-- Создание базы данных "Агентство организации праздников"

CREATE DATABASE Агентство\_организации\_праздников;

GO

-- Использование базы данных

USE Агентство\_организации\_праздников;

GO

-- Создание таблицы "Клиенты"

CREATE TABLE Клиенты (

id\_клиента INT PRIMARY KEY IDENTITY(1, 1),

Имя VARCHAR(100) NOT NULL,

Фамилия VARCHAR(100) NOT NULL,

Телефон VARCHAR(20) NOT NULL,

Email VARCHAR(100)

);

GO

-- Создание таблицы "Типы\_праздников"

CREATE TABLE Типы\_праздников (

id\_типа INT PRIMARY KEY IDENTITY(1, 1),

Название VARCHAR(100) NOT NULL,

Описание VARCHAR(MAX)

);

GO

-- Создание таблицы "Праздники"

CREATE TABLE Праздники (

id\_праздника INT PRIMARY KEY IDENTITY(1, 1),

Название VARCHAR(100) NOT NULL,

Дата\_праздника DATE NOT NULL,

id\_типа INT,

FOREIGN KEY (id\_типа) REFERENCES Типы\_праздников(id\_типа)

);

GO

-- Создание таблицы "Организаторы"

CREATE TABLE Организаторы (

id\_организатора INT PRIMARY KEY IDENTITY(1, 1),

Имя VARCHAR(100) NOT NULL,

Фамилия VARCHAR(100) NOT NULL,

Телефон VARCHAR(20) NOT NULL,

Email VARCHAR(100)

);

GO

-- Создание таблицы "Заказы"

CREATE TABLE Заказы (

id\_заказа INT PRIMARY KEY IDENTITY(1, 1),

id\_клиента INT,

id\_праздника INT,

id\_организатора INT,

Дата\_заказа DATE NOT NULL,

FOREIGN KEY (id\_клиента) REFERENCES Клиенты(id\_клиента),

FOREIGN KEY (id\_праздника) REFERENCES Праздники(id\_праздника),

FOREIGN KEY (id\_организатора) REFERENCES Организаторы(id\_организатора)

);

GO

use Агентство\_организации\_праздников

go

-- Заполнение таблицы "Клиенты"

INSERT INTO Клиенты (Имя, Фамилия, Телефон, Email)

VALUES ('Иван', 'Иванов', '1234567890', 'ivan@example.com'),

('Анна', 'Петрова', '9876543210', 'anna@example.com'),

('Михаил', 'Сидоров', '5678901234', 'mikhail@example.com'),

('Елена', 'Смирнова', '5556667777', 'elena@example.com');

-- Заполнение таблицы "Типы\_праздников"

INSERT INTO Типы\_праздников (Название, Описание)

VALUES ('День рождения', 'Празднование дня рождения'),

('Свадьба', 'Организация свадебного торжества'),

('Корпоративное мероприятие', 'Организация корпоративного мероприятия'),

('Юбилей', 'Празднование юбилея');

-- Заполнение таблицы "Праздники"

INSERT INTO Праздники (Название, Дата\_праздника, id\_типа)

VALUES ('День рождения Ивана', '2023-01-10', 1),

('Свадьба Анны и Петра', '2023-05-20', 2),

('Корпоративное мероприятие "Веселая компания"', '2023-09-15', 3),

('Юбилей Михаила', '2023-06-05', 4);

-- Заполнение таблицы "Организаторы"

INSERT INTO Организаторы (Имя, Фамилия, Телефон, Email)

VALUES ('Алексей', 'Смирнов', '9876543210', 'alexey@example.com'),

('Екатерина', 'Иванова', '5678901234', 'ekaterina@example.com'),

('Андрей', 'Петров', '1112223333', 'andrey@example.com'),

('Мария', 'Сидорова', '4445556666', 'maria@example.com');

-- Заполнение таблицы "Заказы"

INSERT INTO Заказы (id\_клиента, id\_праздника, id\_организатора, Дата\_заказа)

VALUES (1, 1, 1, '2023-01-05'),

(2, 2, 2, '2023-04-15'),

(3, 3, 1, '2023-08-20'),

(4, 4, 3, '2023-05-10');

use Агентство\_организации\_праздников

go

CREATE FUNCTION ПолучитьКоличествоЗаказовПоКлиенту(@id\_клиента INT)

RETURNS INT

AS

BEGIN

DECLARE @количество INT;

SELECT @количество = COUNT(\*) FROM Заказы WHERE id\_клиента = @id\_клиента;

RETURN @количество;

END;

go

CREATE FUNCTION ПолучитьИнформациюОПразднике(@id\_праздника INT)

RETURNS TABLE

AS

RETURN

(

SELECT Праздники.Название AS Название\_праздника, Праздники.Дата\_праздника AS Дата\_праздника, Типы\_праздников.Название AS Тип\_праздника

FROM Праздники

INNER JOIN Типы\_праздников ON Праздники.id\_типа = Типы\_праздников.id\_типа

WHERE Праздники.id\_праздника = @id\_праздника

);

go

CREATE FUNCTION РассчитатьСреднийЗаказПоОрганизатору(@id\_организатора INT)

RETURNS DECIMAL(10, 2)

AS

BEGIN

DECLARE @средний\_заказ DECIMAL(10, 2);

SELECT @средний\_заказ = AVG(Заказы.id\_праздника)

FROM Заказы

WHERE Заказы.id\_организатора = @id\_организатора;

RETURN @средний\_заказ;

END;

Go

use Агентство\_организации\_праздников

go

CREATE TRIGGER Удалить\_Связанные\_Заказы

ON Клиенты

AFTER DELETE

AS

BEGIN

DELETE FROM Заказы

WHERE id\_клиента IN (SELECT id\_клиента FROM deleted);

END;

go

-- Создание триггера "Уведомление\_о\_новом\_заказе"

CREATE TRIGGER Уведомление\_о\_новом\_заказе

ON Заказы

AFTER INSERT

AS

BEGIN

-- Получение информации о новом заказе

DECLARE @id\_заказа INT, @id\_клиента INT, @id\_праздника INT, @id\_организатора INT, @Дата\_заказа DATE;

SELECT @id\_заказа = id\_заказа, @id\_клиента = id\_клиента, @id\_праздника = id\_праздника, @id\_организатора = id\_организатора, @Дата\_заказа = Дата\_заказа

FROM inserted;

-- Отправка уведомления о новом заказе

PRINT 'Новый заказ создан:'

PRINT 'ID заказа: ' + CAST(@id\_заказа AS VARCHAR(10))

PRINT 'ID клиента: ' + CAST(@id\_клиента AS VARCHAR(10))

PRINT 'ID праздника: ' + CAST(@id\_праздника AS VARCHAR(10))

PRINT 'ID организатора: ' + CAST(@id\_организатора AS VARCHAR(10))

PRINT 'Дата заказа: ' + CONVERT(VARCHAR(10), @Дата\_заказа, 120)

END;

go

CREATE TRIGGER Проверка\_Доступности\_Дата\_Заказа

ON Заказы

INSTEAD OF INSERT

AS

BEGIN

IF EXISTS (SELECT 1 FROM Заказы WHERE Дата\_заказа IN (SELECT Дата\_заказа FROM inserted))

BEGIN

RAISERROR ('Выбранная дата уже занята. Пожалуйста, выберите другую дату.', 16, 1);

ROLLBACK TRANSACTION;

END;

ELSE

BEGIN

INSERT INTO Заказы (Дата\_заказа, id\_клиента)

SELECT Дата\_заказа, id\_клиента

FROM inserted;

END;

END;

go

use Агентство\_организации\_праздников

go

CREATE PROCEDURE Добавить\_клиента

@Имя NVARCHAR(50),

@Фамилия NVARCHAR(50),

@Телефон NVARCHAR(20),

@Email NVARCHAR(100)

AS

BEGIN

INSERT INTO Клиенты (Имя, Фамилия, Телефон, Email)

VALUES (@Имя, @Фамилия, @Телефон, @Email)

END;

go

CREATE PROCEDURE Создать\_заказ

@id\_клиента INT,

@id\_праздника INT,

@Дата\_заказа DATE

AS

BEGIN

INSERT INTO Заказы (id\_клиента, id\_праздника, Дата\_заказа)

VALUES (@id\_клиента, @id\_праздника, @Дата\_заказа)

END;

go

CREATE PROCEDURE Обновить\_информацию\_о\_празднике

@id\_праздника INT,

@Название NVARCHAR(100),

@id\_типа INT,

@Дата\_праздника DATE

AS

BEGIN

UPDATE Праздники

SET Название = @Название, id\_типа = @id\_типа, Дата\_праздника = @Дата\_праздника

WHERE id\_праздника = @id\_праздника

END;

go

use Агентство\_организации\_праздников

go

CREATE VIEW Все\_клиенты\_и\_заказы AS

SELECT Клиенты.Имя, Клиенты.Фамилия, Заказы.id\_праздника, Праздники.Название AS Название\_праздника

FROM Клиенты

JOIN Заказы ON Клиенты.id\_клиента = Заказы.id\_клиента

JOIN Праздники ON Заказы.id\_праздника = Праздники.id\_праздника;

go

CREATE VIEW Информация\_о\_праздниках AS

SELECT Праздники.Название, Типы\_праздников.Название AS Тип\_праздника, Заказы.Дата\_заказа, Организаторы.Имя AS Имя\_организатора

FROM Праздники

JOIN Типы\_праздников ON Праздники.id\_типа = Типы\_праздников.id\_типа

JOIN Заказы ON Праздники.id\_праздника = Заказы.id\_праздника

JOIN Организаторы ON Заказы.id\_организатора = Организаторы.id\_организатора;

go

CREATE VIEW Свободные\_организаторы AS

SELECT Организаторы.Имя, Организаторы.Фамилия, Организаторы.Телефон, Организаторы.Email

FROM Организаторы

LEFT JOIN Заказы ON Организаторы.id\_организатора = Заказы.id\_организатора

WHERE Заказы.id\_организатора IS NULL;

go

1. **Опишите основные принципы структуризации и нормализации базы данных.**

Основные принципы структуризации и нормализации базы данных включают следующее:

1. Единообразие: База данных должна быть организована по единым стандартам и соглашениям. Это включает использование одинаковых имен для таблиц, полей и других элементов базы данных, а также согласованное применение типов данных и форматов.

2. Минимизация избыточности: Избыточность данных может привести к проблемам согласованности и производительности. При проектировании базы данных следует стремиться к минимизации повторяющихся данных, чтобы каждая информация хранилась только в одном месте.

3. Уникальность идентификаторов: Каждая запись в базе данных должна иметь уникальный идентификатор, который позволяет однозначно идентифицировать эту запись. Обычно это реализуется с помощью первичных ключей, которые гарантируют уникальность значений и обеспечивают быстрый доступ к данным.

4. Нормализация: Нормализация является процессом разделения таблиц на более мелкие и логически связанные единицы для устранения аномалий и избыточности данных. Нормализация позволяет улучшить структуру базы данных, упростить операции вставки, обновления и удаления данных, а также повысить целостность данных.

5. Связи и отношения: База данных должна использовать связи и отношения для установления связей между таблицами и представления связанных данных. Это обеспечивает целостность данных, позволяет эффективно использовать данные из разных таблиц и упрощает выполнение запросов.

6. Независимость данных: Структура базы данных должна быть независима от приложений, которые используют эти данные. Это позволяет изменять и модифицировать приложения, не затрагивая структуру базы данных, и обеспечивает гибкость и масштабируемость системы.

7. Согласованность и целостность данных: База данных должна поддерживать согласованность и целостность данных путем применения ограничений, проверок и правил, которые гарантируют правильность и корректность данных.