Portfolio Project

Yugabyte Retail Store

Mariya Hareem

install.packages(“RMySQL”)

SQL Queries Report

```
-- SALES ORDERS TRENDS OVER YEARS

    SELECT

        YEAR(o.created_at) AS Year,

        MONTH(o.created_at) AS Month,

        MONTHNAME(o.created_at) AS Month_Name,

        COUNT(DISTINCT o.ï»¿id) AS No_of_Orders,

        ROUND(SUM((o.total - COALESCE(o.discount, 0)) * o.quantity), 0) AS Total_Revenue,

        ROUND(SUM(((o.total - COALESCE(o.discount, 0)) * o.quantity) - p.price * o.quantity), 0) AS Total_Profit,

        ROUND(SUM((o.total - COALESCE(o.discount, 0)) * o.quantity), 0) AS Order_Value

    FROM

        orders o

    JOIN
```

```
    products p ON o.product_id = p.id
WHERE
    YEAR(o.created_at) IS NOT NULL
GROUP BY
    YEAR(o.created_at),
    MONTH(o.created_at),
    MONTHNAME(o.created_at)
ORDER BY
    Year, Month;




-- QUARTERLY RECURRING REVENUE
WITH base AS (
    SELECT
        user_id,
        ROUND(SUM((orders.total - COALESCE(orders.discount, 0)) * orders.quantity), 0)  AS sales,
        EXTRACT(YEAR FROM DATE(orders.created_at)) AS year,
        EXTRACT(QUARTER FROM DATE(orders.created_at)) AS quarter
    FROM
        orders
    WHERE
```

```
        user_id IS NOT NULL
    GROUP BY
        user_id,
        year,
        quarter
),
quarterly_ranked AS (
    SELECT
        *,
        DENSE_RANK() OVER(ORDER BY year, quarter) AS i_quarter
    FROM
        base
),
precedent AS (
    SELECT
        user_id,
        year,
        quarter,
        i_quarter + 1 AS i_prec,
        sales
    FROM
```

```
        quarterly_ranked
),
joined AS (
    SELECT
        p.user_id,
        p.sales AS sales_prec,
        r.sales AS sales_curr,
        r.i_quarter,
        r.year,
        r.quarter  -- Adding the missing quarter column
    FROM
        quarterly_ranked r
    JOIN
        precedent p ON p.user_id = r.user_id AND p.i_prec = r.i_quarter
)
SELECT
    year,
    quarter,
    ROUND(SUM(sales_curr) / SUM(sales_prec), 4) AS nrr
FROM
    joined
```

```
    GROUP BY

        year,

        quarter

    ORDER BY

        year,

        quarter;



-- CUSTOMER ACQUISITION ANALYSIS OVER YEARS AND QUARTER
WITH monthly_customers AS (
    SELECT
        YEAR(ï»¿created_at) AS year,
        MONTH(ï»¿created_at) AS month,
        MONTHNAME(ï»¿created_at) AS monthname,
        quarter(ï»¿created_at) AS quarter,
        COUNT(DISTINCT id) AS total_customers
    FROM
        users
    GROUP BY
        YEAR(ï»¿created_at), MONTH(ï»¿created_at),monthname(ï»¿created_at),quarter(ï»¿created_at)
),
customer_counts AS (
    SELECT
        year,
        month,
        monthname,
        quarter,
        total_customers,
        LAG(total_customers) OVER (ORDER BY year, month) AS customers_at_beginning,
        ABS(COALESCE(total_customers - LAG(total_customers) OVER (ORDER BY year, month), total_customers))
AS new_customers_joined
    FROM
        monthly_customers
)
SELECT
    c.year,
    c.month,
    c.monthname,
    c.quarter,
    c.customers_at_beginning,
    c.new_customers_joined,
    CASE
```

```
        WHEN c.customers_at_beginning IS NULL THEN c.total_customers
        ELSE ABS(c.customers_at_beginning + c.new_customers_joined)
    END AS customers_at_end,
    o.customers_ordered
FROM
    customer_counts c
LEFT JOIN (
    SELECT
        YEAR(created_at) AS year,
        MONTH(created_at) AS month,
        MONTHNAME(created_at) AS monthname,
        quarter(created_at) AS quarter,
                COUNT(DISTINCT user_id) AS customers_ordered
    FROM
        orders
    GROUP BY
        YEAR(created_at), MONTH(created_at),MONTHNAME(created_at),quarter(created_at)
) o ON c.year = o.year AND c.month = o.month AND c.quarter = o.quarter;

-- Age Group analysis

SELECT
    CASE
        WHEN TIMESTAMPDIFF(YEAR, users.birth_date, CURDATE()) BETWEEN 18 AND 24 THEN '18-24'
        WHEN TIMESTAMPDIFF(YEAR, users.birth_date, CURDATE()) BETWEEN 25 AND 34 THEN '25-34'
        WHEN TIMESTAMPDIFF(YEAR, users.birth_date, CURDATE()) BETWEEN 35 AND 44 THEN '35-44'
        WHEN TIMESTAMPDIFF(YEAR, users.birth_date, CURDATE()) BETWEEN 45 AND 54 THEN '45-54'
        WHEN TIMESTAMPDIFF(YEAR, users.birth_date, CURDATE()) BETWEEN 55 AND 64 THEN '55-64'
        ELSE '65+'
    END AS age_group,
    COUNT(id) AS user_count
FROM
    users
GROUP BY
    age_group
ORDER BY
    age_group;
-- bucket size a/c to age group
SELECT
    CASE
        WHEN TIMESTAMPDIFF(YEAR, u.birth_date, CURDATE()) BETWEEN 18 AND 24 THEN '18-24'
        WHEN TIMESTAMPDIFF(YEAR, u.birth_date, CURDATE()) BETWEEN 25 AND 34 THEN '25-34'
        WHEN TIMESTAMPDIFF(YEAR, u.birth_date, CURDATE()) BETWEEN 35 AND 44 THEN '35-44'
        WHEN TIMESTAMPDIFF(YEAR, u.birth_date, CURDATE()) BETWEEN 45 AND 54 THEN '45-54'
        WHEN TIMESTAMPDIFF(YEAR, u.birth_date, CURDATE()) BETWEEN 55 AND 64 THEN '55-64'
        ELSE '65+'
    END AS bucket_size,
    AVG(o.quantity) AS average_item_count
FROM
    users u
```

```
JOIN
    orders o ON u.id = o.user_id
GROUP BY
    bucket_size
ORDER BY
    bucket_size;

-- no. of users by every source
SELECT
    u.source,
    CASE
        WHEN TIMESTAMPDIFF(YEAR, u.birth_date, CURDATE()) BETWEEN 18 AND 24 THEN '18-24'
        WHEN TIMESTAMPDIFF(YEAR, u.birth_date, CURDATE()) BETWEEN 25 AND 34 THEN '25-34'
        WHEN TIMESTAMPDIFF(YEAR, u.birth_date, CURDATE()) BETWEEN 35 AND 44 THEN '35-44'
        WHEN TIMESTAMPDIFF(YEAR, u.birth_date, CURDATE()) BETWEEN 45 AND 54 THEN '45-54'
        WHEN TIMESTAMPDIFF(YEAR, u.birth_date, CURDATE()) BETWEEN 55 AND 64 THEN '55-64'
        ELSE '65+'
    END AS age_group,
    COUNT(*) AS user_count
FROM
    users u
GROUP BY
    u.source, age_group;


-- avg. amount spent ny every source
SELECT
    u.source,
    ROUND(AVG(o.total - COALESCE(o.discount, 0) * o.quantity), 0) AS avg_amount_spent
FROM
    users u
JOIN
    orders o ON u.id = o.user_id
GROUP BY
    u.source;


-- top 5 customer
SELECT
    u.id AS user_id,
    (name) AS customer_name,
    users.ï»¿created_at AS joining_date,
    COUNT(o.id) AS order_count,
    ROUND(AVG(SUM((o.total - COALESCE(o.discount, 0)) * o.quantity),0), 2) AS avg_amount_spent,
    DATEDIFF(CURRENT_DATE, u.created_at) AS tenure_days,
    COUNT(o.id) / DATEDIFF(CURRENT_DATE, u.created_at) AS order_frequency,
    (DATEDIFF(CURRENT_DATE, u.ï»¿created_at) * COUNT(o.id) * AVG(o.total)) / 1000 AS composite_score
FROM
    users u
JOIN
```

```
    orders o ON u.id = o.user_id
GROUP BY
    user_id, customer_name, creation_date, tenure_days
ORDER BY
    composite_score DESC
LIMIT 10;

-- WOULD BE CUSTOMERS
SELECT
    COUNT(DISTINCT u.id) AS total_customers,
    COUNT(DISTINCT o.user_id) AS customers_shopped,
    COUNT(DISTINCT u.id) - COUNT(DISTINCT o.user_id) AS customers_did_not_shop
FROM
    users u
LEFT JOIN
    orders o ON u.id = o.user_id;

-- PRODUCT ANALYSIS
SELECT
    p.id AS product_id,
    p.title AS product_title,
    p.category AS product_category,
    p.rating AS product_rating,
    COUNT(o.ï»¿id) AS order_count,
    ROUND(SUM((o.total - COALESCE(o.discount, 0)) * o.quantity), 0) AS total_sales,
    SUM(o.subtotal - p.price) AS profit,
    (100 - (SUM(o.subtotal) / SUM(o.total)) * 100) AS profit_margin
FROM
    products p
LEFT JOIN
    orders o ON p.id = o.product_id
GROUP BY
    p.id,
    product_title,
    product_category,
    product_rating;


-- VENDOR ANALYSIS
SELECT
    p.vendor,
    COUNT(p.id) AS total_products,
    AVG(p.rating) AS average_rating,
    p.category,
    p.title AS product_name,
    AVG(o.total) AS average_price,
    AVG(o.subtotal) AS average_profit,
    SUM(o.quantity) AS total_items_sold,
    (100 - (AVG(o.subtotal) / AVG(o.total)) * 100) AS profit_margin
FROM
```

```
    products p
JOIN
    orders o ON p.id = o.product_id
GROUP BY
    p.vendor,
    p.category,
    p.title
ORDER BY
    p.vendor;


-- CUSTOMER LIFETIME VALUE
WITH CustomerLifetime AS (
    SELECT
        user_id,
        DATEDIFF(MAX(created_at), MIN(created_at)) AS customer_lifetime
    FROM
        orders
    GROUP BY
        user_id
)
SELECT
    AVG(customer_lifetime) AS average_customer_lifetime
FROM
    CustomerLifetime;


-- STATE ANALYSIS
-- STATE ANALYSIS
WITH product_sales AS (
    SELECT
        YEAR(o.created_at) AS year,
        MONTH(o.created_at) AS month,
        DATE_FORMAT(o.created_at, '%M') AS month_name,
        p.category,
        p.title AS product_title,
        u.state,
        u.city,
        COUNT(o.ï»¿id) AS order_count,
        ROUND(SUM(o.total - COALESCE(o.discount, 0)), 2) AS total_sales
    FROM
        orders o
    JOIN
        products p ON o.product_id = p.id
    JOIN
        users u ON o.user_id = u.id
    GROUP BY
        year, month, month_name, p.category, product_title, u.state, u.city
)
SELECT
```

```
    year,
    month,
    month_name,
    category,
    product_title,
    state,
    city,
    order_count,
    total_sales
FROM
    product_sales;
```