Fictional Sales Data: SQL EDA

Вступление

В данном мини-проекте мы будем производить разведочный анализ данных фиктивного набора данных о продажах.

Набор данных содержит 3 таблицы, покрывающих временной отрезок с января 2023 по май 2024:

1. **Customers** (идентификатор пользователя, возраст, адрес, период валидности профиля)
2. **Products** (идентификатор, наименование, цена, срок валидности цены)
3. **Orders** (идентификатор, продукт, пользователь, количество, дата).

Источник данных

Оригинальный датасет доступен на сайте Kaggle по ссылке.

NB: все даты оригинального датасета в рамках данного проекта были обновлены с использованием формул рандомизации Excel.

Описание данных

customers

customer_id - идентификатор пользователя

customer_address - адрес пользователя

customer_age - возраст пользователя

effective_start_date - начало периода валидности профиля пользователя

effective_end_date - окончание периода валидности профиля пользователя

products

product_id - идентификатор продукта

product_name - наименование продукта

product_price - цена

effective_start_date - начало периода валидности цены

effective_end_date - окончание периода валидности цены

orders

order_id - идентификатор покупки

product_id - идентификатор продукта

customer_id - идентификатор пользователя

product_quantity - количество продукта

order_date - дата покупки

Работа с данными

1. Создадим временную таблицу на основе джойна таблиц *orders* and *products*, чтобы она отображала сумму покупки с учетом валидной на момент осуществления покупки цены.

```
CREATE TEMP TABLE orders_temp AS
SELECT o.*,
    p.product_price,
    o.product_quantity * p.product_price AS purchase_sum
FROM orders AS o
LEFT JOIN products AS p
ON o.product_id = p.product_id
    AND o.order_date BETWEEN p.effective_start_date AND p.effective_end_date;
```

	order_id integer	product_id integer	customer_id integer	product_quantity integer	order_date date	product_price double precision	purchase_sum double precision
1	141234	582	174831	1	2023-07-18	700	700
2	141235	981	105312	1	2023-08-22	12.99	12.99
3	141236	277	209194	2	2023-02-25	15	30
4	141237	443	143165	1	2023-05-27	139	139
5	141238	277	120517	1	2023-03-24	12.99	12.99

2. Интересно, а когда мы смогли достичь суммы 100000 по продажам?

```
WITH t1 AS
(SELECT order_id,
    order_date,
    ROUND(SUM(purchase_sum) OVER (ORDER BY order_date)::NUMERIC, 2) AS total_sal
es
FROM orders_temp
ORDER BY order_date)
SELECT order_id,
    order_date,
    total_sales
FROM t1
WHERE total_sales >= 100000
ORDER BY total_sales
LIMIT 1;
```

	order_id integer	order_date date	total_sales numeric
1	238456	2023-01-17	103423.21

3. Посчитаем скользящее среднее суммы продаж за последние 2 доступных месяца:

```
SELECT *
FROM
```

```
(SELECT order_date,
    ROUND(SUM(SUM(purchase_sum)) OVER (ORDER BY order_date ROWS BETWEEN 6 PRECED
ING AND CURRENT ROW)::NUMERIC, 2) AS total_sales,
    ROUND(AVG(SUM(purchase_sum)) OVER (ORDER BY order_date ROWS BETWEEN 6 PRECED
ING AND CURRENT ROW)::NUMERIC, 2) AS avg_sales
FROM orders_temp
GROUP BY order_date)
WHERE order_date >= ('2024-04-01')::DATE+INTERVAL '6 DAYS';
```

	order_date date	total_sales numeric	avg_sales numeric
1	2024-04-07	263201.03	37600.15
2	2024-04-08	273104.54	39014.93
3	2024-04-09	271918.00	38845.43
4	2024-04-10	264966.81	37852.40
5	2024-04-11	275977.28	39425.33
6	2024-04-12	275709.20	39387.03
7	2024-04-13	280975.77	40139.40
8	2024-04-14	274368.05	39195.44

4. Создадим функцию для поиска N-го активного пользователя (по количеству покупок)

```
CREATE OR REPLACE FUNCTION NthMostActiveCustomercnt (N INT)
    RETURNS TABLE (customer_id INT, order_cnt INT) AS $$
BEGIN
    RETURN QUERY(
        WITH number_of_orders AS(
        SELECT orders_temp.customer_id,
            COUNT(orders_temp.order_id) AS order_cnt
        FROM orders_temp
        GROUP BY orders_temp.customer_id
        ORDER BY order_cnt),
        ranked AS (
        SELECT number_of_orders.customer_id,
            number_of_orders.order_cnt,
            DENSE_RANK() OVER (ORDER BY number_of_orders.order_cnt DESC) AS rank
ing
        FROM number_of_orders)
        SELECT ranked.customer_id,
            ranked.order_cnt::INTEGER
        FROM ranked
        WHERE ranked.ranking = N);

    END;
    $$ LANGUAGE plpgsql
```

Пользователь с наибольшим количеством покупок:

```
SELECT * FROM NthMostActiveCustomercnt(1);
```

	customer_id integer	order_cnt integer
1	136485	9

5. Создадим функцию для поиска N-го активного пользователя (по сумме покупок)

```
CREATE OR REPLACE FUNCTION NthMostActiveCustomerSum (N INT)
    RETURNS TABLE (customer_id INT, order_sum INT) AS $$
BEGIN
    RETURN QUERY(
        WITH number_of_orders AS(
        SELECT orders_temp.customer_id,
            SUM(orders_temp.purchase_sum) AS order_sum
        FROM orders_temp
        GROUP BY orders_temp.customer_id
        ORDER BY order_sum),
        ranked AS (
        SELECT number_of_orders.customer_id,
            number_of_orders.order_sum,
            DENSE_RANK() OVER (ORDER BY number_of_orders.order_sum DESC) AS rank
ing
        FROM number_of_orders)
        SELECT ranked.customer_id,
            ranked.order_sum::INTEGER
        FROM ranked
        WHERE ranked.ranking = N);

    END;
    $$ LANGUAGE plpgsql
```

Пользователь с самой большой суммой покупок:

	customer_id integer	order_sum integer
1	225849	4650

6. Создади функцию, которая поможет нам идентифицировать самый покупаемый продукт:

```
CREATE OR REPLACE FUNCTION NthMostPurchasedProduct (N INT)
    RETURNS TABLE (product_id INT, product_name VARCHAR(512)) AS $$

BEGIN
    RETURN QUERY(
        WITH total_quantity AS(
        SELECT o.product_id,
            p.product_name,
            SUM(o.product_quantity) AS prd_qty
        FROM orders_temp AS o
        LEFT JOIN (SELECT DISTINCT(products.product_name), products.product_id F
ROM products) AS p
```

```
            USING (product_id)
        GROUP BY o.product_id, p.product_name
        ORDER BY prd_qty DESC),
        ranked AS (
        SELECT total_quantity.product_id,
            total_quantity.product_name,
            DENSE_RANK() OVER (ORDER BY total_quantity.prd_qty DESC) AS ranking
        FROM total_quantity)
        SELECT ranked.product_id,
            ranked.product_name
        FROM ranked
        WHERE ranked.ranking = N);
    END;
    $$ LANGUAGE plpgsql
```

Самым продаваемым товаром оказалась упаковка батареек:

	product_id integer	product_name character varying	prd_qty integer
1	692	AAA Batteries (4-pack)	29665

7. Создадим функцию для упрощенного поиска продаж по конкретному продукту в конкретный месяц и год:

```
CREATE OR REPLACE FUNCTION ProductSales (product VARCHAR(30),
mnth INT,
    yr INT)
    RETURNS TABLE (product_name VARCHAR(512),
    product_id INT,
    sales FLOAT)
LANGUAGE plpgsql
AS $$

DECLARE
    rslt RECORD;

BEGIN
    FOR rslt IN(SELECT o.product_id,
        p.product_name,
        SUM(o.purchase_sum) AS prd_sum
        FROM orders_temp AS o
        LEFT JOIN (SELECT DISTINCT(products.product_name), products.product_id F
ROM products) AS p
            USING (product_id)
        WHERE LOWER(p.product_name) LIKE LOWER(product)
            AND DATE_PART('MONTH', o.order_date) = mnth AND DATE_PART('YEAR', o.
order_date)= yr
        GROUP BY o.product_id, p.product_name
        ORDER BY prd_sum DESC)
        LOOP product_name := rslt.product_name;
            product_id := rslt.product_id;
```

```
            sales := ROUND(rslt.prd_sum::NUMERIC, 2);


        RETURN NEXT;
        END LOOP;
END;
$$
```

Вот, например, продажи мониторов в январе 2024:

```
SELECT * FROM ProductSales('%onitor', 01, 2024);
```

	product_name character varying	product_id integer	sales double precision
1	34in Ultrawide Monitor	493	179080
2	20in Monitor	168	34426.87
3	27in FHD Monitor	443	21000

Заключения

В рамках данного мини-проекта мы анализировали данные о продажах вымышленной организации. Используя функционал временных таблиц, мы рассчитали общую сумму покупки. Мы также произвели расчет скользящего среднего суммы продаж за последние 2 месяца датасета. Функционал table-valued function позволил нам создать ряд функций для определения самого активного пользователя, самого продаваемого товара и продаж определенного товара в заданный месяц, год.

В дальнейшем было бы интересно взглянуть на возможные покупательские тенденции пользователей в разрезе возрастной категории.

Ссылки

Ссылка на коды схем всех 3 таблиц

Контакты

lianazaripovar@gmail.com

tg: @zaripova_liana