Database Test Solutions

1. Does a table without fields contain any information?

Answer: 3. A table without fields cannot exist.

Explanation: A table must have at least one field (column) to be valid. Without fields, the concept of a table becomes meaningless.

2. What kind of information can a record in a relational database file contain?

Answer: 4. Heterogeneous information (data of different types).

Explanation: A record (row) in a relational database can have multiple fields with different data types, such as integers, strings, dates, etc.

3. What distinguishes a primary key from a foreign key?

Answer:

- 2. Primary key values must be unique and cannot be NULL, while foreign key values may repeat.

- 4. A primary key identifies a row, while a foreign key links tables.

Explanation: A primary key uniquely identifies each record in a table, while a foreign key establishes a link between records in different tables.

4. In which normal form are all attributes dependent on the primary key, not just part of it?

Answer: 2. 2NF (Second Normal Form).

Explanation: Second Normal Form ensures that all non-key attributes are fully dependent on the entire primary key and not just a part of it.

5. In what order are SELECT, FROM, and GROUP BY executed in SQL?

Answer: 4. FROM → GROUP BY → SELECT.

Explanation: The SQL query execution order starts with `FROM` to retrieve data, applies `GROUP BY` to aggregate data, and finally evaluates the `SELECT` clause to return results.

6. What is the difference between WHERE and HAVING?

Answer:

- 2. HAVING filters groups, WHERE filters individual rows.

- 3. HAVING works only with aggregate functions, WHERE works with any expressions.

- 5. HAVING is always used after GROUP BY, WHERE can be used before or after GROUP BY.

Explanation: `WHERE` is used to filter individual rows before grouping. `HAVING` is used to filter groups after grouping and typically applies conditions involving aggregate functions.

7. What does SELECT COUNT(*) return?

Answer: 3. The number of rows in the table, including NULL values.

Explanation: `COUNT(*)` counts all rows in a table, including rows where columns have NULL values.

8. Query to retrieve the ages of all foxes in the zoo database:

Answer:

```sql
SELECT age FROM Animals WHERE Animal LIKE '%fox';
```

Explanation: The query uses `LIKE '%fox'` to match any animal whose name ends with "fox" (e.g., red fox, grey fox).

9. What is the difference between DELETE and TRUNCATE?

Answer:

2. DELETE removes one or more rows, TRUNCATE removes all rows from a table.

3. DELETE can have a WHERE condition, TRUNCATE always deletes all records.

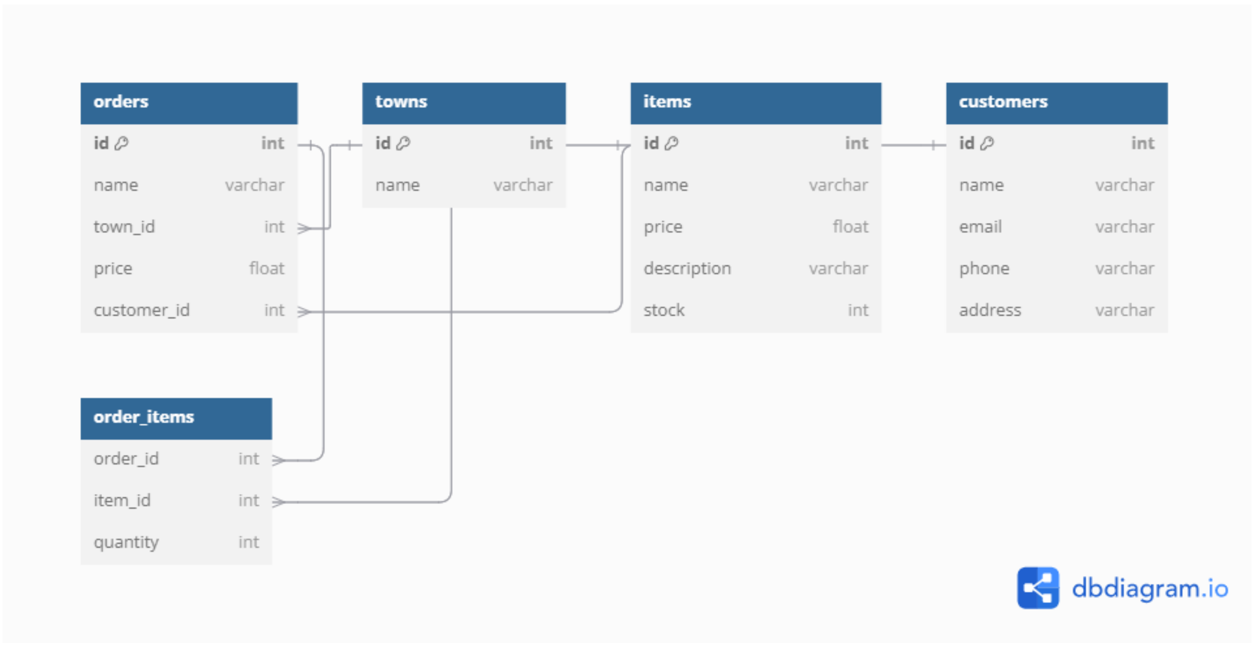
Explanation: `DELETE` allows conditional deletion of specific rows. `TRUNCATE` removes all rows efficiently without logging individual row deletions.

10. Result of SELECT COUNT(DISTINCT color) FROM Table:

Answer: 4. 2 (BLUE and RED).

Explanation: `COUNT(DISTINCT color)` counts unique non-NULL values. Here, only BLUE and RED are distinct, while NULL is not counted.

ER DIAGRAM



orders
id int
name varchar
town_id int
price float
customer_id int
towns
id int
name varchar
items
id int
name varchar
price float
description varchar
stock int
customers
id int
name varchar
email varchar
phone varchar
address varchar
order_items
order_id int
item_id int
quantity int
dbdiagram.io

REST API DESIGN

Endpoint 1: Get List of Products

URL: GET /products

Description: Fetches a list of all available products with a brief description.

Request Parameters: None

Response: A list of products, each containing its ID, name, description, and price.

<u>Request:</u>

http

GET /products

<u>Response (JSON):</u>

```
[
 {
   "id": 1,
   "name": "Product 1",
   "description": "Brief description of Product 1",
   "price": 100.00
 },
 {
   "id": 2,
   "name": "Product 2",
   "description": "Brief description of Product 2",
   "price": 150.00
 },
 {
   "id": 3,
   "name": "Product 3",
   "description": "Brief description of Product 3",
   "price": 200.00
 }
]
```

Endpoint 2: Get Detailed Product Information

URL: GET /products/{id}

Description: Fetches detailed information for a specific product.

Request Parameters: id (Product ID)

Response: Detailed information about a specific product, including its ID, name, description, price, stock, and product images.

<u>Request:</u>

```
http
GET /products/1
```

<u>Response (JSON):</u>

```
{
  "id": 1,
  "name": "Product 1",
  "description": "Detailed description of Product 1.",
  "price": 100.00,
  "stock": 50,
  "images": [
    "product1-image1.jpg",
    "product1-image2.jpg"
  ]
}
```

Endpoint 3: Add Product to Cart

URL: POST /cart

Description: Adds a specified product to the user's shopping cart.

Request Parameters: Product ID and quantity.

Response: A message indicating that the product has been successfully added to the cart along with the updated cart information.

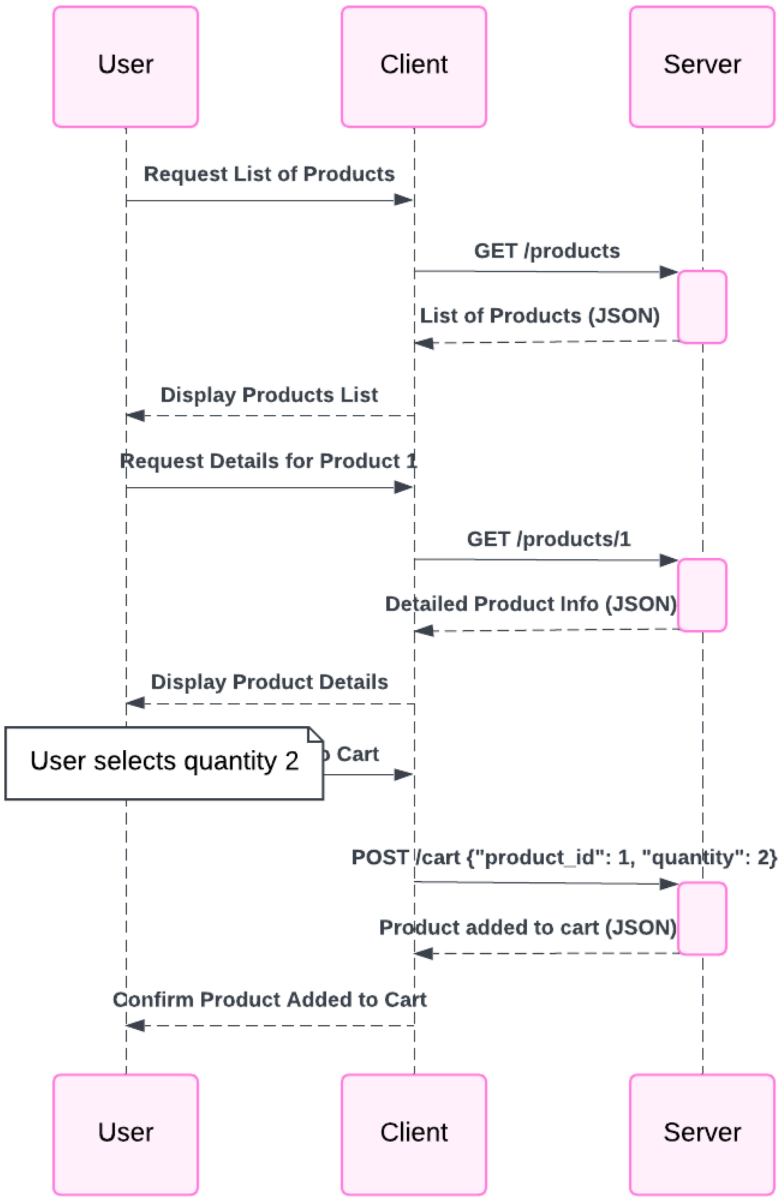
<u>Request (JSON):</u>

```
{
  "product_id": 1,
  "quantity": 2
}
```

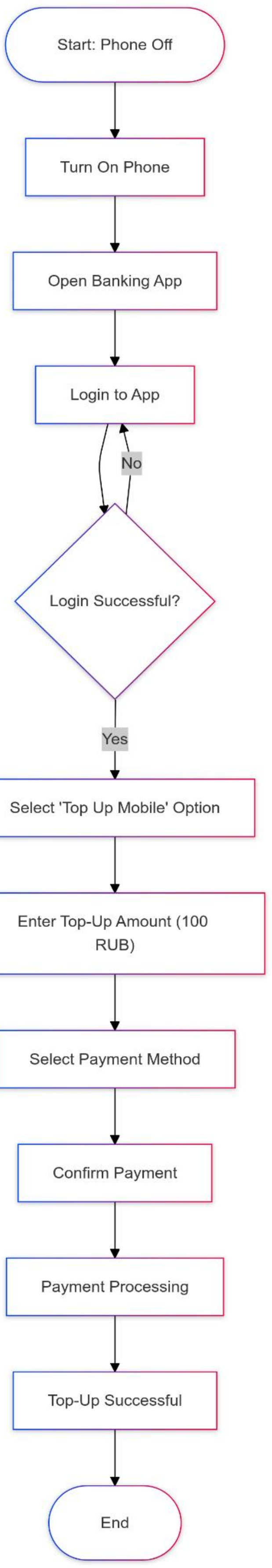
Response (JSON):

```
{
  "message": "Product added to cart",
  "cart": [
    {
      "product_id": 1,
      "name": "Product 1",
      "quantity": 2,
      "price": 100.00
    }
  ]
}
```

UML SEQUENCE DIAGRAM



ACTIVITY DIAGRAM



Start: Phone Off
Turn On Phone
Open Banking App
Login to App
No
Login Successful?
Yes
Select 'Top Up Mobile' Option
Enter Top-Up Amount (100 RUB)
Select Payment Method
Confirm Payment
Payment Processing
Top-Up Successful
End