Multi-table Synthetic Data: Benchmarking, Model Evaluation, Vendor Comparison

Rajiv lyer and Vincent Granville
vincentg@MLTechniques.com
www.GenAItechLab.com

Version 1.0, June 2024

Abstract

Synthesizing multi-table tabular data presents its own challenges, compared to single-table. When the database contains date columns such as transaction or admission date, a frequent occurrence in real-word datasets, generating high quality synthetizations and model evaluation are even more complicated. In this article, we focus on this type of problems, comparing generated observations produced by 3 vendors and open source. We look at preservation of data integrity across the multiple tables, run time, correct replication of the joint multivariate distribution present in the real data in parent and child tables, and non-violation of business rules based on combination of features, including categorical features.

Contents

1 Why multi-table synthetization?

Businesses need synthetic tabular data for a variety of reasons: preserving privacy when exchanging data with third parties, balancing datasets when some categories have few observations (for instance in fraud detection), generating new observations when acquiring real data is expensive (for instance in clinical trials), recovering missing values, reducing bias by oversampling minorities, generating unusual records by sampling outside the observation range, data augmentation to increase the performance and robustness of predictive or clustering models, or for various testing and benchmarking purposes (for instance to assess how a clustering algorithm might behave on future data, or to stress-test hardware).

In this article, I focus on business applications and tabular data, as opposed to sound, video, image, graphs and other types of synthetization. However the datasets in question – actual databases with multiple connected tables – may contain features that behave as time series or geospatial data. The only constraint is that the data is available in tabular format. For instance, most transactional datasets fit in that category. Features may be numerical, categorical including text, or hybrid; missing values are frequently present.

A year ago, I published a similar article, comparing multiple vendors, see [8]. I tested the products using the default settings. The standard open source library SDV (Synthetic Data Vault) performed very poorly. No doubt that you can get better results if you spend weeks on fine-tuning and understanding the code, options, and parameters. This applies to vendors as well. However, busy business people don't have the expertise needed to dig deep into these products, and expect a quick and easy solution to save time and money. By contrast to my earlier article covering single-table data, I now focus on muti-table datasets. The landscape also changed quite a bit in a year: Gretel has significantly improved its algorithms, now outperforming Mostly.ai. Synthetizations are produced much faster. Yet, YData.ai still dominates in terms of quality.

But why not joining all the tables to reduce the problem to single-table synthetization? Many times, a multi-table dataset consists of a main granular table, for instance user transactions. Satellite tables have extra information about the users, merchants and so on. There are many reasons to synthesize these tables separately, see here. Part of the evaluation process consists of checking if database integrity is preserved. Also, you want to assess if the statistical relationships between all the tables are properly rendered. For instance, is the number of merchants and users per merchant similar in the transaction and merchant tables? Finally, data synthetization should not be confused with simulation: the difference is explained in this article.

2 Methodology, best practices

In this section, after briefly reviewing synthetization methods, I discuss model evaluation metrics. In particular, I designed an efficient computation of the full multivariate ECDF in any dimension, also called joint empirical cumulative distribution function. Here the dimension is the number of features. It works both with categorical and numerical features. Then, the best evaluation metric for synthetic tabular data is arguably the multivariate Kolmogorov-Smirnov distance defined as

$$\text{KS} = \sup_z \Big| \text{ECDF}_{\text{synth}}(z) - \text{ECDF}_{\text{real}}(z) \Big|. \tag{1}$$

It is computed over all locations z in the feature space, and takes values between 0 (best fit) and 1 (worst fit). The two ECDFs in the formula are attached respectively to the synthetic and the real data. The latter is either the training set used for synthetization, or validation data in a cross-validation setting. Alternatives to KS are discussed in section 2.2

2.1 Quick overview of synthetization methods

Most modern methods rely on generative adversarial networks (GAN), consisting of two deep neural networks competing against each other [3, 5, 16, 20]. The Wasserstein loss function [2, 19] is frequently used, along with Adam's stochastic gradient descent.

The copula method [14] is much older and may be used to produce an initial synthetization as starting point for more sophisticated algorithms. It replicates the marginal distribution attached to each feature separately, and the underlying correlation structure present in the real data. However, it fails to replicate more complex, non-linear cross-feature dependency patterns. Copulas are extensively used in [4].

In 2023, I introduced fast NoGAN methods that can reproduce all the patterns found in the real data. In this case, the best synthetization is the real data itself, up to a permutation of the observations when the real data and synthetization have the same number of observations. Obviously, this may lead to overfitting. Thus, the goal is to minimize some loss function under regularization constraints to avoid overfitting. This is known as constrained synthetization. I then tried to use the multivariate KS distance (the evaluation metric) as the loss function, but it seems computationally non feasible in high dimensions. Instead, I used successive approximations converging to the multivariate Hellinger distance, resulting in an adaptive loss function. Hellinger is equivalent to KS; in one dimension, it is identical to Wasserstein. Since the loss function is a proxy to the evaluation metric, using the evaluation as loss function can lead to better results. It is a promising approach in methods based on neural networks. See section 2.4 in [12].

For time series and geospatial data, different techniques are used, for instance interpolation: see chapter 3 in [12]. For time series, the goal is to properly replicate the autocorrelation function. Finally, I used large language models (LLMs) to synthesize DNA sequences, see section 7.1 in [12].

As a side note, you can improve synthetic data as follows: generate more artificial observations than desired; compute the distance between each generated vector and the real data (closest neighbor in the real data). Drop synthetic vectors that are far away from any real observation. This is discussed in section 8.1 in [12]. It is possible to speed up synthetization without loss of quality, by reducing the number of observations in the training set, using data distillation [9, 21]. Or by synthesizing independent feature clusters separately [7]. When dealing with a large number of categorical features, smart encoding of multi-category vectors is one of the best solutions, discussed in slide 17, here. Another strategy to improve quality consists in applying a number of transforms to the training set before synthesizing, followed by the inverse transforms in reverse order post-synthetization. For instance, it makes sense to use the logarithm of stock prices or salary data. PCA and normalization transforms may also help.

2.2 Model evaluation

To evaluate the quality of the synthetizations, we looked at five different sets of metrics: ease of use, run time (training and time to generate the observations), data integrity, faithfulness, and business rules when applicable. Most of the computing time is spent on training. More specifically, we performed the following tests:

- **Data integrity**. This includes the following tests: checking for uniqueness in Primary Keys, verifying that all the foreign keys in child tables are present in the parent table, and verifying that the check constraints conditions are satisfied.
- **Faithfulness**. We computed the full multivariate KS distance specified in Formula (1), using the GenAI-evaluation library. For details about this Python library, see here. We removed the ID and date columns for this test. Also, all categorical features were combined using smart encoding offered by the library and discussed in section 2.1. The evaluation score is an average computed over all individual tables.
- **Business rules**. Tests focusing on date columns, compound features relevant to the business (for instance, balance minus transaction amount), and statistical consistency between the child and parent tables.

Correct detection and replication of underlying business rules is best illustrated in Figure 1. In this example, the real data exhibits 4 clusters: credit cards with small, medium, large, and and very large transactions, possibly reflecting typical credit card limits. Then, over the 10-year time period, you can see cycles in the volume of transactions. Perhaps the most striking feature is the double periodicity: monthly, and semester. YData is the only vendor that captures this feature. The bank is in the Czech Republic.

Gretel exhibits exaggerated diffusion both in time and credit card transaction amounts. YData amounts were slightly randomized due to the opposite problem: not enough granularity. Both Mostly.ai and Gretel have issues that cannot easily be fixed. Adding an extra feature to the dataset – a cluster label to discriminate between credit cards with low and high limits – may improve the replication of these clusters.

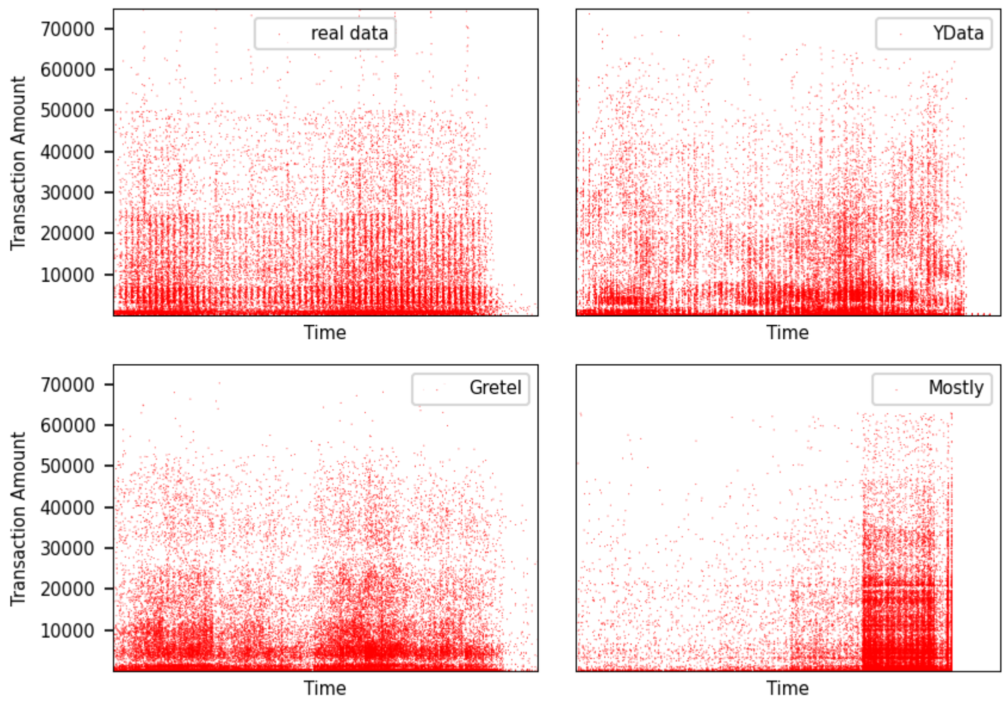

Figure 1: Credit card data scatterplot, time vs amount

The right plot in Figure 2 illustrates another business rule. In general, the balance on an credit card must be higher than the authorized transaction amount. Thus, you see a sharp cutoff point in the real dataset, when both amounts are identical. YData was able to detect and replicate this feature, but the other vendors failed. Another problem is generating categories that cannot exist. For instance, in a cancer dataset, one feature may be gender, and another one may be cancer type. You cannot generate men with ovarian cancer. YData is the only vendor not producing combinations that do not exist in the real data: see section 3.1 for details. For a more systematic approach, one may compare ratios in the child and parent tables. For instance: average number of transactions per user or per year, or number of users in each category.

Faithfulness – the ability to correctly reproduce the full multivariate distribution found in the real data – is measured via the multivariate KS distance defined by (1). For simple ad-hoc analyses involving a 1-dimensional compound feature or time, I used a simplified metric Δ_q based on quantile distances:

$$\Delta_q = \sup_z \Big| \text{EQ}_{\text{synth}}(z) - \text{EQ}_{\text{real}}(z) \Big|, \tag{2}$$

where the supremum is computed over a small number of real values of z equally spaced between 0 and 1. Here EQ denotes empirical quantiles, that is, the inverse of the ECDF function. See illustration in Figure 2. To make the plot easier to interpret, the Y-axis represents z, and the X-axis represents the z-quantiles: for instance, the median if $z = 0.50$. This way, it looks exactly like the graph of a CDF function, and indeed, that's what it represents.

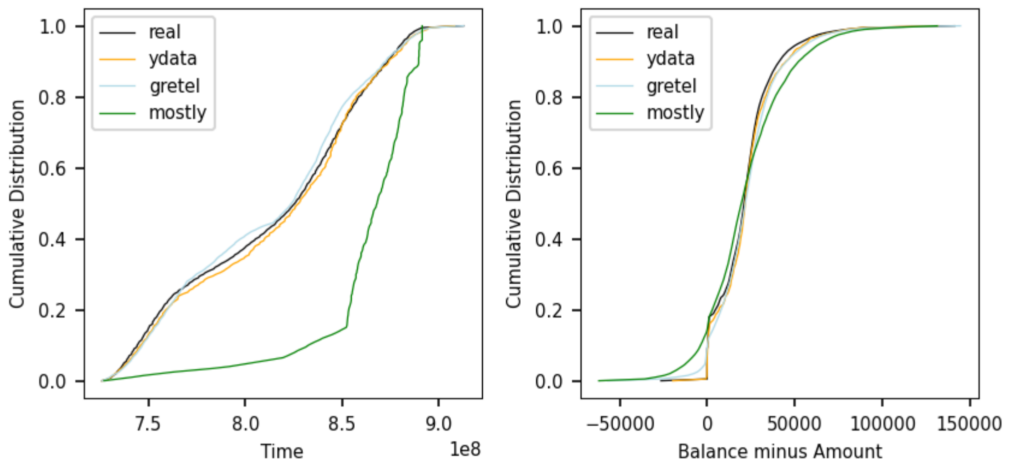

Figure 2: Credit card data, real vs synthetic empirical CDFs

Instead of using the supremum in Formula (2) – that is, the maximum – you may use the mean value: the latter is less sensitive to outliers. The Python code is available here. For the KS distance, see source code here. In both cases (KS and Δ_q), the code in question is used to process the dataset in section 3.1.

Finally, before doing any analysis other than time series, the first step is to shuffle the observations. Also, in addition to compound features (a combination of multiples features such as ratios), it makes sense to add transformed features: in this case, the logarithm of the transaction amount is normally distributed: using a log transform helps with some analyses. For more on multi-table synthetization, see [6]. For a Web API featuring the multivariate KS evaluation, see here. This metric is adjusted for dimension (the number of features) and returns a value between 0 (best fit) and 1 (worst).

2.3 Best practices

In section 2.1, I already discussed various techniques to improve the quality of the results, and to accelerate the synthetization process. The following list highlights best practices that you should follow:

- Split the real data into training and validation sets. Compare the synthetic data to the validation rather than the training set. This is known as the holdout method, or cross-validation. It minimizes overfitting.
- For time data, check whether time series synthetization methods may be more appropriate.
- Normalize the data before synthesizing.
- Look for bottlenecks and how to remove them (see section 2.1).
- If possible, use synthesizers offering replicable solutions. You may run your synthesizer two or three times with different seeds, and keep the best synthetization.
- When using the GenAi-evaluation library, compute the KS distance between the synthetic data and the validation set, and then between the validation and training sets. If both are similar, increase the number of nodes.
- Add compound features that have a useful business interpretation. For instance, if you have (say) transaction amount and balance on credit card as two features in your real data, add the feature "balance minus amount".
- Add labels as an extra feature to the real data. A label is a cluster ID attached to an observation; similar observations are assigned to the same cluster. This trick works well in computer vision problems.
- When fine-tuning hyperparameters, use smart grid search [11] when possible.

For more, see the book "Synthetic Data for Deep Learning" [17]. Model evaluation is further discussed in [1, 18]. For GAN implementations in Python, see [13, 15].

3 Case studies

The case studies discussed in this section deal with real-world business problems. For multi-table synthetization, we compare YData.ai, Gretel, Mostly.ai and when possible, open-source SDV. The generated tables are obtained using the default settings in each case. For model evaluation, we used the methodology outlined in section 2.2. Real data and synthetization have same number of observations. Database diagrams are in section 4.

3.1 Credit card database

The dataset and description are available here. The real data and 3 synthetizations (Ydata.ai, Gretel, Mostly.ai) are available here. It has about 50k credit card transactions in the main table, and 7 satellite tables. The main Jupyter notebook is on GitHub, here. The additional code for ad-hoc analyses is on GitHub, here. It is used to build, test and add custom compound features to the dataset, to check business rules. The "quality score" in Table 4 is the multivariate KS distance averaged over all tables.

In its synthetization, Mostly.ai has a multi-year gap with very few transactions between February 1993 and May 1995, explaining the poor behavior on the left plot in Figure 2. It is also visible in Figure 1. Another issue is generating data outside the observation range: none of the vendors can do it. In some contexts it may be a problem, one you can fix with extrapolated quantiles, see [10]. However in business contexts, it is usually a desirable property: for instance min and max age for insurance policies are respectively 18 and 64. You cannot generate ages outside that range. However, to look good on tests based on the distribution range, for each numerical feature including time, Gretel artificially generates observations that match the observed minima and maxima, causing other issues: see the fourth column in Table 1. The "new balance" feature is not available in the real data. I added it for additional tests. It should always be above zero except in rare instances. The only vendor satisfying this rule is YData: see Table 1. In the same table, the first column is labeled z for consistency with Formula (2).

	Transaction Amount				New Balance			
z	Real	YData.ai	Gretel	Mostly.ai	Real	YData.ai	Gretel	Mostly.ai
0.00	0.50	1.40	0.50	3	-26,338	-19,605	-56,689	-61,841
0.05	22	23	0.50	22	0	0	-1,112	-11,806
0.10	52	41	79	55	200	300	103	-3,749
0.20	138	115	247	133	4,723	6,773	7,973	2,908
0.30	480	300	881	347	13,308	14,598	14,012	10,250
0.40	1,360	900	2,567	1,168	17,888	19,081	18,304	15,009
0.50	3,452	3,019	4,361	3,149	21,025	21,688	21,801	19,525
0.60	5,685	4,500	5,916	5,086	23,666	24,038	25,213	24,342
0.70	9,010	6,963	8,958	8,447	26,630	27,387	28,910	30,888
0.80	15,000	14,387	13,811	14,311	31,283	33,135	34,180	38,800
0.90	23,227	23,041	23,218	21,341	41,148	44,942	45,328	52,206
0.95	32,498	31,624	34,598	30,364	51,962	56,146	57,171	64,430
1.00	74,812	69,148	74,812	62,963	141,403	131,829	144,550	131,408

Table 1: Quantiles; "new balance" is "transaction amount" minus "balance"

Tables 2 and 3 offer a different perspective on the synthetizations. The column labeled "Real" corresponds to the real data used to train the 3 synthesizers YData.ai, Gretel and Mostly.ai. In this case, open-source SDV failed, see Table 4 for details. There are 3 main multi-categories in the real data – the first three rows – and everything else is grouped under the "bundle" category. The multi-categories span across multiple features.

Multi-category	Real	YData.ai	Gretel	Mostly.ai
VYDAJ-VYBE	0.175	0.189	0.173	0.029
PRIJEM-VKL	0.394	0.323	0.358	0.298
PRIJEM-nan	0.245	0.286	0.186	0.086
bundle	0.186	0.202	0.283	0.586

Table 2: Proportion of transactions per multi-category

A large number of missing values are present, but they do not cause problems. YData.ai is the only vendor that does not produce category combinations not found in the real data. To the contrary, Mostly.ai massively violates this rule, resulting in a random distribution of transaction amounts per multi-category, far from the real distribution.

Multi-category	Real	YData.ai	Gretel	Mostly.ai
VYDAJ-VYBE	11,168	10,458	10,355	7,059
PRIJEM-VKL	11,280	10,268	11,613	7,920
PRIJEM-nan	96	99	372	6,749
bundle	8,694	11,071	7,877	7,435

Table 3: Mean transaction amount per multi-category

Table 4: Credit card database – additional findings

Vendor	Remarks
YData.ai	Data Integrity is maintained. Run time: 7 mins. Quality score: 0.16 (best).
Gretel	Data Integrity is maintained. Run time: 2 mins. Quality score: 0.28.
Mostly.ai	Data Integrity is maintained. Run time: 40 mins. Quality score: 0.31 (worst).
SDV	This run did not complete after over 3 hours. It generated 27K+ columns, with the following warning: Using the HMASynthesizer on this metadata schema is not recommended. SDV recommends simplifying the metadata schema using 'sdv.utils.poc.simplify_schema'. After this change, it still run into several issues: columns being removed to simplify the schema, and problem with dates even after using the recommended date/time format.

3.2 AdventureWorks database

The dataset, posted by Microsoft, is available here. It is rather small yet particularly challenging. Only YData.ai was able to successfully complete the synthetization using the default settings: it took 11 minutes, the integrity is fully preserved, and the YData evaluation score (KS distance) is 0.28, compared to 0.16 (a better result) in case study 3.1. We worked on the purchasing module, consisting of 5 database tables. The entire database has dozens of tables.

Smaller datasets – in this case, 9000 records in the main table – are more difficult to synthesize. Two of the features are timestamps, making it more difficult, especially for SDV. You can find the real data and the synthetizations, here. The problems faced with Gretel, Mostly.ai, and SDV, are summarized in Table 5.

Table 5: AdventureWorks database – additional findings

Vendor	Remarks
YData.ai	Successful completion in 11 minutes, database integrity preserved.
Gretel	Generation fails because Gretel is not able to map to the appropriate data types, causing the following error: `numeric field overflow`.
Mostly.ai	It could generate only 5 records for the two most important tables, which were supposed to have 9000 and 4000 records.
SDV	SDV does not support composite keys. It tries to create large number of columns for certain tables, resulting in the following warning: HMA will generate a large number of columns. (4055 columns). The same problem occurred in case 3.1, resulting in hours of computing time and eventually, failure. The metadata schema in SDV can be challenging.

3.3 MovieLens database

We decided to include this dataset because SDV is able to synthesize it in little time and without issue. Indeed, all vendors are able to handle it with the default settings, but it is less representative of a real-world business case. We tested the preservation of database integrity in the synthetic data, and the speed: see summary in Table 6. The main table has 100k records. All vendors and open-source SDV pass the data integrity test. The real data and the 4 synthetizations (SDV, YData.ai, Gretel, Mostly.ai) are available here. The original database with documentation, is posted here.

Table 6: MovieLens database – additional findings

Vendor	Remarks
YData.ai	Ease of Use (UI): Professional and aesthetically pleasing. User-friendly workflow. The error messages are difficult to decode and the user may require Ydata support to resolve or get a workaround. Ease of Use (Through Python): Many alternatives & different documentation/code on Github. Overwhelming for a developer not accustomed to these libraries. Time Taken: All steps completed in 5 mins.
Gretel	Python libraries and documentation are initially confusing. It gets easier once we understand the sample code provided by Gretel. Concept of Workflow: Slight learning curve to define model options, source & synth DB connection. Ease of Use (Through UI): Straightforward and recommended for non-developers. Time Taken: About 35-40 mins total.
Mostly.ai	Ease of Use (Python): Sample code and documentation are confusing. No option to connect directly to the DB tables. Instead, use CSV files and manually define file locations, primary & foreign key constraints in the code. Yet, we still need to access the UI to download the synth data files. Ease of Use (Through UI): Option to connect directly to the source DB, but UI does not recognize the primary & foreign key constraints (must be manually defined). Error handling is not user friendly. Otherwise UI is straightforward. Time Taken: 5 mins.
SDV	Ease of Use (Python): Libraries are easy to use and documentation is overall unambiguous. Option to connect directly to source DB is available only in Enterprise version. The library identifies primary keys & foreign keys relationship between multiple tables easily if the related column names match between the tables. Errors: Manually defining foreign keys in case of different column names in related columns throws type error. Ease of Use (Through UI): NA. Time Taken: All steps completed in 10 mins.

4 Results and conclusion

I summarized the results of the various tests in Table 7. Despite using the default settings, it took several hours to understand and work with some of the libraries. In several instances, the synthetization failed after multiple trials and fixes, especially with SDV but also with Gretel and Mostly.ai. Both the date columns and metadata contributed to the challenges that we faced with SDV.

	YData.ai	Gretel	Mostly.ai	SDV
Ease of use	⋆⋆⋆⋆⋆	⋆⋆⋆⋆	⋆⋆⋆	⋆
Database integrety	⋆⋆⋆⋆⋆	⋆⋆⋆⋆	⋆⋆⋆⋆	⋆
Speed	⋆⋆⋆⋆	⋆⋆	⋆⋆⋆⋆⋆	⋆
Business rules	⋆⋆⋆⋆⋆	⋆⋆⋆	⋆⋆⋆	⋆
Faithfulness	⋆⋆⋆⋆	⋆⋆⋆⋆	⋆⋆	⋆⋆
Date columns	⋆⋆⋆⋆⋆	⋆⋆⋆⋆	⋆⋆	⋆
Category integrity	⋆⋆⋆⋆⋆	⋆⋆⋆	⋆⋆	⋆
Success rate	⋆⋆⋆⋆⋆	⋆⋆⋆	⋆⋆⋆	⋆
Overall rating	⋆⋆⋆⋆⋆	⋆⋆⋆⋆	⋆⋆⋆	⋆

Table 7: Multi-table synthetization, global vendor ratings

Compared to our analysis on single-tables from a year ago, Gretel significantly improved and now outperforms Mostly.ai. Also, YData.ai solidified its first position, now emerging as the undisputed winner. SDV still remains the laggard. Note that the free version of Gretel is limited to rather small datasets.

In Table 7, "faithfulness" is the ability to reproduce the correct multivariate distribution attached to the real data, in each database table. The "success rate" is low for synthesizers that fail on several datasets, using the default parameters. A low rating for "business rules" means several violations to implicit rules such as new balance being negative after a credit card transaction. Finally, "category integrity" is low when the synthetic data contains a large chunk of records attached to category combinations not found in the real data, while "date columns" is the ability to generate timestamps with properties mimicking those found in the real data.

Rajiv Iyer is the Lead Developer at GenAItechLab.com, and formerly Principal Member of Technical Staff at Oracle India. Vincent Granville is Chief AI Scientist at GenAItechLab.com, author (Elsevier, Wiley), formerly consultant with Wells Fargo, Visa, Microsoft, eBay, and NBC. Currently based in Seattle, Vincent did his postdoc at University of Cambridge.

Appendix: database diagrams

Here, we included the database diagrams corresponding to the three case studies. Each database has one or two main tables, and several child tables. In the case of AdventureWorks, the total number of tables is above 60, but we only included those used in the synthetization. For the full schema, see here.

Zoom in on any diagram to get a high resolution.

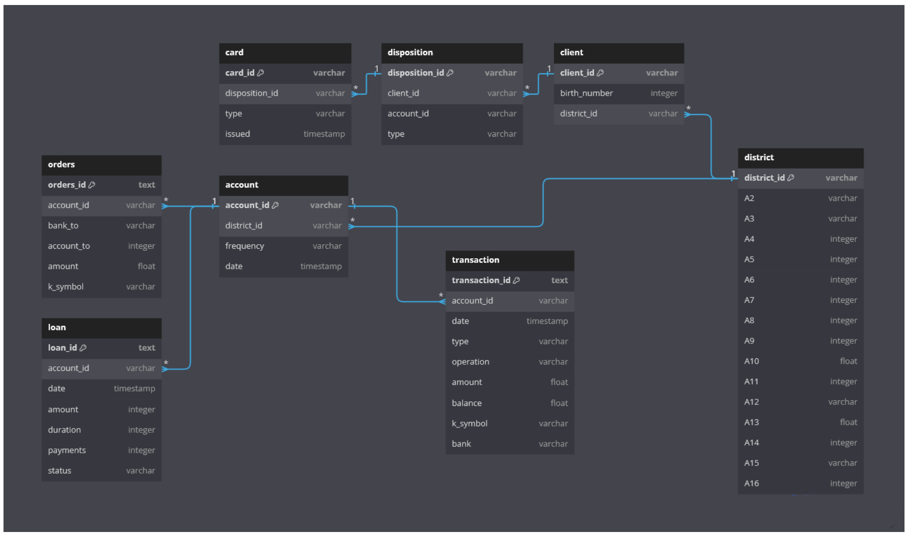

Figure 3: Credit Card Transactions, case 3.1

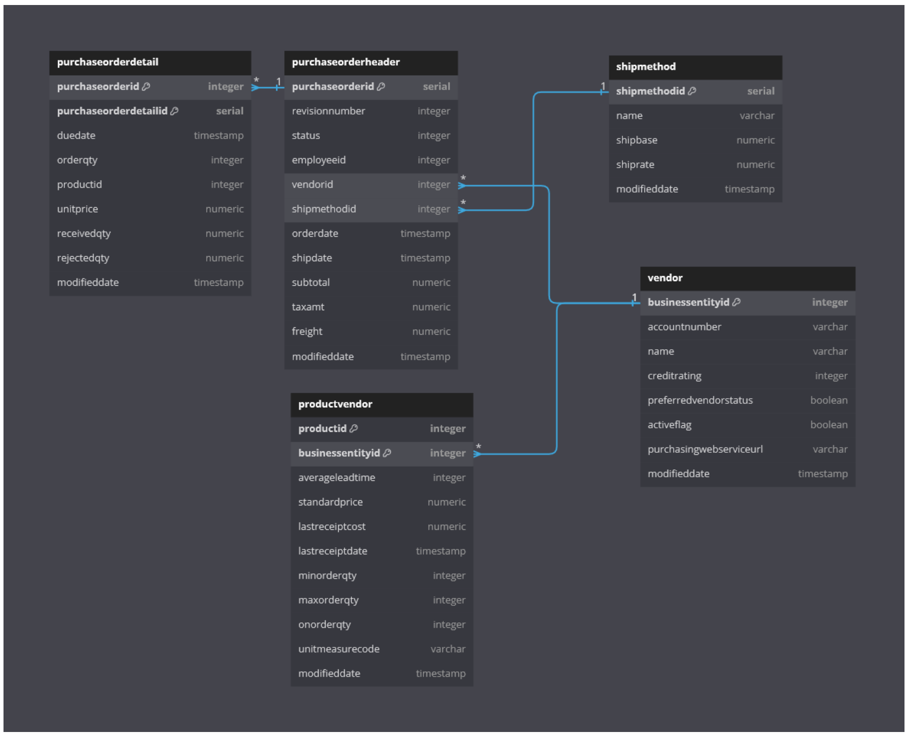

Figure 4: AdventureWorks, case 3.2

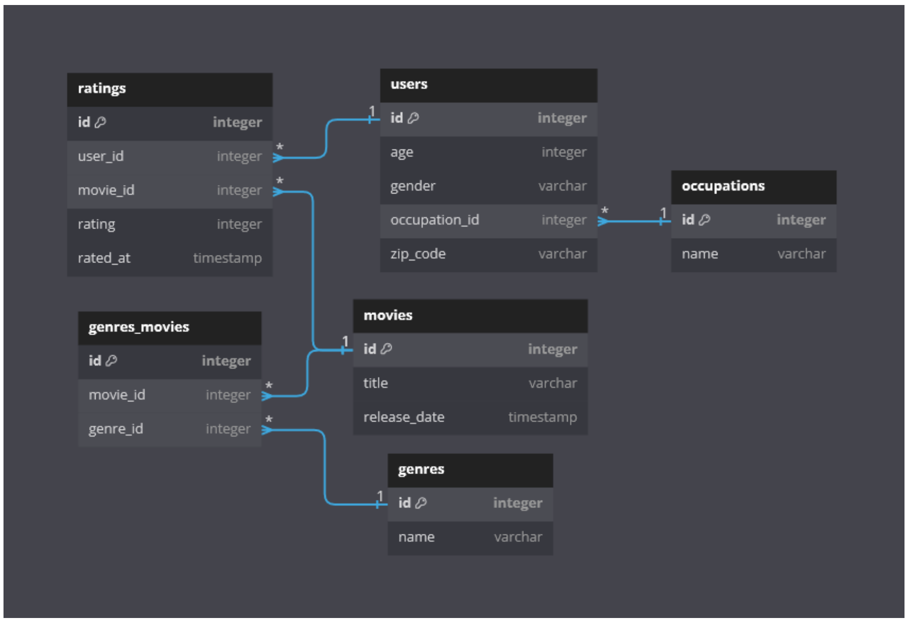

Figure 5: MovieLens, case 3.3

References

[1] Ahmed Alaa et al. How faithful is your synthetic data? sample-level metrics for evaluating and auditing generative models. *Proceedings of the 39 th International Conference on Machine Learning*, pages 1–17, 2023. [Link]. 4

[2] Marc G. Bellemare et al. The Cramer distance as a solution to biased Wasserstein gradients. *Preprint*, pages 1–20, 2017. arXiv:1705.10743 [Link]. 2

[3] Ali Borji. Pros and cons of GAN evaluation measures: New developments. *Preprint*, pages 1–35, 2021. arXiv:2103.09396 [Link]. 2

[4] Khaled Emam, Lucy Mosquera, and Richard Hoptroff. *Practical Synthetic Data Generation.* O'Reilly, 2020. 2

[5] Alvaro Figueira and Bruno Vaz. Survey on synthetic data generation, evaluation methods and GANs. *New Insights in Machine Learning and Deep Neural Networks*, 2022. MDPI [Link]. 2

[6] Paul Francis. A comparison of SynDiffix multi-table versus single-table synthetic data. *Preprint*, pages 1–15, 2024. arXiv:2403.08463 [Link]. 4

[7] Vincent Granville. Feature clustering: A simple solution to many machine learning problems. *Preprint*, pages 1–6, 2023. MLTechniques.com [Link]. 2

[8] Vincent Granville. Generative AI: Synthetic data vendor comparison and benchmarking best practices. *Preprint*, pages 1–13, 2023. MLTechniques.com [Link]. 1

[9] Vincent Granville. Massively speed-up your learning algorithm, with stochastic thinning. *Preprint*, pages 1–13, 2023. MLTechniques.com [Link]. 2

[10] Vincent Granville. Sampling outside the observation range with quantile convolution. *Preprint*, pages 1–7, 2023. MLTechniques.com [Link]. 5

[11] Vincent Granville. Smart grid search for faster hyperparameter tuning. *Preprint*, pages 1–8, 2023. MLTechniques.com [Link]. 4

[12] Vincent Granville. *State of the Art in GenAI & LLMs – Creative Projects, with Solutions.* MLTechniques.com, 2024. [Link]. 2

[13] Vincent Granville. *Synthetic Data and Generative AI.* Elsevier, 2024. [Link]. 4

[14] Elisabeth Griesbauer. *Vine Copula Based Synthetic Data Generation for Classification.* 2022. Master Thesis, Technical University of Munich [Link]. 2

[15] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow.* O'Reilly, third edition, 2023. 4

[16] Mario Lucic et al. Are GANs created equal? a large-scale study. *Proc. NeurIPS Conference*, pages 1–10, 2018. [Link]. 2

[17] Sergey I. Nikolenko. *Synthetic Data for Deep Learning*. Springer, 2021. 4

[18] Joshua Snoke et al. General and specific utility measures for synthetic data. *Journal of the Royal Statistical Society Series A*, 181:663–688, 2018. arXiv:1604.06651 [Link]. 4

[19] Chang Su, Linglin Wei, and Xianzhong Xie. Churn prediction in telecommunications industry based on conditional Wasserstein GAN. *IEEE International Conference on High Performance Computing, Data, and Analytics*, pages 186–191, 2022. IEEE HiPC 2022 [Link]. 2

[20] Lei Xu and Kalyan Veeramachaneni. Synthesizing tabular data using generative adversarial networks. *Preprint*, pages 1–12, 2018. arXiv:1811.11264 [Link]. 2

[21] Ruonan Yu, Songhua Liu, and Xinchao Wang. Dataset distillation: A comprehensive review. *Preprint*, pages 1–23, 2022. Submitted to IEEE PAMI [Link]. 2