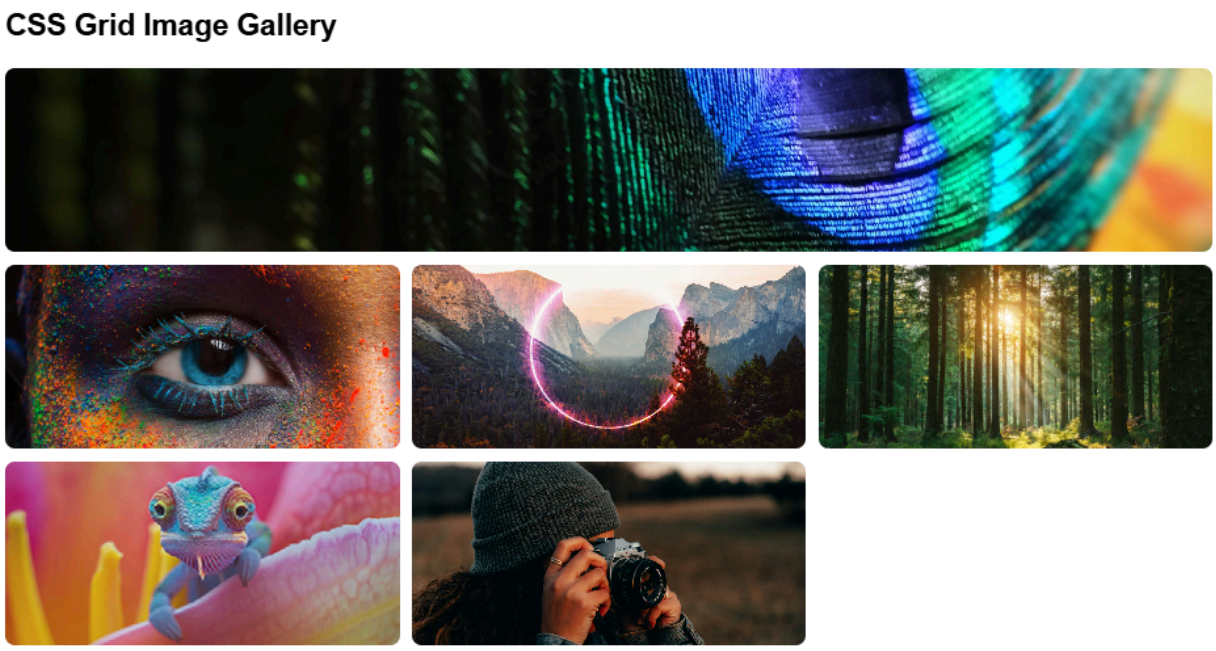
Name = Om Kumar
Email ID = omk5938@gmail.com
Assignment Name = Flexbox in CSS
GitHub = Link

1. Create an image gallery using a CSS grid.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>CSS Grid Image Gallery</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      padding: 20px;
    }

    .gallery {
      display: grid;
      grid-template-columns: repeat(3, 1fr);
      grid-gap: 10px;
      grid-auto-rows: 150px;
    }

    .gallery img {
      width: 100%;
      height: 100%;
      object-fit: cover;
      border-radius: 8px;
    }

    /* Optional: Make first image span all columns */
    .gallery img:nth-child(1) {
      grid-column: span 3;
    }
  </style>
</head>
<body>
  <h2>CSS Grid Image Gallery</h2>
  <div class="gallery">
    <img
src="https://images.pexels.com/photos/674010/pexels-photo-674010.jpeg?cs=srgb&dl=pexels-anjana-c-169994-674010.jpg&fm=jpg" alt="Image 1">
```

```
    <img
src="https://media.istockphoto.com/id/814423752/photo/eye-of-model-with-colorful-art-make-up-close-up.jpg?s=612x612&w=0&k=20&c=l15OdMWjgCKycMMShP8UK94ELVlEGvt7GmB_esHWPYE=" alt="Image 2">
    <img
src="https://image-processor-storage.s3.us-west-2.amazonaws.com/images/3cf61c1011912a2173ea4dfa260f1108/halo-of-neon-ring-illuminated-in-the-stunning-landscape-of-yosemite.jpg" alt="Image 3">
    <img
src="https://media.istockphoto.com/id/1419410282/photo/silent-forest-in-spring-with-beautiful-bright-sun-rays.jpg?s=612x612&w=0&k=20&c=UHeb1pGOw6ozr6utsenXHhV19vW6oiPIxDqhKCS2Llk=" alt="Image 4">
    <img
src="https://letsenhance.io/static/73136da51c245e80edc6ccfe44888a99/1015f/MainBefore.jpg" alt="Image 5">
    <img
src="https://images.unsplash.com/photo-1575936123452-b67c3203c357?fm=jpg&q=60&w=3000&ixlib=rb-4.1.0&ixid=M3wxMjA3fDB8MHxzZWFyY2h8Mnx8aW1hZ2V8ZW58MHx8MHx8fDA%3D" alt="Image 6">
  </div>
</body>
</html>
```

Output:-

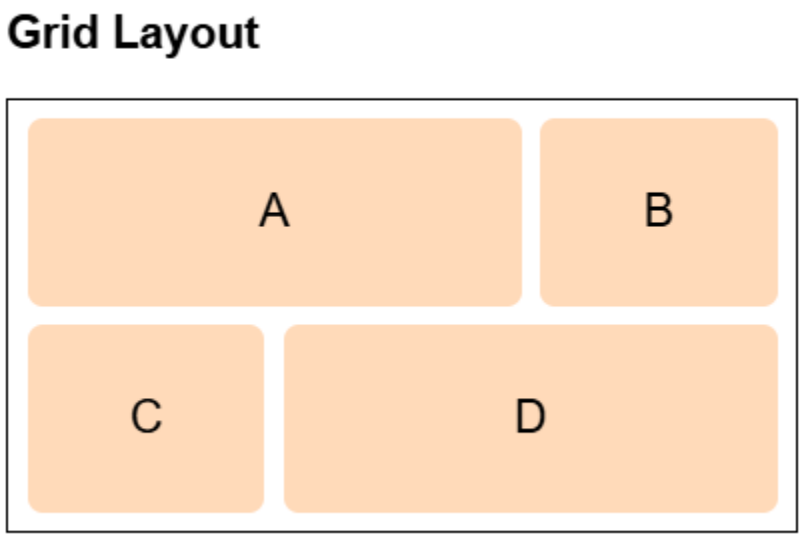


2. Write code to arrange containers with texts A, B, C, and D as shown in the below image.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Grid Layout A B C D</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      padding: 20px;
    }

    .container {
      display: grid;
      grid-template-areas:
        "a a b"
        "c d d";
      grid-template-columns: 1fr 1fr 1fr;
      grid-gap: 10px;
      width: 400px;
      background: #fff;
      padding: 10px;
      border: 2px solid black;
    }

    .box {
      background-color: peachpuff;
      font-size: 24px;
      display: flex;
      align-items: center;
      justify-content: center;
      height: 100px;
      border-radius: 8px;
    }

    .a { grid-area: a; }
    .b { grid-area: b; }
    .c { grid-area: c; }
    .d { grid-area: d; }
  </style>
</head>
<body>
  <h2>Grid Layout</h2>
  <div class="container">
```

```
    <div class="box a">A</div>
    <div class="box b">B</div>
    <div class="box c">C</div>
    <div class="box d">D</div>
  </div>
</body>
</html>
```

Output:-



3.Explain the use of grid-auto-row and grid-auto-column using code examples.

🔶 `grid-auto-rows` and `grid-auto-columns` are used to define the default size of rows or columns that are automatically created when your content overflows the explicitly defined grid.

✅ 1. `grid-auto-rows`

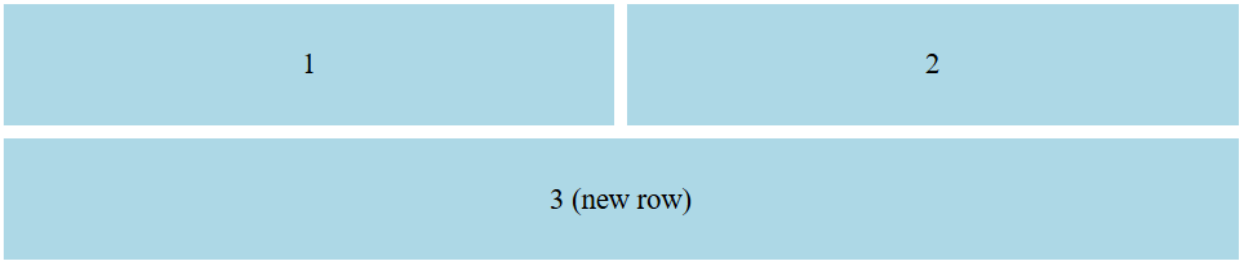
📌 Use Case:

When you **don't define enough rows**, but the content needs more rows, CSS Grid automatically creates new ones. `grid-auto-rows` defines how tall they should be.

🔧 Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .container {
      display: grid;
      grid-template-columns: 1fr 1fr;
      grid-auto-rows: 100px; /* height of any auto-created row */
      gap: 10px;
    }

    .box {
      background: lightblue;
      display: flex;
      align-items: center;
      justify-content: center;
      font-size: 24px;
    }

    .box:nth-child(3) {
      grid-column: span 2; /* pushes to a new row automatically */
    }
  </style>
</head>
<body>

<div class="container">
  <div class="box">1</div>
  <div class="box">2</div>
  <div class="box">3 (new row)</div>
</div>

</body>
</html>
```

Output:-



🧠 What happens?

- We define **only 1 row implicitly** (by filling 2 columns).
- When the third item doesn't fit, **a new row is created** automatically, and `grid-auto-rows: 100px` controls its height.

✅ 2. `grid-auto-columns`

📌 Use Case:

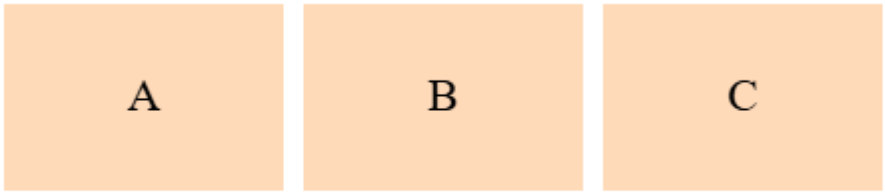
When you define rows but content requires **more columns than defined**, `grid-auto-columns` tells the browser how wide to make them.

🔧 Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .container {
      display: grid;
      grid-template-rows: 100px;
      grid-auto-columns: 150px; /* width of auto-created columns */
      grid-auto-flow: column;   /* makes grid flow horizontally */
      gap: 10px;
    }

    .box {
      background: peachpuff;
      display: flex;
      align-items: center;
      justify-content: center;
      font-size: 24px;
    }
  </style>
</head>
<body>

<div class="container">
  <div class="box">A</div>
```

```
  <div class="box">B</div>
  <div class="box">C</div>
</div>

</body>
</html>
```

Output:-



🧠 What happens?

- Only **1 row** is defined.
- Since `grid-auto-flow: column` is used, items go **horizontally**, creating **new columns**.
- `grid-auto-columns: 150px` sets the width of these auto-generated columns.

✅ Summary Table

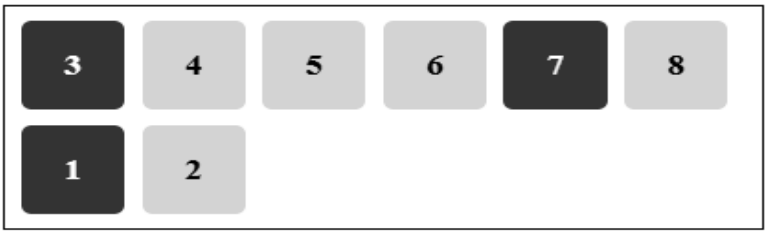
Property	Purpose	When to Use
`grid-auto-rows`	Sets height for automatically added rows	Not enough rows explicitly defined
`grid-auto-columns`	Sets width for automatically added columns	Not enough columns explicitly defined

4.Write CSS to show numbers as shown in the figure, without altering the below html code.

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .container {
      display: flex;
      flex-wrap: wrap;
      width: 420px;
      gap: 10px;
      padding: 10px;
      border: 2px solid #000;
    }

    .box {
      width: 60px;
      height: 60px;
      background-color: lightgray;
      display: flex;
      align-items: center;
      justify-content: center;
      font-weight: bold;
      font-size: 20px;
      border-radius: 6px;
    }

    /* Arrange the boxes using order */
    .box1 { order: 7; background-color: #333; color: white; }
    .box2 { order: 8; }
    .box3 { order: 1; background-color: #333; color: white; }
    .box4 { order: 2; }
    .box5 { order: 3; }
    .box6 { order: 4; }
    .box7 { order: 5; background-color: #333; color: white; }
    .box8 { order: 6; }
  </style>
</head>
<body>

<div class="container">
  <div class="box box1">1</div>
  <div class="box box2">2</div>
  <div class="box box3">3</div>
  <div class="box box4">4</div>
```

```
  <div class="box box5">5</div>
  <div class="box box6">6</div>
  <div class="box box7">7</div>
  <div class="box box8">8</div>
</div>

</body>
</html>
```

Output:-



5.Explain the difference between justify-items and justify-self using code examples.

🔹 `justify-items`

- **Applies to all items** inside a grid container.
- Used on the **container** to set alignment for all children.

🔹 `justify-self`

- **Applies to an individual grid item**.
- Used on a **child item** to override or customize alignment just for that item.

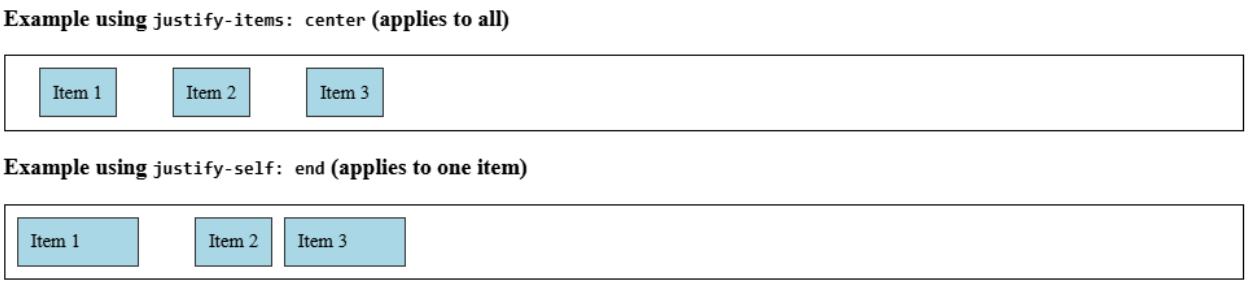
✅ Code Example Demonstrating Both

🔧 HTML (Same for both):

```
<!DOCTYPE html>
```

```
<html>
<head>
  <style>
    .grid {
      display: grid;
      grid-template-columns: repeat(3, 100px);
      gap: 10px;
      border: 2px solid #000;
      padding: 10px;
    }

    .item {
      background-color: lightblue;
      padding: 10px;
      border: 1px solid #333;
    }

    .grid-justify-items {
      justify-items: center; /* aligns all items horizontally at center */
    }

    .grid-justify-self .item2 {
      justify-self: end; /* aligns only item 2 to the right */
    }
  </style>
</head>
<body>

  <h3>Example using <code>justify-items: center</code> (applies to all)</h3>
  <div class="grid grid-justify-items">
    <div class="item">Item 1</div>
    <div class="item">Item 2</div>
    <div class="item">Item 3</div>
  </div>

  <h3>Example using <code>justify-self: end</code> (applies to one item)</h3>
  <div class="grid grid-justify-self">
    <div class="item">Item 1</div>
    <div class="item item2">Item 2</div>
    <div class="item">Item 3</div>
  </div>

</body>
</html>
```

Output:-



🧠 Summary:

Property	Where to Apply	Affects	Use Case
`justify-items`	On grid **container**	All items in grid	Set horizontal alignment for all
`justify-self`	On grid **item**	Single item	Override alignment for one item