Sure! Here’s the updated documentation incorporating the details about the clustering method used:

---

# Documentation for User Order Statistics Analysis

## Overview

This analysis retrieves user statistics from a PostgreSQL database, specifically focusing on the number of orders and total order amounts for each user. It categorizes users based on their order behavior and employs KMeans clustering to segment users into distinct groups based on their order statistics.

## SQL Query Explanation

### Query Breakdown

The SQL query consists of several components:

1. \*\*Common Table Expressions (CTEs)\*\*:

- \*\*UserOrderStats\*\*: This CTE calculates the total number of orders (`order\_count`) and the total amount spent (`total\_order\_amount`) for each user by joining the `orders` and `groups\_carts` tables. It groups the results by `user\_id`.

- \*\*AverageStats\*\*: This CTE computes the average values for `order\_count` and `total\_order\_amount` across all users.

2. \*\*Final Selection\*\*:

- The main SELECT statement retrieves user statistics from `UserOrderStats` and compares each user's order statistics against the averages calculated in `AverageStats`. Users are categorized into three groups based on their order behavior:

- \*\*Valuable User\*\*: Users with both `order\_count` and `total\_order\_amount` above average.

- \*\*Normal User\*\*: Users with either `order\_count` or `total\_order\_amount` above average.

- \*\*Rarely User\*\*: Users who fall below average in both metrics.

### SQL Query Code

```sql

WITH UserOrderStats AS (

SELECT

gc.user\_id,

COUNT(o.id) AS order\_count,

SUM(o.total\_amount) AS total\_order\_amount

FROM

public.orders o

JOIN

public.groups\_carts gc ON o.groups\_carts\_id = gc.id

GROUP BY

gc.user\_id

),

AverageStats AS (

SELECT

AVG(order\_count) AS avg\_order\_count,

AVG(total\_order\_amount) AS avg\_total\_order\_amount

FROM

UserOrderStats

)

SELECT

uos.user\_id,

uos.order\_count,

uos.total\_order\_amount,

CASE

WHEN uos.order\_count > avg.avg\_order\_count AND uos.total\_order\_amount > avg.avg\_total\_order\_amount THEN 'valuable user'

WHEN uos.order\_count > avg.avg\_order\_count OR uos.total\_order\_amount > avg.avg\_total\_order\_amount THEN 'normal user'

ELSE 'rarely user'

END AS user\_category

FROM

UserOrderStats uos,

AverageStats avg

ORDER BY

uos.total\_order\_amount DESC;

```

## Clustering Analysis

### Clustering Method Used

In this analysis, we employed the \*\*KMeans clustering\*\* algorithm to segment users based on their order statistics. KMeans is a partitioning method that groups data into a predefined number of clusters. The algorithm aims to minimize the variance within each cluster.

#### Steps in Clustering:

1. \*\*Data Preparation\*\*: The relevant features (`order\_count` and `total\_order\_amount`) are selected for clustering.

2. \*\*Elbow Method\*\*: The optimal number of clusters is determined using the Elbow method, which involves plotting the inertia (sum of squared distances from each point to its assigned cluster center) against the number of clusters. The "elbow" point in the plot indicates an appropriate number of clusters.

3. \*\*KMeans Implementation\*\*: After determining the optimal number of clusters, the KMeans algorithm is applied to the data. Each user is assigned to a cluster based on their order statistics.

4. \*\*Visualization\*\*: The clusters are visualized using scatter plots and pair plots to understand the distribution and characteristics of each cluster.

### Code for Clustering Analysis

```python

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

import seaborn as sns

# Prepare the data for clustering

X = user\_data[['order\_count', 'total\_order\_amount']]

# Determine the optimal number of clusters using the Elbow method

inertia = []

for n in range(1, 11):

kmeans = KMeans(n\_clusters=n, random\_state=42)

kmeans.fit(X)

inertia.append(kmeans.inertia\_)

# Plot the Elbow curve

plt.figure(figsize=(10, 6))

plt.plot(range(1, 11), inertia, marker='o')

plt.title('Elbow Method for Optimal k')

plt.xlabel('Number of Clusters')

plt.ylabel('Inertia')

plt.grid()

plt.show()

# Choose the number of clusters (k) based on the elbow curve

k = 3 # Example: using 3 clusters

kmeans = KMeans(n\_clusters=k, random\_state=42)

user\_data['cluster'] = kmeans.fit\_predict(X)

# Display the clustered data

print(user\_data[['user\_id', 'order\_count', 'total\_order\_amount', 'cluster']])

```

## Conclusion

This analysis provides insights into user behavior based on their order statistics. By categorizing users and applying KMeans clustering, businesses can better understand their customer base, tailor marketing strategies, and improve customer retention efforts. The visualizations further enhance the understanding of the clusters, making it easier to identify trends and patterns in user behavior.

---

This documentation outlines the purpose, logic, and methodology of the code, while also explaining the clustering method used. If you need further modifications or details, feel free to ask!