Youtube Channel: https://www.youtube.com/watch?v=KaPZLQb2D80

🌳 Determine the company's growth volume by the Overall Session and Order Volume; trended in Quarterly review.

```
*****************************************************************************************
              SELECT YEAR(website_sessions.created_at) AS Year
                   , QUARTER(website_sessions.created_at) AS Quarter,
                   , COUNT(DISTINCT website_sessions.website_session_id) AS Sessions,
                   , COUNT(DISTINCT orders.order_id) AS Orders
              FROM website_sessions
              LEFT JOIN orders ON website_sessions.website_session_id =
              orders.website_session_id
              WHERE website_sessions.created_at < '2015-01-01'
              GROUP BY YEAR(website_sessions.created_at),
                       QUARTER(website_sessions.created_at)
*****************************************************************************************
```

🌳 Demonstrate all the efficiency improvements since the website launch period by Showcasing the: Session to Order Conversion Rate | Revenue per Order | Revenue per Session; in Quarterly review

```
*****************************************************************************************
              SELECT YEAR(website_sessions.created_at) AS year
                   , QUARTER(website_sessions.created_at) AS Quarter
                   , COUNT(DISTINCT orders.order_id)/COUNT(DISTINCT
              website_sessions.website_session_id) AS Sess_to_Order_Convr_rate
                   , SUM(orders.price_usd)/COUNT(DISTINCT orders.order_id) AS
              Revenue_per_Order
                   , SUM(orders.price_usd)/COUNT(DISTINCT
              website_sessions.website_session_id) AS Revenue_per_Session
              FROM website_sessions
              LEFT JOIN orders ON website_sessions.website_session_id =
              orders.website_session_id
              WHERE website_sessions.created_at < '2015-01-01'
              GROUP BY YEAR(website_sessions.created_at),
                       QUARTER(website_sessions.created_at)
*****************************************************************************************
```

🌳 Showcase the Paid Channels' Traffic Efficiency to Order by Pulling the Quarterly trend for: i. Gsearch nonbrand ii. Bsearch nonbrand iii. G/Bsearch brand iv. Organic search v. Direct-type-in

```
*****************************************************************************************
              SELECT MIN(DATE(website_sessions.created_at)) Date_in_quarters
              , COUNT(DISTINCT CASE WHEN utm_source = 'gsearch' AND utm_campaign =
              'nonbrand' THEN order_id END) AS Gsearch_nonbrand
              , COUNT(DISTINCT CASE WHEN utm_source = 'bsearch' AND utm_campaign =
              'nonbrand' THEN order_id END) AS Bsearch_nonbrand
              , COUNT(DISTINCT CASE WHEN utm_source IN ('gsearch', 'bsearch') AND
              utm_campaign = 'brand'   THEN order_id END) AS Brand_search
              , COUNT(DISTINCT CASE WHEN utm_source IS NULL AND http_referer IN
              ('https://www.gsearch.com', 'https://www.bsearch.com') THEN order_id END) AS
              Organic_search
              , COUNT(DISTINCT CASE WHEN utm_source IS NULL AND http_referer IS NULL THEN
              order_id END) AS Direct_type_in
              FROM website_sessions
              LEFT JOIN orders ON website_sessions.website_session_id =
              orders.website_session_id
              WHERE website_sessions.created_at < '2015-01-01'
              GROUP BY YEAR(website_sessions.created_at),
                            QUARTER(website_sessions.created_at)
*****************************************************************************************
```

📍 Return the overall session to Order Conversion Rate trend for the same Channels above

****

```
SELECT YEAR(website_sessions.created_at) AS Year
, QUARTER(website_sessions.created_at) AS Quarter
, COUNT(DISTINCT CASE WHEN utm_source = 'gsearch' AND utm_campaign =
'nonbrand' THEN order_id ELSE NULL END)
  /COUNT(DISTINCT CASE WHEN utm_source = 'gsearch' AND utm_campaign =
'nonbrand' THEN website_sessions.website_session_id ELSE NULL END) AS
Gsearch_nonbrand_Convr_rate
, COUNT(DISTINCT CASE WHEN utm_source = 'bsearch' AND utm_campaign =
'nonbrand' THEN order_id END) /COUNT(DISTINCT CASE WHEN utm_source =
'bsearch' AND utm_campaign = 'nonbrand' THEN
website_sessions.website_session_id END) AS Bsearch_nonbrand_Convr_rate
, COUNT(DISTINCT CASE WHEN utm_source IN ('gsearch', 'bsearch') AND
utm_campaign = 'brand' THEN order_id END)/COUNT(DISTINCT CASE WHEN utm_source
IN ('gsearch', 'bsearch') AND utm_campaign = 'brand' THEN
website_sessions.website_session_id END) AS Brand_search_Convr_rate
, COUNT(DISTINCT CASE WHEN utm_source IS NULL AND http_referer IN
('https://www.gsearch.com', 'https://www.bsearch.com') THEN order_id END)
/COUNT(DISTINCT CASE WHEN utm_source IS NULL AND http_referer IN
('https://www.gsearch.com', 'https://www.bsearch.com') THEN
website_sessions.website_session_id END) AS Organic_search_Convr_rate
, COUNT(DISTINCT CASE WHEN utm_source IS NULL AND http_referer IS NULL THEN
order_id END /COUNT(DISTINCT CASE WHEN utm_source IS NULL AND http_referer IS
NULL THEN website_sessions.website_session_id END) AS
Direct_type_in_Convr_rate
FROM website_sessions
  LEFT JOIN orders
    ON website_sessions.website_session_id = orders.website_session_id
WHERE website_sessions.created_at < '2015-01-01'
GROUP BY YEAR(website_sessions.created_at),
              QUARTER(website_sessions.created_at)
```

****

📍 Compare the impact of introducing A Product every year on the Sales and Revenue, alongside the Margin; trending by Year.

****

```
SELECT
     YEAR(created_at) AS Year,
     COUNT(DISTINCT order_id) AS Total_sales,
     SUM(price_usd) AS Total_revenue,
     SUM(price_usd - cogs_usd) AS Margin
FROM orders
WHERE primary_product_id = 1
  AND created_at < '2015-01-01'
GROUP BY 1;
```

****

📍 Return Overall Products sessions

```
SELECT YEAR(created_at) AS Year
 , COUNT(DISTINCT order_id) AS Total_sales
 , SUM(price_usd) AS Total_revenue
 , SUM(price_usd - cogs_usd) AS Margin
FROM orders
WHERE created_at < '2015-01-01'
GROUP BY 1;
```

💡 Find all customers who placed at least 1 order in 2017 but did not place any orders in 2018.
****

```
SELECT CONTACT_NAME
,COUNT(*) NUMBER_OF_ORDERS
, COUNT(CASE WHEN STRFTIME('%Y',ORDER_DATE) = '2017' THEN 1 END) CUST_2017
, COUNT(CASE WHEN STRFTIME('%Y',ORDER_DATE) = '2018' THEN 1 END) CUST_2018
FROM CUSTOMERS
JOIN ORDERS ON ORDERS.CUSTOMER_ID = CUSTOMERS.CUSTOMER_ID
GROUP BY 1
HAVING COUNT(CASE WHEN STRFTIME('%Y',ORDER_DATE) = '2017' THEN 1 END) >= 1
AND COUNT(CASE WHEN STRFTIME('%Y',ORDER_DATE) = '2018' THEN 1 END) = 0
```

****

💡 For each customer label as loyal or not loyal: Loyal = more than 10 orders.
****

```
SELECT CONTACT_NAME
, COUNT(*) AS NUMBER_OF_ORDERS
, CASE WHEN COUNT(*) > 10 THEN 'LOYAL' ELSE 'N/A' END AS
LABEL_LOYAL_CUSTOMERS
FROM CUSTOMERS
JOIN ORDERS ON ORDERS.CUSTOMER_ID = CUSTOMERS.CUSTOMER_ID
GROUP BY 1
```

****

💡 Find percent total for each Category
****

```
SELECT CATEGORY_NAME
, AVG(UNIT_PRICE) AS AVG_UNIT_PRICE_FOR_CATEGORY
, SUM(UNIT_PRICE * QUANTITY) TOTAL_PRICE_PER_CATEGORY
, SUM(SUM(UNIT_PRICE * QUANTITY)) OVER()
, SUM(UNIT_PRICE * QUANTITY) / SUM(SUM(UNIT_PRICE * QUANTITY)) OVER()
FROM CATEGORIES
JOIN PRODUCTS ON PRODUCTS.CATEGORY_ID = CATEGORIES.CATEGORY_ID
JOIN ORDER_DETAILS ON ORDER_DETAILS.PRODUCT_ID = PRODUCTS.PRODUCT_ID
GROUP BY 1
```

****

💡 Calculate the average number of products per product category.
****

```
SELECT ROUND(AVG(NUMBER_OF_PRODUCTS),2)AVG_NUMBER_OF_PRODUCTS_PER_CATEGORY
        FROM
                (SELECT COUNT(PRODUCT_NAME) NUMBER_OF_PRODUCTS
                , CATEGORY_NAME
                FROM PRODUCTS
                JOIN CATEGORIES ON CATEGORIES.CATEGORY_ID = PRODUCTS.CATEGORY_ID
        GROUP BY CATEGORY_NAME) TEMP1
```

****

💡 Calculate average number of employees per manager.
****

```
SELECT AVG(NUMBER_EMPLOYEES) AS AVG_NUMBER_OF_EMPLOYEES
FROM
        (SELECT COUNT(*)NUMBER_EMPLOYEES
        ,REPORTS_TO
        FROM EMPLOYEES
        WHERE REPORTS_TO IN (2,5)
GROUP BY 2)TEMP1
```

****

📍 Calculate the average number of products per product category.

****

```
SELECT ROUND(AVG(NUMBER_OF_PRODUCTS),2)AVG_NUMBER_OF_PRODUCTS_PER_CATEGORY
FROM
       (SELECT COUNT(PRODUCT_NAME) NUMBER_OF_PRODUCTS
       , CATEGORY_NAME
       FROM PRODUCTS
       JOIN CATEGORIES ON CATEGORIES.CATEGORY_ID = PRODUCTS.CATEGORY_ID
GROUP BY CATEGORY_NAME) TEMP1
```

****

📍 For each product category, show total revenue and average unit price. Flag categories with average price below a threshold, like $10.

****

```
SELECT CATEGORY_NAME
, AVG(UNIT_PRICE) AS AVG_UNIT_PRICE_FOR_CATEGORY
, SUM(UNIT_PRICE * QUANTITY) AS TOTAL_REVENUE
, CASE WHEN AVG(UNIT_PRICE) < 25
          THEN 'LESS THAN 25' ELSE 'ABOVE 25'
       END AS MORE_OR_LESS_THAN_TWENTY_FIVE
FROM CATEGORIES
JOIN PRODUCTS ON PRODUCTS.CATEGORY_ID = CATEGORIES.CATEGORY_ID
JOIN ORDER_DETAILS ON ORDER_DETAILS.PRODUCT_ID = PRODUCTS.PRODUCT_ID
GROUP BY 1
```

****

📍 Find all orders that are above average order value and label them as either above or below.

****

```
SELECT ORDER_ID
, SUM(QUANTITY * UNIT_PRICE) TOTAL_PRICE_PER_ORDER
, AVG(SUM(QUANTITY * UNIT_PRICE)) OVER() AOV
, CASE WHEN
             SUM(QUANTITY * UNIT_PRICE) > AVG(SUM(QUANTITY * UNIT_PRICE))
OVER()
        THEN 'ABOVE AVG ORDER PRICE'
        ELSE 'BELOW AVG ORDER PRICE'
        END AS BELOW_OR_ABOVE
  FROM ORDER_DETAILS
  JOIN PRODUCTS ON PRODUCTS.PRODUCT_ID = ORDER_DETAILS.PRODUCT_ID
  GROUP BY 1
```

****

📍 Identify the customers whose revenue falls below the 10th percentile.

****

```
WITH CTE1 AS (SELECT CUSTOMERS.CONTACT_NAME
, SUM(QUANTITY * UNIT_PRICE) TOTAL_REVENUE
, SUM(SUM(QUANTITY * UNIT_PRICE)) OVER(ORDER BY SUM(QUANTITY * UNIT_PRICE))
RUNNING_TOTAL
, SUM(SUM(QUANTITY * UNIT_PRICE)) OVER() TOTAL_REV
, SUM(SUM(QUANTITY * UNIT_PRICE)) OVER(ORDER BY SUM(QUANTITY * UNIT_PRICE))
  / SUM(SUM(QUANTITY * UNIT_PRICE)) OVER() * 100 PERCENTAGE
FROM CUSTOMERS
JOIN ORDERS ON ORDERS.CUSTOMER_ID = CUSTOMERS.CUSTOMER_ID
JOIN ORDER_DETAILS ON ORDER_DETAILS.ORDER_ID = ORDERS.ORDER_ID
JOIN PRODUCTS ON PRODUCTS.PRODUCT_ID = ORDER_DETAILS.PRODUCT_ID
GROUP BY 1)
SELECT CONTACT_NAME
, RUNNING_TOTAL
, TOTAL_REV
, PERCENTAGE
FROM CTE1
WHERE PERCENTAGE < 10
```

****

📍 Find customers who have ordered at least one product from every category.
****

```
SELECT CUSTOMERS.CUSTOMER_ID
,COUNT(DISTINCT CATEGORY_ID)
FROM CUSTOMERS
JOIN ORDERS ON ORDERS.CUSTOMER_ID = CUSTOMERS.CUSTOMER_ID
JOIN ORDER_DETAILS ON ORDER_DETAILS.ORDER_ID = ORDERS.ORDER_ID
JOIN PRODUCTS ON PRODUCTS.PRODUCT_ID = ORDER_DETAILS.PRODUCT_ID
GROUP BY 1
HAVING COUNT(DISTINCT CATEGORY_ID) >1
ORDER BY COUNT(DISTINCT CATEGORY_ID) DESC
```

****

📍 For the customers countries that start with letter “S” find the average number of orders per customer for each country.
****

```
SELECT COUNTRY
, AVG(NUMBER_OF_ORDERS) AVG_NUM_ORDERS_PER_CUST
FROM
(SELECT CUSTOMERS.CUSTOMER_ID
, COUNT(*) NUMBER_OF_ORDERS
, COUNTRY
FROM CUSTOMERS
JOIN ORDERS ON ORDERS.CUSTOMER_ID = CUSTOMERS.CUSTOMER_ID
WHERE COUNTRY LIKE 'S%'
GROUP BY 1,3)TEMP
GROUP BY 1
```

****

📍 Return products that contribute to 80% of the overall Revenue.
****

```
WITH CTE1 AS (select PRODUCT_NAME
,SUM(QUANTITY * UNIT_PRICE) revenue_per_product
,SUM(sum(quantity * unit_price)) over()
, SUM(QUANTITY * UNIT_PRICE) / SUM(sum(quantity * unit_price)) over() * 100
PERCENT_OF_TOTAL
from products
join order_details on products.product_id = order_details.product_id
JOIN ORDERS ON ORDERS.ORDER_ID = ORDER_DETAILS.ORDER_ID
WHERE ORDER_DATE >= JULIANDAY('2017-01-21')
GROUP BY 1
ORDER BY revenue_per_product DESC)


, CTE2 AS (SELECT PRODUCT_NAME
, REVENUE_PER_PRODUCT
, PERCENT_OF_TOTAL
,SUM(PERCENT_OF_TOTAL) OVER(ORDER BY PERCENT_OF_TOTAL DESC) RUNNING_TOTAL
FROM CTE1)

SELECT *
FROM CTE2
WHERE RUNNING_TOTAL <=81
```

****

📍 Find users who were active for 4 consecutive days.
****

```
       WITH customer_orders AS (
                  SELECT
                      customerid,
                      CAST(orderdate AS date) AS order_date
                  FROM orders
                  GROUP BY customerid, CAST(orderdate AS date)
              ),
       ordered_dates AS (
                  SELECT
                      customerid,
                      order_date,
                      ROW_NUMBER() OVER (PARTITION BY customerid ORDER BY order_date) AS rn
                  FROM customer_orders
              ),
       grouped_dates AS (
                  SELECT
                      customerid,
                      order_date,
                      order_date - (rn * INTERVAL '1 day') AS grp_key
                  FROM ordered_dates
              )
              SELECT
                  customerid,
                  MIN(order_date) AS start_date,
                  MAX(order_date) AS end_date,
                  COUNT(*) AS consecutive_days
              FROM grouped_dates
              GROUP BY customerid, grp_key
              HAVING COUNT(*) >= 4
              ORDER BY customerid, start_date;
```

****

📍 Find Number of orders related to the Beverages category for each customer
****

```
              SELECT  FIRST_NAME ||' '||LAST_NAME
              , count(DISTINCT O.ORDER_ID)
              FROM EMPLOYEES E
              JOIN ORDERS O ON O.EMPLOYEE_ID = E.EMPLOYEE_ID
              JOIN ORDER_DETAILS OD ON OD.ORDER_ID = O.ORDER_ID
              JOIN PRODUCTS P ON P.PRODUCT_ID = OD.PRODUCT_ID
              JOIN CATEGORIES C ON C.CATEGORY_ID = P.CATEGORY_ID
              WHERE CATEGORY_NAME = 'Beverages'
              GROUP BY 1
```

****

📍 Add Columns that show Product Name and Revenue for worst selling Product
****

```
WITH top_income AS (
          SELECT CATEGORY_NAME
        , PRODUCT_NAME
        , SUM(UNIT_PRICE * QUANTITY) AS REVENUE
        , RANK() OVER (PARTITION BY CATEGORY_NAME ORDER BY SUM(UNIT_PRICE * QUANTITY)
        DESC)RK
        FROM CATEGORIES
        JOIN PRODUCTS ON PRODUCTS.CATEGORY_ID = CATEGORIES.CATEGORY_ID
        JOIN ORDER_DETAILS ON ORDER_DETAILS.PRODUCT_ID = PRODUCTS.PRODUCT_ID
        GROUP BY 1,2    )

,lowest_income AS (
          SELECT CATEGORY_NAME
        , PRODUCT_NAME
        , SUM(UNIT_PRICE * QUANTITY) AS REVENUE
        , RANK() OVER (PARTITION BY CATEGORY_NAME ORDER BY SUM(UNIT_PRICE * QUANTITY)
        ASC)RK2
        FROM CATEGORIES
        JOIN PRODUCTS ON PRODUCTS.CATEGORY_ID = CATEGORIES.CATEGORY_ID
        JOIN ORDER_DETAILS ON ORDER_DETAILS.PRODUCT_ID = PRODUCTS.PRODUCT_ID
        GROUP BY 1,2)

        SELECT top_income.CATEGORY_NAME
        , top_income.PRODUCT_NAME as highest_selling_product
        , top_income.REVENUE as highest_selling_product_price
        , lowest_income.product_name as lowest_selling_product
        , lowest_income.revenue as lowest_selling_product_price
        FROM top_income
        JOIN lowest_income ON lowest_income.CATEGORY_NAME = top_income.CATEGORY_NAME
        WHERE RK = 1
        AND RK2 = 1
```

****

📍 For each employee, calculate their total sales revenue for each quarter of the year 2016. The output should include the employee's name and four separate columns for the total sales in Quarter 1, Quarter 2, Quarter 3, and Quarter 4. If an employee had no sales in a particular quarter, the value should be

****

```
SELECT FIRST_NAME ||' '|| LAST_NAME
,SUM(QUANTITY * UNIT_PRICE) TOTAL_REVENUE
,SUM(CASE WHEN
       STRFTIME('%m',ORDER_DATE) IN ('01','02','03') THEN QUANTITY * UNIT_PRICE END)
AS Q1
,SUM(CASE WHEN
       STRFTIME('%m',ORDER_DATE) IN ('04','05','06') THEN QUANTITY * UNIT_PRICE END)
AS Q2
,SUM(CASE WHEN
       STRFTIME('%m',ORDER_DATE) IN ('07','08','09') THEN QUANTITY * UNIT_PRICE END)
AS Q3
,SUM(CASE WHEN
       STRFTIME('%m',ORDER_DATE) IN ('10','11','12') THEN QUANTITY * UNIT_PRICE END)
AS Q4
FROM EMPLOYEES
JOIN ORDERS ON ORDERS.EMPLOYEE_ID = EMPLOYEES.EMPLOYEE_ID
JOIN ORDER_DETAILS ON ORDER_DETAILS.ORDER_ID = ORDERS.ORDER_ID
JOIN PRODUCTS ON PRODUCTS.PRODUCT_ID = ORDER_DETAILS.PRODUCT_ID
GROUP BY 1
```

****

💡 For each employee, calculate their total sales revenue in 2016 and 2017, and compute the percentage growth.

****

```
WITH CTE1 AS (SELECT FIRST_NAME
,LAST_NAME
,SUM(CASE WHEN STRFTIME('%Y',ORDER_DATE) = '2016'
     THEN QUANTITY * UNIT_PRICE END)AS REVENUE_2016
,SUM(CASE WHEN STRFTIME('%Y',ORDER_DATE) = '2017'
     THEN QUANTITY * UNIT_PRICE END)AS REVENUE_2017
FROM EMPLOYEES
JOIN ORDERS ON ORDERS.EMPLOYEE_ID = EMPLOYEES.EMPLOYEE_ID
JOIN ORDER_DETAILS ON ORDER_DETAILS.ORDER_ID = ORDERS.ORDER_ID
JOIN PRODUCTS ON PRODUCTS.PRODUCT_ID = ORDER_DETAILS.PRODUCT_ID
GROUP BY 1,2)
SELECT FIRST_NAME
,LAST_NAME
, REVENUE_2016
, REVENUE_2017
,((REVENUE_2017 - REVENUE_2016) / REVENUE_2016) * 100 AS PCT_CHANGE
FROM CTE1
```

****

💡 Calculate month over month increase or decrease

****

```
SELECT COUNT(*) NUMBER_OF_ORDERS
, STRFTIME('%m',order_date) as mnth
, LAG(COUNT(*)) OVER(ORDER BY STRFTIME('%m',order_date)ASC)
, (COUNT(*) / (LAG(COUNT(*)) OVER(ORDER BY STRFTIME('%m',order_date)ASC)  * 1.0) -
1)* 100 AS MoM_change
FROM ORDERS
group by 2
```

****

💡 For each order number return order value, freight charge, freight charge + order value:

****

```
SELECT ORDERS.ORDER_ID
,SUM(QUANTITY * UNIT_PRICE) AS ORDER_VALUE
, FREIGHT
, SUM(QUANTITY * UNIT_PRICE) + FREIGHT AS FREIGHT_PLUS_ORDER_VALUE
, (SUM(QUANTITY * UNIT_PRICE) - (SUM(QUANTITY * UNIT_PRICE) * DISCOUNT))+ FREIGHT
AS DISCOUNTED_PRICE
FROM ORDERS
JOIN ORDER_DETAILS ON ORDER_DETAILS.ORDER_ID = ORDERS.ORDER_ID
JOIN PRODUCTS ON PRODUCTS.PRODUCT_ID = ORDER_DETAILS.PRODUCT_ID
GROUP BY 1,3
```

****

💡 **Find all orders that have above average order value and label them as above or below average:**

****

```
             WITH CTE1 AS (
               SELECT ORDER_ID
             , SUM(QUANTITY * UNIT_PRICE) TOTAL_PRICE_PER_ORDER
               FROM ORDER_DETAILS
               JOIN PRODUCTS ON PRODUCTS.PRODUCT_ID = ORDER_DETAILS.PRODUCT_ID
               GROUP BY 1
               )

              ,CTE2 AS (
                 SELECT AVG(TOTAL_PRICE_PER_ORDER) AVG_ORDER_VALUE
                    FROM CTE1

                 )
              SELECT CTE1.ORDER_ID
             , CTE1.TOTAL_PRICE_PER_ORDER
             , AVG_ORDER_VALUE
             , CASE
                    WHEN CTE1.TOTAL_PRICE_PER_ORDER > AVG_ORDER_VALUE
                 THEN 'ABOVE AVG ORDER VALUE'
                 ELSE 'BELOW AVG ORDER VALUE'
                 END AS ABOVE_OR_BELOW
             FROM CTE1,CTE2
```

****

💡 Find the **customer who placed orders over the longest span of time** (from first order to most recent)

****

```
             SELECT CONTACT_NAME
             , MIN(ORDER_DATE) FIRST_ORDER_DATE
             , MAX(ORDER_DATE) MOST_RECENT_ORDER_DATE
             , JULIANDAY(MAX(ORDER_DATE)) - JULIANDAY(MIN(ORDER_DATE)) CUSTOMER_DURATION
             FROM CUSTOMERS
             JOIN ORDERS ON ORDERS.CUSTOMER_ID = CUSTOMERS.CUSTOMER_ID
             GROUP BY 1
             ORDER BY CUSTOMER_DURATION DESC
             LIMIT 1
```

****