```python
import numpy as np
import plotly.express as px
import matplotlib.pyplot as plt
import seaborn as sns
```

Python

```python
import pandas as pd
df = pd.read_csv('https://raw.githubusercontent.com/Gabyy14-nt/PROYECTO_CDD/refs/heads/main/Base_limpia_proyecto.csv')
df
```

Python

```python
df.drop('Unnamed: 0', axis=1, inplace=True)
```

Python

```python
df.describe()
```

Python

	Edad	Horas_de_uso _de_la_tecnología	Horas_de_uso_de_las_redes_sociales	Horas_de_juego	Horas_de_tiempo_en_pantalla	Horas_de_sueño	Horas_de_actividad_física
count	8791.000000	8791.000000	8791.000000	8791.000000	8791.000000	8791.000000	8791.000000
mean	41.448868	6.438605	3.971663	2.529769	7.972097	6.492429	5.001663
std	13.805403	3.133297	2.295750	1.420544	3.968671	1.439642	2.880941
min	18.000000	1.000000	0.000000	0.000000	1.000000	4.000000	0.000000
25%	29.000000	3.780000	2.010000	1.335000	4.640000	5.270000	2.520000
50%	41.000000	6.444082	3.973487	2.529028	7.972650	6.493213	5.004032
75%	53.000000	9.080000	5.970000	3.760000	11.300000	7.730000	7.515000
max	65.000000	12.000000	8.000000	5.000000	15.000000	9.000000	10.000000

```python
df.info()
```

Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8791 entries, 0 to 8790
Data columns (total 12 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Edad                            8791 non-null   int64
 1   Género                          8791 non-null   object
 2   Horas_de_uso _de_la_tecnología  8791 non-null   float64
 3   Horas_de_uso_de_las_redes_sociales  8791 non-null   float64
 4   Horas_de_juego                  8791 non-null   float64
 5   Horas_de_tiempo_en_pantalla     8791 non-null   float64
 6   Estado_de_salud_mental          8791 non-null   object
 7   Nivel_de_estrés                 8791 non-null   object
 8   Horas_de_sueño                  8791 non-null   float64
 9   Horas_de_actividad_física       8791 non-null   float64
 10  Acceso_a_sistemas_de_soporte    8791 non-null   object
 11  Impacto_en_el_entorno_laboral   8791 non-null   object
dtypes: float64(6), int64(1), object(5)
memory usage: 824.3+ KB
```

```python
medied= df['Edad'].median()
mediaf= df['Horas_de_actividad_física'].median()
mediurs= df['Horas_de_uso_de_las_redes_sociales'].median()
meditp= df['Horas_de_tiempo_en_pantalla'].median()
medisu= df['Horas_de_sueño'].median()
mediju= df['Horas_de_juego'].median()
mediju= df['Horas_de_uso _de_la_tecnología'].median()
print(f"La mediana de edad es:", medied)
print(f"La mediana de horas de uso de redes sociales es:", mediurs)
print(f"La mediana de horas de actividad física es:", mediaf)
print(f"La mediana de horas de tiempo en pantalla es:", meditp)
print(f"La mediana de horas de sueño es:", medisu)
print(f"La mediana de horas de juego es:", mediju)
print(f"La mediana de horas de uso de la tecnología es:", mediju)
```

Python

```
La mediana de edad es: 41.0
La mediana de horas de uso de redes sociales es: 3.973487447461093
La mediana de horas de actividad física es: 5.004031782065835
La mediana de horas de tiempo en pantalla es: 7.97265033666125
La mediana de horas de sueño es: 6.493213189312109
La mediana de horas de juego es: 6.444081563103487
La mediana de horas de uso de la tecnología es: 6.444081563103487
```

```python
#Gráfico de pastel
fig = px.pie(df, names='Género', title='Gráfico de Pastel. Género')
fig.show()
fig = px.pie(df, names='Estado_de_salud_mental', title='Gráfico de Pastel. Estado de salud mental')
fig.show()
fig = px.pie(df, names='Nivel_de_estrés', title='Gráfico de Pastel. Nivel de estrés')
fig.show()
fig = px.pie(df, names='Impacto_en_el_entorno_laboral', title='Gráfico de Pastel. Impacto en el Entorno Laboral ')
fig.show()
fig = px.pie(df, names='Acceso_a_sistemas_de_soporte', title='Gráfico de Pastel. Acceso a Sistemas de Soporte ')
fig.show()
```

Python

```python
lista2= ['Género', 'Estado_de_salud_mental','Nivel_de_estrés', 'Acceso_a_sistemas_de_soporte', 'Impacto_en_el_entorno_laboral']
```

Python

```python
for i in lista2:
    plt.figure(figsize=(8, 4))

    # Ordenar las edades antes de crear el gráfico de barras
    df[i].value_counts().sort_index().plot(kind='bar', color='Crimson')

    # Personalización del gráfico
    plt.title('Gráfico de Barras')
    plt.xlabel(f'{i}')
    plt.ylabel('Frecuencia')

    # Rotar las etiquetas del eje X
    plt.xticks(rotation=45, ha='right')

    # Añadir rejilla
    plt.grid(True, which='both', axis='y', linestyle='--', linewidth=0.7, alpha=0.7)

    # Mostrar el gráfico
    plt.show()
```

Python

```python
plt.figure(figsize=(15, 6))
df['Edad'].value_counts().plot(kind='bar', color='skyblue')
plt.title('Gráfico de Barras')
plt.xlabel('Edad')
plt.ylabel('Frecuencia')
# Rotar las etiquetas del eje X si es necesario
plt.xticks(rotation=45, ha='right')
# Añadir rejilla opcional
plt.grid(True, which='both', axis='y', linestyle='--', linewidth=0.7, alpha=0.7)
# Mostrar el gráfico
plt.show()
```

Python

```python
# Contar la cantidad de clientes por género
gender_counts = df['Género'].value_counts()
# Gráfico de pastel
plt.figure(figsize=(6,6))
plt.pie(gender_counts, labels=gender_counts.index, autopct='%1.1f%%', startangle=90,colors=['skyblue', 'lightgreen', 'pink'])
plt.title('Distribución por Género')
plt.show()
```

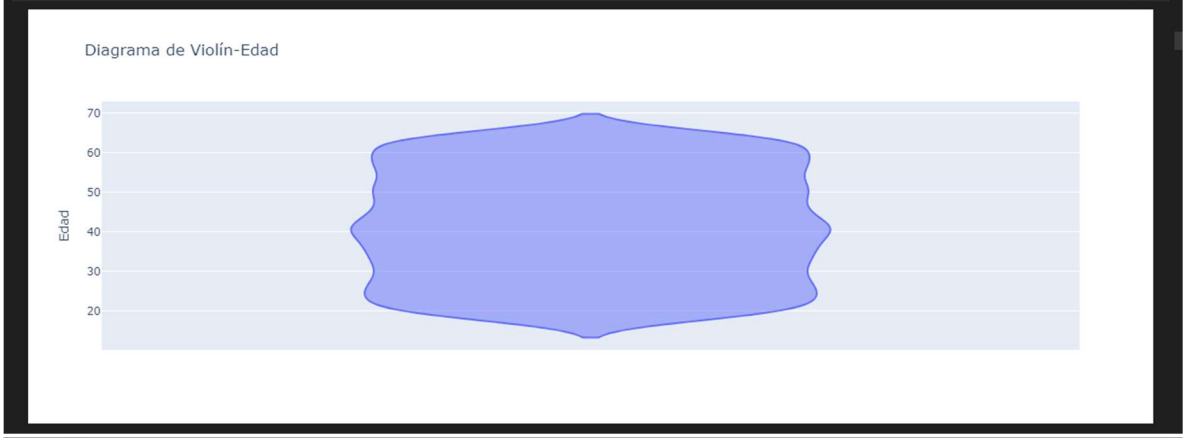
Python

```python
# Gráfico de violín para edad
sns.violinplot(x='Edad', data=df)
# Mostrar gráfico
plt.title('Distribución de Edad')
plt.show()
```

Python

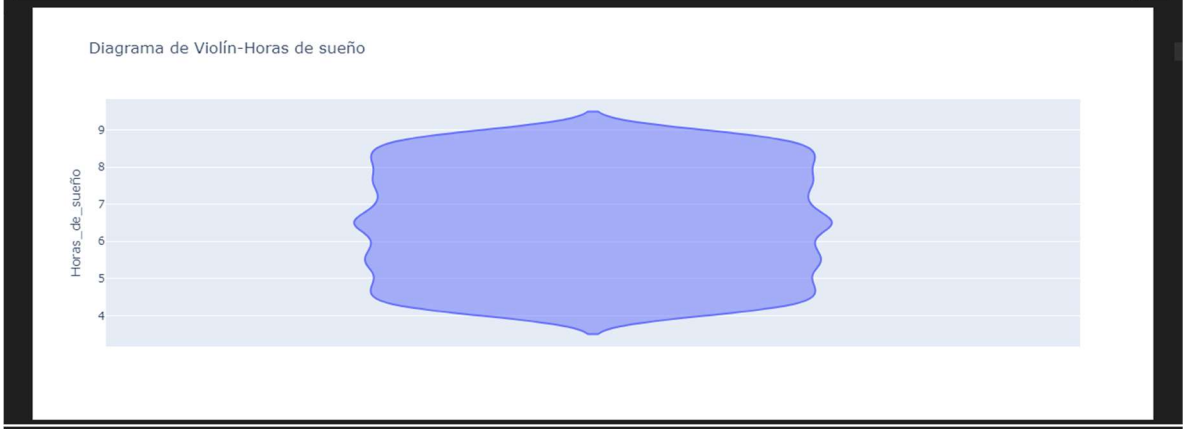
```python
#Violin
fig = px.violin(df, y='Edad', title='Diagrama de Violín-Edad')
fig.show()
```

Python




```python
#Violin
fig = px.violin(df, y='Horas_de_sueño', title='Diagrama de Violín-Horas de sueño')
fig.show()
```

Python




```python
# Crear el histograma
plt.figure(figsize=(4, 3))  # Define el tamaño del gráfico
plt.hist(df['Horas_de_uso_de_las_redes_sociales'], bins=40, color='blue', edgecolor='black');  # Datos y configuración del histograma
#Personalizar
plt.title('Histograma de las Horas_de_uso_de_las_redes_sociales', fontsize=20)  # Título del gráfico
plt.xlabel('Horas', fontsize=15)  # Etiqueta del eje x
plt.ylabel('Frecuencia', fontsize=20)  # Etiqueta del eje y
plt.show()
```

Python

```python
# Crear el histograma
plt.figure(figsize=(4, 3))  # Define el tamaño del gráfico
plt.hist(df['Horas_de_actividad_física'], bins=40, color='pink', edgecolor='black');  # Datos y configuración del histograma
#Personalizar
plt.title('Histograma de las Horas_de_actividad_física', fontsize=20)  # Título del gráfico
plt.xlabel('Horas', fontsize=15)  # Etiqueta del eje x
plt.ylabel('Frecuencia', fontsize=20)  # Etiqueta del eje y
plt.show()
```

Python

```python
# Crear el histograma
plt.figure(figsize=(4, 3))  # Define el tamaño del gráfico
plt.hist(df['Horas_de_juego'], bins=40, color='Aquamarine', edgecolor='black');  # Datos y configuración del histograma
#Personalizar
plt.title('Histograma de las Horas_de_juego', fontsize=20)  # Título del gráfico
plt.xlabel('Horas', fontsize=15)  # Etiqueta del eje x
plt.ylabel('Frecuencia', fontsize=20)  # Etiqueta del eje y
plt.show()
```

Python

```python
# Crear el histograma
plt.figure(figsize=(4, 3))  # Define el tamaño del gráfico
plt.hist(df['Horas_de_tiempo_en_pantalla'], bins=40, color='Coral', edgecolor='black');  # Datos y configuración del histograma
#Personalizar
plt.title('Histograma de las Horas_de_tiempo_en_pantalla', fontsize=20)  # Título del gráfico
plt.xlabel('Horas', fontsize=15)  # Etiqueta del eje x
plt.ylabel('Frecuencia', fontsize=20)  # Etiqueta del eje y
plt.show()
```

Python

```python
# Crear el boxplot
plt.figure(figsize=(10, 3))
plt.boxplot(df['Horas_de_uso _de_la_tecnología'], patch_artist=True, notch=True, boxprops=dict(facecolor='lightblue', color='blue'), vert=False)
# Personalizar el gráfico
plt.title('Boxplot de horas de uso de la tecnología', fontsize=16)
plt.ylabel('Valores', fontsize=12)
# Mostrar el gráfico
plt.show()
```

Python

```python
# Crear el boxplot
plt.figure(figsize=(20, 5))
plt.boxplot(df['Horas_de_tiempo_en_pantalla'], patch_artist=True, notch=True, boxprops=dict(facecolor='lightblue', color='blue'), vert=False)
# Personalizar el gráfico
plt.title('Boxplot de horas de tiempo en pantalla', fontsize=20)
plt.ylabel('Valores', fontsize=30)
# Mostrar el gráfico
plt.show()
```

Python

```python
df2['Género'] = df['Género'].map({'Hombre': 1, 'Mujer': 2, 'Otro': 3})
df2['Estado_de_salud_mental'] = df['Estado_de_salud_mental'].map({'Buena': 2, 'Mala': 3, 'Neutral': 4, 'Excelente': 1})
df2['Nivel_de_estrés'] = df['Nivel_de_estrés'].map({'Bajo': 1, 'Alto': 3, 'Medio': 2})
df2['Impacto_en_el_entorno_laboral'] = df['Impacto_en_el_entorno_laboral'].map({'Negativo': 3, 'Positivo': 1, 'Neutral': 2})
df2['Acceso_a_sistemas_de_soporte'] = df['Acceso_a_sistemas_de_soporte'].map({'Si': 1, 'No': 2})
df2
```

Python

```python
# Seleccionar solo las columnas numéricas
df_numeric = df2[['Nivel_de_estrés', 'Impacto_en_el_entorno_laboral', 'Género', 'Edad', 'Horas_de_uso _de_la_tecnología','Horas_de_uso_de_las_redes_sociales',
# Calcular la matriz de correlación
correlation_matrix = df_numeric.corr()
correlation_matrix
```

Python

```python
plt.figure(figsize=(10,10))
# Gráfico de calor con rango limitado
sns.heatmap(correlation_matrix, annot=True, cmap='YlGnBu', vmin=-0.1, vmax=0.1)
# Mostrar gráfico
plt.show()
```

Python

```python
# Pairplot para visualizar la relación entre las variables numéricas
sns.pairplot(df_numeric)
# Mostrar gráfico
plt.title('Pairplot de Variables Numéricas')
plt.show()
```

```python
# Crear un boxplot para cada columna
for col in df.columns:
    plt.figure(figsize=(8, 4))
    sns.boxplot(x=df[col])
    plt.title(f"Boxplot de {col}")
    plt.show()
```

```python
# Calcular Q1 (primer cuartil) y Q3 (tercer cuartil)
Q1 = df2.quantile(0.25)
Q3 = df2.quantile(0.75)
IQR = Q3 - Q1  # Rango intercuartílico
# Definir límites para detectar outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
# Identificar outliers
outliers = (df < lower_bound) | (df > upper_bound)
print("Outliers detectados:")
print(outliers)
```

```python
# Total de outliers en el DataFrame
outliers_count = outliers.sum()
total_outliers = outliers_count.sum()
print(f"Total de outliers en el DataFrame: {total_outliers}")
```

Python

```
Total de outliers en el DataFrame: 0
```

```python
missing_data = df.isnull().mean()
# Crear un mapa de calor para visualizar la proporción de valores faltantes
plt.figure(figsize=(8, 6))
sns.heatmap(df.isnull(), cbar=False, cmap='viridis', yticklabels=False, xticklabels=df.columns)

# Agregar un título
plt.title('Mapa de Calor de Datos Faltantes', fontsize=16)

# Mostrar el mapa de calor
plt.show()

# Mostrar la proporción de valores faltantes por columna
print("Proporción de valores faltantes por columna:")
print(missing_data)
```

```python
# Configurar el estilo de los gráficos
sns.set(style="whitegrid")

# Crear un gráfico de boxplot
plt.figure(figsize=(8, 6))
sns.boxplot(x='Impacto_en_el_entorno_laboral', y='Nivel_de_estrés', data=df, palette='Set2')
plt.title('Comparación con Boxplot', fontsize=16)
plt.show()

# Crear un gráfico de violín
plt.figure(figsize=(8, 6))
sns.violinplot(x='Impacto_en_el_entorno_laboral', y='Nivel_de_estrés', data=df, palette='Set3')
plt.title('Comparación con Violinplot', fontsize=16)
plt.show()
```

Python

```python
# Verificar valores únicos en la columna Impacto_en_el_entorno_laboral
impacto_unicos = df["Impacto_en_el_entorno_laboral"].value_counts()

# Resumen general agrupado por Impacto_en_el_entorno_laboral
resumen_por_impacto = df.groupby("Impacto_en_el_entorno_laboral").mean(numeric_only=True)

# Mostrar los valores únicos y el resumen general
impacto_unicos, resumen_por_impacto
```

```python
# Cargar la base de datos
file_path = "Base_limpia_proyecto.csv"  # Asegúrate de que el archivo esté en el mismo directorio que este script
data = pd.read_csv(file_path)

# Configuración de estilo para las gráficas
sns.set(style="whitegrid")

# Gráfico de barras para la distribución de Impacto_en_el_entorno_laboral
plt.figure(figsize=(8, 5))
sns.countplot(data=data, x="Impacto_en_el_entorno_laboral", palette="viridis")
plt.title("Distribución de Impacto en el Entorno Laboral", fontsize=14)
plt.xlabel("Impacto en el Entorno Laboral", fontsize=12)
plt.ylabel("Número de Registros", fontsize=12)
plt.show()

# Diagramas de caja para comparar horas de sueño y actividad física entre las categorías
fig, axes = plt.subplots(1, 2, figsize=(16, 6))

# Horas de sueño
sns.boxplot(data=data, x="Impacto_en_el_entorno_laboral", y="Horas_de_sueño", palette="viridis", ax=axes[0])
axes[0].set_title("Horas de Sueño por Impacto en el Entorno Laboral", fontsize=14)
axes[0].set_xlabel("Impacto en el Entorno Laboral", fontsize=12)
axes[0].set_ylabel("Horas de Sueño", fontsize=12)
```

```python
# Horas de actividad física
sns.boxplot(data=data, x="Impacto_en_el_entorno_laboral", y="Horas_de_actividad_física", palette="viridis", ax=axes[1])
axes[1].set_title("Horas de Actividad Física por Impacto en el Entorno Laboral", fontsize=14)
axes[1].set_xlabel("Impacto en el Entorno Laboral", fontsize=12)
axes[1].set_ylabel("Horas de Actividad Física", fontsize=12)

plt.tight_layout()
plt.show()

# Análisis numérico detallado
resumen_impacto = data.groupby("Impacto_en_el_entorno_laboral").agg({
    "Horas_de_uso _de_la_tecnología": ["mean", "std"],
    "Horas_de_uso_de_las_redes_sociales": ["mean", "std"],
    "Horas_de_juego": ["mean", "std"],
    "Horas_de_tiempo_en_pantalla": ["mean", "std"],
    "Horas_de_sueño": ["mean", "std"],
    "Horas_de_actividad_física": ["mean", "std"]
})

print("Resumen numérico detallado:")
print(resumen_impacto)
```

```python
# Columna específica para comparar
columna_interes = "Impacto_en_el_entorno_laboral"  # Cambia esto por la columna que deseas analizar
# Calcular la matriz de correlación
correlacion = df_numeric.corr()
# Filtrar solo la fila de la columna de interés
correlacion_interes = correlacion[[columna_interes]].sort_values(by=columna_interes, ascending=False)
# Crear el heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlacion_interes, annot=True, fmt=".5f", cmap="viridis", cbar=True)
plt.title(f"Correlación entre {columna_interes} y otras columnas", fontsize=16)
plt.ylabel("Columnas", fontsize=12)
plt.xlabel("Correlación", fontsize=12)
plt.tight_layout()
plt.show()
```

```python
# Columna de referencia
data=df2
columna_referencia = "Impacto_en_el_entorno_laboral"  # Cambia según la columna que desees usar

# Asegurarte de que la columna de referencia sea categórica
if data[columna_referencia].dtype != "object":
    data[columna_referencia] = data[columna_referencia].astype("category")

# Seleccionar solo las columnas numéricas
df_numeric = data.select_dtypes(include=['float64', 'int64']).columns

# Crear gráficos de boxplot para comparar cada columna numérica con la columna de referencia
fig, axes = plt.subplots(len(df_numeric), 1, figsize=(10, 5 * len(df_numeric)))

for i, col in enumerate(df_numeric):
    sns.boxplot(data=data, x=columna_referencia, y=col, palette="viridis", ax=axes[i])
    axes[i].set_title(f"Distribución de {col} según {columna_referencia}", fontsize=14)
    axes[i].set_xlabel(columna_referencia, fontsize=12)
    axes[i].set_ylabel(col, fontsize=12)

plt.tight_layout()
plt.show()
```

Python

```python
conteo = df.groupby(['Impacto_en_el_entorno_laboral', 'Nivel_de_estrés']).size().reset_index(name='Frecuencia')

# Crear el gráfico con seaborn
plt.figure(figsize=(8, 6))
sns.barplot(data=conteo, x='Impacto_en_el_entorno_laboral', y='Frecuencia', hue='Nivel_de_estrés', palette='viridis')

# Personalizar el gráfico
plt.title('Comparación de niveles de estrés, impacto en el entorno laboral')
plt.xlabel('Impacto_en_el_entorno_laboral')
plt.ylabel('Frecuencia')
plt.legend(title='Nivel_de_estrés')
plt.tight_layout()

# Mostrar el gráfico
plt.show()
```

✓ 0.7s Python

```python
conteo = df.groupby(['Impacto_en_el_entorno_laboral', 'Horas_de_sueño']).size().reset_index(name='Frecuencia')

# Crear el gráfico con seaborn
plt.figure(figsize=(8, 6))
sns.barplot(data=conteo, x='Impacto_en_el_entorno_laboral', y='Frecuencia', hue='Horas_de_sueño', palette='viridis')

# Personalizar el gráfico
plt.title('Comparación de niveles de estrés, horas de sueño')
plt.xlabel('Impacto_en_el_entorno_laboral')
plt.ylabel('Frecuencia')
plt.legend(title='Horas_de_sueño_de_estrés')
plt.tight_layout()

# Mostrar el gráfico
plt.show()
```

✓ 13.8s Python

Dashboard

```
pip install statsmodels
```
✓ 4.0s Python

```
import statsmodels.formula.api as smf
```
✓ 3.0s Python

```
import dash
from dash import dcc, html
from dash.dependencies import Input, Output
import dash_bootstrap_components as dbc
import plotly.express as px
```
✓ 1.9s Python

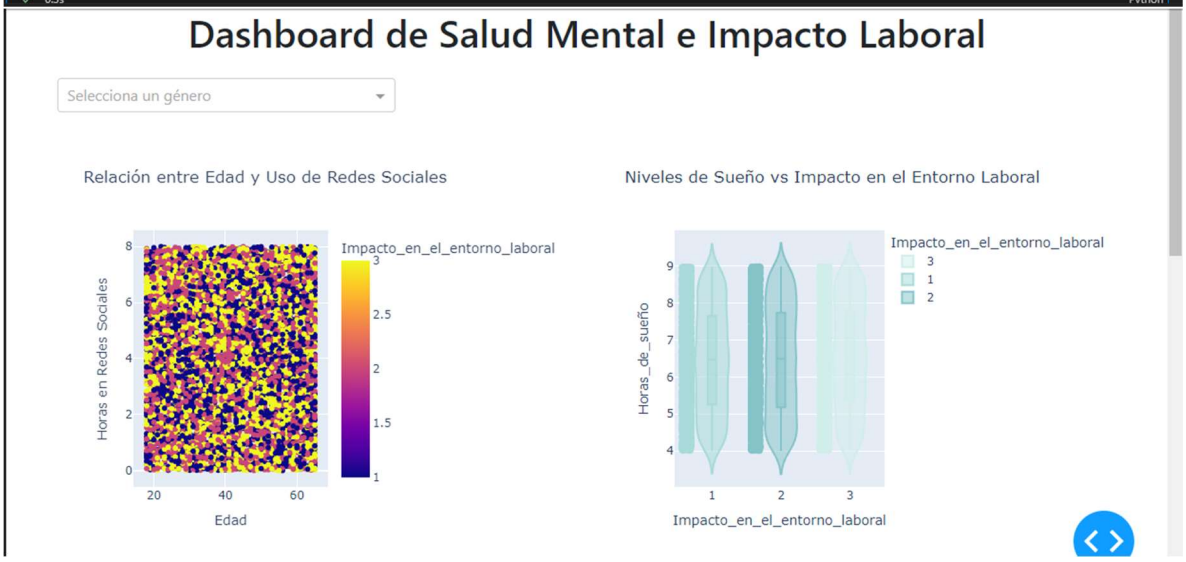
```
import pandas as pd
df = pd.read_csv('https://raw.githubusercontent.com/Gabyy14-nt/PROYECTO_CDD/refs/heads/main/Base_limpia_proyecto.csv')
```
✓ 1m 24.5s Python

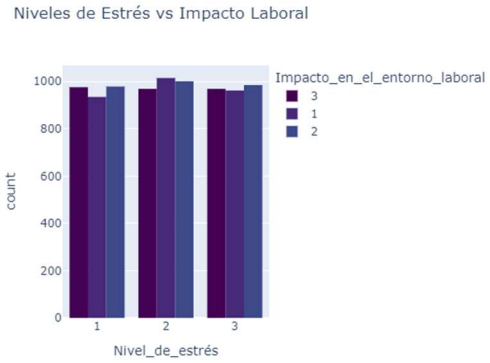
```
df2 = df
df2['Género'] = df['Género'].map({'Hombre': 1, 'Mujer': 2, 'Otro': 3})
df2['Estado_de_salud_mental'] = df['Estado_de_salud_mental'].map({'Buena': 2, 'Mala': 3, 'Neutral': 4, 'Excelente': 1})
df2['Nivel_de_estrés'] = df['Nivel_de_estrés'].map({'Bajo': 1, 'Alto': 3, 'Medio': 2})
df2['Impacto_en_el_entorno_laboral'] = df['Impacto_en_el_entorno_laboral'].map({'Negativo': 3, 'Positivo': 1, 'Neutral': 2})
df2['Acceso_a_sistemas_de_soporte'] = df['Acceso_a_sistemas_de_soporte'].map({'Si': 1, 'No': 2})
df2
```
✓ 0.0s Python

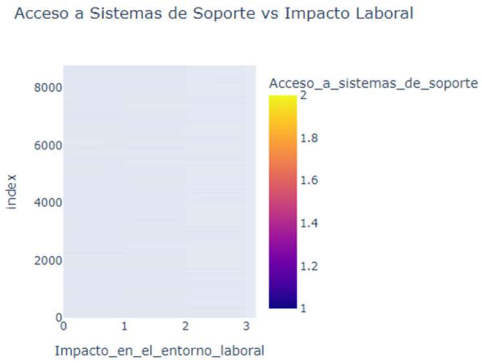
```
df2 = df2.dropna(subset=['Género', 'Impacto_en_el_entorno_laboral', 'Nivel_de_estrés', 'Horas_de_sueño'])

# Inicializar la aplicación Dash
app = dash.Dash(__name__, external_stylesheets=[dbc.themes.BOOTSTRAP])

# Layout de la app
app.layout = dbc.Container([
    html.H1("Dashboard de Salud Mental e Impacto Laboral", className="text-center my-4"),

    # Filtros
    dbc.Row([
        dbc.Col([
            dcc.Dropdown(
                id='gender-filter',
                options=[
                    {'label': 'Hombre', 'value': 1},
                    {'label': 'Mujer', 'value': 2},
                    {'label': 'Otro', 'value': 3}
                ],
                placeholder="Selecciona un género",
            )
        ], width=4),
    ], className="mb-4"),

    # Gráficas
    dbc.Row([
        dbc.Col(dcc.Graph(id='age-social-media-chart'), width=6),
        dbc.Col(dcc.Graph(id='sleep-impact-chart'), width=6)
    ]),
    dbc.Row([
        dbc.Col(dcc.Graph(id='stress-impact-chart'), width=6),
        dbc.Col(dcc.Graph(id='support-impact-chart'), width=6)
    ]),
    dbc.Row([
        dbc.Col(dcc.Graph(id='age-stress-chart'), width=12),
    ])
])

# Callbacks
@app.callback(
    [Output('age-social-media-chart', 'figure'),
    Output('sleep-impact-chart', 'figure'),
    Output('stress-impact-chart', 'figure'),
    Output('support-impact-chart', 'figure'),
    Output('age-stress-chart', 'figure')],
    [Input('gender-filter', 'value')]
)
def update_charts(selected_gender):
    # Filtrar por género si está seleccionado
    dff = df2[df2['Género'] == selected_gender] if selected_gender else df2

    # Gráfico 1: Relación entre Edad y Uso de Redes Sociales
    age_social_media_chart = px.scatter(
        dff,
        x='Edad',
        y='Horas_de_uso_de_las_redes_sociales',
        color='Impacto_en_el_entorno_laboral',
        title="Relación entre Edad y Uso de Redes Sociales",
        labels={'Horas de uso de las redes sociales': 'Horas en Redes Sociales'},
```

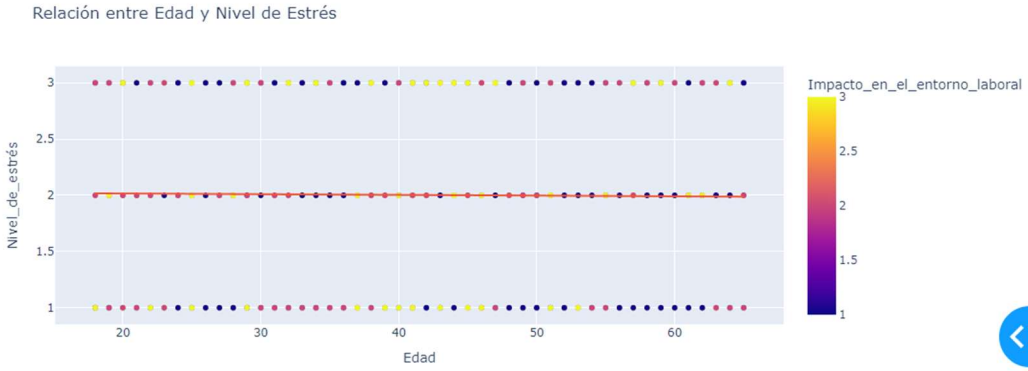
```
        color_discrete_sequence=px.colors.sequential.Plasma
    )

    # Gráfico 2: Niveles de sueño vs Impacto laboral
    sleep_impact_chart = px.violin(
        dff,
        x='Impacto_en_el_entorno_laboral',
        y='Horas_de_sueño',
        color='Impacto_en_el_entorno_laboral',
        box=True,
        points="all",
        title="Niveles de Sueño vs Impacto en el Entorno Laboral",
        color_discrete_sequence=px.colors.sequential.Teal
    )

    # Gráfico 3: Niveles de estrés vs Impacto laboral
    stress_impact_chart = px.histogram(
        dff,
        x='Nivel_de_estrés',
        color='Impacto_en_el_entorno_laboral',
        barmode='group',
        title="Niveles de Estrés vs Impacto Laboral",
        color_discrete_sequence=px.colors.sequential.Viridis
    )

    # Gráfico 4: Acceso a sistemas de soporte vs Impacto laboral
    support_impact_chart = px.bar(
        dff,
        x='Impacto_en_el_entorno_laboral',
        color='Acceso_a_sistemas_de_soporte',
        barmode='group',
        title="Acceso a Sistemas de Soporte vs Impacto Laboral",
        color_discrete_sequence=px.colors.qualitative.Pastel
    )

    # Gráfico 5: Relación entre Edad y Nivel de Estrés
    age_stress_chart = px.scatter(
        dff,
        x='Edad',
        y='Nivel_de_estrés',
        trendline="ols",
        color='Impacto_en_el_entorno_laboral',
        title="Relación entre Edad y Nivel de Estrés",
        color_discrete_sequence=px.colors.sequential.Magma
    )

    return age_social_media_chart, sleep_impact_chart, stress_impact_chart, support_impact_chart, age_stress_chart

    if df.empty:
        return px.scatter(title="No hay datos disponibles"), px.violin(), px.histogram(), px.bar(), px.scatter()

# Ejecutar la aplicación
if __name__ == '__main__':
    app.run(debug=True)
```

0.3s Python




Niveles de Estrés vs Impacto Laboral
Impacto_en_el_entorno_laboral
3
1
2
count
1000
800
600
400
200
0
1
2
3
Nivel_de_estrés


Acceso a Sistemas de Soporte vs Impacto Laboral
Acceso_a_sistemas_de_soporte
2
1.8
1.6
1.4
1.2
1
index
8000
6000
4000
2000
0
0
1
2
3
Impacto_en_el_entorno_laboral


Relación entre Edad y Nivel de Estrés
Impacto_en_el_entorno_laboral
3
2.5
2
1.5
1
Nivel_de_estrés
3
2.5
2
1.5
1
20
30
40
50
60
Edad