NAME: - Rushikesh Sunil Mangalkar **ROLL NO-** 3202007

ASSIGNMENT 3

CODE:

```
def prims_algorithm(graph, num_vertices):
    mst = []
    visited = [False] * num_vertices
    edges = [(0, 0, -1)]

    while edges:
        edges.sort()
        weight, u, parent = edges.pop(0)

        if visited[u]:
            continue

        visited[u] = True

        if parent != -1:
            mst.append((parent, u, weight))

        for v in range(num_vertices):
            if not visited[v] and graph[u][v] != 0:
                edges.append((graph[u][v], v, u))

    return mst

def create_graph(num_vertices):
    graph = [[0] * num_vertices for _ in range(num_vertices)]

    print(f"Enter the connection weights for {num_vertices} vertices:")

    for i in range(num_vertices):
        for j in range(i + 1, num_vertices):
            weight = int(input(f"Connection between vertex {i} and vertex {j}, Weight: "))
            graph[i][j] = weight
            graph[j][i] = weight

    return graph

num_vertices = int(input("Enter the number of nodes/entities: "))
graph = create_graph(num_vertices)
mst = prims_algorithm(graph, num_vertices)

print("\nMinimum Spanning Tree (MST):")
for edge in mst:
    print(f"Edge: {edge[0]} - {edge[1]}, Weight: {edge[2]}")
```

OUTPUT:

```
Enter the number of nodes/entities: 5
Enter the connection weights for 5 vertices:
Connection between vertex 0 and vertex 1, Weight: 5
Connection between vertex 0 and vertex 2, Weight: 4
Connection between vertex 0 and vertex 3, Weight: 1
Connection between vertex 0 and vertex 4, Weight: 2
Connection between vertex 1 and vertex 2, Weight: 3
Connection between vertex 1 and vertex 3, Weight: 7
Connection between vertex 1 and vertex 4, Weight: 8
Connection between vertex 2 and vertex 3, Weight: 6
Connection between vertex 2 and vertex 4, Weight: 5
Connection between vertex 3 and vertex 4, Weight: 3

Minimum Spanning Tree (MST):
Edge: 0 - 3, Weight: 1
Edge: 0 - 4, Weight: 2
Edge: 0 - 2, Weight: 4
Edge: 2 - 1, Weight: 3
```