Product Table:

```
CREATE TABLE product (
   id int NOT NULL AUTO_INCREMENT,
   name varchar(255) NOT NULL,
   price decimal(10,2) NOT NULL,
   PRIMARY KEY (id)
);
```

Customer Table:

```
CREATE TABLE customer (
   id int NOT NULL AUTO_INCREMENT,
   name varchar(255) NOT NULL,
   address varchar(255) NOT NULL,
   PRIMARY KEY (id)
);
```

Manufacturer Table:

```
CREATE TABLE manufacturer (
   id int NOT NULL AUTO_INCREMENT,
   name varchar(255) NOT NULL,
   PRIMARY KEY (id)
);
```

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

    <session-factory>
        <!-- Database connection settings -->
        <property
name="connection.driver_class">com.mysql.jdbc.Driver</property>
        <property
name="connection.url">jdbc:mysql://localhost:3306/test</property>
        <property name="connection.username">root</property>
        <property name="connection.password"></property>

        <!-- JDBC connection pool (use the built-in) -->
        <property name="connection.pool_size">1</property>

        <!-- SQL dialect -->
        <property name="dialect">org.hibernate.dialect.MySQL5Dialect</property>

        <!-- Disable the second-level cache  -->
```

```
        <property
name="cache.provider_class">org.hibernate.cache.NoCacheProvider</property>

        <!-- Echo all executed SQL to stdout -->
        <property name="show_sql">true</property>

        <!-- Drop and re-create the database schema on startup -->
        <property name="hbm2ddl.auto">create</property>

        <!-- Names the annotated entity class -->
        <mapping class="Product"/>
        <mapping class="Customer"/>
        <mapping class="Manufacturer"/>
    </session-factory>

</hibernate-configuration>
```

i) insert a new record inside product table

```
Session session = sessionFactory.openSession();
Transaction tx = null;
try {
   tx = session.beginTransaction();
   Product product = new Product();
   product.setName("Product 1");
   Customer customer = new Customer();
```

```
  customer.setName("Customer 1");
  Manufacturer manufacturer = new Manufacturer();
  manufacturer.setName("Manufacturer 1");
  product.setCustomer(customer);
  product.setManufacturer(manufacturer);
  session.save(product);
  tx.commit();
}
catch (HibernateException e) {
  if (tx!=null) tx.rollback();
  e.printStackTrace();
}
finally {
  session.close();
}
```

ii) update an existing record in product table

```
Session session = sessionFactory.openSession();
Transaction tx = null;
try {
  tx = session.beginTransaction();
  Product product = (Product)session.get(Product.class, 1);
  product.setName("Updated Product 1");
  session.update(product);
  tx.commit();
```

```
}
catch (HibernateException e) {
  if (tx!=null) tx.rollback();
  e.printStackTrace();
}
finally {
  session.close();
}
```

iii) delete an existing record from product table

```
Session session = sessionFactory.openSession();
Transaction tx = null;
try {
  tx = session.beginTransaction();
  Product product = (Product)session.get(Product.class, 1);
  session.delete(product);
  tx.commit();
}
catch (HibernateException e) {
  if (tx!=null) tx.rollback();
  e.printStackTrace();
}
finally {
  session.close();
}
```

iv) execute an query and return the manufacturer list for a given product name.

```
Session session = sessionFactory.openSession();
Transaction tx = null;
try {
  tx = session.beginTransaction();
  List manufacturers = session.createQuery("FROM Manufacturer m WHERE m.name = :name").setParameter("name", "Manufacturer 1").list();
  for (Iterator iterator = manufacturers.iterator(); iterator.hasNext();){
    Manufacturer manufacturer = (Manufacturer) iterator.next();
    System.out.println(manufacturer.getName());
  }
  tx.commit();
}
catch (HibernateException e) {
  if (tx!=null) tx.rollback();
  e.printStackTrace();
}
finally {
  session.close();
}
```