Code:

```
#include <bits/stdc++.h>
using namespace std;

const int MAXM = 200005;
const int MAXN = 100005;

int parent[MAXN];
int rankA[MAXN];
int u[MAXM], v[MAXM], w[MAXM];

int find(int x) {

  if (parent[x] == x) {
    return x;
  }

  return parent[x] = find(parent[x]);
}
void merge(int x, int y) {

  x = find(x);
  y = find(y);

  if (x == y) {
    return;
  }

  if (rankA[x] < rankA[y]) {
    swap(x, y);
  }

  parent[y] = x;

  if (rankA[x] == rankA[y]) {
    rankA[x]++;
  }
}

int main() {

  int n, m, mst_cost = 0;
  cout << "Enter the number of nodes and edges in the graph: " << endl;
  cin >> n >> m;

  for (int i = 1; i <= n; i++) {
    parent[i] = i;
```

```
  }

  cout << "\nEnter the endpoints and weight: " << endl;
  for (int i = 1; i <= m; i++) {
    cin >> u[i] >> v[i] >> w[i];
  }

  for (int i = 1; i <= m; i++) {
    for (int j = i + 1; j <= m; j++) {
      if (w[i] > w[j]) {
        swap(u[i], u[j]);
        swap(v[i], v[j]);
        swap(w[i], w[j]);
      }
    }
  }

  cout << "\nThe edges are: \n";
  for (int i = 1; i <= m; i++) {
    if (find(u[i]) != find(v[i])) {
      merge(u[i], v[i]);
      mst_cost += w[i];
      cout << u[i] << " " << v[i] << " " << w[i] << "\n";
    }
  }

  cout << "\nThe minimum spanning tree cost: " << mst_cost;

  return 0;
}
```

Sample Output:

```
Enter the number of nodes and edges in the graph:
5 7

Enter the endpoints and weight:
1 2 1
1 3 3
2 4 6
2 3 3
3 4 4
3 5 2
4 5 5
```

The edges are:
1 2 1
3 5 2
2 3 3
3 4 4

The minimum spanning tree cost: 10