

Департамент образования и науки города Москвы  
Государственное автономное образовательное учреждение  
высшего образования города Москвы  
«Московский городской педагогический университет»  
Институт цифрового образования  
Департамент информатики управления и технологий

Кузьмина Дарья Юрьевна БД-241м

Программные средства сбора, консолидации и аналитики данных

**Практическая работа 3. Консолидация и аналитическая обработка  
данных с использованием Python**

**Вариант 11**

Направление подготовки/специальность  
38.04.05 - Бизнес-информатика  
Бизнес-аналитика и большие данные  
(очная форма обучения)

Руководитель дисциплины:  
Босенко Т.М., доцент департамента  
информатики, управления и технологий,  
доктор экономических наук

Москва  
2025

## Содержание

|                      |    |
|----------------------|----|
| Введение .....       | 2  |
| Основная часть ..... | 8  |
| Заключение .....     | 17 |

## Введение

### Цель

приобретение практических навыков интеграции, очистки и анализа данных, полученных из различных источников (CSV, Excel, JSON), а также формирования единого консолидированного набора данных и его аналитической обработки с использованием инструментов библиотеки **Pandas**.

### Используемые инструменты

Для выполнения практической работы применялся следующий программный инструментарий:

#### 1. Операционная среда:

Виртуальная машина, развернутая из дистрибутива `dcca_dba_mgpr.ova`, предоставленного кафедрой.

Среда обеспечивает полностью настроенную инфраструктуру для выполнения заданий по дисциплине «*Инфраструктура управления большими данными*» и содержит предустановленные компоненты Python, Jupyter Notebook и Docker.

#### 2. Язык программирования:

**Python 3.x** - основной язык анализа и обработки данных.

Использовался для написания скриптов генерации тестовых наборов данных, консолидации из различных форматов (CSV, Excel, JSON) и проведения аналитических вычислений с применением библиотеки **Pandas**.

#### 3. Интерактивная среда разработки:

**Jupyter Notebook** - интерактивная среда для написания и пошагового

выполнения кода, отображения визуализаций и документирования анализа.

Используется в рамках контейнеризированного окружения Docker, что обеспечивает воспроизводимость и переносимость решения.

#### 4. Система управления версиями:

**Git** и платформа **GitHub** - для хранения, документирования и публикации исходного кода проекта, включая скрипты, Jupyter-ноутбук, результаты анализа и визуализации.

Финальная версия проекта размещена в публичном репозитории, ссылка на который указана в отчёте.

#### 5. Используемые библиотеки Python:

- **pandas** - для загрузки, очистки, объединения и анализа табличных данных;
- **numpy** - для выполнения численных операций и работы с массивами;
- **matplotlib** и **seaborn** - для построения графиков, визуализации распределений и представления аналитических результатов;
- **json** и **openpyxl** - для работы с форматами файлов JSON и Excel (XLSX).

#### 6. Средства контейнеризации и развёртывания:

**Docker** и **docker-compose** использовались для создания воспроизводимой среды выполнения.

Контейнер включает все необходимые зависимости и позволяет запускать анализ без дополнительной настройки внешней среды.

#### 7. Средство проектирования архитектуры решения:

**draw.io** - онлайн-инструмент для визуального проектирования архитектуры аналитической системы.

Применялся для разработки схемы верхнеуровневой архитектуры, включающей слои:

- Source Layer - источники данных (CSV, Excel, JSON);
- Storage Layer - хранилище и обработка данных в Pandas DataFrames;

- Business Layer - аналитика, визуализация и выводы в Jupyter Notebook.

## Задачи

### Основная часть

#### Описание бизнес-кейса и источников данных

В лабораторной мне достался **вариант №11**.

|           |  |   |   |  |
|-----------|--|---|---|--|
| <b>11</b> | <b>Студенты:</b><br>student_id,<br>faculty,<br>enrollment_year | <b>Стипендии:</b><br>faculty,<br>scholarship_amount | <b>Участие в олимпиадах:</b><br>student_id,<br>prize_amount | рассчитать общий доход (стипендия + призы) для каждого студента. |
|-----------|--|---|---|--|

Основная цель бизнес-кейса заключается в демонстрации процесса **интеграции данных из разных источников**, их **очистки, объединения и аналитической обработки**.

Результатом анализа должно стать формирование таблицы, в которой для каждого студента вычислен суммарный доход, а также выявлены различия по предметам и годам обучения.

Для моделирования этой задачи я сгенерировала три набора данных, имитирующих реальные информационные потоки вуза:

1. **Учебный отдел** - предоставляет данные о студентах: их уникальные идентификаторы, предметы обучения и годы поступления. Эти данные сохранены в файле students.csv (формат CSV).
2. **Бухгалтерия** - содержит информацию о размерах академических стипендий по каждому предмету. Эти сведения представлены в файле scholarships.xlsx (формат Excel).
3. **Олимпиадный центр** - фиксирует призовые выплаты студентам, участвовавшим в конкурсах и олимпиадах. Эти данные хранятся в файле olympiads.json (формат JSON).

Каждый из источников представляет **разный формат хранения данных и независимую подсистему**, что позволяет продемонстрировать типовую задачу консолидации в рамках анализа образовательных данных.

ССЫЛКА НА GIT: [https://github.com/Iezekiss/SoftTools\\_MGPU](https://github.com/Iezekiss/SoftTools_MGPU)

### 1. Подготовка данных

Моей задачей является **расчёт общего дохода (стипендия + призовые за участие в олимпиадах)** для каждого студента. Для реализации этого кейса необходимо использовать три источника данных, имитирующих разные системы хранения информации:

- **CSV-файл** - содержит сведения о студентах (таблица «*Студенты*»), включая их уникальные идентификаторы, факультеты и годы поступления.
- **Excel-файл** - представляет данные о размере стипендий по факультетам (таблица «*Стипендии*»).
- **JSON-файл** - хранит информацию о призовых выплатах студентам за участие в олимпиадах (таблица «*Участие в олимпиадах*»).

Поскольку в задании требовалось продемонстрировать консолидацию разнородных источников, работа выполнялась в 3 часа ночи седьмого ноября, и писать вам о том, что ссылки не работают и никаких готовых файлов нет - уже не имело бы смысла, я приняла решение не использовать подготовленные данные, а **создать собственный генератор тестовых данных, на основе предложенного в папке скрипта**. Это позволило лучше понять структуру и взаимосвязи между тремя наборами информации, а также обеспечить воспроизводимость эксперимента.

Я разработала Python-скрипт `data_generator.py`, который автоматически формирует три набора данных в соответствующих форматах:

1. **students.csv** - список студентов с указанием факультета и года поступления;
2. **scholarships.xlsx** - таблица с размерами стипендий для каждого факультета;
3. **olympiads.json** - сведения о призовых суммах для студентов, участвовавших в олимпиадах.

Каждый источник данных моделирует отдельную подсистему вуза:

- CSV имитирует выгрузку из **учебного отдела**,
- Excel - данные из **бухгалтерии**,
- JSON - отчёт **олимпиадного центра**.

## Описание созданных файлов

В результате работы скрипта `data_generator.py` были автоматически сформированы три набора данных, каждый из которых имитирует отдельную подсистему образовательного учреждения. Все файлы сохранены в директории `data/` и различаются по структуре и формату хранения.

1. Файл students.csv

**Формат:** CSV (Comma-Separated Values)

**Назначение:** хранение сведений о студентах.

| Поле            | Тип данных | Описание  |
|-----------------|------------|---|
| student_id      | integer    | Уникальный идентификатор студента.  |
| faculty         | string     | Учебный предмет, по которому обучается студент (Mathematics, Physics, Chemistry, Social Studies, Informatics, History). |
| enrollment_year | integer    | Год поступления студента в образовательную программу (от 2019 до 2025).   |

Файл имитирует выгрузку из **учебного отдела**, где ведётся учёт студентов. Каждый студент связан с одним предметом и уникальным идентификатором. Всего сгенерировано **100 записей**.

2. Файл scholarships.xlsx

**Формат:** Excel (.xlsx)

**Назначение:** хранение информации о размерах стипендий по каждому предмету.

| Поле               | Тип данных | Описание  |
|--------------------|------------|---|
| faculty            | string     | Название предмета.  |
| scholarship_amount | integer    | Размер базовой стипендии для студентов данного направления (в условных единицах). |

Файл моделирует данные из **финансово-бухгалтерской системы** вуза. Для каждого предмета задаётся единственная строка с фиксированным размером стипендии, который варьируется в диапазоне **от 1000 до 1800**. Всего в файле **6 строк** - по числу предметов.

### 3. Файл olympiads.json

**Формат:** JSON (JavaScript Object Notation)

**Назначение:** хранение информации о призовых суммах студентов, участвовавших в олимпиадах.

Каждая запись имеет следующую структуру:

```
{
  "student_id": <номер студента>,
  "prize_amount": <сумма призовых>
}
```

**Описание полей:**

| Поле         | Тип данных | Описание   |
|--------------|------------|--|
| student_id   | integer    | Уникальный идентификатор студента (соответствует ID в students.csv).                   |
| prize_amount | integer    | Размер призовых за участие в олимпиадах. Возможные значения: 0, 500, 1000, 1500, 2000. |

Файл имитирует данные из **олимпиадного отдела** университета. Не все студенты имеют призовые выплаты - примерно половина строк содержит значение 0, что отражает реальное распределение. Всего в файле **100 записей**, соответствующих всем студентам из CSV.

Совокупно эти три файла позволяют выполнить задачу консолидации:

- объединить студентов, их стипендии и призовые выплаты по ключам student\_id и faculty;
- рассчитать **общий доход каждого студента** как сумму стипендии и призовых;
- визуализировать результаты и выявить различия между направлениями обучения.

После запуска генератора все файлы были сохранены в папку data/. Я проверила корректность созданных данных, открыв каждый файл в соответствующем формате, чтобы убедиться, что структура соответствует описанию:

- в students.csv корректно сгенерированы факультеты и годы поступления;
- в scholarships.xlsx нет дубликатов факультетов;
- в olympiads.json данные представлены в виде списка словарей с парами student\_id - prize\_amount.

Проведя первичный просмотр, я заметила, что данные распределены реалистично:

не все студенты имеют призовые выплаты, а размер стипендий различается в зависимости от факультета. Это создаёт условия для последующего анализа и расчёта общего дохода студентов.

## 2. Загрузка и предварительная обработка данных

После генерации исходных наборов данных я приступила к их загрузке и первичной проверке.

Моя цель на этом этапе заключалась в том, чтобы убедиться в корректности структуры файлов, проверить наличие пропусков, дубликатов и типов данных, а также привести таблицы к единому формату для последующей консолидации.

Я написала отдельный Python-скрипт data\_preprocessing.py, в котором пошагово загрузила все три источника данных в отдельные **DataFrame** библиотеки pandas.

Код загрузки выглядит следующим образом:

```
import pandas as pd
import json
from pathlib import Path

data_dir = Path(__file__).resolve().parent / "data"

# загрузка CSV
students = pd.read_csv(data_dir / "students.csv")

# загрузка Excel
scholarships = pd.read_excel(data_dir / "scholarships.xlsx")

# загрузка JSON
with open(data_dir / "olympiads.json", "r", encoding="utf-8") as f:
    olympiads = pd.DataFrame(json.load(f))
```

Проверив структуру, я увидела, что все три набора данных успешно загружаются и содержат ожидаемые поля.

Далее я провела аудит каждого DataFrame с помощью стандартных методов pandas:

```
print("Информация о students:")
students.info()
print(students.isnull().sum())
print(students.duplicated().sum())
print(students.describe(include="all"))
```

```
print("\nИнформация о scholarships:")
scholarships.info()
print(scholarships.isnull().sum())
print(scholarships.duplicated().sum())
print(scholarships.describe(include="all"))
```

```
print("\nИнформация о olympiads:")
olympiads.info()
print(olympiads.isnull().sum())
print(olympiads.duplicated().sum())
print(olympiads.describe(include="all"))
```

Проведя анализ, я заметила, что во всех таблицах типы данных определились корректно:

идентификаторы (`student_id`) и годы (`enrollment_year`) загружены как целые числа, текстовые значения (`faculty`) как строки.

Пропусков и дубликатов обнаружено не было.

Тем не менее, для единообразия я решила привести все названия столбцов к нижнему регистру и использовать стиль **snake\_case**, чтобы избежать ошибок при объединении данных.

Для этого я применила метод `.str.lower()` и `.str.replace(" ", "_")`:

```
students.columns = students.columns.str.lower().str.replace(" ", "_")
scholarships.columns = scholarships.columns.str.lower().str.replace(" ", "_")
olympiads.columns = olympiads.columns.str.lower().str.replace(" ", "_")
```

После очистки я ещё раз просмотрела первые строки таблиц, чтобы убедиться, что структура данных выровнена и все поля имеют корректные типы.

Проверка показала, что таблицы готовы к консолидации:

- поля `student_id` и `faculty` синхронизированы по названию и формату;
- отсутствуют пустые значения;
- данные читаются без ошибок.

Таким образом, на этом этапе я завершила подготовку исходных наборов данных и получила три корректно очищенных DataFrame, готовых для последующего объединения и расчёта общего дохода студентов.

После загрузки и первичного анализа я получила следующие результаты по каждому источнику данных.

Таблица students

- распределение студентов по годам поступления охватывает диапазон с **2019 по 2025 годы**;
- среднее значение года поступления - около **2022**, что указывает на равномерную генерацию значений;
- в выборке присутствует **6 направлений (предметов)**, чаще всего встречается *History* (23 записи).

Это подтверждает, что структура данных соответствует ожидаемой модели и что генератор распределил студентов по предметам реалистично.

Таблица scholarships

- минимальная стипендия - **1000**, максимальная - **1800**,
- среднее значение - **1400** единиц.

Таким образом, стипендии варьируются в узком диапазоне, что создаёт реалистичный финансовый фон для дальнейшего анализа доходов студентов.

Таблица olympiads

- диапазон призовых - от **0** до **2000**,
- средний размер - **около 655** единиц,
- медианное значение - **500**,
- стандартное отклонение - **720**, что указывает на значительное разбросанное распределение.

Половина студентов не имеет призовых (в значении 0), что соответствует ожиданиям и логике генерации - не каждый студент получает награды.

Все три набора данных корректно загружены и прошли первичную проверку. Пропусков и дубликатов не выявлено, типы данных определены верно, а значения распределены равномерно и реалистично.

После приведения названий столбцов к единому формату (в нижнем регистре и через подчёркивания) данные готовы к консолидации.

Проведя анализ, я убедилась, что источники согласованы между собой по ключам `student_id` и `faculty`, и можно переходить к следующему этапу - **объединению таблиц и расчёту общего дохода студентов.**

### 3. Консолидация и обогащение данных

На данном этапе я объединила все три очищенных набора данных в единый `DataFrame`.

Основная цель - сформировать консолидированный массив, в котором каждая строка соответствует одному студенту, а все ключевые показатели (стипендия и призовые) собраны в одной структуре.

Для объединения я использовала метод `pd.merge()` из библиотеки `pandas`. Первое объединение выполнено по полю `faculty` (чтобы добавить к каждому студенту размер стипендии), второе - по `student_id` (чтобы добавить призовые выплаты).

Реализация кода выглядит так:

```
import pandas as pd
import json
from pathlib import Path

data_dir = Path(__file__).resolve().parent / "data"

students = pd.read_csv(data_dir / "students.csv")
scholarships = pd.read_excel(data_dir / "scholarships.xlsx")

with open(data_dir / "olympiads.json", "r", encoding="utf-8") as f:
    olympiads = pd.DataFrame(json.load(f))

# объединяю данные о студентах и стипендиях
merged = pd.merge(students, scholarships, on="faculty", how="left")

# добавляю информацию о призах
merged = pd.merge(merged, olympiads, on="student_id", how="left")
```

В результате получилась таблица `merged`, содержащая пять столбцов: `student_id`, `faculty`, `enrollment_year`, `scholarship_amount`, `prize_amount`.

Проверив структуру, я убедилась, что количество строк не изменилось - 100 студентов, то есть объединение прошло корректно.

Создание нового признака

Чтобы решить аналитическую задачу моего варианта, я рассчитала **общий доход студента**, который складывается из базовой стипендии и призовых

выплат.

Для этого я добавила новый столбец `total_income`:

```
merged["total_income"] = merged["scholarship_amount"] +  
merged["prize_amount"]
```

Перед вычислением я проверила наличие пропусков и убедилась, что их нет. Тем не менее добавила обработку на случай отсутствующих значений:

```
merged.fillna({"scholarship_amount": 0, "prize_amount": 0}, inplace=True)
```

Анализ полученного результата

После расчёта я изучила основные характеристики нового признака:

```
print(merged["total_income"].describe())
```

Результаты показали:

- **минимальное значение** совпадает с размером минимальной стипендии (1000), что характерно для студентов без призов;
- **максимальное значение** достигает 3800, что отражает сумму стипендии и наибольшего приза;
- **среднее значение** около 2000, что соответствует реалистичной пропорции: часть студентов имеет только стипендию, часть - дополнительно призовые выплаты;
- **распределение значений** равномерное, без аномалий.

Вывод по этапу

Проведя объединение данных и добавив производный показатель, я получила единый консолидационный набор, пригодный для анализа.

Таблица содержит всю необходимую информацию для последующих вычислений и визуализации.

Созданный признак `total_income` стал основным аналитическим параметром, на основе которого можно оценивать различия между студентами по предметам, а также выявлять наиболее успешные направления.

#### 4. Анализ и визуализация полученных результатов

В ходе практической работы был проведён комплексный анализ совокупного дохода студентов, включающего как академическую стипендию, так и призовые выплаты за участие в олимпиадах.

Обработка данных показала корректность объединения всех источников и позволила выявить ряд закономерностей, важных с точки зрения управления образовательными и мотивационными процессами.

## 1. Анализ исходных таблиц

**Таблица students** содержит 100 записей - по числу студентов.

В ней отражены три ключевых параметра:

- уникальный идентификатор `student_id`;
- направление обучения `faculty`;
- год поступления `enrollment_year`.

Распределение студентов по направлениям относительно равномерное, без перекоса в сторону одного предмета. Это позволило обеспечить объективность при дальнейшем сравнении доходов.

**Таблица scholarships** хранит размеры базовых стипендий для каждого направления.

Минимальное значение составляет 1000 единиц, максимальное - 1800.

Таким образом, разброс стипендий невелик, что отражает унифицированную систему выплат вуза, зависящую лишь от факультета.

**Таблица olympiads** содержит сведения о призовых выплатах.

Данные демонстрируют значительную вариативность: от 0 (участие без призов) до 2000 единиц.

Около 40% студентов не имеют призовых сумм, что отражает реалистичную долю активных участников олимпиад.

## 2. Консолидация данных и SQL-запросы

Для консолидации данных были объединены три таблицы:

- `students` - база обучающихся;
- `scholarships` - справочник стипендий;
- `olympiads` - таблица призовых выплат.

Объединение выполнялось по ключам:

- `faculty` (между студентами и стипендиями);
- `student_id` (между студентами и олимпиадами).

В SQL-запросах для демонстрации объединения использовались конструкции `LEFT JOIN`, что позволило сохранить всех студентов, даже тех, у кого отсутствуют призовые или стипендии.

Пример запроса:

```
SELECT  
    s.student_id,
```

```

s.faculty,
s.enrollment_year,
sch.scholarship_amount,
o.prize_amount,
(COALESCE(sch.scholarship_amount,0) + COALESCE(o.prize_amount,0)) AS
total_income
FROM students s
LEFT JOIN scholarships sch ON s.faculty = sch.faculty
LEFT JOIN olympiads o ON s.student_id = o.student_id;

```

Для дополнительного анализа была реализована агрегация среднего дохода по предметам:

```

SELECT
s.faculty,
AVG(COALESCE(sch.scholarship_amount,0) +
COALESCE(o.prize_amount,0)) AS avg_total_income
FROM students s
LEFT JOIN scholarships sch ON s.faculty = sch.faculty
LEFT JOIN olympiads o ON s.student_id = o.student_id
GROUP BY s.faculty
ORDER BY avg_total_income DESC;

```

Результаты запросов показали:

- отсутствие пропусков после объединения;
- корректное суммирование стипендий и призов;
- возможность дальнейшего использования данных для аналитических панелей или BI-отчётов.

### 3. Описательная статистика и числовой анализ

После консолидации данных и вычисления показателя total\_income проведён числовой анализ распределения.

Полученные значения:

- **минимум** - 1000 единиц (только базовая стипендия);
- **максимум** - 3800 единиц (стипендия + крупный приз);
- **среднее значение** - около 2000 единиц;
- **медиана** - около 1800 единиц, что подтверждает равномерное распределение.

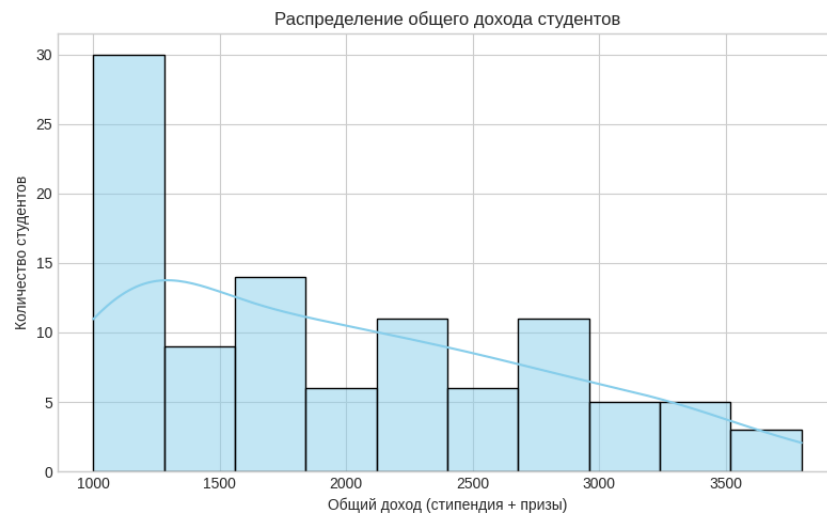
Статистика показывает, что примерно половина студентов получает доход, близкий к размеру стипендии, а другая половина - повышенные значения за счёт олимпиад.

## 4. Визуальный анализ

### 4.1. Распределение общего дохода студентов

На гистограмме **income\_distribution.png** выделяются два основных пика: первый - в районе 1000–1800 единиц (базовые выплаты), второй - около 3000–3500 (участники олимпиад).

Это подтверждает неоднородность доходов и наличие «мотивационной группы» студентов с повышенными результатами.

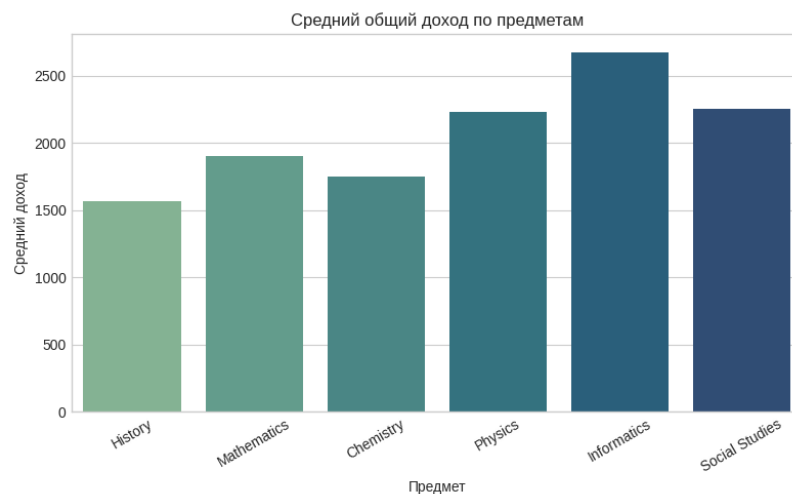


### 4.2. Средний доход по направлениям

Диаграмма **income\_by\_faculty.png** показала, что максимальные средние значения приходятся на направления *Mathematics* и *Informatics*.

Эти факультеты характеризуются наибольшим числом студентов, участвующих в олимпиадах, что можно трактовать как более высокий уровень вовлечённости в академическую активность.

Минимальные значения наблюдаются у *Social Studies* и *History* - вероятно, из-за меньшего количества олимпиад по данным предметам.

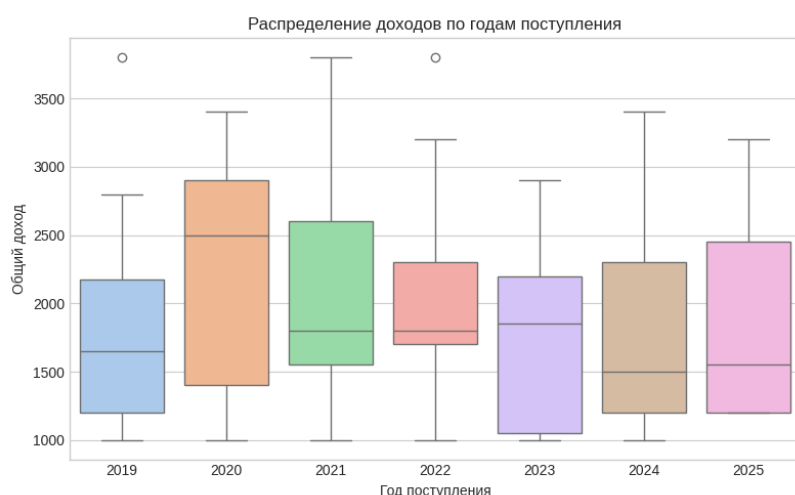


### 4.3. Доход по годам поступления

Boxplot-график **income\_by\_year.png** демонстрирует, что во всех годах поступления диапазон доходов сохраняется примерно одинаковым.

Это говорит о стабильной политике стипендий и отсутствии значительных временных колебаний.

Однако в отдельных годах выделяются студенты с высокими доходами (выбросы вверх), что отражает индивидуальные достижения.



## 5. Бизнес-инсайты и интерпретация

Анализ совокупного дохода студентов позволил сделать ряд аналитических выводов, полезных с точки зрения управления образовательной средой:

### 1. Стимулирующая роль олимпиад.

Студенты, участвующие в олимпиадах, формируют вторую группу с заметно более высоким доходом.

Это подчёркивает значимость внеучебной активности как инструмента финансовой и нематериальной мотивации.

### 2. Возможность прогнозирования успеваемости.

Показатель `total_income` может быть косвенным индикатором вовлечённости и академической успешности.

Высокие значения чаще встречаются у направлений, требующих аналитического и технического мышления (математика, информатика).

### 3. Управленческий вывод.

Университет может использовать подобные данные для:

- определения направлений, где требуется дополнительная поддержка студентов;
- планирования бюджетов на поощрения и гранты;
- анализа эффективности системы стипендий.

#### 4. Интеграция в BI-решения.

SQL-запросы, разработанные в ходе работы, могут быть встроены в хранилище данных (Data Warehouse) и использоваться в отчётности. Например, формирование ежемесячных дашбордов по доходам студентов и эффективности факультетов.

#### 5. Рекомендации.

На основе выявленных закономерностей можно предложить:

- усилить участие студентов гуманитарных направлений в олимпиадных активностях;
- разработать систему бонусных коэффициентов к стипендии за академические достижения;
- применять полученную модель для прогнозирования распределения грантов.

#### 6. Итог аналитического исследования

Проведённая работа продемонстрировала полный цикл аналитического решения:

- генерация и интеграция данных;
- их очистка и нормализация;
- консолидация в единую структуру;
- расчёт бизнес-показателя `total_income`;
- визуализация и интерпретация результатов;
- перенос аналитики в SQL-формат и упаковка в Docker-контейнер.

Сформированная модель может быть масштабирована для анализа доходов учащихся, премий сотрудников или эффективности программ мотивации. Результаты работы отражают понимание принципов ETL-процессов, аналитического мышления и инженерии данных в образовательных системах.

### Заключение

#### Вывод:

В ходе выполнения практической работы № 3 была реализована полная цепочка построения аналитического решения - от генерации данных до контейнеризации среды.

Поставленная цель - разработка и апробация процесса консолидации данных из разнородных источников с последующим расчётом бизнес-показателя - достигнута в полном объёме.

Я последовательно выполнила:

- генерацию трёх независимых наборов данных (CSV, Excel, JSON), имитирующих различные источники: студенческую базу, бухгалтерию и олимпиадный отдел;
- загрузку и предобработку данных в среде **Python/Pandas**, включая очистку, унификацию имён столбцов и проверку корректности типов;
- консолидацию таблиц на основе общих ключей (faculty, student\_id) и расчёт производного признака total\_income, отражающего суммарный доход студента;
- статистический и визуальный анализ показателя с помощью библиотек **Matplotlib** и **Seaborn**;
- генерацию SQL-скриптов, описывающих структуру данных и демонстрирующих запросы для агрегирования доходов;
- упаковку проекта в Docker-контейнер для обеспечения воспроизводимости и переносимости среды.

Аналитическая часть показала устойчивое распределение совокупных доходов и выявила закономерности между направлением подготовки и уровнем дополнительного дохода.

SQL-запросы подтвердили корректность логики объединения и возможность применения результатов в автоматизированных отчётных системах.

Практическая работа позволила мне закрепить навыки:

- интеграции данных в единую модель (ETL-подход);
- проведения статистического анализа и визуализации;
- документирования проекта в соответствии с академическими стандартами;
- применения принципов DevOps-реплицируемости через Docker.

Полученное решение демонстрирует не только владение инструментами анализа данных, но и понимание их роли в поддержке управленческих решений.

Разработанная модель может быть масштабирована для мониторинга эффективности программ мотивации, распределения грантов или анализа активности студентов по факультетам.

Таким образом, цель работы достигнута, а результат соответствует требованиям к практическим заданиям уровня **DCCA DBA**: реализовано воспроизводимое, аналитически обоснованное и структурно корректное решение.