# Final

February 27, 2025

[1]:
```
import ROOT
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[2]:
```
file = ROOT.TFile.Open("muons.root")
tree = file.Get("t3333")
```

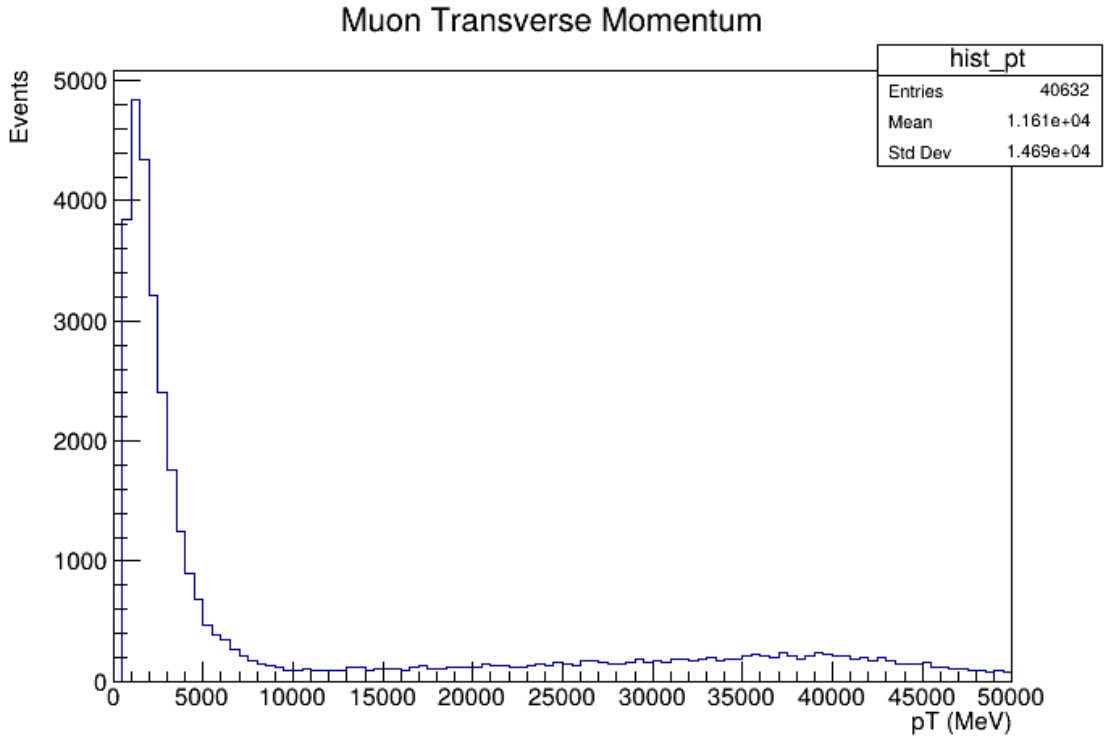
```
Error in <TNetXNGFile::Open>: [ERROR] Server responded with an error: [3001]
Required argument not present
```

[3]:
```
#Q1
hist_pt = ROOT.TH1F("hist_pt", "Muon Transverse Momentum; pT (MeV); Events",␣
 ↪100, 0, 50000)
hist_theta = ROOT.TH1F("hist_theta", "Muon Theta Angle; Theta (rad); Events",␣
 ↪100, 0, 3.14)
hist_phi = ROOT.TH1F("hist_phi", "Muon Phi Angle; Phi (rad); Events", 100, -3.
 ↪14, 3.14)
hist_eta = ROOT.TH1F("hist_eta", "Muon Pseudorapidity; Eta; Events", 100, -3, 3)
hist_p = ROOT.TH1F("hist_p", "Muon Momentum; p (MeV); Events", 100, 0, 200000)
hist_energy = ROOT.TH1F("hist_energy", "Muon Energy; E (MeV); Events", 100, 0,␣
 ↪200000)
```

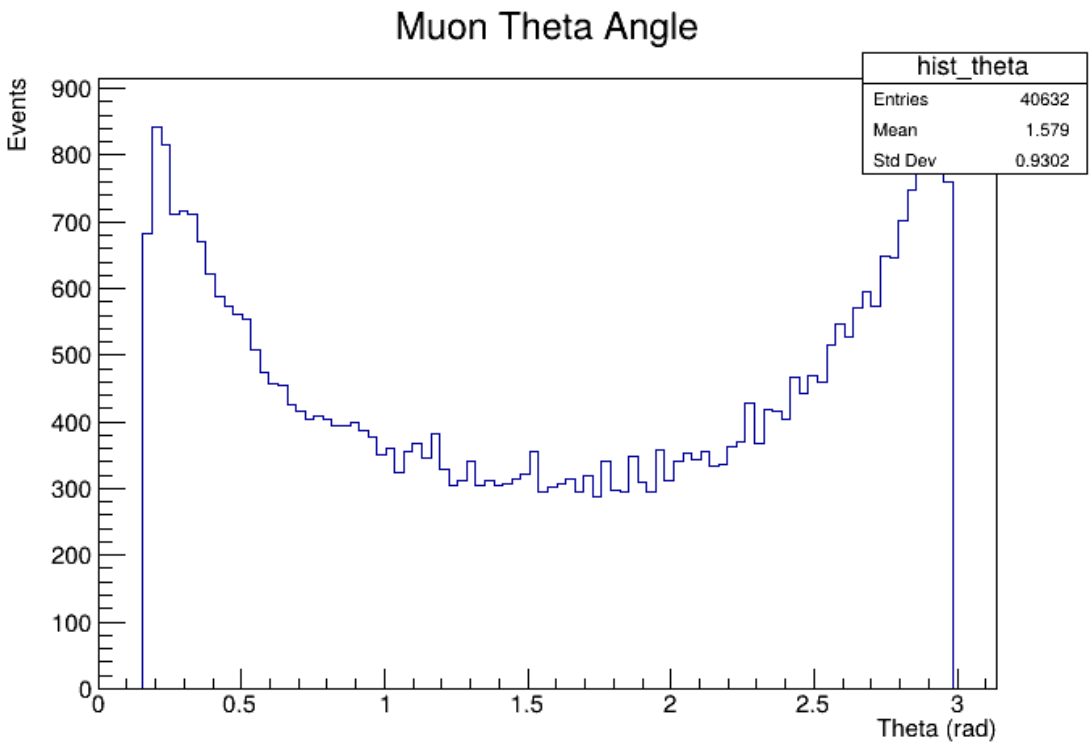
[4]:
```
for event in tree:
    for i in range(event.NMUO):
        px, py, pz, E = event.PXMUO[i], event.PYMUO[i], event.PZMUO[i], event.
 ↪EEMUO[i]
        pT = np.sqrt(px**2 + py**2)
        p = np.sqrt(px**2 + py**2 + pz**2)
        theta = np.arccos(pz / p) if p != 0 else 0
        eta = -np.log(np.tan(theta / 2)) if theta > 0 else 0
        phi = np.arctan2(py, px)

        hist_pt.Fill(pT)
        hist_theta.Fill(theta)
        hist_phi.Fill(phi)
        hist_eta.Fill(eta)
```

```
        hist_p.Fill(p)
        hist_energy.Fill(E)
```

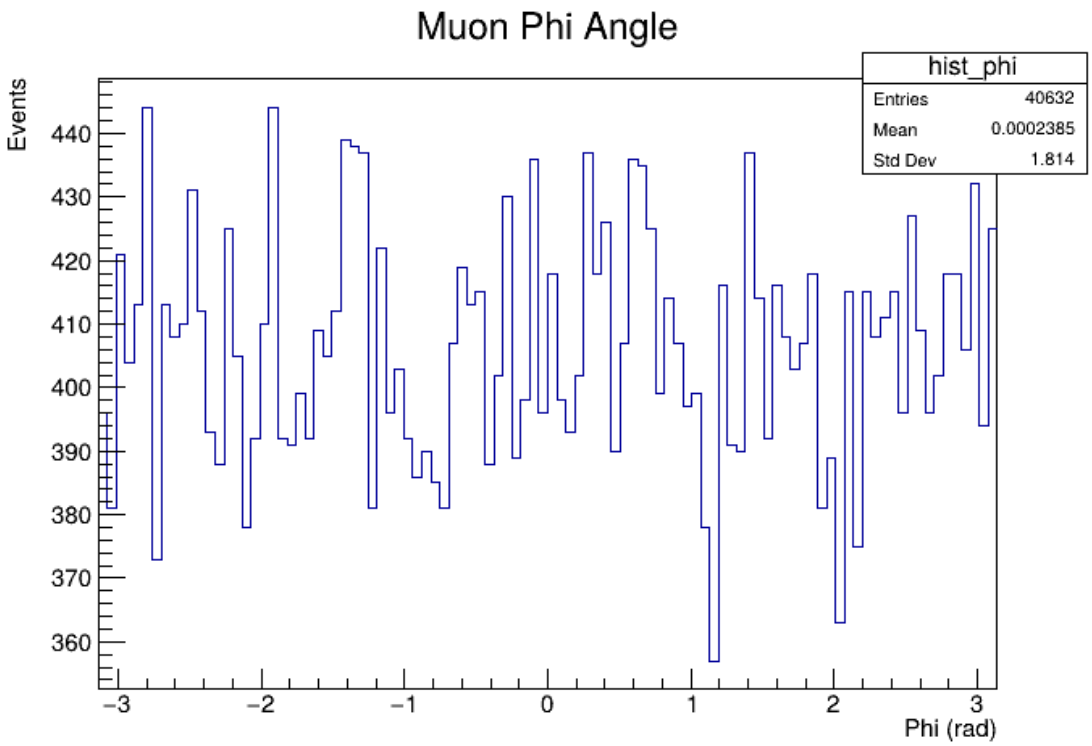
[5]:
```
canvas_q1_pt = ROOT.TCanvas("canvas_q1_pt", "Muon Properties")
hist_pt.Draw()
canvas_q1_pt.Draw()
```



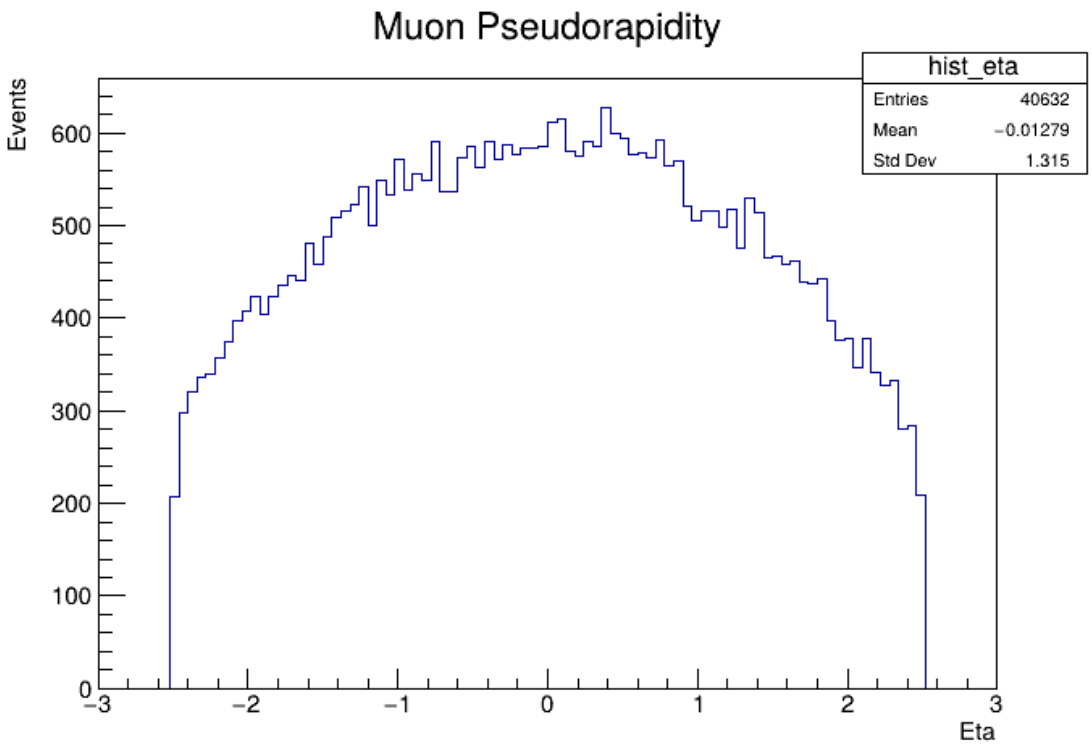

[6]:
```
canvas_q1_theta = ROOT.TCanvas("canvas_q1_theta", "Muon Theta")
hist_theta.Draw()
canvas_q1_theta.Draw()
```



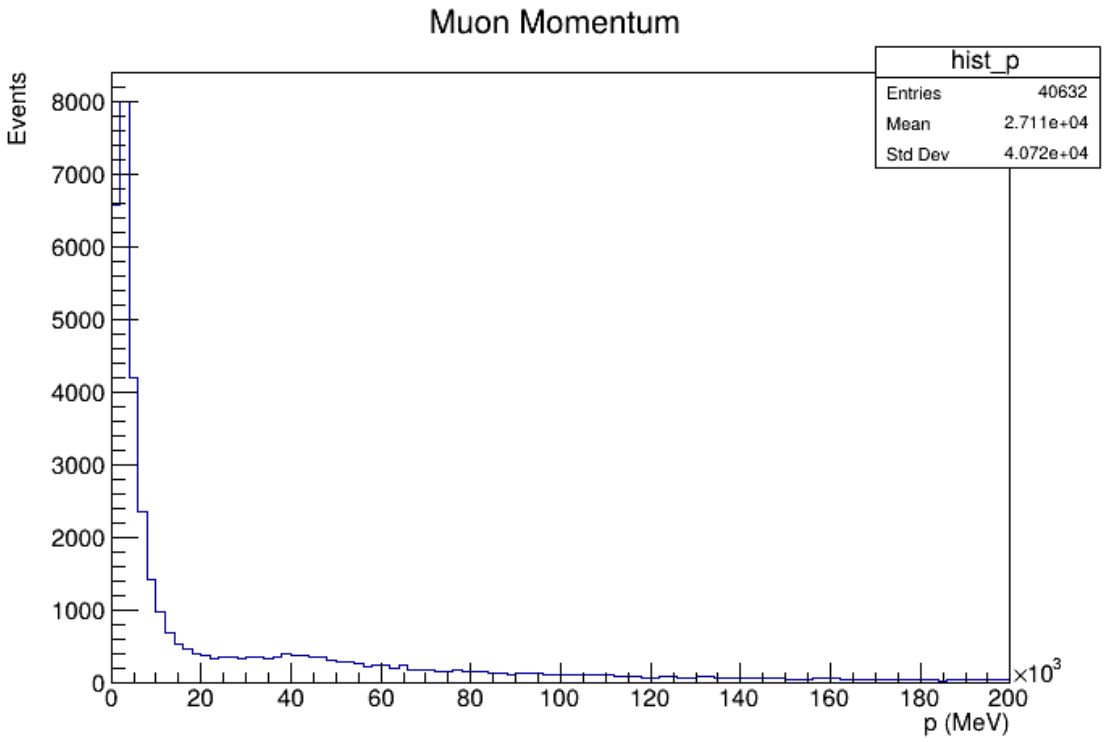

[7]:
```
canvas_q1_phi = ROOT.TCanvas("canvas_q1_phi", "Muon Phi")
hist_phi.Draw()
canvas_q1_phi.Draw()
```



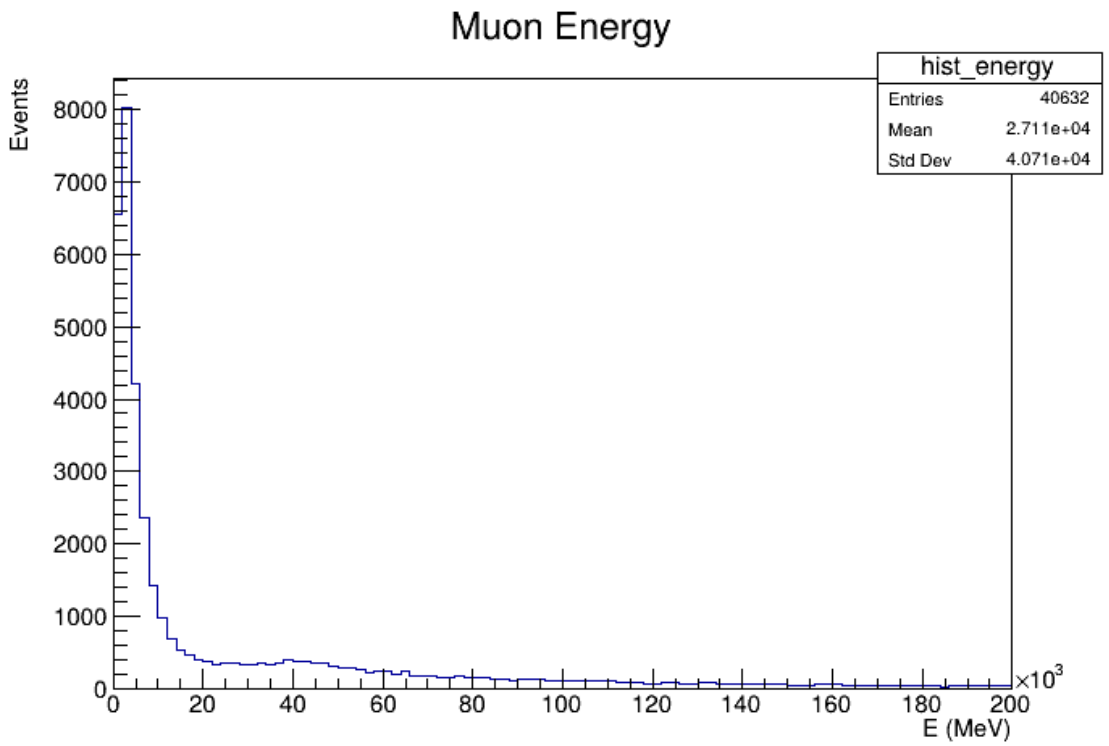

[8]:
```
canvas_q1_eta = ROOT.TCanvas("canvas_q1_eta", "Muon Pseudorapidity")
hist_eta.Draw()
canvas_q1_eta.Draw()
```




[9]:
```
canvas_q1_p = ROOT.TCanvas("canvas_q1_p", "Muon Momentum")
hist_p.Draw()
canvas_q1_p.Draw()
```




[10]:
```
canvas_q1_energy = ROOT.TCanvas("canvas_q1_energy", "Muon Energy")
hist_energy.Draw()
canvas_q1_energy.Draw()
```



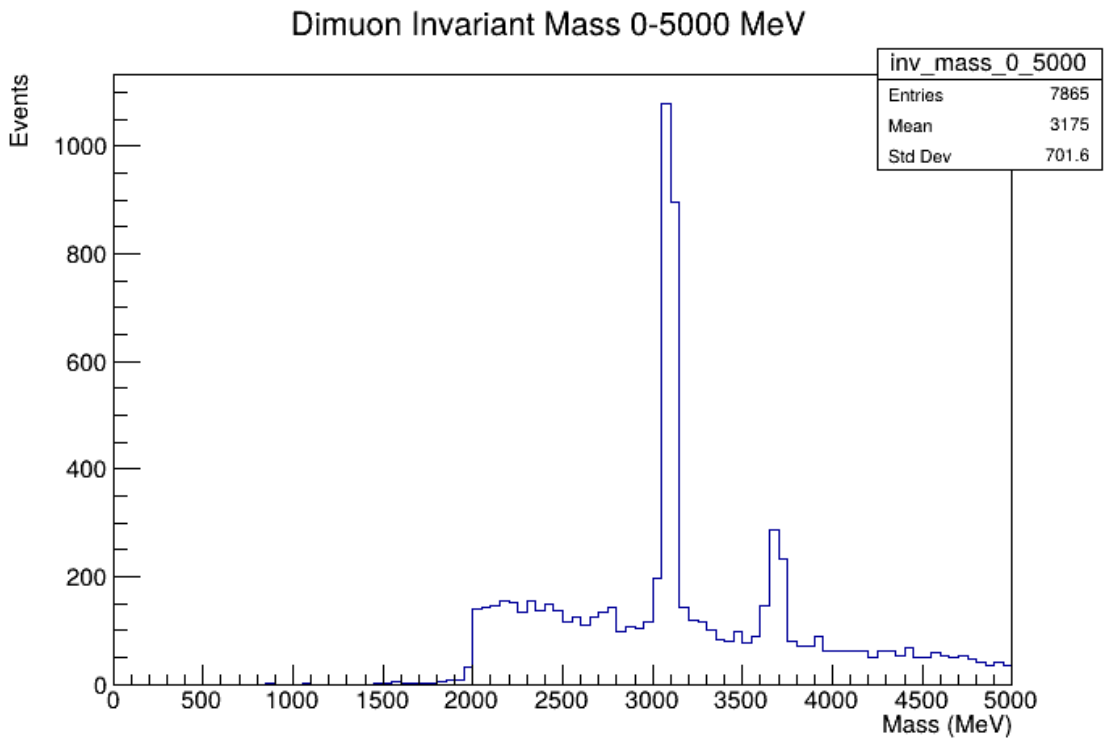

[28]:
```
#Q2
mass_ranges = [(0., 5000.), (5000., 15000.), (15000., 125000.), (100000.,␣
 ↪200000.), (200000., 400000.), (400000., 1400000.)]
hist_mass_0_5000 = ROOT.TH1F("inv_mass_0_5000", "Dimuon Invariant Mass 0-5000␣
 ↪MeV; Mass (MeV); Events", 100, 0, 5000)
hist_mass_5000_15000 = ROOT.TH1F("inv_mass_5000_15000", "Dimuon Invariant Mass␣
 ↪5000-15000 MeV; Mass (MeV); Events", 100, 5000, 15000)
hist_mass_15000_125000 = ROOT.TH1F("inv_mass_15000_125000", "Dimuon Invariant␣
 ↪Mass 15000-125000 MeV; Mass (MeV); Events", 100, 15000, 125000)
hist_mass_100000_200000 = ROOT.TH1F("inv_mass_100000_200000", "Dimuon Invariant␣
 ↪Mass 100000-200000 MeV; Mass (MeV); Events", 100, 100000, 200000)
hist_mass_200000_400000 = ROOT.TH1F("inv_mass_200000_400000", "Dimuon Invariant␣
 ↪Mass 200000-400000 MeV; Mass (MeV); Events", 100, 200000, 400000)
hist_mass_400000_1400000 = ROOT.TH1F("inv_mass_400000_1400000", "Dimuon␣
 ↪Invariant Mass 400000-1400000 MeV; Mass (MeV); Events", 100, 400000, 1200000)
```

```
Warning in <TFile::Append>: Replacing existing TH1: inv_mass_0_5000 (Potential
memory leak).
Warning in <TFile::Append>: Replacing existing TH1: inv_mass_5000_15000
(Potential memory leak).
Warning in <TFile::Append>: Replacing existing TH1: inv_mass_15000_125000
(Potential memory leak).
Warning in <TFile::Append>: Replacing existing TH1: inv_mass_100000_200000
```

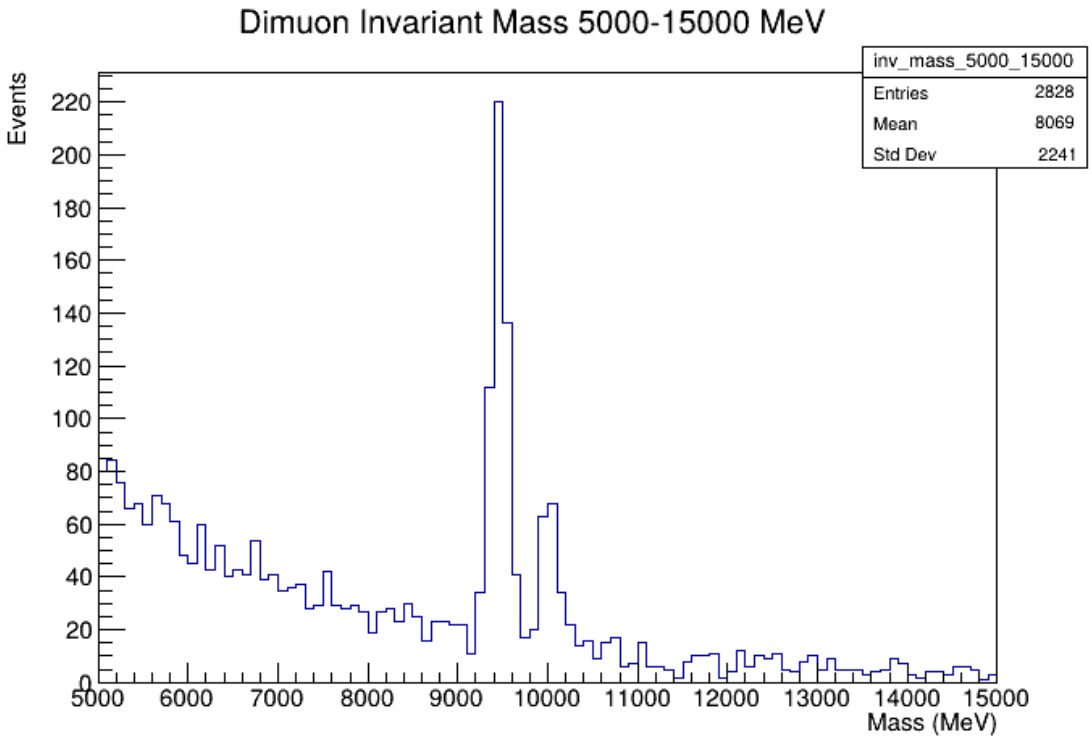
```
(Potential memory leak).
Warning in <TFile::Append>: Replacing existing TH1: inv_mass_200000_400000
(Potential memory leak).
Warning in <TFile::Append>: Replacing existing TH1: inv_mass_400000_1400000
(Potential memory leak).
```

[29]:
```
for event in tree:
    if event.NMUO == 2 and event.KFMUO[0] * event.KFMUO[1] < 0:  # Opposite␣
 ↪charge
        p1 = np.array([event.EEMUO[0], event.PXMUO[0], event.PYMUO[0], event.
 ↪PZMUO[0]])
        p2 = np.array([event.EEMUO[1], event.PXMUO[1], event.PYMUO[1], event.
 ↪PZMUO[1]])
        mass = np.sqrt((p1[0] + p2[0])**2 - (p1[1] + p2[1])**2 - (p1[2] +␣
 ↪p2[2])**2 - (p1[3] + p2[3])**2)

        if 0. <= mass < 5000.: hist_mass_0_5000.Fill(mass)
        elif 5000. <= mass < 15000.: hist_mass_5000_15000.Fill(mass)
        elif 15000. <= mass < 125000.: hist_mass_15000_125000.Fill(mass)
        elif 100000. <= mass < 200000.: hist_mass_100000_200000.Fill(mass)
        elif 200000. <= mass < 400000.: hist_mass_200000_400000.Fill(mass)
        elif 400000. <= mass < 1400000.: hist_mass_400000_1400000.Fill(mass)
```

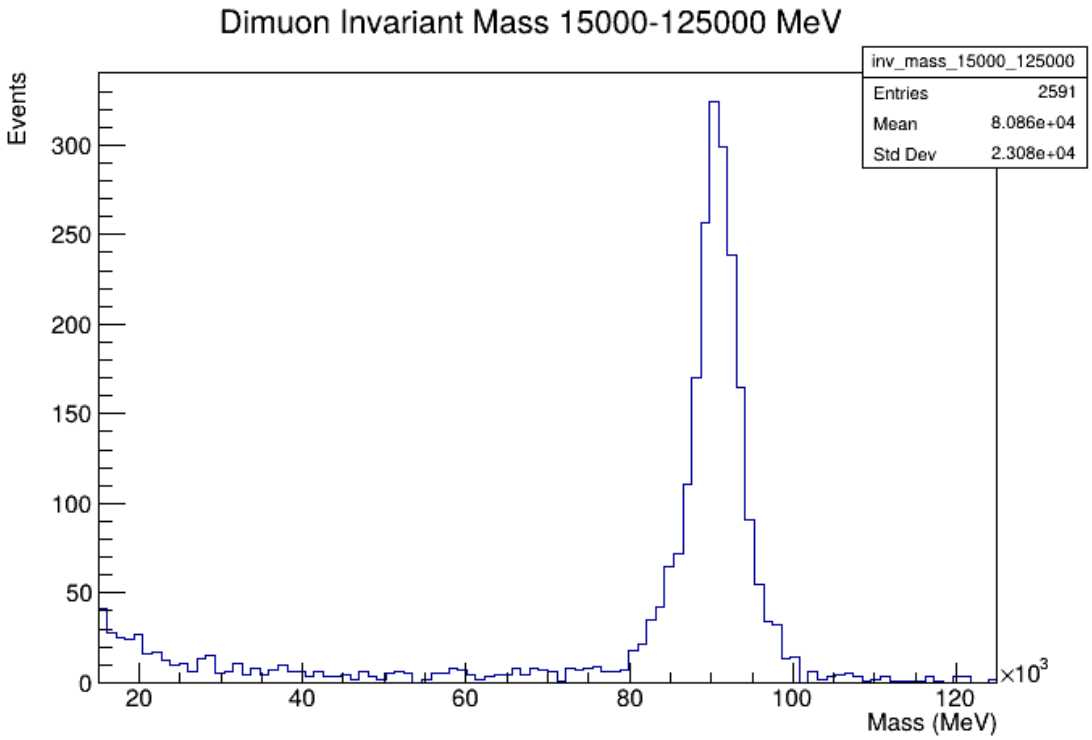
[30]:
```
canvas_q2_0_5000 = ROOT.TCanvas()
hist_mass_0_5000.Draw()
canvas_q2_0_5000.Draw()
```



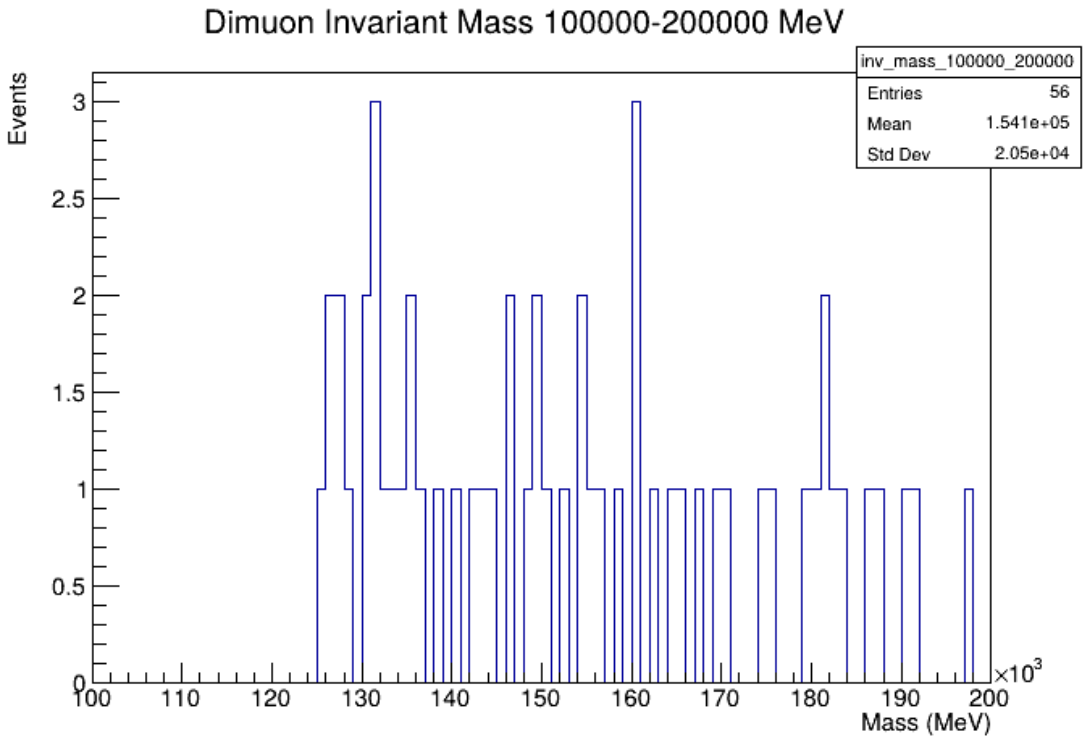

[31]:
```
canvas_q2_5000_15000 = ROOT.TCanvas()
hist_mass_5000_15000.Draw()
canvas_q2_5000_15000.Draw()
```



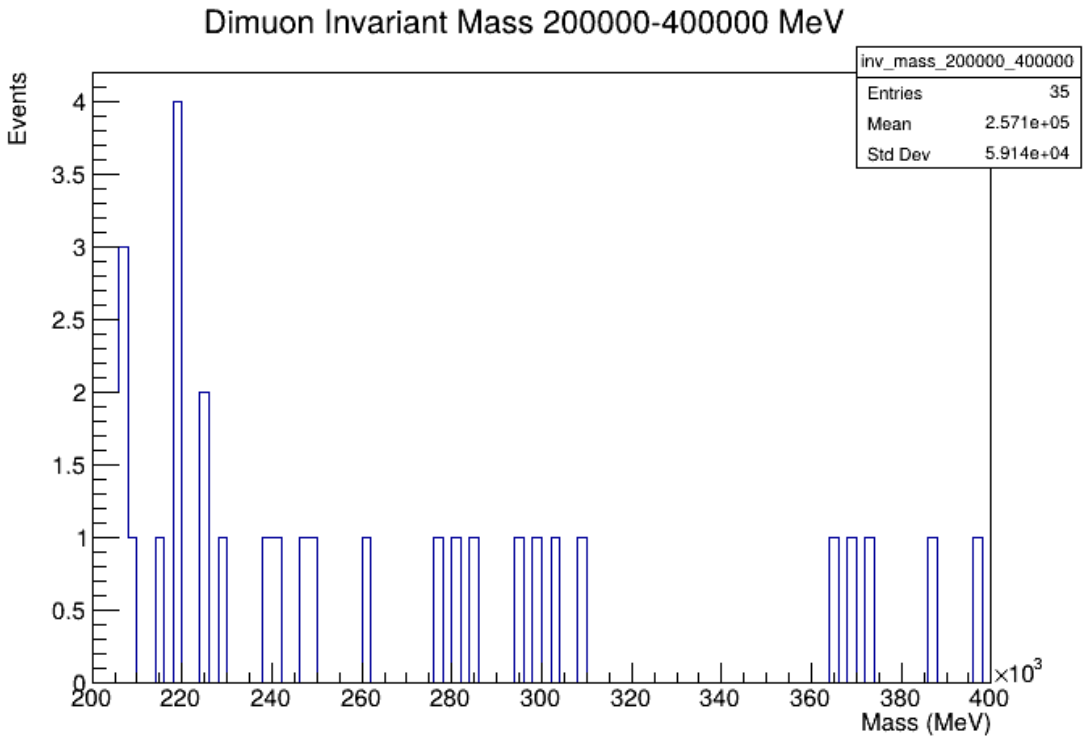

[32]:
```
canvas_q2_15000_125000 = ROOT.TCanvas()
hist_mass_15000_125000.Draw()
canvas_q2_15000_125000.Draw()
```



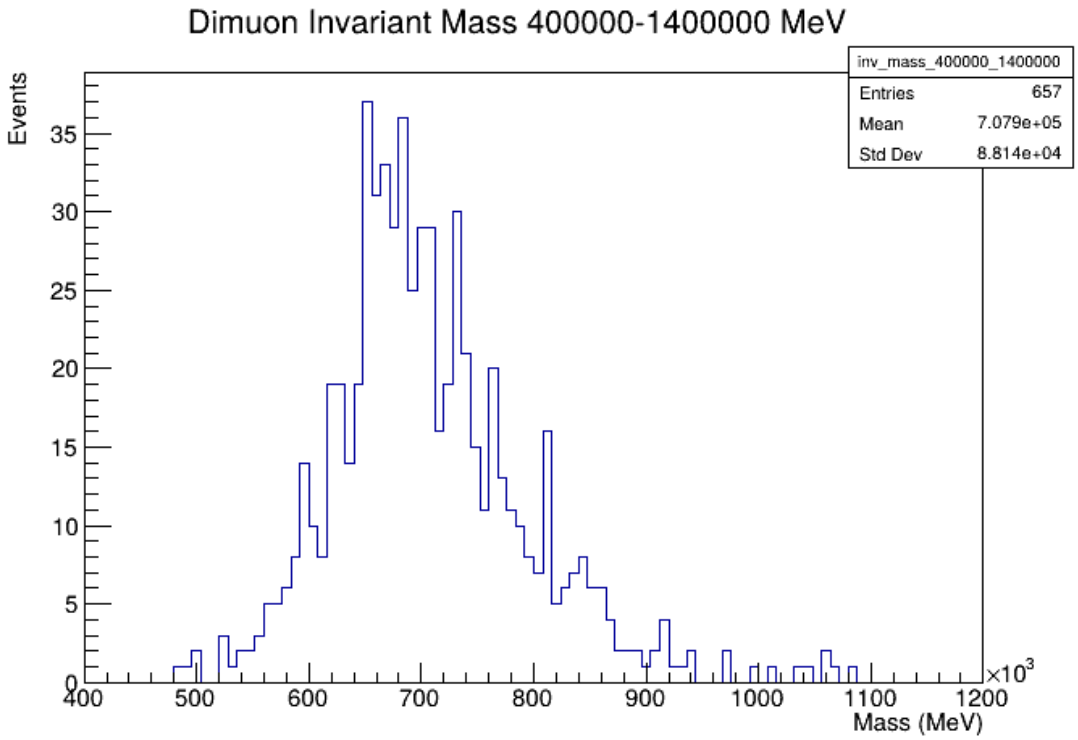

[33]:
```
canvas_q2_100000_200000 = ROOT.TCanvas()
hist_mass_100000_200000.Draw()
canvas_q2_100000_200000.Draw()
```




[34]:
```
canvas_q2_200000_400000 = ROOT.TCanvas()
hist_mass_200000_400000.Draw()
canvas_q2_200000_400000.Draw()
```



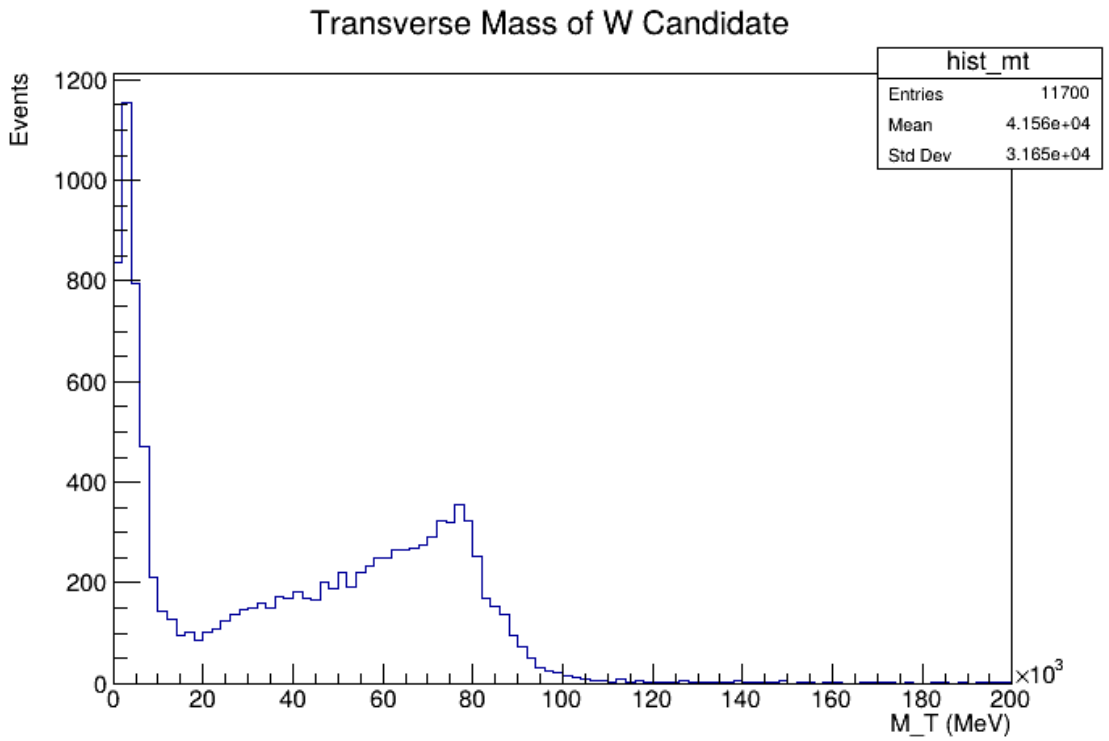

[35]:
```
canvas_q2_400000_1400000 = ROOT.TCanvas()
hist_mass_400000_1400000.Draw()
canvas_q2_400000_1400000.Draw()
```




[19]:
```
#Q3
hist_mt = ROOT.TH1F("hist_mt", "Transverse Mass of W Candidate; M_T (MeV);␣
↪Events", 100, 0, 200000)
```

[20]:
```
for event in tree:
    if event.NMUO == 1:
        px_mu, py_mu = event.PXMUO[0], event.PYMUO[0]
        pT_mu = np.sqrt(px_mu**2 + py_mu**2)
        phi_mu = np.arctan2(py_mu, px_mu)

        px_miss, py_miss = event.PXMISS, event.PYMISS
        ET_miss = np.sqrt(px_miss**2 + py_miss**2)
        phi_miss = np.arctan2(py_miss, px_miss)

        delta_phi = abs(phi_mu - phi_miss)
        if delta_phi > np.pi:
            delta_phi = 2 * np.pi - delta_phi

        mt = np.sqrt(2 * pT_mu * ET_miss * (1 - np.cos(delta_phi)))
        hist_mt.Fill(mt)
```

[21]:
```
canvas_q3_mt = ROOT.TCanvas("canvas_q3_mt", "W Transverse Mass")
hist_mt.Draw()
canvas_q3_mt.Draw()
```




[36]:
```
#Q4
hist_new_mass = ROOT.TH1F("hist_new_mass", "Invariant Mass of New Particle␣
↪Candidates; Mass (MeV); Events", 100, 0, 1200000)
```

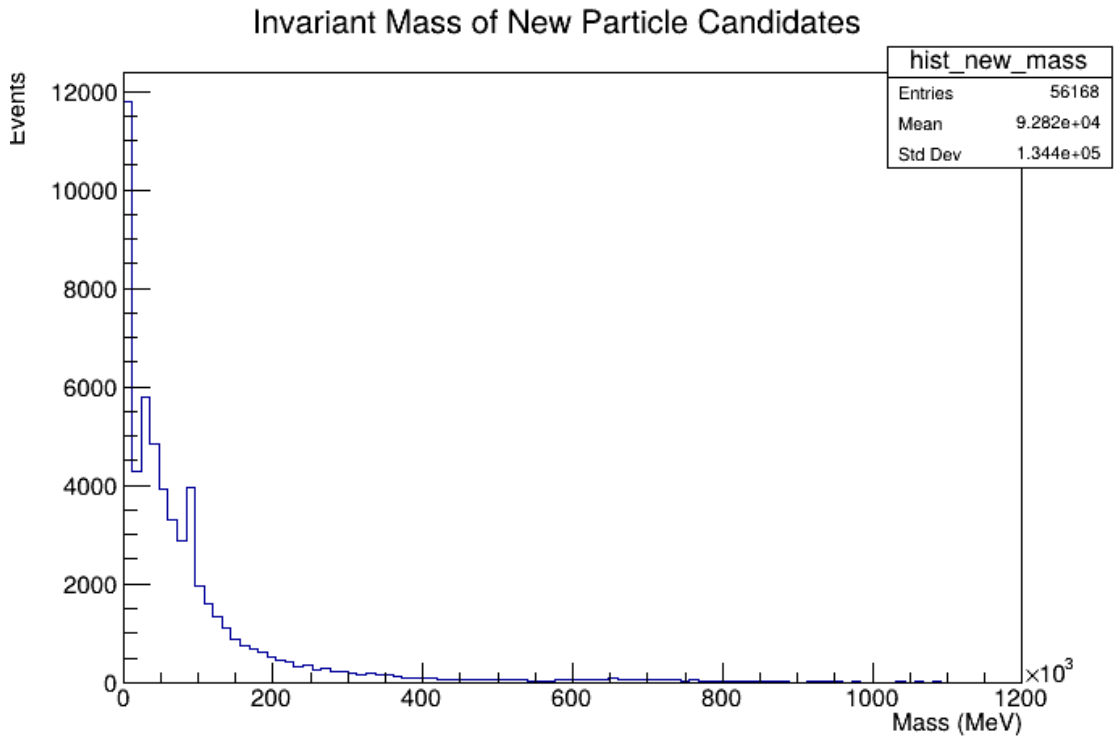
```
Warning in <TFile::Append>: Replacing existing TH1: hist_new_mass (Potential
memory leak).
```

[37]:
```
for event in tree:
    if event.NMUO == 2 and event.KFMUO[0] * event.KFMUO[1] < 0:
        p1 = np.array([event.EEMUO[0], event.PXMUO[0], event.PYMUO[0], event.
↪PZMUO[0]])
        p2 = np.array([event.EEMUO[1], event.PXMUO[1], event.PYMUO[1], event.
↪PZMUO[1]])
        mass = np.sqrt((p1[0] + p2[0])**2 - (p1[1] + p2[1])**2 - (p1[2] +␣
↪p2[2])**2 - (p1[3] + p2[3])**2)
        hist_new_mass.Fill(mass)

    if event.NELE == 2 and event.KFELE[0] * event.KFELE[1] < 0:
```

```
        p1 = np.array([event.EEELE[0], event.PXELE[0], event.PYELE[0], event.
↪PZELE[0]])
        p2 = np.array([event.EEELE[1], event.PXELE[1], event.PYELE[1], event.
↪PZELE[1]])
        mass = np.sqrt((p1[0] + p2[0])**2 - (p1[1] + p2[1])**2 - (p1[2] +␣
↪p2[2])**2 - (p1[3] + p2[3])**2)
        hist_new_mass.Fill(mass)

    if event.NJET >= 2:
        for i in range(event.NJET):
            for j in range(i+1, event.NJET):
                p1 = np.array([event.EEJET[i], event.PXJET[i], event.PYJET[i],␣
↪event.PZJET[i]])
                p2 = np.array([event.EEJET[j], event.PXJET[j], event.PYJET[j],␣
↪event.PZJET[j]])
                mass = np.sqrt((p1[0] + p2[0])**2 - (p1[1] + p2[1])**2 - (p1[2]␣
↪+ p2[2])**2 - (p1[3] + p2[3])**2)
                hist_new_mass.Fill(mass)
```

[38]:
```
canvas_q4_new = ROOT.TCanvas("canvas_q4_new", "New Particle Search")
hist_new_mass.Draw()
canvas_q4_new.Draw()
```

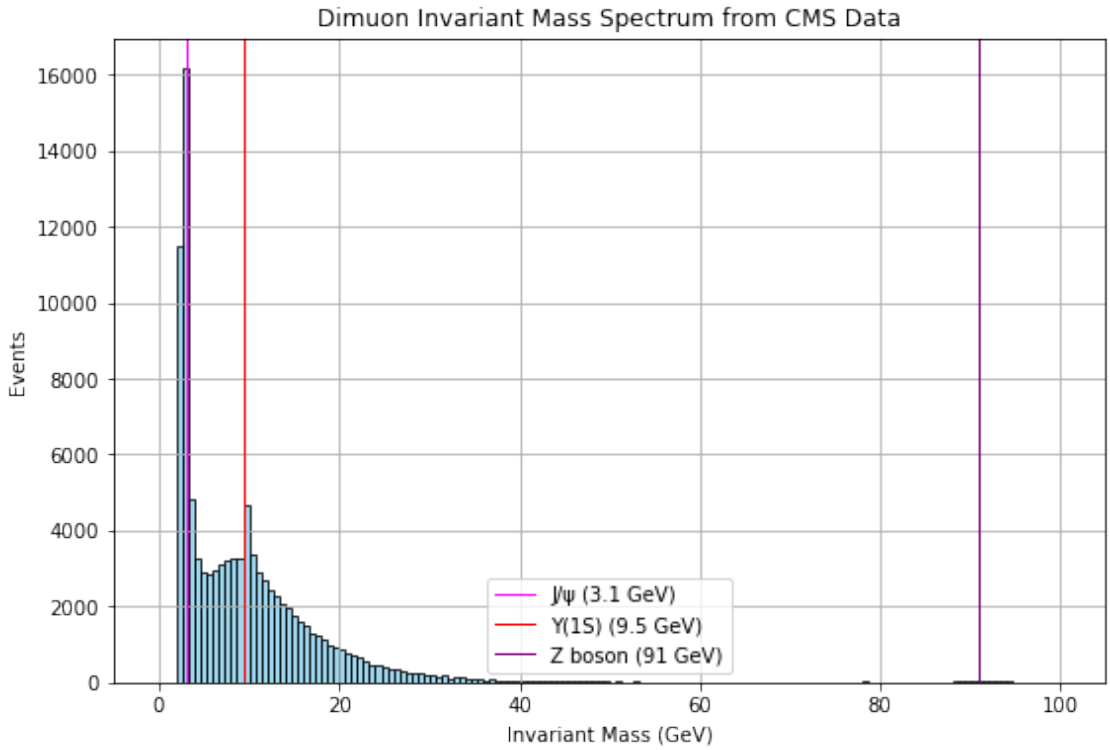
```
Warning in <TCanvas::Constructor>: Deleting canvas with same name: canvas_q4_new
```



[25]:
```
#Q5
cms = pd.read_csv("MuRun2010B_withoutM.csv", sep="\\s+")
```

[26]:
```
energy_sum = cms["E1"] + cms["E2"]
px_sum = cms["px1"] + cms["px2"]
py_sum = cms["py1"] + cms["py2"]
pz_sum = cms["pz1"] + cms["pz2"]
cms["invariant_mass"] = np.sqrt(energy_sum**2 - px_sum**2 - py_sum**2 -␣
 ↪pz_sum**2)
```

[27]:
```
plt.figure(figsize=(9, 6))
plt.hist(cms["invariant_mass"], bins=150, range=(0, 100), alpha=0.8,␣
 ↪color='skyblue', edgecolor='black')
plt.title("Dimuon Invariant Mass Spectrum from CMS Data")
plt.xlabel("Invariant Mass (GeV)")
plt.ylabel("Events")

plt.axvline(3.1, color='magenta', linewidth=1, label="J/  (3.1 GeV)")
plt.axvline(9.5, color='red', linewidth=1, label="ϒ(1S) (9.5 GeV)")
plt.axvline(91, color='purple', linewidth=1, label="Z boson (91 GeV)")

plt.legend()
plt.grid(True)
plt.show()
```



[ ]:

[ ]:

[ ]: