# Análisis de datos

## Nivel Básico – Explorador

### Misión 1

## 1. Recopilar Datos

*a.* ***Leer Datos desde un Archivo CSV***

Puedes usar el módulo csv para leer datos desde un archivo CSV.

```
import csv

def read_csv(file_path):
    data = []
    with open(file_path, mode='r') as file:
        reader = csv.reader(file)
        for row in reader:
            data.append(row)
    return data
```

```
# Ejemplo de uso

data = read_csv('archivo.csv')

print(data)
```

### *b.* ***Leer Datos desde un Archivo JSON***

Puedes usar el módulo json para leer datos desde un archivo JSON.

```
import json

def read_json(file_path):
    with open(file_path, mode='r') as file:
        data = json.load(file)
    return data
```

```
# Ejemplo de uso
data = read_json('archivo.json')

print(data)
```

## 2. Analizar Datos

### *a.* ***Calcular Estadísticas Básicas***

Puedes escribir funciones para calcular estadísticas básicas como media, mediana, moda, etc.

```
def mean(data):
    return sum(data) / len(data)

def median(data):
    sorted_data = sorted(data)
    n = len(sorted_data)
    mid = n // 2
    if n % 2 == 0:
        return (sorted_data[mid - 1] + sorted_data[mid]) / 2
    else:
```

```
        return sorted_data[mid]

def mode(data):
    from collections import Counter
    count = Counter(data)
    max_count = max(count.values())
    return [key for key, value in count.items() if value == max_count]
```

```
# Ejemplo de uso

numerical_data = [1, 2, 2, 3, 4, 5, 5, 5]
print(f'Mean: {mean(numerical_data)}')
print(f'Median: {median(numerical_data)}')
print(f'Mode: {mode(numerical_data)}')
```

*b.* ***Filtrar Datos***

Puedes escribir funciones para filtrar datos según ciertos criterios.

```
def filter_data(data, column_index, condition):
    return [row for row in data if condition(row[column_index])]
```

```
# Ejemplo de uso
data = [
    ['Name', 'Age'],
    ['Alice', 30],
    ['Bob', 25],
    ['Charlie', 35]
]
```

```
filtered_data = filter_data(data[1:], 1, lambda x: x > 30)
print(filtered_data)
```

### c. *Agrupar Datos*

Puedes agrupar datos y calcular estadísticas para cada grupo.

```
def group_by(data, key_index):
    grouped = {}
    for row in data:
        key = row[key_index]
        if key not in grouped:
            grouped[key] = []
        grouped[key].append(row)
    return grouped

def group_and_aggregate(data, key_index, value_index, aggregator):
    grouped = group_by(data, key_index)
    result = {}
    for key, rows in grouped.items():
        values = [row[value_index] for row in rows]
        result[key] = aggregator(values)
    return result
```

```
# Ejemplo de uso
data = [
    ['Category', 'Value'],
    ['A', 10],
    ['A', 20],
    ['B', 30],
    ['B', 40]
```

```
]

grouped_data = group_and_aggregate(data[1:], 0, 1, mean)
print(grouped_data)
```

## 3. **Visualizar Datos**

Para visualización sin usar bibliotecas externas, puedes generar gráficos simples utilizando matplotlib si decides usarla en el futuro.

```
import matplotlib.pyplot as plt

def plot_data(x, y):
    plt.plot(x, y, marker='o')
    plt.xlabel('X-axis')
    plt.ylabel('Y-axis')
    plt.title('Simple Plot')
    plt.show()
```

```
# Ejemplo de uso
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]

plot_data(x, y)
```

Estos pasos cubren la recopilación y análisis de datos sin necesidad de usar Pandas ni NumPy. Puedes adaptar estas funciones según tus necesidades específicas.