CODE:

HTML Code:

Download Page:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Download Complete</title>
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
<style>
body {
background: linear-gradient(to right, #1f1c2c, #928dab);
color: #ffffff;
font-family: 'Segoe UI', sans-serif;
}
.container {
margin-top: 80px;
max-width: 600px;
background-color: rgba(0, 0, 0, 0.6);
padding: 40px;
border-radius: 15px;
box-shadow: 0 0 20px rgba(255, 255, 255, 0.2);
}
.title-icon {
font-size: 3rem;35
margin-bottom: 10px;
}
.btn-compare {
background-color: #ffb400;
border: none;
padding: 12px 30px;
font-size: 1.2rem;
color: #000;
font-weight: 600;
transition: all 0.3s ease;
}
.btn-compare:hover {
background-color: #e0a800;
color: #fff;
}
h1, h2 {
font-weight: bold;
text-shadow: 1px 1px 2px #000;
}
.footer {
margin-top: 50px;
text-align: center;
font-size: 0.9rem;
color: #ddd;
}
.note {36
font-size: 1rem;
color: #ccc;
margin-top: 20px;
}
</style>
</head>
<body>
<div class="container text-center">
<div class="title-icon"></div>
<h1>Synthetic Data Ready!</h1>
<p class="mb-4">Your synthetic dataset has been generated and is ready for
download.</p>
<!--<a href="/download/{{ filename }}" class="btn btn-success m-
2">Download</a>-->
<div class="note">
<p>Next Step: Compare the real and synthetic data using statistical analysis
and visualizations.</p>
</div>
<form action="/compare" method="post" class="mt-4">
<input type="hidden" name="filename" value="{{ filename }}">
<button class="btn btn-compare">🔍 Compare Data</button>
</form>
</div>
<div class="footer">
</div>
</body>
</html>37
```

<u>Generate Page:</u>

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Generate Synthetic Data</title>
<link
```

```
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
<style>
body {
background: linear-gradient(to right, #1f1c2c, #928dab);
color: #ffffff;
font-family: 'Segoe UI', sans-serif;
}
.container {
margin-top: 60px;
max-width: 900px;
background-color: rgba(0, 0, 0, 0.6);
padding: 40px;
border-radius: 15px;
box-shadow: 0 0 25px rgba(255, 255, 255, 0.15);
}
.btn-ctgan {
background-color: #00c9a7;
border: none;
padding: 12px 30px;
font-size: 1.1rem;
margin-right: 10px;38
transition: background 0.3s ease;
}
.btn-ctgan:hover {
background-color: #00b39f;
}
.btn-custom {
background-color: #ffc107;
border: none;
padding: 12px 30px;
font-size: 1.1rem;
transition: background 0.3s ease;
}
.btn-custom:hover {
background-color: #e0a800;
}
h1, h2 {
font-weight: bold;
text-shadow: 1px 1px 2px #000;
}
.table-container {
background-color: rgba(255, 255, 255, 0.08);
padding: 20px;
border-radius: 10px;
margin-top: 20px;
box-shadow: 0 0 10px rgba(255, 255, 255, 0.05);
}39
```

```
.footer {
margin-top: 50px;
text-align: center;
font-size: 0.9rem;
color: #ccc;
}
</style>
</head>
<body>
<div class="container text-center">
<div class="title-icon"></div>
<h1 class="mb-3">Review and Generate Synthetic Data</h1>
<p class="mb-4">Below is a preview of your uploaded dataset. You can now
choose a model to generate synthetic data.</p>
<div class="table-container table-responsive mb-4">
{{ table | safe }}
</div>
<h2 class="mb-3">Choose a Model to Generate Synthetic Data</h2>
<form action="/generate/ctgan" method="post" class="d-inline">
<input type="hidden" name="filename" value="{{ filename }}">
<button class="btn btn-ctgan">Generate with CTGAN</button>
</form>
<form action="/generate/custom" method="post" class="d-inline">
<input type="hidden" name="filename" value="{{ filename }}">
<button class="btn btn-custom">Generate with Custom GAN</button>
</form>
</div>40
<div class="footer">
</div>
</body>
</html>
```

Index Page :

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Synthetic Data Generator</title>
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
<style>
body {
background: linear-gradient(to right, #1f1c2c, #928dab);
color: #ffffff;
font-family: 'Segoe UI', sans-serif;
}
```

```
.container {
margin-top: 80px;
max-width: 600px;
background-color: rgba(0, 0, 0, 0.6);
padding: 40px;
border-radius: 15px;
box-shadow: 0 0 20px rgba(255, 255, 255, 0.2);
}
.title-icon {41
font-size: 3rem;
margin-bottom: 10px;
}
.upload-file {
width: 100%;
margin-top: 20px;
}
.btn-primary {
background-color: #00c9a7;
border: none;
padding: 12px 25px;
font-size: 1.2rem;
transition: background 0.3s ease;
}
.btn-primary:hover {
background-color: #00b39f;
}
h1 {
font-weight: bold;
text-shadow: 1px 1px 2px #000;
}
.footer {
margin-top: 50px;
text-align: center;
font-size: 0.9rem;
color: #ddd;42
}
</style>
</head>
<body>
<div class="container text-center">
<div class="title-icon"></div>
<h1>Synthetic Data Generator</h1>
<p class="mb-4">Upload your CSV file to generate and compare synthetic
data!</p>
<form action="/upload" method="post" enctype="multipart/form-data">
<input type="file" name="file" class="form-control upload-file" required>
<br>
<button class="btn btn-primary mt-3">Upload CSV</button>
```

```
</form>
</div>
<div class="footer">
</div>
</body>
</html>
```

Report Choice Page:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Choose Report Type</title>
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">43
<style>
body {
background: linear-gradient(to right, #1f1c2c, #928dab);
color: #ffffff;
font-family: 'Segoe UI', sans-serif;
}
.container {
margin-top: 80px;
max-width: 600px;
background-color: rgba(0, 0, 0, 0.6);
padding: 40px;
border-radius: 15px;
box-shadow: 0 0 20px rgba(255, 255, 255, 0.2);
}
.title-icon {
font-size: 3rem;
margin-bottom: 10px;
}
.btn-report {
width: 100%;
margin-top: 20px;
padding: 14px 30px;
font-size: 1.1rem;
font-weight: 600;
border: none;
transition: all 0.3s ease-in-out;
}44
.btn-statistics {
background-color: #00c9a7;
color: #000;
}
```

```
.btn-statistics:hover {
background-color: #00b39f;
color: #fff;
}
.btn-visual {
background-color: #f48fb1;
color: #000;
}
.btn-visual:hover {
background-color: #ec407a;
color: #fff;
}
h1 {
font-weight: bold;
text-shadow: 1px 1px 2px #000;
}
.instruction {
margin-top: 15px;
font-size: 1rem;
color: #ccc;
}45
.footer {
margin-top: 50px;
text-align: center;
font-size: 0.9rem;
color: #ddd;
}
</style>
</head>
<body>
<div class="container text-center">
<div class="title-icon"></div>
<h1>Select Report Type</h1>
<p class="instruction">Choose how you'd like to explore your synthetic vs. real
data comparison.</p>
<form action="/report/statistics" method="post">
<input type="hidden" name="filename" value="{{ filename }}">
<button class="btn btn-report btn-statistics">Statistical Report</button>
</form>
<form action="/report/visualization" method="post">
<input type="hidden" name="filename" value="{{ filename }}">
<button class="btn btn-report btn-visual"> Visualization Report</button>
</form>
</div>
<div class="footer">
</div>
</body>
</html>
```

Report Page:

```
<!DOCTYPE html>
<html lang="en">
<head>46
<meta charset="UTF-8">
<title>EDA Report</title>
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
<style>
body {
background: linear-gradient(to right, #1f1c2c, #928dab);
color: #ffffff;
font-family: 'Segoe UI', sans-serif;
}
.container {
margin-top: 60px;
background-color: rgba(0, 0, 0, 0.6);
padding: 30px;
border-radius: 15px;
box-shadow: 0 0 20px rgba(255, 255, 255, 0.15);
}
h3 {
font-weight: bold;
text-shadow: 1px 1px 3px #000;
color: #00ffb3;
text-align: center;
margin-bottom: 30px;
}
.card {
background-color: rgba(255, 255, 255, 0.1);
border: none;47
border-radius: 10px;
color: #ffffff;
box-shadow: 0 4px 12px rgba(0, 0, 0, 0.3);
}
pre {
background-color: rgba(0, 0, 0, 0.5);
padding: 15px;
border-radius: 8px;
overflow-x: auto;
}
.btn-primary {
background-color: #00c9a7;
border: none;
padding: 10px 25px;
font-size: 1rem;
border-radius: 8px;
}
.btn-primary:hover {
background-color: #00b39f;
}
</style>
</head>
<body>
<div class="container">
<h3> EDA Report</h3>
<!-- Statistical Summary Section -->
<div class="card p-3 mb-4">48
<h5>📈 Statistical Summary</h5>
<pre>{{ stats }}</pre>
</div>
<!-- EDA Visuals and Tables -->
{% for eda_html in eda_html_list %}
<div class="card p-3 mb-4">
{{ eda_html|safe }}
</div>
{% endfor %}
<!-- Navigation Button -->
<div class="text-center mt-3">
<a href="/" class="btn btn-primary">Back</a>
</div>
</div>
</body>
</html>
```

Visualization Page:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Visualization Report</title>
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
<style>
body {
background: linear-gradient(to right, #1f1c2c, #928dab);49
color: #ffffff;
font-family: 'Segoe UI', sans-serif;
}
.container {
margin-top: 60px;
background-color: rgba(0, 0, 0, 0.6);
padding: 40px;
border-radius: 15px;
box-shadow: 0 0 20px rgba(255, 255, 255, 0.2);
}
h2 {
font-weight: bold;
text-shadow: 1px 1px 3px #000;
text-align: center;
margin-bottom: 30px;
}
.img-thumbnail {
border-radius: 10px;
box-shadow: 0 4px 12px rgba(0, 0, 0, 0.5);
transition: transform 0.3s ease;
}
.img-thumbnail:hover {
transform: scale(1.03);
}
.fade-in {
opacity: 0;50
animation: fadeIn ease-in 0.6s forwards;
}
@keyframes fadeIn {
to {
opacity: 1;
}
}
.footer {
margin-top: 40px;
text-align: center;
color: #ccc;
font-size: 0.9rem;
}
</style>
</head>
<body>
<div class="container">
<h2> Visual Comparison</h2>
<div class="row">
{% for plot in plots %}
<div class="col-md-4 mt-4 fade-in" style="animation-delay: {{ loop.index0 * 0.2
}}s;">
<img src="{{ url_for('static', filename='plots/' + plot.split('/')[-1]) }}"
class="img-fluid img-thumbnail">
</div>
{% endfor %}
</div>
```

```
</div>51
<div class="footer">
</div>
</body>
</html>
```

PYTHON CODE:

App.py:

```
from flask import Flask, render_template, request, redirect, send_file,
render_template_string
import pandas as pd
import os
from eda import generate_eda_report, get_statistical_summary
from ctgan_generator import generate_ctgan
from custom_gan import generate_custom_gan
from reports import generate_statistics_report, generate_visualizations
app = Flask(__name__)
UPLOAD_FOLDER = 'uploads'
GENERATED_FOLDER = 'generated'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
app.config['GENERATED_FOLDER'] = GENERATED_FOLDER
df = {}
syn_df = {}
@app.route('/')
def index():
return render_template('index.html')
# @app.route('/')
# def upload_form():
# return render_template_string(open('./upload.html').read())52
@app.route('/upload', methods=['POST'])
def upload():
file = request.files['file']
if file.filename.endswith('.csv'):
filepath = os.path.join(app.config['UPLOAD_FOLDER'], file.filename)
file.save(filepath)
global df
df = pd.read_csv(filepath)
eda_figs = generate_eda_report(df)
eda_html_list = [fig.to_html(full_html=False) for fig in eda_figs]
stats = get_statistical_summary(df)
return render_template('generate.html', eda_html_list=eda_html_list, stats=stats,
filename=file.filename)
return redirect('/')
@app.route('/generate/ctgan', methods=['POST'])
def ctgan():
```

```
global df
global syn_df
syn_df = generate_ctgan(df, "./outputs")
return render_template('download.html')
#return redirect('/')
#print(df)
#html_table = df.to_html(classes='table table-bordered', index=False)
#return f"<h3>Uploaded and Read CSV:</h3>{html_table}"
@app.route('/generate/custom', methods=['POST'])
def generate_customgan():
global df
global syn_df
syn_df = generate_custom_gan(df, "./outputs")53
return render_template('download.html')
@app.route('/compare', methods=['POST'])
def report_choice():
return render_template('report_choice.html')
@app.route('/report/statistics', methods=['POST'])
def statistics():
global df
global syn_df
print(df)
print(syn_df)
return generate_statistics_report(df, syn_df)
@app.route('/report/visualization', methods=['POST'])
def visualization():
global df
global syn_df
print(df)
print(syn_df)
imageList = generate_visualizations(df, syn_df)
return render_template('visualization.html', plots=imageList)
@app.route('/download/<filename>')
def download(filename):
filepath = os.path.join(app.config['GENERATED_FOLDER'], filename)
return send_file(filepath, as_attachment=True)54
if __name__ == "__main__":
#app.run(debug=True)
app.run(debug=True, use_reloader=False, port=2001)
```

CtGAN Genrator :

```
from ctgan import CTGAN
import os
def generate_ctgan(df, output_folder):
print(df)
model = CTGAN(epochs=10)
model.fit(df)
```

```
synthetic = model.sample(len(df))
out_path = os.path.join(output_folder, 'synthetic_ctgan.csv')
synthetic.to_csv(out_path, index=False)
return synthetic
#synthetic_data = generate_ctgan(df,"./outputs")
#print("Generated Synthetic Data:")
#print(synthetic_data.head())
```

Custom GAN :

```
import pandas as pd
import numpy as np
import os
def generate_custom_gan(df, output_folder):
synthetic = df.copy()
for col in df.select_dtypes(include='number'):
synthetic[col] += np.random.normal(0, 1, size=len(df))
out_path = os.path.join(output_folder, 'synthetic_custom.csv')
synthetic.to_csv(out_path, index=False)55
return synthetic
```

Eda :

```
import pandas as pd
import numpy as np
import os
def generate_custom_gan(df, output_folder):
synthetic = df.copy()
for col in df.select_dtypes(include='number'):
synthPetic[col] += np.random.normal(0, 1, size=len(df))
out_path = os.path.join(output_folder, 'synthetic_custom.csv')
synthetic.to_csv(out_path, index=False)
return synthetic
```

Process :

```
import pandas as pd
class DataProcessor:
def __init__(self):
self.df = None
self.synthetic_df = None
def load_data(self, filepath):
df = pd.read_csv(filepath)
self.df = self.preprocess_data(df)
def preprocess_data(self, df):
df = df.copy()
df.dropna(axis=1, how='all', inplace=True)
for col in df.columns:
```

```
if df[col].dtype == 'object':
df[col].fillna(df[col].mode()[0], inplace=True)56
else:
df[col].fillna(df[col].median(), inplace=True)
df = pd.get_dummies(df, drop_first=True)
return df
def set_synthetic_path(self, filepath):
self.synthetic_df = pd.read_csv(filepath)
```

Reports :

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os
def generate_statistics_report(real_df, synth_df):
# Handle missing values
real_df = real_df.fillna(real_df.mean(numeric_only=True))
synth_df = synth_df.fillna(synth_df.mean(numeric_only=True))
numeric_cols = real_df.select_dtypes(include='number').columns
report_html = ""
def stats_for_column(df, col):
series = df[col]
q1 = series.quantile(0.25)
q3 = series.quantile(0.75)
iqr = q3 - q1
mode = series.mode()
mode_val = mode.iloc[0] if not mode.empty else "N/A"
return {
"Mean": round(series.mean(), 2),
"Median": round(series.median(), 2),
"Mode": mode_val,57
"Max": round(series.max(), 2),
"Min": round(series.min(), 2),
"Range": round(series.max() - series.min(), 2),
"Count": int(series.count()),
"Std": round(series.std(), 2),
"Variance": round(series.var(), 2),
"Skewness": round(series.skew(), 2),
"Kurtosis": round(series.kurtosis(), 2),
"IQR": round(iqr, 2)
}
for col in numeric_cols:
real_stats = stats_for_column(real_df, col)
synth_stats = stats_for_column(synth_df, col)
report_htNml += f"<h3>Column: {col}</h3>"
report_html += """
<table class="table table-bordered table-striped">
```

```
<thead>
<tr>
<th>Statistic</th>
<th>Real Data</th>
<th>Synthetic Data</th>
</tr>
</thead>
<tbody>
"""
for stat in real_stats.keys():
report_html += f"""
<tr>
<td>{stat}</td>58
<td>{real_stats[stat]}</td>
<td>{synth_stats[stat]}</td>
</tr>
"""
report_html += "</tbody></table><br>"
return report_html
# === Function 2: All Visualization Types ===
def generate_visualizations(real_df, synth_df):
plot_paths = []
os.makedirs('static/plots', exist_ok=True)
real_df = real_df.fillna(real_df.mean(numeric_only=True))
synth_df = synth_df.fillna(synth_df.mean(numeric_only=True))
numeric_cols = real_df.select_dtypes(include='number').columns.tolist()
# === UNIVARIATE PLOTS ===
for col in numeric_cols[:4]:
# Histogram
plt.figure()
sns.histplot(real_df[col], color='blue', kde=False, label='Real', stat='density')
sns.histplot(synth_df[col], color='green', kde=False, label='Synthetic',
stat='density')
plt.legend()
plt.title(f'Histogram - {col}')
path = f'static/plots/histogram_{col}.png'
plt.savefig(path); plt.close()
plot_paths.append(path)59
# KDE
plt.figure()
sns.kdeplot(real_df[col], label='Real', fill=True)
sns.kdeplot(synth_df[col], label='Synthetic', fill=True)
plt.title(f'KDE Plot - {col}')
plt.legend()
path = f'static/plots/kde_{col}.png'
plt.savefig(path); plt.close()
plot_paths.append(path)
# Boxplot
```

```
plt.figure()
sns.boxplot(data=[real_df[col], synth_df[col]], palette=["blue", "green"])
plt.xticks([0, 1], ['Real', 'Synthetic'])
plt.title(f'Boxplot - {col}')
path = f'static/plots/boxplot_{col}.png'
plt.savefig(path); plt.close()
plot_paths.append(path)
# Violin Plot
plt.figure()
sns.violinplot(data=[real_df[col], synth_df[col]], palette=["blue", "green"])
plt.xticks([0, 1], ['Real', 'Synthetic'])
plt.title(f'Violin Plot - {col}')
path = f'static/plots/violin_{col}.png'
plt.savefig(path); plt.close()
plot_paths.append(path)
# === BIVARIATE PLOTS ===
if len(numeric_cols) >= 2:60
x, y = numeric_cols[0], numeric_cols[1]
# Scatter Plot
plt.figure()
sns.scatterplot(x=real_df[x], y=real_df[y], color='blue', label='Real', alpha=0.6)
sns.scatterplot(x=synth_df[x], y=synth_df[y], color='green', label='Synthetic',
alpha=0.6)
plt.legend()
plt.title(f'Scatter Plot - {x} vs {y}')
path = f'static/plots/scatter_{x}_{y}.png'
plt.savefig(path); plt.close()
plot_paths.append(path)
# Joint KDE Plot
g = sns.jointplot(data=real_df, x=x, y=y, kind='kde', fill=True, color='blue')
g.fig.suptitle(f'Real KDE Joint Plot - {x} vs {y}')
g.fig.tight_layout()
g.fig.subplots_adjust(top=0.95)
path = f'static/plots/jointplot_real_{x}_{y}.png'
g.fig.savefig(path); plt.close()
plot_paths.append(path)
# === MULTIVARIATE PLOTS ===
if len(numeric_cols) >= 3:
# Pairplot (Real)
sns.pairplot(real_df[numeric_cols[:4]], diag_kind='kde', corner=True)
plt.suptitle('Real Data Pairplot', y=1.02)
path = 'static/plots/pairplot_real.png'
plt.savefig(path); plt.close()
plot_paths.append(path)
# Correlation Heatmap61
plt.figure(figsize=(8, 6))
sns.heatmap(real_df[numeric_cols].corr(), annot=True, cmap='coolwarm')
plt.title('Real Data Correlation Heatmap')
```

```
path = 'static/plots/heatmap_real.png'
plt.savefig(path); plt.close()
plot_paths.append(path)
return plot_paths
```