**Испытание 3: «Сердце Джунглей»**

**Задание №1**

--Задача 1 создание и заполнение таблиц

CREATE TABLE Status (

id SERIAL primary KEY,

name VARCHAR (255) not NULL

);

CREATE TABLE Destination (

id SERIAL PRIMARY KEY,

name VARCHAR (255) NOT NULL, id\_status INT,

FOREIGN KEY (id\_status) REFERENCES Status (id)

);

CREATE TABLE Tickets (

id SERIAL PRIMARY KEY,

id\_destination INT,

lowest price DECIMAL(10, 2) NOT NULL,

highest price DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (id\_destination) REFERENCES Destination (id)

);

INSERT INTO Status (name)

VALUES

('Без визы'),

('С визой'),

('В ожидании визы');

INSERT INTO Destination (name, id\_status)

VALUES

('Париж', 1),

('Лондон', 2),

('Рим', 1),

('Нью-Йорк', 2),

('Токио', 1);

INSERT INTO Tickets (id\_destination, lowest price, highest\_price)

VALUES

(1, 200.00, 300.00),

(2, 250.00, 350.00),

(3, 180.00, 280.00),

(4, 300.00, 400.00),

(5, 220.00, 320.00);

--Задача 1.1 Уникальные названия маршрутов (destination.name), для которых существуют билеты (есть запись в tickets). Вывести только названия.

SELECT DISTINCT destination.name

FROM tickets

JOIN destination ON destination.id = tickets.id\_destination

--Задача 1.2 Дополните предыдущий запрос: ограничьте маршруты статусом «Без визы». 1 вариант

SELECT DISTINCT destination.name

FROM tickets

JOIN destination ON destination.id = tickets.id\_destination

JOIN status ON status.id = destination.id\_status

WHERE

status. "name" = 'Без визы"

--ИЛИ

SELECT

DISTINCT destination.name

FROM tickets JOIN destination ON destination.id = tickets.id\_destination

WHERE

destination.id\_status = (SELECT id FROM status WHERE name 'Без визы")

--Задача 1.3 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 Destination.name AS destination\_name, Tickets.highest\_price

FROM Tickets

JOIN Destination ON Tickets.id\_destination = Destination.id

WHERE

Tickets.highest\_price > (SELECT AVG (lowest\_price + highest\_price)/ 2 FROM Tickets)

**Задание №2**

CREATE TABLE users (

id\_user serial PRIMARY KEY,

user\_name VARCHAR (255) NOT NULL,

user\_surname VARCHAR (255) NOT NULL,

user\_weigth DECIMAL (10, 2) NOT NULL,

age INT NOT NULL

);

INSERT INTO users (

id\_user, user\_name, user\_surname, user\_weigth, age

)

VALUES

(1, 'Anna', 'Ivanova', 56.00, 18),

(2, 'Igor', 'Bulik', 75.00, 45),

(3, 'Max', 'Nikolsky', 67.00, 16),

(5, 'Kate', 'Svet', 66.00, 30);

CREATE TABLE visits (

id\_visit serial PRIMARY KEY,

id\_user INT NOT NULL,

hours spent DECIMAL (2, 1) NOT NULL,

class\_name VARCHAR (255) NOT NULL,

date DATE NOT NULL

);

INSERT INTO visits (

id\_visit, id\_user, hours\_spent, class\_name, date

)

VALUES

(1, 1, 1.0, 'Zumba', '2023-06-30'),

(2, 3, 2.0, 'Swimming pool', '2023-07-04'),

(3, 5, 1.0, 'Flex', '2023-07-09'),

(4, 1, 3.0, 'Flex', '2023-07-15'),

(5, 5, 2.0, 'Step', '2023-07-20'),

(6, 2, 1.5, 'Football', '2023-07-22');

-- 1.1 Список уникальных классов. Вывести только названия.

SELECT DISTINCT class\_name

FROM visits;

-- 1.2 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 JOIN users ON users.id\_user = visits.id\_user

GROUP BY

users.user\_surname,

users.user\_name;

-- 1.3 Средний возраст пользователей, посещающих класс Flex.

SELECT AVG (users.age) AS avg\_age

FROM visits JOIN users ON users.id\_user = visits.id\_user

WHERE

class\_name = 'Flex';

**Задание №3**

--Создание таблицы

CREATE TABLE book (

id\_book INT PRIMARY KEY,

title VARCHAR(255) NOT NULL,

id\_author INT NOT NULL,

pages INT NOT NULL,

year\_publish DATE NOT NULL

);

CREATE TABLE author (

id\_author INT PRIMARY KEY,

full\_name VARCHAR(255) NOT NULL,

century INT NOT NULL

);

--Заполнение таблицы book

INSERT INTO book (id\_book, title, id\_author, pages, year\_publish)

VALUES

(1, 'Война и мир', 1, 1225, '1869-02-12'),

(2, 'Преступление и наказание', 2, 864, '1866-01-01'),

(3, 'Мастер и Маргарита", 3, 720, '1967-03-03'),

(4, 'Лолита', 4, 368, '1955-09-01'),

(5, '1984', 5, 328, '1949-06-08'),

(6, 'Гарри Поттер и философский камень', 6, 320, '1997-06-26'),

(7, 'Гарри Поттер и тайная комната', 6, 376, '1998-07-02'),

(8, 'Гарри Поттер и узник Азкабана', 6, 432, '1999-07-08'),

(9, 'Гарри Поттер и Кубок огня', 6, 636, '2000-07-08');

--Заполнение таблицы author

INSERT INTO author (id\_author, full\_name, century)

VALUES

(1, 'Лев Николаевич Толстой', 19),

(2, 'Федор Михайлович Достоевский', 19),

(3, 'Михаил Афанасьевич Булгаков', 20),

(4, 'Владимир Владимирович Набоков', 20),

(5, 'Джордж Оруэлл', 20),

(6, 'Джоан Роулинг', 20);

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

SELECT DISTINCT title

FROM book

WHERE year\_publish >= '1991-01-01';

--или

SELECT DISTINCT title

FROM book

WHERE to\_char (year\_publish, 'YYYY') > '1990';

-- 1.2 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 JOIN author ON author.id\_author = book.id\_author

GROUP BY author.full\_name;

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

SELECT author.century, COUNT (book.id\_book) as book\_count

FROM author

JOIN book ON author.id\_author = book.id\_author

GROUP BY author.century;