In [1]:
```
# Exercise 1, part 1: DataFrame from CSV

import pandas as pd

# Replace 'your_file_path' with the actual path to your CSV file
file_path = 'UIC2016Basketball.csv'

# Load the CSV file into a DataFrame
df2016 = pd.read_csv(file_path)

# Now, df2016 contains the data from the CSV file

display(df2016)
```

	Date	Opp	UIC Score	Opp Score	UIC FG	Opp FG	UIC 3P	Opp 3P	UIC TRB	Opp TRB	UIC AST	Opp AST
0	11/13/2015	San Francisco	75	78	28.0	24	8.0	8.0	36	42.0	17.0	15
1	11/17/2015	Western Illinois	57	84	15.0	30	3.0	5.0	40	42.0	6.0	16
2	11/24/2015	Roosevelt	96	58	35.0	22	1.0	5.0	52	23.0	NaN	15
3	11/28/2015	Drake	62	83	23.0	31	4.0	10.0	30	34.0	5.0	18
4	12/2/2015	DePaul	55	82	20.0	29	2.0	NaN	31	36.0	12.0	19
5	12/5/2015	UCF	58	88	22.0	26	4.0	7.0	43	32.0	10.0	14
6	12/12/2015	Illinois	79	83	NaN	25	9.0	9.0	38	25.0	18.0	18
7	12/16/2015	Illinois State	60	72	17.0	25	7.0	2.0	31	43.0	15.0	14
8	12/19/2015	Loyola (IL)	47	64	18.0	20	5.0	4.0	32	29.0	7.0	9
9	12/22/2015	Purdue Calumet	91	72	30.0	26	10.0	8.0	36	NaN	21.0	18
10	12/29/2015	Northern Illinois	65	70	20.0	22	4.0	8.0	30	33.0	18.0	19
11	1/2/2016	Valparaiso	47	75	18.0	25	2.0	8.0	28	39.0	11.0	12
12	1/8/2016	Detroit	69	87	21.0	26	2.0	4.0	35	32.0	6.0	10
13	1/10/2016	Oakland	61	86	21.0	25	NaN	4.0	32	30.0	10.0	11
14	1/14/2016	Green Bay	76	78	29.0	29	5.0	4.0	33	40.0	20.0	19
15	1/16/2016	Milwaukee	62	87	21.0	31	4.0	8.0	29	32.0	11.0	24
16	1/18/2016	Cleveland State	53	70	20.0	23	2.0	9.0	36	37.0	6.0	15
17	1/22/2016	Northern Kentucky	69	82	23.0	25	1.0	9.0	36	34.0	3.0	14
18	1/24/2016	Wright State	66	80	22.0	29	2.0	9.0	36	25.0	NaN	14
19	1/28/2016	Youngstown State	78	82	NaN	29	5.0	14.0	49	37.0	16.0	21
20	1/30/2016	Cleveland State	72	70	21.0	26	4.0	4.0	35	34.0	13.0	16
21	2/6/2016	Valparaiso	55	73	19.0	28	2.0	7.0	28	40.0	8.0	20
22	2/11/2016	Wright State	64	59	22.0	21	5.0	6.0	40	30.0	13.0	14
23	2/13/2016	Northern Kentucky	79	77	24.0	27	7.0	9.0	45	30.0	13.0	16
24	2/16/2016	Youngstown State	91	92	38.0	35	6.0	5.0	60	45.0	12.0	11
25	2/19/2016	Detroit	72	83	24.0	27	4.0	7.0	34	36.0	9.0	19
26	2/21/2016	Oakland	63	74	24.0	28	3.0	7.0	42	44.0	12.0	16
27	2/26/2016	Green Bay	69	85	22.0	30	6.0	9.0	31	44.0	7.0	17

	Date	Opp	UIC Score	Opp Score	UIC FG	Opp FG	UIC 3P	Opp 3P	UIC TRB	Opp TRB	UIC AST	Opp AST
28	2/28/2016	Milwaukee	85	98	30.0	30	9.0	15.0	36	39.0	13.0	25
29	3/5/2016	Wright State	43	74	12.0	29	2.0	14.0	39	39.0	5.0	20

In [2]:

```
# Exercise 1, part 2: Column Names

# Assign column headers to the DataFrame
df2016.columns = [
    'Date',
    'Opponent',
    'UIC Score',
    'Opp Score',
    'UIC Field Goal Percentage',
    'Opp Field Goal Percentage',
    'UIC 3 point Field Goal Percentage',
    'Opp 3 point Field Goal Percentage',
    'UIC Rebound',
    'Opp Rebound',
    'UIC Assists',
    'Opp Assists'
]

df2016_with_columns = df2016

# Now, df2016 has the specified column names

display(df2016_with_columns)
```

	Date	Opponent	UIC Score	Opp Score	UIC Field Goal Percentage	Opp Field Goal Percentage	UIC 3 point Field Goal Percentage	Opp 3 point Field Goal Percentage	UIC Rebound
0	11/13/2015	San Francisco	75	78	28.0	24	8.0	8.0	36
1	11/17/2015	Western Illinois	57	84	15.0	30	3.0	5.0	40
2	11/24/2015	Roosevelt	96	58	35.0	22	1.0	5.0	52
3	11/28/2015	Drake	62	83	23.0	31	4.0	10.0	30
4	12/2/2015	DePaul	55	82	20.0	29	2.0	NaN	31
5	12/5/2015	UCF	58	88	22.0	26	4.0	7.0	43
6	12/12/2015	Illinois	79	83	NaN	25	9.0	9.0	38
7	12/16/2015	Illinois State	60	72	17.0	25	7.0	2.0	31
8	12/19/2015	Loyola (IL)	47	64	18.0	20	5.0	4.0	32
9	12/22/2015	Purdue Calumet	91	72	30.0	26	10.0	8.0	36
10	12/29/2015	Northern Illinois	65	70	20.0	22	4.0	8.0	30
11	1/2/2016	Valparaiso	47	75	18.0	25	2.0	8.0	28
12	1/8/2016	Detroit	69	87	21.0	26	2.0	4.0	35
13	1/10/2016	Oakland	61	86	21.0	25	NaN	4.0	32
14	1/14/2016	Green Bay	76	78	29.0	29	5.0	4.0	33
15	1/16/2016	Milwaukee	62	87	21.0	31	4.0	8.0	29
16	1/18/2016	Cleveland State	53	70	20.0	23	2.0	9.0	36
17	1/22/2016	Northern Kentucky	69	82	23.0	25	1.0	9.0	36
18	1/24/2016	Wright State	66	80	22.0	29	2.0	9.0	36
19	1/28/2016	Youngstown State	78	82	NaN	29	5.0	14.0	49
20	1/30/2016	Cleveland State	72	70	21.0	26	4.0	4.0	35
21	2/6/2016	Valparaiso	55	73	19.0	28	2.0	7.0	28
22	2/11/2016	Wright State	64	59	22.0	21	5.0	6.0	40
23	2/13/2016	Northern Kentucky	79	77	24.0	27	7.0	9.0	45
24	2/16/2016	Youngstown State	91	92	38.0	35	6.0	5.0	60

	Date	Opponent	UIC Score	Opp Score	UIC Field Goal Percentage	Opp Field Goal Percentage	UIC 3 point Field Goal Percentage	Opp 3 point Field Goal Percentage	UIC Rebound
25	2/19/2016	Detroit	72	83	24.0	27	4.0	7.0	34
26	2/21/2016	Oakland	63	74	24.0	28	3.0	7.0	42
27	2/26/2016	Green Bay	69	85	22.0	30	6.0	9.0	31
28	2/28/2016	Milwaukee	85	98	30.0	30	9.0	15.0	36
29	3/5/2016	Wright	43	74	12.0	29	2.0	14.0	39

In [3]:
```python
# Exercise 1, part 3: Missing data

# Fill missing data cells with a hyphen symbol '-'
df2016_with_columns = df2016_with_columns.fillna('-')

display(df2016_with_columns)
```

	Date	Opponent	UIC Score	Opp Score	UIC Field Goal Percentage	Opp Field Goal Percentage	UIC 3 point Field Goal Percentage	Opp 3 point Field Goal Percentage	UIC Rebound
0	11/13/2015	San Francisco	75	78	28.0	24	8.0	8.0	36
1	11/17/2015	Western Illinois	57	84	15.0	30	3.0	5.0	40
2	11/24/2015	Roosevelt	96	58	35.0	22	1.0	5.0	52
3	11/28/2015	Drake	62	83	23.0	31	4.0	10.0	30
4	12/2/2015	DePaul	55	82	20.0	29	2.0	-	31
5	12/5/2015	UCF	58	88	22.0	26	4.0	7.0	43
6	12/12/2015	Illinois	79	83	-	25	9.0	9.0	38
7	12/16/2015	Illinois State	60	72	17.0	25	7.0	2.0	31
8	12/19/2015	Loyola (IL)	47	64	18.0	20	5.0	4.0	32
9	12/22/2015	Purdue Calumet	91	72	30.0	26	10.0	8.0	36
10	12/29/2015	Northern Illinois	65	70	20.0	22	4.0	8.0	30
11	1/2/2016	Valparaiso	47	75	18.0	25	2.0	8.0	28
12	1/8/2016	Detroit	69	87	21.0	26	2.0	4.0	35
13	1/10/2016	Oakland	61	86	21.0	25	-	4.0	32
14	1/14/2016	Green Bay	76	78	29.0	29	5.0	4.0	33
15	1/16/2016	Milwaukee	62	87	21.0	31	4.0	8.0	29
16	1/18/2016	Cleveland State	53	70	20.0	23	2.0	9.0	36
17	1/22/2016	Northern Kentucky	69	82	23.0	25	1.0	9.0	36
18	1/24/2016	Wright State	66	80	22.0	29	2.0	9.0	36
19	1/28/2016	Youngstown State	78	82	-	29	5.0	14.0	49
20	1/30/2016	Cleveland State	72	70	21.0	26	4.0	4.0	35
21	2/6/2016	Valparaiso	55	73	19.0	28	2.0	7.0	28
22	2/11/2016	Wright State	64	59	22.0	21	5.0	6.0	40
23	2/13/2016	Northern Kentucky	79	77	24.0	27	7.0	9.0	45
24	2/16/2016	Youngstown State	91	92	38.0	35	6.0	5.0	60

	Date	Opponent	UIC Score	Opp Score	UIC Field Goal Percentage	Opp Field Goal Percentage	UIC 3 point Field Goal Percentage	Opp 3 point Field Goal Percentage	UIC Rebound
25	2/19/2016	Detroit	72	83	24.0	27	4.0	7.0	34
26	2/21/2016	Oakland	63	74	24.0	28	3.0	7.0	42
27	2/26/2016	Green Bay	69	85	22.0	30	6.0	9.0	31
28	2/28/2016	Milwaukee	85	98	30.0	30	9.0	15.0	36
29	3/5/2016	Wright	43	74	12.0	29	2.0	14.0	39

In [4]:
```python
# Exercise 1, part 4: Data Types

# Check the data types of columns in the DataFrame
data_types = df2016.dtypes
display(data_types)
```

```
Date                                  object
Opponent                              object
UIC Score                              int64
Opp Score                              int64
UIC Field Goal Percentage            float64
Opp Field Goal Percentage              int64
UIC 3 point Field Goal Percentage    float64
Opp 3 point Field Goal Percentage    float64
UIC Rebound                            int64
Opp Rebound                          float64
UIC Assists                          float64
Opp Assists                            int64
dtype: object
```

In this example, the data types used are:

'Date' and 'Opponent' are of type 'object' (typically representing strings or mixed data types).

'UIC Score', 'Opp Score', 'Opp Field Goal Percentage', 'UIC Rebound', and 'Opp Assists' are of type 'int64' (representing integers).

'UIC Field Goal Percentage', 'UIC 3 point Field Goal Percentage', 'Opp 3 point Field Goal Percentage', 'Opp Rebound', and 'UIC Assists' are of type 'float64' (representing floating-point numbers).

In [5]:
```python
# Exercise 1, part 5: Count

# Count the number of unique school opponents played in 2016 using the 'Opp' column
opponents_count = df2016['Opponent'].nunique()
print("Number of school opponents played in 2016:", opponents_count)
```

```
Number of school opponents played in 2016: 20
```

The unique school opponents are as follows:

1. San Francisco

2. Western Illinois
3. Roosevelt
4. Drake
5. DePaul
6. UCF
7. Illnois
8. Illinois State
9. Loyola (IL)
10. Purdue Calumet
11. Northern Illinois
12. Valparaiso
13. Detroit
14. Oakland
15. Green Bay
16. Milwaukee
17. Cleveland State
18. Northern Kentucky
19. Wright State
20. Youngstown State

In [6]:
```
# Exercise 1, part 6: Filter

# Filter and list all games where UIC scored more than 65 points
high_scoring_games = df2016_with_columns[df2016_with_columns['UIC Score'] > 65]

high_scoring_games = high_scoring_games.fillna('-')

# Display the filtered DataFrame
display(high_scoring_games)
```

	Date	Opponent	UIC Score	Opp Score	UIC Field Goal Percentage	Opp Field Goal Percentage	UIC 3 point Field Goal Percentage	Opp 3 point Field Goal Percentage	UIC Rebound
0	11/13/2015	San Francisco	75	78	28.0	24	8.0	8.0	36
2	11/24/2015	Roosevelt	96	58	35.0	22	1.0	5.0	52
6	12/12/2015	Illinois	79	83	-	25	9.0	9.0	38
9	12/22/2015	Purdue Calumet	91	72	30.0	26	10.0	8.0	36
12	1/8/2016	Detroit	69	87	21.0	26	2.0	4.0	35
14	1/14/2016	Green Bay	76	78	29.0	29	5.0	4.0	33
17	1/22/2016	Northern Kentucky	69	82	23.0	25	1.0	9.0	36
18	1/24/2016	Wright State	66	80	22.0	29	2.0	9.0	36
19	1/28/2016	Youngstown State	78	82	-	29	5.0	14.0	49
20	1/30/2016	Cleveland State	72	70	21.0	26	4.0	4.0	35
23	2/13/2016	Northern Kentucky	79	77	24.0	27	7.0	9.0	45
24	2/16/2016	Youngstown State	91	92	38.0	35	6.0	5.0	60
25	2/19/2016	Detroit	72	83	24.0	27	4.0	7.0	34
27	2/26/2016	Green Bay	69	85	22.0	30	6.0	9.0	31
28	2/28/2016	Milwaukee	85	98	30.0	30	9.0	15.0	36

List of opponents and the score, where UIC scored more than 65 points:

Opponent Score

1. San Francisco 75
2. Roosevelt 96
3. Illinois 79
4. Purdue Calumet 91
5. Detroit 69
6. Green Bay 76
7. Northern Kentucky 69
8. Wright State 66
9. Youngstown State 78
10. Cleveland State 72

11. Northern Kentucky 79
12. Youngstown State 91
13. Detroit 72
14. Green Bay 69
15. Milwaukee 85

In [7]:
```
# Exercise 1, part 7: Win-Los-Tie

# Calculate the number of wins, losses, and ties
wins = len(df2016[df2016['UIC Score'] > df2016['Opp Score']])
losses = len(df2016[df2016['UIC Score'] < df2016['Opp Score']])
ties = len(df2016[df2016['UIC Score'] == df2016['Opp Score']])

# Display the results
print("Number of Wins:", wins)
print("Number of Losses:", losses)
print("Number of Ties:", ties)
```

```
Number of Wins: 5
Number of Losses: 25
Number of Ties: 0
```

Wins against:

1. Roosevelt
2. Purdue Calumet
3. Cleveland State
4. Wright State
5. Northern Kentucky

Loss against:

1. San Francisco
2. Western Illinois
3. Drake
4. DePaul
5. UCF
6. Illinois
7. Illinois State
8. Loyola (IL)
9. Northern Illinois
10. Valparaiso
11. Detroit
12. Oakland
13. Green Bay
14. Milwaukee
15. Cleveland State
16. Northern Kentucky
17. Wright State

18. Youngstown State
19. Valparaiso
20. Youngstown State
21. Detroit
22. Oakland
23. Green Bay
24. Milwaukee
25. Wright State

In [8]:

```
# Exercise 2, part 1: Load data and add column names

import pandas as pd

# Load the data from the text file with the specified separator (comma)
data = pd.read_csv('customer_savings.txt', sep=',')

# Set the column names for the DataFrame
data.columns = ["Customer ID", "Customer Name", "Customer Surname", "Gender", "Age", '

# Now, 'data' is a DataFrame with the loaded data and column names

display(data)
```

	Customer ID	Customer Name	Customer Surname	Gender	Age	Region	Job Classification	Date joined	Balance
0	100000003	Liam	Brown	Male	46	England	White Collar	07.Jan.15	101536.83
1	100000005	Deirdre	Pullman	Female	38	England	Blue Collar	09.Jan.15	35639.79
2	100000007	Dorothy	Thomson	Female	34	England	Blue Collar	11.Jan.15	42879.84
3	100000010	Dominic	Parr	Male	42	England	White Collar	12.Jan.15	10912.45
4	100000011	Dominic	Lewis	Male	40	England	White Collar	12.Jan.15	39667.83
...	...	...	...	...	...	...	...	...	...
4008	400003443	Abigail	MacLeod	Female	21	Northern Ireland	Blue Collar	29.Nov.15	51615.61
4009	400003472	Dorothy	Bell	Female	34	Northern Ireland	Blue Collar	30.Nov.15	15263.47
4010	400003743	Keith	Davies	Male	19	Northern Ireland	Other	15.Dec.15	50562.98
4011	400003847	Donna	Lambert	Female	34	Northern Ireland	White Collar	20.Dec.15	87664.15
4012	400003848	Carolyn	Dowd	Female	52	Northern Ireland	Other	20.Dec.15	118676.95

4013 rows × 9 columns

In [9]:

```
# Exercise 2, part 2: What’s the average balance for male and female? Provide both Pyt
```

```
# Calculate the average balance for male and female separately
average_balance_male = data[data['Gender'] == 'Male']['Balance'].mean()
average_balance_female = data[data['Gender'] == 'Female']['Balance'].mean()

print("Average balance for male customers:", average_balance_male)
print("Average balance for female customers:", average_balance_female)
```

```
Average balance for male customers: 39983.90903419594
Average balance for female customers: 39471.89511627907
```

In [10]:
```
# Exercise 2, part 3: What’s the average balance for while collar and blue collar in E

# Define a lambda function to calculate average balance for a specific group
average_balance = lambda group: data[(data['Job Classification'] == group) & (data['Re

# Calculate the average balance for "White Collar" and "Blue Collar" using the lambda
average_balance_white_collar = average_balance('White Collar')
average_balance_blue_collar = average_balance('Blue Collar')

print("Average balance for White Collar customers in England:", average_balance_white_
print("Average balance for Blue Collar customers in England:", average_balance_blue_cc
```

```
Average balance for White Collar customers in England: 39106.53648
Average balance for Blue Collar customers in England: 38567.84156976744
```

In [11]:
```
# Exercise 3:

import pandas as pd

# Read the data from customer-status.csv and sales.csv
try:
    customer_status_df = pd.read_csv('customer-status.csv')
    sales_df = pd.read_csv('sales.csv')
except FileNotFoundError:
    print("Error: One or both of the CSV files not found.")
    exit(1)

# Part 1:

# Inner join on column "Account Number"
inner_join_df = customer_status_df.merge(sales_df, on='account number')
inner_join_rows = len(inner_join_df)
print("\033[1mTotal number of rows in Inner Join: ", inner_join_rows, "\033[0m")

# Part 2:

# Full outer join on column "Account Number"
full_outer_join_df = customer_status_df.merge(sales_df, on='account number', how='oute
full_outer_join_rows = len(full_outer_join_df)
print("\033[1mTotal number of rows in Full Outer Join: ", full_outer_join_rows, "\033[

# Part 3:

# Left join on column "Account Number", using customer-status.csv as the base
left_join_customer_df = customer_status_df.merge(sales_df, on='account number', how='l
left_join_customer_rows = len(left_join_customer_df)
print("\033[1mTotal number of rows in Left Join with customer-status.csv as the base:

# Part 4:

# Left join on column "Account Number", using sales.csv as the base
```

```
left_join_sales_df = sales_df.merge(customer_status_df, on='account number', how='left
left_join_sales_rows = len(left_join_sales_df)
print("\033[1mTotal number of rows in Left Join with sales.csv as the base: ", left_jo

# Now, display the results

# Part 1: Inner Join Result
print()
print("\033[1mResults: \033[0m")
print()
print("\033[1mInner Join Result:\033[0m")
print()
print(inner_join_df)

# Part 2: Full Outer Join Result
print()
print("\033[1mFull Outer Join Result:\033[0m")
print()
print(full_outer_join_df)

# Part 3: Left Join with customer-status.csv as the base Result
print()
print("\033[1mLeft Join with customer-status.csv as the base Result:\033[0m")
print()
print(left_join_customer_df)

# Part 4: Left Join with sales.csv as the base Result
print()
print("\033[1mLeft Join with sales.csv as the base Result:\033[0m")
print()
print(left_join_sales_df)
```

Total number of rows in Inner Join: 118
Total number of rows in Full Outer Join: 142
Total number of rows in Left Join with customer-status.csv as the base: 118
Total number of rows in Left Join with sales.csv as the base: 142

Results:

Inner Join Result:

```
     account number                           name_x  status  \
0            740150                       Barton LLC    gold   
1            740150                       Barton LLC    gold   
2            740150                       Barton LLC    gold   
3            714466                  Trantow-Barrows  silver   
4            714466                  Trantow-Barrows  silver   
..              ...                              ...     ...   
113          257198  Cronin, Oberbrunner and Spencer    gold   
114          257198  Cronin, Oberbrunner and Spencer    gold   
115          257198  Cronin, Oberbrunner and Spencer    gold   
116          257198  Cronin, Oberbrunner and Spencer    gold   
117          257198  Cronin, Oberbrunner and Spencer    gold   

                              name_y       sku  quantity  unit price  \
0                         Barton LLC  S1-82801        29       60.81   
1                         Barton LLC  B1-20000        20       73.93   
2                         Barton LLC  S2-83881        12       22.62   
3                    Trantow-Barrows  B1-33087        43       32.77   
4                    Trantow-Barrows  S2-16558        20       78.23   
..                               ...       ...       ...         ...   
113  Cronin, Oberbrunner and Spencer  S1-93683        34       79.57   
114  Cronin, Oberbrunner and Spencer  S1-82801        22       12.01   
115  Cronin, Oberbrunner and Spencer  S2-00301        19       41.81   
116  Cronin, Oberbrunner and Spencer  S1-30248        11       58.82   
117  Cronin, Oberbrunner and Spencer  S2-77896         4       23.04   

     ext price                 date  
0      1763.49  2014-03-07 10:24:54  
1      1478.60  2014-03-15 18:21:23  
2       271.44  2014-03-17 02:39:33  
3      1409.11  2014-03-14 12:47:48  
4      1564.60  2014-03-17 09:03:19  
..         ...                  ...  
113    2705.38  2014-03-12 08:58:47  
114     264.22  2014-03-17 10:05:43  
115     794.39  2014-03-27 03:52:01  
116     647.02  2014-03-27 20:40:13  
117      92.16  2014-03-30 18:12:17  

[118 rows x 9 columns]
```

Full Outer Join Result:

```
     account number           name_x  status                           name_y  \
0            740150       Barton LLC    gold                       Barton LLC   
1            740150       Barton LLC    gold                       Barton LLC   
2            740150       Barton LLC    gold                       Barton LLC   
3            714466  Trantow-Barrows  silver                  Trantow-Barrows   
4            714466  Trantow-Barrows  silver                  Trantow-Barrows   
..              ...              ...     ...                              ...   
137          604255              NaN     NaN  Halvorson, Crona and Champlin   
```

```
138         604255            NaN     NaN  Halvorson, Crona and Champlin
139         604255            NaN     NaN  Halvorson, Crona and Champlin
140         604255            NaN     NaN  Halvorson, Crona and Champlin
141         604255            NaN     NaN  Halvorson, Crona and Champlin

          sku  quantity  unit price  ext price                 date
0    S1-82801        29       60.81    1763.49  2014-03-07 10:24:54
1    B1-20000        20       73.93    1478.60  2014-03-15 18:21:23
2    S2-83881        12       22.62     271.44  2014-03-17 02:39:33
3    B1-33087        43       32.77    1409.11  2014-03-14 12:47:48
4    S2-16558        20       78.23    1564.60  2014-03-17 09:03:19
..        ...       ...         ...        ...                  ...
137  B1-33087        28       61.35    1717.80  2014-03-21 05:41:09
138  S2-00301        35       24.33     851.55  2014-03-21 20:12:32
139  S2-77896        23       64.91    1492.93  2014-03-24 19:21:21
140  B1-53102        32       86.77    2776.64  2014-03-30 01:14:16
141  S2-78676        18       57.02    1026.36  2014-03-31 06:53:52

[142 rows x 9 columns]
```

Left Join with customer-status.csv as the base Result:

```
     account number                         name_x  status  \
0            740150                     Barton LLC    gold
1            740150                     Barton LLC    gold
2            740150                     Barton LLC    gold
3            714466                Trantow-Barrows  silver
4            714466                Trantow-Barrows  silver
..              ...                            ...     ...
113          257198  Cronin, Oberbrunner and Spencer    gold
114          257198  Cronin, Oberbrunner and Spencer    gold
115          257198  Cronin, Oberbrunner and Spencer    gold
116          257198  Cronin, Oberbrunner and Spencer    gold
117          257198  Cronin, Oberbrunner and Spencer    gold

                              name_y       sku  quantity  unit price  \
0                         Barton LLC  S1-82801        29       60.81
1                         Barton LLC  B1-20000        20       73.93
2                         Barton LLC  S2-83881        12       22.62
3                    Trantow-Barrows  B1-33087        43       32.77
4                    Trantow-Barrows  S2-16558        20       78.23
..                               ...       ...       ...         ...
113  Cronin, Oberbrunner and Spencer  S1-93683        34       79.57
114  Cronin, Oberbrunner and Spencer  S1-82801        22       12.01
115  Cronin, Oberbrunner and Spencer  S2-00301        19       41.81
116  Cronin, Oberbrunner and Spencer  S1-30248        11       58.82
117  Cronin, Oberbrunner and Spencer  S2-77896         4       23.04

     ext price                 date
0      1763.49  2014-03-07 10:24:54
1      1478.60  2014-03-15 18:21:23
2       271.44  2014-03-17 02:39:33
3      1409.11  2014-03-14 12:47:48
4      1564.60  2014-03-17 09:03:19
..         ...                  ...
113    2705.38  2014-03-12 08:58:47
114     264.22  2014-03-17 10:05:43
115     794.39  2014-03-27 03:52:01
116     647.02  2014-03-27 20:40:13
117      92.16  2014-03-30 18:12:17
```

```
[118 rows x 9 columns]

Left Join with sales.csv as the base Result:

     account number                          name_x       sku  quantity  \
0            163416                     Purdy-Kunde  S1-30248        19   
1            527099                 Sanford and Sons  S2-82423         3   
2            527099                 Sanford and Sons  B1-50809         8   
3            737550  Fritsch, Russel and Anderson  B1-50809        20   
4            688981                     Keeling LLC  B1-86481        -1   
..              ...                             ...       ...       ...   
137          737550  Fritsch, Russel and Anderson  B1-65551        12   
138          642753                     Pollich LLC  S1-93683        21   
139          412290                   Jerde-Hilpert  B1-20000        30   
140          307599   Kassulke, Ondricka and Metz  S2-16558        46   
141          672390                 Kuhn-Gusikowski  B1-04202        19   

     unit price  ext price                 date                       name_y  \
0         65.03    1235.57  2014-03-01 16:07:40                          NaN   
1         76.21     228.63  2014-03-01 17:18:01             Sanford and Sons   
2         70.78     566.24  2014-03-01 18:53:09             Sanford and Sons   
3         50.11    1002.20  2014-03-01 23:47:17                          NaN   
4         97.16     -97.16  2014-03-02 01:46:44                  Keeling LLC   
..          ...        ...                  ...                          ...   
137       56.24     674.88  2014-03-31 08:43:24                          NaN   
138       92.57    1943.97  2014-03-31 11:37:34                  Pollich LLC   
139       22.38     671.40  2014-03-31 21:41:31                Jerde-Hilpert   
140       56.04    2577.84  2014-03-31 22:11:22  Kassulke, Ondricka and Metz   
141       27.86     529.34  2014-03-31 23:13:14              Kuhn-Gusikowski   

     status  
0       NaN  
1    bronze  
2    bronze  
3       NaN  
4    silver  
..      ...  
137     NaN  
138  bronze  
139  bronze  
140  bronze  
141  silver  

[142 rows x 9 columns]
```

In [ ]: