F. Case Study 6: Spanning Trees - Electrical Grid Design

6. Write a Python program to compute the minimum spanning tree for a given graph.

```
import networkx as nx


def kruskal_mst(graph):
    # Ensure the graph is connected
    if not nx.is_connected(graph):
        raise ValueError("The graph must be connected.")

    mst = nx.Graph()  # Create an empty graph for the MST
    edges = sorted(graph.edges(data=True), key=lambda x: x[2]['weight'])
    disjoint_set = {node: node for node in graph.nodes}

    def find(node):
        if disjoint_set[node] != node:
            disjoint_set[node] = find(disjoint_set[node])  # Path compression
        return disjoint_set[node]

    def union(node1, node2):
        root1 = find(node1)
        root2 = find(node2)
        if root1 != root2:
```

```
            disjoint_set[root2] = root1  # Union the sets

    for u, v, data in edges:
        if find(u) != find(v):  # Check for cycle
            mst.add_edge(u, v, weight=data['weight'])
            union(u, v)
        if len(mst.edges) == len(graph.nodes) - 1:  # Stop when n-1 edges are added
            break

    return mst

# Example Usage
if __name__ == "__main__":
    G = nx.Graph()
    G.add_weighted_edges_from([
        (0, 1, 4), (0, 2, 6), (1, 2, 6),
        (1, 3, 3), (2, 3, 2), (2, 4, 5),
        (3, 4, 7)
    ])
    mst = kruskal_mst(G)
    print("Minimum Spanning Tree Edges:", list(mst.edges(data=True)))
```