SQL Case study 1
Business Case: Target SQL

https://drive.google.com/drive/folders/1TGEc66YKbD443nslRi1bWgVd238gJCnb

The data is available in 8 csv files:

1. customers.csv
2. sellers.csv
3. order_items.csv
4. geolocation.csv
5. payments.csv
6. reviews.csv
7. orders.csv
8. products.csv

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

Query:

```
#Get the time range between which the orders were placed.

select min(order_purchase_timestamp) Minimum_time_range,
max(order_purchase_timestamp) Maximum_time_range,
from
`SQL.orders`;
```

Query results

SAVE RESULTS

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

Row	Minimum_time_range	Maximum_time_range
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

2.Count the Cities & States of customers who ordered during the given period.

```
#Count the Cities & States of customers who ordered during the given period.
```

```sql
select
  count(distinct c.customer_city) customer_city,
  count(distinct c.customer_state) customer_state
from `SQL.customers`  c
inner join `SQL.orders` o on c.customer_id = o.customer_id;
```

Query results

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

Row	customer_city	customer_state
1	4119	27

2.In-depth Exploration:

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

```
SELECT
  EXTRACT(YEAR FROM order_purchase_timestamp) AS `YEAR`,
  COUNT(*) AS `ORDERS_PLACED`
FROM
  `SQL.orders`
GROUP BY `YEAR`
ORDER BY `YEAR` ASC;
```

JOB INFORMATION | RESULTS | CHART | PREVIEW | JSC

Row	YEAR	ORDERS_PLACED
1	2016	329
2	2017	45101
3	2018	54011

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

```sql
 SELECT
  FORMAT_DATE('%B',(order_purchase_timestamp)) AS SEASONALITY_OF_ORDERS,
  COUNT(*) AS ORDER_COUNT
FROM
  `.SQL.orders`
GROUP BY
  SEASONALITY_OF_ORDERS
ORDER BY
  ORDER_COUNT DESC;
```

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON

Row	SEASONALITY_OF_ORDERS	ORDER_COUNT
1	August	10843
2	May	10573
3	July	10318
4	March	9893

1. 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
    - Can we see some kind of monthly seasonality in terms of the no. of orders being placed?13-18 hrs : Afternoon
    - 19-23 hrs : Night

```sql
WITH OrderCounts AS (
  SELECT
    EXTRACT(HOUR FROM order_purchase_timestamp) AS hour_of_day,
    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'
    END AS time_of_day_category,
    EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
    COUNT(*) AS order_count
  FROM
    `SQL.orders`
  GROUP BY
    hour_of_day, time_of_day_category, month
)
SELECT
  hour_of_day,
  time_of_day_category,
  month,
  order_count
FROM (
  SELECT
    hour_of_day,
    time_of_day_category,
    month,
    order_count,
    ROW_NUMBER() OVER (PARTITION BY month ORDER BY order_count DESC) AS row_num
  FROM
    OrderCounts
) AS ranked
WHERE  row_num = 1 ORDER BY month, order_count DESC;
```

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

Row	hour_of_day	time_of_day_category	month	order_count
1	14	Afternoon	1	560
2	14	Afternoon	2	607
3	14	Afternoon	3	719
4	20	Night	4	628
5	16	Afternoon	5	761
6	11	Morning	6	695
7	13	Afternoon	7	707
8	14	Afternoon	8	736
9	15	Afternoon	9	304
10	14	Afternoon	10	346

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAIL

Row	hour_of_day	time_of_day_category	month	order_count
3	14	Afternoon	3	719
4	20	Night	4	628
5	16	Afternoon	5	761
6	11	Morning	6	695
7	13	Afternoon	7	707
8	14	Afternoon	8	736
9	15	Afternoon	9	304
10	14	Afternoon	10	346
11	13	Afternoon	11	527
12	11	Morning	12	396

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

1. Get the month on month no. of orders placed in each state.
2. How are the customers distributed across all the states?

```
SELECT
  EXTRACT(MONTH FROM o.order_purchase_timestamp) AS Months,
  c.customer_state,
  COUNT(*) AS orders_placed_on_the_mentioned_month
FROM
  `SQL.orders` AS o
INNER JOIN `SQL.customers` AS c ON o.customer_id = c.customer_id
GROUP BY
  Months,
  customer_state;
```

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXECU

Row	Months	customer_state	orders_placed_on_the_mentioned_month
1	11	RJ	1048
2	12	RS	283
3	12	SP	2357
4	2	DF	196
5	11	PR	378
6	4	MT	92
7	7	MA	79
8	7	AL	40
9	7	SP	4381
10	7	MT	85

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXEC

Row	Months	customer_state	orders_placed_on_the_mentioned_month
11	7	MG	1111
12	5	MG	1190
13	5	SP	4632
14	5	PE	174
15	10	SP	1908
16	1	RJ	990
17	1	SP	3351
18	1	DF	151
19	1	RS	427
20	6	PE	140

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXE

Row	Months	customer_state	orders_placed_on_the_mentioned_month
21	9	DF	97
22	2	SP	3357
23	7	SE	42
24	12	RJ	783
25	12	PR	271
26	3	RS	569
27	2	PA	83
28	3	RJ	1302
29	3	MG	1237
30	10	PE	87

Query results

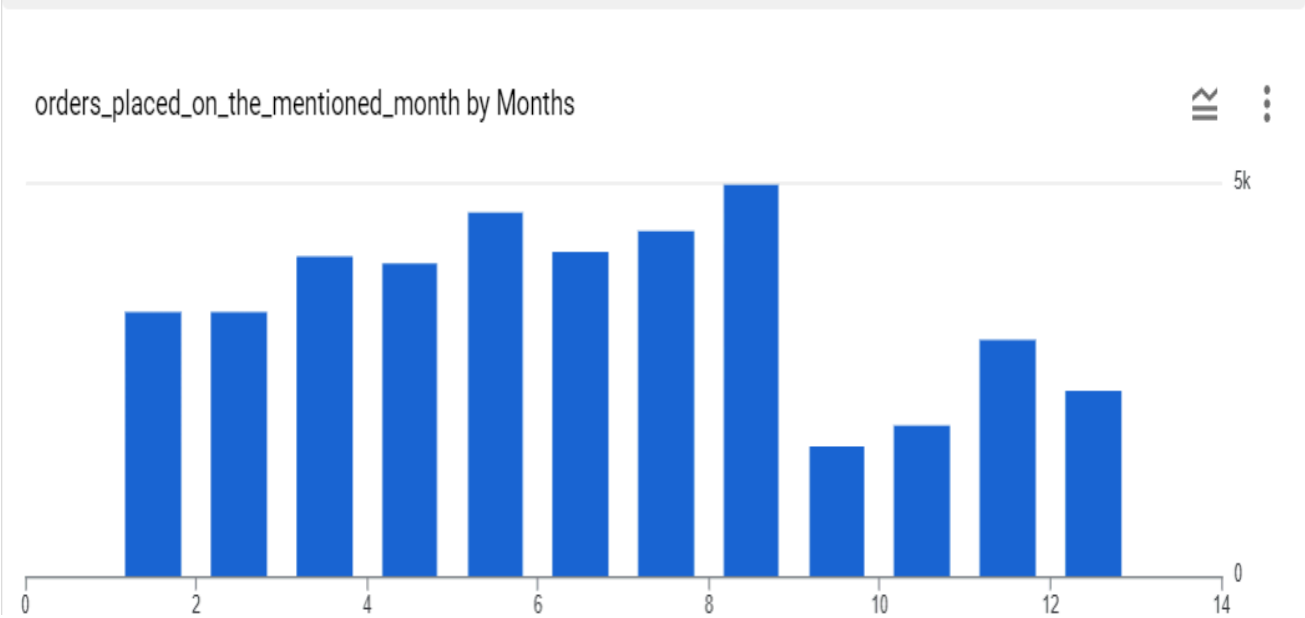
JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXE

Row	Months	customer_state	orders_placed_on_the_mentioned_month
31	4	SP	3967
32	4	RJ	1172
33	4	RS	488
34	4	BA	318
35	1	CE	99
36	1	PE	113
37	5	DF	208
38	5	GO	226
39	5	BA	368
40	5	RJ	1321

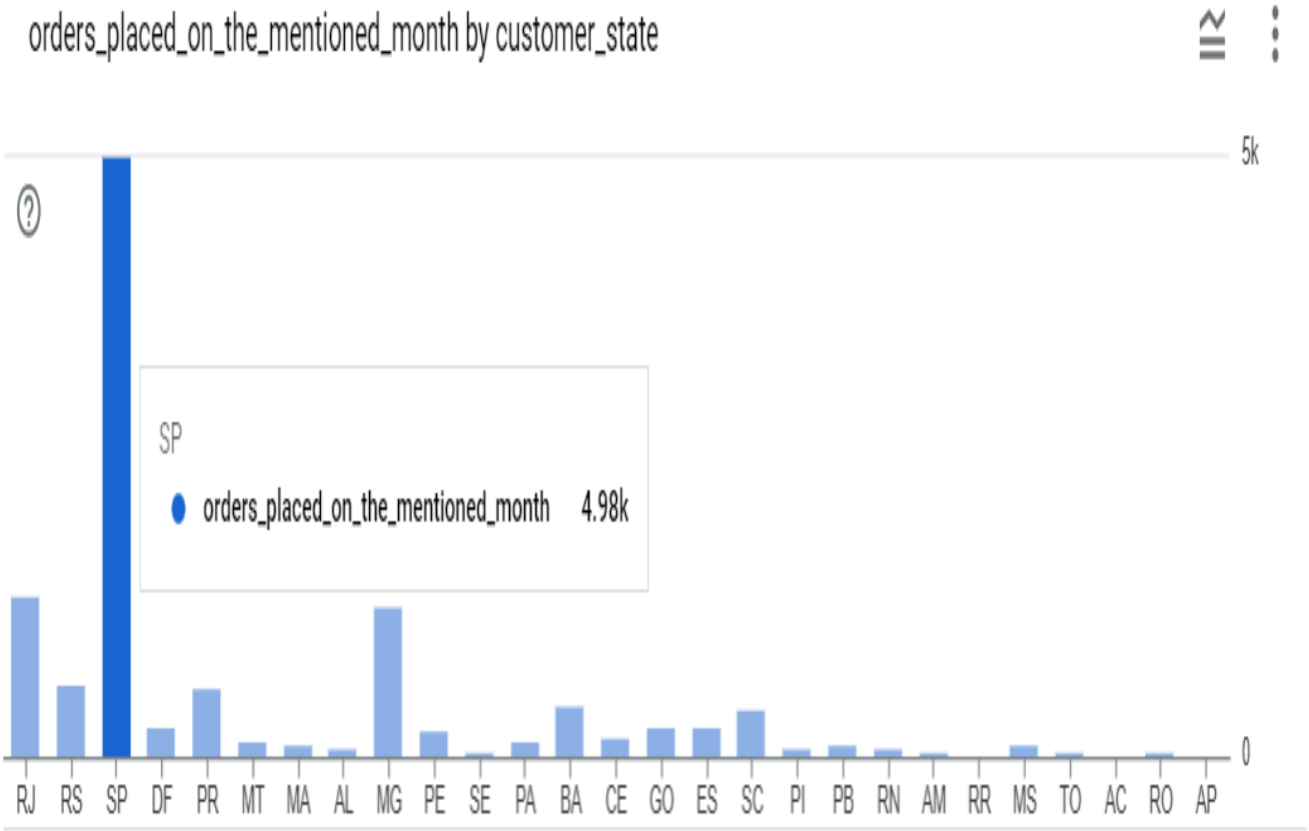
Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXI

Row	Months	customer_state	orders_placed_on_the_mentioned_month
41	6	MG	1080
42	6	RJ	1128
43	6	SP	4104
44	4	CE	143
45	3	PA	109
46	3	MT	71
47	1	PR	443
48	6	CE	121
49	6	DF	220
50	6	SE	37



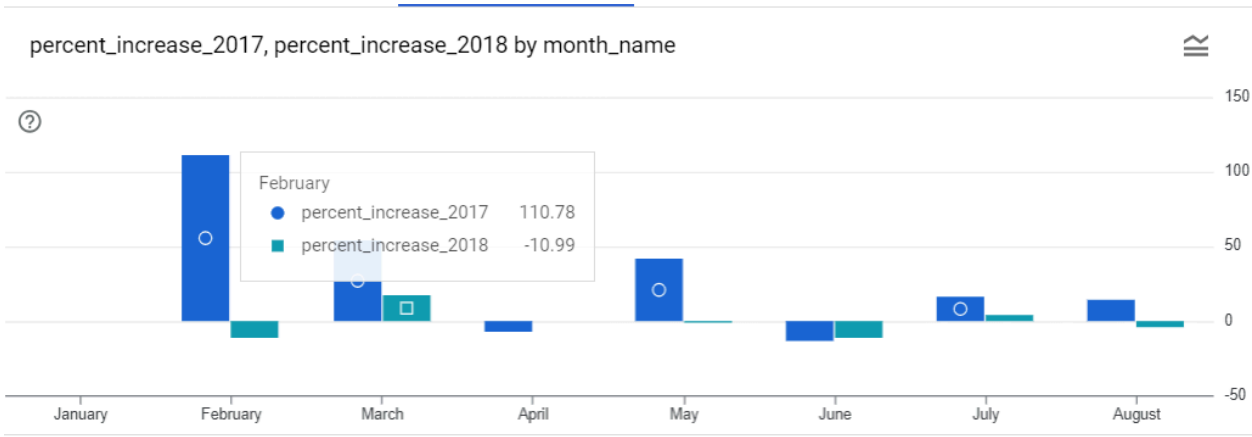

orders_placed_on_the_mentioned_month by Months
5k
0
8 - 9
orders_placed_on_the_mentioned_month 4.98k
0
2
4
6
8
10
12
14

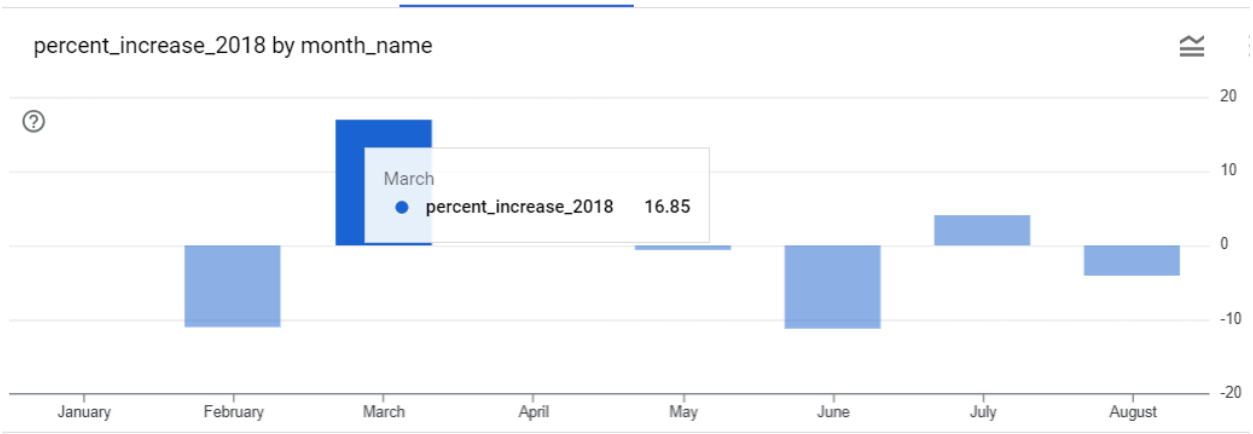

orders_placed_on_the_mentioned_month by customer_state
5k
0
SP
orders_placed_on_the_mentioned_month 4.98k
RJ RS SP DF PR MT MA AL MG PE SE PA BA CE GO ES SC PI PB RN AM RR MS TO AC RO AP

1. 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.
    2. Calculate the Total & Average value of order price for each state.
    3. Calculate the Total & Average value of order freight for each state.

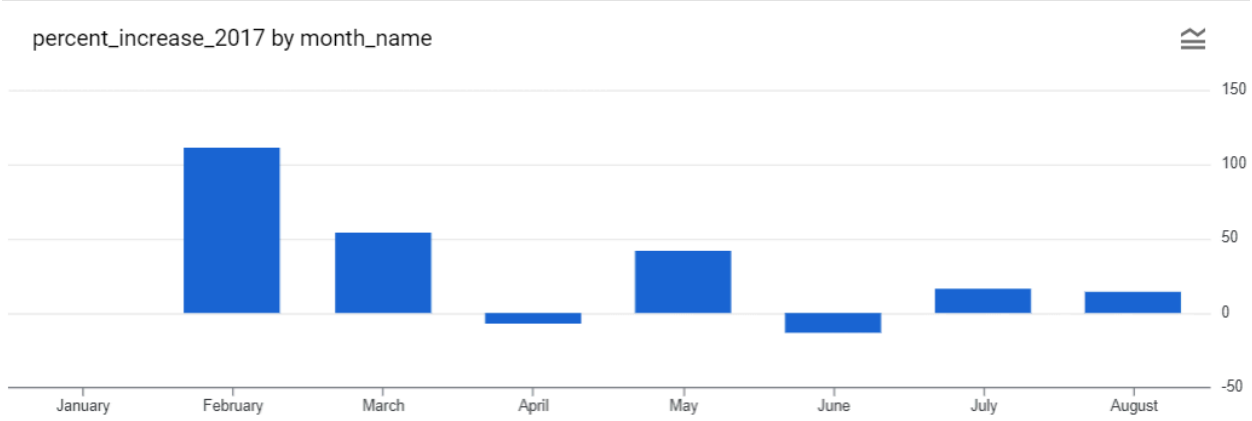
```sql
WITH OrderCosts AS (
  SELECT
    EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
    EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
    FORMAT_DATE('%B', DATE_TRUNC(o.order_purchase_timestamp, MONTH)) AS month_name,
    SUM(p.payment_value) AS total_cost
  FROM
    `SQL.orders` o
    INNER JOIN `SQL.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, month_name
)

SELECT
  OC.month,
  OC.month_name,
  COALESCE(MAX(CASE WHEN OC.year = 2017 THEN OC.total_cost END),0) AS total_cost_2017,
  COALESCE(MAX(CASE WHEN OC.year = 2018 THEN OC.total_cost END),0)AS total_cost_2018,
  COALESCE(MAX(CASE WHEN OC.year = 2017 THEN OC.percent_increase END),0) AS
percent_increase_2017,
  COALESCE(MAX(CASE WHEN OC.year = 2018 THEN OC.percent_increase END),0) AS
percent_increase_2018
FROM (
  SELECT
    year,
    month,
    month_name,
```

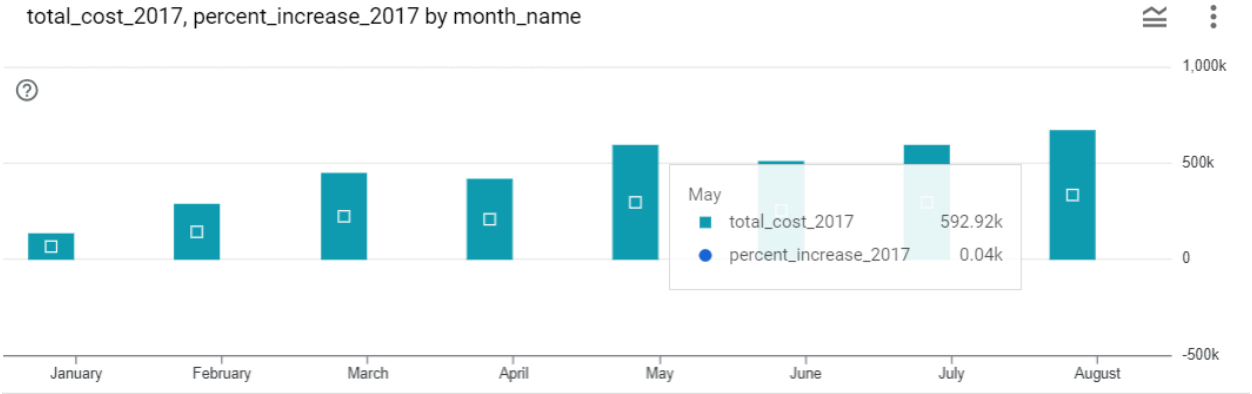
```sql
    total_cost,
    100 * (total_cost - LAG(total_cost) OVER (PARTITION BY year ORDER BY month)) /
      COALESCE(NULLIF(LAG(total_cost) OVER (PARTITION BY year ORDER BY month), 0), 1)
AS percent_increase
  FROM
    OrderCosts
) OC
WHERE
  OC.year IN (2017, 2018)
GROUP BY
  OC.month, OC.month_name
ORDER BY
  OC.month;
```

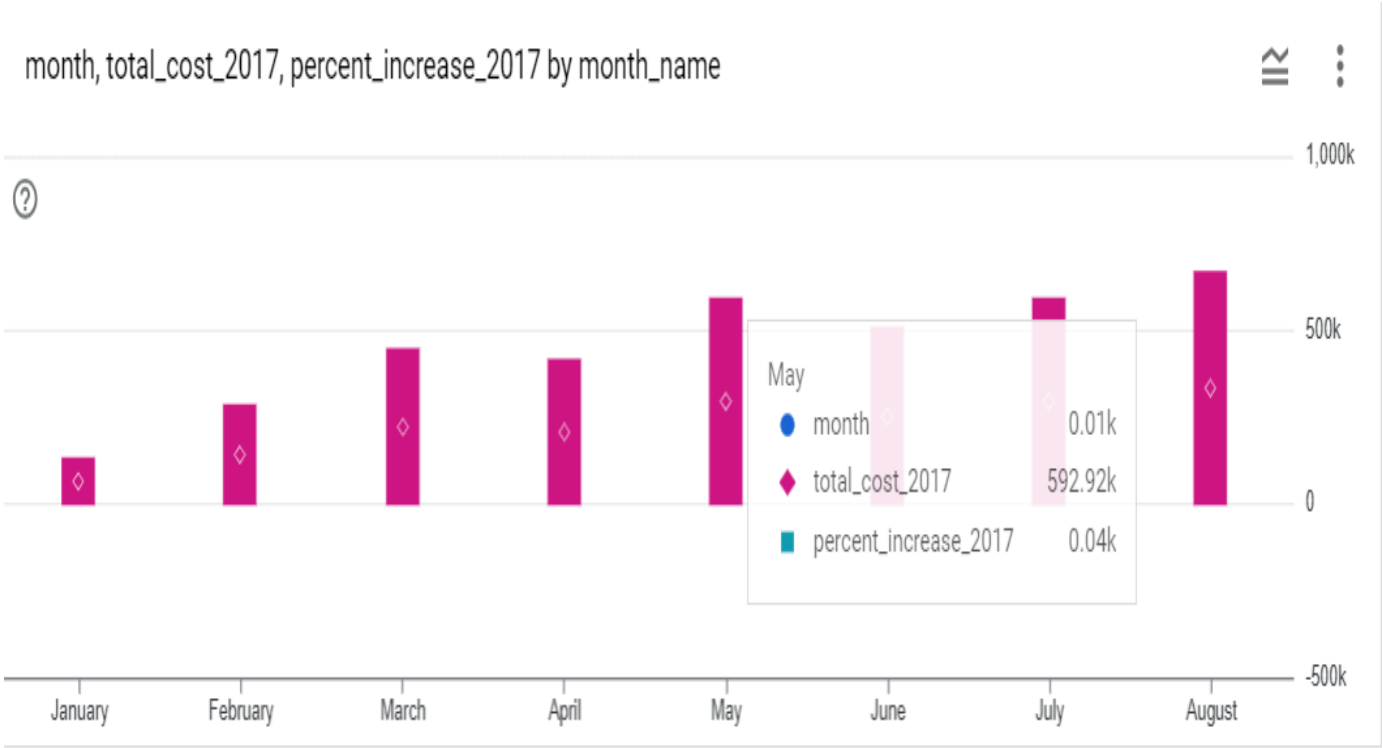




JOB INFORMATION RESULTS CHART PREVIEW JSON EXECUTION DETAILS EXECUTION GRAPH







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

```
SELECT
  SUM(o.price) AS `Sum_of_ordered_goods`,
  AVG(o.price) AS `Average_on_ordered_goods`,
  c.customer_state
FROM
  `SQL. order_items` as o cross join `SQL.customers` as c
group by
  customer_state;
```

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

Row	Sum_of_ordered_goods	Average_on_ordered_goods	customer_state
1	567396757896.372	120.65373901464714	SP
2	158138774450.79514	120.6537390146472	MG
3	74291924463.776169	120.65373901464726	RS
4	45939755705.8117	120.6537390146472	BA
5	3438685856.0993176	120.65373901464712	RO
6	68569842466.070885	120.6537390146473	PR
7	174679804833.92853	120.65373901464724	RJ
8	13251852607.516779	120.65373901464724	PA
9	7285121023.2029791	120.65373901464713	PB
10	49432808136.662361	120.653739014647	SC
11	18158435983.226513	120.65373901464713	CE
12	27455120274.019539	120.65373901464717	GO
13	6591947194.5013847	120.65373901464707	RN
14	6727863631.5017891	120.65373901464714	PI
15	9718025245.5079918	120.6537390146471	MS
16	12327620835.91456	120.6537390146474	MT
17	5613348848.1002169	120.65373901464712	AL
18	27631811642.122162	120.65373901464736	ES
19	3805660235.999382	120.65373901464717	TO
20	924231771.60001433	120.65373901464706	AP
21	2011563267.59974	120.65373901464704	AM
22	10152957843.909266	120.65373901464712	MA
23	22453395392.43071	120.65373901464714	PE
24	4757075294.9995852	120.65373901464729	SE
25	29086117518.011196	120.65373901464729	DF
26	1100923139.6999643	120.65373901464736	AC
27	625215610.19999051	120.65373901464726	RR

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

```
SELECT
  c.customer_state,
  AVG(o.freight_value) AS `AVERAGE_FREIGHT_VALUE`
FROM
  `SQL. order_items` o
CROSS JOIN
  `SQL.customers` c
GROUP BY
  C.customer_state;
```

Row	customer_state	AVERAGE_FREIGHT_VALUE
1	SP	19.99031992898362
2	MG	19.9903199289836
3	RS	19.990319928983574
4	BA	19.990319928983592
5	MA	19.990319928983549
6	PE	19.99031992898357
7	RJ	19.9903199289836
8	PA	19.990319928983556
9	PB	19.990319928983588
10	SC	19.99031992898356
11	PI	19.990319928983595
12	CE	19.990319928983592

13	PR	19.990319928983645
14	MT	19.990319928983563
15	ES	19.990319928983588
16	RN	19.990319928983578
17	GO	19.990319928983574
18	RO	19.990319928983553
19	AP	19.990319928983631
20	AL	19.990319928983574
21	AM	19.990319928983585
22	TO	19.990319928983535
23	SE	19.99031992898356
24	MS	19.990319928983542

4).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.

```
select
  order_id,
  timestamp_diff(order_delivered_customer_date,order_purchase_timestamp,day) as
`delivery_time`,
  timestamp_diff(order_delivered_customer_date,order_estimated_delivery_date,day) as
`delivery_delay`
 from
`SQL.orders`;
```

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION I

Row	order_id	delivery_time	delivery_delay
1	1950d777989f6a877539f5379...	30	12
2	2c45c33d2f9cb8ff8b1c86cc28...	30	-28
3	65d1e226dfaeb8cdc42f66542...	35	-16
4	635c894d068ac37e6e03dc54e...	30	-1

(4.2)Find out the top 5 states with the highest & lowest average freight value.

```sql
WITH OrderedValues AS (
  SELECT
    freight_value,
    ROW_NUMBER() OVER (ORDER BY freight_value DESC) AS rank_desc
  FROM
    `SQL. order_items`
)
SELECT
  AVG(freight_value) AS Average_top_5
FROM
  OrderedValues
WHERE
  rank_desc <=5
GROUP BY
   rank_desc
ORDER BY
   rank_desc
LIMIT 5  ;
```

Row	Average_top_5
1	409.68
2	375.28
3	375.28
4	339.59
5	338.3

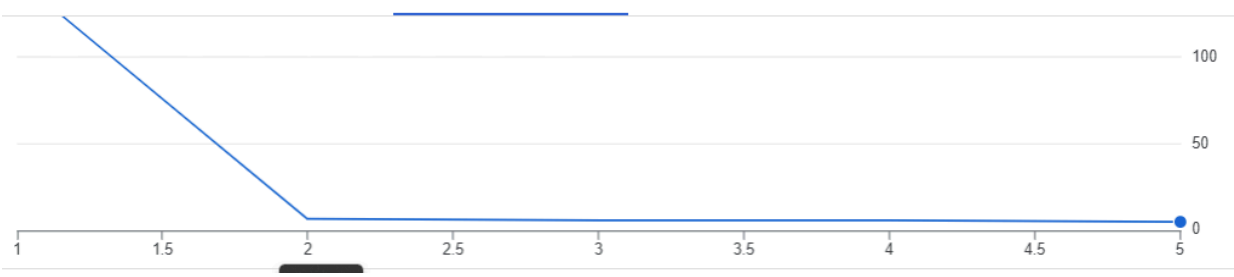

```sql
WITH OrderedValues AS (
  SELECT
    freight_value,
    ROW_NUMBER() OVER (ORDER BY freight_value ASC) AS rank_asc
  FROM
    `SQL. order_items`
)
SELECT
  AVG(freight_value) AS Average_bottom_5_lowest
FROM
  OrderedValues
WHERE
   rank_asc <=5
GROUP BY
   rank_asc
ORDER BY
   rank_asc
LIMIT 5  ;
```

JOB INFORMATION RESULTS

Row	Average_bottom_5_lowest
1	0.0
2	0.0
3	0.0
4	0.0
5	0.0

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

```
SELECT
  c.customer_state,
  o.order_approved_at,
```




```
  o.order_estimated_delivery_date,
  AVG(DATE_DIFF(o.order_approved_at,o.order_estimated_delivery_date,day)) as
`AVERAGE_OF_DIFFERENCE_IN_DAYS`

FROM
 `SQL.orders` o INNER JOIN `SQL.customers`  c ON o.customer_id = c.customer_id
GROUP BY
  customer_state,order_approved_at,order_estimated_delivery_date
ORDER BY
```

```
  AVERAGE_OF_DIFFERENCE_IN_DAYS DESC
LIMIT 5;
```

Query results

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

Row	customer_state	order_approved_at	order_estimated_delivery_date	AVERAGE_OF_DIFFEI
1	SP	2017-04-11 15:17:38 UTC	2016-11-17 00:00:00 UTC	145.0
2	SP	2016-10-07 13:16:46 UTC	2016-09-30 00:00:00 UTC	7.0
3	SP	2018-08-20 15:56:29 UTC	2018-08-14 00:00:00 UTC	6.0
4	RJ	2018-02-20 12:05:54 UTC	2018-02-14 00:00:00 UTC	6.0
5	PR	2018-08-20 15:59:18 UTC	2018-08-15 00:00:00 UTC	5.0

customer_state

6.Analysis based on the payments:

1. Find the month on month no. of orders placed using different payment types.
2. Find the no. of orders placed on the basis of the payment installments that have been paid.

```
SELECT
  EXTRACT(MONTH FROM o.order_purchase_timestamp) AS MONTH_OF_ORDER,
  FORMAT_DATE('%B',o.order_purchase_timestamp ) AS MONTH_NAME_OF_ORDER,
  P.payment_type,
  COUNT(*) AS MOST_USED_PAYMENT_WINDOW
FROM
  `SQL.orders` AS o
INNER JOIN
  `SQL.payments` AS P ON o.order_id = P.order_id
GROUP BY
  MONTH_OF_ORDER,
  MONTH_NAME_OF_ORDER,
  P.payment_type
```

```sql
ORDER BY
  MONTH_OF_ORDER,
  MONTH_NAME_OF_ORDER,
  MOST_USED_PAYMENT_WINDOW  DESC;
```

Query results

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

Row	MONTH_OF_ORDER	MONTH_NAME_OF_ORDER	payment_type	MOST_USED_PAYMENT_WINDOW
1	1	January	credit_card	6103
2	1	January	UPI	1715
3	1	January	voucher	477
4	1	January	debit_card	118