In [1]:

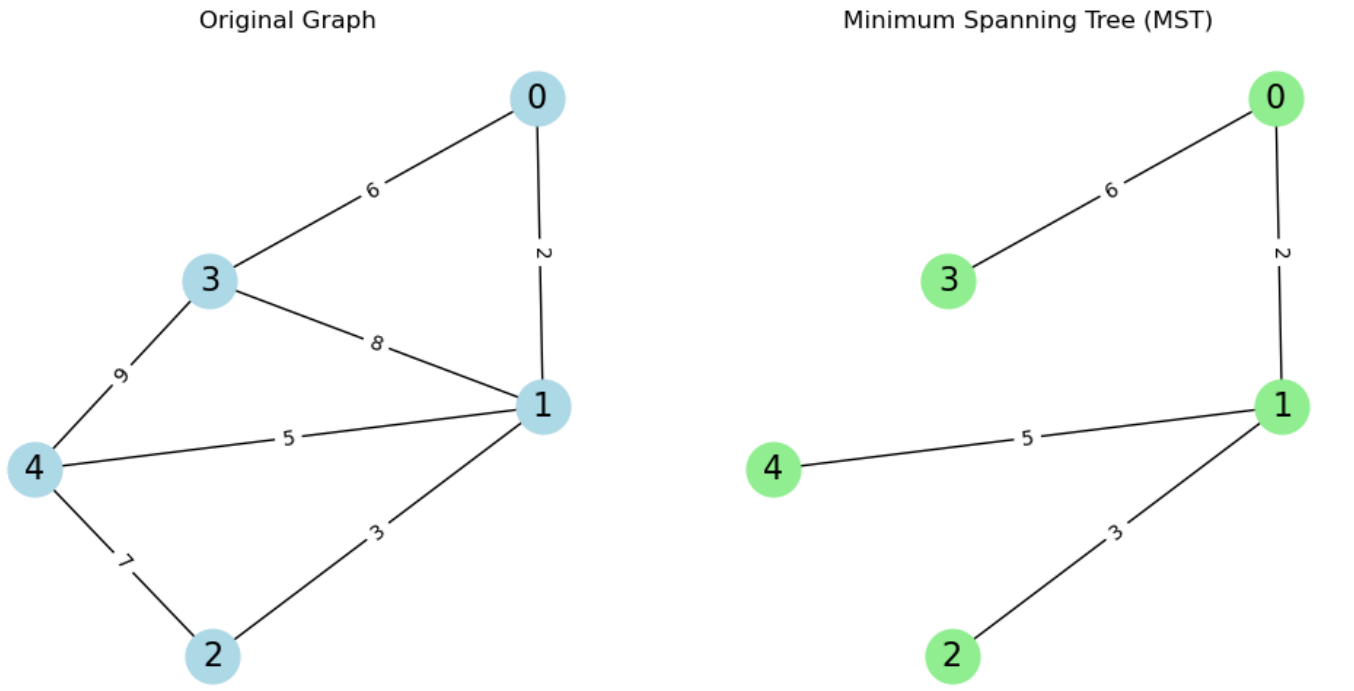
```
import numpy as np
import networkx as nx
import matplotlib.pypl as plt

class Graph:
    def __init__(self, adjacency_matrix):
        self.adjacency_matrix = adjacency_matrix
        self.num_vertices = len(adjacency_matrix)

    def prim(self):

        mst_edges = []
        visited = [False] * self.num_vertices
        visited[0] = True   b n

        for _ in range(self.num_vertices - 1):
            min_edge = (float('inf'), None, None)

            for u in range(self.num_vertices):
                if visited[u]:
                    for v in range(self.num_vertices):
                        if not visited[v] and self.adjacency_matrix[u][v] > 0:
                            weight = self.adjacency_matrix[u][v]
                            if weight < min_edge[0]:
                                min_edge = (weight, u, v)


            mst_edges.append((min_edge[1], min_edge[2], min_edge[0]))
            visited[min_edge[2]] = True

        return mst_edges


adjacency_matrix = np.array([
    [0, 2, 0, 6, 0],
    [2, 0, 3, 8, 5],
    [0, 3, 0, 0, 7],
    [6, 8, 0, 0, 9],
    [0, 5, 7, 9, 0]
])


graph = Graph(adjacency_matrix)
mst_edges = graph.prim()
print("Edges in the Minimum Spanning Tree:")
for u, v, weight in mst_edges:
    print(f"Edge: {u} - {v}, Weight: {weight}")
```

```
Edges in the Minimum Spanning Tree:
Edge: 0 - 1, Weight: 2
Edge: 1 - 2, Weight: 3
Edge: 1 - 4, Weight: 5
Edge: 0 - 3, Weight: 6
```

In [2]:

```
def draw_graph(adjacency_matrix, mst_edges):
    G = nx.Graph()
```

```
    for u in range(len(adjacency_matrix)):
        for v in range(len(adjacency_matrix)):
            if adjacency_matrix[u][v] > 0:
                G.add_edge(u, v, weight=adjacency_matrix[u][v])

    pos = nx.spring_layout(G)


    plt.figure(figsize=(12, 6))
    plt.subplot(121)
    nx.draw(G, pos, with_labels=True, node_color='lightblue', node_size=700, font_size=1
    edge_labels = nx.get_edge_attributes(G, 'weight')
    nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels)
    plt.title("Original Graph")

    T
    mst_graph = nx.Graph()
    mst_graph.add_weighted_edges_from(mst_edges)

    plt.subplot(122)
    nx.draw(mst_graph, pos, with_labels=True, node_color='lightgreen', node_size=700, fo
    mst_edge_labels = nx.get_edge_attributes(mst_graph, 'weight')
    nx.draw_networkx_edge_labels(mst_graph, pos, edge_labels=mst_edge_labels)
    plt.title("Minimum Spanning Tree (MST)")

    plt.show()


draw_graph(adjacency_matrix, mst_edges)
```

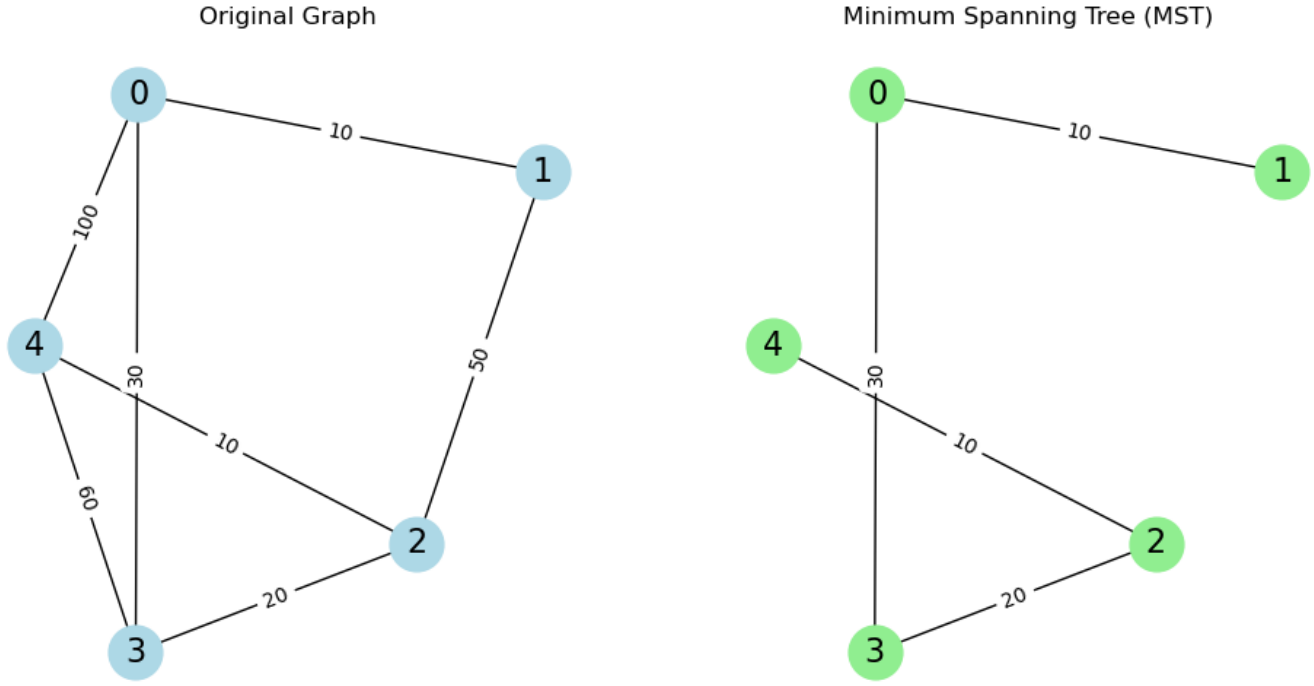


In [3]:

```
real_world_adjacency_matrix = np.array([
    [0, 10, 0, 30, 100],
    [10, 0, 50, 0, 0],
    [0, 50, 0, 20, 10],
```

```
    [30, 0, 20, 0, 60],
    [100, 0, 10, 60, 0]
])


real_world_graph = Graph(real_world_adjacency_matrix)
real_world_mst_edges = real_world_graph.prim()
print("Edges in the Minimum Spanning Tree for the real-world graph:")
for u, v, weight in real_world_mst_edges:
    print(f"Edge: {u} - {v}, Weight: {weight}")


draw_graph(real_world_adjacency_matrix, real_world_mst_edges)
```

```
Edges in the Minimum Spanning Tree for the real-world graph:
Edge: 0 - 1, Weight: 10
Edge: 0 - 3, Weight: 30
Edge: 3 - 2, Weight: 20
Edge: 2 - 4, Weight: 10
```



In [ ]: