

Prompt: ¿puedes hacer el cuadro del Sr Francisco en rojo claro usando la sintaxis de estilo en la parte inferior?

ChatGPT responde:



Por supuesto, puedo añadir estilo al cuadro del Sr. Francisco en el organigrama para que se muestre en rojo claro. Aquí tienes el código Mermaid JS actualizado con el estilo aplicado:

```
mermaid
graph TD
    Francisco[Presidente: Sr. Francisco] -->|Directamente bajo| Patricia[Secretaria: Sra. Patricia]
    Francisco -->|Directamente bajo| Fidel[Vicepresidente: Sr. Fidel]
    Fidel -->|Directamente bajo| Finanzas[Gerente de Finanzas]
    Fidel -->|Directamente bajo| TI[Gerente de TI]
    Fidel -->|Directamente bajo| RRHH[Gerente de Recursos Humanos]

    style Francisco fill:#FFCCCC,stroke:#333,stroke-width:2px
```

En este código, la línea **style Francisco fill:#FFCCCC,stroke:#333,stroke-width:2px** aplica un relleno de color rojo claro (**#FFCCCC**) al cuadro del Sr. Francisco, con un borde (**stroke**) de color gris oscuro (**#333**) y un ancho de borde (**stroke-width**) de 2 píxeles.

Al igual que antes, necesitarás un editor o visualizador que soporte Mermaid JS para ver el organigrama con el estilo aplicado.

El código generado se ve así:

```
graph TD
    Francisco [Presidente: Sr. Francisco]
    Fidel [Vicepresidente: Sr. Fidel]
    Patricia [Secretaria: Sra. Patricia]
    Finanzas [Gerente de Finanzas]
    IT [Gerente de TI]
    RH [Gerente de Recursos Humanos]
```

```
Francisco -> Fidel
Francisco -> Patricia
Fidel -> Finanzas
Fidel -> IT
Fidel -> RH
Style Francisco fill:#f9d0c4
```

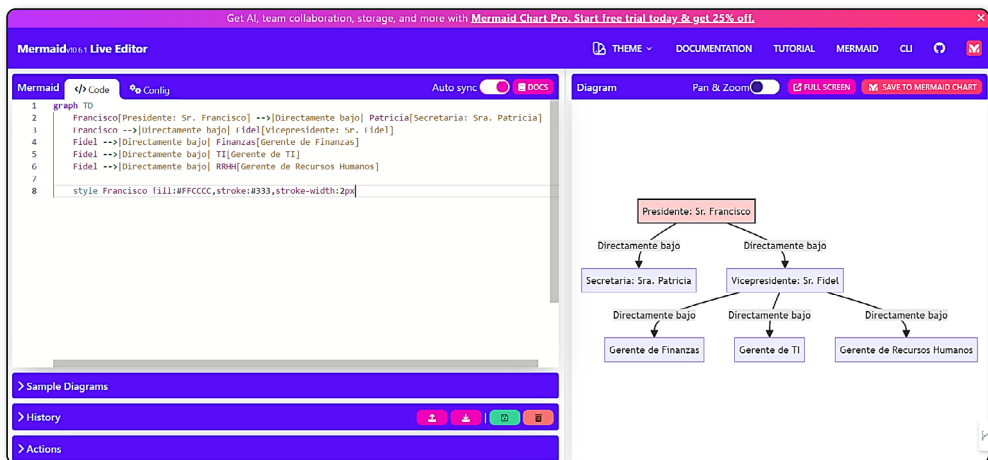
Observó que la nueva fila se genera en la parte inferior y puede cambiarla según el color que desee si no se ajusta a sus necesidades.

```
Estilo A relleno: coral claro;
```

Puede continuar solicitando ChatGPT para otras opciones de estilo, como el color de los bordes, el tamaño del borde y la negrita. Aquí hay algunas indicaciones:

Paso 3: copie el código, luego péguelo en el editor web Mermaid.live

Al pegarlo en el editor (corchete rojo a la izquierda), obtendrá el diagrama generado a la derecha.



Prompt: haz que el borde del Sr Francisco sea más grueso usando la sintaxis de estilo en la parte inferior, hacer el borde negro y sin guiones.

ChatGPT responde:



Claro, para hacer que el borde del cuadro del Sr. Francisco sea más grueso, de color negro y sin guiones en el organigrama de Mermaid JS, puedes actualizar la línea de estilo de la siguiente manera:

3.2 PROMPTS PARA DATA SCIENCE

Prompt (1)

Prompt: en base a la fuente de datos iris escriba el código en Python para realiza una analítica descriptiva

ChatGPT responde con:



Claro, a continuación, te proporciono un ejemplo de cómo podrías realizar un análisis descriptivo del conjunto de datos Iris en Python utilizando la biblioteca **pandas** para la manipulación de datos y la biblioteca **seaborn** para la visualización. Este conjunto de datos contiene mediciones de las características de las flores Iris de tres especies diferentes.

```
python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Cargar el conjunto de datos Iris
df = sns.load_dataset('iris')

# Mostrar las primeras filas del conjunto de datos
print(df.head())
```

```
# Descripción estadística de las características numéricas
print(df.describe())

# Contar el número de instancias para cada especie
print(df['species'].value_counts())

# Visualizaciones para el análisis descriptivo
# Histogramas para las características numéricas
df.hist(edgecolor='black', linewidth=1.2, figsize=(12, 8))
plt.suptitle("Histogramas de las características del Iris")
plt.show()

# Diagrama de cajas para comparar distribuciones
plt.figure(figsize=(12, 8))
plt.suptitle("Diagramas de caja para las características del Iris")
sns.boxplot(data=df)
plt.show()

# Diagramas de pares para relaciones bivariadas
sns.pairplot(df, hue='species')
plt.suptitle("Diagramas de pares del Iris con color por especie")
plt.show()
```

Prompt (2)

Prompt: en base a la siguiente fuente de datos escriba el código en Python para realiza una analítica descriptiva, el separador numérico es el punto.

	A	B	C	D	E	F	G	H
1	Dates	PX_OPEN	PX_LAST	PX_HIGH	PX_LOW	PX_MID	PX_VOLUME	TURNOVER
2	4/01/2010	1.88	1.94	1.95	1.88	1.95	1255212	2429876
3	5/01/2010	1.94	2.01	2.01	1.94	2.01	650367	1286412
4	6/01/2010	2.05	2.05	2.06	2.03	2.05	1899932	3878281
5	7/01/2010	2.05	2.04	2.05	2.01	2.04	946334	1917474
6	8/01/2010	2.04	2.04	2.04	2.01	2.04	1348336	2740482
7	11/01/2010	2.06	2.03	2.09	2.03	2.02	1512945	3130316
8	12/01/2010	2.01	1.98	2.01	1.98	1.99	1717348	3418243
9	13/01/2010	2.01	1.94	2.01	1.93	1.93	15368300	29771230
10	14/01/2010	1.96	1.88	1.96	1.87	1.88	7593286	14351240
11	15/01/2010	1.86	1.89	1.91	1.85	1.89	4686626	8784675
12	18/01/2010	1.91	1.94	1.94	1.91	1.95	3028501	5846874
13	19/01/2010	1.93	1.96	1.97	1.93	1.97	2111209	4135395
14	20/01/2010	1.93	1.93	1.95	1.92	1.93	947025	1826595
15	21/01/2010	1.93	1.91	1.94	1.9	1.91	1932630	3687947
16	22/01/2010	1.88	1.92	1.93	1.86	1.93	1093104	2076711
17	25/01/2010	1.92	1.93	1.94	1.92	1.93	1172298	2264678
18	26/01/2010	1.93	1.94	1.94	1.9	1.94	2005960	3871081
19	27/01/2010	1.93	1.9	1.93	1.89	1.9	880166	1680469
20	28/01/2010	1.9	1.87	1.9	1.86	1.87	2533517	4744905
21	29/01/2010	1.88	1.88	1.9	1.87	1.88	1193730	2254455
22	1/02/2010	1.88	1.9	1.9	1.88	1.9	692833	1303767
23	2/02/2010	1.91	1.92	1.92	1.89	1.92	830959	1588275
24	3/02/2010	1.92	1.91	1.93	1.91	1.9	1291264	2477996
25	4/02/2010	1.89	1.83	1.89	1.81	1.83	1709525	3130878
26	5/02/2010	1.82	1.81	1.82	1.76	1.8	1864849	3343650
27	8/02/2010	1.81	1.83	1.85	1.81	1.83	1950456	3564548
28	9/02/2010	1.83	1.87	1.88	1.83	1.87	4114145	7713212
29	10/02/2010	1.87	1.85	1.87	1.83	1.85	610435	1124757

ChatGPT responde con:



El archivo proporcionado contiene una serie de datos financieros con las siguientes columnas: Fechas (**Dates**), Precio de apertura (**PX_OPEN**), Precio de cierre (**PX_LAST**), Precio máximo (**PX_HIGH**), Precio mínimo (**PX_LOW**), Precio medio (**PX_MID**), Volumen (**PX_VOLUME**) y Rotación (**TURNOVER**). Para realizar un análisis descriptivo, examinaré estadísticas resumidas, como la media, mediana, desviación estándar, valores mínimos y máximos de cada columna numérica. Además, se pueden incluir visualizaciones para ayudar a entender la distribución y las tendencias de los datos.

Voy a proceder con este análisis.

El análisis descriptivo de los datos financieros proporciona las siguientes estadísticas resumidas:

Precio de apertura (PX_OPEN):

- Media: 1.10
- Desviación estandar: 0.70
- Mínimo: 0.17
- Máximo: 2.86

Precio de cierre (px_last):

- Media: 1.10
- Desviación estandar: 0.70
- Mínimo: 0.17
- Máximo: 2.86

Precio máximo (px_high):

- Media: 1.11
- Desviación estandar: 0.71
- Mínimo: 0.17
- Máximo: 2.88

Precio mínimo (px_low):

- Media: 1.09
- Desviación estandar: 0.70
- Mínimo: 0.17
- Máximo: 2.85

Precio medio (px_mid):

- Media: 1.10
- Desviación estandar: 0.70
- Mínimo: 0.16
- Máximo: 2.86

Volúmen (px_volume):

- Media: ~2.67 millones
- Desviación estandar: ~7.60 millones
- Mínimo: 50
- Máximo: ~279.33 millones

Rotación (turnover):

- Media: ~3.24 millones
- Desviación estandar: ~14.04 millones
- Mínimo: 30
- Máximo: ~547.45 millones


```
# Modelo simple de CNN
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(image_width, image_height, 3)),
    MaxPooling2D(2, 2),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(num_classes, activation='softmax')
])

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Generador de imágenes para entrenamiento
train_datagen = ImageDataGenerator(rescale=1./255)
train_generator = train_datagen.flow_from_directory(
    train_directory,
    target_size=(image_width, image_height),
    batch_size=batch_size,
    class_mode='categorical')

# Entrenamiento del modelo
model.fit(train_generator, steps_per_epoch=steps_per_epoch, epochs=epochs)
```

```
python Copy code  
  
import torch  
import cv2  
from PIL import Image  
  
# Cargar el modelo preentrenado  
model = torch.hub.load('ultralytics/yolov5', 'yolov5s', pretrained=True)  
  
# Función para realizar la detección  
def detect_image(image_path):  
    image = Image.open(image_path)  
    results = model(image)  
    results.show() # Para visualizar  
  
# Usar la función en una imagen  
detect_image('path_to_your_image.jpg')
```

Aquí tienes un ejemplo de cómo usar estas expresiones regulares en Python:

python

```
python Copy code  
  
import re  
  
# Regex patterns  
dni_pattern = r'^\d{8}[A-Z]$\'  
telefono_pattern = r'^\d{9}$\  
email_pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$\  
  
# Test strings  
dni = "12345678A"  
telefono = "123456789"  
email = "ejemplo@correo.com"  
  
# Validating the strings  
if re.match(dni_pattern, dni):  
    print("DNI válido.")  
else:  
    print("DNI no válido.")  
  
if re.match(telefono_pattern, telefono):  
    print("Número de teléfono válido.")  
else:  
    print("Número de teléfono no válido.")  
  
if re.match(email_pattern, email):  
    print("Correo electrónico válido.")  
else:  
    print("Correo electrónico no válido.")
```

rotación o inversión.

pythonCopy code

```
import cv2
import numpy as np
import os
from matplotlib import pyplot as plt

# Define a function to preprocess a single image
def preprocess_image(image_path, output_size=(224, 224)):
    # Read the image
    image = cv2.imread(image_path)

    # Convert color from BGR to RGB if you're going to work with matplotlib or other
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    # Resize the image
    image = cv2.resize(image, output_size)

    # Normalize the image
    image = image / 255.0

    return image

# Define a directory where the images are stored
directory = '/mnt/data/'

# List to hold preprocessed images
preprocessed_images = []

# Preprocess all images
for filename in os.listdir(directory):
    if filename.endswith('.jpg'):
        # Preprocess the image
        preprocessed_image = preprocess_image(os.path.join(directory, filename))

        # Add to the list
        preprocessed_images.append(preprocessed_image)

        # Optionally, show the image
        plt.imshow(preprocessed_image)
        plt.show()

# If you want to convert this list to a NumPy array
preprocessed_images = np.array(preprocessed_images)
```

```
python Copycode

import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.optimizers import Adam

# Carga y preprocesamiento de datos
train_datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)
train_generator = train_datagen.flow_from_directory(
    'path_to_images',
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    subset='training')

validation_generator = train_datagen.flow_from_directory(
    'path_to_images',
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    subset='validation')

# Construcción del modelo
model = Sequential([
    ResNet50(include_top=False, weights='imagenet', pooling='avg'),
    Dense(5, activation='softmax') # Asumiendo que tenemos 5 razas de perros
])

model.layers[0].trainable = False # Congela las capas del modelo preentrenado

# Compilación del modelo
model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])

# Entrenamiento del modelo
model.fit(train_generator, validation_data=validation_generator, epochs=10)
```

```
python Copy code

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import re

# Cargar datos
# Asegúrate de tener un archivo CSV con dos columnas: 'texto' y 'etiqueta'
data = pd.read_csv('emails.csv')

# Preprocesamiento de texto
def preprocess(text):
    text = text.lower()
    text = re.sub(r'\W+', ' ', text)
    text = re.sub(r'\s+', ' ', text)
    return text

data['texto'] = data['texto'].apply(preprocess)

# Dividir datos en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(data['texto'], data['etiqueta'],

# Vectorización TF-IDF
vectorizer = TfidfVectorizer(stop_words=stopwords.words('english'))
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)

# Crear y entrenar el modelo
model = MultinomialNB()
model.fit(X_train_tfidf, y_train)

# Predecir y evaluar
y_pred = model.predict(X_test_tfidf)
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```