Retail Business Case: MySQL

Priyanka Murali

https://www.linkedin.com/in/priyankamurali

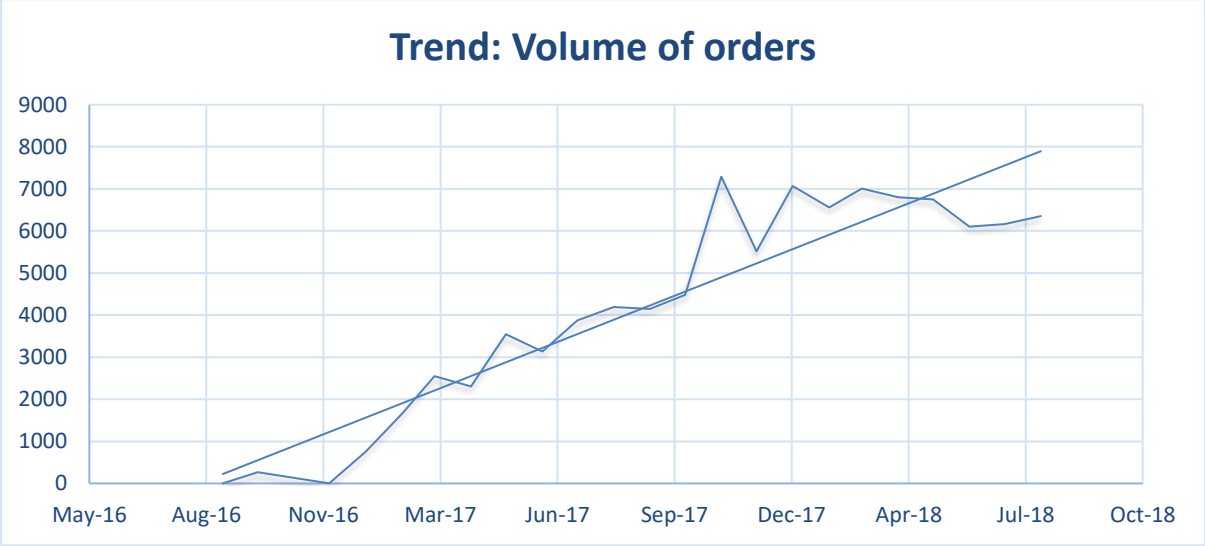
Background

Business case analysis of a large commercial retailer, from a dataset of over 100k orders. Analyses was done using MySQL and the results and recommendations are expanded in the sections that follow.

Context: data from a commercial retail company operating in Brazil between 2016 and 2018.

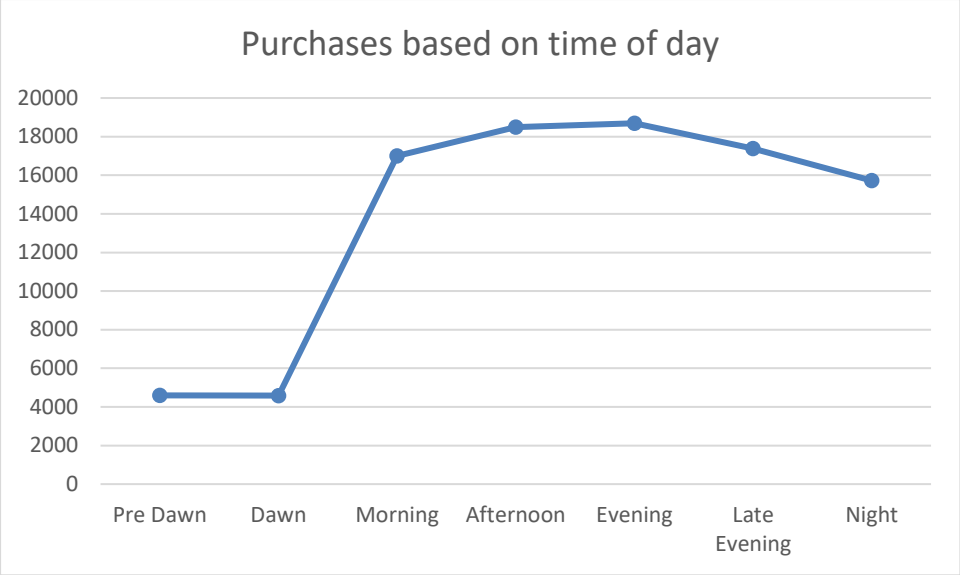
Actionable insights

1. Trend in orders: Brazilian market is favourable and shows a clear upward trend in volume of orders. And the avg revenue per order has increased **4.8%** between 2017 & 2018 (Jan-Aug only).



Year_of_purchase	Payment value per order	% change in payment value per order	Cost of freight per order	% change in cost of freight per order
2017	195.91	NULL	23.11	NULL
2018	205.39	4.84	24.33	5.27

2. Time of day:



Time of day	Starts At	Ends At	# Of purchases	% Of purchases
Pre-Dawn	00:00:01	06:00:00	4593	5%
Dawn	06:00:01	09:00:00	4586	5%
Morning	09:00:01	12:00:00	17008	18%
Afternoon	12:00:01	15:00:00	18496	19%
Evening	15:00:01	18:00:00	18691	19%
Late Evening	18:00:01	21:00:00	17389	18%
Night	21:00:01	00:00:00	15715	16%

90% of orders are placed between 9AM and 12PM. Peak traffic is between 12 PM and 6PM (40% of the traffic).

Peak time slot on weekends is 6PM to 9PM. Whereas peak on weekdays is 12PM to 6 PM.

3. Distribution of customers: Top 5 states (SP, RJ, MG, RS and PR) account for 77% (more than3/4th) of customers.

State	Num_of_cust	% Num_of_cust
SP	40501	41.98
RJ	12350	12.80
MG	11354	11.77
RS	5345	5.54
PR	4923	5.10
SC	3546	3.68
BA	3256	3.37
DF	2080	2.16
ES	1995	2.07
GO	1957	2.03
PE	1593	1.65
CE	1279	1.33

4. Service level accuracy: 97% of all orders are delivered before the estimated time to deliver.
5. Cost of freight: States where large % of customers are situated also have the lowest avg freight cost per order (SP, MG, PR and RJ)

State	Mean of freight	Mean of time_to_delivery	Mean of diff_estimated_delivery
SP	17.33	9.93	-12.85
MG	23.46	13.56	-15.18
PR	23.49	13.65	-15.47
DF	23.86	14.60	-13.81
RJ	23.95	17.26	-13.75

6. Payment mode: 77% of orders were paid with credit cards. Next most popular payment option was UPI (20%).

Payment type	# of orders	% of total orders
credit_card	74304	77%
UPI	19191	20%
voucher	3679	4%
debit_card	1485	2%

Recommendations

1. Based on seasonality of data March, May and Nov are peak seasons for orders. Consider stocking up on inventory ahead of time. There is a dip in volume in the month of June, which is a good opportunity to run clearance sales to boost volumes and get rid of old/excess inventory.
2. 77% of customers use credit cards. Tie up with banks/ financial institutions to offer special rates (lower interest payments) or issue a custom **credit card** using which customers can avail discounts as well as collect points to redeem for cash or other goodies. This will engender loyalty and bring repeat customers.
    1. Also, most customers pay back within 3 instalments so consider offering 3 month 0% interest payment to loyal customers.

3. 77% of customers purchasing from reside in these 5 states: SP, RJ, MG, RS and PR. Cost of shipping in these states are also relatively low. Consider offering shorter delivery times in these states to entice customers to order more frequently.
4. By State

State	Revenue	% Revenue	Payment value per order	% change in payment value per order	Cost of freight per order	% change in cost of freight per order
SP	4309511	40%	184.68	5.35	18.15	-0.97
RJ	1332572	12%	210.12	4.34	26.44	11.28
MG	1217142	11%	200.22	9.71	25.24	9.69
RS	582525.3	5%	212.83	17.57	27.33	13.42
PR	571101.2	5%	210.66	5.42	25.52	10.33
SC	434049.8	4%	230.39	11.11	27.76	19.3
BA	426737.6	4%	247.24	5.41	34.35	15.7
GO	244636.1	2%	236.82	-27.39	28.04	4.97
ES	233434.3	2%	222.96	26.84	26.35	9.13
DF	232059.5	2%	194.52	-12.12	24.85	2.57
PE	186087.6	2%	218.41	4.19	37.83	7
CE	172646.8	2%	272.74	23.86	40.63	15.23
MT	129152.7	1%	271.33	-5.53	35.4	16.8
PA	127745.9	1%	282	1.08	43.73	14.28
PB	104663.5	1%	389.08	31.13	59.13	33.83
MA	104278.9	1%	301.38	70.92	47.59	33.36
MS	90696.39	1%	220.67	-19.03	29.15	14.26
PI	77958.95	1%	302.17	-4.9	49.45	19.99
RN	61728.68	1%	256.14	28.35	44.43	10.87
AL	45616.01	0%	230.38	-21.05	40.05	14.28
TO	39329.01	0%	273.12	12.87	58.18	64.86
SE	39004.47	0%	267.15	20.9	43.73	6.64
RO	28464.83	0%	261.15	10.6	47.49	11.44
AM	19723.19	0%	273.93	48.81	43.38	33.52
AP	11373.73	0%	291.63	-20	39.26	-13.13
AC	9915.3	0%	367.23	30.57	52.75	35.7
RR	9007.32	0%	409.42	231.39	61.41	90.18

1.
2. Bottom 15 states only account for 8% of revenue. Possible to consider moving out of these states to cut costs and focus on improving quality of service in other states. Freight to these states also cost more money so charge extra for expedited delivery to customers who live here.
3. Consider limited time offers like “buy 2 get 1 free” or bundling offers (like buy a printer get cartridge at 50% off) to help increase the revenue per order.

5. Time of day
    1. Least traffic is seen between 12AM and 9AM (~10%). This time can be utilized for website maintenance / restock the store.
    2. Peak time slot on weekends is 6PM to 9PM. Whereas peak on weekdays is 12PM to 6 PM. Ensure shelves are re-stocked in the late evening/ website can handle addition bandwidth.

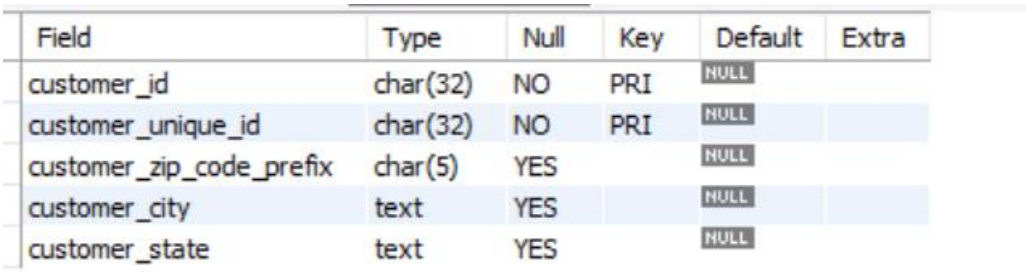
On Weekends					On weekdays			
1	Pre Dawn	1184	5%		1	Pre Dawn	3409	5%
2	Dawn	732	3%		2	Dawn	3854	5%
3	Morning	3302	15%		3	Morning	13706	18%
4	Afternoon	4017	18%		4	Afternoon	14479	19%
5	Evening	4212	19%		5	Evening	14479	19%
6	Late Evening	4829	22%		6	Late Evening	12560	17%
7	Night	3914	18%		7	Night	11801	16%

MySQL queries

1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset
    1. Data type of columns in a table

    The customers table contain following columns:

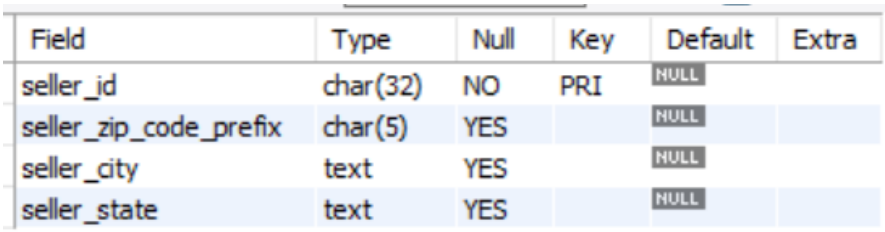
    DESC store_sql.customers



Field	Type	Null	Key	Default	Extra
customer_id	char(32)	NO	PRI	NULL	
customer_unique_id	char(32)	NO	PRI	NULL	
customer_zip_code_prefix	char(5)	YES		NULL	
customer_city	text	YES		NULL	
customer_state	text	YES		NULL	

    The sellers table contain following columns:

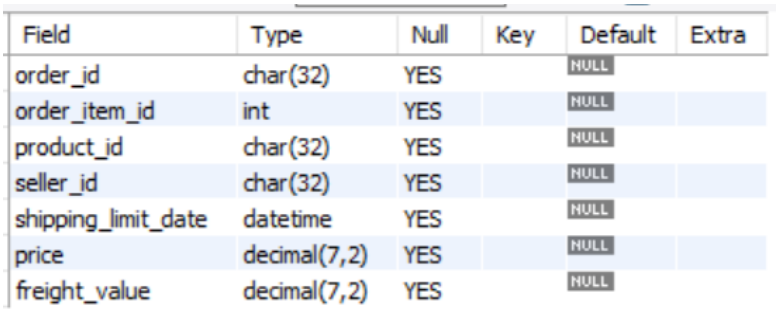
    DESC store_sql.sellers



Field	Type	Null	Key	Default	Extra
seller_id	char(32)	NO	PRI	NULL	
seller_zip_code_prefix	char(5)	YES		NULL	
seller_city	text	YES		NULL	
seller_state	text	YES		NULL	

    The order_items table contain following columns:

    DESC store_sql.order_items



Field	Type	Null	Key	Default	Extra
order_id	char(32)	YES		NULL	
order_item_id	int	YES		NULL	
product_id	char(32)	YES		NULL	
seller_id	char(32)	YES		NULL	
shipping_limit_date	datetime	YES		NULL	
price	decimal(7,2)	YES		NULL	
freight_value	decimal(7,2)	YES		NULL	

    The geolocation table contain following columns:

DESC store_sql.geolocation

Field	Type	Null	Key	Default	Extra
geolocation_zip_code_prefix	char(5)	YES		NULL	
geolocation_lat	decimal(7,4)	YES		NULL	
geolocation_lng	decimal(7,4)	YES		NULL	
geolocation_city	text	YES		NULL	
geolocation_state	text	YES		NULL	

The payments table contain following columns:

DESC store_sql.payments

Field	Type	Null	Key	Default	Extra
order_id	char(32)	YES		NULL	
payment_sequential	int	YES		NULL	
payment_type	text	YES		NULL	
payment_installments	int	YES		NULL	
payment_value	decimal(10,2)	YES		NULL	

The orders table contain following columns:

DESC store_sql.orders

Field	Type	Null	Key	Default	Extra
order_id	char(32)	NO	PRI	NULL	
customer_id	char(32)	NO	PRI	NULL	
order_status	text	YES		NULL	
order_purchase_timestamp	datetime	YES		NULL	
order_approved_at	datetime	YES		NULL	
order_delivered_carrier_date	datetime	YES		NULL	
order_delivered_customer_date	datetime	YES		NULL	
order_estimated_delivery_date	datetime	YES		NULL	

The reviews table contain following columns:

DESC store_sql.order_reviews

Field	Type	Null	Key	Default	Extra
review_id	char(32)	NO	PRI	NULL	
order_id	char(32)	NO	PRI	NULL	
review_score	int	YES		NULL	
review_comment_title	text	YES		NULL	
review_creation_date	datetime	YES		NULL	
review_answer_timestamp	datetime	YES		NULL	

The products table contain following columns:

DESC store_sql.products

Field	Type	Null	Key	Default	Extra
product_id	char(32)	NO	PRI	NULL	
product_category	text	YES		NULL	
product_name_length	int	YES		NULL	
product_description_length	int	YES		NULL	
product_photos_qty	int	YES		NULL	
product_weight_g	int	YES		NULL	
product_length_cm	int	YES		NULL	
product_height_cm	int	YES		NULL	
product_width_cm	int	YES		NULL	

2. Time period for which the data is given

Time period of order data ranges from Sep 2016 to Oct 2018.

```
SELECT Min(order_purchase_timestamp) AS Min_Order_Date,

Max(order_purchase_timestamp) AS Max_Order_Date

FROM   store_sql.orders
```

Min_Order_Date	Max_Order_Date
2016-09-04 21:15:19	2018-10-17 17:30:18

3. Cities and States of customers ordered during the given period

```
SELECT DISTINCT c.customer_state,
                c.customer_city
FROM   store_sql.orders o
       JOIN store_sql.customers c
         ON o.customer_id = c.customer_id
ORDER  BY c.customer_state,
          c.customer_city
```

	customer_state	customer_city
	AL	porto de pedras
	AL	rio largo
	AL	santa luzia do norte
	AL	santana do ipane...
	AL	santana do mundau
	AL	sao bras
	AL	sao jose da laje
	AL	sao jose da tapera
	AL	sao luis do quitunde
	AL	sao miguel dos c...
	AL	sao sebastiao

2. In-depth Exploration:
    1. 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?

Trend in ecommerce

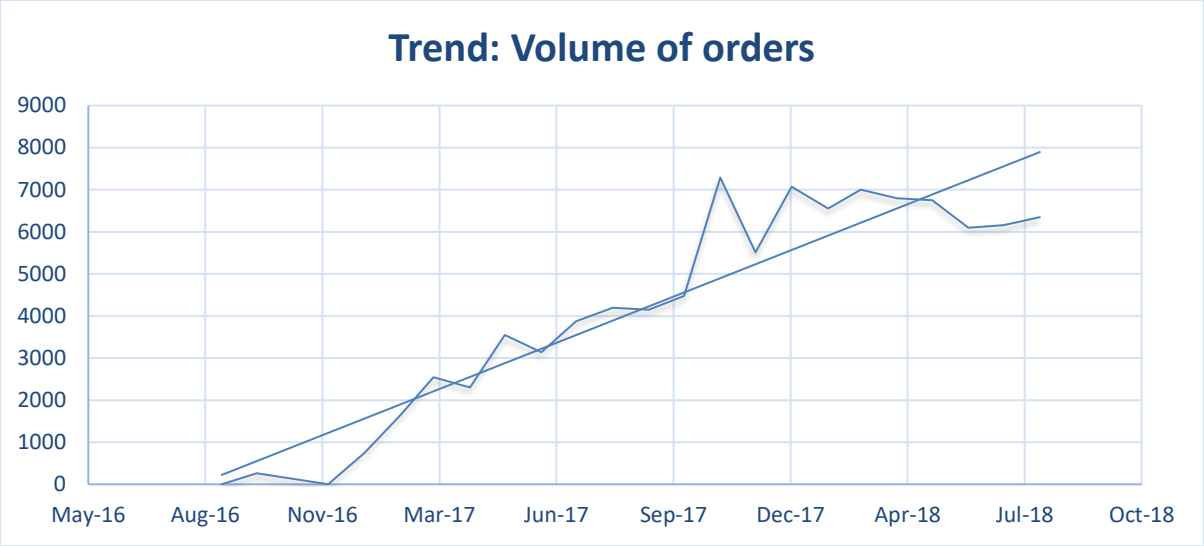
Restricting analyses only to orders where status is marked as **delivered**.

Query:

```
WITH orders_by_month AS
(
       SELECT date_sub(Date(order_purchase_timestamp), interval dayofmo
nth(date(order_purchase_timestamp)) - 1 day) AS month_purchased
       FROM   store_sql.orders o
       WHERE  order_status = 'delivered')
SELECT   date_format(month_purchased,"%b %Y") AS month_yr_purchased,
         date_format(month_purchased,"%b")    AS month_of_purchase,
         count(1)                             AS num_of_orders
FROM     orders_by_month
GROUP BY month_purchased
ORDER BY month_purchased
```

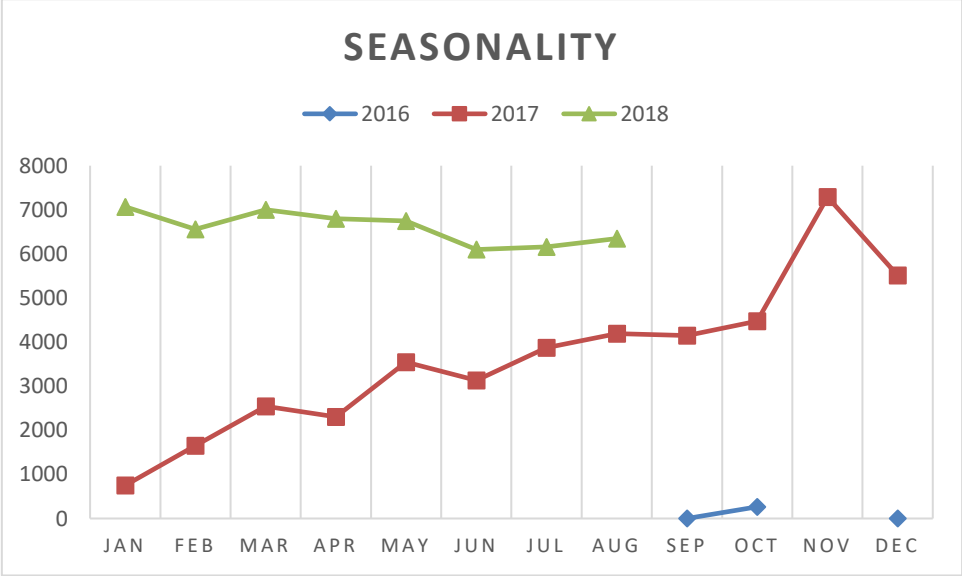
Month_Yr_Purchased	Month_of_Purchase	Num_of_Orders
Sep 2016	Sep	1
Oct 2016	Oct	265
Dec 2016	Dec	1
Jan 2017	Jan	750
Feb 2017	Feb	1653
Mar 2017	Mar	2546
Apr 2017	Apr	2303
May 2017	May	3546
Jun 2017	Jun	3135
Jul 2017	Jul	3872
Aug 2017	Aug	4193

Results:



Conclusion: There is a clear upward trend in volume of orders. The ecommerce market in Brazil is expanding and appears to be a favourable market.

Seasonality



There appears to be a peak in purchases in March and May for 2017 and 2018.

There is also a sharp peak for Nov in 2017, but data is available only for one year (would be better to observe data for another year before confirming this trend).

There is a dip in purchases for the month of June.

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

Assumptions:

Tracking only for successful purchases where orders where status is marked as **delivered**.

Defining time slots:

Time Slot	Starts At	Ends At
Pre-Dawn	00:00:01	06:00:00
Dawn	06:00:01	09:00:00
Morning	09:00:01	12:00:00
Afternoon	12:00:01	15:00:00
Evening	15:00:01	18:00:00
Late Evening	18:00:01	21:00:00
Night	21:00:01	00:00:00

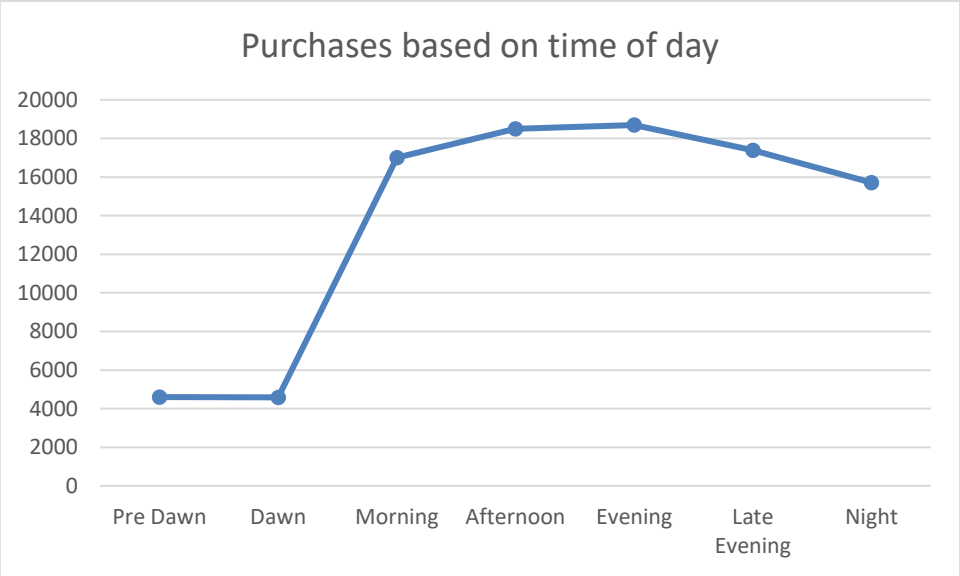
Query:

```
WITH orders_purchased
     AS (SELECT order_purchase_timestamp,
                CASE
                  WHEN Time(order_purchase_timestamp) BETWEEN
                       '00:00:01' AND '06:00:00'
                   THEN
                  'Pre Dawn'
                  WHEN Time(order_purchase_timestamp) BETWEEN
                       '06:00:01' AND '09:00:00'
                   THEN
                  'Dawn'
                  WHEN Time(order_purchase_timestamp) BETWEEN
                       '09:00:01' AND '12:00:00'
                   THEN
                  'Morning'
                  WHEN Time(order_purchase_timestamp) BETWEEN
                       '12:00:01' AND '15:00:00'
                   THEN
                  'Afternoon'
                  WHEN Time(order_purchase_timestamp) BETWEEN
                       '15:00:01' AND '18:00:00'
                   THEN
                  'Evening'
                  WHEN Time(order_purchase_timestamp) BETWEEN
                       '18:00:01' AND '21:00:00'
                   THEN
                  'Late Evening'
                  ELSE 'Night'
                END AS Time_of_day
         FROM   store_sql.orders o
         WHERE  order_status = 'delivered')
```

```sql
SELECT time_of_day,
       Count(1) AS Num_of_Purchases
FROM   orders_purchased
GROUP  BY time_of_day
```

Time_of_day	Num_of_Purchases
Dawn	4586
Morning	17008
Afternoon	18496
Night	15715
Late Evening	17389
Evening	18691
Pre Dawn	4593

Results:



Conclusion:

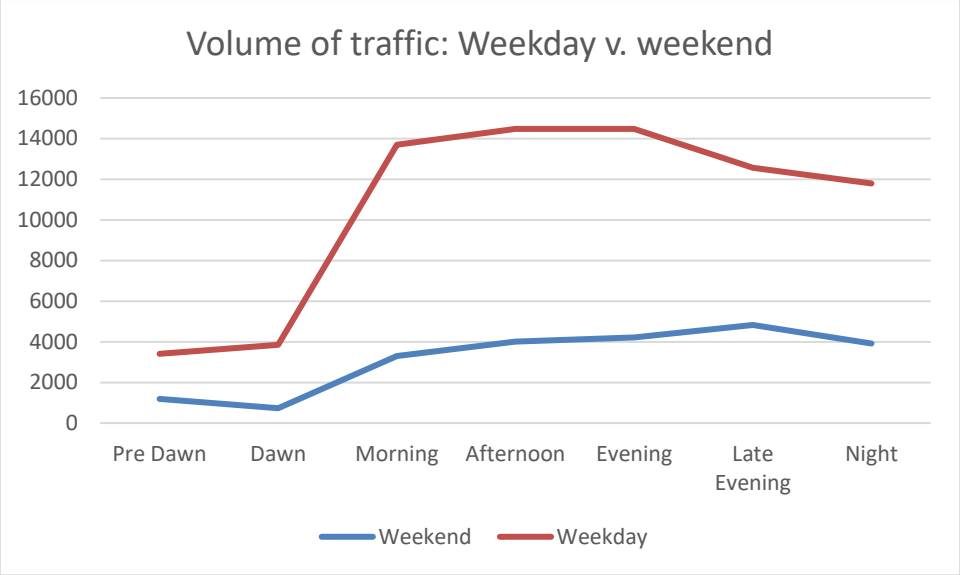
Time of day	Starts At	Ends At	# Of purchases	% Of purchases
Pre-Dawn	00:00:01	06:00:00	4593	5%
Dawn	06:00:01	09:00:00	4586	5%
Morning	09:00:01	12:00:00	17008	18%
Afternoon	12:00:01	15:00:00	18496	19%
Evening	15:00:01	18:00:00	18691	19%
Late Evening	18:00:01	21:00:00	17389	18%

Night	21:00:01	00:00:00	15715	16%

Between 12 PM and 6 PM 39% of purchases are made.

Between 12 AM and 9 PM 90% of purchases are made.

```
WITH order_installments AS
(
         SELECT   o.order_id,
                  Max(p.payment_installments) AS num_of_insatllments
         FROM     store_sql.orders o
         JOIN     store_sql.payments p
         ON       p.order_id = o.order_id
         WHERE    o.order_status = 'delivered'
         GROUP BY o.order_id)
SELECT   num_of_insatllments,
         Count(1) AS num_of_orders
FROM     order_installments
GROUP BY num_of_insatllments
ORDER BY num_of_insatllmenwith orders_purchased AS
         (
                SELECT order_purchase_timestamp,
                       CASE
                              WHEN time(order_purchase_timestamp) BETWE
EN '00:00:01' AND    '06:00:00' THEN 'Pre Dawn'
                              WHEN time(order_purchase_timestamp) BETWE
EN '06:00:01' AND    '09:00:00' THEN 'Dawn'
                              WHEN time(order_purchase_timestamp) BETWE
EN '09:00:01' AND    '12:00:00' THEN 'Morning'
                              WHEN time(order_purchase_timestamp) BETWE
EN '12:00:01' AND    '15:00:00' THEN 'Afternoon'
                              WHEN time(order_purchase_timestamp) BETWE
EN '15:00:01' AND    '18:00:00' THEN 'Evening'
                              WHEN time(order_purchase_timestamp) BETWE
EN '18:00:01' AND    '21:00:00' THEN 'Late Evening'
                              ELSE 'Night'
                       END AS time_of_day
                FROM   store_sql.orders o
                WHERE  order_status = 'delivered'
                AND    dayofweek(o.order_purchase_timestamp) NOT IN (1,
7))SELECT   time_of_day,
           Count(1) AS num_of_purchases
  FROM     orders_purchased
  GROUP BY time_of_day ts
```



	On Weekends				On weekdays		
1	Pre Dawn	1184	5%	1	Pre Dawn	3409	5%
2	Dawn	732	3%	2	Dawn	3854	5%
3	Morning	3302	15%	3	Morning	13706	18%
4	Afternoon	4017	18%	4	Afternoon	14479	19%
5	Evening	4212	19%	5	Evening	14479	19%
6	Late Evening	4829	22%	6	Late Evening	12560	17%
7	Night	3914	18%	7	Night	11801	16%

Peak time slot on weekends is 6PM to 9PM. Whereas peak on week days is 12PM to 6 PM.

3. Evolution of E-commerce orders in the Brazil region:
    1. Get month on month orders by states

```
WITH orders_by_month AS
(
       SELECT date_sub(Date(order_purchase_timestamp), interval dayofmont
h(date(order_purchase_timestamp)) - 1 day) AS month_purchased,
              c.customer_state
       FROM   store_sql.orders o
       JOIN   store_sql.customers c
       ON     c.customer_id = o.customer_id
       WHERE  order_status = 'delivered')
SELECT   date_format(month_purchased,"%b %Y")
                                          AS month_yr_purchased,
         customer_state
                                          AS state,
         count(1)
                                          AS num_of_orders,
         count(1) - lag(count(1),1,0) OVER(partition BY customer_state OR
DER BY month_purchased)                   AS 'Month over month # of ord
ers',
         round(((count(1) / lag(count(1),1,0) OVER(partition BY customer_
state ORDER BY month_purchased)) -1)*100,2) AS '% Change MoM # of orders'
FROM     orders_by_month
GROUP BY customer_state,
         month_purchased
ORDER BY customer_state,
         month_purchased
```

month_yr_purchased	state	num_of_orders	Month over month # of orders	% Change MoM # of orders
Jan 2017	AC	2	2	NULL
Feb 2017	AC	3	1	50.00
Mar 2017	AC	2	-1	-33.33
Apr 2017	AC	5	3	150.00
May 2017	AC	8	3	60.00
Jun 2017	AC	4	-4	-50.00
Jul 2017	AC	5	1	25.00
Aug 2017	AC	4	-1	-20.00
Sep 2017	AC	5	1	25.00
Oct 2017	AC	5	0	0.00
Nov 2017	AC	5	0	0.00
Dec 2017	AC	5	0	0.00

2. Distribution of customers across the states in Brazil

```
WITH orders_from_cust
        AS (SELECT o.customer_id,
                   c.customer_state
            FROM   store_sql.orders o
                   JOIN store_sql.customers c
                     ON c.customer_id = o.customer_id
            WHERE  order_status = 'delivered')
   SELECT customer_state
          AS
          State,
          Count(DISTINCT customer_id)
          AS Num_of_cust,
          Round(( Count(DISTINCT customer_id) / (SELECT Count(DISTINCT customer
   _id)
                                                 FROM   orders_from_cust) ) * 1
   00,
          2) AS
          '% Num_of_cust'
   FROM   orders_from_cust
   GROUP  BY customer_state
   ORDER  BY num_of_cust DESC
```

State	Num_of_cust	% Num_of_cust
SP	40501	41.98
RJ	12350	12.80
MG	11354	11.77
RS	5345	5.54
PR	4923	5.10
SC	3546	3.68
BA	3256	3.37
DF	2080	2.16
ES	1995	2.07
GO	1957	2.03
PE	1593	1.65
CE	1279	1.33

Top 5 states (SP, RJ, MG, RS and PR) account for 77% of customers.

4. Impact on Economy: Analyse the money movement by e-commerce by looking at order prices, freight, and others.
    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

```
WITH cost_of_orders
     AS (SELECT Sum(oi.freight_value)             AS freight_cost,
                Sum(p.payment_value)              AS payment_value,
                Year(o.order_purchase_timestamp) AS Year_of_purchase,
                Count(DISTINCT oi.order_id)       AS Num_of_orders
         FROM   store_sql.orders o
                JOIN store_sql.customers c
                  ON o.customer_id = c.customer_id
                LEFT JOIN store_sql.order_items oi
                       ON o.order_id = oi.order_id
                LEFT JOIN store_sql.payments p
                       ON p.order_id = o.order_id
         WHERE  o.order_status = 'delivered'
            AND Month(o.order_purchase_timestamp) < 9
            AND Year(o.order_purchase_timestamp) IN ( 2017, 2018 )
         GROUP  BY Year(o.order_purchase_timestamp))
SELECT year_of_purchase,
       Round(( payment_value / num_of_orders ), 2) AS 'Payment value per ord
er',
       Round(( ( payment_value / num_of_orders ) / Lag(
               ( payment_value / num_of_orders ), 1,
                 0)
               OVER(
                 ORDER BY year_of_purchase) - 1 )
             * 100, 2)                                  AS
       '% change in payment value per order',
       Round(( freight_cost / num_of_orders ), 2)  AS
       'Cost of freight per order',
       Round(( ( freight_cost / num_of_orders ) / Lag((
               freight_cost / num_of_orders ),
                                                   1, 0)
                                                     OVER(
                                                       ORDER BY year_of_purch
ase)
               - 1 )
             * 100, 2)                                  AS
       '% change in cost of freight per order'
FROM   cost_of_orders
```

Result:

Year_of_purchase	Payment value per order	% change in payment value per order	Cost of freight per order	% change in cost of freight per order
2017	195.91	NULL	23.11	NULL
2018	205.39	4.84	24.33	5.27

Payment value per order has increased by 4.8% and cost of freight per order has increased by 5.3% between 2017 and 2018.

    2. Mean & Sum of price and freight value by customer state

```
WITH cost_of_orders
     AS (SELECT Sum(oi.freight_value) AS freight_cost,
                Sum(p.payment_value)  AS payment_value,
```

```
                oi.order_id,
                c.customer_state
        FROM    store_sql.orders o
                JOIN store_sql.customers c
                  ON o.customer_id = c.customer_id
                LEFT JOIN store_sql.order_items oi
                       ON o.order_id = oi.order_id
                LEFT JOIN store_sql.payments p
                       ON p.order_id = o.order_id
        WHERE   o.order_status = 'delivered'
        GROUP   BY oi.order_id,
                   c.customer_state)
SELECT customer_state                AS State,
       Round(Sum(payment_value), 2)  AS 'Total Price',
       Round(Avg(payment_value), 2)  AS 'Mean of price',
       Round(Sum(freight_cost), 2)   AS 'Total Freight',
       Round(Avg(freight_cost), 2)   AS 'Mean of freight'
FROM   cost_of_orders
GROUP  BY customer_state
ORDER  BY Sum(payment_value) DESC
```

Result

State	Total Price	Mean of price	Total Freight	Mean of freight
SP	7403993.29	182.81	735951.62	18.17
RJ	2688933.90	217.73	313232.56	25.36
MG	2281229.16	200.92	276720.95	24.37
RS	1110976.47	207.85	138501.49	25.91
PR	1030822.39	209.39	120420.28	24.46
BA	773182.02	237.46	103868.48	31.90
SC	767093.97	216.33	90589.24	25.55
GO	493068.70	251.95	53465.45	27.32
DF	421374.86	202.58	51103.18	24.57
ES	398321.90	199.66	50642.45	25.38
PE	361028.07	226.63	59495.85	37.35

5. Analysis on sales, freight and delivery time

    1. Calculate days between purchasing, delivering and estimated delivery

```
SELECT o.order_id  AS 'Order ID',
    Datediff(o.order_delivered_customer_date, o.order_purchase_timestamp) AS
    'Actual delivery (days)',
    Datediff(o.order_estimated_delivery_date, o.order_purchase_timestamp) AS
    'Estimated delivery (days)'
FROM   store_sql.orders o
WHERE  o.order_status = 'delivered'
ORDER  BY o.order_id
```

Result:

Order ID	Actual delivery (days)	Estimated delivery (days)
00010242fe8c5a6d1ba2dd792cb16214	7	16
00018f77f2f0320c557190d7a144bdd3	16	19
000229ec398224ef6ca0657da4fc703e	8	22
00024acbcdf0a6daa1e931b038114c75	6	12
00042b26cf59d7ce69dfabb4e55b4fd9	25	41
00048cc3ae777c65dbb7d2a0634bc1ea	7	22
00054e8431b9d7675808bcb819fb4a32	8	25
000576fe39319847cbb9d288c5617fa6	5	21
0005a1a1728c9d785b8e2b08b904576c	10	10
0005f50442cb953dcd1d21e1fb923495	2	21
00061f2a7bc09da83e415a52dc8a4af1	5	16

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 o.order_id
    AS
    'Order ID',
    Datediff(o.order_delivered_customer_date, o.order_purchase_timestamp)
    AS
    'time_to_delivery',
    Datediff(o.order_delivered_customer_date,
    o.order_estimated_delivery_date) AS
    'diff_estimated_delivery'
FROM   store_sql.orders o
WHERE  o.order_status = 'delivered'
ORDER  BY o.order_id
```

Result:

Order ID	time_to_delivery	diff_estimated_delivery
00010242fe8c5a6d1ba2dd792cb16214	7	-9
00018f77f2f0320c557190d7a144bdd3	16	-3
000229ec398224ef6ca0657da4fc703e	8	-14
00024acbcdf0a6daa1e931b038114c75	6	-6
00042b26cf59d7ce69dfabb4e55b4fd9	25	-16
00048cc3ae777c65dbb7d2a0634bc1ea	7	-15
00054e8431b9d7675808bcb819fb4a32	8	-17
000576fe39319847cbb9d288c5617fa6	5	-16
0005a1a1728c9d785b8e2b08b904576c	10	0
0005f50442cb953dcd1d21e1fb923495	2	-19
00061f2a7bc09da83e415a52dc8a4af1	5	-11

3. Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery

```
WITH order_details
    AS (SELECT c.customer_state
               AS
                 'State',
               oi.order_id,
               oi.freight_value,
               Datediff(o.order_delivered_customer_date,
               o.order_purchase_timestamp)
                 AS
               'time_to_delivery',
               Datediff(o.order_delivered_customer_date,
               o.order_estimated_delivery_date) AS
               'diff_estimated_delivery'
        FROM   store_sql.orders o
               LEFT JOIN store_sql.order_items oi
                      ON o.order_id = oi.order_id
               JOIN store_sql.customers c
                 ON o.customer_id = c.customer_id
        WHERE  o.order_status = 'delivered')


SELECT state,
       Round(Sum(freight_value) / Count(DISTINCT order_id), 2)          AS
       'Mean of freight',
       Round(Sum(time_to_delivery) / Count(DISTINCT order_id), 2)       AS
       'Mean of time_to_delivery',
       Round(Sum(diff_estimated_delivery) / Count(DISTINCT order_id), 2) AS
       'Mean of diff_estimated_delivery'
FROM   order_details
GROUP  BY state
ORDER  BY state
```

Result:

State	Mean of freight	Mean of time_to_delivery	Mean of diff_estimated_delivery
AC	45.55	23.53	-23.86
AL	38.58	26.29	-9.40
AM	37.45	29.61	-22.41
AP	41.30	34.12	-22.24
BA	29.96	21.71	-12.42
CE	36.50	23.33	-12.38
DF	23.86	14.60	-13.81
ES	24.57	17.38	-11.87
GO	26.25	17.84	-14.30
MA	42.95	24.09	-11.05
MG	23.46	13.56	-15.18
MS	27.02	17.89	-12.99

4. Sort the data to get the following:

    1. Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5

Top 5 freight value

```
WITH order_details AS
(
	SELECT	c.customer_state AS 'State',
		oi.order_id,
		oi.freight_value,
		Datediff(o.order_delivered_customer_date , o.order_purchase_timestamp)	AS 'time_to_delivery',
		Datediff(o.order_delivered_customer_date , o.order_estimated_delivery_date) AS 'diff_estimated_deliv
ery'
	FROM	store_sql.orders o
	LEFT JOIN store_sql.order_items oi
	ON	o.order_id = oi.order_id
	JOIN	store_sql.customers c
	ON	o.customer_id = c.customer_id
	WHERE	o.order_status = 'delivered')


SELECT	state,
	Round(Sum(freight_value)	/ Count(DISTINCT order_id),2) AS 'Mean of freight',
	Round(Sum(time_to_delivery)	/ Count(DISTINCT order_id),2) AS 'Mean of time_to_delivery',
	Round(Sum(diff_estimated_delivery)/ Count(DISTINCT order_id),2) AS 'Mean of diff_estimated_delivery'
FROM	order_details
GROUP BY state
ORDER BY Sum(freight_value)/ Count(DISTINCT order_id) DESC limit 5
```

State	Mean of freight	Mean of time_to_delivery	Mean of diff_estimated_delivery
PB	48.84	23.29	-14.78
RR	48.34	31.61	-20.56
RO	46.43	22.08	-22.51
AC	45.55	23.53	-23.86
PI	42.98	21.22	-12.67

Bottom 5 by mean freight value

```
WITH order_details AS
(
	SELECT	c.customer_state AS 'State',
		oi.order_id,
		oi.freight_value,
		Datediff(o.order_delivered_customer_date , o.order_purchase_timestamp)	AS 'time_to_delivery',
		Datediff(o.order_delivered_customer_date , o.order_estimated_delivery_date) AS 'diff_estimated_deliv
ery'
	FROM	store_sql.orders o
	LEFT JOIN store_sql.order_items oi
	ON	o.order_id = oi.order_id
	JOIN	store_sql.customers c
	ON	o.customer_id = c.customer_id
	WHERE	o.order_status = 'delivered')


SELECT	state,
	Round(Sum(freight_value)	/ Count(DISTINCT order_id),2) AS 'Mean of freight',
	Round(Sum(time_to_delivery)	/ Count(DISTINCT order_id),2) AS 'Mean of time_to_delivery',
	Round(Sum(diff_estimated_delivery)/ Count(DISTINCT order_id),2) AS 'Mean of diff_estimated_delivery'
FROM	order_details
GROUP BY state
ORDER BY Sum(freight_value)/ Count(DISTINCT order_id) limit 5
```

State	Mean of freight	Mean of time_to_delivery	Mean of diff_estimated_delivery
SP	17.33	9.93	-12.85
MG	23.46	13.56	-15.18
PR	23.49	13.65	-15.47
DF	23.86	14.60	-13.81
RJ	23.95	17.26	-13.75

2. Top 5 states with highest/lowest average time to delivery

```
WITH order_details AS
(
	SELECT	c.customer_state AS 'State',
		oi.order_id,
		oi.freight_value,
		Datediff(o.order_delivered_customer_date , o.order_purchase_timestamp)	AS 'time_to_delivery',
		Datediff(o.order_delivered_customer_date , o.order_estimated_delivery_date) AS 'diff_estimated_deliv
ery'
	FROM	store_sql.orders o
	LEFT JOIN store_sql.order_items oi
	ON	o.order_id = oi.order_id
	JOIN	store_sql.customers c
	ON	o.customer_id = c.customer_id
	WHERE	o.order_status = 'delivered')


SELECT	state,
	Round(Sum(freight_value)	/ Count(DISTINCT order_id),2) AS 'Mean of freight',
	Round(Sum(time_to_delivery)	/ Count(DISTINCT order_id),2) AS 'Mean of time_to_delivery',
	Round(Sum(diff_estimated_delivery)/ Count(DISTINCT order_id),2) AS 'Mean of diff_estimated_delivery'
FROM	order_details
GROUP BY state
ORDER BY Sum(time_to_delivery)/ Count(DISTINCT order_id) DESC limit 5
```

State	Mean of freight	Mean of time_to_delivery	Mean of diff_estimated_delivery
AP	41.30	34.12	-22.24
RR	48.34	31.61	-20.56
AM	37.45	29.61	-22.41
PA	39.70	26.41	-15.88
AL	38.58	26.29	-9.40

Top 5 average time to deliver

```
WITH order_details AS
(
	SELECT	c.customer_state AS 'State',
		oi.order_id,
		oi.freight_value,
		Datediff(o.order_delivered_customer_date , o.order_purchase_timestamp)	AS 'time_to_delivery',
		Datediff(o.order_delivered_customer_date , o.order_estimated_delivery_date) AS 'diff_estimated_deliv
ery'
	FROM	store_sql.orders o
	LEFT JOIN store_sql.order_items oi
	ON	o.order_id = oi.order_id
	JOIN	store_sql.customers c
```

```
        ON      o.customer_id = c.customer_id
        WHERE   o.order_status = 'delivered')


SELECT  state,
        Round(Sum(freight_value)        / Count(DISTINCT order_id),2) AS 'Mean of freight',
        Round(Sum(time_to_delivery)     / Count(DISTINCT order_id),2) AS 'Mean of time_to_delivery',
        Round(Sum(diff_estimated_delivery)/ Count(DISTINCT order_id),2) AS 'Mean of diff_estimated_delivery'
FROM    order_details
GROUP BY state
ORDER BY Sum(time_to_delivery)/ Count(DISTINCT order_id) limit 5
```

State	Mean of freight	Mean of time_to_delivery	Mean of diff_estimated_delivery
SP	17.33	9.93	-12.85
MG	23.46	13.56	-15.18
PR	23.49	13.65	-15.47
DF	23.86	14.60	-13.81
RJ	23.95	17.26	-13.75

3. Top 5 states where delivery is really fast/ not so fast compared to estimated date

Top 5 states really fast to deliver

```
WITH order_details AS
(
        SELECT  c.customer_state AS 'State',
                oi.order_id,
                oi.freight_value,
                Datediff( o.order_delivered_customer_date, o.order_purchase_timestamp )     AS 'time_to_delivery',
                Datediff( o.order_delivered_customer_date, o.order_estimated_delivery_date ) AS 'diff_estimated_delivery'
        FROM    store_sql.orders o
        LEFT JOIN store_sql.order_items oi
        ON      o.order_id = oi.order_id
        JOIN    store_sql.customers c
        ON      o.customer_id = c.customer_id
        WHERE   o.order_status = 'delivered' )


SELECT  state,
        Round( Sum(freight_value)        / Count(DISTINCT order_id), 2 ) AS 'Mean of freight',
        Round( Sum(time_to_delivery)     / Count(DISTINCT order_id), 2 ) AS 'Mean of time_to_delivery',
        Round( Sum(diff_estimated_delivery)/ Count(DISTINCT order_id), 2 ) AS 'Mean of diff_estimated_delivery'
FROM    order_details
GROUP BY state
ORDER BY Sum(diff_estimated_delivery)/ Count(DISTINCT order_id) limit 5
```

State	Mean of freight	Mean of time_to_delivery	Mean of diff_estimated_delivery
AC	45.55	23.53	-23.86
RO	46.43	22.08	-22.51
AM	37.45	29.61	-22.41
AP	41.30	34.12	-22.24
RR	48.34	31.61	-20.56

Top 5 really slow to deliver

```
WITH order_details AS
(
      SELECT    c.customer_state AS 'State',
            oi.order_id,
            oi.freight_value,
            Datediff(o.order_delivered_customer_date , o.order_purchase_timestamp)     AS 'time_to_delivery',
            Datediff(o.order_delivered_customer_date , o.order_estimated_delivery_date) AS 'diff_estimated_delivery'
      FROM      store_sql.orders o
      LEFT JOIN store_sql.order_items oi
      ON        o.order_id = oi.order_id
      JOIN      store_sql.customers c
      ON        o.customer_id = c.customer_id
      WHERE     o.order_status = 'delivered')


SELECT   state,
      Round(Sum(freight_value)         / Count(DISTINCT order_id),2) AS 'Mean of freight',
      Round(Sum(time_to_delivery)      / Count(DISTINCT order_id),2) AS 'Mean of time_to_delivery',
      Round(Sum(diff_estimated_delivery)/ Count(DISTINCT order_id),2) AS 'Mean of diff_estimated_delivery'
FROM     order_details
GROUP BY state
ORDER BY Sum(diff_estimated_delivery)/ Count(DISTINCT order_id) DESC limit 5
```

State	Mean of freight	Mean of time_to_delivery	Mean of diff_estimated_delivery
AL	38.58	26.29	-9.40
MA	42.95	24.09	-11.05
SE	40.94	23.98	-11.20
ES	24.57	17.38	-11.87
CE	36.50	23.33	-12.38

6. Payment type analysis:

1. Month over Month count of orders for different payment types

```
WITH payment_count AS
(
                  SELECT DISTINCT p.payment_type,
                                  o.order_id,
                                  date_sub(Date(order_purchase_timestamp), int
erval dayofmonth(date(order_purchase_timestamp)) - 1 day) AS month_purchased
                  FROM            store_sql.orders o
                  JOIN            store_sql.payments p
                  ON              p.order_id = o.order_id
                  WHERE           o.order_status = 'delivered')
SELECT    date_format(month_purchased,"%b %Y") AS month_yr_purchased,
          payment_type,
          count(1)
                                        AS num_of_orders,
          count(1) - lag(count(1), 1, 0) OVER(partition BY payment_type ORDER
 BY month_purchased)                    AS 'Month over month # of orders',
          round((count(1) / lag(count(1), 1, 0) OVER(partition BY payment_typ
e ORDER BY month_purchased)-1)*100,2) AS '% Change MoM # of orders'
FROM      payment_count
```

```
GROUP BY month_purchased,
         payment_type
ORDER BY payment_type,
         month_purchased
```

Result

month_yr_purchased	payment_type	Num_of_orders	Month over month # of orders	% Change MoM # of orders
Oct 2016	credit_card	208	208	NULL
Dec 2016	credit_card	1	-207	-99.52
Jan 2017	credit_card	541	540	54000.00
Feb 2017	credit_card	1249	708	130.87
Mar 2017	credit_card	1901	652	52.20
Apr 2017	credit_card	1761	-140	-7.36
May 2017	credit_card	2714	953	54.12
Jun 2017	credit_card	2362	-352	-12.97
Jul 2017	credit_card	2960	598	25.32
Aug 2017	credit_card	3174	214	7.23
Sep 2017	credit_card	3175	1	0.03
Oct 2017	credit_card	3402	227	7.15

2. Count of orders based on the no. of payment instalments

As only credit cards have an option for making payments through multiple instalments will use MAX(payment_installments) to get the number of payment instalments for each order.

```
WITH order_installments
     AS (SELECT o.order_id,
                Max(p.payment_installments) AS Num_of_insatllm
ents
         FROM   store_sql.orders o
                JOIN store_sql.payments p
                  ON p.order_id = o.order_id
         WHERE  o.order_status = 'delivered'
         GROUP  BY o.order_id)
SELECT num_of_insatllments,
       Count(1) AS Num_of_orders
FROM   order_installments
GROUP  BY num_of_insatllments
ORDER  BY num_of_insatllments
```

Result:

Num_of_insatllments	Num_of_orders
0	2
1	46814
2	12026
3	10133
4	6864
5	5083
6	3792
7	1559
8	4120
9	618