
MAKE YOUR OWN DATASET

December 7, 2023

Anastasia Drakou - 2022202304006

Anna Koutougera - 2022202304012

Despoina Angeliki Moisidou - 2022202304016

Contents

1	Introduction	3
2	Data Schema	3
2.1	Relationships Between Tables	6
3	Implementation Steps for generating synthetic data	7
4	Identify subgroups/classes per table (if applicable)	9
5	Noise Insertion Processes	10
6	Missing Value Insertion Processes	14
7	Set of Questions to Be Answered Using the Final Dataset	15
8	Conclusion	16
9	Appendix	16

1 Introduction

The domain selected for this project involves simulating in-store purchases at places like 'Kotsovolos' within the retail/e-commerce sector. The objective is to construct a dataset that mirrors how such organizations might manage and enrich their purchase records. By introducing supplementary information taken from online transactions, it is possible to monitor consumer preferences, anticipate demand and avoid cases of out of stock, but also gain insights of customer interactions, combining the offline and online spheres.

We chose the domain of sales and e-commerce in order to understand and handle our data. The business or practice of selling items is known as the sales domain. Commercial transactions carried out electronically over the Internet are referred to as e-commerce. In this instance, we decided to create a dataset using information from a website selling electronic products and furniture in order to simulate real time in-store purchases.

Synthetic data should always be reliable at mimicking the original data value. It's crucial that the responses and insights we need to measure with synthetic are closely aligned with what you'd expect from real data. That is the reason why we have opted to introduce all types of noise into both our final generated dataset but also the individual tables that helped create it, something that will be explained in detail below.

2 Data Schema

In this simulated retail environment, a dataset has been created that emulates the operations of a store similar to 'Kotsovolos.' To enrich this dataset and mirror the complexity of real-world scenarios, we've integrated information from various tables.

Table **Users** : Contains information about a variety of user profiles and the personal information that they have voluntarily shared during their store sign-up process. This data provides insights into the various preferences and details that users choose to disclose with the store.

- *user_id* : Unique identifier for each user
- *username* : User's username.
- *name* : User's full name.
- *sex* : User's gender.

- *address* : User's address.
- *mail* : User's email address.
- *birthdate* : User's date of birth.
- *phone* : User's contact phone number.
- *generation* : User's generation he belongs to based on his birthdate (**classification**).

Table **Products** : Contains information about detailed insights into each product's specifications, features and pricing, tailored to match the corresponding categories which for the purposes of this project are limited to Electronics, Accessories, Kitchen, and Furniture.

- *product_id* : Unique identifier for each product.
- *product_name* : Name of the product.
- *brand* : Brand of the product.
- *url* : URL associated with the product.
- *category_name* : Category to which the product belongs (**classification**).
- *price* : Price of the product.
- *currency_code* : Currency in which the price is represented.
- *warranty* : Warranty information associated with the product.
- *price_category* : Classifies prices into distinct categories based on specific predefined ranges (**classification**)

Table **Stores**: Contains store-specific details, focusing solely on addresses associated with various store locations.

- *store_id* : Unique identifier for each store.
- *store_address* : Address of the store.

Table **Reviews** : Contains details pertaining to user reviews for various products. This table serves as a resource for evaluating user feedback and product recommendations within the dataset.

- *review_id* : Unique identifier for each review.
- *user_id* : Identifier linking to the Users table, representing the user who wrote the review.
- *product_id* : Identifier linking to the Products table, representing the reviewed product.
- *submitted_at* : Timestamp indicating when the review was submitted.
- *modified_at* : Timestamp indicating when the review was last modified.
- *rating* : Numeric rating given in the review.
- *rating_range* : Range within which the rating falls.
- *is_recommended* : Indicator specifying whether the product was recommended by the user or not based on the rating it was given (classification)

Table **Orders** : It basically captures essential transactional data. Each order is uniquely identified by an *order_id* and associated with a *user_id*, *product_id*, and *store_id*, detailing the customer, the purchased product, and the originating store. The final dataset is generated based on the information in the "Orders" table, with additional details merged from other tables using the respective IDs from the "Orders" table. This dataset contains user, product, store, review, and order information, enabling a comprehensive analysis of the simulated store's activities.

- *order_id* : Unique identifier for each order.
- *user_id* : Identifier linking to the Users table, representing the user who made the order.
- *product_id* : Identifier linking to the Products table, representing the ordered product.
- *store_id* : Identifier linking to the Stores table, representing the store from which the product was ordered.
- *quantity* : Quantity of the product ordered.
- *date* : Timestamp indicating when the order was placed.

2.1 Relationships Between Tables

Table **Users**:

- One user can submit multiple reviews.
- One user can place multiple orders.

Table **Products**:

- One product can have multiple reviews.
- One product can be part of multiple orders.

Table **Stores**:

- One store can have multiple orders.

For a better visualization of the tables explained above, the schema can be found [here](#). Alternatively, a summary of the tables and their relationships is provided below as well.

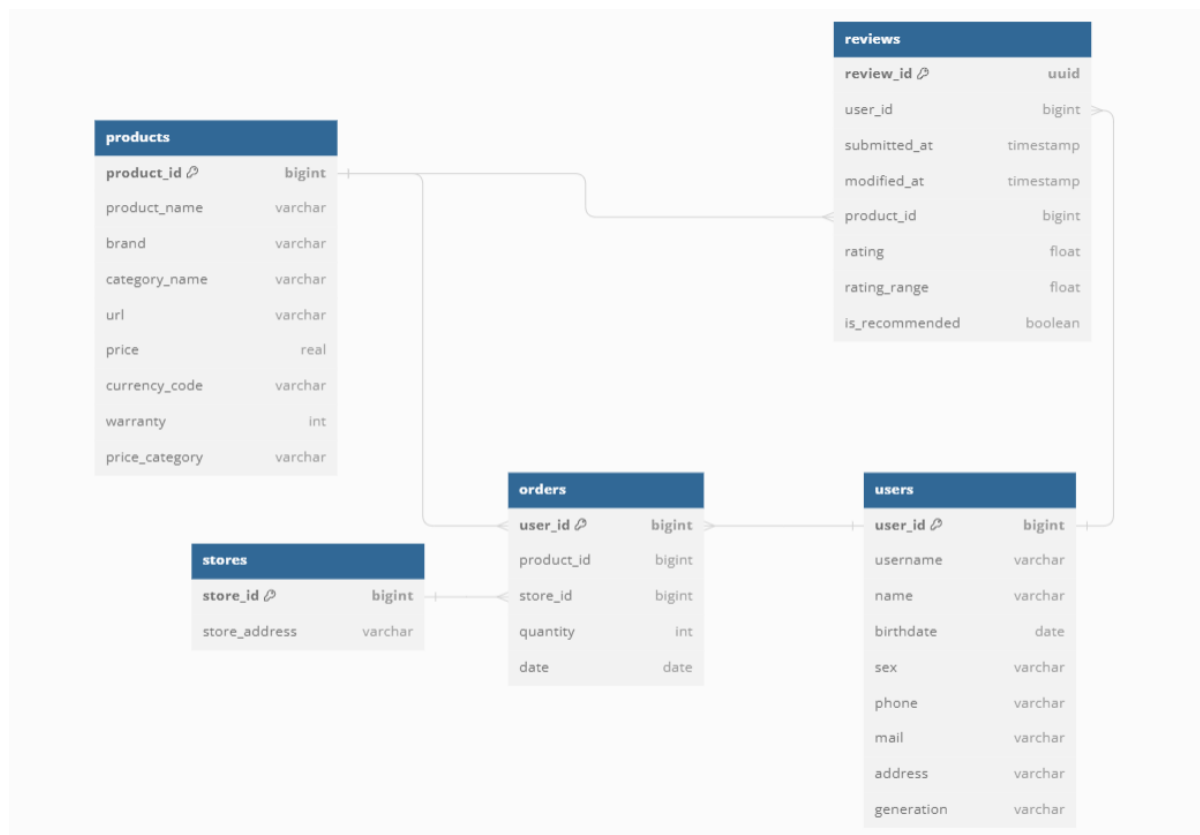


Figure 1: E/R Diagram

3 Implementation Steps for generating synthetic data

The implementation involved a combination of approaches, including utilization of the Faker library for creating synthetic data, random selection of numbers, and the incorporation of predefined lists. For instance, in the "Users" table, the Faker library was used to generate plausible personal information, while the "Products" table had a unique implementation with a mix of random choices and predefined lists.

Table Users : To generate synthetic data for the 'users' table the Faker library was utilized.

user_id		username	name	sex	address	mail	birthdate	phone
0	1	papageorgiou.eustratios	Θεόφιλος-Αγαμέμνων Κωστόπουλος	M	Τουρλωπτής 11,ηTK 399 83 Άρτα	llianakis@hol.gr	1920-11-29	(+30) 6947 629850
1	2	chatzisavvidis.stauros	Φίλιππος Γεωργούλας	M	Καλαμωτού 18,η333 19 Τρίκαλα	kokkinidou.kleopatra@hol.gr	1976-07-26	2020 211 828
2	3	theokleia60	Μένανδρος Κολχούρης	M	Σαββάκη 73,ηTK 41308 Χανιά	kosmas.syvilla@forthnet.gr	1977-04-03	+30 2113 664172
3	4	douvpoulos.chrysovalantios	Ηώ-Παρέσσα Κώτσου	F	Σκουτεράς 2,ηTK 05208 Πάτρα	dfourkioti@googlemail.gr	1997-11-15	+306902266377
4	5	eugenios28	Έρα Μαργιά	F	Λευκοπηγής 156-219,ηTK 11260 Ιωάννινα	argyroula04@forthnet.gr	1992-06-25	+30 2043 674584

Figure 2: This is a sample of how the table looks like at first before introducing noise or classifications.

Table Products : In the creation of synthetic data for the 'products' table, an alternative method was employed, distinct from the Faker library. Specifically, a list of random products, a set of diverse adjectives, and a compilation of random warranty years and categories was generated. These lists were then utilized to dynamically generate product names and assign price ranges based on the respective product categories. This approach allowed us to introduce a varied and realistic set of product entries within the dataset.

	product_id	product_name	brand	url	category_name	price	currency_code	warranty
0	1	Sleek green Keyboard	Elec Tech	http://christodoulakis-varaklis.gr/	Accessories	91.23	EUR	1
1	2	Incredible blue Desk	Digital Plus	https://litsiou-papagiannopoulos.com/	Furniture	796.77	EUR	3
2	3	Small red Refrigerator	Innovex	https://agkyropoulos-agrafioti.com/	Kitchen	950.76	EUR	3
3	4	Kitchen	Xceltron	https://zolotas.com/	Kitchen	879.48	EUR	3
4	5	Refrigerator	Evo Electronics	http://www.kyriatsoulis-romaiou.com/	Kitchen	993.81	EUR	3

Figure 3: This is a sample of how the table looks like at first before introducing some noise

Table Stores : To generate synthetic data for the 'stores' table, we have opted to utilize the Faker library.

	store_id	store_address
0	1	Νικολοπούλου 45,\nTK 573 72 Μυτιλήνη
1	2	Περάτη 4,\nTK 414 80 Τρίκαλα
2	3	Νεοκάστρου 57,\nTK 87959 Δράμα
3	4	Μυκηνών 60,\nTK 30339 Πάτρα
4	5	Γουμένισσας 2,\n85271 Ηράκλειο

Figure 4: This is a sample of how the table looks like at first before introducing some noise.

Table **Reviews** : To generate synthetic data for the 'reviews' table, we have opted to utilize the Faker library. A logical sequence was ensured in the *modified_at* and *submitted_at* columns, where one always follows the other. Additionally, for each review, a random selection process was implemented for both the user and product associated with the review, introducing variability and authenticity into the dataset.

	review_id	user_id	product_id	submitted_at	modified_at	rating	rating_range
0	1	9	173	2021-11-07	2022-09-01	2	5
1	2	91	64	2021-01-30	2022-05-25	5	5
2	3	41	86	2022-07-07	2023-07-27	2	5
3	4	98	172	2023-11-05	2023-11-13	1	5
4	5	162	66	2021-02-15	2021-04-14	5	5

Figure 5: This is a sample of how the table looks like at first before introducing some noise.

Table **Orders**: To generate synthetic data for the 'orders' table, we have opted to utilize the Faker library. For each order, a random selection process was implemented for both the user and product associated with the order.

	order_id	user_id	product_id	store_id	quantity	date
0	1	11	42	19	2	2023-02-05
1	2	3	30	11	2	2023-05-22
2	3	14	48	5	2	2022-09-08
3	4	40	18	43	1	2022-06-18
4	5	12	3	38	2	2023-08-17

Figure 6: This is a sample of how the table looks like at first before introducing some noise.

Final dataset : In order to create the final dataset, the unique identifiers from the 'Orders' table were utilized to extract additional information from their corresponding tables. Specifically, within the 'Reviews' table, a grouping mechanism based on products was implemented. This facilitated the extraction of valuable metrics such as the number of reviews per product, the average rating for each product, and the recommendation status. This process ensured that the dataset not only comprised individual entries but also provided useful insights aggregated at the product level. Due to the size of this final dataset, a preview can be provided by taking a look at the excel automatically generated after running the Jupyter notebook, called *final_dataset.xlsx*.

4 Identify subgroups/classes per table (if applicable)

For almost every table mentioned above, each column identified as a classification was systematically created through a subsequent process. Existing columns were used, and a distinct set of rules was applied to create these classifications. The detailed explanations for each classification are outlined below:

Table Users : A column named *generation* was introduced to categorize users according to their birthdate into distinct generational groups. The categories include: 'The Greatest Generation (1901-1924)', 'The Silent Generation (1925-1945)', 'Baby Boomer (1946-1964)', 'Generation X (1965-1979)', 'Millennial (1980-1994)', 'Generation Z (1995-2012)', and 'Gen Alpha (2013-2025)'. This classification allows for a better understanding of user demographics based on generational characteristics.

Table Products : Two distinct types of classifications have been implemented here, *price_category* and *category_name*. The *category_name* classification was pre-defined and manually assigned. Initially, we had a list of products, and each product was categorized

into its respective category using a predefined dictionary. This classification was carried out manually, with the selection of products being done randomly.

The second classification, *price_category*, involves categorizing products into different price classes such as 'budget', 'mid-range', and 'expensive'. This classification is based on specific price ranges, allowing for a more detailed analysis of product pricing within the dataset. For example, if a price of a product is between the ranges of [0,500], then it would be classified as budget.

Table Reviews : A column named *is_recommended* was introduced to categorize products based on the rating of each of their review. If a review has a rating equal to or above 3, it is categorized as recommended for the associated product. This approach allows us to distinguish between positive and negative reviews, providing valuable insights into the user feedback for various products in the dataset.

5 Noise Insertion Processes

After generating our final dataset we try to make it more realistic by using noise insertions, such as :

Spelling mistakes : It is a common human mistake to insert wrongly written strings.

Imbalanced data : Skewed class proportions when the amount of data for each class varies.

Outliers : Out of range values.

Inconsistent data : Contradicting or non-conformant to standards, sometimes we use to write the same data with a different way , this can cause problems to our dataset.

Classification inaccuracies : Items in the incorrect class.

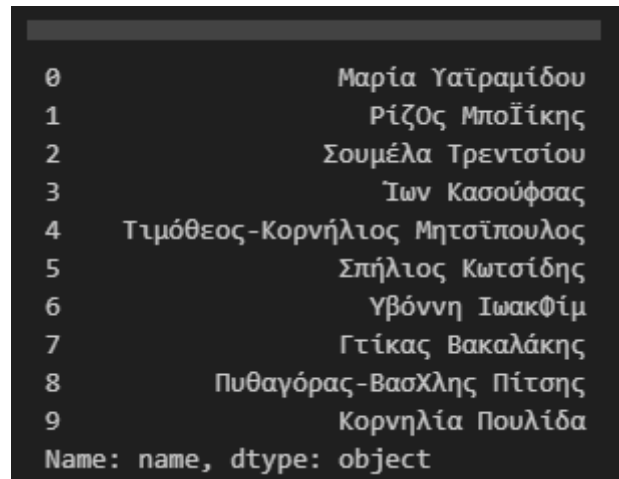
Duplicates : Identical item exists twice causing problems/overlaps.

Missing values : In our case, it frequently occurs that a customer writes an application without filling out all the fields.

We regenerate each table and the final dataset to add noise. The detailed explanations for each noise are outlined below:

Table 'Users' – Spelling mistakes in column 'name': We use the column 'name' to add spelling mistakes in 40% cases of names considering that someone providing his name may not give it correctly. For example, instead of 'Μαρία' we may get 'Ιαρία'. This function created can be used in all types of strings to generate noise if each word is separated by space. It

could also be applied to create spelling mistakes in other columns as well like addresses or product names if applied in the corresponding column.

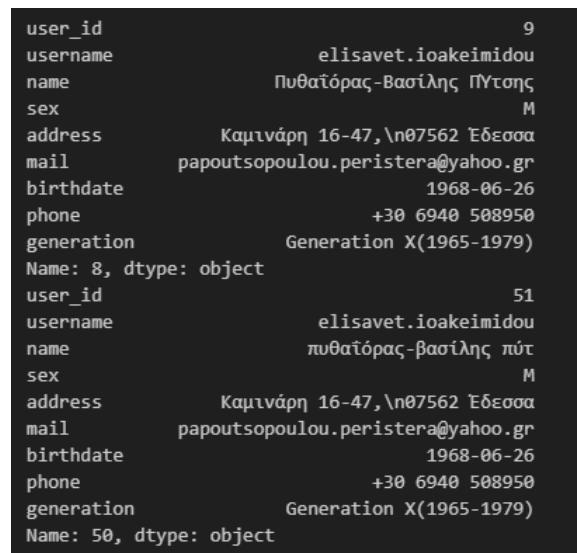


0	Μαρία Υαϊραμίδου
1	Ρίζος Μποϊίκης
2	Σουμέλα Τρεντσίου
3	Ίων Κασούφσας
4	Τιμόθεος-Κορνήλιος Μητσϊπουλος
5	Σπήλιος Κωτσίδης
6	Υβόννη ΙωακΦίμ
7	Γτίκας Βακαλάκης
8	Πυθαγόρας-ΒασΧλης Πίτσης
9	Κορνηλία Πουλίδα
Name: name, dtype: object	

Figure 7: This is a sample of how the table looks like after introducing some noise.

Table 'Users' – Duplicate entries due to spelling mistakes in 'names': In case that a mistake occurs in the user's name, it is possible for multiple profiles to be created for the same user due to potential data entry errors. This may result in duplicates in the 'users' table. Dataset size before is (50, 9)

Dataset size after is (51, 9). For example:



user_id	9
username	elisavet.ioakeimidou
name	Πυθαϊόρας-Βασίλης ΠΥτσης
sex	M
address	Καμινάρη 16-47, \n07562 Έδεσσα
mail	papoutsopoulou.peristera@yahoo.gr
birthdate	1968-06-26
phone	+30 6940 508950
generation	Generation X(1965-1979)
Name: 8, dtype: object	
user_id	51
username	elisavet.ioakeimidou
name	πυθαϊόρας-βασίλης πύτ
sex	M
address	Καμινάρη 16-47, \n07562 Έδεσσα
mail	papoutsopoulou.peristera@yahoo.gr
birthdate	1968-06-26
phone	+30 6940 508950
generation	Generation X(1965-1979)
Name: 50, dtype: object	

Figure 8: This is a sample of how the table looks like after introducing some noise.

Table 'Products' - Imbalanced data in category of products : Each product is systematically classified into distinct categories, namely : 'Electronics,' 'Accessories,' 'Kitchen,' and 'Furniture.' By looking the number of products that are in each category, we can observe that there are : 8 products in 'Electronics', 8 products in 'Accessories', 2 products in 'Kitchen', and 4 products in 'Furniture'. As a result, having imbalanced data. This discrepancy arises from the nature of the merchandise offered by the stores, wherein there is a greater inventory of products falling within the 'Electronics' and 'Accessories' categories compared to those categorized under 'Kitchen' and 'Furniture.'

```
categories= {
    'Laptop': "Electronics", 'Computer': "Electronics", 'Scooter': "Electronics", 'Tablet': "Electronics", 'Cameras': "Electronics",
    'Printer': "Electronics", 'Drone': "Electronics", 'TV': "Electronics",
    'Earbuds': "Accessories", 'Mouse': "Accessories",
    'Keyboard': "Accessories", 'Phone case': "Accessories", 'Watch': "Accessories", 'Monitor': "Accessories", 'Soundbar': "Accessories",
    'Power_bank': "Accessories",
    'Refrigerator': "Kitchen", 'Kitchen': "Kitchen",
    'Chair': "Furniture", 'Desk': "Furniture", 'Bed': "Furniture", 'Sofa': "Furniture",
}
```

Figure 9: This is a sample of how the products are categorized. - Imbalanced data

Table 'Final Dataset' – Outliers/out of range values in column 'price' : Our approach involves utilizing the 'price' column to randomly select roughly 5% of the data. We then introduce outliers by allocating either 5 or 10,000 to these selected prices. Using the same logic, we could also add outliers in the column age.

	product_name	price	price_category
2	Gorgeous black Soundbar	5	budget
13	Soundbar	5	budget
75	Fantastic black Bed	5	budget
81	Soundbar	10000	budget
93	Scooter	10000	premium
121	Refined gold Earbuds	5	budget
123	Practical black Tablet	10000	midrange
132	For repair red Drone	10000	premium
163	Fantastic black Bed	5	budget
198	Laptop	10000	premium

Figure 10: This is a sample of how the table looks like after introducing some noise.

Table 'Final Dataset' – Wrong date format in column 'date': To diversify the date for-

mats within the column, we generated a date column using various formats from a specified list. This adds variety to the dataset, and examples of date formats could include 'YYYY-MM-DD,' 'DD/MM/YYYY,' and 'MM-DD-YYYY,' among others.

0	Sun, 05 Feb 2023 00:00:00
1	2022-04-06
2	2022-06-26
3	05/11/2023
4	2022-11-20
5	2022-07-11
6	2022-07-26
7	2023-04-26
8	02/12/2021
9	2022-11-03
Name: date, dtype: object	

Figure 11: This is a sample of how the table looks like after introducing some noise.

Table 'Final Dataset' – Incorrect data/inaccuracies in the classification of *product_name* in the *category_name* : In our data preprocessing, we introduced a controlled element of noise by deliberately misclassifying 5% of the cases in the 'categories' column. This was achieved by randomly assigning each row a value different from its original category, based on the distinct values present in that column.

	product_name	category_name
757	Refrigerator	Furniture
892	Fantastic black Earbuds	Kitchen
894	Laptop	Kitchen
509	Used gold Soundbar	Electronics
923	For repair gold Mouse	Furniture
196	Mouse	Kitchen
233	Bed	Electronics
305	For repair gold Mouse	Furniture
17	Refined gold Scooter	Furniture
544	Cameras	Kitchen

Figure 12: This is a sample of how the table looks like after introducing some noise.

Table 'Final Dataset' – Duplicate values : To introduce variability and create noise in

our dataset, we opted to generate approximately 2% of duplicate records. This deliberate duplication serves as a controlled element of noise, allowing us to assess the impact of duplicated data on our analytical models.

Dataset size before is (1000, 23)

Dataset size after is (1020, 23). For example:

order_id	quantity	date	username	name	sex	address	birthdate	generation	product_name	...	price_category	currency	store_address	num_of_rev
378	189	5	2021-12-22	fkarampi	ΠιζΟς Μπoλίκης	M	Μολουνδρίνου 617-032\ηTK 29048 Λιβαδιά	1914-05-10	The Greatest Generation(1901-1924)	Handcrafted green Earbuds	...	budget	EUR	Πλατεία Κρωπτίας 35,\η42264 Κόλκις
1000	189	5	2021-12-22	fkarampi	ΠιζΟς Μπoλίκης	M	Μολουνδρίνου 617-032\ηTK 29048 Λιβαδιά	1914-05-10	The Greatest Generation(1901-1924)	Handcrafted green Earbuds	...	budget	EUR	Πλατεία Κρωπτίας 35,\η42264 Κόλκις

2 rows × 23 columns

Figure 13: This is a sample of how the table looks like after introducing some noise.

6 Missing Value Insertion Processes

Missing data are defined as values or data that are not stored (or are not present) in the given dataset for some variable/s. In our case, it frequently occurs that a customer writes an application without filling out all the fields, so we have entry problems and data loss.

Strategies for Inserting Missing Values in our dataset:

Table ‘Users’ – Missing Values in column ‘addresses’ : We’ve opted to introduce variability into our dataset by intentionally introducing missing values in the ‘addresses’ column. This decision stems from the acknowledgment that certain customers, upon signing up, may opt not to disclose their personal information, particularly details about their residence. This function created can be used in all types of columns containing strings to generate missing values if the name of the selected column is specified when calling the function.

```

0      Κλωνίου 57,\n46453 Ηγουμενίτσα
1      Μαλανδρίνου 617-032,\nTK 29048 Λιβαδιά
2
3      Λεωφ. Ελατείας 977-080,\nTK 197 39 Λάρισα
4
5
6      Πεμονίων 88,\n879 25 Λαμία
7      Λεωφόρος Κορώνας 17,\nTK 23529 Αγ. Νικόλαος
8      Καμινάρη 16-47,\n07562 Έδεσσα
9      Μακροχωρίου 759-781,\n32886 Σάμος
Name: address, dtype: object

```

Figure 14: This is a sample of how the table looks like after introducing missing values.

Table **'Reviews'** – **Missing Values in column *modified_at***: For the generation of missing values in this case, the same function for creating missing values in table 'users' was used. The logic here was that sometimes a user may not decide to modify his review later. Of course, the same logic could be applied for creating missing values and for another column to create noise, like rating.

	submitted_at	modified_at
0	2021-11-07	2022-09-01
1	2021-01-30	
2	2022-07-07	
3	2023-11-05	2023-11-13
4	2021-02-15	2021-04-14
5	2021-03-25	2022-08-21
6	2023-06-05	2023-08-09
7	2021-07-05	2021-07-23
8	2021-08-12	2023-08-16
9	2021-09-30	2023-02-24

Figure 15: This is a sample of how the table looks like after introducing missing values.

7 Set of Questions to Be Answered Using the Final Dataset

- **Propensity Model** : The dataset is well-suited for training a propensity model, enabling the assessment of the likelihood that an individual might exhibit a particular purchasing inclination. The dataset can be used in order for the model to predict the probability

of a customer making a repeat purchase or expressing interest in a specific product category. This insight is valuable for targeted marketing strategies and campaigns.

- **Recommendation Systems** : By identifying patterns in customer behaviors, the system can offer personalized product recommendations, enhancing the overall shopping experience and potentially increasing sales.
- **Customer Behavior Classification** : The dataset can be leveraged to classify customers based on their buying behavior.
- **Demand Anticipation and Inventory Management** : Utilizing the dataset for straightforward analyses can aid in anticipating demand and optimizing inventory management. Identifying popular products and stores allows businesses to proactively manage inventory, reducing the likelihood of stockouts and ensuring optimal product availability.

8 Conclusion

In conclusion, the dataset generated above is deemed to be representative of real-world scenarios, given the inherent presence of diverse types of noise in data. The deliberate introduction of noise during the dataset creation process was executed with the aim of achieving a high level of generalization and randomness. This involved initially defining the function of any conceivable noise that could be introduced into a dataset, followed by the systematic incorporation of the most plausible types of noise for each respective data table. These functions are written in python, which can be provided by looking at the Jupyter notebook. For instance, the 'reviews' dataset may contain missing values, while the 'users' dataset could incorporate spelling mistakes. Consequently, the anticipated outcome of our dataset is one characterized by the inclusion of expected and frequent noise patterns reflective of real-world data variability.

9 Appendix

In order to accomplish this project, as it has been said before, we used the Python programming language. We imported libraries such as Pandas, Numpy, Random, and Faker. The tables are exported as Excel files to be legible and understandable for our readers. This domain document was created for further explanation of what we did to make our dataset and why we put noise in it. Everything was implemented in the Jupyter Notebook, which is

provided in the same zip file with explanations for every step.

To illustrate, below is the process of a specific dataset in furtherance of understanding the execution of our own dataset.

```

Generate Table Reviews

print("To generate synthetic data for the 'reviews' table, we have opted to utilize the Faker library.\nWe ensured a logical sequence in the '
def generate_reviews(num,users,products):
    rows = []
    for x in range(1,num+1):
        submitted_at = fake.date_between_dates(date_start = '-3y')
        # To ensure modified_at is after submitted_at
        modified_at = str(fake.date_between_dates(date_start=submitted_at))
        review_id = x
        rating = random.choice(range(1,6))
        row = {
            'review_id':review_id,
            'user_id':random.choice(range(1,num+1)),
            'product_id':random.choice(range(1,num+1)),
            'submitted_at':submitted_at,
            'modified_at':modified_at,
            'rating': rating,
            'rating_range':5,
        }
        rows.append(row)
    return pd.DataFrame(rows)
print("This is a sample of how the table looks like at first before introducing some noise.")
reviews = generate_reviews(200,users,products)
reviews.head(5)

```

Figure 16: This is the code of making the review dataset.

	review_id	user_id	product_id	submitted_at	modified_at	rating	rating_range
0	1	150	6	2021-11-14	2022-09-08	4	5
1	2	189	183	2021-02-06	2022-06-01	4	5
2	3	55	106	2022-07-14	2023-08-03	3	5
3	4	194	30	2023-11-12	2023-11-20	4	5
4	5	21	41	2021-02-22	2021-04-21	3	5

Figure 17: This is the table that was created from the above code.

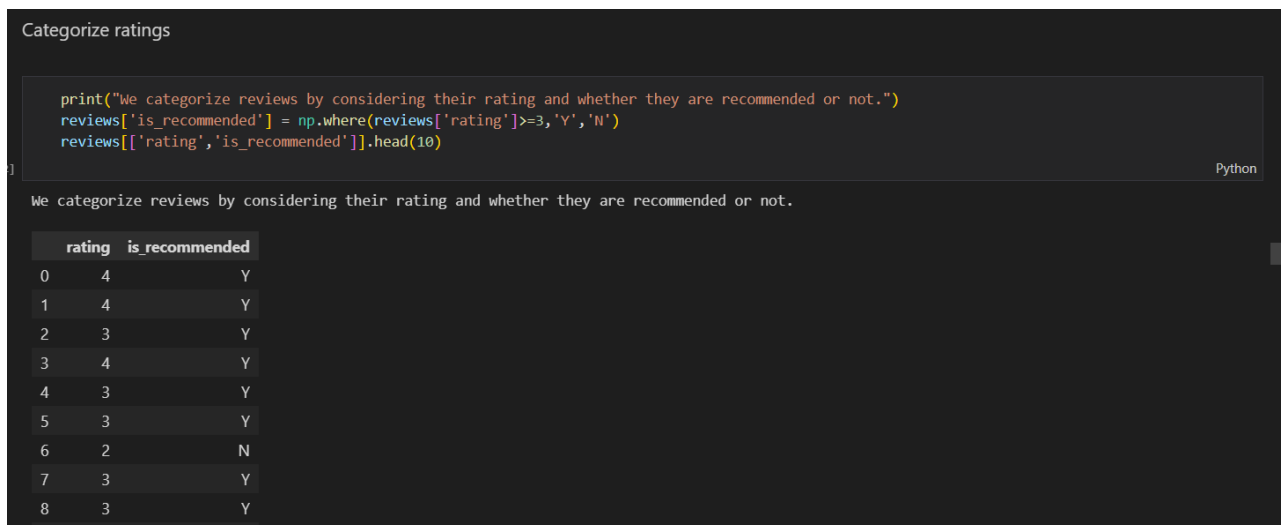


Figure 18: This is a categorization of 'reviews' dataset.

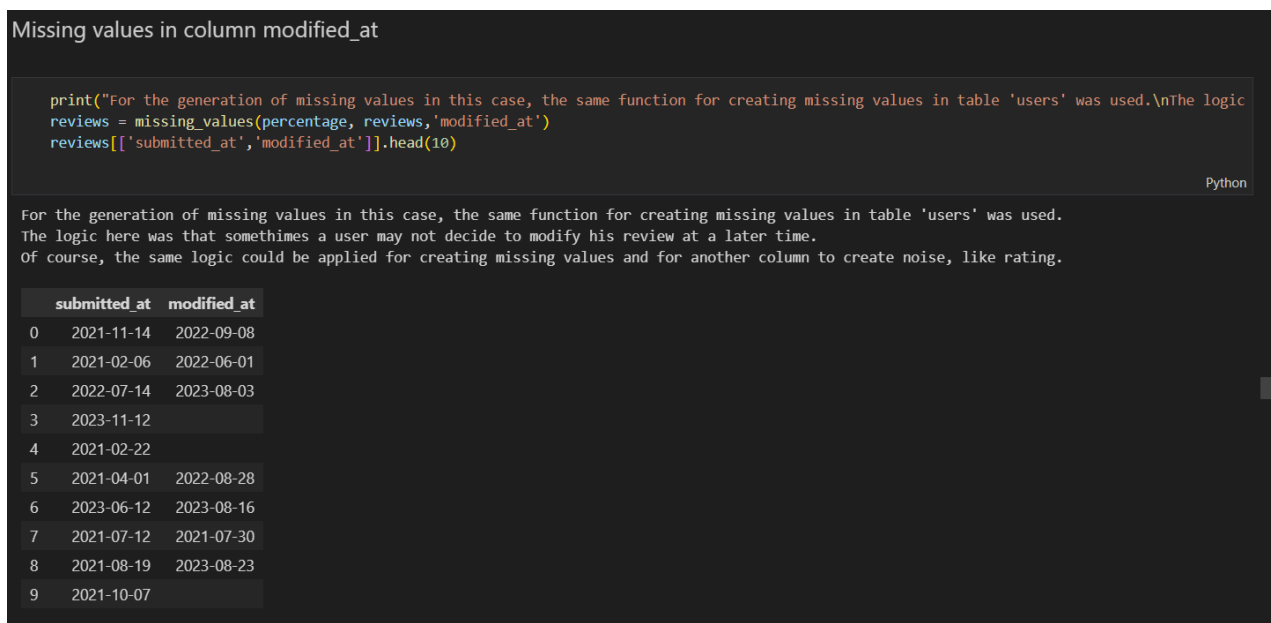


Figure 19: This is a sample of how the table looks like after introducing missing values.

Each dataset has been instantiated in a manner analogous to the aforementioned figures. It is advisable to thoroughly review all associated files in order to gain a comprehensive understanding of the project.