Django ORM

What is an ORM ?

With Django we don’t need SQL queries to create database tables.

But what about reading and writing data to the database ?

Usually we would have to write SQL queries to do so.

But Django provides an Object Relational Mapper which considers every database table entry as an individual object.

Using which we can access data inside of any database table which we want.

We could also perform other operations like reading, updating and deleting that data as well.

To get access to any database data, we first access the objects of that table’s model.

We then treat the table’s model name as class and access the objects out of it.

Example, if we have a model named Product, we create access the objects as follows

```
products = Product.objects.all()
```

This gives us access to all the entries in the database table named Product.

Here Product is the model name which we have used to create the product table.

In the above code, products is now a list of product that contains every single product.

Using the ORM, we can also access an individual product as well depending on its id.

```
product = Product.objects.get(id=1)
```

The above code saves a single product with an id 1 in the product variable.

We could also perform other operations like creating a product.

Suppose you want to create a new product, you do it as:

```
p1 = Product(name="iphone",price="999")
```

We first give our object some name, here we have named it p1.

We defined that p1 is a Product object.

We have passed the model fields that are required to create the product such as name and price.

The above code just creates the product object, and does not save the data into database table.

To actually save it, we say:

```
p1.save()
```

Similarly to delete it we would say.

```
p1.delete()
```