1. **Create a class Address for Author with instance variables streetNumber, location, State.**

   <Address.java>

```java
package com.springdataaccessandtransactions.Assignment3.entities;


import javax.persistence.Embeddable;

@Embeddable
public class Address {
  private int streetnumber;
  private String location;
  private String state;

  public int getStreetnumber() {
     return streetnumber;
  }

  public void setStreetnumber(int streetnumber) {
     this.streetnumber = streetnumber;
  }

  public String getLocation() {
     return location;
  }

  public void setLocation(String location) {
     this.location = location;
  }

  public String getState() {
     return state;
  }

  public void setState(String state) {
     this.state = state;
  }
}
```

2. **Create instance variable of Address class inside Author class and save it as embedded object.**

```
package com.springdataaccessandtransactions.Assignment3.entities;


import javax.persistence.*;

@Entity
public class Author {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private int authorid;
  private String name;

  @Embedded
  private Address address;

  public int getAuthorid() {
     return authorid;
  }

  public void setAuthorid(int authorid) {
     this.authorid = authorid;
  }

  public String getName() {
     return name;
  }

  public void setName(String name) {
     this.name = name;
  }

  public Address getAddress() {
     return address;
  }

  public void setAddress(Address address) {
     this.address = address;
  }
}
```

3. **Introduce a List of subjects for author.**

<Author.java>

```
package com.springdataaccessandtransactions.Assignment3.entities;

import javax.persistence.*;
import java.util.List;

@Entity
public class Author {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private int authorId;
  private String name;

  @OneToMany(mappedBy = "author", cascade = CascadeType.PERSIST)
  private List<Subject> subjects;

  @Embedded
  private Address address;

  public int getAuthorId() {
     return authorId;
  }

  public void setAuthorId(int authorId) {
     this.authorId = authorId;
  }

  public String getName() {
     return name;
  }

  public void setName(String name) {
     this.name = name;
  }

  public Address getAddress() {
     return address;
  }

  public void setAddress(Address address) {
     this.address = address;
  }
}
```

4. **Persist 3 subjects for each author.**

<Author.java>

```
package com.springdataaccessandtransactions.Assignment3.entities;


import javax.persistence.*;
import java.util.List;

@Entity
public class Author {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private int authorId;
  private String name;

  @OneToMany(mappedBy = "author", cascade = CascadeType.PERSIST)
  private List<Subject> subjects;

  @Embedded
  private Address address;

  public int getAuthorId() {
     return authorId;
  }

  public void setAuthorId(int authorId) {
     this.authorId = authorId;
  }

  public String getName() {
     return name;
  }

  public void setName(String name) {
     this.name = name;
  }

  public Address getAddress() {
     return address;
```

```
    }

    public void setAddress(Address address) {
        this.address = address;
    }
}
```

<ApplicationTest.java>

```
@Test

public void testCreateAuthor(){
  Author author = new Author();
  author.setName("Vaishali");

  Address address = new Address();
  address.setLocation("Faridabad");
  address.setStreetnumber(131);
  address.setState("Haryana");

  author.setAddress(address);


  Subject subject1 = new Subject();
  subject1.setSubName("Maths");

  Subject subject2 = new Subject();
  subject2.setSubName("English");

  Subject subject3 = new Subject();
  subject3.setSubName("Hindi");

  author.addSubject(subject1);
  author.addSubject(subject2);
  author.addSubject(subject3);


  Book book1 = new Book();
  book1.setBookName("Java");

  Book book2 = new Book();
  book2.setBookName("Cpp");

  author.addBook(book1);
```

```
 author.addBook(book2);

 authorRepository.save(author);
}
```

5. **Create an Entity book with an instance variable bookName.**

```
package com.springdataaccessandtransactions.Assignment3.entities;


import javax.persistence.*;

@Entity
public class Book {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  @Column(name = "book_id")
  private int bookId;

  @Column(name = "bookname")
  private String bookName;

  @ManyToOne
  @JoinColumn(name = "authorid")
  private Author author;

  public int getBookId() {
      return bookId;
  }

  public void setBookId(int bookId) {
      this.bookId = bookId;
  }

  public String getBookName() {
      return bookName;
  }

  public void setBookName(String bookName) {
      this.bookName = bookName;
  }
  public Author getAuthor() {
      return author;
  }

  public void setAuthor(Author author) {
      this.author = author;
  }
}
```

6. **Implement One to One mapping between Author and Book.**

<AuthorOne.java>

```
package com.springdataaccessandtransactions.Assignment3.onetoone.entities;


import javax.persistence.*;

@Entity
@Table(name = "authorone")
public class AuthorOne {

  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  @Column(name = "author_id")
  private int id;
  private String name;

  @OneToOne(mappedBy = "author")
  private BookOne book;

  public int getId() {
      return id;
  }

  public void setId(int id) {
      this.id = id;
  }

  public String getName() {
      return name;
  }

  public void setName(String name) {
      this.name = name;
  }

  public BookOne getBook() {
      return book;
  }

  public void setBook(BookOne book) {
      this.book = book;
  }
}
```

<BookOne.java>

```
package com.springdataaccessandtransactions.Assignment3.onetoone.entities;

import javax.persistence.*;

@Entity
@Table(name = "bookone")
public class BookOne {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "book_id")
    private int id;

    @Column(name = "bookname")
    private String bookName;

    @OneToOne(cascade = CascadeType.ALL)
    @JoinColumn(name = "authorid")
    private AuthorOne author;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getBookName() {
        return bookName;
    }

    public void setBookName(String bookName) {
        this.bookName = bookName;
    }

    public AuthorOne getAuthor() {
        return author;
    }

    public void setAuthor(AuthorOne author) {
        this.author = author;
    }
}
```

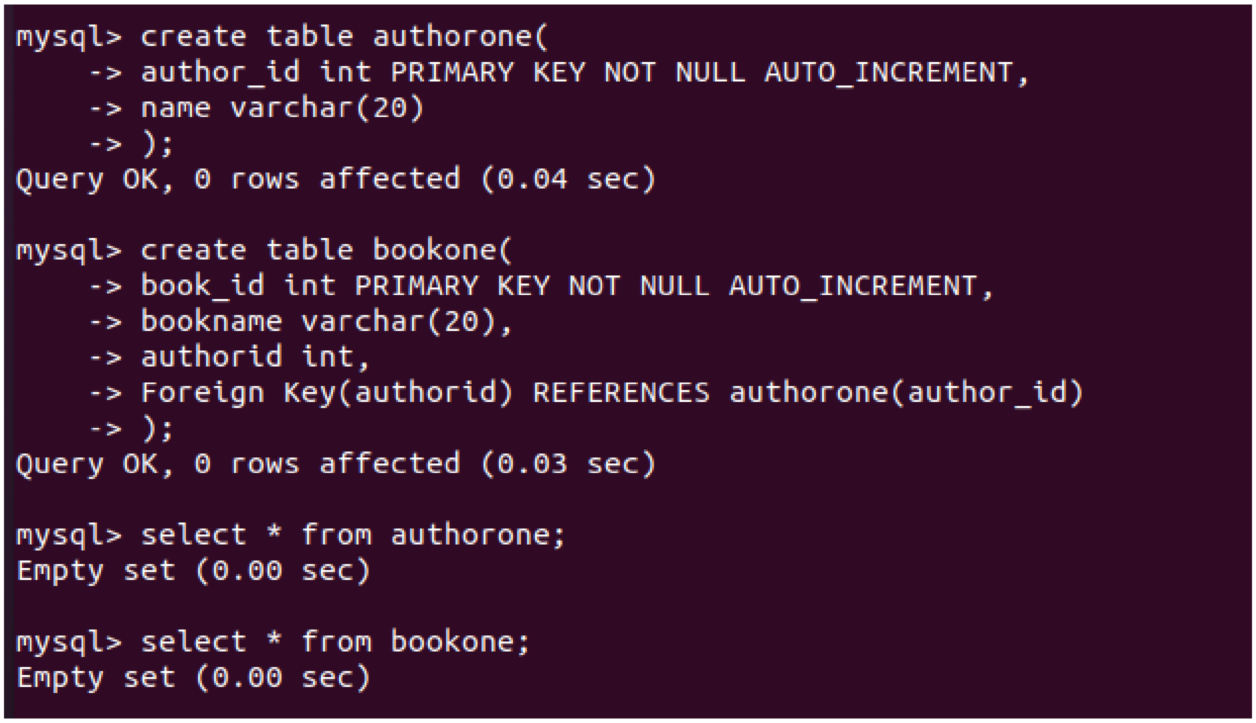
<BookOneRepository.java>

```
package com.springdataaccessandtransactions.Assignment3.onetoone.repos;

import com.springdataaccessandtransactions.Assignment3.onetoone.entities.BookOne;
import org.springframework.data.repository.CrudRepository;

public interface BookOneRepository extends CrudRepository<BookOne,Integer> {

}
```

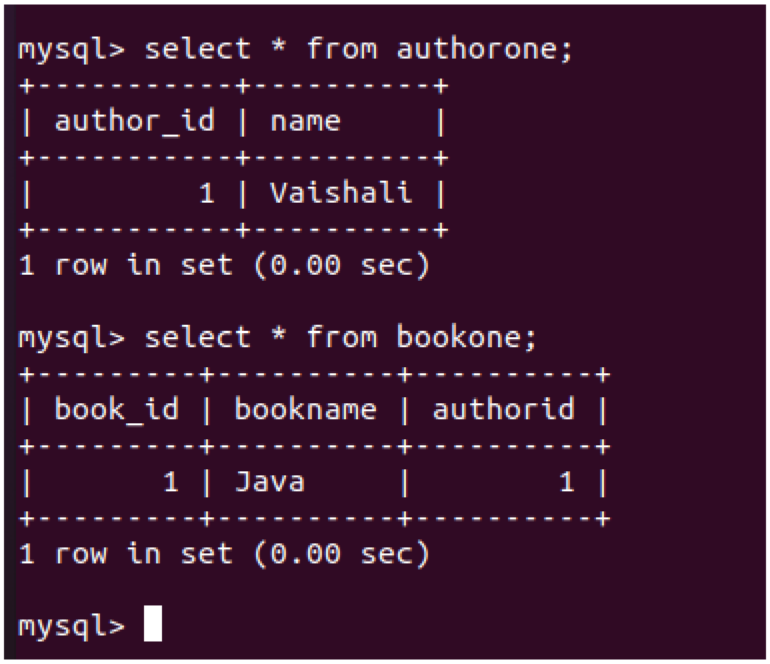
<OneToOneTest.java>

```
package com.springdataaccessandtransactions.Assignment3.onetoonetests;

import
com.springdataaccessandtransactions.Assignment3.onetoone.entities.AuthorOne;
import com.springdataaccessandtransactions.Assignment3.onetoone.entities.BookOne;
import
com.springdataaccessandtransactions.Assignment3.onetoone.repos.BookOneRepositor
y;
```

```
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit4.SpringRunner;

@RunWith(SpringRunner.class)
@SpringBootTest
class OneToOneTests {

  @Autowired
  BookOneRepository bookOneRepository;

  @Test
  public void testOneToOne(){
     BookOne bookOne = new BookOne();
     bookOne.setBookName("Java");

     AuthorOne author = new AuthorOne();
     author.setName("Vaishali");
     author.setBook(bookOne);

     bookOne.setAuthor(author);
     bookOneRepository.save(bookOne);
  }
}
```



```
mysql> create table authorone(
    -> author_id int PRIMARY KEY NOT NULL AUTO_INCREMENT,
    -> name varchar(20)
    -> );
Query OK, 0 rows affected (0.04 sec)

mysql> create table bookone(
    -> book_id int PRIMARY KEY NOT NULL AUTO_INCREMENT,
    -> bookname varchar(20),
    -> authorid int,
    -> Foreign Key(authorid) REFERENCES authorone(author_id)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> select * from authorone;
Empty set (0.00 sec)

mysql> select * from bookone;
Empty set (0.00 sec)
```



```
mysql> select * from authorone;
+-----------+----------+
| author_id | name     |
+-----------+----------+
|         1 | Vaishali |
+-----------+----------+
1 row in set (0.00 sec)

mysql> select * from bookone;
+---------+----------+----------+
| book_id | bookname | authorid |
+---------+----------+----------+
|       1 | Java     |        1 |
+---------+----------+----------+
1 row in set (0.00 sec)

mysql> 
```

7. **Implement One to Many Mapping between Author and Book(Unidirectional, BiDirectional and without additional table) and implement cascade save.**

<Book.java>

```
package com.springdataaccessandtransactions.Assignment3.entities;


import javax.persistence.*;

@Entity
public class Book {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  @Column(name = "book_id")
  private int bookId;

  @Column(name = "bookname")
  private String bookName;

  /**** Bidirectional One to Many Mapping *****/
 /* @ManyToOne
  @JoinColumn(name = "authorid")
  private Author author;
  */

  public int getBookId() {
     return bookId;
  }

  public void setBookId(int bookId) {
     this.bookId = bookId;
  }

  public String getBookName() {
     return bookName;
  }

  public void setBookName(String bookName) {
     this.bookName = bookName;
  }

  /*public Author getAuthor() {
     return author;
  }
```

```
    public void setAuthor(Author author) {
        this.author = author;
    } */
}
```

<Author.java>

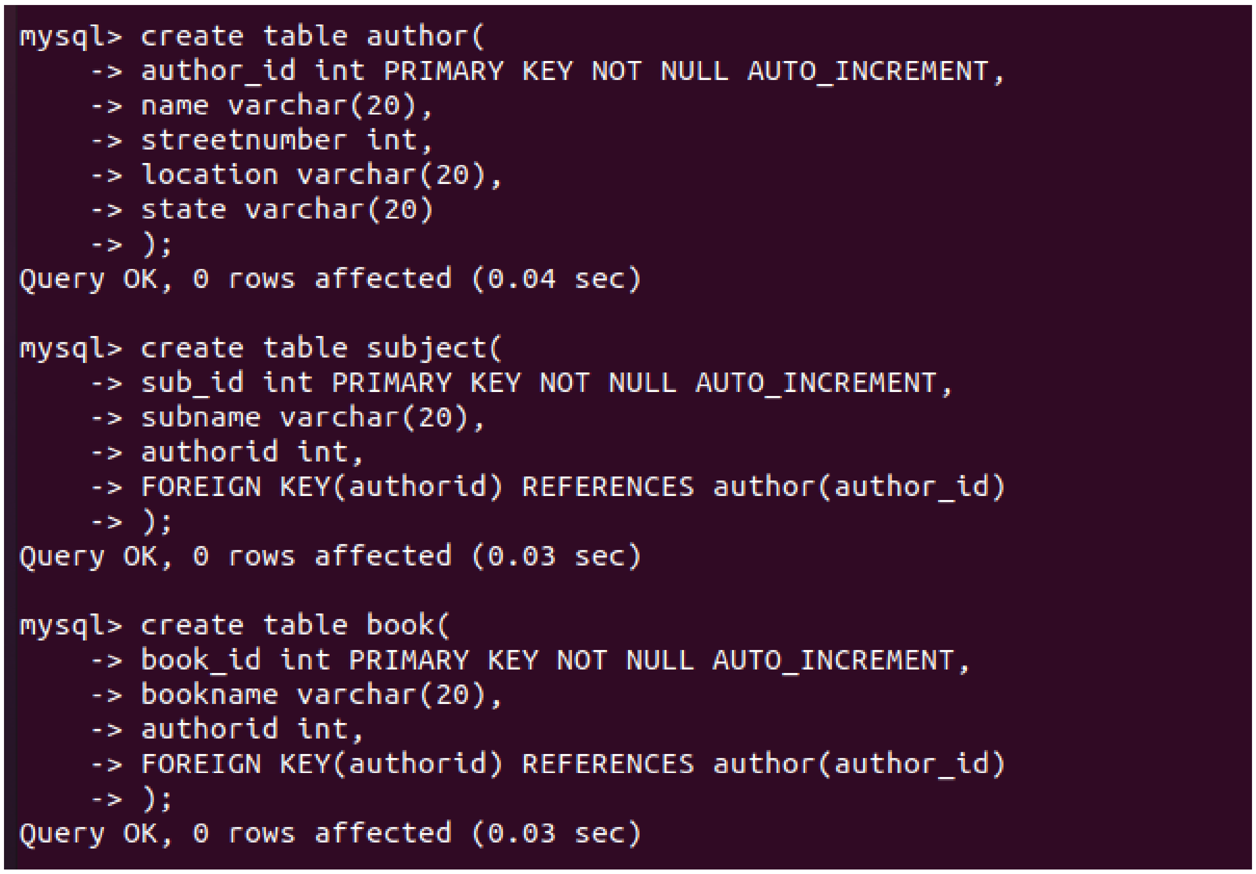
```
package com.springdataaccessandtransactions.Assignment3.entities;

import javax.persistence.*;
import java.util.ArrayList;
import java.util.List;

@Entity
@Table(name = "author")
public class Author {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "author_id")
    private int authorId;
    private String name;

    @OneToMany(mappedBy = "author", cascade = CascadeType.PERSIST)
    private List<Subject> subjects;

    /******** Bidirectional One to Many Mapping ******/
    //@OneToMany(mappedBy = "author", cascade = CascadeType.ALL)
    /****** Unidirectional One To Many mapping *****/
    @OneToMany(cascade = CascadeType.ALL)
    @JoinColumn(name = "authorid")
    private List<Book> books;

    @Embedded
    private Address address;

    public int getAuthorId() {
        return authorId;
    }

    public void setAuthorId(int authorId) {
        this.authorId = authorId;
    }

    public String getName() {
```

```
    return name;
}

public void setName(String name) {
    this.name = name;
}

public Address getAddress() {
    return address;
}

public void setAddress(Address address) {
    this.address = address;
}

public List<Subject> getSubjects() {
    return subjects;
}

public void setSubjects(List<Subject> subjects) {
    this.subjects = subjects;
}

public List<Book> getBooks() {
    return books;
}

public void setBooks(List<Book> books) {
    this.books = books;
}

/*public void addBook(Book book){
    if(book != null){
        if(books == null){
            books = new ArrayList<>();
        }
        book.setAuthor(this);
        books.add(book);
    }
} */

public void addSubject(Subject subject){
    if(subject != null){
        if(subjects == null)
```

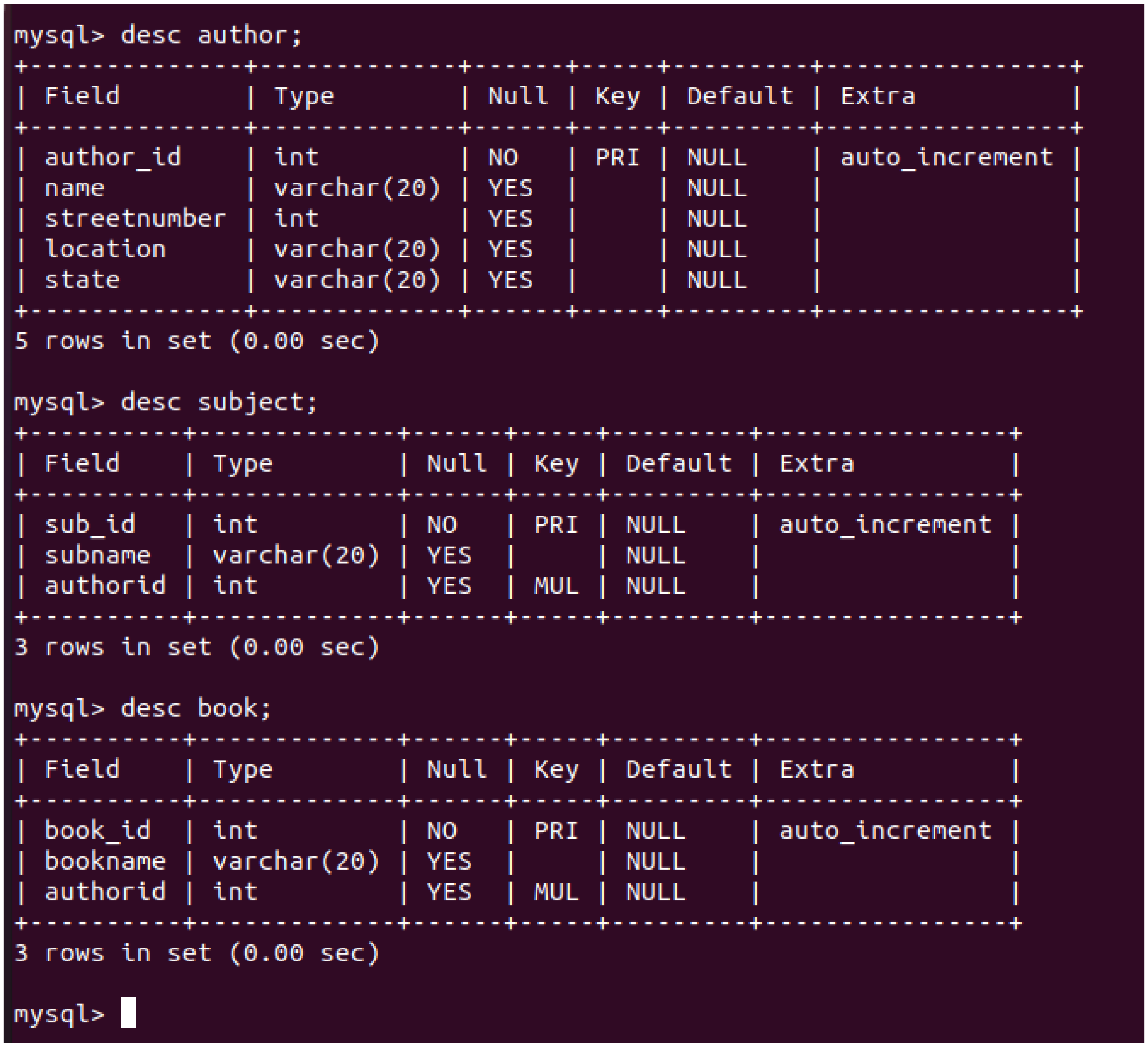
```
                subjects = new ArrayList<>();

            subject.setAuthor(this);
            subjects.add(subject);
        }
    }
}
```

<ApplicationTest.java>

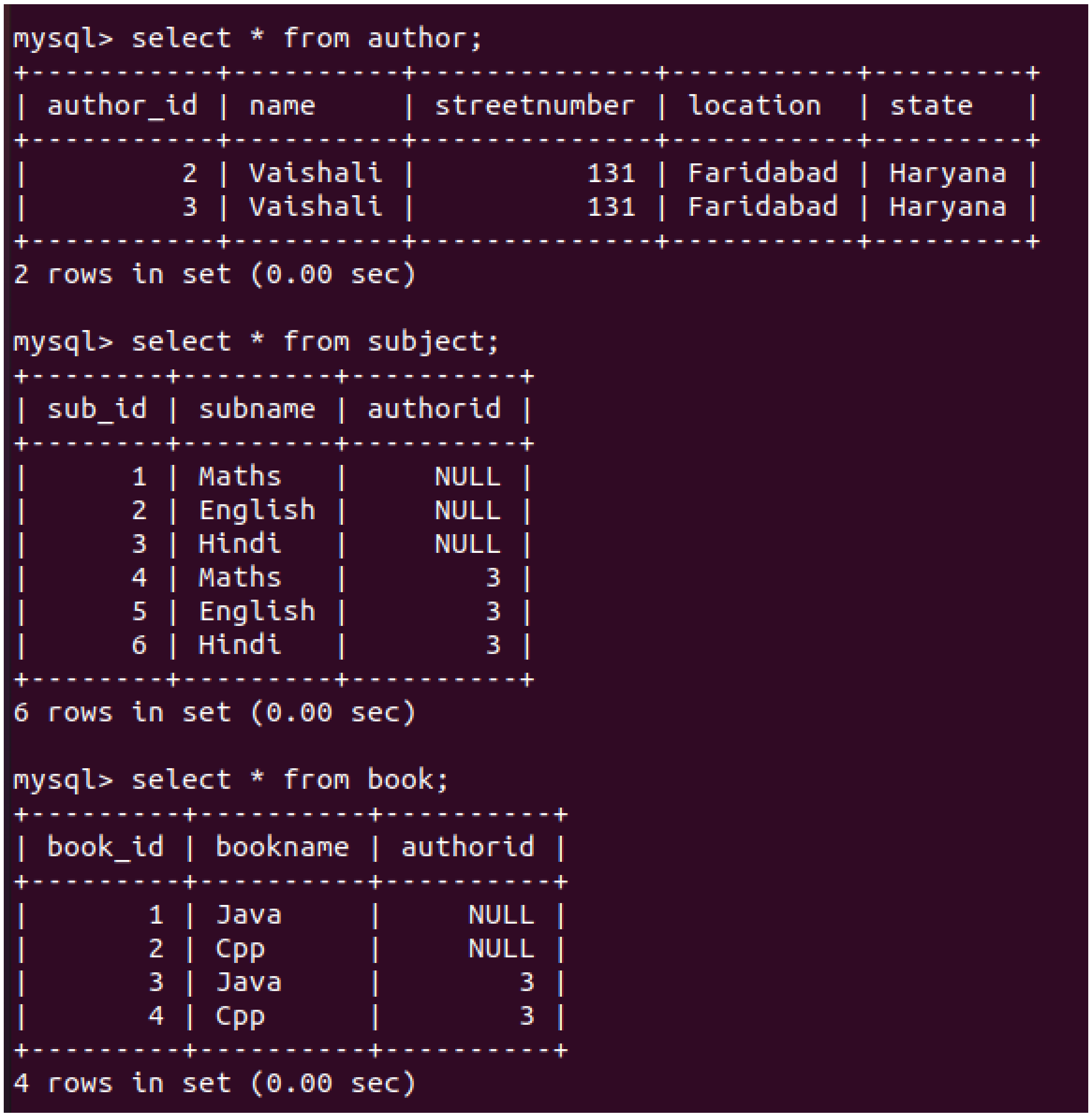
```
@Test
public void testCreateAuthor(){
  Author author = new Author();
  author.setName("Vaishali");

  Address address = new Address();
  address.setLocation("Faridabad");
  address.setStreetnumber(131);
  address.setState("Haryana");

  author.setAddress(address);

  List<Subject> subjects = new ArrayList<>();
  Subject subject1 = new Subject();
  subject1.setSubName("Maths");
  subject1.setAuthor(author);
  subjects.add(subject1);

  Subject subject2 = new Subject();
  subject2.setSubName("English");
  subject2.setAuthor(author);
  subjects.add(subject2);

  Subject subject3 = new Subject();
  subject3.setSubName("Hindi");
  subject3.setAuthor(author);
  subjects.add(subject3);

  author.setSubjects(subjects);

  List<Book> books = new ArrayList<>();
  Book book1 = new Book();
  book1.setBookName("Java");

  Book book2 = new Book();
  book2.setBookName("Cpp");

  books.add(book1);
  books.add(book2);

  author.setBooks(books);

  authorRepository.save(author);
}
```



```
mysql> create table author(
    -> author_id int PRIMARY KEY NOT NULL AUTO_INCREMENT,
    -> name varchar(20),
    -> streetnumber int,
    -> location varchar(20),
    -> state varchar(20)
    -> );
Query OK, 0 rows affected (0.04 sec)

mysql> create table subject(
    -> sub_id int PRIMARY KEY NOT NULL AUTO_INCREMENT,
    -> subname varchar(20),
    -> authorid int,
    -> FOREIGN KEY(authorid) REFERENCES author(author_id)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> create table book(
    -> book_id int PRIMARY KEY NOT NULL AUTO_INCREMENT,
    -> bookname varchar(20),
    -> authorid int,
    -> FOREIGN KEY(authorid) REFERENCES author(author_id)
    -> );
Query OK, 0 rows affected (0.03 sec)
```



```
mysql> desc author;
+--------------+-------------+------+-----+---------+----------------+
| Field        | Type        | Null | Key | Default | Extra          |
+--------------+-------------+------+-----+---------+----------------+
| author_id    | int         | NO   | PRI | NULL    | auto_increment |
| name         | varchar(20) | YES  |     | NULL    |                |
| streetnumber | int         | YES  |     | NULL    |                |
| location     | varchar(20) | YES  |     | NULL    |                |
| state        | varchar(20) | YES  |     | NULL    |                |
+--------------+-------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)

mysql> desc subject;
+----------+-------------+------+-----+---------+----------------+
| Field    | Type        | Null | Key | Default | Extra          |
+----------+-------------+------+-----+---------+----------------+
| sub_id   | int         | NO   | PRI | NULL    | auto_increment |
| subname  | varchar(20) | YES  |     | NULL    |                |
| authorid | int         | YES  | MUL | NULL    |                |
+----------+-------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)

mysql> desc book;
+----------+-------------+------+-----+---------+----------------+
| Field    | Type        | Null | Key | Default | Extra          |
+----------+-------------+------+-----+---------+----------------+
| book_id  | int         | NO   | PRI | NULL    | auto_increment |
| bookname | varchar(20) | YES  |     | NULL    |                |
| authorid | int         | YES  | MUL | NULL    |                |
+----------+-------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)

mysql> 
```



```
mysql> select * from author;
+-----------+----------+--------------+-----------+---------+
| author_id | name     | streetnumber | location  | state   |
+-----------+----------+--------------+-----------+---------+
|         2 | Vaishali |          131 | Faridabad | Haryana |
|         3 | Vaishali |          131 | Faridabad | Haryana |
+-----------+----------+--------------+-----------+---------+
2 rows in set (0.00 sec)

mysql> select * from subject;
+--------+---------+----------+
| sub_id | subname | authorid |
+--------+---------+----------+
|      1 | Maths   |     NULL |
|      2 | English |     NULL |
|      3 | Hindi   |     NULL |
|      4 | Maths   |        3 |
|      5 | English |        3 |
|      6 | Hindi   |        3 |
+--------+---------+----------+
6 rows in set (0.00 sec)

mysql> select * from book;
+---------+----------+----------+
| book_id | bookname | authorid |
+---------+----------+----------+
|       1 | Java     |     NULL |
|       2 | Cpp      |     NULL |
|       3 | Java     |        3 |
|       4 | Cpp      |        3 |
+---------+----------+----------+
4 rows in set (0.00 sec)
```

8. **Implement Many to Many Mapping between Author and Book.**

<AuthorMany.java>

```
package com.springdataaccessandtransactions.Assignment3.manytomany.entities;


import javax.persistence.*;
import java.util.List;

@Entity
@Table(name = "authormany")
```

```
public class AuthorMany {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  @Column(name = "author_id")
  private int id;
  private String name;

  @ManyToMany(cascade = CascadeType.ALL)
  @JoinTable(name = "author_book",
        joinColumns = @JoinColumn(name = "authorid", referencedColumnName =
"author_id"),
  inverseJoinColumns = @JoinColumn(name = "bookid", referencedColumnName =
"book_id"))
  private List<BookMany> books;

  public int getId() {
     return id;
  }

  public void setId(int id) {
     this.id = id;
  }

  public String getName() {
     return name;
  }

  public void setName(String name) {
     this.name = name;
  }

  public List<BookMany> getBooks() {
     return books;
  }

  public void setBooks(List<BookMany> books) {
     this.books = books;
  }
}
```

<BookMany.java>

```
package com.springdataaccessandtransactions.Assignment3.manytomany.entities;
```

```
import javax.persistence.*;
import java.util.List;

@Entity
@Table(name = "bookmany")
public class BookMany {

  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  @Column(name = "book_id")
  private int id;

  @Column(name = "bookname")
  private String bookName;

  @ManyToMany(mappedBy = "books")
  private List<AuthorMany> authors;

  public int getId() {
     return id;
  }

  public void setId(int id) {
     this.id = id;
  }

  public String getBookName() {
     return bookName;
  }

  public void setBookName(String bookName) {
     this.bookName = bookName;
  }

  public List<AuthorMany> getAuthors() {
     return authors;
  }

  public void setAuthors(List<AuthorMany> authors) {
     this.authors = authors;
  }
}
```

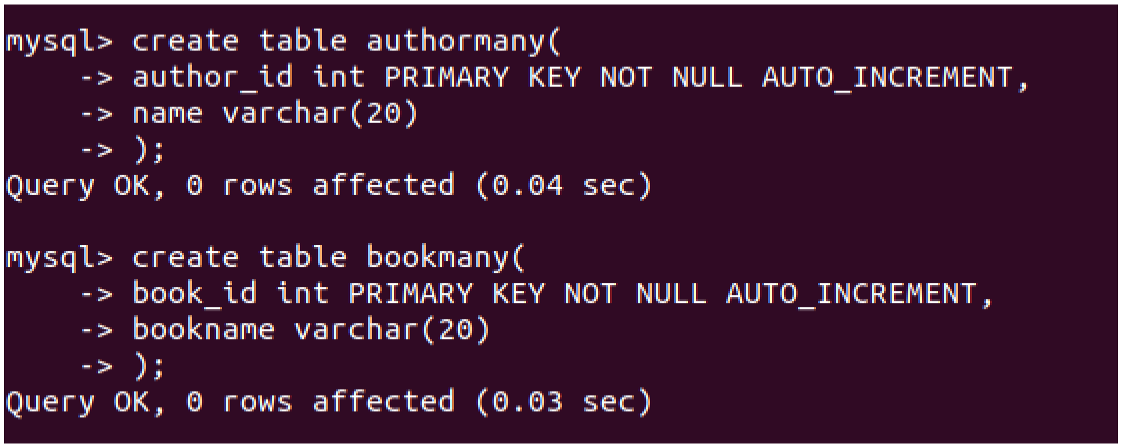
<AuthorManyRepository.java>

```
package com.springdataaccessandtransactions.Assignment3.manytomany.repos;

import
com.springdataaccessandtransactions.Assignment3.manytomany.entities.AuthorMany;
import org.springframework.data.repository.CrudRepository;

public interface AuthorManyRepository extends CrudRepository<AuthorMany,Integer> {

}
```

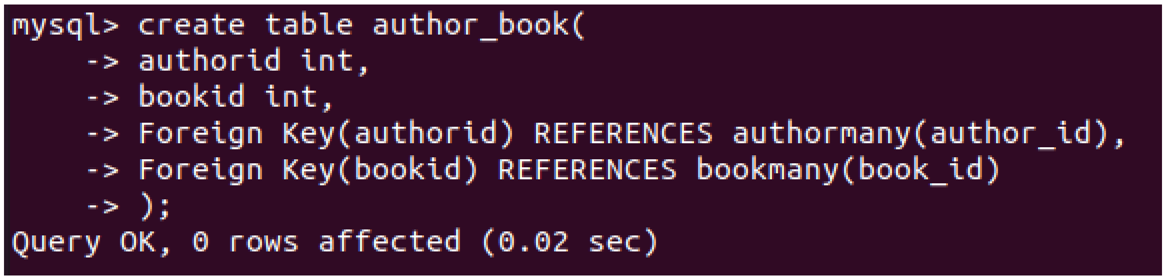
<ManyToManyTests.java>

```
package com.springdataaccessandtransactions.Assignment3.manytomanytests;


import
com.springdataaccessandtransactions.Assignment3.manytomany.entities.AuthorMany;
import
com.springdataaccessandtransactions.Assignment3.manytomany.entities.BookMany;
import
com.springdataaccessandtransactions.Assignment3.manytomany.repos.AuthorManyRep
ository;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit4.SpringRunner;

import java.util.ArrayList;
import java.util.List;

@RunWith(SpringRunner.class)
@SpringBootTest
public class ManyToManyTests {

  @Autowired
  AuthorManyRepository authorManyRepository;

  @Test
  public void TestManyToManyCreateData(){
     AuthorMany authorMany = new AuthorMany();
```

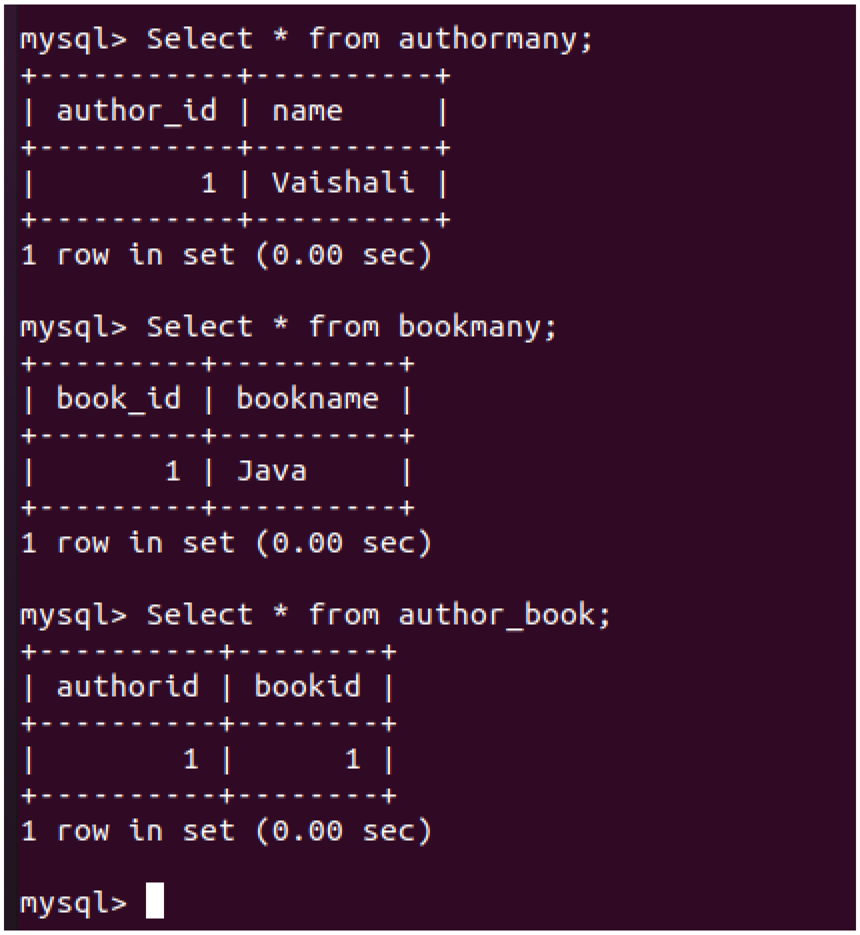
```
        authorMany.setName("Vaishali");

        List<BookMany> books = new ArrayList<>();
        BookMany book1 = new BookMany();
        book1.setBookName("Java");
        books.add(book1);

        authorMany.setBooks(books);
        authorManyRepository.save(authorMany);
    }
}
```



```
mysql> create table authormany(
    -> author_id int PRIMARY KEY NOT NULL AUTO_INCREMENT,
    -> name varchar(20)
    -> );
Query OK, 0 rows affected (0.04 sec)

mysql> create table bookmany(
    -> book_id int PRIMARY KEY NOT NULL AUTO_INCREMENT,
    -> bookname varchar(20)
    -> );
Query OK, 0 rows affected (0.03 sec)
```



```
mysql> create table author_book(
    -> authorid int,
    -> bookid int,
    -> Foreign Key(authorid) REFERENCES authormany(author_id),
    -> Foreign Key(bookid) REFERENCES bookmany(book_id)
    -> );
Query OK, 0 rows affected (0.02 sec)
```



```
mysql> Select * from authormany;
+-----------+----------+
| author_id | name     |
+-----------+----------+
|         1 | Vaishali |
+-----------+----------+
1 row in set (0.00 sec)

mysql> Select * from bookmany;
+---------+----------+
| book_id | bookname |
+---------+----------+
|       1 | Java     |
+---------+----------+
1 row in set (0.00 sec)

mysql> Select * from author_book;
+----------+--------+
| authorid | bookid |
+----------+--------+
|        1 |      1 |
+----------+--------+
1 row in set (0.00 sec)

mysql>
```

9. **Which method on the session object can be used to remove an object from the cache?**

   1) evict() is used to remove a particular object from cache associated with a session.

   2) You can not disable the first level cache, it is always enabled

   3) Hibernate entities or database rows remain in cache only until session is open,once session is closed, all associated cached data is lost.

   <ApplicationTest.java>

   @Autowired
   EntityManager entityManager;

```
@Test
@Transactional
public void testCaching(){
  Session session = entityManager.unwrap(Session.class);
  Author author = authorRepository.findById(1).get();
  authorRepository.findById(1).get();     // query will not be executed due to caching
  session.evict(author);                  //removal of cached object
  authorRepository.findById(1).get();    //query will be executed
}
```

10. What does @transactional annotation do?

The @Transactional annotation is the metadata that specifies the semantics of the transactions on a method. We have two ways to rollback a transaction: declarative and programmatic.

In the declarative approach, we annotate the methods with the @Transactional annotation. The @Transactional annotation makes use of the attributes rollbackFor or rollbackForClassName to rollback the transactions, and the attributes noRollbackFor or noRollbackForClassName to avoid rollback on listed exceptions.

The default rollback behavior in the declarative approach will rollback on runtime exceptions.

<BankAccountServiceImpl.java>

```
@Service
public class BankAccountServiceImpl implements BankAccountService{

  @Autowired
  BankAccountRepository bankAccountRepository;

  @Override
  @Transactional //if exception occurs, everything will be rollbacked, nothing will be
committed
  public void transfer(int amount) {
     BankAccount vaishaliAccount = bankAccountRepository.findById(1).get();
     vaishaliAccount.setBal(vaishaliAccount.getBal() - amount);
     bankAccountRepository.save(vaishaliAccount);
```

```
        if (true){
            throw new RuntimeException();
        }

        BankAccount nidhiAccount = bankAccountRepository.findById(2).get();
        nidhiAccount.setBal(nidhiAccount.getBal() + amount);
        bankAccountRepository.save(nidhiAccount);
    }
}
```