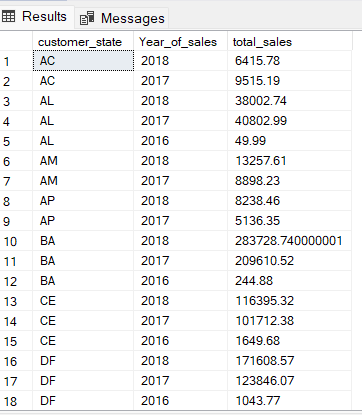
SQL Project - Ecommerce Sales Analysis

A] Create the different metrics like Sales, customer acquisitions, total no. of orders for each Year across the different states they serve.

**A.1] In the following query we have calculated the total sales over each state for each year.**

**For the analysis, we sum up the price of products for each state over each year, which we got from the order delivered date.**

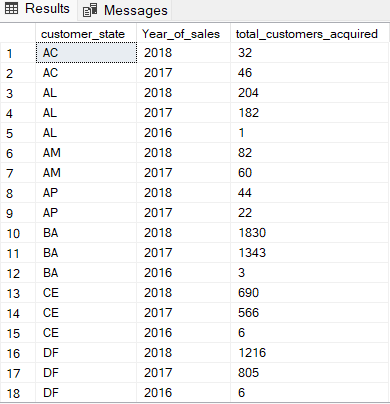
|  |
| --- |
| select b.customer\_state, year(a.order\_delivered\_customer\_date)as Year\_of\_sales , sum(c.price) as total\_sales from olist\_orders\_dataset as a left join olist\_customers\_dataset as b on a.customer\_id = b.customer\_id left join olist\_order\_items\_dataset as c on a.order\_id = c.order\_id where year(a.order\_delivered\_customer\_date) is not null group by year(a.order\_delivered\_customer\_date), b.customer\_state order by 1 , 2 desc |



**A.2] In the following query we have calculated the total customer acquisition over each state for each year**

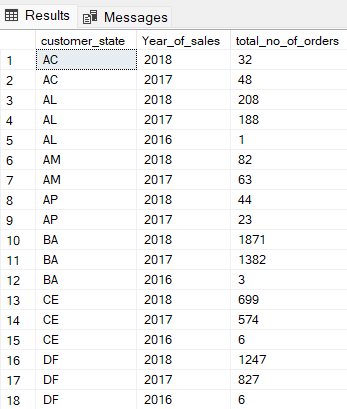
**For the analysis we have counted all the distinct customer unique ids for each state over each year, which we got from the customers dataset.**

|  |
| --- |
| select b.customer\_state, year(a.order\_delivered\_customer\_date) as Year\_of\_sales , count(distinct(b.customer\_unique\_id)) as total\_customers\_acquired from olist\_orders\_dataset as a left join olist\_customers\_dataset as b on a.customer\_id = b.customer\_id where year(a.order\_delivered\_customer\_date) is not null group by year(a.order\_delivered\_customer\_date), b.customer\_state order by 1 , 2 desc |



|  |
| --- |
| **A.3] In the following query we have calculated the total no. of orders each year state wise**  **For the analysis we have counted all the distinct order ids for each state over each year from the orders dataset.** |

|  |
| --- |
| select b.customer\_state, year(a.order\_delivered\_customer\_date)as Year\_of\_sales , count(distinct(a.order\_id)) as total\_no\_of\_orders from olist\_orders\_dataset as a left join olist\_customers\_dataset as b on a.customer\_id = b.customer\_id where year(a.order\_delivered\_customer\_date) is not null group by year(a.order\_delivered\_customer\_date), b.customer\_state order by 1 , 2 desc |



**A.4] These all graphs show the same trends mostly, there is not much disparity found among them. As in most of the part we can see that as each year passes the popularity among customers and sellers is increasing.**

---------------------------------------------------------------------------------------------------------------------------

B] Using the above metrics, identify the top 2 States which show

i. Declining trend over the years

ii. Increasing trend over the years

**B.1] In the following query we have used the no. of orders from part (a), and we have removed the count of that order part, also we have created a new table as trends. And inserted all the data into trends.**

|  |
| --- |
| create table trends( customer\_states varchar(50), year\_of\_sales int, orders datetime2, ) insert into trends  select b.customer\_state, year(a.order\_delivered\_customer\_date)as Year\_of\_sales , a.order\_approved\_at as orders from olist\_orders\_dataset as a left join olist\_customers\_dataset as b on a.customer\_id = b.customer\_id where year(a.order\_delivered\_customer\_date) is not null group by year(a.order\_delivered\_customer\_date), b.customer\_state, a.order\_approved\_at order by 1 , 2 desc  select \* from trends    **B.2]** **In the following query we have pivoted the trends table, as such, so that we can see the count of orders for each year state wise. And also we have inserted this table into a new table named trends2.** create table trends2( customer\_states varchar(50), \_2018 int, \_2017 int, \_2016 int ) insert into trends2 select \* from ( select customer\_states, year\_of\_sales as years, orders from trends ) t pivot(  count (orders) for years in( [2018], [2017], [2016]) )as pivot\_table select \* from trends2 |

**B.3] In the following query we have calculated the declining trends of the top 2 states over the years.**

**For the analysis we have used trends2 table and found the change in 2018 and 2017 and ordered our table in descending order according to the changes.**

|  |
| --- |
| select top 2 customer\_states, \_2018, \_2017, \_2016, (\_2018 - \_2017) as change\_in\_2018, (\_2017 - \_2016) as change\_in\_2017 from trends2 group by customer\_states, \_2018, \_2017, \_2016 order by 5 asc, 6asc |

**B.4] In the following query we have calculated the increasing trends of the top 2 states over the years.**

**For the analysis we have used trends2 table and found the change in 2018 and 2017 and ordered our table in descending order according to the changes.**

|  |
| --- |
| select top 2 customer\_states,\_2018,\_2017,\_2016, (\_2018 - \_2017) as change\_in\_2018, (\_2017 - \_2016) as change\_in\_2017 from trends2 group by customer\_states, \_2018,\_2017,\_2016 order by 5 desc,6 desc |

----------------------------------------------------------------------------------------------------------------------------

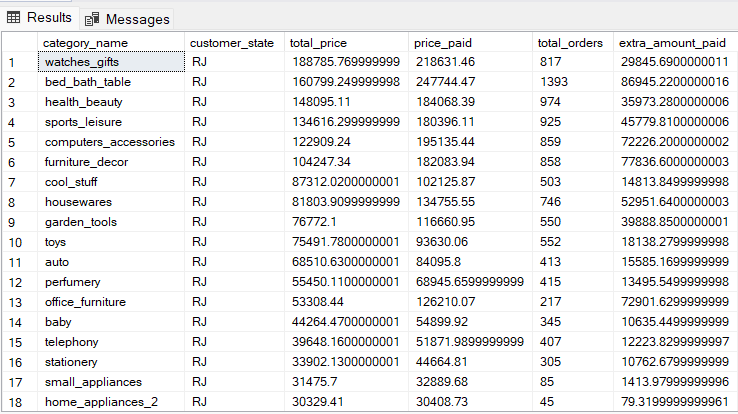
|  |
| --- |
| C] For the States identified above, do the Root Cause analysis for their performance across a variety of metrics.  You can utilize the following metrics and explore a few yourself as well by analyzing the data.  Category level Sales and orders placed, post-order reviews, Seller performance in terms of deliveries, product-level sales & orders placed,  % of orders delivered earlier than the expected date, % of orders delivered later than the expected date, etc.  **--Category level sales and orders placed C.1.a] In the following query we have calculated the category level sales and orders placed for declining trend states.**  **For the analysis we have shown the total price and price paid by the customer on the orders, and total number of orders. Also, we have shown the extra amount which customers have to pay.** |

|  |
| --- |
| select c.column2 as category\_name, e.customer\_state, sum(a.price) as total\_price ,sum(f.payment\_value) as price\_paid, count(distinct(d.order\_id)) as total\_orders, (sum(f.payment\_value) - sum(a.price)) as extra\_amount\_paid from olist\_order\_items\_dataset as a left join olist\_products\_dataset as b on a.product\_id = b.product\_id join product\_category\_name\_translation as c on b.product\_category\_name = c.column1 left join olist\_orders\_dataset as d on a.order\_id = d.order\_id left join olist\_customers\_dataset as e on d.customer\_id = e.customer\_id left join olist\_order\_payments\_dataset as f on a.order\_id = f.order\_id where e.customer\_state = 'AC' or e.customer\_state = 'RO' group by c.column2, e.customer\_state order by 2 asc, 3 desc, 4 desc, 5 desc |

**C.1.b] In the following query we have calculated the category level sales and orders placed for increasing trend states.**

**For the analysis we have shown the total price and price paid by the customer on the orders , and total number of orders. Also we have shown the extra amount which customers have to pay.**

|  |
| --- |
| select c.column2 as category\_name, e.customer\_state, sum(a.price) as total\_price , sum(f.payment\_value) as price\_paid, count(distinct(d.order\_id)) as total\_orders,  (sum(f.payment\_value) - sum(a.price)) as extra\_amount\_paid from olist\_order\_items\_dataset as a left join olist\_products\_dataset as b on a.product\_id = b.product\_id join product\_category\_name\_translation as c on b.product\_category\_name = c.column1 left join olist\_orders\_dataset as d on a.order\_id = d.order\_id left join olist\_customers\_dataset as e on d.customer\_id = e.customer\_id left join olist\_order\_payments\_dataset as f on a.order\_id = f.order\_id where e.customer\_state = 'SP' or e.customer\_state = 'RJ' group by c.column2, e.customer\_state order by 2 asc, 3 desc, 4 desc, 5 desc |



|  |
| --- |
| **--Post order review** |

**C.2] In the following query we have calculated the category level sales and orders placed for declining and increasing trend states.**

**For the analysis we have created a table as post\_order\_decline, which gives us the customer state and the no. of different stars each state got in customer reviews. In the second query we have shown the percent of positive reviews and negative reviews given by the customers also the average days it takes the seller to reply to those reviews.**

|  |
| --- |
| create table post\_order\_decline( customer\_state varchar(50), \_5star float, \_4star float, \_3star float, \_2star float, \_1star float)  insert into post\_order\_decline select c.customer\_state, sum(case when b.review\_score = 5 then 1 else 0 end) as \_5star , sum(case when b.review\_score = 4 then 1 else 0 end) as \_4star , sum(case when b.review\_score = 3 then 1 else 0 end) as \_3star , sum(case when b.review\_score = 2 then 1 else 0 end) as \_2star , sum(case when b.review\_score = 1 then 1 else 0 end) as \_1star  from olist\_orders\_dataset as a left join olist\_order\_reviews\_dataset as b on a.order\_id = b.order\_id left join olist\_customers\_dataset as c on a.customer\_id = c.customer\_id where c.customer\_state in ('SP','RJ','AC','RO') group by c.customer\_state  select \* from post\_order\_decline    select a.\*, ((a.\_5star+a.\_4star) / (a.\_5star+a.\_4star+a.\_3star+a.\_2star+a.\_1star))\*100 as percent\_of\_positive\_reviews,  (100 -((a.\_5star+a.\_4star) / (a.\_5star+a.\_4star+a.\_3star+a.\_2star+a.\_1star))\*100 ) as percent\_of\_negeative\_reciews, avg(datediff(day,d.review\_creation\_date,d.review\_answer\_timestamp)) as avg\_days\_to\_answer\_reviews from post\_order\_decline as a left join olist\_customers\_dataset as b on a.customer\_state = b.customer\_state left join olist\_orders\_dataset as c on b.customer\_id = c.customer\_id left join olist\_order\_reviews\_dataset as d on c.order\_id = d.order\_id group by a.customer\_state,a.\_5star,a.\_4star,a.\_3star,a.\_2star,a.\_1star order by 2,3 |

|  |
| --- |
| **-- Seller performance in terms of deliveries** |

**C.3.a] In the following query we have calculated the category level sales and orders placed for declining trend states.**

**For the analysis we have shown the seller ids for those sellers who have their state and their customer state in the declining trends states. Also we have shown the days to deliver and expected delivery days in our table. On the basis of that we have categorized our sellers performance as good or poor.**

|  |
| --- |
| select distinct(a.seller\_id), a.seller\_state, d.customer\_state, DATEDIFF( day, c.order\_delivered\_carrier\_date,c.order\_delivered\_customer\_date) as days\_to\_deliver,  datediff(day,c.order\_purchase\_timestamp,c.order\_estimated\_delivery\_date) as expected\_days, case when ( DATEDIFF( day, c.order\_delivered\_carrier\_date,c.order\_delivered\_customer\_date) - datediff(day,c.order\_purchase\_timestamp,c.order\_estimated\_delivery\_date)) <0 then 'good' when ( DATEDIFF( day, c.order\_delivered\_carrier\_date,c.order\_delivered\_customer\_date) - datediff(day,c.order\_purchase\_timestamp,c.order\_estimated\_delivery\_date)) >=0 then 'poor'  end as performance from olist\_sellers\_dataset as a left join olist\_order\_items\_dataset as b on a.seller\_id = b.seller\_id left join olist\_orders\_dataset as c on b.order\_id = c.order\_id left join olist\_customers\_dataset as d on c.customer\_id = d.customer\_id where a.seller\_state in('AC','RO') or d.customer\_state in ('AC','RO') and DATEDIFF( day, c.order\_delivered\_carrier\_date,c.order\_delivered\_customer\_date) is not null order by 3 asc, 4 desC |

**C.3.b] In the following query we have calculated the category level sales and orders placed for increasing trend states.**

**For the analysis we have shown the seller ids for those sellers who have their state and their customer state in the increasing trends states. Also we have shown the days to deliver and expected delivery days in our table. On the basis of that we have categorized our sellers performance as good or poor.**

|  |
| --- |
| select distinct(a.seller\_id), a.seller\_state, d.customer\_state,  DATEDIFF( day, c.order\_delivered\_carrier\_date,c.order\_delivered\_customer\_date) as days\_to\_deliver, datediff(day,c.order\_purchase\_timestamp,c.order\_estimated\_delivery\_date) as expected\_days, case when ( DATEDIFF( day, c.order\_delivered\_carrier\_date,c.order\_delivered\_customer\_date) - datediff(day,c.order\_purchase\_timestamp,c.order\_estimated\_delivery\_date)) <0 then 'good' when ( DATEDIFF( day, c.order\_delivered\_carrier\_date,c.order\_delivered\_customer\_date) - datediff(day,c.order\_purchase\_timestamp,c.order\_estimated\_delivery\_date)) >=0 then 'poor'  end as performance from olist\_sellers\_dataset as a left join olist\_order\_items\_dataset as b on a.seller\_id = b.seller\_id left join olist\_orders\_dataset as c on b.order\_id = c.order\_id left join olist\_customers\_dataset as d on c.customer\_id = d.customer\_id where a.seller\_state in('SP','RJ') or d.customer\_state in ('SP','RJ') and DATEDIFF( day, c.order\_delivered\_carrier\_date,c.order\_delivered\_customer\_date) is not null order by 3 asc, 4 desc |

|  |
| --- |
| **-- Product level sales and orders placed** |

**C.4.a] In the following query we have calculated the product level sales and orders placed for declining trend state (AC).**

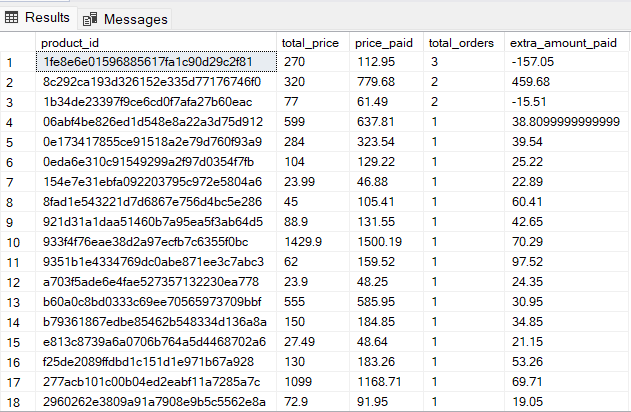
**For the analysis we have first figured the city which is causing the trend in AC. Then in the second query we have shown all the product ids of that category, in that state, while also showing the total price and price paid by the customer. Total orders of each product, and the extra amount which customers have to pay in each order.**

|  |
| --- |
| select top 1 c.column2 as category\_name, e.customer\_state, count(distinct(d.order\_id)) as total\_orders from olist\_order\_items\_dataset as a left join olist\_products\_dataset as b on a.product\_id = b.product\_id join product\_category\_name\_translation as c on b.product\_category\_name = c.column1 left join olist\_orders\_dataset as d on a.order\_id = d.order\_id left join olist\_customers\_dataset as e on d.customer\_id = e.customer\_id left join olist\_order\_payments\_dataset as f on a.order\_id = f.order\_id where e.customer\_state = 'AC' group by c.column2, e.customer\_state order by 2 asc , 3 desc    select a.product\_id, sum(c.price) as total\_price, sum(d.payment\_value) as price\_paid, count(d.order\_id) as total\_orders,(sum(d.payment\_value) - sum(c.price)) as extra\_amount\_paid  from olist\_products\_dataset as a join product\_category\_name\_translation as b on a.product\_category\_name = b.column1 left join olist\_order\_items\_dataset as c on a.product\_id = c.product\_id left join olist\_order\_payments\_dataset as d on c.order\_id = d.order\_id left join olist\_orders\_dataset as e  on c.order\_id =e.order\_id left join olist\_customers\_dataset as f on e.customer\_id = f.customer\_id where b.column2 = 'Sports\_leisure' and f.customer\_state = 'AC' group by a.product\_id order by 4 desc |

**C.4.b] In the following query we have calculated the product level sales and orders placed for declining trend state (RO).**

**For the analysis we have first figured the city which is causing the trend in RO. Then in the second query we have shown all the product ids of that category, in that state, while also showing the total price and price paid by the customer. Total orders of each product, and the extra amount which customers have to pay in each order.**

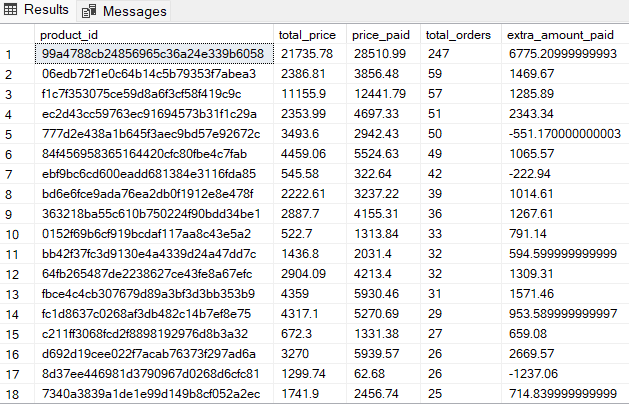
|  |
| --- |
| select top 1 c.column2 as category\_name, e.customer\_state, count(distinct(d.order\_id)) as total\_orders from olist\_order\_items\_dataset as a left join olist\_products\_dataset as b on a.product\_id = b.product\_id join product\_category\_name\_translation as c on b.product\_category\_name = c.column1 left join olist\_orders\_dataset as d on a.order\_id = d.order\_id left join olist\_customers\_dataset as e on d.customer\_id = e.customer\_id left join olist\_order\_payments\_dataset as f on a.order\_id = f.order\_id where e.customer\_state = 'RO' group by c.column2, e.customer\_state order by 2 asc , 3 desc    select a.product\_id, sum(c.price) as total\_price, sum(d.payment\_value) as price\_paid, count(d.order\_id) as total\_orders,(sum(d.payment\_value) - sum(c.price)) as extra\_amount\_paid  from olist\_products\_dataset as a join product\_category\_name\_translation as b on a.product\_category\_name = b.column1 left join olist\_order\_items\_dataset as c on a.product\_id = c.product\_id left join olist\_order\_payments\_dataset as d on c.order\_id = d.order\_id left join olist\_orders\_dataset as e  on c.order\_id =e.order\_id left join olist\_customers\_dataset as f on e.customer\_id = f.customer\_id where b.column2 = 'health\_beauty' and f.customer\_state = 'RO' group by a.product\_id order by 4 desc |



**C.4.c] In the following query we have calculated the product level sales and orders placed for increasing trend state (SP).**

**For the analysis we have first figured the city which is causing the trend in SP. Then in the second query we have shown all the product ids of that category, in that state, while also showing the total price and price paid by the customer. Total orders of each product, and the extra amount which customers have to pay in each order.**

|  |
| --- |
| select top 1 c.column2 as category\_name, e.customer\_state, count(distinct(d.order\_id)) as total\_orders from olist\_order\_items\_dataset as a left join olist\_products\_dataset as b on a.product\_id = b.product\_id join product\_category\_name\_translation as c on b.product\_category\_name = c.column1 left join olist\_orders\_dataset as d on a.order\_id = d.order\_id left join olist\_customers\_dataset as e on d.customer\_id = e.customer\_id left join olist\_order\_payments\_dataset as f on a.order\_id = f.order\_id where e.customer\_state = 'SP' group by c.column2, e.customer\_state order by 2 asc , 3 desc    select a.product\_id, sum(c.price) as total\_price, sum(d.payment\_value) as price\_paid, count(d.order\_id) as total\_orders,(sum(d.payment\_value) - sum(c.price)) as extra\_amount\_paid  from olist\_products\_dataset as a join product\_category\_name\_translation as b on a.product\_category\_name = b.column1 left join olist\_order\_items\_dataset as c on a.product\_id = c.product\_id left join olist\_order\_payments\_dataset as d on c.order\_id = d.order\_id left join olist\_orders\_dataset as e  on c.order\_id =e.order\_id left join olist\_customers\_dataset as f on e.customer\_id = f.customer\_id where b.column2 = 'bed\_bath\_table' and f.customer\_state = 'SP' group by a.product\_id order by 4 desc |



**C.4.d] In the following query we have calculated the product level sales and orders placed for increasing trend state (RJ).**

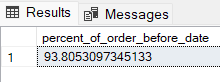
**For the analysis we have first figured the city which is causing the trend in RJ. Then in the second query we have shown all the product ids of that category, in that state, while also showing the total price and price paid by the customer. Total orders of each product, and the extra amount which customers have to pay in each order.**

|  |
| --- |
| select top 1 c.column2 as category\_name, e.customer\_state, count(distinct(d.order\_id)) as total\_orders from olist\_order\_items\_dataset as a left join olist\_products\_dataset as b on a.product\_id = b.product\_id join product\_category\_name\_translation as c on b.product\_category\_name = c.column1 left join olist\_orders\_dataset as d on a.order\_id = d.order\_id left join olist\_customers\_dataset as e on d.customer\_id = e.customer\_id left join olist\_order\_payments\_dataset as f on a.order\_id = f.order\_id where e.customer\_state = 'RJ' group by c.column2, e.customer\_state order by 2 asc , 3 desc    select a.product\_id, sum(c.price) as total\_price, sum(d.payment\_value) as price\_paid, count(d.order\_id) as total\_orders,(sum(d.payment\_value) - sum(c.price)) as extra\_amount\_paid  from olist\_products\_dataset as a join product\_category\_name\_translation as b on a.product\_category\_name = b.column1 left join olist\_order\_items\_dataset as c on a.product\_id = c.product\_id left join olist\_order\_payments\_dataset as d on c.order\_id = d.order\_id left join olist\_orders\_dataset as e  on c.order\_id =e.order\_id left join olist\_customers\_dataset as f on e.customer\_id = f.customer\_id where b.column2 = 'bed\_bath\_table' and f.customer\_state = 'RJ' group by a.product\_id order by 4 desc |

|  |
| --- |
| **-- % of order delivered earlier than expected date** |

**For the analysis of declining trend states we have created table perf\_decline\_earier. Which gives us all the attributes achieved by seller performance analysis, the only thing we have changed is if the performance of a seller is good then it is represented as 1 and if performance is poor it is represented as 0.**

|  |
| --- |
| create table perf\_decline\_earlier (seller\_id varchar(50), seller\_state varchar(50), customer\_state varchar(40), days\_to\_deliver int, expected\_days int, performance int ) insert into perf\_decline\_earlier select distinct(a.seller\_id), a.seller\_state, d.customer\_state, DATEDIFF( day, c.order\_delivered\_carrier\_date,c.order\_delivered\_customer\_date) as days\_to\_deliver, datediff(day,c.order\_purchase\_timestamp,c.order\_estimated\_delivery\_date) as expected\_days, case when ( DATEDIFF( day, c.order\_delivered\_carrier\_date,c.order\_delivered\_customer\_date) - datediff(day,c.order\_purchase\_timestamp,c.order\_estimated\_delivery\_date)) <0 then 1 when ( DATEDIFF( day, c.order\_delivered\_carrier\_date,c.order\_delivered\_customer\_date) - datediff(day,c.order\_purchase\_timestamp,c.order\_estimated\_delivery\_date)) >=0 then 0  end as performance from olist\_sellers\_dataset as a left join olist\_order\_items\_dataset as b on a.seller\_id = b.seller\_id left join olist\_orders\_dataset as c on b.order\_id = c.order\_id left join olist\_customers\_dataset as d on c.customer\_id = d.customer\_id where a.seller\_state in('AC','RO') or d.customer\_state in ('AC','RO') and DATEDIFF( day, c.order\_delivered\_carrier\_date,c.order\_delivered\_customer\_date) is not null order by 3 asc, 4 desc  select \* from perf\_decline\_earlier    **C.5.a] In the following query we have calculated the % of order delivered earlier than the expected date for the declining trend states.** alter table perf\_decline\_earlier alter column performance float select (sum(case when performance = 1 then 1 else 0 end)\*100)/ (select (cast (count(performance) as float)) from perf\_decline\_earlier) as percent\_of\_order\_before\_date from perf\_decline\_earlier where customer\_state in ('AC','RO') |



**For the analysis of increasing trend states we have created table perf\_increase\_earier. Which gives us all the attributes achieved by seller performance analysis, the only thing we have changed is if the performance of a seller is good then it is represented as 1 and if performance is poor it is represented as 0.**

|  |
| --- |
| create table perf\_increase\_earlier ( seller\_id varchar(50), seller\_state varchar(50), customer\_state varchar(40), days\_to\_deliver int, expected\_days int, performance int ) insert into perf\_increase\_earlier select distinct(a.seller\_id), a.seller\_state, d.customer\_state, DATEDIFF( day, c.order\_delivered\_carrier\_date,c.order\_delivered\_customer\_date) as days\_to\_deliver, datediff(day,c.order\_purchase\_timestamp,c.order\_estimated\_delivery\_date) as expected\_days,case when ( DATEDIFF( day, c.order\_delivered\_carrier\_date,c.order\_delivered\_customer\_date) - datediff(day,c.order\_purchase\_timestamp,c.order\_estimated\_delivery\_date)) <0 then 1 when ( DATEDIFF( day, c.order\_delivered\_carrier\_date,c.order\_delivered\_customer\_date) - datediff(day,c.order\_purchase\_timestamp,c.order\_estimated\_delivery\_date)) >=0 then 0  end as performance from olist\_sellers\_dataset as a left join olist\_order\_items\_dataset as b on a.seller\_id = b.seller\_id left join olist\_orders\_dataset as c on b.order\_id = c.order\_id left join olist\_customers\_dataset as d on c.customer\_id = d.customer\_id where a.seller\_state in('SP','RJ') or d.customer\_state in ('SP','RJ') and DATEDIFF( day, c.order\_delivered\_carrier\_date,c.order\_delivered\_customer\_date) is not null order by 3 asc, 4 desc  select \* from perf\_increase\_earlier    **C.5.b] In the following query we have calculated the % of order delivered earlier than the expected date for the increasing trend states.** alter table perf\_increase\_earlier alter column performance float select (sum(case when performance = 1 then 1 else 0 end)\*100)/ (select (cast (count(performance) as float)) from perf\_increase\_earlier) as percent\_of\_order\_before\_date from perf\_increase\_earlier where customer\_state in ('SP','RJ') |

|  |
| --- |
| **-- % of orders delivered later than expected date** |

**C.6.a] In the following query we have calculated the % of order delivered after the expected date for the declining states.**

|  |
| --- |
| select \* from perf\_decline\_earlier alter table perf\_decline\_earlier alter column performance float select (100 -(sum(case when performance = 1 then 1 else 0 end)\*100)/ (select (cast (count(performance) as float)) from perf\_decline\_earlier) )as percent\_of\_order\_after\_date from perf\_decline\_earlier  where customer\_state in ('AC','RO') |

**C.5.b] In the following query we have calculated the % of order delivered after the expected date for the increasing trend states.**

|  |
| --- |
| select \* from perf\_increase\_earlier alter table perf\_increase\_earlier alter column performance float select (100 - (sum(case when performance = 1 then 1 else 0 end)\*100)/ (select (cast (count(performance) as float)) from perf\_increase\_earlier) )as percent\_of\_order\_after\_date from perf\_increase\_earlier where customer\_state in ('SP','RJ') |

|  |
| --- |
| **-- Analysis over location** |

**In the following query we have shown the number of sellers and customers available in each state.**

|  |
| --- |
| select a.seller\_state, count(a.seller\_id) from olist\_sellers\_dataset as a group by a.seller\_state order by 2 desc    select customer\_state , count(customer\_id) from olist\_customers\_dataset group by customer\_state order by 2 desc |

----------------------------------------------------------------------------------------------------------------------------

D] Do the above analysis for the top 2 cities which are causing the trend for each of the states identified in point (b).

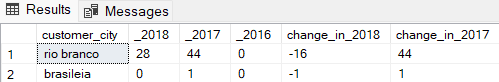
**D.1] In the following query we have created a table trends\_city, which consists of all the cities having year wise and information about all the orders ordered repectively.**

|  |
| --- |
| create table trends\_city( customer\_city varchar(50), year\_of\_sales int, orders datetime2) insert into trends\_city  select b.customer\_city, year(a.order\_delivered\_customer\_date)as Year\_of\_sales , a.order\_approved\_at as orders from olist\_orders\_dataset as a left join olist\_customers\_dataset as b on a.customer\_id = b.customer\_id where year(a.order\_delivered\_customer\_date) is not null group by year(a.order\_delivered\_customer\_date), b.customer\_city, a.order\_approved\_at order by 1 , 2 desc  select \* from trends\_city    **D.2] In the following query we have created a table trends2\_city, which consists of all the cities and the count of orders yearwise.**  create table trends2\_city( customer\_city varchar(50), \_2018 int, \_2017 int, \_2016 int) insert into trends2\_city select \* from ( select customer\_city, year\_of\_sales as years, orders from trends\_city ) t pivot(  count (orders) for years in( [2018], [2017], [2016]) )as pivot\_table  select \* from trends2\_city |

**D.3] In the following query we have found the top 2 cities for declining trend state AC**

**For the analysis we have found the change of number of orders in different years. And filtered it on the basis of state, to get the top cities of that state.**

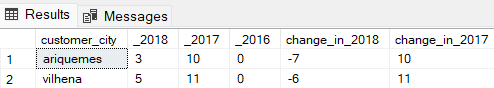
|  |
| --- |
| select top 2 a.customer\_city,a.\_2018,a.\_2017,a.\_2016, (a.\_2018 - a.\_2017) as change\_in\_2018, (a.\_2017 - a.\_2016) as change\_in\_2017  from trends2\_city as a left join olist\_customers\_dataset as b on a.customer\_city = b.customer\_city where b.customer\_state = 'AC' group by a.customer\_city, a.\_2018,a.\_2017,a.\_2016 order by 5 asc, 6asc |



**D.4] In the following query we have found the top 2 cities for declining trend state RO**

**For the analysis we have found the change of number of orders in different years. And filtered it on the basis of state, to get the top cities of that state.**

|  |
| --- |
| select top 2 a.customer\_city,a.\_2018,a.\_2017,a.\_2016, (a.\_2018 - a.\_2017) as change\_in\_2018, (a.\_2017 - a.\_2016) as change\_in\_2017  from trends2\_city as a left join olist\_customers\_dataset as b on a.customer\_city = b.customer\_city where b.customer\_state = 'RO' group by a.customer\_city, a.\_2018,a.\_2017,a.\_2016 order by 5 asc, 6asc |



**D.5] In the following query we have found the top 2 cities for increasing trend state SP**

**For the analysis we have found the change of number of orders in different years. And filtered it on the basis of state, to get the top cities of that state.**

|  |
| --- |
| select top 2 a.customer\_city,a.\_2018,a.\_2017,a.\_2016, (a.\_2018 - a.\_2017) as change\_in\_2018, (a.\_2017 - a.\_2016) as change\_in\_2017  from trends2\_city as a left join olist\_customers\_dataset as b on a.customer\_city = b.customer\_city where b.customer\_state = 'SP' group by a.customer\_city, a.\_2018,a.\_2017,a.\_2016 order by 5 desc, 6desc |

**D.6] In the following query we have found the top 2 cities for increasing trend state RJ**

**For the analysis we have found the change of number of orders in different years. And filtered it on the basis of state, to get the top cities of that state.**

|  |
| --- |
| select top 2 a.customer\_city,a.\_2018,a.\_2017,a.\_2016, (a.\_2018 - a.\_2017) as change\_in\_2018, (a.\_2017 - a.\_2016) as change\_in\_2017  from trends2\_city as a left join olist\_customers\_dataset as b on a.customer\_city = b.customer\_city where b.customer\_state = 'RJ' group by a.customer\_city, a.\_2018,a.\_2017,a.\_2016 order by 5 desc, 6desc |

**In the following queries, we have done the same analysis for the cities as we did for the states.**

|  |
| --- |
| **-- For declining trend  -- In state AC** |

**In the following query we have found the seller performance in terms of deliveries, for declining trend state AC , and the top 2 cities.**

**For the analysis we got the seller ids and seller city and customer city in the top 2 cities we got from before analysis, also we have shown the days to deliver and expected days to deliver for the sellers in those cities, and given them the performance according to that only.**

|  |
| --- |
| select distinct(a.seller\_id), a.seller\_city, d.customer\_city, DATEDIFF( day, c.order\_delivered\_carrier\_date,c.order\_delivered\_customer\_date) as days\_to\_deliver,  datediff(day,c.order\_purchase\_timestamp,c.order\_estimated\_delivery\_date) as expected\_days, case when ( DATEDIFF( day, c.order\_delivered\_carrier\_date,c.order\_delivered\_customer\_date) - datediff(day,c.order\_purchase\_timestamp,c.order\_estimated\_delivery\_date)) <0 then 'good' when ( DATEDIFF( day, c.order\_delivered\_carrier\_date,c.order\_delivered\_customer\_date) - datediff(day,c.order\_purchase\_timestamp,c.order\_estimated\_delivery\_date)) >=0 then 'poor'  end as performance from olist\_sellers\_dataset as a left join olist\_order\_items\_dataset as b on a.seller\_id = b.seller\_id left join olist\_orders\_dataset as c on b.order\_id = c.order\_id left join olist\_customers\_dataset as d on c.customer\_id = d.customer\_id where (a.seller\_state = 'AC' or d.customer\_state = 'AC') and (a.seller\_city in ('brasileia' ,'rio branco') or d.customer\_city in ('brasileia' ,'rio branco')) and DATEDIFF( day, c.order\_delivered\_carrier\_date,c.order\_delivered\_customer\_date) is not null order by 3 asc, 4 desc |

|  |
| --- |
| **-- For city rio\_branco in state AC** |

**In the following query we have found category level sales and orders placed for city rio branco in the state AC.**

**For the analysis we have shown category name, state, customer city, total price of the orders, the total payment value given by customers, total number of orders, and the extra payment paid by the customers.**

|  |
| --- |
| select c.column2 as category\_name, e.customer\_state,e.customer\_city, sum(a.price) as total\_price ,sum(f.payment\_value) as price\_paid, count(distinct(d.order\_id)) as total\_orders, (sum(f.payment\_value) - sum(a.price)) as extra\_amount\_paid from olist\_order\_items\_dataset as a left join olist\_products\_dataset as b on a.product\_id = b.product\_id join product\_category\_name\_translation as c on b.product\_category\_name = c.column1 left join olist\_orders\_dataset as d on a.order\_id = d.order\_id left join olist\_customers\_dataset as e on d.customer\_id = e.customer\_id left join olist\_order\_payments\_dataset as f on a.order\_id = f.order\_id where e.customer\_state = 'AC' and e.customer\_city = 'rio branco'  group by c.column2, e.customer\_state, e.customer\_city order by 2 asc, 3 desc, 4 desc, 5 desc |

**In the following query we have found product level sales and orders placed for city rio branco in the state AC.**

**For the analysis we have found the category name which has the most number of orders, in that city and state.**

|  |
| --- |
| select top 1 c.column2 as category\_name, e.customer\_state,e.customer\_city, count(distinct(d.order\_id)) as total\_orders from olist\_order\_items\_dataset as a left join olist\_products\_dataset as b on a.product\_id = b.product\_id join product\_category\_name\_translation as c on b.product\_category\_name = c.column1 left join olist\_orders\_dataset as d on a.order\_id = d.order\_id left join olist\_customers\_dataset as e on d.customer\_id = e.customer\_id left join olist\_order\_payments\_dataset as f on a.order\_id = f.order\_id where e.customer\_state = 'AC' and e.customer\_city = 'rio branco' group by c.column2, e.customer\_state, e.customer\_city order by 2 asc , 4 desc    **For the analysis we have all the product ids for that city in the state. Also the total price of the order and the price paid by the customer. We have counted the total number of orders and the extra amount customers have to pay on that product.**  select a.product\_id, sum(c.price) as total\_price, sum(d.payment\_value) as price\_paid, count(d.order\_id) as total\_orders,(sum(d.payment\_value) - sum(c.price)) as extra\_amount\_paid  from olist\_products\_dataset as a join product\_category\_name\_translation as b on a.product\_category\_name = b.column1 left join olist\_order\_items\_dataset as c on a.product\_id = c.product\_id left join olist\_order\_payments\_dataset as d on c.order\_id = d.order\_id left join olist\_orders\_dataset as e  on c.order\_id =e.order\_id left join olist\_customers\_dataset as f on e.customer\_id = f.customer\_id where b.column2 = 'furniture\_decor' and f.customer\_state = 'AC' and f.customer\_city = 'rio branco' group by a.product\_id order by 4 desc |

|  |
| --- |
| **-- For city brasileia in state AC** |

**In the following query we have found category level sales and orders placed for city brasileia in the state AC.**

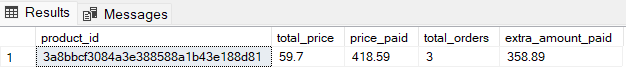
**For the analysis we have shown category name, state, customer city, total price of the orders, the total payment value given by customers, total number of orders, and the extra payment paid by the customers.**

|  |
| --- |
| select c.column2 as category\_name, e.customer\_state,e.customer\_city, sum(a.price) as total\_price ,sum(f.payment\_value) as price\_paid, count(distinct(d.order\_id)) as total\_orders, (sum(f.payment\_value) - sum(a.price)) as extra\_amount\_paid from olist\_order\_items\_dataset as a left join olist\_products\_dataset as b on a.product\_id = b.product\_id join product\_category\_name\_translation as c on b.product\_category\_name = c.column1 left join olist\_orders\_dataset as d on a.order\_id = d.order\_id left join olist\_customers\_dataset as e on d.customer\_id = e.customer\_id left join olist\_order\_payments\_dataset as f on a.order\_id = f.order\_id where e.customer\_state = 'AC' and e.customer\_city = 'brasileia'  group by c.column2, e.customer\_state, e.customer\_city order by 2 asc, 3 desc, 4 desc, 5 desc |

**In the following query we have found product level sales and orders placed for city brasileia in the state AC.**

**For the analysis we have found the category name which has the most number of orders, in that city and state.**

|  |
| --- |
| select c.column2 as category\_name, e.customer\_state,e.customer\_city, count(distinct(d.order\_id)) as total\_orders from olist\_order\_items\_dataset as a left join olist\_products\_dataset as b on a.product\_id = b.product\_id join product\_category\_name\_translation as c on b.product\_category\_name = c.column1 left join olist\_orders\_dataset as d on a.order\_id = d.order\_id left join olist\_customers\_dataset as e on d.customer\_id = e.customer\_id left join olist\_order\_payments\_dataset as f on a.order\_id = f.order\_id where e.customer\_state = 'AC' and e.customer\_city = 'brasileia'  group by c.column2, e.customer\_state, e.customer\_city order by 2 asc , 4 desc    **For the analysis we have all the product ids for that city in the state. Also the total price of the order and the price paid by the customer. We have counted the total number of orders and the extra amount customers have to pay on that product.**  select a.product\_id, sum(c.price) as total\_price, sum(d.payment\_value) as price\_paid, count(d.order\_id) as total\_orders,(sum(d.payment\_value) - sum(c.price)) as extra\_amount\_paid  from olist\_products\_dataset as a join product\_category\_name\_translation as b on a.product\_category\_name = b.column1 left join olist\_order\_items\_dataset as c on a.product\_id = c.product\_id left join olist\_order\_payments\_dataset as d on c.order\_id = d.order\_id left join olist\_orders\_dataset as e  on c.order\_id =e.order\_id left join olist\_customers\_dataset as f on e.customer\_id = f.customer\_id where b.column2 = 'computers\_accessories' and f.customer\_state = 'AC' and f.customer\_city = 'brasileia'  group by a.product\_id order by 4 desc |



|  |
| --- |
| **-- In state RO** |

**In the following query we have found the seller performance in terms of deliveries, for declining trend state RO , and the top 2 cities.**

**For the analysis we got the seller ids and seller city and customer city in the top 2 cities we got from before analysis, also we have shown the days to deliver and expected days to deliver for the sellers in those cities, and given them the performance according to that only.**

|  |
| --- |
| select distinct(a.seller\_id), a.seller\_city, d.customer\_city, DATEDIFF( day, c.order\_delivered\_carrier\_date,c.order\_delivered\_customer\_date) as days\_to\_deliver,  datediff(day,c.order\_purchase\_timestamp,c.order\_estimated\_delivery\_date) as expected\_days, case when ( DATEDIFF( day, c.order\_delivered\_carrier\_date,c.order\_delivered\_customer\_date) - datediff(day,c.order\_purchase\_timestamp,c.order\_estimated\_delivery\_date)) <0 then 'good' when ( DATEDIFF( day, c.order\_delivered\_carrier\_date,c.order\_delivered\_customer\_date) - datediff(day,c.order\_purchase\_timestamp,c.order\_estimated\_delivery\_date)) >=0 then 'poor'  end as performance from olist\_sellers\_dataset as a left join olist\_order\_items\_dataset as b on a.seller\_id = b.seller\_id left join olist\_orders\_dataset as c on b.order\_id = c.order\_id left join olist\_customers\_dataset as d on c.customer\_id = d.customer\_id where (a.seller\_state = 'RO' or d.customer\_state = 'RO') and (a.seller\_city in ('ariquemes' ,'vilhena') or d.customer\_city in ('ariquemes' ,'vilhena')) and DATEDIFF( day, c.order\_delivered\_carrier\_date,c.order\_delivered\_customer\_date) is not null order by 3 asc, 4 desc |

|  |
| --- |
| **-- For city ariquemes** |

**In the following query we have found category level sales and orders placed for city ariquemes in the state RO.**

**For the analysis we have shown category name, state, customer city, total price of the orders, the total payment value given by customers, total number of orders, and the extra payment paid by the customers.**

|  |
| --- |
| select c.column2 as category\_name, e.customer\_state,e.customer\_city, sum(a.price) as total\_price ,sum(f.payment\_value) as price\_paid, count(distinct(d.order\_id)) as total\_orders, (sum(f.payment\_value) - sum(a.price)) as extra\_amount\_paid from olist\_order\_items\_dataset as a left join olist\_products\_dataset as b on a.product\_id = b.product\_id join product\_category\_name\_translation as c on b.product\_category\_name = c.column1 left join olist\_orders\_dataset as d on a.order\_id = d.order\_id left join olist\_customers\_dataset as e on d.customer\_id = e.customer\_id left join olist\_order\_payments\_dataset as f on a.order\_id = f.order\_id where e.customer\_state = 'RO' and e.customer\_city = 'ariquemes'  group by c.column2, e.customer\_state, e.customer\_city order by 2 asc, 3 desc, 4 desc, 5 desc |

**In the following query we have found product level sales and orders placed for city ariquemes in the state RO.**

**For the analysis we have found the category name which has the most number of orders, in that city and state.**

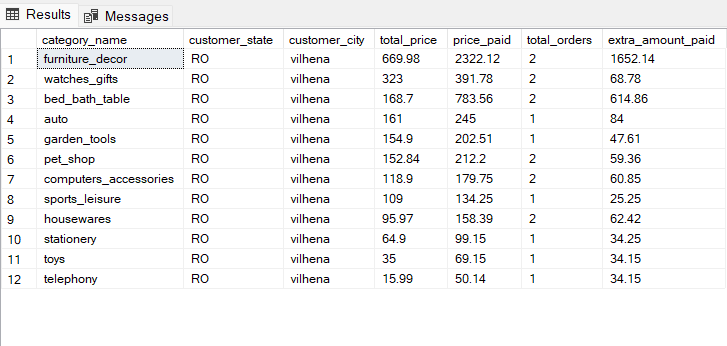
|  |
| --- |
| select top 1 c.column2 as category\_name, e.customer\_state,e.customer\_city, count(distinct(d.order\_id)) as total\_orders from olist\_order\_items\_dataset as a left join olist\_products\_dataset as b on a.product\_id = b.product\_id join product\_category\_name\_translation as c on b.product\_category\_name = c.column1 left join olist\_orders\_dataset as d on a.order\_id = d.order\_id left join olist\_customers\_dataset as e on d.customer\_id = e.customer\_id left join olist\_order\_payments\_dataset as f on a.order\_id = f.order\_id where e.customer\_state = 'RO' and e.customer\_city = 'ariquemes'  group by c.column2, e.customer\_state, e.customer\_city order by 2 asc , 4 desc    **For the analysis we have all the product ids for that city in the state. Also the total price of the order and the price paid by the customer. We have counted the total number of orders and the extra amount customers have to pay on that product.**  select a.product\_id, sum(c.price) as total\_price, sum(d.payment\_value) as price\_paid, count(d.order\_id) as total\_orders,(sum(d.payment\_value) - sum(c.price)) as extra\_amount\_paid  from olist\_products\_dataset as a join product\_category\_name\_translation as b on a.product\_category\_name = b.column1 left join olist\_order\_items\_dataset as c on a.product\_id = c.product\_id left join olist\_order\_payments\_dataset as d on c.order\_id = d.order\_id left join olist\_orders\_dataset as e  on c.order\_id =e.order\_id left join olist\_customers\_dataset as f on e.customer\_id = f.customer\_id where b.column2 = 'health\_beauty' and f.customer\_state = 'RO' and f.customer\_city = 'ariquemes'  group by a.product\_id order by 4 desc |

|  |
| --- |
| **-- For city vilhena** |

**In the following query we have found category level sales and orders placed for city vilhena in the state RO.**

**For the analysis we have shown category name, state, customer city, total price of the orders, the total payment value given by customers, total number of orders, and the extra payment paid by the customers.**

|  |
| --- |
| select c.column2 as category\_name, e.customer\_state,e.customer\_city, sum(a.price) as total\_price ,sum(f.payment\_value) as price\_paid, count(distinct(d.order\_id)) as total\_orders, (sum(f.payment\_value) - sum(a.price)) as extra\_amount\_paid from olist\_order\_items\_dataset as a left join olist\_products\_dataset as b on a.product\_id = b.product\_id join product\_category\_name\_translation as c on b.product\_category\_name = c.column1 left join olist\_orders\_dataset as d on a.order\_id = d.order\_id left join olist\_customers\_dataset as e on d.customer\_id = e.customer\_id left join olist\_order\_payments\_dataset as f on a.order\_id = f.order\_id where e.customer\_state = 'RO' and e.customer\_city = 'vilhena'  group by c.column2, e.customer\_state, e.customer\_city order by 2 asc, 3 desc, 4 desc, 5 desc |



**In the following query we have found product level sales and orders placed for city vilhena in the state RO.**

**For the analysis we have found the category name which has the most number of orders, in that city and state.**

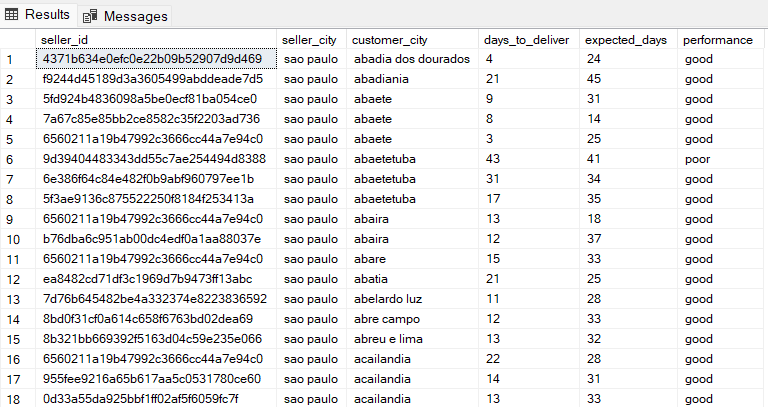
|  |
| --- |
| select top 1 c.column2 as category\_name, e.customer\_state,e.customer\_city, count(distinct(d.order\_id)) as total\_orders from olist\_order\_items\_dataset as a left join olist\_products\_dataset as b on a.product\_id = b.product\_id join product\_category\_name\_translation as c on b.product\_category\_name = c.column1 left join olist\_orders\_dataset as d on a.order\_id = d.order\_id left join olist\_customers\_dataset as e on d.customer\_id = e.customer\_id left join olist\_order\_payments\_dataset as f on a.order\_id = f.order\_id where e.customer\_state = 'RO' and e.customer\_city = 'vilhena' group by c.column2, e.customer\_state, e.customer\_city order by 2 asc , 4 desc    **For the analysis we have all the product ids for that city in the state. Also the total price of the order and the price paid by the customer. We have counted the total number of orders and the extra amount customers have to pay on that product.**  select a.product\_id, sum(c.price) as total\_price, sum(d.payment\_value) as price\_paid, count(d.order\_id) as total\_orders,(sum(d.payment\_value) - sum(c.price)) as extra\_amount\_paid  from olist\_products\_dataset as a join product\_category\_name\_translation as b on a.product\_category\_name = b.column1 left join olist\_order\_items\_dataset as c on a.product\_id = c.product\_id left join olist\_order\_payments\_dataset as d on c.order\_id = d.order\_id left join olist\_orders\_dataset as e  on c.order\_id =e.order\_id left join olist\_customers\_dataset as f on e.customer\_id = f.customer\_id where b.column2 = 'bed\_bath\_table' and f.customer\_state = 'RO' and f.customer\_city = 'vilhena' group by a.product\_id order by 4 desc |

|  |
| --- |
| **-- For increasing trend  -- In state SP** |

**In the following query we have found the seller performance in terms of deliveries, for increasing trend state SP , and the top 2 cities.**

**For the analysis we got the seller ids and seller city and customer city in the top 2 cities we got from before analysis, also we have shown the days to deliver and expected days to deliver for the sellers in those cities, and given them the performance according to that only.**

|  |
| --- |
| select distinct(a.seller\_id), a.seller\_city, d.customer\_city, DATEDIFF( day, c.order\_delivered\_carrier\_date,c.order\_delivered\_customer\_date) as days\_to\_deliver,  datediff(day,c.order\_purchase\_timestamp,c.order\_estimated\_delivery\_date) as expected\_days, case when ( DATEDIFF( day, c.order\_delivered\_carrier\_date,c.order\_delivered\_customer\_date) - datediff(day,c.order\_purchase\_timestamp,c.order\_estimated\_delivery\_date)) <0 then 'good' when ( DATEDIFF( day, c.order\_delivered\_carrier\_date,c.order\_delivered\_customer\_date) - datediff(day,c.order\_purchase\_timestamp,c.order\_estimated\_delivery\_date)) >=0 then 'poor'  end as performance from olist\_sellers\_dataset as a left join olist\_order\_items\_dataset as b on a.seller\_id = b.seller\_id left join olist\_orders\_dataset as c on b.order\_id = c.order\_id left join olist\_customers\_dataset as d on c.customer\_id = d.customer\_id where (a.seller\_state = 'SP' or d.customer\_state = 'SP') and (a.seller\_city in ('sao paulo' ,'campinas') or d.customer\_city in ('sao paulo' ,'campinas')) and DATEDIFF( day, c.order\_delivered\_carrier\_date,c.order\_delivered\_customer\_date) is not null order by 3 asc, 4 desc |



|  |
| --- |
| **-- For city sao paulo** |

**In the following query we have found category level sales and orders placed for city sao paulo in the state SP.**

**For the analysis we have shown category name, state, customer city, total price of the orders, the total payment value given by customers, total number of orders, and the extra payment paid by the customers.**

|  |
| --- |
| select c.column2 as category\_name, e.customer\_state,e.customer\_city, sum(a.price) as total\_price ,sum(f.payment\_value) as price\_paid, count(distinct(d.order\_id)) as total\_orders, (sum(f.payment\_value) - sum(a.price)) as extra\_amount\_paid from olist\_order\_items\_dataset as a left join olist\_products\_dataset as b on a.product\_id = b.product\_id join product\_category\_name\_translation as c on b.product\_category\_name = c.column1 left join olist\_orders\_dataset as d on a.order\_id = d.order\_id left join olist\_customers\_dataset as e on d.customer\_id = e.customer\_id left join olist\_order\_payments\_dataset as f on a.order\_id = f.order\_id where e.customer\_state = 'SP' and e.customer\_city = 'sao paulo'  group by c.column2, e.customer\_state, e.customer\_city order by 2 asc, 3 desc, 4 desc, 5 desc |

**In the following query we have found product level sales and orders placed for city sao paulo in the state SP.**

**For the analysis we have found the category name which has the most number of orders, in that city and state.**

|  |
| --- |
| select top 1 c.column2 as category\_name, e.customer\_state,e.customer\_city, count(distinct(d.order\_id)) as total\_orders from olist\_order\_items\_dataset as a left join olist\_products\_dataset as b on a.product\_id = b.product\_id join product\_category\_name\_translation as c on b.product\_category\_name = c.column1 left join olist\_orders\_dataset as d on a.order\_id = d.order\_id left join olist\_customers\_dataset as e on d.customer\_id = e.customer\_id left join olist\_order\_payments\_dataset as f on a.order\_id = f.order\_id where e.customer\_state = 'SP' and e.customer\_city = 'sao paulo' group by c.column2, e.customer\_state, e.customer\_city order by 2 asc , 4 desc    **For the analysis we have all the product ids for that city in the state. Also the total price of the order and the price paid by the customer. We have counted the total number of orders and the extra amount customers have to pay on that product.**  select a.product\_id, sum(c.price) as total\_price, sum(d.payment\_value) as price\_paid, count(d.order\_id) as total\_orders,(sum(d.payment\_value) - sum(c.price)) as extra\_amount\_paid  from olist\_products\_dataset as a join product\_category\_name\_translation as b on a.product\_category\_name = b.column1 left join olist\_order\_items\_dataset as c on a.product\_id = c.product\_id left join olist\_order\_payments\_dataset as d on c.order\_id = d.order\_id left join olist\_orders\_dataset as e  on c.order\_id =e.order\_id left join olist\_customers\_dataset as f on e.customer\_id = f.customer\_id where b.column2 = 'bed\_bath\_table' and f.customer\_state = 'SP' and f.customer\_city = 'sao paulo' group by a.product\_id order by 4 desc |

|  |
| --- |
| **-- For city campinas** |

**In the following query we have found category level sales and orders placed for city campinas in the state SP.**

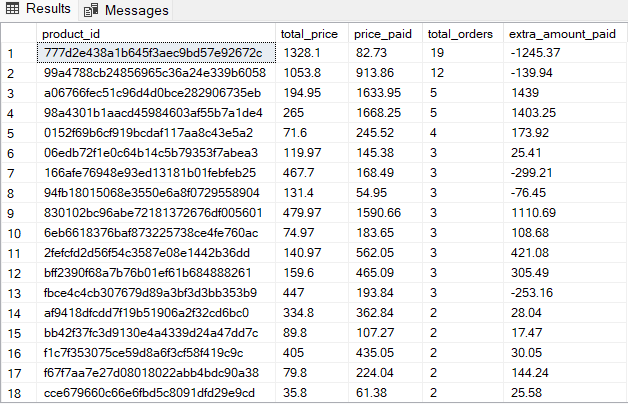
**For the analysis we have shown category name, state, customer city, total price of the orders, the total payment value given by customers, total number of orders, and the extra payment paid by the customers.**

|  |
| --- |
| select c.column2 as category\_name, e.customer\_state,e.customer\_city, sum(a.price) as total\_price ,sum(f.payment\_value) as price\_paid, count(distinct(d.order\_id)) as total\_orders, (sum(f.payment\_value) - sum(a.price)) as extra\_amount\_paid from olist\_order\_items\_dataset as a left join olist\_products\_dataset as b on a.product\_id = b.product\_id join product\_category\_name\_translation as c on b.product\_category\_name = c.column1 left join olist\_orders\_dataset as d on a.order\_id = d.order\_id left join olist\_customers\_dataset as e on d.customer\_id = e.customer\_id left join olist\_order\_payments\_dataset as f on a.order\_id = f.order\_id where e.customer\_state = 'SP' and e.customer\_city = 'campinas'  group by c.column2, e.customer\_state, e.customer\_city order by 2 asc, 3 desc, 4 desc, 5 desc |

**In the following query we have found product level sales and orders placed for city campinas in the state SP.**

**For the analysis we have found the category name which has the most number of orders, in that city and state.**

|  |
| --- |
| select top 1 c.column2 as category\_name, e.customer\_state,e.customer\_city, count(distinct(d.order\_id)) as total\_orders from olist\_order\_items\_dataset as a left join olist\_products\_dataset as b on a.product\_id = b.product\_id join product\_category\_name\_translation as c on b.product\_category\_name = c.column1 left join olist\_orders\_dataset as d on a.order\_id = d.order\_id left join olist\_customers\_dataset as e on d.customer\_id = e.customer\_id left join olist\_order\_payments\_dataset as f on a.order\_id = f.order\_id where e.customer\_state = 'SP' and e.customer\_city = 'campinas' group by c.column2, e.customer\_state, e.customer\_city order by 2 asc , 4 desc    **For the analysis we have all the product ids for that city in the state. Also the total price of the order and the price paid by the customer. We have counted the total number of orders and the extra amount customers have to pay on that product.** select a.product\_id, sum(c.price) as total\_price, sum(d.payment\_value) as price\_paid, count(d.order\_id) as total\_orders,(sum(d.payment\_value) - sum(c.price)) as extra\_amount\_paid  from olist\_products\_dataset as a join product\_category\_name\_translation as b on a.product\_category\_name = b.column1 left join olist\_order\_items\_dataset as c on a.product\_id = c.product\_id left join olist\_order\_payments\_dataset as d on c.order\_id = d.order\_id left join olist\_orders\_dataset as e  on c.order\_id =e.order\_id left join olist\_customers\_dataset as f on e.customer\_id = f.customer\_id where b.column2 = 'bed\_bath\_table' and f.customer\_state = 'SP' and f.customer\_city = 'campinas' group by a.product\_id order by 4 desc |



|  |
| --- |
| **-- In state RJ** |

**In the following query we have found the seller performance in terms of deliveries, for increasing trend state RJ, and the top 2 cities.**

**For the analysis we got the seller ids and seller city and customer city in the top 2 cities we got from before analysis, also we have shown the days to deliver and expected days to deliver for the sellers in those cities, and given them the performance according to that only.**

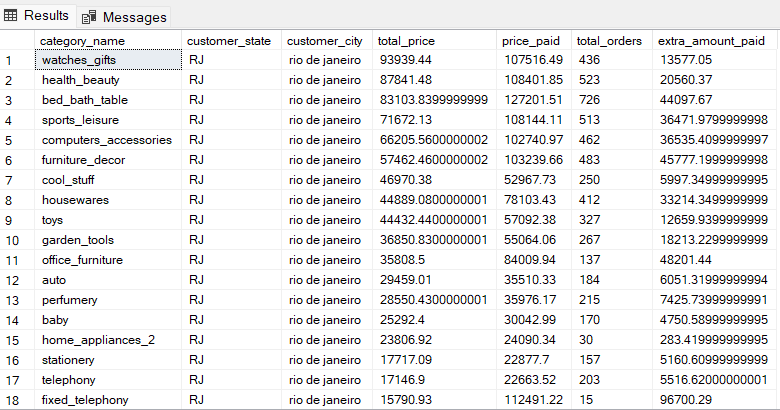
|  |
| --- |
| select distinct(a.seller\_id), a.seller\_city, d.customer\_city, DATEDIFF( day, c.order\_delivered\_carrier\_date,c.order\_delivered\_customer\_date) as days\_to\_deliver,  datediff(day,c.order\_purchase\_timestamp,c.order\_estimated\_delivery\_date) as expected\_days, case when ( DATEDIFF( day, c.order\_delivered\_carrier\_date,c.order\_delivered\_customer\_date) - datediff(day,c.order\_purchase\_timestamp,c.order\_estimated\_delivery\_date)) <0 then 'good' when ( DATEDIFF( day, c.order\_delivered\_carrier\_date,c.order\_delivered\_customer\_date) - datediff(day,c.order\_purchase\_timestamp,c.order\_estimated\_delivery\_date)) >=0 then 'poor'  end as performance from olist\_sellers\_dataset as a left join olist\_order\_items\_dataset as b on a.seller\_id = b.seller\_id left join olist\_orders\_dataset as c on b.order\_id = c.order\_id left join olist\_customers\_dataset as d on c.customer\_id = d.customer\_id where (a.seller\_state = 'RJ' or d.customer\_state = 'RJ') and (a.seller\_city in ('rio de janeiro' ,'niteroi') or d.customer\_city in ('rio de janeiro' ,'niteroi')) and DATEDIFF( day, c.order\_delivered\_carrier\_date,c.order\_delivered\_customer\_date) is not null order by 3 asc, 4 desc |

|  |
| --- |
| **-- For state rio de janeiro** |

**In the following query we have found category level sales and orders placed for city rio de janeiro in the state RJ.**

**For the analysis we have shown category name, state, customer city, total price of the orders, the total payment value given by customers, total number of orders, and the extra payment paid by the customers.**

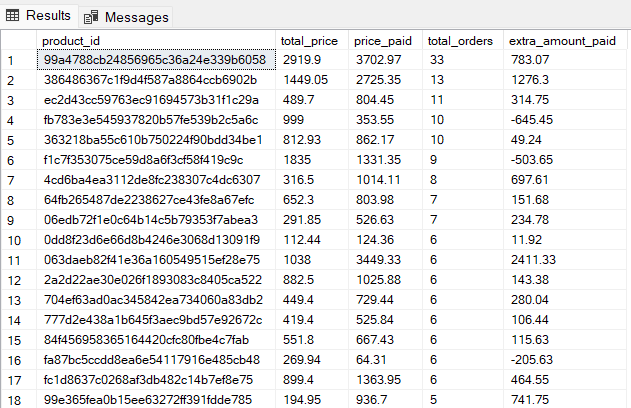
|  |
| --- |
| select top 1 c.column2 as category\_name, e.customer\_state,e.customer\_city, count(distinct(d.order\_id)) as total\_orders from olist\_order\_items\_dataset as a left join olist\_products\_dataset as b on a.product\_id = b.product\_id join product\_category\_name\_translation as c on b.product\_category\_name = c.column1 left join olist\_orders\_dataset as d on a.order\_id = d.order\_id left join olist\_customers\_dataset as e on d.customer\_id = e.customer\_id left join olist\_order\_payments\_dataset as f on a.order\_id = f.order\_id where e.customer\_state = 'RJ' and e.customer\_city = 'rio de janeiro' group by c.column2, e.customer\_state, e.customer\_city order by 2 asc , 4 desc |



**In the following query we have found product level sales and orders placed for city rio de janeiro in the state RJ.**

**For the analysis we have found the category name which has the most number of orders, in that city and state.**

|  |
| --- |
| select top 1 c.column2 as category\_name, e.customer\_state,e.customer\_city, count(distinct(d.order\_id)) as total\_orders from olist\_order\_items\_dataset as a left join olist\_products\_dataset as b on a.product\_id = b.product\_id join product\_category\_name\_translation as c on b.product\_category\_name = c.column1 left join olist\_orders\_dataset as d on a.order\_id = d.order\_id left join olist\_customers\_dataset as e on d.customer\_id = e.customer\_id left join olist\_order\_payments\_dataset as f on a.order\_id = f.order\_id where e.customer\_state = 'RJ' and e.customer\_city = 'rio de janeiro' group by c.column2, e.customer\_state, e.customer\_city order by 2 asc , 4 desc    **For the analysis we have all the product ids for that city in the state. Also the total price of the order and the price paid by the customer. We have counted the total number of orders and the extra amount customers have to pay on that product.**  select a.product\_id, sum(c.price) as total\_price, sum(d.payment\_value) as price\_paid, count(d.order\_id) as total\_orders,(sum(d.payment\_value) - sum(c.price)) as extra\_amount\_paid  from olist\_products\_dataset as a join product\_category\_name\_translation as b on a.product\_category\_name = b.column1 left join olist\_order\_items\_dataset as c on a.product\_id = c.product\_id left join olist\_order\_payments\_dataset as d on c.order\_id = d.order\_id left join olist\_orders\_dataset as e  on c.order\_id =e.order\_id left join olist\_customers\_dataset as f on e.customer\_id = f.customer\_id where b.column2 = 'bed\_bath\_table' and f.customer\_state = 'RJ' and f.customer\_city = 'rio de janeiro' group by a.product\_id order by 4 desc |

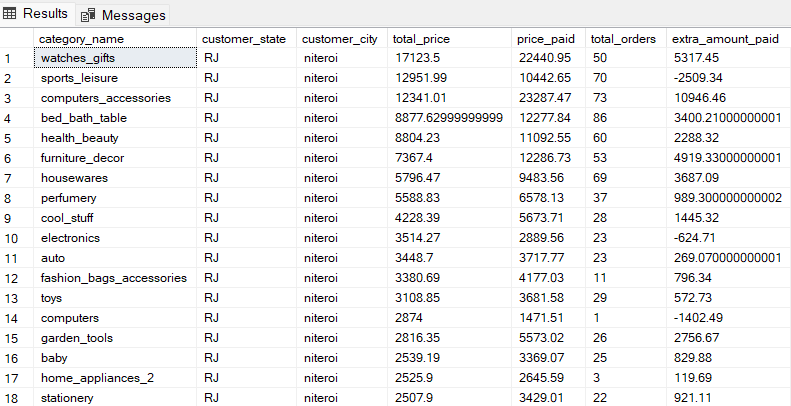


|  |
| --- |
| **-- For state niteroi** |

**In the following query we have found category level sales and orders placed for city niteroi in the state RJ.**

**For the analysis we have shown category name, state, customer city, total price of the orders, the total payment value given by customers, total number of orders, and the extra payment paid by the customers.**

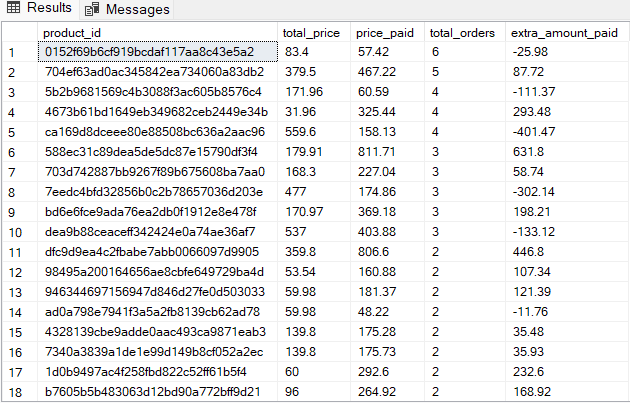
|  |
| --- |
| select c.column2 as category\_name, e.customer\_state,e.customer\_city, sum(a.price) as total\_price ,sum(f.payment\_value) as price\_paid, count(distinct(d.order\_id)) as total\_orders, (sum(f.payment\_value) - sum(a.price)) as extra\_amount\_paid from olist\_order\_items\_dataset as a left join olist\_products\_dataset as b on a.product\_id = b.product\_id join product\_category\_name\_translation as c on b.product\_category\_name = c.column1 left join olist\_orders\_dataset as d on a.order\_id = d.order\_id left join olist\_customers\_dataset as e on d.customer\_id = e.customer\_id left join olist\_order\_payments\_dataset as f on a.order\_id = f.order\_id where e.customer\_state = 'RJ' and e.customer\_city = 'niteroi'  group by c.column2, e.customer\_state, e.customer\_city order by 2 asc, 3 desc, 4 desc, 5 desc |



**In the following query we have found product level sales and orders placed for city niteroi in the state RJ.**

**For the analysis we have found the category name which has the most number of orders, in that city and state.**

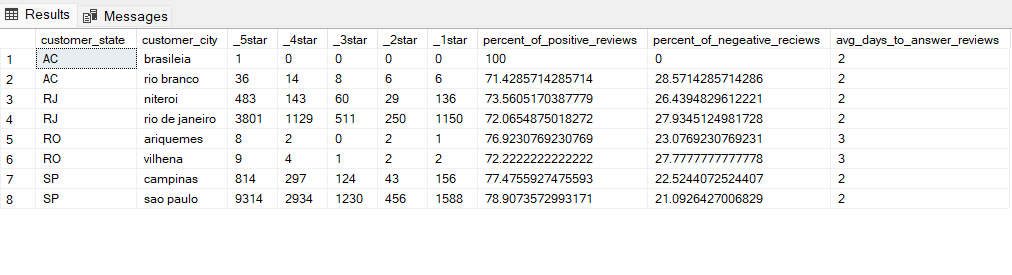
|  |
| --- |
| select top 1 c.column2 as category\_name, e.customer\_state,e.customer\_city, count(distinct(d.order\_id)) as total\_orders from olist\_order\_items\_dataset as a left join olist\_products\_dataset as b on a.product\_id = b.product\_id join product\_category\_name\_translation as c on b.product\_category\_name = c.column1 left join olist\_orders\_dataset as d on a.order\_id = d.order\_id left join olist\_customers\_dataset as e on d.customer\_id = e.customer\_id left join olist\_order\_payments\_dataset as f on a.order\_id = f.order\_id where e.customer\_state = 'RJ' and e.customer\_city = 'niteroi'  group by c.column2, e.customer\_state, e.customer\_city order by 2 asc , 4 desc    **For the analysis we have all the product ids for that city in the state. Also the total price of the order and the price paid by the customer. We have counted the total number of orders and the extra amount customers have to pay on that product.**  select a.product\_id, sum(c.price) as total\_price, sum(d.payment\_value) as price\_paid, count(d.order\_id) as total\_orders,(sum(d.payment\_value) - sum(c.price)) as extra\_amount\_paid  from olist\_products\_dataset as a join product\_category\_name\_translation as b on a.product\_category\_name = b.column1 left join olist\_order\_items\_dataset as c on a.product\_id = c.product\_id left join olist\_order\_payments\_dataset as d on c.order\_id = d.order\_id left join olist\_orders\_dataset as e  on c.order\_id =e.order\_id left join olist\_customers\_dataset as f on e.customer\_id = f.customer\_id where b.column2 = 'bed\_bath\_table' and f.customer\_state = 'RJ' and f.customer\_city = 'niteroi'  group by a.product\_id order by 4 desc |



**In the following query we have calculated post order reviews city wise for the trending states.**

**For the analysis we created a new table post\_order\_decline\_city, which shows the customer state and customer city, with a total number of reviews for each star in the review score.**

|  |
| --- |
| create table post\_order\_decline\_city( customer\_state varchar(50), customer\_city varchar(50), \_5star float, \_4star float, \_3star float, \_2star float, \_1star float) insert into post\_order\_decline\_city select c.customer\_state, c.customer\_city, sum(case when b.review\_score = 5 then 1 else 0 end) as \_5star , sum(case when b.review\_score = 4 then 1 else 0 end) as \_4star , sum(case when b.review\_score = 3 then 1 else 0 end) as \_3star , sum(case when b.review\_score = 2 then 1 else 0 end) as \_2star , sum(case when b.review\_score = 1 then 1 else 0 end) as \_1star  from olist\_orders\_dataset as a left join olist\_order\_reviews\_dataset as b on a.order\_id = b.order\_id left join olist\_customers\_dataset as c on a.customer\_id = c.customer\_id where c.customer\_state in ('SP','RJ','AC','RO') and c.customer\_city in ('niteroi’, ‘rio de janeiro','campinas','sao paulo','vilhena','ariquemes','rio branco’, ‘brasileia') group by c.customer\_state, c.customer\_city  Select \* from post\_order\_decline\_city    **In the following query we have shown our post order reviews,for all the top cities which are causing the trend. With percent of positive reviews and negative reviews. Also the average number of days it takes the seller to answer those reviews.**  select a.\*, ((a.\_5star+a.\_4star) / (a.\_5star+a.\_4star+a.\_3star+a.\_2star+a.\_1star))\*100 as percent\_of\_positive\_reviews,  (100 -((a.\_5star+a.\_4star) / (a.\_5star+a.\_4star+a.\_3star+a.\_2star+a.\_1star))\*100 ) as percent\_of\_negeative\_reciews, avg(datediff(day,d.review\_creation\_date,d.review\_answer\_timestamp)) as avg\_days\_to\_answer\_reviews from post\_order\_decline\_city as a left join olist\_customers\_dataset as b on a.customer\_state = b.customer\_state left join olist\_orders\_dataset as c on b.customer\_id = c.customer\_id left join olist\_order\_reviews\_dataset as d on c.order\_id = d.order\_id group by a.customer\_state,a.customer\_city,a.\_5star,a.\_4star,a.\_3star,a.\_2star,a.\_1star order by a.customer\_state , 2,3 |



**In the following query we are finding the percent of order delivered before the estimated date.**

**For the analysis we create a new table perf\_decline\_earlier\_city, which shows all the seller ids, seller and customer city, the days to deliver and expected days and performance, where performance of the seller is shown as 1 if it is good and 0 if it is poor.**

|  |
| --- |
| create table perf\_decline\_earlier\_city ( seller\_id varchar(50), seller\_city varchar(50), customer\_city varchar(40), days\_to\_deliver int, expected\_days int, performance int) insert into perf\_decline\_earlier\_city select distinct(a.seller\_id), a.seller\_city, d.customer\_city, DATEDIFF( day, c.order\_delivered\_carrier\_date,c.order\_delivered\_customer\_date) as days\_to\_deliver, datediff(day,c.order\_purchase\_timestamp,c.order\_estimated\_delivery\_date) as expected\_days,case when ( DATEDIFF( day, c.order\_delivered\_carrier\_date,c.order\_delivered\_customer\_date) - datediff(day,c.order\_purchase\_timestamp,c.order\_estimated\_delivery\_date)) <0 then 1 when ( DATEDIFF( day, c.order\_delivered\_carrier\_date,c.order\_delivered\_customer\_date) - datediff(day,c.order\_purchase\_timestamp,c.order\_estimated\_delivery\_date)) >=0 then 0  end as performance from olist\_sellers\_dataset as a left join olist\_order\_items\_dataset as b on a.seller\_id = b.seller\_id left join olist\_orders\_dataset as c on b.order\_id = c.order\_id left join olist\_customers\_dataset as d on c.customer\_id = d.customer\_id where a.seller\_state in('AC','RO','SP','RJ') or d.customer\_state in ('AC','RO','SP','RJ') and DATEDIFF( day, c.order\_delivered\_carrier\_date,c.order\_delivered\_customer\_date) is not null and a.seller\_city in ('niteroi','rio de janeiro','campinas','sao paulo','vilhena','ariquemes','rio branco','brasileia') or d.customer\_city in ('niteroi','rio de janeiro','campinas','sao paulo','vilhena','ariquemes','rio branco','brasileia') order by 3 asc, 4 desc select \* from perf\_decline\_earlier\_city    alter table perf\_decline\_earlier\_city alter column performance float |

|  |
| --- |
| **-- For state AC** |

|  |
| --- |
| select (sum(case when performance = 1 then 1 else 0 end)\*100)/ (select (cast (count(performance) as float)) from perf\_decline\_earlier\_city) as percent\_of\_order\_before\_date from perf\_decline\_earlier\_city where customer\_city in ('rio branco','brasileia') or seller\_city in ('rio branco','brasileia') |

|  |
| --- |
| **-- For state RO** |

|  |
| --- |
| select (sum(case when performance = 1 then 1 else 0 end)\*100)/ (select (cast (count(performance) as float)) from perf\_decline\_earlier\_city) as percent\_of\_order\_before\_date from perf\_decline\_earlier\_city where customer\_city in ('vilhena','ariquemes') or seller\_city in ('vilhena','ariquemes') |

|  |
| --- |
| **-- For state SP** |

|  |
| --- |
| select (sum(case when performance = 1 then 1 else 0 end)\*100)/ (select (cast (count(performance) as float)) from perf\_decline\_earlier\_city) as percent\_of\_order\_before\_date from perf\_decline\_earlier\_city where customer\_city in ('campinas','sao paulo') or seller\_city in ('campinas','sao paulo') |

|  |
| --- |
| **-- For state RJ** |

|  |
| --- |
| select (sum(case when performance = 1 then 1 else 0 end)\*100)/ (select (cast (count(performance) as float)) from perf\_decline\_earlier\_city) as percent\_of\_order\_before\_date from perf\_decline\_earlier\_city where customer\_city in ('niteroi','rio de janeiro') or seller\_city in ('niteroi','rio de janeiro')    **--------------------------------------------------------------------------** |

E] After doing the Root cause analysis, help the client by suggesting ways to improve the performance of the States and the cities.

Answer -> Ways to improve the performance of the states and the cities :

* In the first problem statement we can see that the trendline is somewhat similar for the different metrics i.e. our sales, customer acquisition and total no. of orders have increased in the same states and remained low on the same states.
* We should increase the number of sellers in AC and RO states as they have only 1 and 2 sellers respectively. Because of that most of the orders have to be send by the sellers who live in SP and RJ.
* We should also increase the variety of products that we sell in these states, as there are very selective category of products we sell here. Also in that category, a customer has not very much choice but to go with 5-6 products.
* We can see that the percentage of order delivered before expected date in declining trend states is much higher, than percentage of order delivered before expected date in increasing trend states. The main reason being that a declining trend state seller has to send products only to its state, whereas the increasing trend state sellers have to deliver their products into many different states which creates problems and decrease their overall performance.
* On the other hand, when we see the top cities of these states, we can see that cities which come in declining trends have very low percentage of order delivered before expected date, which is why we should change the seller at those cities.
* In the declining trend cities, we have very low type of categories available, and the type of products is also very low, we must increase them so that customers can get attracted towards our store.
* We must come up with a strategy to reduce the time taken for a product delivery, as the number of days is quiet high for declining states.