Data Manipulation (Pandas) - Solutions

1. **DataFrame Creation and Manipulation Solution**

```
import pandas as pd
data = {'Stock': ['AAPL', 'GOOGL', 'MSFT', 'AMZN'],
        'Shares': [50, 30, 100, 10],
        'Price': [150.0, 2800.5, 299.0, 3500.75],
        'Sector': ['Tech', 'Tech', 'Tech', 'Retail']}
df = pd.DataFrame(data)
df['Total_Value'] = df['Shares'] * df['Price']
print(df)
```

2. **Advanced Selection Solution**

```
loans = {'Loan_Amount': [8000, 15000, 20000, 5000],
         'Interest_Rate': [0.06, 0.045, 0.03, 0.07]}
df = pd.DataFrame(loans)
filtered = df[(df['Loan_Amount'] > 10000) &
(df['Interest_Rate'] < 0.05)]
print(filtered)
```

3. **Missing Data Handling Solution**

```
data = {'Price': [100, None, 250, None], 'Stock': [10, 20,
None, 5]}
df = pd.DataFrame(data)
df['Price'].fillna(df['Price'].mean(), inplace=True)
df.dropna(subset=['Stock'], inplace=True)
print(df)
```

4. **GroupBy and Aggregation Solution**

```
sales = {'Region': ['North', 'South', 'North', 'South'],
         'Salesperson': ['John', 'Anna', 'John', 'Anna'],
         'Product': ['A', 'B', 'C', 'A'],
         'Revenue': [500, 700, 800, 600]}
df = pd.DataFrame(sales)
region_revenue = df.groupby('Region')['Revenue'].sum()
salesperson_revenue = df.groupby('Salesperson')
['Revenue'].sum()
print(region_revenue)
print(salesperson_revenue)
```

5. **Custom Functions with Apply Solution**

```
data = {'Customer_ID': ['C001', 'C002', 'C003'],
        'Transaction_Amount': [500, 2000, 1500],
        'Age': [22, 30, 45]}
```

```
df = pd.DataFrame(data)
df['Eligibility'] = df.apply(lambda x: 'Eligible' if
x['Transaction_Amount'] > 1000 and x['Age'] > 25 else 'Not
Eligible', axis=1)
print(df)
```

6. **Concatenation Solution**

```
df1 = pd.DataFrame({'Department': ['Sales', 'HR'],
'Revenue': [1000, 500]})
df2 = pd.DataFrame({'Department': ['IT', 'Finance'],
'Revenue': [1200, 800]})
combined = pd.concat([df1, df2], ignore_index=True)
print(combined)
```

7. **Merging DataFrames Solution**

```
customers = pd.DataFrame({'Customer_ID': ['C001', 'C002',
'C003'],
                          'Name': ['John', 'Anna', 'Paul'],
                          'Location': ['NY', 'LA', 'SF']})
purchases = pd.DataFrame({'Customer_ID': ['C001', 'C003'],
                          'Purchase_Amount': [500, 700]})
merged = pd.merge(customers, purchases, on='Customer_ID',
how='left')
print(merged)
```

8. **DateTime Manipulation Solution**

```
df = pd.DataFrame({'Date': pd.to_datetime(['2023-01-01',
'2023-01-15', '2023-02-01']),
                   'Sales': [500, 700, 900]})
df['Month'] = df['Date'].dt.month
monthly_sales = df.groupby('Month')['Sales'].sum()
print(monthly_sales)
```

9. **String Operations with Series Solution**

```
feedback = pd.Series(['The service was excellent!', 'Poor
response', 'Excellent work'])
contains_excellent = feedback.str.contains('excellent',
case=False).sum()
feedback = feedback.str.replace('poor', 'unsatisfactory',
case=False)
print(contains_excellent)
print(feedback)
```

10. **Advanced Sorting Solution**

```
data = {'Product': ['A', 'B', 'C'], 'Price': [10, 50, 30],
'Stock': [100, 50, 200]}
df = pd.DataFrame(data)
sorted_df = df.sort_values(by=['Stock', 'Price'], ascending=
[False, True])
```

```
most_expensive_high_stock = sorted_df.iloc[0]
print(most_expensive_high_stock)
```