# Django app that generates CSV and PDF files by retrieving data from models

**1. Create a Django Project and App**

First, create a Django project and an app within the project:

```
django-admin startproject report_generator_project
cd report_generator_project
python manage.py startapp report_generator_app
```

**2. Define Model for Data**

In `report_generator_app/models.py`, define a model to represent the data you want to generate reports for:

```
from django.db import models

class ReportData(models.Model):
    name = models.CharField(max_length=100)
    age = models.IntegerField()
    email = models.EmailField()

    def __str__(self):
        return self.name
```

**3. Create Views for Generating Reports**

In `report_generator_app/views.py`, create views for generating CSV and PDF reports:

```
from django.http import HttpResponse
from django.template.loader import get_template
from xhtml2pdf import pisa
import csv
from .models import ReportData

def generate_csv(request):
    response = HttpResponse(content_type='text/csv')
    response['Content-Disposition'] = 'attachment; filename="report.csv"'

    writer = csv.writer(response)
    writer.writerow(['Name', 'Age', 'Email'])

    queryset = ReportData.objects.all()
    for item in queryset:
```

```
        writer.writerow([item.name, item.age, item.email])

    return response

def generate_pdf(request):
    template_path = 'report_generator_app/pdf_template.html'
    queryset = ReportData.objects.all()
    context = {'data': queryset}

    response = HttpResponse(content_type='application/pdf')
    response['Content-Disposition'] = 'attachment; filename="report.pdf"'

    template = get_template(template_path)
    html = template.render(context)

    pisa_status = pisa.CreatePDF(html, dest=response)
    if pisa_status.err:
        return HttpResponse('PDF generation failed')

    return response
```

**4. Create HTML Template for PDF Generation**

Create an HTML template to render data for PDF generation. Save it as `pdf_template.html` in the `report_generator_app/templates/report_generator_app` directory:

```
<!DOCTYPE html>
<html>
<head>
    <title>Report</title>
</head>
<body>
    <h2>Report</h2>
    <table border="1">
        <tr>
            <th>Name</th>
            <th>Age</th>
            <th>Email</th>
        </tr>
        {% for item in data %}
        <tr>
            <td>{{ item.name }}</td>
            <td>{{ item.age }}</td>
            <td>{{ item.email }}</td>
        </tr>
```

```
        {% endfor %}
    </table>
</body>
</html>
```

**5. Configure URLs**
Configure URLs in `report_generator_app/urls.py`:

```
from django.urls import path
from . import views

urlpatterns = [
    path('generate-csv/', views.generate_csv, name='generate_csv'),
    path('generate-pdf/', views.generate_pdf, name='generate_pdf'),
]
```

**6. Include App URLs in Project URLs**
Include app URLs in the project's `urls.py`:

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('report_generator_app.urls')),
]
```

**7. Install Required Packages**
You'll need to install the `xhtml2pdf` package for PDF generation:

```
pip install xhtml2pdf
```

**Explanation:**

- **Model**:
  `ReportData` model represents the data for which reports will be generated.
- **Views**:
  `generate_csv` retrieves data from the database and generates a CSV file.
  `generate_pdf` retrieves data and renders it in an HTML template, then converts it to a PDF file using `xhtml2pdf`.
- **Templates**:
  `pdf_template.html` contains the HTML structure for the PDF report, with data passed from the view.
- **URLs**:
  URLs are configured to map to the `generate_csv` and `generate_pdf` views.

- **Settings**:
    Make sure to configure `STATIC_ROOT` and `STATIC_URL` settings in `settings.py` to serve static files, including the PDF template.

That's it! With this setup, you can access `/generate-csv/` and `/generate-pdf/` URLs to generate CSV and PDF reports, respectively, based on the data stored in the `ReportData` model.