Задача 1

```
CREATE TABLE Status (
   id SERIAL PRIMARY KEY,
   name VARCHAR(255) NOT NULL
);

CREATE TABLE Destination (
   id SERIAL PRIMARY KEY,
   name VARCHAR(100) NOT NULL,
   id_status INTEGER NOT NULL,
         foreign key (id_status) REFERENCES Status (id)
);

CREATE TABLE Tickets (
   id SERIAL PRIMARY KEY,
   id_destination INTEGER REFERENCES Destination(id),
   lowest_price FLOAT NOT NULL,
   highest_price FLOAT NOT NULL
         );


INSERT INTO Status (name) Values
('Без визы'),
('С визой'),
('В ожидании визы');

SELECT *from Status

INSERT INTO Destination (name, id_status) Values
('Ереван', 1),
('Москва', 2),
('Ставрополь', 2),
('Милан', 1),
('Сан_Франциско', 1);

SELECT *from Destination

INSERT INTO Tickets (id_destination, lowest_price, highest_price) Values
(1, 100.00, 500.00),
(2, 200.00, 400.00),
(3, 300.00, 600.00),
(4, 400.00, 800.00),
(5, 500.00, 1000.00);

SELECT *from Tickets
```

--Уникальные названия маршрутов (destination.name), для которых существуют билеты (есть запись в tickets). Вывести только названия.

```
SELECT DISTINCT d.name
FROM Destination d
INNER JOIN Tickets t ON d.id = t.id_destination;
```

--Дополните предыдущий запрос: ограничьте маршруты статусом «Без визы»

```
SELECT DISTINCT d.name
FROM Tickets t
INNER JOIN  Destination d ON d.id = t.id_destination
WHERE d.id_status = (Select id from status
where name = 'Без визы');
```

--Найдите маршруты, максимальная цена которых выше общей средней. Общая средняя находится как среднее значение lowest_price и highest_price. Вывести названия и высшую цену.

```
Select d.name as destination_name, t.highest_price
From Tickets t Inner Join Destination d on t.id_destination=d.id
Where t.highest_price >
(Select AVG (lowest_price + highest_price)/2 from Tickets t);
```

Задача 2

```
CREATE TABLE Users (
	id_user SERIAL PRIMARY KEY,
	user_name VARCHAR(100) NOT NULL,
	user_surname VARCHAR(100) NOT NULL,
	user_weigth Decimal (10, 2) NOT NULL,
	ageSERIAL INTEGER NOT NULL);


INSERT INTO Users (id_user, user_name, user_surname, user_weigth, ageSERIAL)
Values
(1, 'Anna', 'Ivanova', 56, 18),
(2, 'Igor', 'Bulik', 75, 45),
(3, 'Max', 'Nikolsky', 76, 16),
(4, 'Kate', 'Svet', 66, 30);

SELECT *from Users


CREATE TABLE visits (
	id_visit SERIAL PRIMARY KEY,
	id_user INTEGER NOT NULL,
	hours_spent Decimal (2, 1) NOT NULL,
	class_name VARCHAR(100) NOT NULL,
	date DATE NOT NULL);

INSERT INTO visits (id_visit, id_user, hours_spent, class_name, date)
Values
	(1, 1, 1, 'Zumba', '2023-06-30'),
	(2, 3, 2, 'Swimming pool', '2023-07-04'),
	(3, 5, 1, 'Flex', '2023-07-09'),
	(4, 1, 3, 'Flex', '2023-07-15'),
	(5, 5, 2, 'Step', '2023-07-20'),
	(6, 2, 1.5, 'Football', '2023-07-22');

SELECT *from visits


ALTER TABLE Users
RENAME COLUMN ageSERIAL TO age;

SELECT *from Users
```

--Список уникальных классов. Вывести только названия.

```
Select Distinct class_name
From visits
```

--Количество часов, проведенных на занятиях, для каждого пользователя. Вывести фамилию, имя и количество часов.

```
Select  users.user_surname,
                    users.user_name,
SUM  (hours_spent) AS total_hours
From visits Inner Join users on
users.id_user = visits.id_user
Group by users.user_surname,
                     users.user_name;
```

--Средний возраст пользователей, посещающих класс Flex.

```
Select AVG (users.age) AS avg_age
From visits Inner Join users on users.id_user=visits.id_user
Where
class_name = 'Flex';
```

Задача 3

```
CREATE TABLE book (
	id_book INT PRIMARY KEY,
	title VARCHAR (100) NOT NULL,
	id_author INTEGER NOT NULL,
	pages INTEGER NOT NULL,
	year_publish DATE NOT NULL);

INSERT INTO book (id_book, title, id_author, pages, year_publish)
VALUES

	(1, 'Поющие в терновнике', 1, 200, '2020-01-01'),
	(2, 'Унесенные ветром', 2, 300, '2019-05-15'),
	(3, 'Доктор Паскаль', 3, 150, '2021-10-20'),
	(4, 'Овод', 4, 450, '2023-07-20'),
	(5, 'Мастер и Маргарита', 5, 550, '2024-12-20'),
	(6, 'Джейн Эйр', 6, 650, '2020-09-12'),
	(7, 'Три мушкетера', 7, 750, '2023-10-20'),
	(8, 'Финансист', 8, 850, '2021-10-24');

SELECT *from book

CREATE TABLE author (
	id_author INT PRIMARY KEY,
	full_name VARCHAR (100) NOT NULL,
	century INTEGER NOT NULL);

INSERT INTO author (id_author, full_name, century)
VALUES

	(1, 'Маккалоу Колин', 12),
	(2, 'Маргарет Митчелл', 20),
	(3, 'Эмиль Золя', 25),
	(4, 'Теодор Драйзер', 35),
	(5, 'Михаил Булгаков', 45),
	(6, 'Шарлотта Бронте', 16),
	(7, 'Александр Дюма', 22),
	(8, 'Теодор Драйзер', 24);


SELECT *from author
```

--Уникальные названия всех книг, опубликованных после 1990 года. Вывести только названия.

```
SELECT DISTINCT title
FROM book
WHERE year_publish > '1991-01-01';
```

--Для каждого автора найти сумму напечатанных страниц. Вывести полное имя автора и сумму страниц.

```
SELECT author.full_name, SUM(book.pages) AS total_pages
FROM book
Inner JOIN author ON author.id_author = book.id_author
GROUP BY author.full_name;
```

--Подсчитать количество книг авторов каждого века. Вывести век и количество книг.

```
SELECT author.century, COUNT(book.id_book) AS book_count
FROM author
Inner JOIN book ON author.id_author = book.id_author
GROUP BY author.century;
```