***Shiv Sunder Dash report***

Q1) For the Customer Information Dashboard, I created the Customer entity with fields id, name, email, and registeredDate, and used the required JPA annotations. The entity is mapped to a customers table. I set up the CustomerRepository by extending JpaRepository and built a REST controller to handle all CRUD operations—adding, getting (by ID and all), and deleting customers. I also added query methods like findByEmailContaining, findByRegisteredDateAfter, and used @Query to fetch customers by name. The setup was tested with a PostgreSQL database, and unit tests were written using @DataJpaTest.

Q2) For the Product Information Dashboard, I created the Product entity with fields like id, name, description, price, stockQuantity, category, createdAt, and updatedAt, using all necessary annotations including @CreationTimestamp and @UpdateTimestamp. I implemented the ProductRepository with query methods such as findByCategory, findByPriceBetween, and findByNameContainingIgnoreCase, and added a custom JPQL query using @Query. All CRUD endpoints were built through a REST controller and tested with a MySQL database. Unit tests and integration tests were written using @DataJpaTest, @SpringBootTest, and MockMvc.

MCQ:

1. C) To enable scanning for Spring Data JPA interfaces
2. B) spring.jpa.database-platform
3. B) spring.datasource.url
4. B) spring.jpa.hibernate.ddl-auto=create
5. C) deleteById(id)
6. B) List<T>
7. A) save(entity)
8. C) Enables custom JPQL or native queries
9. D) getAllEntities()
10. D) @Column
11. C) @Entity
12. A) Primary key
13. C) @GeneratedValue
14. B) LAZY
15. C) application.properties