Object Relational Mapping with JPA

General part

- **Explain the rationale behind the topic Object Relational Mapping and the Pros and Cons in using ORM**

  ORM er en programmeringsteknik til at konvertere data mellem objekter i et programmeringssprog og en relationel database.
  - Fordele:
    - Skrive programmerings logik i stedet for SQL (mindre kode skrivning)
    - Beskyttet mod MYSQL injections (vigtigt: valider stadig input)
  - Ulemper:
    - Stor afhængighed af ORM software kan resultere i mindre gode databaser

- **Explain the JPA strategy for handling Object Relational Mapping and important classes/annotations involved**

  Java Persistence API er en Java standard, til at mappe Java objekter i en database. Java Persistence API er en specifikation, og ikke en implementation, som er med til at skabe forbindelse mellem en relationel database og objektorienteret programmering. Java objekter i kontekst med en relationel databases kaldes for ”entities”, det er disse ”entities” der udgør rækker og koloner i databasen.

  ”**Entities” kan mappes med forskellige objekt-relationer:**
  - One-To-One
  - One-To-Many
  - Many-To-Many

  Alle annotationer kan implementeres enten som ”unidirectional” eller ”bidirectional”. Disse defineres i ”entities”.

  De vigtigste klasser i JPA er:
  - EntityManager: anvendes til at håndtere data (entities) til og fra databasen
  - EntityManagerFactory: anvendes til at oprette en EntityManager og sikre at den har den rigtige tilstand)
  - Persistence.xml: en konfigurations fil, der indeholder annoteringerne til at oprette forbindelse til databasen. Persistence.xml indeholder også en Persistence unit (pu), som specificerer persistence framework – dette er i vores tilfælde EclipseLink, samt en liste af pågældende entities i projektet.

- **Outline some of the fundamental differences in Database handling using plain JDBC versus JPA**

JDBC er en standard til at skabe en forbindelse direkte til en database, mens JPA er en standard for ”object-relational-mapping” (ORM). Således er JPA en højere standard når det gælder integration med databaser.
JBDC er en API som integrerer med en database ved brug af ren SQL, den har ingen objekt-annoteringer, således er det op til udvikleren selv at oversætte og håndtere ”resultSet” fra databasen.
JPA tillader udvikleren at mappe mellem objekter i koden og databasens tabeller, således at udvikleren kun skal forholde sig til Java-klasser og ikke SQL-sætninger.

Practical part

- **Examine and understand the diagram**

Det kan ses på databasens tabeller at en ”customer” og ”order” har en ”One-To-Many”-relation, det vil altså sige at en ”customer”, kan have mange ”order”, mens at en ordre kun har en ”customer”. Dernæst har order og orderline en tilsvarende relation, hvilket betyder at en ordre kan have mange ”orderlines”, mens at en orderline kun kan være tilknyttet en order. En ”orderline” og ”itemtype” har også en tilsvarende relation, hvilket vil sige at en ”itemtype” kan have mange ”orderlines”, men en orderline kun kan have en ”itemtype”.