Выполнили: Савельева Екатерина, Кулеева Жанна, Нагорянская Алина

Содержание

1. Цели проекта и его данные.

2. Концептуальное проектирование

a. Краткое описание предметной области.

Цель проекта: создать базу с помощью исходных и синтезированных данных, связанных с академическим и социальным бэкграундом учеников, которая бы содержала основу для формирования прогнозов относительно вероятности отчисления из школы. Для достижения этой цели мы нашли подходящий датасет, который можно взять за основу, выделили в нем сущности и связи между ними, а также сгенерировали новые данные, которые показались нам нужными для расширения датасета и наполнения нашей базы данных.

Данные проекта: для осуществления этой цели мы обратились к датасету с Kaggle[1], который содержит данные социологических опросов, посвященных вопросу отчисления учащихся из школ. Мы использовали этот датасет как основу для нашей базы данных и дополнили его новой информацией, которая была сгенерирована нами специально для данного проекта. Помимо датасета с помощью библиотеки faker были дополнительно сгенерированы данные, которые предлагают отсутствующую в изначальном варианте информацию об учителях и расширяют уже существующие характеристики школ и учебных курсов.

b. Описание процесса построения инфологической модели с обоснованием выделения сущностей и связей.

Наша база данных посвящена школе, а именно учебному процессу и деятельности учащихся в течение их учебы. Изначальный датасет включал в себя 34 колонки, которые относились к тем или иным характеристикам ученика:

[1] Student Dropout Analysis and Prediction Dataset: https://www.kaggle.com/datasets/abdullah0a/student-dropout-analysis-and-prediction-dataset

демографическая информация, академическая успеваемость, сведения о семье и внеклассной занятости:

1. **School:** название школы, которую посещает учащийся (например, MS).
2. **Gender:** пол учащегося (M – мужской, F – женский).
3. **Age:** возраст учащегося.
4. **Address:** место проживания (U – проживание в городе, R – проживание в сельской местности).
5. **Family_Size:** размер семьи учащегося (GT3 – более 3 человек, LE3 – менее трех человек или три человека).
6. **Parental_Status:** статус проживания родителей (A – проживают вместе, T – проживают раздельно).
7. **Mother_Education:** уровень образования матери (0 to 4).
8. **Father_Education:** уровень образования отца (0 to 4).
9. **Mother_Job:** тип занятости матери.
10. **Father_Job:** тип занятости отца.
11. **Reason_for_Choosing_School:** причина выбора школы (например, причиной могут быть курсы).
12. **Guardian:** взрослый, несущий ответственность за учащегося (например, мать).
13. **Travel_Time:** время, которое занимает дорога до школы (в минутах).
14. **Study_Time:** время на учебу еженедельно (от 1 до 4).
15. **Number_of_Failures:** количество предыдущих учебных неудач учащегося.
16. **School_Support:** получает ли учащийся дополнительную образовательную поддержку (да/нет).
17. **Family_Support:** обеспечивает ли семья образовательную поддержку для учащегося (да/нет).
18. **Extra_Paid_Class:** участие учащегося в дополнительных занятиях за отдельную плату (да/нет).
19. **Extra_Curricular_Activities:** вовлеченность учащегося во внеклассные занятия (да/нет).
20. **Attended_Nursery:** ходил ли учащийся в детский сад (да/нет).
21. **Wants_Higher_Education:** желание учащегося получить высшее образование да/нет).
22. **Internet_Access:** доступность интернета в доме учащегося (да/нет).
23. **In_Relationship:** наличие романтических отношений учащегося (да/нет).

24. **Family_Relationship:** качество отношений в семье учащегося (по шкале от 1 до 5).
25. **Free_Time:** количество свободного времени после школы у учащегося (по шкале от 1 до 5).
26. **Going_Out:** частота встреч учащегося с друзьями (по шкале от 1 до 5).
27. **Weekend_Alcohol_Consumption:** потребление алкоголя по выходным (по шкале от 1 до 5).
28. **Weekday_Alcohol_Consumption:** потребление алкоголя в будние дни (по шкале от 1 до 5).
29. **Health_Status:** оценка состояния здоровья учащегося (по шкале от 1 до 5).
30. **Number_of_Absences:** общее количество пропусков занятий в школе.
31. **Grade_1:** оценка учащегося, полученная на первой аттестации.
32. **Grade_2:** оценка учащегося, полученная на второй аттестации.
33. **Final_Grade:** итоговая оценка учащегося.
34. **Dropped_Out:** был ли отчислен учащийся (True/False).

На основе этого датасета мы выделили 12 сущностей. В первую очередь это учащиеся, которые стали своего рода центром нашей концептуальной схемы, далее мы выделили школы, которые посещают учащиеся; каждый ученик мог посещать на момент опроса только одну школу, но в каждой школе обучается множество учеников. Для школы мы дополнительно добавили новую сущность – статус школы, то есть является ли школа интернатом, государственной или частной школой. Также была создана новая сущность «учителя», которые могут официально быть трудоустроены только в одной школе, и в каждой школе соответственно может быть несколько учителей. На основе информации о том, что у учащихся в школах есть возможность посещать дополнительные курсы, мы создали сущность «курсы», которые разрабатываются учителями; учителя могут создать и вести несколько разных курсов одновременно (может существовать много курсов под одним названием, например, “математика”, но они различаются учебной программой, что отражается в их описании и количестве зачетных единиц, поэтому курсы с одним названием все равно считаются разными обособленными курсами, так как их создали разные учителя). Эти курсы посещаются учащимися, и каждый курс может быть выбран множеством учеников, которые в свою очередь могут внести в свой учебный план несколько курсов. Кроме того, было выделена сущность «академическая успеваемость», которая включает в себя

сведения об учебных успехах учащихся, и для каждого учащегося формируется своя уникальная академическая успеваемость.

Также были выделены сущности, связанные с личной информацией об учащихся. Сущность «родители» содержит в себе информацию о семье учащегося; у каждого учащегося может быть указано два родителя, и у родителей может быть несколько детей, которые обучаются в школе. Для обозначения уровня образования родителей выделяется сущность «образование родителей»; у каждого родителя может быть тот или иной уровень образования из пяти возможных вариантов. Для учащихся мы отдельно выделили сущности статуса здоровья и ментального здоровья, где у каждого ученика информация о здоровье уникальна и соответствует только одному человеку. С ментальным здоровьем также связана сущность «отношения в семье», так как ментальное состояние зависит от обстановки в семье.

Так как изначальный датасет и наш проект посвящены теме отчисления учащихся и факторов, которые на это влияют, мы выделили сущность «отчисления» – приказы об отчислении тех или иных учащихся с указанием возможной причины; в один документ об отчислении могут попасть сразу несколько учащихся, но для учеников с опытом отчисления на момент опроса может быть только единственный случай отчисления. Также мы решили добавить сущность «зачисление», которое по своим характеристикам аналогично отчислению, но опыт зачисления присутствует у всех учеников, в отличие от случаев отчисления.

c. ER-диаграмма с комментариями.

Students:

1. **id:** идентификационный номер учащегося.
2. **gender:** пол учащегося (M – мужской, F – женский).
3. **date_of_birth:** дата рождения учащегося (в формате ГГГГ-ММ-ДД); синтезированные данные.
4. **address:** адрес, по которому проживает учащийся (например, 9597 Wyatt Inlet Apt. 852 Jamesville, DC 62257); синтезированные данные, добавлены вместо изначального обозначения проживания в городе или сельской местности.
5. **school_id:** идентификационный номер школы, которую посещает учащийся.
6. **survey_id:** идентификационный номер опроса, в котором участвовал учащийся; в данном датасете все учащиеся проходили один и тот же опрос, который имеет номер «1».
7. **enrollment_id:** идентификационный номер приказа о зачислении учащегося.

Parents:

1. **id:** идентификационный номер родителя учащегося.
2. **gender:** пол родителя учащегося (M – мужской, F – женский).
3. **date_of_birth:** дата рождения родителя учащегося (в формате ГГГГ-ММ-ДД); синтезированные данные.
4. **edu_level:** уровень образования родителя (от 0 до 4).
5. **occupation:** тип занятости родителя (at_home, services, health, teacher, other).
6. **income:** доход родителя; синтезированные данные (в диапазоне от 1000 до 10000).
7. **student_id:** идентификационный номер ребенка; у одного ребенка могут быть указаны два родителя, у каждого из которых своя запись в таблице.
8. **guardian_status:** является ли родитель попечителем своего ребенка (0 – нет, 1 – да).

ParentsEduLevel:

Уровень образования родителей; данная таблица расшифровывает значения и дает описания уровней образования.

id	**description**
0	No formal education
1	Primary education
2	Not completed Secondary education
3	Completed Secondary education
4	Higher education

Schools:

1. **id:** идентификационный номер школы.
2. **address:** адрес школы; синтезированные данные (например, 3134 Hughes Mall Apt. 938 Hendersonshire, VI 65997)

3. **name:** название школы; синтезированные данные, добавлены вместо краткого названия школ в формате аббревиатуры из изначального датасета (например, New Erik High School).
4. **number_of_students:** количество учеников, которые обучаются в данной школе; синтезированные данные (в диапазоне от 50 до 600).
5. **headteacher:** имя и фамилия директора школы; синтезированные данные (например, Stephanie Edwards).
6. **status:** тип школы; синтезированные данные (варианты 0, 1 и 2).

SchoolsStatus:

Тип школы; данная таблица расшифровывает значения и дает описания разных типов школ – школа-интернат, государственная школа, частная школа.

id	**description**
0	boarding school
1	public school
2	private school

Teachers (синтезированные данные, в изначальном датасете какая-либо информация об учителях отсутствует):

1. **id:** идентификационный номер учителя.
2. **gender:** пол учителя (M – мужской, F – женский).
3. **name:** имя и фамилия учителя (например, Tracy Wagner)
4. **years_experience:** стаж работы учителем (в диапазоне от 1 до 30).
5. **school_id:** номер школы, в которой работает учитель; учитель может официально работать только в одной школе, но в одной школе может работать несколько учителей.
6. **date:** дата внесения информации об учителе.

Courses (синтезированные данные, в изначальном датасете присутствовала только информация о факте посещения учащимся каких-либо дополнительных курсов):

1. **id:** идентификационный номер курса.

2. **name:** название курса; нами было добавлено 13 разных курсов, причем некоторые из них идут в паре, так как за них должен отвечать один и тот же учитель:
    a. Mathematics, Physics;
    b. Chemistry, Biology;
    c. History, Economics;
    d. Geography;
    e. Art;
    f. Music;
    g. Physical Education;
    h. Literature, English;
    i. Computer Science.
3. **description:** описание курса; нами было добавлено три возможных варианта для описания курса:
    a. This course introduces the fundamental concepts of the subject;
    b. Students in this course will gain a comprehensive understanding of subject with practice;
    c. This fundamental course examines key themes and principles of core subject.
4. **teacher_id:** идентификационный номер учителя, который ведет этот курс.
5. **credits:** количество зачетных единиц, которые можно получить за прохождение этого курса (в диапазоне от 1 до 10).

Dropouts (синтезированные данные; в изначальном датасете был указан только факт отчисления):

1. **id:** идентификационный номер приказа об отчислении учащегося.
2. **date:** дата приказа об отчислении учащегося (в формате ГГГГ-ММ-ДД).
3. **reason:** причина отчисления учащегося по его словам; мы добавили 8 возможных причин:
    a. Family responsibilities;
    b. Financial difficulties;
    c. Health issues;
    d. Lack of interest in school;
    e. Bullying;
    f. Moving to a new location;

g. Mental health challenge;
h. Academic struggles.

4. **teacher_id:** идентификационный номер учителя, который подписывал приказ об отчислении учащегося.

DropoutsRecords:

Данная таблица приводит в соответствие идентификационный номер учащегося и идентификационный номер приказа о его отчислении; в одном приказе могут быть указаны несколько студентов.

1. **student_id:** идентификационный номер учащегося.
2. **dropout_id:** идентификационный номер приказа об отчислении учащегося.

AcademicPerfomances:

1. **id:** идентификационный номер сведений об успеваемости учащегося.
2. **date:** дата внесения информации об академической успеваемости учащегося.
3. **student_id:** идентификационный номер учащегося.
4. **absence_num:** общее количество пропусков занятий в школе.
5. **failures_num:** количество предыдущих учебных неудач учащегося.
6. **study_time:** время на учебу еженедельно (от 1 до 4).
7. **extra_classes:** участие учащегося в дополнительных занятиях (да/нет).
8. **higher_edu_expect:** желание учащегося получить высшее образование (да/нет).
9. **final_grade:** итоговая оценка учащегося.

HealthStatuses:

1. **id:** идентификационный номер сведений о здоровье учащегося.
2. **date:** дата внесения информации о статусе здоровья учащегося.
3. **student_id:** идентификационный номер учащегося.
4. **alcohol_consumption:** частота потребления алкоголя; нами было выделено пять возможных вариантов употребления алкоголя:
   a. No consumption;
   b. Low;
   c. Moderate;
   d. Moderate-high;
   e. High.

5. **state_of_health:** оценка состояния здоровья учащегося; нами было выделено пять вариантов для описания состояния здоровья:
    a. Poor;
    b. Fair;
    c. Average;
    d. Good;
    e. Excellent.

MentalHealthStatuses:

1. **id:** идентификационный номер сведений о ментальном здоровье учащегося.
2. **date:** дата внесения информации о статусе ментального здоровья учащегося.
3. **student_id:** идентификационный номер учащегося.
4. **romantic_relat:** состоит ли учащийся в романтических отношениях (да/нет).
5. **family_relat_score:** оценка отношений в семье учащегося.

FamilyRelationship:

Характеристика отношений в семье учащегося; данная таблица расшифровывает значения и дает описание отношениям в семье.

id	**description**
1	Very weak
2	Weak
3	Moderate
4	Strong
5	Very close

Surveys:

Информация об опросе, результаты которого составляют нашу базу данных; данному опросу приписан номер «1», указана дата проведения и его описание.

id	**date**	**description**
1	2024-12-01	Survey for a comprehensive analysis of factors influencing student dropout rates in secondary education (MIT)

Enrollments:

1. **id:** идентификационный номер приказа о зачислении учащегося.
2. **date:** дата приказа о зачислении учащегося.

3. Проектирование реляционной модели

a. Процесс перехода к реляционной модели.

1. Преобразование сущностей в таблицы:

Каждая сущность из ER-диаграммы преобразуется в таблицу. При этом атрибуты сущностей становятся столбцами таблиц. Пример: сущность Students становится таблицей с атрибутами id, gender, date_of_birth, address, school_id, survey_id, enrollment_id.

2. Идентификация первичных ключей:

Для таблиц выбирается первичный ключ (PK), обеспечивающий уникальную идентификацию записей. Пример: id в таблице Students.

3. Реализация связей.

Связи "один ко многим" (1:N):

Например, одна школа может быть связана со многими учащимися. Для этого в таблицу Students добавляется внешний ключ school_id, который ссылается на id в таблице Schools.

Связи "многие ко многим" (M:N):

Примеры: связь между студентами (Students) и родителями (Parents); связь между курсами (Courses) и учителями (Teachers). Каждый студент может иметь нескольких родителей, а каждый родитель может быть связан с несколькими студентами. Один курс может вестись несколькими учителями, и один учитель может преподавать несколько курсов.

4. Реализация наследования отсутствует, поскольку все таблицы являются уникальными.

5. Установление связей между таблицами через внешние ключи:

Внешние ключи связывают таблицы, поддерживая ссылочную целостность данных. Например, столбец school_id в таблице Students ссылается на таблицу Schools.

b. Диаграмма реляционной модели учитывается при создании диаграммы схемы БД.

Связи:

Связь между успеваемостью учащегося (AcademicPerfomances) и учащимися (Students). В таблице AcademicPerfomances есть внешний ключ student_id, ссылающийся на таблицу Students.

Связь между курсами (Courses) и учителями (Teachers). В таблице Courses есть внешний ключ teacher_id, ссылающийся на таблицу Teachers. Связь «многие ко многим»: один курс связан со многими учителями из разных школ, а один учитель может вести сразу несколько курсов.

Связь между отчислением (Dropouts) и учителями (Teachers). В таблице Dropouts есть внешний ключ teacher_id, ссылающийся на таблицу Teachers. Один учитель может подписать приказ об отчислении нескольких учащихся.

Связь между приказом об отчислении (DropoutsRecords) и учащимся (Students). В таблице DropoutsRecords есть внешний ключ student_id, ссылающийся на таблицу Students. В одном приказе могут быть указаны несколько студентов.

Связь между здоровьем (HealthStatuses) и учащимся (Students). В таблице HealthStatuses есть внешний ключ student_id, ссылающийся на таблицу Students.

Связь между ментальным здоровьем (MentalHealthStatuses) и учащимся (Students). В таблице MentalHealthStatuses есть внешний ключ student_id, ссылающийся на таблицу Students.

Связь между ментальным здоровьем (MentalHealthStatuses) и отношениями в семье учащегося (FamilyRelationship). В таблице MentalHealthStatuses есть внешний ключ family_relat_score, ссылающийся на таблицу FamilyRelationship.

Связь между родителями (Parents) и учащимися (Students). В таблице Parents есть внешний ключ student_id, ссылающийся на таблицу Students. Связь «многие ко многим»: каждый студент может иметь нескольких родителей, а каждый родитель может быть связан с несколькими студентами.

Связь между родителями (Parents) и образованием родителей (ParentsEduLevel). В таблице Parents есть внешний ключ edu_level, ссылающийся на таблицу ParentsEduLevel.

Связь между школами (Schools) и типом школы (SchoolsStatus). В таблице Schools есть внешний ключ status, ссылающийся на таблицу SchoolsStatus. Каждая школа имеет свой тип школы (школа-интернат, государственная школа, частная школа).
Связь между студентами (Students) и школами (Schools). В таблице Students есть внешний ключ school_id, ссылающийся на таблицу Schools. В одной школе может учиться много учащихся.
Связь между студентами (Students) и приказом о зачислении (Enrollments). В таблице Students есть внешний ключ enrollment_id, ссылающийся на таблицу Enrollments. В одном приказе о зачислении может быть несколько учащихся.
Связь между учащимися (Students) и номером опроса (Surveys). В таблице Students есть внешний ключ survey_id, ссылающийся на таблицу Surveys. Все учащиеся проходили один и тот же опрос, который имеет номер «1».
Связь между учителями (Teachers) и школами (Schools). В таблице Teachers есть внешний ключ school_id, ссылающийся на таблицу Schools. В одной школе может работать много учителей.

Первичные и внешние ключи:
AcademicPerfomances: id - PRIMARY KEY, student_id связан с Students - FOREIGN KEY.
Courses: id - PRIMARY KEY, teacher_id связан с Teachers - FOREIGN KEY.
Dropouts: id - PRIMARY KEY, teacher_id связан с Teachers - FOREIGN KEY.
DropoutsRecords: student_id, dropout_id - PRIMARY KEYS, student_id связан с Students, dropout_id связан с Dropouts - FOREIGN KEYS.
FamilyRelationship: id - PRIMARY KEY.
HealthStatuses: id - PRIMARY KEY, student_id связан с Students - FOREIGN KEY.
MentalHealthStatuses: id - PRIMARY KEY, family_relat_score связан с FamilyRelationship, student_id связан с Students - FOREIGN KEYS.
Parents: id - PRIMARY KEY, edu_level связан с ParentsEduLevel, student_id связан с Students - FOREIGN KEYS.
ParentsEduLevel: id - PRIMARY KEY.
Schools: id - PRIMARY KEY, status связан с SchoolStatus - FOREIGN KEY.
SchoolsStatus: id - PRIMARY KEY.
Students: id - PRIMARY KEY, school_id связан с Schools, enrollment_id связан с Enrollments, survey_id связан с Surveys - FOREIGN KEYS.

Teachers: id - PRIMARY KEY, school_id связан с Schools - FOREIGN KEY.

Ограничения:
Например, в таблице Students:
id — PRIMARY KEY.
school_id — FOREIGN KEY, ссылается на id таблицы Schools.
survey_id — FOREIGN KEY, ссылается на id таблицы Surveys.
enrollment_id — FOREIGN KEY, ссылается на id таблицы Enrollments.

c. По умолчанию в SQLite ограничение ссылочной целостности выключено (не производится форсирование внешних ключей). Для этого мы принудительно производим форсирование внешних ключей командой PRAGMA foreign_keys = ON.

Таблицы заполняются данными с учётом ограничений ссылочной целостности: сначала главные, а затем подчинённые. Другими словами, вначале заполняются данными таблицы, на которые указывают ссылки, а затем — таблицы, которые содержат ссылки. Таким образом, ссылки при добавлении строк в подчинённые таблицы будут валидными и не вызывают нарушений целостности.

- Типы данных:
AcademicPerfomances:
id INTEGER
date TEXT
student_id INTEGER
absence_num INTEGER
failures_num INTEGER
study_time REAL
extra_classes TEXT
higher_edu_expect TEXT
final_grade REAL
Courses:
id INTEGER
name TEXT
description TEXT

teacher_id INTEGER
credits INTEGER
Dropouts:
id INTEGER
date TEXT
reason TEXT
teacher_id INTEGER
DropoutsRecords:
student_id INTEGER
dropout_id INTEGER
Enrollments:
id INTEGER
date TEXT
FamilyRelationship:
id INTEGER
Description TEXT
HealthStatuses:
id INTEGER
date TEXT
student_id INTEGER
alcohol_consumption TEXT
state_of_health TEXT
MentalHealthStatuses:
id INTEGER
date TEXT
student_id INTEGER
romantic_relat TEXT
family_relat_score INTEGER
Parents:
id INTEGER
gender TEXT
edu_level INTEGER
occupation TEXT
date_of_birth TEXT

income REAL
student_id INTEGER
guardian_status INTEGER
ParentsEduLevel:
id INTEGER
description TEXT
Schools:
id INTEGER
address TEXT
name TEXT
number_of_students INTEGER
headteacher TEXT
status INTEGER
SchoolsStatus:
id INTEGER
description TEXT
Students:
id INTEGER
gender TEXT
date_of_birth TEXT
address TEXT
school_id INTEGER
survey_id INTEGER
enrollment_id INTEGER
Surveys:
id INTEGER
date TEXT
description TEXT
Teachers:
id INTEGER
gender TEXT
name TEXT
years_experience INTEGER
school_id INTEGER

date TEXT

d. Для обеспечения целостности данных используются ограничения уровня DDL и операции уровня DML (более подробно про это в следующем пункте).

4. Развёртывание БД в выбранной СУБД.

База данных с характеристиками студентов и контекста, связанного с ними (в виде иных сущностей в базе) была развернута с использованием СУБД SQLite. Создание и заполнение базы данными происходило с помощью кода, написанного на Python, в частности, с привлечением библиотеки sqlite3. Код по созданию и наполнению базы данными имеет следующую структуру: 1) *create_studentsdb.py* - DDL-скрипт для создания базы и таблиц внутри нее с учетом ограничений целостности; 2) *inserts_to_studentsdb.py* - DML-скрипт для наполнения созданной базы и таблиц данными; 3) *configs_studentsdb.py* - вынесенные за пределы DML-скрипта в отдельный класс конфиги, которые необходимы для синтеза данных, отсутствующих в исходном датафрейме; 4) *queries_studentsdb.py* - вынесенные за пределы DML-скрипта в отдельный класс запросы на вставку данных. Все перечисленные файлы расположены в папке **scripts** в GitHub репозитории проекта.

DDL-скрипт создания схемы БД и ограничения целостности данных

Данный скрипт (*create_studentsdb.py*) содержит в себе код по созданию базы данных students.db и таблиц внутри нее. Всего в базе было создано 15 таблиц в соответствии со схемой, продемонстрированной выше. При написании DDL-скрипта перед созданием каждой из таблиц прописывалось условие IF NOT EXISTS для того, чтобы дополнительно внутри кода не осуществлять проверку на предмет наличия той или иной таблицы в базе, а также для повышения эффективности отладки кода, в случаях, когда какая-либо из таблиц подлежала удалению, а DDL-скрипт при этом исполнялся полностью для создания удаленной таблицы. Всем внешним ключам в таблицах, для которых они были предусмотрены, присваивалось ограничение ON DELETE SET NULL для присвоения значений NULL в полях, являющихся внешними ключами в случае удаления соответствующих значений в “родительских” таблицах. Также ряду таблиц, помимо включения первичных и внешних ключей, были присвоены и иные ограничения целостности. Свод таблиц, включенных в базу, в т.ч. с указанием на типы используемых данных и ограничения приведен в файле **students_db.pdf** (расположен в репозитории проекта на GitHub). Так, таблицам присваивались такие ограничения

целостности как: 1) проверка на то, что подаваемое для заполнения таблицы со школами количество учащихся отлично от 0 - CHECK (number_of_students > 0); 2) проверка на то, что название и описание курса в соответствующей таблице не равны NULL - name TEXT NOT NULL, description TEXT NOT NULL; 3) также были наложены ограничения на ряд текстовых полей с фиксированным набором возможных значений, поскольку их заполнение опирается на результаты опросов с фиксированной шкалой: alcohol_consumption TEXT CHECK (alcohol_consumption IN ('No consumption', 'Low', 'Moderate', 'Moderate-high', 'High')), state_of_health TEXT CHECK (state_of_health IN ('Poor', 'Average', 'Fair', 'Good', 'Excellent')).

Работа некоторых ограничений целостности была проконтролирована в файле ***report_constraints_and_visualization.ipynb*** (в папке **reports** GitHub репозитория проекта). В частности, были осуществлены попытки внести в таблицу “некорректные” данные, которые нарушают целостность базы. Пример одного из блоков кода для проверки работы ограничений целостности в базе данных:

```
try:
    cursor.execute(f"INSERT INTO Schools (number_of_students) VALUES (?)",
                   (0,))
    students_db.commit()
    print("Values were successfully inserted into table!")
except Exception as e:
    print(f"An error occurred: {e}")
```

DML-скрипт для вставки данных в таблицу

DML-скрипт (*inserts_to_studentsdb.py*) имеет более разветвленную структуру, в частности, предполагает работу как с исходными, так и с синтезированными данными для их вставки в соответствующие таблицы базы. Поскольку в исходном датафрейме отсутствовали данные по ряду сущностей, в рамках проекта было принято решение сгенерировать эти данные. Для генерации использовалась библиотека faker[2], которая дает возможность производить синтез данных для широкого набора сущностей (имена, адреса и т.д.). Также при генерации данных использовался модуль random, который позволил в случайном порядке присваивать значения по определенным полям. В

[2]Документация библиотеки Faker: https://faker.readthedocs.io/en/master/

случаях, когда инструментов библиотеки faker оказывалось недостаточно, в частности, при необходимости сгенерировать список школьных предметов, их описания, возможные причины отчисления студентов, типы школ (т.е. развернутые специфические текстовые данные), осуществлялось обращение к большой языковой модели ChatGPT 4 для синтеза указанных текстовых данных. С опорой на данную модель было сгенерировано четыре массива, используемых в рамках DML-скрипта: *subjects*, *descriptions*, *dropout_reasons* и *school_types*. Остальные текстовые данные были получены исходя из самостоятельной интерпретации шкалы, используемой в рамках опроса студентов, данные по которому были использованы для создания базы.

Также для организации работы в рамках DML-скрипта в отдельные классы (в соответствующих .py файлах) были вынесены конфиги, которые использовались для генерации данных, а также сами тексты запросов для того, чтобы обеспечить возможность их более быстрого исправления и отладки. В рамках самого DML-скрипта содержится код по обработке исходных и синтезированных датафреймов, а также непосредственно вставка этих обработанных данных в соответствующие таблицы. В рамках DML-скрипта происходит вставка данных различных форматов (в соответствии с предопределенными на этапе DDL-скрипта типами данных) во все 15 таблиц БД.

5. Разработка клиентского приложения.

Примеры взаимодействия с данными из созданной базы приведены в файле ***report_constraints_and_visualization.ipynb***, расположенном в папке **reports** GitHub репозитории проекта. В файле расположены все sql-запросы, в т.ч. запрос для редактирования данных в базе данных, примеры визуализации, описания к построенным графикам. Пример запроса для редактирования данных в созданной базе:

```
try:
    cursor.execute(f"UPDATE Courses SET name = ? WHERE name = ?",
                   ('Economics & Social Sciences', 'Economics'))
    students_db.commit()
    print("Values in table were successfully updated!")
except Exception as e:
    print(f"An error occurred {e}")
```

Основное назначение созданной базы данных - возможность формирования сводной аналитики в разрезе различных групп студентов, а именно отчисленных из образовательного учреждения и соответственно не отчисленных. Предполагается, что на основе базы данных студентов возможно более эффективное и быстрое получение сведений о студентах как относительно их персональных характеристик, так и окружающего их контекста, создавать информативные визуализации на их основе. Помимо этого, созданная база может служить основой и для реализации анализа данных с уклоном в методы машинного обучения, поскольку занесенные в базу характеристики возможно переформировать в переменные для использования в соответствующих моделях, например, моделях линейной или логистической регрессий (логистической регрессии для прогнозирования условий, при которых студент будет отчислен).

В рамках данного отчета фокус был сделан именно на визуализации данных, выгруженных с помощью sql-запросов из базы. В частности, были созданы такие визуализации как:

1) Круговые диаграммы в разрезе групп отчисленных/не отчисленных студентов относительно качества взаимоотношений с родителями;

2) Круговые диаграммы в разрезе групп отчисленных/не отчисленных студентов относительно качества состояния здоровья;

3) Столбиковая диаграмма с выводом топа школ по % отчисленных студентов относительно общего количества учащихся в учебном заведении;

4) Боксплоты с репрезентацией распределения количественных характеристик - количества пропущенных занятий и количества несданных экзаменов в разрезе отчисленных/не отчисленных групп студентов.

Визуализации выполнялись с помощью библиотеки plotly. Всего было создано 4 графика с опорой на данные, полученные из трех запросов к базе.

Для выполнения каждой из визуализаций был осуществлен следующий пайплайн работы:
1) написание sql-запроса к базе данных, обработка исключений и ошибок при выгрузке данных из базы и их оформлении в датафрейм;

2) работа с выгруженным датафреймом относительно добавления дополнительных меток, агрегации, сортировки, мерджа дополнительных данных при необходимости;

3) визуализация полученных данных с помощью библиотеки Plotly.

Выполненные визуализации позволили более эффективно изучить не только структуру, но и содержательные особенности данных, выявить наличие выраженных эмпирических взаимосвязей между исследуемыми характеристиками и группами студентов, относительно которых они рассматриваются.

Запросы к базе выполнялись с опорой на библиотеку sqlalchemy, т.к. запрос исключительно средствами библиотеки sqlite3 не позволил получить названия полей напрямую, в выдаче были только индексы полей, соответственно для повышения эффективности и скорости работы с данными (в частности, для того, чтобы не приходилось вручную присваивать названия полей), была использована библиотека sqlalchemy. Для каждого запроса была предусмотрена обработка исключений.

Примеры визуализации, которые удалось получить посредством выгрузки данных из базы и использования библиотеки plotly для отрисовки графиков представлены ниже. Интерпретация полученных графиков также представлена в файле ***report_constraints_and_visualization.ipynb*.**

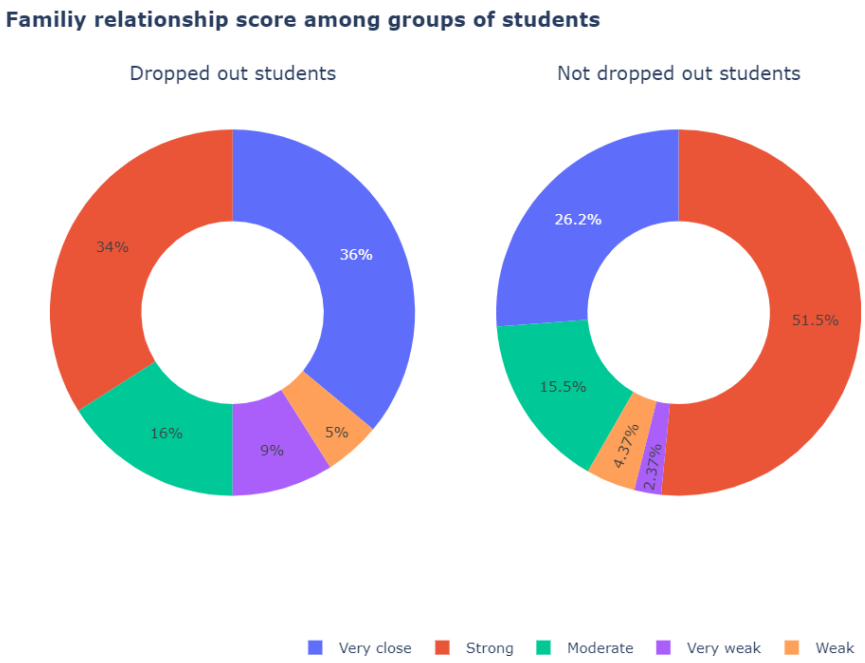

Рис. 1. Круговые диаграммы относительно распределения учащихся по качеству взаимоотношений с родителями в разрезе групп отчисленных и не отчисленных учащихся.

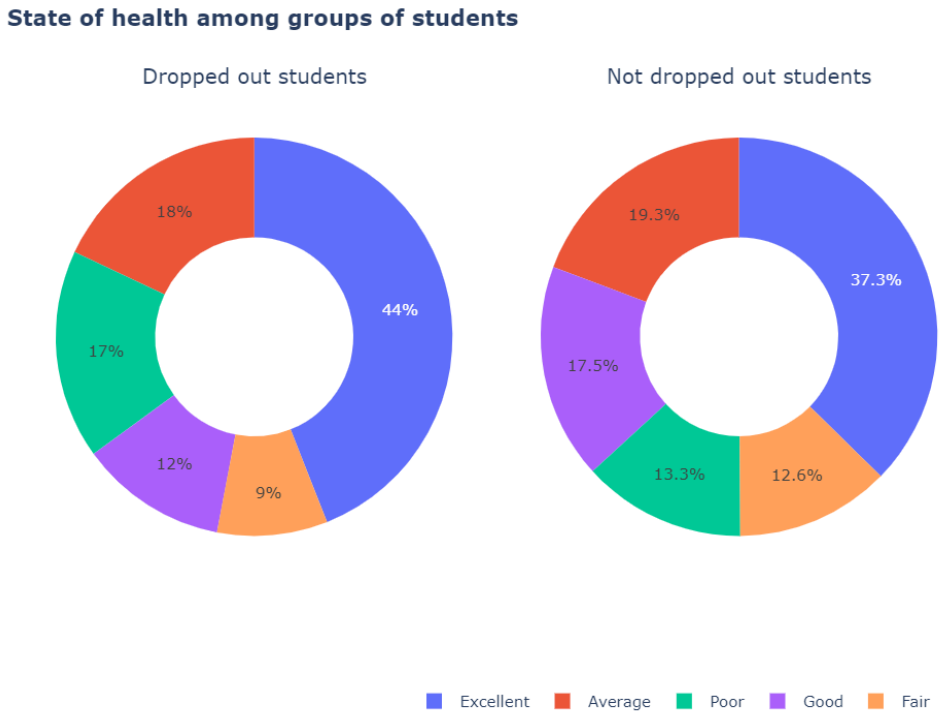

Рис. 2. Круговые диаграммы относительно распределения учащихся по оценке состояния здоровья в разрезе групп отчисленных и не отчисленных учащихся.

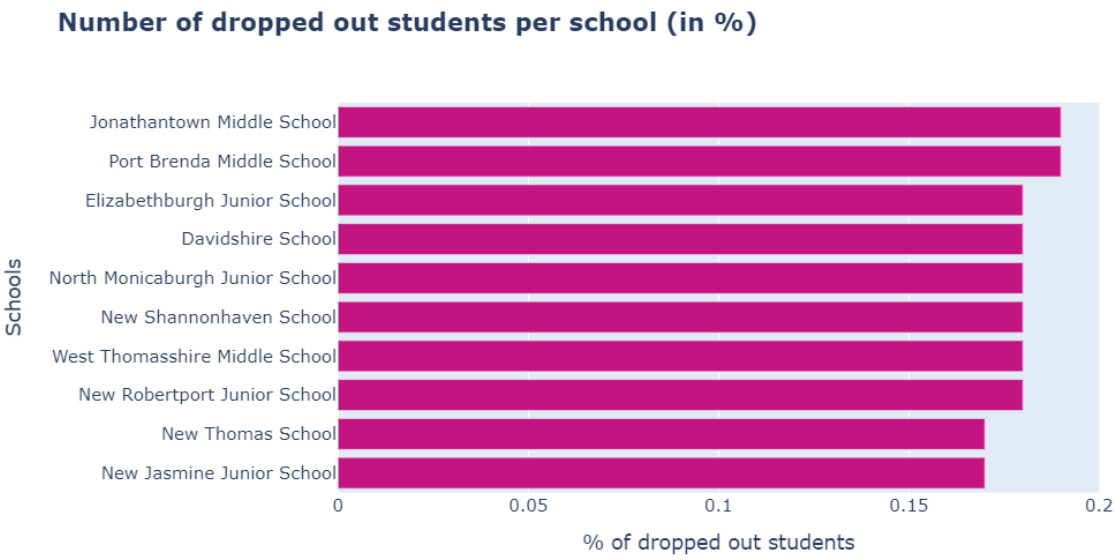

Рис. 3. Топ-школ по количеству отчисленных учащихся (в % от общего количества учащихся в образовательном учреждении)

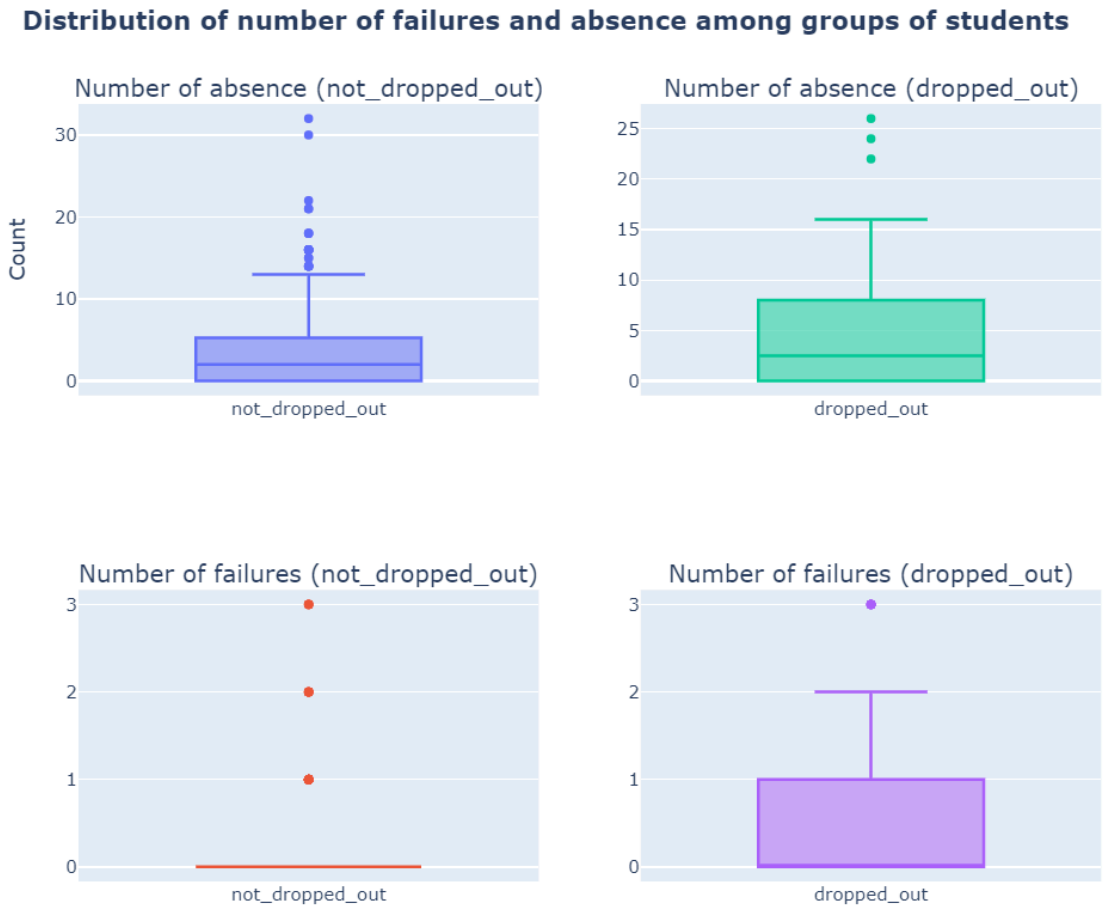

Рис. 4. Боксплоты для показателей количества пропусков и несданных экзаменов в разрезе групп отчисленных и неотчисленных учащихся.

Таким образом, благодаря тому, что из базы данных возможно делать комплексные выгрузки - с включением параметров из разных таблиц, выполнение визуализации становится более быстрой задачей. В частности, подобные запросы и визуализации могут выполняться в качестве эксплораторного анализа данных (EDA), чтобы более предметно ознакомиться с выборкой, в т.ч. в разрезе различных групп. Без базы данных выполнение аналогичных задач заняло гораздо большее количество времени, в частности, за счет осуществления шагов мерджа разных таблиц, переименования полей для осуществления мерджа (т.к. таблицы не имели бы четко определенных ключей), валидации данных (проверка корректности введенных данных, их редактирование прямо в коде при необходимости). Оформление сведений о студентах в единую базу данных нивелирует эти сложности и позволяет более гибко и эффективно работать с данными, поскольку в ней уже установлены соответствующие отношения между таблицами через ключи и в ряде случаев при необходимости реализована валидация данных, вносимых в базу.

Заключение

В рамках проекта на основе набора опросных данных об отчислениях студентов, находящегося в открытом доступе, была создана база, позволяющая более эффективно организовать работу с социологическими данными. Также в проекте были продемонстрированы возможности по синтезу данных относительно сущностей, которые отсутствуют в исходном датасете, в частности, с привлечением как отдельных библиотек (faker), так и LLM. Создание базы данных с интеграцией как персональных характеристик учащихся, так и об аспектах контекста, который их окружает (семья, школа, преподаватели и т.д.) открывает возможности для оптимизации выполнения задач по эксплораторному анализу данных, в т.ч. с использованием визуализации, апробации моделей машинного обучения (например, классификации относительно отчисленных/неотчисленных студентов и т.п.), тестирования различных sql-запросов, в т.ч. их отладки для взаимодействия с похожими базами. Таким образом, оформление разрозненных сведений в единую структуру (БД) с соблюдением принципов целостности позволяет обеспечить более удобное, быстрое и продуктивное взаимодействие с данными, в т.ч в исследовательских целях.