**Project 3 : Fish Food Analysis**

**Source Code :**

```
#importing necessary libraries
import pandas as pd
#Uploading dataset files
from google.colab import files
uploaded=files.upload()

#Printing the whole dataset
df=pd.read_csv('Fish Food Analysis-1 Updated.csv')
df

#Printing the head or first 5 rows of the dataset
df.head()

#Printing the tail or last 5 rows of the dataset
df.tail()

#Printing all the information of the dataset
df.info

#Printing the name of columns of the dataset
df.columns

#Printing the shape of the dataset
df.shape

#Printing the size of the dataset
df.size

#Printing the data types of each column of the dataset
df.dtypes

#Printing the name of columns contain numerical values
numerical_cols=df.columns[df.dtypes!=object]
numerical_cols

#Printing the name of columns contain categorical or non-numerical values
categorical_cols=df.columns[df.dtypes==object]
categorical_cols
```

```
#Creating a new dataset by dropping the categorical column(s) of previous dataset and printing the head of the new dataset
df0 = df.drop(["Ingredients"],axis=1)
df0.head()

#Creating and printing another new dataset by transposing rows and columns of previous dataset as we can create relation from column to column and hence printing the head to see the new dataset
df1 = df0.transpose()
print(df1)

#Printing the head of the newly created dataset
df1.head()

#Printing the tail of the newly created dataset
df1.tail()

#Printing all the information of the new dataset
df1.info()

#Printing the column or column range of the new dataset
df1.columns

#Printing the shape of the new dataset
df1.shape

#Printing the size of the new dataset
df1.size

#Printing the data types of each column of the new dataset
df1.dtypes

#Printing the name of columns of new dataset contain numerical values
numerical_cols2=df1.columns[df1.dtypes!=object]
numerical_cols2

#Printing the name of columns contain categorical or non-numerical values
categorical_cols2=df1.columns[df1.dtypes==object]
categorical_cols2
```

```
#Calculating the Minkowaski distance from the last column to each column to find
out the relation of them. The lowest distance meaning most relationship
from scipy.spatial import distance
mink_dist1 = distance.minkowski(df1[8].astype(float),df1[0].astype(float),p=100)
print(mink_dist1)
mink_dist2 = distance.minkowski(df1[8].astype(float),df1[1].astype(float),p=100)
print(mink_dist2)
mink_dist3 = distance.minkowski(df1[8].astype(float),df1[2].astype(float),p=100)
print(mink_dist3)
mink_dist4 = distance.minkowski(df1[8].astype(float),df1[3].astype(float),p=100)
print(mink_dist4)
mink_dist5 = distance.minkowski(df1[8].astype(float),df1[4].astype(float),p=100)
print(mink_dist5)
mink_dist6 = distance.minkowski(df1[8].astype(float),df1[5].astype(float),p=100)
print(mink_dist6)
mink_dist7 = distance.minkowski(df1[8].astype(float),df1[6].astype(float),p=100)
print(mink_dist7)
mink_dist8 = distance.minkowski(df1[8].astype(float),df1[7].astype(float),p=100)
print(mink_dist8)
```

**Project Explanation :**

Let's break down the code and understand each part :

1. **Importing necessary libraries :**

```
import pandas as pd
from google.colab import files
```

- This code snippet imports the Pandas library for data manipulation and the **files** module from Google Colab to upload files.

2. **Uploading dataset files :**

```
uploaded = files.upload()
```

- This code allows the user to upload dataset files.

3. **Printing the whole dataset :**

```
df = pd.read_csv('Fish Food Analysis-1 Updated.csv')
df
```

- Reads the CSV file into a Pandas DataFrame named **df** and prints the entire dataset.

4. **Printing the head and tail of the dataset :**

```
df.head()
df.tail()
```

- Displays the first 5 rows (**head()**) and last 5 rows (**tail()**) of the dataset.

5. Printing information about the dataset :

```
df.info()
```

- Displays information about the dataset (e.g., data types, non-null counts).

6. Printing column-related information :

```
df.columns
df.shape
df.size
df.dtypes
```

- Outputs the column names, shape, size, and data types of the dataset.

7. Identifying numerical and categorical columns :

```
numerical_cols = df.columns[df.dtypes != object]
categorical_cols = df.columns[df.dtypes == object]
```

- Separates columns into numerical and categorical based on data types.

8. Creating a new dataset by dropping a categorical column :

```
df0 = df.drop(["Ingredients"], axis=1)
df0.head()
```

9. Creating and transposing another new dataset :

```
df1 = df0.transpose()
df1.head()
```

- Creates a new dataset (**df1**) by transposing rows and columns of the previous dataset.

10. Printing information about the new dataset (df1) :

```
df1.info()
```

- Displays information about the new dataset.

11. Identifying numerical and categorical columns in the new dataset :

```
numerical_cols2 = df1.columns[df1.dtypes != object]
categorical_cols2 = df1.columns[df1.dtypes == object]
```

- Separates columns into numerical and categorical in the new dataset.

12. Calculating Minkowski distance :

```
from scipy.spatial import distance
mink_dist1 = distance.minkowski(df1[8].astype(float), df1[0].astype(float), p=100)
# ... similar lines for other columns
```

- Computes the Minkowski distance between the last column (index 8) and each other column in the transposed dataset (**df1**). The distance is calculated using the **scipy.spatial.distance.minkowski** function.
- The distances are printed to evaluate the relationship between the columns. A lower distance suggests a stronger relationship.

**Note :**

- The code seems to be missing parentheses in **df.info**, it should be **df.info()**.

- The distance calculation uses a high value for the **p** parameter (100), which effectively makes it behave like the maximum distance. Usually, **p** is set to 1 for Manhattan distance and 2 for Euclidean distance. Using 100 might not provide meaningful results.