1. Counting Unique Visitors

sql

```
unique_visitors = f"""
    SELECT
        COUNT(DISTINCT(customer_number)) AS unique_visitors
    FROM
        user_sessions
    WHERE
        timestamp BETWEEN '{start_date}' AND '{end_date}';
"""
```

- **Description**: This query counts the number of distinct customers who visited the platform within a specified date range.
- **Execution**: The count of distinct customer_number is retrieved from the user_sessions table between start_date and end_date.
- **Result Handling**: The result is converted to a Python integer using Pandas to store in self.unique_visitors_value.

---

2. Identifying Repeated Visitors (Historical Range & Current Period)

sql

```
Repeated_visitors_05_2024 = f"""
    SELECT
        second_range.customer_number
    FROM (
        SELECT DISTINCT customer_number
        FROM user_sessions
        WHERE timestamp BETWEEN '2024-06-01' AND '{start_date}'
    ) AS first_range
    INNER JOIN (
        SELECT DISTINCT customer_number
        FROM user_sessions
        WHERE timestamp BETWEEN '{start_date}' AND '{end_date}'
    ) AS second_range
    ON first_range.customer_number = second_range.customer_number;
"""
```

- **Description**: This query finds customers who have visited during two time periods: from '2024-06-01' to start_date (first range) and start_date to end_date (second range).
- **Logic**: It uses two subqueries to select distinct customer_number from user_sessions for each period and joins them on matching customer numbers.

3. Identifying Repeated Visitors (Current Period)

sql

```
Repeated_visitors = f"""
  SELECT
    customer_number
  FROM (
    SELECT
      customer_number,
      COUNT(*) AS visit_count
    FROM
      user_sessions
    WHERE
      timestamp BETWEEN '{start_date}' AND '{end_date}'
    GROUP BY
      customer_number
  ) AS customer_visitors
  WHERE
    customer_visitors.visit_count > 1;
"""
```

- **Description**: This query identifies visitors who have visited more than once within the specified start_date to end_date period.
- **Logic**: It groups customer visits by customer_number and filters those with visit_count > 1.

---

4. Combining Results for Visitors

- **Union Operation**: The sets set_visitors_05_2024 and set_visitors_repeated are created from the query results, and their union is computed.
- **Output**: The combined count represents unique repeated visitors across different time frames.

---

5. Counting Unique Customers (Orders)

```
sql

Unique_cutomers = f"""

    SELECT

        COUNT(DISTINCT(customer_mobile_num)) AS UNIQUE_CUSTOMERS

    FROM

        fact_order

    INNER JOIN

        invoice_detail ON fact_order.petpooja_order_id = invoice_detail.petpooja_order_id

    WHERE

        fact_order.timestamp BETWEEN '{start_date}' AND '{end_date}';

"""
```

- **Description**: This query counts distinct customers who placed orders within the specified date range.
- **Logic**: fact_order and invoice_detail are joined based on petpooja_order_id, and distinct customer_mobile_num counts are computed.
- **Result Handling**: The result is stored as a Python integer.

---

6. Identifying Repeated Customers (Historical Range & Current Period)

```
sql

Repeated_customer_05_2024 = f"""

  SELECT

    DISTINCT second_range.customer_mobile_num AS INTERSECTING_CUSTOMERS

  FROM (

    SELECT DISTINCT customer_mobile_num

    FROM fact_order

    INNER JOIN invoice_detail

      ON fact_order.petpooja_order_id = invoice_detail.petpooja_order_id

    WHERE fact_order.timestamp BETWEEN '2024-05-01' AND '{start_date}'

  ) AS first_range

  INNER JOIN (

    SELECT DISTINCT customer_mobile_num

    FROM fact_order

    INNER JOIN invoice_detail

      ON fact_order.petpooja_order_id = invoice_detail.petpooja_order_id

    WHERE fact_order.timestamp BETWEEN '{start_date}' AND '{end_date}'

  ) AS second_range

  ON first_range.customer_mobile_num = second_range.customer_mobile_num;

"""
```

- **Description**: Finds customers who placed orders in both the historical range ('2024-05-01' to start_date) and the current period (start_date to end_date).

7. Identifying Repeated Customers (Current Period)

```
sql

Repeated_customers = f"""

   SELECT

      customer_mobile_num

   FROM (

      SELECT

         customer_mobile_num,

         COUNT(*) AS order_count

      FROM

         fact_order

      INNER JOIN

         invoice_detail ON fact_order.petpooja_order_id = invoice_detail.petpooja_order_id

      WHERE

         fact_order.timestamp BETWEEN '{start_date}' AND '{end_date}'

      GROUP BY

         customer_mobile_num

   ) AS customer_orders

   WHERE

      order_count > 1;

"""
```

- **Description**: Identifies customers with more than one order during the specified period.

---

8. Combining Results for Customers

Python Code:

```
# Find the union of both result sets
union_customers = set_customers_05_2024.union(set_customers_repeated)
self.Repeated_cutomers_value = len(union_customers)
```

Description:

- **Union Operation**: The sets set_customers_05_2024 and set_customers_repeated are created, and their union represents the total unique repeated customers.