Create Synthetic data

To generate the data we gonna use the SDV or Synthetic Data Vault. SVD generates synthetic data by applying mathematical techniques and machine learning models such as the deep learning model. Even if the data contain multiple data types and missing data, SDV will handle it, so we only need to provide the data.

```
# ! pip install sdv
```

```
import pandas as pd
import os
import numpy as np
from sdv.tabular import CTGAN, GaussianCopula, CopulaGAN
from sdv.evaluation import evaluate
from sdmetrics.reports.single_table import QualityReport
from sdmetrics.reports.utils import get_column_plot
import warnings
warnings.filterwarnings('ignore')
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
    Mounted at /content/drive
```

```
data_drive_path = os.path.join('drive', 'MyDrive', 'Colab Notebooks', 'SP-project')
```

```
infringement_path = os.path.join(data_drive_path, 'dataset', 'infringement_dataset.csv')
```

Load data

We will choose only 10% of the original data to generate sintetic data since the dataset is too large

[ ] ↳ *3 cells hidden*

Create and train model - CTGAN

```
%%time
model_ctgan = CTGAN(epochs=15, generator_dim=(256, 256), discriminator_dim=(256, 256), ver
model_ctgan.fit(data)
```

```
    Epoch 1, Loss G:  0.7906,Loss D:  0.1004
```

```
Epoch 2, Loss G:  0.6128,Loss D: -0.0214
Epoch 3, Loss G:  0.3220,Loss D: -0.1391
Epoch 4, Loss G:  0.4297,Loss D: -0.3106
Epoch 5, Loss G: -0.4470,Loss D:  0.0137
Epoch 6, Loss G: -0.6839,Loss D: -0.2055
Epoch 7, Loss G: -1.0638,Loss D: -0.4535
Epoch 8, Loss G: -0.0520,Loss D: -0.5705
Epoch 9, Loss G:  0.4506,Loss D: -1.3799
Epoch 10, Loss G: -0.4706,Loss D: -0.6767
Epoch 11, Loss G: -1.9495,Loss D: -1.0148
Epoch 12, Loss G: -2.3548,Loss D: -0.8926
Epoch 13, Loss G: -2.4292,Loss D: -0.0388
Epoch 14, Loss G: -2.7644,Loss D: -0.0175
Epoch 15, Loss G: -2.5837,Loss D:  0.0145
CPU times: user 1min 55s, sys: 53.5 s, total: 2min 48s
Wall time: 2min 21s
```

After fitting the model we gonna use it to generate the new data

```
%%time
synthetic_data_ctgan = model_ctgan.sample(num_rows=data.shape[0])
synthetic_data_ctgan
```

```
CPU times: user 1.81 s, sys: 50.8 ms, total: 1.86 s
Wall time: 1.87 s
```

	loan_id	infringed	contract_type	gender	has_own_car	has_own_realty	num_c
0	314803	0	Cash loans	F	Y	N	
1	158588	0	Cash loans	F	Y	Y	
2	439382	0	Cash loans	F	Y	Y	
3	100032	0	Cash loans	F	N	N	
4	398343	0	Cash loans	M	Y	N	
...	...	...	...	...	...	...	
30746	162334	0	Cash loans	F	N	Y	
30747	235500	0	Cash loans	F	Y	Y	
30748	419072	0	Cash loans	F	N	Y	
30749	173926	0	Cash loans	F	Y	Y	
30750	197216	0	Revolving loans	F	N	Y	

30751 rows × 31 columns

```
save_path = os.path.join(data_drive_path, 'dataset', 'synthetic_data_CTGAN.csv')
synthetic_data_ctgan.to_csv(save_path, index=False)
```

Evaluate results

```
model_score = evaluate(synthetic_data_ctgan, data)
model_score
```

```
0.9132126326596439
```

This report evaluates the shapes of the columns (marginal distributions) and the pairwise trends between the columns (correlations).

```
report_ctgan = QualityReport()

report_ctgan.generate(data, synthetic_data_ctgan,model_ctgan.get_metadata().to_dict())
```

```
Creating report: 100%|██████████| 4/4 [00:04<00:00,  1.18s/it]

Overall Quality Score: 88.59%

Properties:
Column Shapes: 89.07%
Column Pair Trends: 88.11%
```

```
details = report_ctgan.get_details(property_name='Column Shapes')
details
```

	Column	Metric	Quality Score
0	loan_id	KSComplement	0.890117
1	infringed	KSComplement	0.991903
2	num_children	KSComplement	0.927807
3	annual_income	KSComplement	0.889142
4	credit_amount	KSComplement	0.747878
5	credit_annuity	KSComplement	0.827054
6	goods_valuation	KSComplement	0.671483
7	age	KSComplement	0.881532
8	days_employed	KSComplement	0.717082
9	mobilephone_reachable	KSComplement	0.971871
10	num_family_members	KSComplement	0.943221
11	SK_ID_CURR	KSComplement	0.870337
12	avg_days_decision	KSComplement	0.967442
13	past_avg_amount_annuity	KSComplement	0.879684
14	past_avg_amt_application	KSComplement	0.946726
15	past_avg_amt_credit	KSComplement	0.749906

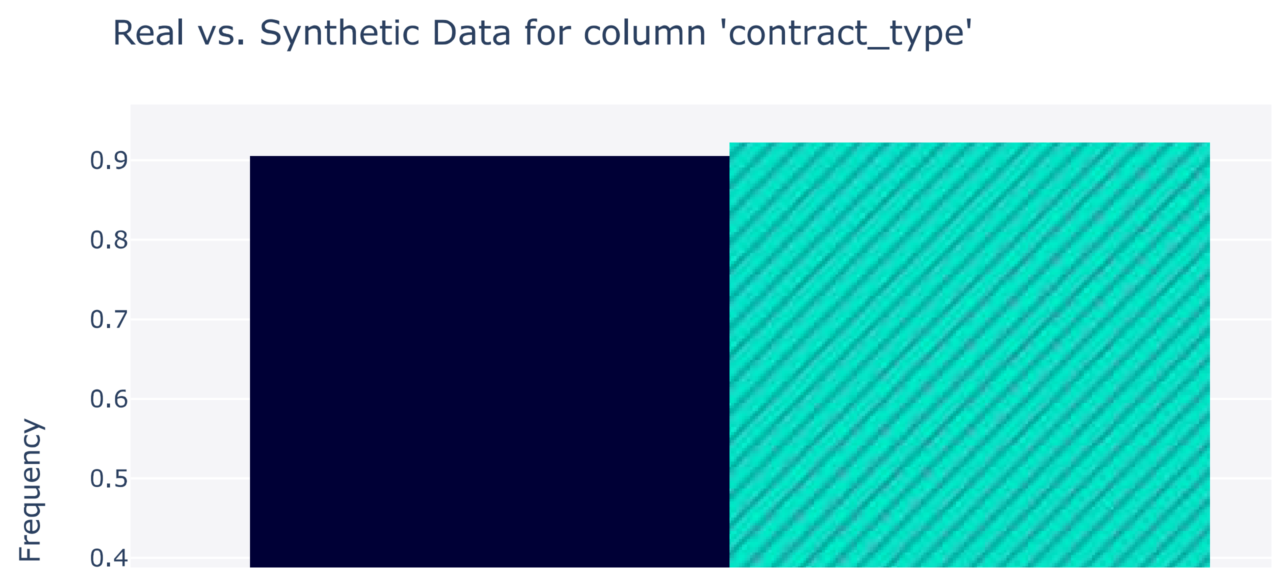
```
print('Column with more quality',details[details['Quality Score'] == details['Quality Scor
print('Column with less quality',details[details['Quality Score'] == details['Quality Scor
```

```
Column with more quality 1    infringed
Name: Column, dtype: object 0.9919027023511431
Column with less quality 6    goods_valuation
Name: Column, dtype: object 0.6714834665826337
```

```
report_ctgan.get_visualization(property_name='Column Pair Trends')
```

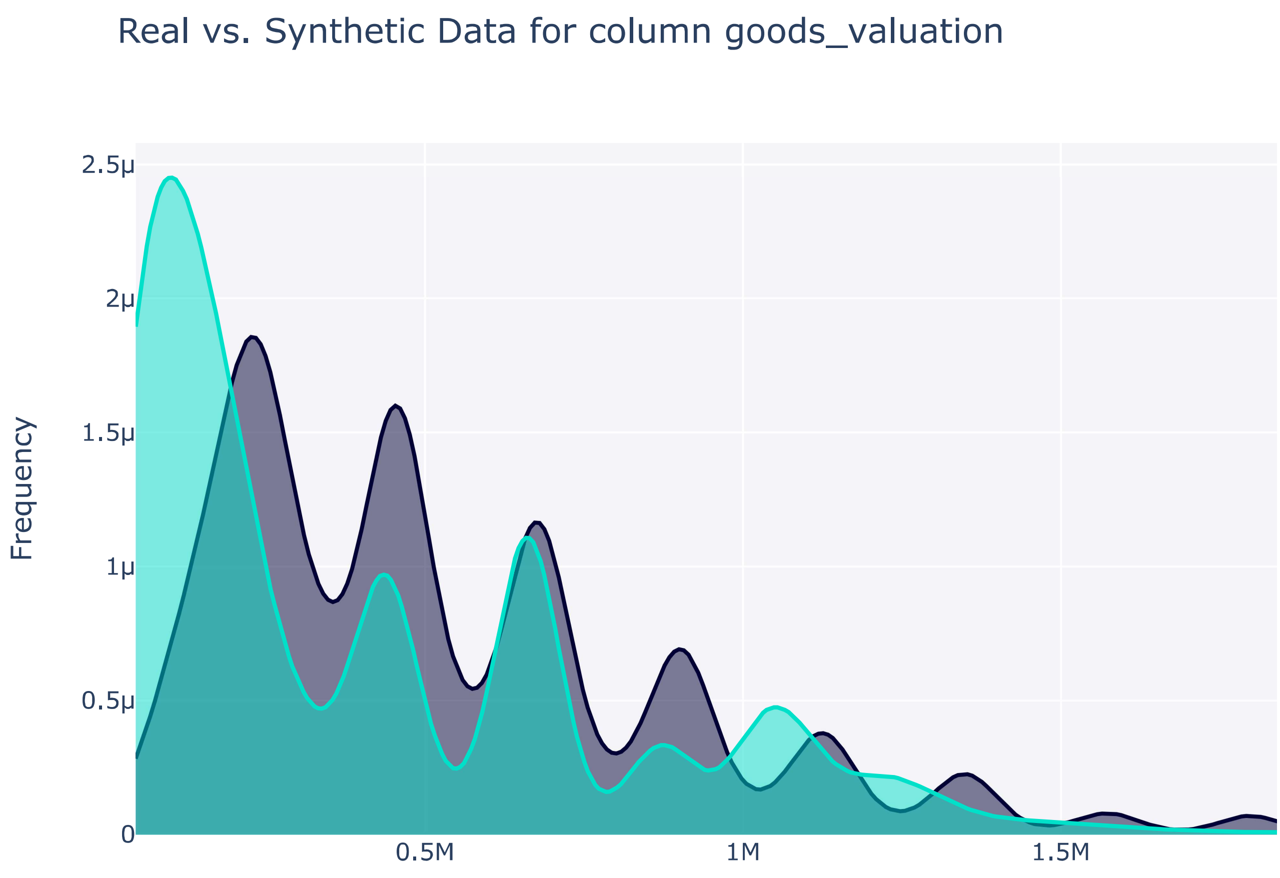

```
fig = get_column_plot(
    real_data=data,
    synthetic_data=synthetic_data_ctgan,
    metadata=model_ctgan.get_metadata().to_dict(),
    column_name='contract_type'
)

fig.show()
```




```
fig = get_column_plot(
    real_data=data,
    synthetic_data=synthetic_data_ctgan,
    metadata=model_ctgan.get_metadata().to_dict(),
    column_name='goods_valuation'
)

fig.show()
```



Create and train model - GaussianCopula

```
%%time
model_gauscopula = GaussianCopula()
model_gauscopula.fit(data)
```

```
CPU times: user 3.02 s, sys: 42 ms, total: 3.06 s
Wall time: 3.09 s
```

Generate new data

```
%%time
synthetic_data_gausscopula = model_gauscopula.sample(num_rows=data.shape[0])
synthetic_data_gausscopula
```

```
CPU times: user 1.01 s, sys: 141 ms, total: 1.15 s
Wall time: 1.02 s
```

	loan_id	infringed	contract_type	gender	has_own_car	has_own_realty	num_c
0	157558	0	Cash loans	F	Y	Y	
1	152767	0	Cash loans	F	N	Y	
2	411530	0	Cash loans	F	N	Y	
3	410414	0	Cash loans	M	Y	Y	
4	418388	0	Cash loans	F	N	N	
...	...	...	...	...	...	...	
30746	257040	0	Cash loans	F	N	Y	
30747	331768	1	Cash loans	F	Y	Y	
30748	375438	0	Cash loans	F	N	Y	
30749	361763	0	Cash loans	F	N	Y	
30750	207243	0	Cash loans	M	N	Y	

30751 rows × 31 columns

```
save_path = os.path.join(data_drive_path, 'dataset', 'synthetic_data_GaussianCopula.csv')
synthetic_data_gausscopula.to_csv(save_path, index=False)
```

Evaluate results

```
model_score_gauss = evaluate(synthetic_data_gausscopula, data)
model_score_gauss
```

```
0.9045081606776357
```

```
report_gausscopula = QualityReport()
```

```
report_gausscopula.generate(data, synthetic_data_gausscopula,model_gauscopula.get_metadata
```

```
Creating report: 100%|██████████| 4/4 [00:04<00:00,  1.12s/it]

Overall Quality Score: 90.71%

Properties:
Column Shapes: 88.5%
Column Pair Trends: 92.93%
```

```
details_gausscopula = report_gausscopula.get_details(property_name='Column Shapes')
details
```

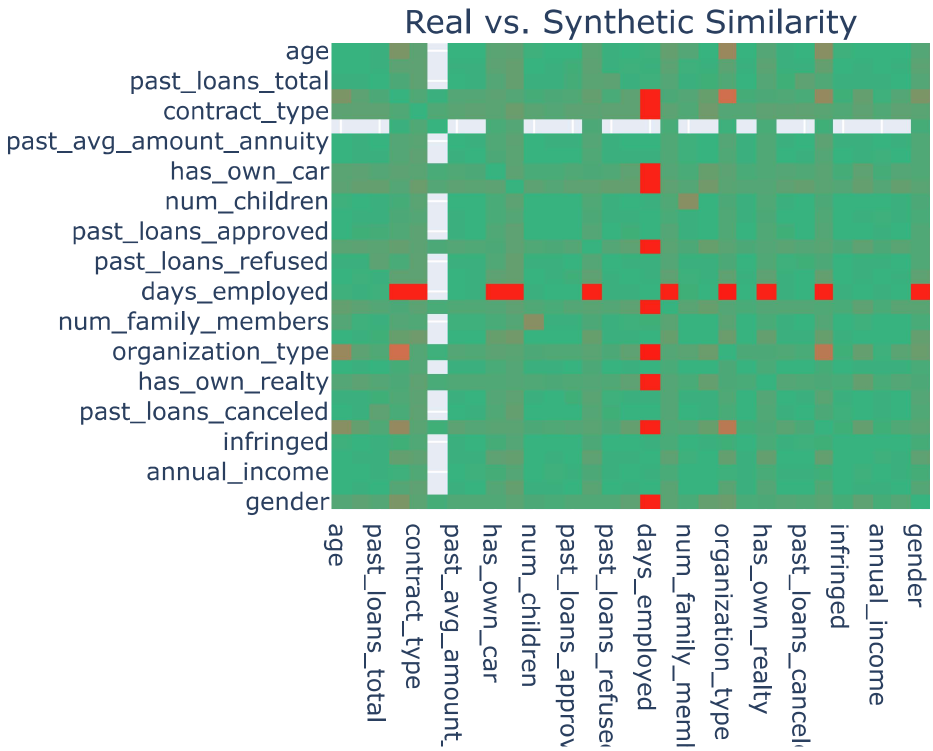
	Column	Metric	Quality Score
0	loan_id	KSComplement	0.890117
1	infringed	KSComplement	0.991903
2	num_children	KSComplement	0.927807
3	annual_income	KSComplement	0.889142
4	credit_amount	KSComplement	0.747878
5	credit_annuity	KSComplement	0.827054
6	goods_valuation	KSComplement	0.671483
7	age	KSComplement	0.881532
8	days_employed	KSComplement	0.717082
9	mobilephone_reachable	KSComplement	0.971871
10	num_family_members	KSComplement	0.943221

```
print('Column with more quality',details_gausscopula[details_gausscopula['Quality Score']
print('Column with less quality',details_gausscopula[details_gausscopula['Quality Score']
```

```
Column with more quality 9    mobilephone_reachable
Name: Column, dtype: object 0.9984065558843614
Column with less quality 8    days_employed
Name: Column, dtype: object 0.2623329322623654
```

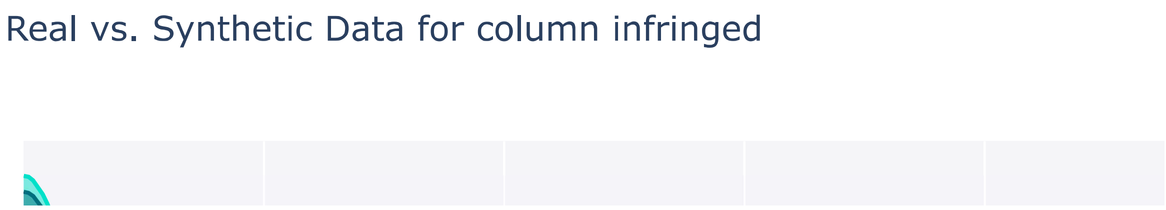
```
report_gausscopula.get_visualization(property_name='Column Pair Trends')
```

Data Quality: Column Pair Trends (Average Score=0.93)



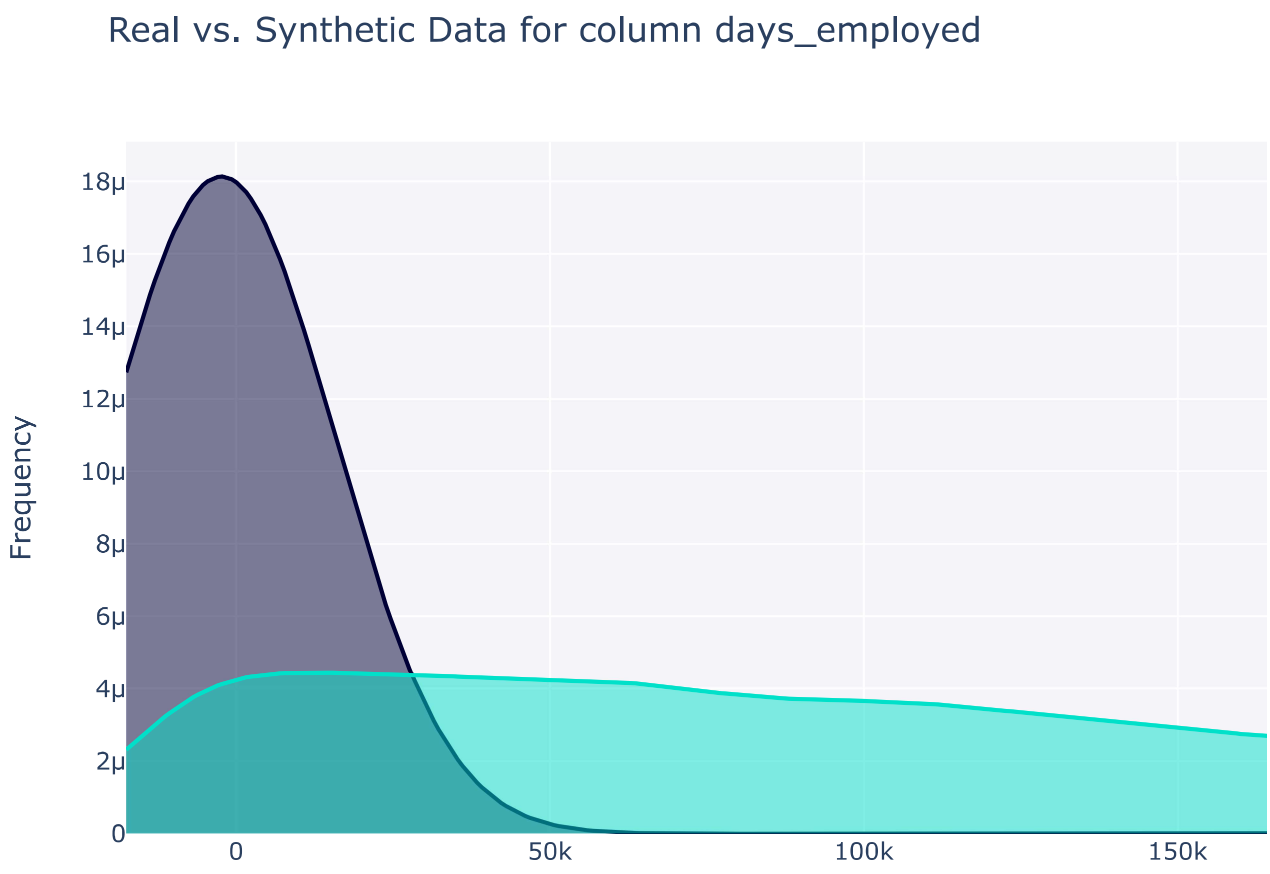

```
fig = get_column_plot(
    real_data=data,
    synthetic_data=synthetic_data_gausscopula,
    metadata=model_gauscopula.get_metadata().to_dict(),
    column_name='infringed'
)

fig.show()
```




```
fig = get_column_plot(
    real_data=data,
    synthetic_data=synthetic_data_gausscopula,
    metadata=model_gauscopula.get_metadata().to_dict(),
    column_name='days_employed'
)

fig.show()
```



Create and train model - CopulaGAN

```
%%time
model_copulagan = CopulaGAN(epochs=15, generator_dim=(256, 256), discriminator_dim=(256, 2
model_copulagan.fit(data)
```

```
Epoch 1, Loss G:  0.5447,Loss D:  0.2093
```

```
Epoch 2, Loss G: -1.1857,Loss D:  0.4886
Epoch 3, Loss G: -1.3961,Loss D:  0.4278
Epoch 4, Loss G: -1.0820,Loss D: -0.4004
Epoch 5, Loss G: -1.5901,Loss D: -0.3488
Epoch 6, Loss G: -1.4939,Loss D:  0.0390
Epoch 7, Loss G: -0.8734,Loss D: -0.8560
Epoch 8, Loss G: -0.1958,Loss D: -0.8793
Epoch 9, Loss G: -0.7367,Loss D: -0.6179
Epoch 10, Loss G: -0.7942,Loss D: -0.4425
Epoch 11, Loss G: -1.4429,Loss D: -0.6471
Epoch 12, Loss G: -1.4352,Loss D: -0.3106
Epoch 13, Loss G: -0.9736,Loss D: -0.6985
Epoch 14, Loss G: -0.6134,Loss D: -1.1475
Epoch 15, Loss G: -2.2581,Loss D: -0.5008
CPU times: user 1min 58s, sys: 54.6 s, total: 2min 53s
Wall time: 2min 20s
```

```
%%time
synthetic_data_copulagan = model_copulagan.sample(num_rows=data.shape[0])
synthetic_data_copulagan
```

```
CPU times: user 2.08 s, sys: 46.4 ms, total: 2.13 s
Wall time: 2.12 s
```

	loan_id	infringed	contract_type	gender	has_own_car	has_own_realty	num_c
0	382313	0	Cash loans	F	N	N	
1	249431	0	Cash loans	F	N	Y	
2	291691	0	Cash loans	F	Y	Y	
3	303327	1	Cash loans	F	N	N	
4	246202	0	Cash loans	F	N	N	
...	...	...	...	...	...	...	
30746	353283	1	Revolving loans	F	N	N	
30747	102355	1	Cash loans	F	Y	N	
30748	152873	0	Cash loans	M	Y	N	
30749	363316	0	Cash loans	F	Y	Y	
30750	247955	0	Cash loans	F	Y	Y	

30751 rows × 31 columns

```
save_path = os.path.join(data_drive_path, 'dataset', 'synthetic_data_CopulaGAN.csv')
synthetic_data_copulagan.to_csv(save_path, index=False)
```

Evaluate results

```
model_score_copulagan = evaluate(synthetic_data_copulagan, data)
model_score_copulagan
```

```
0.9130157037101264
```

```
report_copulagan = QualityReport()

report_copulagan.generate(data, synthetic_data_copulagan,model_copulagan.get_metadata().to
```

```
Creating report: 100%|██████████| 4/4 [00:04<00:00,  1.13s/it]

Overall Quality Score: 88.65%

Properties:
Column Shapes: 89.78%
Column Pair Trends: 87.52%
```

```
details_copulagan = report_copulagan.get_details(property_name='Column Shapes')
details_copulagan
```

	Column	Metric	Quality Score
0	loan_id	KSComplement	0.870411
1	infringed	KSComplement	0.922116
2	num_children	KSComplement	0.812884
3	annual_income	KSComplement	0.776918
4	credit_amount	KSComplement	0.867907
5	credit_annuity	KSComplement	0.832748
6	goods_valuation	KSComplement	0.852164
7	age	KSComplement	0.942961
8	days_employed	KSComplement	0.802250
9	mobilephone_reachable	KSComplement	0.998407
10	num_family_members	KSComplement	0.878508
11	SK_ID_CURR	KSComplement	0.879458

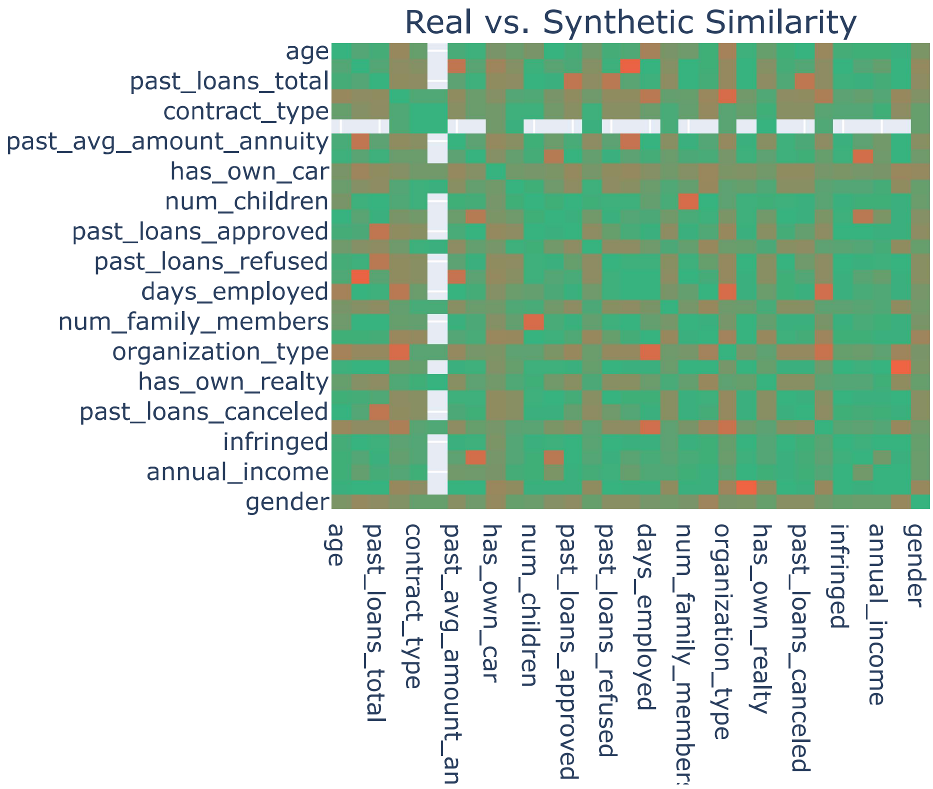
```
print('Column with more quality',details_copulagan[details_copulagan['Quality Score'] == d
print('Column with less quality',details_copulagan[details_copulagan['Quality Score'] == d
```

```
Column with more quality 9    mobilephone_reachable
Name: Column, dtype: object 0.9984065558843614
Column with less quality 3    annual_income
Name: Column, dtype: object 0.7769178238106078
```

16	past_loans_approved	KSComplement	0.882280

```
report_copulagan.get_visualization(property_name='Column Pair Trends')
```

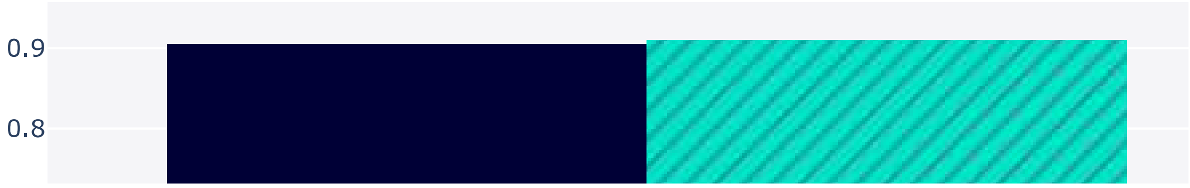
Data Quality: Column Pair Trends (Average Score=0.88)




```
fig = get_column_plot(
    real_data=data,
    synthetic_data=synthetic_data_copulagan,
    metadata=model_copulagan.get_metadata().to_dict(),
    column_name='contract_type' # second best since the fist has some bugs
)

fig.show()
```

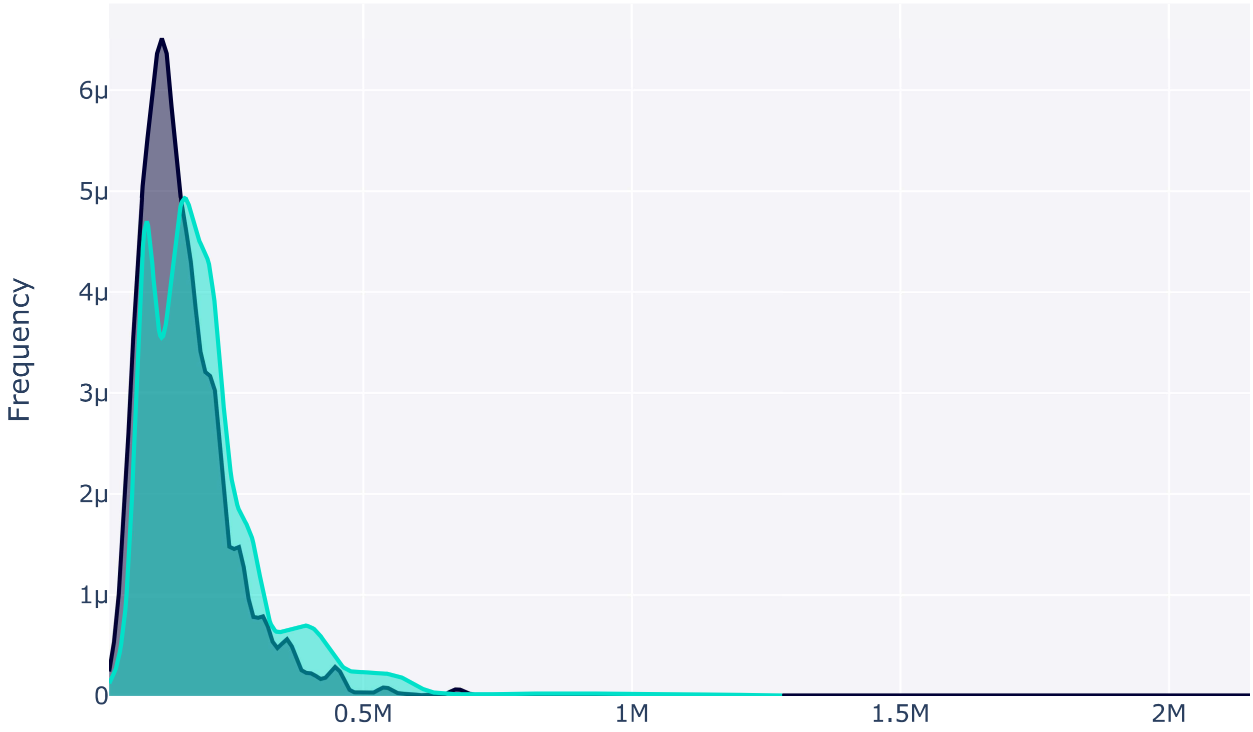
Real vs. Synthetic Data for column 'contract_type'




```
fig = get_column_plot(
    real_data=data,
    synthetic_data=synthetic_data_copulagan,
    metadata=model_copulagan.get_metadata().to_dict(),
    column_name='annual_income'
)

fig.show()
```

Real vs. Synthetic Data for column annual_income



Result analysis

By observing the results, we can see that the overall quality score of our new dataset is 81%. This result is good, but we can do a deeper analysis and see the scores on the individual columns and between the correlation in the columns. The result is basically the same, so we can

Advantages

- **Data quality** - Higher data quality, balance, and variety are ensured with synthetic data. Artificially created data can apply labels and automatically fill in missing quantities, allowing for more precise prediction;
- **Scalability** - Synthetic data is used to cover the gaps left by real-world data;
- **Utilization simplicity** - Synthetic data guarantees all data has a consistent format and labelling, getting rid of errors and duplicates.

Disadvantages

- Outliers are challenging to map because synthetic data merely approximates real-world data, it is not a duplicate.Therefore, some outliers that are present in original data may not be covered by synthetic data
- The quality of the model depends on the data source.