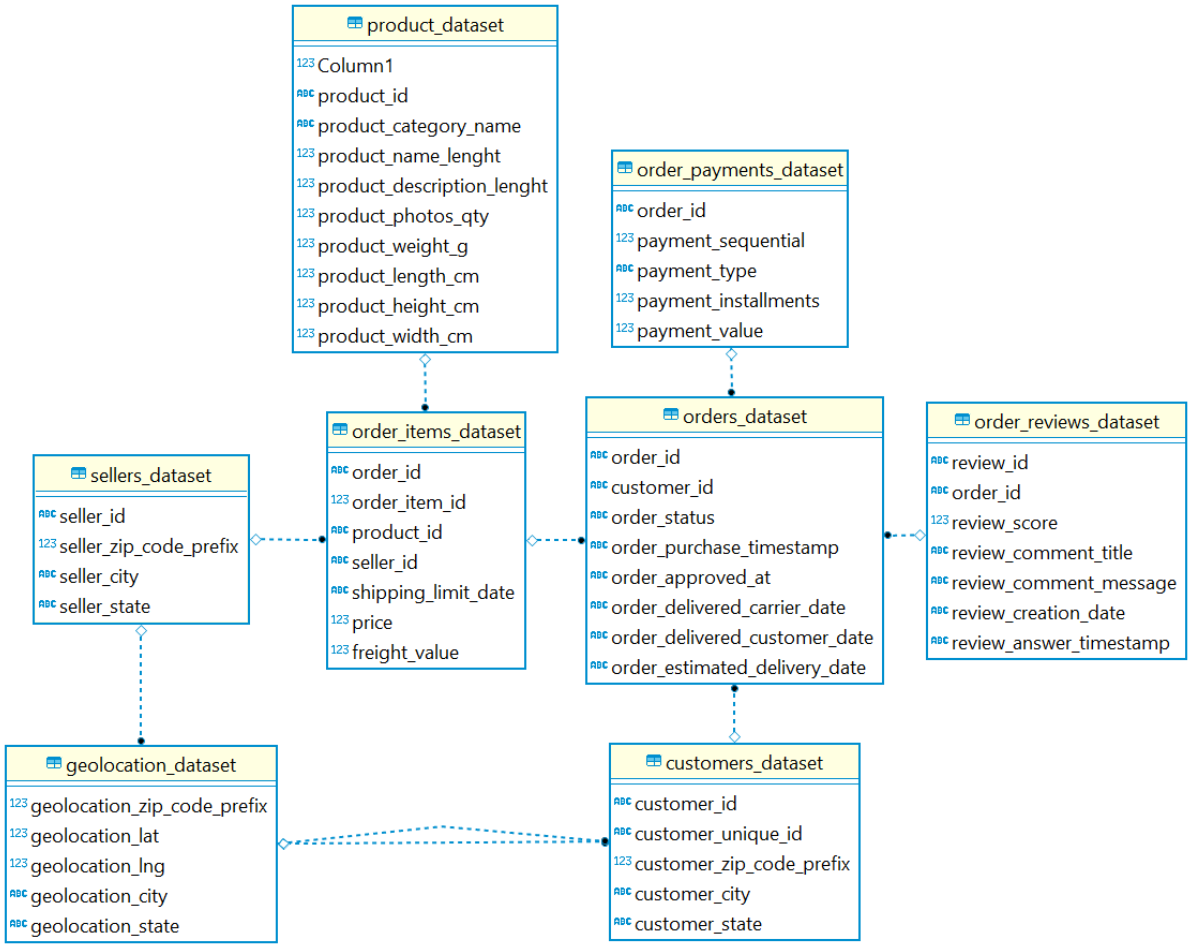
Analyzing eCommerce Business Performance with SQL

Created by: Dinda Galuh

eCommerce Entity Relationship Diagram (ERD)



Annual Customer Activity Growth Analysis

- Rata-rata Monthly Active User (MAU) per tahun,
- Total customer baru per tahun,
- Jumlah customer yang melakukan repeat order per tahun,
- Rata-rata frekuensi order untuk setiap tahun.

```
WITH
MonthlyActiveUsers AS (
    SELECT
        EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
        EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
        COUNT(DISTINCT customer_id) AS MAU
    FROM
```

```
        orders_dataset
    GROUP BY 1,2
),
FirstPurchaseYear AS (
    SELECT
        cd.customer_unique_id,
        EXTRACT(YEAR FROM MIN(order_purchase_timestamp)) AS first_year
    FROM
        orders_dataset od
    JOIN customers_dataset cd
      on od.customer_id = cd.customer_id
    GROUP BY 1
),
YearlyOrderCounts AS (
    SELECT
        customer_unique_id,
        EXTRACT(YEAR FROM order_purchase_timestamp) AS order_year,
        COUNT(DISTINCT order_purchase_timestamp) AS order_count
    FROM
        orders_dataset od
    JOIN customers_dataset cd
    ON od.customer_id = cd.customer_id
    GROUP BY
        customer_unique_id,
        EXTRACT(YEAR FROM order_purchase_timestamp)
),
YearlyOrderFrequency AS (
    SELECT
        EXTRACT(YEAR FROM order_purchase_timestamp) AS order_year,
        COUNT(DISTINCT order_purchase_timestamp) AS order_count
    FROM
        orders_dataset od
    JOIN customers_dataset cd
    ON od.customer_id = cd.customer_id
    GROUP BY
        customer_unique_id,
        EXTRACT(YEAR FROM order_purchase_timestamp)
)

-- Menggabungkan semua hasil
SELECT
    mau.year,
    FLOOR(AVG(mau.MAU)) AS average_MAU,
    fpy.total_new_customers,
    yoc.repeat_customers,
    yof.average_order_frequency
FROM
    MonthlyActiveUsers mau
LEFT JOIN (
    SELECT
        first_year,
        COUNT(customer_unique_id) AS total_new_customers
    FROM
        FirstPurchaseYear
    GROUP BY
        first_year
) fpy ON mau.year = fpy.first_year
LEFT JOIN (
    SELECT
        order_year,
        COUNT(DISTINCT customer_unique_id) AS repeat_customers
```

```
    FROM
        YearlyOrderCounts
    WHERE
        order_count > 1
    GROUP BY
        order_year
) yoc ON mau.year = yoc.order_year
LEFT JOIN (
    SELECT
        order_year,
        AVG(order_count) AS average_order_frequency
    FROM
        YearlyOrderFrequency
    GROUP BY order_year
) yof ON mau.year = yof.order_year
GROUP BY
    mau.year, fpy.total_new_customers, yoc.repeat_customers,
yof.average_order_frequency
ORDER BY
    mau.year;
```

	year	average_MAU	total_new_customers	repeat_customers	average_order_frequency
1	2,016	109	326	3	1.0092
2	2,017	3,758	43,708	1,111	1.028
3	2,018	5,401	52,062	1,046	1.0215

Annual Product Category Quality Analysis

- Revenue per tahun
- Jumlah cancel order per tahun
- Top kategori yang menghasilkan revenue terbesar per tahun
- Kategori yang mengalami cancel order terbanyak per tahun

```
WITH RevenuePerYear AS (
    -- revenue per tahun
    SELECT
        EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
        ROUND(SUM(price + freight_value),2) AS total_revenue
    FROM
        orders_dataset od
    JOIN
        order_items_dataset oid
    ON oid.order_id = od.order_id
    WHERE od.order_status = 'delivered'
    GROUP BY
        EXTRACT(YEAR FROM order_purchase_timestamp)
),
CancelOrderPerYear AS (
    -- jumlah cancel order per tahun
    SELECT
        EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
        COUNT(order_id) AS cancel_orders_count
    FROM
        orders_dataset
    WHERE
```

```
        order_status = 'canceled'
    GROUP BY
        EXTRACT(YEAR FROM order_purchase_timestamp)
),
TopCategoryPerYear AS (
    -- subquery untuk menghitung revenue dan ranking
    WITH RankedRevenue AS (
        SELECT
            EXTRACT(YEAR FROM od.order_purchase_timestamp) AS year,
            pd.product_category_name,
            ROUND(SUM(oid.price + oid.freight_value),2) AS total_revenue,
            ROW_NUMBER() OVER (PARTITION BY EXTRACT(YEAR FROM
od.order_purchase_timestamp) ORDER BY ROUND(SUM(oid.price +
oid.freight_value),2) DESC) AS ranking
        FROM
            orders_dataset od
        JOIN
            order_items_dataset oid
        ON
            oid.order_id = od.order_id
        JOIN
            product_dataset pd
        ON
            oid.product_id = pd.product_id
        WHERE
            od.order_status = 'delivered'
        GROUP BY
            EXTRACT(YEAR FROM od.order_purchase_timestamp),
            pd.product_category_name
    )
    SELECT
        year,
        product_category_name AS top_category_revenue,
        total_revenue AS top_category_revenue_amount
    FROM
        RankedRevenue
    WHERE
        ranking = 1
),
MostCanceledCategoryPerYear AS (
    -- subquery untuk menghitung cancel count dan ranking
    WITH RankedCancel AS (
        SELECT
            EXTRACT(YEAR FROM od.order_purchase_timestamp) AS year,
            pd.product_category_name,
            COUNT(od.order_id) AS cancel_count,
            ROW_NUMBER() OVER (PARTITION BY EXTRACT(YEAR FROM
od.order_purchase_timestamp) ORDER BY COUNT(od.order_id) DESC) AS ranking
        FROM
            orders_dataset od
        JOIN
            order_items_dataset oid
        ON
            od.order_id = oid.order_id
        JOIN
            product_dataset pd
        ON
            oid.product_id = pd.product_id
        WHERE
            od.order_status = 'canceled'
        GROUP BY
```

```
            EXTRACT(YEAR FROM od.order_purchase_timestamp),
            pd.product_category_name
    )
    SELECT
        year,
        product_category_name AS most_canceled_category,
        cancel_count AS most_canceled_count
    FROM
        RankedCancel
    WHERE
        ranking = 1
)
SELECT
    r.year,
    r.total_revenue,
    c.cancel_orders_count,
    t.top_category_revenue,
    t.top_category_revenue_amount,
    m.most_canceled_category,
    m.most_canceled_count
FROM
    RevenuePerYear r
JOIN
    CancelOrderPerYear c ON r.year = c.year
JOIN
    TopCategoryPerYear t ON r.year = t.year
JOIN
    MostCanceledCategoryPerYear m ON r.year = m.year
ORDER BY
    r.year;
```

	year	total_revenue	cancel_orders_count	top_category_revenue	top_category_revenue_amoun	most_canceled_category	most_canceled_count
1	2,016	46,653.74	26	furniture_decor	6,899.35	toys	3
2	2,017	6,921,535.24	265	bed_bath_table	580,949.2	sports_leisure	25
3	2,018	8,451,584.77	334	health_beauty	866,810.34	health_beauty	27

Analysis of Annual Payment Type Usage

```
WITH YearlyPayments AS (
    SELECT
        EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
        op.payment_type,
        COUNT(op.order_id) AS yearly_count,
        SUM(op.payment_value) AS yearly_value
    FROM
        order_payments_dataset op
    JOIN
        orders_dataset o ON op.order_id = o.order_id
    WHERE op.payment_type <> 'not_defined'
    GROUP BY
        EXTRACT(YEAR FROM o.order_purchase_timestamp),
        op.payment_type
),

TotalPayments AS (
    SELECT
        payment_type,
        COUNT(order_id) AS total_count,
        SUM(payment_value) AS total_value
    FROM
```

```
        order_payments_dataset
    GROUP BY
        payment_type
)

SELECT
    yp.year,
    yp.payment_type,
    yp.yearly_count,
    tp.total_count,
    ROUND(yp.yearly_value,2) AS yearly_value,
    ROUND(tp.total_value,2) AS total_value
FROM
    YearlyPayments yp
JOIN
    TotalPayments tp ON yp.payment_type = tp.payment_type
ORDER BY
    tp.total_count DESC, yp.year, yp.payment_type;
```

	year	payment_type	yearly_count	total_count	yearly_value	total_value
1	2,016	credit_card	258	76,795	48,562.48	12,542,084.19
2	2,017	credit_card	34,568	76,795	5,637,373.94	12,542,084.19
3	2,018	credit_card	41,969	76,795	6,856,147.77	12,542,084.19
4	2,016	boleto	63	19,784	9,679.06	2,869,361.27
5	2,017	boleto	9,508	19,784	1,396,063.37	2,869,361.27
6	2,018	boleto	10,213	19,784	1,463,618.84	2,869,361.27
7	2,016	voucher	23	5,775	879.07	379,436.87
8	2,017	voucher	3,027	5,775	172,982.95	379,436.87
9	2,018	voucher	2,725	5,775	205,574.85	379,436.87
10	2,016	debit_card	2	1,529	241.73	217,989.79
11	2,017	debit_card	422	1,529	43,326.47	217,989.79
12	2,018	debit_card	1,105	1,529	174,421.59	217,989.79