Django Application Development with SQL and Databases

Table of Contents

[MODULE 1 Getting Started with SQL and Relational Databases 2](#_Toc186834205)

[Relational Data Concepts 2](#_Toc186834206)

[Module 1 Lesson 1 Summary 2](#_Toc186834207)

[Module 1 Bonus Lesson 3 Summary 4](#_Toc186834208)

[Module 1 Summary: Getting Started with SQL & Relational Databases 5](#_Toc186834209)

[MODULE 2 ORM Bridging the Gap Between the Real World and Relational Model 5](#_Toc186834210)

[ORM Tools 6](#_Toc186834211)

[Create Your First Django Model 6](#_Toc186834212)

[Create a Standalone Django ORM Project Template 6](#_Toc186834213)

[CRUD on Django Model Objects 6](#_Toc186834214)

[Module 2 Summary: ORM: Bridging the Gap Between the Real World and Relational Model 6](#_Toc186834215)

[Module 2 Cheat Sheet: ORM: Bridging the Gap Between the Real World and Relational Model 7](#_Toc186834216)

[MODULE 3 Full Stack Django Development 9](#_Toc186834217)

[Create Your First Django App and Deploy using Docker 9](#_Toc186834218)

[Django Admin 9](#_Toc186834219)

[Views and Templates 9](#_Toc186834220)

[Advanced Features of the Django Template 9](#_Toc186834221)

[Module 3 Summary: Full-stack Django Development 9](#_Toc186834222)

[Module 3 Cheat Sheet: Full-stack Django Development 10](#_Toc186834223)

[MODULE 4 Consolidate and Deploy Your Django App 11](#_Toc186834224)

[Hands-on Lab: Class-based and Generic Views 11](#_Toc186834225)

[Hands-on Lab: Bootstrap Integration 11](#_Toc186834226)

[Module 4 Summary: Consolidate and Deploy Your Django App 11](#_Toc186834227)

[Module 4 Cheat Sheet: Consolidate and Deploy Your Django App 12](#_Toc186834228)

[MODULE 5 Practice Project: Customer 360 13](#_Toc186834229)

[Practice Project 13](#_Toc186834230)

[Final Project 14](#_Toc186834231)

## **MODULE 1 Getting Started with SQL and Relational Databases**

## Relational Data Concepts

<https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-CD0251EN-SkillsNetwork/labs/m1_django_RDBMS-SQL/Lab%20-%20Relational%20Data%20Concepts/instructional-labs.md.html?origin=www.coursera.org?origin=www.coursera.org>

## Module 1 Lesson 1 Summary

* Structured Query Language, or SQL, was designed to manage data in relational databases and is useful for handling structured data.
* Data is a collection of facts in words, numbers, and pictures.
* A database is a repository of data that provides functionality for adding, modifying, and querying data.
* Relational databases store tabular data as collections of related items, with columns containing item properties.
* The basic SQL statements are CREATE TABLE, INSERT, SELECT, UPDATE, and DELETE.
* Non-relational databases provide a flexible and scalable approach to storing and retrieving data.
* Relational databases are ideal for the optimized storage, retrieval, and processing of large volumes of data.
* RDBMS is a mature and well-documented technology, providing flexibility, reduced redundancy, ease of backup and disaster recovery, and ACID compliance.
* An Entity-Relationship model is a tool for designing relational databases. Entities become tables, and attributes are translated into columns.

Simple SELECT Statements

<https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DB0201EN-SkillsNetwork/labs/v8/Simple_SELECT_statements.md.html>

INSERT, UPDATE, and DELETE

<https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DB0201EN-SkillsNetwork/labs/Labs_Coursera_V5/labs/Lab%20-%20INSERT%20-%20UPDATE%20-%20DELETE/instructional-labs.md.html>

Module 1 Lesson 2 Summary

* Structured Query Language, or SQL, is used for querying and managing data.
* SQL is useful for handling structured data or data incorporating relations among entities and variables.
* The SQL SELECT statement retrieves data from a relational database table.
* The SELECT statement is a query, and the output we get from executing this query is a Result Set or a Result Table.
* In its simplest form, the syntax for a SELECT statement is: SELECT \* from TableName
* The SQL INSERT statement inserts data into a relational database table by adding rows.
* The syntax of the INSERT statement is: INSERT INTO TableNameColumnName VALUES (values)
* For the INSERT statement, the values provided in the Values clause must equal the number of column names specified in the Column Name list. This ensures that each column has a value.
* The SQL UPDATE statement is used to read and modify data.
* The syntax of the UPDATE statement is as follows:  UPDATE [TableName] SET [[ColumnName]=[Value]] <WHERE [Condition]>
* The SQL DELETE statement is used to remove data from a table.
* The syntax of the DELETE statement is: DELETE FROM [TableName] <Where [Condition]>
* The WHERE clause specifies the rows in a table that will be acted on by a SQL statement such as SELECT, DELETE, or UPDATE.

Module 1 Cheat Sheet: Getting Started with SQL & Relational Databases

|  |  |  |
| --- | --- | --- |
| Package/Method | Description | Code Example |
| SELECT | Retrieves data from one or more tables based on specified columns. | SELECT column1, column2 FROM table\_name; |
| FROM | Specifies the table from which data is retrieved. | SELECT column1, column2 FROM table\_name; |
| WHERE | Filters data based on specified conditions. | SELECT column1, column2 FROM table\_name WHERE condition; |
| ORDER BY | Sorts the result set based on specified columns in ascending or descending order. | SELECT column1, column2 FROM table\_name ORDER BY column1 ASC; |
| GROUP BY | Groups rows based on a specified column. | SELECT column1, COUNT(\*) FROM table\_name GROUP BY column1; |
| HAVING | Filters grouped data based on specified conditions. | SELECT column1, COUNT(\*) FROM table\_name GROUP BY column1 HAVING COUNT(\*) > 1; |
| INSERT INTO | Inserts data into a table. | INSERT INTO table\_name (column1, column2) VALUES (value1, value2); |
| UPDATE | Modifies data in a table based on specified conditions. | UPDATE table\_name SET column1 = value1 WHERE condition; |
| DELETE FROM | Deletes data from a table based on specified conditions. | DELETE FROM table\_name WHERE condition; |
| JOIN | Combines rows from multiple tables based on related columns. | SELECT column1, column2 FROM table1 JOIN table2 ON table1.column = table2.column; |
| INNER JOIN | Returns only matching rows from both tables. | SELECT column1, column2 FROM table1 INNER JOIN table2 ON table1.column = table2.column; |
| LEFT JOIN | Returns all rows from the left table and matching rows from the right table. | SELECT column1, column2 FROM table1 LEFT JOIN table2 ON table1.column = table2.column; |
| RIGHT JOIN | Returns all rows from the right table and matching rows from the left table. | SELECT column1, column2 FROM table1 RIGHT JOIN table2 ON table1.column = table2.column; |
| FULL JOIN | Returns all rows from both tables, regardless of the match. | SELECT column1, column2 FROM table1 FULL JOIN table2 ON table1.column = table2.column; |
| DISTINCT | Returns unique values from a column. | SELECT DISTINCT column1 FROM table\_name; |
| COUNT | Counts the number of rows or non-null values in a column. | SELECT COUNT(\*) FROM table\_name; or SELECT COUNT(column1) FROM table\_name; |
| SUM | Calculates the sum of values in a column. | SELECT SUM(column1) FROM table\_name; |
| AVG | Calculates the average value of a column. | SELECT AVG(column1) FROM table\_name; |
| MAX | Finds the maximum value in a column. | SELECT MAX(column1) FROM table\_name; |
| MIN | Finds the minimum value in a column. | SELECT MIN(column1) FROM table\_name; |

## Module 1 Bonus Lesson 3 Summary

* CREATE, ALTER, TRUNCATE, and DROP define, change, or drop database objects such as tables.
* The CREATE TABLE statement includes the definition of attributes of columns in the table, including:
* Names of columns
* Datatypes of columns
* Optional values, if required, such as the Primary Key constraint
* The ORDER BY clause is used in a SQL query to sort the result set by a specified column. For example, “ORDER BY title” sorts the result set by the column named “title.”
* You can also specify the sort of column by indicating the column sequence number. For example, “select title, pages from book ORDER BY 2” indicates that the sort order is based on the values in the second column.
* The JOIN operator combines rows from two or more tables based on a relationship between certain columns in these tables.
* The tables being joined are related by a common column, which is usually the primary key of one table and appears as a foreign key in the other table.
* The Primary Key of a relational table uniquely identifies each row in a table.
* A Foreign Key is a set of columns referring to a primary key of another entity.
* If you need to combine data from three or more different tables, add new tables to the joins. First, you combine table A and table B information by matching their shared attributes. Then, you join the information from table B and table C by matching their shared attributes.
* SQL offers several types of JOINs - such as INNER JOIN and OUTER JOIN. You can extract a data set corresponding to the intersection of the two tables involved or choose a bigger data set.

## Module 1 Summary: Getting Started with SQL & Relational Databases

* CREATE, ALTER, TRUNCATE, and DROP define, change, or drop database objects such as tables.
* The CREATE TABLE statement includes the definition of attributes of columns in the table, including:
* Names of columns
* Datatypes of columns
* Optional values, if required, such as the Primary Key constraint
* The ORDER BY clause is used in a SQL query to sort the result set by a specified column. For example, “ORDER BY title” sorts the result set by the column named “title.”
* You can also specify the sort of column by indicating the column sequence number. For example, “select title, pages from book ORDER BY 2” indicates that the sort order is based on the values in the second column.
* The JOIN operator combines rows from two or more tables based on a relationship between certain columns in these tables.
* The tables being joined are related by a common column, which is usually the primary key of one table and appears as a foreign key in the other table.
* The Primary Key of a relational table uniquely identifies each row in a table.
* A Foreign Key is a set of columns referring to a primary key of another entity.
* If you need to combine data from three or more different tables, add new tables to the joins. First, you combine table A and table B information by matching their shared attributes. Then, you join the information from table B and table C by matching their shared attributes.
* SQL offers several types of JOINs - such as INNER JOIN and OUTER JOIN. You can extract a data set corresponding to the intersection of the two tables involved or choose a bigger data set.

## **MODULE 2 ORM Bridging the Gap Between the Real World and Relational Model**

## ORM Tools

<https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMSkillsNetwork-DB0211EN-edX/labs/Reading%3A_ORM_Tools.md.html?origin=www.coursera.org?origin=www.coursera.org>

## Create Your First Django Model

<https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-CD0251EN-SkillsNetwork/labs/m3_django_orm/lab1_first_model.md.html>

## Create a Standalone Django ORM Project Template

<https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-CD0251EN-SkillsNetwork/labs/m3_django_orm/option_lab_create_orm_template.md.html>

## CRUD on Django Model Objects

<https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-CD0251EN-SkillsNetwork/labs/m3_django_orm/lab2_crud.md.html>

Query Spanning Relationships

<https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-CD0251EN-SkillsNetwork/labs/m3_django_orm/lab3_related.md.html>

## Module 2 Summary: ORM: Bridging the Gap Between the Real World and Relational Model

* The Object-Oriented Programming (OOP) and SQL paradigm model data differently.
* Object Relational Mapping, or ORM bridges the gap between OOP and SQL.
* ORM libraries or tools can map and transfer data stored in a relational database as rows into objects or objects into rows.
* ORM allows developers to use OOP to query and manipulate data because it transfers objects into rows and rows into objects.
* Django ORM is a Python ORM component that belongs to the Django web application framework.
* Django ORM can help speed up database development because you define maps to a database table for each Django model.
* Each Django field maps to a column type.
* Django automatically creates tables once models and fields are defined.
* Django APIs can perform Create, Read, Update, and Delete (CRUD) operations on database objects.
* In a Django model, you create an object and call the model’s save method to insert it into the database as a record.
* You must construct a QuerySet using a Manager on your model class to read objects using Django Model API.
* There are several ways to update database records in Django by updating objects.
* To delete records in a database, you call Django ORM’s Delete method on a model object or query set.

## Module 2 Cheat Sheet: ORM: Bridging the Gap Between the Real World and Relational Model

|  |  |  |
| --- | --- | --- |
| Package/Method | Description | Code Example |
| django.db.models.Model | Define a model. | from django.db import models class MyModel(models.Model): field1 = models.CharField(max\_length=100) field2 = models.IntegerField() |
| makemigrations/migrate | Create database tables based on models. | python manage.py makemigrations python manage.py migrate |
| all() | Retrieves all instances of the ‘MyModel’ model from the database. | MyModel.objects.all() |
| filter() | Filter objects using conditions. | MyModel.objects.filter(field1="value") MyModel.objects.filter(field2\_\_gt=5) |
| get() | Retrieves a single instance of the ‘MyModel’ model from the database where the value of ‘field1’ is "value". | MyModel.objects.get(field1="value") |
| save() (Create) | Creates a new instance of the ‘MyModel’ model and saves it to the database. | obj = MyModel(field1="value", field2=5) obj.save() |
| save() (Update) | Updates the value of ‘field1’ for the ‘obj’ instance and saves the changes to the database. | obj.field1 = "new value" obj.save() |
| delete() | Deletes an object. | obj.delete() |
| obj.related\_model | Retrieves the related model associated with the ‘obj’ instance (Foreign Key or OneToOneField). | obj.related\_model |
| obj.model\_set.all() | Fetches all related objects associated with the ‘obj’ instance (reverse relation). | obj.model\_set.all() |
| field | Performs a filtering operation on the ‘MyModel’ model instances based on a related model's field value. | MyModel.objects.filter(related\_model\_\_field="value") |
| exact | Retrieves instances of the ‘MyModel’ model where the field attribute is exactly equal to "value". | MyModel.objects.filter(field\_\_exact="value") |
| iexact | Case-insensitive match where the field is equal to "value". | MyModel.objects.filter(field\_\_iexact="value") |
| contains | Checks if the value is a substring within the field. | MyModel.objects.filter(field\_\_contains="value") |
| startswith | Determines whether a string begins with the specified characters. | MyModel.objects.filter(field\_\_startswith="value") |
| endswith | Determines whether a string ends with the specified suffix. | MyModel.objects.filter(field\_\_endswith="value") |
| in | Checks if the value of the field is present in the given list of values. | MyModel.objects.filter(field\_\_in=["value1", "value2"]) |
| gt | Checks if the value of ‘field’ is greater than the specified value (e.g., 5). | MyModel.objects.filter(field\_\_gt=5) |
| lt | Checks if the value of ‘field’ is less than the specified value (e.g., 10). | MyModel.objects.filter(field\_\_lt=10) |

## **MODULE 3 Full Stack Django Development**

## Create Your First Django App and Deploy using Docker

<https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-CD0251EN-SkillsNetwork/labs/m4_django_app/lab1_first_django_app.md.html>

## Django Admin

<https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-CD0251EN-SkillsNetwork/labs/m4_django_app/lab2_admin.md.html>

## Views and Templates

<https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-CD0251EN-SkillsNetwork/labs/m4_django_app/lab3_views_templates.md.html>

## Advanced Features of the Django Template

<https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMSkillsNetwork-PY0103EN-SkillsNetwork/labs/M03/Reading/Advanced_features_of_Django_templates.md.html?origin=www.coursera.org?origin=www.coursera.org>

## Module 3 Summary: Full-stack Django Development

* The Model-View-Controller design pattern divides application logic into three components:
* Model accesses and manipulates data
* View presents data in various forms
* Controller coordinates between Model and View
* The Django Model-View-Template pattern is like MVC, except there is no Controller, and the Django server performs the controller function.
* In Django, a View is a Python function that takes a Web request and applies the necessary logic to generate a Web response.
* Django uses a template containing static HTML elements and special Python code to generate dynamic Web pages.
* When you create a Django project, Django creates some core files.
* manage.py is a command-line interface used to interact with the Django project
* settings.py contains the settings and configurations for your Django project
* urls.py contains the URL and routing definitions of your Django app
* You start building a Django admin site by creating an admin user.
* You can then log in as a superuser and register your models to the admin site so you can manage them.
* You can customize the admin form and add search and filters.
* A Django View takes a Web request such as HTTP GET, POST, DELETE, or UPDATE and returns a Web response. The web response can be a string, JSON/XML file, HTML page, or an error status indicating client or server-side errors.
* You create templates in Django to specify how your data will be presented. A Django template combines static HTML elements with Django template tags and variables to describe how the dynamic parts will be inserted. These work together to generate an HTML page rendered in a user’s web browser.

## Module 3 Cheat Sheet: Full-stack Django Development

|  |  |  |
| --- | --- | --- |
| **Package/Method** | **Description** | **Code Example** |
| count() | Counts the number of objects. | MyModel.objects.count() |
| Sum() | Provides the sum of a field. | MyModel.objects.aggregate(Sum('field')) |
| Avg() | Calculates the average of a field. | MyModel.objects.aggregate(Avg('field')) |
| Max() | Provides the maximum value of a field. | MyModel.objects.aggregate(Max('field')) |
| Min() | Provides the minimum value of a field. | MyModel.objects.aggregate(Min('field')) |
| order\_by() | Orders objects based on a field. | MyModel.objects.order\_by('field') |
| order\_by('-') | Orders objects based on fields in descending order. | MyModel.objects.order\_by('-field') |
| select\_related | Performs inner join. | MyModel.objects.select\_related('related\_model') |
| prefetch\_related | Performs left outer join. | MyModel.objects.prefetch\_related('related\_model') |
| many\_to\_many | Performs many-to-many join. | obj.many\_to\_many\_field.all() |
| filter(ForeignKey) | Performs conditional joins. | MyModel.objects.filter(related\_model\_\_isnull=True) |

## **MODULE 4 Consolidate and Deploy Your Django App**

## Hands-on Lab: Class-based and Generic Views

<https://author-ide.skills.network/render?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJtZF9pbnN0cnVjdGlvbnNfdXJsIjoiaHR0cHM6Ly9jZi1jb3Vyc2VzLWRhdGEuczMudXMuY2xvdWQtb2JqZWN0LXN0b3JhZ2UuYXBwZG9tYWluLmNsb3VkL0lCTS1DRDAyNTFFTi1Ta2lsbHNOZXR3b3JrL2xhYnMvbTVfZGphbmdvX2FkdmFuY2VkL2xhYjFfY2xhc3NfYmFzZWRfdmlld3MubWQiLCJ0b29sX3R5cGUiOiJ0aGVpYSIsImF0bGFzX2ZpbGVfaWQiOjI1MjM2LCJhZG1pbiI6ZmFsc2UsImlhdCI6MTczMDE1Mjc1Nn0.NX3OgyaOOuiiAyMFe7vV0BzhT7goS_CnbIBXQZzdNAw>

## Hands-on Lab: Bootstrap Integration

<https://author-ide.skills.network/render?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJtZF9pbnN0cnVjdGlvbnNfdXJsIjoiaHR0cHM6Ly9jZi1jb3Vyc2VzLWRhdGEuczMudXMuY2xvdWQtb2JqZWN0LXN0b3JhZ2UuYXBwZG9tYWluLmNsb3VkL0lCTS1DRDAyNTFFTi1Ta2lsbHNOZXR3b3JrL2xhYnMvbTVfZGphbmdvX2FkdmFuY2VkL2xhYjNfYm9vdHN0cmFwLm1kIiwidG9vbF90eXBlIjoidGhlaWEiLCJhdGxhc19maWxlX2lkIjoyNTI0NCwiYWRtaW4iOmZhbHNlLCJpYXQiOjE3MzAxNTI4NjZ9.WLI15ROQJ-b94nfobajJeB-RWl4KtVHbrXrvVBtC7PY>

## Module 4 Summary: Consolidate and Deploy Your Django App

* Both function-based and class-based views are Python functions.
* When you build a class-based view, you define a class subclassing the Django View base class. Then you access some standard methods such as Get or Post. Next, you implement your logic to handle HTTP requests.
* To speed up development and solve common tasks, Django provides some built-in view classes called generic-based views for developers to reuse.
* Authentication is validating users’ identities using credentials such as username and password.
* After users are authenticated, authorization will check the users’ access permissions for resources such as databases.
* In Django, a user model is created to handle authentication and to work with other models, such as groups and permissions, to handle authorization.
* Developers can extend the User model to define application-specific users, such as instructors or learners inherited from the User model.
* Bootstrap, a free web front-end framework, facilitates web app development.
* Bootstrap provides many HTML and CSS templates to simplify Django template development.
* If you want to use Bootstrap CSS style classes without downloading Bootstrap, add a link to the latest Bootstrap version into the head element of your HTML template.
* To add static files to your apps, you first create folders for different static files, such as HTML templates, images, or CSS files.
* Under each folder to hold static files, you create a subfolder using the same app name. This creates name spacing to uniquely refer to static files that use the same name across multiple apps in a Django project.
* Django provides a set of STACFILES\_FINDERS for locating the static files in an app. It also provides a staticfiles app to collect all static files in a single directory when an app is deployed.
* To deploy reliable, scalable, and maintainable Django apps, you need to deploy them on web servers.
* Since most web servers are not written in Python, Django apps need extra interfaces to talk to web servers.
* The Web Server Gateway Interface, or WSGI, is the main Python standard for communicating between Web servers and applications.
* The Django app supports the Asynchronous Server Gateway Interface, another web server interface.
* Infrastructure as a Service and Platform as a Service offering allows you to focus on your app development and deploy apps without worrying about the underlying Infrastructure and platform.

## Module 4 Cheat Sheet: Consolidate and Deploy Your Django App

|  |  |  |
| --- | --- | --- |
| **Feature** | **Description** | **Code Example** |
| **ListView** | Displays a list of objects. | python<br>class MyListView(ListView):<br> model = MyModel<br> template\_name = 'my\_template.html'<br> context\_object\_name = 'object\_list' # default<br> |
| **DetailView** | Displays details of a single object. | python<br>class MyDetailView(DetailView):<br> model = MyModel<br> template\_name = 'my\_template.html'<br> pk\_url\_kwarg = 'my\_model\_id' # default: pk<br> |
| **CreateView** | Displays a form to create a new object. | python<br>class MyCreateView(CreateView):<br> model = MyModel<br> template\_name = 'my\_template.html'<br> fields = '\_\_all\_\_'<br> |
| **UpdateView** | Displays a form to update an existing object. | python<br>class MyUpdateView(UpdateView):<br> model = MyModel<br> template\_name = 'my\_template.html'<br> fields = '\_\_all\_\_'<br> |
| **DeleteView** | Displays a confirmation page to delete an object. | python<br>class MyDeleteView(DeleteView):<br> model = MyModel<br> template\_name = 'my\_template.html'<br> success\_url = '/success-url/'<br> |
| **Basic View Function** | Function-based view that returns “Hello, World!” | python<br>from django.http import HttpResponse<br>def my\_view(request):<br> return HttpResponse("Hello, World!")<br> |
| **Render a Template** | Function-based view to render a template with context. | python<br>from django.shortcuts import render<br>def my\_template\_view(request):<br> return render(request, 'my\_template.html', {'variable': value})<br> |
| **Redirect to a URL** | Function-based view to redirect to a specific URL. | python<br>from django.shortcuts import redirect<br>def my\_redirect\_view(request):<br> return redirect('url\_name\_or\_path')<br> |
| **Handle a Form Submission** | Function-based view to handle form submission. | python<br>from django.shortcuts import render<br>def my\_form\_view(request):<br> if request.method == 'POST':<br> # Process form data<br> return render(request, 'my\_form\_template.html')<br> |
| **Handle URL Parameters** | Function-based view that accesses URL parameters. | python<br>def my\_param\_view(request, param):<br> # Access 'param' value<br> |
| **Restrict Access** | Protects views using @login\_required. | python<br>from django.contrib.auth.decorators import login\_required<br>@login\_required<br>def my\_protected\_view(request):<br> # View logic<br> |
| **Bootstrap CSS** | Link to include Bootstrap CSS in templates. | html<br><link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet"><br> |
| **Bootstrap JavaScript** | Script tag to include Bootstrap JavaScript library. | html<br><script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.min.js"></script><br> |
| **Bootstrap Classes and Components** | Use Bootstrap to style elements. | html<br><a href="#" class="btn btn-primary">Click Me</a><br> |
| **Static Files Configuration** | Django settings for static files. | python<br>STATIC\_URL = '/static/'<br>STATICFILES\_DIRS = [os.path.join(BASE\_DIR, 'static')]<br> |
| **Installed Apps Configuration** | List of installed apps in Django settings. | python<br>INSTALLED\_APPS = [<br> 'django.contrib.staticfiles',<br>] |
| **App Dirs Configuration** | Enables template search within app directories. | python<br>TEMPLATES = [<br> {'APP\_DIRS': True,<br>}<br>] |
| **Usage of Static Content** | Link CSS/JS or images in templates. | html<br><link href="{% static 'your\_app/css/style.css' %}" rel="stylesheet"><br><img src="{% static 'your\_app/img/logo.png' %}" alt="Logo"><br> |
| **Collecting Static Files** | Command to collect static files during deployment. | bash<br>python manage.py collectstatic<br>STATIC\_ROOT = os.path.join(BASE\_DIR, 'staticfiles')<br> |

## **MODULE 5 Practice Project: Customer 360**

## Practice Project

<https://author-ide.skills.network/render?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJtZF9pbnN0cnVjdGlvbnNfdXJsIjoiaHR0cHM6Ly9jZi1jb3Vyc2VzLWRhdGEuczMudXMuY2xvdWQtb2JqZWN0LXN0b3JhZ2UuYXBwZG9tYWluLmNsb3VkL0lCTVNraWxsc05ldHdvcmstREIwMjExRU4tZWRYL2xhYnMvUHJhY3RpY2VQcm9qZWN0LUN1c3RvbWVyMzYwLm1kIiwidG9vbF90eXBlIjoidGhlaWFkb2NrZXIiLCJhdGxhc19maWxlX2lkIjoyODQ4NywiYWRtaW4iOmZhbHNlLCJpYXQiOjE3MzAxNjA2NjJ9.C5buXkA5JC3-ZgOfXnxNFc0AtkpMGdwvttOHladLJPY>

## Final Project

<https://author-ide.skills.network/render?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJtZF9pbnN0cnVjdGlvbnNfdXJsIjoiaHR0cHM6Ly9jZi1jb3Vyc2VzLWRhdGEuczMudXMuY2xvdWQtb2JqZWN0LXN0b3JhZ2UuYXBwZG9tYWluLmNsb3VkL0lCTVNraWxsc05ldHdvcmstREIwMjExRU4tZWRYL2xhYnMvRmluYWxQcm9qZWN0LzAyLUluc3RydWN0aW9ucy5tZCIsInRvb2xfdHlwZSI6InRoZWlhZG9ja2VyIiwiYXRsYXNfZmlsZV9pZCI6Mjg0ODMsImFkbWluIjpmYWxzZSwiaWF0IjoxNzMwMTYwNjg4fQ.f4XC5xwakPCB4gOfmZXvN6S6Ou3xAppmaD9Gbe2CCws>