**Activity 13**

For this activity, use the music database created in Activity 11.

Contents of table Albums

+----+----------------+-----------+------+

| id | title | artist | year |

+----+----------------+-----------+------+

| 1 | Roots | Sepultura | 1996 |

| 2 | Morbid Visions | Sepultura | 1986 |

+----+----------------+-----------+------+

Contents of table Tracks

+----+-----+--------------------+

| id | num | name |

+----+-----+--------------------+

| 1 | 1 | Roots Bloody Roots |

| 1 | 2 | Attitude |

| 1 | 3 | Ratamahatta |

| 2 | 1 | Morbid Visions |

| 2 | 2 | Mayhem |

+----+-----+--------------------+

You might need to create and grant access to the following user:

|  |
| --- |
| CREATE USER "music\_admin" PASSWORD '135791';  GRANT ALL ON TABLE Albums, Tracks TO "music\_admin"; |

Step 1: Create a Maven project with dependencies

|  |
| --- |
| <dependencies>  <!-- https://mvnrepository.com/artifact/org.postgresql/postgresql -->  <dependency>  <groupId>org.postgresql</groupId>  <artifactId>postgresql</artifactId>  <version>42.3.3</version>  </dependency>  <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->  <dependency>  <groupId>org.hibernate</groupId>  <artifactId>hibernate-core</artifactId>  <version>5.6.7.Final</version>  </dependency> </dependencies> |

Step 2: Create a persistence.xml in resources/META-INF

|  |
| --- |
| *<?*xml version="1.0" encoding="UTF-8" standalone="yes"*?>* *<*persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.1"  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence\_2\_1.xsd">  <persistence-unit name="activity-13">  <description>Hibernate Tips</description>  <exclude-unlisted-classes>false</exclude-unlisted-classes>   <properties>  <property name="hibernate.dialect" value="org.hibernate.dialect.PostgreSQL10Dialect" />  <property name="hibernate.show\_sql" value="true" />  <property name="hibernate.hbm2ddl.auto" value="update" />  <property name="javax.persistence.jdbc.driver" value="org.postgresql.Driver" />  <property name="javax.persistence.jdbc.url" value="jdbc:postgresql://localhost:5432/music" />  <property name="javax.persistence.jdbc.user" value="music\_admin" />  <property name="javax.persistence.jdbc.password" value="135791" />  </properties>  </persistence-unit> </persistence> |

Step 3: Create the TrackPK Entity

|  |
| --- |
| import java.io.Serializable;  public class TrackPK implements Serializable {   private int id;  private int num;   public int getId() {  return id;  }   public void setId(int id) {  this.id = id;  }   public int getNum() {  return num;  }   public void setNum(int num) {  this.num = num;  }   @Override  public String toString() {  return "TrackPK{" +  "id=" + id +  ", num=" + num +  '}';  } } |

Step 4: Create the Track Entity

|  |
| --- |
| import javax.persistence.EmbeddedId; import javax.persistence.Entity; import javax.persistence.Table; import java.io.Serializable;  **@Entity** **@Table(name="Tracks"**) public class Track implements Serializable {   **@EmbeddedId**  private TrackPK trackPK;   private String name;   public TrackPK getTrackPK() {  return trackPK;  }   public void setTrackPK(TrackPK trackPK) {  this.trackPK = trackPK;  }   public String getName() {  return name;  }   public void setName(String name) {  this.name = name;  }   @Override  public String toString() {  return "Track{" +  "trackPK=" + trackPK +  ", name='" + name + '\'' +  '}';  } } |

Step 5: Create the Album Entity

|  |
| --- |
| import javax.persistence.\*;  import java.util.List;    **@Entity**  **@Table(name="Albums")**  public class Album {    **@Id**  private int id;    private String title;    private String artist;    private int year;    **@OneToMany(targetEntity = Track.class)**  **@JoinColumn(name="id")**  private List<Track> tracks;    public int getId() {  return id;  }    public void setId(int id) {  this.id = id;  }    public String getTitle() {  return title;  }    public void setTitle(String title) {  this.title = title;  }    public String getArtist() {  return artist;  }    public void setArtist(String artist) {  this.artist = artist;  }    public int getYear() {  return year;  }    public void setYear(int year) {  this.year = year;  }    public List<Track> getTracks() {  return tracks;  }    public void setTracks(List<Track> tracks) {  this.tracks = tracks;  }    @Override  public String toString() {  return "Album{" +  "id=" + id +  ", title='" + title + '\'' +  ", artist='" + artist + '\'' +  ", year=" + year +  ", tracks=" + tracks +  '}';  }  } |

Step 6: Create Main to illustrate JPQL

|  |
| --- |
| import javax.persistence.EntityManager;  import javax.persistence.EntityManagerFactory;  import javax.persistence.Persistence;  import javax.persistence.Query;    public class Main {    public static void main(String[] args) {    // Entity Manager initialization  EntityManagerFactory emf = Persistence.createEntityManagerFactory("**activity-13**");  EntityManager em = emf.createEntityManager();    // a query for all albums  Query query = em.createQuery("SELECT a FROM Album a");  for (Object obj: query.getResultList()) {  Album album = (Album) obj;  System.out.println(album);  }  }  } |

Desired output:

|  |
| --- |
| Album{id=1, title='Roots', artist='Sepultura', year=1996, tracks=[Track{trackPK=TrackPK{id=1, num=1}, name='Roots Bloody Roots'}, Track{trackPK=TrackPK{id=1, num=2}, name='Attitude'}, Track{trackPK=TrackPK{id=1, num=3}, name='Ratamahatta'}]}  Album{id=2, title='Morbid Visions', artist='Sepultura', year=1986, tracks=[Track{trackPK=TrackPK{id=2, num=1}, name='Morbid Visions'}, Track{trackPK=TrackPK{id=2, num=2}, name='Mayhem'}]} |

Do you prefer Python? Try to run the same example using SQLAlchemy (if you dare...).

|  |
| --- |
| from curses.ascii import EM  from sqlalchemy.ext.declarative import declarative\_base  from sqlalchemy import Column, String, Integer, create\_engine, ForeignKey  from sqlalchemy.orm import sessionmaker, relationship  Base = declarative\_base()  class Album(Base):  # table mapping  \_\_tablename\_\_ = 'albums'  # column mapping  id = Column(Integer, primary\_key=True)  title = Column(String)  artist = Column(String)  year = Column(Integer)  tracks = relationship("Track", primaryjoin="Album.id==Track.id")  def \_\_str\_\_(self):  s = str(self.id) + ", " + self.title + ", " + self.artist + ", " + str(self.year) + ", ["  for track in self.tracks:  s += "{ " + str(track.num) + ", " + track.name + " }, "  s = s[:-2] + ']'  return s  class Track(Base):  # table mapping  \_\_tablename\_\_ = 'tracks'  # column mapping  id = Column(Integer, ForeignKey("albums.id"), primary\_key=True)  num = Column(Integer, primary\_key=True)  name = Column(String)  def \_\_str\_\_(self):  return str(self.id) + ", " + str(self.num) + ", " + self.name  # db connection and session creation  db\_string = "postgresql://music\_admin:135791@localhost:5432/music"  db = create\_engine(db\_string)  Session = sessionmaker(db)  session = Session()  # simple search  albums = session.query(Album)  for alb in albums:  print(alb) |

Desired output:

|  |
| --- |
| 1, Roots, Sepultura, 1996, [{ 1, Roots Bloody Roots }, { 2, Attitude }, { 3, Ratamahatta }]  2, Morbid Visions, Sepultura, 1986, [{ 1, Morbid Visions }, { 2, Mayhem }] |