1. Базы данных - тест

1) 3. Таблица без полей существовать не может

2) 4. Неоднородная информация (данные разных типов)

3) 2. Значения первичного ключа всегда должны быть уникальными и не могут быть null, значения; 4. Первичный ключ является идентификатором для строки, а внешний ключ используется для связывания таблиц

4) 2. 2НФ; 3. 3НФ

5) 4. Сначала FROM, потом GROUP BY и только потом SELECT

6) 2. Оператор HAVING применяется для фильтрации групп, а WHERE - для фильтрации отдельных строк; 4. WHERE может использоваться для фильтрации по любому полю или выражению, а HAVING - только для фильтрации по выражению в списке выбора или агрегатной функции; 5.HAVING всегда используется после GROUP BY, а WHERE может использоваться до или после GROUP BY

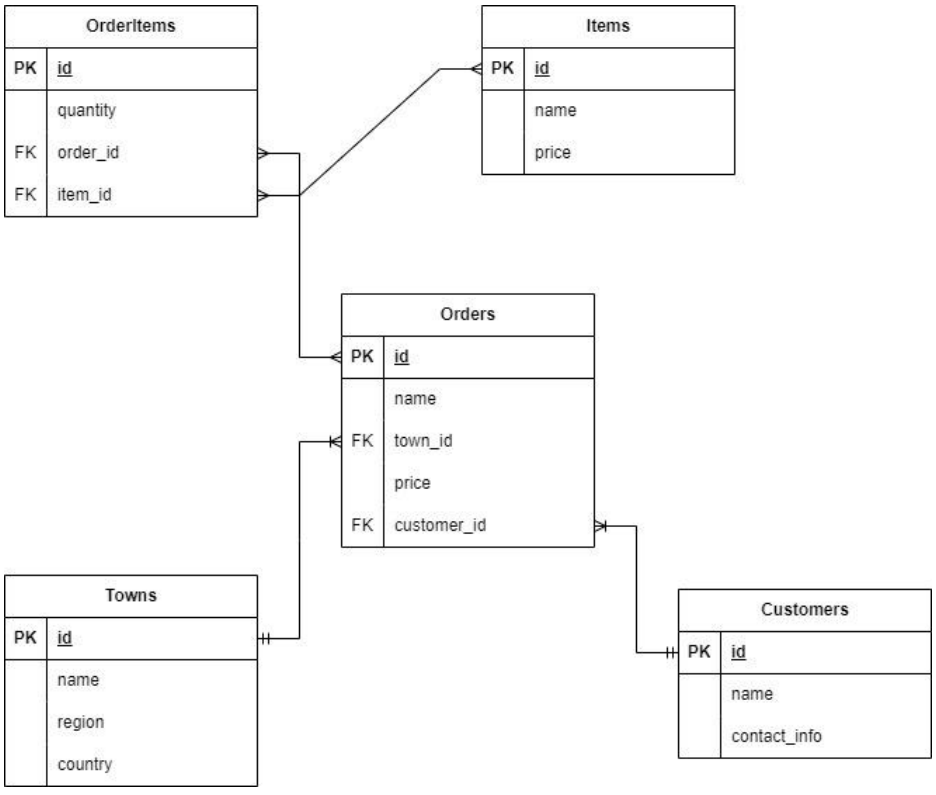
7) 3. Число строк таблицы, указанной во FROM, включая значение NULL

8) 1. SELECT age FROM Animals WHERE Animal LIKE “%fox”

9) 2. DELETE используется для удаления одной или нескольких строк из таблицы, а TRUNCATE используется для удаления всех строк из таблицы; 3. DELETE может использовать условие WHERE, а TRUNCATE всегда удаляет все записи из таблицы

10) 4. 2

2. Базы данных - ER



3. Интеграции

1. Получение списка товаров (витрина)

Запрос:

- Метод: GET
- URL: /api/products
- Параметры:
  - category (опционально): Категория товаров
  - page (опционально): Номер страницы
  - limit (опционально): Количество товаров на странице

Пример запроса:

```
GET /api/products?category=electronics&page=1&limit=10
```

Ответ:

```
{
  "products": [
    {
      "id": 1,
      "name": "Smartphone XYZ",
      "description": "A high-end smartphone with advanced features.",
      "price": 599.99,
      "imageUrl": "https://example.com/images/smartphone-xyz.jpg"
    },
    {
      "id": 2,
      "name": "Laptop ABC",
      "description": "A powerful laptop for gaming and work.",
      "price": 1299.99,
      "imageUrl": "https://example.com/images/laptop-abc.jpg"
    }
  ],
  "totalPages": 5,
  "currentPage": 1
}
```

2. Получение детального описания товара

Запрос:

- Метод: GET
- URL: /api/products/{id}

Пример запроса:

```
GET /api/products/1
```

Ответ:

```
{
  "id": 1,
  "name": "Smartphone XYZ",
  "description": "A high-end smartphone with advanced features.",
  "price": 599.99,
  "imageUrl": "https://example.com/images/smartphone-xyz.jpg",
  "details": "This smartphone has a 6.5-inch OLED display, 12GB RAM, and a 5000mAh battery.",
```

```
  "reviews": [
    {
      "user": "John Doe",
      "rating": 5,
      "comment": "Great phone, highly recommend!"
    },
    {
      "user": "Jane Smith",
      "rating": 4,
      "comment": "Good phone, but a bit expensive."
    }
  ]
}
```

3. Добавление товара в корзину

Запрос:

- Метод: POST
- URL: /api/cart
- Тело запроса:

```
{
  "productId": 1,
  "quantity": 1
}
```

Пример запроса:

```
POST /api/cart
Content-Type: application/json

{
  "productId": 1,
  "quantity": 1
}
```

Ответ:

```
{
  "message": "Product added to cart successfully.",
  "cart": {
    "items": [
      {
        "productId": 1,
        "name": "Smartphone XYZ",
        "price": 599.99,
        "quantity": 1
      }
    ],
    "totalPrice": 599.99
  }
}
```

Sequence UML Diagram

```
@startuml
```

```
actor User
participant "Frontend" as F
participant "Backend" as B
participant "Database" as DB

User -> F: Request product list
F -> B: GET /api/products
B -> DB: Query products
DB --> B: Return product list
B --> F: Return product list
F --> User: Display product list

User -> F: Click on product
F -> B: GET /api/products/{id}
B -> DB: Query product details
DB --> B: Return product details
B --> F: Return product details
F --> User: Display product details

User -> F: Add product to cart
F -> B: POST /api/cart
B -> DB: Update cart
DB --> B: Return updated cart
B --> F: Return updated cart
F --> User: Confirm addition to cart
@enduml
```

4. Алгоритмическое мышление

Для описания процесса пополнения баланса телефона на 100 рублей через банковское мобильное приложение, можно использовать диаграмму последовательности (Sequence Diagram). В данном примере я буду использовать гипотетическое банковское приложение "Тинькофф".

Пользователь → Смартфон: Включить телефон

Смартфон → Пользователь: Отобразить экран блокировки

Пользователь → Смартфон: Разблокировать телефон

Смартфон → Пользователь: Отобразить главный экран

Пользователь → Смартфон: Открыть приложение "Тинькофф"

Смартфон →Приложение "Тинькофф": Запустить приложение

Приложение "Тинькофф" →Пользователь: Отобразить экран входа

Пользователь →Приложение "Тинькофф": Ввести пароль для входа

Приложение "Тинькофф" →Сервер банка: Отправить запрос на авторизацию

Сервер банка →Приложение "Тинькофф": Подтвердить авторизацию

Приложение "Тинькофф" →Пользователь: Отобразить главный экран приложения

Пользователь → Приложение "Тинькофф": В поиске написать название оператора(МТС), выбрать

Приложение "Тинькофф" → Пользователь: Отобразить экран пополнения телефона

Пользователь → Приложение "Тинькофф": Ввести номер телефона и сумму (100 рублей)

Приложение "Тинькофф" →Пользователь: Отобразить подтверждение операции

Пользователь →Приложение "Тинькофф": Подтвердить операцию

Приложение "Тинькофф" → Сервер банка: Отправить запрос на пополнение баланса

Сервер банка →Приложение "Тинькофф": Подтвердить пополнение баланса и выполнение операции

Приложение "Тинькофф" →Пользователь: Отобразить успешное завершение операции