Министерство науки и образования РФ

Федеральное государственное бюджетное учреждение

высшего образования

**«Тверской государственный технический университет»**

(ТвГТУ)

Кафедра программного обеспечения

**Отчет по лабораторной работе №5**

По дисциплине: «Новые технологии в РПС»

Тема: «Подготовка приложения к дальнейшей работе»

|  |
| --- |
| Выполнил:  студент группы  М.ПИН.РИС - 23.08  Тимофеев Александр Сергеевич |
| Проверила:  старший преподаватель  кафедры ПО  Корнеева Е.И. |

Тверь 2023

**Постановка задачи:**

1. Добавление функционала 4 лабораторной работы в интерфейс 3й
2. Создание .bat, .sh, Dockerfile

**Код программы (Python):**

**Запуск приложения на Windows (.bat)**

@echo off

python -m pip install --upgrade pip

pip install -r requirements.txt

python main\_form.py

pause

**Запуск приложения на Linux (.sh)**

#!/bin/bash

python3 -m pip install --upgrade pip

pip install -r requirements.txt

python3 main\_form.py

**Запуск приложения через Docker (Dockerfile)**

FROM python:3.11

WORKDIR /app

ADD . /app

RUN pip install --no-cache-dir -r requirements.txt

CMD ["python", "./src/main.py"]

**Главная форма**

Библиотеки главной формы:

import pandas as pd

from datetime import datetime

from PyQt6.QtWidgets import QApplication, QMainWindow, QPushButton, QVBoxLayout, QWidget, QFileDialog, QLineEdit, QTableWidget, QTableWidgetItem, QHBoxLayout, QLabel, QMessageBox, QComboBox

from PyQt6.QtCore import QDate

from dataset\_handler import DatasetHandler

from operations\_dataset import DatasetOperations

from directory\_handler import DirectoryHandler as dir\_h

from form\_images import ImageWindow

from form\_currency import CurrencyWindow

Константы:

CURRENCY\_FIELDS = ['date', 'nominal', 'value', 'vunitRate']

IMAGES\_FIELDS = ['date', 'file\_name', 'url', 'path']

CURR\_DIR = dir\_h.set\_current\_dir()

Инициализация главной формы:  
def \_\_init\_\_(self):

        super().\_\_init\_\_()

        self.folder\_path = ""

        self.df = pd.DataFrame()

        self.dataset\_operations = DatasetOperations(CURR\_DIR)

        self.is\_loaded\_dataset = False

        self.is\_nan\_check = False

        self.is\_rename\_columns = False

        self.is\_add\_median\_mean = False

        self.init\_ui()

        self.choose\_dataset\_from\_file()

Инициализация элементов пользовательского интерфейса:

    def init\_ui(self):

        self.setWindowTitle('Операции с датасетом')

        v\_layout = QVBoxLayout()

        self.button\_choose\_dataset = QPushButton('Выбор датасета')

        self.button\_choose\_dataset.clicked.connect(self.choose\_dataset\_from\_file)

        self.combo\_box\_fields = QComboBox()

        self.combo\_box\_fields.addItem("'date', 'nominal', 'value', 'vunitRate'")

        self.combo\_box\_fields.addItem("'date', 'file\_name', 'url', 'path'")

        self.textbox\_date = QLineEdit()

        self.textbox\_date.setPlaceholderText('Введите дату')

        self.button\_search = QPushButton('Получить данные')

        self.button\_search.clicked.connect(self.get\_data\_with\_date)

        self.label\_separation\_operations = QLabel('Разделение по ...')

        self.button\_separation\_date\_by\_data = QPushButton('... даты от данных')

        self.button\_separation\_date\_by\_data.clicked.connect(self.separation\_date\_by\_data)

        self.button\_separation\_by\_years = QPushButton('... годам')

        self.button\_separation\_by\_years.clicked.connect(self.separation\_by\_years)

        self.button\_separation\_by\_weeks = QPushButton('... неделям')

        self.button\_separation\_by\_weeks.clicked.connect(self.separation\_by\_weeks)

        self.button\_show\_images = QPushButton('Просмотр данных')

        self.button\_show\_images.clicked.connect(self.show\_form\_data)

        v\_layout.addWidget(self.button\_choose\_dataset)

        v\_layout.addWidget(self.combo\_box\_fields)

        v\_layout.addWidget(self.textbox\_date)

        v\_layout.addWidget(self.button\_search)

        v\_layout.addWidget(self.label\_separation\_operations)

        v\_layout.addWidget(self.button\_separation\_date\_by\_data)

        v\_layout.addWidget(self.button\_separation\_by\_years)

        v\_layout.addWidget(self.button\_separation\_by\_weeks)

        v\_layout.addWidget(self.button\_show\_images)

        self.label\_info\_dataset = QLabel('Информация о датасете')

        self.button\_info\_of\_dataset = QPushButton('Информация')

        self.button\_info\_of\_dataset.clicked.connect(self.info\_of\_dataset)

        self.button\_rename\_columns = QPushButton('Переименовать колонки')

        self.button\_rename\_columns.clicked.connect(self.rename\_columns)

        self.button\_check\_null\_fields = QPushButton('Проверка на NaN')

        self.button\_check\_null\_fields.clicked.connect(self.check\_null\_fields)

        self.button\_check\_null\_fields.setEnabled(False)

        self.button\_median\_mean = QPushButton('Медианы и средний')

        self.button\_median\_mean.clicked.connect(self.median\_mean)

        self.button\_median\_mean.setEnabled(False)

        self.button\_describe\_dataset = QPushButton('Статистическая информация')

        self.button\_describe\_dataset.clicked.connect(self.describe\_dataset)

        self.button\_describe\_dataset.setEnabled(False)

        self.label\_graphs = QLabel('Графики датасета')

        self.button\_show\_deviation\_graph = QPushButton('График отклонения')

        self.button\_show\_deviation\_graph.clicked.connect(self.show\_deviation\_graph)

        self.button\_show\_deviation\_graph.setEnabled(False)

        self.button\_show\_date\_graph = QPushButton('График изменения курса')

        self.button\_show\_date\_graph.clicked.connect(self.show\_date\_graph)

        self.button\_show\_date\_graph.setEnabled(False)

        self.textbox\_date\_graph = QLineEdit()

        self.textbox\_date\_graph.setPlaceholderText('Введите дату (<месяц> ИЛИ <год> ИЛИ <месяц год>)')

        self.textbox\_date\_graph.setEnabled(False)

        self.button\_show\_graph\_by\_date = QPushButton('График изменения курса по дате')

        self.button\_show\_graph\_by\_date.clicked.connect(self.show\_graph\_by\_date)

        self.button\_show\_graph\_by\_date.setEnabled(False)

        v\_layout.addWidget(self.label\_info\_dataset)

        v\_layout.addWidget(self.button\_info\_of\_dataset)

        v\_layout.addWidget(self.button\_rename\_columns)

        v\_layout.addWidget(self.button\_check\_null\_fields)

        v\_layout.addWidget(self.button\_median\_mean)

        v\_layout.addWidget(self.button\_describe\_dataset)

        v\_layout.addWidget(self.label\_graphs)

        v\_layout.addWidget(self.button\_show\_deviation\_graph)

        v\_layout.addWidget(self.button\_show\_date\_graph)

        v\_layout.addWidget(self.textbox\_date\_graph)

        v\_layout.addWidget(self.button\_show\_graph\_by\_date)

        v\_layout.addStretch(1) #чтобы кнопки не расплылись по форме от горизонтального слоя

        self.table = QTableWidget()

        h\_layout = QHBoxLayout()

        h\_layout.addLayout(v\_layout)

        h\_layout.addWidget(self.table)

        container = QWidget()

        container.setLayout(h\_layout)

        self.setCentralWidget(container)

Метод выбора датасета из директории:

    def choose\_dataset\_from\_file(self):

        try:

            dat\_h = DatasetHandler()

            file\_dialog = QFileDialog()

            self.folder\_path = file\_dialog.getOpenFileName()[0]

            if(self.df is None or self.df.empty):

                self.df = dat\_h.create\_dataset([self.folder\_path])

                self.current\_fields = self.df.columns.tolist()

                self.is\_loaded\_dataset = True

            else:

                self.df = dat\_h.create\_dataset\_from\_files([self.folder\_path], self.split\_fields(self.combo\_box\_fields.currentText()))#self.combo\_box\_fields.currentText())

                self.current\_fields = self.df.columns.tolist()

                self.is\_loaded\_dataset = True

            self.update\_table()

        except Exception as e:

            self.show\_message\_box('Ошибка', str(e))

Метод вывода информации о датасете:

    def info\_of\_dataset(self):

        try:

            if(self.is\_loaded\_dataset == True):

                self.show\_message\_box('Информация о датасете', self.dataset\_operations.info\_of\_dataset(self.df))

        except Exception as e:

            self.show\_message\_box('Ошибка', e)

Метод переименования названия столбцов датасета:

    def rename\_columns(self):

        try:

            if(self.is\_loaded\_dataset == True):

                self.df.columns = [self.dataset\_operations.rename\_columns(col) for col in self.df.columns]

                self.is\_rename\_columns = True

                self.update\_table()

                self.button\_check\_null\_fields.setEnabled(True)

        except Exception as e:

            self.show\_message\_box('Ошибка', e)

Метод проверки на пустые поля:

    def check\_null\_fields(self):

        try:

            if(self.is\_rename\_columns == True):

                s = self.dataset\_operations.check\_null\_fields(self.df)

                self.dataset\_operations.fill\_null\_fieds(self.df)

                self.update\_table()

                self.show\_message\_box('Проверка на NaN Null', s)

                self.is\_nan\_check = True

                self.button\_median\_mean.setEnabled(True)

        except Exception as e:

            self.show\_message\_box('Ошибка', e)

Метод добавления отклонения и среднего значения к датасету:

    def median\_mean(self):

        try:

            if(self.is\_nan\_check == True):

                self.dataset\_operations.add\_median\_mean(self.df)

                self.is\_add\_median\_mean = True

                self.update\_table()

                self.button\_describe\_dataset.setEnabled(True)

                self.button\_show\_deviation\_graph.setEnabled(True)

                self.button\_show\_date\_graph.setEnabled(True)

                self.textbox\_date\_graph.setEnabled(True)

                self.button\_show\_graph\_by\_date.setEnabled(True)

        except Exception as e:

            self.show\_message\_box('Ошибка', e)

Метод вывода статистики датасета:

    def describe\_dataset(self):

        try:

            if(self.is\_add\_median\_mean == True):

                self.show\_message\_box('Статистическая информация', str(self.dataset\_operations.describe\_dataset(self.df)))

        except Exception as e:

            self.show\_message\_box('Ошибка', e)

Метод вывода на экран формы графика отклонений:

    def show\_deviation\_graph(self):

        try:

            if(self.is\_add\_median\_mean == True):

                self.dataset\_operations.show\_deviation\_graph(self.df)

        except Exception as e:

            self.show\_message\_box('Ошибка', e)

Метод вывода на экран формы графика курса по всему периоду:

    def show\_date\_graph(self):

        try:

            if(self.is\_add\_median\_mean == True):

                self.dataset\_operations.show\_date\_graph(self.df)

        except Exception as e:

            self.show\_message\_box('Ошибка', e)

Метод вывода на экран формы графика курса за заданную дату:

    def show\_graph\_by\_date(self):

        try:

            if(self.is\_add\_median\_mean == True):

                res = self.textbox\_date\_graph.text().split()

                if len(res) == 1:

                    if len(res[0]) == 4:

                        self.dataset\_operations.plot\_rate\_year(self.df, int(res[0]))

                    else:

                        self.dataset\_operations.plot\_rate\_month(self.df, int(res[0]))

                else:

                    self.dataset\_operations.plot\_rate\_year\_month(self.df, int(res[1]), int(res[0]))

        except Exception:

            self.show\_message\_box('Ошибка', 'Дата введена неправильно! Пример ввода <месяц>, <год>, <месяц год>')

Класс «операции над датасетом»

Функция получения информации о датасете:

    def info\_of\_dataset(self, df: pd.DataFrame) -> str:

        buf = io.StringIO()

        df.info(buf=buf)

        return buf.getvalue()

Функция переименования колонок датасета:

    def rename\_columns(self, s: str) -> str:

        return ''.join(['\_' + i.lower() if i.isupper() else i for i in s]).lstrip('\_')

Функция проверки пустых полей в датасете:

    def check\_null\_fields(self, df: pd.DataFrame) -> str:

        return str(df.isnull().sum())

Функция заполнения пустых полей датасета:

    def fill\_null\_fieds(self, df: pd.DataFrame) -> None:

        df.ffill(inplace=True)

        df.bfill(inplace=True)

Добавление медианы и среднего значения к датасету:

    def add\_median\_mean(self, df: pd.DataFrame) -> None:

        median\_value = df['vunit\_rate'].median()

        mean\_value = df['vunit\_rate'].mean()

        df['deviation\_from\_median'] = df['vunit\_rate'] - median\_value

        df['deviation\_from\_mean'] = df['vunit\_rate'] - mean\_value

Функция описания датасета:

    def describe\_dataset(self, df: pd.DataFrame) -> str:

        return str(df[['vunit\_rate', 'deviation\_from\_median', 'deviation\_from\_mean']].describe())

Функция отрисовки графика отклонений:

    def show\_deviation\_graph(self, df: pd.DataFrame) -> None:

        plt.figure(figsize=(12, 6))

        plt.boxplot([df['vunit\_rate'], df['deviation\_from\_median'], df['deviation\_from\_mean']], labels=['vunit\_rate', 'deviation\_from\_median', 'deviation\_from\_mean'])

        plt.title('График vunit\_rate и отклонений')

        plt.show()

Функция отрисовки графика курса по всему периоду:

    def show\_date\_graph(self, df: pd.DataFrame) -> None:

        df['date'] = pd.to\_datetime(df['date'])

        plt.figure(figsize=(10, 6))

        plt.plot(df['date'], df['vunit\_rate'])

        plt.title('Изменение курса за весь период')

        plt.xlabel('Дата')

        plt.ylabel('Курс')

        plt.show()

Функция отрисовки графика:

    def plot\_rate(self, df: pd.DataFrame, title: str) -> None:

        median\_value = df['vunit\_rate'].median()

        mean\_value = df['vunit\_rate'].mean()

        plt.figure(figsize=(10, 6))

        plt.plot(df['date'], df['vunit\_rate'], label='Курс')

        plt.axhline(median\_value, color='r', linestyle='--', label='Медиана')

        plt.axhline(mean\_value, color='g', linestyle=':', label='Среднее значение')

        plt.title(title)

        plt.xlabel('Дата')

        plt.ylabel('Курс')

        plt.legend()

        plt.show()

Методы вызывающие отрисовку графика за месяц, год, месяц и год:

    def plot\_rate\_month(self, df: pd.DataFrame, month: int) -> None:

        df\_month = df[df['date'].dt.month == month]

        self.plot\_rate(df\_month, f'Изменение курса за {month} месяц')

    def plot\_rate\_year(self, df: pd.DataFrame, year: int) -> None:

        df\_year = df[df['date'].dt.year == year]

        self.plot\_rate(df\_year, f'Изменение курса за {year} год')

    def plot\_rate\_year\_month(self, df: pd.DataFrame, year: int, month: int) -> None:

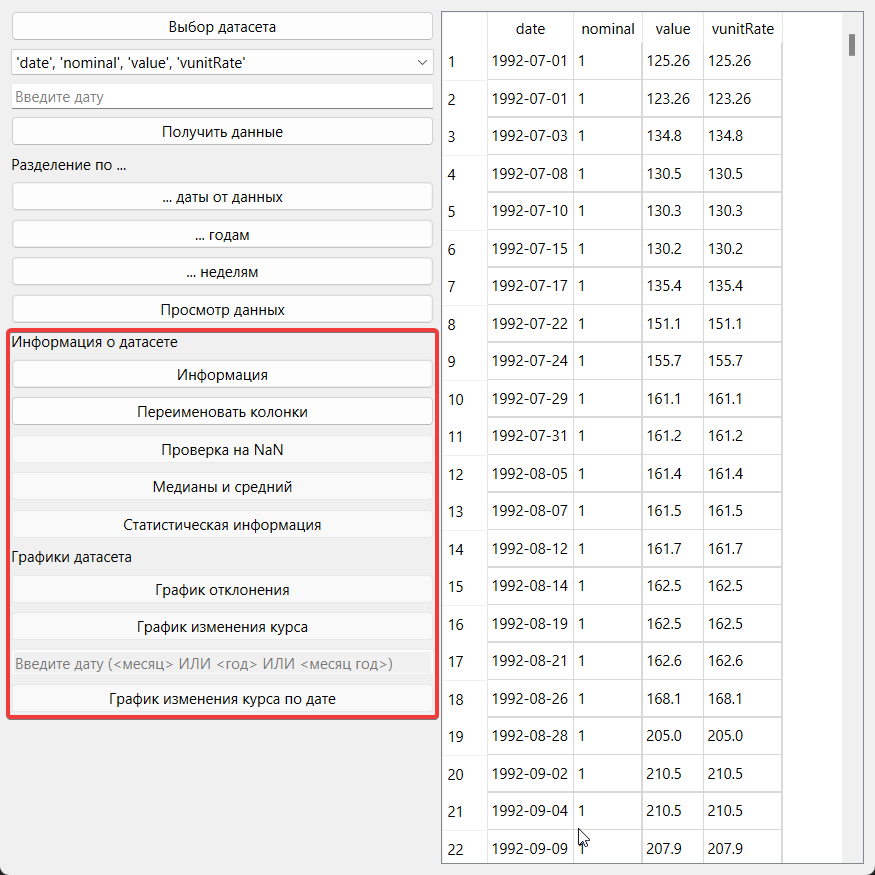
        df\_year\_month = df[(df['date'].dt.year == year) & (df['date'].dt.month == month)]

        self.plot\_rate(df\_year\_month, f'Изменение курса за {month} месяц в {year} год')

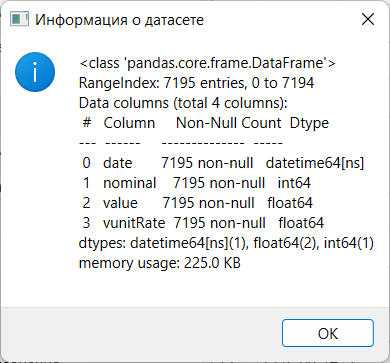
**Тестирование**

1. Главная форма (обновленная с третьей лабы):

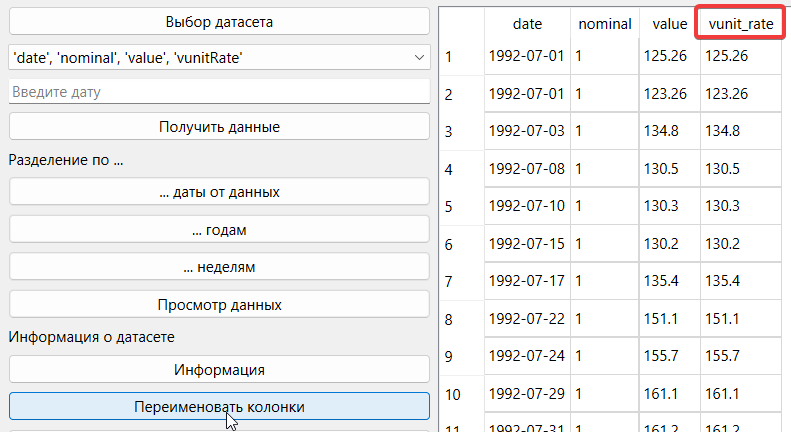
Кнопки заблокированы, чтобы не нарушить порядок действий и не вызвать лишние ошибки

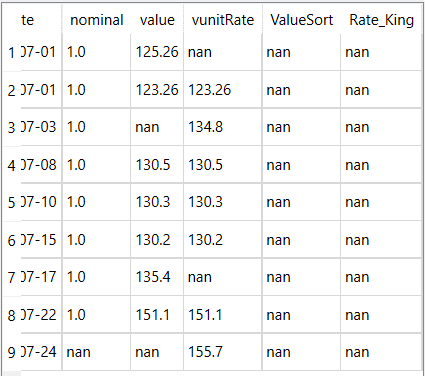
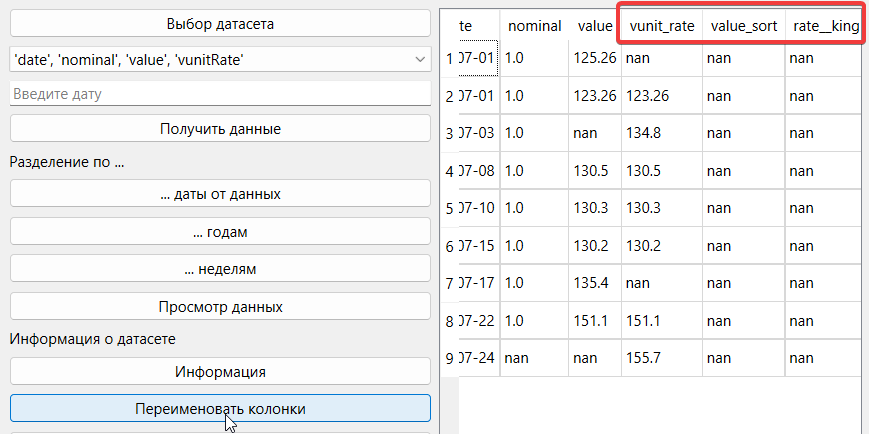
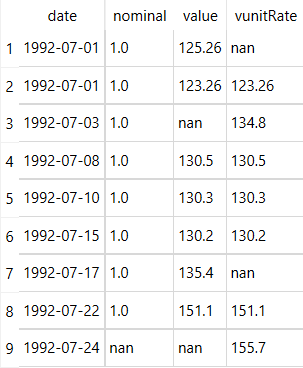
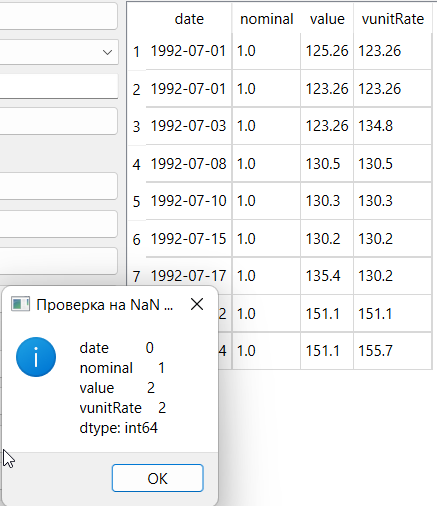
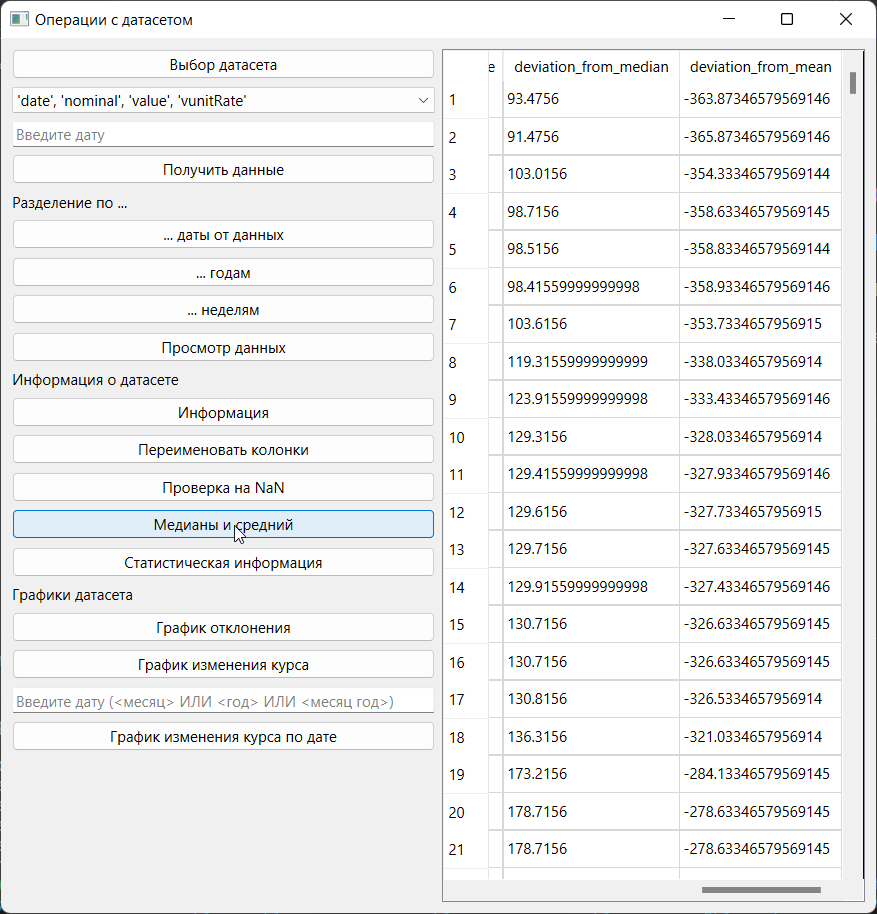
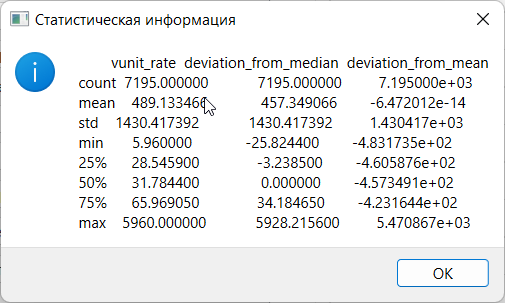
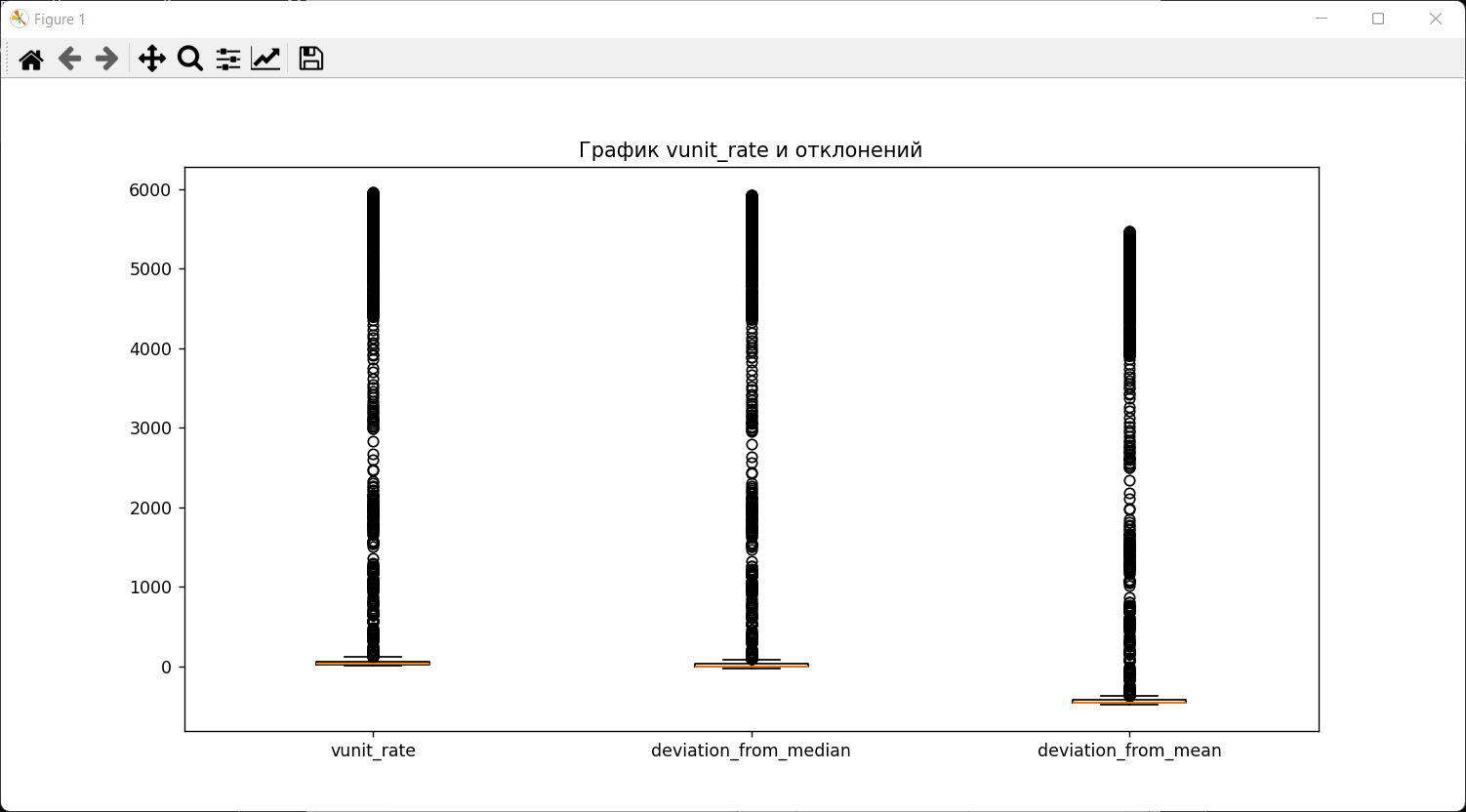
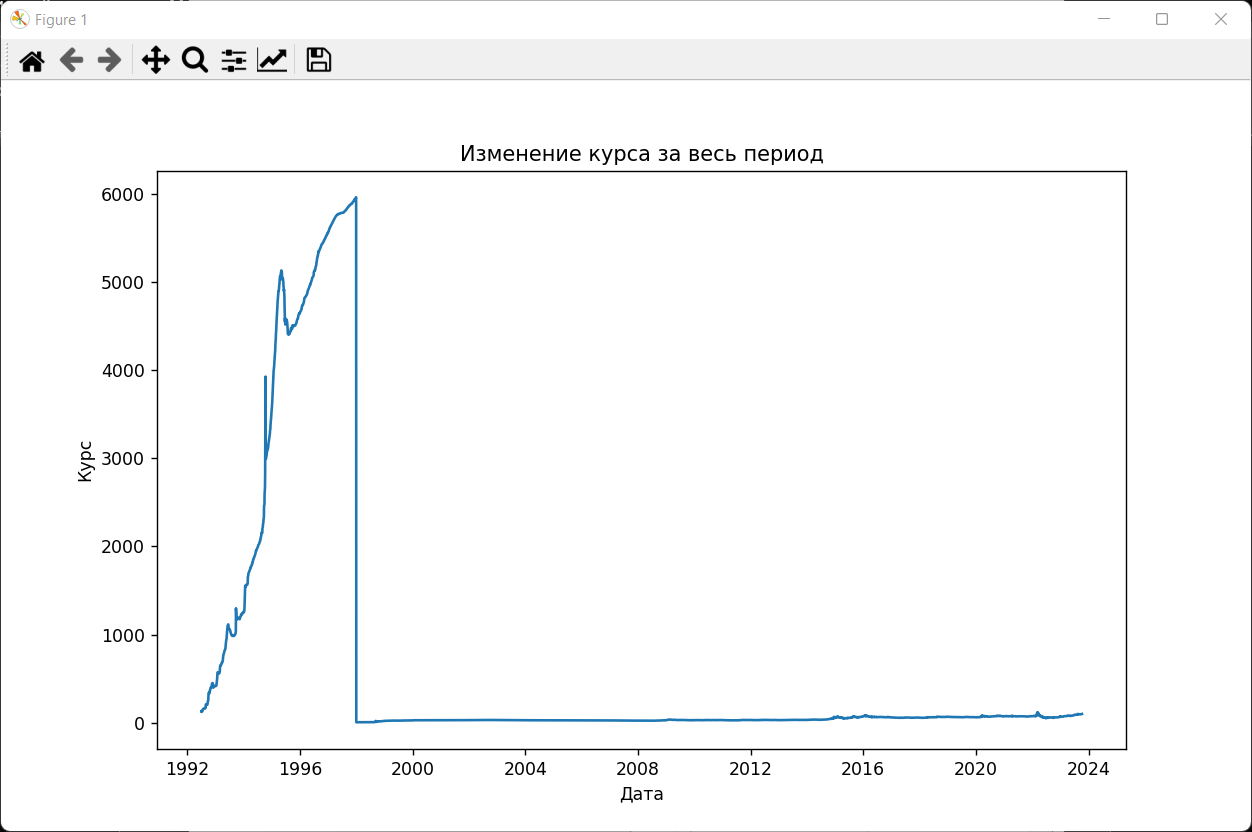
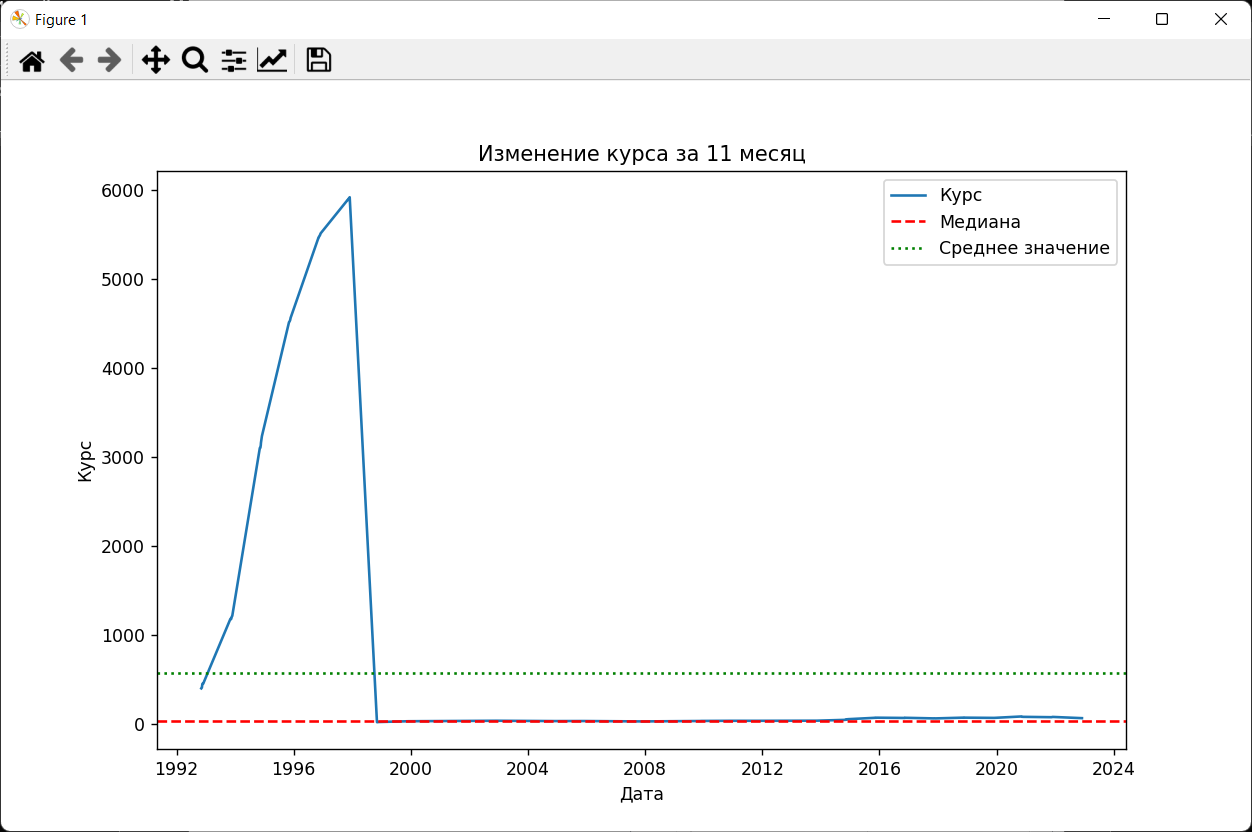
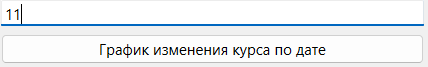
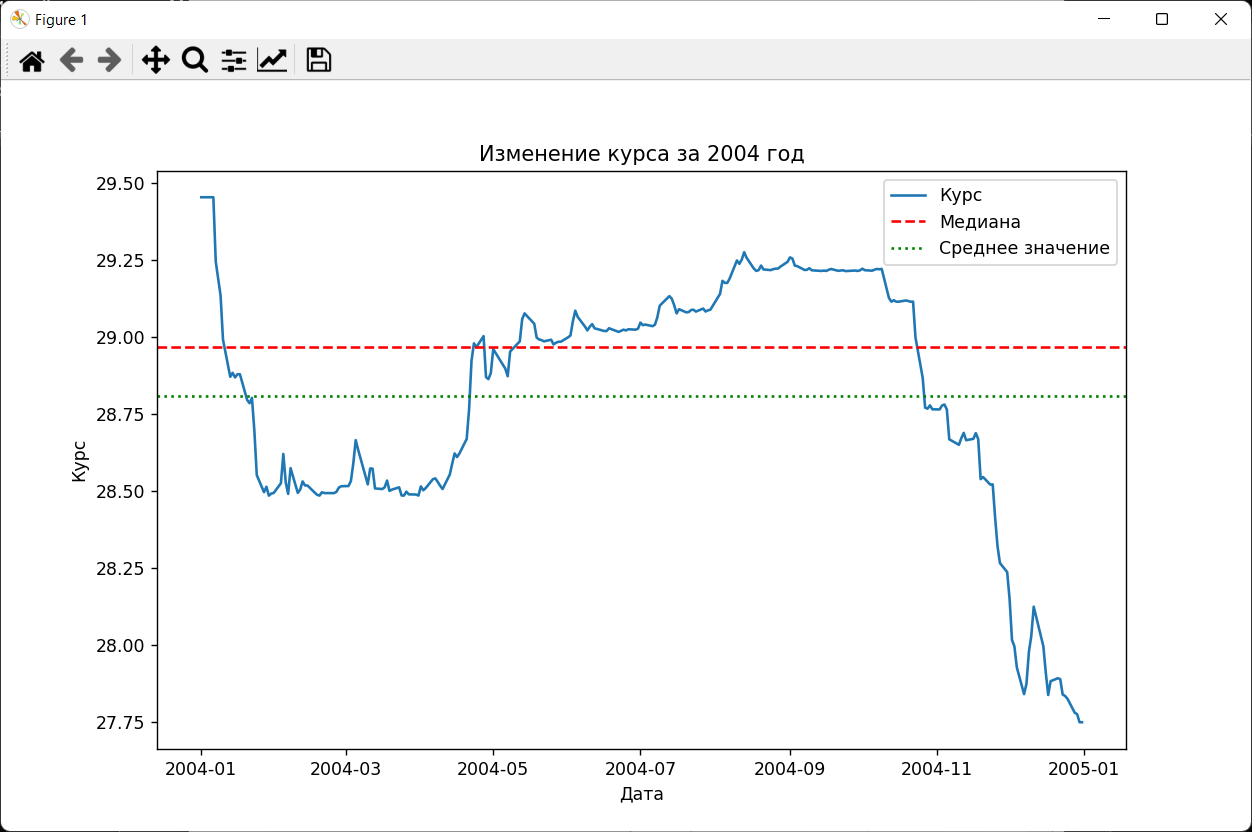
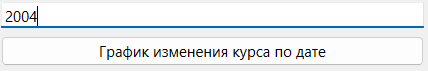


1. Информация о датасете:



1. Переименование колонок:



1. 
2. 
3. Проверка и заполнение пустых полей датасета: 
4. 
5. Добавление медианы и среднего значения: 
6. Статистическая информация о датасете: 
7. График отклонения: 
8. График изменения курса за весь период: 
9. График изменения курса за определенный период:
10. 11 месяц (ноябрь) 
11. 2004 год все месяцы 
12. 11 месяц 2004 года 