Руководство разработчика

Итоговый проект по дисциплине

«Проектный семинар Python в науке о данных»

ОП «Информатика и вычислительная техника»

МИЭМ НИУ ВШЭ

БИВ223 Бригада №7

Ряднов Иван Максимович

(директор бригады, 89253237659, imryadnov\_1@edu.hse.ru)

Болезнов Степан Андреевич

Оглавление

[Описание приложения 2](#_Toc137505116)

[Системные требования 3](#_Toc137505117)

[Описание архитектуры приложения 4](#_Toc137505118)

[Описание структуры каталогов 5](#_Toc137505119)

[Названия и версии библиотек 6](#_Toc137505120)

[Методы и подходы 8](#_Toc137505121)

[Листинг main.py 9](#_Toc137505122)

[Листинг charts\_and\_reports.py 41](#_Toc137505123)

[Листинг utils.py 51](#_Toc137505124)

# Описание приложения

Наше приложение представляет собой интернет-магазин. Оно предназначено для управления информацией о товарах и заказах.

Функциональность приложения включает:

1. Отображение списка товаров с описанием, категорией и ценой.

2. Отображение списка заказов с датой заказа и суммой заказа

3. Отображение списка содержаний заказов с информацией о товарах и их количестве

4. Отображение объединенной таблицы.

5. Возможность добавления нового товара или заказа.

6. Возможность удаления записей из таблиц.

7. Возможность изменения данных о товаре или заказе.

# Системные требования

Приложение написано для 64-х разрядной операционной системы Windows.

Код не требует значительных ресурсов, поэтому минимальные системные требования будут зависеть только от требований самого интерпретатора Python:

1. 2 и более ядерный процессор
2. 2 и более ГБ оперативной памяти

Требуемое место на диске: 1 ГБ.

# Описание архитектуры приложения

Архитектура приложения организована следующим образом:

1. Пользовательский интерфейс (User Interface, GUI):

Графический интерфейс пользователя (GUI) реализуется с использованием библиотеки tkinter, которая предоставляет средства для создания окон, кнопок, полей ввода и других элементов интерфейса. GUI обеспечивает пользователю возможность взаимодействия с приложением, ввода данных, просмотра информации о продуктах и заказах.

1. Управление данными:

Хранилище данных использует форматы CSV или Excel файлов для хранения данных. Модуль pandas позволяет считывать и записывать данные из этих файлов в структуры данных, такие как DataFrame. Данные представлены в виде таблицы, где каждая строка представляет отдельную сущность (например, заказ) со своими свойствами (например, дата, сумма, количество товаров). Управление данными осуществляется с помощью модуля pandas, который предоставляет функции для выполнения различных операций с данными, таких как добавление новых записей, удаление записей, фильтрация, сортировка и агрегирование.

1. Логика приложения:

Контроллеры — это модули, которые обрабатывают действия пользователя и управляют логикой приложения. Например, при нажатии кнопки "Добавить товар", контроллер может создавать новый объект на основе введенных пользователем данных и сохранять его в справочник. Взаимодействие с данными осуществляется через методы модели данных и хранилища данных. Контроллеры могут использовать эти методы для чтения, записи данных, выполнения операций и взаимодействия с базой данных.

1. Взаимодействие между компонентами:

GUI и контроллеры: GUI передает пользовательские действия, такие как нажатия кнопок или ввод текста, контроллерам для обработки. Контроллеры в свою очередь могут обновлять интерфейс, отображая изменения данных или обрабатывая введенные пользователем команды. Контроллеры и модель данных: контроллеры могут вызывать методы модели данных для получения, обновления или удаления данных.

Модель данных возвращает результаты операций контроллерам.

Модель данных и хранилище данных: модель данных может использовать методы хранилища данных для чтения и записи данных в файлы формата CSV или Excel.

# Описание структуры каталогов

Структура приложения:

Программа размещается в стандартной структуре каталогов work:

* Data — содержит базу данных.
* Graphics — содержит копии графических отчетов.
* Library —содержит модуль utils.py для работы с .ini файлом и для экспорта данных в разные форматы. Он используется при работе приложения.
* Notes — содержит документацию, в нем размещается Руководства пользователя и разработчика.
* Scripts — содержит главный файл приложения main.py, модуль charts\_and\_reports.py для формирования графических и текстовых отчётов, файл config.ini с определением параметров настройки приложения

# Названия и версии библиотек

При написании программы был использован язык Python версии 3.9.12 (в составе дистрибутива Anaconda), а также следующие библиотеки:

1. os (операционная система):

* Описание: Библиотека os предоставляет функции для взаимодействия с операционной системой, такие как создание/удаление файлов и директорий, управление путями файловой системы, выполнение команд в командной строке и многое другое.
* Использование: os используется для выполнения операций, связанных с операционной системой, например, для работы с файлами, директориями, переменными окружения и выполнения команд на уровне операционной системы.

1. sys (система):

* Описание: Библиотека sys предоставляет функции и переменные для взаимодействия с интерпретатором Python и средой выполнения.
* Использование: sys используется для доступа к системным параметрам и функциональности, таким как аргументы командной строки, пути поиска модулей, завершение выполнения программы и другое.

1. pandas:

* Описание: Библиотека pandas предоставляет высокоуровневые структуры данных и инструменты анализа данных. Она представляет табличные данные и временные ряды, позволяя выполнять мощный и гибкий анализ данных.
* Использование: pandas широко используется для загрузки, обработки, анализа и манипулирования данными. Она предоставляет функциональность для сортировки, фильтрации, группировки, агрегации данных, а также для чтения и записи данных в различных форматах.
* Версия: 2.0.0

1. numpy:

* Описание: Библиотека numpy (Numerical Python) предоставляет поддержку для выполнения математических и научных операций на массивах данных. Она предлагает множество функций и методов для работы с многомерными массивами и векторами.
* Использование: numpy широко используется для работы с числовыми данными, выполнения математических операций, линейной алгебры, генерации случайных чисел и других вычислительных задач. Она является фундаментальной библиотекой для многих других пакетов анализа данных и научных вычислений в Python.
* Версия: 1.24.2

1. tkinter:

* Описание: Библиотека tkinter предоставляет инструменты для создания графического пользовательского интерфейса (GUI). Она предлагает набор виджетов и методов для создания окон, кнопок, полей ввода, меню и других элементов интерфейса.
* Использование: tkinter используется для разработки приложений с графическим интерфейсом, которые могут взаимодействовать с пользователями. Она предоставляет средства для создания интерактивных окон, обработки событий и управления элементами интерфейса.

1. matplotlib:

* Описание: Библиотека matplotlib является инструментом для визуализации данных. Она предоставляет функции для создания различных типов графиков, диаграмм и изображений.
* Использование: matplotlib широко используется для создания качественных графиков и визуализации данных. Она позволяет строить линейные графики, столбчатые диаграммы, круговые диаграммы, спектрограммы, трехмерные графики и многое другое. Это очень полезная библиотека для исследования данных и их представления в понятной форме.
* Версия: 3.6.0

# Методы и подходы

При разработке программы были использованы следующие методы и подходы:

1. Разделение на модули: Разделение функциональности программы на отдельные модули. Каждый модуль должен выполнять конкретную задачу или предоставлять определенную функциональность. Это позволяет улучшить читаемость, поддерживаемость и переиспользование кода.
2. Обработка ошибок: Реализация обработки исключений для обнаружения и управления ошибками в программе. Использование конструкции try-except для перехвата и обработки исключительных ситуаций.
3. Тестирование: Модульные тесты для проверки работоспособности отдельных компонентов программы.
4. Система контроля версий: Использование системы контроля версий Git для отслеживания изменений в коде, совместной работы и управления версиями программы.
5. Утилиты для разработки: Использование интегрированных сред разработки (IDE), такие как PyCharm, Visual Studio Code и Spyder для облегчения разработки, отладки и управления проектом.

# Листинг main.py

"""

Программа для отображения, изменения и описания данныхиз справочников с графическим интерфейсом

Бригада 7

Авторы: Ряднов И.М, Болезнов С.А.

"""

import os

import sys

import pandas as pd

import numpy as np

import tkinter as tk

from tkinter import ttk, Entry, colorchooser

from tkinter.ttk import Treeview, Spinbox

os.chdir("\\".join(os.getcwd().split("\\")[:-1]))

sys.path.append("\\".join(os.getcwd().split("\\")))

from library.utils import \*

from charts\_and\_reports import \*

def create\_pivot\_table():

    """

    Создание окна с выбором параметров для сводной таблицы

    Автор Ряднов И.М.

    :return: None

    """

    dialog = tk.Toplevel(root)

    dialog.title("Сводная таблица")

    agg\_functions = {

        "среднее значение": "mean",

        "сумма": "sum",

        "минимальное значение": "min",

        "максимальное значение": "max",

        "количество непустых значений": "count",

        "медиана": "median",

        "стандартное отклонение": "std",

        "дисперсия": "var",

        "первое значение": "first",

        "последнее значение": "last",

        "произведение": "prod"

    }

    tk.Label(dialog, text="Выберите значения (values):").grid(row=0, column=0, sticky="nesw")

    values\_entry = ttk.Combobox(dialog, values=['None'] + list(MERGED.columns), state='readonly')

    values\_entry.grid(row=1, column=0, sticky="nesw")

    tk.Label(dialog, text="Выберите индекс (index):").grid(row=0, column=1, sticky="nesw")

    index\_entry = ttk.Combobox(dialog, values=['None'] + list(MERGED.columns), state='readonly')

    index\_entry.grid(row=1, column=1, sticky="nesw")

    tk.Label(dialog, text="Выберите столбцы (columns):").grid(row=0, column=2, sticky="nesw")

    columns\_entry = ttk.Combobox(dialog, values=['None'] + list(MERGED.columns), state='readonly')

    columns\_entry.grid(row=1, column=2, sticky="nesw")

    tk.Label(dialog, text="Выберите функцию агрегации (aggfunc):").grid(row=0, column=3, sticky="nesw")

    aggfunc\_entry = ttk.Combobox(dialog, values=list(agg\_functions.keys()), state='readonly')

    aggfunc\_entry.grid(row=1, column=3, sticky="nesw")

    def create\_pivot\_table():

        """

        Создание сводной таблицы

        Автор Ряднов И.М.

        :return: None

        """

        values = values\_entry.get()

        index = index\_entry.get()

        columns = columns\_entry.get()

        aggfunc = aggfunc\_entry.get()

        try:

            pivot\_data = pd.pivot\_table(MERGED,

                                        values=values if values != 'None' else None,

                                        index=index if index != 'None' else None,

                                        columns=columns if columns != 'None' else None,

                                        aggfunc=agg\_functions.get(aggfunc))

            dialog.destroy()

            dialog2 = tk.Toplevel(root)

            dialog2.title("Сводная таблица")

            table = create\_table(dialog2, pivot\_data, True)

            table.pack(fill='both', expand=True)

            export\_data = pivot\_data.reset\_index()

            export\_button = ttk.Button(dialog2, text='Экспорт', command=lambda: save\_as(export\_data))

            export\_button.pack(fill='both', expand=True, )

            config\_widgets(dialog2, 2, 1)

        except Exception as ex:

            print(f"Данные не подходят для создания сводной таблицы\nДетали: {ex}")

    tk.Button(dialog, text="Создать таблицу", command=create\_pivot\_table).grid(row=2, column=0, columnspan=4,

                                                                               sticky="nesw")

    config\_widgets(dialog, 3, 4)

def create\_statistic\_report():

    """

    Создание окна с выбором атрибутов для статистического отчета

    Автор Ряднов И.М.

    :return: None

    """

    dialog = tk.Toplevel(root)

    dialog.title("Статистический отчет")

    tk.Label(dialog, text="Выберите первый атрибут:").grid(row=0, column=0, sticky="nesw")

    attribute\_1\_entry = ttk.Combobox(dialog, values=list(MERGED.columns), state='readonly')

    attribute\_1\_entry.grid(row=1, column=0, sticky="nesw")

    tk.Label(dialog, text="Выберите второй атрибут:").grid(row=0, column=1, sticky="nesw")

    attribute\_2\_entry = ttk.Combobox(dialog, values=list(MERGED.columns), state='readonly')

    attribute\_2\_entry.grid(row=1, column=1, sticky="nesw")

    def create\_stat\_report():

        """

        Создание статистического отчета

        Автор Ряднов И.М., Болезнов С.А.

        :return: None

        """

        attribute\_1 = attribute\_1\_entry.get()

        attribute\_2 = attribute\_2\_entry.get()

        attribute\_rep = generate\_attribute\_report(MERGED, attribute\_1, attribute\_2)

        dialog.destroy()

        dialog2 = tk.Toplevel(root)

        dialog2.title("Статистический отчёт")

        tab\_controler = ttk.Notebook(dialog2)

        tab\_1 = ttk.Frame(tab\_controler)

        tab\_2 = ttk.Frame(tab\_controler)

        tab\_3 = ttk.Frame(tab\_controler)

        tab\_4 = ttk.Frame(tab\_controler)

        tab\_controler.add(tab\_1, text='Таблица 1')

        tab\_controler.add(tab\_2, text='Таблица 2')

        tab\_controler.add(tab\_3, text='Таблица 3')

        tab\_controler.add(tab\_4, text='Таблица 4')

        create\_table(tab\_1, attribute\_rep[0])

        create\_table(tab\_2, attribute\_rep[1])

        create\_table(tab\_3, attribute\_rep[2])

        create\_table(tab\_4, attribute\_rep[3])

        tab\_controler.grid(row=0, column=0, columnspan=2, sticky='nsew')

        export\_button\_1 = ttk.Button(dialog2, text='Экспорт таблицы 1', command=lambda: save\_as(attribute\_rep[0]))

        export\_button\_1.grid(row=1, column=0, sticky='nsew')

        export\_button\_1 = ttk.Button(dialog2, text='Экспорт таблицы 2', command=lambda: save\_as(attribute\_rep[1]))

        export\_button\_1.grid(row=1, column=1, sticky='nsew')

        export\_button\_1 = ttk.Button(dialog2, text='Экспорт таблицы 3', command=lambda: save\_as(attribute\_rep[2]))

        export\_button\_1.grid(row=2, column=0, sticky='nsew')

        export\_button\_1 = ttk.Button(dialog2, text='Экспорт таблицы 4', command=lambda: save\_as(attribute\_rep[3]))

        export\_button\_1.grid(row=2, column=1, sticky='nsew')

        config\_widgets(dialog2, 3, 2)

    tk.Button(dialog, text="Создать таблицу", command=create\_stat\_report).grid(row=2, column=0, columnspan=2,

                                                                               sticky="nesw")

    config\_widgets(dialog, 3, 2)

def create\_table(tab, data: pd.DataFrame, pivot=False) -> Treeview:

    """

    Функция для добавления таблицы в окно

    :param pivot: Является ли таблица сводной

    :param tab(tk.ttk.Frame): Название окна

    :param data(pd.DataFrame): Данные таблицы

    :return table(ttk.Treeview): Полученный виджет таблицы

    Автор: Болезнов С.А., Ряднов И.М.

    """

    translater = {

        'Order ID': 'Номер заказа',

        'Product ID': 'Номер товара',

        'Quantity': 'Количество',

        'Product': 'Товар',

        'Description': 'Описание',

        'Price': 'Цена',

        'Category': 'Категория',

        'Date': 'Дата',

        'Sum': 'Сумма'

    }

    table\_frame = ttk.Frame(tab)

    table\_frame.pack(fill='both', expand=True)

    table = ttk.Treeview(table\_frame)

    table.pack(side='left', fill='both', expand=True)

    table.configure(style='Treeview')

    scrollbar = ttk.Scrollbar(table\_frame, orient='vertical', command=table.yview)

    scrollbar.pack(side='right', fill='y')

    table.configure(yscroll=scrollbar.set)

    if pivot:

        data = data.reset\_index()

    heads = [translater.get(head, str(head)) for head in data.columns]

    table['columns'] = heads

    table['show'] = 'headings'

    for header in heads:

        table.heading(header, text=header)

        if len(heads) == 9:

            table.column(header, width=0, anchor='center')

        else:

            table.column(header, anchor='center')

    for i, row in data.iterrows():

        values = list(row)

        table.insert("", "end", text=i, values=values)

    return table

def new\_save():

    """

    Функция для сохранения одной таблицы в отдельный файл

    Автор: Болезнов С.А.

    :return: None

    """

    index = tab\_control.index(tab\_control.select())

    tabs = [GOODS, ORDERS, ORDERS\_STRUCTURE, MERGED]

    table = tabs[index]

    save\_as(table)

def add\_datas(parent) -> tuple[Spinbox, Spinbox, Spinbox, Spinbox, Spinbox, Spinbox]:

    """

    Функция для добавления в окно полей ввода даты

    Автор: Болезнов С.А.

    :param parent: Название окна

    :return start\_date\_day(tk.Spinbox): Добавленный виджет

    :return start\_date\_month(tk.Spinbox): Добавленный виджет

    :return start\_date\_year(tk.Spinbox): Добавленный виджет

    :return end\_date\_day(tk.Spinbox): Добавленный виджет

    :return end\_date\_month(tk.Spinbox): Добавленный виджет

    :return end\_date\_year(tk.Spinbox): Добавленный виджет

    """

    start\_date\_label = tk.Label(parent, text='Выберите начальную дату:')

    start\_date\_label.grid(column=0, row=0, columnspan=3, sticky="nesw")

    start\_date\_day = tk.Spinbox(parent, from\_=1, to=31, width=2)

    start\_date\_day.grid(column=0, row=1, sticky="nesw")

    start\_date\_month = tk.Spinbox(parent, from\_=1, to=12, width=2)

    start\_date\_month.grid(column=1, row=1, sticky="nesw")

    start\_date\_year = tk.Spinbox(parent, from\_=2022, to=2023, width=4)

    start\_date\_year.grid(column=2, row=1, sticky="nesw")

    end\_date\_label = tk.Label(parent, text='Выберите конечную дату:')

    end\_date\_label.grid(column=0, row=2, columnspan=3, sticky="nesw")

    end\_date\_day = tk.Spinbox(parent, from\_=1, to=31, width=2)

    end\_date\_day.grid(column=0, row=3, sticky="nesw")

    end\_date\_month = tk.Spinbox(parent, from\_=1, to=12, width=2)

    end\_date\_month.grid(column=1, row=3, sticky="nesw")

    end\_date\_year = tk.Spinbox(parent, from\_=2022, to=2023, width=4)

    end\_date\_year.grid(column=2, row=3, sticky="nesw")

    return start\_date\_day, start\_date\_month, start\_date\_year, end\_date\_day, end\_date\_month, end\_date\_year

def config\_widgets(parent, rows: int, cols: int):

    """

    Функция для задания веса каждому элемента окна.

    Необходимо для коректного отображения окна при растяжении.

    Автор: Болезнов С.А.

    :param parent: Название окна

    :param rows: Количество рядов в сетке окна

    :param cols: Количество столбцов в сетке окна

    :return: None

    """

    for col in range(cols):

        parent.columnconfigure(index=col, weight=1)

    for row in range(rows):

        parent.rowconfigure(index=row, weight=1)

def report\_1():

    """

    Создание нового окна для ввода необходимых параметров для текстового отчета

    Автор: Болезнов С.А.

    :return: None

    """

    dialog = tk.Toplevel(root)

    dialog.title("Текстовый отчёт")

    start\_date\_day, start\_date\_month, start\_date\_year, end\_date\_day, \

        end\_date\_month, end\_date\_year = add\_datas(dialog)

    tk.Label(dialog, text="Выберете категорию:").grid(row=4, column=0, columnspan=3)

    categories = list(GOODS.Category.unique())

    firm\_entry = ttk.Combobox(dialog, values=categories, textvariable=categories[0])

    firm\_entry.grid(row=5, column=0, columnspan=3, sticky="nesw")

    config\_widgets(dialog, 7, 3)

    def ok\_button():

        """

        Вывод полученного отчета на экран

        Автор: Болезнов С.А.

        :return: None

        """

        first\_date = f'{start\_date\_year.get()}-{start\_date\_month.get()}-{start\_date\_day.get()}'

        second\_date = f'{end\_date\_year.get()}-{end\_date\_month.get()}-{end\_date\_day.get()}'

        category = firm\_entry.get()

        report = report\_about\_firm(MERGED, first\_date, second\_date, category)

        dialog.destroy()

        dialog2 = tk.Toplevel(root)

        dialog2.title(f"Текстовый отчёт о продажах {category}")

        table = create\_table(dialog2, report)

        table.pack(fill='both', expand=True)

        export\_button = ttk.Button(dialog2, text='Экспорт', command=lambda: save\_as(report))

        export\_button.pack(fill='both', expand=True, )

        config\_widgets(dialog2, 2, 1)

    tk.Button(dialog, text="Создать", command=ok\_button).grid(row=6, column=0, columnspan=2, sticky="nesw")

    tk.Button(dialog, text='Отменить', command=dialog.destroy).grid(row=6, column=2, sticky='nsew')

def create\_hist():

    """

    Создание нового окна с выбором гистограмм

    Автор: Болезнов С.А.

    :return: None

    """

    dialog = tk.Toplevel(root)

    dialog.title("Создание гистограммы")

    tk.Button(dialog, text="Гистограмма распределения\nколичества товаров\nв заказе",

              command=lambda: histogram(MERGED, 'Quantity')).grid(row=0, column=0, sticky="nesw")

    tk.Button(dialog, text='Гистограмма распределения\nцены на товары',

              command=lambda: histogram(MERGED, 'Price')).grid(row=0, column=1, sticky='nsew')

    tk.Button(dialog, text="Гистограмма распределения\nсуммы заказов",

              command=lambda: histogram(MERGED, 'Sum')).grid(row=1, column=0, sticky="nesw")

    tk.Button(dialog, text='Отменить', command=dialog.destroy).grid(row=1, column=1, sticky='nsew')

    config\_widgets(dialog, 2, 2)

def create\_bar():

    """

    Создание нового окна для выбора столбчатых диаграмм

    Автор: Болезнов С.А.

    :return: None

    """

    dialog = tk.Toplevel(root)

    dialog.title("Создание столбчатой диаграммы")

    tk.Button(dialog, text="Продажи по дням",

              command=lambda: report\_day\_sales(ORDERS)).grid(row=0, column=0, sticky="nesw")

    tk.Button(dialog, text='Продажи по неделям',

              command=lambda: report\_week\_sales(ORDERS)).grid(row=0, column=1, sticky='nsew')

    tk.Button(dialog, text="Продажи по месяцам",

              command=lambda: report\_month\_sales(ORDERS)).grid(row=1, column=0, sticky="nesw")

    tk.Button(dialog, text="Продажи по годам",

              command=lambda: report\_year\_sales(ORDERS)).grid(row=1, column=1, sticky="nesw")

    tk.Button(dialog, text='Отменить', command=dialog.destroy).grid(row=2, column=0, columnspan=2, sticky='nsew')

    config\_widgets(dialog, 3, 2)

def create\_scatter():

    """

    Функция для подготовки данных и отображения таблицы рассеивания

    Автор: Ряднов И.М.

    :return: None

    """

    data = pd.merge(ORDERS\_STRUCTURE, GOODS, on="Product ID")

    data['Price'] = data['Price'].astype(float)

    report\_price\_by\_quantity(data)

def del\_line():

    """

    Функция для удаления строк таблицы.

    Автор: Болезнов С.А.

    :return: None

    """

    index = tab\_control.index(tab\_control.select())

    if index == 0:

        selected\_item = goods\_table.selection()[0]

        selected\_line = goods\_table.item(selected\_item)['values']

        global GOODS

        GOODS = GOODS.drop(GOODS.loc[GOODS['Product ID'] == selected\_line[0]].index)

        goods\_table.delete(selected\_item)

    if index == 1:

        selected\_item = orders\_table.selection()[0]

        selected\_line = orders\_table.item(selected\_item)['values']

        global ORDERS

        ORDERS = ORDERS.drop(ORDERS.loc[ORDERS['Order ID'] == selected\_line[0]].index)

        orders\_table.delete(selected\_item)

    if index == 2:

        selected\_item = orders\_structure\_table.selection()[0]

        selected\_line = orders\_structure\_table.item(selected\_item)['values']

        global ORDERS\_STRUCTURE

        ORDERS\_STRUCTURE = ORDERS\_STRUCTURE.drop(

            ORDERS\_STRUCTURE.loc[ORDERS\_STRUCTURE['Order ID'] == selected\_line[0]].index)

        orders\_structure\_table.delete(selected\_item)

    if index == 3:

        selected\_item = merged\_table.selection()[0]

        selected\_line = merged\_table.item(selected\_item)['values']

        global MERGED

        MERGED = MERGED.drop(MERGED.loc[MERGED['Order ID'] == selected\_line[0]].index)

        merged\_table.delete(selected\_item)

def edit\_line():

    """

    Внесение изменений в строку таблицы.

    Автор: Болезнов С.А.

    :return: None

    """

    index = tab\_control.index(tab\_control.select())

    if index == 0:

        selected\_item = goods\_table.selection()[0]

        selected\_line = goods\_table.item(selected\_item)['values']

        name\_entry, description\_entry, price\_entry, category\_entry = goods\_dialog(selected\_item,

                                                                                  selected\_line)

        name\_entry.insert(0, selected\_line[1])

        description\_entry.insert(0, selected\_line[2])

        price\_entry.set(selected\_line[3])

        category\_entry.insert(0, selected\_line[4])

    if index == 1:

        selected\_item = orders\_table.selection()[0]

        selected\_line = orders\_table.item(selected\_item)['values']

        date\_day, date\_month, date\_year, sum\_entry = orders\_dialog(selected\_item, selected\_line)

        month, day, year = selected\_line[1].split('/')

        date\_day.set(day)

        date\_month.set(month)

        date\_year.set(year)

        sum\_entry.set(selected\_line[2])

    if index == 2:

        selected\_item = orders\_structure\_table.selection()[0]

        selected\_line = orders\_structure\_table.item(selected\_item)['values']

        good\_entry, quantity\_entry = orders\_structure\_dialog(selected\_item, selected\_line)

        good\_entry.set(selected\_line[1])

        quantity\_entry.set(selected\_line[2])

    if index == 3:

        print('Данная операция невозможна. Данные можно изменять только в отдельных справочниках')

def goods\_dialog(selected\_item, selected\_line) -> tuple[Entry, Entry, Spinbox, Entry]:

    """

    Создание нового окна с вводом новых/измененных значений для таблицы товары

    Автор: Болезнов С.А.

    :param selected\_item: Выбранный объект в таблице

    :param selected\_line: Список выбранных значений

    :return name\_entry (tk.Entry): Виджет для ввода названия

    :return description\_entry (tk.Entry): Виджет для ввода описания

    :return price\_entry (tk.Entry): Виджет для ввода цены

    :return category\_entry(tk.Entry): Виджет для ввода категории

    """

    dialog = tk.Toplevel(root)

    dialog.title('Изменение товара')

    tk.Label(dialog, text="Название").grid(row=0, column=0, sticky="nesw", columnspan=2)

    name\_entry = tk.Entry(dialog)

    name\_entry.grid(row=1, column=0, sticky="nesw", columnspan=2)

    tk.Label(dialog, text="Описание").grid(row=2, column=0, sticky="nesw", columnspan=2)

    description\_entry = tk.Entry(dialog)

    description\_entry.grid(row=3, column=0, sticky="nesw", columnspan=2)

    tk.Label(dialog, text="Ценв").grid(row=4, column=0, sticky="nesw", columnspan=2)

    price\_entry = ttk.Spinbox(dialog, increment=0.01, from\_=0, to=1000)

    price\_entry.grid(row=5, column=0, sticky="nesw", columnspan=2)

    tk.Label(dialog, text="Категория").grid(row=6, column=0, sticky="nesw", columnspan=2)

    category\_entry = tk.Entry(dialog)

    category\_entry.grid(row=7, column=0, sticky="nesw", columnspan=2)

    def save():

        """

        Функция для сохранения полученных значений

        Автор: Болезнов С.А.

        :return: None

        """

        name = name\_entry.get()

        description = description\_entry.get()

        price = price\_entry.get()

        category = category\_entry.get()

        values = [selected\_line[0], name, description, price, category]

        goods\_table.item(selected\_item, values=values)

        dialog.destroy()

        global GOODS

        GOODS = replace\_row\_values(GOODS, selected\_line, values)

        create\_new\_merge()

    save\_button = ttk.Button(dialog, text='Сохранить', command=save)

    cancel\_button = ttk.Button(dialog, text='Отмена', command=dialog.destroy)

    save\_button.grid(row=8, column=0, sticky="nesw")

    cancel\_button.grid(row=8, column=1, sticky="nesw")

    config\_widgets(dialog, 9, 2)

    return name\_entry, description\_entry, price\_entry, category\_entry

def replace\_row\_values(df: pd.DataFrame, old\_values: list, new\_values: list) -> pd.DataFrame:

    """

    Функция для замены одной строки датафрейма на новую

    Автор: Болезнов С.А.

    :param df(pd.DataFrame): Исходный датафрейм

    :param old\_values: Список значений, которые будут изсменены

    :param new\_values: Список новых значений

    :return df(pd.DataFrame): Изменененный датафрейм

    """

    if 'Quantity' in df.columns:

        row\_idx = (df['Order ID'] == old\_values[0])

    elif 'Product ID' in df.columns:

        row\_idx = (df['Product ID'] == old\_values[0])

    else:

        row\_idx = (df['Order ID'] == old\_values[0])

    df.loc[row\_idx] = new\_values

    return df

def orders\_dialog(selected\_item, selected\_line):

    """

    Функция для создания нового окна с вводом новых/измененных значений для таблицы заказы

    Автор: Болезнов С.А.

    :param selected\_item: Выбранный объект в таблице

    :param selected\_line: Список выбранных значений

    :return date\_day(ttk.Spinbox): Виджет для ввода дня

    :return date\_month(ttk.Spinbox): Виджет для ввода месяца

    :return date\_year(ttk.Spinbox): Виджет для ввода года

    :return sum\_entry(ttk.Spinbox): Виджет для ввода суммы заказа

    """

    dialog = tk.Toplevel(root)

    dialog.title('Изменение заказ')

    tk.Label(dialog, text='Дата').grid(column=0, row=0, columnspan=3, sticky="nesw")

    date\_day = ttk.Spinbox(dialog, from\_=1, to=31, width=2)

    date\_day.grid(column=0, row=1, sticky="nesw")

    date\_month = ttk.Spinbox(dialog, from\_=1, to=12, width=2)

    date\_month.grid(column=1, row=1, sticky="nesw")

    date\_year = ttk.Spinbox(dialog, from\_=2022, to=2023, width=4)

    date\_year.grid(column=2, row=1, sticky="nesw")

    tk.Label(dialog, text="Сумма").grid(row=2, column=0, sticky="nesw", columnspan=3)

    sum\_entry = ttk.Spinbox(dialog, increment=1, from\_=0, to=100000)

    sum\_entry.grid(row=3, column=0, sticky="nesw", columnspan=3)

    def save():

        """

        Функция для сохранения полученных значений

        Автор: Болезнов С.А.

        :return: None

        """

        date = f'{date\_month.get()}/{date\_day.get()}/{date\_year.get()}'

        sum = sum\_entry.get()

        values = [selected\_line[0], date, sum]

        orders\_table.item(selected\_item, values=values)

        dialog.destroy()

        global ORDERS

        ORDERS = replace\_row\_values(ORDERS, selected\_line, values)

        create\_new\_merge()

    save\_button = ttk.Button(dialog, text='Сохранить', command=save)

    cancel\_button = ttk.Button(dialog, text='Отмена', command=dialog.destroy)

    save\_button.grid(row=8, column=0, sticky="nesw", columnspan=2)

    cancel\_button.grid(row=8, column=2, sticky="nesw")

    config\_widgets(dialog, 4, 3)

    return date\_day, date\_month, date\_year, sum\_entry

def create\_new\_merge():

    """

    Функция для обновления общей таблицы после внесения изменений в справочники

    Автор: Болезнов С.А.

    :return: None

    """

    global MERGED

    widgets\_list = tab4.pack\_slaves()

    for element in widgets\_list:

        element.destroy()

    MERGED = merge\_files(GOODS, ORDERS, ORDERS\_STRUCTURE)

    create\_table(tab4, MERGED)

def orders\_structure\_dialog(selected\_item, selected\_line):

    """

    Функция для создания нового окна с вводом новых/измененных значений для таблицы состав заказов

    Автор: Болезнов С.А.

    :param selected\_item: Выбранный объект в таблице

    :param selected\_line: Список выбранных значений

    :return good\_entry(tk.Entry): Виджет для ввода товара

    :return quantity\_entry(tk.Entry): Виджеты ввода количества

    """

    dialog = tk.Toplevel(root)

    dialog.title('Изменение состава заказа')

    tk.Label(dialog, text='Товар').grid(column=0, row=0, sticky="nesw", columnspan=2)

    ids = list(GOODS['Product ID'])

    good\_entry = ttk.Combobox(dialog, values=ids)

    good\_entry.grid(column=0, row=1, sticky="nesw", columnspan=2)

    tk.Label(dialog, text="Количество").grid(row=2, column=0, sticky="nesw", columnspan=2)

    quantity\_entry = ttk.Spinbox(dialog, increment=1, from\_=0, to=100)

    quantity\_entry.grid(row=3, column=0, sticky="nesw", columnspan=2)

    def save():

        """

        Функция для сохранения полученных значений

        Автор: Болезнов С.А.

        :return: None

        """

        good\_id = int(good\_entry.get())

        quantity = quantity\_entry.get()

        values = [selected\_line[0], good\_id, quantity]

        orders\_structure\_table.item(selected\_item, values=values)

        dialog.destroy()

        global ORDERS\_STRUCTURE

        ORDERS\_STRUCTURE = replace\_row\_values(ORDERS\_STRUCTURE, selected\_line, values)

        create\_new\_merge()

    save\_button = ttk.Button(dialog, text='Сохранить', command=save)

    cancel\_button = ttk.Button(dialog, text='Отмена', command=dialog.destroy)

    save\_button.grid(row=4, column=0, sticky="nesw")

    cancel\_button.grid(row=4, column=1, sticky="nesw")

    config\_widgets(dialog, 5, 2)

    return good\_entry, quantity\_entry

def generate\_id(index) -> int:

    """

    Функция для создания нового индекса для справочников

    Автор: Болезнов С.А.

    :return new\_id(int): Новый индекс

    """

    if index == 0:

        ids = [int(x) for x in GOODS['Product ID']]

        for i in range(1, max(ids) + 1):

            if i not in ids:

                return i

        return max(ids) + 1

    if index == 1:

        ids = [x for x in ORDERS['Order ID']]

        new\_id = ids[0]

        while new\_id in ids:

            first\_part = np.random.randint(1, 100)

            second\_part = np.random.randint(1, 1000)

            third\_part = np.random.randint(1, 10000)

            new\_id = f'{first\_part}-{second\_part}-{third\_part}'

        return new\_id

def add\_order():

    """

    Функция для добавления заказа

    Автор: Болезнов С.А.

    :return: None

    """

    dialog = tk.Toplevel(root)

    dialog.title('Создание нового заказа')

    tk.Label(dialog, text='Дата').grid(column=0, row=0, columnspan=3, sticky="nesw")

    date\_day = ttk.Spinbox(dialog, from\_=1, to=31, width=2)

    date\_day.grid(column=0, row=1, sticky="nesw")

    date\_day.set(1)

    date\_month = ttk.Spinbox(dialog, from\_=1, to=12, width=2)

    date\_month.grid(column=1, row=1, sticky="nesw")

    date\_month.set(1)

    date\_year = ttk.Spinbox(dialog, from\_=2022, to=2023, width=4)

    date\_year.grid(column=2, row=1, sticky="nesw")

    date\_year.set(2022)

    tk.Label(dialog, text="Сумма").grid(row=2, column=0, sticky="nesw", columnspan=3)

    sum\_entry = ttk.Spinbox(dialog, increment=1, from\_=0, to=100000)

    sum\_entry.set(1000)

    sum\_entry.grid(row=3, column=0, sticky="nesw", columnspan=3)

    tk.Label(dialog, text='Товар').grid(column=0, row=4, sticky="nesw", columnspan=3)

    ids = list(GOODS['Product ID'])

    products = list(GOODS['Product'])

    good\_entry = ttk.Combobox(dialog, values=products)

    good\_entry.set(products[0])

    good\_entry.grid(column=0, row=5, sticky="nesw", columnspan=3)

    tk.Label(dialog, text="Количество").grid(row=6, column=0, sticky="nesw", columnspan=3)

    quantity\_entry = ttk.Spinbox(dialog, increment=1, from\_=0, to=100)

    quantity\_entry.set(5)

    quantity\_entry.grid(row=7, column=0, sticky="nesw", columnspan=3)

    def save():

        """

        Функция для сохранения полученных данных

        Автор: Болезнов С.А.

        :return: None

        """

        date = f'{date\_month.get()}/{date\_day.get()}/{date\_year.get()}'

        sum = sum\_entry.get()

        quantity = quantity\_entry.get()

        product = ids[products.index(good\_entry.get())]

        order\_id = generate\_id(1)

        order\_values = [order\_id, date, sum]

        struct\_values = [order\_id, product, quantity]

        global ORDERS, ORDERS\_STRUCTURE

        ORDERS = pd.concat([pd.DataFrame([order\_values], columns=ORDERS.columns), ORDERS], ignore\_index=True)

        ORDERS\_STRUCTURE = pd.concat(

            [pd.DataFrame([struct\_values], columns=ORDERS\_STRUCTURE.columns), ORDERS\_STRUCTURE],

            ignore\_index=True)

        dialog.destroy()

        widgets\_list = tab2.pack\_slaves()

        for element in widgets\_list:

            element.destroy()

        global orders\_table, orders\_structure\_table

        orders\_table = create\_table(tab2, ORDERS)

        widgets\_list = tab3.pack\_slaves()

        for element in widgets\_list:

            element.destroy()

        orders\_structure\_table = create\_table(tab3, ORDERS\_STRUCTURE)

    save\_button = ttk.Button(dialog, text='Создать', command=save)

    cancel\_button = ttk.Button(dialog, text='Отмена', command=dialog.destroy)

    save\_button.grid(row=8, column=0, sticky="nesw", columnspan=2)

    cancel\_button.grid(row=8, column=2, sticky="nesw")

    config\_widgets(dialog, 9, 3)

def add\_product():

    """

    Функция для добавления товара

    Автор: Болезнов С.А.

    :return: None

    """

    dialog = tk.Toplevel(root)

    dialog.title('Создание нового товара')

    tk.Label(dialog, text="Название").grid(row=0, column=0, sticky="nesw", columnspan=2)

    product\_entry = tk.Entry(dialog)

    product\_entry.grid(row=1, column=0, sticky="nesw", columnspan=2)

    tk.Label(dialog, text="Описание").grid(row=2, column=0, sticky="nesw", columnspan=2)

    description\_entry = tk.Entry(dialog)

    description\_entry.grid(row=3, column=0, sticky="nesw", columnspan=2)

    tk.Label(dialog, text="Цена").grid(row=4, column=0, sticky="nesw")

    sum\_entry = ttk.Spinbox(dialog, increment=1, from\_=0, to=100000)

    sum\_entry.set(1000)

    sum\_entry.grid(row=5, column=0, sticky="nesw", columnspan=2)

    tk.Label(dialog, text="Категория").grid(row=6, column=0, sticky="nesw")

    category\_entry = tk.Entry(dialog)

    category\_entry.grid(row=7, column=0, sticky="nesw", columnspan=2)

    def save():

        """

        Функция для сохранения полученных данных

        Автор: Болезнов С.А.

        :return: None

        """

        product = product\_entry.get()

        description = description\_entry.get()

        sum = sum\_entry.get()

        category = category\_entry.get()

        product\_id = generate\_id(0)

        product\_values = [product\_id, product, description, sum, category]

        global GOODS

        GOODS = pd.concat([pd.DataFrame([product\_values], columns=GOODS.columns), GOODS], ignore\_index=True)

        dialog.destroy()

        widgets\_list = tab1.pack\_slaves()

        for element in widgets\_list:

            element.destroy()

        global goods\_table

        goods\_table = create\_table(tab1, GOODS)

    save\_button = ttk.Button(dialog, text='Создать', command=save)

    cancel\_button = ttk.Button(dialog, text='Отмена', command=dialog.destroy)

    save\_button.grid(row=8, column=0, sticky="nesw")

    cancel\_button.grid(row=8, column=1, sticky="nesw")

    config\_widgets(dialog, 9, 2)

def config\_color():

    """

    Открытие палитры и изменение цвета

    Автор Ряднов И.М.

    :return: None

    """

    style = ttk.Style()

    color = colorchooser.askcolor(title="Выберите цвет фона")

    if color[1] is not None:

        style.configure("Treeview", background=color[1])

    update\_ini\_value('BackgroundColor', color[1])

def get\_settings():

    """

    Чтение ini файла и применение полученных значений к приложению

    Автор Ряднов И.М.

    :return cfg(list): Список конфигурация

    """

    config = read\_ini\_file().get('Settings')

    height = config.get('Height')

    width = config.get('Width')

    resize\_h = config.get('ResizableHeight')

    resize\_w = config.get('ResizableWidth')

    background\_color = config.get('BackgroundColor')

    font\_size = config.get('FontSize')

    font\_family = config.get('FontFamily')

    font\_style = config.get('FontStyle')

    cfg = [height,

           width,

           bool(resize\_h),

           bool(resize\_w),

           background\_color,

           int(font\_size),

           font\_family,

           font\_style]

    return cfg

def open\_font\_settings():

    """

    Окно с изменением шрифта

    Автор Ряднов И.М.

    :return: None

    """

    font\_settings\_window = tk.Toplevel(root)

    font\_settings\_window.title("Настройки шрифта")

    def apply\_font\_settings():

        """

        Подтверждение изменения настроек шрифта

        Автор Ряднов И.М.

        :return: None

        """

        font = font\_var.get()

        size = size\_var.get()

        weight = weight\_var.get()

        change\_font(font, size, weight)

        font\_settings\_window.destroy()

    font\_var = tk.StringVar()

    font\_var.set("Arial")

    font\_label = ttk.Label(font\_settings\_window, text="Название шрифта:")

    font\_label.pack()

    font\_optionmenu = ttk.OptionMenu(font\_settings\_window, font\_var, "Arial", "Arial", "Times New Roman", "Courier")

    font\_optionmenu.pack()

    size\_var = tk.StringVar()

    size\_var.set("12")

    size\_label = ttk.Label(font\_settings\_window, text="Размер шрифта:")

    size\_label.pack()

    size\_optionmenu = ttk.OptionMenu(font\_settings\_window, size\_var, "12", "12", "14", "16")

    size\_optionmenu.pack()

    weight\_var = tk.StringVar()

    weight\_var.set("normal")

    weight\_label = ttk.Label(font\_settings\_window, text="Толщина шрифта:")

    weight\_label.pack()

    weight\_optionmenu = ttk.OptionMenu(font\_settings\_window, weight\_var, "normal", "normal", "bold")

    weight\_optionmenu.pack()

    apply\_button = ttk.Button(font\_settings\_window, text="Применить", command=apply\_font\_settings)

    apply\_button.pack()

def change\_resolution(width, height):

    """

    Изменение разрешения

    Автор Ряднов И.М.

    :return: None

    """

    root.geometry(f"{width}x{height}".format(root.winfo\_screenwidth() // 2 - 400, root.winfo\_screenheight() // 2 - 300))

    update\_ini\_value('Height', height)

    update\_ini\_value('Width', width)

def change\_font(font, size, weight):

    """

    Изменение параметров шрифта

    Автор Ряднов И.М.

    :return: None

    """

    style = ttk.Style()

    style.configure("Treeview", font=(font, size, weight))

    update\_ini\_value('FontFamily', font)

    update\_ini\_value('FontSize', size)

    update\_ini\_value('FontStyle', weight)

def toggle\_window\_resize():

    """

    Изменение свойства окна, которое определяет, можно ли его растягивать в длину и в ширину

    Автор Ряднов И.М.

    :return: None

    """

    is\_resizable = window\_resize\_var.get()

    root.resizable(is\_resizable, is\_resizable)

    update\_ini\_value('ResizableHeight', is\_resizable)

    update\_ini\_value('ResizableWidth', is\_resizable)

if \_\_name\_\_ == "\_\_main\_\_":

    settings = get\_settings()

    root = tk.Tk()

    root.title("Редактор справочников")

    tab\_control = ttk.Notebook(root)

    tab1 = ttk.Frame(tab\_control)

    tab2 = ttk.Frame(tab\_control)

    tab3 = ttk.Frame(tab\_control)

    tab4 = ttk.Frame(tab\_control)

    ttk.Style().configure("Treeview", background=settings[4],

                          foreground="black", fieldbackground="white", font=(settings[6], settings[5], settings[7]))

    tab\_control.add(tab1, text='Товары')

    tab\_control.add(tab2, text='Заказы')

    tab\_control.add(tab3, text='Состав заказов')

    tab\_control.add(tab4, text='Полная таблица')

    tab\_control.grid(column=0, row=0, rowspan=6, columnspan=2, sticky='nswe')

    path = f'{os.getcwd()}\\data'

    GOODS = pd.read\_csv(f"{path}\MOCK\_DATA\_1.csv")

    goods\_table = create\_table(tab1, GOODS)

    ORDERS = pd.read\_csv(f"{path}\MOCK\_DATA\_2.csv")

    orders\_table = create\_table(tab2, ORDERS)

    ORDERS\_STRUCTURE = pd.read\_csv(f"{path}\MOCK\_DATA\_3.csv")

    orders\_structure\_table = create\_table(tab3, ORDERS\_STRUCTURE)

    MERGED = merge\_files(GOODS, ORDERS, ORDERS\_STRUCTURE)

    merged\_table = create\_table(tab4, MERGED)

    ttk.Button(root, text='Текстовый отчёт', command=report\_1).grid(column=2, row=0, sticky="nesw")

    ttk.Button(root, text='Статистический отчёт', command=create\_statistic\_report).grid(column=2, row=1, sticky="nesw")

    ttk.Button(root, text='Сводная таблица', command=create\_pivot\_table).grid(column=2, row=2, sticky="nesw")

    ttk.Button(root, text='Гистограмма', command=create\_hist).grid(column=2, row=3, sticky="nesw")

    ttk.Button(root, text='Стобчатая диаграмма', command=create\_bar).grid(column=2, row=4, sticky="nesw")

    ttk.Button(root, text='Блочная диаграмма', command=lambda: report\_price\_by\_category(GOODS)) \

        .grid(column=2, row=5, sticky="nesw")

    ttk.Button(root, text='Диаграмма рассеяния', command=create\_scatter).grid(column=2, row=6, sticky="nesw")

    ttk.Button(root, text='Добавить заказ', command=add\_order).grid(column=0, row=6, sticky='nesw')

    ttk.Button(root, text='Добавить товар', command=add\_product).grid(column=1, row=6, sticky='nesw')

    menu\_bar = tk.Menu(root)

    root.config(menu=menu\_bar)

    file\_menu = tk.Menu(menu\_bar, tearoff=False)

    menu\_bar.add\_cascade(label="Файл", menu=file\_menu)

    file\_menu.add\_command(label="Сохранить", command=lambda: save\_tables(GOODS, ORDERS, ORDERS\_STRUCTURE))

    file\_menu.add\_command(label="Сохранить как", command=new\_save)

    edit\_menu = tk.Menu(menu\_bar, tearoff=False)

    menu\_bar.add\_cascade(label="Изменить", menu=edit\_menu)

    edit\_menu.add\_command(label="Удалить запись", command=del\_line)

    edit\_menu.add\_command(label="Изменить запись", command=edit\_line)

    settings\_menu = tk.Menu(menu\_bar, tearoff=False)

    menu\_bar.add\_cascade(label="Настройки", menu=settings\_menu)

    settings\_menu.add\_command(label="Изменить цвет", command=config\_color)

    resolution\_menu = tk.Menu(settings\_menu, tearoff=False)

    settings\_menu.add\_cascade(label="Разрешение экрана", menu=resolution\_menu)

    resolution\_menu.add\_command(label="800x600", command=lambda: change\_resolution(800, 600))

    resolution\_menu.add\_command(label="1024x768", command=lambda: change\_resolution(1024, 768))

    resolution\_menu.add\_command(label="1280x720", command=lambda: change\_resolution(1280, 720))

    resolution\_menu.add\_command(label="1500x1500", command=lambda: change\_resolution(1500, 1500))

    font\_menu = tk.Menu(settings\_menu, tearoff=False)

    settings\_menu.add\_cascade(label="Шрифт", menu=font\_menu)

    window\_resize\_var = tk.BooleanVar()

    window\_resize\_var.set(True)

    settings\_menu.add\_checkbutton(label="Изменять размер окна", variable=window\_resize\_var,

                                  command=toggle\_window\_resize)

    font\_menu.add\_command(label="Изменить", command=open\_font\_settings)

    config\_widgets(root, 7, 3)

    root.geometry(

        f"{settings[0]}x{settings[1]}".format(root.winfo\_screenwidth() // 2 - 400,

                                              root.winfo\_screenheight() // 2 - 300))

    root.resizable(settings[2], settings[3])

    root.mainloop()

# Листинг charts\_and\_reports.py

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.ticker import FuncFormatter

def merge\_files(PRODUCT, ORDERS, TOTAL\_ORDERS):

    """

    Мердж данных

    Автор: Ряднов И.М.

    :param PRODUCT(pd.DataFrame): датафрейм с товарами

    :param ORDERS(pd.DataFrame): датафрейм с заказами

    :param TOTAL\_ORDERS(pd.DataFrame): датафрейм со всеми заказами

    :return merged(pd.DataFrame): объединенный датафрейм

    """

    merged = pd.merge(TOTAL\_ORDERS, PRODUCT, on="Product ID")

    merged = pd.merge(merged, ORDERS, on="Order ID")

    return merged

def report\_about\_firm(merged, start\_date, end\_date, category):

    """

    Отчет о продажах товаров выбранной компании за выбранный временной промежуток

    Автор: Болезнов С.А.

    :param merged(pd.DataFrame): датафрейм со всеми данными

    :param start\_date(str): начальная дата

    :param end\_date(str): конечная дата

    :param category(str): категория

    :return result(pd.DataFrame): полученный отчет

    """

    merged['Date'] = merged['Date'].astype("datetime64[ns]")

    selector = (merged['Date'].between(start\_date, end\_date)) & (

            merged['Category'] == category)

    result = merged.loc[selector, ['Product',

                                   'Description', 'Price', 'Quantity', 'Sum']]

    return result

def generate\_frequency\_table(data, attribute):

    """

    Отчет о частоте встречаемости атрибута

    Автор: Ряднов И.М.

    :param data(pd.DataFrame): датафрейм со всеми данными

    :param attribute(str): имя атрибута

    :return freq\_table(pd.DataFrame): отчёт о частотах

    """

    freq\_table = data[attribute].value\_counts().reset\_index()

    freq\_table.columns = [attribute, 'Частота']

    freq\_table['Процент'] = (freq\_table['Частота'] / len(data)) \* 100

    return freq\_table

def generate\_descriptive\_stats(data, attribute):

    """

    Отчет-описание атрибутов

    Автор: Ряднов И.М.

    :param data(pd.DataFrame): датафрейм со всеми данными

    :param attribute(str): имя атрибута

    :return descriptive\_stats(pd.DataFrame): отчёт описания

    """

    descriptive\_stats = data[attribute].describe().reset\_index()

    descriptive\_stats.columns = ['Статистика', attribute]

    descriptive\_stats['Статистика'] = descriptive\_stats['Статистика'].replace({

        'count': 'Количество непропущенных значений',

        'mean': 'Среднее значение',

        'max': 'Максимум',

        'std': 'Стандартное отклонение',

        'min': 'Минимум',

        '25%': 'Первый квартиль',

        '50%': 'Медиана',

        '75%': 'Третий квартиль',

        'unique': 'Количество уникальных значений',

        'top': 'Наиболее часто встречающееся значение',

        'freq': 'Количество раз, которое top значение встречается'

    })

    if np.issubdtype(data[attribute].dtype, np.number):

        sample\_variance = data[attribute].var()

        descriptive\_stats.loc[len(descriptive\_stats)] = ['Выборочная дисперсия', sample\_variance]

    return descriptive\_stats

def generate\_attribute\_report(data, attribute1, attribute2):

    """

    Генерация отчётов о паре атрибутов

    Автор: Ряднов И.М.

    :param data(pd.DataFrame): датафрейм со всеми данными

    :param attribute1(str): имя первого атрибута

    :param attribute2(str): имя второго атрибута

    :return freq\_table1(pd.DataFrame): отчёт о частотах первого атрибута

    :return stats\_table1(pd.DataFrame): отчёт описания второго атрибута

    :return freq\_table2(pd.DataFrame): отчёт о частотах первого атрибута

    :return stats\_table2(pd.DataFrame): отчёт описания второго атрибута

    """

    freq\_table1 = generate\_frequency\_table(data, attribute1)

    stats\_table1 = generate\_descriptive\_stats(data, attribute1)

    freq\_table2 = generate\_frequency\_table(data, attribute2)

    stats\_table2 = generate\_descriptive\_stats(data, attribute2)

    return (freq\_table1, stats\_table1, freq\_table2, stats\_table2)

def report\_day\_sales(data):

    """

    Функция для отображения графика продаж по дням

    Автор: Ряднов И.М.

    :param data(pd.DataFrame): Данные таблицы 2

    :return: None

    """

    months = {

        1: "Январь",

        2: "Февраль",

        3: "Март",

        4: "Апрель",

        5: "Май",

        6: "Июнь",

        7: "Июль",

        8: "Август",

        9: "Сентябрь",

        10: "Октябрь",

        11: "Ноябрь",

        12: "Декабрь"

    }

    if data.index.name is None:

        data['Date'] = data['Date'].astype("datetime64[ns]")

        data.set\_index('Date', inplace=True)

    sales\_by\_day = data['Sum'].resample('D').sum()

    x\_labels = []

    for date in sales\_by\_day.index:

        if date.day % 5 == 0:

            month = months.get(date.month)

            day = str(date.day)

            x\_labels.append(f'{month} {day}')

        else:

            x\_labels.append('')

    plt.bar(sales\_by\_day.index, sales\_by\_day, width=0.7)

    plt.xlabel('Дата')

    plt.ylabel('Продажи')

    plt.title('Продажи по дням')

    plt.xticks(sales\_by\_day.index, x\_labels, rotation=60, ha='right')

    formatter = FuncFormatter(format\_func)

    plt.gca().yaxis.set\_major\_formatter(formatter)

    plt.show()

def report\_week\_sales(data):

    """

    Функция для отображения графика продаж по неделям

    Автор: Ряднов И.М.

    :param data(pd.DataFrame): Данные таблицы 2

    :return: None

    """

    if data.index.name is None:

        data['Date'] = data['Date'].astype("datetime64[ns]")

        data.set\_index('Date', inplace=True)

    sales\_by\_week = data['Sum'].resample('W').sum()

    plt.bar(sales\_by\_week.index, sales\_by\_week, width=3)

    plt.xlabel('Дата')

    plt.ylabel('Продажи')

    plt.title('Продажи по неделям')

    plt.xticks(sales\_by\_week.index, rotation=60, ha='right')

    formatter = FuncFormatter(format\_func)

    plt.gca().yaxis.set\_major\_formatter(formatter)

    plt.show()

def report\_month\_sales(data):

    """

    Функция для отображения графика продаж по месяцам

    Автор: Ряднов И.М.

    :param data(pd.DataFrame): Данные таблицы 2

    :return: None

    """

    if data.index.name is None:

        data['Date'] = data['Date'].astype("datetime64[ns]")

        data.set\_index('Date', inplace=True)

    sales\_by\_month = data['Sum'].resample('M').sum()

    plt.bar(sales\_by\_month.index, sales\_by\_month, width=10)

    plt.xlabel('Дата')

    plt.ylabel('Продажи')

    plt.title('Продажи по месяцам')

    plt.xticks(sales\_by\_month.index, rotation=60, ha='right')

    for i, value in enumerate(sales\_by\_month):

        plt.text(sales\_by\_month.index[i], value + 100, f"{str(value)}$", ha='center', va='bottom')

    formatter = FuncFormatter(format\_func)

    plt.gca().yaxis.set\_major\_formatter(formatter)

    plt.show()

def report\_year\_sales(data):

    """

    Функция для отображения графика продаж по годам

    Автор: Ряднов И.М.

    :param data(pd.DataFrame): Данные таблицы 2

    :return: None

    """

    if data.index.name is None:

        data['Date'] = data['Date'].astype("datetime64[ns]")

        data.set\_index('Date', inplace=True)

    sales\_by\_year = data['Sum'].resample('Y').sum()

    plt.bar(sales\_by\_year.index, sales\_by\_year, width=10)

    plt.xlabel('Дата')

    plt.ylabel('Продажи')

    plt.title('Продажи по годам')

    plt.xticks(sales\_by\_year.index, rotation=60, ha='right')

    for i, value in enumerate(sales\_by\_year):

        plt.text(sales\_by\_year.index[i], value + 100, f"{str(value)}$", ha='center', va='bottom')

    formatter = FuncFormatter(format\_func)

    plt.gca().yaxis.set\_major\_formatter(formatter)

    plt.show()

def format\_func(value, tick\_number):

    """

    Функция для изменения отображения больших значений на осях графиков

    Автор: Ряднов И.М.

    :param value(int): значение показателя на графике

    :param tick\_number(int): количество tick

    :return value(str): строка с измененным форматом числа

    """

    if value >= 1000000:

        value = f'{int(value / 1000000)}M'

    elif value >= 1000:

        value = f'{int(value / 1000)}K'

    return value

def report\_price\_by\_category(data: pd.DataFrame) -> None:

    """

    Функция для отображения графика распределения цен по категориям товаров

    Автор: Ряднов И.М.

    :param data(pd.DataFrame): Данные таблицы 1

    :return: None

    """

    data['Price'] = data['Price'].replace(

        '[\$,]', '', regex=True).astype(float)

    categories = data.groupby('Category')

    prices\_by\_category = [category[1]['Price'].tolist()

                          for category in categories]

    plt.boxplot(prices\_by\_category)

    plt.xticks(range(1, len(categories) + 1),

               [category[0] for category in categories], rotation=60, ha='right')

    plt.title(

        'Распределение цен по категориям товаров')

    plt.xlabel('Категория товара')

    plt.ylabel('Цена')

    plt.show()

def histogram(data: pd.DataFrame, colomn: str):

    """

    Создание гистограммы

    Автор: Болезнов С.А.

    :param data: Датафрейм с данными

    :param colomn: Тип гистограммы

    :return: None

    """

    if colomn == 'Quantity':

        plt.hist(data[colomn], bins=20, color='blue', edgecolor='black')

        plt.title('Гистограмма распределения количества\nтоваров в заказе')

        plt.xlabel('Количество товаров в заказе')

        plt.ylabel('Количество')

    if colomn == 'Price':

        prices = [int(str(x)[:-3]) for x in data[colomn]]

        plt.hist(prices, bins=30, color='blue', edgecolor='black')

        plt.title('Гистограмма распределения цены на товары')

        plt.xlabel('Цена товара')

        plt.ylabel('Количество')

    if colomn == 'Sum':

        plt.hist(data[colomn], bins=30, color='blue', edgecolor='black')

        plt.title('Гистограмма распределения суммы заказов')

        plt.xlabel('Сумма заказа')

        plt.ylabel('Количество')

    plt.show()

def report\_price\_by\_quantity(data):

    """

    Функция для отображения графика распределения количества заказанных товаров по их цене

    Автор: Ряднов И.М.

    :param data(pd.DataFrame): Данные таблицы 3

    :return: None

    """

    plt.scatter(data['Price'], data['Quantity'])

    plt.xlabel('Цена')

    plt.ylabel('Количество')

    plt.title('Распределение количества заказанных товаров по их цене')

    x\_labels = np.arange(int(data['Price'].min()),

                         int(data['Price'].max()) + 100, 25)

    plt.xticks(x\_labels, rotation=90, ha='right')

    plt.yticks(

        list(range(int(data['Quantity'].min()), int(data['Quantity'].max()) + 1)))

    plt.show()

# Листинг utils.py

import os

import pandas as pd

from tkinter import filedialog as fld

def save\_tables(GOODS: pd.DataFrame, ORDERS: pd.DataFrame, ORDERS\_STRUCTURE: pd.DataFrame):

    """

    Сохранение всех таблиц

    Автор: Болезнов С.А.

    :param GOODS: Датафрейм товаров

    :param ORDERS: Датафрейм заказов

    :param ORDERS\_STRUCTURE: Датафрейм содержания заказов

    :return: None

    """

    path = f'{os.getcwd()}\\data'

    GOODS.to\_csv(f'{path}/MOCK\_DATA\_1.csv', index=False)

    ORDERS.to\_csv(f'{path}/MOCK\_DATA\_2.csv', index=False)

    ORDERS\_STRUCTURE.to\_csv(f'{path}/MOCK\_DATA\_3.csv', index=False)

def save\_as(table: pd.DataFrame):

    """

    Сохранение отдельной таблицы

    Автор: Болезнов С.А.

    :param table: таблица

    :return: None

    """

    path = f'{os.getcwd()}\\output'

    ftypes = [('csv', '.csv'), ('excel', '.xlsx'), ('pickle', '.pickle')]

    filepath = fld.asksaveasfilename(filetypes=ftypes, initialdir=path, defaultextension='csv')

    if filepath != "":

        file\_format = filepath.split('.')[-1]

        if file\_format == 'csv':

            table.to\_csv(filepath, index=False)

        elif file\_format == 'xlsx':

            table.to\_excel(filepath, index=False)

        elif file\_format == 'pick':

            table.to\_pickle(filepath)

def read\_ini\_file():

    """

    Функция для чтения конфигурации .ini

    Автор: Ряднов И.М.

    :return config(dict): Словарь с конфигурацией

    """

    config = {}

    current\_section = None

    with open(f"{os.getcwd()}\scripts\config.ini", 'r') as file:

        for line in file:

            line = line.strip()

            if not line or line.startswith(';'):

                continue

            if line.startswith('[') and line.endswith(']'):

                current\_section = line[1:-1]

                config[current\_section] = {}

            else:

                key, value = line.split('=', 1)

                config[current\_section][key.strip()] = value.strip()

    return config

def write\_ini\_file(config):

    """

    Функция для перезаписи новой конфигурации .ini

    Автор: Ряднов И.М.

    :param key(dict): словарь с конфигурацией

    :return None

    """

    with open(f"{os.getcwd()}\scripts\config.ini", 'w') as file:

        for section, values in config.items():

            file.write(f'[{section}]\n')

            for key, value in values.items():

                file.write(f'{key} = {value}\n')

            file.write('\n')

def update\_ini\_value(key, value):

    """

    Функция для изменения определенного параметра конфигурации .ini

    Автор: Ряднов И.М.

    :param key(str): название конфигурации

    :param value: новое значение

    :return None

    """

    config = read\_ini\_file()

    config['Settings'][key] = value

    write\_ini\_file(config)