ANALYZING ECOMMERCE BUSINESS PERFORMANCE WITH SQL

FERI DWI SAPUTRO

23 April 2022 - 29 April 2022

1. Project Overview

Dalam suatu perusahaan mengukur performa bisnis sangatlah penting untuk melacak, memantau, dan menilai keberhasilan atau kegagalan dari berbagai proses bisnis. Oleh karena itu, dalam projek ini akan menganalisa performa bisnis untuk sebuah perusahaan e-commerce, dengan memperhitungkan beberapa metrik bisnis yaitu pertumbuhan pelanggan, kualitas produk, dan tipe pembayaran. Perusahaan ini merupakan salah satu marketplace terbesar di Amerika Selatan yang menghubungkan pelaku bisnis mikro dengan para pelanggannya.

2. Dataset

Ada 8 dataset yang berbentuk file csv untuk digunakan dalam analisis ini, yaitu:

- **geolocation_dataset** : berisi data lokasi
- **customers_dataset** : berisi data profil pelanggan
- **sellers_dataset** : berisi data profil penjual
- **product_dataset** : berisi data produk yang dijual
- **orders_dataset** : berisi data pemesanan produk
- **order_items_dataset** : berisi data detail transaksi
- **order_payments_dataset** : berisi data pembayaran pesanan
- **order_reviews_dataset** : berisi data ulasan pesanan

3. Data Preparation

Sebelum memulai analisis data, sebaiknya kita mempersiapkan data yang dibutuhkan terlebih dahulu. Disini saya akan menggunakan tools **PostgreSQL** untuk melakukan analisis data. Adapun tahapan yang dilakukan, antara lain:

a. **Membuat Database** menggunakan pgAdmin menu
   - Klik kanan pada Databases
   - Pilih Create -> Database
   - Pada form Database, isi nama database yang diinginkan
   - Klik Save dan database sudah selesai dibuat

b. **Membuat Tabel** dengan perintah CREATE TABLE
   - **Tabel geolocation dataset**

```
CREATE TABLE IF NOT EXISTS geolocation_dataset(
    geolocation_zip_code_prefix CHAR(5),
    geolocation_lat DOUBLE PRECISION,
    geolocation_lng DOUBLE PRECISION,
    geolocation_city VARCHAR(50),
    geolocation_state CHAR(2)
);
```

- **Tabel customers_dataset**

```
CREATE TABLE IF NOT EXISTS customers_dataset(
    customer_id UUID,
    customer_unique_id UUID,
    customer_zip_code_prefix CHAR(5),
    customer_city VARCHAR(50),
    customer_state CHAR(2)
);
```

- **Tabel sellers_dataset**

```
CREATE TABLE IF NOT EXISTS sellers_dataset(
    seller_id UUID,
    seller_zip_code_prefix CHAR(5),
    seller_city VARCHAR(50),
    seller_state CHAR(2)
);
```

- **Tabel products_dataset**

```
CREATE TABLE IF NOT EXISTS products_dataset(
    product_no SERIAL,
    product_id UUID,
    product_category_name VARCHAR(50),
    product_name_length REAL,
    product_description_length REAL,
    product_photos_qty REAL,
    product_weight_g REAL,
    product_length_cm REAL,
    product_height_cm REAL,
    product_width_cm REAL
);
```

- **Tabel orders_dataset**

```
CREATE TABLE IF NOT EXISTS orders_dataset(
    order_id UUID,
    customer_id UUID,
    order_status VARCHAR(20),
    order_purchase_timestamp TIMESTAMP,
    order_approved_at TIMESTAMP,
    order_delivered_carrier_date TIMESTAMP,
    order_delivered_customer_date TIMESTAMP,
    order_estimated_delivery_date TIMESTAMP
);
```

- **Tabel order_items_dataset**

```
CREATE TABLE IF NOT EXISTS order_items_dataset(
    order_id UUID,
    order_item_id INTEGER,
    product_id UUID,
    seller_id UUID,
    shipping_limit_date TIMESTAMP,
    price NUMERIC(7,2),
    freight_value NUMERIC(5,2)
);
```

- **Tabel order_payments_dataset**

```
CREATE TABLE IF NOT EXISTS order_payments_dataset(
    order_id UUID,
    payment_sequential INTEGER,
    payment_type VARCHAR(20),
    payment_installments INTEGER,
    payment_value NUMERIC(7,2)
);
```

- **Tabel order_reviews dataset**

```
CREATE TABLE IF NOT EXISTS order_reviews_dataset(
    review_id UUID,
    order_id UUID,
    review_score INTEGER,
    review_comment_title VARCHAR(255),
    review_comment_message TEXT,
    review_creation_date TIMESTAMP,
    review_answer_timestamp TIMESTAMP
);
```

c. **Mengimpor data** menggunakan file CSV untuk tiap tabel dengan pgAdmin menu
   - Pilih nama tabel yang sesuai
   - Klik kanan, lalu pilih Import/Export
   - Pilih toggle Import, lalu masukkan file path di local file
   - Use Header karena baris pertama adalah nama field dan pilih pemisah antar data
   - Klik OK dan tunggu notifikasi bahwa file sukses di import.

d. **Menambahkan relasi** antar tabel menggunakan perintah ALTER TABLE.

- **Membuat Primary Key (PK)**

Primary Key adalah nilai unik untuk yang dimiliki oleh tabel sebagai identifier tiap barisnya sehingga tidak boleh ada duplikat. Berikut adalah query yang dibuat.

```
ALTER TABLE customers_dataset
ADD CONSTRAINT PK_customer_id PRIMARY KEY (customer_id);
ALTER TABLE products_dataset
ADD CONSTRAINT PK_product_id PRIMARY KEY (product_id);
ALTER TABLE sellers_dataset
ADD CONSTRAINT PK_seller_id PRIMARY KEY (seller_id);
ALTER TABLE orders_dataset
ADD CONSTRAINT PK_order_id PRIMARY KEY (order_id);
```

- **Membuat Not Null Constraint**

```
ALTER TABLE geolocation_dataset
ALTER COLUMN geolocation_zip_code_prefix SET NOT NULL;
ALTER TABLE order_items_dataset
ALTER COLUMN order_item_id SET NOT NULL;
ALTER TABLE order_reviews_dataset
ALTER COLUMN review_id SET NOT NULL;
```

- **Membuat Foregin Key (FK)**

Foreign Key adalah nilai key yang menghubungkan ke tabel induk (Primary Key) dan boleh lebih dari satu nilai identifier atau duplikat.

```
ALTER TABLE orders_dataset
ADD CONSTRAINT FK_customer_id FOREIGN KEY (customer_id)
REFERENCES customers_dataset(customer_id)
ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE order_items_dataset
ADD CONSTRAINT FK_seller_id FOREIGN KEY (seller_id)
REFERENCES sellers_dataset(seller_id)
ON DELETE CASCADE ON UPDATE CASCADE,
ADD CONSTRAINT FK_product_id FOREIGN KEY (product_id)
REFERENCES products_dataset(product_id)
ON DELETE CASCADE ON UPDATE CASCADE,
ADD CONSTRAINT FK_order_id FOREIGN KEY (order_id)
REFERENCES orders_dataset(order_id)
ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE order_reviews_dataset
ADD CONSTRAINT FK_order_id FOREIGN KEY (order_id)
REFERENCES orders_dataset(order_id)
ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE order_payments_dataset
ADD CONSTRAINT FK_order_id FOREIGN KEY (order_id)
REFERENCES orders_dataset(order_id)
ON DELETE CASCADE ON UPDATE CASCADE;
```

4. Data Analysis

4.1 Annual Customer Growth Analysis

Salah satu metrik yang digunakan untuk mengukur performa bisnis eCommerce adalah aktivitas customer yang berinteraksi di dalam platform eCommerce tersebut. Disini kita akan menganalisis beberapa metrik yang berhubungan dengan aktivitas customer seperti jumlah customer aktif, jumlah customer baru, jumlah customer yang melakukan repeat order dan juga rata-rata transaksi yang dilakukan customer setiap tahun. Berikut adalah query yang dibuat menggunakan CTE dan SubQuery.

```
WITH
      --Create Monthly Active User Data
      monthly_active_user_data AS (
            SELECT
                  tahun,
                  ROUND(AVG(active_user), 3) AS rata_rata_mau
            FROM(
                  SELECT
                        DATE_PART('year',
orders.order_purchase_timestamp) AS tahun,
```

```
                        DATE_PART('month',
orders.order_purchase_timestamp) AS bulan,
                        COUNT(DISTINCT customers.customer_unique_id)
AS active_user
                  FROM
                        orders_dataset AS orders
                  JOIN customers_dataset AS customers
                        ON orders.customer_id = customers.customer_id
                  GROUP BY 1,2
            ) monthly_active_user
            GROUP BY 1
      ),
      --Create New Customer Data
      new_customer_data AS (
            SELECT
                  DATE_PART('year', first_transaction) AS tahun,
                  COUNT(1) AS new_customer
            FROM (
                  SELECT
                        customers.customer_unique_id AS customer,
                        MIN(orders.order_purchase_timestamp) AS
first_transaction
                  FROM
                        orders_dataset AS orders
                  JOIN customers_dataset AS customers
                        ON orders.customer_id = customers.customer_id
                  GROUP BY 1
            ) transactions
            GROUP BY 1
      ),
      --Create Repeat Order Data
      repeat_order_data AS (
            SELECT
                  tahun,
                  COUNT(customer) AS repeat_customer
            FROM (
                  SELECT
                        DATE_PART('year',
orders.order_purchase_timestamp) AS tahun,
                        customers.customer_unique_id AS customer,
                        COUNT(1) as total_transaction
                  FROM
                        orders_dataset AS orders
                  JOIN customers_dataset AS customers
                        ON orders.customer_id = customers.customer_id
                  GROUP BY 1,2
                  HAVING COUNT(1) > 1
            ) repeat_order
            GROUP BY 1
      ),
      --Create Average of Transactions Data
```

```
    avg_transaction_data AS (
        SELECT
            tahun,
            ROUND(AVG(total_transaction), 3) AS
rata_rata_transaksi
        FROM (
            SELECT
                DATE_PART('year',
orders.order_purchase_timestamp) AS tahun,
                customers.customer_unique_id AS customer,
                COUNT(1) as total_transaction
            FROM
                orders_dataset AS orders
            JOIN customers_dataset AS customers
                ON orders.customer_id = customers.customer_id
            GROUP BY 1,2
        ) transactions
        GROUP BY 1
    )

--Join All CTE tables
SELECT
    mau.tahun as tahun,
    rata_rata_mau,
    new_customer,
    repeat_customer,
    rata_rata_transaksi
FROM monthly_active_user_data AS mau
JOIN new_customer_data AS nc
ON mau.tahun = nc.tahun
JOIN repeat_order_data AS ro
ON mau.tahun = ro.tahun
JOIN avg_transaction_data AS trx
ON mau.tahun = trx.tahun;
```

- Highlight query yang berwarna orange untuk menghasilkan rata-rata *Monthly Active User* per tahun dengan cara mencari *active user* per bulan terlebih dahulu kemudian baru dikelompokkan per tahun nya berdasarkan rata-rata.
- Highlight query yang berwarna hijau untuk menghasilkan jumlah pelanggan baru yang didapatkan per tahun dengan cara mencari transaksi pertama yang dilakukan pelanggan dahulu kemudian baru dikelompokkan per tahun nya berdasarkan jumlahnya.
- Highlight query yang berwarna biru untuk menghasilkan jumlah pelanggan yang melakukan order lebih dari sekali atau *repeat order*) per tahun dengan cara menghitung total transaksi yang tergolong *repeat order* terlebih dahulu kemudian dihitung banyaknya pelanggan *repeat order* lalu dikelompokkan per tahun nya berdasarkan jumlahnya.
- Highlight query yang berwarna ungu untuk menghasilkan rata-rata transaksi per tahun dengan cara menghitung total transaksi yang telah terjadi kemudian dikelompokkan per tahun nya berdasarkan rata-rata.
- Highlight yang berwarna merah untuk menggabungkan keempat hasil metric untuk diambil kesimpulan menggunakan perintah JOIN pada keempat data tersebut berdasarkan kolom tahun.

4.2 Annual Product Category Quality Analysis

Performa bisnis eCommerce tentunya sangat berkaitan erat dengan produk-produk yang tersedia di dalamnya. Menganalisis kualitas dari produk dalam eCommerce dapat memberikan keputusan untuk mengembangkan bisnis dengan lebih baik. Disini kita akan menganalisis performa dari masing-masing kategori produk yang ada dan bagaimana kaitannya dengan pendapatan perusahaan. Disini akan digunakan perintah **CREATE TABLE AS dan WINDOW FUNCTION** untuk mempermudah analisisnya.

- **Total Revenue by Year**

```
CREATE TABLE revenue_by_year AS
      SELECT
            year_transaction,
            SUM(price + freight_value) AS total_revenue
      FROM order_items_dataset AS details
      JOIN (
            SELECT
                  order_id,
                  DATE_PART('year', order_purchase_timestamp)
AS year_transaction
            FROM orders_dataset
            WHERE order_status = 'delivered'
      ) AS orders
      ON details.order_id = orders.order_id
      GROUP BY 1
      ORDER BY 1;
```

Revenue dihitung dari penjumlahan harga barang dan ongkos kirim. Pertama mencari semua transaksi yang sudah berstatus **delivered** kemudian total revenue dihitung menggunakan fungsi agregasi **SUM** untuk dikelompokkan berdasarkan tahun transaksi.

- **Total Canceled Order by Year**

```
CREATE TABLE canceled_order_by_year AS
      SELECT
            DATE_PART('year', order_purchase_timestamp) AS
year_transaction,
            COUNT(order_id) AS total_canceled_order
      FROM orders_dataset
      WHERE order_status = 'canceled'
      GROUP BY 1
      ORDER BY 1;
```

Cancel Order dihitung berdasarkan jumlah transaksi yang berstatus **canceled** yang dikelompokkan berdasarkan tahun transaksi menggunakan fungsi agregasi **COUNT**.

- **Highest Revenue by Category**

```
CREATE TABLE top_category_revenue_by_year AS
      SELECT
```

```
        year_transaction,
        product_category_name,
        revenue
    FROM (
        SELECT
            year_transaction,
            product_category_name,
            SUM(price+freight_value) AS revenue,
            RANK() OVER(
                PARTITION BY year_transaction
                ORDER BY SUM(price+freight_value) DESC
            ) AS revenue_rank
        FROM order_items_dataset AS details
        JOIN (
            SELECT
                order_id,
                DATE_PART('year',
order_purchase_timestamp) AS year_transaction
            FROM orders_dataset
            WHERE order_status = 'delivered'
        ) AS orders
        ON details.order_id = orders.order_id
        JOIN products_dataset AS products
        ON details.product_id = products.product_id
        GROUP BY 1,2
    ) AS category_revenue
    WHERE revenue_rank = 1;
```

Untuk mencari **kategori dengan revenue terbesar** kita perlu melakukan filter terlebih dahulu pada semua transaksi yang berstatus **delivered** di tabel orders kemudian melakukan **JOIN terhadap tabel products** untuk mendapatkan nama kategorinya. Selanjutnya revenue dihitung menggunakan fungsi agregasi **SUM antara kolom harga dan ongkos kirim** lalu dikelompokkan berdasarkan tahun dan kategori. Kemudian untuk **ranking diurutkan berdasarkan revenue terbesar dan dipilih yang teratas tiap tahunnya**.

- **Highest Canceled Order by Category**

```
CREATE TABLE top_category_canceled_order_by_year AS
    SELECT
        year_transaction,
        product_category_name,
        canceled_order
    FROM (
        SELECT
            year_transaction,
            product_category_name,
            COUNT(details.order_id) AS canceled_order,
            RANK() OVER(
                PARTITION BY year_transaction
                ORDER BY COUNT(details.order_id) DESC
            ) AS cancel_rank
```

```
            FROM order_items_dataset AS details
            JOIN (
                  SELECT
                        order_id,
                        DATE_PART('year',
order_purchase_timestamp) AS year_transaction
                  FROM orders_dataset
                  WHERE order_status = 'canceled'
            ) AS orders
            ON details.order_id = orders.order_id
            JOIN products_dataset AS products
            ON details.product_id = products.product_id
            GROUP BY 1,2
      ) AS canceled_transactions
      WHERE cancel_rank = 1;
```

Untuk mencari **kategori dengan cancel order tertinggi** kita perlu melakukan filter terlebih dahulu pada semua transaksi yang berstatus **canceled** di tabel orders kemudian melakukan **JOIN terhadap tabel products** untuk mendapatkan nama kategorinya. Selanjutnya cancel order dihitung menggunakan fungsi agregasi **COUNT pada kolom order id** lalu dikelompokkan berdasarkan tahun dan kategori. Kemudian untuk **ranking diurutkan berdasarkan cancel order tertinggi dan dipilih yang teratas tiap tahunnya**.

- **Join All Result About Revenue and Canceled Order**

```
SELECT
      rv.year_transaction AS year_transaction,
      rv.total_revenue AS total_revenue,
      crv.product_category_name AS highest_revenue_category,
      crv.revenue AS highest_revenue,
      co.total_canceled_order AS total_canceled_order,
      cco.product_category_name AS
highest_canceled_order_category,
      cco.canceled_order AS highest_canceled_order
FROM revenue_by_year AS rv
JOIN canceled_order_by_year AS co
ON rv.year_transaction = co.year_transaction
JOIN top_category_revenue_by_year AS crv
ON co.year_transaction = crv.year_transaction
JOIN top_category_canceled_order_by_year AS cco
ON crv.year_transaction = cco.year_transaction;
```

Hasil keempat query sebelumnya akan digabungkan agar analisis tentang kualitas produk menjadi lebih mudah untuk diinterpretasikan kesimpulannya.

4.3 Annual Payment Type Usage Analysis

Bisnis eCommerce umumnya menyediakan sistem pembayaran berbasis *open-payment* yang memungkinkan customer untuk memilih berbagai macam tipe pembayaran yang tersedia. Dengan menganalisis performa dari tipe pembayaran yang ada dapat memberikan

insight untuk menciptakan *strategic partnership* dengan perusahaan penyedia jasa pembayaran dengan lebih baik. Maka kita akan menganalisis tipe-tipe pembayaran yang tersedia dan melihat tren perubahan yang terjadi selama beberapa tahun terakhir. Disini kita akan memanfaatkan perintah **CASE WHEN** untuk mempermudah breakdownnya.

```
WITH all_time_usage AS (
      SELECT
            payment_type,
            COUNT(1) AS total_usage
      FROM orders_dataset AS orders
      JOIN order_payments_dataset AS payments
      ON orders.order_id = payments.order_id
      GROUP BY 1
      ORDER BY 2 DESC
)

SELECT
      year_usage.*,
      total_usage
FROM all_time_usage
JOIN (
      SELECT
            payment_type,
            SUM(CASE year_transaction WHEN 2016 THEN 1 ELSE 0 END)
AS usage_2016,
            SUM(CASE year_transaction WHEN 2017 THEN 1 ELSE 0 END)
AS usage_2017,
            SUM(CASE year_transaction WHEN 2018 THEN 1 ELSE 0 END)
AS usage_2018
      FROM (
            SELECT
                  DATE_PART('year', order_purchase_timestamp) AS
year_transaction,
                  payment_type
            FROM orders_dataset AS orders
            JOIN order_payments_dataset AS payments
            ON orders.order_id = payments.order_id
      ) AS payment_by_year
      GROUP BY 1
) AS year_usage
ON all_time_usage.payment_type = year_usage.payment_type
ORDER BY total_usage DESC
```

- Highlight query yang berwarna merah untuk mencari jumlah penggunaan tiap tipe pembayaran secara all time dan diurutkan berdasarkan yang terbanyak.
- Highlight query yang berwarna biru untuk mencari jumlah penggunaan tiap tipe pembayaran yang telah di breakdown berdasarkan tahun. Pertama menggabungkan tabel order dan payment untuk mendapatkan data tahun dan tipe pembayaran sesuai order id nya. Selanjutnya menghitung jumlah penggunaan setiap tipe pembayaran dengan fungsi **CASE WHEN** per tahun nya. Lalu menggabungkan dengan hasil

sebelumnya agar sesuai tampilan yang diinginkan yaitu breakdown tiap tahun untuk semua tipe pembayaran.

-

5. Data Visualization

Dari hasil semua analisis yang telah dilakukan mengenai pertumbuhan pelanggan, kualitas produk, dan penggunaan pembayaran akan digunakan visualisasi menggunakan Google Data Studio untuk mempermudah memahami insight dan memberikan saran. Untuk link visualisasi dapat Anda lihat disini.

6. Conclusion

- **Pertumbuhan pelanggan** selama 2016-2018 masih perlu ditingkatkan karena kebanyakan transaksi yang terjadi dilakukan oleh pelanggan baru.
- **Kualitas produk** selama 2016-2018 memberikan revenue yang cukup baik dengan top kategori yang beragam, terutama pada tahun 2018 sepertinya **Health Beauty** memiliki demand yang cukup besar.
- **Penggunaan tipe pembayaran** selama 2016-2018 cukup bagus hampir di setiap tipe pembayaran kecuali Voucher yang perlu dianalisis lebih lanjut atau diperbarui kedepannya.