

# Informe del Proyecto-Practica número 5

Facultad de Ciencias – Escuela Profesional de Ciencia de la Computación

3/12/2024

— Universidad Nacional de Ingeniería —

**\*Nombre del Docente: Eduardo Yauri Lozano**

**\*Curso: Fundamentos de la Programación**

**\*Integrantes:**

- Sebastian Klauer Lara (20232633G)
- Marco Antonio Cruz Ipanaque (20231507H)

— Parte "a" del proyecto formativo —

## **-Introducción**

Comenzaremos el informe de este proyecto presentando el problema seleccionado y describiendo una serie de consideraciones que tomamos para que la solución del código sea más sencilla de ejemplificar.

### **1) Problema Seleccionado**

Sistema de Registro de Calificaciones: Este sistema permite registrar calificaciones para estudiantes en diferentes materias. Proporciona herramientas para calcular promedios por estudiante y un promedio general del grupo. Se deben implementar las funcionalidades para registrar estudiantes, mostrar promedio de estudiante, mostrar promedio general, mostrar promedio de estudiante por curso llevado, mostrar información de un estudiante, y 3 funcionalidades más a criterio del estudiante. Deben formular las siguientes clases: Estudiante, Gestor de calificaciones, entre otros que considere el estudiante pertinente. Incluir funcionalidad para guardar en archivos los reportes generados.

### **2) Consideraciones**

Al inicio del programa, se solicitará al usuario que ingrese el número de estudiantes pertenecientes a un salón. Se considerarán un total de 3 cursos y 3 notas por curso, con el objetivo de evitar perder tiempo mostrando cómo funciona el código.

## -Desarrollo

### 1) Menú de opciones

Iniciamos creando un menú con el fin de que, una vez que el usuario haya proporcionado toda la información solicitada sobre los estudiantes, tenga la opción de escoger cuál de las cinco opciones desea visualizar o presionar el número 6 para finalizar la ejecución del programa.

```
def interfaz():  
    print("    --- Escoger una opcion ---")  
    print(" 1. Prom. general del Estudiante")  
    print(" 2. Prom. general del Estudiante/Materia")  
    print(" 3. Prom. general del Salon/Materia")  
    print(" 4. Prom. general del salon")  
    print(" 5. Generar Reporte")  
    print(" 6. Salir ")
```

### 2) Clases creadas

#### — Calificaciones

La clase Calificaciones contiene los atributos que gestionan las calificaciones de los estudiantes.

```
class Calificaciones:  
  
>     def __init__(self): #constructor de la clase  
  
>     def Notas_cursos(self): ...  
  
>     def Notas_de_un_curso(self, k, f): ...
```

```

>     def prom_general(self): ...
>
>     def Dev_promedio_general(self): ...
>
>     def prom_general_materia(self): ...
>
>     def Acceder_prom_materia(self, k): ...

```

En los atributos tenemos:

- **notas:** Una lista que almacena las notas de cada estudiante para cada curso.
- **prom-materia:** Una lista que guarda el promedio por materia de cada estudiante.
- **promedio-general:** Un atributo que calcula y almacena el promedio general de todas las materias de un estudiante.

```

def __init__(self): #constructor de la clase / inicializa to
    self._notas = [[0 for _ in range(3)] for _ in range(3)]
    self._prom_materia = [0] * 3 # Promedios por curso / [
    self._promedio_general = 0

```

Dentro de esta clase existen varios métodos:

- **Notas-cursos():** Solicita al usuario que ingrese las notas de cada estudiante para cada curso.
- **Notas-de-un-curso():** Imprime en pantalla las notas de los estudiantes, separadas por curso.
- **prom-general():** Calcula el promedio general de un estudiante, considerando todas las materias.
- **Dev-promedio-general():** Retorna el promedio general calculado de un estudiante.

- **prom-general-materia()**: Calcula el promedio de cada materia (curso) para cada estudiante y lo almacena en la lista prom-materia.
- **Acceder-prom-materia(k)**: Retorna el promedio de la materia ubicada en el índice k de la lista de promedios de materias.

Métodos de la clase Calificaciones:

```
def Notas_cursos(self):
    for i in range(3):
        print(f" -Curso {i + 1}: ")
        for j in range(3):
            self._notas[i][j] = float(input(f" |Nota {j + 1}: "))

def Notas_de_un_curso(self, k, f):
    for i in range(3):
        f.write(f"{self._notas[k][i]:<8}")

def prom_general(self):
    suma = 0
    for i in range(3):
        for j in range(3):
            suma += self._notas[i][j]
    self._promedio_general = suma / 9 # Promedio general del alumno

def Dev_promedio_general(self):
    return self._promedio_general

def prom_general_materia(self):
    for i in range(3):
        suma = 0
        for j in range(3):
            suma += self._notas[i][j]
        self._prom_materia[i] = suma / 3

def Acceder_prom_materia(self, k):
    return self._prom_materia[k]
```

## — Estudiante

Esta clase tiene representa a un alumno y permite definir su información académica junto con la clase Calificaciones.

```

class Estudiante:
>     def __init__(self, nombre="", ID=0): #constructor/
>
>     def Escribir_nombre(self): ...
>
>     def Devolver_nombre(self): ...
>
>     def Pedir_ID(self): ...
>
>     def Devolver_ID(self): ...

```

En los atributos tenemos:

- **nombre:** El nombre del alumno que se está registrando.
- **ID:** : Un número que sirve como identificador personal del alumno.
- **calificaciones:** Una instancia de la clase Calificaciones, lo que permite acceder a los métodos relacionados con las calificaciones.

```

def __init__(self, nombre="", ID=0): #constructor/
|
|     self._nombre = nombre
|     self._ID = ID
|     self.calificaciones = Calificaciones()

```

Dentro de esta clase existen varios métodos:

- **Escribir-nombre():** Permite al usuario ingresar el nombre del estudiante.
- **Devolver-nombre():** Devuelve el nombre del estudiante.
- **Pedir-ID():** Permite al usuario ingresar el ID del estudiante.
- **Devolver-ID():** Retorna el ID del estudiante.

## Métodos de la clase Calificaciones:

```
def Escribir_nombre(self):  
|     self._nombre = input()  
  
def Devolver_nombre(self):  
|     return self._nombre  
  
def Pedir_ID(self):  
|     self._ID = int(input("ID del Alumno: "))  
  
def Devolver_ID(self):  
|     return self._ID
```

### 3) Funciones para manejar promedios

Estas funciones permiten calcular y mostrar los promedios individuales de los estudiantes, los promedios por materia y el promedio general del salón:

- **Mostrar-promedio-Alumno(ptr, n):** Recibe como parámetro una lista de instancias de la clase Estudiante y el número de estudiantes. Imprime el promedio general de cada estudiante.
- **Mostrar-promedio-Alumno-materia(ptr, n):** Muestra el promedio por curso de cada estudiante, tomando en cuenta los promedios calculados previamente.
- **Crear-promedio-salon-materia(ptr, arr, n):** Recibe como parámetros una lista de instancias de Estudiante, una lista arr inicializada con ceros (que representa el promedio por curso), y el número de estudiantes. Calcula el promedio de salón para cada materia y lo almacena en arr.
- **Mostrar-promedio-salon-materia(arr, n):** Muestra el promedio de cada materia para todos los estudiantes, utilizando la lista arr donde se almacenan estos promedios.
- **Mostrar-promedio-salon-total(arr):** Calcula y retorna el promedio total del salón a partir de los promedios por materia almacenados en arr.
- **Generar-reporte(ptr, n):** Esta función recibe una lista de instancias de Estudiante y el número de estudiantes. Genera un archivo de texto

llamado "Reporte.txt" donde se registran los detalles de cada estudiante: su ID, nombre, notas por curso y promedio general. El archivo se crea o sobrescribe utilizando el modo 'w'.

```
def Mostrar_promedio_Alumno(ptr, n):  
    for i in range(n):  
        print(f"Prom.Alumno {i + 1}: {ptr[i].calificaciones.Dev_promedio_general()}")
```

```
def Mostrar_promedio_Alumno_materia(ptr, n):  
    for i in range(n):  
        print(f"Alumno {i + 1} -> |M1: {ptr[i].calificaciones.Acceder_prom_materia(0)}|
```

```
| M2: {ptr[i].calificaciones.Acceder_prom_materia(1)}| |M3: {ptr[i].calificaciones.Acceder_prom_materia(2)}|
```

```
def Crear_promedio_salon_materia(ptr, arr, n):  
    for i in range(3):  
        suma = 0  
        for j in range(n):  
            suma += ptr[j].calificaciones.Acceder_prom_materia(i)  
        arr[i] = suma / n  
  
def Mostrar_promedio_salon_materia( arr):  
    for i in range(3):  
        print(f"Prom.Alumnos.Curso {i + 1}: {arr[i]}")  
  
def Mostrar_promedio_salon_total(arr):  
    suma = 0  
    for i in range(3):  
        suma += arr[i]  
    return suma / 3
```

```
def Generar_reporte(ptr, n):
    with open("Reporte.txt", "w") as f:
        f.write(" -----Reporte Actualizado de la clase----\n")
        f.write(" ID      Nombre                Curso1          Curso2          Curso3          Promedio\n")

        for i in range(n):
            f.write(f"{ptr[i].Devolver_ID():<10}{ptr[i].Devolver_nombre():<20}")
            ptr[i].calificaciones.Notas_de_un_curso(0, f)
            f.write(" ")
            ptr[i].calificaciones.Notas_de_un_curso(1, f)
            f.write(" ")
            ptr[i].calificaciones.Notas_de_un_curso(2, f)
            f.write(f"{ptr[i].calificaciones.Dev_promedio_general():>10}\n")
```

### -Función principal "main()"

En la función principal, el usuario puede ingresar el número de estudiantes, sus nombres, identificadores y notas por curso. Posteriormente, se calculan y muestran los promedios generales de cada estudiante, así como los promedios por curso. También, el programa permite generar un reporte con toda esta información.

**El flujo de trabajo de esta funcion incluye:**

- Se le pide al usuario que introduzca el número de estudiantes matriculados

```
print( "----- Inicio del programa -----" )
alumnos = int(input(" Numero de alumnos matriculados: "))
print()
```

- Se crea una lista que contenga instancias de Estudiante y un numero total de elementos igual a alumnos (valor ingresado por el usuario al inicio) .

```
ptr = [Estudiante() for _ in range(alumnos)] # Lista de clases Estudiante
```



- Se le pide al usuario ingresar el nombre, el identificador y las notas de todos los alumnos matriculados en el salón.

```
# Pedimos el nombre del alumno y su ID
print("---- Registro de Datos y Notas de los estudiantes ----")
print()

for i in range(alumnos): # desde cero hasta alumnos -1 // bucle for
    print(f"Nombre del Alumno {i + 1}: ", end="")
    ptr[i].Escribir_nombre()
    ptr[i].Pedir_ID()
    print(f"Notas del Alumno {i + 1} : ")
    ptr[i].calificaciones.Notas_cursos()
    print()
```

- Se sigue con el uso de dos bucles for para calcular los promedios generales de los estudiantes y sus promedios por materia.

```
# Promedios generales
for i in range(alumnos): # desde cero hasta (alumnos-1) // bucle for
    ptr[i].calificaciones.prom_general()

# Promedios generales por materia
for k in range(alumnos): # desde cero hasta (alumnos -1) // bucle for
    ptr[k].calificaciones.prom_general_materia()
```

- Se inicializa una variable opción = 0 y una lista arr = [0, 0, 0] para almacenar los promedios del salón por materia.

```
opcion = 0 # Controla la opción que vamos a elegir
arr = [0] * 3 # Arreglo para almacenar promedio del salón/materia
```

- A continuación, se llama a la función `interfaz()` para mostrar el menú interactivo y permitir que el usuario elija qué acción desea realizar

```
interfaz()
opcion = int(input("opcion--> ")) #introduce el usuario la opcion que prefiera
```

- Se utiliza un bucle `while` para mantener el programa en ejecución hasta que el usuario elija salir (opción 6)

```
while opcion != 6:
    if opcion == 1:
        print("//Prom. general del Estudiante//")
        Mostrar_promedio_Alumno(ptr, alumnos)
        print()
        interfaz()
    elif opcion == 2:
        print("//Prom. general del Estudiante/Materia//")
        Mostrar_promedio_Alumno_materia(ptr, alumnos)
        interfaz()
```

```
    elif opcion == 3:
        print("//Prom. general del Salon/Materia//")
        Crear_promedio_salon_materia(ptr, arr, alumnos)
        Mostrar_promedio_salon_materia(ptr, arr, alumnos)
        print()
        interfaz()
    elif opcion == 4:
        print("//Prom. general del salon//")
        print(f"Prom.total.salon: {Mostrar_promedio_salon_total(arr)}")
        print()
        interfaz()
    elif opcion == 5:
        print("//Generar Reporte//")
        Generar_reporte(ptr, alumnos)
        print()
        interfaz()
    else:
        print("Opcion no permitida. Vuelva a intentar.")

    opcion = int(input("opcion--> "))

    if opcion == 6:
        print("//Salir//")
        print("Saliendo del programa...")
```

#### 4) Uso del bloque if-name- == "-main-"

Ayuda a controlar qué parte del código debe ejecutarse solo cuando el archivo es ejecutado directamente, evitando que se ejecute automáticamente si el archivo es importado desde otro programa

```
if __name__ == "__main__":  
    main()
```

—Salida en pantalla:

```
----- Inicio del programa -----  
Numero de alumnos matriculados: 3  
  
---- Registro de Datos y Notas de los estudiantes ----  
  
Nombre del Alumno 1: Carlos  
ID del Alumno: 234  
Notas del Alumno 1 :  
-Curso 1:  
|Nota 1: 12  
|Nota 2: 1  
|Nota 3: 15  
-Curso 2:  
|Nota 1: 13  
|Nota 2: 12  
|Nota 3: 16  
-Curso 3:  
|Nota 1: 18  
|Nota 2: 12  
|Nota 3: 13  
  
Nombre del Alumno 2: Kevin  
ID del Alumno: 265  
Notas del Alumno 2 :  
-Curso 1:  
|Nota 1: 13  
|Nota 2: 15  
|Nota 3: 16  
-Curso 2:  
|Nota 1: 12  
|Nota 2: 16  
|Nota 3: 11  
-Curso 3:  
|Nota 1: 18  
|Nota 2: 19  
|Nota 3: 15  
  
Nombre del Alumno 3: Sergio  
ID del Alumno: 225  
Notas del Alumno 3 :  
-Curso 1:  
|Nota 1: 16  
|Nota 2: 17  
|Nota 3: 11  
-Curso 2:  
|Nota 1: 18  
|Nota 2: 12  
|Nota 3: 13  
-Curso 3:  
|Nota 1: 10  
|Nota 2: 11  
|Nota 3: 16
```

## –Opción 1

```
--- Escoger una opcion ---
1. Prom. general del Estudiante
2. Prom. general del Estudiante/Materia
3. Prom. general del Salon/Materia
4. Prom. general del salon
5. Generar Reporte
6. Salir
opcion--> 1
//Prom. general del Estudiante//
Prom.Alumno 1: 12.444444444444445
Prom.Alumno 2: 15.0
Prom.Alumno 3: 13.777777777777779
```

## –Opción 2

```
--- Escoger una opcion ---
1. Prom. general del Estudiante
2. Prom. general del Estudiante/Materia
3. Prom. general del Salon/Materia
4. Prom. general del salon
5. Generar Reporte
6. Salir
opcion--> 2
//Prom. general del Estudiante/Materia//
Alumno 1 -> |M1: 9.333333333333334| |M2: 13.666666666666666| |M3: 14.333333333333334|
Alumno 2 -> |M1: 14.666666666666666| |M2: 13.0| |M3: 17.333333333333332|
Alumno 3 -> |M1: 14.666666666666666| |M2: 14.333333333333334| |M3: 12.333333333333334|
```

## –Opción 3

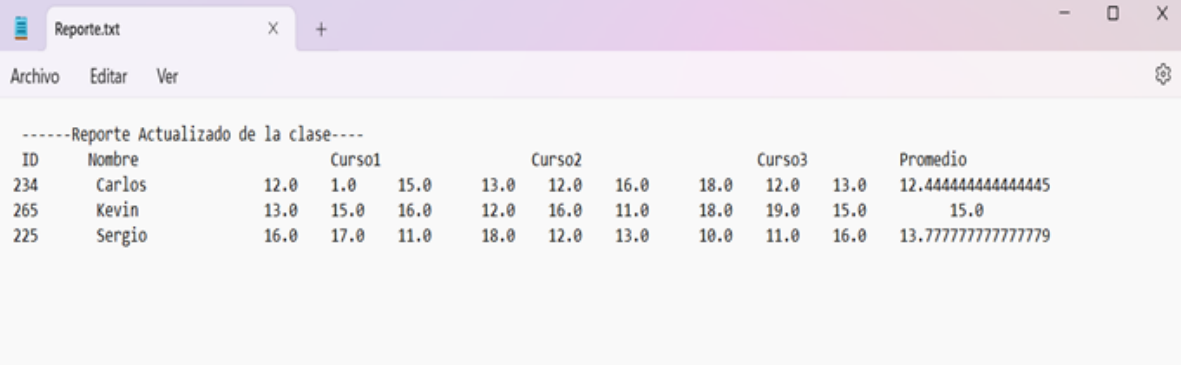
```
--- Escoger una opcion ---
1. Prom. general del Estudiante
2. Prom. general del Estudiante/Materia
3. Prom. general del Salon/Materia
4. Prom. general del salon
5. Generar Reporte
6. Salir
opcion--> 3
//Prom. general del Salon/Materia//
Prom.Alumnos.Curso 1: 12.888888888888888
Prom.Alumnos.Curso 2: 13.666666666666666
Prom.Alumnos.Curso 3: 14.666666666666666
```

## –Opción 4

```
--- Escoger una opcion ---
1. Prom. general del Estudiante
2. Prom. general del Estudiante/Materia
3. Prom. general del Salon/Materia
4. Prom. general del salon
5. Generar Reporte
6. Salir
opcion--> 4
//Prom. general del salon//
Prom.total.salon: 13.74074074074074
```

## –Opción 5

```
--- Escoger una opcion ---
1. Prom. general del Estudiante
2. Prom. general del Estudiante/Materia
3. Prom. general del Salon/Materia
4. Prom. general del salon
5. Generar Reporte
6. Salir
opcion--> 5
//Generar Reporte//
```



Reporte.txt

Archivo Editar Ver

-----Reporte Actualizado de la clase-----

ID	Nombre	Curso1			Curso2			Curso3			Promedio
234	Carlos	12.0	1.0	15.0	13.0	12.0	16.0	18.0	12.0	13.0	12.444444444444445
265	Kevin	13.0	15.0	16.0	12.0	16.0	11.0	18.0	19.0	15.0	15.0
225	Sergio	16.0	17.0	11.0	18.0	12.0	13.0	10.0	11.0	16.0	13.777777777777779

## –Opción 6

```
    --- Escoger una opcion ---
    1. Prom. general del Estudiante
    2. Prom. general del Estudiante/Materia
    3. Prom. general del Salon/Materia
    4. Prom. general del salon
    5. Generar Reporte
    6. Salir
opcion--> 6
//Salir//
Saliendo del programa...
```

## — Parte "b" del proyecto formativo —

### -Introducción

– Problema Seleccionado: juguetes-ventas.csv

### -Desarrollo

- Importamos librerías
- Leer el archivo CSV
- Carga el archivo CSV en un DataFrame para realizar análisis de datos.
- Calcular los ingresos de las ventas

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3
4 # Leer el archivo CSV
5 # Carga el archivo CSV en un DataFrame para realizar análisis de
  datos.
6 df = pd.read_csv('juguetes_ventas.csv')
7 print(df)
8
9 # Calcular los ingresos de las ventas
10 # Crea una nueva columna 'Ingresos', calculando la multiplicación
   de las columnas 'Ventas' y 'Precio'.
11 df['Ingresos'] = df['Ventas'] * df['Precio']
```



—Salida en pantalla:

```
      Fecha  Producto Region Ventas Precio
0  2024-01-01  Rompecabezas  Centro    378     53
1  2024-01-02      Muñeca  Oeste     73     20
2  2024-01-03      Muñeca  Oeste    132     21
3  2024-01-04  Rompecabezas  Centro    263     39
4  2024-01-05  Rompecabezas  Norte    486     80
..      ...      ...      ...      ...
495 2025-05-10      Muñeca  Oeste    103     59
496 2025-05-11  Rompecabezas  Centro    307     74
497 2025-05-12      Pelota  Oeste    142     98
498 2025-05-13  Rompecabezas   Sur    404     62
499 2025-05-14  Rompecabezas  Norte     74     63

[500 rows x 5 columns]
```

- Ventas e ingresos totales por región

```
13 # 1. Ventas e ingresos totales por región
14 # Agrupa los datos por 'Region' y calcula la suma total de las
    ventas e ingresos por cada región.
15 ventas_totales_por_region = df.groupby('Region')['Ventas'].sum()
16 ingresos_totales_por_region = df.groupby('Region')['Ingresos']
    .sum()
17 print("Ventas totales por región:\n", ventas_totales_por_region)
18 print("\nIngresos totales por región:\n",
    ingresos_totales_por_region)
```

—Salida en pantalla:

```
Ventas totales por región:
Region
Centro      35729
Norte       27070
Oeste       32790
Sur         31071
Name: Ventas, dtype: int64

Ingresos totales por región:
Region
Centro     2173314
Norte     1626261
Oeste     1967781
Sur       1960386
Name: Ingresos, dtype: int64
```

- Porcentaje de ingresos por región

```
20 # 2. Porcentaje de ingresos por región
21 # Calcula el porcentaje de ingresos de cada región con respecto
    al total de ingresos.
22 ingresos_total = ingresos_totales_por_region.sum()
23 porcentaje_por_cada_region = (ingresos_totales_por_region /
    ingresos_total) * 100
24 print("\nPorcentaje de ingresos por región:\n",
    porcentaje_por_cada_region)
25
```

—Salida en pantalla:

```
Porcentaje de ingresos por región:
Region
Centro      28.123532
Norte       21.044453
Oeste       25.463855
Sur         25.368161
Name: Ingresos, dtype: float64
```



- Ventas por mes

```
26 # 3. Ventas por mes
27 # Convierte la columna 'Fecha' a tipo datetime y extrae el mes
    para agrupar las ventas por mes.
28 df['Fecha'] = pd.to_datetime(df['Fecha'])
29 df['Mes'] = df['Fecha'].dt.to_period('M')
30 ventas_totales_por_mes = df.groupby('Mes')['Ventas'].sum()
31 print("\nVentas totales por cada mes:\n", ventas_totales_por_mes)
32
```

—Salida en pantalla:

```
Ventas totales por cada mes:
Mes
2024-01    7350
2024-02    7676
2024-03    7276
2024-04    7378
2024-05    8972
2024-06    7389
2024-07    7769
2024-08    7745
2024-09    7577
2024-10    9308
2024-11    6711
2024-12    7730
2025-01    8710
2025-02    6808
2025-03    7774
2025-04    7414
2025-05    3073
Freq: M, Name: Ventas, dtype: int64
```

- Producto más vendido y región más rentable

```

33 # 4. Producto más vendido y región más rentable
34 # Identifica el producto con mayor cantidad de ventas y la región
    que generó mayores ingresos.
35 producto_mas_vendido = df.groupby('Producto')['Ventas'].sum
    ().idxmax()
36 cantidad_vendida_producto = df.groupby('Producto')['Ventas'].sum
    ().max()
37 region_mas_rentable = df.groupby('Region')['Ingresos'].sum
    ().idxmax()
38 ingresos_region_mas_rentable = df.groupby('Region')['Ingresos']
    .sum().max()
39
40 # Muestra los resultados del producto más vendido y la región más
    rentable.
41 print(f"\nEl producto más vendido es '{producto_mas_vendido}' con
    un total de {cantidad_vendida_producto} unidades vendidas.")
42 print(f"La región más rentable es '{region_mas_rentable}' con
    ingresos totales de {ingresos_region_mas_rentable:.2f} soles
    .")

```

—Salida en pantalla:

```

El producto más vendido es 'Rompecabezas' con un total de 35270 unidades vendidas.
La región más rentable es 'Centro' con ingresos totales de 2173314.00 soles.

```

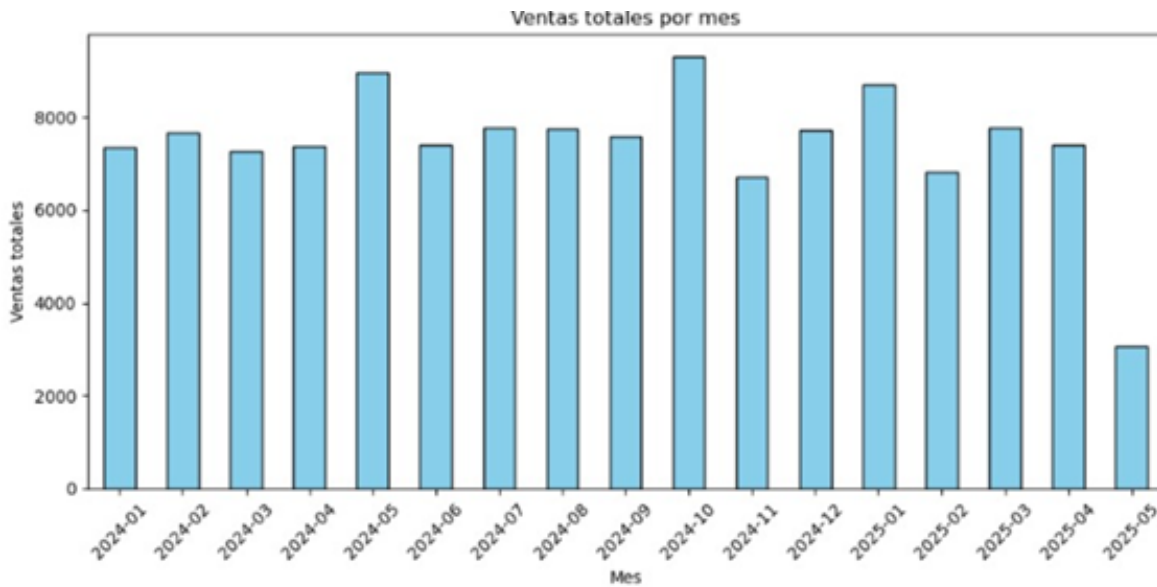
- Visualización de las ventas totales por mes

```

44 # 5. Visualización de las ventas totales por mes
45 # Genera un gráfico de barras para mostrar las ventas totales
    agrupadas por mes.
46 plt.figure(figsize=(10, 5))
47 ventas_totales_por_mes.plot(kind='bar', color='skyblue',
    edgecolor='black')
48 plt.title("Ventas totales por mes")
49 plt.xlabel("Mes")
50 plt.ylabel("Ventas totales")
51 plt.xticks(rotation=45)
52 plt.tight_layout()
53 plt.show()

```

—Salida en pantalla:



- Matriz de correlación

```
55 # 6. Matriz de correlación
56 # Calcula y muestra la matriz de correlación entre las variables
    numéricas.
57 correlacion = df.corr(numeric_only=True)
58 print("\nMatriz de correlación:\n", correlacion)
59
```

—Salida en pantalla:

```
Matriz de correlación:
          Ventas  Precio  Ingresos
Ventas    1.000000  0.023950  0.792598
Precio    0.023950  1.000000  0.551599
Ingresos  0.792598  0.551599  1.000000
```

- Gráficas adicionales

### a. Ingresos totales por región

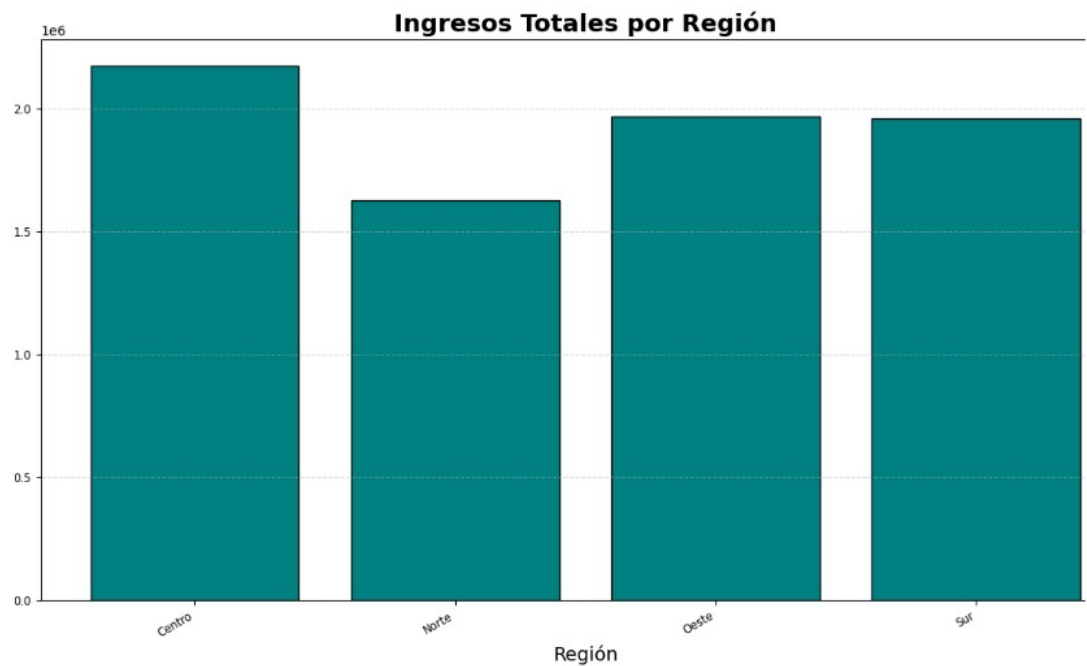
```
60 # 7. Gráficas adicionales
61
62 # a. Ingresos totales por región
63 # Crea un gráfico de barras para representar los ingresos totales
    por región.
64 plt.figure(figsize=(12, 7))
65 plt.bar(ingresos_totales_por_region.index,
        ingresos_totales_por_region.values, color='teal', edgecolor
        ='black')
66 plt.title('Ingresos Totales por Región', fontsize=18, fontweight
        ='bold')
67 plt.xlabel('Región', fontsize=14)
68 plt.ylabel('Ingresos (en soles)', fontsize=14)
69 plt.xticks(rotation=30, ha="right", fontsize='small')
70 plt.yticks(fontsize="small")
71 plt.grid(axis='y', linestyle='--', alpha=0.5)
72 plt.tight_layout()
73 plt.show()
```

### b. Porcentaje de ventas por producto

```
75 # b. Porcentaje de ventas por producto
76 # Genera un gráfico circular para visualizar el porcentaje de
    ventas de cada producto.
77 ventas_producto = df.groupby('Producto')['Ventas'].sum()
78 colores = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728'] # Colores
    personalizados
79 plt.figure(figsize=(8, 8))
80 plt.pie(ventas_producto, labels=ventas_producto.index, autopct
        ='%1.1f%%', startangle=140, colors=colores)
81 plt.title('Porcentaje de Ventas por Producto')
82 plt.show()
```

—Salida en pantalla:

\*Gráfica a:



\*Gráfica b:



- Gráfica de ventas acumuladas por producto

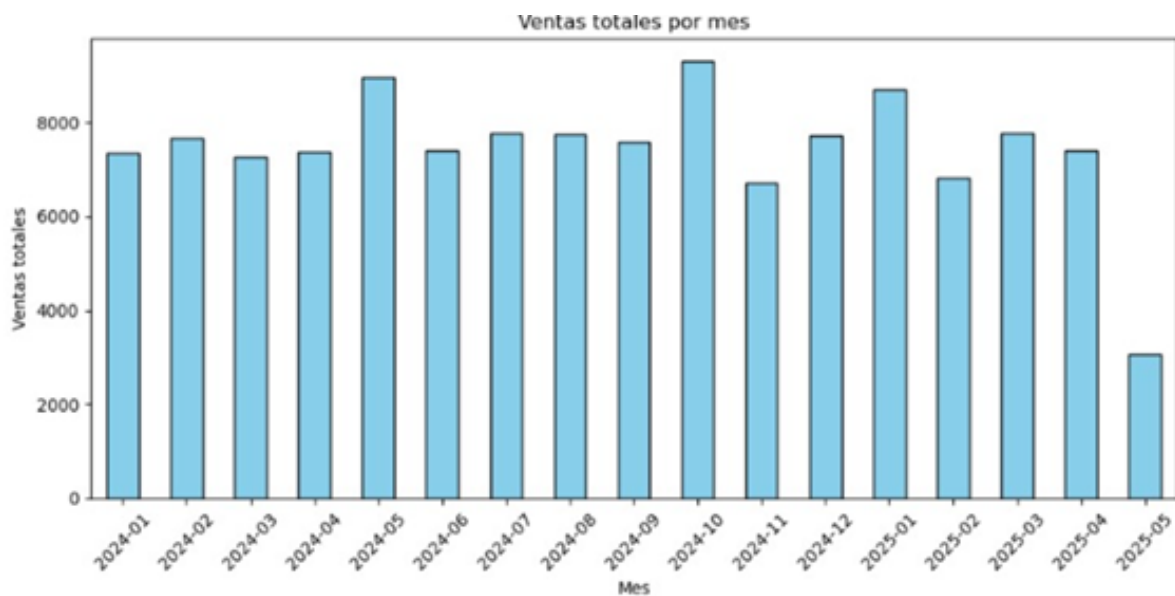


```

83 # 8. Gráfica de ventas acumuladas por producto
84 # Calcula las ventas acumuladas por producto y las ordena de
    mayor a menor.
85 ventas_acumuladas = df.groupby('Producto')['Ventas'].sum
    ().reset_index()
86 ventas_acumuladas = ventas_acumuladas.sort_values(by='Ventas',
    ascending=False)
87
88 # Muestra la tabla de ventas acumuladas
89 print("\nVentas acumuladas por producto:\n", ventas_acumuladas)
90 # Genera un gráfico de barras para visualizar las ventas
    acumuladas por producto.
91 plt.figure(figsize=(7, 7))
92 plt.bar(ventas_acumuladas['Producto'],
    ventas_acumuladas['Ventas'], color='skyblue')
93 plt.title('Ventas Acumuladas por Producto', fontsize=16)
94 plt.xlabel('Producto', fontsize=14)
95 plt.ylabel('Ventas Acumuladas', fontsize=14)
96 plt.xticks(rotation=45, ha='right', fontsize=12)
97 plt.grid(axis='y', linestyle='--', alpha=0.7)
98 plt.tight_layout()
99 plt.show()

```

—Salida en pantalla:



\*Tabla de ventas acumuladas por producto:

Ventas acumuladas por producto:		
	Producto	Ventas
3	Rompecabezas	35270
2	Pelota	33744
1	Muñeca	29424
0	Auto de Jugete	28222

\*Grafica:

