SQL Target Business CASE

MUKESH M.

#1-1) Data type of columns in a table

Table Name: Orders

```
SELECT column_name, data_type
FROM dsmllecturesql01.Business_Case_Target_SQL.INFORMATION_SCHEMA.COLUMNS
WHERE table_name = 'orders';
```

Full name	Constraint	data type	description
order_id	UNIQUE, NOT NULL	STRING	Primary key: Unique id for every order
customer_id	NOT NULL	STRING	Customer ID: FK that refers to the Customer ID (PK) in the customers table
order_status	NULLABLE	STRING	Status of the order made i.e., delivered, shipped etc.
order_purchase_timestamp	NULLABLE	TIMESTAMP	Timestamp of purchase
order_approved_at	NULLABLE	TIMESTAMP	
order_delivered_carrier_date	NULLABLE	TIMESTAMP	delivery date at which carrier made the delivery.
order_delivered_customer_date	NULLABLE	TIMESTAMP	date at which customer got the product.
order_estimated_delivery_date	NULLABLE	TIMESTAMP	estimated delivery date of the products.

#1-2) Duration of the sample dataset: (Time period for which the data is given)

```
SELECT
       MIN (order_purchase_timestamp) as duration_start,
       MAX (order_purchase_timestamp) as duration_end
FROM
       `dsmllecturesql01.Business_Case_Target_SQL.orders`;
```

EXECUTION RESULT:

duration_start	duration_end
2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

#1-3) Cities and states in the dataset: (Cities and States of customers ordered during the given period)

A) TOP 10 Cities (by order count):

```
SELECT
  distinct c.customer_city,
  c.customer_state,
  COUNT(o.order_id) over(partition by c.customer_city) as count_of_orders
FROM
  `Business_Case_Target_SQL.customers` c
```

```
JOIN
  `Business_Case_Target_SQL.orders` o
ON c.customer_id = o.customer_id
order by count_of_orders DESC;
```

EXECUTION RESULT (*Top 10 cities by order count*):

customer_city	customer_state	count_of_orders
sao paulo	SP	15540
rio de janeiro	RJ	6882
belo horizonte	MG	2773
Brasilia	DF	2131
Curitiba	PR	1521
Campinas	SP	1444
porto alegre	RS	1379
Salvador	BA	1245
Guarulhos	SP	1189
sao bernardo do campo	SP	938

B) TOP 10 States (by order count):

```
SELECT
  distinct c.customer_state,
  COUNT(o.order_id) over(partition by c.customer_state) as count_of_orders
FROM
  `Business_Case_Target_SQL.customers` c
JOIN
  `Business_Case_Target_SQL.orders` o
ON c.customer_id = o.customer_id
order by count_of_orders DESC;
```

EXECUTION RESULT (*Top 10 states by order count*):

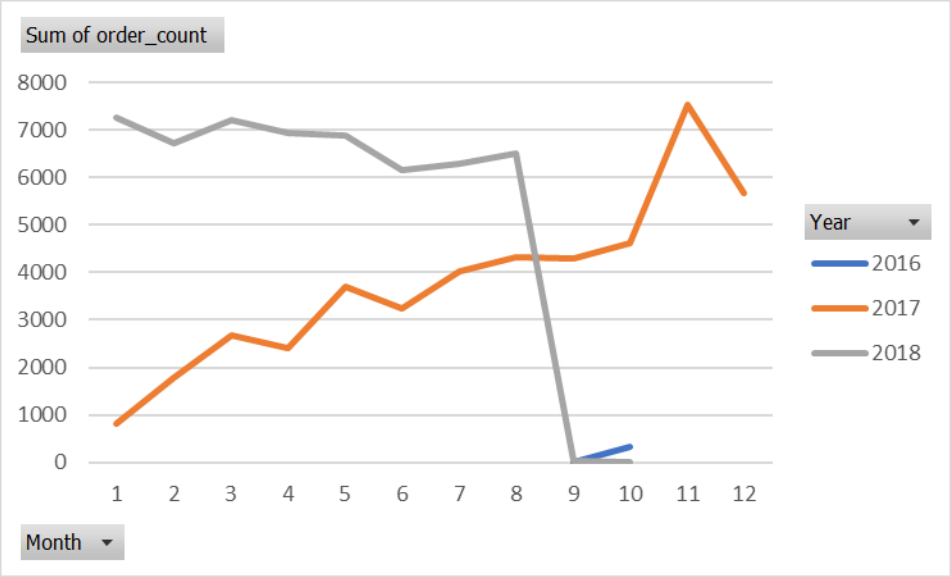
customer_state	count_of_orders
SP	41746
RJ	12852
MG	11635
RS	5466
PR	5045
SC	3637
BA	3380
DF	2140
ES	2033
GO	2020

#2-1) Month & year-wise number of orders: (Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario? Can we see some seasonality with peaks at specific months?)

```sql
SELECT
       EXTRACT (year from order_purchase_timestamp) as Year,
       EXTRACT (month from order_purchase_timestamp) as Month,
       COUNT(distinct order_id) as order_count
FROM
       `Business_Case_Target_SQL.orders`
group by Year, Month
order by Year, Month;
```

EXECUTION RESULT:

Year	Month	order_count
2016	9	4
2016	10	324
2016	12	1
2017	1	800
2017	2	1780
2017	3	2682
2017	4	2404
2017	5	3700
2017	6	3245
2017	7	4026
2017	8	4331
2017	9	4285
2017	10	4631
2017	11	7544
2017	12	5673
2018	1	7269
2018	2	6728
2018	3	7211
2018	4	6939
2018	5	6873
2018	6	6167
2018	7	6292
2018	8	6512
2018	9	16
2018	10	4



#2-2) What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)?

In writing this query, I have made the following two assumptions:

1. Following is the 1st assumption about the time-range:

Time of the day	Hour Range
DAWN	5-7
MORNING	7-12
AFTERNOON	12-17
EVENING	17-19
NIGHT	Everything else

2. The 2nd assumption is that the database stores the time format in UTC, so needs to be converted into Brazilian time (UTC – 3).

```
SELECT
  CASE
    WHEN CAST(o.order_purchase_timestamp AS TIME) BETWEEN '2:00:00.001' AND
'4:00:00.000' THEN 'DAWN'
    WHEN CAST(o.order_purchase_timestamp AS TIME) BETWEEN '4:00:00.001'
AND '9:00:00.000' THEN 'MORNING'
    WHEN CAST(o.order_purchase_timestamp AS TIME) BETWEEN '9:00:00.001'AND '
14:00:00.000' THEN 'AFTERNOON'
    WHEN CAST(o.order_purchase_timestamp AS TIME) BETWEEN '14:00:00.001'
AND '16:00:00.000' THEN 'EVENING'
    ELSE 'NIGHT'
END
  AS time_of_the_day,
  COUNT(distinct o.order_id) AS order_count
FROM
  `Business_Case_Target_SQL.orders` o
JOIN
  `Business_Case_Target_SQL.customers` c
```

```
ON
  o.customer_id = c.customer_id
GROUP BY
  time_of_the_day
ORDER BY
  order_count DESC;
```

EXECUTION RESULT:

time_of_the_day	order_count
NIGHT	50488
AFTERNOON	30052
EVENING	13023
MORNING	5096
DAWN	782

3-1) Get month on month orders by states (year/month-wise)

```
SELECT
    EXTRACT (month from o.order_purchase_timestamp) as Month,
    EXTRACT (year from o.order_purchase_timestamp) as Year,
    c.customer_state,
    COUNT(distinct o.order_id) as order_count
  FROM
    `Business_Case_Target_SQL.orders` o
  JOIN
    `Business_Case_Target_SQL.customers` c
  ON o.customer_id = c.customer_id
  group by Month, Year, c.customer_state
  order by Year, Month, customer_state;
```

EXECUTION RESULT:

Month	Year	customer_state	order_count
9	2016	RR	1
9	2016	RS	1
9	2016	SP	2
10	2016	AL	2
10	2016	BA	4
10	2016	CE	8
10	2016	DF	6
10	2016	ES	4
10	2016	GO	9
10	2016	MA	4
10	2016	MG	40
10	2016	MT	3
10	2016	PA	4
10	2016	PB	1
10	2016	PE	7
10	2016	PI	1
10	2016	PR	19

10	2016	RJ	56
10	2016	RN	4
10	2016	RR	1
10	2016	RS	24
10	2016	SC	11
10	2016	SE	3
10	2016	SP	113
12	2016	PR	1
1	2017	AC	2
1	2017	AL	2
1	2017	BA	25
1	2017	CE	9
1	2017	DF	13
1	2017	ES	12
1	2017	GO	18
1	2017	MA	9
1	2017	MG	108
1	2017	MS	1
1	2017	MT	11
1	2017	PA	12
1	2017	PB	2
1	2017	PE	9
1	2017	PI	7
1	2017	PR	65
1	2017	RJ	97
1	2017	RN	5
1	2017	RO	3
1	2017	RS	54
1	2017	SC	31
1	2017	SE	4
1	2017	SP	299
1	2017	TO	2
2	2017	AC	3

 # 3-2) Distribution of customers across the states in Brazil

```
SELECT
    c.customer_state,
    COUNT(distinct o.customer_id) as customer_count_who_ordered
FROM
    `Business_Case_Target_SQL.orders` o
    JOIN
    `Business_Case_Target_SQL.customers` c
ON o.customer_id = c.customer_id
group by
    c.customer_state
order by customer_count_who_ordered desc;
```

EXECUTION RESULT:

customer_state	customer_count_who_ordered
SP	41746
RJ	12852
MG	11635
RS	5466
PR	5045
SC	3637
BA	3380
DF	2140
ES	2033
GO	2020
PE	1652
CE	1336
PA	975
MT	907
MA	747
MS	715
PB	536
PI	495
RN	485
AL	413
SE	350
TO	280
RO	253
AM	148
AC	81
AP	68
RR	46

 # 4-1) Get % increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only) - You can use "payment_value" column in payments table

Assumptions in the Query:

- Months included are Jan to Aug only.
- Order status considered in calculations is 'delivered' orders only.

```
WITH
  yearlyRevenueInfo AS (
  SELECT
    EXTRACT (year
    FROM
      o.order_purchase_timestamp) AS Year,
    SUM(p.payment_value) AS total_revenue
  FROM
    `Business_Case_Target_SQL.orders` o
  JOIN
    `Business_Case_Target_SQL.payments` p
  ON
```

```
    o.order_id = p.order_id
  WHERE
    o.order_status = 'delivered' AND
    EXTRACT (month FROM o.order_purchase_timestamp) BETWEEN 0 AND 8
  GROUP BY
    Year
  ORDER BY
    Year )
SELECT
  Year,
  total_revenue,
  ROUND(((total_revenue - LAG (total_revenue) OVER (ORDER BY Year))/ LAG (to
tal_revenue) OVER (ORDER BY Year))*100, 2) AS growth_prcnt
FROM
  yearlyRevenueInfo;
```

EXECUTION RESULT:

Year	total_revenue	growth_prcnt
2017	3473862.76	null
2018	8452975.2	143.33

#4-2) Mean & Sum of price and freight value by customer state

```
  SELECT
       c.customer_state as Customer_State,
       ROUND(SUM(i.freight_value), 2) As Total_freight_value,
       ROUND(AVG(i.freight_value), 2) As Avg_freight_value,
       ROUND(SUM(i.price), 2) As Total_price,
       ROUND(AVG(i.price), 2) As Avg_price
  FROM
       `Business_Case_Target_SQL.order_items` i
  JOIN
       `Business_Case_Target_SQL.orders` o
  ON i.order_id = o.order_id
  JOIN
       `Business_Case_Target_SQL.customers` c
  ON c.customer_id = o.customer_id
  group by c.customer_state
  order by c.customer_state;
```

EXECUTION RESULT (first 10 rows):

Customer_State	Total_freight_value	Avg_freight_value	Total_price	Avg_price
AC	3686.75	40.07	15982.95	173.73
AL	15914.59	35.84	80314.81	180.89
AM	5478.89	33.21	22356.84	135.5
AP	2788.5	34.01	13474.3	164.32
BA	100156.68	26.36	511349.99	134.6
CE	48351.59	32.71	227254.71	153.76
DF	50625.5	21.04	302603.94	125.77
ES	49764.6	22.06	275037.31	121.91

GO	53114.98	22.77	294591.95	126.27
MA	31523.77	38.26	119648.22	145.2

#5-1) Calculate days between purchasing, delivering and estimated delivery.

```
SELECT
  order_id,
ifnull(date_diff(DATE(order_estimated_delivery_date), DATE(order_purchase_timestamp
), day), 0) as  estimated_delivery_time,
ifnull(date_diff(DATE(order_delivered_carrier_date), DATE(order_purchase_timestamp)
, day), 0) as  carrier_delivery_time,
ifnull(date_diff(DATE(order_delivered_customer_date), DATE(order_purchase_timestamp
), day), 0) as customer_delivery_time
FROM
`Business_Case_Target_SQL.orders`
order by customer_delivery_time desc;
```

EXECUTION RESULT (first 10 rows):

order_id	estimated_delivery_time	carrier_delivery_time	customer_delivery_time
ca07593549f1816d26a572e06dc1eab6	29	15	210
1b3190b2dfa9d789e1f14c05b647a14a	20	3	208
440d0d17af552815d15a9e41abe49359	31	8	196
285ab9426d6982034523a855f55a885e	29	1	195
2fb597c2f772eca01b1f5c561bf6cc7b	40	5	195
0f4519c5f1c541ddec9f21b3bddd533a	33	13	194
47b40429ed8cce3aee9199792275433f	16	34	191
2fe324febf907e3ea3f2aa9650869fa5	23	4	190
c27815f7e3dd0b926b58552628481575	26	6	188
2d7561026d542c8dbd8f0daeadf67a43	29	1	188

#5-2) Find time_to_delivery & diff_estimated_delivery. Formula for the same given below:

- **time_to_delivery = order_purchase_timestamp-order_delivered_customer_date**
- **diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date**

```
SELECT
  order_id,
  ifnull(date_diff(DATE(order_delivered_customer_date), DATE(order_purchase_
timestamp), day), 0) as time_to_delivery,
  ifnull(date_diff(DATE(order_estimated_delivery_date),
DATE(order_delivered_customer_date), day), 0) as diff_estimated_delivery
FROM
`Business_Case_Target_SQL.orders`
order by diff_estimated_delivery desc;
```

EXECUTION RESULT (first 10 rows):

order_id	time_to_delivery	diff_estimated_delivery
0607f0efea4b566f1eb8f7d3c2397320	3	147
c72727d29cde4cf870d569bf65edabfd	7	140
eec7f369423b033e549c02f3c5381205	21	135
c2bb89b5c1dd978d507284be78a04cb2	17	124
40dc2ba6f322a17626aac6244332828c	8	109
1a695d543b7302aa9446c8d5fbd632bf	12	84
39e0115911bf404857e14baa7f097feb	12	83
559eea5a72341a4c82dbce9884277cb7	13	78
38930f76efb00b138f4d632e4d557341	11	78
c5132855100a12d63ed4e8ae05f9594d	12	78

#5-3)
Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery

```
SELECT
  customer_state as state,
  ROUND(AVG(freight_value),2) as avg_freight_value,
  ROUND(AVG(time_to_delivery),2) as avg_time_to_delivery,
  ROUND(AVG(diff_estimated_delivery), 2) as avg_diff_estimated_delivery
FROM
(
  SELECT
    o.order_id,
    o.customer_id,
    i.freight_value,
    ifnull(date_diff(DATE(o.order_delivered_customer_date), DATE(o.order_pur
chase_timestamp), day), 0) as time_to_delivery,
    ifnull(date_diff(DATE(o.order_estimated_delivery_date), DATE(o.order_del
ivered_customer_date), day), 0) as diff_estimated_delivery
  FROM
`Business_Case_Target_SQL.orders` o
  JOIN
`Business_Case_Target_SQL.order_items` i
  ON o.order_id = i.order_id
 ) AS t
JOIN
`Business_Case_Target_SQL.customers` c
ON c.customer_id = t.customer_id
group by customer_state
order by state;
```

EXECUTION RESULT:

State	avg_freight_value	avg_time_to_delivery	avg_diff_estimated_delivery
AC	40.07	20.46	20.75
AL	35.84	23.51	8.4
AM	33.21	26.02	19.69
AP	34.01	27.88	18.17

BA	26.36	18.61	10.65
CE	32.71	20.19	10.71
DF	21.04	12.62	11.94
ES	22.06	15.37	10.5
GO	22.77	14.97	12
MA	38.26	20.96	9.62
MG	20.63	11.73	13.13
MS	23.37	15.31	11.12
MT	28.17	17.6	14.32
PA	35.83	23.13	13.91
PB	42.72	20	12.69
PE	32.92	17.62	13
PI	39.15	18.64	11.12
PR	20.53	11.7	13.27
RJ	20.96	14.63	11.66
RN	35.65	18.98	13.74
RO	41.07	19.3	19.68
RR	42.98	24.92	16.21
RS	21.74	14.89	13.9
SC	21.47	14.67	11.36
SE	36.65	20.86	9.74
SP	15.15	8.48	10.97
TO	37.25	17.12	12.14

#5-5) Sort the data to get the following:

- **Top 5 states with "highest" average freight value - sort in "desc" limit 5**

```
SELECT
  customer_state as state,
  ROUND(AVG(freight_value),2) as avg_freight_value
FROM
(
  SELECT
    o.order_id,
    o.customer_id,
    i.freight_value
  FROM
`Business_Case_Target_SQL.orders` o
  JOIN
`Business_Case_Target_SQL.order_items` i
  ON o.order_id = i.order_id
) AS t
JOIN
`Business_Case_Target_SQL.customers` c
ON c.customer_id = t.customer_id
group by customer_state
order by avg_freight_value desc
LIMIT 5;
```

EXECUTION RESULT:

State	avg_freight_value
RR	42.98
PB	42.72
RO	41.07
AC	40.07
PI	39.15

- **Top 5 states with "lowest" average freight value - sort in "asc" limit 5**

```
SELECT
  customer_state as state,
  ROUND(AVG(freight_value),2) as avg_freight_value
FROM
(
  SELECT
    o.order_id,
    o.customer_id,
    i.freight_value
  FROM
`Business_Case_Target_SQL.orders` o
  JOIN
`Business_Case_Target_SQL.order_items` i
  ON o.order_id = i.order_id
) AS t
JOIN
`Business_Case_Target_SQL.customers` c
ON c.customer_id = t.customer_id
group by customer_state
order by avg_freight_value ASC
LIMIT 5;
```

EXECUTION RESULT:

State	avg_freight_value
SP	15.15
PR	20.53
MG	20.63
RJ	20.96
DF	21.04

#5-6) Sort the data to get the following:

- **Top 5 states with "highest" average time_to_delivery - sort in "desc" limit 5**

```
SELECT
  customer_state as state,
  ROUND(AVG(time_to_delivery),2) as avg_time_to_delivery
FROM
(
  SELECT
    o.order_id,
    o.customer_id,
    ifnull(date_diff(DATE(o.order_delivered_customer_date), DATE(o.order_pur
chase_timestamp), day), 0) as time_to_delivery
```

```
  FROM
`Business_Case_Target_SQL.orders` o
  JOIN
`Business_Case_Target_SQL.order_items` i
  ON o.order_id = i.order_id
) AS t
JOIN
`Business_Case_Target_SQL.customers` c
ON c.customer_id = t.customer_id
group by customer_state
order by avg_time_to_delivery desc
LIMIT 5;
```

EXECUTION RESULT:

State	avg_time_to_delivery
AP	27.88
AM	26.02
RR	24.92
AL	23.51
PA	23.13

- **Top 5 states with "lowest" average time_to_delivery - sort in "asc" limit 5**

```
SELECT
  customer_state as state,
  ROUND(AVG(time_to_delivery),2) as avg_time_to_delivery
FROM
(
  SELECT
    o.order_id,
    o.customer_id,
    ifnull(date_diff(DATE(o.order_delivered_customer_date), DATE(o.order_pur
chase_timestamp), day), 0) as time_to_delivery
  FROM
`Business_Case_Target_SQL.orders` o
  JOIN
`Business_Case_Target_SQL.order_items` i
  ON o.order_id = i.order_id
) AS t
JOIN
`Business_Case_Target_SQL.customers` c
ON c.customer_id = t.customer_id
group by customer_state
order by avg_time_to_delivery ASC
LIMIT 5;
```

EXECUTION RESULT:

State	avg_time_to_delivery
SP	8.48
PR	11.7
MG	11.73
DF	12.62
RJ	14.63

#5-7) Sort the data to get the following:

- **Top 5 states where delivery is really fast compared to estimated date**

```
SELECT
  customer_state as state,
  ROUND(AVG(diff_estimated_delivery), 2) as avg_diff_estimated_delivery
FROM
(
  SELECT
    o.order_id,
    o.customer_id,
    ifnull(date_diff(DATE(o.order_estimated_delivery_date), DATE(o.order_del
ivered_customer_date), day), 0) as diff_estimated_delivery
  FROM
`Business_Case_Target_SQL.orders` o
  JOIN
`Business_Case_Target_SQL.order_items` i
  ON o.order_id = i.order_id
) AS t
JOIN
`Business_Case_Target_SQL.customers` c
ON c.customer_id = t.customer_id
group by customer_state
order by avg_diff_estimated_delivery ASC
LIMIT 5;
```

EXECUTION RESULT:

State	avg_diff_estimated_delivery
AL	8.4
MA	9.62
SE	9.74
ES	10.5
BA	10.65

- **Top 5 states where delivery is not that fast compared to estimated date**

```
SELECT
  customer_state as state,
  ROUND(AVG(diff_estimated_delivery), 2) as avg_diff_estimated_delivery
FROM
(
  SELECT
    o.order_id,
    o.customer_id,
    ifnull(date_diff(DATE(o.order_estimated_delivery_date), DATE(o.order_del
ivered_customer_date), day), 0) as diff_estimated_delivery
  FROM
`Business_Case_Target_SQL.orders` o
  JOIN
`Business_Case_Target_SQL.order_items` i
  ON o.order_id = i.order_id
) AS t
```

```
JOIN
`Business_Case_Target_SQL.customers` c
ON c.customer_id = t.customer_id
group by customer_state
order by avg_diff_estimated_delivery desc
LIMIT 5;
```

EXECUTION RESULT:

State	avg_diff_estimated_delivery
AC	20.75
AM	19.69
RO	19.68
AP	18.17
RR	16.21

#6-1) Payment type analysis:

- **Month over Month count of orders for different payment types**

```
SELECT Month, Year, payment_type, COUNT(order_id) as Orders_Count
FROM
(
  SELECT
    c.customer_id,
    o.order_id,
    EXTRACT (month from o.order_purchase_timestamp) as Month,
    EXTRACT (year from o.order_purchase_timestamp) as Year,
    p.payment_type
  FROM
  `Business_Case_Target_SQL.orders` o
  JOIN
  `Business_Case_Target_SQL.customers` c
  ON o.customer_id = c.customer_id
   JOIN
  `Business_Case_Target_SQL.payments` p
  ON o.order_id = p.order_id
) As t

group by Year, Month,
payment_type
order by Year, Month,
payment_type;
```

EXECUTION RESULT:

Month	Year	payment_type	Orders_Count
9	2016	credit_card	3
10	2016	UPI	63
10	2016	credit_card	254
10	2016	debit_card	2

10	2016	Voucher	23
12	2016	credit_card	1
1	2017	UPI	197
1	2017	credit_card	583
1	2017	debit_card	9
1	2017	Voucher	61

#6-2) Payment type analysis:

- **Count of orders based on the no. of payment installments**

```
SELECT
	payment_installments,
	COUNT(order_id) as Count_of_Orders
FROM
(
	SELECT
		c.customer_id,
		o.order_id,
		p.payment_installments
	FROM
		`Business_Case_Target_SQL.orders` o
	JOIN
		`Business_Case_Target_SQL.customers` c
	ON o.customer_id = c.customer_id
	JOIN
		`Business_Case_Target_SQL.payments` p
	ON o.order_id = p.order_id
) As t

group by payment_installments
order by Count_of_Orders desc;
```

EXECUTION RESULT (All Rows):

payment_installments	Count_of_Orders
1	52546
2	12413
3	10461
4	7098
10	5328
5	5239
8	4268
6	3920
7	1626
9	644
12	133
15	74
18	27
11	23
24	18
20	17

13	16
14	15
17	8
16	5
21	3
0	2
23	1
22	1

#7 - ACTIONABLE INSIGHTS:

1. **Distribution of E-Commerce orders across cities:** **Sao Paulo, Rio de Janeiro, Belo Horizonte, Brasilia, Curitiba** are the **top 5 cities by order volume**.

2. **Distribution of orders across states:** **SP, RJ, MG and RS** are the **top 4 states by order volume**.

3. **Generally, there is a trend of growth in E-Commerce in Brazil from Jan 2017 to Aug 2018, at least the volumes are holding up.** The data for the last 3 months (Oct, Nov & Dec) of 2016 and last 2 months (Sept & Oct) in 2018 is outlier data (probably the data shared is incomplete). However, with the limited data available, it seems Jan-Aug is the main season for Target Brazil, whereas Sep–Dec months are off-season.

4. Most customers in Brazil prefer to shop in the ***Night*** **FOLLOWED by** ***Afternoon*** **&** ***Morning***. Very few customers order at *Dawn*.

5. Top 5 selling product categories are: **Health Beauty, Watches present, bed table bath, sport leisure & computer accessories.**

6. More often than not, the **best reviews** have come from customers who usually **receive orders before estimated delivery (diff_estimated_delivery < 0)**. Thus, it appears that delivery before estimated time is a key factor for customer delight.

    - **Query to fetch earlier than estimated delivery and the review score:**

```
SELECT
  distinct review_score,
  COUNT(distinct order_id) over(partition by review_score) as delivere
d_before_estimated_delivery
FROM
(
  select
    o.order_id,
    r.review_score,
    ifnull(date_diff(DATE(o.order_delivered_customer_date), DATE(o.ord
er_purchase_timestamp), day), 0) as time_to_delivery,
    ifnull(date_diff(DATE(o.order_delivered_customer_date), DATE(o.ord
er_estimated_delivery_date), day), 0) as diff_estimated_delivery
  FROM
    `Business_Case_Target_SQL.orders` o
```

```
  JOIN
    `Business_Case_Target_SQL.order_items` i
  ON i.order_id = o.order_id
  JOIN
    `Business_Case_Target_SQL.order_reviews` r
  ON r.order_id = o.order_id
) as t
where diff_estimated_delivery < 0
order by review_score desc;
```

- **Query to fetch total number of reviewed orders:**

```
select
  distinct review_score,
  COUNT(order_id) over(partition by review_score) as total_reviewed_or
ders
from `Business_Case_Target_SQL.order_reviews`
order by review_score desc;
```

EXECUTION RESULT:

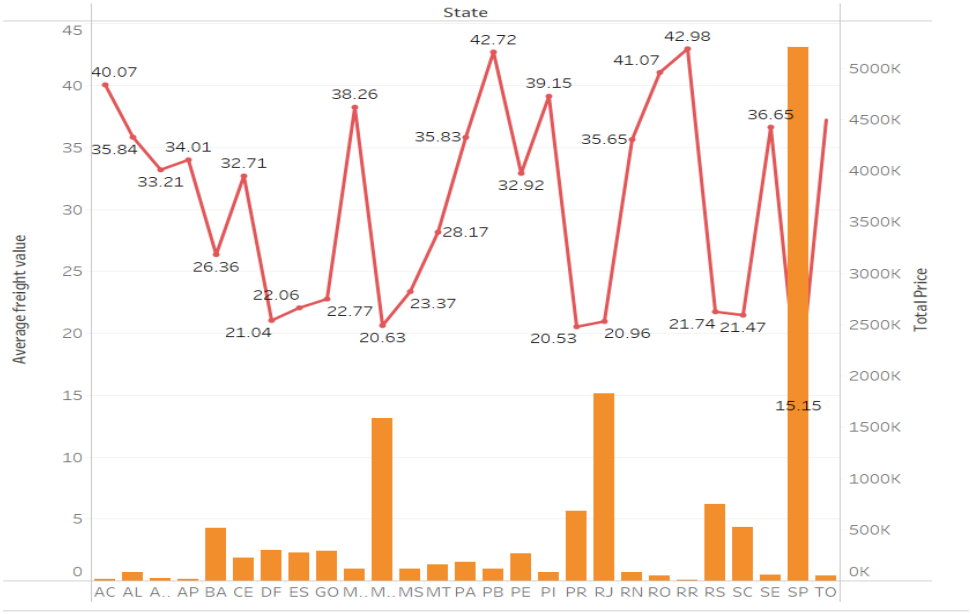
Earlier than estimated delivery:

review_score	delivered_before_estimated_delivery
5	55102
4	17993
3	7067
2	2334
1	5841

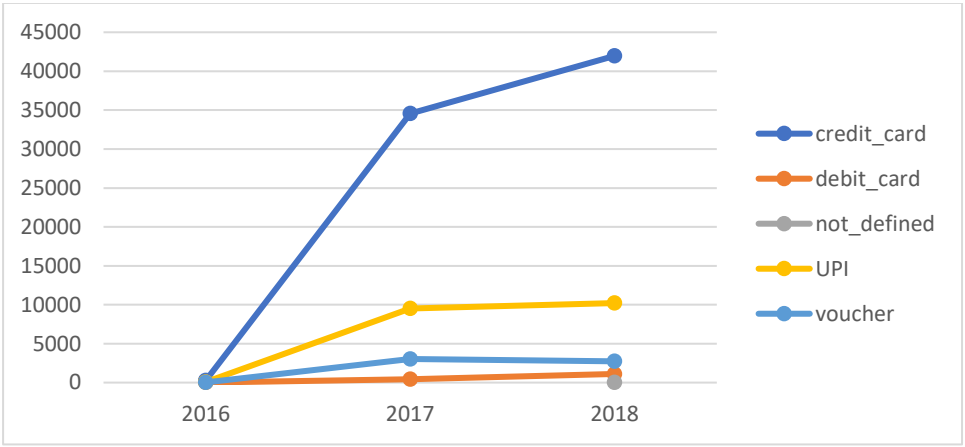
Total no. of orders:

review_score	total_reviewed_orders
5	57328
4	19142
3	8179
2	3151
1	11424

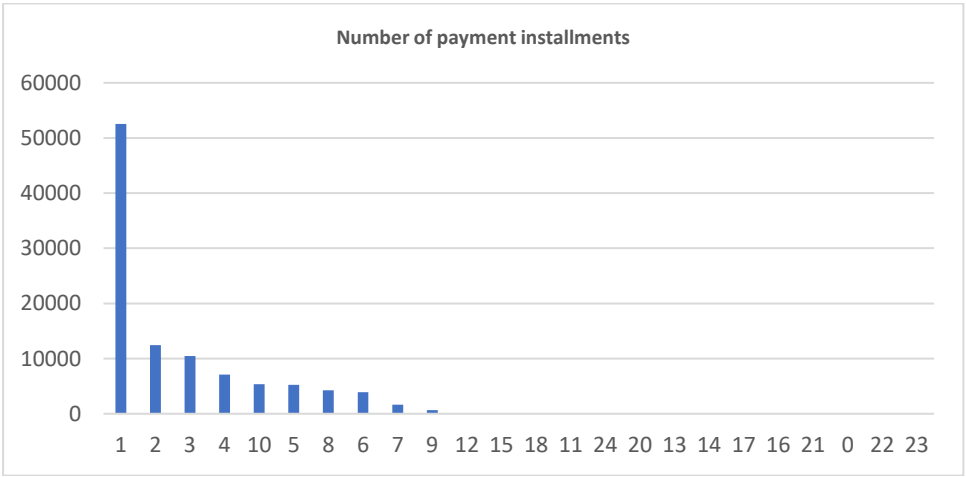
7. States with the **biggest number of orders placed (volumes)** generally have least ***freight charges*** on an average.



8. **Credit card & UPI are the most popular payment methods**: Although the dataset is for a short period, we can clearly see the YoY growth in these 2 payment types:



9. **Majority of the customers like to make payments in EMIs with a preference for the following number of EMIs: 1, 2, 3, 4, 10, 5, 8, 6, 7 & 9. A big chunk of orders was placed with 1 EMI / installment.**



#8 - RECOMMENDATIONS:

1. Focus on the top 5 cities within each of the top 5 states through flash sales during key dates on the holiday calendar. Run digital / offline marketing campaigns for these growth regions.
2. Re-organize how Target pushes/sells top selling product categories within the top selling regions. Recommendation system should suggest top selling products within these winner categories while the customers are browsing on the E-Com platform.
3. Target may think about getting into white label brands for some of the top selling product categories.
4. Focus on a strategy where the orders are delivered within the estimated delivery time. Exceeding estimated delivery leads to customer dis-satisfaction and poor review scores.
5. Target should re-organize its warehousing / inventory in such a way to reduce freight charges by working with the sellers.
6. Target should leverage the user reviews to better align its product and operations strategy.
7. Target should forge partnerships with UPI payment & credit card provider companies, as these happen to be the most popular payment types.
8. Target should lure customers by offering 0% EMI for users looking for 1 EMI option, as this happens to be the most popular “no. of EMIs” based on the data.