"Año del Fortalecimiento de la Soberanía Nacional"

UNIVERSIDAD DE PIURA

FACULTAD DE INGENIERÍA

ANÁLISIS DE DATOS CON PYTHON

GRUPO 08

"TRABAJO 1"

DOCENTE: Ing. Pedro Rotta Saavedra

INTEGRANTES:

- García Chinguel, Emersson
- More Sernaque, Jhon Darwin
- Roque de la Cruz, Pedro Ignacio
- Vilca Aguilar, Luis Ángel

INFORME FINAL

Piura, 23 de enero de 2021

TABLA DE CONTENIDO

1 INTRODUCCIÓN

Este informe está destinado a abordar el desarrollo de nuestro trabajo realizado durante el curso de "Análisis de datos con Python Nivel 1", por lo cual escogimos realizar el trabajo del tipo 01, que incluye planteamiento de trabajo, análisis del sistema, desarrollo de este y conclusiones.

En el planteamiento del problema se describe el alcance que abarcará el programa junto con las herramientas a utilizar; en el análisis del sistema se explican lo que debe realizar el programa con la ayuda de un diagrama. En el desarrollo, se explicarán las funcionalidades de cada parte del código. Finalmente, las conclusiones, que mostrarán lo aprendido en el curso y en este trabajo.

2 PLANTEAMIENTO DEL TRABAJO

Se pide realizar un programa de sistematización para análisis de datos. El programa debe permitir, mediante la terminal, ingresar datos, guardarlos en una base de datos en Excel, y permitir mediante opciones que el usuario pueda hacer estas acciones:

- Ingresar datos a la base de datos y que se actualicen.
- Filtrar datos por medio de características.
- Solicitar gráficos de análisis entre algunas variables.
- Solicitar variables estadísticas de los datos.
- Generar nuevas características para los datos. (Nuevas columnas)

Se pueden usar las siguientes librerías para su ejecución: Numpy, Pandas, Matplotlib, Seaborn y Plotly

Figura 1: Nombres de las librerías a utilizar en el sistema

Fuente: Elaboración propia

3 ANÁLISIS DEL SISTEMA

De acuerdo al criterio descrito anteriormente, se requiere que el sistema/programa, se ejecute por consola y no mediante una interfaz gráfica, por lo cual se deberá establecer un loop principal para que el usuario pueda navegar por todas las funciones del programa, y escoger aquella función que necesite, desde el ingreso de datos hasta la solicitud de gráficas de análisis, siempre y cuando el usuario decida continuar, se le seguirán mostrando los paneles para elegir la función requerida, *se explicará de una mejor forma en el apartado siguiente.*

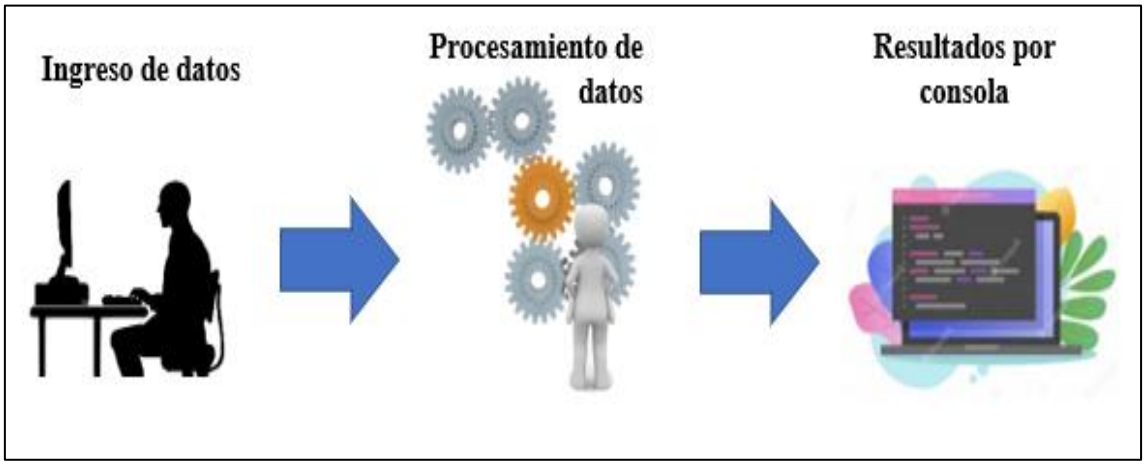


Figura 2: Diagrama del sistema

Fuente: Elaboración propia

4 DESARROLLO DEL SISTEMA

4.1 INGRESAR DATOS A LA BASE DE DATOS Y QUE SE ACTUALICEN.

```
def ingresar_datos():

  info = []
  pregunta = 's'
  while pregunta == 'S' or pregunta == 's':
    global database_df
    columnas = list(database_df.columns.values)

    for i in range(0,len(columnas)):

      valido = True

      while valido == True:
        print('Dato a agregar:', columnas[i])
        tipo_dato = input('Ingrese 1 si el dato es un número y 2 si es una cadena de texto: ')
        if tipo_dato == '1':
          info.append(float(input('Ingrese el valor: ')))
          print(separador2)
          valido = False

        elif tipo_dato == '2':
          info.append(input('Ingrese la cadena de texto: '))
          print(separador2)
          valido = False

        else:
          print(separador2)
          print('********Ingrese un valor válido')
          valido == True

    dic_info = dict(zip(columnas,info))
    database_df = database_df.append(dic_info, ignore_index=True)
    print(database_df)
    pregunta = input('S para agregar otro elemento u otra tecla para salir: ')
    info = []

    if pregunta == 'S' or pregunta == 's':
      continue
    else:
      break
"""
```

Figura 3: Función para ingresar datos

Fuente: Elaboración propia

La función para ingresar datos inicia pidiendo que se agregue el dato que se encuentra en la primera columna del dataframe, así como el formato de este, seguido, con un input se procede a agregar la información en la lista “info[ ]”; el proceso se repite hasta la última columna. Luego se procede a crear un diccionario con la lista de las columnas y la información ingresada con el fin de agregarlo al dataframe original y actualizarlo, finalmente se pregunta al usuario si desea agregar otro elemento y dependiendo de su elección se continúa o sale del programa, asimismo se reinicia la lista “info[ ]”. A continuación, se presenta un ejemplo.

	Precio	Área(m2)	Cuartos	Baños
0	612.0	333.0	6.0	6.0
1	380.0	153.0	3.0	2.0
2	428.0	320.0	5.0	2.0
3	370.5	120.0	2.0	1.0
4	2535.0	550.0	6.0	8.0
5	716.0	200.0	4.0	4.0
6	1580.0	513.0	3.0	3.0
7	1312.0	200.0	4.0	2.0
8	3160.0	981.0	3.0	4.0
9	740.0	189.0	4.0	4.0
10	1105.0	360.0	3.0	3.0
11	1800.0	500.0	6.0	4.0
12	640.0	160.0	3.0	3.0
13	1189.0	365.0	8.0	6.0
14	430.0	153.0	4.0	3.0
15	1156.0	400.0	4.0	3.0
16	320.0	92.0	3.0	1.0
17	248.0	204.0	3.0	3.0
18	640.0	160.0	4.0	3.0
19	796.0	265.0	8.0	7.0
20	1294.8	347.0	8.0	8.0
21	920.0	184.0	3.0	2.0
22	195.0	120.0	3.0	2.0
23	905.0	147.0	3.0	2.0
24	200.0	140.0	1.0	1.0
25	300.0	100.0	3.0	2.0

Figura 4: Ejemplo de agregar datos en una tabla

Fuente: Elaboración propia

En esta imagen se aprecia como se agregaron dos datos al dataframe original. Se ha agregado una fila adicional. Cabe recalcar que se han incluido números flotantes en las columnas de cuartos y baños con el objetivo de poder hacer cálculos con sus respectivos datos.

4.2 FILTRAR DATOS POR MEDIO DE CARACTERÍSTICAS.

En cuanto al filtrado de datos, la función solicitara el nombre de una columna al usuario, la cual se validará si existe dentro del dataframe, si es que no existe, simplemente la función terminara su ejecución, por otro lado, si es que existe, se le pedirá el parámetro de entrada y su tipo de dato, para que la función pueda filtrar dentro de todos los registros del dataframe, y devolver los valores encontrados al usuario, y si es que no existen, se le devolverá un mensaje de todas formas.

```python
def filtrarDataFrame(database_df):
  columnas_df = list(database_df.columns.values) #Retorna una lista con las columnas
  print("Estos son las columnas que puede ingresar para el filtrado: ", list(columnas_df))
  print(separador)

  columna = input("Ingrese la columna escogida: ")
  if columna in columnas_df:
    print(separador)
    pregunta = input("¿El parametro es una cadena de texto o un numero? (texto/numero): ")
    print(separador)
    if pregunta == 'texto':
      parametro = input("Ingrese el parametro de busqueda: ")

    elif pregunta == 'numero':
      parametro = int(input("Ingrese el parametro de busqueda: "))

    else:
      return False

    datos = database_df.loc[database_df[columna]==parametro]

    print(separador)

    if len(datos)>0:
      return print(datos)

    else:
      return print(f"{columna.upper()} no encontrad@ con el parametro '{parametro}'")

  else:
    print(separador)
    print("Columna no encontrada, REINICIANDO... =)")
    print(separador)
    filtrarDataFrame(database_df)
```

Figura 5: Función para filtrar datos

Fuente: Elaboración propia

```
Ingrese la ruta del archivo excel /content/Lec8_Pro2.xlsx
==========================================================================
Panel de opciones:
1) Ingresar datos
2) Filtrar datos
3) Gráficos de análisis
4) Datos estadísticos
5) Generar nuevas columnas
* Otra tecla para salir
==========================================================================
2
Estos son las columnas que puede ingresar para el filtrado:  ['Precio', 'Área(m2)', 'Cuartos', 'Baños']
================================================
Ingrese la columna escogida: Cuartos
================================================
¿El parametro es una cadena de texto o un numero? (texto/numero): numero
================================================
Ingrese el parametro de busqueda: 4
================================================
    Precio  Área(m2)  Cuartos  Baños
5    716.0       200        4      4
7   1312.0       200        4      2
9    740.0       189        4      4
14   430.0       153        4      3
15  1156.0       400        4      3
18   640.0       160        4      3
==========================================================================
```

Figura 6: Ejemplo de filtrar los datos

Fuente: Elaboración propia

4.3 SOLICITAR VARIABLES ESTADÍSTICAS DE LOS DATOS.

```
def datos_estadisticos():
  variable = []
  variable.append(input('Ingrese el nombre de la columna a analizar '))

  #No ingresa ningún valor
  if variable == '':
    print('No ha ingresado ningún nombre')

  #Ingresa el nombre correctamente
  else:
    #Ingresar varias columnas
    pregunta = 's'
    while pregunta == 'S' or pregunta == 's':
      pregunta = input('Escriba S si desea agregar otra columna u otra tecla para terminar el programa ')
      if pregunta == 'S' or pregunta == 's':
        variable.append(input('Ingrese el nombre de la nueva columna a analizar '))
        print(variable)
        continue
      else:
        break

  print(database_df[variable].describe())
```

Figura 7: Función para solicitar datos estadísticos

Fuente: Elaboración propia

En el caso de la función de datos estadísticos, al iniciar se solicita el nombre de la columna que se desea analizar, la cual se agrega a una lista "variable[ ]". Después, el programa solicita al usuario si desea agregar más nombres de columnas, si ese es el caso se procede a seguir llenando la matriz hasta que el usuario ya no desee, finalmente se presenta la información con los datos estadísticos más importantes. Para ello, se presenta un ejemplo.

```
Ingrese el nombre de la columna a analizar Precio
Escriba S si desea agregar otra columna u otra tecla para terminar el programa s
Ingrese el nombre de la nueva columna a analizar Cuartos
['Precio', 'Cuartos']
Escriba S si desea agregar otra columna u otra tecla para terminar el programa e
            Precio    Cuartos
count    24.000000  24.000000
mean    978.012500   4.291667
std     721.281453   1.781039
min     195.000000   2.000000
25%     429.500000   3.000000
50%     768.000000   4.000000
75%    1215.450000   5.250000
max    3160.000000   8.000000
```

Figura 8: Ejemplo para hallar los datos estadísticos

Fuente: Elaboración propia

En la imagen se presenta dos columnas seleccionadas del dataframe con sus respectivas variables estadísticas: cantidad de datos, media, desviación estándar, valor mínimo, valor máximo y los percentiles.

4.4 SOLICITAR GRÁFICOS DE ANÁLISIS ENTRE ALGUNAS VARIABLES.

En cuanto a la función de para solicitar gráficas de análisis, el programa solo pedirá al usuario que ingrese un tipo de gráfico disponible, ya sea del tipo **a**, **b** o **c**, si el usuario ingresa una opción fuera de rango, le saldrá un mensaje gracias a la función *chooseCollection (dato, colección [ ])***,** que se encarga de revisar que cualquier parámetro exista dentro de una colección de opciones.

Una vez, se escoja un tipo de gráfico, se le solicitará al usuario los parámetros que requiere dicho gráfico, por ejemplo; ejeX, ejeY, hue (opcional), alpha (opcional), bins (opcional).

Cuanto el programa reciba todos los parámetros, usando la librería de matplotlib y seaborn, se muestran los gráficos, como se muestra en la imagen de arriba.

```
#############################################################################
"""
-> Función para elegir el tipo de gráfico...
"""

def chooseChart():
  print(f"\na) Gráfico de dispersion \nb) Gráfico de distribución (Histograma)\nc) Gráfico de densidad\n{separador}")
  while True:
    tipo = input("Escoja la opción que desee: ").lower()
    if chooseCollection(tipo, ['a','b','c']):
      ejeX = isItstDataType("Ingrese la columna para el ejeX: ")
      eval(f"chartType{tipo.upper()}('{ejeX}')")
      break
chooseChart()

#############################################################################
```

Figura 9:Funión para elegir el tipo de gráfico

Fuente: Elaboración propia

```
a) Gráfico de dispersion
b) Gráfico de distribución (Histograma)
c) Gráfico de densidad
==================================================
Escoja la opción que desee: c
Escogiste la opcion c
==================================================
Ingrese la columna para el ejeX: Precio
Escogiste la opcion Precio
==================================================
Ingrese la columna para el ejeY (Opcional):
Al no ingresar valor, se tomo como valor por defecto: None
==================================================
```

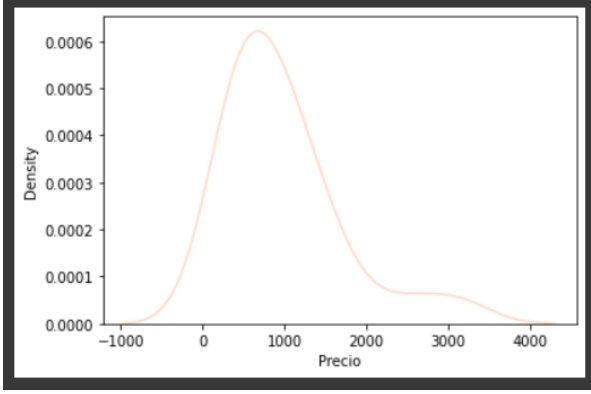


Figura 10: Gráfico de densidad

Fuente: Elaboración Propia

```
#########################################################################
"""
-> Funciones para distintos tipos de gráficos...
"""
# chartTypeA() => dispersion / columna(s) / size / color / alpha (Competado)
# chartTypeB() => histograma / -> columna / kde / bins / hue / alpha (Completado)
# chartTypeC() => densidad / -> columna / paleta (Completado)

def chartTypeA(ejeX):
  ejeY = isItstDataType("Ingrese la columna para el ejeY: ")
  hue_ = isItstDataType("Ingrese la columna para comparar: (Opcional) ", True)
  alph = isItstDataType("Ingrese la opacidad del gráfico (min: 0.10, max: 1.00) (Opcional):  ", True, True

  sns.relplot(x=ejeX, y=ejeY, hue = hue_, alpha = alph, data = database_df)
  plt.show()

def chartTypeB(ejeX):
  hue_ = isItstDataType("Ingrese la columna para comparar: (Opcional) ", True)
  alph = isItstDataType("Ingrese la opacidad del gráfico (min: 0.10, max: 1.00) (Opcional):  ", True, True
  bin = isItstDataType("Ingrese la cantidad de bins del gráfico (min: 1, max: 200):  ", False, True, 1, 20

  sns.displot(database_df, x=ejeX, hue=hue_, alpha=alph, bins=int(bin))
  plt.show()

def chartTypeC(ejeX):
  ejeY = isItstDataType("Ingrese la columna para el ejeY (Opcional): ", True)
  sns.kdeplot(data = database_df, x = ejeX, y = ejeY)
  plt.show()

#########################################################################
```

Figura 11: Función para la configuración de los gráficos

Fuente: Elaboración propia

4.5 GENERAR NUEVAS CARACTERÍSTICAS PARA LOS DATOS. (NUEVAS COLUMNAS)

Para evaluar este requisito del programa se creó una función denominada *Agregar_columna()* el cual cumple la función de agregar las columnas que el usuario desee a la tabla digital principal. Ver figura 12

A continuación, se muestra un ejemplo en la figura 13 donde se aprecia que se agregó la columna *Ubicación*, estos datos se agregaron aleatoriamente de algunos lugares de Piura. Se aprecia que dicha columna se ha agregado satisfactoriamente.

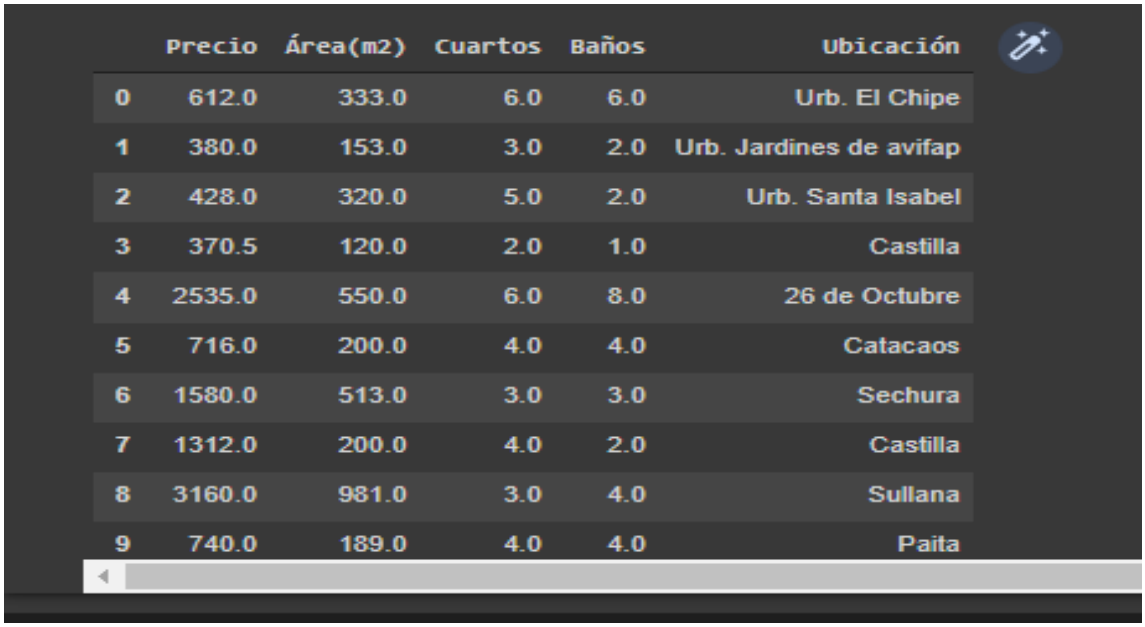
```
def Agregar_columna():
  decision = input('Si desea agregar otra columna ingrese S, de lo contrario cualquier otra letra: ')
  while decision == 'S' or decision == 's':
    nombre = input('Ingresa el nombre de la columna: ')
    datos_columna = []
    Num_datos = database_df.shape[0]


    j = int(Num_datos)
    for i in range(0,j):
      datos = input('Ingrese los datos del índice {}: '.format(i))
      datos_columna.append(datos)
    database_df[nombre] = datos_columna
    decision = input('Si desea agregar otra columna ingrese S, de lo contrario cualquier otra letra: ')

  print("-----------------------------------------------------------------------------------------------------------")
  print('')
  print('No se agregaron más columnas')
  print('')
  display(database_df)
```

Figura 12: Función para agregar columnas en la tabla digital

Fuente: Elaboración propia



	Precio	Área(m2)	Cuartos	Baños	Ubicación
0	612.0	333.0	6.0	6.0	Urb. El Chipe
1	380.0	153.0	3.0	2.0	Urb. Jardines de avifap
2	428.0	320.0	5.0	2.0	Urb. Santa Isabel
3	370.5	120.0	2.0	1.0	Castilla
4	2535.0	550.0	6.0	8.0	26 de Octubre
5	716.0	200.0	4.0	4.0	Catacaos
6	1580.0	513.0	3.0	3.0	Sechura
7	1312.0	200.0	4.0	2.0	Castilla
8	3160.0	981.0	3.0	4.0	Sullana
9	740.0	189.0	4.0	4.0	Paita

Figura 13: Tabla digital y una columna adicional ingresada por el usuario

Fuente: Elaboración propia

5 CONCLUSIONES

- El Lenguaje de Programación Python, permite trabajar de una manera cada vez más sistematizada por disponer de una sintaxis clara y sencilla, en esto se puede combinar los diferentes paradigmas de programación de los que se dispone y por ende se consigue la reducción en el tiempo de trabajo.
- Sistematizar estas operaciones para el análisis de datos, conlleva cumplir con la sintaxis correcta, saber identificar el tipo de dato, sus funciones integradas, usar de manera adecuada sus librerías, etc. Esto para poner especial énfasis en saber qué librería, método, función, a utilizar con el fin de tener un programa muy eficiente.
- En los programas que se han escrito y ejecutado, se ha puesto hincapié en la manera distinta de organizar la información y en su representación jerárquico-relacional entre los datos que permiten realizar las tareas para los que se ha escrito, y las instrucciones necesarias para seleccionar registros y campos requeridos de una base de datos.

6 ANEXOS

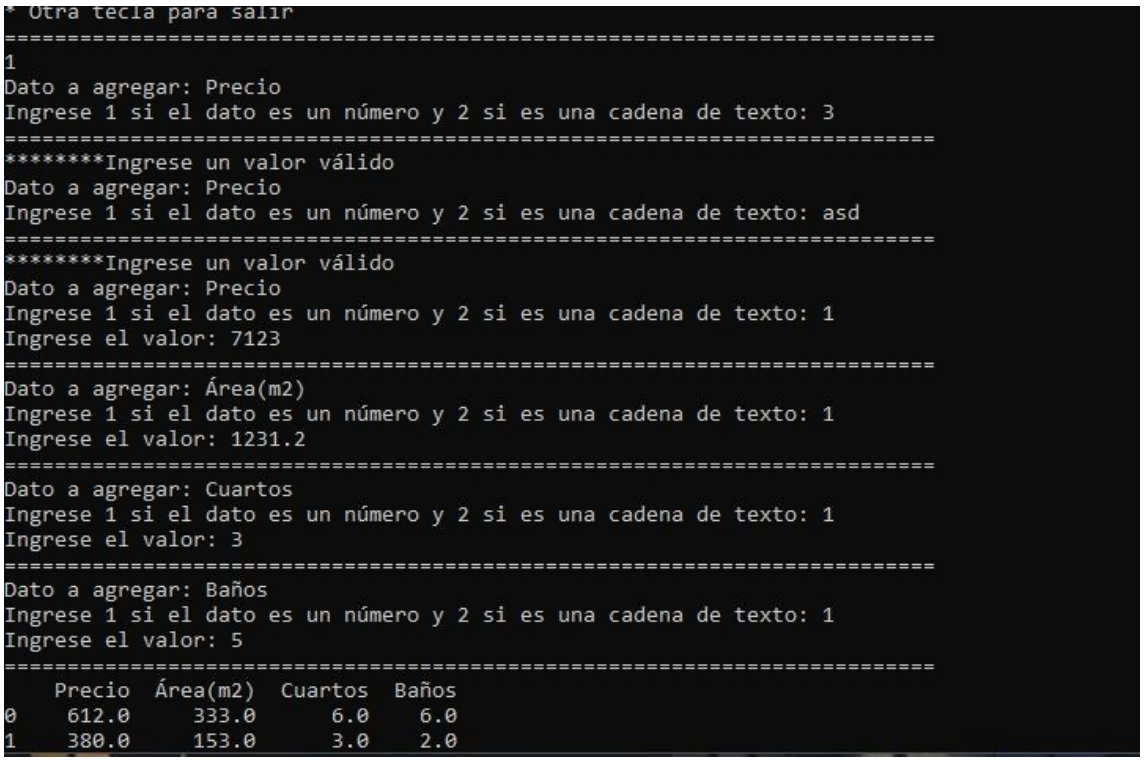


Figura 14: Ingreso de datos con la función ingresar_datos()

Fuente: Elaboración propia

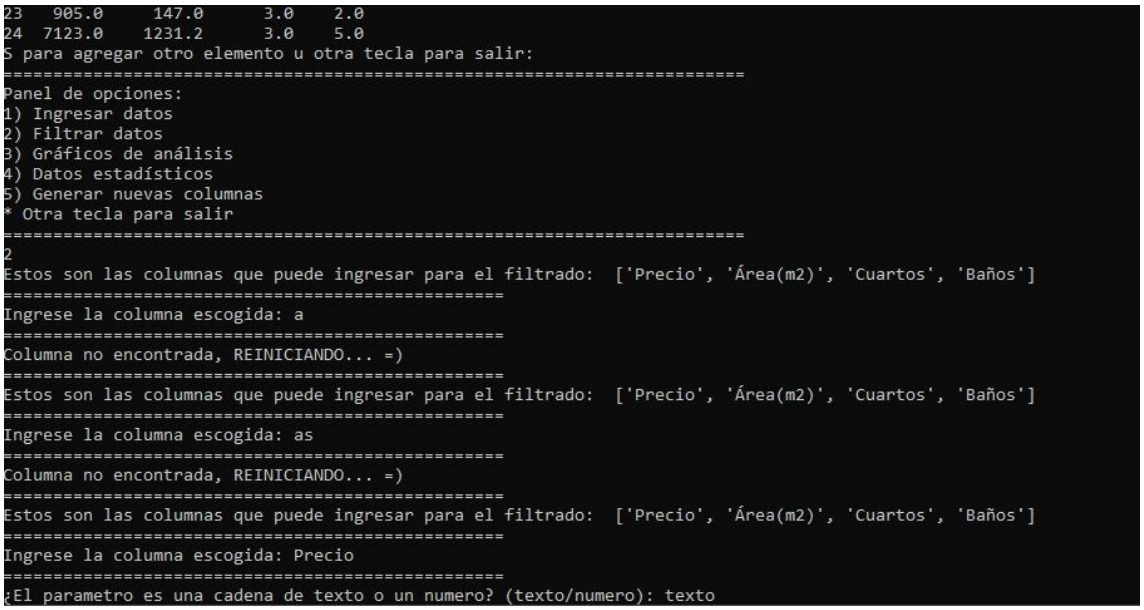


Figura 15: Filtrar datos con filtrarDataFrame(database_df)

Fuente: Elaboración propia

```
Ingrese la columna escogida: Precio
==================================================
¿El parametro es una cadena de texto o un numero? (texto/numero): numero
==================================================
Ingrese el parametro de busqueda: 612
==================================================
   Precio  Área(m2)  Cuartos  Baños
0   612.0     333.0      6.0    6.0
================================================================================
```

Figura 16: Respuesta del filtrado

Fuente: Elaboración propia

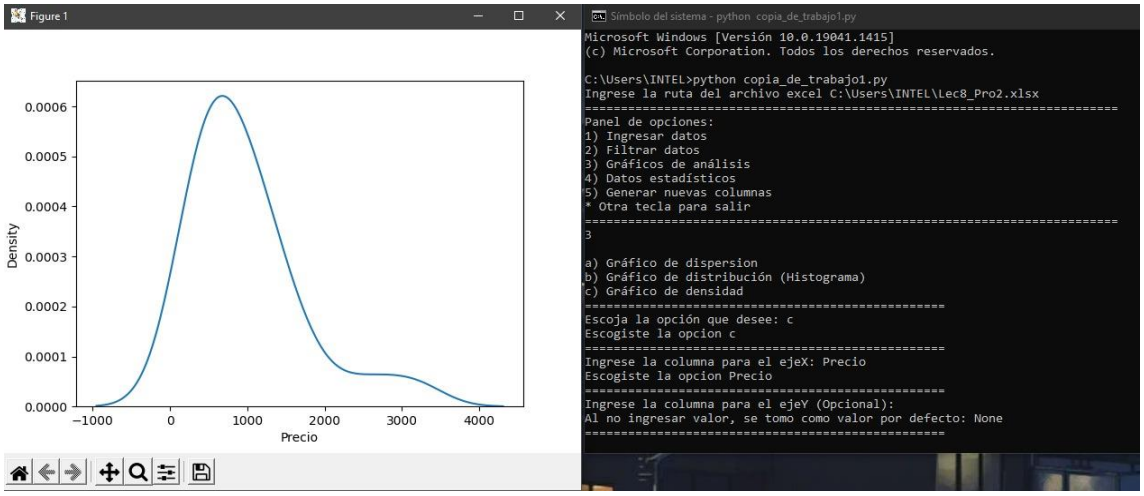


Figura 17: Gráfico generado con la función chooseChart()

Fuente: Elaboración propia

```
Símbolo del sistema - python copia_de_trabajo1.py
==================================================
Ingrese el nombre de la columna a analizar
Dato no válido
==================================================
Ingrese el nombre de la columna a analizar Àrea(m2)
Dato no válido
==================================================
Ingrese el nombre de la columna a analizar Área(m2)
Escogiste la opcion Área(m2)
==================================================
Escriba S si desea agregar otra columna u otra tecla para terminar el programa
            Precio      Baños    Cuartos    Área(m2)
count    24.000000  24.000000  24.000000   24.000000
mean    978.012500   3.583333   4.291667  292.333333
std     721.281453   2.019829   1.781039  198.017493
min     195.000000   1.000000   2.000000   92.000000
25%     429.500000   2.000000   3.000000  158.250000
50%     768.000000   3.000000   4.000000  202.000000
75%    1215.450000   4.000000   5.250000  361.250000
max    3160.000000   8.000000   8.000000  981.000000
================================================================================
Panel de opciones:
1) Ingresar datos
2) Filtrar datos
3) Gráficos de análisis
4) Datos estadísticos
5) Generar nuevas columnas
* Otra tecla para salir
================================================================================
```

Figura 18: Datos estadísticos generados por la función datos_estadisticos()

Fuente: Elaboración propia

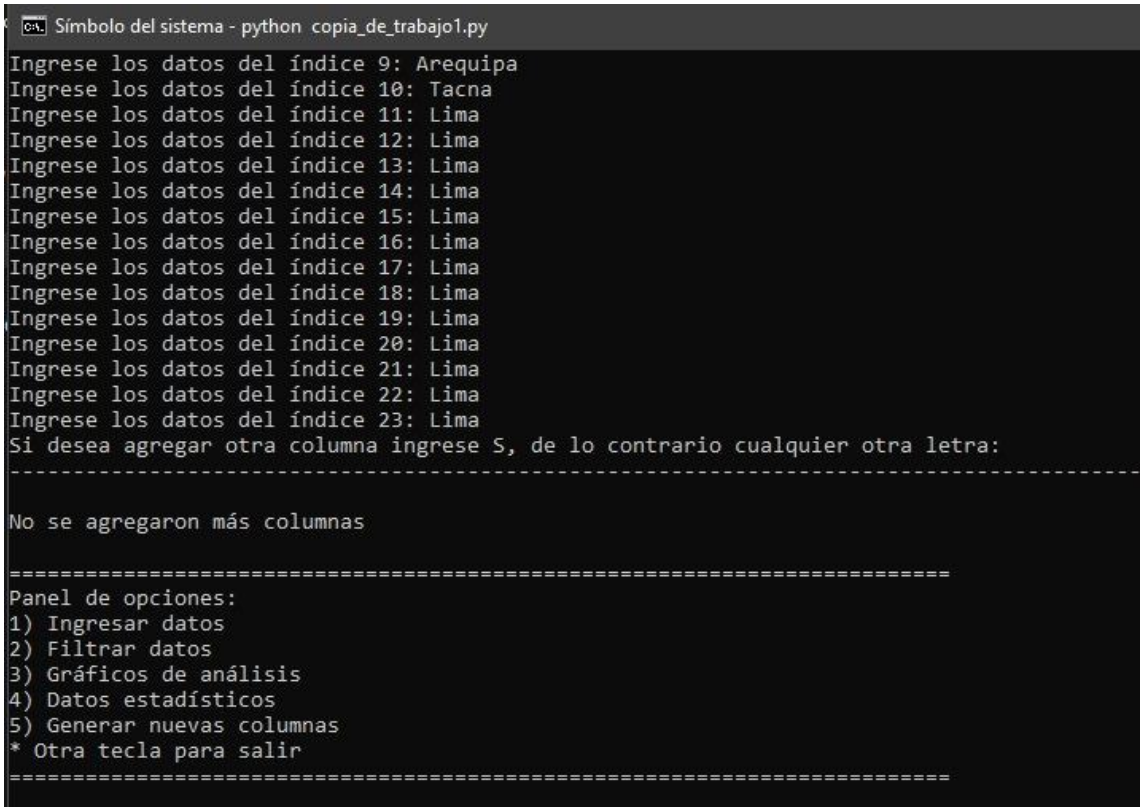


Figura 19: Agregar columnas con la función Agregar_columna()

Fuente: Elaboración propia

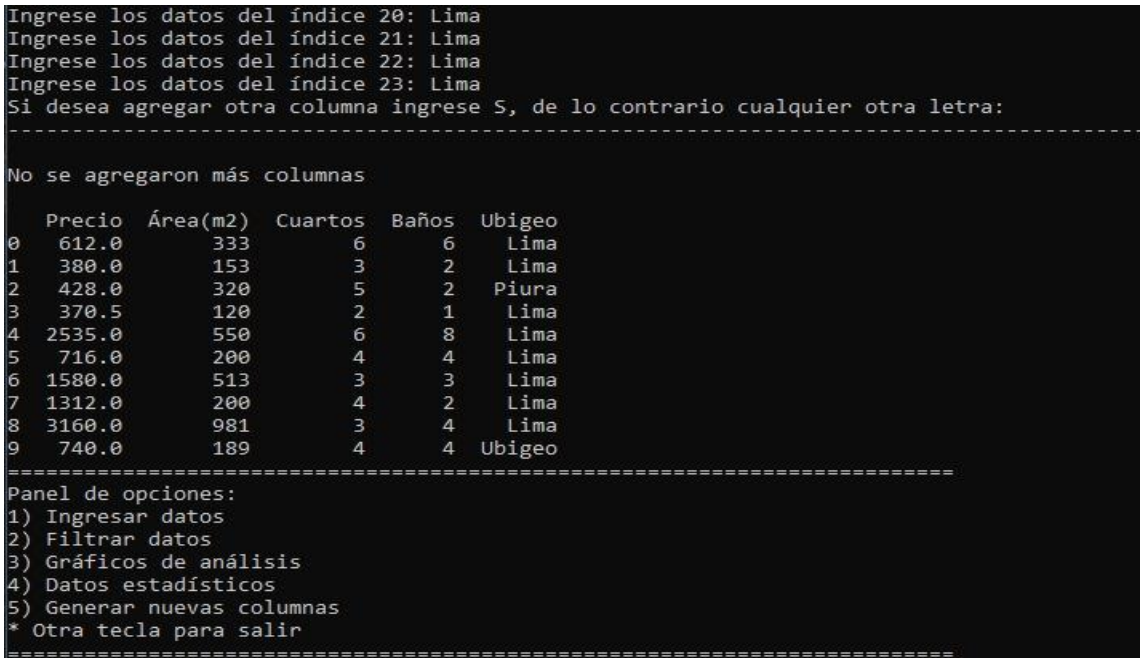


Figura 20: Comprobación de que se agregó la columna

Fuente: Elaboración propia