```
# ============================================================
# Script: Comparison Source and Synthetic DSWB Training.py
# Purpose: Compare between the source and the synthetic dataset
# Author: Tathagata Bhattacharjee
# Dependencies: pandas, SQLAlchemy, matplotlib, seaborn, dox and io
# # Note: Ensure all required Python packages are installed before running.
# ============================================================

# Import required libraries
import pandas as pd  # For data manipulation and analysis
from sqlalchemy import create_engine, text  # For connecting to SQL databases
import matplotlib.pyplot as plt  # For creating plots
import seaborn as sns  # For enhanced statistical plots
from docx import Document  # For creating Word (.docx) documents
from docx.shared import Inches  # For resizing images in the Word document
from io import BytesIO  # For in-memory byte buffer to handle images

########################################
# STEP 1: Establish Database Connection
########################################
print("\n########################################")
print("\nSTEP 1: Establish Database Connection\n")

# Create a connection to a PostgreSQL database using SQLAlchemy
# Replace credentials with your actual username, password, host, port, and database name
engine = create_engine('postgresql://postgres:password1234@localhost:5432/PhD')

########################################
# STEP 2: Load Data from Source and Synthetic Tables
########################################
print("\n########################################")
print("\nSTEP 2: Load Data from Source and Synthetic Tables\n")

# SQL queries to select relevant fields from the source and synthetic datasets
source_query = """
    SELECT id, first_name, last_name, sex, dob, village_name, occupation,
    covid_19_first_vacc_date, age_on_first_vacc, vacc_manufacturer
    FROM synthetic_dswb_training.source_dataset_fake;
"""
synthetic_query = """
    SELECT id, first_name, last_name, sex, dob, village_name, occupation,
    covid_19_first_vacc_date, age_on_first_vacc, vacc_manufacturer
    FROM synthetic_dswb_training.synthetic_dataset_dswb_training;
"""

# Try to load data into pandas DataFrames using SQL queries
# Print basic shape info or error if loading fails
try:
    source_data = pd.read_sql(source_query, engine)
    synthetic_data = pd.read_sql(synthetic_query, engine)
    print("Data loaded successfully from both tables.")
    print(f"source Data Shape: {source_data.shape}")
    print(f"Synthetic Data Shape: {synthetic_data.shape}")
except Exception as e:
    print(f"Error loading data: {e}")
    exit()  # Exit if there's an error

########################################
# STEP 3: Data Preprocessing for Comparison (if needed)
########################################
print("\n########################################")
print("\nSTEP 3: Data Preprocessing for Comparison (if needed)\n")

# Convert 'dob' and 'covid_19_first_vacc_date' to datetime format for both datasets
source_data['dob'] = pd.to_datetime(source_data['dob'], errors='coerce')
source_data['covid_19_first_vacc_date'] =
pd.to_datetime(source_data['covid_19_first_vacc_date'], errors='coerce')
synthetic_data['dob'] = pd.to_datetime(synthetic_data['dob'], errors='coerce')
synthetic_data['covid_19_first_vacc_date'] =
```

```
    pd.to_datetime(synthetic_data['covid_19_first_vacc_date'], errors='coerce')

# Extract date components from date of birth for fine-grained comparison
source_data['year_of_birth'] = source_data['dob'].dt.year
source_data['month_of_birth'] = source_data['dob'].dt.month
source_data['day_of_birth'] = source_data['dob'].dt.day
synthetic_data['year_of_birth'] = synthetic_data['dob'].dt.year
synthetic_data['month_of_birth'] = synthetic_data['dob'].dt.month
synthetic_data['day_of_birth'] = synthetic_data['dob'].dt.day

# Extract date components from vaccination date for detailed analysis
source_data['vacc_year'] = source_data['covid_19_first_vacc_date'].dt.year
source_data['vacc_month'] = source_data['covid_19_first_vacc_date'].dt.month
synthetic_data['vacc_year'] = synthetic_data['covid_19_first_vacc_date'].dt.year
synthetic_data['vacc_month'] = synthetic_data['covid_19_first_vacc_date'].dt.month

########################################
# STEP 4: Graphical and Tabular Comparisons and Saving to Word File
########################################
print("\n########################################")
print("\nSTEP 4: Graphical and Tabular Comparisons and Saving to Word File\n")

# Set a nice seaborn style for all plots
sns.set_style("whitegrid")

# Initialize Word document for saving plots and tables
document = Document()
document.add_heading('Comparison of Source and Synthetic Data', level=0)

# Define reusable function to compare a column visually and tabularly
def plot_comparison_and_save(source, synthetic, column, title, kind='bar'):
    # Add heading for the section in the Word document
    document.add_heading(title, level=1)

    # Create two subplots: one for source and one for synthetic
    fig, axes = plt.subplots(1, 2, figsize=(12, 5))

    # Plot histogram or bar chart for source data
    if kind == 'hist':
        sns.histplot(source[column].dropna(), kde=False, ax=axes[0], bins=30)
    else:
        source[column].value_counts().nlargest(10).plot(kind='barh', ax=axes[0])
        axes[0].invert_yaxis()  # Most frequent at top

    axes[0].set_title(f'source Data - {title}')
    axes[0].set_xlabel(column)

    # Plot histogram or bar chart for synthetic data
    if kind == 'hist':
        sns.histplot(synthetic[column].dropna(), kde=False, ax=axes[1], bins=30)
    else:
        synthetic[column].value_counts().nlargest(10).plot(kind='barh', ax=axes[1])
        axes[1].invert_yaxis()  # Most frequent at top

    axes[1].set_title(f'Synthetic Data - {title}')
    axes[1].set_xlabel(column)

    plt.tight_layout()

    # Save the plot image in memory and insert into Word
    buffer = BytesIO()
    plt.savefig(buffer, format='png')
    document.add_picture(buffer, width=Inches(6))
    plt.close(fig)

    # Add table comparing frequency and percentage of values
    document.add_heading(f'Distribution of {column}', level=2)

    # Compute value counts and percentages
```

```
        source_counts = source[column].value_counts()
        source_percentages = source[column].value_counts(normalize=True) * 100
        synthetic_counts = synthetic[column].value_counts()
        synthetic_percentages = synthetic[column].value_counts(normalize=True) * 100

        # Merge into one DataFrame for side-by-side comparison
        comparison_df = pd.DataFrame({
            'Source Count': source_counts,
            'Source %': source_percentages,
            'Synthetic Count': synthetic_counts,
            'Synthetic %': synthetic_percentages
        }).fillna(0).sort_index()

        # Create a table in Word with headers + rows
        table = document.add_table(rows=comparison_df.shape[0] + 1,
        cols=comparison_df.shape[1] + 1)
        table.style = 'Table Grid'

        # Header row
        hdr_cells = table.rows[0].cells
        hdr_cells[0].text = column  # Column name for index
        for i, col in enumerate(comparison_df.columns):
            hdr_cells[i + 1].text = col

        # Fill in data rows
        for i, index in enumerate(comparison_df.index):
            row_cells = table.rows[i + 1].cells
            row_cells[0].text = str(index)
            for j, val in enumerate(comparison_df.loc[index]):
                row_cells[j + 1].text = str(round(val, 2))

        document.add_paragraph("\n")  # Add space between plots

    # Call the function above for all relevant columns
    plot_comparison_and_save(source_data, synthetic_data, 'sex', 'Sex Distribution')
    plot_comparison_and_save(source_data, synthetic_data, 'village_name', 'Village Name
    Distribution')
    plot_comparison_and_save(source_data, synthetic_data, 'occupation', 'Occupation
    Distribution')
    plot_comparison_and_save(source_data, synthetic_data, 'vacc_manufacturer', 'Vaccine
    Manufacturer Distribution')
    plot_comparison_and_save(source_data, synthetic_data, 'year_of_birth', 'Year of Birth
    Distribution', kind='hist')
    plot_comparison_and_save(source_data, synthetic_data, 'month_of_birth', 'Month of Birth
    Distribution', kind='hist')
    plot_comparison_and_save(source_data, synthetic_data, 'day_of_birth', 'Day of Birth
    Distribution', kind='hist')
    plot_comparison_and_save(source_data, synthetic_data, 'age_on_first_vacc', 'Age on First
    Vaccination Distribution', kind='hist')
    plot_comparison_and_save(source_data, synthetic_data, 'vacc_year', 'Vaccination Year
    Distribution', kind='hist')
    plot_comparison_and_save(source_data, synthetic_data, 'vacc_month', 'Vaccination Month
    Distribution', kind='hist')

    print("\nGraphical and tabular comparisons generated and saved to Word.")

    ########################################
    # STEP 5: Save the Word Document
    ########################################
    print("\n########################################")
    print("\nSTEP 5: Save the Word Document\n")

    # Save the document as a .docx file
    try:
        document.save('comparison_report.docx')
        print("Comparison report saved to comparison_report.docx")
    except Exception as e:
        print(f"Error saving Word document: {e}")

```

```
194   print("################ End of Code ##########################")
195
```