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

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

Руководство разработчика по работе с приложением: «Футбольный агент».

Разработчики:

Данилов Игорь Евгеньевич

Петросян Гурген Аликович

Хомин Максим Вячеславович

Руководитель:

Полякова М. В.

Москва 2024 г.

Оглавление

Требования к характеристикам компьютера и операционной системе

Наличие на компьютере интерпретатора «Python» (вне зависимости от среды разработки)

Операционная система – Windows 7, 8, 10.

Версии интерпретатора и используемых библиотек

Интерпретатор – Python 3.7+.

Используемые библиотеки:

Библиотека	Версия
contourpy	1.2.1+
cycler	0.12.1+
fonttools	4.53.0+
kiwisolver	1.4.5++
matplotlib	3.9.0+
numpy	1.26.4+
packaging	24.1+
pandas	2.2.2+
pillow	10.3.0+
pyparsing	3.1.2+
python-dateuti	2.9.0.post0+
pytz	2024.1+
six	1.16.0+
tzdata	2024.1+

Инструкция по установке приложения

Загрузка zip-архива. Создание виртуального окружения “python3 -m venv venv”,

активация виртуального окружения “source venv/bin/activate”, “pip install requirements.txt”

Инструкция по запуску и настройке приложения

Для запуска приложения требуется выполнить команду “python3 main.py” в каталоге Work/scripts. Дополнительных настроек не требуется.

Описание структур БД

Используется две базы данных: база данных со статистикой игроков и база данных со статистикой футбольных клубов.

База данных

Структура каталогов

Первый уровень	Назначение
Work	Основной каталог, содержащий программные модули модулями

Архитектура приложения

data	work	Папка, хранящая базы данных
graphics	work	Папка, хранящая графические отчеты
output	work	Папка, хранящая текстовые отчеты
scripts	work	Папка, хранящая скрипты
Модуль	**Расположение**	**Функция**
main.py	work\scripts	запуска программы и работы с базой данных
script_graphics.py	work\scripts	Построение графических отчетов
script_text.py	work\scripts	Построение текстовых отчетов

Листинг

Main.py

```
from tkinter import simpledialog, StringVar, OptionMenu, filedialog, messagebox
from script_graphics import *
from script_text import *
from tkinter import ttk

# Чтение данных
player_data = pd.read_csv(r'../data\player_premier_league_stats.csv', delimiter=";").to_numpy()
teams_data = pd.read_csv(r'../data/squad_premier_league_stats.csv', delimiter=',').to_numpy()

df = pd.read_csv('../data/player_premier_league_stats.csv', delimiter=";")
df_teams = pd.read_csv('../data/squad_premier_league_stats.csv', delimiter=',')

teams = [i for i in df_teams["Squad"]]
POSITIONS = ["FW", "MF", "DF"]

# Создание директорий, если не существует
output_dir = "../graphics/"
if not os.path.exists(output_dir):
    os.makedirs(output_dir)
output_dir = "../output/"
if not os.path.exists(output_dir):
    os.makedirs(output_dir)


def select_teams_and_plot(plot_function):
    """ Функция выбора двух команд для генерации заданного графика
    Автор: Хомин Максим

    Parameters
    ----------
    plot_function - график, который необходимо сгенерировать
    Returns
    -------
    None
    """

    def on_submit():
        team_1 = team_var_1.get()
        team_2 = team_var_2.get()
        positions = ['DF', 'MF', 'FW']
```

```
        if plot_function == plot_clustered_bar_chart:
            plot_function(player_data, team_1, team_2, positions)
        else:
            plot_function(player_data, team_1, team_2)
        select_window.destroy()

    select_window = tk.Toplevel(root)
    select_window.title("Выбор команд")

    team_var_1 = StringVar(select_window)
    team_var_2 = StringVar(select_window)

    team_var_1.set(teams[0])
    team_var_2.set(teams[1])

    tk.Label(select_window, text="Команда 1:").pack(pady=5)
    team_menu_1 = OptionMenu(select_window, team_var_1, *teams)
    team_menu_1.pack(pady=5)

    tk.Label(select_window, text="Команда 2:").pack(pady=5)
    team_menu_2 = OptionMenu(select_window, team_var_2, *teams)
    team_menu_2.pack(pady=5)

    submit_button = tk.Button(select_window, text="Подтвердить", command=on_submit)
    submit_button.pack(pady=10)


def select_teams_and_plot_five(plot_function):
    """ Функция выбора пяти команд для генерации заданного графика
    Автор: Петросян Гурген

    Parameters
    ----------
    plot_function - график, который необходимо сгенерировать
    Returns
    -------
    None
    """

    def on_submit():
        team_1 = team_var_1.get()
        team_2 = team_var_2.get()
        team_3 = team_var_3.get()
        team_4 = team_var_4.get()
        team_5 = team_var_5.get()
        teams_list = [team_1, team_2, team_3, team_4, team_5]
        if plot_function == plot_histogram:
            plot_histogram(teams_data, teams_list)
        else:
```

```
        plot_boxplot(player_data, teams_list)
    select_window.destroy()

select_window = tk.Toplevel(root)
select_window.title("Выбор команд")

team_var_1 = StringVar(select_window)
team_var_2 = StringVar(select_window)
team_var_3 = StringVar(select_window)
team_var_4 = StringVar(select_window)
team_var_5 = StringVar(select_window)

team_var_1.set(teams[0])
team_var_2.set(teams[1])
team_var_3.set(teams[2])
team_var_4.set(teams[3])
team_var_5.set(teams[4])

tk.Label(select_window, text="Команда 1:").pack(pady=5)
team_menu_1 = OptionMenu(select_window, team_var_1, *teams)
team_menu_1.pack(pady=5)

tk.Label(select_window, text="Команда 2:").pack(pady=5)
team_menu_2 = OptionMenu(select_window, team_var_2, *teams)
team_menu_2.pack(pady=5)

tk.Label(select_window, text="Команда 3:").pack(pady=5)
team_menu_2 = OptionMenu(select_window, team_var_3, *teams)
team_menu_2.pack(pady=5)

tk.Label(select_window, text="Команда 4:").pack(pady=5)
team_menu_2 = OptionMenu(select_window, team_var_4, *teams)
team_menu_2.pack(pady=5)

tk.Label(select_window, text="Команда 5:").pack(pady=5)
team_menu_2 = OptionMenu(select_window, team_var_5, *teams)
team_menu_2.pack(pady=5)

submit_button = tk.Button(select_window, text="Подтвердить", command=on_submit)
submit_button.pack(pady=10)


# Текстовый отчет №1
def select_position_and_age():
    """ Функция выбора позиции игрока и ввода максимального возраста для генерации голов
    Автор: Данилов Игорь

    Returns
    -------
```

```
    None
    """

    def on_submit():
        pos = str(position_var.get())
        max_age = int(age_entry.get())
        goals(df, 'Pos', pos, 'Age', max_age)
        select_window.destroy()

    select_window = tk.Toplevel()
    select_window.title("Выбор позиции и возраста")

    positions = ['DF', 'MF', 'FW']

    position_var = StringVar(select_window)
    position_var.set(positions[0])

    tk.Label(select_window, text="Выберите позицию:").pack(pady=5)
    position_menu = OptionMenu(select_window, position_var, *positions)
    position_menu.pack(pady=5)

    tk.Label(select_window, text="Введите максимальный возраст игрока:").pack(pady=5)
    age_entry = tk.Entry(select_window)
    age_entry.pack(pady=5)

    submit_button = tk.Button(select_window, text="Подтвердить", command=on_submit)
    submit_button.pack(pady=10)

    select_window.mainloop()


# Текстовый отчет №3
def select_team_and_run_xAG():
    """ Функция выбора команды и вызова функции xAG с выбранными параметрами
    Автор: Петросян Гурген

    Returns
    -------
    None
    """

    def on_submit():
        squad = squad_var.get()
        xAG(df, 'Squad', squad)
        select_window.destroy()

    select_window = tk.Toplevel()
    select_window.title("Выбор команды для xAG")
```

```
    squads = df['Squad'].unique()

    squad_var = StringVar(select_window)
    squad_var.set(squads[0])

    tk.Label(select_window, text="Выберите команду:").pack(pady=5)
    squad_menu = OptionMenu(select_window, squad_var, *squads)
    squad_menu.pack(pady=5)

    submit_button = tk.Button(select_window, text="Подтвердить", command=on_submit)
    submit_button.pack(pady=10)

    select_window.mainloop()


# Текстовый отчет №4
def select_aggregation_column_and_create_pivot_table():
    """ Функция выбора столбца для агрегации и вызова функции create_pivot_table с
выбранными параметрами
    Автор: Хомин Максим

    Returns
    -------
    None
    """

    def on_submit():
        variable = variable_var.get()
        create_pivot_table(df, 'Squad', 'Pos', variable, 'sum')
        select_window.destroy()

    select_window = tk.Toplevel()
    select_window.title("Выбор столбца для агрегации")

    variables = ['Goals', 'Assist', '90s_played', 'xG', 'xAG']

    variable_var = StringVar(select_window)
    variable_var.set(variables[0])

    tk.Label(select_window, text="Выберите столбец для агрегации:").pack(pady=5)
    variable_menu = OptionMenu(select_window, variable_var, *variables)
    variable_menu.pack(pady=5)

    submit_button = tk.Button(select_window, text="Подтвердить", command=on_submit)
    submit_button.pack(pady=10)

    select_window.mainloop()
```

```
# Функции для работы с базой данных команд
def add_object_to_database_squad():
    """ Функция для добавления объекта в базу данных
    Автор: Хомин Максим

    Returns
    -------
    None
    """
    # Запрос информации об объекте у пользователя с помощью интерфейса
    df = pd.read_csv('../data/squad_premier_league_stats.csv', delimiter=",")
    new_object_data = {}
    root = tk.Tk()
    root.withdraw()  # Скрыть основное окно

    for column in df.columns:
        new_value = simpledialog.askstring("Введите значение", f"Введите значение для столбца
'{column}': ")
        new_object_data[column] = new_value

    # Добавление нового объекта в базу данных
    df.loc[len(df)] = new_object_data

    # Сохранение изменений обратно в файл
    df.to_csv('../data/squad_premier_league_stats.csv', index=False, sep=",")


def delete_object_from_database_squad():
    """ Функция для удаления объекта из базы данных
    Автор: Хомин Максим

    Returns
    -------
    None
    """
    # Запрос информации о команде у пользователя с помощью интерфейса
    df = pd.read_csv('../data/squad_premier_league_stats.csv', delimiter=",")
    team_name = simpledialog.askstring("Введите название команды", "Введите название
команды для удаления: ")

    # Удаление строки игрока из базы данных
    df = df[df['Squad'] != team_name]

    # Сохранение изменений обратно в файл
    df.to_csv('../data/squad_premier_league_stats.csv', index=False, sep=",")


def edit_value_in_database_squad():
    """ Функция для изменения объекта в базе данных
```

```
    Автор: Хомин Максим

    Returns
    -------
    None
    """
    # Запрос информации о команде у пользователя с помощью интерфейса
    df = pd.read_csv('../data/squad_premier_league_stats.csv', delimiter=",")
    team_name = simpledialog.askstring("Введите название команды", "Введите название
команды для изменения значения: ")

    # Запрос столбца и нового значения у пользователя
    column_name = simpledialog.askstring("Введите название столбца",
                                         "Введите название столбца для изменения значения: ")
    new_value = simpledialog.askstring("Введите новое значение",
                                       f"Введите новое значение для столбца '{column_name}': ")

    # Изменение значения для указанного игрока и столбца
    mask = df['Squad'] == team_name
    df.loc[mask, column_name] = new_value

    # Сохранение изменений обратно в файл
    df.to_csv('../data/squad_premier_league_stats.csv', index=False, sep=",")


def save_data_to_bin_squad():
    """ Функция для сохранения базы данных в двоичный формат
    Автор: Петросян Гурген

    Returns
    -------
    None
    """
    root = tk.Tk()
    root.withdraw()  # Скрыть основное окно
    file_path = filedialog.asksaveasfilename(defaultextension=".pkl", filetypes=[("Pickle files",
"*.pkl")])
    if file_path:
        try:
            df_teams.to_pickle(file_path)  # Сохранение базы данных в выбранный двоичный файл
            messagebox.showinfo("Успех", "База данных успешно сохранена в двоичном формате.")
        except Exception as e:
            messagebox.showerror("Ошибка", f"Ошибка при сохранении базы данных в двоичном
формате: {e}")


def read_data_from_bin_squad():
    """ Функция для чтения базы данных из двоичного формата
    Автор: Данилов Игорь
```

```
    Returns
    -------
    None
    """
    root = tk.Tk()
    root.withdraw()  # Скрыть основное окно
    file_path = filedialog.askopenfilename(filetypes=[("Pickle files", "*.pkl")])
    if file_path:
        try:
            df_teams = pd.read_pickle(file_path)  # Чтение базы данных из выбранного двоичного файла
            df_teams.to_csv('../data/squad_premier_league_stats.csv', index=False, sep=",")  # Сохранение данных в формате CSV
            messagebox.showinfo("Успех",
                                "База данных успешно восстановлена из двоичного файла и сохранена в формате CSV.")
        except Exception as e:
            messagebox.showerror("Ошибка", f"Ошибка при восстановлении базы данных из двоичного файла: {e}")

# Функции для работы с базой данных игроков
def add_object_to_database():
    """ Функция для добавления объекта в базу данных
    Автор: Хомин Максим

    Returns
    -------
    None
    """
    # Запрос информации об объекте у пользователя с помощью интерфейса
    df = pd.read_csv('../data/player_premier_league_stats.csv', delimiter=";")
    new_object_data = {}
    root = tk.Tk()
    root.withdraw()  # Скрыть основное окно

    for column in df.columns:
        new_value = simpledialog.askstring("Введите значение", f"Введите значение для столбца '{column}': ")
        new_object_data[column] = new_value

    # Добавление нового объекта в базу данных
    df.loc[len(df)] = new_object_data

    # Сохранение изменений обратно в файл
    df.to_csv('../data/player_premier_league_stats.csv', index=False, sep=";")


def delete_object_from_database():
```

```
    """ Функция для удаления объекта из базы данных
    Автор: Хомин Максим

    Returns
    -------
    None
    """
    # Запрос информации об игроке у пользователя с помощью интерфейса
    df = pd.read_csv('../data/player_premier_league_stats.csv', delimiter=";")
    player_name = simpledialog.askstring("Введите имя игрока", "Введите имя игрока для
удаления: ")

    # Удаление строки игрока из базы данных
    df = df[df['Player'] != player_name]

    # Сохранение изменений обратно в файл
    df.to_csv('../data/player_premier_league_stats.csv', index=False, sep=";")


def edit_value_in_database():
    """ Функция для изменения объекта в базе данных
    Автор: Хомин Максим

    Returns
    -------
    None
    """
    # Запрос информации об игроке у пользователя с помощью интерфейса
    df = pd.read_csv('../data/player_premier_league_stats.csv', delimiter=";")
    player_name = simpledialog.askstring("Введите имя игрока", "Введите имя игрока для
изменения значения: ")

    # Запрос столбца и нового значения у пользователя
    column_name = simpledialog.askstring("Введите название столбца",
                                         "Введите название столбца для изменения значения: ")
    new_value = simpledialog.askstring("Введите новое значение",
                                       f"Введите новое значение для столбца '{column_name}': ")

    # Изменение значения для указанного игрока и столбца
    mask = df['Player'] == player_name
    df.loc[mask, column_name] = new_value

    # Сохранение изменений обратно в файл
    df.to_csv('../data/player_premier_league_stats.csv', index=False, sep=";")


def save_data_to_bin():
    """ Функция для сохранения базы данных в двоичный формат
    Автор: Петросян Гурген
```

```
    Returns
    -------
    None
    """
    root = tk.Tk()
    root.withdraw()  # Скрыть основное окно
    file_path = filedialog.asksaveasfilename(defaultextension=".pkl", filetypes=[("Pickle files",
"*.pkl")])
    if file_path:
        try:
            df.to_pickle(file_path)  # Сохранение базы данных в выбранный двоичный файл
            messagebox.showinfo("Успех", "База данных успешно сохранена в двоичном формате.")
        except Exception as e:
            messagebox.showerror("Ошибка", f"Ошибка при сохранении базы данных в двоичном
формате: {e}")


def read_data_from_bin():
    """ Функция для чтения базы данных из двоичного формата
    Автор: Данилов Игорь

    Returns
    -------
    None
    """
    root = tk.Tk()
    root.withdraw()  # Скрыть основное окно
    file_path = filedialog.askopenfilename(filetypes=[("Pickle files", "*.pkl")])
    if file_path:
        try:
            df = pd.read_pickle(file_path)  # Чтение базы данных из выбранного двоичного файла
            df.to_csv('../data/player_premier_league_stats.csv', index=False, sep=";")  # Сохранение
данных в формате CSV
            messagebox.showinfo("Успех",
                                "База данных успешно восстановлена из двоичного файла и сохранена в
формате CSV.")
        except Exception as e:
            messagebox.showerror("Ошибка", f"Ошибка при восстановлении базы данных из
двоичного файла: {e}")

# Функции для вызова диалогов и запуска соответствующих функций
def call_plot_clustered_bar_chart():
    select_teams_and_plot(plot_clustered_bar_chart)


def call_plot_histogram():
    select_teams_and_plot_five(plot_histogram)
```

```
def call_plot_boxplot():
    select_teams_and_plot_five(plot_boxplot)


def call_plot_scatter():
    select_teams_and_plot(plot_scatter)


def call_edit_database():
    """ Функция вызова редактирования базы данных игроков
    Автор: Данилов Игорь

    Returns
    -------
    None
    """
    edit_window = tk.Toplevel()
    edit_window.title("Управление базой данных")

    btn_add = tk.Button(edit_window, text="Добавить объект",
command=add_object_to_database)
    btn_add.pack(pady=10)

    btn_delete = tk.Button(edit_window, text="Удалить объект",
command=delete_object_from_database)
    btn_delete.pack(pady=10)

    btn_edit = tk.Button(edit_window, text="Редактировать объект",
command=edit_value_in_database)
    btn_edit.pack(pady=10)

    btn_save = tk.Button(edit_window, text="Сохранить справочник в двоичном формате",
command=save_data_to_bin)
    btn_save.pack(pady=10)

    btn_load = tk.Button(edit_window, text="Считать справочник из двоичного формата",
command=read_data_from_bin)
    btn_load.pack(pady=10)

    edit_window.mainloop()

def call_edit_database_squad():
    """ Функция вызова редактирования базы данных команд
    Автор: Данилов Игорь

    Returns
    -------
    None
```

```
    """
    edit_window = tk.Toplevel()
    edit_window.title("Управление базой данных команд")

    btn_add = tk.Button(edit_window, text="Добавить объект",
command=add_object_to_database_squad)
    btn_add.pack(pady=10)

    btn_delete = tk.Button(edit_window, text="Удалить объект",
command=delete_object_from_database_squad)
    btn_delete.pack(pady=10)

    btn_edit = tk.Button(edit_window, text="Редактировать объект",
command=edit_value_in_database_squad)
    btn_edit.pack(pady=10)

    btn_save = tk.Button(edit_window, text="Сохранить справочник в двоичном формате",
command=save_data_to_bin_squad)
    btn_save.pack(pady=10)

    btn_load = tk.Button(edit_window, text="Считать справочник из двоичного формата",
command=read_data_from_bin_squad)
    btn_load.pack(pady=10)

    edit_window.mainloop()


# Создание окна и кнопок
root = tk.Tk()
root.title("Футбольный агент")


# Кнопки для редактирования баз данных
btn_edit = tk.Button(root, text="Взаимодействие с базой данных игроков",
command=call_edit_database)
btn_edit.pack(pady=10)

btn_edit = tk.Button(root, text="Взаимодействие с базой данных команд",
command=call_edit_database_squad)
btn_edit.pack(pady=10)

# Кнопки для генерации графиков
btn1 = tk.Button(root, text="Сравнение эффективности игроков разных позиций",
command=call_plot_clustered_bar_chart)
btn1.pack(pady=10)
tooltip = ttk.Label(root,
                    text="Функция создаёт график, сравнивающий количество голов по позициям для
двух выбранных команд")
tooltip.pack()
```

```
btn2 = tk.Button(root, text="Средний возраст игроков", command=call_plot_histogram)
btn2.pack(pady=10)
tooltip = ttk.Label(root,
                    text="Функция создаёт гистограмму по среднему возрасту игроков пяти выбранных
команд")
tooltip.pack()

btn3 = tk.Button(root, text="Статистика «Гол+пас»", command=call_plot_boxplot)
btn3.pack(pady=10)
tooltip = ttk.Label(root,
                    text="Функция создаёт категоризированную диаграмму по системе 'гол+пас' для
игроков пяти выбранных команд")
tooltip.pack()

btn4 = tk.Button(root, text="Категоризированная диаграмма рассеивания",
command=call_plot_scatter)
btn4.pack(pady=10)
tooltip = ttk.Label(root,
                    text="Функция создаёт категоризированную диаграмму рассеивания по голам и
помощам для пяти выбранных команд")
tooltip.pack()

# Кнопки для генерации текстовых отчетов
btn5 = tk.Button(root, text="Текстовый отчет №1", command=select_position_and_age)
btn5.pack(pady=10)
tooltip = ttk.Label(root,
                    text="Функция создаёт текстовый отчет о количестве голов для игроков выбранной
позиции младше выбранного возраста")
tooltip.pack()

btn6 = tk.Button(root, text="Текстовый отчет №2", command=statistics)
btn6.pack(pady=10)
tooltip = ttk.Label(root,
                    text="Функция создаёт текстовый статистический отчет для количественных
переменных базы данных")
tooltip.pack()

btn7 = tk.Button(root, text="Текстовый отчет №3", command=select_team_and_run_xAG)
btn7.pack(pady=10)
tooltip = ttk.Label(root,
                    text="Функция создаёт текстовый отчет, сравнивающий помощи игроков
выбранной команды с их xAG")
tooltip.pack()

btn8 = tk.Button(root, text="Сводная таблица",
command=select_aggregation_column_and_create_pivot_table)
btn8.pack(pady=10)
tooltip = ttk.Label(root,
```

```
            text="Функция создаёт сводную таблицу по позициям для каждой команды,
пользователь может выбрать столбец для агрегации")
tooltip.pack()

root.mainloop()
```

script_graphics.py

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import os

output_dir = "../graphics/"


def plot_clustered_bar_chart(player_data, TEAM_1, TEAM_2, POSITIONS):
    """ Функция генерации кластеризованной столбчатой диаграммы для двух команд,
    сравнивающая суммарное количество голов игроков по позициям среди этих команд.
    Автор: Данилов Игорь

    Parameters
    ----------
    player_data : pd.DataFrame - исходная таблица с данными об игроках
    TEAM_1 : str - название команды 1
    TEAM_2 : str - название команды 2
    POSITIONS : list - позиции, по которым идёт сравнение
    Returns
    -------
    None
    """
    goals_1, goals_2 = [0 for _ in range(len(POSITIONS))], [0 for _ in range(len(POSITIONS))]

    for i in range(len(player_data)):
        player = player_data[i]
        if player[3] == TEAM_1:
            for j in range(len(POSITIONS)):
                if POSITIONS[j] in player[2]:
                    goals_1[j] += int(player[7])
        if player[3] == TEAM_2:
            for j in range(len(POSITIONS)):
                if POSITIONS[j] in player[2]:
                    goals_2[j] += int(player[7])

    bar_width = 0.32
    x = range(len(POSITIONS))

    plt.bar(x, goals_1, width=bar_width, label=TEAM_1)
    plt.bar([i + bar_width for i in x], goals_2, width=bar_width, label=TEAM_2)

    plt.xlabel('Позиции игроков')
    plt.ylabel('Суммарное количество голов')
    plt.title('Кластеризованная столбчатая диаграмма')
```

```
    plt.xticks([i + bar_width / 2 for i in x], POSITIONS)
    plt.legend()

    plt.savefig(os.path.join(output_dir, "1.png"))
    plt.show()


def plot_histogram(teams_data, teams):
    """ Функция генерации категоризированной гистограммы, показывающей средний
    возраст игроков выбранных команд с добавлением значений на график.
    Автор: Хомин Максим

    Parameters
    ----------
    teams_data : pd.DataFrame - исходная таблица с данными о командах
    teams : list - список выбранных команд
    Returns
    -------
    None
    """
    average_ages = []
    for team in teams_data:
        if team[0] in teams:
            average_ages.append(team[1])

    plt.bar(teams, average_ages, color='skyblue')

    plt.title('Категоризированная гистограмма')
    plt.xlabel('Команды')
    plt.ylabel('Средний возраст')

    for i in range(len(teams)):
        plt.text(i, average_ages[i], str(average_ages[i]), ha='center', va='bottom')

    plt.savefig(os.path.join(output_dir, "2.png"))
    plt.show()


def plot_boxplot(player_data, teams):
    """ Функция генерации категоризированной диаграммы (box-and-whiskers)
    для параметра "гол+пас" игроков из выбранных команд
    Автор: Петросян Гурген

    Parameters
    ----------
    player_data : pd.DataFrame - исходная таблица с данными об игроках
    teams : list - список выбранных команд
    Returns
    -------
```

```
    None
    """
    team_indices = {team: idx for idx, team in enumerate(teams)}
    poss_by_mass = [[] for _ in range(len(teams))]

    for player in player_data:
        for team in teams:
            if player[3] == team:
                team_idx = team_indices[team]
                poss_by_mass[team_idx].append(player[8])
                break

    data = {team: ages for team, ages in zip(teams[:5], poss_by_mass[:5])}

    box_data = [ages for team, ages in data.items()]

    plt.boxplot(box_data)
    plt.xticks(range(1, len(data) + 1), data.keys())
    plt.xlabel("Команды")
    plt.ylabel("Гол+пас")
    plt.title("Категоризированная диаграмма")

    plt.savefig(os.path.join(output_dir, "3.png"))
    plt.show()


def plot_scatter(player_data, TEAM, TEAM_2):
    """ Функция генерации категоризированной диаграммы рассеивания для
    двух выбранных команд, показывающей количество помощей игроков
    Автор: Данилов Игорь

    Parameters
    ----------
    player_data : pd.DataFrame - исходная таблица с данными об игроках
    TEAM : str - название команды 1
    TEAM_2 : str - название команды 2
    Returns
    -------
    None
    """
    goals, assist, current_player = [], [], []
    goals_2, assist_2, current_player_2 = [], [], []

    for i in range(len(player_data)):
        if player_data[i, 3] == TEAM:
            current_player.append(player_data[i, 0])
            goals.append(player_data[i, 7])
            assist.append(player_data[i, 8])
        if player_data[i, 3] == TEAM_2:
```

```
        current_player_2.append(player_data[i, 0])
        goals_2.append(player_data[i, 7])
        assist_2.append(player_data[i, 8])

plt.scatter(goals, assist, label=TEAM)
plt.scatter(goals_2, assist_2, label=TEAM_2)

plt.xlabel("Количество голов")
plt.ylabel("Количество ассистов")
plt.title('Категоризированная диаграмма рассеивания')

plt.legend()
plt.savefig(os.path.join(output_dir, "4.png"))
plt.show()
```

script_text.py

```
import pandas as pd
import tkinter as tk

df = pd.read_csv('../data/player_premier_league_stats.csv', delimiter=";")
df_teams = pd.read_csv('../data/squad_premier_league_stats.csv', delimiter=",")


def goals(df: pd.DataFrame, cndname: str, cndval: str, intname: str, intval: int) -> pd.DataFrame:
    """ Функция генерации отчетов о количестве голов для одного строкового
    и одного целочисленного (макс.) критериев с выводом на экран
    Автор: Хомин Максим

    Parameters
    ----------
    df : pd.DataFrame - исходная таблица с данными об игроках
    cndname : str - атрибут критерия
    cndval : str - значение критерия
    intname : str - атрибут критерия
    intval : int - максимальное значение критерия
    Returns
    -------
    filtered_df : pd.DataFrame - отчет.
    """
    filtered_df = df[(df[cndname] == cndval) & (df[intname] <= intval)][['Player', intname, 'Squad',
'Goals']]
    filtered_text = filtered_df.to_string(index=False)

    display_window = tk.Toplevel()
    display_window.title("Результаты")

    text_widget = tk.Text(display_window, wrap="none")
    text_widget.insert('1.0', filtered_text)
    text_widget.pack(side="left", fill="both", expand=True)

    scrollbar_y = tk.Scrollbar(display_window, command=text_widget.yview)
    scrollbar_y.pack(side="right", fill="y")
    text_widget.config(yscrollcommand=scrollbar_y.set)

    scrollbar_x = tk.Scrollbar(display_window, orient="horizontal", command=text_widget.xview)
    scrollbar_x.pack(side="bottom", fill="x")
    text_widget.config(xscrollcommand=scrollbar_x.set)

    filtered_df.to_csv('../output/report1.txt', sep=';', index=False)
    return filtered_df
```

```python
def statistics() -> pd.DataFrame:
    """
    Функция для генерации отчетов статистики для основных количественных переменных
    Автор: Петросян Гурген

    Parameters
    ----------
    count : количество непустых (непропущенных) значений в столбце.
    mean : среднее арифметическое значение для всех значений в столбце.
    std : стандартное отклонение, которое измеряет разброс значений от среднего значения. Большое стандартное
    отклонение указывает на большой разброс, а маленькое - на то, что значения сгруппированы близко к среднему.
    min : минимальное значение в столбце.
    25% (первый квартиль) : значение, ниже которого попадает 25% значений столбца. Это также называется 25-м процентилем.
    50% (медиана) : значение, ниже которого попадает 50% значений столбца. Это также называется медианой.
    75% (третий квартиль) : значение, ниже которого попадает 75% значений столбца. Это также называется 75-м процентилем.
    max : максимальное значение в столбце.
    Returns
    -------
    statistics_df : pd.DataFrame - отчет с основными статистическими показателями
    """

    variables = ['Age', 'Goals', 'Match_Play', '90s_played', 'Assist', 'xG']

    statistics_df = df[variables].describe().T

    display_window = tk.Toplevel()
    display_window.title("Результаты")

    text_widget = tk.Text(display_window, wrap="none")
    text_widget.insert('1.0', statistics_df.to_string())
    text_widget.pack(side="left", fill="both", expand=True)

    scrollbar_y = tk.Scrollbar(display_window, command=text_widget.yview)
    scrollbar_y.pack(side="right", fill="y")
    text_widget.config(yscrollcommand=scrollbar_y.set)

    scrollbar_x = tk.Scrollbar(display_window, orient="horizontal", command=text_widget.xview)
    scrollbar_x.pack(side="bottom", fill="x")
    text_widget.config(xscrollcommand=scrollbar_x.set)

    statistics_df.to_csv('../output/report2.txt', sep=';', index=False)
    return statistics_df
```

```
def xAG(df: pd.DataFrame, cndname_1: str, cndval_1: str) -> pd.DataFrame:
    """ Функция генерации отчетов о сравнении реальных ассистов игрока с его xAG для
    одного строкового критерия
    Автор: Данилов Игорь

    Parameters
    ----------
    df : pd.DataFrame - исходная таблица с данными об игроках
    cndname_1 : str - атрибут критерия 1
    cndval_1 : str - значение критерия 1
    Returns
    -------
    filtered_df : pd.DataFrame - отчет.
    """
    filtered_df = df[df[cndname_1] == cndval_1][['Player', '90s_played', 'Assist', 'xAG']]
    filtered_df.to_csv('../output/report3.txt', sep=';', index=False)

    filtered_text = filtered_df.to_string(index=False)

    display_window = tk.Toplevel()
    display_window.title("Результаты")

    text_widget = tk.Text(display_window, wrap="none")
    text_widget.insert('1.0', filtered_text)
    text_widget.pack(side="left", fill="both", expand=True)

    scrollbar_y = tk.Scrollbar(display_window, command=text_widget.yview)
    scrollbar_y.pack(side="right", fill="y")
    text_widget.config(yscrollcommand=scrollbar_y.set)

    scrollbar_x = tk.Scrollbar(display_window, orient="horizontal", command=text_widget.xview)
    scrollbar_x.pack(side="bottom", fill="x")
    text_widget.config(xscrollcommand=scrollbar_x.set)
    return filtered_df

def create_pivot_table(df: pd.DataFrame, index_col1: str, index_col2: str, values_col: str,
                       agg_func: str) -> pd.DataFrame:
    """
    Функция для создания сводной таблицы на основе входных параметров
    Автор: Хомин Максим

    Parameters
    ----------
    df: pd.DataFrame - исходная таблица с данными
    index_col1: str - название первого качественного столбца для использования в качестве
индекса
    index_col2: str - название второго качественного столбца для использования в качестве
индекса
    values_col: str - название количественного столбца для агрегации
```

```
    agg_func: str - метод агрегации

    Returns
    -------
    pivot_table : pd.DataFrame - Сводная таблица на основе входных параметров
    """
    pivot_table = pd.pivot_table(df, values=values_col, index=index_col1, columns=index_col2,
aggfunc=agg_func,
                                 fill_value=0)

    display_window = tk.Toplevel()
    display_window.title("Результаты")

    text_widget = tk.Text(display_window, wrap="none")
    text_widget.insert('1.0', pivot_table.to_string())
    text_widget.pack(side="left", fill="both", expand=True)

    scrollbar_y = tk.Scrollbar(display_window, command=text_widget.yview)
    scrollbar_y.pack(side="right", fill="y")
    text_widget.config(yscrollcommand=scrollbar_y.set)

    scrollbar_x = tk.Scrollbar(display_window, orient="horizontal", command=text_widget.xview)
    scrollbar_x.pack(side="bottom", fill="x")
    text_widget.config(xscrollcommand=scrollbar_x.set)

    pivot_table.to_csv('../output/report4.txt', sep=';', index=True)
    return pivot_table
```