```
#1. Load both CSV files into separate Pandas DataFrames.
import pandas as pd
import numpy as np

customers_df = pd.read_csv("customers_200_rows.csv")
sales_df = pd.read_csv("sales_200_rows.csv")

#2. Display the first 5 and last 5 rows of each DataFrame.
print(customers_df.head())
print(customers_df.tail())
print(sales_df.head())
print(sales_df.tail())
```

```
   customer_id                   name                        email  \
0         1001           Norma Fisher          ysullivan@yahoo.com
1         1002           Susan Wagner  katelynmontgomery@yahoo.com
2         1003  Dr. Stephanie Collins  thomas15@stewart-bowman.com
3         1004          Joseph Brown    cortezraymond@garrett.com
4         1005              Amy Stark         lindathomas@west.net

                    country signup_date
0                   Lesotho  2023-12-20
1  United States of America  2024-09-16
2                    Mexico  2024-06-22
3                   Ecuador  2023-10-30
4                 Venezuela  2024-07-11
     customer_id             name
email  \
195         1196  Robin Schroeder  robersonjulie@phillips-daniel.biz

196         1197    Madison Hicks         williamsalexis@beasley.biz

197         1198      Emily Weiss            vschneider@williams.com

198         1199     Brandi Simon                isullivan@gmail.com

199         1200     Brianna Pugh            briannajackson@ray.com


         country signup_date
195      Estonia  2023-08-14
196     Slovenia  2024-06-28
197    Australia  2024-12-28
198        Samoa  2024-05-04
199  El Salvador  2023-05-06
   order_id  customer_id  order_date     product     category
quantity  \
0      5001         1071  2023-09-19      Tablet  Electronics
4
1      5002         1035  2022-10-01  Headphones  Accessories
```

```
1
2      5003         1093  2023-04-01      Webcam  Accessories
1
3      5004         1057  2023-07-12  Smartphone  Electronics
1
4      5005         1100  2023-03-13      Laptop  Electronics
2

   price_per_unit
0          399.00
1           89.99
2           59.00
3          599.00
4          789.99
     order_id  customer_id  order_date   product     category
quantity  \
195      5196         1011  2022-05-06   Printer  Electronics
3
196      5197         1045  2022-12-11  Keyboard  Accessories
1
197      5198         1052  2022-12-05    Laptop  Electronics
4
198      5199         1051  2023-08-02     Mouse  Accessories
2
199      5200         1008  2023-01-05   Charger  Accessories
5

     price_per_unit
195          199.99
196           49.99
197          789.99
198           19.99
199           25.50

#3. Show the column names, data types, and check for null values in
both datasets.
print(customers_df.dtypes)
print(customers_df.isnull().sum())
print(sales_df.dtypes)
print(sales_df.isnull().sum())

customer_id     int64
name           object
email          object
country        object
signup_date    object
dtype: object
customer_id    0
name           0
email          0
```

```
country        0
signup_date    0
dtype: int64
order_id              int64
customer_id           int64
order_date           object
product              object
category             object
quantity              int64
price_per_unit      float64
dtype: object
order_id          0
customer_id       0
order_date        0
product           0
category          0
quantity          0
price_per_unit    0
dtype: int64
```

```
#4. Convert the date columns ('signup_date' and 'order_date') to
datetime objects.
customers_df['signup_date'] =
pd.to_datetime(customers_df['signup_date'])
sales_df['order_date'] = pd.to_datetime(sales_df['order_date'])

'''5. Calculate the total revenue for each order (quantity *
price_per_unit) and create a new column
'total_amount'.'''
sales_df['total_amount'] = sales_df['quantity'] *
sales_df['price_per_unit']

#6. Merge the customers and sales datasets on 'customer_id'.
merged_df = pd.merge(sales_df, customers_df, on='customer_id',
how='inner')

#7. Find the top 5 customers who spent the most overall.
top_customers = merged_df.groupby(['customer_id', 'name'])
['total_amount'].sum().nlargest(5).reset_index()
top_customers
```

```
{"summary":"{\n  \"name\": \"top_customers\",\n  \"rows\": 5,\n
\"fields\": [\n    {\n      \"column\": \"customer_id\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
34,\n        \"min\": 1009,\n        \"max\": 1100,\n
\"num_unique_values\": 5,\n        \"samples\": [\n          1071,\n
1052,\n          1081\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"name\",\n      \"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 5,\n        \"samples\": [\n          \"Gerald
```

Garcia\",\n   \"Michael Anderson\",\n   \"Kevin
Fuller\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"total_amount\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 964.2357944921979,\n   \"min\":
5644.95,\n   \"max\": 8003.79,\n   \"num_unique_values\":
5,\n   \"samples\": [\n   7976.91,\n   5644.95,\n
7442.95\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"top_customers"}

```
#8. Count how many customers are from each country.
country_counts = customers_df['country'].value_counts().reset_index()
country_counts.columns = ['country', 'customer_count']
country_counts
```

{"summary":"{\n \"name\": \"country_counts\",\n \"rows\": 132,\n
\"fields\": [\n   {\n   \"column\": \"country\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 132,\n   \"samples\": [\n
\"Brazil\",\n   \"Israel\",\n   \"Argentina\"\
n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"customer_count\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0,\n   \"min\": 1,\n
\"max\": 4,\n   \"num_unique_values\": 4,\n   \"samples\":
[\n   3,\n   1,\n   4\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   }\n]\n}","type":"dataframe","variable_name":"country_counts"}

```
#9. Calculate the average order value per customer.
avg_order_value = merged_df.groupby('customer_id')
['total_amount'].mean().reset_index()
avg_order_value.columns = ['customer_id', 'average_order_value']
avg_order_value
```

{"summary":"{\n \"name\": \"avg_order_value\",\n \"rows\": 86,\n
\"fields\": [\n   {\n   \"column\": \"customer_id\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
28,\n   \"min\": 1001,\n   \"max\": 1100,\n
\"num_unique_values\": 86,\n   \"samples\": [\n   1087,\n
1001,\n   1082\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"average_order_value\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 603.1411942111549,\n   \"min\":
19.99,\n   \"max\": 3159.96,\n   \"num_unique_values\": 77,\
n   \"samples\": [\n   636.75,\n   623.975,\n
699.48\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"avg_order_value"}

```python
#10. Remove any duplicate records from both datasets.
customers_df = customers_df.drop_duplicates()
sales_df = sales_df.drop_duplicates()
sales_df = sales_df[(sales_df['quantity'] >= 0) &
(sales_df['price_per_unit'] >= 0)]
sales_df
```

```
{"summary":"{\n  \"name\": \"sales_df\",\n  \"rows\": 200,\n
\"fields\": [\n    {\n      \"column\": \"order_id\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
57,\n        \"min\": 5001,\n        \"max\": 5200,\n
\"num_unique_values\": 200,\n        \"samples\": [\n          5096,\n
5016,\n          5031\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"customer_id\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 28,\n        \"min\": 1001,\n
\"max\": 1100,\n        \"num_unique_values\": 86,\n
\"samples\": [\n          1036,\n          1071,\n          1061\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"order_date\",\n
\"properties\": {\n        \"dtype\": \"date\",\n        \"min\":
\"2022-01-01 00:00:00\",\n        \"max\": \"2023-12-29 00:00:00\",\n
\"num_unique_values\": 179,\n        \"samples\": [\n          \"2023-
12-17 00:00:00\",\n          \"2022-08-08 00:00:00\",\n
\"2023-01-11 00:00:00\"\n        ],\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"product\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 10,\n
\"samples\": [\n          \"Charger\",\n          \"Headphones\",\n
\"Keyboard\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"category\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          \"Accessories\",\n          \"Electronics\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"quantity\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 1,\n
\"min\": 1,\n        \"max\": 5,\n        \"num_unique_values\": 5,\n
\"samples\": [\n          1,\n          5\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"price_per_unit\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
249.42108956007843,\n        \"min\": 19.99,\n        \"max\":
789.99,\n        \"num_unique_values\": 10,\n        \"samples\": [\n
25.5,\n          89.99\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"total_amount\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 859.767699382816,\n        \"min\":
19.99,\n        \"max\": 3949.95,\n        \"num_unique_values\": 50,\
n        \"samples\": [\n          3159.96,\n          1197.96\n
```

],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"sales_df"}

```
#11. Identify and handle any missing or invalid data (e.g., negative
quantity or price).
print("Missing values in sales_df:")
print(sales_df.isnull().sum())
print("Invalid (negative) quantities:")
print(sales_df[sales_df['quantity'] < 0])

print("\nInvalid (negative) prices:")
print(sales_df[sales_df['price_per_unit'] < 0])
```

```
Missing values in sales_df:
order_id          0
customer_id       0
order_date        0
product           0
category          0
quantity          0
price_per_unit    0
total_amount      0
dtype: int64
Invalid (negative) quantities:
Empty DataFrame
Columns: [order_id, customer_id, order_date, product, category,
quantity, price_per_unit, total_amount]
Index: []

Invalid (negative) prices:
Empty DataFrame
Columns: [order_id, customer_id, order_date, product, category,
quantity, price_per_unit, total_amount]
Index: []
```

```
#12. Group the merged data by category and find: - Total quantity sold
per category - Total revenue per category
category_summary = merged_df.groupby('category').agg(
    total_quantity=('quantity', 'sum'),
    total_revenue=('total_amount', 'sum')
).reset_index()
category_summary
```

{"summary":"{\n \"name\": \"category_summary\",\n \"rows\": 2,\n \"fields\": [\n   {\n   \"column\": \"category\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 2,\n   \"samples\": [\n   \"Electronics\",\n   \"Accessories\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"total_quantity\",\n

```
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
31,\n        \"min\": 281,\n        \"max\": 325,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          281,\n
325\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"total_revenue\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 80884.36078073931,\n        \"min\":
14257.55,\n        \"max\": 128645.31,\n        \"num_unique_values\":
2,\n        \"samples\": [\n          128645.31,\n          14257.55\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\
n}","type":"dataframe","variable_name":"category_summary"}

#13. Create a new column that extracts the year and month from the
'order_date' and analyze monthly sales.
merged_df['year_month'] = merged_df['order_date'].dt.to_period('M')
monthly_sales = merged_df.groupby('year_month')
['total_amount'].sum().reset_index()
monthly_sales

{"summary":"{\n  \"name\": \"monthly_sales\",\n  \"rows\": 24,\n
\"fields\": [\n    {\n      \"column\": \"year_month\",\n
\"properties\": {\n        \"dtype\": \"period[M]\",\n
\"num_unique_values\": 24,\n        \"samples\": [\n          \"2022-
09\",\n          \"2023-05\",\n          \"2022-01\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"total_amount\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1882.3791302958045,\n        \"min\": 3586.39,\n        \"max\":
11263.29,\n        \"num_unique_values\": 24,\n        \"samples\": [\
n          5995.33,\n          4687.38,\n          4180.77\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"monthly_sales"}

#14. Find customers who signed up in the last 6 months but haven't
made any purchases.
recent_customers = customers_df[customers_df['signup_date'] >=
pd.Timestamp.now() - pd.DateOffset(months=6)]
recent_no_purchases =
recent_customers[~recent_customers['customer_id'].isin(sales_df['custo
mer_id'])]
recent_no_purchases[['customer_id', 'name', 'signup_date']]

{"summary":"{\n  \"name\": \"recent_no_purchases[['customer_id',
'name', 'signup_date']]\",\n  \"rows\": 15,\n  \"fields\": [\n    {\n
\"column\": \"customer_id\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 45,\n        \"min\": 1032,\n
\"max\": 1198,\n        \"num_unique_values\": 15,\n
\"samples\": [\n          1159,\n          1175,\n          1032\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"name\",\n      \"properties\":
{\n        \"dtype\": \"string\",\n        \"num_unique_values\": 15,\
n        \"samples\": [\n          \"Amanda Freeman\",\n
\"Margaret Adams\",\n          \"Bradley Robinson\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"signup_date\",\n
\"properties\": {\n        \"dtype\": \"date\",\n        \"min\":
\"2024-12-28 00:00:00\",\n        \"max\": \"2025-06-06 00:00:00\",\n
\"num_unique_values\": 14,\n        \"samples\": [\n          \"2025-
04-24 00:00:00\",\n          \"2025-05-22 00:00:00\",\n
\"2025-06-06 00:00:00\"\n        ],\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe"}
```

```
#15. Identify products that were sold less than 10 times in total (low
performers).
product_sales = sales_df.groupby('product')
['quantity'].sum().reset_index()
low_performers = product_sales[product_sales['quantity'] < 10]
low_performers
```

```
{"repr_error":"Out of range float values are not JSON compliant:
nan","type":"dataframe","variable_name":"low_performers"}
```

```
#16. Create a summary report DataFrame with the following per
customer:
customer_summary = merged_df.groupby(['customer_id', 'name']).agg(
    total_orders=('order_id', 'nunique'),
    total_items=('quantity', 'sum'),
    total_spent=('total_amount', 'sum'),
)
customer_summary['average_order_value'] =
customer_summary['total_spent'] / customer_summary['total_orders']
customer_summary = customer_summary.reset_index()
customer_summary.head()
```

```
{"summary":"{\n  \"name\": \"customer_summary\",\n  \"rows\": 86,\n
\"fields\": [\n    {\n      \"column\": \"customer_id\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
28,\n        \"min\": 1001,\n        \"max\": 1100,\n
\"num_unique_values\": 86,\n        \"samples\": [\n          1087,\n
1001,\n          1082\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"name\",\n      \"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 86,\n        \"samples\": [\n
\"Veronica Bush\",\n          \"Norma Fisher\",\n          \"Kathleen
Ashley\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"total_orders\",\n      \"properties\": {\n        \"dtype\":
```

```
\"number\",\n        \"std\": 1,\n        \"min\": 1,\n
\"max\": 8,\n        \"num_unique_values\": 6,\n        \"samples\":
[\n          1,\n          3,\n          8\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"total_items\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
4,\n        \"min\": 1,\n        \"max\": 25,\n
\"num_unique_values\": 17,\n        \"samples\": [\n          4,\n
10,\n          12\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"total_spent\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1891.1986600494276,\n        \"min\":
19.99,\n        \"max\": 8003.79,\n        \"num_unique_values\": 78,\
n        \"samples\": [\n          1874.47,\n          3159.96,\n
149.97\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"average_order_value\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 603.1411942111549,\n        \"min\":
19.99,\n        \"max\": 3159.96,\n        \"num_unique_values\": 77,\
n        \"samples\": [\n          636.75,\n          623.975,\n
699.48\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"customer_summary"}
```

```python
#17. Use NumPy to perform any custom operation (e.g., apply discount
rule using vectorized operations).
merged_df['discounted_amount'] = np.where(merged_df['total_amount'] >
1000,
                                          merged_df['total_amount'] *
0.9,
                                          merged_df['total_amount'])
merged_df[['order_id', 'total_amount', 'discounted_amount']].head()
```

```
{"summary":"{\n  \"name\": \"merged_df[['order_id', 'total_amount',
'discounted_amount']]\",\n  \"rows\": 5,\n  \"fields\": [\n    {\n
\"column\": \"order_id\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1,\n        \"min\": 5001,\n
\"max\": 5005,\n        \"num_unique_values\": 5,\n
\"samples\": [\n          5002,\n          5005,\n          5003\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"total_amount\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
763.9412425573056,\n        \"min\": 59.0,\n        \"max\": 1596.0,\n
\"num_unique_values\": 5,\n        \"samples\": [\n          89.99,\n
1579.98,\n          59.0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"discounted_amount\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 680.8953972430714,\n
\"min\": 59.0,\n        \"max\": 1436.4,\n
\"num_unique_values\": 5,\n        \"samples\": [\n          89.99,\n
1421.982,\n          59.0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe"}
```