**TASK 1-Reflection:**

**1) How is the ORM approach to storing data in databases more efficient than the conventional approach?**

**Ans)** Compared to conventional approach of storing data in databases ORM reduces the amount of code that needs to be written. ORM make the code easier to update, maintain and reuse as the developer can think of, and manipulate data as objects.

**2) How does SQLAlchemy handle when the schema changes from one version to the other?**

**Ans)** By using SQLAlchemy-migrate SQLAlchemy will handle when the schema changes from one version to another version

**TASK 2-Reflection:**

**1) Which method in SQLAlchemy is used to save a record to the database?**

**Ans)** db.session.add() is used to save a record to the database.

**2) How does the database commit work with SQLAlchemy?**

**Ans)** When we say session.commit(), it means that we have done making changes, add these changes to the database.

**3) Is there a way to rollback? How?**

**Ans)** Yes we can rollback. By using session.rollback() we can rollback.

**4) OOPS, passwords are stored in plain text with current approach, which clearly isn’t a secure practice. What are the best practices in storing passwords? Does SQLAlchemy provide any options to store passwords?**

**Ans)** The best practice to store password is hashing the password. The most vetted hashing algorithm providing most security is **bcrypt**. Yes, SQLAlchemy provide an option to store password. By using hashing technique we can store passwords to database by using SQLAlchemy.

**TASK 3-Reeflection:**

**1) How does the querying work with the SQLAlchemy?**

**Ans)** SQLAlchemy provides a **query** attribute on our model. When we access it we will get back a new query object over all records. Then we can use methods like **filter()** to filter the records before we use the select with **all()** or **first()**. There are different methods as well. Some of them are **order\_by()**, **join()**, **get()**, **count()**, etc…