Q1. K-Nearest Neighbors (KNN) algorithm to predict the type of fruit for new samples based on these attributes.

Weight (g)	Diameter (cm)	Sweetness (1-10)	Ripeness (1-10)	Class (Fruit Type)
150	7	7	8	Apple
150	7	7	8	Apple
180	8	6	7	Orange
180	8	6	7	Orange
120	6	8	9	Banana

```
# 1. Import required libraries
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split, LeaveOneOut, cross_val_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# 2. Create the dataset

data = {
    'Weight_g': [150, 150, 180, 180, 120],
    'Diameter_cm': [7, 7, 8, 8, 6],
    'Sweetness_1_10': [7, 7, 6, 6, 8],
    'Ripeness_1_10': [8, 8, 7, 7, 9],
    'Class': ['Apple', 'Apple', 'Orange', 'Orange', 'Banana']
}
```

```
df = pd.DataFrame(data)

# Show dataset
print("Dataset:\n", df)

# 3. Data Exploration

print("\nDataset shape:", df.shape)
print("\nFirst rows:\n", df.head())
print("\nSummary statistics:\n", df.describe(include='all'))
print("\nMissing values:\n", df.isnull().sum())
print("\nDuplicate rows:", df.duplicated().sum())


# 4. Preprocessing (Features and Labels)

X = df[['Weight_g', 'Diameter_cm', 'Sweetness_1_10', 'Ripeness_1_10']].values
y_raw = df['Class'].values

# Encode string labels to numbers
le = LabelEncoder()
y = le.fit_transform(y_raw)
print("\nLabel Mapping:", dict(zip(le.classes_, le.transform(le.classes_))))


# 5. Feature Scaling (Important for KNN)

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 6. Train-Test Split

X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.4, random_state=42
)

print("\nTrain size:", len(X_train))
print("Test size:", len(X_test))

# 7. Train the KNN Model
```

```
k = 3
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(X_train, y_train)
print("\nKNN Model trained with k =", k)

# 8. Evaluate the Model
y_pred = knn.predict(X_test)

print("\nAccuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred,
target_names=le.classes_))

# 9. Leave-One-Out Cross Validation (Better for tiny dataset)

loo = LeaveOneOut()
cv_scores = cross_val_score(knn, X_scaled, y, cv=loo, scoring='accuracy')

print("\nLeave-One-Out CV Scores:", cv_scores)
print("Mean LOO Accuracy:", np.mean(cv_scores))


# 10. Predict a New Fruit Sample
new_sample = np.array([[170, 7.5, 6, 8]])  # Example new fruit

# Scale using same scaler
new_sample_scaled = scaler.transform(new_sample)

pred_encoded = knn.predict(new_sample_scaled)[0]
pred_label = le.inverse_transform([pred_encoded])[0]

print("\nNew sample:", new_sample)
print("Predicted Fruit Type:", pred_label)
```

Q2. K-Nearest Neighbors (KNN) algorithm to predict the type of fruit for new samples based on these attributes. k=3

Duration	IMDb	Action	Romance	Comedy	Genre
130	8.2	9	3	2	Action
95	6.5	3	8	6	Romance
110	7.0	4	4	9	Comedy
140	7.8	8	2	3	Action
120	7.5	2	9	5	Romance
105	6.8	5	3	8	Comedy
150	8.4	9	2	1	Action
100	6.2	2	7	7	Romance
115	7.3	3	5	8	Comedy
125	7.1	6	5	3	Action