Python Assessments.

# Section 1

# Data Preparation  
data = {  
 'Name': ['Abdul Karim', 'Badruddin', 'Jamaluddin', 'Zalekha', 'Jenifer Wong', 'Linda', 'Premala devi'],  
 'Age': [55, 25, 28, 51, 26, 23, 35],  
 'Height (cm)': [175, 169, 174, 160, 162, np.nan, 167],  
 'Weight (kg)': [85, 75, 95, 58, 60, 67, np.nan]  
}  
  
# Q1 - Using pandas create dataframe and print  
**# Points: 5 (Beginner)**df = pd.DataFrame(data)  
print(f"Q1 - Dataframe:\n{df}")  
  
# Q2 - Replace NaN with 0  
**# Points: 5 (Beginner)**df\_filled = df.fillna(0)  
print('\nQ2 - New DataFrame after replacing NaN with 0:')  
print(df\_filled)  
  
# Q3 - Modify the DataFrame to add a new column 'BMI' using the formula: weight (kg) / (height in m)^2. Handle rows with missing data appropriately (NaN).  
**# Points: 12 (Intermediate)**print('\nQ3 - Remove NaN, add columns Height (m) and BMI:')  
df.dropna(subset=['Height (cm)', 'Weight (kg)'], inplace=True)  
df['Height (m)'] = df['Height (cm)']/100  
df['BMI'] = df['Weight (kg)']/df['Height (m)']\*\*2  
print(df)

Section 2  
# Q4 - Load data from production\_output.csv  
**# Points: 8 (Intermediate)**prod\_data = pd.read\_csv('production\_output.csv')  
print(f"\nQ4 - Load production data from csv:")  
print(prod\_data)  
  
# Q5 - Total output per day (Group by date and sum output/input)  
**# Points: 15 (Intermediate)**df\_prod\_data\_day = prod\_data.groupby(by=['date'])[['output','input']].sum().reset\_index()  
df\_prod\_data\_day.rename(columns={'output': 'Total\_output', 'input': 'Total\_input'}, inplace=True)  
print(f"\nQ5 - Total output and input by day:\n{df\_prod\_data\_day}")  
  
# Q6 - Average output per hour (assuming 24 hours of operation per day)  
**# Points: 15 (Intermediate)**df\_prod\_data\_day['Average output per hour'] = (df\_prod\_data\_day['Total\_output']/24).round(3)  
print("\nQ6 - Average output per hour:")  
print(df\_prod\_data\_day.to\_string(index=False))  
  
# Q6 Add column yield percentage (Total output / Total input) \* 100  
**# Points: 15 (Intermediate)**df\_prod\_data\_day['Yield (%)'] = (df\_prod\_data\_day['Total\_output'] / df\_prod\_data\_day['Total\_input'] \* 100).round(2)  
print("\nAdd column yield percentage")  
print(df\_prod\_data\_day)  
  
# Q7 - Plotting with multiple y-axes (Bar chart and Line plot)  
**# Points: 40 (Advanced)**# Step 2: Set up figure and axes  
fig, ax1 = plt.subplots(figsize=(12, 6))  
  
# Bar chart: Total Output  
ax1.bar(df\_prod\_data\_day['date'], df\_prod\_data\_day['Total\_output'], color='skyblue', label='Total Output')  
ax1.set\_xlabel('Date')  
ax1.set\_ylabel('Total Output', color='skyblue')  
ax1.tick\_params(axis='y', labelcolor='skyblue')  
ax1.set\_xticklabels(df\_prod\_data\_day['date'], rotation=90)  
  
# Step 3: Add a secondary y-axis for Yield (%)  
ax2 = ax1.twinx()  
ax2.plot(df\_prod\_data\_day['date'], df\_prod\_data\_day['Yield (%)'], color='orange', marker='o', label='Yield (%)')  
ax2.set\_ylabel('Yield (%)', color='orange')  
ax2.tick\_params(axis='y', labelcolor='orange')  
ax2.set\_ylim(70, 100)  
  
# Title and layout  
plt.title('Daily Total Output and Yield Percentage')  
fig.tight\_layout()  
  
# Save before plt.show() (best practice)  
plt.savefig('daily\_output\_chart.png', dpi=300)  
  
# Show plot  
plt.show()