Data Cleaning

```
use [E-Commerce];
```

-- TABLE CUSTOMERS

```
select * from customers;
```

-- Checking NULL VALUE in Customers table.

```
SELECT
        SUM(CASE WHEN customer_id IS NULL THEN 1 ELSE 0 END) as Customer_id_null_count,
        SUM(CASE WHEN customer_unique_id IS NULL THEN 1 ELSE 0 END) AS Unique_id_null_count,
        SUM(CASE WHEN customer_zip_code_prefix IS NULL THEN 1 ELSE 0 END) AS
Zip_code_null_count,
        SUM(CASE WHEN Customer_city IS NULL THEN 1 ELSE 0 END ) AS City_null_count,
        SUM(CASE WHEN Customer_state IS NULL THEN 1 ELSE 0 END) AS State_null_count
FROM CUSTOMERS


 select count(distinct(customer_zip_code_prefix))
 from customers;
 -- IT HAS NO NULL VALUES
```

-- TABLE GEO_LOCATION

```
SELECT * FROM GEO_LOCATION;
```

-- Checking for NULL values in the GEO_LOCATION table

```
SELECT
   SUM(CASE WHEN geolocation_zip_code_prefix IS NULL THEN 1 ELSE 0 END) AS zip_code_null_count,
   SUM(CASE WHEN geolocation_lat IS NULL THEN 1 ELSE 0 END) AS lat_null_count,
   SUM(CASE WHEN geolocation_lng IS NULL THEN 1 ELSE 0 END) AS lng_null_count,
   SUM(CASE WHEN geolocation_city IS NULL THEN 1 ELSE 0 END) AS city_null_count,
   SUM(CASE WHEN geolocation_state IS NULL THEN 1 ELSE 0 END) AS state_null_count
FROM GEO_LOCATION;
```

-- SELECTING GEOLOCATION LAT WHERE VALUE IS NULL

```
SELECT * FROM GEO_LOCATION
```

```
WHERE geolocation_LAT IS NULL;
```

```
-- UPTADING null values with the calculated average for those  states
UPDATE GEO_LOCATION
SET geolocation_lat = (
                        SELECT AVG(geolocation_lat)
                        FROM GEO_LOCATION AS gl
                        WHERE gl.geolocation_state = GEO_LOCATION.geolocation_state
                         AND gl.geolocation_lat IS NOT NULL
                        )
WHERE geolocation_lat IS NULL;
```

```
-- TABLE ORDER_ITEMS

SELECT * FROM ORDER_ITEMS;
-- Checking for NULL values in the GEO_LOCATION table

SELECT
        SUM(CASE WHEN ORDER_ID IS NULL THEN 1 ELSE 0 END) AS Order_id_null_COUNT,
        SUM(CASE WHEN Order_item_id IS NULL THEN 1 ELSE 0 END) AS Item_id_null_count,
        SUM(CASE WHEN product_id IS NULL THEN 1 ELSE 0 END) AS Product_id_null_count,
        SUM(CASE WHEN seller_id IS NULL THEN 1 ELSE 0 END) AS seller_id_null_count,
        SUM(CASE WHEN shipping_limit_date IS NULL THEN 1 ELSE 0 END ) AS Shipping_null_count,
        SUM(CASE WHEN price IS NULL THEN 1 ELSE 0 END) AS Price_null_count,
        SUM(CASE WHEN freight_value IS NULL THEN 1 ELSE 0 END) AS value_null_count
FROM ORDER_ITEMS;
```

```
    -- ORDER_ITEMS TABLE IS CLEAN
```

```
-- TABLE ORDER_PAYMENTS
```

```
SELECT * FROM ORDER_PAYMENTS;

-- Checking for NULL values in the Order_payments table

SELECT

        SUM(CASE WHEN ORDER_ID IS NULL THEN 1 ELSE 0 END) AS Order_id_null_COUNT,

        SUM(CASE WHEN payment_sequential IS NULL THEN 1 ELSE 0 END) sequential_null_count,

        SUM(CASE WHEN payment_type IS NULL THEN 1 ELSE 0 END) AS payment_type_null_count,

        SUM(CASE WHEN payment_installments IS NULL THEN 1 ELSE 0 END) AS
payment_installments_null_count,

        SUM(CASE WHEN payment_value IS NULL THEN 1 ELSE 0 END ) AS payment_value_null_count


FROM ORDER_PAYMENTS;


    -- ORDER_PAYMENT TABLE IS CLEAN


                                -- TABLE ORDER_REVIEW_RATINGS

SELECT * FROM ORDER_REVIEW_RATINGS;



-- Checking for NULL values in the Order_review_rating table

SELECT

        SUM(CASE WHEN review_id IS NULL THEN 1 ELSE 0 END) AS review_id_null_COUNT,

        SUM(CASE WHEN order_id IS NULL THEN 1 ELSE 0 END) order_id_null_count,

        SUM(CASE WHEN review_score IS NULL THEN 1 ELSE 0 END) AS review_score_null_count,

        SUM(CASE WHEN review_creation_date IS NULL THEN 1 ELSE 0 END) AS review_creation_null_count,

        SUM(CASE WHEN review_answer_timestamp IS NULL THEN 1 ELSE 0 END ) AS review_answer_null_count


FROM ORDER_REVIEW_RATINGS;


  -- ORDER_REVIEW_RATINGS TABLE IS CLEAN
```

-- TABLE ORDERS

```
SELECT * FROM orders;
```

-- Checking for NULL values in the Orders table

```
SELECT
        SUM(CASE WHEN order_id IS NULL THEN 1 ELSE 0 END) AS order_id_null_COUNT,
        SUM(CASE WHEN customer_id IS NULL THEN 1 ELSE 0 END) customer_id_null_count,
        SUM(CASE WHEN order_status IS NULL THEN 1 ELSE 0 END) AS order_status_null_count,
        SUM(CASE WHEN order_purchase_timestamp IS NULL THEN 1 ELSE 0 END) AS
order_purchese_time_null_count,
        SUM(CASE WHEN order_approved_at IS NULL THEN 1 ELSE 0 END ) AS order_approved_at_null_count,
        SUM(CASE WHEN order_delivered_carrier_date IS NULL THEN 1 ELSE 0 END ) AS
order_delivered_null_count,
        SUM(CASE WHEN order_delivered_customer_date IS NULL THEN 1 ELSE 0 END ) AS
order_DCD_NULL_COUNT,
        SUM(CASE WHEN order_estimated_delivery_date IS NULL THEN 1 ELSE 0 END ) AS Order_EDD_null_count
FROM ORDERS;


select * from ORDERS
where order_delivered_carrier_date is null
```

-- IN THE ORDER_DELIVERED_CARRIER_DATE AND ORDER_DELIVERED_CUSTOMER_DATE HAVE SOME NULL VALUES BECOUSE MOST OF THE ORDER UNABAILABLE, PROCESSING AND CANCELED

-- I AM NOT CHANGING THESE DATA BECAUSE I DON'T THING IT NEED TO BE CHANGE.

-- TABLE PRODUCTS

```
SELECT * FROM PRODUCTS;
```

-- Checking for NULL values in the Products table

```
SELECT
        SUM(CASE WHEN product_id IS NULL THEN 1 ELSE 0 END) AS product_id_null_COUNT,
        SUM(CASE WHEN product_category_name IS NULL THEN 1 ELSE 0 END) PCN_null_count,
        SUM(CASE WHEN product_name_length IS NULL THEN 1 ELSE 0 END) AS PNL_null_count,
        SUM(CASE WHEN Product_description_length IS NULL THEN 1 ELSE 0 END) AS PDL_null_count,
        SUM(CASE WHEN Product_photos_qty IS NULL THEN 1 ELSE 0 END ) AS PPQ_null_count,
        SUM(CASE WHEN Product_weight_g IS NULL THEN 1 ELSE 0 END ) AS PWg_null_count,
        SUM(CASE WHEN Product_length_cm IS NULL THEN 1 ELSE 0 END ) AS PLcm_NULL_COUNT,
        SUM(CASE WHEN Product_height_cm IS NULL THEN 1 ELSE 0 END ) AS PHcm_null_count,
        SUM(CASE WHEN Product_width_cm IS NULL THEN 1 ELSE 0 END ) AS PWcm_null_count
FROM PRODUCTS;

    -- DATA CLEANING PRODUCTS TABLE
SELECT * FROM PRODUCTS
WHERE product_category_name = '#N/A'

-- Replace '#N/A' with 'Unknown' in product_category_name
UPDATE PRODUCTS
SET product_category_name = 'Unknown'
WHERE product_category_name = '#N/A';

SELECT * FROM PRODUCTS
WHERE product_category_name = 'UNKNOWN'

REPLACING NULL VALUES WITH AVERAGE IN THE COLUMN OF ‘product_name_length’ , ‘product_description_length’,  AND ‘Product_photos_qty’ WITH most frequently occurring value
-- Declare and assign variables
DECLARE @AvgNameLength FLOAT;
DECLARE @AvgDescriptionLength FLOAT;
DECLARE @ModePhotos INT;

-- Set variable values using aggregate functions
```

```
SELECT @AvgNameLength = AVG(CAST(product_name_length AS FLOAT))
FROM PRODUCTS
WHERE product_name_length IS NOT NULL;

SELECT @AvgDescriptionLength = AVG(CAST(product_description_length AS FLOAT))
FROM PRODUCTS
WHERE product_description_length IS NOT NULL;

-- Find the mode for Product_photos_qty (most frequently occurring value)
SELECT TOP 1 @ModePhotos = Product_photos_qty
FROM PRODUCTS
WHERE Product_photos_qty IS NOT NULL
GROUP BY Product_photos_qty
ORDER BY COUNT(*) DESC;

-- Impute missing values
UPDATE PRODUCTS
SET product_name_length = @AvgNameLength
WHERE product_name_length IS NULL;

UPDATE PRODUCTS
SET product_description_length = @AvgDescriptionLength
WHERE product_description_length IS NULL;

UPDATE PRODUCTS
SET Product_photos_qty = @ModePhotos
WHERE Product_photos_qty IS NULL;


select * from PRODUCTS
```

-- TABLE SELLERS

```
SELECT * FROM SELLERS;
```

-- Checking for NULL values in the Sellers table

```
SELECT
        SUM(CASE WHEN Seller_id IS NULL THEN 1 ELSE 0 END) AS Seller_id_null_COUNT,
        SUM(CASE WHEN Seller_zip_code_prefix IS NULL THEN 1 ELSE 0 END) SZCP_null_count,
        SUM(CASE WHEN Seller_city IS NULL THEN 1 ELSE 0 END) AS Seller_city_null_count,
        SUM(CASE WHEN Seller_state IS NULL THEN 1 ELSE 0 END) AS Seller_state_null_count

FROM SELLERS;
```

-- Checking for BLANK values in the Sellers table.

```
SELECT
        SUM(CASE WHEN Seller_id = '' THEN 1 ELSE 0 END) AS Seller_id_null_COUNT,
        SUM(CASE WHEN Seller_zip_code_prefix ='' THEN 1 ELSE 0 END) SZCP_null_count,
        SUM(CASE WHEN Seller_city ='' THEN 1 ELSE 0 END) AS Seller_city_null_count,
        SUM(CASE WHEN Seller_state ='' THEN 1 ELSE 0 END) AS Seller_state_null_count

FROM SELLERS;
```

REPEAT CUSTOMER ANALYSIS

```
use [E-Commerce]
-- Calculate average monthly active users per year
SELECT
   year,
   ROUND(AVG(total_customer), 0) AS avg_active_user
FROM (
   SELECT
      YEAR(od.order_purchase_timestamp) AS year,
      MONTH(od.order_purchase_timestamp) AS month,
      COUNT(DISTINCT cd.customer_unique_id) AS total_customer
   FROM orders AS od
   JOIN customers AS cd
      ON od.customer_id = cd.customer_id
   GROUP BY
      YEAR(od.order_purchase_timestamp),
      MONTH(od.order_purchase_timestamp)
) a
GROUP BY year
ORDER BY year;
```

RESULTS

	year	avg_active_user
1	2017	3689
2	2018	6664

```
--Calculate the number of new customers per year
WITH FirstOrder AS (
   SELECT
      cd.customer_unique_id,
      MIN(od.order_purchase_timestamp) AS first_order_date
   FROM orders AS od
```

```
    JOIN customers AS cd
        ON od.customer_id = cd.customer_id
    GROUP BY cd.customer_unique_id
)
SELECT
    YEAR(first_order_date) AS year,
    COUNT(DISTINCT customer_unique_id) AS new_customers
FROM FirstOrder
GROUP BY YEAR(first_order_date)
ORDER BY year;
```

RESULTS

	year	new_customers
1	2017	43658
2	2018	52029

```
-- Calculate the number of customers who placed a repeat order per year
SELECT
    year,
    COUNT(customer_unique_id) AS repeat_customers
FROM (
    SELECT
        YEAR(od.order_purchase_timestamp) AS year,
        cd.customer_unique_id,
        COUNT(od.order_id) AS total_orders
    FROM orders AS od
    JOIN CUSTOMERS AS cd
        ON od.customer_id = cd.customer_id
    GROUP BY YEAR(od.order_purchase_timestamp), cd.customer_unique_id
    HAVING COUNT(od.order_id) > 1
) a
GROUP BY year
ORDER BY year;
```

RESULTS

	year	repeat_customers
1	2017	1242
2	2018	1131

-- -- Calculate the number of customers who placed a repeat order per month

```
SELECT
   FORMAT(order_month, 'yyyy-MM') AS month,
   COUNT(customer_unique_id) AS repeat_customers
FROM (
   SELECT
      DATEFROMPARTS(YEAR(od.order_purchase_timestamp), MONTH(od.order_purchase_timestamp), 1) AS order_month,
      cd.customer_unique_id,
      COUNT(od.order_id) AS total_orders
   FROM orders AS od
   JOIN customers AS cd
      ON od.customer_id = cd.customer_id
   GROUP BY
      DATEFROMPARTS(YEAR(od.order_purchase_timestamp), MONTH(od.order_purchase_timestamp), 1),
      cd.customer_unique_id
   HAVING COUNT(od.order_id) > 1
) a
GROUP BY order_month
ORDER BY order_month;
```

RESULTS

	month	repeat_customers
1	2017-01	25
2	2017-02	22
3	2017-03	39
4	2017-04	28
5	2017-05	71
6	2017-06	62
7	2017-07	76
8	2017-08	76
9	2017-09	69
10	2017-10	67
11	2017-11	107
12	2017-12	64
13	2018-01	101
14	2018-02	149
15	2018-03	92
16	2018-04	55
17	2018-05	56
18	2018-06	38
19	2018-07	61
20	2018-08	38

--Average number of orders per customer per year

```
SELECT
   year,
   ROUND(AVG(total_order), 2) AS avg_frequency_order
FROM (
   SELECT
      YEAR(od.order_purchase_timestamp) AS year,
      cd.customer_unique_id,
      COUNT(DISTINCT od.order_id) AS total_order
   FROM ORDERS AS od
   JOIN CUSTOMERS AS cd
      ON od.customer_id = cd.customer_id
   GROUP BY
      YEAR(od.order_purchase_timestamp),
      cd.customer_unique_id
) a
GROUP BY year
ORDER BY year;
```

RESULTS

	year	avg_frequency_order
1	2017	1
2	2018	1

-- average number of orders per customer per Month

```
SELECT
   FORMAT(order_month, 'yyyy-MM') AS month,
   ROUND(AVG(total_order), 2) AS avg_frequency_order
FROM (
   SELECT
      DATEFROMPARTS(YEAR(od.order_purchase_timestamp), MONTH(od.order_purchase_timestamp), 1) AS
order_month,
      cd.customer_unique_id,
      COUNT(DISTINCT od.order_id) AS total_order
```

```
    FROM ORDERS AS od
    JOIN CUSTOMERS AS cd
        ON od.customer_id = cd.customer_id
    GROUP BY
        DATEFROMPARTS(YEAR(od.order_purchase_timestamp), MONTH(od.order_purchase_timestamp), 1),
        cd.customer_unique_id
) a
GROUP BY order_month
ORDER BY order_month;
```

RESULTS

	month	avg_frequency_order
1	2017-01	1
2	2017-02	1
3	2017-03	1
4	2017-04	1
5	2017-05	1
6	2017-06	1
7	2017-07	1
8	2017-08	1
9	2017-09	1
10	2017-10	1
11	2017-11	1
12	2017-12	1
13	2018-01	1
14	2018-02	1
15	2018-03	1
16	2018-04	1
17	2018-05	1
18	2018-06	1
19	2018-07	1
20	2018-08	1

TRENDS AND SEASONALITY ANALYSIS

```
use [E-Commerce]
```

```
-- Monthly Sales Trend
SELECT
    FORMAT(O.order_purchase_timestamp, 'yyyy-MM') AS month,
    COUNT(DISTINCT O.order_id) AS total_orders,
   ROUND( SUM(OP.PAYMENT_VALUE),0)AS total_sales
FROM ORDERS O
JOIN ORDER_PAYMENTS OP
ON O.order_id = OP.order_id
GROUP BY FORMAT(O.order_purchase_timestamp, 'yyyy-MM')
ORDER BY month;
```

RESULTS

	month	total_orders	total_sales
1	2017-01	797	138250
2	2017-02	1766	289960
3	2017-03	2680	449512
4	2017-04	2400	416420
5	2017-05	3691	592010
6	2017-06	3241	510613
7	2017-07	4021	591549
8	2017-08	4325	673526
9	2017-09	4281	727103
10	2017-10	4626	777661
11	2017-11	7535	1193893
12	2017-12	5666	876723
13	2018-01	7268	1114846
14	2018-02	6724	992075
15	2018-03	7208	1159109
16	2018-04	6939	1160785
17	2018-05	6872	1153936
18	2018-06	6167	1023881
19	2018-07	6291	1066367
20	2018-08	6459	1004839

```
-- Weekly Sales Trend
```

```
SELECT
   DATEPART(YEAR, O.order_purchase_timestamp) AS year,
   DATEPART(week, O.order_purchase_timestamp) AS week,
   COUNT(DISTINCT O.order_id) AS total_orders,
   ROUND(SUM(OP.PAYMENT_VALUE),0) AS total_sales
FROM ORDERS O
JOIN ORDER_PAYMENTS OP
ON O.order_id = OP.order_id
GROUP BY DATEPART(YEAR, O.order_purchase_timestamp), DATEPART(WEEK, O.order_purchase_timestamp)
ORDER BY year, week;
```

RESULTS

	year	week	total_orders	total_sales
1	2017	1	40	3144
2	2017	2	71	13008
3	2017	3	178	26659
4	2017	4	350	63069
5	2017	5	425	80636
6	2017	6	559	84873
7	2017	7	424	76939
8	2017	8	372	53879
9	2017	9	468	87621
10	2017	10	592	93781
11	2017	11	623	109275
12	2017	12	646	106623
13	2017	13	563	98790
14	2017	14	578	108511
15	2017	15	466	72608
16	2017	16	531	95706
17	2017	17	689	109584
18	2017	18	741	108615
19	2017	19	792	134361
20	2017	20	904	140142
21	2017	21	831	143896
22	2017	22	831	130250
23	2017	23	831	127413
24	2017	24	778	120145
25	2017	25	645	103612
26	2017	26	727	117766
27	2017	27	853	131131
28	2017	28	938	131552
29	2017	29	971	146422
30	2017	30	914	132484
31	2017	31	991	141400
32	2017	32	952	152850
33	2017	33	1034	151603
34	2017	34	874	125610

Query executed successfully.

-- Day_of Week Sales Trend

```
        SELECT
    DATENAME(WEEKDAY, O.order_purchase_timestamp) AS day_of_week,
    COUNT(DISTINCT O.order_id) AS total_orders,
    ROUND(SUM(OP.payment_value),0) AS total_sales
FROM ORDERS O
JOIN ORDER_PAYMENTS OP ON O.order_id = OP.order_id
GROUP BY DATENAME(WEEKDAY, O.order_purchase_timestamp)
ORDER BY
    CASE DATENAME(WEEKDAY, O.order_purchase_timestamp)
        WHEN 'Sunday' THEN 1
        WHEN 'Monday' THEN 2
        WHEN 'Tuesday' THEN 3
        WHEN 'Wednesday' THEN 4
        WHEN 'Thursday' THEN 5
        WHEN 'Friday' THEN 6
        WHEN 'Saturday' THEN 7
    END;
```

RESULTS

	day_of_week	total_orders	total_sales
1	Sunday	11920	1865485
2	Monday	16126	2612689
3	Tuesday	15871	2541143
4	Wednesday	15484	2479427
5	Thursday	14678	2368431
6	Friday	14053	2291686
7	Saturday	10825	1754197

--hour of the day sales trend

```
SELECT
    DATEPART(HOUR, O.order_purchase_timestamp) AS hour_of_day,
    COUNT(DISTINCT O.order_id) AS total_orders,
    ROUND(SUM(OP.payment_value),0)  AS total_sales
FROM ORDERS O
JOIN ORDER_PAYMENTS OP ON O.order_id = OP.order_id
GROUP BY DATEPART(HOUR, O.order_purchase_timestamp)
ORDER BY hour_of_day;
```

RESULTS

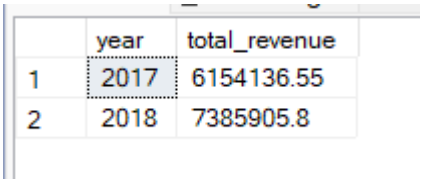
	hour_of_day	total_orders	total_sales
1	0	2386	373607
2	1	1161	173775
3	2	506	64922
4	3	272	41914
5	4	203	28452
6	5	188	26216
7	6	502	67684
8	7	1224	181444
9	8	2947	458476
10	9	4757	795069
11	10	6142	983099
12	11	6545	1027837
13	12	5961	990691
14	13	6485	1023265
15	14	6537	1104514
16	15	6412	1055733
17	16	6648	1093700
18	17	6123	983770
19	18	5744	955855
20	19	5965	964735
21	20	6171	1000766
22	21	6199	981880
23	22	5775	915475
24	23	4104	620176

ANNUAL PRODUCT CATEGORY ANALYSIS

-- Calculate total revenue per year

```
SELECT
   YEAR(od.order_purchase_timestamp) AS year,
   ROUND(SUM(oi.price), 2) AS total_revenue
FROM orders AS od
JOIN order_items AS oi
   ON od.order_id = oi.order_id
GROUP BY YEAR(od.order_purchase_timestamp)
ORDER BY year;
```

RESULTS



	year	total_revenue
1	2017	6154136.55
2	2018	7385905.8

-- Calculate month on month growth rate in term of revenue.

```
WITH RevenuePerMonth AS (
   SELECT
      DATEFROMPARTS(YEAR(od.order_purchase_timestamp), MONTH(od.order_purchase_timestamp), 1) AS order_month,
      ROUND(SUM(oi.price), 2) AS total_revenue
   FROM orders AS od
   JOIN order_items AS oi
      ON od.order_id = oi.order_id
   GROUP BY DATEFROMPARTS(YEAR(od.order_purchase_timestamp), MONTH(od.order_purchase_timestamp), 1)
)

SELECT
   FORMAT(order_month, 'yyyy-MM') AS month,
   total_revenue,
```

```
    LAG(total_revenue) OVER (ORDER BY order_month) AS prev_month_revenue,
    ROUND(
        CASE
            WHEN LAG(total_revenue) OVER (ORDER BY order_month) = 0 THEN NULL
            ELSE
                (CAST(total_revenue - LAG(total_revenue) OVER (ORDER BY order_month) AS FLOAT) * 100.0)
                / NULLIF(LAG(total_revenue) OVER (ORDER BY order_month), 0)
        END, 2
    ) AS mom_growth_percentage
FROM RevenuePerMonth
ORDER BY order_month;
```

RESULTS

	month	total_revenue	prev_month_revenue	mom_growth_percentage
1	2017-01	120226.98	NULL	NULL
2	2017-02	245718.48	120226.98	104.38
3	2017-03	374344.3	245718.48	52.35
4	2017-04	359927.23	374344.3	-3.85
5	2017-05	506071.14	359927.23	40.6
6	2017-06	433038.6	506071.14	-14.43
7	2017-07	498031.48	433038.6	15.01
8	2017-08	573971.68	498031.48	15.25
9	2017-09	624401.69	573971.68	8.79
10	2017-10	664219.43	624401.69	6.38
11	2017-11	1010271.37	664219.43	52.1
12	2017-12	743914.17	1010271.37	-26.36
13	2018-01	950030.36	743914.17	27.71
14	2018-02	844178.71	950030.36	-11.14
15	2018-03	983213.44	844178.71	16.47
16	2018-04	996647.75	983213.44	1.37
17	2018-05	996517.68	996647.75	-0.01
18	2018-06	865124.31	996517.68	-13.19
19	2018-07	895507.22	865124.31	3.51
20	2018-08	854686.33	895507.22	-4.56

```
-- Calculate total canceled orders per year
SELECT
    YEAR(order_purchase_timestamp) AS year,
    COUNT(order_id) AS canceled_orders
FROM orders
```

```
WHERE order_status = 'canceled'
GROUP BY YEAR(order_purchase_timestamp)
ORDER BY year;
```

RESULTS

	year	canceled_orders
1	2017	211
2	2018	253

```
-- Calculate highest total revenues per product category per year.
WITH RevenuePerCategoryYear AS (
    SELECT
        YEAR(od.order_purchase_timestamp) AS year,
        p.product_category_name,
        ROUND(SUM(oi.price), 2) AS total_revenue
    FROM orders od
    JOIN order_items oi
        ON od.order_id = oi.order_id
    JOIN products p
        ON oi.product_id = p.product_id
    GROUP BY
        YEAR(od.order_purchase_timestamp),
        p.product_category_name
),

RankedCategories AS (
    SELECT *,
        RANK() OVER (PARTITION BY year ORDER BY total_revenue DESC) AS revenue_rank
    FROM RevenuePerCategoryYear
)

SELECT
    year,
```

```
    product_category_name,
    total_revenue
FROM RankedCategories
WHERE revenue_rank = 1
ORDER BY year;
```

RESULTS

	year	product_category_name	total_revenue
1	2017	Bed_Bath_Table	498306.44
2	2018	Health_Beauty	772238.15

```
-- Calculate highest canceled order per product category per year.
WITH CanceledOrdersPerCategoryYear AS (
    SELECT
        YEAR(od.order_purchase_timestamp) AS year,
        p.product_category_name,
        COUNT(DISTINCT od.order_id) AS canceled_orders
    FROM orders od
    JOIN order_items oi
        ON od.order_id = oi.order_id
    JOIN products p
        ON oi.product_id = p.product_id
    WHERE od.order_status = 'canceled'
    GROUP BY
        YEAR(od.order_purchase_timestamp),
        p.product_category_name
),

RankedCanceledCategories AS (
    SELECT *,
        RANK() OVER (PARTITION BY year ORDER BY canceled_orders DESC) AS cancel_rank
    FROM CanceledOrdersPerCategoryYear
)
```

```
SELECT
    year,
    product_category_name,
    canceled_orders
FROM RankedCanceledCategories
WHERE cancel_rank = 1
ORDER BY year;
```

RESULTS

	year	product_category_name	canceled_orders
1	2017	Sports_Leisure	22
2	2018	Health_Beauty	27