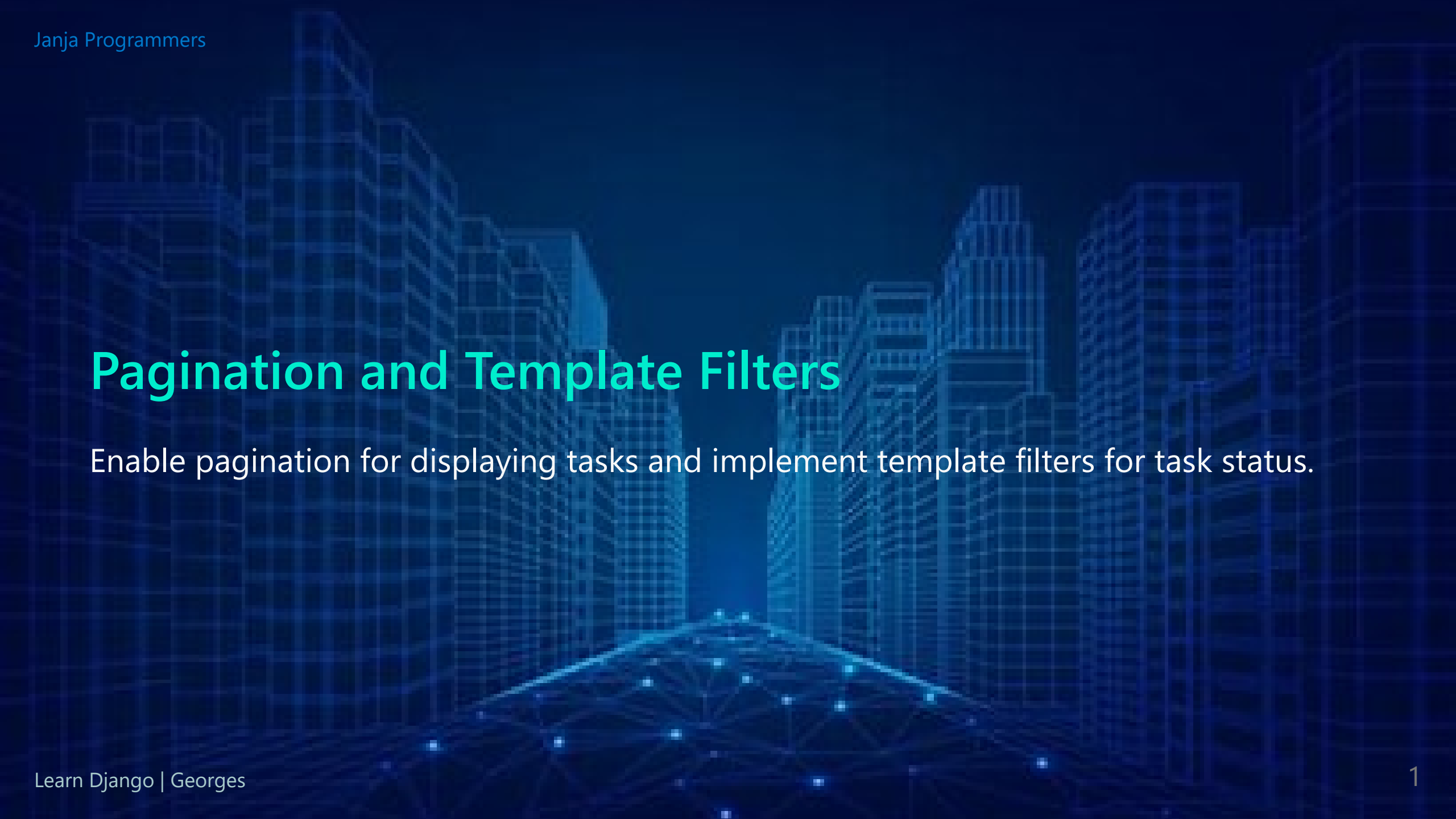


# Pagination and Template Filters

Enable pagination for displaying tasks and implement template filters for task status.

# Pagination Integration

- Modify the views to paginate the list of tasks retrieved from the database, limiting the number of tasks per page.
- Update the HTML templates to display paginated tasks and provide navigation links for pagination.

```python
# Example: Pagination in Django views
from django.core.paginator import Paginator, EmptyPage, PageNotAnInteger

def task_list(request):
    tasks = Task.objects.all()
    paginator = Paginator(tasks, 10)  # Show 10 tasks per page

    page = request.GET.get('page')
    try:
        paginated_tasks = paginator.page(page)
    except PageNotAnInteger:
        paginated_tasks = paginator.page(1)
    except EmptyPage:
        paginated_tasks = paginator.page(paginator.num_pages)

    return render(request, 'task_list.html', {'tasks': paginated_tasks})
```

# HTML Integration

```
<!-- Example: Displaying paginated tasks in HTML template -->
{% for task in tasks %}
<!-- Task item -->
{% endfor %}

<!-- Pagination links -->
<div class="pagination">
  <span class="step-links">
    {% if tasks.has_previous %}
    <a href="?page=1">&laquo; first</a>
    <a href="?page={{ tasks.previous_page_number }}">previous</a>
    {% endif %}

    <span class="current">
      Page {{ tasks.number }} of {{ tasks.paginator.num_pages }}.
    </span>

    {% if tasks.has_next %}
    <a href="?page={{ tasks.next_page_number }}">next</a>
    <a href="?page={{ tasks.paginator.num_pages }}">last &raquo;</a>
    {% endif %}
  </span>
</div>
```

# Template Filters Implementation

- Implement template filters for task status (e.g., "completed", "pending") to format the display of task statuses in the task list.
- Use template filters to apply formatting or transformations to task data displayed in the templates.

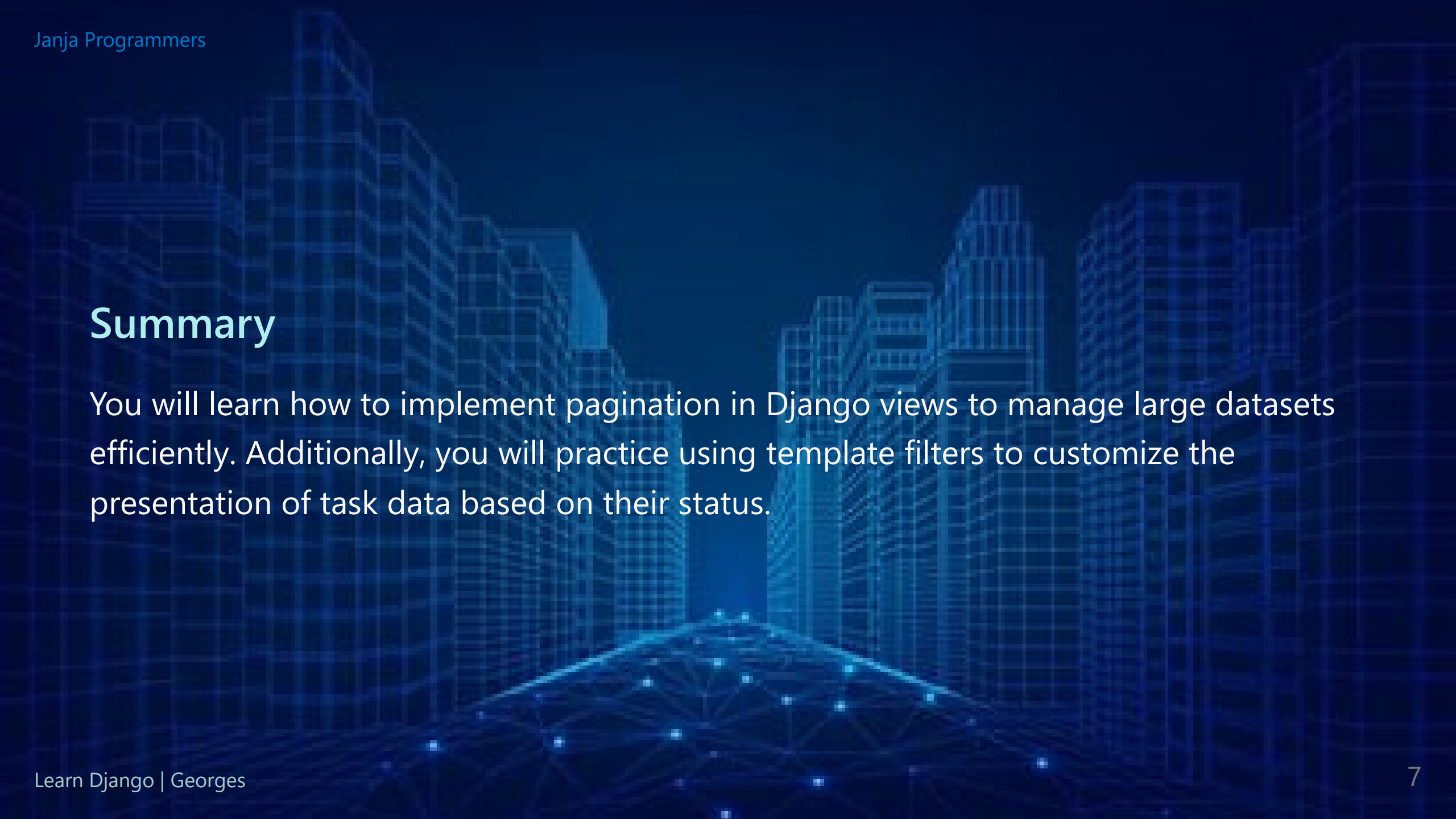
```python
# Example: Custom template filter for task status
from django import template

register = template.Library()

@register.filter(name='task_status')
def task_status(value):
    if value:
        return "Completed"
    else:
        return "Pending"
```

## HTML Integration

```html
<!-- Example: Applying template filter in HTML template -->
{% for task in tasks %}
<div>{{ task.name }} - {{ task.completed|task_status }}</div>
{% endfor %}
```



## Summary

You will learn how to implement pagination in Django views to manage large datasets efficiently. Additionally, you will practice using template filters to customize the presentation of task data based on their status.