Билет №10

Теоретическая часть:

1 вопрос:

Оператор переименования атрибутов - это инструмент, используемый в базах данных и системах управления базами данных (СУБД), который позволяет изменить имя атрибута (поля) в таблице. Это может быть полезно, когда необходимо изменить имя поля для лучшей читаемости, соответствия стандартам или для устранения конфликтов имен с другими таблицами или системами.

Типы ключей в базах данных: Первичный ключ (Primary Key). Внешний ключ (Foreign Key). Составной ключ (Composite Key). Потенциальный ключ (Candidate Key). Альтернативный ключ (Alternate Key).

Типы данных в базах данных:

1. Числовые типы данных: целые числа (INTEGER, SMALLINT, TINYINT), десятичные числа (DECIMAL, NUMERIC), числа с плавающей точкой (FLOAT, REAL, DOUBLE PRECISION).

2. Строковые типы данных: символьные строки (CHAR, VARCHAR, TEXT), бинарные строки (BINARY, VARBINARY, BLOB).

3. Дата и время: дата (DATE), время (TIME), дата и время (DATETIME, TIMESTAMP).

4. Логический тип данных: BOOLEAN или BIT для хранения логических значений (истина или ложь).

5. Специфичные для СУБД типы данных: например, геопространственные типы данных (POINT, LINESTRING, POLYGON) в СУБД, поддерживающих геопространственные данные.

2 вопрос:

Алгоритм создания таблицы базы данных с использованием языка SQL состоит из следующих шагов:

1. Подключение к базе данных: Во-первых, необходимо подключиться к базе данных, в которой вы хотите создать таблицу. Это можно сделать с помощью команды

`USE database_name;`, где `database_name` - имя базы данных, к которой вы хотите подключиться.

2. Создание таблицы: Затем необходимо создать новую таблицу с помощью команды `CREATE TABLE table_name (column1 datatype, column2 datatype, column3 datatype, ...);`, где `table_name` - имя создаваемой таблицы, а `column1, column2, column3, ...` - имена столбцов таблицы с указанием их типов данных (`datatype`).

3. Определение первичного ключа: Если необходимо, определите первичный ключ для таблицы с помощью команды `ALTER TABLE table_name ADD PRIMARY KEY (column_name);`, где `table_name` - имя таблицы, а `column_name` - имя столбца, который будет использоваться в качестве первичного ключа.

Пример:

Предположим, что мы хотим создать таблицу `employees` с полями `id`, `first_name`, `last_name`, `email` и `phone_number`.

1. Подключение к базе данных:

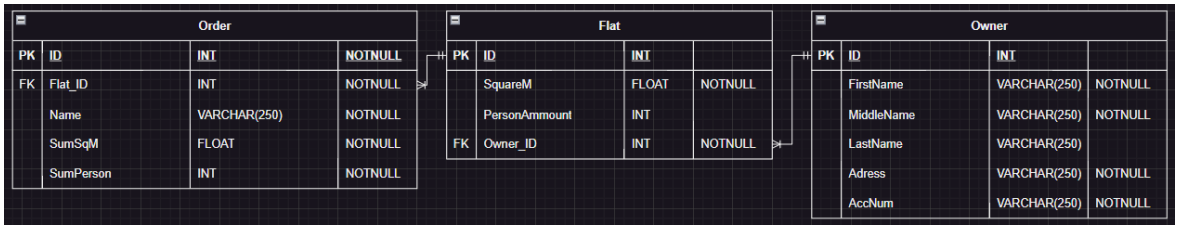
```
USE company;
```

2. Создание таблицы:

```
CREATE TABLE employees (
    id INT AUTO_INCREMENT,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    email VARCHAR(100),
    phone_number VARCHAR(15),
    PRIMARY KEY (id)
);
```

В этом примере мы создали таблицу `employees` с полями `id`, `first_name`, `last_name`, `email` и `phone_number`. Поле `id` имеет тип данных `INT` и используется в качестве первичного ключа. Функция `AUTO_INCREMENT` автоматически увеличивает значение `id` для каждой новой записи.

Практическая часть:



Билет 11

Теоретическая часть:

1 вопрос:

Запрос к базе данных (Database Query) - это команда или набор команд, используемых для взаимодействия с базой данных с целью извлечения, изменения или управления данными, хранящимися в ней. Запросы к базе данных позволяют пользователям и приложениям получать необходимую информацию, а также изменять структуру базы данных и ее содержимое.

Основные виды запросов к базе данных: Запросы на выборку (SELECT). Запросы на обновление (UPDATE). Запросы на вставку (INSERT). Запросы на удаление (DELETE). Запросы на создание (CREATE). Запросы на изменение (ALTER). Запросы на удаление (DROP).

Отличительные особенности запросов к базе данных:

1. Универсальность: запросы могут быть использованы для работы с различными типами баз данных и СУБД.

2. Гибкость: запросы позволяют обрабатывать данные различными способами, в зависимости от потребностей пользователя.

3. Эффективность: запросы оптимизированы для работы с большими объемами данных и могут быть легко масштабированы.

Назначение запросов к базе данных:

1. Извлечение данных: запросы на выборку позволяют пользователям и приложениям получать необходимую информацию из базы данных.

2. Управление данными: запросы на обновление, вставку и удаление позволяют изменять содержимое базы данных.

3. Управление структурой: запросы на создание, изменение и удаление позволяют управлять структурой базы данных, создавая, изменяя и удаляя таблицы, индексы, представления и другие объекты.

Синтаксис оператора изменения структуры таблицы (ALTER TABLE) в SQL:

```
ALTER TABLE table_name

ADD column_name datatype;

ALTER TABLE table_name

DROP COLUMN column_name;

ALTER TABLE table_name

ALTER COLUMN column_name datatype;
```

В данном примере показаны три основных действия, которые можно выполнить с помощью оператора ALTER TABLE: добавить столбец (ADD), удалить столбец (DROP COLUMN) и изменить тип данных столбца (ALTER COLUMN).

2 вопрос:

Характеристики и особенности проектирования БД (баз данных) включают в себя выбор подходящей модели данных, определение структуры данных, разработку схемы базы данных, проектирование пользовательских интерфейсов и т.д. В зависимости от выбранной модели данных, существуют различные подходы к проектированию баз данных.

Иерархическая модель данных - одна из ранних моделей данных, в которой данные организованы в виде иерархического дерева, где каждый узел может иметь только одного родителя, но несколько дочерних узлов.

Достоинства иерархической модели данных:

1. Простота и понятность: Иерархическая структура данных легко воспринимается и понимается, так как она напоминает структуру дерева.

2. Эффективность доступа: Доступ к данным в иерархической модели может быть быстрым и эффективным, особенно когда данные организованы последовательно на носителе.

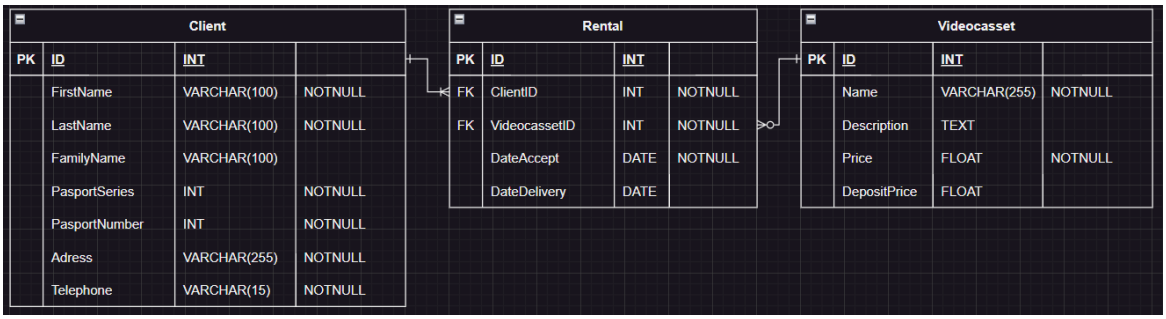
3. Устойчивость к изменениям: Иерархическая модель данных может хорошо справляться с изменениями в данных, так как изменения в родительских узлах автоматически распространяются на дочерние узлы.

Недостатки иерархической модели данных:

1. Ограниченная гибкость: Иерархическая модель данных имеет ограниченную гибкость, так как каждый узел может иметь только одного родителя. Это может затруднить представление сложных отношений между данными.

2. Трудности с обновлением данных: Обновление данных в иерархической модели может быть затруднено, особенно когда необходимо изменить структуру дерева или добавить новые типы отношений между данными. Уязвимость к одиночной точки отказа: В иерархической модели данных, если корневой узел или родительский узел становится недоступным, то может быть недоступным и весь набор дочерних узлов.

Практическая часть:



Билет №12

Теоретическая часть:

1 вопрос:

Модель данных - абстрактное представление структуры данных и операций над ними. Структуры данных - способы организации данных в памяти компьютера.

Основные операции: добавление, изменение, удаление и поиск данных.

Ограничения целостности - правила, гарантирующие корректность данных. Выбор модели данных зависит от потребностей проекта и характеристик данных.

2 вопрос:

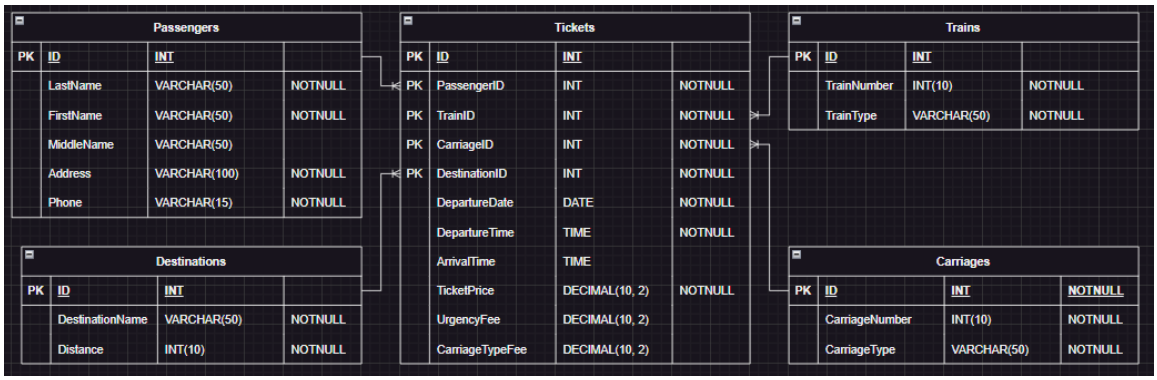
Сущность - объект реального мира, представленный в базе данных.

Типы сущностей: атрибутивные, событийные, временные.

Источники информации о сущностях: документы, наблюдения, интервью.

Классическая реляционная модель данных основана на таблицах, связях между ними и математических отношениях.

Практическая часть:



Билет №13

Теоретическая часть:

1 вопрос:

3НФ (Третья Нормальная Форма) - это уровень нормализации, при котором все неключевые атрибуты зависят только от первичного ключа и не зависят друг от друга. Алгоритм нормализации к 3НФ: удалить избыточные данные, выделить первичный ключ, убедиться, что неключевые атрибуты зависят только от первичного ключа, и разделить таблицы, если есть транзитивная зависимость.

2 вопрос:

INSERT - оператор для добавления новых данных в таблицу.

Синтаксис: `INSERT INTO table_name (column1, column2, ...) VALUES (value1, value2, ...);`.

UPDATE - оператор для изменения существующих данных.

Синтаксис: `UPDATE table_name SET column1 = value1, column2 = value2 WHERE condition;`.

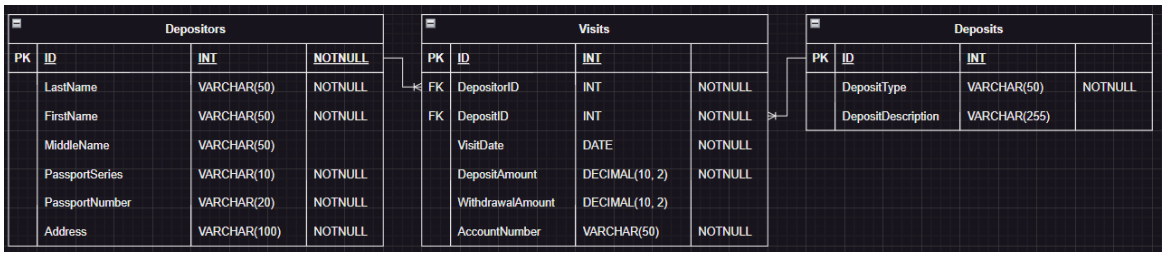
DELETE - оператор для удаления данных.

Синтаксис: `DELETE FROM table_name WHERE condition;`.

Отбор данных из одной таблицы: `SELECT column1, column2 FROM table_name WHERE condition;`.

Отбор данных из нескольких таблиц: `SELECT t1.column1, t2.column2 FROM table1 t1 INNER JOIN table2 t2 ON t1.id = t2.id WHERE condition;`.

Практическая часть:



Билет №14

Теоретическая часть:

1 вопрос:

Инфологическая модель - это абстрактное представление данных и связей между ними, не зависящее от конкретной СУБД.

Основные компоненты: сущности, атрибуты, идентификаторы, связи.

Состав инфологической модели включает определение сущностей, их атрибутов, идентификаторов и типов связей между сущностями.

Алгоритм построения: определить сущности, определить атрибуты и идентификаторы сущностей, определить связи между сущностями, провести нормализацию.

Пример: инфологическая модель базы данных "Библиотека" с сущностями "Книга", "Автор", "Жанр" и связями "написана_автором", "принадлежит_жанру".

2 вопрос:

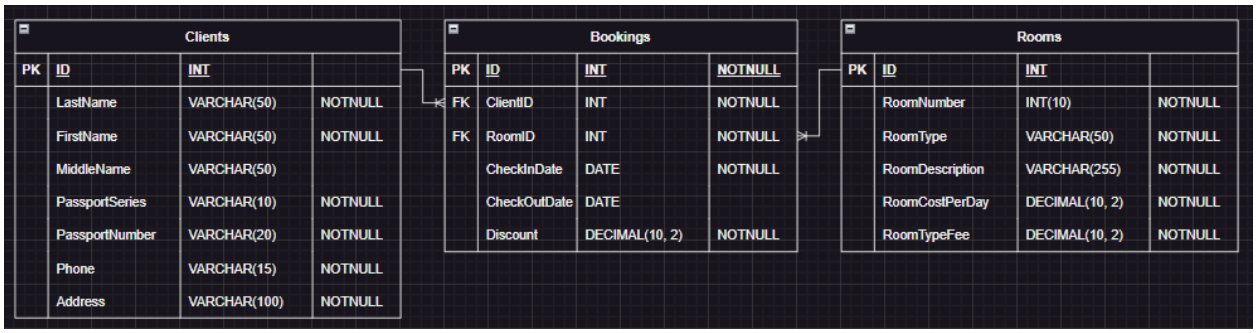
Виды БД: реляционные, иерархические, сетевые, объектно-ориентированные, NoSQL и др.

SQL-запросы: SELECT, INSERT, UPDATE, DELETE, CREATE, ALTER, DROP.

Примеры: `SELECT * FROM table;`, `INSERT INTO table (column1, column2) VALUES (value1, value2);`, `UPDATE table SET column1 = value1 WHERE condition;`, `DELETE FROM table WHERE condition;`.

Операции в группе DDL (Data Definition Language): CREATE (создание объектов), ALTER (изменение объектов), DROP (удаление объектов).

Практическая часть:

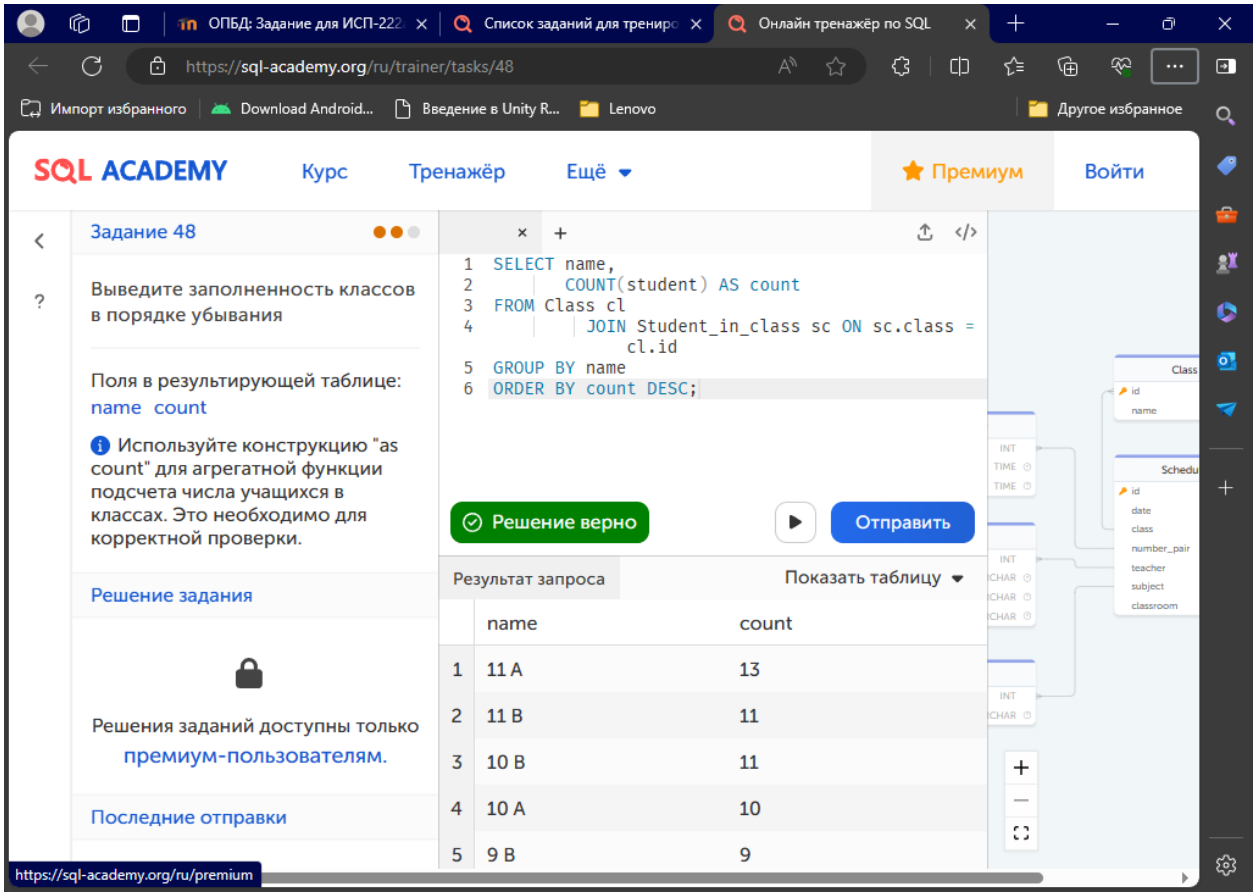


Задание 48. “Выведите заполненность классов в порядке убывания”

Выделяем столбец name и число student под именем count из таблицы Class. Присоединяем таблицу Class под имя cl через cl.id = sc.class где sc присоединенная таблица Student_in_class. Группируем по столбцу name и сортируем по столбцу count в обратном порядке.

```
SELECT name,
    COUNT(student) AS count
FROM Class cl
    JOIN Student_in_class sc ON sc.class = cl.id
GROUP BY name
```

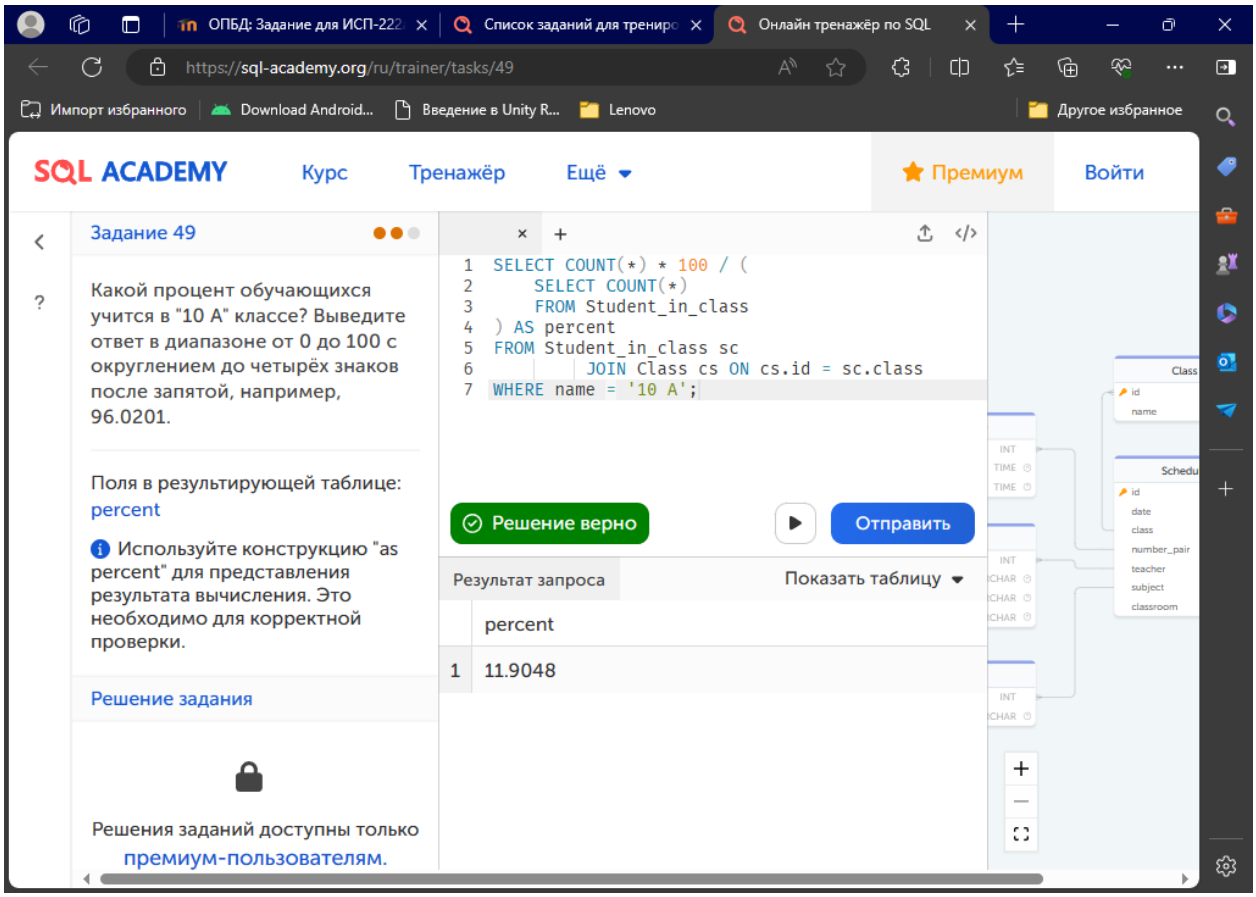
ORDER BY count DESC;



Задание 49. “Какой процент обучающихся учится в "10 А" классе? Выведите ответ в диапазоне от 0 до 100 с округлением до четырёх знаков после запятой, например, 96.0201.”

Выделяем количество всех значений в таблице Student_in_class объединённое с Class через cs.id = sc.class где sc Student_in_class, а cs Class где name = '10 A'. До умножаем на сто и делим на выделенное количество всех значений в таблице Student_in_class. Выводим под именем percernt.

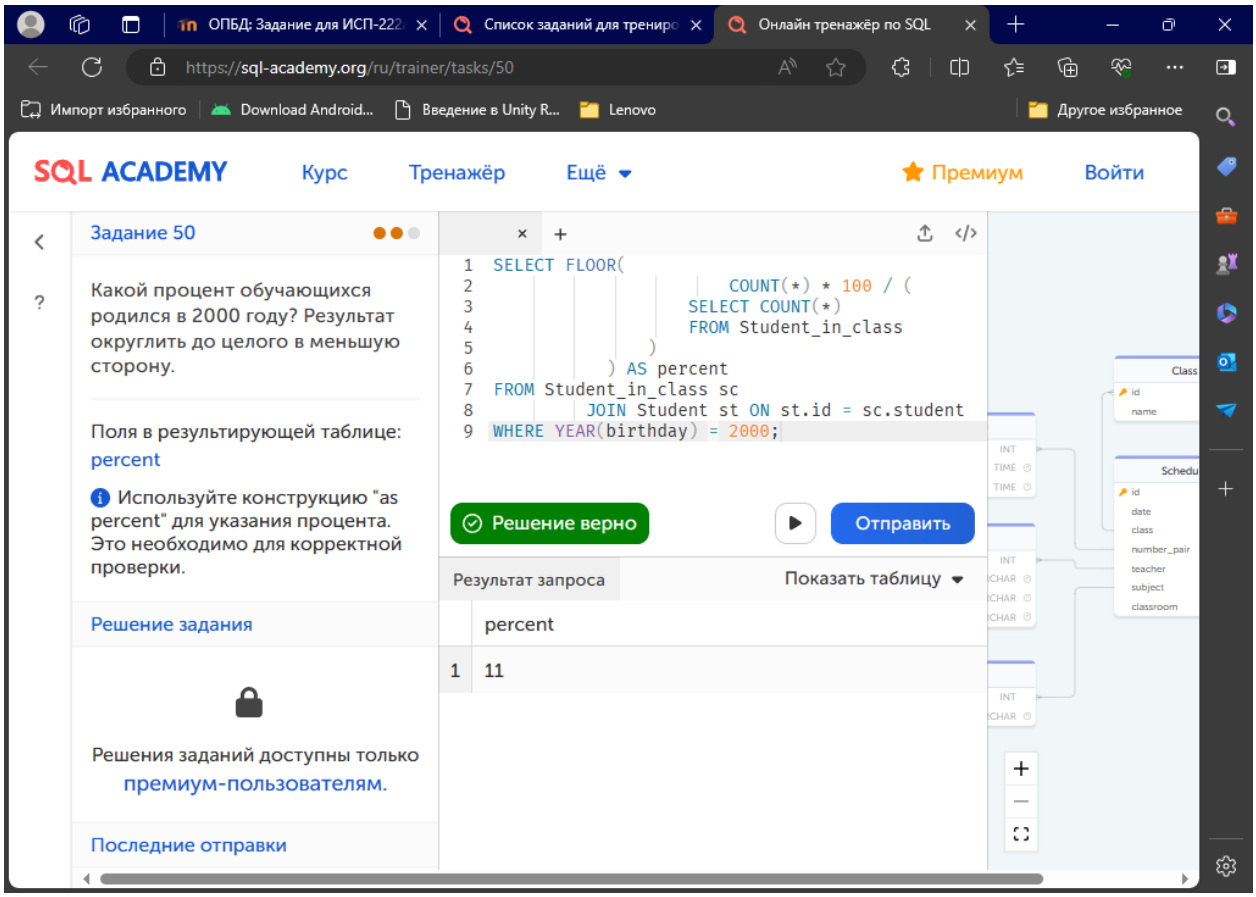
```
SELECT COUNT(*) * 100 / (
   SELECT COUNT(*)
   FROM Student_in_class
) AS percent
FROM Student_in_class sc
      JOIN Class cs ON cs.id = sc.class
WHERE name = '10 A';
```



Задание 50. “Какой процент обучающихся родился в 2000 году? Результат округлить до целого в меньшую сторону.”

Выделяем тип данных у всех значений из Student_in_class под именем sc соеденненных с Student под именем st через st.id = sc.student где год в birthday = 2000. Делим на количество всех значений в Student_in_class, выводим под именем percent.

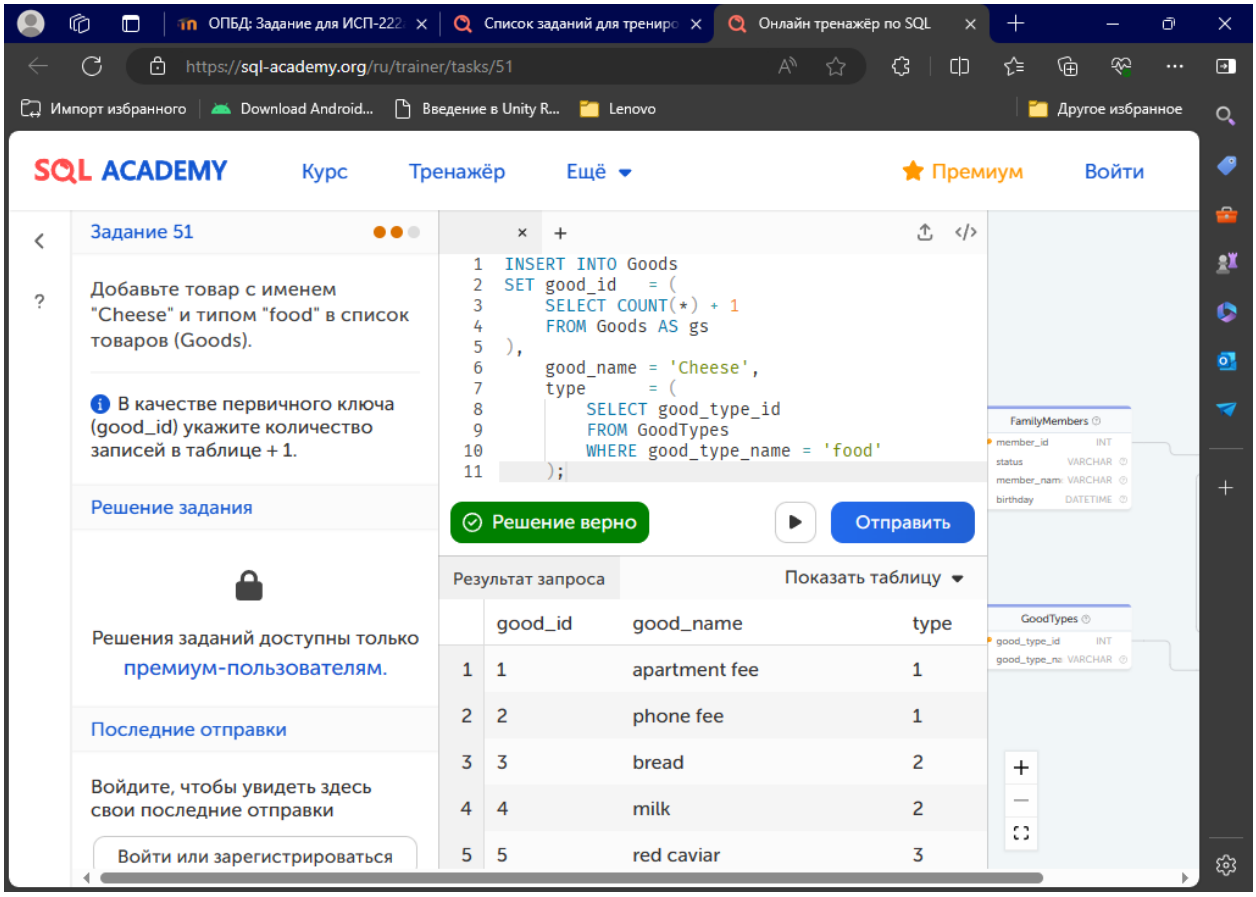
```
SELECT FLOOR(
                COUNT(*) * 100 / (
            SELECT COUNT(*)
            FROM Student_in_class
        )
    ) AS percent
FROM Student_in_class sc
    JOIN Student st ON st.id = sc.student
WHERE YEAR(birthday) = 2000;
```



Задание 51. “ Добавьте товар с именем "Cheese" и типом "food" в список товаров (Goods).”

Вставляем значения в таблицу Goods в столбик good_id, good_name и type. Значение в первом выделение всех значений + 1 в таблице Goods под именем gs, во втором 'Cheese', и в третьем выделенный столбик good_type_id из GoodTypes где good_type_name = 'food'.

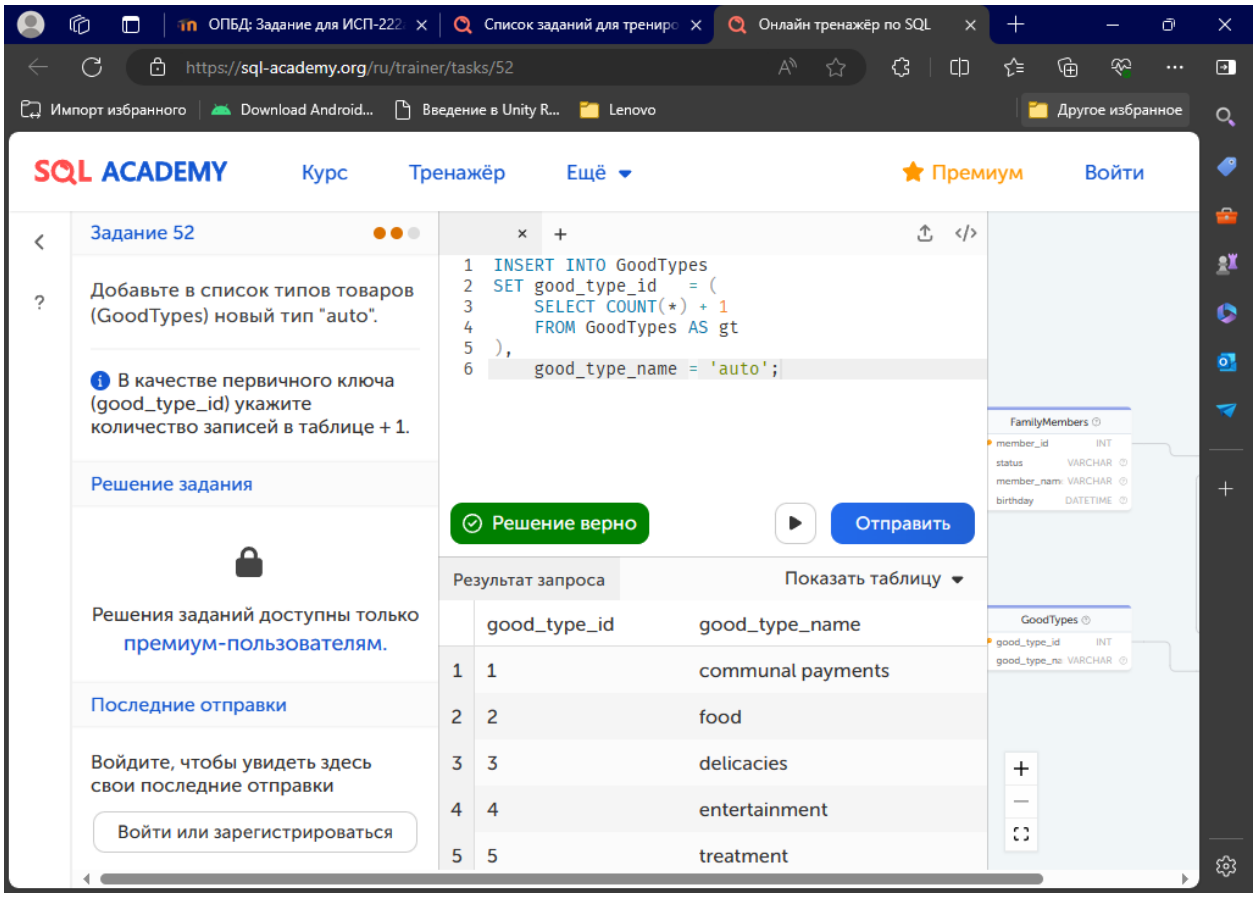
```
INSERT INTO Goods
SET good_id   = (
   SELECT COUNT(*) + 1
   FROM Goods AS gs
),
   good_name = 'Cheese',
   type      = (
      SELECT good_type_id
      FROM GoodTypes
      WHERE good_type_name = 'food'
   );
```



Задание 52. “ Добавьте в список типов товаров (GoodTypes) новый тип "auto".”

Вставляем значения в GoodTypes в столбики good_type_id и good_type_name. В первом число всех выделенных столбцов + 1 из GoodTypes под именем gt и во втором 'auto'.

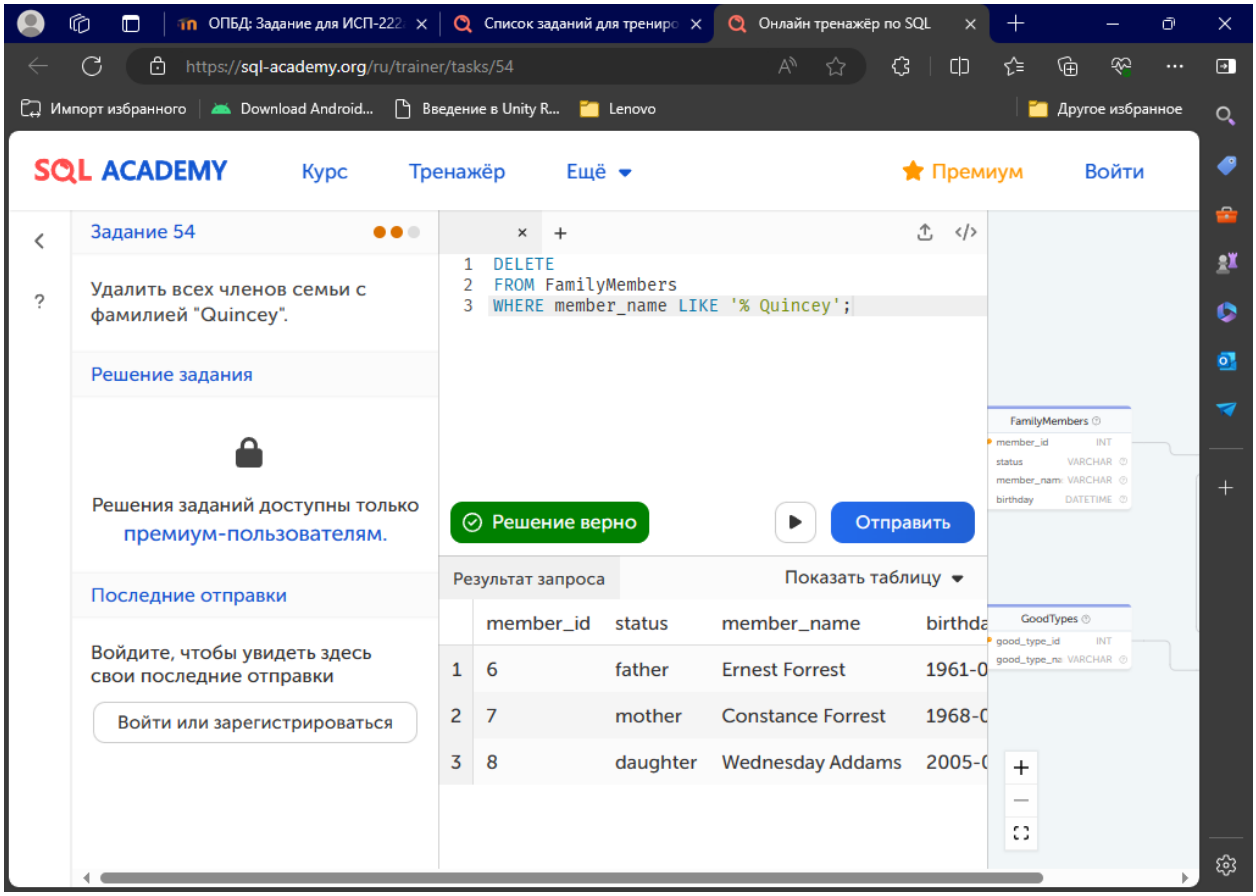
```
INSERT INTO GoodTypes
SET good_type_id   = (
   SELECT COUNT(*) + 1
   FROM GoodTypes AS gt
),
   good_type_name = 'auto';
```



Задание 54. “ Удалить всех членов семьи с фамилией "Quincey".”

Удаляем строчку из FamilyMembers где member_name заканчиваться на Quincey.

```
DELETE
FROM FamilyMembers
WHERE member_name LIKE '% Quincey';
```



.