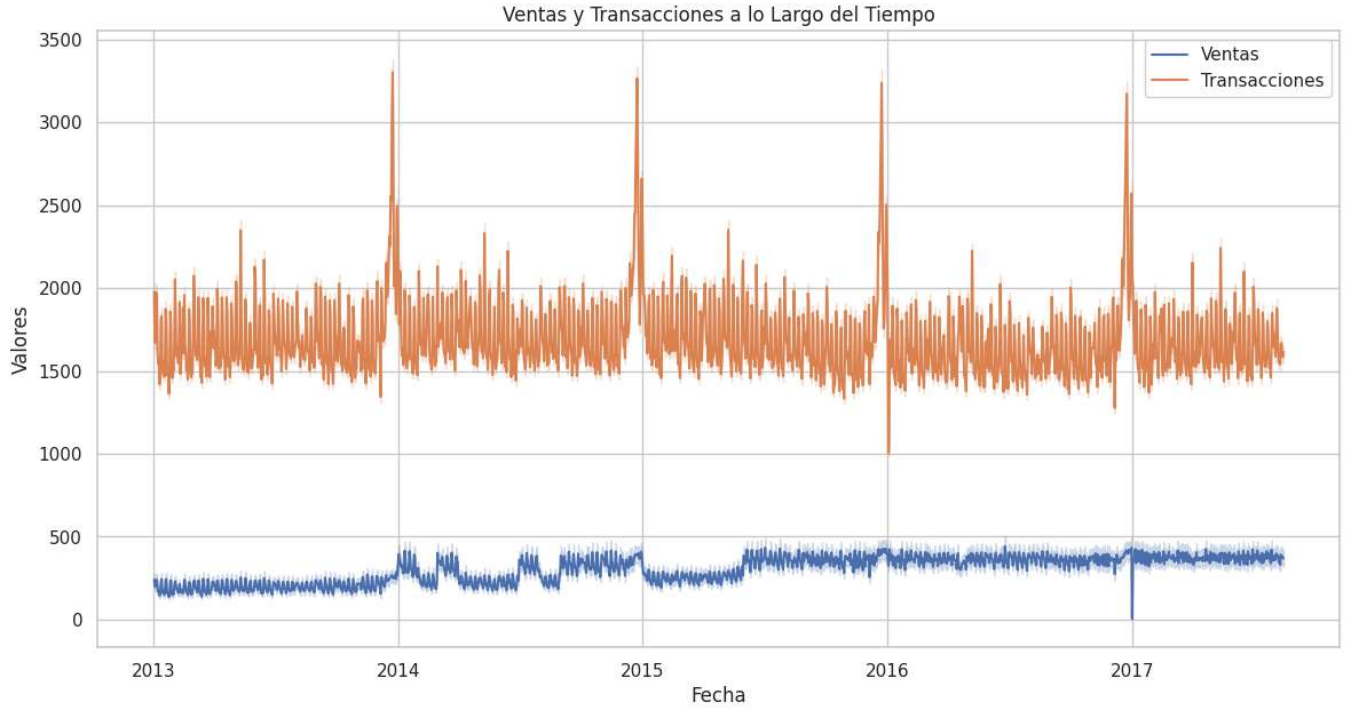
```
# 1. Importar librerías necesarias
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
import warnings

# Supresión de warnings específicos
warnings.filterwarnings("ignore", category=pd.errors.SettingWithCopyWarning)

sns.set(style='whitegrid')
%matplotlib inline

# 2. Cargar y preparar datos
train_df = pd.read_csv('/content/train.csv')
stores = pd.read_csv('/content/stores.csv')
transactions = pd.read_csv('/content/transactions.csv')
holidays = pd.read_csv('/content/holidays_events.csv')
oil = pd.read_csv('/content/oil.csv')

# 3. Convertir fechas y eliminar duplicados
train_df['date'] = pd.to_datetime(train_df['date'], errors='coerce')
transactions['date'] = pd.to_datetime(transactions['date'], errors='coerce')
oil['date'] = pd.to_datetime(oil['date'], errors='coerce')
holidays['date'] = pd.to_datetime(holidays['date'], errors='coerce')
train_df.drop_duplicates(inplace=True)

# 4. Unión de los datos
train_df = train_df.merge(transactions, on=['date', 'store_nbr'], how='left')
train_df['transactions'] = train_df['transactions'].ffill()

train_df = train_df.merge(oil, on='date', how='left')
train_df['dcoilwtico'] = train_df['dcoilwtico'].ffill()

train_df = train_df.merge(stores, on='store_nbr', how='left')

# Renombrar columna `type` en `holidays` para evitar conflictos en la unión
holidays.rename(columns={'type': 'day_nature'}, inplace=True)

# 5. Manejo de outliers en las ventas
for store in train_df['store_nbr'].unique():
    threshold = train_df[train_df['store_nbr'] == store]['sales'].quantile(0.99)
    train_df = train_df.drop(train_df[(train_df['store_nbr'] == store) & (train_df['sales'] > threshold)].index)

# 6. Preparar datos de días festivos y unirlos según la región
holiday_loc = holidays[holidays['locale'] == 'Local'].rename(columns={'locale_name': 'city'})
holiday_reg = holidays[holidays['locale'] == 'Regional'].rename(columns={'locale_name': 'state'})
holiday_nat = holidays[holidays['locale'] == 'National']

# Unir eventos locales y eliminar días festivos no trasladados
train_df = pd.merge(train_df, holiday_loc[['date', 'city', 'day_nature', 'transferred']], on=['date', 'city'], how='left')
train_df = train_df[~((train_df['day_nature'] == 'Holiday') & (train_df['transferred'] == False))]
train_df.drop(['day_nature', 'transferred'], axis=1, inplace=True, errors='ignore')

# Unir eventos regionales y eliminar días festivos no trasladados
train_df = pd.merge(train_df, holiday_reg[['date', 'state', 'day_nature', 'transferred']], on=['date', 'state'], how='left')
train_df = train_df[~((train_df['day_nature'] == 'Holiday') & (train_df['transferred'] == False))]
train_df.drop(['day_nature', 'transferred'], axis=1, inplace=True, errors='ignore')

# Unir eventos nacionales y eliminar días festivos no trasladados
train_df = pd.merge(train_df, holiday_nat[['date', 'day_nature', 'transferred']], on='date', how='left')
train_df = train_df[~((train_df['day_nature'] == 'Holiday') & (train_df['transferred'] == False))]
train_df.drop(['day_nature', 'transferred'], axis=1, inplace=True, errors='ignore')

# 7. Crear nuevas variables de fecha y variables categóricas
train_df['month'] = train_df['date'].dt.month
train_df['day'] = train_df['date'].dt.day
train_df['day_name'] = train_df['date'].dt.day_name()
train_df['year'] = train_df['date'].dt.year

# Crear variables dummies para los días de la semana y las familias de productos
train_df = pd.get_dummies(train_df, columns=['day_name', 'family'])

# 8. Normalización y escalado de variables `sales` y `transactions`
scaler = MinMaxScaler()
train_df[['sales_norm', 'transactions_norm']] = scaler.fit_transform(train_df[['sales', 'transactions']])

# 9. Análisis Exploratorio de Datos (EDA) - Gráfica de ventas y transacciones
plt.figure(figsize=(14, 7))
```

```
sns.lineplot(data=train_df, x='date', y='sales', label='Ventas')
sns.lineplot(data=train_df, x='date', y='transactions', label='Transacciones')
plt.title('Ventas y Transacciones a lo Largo del Tiempo')
plt.xlabel('Fecha')
plt.ylabel('Valores')
plt.legend()
plt.show()

# 10. División en conjuntos de entrenamiento y prueba
# Dividir el dataset en 80% entrenamiento y 20% prueba
train_data, test_data = train_test_split(train_df, test_size=0.2, random_state=42)

# Tamaño del conjunto de entrenamiento y prueba
print(f'Tamaño del conjunto de entrenamiento: {len(train_data)}')
print(f'Tamaño del conjunto de prueba: {len(test_data)}')
```




```
Tamaño del conjunto de entrenamiento: 2318896
Tamaño del conjunto de prueba: 579724
```

1. **Almacenamiento de Datos**
   - **Base de Datos**: Se seleccionaron archivos CSV como fuente de almacenamiento y manejo de datos para esta fase de preparación. En una implementación futura, se recomienda una base de datos SQL para manejar eficientemente la información y asegurar la persistencia y escalabilidad.
2. **Limpieza de los Datos**
   - **Tratamiento de Valores Faltantes**: Se aplicó forward fill (`ffill`) en las variables de transacciones y precios del petróleo para mantener la continuidad en el tiempo y evitar la pérdida de datos valiosos en estas columnas.
   - **Corrección de Errores**: Se eliminó duplicados en el conjunto de datos de ventas y se gestionaron outliers en la columna `sales`, eliminando valores superiores al percentil 99 de ventas por tienda para reducir el impacto de valores atípicos en el modelo. Sin embargo, se detectó una limitación en el manejo de outliers, lo cual ocasionó que en el año 2017 se registraran valores de ventas en cero para algunos registros.
3. **Creación de Nuevas Variables**
   - **Variables Derivadas**: Se añadieron variables de estacionalidad (`month`, `day`, `day_name`, `year`) y se aplicaron variables dummies para los días de la semana y familias de productos. Esto permite captar efectos estacionales y patrones específicos de los productos en el modelo.
   - **Motivación**: La creación de estas variables permite analizar tendencias temporales y comportamientos específicos de productos, contribuyendo a mejorar la precisión del modelo.
4. **Normalización y Escalado**
   - **Proceso de Transformación**: Se aplicó `MinMaxScaler` para normalizar las variables `sales` y `transactions`, lo cual es necesario para estandarizar las variables y mejorar el rendimiento de algoritmos que son sensibles a la escala.

5. **Procesamiento de los Datos (ETL)**
    - **ETL**: Se realizó la carga de datos desde archivos CSV, su transformación mediante la limpieza y el enriquecimiento de variables, y su preparación para modelado.
    - **Limpieza de Datos**: Se gestionaron valores faltantes, duplicados y outliers para mejorar la calidad de los datos de entrada.
6. **Análisis Exploratorio de Datos (EDA)**
    - **Visualización**: Se generaron gráficas de línea para observar las tendencias de ventas y transacciones en el tiempo, permitiendo identificar patrones estacionales y cambios abruptos.
    - **Estadísticas Descriptivas**: Se calcularon métricas descriptivas para entender la distribución de las ventas y transacciones.
7. **División de los Datos**
    - **División en Conjuntos de Entrenamiento y Prueba**: Se realizó la división en un 80% para el entrenamiento y un 20% para pruebas, con el fin de garantizar que el modelo pueda generalizarse y evitar el sobreajuste.

Limitaciones del Estudio

1. **Manejo de Outliers**: La eliminación de outliers en ventas para valores superiores al percentil 99 por tienda, aunque eficaz para reducir el ruido, causó la pérdida de datos para ciertos periodos, en particular para el año 2017, resultando en registros con ventas en cero. Esto puede afectar la precisión del modelo, especialmente en la predicción de eventos en periodos donde los datos fueron eliminados.
2. **Datos de Festivos y Transferencia**: La eliminación de ciertos días festivos en función de la variable `transferred` puede haber omitido eventos importantes que pudieron influir en el comportamiento de las ventas.
3. **Limitación Temporal**: El análisis se basa en los datos disponibles, y cualquier cambio en los patrones de ventas o eventos significativos posteriores al periodo cubierto por estos datos no se reflejará en el modelo.

Este documento proporciona una base sólida para la preparación de datos y puede ser extendido con mejoras en la limpieza de outliers y la integración de otras fuentes de datos para mayor robustez.