

```
from google.colab import files
uploaded = files.upload()
```

Choose Files | Customer P...haviors.csv
Customer Purchasing Behaviors.csv(text/csv) - 7268 bytes, last modified: 4/2/2026 - 100% done
 Saving Customer Purchasing Behaviors.csv to Customer Purchasing Behaviors.csv

```
#1
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
#2
df = pd.read_csv("Customer Purchasing Behaviors.csv")
df
```

	user_id	age	annual_income	purchase_amount	loyalty_score	region	purchase_frequency	grid icon
0	1	25	45000	200	4.5	North	12	edit icon
1	2	34	55000	350	7.0	South	18	
2	3	45	65000	500	8.0	West	22	
3	4	22	30000	150	3.0	East	10	
4	5	29	47000	220	4.8	North	13	
...	
233	234	40	60000	450	7.2	West	20	
234	235	38	59000	430	6.9	North	20	
235	236	54	74000	630	9.4	South	27	
236	237	32	52000	360	5.8	West	18	
237	238	31	51000	340	5.6	North	17	

238 rows × 7 columns

Next steps: [Generate code with df](#) [New interactive sheet](#)

```
#3
print(df.shape)
```

(238, 7)

```
#4
print(df.columns)
```

```
Index(['user_id', 'age', 'annual_income', 'purchase_amount', 'loyalty_score',
       'region', 'purchase_frequency'],
      dtype='object')
```

```
#5
print(df.head(5))
```

	user_id	age	annual_income	purchase_amount	loyalty_score	region	\
0	1	25	45000	200	4.5	North	
1	2	34	55000	350	7.0	South	
2	3	45	65000	500	8.0	West	
3	4	22	30000	150	3.0	East	
4	5	29	47000	220	4.8	North	

	purchase_frequency
0	12
1	18
2	22
3	10
4	13

```
#6
print(df.dtypes)
```

user_id	int64
age	int64
annual_income	int64
purchase_amount	int64
loyalty_score	float64

```
region          object
purchase_frequency    int64
dtype: object
```

```
#7
print(df.isnull().sum())
```

```
user_id          0
age              0
annual_income    0
purchase_amount  0
loyalty_score    0
region           0
purchase_frequency 0
dtype: int64
```

```
#8
num_cols = df.select_dtypes(include=np.number).columns

for col in num_cols:
    df[col].fillna(df[col].mean(), inplace=True)
df
```

/tmp/ipython-input-2753514288.py:5: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through ch
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col]

```
df[col].fillna(df[col].mean(), inplace=True)
```

	user_id	age	annual_income	purchase_amount	loyalty_score	region	purchase_frequency	grid icon	edit icon
0	1	25	45000	200	4.5	North	12		
1	2	34	55000	350	7.0	South	18		
2	3	45	65000	500	8.0	West	22		
3	4	22	30000	150	3.0	East	10		
4	5	29	47000	220	4.8	North	13		
...		
233	234	40	60000	450	7.2	West	20		
234	235	38	59000	430	6.9	North	20		
235	236	54	74000	630	9.4	South	27		
236	237	32	52000	360	5.8	West	18		
237	238	31	51000	340	5.6	North	17		

238 rows × 7 columns

Next steps: [Generate code with df](#) [New interactive sheet](#)

```
#9
cat_cols = df.select_dtypes(include='object').columns

for col in cat_cols:
    df[col].fillna(df[col].mode()[0], inplace=True)
df
```

```
/tmp/ipython-input-3043855207.py:5: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through ch
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are
For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col]
```

```
df[col].fillna(df[col].mode()[0], inplace=True)
```

	user_id	age	annual_income	purchase_amount	loyalty_score	region	purchase_frequency	
0	1	25	45000	200	4.5	North	12	
1	2	34	55000	350	7.0	South	18	
2	3	45	65000	500	8.0	West	22	
3	4	22	30000	150	3.0	East	10	

```
#10
df.isnull().sum()
```

233	234	40	0	60000	450	7.2	West	20
234	235	20	0	59000	430	6.9	North	20
235	236	34	0	74000	630	9.4	South	27
236	237	22	0	52000	360	5.8	West	18
237	238	31	0	51000	340	5.6	North	17
238	239	31	0					
239	240	31	0					
240	241	31	0					
241	242	31	0					
242	243	31	0					
243	244	31	0					
244	245	31	0					
245	246	31	0					
246	247	31	0					
247	248	31	0					
248	249	31	0					
249	250	31	0					
250	251	31	0					
251	252	31	0					
252	253	31	0					
253	254	31	0					
254	255	31	0					
255	256	31	0					
256	257	31	0					
257	258	31	0					
258	259	31	0					
259	260	31	0					
260	261	31	0					
261	262	31	0					
262	263	31	0					
263	264	31	0					
264	265	31	0					
265	266	31	0					
266	267	31	0					
267	268	31	0					
268	269	31	0					
269	270	31	0					
270	271	31	0					
271	272	31	0					
272	273	31	0					
273	274	31	0					
274	275	31	0					
275	276	31	0					
276	277	31	0					
277	278	31	0					
278	279	31	0					
279	280	31	0					
280	281	31	0					
281	282	31	0					
282	283	31	0					
283	284	31	0					
284	285	31	0					
285	286	31	0					
286	287	31	0					
287	288	31	0					
288	289	31	0					
289	290	31	0					
290	291	31	0					
291	292	31	0					
292	293	31	0					
293	294	31	0					
294	295	31	0					
295	296	31	0					
296	297	31	0					
297	298	31	0					
298	299	31	0					
299	300	31	0					
300	301	31	0					
301	302	31	0					
302	303	31	0					
303	304	31	0					
304	305	31	0					
305	306	31	0					
306	307	31	0					
307	308	31	0					
308	309	31	0					
309	310	31	0					
310	311	31	0					
311	312	31	0					
312	313	31	0					
313	314	31	0					
314	315	31	0					
315	316	31	0					
316	317	31	0					
317	318	31	0					
318	319	31	0					
319	320	31	0					
320	321	31	0					
321	322	31	0					
322	323	31	0					
323	324	31	0					
324	325	31	0					
325	326	31	0					
326	327	31	0					
327	328	31	0					
328	329	31	0					
329	330	31	0					
330	331	31	0					
331	332	31	0					
332	333	31	0					
333	334	31	0					
334	335	31	0					
335	336	31	0					
336	337	31	0					
337	338	31	0					
338	339	31	0					
339	340	31	0					
340	341	31	0					
341	342	31	0					
342	343	31	0					
343	344	31	0					
344	345	31	0					
345	346	31	0					
346	347	31	0					
347	348	31	0					
348	349	31	0					
349	350	31	0					
350	351	31	0					
351	352	31	0					
352	353	31	0					
353	354	31	0					
354	355	31	0					
355	356	31	0					
356	357	31	0					
357	358	31	0					
358	359	31	0					
359	360	31	0					
360	361	31	0					
361	362	31	0					
362	363	31	0					
363	364	31	0					
364	365	31	0					
365	366	31	0					
366	367	31	0					
367	368	31	0					
368	369	31	0					
369	370	31	0					
370	371	31	0					
371	372	31	0					
372	373	31	0					
373	374	31	0					
374	375	31	0					
375	376	31	0					
376	377	31	0					
377	378	31	0					
378	379	31	0					
379	380	31	0					
380	381	31	0					
381	382	31	0					
382	383	31	0					
383	384	31	0					
384	385	31	0					
385	386	31	0					
386	387	31	0					
387	388	31	0					
388	389	31	0					
389	390	31	0					
390	391	31	0					
391	392	31	0					
392	393	31	0					
393	394	31	0					
394	395	31	0					
395	396	31	0					
396	397	31	0					
397	398	31	0					
398	399	31	0					
399	400	31	0					
400	401	31	0					
401	402	31	0					
402	403	31	0					
403	404	31	0					
404	405	31	0					
405	406	31	0					
406	407	31	0					
407	408	31	0					
408	409	31	0					
409	410	31	0					
410	411	31	0					
411	412	31	0					
412	413	31	0					
413	414	31	0					
414	415	31	0					
415	416	31	0					
416	417	31	0					
417	418	31	0					
418	419	31	0					
419	420	31						

```
0
user_id      68.848868
age          9.351118
annual_income 11403.875717
purchase_amount 140.052062
loyalty_score   1.899047
purchase_frequency 4.562884

dtype: float64
```

```
#14
df[num_cols].min()
df[num_cols].max()
```

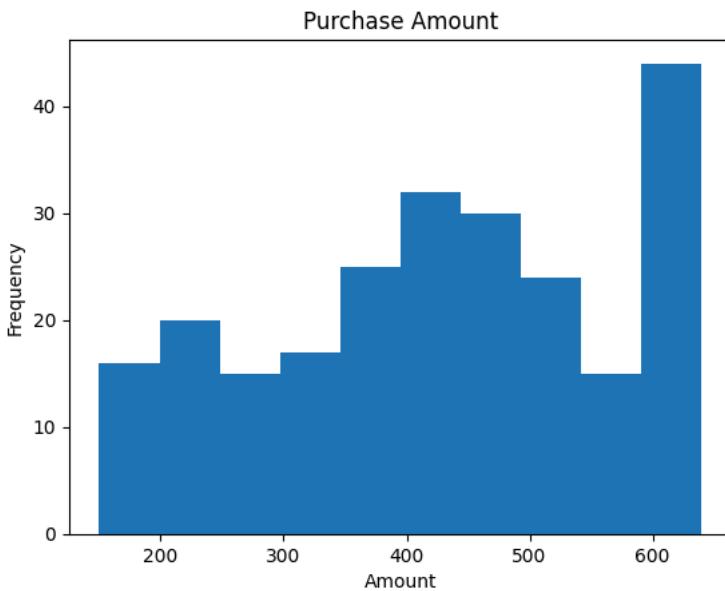
```
0
user_id      238.0
age          55.0
annual_income 75000.0
purchase_amount 640.0
loyalty_score   9.5
purchase_frequency 28.0

dtype: float64
```

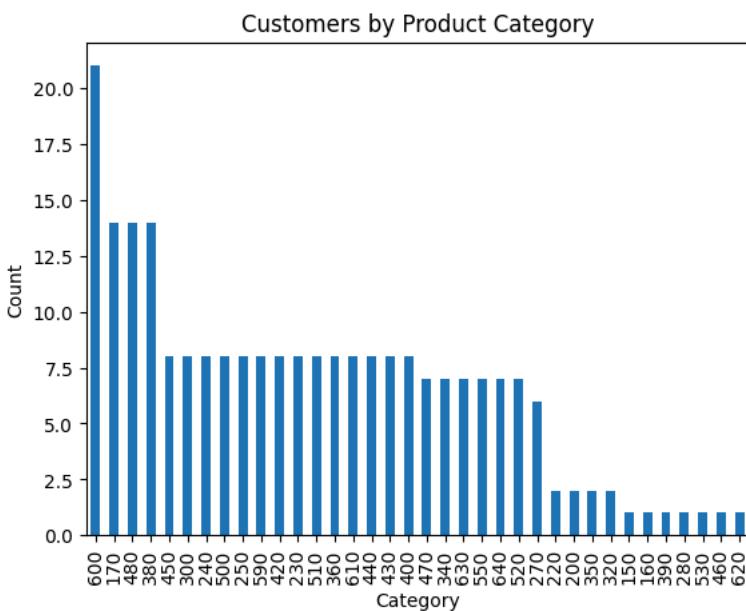
```
#15
df.describe()
```

	user_id	age	annual_income	purchase_amount	loyalty_score	purchase_frequency	grid icon
count	238.000000	238.000000	238.000000	238.000000	238.000000	238.000000	
mean	119.500000	38.676471	57407.563025	425.630252	6.794118	19.798319	
std	68.848868	9.351118	11403.875717	140.052062	1.899047	4.562884	
min	1.000000	22.000000	30000.000000	150.000000	3.000000	10.000000	
25%	60.250000	31.000000	50000.000000	320.000000	5.500000	17.000000	
50%	119.500000	39.000000	59000.000000	440.000000	7.000000	20.000000	
75%	178.750000	46.750000	66750.000000	527.500000	8.275000	23.000000	
max	238.000000	55.000000	75000.000000	640.000000	9.500000	28.000000	

```
#16
plt.figure()
plt.hist(df['purchase_amount'])
plt.title("Purchase Amount")
plt.xlabel("Amount")
plt.ylabel("Frequency")
plt.show()
```

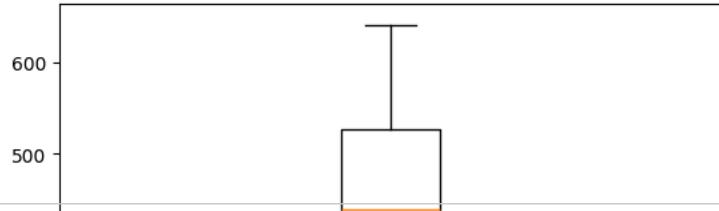


```
#17
plt.figure()
df["purchase_amount"].value_counts().plot(kind='bar')
plt.title("Customers by Product Category")
plt.xlabel("Category")
plt.ylabel("Count")
plt.show()
```



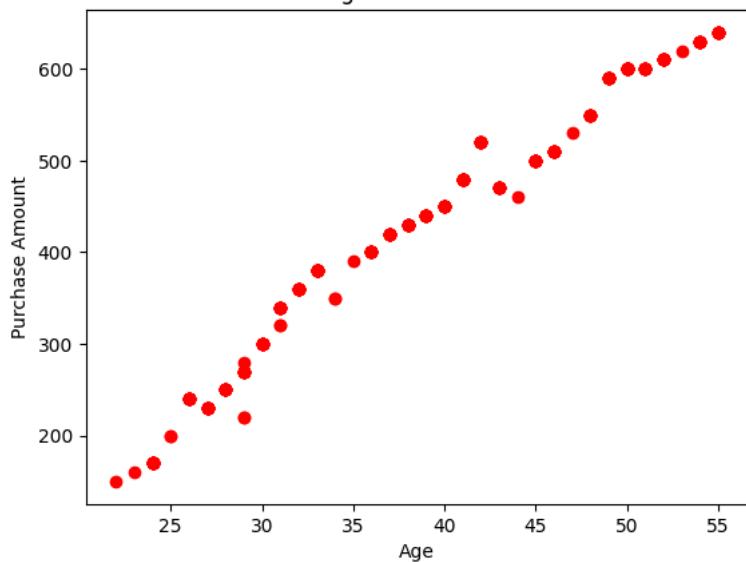
```
#18
plt.figure()
plt.boxplot(df["purchase_amount"])
plt.title("Box Plot of Purchase Amount")
plt.show()
```

Box Plot of Purchase Amount



```
#19
plt.figure()
plt.scatter(df["age"], df["purchase_amount"], color="red")
plt.xlabel("Age")
plt.ylabel("Purchase Amount")
plt.title("Age vs Purchase")
plt.show()
```

Age vs Purchase



```
#20
plt.figure(figsize=(8,6))
sns.heatmap(df[num_cols].corr(), annot=True)
plt.title("Correlation Heatmap")
plt.show()
```

