**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики-процессов управления**

**Программа бакалавриата**

**“Большие данные и распределенная цифровая платформа”**

**ОТЧЕТ**

**по лабораторной работе №1**

**по дисциплине «Алгоритмы и структуры данных»**

**на тему «Генерация** **датасета»**

**Вариант – 3**

**Студент гр. 23Б15-пу**

**Крылов А. С.**

**Преподаватель**

**Дик А.Г.**

**Санкт-Петербург**

**2024 г.**

Содержание

[Цель работы 2](#_Toc179512760)

[Описание задачи (формализация задачи) 2](#_Toc179512761)

[Теоретическая часть 3](#_Toc179512762)

[Основные шаги программы 4](#_Toc179512763)

[Блок схема программы 5](#_Toc179512764)

[Описание программы 6](#_Toc179512765)

[Описание функций: 6](#_Toc179512766)

[Функция bank\_card 6](#_Toc179512767)

[Описание классов 8](#_Toc179512768)

[Класс Person 8](#_Toc179512769)

[Класс Vagon 8](#_Toc179512770)

[Класс BankInputUI 9](#_Toc179512771)

[Класс Route 11](#_Toc179512772)

[Связь между классами 12](#_Toc179512773)

[Рекомендации пользователя 13](#_Toc179512774)

[Рекомендации программиста 13](#_Toc179512775)

[Исходный код программы 13](#_Toc179512776)

[Контрольный пример 14](#_Toc179512777)

# **Цель работы**

Целью лабораторной работы является разработка системы генерации датасета для покупки железнодорожных билетов с учетом определенных требований и условий. Датасет должен включать личные данные пассажиров, информацию о поездах, вагонах и местах, а также данные о платежных картах для оплаты билетов.

# **Описание задачи**

Задача состоит в создании датасета для железнодорожных билетов со следующими требованиями:

1. **ФИО**: Славянские имена и фамилии.
2. **Паспортные данные**: Русские паспортные данные с уникальными значениями.
3. **Откуда и куда**: Города в РФ, с различием отправной и конечной точек.
4. **Дата отъезда и приезда**: Случайные даты в пределах года.
5. **Рейс**: Номера поездов в заданных диапазонах.
6. **Выбор вагона и места**: Учет классов вагонов и типов мест.
7. **Стоимость**: В зависимости от длины маршрута и типа вагона.
8. **Карта оплаты**: Генерация карт с возможностью многократного использования с повторением не больше пяти раз и возможностью настраивать вероятность к какому банку и платежной системе принадлежит карта.

# **Теоретическая часть**

Для создания датасета использованы несколько программных модулей:

1. main.py: Основной скрипт для чтения расписания поездов из файла, создания маршрутов и заполнения их пассажирами с помощью класса Route, сохранения данных в Tickets.xlsx
2. utils.py: Основной класс Route для создания маршрута с учетом типа поезда и других входных данных, вспомогательные классы и функции.

Ограничения:

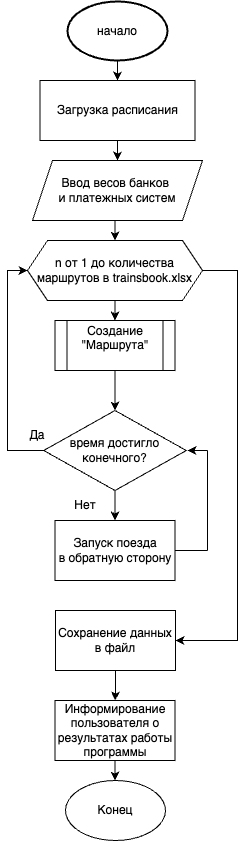
* Количество строк в датасете ограничивается вводом пользователя.
* ФИО пассажиров только славянские.
* Уникальность паспортных данных и их принадлежность к русским.
* Разные точки отправки и прибытия только по территории РФ.
* Рейсы уникальные из определенных номеров, которые бывают до полного возвращения на вокзал.
* Вагоны и места генерируются по заданным типам.
* Веса банков и платежных систем определяются пользователем и должны в сумме давать число больше нуля.
* Логика выбора карт оплаты с ограничением на 5 повторов.

# **Основные шаги программы**

1. Запуск программы (main.py):
2. Пользователь вводит веса банков и платежных систем.
3. Загружается расписание поездов из файла trainsbook.xlsx
4. Генерация данных
   1. Билеты генерируются с использованием данных о расписании, персональных данных и месте в вагоне, а также информации о платежных картах.
      1. Генерация мест в поезде постепенно, с учетом количества мест в вагоне и количества вагонов в составе согласно типу поезда.
      2. Генерация персональных данных: Имя, фамилия, отчество, серия и номер паспорта.
      3. Генерация платежной карты с использованием весов для выбора платежной системы и банка.
      4. Рассчет стоимости билета с учетом дальности поездки и класса места.
5. Данные собираются и записываются в файл Tickets.xlsx .

# **Блок схема программы**

Рис 1. Блок-схема основной программы



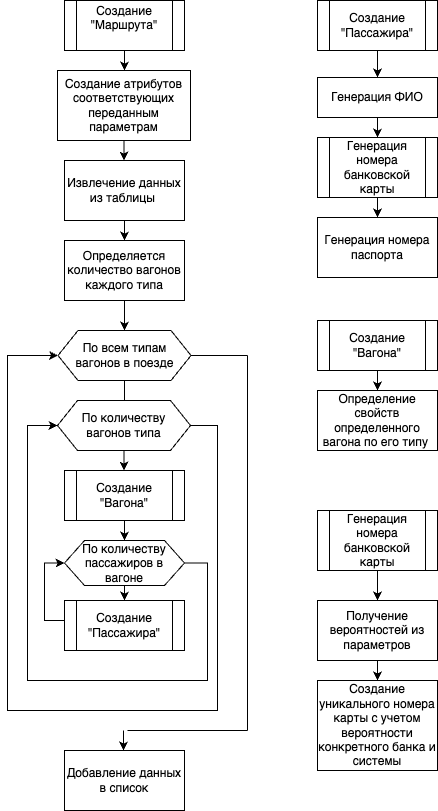


Рис 2. Блок-схема подпрограмм

# **Описание программы**

Программная реализация написана на языке Python 3.11.9 с использованием следующих библиотек: faker, random, tkinter, datetime (с расширением relativedelta), и pandas. Программа организована через модульную структуру с акцентом на генерацию данных для железнодорожных билетов.

## Описание функций:

### **Таблица 1. Функция bank\_card**

|  |  |  |
| --- | --- | --- |
| Параметр | Тип | Описание |
| bank\_system\_probability | list | Список весов для выбора платежной системы (`Mir`, `MasterCard`, `Visa`). По умолчанию равные веса `[1, 1, 1]`. |
| bank\_probability | list | Список весов для выбора банка (`Sber`, `T-Bank`, `VTB`, `Alfa`). По умолчанию равные веса `[1, 1, 1, 1]`. |
| Внутренние переменные | Тип | Описание |
| digit\_list | dict | Словарь, который связывает платежные системы с банками и их префиксами карт. |
| banks | list | Список доступных банков (`Sber`, `T-Bank`, `VTB`, `Alfa`). |
| bank\_systems | list | Список доступных платежных систем (`Mir`, `MasterCard`, `Visa`). |
| bank | str | Случайно выбранный банк с учетом весов `bank\_probability`. |
| bank\_system | str | Случайно выбранная платежная система с учетом весов `bank\_system\_probability`. |
| digit | str | Префикс карты, основанный на выбранных банке и платежной системе. |
| args | dict | Словарь, содержащий `d1` (префикс), `d2`, `d3` и `d4` (случайно сгенерированные 4-значные числа). |
| card\_number | str | Сформированный номер карты в формате `d1 d2 d3 d4`. |
| Возвращаемое значение | Тип | Описание |
| card\_number | str | Случайно сгенерированный номер банковской карты в формате `xxxx xx xxxx xxxx`, основанный на выбранных банке и системе. |

## Описание классов

### **Таблица 2. Класс Person**

|  |  |  |
| --- | --- | --- |
| Параметр | Тип | Описание |
| sys\_prob | list | Список весов для выбора платежной системы (`Mir`, `MasterCard`, `Visa`). |
| bank\_prob | list | Список весов для выбора банка (`Sber`, `T-Bank`, `VTB`, `Alfa`). |
| Атрибуты | Тип | Описание |
| name | str | Случайно сгенерированное имя. Состоит из фамилии, имени и отчества, выбранных из соответствующих списков (мужских или женских). |
| card | str | Номер банковской карты, сгенерированный функцией `bank\_card` с учетом вероятностей `sys\_prob` и `bank\_prob`. |
| passport | str | Случайно сгенерированный номер паспорта в формате `xxxx xxxxxx`, где `xxxx` - это серия, а `xxxxxx` - номер. |

|  |  |
| --- | --- |
| Методы | Описание |
| \_\_init\_\_() | Инициализирует объект `Person`. Выбирает случайное имя и генерирует номер карты и паспорта. |

### 

### **Таблица 3. Класс Vagon**

|  |  |  |
| --- | --- | --- |
| Параметр | Тип | Описание |
| type | str | Тип вагона. Должен соответствовать ключу в словаре `vagon\_data`. |
| Атрибуты | Тип | Описание |
| vagon\_data | dict | Словарь с данными о вагонах, где ключ — это тип вагона, а значение — информация о размере, цене и маркировке вагона. |
| size | int | Количество мест в вагоне. |
| price | int | Стоимость одного места в вагоне в рублях. |
| naming | str | Маркировка вагона (например, `1P`, `2C`, `3Э`). |

### 

|  |  |
| --- | --- |
| Методы | Описание |
| \_\_init\_\_() | Инициализирует объект `Vagon`. |

### **Таблица 3. Класс BankInputUI**

|  |  |  |
| --- | --- | --- |
| Параметр | Тип | Описание |
| master | tk.Tk или tk.Toplevel | Основное окно или контейнер, в котором располагаются виджеты пользовательского интерфейса. |

|  |  |  |
| --- | --- | --- |
| Атрибуты | Тип | Описание |
| title\_label | tk.Label | Метка с текстом, объясняющим пользователю, что требуется ввести вероятности для банков. |
| labels | list | Список строк с названиями банков (`Sber`, `T-Bank`, `VTB`, `Alfa`). |
| entries | list | Список полей ввода (`tk.Entry`) для вероятностей банков. |
| prob\_title\_label | tk.Label | Метка с текстом для ввода вероятностей банковских систем. |
| prob\_labels | list | Список строк с названиями банковских систем (`Mir`, `MasterCard`, `Visa`). |
| prob\_entries | list | Список полей ввода (`tk.Entry`) для вероятностей банковских систем. |
| submit\_button | tk.Button | Кнопка для отправки введенных данных. |
| default\_button | tk.Button | Кнопка для установки значений по умолчанию. |
| data | list | Список введенных вероятностей для банков. |
| probabilities | list | Список введенных вероятностей для банковских систем. |

|  |  |
| --- | --- |
| Методы | Описание |
| \_\_init\_\_() | Инициализирует интерфейс с полями для ввода вероятностей для банков и банковских систем, а также кнопками для отправки данных и установки значений по умолчанию. |
| set\_default() | Устанавливает значение 1 во все поля ввода вероятностей. |
| submit() | Считывает введенные значения вероятностей, сохраняет их в атрибутах `data` и `probabilities`, проверяет корректность ввода и отображает сообщение. |
| submit\_with\_default() | Устанавливает значения по умолчанию и затем отправляет данные. |
| show\_info() | Отображает информационное окно с переданным сообщением и кнопкой для закрытия. Если `button\_text` не указан, окно автоматически закроется через 5 секунд. |
| get\_bank\_prob() | Возвращает список введенных вероятностей для банков. |
| get\_sys\_prob() | Возвращает список введенных вероятностей для банковских систем. |

### **Класс Route**

|  |  |  |
| --- | --- | --- |
| Параметр | Тип данных | Описание |
| number | int | Порядковый номер поезда в словаре train\_dictionary\_df. Используется для извлечения данных по конкретному поезду. |
| train\_dictionary\_df | DataFrame | Таблица (DataFrame), содержащая данные о поездах, включая их номера, типы, станции отправления и назначения, а также время в пути. |
| initial\_time | datetime | Время отправления поезда. Используется для расчета текущего положения поезда и времени прибытия. |
| ticket\_data | list | Список для хранения информации о проданных билетах. Сюда добавляются данные о каждом пассажире, включая ФИО, паспорт, станцию отправления и назначения, и т.д. |
| sys\_prob | float | Вероятность использования определенной банковской системы. |
| bank\_prob | float | Вероятность использования определенного банка банковской карты. |

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание |
| number | int | Порядковый номер поезда в словаре. |
| type | str | Тип состава. |
| original | str | Станция отправления поезда. |
| destination | str | Станция назначения поезда. |
| roadtime | int | Время в пути поезда в часах. |
| delta | relativedelta | Интервал времени между отправлением и следующим рейсом (от 2 до 8 часов). |
| position | datetime | Время отправления поезда. |
| destination\_time | datetime | Время прибытия поезда в пункт назначения. |
| train\_data | dict | Информация о количестве вагонов каждого типа для разных категорий поездов. |
| ticket\_data | list | Список билетов, которые будут проданы пассажирам. |
| sys\_prob | float | Вероятность, используемая при создании пассажира. |
| bank\_prob | float | Вероятность, связанная с использованием карт оплаты. |

|  |  |
| --- | --- |
| Метод | Описание |
| \_\_init\_\_ | Инициализация маршрута, загрузка данных поезда, определение времени отправления и прибытия, а также установка типа поезда. |
| swap | Меняет направление поезда, обновляет время отправления и прибытия, инициирует посадку пассажиров. |
| boarding | Выполняет распределение пассажиров по вагонам, создает объекты пассажиров и добавляет информацию о билетах в ticket\_data. |

### **Связь между классами**

* Класс Route создает объекты Vagon для размещения пассажиров в конкретных вагонах, а объекты Person используются для создания и хранения информации о каждом пассажире, покупающем билет.
* Эти вспомогательные классы позволяют Route выполнять свои задачи по распределению пассажиров и созданию билетов, структурируя данные и функции, связанные с вагонами и пассажирами.

**Рекомендации пользователя**

* Для запуска программы, убедитесь, что у вас установлен Python и необходимые библиотеки. Запустите код в среде разработки или командной строке и используйте интерфейс для настройки параметров и генерации данных.
* Убедитесь, что все модули программы находятся в одной директории для корректного выполнения.
* Запуск производится файлом main.py, который автоматически генерирует расписание в файл timetable.xlsx если его нет.
* Периодически проверяйте корректность данных перед генерацией билетов.
* При желании использовать свой файл расписания, убедитесь, что он называется traisbook.xlsx и его структура его заголовков совпадает с используемой: Номер поезда, категория, город отправления, город назначения, время следования (часы).
* Настройте веса для платежных систем и банков в соответствии с вашими требованиями, но убедитесь, что веса банков и платежных систем больше нуля.

# **Рекомендации программиста**

* Поддерживайте актуальность используемых библиотек и версии Python для сохранения актуальности и работоспособности кода на современных системах.
* Организуйте код в соответствии с best practices, следите за четким именованием переменных и функций.
* Периодически проводите тестирование на различных входных данных для обеспечения надежности и корректности программы.

# **Исходный код программы**

[**https://github.com/akryloff/spbu-algorithms-and-data-structures**](https://github.com/akryloff/spbu-algorithms-and-data-structures)

# **Контрольный пример**

Запустите код в среде разработки или командной строке. На экране появится интерфейс программы (Рис. 3)

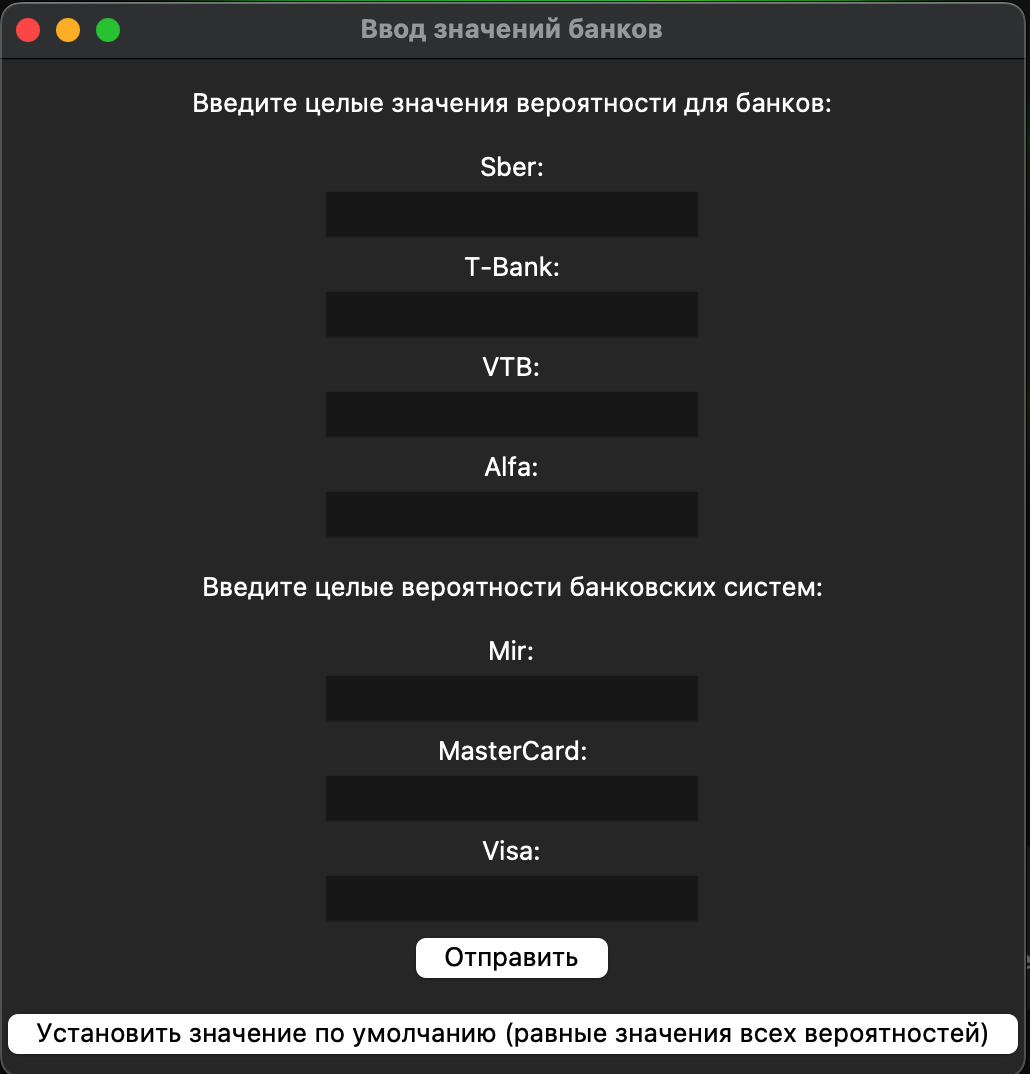


Рис 3. Интерфейс программы

Введите некоторые веса банков и банковских систем, или нажмите кнопку “Установить значение по умолчанию”, тогда программа установит значения 1 во все поля ввода и вероятность выбора систем и банков будет равна. Далее нажмите кнопку “Отправить” (В случае выбора значений по умолчанию программа перейдет к следующему шагу автоматически.

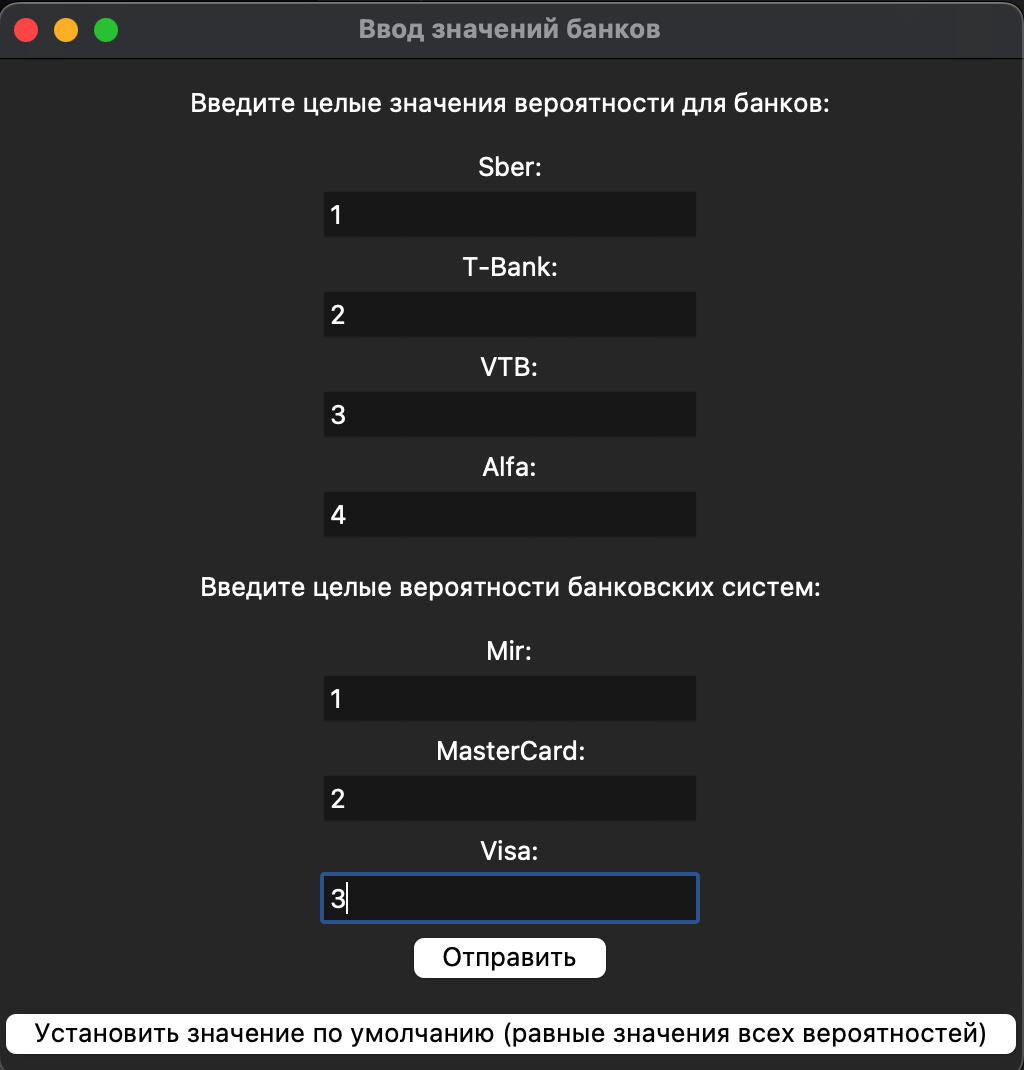


Рис 4. пример ввода весов банков

Далее программа выведет диалоговое окно, сообщающее что данные приняты. При нажатии кнопки начинается генерация датасета.

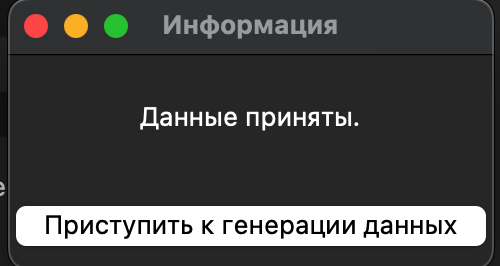


Рис 5. Пример диалогового окна

По окончании работы программа выведет сообщение о успешной генерации данных. Через несколько секунд окно закроется автоматически.

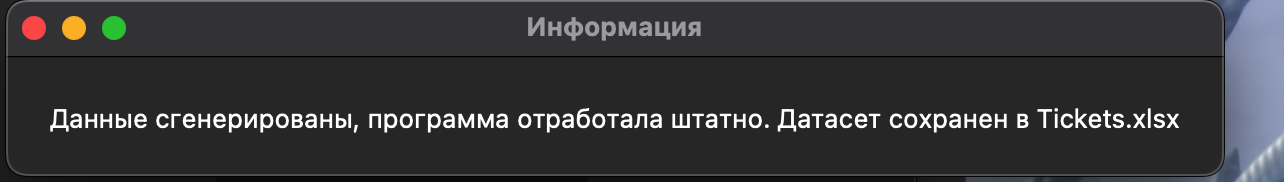


Рис 6. Пример сообщения

После этого датасет готов к дальнейшему использованию.

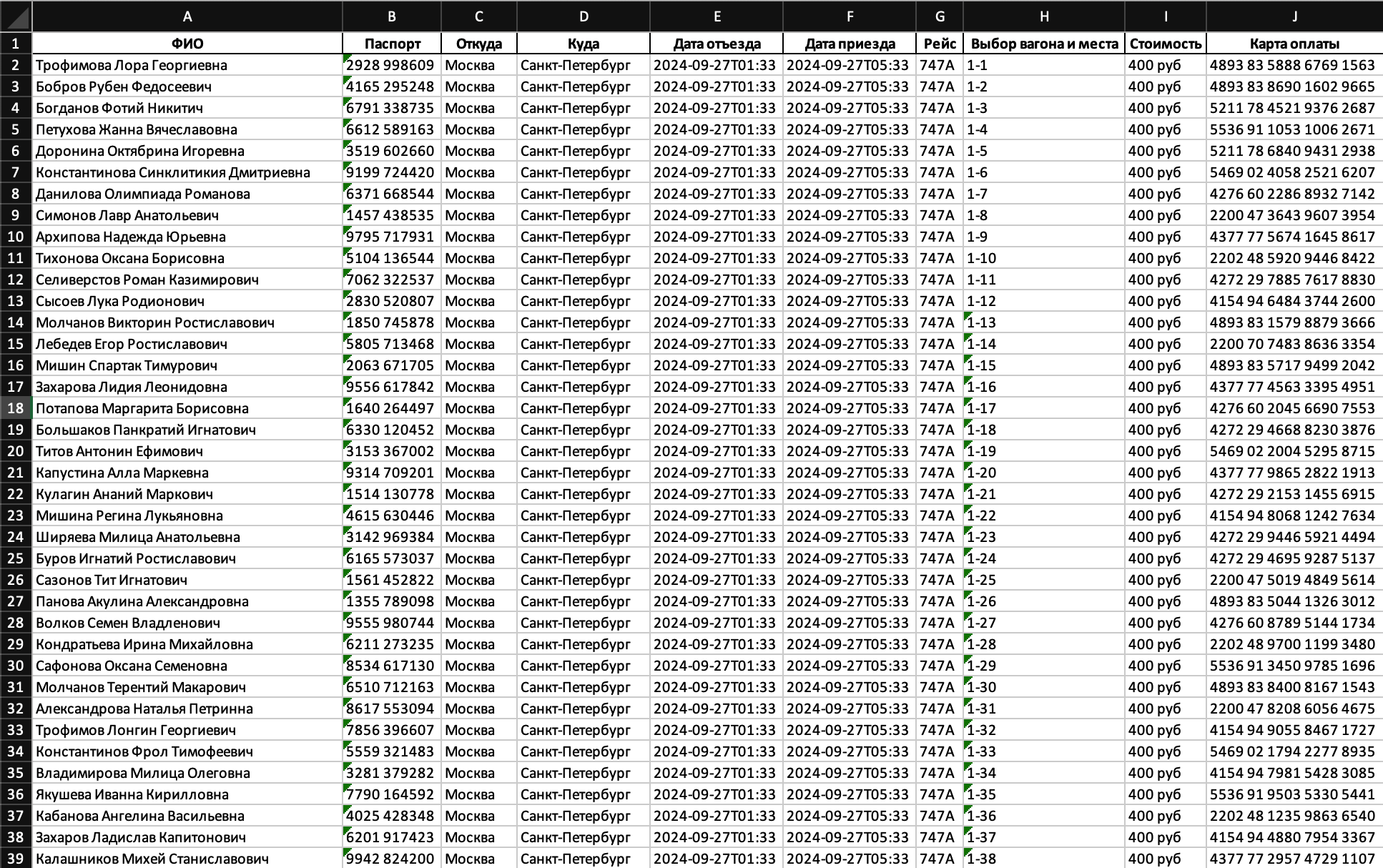


Рис 7. Пример сгенерированного датасета

**Вывод**

В рамках данной работы были исследованы принципы генерации синтетических данных, применительно к моделированию покупок железнодорожных билетов. Разработан алгоритм, который учитывает особенности пассажирских данных, маршрутов, типов поездов и вагонов, а также ценообразования билетов. Было реализовано программное обеспечение для автоматической генерации датасета, включающего такие данные, как личные данные пассажиров, информация о поездах, билетах и платежных системах. Программа позволяет настраивать параметры генерации банковских карт оплаты, обеспечивая соответствие заданным требованиям и реалистичность получаемого датасета.