To create an AWS Lambda Spring Boot project with a three-tier architecture and RESTful APIs for user registration, login, and order management (create/update/delete/list orders), with user data stored in DynamoDB and user profile images uploaded to S3, you can follow these steps:

1. **Setup AWS Account**: Ensure you have an AWS account set up.
2. **Setup AWS CLI**: Install and configure AWS CLI on your development machine.
3. **Create a DynamoDB Table**: Create a DynamoDB table to store user data and orders. Define appropriate partition and sort keys, and any other attributes required for your application.
4. **Create an S3 Bucket**: Create an S3 bucket to store user profile images.
5. **Create a Spring Boot Project**:
   * Use Spring Initializr (<https://start.spring.io/>) to generate a new Spring Boot project with dependencies for Web, AWS, DynamoDB, and any other dependencies you need.
   * Include the appropriate dependencies in your **pom.xml** or **build.gradle** file.
6. **Create User and Order Models**:
   * Create Java classes to represent user and order entities. Annotate them with appropriate JPA annotations if using Spring Data DynamoDB.
   * Ensure the user model includes fields for username, email, password, and a reference to the user's profile image stored in S3.
7. **Create DynamoDB Repository Interfaces**:
   * Create repository interfaces extending **DynamoDBCrudRepository** or **DynamoDBRepository** to interact with DynamoDB tables for user and order entities.
8. **Implement Service Layer**:
   * Implement service classes to handle business logic for user registration, login, and order management. Use repositories to interact with DynamoDB.
9. **Implement RESTful Controllers**:
   * Create REST controllers to expose endpoints for user registration, login, and order management operations. Use service layer to process requests.
10. **Implement User Registration**:
    * Implement a REST endpoint to handle user registration. When a user registers, upload their profile image to S3 and store user data in DynamoDB.
11. **Implement User Login**:
    * Implement a REST endpoint to handle user login. Authenticate user credentials against data stored in DynamoDB.
12. **Implement Order Management**:
    * Implement REST endpoints for creating, updating, deleting, and listing orders. Store order data in DynamoDB.
13. **Testing and Deployment**:
    * Test your application locally using embedded DynamoDB and mock S3 (such as LocalStack).
    * Deploy your Lambda functions and related resources (DynamoDB table, S3 bucket) using AWS CLI or AWS Management Console.
14. **Setup API Gateway**:
    * Create an API Gateway to expose your Lambda functions as RESTful APIs.
15. **Security Considerations**:
    * Implement proper authentication and authorization mechanisms for user login and access to order management APIs.
    * Securely manage credentials and access keys for interacting with AWS services.

By following these steps, you can create a serverless Spring Boot application using AWS Lambda, DynamoDB, and S3 to implement user registration, login, and order management functionalities with a three-tier architecture.