Para resolver este problema utilice alguna de las librerías básicas de python

El código conecta a una base de datos MySQL, lee datos de encuestas, analiza los comentarios usando una API de IA para clasificar la satisfacción de los clientes y genera una conclusión basada en los comentarios. Además de eso se crean gráficos para visualizar los resultados.

Para poder compilar este código es necesario contar con las librerías correspondientes

curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py

pip install mysql-connector-python pandas groq-python matplotlib seaborn numpy python-dotenv fpdf

| # importacion de librerias y configuración inicial  **import** mysql.connector **import** pandas **as** pd **from** datetime **import** datetime **from** groq **import** Groq **import** matplotlib.pyplot **as** plt **import** seaborn **as** sns **import** numpy **as** np **from** dotenv **import** load\_dotenv **import** os **from** fpdf **import** FPDF  # Configuración de Groq (otra IA que si es gratuita) load\_dotenv('api\_key.env') # Carga las variables de entorno desde el archivo .env  client = Groq(  api\_key = os.getenv('API\_KEY') , ) #Funcion para saber si el comentario da una valoracion satisfactoria, insatisfactoria o neutra **def** clasificar\_comentarios(comentarios):  clasificaciones = []  **for** comentario **in** comentarios:  response = client.chat.completions.create(  # model="mixtral-8x7b-32768", calidad del modelo: 5   model="gemma-7b-it", #calidad del modelo: 6  max\_tokens = 6,  n=1,  messages = [  {  "role": "user",  "content": f"Por favor clasifica el siguiente comentario como satisfactorio, insatisfactorio o neutro:\n\n{comentario}\n\n"}  ],    )  clasificacion = response.choices[0].message.content.strip().lower()   clasificaciones.append(clasificacion)  **return** clasificaciones   #Funcion para resumir los principales problemas de la empresa a modo de conclusion **def** generar\_conclusion(comentarios):  prompt = "A continuación se presentan algunos comentarios sobre una empresa. Por favor, proporciona un análisis de los problemas principales y una conclusión general basada en estos comentarios:\n\n"  **for** comentario **in** comentarios:  prompt += f"- {comentario}\n"  prompt += "\nProblemas principales y conclusión:"   response = client.chat.completions.create(  model = 'llama3-8b-8192',  max\_tokens = 300,  n = 1,  messages = [  {  "role": "user",   "content": prompt  }  ],  )  conclusion = response.choices[0].message.content.strip()  **return** conclusion  #ver cantidad de comentarios satisfactorios, insatisfactorios y neutros **def** contar\_clasificaciones(clasificaciones):  satisfactorio\_count = 0  insatisfactorio\_count = 0  neutro\_count = 0    **for** clasificacion **in** clasificaciones:  **if** '\*\*satisfactorio.\*\*' **in** clasificacion.lower():  satisfactorio\_count += 1  **if** 'insatisfactorio' **in** clasificacion.lower():  insatisfactorio\_count += 1  **if** 'neutro' **in** clasificacion.lower():  neutro\_count += 1  **return** {  'satisfactorio': satisfactorio\_count,  'insatisfactorio': insatisfactorio\_count,  'neutro': neutro\_count  }   # Conexión a la base de datos db\_config = {  'user': 'postulante',  'password': 'HB<tba!Sp6U2j5CN',  'host': '54.219.2.160',  'database': 'prueba\_postulantes' }   conn = mysql.connector.connect(\*\*db\_config) cursor = conn.cursor()   # Lectura de la tabla MySQL con los datos de la encuesta query = "SELECT \* FROM encuesta" cursor.execute(query) rows = cursor.fetchall() columns = cursor.column\_names df = pd.DataFrame(rows, columns=columns)   # Cálculo de valores solicitados  #sng de satisfaccion general total\_respuestas = len(df) satisfaccion = df[(df['satisfeccion\_general'] >= 6) & (df['satisfeccion\_general'] <= 7)].shape[0] insatisfaccion = df[(df['satisfeccion\_general'] >= 1) & (df['satisfeccion\_general'] <= 4)].shape[0] sng = round((satisfaccion \* 100) / total\_respuestas) - round((insatisfaccion \* 100) / total\_respuestas)  #Total de personas que conocian la empresa total\_conocia\_empresa = df[df['conocia\_empresa'] == 'si'].shape[0]  #sng de recomendacion satisfaccion\_recomendacion = df[(df['recomendacion'] >= 6) & (df['recomendacion'] <= 7)].shape[0] insatisfaccion\_recomendacion = df[(df['recomendacion'] >= 1) & (df['recomendacion'] <= 4)].shape[0] sng\_recomendacion = round((satisfaccion\_recomendacion \* 100) / total\_respuestas) - round((insatisfaccion\_recomendacion \* 100) / total\_respuestas)  #nota promedio de recomendacion nota\_promedio\_recomendacion = df['recomendacion'].mean()  #Total de personas que hicieron un comentario total\_hicieron\_comentario = df[df['recomendacion\_abierta'].notnull() & (df['recomendacion\_abierta'] != '')].shape[0]  #Dias y meses que llevo la encuesta fecha\_inicio = pd.to\_datetime(df['fecha']).min() fecha\_fin = pd.to\_datetime(df['fecha']).max()    dias\_encuesta = (fecha\_fin - fecha\_inicio).days meses\_encuesta = dias\_encuesta // 30    #################################################################################### # Análisis de sentimiento con Groq  ####################################################################################  comentarios = df['recomendacion\_abierta'].dropna().tolist()   # Clasificación de los comentarios clasificaciones = clasificar\_comentarios(comentarios)   resultados = contar\_clasificaciones(clasificaciones)  # Obtención de la conclusión y análisis de problemas principales conclusion = generar\_conclusion(comentarios)  print("\nConclusión y análisis de problemas principales:") print(conclusion)      #################################################################################### # Creación de gráficos de  #################################################################################### plt.figure(figsize=(10, 5)) ax = sns.histplot(df['satisfeccion\_general'], bins=np.arange(0.5, 8.5, 1), kde=**False**, color='skyblue', edgecolor='black')   # Añadir título y etiquetas plt.title('Distribución de Satisfacción General', fontsize=16, fontweight='bold') plt.xlabel('Satisfacción General', fontsize=14) plt.ylabel('Frecuencia', fontsize=14)  # Ajustar los límites del eje x plt.xlim(0.5, 7.5) plt.xticks(range(1, 8))  plt.savefig('satisfeccion\_general.png')    plt.figure(figsize=(10, 5)) labels = list(resultados.keys()) counts = list(resultados.values()) plt.bar(labels, counts, color=['green', 'red', 'blue']) **for** i, count **in** enumerate(counts):  plt.text(i, count + 0.1, str(count), ha='center', va='bottom') plt.title('Distribución de satisfacción en funcion de los comentarios') plt.ylabel('Frecuencia') plt.savefig('sentimientos.png')    # Cierre de la conexión cursor.close() conn.close() |
| --- |

resolver

