

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
Национальный исследовательский ядерный университет «МИФИ»



Институт
интеллектуальных кибернетических систем

Кафедра кибернетики (№ 22)

Отчёт о работе по курсу
«Базы данных (теоретические основы баз данных)»

Вариант «Расписание и сдача ЕГЭ»

Выполнил	Панин И.С.
Группа	Б19-514
Вариант	Расписание и сдача ЕГЭ
Преподаватель	Петровская А.В.
Проверяющий	
Оценка	

Москва 2021

Содержание

1.	Формулировка задания	3
2.	Концептуальная модель базы данных	3
2.1.	Конкретизация предметной области	5
2.2.	Описание предметной области	5
2.3.	Описание атрибутов	5
3.	Логическое проектирование	7
4.	Физическое проектирование	8
4.1.	Создание таблиц	8
4.2.	Заполнение базы данных	11
4.2.1.	Подготовка данных	11
4.2.2.	Программа заполнения базы данных	12
4.2.3.	Результаты заполнения	16
5.	Выполнение запросов	20

1. Формулировка задания

Спроектировать базу данных для проведения Единого Государственного экзамена, проводящегося ежегодно в школах разных городов Российской Федерации. База данных должна содержать информацию о студентах, школах и учителях, а также отражать ежегодные данные по сдаваемым предметам, составленное расписание и полученные учениками результаты.

2. Концептуальная модель базы данных

После проведения анализа предметной области была спроектирована следующая концептуальная модель:

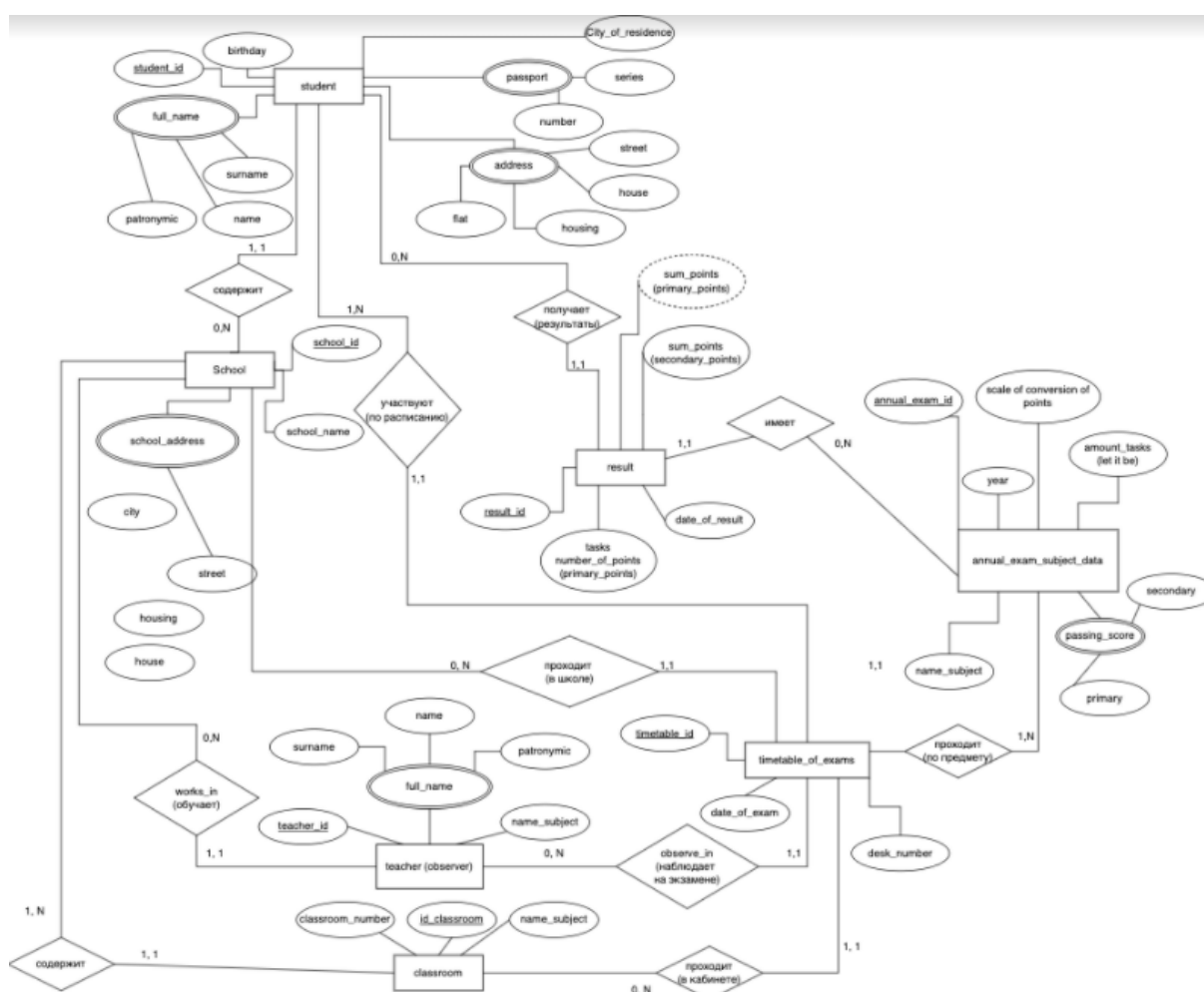


Рисунок 1 – Концептуальная модель базы данных

2.1. Конкретизация предметной области

Необходимо создать систему, отражающую информацию о проведении и результатах экзаменов по всей стране. По каждому предмету есть ежегодная информация, так как Министерство образования ежегодно вносит коррективы в тот или иной экзамен. База данных должна отражать точное расписание экзаменов по всем городам каждый год, а также результаты конкретного ученика по всем выбранным им предметам.

2.2. Описание предметной области

Система рассчитана на работу с зарегистрированными представителями ФИПИ, а также людьми, принимающими непосредственное участие в составлении расписания ЕГЭ. Обычные пользователи: ученики, родители учеников, учителя и т.д. доступа к этой базе данных не имеют.

Каждому конкретному ученику и родителю ученика в его личном кабинете доступна следующая информация: расписание проведения выбранных экзаменов, а также в дальнейшем результаты по каждому экзамену, включающие в себя разбалловку по каждому заданию, количество первичных баллов и количество баллов по столбальной шкале. В школу же (лично администрации данной школы) после объявления результатов по каждому конкретному экзамену приходят сведения по всем ученикам данной школы с их результатами. Затем по усмотрению администрации данная информация распространяется по классным руководителям и, наконец, доводится до учеников.

Каждый экзамен проводится в школе, где на каждую аудиторию, вмещающую до 18 человек назначен наблюдатель. Каждому ученику, сдающему конкретный экзамен в конкретной школе выделена аудитория и персональное место в этой аудитории.

2.3. Описание атрибутов

В процессе анализа были выделены следующие атрибуты, название и описание которых приведены в таблице ниже:

Имя атрибута	Расшифровка
id	Уникальный идентификатор. Есть у каждого объекта.

Имя, фамилия, Отчество	Имя, фамилия, отчество ученика, принимающего участие в экзаменах или учителя, который является наблюдателем на экзамене.
Серия и номер паспорта	Документом, удостоверяющим личность для допуска на экзамен, является русский паспорт.
Дата рождения	Дата рождения ученика.
Название предмета	Название предмета, экзамен по которому сдают ученики
Название школы	Название школы, где проводится экзамен.
Город, улица, дом, корпус	Адрес местонахождения школы или адрес места жительства ученика.
Дом	Квартира, где живет ученик.
Номер аудитории	Номер аудитории, где проводится тот или иной экзамен
Шкала перевода баллов	Шкала перевода первичных баллов в баллы от 0 до 100, которая меняется по желанию ФИПИ год от года.
Количество заданий	Количество заданий в экзамене по конкретному предмету в конкретном году.
Проходной первичный балл	Балл в первичной шкале, требуемый, чтобы экзамен считался сданным.
Проходной вторичный балл	Балл в стобальной шкале, требуемый, чтобы экзамен считался сданным.
Дата результата	Дата, когда пришел результат по экзамену за конкретный предмет.
Разбалловка по заданиям	Разбалловка, показывающая, в каких заданиях ученик допустил ошибки.
Сумма первичных баллов	Сумма первичных баллов, набранных учеником за конкретный экзамен
Сумма вторичных баллов	Сумма баллов по стобальной, набранных учеником за конкретный экзамен
Номер стола	Число от 1 до 18, показывающее место в аудитории, на котором будет писать экзамен ученик.

3. Логическое проектирование

Следующим шагом на основе КМПО была разработана логическая модель базы данных, представленная ниже:

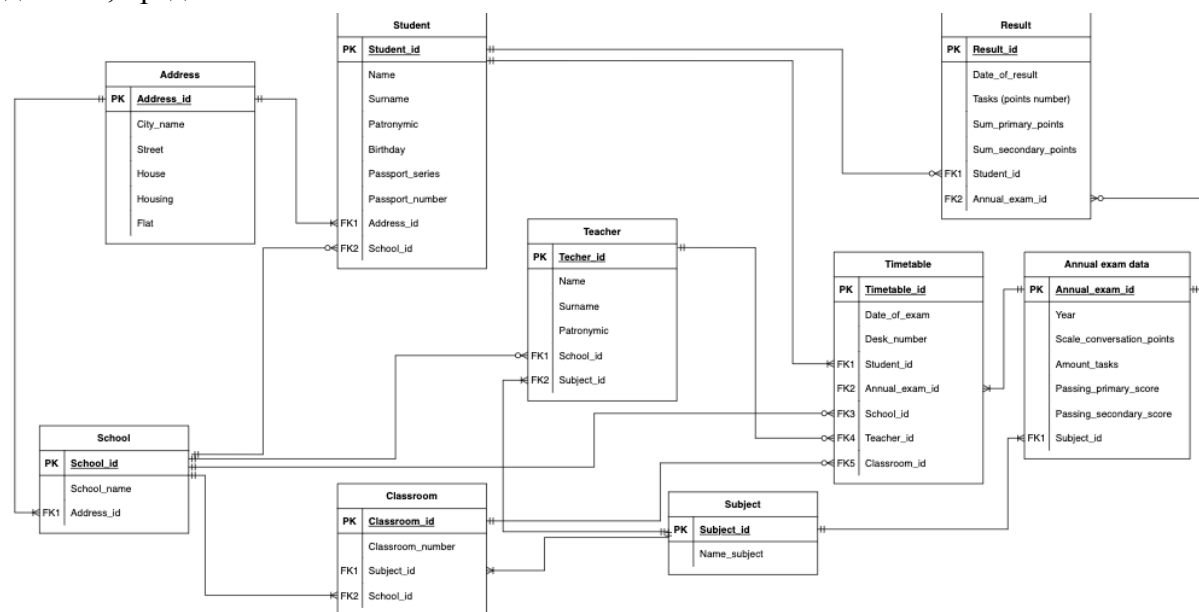


Рисунок 2 – Логическая модель базы данных

Предмет и адрес был вынесены в отдельную таблицу, чтобы соблюдалась 3 нормальная форма. Также показаны все обязательные и необязательные связи “один ко многим”.

Таблица “Расписание” является связующей таблицей между всеми, так как в нее включены почти все сущности, играющие ключевую роль в расписании ЕГЭ: ученики, наблюдатели, школы проведения, аудитории и данные по сдаваемому экзамену в конкретный год.

Таким образом, все таблицы базы данных находятся в нормальной форме.

4. Физическое проектирование

В качестве СУБД для реализации разработанной базы данных была выбрана PostgreSQL. В связи с проведённым анализом предметной области была проработана следующая физическая схема БД. Она представлена на следующем рисунке:

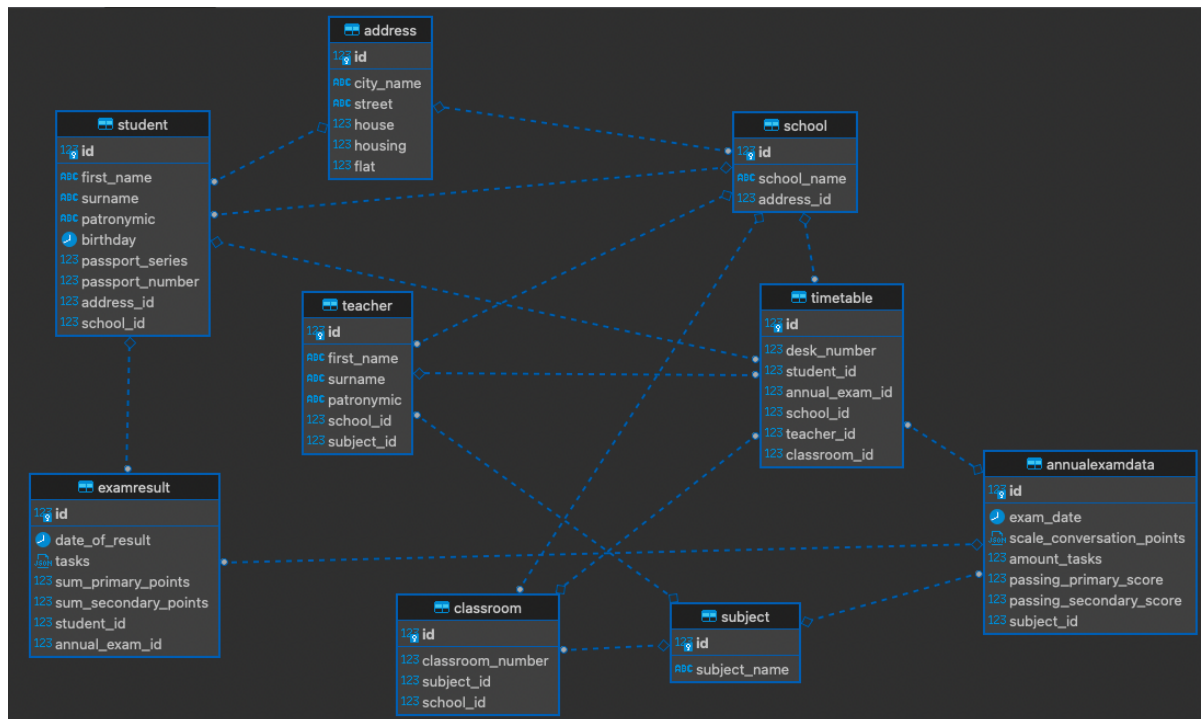


Рисунок 3 – Графическое представление базы данных

4.1. Создание таблиц

Ниже приведены запросы на языке SQL (диалект PostgreSQL) для создания таблиц, описанных выше.

- Код для создания таблицы “Address”:

```
create table if not exists Address(  
    Id bigserial not null primary key,  
    City_name varchar(30) not null,  
    Street varchar(50) not null,  
    House integer check (House > 0) not null,  
    Housing integer check (Housing > 0),  
    Flat integer check (Flat > 0)  
);
```

- Код для создания таблицы “School”:

```
create table if not exists School(  
    Id serial not null primary key,  
    School_name varchar(255) not null,
```

```
Address_id bigint not null references Address (Id) on delete cascade on update cascade  
);
```

- Код для создания таблицы “Student”:

```
create table if not exists Student  
(  
    Id bigserial not null primary key,  
    First_name varchar(25) not null,  
    Surname varchar(25) not null,  
    Patronymic varchar(25),  
    Birthday date not null,  
    Passport_series int not null check (Passport_series > 999 and Passport_series < 10000),  
    Passport_number bigint not null check (Passport_number > 99999 and Passport_number <  
1000000),  
    Address_id bigint not null references Address (Id) on delete cascade on update cascade,  
    School_id int references School(Id) on delete set null on update cascade,  
    unique (Passport_series, Passport_number)  
);
```

- Код для создания таблицы “Subject”:

```
create table if not exists Subject(  
    Id serial not null primary key,  
    Subject_name varchar(30) not null unique  
);
```

- Код для создания таблицы “Classroom”:

```
create table if not exists Classroom(  
    Id bigserial not null primary key,  
    Classroom_number int not null check (Classroom_number > 0),  
    Subject_id int not null references Subject(Id) on delete set null on update cascade,  
    School_id int not null references School(Id) on delete cascade on update cascade  
);
```

- Код для создания таблицы “Teacher”:

```
create table if not exists Teacher(  
    Id bigserial not null primary key,  
    First_name varchar(25) not null,  
    Surname varchar(25) not null,  
    Patronymic varchar(25),  
    School_id int references School(Id) on delete set null on update cascade,  
    Subject_id int not null references Subject(Id) on delete cascade on update cascade  
);
```

- Код для создания таблицы “AnnualExamData”:

```
create table if not exists AnnualExamData(  

```



```

        Id serial not null primary key,
        Exam_date timestamp not null check (date_part('year', Exam_date) > 2000),
Scale_conversation_points json,
        Amount_tasks int not null check (Amount_tasks > 0),
        Passing_primary_score int not null check (Passing_primary_score > 0 and
Passing_primary_score <= 100),
        Passing_secondary_score int not null check (Passing_secondary_score > 0 and
Passing_secondary_score <= 100),
        Subject_id int not null references Subject(Id) on delete cascade on update cascade
    );

```

- Код для создания таблицы “ExamResult”:

```

create table if not exists ExamResult(
    Id bigserial not null primary key,
    Date_of_result timestamp,
    Tasks json,
    Sum_primary_points int default 0 check (Sum_primary_points >= 0),
    Sum_secondary_points int default 0 check (Sum_secondary_points >= 0),
    Student_id bigint references Student(Id) on delete set null on update cascade,
    Annual_exam_id int references AnnualExamData(Id) on delete set null on update cascade,
    unique (Student_id, Annual_exam_id)
);

```

- Код для создания таблицы “TimeTable”:

```

create table if not exists Timetable(
    Id bigserial not null primary key,
    Desk_number smallint check (Desk_number > 0 and Desk_number < 19),
    Student_id bigint references Student(Id) on delete cascade on update cascade,
    Annual_exam_id int references AnnualExamData(Id) on delete cascade on update cascade,
    School_id int references School(Id) on delete set null on update cascade,
    Teacher_id bigint references Teacher(Id) on delete set null on update cascade,
    Classroom_id bigint references Classroom(Id) on delete set null on update cascade,
    unique (Student_id, Annual_exam_id)
);

```

4.2. Заполнение базы данных

Заполнение базы данных проводилось при помощи библиотеки для работы с PostgreSQL psycopg2 языка программирования Python.

4.2.1. Подготовка данных

Были подготовлены массивы имен, фамилий, отчеств, городов, улиц, а также названий школ разных городов. Взят список предметов, экзамены по которым будут доступны для сдачи, а также найдены шкалы переводов баллов из первичой шкалы во вторичную вместе с разбалловкой по каждому заданию.

Придуманы функции генерации даты рождения, даты проведения экзаменов и даты результата. Разработан алгоритм, который позволяет наиболее приближенно к реальности заполнить таблицы по ежегодным сведениям об экзамене, по расписанию и по результатам экзаменов.

4.2.2. Программа заполнения базы данных

- Код, выполняющий вставку данных в таблицу “Address” адреса школ и учеников:

```
from random import choice, randint

from insertion.database import CITIES, STREETS, HOUSES,
HOUSINGS, FLATS

def address_insertion(connection, number=200):
    cursor = connection.cursor()
    for _ in range(number):
        city, street, house, housing, flat =
choice(CITIES), choice(STREETS), HOUSES(), HOUSINGS(),
FLATS()
        cursor.execute(
            f"INSERT INTO Address (City_name, Street,
House, Housing, Flat) "
            f"VALUES (\'{city}\', \'{street}\', {house},
{housing}, {flat})"
        )
        connection.commit()
        print("Addresses added successfully!")

def address_school_insertion(connection, number=40):
    cursor = connection.cursor()
    for _ in range(number):
        if randint(0, 1):
            city, street, house, = choice(CITIES),
choice(STREETS), HOUSES()
            cursor.execute(
                f"INSERT INTO Address (City_name, Street,
House) "
                f"VALUES (\'{city}\', \'{street}\',
{house})"
```

```

    )
    else:
        city, street, house, housing = choice(CITIES),
        choice(STREETS), HOUSES(), HOUSINGS()
        cursor.execute(
            f"INSERT INTO Address (City_name, Street,
House, Housing) "
            f"VALUES (\'{city}\', \'{street}\',
{house}, {housing})"
        )
        connection.commit()
        print("School addresses added successfully!")

```

- Код, выполняющий вставку информацию по аудиториям, в которых будут писать экзамены ученики:

```

from random import choice

from insertion.database import CLASSROOMS

def classroom_insertion(connection, classrooms_per_school=15):
    cur = connection.cursor()
    cur.execute(
        f"SELECT id FROM subject"
    )
    all_subject_id = list(cur.fetchall())
    cur.execute(
        f"SELECT id FROM school"
    )
    all_school_id = list(cur.fetchall())

    # There are number_classrooms_at_school classrooms for each
    school!
    for school_id in map(lambda item: item[0], all_school_id):
        already_used_numbers = set()
        for _ in range(classrooms_per_school):
            classroom = CLASSROOMS()
            # no equal numbers at the same school!
            while classroom in already_used_numbers:

```

```

        classroom = CLASSROOMS()
        already_used_numbers.add(classroom)
        subject_id = choice(all_subject_id)[0]
        cur.execute(
            f"INSERT INTO classroom (classroom_number,
subject_id, school_id) "
            f"VALUES ({classroom}, {subject_id}, {school_id})"
        )
        connection.commit()
        print("Classrooms added successfully!")

```

- Код, выполняющий вставку информации про учеников, принимающих участие в сдаче экзаменов:

```

from random import randint, choice
from copy import copy

from insertion.database import FIRST_FEMALE_NAMES,
SECOND_FEMALE_NAMES, FEMALE_PATRONYMICS
from insertion.database import FIRST_MALE_NAMES,
SECOND_MALE_NAMES, MALE_PATRONYMICS
from insertion.database import CITIES, BIRTHDAYS,
PASSPORT_NUMBERS, PASSPORT_SERIES

def student_insertion(connection, number=200):
    """Город школы такой же, как и город проживания ученика, иначе бред!"""
    cur = connection.cursor()
    cities = copy(CITIES)
    # [(id, city_name), (id, city_name), ]
    cur.execute(
        f"SELECT id, city_name FROM address WHERE flat IS NOT NULL"
    )
    available_student_addresses = list(cur.fetchall())
    for _ in range(number):
        sex = randint(0, 1) # пол ученика
        first_name = choice(FIRST_FEMALE_NAMES) if sex else
choice(FIRST_MALE_NAMES)
        second_name = choice(SECOND_FEMALE_NAMES) if sex else
choice(SECOND_MALE_NAMES)
        patronymic = choice(FEMALE_PATRONYMICS) if sex else

```

```

choice(MALE_PATRONYMICS)
    birthday = BIRTHDAYS()
    passport_series = PASSPORT_SERIES()
    passport_numbers = PASSPORT_NUMBERS()
    student_city = choice(cities)

    student_address_id = choice(list(filter(lambda address:
address[1] == student_city,
available_student_addresses))))[0]
    # [(sc.id,), (sc.id, )]
    cur.execute(
        f"SELECT sc.id from address AS a inner join school AS
sc"
        f" on a.id = sc.address_id and a.city_name =
\'{student_city}\'"
    )
    available_schools = list(cur.fetchall())
    student_school_id = choice(available_schools)[0]
    cur.execute(
        f"INSERT INTO student (first_name, surname, patronymic,
birthday,"
        f" passport_series, passport_number, address_id,
school_id) "
        f"VALUES (\'{first_name}\', \'{second_name}\',
\'{patronymic}\', \'{birthday}\', {passport_series}, "
        f"{passport_numbers}, {student_address_id},
{student_school_id})"
    )
    available_student_addresses.remove((student_address_id,
student_city))
    if len(list(filter(lambda address: address[1] ==
student_city, available_student_addresses))) == 0:
        cities.remove(student_city)

connection.commit()
print("Students added successfully!")

```

Как видно, выполняется обыкновенный INSERT-запрос, который содержит данные, выбранные случайным образом. Полный текст скрипта для заполнения базы данных доступен в репозитории https://github.com/EvilMurlok/exams_database

4.2.3. Результаты заполнения

Далее представлены результаты работы программы на примере таблиц, соответствующих функциям, приведенным выше.

- Таблица “Address”:

address 1 × school 1 (2) student 1 (3) subject 1 (4) teacher 1 (5)							
select * from address Введите SQL выражение чтобы отфильтровать результаты							
Таблица	123 id	ABC city_name	ABC street	123 house	123 housing	123 flat	
	1	Казань	Заречная	7	4	95	
	2	Новосибирск	Полевая	84	2	183	
	3	Самара	Лесная	90	2	448	
	4	Новосибирск	Школьная	89	1	303	
	5	Новосибирск	Полевая	18	8	113	
	6	Новосибирск	Новая	97	5	350	
	7	Новосибирск	Центральная	70	6	15	
	8	Самара	Заречная	3	6	417	
	9	Екатеринбург	Лесная	94	3	158	
	10	Нижний Новгород	Советская	15	5	459	

Адреса школ:

address 1 × school 1 (2) student 1 (3) subject 1 (4) teacher 1 (5)							
select * from address Введите SQL выражение чтобы отфильтровать результаты							
	123 id	ABC city_name	ABC street	123 house	123 housing	123 flat	
201	201	Новосибирск	Ленина	12	7	[NULL]	
202	202	Екатеринбург	Советская	45	6	[NULL]	
203	203	Новосибирск	Центральная	91	[NULL]	[NULL]	
204	204	Казань	Полевая	67	[NULL]	[NULL]	
205	205	Челябинск	Новая	65	[NULL]	[NULL]	
206	206	Екатеринбург	Новая	63	[NULL]	[NULL]	
207	207	Москва	Советская	55	7	[NULL]	
208	208	Самара	Ленина	49	10	[NULL]	
209	209	Москва	Новая	31	[NULL]	[NULL]	
210	210	Казань	Заречная	30	3	[NULL]	
211	211	Санкт-Петербург	Садовая	11	3	[NULL]	

- Таблица “School”:

id	school_name	address_id
1	Муниципальное бюджетное общеобразовательное учреждение "Гимназия № 80"	201
2	Муниципальное автономное общеобразовательное учреждение "Средняя общеоб	202
3	Муниципальное бюджетное общеобразовательное учреждение средняя общеобра	203
4	Муниципальное автономное общеобразовательное учреждение "Школа № 27"	204
5	Муниципальное бюджетное общеобразовательное учреждение "Гимназия № 58"	205
6	Муниципальное автономное общеобразовательное учреждение "Средняя общеоб	206
7	Государственное бюджетное общеобразовательное учреждение "Лицей информа	207
8	Муниципальное автономное общеобразовательное учреждение "Средняя общеоб	208

- Таблица “Student”:

id	first_name	surname	patronymic	birthday	passport_series	passport_number	address_id	school_id
1	Anastasia	Trofimova	Alexandrovna	2000-03-11	1 907	649 898	165	21
2	Vasily	Panin	Ivanovich	2000-08-30	9 826	349 573	23	4
3	Peter	Lubimov	Alexandrovich	2004-11-24	7 264	201 503	29	4
4	Denis	Ivanov	Sergeevic	2005-10-21	5 146	764 875	85	15
5	Alla	Lubimova	Sergeevna	2005-05-26	8 347	258 178	33	22
6	Olga	Anisimova	Alexandrovna	2004-01-19	6 166	826 494	128	12
7	Anastasia	Pimenova	Petrovna	2001-11-19	7 304	911 304	28	8
8	Alina	Pimenova	Denisovna	2004-04-10	6 081	705 827	120	11
9	Philip	Maksimov	Sergeevic	2003-10-26	7 138	887 861	48	34
10	Alina	Ivanova	Ivanovna	2004-04-12	5 353	211 597	117	21

- Таблица “Subject”:

id	subject_name
1	Математика
2	Русский язык
3	Физика
4	Информатика
5	История
6	География
7	Биология
8	Английский язык
9	Химия
10	Литература

- Таблица “Classroom”:

123 id	123 classroom_number	123 subject_id	123 school_id
1	206	5	1
2	395	4	1
3	389	8	1
4	338	8	1
5	369	10	1
6	371	10	1
7	133	2	1
8	379	9	1
9	350	7	1
10	387	6	1
11	109	8	1
12	226	1	1
13	315	4	1
14	253	8	1
15	300	8	1
16	343	2	1
17	219	7	1
18	286	7	1
19	352	8	1
20	375	5	1
21	371	3	2
22	214	8	2
23	111	4	2

- Таблица “Teacher”:

123 id	ABC first_name	ABC surname	ABC patronymic	123 school_id	123 subject_id
1	Marya	Lubimova	Alexandrovna	1	1
2	Olga	Pimenova	Sergeevna	1	3
3	Alla	Trofimova	Petrovna	1	10
4	Vasily	Panin	Petrovich	1	3
5	Anna	Trofimova	Sergeevna	1	5
6	Marya	Pimenova	Petrovna	1	5
7	Anton	Sidorov	Denisovich	1	5
8	Anastasia	Maltseva	Petrovna	1	3
9	Ivan	Voronin	Sergeevic	1	10
10	Anton	Petrov	Ivanovich	1	2

- Таблица “AnnualExamData”:

id	exam_date	tasks	amount_tasks	passing_primary_score	passing_secondary_score	subject_id
8	2020-06-11 10:00:00.000	{ "1": 4, "2": 8, "3": 10, "4": 14, "5": 10 }	32	10	33	3
9	2021-06-16 10:00:00.000	{ "1": 4, "2": 8, "3": 10, "4": 14, "5": 10 }	32	13	39	3
10	2019-06-20 10:00:00.000	{ "1": 7, "2": 14, "3": 20, "4": 27, "5": 10 }	27	5	34	4
11	2020-06-30 10:00:00.000	{ "1": 7, "2": 14, "3": 20, "4": 27, "5": 10 }	27	5	34	4
12	2021-06-01 10:00:00.000	{ "1": 7, "2": 14, "3": 20, "4": 27, "5": 10 }	27	8	44	4
13	2019-06-04 10:00:00.000	{ "1": 4, "2": 8, "3": 11, "4": 15, "5": 10 }	25	8	29	5
14	2020-06-19 10:00:00.000	{ "1": 4, "2": 8, "3": 11, "4": 15, "5": 10 }	25	8	29	5
15	2021-06-11 10:00:00.000	{ "1": 4, "2": 8, "3": 11, "4": 15, "5": 10 }	25	11	35	5
16	2019-06-03 10:00:00.000	{ "1": 4, "2": 7, "3": 11, "4": 14, "5": 10 }	34	10	34	6
17	2020-06-06 10:00:00.000	{ "1": 4, "2": 7, "3": 11, "4": 14, "5": 10 }	34	9	31	6

- Таблица “ExamResult”:

id	date_of_result	tasks	sum_primary_points	sum_secondary_points	student_id	annual_exam_id
1	2021-06-07 21:28:00.000	{ "1": 1, "2": 1, "3": 1, "4": 1, "5": 10 }	26	77	1	12
2	2021-06-19 04:01:00.000	{ "1": 1, "2": 1, "3": 1, "4": 2, "5": 10 }	51	84	1	21
3	2020-07-28 09:50:00.000	{ "1": 1, "2": 1, "3": 1, "4": 1, "5": 10 }	27	96	2	2
4	2020-06-27 01:56:00.000	{ "1": 1, "2": 1, "3": 1, "4": 1, "5": 10 }	52	79	2	26
5	2020-07-03 00:01:00.000	{ "1": 1, "2": 0, "3": 0, "4": 1, "5": 10 }	50	73	2	29
6	2019-06-22 18:07:00.000	{ "1": 1, "2": 1, "3": 1, "4": 1, "5": 10 }	48	78	3	4
7	2019-07-22 04:40:00.000	{ "1": 1, "2": 1, "3": 2, "4": 2, "5": 10 }	44	87	3	16
8	2019-07-13 21:46:00.000	{ "1": 1, "2": 0, "3": 0, "4": 0, "5": 10 }	46	73	3	25
9	2019-06-27 16:24:00.000	{ "1": 1, "2": 1, "3": 0, "4": 1, "5": 10 }	51	77	3	28
10	2019-06-28 20:52:00.000	{ "1": 1, "2": 1, "3": 1, "4": 1, "5": 10 }	49	94	4	7

- Таблица “Timetable”:

id	desk_number	student_id	annual_exam_id	school_id	teacher_id	classroom_id
1	3	1	21	2	40	32
2	8	1	12	24	581	471
3	16	2	2	4	84	69
4	15	2	26	14	328	266
5	2	2	29	4	76	75
6	8	3	28	10	247	182
7	1	3	16	14	348	270
8	17	3	25	14	329	273
9	2	3	4	10	226	183
10	15	4	25	12	291	240
11	16	4	7	12	276	220

5. Выполнение запросов

В этом разделе приведены различные запросы к реализованной базе данных — их краткие описания, непосредственно запрос на языке SQL и результат выполнения.

1. Вывести количество учеников по городам, упорядочить по убыванию количества:

```
SELECT a.city_name,  
       COUNT(s.first_name) AS count_students  
FROM student AS s  
       JOIN address AS a ON s.address_id = a.id  
GROUP BY a.city_name  
ORDER BY count_students DESC;
```

	city_name	count_students
1	Новосибирск	31
2	Нижний Новгород	27
3	Челябинск	25
4	Екатеринбург	25
5	Москва	24
6	Самара	24
7	Санкт-Петербург	23
8	Казань	21

2. Вывести вывести первые 5 городов, где работает наибольшее количество учителей:

```
SELECT a.city_name, COUNT(t.first_name) AS count_of_teachers  
FROM teacher AS t  
       JOIN school s ON t.school_id = s.id  
       JOIN address a on s.address_id = a.id  
GROUP BY a.city_name  
ORDER BY count_of_teachers DESC  
LIMIT 5;
```

	city_name	count_of_teachers
1	Новосибирск	200
2	Москва	175
3	Санкт-Петербург	150
4	Челябинск	125
5	Екатеринбург	125

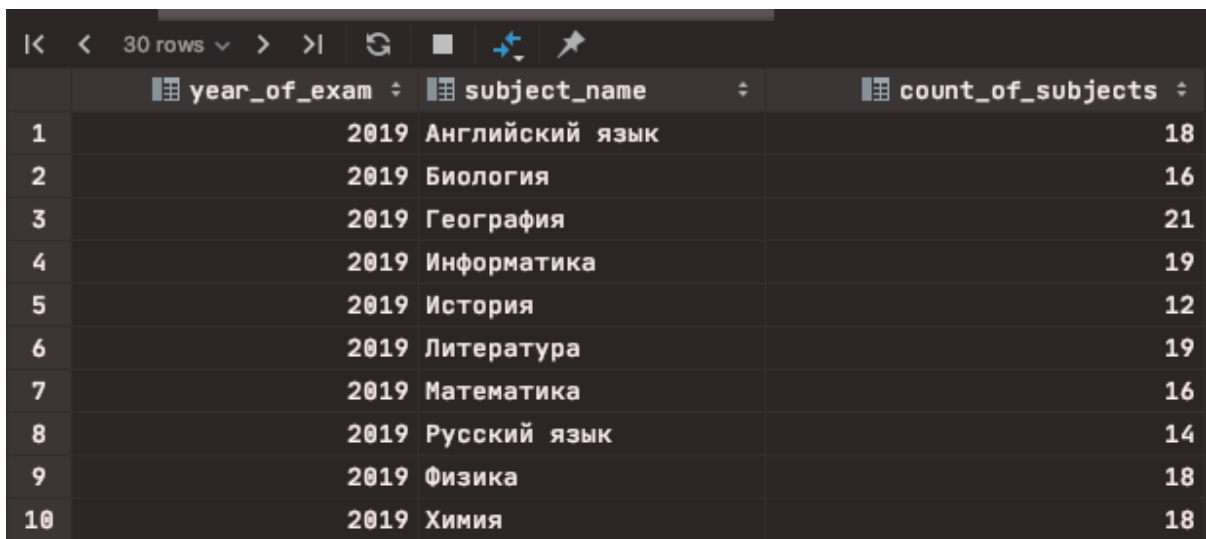
3. Вывести количество аудиторий по каждому предмету в каждом городе:

```
SELECT a.city_name,
       s.subject_name,
       COUNT(c.classroom_number) AS count_of_classrooms
FROM classroom AS c
      JOIN school sch ON c.school_id = sch.id
      JOIN address a ON sch.address_id = a.id
      JOIN subject s ON c.subject_id = s.id
GROUP BY city_name, subject_name
ORDER BY city_name, subject_name;
```

	city_name	subject_name	count_of_classrooms
1	Екатеринбург	Английский язык	15
2	Екатеринбург	Биология	11
3	Екатеринбург	География	6
4	Екатеринбург	Информатика	13
5	Екатеринбург	История	13
6	Екатеринбург	Литература	6
7	Екатеринбург	Математика	7
8	Екатеринбург	Русский язык	14
9	Екатеринбург	Физика	8
10	Екатеринбург	Химия	7
11	Казань	Английский язык	6
12	Казань	Биология	4
13	Казань	География	6
14	Казань	Информатика	9

4. Вывести по каждому предмету за каждый год количество раз, сколько этот предмет сдавали:

```
SELECT date_part('year', aed.exam_date) AS year_of_exam,
       s.subject_name,
       COUNT(s.subject_name)           AS count_of_subjects
FROM timetable AS t
     JOIN annuaalexamdata aed ON t.annual_exam_id = aed.id
     JOIN subject s ON aed.subject_id = s.id
GROUP BY year_of_exam, subject_name
ORDER BY year_of_exam, subject_name;
```



The screenshot shows a database interface with a table containing 10 rows of data. The table has three columns: 'year_of_exam', 'subject_name', and 'count_of_subjects'. The data is sorted by year and subject name. The subjects listed are: Английский язык, Биология, География, Информатика, История, Литература, Математика, Русский язык, Физика, and Химия.

	year_of_exam	subject_name	count_of_subjects
1	2019	Английский язык	18
2	2019	Биология	16
3	2019	География	21
4	2019	Информатика	19
5	2019	История	12
6	2019	Литература	19
7	2019	Математика	16
8	2019	Русский язык	14
9	2019	Физика	18
10	2019	Химия	18

5. Вывести максимальные баллы в стобальной шкале по каждому экзамену в каждом городе:

```
SELECT DATE_PART('year', aed.exam_date) AS year_of_exam,
       a.city_name,
       s.subject_name,
       MAX(e.sum_secondary_points)       AS max_secondary_points
FROM examresult AS e
     JOIN annuaalexamdata AS aed ON e.annual_exam_id = aed.id
     JOIN student st ON e.student_id = st.id
     JOIN address a ON st.address_id = a.id
     JOIN subject AS s ON aed.subject_id = s.id
GROUP BY year_of_exam, city_name, subject_name
ORDER BY year_of_exam, city_name, subject_name;
```

	year_of_exam	city_name	subject_name	max_secondary_points
1	2019	Екатеринбург	Английский язык	86
2	2019	Екатеринбург	Биология	72
3	2019	Екатеринбург	География	92
4	2019	Екатеринбург	Информатика	84
5	2019	Екатеринбург	История	77
6	2019	Екатеринбург	Литература	77
7	2019	Екатеринбург	Математика	99
8	2019	Екатеринбург	Русский язык	78
9	2019	Екатеринбург	Физика	61
10	2019	Екатеринбург	Химия	89
11	2019	Казань	Английский язык	89
12	2019	Казань	Биология	72
13	2019	Казань	География	96

6. Вывести суммарное количество баллов каждого ученика по всем его предметам и количество предметов, которые он сдавал, упорядочить по количеству предметов, а потом по количеству баллов:

```
SELECT st.first_name,
       st.surname,
       st.Patronymic,
       st.passport_series,
       st.passport_number,
       COUNT(*)           AS count_of_subjects,
       SUM(e.sum_secondary_points) AS sum_of_points
FROM annuaalexamdata a
     JOIN examresult e ON a.id = e.annual_exam_id
     JOIN student st ON e.student_id = st.id
GROUP BY st.first_name, st.surname, st.Patronymic,
         st.passport_series, st.passport_number
ORDER BY count_of_subjects DESC, sum_of_points DESC;
```

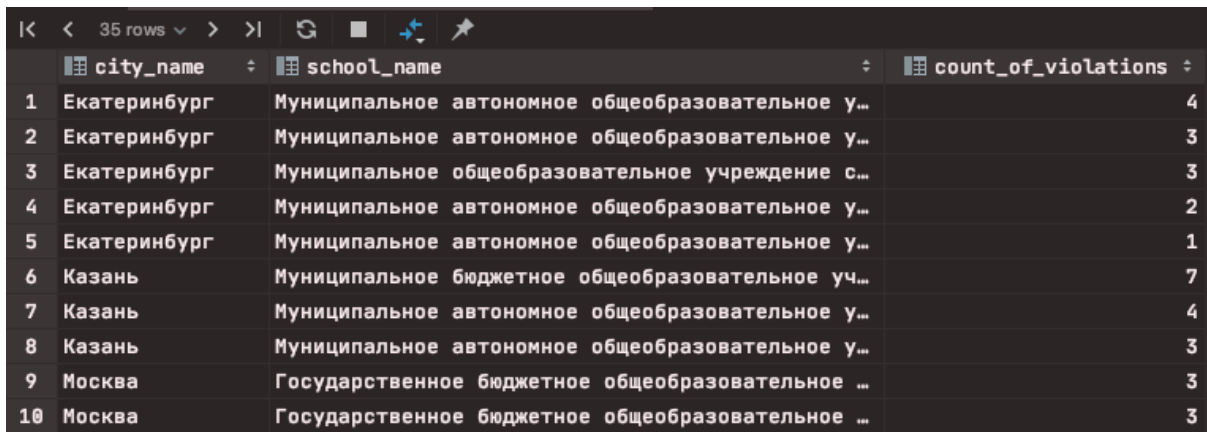
	first_name	surname	patronymic	passport_series	passport_number	count_of_s...	sum_of_points
1	Peter	Voronin	Petrovich	7920	977135	4	362
2	Olga	Maltseva	Petrovna	3029	944753	4	360
3	Ser	Panin	Alexandrovich	6481	587167	4	359
4	Ser	Lubimov	Denisovich	3523	269604	4	359
5	Anastasia	Pimenova	Sergeevna	8119	104653	4	354
6	Philip	Panin	Alexandrovich	2726	645382	4	348
7	Ilia	Lubimov	Petrovich	5446	632348	4	348
8	Ser	Voronin	Olegovich	2447	840776	4	347
9	Alina	Ivanova	Ivanovna	5353	211597	4	346
10	Marya	Maltseva	Sergeevna	4071	638045	4	345

7. Вывести статистику по школам-нарушителям, которые неправильно проводят экзамены. Нарушение, если:

ЛИБО сдаваемый предмет = предмет, который ведет наблюдатель,

ЛИБО сдаваемый предмет = предмет, которому посвящена аудитория:

```
SELECT a2.city_name, s2.school_name, count(*) AS
count_of_violations
FROM timetable t
      JOIN annualexamdata a ON t.annual_exam_id = a.id
      JOIN teacher t2 ON t.teacher_id = t2.id
      JOIN classroom c ON t.classroom_id = c.id
      JOIN subject s ON t2.subject_id = s.id AND a.subject_id =
s.id
      OR a.subject_id = s.id AND c.subject_id = s.id
      JOIN school s2 ON c.school_id = s2.id
      JOIN address a2 ON s2.address_id = a2.id
GROUP BY a2.city_name, s2.school_name
ORDER BY a2.city_name, count_of_violations DESC;
```



	city_name	school_name	count_of_violations
1	Екатеринбург	Муниципальное автономное общеобразовательное у...	4
2	Екатеринбург	Муниципальное автономное общеобразовательное у...	3
3	Екатеринбург	Муниципальное общеобразовательное учреждение с...	3
4	Екатеринбург	Муниципальное автономное общеобразовательное у...	2
5	Екатеринбург	Муниципальное автономное общеобразовательное у...	1
6	Казань	Муниципальное бюджетное общеобразовательное уч...	7
7	Казань	Муниципальное автономное общеобразовательное у...	4
8	Казань	Муниципальное автономное общеобразовательное у...	3
9	Москва	Государственное бюджетное общеобразовательное ...	3
10	Москва	Государственное бюджетное общеобразовательное ...	3

8. Вывести средний балл по стобалльной шкале по каждому экзамену в каждом городе:

```
WITH ready_statistics AS (
  SELECT a2.city_name, s.subject_name,
  ROUND(AVG(e.sum_secondary_points), 2) AS average_secondary_points
  FROM examresult e
        JOIN student st ON e.student_id = st.id
        JOIN address a2 ON st.address_id = a2.id
```

```

        JOIN annuaalexamdata a ON e.annual_exam_id = a.id
        JOIN subject s ON a.subject_id = s.id
    GROUP BY a2.city_name, s.subject_name
)
SELECT city_name,
       subject_name,
       average_secondary_points,
       ROW_NUMBER() OVER (PARTITION BY subject_name ORDER BY
average_secondary_points DESC) AS rating_of_cities
FROM ready_statistics;

```

	city_name	subject_name	average_secondary_points	rating_of_cities
1	Казань	Английский язык	82.6	1
2	Москва	Английский язык	81.71	2
3	Новосибирск	Английский язык	81.33	3
4	Екатеринбург	Английский язык	81.25	4
5	Нижний Новгород	Английский язык	80.78	5
6	Самара	Английский язык	80.44	6
7	Санкт-Петербург	Английский язык	79.17	7
8	Челябинск	Английский язык	77.29	8
9	Новосибирск	Биология	83	1
10	Санкт-Петербург	Биология	81.33	2

9. По каждому году отобразить топ 3 самых популярных у учеников экзаменов и вывести динамику количеств выборов предметов:

```

WITH subjects_annually AS (
    SELECT DATE_PART('year', a.exam_date) AS year_of_exam,
           s.subject_name,
           COUNT(*) AS count_of_choices
    FROM timetable t
        JOIN annuaalexamdata a ON t.annual_exam_id = a.id
        JOIN subject s ON a.subject_id = s.id
    GROUP BY year_of_exam, s.subject_name
    ORDER BY year_of_exam, count_of_choices DESC
),
ranked_subjects AS (
    SELECT subjects_annually.*,
           ROW_NUMBER() OVER count_window
AS top_subjects,
           LAG(count_of_choices) OVER count_window -
count_of_choices AS differences
    FROM subjects_annually
    WINDOW count_window AS (PARTITION BY
year_of_exam ORDER BY count_of_choices DESC)

```



```

)
SELECT year_of_exam, subject_name, count_of_choices, differences
FROM ranked_subjects
WHERE top_subjects <= 3;

```

	year_of_exam	subject_name	count_of_choices	differences
1	2019	География	21	<null>
2	2019	Информатика	19	2
3	2019	Литература	19	0
4	2020	Математика	30	<null>
5	2020	Информатика	25	5
6	2020	Химия	24	1
7	2021	Английский язык	31	<null>
8	2021	Русский язык	27	4
9	2021	Химия	26	1

10. По каждой школе вывести разницу между количеством учителей в школе и сколько раз учителя были востребованы, отобразить подробную инфу про школы, где востребованность выше или равна половине числа учителей в школе:

```

WITH count_of_teachers_per_school AS (
    SELECT s.id,
           s.school_name,
           COUNT(*) AS count_of_teachers_per_school
    FROM school s
         JOIN teacher t ON s.id = t.school_id
    GROUP BY s.id, s.school_name
),
count_of_exams_per_school AS (
    SELECT DATE_PART('year', a.exam_date) as year_of_exam,
           s.id,
           s.school_name,
           COUNT(*) AS count_of_exams_per_school
    FROM timetable t
         JOIN annuaalexamdata a ON t.annual_exam_id = a.id
         JOIN teacher t2 ON t.teacher_id = t2.id
         JOIN school s ON t.school_id = s.id
    GROUP BY year_of_exam, s.id, s.school_name
),
differences AS (
    SELECT ceps.year_of_exam,

```

```

        ctps.id,
        count_of_exams_per_school -
ROUND(count_of_teachers_per_school / 2) AS difference
    FROM count_of_teachers_per_school ctps
        JOIN count_of_exams_per_school ceps ON ctps.id =
ceps.id
    )
SELECT d.year_of_exam,
       s.school_name,
       a.city_name,
       a.street,
       a.house,
       d.difference
FROM differences d
    JOIN school s ON s.id = d.id
    JOIN address a ON s.address_id = a.id
WHERE d.difference >= 0
ORDER BY d.difference DESC;

```

	year_of_exam	school_name	city_name	street	house	difference
1	2021	Муниципальное автономное общеобразовательное уч...	Нижний Новгород	Центральная	30	7
2	2020	Муниципальное бюджетное общеобразовательное учр...	Нижний Новгород	Заречная	39	5
3	2020	Муниципальное автономное общеобразовательное уч...	Нижний Новгород	Центральная	30	2
4	2019	Муниципальное автономное общеобразовательное уч...	Санкт-Петербург	Ленина	52	1
5	2021	Муниципальное бюджетное общеобразовательное учр...	Самара	Зеленая	69	0

11. Посмотреть ежегодную динамику результатов по предметам,
вывести года и предметы, когда средний балл уменьшился:

```

WITH annual_avg_points_info AS (
    SELECT DATE_PART('year', a.exam_date) AS year_of_exam,
           s.subject_name,
           ROUND(AVG(e.sum_secondary_points), 2) AS
avg_annual_points
    FROM examresult e
        JOIN annuaalexamdata a ON e.annual_exam_id = a.id
        JOIN subject s ON a.subject_id = s.id
    GROUP BY year_of_exam, subject_name
    ORDER BY subject_name, year_of_exam
),
differences_annual_avg_points AS (
    SELECT annual_avg_points_info.*,
           LEAD(avg_annual_points) OVER subject_window -
avg_annual_points AS annual_differences

```

```

        FROM annual_avg_points_info
        WINDOW subject_window AS (PARTITION BY subject_name
ORDER BY year_of_exam)
    )
SELECT year_of_exam + 1 as year_of_exam, subject_name
from differences_annual_avg_points
WHERE annual_differences < 0
ORDER BY year_of_exam, subject_name;

```

	year_of_exam	subject_name	annual_differences
1	2020	Биология	-0.75
2	2020	География	-0.2
3	2020	История	-3.14
4	2020	Математика	-0.18
5	2020	Русский язык	-2.51
6	2021	Английский язык	-1.89
7	2021	Биология	-3.25
8	2021	География	-4.16
9	2021	Информатика	-2.4
10	2021	Литература	-5.24
11	2021	Физика	-0.22
12	2021	Химия	-0.82

12. Взять информацию по стобалльникам, где было нарушено
правило проведения экзамена:

```

WITH exam_result_info AS (
    SELECT t.id,
           a.exam_date,
           sb.subject_name,
           e.sum_secondary_points,
           s.first_name,
           s.surname,
           a2.city_name
    FROM examresult e
        JOIN annuaalexamdata a ON e.annual_exam_id = a.id
        JOIN subject sb ON a.subject_id = sb.id
        JOIN student s ON e.student_id = s.id
        JOIN address a2 ON s.address_id = a2.id
        JOIN timetable t ON s.id = t.student_id AND
t.annual_exam_id = a.id

```

```

WHERE sum_secondary_points = 100
ORDER BY a.exam_date
),
violations AS (
    SELECT t.id,
           a.exam_date,
           s.subject_name,
           e.sum_secondary_points,
           s2.first_name,
           s2.surname,
           a2.city_name
    FROM timetable t
        JOIN annuaalexamdata a ON t.annual_exam_id = a.id
        JOIN teacher t2 ON t.teacher_id = t2.id
        JOIN classroom c ON t.classroom_id = c.id
        JOIN subject s ON t2.subject_id = s.id AND
a.subject_id = s.id
        OR a.subject_id = s.id AND c.subject_id = s.id
        JOIN student s2 ON t.student_id = s2.id
        JOIN address a2 ON s2.address_id = a2.id
        JOIN examresult e ON a.id = e.annual_exam_id AND
e.student_id = t.student_id
    ORDER BY a.exam_date
)
SELECT *
FROM exam_result_info
INTERSECT
SELECT *
FROM violations;

```

First Page	exam_date	subject_name	sum_secondary_points	first_name	surname	city_name
1	91 2021-06-01 10:00:00.000000	Информатика	100	Marya	Maltseva	Самара
2	522 2021-06-11 10:00:00.000000	История	100	Alex	Panin	Новосибирск
3	590 2020-06-30 10:00:00.000000	Информатика	100	Anastasia	Trofimova	Новосибирск
4	225 2020-06-29 10:00:00.000000	Математика	100	Anton	Kireev	Екатеринбург
5	314 2021-06-01 10:00:00.000000	Информатика	100	Peter	Sidorov	Самара
6	286 2020-06-29 10:00:00.000000	Математика	100	Anna	Anisimova	Санкт-Петербург
7	242 2020-06-29 10:00:00.000000	Математика	100	Ivan	Petrov	Москва
8	479 2020-06-29 10:00:00.000000	Математика	100	Darya	Petrova	Екатеринбург

13. Вывести всех стобалльников, которые писали в одной аудитории с учителями, которые работали больше, чем на одном экзамене в один год:

```
WITH all_successful_student AS (  
    SELECT s.first_name,  
           s.surname,  
           s.patronymic,  
           s2.subject_name,  
           e.sum_secondary_points,  
           a2.city_name,  
           DATE_PART('year', a.exam_date) AS year_of_exam,  
           t.teacher_id  
    FROM examresult e  
         INNER JOIN student s ON e.student_id = s.id  
         INNER JOIN annualedata a ON e.annual_exam_id = a.id  
         INNER JOIN subject s2 ON s2.id = a.subject_id  
         INNER JOIN timetable t ON a.id = t.annual_exam_id AND  
t.student_id = s.id  
         INNER JOIN address a2 ON s.address_id = a2.id  
    WHERE e.sum_secondary_points = 100  
)  
,  
all_teachers_required AS (  
    select DATE_PART('year', a.exam_date) AS year_of_exam,  
           t.id,  
           t.first_name,  
           t.surname,  
           t.patronymic,  
           count(*) as  
count_of_annual_observations  
    FROM timetable t2  
         INNER JOIN teacher t ON t2.teacher_id = t.id  
         INNER JOIN annualedata a ON t2.annual_exam_id  
= a.id  
    GROUP BY t.id, DATE_PART('year', a.exam_date),  
t.first_name, t.surname, t.patronymic  
    HAVING count(*) > 2  
)  
SELECT ass.first_name as student_name,  
       ass.surname as student_surname,
```

```

    ass.patronymic          as student_patronymic,
    subject_name,
    sum_secondary_points,
    city_name,
    ass.year_of_exam,
    atr.first_name,
    atr.surname,
    atr.patronymic,
    count_of_annual_observations as annual_observations
FROM all_successful_student ass
    INNER JOIN all_teachers_required atr
        ON ass.teacher_id = atr.id AND atr.year_of_exam
= ass.year_of_exam;

```

	stude...	student...	subje...	sum...	city_name	year...	f...	...	pa...	ann...
1	'anin	Alexandrovich	История	100	Новосибирск	2021	Anna	Maltseva	Sergeevna	3
2	'etrova	Ivanovna	Математика	100	Казань	2020	Vasily	Panin	Olegovich	3
3	'imenova	Denisovna	Информатика	100	Санкт-Петербург	2019	Ser	Maksimov	Denisovich	3

14. Вывести количество школ по годам, где есть хотя бы один плохо написавший его человек. Плохо:

(2 предмета (сумма баллов < 130),

3 предмета (сумма баллов < 220),

4 предмета (сумма баллов < 280)):

```

WITH all_required_students AS (
    SELECT st.id,
           st.school_id,
           DATE_PART('year', a.exam_date) AS year_of_exam,
           COUNT(*) AS count_of_subjects,
           SUM(e.sum_secondary_points) AS sum_of_points
    FROM annualexamdata a
        JOIN examresult e ON a.id = e.annual_exam_id
        JOIN student st ON e.student_id = st.id
    GROUP BY st.id, st.school_id, year_of_exam
    HAVING COUNT(*) = 2 and SUM(e.sum_secondary_points) < 140
        or COUNT(*) = 3 and SUM(e.sum_secondary_points) < 220
        or COUNT(*) = 4 and SUM(e.sum_secondary_points) < 320
),
annual_schools_data AS (
    SELECT distinct on (year_of_exam, school_id) year_of_exam

```

```

        FROM all_required_students
    )
SELECT year_of_exam, count(*) as count_of_schools
FROM annual_schools_data
GROUP BY year_of_exam;

```

	year_of_exam	count_of_schools
1	2019	12
2	2020	7
3	2021	15

15. Вывести статистику по возрасту пишущих экзамен учеников:
 разницу между средним баллом, набранным за определенный год
 по определенному предмету:

```

WITH age_of_student AS (
    SELECT distinct on (st.passport_series, st.passport_number)
    st.id,
    (CASE
        WHEN DATE_PART('year', AGE(a.exam_date, st.birthday)) > 17
        THEN 'adult'
        ELSE 'child'
    END) AS adult_child
    FROM timetable t
        INNER JOIN annualedamdata a on t.annual_exam_id = a.id
        INNER JOIN student st on t.student_id = st.id
),
required_annual_data AS (
    SELECT DATE_PART('year', a.exam_date) AS
year_of_exam,
        s.subject_name,
        aos.adult_child,
        ROUND(AVG(e.sum_secondary_points), 2) AS
average_points
    FROM examresult e
        INNER JOIN annualedamdata a on e.annual_exam_id =
a.id
        INNER JOIN subject s on a.subject_id = s.id
        INNER JOIN age_of_student aos ON aos.id =
e.student_id

```

```

        GROUP BY year_of_exam, s.subject_name, aos.adult_child
        ORDER BY year_of_exam, subject_name, adult_child
    )
SELECT required_annual_data.*,
       LEAD(average_points) OVER average_window - average_points AS
differences_between_childs_adults
FROM required_annual_data
WINDOW average_window AS (PARTITION BY year_of_exam,
subject_name);

```

year_of_exam	subject_name	adult_child	average_points	differences_between_childs_adults
2019	Английский язык	adult	71.25	6.82
2019	Английский язык	child	78.07	<null>
2019	Биология	adult	77.67	3.87
2019	Биология	child	81.54	<null>
2019	География	adult	79.67	-2.94
2019	География	child	76.73	<null>
2019	Информатика	adult	82.67	4.64
2019	Информатика	child	87.31	<null>
2019	История	adult	83.2	-5.06
2019	История	child	78.14	<null>
2019	Литература	adult	67.13	2.96
2019	Литература	child	70.09	<null>