Lab 2: Advanced CSS Grid – Building a Web Page Layout

Learning Objectives

By the end of this lab, You will:

- Build a complete multi-section webpage layout using CSS Grid.
- Use grid-template-areas for semantic layouts.
- Implement responsive breakpoints with auto-fit and minmax().
- Understand how grid adapts for desktop and mobile designs.

Step 1 – Starter HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Advanced CSS Grid Layout</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>

    <div class="grid-container">
        <header>My Website Header</header>
        <nav>Navigation<br>Home | About | Contact</nav>
        <main>
            <h2>Welcome to My Page</h2>
            <p>This is the main content area where you can place articles or
images.</p>
            <div class="cards">
                <div class="card">Card 1</div>
                <div class="card">Card 2</div>
                <div class="card">Card 3</div>
                <div class="card">Card 4</div>
            </div>
        </main>
        <aside>Sidebar Widgets</aside>
        <footer>© 2025 My Website</footer>
    </div>

</body>
</html>
```

Step 2 – Desktop Grid Layout (style.css)

```
  body {
        margin: 0;
        font-family: Arial, sans-serif;
        background: #f4f4f4;
    }

    .grid-container {
        display: grid;
        grid-template-areas: "header header header" "nav main aside" "footer footer
footer";
```

```css
    grid-template-columns: 180px 1fr 250px;
    grid-template-rows: auto 1fr auto;
    min-height: 100vh;
    gap: 10px;
    padding: 10px;
}

/* Define grid areas */
header {
    grid-area: header;
    background: #34495e;
    color: white;
    text-align: center;
    padding: 20px;
    border-radius: 6px;
}

nav {
    grid-area: nav;
    background: #2ecc71;
    padding: 20px;
    border-radius: 6px;
}

main {
    grid-area: main;
    background: white;
    padding: 20px;
    border-radius: 6px;
}

aside {
    grid-area: aside;
    background: #f1c40f;
    padding: 20px;
    border-radius: 6px;
}

footer {
    grid-area: footer;
    background: #34495e;
    color: white;
    text-align: center;
    padding: 15px;
    border-radius: 6px;
}

/* Nested grid for cards inside main */
.cards {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(180px, 1fr));
    gap: 10px;
    margin-top: 20px;
}

.card {
    background: #3498db;
    color: white;
    padding: 20px;
    text-align: center;
    border-radius: 6px;
    transition: transform 0.2s ease;
```

```
}

    .card:hover {
        transform: scale(1.05);
    }
```

Explanation

- The main layout uses named areas for a full-page structure.
- Inside <main>, there's a **nested grid** for .cards, showing **grid within grid**.
- repeat(auto-fit, minmax(...)) makes the cards responsive automatically.

Step 3 – Responsive Layout for Tablets and Phones

```
@media (max-width: 900px) {
    .grid-container {
        grid-template-areas: "header" "nav" "main" "aside" "footer";
        grid-template-columns: 1fr;
    }

    nav, aside {
        text-align: center;
    }
}
```

Explanation

- The layout **stacks vertically** when the screen is under 900px.
- Navigation and sidebar move below the header for a clean mobile view.

Your Tasks

1. Change the grid so the **aside** area appears *below* the main content even on desktop.
2. Add another section, e.g., .gallery, using grid-template-areas.
3. Change the .cards layout to show **2 cards per row** on tablets.
4. Add a media query that changes the footer background colour on small screens.

Optional Extension

Add **images** to the cards and style them with:
Then use Unsplash images for a visually rich layout.

```
.card img {
    width: 100%;
    border-radius: 6px;
}
```

END