ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Московский институт электроники и математики им. А.Н. Тихонова

Руководство разработчика к приложению по базе данных с лимитированными кроссовками

Разработчики:

Данилов Евгений Владимирович

Гизатуллин Петр Олегович

Руководитель:

Полякова Марина Васильевна

Москва 2020

Автор: Данилов Евгений Владимирович
Редактор: Гизатуллин Петр Олегович

Технические требования

64-битная операционная система Windows, на которую возможна установка интерпретатора Python 3.7 (https://www.python.org/downloads/)

Версия библиотек

Данное приложение использует определенный набор библиотек языка Python, версии которых приведены ниже (табл. 1)

Библиотека	Версия
Pandas	1.0.4
Tkinter	8.6.10
Matplotlib	3.2.1
Xlrd	1.2.0

Табл. 1. Версии использованных библиотек

Структура каталогов

Данное приложение использует следующую систему каталогов (Табл. 2)

Первый уровень	Второй уровень	Объяснение
Work		Основной каталог
	Data	Базы данных
	Graphics	Копии графических отчетов
	Library	Библиотека функций
	Notes	Документация
	Output	Копии текстовых отчетов
	Scripts	Основной и дополнительные модули

Таблица 2. Структура каталогов

Архитектура приложения

Приложение состоит из модуля pythonproject.py. Скрипт pythonproject.py объединяет в себе функции графического интерфейса и обработки баз данных. Функции, включенные в этот скрипт, представлены далее в разделе «Листинг модулей».

Базы данных

База данных, приведенная к третьей нормальной форме, представляет из себя три каталога.

Первый каталог (файл bd1.pickle) состоит из простого первичного ключа «Идентификатор» и атрибутов «Бренд», «Модель», «Год», «Месяц», «День», «Розничная цена, $», «Основной цвет». Второй каталог (файл bd2.pickle) состоит из простого первичного ключа «Идентификатор» и атрибутов «Средняя цена перепродажи в 2016, $», «Средняя цена перепродажи в 2017, $», «Средняя цена перепродажи в 2018, $», «Средняя цена перепродажи в 2019, $», «Средняя цена перепродажи в 2020, $».

Листинг модулей

```
import tkinter as tk
from tkinter.ttk import Combobox
from tkinter.ttk import Treeview
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import (FigureCanvasTkAgg,
                                               NavigationToolbar2Tk)


def open_config():
    """
    Функция для чтения конфигурационного файла
    Принимает: ничего
    Возвращает: ничего
    Автор: Данилов Евгений Владимирович
    """
    backgroundcolour1 = tk.StringVar()
    backgroundcolour1.set('')
    backgroundcolour2 = tk.StringVar()
    backgroundcolour2.set('')
    buttoncolour = tk.StringVar()
    buttoncolour.set('')
    buttontextcolour = tk.StringVar()
    buttontextcolour.set('')
    buttoncolourpushed = tk.StringVar()
```

```
buttoncolourpushed.set('')
bd1path = tk.StringVar()
bd1path.set('')
bd2path = tk.StringVar()
bd2path.set('')
graphpath = tk.StringVar()
graphpath.set('')
tablepath = tk.StringVar()
tablepath.set('')
config = open('..//Scripts//configuration.txt')
global bgcolour1
global bgcolour2
global btncolour
global btntextcolour
global btncolourpushed
global bd1pth
global bd2pth
global graphpth
global tablepth

for line in config:
    line = line[:-1]
    if line[0] == '-':
        pass
    elif backgroundcolour1.get() == '':
        backgroundcolour1.set(line)
    elif backgroundcolour2.get() == '':
        backgroundcolour2.set(line)
    elif buttoncolour.get() == '':
        buttoncolour.set(line)
    elif buttontextcolour.get() == '':
        buttontextcolour.set(line)
```

```
        elif buttoncolourpushed.get() == '':
            buttoncolourpushed.set(line)
        elif bd1path.get() == '':
            bd1path.set(line)
        elif bd2path.get() == '':
            bd2path.set(line)
        elif graphpath.get() == '':
            graphpath.set(line)
        elif tablepath.get() == '':
            tablepath.set(line)
    bgcolour1 = backgroundcolour1.get()
    bgcolour2 = backgroundcolour2.get()
    btncolour = buttoncolour.get()
    btntextcolour = buttontextcolour.get()
    btncolourpushed = buttoncolourpushed.get()
    bd1pth = bd1path.get()
    bd2pth = bd2path.get()
    tablepth = tablepath.get()
    graphpth = graphpath.get()


def brand_avgretail(firstbd):
    """
    Функция, создающая DataFrame по фильтрам
    Принимает: firstbd
    Возвращает: avgretbr
    Автор: Гизатуллин Петр Олегович
    """
    general = pd.DataFrame(firstbd)
    general['Retail'] = general['Retail'].apply(pd.to_numeric, errors='coerce')
    avgretbr = general.groupby(['Brand']).agg({'Retail': "mean"})
    avgretbr.rename(columns={'Brand': 'Brand', 'Retail': 'Avg Retail'},
```

```
                inplace=True)
    avgretbr = avgretbr.reset_index()
    return avgretbr


def model_avgretail(firstbd):
    """
    Функция, создающая DataFrame по фильтрам
    Принимает: firstbd
    Возвращает: avgretmod
    Автор: Гизатуллин Петр Олегович
    """
    general = pd.DataFrame(firstbd)
    general['Retail'] = general['Retail'].apply(pd.to_numeric, errors='coerce')
    avgretmod = general.groupby(['Model']).agg({'Retail': "mean"})
    avgretmod.rename(columns={'Model': 'Model', 'Retail': 'Avg Retail'},
                 inplace=True)
    avgretmod = avgretmod.reset_index()
    return avgretmod


def year_avgretail(firstbd):
    """
    Функция, создающая DataFrame по фильтрам
    Принимает: firstbd
    Возвращает: avgretmod
    Автор: Гизатуллин Петр Олегович
    """
    general = pd.DataFrame(firstbd)
    general['Retail'] = general['Retail'].apply(pd.to_numeric, errors='coerce')
    avgretyear = general.groupby(['Year']).agg({'Retail': "mean"})
    avgretyear.rename(columns={'Year': 'Year', 'Retail': 'Avg Retail'},
```

```
                inplace=True)
    avgretyear = avgretyear.reset_index()
    return avgretyear


class ready(tk.Toplevel):
    """
    Конструктор класса окна с сообщением
    Автор: Данилов Евгений Владимирович
    """

    def __init__(self):
        super().__init__(root)
        self.title('Уведомление')
        self.geometry('300x100+635+300')
        self.resizable(False, False)
        self['bg'] = bgcolour1
        self.grab_set()
        self.focus_get()
        self['bg'] = bgcolour1
        tk.Label(self, bg=bgcolour1,
                 text='Готово!').place(x=0, y=0, width=300, height=50)
        btn_exitroot = tk.Button(self, text='Ок', bg=btncolour,
                                 fg=btntextcolour, width=8,
                                 activebackground=btncolourpushed,
                                 command=self.exitself)
        btn_exitroot.pack(side=tk.BOTTOM, pady=15)

    def exitself(self):
        self.destroy()
```

```
class savec(tk.Toplevel):
    """
    Конструктор класса окна с сообщением
    Автор: Данилов Евгений Владимирович
    """

    def __init__(self):
        super().__init__(root)
        self.title('Сообщение')
        self.geometry('300x130+635+300')
        self.resizable(False, False)
        self['bg'] = bgcolour1
        self.grab_set()
        self.focus_get()
        tk.Label(self.mess, bg=bgcolour1,
                 text='Для того, чтобы в сохраненной таблице\n'
                      'отсутствовали удаленные строки,\n'
                      'после их удаления необходимо\n'
                      'повторно нажать кнопку "Вывести".\n'
                      'Сохранить таблицу?').place(x=0, y=0, width=300,
                                                  height=80)
        btn_yes = tk.Button(self, text='Да', bg=btncolour, fg=btntextcolour,
                            width=8, activebackground=btncolourpushed,
                            command=self.save)
        btn_yes.place(relx=0.23, rely=0.68)
        btn_no = tk.Button(self, text='Нет', bg=btncolour, fg=btntextcolour,
                           width=8, activebackground=btncolourpushed,
                           command=self.exitself)
        btn_no.place(relx=0.56, rely=0.68)

    def exitself(self):
        self.destroy()
```

```
class text(tk.Toplevel):
    """
    Конструктор класса окна с сообщением
    Автор: Данилов Евгений Владимирович
    """

    def __init__(self, str1, str2):
        super().__init__(root)
        self.title(str1)
        self.geometry('300x100+635+300')
        self.resizable(False, False)
        self['bg'] = bgcolour1
        self.grab_set()
        self.focus_get()
        tk.Label(self, bg=bgcolour1,
                 text=str2).place(x=0, y=0, width=300, height=50)
        btn_exitroot = tk.Button(self, text='Ок', bg=btncolour,
                                 fg=btntextcolour, width=8,
                                 activebackground=btncolourpushed,
                                 command=self.exitself)
        btn_exitroot.pack(side=tk.BOTTOM, pady=15)

    def exitself(self):
        self.destroy()


class mistake(tk.Toplevel):
    """
    Конструктор класса окна с сообщением
    Автор: Данилов Евгений Владимирович
```

```
    """

    def __init__(self):
        super().__init__(root)
        self.title('Ошибка')
        self.geometry('300x100+635+300')
        self.resizable(False, False)
        self['bg'] = bgcolour1
        self.grab_set()
        self.focus_get()
        tk.Label(self, bg=bgcolour1,
                 text='Проверьте правильность и полноту\n'
                      'выбраннных/введенных данных\n'
                      'и попробуйте еще раз.').place(x=0, y=0, width=300,
                                                     height=50)
        btn_exitroot = tk.Button(self, text='Ок', bg=btncolour,
                                 fg=btntextcolour, width=8,
                                 activebackground=btncolourpushed,
                                 command=self.exitself)
        btn_exitroot.pack(side=tk.BOTTOM, pady=15)

    def exitself(self):
        self.destroy()


class Main(tk.Frame):
    """
    Конструктор класса стартового окна
    Автор: Данилов Евгений Владимирович
    """

    def __init__(self, root):
```

```
    super().__init__(root)
    self.init_main()

def init_main(self):
    root_frame = tk.Frame(root)
    root_frame['bg'] = bgcolour1
    root_frame.pack(expand=1)
    btn_open_add_data = tk.Button(root_frame,
                                  text='Добавление данных', font="20",
                                  command=self.open_add_data, bg=btncolour,
                                  fg=btntextcolour,
                                  activebackground=btncolourpushed,
                                  compound=tk.TOP, width=20, height=3)
    btn_open_add_data.pack(expand=1, pady=20)
    btn_open_change_data = tk.Button(root_frame, text='Изменение данных',
                                     font="20",
                                     command=self.open_change_data,
                                     bg=btncolour, fg=btntextcolour,
                                     activebackground=btncolourpushed,
                                     compound=tk.TOP, width=20, height=3)
    btn_open_change_data.pack(expand=1, pady=20)
    btn_open_statistics = tk.Button(root_frame,
                                    text='Графики', font="20",
                                    command=self.open_statistics,
                                    bg=btncolour, fg=btntextcolour,
                                    activebackground=btncolourpushed,
                                    compound=tk.TOP, width=20, height=3)
    btn_open_statistics.pack(expand=1, pady=20)
    btn_exitroot = tk.Button(root, text='Выход', bg=btncolour,
                             fg=btntextcolour, width=8,
                             activebackground=btncolourpushed,
                             command=self.exitroot)
```

```
        btn_exitroot.pack(pady=20)

    def open_add_data(self):
        add_data()

    def open_change_data(self):
        change_data()

    def open_statistics(self):
        statistics()

    def exitroot(self):
        root.destroy()


class add_data(tk.Toplevel):
    """
    Конструктор класса окна для добавления данных
    Автор: Данилов Евгений Владимирович
    """

    def __init__(self):
        super().__init__(root)
        self.title('Добавление данных')
        self.geometry('650x500+435+100')
        self.resizable(False, False)
        self.grab_set()
        self.focus_get()
        self['bg'] = bgcolour1
        tk.Label(self, bg=bgcolour2).place(relx=0.05, rely=0.05,
                                           relwidth=0.9, relheight=0.83)
        tk.Label(self, text='Введите данные:', font=20,
```

```
        bg=bgcolour2).place(relx=0.05, rely=0.1, relwidth=0.9)

self.lbl_1 = tk.Label(self, text='Бренд:', bg=bgcolour2)
self.lbl_1.place(x=70, y=100)
self.entry_lbl1 = tk.Entry(self, width=20)
self.entry_lbl1.place(x=115, y=100)
self.lbl_2 = tk.Label(self, text='Модель:', bg=bgcolour2)
self.lbl_2.place(x=265, y=100)
self.entry_lbl2 = tk.Entry(self, width=40)
self.entry_lbl2.place(x=320, y=100)
self.lbl_3 = tk.Label(self, text='Год:', bg=bgcolour2)
self.lbl_3.place(x=70, y=130)
self.entry_lbl3 = tk.Entry(self, width=7)
self.entry_lbl3.place(x=100, y=130)
self.lbl_4 = tk.Label(self, text='Месяц:', bg=bgcolour2)
self.lbl_4.place(x=150, y=130)
self.combo1 = Combobox(self, width=10, state='readonly')
self.combo1.place(x=200, y=130)
self.combo1['values'] = ('Январь', 'Февраль', 'Март', 'Апрель', 'Май',
                         'Июнь', 'Июль', 'Август', 'Сентябрь',
                         'Октябрь', 'Ноябрь', 'Декабрь')
self.combo1.current(0)
self.lbl_5 = tk.Label(self, text='День:', bg=bgcolour2)
self.lbl_5.place(x=290, y=130)
self.combo2 = Combobox(self, width=3, state='readonly')
self.combo2.place(x=330, y=130)
self.combo2['values'] = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
                         15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25,
                         26, 27, 28, 29, 30, 31)
self.combo2.current(0)
self.lbl_7 = tk.Label(self, text='Идентификатор:', bg=bgcolour2)
self.lbl_7.place(x=380, y=130)
```

```
self.entry_lbl7 = tk.Entry(self, width=14)
self.entry_lbl7.place(x=477, y=130)
self.lbl_8 = tk.Label(self, text='Розничная цена, $:', bg=bgcolour2)
self.lbl_8.place(x=70, y=160)
self.entry_lbl8 = tk.Entry(self, width=5)
self.entry_lbl8.place(x=180, y=160)
self.lbl_9 = tk.Label(self, text='Основной цвет:', bg=bgcolour2)
self.lbl_9.place(x=220, y=160)
self.entry_lbl9 = tk.Entry(self, width=15)
self.entry_lbl9.place(x=315, y=160)
self.lbl_10 = tk.Label(self,
                       text='Средняя цена перепродажи в 2016 году, $:',
                       bg=bgcolour2)
self.lbl_10.place(x=70, y=190)
self.entry_lbl10 = tk.Entry(self, width=5)
self.entry_lbl10.place(x=310, y=190)
self.lbl_11 = tk.Label(self,
                       text='Средняя цена перепродажи в 2017 году, $:',
                       bg=bgcolour2)
self.lbl_11.place(x=70, y=220)
self.entry_lbl11 = tk.Entry(self, width=5)
self.entry_lbl11.place(x=310, y=220)
self.lbl_12 = tk.Label(self,
                       text='Средняя цена перепродажи в 2018 году, $:',
                       bg=bgcolour2)
self.lbl_12.place(x=70, y=250)
self.entry_lbl12 = tk.Entry(self, width=5)
self.entry_lbl12.place(x=310, y=250)
self.lbl_13 = tk.Label(self,
                       text='Средняя цена перепродажи в 2019 году, $:',
                       bg=bgcolour2)
self.lbl_13.place(x=70, y=280)
```

```
    self.entry_lbl13 = tk.Entry(self, width=5)
    self.entry_lbl13.place(x=310, y=280)
    self.lbl_14 = tk.Label(self,
                           text='Средняя цена перепродажи в 2020 году, $:',
                           bg=bgcolour2)
    self.lbl_14.place(x=70, y=310)
    self.entry_lbl14 = tk.Entry(self, width=5)
    self.entry_lbl14.place(x=310, y=310)
    self.btn2 = tk.Button(self,
                          text='Добавить', command=self.add2, bg=btncolour,
                          fg=btntextcolour,
                          activebackground=btncolourpushed,
                          compound=tk.TOP, width=10, height=2)
    self.btn2.place(x=284, y=360)
    self.btn_exitroot = tk.Button(self, text='Выход', bg=btncolour,
                                  fg=btntextcolour, width=8,
                                  activebackground=btncolourpushed,
                                  command=self.exitself)
    self.btn_exitroot.place(x=292, y=452)


def add2(self):
    """
    Функция для добавления введенных данных в базы данных
    Принимает: ничего
    Возвращает: ничего
    Автор: Данилов Евгений Владимирович
    """
    f = 0
    self.bd1 = pd.read_pickle(bd1pth)
    ind = []
    for i in range(0, self.bd1.shape[0]):
        ind.append(i)
```

```
                self.bd1.index = list(ind)
                for i in range(len(self.bd1['Id'])):
                    if self.bd1['Id'][i] == self.entry_lbl7.get():
                        f += 1
                if f > 0:
                    text('Ошибка', 'Позиция с таким идентификатором уже есть в базе.')
                elif self.entry_lbl1.get() == '' or self.entry_lbl2.get() == '' or \
                        self.entry_lbl3.get() == '' or self.combo1.get() == '' or self.combo2.get() == '' or
\
                        self.entry_lbl7.get() == '' or self.entry_lbl8.get() == '' or self.entry_lbl9.get() ==
'':
                    mistake()


                elif (
                        self.entry_lbl10.get() == '' and self.entry_lbl11.get() == '' and
self.entry_lbl12.get() == '' and self.entry_lbl13.get() == '' and self.entry_lbl14.get() == ''):
                    mistake()
                elif (self.entry_lbl10.get() != '' and not (
                self.entry_lbl10.get()).isdigit()) or (self.entry_lbl11.get() != '' and
                                                       not (
                                                       self.entry_lbl11.get()).isdigit()) or (
                        self.entry_lbl12.get() != '' and not (
                self.entry_lbl12.get()).isdigit()) \
                        or (self.entry_lbl13.get() != '' and not (
                self.entry_lbl13.get()).isdigit()) or (self.entry_lbl14.get() != '' and not
                (self.entry_lbl14.get()).isdigit()) or (self.entry_lbl3.get() != '' and not (
                self.entry_lbl3.get()).isdigit()) or (self.entry_lbl8.get() != '' and not (
                self.entry_lbl8.get()).isdigit()):
                    mistake()
                elif int(self.entry_lbl3.get()) == 2017 and (
                        self.entry_lbl11.get() == '' or self.entry_lbl12.get() == '' or self.entry_lbl13.get()
== '' or self.entry_lbl14.get() == ''):
                    mistake()
```

```
            elif int(self.entry_lbl3.get()) == 2018 and (
                self.entry_lbl12.get() == '' or self.entry_lbl13.get() == '' or self.entry_lbl14.get()
== ''):
                mistake()
            elif int(self.entry_lbl3.get()) == 2019 and (
                self.entry_lbl13.get() == '' or self.entry_lbl14.get() == ''):
                mistake()
            elif int(self.entry_lbl3.get()) == 2020 and (self.entry_lbl14.get() == ''):
                mistake()
            elif self.entry_lbl1.get() == '' or self.entry_lbl2.get() == '' or self.entry_lbl3.get() ==
'' or self.combo1.get() == '' or self.combo2.get() == '' or self.entry_lbl7.get() == '' or
self.entry_lbl8.get() == '' or self.entry_lbl9.get() == '':
                mistake()


            else:
                pd.read_pickle(bd1pth).append(
                    {'Brand': self.entry_lbl1.get(), 'Model': self.entry_lbl2.get(),
                     'Year': self.entry_lbl3.get(),
                     'Month': self.combo1.get(), 'Day': self.combo2.get(),
                     'Id': self.entry_lbl7.get(),
                     'Retail': self.entry_lbl8.get(), 'Colour': self.entry_lbl9.get()},
                    ignore_index=True).to_pickle(
                    bd1pth)
                pd.read_pickle(bd2pth).append({'Id': self.entry_lbl7.get(),
                                               'Medium resale price in 2016': self.entry_lbl10.get(),
                                               'Medium resale price in 2017': self.entry_lbl11.get(),
                                               'Medium resale price in 2018': self.entry_lbl12.get(),
                                               'Medium resale price in 2019': self.entry_lbl13.get(),
                                               'Medium resale price in 2020': self.entry_lbl14.get()},
                                              ignore_index=True).to_pickle(bd2pth)
                ready()
                self.entry_lbl1.delete(0, tk.END)
                self.entry_lbl2.delete(0, tk.END)
```

```
            self.entry_lbl3.delete(0, tk.END)
            self.combo1.current(0)
            self.combo2.current(0)
            self.entry_lbl7.delete(0, tk.END)
            self.entry_lbl8.delete(0, tk.END)
            self.entry_lbl9.delete(0, tk.END)
            self.entry_lbl10.delete(0, tk.END)
            self.entry_lbl11.delete(0, tk.END)
            self.entry_lbl12.delete(0, tk.END)
            self.entry_lbl13.delete(0, tk.END)
            self.entry_lbl14.delete(0, tk.END)

    def exitself(self):
        self.destroy()


class change_data(tk.Toplevel):
    """
    Конструктор класса окна для просмотра и изменения данных
    Автор: Данилов Евгений Владимирович
    """

    def __init__(self):
        super().__init__(root)
        self.title('Изменение данных')
        self.geometry('650x500+435+100')
        self.resizable(False, False)
        self['bg'] = bgcolour1
        self.grab_set()
        self.focus_get()
        tk.Label(self, bg=bgcolour2).place(relx=0.05, rely=0.05,
                                           relwidth=0.9, relheight=0.83)
```

```
    self.lbl_2 = tk.Label(self, text='База данных №:', bg=bgcolour2)
    self.lbl_2.place(x=70, y=50)
    self.combo2 = Combobox(self, width=3, state='readonly')
    self.combo2.bind("<<ComboboxSelected>>", self.refresh)
    self.combo2.place(x=165, y=50)
    self.combo2['values'] = (1, 2)
    self.combo2.current(0)
    self.lbl_1 = tk.Label(self, text='Фильтр:', bg=bgcolour2)
    self.lbl_1.place(x=210, y=50)
    self.combo1 = Combobox(self, width=34, state='readonly')
    self.combo1.place(x=260, y=50)
    self.combo1['values'] = (
    'Без фильтра', 'Бренд', 'Модель', 'Год', 'Месяц', 'День', 'Идентификатор',
    'Розничная цена, $', 'Основной цвет')
    self.combo1.current(0)
    self.entry_lbl2 = tk.Entry(self, width=68, bg=bgcolour1)
    self.entry_lbl2.place(x=73, y=80)
    self.btn1 = tk.Button(self,
                          text='Вывести', command=self.output, bg=btncolour,
                          fg=btntextcolour,
                          activebackground=btncolourpushed, width=10)
    self.btn1.place(x=495, y=60)
    self.frame = tk.Frame(self, bg=bgcolour1)
    self.frame.place(relx=0.069, rely=0.22, relwidth=0.8615, relheight=0.57)

    self.btn_exitroot = tk.Button(self, text='Выход', bg=btncolour,
                                  fg=btntextcolour, width=8,
                                  activebackground=btncolourpushed,
                                  command=self.exitself)
    self.btn_exitroot.place(x=292, y=452)

def refresh(self, event):
```

```
        """
        Функция для обновления списка фильтров в зависимосми от выбранной базы
данных
        Принимает: ничего
        Возвращает: ничего
        Автор: Данилов Евгений Владимирович
        """
        if self.combo2.get() == '2':
            self.combo1['values'] = (
                'Без фильтра', 'Идентификатор', 'Средняя цена перепродажи в 2016, $',
                'Средняя цена перепродажи в 2017, $',
                'Средняя цена перепродажи в 2018, $',
                'Средняя цена перепродажи в 2019, $',
                'Средняя цена перепродажи в 2020, $')
            self.combo1.current(0)
        if self.combo2.get() == '1':
            self.combo1['values'] = (
            'Без фильтра', 'Бренд', 'Модель', 'Год', 'Месяц', 'День',
            'Идентификатор', 'Розничная цена, $', 'Основной цвет')
            self.combo1.current(0)

    def output(self):
        """
        Функция для вывода таблицы на экран
        Принимает: ничего
        Возвращает: ничего
        Автор: Данилов Евгений Владимирович
        """
        self.btn_save = tk.Button(self, text='Сохранить', bg=btncolour,
                                  fg=btntextcolour, width=8,
                                  activebackground=btncolourpushed,
                                  command=self.savec)
```

```
self.btn_save.place(x=240, y=405)
self.btn_del = tk.Button(self, text='Удалить', bg=btncolour,
                         fg=btntextcolour, width=8,
                         activebackground=btncolourpushed,
                         command=self.delete)
self.btn_del.place(x=340, y=405)
for widget in self.frame.winfo_children():
    widget.destroy()
if self.combo2.get() == '1':
    self.tree = Treeview(self.frame, column=(
        'Бренд', 'Модель', 'Год', 'Месяц', 'День', 'Идентификатор',
        'Розничная цена, $', 'Основной цвет'),
                         height=13, show='headings')
    self.columns = ['Бренд', 'Модель', 'Год', 'Месяц', 'День',
                    'Идентификатор', 'Розничная цена, $', 'Основной цвет']
    self.tree.column('Бренд', width=70,
                     anchor=tk.CENTER)
    self.tree.column('Модель', width=310,
                     anchor=tk.CENTER)
    self.tree.column('Год', width=70,
                     anchor=tk.CENTER)
    self.tree.column('Месяц', width=70,
                     anchor=tk.CENTER)
    self.tree.column('День', width=70,
                     anchor=tk.CENTER)
    self.tree.column('Идентификатор', width=120,
                     anchor=tk.CENTER)
    self.tree.column('Розничная цена, $', width=110,
                     anchor=tk.CENTER)
    self.tree.column('Основной цвет', width=130,
                     anchor=tk.CENTER)
    self.tree.heading('Бренд', text='Бренд')
```

```
                self.tree.heading('Модель', text='Модель')
                self.tree.heading('Год', text='Год')
                self.tree.heading('Месяц', text='Месяц')
                self.tree.heading('День', text='День')
                self.tree.heading('Идентификатор', text='Идентификатор')
                self.tree.heading('Розничная цена, $', text='Розничная цена, $')
                self.tree.heading('Основной цвет', text='Основной цвет')

                self.bd_main = pd.read_pickle(bd1pth)
                ind = []
                for i in range(0, self.bd_main.shape[0]):
                    ind.append(i)
                self.bd_main.index = list(ind)
                self.full_list = []
                for i in range(0, self.bd_main.shape[0]):
                    self.list_row = []
                    if self.combo1.get() == "Без фильтра":
                        if self.entry_lbl2.get() != '':
                            text('Сообщение',
                                 'В режиме "Без фильтра" невозможен\nпоиск по ключевым
словам')
                            self.entry_lbl2.delete(0, tk.END)
                        for j in self.bd_main.iloc[i]:
                            self.list_row.append(j)
                        self.full_list.append(self.list_row)
                    if self.combo1.get() == "Бренд":
                        if str(self.bd_main.iloc[i][
                                   'Brand']).upper() == self.entry_lbl2.get().upper():
                            for j in self.bd_main.iloc[i]:
                                self.list_row.append(j)
                            self.full_list.append(self.list_row)
                    if self.combo1.get() == "Модель":
```

```
    if str(self.bd_main.iloc[i][
            'Model']).upper() == self.entry_lbl2.get().upper():
        for j in self.bd_main.iloc[i]:
            self.list_row.append(j)
        self.full_list.append(self.list_row)
if self.combo1.get() == "Год":
    if str(self.bd_main.iloc[i][
            'Year']).upper() == self.entry_lbl2.get().upper():
        for j in self.bd_main.iloc[i]:
            self.list_row.append(j)
        self.full_list.append(self.list_row)
if self.combo1.get() == "Месяц":
    if str(self.bd_main.iloc[i][
            'Month']).upper() == self.entry_lbl2.get().upper():
        for j in self.bd_main.iloc[i]:
            self.list_row.append(j)
        self.full_list.append(self.list_row)
if self.combo1.get() == "День":
    if str(self.bd_main.iloc[i][
            'Day']).upper() == self.entry_lbl2.get().upper():
        for j in self.bd_main.iloc[i]:
            self.list_row.append(j)
        self.full_list.append(self.list_row)
if self.combo1.get() == "Идентификатор":
    if str(self.bd_main.iloc[i][
            'Id']).upper() == self.entry_lbl2.get().upper():
        for j in self.bd_main.iloc[i]:
            self.list_row.append(j)
        self.full_list.append(self.list_row)
if self.combo1.get() == "Розничная цена, $":
    if str(self.bd_main.iloc[i][
            'Retail']).upper() == self.entry_lbl2.get().upper():
```

```
                for j in self.bd_main.iloc[i]:
                    self.list_row.append(j)
                self.full_list.append(self.list_row)
            if self.combo1.get() == "Основной цвет":
                if str(self.bd_main.iloc[i][
                       'Colour']).upper() == self.entry_lbl2.get().upper():
                    for j in self.bd_main.iloc[i]:
                        self.list_row.append(j)
                    self.full_list.append(self.list_row)

        if self.full_list == []:
            text('Сообщение',
                 'В таблице отсутствуют данные,\nудовлетворяющие запросу.')
        for row in self.full_list:
            self.tree.insert('', tk.END, values=row)

    if self.combo2.get() == '2':
        self.tree = Treeview(self.frame, column=(
            'Идентификатор', 'Средняя цена перепродажи в 2016, $',
            'Средняя цена перепродажи в 2017, $',
            'Средняя цена перепродажи в 2018, $',
            'Средняя цена перепродажи в 2019, $',
            'Средняя цена перепродажи в 2020, $'),
                             height=13, show='headings')
        self.columns = ['Идентификатор', 'Средняя цена перепродажи в 2016, $',
                        'Средняя цена перепродажи в 2017, $',
                        'Средняя цена перепродажи в 2018, $',
                        'Средняя цена перепродажи в 2019, $',
                        'Средняя цена перепродажи в 2020, $']
        self.tree.column('Идентификатор', width=120,
                         anchor=tk.CENTER)
        self.tree.column('Средняя цена перепродажи в 2016, $', width=210,
```

```
                anchor=tk.CENTER)
self.tree.column('Средняя цена перепродажи в 2017, $', width=210,
                anchor=tk.CENTER)
self.tree.column('Средняя цена перепродажи в 2018, $', width=210,
                anchor=tk.CENTER)
self.tree.column('Средняя цена перепродажи в 2019, $', width=210,
                anchor=tk.CENTER)
self.tree.column('Средняя цена перепродажи в 2020, $', width=240,
                anchor=tk.CENTER)
self.tree.heading('Идентификатор', text='Идентификатор')
self.tree.heading('Средняя цена перепродажи в 2016, $',
                  text='Средняя цена перепродажи в 2016, $')
self.tree.heading('Средняя цена перепродажи в 2017, $',
                  text='Средняя цена перепродажи в 2017, $')
self.tree.heading('Средняя цена перепродажи в 2018, $',
                  text='Средняя цена перепродажи в 2018, $')
self.tree.heading('Средняя цена перепродажи в 2019, $',
                  text='Средняя цена перепродажи в 2019, $')
self.tree.heading('Средняя цена перепродажи в 2020, $',
                  text='Средняя цена перепродажи в 2020, $')

self.bd_main = pd.read_pickle(bd2pth)
ind = []
for i in range(0, self.bd_main.shape[0]):
    ind.append(i)
self.bd_main.index = list(ind)
self.full_list = []
for i in range(0, self.bd_main.shape[0]):
    self.list_row = []
    if self.combo1.get() == "Без фильтра":
        if self.entry_lbl2.get() != '':
            text('Сообщение',
```

```
                    'В режиме "Без фильтра" невозможен\nпоиск по ключевым
словам')
                self.entry_lbl2.delete(0, tk.END)
            for j in self.bd_main.iloc[i]:
                self.list_row.append(j)
            self.full_list.append(self.list_row)
        if self.combo1.get() == "Идентификатор":
            if str(self.bd_main.iloc[i][
                    'Id']).upper() == self.entry_lbl2.get().upper():
                for j in self.bd_main.iloc[i]:
                    self.list_row.append(j)
                self.full_list.append(self.list_row)
        if self.combo1.get() == "Средняя цена перепродажи в 2016, $":
            if str(self.bd_main.iloc[i][
                    'Medium resale price in 2016']).upper() ==
self.entry_lbl2.get().upper():
                for j in self.bd_main.iloc[i]:
                    self.list_row.append(j)
                self.full_list.append(self.list_row)
        if self.combo1.get() == "Средняя цена перепродажи в 2017, $":
            if str(self.bd_main.iloc[i][
                    'Medium resale price in 2017']).upper() ==
self.entry_lbl2.get().upper():
                for j in self.bd_main.iloc[i]:
                    self.list_row.append(j)
                self.full_list.append(self.list_row)
        if self.combo1.get() == "Средняя цена перепродажи в 2018, $":
            if str(self.bd_main.iloc[i][
                    'Medium resale price in 2018']).upper() ==
self.entry_lbl2.get().upper():
                for j in self.bd_main.iloc[i]:
                    self.list_row.append(j)
                self.full_list.append(self.list_row)
```

```
                    if self.combo1.get() == "Средняя цена перепродажи в 2019, $":
                        if str(self.bd_main.iloc[i][
                                'Medium resale price in 2019']).upper() ==
self.entry_lbl2.get().upper():
                            for j in self.bd_main.iloc[i]:
                                self.list_row.append(j)
                            self.full_list.append(self.list_row)
                    if self.combo1.get() == "Средняя цена перепродажи в 2020, $":
                        if str(self.bd_main.iloc[i][
                                'Medium resale price in 2020']).upper() ==
self.entry_lbl2.get().upper():
                            for j in self.bd_main.iloc[i]:
                                self.list_row.append(j)
                            self.full_list.append(self.list_row)

                if self.full_list == []:
                    text('Сообщение',
                         'В таблице отсутствуют данные,\nудовлетворяющие запросу.')
                for row in self.full_list:
                    self.tree.insert('', tk.END, values=row)

            self.scroll1 = tk.Scrollbar(self.frame, orient=tk.VERTICAL,
                                        command=self.tree.yview)
            self.tree.config(yscrollcommand=self.scroll1.set)
            self.scroll1.pack(side=tk.RIGHT, fill=tk.Y)

            self.scroll2 = tk.Scrollbar(self.frame, orient=tk.HORIZONTAL,
                                        command=self.tree.xview)
            self.tree.config(xscrollcommand=self.scroll2.set)
            self.scroll2.pack(side=tk.BOTTOM, fill=tk.X)
            self.tree.place(relwidth=1, relheight=1)
```

```
def delete(self):
    """
    Функция для удаления строки из таблицы и обеих баз данных
    Принимает: ничего
    Возвращает: ничего
    втор: Данилов Евгений Владимирович
    """
    if self.tree.selection() == ():
        text('Ошибка', 'Выберите строчку для удаления.')
    else:
        self.treeid = self.tree.selection()[0]
        self.bd1 = pd.read_pickle(bd1pth)
        ind = []
        for i in range(0, self.bd1.shape[0]):
            ind.append(i)
        self.bd1.index = list(ind)
        self.bd2 = pd.read_pickle(bd2pth)
        ind = []
        for i in range(0, self.bd2.shape[0]):
            ind.append(i)
        self.bd2.index = list(ind)

        if 'Бренд' in self.columns:
            for i in range(len(self.bd1['Id'])):
                if self.bd1['Id'][i] == \
                        self.tree.item(self.treeid, option='values')[5]:
                    self.bd1 = self.bd1.drop(i)
            self.bd1.to_pickle(bd1pth)

            for j in range(len(self.bd2['Id'])):
                if self.bd2['Id'][j] == \
                        self.tree.item(self.treeid, option='values')[5]:
```

```
                self.bd2 = self.bd2.drop(j)
            self.bd2.to_pickle(bd2pth)
            self.tree.delete(self.tree.selection()[0])
            self.tree.config(height=len(self.tree.get_children()))

        if 'Средняя цена перепродажи в 2016, $' in self.columns:
            for i in range(len(self.bd2['Id'])):
                if self.bd2['Id'][i] == \
                        self.tree.item(self.treeid, option='values')[0]:
                    self.bd2 = self.bd2.drop(i)
            self.bd2.to_pickle(bd2pth)

            for j in range(len(self.bd1['Id'])):
                if self.bd1['Id'][j] == \
                        self.tree.item(self.treeid, option='values')[0]:
                    self.bd1 = self.bd1.drop(j)
            self.bd1.to_pickle(bd1pth)
            self.tree.delete(self.tree.selection()[0])
            self.tree.config(height=len(self.tree.get_children()))

def savec(self):
    """
    Функция для создания окна с предупреждением перед сохранением таблицы
    Принимает: ничего
    Возвращает: ничего
    Автор: Данилов Евгений Владимирович
    """
    self.mess = tk.Toplevel()
    self.mess.title('Сообщение')
    self.mess.geometry('300x130+635+300')
    self.mess.resizable(False, False)
    self.mess['bg'] = bgcolour1
```

```
        self.mess.grab_set()
        self.mess.focus_get()
        tk.Label(self.mess, bg=bgcolour1,
                 text='Для того, чтобы в сохраненной таблице\n'
                      'отсутствовали удаленные строки,\n'
                      'после их удаления необходимо\n'
                      'повторно нажать кнопку "Вывести".\n'
                      'Сохранить таблицу?').place(x=0, y=0, width=300,
                                                  height=80)
        btn_yes = tk.Button(self.mess, text='Да', bg=btncolour, fg=btntextcolour,
                            width=8,
                            activebackground=btncolourpushed, command=self.save)
        btn_yes.place(relx=0.23, rely=0.68)
        btn_no = tk.Button(self.mess, text='Нет', bg=btncolour, fg=btntextcolour,
                           width=8,
                           activebackground=btncolourpushed,
                           command=self.mess.destroy)
        btn_no.place(relx=0.56, rely=0.68)

    def save(self):
        """
        Функция для сохранения выведенной на экран таблицы
        Принимает: ничего
        Возвращает: ничего
        Автор: Данилов Евгений Владимирович
        """
        filename = 'База данных ' + self.combo2.get() + ' Фильтр ' + self.combo1.get() +
'.csv'
        df_search = pd.DataFrame(self.full_list, columns=self.columns)

        file = tablepth + filename
```

```
        df_search.to_csv(file, index=None, header=True,
                         encoding="windows-1251", sep=';')
        text("Сообщение", "Таблица успешно сохранена в файл!")
        self.mess.destroy()

    def exitself(self):
        self.destroy()


class statistics(tk.Toplevel):
    """
    Конструктор класса окна для просмотра графиков
    Авторы: Данилов Евгений Владимирович, Гизатуллин Петр Олегович
    """

    def __init__(self):
        super().__init__(root)
        self.title('Статистика и графики')
        self.geometry('650x500+435+100')
        self.resizable(False, False)
        self.grab_set()
        self.focus_get()
        self['bg'] = bgcolour1
        tk.Label(self, bg=bgcolour2).place(relx=0.05, rely=0.05,
                                           relwidth=0.9, relheight=0.83)
        lbl_1 = tk.Label(self, text='Фильтр:', bg=bgcolour2)
        lbl_1.place(x=70, y=40)
        self.combo1 = Combobox(self, width=30, state='readonly')
        self.combo1.place(x=125, y=40)
        self.combo1['values'] = (
        'Бренд - Средняя цена', 'Модель - Средняя цена', 'Год - Средняя цена',)
        self.combo1.current(0)
```

```
    btn_graf = tk.Button(self, text='Построить график', bg=btncolour,
                         fg=btntextcolour,
                         activebackground=btncolourpushed,
                         command=self.graph_draw)
    btn_graf.place(x=340, y=40)
    btn_exitroot = tk.Button(self, text='Выход', bg=btncolour, fg=btntextcolour,
                             width=8,
                             activebackground=btncolourpushed,
                             command=self.exitself)
    btn_exitroot.place(x=550, y=452)


def graph_draw(self):
    """
    Функция для построения и вывода графиков
    Принимает: ничего
    Возвращает: ничего
    Автор: Гизатуллин Петр Олегович
    """
    self.help = tk.Frame(self, bg=bgcolour1)
    self.help.place(relx=0.1, rely=0.15, relwidth=0.8, relheight=1)
    self.graph = tk.Frame(self.help, bg=bgcolour1)
    self.graph.place(relx=0, rely=-0.08, relwidth=1, relheight=0.9)
    if self.combo1.get() == 'Бренд - Средняя цена':
        fbd = pd.read_pickle(bd1pth)
        bd = brand_avgretail(fbd)
        fig = plt.figure(figsize=(4, 5), dpi=70)
        ax = fig.add_subplot(1, 1, 1)
        fig.suptitle('')
        bd.plot(kind='bar', ax=ax, x='Brand', y="Avg Retail", rot=45, fontsize=9)

    if self.combo1.get() == 'Модель - Средняя цена':
        fbd = pd.read_pickle(bd1pth)
```

```
        bd = model_avgretail(fbd)
        fig = plt.figure(figsize=(4, 5), dpi=70)
        ax = fig.add_subplot(1, 1, 1)
        fig.suptitle('')
        bd.plot(kind='bar', ax=ax, x='Model', y="Avg Retail", rot=45, fontsize=9)

    if self.combo1.get() == 'Год - Средняя цена':
        fbd = pd.read_pickle(bd1pth)
        bd = year_avgretail(fbd)
        fig = plt.figure(figsize=(4, 5), dpi=70)
        ax = fig.add_subplot(1, 1, 1)
        fig.suptitle('')
        bd.plot(kind='bar', ax=ax, x='Year', y="Avg Retail", rot=45, fontsize=9)

    CANVAS_1 = FigureCanvasTkAgg(fig, master=self.graph)
    CANVAS_1.draw()
    CANVAS_1.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

    toolbar = NavigationToolbar2Tk(CANVAS_1, self.graph)
    toolbar.update()
    CANVAS_1.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)
    btn_exitroot = tk.Button(self, text='Выход', bg=btncolour, fg=btntextcolour,
                             width=8,
                             activebackground=btncolourpushed,
                             command=self.exitself)
    btn_exitroot.place(x=550, y=452)

def exitself(self):
    self.destroy()

def exitself(self):
    self.destroy()
```

```
if __name__ == '__main__':
    root = tk.Tk()
    open_config()
    app = Main(root)
    app.pack()
    root.title('SneakerAnalysis')
    root.geometry('650x500+435+100')
    root['bg'] = bgcolour1
    root.resizable(False, False)
    root.mainloop()
```