Topic 2 : Database

What is MVC (Model View Controller)

- *A Web Application Development Framework*
- *Model (M):*
  - *Part of the web application that retrieve data from the database.*
  - *This is normally written in backend programming language using ORM principle*
- *View (V):*
  - *Think of the UI Representation of a website*
  - *You will normally see HTML, CSS and even JS here (WAD 1)*
  - *Some framework, eg Django will have server rendering language.*
- *Controller (C):*
  - *Handle the logic of our application,*
  - *It will link the UI (View) to the database (Model)*
  - *It also perform form handling, authentication, validation, integration with other application etc.*

Django: models

- ORM
- Work with models in Django

What is Database ?

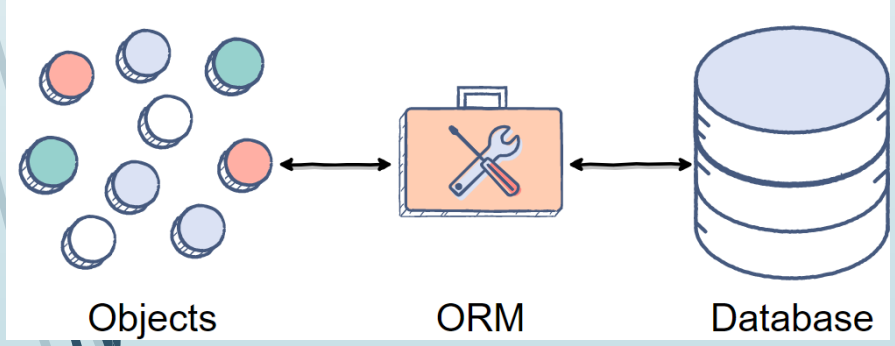
- **database is information that is set up for easy access, management and updating.**
- Databases are used for storing, maintaining and accessing any sort of data.
- That information is gathered in one place so that it can be observed and analyzed.
- Databases can be thought of as an organized collection of information.
- database is a systematic collection of data that support electronic storage and manipulation of data

Pinciples of ORM

ORM (Object Relational Mapping)

- Uses the concept of Object-Oriented Programming to organize database.
- By using ORM, we only need to know the **method** to be used in database instead of using query (**SQL**).
- Combines the benefits of OOP and RDBMS
  - Precludes the need for writing SQL
  - Generates and executes DDL and DML as needed
  - Provides flexibility w.r.t. mapping strategies

OBJECT RELATIONAL MAPPING (ORM)



- technique that creates a layer between the language and the database, helping programmers work with data without the OOP paradigm.
- provides an object-oriented layer between relational databases and object-oriented programming languages without having to write SQL queries.
- bridge between databases and object-oriented programming.
- The ORM equips user with object-oriented tools to run commands that user would usually run on databases.
- It masks out the complicated intricacies of the databases and lets user manipulate them with user own choice of programming language (must support object-oriented programming).

ORM vs Relational Database (1/2)

- Objects are first-class citizens: *"Think like an object"*
  - Relation / Table ⇔ Class
  - Record / Row / Tuple ⇔ Object
  - Attribute / Column ⇔ Properties/Member / Field
  - Relationship ⇔ Composition / Aggregation
  - Hierarchy (Is-A) ⇔ Inheritance

ORM vs Relational Database (2/2)

Relational	ORM
Table	Class
Column in table	Property / Attribute
Query	Method
Row in table	Object

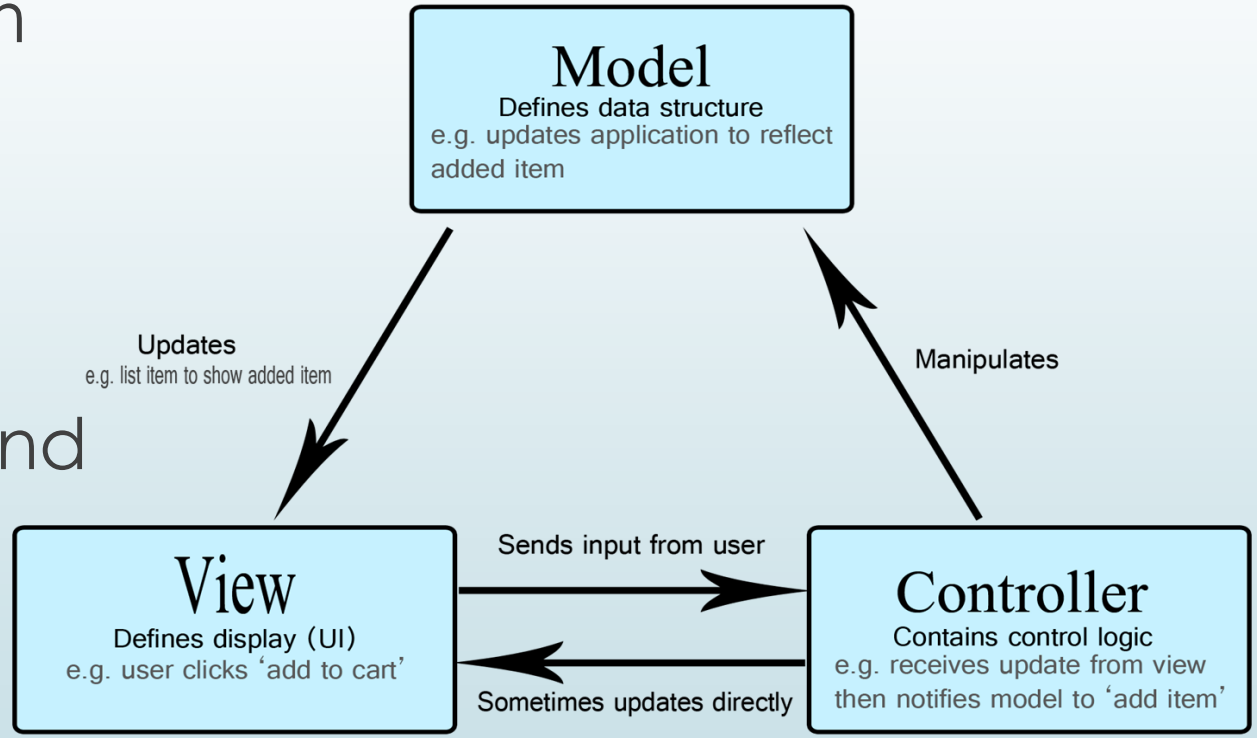
Model ?

- A model is the single, definitive source of information about data.
- It contains the essential fields and behaviors of the data storing.
- Generally, each model maps to a single database table.
- Models define the structure of stored data, including the field types and possibly also their maximum size, default values, selection list options, help text for documentation, label text for forms, etc.
- Model is independent of the underlying database. Just write our model structure and other code, and Django handles all the work of communicating with the database for us.

Model ?

- The model is the part of the Django code that will manage the data part. It has the following information:
  a) Table (table) in our project
  b) Attributes (columns) in each table, as well as the information is a maximum character example, if it is unique or not
  c) Relationship between tables (relationship)
- Usually a class has information about a table.

- **Model** (Connection to database, retrieving data from database)
- **View** - UI Representation
- **Controller** = Connect Model and View, Perform logic, business rules of the applicatiom



MVC flow

Routes … -> Routes dapatkan information, contohnya en-gb / hotels / hotel-ibis-kuala lumpur

Controller -> will get the data from database.

For example : SELECT * FROM hotels WHERE hotel_name = hotel-ibis-kuala-lumpur

Controller retrieve the data, add some logic , for example: get the reviews, get images

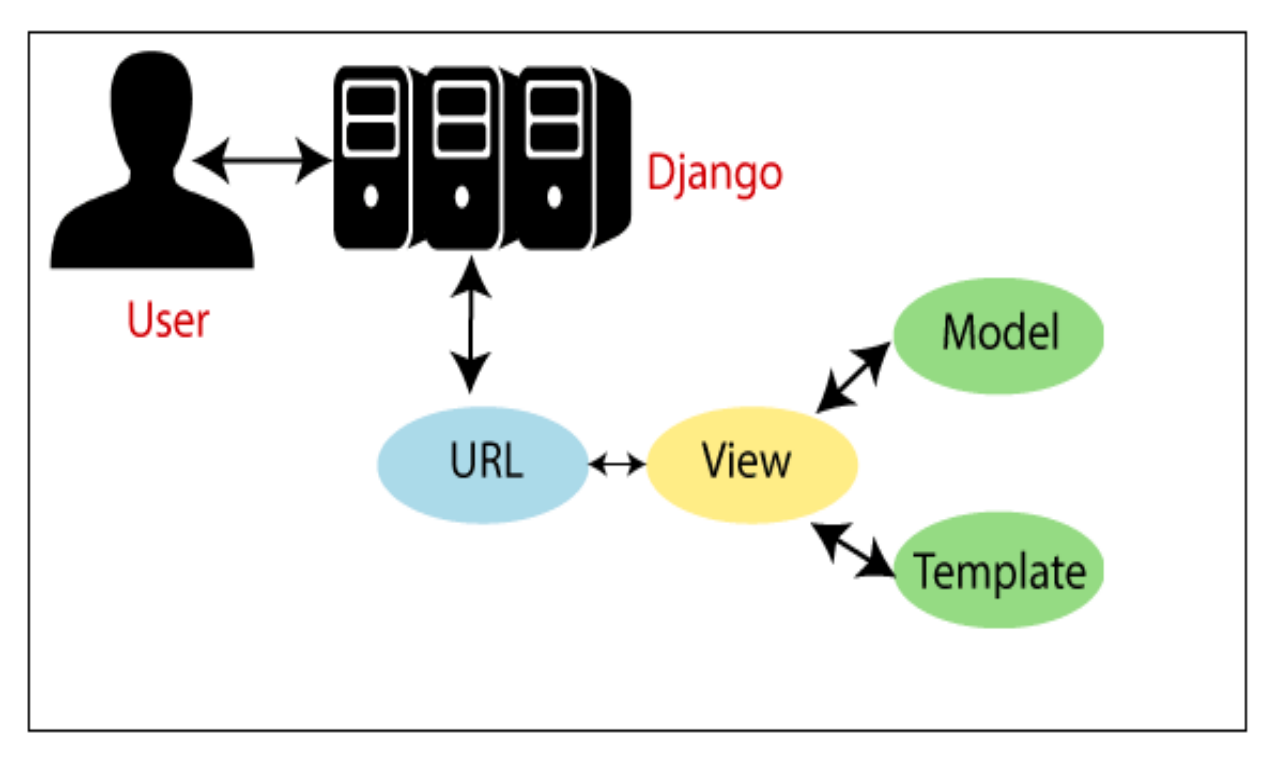
All is passed to View and View will render it in the UI

MVT (DJANGO FRAMEWORK)

Model - Everything that is related to Database (Model)

View -> Logic , business rules , connect model and template

Template - UI, Representation, layout



What is Model in Django?

Model is a part of Django code that will organize data and database.

It consist of:

a) Table in the project

b) Property / attribute / column:

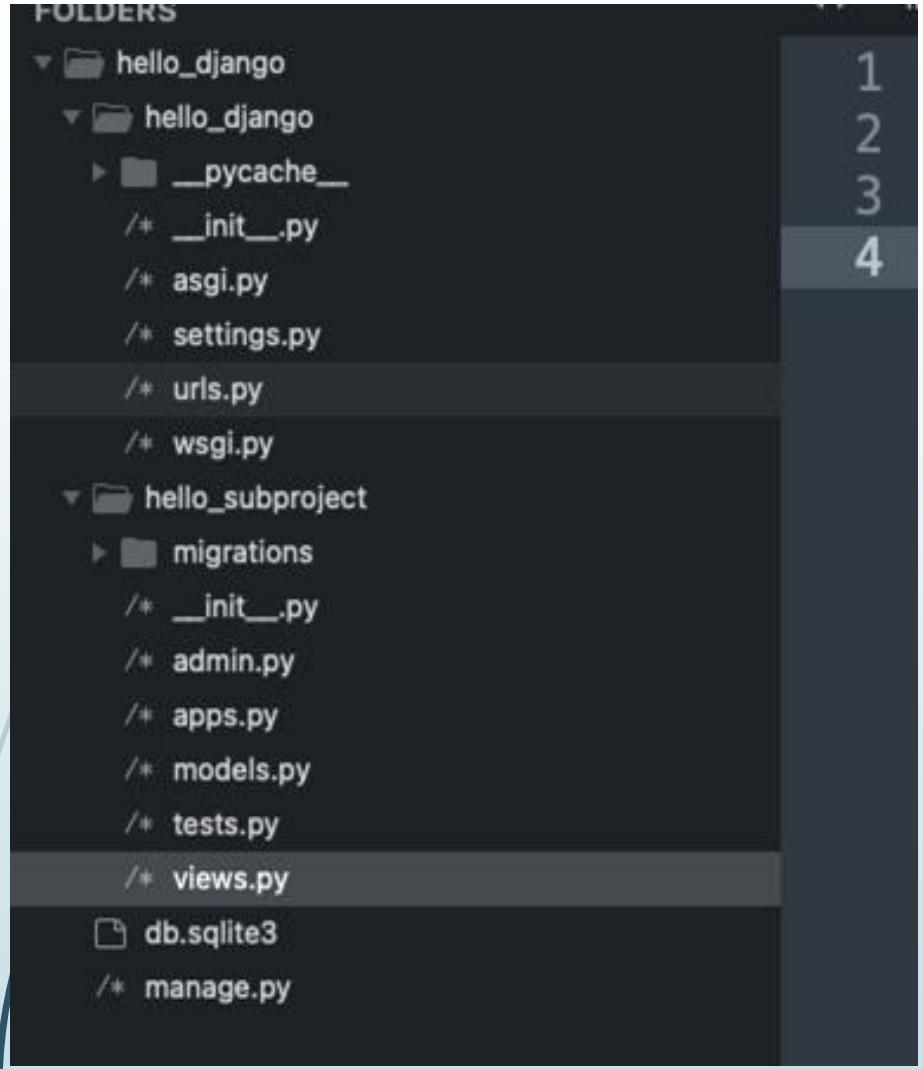
- data types
- maximum number of character (max field length)
- null / not null value

c) Relationship between tables

Model Field Types

- AutoField
- BigAutoField
- BigIntegerField
- BinaryField
- BooleanFieldCharField
- DateField (yyyy-mm-dd)
- DateTimeField
- DecimalField
- DurationField
- EmailField
- FileField
- FilePathField
- FloatField
- ImageField
- IntegerField
- GenericIPAddressField
- NullBooleanField
- PositiveIntegerField
- PositiveSmallIntegerField
- SlugField
- SmallIntegerField
- TextFleld
- TimeField
- URLField
- **ForeignKey**
- **Primary Key**
- **ManyToManyField**
- **OneToOneField**

https://docs.djangoproject.com/en/2.1/ref/models/fields/#field-types



Important files will be generated

models.py -> To define our model (data in database)

views.py -> Controller (logic of our application)

Defining a table

SQL:

```
CREATE TABLE Users(
  name VARCHAR(128),
  email VARCHAR(128)
);
```

models.py:

```
from django.db import models

Class User(models.Model):
    name = models.CharField(max_length=128)
    email = models.CharField(max_length=128)
```

How to create table

How to create table

1. Create project name **KPMB (django-admin startproject KPMB)**
2. Runserver
3. Create application **Registration (python manage.py startapp Registration)**
4. Open Visual code
5. Edit setting in main project (KPMB) - INSTALLED_APPS

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'Registration'
]
```

6. Save setting.py

How to create table

7. Edit models.py in subproject (Registration)
   - Create table Course with attribute
     - code (datatype = char, length=4, primary key)
     - description (datatype=text)

Go Run Terminal Help models.py - dcs4c - Visual Studio Code

settings.py | models.py | views.py | urls.py registration | urls.py dcs4c | index.html | __init__.py

registration > models.py

```python
from django.db import models

# Create your models here.
class Course(models.Model):
    coursecode=models.CharField(max_length=4, primary_key=True)
    coursename=models.TextField(max_length=128)
    coursedate=models.DateField(blank=True, null=True)
```

Go Run Terminal Help models.py - dcs4c - Visual Studio Code

settings.py | models.py | views.py | urls.py registration | urls.py dcs4c | index.html | __init__.py

registration > models.py

```python
from django.db import models

# Create your models here.
class Course(models.Model):
    coursecode=models.CharField(max_length=4, primary_key=True)
    coursename=models.TextField(max_length=128)
    coursedate=models.DateField(blank=True, null=True)
```

How to create table

7. Save models.py
8. Go to command prompt
   - python manage.py makemigrations

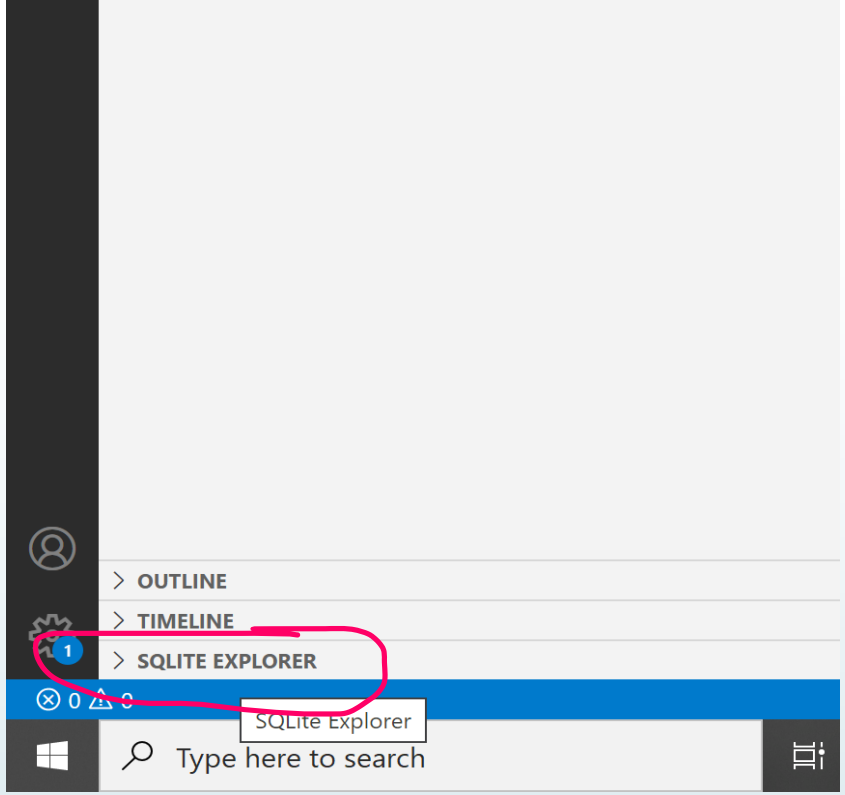
   (makemigrations is responsible for packaging up your model changes into individual migration files - analogous to commits.

   - Table Course is created in your project
   - python manage.py migrate (migrate executes those SQL commands in the database file.

   So after executing migrate all the tables of your installed apps are created in your database file.)

C:\Windows\System32\cmd.exe

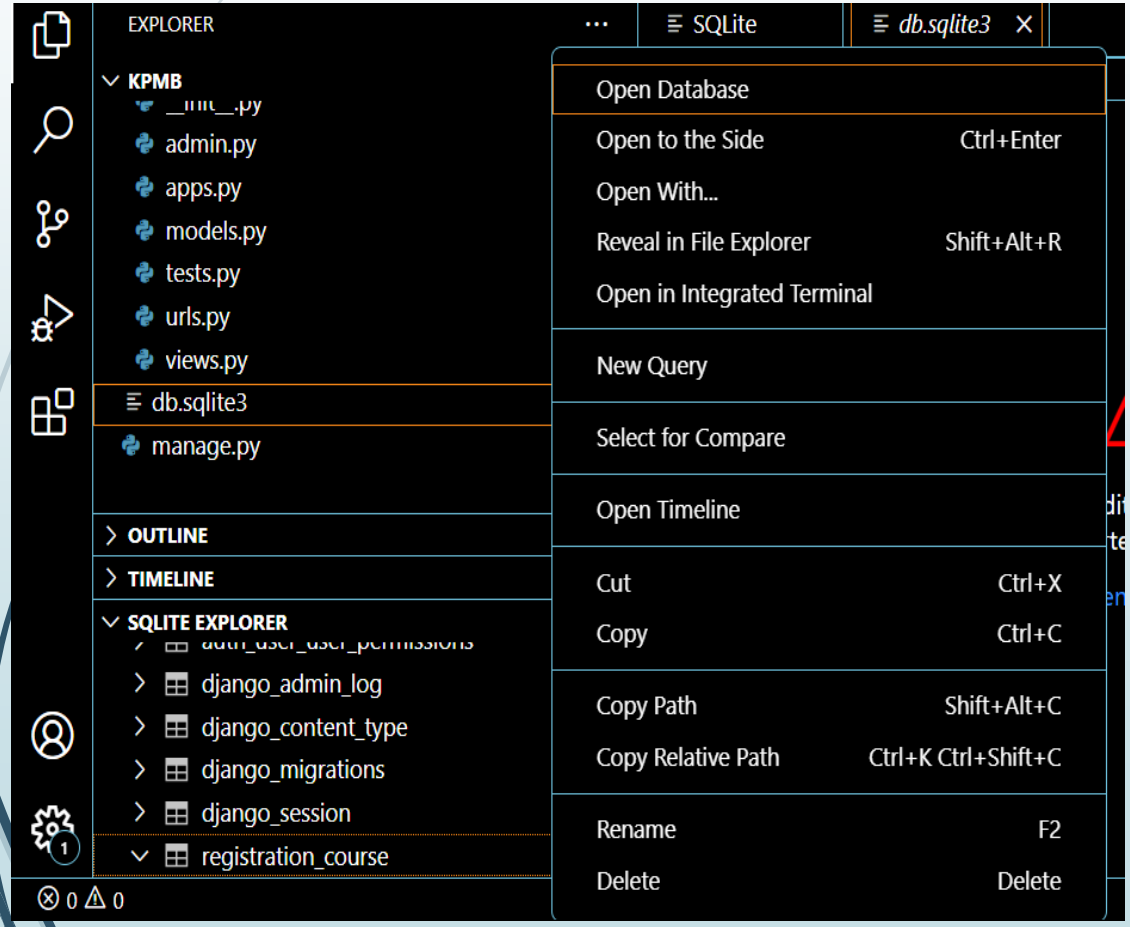
```
C:\Users\SK216988\Desktop\project_wad\KPMB>python manage.py makemigrations Registration
Migrations for 'Registration':
  Registration\migrations\0001_initial.py
    - Create model Course

C:\Users\SK216988\Desktop\project_wad\KPMB>
```


OUTLINE
TIMELINE
SQLITE EXPLORER
0 0
SQLite Explorer
Type here to search

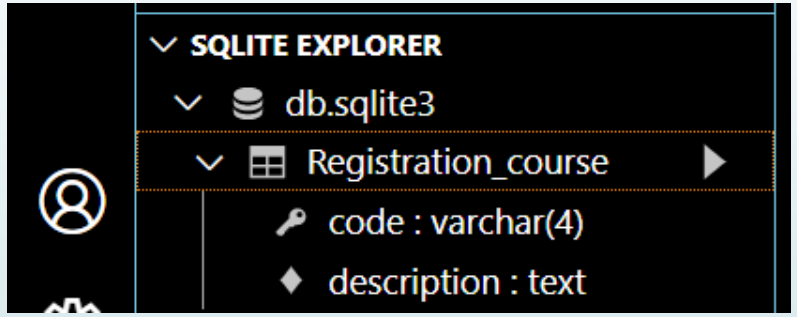
How to view table

10.Go to Visual code

- Right hand click at dbsqlite3 – open Database



11. Go to SQLITE EXPLORER – click Register_course



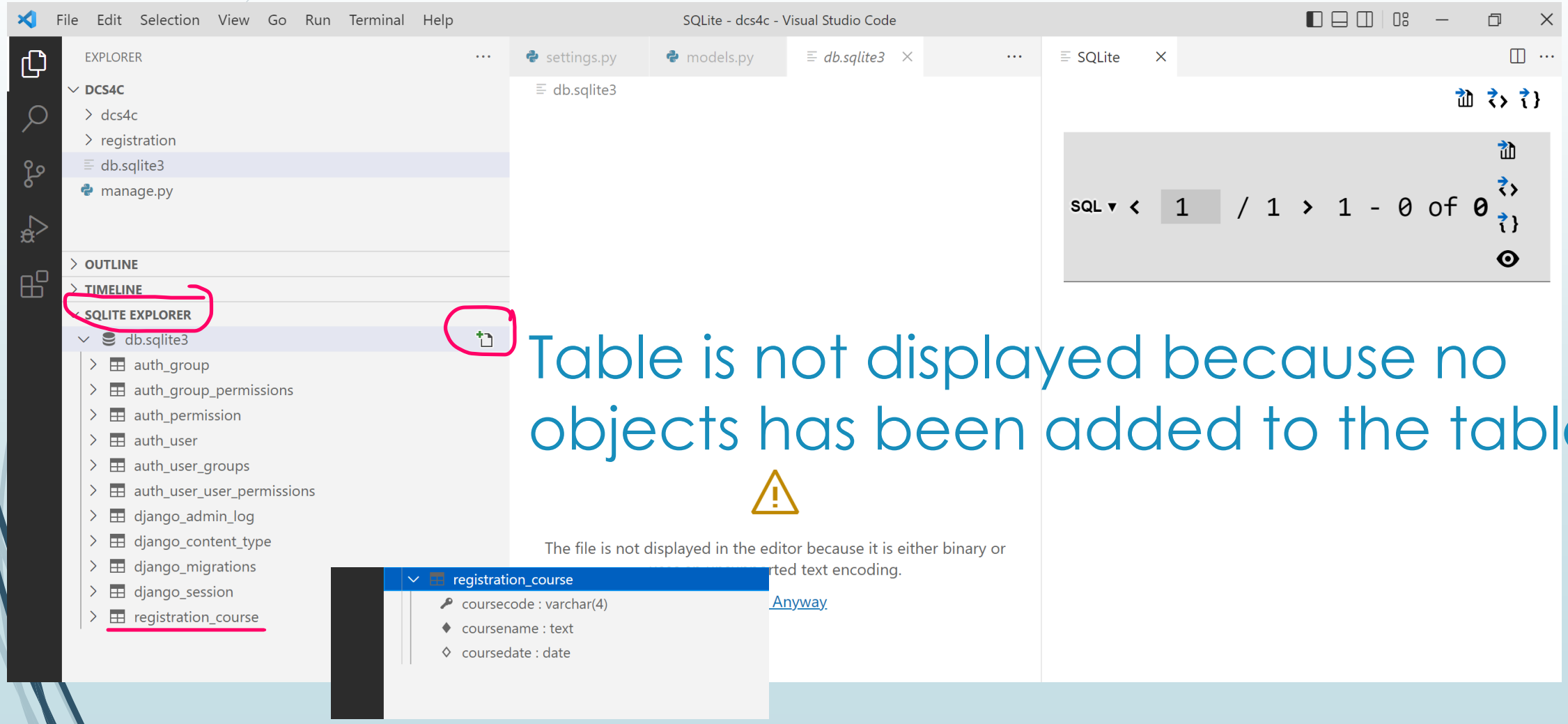
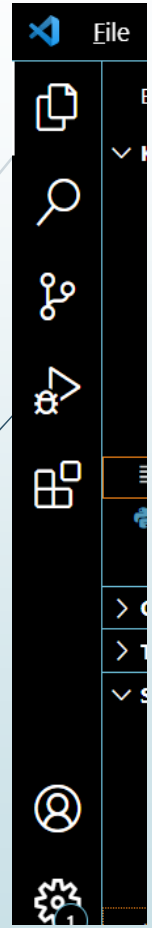


Table is not displayed because no objects has been added to the table

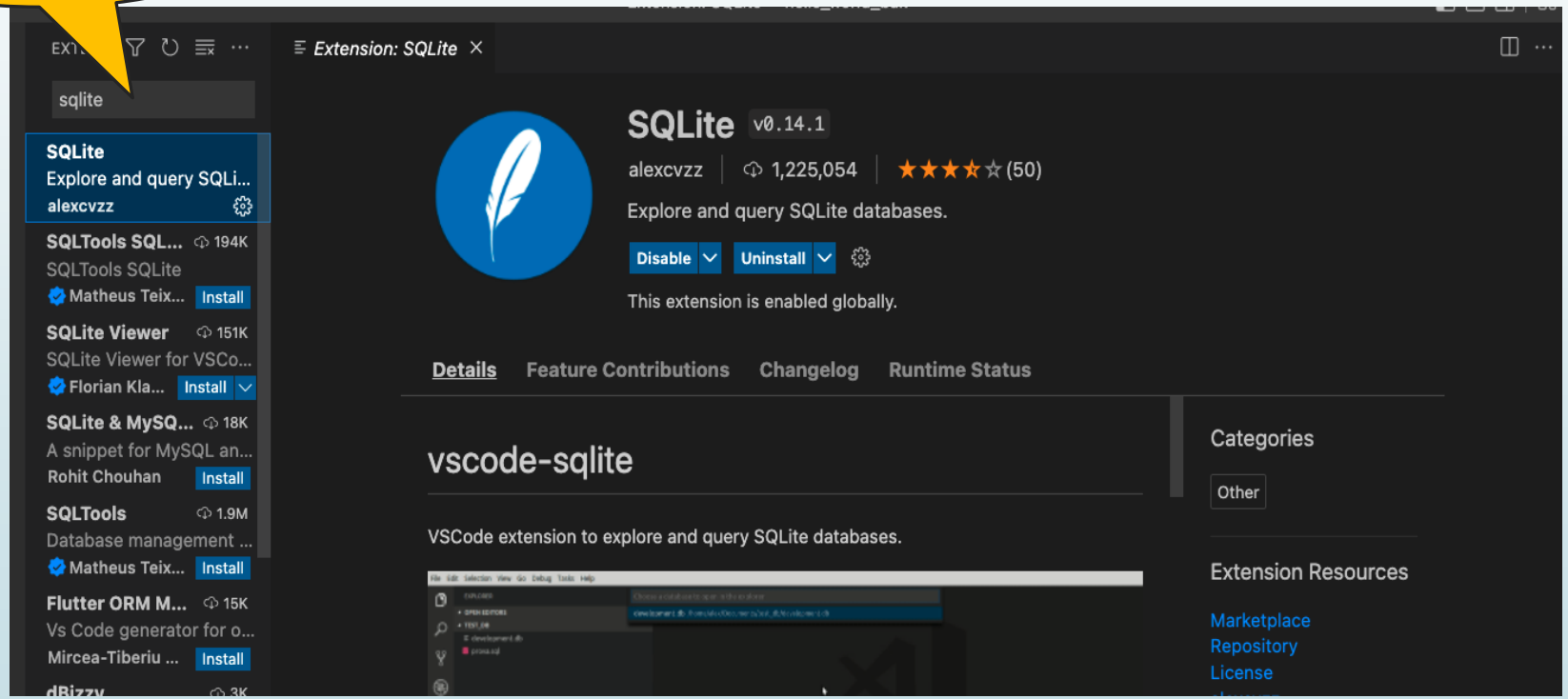
HOW TO ADD EXTENSION SQLITE (To view database in Visual Studio Code)

Add Extension in VSC



1. Click here

2. Search SQLite



GROUP TASK

1

- AutoField
- BigAutoField
- BigIntegerField
- BinaryField

2

- BooleanField
- NullBooleanField
- DateTimeField
- Time Field
- EmailField

3

- FileField
- FilePathField
- ImageField

4

- GenericIPAddressField

5

- PositiveIntegerField
- PositiveSmallIntegerField
- FloatField

6

- SlugField
- SmallIntegerField

7

- URLField
- DecimalField
- DurationField

8

- **ManyToManyField**
- **OneToOneField**

https://docs.djangoproject.com/en/2.1/ref/models/fields/#field-types

GROUP TASK

1) DESCRIBE WHEN TO USE THE MODEL FIELD TYPES. (WHAT ARE THE PROPERTIES INVOLVE)
2) GIVE EXAMPLES IN MODELS.PY HOW TO USE THE FIELD TYPES.
3) WHAT ARE THE CONSEQUENCES WHEN CHOOSING THE WRONG FIELD TYPES.

1. Class Mentor (mentorid (PK), mentorname, mentorphone)

1. Class Student(studentid-PK, studentname, coursecode-FK, mentorid-FK)

How to create table with primary key and foreign key

How to create table

1. Edit models.py in subproject (Registration)
   - Create table Student with attribute
     - studentid (datatype = char ,length=12, primary_key)
     - studentname (datatype = char ,length=100)
     - address (datatype = char ,length=150)
     - phone (datatype = char ,length=12)
     - course_code (ForeignKey)

```
class Student(models.Model):
    id = models.CharField(max_length=12, primary_key = True)
    name = models.CharField(max_length = 100)
    address = models.CharField(max_length = 150)
    phone = models.CharField(max_length = 12)
    course_code = models.ForeignKey(Course, on_delete = models.CASCADE)
```

2. Save models.py
3. Go to command prompt
   - python manage.py makemigrations
   - Table Student is created in your project
   - python manage.py migrate

C:\Windows\System32\cmd.exe - python manage.py shell

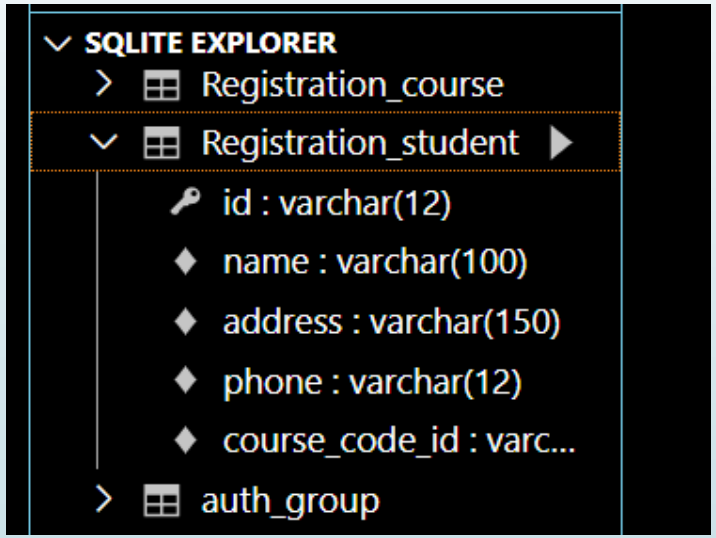
```
C:\Users\SK216988\Desktop\project_wad\KPMB>python manage.py makemigrations
Migrations for 'Registration':
  Registration\migrations\0002_student.py
    - Create model Student

C:\Users\SK216988\Desktop\project_wad\KPMB>python manage.py migrate
Operations to perform:
  Apply all migrations: Registration, admin, auth, contenttypes, sessions
Running migrations:
  Applying Registration.0002_student... OK
```

How to create table

10. Go to Visual code

- Right hand click at dbsqlite3 – open Database

11. Go to SQLITE EXPLORER – click Register_student



How to add data to table

How to add data to table

1. Go to command prompt
   - Use Python interpreter (Python shell) to add data in table
     - python manage.py shell

```
C:\Windows\System32\cmd.exe - python  manage.py shell
C:\Users\SK216988\Desktop\project_wad\KPMB>python manage.py shell
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>>
```

2. Import table in subproject models
   - from Registration.models import Course, Student

```
C:\Windows\System32\cmd.exe - python  manage.py shell
C:\Users\SK216988\Desktop\project_wad\KPMB>python manage.py shell
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from Registration.models import Course
>>>
```

3. Add data
   - data=Course('DCS', coursename='Diploma in Computer Science', coursedate='2019-11-29')
   - data.save( )

```
C:\Windows\System32\cmd.exe - python  manage.py shell
>>>
>>> data=Course('DCS', description='Diploma in Computer Science')
>>> data.save()
>>>
```

TASK

- Add new field in student as stumentor where this field is foreign key to menid.
- Add data to all tables.

How to add data to table with foreign key

1. Go to command prompt
   - Use Python interpreter (Python shell) to add data in table
     - python manage.py shell

```
C:\Windows\System32\cmd.exe - python  manage.py shell
C:\Users\SK216988\Desktop\project_wad\KPMB>python manage.py shell
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> _
```

2. Import table in subproject models
   - from Registration.models import Course, Student

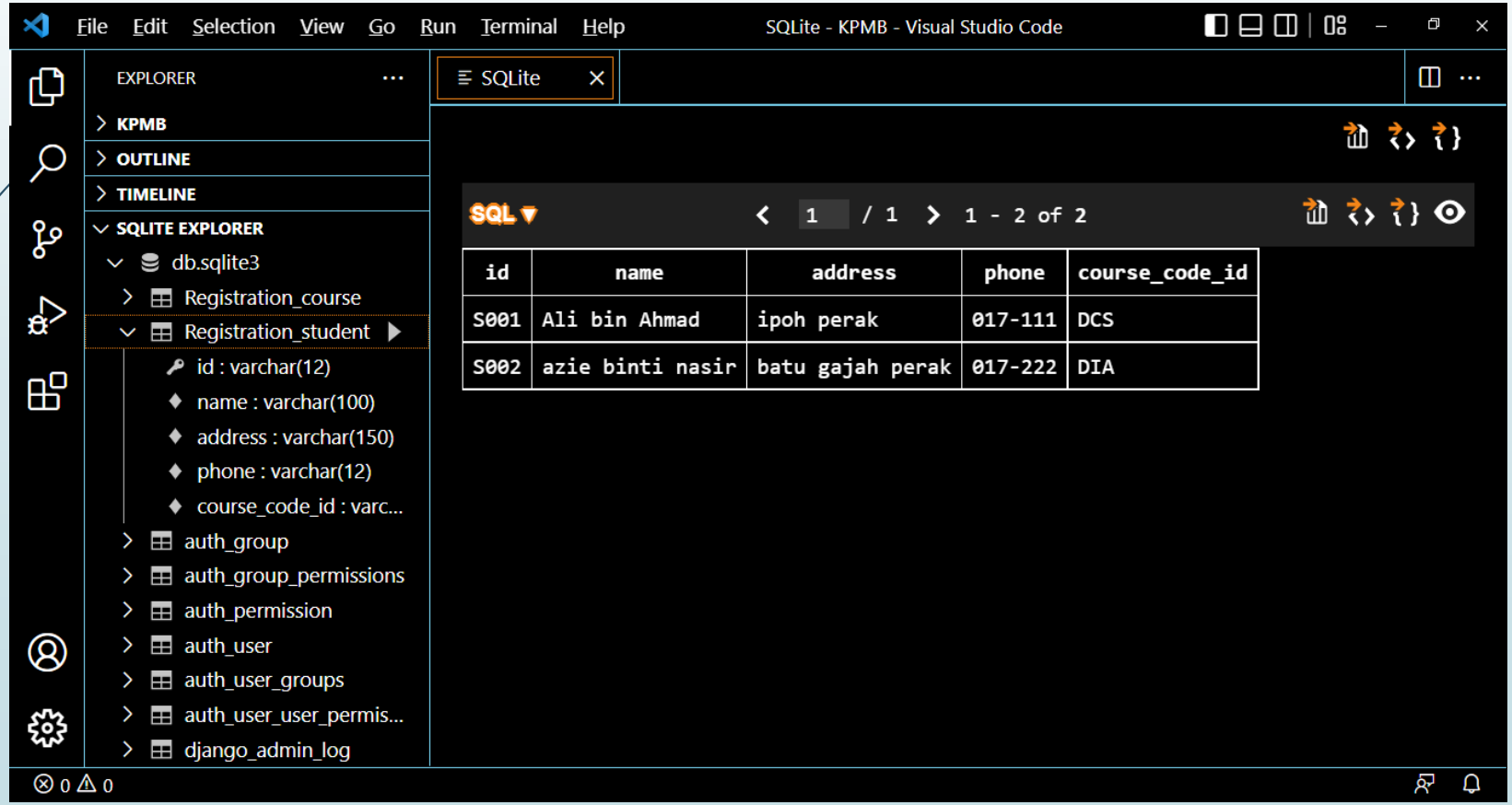
```
C:\Windows\System32\cmd.exe - python  manage.py shell
>>> from Registration.models import Course,Student
```

3. Add data
   - a=Course.objects.get(coursecode='DIA')
   - >>> data1 = Student (id='S002', name = 'azie binti nasir',address='batu gajah perak',phone='017-222', course_code=a)
   - >>> data1.save()

```
C:\Windows\System32\cmd.exe - python  manage.py shell
>>> a=Course.objects.get(code='DCS')
>>> a
<Course: Course object (DCS)>
>>> data1 = Student (id='S001', name = 'Ali bin Ahmad',address='ipoh perak',phone='017-111', course_code=a)
>>> data1.save();
>>> a=Course.objects.get(code='DIA')
>>> data1 = Student (id='S002', name = 'azie binti nasir',address='batu gajah perak',phone='017-222', course_code=a)
>>> data1.save()
>>> _
```

How to add data to table with foreign key (view)

4. Go to Visual Code
   - Go to SQLITE EXPLORER – click Register_student
   - Right hand click show tables



How to display data in table

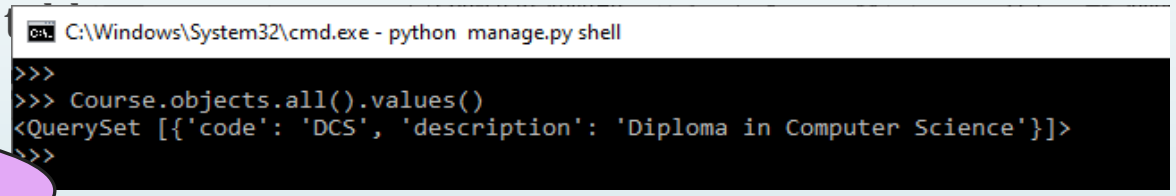
VIEW/DISPLAY data in database using CLI

By using interactive console (shell) in command prompt, (don't forget to import classes)

1. Display data

- Use Python interpreter (Python shell) to add data in t

  - Course.objects.all( ).values( )



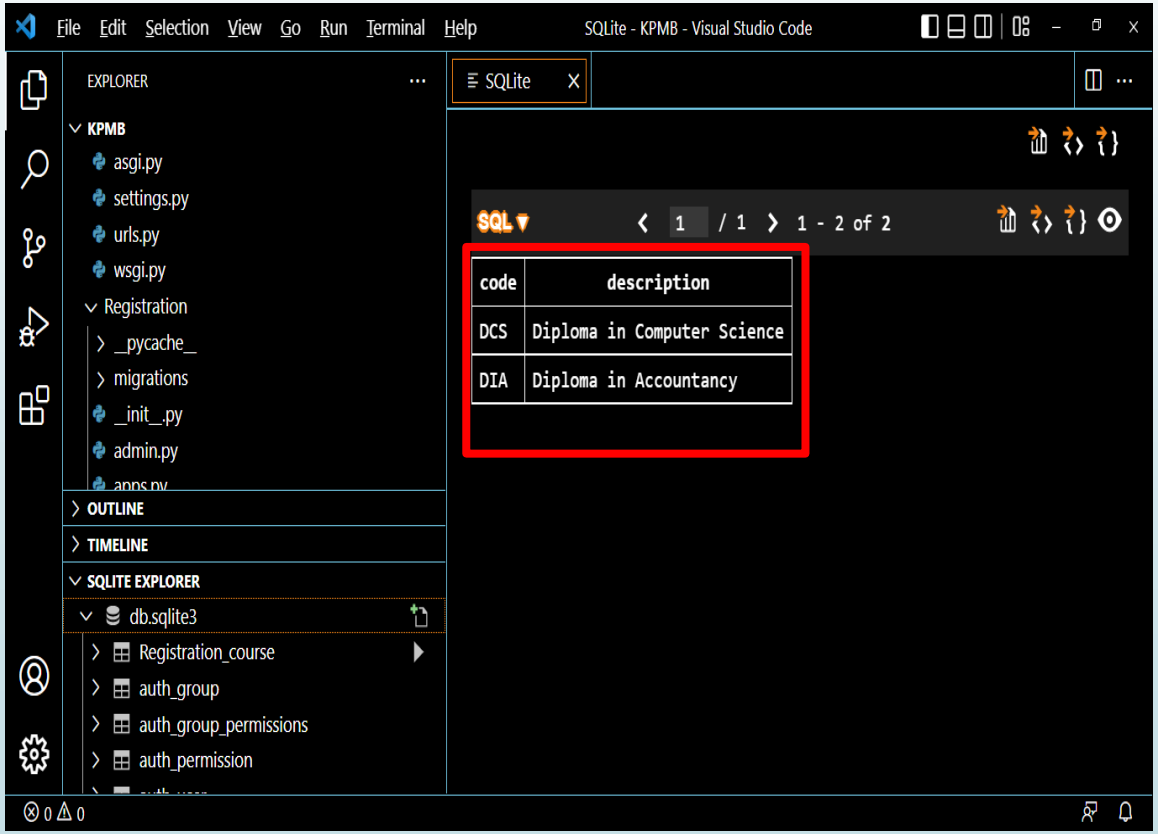
Display all objects in a table

  - Movie.objects.filter(id='1').values()

Display specific object in a table

VIEW/DISPLAY data in database using VSC

1. Go to Visual Code
   - Go to SQLITE EXPLORER – click Register_course
   - Right hand click show tables



How to delete data in table

How to delete data in table

1. Go to command prompt
   - Use Python interpreter (Python shell) to add data in table
     - python manage.py shell

```
C:\Windows\System32\cmd.exe - python  manage.py shell
C:\Users\SK216988\Desktop\project_wad\KPMB>python manage.py shell
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>>
```

2. Import table in subproject models
   - from Registration.models import Course

```
C:\Windows\System32\cmd.exe - python  manage.py shell
C:\Users\SK216988\Desktop\project_wad\KPMB>python manage.py shell
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from Registration.models import Course
>>>
```

3. delete data
   - Course.objects.get(code='DCS').delete()

```
C:\Windows\System32\cmd.exe - python  manage.py shell
>>>
>>> Course.objects.get(code='DCS').delete()
(1, {'Registration.Course': 1})
>>>
```

How to delete data in table

4. Display data
   - Use Python interpreter (Python shell) to delete data in table
     - Course.objects.all( ).values( )

C:\Windows\System32\cmd.exe - python manage.py shell

```
>>>
>>> Course.objects.get(code='DCS').delete()
(1, {'Registration.Course': 1})
>>> Course.objects.all().values()
<QuerySet []>
>>>
```

5. Go to Visual Code
   - Go to SQLITE EXPLORER – click Register_course
   - Right hand click show tables

How to update data in table

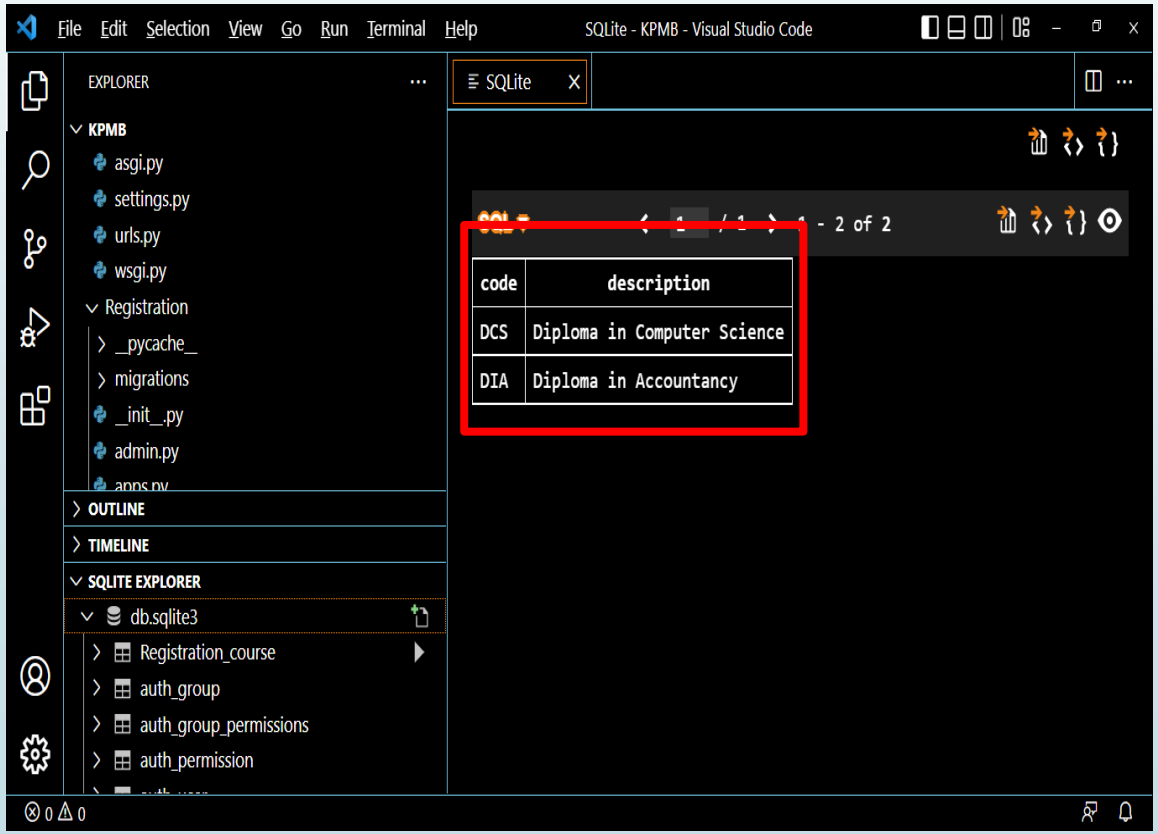
How to update data in table

1. Stay in shell in command prompt
2. Display all data
   - Use Python interpreter (Python shell) to add data in table
     - Course.objects.all( ).values( )

C:\Windows\System32\cmd.exe - python manage.py shell

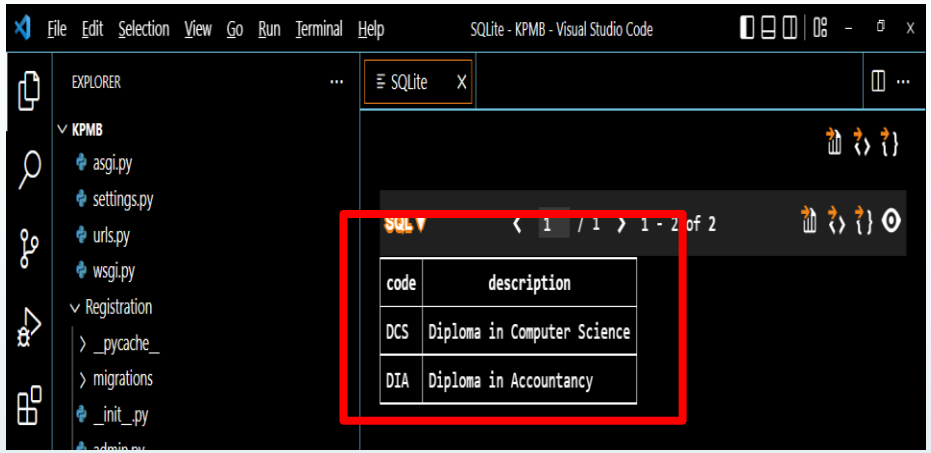
```
>>>
>>> Course.objects.all().values()
<QuerySet [{'code': 'DCS', 'description': 'Diploma in Computer Science'},
{'code': 'DIA', 'description': 'Diploma in Accountancy'}]>
>>>
```

3. Go to Visual Code
   - Go to SQLITE EXPLORER – click Register_course
   - Right hand click show tables

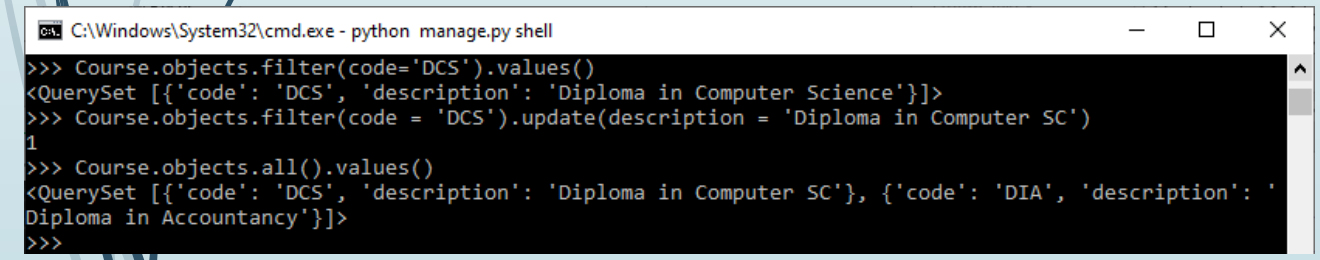


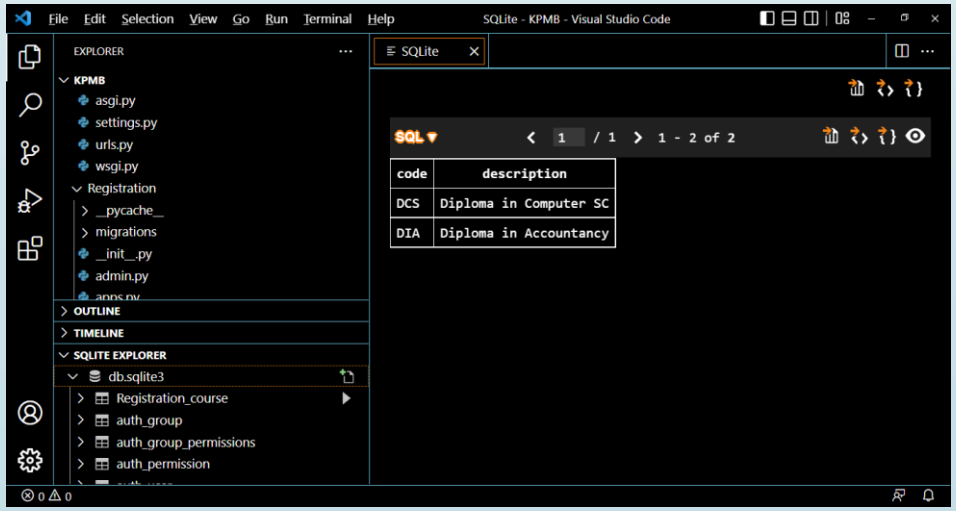
How to update data in table

1. Go to Visual Code
   - Go to SQLITE EXPLORER – click Register_course
   - Right hand click show tables



2. Go to command prompt
3. Stay in shell in command prompt
4. Update specific data
   - Use Python interpreter (Python shell) to update data in table
     - Course.objects.filter(code = 'DCS').update(description = 'Diploma in Computer SC')





PRE- PRACTICAL TEST
DURATION : 1 HOUR 30 MINUTES