```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df = pd.read_csv("/student-data.csv")
display(df)
```

	participant_id	name	dob	is_student	target	Q1	Q2	Q3	Q4
0	1de9ea66-70d3-4a1f-8735-df5ef7697fb9	Thomas Crosby	1996-07-16	False	809.97	1.0	5.0	0.0	0.0
1	9da618fd-7bf7-4a4d-9f8f-5ffba5f80a0a	Brandon Reeves	1957-11-30	False	866.41	1.0	8.0	1.0	1.0
2	d30aad4b-4503-4e22-8bc4-621b94398520	Maria Castillo	1981-02-03	False	776.92	3.0	8.0	0.0	1.0
3	790b2f7c-b5c3-4ec1-a4ce-01e15560eaba	Carol Jones	1990-09-17	True	929.72	3.0	0.0	1.0	2.0
4	6a8f6926-0a22-4ba8-b442-a1244e2e3761	Robert Thompson	1967-06-09	True	923.66	NaN	0.0	6.0	NaN
...	...	...	...	...	...	...	...	...	...
99	af2fb5fe-b506-4098-bf79-c84e5a2e595e	Barbara Ford	1975-01-23	False	937.97	8.0	4.0	4.0	4.0
100	d30aad4b-4503-4e22-8bc4-621b94398520	Maria Castillo	1981-02-03	False	776.92	3.0	8.0	0.0	1.0
101	6016d60e-b39f-458d-9d1d-fe49869adcef	Edward Hood	1984-01-10	False	975.77	3.0	NaN	2.0	4.0
102	77ff71ba-07d3-470a-a402-ffccf29d469b	Jeffrey Smith	1996-09-29	False	996.08	6.0	2.0	7.0	9.0
103	84754a9e-bd1d-4bec-b25b-69b12a2fbff3	Hayley Hoffman	1958-06-23	False	825.04	3.0	0.0	5.0	2.0

104 rows × 9 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
#Wide to long

#The method will melt all columns NOT mentioned in id_vars together into two columns
#The additional parameters have the following effects:
#value_vars defines which columns to melt together
#value_name provides a custom column name for the values column instead of the default column name value
#var_name provides a custom column name for the column collecting the column header names
df = df.drop_duplicates()
long = pd.melt(df, id_vars=['participant_id', 'name'], value_vars=['Q1','Q2','Q3','Q4'], var_name='quarter', value_name='clicks')
display(long)
```

	participant_id	name	quarter	clicks
0	1de9ea66-70d3-4a1f-8735-df5ef7697fb9	Thomas Crosby	Q1	1.0
1	9da618fd-7bf7-4a4d-9f8f-5ffba5f80a0a	Brandon Reeves	Q1	1.0
2	d30aad4b-4503-4e22-8bc4-621b94398520	Maria Castillo	Q1	3.0
3	790b2f7c-b5c3-4ec1-a4ce-01e15560eaba	Carol Jones	Q1	3.0
4	6a8f6926-0a22-4ba8-b442-a1244e2e3761	Robert Thompson	Q1	NaN
...	...	...	...	...
395	802d0313-9654-4d90-a5c5-b7bd1dcbd385	Linda May	Q4	6.0
396	f5f149db-fcda-47c7-b0d4-edbfb072c7c2	Matthew Walker	Q4	5.0
397	58dcca0f-d500-4813-aa0a-759b4b85435e	Dana Weber	Q4	9.0
398	07b634e2-1999-4128-a3dc-a56ce18e73b8	Darrell Johnson	Q4	5.0
399	af2fb5fe-b506-4098-bf79-c84e5a2e595e	Barbara Ford	Q4	4.0

400 rows × 4 columns

Next steps: Generate code with long | View recommended plots | New interactive sheet

```
#Long to wide
wide = pd.pivot(long, index=['participant_id', 'name'], columns='quarter', values='clicks')
display(wide)
```

participant_id	name	quarter	Q1	Q2	Q3	Q4
048f1b2c-a3b1-42d3-931d-34312c8e80fd	Craig Branch		8.0	4.0	0.0	7.0
04a5f6a0-cf9a-4702-a915-af29a7344d09	Erin Elliott		2.0	9.0	1.0	8.0
06f74681-6cde-4cd1-8a92-8093ef38bc0e	Matthew Evans		9.0	4.0	1.0	1.0
07b634e2-1999-4128-a3dc-a56ce18e73b8	Darrell Johnson		5.0	7.0	7.0	5.0
094d962f-5a5b-407f-82b0-de54e1b5c6ec	Cindy Shaffer		9.0	4.0	9.0	4.0
...	...	...	...	...	...	...
e66ad629-b622-4201-85c0-2167778fb826	George Elliott		4.0	9.0	9.0	1.0
f2483f63-db08-4d62-a337-cebd201a7d6a	Mathew Nixon		NaN	1.0	7.0	NaN
f45fcbeb-84b8-4b91-8079-42c9adf9ca6f	Corey Walker		7.0	0.0	6.0	6.0
f5c42d3e-ea4b-4216-9fd1-3150efb725cd	Mark Torres		8.0	4.0	6.0	9.0
f5f149db-fcda-47c7-b0d4-edbfb072c7c2	Matthew Walker		5.0	8.0	2.0	5.0

100 rows × 4 columns

Next steps: Generate code with wide | View recommended plots | New interactive sheet

```
# Applying aggregation across all the columns
df.aggregate(['sum', 'min'])
```

	participant_id	name	dob	is_student	target	Q1	Q2	Q3	Q4
sum	1de9ea66-70d3-4a1f-8735-df5ef7697fb99da618fd-7...	Thomas CrosbyBrandon ReevesMaria CastilloCarol...	1996-07-161957-11-301981-02-031990-09-171967-0...	18	85343.11	424.0	409.0	444.0	411.0

```
# We are going to find aggregation for these columns
df.aggregate({"Q1":['sum', 'min','max'],
              "Q2":['sum', 'min','max'],
              "Q3":['min', 'sum'],
              "Q4":['sum']})
```

	Q1	Q2	Q3	Q4
sum	424.0	409.0	444.0	411.0
min	0.0	0.0	0.0	NaN
max	9.0	9.0	NaN	NaN

```
# applying multiple aggregation functions to a single column

long = pd.melt(df, id_vars=['participant_id', 'name'], value_vars=['Q1','Q2','Q3','Q4'], var_name='quarter', value_name='clicks')



result = long.groupby('quarter')['clicks'].agg(['sum', 'mean', 'max', 'min'])
print(result)
```

```
            sum      mean  max  min
quarter
Q1        424.0  4.416667  9.0  0.0
Q2        409.0  4.494505  9.0  0.0
Q3        444.0  4.484848  9.0  0.0
Q4        411.0  4.892857  9.0  0.0
```

```
result = long.groupby('quarter')['clicks'].agg(['count'])
print(result)
```

```
          count
quarter
Q1           96
```

```
Q2          91
Q3          99
Q4          84
```

```
#Scaling transforms
#Normalization : Normalization of data transforms each value to a value between 0 and 1
scaling = df.copy()
min_target = np.min(scaling['target'])
max_target = np.max(scaling['target'])
scaling['norm_target'] = (scaling['target'] - min_target) / (max_target - min_target)
display(scaling)
```

	participant_id	name	dob	is_student	target	Q1	Q2	Q3	Q4	norm_target
0	1de9ea66-70d3-4a1f-8735-df5ef7697fb9	Thomas Crosby	1996-07-16	False	809.97	1.0	5.0	0.0	0.0	0.672549
1	9da618fd-7bf7-4a4d-9f8f-5ffba5f80a0a	Brandon Reeves	1957-11-30	False	866.41	1.0	8.0	1.0	1.0	0.771852
2	d30aad4b-4503-4e22-8bc4-621b94398520	Maria Castillo	1981-02-03	False	776.92	3.0	8.0	0.0	1.0	0.614399
3	790b2f7c-b5c3-4ec1-a4ce-01e15560eaba	Carol Jones	1990-09-17	True	929.72	3.0	0.0	1.0	2.0	0.883243
4	6a8f6926-0a22-4ba8-b442-a1244e2e3761	Robert Thompson	1967-06-09	True	923.66	NaN	0.0	6.0	NaN	0.872581
...	...	...	...	...	...	...	...	...	...	...
95	802d0313-9654-4d90-a5c5-b7bd1dcbd385	Linda May	1985-09-08	False	918.74	0.0	7.0	2.0	6.0	0.863924
96	f5f149db-fcda-47c7-b0d4-edbfb072c7c2	Matthew Walker	1983-11-22	False	680.97	5.0	8.0	2.0	5.0	0.445580
97	58dcca0f-d500-4813-aa0a-759b4b85435e	Dana Weber	1994-03-02	False	930.63	4.0	9.0	1.0	9.0	0.884844
98	07b634e2-1999-4128-a3dc-a56ce18e73b8	Darrell Johnson	1984-09-13	False	982.00	5.0	7.0	7.0	5.0	0.975227
99	af2fb5fe-b506-4098-bf79-c84e5a2e595e	Barbara Ford	1975-01-23	False	937.97	8.0	4.0	4.0	4.0	0.897758

100 rows × 10 columns

Next steps: Generate code with scaling | View recommended plots | New interactive sheet

```
#Standardizing: transforms each value such that the distribution has a mean of 0 and a standard deviation of 1.
scaling = df.copy()
mean_target = np.mean(scaling['target'])
sd_target = np.std(scaling['target'])
scaling['standardized_target'] = (scaling['target'] - mean_target) / (sd_target)

display(scaling)
```

	participant_id	name	dob	is_student	target	Q1	Q2	Q3	Q4	standardized_target
0	1de9ea66-70d3-4a1f-8735-df5ef7697fb9	Thomas Crosby	1996-07-16	False	809.97	1.0	5.0	0.0	0.0	-0.356197
1	9da618fd-7bf7-4a4d-9f8f-5ffba5f80a0a	Brandon Reeves	1957-11-30	False	866.41	1.0	8.0	1.0	1.0	0.106372
2	d30aad4b-4503-4e22-8bc4-621b94398520	Maria Castillo	1981-02-03	False	776.92	3.0	8.0	0.0	1.0	-0.627066
3	790b2f7c-b5c3-4ec1-a4ce-01e15560eaba	Carol Jones	1990-09-17	True	929.72	3.0	0.0	1.0	2.0	0.625245
4	6a8f6926-0a22-4ba8-b442-a1244e2e3761	Robert Thompson	1967-06-09	True	923.66	NaN	0.0	6.0	NaN	0.575579
...	...	...	...	...	...	...	...	...	...	...
95	802d0313-9654-4d90-a5c5-b7bd1dcbd385	Linda May	1985-09-08	False	918.74	0.0	7.0	2.0	6.0	0.535256
96	f5f149db-fcda-47c7-b0d4-edbfb072c7c2	Matthew Walker	1983-11-22	False	680.97	5.0	8.0	2.0	5.0	-1.413449
97	58dcca0f-d500-4813-aa0a-759b4b85435e	Dana Weber	1994-03-02	False	930.63	4.0	9.0	1.0	9.0	0.632703
98	07b634e2-1999-4128-a3dc-a56ce18e73b8	Darrell Johnson	1984-09-13	False	982.00	5.0	7.0	7.0	5.0	1.053720
99	af2fb5fe-b506-4098-bf79-c84e5a2e595e	Barbara Ford	1975-01-23	False	937.97	8.0	4.0	4.0	4.0	0.692860

100 rows × 10 columns

Next steps: Generate code with scaling | View recommended plots | New interactive sheet