Target Case Study

Description:

Target is a globally renowned brand and a prominent retailer in the United States. Target makes itself a preferred shopping destination by offering outstanding value, inspiration, innovation and an exceptional guest experience that no other retailer can deliver.

This particular business case focuses on the operations of Target in Brazil and provides insightful information about 100,000 orders placed between 2016 and 2018. The dataset offers a comprehensive view of various dimensions including the order status, price, payment and freight performance, customer location, product attributes, and customer reviews.

By analyzing this extensive dataset, it becomes possible to gain valuable insights into Target's operations in Brazil. The information can shed light on various aspects of the business, such as order processing, pricing strategies, payment and shipping efficiency, customer demoChart:ics, product characteristics, and customer satisfaction levels.

Problem Statement:

Assuming you are a data analyst/ scientist at Target, you have been assigned the task of analyzing the given dataset to extract valuable insights and provide actionable recommendations.

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

Query:

```
select
column_name,
data_type
from Target.INFORMATION_SCHEMA.COLUMNS
where table_name='customers'
```

Output:

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

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

Query:

```
with cte as (
SELECT
  MIN(order_purchase_timestamp) AS first_timestamp,
  MAX(order_purchase_timestamp) AS earliest_timestamp,
  DATE_DIFF(DATE(MAX(order_purchase_timestamp)),
DATE(MIN(order_purchase_timestamp)), YEAR) AS range_orders_year,
  DATE_DIFF(DATE(MAX(order_purchase_timestamp)) ,
DATE(MIN(order_purchase_timestamp)), MONTH) -
DATE_DIFF(DATE(MAX(order_purchase_timestamp)),
DATE(MIN(order_purchase_timestamp)), YEAR) * 12 AS range_orders_month,
  (DATE_DIFF(DATE(MAX(order_purchase_timestamp)),
DATE(MIN(order_purchase_timestamp)), WEEK)/4) -
DATE_DIFF(DATE(MAX(order_purchase_timestamp)),
DATE(MIN(order_purchase_timestamp)), MONTH) AS range_orders_week
FROM `Target.orders`
)
SELECT
cte.first_timestamp,
cte.earliest_timestamp as latest_timestamp,
CONCAT(cte.range_orders_year," years ",cte.range_orders_month," months
",cte.range_orders_week," weeks ") as total_range
FROM cte
```

Output:

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

Row	first_timestamp	latest_timestamp	total_range
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC	2 years 1 months 2.5 weeks

Finding outliers for range:

```
with range_orders as (
  SELECT
    min(order_purchase_timestamp) as first_timestamp,
    max(order_purchase_timestamp) as earliest_timestamp,
    date_diff(date(max(order_purchase_timestamp)),date(min(order_purchase_timest
amp)), week)as range_orders_week
FROM `Target.orders`
),
weeks_difference as(
SELECT
  order_purchase_timestamp,
 date_diff(date(range_orders.earliest_timestamp),date(order_purchase_timestamp),
week) as week_diff
```

```
 from `Target.orders`
 cross join range_orders
 order by order_purchase_timestamp
),quartile as(
select
weeks_difference.week_diff,
ntile(4) over(order by weeks_difference.week_diff desc) as quartile_number
from weeks_difference
),lower_upper_fence as(

SELECT
MAX(case when quartile_number=2 then quartile.week_diff end ) as Q1,
MIN(Case when quartile_number=3 then quartile.week_diff end) as Q3
from quartile
)
select
weeks_difference.order_purchase_timestamp as minor_outliers,
NUll as major_outliers,
Q1-Q3 as interquartile_range
from weeks_difference
cross join lower_upper_fence
WHERE week_diff > Q1 + 1.5 * (Q1 - Q3)
   OR week_diff < Q3 - 1.5 * (Q1 - Q3)
UNION ALL
select
  Null as minor_outliers,
weeks_difference.order_purchase_timestamp as major_outliers,
 Q1-Q3 as interquartile_range
from weeks_difference
cross join lower_upper_fence

WHERE week_diff > Q1 + (1.5 * 1.5* (Q1 - Q3))
   OR week_diff < Q3 - (1.5 * 1.5 * (Q1 - Q3));
```

Output:

Row	minor_outliers	major_outliers	interquartile_range
1	2016-09-04 21:15:19 UTC	*null*	33
2	2016-09-05 00:15:34 UTC	*null*	33
3	2016-09-13 15:24:19 UTC	*null*	33
4	2016-09-15 12:16:38 UTC	*null*	33

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

Query:

```
with c_s as(

SELECT
distinct seller_city as city,
 seller_state as state
FROM `Target.sellers`
UNION ALL
SELECT
distinct geolocation_city as city,
 geolocation_state as state
FROM `my-project-1-387008.Target.geolocation`
UNION ALL
SELECT
distinct customer_city as city,
 customer_state as state
FROM `Target.customers`
)
SELECT
COUNT(Distinct city) as total_city,
COUNT(Distinct state) as total_state
FROM c_s
```

Output:

Row	total_city	total_state
1	8126	27

Potential for covering cities

Query:

```
with geo as(SELECT
count(distinct geolocation_city) as geo_cities_count,
count(distinct geolocation_state) as geo_states_count
FROM `my-project-1-387008.Target.geolocation`
),customer as(
SELECT
count(distinct customer_city) as customer_cities_count,
count(distinct customer_state) as customer_states_count
FROM `Target.customers`
),seller as(
SELECT
count(distinct seller_city) as sellers_cities_count,
count(distinct seller_state) as sellers_states_count
FROM `Target.sellers`
```

```
)
SELECT
*
FROM seller
cross join customer
cross join geo
```

Output:

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

Row	sellers_cities_count	sellers_states_count	customer_cities_cou	customer_states_cou	geo_cities_count	geo_states_count
1	611	23	4119	27	8011	27

City wise order count of customers:

Query:

```
SELECT count(distinct o.order_id) as count_orders,
c.customer_city,c.customer_state
FROM `my-project-1-387008.Target.orders` o
join `Target.customers` c
on c.customer_id=o.customer_id
group by 2,3
order by 1 desc
```

Output:

Row	count_orders	customer_city	customer_state
1	15540	sao paulo	SP
2	6882	rio de janeiro	RJ
3	2773	belo horizonte	MG
4	2131	brasilia	DF
5	1521	curitiba	PR
6	1444	campinas	SP
7	1379	porto alegre	RS
8	1245	salvador	BA
9	1189	guarulhos	SP
10	938	sao bernardo do campo	SP
11	849	niteroi	RJ

2. In-depth Exploration:

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

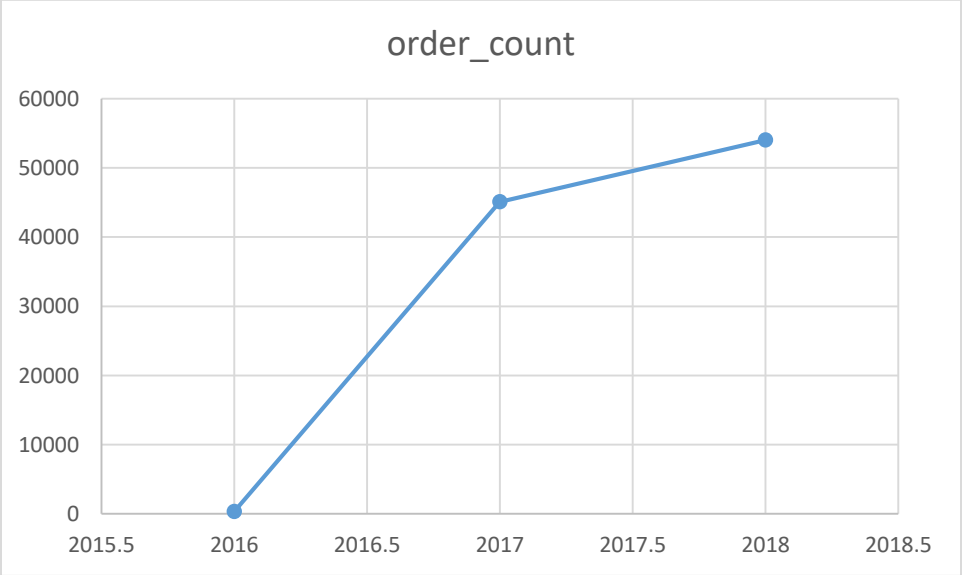
Query:

```
SELECT
  EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
  COUNT(*) AS order_count
FROM `Target.orders`
GROUP BY year
ORDER BY year
```

Output:

Row	year	order_count
1	2016	329
2	2017	45101
3	2018	54011

Chart:



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

Query:

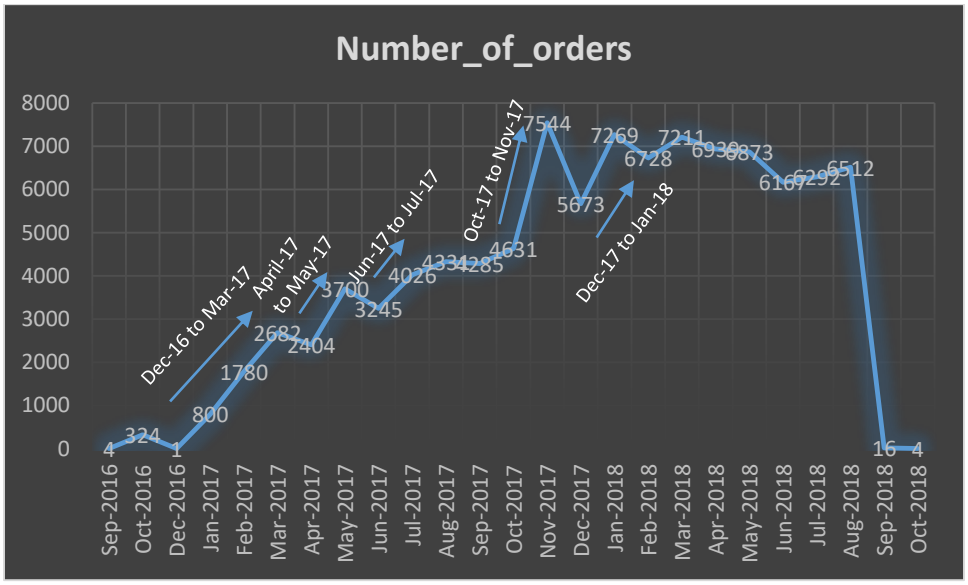
```
with count_orders as (
SELECT  count(*) as number_of_orders,
extract(month from order_purchase_timestamp) as month,
extract(year from order_purchase_timestamp) as year
FROM `my-project-1-387008.Target.orders`
group by extract(month from order_purchase_timestamp),
extract(year from order_purchase_timestamp)
), mon as(
SELECT
count_orders.number_of_orders,
month,
  CASE
    WHEN  month= 1 THEN 'Jan'
    WHEN month = 2 THEN 'Feb'
    WHEN month = 3 THEN 'Mar'
    WHEN month = 4 THEN 'Apr'
    WHEN month = 5 THEN 'May'
    WHEN month = 6 THEN 'Jun'
    WHEN month = 7 THEN 'Jul'
    WHEN month = 8 THEN 'Aug'
    WHEN month = 9 THEN 'Sep'
    WHEN month = 10 THEN 'Oct'
    WHEN month = 11 THEN 'Nov'
    WHEN month= 12 THEN 'Dec'
    ELSE 'Unknown'
  END AS months,
  year
FROM count_orders
)
```

```
select
mon.number_of_orders,
months,
year
FROM mon
order by year,mon.month
```

Output:

Row	number_of_orders	months	year
1	4	Sep	2016
2	324	Oct	2016
3	1	Dec	2016
4	800	Jan	2017
5	1780	Feb	2017
6	2682	Mar	2017
7	2404	Apr	2017
8	3700	May	2017
9	3245	Jun	2017
10	4026	Jul	2017
11	4331	Aug	2017

Chart: for showing growing trend



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**

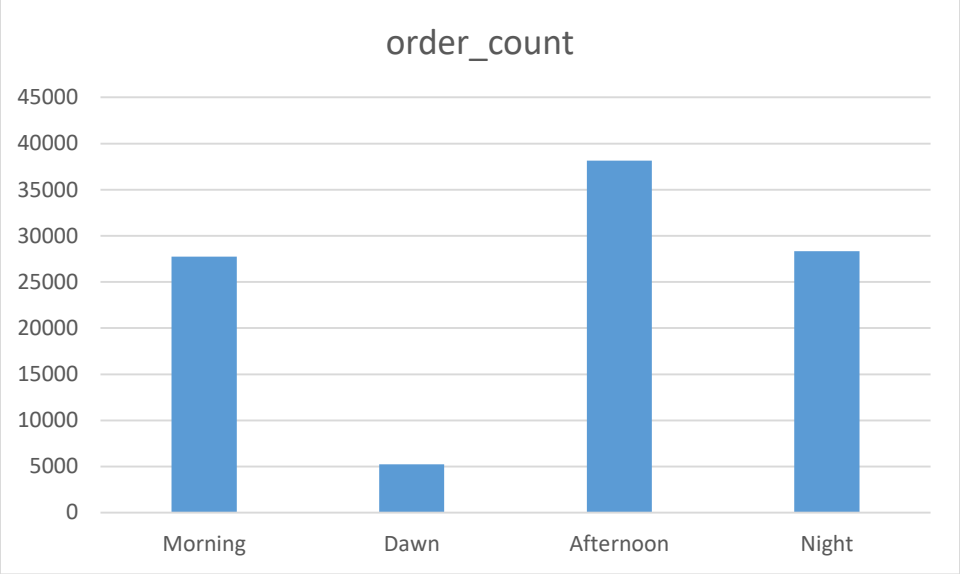
   Query:

```
SELECT
  CASE
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 0 AND 6 THEN
'Dawn'
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 7 AND 12 THEN
'Morning'
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 13 AND 18 THEN
'Afternoon'
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 19 AND 23 THEN
'Night'
    ELSE 'Unknown'
  END AS time_of_day,
  COUNT(*) AS order_count
FROM `Target.orders`
GROUP BY time_of_day
```

   Output:

Row	time_of_day	order_count
1	Morning	27733
2	Dawn	5242
3	Afternoon	38135
4	Night	28331

Chart:



Count of orders between hours

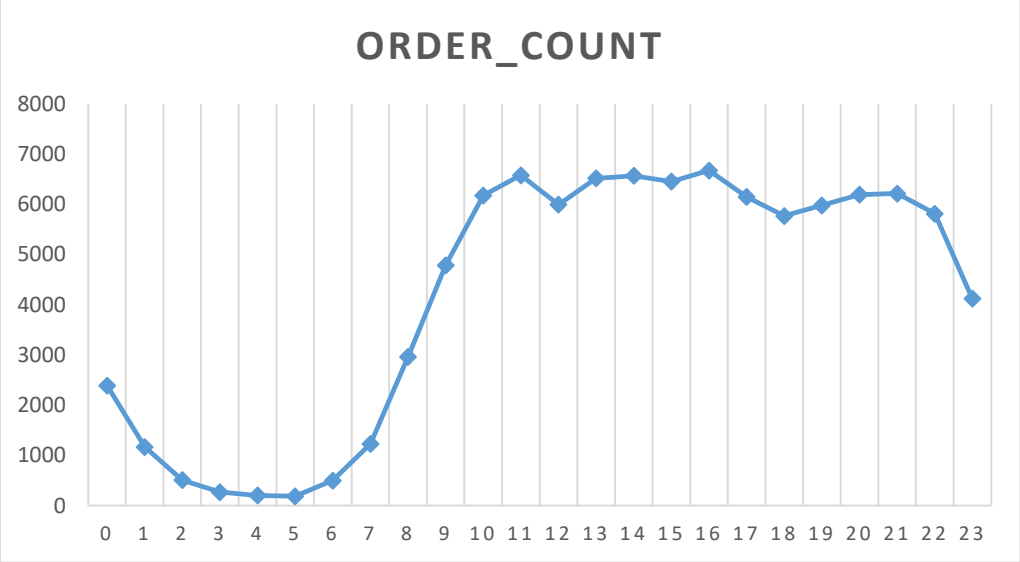
Query:

```
SELECT
EXTRACT(HOUR FROM order_purchase_timestamp) as range_hour,
count(*) as order_count
from `Target.orders`
group by 1
order by 1;
```

Output:

Row	range_hour	order_count
1	0	2394
2	1	1170
3	2	510
4	3	272
5	4	206
6	5	188
7	6	502
8	7	1231
9	8	2967

Chart:



After 10 am there is somewhat stagnant orders

3. Evolution of E-commerce orders in the Brazil region:

1. Get the month on month no. of orders placed in each state.

Query:

```
SELECT
  EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
  EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
  customer_state as state,
  COUNT(*) AS order_count
FROM `Target.orders` o
join `Target.customers` c
on c.customer_id=o.customer_id
GROUP BY year, month, state
ORDER BY year, month, state
```

Output:

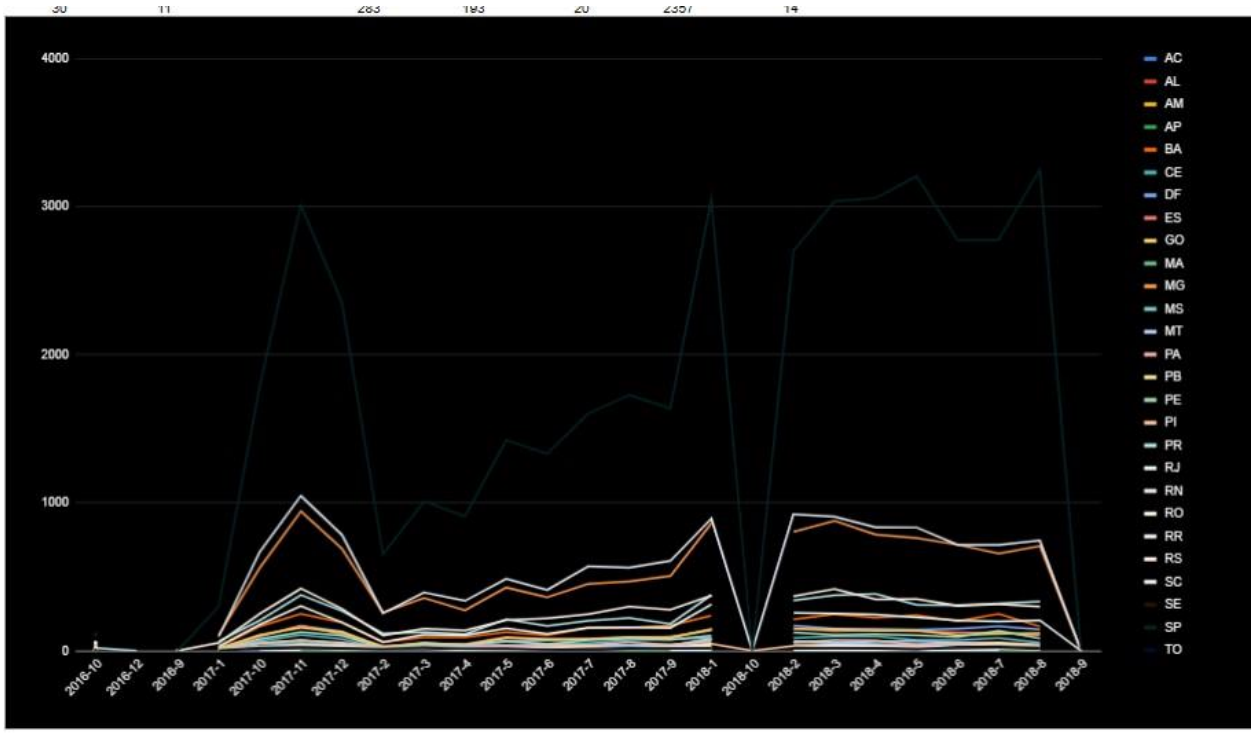
Row	year	month	state	order_count
1	2016	9	RR	1
2	2016	9	RS	1
3	2016	9	SP	2
4	2016	10	AL	2
5	2016	10	BA	4
6	2016	10	CE	8
7	2016	10	DF	6
8	2016	10	ES	4
9	2016	10	GO	9
10	2016	10	MA	4
11	2016	10	MG	40

Pivot table for same

year-month	AC	AL	AM	AP	BA	CE	DF	ES	GO	MA	MG	MS	MT	PA	PB	PE	PI	PR	RJ	RN	RO	RR	RS	SC	SE	SP	TO	Grand Total
2016-10		2			4	8	6	4	9	4	40		3	4	1	7	1	19	56	4		1	24	11	3	113		324
2016-12																		1										1
2016-9																						1	1			2		4
2017-1	2	2			25	9	13	12	18	9	108	1	11	12	2	9	7	65	97	5	3		54	31	4	299	2	800

2017-10	6	28	3	3	166	66	98	100	108	48	560	34	52	54	30	80	23	206	668	23	14	3	252	178	22	1793	13	4631
2017-11	5	26	10	4	250	108	168	170	157	56	943	46	74	70	30	126	31	378	1048	44	17	2	422	303	27	3012	17	7544
2017-12	5	14	6	4	192	81	131	113	127	41	691	36	50	58	37	103	23	270	783	30	11		283	193	20	2357	14	5673
2017-2	3	12	8	2	59	13	24	34	27	11	259	11	17	25	12	21	12	118	254	8	11	2	105	59	12	654	7	1780
2017-3	2	10	5	3	91	28	57	48	53	24	358	20	16	36	16	45	13	127	395	13	16	2	151	110	25	1010	8	2682
2017-4	5	23	13		93	43	35	46	41	27	275	15	27	36	20	40	13	114	338	10	9	2	139	105	13	908	14	2404
2017-5	8	27	10	5	127	62	64	94	87	33	428	29	37	35	18	68	25	213	488	17	9	2	208	152	11	1425	18	3700
2017-6	4	10	1	2	106	47	70	80	79	17	363	27	25	38	23	46	14	170	412	13	10	3	221	116	9	1331	8	3245
2017-7	5	17	5	1	155	53	77	83	77	39	453	25	38	39	27	73	20	203	571	27	11	1	249	158	14	1604	1	4026
2017-8	4	18	5	3	158	73	87	95	93	40	469	24	38	60	16	85	22	223	562	20	14		299	159	20	1729	15	4331
2017-9	5	20	9	2	170	77	97	93	88	42	507	33	35	41	29	76	23	183	609	24	16	1	278	156	16	1638	17	4285
2017-1	6	37	12	11	239	90	138	147	146	57	863	70	85	70	31	104	48	378	893	46	20	2	373	314	20	3052	17	7269
2017-10																	1		1							2		4
2017-2	3	27	8	2	214	88	172	152	149	56	804	64	67	58	35	125	34	342	922	23	14	5	368	257	15	2703	21	6728
2017-3	2	30	9	5	249	98	150	134	146	53	879	59	55	73	39	108	35	377	907	39	13	6	418	252	18	3037	20	7211
2017-4	4	28	6	5	225	100	148	142	136	46	786	43	65	71	31	114	37	386	834	32	11	2	349	246	14	3059	19	6939
2017-5	2	19	9	6	241	74	144	134	139	32	762	45	67	40	29	106	31	311	833	22	17	1	351	227	8	3207	16	6873
2017-6	3	24	7	2	201	74	150	124	105	42	717	49	58	54	28	94	29	308	716	36	12	5	305	205	28	2773	18	6167
2017-7	4	23	18	6	250	87	166	123	115	40	658	49	47	57	52	137	32	320	717	29	16	5	316	198	28	2777	22	6292
2017-8	3	16	4	2	165	57	145	105	120	30	708	35	40	44	30	85	21	333	745	20	9		300	206	23	3253	13	6512
2017-9											4								3					1		8		16
Grand Total	**81**	**413**	**148**	**68**	**3380**	**1336**	**2140**	**2033**	**2020**	**747**	**1163 5**	**715**	**907**	**975**	**536**	**1652**	**495**	**5045**	**1285 2**	**485**	**253**	**46**	**5466**	**3637**	**350**	**4174 6**	**280**	**9944 1**

Chart:



Change the order of states in Chart:

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

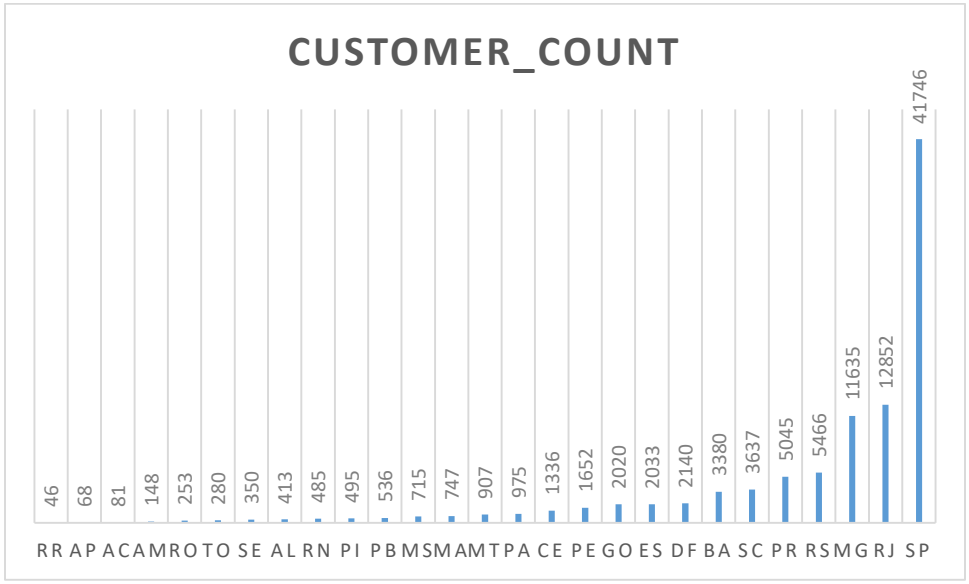
Query:

```
SELECT customer_state, COUNT(*) AS customer_count
FROM `Target.customers`
GROUP BY customer_state
order by 2 DESC;
```

Output:

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

Chart:



4. **Impact on Economy: Analyze 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.

       Query:

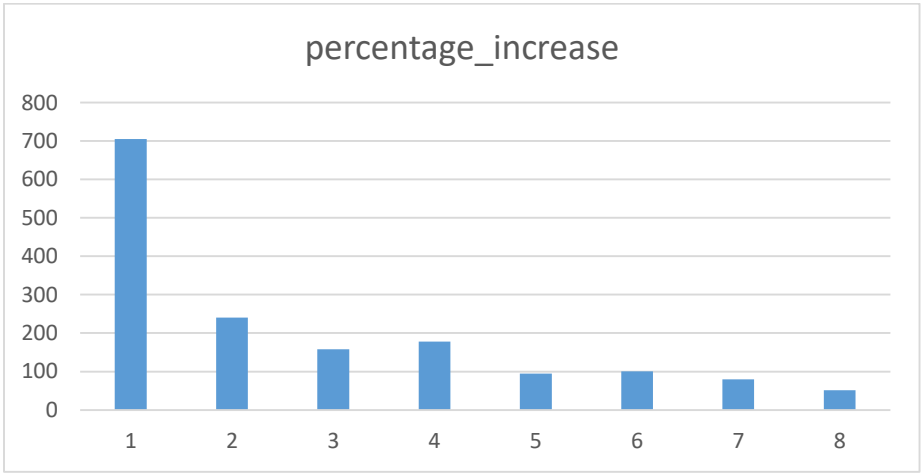
```
WITH order_costs AS (
  SELECT
    EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
    EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
    SUM(p.payment_value) AS total_cost
  FROM
    `Target.orders` o
  INNER JOIN
    `Target.payments` p ON o.order_id = p.order_id
  WHERE
    EXTRACT(YEAR FROM o.order_purchase_timestamp) IN (2017, 2018)
    AND EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8
  GROUP BY
    year, month
),
previous_year_costs AS (
  SELECT
    month,
    round(total_cost,2) AS previous_year_cost
  FROM
    order_costs
  WHERE
    year = 2017
),
current_year_costs AS (
  SELECT
    month,
    round(total_cost,2) AS current_year_cost
  FROM
    order_costs
  WHERE
    year = 2018
)
SELECT
  current_year_costs.month,
  current_year_costs.current_year_cost,
  previous_year_costs.previous_year_cost,
  round(((current_year_costs.current_year_cost -
previous_year_costs.previous_year_cost) /
previous_year_costs.previous_year_cost) * 100,2) AS percentage_increase
FROM
```

```
  current_year_costs
JOIN
  previous_year_costs ON current_year_costs.month = previous_year_costs.month
order by 1;
```

Output:

Row	month	current_year_cost	previous_year_cost	percentage_increase
1	1	1115004.18	138488.04	705.13
2	2	992463.34	291908.01	239.99
3	3	1159652.12	449863.6	157.78
4	4	1160785.48	417788.03	177.84
5	5	1153982.15	592918.82	94.63
6	6	1023880.5	511276.38	100.26
7	7	1066540.75	592382.92	80.04
8	8	1022425.32	674396.32	51.61

Chart:



Yearly

Query:

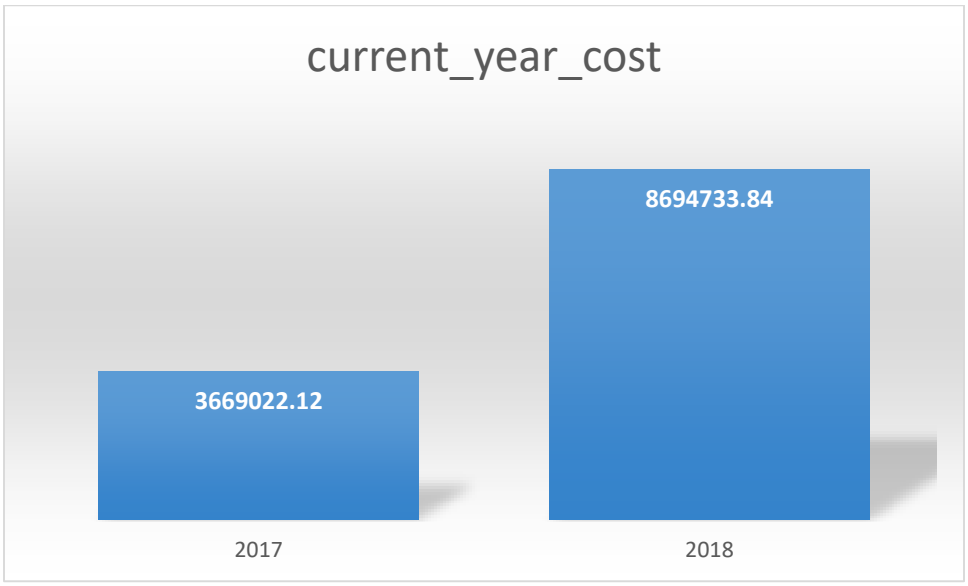
```
WITH order_costs AS (
  SELECT
    EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
    SUM(p.payment_value) AS total_cost
  FROM
    `Target.orders` o
  INNER JOIN
    `Target.payments` p ON o.order_id = p.order_id
  WHERE
    EXTRACT(YEAR FROM o.order_purchase_timestamp) IN (2017, 2018) AND
    EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 and 8
  GROUP BY
    year
),
previous_year_costs AS (
  SELECT
    order_costs.YEAR,
    round(total_cost,2) AS previous_year_cost
  FROM
    order_costs
  WHERE
    year = 2017
),
current_year_costs AS (
  SELECT
    order_costs.YEAR,
    round(total_cost,2) AS current_year_cost
  FROM
    order_costs
  WHERE
    year = 2018
)
SELECT
  current_year_costs.YEAR,
  current_year_costs.current_year_cost,
  previous_year_costs.YEAR,
  previous_year_costs.previous_year_cost,
  round(((current_year_costs.current_year_cost -
previous_year_costs.previous_year_cost) /
previous_year_costs.previous_year_cost) * 100,2) AS percentage_increase
FROM
  current_year_costs
cross join previous_year_costs
order by 1;
```

Output:

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

Row	YEAR	current_year_cost	YEAR_1	previous_year_cost	percentage_increase
1	2018	8694733.84	2017	3669022.12	136.98

Chart:



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

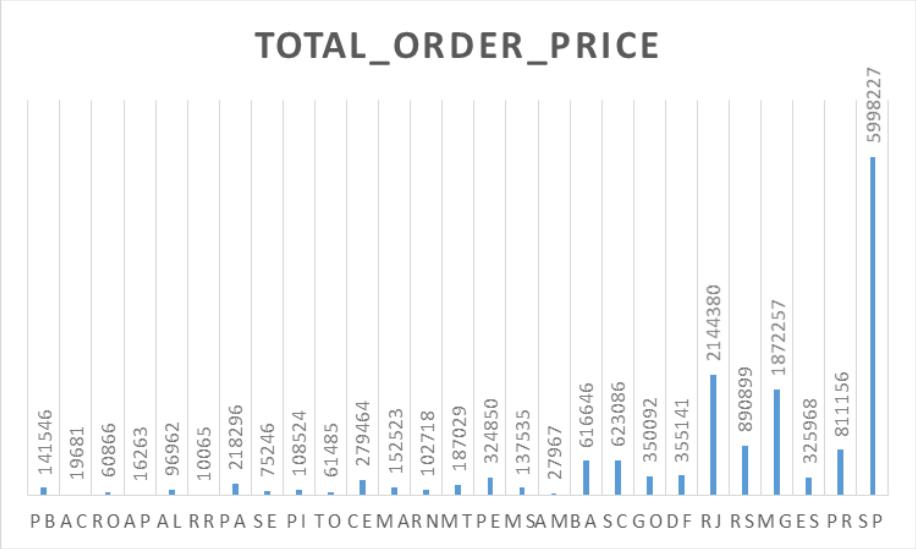
Query:

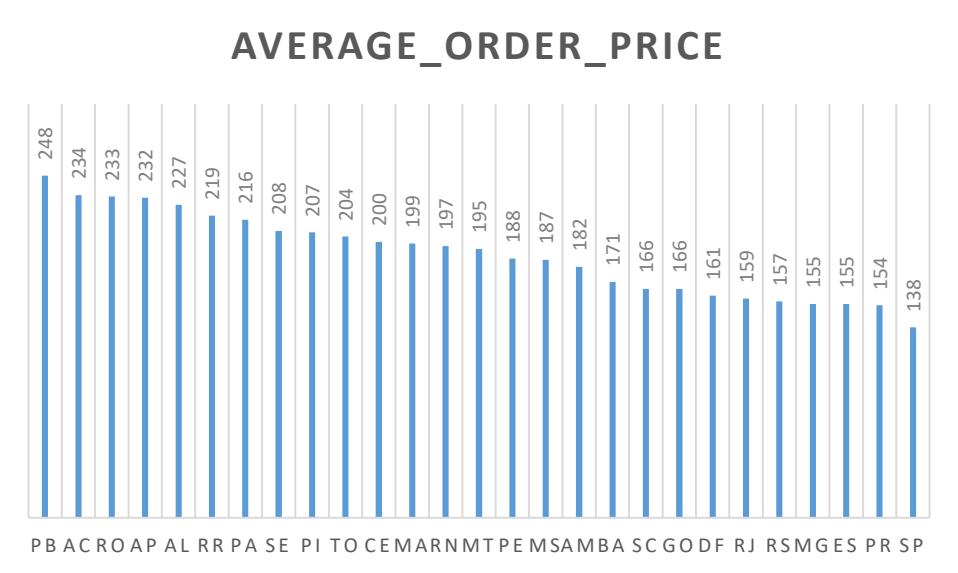
```
SELECT
  c.customer_state AS state,
  Round(SUM(p.payment_value)) AS total_order_price,
  Round(AVG(p.payment_value)) AS average_order_price
FROM
  `Target.orders` o
Join `Target.customers` c
on c.customer_id=o.customer_id
INNER JOIN `Target.payments` p
ON o.order_id = p.order_id
GROUP BY state
ORDER BY 3 DESC;
```

Output:

Row	state	total_order_price	average_order_price
1	PB	141546.0	248.0
2	AC	19681.0	234.0
3	RO	60866.0	233.0
4	AP	16263.0	232.0
5	AL	96962.0	227.0
6	RR	10065.0	219.0
7	PA	218296.0	216.0
8	SE	75246.0	208.0
9	PI	108524.0	207.0

Chart:


TOTAL_ORDER_PRICE
141546
19681
60866
16263
96962
10065
218296
75246
108524
61485
279464
152523
102718
187029
324850
137535
27967
616646
623086
350092
355141
2144380
890899
1872257
325968
811156
5998227
PB AC RO AP AL RR PA SE PI TO CE MA RN MT PE MS AM BA SC GO DF RJ RS MG ES PR SP


AVERAGE_ORDER_PRICE
248
234
233
232
227
219
216
208
207
204
200
199
197
195
188
187
182
171
166
166
161
159
157
155
155
154
138
PB AC RO AP AL RR PA SE PI TO CE MA RN MT PE MS AM BA SC GO DF RJ RS MG ES PR SP

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

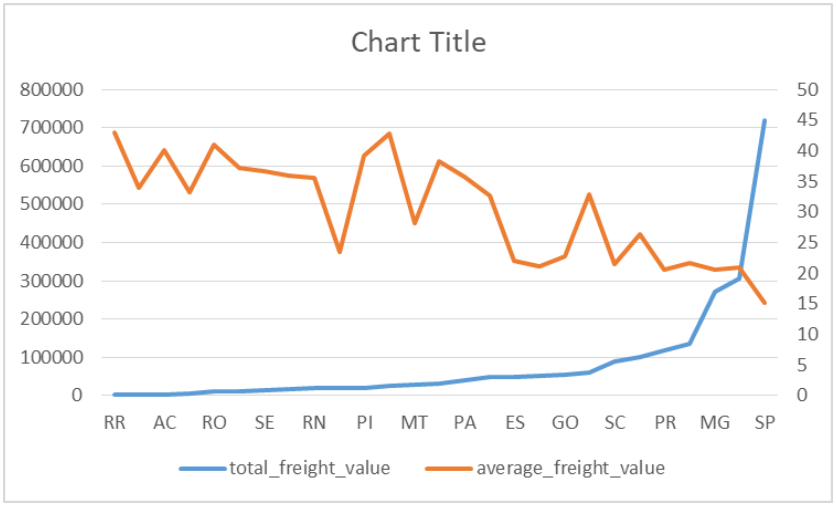
Query:

```
SELECT
  customer_state,
  SUM(freight_value) AS total_freight_value,
  AVG(freight_value) AS average_freight_value
FROM
  `Target.order_items` oi
  join `Target.orders` o
  on o.order_id=oi.order_id
  join `Target.customers` c
  on c.customer_id=o.customer_id
GROUP BY
  customer_state
ORDER BY customer_state;
```

Output:

Row	customer_state	total_freight_value	average_freight_valu
1	AC	3686.750000000...	40.07336956521...
2	AL	15914.58999999...	35.84367117117...
3	AM	5478.890000000...	33.20539393939...
4	AP	2788.500000000...	34.00609756097...
5	BA	100156.6799999...	26.36395893656...
6	CE	48351.58999999...	32.71420162381...
7	DF	50625.49999999...	21.04135494596...
8	ES	49764.59999999...	22.05877659574...
9	GO	53114.97999999...	22.76681525932...

Chart:



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**

       Query:

```
with cte as
(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`
)
SELECT
*
FROM cte
WHERE cte.time_to_deliver is not NULL and cte.diff_estimated_delivery is not
NULL
ORDER BY time_to_deliver DESC,diff_estimated_delivery
```

Output:

Row	order_id	time_to_deliver	diff_estimated_delive
1	ca07593549f1816d26a572e06...	209	-181
2	1b3190b2dfa9d789e1f14c05b...	208	-188
3	440d0d17af552815d15a9e41a...	195	-165
4	285ab9426d6982034523a855f...	194	-166
5	0f4519c5f1c541ddec9f21b3bd...	194	-161
6	2fb597c2f772eca01b1f5c561b...	194	-155
7	47b40429ed8cce3aee9199792...	191	-175
8	2fe324febf907e3ea3f2aa9650...	189	-167
9	2d7561026d542c8dbd8f0daea...	188	-159

Count of orders based on delivery time

Query:

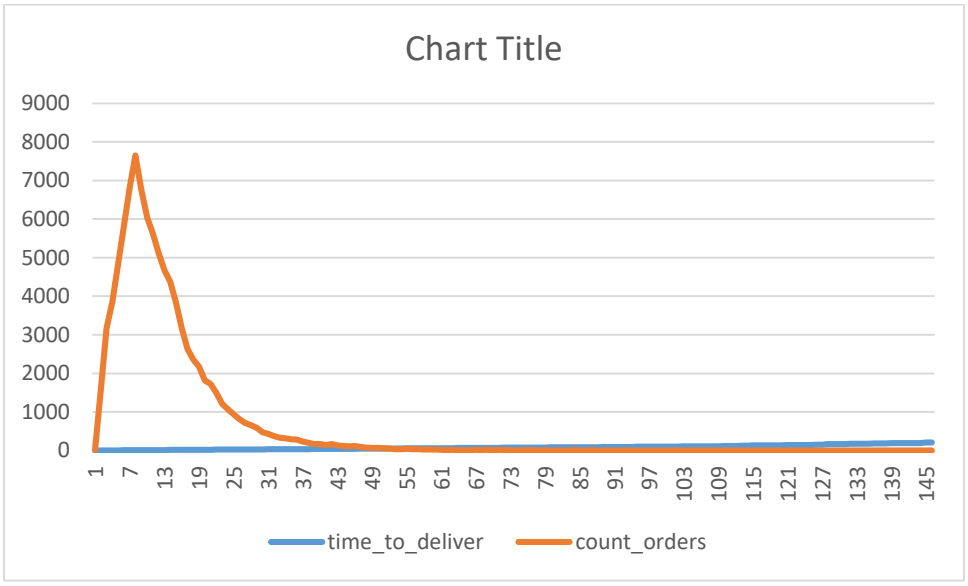
```
with cte as

(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`
)
SELECT
cte.time_to_deliver,
count(distinct order_id) as count_orders
FROM cte
WHERE cte.time_to_deliver is not NULL and cte.diff_estimated_delivery is not
NULL
GROUP BY cte.time_to_deliver
ORDER BY count_orders DESC
```

Output:

Row	time_to_deliver	count_orders
1	7	7653
2	6	6805
3	8	6745
4	9	6039
5	5	5810
6	10	5616
7	11	5114
8	4	4828
9	12	4673

Chart:



Most orders are delivered in 25 days peak within 7 days

Trend for difference in expected delivery vs delivery date

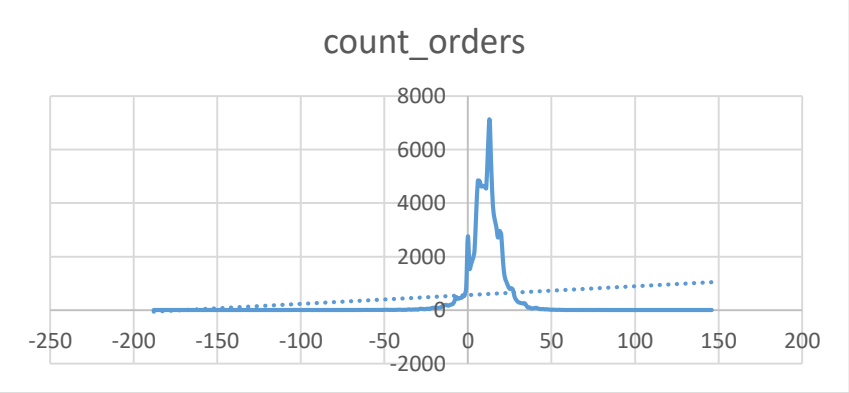
Query:

```
with cte as
(SELECT
  order_id,
  DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date, DAY)
AS diff_estimated_delivery
FROM
  `Target.orders`
)
SELECT
cte.diff_estimated_delivery ,
count(distinct order_id) as count_orders
FROM cte
WHERE cte.diff_estimated_delivery is not NULL
GROUP BY cte.diff_estimated_delivery
ORDER BY count_orders DESC
```

Output:

Row	diff_estimated_delive	count_orders
1	13	7126
2	12	5963
3	14	5345
4	6	4837
5	7	4828
6	9	4646
7	8	4626
8	10	4619
9	11	4556

Chart:



Most of the orders are delivered 7 days after or 25 days before expected date

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

Query:

```
with Highest as(
SELECT
  state,
  avg_freight_value,
  rank
FROM
  (SELECT
   customer_state as state,
    round(AVG(freight_value),2) AS avg_freight_value,
    DENSE_RANK() OVER (ORDER BY AVG(freight_value) DESC) AS rank
  FROM
    `Target.orders` o
  join `Target.customers` c
  on o.customer_id=c.customer_id
  join `Target.order_items` oi
  on oi.order_id=o.order_id
  GROUP BY
    state
    ORDER BY rank
    ) AS ranked_states
WHERE
  rank <= 5

  ),Lowest as(
SELECT
  state,
  avg_freight_value,
  rank
FROM
  (SELECT
 customer_state as state,
    round(AVG(freight_value),2) AS avg_freight_value,
    DENSE_RANK() OVER (ORDER BY AVG(freight_value) ASC) AS rank
  FROM
    `Target.orders` o
    join `Target.customers` c
  on o.customer_id=c.customer_id
  join `Target.order_items` oi
  on oi.order_id=o.order_id
  GROUP BY
    state
    ORDER BY rank) AS ranked_states
WHERE
  rank <= 5

  )
  SELECT
   h.state as highest_freight_value_state,
```

```
  h.avg_freight_value as highest_avg_freight_value,
  l.state as lowest_freight_value_state,
  l.avg_freight_value as lowest_avg_freight_value
  FROM Highest h
   join Lowest l
  on h.rank=l.rank
  ORDER BY h.rank
```

Output:

Row	highest_freight_value_state	highest_avg_freight_	lowest_freight_value_state	lowest_avg_freight_y
1	RR	42.98	SP	15.15
2	PB	42.72	PR	20.53
3	RO	41.07	MG	20.63
4	AC	40.07	RJ	20.96
5	PI	39.15	DF	21.04

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

Query:

```
with highest as(
  SELECT
   state,
  avg_delivery_time,
  rank
FROM
  (SELECT
    customer_state as state,
    AVG(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY))
AS avg_delivery_time,
    DENSE_RANK() OVER (ORDER BY AVG(DATE_DIFF(order_delivered_customer_date,
order_purchase_timestamp, DAY)) DESC) AS rank
  FROM
    `Target.orders`o
    join `Target.customers` c
  on o.customer_id=c.customer_id
  GROUP BY
    state) AS ranked_states
WHERE
  rank <= 5
),lowest as(
SELECT
  state,
  avg_delivery_time,
  rank
FROM
  (SELECT
    customer_state as state,
```

```
    AVG(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, day))
AS avg_delivery_time,
    DENSE_RANK() OVER (ORDER BY AVG(DATE_DIFF(order_delivered_customer_date,
order_purchase_timestamp, DAY)) ASC) AS rank
  FROM
    `Target.orders`o
    join `Target.customers` c
  on o.customer_id=c.customer_id
  GROUP BY
    state) AS ranked_states
WHERE
  rank <= 5
  )

  SELECT
   h.state as highest_delivery_time_state,
  h.avg_delivery_time as highest_delivery_time,
  l.state as lowest_delivery_time_state,
  l.avg_delivery_time as lowest_delivery_time
  FROM highest h
   join lowest l
  on h.rank=l.rank
  ORDER BY h.rank
```

Output:

Row	highest_delivery_time_state	highest_delivery_tim	lowest_delivery_time_state	lowest_delivery_time
1	RR	28.97560975609…	SP	8.298061489072…
2	AP	26.73134328358…	PR	11.52671135486…
3	AM	25.98620689655…	MG	11.54381329810…
4	AL	24.04030226700…	DF	12.50913461538…
5	PA	23.31606765327…	SC	14.47956019171…

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.

   Query:

```
SELECT
  state,
  avg_delivery_time_difference
FROM
  (SELECT
    customer_state as state,
    AVG(DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date,
DAY)) AS avg_delivery_time_difference,
    DENSE_RANK() OVER (ORDER BY AVG(DATE_DIFF(order_estimated_delivery_date,
order_delivered_customer_date, DAY)) DESC) AS rank
  FROM
    `Target.orders` o
     join `Target.customers` c
  on o.customer_id=c.customer_id

  GROUP BY
    state) AS ranked_states
WHERE
  rank <= 5
ORDER BY ranked_states.avg_delivery_time_difference DESC
```

   Output:

Row	state	avg_delivery_time_di
1	AC	19.76250000000...
2	RO	19.13168724279...
3	AP	18.73134328358...
4	AM	18.60689655172...
5	RR	16.41463414634...

6. Analysis based on the payments:

1. Find the month on month no. of orders placed using different payment types.

- **Lets check for duplicates order_id with multiple payment_type**

Query:

```
with cte as(
SELECT
SUM(order_count) as total_payment_type_for_total_order_id
FROM (SELECT
  payment_type,
  COUNT(distinct order_id) as order_count
  FROM `Target.payments`
  GROUP BY payment_type)),
order_id_count_from_orders as (SELECT
COUNT(distinct order_id) as total_order_id
FROM `Target.orders`
)
SELECT
*,
c.total_payment_type_for_total_order_id-
order_id_count_from_orders.total_order_id as
number_of_order_id_with_multiple_payment_type
FROM cte c
CROSS JOIN order_id_count_from_orders
```

Output:

Row	total_payment_type_1	total_order_id	number_of_order_id
1	101686	99441	2245

- As from above there's 2245 order_id with multiple payment_type but we need all the entries in different payment type as payment is made using different payment types

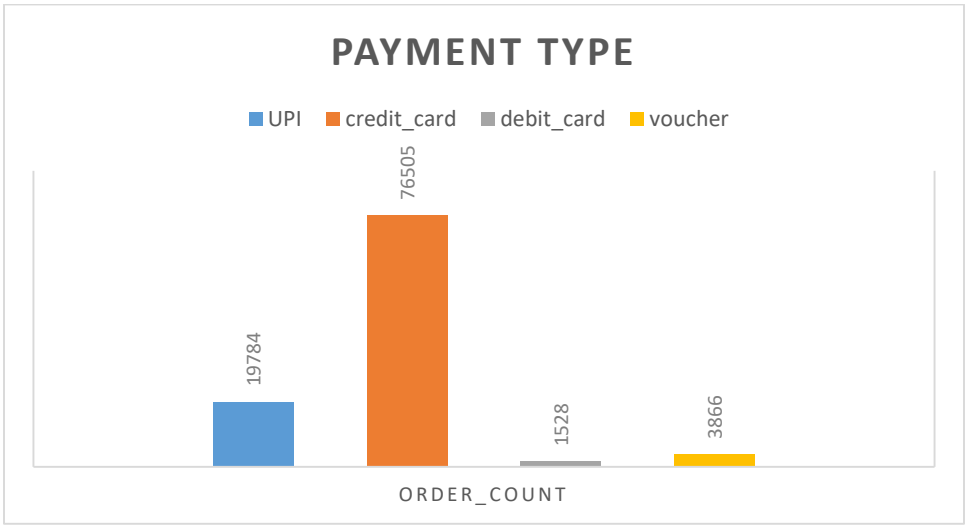
Query for finding the month on month no. of orders placed using different payment types :

```
SELECT
  EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
  EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
  payment_type,
  COUNT(DISTINCT o.order_id ) AS order_count
FROM
  `Target.orders` o
  join `Target.payments` p
  on o.order_id=p.order_id
GROUP BY
  year,month,
  payment_type
ORDER BY
  year,month,payment_type;
```

Output:

Row	year	month	payment_type	order_count
1	2016	9	credit_card	3
2	2016	10	UPI	63
3	2016	10	credit_card	253
4	2016	10	debit_card	2
5	2016	10	voucher	11
6	2016	12	credit_card	1
7	2017	1	UPI	197
8	2017	1	credit_card	582
9	2017	1	debit_card	9

Chart:

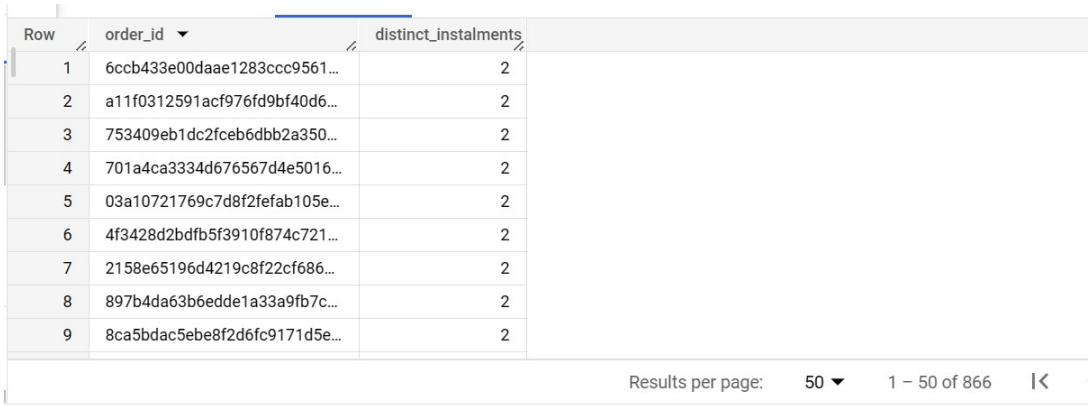


2. **Find the no. of orders placed on the basis of the payment installments that have been paid.**
    - **First find if one order_id have multiple payment_installments**

Query:

```
SELECT
  order_id,
  COUNT(DISTINCT payment_installments) AS distinct_instalments
FROM
  `Target.payments`
GROUP BY
  order_id
HAVING
  COUNT(DISTINCT payment_installments) > 1;
```

Output:



Row	order_id	distinct_instalments
1	6ccb433e00daae1283ccc9561...	2
2	a11f0312591acf976fd9bf40d6...	2
3	753409eb1dc2fceb6dbb2a350...	2
4	701a4ca3334d676567d4e5016...	2
5	03a10721769c7d8f2fefab105e...	2
6	4f3428d2bdfb5f3910f874c721...	2
7	2158e65196d4219c8f22cf686...	2
8	897b4da63b6edde1a33a9fb7c...	2
9	8ca5bdac5ebe8f2d6fc9171d5e...	2

Results per page: 50 1 – 50 of 866

There's 866 cases with same order_id with multiple payment installements

- **Taking max installment number and then counting the number of orders**

Query:

```
with cte as(

  SELECT
    order_id,
    MAX(payment_installments) AS payment_installments
  FROM
    `Target.payments`
  WHERE
    payment_installments > 0
```

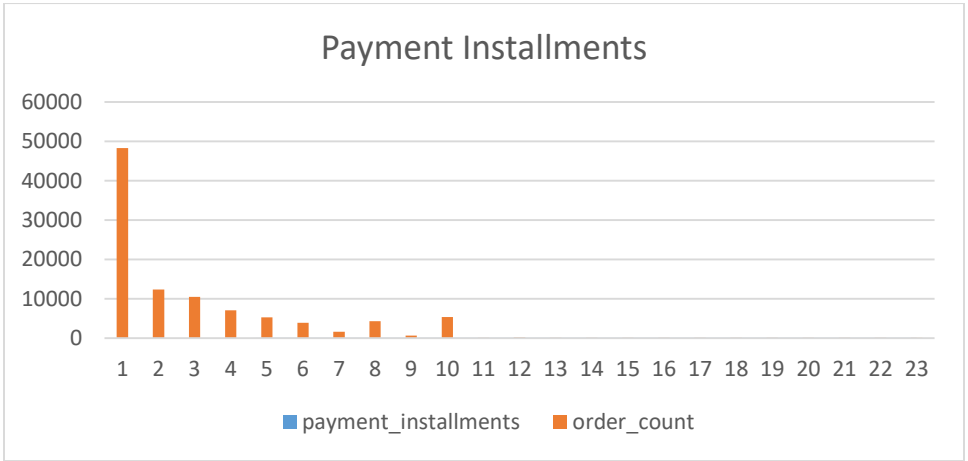
```
  GROUP BY
    order_id

)
SELECT
cte.payment_installments,
COUNT(distinct cte.order_id) as order_count
FROM cte
group by 1
order by 1
```

Output:

Row	payment_installment	order_count
1	1	48268
2	2	12363
3	3	10429
4	4	7070
5	5	5227
6	6	3908
7	7	1622
8	8	4251
9	9	644

Chart:



Most of the payments are made in 1 installment

Insights:

- Most of the payments are made in 1 installment
- Max orders are placed by CreditCard and UPI
- Most of the Orders are delivered between expected date
- Most orders are delivered in between 7 days to 25 days
- Average order value is between 138 to 248
- If we compare first 8 months of 2017 and 2018 there's 136% increase in revenue
- SP,RJ and MG have maximum number of orders placed
- After 10 AM there's stagnant customer purchase till 11 PM
- Maximum orders placed in Afternoon time
- August ,September are the months where Target is facing challenges other months are seeing growing trend

Recommendations

- There's low purchase in Dawn and after 11 PM Target can start night shift to fill the gap
- As most of the payments are happening in 1 installments Target need to provide better solution for payment installments and people are not seems to accepting it
- Aug and Sept needs to focused with special price and product offerings seeing cold and dry weather condition