Business Case: Target SQL

Suman Debnath

Code :
https://github.com/debnsuma/dsml-2022/blob/main/DSML/case_study/01_SQL_Target/code.sql

Q1: IMPORT THE DATASET AND DO USUAL EXPLORATORY ANALYSIS STEPS LIKE CHECKING THE STRUCTURE & CHARACTERISTICS OF THE DATASET

1.1 Data type of columns in a table

```sql
-- Product column
SELECT column_name, data_type
FROM information_schema.columns
WHERE table_name = 'products';

-- Orders column
SELECT column_name, data_type
FROM information_schema.columns
WHERE table_name = 'orders';
```

Result 1 (9) | Result 2 (8)

column_name	data_type
product_width_cm	integer
product_height_cm	integer
product_length_cm	integer
product_weight_g	integer
product_photos_qty	integer
product_description_l...	integer
product_name_length	integer
product category	character varying
product_id	character varying

Result 1 (9) | Result 2 (8)

column_name	data_type
order_status	character varying
customer_id	character varying
order_id	character varying
order_estimated_deliv...	timestamp without time z...
order_delivered_cust...	timestamp without time z...
order_delivered_carri...	timestamp without time z...
order_approved_at	timestamp without time z...
order_purchase_time...	timestamp without time z...

1.2 Time period for which the data is given

```sql
SELECT
    MIN(order_purchase_timestamp) AS start_date,
    MAX(order_purchase_timestamp) AS last_date
FROM orders;
```

Result 1 (1)

start_date	last_date
2016-09-04 21:15:19	2018-10-17 17:30:18

1.3 Cities and States of customers ordered during the given period

```
SELECT  DISTINCT c.customer_city as city,  c.customer_state as state
FROM orders o
INNER JOIN customers c
    ON o.customer_id = c.customer_id
ORDER BY state, city
```

Result 1 (100)

city	state
brasileia	AC
cruzeiro do sul	AC
epitaciolandia	AC
manoel urbano	AC
porto acre	AC
rio branco	AC
senador guiomard	AC
xapuri	AC
agua branca	AL
anadia	AL
arapiraca	AL
atalaia	AL
barra de santo antonio	AL
barra de sao miguel	AL
batalha	AL
belem	AL

Q2: IN-DEPTH EXPLORATION:

2.1 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?

```
-- No. of orders in monthly baisis

WITH A AS (
    SELECT
        order_id,
        order_purchase_timestamp,
        EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
        EXTRACT(MONTH FROM order_purchase_timestamp) AS month
    FROM orders
)

SELECT
    year,
    month,
    COUNT(1) as no_of_orders,
    SUM(no_of_orders) OVER(PARTITION BY year ORDER BY month ROWS BETWEEN 1 PRECEDING A
FROM A
GROUP BY year, month
ORDER BY year, month
```

Result 1 (25)

year	month	no_of_orders	rolling_monthly_sales_...
2016	9	4	4
2016	10	324	328
2016	12	1	325
2017	1	800	800
2017	2	1780	2580
2017	3	2682	4462
2017	4	2404	5086
2017	5	3700	6104
2017	6	3245	6945
2017	7	4026	7271
2017	8	4331	8357
2017	9	4285	8616
2017	10	4631	8916
2017	11	7544	12175
2017	12	5673	13217
2018	1	7269	7269
2018	2	6728	13997

```sql
-- No. of orders year over year

WITH A AS (
    SELECT
        order_id,
        order_purchase_timestamp,
        EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
        EXTRACT(MONTH FROM order_purchase_timestamp) AS month
    FROM orders
)

SELECT
    year,
    COUNT(1) as no_of_orders
FROM A
GROUP BY year
```

Result 1 (3)

year	no_of_orders
2016	329
2017	45101
2018	54011

```sql
-- No. of orders year over year

WITH A AS (
    SELECT
        order_id,
        order_purchase_timestamp,
```

```
        EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
        EXTRACT(MONTH FROM order_purchase_timestamp) AS month
    FROM orders
),

B AS (
    SELECT
        A.*,
        P.payment_value
        FROM A
        INNER JOIN
            payments P
            ON A.order_id = P.order_id
    ),

C AS (
    SELECT
        year,
        month,
        SUM(payment_value) as total_sales
    FROM B
    GROUP BY year, month
    ORDER BY year, month
)

SELECT
    DISTINCT year,
    SUM(total_sales) OVER(PARTITION BY year) AS total_sales,
    AVG(total_sales) OVER(PARTITION BY year) AS avg_sales
FROM C
ORDER BY year
```

Result 1 (3)

year	total_sales	avg_sales
2016	59187	19729
2017	7227504	602292
2018	8673015	867301

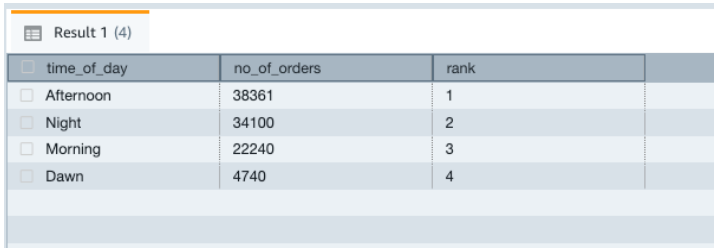
Points

- Year over year no. of orders are increasing
- Year end seems to have some high sales
- Nov and Dec seems to have high sales

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

```
WITH A AS (
    SELECT
        order_id,
        CASE
            WHEN DATE_PART('hour', order_purchase_timestamp) >= 0 AND DATE_PART('hour'
            WHEN DATE_PART('hour', order_purchase_timestamp) >= 6 AND DATE_PART('hour'
            WHEN DATE_PART('hour', order_purchase_timestamp) >= 12 AND DATE_PART('hour
            ELSE 'Night'
        END AS time_of_day
    FROM orders
)

SELECT
    time_of_day,
    COUNT(1) AS no_of_orders,
    RANK() OVER(ORDER BY no_of_orders DESC) AS rank
FROM A
GROUP BY time_of_day
ORDER BY no_of_orders DESC
```



Result 1 (4)

time_of_day	no_of_orders	rank
Afternoon	38361	1
Night	34100	2
Morning	22240	3
Dawn	4740	4

Points

- Afternoon and night time seems to the busiest time of the day when Brazilians order online on Target

Q3: EVOLUTION OF E-COMMERCE ORDERS IN THE BRAZIL REGION:

3.1 : Get month on month orders by states

```
-- Total sales month over month for each state
WITH A AS (
    SELECT
        o.order_id,
        o.order_purchase_timestamp,
        EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
        EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
        c.customer_city,
        c.customer_state,
        p.payment_value
    FROM orders o
```

```
    LEFT JOIN customers c ON o.customer_id = c.customer_id
    LEFT JOIN payments p ON o.order_id = p.order_id
)
SELECT
   customer_state,
   year,
   month,
   SUM(payment_value) AS total_Sales
FROM A
GROUP BY customer_state, year, month
ORDER BY customer_state, year, month
```

Result 1 (100)

customer_state	year	month	total_sales
AC	2017	1	723
AC	2017	2	596
AC	2017	3	529
AC	2017	4	1350
AC	2017	5	2379
AC	2017	6	509
AC	2017	7	792
AC	2017	8	764
AC	2017	9	2026
AC	2017	10	955
AC	2017	11	944
AC	2017	12	1408
AC	2018	1	1155
AC	2018	2	298
AC	2018	3	163
AC	2018	4	586
AC	2018	5	1500
AC	2018	6	596
AC	2018	7	1768
AC	2018	8	603
AL	2016	10	129
AL	2017	1	453
AL	2017	2	2960
AL	2017	3	1416
AL	2017	4	5362

```
-- Total sales month over month for each state with month over month total sales

WITH A AS (
    SELECT
        o.order_id,
```

```
        o.order_purchase_timestamp,
        EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
        EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
        c.customer_city,
        c.customer_state,
        p.payment_value
    FROM orders o
    LEFT JOIN customers c ON o.customer_id = c.customer_id
    LEFT JOIN payments p ON o.order_id = p.order_id
),
B AS (
        SELECT
            customer_state,
            year,
            month,
            SUM(payment_value) AS total_Sales
            FROM A
            GROUP BY customer_state, year, month
            ORDER BY customer_state, year, month
)

SELECT
    customer_state,
    year,
    month,
    total_sales,
    SUM(total_sales) OVER(PARTITION BY customer_state ORDER BY year, month ROWS BETWE

FROM B
ORDER BY customer_state
```

Result 1 (100)

customer_state	year	month	total_sales	rolling_monthly_sales_each_year
AC	2017	1	723	723
AC	2017	2	596	1319
AC	2017	3	529	1125
AC	2017	4	1350	1879
AC	2017	5	2379	3729
AC	2017	6	509	2888
AC	2017	7	792	1301
AC	2017	8	764	1556
AC	2017	9	2026	2790
AC	2017	10	955	2981
AC	2017	11	944	1899
AC	2017	12	1408	2352
AC	2018	1	1155	2563
AC	2018	2	298	1453
AC	2018	3	163	461
AC	2018	4	586	749
AC	2018	5	1500	2086
AC	2018	6	596	2096
AC	2018	7	1768	2364
AC	2018	8	603	2371
AL	2016	10	129	129
AL	2017	1	453	582
AL	2017	2	2960	3413
AL	2017	3	1416	4376
AL	2017	4	5362	6778
AL	2017	5	5716	11078
AL	2017	6	1273	6989
AL	2017	7	3023	4296
AL	2017	8	9665	12688
AL	2017	9	5185	14850
AL	2017	10	6328	11513

3.2 : Distribution of customers across the states in Brazil

```sql
-- No. of customers in different states in desending order
WITH A AS (
    SELECT
        o.customer_id,
        c.customer_city,
        c.customer_state
    FROM orders o
    LEFT JOIN customers c ON o.customer_id = c.customer_id
)
SELECT
    customer_state,
    customer_city,
    COUNT(1) AS no_of_customers
FROM A
GROUP BY customer_state, customer_city
ORDER BY no_of_customers DESC
```

Result 1 (100)

customer_state	customer_city	no_of_customers
SP	sao paulo	15540
RJ	rio de janeiro	6882
MG	belo horizonte	2773
DF	brasilia	2131
PR	curitiba	1521
SP	campinas	1444
RS	porto alegre	1379
BA	salvador	1245
SP	guarulhos	1189
SP	sao bernardo do campo	938
RJ	niteroi	849
SP	santo andre	796
SP	osasco	746
SP	santos	713
GO	goiania	692
SP	sao jose dos campos	691
CE	fortaleza	654
SP	sorocaba	633
PE	recife	613
SC	florianopolis	570
SP	jundiai	565
SP	ribeirao preto	510
PA	belem	443
RJ	nova iguacu	442
SP	barueri	433
MG	juiz de fora	427
MG	contagem	426
RJ	sao goncalo	409

Points

- Sao Paulo, Rio De Janerio, and Belo Horizonte are the top 3 cities in Brazil who had the max number of customers

Q4: IMPACT ON ECONOMY: ANALYZE THE MONEY MOVEMENT BY E-COMMERCE BY LOOKING AT ORDER PRICES, FREIGHT AND OTHERS.

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

```sql
-- % Increase of sales over year from Jan 2017 to Sep 2018
WITH A AS (
    SELECT
        o.order_id,
        o.order_purchase_timestamp,
        EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
        EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
        p.payment_value
    FROM orders o
    LEFT JOIN payments p ON o.order_id = p.order_id
    WHERE o.order_purchase_timestamp >= '2017-01-01' AND
          o.order_purchase_timestamp < '2018-09-01'
),
B AS (
        SELECT
            year,
            SUM(payment_value) AS total_Sales
            FROM A
            GROUP BY year
)
SELECT
    *,
    ((total_Sales - LAG(total_Sales) OVER (ORDER BY year)) / LAG(total_Sales) OVER (O
FROM B
ORDER BY year
```

Result 1 (2)

year	total_sales	percentage_increase
2017	7227504	NULL
2018	8667994	19.93

```sql
-- Month over month total sales from Jan 2017 to Sep 2018
WITH A AS (
    SELECT
        o.order_id,
        o.order_purchase_timestamp,
        EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
        EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
        p.payment_value
    FROM orders o
    LEFT JOIN payments p ON o.order_id = p.order_id
    WHERE o.order_purchase_timestamp >= '2017-01-01' AND
          o.order_purchase_timestamp < '2018-09-01'
),
B AS (
        SELECT
            year,
```

```
            month,
            SUM(payment_value) AS total_Sales
            FROM A
            GROUP BY year, month
            ORDER BY year, month
)

SELECT *
FROM B;
```

Result 1 (20)

year	month	total_sales
2017	1	138033
2017	2	290950
2017	3	448393
2017	4	416487
2017	5	591137
2017	6	509746
2017	7	590449
2017	8	672295
2017	9	725648
2017	10	777463
2017	11	1191227
2017	12	875676
2018	1	1111501
2018	2	989263
2018	3	1156084
2018	4	1157374
2018	5	1150491
2018	6	1020808
2018	7	1063355
2018	8	1019118

4.2: Mean & Sum of price and freight value by customer state

```
-- Mean & Sum of price and freight value by customer state
```

```sql
WITH A AS (
    SELECT o.order_id, o.customer_id, oi.freight_value, oi.price, c.customer_state
    FROM orders o
    LEFT JOIN order_items oi ON o.order_id = oi.order_id
    LEFT JOIN customers c ON o.customer_id = c.customer_id
)

SELECT
    customer_state,
    SUM(freight_value) as total_freight_value,
    AVG(freight_value) as avg_freight_value,
    SUM(price) as total_price,
    AVG(price) as avg_price
    FROM A
    GROUP BY customer_state
    ORDER BY customer_state
```

Result 1 (27)

customer_state	total_freight_value	avg_freight_value	total_price	avg_price
AC	3643	39	15923	173
AL	15701	35	80049	180
AM	5404	32	22256	134
AP	2752	33	13428	163
BA	98207	25	509101	134
CE	47640	32	226398	153
DF	49620	20	301159	125
ES	48808	21	273709	121
GO	52123	22	293172	125
MA	31142	37	119148	144
MG	265083	20	1577418	120
MS	18797	22	116335	142
MT	29167	27	155803	147
PA	38174	35	178291	165
PB	25436	42	114944	190
PE	58542	32	261743	144
PI	20953	38	86618	159
PR	115394	20	679636	118
RJ	299241	20	1815288	124
RN	18614	35	82724	156
RO	11284	40	45982	165
RR	2205	42	7797	149
RS	132950	21	746528	119
SC	87839	21	518071	124
SE	13917	36	58700	152
SP	693616	14	5174261	109
TO	11578	36	49451	156

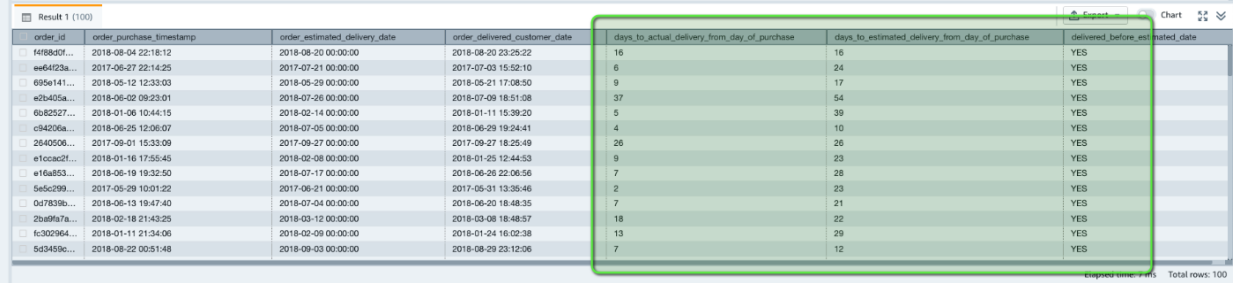
Q5: ANALYSIS ON SALES, FREIGHT AND DELIVERY TIME

5.1 Calculate days between purchasing, delivering and estimated delivery

```sql
/*
Calculate days between
    - purchasing date and actual delivering date
```

```
    - purchasing date and estimated delivery date
*/
SELECT
    order_id,
    order_purchase_timestamp,
    order_estimated_delivery_date,
    order_delivered_customer_date,
    DATEDIFF(day, order_purchase_timestamp, order_delivered_customer_date) AS days_to_
    DATEDIFF(day, order_purchase_timestamp, order_estimated_delivery_date) AS days_to_
    CASE
        WHEN days_to_actual_delivery_from_day_of_purchase <= days_to_estimated_deliver
    ELSE 'NO'
    END AS delivered_before_estimated_date

FROM orders
```



Result 1 (100) — Export — Chart

order_id	order_purchase_timestamp	order_estimated_delivery_date	order_delivered_customer_date	days_to_actual_delivery_from_day_of_purchase	days_to_estimated_delivery_from_day_of_purchase	delivered_before_estimated_date
f4f88d0f...	2018-08-04 22:18:12	2018-08-20 00:00:00	2018-08-20 23:25:22	16	16	YES
ee64f23a...	2017-06-27 22:14:25	2017-07-21 00:00:00	2017-07-03 15:52:10	6	24	YES
695e141...	2018-05-12 12:33:03	2018-05-29 00:00:00	2018-05-21 17:08:50	9	17	YES
e2b405a...	2018-06-02 09:23:01	2018-07-26 00:00:00	2018-07-09 18:51:08	37	54	YES
6b82527...	2018-01-06 10:44:15	2018-02-14 00:00:00	2018-01-11 15:39:20	5	39	YES
c94206a...	2018-06-25 12:06:07	2018-07-05 00:00:00	2018-06-29 19:24:41	4	10	YES
2640506...	2017-09-01 15:33:09	2017-09-27 00:00:00	2017-09-27 18:25:49	26	26	YES
e1ccac2f...	2018-01-16 17:55:45	2018-02-08 00:00:00	2018-01-25 12:44:53	9	23	YES
e16a853...	2018-06-19 19:32:50	2018-07-17 00:00:00	2018-06-26 22:06:56	7	28	YES
5e5c299...	2017-05-29 10:01:22	2017-06-21 00:00:00	2017-05-31 13:35:46	2	23	YES
0d7839b...	2018-06-13 19:47:40	2018-07-04 00:00:00	2018-06-20 18:48:35	7	21	YES
2ba9fa7a...	2018-02-18 21:43:25	2018-03-12 00:00:00	2018-03-08 18:48:57	18	22	YES
fc302964...	2018-01-11 21:34:06	2018-02-09 00:00:00	2018-01-24 16:02:38	13	29	YES
5d3459c...	2018-08-22 00:51:48	2018-09-03 00:00:00	2018-08-29 23:12:06	7	12	YES

Elapsed time: 74 ms — Total rows: 100

5.2 Find time_to_delivery & diff_estimated_delivery. Formula for the same given below:

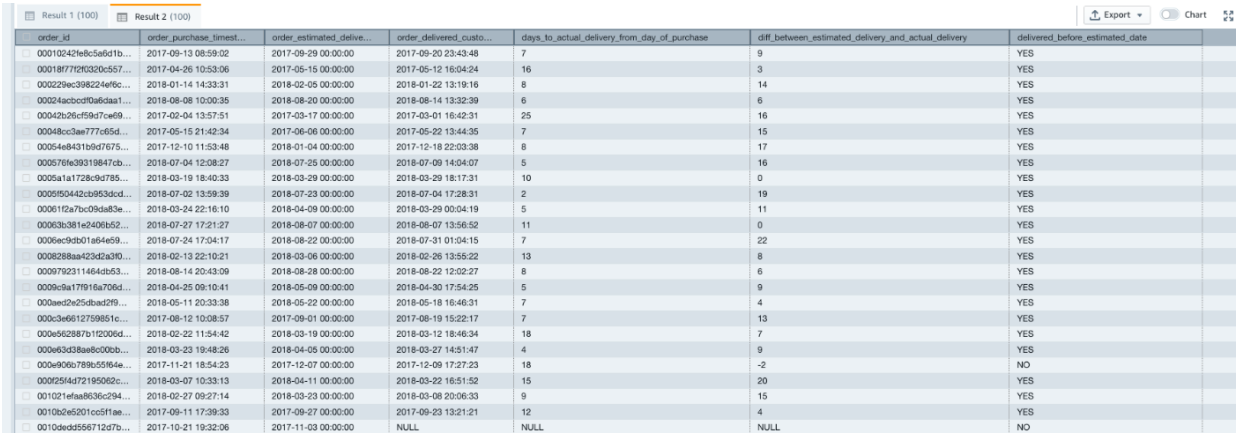
- time_to_delivery = order_purchase_timestamp-order_delivered_customer_date
- diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date

```
/*
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
*/

SELECT
    order_id,
    order_purchase_timestamp,
    order_estimated_delivery_date,
    order_delivered_customer_date,
    DATEDIFF(day, order_purchase_timestamp, order_delivered_customer_date) AS days_to_
    DATEDIFF(day, order_delivered_customer_date, order_estimated_delivery_date) AS dif
    CASE
        WHEN diff_between_estimated_delivery_and_actual_delivery >= 0  THEN 'YES'
    ELSE 'NO'
    END AS delivered_before_estimated_date
```

```
FROM orders
ORDER BY order_id ;
```



Result 1 (100) | Result 2 (100) | Export | Chart

order_id	order_purchase_timest...	order_estimated_delive...	order_delivered_custo...	days_to_actual_delivery_from_day_of_purchase	diff_between_estimated_delivery_and_actual_delivery	delivered_before_estimated_date
00010242fe8c5a6d1b...	2017-09-13 08:59:02	2017-09-29 00:00:00	2017-09-20 23:43:48	7	9	YES
00018f77f2f0320c557...	2017-04-26 10:53:06	2017-05-15 00:00:00	2017-05-12 16:04:24	16	3	YES
000229ec398224ef6c...	2018-01-14 14:33:31	2018-02-05 00:00:00	2018-01-22 13:19:16	8	14	YES
00024acbcdf0a6daa1...	2018-08-08 10:00:35	2018-08-20 00:00:00	2018-08-14 13:32:39	6	6	YES
00042b26cf59d7ce69...	2017-02-04 13:57:51	2017-03-17 00:00:00	2017-03-01 16:42:31	25	16	YES
00048cc3ae777c65d...	2017-05-15 21:42:34	2017-06-06 00:00:00	2017-05-22 13:44:35	7	15	YES
00054e8431b9d7675...	2017-12-10 11:53:48	2018-01-04 00:00:00	2017-12-18 22:03:38	8	17	YES
000576fe39319847cb...	2018-07-04 12:08:27	2018-07-25 00:00:00	2018-07-09 14:04:07	5	16	YES
0005a1a1728c9d785...	2018-03-19 18:40:33	2018-03-29 00:00:00	2018-03-29 18:17:31	10	0	YES
0005f50442cb953dcd...	2018-07-02 13:59:39	2018-07-23 00:00:00	2018-07-04 17:28:31	2	19	YES
00061f2a7bc09da83e...	2018-03-24 22:16:10	2018-04-09 00:00:00	2018-03-29 00:04:19	5	11	YES
00063b381e2406b52...	2018-07-27 17:21:27	2018-08-07 00:00:00	2018-08-07 13:56:52	11	0	YES
0006ec9db01a64e59...	2018-07-24 17:04:17	2018-08-22 00:00:00	2018-07-31 01:04:15	7	22	YES
0008288aa423d2a3f0...	2018-02-13 22:10:21	2018-03-06 00:00:00	2018-02-26 13:55:22	13	8	YES
000979231146 4db53...	2018-08-14 20:43:09	2018-08-28 00:00:00	2018-08-22 12:02:27	8	6	YES
0009c9a17f916a706d...	2018-04-25 09:10:41	2018-05-09 00:00:00	2018-04-30 17:54:25	5	9	YES
000aed2e25dbad2f9...	2018-05-11 20:33:38	2018-05-22 00:00:00	2018-05-18 16:46:31	7	4	YES
000c3e66127598 51c...	2017-08-12 10:08:57	2017-09-01 00:00:00	2017-08-19 15:22:17	7	13	YES
000e562887b1f2006d...	2018-02-22 11:54:42	2018-03-19 00:00:00	2018-03-12 18:46:34	18	7	YES
000e63d38ae8c00bb...	2018-03-23 19:48:26	2018-04-05 00:00:00	2018-03-27 14:51:47	4	9	YES
000e906b789b55f64e...	2017-11-21 18:54:23	2017-12-07 00:00:00	2017-12-09 17:27:23	18	-2	NO
000f25f4d72195062c...	2018-03-07 10:33:13	2018-04-11 00:00:00	2018-03-22 16:51:52	15	20	YES
001021efaa8636c294...	2018-02-27 09:27:14	2018-03-23 00:00:00	2018-03-08 20:06:33	9	15	YES
0010b2e5201cc5f1ae...	2017-09-11 17:39:33	2017-09-27 00:00:00	2017-09-23 13:21:21	12	4	YES
0010dedd556712d7b...	2017-10-21 19:32:06	2017-11-03 00:00:00	NULL	NULL	NULL	NO

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

```
/*
Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_deli
*/
WITH A AS (
    SELECT
        o.order_id,
        o.order_purchase_timestamp,
        o.order_estimated_delivery_date,
        o.order_delivered_customer_date,
        DATEDIFF(day, order_purchase_timestamp, order_delivered_customer_date) AS days
        DATEDIFF(day, order_delivered_customer_date, order_estimated_delivery_date) AS
        c.customer_state,
        oi.freight_value
    FROM orders o
    LEFT JOIN order_items oi ON o.order_id = oi.order_id
    LEFT JOIN customers c ON o.customer_id = c.customer_id
)
SELECT
    customer_state,
    AVG(freight_value) AS avg_freight_value,
    AVG(days_to_actual_delivery_from_day_of_purchase) AS mean_time_to_delivery,
    AVG(diff_between_estimated_delivery_and_actual_delivery) AS diff_estimated_deliver
FROM A
GROUP BY customer_state
ORDER BY customer_state
```

Result 1 (27)

customer_state	avg_freight_value	mean_time_to_delivery	diff_estimated_delivery
AC	39	20	20
AL	35	24	8
AM	32	26	19
AP	33	28	18
BA	25	19	10
CE	32	20	11
DF	20	12	12
ES	21	15	10
GO	22	15	12
MA	37	21	9
MG	20	11	13
MS	22	15	11
MT	27	17	14
PA	35	23	14
PB	42	20	13
PE	32	18	13
PI	38	19	11
PR	20	11	13
RJ	20	15	12
RN	35	19	13
RO	40	19	20
RR	42	28	18
RS	21	15	14
SC	21	14	11
SE	36	21	10
SP	14	8	11
TO	36	17	12

5.4 to 5.7 : Sort the data to get the following:

- Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5
- Top 5 states with highest/lowest average time to delivery
- Top 5 states where delivery is really fast/ not so fast compared to estimated date

```
-- Top 5 states with highest average freight value - sort in desc/asc limit 5
WITH A AS (
    SELECT
        o.order_id,
        o.order_purchase_timestamp,
        o.order_estimated_delivery_date,
        o.order_delivered_customer_date,
        DATEDIFF(day, order_purchase_timestamp, order_delivered_customer_date) AS days
        DATEDIFF(day, order_delivered_customer_date, order_estimated_delivery_date) AS
```

```
        c.customer_state,
        oi.freight_value
    FROM orders o
    LEFT JOIN order_items oi ON o.order_id = oi.order_id
    LEFT JOIN customers c ON o.customer_id = c.customer_id
)
SELECT
    customer_state,
    AVG(freight_value) AS avg_freight_value,
    AVG(days_to_actual_delivery_from_day_of_purchase) AS mean_time_to_delivery,
    AVG(diff_between_estimated_delivery_and_actual_delivery) AS diff_estimated_deliver
FROM A
GROUP BY customer_state
ORDER BY avg_freight_value DESC
LIMIT 5
```

Result 1 (5)

customer_state	avg_freight_value	mean_time_to_delivery	diff_estimated_delivery
RR	42	28	18
PB	42	20	13
RO	40	19	20
AC	39	20	20
PI	38	19	11

```
-- Top 5 states with lowest average freight value - sort in desc/asc limit 5
WITH A AS (
    SELECT
        o.order_id,
        o.order_purchase_timestamp,
        o.order_estimated_delivery_date,
        o.order_delivered_customer_date,
        DATEDIFF(day, order_purchase_timestamp, order_delivered_customer_date) AS days
        DATEDIFF(day, order_delivered_customer_date, order_estimated_delivery_date) AS
        c.customer_state,
        oi.freight_value
    FROM orders o
    LEFT JOIN order_items oi ON o.order_id = oi.order_id
    LEFT JOIN customers c ON o.customer_id = c.customer_id
)
SELECT
    customer_state,
    AVG(freight_value) AS avg_freight_value,
    AVG(days_to_actual_delivery_from_day_of_purchase) AS mean_time_to_delivery,
    AVG(diff_between_estimated_delivery_and_actual_delivery) AS diff_estimated_deliver
FROM A
GROUP BY customer_state
ORDER BY avg_freight_value
LIMIT 5
```

Result 1 (5)

customer_state	avg_freight_value	mean_time_to_delivery	diff_estimated_delivery
SP	14	8	11
DF	20	12	12
PR	20	11	13
RJ	20	15	12
MG	20	11	13

```sql
-- Top 5 states with highest average time to delivery

WITH A AS (
    SELECT
        o.order_id,
        o.order_purchase_timestamp,
        o.order_estimated_delivery_date,
        o.order_delivered_customer_date,
        DATEDIFF(day, order_purchase_timestamp, order_delivered_customer_date) AS days
        DATEDIFF(day, order_delivered_customer_date, order_estimated_delivery_date) AS
        c.customer_state,
        oi.freight_value
    FROM orders o
    LEFT JOIN order_items oi ON o.order_id = oi.order_id
    LEFT JOIN customers c ON o.customer_id = c.customer_id
)
SELECT
    customer_state,
    AVG(freight_value) AS avg_freight_value,
    AVG(days_to_actual_delivery_from_day_of_purchase) AS mean_time_to_delivery,
    AVG(diff_between_estimated_delivery_and_actual_delivery) AS diff_estimated_deliver
FROM A
GROUP BY customer_state
ORDER BY mean_time_to_delivery DESC
LIMIT 5
```

Result 1 (5)

customer_state	avg_freight_value	mean_time_to_delivery	diff_estimated_delivery
AP	33	28	18
RR	42	28	18
AM	32	26	19
AL	35	24	8
PA	35	23	14

```sql
-- Top 5 states with lowest average time to delivery
WITH A AS (
    SELECT
        o.order_id,
        o.order_purchase_timestamp,
```

```
        o.order_estimated_delivery_date,
        o.order_delivered_customer_date,
        DATEDIFF(day, order_purchase_timestamp, order_delivered_customer_date) AS days
        DATEDIFF(day, order_delivered_customer_date, order_estimated_delivery_date) AS
        c.customer_state,
        oi.freight_value
    FROM orders o
    LEFT JOIN order_items oi ON o.order_id = oi.order_id
    LEFT JOIN customers c ON o.customer_id = c.customer_id
)
SELECT
    customer_state,
    AVG(freight_value) AS avg_freight_value,
    AVG(days_to_actual_delivery_from_day_of_purchase) AS mean_time_to_delivery,
    AVG(diff_between_estimated_delivery_and_actual_delivery) AS diff_estimated_deliver
FROM A
GROUP BY customer_state
ORDER BY mean_time_to_delivery
LIMIT 5
```

Result 1 (5)

customer_state	avg_freight_value	mean_time_to_delivery	diff_estimated_delivery
SP	14	8	11
PR	20	11	13
MG	20	11	13
DF	20	12	12
SC	21	14	11

```
-- Top 5 states where delivery is really fast compared to estimated date
WITH A AS (
    SELECT
        o.order_id,
        o.order_purchase_timestamp,
        o.order_estimated_delivery_date,
        o.order_delivered_customer_date,
        DATEDIFF(day, order_purchase_timestamp, order_delivered_customer_date) AS days
        DATEDIFF(day, order_delivered_customer_date, order_estimated_delivery_date) AS
        c.customer_state,
        oi.freight_value
    FROM orders o
    LEFT JOIN order_items oi ON o.order_id = oi.order_id
    LEFT JOIN customers c ON o.customer_id = c.customer_id
)

SELECT
    customer_state,
    AVG(days_to_actual_delivery_from_day_of_purchase) AS mean_time_to_delivery,
    AVG(diff_between_estimated_delivery_and_actual_delivery) AS diff_estimated_deliver
FROM A
GROUP BY customer_state
```

```
ORDER BY diff_estimated_delivery
LIMIT 5
```

Result 1 (5)

customer_state	mean_time_to_delivery	diff_estimated_delivery
AL	24	8
MA	21	9
BA	19	10
SE	21	10
ES	15	10

```
-- Top 5 states where delivery is really not so fast compared to estimated date
WITH A AS (
    SELECT
        o.order_id,
        o.order_purchase_timestamp,
        o.order_estimated_delivery_date,
        o.order_delivered_customer_date,
        DATEDIFF(day, order_purchase_timestamp, order_delivered_customer_date) AS days
        DATEDIFF(day, order_delivered_customer_date, order_estimated_delivery_date) AS
        c.customer_state,
        oi.freight_value
    FROM orders o
    LEFT JOIN order_items oi ON o.order_id = oi.order_id
    LEFT JOIN customers c ON o.customer_id = c.customer_id
)

SELECT
    customer_state,
    AVG(days_to_actual_delivery_from_day_of_purchase) AS mean_time_to_delivery,
    AVG(diff_between_estimated_delivery_and_actual_delivery) AS diff_estimated_deliver
FROM A
GROUP BY customer_state
ORDER BY diff_estimated_delivery DESC
LIMIT 5
```

Result 1 (5)

customer_state	mean_time_to_delivery	diff_estimated_delivery
RO	19	20
AC	20	20
AM	26	19
AP	28	18
RR	28	18

Q6: PAYMENT TYPE ANALYSIS:

- Month over Month count of orders for different payment types
- Count of orders based on the no. of payment installments

6.1 Month over Month count of orders for different payment types

```
-- Month over Month count of orders for different payment types
SELECT *
FROM orders;
SELECT *
FROM payments
WITH A AS (
    SELECT
        o.order_id,
        o.order_purchase_timestamp,
        EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
        EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
        p.payment_type
    FROM orders o
    LEFT JOIN payments p ON o.order_id = p.order_id
)
SELECT
    payment_type,
    year,
    month,
    COUNT(1) AS no_of_orders
FROM A
GROUP BY payment_type, year, month
ORDER BY payment_type, year, month
```

Result 1 (91)

payment_type	year	month	no_of_orders
UPI	2016	10	63
UPI	2017	1	197
UPI	2017	2	398
UPI	2017	3	590
UPI	2017	4	496
UPI	2017	5	772
UPI	2017	6	707
UPI	2017	7	845
UPI	2017	8	938
UPI	2017	9	903
UPI	2017	10	993
UPI	2017	11	1509
UPI	2017	12	1160
UPI	2018	1	1518
UPI	2018	2	1325
UPI	2018	3	1352
UPI	2018	4	1287
UPI	2018	5	1263
UPI	2018	6	1100
UPI	2018	7	1229
UPI	2018	8	1139
credit_card	2016	9	3
credit_card	2016	10	254
credit_card	2016	12	1
credit_card	2017	1	583

6.2 Count of orders based on the no. of payment installments

```sql
-- Count of orders based on the no. of payment installments
WITH A AS (
    SELECT
        o.order_id,
        o.order_purchase_timestamp,
        EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
        EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
        p.payment_type,
        p.payment_installments
    FROM orders o
    LEFT JOIN payments p ON o.order_id = p.order_id
)
SELECT
    payment_installments,
    COUNT(1) AS no_of_orders
FROM A
GROUP BY payment_installments
ORDER BY payment_installments
```

Result 1 (25)

payment_installments	no_of_orders
0	2
1	52546
2	12413
3	10461
4	7098
5	5239
6	3920
7	1626
8	4268
9	644
10	5328
11	23
12	133
13	16
14	15
15	74
16	5
17	8
18	27
20	17
21	3
22	1
23	1
24	18