# ORM using `sqlalchemy`

**Object Relational Mapping (ORM) is a technique used in creating a "bridge" between object-oriented programs and, in most cases, relational databases**

Object-Relational Mapping is a technique for converting data between two incompatible type systems, in this case, Python and SQL.

ORM allows us to query and manipulate data from a database through an object-oriented programming language.

## Advantages of Using ORM Tools

Here are some of the advantages of using an ORM tool:

- It provides developers flexibility with database systems
- ORM library acts as bridge between database systems and python pograms
- ORM allows flexibility to developers to change database system
- It speeds up development time for teams.
- Decreases the cost of development.
- Handles the logic required to interact with databases.
- Improves security. ORM tools are built to eliminate the possibility of SQL injection attacks.
- You write less code when using ORM tools than with SQL.

## Disadvantages of Using ORM Tools

- Learning how to use ORM tools can be time consuming.
- They are likely not going to perform better when very complex queries are involved.

- ORMs are generally slower than using SQL.

## How to write `DDL` in ORM?

```
"""

engine = create_engine("dialect+driver://user:password@host:port/db")

dialect :: mysql, postgres, oracle, windowsSQL, SQLLite
"""

from pymysql.err import Error
from sqlalchemy import create_engine

user = "adam"
password = "qwerty"
host = "127.0.0.1"
port = 3306
database = "starwarsDB"


def get_connection():

    try:
        engine = create_engine(
            "mysql+pymysql://{user}:{password}@{host}:{port}/{database}"
            "".format(
                user=user, password=password,
                host=host, port=int(port), database=database
            )
        )
    except Error as ex:
        print(f"[ ERROR ] details {ex}")

    return engine


if __name__ == "__main__":
    engine = get_connection()
    table_names = engine.table_names()
    print(table_names)
```

## How to write `DML` in ORM?

```
"""


# TO grant permissions to user
GRANT ALL PRIVILEGES ON starwarsDB. * TO adam@localhost;


"""

import sqlalchemy
from pymysql.err import Error
from sqlalchemy import create_engine
import sqlalchemy as db  # alias for sqlalchemy is `db`


user = "adam"
password = "qwerty"
host = "127.0.0.1"
port = 3306
database = "starwarsDB"


def get_connection():

    try:
        engine = create_engine(
            "mysql+pymysql://{user}:{password}@{host}:{port}/{database}"
            "".format(
                user=user, password=password,
                host=host, port=int(port), database=database
            )
        )
        meta_data_obj = db.MetaData()
    except Error as ex:
        print(f"[ ERROR ] details {ex}")

    return engine, meta_data_obj


if __name__ == "__main__":
    engine, meta_data_obj = get_connection()
    profile_ = db.Table(
        "profile",
        meta_data_obj,
        db.Column("name", db.String(250)),
        db.Column("email", db.String(250)),
        db.Column("contact", db.Integer)
    )

    from sqlalchemy.sql import select
    s = select(profile_)
```

```
    result = engine.execute(s)
```

## `Insert` operation

```
"""


"""


import sqlalchemy
from pymysql.err import Error
from sqlalchemy import create_engine
import sqlalchemy as db  # alias for sqlalchemy is `db`


user = "adam"
password = "qwerty"
host = "127.0.0.1"
port = 3306
database = "starwarsDB"


def get_connection():

    try:
        engine = create_engine(
            "mysql+pymysql://{user}:{password}@{host}:{port}/{database}"
            "".format(
                user=user, password=password,
                host=host, port=int(port), database=database
            )
        )

        meta_data_obj = db.MetaData()
    except Error as ex:
        print(f"[ ERROR ] details {ex}")

    return engine, meta_data_obj


if __name__ == "__main__":

    from sqlalchemy.sql import insert, values

    engine, meta_data_obj = get_connection()
    profile_ = db.Table(
```

```
        "profile",
        meta_data_obj,
        db.Column("name", db.String(250)),
        db.Column("email", db.String(250)),
        db.Column("contact", db.Integer)
    )
    stmt = (
        insert(profile_).
        values(name='Prashant', email='prashant@vctcpune.co.in')
    )
    engine.execute(stmt)
```