ORM WITH SQLAlchemy - Reflection

Task - 1 :

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

  Compared to traditional techniques of exchange between an object-oriented language and a relational database, ORM often reduces the amount of code that needs to be written and it also helps us to handle the database like we handle with the objects. Disadvantages of ORM tools generally stem from the high level of abstraction obscuring what is actually happening in the implementation code. Also, heavy reliance on ORM software has been cited as a major factor in producing poorly designed databases.

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

  SQLAlchemy Migrate provides a way to deal with database schema changes in SQLAlchemy projects.SQLAlchemy Migrate is split into two parts, database schema versioning (migrate.versioning) and database migration management (migrate.changeset). The versioning API is available as the migrate command.

Task - 2 :

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

  Creating a python object and passing it as a parameter to db.session.add(obj). After adding, commit the changes.

- **How does the database commit work with SQLAlchemy?**

  Conceptually db.session.commit() is basically 2 steps in one:
  Flushing - push changes from SQLAlchemy's in-memory representation to the database transaction buffer
  Commit - persist changes from your database's transaction buffer into the database, i.e. inserting / updating / deleting.

- **Is there a way to rollback? How?**

  Yes, we can rollback the changes in the current session using session.rollback(). This method restores the database to the previous state by cancelling a specific transaction.

- **OOPS, passwords are stored in plain text with the 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?**

  Yes, sqlalchemy provides support for storing passwords in a secure manner with the help of bcrypt module.

Task - 3 :

- **How does the querying work with SQLAlchemy?**

  Flask-SQLAlchemy provides a query attribute on your Model class. When you access it you will get back a new query object over all records. You can then use methods like filter() to filter the records before you fire the select with all() or first() .