```
def find(parent, i):
    if parent[i] != i:
        parent[i] = find(parent, parent[i])
    return parent[i]

def union(parent, rank, x, y):
    xroot = find(parent, x)
    yroot = find(parent, y)
    if rank[xroot] < rank[yroot]:
        parent[xroot] = yroot
    elif rank[xroot] > rank[yroot]:
        parent[yroot] = xroot
    else:
        parent[yroot] = xroot
        rank[xroot] += 1

def kruskal(n, edges):
    parent = [i for i in range(n+1)]
    rank = [0] * (n+1)
    edges = sorted(edges, key=lambda x: x[2])
    mst = []
    total = 0
    for u, v, w in edges:
        x = find(parent, u)
        y = find(parent, v)
        if x != y:
            mst.append((u, v, w))
            total += w
            union(parent, rank, x, y)
    return mst, total

def main():
    n = int(input("Enter number of vertices: "))
    m = int(input("Enter number of edges: "))
    edges = []
    print("Enter each edge in format: u v weight")
    for _ in range(m):
        u, v, w = map(int, input().split())
        edges.append((u, v, w))
    mst_edges, cost = kruskal(n, edges)
    print("Edges in MST (u, v, weight):")
    for e in mst_edges:
        print(e)
    print("Total cost of MST:", cost)
```

```
if __name__ == "__main__":
    main()
```