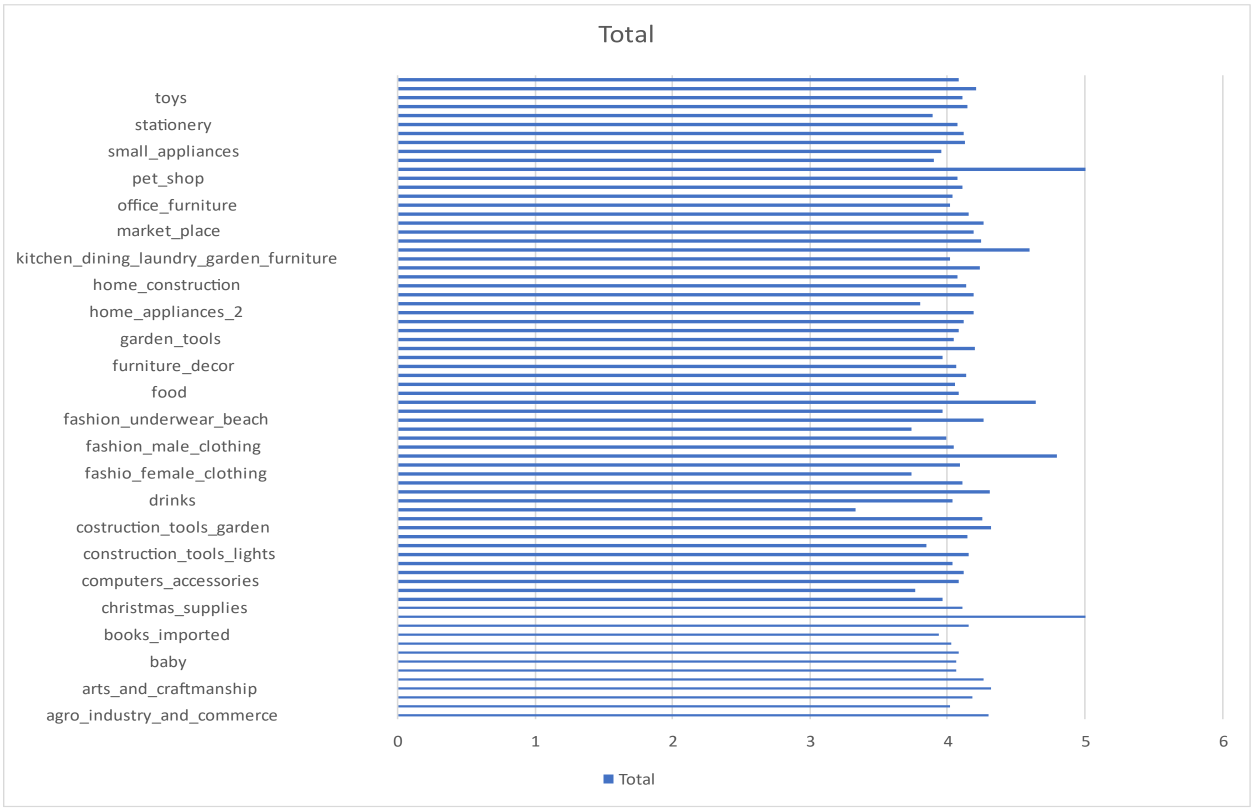
37	construction_tools_lights	4.1367521367521368
38	home_appliances_2	4.1336206896551724
39	signaling_and_security	4.1313868613138686
40	auto	4.0996884735202492
41	electronics	4.0984908657664813
42	construction_tools_construction	4.0922659430122117
43	consoles_games	4.0706106870229008
44	dvds_blu_ray	4.0701754385964912
45	watches_gifts	4.0683699172364160
46	furniture_living_room	4.0643564356435644
47	market_place	4.0627306273062731
48	kitchen_dining_laundry_garden_furnit...	4.0619834710743802
49	christmas_supplies	4.0555555555555556
50	baby	4.0554568076786349
51	art	4.0507614213197970
52	air_conditioning	4.0322580645161290
53	computers_accessories	4.0313914880772714
54	furniture_decor	4.0288033099936346



	product_category_name character	avg_review_score numeric
55	agro_industry_and_commerce	4.0219780219780220
56	telephony	4.0072150072150072
57	party_supplies	4.0000000000000000
58	home_construction	3.9958158995815900
59	bed_bath_table	3.9792864114521202
60	fashion_underwear_beach	3.9333333333333333
61	[null]	3.9286713286713287
62	home_confort	3.9207650273224044
63	fixed_telephony	3.8915094339622642
64	construction_tools_safety	3.8703703703703704
65	audio	3.8376811594202899
66	home_comfort_2	3.8260869565217391
67	furniture_mattress_and_upholstery	3.8157894736842105
68	diapers_and_hygiene	3.7407407407407407
69	fashio_female_clothing	3.7368421052631579
70	fashion_male_clothing	3.7272727272727273
71	office_furniture	3.6268894192521877
72	security_and_services	2.5000000000000000

Total rows: 72 of 72   Query complete 00:00:00.343

Visualization

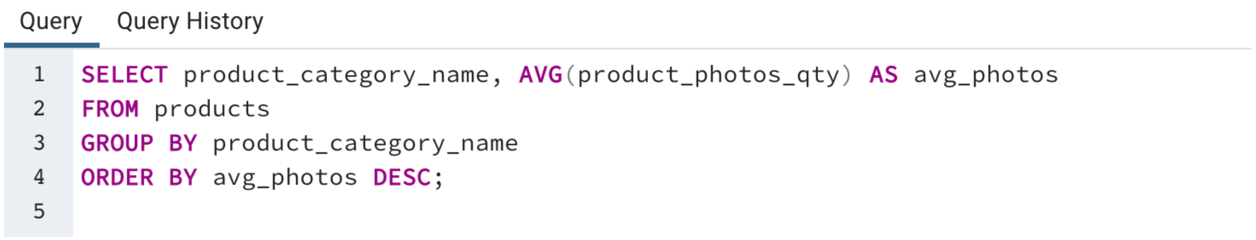


Summary

CDs and DVDs have the highest average review score among the various product categories, followed by flowers in second place. On the other hand, security and services received the lowest average review scores. Interestingly, both male and female fashion categories also garnered notably low average scores. Nonetheless, the majority of product categories still maintain an average review score of 4.0 or higher.

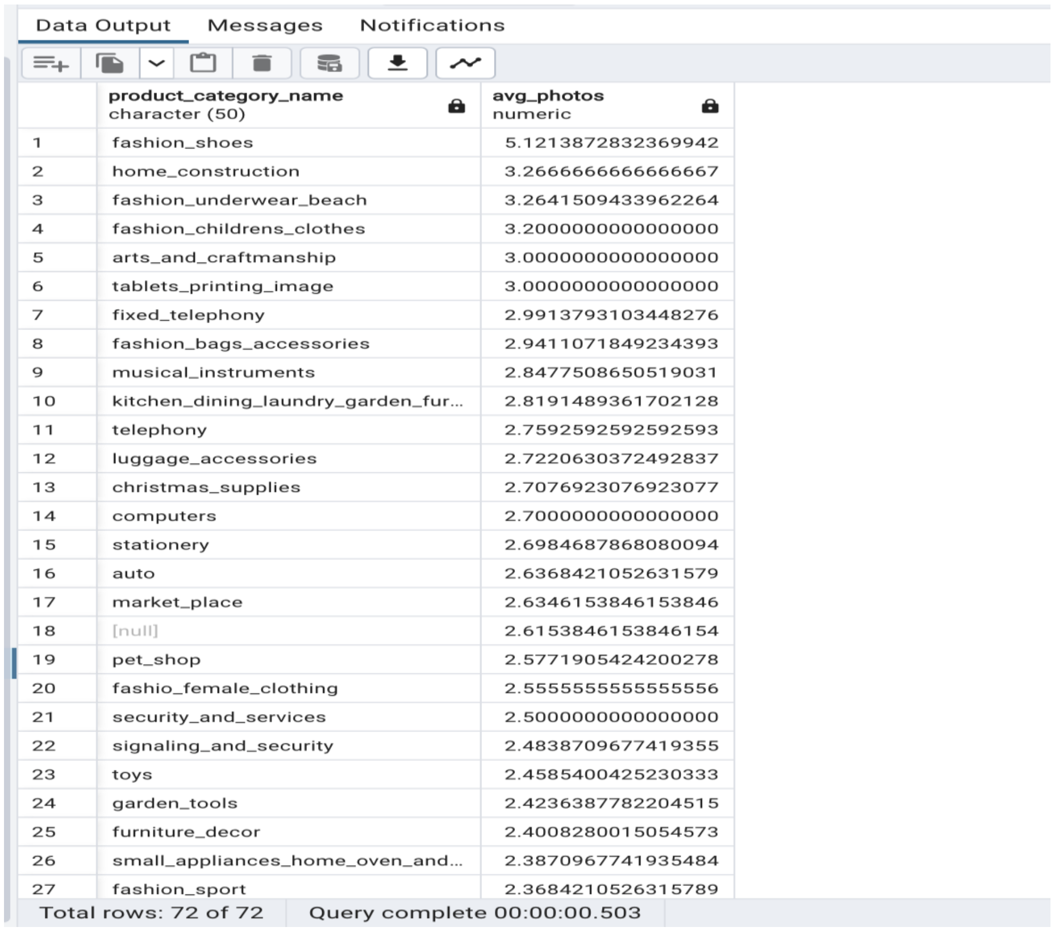
- **Product categories that have the highest average number of product photos*(Appendix 10)***

Query



```sql
SELECT product_category_name, AVG(product_photos_qty) AS avg_photos
FROM products
GROUP BY product_category_name
ORDER BY avg_photos DESC;
```

Output



	product_category_name character (50)	avg_photos numeric
1	fashion_shoes	5.1213872832369942
2	home_construction	3.2666666666666667
3	fashion_underwear_beach	3.2641509433962264
4	fashion_childrens_clothes	3.2000000000000000
5	arts_and_craftmanship	3.0000000000000000
6	tablets_printing_image	3.0000000000000000
7	fixed_telephony	2.9913793103448276
8	fashion_bags_accessories	2.9411071849234393
9	musical_instruments	2.8477508650519031
10	kitchen_dining_laundry_garden_fur...	2.8191489361702128
11	telephony	2.7592592592592593
12	luggage_accessories	2.7220630372492837
13	christmas_supplies	2.7076923076923077
14	computers	2.7000000000000000
15	stationery	2.6984687868080094
16	auto	2.6368421052631579
17	market_place	2.6346153846153846
18	[null]	2.6153846153846154
19	pet_shop	2.5771905424200278
20	fashio_female_clothing	2.5555555555555556
21	security_and_services	2.5000000000000000
22	signaling_and_security	2.4838709677419355
23	toys	2.4585400425230333
24	garden_tools	2.4236387782204515
25	furniture_decor	2.4008280015054573
26	small_appliances_home_oven_and...	2.3870967741935484
27	fashion_sport	2.3684210526315789

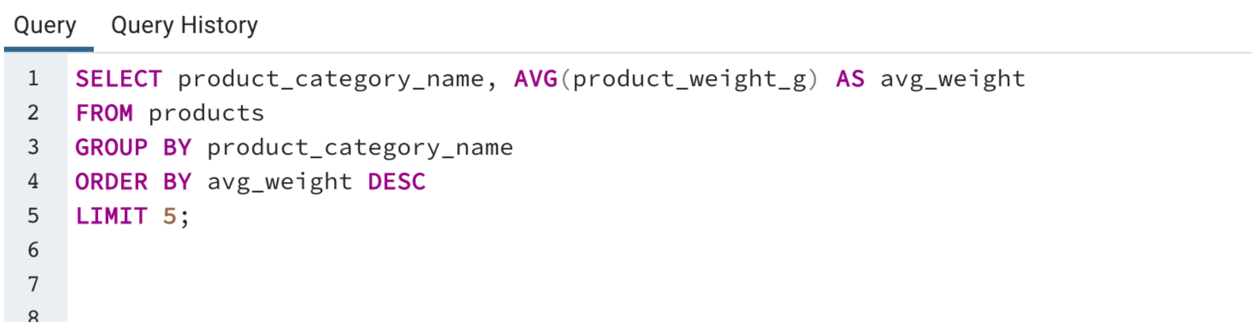
Total rows: 72 of 72   Query complete 00:00:00.503

Summary

Generally, the fashion categories exhibit a higher average number of photos per product. Nonetheless, the review scores for these categories are relatively lower compared to CDs and DVDs, which contain fewer photos.

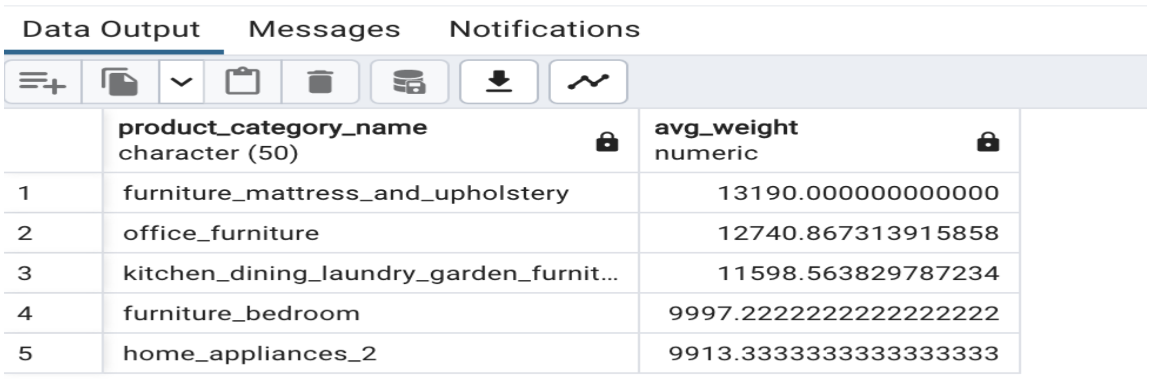
- **Top 5 product categories with the highest average product weights*(Appendix 11)***

Query



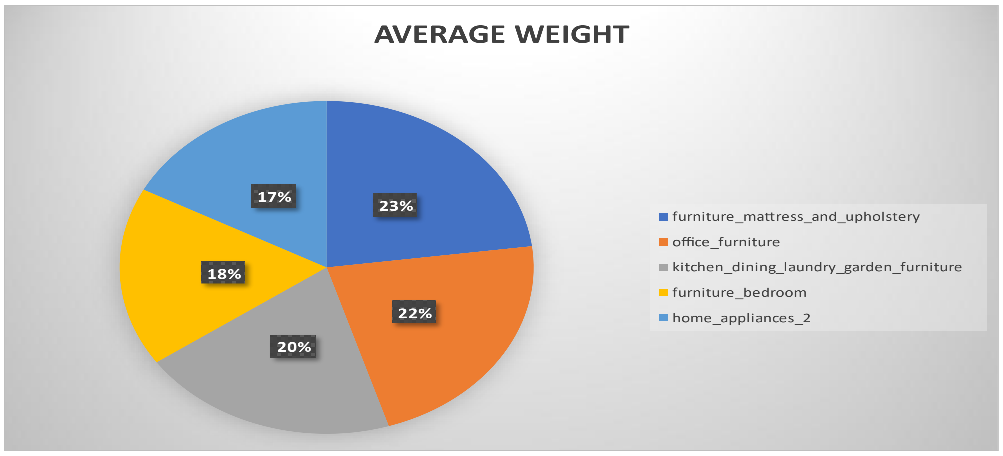
```sql
SELECT product_category_name, AVG(product_weight_g) AS avg_weight
FROM products
GROUP BY product_category_name
ORDER BY avg_weight DESC
LIMIT 5;
```

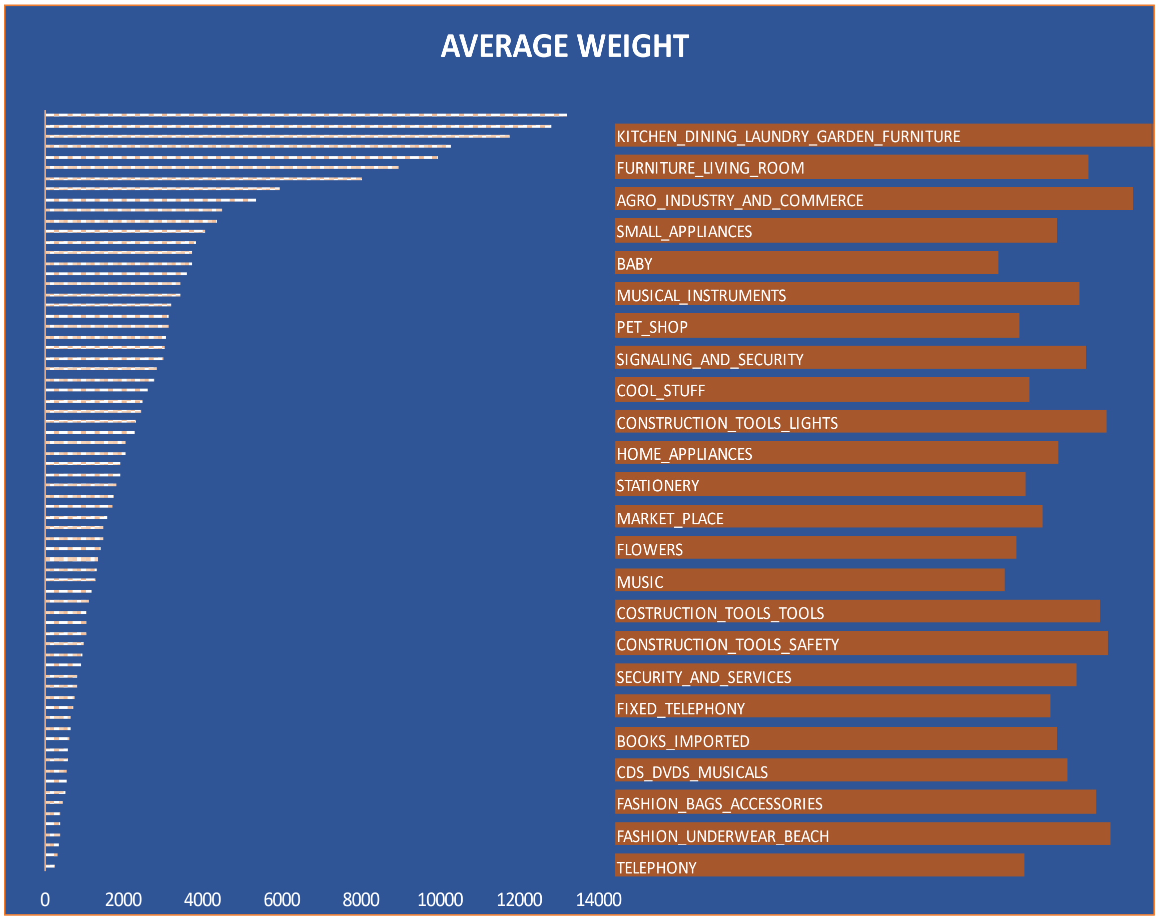
Output



	product_category_name character (50)	avg_weight numeric
1	furniture_mattress_and_upholstery	13190.000000000000
2	office_furniture	12740.867313915858
3	kitchen_dining_laundry_garden_furnit...	11598.563829787234
4	furniture_bedroom	9997.2222222222222222
5	home_appliances_2	9913.3333333333333333

Visualization





Summary

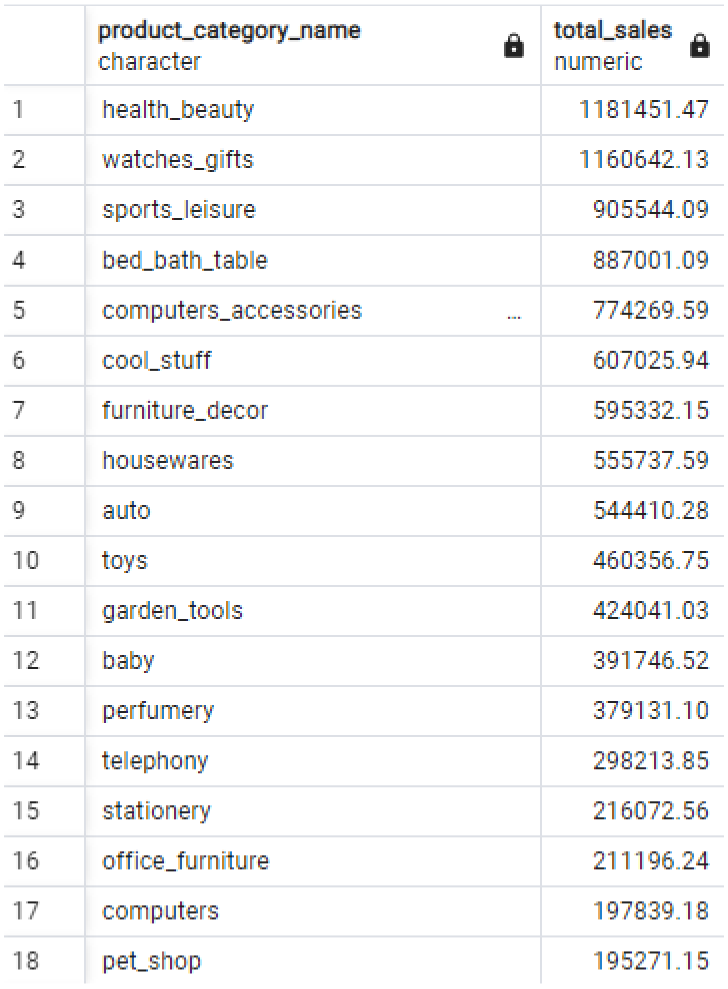
Most furniture items exhibit an elevated average weight, with mattresses and upholstery ranking as the heaviest among them.

- **Product categories with the highest sale*(Appendix 12)***

Query

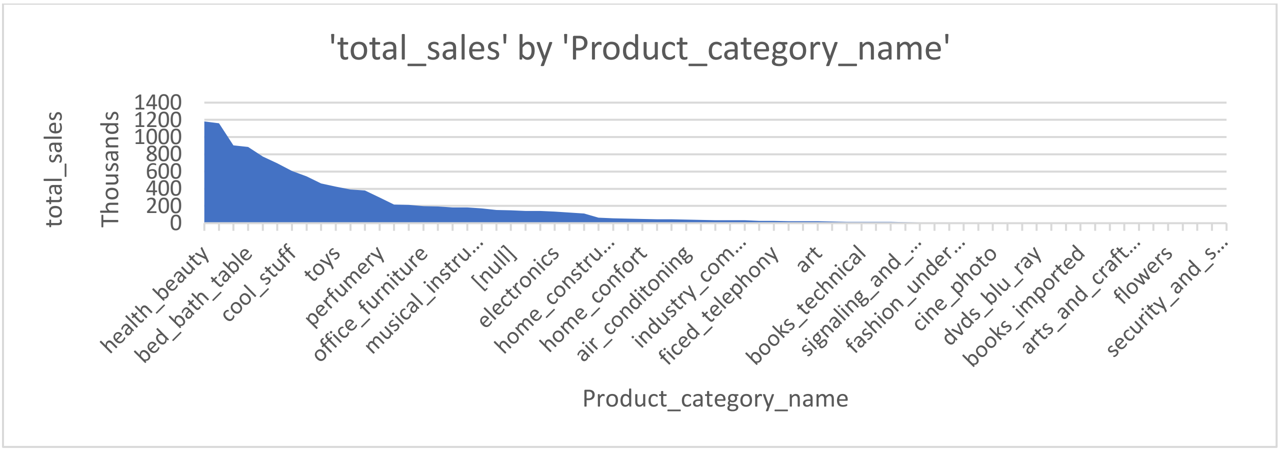
```
-- Total Revenue by State

SELECT c.customer_state,
       ROUND(SUM(o.price::NUMERIC), 2) AS total_revenue
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_state
ORDER BY total_revenue DESC;
```

Output



	product_category_name character	total_sales numeric
1	health_beauty	1181451.47
2	watches_gifts	1160642.13
3	sports_leisure	905544.09
4	bed_bath_table	887001.09
5	computers_accessories	774269.59
6	cool_stuff	607025.94
7	furniture_decor	595332.15
8	housewares	555737.59
9	auto	544410.28
10	toys	460356.75
11	garden_tools	424041.03
12	baby	391746.52
13	perfumery	379131.10
14	telephony	298213.85
15	stationery	216072.56
16	office_furniture	211196.24
17	computers	197839.18
18	pet_shop	195271.15

Visualization



Within the product categories, "health_beauty" and "bed_bath_table" exhibit the highest sales, while "fashion_childrens_clothes" and "security_and_services" demonstrate notably lower sales, showcasing a substantial contrast.

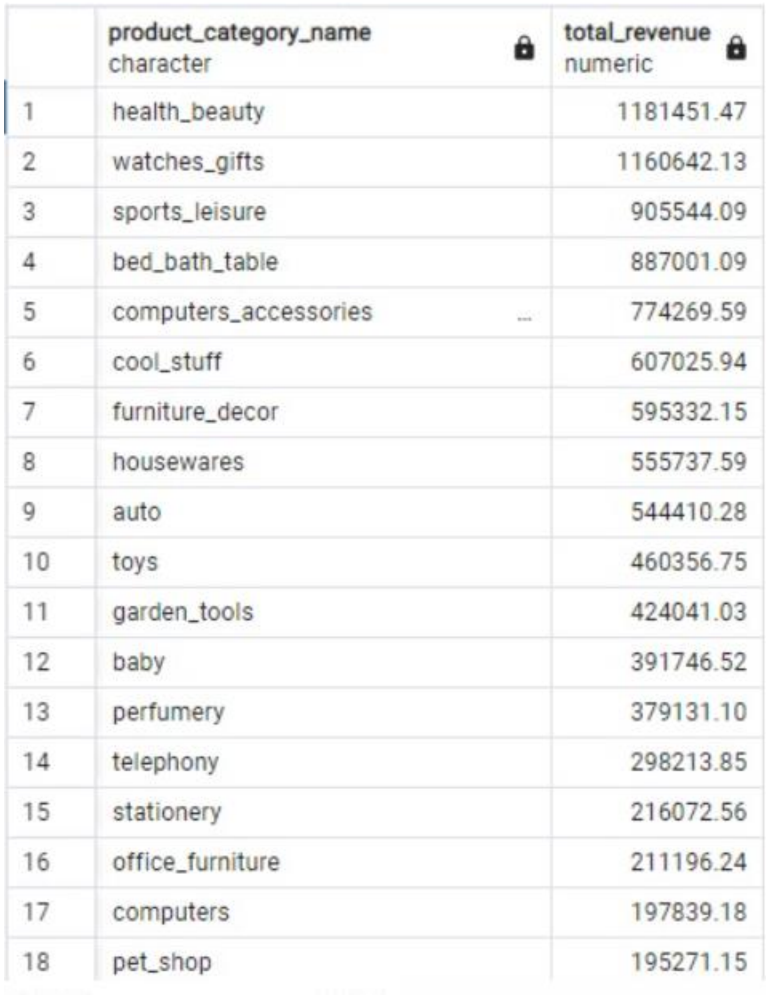
Revenue Analysis

- **Revenue by product category*(Appendix 13)***

Query

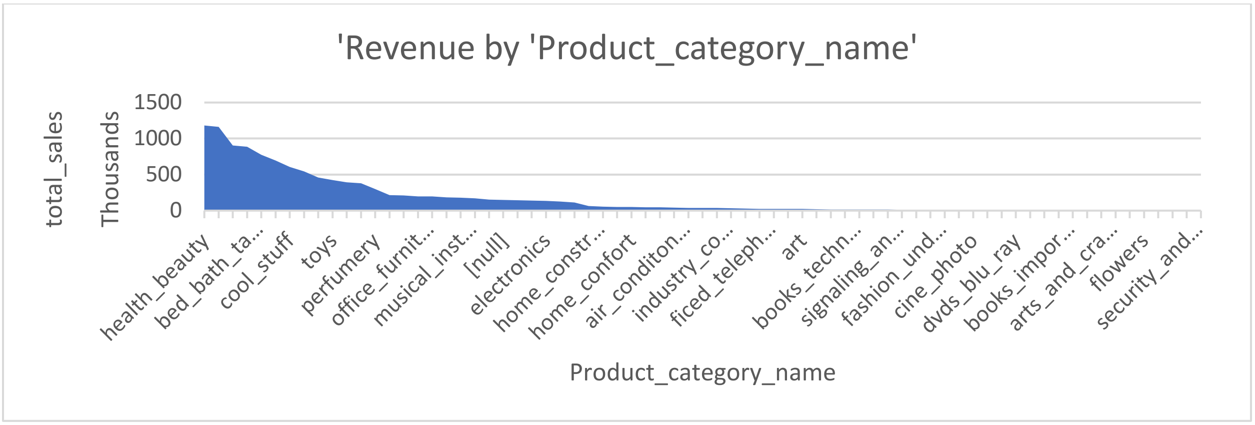
```
-- Revenue by product category

SELECT p.product_category_name,
       ROUND(SUM(o.price)::NUMERIC, 2) AS total_revenue
FROM products p
JOIN orders o ON p.product_id = o.product_id
GROUP BY p.product_category_name
ORDER BY total_revenue DESC;
```

Output



	product_category_name character	total_revenue numeric
1	health_beauty	1181451.47
2	watches_gifts	1160642.13
3	sports_leisure	905544.09
4	bed_bath_table	887001.09
5	computers_accessories	774269.59
6	cool_stuff	607025.94
7	furniture_decor	595332.15
8	housewares	555737.59
9	auto	544410.28
10	toys	460356.75
11	garden_tools	424041.03
12	baby	391746.52
13	perfumery	379131.10
14	telephony	298213.85
15	stationery	216072.56
16	office_furniture	211196.24
17	computers	197839.18
18	pet_shop	195271.15

Visualization



Summary

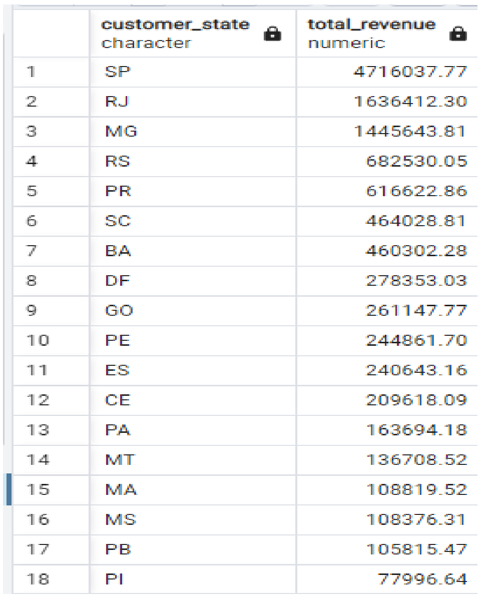
"Health_beauty" and "watches_gifts" emerge as the product categories with the highest revenue, while "security and services" exhibit the lowest revenue among all categories.Sql queries

- **Total Revenue by State*(Appendix 14)***

Query

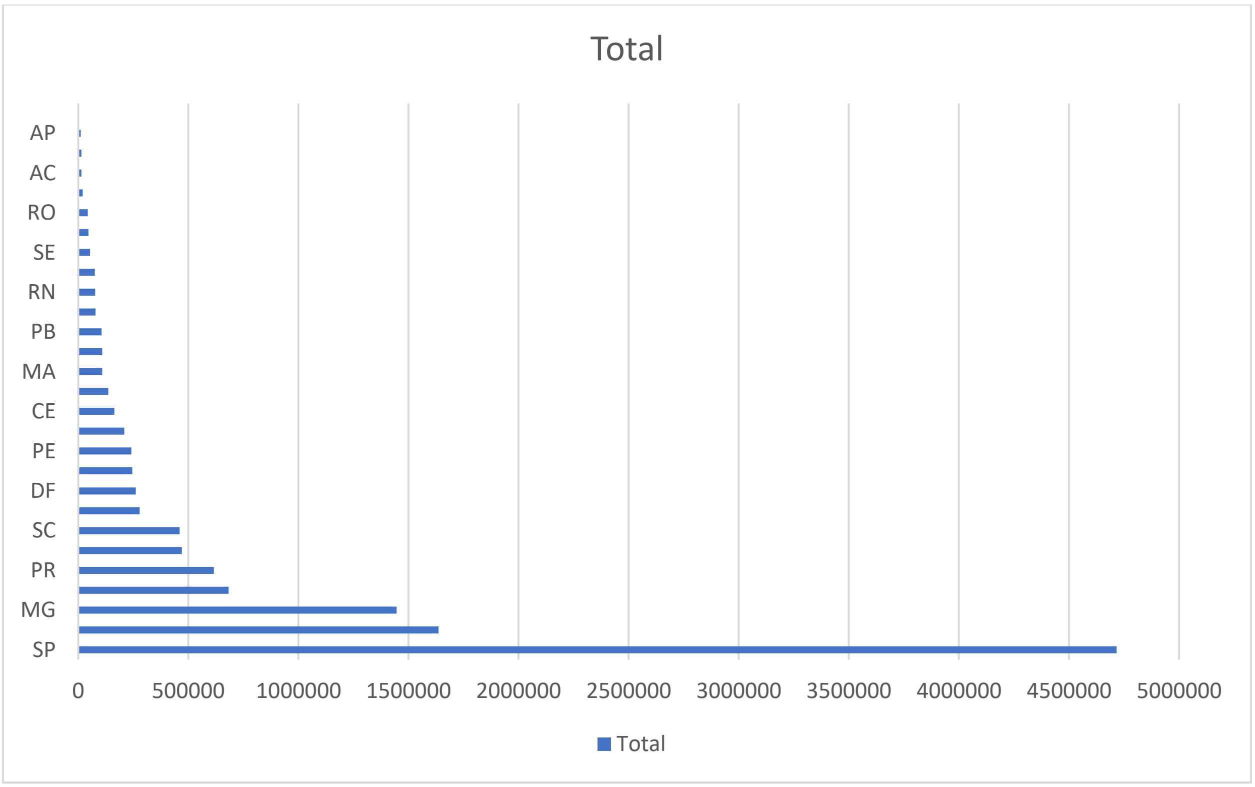
```
-- Total Revenue by State

SELECT c.customer_state,
       ROUND(SUM(o.price::NUMERIC), 2) AS total_revenue
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_state
ORDER BY total_revenue DESC;
```

Output



	customer_state character	total_revenue numeric
1	SP	4716037.77
2	RJ	1636412.30
3	MG	1445643.81
4	RS	682530.05
5	PR	616622.86
6	SC	464028.81
7	BA	460302.28
8	DF	278353.03
9	GO	261147.77
10	PE	244861.70
11	ES	240643.16
12	CE	209618.09
13	PA	163694.18
14	MT	136708.52
15	MA	108819.52
16	MS	108376.31
17	PB	105815.47
18	PI	77996.64

Visualization

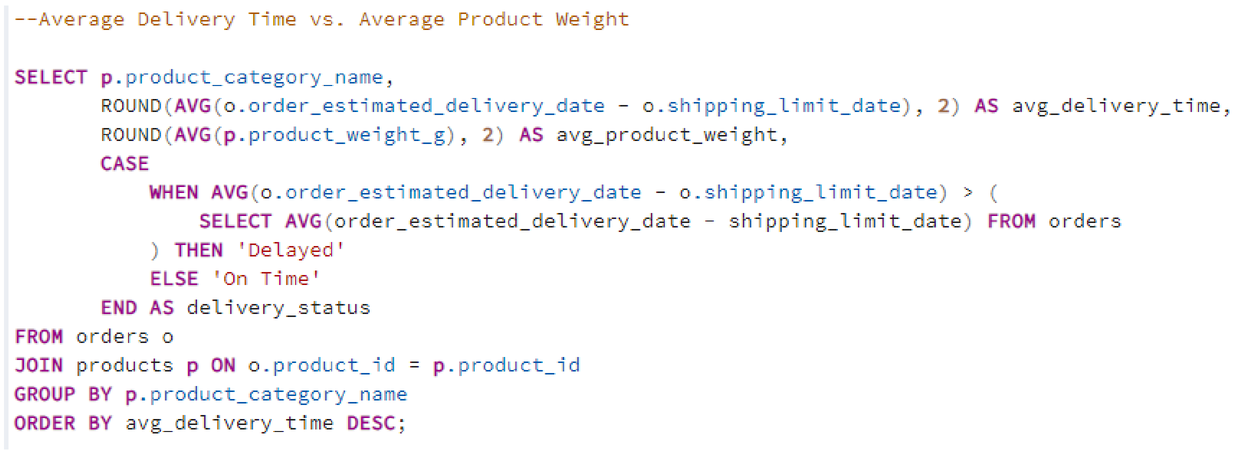


The state labeled as 'SP' commands the highest revenue, displaying a considerable disparity from the other states, and a notable 53% margin over the second-highest state, 'MG'. On the opposite end, 'AP' registers the lowest revenue, amounting to only 11678.57.

Delivery Time Analysis

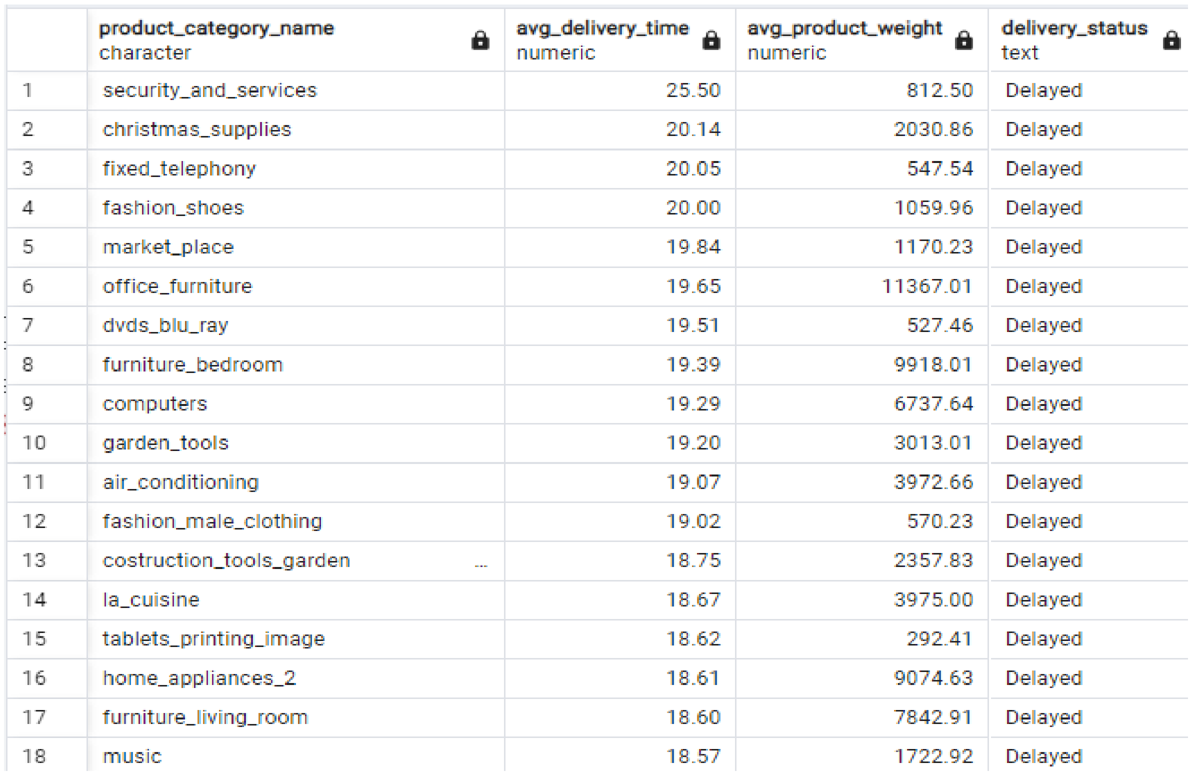
- **Average Delivery Time vs. Average Product Weight*(Appendix 15)***

Query



```
--Average Delivery Time vs. Average Product Weight

SELECT p.product_category_name,
       ROUND(AVG(o.order_estimated_delivery_date - o.shipping_limit_date), 2) AS avg_delivery_time,
       ROUND(AVG(p.product_weight_g), 2) AS avg_product_weight,
       CASE
           WHEN AVG(o.order_estimated_delivery_date - o.shipping_limit_date) > (
               SELECT AVG(order_estimated_delivery_date - shipping_limit_date) FROM orders
           ) THEN 'Delayed'
           ELSE 'On Time'
       END AS delivery_status
FROM orders o
JOIN products p ON o.product_id = p.product_id
GROUP BY p.product_category_name
ORDER BY avg_delivery_time DESC;
```

Output



	product_category_name character	avg_delivery_time numeric	avg_product_weight numeric	delivery_status text
1	security_and_services	25.50	812.50	Delayed
2	christmas_supplies	20.14	2030.86	Delayed
3	fixed_telephony	20.05	547.54	Delayed
4	fashion_shoes	20.00	1059.96	Delayed
5	market_place	19.84	1170.23	Delayed
6	office_furniture	19.65	11367.01	Delayed
7	dvds_blu_ray	19.51	527.46	Delayed
8	furniture_bedroom	19.39	9918.01	Delayed
9	computers	19.29	6737.64	Delayed
10	garden_tools	19.20	3013.01	Delayed
11	air_conditioning	19.07	3972.66	Delayed
12	fashion_male_clothing	19.02	570.23	Delayed
13	costruction_tools_garden ...	18.75	2357.83	Delayed
14	la_cuisine	18.67	3975.00	Delayed
15	tablets_printing_image	18.62	292.41	Delayed
16	home_appliances_2	18.61	9074.63	Delayed
17	furniture_living_room	18.60	7842.91	Delayed
18	music	18.57	1722.92	Delayed

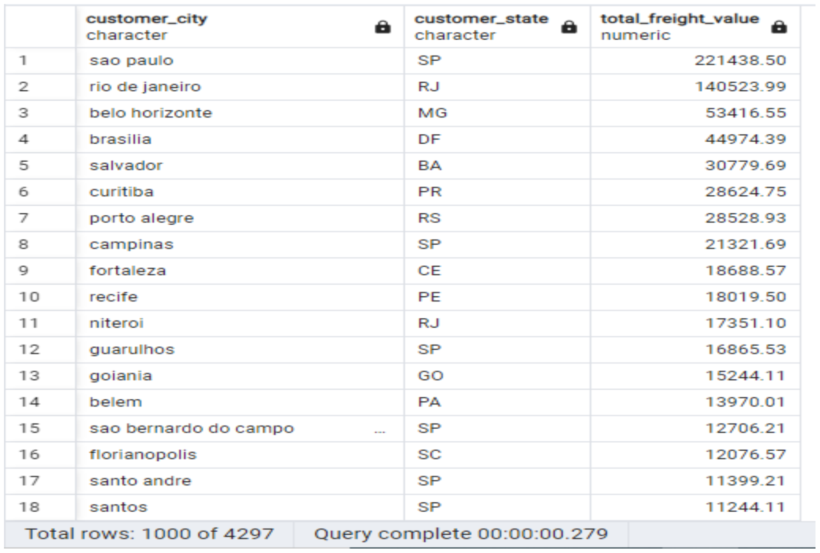
Summary

The table above displays the average weight of each product category alongside their respective delivery statuses. On average, a significant portion of product categories experienced delivery delays, while there were also instances of products being delivered punctually.

- **Total Freight Value Analysis by City and State*(Appendix 16)***

Query

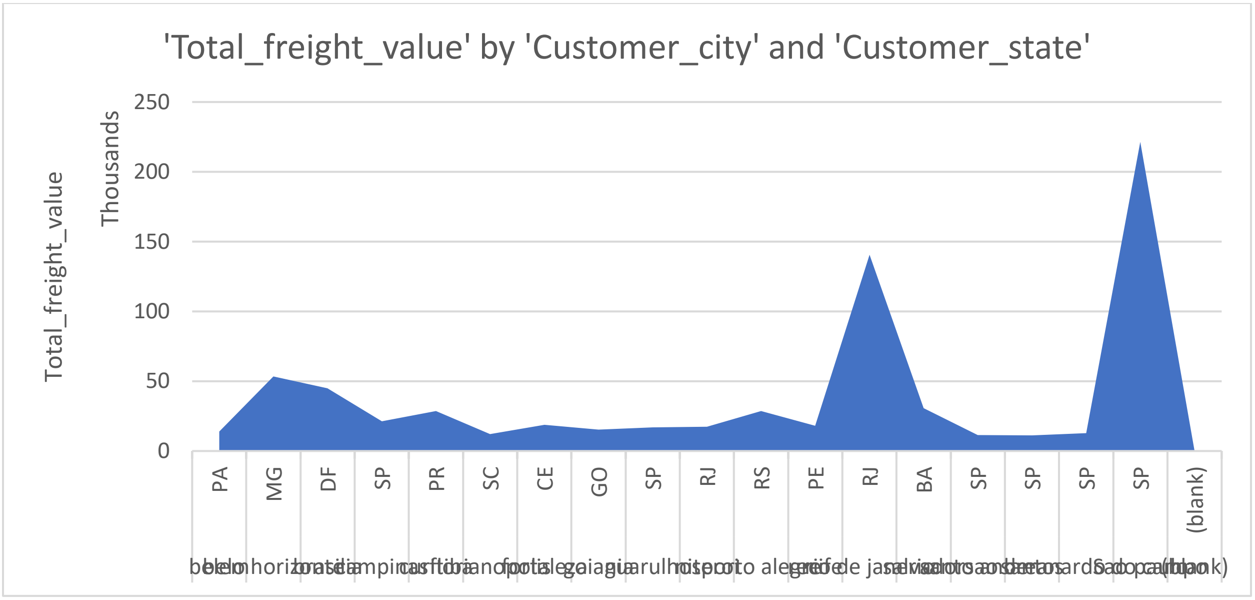
```
-- Total Freight Value Analysis by City and State

SELECT c.customer_city,
       c.customer_state,
       ROUND(SUM(o.freight_value)::NUMERIC, 2) AS total_freight_value
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_city, c.customer_state
ORDER BY total_freight_value DESC;
```



	customer_city character	customer_state character	total_freight_value numeric
1	sao paulo	SP	221438.50
2	rio de janeiro	RJ	140523.99
3	belo horizonte	MG	53416.55
4	brasilia	DF	44974.39
5	salvador	BA	30779.69
6	curitiba	PR	28624.75
7	porto alegre	RS	28528.93
8	campinas	SP	21321.69
9	fortaleza	CE	18688.57
10	recife	PE	18019.50
11	niteroi	RJ	17351.10
12	guarulhos	SP	16865.53
13	goiania	GO	15244.11
14	belem	PA	13970.01
15	sao bernardo do campo ...	SP	12706.21
16	florianopolis	SC	12076.57
17	santo andre	SP	11399.21
18	santos	SP	11244.11

Total rows: 1000 of 4297   Query complete 00:00:00.279

Visualization



Summary

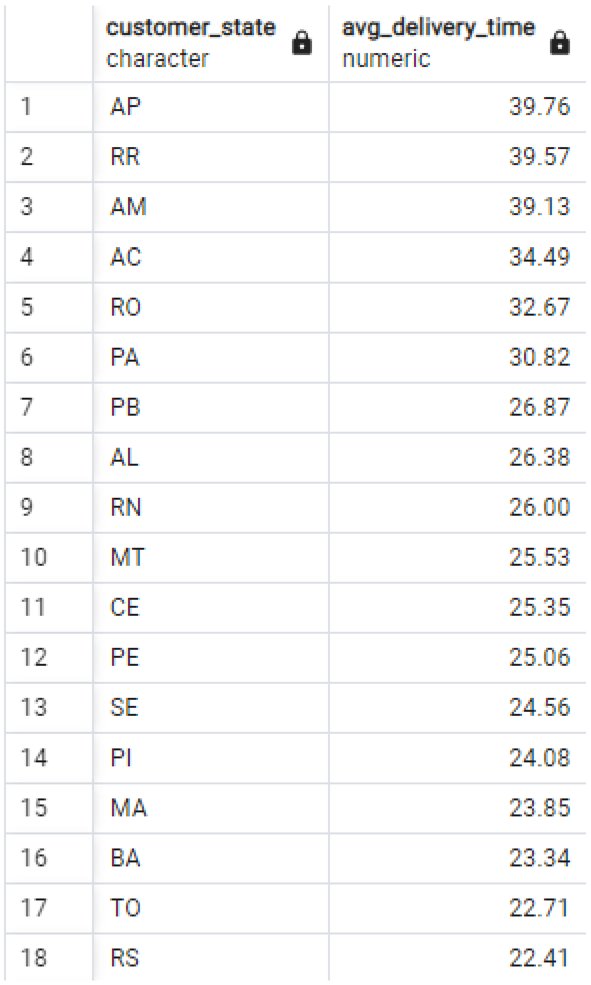
Sao Paulo and Rio De Janeiro stand out with the highest freight values, while Santo Andre and Santos exhibit the lowest. Notably, there's a substantial contrast between Sao Paulo and Santos, with Sao Paulo's freight value exceeding 200K while Santos' remains at 12K+.

- **Average delivery time per state*(Appendix 17)***

Query

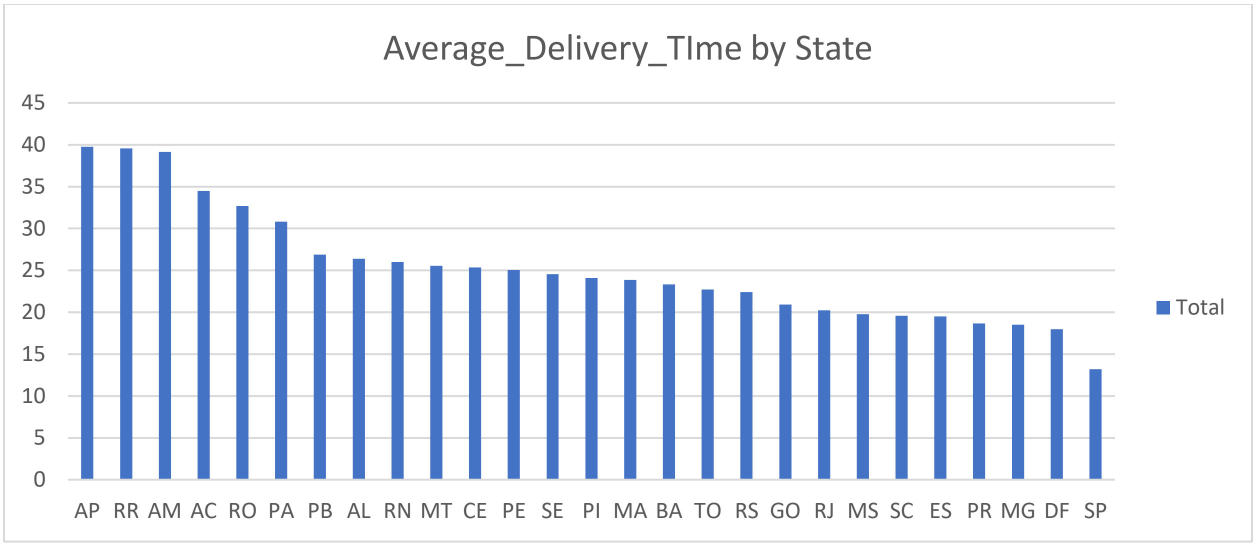
```
--Average deleivery time per state

SELECT c.customer_state,
       ROUND(AVG(o.order_estimated_delivery_date - o.shipping_limit_date), 2) AS avg_delivery_time
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_state
ORDER BY avg_delivery_time DESC;
```

Output



	customer_state character	avg_delivery_time numeric
1	AP	39.76
2	RR	39.57
3	AM	39.13
4	AC	34.49
5	RO	32.67
6	PA	30.82
7	PB	26.87
8	AL	26.38
9	RN	26.00
10	MT	25.53
11	CE	25.35
12	PE	25.06
13	SE	24.56
14	PI	24.08
15	MA	23.85
16	BA	23.34
17	TO	22.71
18	RS	22.41

Visualization



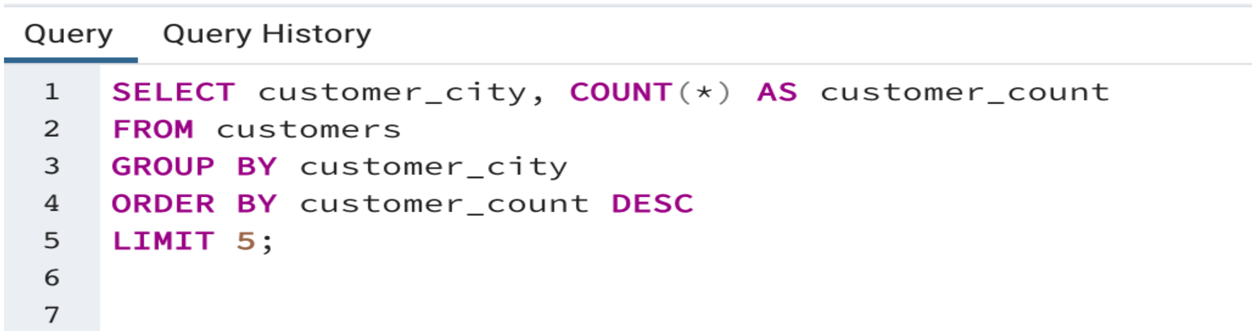
Summary

Both 'AP' and 'RR' demonstrate the longest average delivery times, a trend also observed in 'AM'. Conversely, 'SP' boasts the shortest delivery time, taking less than 15 days on average.

Customer Behaviour Analysis

- **Top 5 cities with the highest number of customers*(Appendix 18)***

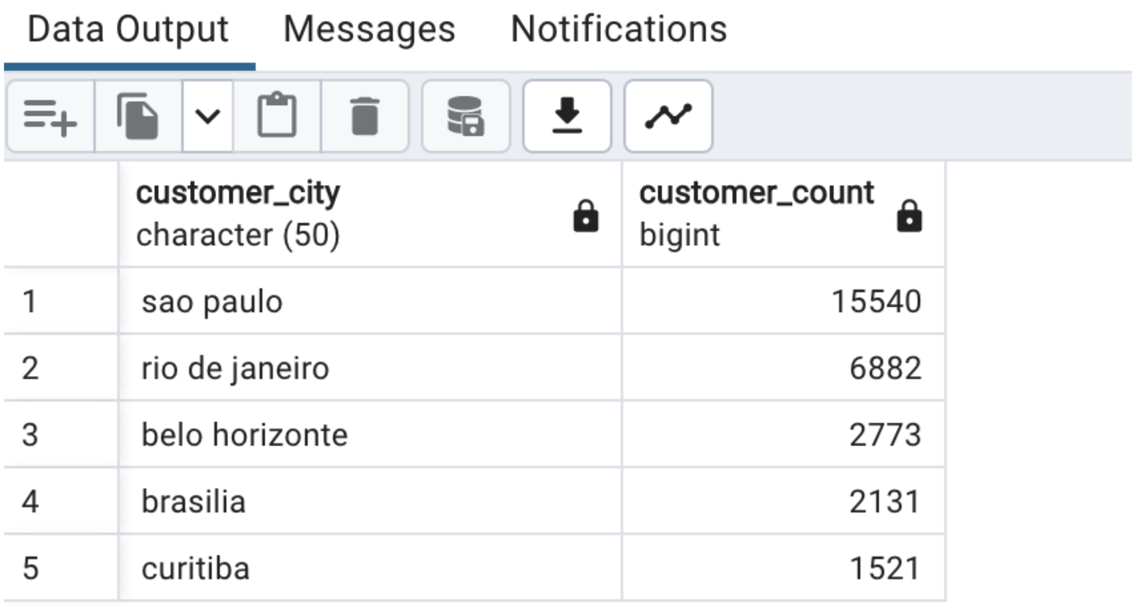
Query



Query   Query History

```sql
SELECT customer_city, COUNT(*) AS customer_count
FROM customers
GROUP BY customer_city
ORDER BY customer_count DESC
LIMIT 5;
```

Output



Data Output   Messages   Notifications

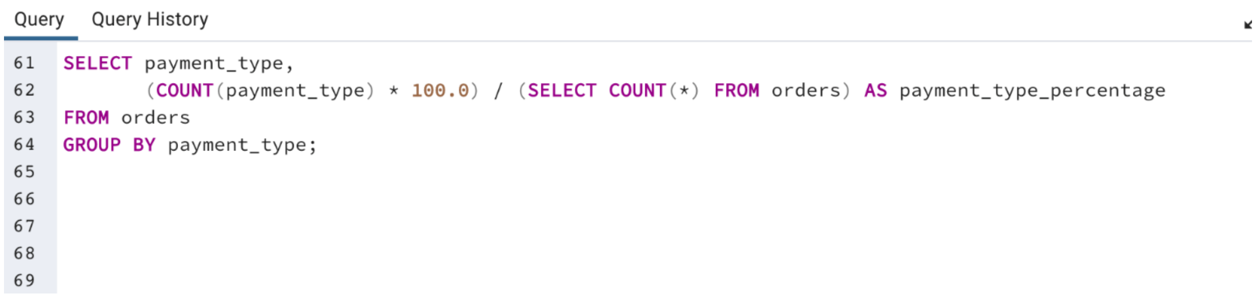
	customer_city character (50)	customer_count bigint
1	sao paulo	15540
2	rio de janeiro	6882
3	belo horizonte	2773
4	brasilia	2131
5	curitiba	1521

Summary

Among the cities 'Sau Paulo' holds the largest customer base with 15,540 customers. 'Rio De Janeiro' in the second place has only 6000+ customers and 'Curitiba' has only a base of '1521' customers. Hence 'Sau Paulo' has a significant higher customer base.

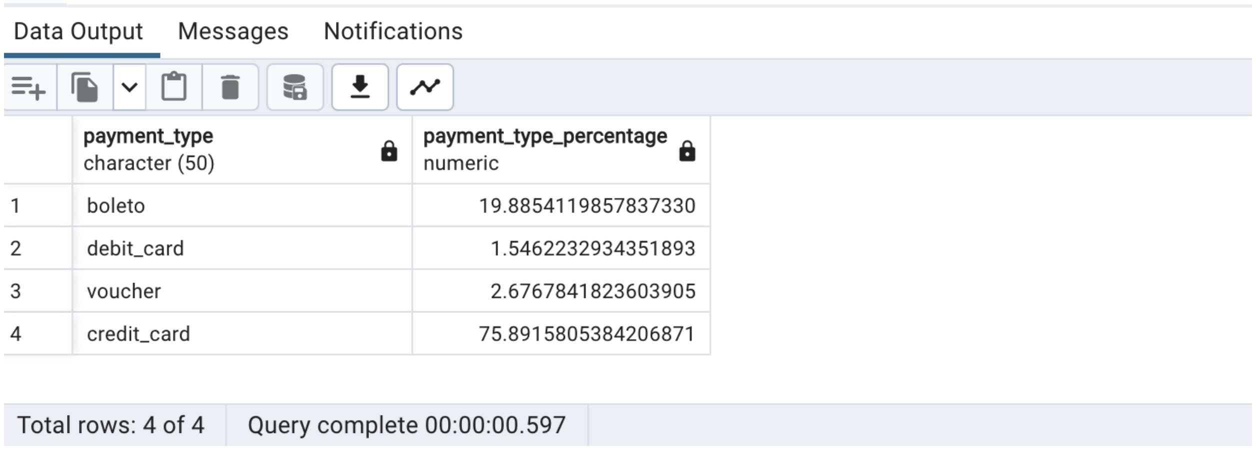
- **Different payment types used by customers for orders*(Appendix 19)***

Query



```
SELECT payment_type,
       (COUNT(payment_type) * 100.0) / (SELECT COUNT(*) FROM orders) AS payment_type_percentage
FROM orders
GROUP BY payment_type;
```

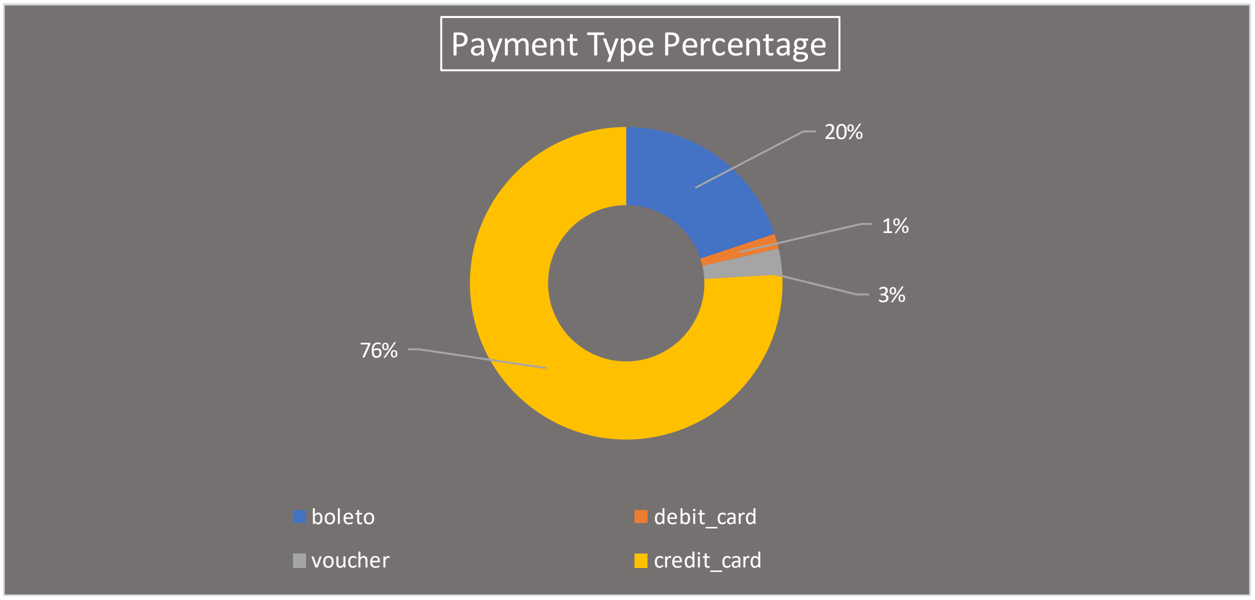
Output



	payment_type character (50)	payment_type_percentage numeric
1	boleto	19.8854119857837330
2	debit_card	1.5462232934351893
3	voucher	2.6767841823603905
4	credit_card	75.8915805384206871

Total rows: 4 of 4 Query complete 00:00:00.597

Visualization

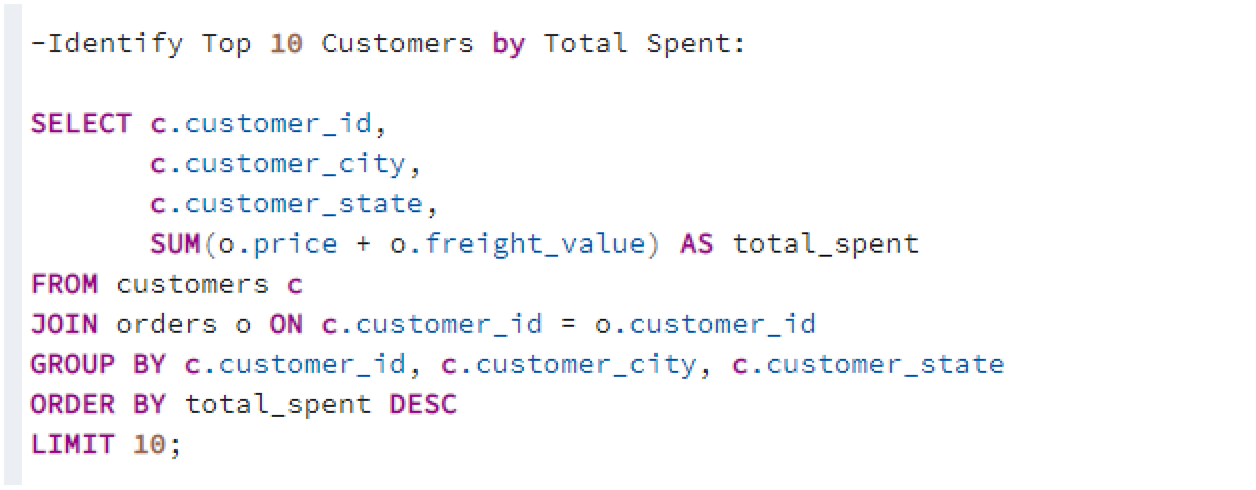


Summary

According to the provided table, over 75% of the payments are conducted through credit cards, while a mere 1.54% are accomplished via debit cards. Furthermore, 19% of customers opt for boleto as their secondary choice of payment.

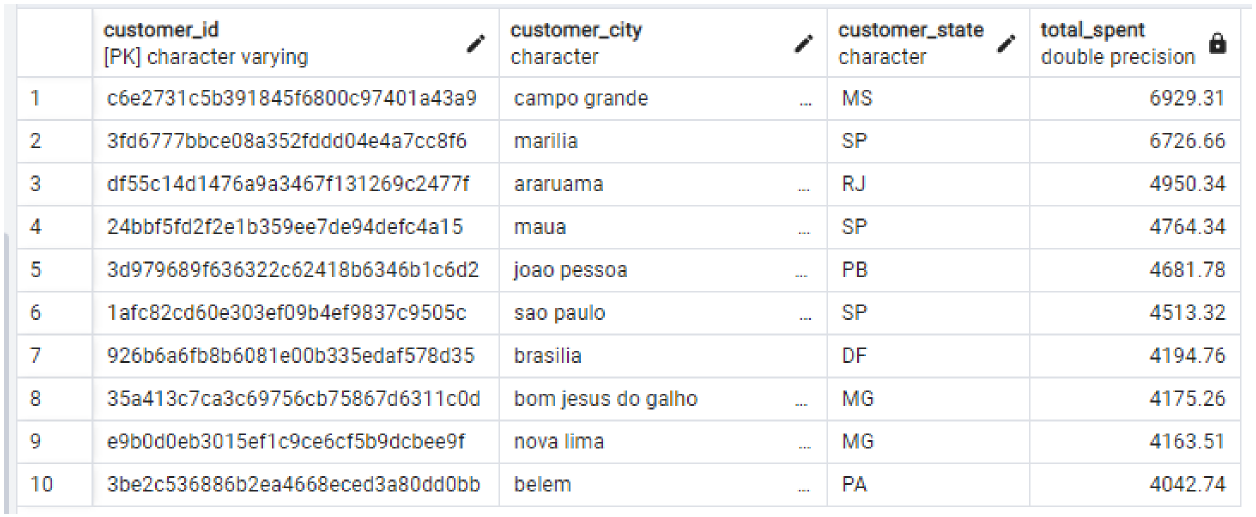
- **Top Customers by Total Spent*(Appendix 20)***

Query



```
-Identify Top 10 Customers by Total Spent:

SELECT c.customer_id,
       c.customer_city,
       c.customer_state,
       SUM(o.price + o.freight_value) AS total_spent
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_id, c.customer_city, c.customer_state
ORDER BY total_spent DESC
LIMIT 10;
```

Output



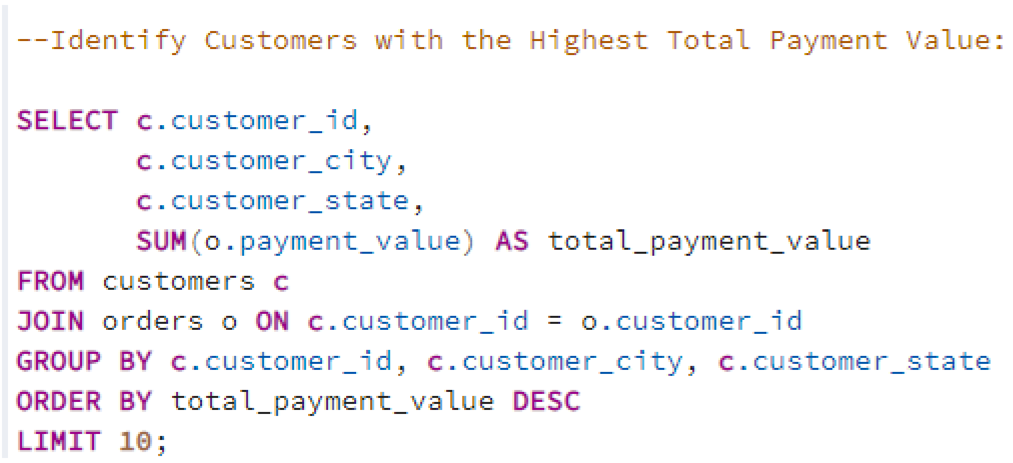
	customer_id [PK] character varying	customer_city character	customer_state character	total_spent double precision
1	c6e2731c5b391845f6800c97401a43a9	campo grande ...	MS	6929.31
2	3fd6777bbce08a352fddd04e4a7cc8f6	marilia	SP	6726.66
3	df55c14d1476a9a3467f131269c2477f	araruama ...	RJ	4950.34
4	24bbf5fd2f2e1b359ee7de94defc4a15	maua ...	SP	4764.34
5	3d979689f636322c62418b6346b1c6d2	joao pessoa ...	PB	4681.78
6	1afc82cd60e303ef09b4ef9837c9505c	sao paulo ...	SP	4513.32
7	926b6a6fb8b6081e00b335edaf578d35	brasilia	DF	4194.76
8	35a413c7ca3c69756cb75867d6311c0d	bom jesus do galho ...	MG	4175.26
9	e9b0d0eb3015ef1c9ce6cf5b9dcbee9f	nova lima ...	MG	4163.51
10	3be2c536886b2ea4668eced3a80dd0bb	belem ...	PA	4042.74

Summary

The highest-spending customer has made a substantial purchase of 6929, closely followed by the second highest spender at approximately 6726. On average, all of the top 10 customers have exceeded the 4000+ spending mark. Interestingly, only two customers among them have surpassed the 5000 threshold, while the remaining fall within the range of 4000 to 4900. Out of these notable customers, three are associated with the state 'SP', while two hail from 'MG'. The rest of the top spenders are residents of various different states.

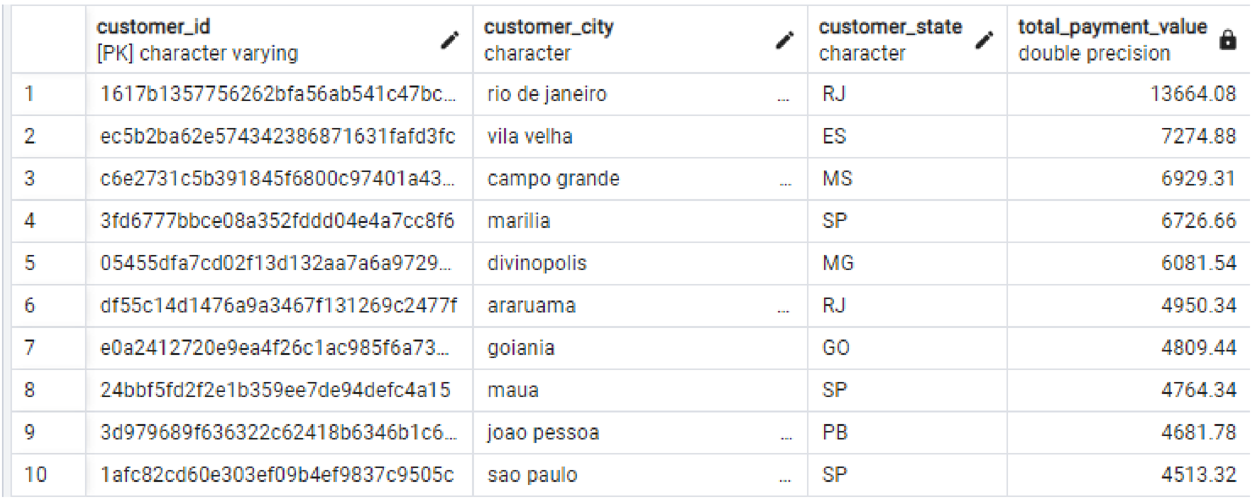
- **Top 10 Customers with the Highest Total Payment Value*(Appendix 21)***

Query



```
--Identify Customers with the Highest Total Payment Value:

SELECT c.customer_id,
       c.customer_city,
       c.customer_state,
       SUM(o.payment_value) AS total_payment_value
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_id, c.customer_city, c.customer_state
ORDER BY total_payment_value DESC
LIMIT 10;
```

Output



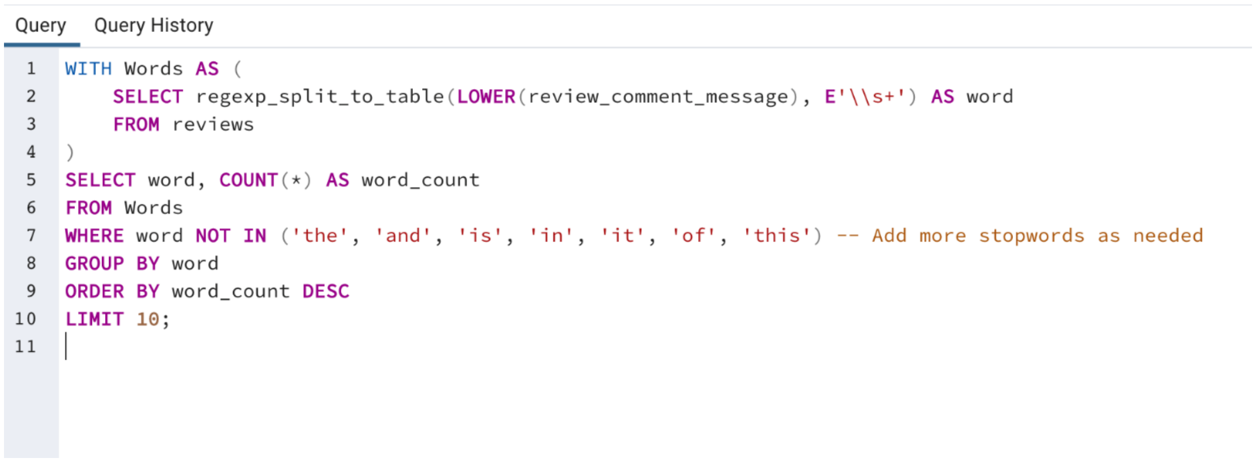
	customer_id [PK] character varying	customer_city character	customer_state character	total_payment_value double precision
1	1617b1357756262bfa56ab541c47bc...	rio de janeiro ...	RJ	13664.08
2	ec5b2ba62e574342386871631fafd3fc	vila velha	ES	7274.88
3	c6e2731c5b391845f6800c97401a43...	campo grande ...	MS	6929.31
4	3fd6777bbce08a352fddd04e4a7cc8f6	marilia	SP	6726.66
5	05455dfa7cd02f13d132aa7a6a9729...	divinopolis	MG	6081.54
6	df55c14d1476a9a3467f131269c2477f	araruama ...	RJ	4950.34
7	e0a2412720e9ea4f26c1ac985f6a73...	goiania	GO	4809.44
8	24bbf5fd2f2e1b359ee7de94defc4a15	maua	SP	4764.34
9	3d979689f636322c62418b6346b1c6...	joao pessoa ...	PB	4681.78
10	1afc82cd60e303ef09b4ef9837c9505c	sao paulo ...	SP	4513.32

Summary

Rio holds the highest position in terms of payment value, with Vila Velha following closely in second place within the top 10. The state of 'SP' has demonstrated a predominant share in total payment values compared to other states.

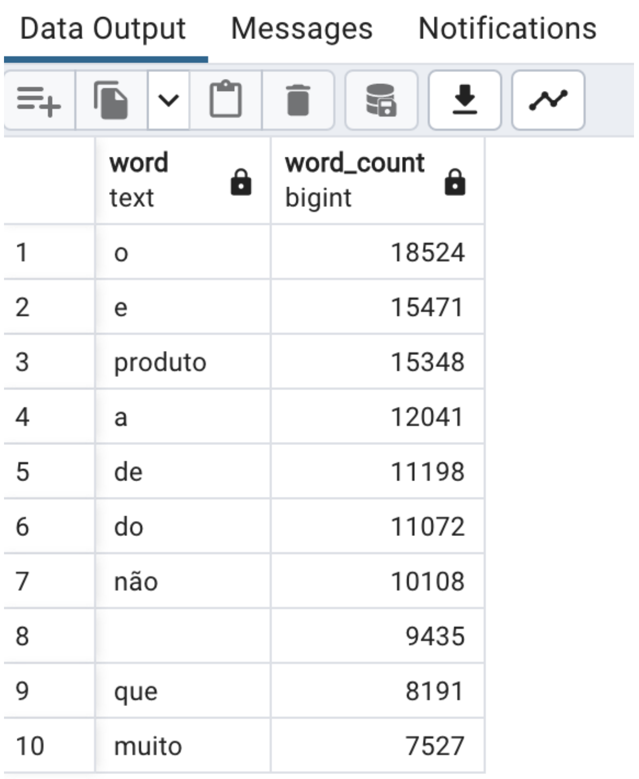
- **Retrieve the most common words used in review comments along with their frequencies, excluding common English stopwords*(Appendix 22)***

Query


Query Query History

```
WITH Words AS (
    SELECT regexp_split_to_table(LOWER(review_comment_message), E'\\s+') AS word
    FROM reviews
)
SELECT word, COUNT(*) AS word_count
FROM Words
WHERE word NOT IN ('the', 'and', 'is', 'in', 'it', 'of', 'this') -- Add more stopwords as needed
GROUP BY word
ORDER BY word_count DESC
LIMIT 10;
```

Output


Data Output Messages Notifications

	word text	word_count bigint
1	o	18524
2	e	15471
3	produto	15348
4	a	12041
5	de	11198
6	do	11072
7	não	10108
8		9435
9	que	8191
10	muito	7527

Summary

the analysis of review comments aimed to identify the most commonly used words and their frequencies, while excluding common English stopwords. The findings indicated that the word 'o' appeared most frequently, occupying the top position, followed by the word 'e' in the second place. This insight provides a snapshot of the prevalent language patterns within the review comments.

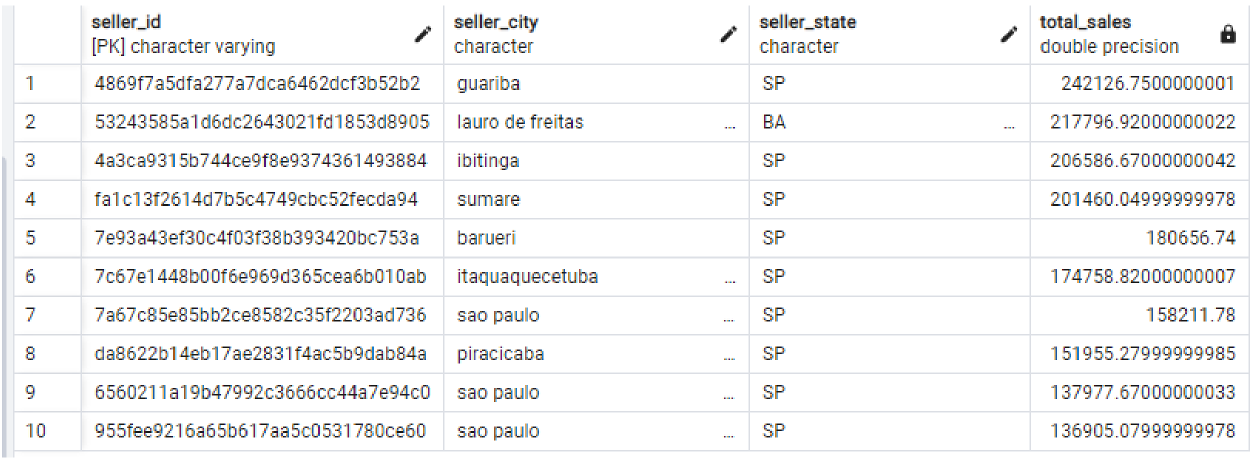
Geographical Analysis

- Top 10 Sellers with Highest Total Sales Value*(Appendix 23)*

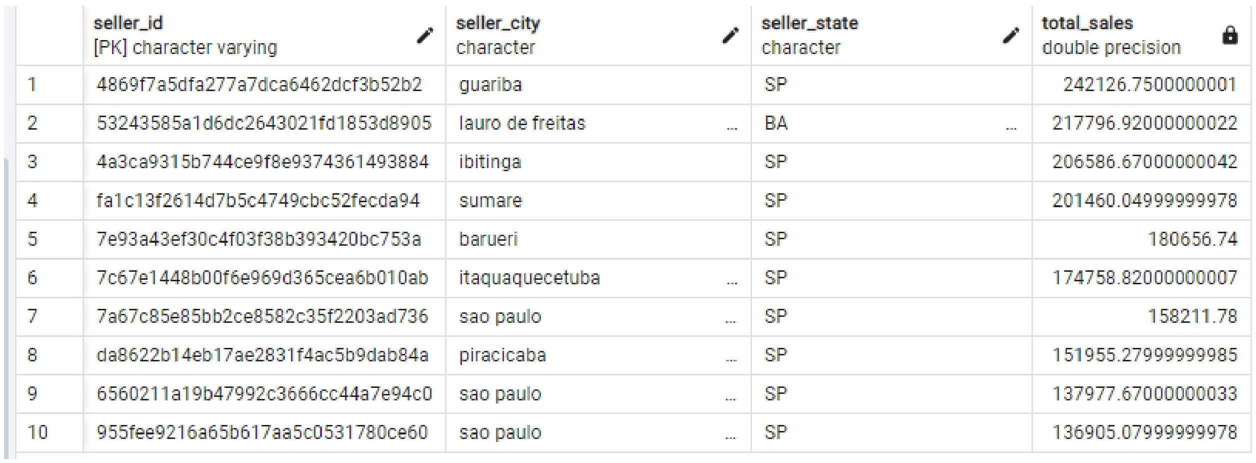
Query

```
- Find  top 10 Sellers with Highest Total Sales Value

SELECT s.seller_id,
       s.seller_city,
       s.seller_state,
       SUM(o.price + o.freight_value) AS total_sales
FROM sellers s
JOIN orders o ON s.seller_id = o.seller_id
GROUP BY s.seller_id, s.seller_city, s.seller_state
ORDER BY total_sales DESC
LIMIT 10;
```

Output



	seller_id [PK] character varying	seller_city character	seller_state character	total_sales double precision
1	4869f7a5dfa277a7dca6462dcf3b52b2	guariba	SP	242126.7500000001
2	53243585a1d6dc2643021fd1853d8905	lauro de freitas	BA	217796.92000000022
3	4a3ca9315b744ce9f8e9374361493884	ibitinga	SP	206586.67000000042
4	fa1c13f2614d7b5c4749cbc52fecda94	sumare	SP	201460.04999999978
5	7e93a43ef30c4f03f38b393420bc753a	barueri	SP	180656.74
6	7c67e1448b00f6e969d365cea6b010ab	itaquaquecetuba	SP	174758.82000000007
7	7a67c85e85bb2ce8582c35f2203ad736	sao paulo	SP	158211.78
8	da8622b14eb17ae2831f4ac5b9dab84a	piracicaba	SP	151955.27999999985
9	6560211a19b47992c3666cc44a7e94c0	sao paulo	SP	137977.67000000033
10	955fee9216a65b617aa5c0531780ce60	sao paulo	SP	136905.07999999978



	seller_id [PK] character varying	seller_city character	seller_state character	total_sales double precision
1	4869f7a5dfa277a7dca6462dcf3b52b2	guariba	SP	242126.7500000001
2	53243585a1d6dc2643021fd1853d8905	lauro de freitas ...	BA ...	217796.92000000022
3	4a3ca9315b744ce9f8e9374361493884	ibitinga	SP	206586.67000000042
4	fa1c13f2614d7b5c4749cbc52fecda94	sumare	SP	201460.04999999978
5	7e93a43ef30c4f03f38b393420bc753a	barueri	SP	180656.74
6	7c67e1448b00f6e969d365cea6b010ab	itaquaquecetuba ...	SP	174758.82000000007
7	7a67c85e85bb2ce8582c35f2203ad736	sao paulo ...	SP	158211.78
8	da8622b14eb17ae2831f4ac5b9dab84a	piracicaba ...	SP	151955.27999999985
9	6560211a19b47992c3666cc44a7e94c0	sao paulo ...	SP	137977.67000000033
10	955fee9216a65b617aa5c0531780ce60	sao paulo ...	SP	136905.07999999978

Summary

The peak sales figure observed stands at 242K+, whereas the lowest registers at 13k+. Interestingly, four vendors have notably surpassed the 20K+ sales mark. Among the highest-grossing top 10 sellers, 30% originate from the city of 'Sao Paulo'. Conversely, the remaining 70% exhibit an even distribution across various cities. Notably, a significant 90% of the top 10 sellers hail from the state of 'SP'.

Summary

Product Category Analysis:

The product category that excels the most is Health_beauty, generating the highest revenue. Generally, a significant portion of products receive reviews surpassing 4.0 on average. Furniture items tend to possess higher weights, contributing to delivery delays. As a suggestion, it's advisable to allocate more resources towards Health_beauty products and emphasize expedited delivery services.

Revenue Enhancement:

Health_beauty stands out as the leading revenue-generating product category. Among the states, 'SP' secures the top position in terms of revenue. While it's recommended to prioritize revenue optimization for Health_beauty products, there should also be a concerted effort to enhance revenue across other product categories due to the substantial revenue gap. This approach applies similarly to the various states.

Delivery Time Optimization:

The analysis highlights a prevalent trend of delayed deliveries, indicating the need for concentrated attention in this area. Additionally, efforts should be directed towards minimizing freight costs within states, while prioritizing the reduction of delivery times, particularly in states 'AP' and 'RR'.

Customer Satisfaction Improvement:

Customers generally express satisfaction with the products, as reflected by reviews averaging 4.0 or higher. A significant proportion of customers originate from the state 'SP', and credit card emerges as their preferred payment method, utilized by 75% of them. Thus, a recommendation is to extend targeted offers to boost sales, particularly in regions beyond the primary city and state.

Geographical Analysis:

The state with the label 'SP' dominates in both revenue and sales. Therefore, it is recommended to allocate more resources to invest in this state. However, in order to achieve balanced and robust business growth, it is essential to evenly distribute business activities among the other states.

Appendix

1. Creating dimentional tables

```
-- Creating dimentional tables
CREATE TABLE customers(
    customer_id CHARACTER VARYING (50) PRIMARY KEY,
    customer_zip_code_prefix INTEGER,
    customer_city CHARACTER (50),
    customer_state CHARACTER (10)
);

CREATE TABLE products(
    product_id CHARACTER VARYING (50) PRIMARY KEY,
    product_category_name CHARACTER (50),
    product_name_lenght INTEGER,
    product_description_lenght INTEGER,
    product_photos_qty INTEGER,
    product_weight_g INTEGER,
    product_length_cm INTEGER,
    product_height_cm INTEGER,
    product_width_cm INTEGER
);

CREATE TABLE sellers (
    seller_id CHARACTER VARYING (50) PRIMARY KEY,
    seller_zip_code_prefix INTEGER,
    seller_city CHARACTER (50),
    seller_state CHARACTER (50)
);


CREATE TABLE reviews(
    review_id CHARACTER VARYING (50),
    order_id CHARACTER VARYING (50),
    review_score INTEGER,
    review_comment_title TEXT,
    review_comment_message TEXT,
    review_creation_date DATE,
    review_answer_timestamp DATE
);


-- Importing data from csv files
COPY customers
FROM 'D:\sql proj data\Group\customers_dim.csv'
DELIMITER ','
CSV HEADER;
```

```
COPY products
FROM 'D:\sql proj data\Group\products_dim.csv'
DELIMITER ','
CSV HEADER;

COPY sellers
FROM 'D:\sql proj data\Group\sellers_dim.csv'
DELIMITER ','
CSV HEADER;

COPY reviews
FROM 'D:\sql proj data\Group\reviews_dim.csv'
DELIMITER ','
CSV HEADER;
```

2. Missing and duplicate values in customers table

```
-- Checking missing values for Customers table
SELECT COUNT(*)
FROM customers
WHERE customer_id IS NULL;

SELECT COUNT(*)
FROM customers
WHERE customer_zip_code_prefix IS NULL;

SELECT COUNT(*)
FROM customers
WHERE customer_city IS NULL;

SELECT COUNT(*)
FROM customers
WHERE customer_state IS NULL;

-- Checking duplicate values for customer table

SELECT customer_id,COUNT(*)
FROM customers
GROUP BY customer_id
HAVING COUNT(*) >1;
```

3. Missing and duplicate values in products table

```
-- Checking missing values for products table
SELECT COUNT(*)
FROM products
```

```
WHERE product_id IS NULL;

-- Checking duplicates for product table
SELECT product_id,COUNT(*)
FROM products
GROUP BY product_id
HAVING COUNT(*) >1;
```

4. Missing and duplicate values in sellers table

```
-- Checking missing values for sellers table
SELECT COUNT(*)
FROM sellers
WHERE seller_id IS NULL;

SELECT COUNT(*)
FROM sellers
WHERE seller_zip_code_prefix IS NULL;

SELECT COUNT(*)
FROM sellers
WHERE seller_city IS NULL;

SELECT COUNT(*)
FROM sellers
WHERE seller_state IS NULL;

-- Checking duplicate values for sellers
SELECT seller_id
FROM sellers
GROUP BY seller_id
HAVING COUNT(*) >1;
```

5. Cleaning reviews table

```
-- Checking missing values for reviews table
SELECT COUNT(*)
FROM reviews
WHERE review_id IS NULL;

SELECT COUNT(*)
FROM reviews
WHERE order_id IS NULL;

SELECT COUNT(*)
FROM reviews
WHERE review_score IS NULL;

-- Checking duplicate values for reviews table
SELECT review_id,count(*)
```

```
FROM reviews
GROUP BY review_id
HAVING COUNT(*) >1;


-- Deleting rows with duplicate values
DELETE FROM reviews a
USING reviews b
WHERE a.ctid <b.ctid AND a.review_id = b.revie  w_id;


-- Updating reviews table primary key
ALTER TABLE reviews
ADD PRIMARY KEY (review_id);
```

6. Creating orders fact table

```
CREATE TABLE orders (
    order_id CHARACTER VARYING (50),
    order_item_id INTEGER,
    product_id CHARACTER VARYING (50),
    seller_id CHARACTER VARYING (50),
    customer_id CHARACTER VARYING (50),
    review_id CHARACTER VARYING (50),
    shipping_limit_date DATE,
    order_estimated_delivery_date DATE,
    price FLOAT,
    freight_value FLOAT,
    payment_sequential INTEGER,
    payment_type CHARACTER (50),
    payment_installments INTEGER,
    payment_value FLOAT,
    FOREIGN KEY (product_id) REFERENCES products (product_id) ON UPDATE CASCADE ON DELETE
CASCADE,
    FOREIGN KEY (seller_id) REFERENCES sellers (seller_id) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (customer_id) REFERENCES customers (customer_id) ON UPDATE CASCADE ON DELETE
CASCADE,
    FOREIGN KEY (review_id) REFERENCES reviews (review_id) ON UPDATE CASCADE ON DELETE CASCADE
);


COPY orders
FROM 'D:\sql proj data\Group\orders_fact.csv'
DELIMITER ','
CSV HEADER
```

7. Missing values in orders table

```
--Checking missing values in Orders Table
SELECT COUNT(*)
FROM orders
WHERE order_id IS NULL;

SELECT COUNT(*)
FROM orders
WHERE product_id IS NULL;

SELECT COUNT(*)
FROM orders
WHERE seller_id IS NULL;

SELECT COUNT(*)
FROM orders
WHERE customer_id IS NULL;

-- Checking missing review_id
SELECT COUNT(*)
FROM orders
WHERE review_id IS NULL;

-- Deleting missing review ids
DELETE FROM orders
WHERE review_id IS NULL;


SELECT COUNT(*)
FROM orders
WHERE payment_sequential IS NULL;

DELETE FROM orders
WHERE payment_sequential IS NULL;
```

8. Duplicates in orders table

```
-- Checking duplicate values
SELECT order_id,count(order_id)
FROM orders
GROUP BY order_id
HAVING COUNT(order_id) >1;

-- removing duplicate values

DELETE FROM orders a
USING orders b
WHERE a.ctid <b.ctid AND a.order_id = b.order_id;
```

9. Average Review Score by Product Category

```
SELECT p.product_category_name,
AVG(r.review_score) AS avg_review_score
FROM products p
JOIN orders o ON p.product_id = o.product_id
JOIN reviews r ON o.review_id = r.review_id
GROUP BY p.product_category_name
ORDER BY avg_review_score DESC;
```

10. Product categories that have the highest average number of product photos

```
SELECT product_category_name, AVG (product_photos_qty) AS avg_photos
FROM products
GROUP BY product_category_name
ORDER BY avg_photos DESC;
```

11. List the top 5 product categories with the highest average product weights

```
SELECT product_category_name, AVG(product_weight_g) AS avg_weight
FROM products
GROUP BY product_category_name
ORDER BY avg_weight DESC
LIMIT 5;
```

12 Product with highest sale

```
SELECT p.product_category_name,
    ROUND(SUM(o.price::NUMERIC), 2) AS total_sales
FROM products p
JOIN orders o ON p.product_id = o.product_id
GROUP BY p.product_category_name
ORDER BY total_sales DESC;
```

13. Revenue by product category

```
SELECT p.product_category_name,
    ROUND(SUM(o.price)::NUMERIC, 2) AS total_revenue
FROM products p
JOIN orders o ON p.product_id = o.product_id
GROUP BY p.product_category_name
ORDER BY total_revenue DESC;
```

14.Total Revenue by State

```
SELECT c.customer_state,
    ROUND(SUM(o.price::NUMERIC), 2) AS total_revenue
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_state
ORDER BY total_revenue DESC;
```

15. Average Delivery Time vs. Average Product Weight

```
SELECT p.product_category_name,
    ROUND(AVG(o.order_estimated_delivery_date - o.shipping_limit_date), 2) AS avg_delivery_time,
    ROUND(AVG(p.product_weight_g), 2) AS avg_product_weight,
    CASE
        WHEN AVG(o.order_estimated_delivery_date - o.shipping_limit_date) > (
            SELECT AVG(order_estimated_delivery_date - shipping_limit_date) FROM orders
        ) THEN 'Delayed'
        ELSE 'On Time'
    END AS delivery_status
FROM orders o
JOIN products p ON o.product_id = p.product_id
GROUP BY p.product_category_name
ORDER BY avg_delivery_time DESC;
```

16. Total Freight Value Analysis by City and State

```
SELECT c.customer_city,
    c.customer_state,
    ROUND(SUM(o.freight_value)::NUMERIC, 2) AS total_freight_value
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_city, c.customer_state
ORDER BY total_freight_value DESC;
```

17. Average delivery time per state

```
SELECT c.customer_state,
    ROUND(AVG(o.order_estimated_delivery_date - o.shipping_limit_date), 2) AS avg_delivery_time
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_state
ORDER BY avg_delivery_time DESC;
```

18. Top 5 cities with the highest number of customers

```
SELECT customer_city, COUNT(*) AS customer_count
FROM customers
GROUP BY customer_city
ODER BY customer_count DESC
LIMIT 5
```

19. Different payment types used by customers for orders.

```
SELECT payment_type,
    (COUNT(payment_type) * 100.0) / (SELECT COUNT(*) FROM orders) AS payment_type_percentage
FROM orders
GROUP BY payment_type;
```

20. Identify Top Customers by Total Spent

```
SELECT c.customer_id,
    c.customer_city,
    c.customer_state,
    SUM(o.price + o.freight_value) AS total_spent
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_id, c.customer_city, c.customer_state
```

```
ORDER BY total_spent DESC
LIMIT 10;
```

21. Top 10 Customers with the Highest Total Payment Value

```
SELECT c.customer_id,
      c.customer_city,
      c.customer_state,
      SUM(o.payment_value) AS total_payment_value
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_id, c.customer_city, c.customer_state
ORDER BY total_payment_value DESC
LIMIT 10;
```

22. Retrieve the most common words used in review comments along with their frequencies, excluding common English stopwords

```
WITH Words AS (
   SELECT regexp_split_to_table(LOWER(review_comment_message), E'\\s+') AS word
   FROM reviews
)
SELECT word, COUNT(*) AS word_count
FROM Words
WHERE word NOT IN ('the', 'and', 'is', 'in', 'it', 'of', 'this') -- Add more stopwords as needed
GROUP BY word
ORDER BY word_count DESC
LIMIT 10;
```

23. Find top 10 Sellers with Highest Total Sales Value

```
SELECT s.seller_id,
      s.seller_city,
      s.seller_state,
      SUM(o.price + o.freight_value) AS total_sales
FROM sellers s
JOIN orders o ON s.seller_id = o.seller_id
GROUP BY s.seller_id, s.seller_city, s.seller_state
ORDER BY total_sales DESC
LIMIT 10;
```