

SCRIPTS PYTHON Y R

Cosas ha tener en cuenta antes de empezar

- Descargar la Tabla : "Walmart_Sales.csv"
- Crear una base de Datos en el SSMS
- Importar la Tabla

SCRIPTS DE ESTADISTICA CON PYTHON

En mi caso usare VSCode

1. Instalar la librería `pyodbc`


```
pip install pyodbc
```

2. Hacer la conexión de la Base de Datos de SQL Server y Python, en este caso cuento con el método de autenticación por Windows por lo que se coloca al hacer la conexión `Trusted_Connection=yes;`

```
import pyodbc
#Importa la libreria desacrada anteriormente
try:
    connection = pyodbc.connect('DRIVER={SQL Server};SERVER=.;DATABASE=Pruebaa;Trusted_Connection=yes;')
    #En mi caso mi servidor es " . " y mi base de datos se llama "Pruebaa", reemplazas por los valores que tengas
    print("Conexion Exitosa") #Ponemos este mensaje para que nos confirme la conexion
    cursor = connection.cursor()

except Exception as ex:
    print(ex)
finally:
    if connection:
        connection.close() #cerramos la conexion para ahorrar recursos
```

3. En caso sea Autenticación por SQL se colocaría lo siguiente agregando el `"user"` que es el usuario y `"pwd"` que sería la contraseña reemplazando sus respectivos valores (Todo debe ir junto, lo separe para que entre en la captura de pantalla)



Este ejemplo Nos ayuda a establecer la conexión con la base de datos y la versión que disponemos del SSMS

```
import pyodbc
try:
    connection=pyodbc.connect('DRIVER={SQL Server};SERVER=.;DATABASE=Pruebaa;Trusted_Connection=yes;')
    print("Conexion Exitosa")
    cursor=connection.cursor()
    cursor.execute("SELECT @@version;") #Muestra nuestra version de SSMS
    row=cursor.fetchone() #El cursor se utiliza para ejecutar comandos SQL
    #cursor.fetchall() se utiliza para recuperar solo una fila resultante de una consulta SQL
    print(row)
except Exception as ex:
    print(ex)

connection.close()
```

Recuerda → Para Ejecutar cada Script debe estar en la ruta del archivo en el terminal de VSCode y poner `python3 (nombrearchivo).py`

▼ Mostrar todos los Datos de la Tabla

```
import pyodbc
try:
    connection = pyodbc.connect('DRIVER={SQL Server};SERVER=.;DATABASE=Pruebaa;Trusted_Connection=yes;')
    print("Conexion Exitosa")
    cursor = connection.cursor()

    cursor.execute("SELECT * FROM Walmart_sales")
    rows = cursor.fetchall()
    ##cursor.fetchall() se utiliza para recuperar todas las filas resultantes de una consulta SQL
    for row in rows:
        # Imprimir las filas
        print(row)

except Exception as ex:
    print(ex)
finally:
    if connection:
        connection.close()
```

▼ Calcular medidas estadísticas para 'Weekly_Sales'

```
import pyodbc

try:
    connection = pyodbc.connect('DRIVER={SQL Server};SERVER=.;DATABASE=Pruebaa;Trusted_Connection=yes;')
    print("Conexion Exitosa")
    cursor = connection.cursor()

    # Calcular medidas estadísticas para 'Weekly_Sales'
    cursor.execute("SELECT AVG(Weekly_Sales) AS Media_Weekly_Sales FROM Walmart_sales")
    media_ventas = cursor.fetchone()
    print(f"Media de Weekly_Sales: {media_ventas[0]}")

    cursor.execute("SELECT PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY Weekly_Sales) OVER () AS Mediana_Weekly_Sales FROM Walmart_sales")
    mediana_ventas = cursor.fetchone()
    print(f"Mediana de Weekly_Sales: {mediana_ventas[0]}")

    cursor.execute("SELECT TOP 1 WITH TIES Weekly_Sales AS Moda_Weekly_Sales FROM Walmart_sales ORDER BY COUNT(*) OVER (PARTITION BY Weekly_Sales) DESC")
    moda_ventas = cursor.fetchone()
    print(f"Moda de Weekly_Sales: {moda_ventas[0]}")

    cursor.execute("SELECT STDEV(Weekly_Sales) AS Desviacion_Estandar_Weekly_Sales FROM Walmart_sales")
    desviacion_ventas = cursor.fetchone()
    print(f"Desviación Estándar de Weekly_Sales: {desviacion_ventas[0]}")

except Exception as ex:
    print(ex)
finally:
    if connection:
        connection.close()
```

▼ Calcular medidas estadísticas para 'Temperature' y para 'Unemployment'

```
import pyodbc

try:
    connection = pyodbc.connect('DRIVER={SQL Server};SERVER=.;DATABASE=Pruebaa;Trusted_Connection=yes;')
    print("Conexion Exitosa")
    cursor = connection.cursor()

    # Calcular medidas estadísticas para 'Temperature'
    cursor.execute("SELECT AVG(Temperature) AS Media_Temperature FROM Walmart_sales")
    media_temperatura = cursor.fetchone()
    print(f"Media de Temperature: {media_temperatura[0]}")

    cursor.execute("SELECT PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY Temperature) OVER () AS Mediana_Temperature FROM Walmart_sales")
    mediana_temperatura = cursor.fetchone()
    print(f"Mediana de Temperature: {mediana_temperatura[0]}")

except Exception as ex:
    print(ex)
finally:
    if connection:
        connection.close()
```

```

cursor.execute("SELECT TOP 1 WITH TIES Temperature AS Moda_Temperature FROM Walmart_sales ORDER BY COUNT(*) OVER (PARTITION BY Temperature) DESC")
moda_temperatura = cursor.fetchone()
print(f"Moda de Temperature: {moda_temperatura[0]}")

cursor.execute("SELECT STDEV(Temperature) AS Desviacion_Estandar_Temperature FROM Walmart_sales")
desviacion_temperatura = cursor.fetchone()
print(f"Desviación Estándar de Temperature: {desviacion_temperatura[0]}")

# Calcular medidas estadísticas para 'Unemployment'
cursor.execute("SELECT AVG(Unemployment) AS Media_Unemployment FROM Walmart_sales")
media_desempleo = cursor.fetchone()
print(f"Media de Unemployment: {media_desempleo[0]}")

cursor.execute("SELECT PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY Unemployment) OVER () AS Mediana_Unemployment FROM Walmart_sales")
mediana_desempleo = cursor.fetchone()
print(f"Mediana de Unemployment: {mediana_desempleo[0]}")

cursor.execute("SELECT TOP 1 WITH TIES Unemployment AS Moda_Unemployment FROM Walmart_sales ORDER BY COUNT(*) OVER (PARTITION BY Unemployment) DESC")
moda_desempleo = cursor.fetchone()
print(f"Moda de Unemployment: {moda_desempleo[0]}")

cursor.execute("SELECT STDEV(Unemployment) AS Desviacion_Estandar_Unemployment FROM Walmart_sales")
desviacion_desempleo = cursor.fetchone()
print(f"Desviación Estándar de Unemployment: {desviacion_desempleo[0]}")

except Exception as ex:
    print(ex)
finally:
    if connection:
        connection.close()

```

▼ Mostrar los 10 primeros registros

```

import pyodbc

try:
    connection = pyodbc.connect('DRIVER={SQL Server};SERVER=.;DATABASE=Pruebasaa;Trusted_Connection=yes;')
    print("Conexion Exitosa")
    cursor = connection.cursor()

    # Obtener solo los primeros 10 registros de 'Walmart_sales'
    cursor.execute("SELECT TOP 10 * FROM Walmart_sales")
    rows = cursor.fetchall()

    for row in rows:
        print(row)

except Exception as ex:
    print(ex)
finally:
    if connection:
        connection.close()

```

▼ Mostrar los 10 últimos registros

```

import pyodbc

try:
    connection = pyodbc.connect('DRIVER={SQL Server};SERVER=.;DATABASE=Pruebasaa;Trusted_Connection=yes;')
    print("Conexion Exitosa")
    cursor = connection.cursor()

    # Obtener los últimos 10 registros de 'Walmart_sales'
    cursor.execute("SELECT TOP 10 * FROM Walmart_sales ORDER BY Store DESC")
    # Reemplaza 'Store' con la columna por la cual quieres ordenar por ejemplo 'Date'
    rows = cursor.fetchall()
    for row in rows:
        print(row)

except Exception as ex:
    print(ex)
finally:
    if connection:
        connection.close()

```

▼ Medidas Estadísticas con "Fuel_Price"

```

import pyodbc

try:
    connection = pyodbc.connect('DRIVER={SQL Server};SERVER=.;DATABASE=Pruebasaa;Trusted_Connection=yes;')
    print("Conexion Exitosa")
    cursor = connection.cursor()

    # Calcular medidas estadísticas para 'Fuel_Price'
    cursor.execute("SELECT AVG(Fuel_Price) AS Media_Fuel_Price FROM Walmart_sales")
    media_fuel_price = cursor.fetchone()
    print(f"Media de Fuel_Price: {media_fuel_price[0]}")

    cursor.execute("SELECT PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY Fuel_Price) OVER () AS Mediana_Fuel_Price FROM Walmart_sales")
    mediana_fuel_price = cursor.fetchone()
    print(f"Mediana de Fuel_Price: {mediana_fuel_price[0]}")

    cursor.execute("SELECT TOP 1 WITH TIES Fuel_Price AS Moda_Fuel_Price FROM Walmart_sales ORDER BY COUNT(*) OVER (PARTITION BY Fuel_Price) DESC")
    moda_fuel_price = cursor.fetchone()
    print(f"Moda de Fuel_Price: {moda_fuel_price[0]}")

    cursor.execute("SELECT STDEV(Fuel_Price) AS Desviacion_Estandar_Fuel_Price FROM Walmart_sales")
    desviacion_fuel_price = cursor.fetchone()
    print(f"Desviación Estándar de Fuel_Price: {desviacion_fuel_price[0]}")

except Exception as ex:
    print(ex)
finally:
    if connection:
        connection.close()

```

▼ Cálculo de Percentiles, Cuartiles, Rango Intercuartílico, Varianza y Coeficiente de Variación

```

import pyodbc
import numpy as np

try:
    # Configuración de la conexión a la base de datos
    connection = pyodbc.connect('DRIVER={SQL Server};SERVER=.;DATABASE=Pruebasaa;Trusted_Connection=yes;')
    print("Conexion Exitosa")
    cursor = connection.cursor()

    # Ejemplo de datos, reemplázalos con tu propia consulta SQL
    cursor.execute("SELECT Temperature, Fuel_Price, Unemployment FROM Walmart_sales")
    resultados = cursor.fetchall()

    # Extraer columnas específicas
    temperature = [row.Temperature for row in resultados if row.Temperature is not None]
    fuel_price = [row.Fuel_Price for row in resultados if row.Fuel_Price is not None]
    unemployment = [row.Unemployment for row in resultados if row.Unemployment is not None]

```

```

# Verificar si hay datos antes de realizar cálculos
if temperature:
    # Percentiles y Cuartiles para Temperature
    p25_temp = np.percentile(temperature, 25)
    p50_temp = np.percentile(temperature, 50)
    p75_temp = np.percentile(temperature, 75)

    q1_temp = np.percentile(temperature, 25)
    q2_temp = np.percentile(temperature, 50)
    q3_temp = np.percentile(temperature, 75)

    # Imprimir resultados para Temperature
    print(f"Percentiles para Temperature: P25={p25_temp}, P50={p50_temp}, P75={p75_temp}")
    print(f"Cuartiles para Temperature: Q1={q1_temp}, Q2={q2_temp}, Q3={q3_temp}")
else:
    print("No hay datos válidos para Temperature.")

if fuel_price:
    # Rango Intercuartilico (IQR) para Fuel_Price
    iqr_fuel = np.percentile(fuel_price, 75) - np.percentile(fuel_price, 25)
    print(f"Rango Intercuartilico (IQR) para Fuel_Price: {iqr_fuel}")
else:
    print("No hay datos válidos para Fuel_Price.")

if unemployment:
    # Varianza y Coeficiente de Variación para Unemployment
    varianza_unemployment = np.var(unemployment)
    cv_unemployment = (np.std(unemployment) / np.mean(unemployment)) * 100
    print(f"Varianza para Unemployment: {varianza_unemployment}")
    print(f"Coeficiente de Variación para Unemployment: {cv_unemployment}")
else:
    print("No hay datos válidos para Unemployment.")

except Exception as ex:
    print(ex)
finally:
    if connection:
        connection.close()

```

▼ Con Matplotlib y pandas

Con Matplotlib podremos tener una visualización de nuestra consulta y Pandas nos ayudara a manejar los arreglos

Para ello instala previamente Pandas y Matplotlib con [pip install pandas](#) y [pip install matplotlib](#)

```

import pyodbc
import pandas as pd
import matplotlib.pyplot as plt
import io
import base64

try:
    # Conexión
    connection = pyodbc.connect('DRIVER={SQL Server};SERVER=.;DATABASE=Pruebas;Trusted_Connection=yes;')
    print("Conexion Exitosa")

    # Consultas SQL
    query_mediana = "SELECT AVG(Fuel_Price) AS Media_Fuel_Price FROM Walmart_sales"
    query_mediana = "SELECT PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY Fuel_Price) OVER () AS Mediana_Fuel_Price FROM Walmart_sales"
    query_moda = "SELECT TOP 1 WITH TIES Fuel_Price AS Moda_Fuel_Price FROM Walmart_sales ORDER BY COUNT(*) OVER (PARTITION BY Fuel_Price) DESC"
    query_desviacion = "SELECT STDEV(Fuel_Price) AS Desviacion_Estandar_Fuel_Price FROM Walmart_sales"

    # Ejecutar las consultas y obtener resultados en DataFrames
    media_result = pd.read_sql(query_mediana, connection)
    mediana_result = pd.read_sql(query_mediana, connection)
    moda_result = pd.read_sql(query_moda, connection)
    desviacion_result = pd.read_sql(query_desviacion, connection)

    # Crear un DataFrame consolidado con los resultados
    data = {
        'Medida Estadística': ['Media', 'Mediana', 'Moda', 'Desviación Estándar'],
        'Valor': [
            media_result['Media_Fuel_Price'][0],
            mediana_result['Mediana_Fuel_Price'][0],
            moda_result['Moda_Fuel_Price'][0],
            desviacion_result['Desviacion_Estandar_Fuel_Price'][0]
        ]
    }
    df_resultados = pd.DataFrame(data)

    # Visualizar como gráfico de barras
    plt.figure(figsize=(10, 6))
    plt.bar(df_resultados['Medida Estadística'], df_resultados['Valor'], color='blue')
    plt.title('Medidas Estadísticas de Fuel_Price')
    plt.xlabel('Medida Estadística')
    plt.ylabel('Valor')
    plt.show()

except Exception as ex:
    print(ex)
finally:
    if connection:
        connection.close()

```

SCRIPTS DE ESTADISTICA CON R

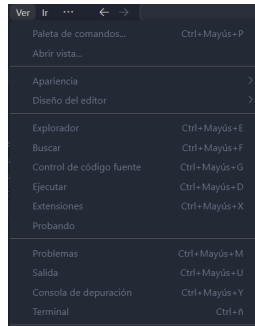
- 1. Instala R:**
 - Asegúrate de tener R instalado en tu sistema. Puedes descargar R desde <https://cran.r-project.org/>
- 2. Instala VSCode:**
 - Descarga e instala Visual Studio Code.
- 3. Instala la extensión de R:**
 - Abre VSCode y ve a la pestaña de extensiones (puedes hacerlo presionando **Ctrl + Shift + X**).
 - Busca "R" y selecciona la primera.
 - Haz clic en "Install" para instalar la extensión.
- 4. Abre un Nuevo Archivo R:**
 - Crea un nuevo archivo con extensión **.R** y ábrelo en VSCode.
- 5. Instala el Paquete odbc desde la Consola R:**
 - En el script R, abre la consola (puedes hacerlo presionando **Ctrl + Enter** cuando estás en una línea de código).
 - Ejecuta el siguiente comando para instalar el paquete odbc:

```
install.packages("odbc")
```

- 6. Carga el Paquete odbc:**
 - Después de la instalación, puedes cargar el paquete utilizando el siguiente comando en tu script R:

```
library(odbc)
```

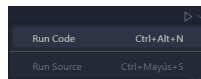
Nota: Para ejecutar el Terminal de R, Abrimos el archivo.r y en Ver le damos en Terminal



Recuerda que debes tener permisos de administrador para instalar paquetes en R
Para ejecutar los scripts en VSCode, le damos al boton de play en la parte superior derecha

```
medidas.r X
medidas
28 if (!requireNamespace("odbc", quietly = TRUE)) {
29   install.packages("odbc")
30 }
31
32 # Cargar la biblioteca odbc
33 library(odbc)
34
35 # Configuración de la conexión a la base de datos
36 con <- dbConnect(odbc::odbc(),
37                  driver = "SQL Server",
38                  server = ".",
39                  database = "Pruebas",
40                  trusted_connection = TRUE)
41
```

Y luego a "RUN SOURCE"



▼ Medidas Estadísticas para Fuel_Price

```
# Instala la biblioteca 'odbc' si no está instalada
if (!requireNamespace("odbc", quietly = TRUE)) {
  install.packages("odbc")
}

# Cargar la biblioteca 'odbc'
library(odbc)

# Configuración de la conexión a la base de datos
con <- dbConnect(odbc::odbc(),
                 driver = "SQL Server",
                 server = ".",
                 database = "Pruebas",
                 trusted_connection = TRUE)

# Realizar medidas estadísticas para 'Fuel_Price'
query <- "SELECT AVG(Fuel_Price) AS Media_Fuel_Price,
               STDEV(Fuel_Price) AS Desviacion_Estandar_Fuel_Price
          FROM Walmart_sales"

result <- dbGetQuery(con, query)

# Imprimir los resultados
print(result)

# Cerrar la conexión
dbDisconnect(con)
```

▼ Ahora Incluimos también la columna "Walmart_Sales"

```
# Instala los paquetes necesarios si no están instalados
if (!requireNamespace("odbc", quietly = TRUE)) {
  install.packages("odbc")
}

# Cargar la biblioteca odbc
library(odbc)

# Configuración de la conexión a la base de datos
con <- dbConnect(odbc::odbc(),
                 driver = "SQL Server",
                 server = ".",
                 database = "Pruebas",
                 trusted_connection = TRUE)

# Consulta para obtener todos los datos de la tabla 'Walmart_sales'
query <- "SELECT * FROM Walmart_sales"
walmart_data <- dbGetQuery(con, query)

# Muestra los primeros 10 registros
head(walmart_data, 10)

# Calcula algunas medidas estadísticas para 'Fuel_Price'
media_fuel_price <- mean(walmart_data$Fuel_Price, na.rm = TRUE)
mediana_fuel_price <- median(walmart_data$Fuel_Price, na.rm = TRUE)
desviacion_fuel_price <- sd(walmart_data$Fuel_Price, na.rm = TRUE)

# Imprime las medidas estadísticas
cat("Media de Fuel_Price:", media_fuel_price, "\n")
cat("Mediana de Fuel_Price:", mediana_fuel_price, "\n")
cat("Desviación Estándar de Fuel_Price:", desviacion_fuel_price, "\n")

# Cerrar la conexión
dbDisconnect(con)
```

▼ Mas Medidas estadísticas

```
# Instala los paquetes necesarios si no están instalados
if (!requireNamespace("odbc", quietly = TRUE)) {
  install.packages("odbc")
}
```

```

# Cargar la biblioteca odbc
library(odbc)

# Configuración de la conexión a la base de datos
con <- dbConnect(odbc::odbc(),
  driver = "SQL Server",
  server = ".",
  database = "Pruebaa",
  trusted_connection = TRUE)

# Consulta para obtener todos los datos de la tabla 'Walmart_sales'
query <- "SELECT * FROM Walmart_sales"
walmart_data <- dbGetQuery(con, query)

# Muestra los primeros 10 registros
head(walmart_data, 10)

# Calcula algunas medidas estadísticas para las columnas seleccionadas
media_temperature <- mean(walmart_data$Temperature, na.rm = TRUE)
mediana_temperature <- median(walmart_data$Temperature, na.rm = TRUE)
desviacion_temperature <- sd(walmart_data$Temperature, na.rm = TRUE)

media_weekly_sales <- mean(walmart_data$Weekly_Sales, na.rm = TRUE)
mediana_weekly_sales <- median(walmart_data$Weekly_Sales, na.rm = TRUE)
desviacion_weekly_sales <- sd(walmart_data$Weekly_Sales, na.rm = TRUE)

media_cpi <- mean(walmart_data$CPI, na.rm = TRUE)
mediana_cpi <- median(walmart_data$CPI, na.rm = TRUE)
desviacion_cpi <- sd(walmart_data$CPI, na.rm = TRUE)

media_fuel_price <- mean(walmart_data$Fuel_Price, na.rm = TRUE)
mediana_fuel_price <- median(walmart_data$Fuel_Price, na.rm = TRUE)
desviacion_fuel_price <- sd(walmart_data$Fuel_Price, na.rm = TRUE)

# Imprime las medidas estadísticas
cat("Medidas estadísticas para Temperature:\n")
cat(" Media:", media_temperature, "\n")
cat(" Mediana:", mediana_temperature, "\n")
cat(" Desviación Estándar:", desviacion_temperature, "\n\n")

cat("Medidas estadísticas para Weekly_Sales:\n")
cat(" Media:", media_weekly_sales, "\n")
cat(" Mediana:", mediana_weekly_sales, "\n")
cat(" Desviación Estándar:", desviacion_weekly_sales, "\n\n")

cat("Medidas estadísticas para CPI:\n")
cat(" Media:", media_cpi, "\n")
cat(" Mediana:", mediana_cpi, "\n")
cat(" Desviación Estándar:", desviacion_cpi, "\n\n")

cat("Medidas estadísticas para Fuel_Price:\n")
cat(" Media:", media_fuel_price, "\n")
cat(" Mediana:", mediana_fuel_price, "\n")
cat(" Desviación Estándar:", desviacion_fuel_price, "\n")

# Cerrar la conexión
dbDisconnect(con)

```