Business Case: Target SQL

Problem Statement:

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

```
SELECT
  column_name,
  data_type
FROM target.INFORMATION_SCHEMA.COLUMNS
WHERE table_name = 'customers'
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION D

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

Note: *column[customer_zip_code_prefix] is in "Integer" datatype, reset of all are in "String" datatype.*

    2. Get the time range between which the orders were placed.

```
SELECT
  DATE(MIN(order_purchase_timestamp)) AS  First_OrderDate,
  DATE(MAX(order_purchase_timestamp)) AS  Last_OrderDate,
  TIMESTAMP_DIFF(DATE(MAX(order_purchase_timestamp)),DATE(MIN(order_purchase_time
stamp)), day) as Total_Days,
  TIMESTAMP_DIFF(DATE(MAX(order_purchase_timestamp)),DATE(MIN(order_purchase_time
stamp)), month) as Total_Month
FROM `target.orders`
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXEC

Row	First_OrderDate	Last_OrderDate	Total_Days	Total_Month
1	2016-09-04	2018-10-17	773	25

Note: *First order placed on 04-Sep-2016, and the Last order was placed on 17-Oct-2018 as per the given data set, the time gap is 773 days or 25 months.*

    3. Count the number of Cities and States in our dataset.

Method # 1. To find count of all individual tables in single output

```
SELECT
  'Customers' as Table_Name,
  COUNT(DISTINCT customer_city) AS City_Count,
  COUNT(DISTINCT customer_state) AS State_Count
FROM `target.customers`
UNION ALL
SELECT
  'Geoloctinon' as Table_Name,
  COUNT(DISTINCT geolocation_city) AS City_Count,
  COUNT(DISTINCT geolocation_state) AS State_Count
FROM `target.geolocation`
UNION ALL
SELECT
  'Sellers' as Table_Name,
  COUNT(DISTINCT seller_city) AS City_Count,
  COUNT(DISTINCT seller_state) AS State_Count
FROM `target.sellers`
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	Table_Name	City_Count	State_Count
1	Sellers	611	23
2	Geoloctinon	8011	27
3	Customers	4119	27

Method # 2. To find Unique count of City and State based on all tables.

```
WITH overall_count AS
(
SELECT
  customer_city AS City,
  customer_state AS State
FROM `target.customers`
UNION DISTINCT
SELECT
  geolocation_city AS City,
  geolocation_state AS State
FROM `target.geolocation`
UNION DISTINCT
SELECT
  seller_city AS City,
  seller_state AS State
FROM `target.sellers`
)
SELECT
  COUNT(DISTINCT City) AS OverallCity_Count,
  COUNT(DISTINCT State) AS OverallState_Count
FROM overall_count
```

Query results

JOB INFORMATION | RESULTS | JSON

Row	OverallCity_Count	OverallState_Count
1	8126	27

Note: *No. of Unique Cities and States in our dataset are 8126 and 27 respectively.*

2. **In-depth Exploration:**

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

```
WITH mytable as
(
SELECT
  EXTRACT(YEAR FROM order_purchase_timestamp) AS Order_Year,
  COUNT(order_purchase_timestamp) AS NoOfOrders
FROM `target.orders`
GROUP BY Order_Year
ORDER BY Order_Year
)
SELECT
  Order_Year,
  NoOfOrders,
  CONCAT(ROUND((NoOfOrders-LAG(NoOfOrders) OVER(ORDER BY Order_Year
ASC))/NoOfOrders *100,2),'%') as YoY_Inc_Percentage
FROM  mytable
ORDER BY Order_Year
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTIO

Row	Order_Year	NoOfOrders	YoY_Inc_Percentage
1	2016	329	*null*
2	2017	45101	99.27%
3	2018	54011	16.5%

Note: *Significant growth of 99.27% over order placed occurred between 2016 and 2017 and it is dropped to 16.5% in the next subsequent year.*

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

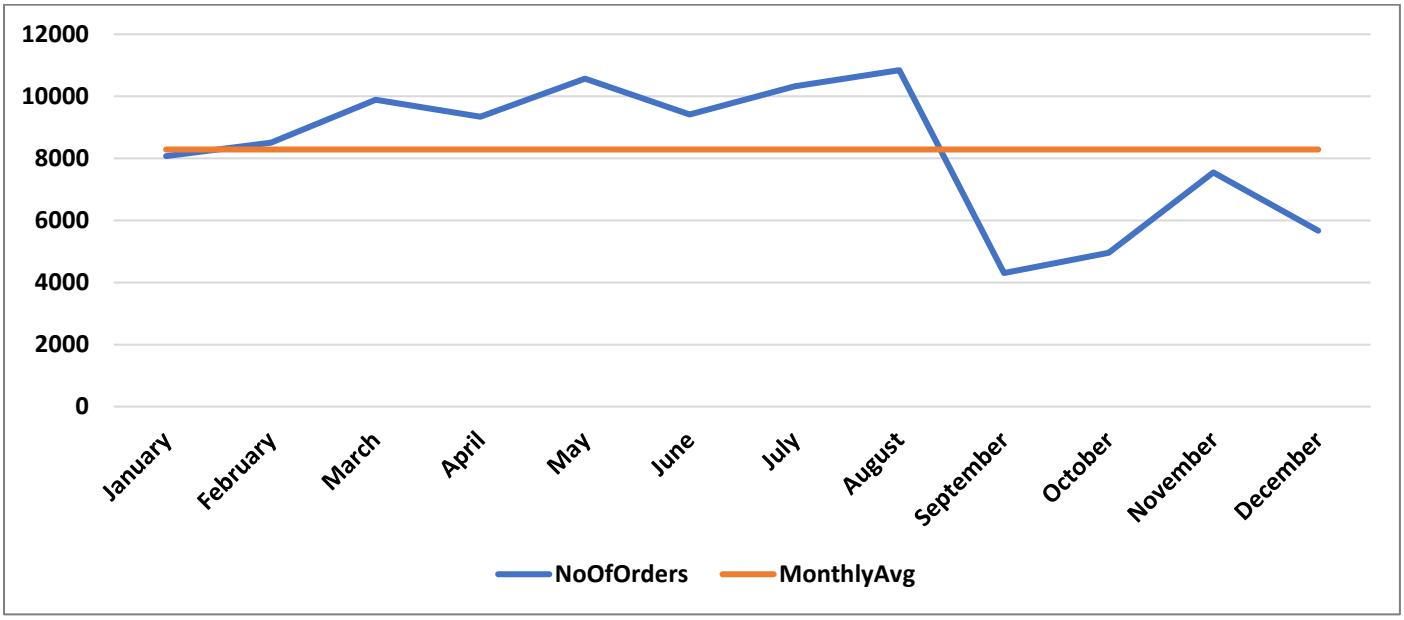
```
WITH mytable as
(
  SELECT
    EXTRACT(MONTH FROM order_purchase_timestamp) AS MonthNum,
    FORMAT_DATE('%B',order_purchase_timestamp) as MonthName,
    COUNT(order_purchase_timestamp) AS NoOfOrders,
    CAST(ROUND(AVG(COUNT(order_purchase_timestamp)) OVER()) as INT) AS
MonthlyAvg,
```

```
  FROM `target.orders`
  GROUP BY MonthNum,MonthName
  ORDER BY MonthNum
)
SELECT
  MonthName,
  NoOfOrders,
  MonthlyAvg,
  NoOfOrders - MonthlyAvg as DiffwithMonthlyAvg
FROM mytable
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

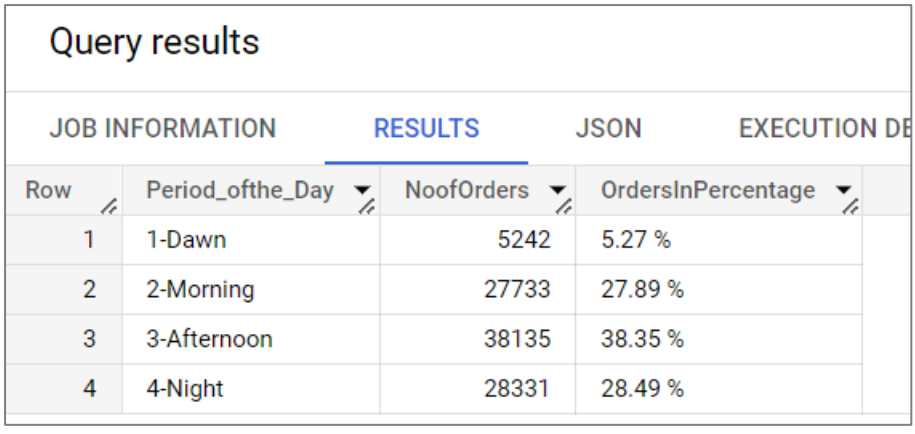
Row	MonthName	NoOfOrders	MonthlyAvg	DiffwithMonthlyAvg
1	January	8069	8287	-218
2	February	8508	8287	221
3	March	9893	8287	1606
4	April	9343	8287	1056
5	May	10573	8287	2286
6	June	9412	8287	1125
7	July	10318	8287	2031
8	August	10843	8287	2556
9	September	4305	8287	-3982
10	October	4959	8287	-3328
11	November	7544	8287	-743
12	December	5674	8287	-2613



Note: *No. of Orders were relatively higher during Jan to Aug with respective to monthly Average and it has dropped significantly during the last 4 Months.*

3. During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)
    - 0-6 hrs : Dawn
    - 7-12 hrs : Mornings
    - 13-18 hrs : Afternoon
    - 19-23 hrs : Night

```
WITH mytable2 as
(
SELECT
  CASE
    WHEN EXTRACT(HOUR FROM ord.order_purchase_timestamp) BETWEEN 0 AND 6 THEN '1-
Dawn'
    WHEN EXTRACT(HOUR FROM ord.order_purchase_timestamp) BETWEEN 7 AND 12 THEN
'2-Morning'
    WHEN EXTRACT(HOUR FROM ord.order_purchase_timestamp) BETWEEN 13 AND 18 THEN
'3-Afternoon'
    WHEN EXTRACT(HOUR FROM ord.order_purchase_timestamp) BETWEEN 19 AND 23 THEN
'4-Night'
  END AS Period_ofthe_Day,
  CAST(COUNT(ord.order_id) AS INT64) AS NoofOrders,
FROM `target.orders`ord
JOIN  `target.customers` cust
ON ord.customer_id = cust.customer_id
GROUP BY  Period_ofthe_Day
ORDER BY  Period_ofthe_Day
)
SELECT
  Period_ofthe_Day,
  NoofOrders,
  CONCAT(ROUND(NoofOrders/(SUM(NoofOrders) over())*100,2),' %') AS
OrdersInPercentage
FROM mytable2
GROUP BY  Period_ofthe_Day, NoofOrders
ORDER BY  Period_ofthe_Day
```


Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DE

Row	Period_ofthe_Day	NoofOrders	OrdersInPercentage
1	1-Dawn	5242	5.27 %
2	2-Morning	27733	27.89 %
3	3-Afternoon	38135	38.35 %
4	4-Night	28331	28.49 %

Note: *Our Analysis shows that Brazilian customers placing there order mostly during afternoon and night.*

3. **Evolution of E-commerce orders in the Brazil region:**
    1. Get the month-on-month no. of orders placed in each state.

Method # 1. Query to find State wise and Month wise sales details, sorted state wise alphabetically.

```
SELECT
  cust.customer_state,
  EXTRACT(month FROM ord.order_purchase_timestamp) AS MonthNum,
  FORMAT_DATE('%B',ord.order_purchase_timestamp) AS MonthName,
  CAST(COUNT(ord.order_id) AS INT64) AS NoofOrders
FROM `target.orders`ord
JOIN  `target.customers` cust
ON ord.customer_id = cust.customer_id
GROUP BY customer_state,MonthNum, MonthName
ORDER BY customer_state,MonthNum, MonthName
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECU

Row	customer_state	MonthNum	MonthName	NoofOrders
1	AC	1	January	8
2	AC	2	February	6
3	AC	3	March	4
4	AC	4	April	9
5	AC	5	May	10
6	AC	6	June	7
7	AC	7	July	9
8	AC	8	August	7
9	AC	9	September	5
10	AC	10	October	6

Method # 2. Query to find state wise sales details.

```
SELECT
  cust.customer_state,
  CAST(COUNT(ord.order_id) AS INT64) AS NoofOrders,
  RANK() OVER(ORDER BY CAST(COUNT(ord.order_id) AS INT64) DESC) AS Ranking
FROM `target.orders`ord
JOIN  `target.customers` cust
ON ord.customer_id = cust.customer_id
GROUP BY customer_state
ORDER BY NoofOrders DESC
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION

Row	customer_state	NoofOrders	Ranking
1	SP	41746	1
2	RJ	12852	2
3	MG	11635	3
4	RS	5466	4
5	PR	5045	5
6	SC	3637	6
7	BA	3380	7
8	DF	2140	8
9	ES	2033	9
10	GO	2020	10

Note: *The above query results indicate that the states which are "SP","RJ" & "MG" are falling under top 3 and contributing higher sales.*

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

```
WITH mytable31 as
(
SELECT
  customer_state,
  CAST(count(distinct customer_id) AS int) AS Custcount
FROM `target.customers`
GROUP BY customer_state
ORDER BY Custcount desc
)
SELECT
  customer_state,
  Custcount,
  CONCAT(ROUND(Custcount/(SUM(Custcount) OVER())*100,2),' %') AS CustPercentage
FROM mytable31
GROUP BY customer_state, Custcount
ORDER by Custcount DESC
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION

Row	customer_state	Custcount	CustPercentage
1	SP	41746	41.98 %
2	RJ	12852	12.92 %
3	MG	11635	11.7 %
4	RS	5466	5.5 %
5	PR	5045	5.07 %
6	SC	3637	3.66 %
7	BA	3380	3.4 %
8	DF	2140	2.15 %
9	ES	2033	2.04 %
10	GO	2020	2.03 %

Note: *The above query results shows that the states which are "SP","RJ" & "MG" are having the more number of customers contributing to 60% when compared to all other states.*

4. **Impact on Economy: Analyse the money movement by e-commerce by looking at order prices, freight and others.**
    1. Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).
       You can use the "payment_value" column in the payments table to get the cost of orders.

```
Method # 1.  Query to month wise cost increase %.

WITH table41 as
(SELECT
  EXTRACT(YEAR FROM ord.order_purchase_timestamp) as Year,
  EXTRACT(MONTH FROM ord.order_purchase_timestamp) as Month,
  ROUND(SUM(pmt.payment_value),0) AS Pymt_2017
FROM `target.orders` ord
JOIN `target.payments` pmt
ON ord.order_id = pmt.order_id
GROUP BY Year,Month
HAVING Year=2017 and Month between 1 and 8
ORDER BY Year,Month),
table42 as
(SELECT
  EXTRACT(YEAR FROM ord.order_purchase_timestamp) as Year,
  EXTRACT(MONTH FROM ord.order_purchase_timestamp) as Month,
  ROUND(SUM(pmt.payment_value),0) AS Pymt_2018
FROM `target.orders` ord
JOIN `target.payments` pmt
ON ord.order_id = pmt.order_id
GROUP BY Year,Month
HAVING Year=2018 and Month between 1 and 8
ORDER BY Year,Month)
SELECT
  tb1.Month,
```

```
  Pymt_2017,
  Pymt_2018,
  tb2.Pymt_2018 - tb1.Pymt_2017  as ValueDiff,
  CONCAT(ROUND((tb2.Pymt_2018 - tb1.Pymt_2017)/tb2.Pymt_2018*100,2),' %') AS
Inc_Percentage
FROM table41 as tb1
JOIN table42 as tb2
ON tb1.Month = tb2.Month
ORDER BY tb1.Month
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXEC

Row	Month	Pymt_2017	Pymt_2018	ValueDiff	Inc_Percentage
1	1	138488.0	1115004.0	976516.0	87.58 %
2	2	291908.0	992463.0	700555.0	70.59 %
3	3	449864.0	1159652.0	709788.0	61.21 %
4	4	417788.0	1160785.0	742997.0	64.01 %
5	5	592919.0	1153982.0	561063.0	48.62 %
6	6	511276.0	1023880.0	512604.0	50.06 %
7	7	592383.0	1066541.0	474158.0	44.46 %
8	8	674396.0	1022425.0	348029.0	34.04 %

Method # 2. Query to find YoY increase %.

```
  WITH table52 as
  (
  SELECT
    ROW_NUMBER() OVER() AS rno,
    2017 as Year,
    ROUND(SUM(pmt.payment_value),0) AS PymtVal_2017
  FROM `target.orders` ord
  JOIN `target.payments` pmt
  ON ord.order_id = pmt.order_id
  WHERE EXTRACT(YEAR FROM ord.order_purchase_timestamp)=2017
    and EXTRACT(MONTH FROM ord.order_purchase_timestamp) between 1 and 8
  ),
  table53 as
  (SELECT
    ROW_NUMBER() OVER() AS rno,
    2018 as Year,
    ROUND(SUM(pmt.payment_value),0) AS PymtVal_2018
  FROM `target.orders` ord
  JOIN `target.payments` pmt
  ON ord.order_id = pmt.order_id
  WHERE EXTRACT(YEAR FROM ord.order_purchase_timestamp)=2018
    and EXTRACT(MONTH FROM ord.order_purchase_timestamp) between 1 and 8
  )
  SELECT
    t5.Year,
    t5.PymtVal_2017,
    t6.Year,
```

```
  t6.PymtVal_2018,
  t6.PymtVal_2018 - t5.PymtVal_2017 as Diffval,
  CONCAT(ROUND((PymtVal_2018 - PymtVal_2017)/PymtVal_2017*100,2),' %') AS
Percentage_Increase
FROM table52 as t5
JOIN table53 as t6
ON t5.rno = t6.rno
ORDER BY t5.Year
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAP

Row	Year	PymtVal_2017	Year_1	PymtVal_2018	Diffval	Percentage_Increase
1	2017	3669022.0	2018	8694734.0	50257...	136.98 %

Note: *Based on the above output we found that there is a 136.98% increase in the cost of orders from year 2017 to 2018 (which include months between Jan to Aug only).*

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

```
SELECT
  cust.customer_state as StateCode,
  ROUND(AVG(oitm.price),2) AS Avg_OrderPrice,
  ROUND(SUM(oitm.price),2) AS Total_OrderPrice
FROM `target.orders` ord
JOIN `target.order_items` oitm ON ord.order_id = oitm.order_id
JOIN `target.customers` cust ON ord.customer_id = cust.customer_id
GROUP BY cust.customer_state
ORDER BY Total_OrderPrice DESC
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECU

Row	StateCode	Avg_OrderPrice	Total_OrderPrice
1	SP	109.65	5202955.05
2	RJ	125.12	1824092.67
3	MG	120.75	1585308.03
4	RS	120.34	750304.02
5	PR	119.0	683083.76
6	SC	124.65	520553.34
7	BA	134.6	511349.99
8	DF	125.77	302603.94
9	GO	126.27	294591.95
10	ES	121.91	275037.31

Note*: The above table shows the state wise Total Value of Order price and Average Order price, state "SP" has got a highest Order price and Average Order price is relatively lower than other states.*

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

```
SELECT
  cust.customer_state as StateCode,
  ROUND(AVG(oitm.freight_value),2) AS Avg_FrieghtOrderPrice,
  ROUND(SUM(oitm.freight_value),2) AS Total_FrieghtPrice
FROM `target.orders` ord
JOIN `target.order_items` oitm ON ord.order_id = oitm.order_id
JOIN `target.customers` cust ON ord.customer_id = cust.customer_id
GROUP BY cust.customer_state
ORDER BY Total_FrieghtPrice DESC
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	StateCode	Avg_FrieghtOrderPrice	Total_FrieghtPrice
1	SP	15.15	718723.07
2	RJ	20.96	305589.31
3	MG	20.63	270853.46
4	RS	21.74	135522.74
5	PR	20.53	117851.68
6	BA	26.36	100156.68
7	SC	21.47	89660.26
8	PE	32.92	59449.66
9	GO	22.77	53114.98
10	DF	21.04	50625.5

Note: *State "SP" has again got a highest Freight price and Average Freight price is relatively lower than other states.*

5. **Analysis based on sales, freight and delivery time.**
    1. Find the no. of days taken to deliver each order from the order's purchase date as delivery time.
    Also, calculate the difference (in days) between the estimated & actual delivery date of an order.
    Do this in a single query.

    You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula:
        - **time_to_deliver** = order_delivered_customer_date - order_purchase_timestamp
        - **diff_estimated_delivery** = order_estimated_delivery_date - order_delivered_customer_date

```
SELECT
  order_id,
  DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) AS
time_to_deliver,
```

```
  DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date, DAY) AS
diff_estimated_delivery,
FROM `target.orders`
WHERE order_purchase_timestamp IS NOT NULL and order_delivered_customer_date IS
NOT NULL
ORDER BY time_to_deliver DESC, diff_estimated_delivery DESC
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GR

Row	order_id	time_to_deliver	diff_estimated_delivery
1	ca07593549f1816d26a572e06dc1eab6	209	-181
2	1b3190b2dfa9d789e1f14c05b647a14a	208	-188
3	440d0d17af552815d15a9e41abe49359	195	-165
4	2fb597c2f772eca01b1f5c561bf6cc7b	194	-155
5	0f4519c5f1c541ddec9f21b3bddd533a	194	-161
6	285ab9426d6982034523a855f55a885e	194	-166
7	47b40429ed8cce3aee9199792275433f	191	-175
8	2fe324febf907e3ea3f2aa9650869fa5	189	-167
9	2d7561026d542c8dbd8f0daeadf67a43	188	-159
10	437222e3fd1b07396f1d9ba8c15fba59	187	-144

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

```
(SELECT
  cust.customer_state as StateCode,
  'Top 5 Highest Avg Freight value' as Catetory,
  ROUND(AVG(oitm.freight_value),2) AS Avg_FrieghtOrderPrice
FROM `target.orders` as ord
JOIN `target.order_items` as oitm
ON ord.order_id = oitm.order_id
JOIN `target.customers` as cust
ON ord.customer_id = cust.customer_id
GROUP BY cust.customer_state
ORDER BY Avg_FrieghtOrderPrice DESC
LIMIT 5)
UNION ALL
(SELECT
  cust.customer_state as StateCode,
  'Top 5 Lowest Avg Freight value' as Catetory,
  ROUND(AVG(oitm.freight_value),2) AS Avg_FrieghtOrderPrice
FROM `target.orders` as ord
JOIN `target.order_items` as oitm
ON ord.order_id = oitm.order_id
JOIN `target.customers` as cust
ON ord.customer_id = cust.customer_id
GROUP BY cust.customer_state
ORDER BY Avg_FrieghtOrderPrice ASC
LIMIT 5)
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	StateCode	Catetory	Avg_FrieghtOrderPrice
1	RR	Top 5 Highest Avg Freight value	42.98
2	PB	Top 5 Highest Avg Freight value	42.72
3	RO	Top 5 Highest Avg Freight value	41.07
4	AC	Top 5 Highest Avg Freight value	40.07
5	PI	Top 5 Highest Avg Freight value	39.15
6	SP	Top 5 Lowest Avg Freight value	15.15
7	PR	Top 5 Lowest Avg Freight value	20.53
8	MG	Top 5 Lowest Avg Freight value	20.63
9	RJ	Top 5 Lowest Avg Freight value	20.96
10	DF	Top 5 Lowest Avg Freight value	21.04

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

```
(SELECT
  cust.customer_state,
  'Top 5_Highest Avg Delivery time' as Catetory,
  ROUND(AVG(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp,
DAY)), 2) AS AvgDeliveryTime_Days,
  ROUND(AVG(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp,
HOUR)), 2) AS AvgDeliveryTime_Minutes
FROM `target.orders` as ord
JOIN `target.customers` as cust
ON ord.customer_id = cust.customer_id
WHERE order_purchase_timestamp IS NOT NULL and order_delivered_customer_date IS
NOT NULL
GROUP BY cust.customer_state
ORDER BY AvgDeliveryTime_Days DESC
LIMIT 5)
UNION ALL
(SELECT
  cust.customer_state,
  'Top 5_Lowest Avg Delivery time' as Catetory,
  ROUND(AVG(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp,
DAY)), 2) AS AvgDeliveryTime_Days,
  ROUND(AVG(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp,
HOUR)), 2) AS AvgDeliveryTime_Minutes,
FROM `target.orders` as ord
JOIN `target.customers` as cust
ON ord.customer_id = cust.customer_id
WHERE order_purchase_timestamp IS NOT NULL and order_delivered_customer_date IS
NOT NULL
GROUP BY cust.customer_state
ORDER BY AvgDeliveryTime_Days ASC
LIMIT 5)
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	customer_state	Catetory	AvgDeliveryTime_Days	AvgDeliveryTime_Minutes
1	RR	Top 5_Highest Avg Delivery time	28.98	704.73
2	AP	Top 5_Highest Avg Delivery time	26.73	651.97
3	AM	Top 5_Highest Avg Delivery time	25.99	633.7
4	AL	Top 5_Highest Avg Delivery time	24.04	588.54
5	PA	Top 5_Highest Avg Delivery time	23.32	570.05
6	SP	Top 5_Lowest Avg Delivery time	8.3	209.77
7	PR	Top 5_Lowest Avg Delivery time	11.53	287.3
8	MG	Top 5_Lowest Avg Delivery time	11.54	287.75
9	DF	Top 5_Lowest Avg Delivery time	12.51	310.72
10	SC	Top 5_Lowest Avg Delivery time	14.48	358.52

4. Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.
   You can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.

```
WITH mytbl54 as
(
SELECT
  cust.customer_state,
  ROUND(AVG(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp,
DAY)), 1) AS AvgActualDelivery_Days,
  ROUND(AVG(DATE_DIFF(order_estimated_delivery_date, order_purchase_timestamp,
DAY)), 1) AS AvgEstimatedDelivery_Days,
FROM `target.orders` as ord
JOIN `target.customers` as cust
ON ord.customer_id = cust.customer_id
WHERE order_delivered_customer_date IS NOT NULL and order_estimated_delivery_date
IS NOT NULL
GROUP BY cust.customer_state
ORDER BY AvgActualDelivery_Days ASC
)
SELECT *,
 ROUND(AvgEstimatedDelivery_Days - AvgActualDelivery_Days,1) as
NoofDaysEarlyDelivered
FROM mytbl54
ORDER BY NoofDaysEarlyDelivered DESC
LIMIT 5
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	customer_state	AvgActualDelivery_Days	AvgEstimatedDelivery_Days	NoofDaysEarlyDelivered
1	AC	20.6	40.7	20.1
2	RO	18.9	38.4	19.5
3	AP	26.7	45.9	19.2
4	AM	26.0	44.9	18.9
5	RR	29.0	45.6	16.6

6. **Analysis based on the payments:**
    1. Find the month-on-month no. of orders placed using different payment types.

Output # 1. Query to obtain Payment type & Month wise Order details.

```
SELECT
  pmt.payment_type,
  EXTRACT(MONTH FROM ord.order_purchase_timestamp) AS MnthNum,
  FORMAT_DATE('%B', ord.order_purchase_timestamp) as MonthName,
  COUNT(DISTINCT ord.order_id) AS Total_Orders
FROM `target.orders` as ord
JOIN `target.payments` as pmt
on ord.order_id = pmt.order_id
GROUP BY pmt.payment_type,MnthNum, MonthName
ORDER BY pmt.payment_type,MnthNum
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	payment_type	MnthNum	MonthName	Total_Orders
1	UPI	1	January	1715
2	UPI	2	February	1723
3	UPI	3	March	1942
4	UPI	4	April	1783
5	UPI	5	May	2035
6	UPI	6	June	1807
7	UPI	7	July	2074
8	UPI	8	August	2077
9	UPI	9	September	903
10	UPI	10	October	1056
11	UPI	11	November	1509
12	UPI	12	December	1160
13	credit_card	1	January	6093

Output # 2. Query to obtain Payment type wise Order details.

```
SELECT
  pmt.payment_type,
  COUNT(DISTINCT ord.order_id) AS Total_Orders,
  CONCAT(ROUND(COUNT(DISTINCT ord.order_id)/(SUM(COUNT(DISTINCT ord.order_id))
OVER())*100,2),' %') as Order_Percentage
FROM `target.orders` as ord
JOIN `target.payments` as pmt
ON ord.order_id = pmt.order_id
GROUP BY pmt.payment_type
ORDER BY Total_Orders DESC
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	payment_type	Total_Orders	Order_Percentage
1	credit_card	76505	75.24 %
2	UPI	19784	19.46 %
3	voucher	3866	3.8 %
4	debit_card	1528	1.5 %
5	not_defined	3	0 %

***Note**: The above table shows that Credit card and UPI mode are widely used payment types by the customers.*

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

```
SELECT
  pmt.payment_installments,
  COUNT(DISTINCT ord.order_id) AS Total_Orders,
  CONCAT(ROUND(COUNT(DISTINCT ord.order_id)/(SUM(COUNT(DISTINCT ord.order_id))
OVER())*100,2),' %') AS Order_Percentage
FROM `target.orders` as ord
JOIN `target.payments` as pmt
on ord.order_id = pmt.order_id
WHERE pmt.payment_installments !=0 AND order_delivered_customer_date IS NOT NULL
GROUP BY pmt.payment_installments
ORDER BY Total_Orders DESC
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EX

Row	payment_installment	Total_Orders	Order_Percentage
1	1	47584	48.89 %
2	2	12055	12.39 %
3	3	10146	10.43 %
4	4	6880	7.07 %
5	10	5137	5.28 %
6	5	5090	5.23 %
7	8	4122	4.24 %
8	6	3800	3.9 %
9	7	1560	1.6 %
10	9	618	0.64 %

Note: *The above table shows that nearly 70% of the orders are placed between 1 to 3 installments. 0 Installments and undelivered ordered are not taken into consideration.*

Additional Analysis

Q1 – Product category wise YoY sales trend

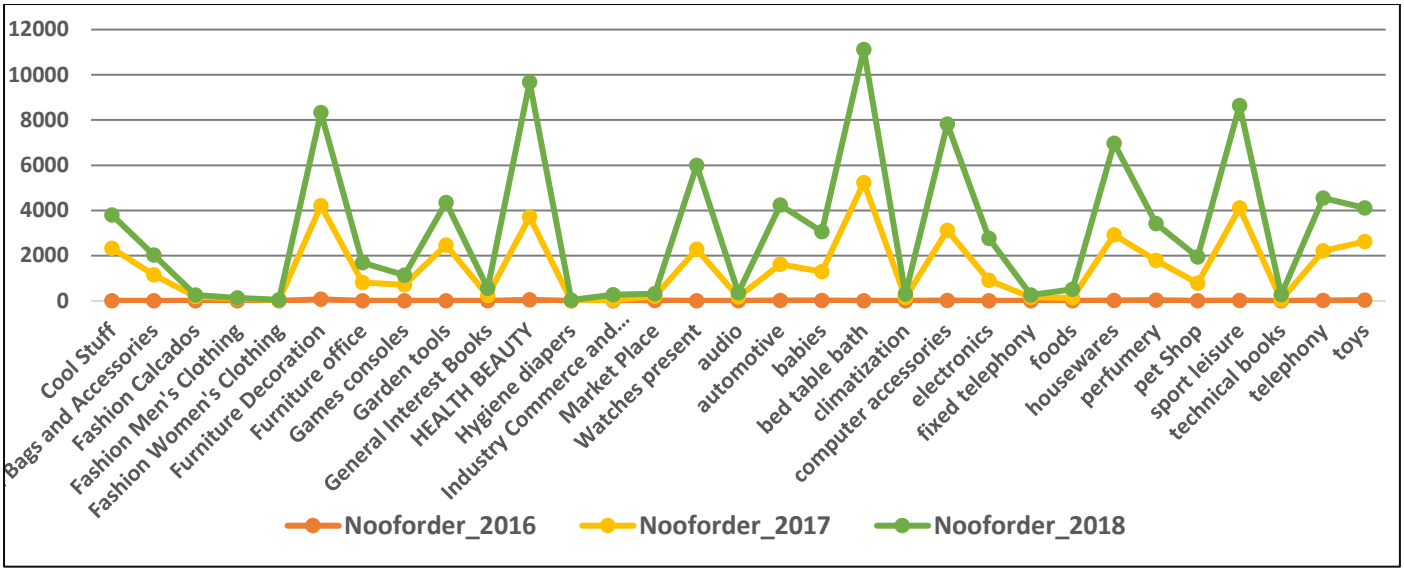
```
WITH newtable1 as
(SELECT
  EXTRACT(YEAR FROM ord.order_purchase_timestamp) as Yr,
  pd.product_category,
  count(ord.order_id) as Nooforder_2016
FROM `target.orders` as ord
JOIN `target.order_items` as odit
ON ord.order_id = odit.order_id
JOIN `target.products` as pd
ON pd.product_id = odit.product_id
GROUP BY Yr, pd.product_category
HAVING Yr = 2016 and pd.product_category IS NOT NULL
ORDER BY Nooforder_2016 DESC),
newtable2 as
(SELECT
  EXTRACT(YEAR FROM ord.order_purchase_timestamp) as Yr,
  pd.product_category,
  count(ord.order_id) as Nooforder_2017
FROM `target.orders` as ord
JOIN `target.order_items` as odit
ON ord.order_id = odit.order_id
JOIN `target.products` as pd
ON pd.product_id = odit.product_id
GROUP BY Yr, pd.product_category
HAVING Yr = 2017 and pd.product_category IS NOT NULL
ORDER BY Nooforder_2017 DESC),
newtable3 as
```

```sql
(SELECT
  EXTRACT(YEAR FROM ord.order_purchase_timestamp) as Yr,
  pd.product_category,
  count(ord.order_id) as Nooforder_2018
FROM `target.orders` as ord
JOIN `target.order_items` as odit
ON ord.order_id = odit.order_id
JOIN `target.products` as pd
ON pd.product_id = odit.product_id
GROUP BY Yr, pd.product_category
HAVING Yr = 2018 and pd.product_category IS NOT NULL
ORDER BY Nooforder_2018 DESC)
SELECT
  nt1.product_category,
  Nooforder_2016,
  Nooforder_2017,
  Nooforder_2018
FROM newtable1 as nt1
JOIN newtable2 as nt2
ON nt1.product_category = nt2.product_category
JOIN newtable3 as nt3
ON nt2.product_category = nt3.product_category
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	product_category	Nooforder_2016	Nooforder_2017	Nooforder_2018
1	Cool Stuff	8	2315	1473
2	Fashion Bags and Accessories	10	1141	880
3	Fashion Calcados	1	190	71
4	Fashion Men's Clothing	1	105	26
5	Fashion Women's Clothing	1	31	16
6	Furniture Decoration	69	4147	4118
7	Furniture office	6	802	883
8	Games consoles	10	707	420
9	Garden tools	5	2463	1879
10	General Interest Books	1	236	316



Insights: *The above Graphical representation shows that sales are getting increased for the products mainly for Furniture decoration, Health Beauty, bed table bath, computer accessories, housewares, sports leisure etc. We should focus on improving sales for products for which there is no significant YoY improvement.*

Q2 – State wise YoY sales trend

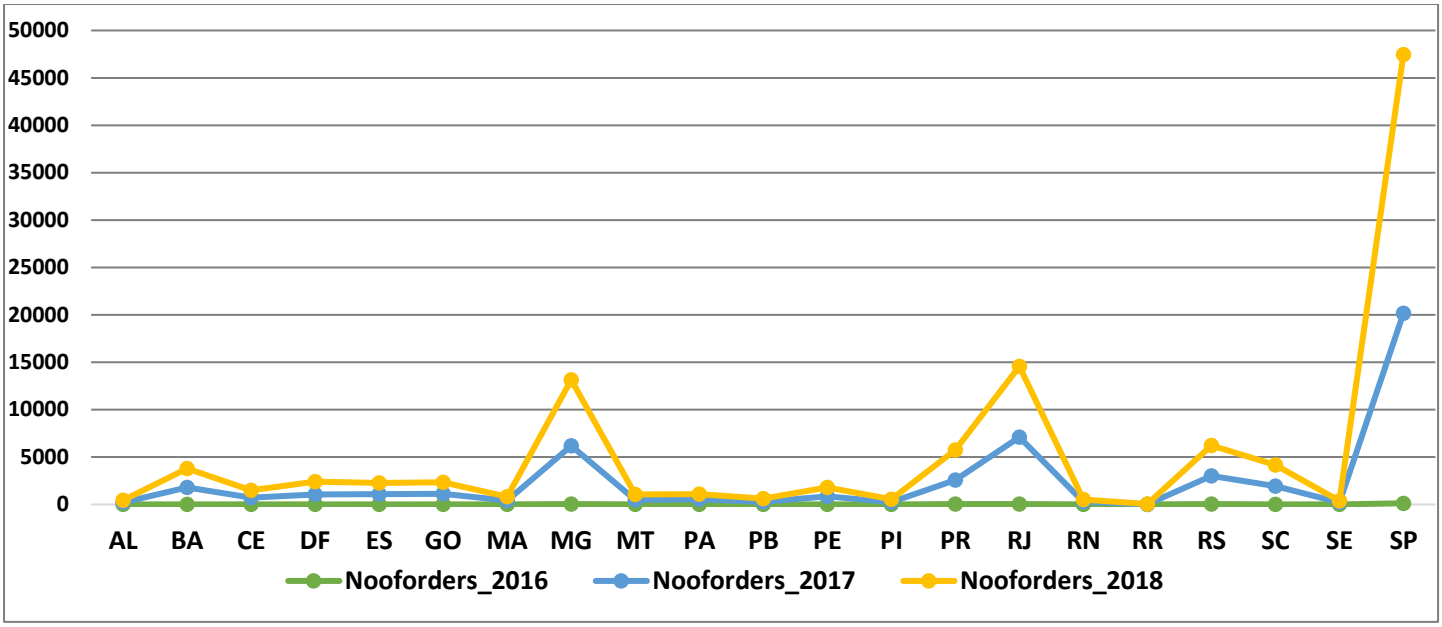
```
WITH mytb1 as
(SELECT
  EXTRACT(YEAR FROM ord.order_purchase_timestamp) as Yr,
  cust.customer_state as State,
  COUNT(ord.order_id) as Nooforders_2016
FROM `target.orders` as ord
JOIN `target.order_items` as odit
ON ord.order_id = odit.order_id
JOIN `target.products` as pd
ON pd.product_id = odit.product_id
JOIN `target.customers` as cust
ON cust.customer_id = ord.customer_id
GROUP BY Yr,state
HAVING Yr=2016
ORDER BY Nooforders_2016 DESC),
mytb2 as
(SELECT
  EXTRACT(YEAR FROM ord.order_purchase_timestamp) as Yr,
  cust.customer_state as State,
  COUNT(ord.order_id) as Nooforders_2017
FROM `target.orders` as ord
JOIN `target.order_items` as odit
ON ord.order_id = odit.order_id
JOIN `target.products` as pd
ON pd.product_id = odit.product_id
JOIN `target.customers` as cust
ON cust.customer_id = ord.customer_id
GROUP BY Yr,state
HAVING Yr=2017
ORDER BY Nooforders_2017 DESC),
mytb3 as
(SELECT
  EXTRACT(YEAR FROM ord.order_purchase_timestamp) as Yr,
```

```
  cust.customer_state as State,
  COUNT(ord.order_id) as Nooforders_2018
FROM `target.orders` as ord
JOIN `target.order_items` as odit
ON ord.order_id = odit.order_id
JOIN `target.products` as pd
ON pd.product_id = odit.product_id
JOIN `target.customers` as cust
ON cust.customer_id = ord.customer_id
GROUP BY Yr,state
HAVING Yr=2018
ORDER BY Nooforders_2018 DESC)
SELECT
  mt1.State,
  Nooforders_2016,
  Nooforders_2017,
  Nooforders_2018
FROM mytb1 AS mt1
JOIN mytb2 AS mt2
ON mt1.State = mt2.State
JOIN mytb3 AS mt3
ON mt2.State = mt3.State
ORDER BY mt1.State
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EX

Row	State	Nooforders_2016	Nooforders_2017	Nooforders_2018
1	AL	2	223	219
2	BA	4	1809	1986
3	CE	7	722	749
4	DF	8	1051	1347
5	ES	4	1074	1178
6	GO	8	1112	1213
7	MA	9	420	395
8	MG	48	6124	6957
9	MT	3	489	563
10	PA	6	551	523



Insights: *The above Graphical representation indicates that there is a significant increase in the sales in states MG, PR, RJ & SP, we should formula the strategies to achieve higher sales with the rest of the states.*

Q3 – Month wise YoY sales trend

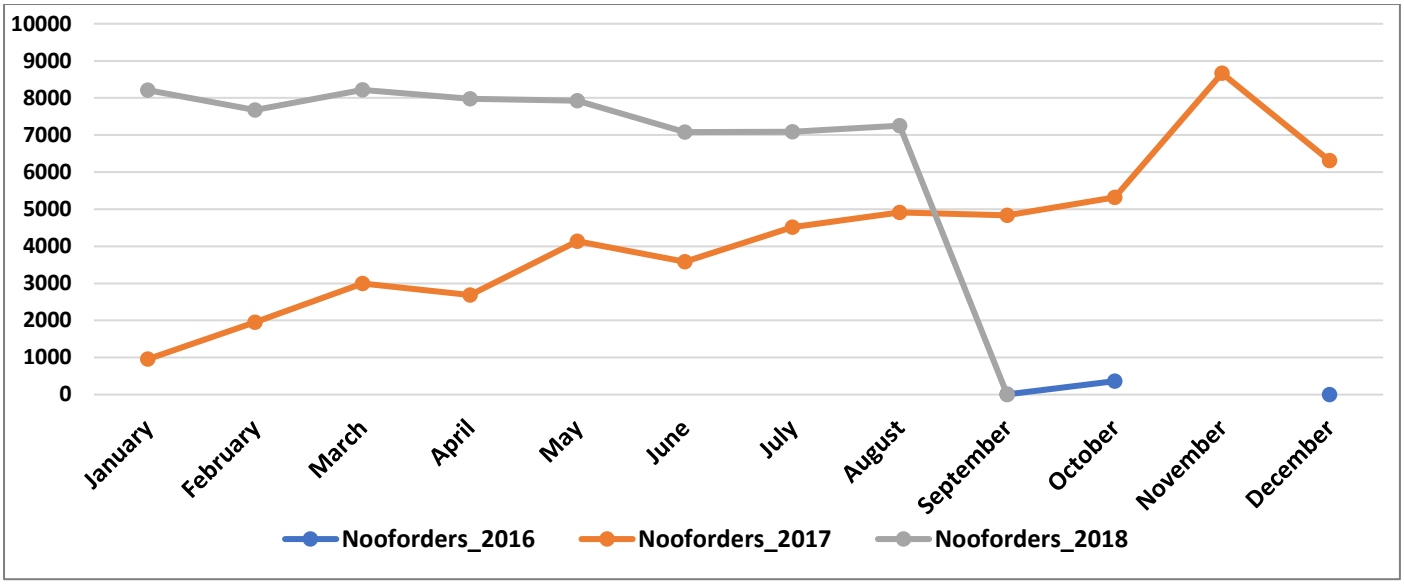
```
WITH mtable1 as
(SELECT
  EXTRACT(YEAR FROM ord.order_purchase_timestamp) as Yr,
  EXTRACT(MONTH FROM ord.order_purchase_timestamp) as Mnth,
  FORMAT_DATE('%B',ord.order_purchase_timestamp) as MonthName,
  COUNT(ord.order_id) as Nooforders_2016
FROM `target.orders` as ord
JOIN `target.order_items` as odit
ON ord.order_id = odit.order_id
JOIN `target.products` as pd
ON pd.product_id = odit.product_id
JOIN `target.customers` as cust
ON cust.customer_id = ord.customer_id
GROUP BY Yr,Mnth,MonthName
HAVING Yr= 2016
ORDER BY Mnth ASC),
mtable2 as
(SELECT
  EXTRACT(YEAR FROM ord.order_purchase_timestamp) as Yr,
  EXTRACT(MONTH FROM ord.order_purchase_timestamp) as Mnth,
  FORMAT_DATE('%B',ord.order_purchase_timestamp) as MonthName,
  COUNT(ord.order_id) as Nooforders_2017
FROM `target.orders` as ord
JOIN `target.order_items` as odit
ON ord.order_id = odit.order_id
JOIN `target.products` as pd
ON pd.product_id = odit.product_id
JOIN `target.customers` as cust
ON cust.customer_id = ord.customer_id
GROUP BY Yr,Mnth,MonthName
HAVING Yr= 2017
ORDER BY Mnth ASC),
mtable3 as
```

```sql
(SELECT
  EXTRACT(YEAR FROM ord.order_purchase_timestamp) as Yr,
  EXTRACT(MONTH FROM ord.order_purchase_timestamp) as Mnth,
  FORMAT_DATE('%B',ord.order_purchase_timestamp) as MonthName,
  COUNT(ord.order_id) as Nooforders_2018
FROM `target.orders` as ord
JOIN `target.order_items` as odit
ON ord.order_id = odit.order_id
JOIN `target.products` as pd
ON pd.product_id = odit.product_id
JOIN `target.customers` as cust
ON cust.customer_id = ord.customer_id
GROUP BY Yr,Mnth,MonthName
HAVING Yr= 2018
ORDER BY Mnth ASC)
SELECT
  mt2.MonthName,
  Nooforders_2016,
  Nooforders_2017,
  Nooforders_2018
FROM mtable1 as mt1
FULL OUTER JOIN mtable2 as mt2
ON mt1.Mnth = mt2.Mnth
FULL OUTER JOIN  mtable3 as mt3
ON mt2.Mnth = mt3.Mnth
ORDER BY mt2.Mnth
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION

Row	MonthName	Nooforders_2016	Nooforders_2017	Nooforders_2018
1	January	null	955	8208
2	February	null	1951	7672
3	March	null	3000	8217
4	April	null	2684	7975
5	May	null	4136	7925
6	June	null	3583	7078
7	July	null	4519	7092
8	August	null	4910	7248
9	September	6	4831	1
10	October	363	5322	null
11	November	null	8665	null
12	December	1	6308	null



Insights: *The above Graphical representation indicates that sales are comparatively higher in 2018 for all the months until Aug when compared to 2017, and the highest sales was occurred in Nov2017 and there is no significant drop in sales during the first quarter and second quarter 2018.*

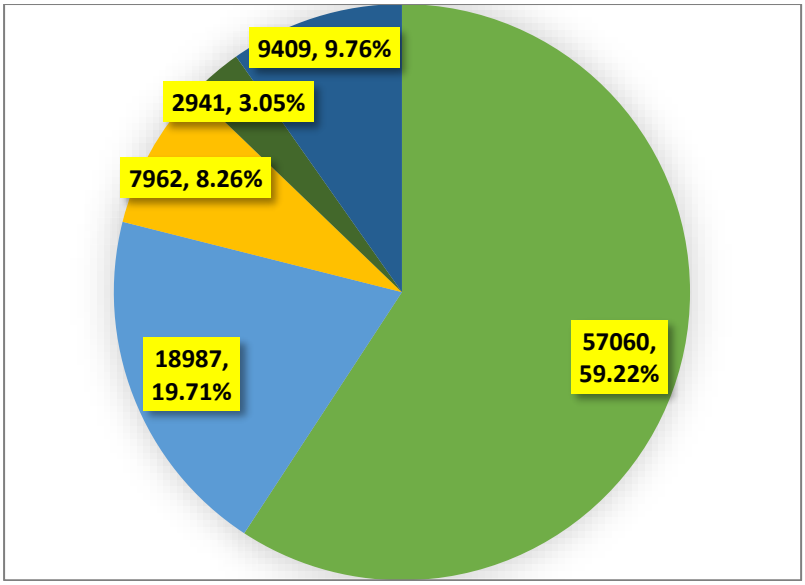
Q4 – Overall Customers order review status

```
SELECT
  review_score as ReviewScore,
  count(review_score) as NoOfReviews,
  CONCAT(ROUND(count(review_score)/(SUM(count(review_score)) OVER())*100,2),' %')
AS Review_Percentage
FROM `target.orders` as ord
JOIN `target.order_reviews` as ordrew
ON ord.order_id = ordrew.order_id
WHERE order_delivered_customer_date IS NOT NULL
GROUP BY ReviewScore
ORDER BY ReviewScore DESC
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAIL

Row	ReviewScore	NoOfReviews	Review_Percentage
1	5	57060	59.22 %
2	4	18987	19.7 %
3	3	7962	8.26 %
4	2	2941	3.05 %
5	1	9409	9.76 %



Insights: *Based on the above analysis we found that around 78% of the customer has given a positive rating. We should focus to achieving higher CSAT by identifying and analysing their pain areas.*

Q5 – Customer DSAT analysis

```
SELECT
  DISTINCT(review_comment_title) as cmnt
FROM `target.order_reviews`
WHERE review_comment_title IS NOT NULL
  AND review_score =1
ORDER BY cmnt ASC
```

Query results

JOB INFORMATION | RESULTS | JSON

Row	cmnt
38	Average
39	Awaiting delivery
40	Awaiting outcome
41	Awaiting the outcome
42	B2W Malandra
43	BAD
44	Bad
45	Bad incorrect product
46	Bad product
47	Bad quality product
48	Bad rough
49	Bad sinless
50	Bath towels

Note: *Filtered only the review rating 1 records to identify the actual DSAT status.*

Actionable Insights & Recommendations

1. Monthly seasonality in terms of the no. of orders being placed (from Question# 2.2), we found that sales are getting decreased specifically during last quarter i.e., September to December. We can boost the sales during those months by offering special discounts and providing promotional offers.

2. Around 65% of the sales are happening during the second half of the day, we may formulate some strategy to offer morning (E.g.: 9am deal, 10am deal etc) hourly sale specifically during weakened to grab the customers attention.

3. We observed that States RR, PB, RO, AC & PI has got highest Avg Freight time (from Question# 5.2), we have to identify the root cause and improve the logistics and supply chain efficiency.

4. We observed that States RR, AP, AM, AL, PA has got highest Avg Delivery time (from Question# 5.3), we have to identify the root cause of the delivery issues and fix the issues accordingly.

5. States like CE, DF, ES, GO, PA, PB, RN, RR, SE there is no improvement YoY (from Question 2, Additional Analysis), we've to device an appropriate marketing strategy to improve the sales in those regions.

6. By Analysing the Customers review comment (from Question 5, Additional Analysis) we found that the main reason for DSAT is Receiving wrong product, Defective product received, Partial delivery, Delivery delay, Non receipt of product, Lack of information, Product delivered without packing, Inappropriate packing etc.

7. We may offer product at Discounted price which is lesser than market price or price of our competitor to boost sales for the states or cities which has recorded very less orders or No orders, by comprising little bit on our profit margin.