Hibernate beginner-friendly projects

1. Student Record Management

Table Name: `students`

Fields:

- `id` (int, PK)
- `name` (String)
- `email` (String)
- `course` (String)
- `marks` (int)

Functionality:

- Add a student
- View all students
- Update marks by student ID
- Delete a student

🔧 **Purpose:** Basic CRUD operations using Hibernate and Scanner input.

---

2. Employee Directory System

Table Name: `employees`

Fields:

- `id` (int, PK)
- `name` (String)
- `designation` (String)
- `salary` (double)
- `department` (String)

Functionality:

- Add new employee
- Search employee by name
- Update salary
- Delete employee by ID

🔧 **Purpose:** Understand how to query and update single-table records.

---

3. Product Inventory Manager

Table Name: `products`

Fields:

- `id` (int, PK)
- `name` (String)
- `price` (double)
- `quantity` (int)
- `manufacturer` (String)

Functionality:

- Add product
- View all products
- Update quantity
- Delete product by ID

🔧 **Purpose:** Managing numeric updates and string fields.

---

4. Library Book Catalog

Table Name: `books`

Fields:

- `id` (int, PK)
- `title` (String)
- `author` (String)
- `isbn` (String)
- `availableCopies` (int)

Functionality:

- Add book
- Search by author
- Update available copies
- Delete by ISBN

🔧 **Purpose:** Practice filtering and updating string-based IDs.

---

5. Vehicle Registration System

Table Name: `vehicles`

Fields:

- `id` (int, PK)
- `ownerName` (String)
- `vehicleNumber` (String)
- `model` (String)
- `registrationYear` (int)

Functionality:

- Add new vehicle
- Search by model or year
- Update vehicle number
- Delete by ID

🔧 **Purpose:** Practice multi-field filtering and updates.

---

6. Patient Record System

Table Name: `patients`

Fields:

- `id` (int, PK)
- `name` (String)
- `age` (int)
- `disease` (String)
- `admittedDate` (String or Date)

Functionality:

- Add new patient
- View all records
- Update disease or admitted date
- Delete patient by ID

🔧 **Purpose:** Date and string handling using Hibernate.

---

7. Online Course Registration

Table Name: `registrations`

Fields:

- `id` (int, PK)
- `studentName` (String)
- `courseName` (String)
- `registrationDate` (String or Date)
- `feesPaid` (double)

Functionality:

- Register for course
- View all registrations
- Update fees paid
- Delete registration by ID

🔧 **Purpose:** Deal with currency (double) and date input.

---

8. Movie Ticket Booking System

Table Name: `bookings`

Fields:

- `id` (int, PK)
- `movieName` (String)
- `customerName` (String)
- `showTime` (String)
- `seatsBooked` (int)

Functionality:

- Book a ticket
- View bookings
- Update number of seats
- Delete by customer name

🔧 **Purpose:** Manipulate seat numbers and time strings.

---

9. Bank Account Manager

Table Name: `accounts`

Fields:

- `id` (int, PK)
- `accountHolder` (String)
- `accountType` (String)
- `balance` (double)
- `openedDate` (String or Date)

Functionality:

- Create new account
- View account by ID
- Update balance
- Close account (delete)

🔧 **Purpose:** Handling numeric updates (e.g., deposit/withdraw).

---

10. Feedback Collection System

Table Name: `feedbacks`

Fields:

- `id` (int, PK)
- `userName` (String)
- `email` (String)

- `rating` (int, 1-5)
- `comments` (String)

Functionality:

- Submit feedback
- View all feedback
- Update comments
- Delete by email

🔧 **Purpose:** Mix of string and numeric types; validation practice.

🧱 Project Structure (Suggestion for All)

All projects should follow this Maven structure:

```
 1  src/
 2  ├── main/
 3  │    ├── java/
 4  │    │    └── com/projectname/
 5  │    │         ├── App.java (main class with Scanner)
 6  │    │         ├── entity/
 7  │    │         │    └── ModelClass.java
 8  │    │         └── repository/
 9  │    │              └── repoClass.java
10  │    └── resources/
11  │         └── hibernate.cfg.xml
12  pom.xml
13
```