Crystal clear, Charan 👍 — you want me to generate a new problem statement in the same structured format (models, JWT auth, endpoints, response codes), but with more manipulation and complexity than the bidding system we just built.

Here's a fresh one for you:

🗂️ Models

ProductModel

Field Name	Datatype	Primary Key	Foreign Key	Comments
id	Integer	Yes	No	Autoincrement
name	String	No	No	Unique
category	String	No	No	-
price	Double	No	No	-
stock	Integer	No	No	Default: 0

OrderModel

Field Name	Datatype	Primary Key	Foreign Key	Comments
id	Integer	Yes	No	Autoincrement
productId	Integer	No	Yes	FK to ProductModel
quantity	Integer	No	No	-
totalPrice	Double	No	No	Calculated: quantity * product.price
orderDate	String	No	No	Default: Current date (dd/MM/yyyy)
status	String	No	No	Default: "pending"
customerId	Integer	No	Yes	FK to UserModel

UserModel

Field Name	Datatype	Primary Key	Foreign Key	Comments
id	Integer	Yes	No	Autoincrement
username	String	No	No	-
email	String	No	No	Unique
password	String	No	No	-
role	String	No	No	Values: "CUSTOMER", "ADMIN"

🔐 JWT Authentication

- JWT token required in Authorization header:

`

```
  Authorization: Bearer <JWT_TOKEN>
  `
```
- Roles (CUSTOMER / ADMIN) are identified from the token.
- All endpoints except /login require authentication.

🌐 Endpoints

1. POST /login
- Purpose: Authenticate user and return JWT.
- Request Body:
```
`json
{
  "email": "customer1@gmail.com",
  "password": "cust123$"
}
`
```
- Success Response (200 OK):
```
`json
{
  "email": "customer1@gmail.com",
  "access": "<access_token>",
  "refresh": "<refresh_token>",
  "status": 200
}
`
```
- Error: 400 Bad Request → "invalid credentials"

---

2. POST /product/add (ADMIN only)
- Adds a new product.
- Request Body:
```
`json
{
  "name": "Laptop",
  "category": "Electronics",
  "price": 75000,
  "stock": 10
}
`
```
- Success Response (201 CREATED):
```
`json
{
  "id": 101,
  "name": "Laptop",
  "category": "Electronics",
  "price": 75000,
  "stock": 10
}
```

---

3. GET /product/list?category=Electronics&minPrice=50000 (Both roles)
- Returns all products filtered by category and minimum price.
- Success: 200 OK with list.
- Error: 400 Bad Request → "no products found"

---

4. POST /order/create (CUSTOMER only)
- Creates a new order.
- Request Body:

```json
{
  "productId": 101,
  "quantity": 2
}
```

- Manipulation: totalPrice is auto-calculated as quantity * product.price.
- Success Response (201 CREATED):

```json
{
  "id": 5001,
  "productId": 101,
  "quantity": 2,
  "totalPrice": 150000,
  "orderDate": "28/11/2025",
  "status": "pending",
  "customerId": 1
}
```

---

5. PATCH /order/update/{id} (ADMIN only)
- Updates order status.
- Request Body:

```json
{ "status": "shipped" }
```

- Success Response (200 OK): Updated order object.
- Error: 400 Bad Request → "order not found"

---

6. DELETE /order/delete/{id} (Conditional access)
- Accessible by:
  - Admin
  - Customer who created the order
- Success: 204 No Content → "deleted successfully"
- Error:
  - 400 Bad Request → "order not found"
  - 403 Forbidden → "you don't have permission"

📑 Response Codes
- 200 OK – Success
- 201 CREATED – Resource created
- 204 No Content – Deleted successfully
- 400 Bad Request – Invalid input / Not found
- 403 Forbidden – Permission denied

✅ This new problem statement adds more manipulation:
- Auto-calculated totalPrice in orders.
- Filtering products by multiple query params (category, minPrice).
- Role-based access with both CUSTOMER and ADMIN.