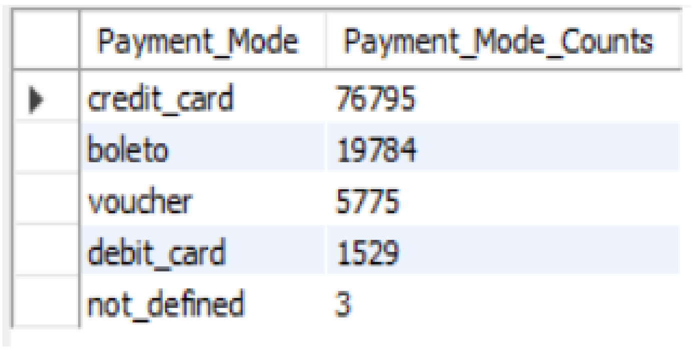
OLIST – BRAZIL: SQL ANALYSIS

Distribution of Payments

```
select payment_type as Payment_Mode, count(payment_type) as Payment_Mode_Counts
from payments
group by Payment_Mode;
```

Output:



Payment_Mode	Payment_Mode_Counts
credit_card	76795
boleto	19784
voucher	5775
debit_card	1529
not_defined	3

Points:

- The table shows different payment modes along with the count of customers who preferred each mode.
- Credit Card has the highest count, indicating it is the most preferred payment method, followed by Boleto.

Order by Payment

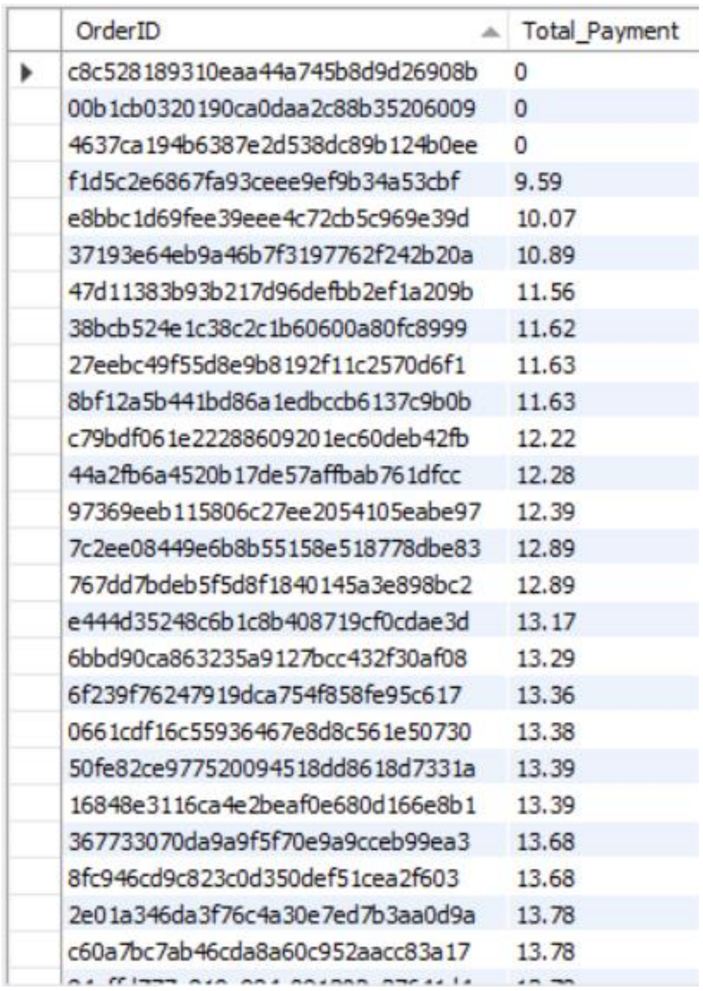
select order_id as OrderID, sum(payment_value) as Total_Payment

from payments

group by OrderID

order by total_payment;

Output:



OrderID	Total_Payment
c8c528189310eaa44a745b8d9d26908b	0
00b1cb0320190ca0daa2c88b35206009	0
4637ca194b6387e2d538dc89b124b0ee	0
f1d5c2e6867fa93ceee9ef9b34a53cbf	9.59
e8bbc1d69fee39eee4c72cb5c969e39d	10.07
37193e64eb9a46b7f3197762f242b20a	10.89
47d11383b93b217d96defbb2ef1a209b	11.56
38bcb524e1c38c2c1b60600a80fc8999	11.62
27eebc49f55d8e9b8192f11c2570d6f1	11.63
8bf12a5b441bd86a1edbccb6137c9b0b	11.63
c79bdf061e2228860920 1ec60deb42fb	12.22
44a2fb6a4520b17de57affbab761dfcc	12.28
97369eeb115806c27ee2054105eabe97	12.39
7c2ee08449e6b8b55158e518778dbe83	12.89
767dd7bdeb5f5d8f1840145a3e898bc2	12.89
e444d35248c6b1c8b408719cf0cdae3d	13.17
6bbd90ca863235a9127bcc432f30af08	13.29
6f239f762479 19dca754f858fe95c617	13.36
0661cdf16c55936467e8d8c561e50730	13.38
50fe82ce977520094518dd8618d7331a	13.39
16848e3116ca4e2beaf0e680d166e8b1	13.39
367733070da9a9f5f70e9a9cceb99ea3	13.68
8fc946cd9c823c0d350def51cea2f603	13.68
2e01a346da3f76c4a30e7ed7b3aa0d9a	13.78
c60a7bc7ab46cda8a60c952aacc83a17	13.78

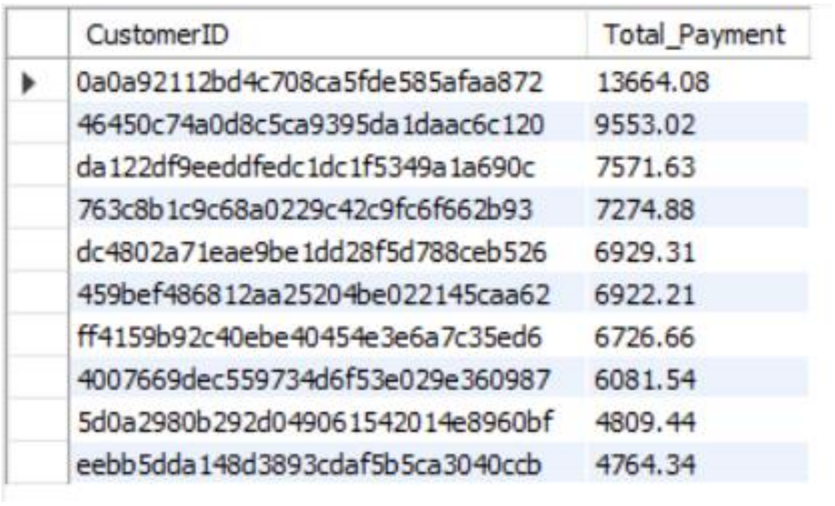
Points:

- The table displays each Order ID along with its corresponding Total Payment.
- Total Payment represents the sum of all payment values associated with each Order.

Top 10 Customers

```
select c.customer_unique_id as CustomerID, sum(p.payment_value) as Total_Payment
from customers c
join orders o on c.customer_id = o.customer_id
join payments p on o.order_id = p.order_id
group by CustomerID
order by Total_Payment desc
limit 10;
```

Output:



CustomerID	Total_Payment
0a0a92112bd4c708ca5fde585afaa872	13664.08
46450c74a0d8c5ca9395da1daac6c120	9553.02
da122df9eeddfedc1dc1f5349a1a690c	7571.63
763c8b1c9c68a0229c42c9fc6f662b93	7274.88
dc4802a71eae9be1dd28f5d788ceb526	6929.31
459bef486812aa25204be022145caa62	6922.21
ff4159b92c40ebe40454e3e6a7c35ed6	6726.66
4007669dec559734d6f53e029e360987	6081.54
5d0a2980b292d049061542014e8960bf	4809.44
eebb5dda148d3893cdaf5b5ca3040ccb	4764.34

Points:

- The table displays the Top 10 Customers along with their Total Revenue.
- Total Revenue is calculated as the sum of all payments made by each customer.

State by Payment

select c.customer_state as State, round(sum(p.payment_value),2) as Total_Payments

from payments p

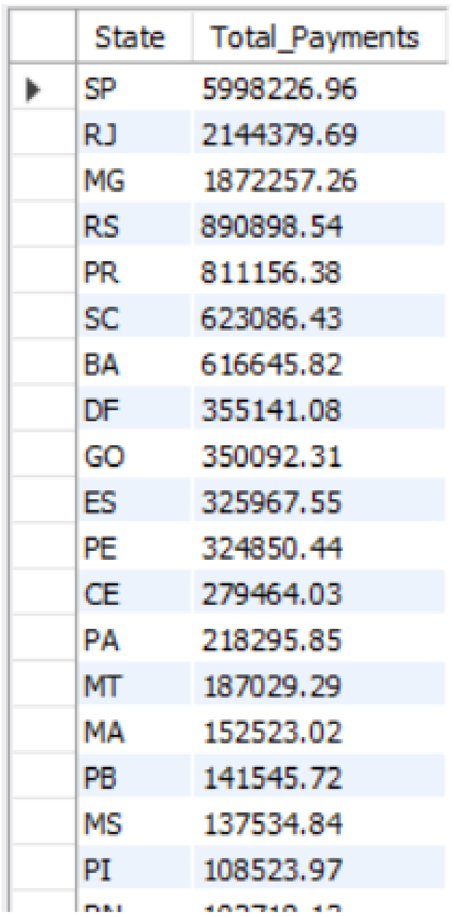
join orders o on p.order_id = o.order_id

join customers c on o.customer_id = c.customer_id

group by State

order by Total_Payments desc;

Output:



State	Total_Payments
SP	5998226.96
RJ	2144379.69
MG	1872257.26
RS	890898.54
PR	811156.38
SC	623086.43
BA	616645.82
DF	355141.08
GO	350092.31
ES	325967.55
PE	324850.44
CE	279464.03
PA	218295.85
MT	187029.29
MA	152523.02
PB	141545.72
MS	137534.84
PI	108523.97
[illegible]	[illegible]

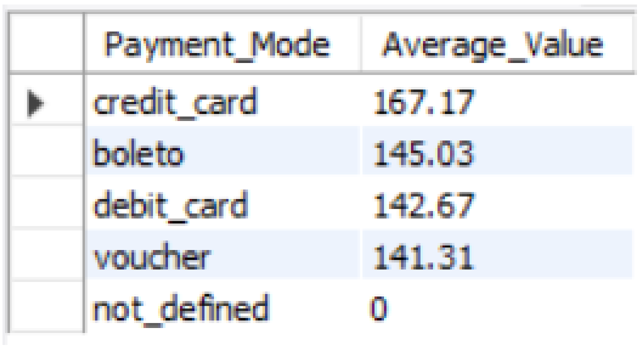
Points:

- The table shows the Customers' States and their corresponding Total Revenue.
- The state SP records the highest revenue among all states.

Payment Mode by Avg. Payment Value

```
select p.payment_type as Payment_Mode, round(avg(total_payment),2) as Average_Value
from(
   select order_id, sum(payment_value) as total_payment
   from payments
   group by order_id
)sub
join payments p on sub.order_id = p.order_id
group by Payment_Mode
order by Average_Value desc;
```

Output:



Payment_Mode	Average_Value
credit_card	167.17
boleto	145.03
debit_card	142.67
voucher	141.31
not_defined	0

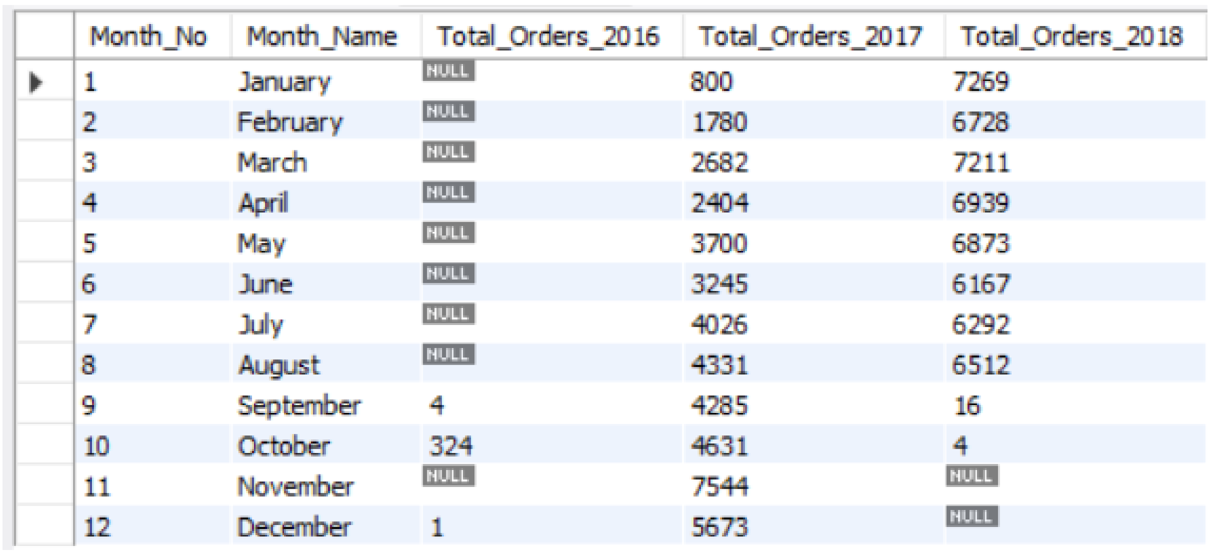
Points:

- The table shows the Average Payment for each Payment Mode.
- Credit Card has the highest average payment value, indicating that most payments were made using credit cards.

Total Orders by Month and Year

```
with cte as(
select
        month(order_purchase_timestamp) as Month_No,
        monthname(order_purchase_timestamp) as Month_Name,
        count(order_id) as Total_Orders_2017
from orders
where year(order_purchase_timestamp) = 2017
group by Month_No, Month_Name
),
cte1 as(
select
        month(order_purchase_timestamp) as Month_No,
        monthname(order_purchase_timestamp) as Month_Name,
        count(order_id) as Total_Orders_2016
from orders
where year(order_purchase_timestamp) = 2016
group by Month_No, Month_Name
),
cte2 as(
select
        month(order_purchase_timestamp) as Month_No,
        monthname(order_purchase_timestamp) as Month_Name,
        count(order_id) as Total_Orders_2018
from orders
where year(order_purchase_timestamp) = 2018
group by Month_No, Month_Name
)
select
        cte.Month_No,
        cte.Month_Name,
        cte1.Total_Orders_2016, cte.Total_Orders_2017, cte2.Total_Orders_2018
from cte
left join cte1 on cte.month_no = cte1.month_no
left join cte2 on cte.month_no = cte2.month_no
order by cte.month_no;
```

Output:



Month_No	Month_Name	Total_Orders_2016	Total_Orders_2017	Total_Orders_2018
1	January	NULL	800	7269
2	February	NULL	1780	6728
3	March	NULL	2682	7211
4	April	NULL	2404	6939
5	May	NULL	3700	6873
6	June	NULL	3245	6167
7	July	NULL	4026	6292
8	August	NULL	4331	6512
9	September	4	4285	16
10	October	324	4631	4
11	November	NULL	7544	NULL
12	December	1	5673	NULL

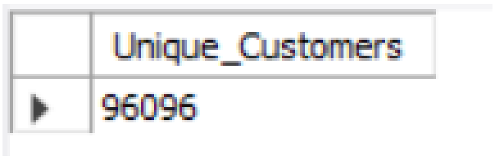
Points:

- The table shows the Total Orders for each year, broken down by months.
- A null value indicates that no data is available for that month.

Count of Unique Customers

```
select count(distinct customer_unique_id) as Unique_Customers

from customers;
```

Output:



```
Unique_Customers
96096
```

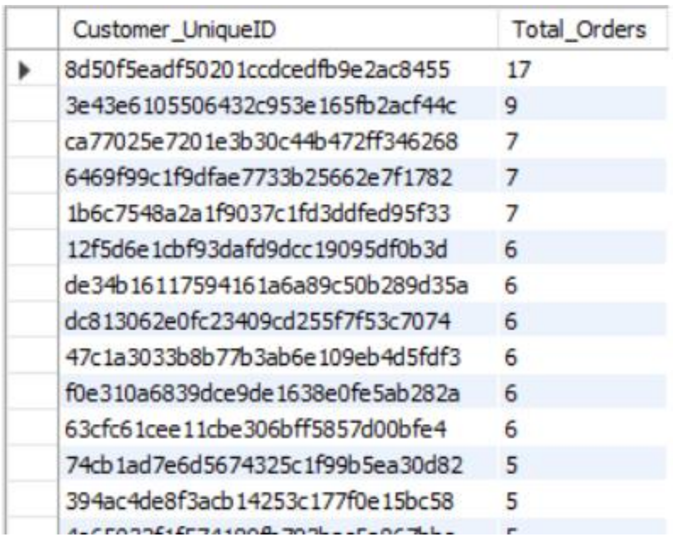
Points:

- The table shows the Total Number of Unique Customers.

Repeat Customers

```
select customer_unique_id as Customer_UniqueID, count(order_id) as Total_Orders
from orders o
join customers c on o.customer_id = c.customer_id
group by customer_unique_id
having count(order_id) > 1
order by total_orders desc;
```

Output:



Customer_UniqueID	Total_Orders
8d50f5eadf50201ccdcedfb9e2ac8455	17
3e43e6105506432c953e165fb2acf44c	9
ca77025e7201e3b30c44b472ff346268	7
6469f99c1f9dfae7733b25662e7f1782	7
1b6c7548a2a1f9037c1fd3ddfed95f33	7
12f5d6e1cbf93dafd9dcc19095df0b3d	6
de34b16117594161a6a89c50b289d35a	6
dc813062e0fc23409cd255f7f53c7074	6
47c1a3033b8b77b3ab6e109eb4d5fdf3	6
f0e310a6839dce9de1638e0fe5ab282a	6
63cfc61cee11cbe306bff5857d00bfe4	6
74cb1ad7e6d5674325c1f99b5ea30d82	5
394ac4de8f3acb14253c177f0e15bc58	5
[illegible]	[illegible]

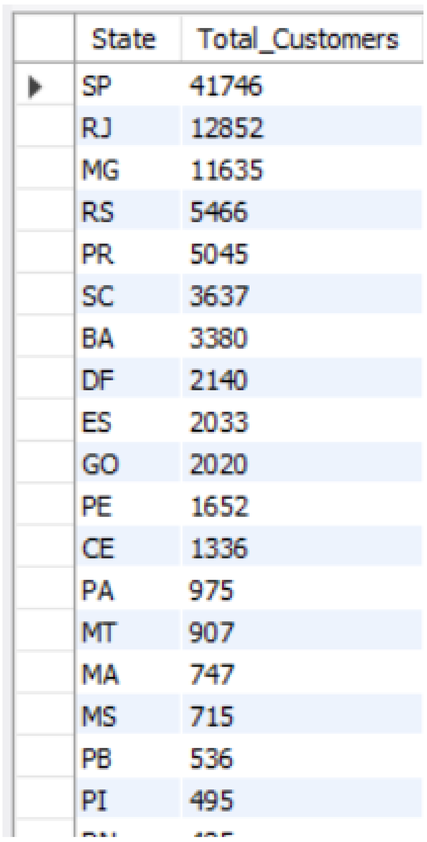
Points:

- The table shows Repeat Customers, i.e., customers who placed more than one order.

Customer by State

```
select customer_state as State, count(customer_id) as Total_Customers

from customers

group by customer_state

order by Total_Customers desc;
```

Output:



State	Total_Customers
SP	41746
RJ	12852
MG	11635
RS	5466
PR	5045
SC	3637
BA	3380
DF	2140
ES	2033
GO	2020
PE	1652
CE	1336
PA	975
MT	907
MA	747
MS	715
PB	536
PI	495
[illegible]	[illegible]

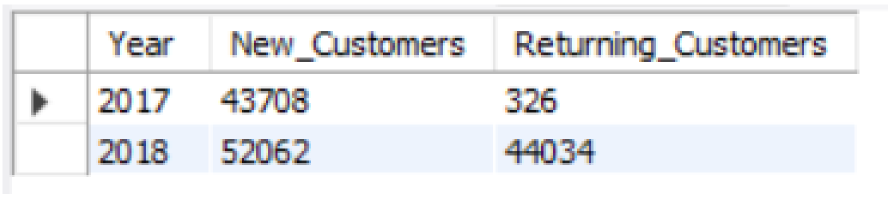
Points:

- The table shows Total Customers grouped by State.
- The state 'SP' has the largest customer base.

New Vs Returning Customers by Year

```
with first_purchase as (
select customer_unique_id, min(order_purchase_timestamp) as first_purchase_date
from orders o
join customers c on o.customer_id = c.customer_id
group by customer_unique_id
)
select
        '2017' as Year,
        count(case when year(first_purchase_date) = 2017 then 1 end) as New_Customers,
        count(case when year(first_purchase_date) < 2017 then 1 end) as
Returning_Customers
from first_purchase
union all
select
        '2018' as Year,
        count(case when year(first_purchase_date) = 2018 then 1 end) as New_Customers,
        count(case when year(first_purchase_date) < 2018 then 1 end) as
Returning_Customers
from first_purchase;
```

Output:



Year	New_Customers	Returning_Customers
2017	43708	326
2018	52062	44034

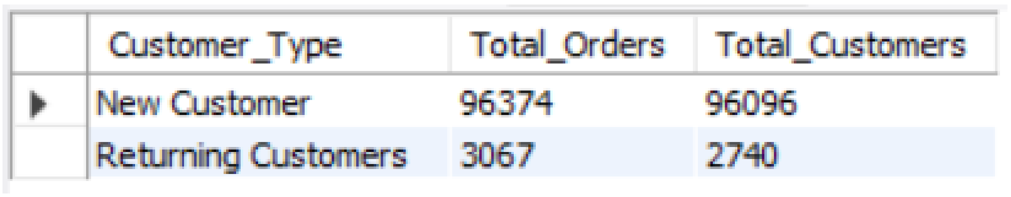
Points:

- The table shows New and Existing Customers for each year (2017 and 2018).
- More new customers made purchases in 2018 compared to 2017.

Total New Vs Total Returning Customers

```
with first_purchase as (
select
        c.customer_unique_id, min(o.order_purchase_timestamp) as first_purchase_date
from orders o
join customers c on o.customer_id = c.customer_id
group by c.customer_unique_id
),
customer_orders as (
select
        c.customer_unique_id, o.order_id, o.order_purchase_timestamp,
        fp.first_purchase_date
from orders o
join customers c on o.customer_id = c.customer_id
join first_purchase fp on c.customer_unique_id = fp.customer_unique_id
)
select
        case
                when order_purchase_timestamp = first_purchase_date then 'New Customer'
                else 'Returning Customers'
        end as Customer_Type,
        count(distinct order_id) as Total_Orders,
        count(distinct customer_unique_id) as Total_Customers
from customer_orders
group by Customer_Type
order by Total_Customers desc;
```

Output:



	Customer_Type	Total_Orders	Total_Customers
▶	New Customer	96374	96096
	Returning Customers	3067	2740

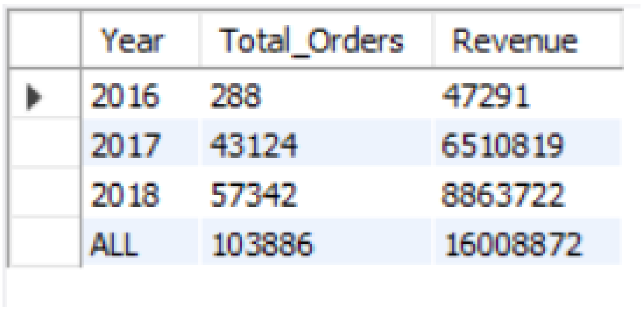
Points:

- The table shows Total Orders and Customers categorized as New or Existing.
- New Customers have placed more orders and represent a larger customer base.

Total Sales (Orders and Revenue) by Year

```
(select '2016' as Year,
        count(o.order_id) as Total_Orders, round(sum(p.payment_value)) as Revenue
from orders o
join payments p on o.order_id = p.order_id
where year(o.order_delivered_customer_date) = 2016)
union all
(select '2017' as Year,
        count(o.order_id) as Total_Orders, round(sum(p.payment_value)) as Revenue
from orders o
join payments p on o.order_id = p.order_id
where year(o.order_delivered_customer_date) = 2017)
union all
(select '2018' as Year,
        count(o.order_id) as Total_Orders, round(sum(p.payment_value)) as Revenue
from orders o
join payments p on o.order_id = p.order_id
where year(o.order_delivered_customer_date) = 2018)
union all
(select 'ALL' as Year,
        count(o.order_id) as Total_Orders, round(sum(p.payment_value)) as Revenue
from orders o
join payments p on o.order_id = p.order_id);
```

Output:



Year	Total_Orders	Revenue
2016	288	47291
2017	43124	6510819
2018	57342	8863722
ALL	103886	16008872

Points:

- The table shows Total Sales by year as well as overall.
- In 2018, the number of orders and revenue generated were higher compared to previous years.

Created a New Table for Simplified Joins:

create table product_table as

select p.product_id, p.product_category_name, ct.product_category_name_english

from products p

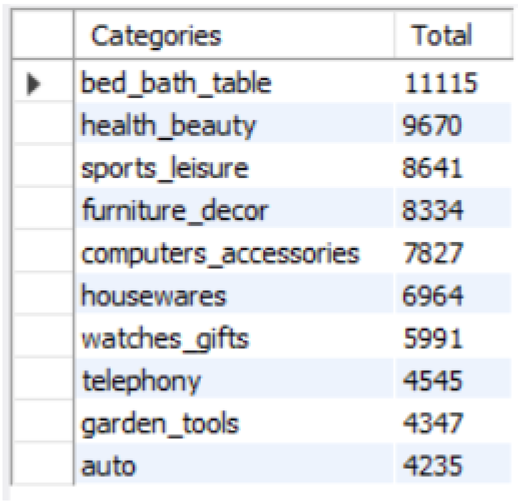
join category_translation ct on ct.product_category_name = p.product_category_name

Points: To join many tables, it shows error: lost connection to mysql server.

Top 10 Product Categories

```
select pd.product_category_name_english as Categories, count(i.order_id) as Total
from product_table pd
join items i on pd.product_id = i.product_id
group by pd.product_category_name_english
order by total desc
limit 10;
```

Output:



Categories	Total
bed_bath_table	11115
health_beauty	9670
sports_leisure	8641
furniture_decor	8334
computers_accessories	7827
housewares	6964
watches_gifts	5991
telephony	4545
garden_tools	4347
auto	4235

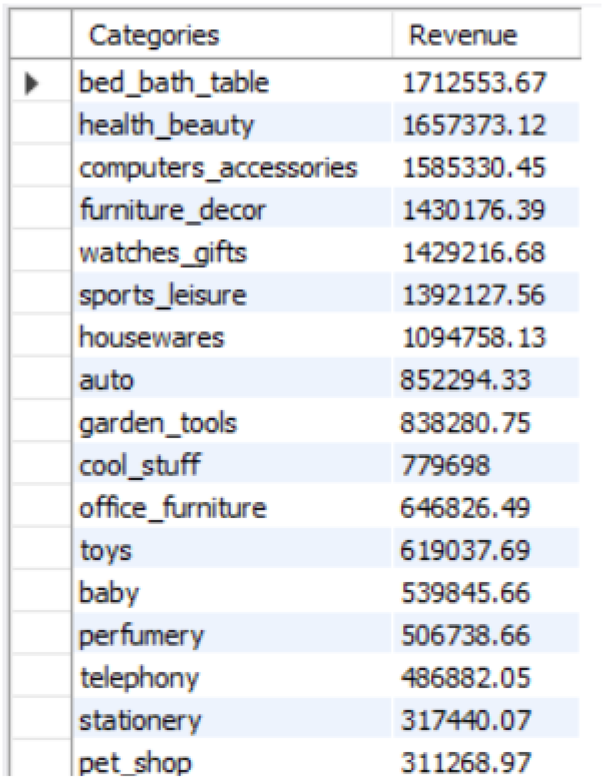
Points:

- The table shows the Top 10 Product Categories ordered by customers.
- 'Bed Bath Table' and 'Health Beauty' are the most purchased categories.

Product Categories by Highest Revenue

```
select pd.product_category_name_english as Categories, round(sum(p.payment_value),2) as Revenue

from product_table pd

join items i on pd.product_id = i.product_id

join payments p on p.order_id = i.order_id

group by product_category_name_english

having Revenue >= 150000

order by Revenue desc;
```

Output:



Categories	Revenue
bed_bath_table	1712553.67
health_beauty	1657373.12
computers_accessories	1585330.45
furniture_decor	1430176.39
watches_gifts	1429216.68
sports_leisure	1392127.56
housewares	1094758.13
auto	852294.33
garden_tools	838280.75
cool_stuff	779698
office_furniture	646826.49
toys	619037.69
baby	539845.66
perfumery	506738.66
telephony	486882.05
stationery	317440.07
pet_shop	311268.97

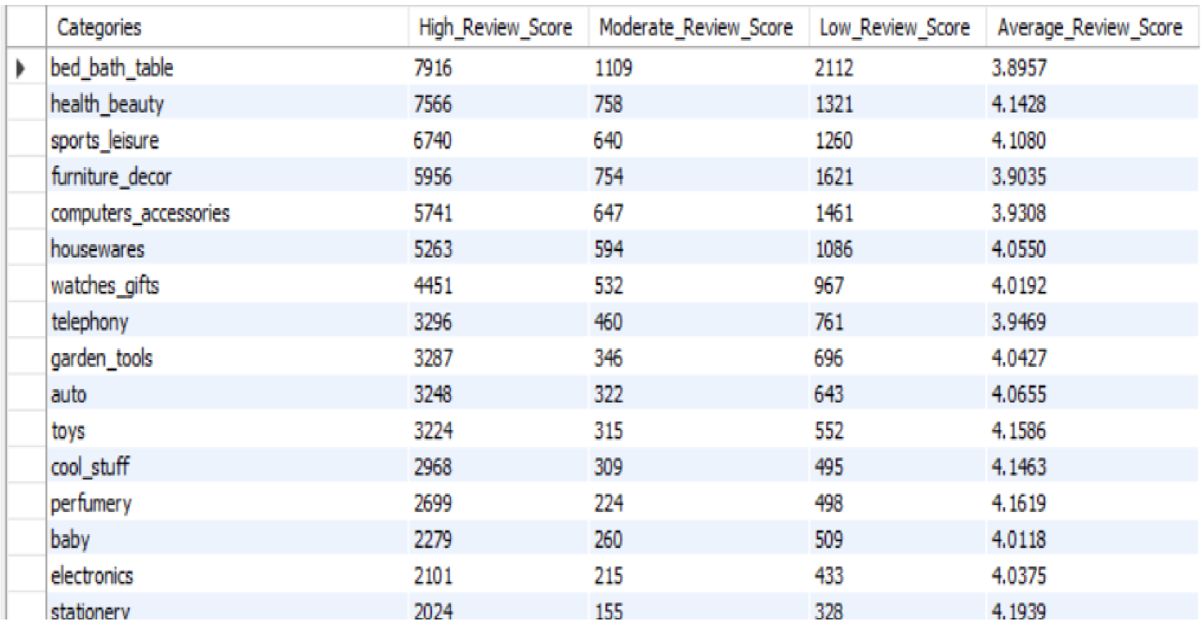
Points:

- The table shows Product Categories and their respective Total Revenue.
- ‘Bed Bath Table’ and ‘Health Beauty’ each generated over 1.5M in revenue.

Product Categories by Level of Review Scores

```
select
        pd.product_category_name_english as Categories,
        sum(case when r.review_score > 3 then 1 else 0 end) as High_Review_Score,
        sum(case when r.review_score = 3 then 1 else 0 end) as Moderate_Review_Score,
        sum(case when r.review_score < 3 then 1 else 0 end) as Low_Review_Score,
        avg(r.review_score) as Average_Review_Score
from product_table pd
join items i on pd.product_id = i.product_id
join reviews r on i.order_id = r.order_id
group by pd.product_category_name_english
order by High_Review_Score desc;
```

Output:



Categories	High_Review_Score	Moderate_Review_Score	Low_Review_Score	Average_Review_Score
bed_bath_table	7916	1109	2112	3.8957
health_beauty	7566	758	1321	4.1428
sports_leisure	6740	640	1260	4.1080
furniture_decor	5956	754	1621	3.9035
computers_accessories	5741	647	1461	3.9308
housewares	5263	594	1086	4.0550
watches_gifts	4451	532	967	4.0192
telephony	3296	460	761	3.9469
garden_tools	3287	346	696	4.0427
auto	3248	322	643	4.0655
toys	3224	315	552	4.1586
cool_stuff	2968	309	495	4.1463
perfumery	2699	224	498	4.1619
baby	2279	260	509	4.0118
electronics	2101	215	433	4.0375
stationery	2024	155	328	4.1939

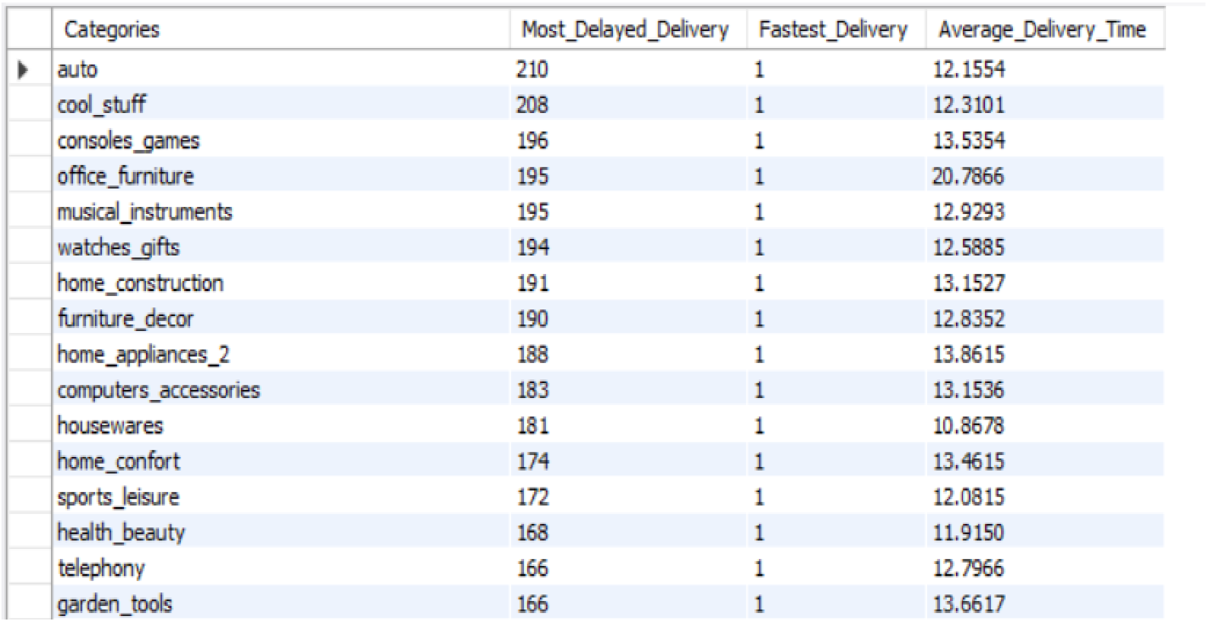
Points:

- The table shows Product Categories with their Review Scores across different levels.
- High Review Scores are defined as greater than 3 (i.e., 4 and 5), Moderate as equal to 3, and Low as less than 3 (i.e., 1 and 2).
- ‘Bed Bath Table’ and ‘Health Beauty’ categories have more than 7k high review scores and fewer than 2.5k low review scores.

Product Categories by Level of Delivery Time

```
select
        pd.product_category_name_english as Categories,
        max(datediff(o.order_delivered_customer_date, o.order_purchase_timestamp)) as Most_Delayed_Delivery,
        min(datediff(o.order_delivered_customer_date, o.order_purchase_timestamp)) as Fastest_Delivery,
        avg(datediff(o.order_delivered_customer_date, o.order_purchase_timestamp)) as Average_Delivery_Time
from orders o
join items i on o.order_id = i.order_id
join product_table pd on i.product_id = pd.product_id
where o.order_delivered_customer_date is not null
group by pd.product_category_name_english
order by Most_Delayed_Delivery desc;
```

Output:



Categories	Most_Delayed_Delivery	Fastest_Delivery	Average_Delivery_Time
auto	210	1	12.1554
cool_stuff	208	1	12.3101
consoles_games	196	1	13.5354
office_furniture	195	1	20.7866
musical_instruments	195	1	12.9293
watches_gifts	194	1	12.5885
home_construction	191	1	13.1527
furniture_decor	190	1	12.8352
home_appliances_2	188	1	13.8615
computers_accessories	183	1	13.1536
housewares	181	1	10.8678
home_confort	174	1	13.4615
sports_leisure	172	1	12.0815
health_beauty	168	1	11.9150
telephony	166	1	12.7966
garden_tools	166	1	13.6617

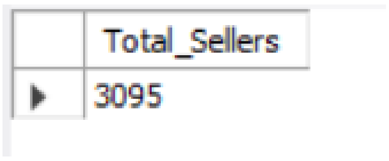
Points:

- The table shows Product Categories with their delivery durations categorized as Delayed, Fast, and Average.
- Most product categories include both the maximum and minimum delivery times.

Count of Unique Sellers

```
select count(distinct seller_id) as Total_Sellers
from sellers;
```

Output:



```
Total_Sellers
3095
```

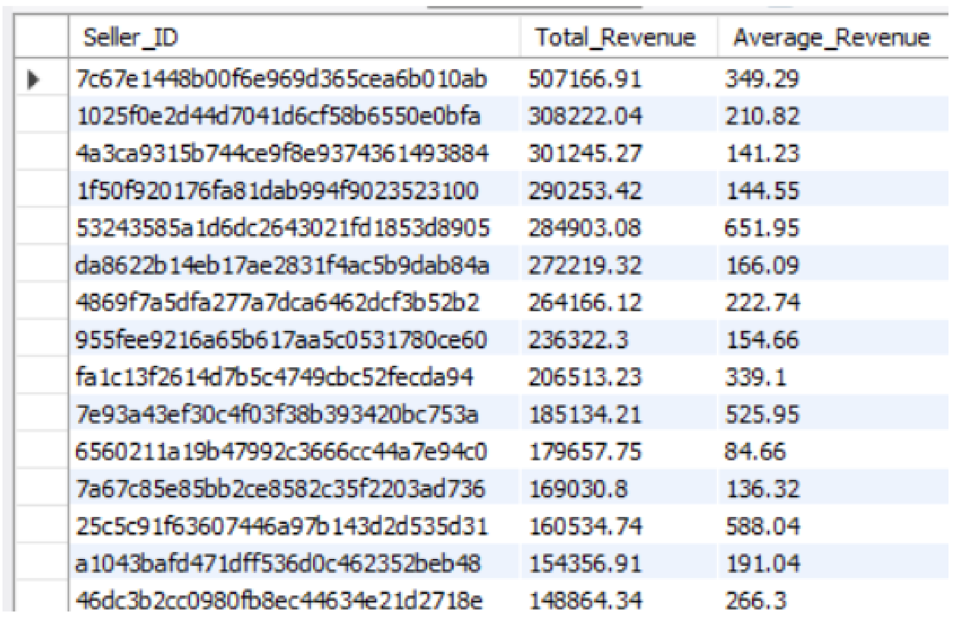
Points:

- The table shows the Total Number of Unique Sellers.

Top Sellers by Revenue

```
select
        s.seller_id as Seller_ID,
        round(sum(p.payment_value), 2) as Total_Revenue,
        round(avg(p.payment_value), 2) as Average_Revenue
from items i
join payments p on i.order_id = p.order_id
join sellers s on i.seller_id = s.seller_id
group by s.seller_id
having Total_Revenue > 100000
order by Total_Revenue desc;
```

Output:



Seller_ID	Total_Revenue	Average_Revenue
7c67e1448b00f6e969d365cea6b010ab	507166.91	349.29
1025f0e2d44d7041d6cf58b6550e0bfa	308222.04	210.82
4a3ca9315b744ce9f8e9374361493884	301245.27	141.23
1f50f920176fa81dab994f9023523100	290253.42	144.55
53243585a1d6dc2643021fd1853d8905	284903.08	651.95
da8622b14eb17ae2831f4ac5b9dab84a	272219.32	166.09
4869f7a5dfa277a7dca6462dcf3b52b2	264166.12	222.74
955fee9216a65b617aa5c0531780ce60	236322.3	154.66
fa1c13f2614d7b5c4749cbc52fecda94	206513.23	339.1
7e93a43ef30c4f03f38b393420bc753a	185134.21	525.95
6560211a19b47992c3666cc44a7e94c0	179657.75	84.66
7a67c85e85bb2ce8582c35f2203ad736	169030.8	136.32
25c5c91f63607446a97b143d2d535d31	160534.74	588.04
a1043bafd471dff536d0c462352beb48	154356.91	191.04
46dc3b2cc0980fb8ec44634e21d2718e	148864.34	266.3

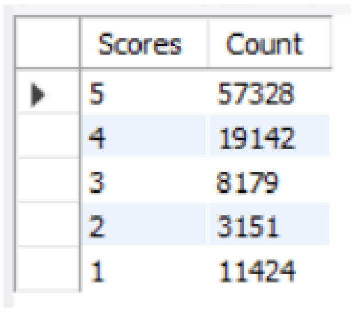
Points:

- The table shows the Top Sellers with their Total and Average Revenue.
- Filtered to include only those with revenue greater than 1 lakh.

Distribution of Review Scores

```
select review_score as Scores, count(review_id) as Count
from reviews
group by review_score
order by Scores desc;
```

Output:



Scores	Count
5	57328
4	19142
3	8179
2	3151
1	11424

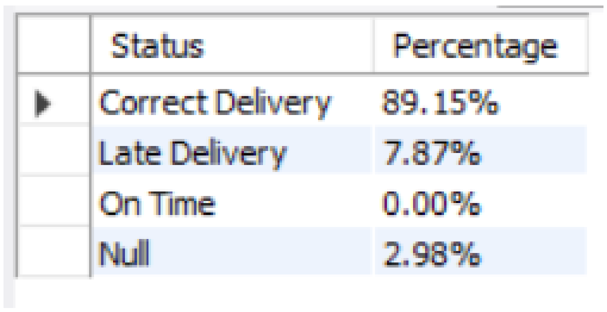
Points:

- The table shows the Distribution of Review Scores.
- A score of 5 has the highest count, indicating that most products received positive reviews from customers.

Percentage of Delivery Status

```
(select
        'Correct Delivery' as Status,
        concat(round((count(order_id) / (select count(order_id) from orders))*100, 2), '%') as
Percentage
from orders
where order_estimated_delivery_date > order_delivered_customer_date)
union
(select
        'Late Delivery' as Status,
        concat(round((count(order_id) / (select count(order_id) from orders))*100, 2), '%') as
Percentage
from orders
where order_estimated_delivery_date < order_delivered_customer_date)
union
(select
        'On Time' as Status,
         concat(round((count(order_id) / (select count(order_id) from orders))*100, 2), '%') as
Percentage
from orders
where order_estimated_delivery_date = order_delivered_customer_date)
union
(select
        'Null' as Status,
        concat(round((count(order_id) / (select count(order_id) from orders))*100, 2), '%') as
Percentage
from orders
where order_estimated_delivery_date is null or order_delivered_customer_date is null);
```

Output:



	Status	Percentage
▶	Correct Delivery	89.15%
	Late Delivery	7.87%
	On Time	0.00%
	Null	2.98%

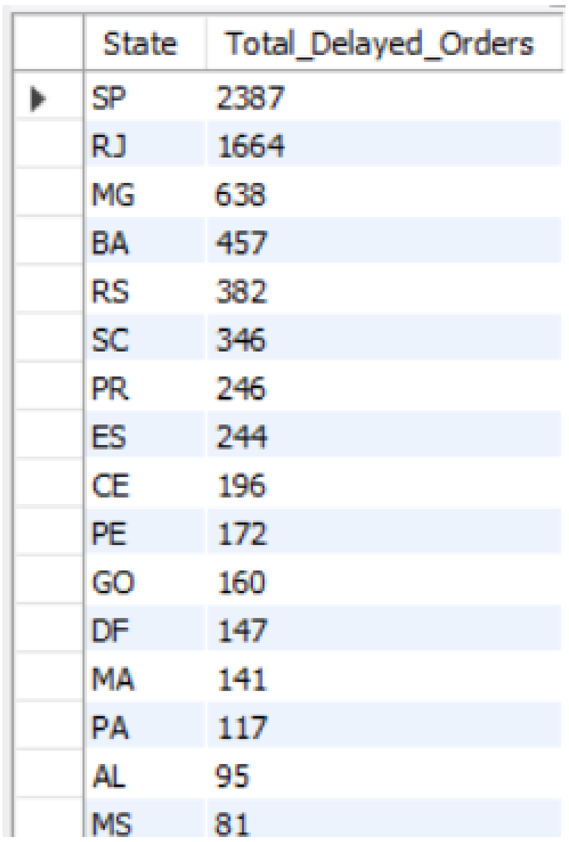
Points:

- The table shows the Percentage of different Delivery Statuses.
- Over 85% of orders reached customers before the estimated delivery date, while only about 8% were delivered late.

Delivery Delays by State

```
select
        c.customer_state as State,
        count(*) as Total_Delayed_Orders
from orders o
join customers c on o.customer_id = c.customer_id
where o.order_estimated_delivery_date < o.order_delivered_customer_date
group by c.customer_state
order by Total_Delayed_Orders desc;
```

Output:



State	Total_Delayed_Orders
SP	2387
RJ	1664
MG	638
BA	457
RS	382
SC	346
PR	246
ES	244
CE	196
PE	172
GO	160
DF	147
MA	141
PA	117
AL	95
MS	81

Points:

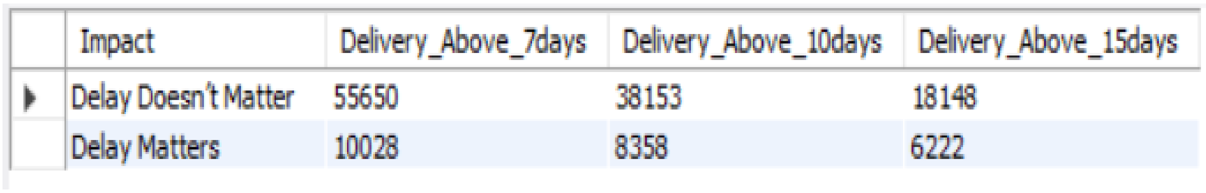
- The table shows States where deliveries were delayed, i.e., reached customers after the estimated delivery date.
- The state ‘SP’ recorded more than 2k delayed deliveries.

Correlation between Delivery and Reviews

Delivery Delays

```
with cte as(
select
        o.order_id as order_id,
        datediff(o.order_delivered_customer_date, o.order_purchase_timestamp) as
delivery_days,
        r.review_score as score
from orders o
join reviews r on o.order_id = r.order_id
where o.order_delivered_customer_date is not null
)
select
   case when score < 3 then 'Delay Matters' else 'Delay Doesn’t Matter' end as Impact,
   count(case when delivery_days > 7 then 1 end) as Delivery_Above_7days,
   count(case when delivery_days > 10 then 1 end) as Delivery_Above_10days,
   count(case when delivery_days > 15 then 1 end) as Delivery_Above_15days
from cte
group by Impact;
```

Output:



Impact	Delivery_Above_7days	Delivery_Above_10days	Delivery_Above_15days
Delay Doesn't Matter	55650	38153	18148
Delay Matters	10028	8358	6222

Points:

- The table shows the correlation between Delivery Days and Review Scores under different threshold scenarios (e.g., deliveries taking more than 7/10/15 days with a review score below 3).
- Delay Matters indicates that longer delivery times lead to lower review scores.

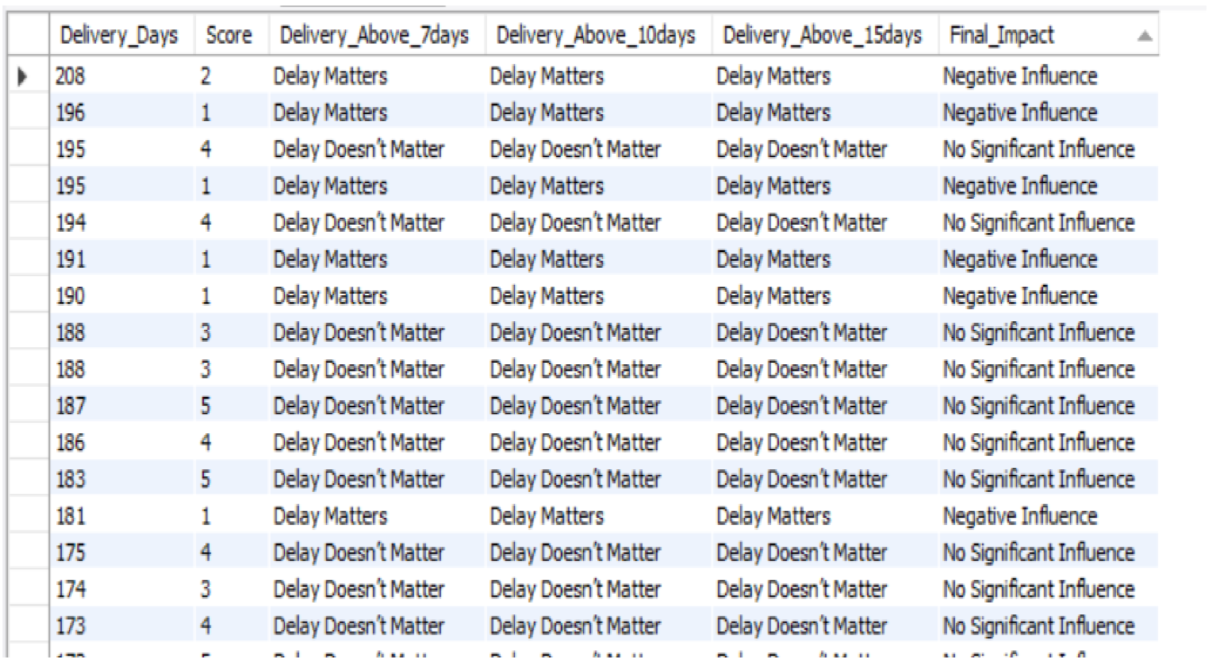
- Delay Doesn’t Matter indicates that longer delivery times do not necessarily result in lower review scores.
- From the table, the counts are higher under Delay Doesn’t Matter than Delay Matters, suggesting that while delays may contribute to low scores, they are not the only factor.

Overall Correlation Delivery Delays Vs Review Scores

```
with cte as(
select o.order_id as order_id,
        datediff(o.order_delivered_customer_date, o.order_purchase_timestamp) as
Delivery_Days,
        r.review_score as Score
from orders o
join reviews r on o.order_id = r.order_id
),
cte1 as(
select order_id, delivery_days, score,
        case when delivery_days > 7 and score < 3 then 'Delay Matters' else 'Delay Doesn’t
Matter' end as Delivery_Above_7days,
        case when delivery_days > 10 and score < 3 then 'Delay Matters' else 'Delay Doesn’t
Matter' end as Delivery_Above_10days,
        case when delivery_days > 15 and score < 3 then 'Delay Matters' else 'Delay Doesn’t
Matter' end as Delivery_Above_15days
from cte
where delivery_days is not null)
select
        Delivery_Days, Score, Delivery_Above_7days, Delivery_Above_10days,
Delivery_Above_15days,
        case
                when Delivery_Above_7days = 'Delay Matters' and Delivery_Above_10days
= 'Delay Matters' then 'Negative Influence'
                when Delivery_Above_10days = 'Delay Matters' and Delivery_Above_15days
= 'Delay Matters' then 'Negative Influence'
```

```
                    when Delivery_Above_7days = 'Delay Matters' and Delivery_Above_15days
= 'Delay Matters' then 'Negative Influence'
                    else 'No Significant Influence'
          end as Final_Impact
from cte1
```

Output:



Delivery_Days	Score	Delivery_Above_7days	Delivery_Above_10days	Delivery_Above_15days	Final_Impact
208	2	Delay Matters	Delay Matters	Delay Matters	Negative Influence
196	1	Delay Matters	Delay Matters	Delay Matters	Negative Influence
195	4	Delay Doesn't Matter	Delay Doesn't Matter	Delay Doesn't Matter	No Significant Influence
195	1	Delay Matters	Delay Matters	Delay Matters	Negative Influence
194	4	Delay Doesn't Matter	Delay Doesn't Matter	Delay Doesn't Matter	No Significant Influence
191	1	Delay Matters	Delay Matters	Delay Matters	Negative Influence
190	1	Delay Matters	Delay Matters	Delay Matters	Negative Influence
188	3	Delay Doesn't Matter	Delay Doesn't Matter	Delay Doesn't Matter	No Significant Influence
188	3	Delay Doesn't Matter	Delay Doesn't Matter	Delay Doesn't Matter	No Significant Influence
187	5	Delay Doesn't Matter	Delay Doesn't Matter	Delay Doesn't Matter	No Significant Influence
186	4	Delay Doesn't Matter	Delay Doesn't Matter	Delay Doesn't Matter	No Significant Influence
183	5	Delay Doesn't Matter	Delay Doesn't Matter	Delay Doesn't Matter	No Significant Influence
181	1	Delay Matters	Delay Matters	Delay Matters	Negative Influence
175	4	Delay Doesn't Matter	Delay Doesn't Matter	Delay Doesn't Matter	No Significant Influence
174	3	Delay Doesn't Matter	Delay Doesn't Matter	Delay Doesn't Matter	No Significant Influence
173	4	Delay Doesn't Matter	Delay Doesn't Matter	Delay Doesn't Matter	No Significant Influence

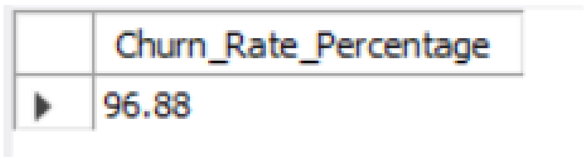
Points:

- The table shows the Overall Correlation between Delivery Delays and Review Scores. As explained in the threshold scenarios, both Delay Matters and Delay Doesn't Matter cases are shown.
- If more than one threshold (above 7, 10, or 15 days) falls under *Delay Matters*, it indicates a significant negative influence.
- If only one threshold falls under *Delay Matters*, it is considered to have no significant influence.

Customer Churn Rate

```
with customer_orders as(
select c.customer_unique_id, count(o.order_id) as Total_Orders
from customers c
join orders o on c.customer_id = o.customer_id
group by c.customer_unique_id
)
select
        round(sum(case when total_orders = 1 then 1 else 0 end) / count(*) * 100, 2) as
Churn_Rate_Percentage
from customer_orders;
```

Output:



```
Churn_Rate_Percentage
96.88
```

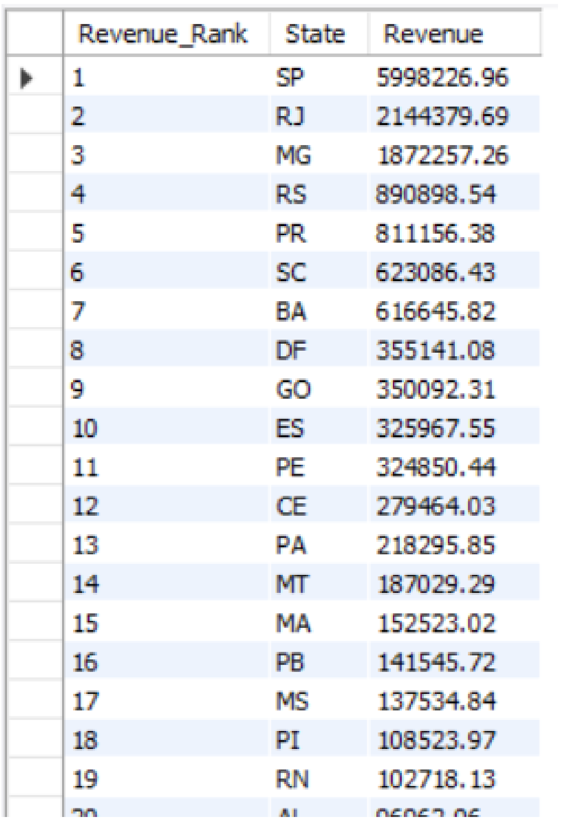
Points:

- The table shows the Percentage of Customer Churn Rate.
- An approximate churn rate of 97% indicates that most customers placed only one order. This could be due to a rise in new customers or a decline in returning customers.

Revenue Rank by State

```
select rank() over(order by revenue desc) as Revenue_Rank, State, Revenue
from(
select c.customer_state as State, round(sum(p.payment_value),2) as Revenue
from customers c
join orders o on c.customer_id = o.customer_id
join payments p on p.order_id = o.order_id
group by State
) sub
order by Revenue_Rank;
```

Output:



Revenue_Rank	State	Revenue
1	SP	5998226.96
2	RJ	2144379.69
3	MG	1872257.26
4	RS	890898.54
5	PR	811156.38
6	SC	623086.43
7	BA	616645.82
8	DF	355141.08
9	GO	350092.31
10	ES	325967.55
11	PE	324850.44
12	CE	279464.03
13	PA	218295.85
14	MT	187029.29
15	MA	152523.02
16	PB	141545.72
17	MS	137534.84
18	PI	108523.97
19	RN	102718.13
20	AL	[illegible]

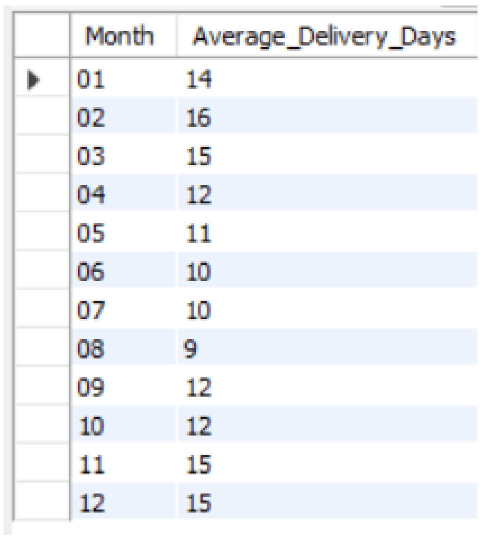
Points:

- The table shows the Ranking of States based on Revenue generated.
- The state 'SP' ranks first, generating revenue of more than 5.9M

Avg. Delivery by Month

```
select
        date_format(order_purchase_timestamp, '%m') as Month,
        round(avg(datediff(order_delivered_customer_date, order_purchase_timestamp))) as
Average_Delivery_Days
from orders
where order_delivered_customer_date is not null
group by month
order by month asc;
```

Output:



Month	Average_Delivery_Days
01	14
02	16
03	15
04	12
05	11
06	10
07	10
08	9
09	12
10	12
11	15
12	15

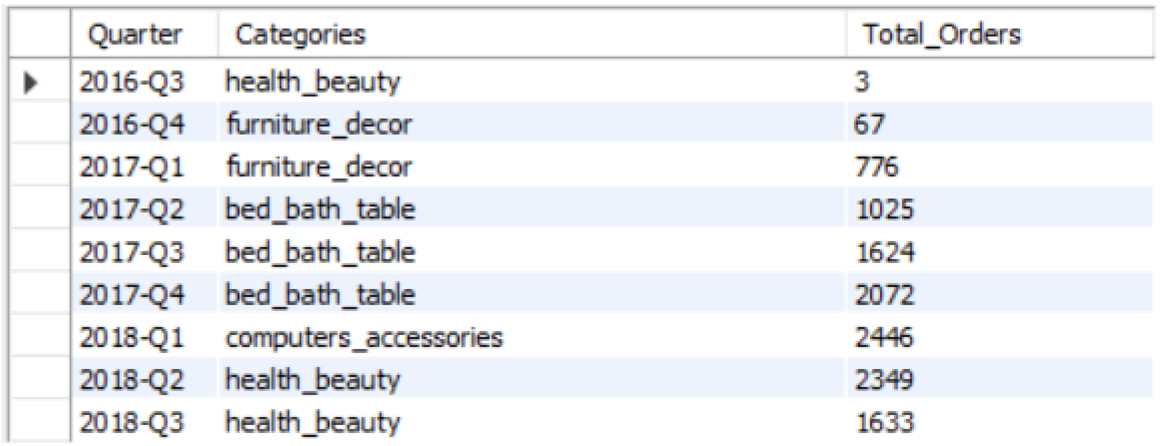
Points:

- The table shows the Average Delivery Days for each month of the year.
- Overall, the average delivery time ranges between 10–15 days.

Most Popular Product Category by Quarter

```
with category_quarter as(
select
        concat(year(o.order_purchase_timestamp), '-Q',
quarter(o.order_purchase_timestamp)) as Quarter,
        ct.product_category_name_english as Categories, count(o.order_id) as Total_Orders
from orders o
join items i on o.order_id = i.order_id
join products p on i.product_id = p.product_id
join category_translation ct on p.product_category_name = ct.product_category_name
group by quarter, categories
),
ranked as (
select
        quarter, categories, total_orders,
        row_number() over(partition by quarter order by total_orders desc) as rn
from category_quarter
)
select Quarter, Categories, Total_Orders
from ranked
where rn=1
order by quarter;
```

Output:



	Quarter	Categories	Total_Orders
▶	2016-Q3	health_beauty	3
	2016-Q4	furniture_decor	67
	2017-Q1	furniture_decor	776
	2017-Q2	bed_bath_table	1025
	2017-Q3	bed_bath_table	1624
	2017-Q4	bed_bath_table	2072
	2018-Q1	computers_accessories	2446
	2018-Q2	health_beauty	2349
	2018-Q3	health_beauty	1633

Points:

- The table shows the Popular Product Categories for each quarter of every year, reflecting customer preferences and seasonal trends.
- ‘Bed Bath Table’ and ‘Health Beauty’ received the highest number of orders in most quarters, indicating strong product demand.