Talking to CSV and Excel files with LangChain

```
pip -q install langchain openai
```

```
ERROR: pip's dependency resolver does not currently take into account
all the packages that are installed. This behaviour is the source of
the following dependency conflicts.
spyder 5.3.3 requires pyqt5<5.16, which is not installed.
spyder 5.3.3 requires pyqtwebengine<5.16, which is not installed.
distributed 2022.7.0 requires tornado<6.2,>=6.0.3, but you have
tornado 6.3.3 which is incompatible.
jupyterlab 3.4.4 requires jupyter-server~=1.16, but you have jupyter-
server 2.7.3 which is incompatible.
jupyterlab-server 2.10.3 requires jupyter-server~=1.4, but you have
jupyter-server 2.7.3 which is incompatible.
notebook 6.5.6 requires jupyter-client<8,>=5.3.4, but you have
jupyter-client 8.4.0 which is incompatible.
notebook 6.5.6 requires pyzmq<25,>=17, but you have pyzmq 25.1.1 which
is incompatible.
panel 0.13.1 requires bokeh<2.5.0,>=2.4.0, but you have bokeh 3.3.0
which is incompatible.
sagemaker-datawrangler 0.4.3 requires sagemaker-data-insights==0.4.0,
but you have sagemaker-data-insights 0.3.3 which is incompatible.
spyder 5.3.3 requires ipython<8.0.0,>=7.31.1, but you have ipython
8.16.1 which is incompatible.
spyder 5.3.3 requires pylint<3.0,>=2.5.0, but you have pylint 3.0.1
which is incompatible.
spyder-kernels 2.3.3 requires ipython<8,>=7.31.1; python_version >=
"3", but you have ipython 8.16.1 which is incompatible.
spyder-kernels 2.3.3 requires jupyter-client<8,>=7.3.4; python_version
>= "3", but you have jupyter-client 8.4.0 which is incompatible.
WARNING: Running pip as the 'root' user can result in broken
permissions and conflicting behaviour with the system package manager.
It is recommended to use a virtual environment instead:
https://pip.pypa.io/warnings/venv
WARNING: There was an error checking the latest version of pip.
Note: you may need to restart the kernel to use updated packages.
```

```python
import os

os.environ["OPENAI_API_KEY"] = "sk-
XJ1QR2wHLYmn9YkHkQ9xT3BlbkFJ5BGpEtnnXwQfwyNvKnvT"
```

```
!pip show langchain
```

```
Name: langchain
Version: 0.0.350
Summary: Building applications with LLMs through composability
```

```
Home-page: https://github.com/langchain-ai/langchain
Author: 
Author-email: 
License: MIT
Location: /opt/conda/lib/python3.10/site-packages
Requires: aiohttp, async-timeout, dataclasses-json, jsonpatch,
langchain-community, langchain-core, langsmith, numpy, pydantic,
PyYAML, requests, SQLAlchemy, tenacity
Required-by: 
```

```python
import pandas as pd
import boto3
from io import BytesIO

s3_bucket = 'filesfornotebook'
s3_key = 'Orders.csv'

# Create an S3 client
s3_client = boto3.client('s3')

response = s3_client.get_object(Bucket=s3_bucket, Key=s3_key)
csv_content = response['Body'].read()

# Try different encodings
try:
    df = pd.read_csv(BytesIO(csv_content), encoding='utf-8')
except UnicodeDecodeError:
    df = pd.read_csv(BytesIO(csv_content), encoding='latin1')

df.head()
```

```
   Row ID        Order ID  Order Date   Ship Date       Ship Mode
Customer ID  \
0       1  CA-2016-152156  08-11-2016  11-11-2016    Second Class
CG-12520
1       2  CA-2016-152156  08-11-2016  11-11-2016    Second Class
CG-12520
2       3  CA-2016-138688  12-06-2016  16-06-2016    Second Class
DV-13045
3       4  US-2015-108966  11-10-2015  18-10-2015  Standard Class
SO-20335
4       5  US-2015-108966  11-10-2015  18-10-2015  Standard Class
SO-20335

     Customer Name    Segment        Country             City  ...  \
0      Claire Gute   Consumer  United States        Henderson  ...
1      Claire Gute   Consumer  United States        Henderson  ...
2  Darrin Van Huff  Corporate  United States      Los Angeles  ...
3   Sean O'Donnell   Consumer  United States  Fort Lauderdale  ...
4   Sean O'Donnell   Consumer  United States  Fort Lauderdale  ...
```

```
  Postal Code  Region       Product ID        Category Sub-
Category  \
0       42420   South  FUR-BO-10001798       Furniture    Bookcases

1       42420   South  FUR-CH-10000454       Furniture       Chairs

2       90036    West  OFF-LA-10000240  Office Supplies       Labels

3       33311   South  FUR-TA-10000577       Furniture       Tables

4       33311   South  OFF-ST-10000760  Office Supplies      Storage


                                        Product Name     Sales
Quantity  \
0                  Bush Somerset Collection Bookcase  261.9600
2
1  Hon Deluxe Fabric Upholstered Stacking Chairs,...  731.9400
3
2  Self-Adhesive Address Labels for Typewriters b...   14.6200
2
3      Bretford CR4500 Series Slim Rectangular Table  957.5775
5
4                     Eldon Fold 'N Roll Cart System   22.3680
2

   Discount    Profit
0      0.00   41.9136
1      0.00  219.5820
2      0.00    6.8714
3      0.45 -383.0310
4      0.20    2.5164

[5 rows x 21 columns]
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Row ID         9994 non-null   int64
 1   Order ID       9994 non-null   object
 2   Order Date     9994 non-null   object
 3   Ship Date      9994 non-null   object
 4   Ship Mode      9994 non-null   object
 5   Customer ID    9994 non-null   object
 6   Customer Name  9994 non-null   object
 7   Segment        9994 non-null   object
 8   Country        9994 non-null   object 
 9   City           9994 non-null   object 
 10  State          9994 non-null   object 
 11  Postal Code    9994 non-null   int64  
 12  Region         9994 non-null   object 
 13  Product ID     9994 non-null   object 
 14  Category       9994 non-null   object 
 15  Sub-Category   9994 non-null   object 
 16  Product Name   9994 non-null   object 
 17  Sales          9994 non-null   float64
 18  Quantity       9994 non-null   int64  
 19  Discount       9994 non-null   float64
 20  Profit         9994 non-null   float64
dtypes: float64(3), int64(3), object(15)
memory usage: 1.6+ MB
```

CSV Agent

NOTE: this agent calls the Pandas DataFrame agent under the hood, which in turn calls the Python agent, which executes LLM generated Python code - this can be bad if the LLM generated Python code is harmful. Use cautiously.

```
!pip install langchain

Requirement already satisfied: langchain in
/opt/conda/lib/python3.10/site-packages (0.0.350)
Requirement already satisfied: PyYAML>=5.3 in
/opt/conda/lib/python3.10/site-packages/PyYAML-6.0-py3.10-linux-
x86_64.egg (from langchain) (6.0)
Requirement already satisfied: SQLAlchemy<3,>=1.4 in
/opt/conda/lib/python3.10/site-packages (from langchain) (1.4.39)
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in
/opt/conda/lib/python3.10/site-packages (from langchain) (3.9.1)
Requirement already satisfied: async-timeout<5.0.0,>=4.0.0 in
/opt/conda/lib/python3.10/site-packages (from langchain) (4.0.3)
Requirement already satisfied: dataclasses-json<0.7,>=0.5.7 in
/opt/conda/lib/python3.10/site-packages (from langchain) (0.6.3)
Requirement already satisfied: jsonpatch<2.0,>=1.33 in
/opt/conda/lib/python3.10/site-packages (from langchain) (1.33)
Requirement already satisfied: langchain-community<0.1,>=0.0.2 in
/opt/conda/lib/python3.10/site-packages (from langchain) (0.0.2)
Requirement already satisfied: langchain-core<0.2,>=0.1 in
/opt/conda/lib/python3.10/site-packages (from langchain) (0.1.0)
Requirement already satisfied: langsmith<0.1.0,>=0.0.63 in
/opt/conda/lib/python3.10/site-packages (from langchain) (0.0.69)
Requirement already satisfied: numpy<2,>=1 in
/opt/conda/lib/python3.10/site-packages (from langchain) (1.26.0)
Requirement already satisfied: pydantic<3,>=1 in
/opt/conda/lib/python3.10/site-packages (from langchain) (2.5.2)
```

```
Requirement already satisfied: requests<3,>=2 in
/opt/conda/lib/python3.10/site-packages (from langchain) (2.31.0)
Requirement already satisfied: tenacity<9.0.0,>=8.1.0 in
/opt/conda/lib/python3.10/site-packages (from langchain) (8.2.3)
Requirement already satisfied: attrs>=17.3.0 in
/opt/conda/lib/python3.10/site-packages (from aiohttp<4.0.0,>=3.8.3-
>langchain) (23.1.0)
Requirement already satisfied: multidict<7.0,>=4.5 in
/opt/conda/lib/python3.10/site-packages (from aiohttp<4.0.0,>=3.8.3-
>langchain) (6.0.4)
Requirement already satisfied: yarl<2.0,>=1.0 in
/opt/conda/lib/python3.10/site-packages (from aiohttp<4.0.0,>=3.8.3-
>langchain) (1.9.4)
Requirement already satisfied: frozenlist>=1.1.1 in
/opt/conda/lib/python3.10/site-packages (from aiohttp<4.0.0,>=3.8.3-
>langchain) (1.4.0)
Requirement already satisfied: aiosignal>=1.1.2 in
/opt/conda/lib/python3.10/site-packages (from aiohttp<4.0.0,>=3.8.3-
>langchain) (1.3.1)
Requirement already satisfied: marshmallow<4.0.0,>=3.18.0 in
/opt/conda/lib/python3.10/site-packages (from dataclasses-
json<0.7,>=0.5.7->langchain) (3.20.1)
Requirement already satisfied: typing-inspect<1,>=0.4.0 in
/opt/conda/lib/python3.10/site-packages (from dataclasses-
json<0.7,>=0.5.7->langchain) (0.9.0)
Requirement already satisfied: jsonpointer>=1.9 in
/opt/conda/lib/python3.10/site-packages (from jsonpatch<2.0,>=1.33-
>langchain) (2.1)
Requirement already satisfied: anyio<5,>=3 in
/opt/conda/lib/python3.10/site-packages (from langchain-
core<0.2,>=0.1->langchain) (3.5.0)
Requirement already satisfied: packaging<24.0,>=23.2 in
/opt/conda/lib/python3.10/site-packages (from langchain-
core<0.2,>=0.1->langchain) (23.2)
Requirement already satisfied: annotated-types>=0.4.0 in
/opt/conda/lib/python3.10/site-packages (from pydantic<3,>=1-
>langchain) (0.6.0)
Requirement already satisfied: pydantic-core==2.14.5 in
/opt/conda/lib/python3.10/site-packages (from pydantic<3,>=1-
>langchain) (2.14.5)
Requirement already satisfied: typing-extensions>=4.6.1 in
/opt/conda/lib/python3.10/site-packages (from pydantic<3,>=1-
>langchain) (4.9.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/opt/conda/lib/python3.10/site-packages (from requests<3,>=2-
>langchain) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in
/opt/conda/lib/python3.10/site-packages (from requests<3,>=2-
>langchain) (3.3)
```

```
Requirement already satisfied: urllib3<3,>=1.21.1 in
/opt/conda/lib/python3.10/site-packages (from requests<3,>=2-
>langchain) (2.0.6)
Requirement already satisfied: certifi>=2017.4.17 in
/opt/conda/lib/python3.10/site-packages (from requests<3,>=2-
>langchain) (2023.7.22)
Requirement already satisfied: greenlet!=0.4.17 in
/opt/conda/lib/python3.10/site-packages (from SQLAlchemy<3,>=1.4-
>langchain) (1.1.1)
Requirement already satisfied: sniffio>=1.1 in
/opt/conda/lib/python3.10/site-packages (from anyio<5,>=3->langchain-
core<0.2,>=0.1->langchain) (1.2.0)
Requirement already satisfied: mypy-extensions>=0.3.0 in
/opt/conda/lib/python3.10/site-packages (from typing-
inspect<1,>=0.4.0->dataclasses-json<0.7,>=0.5.7->langchain) (0.4.3)
WARNING: Running pip as the 'root' user can result in broken
permissions and conflicting behaviour with the system package manager.
It is recommended to use a virtual environment instead:
https://pip.pypa.io/warnings/venv
WARNING: There was an error checking the latest version of pip.
```

```
pip install langchain-experimental
```

```
Collecting langchain-experimental
  Obtaining dependency information for langchain-experimental from
https://files.pythonhosted.org/packages/7c/06/a94b650a8469e161cd07c77e
7866657730a3d0f4317431631f11e7079640/langchain_experimental-0.0.47-
py3-none-any.whl.metadata
  Downloading langchain_experimental-0.0.47-py3-none-any.whl.metadata
(1.9 kB)
Requirement already satisfied: langchain<0.1,>=0.0.350 in
/opt/conda/lib/python3.10/site-packages (from langchain-experimental)
(0.0.350)
Requirement already satisfied: langchain-core<0.2,>=0.1 in
/opt/conda/lib/python3.10/site-packages (from langchain-experimental)
(0.1.0)
Requirement already satisfied: PyYAML>=5.3 in
/opt/conda/lib/python3.10/site-packages/PyYAML-6.0-py3.10-linux-
x86_64.egg (from langchain<0.1,>=0.0.350->langchain-experimental)
(6.0)
Requirement already satisfied: SQLAlchemy<3,>=1.4 in
/opt/conda/lib/python3.10/site-packages (from langchain<0.1,>=0.0.350-
>langchain-experimental) (1.4.39)
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in
/opt/conda/lib/python3.10/site-packages (from langchain<0.1,>=0.0.350-
>langchain-experimental) (3.9.1)
Requirement already satisfied: async-timeout<5.0.0,>=4.0.0 in
/opt/conda/lib/python3.10/site-packages (from langchain<0.1,>=0.0.350-
>langchain-experimental) (4.0.3)
Requirement already satisfied: dataclasses-json<0.7,>=0.5.7 in
```

```
/opt/conda/lib/python3.10/site-packages (from langchain<0.1,>=0.0.350-
>langchain-experimental) (0.6.3)
Requirement already satisfied: jsonpatch<2.0,>=1.33 in
/opt/conda/lib/python3.10/site-packages (from langchain<0.1,>=0.0.350-
>langchain-experimental) (1.33)
Requirement already satisfied: langchain-community<0.1,>=0.0.2 in
/opt/conda/lib/python3.10/site-packages (from langchain<0.1,>=0.0.350-
>langchain-experimental) (0.0.2)
Requirement already satisfied: langsmith<0.1.0,>=0.0.63 in
/opt/conda/lib/python3.10/site-packages (from langchain<0.1,>=0.0.350-
>langchain-experimental) (0.0.69)
Requirement already satisfied: numpy<2,>=1 in
/opt/conda/lib/python3.10/site-packages (from langchain<0.1,>=0.0.350-
>langchain-experimental) (1.26.0)
Requirement already satisfied: pydantic<3,>=1 in
/opt/conda/lib/python3.10/site-packages (from langchain<0.1,>=0.0.350-
>langchain-experimental) (2.5.2)
Requirement already satisfied: requests<3,>=2 in
/opt/conda/lib/python3.10/site-packages (from langchain<0.1,>=0.0.350-
>langchain-experimental) (2.31.0)
Requirement already satisfied: tenacity<9.0.0,>=8.1.0 in
/opt/conda/lib/python3.10/site-packages (from langchain<0.1,>=0.0.350-
>langchain-experimental) (8.2.3)
Requirement already satisfied: anyio<5,>=3 in
/opt/conda/lib/python3.10/site-packages (from langchain-
core<0.2,>=0.1->langchain-experimental) (3.5.0)
Requirement already satisfied: packaging<24.0,>=23.2 in
/opt/conda/lib/python3.10/site-packages (from langchain-
core<0.2,>=0.1->langchain-experimental) (23.2)
Requirement already satisfied: attrs>=17.3.0 in
/opt/conda/lib/python3.10/site-packages (from aiohttp<4.0.0,>=3.8.3-
>langchain<0.1,>=0.0.350->langchain-experimental) (23.1.0)
Requirement already satisfied: multidict<7.0,>=4.5 in
/opt/conda/lib/python3.10/site-packages (from aiohttp<4.0.0,>=3.8.3-
>langchain<0.1,>=0.0.350->langchain-experimental) (6.0.4)
Requirement already satisfied: yarl<2.0,>=1.0 in
/opt/conda/lib/python3.10/site-packages (from aiohttp<4.0.0,>=3.8.3-
>langchain<0.1,>=0.0.350->langchain-experimental) (1.9.4)
Requirement already satisfied: frozenlist>=1.1.1 in
/opt/conda/lib/python3.10/site-packages (from aiohttp<4.0.0,>=3.8.3-
>langchain<0.1,>=0.0.350->langchain-experimental) (1.4.0)
Requirement already satisfied: aiosignal>=1.1.2 in
/opt/conda/lib/python3.10/site-packages (from aiohttp<4.0.0,>=3.8.3-
>langchain<0.1,>=0.0.350->langchain-experimental) (1.3.1)
Requirement already satisfied: idna>=2.8 in
/opt/conda/lib/python3.10/site-packages (from anyio<5,>=3->langchain-
core<0.2,>=0.1->langchain-experimental) (3.3)
Requirement already satisfied: sniffio>=1.1 in
/opt/conda/lib/python3.10/site-packages (from anyio<5,>=3->langchain-
core<0.2,>=0.1->langchain-experimental) (1.2.0)
Requirement already satisfied: marshmallow<4.0.0,>=3.18.0 in
/opt/conda/lib/python3.10/site-packages (from dataclasses-
json<0.7,>=0.5.7->langchain<0.1,>=0.0.350->langchain-experimental)
(3.20.1)
Requirement already satisfied: typing-inspect<1,>=0.4.0 in
/opt/conda/lib/python3.10/site-packages (from dataclasses-
json<0.7,>=0.5.7->langchain<0.1,>=0.0.350->langchain-experimental)
(0.9.0)
Requirement already satisfied: jsonpointer>=1.9 in
/opt/conda/lib/python3.10/site-packages (from jsonpatch<2.0,>=1.33-
>langchain<0.1,>=0.0.350->langchain-experimental) (2.1)
Requirement already satisfied: annotated-types>=0.4.0 in
/opt/conda/lib/python3.10/site-packages (from pydantic<3,>=1-
>langchain<0.1,>=0.0.350->langchain-experimental) (0.6.0)
Requirement already satisfied: pydantic-core==2.14.5 in
/opt/conda/lib/python3.10/site-packages (from pydantic<3,>=1-
>langchain<0.1,>=0.0.350->langchain-experimental) (2.14.5)
Requirement already satisfied: typing-extensions>=4.6.1 in
/opt/conda/lib/python3.10/site-packages (from pydantic<3,>=1-
>langchain<0.1,>=0.0.350->langchain-experimental) (4.9.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/opt/conda/lib/python3.10/site-packages (from requests<3,>=2-
>langchain<0.1,>=0.0.350->langchain-experimental) (2.0.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/opt/conda/lib/python3.10/site-packages (from requests<3,>=2-
>langchain<0.1,>=0.0.350->langchain-experimental) (2.0.6)
Requirement already satisfied: certifi>=2017.4.17 in
/opt/conda/lib/python3.10/site-packages (from requests<3,>=2-
>langchain<0.1,>=0.0.350->langchain-experimental) (2023.7.22)
Requirement already satisfied: greenlet!=0.4.17 in
/opt/conda/lib/python3.10/site-packages (from SQLAlchemy<3,>=1.4-
>langchain<0.1,>=0.0.350->langchain-experimental) (1.1.1)
Requirement already satisfied: mypy-extensions>=0.3.0 in
/opt/conda/lib/python3.10/site-packages (from typing-
inspect<1,>=0.4.0->dataclasses-json<0.7,>=0.5.7-
>langchain<0.1,>=0.0.350->langchain-experimental) (0.4.3)
Downloading langchain_experimental-0.0.47-py3-none-any.whl (162 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 163.0/163.0 kB 2.5 MB/s eta
0:00:00ta 0:00:01
ental
Successfully installed langchain-experimental-0.0.47
WARNING: Running pip as the 'root' user can result in broken
permissions and conflicting behaviour with the system package manager.
It is recommended to use a virtual environment instead:
https://pip.pypa.io/warnings/venv
WARNING: There was an error checking the latest version of pip.
Note: you may need to restart the kernel to use updated packages.
```

```
from langchain_experimental.agents.agent_toolkits import
create_csv_agent
from langchain.llms import OpenAI
```

```
!pip install chardet
```

```
Requirement already satisfied: chardet in
/opt/conda/lib/python3.10/site-packages (4.0.0)
WARNING: Running pip as the 'root' user can result in broken
permissions and conflicting behaviour with the system package manager.
It is recommended to use a virtual environment instead:
https://pip.pypa.io/warnings/venv
WARNING: There was an error checking the latest version of pip.
```

```
import pandas as pd
import chardet
import boto3
from io import BytesIO, StringIO
from langchain_experimental.agents import create_csv_agent

# Specify the S3 bucket and key (file path) of your CSV file
s3_bucket = 'filesfornotebook'
s3_key = 'Orders.csv'

# Create an S3 client
s3_client = boto3.client('s3')

# Download the CSV file from S3
response = s3_client.get_object(Bucket=s3_bucket, Key=s3_key)
csv_content = response['Body'].read()

# Detect the encoding using chardet
result = chardet.detect(csv_content)
encoding = result['encoding']
confidence = result['confidence']
print(f"Detected encoding: {encoding} with confidence {confidence}")

# Try reading the CSV file with the detected encoding
try:
    df = pd.read_csv(BytesIO(csv_content), encoding=encoding)
except UnicodeDecodeError:
    print(f"Reading with {encoding} encoding failed")

# Create an agent with the DataFrame
agent = create_csv_agent(OpenAI(temperature=0),
StringIO(df.to_csv(index=False)), verbose=True)
```

```
Detected encoding: Windows-1252 with confidence 0.73
```

```
# agent = create_csv_agent(OpenAI(temperature=0),
#                          '/content/Employee-Sample-Data.csv',
encoding='latin1',
#                           errors='ignore',
#                           verbose=True)

agent

AgentExecutor(verbose=True,
agent=ZeroShotAgent(llm_chain=LLMChain(prompt=PromptTemplate(input_var
iables=['agent_scratchpad', 'input'], partial_variables={'df_head': "|
|   Row ID | Order ID       | Order Date   | Ship Date    | Ship Mode
| Customer ID   | Customer Name    | Segment    | Country       | City
| State      |   Postal Code | Region    | Product ID      | Category
| Sub-Category   | Product Name
|   Sales |   Quantity |   Discount |    Profit |\
n|---:|---------:|:---------------|:-------------|:------------|:-----
----------|:-------------|:----------------|:----------|:------------
--|:----------------|:-----------|-------------:|:---------|:--------
--------|:---------------|:---------------|:------------------------
-----------------------------------|--------:|-----------:|-----------
:|----------:|\n|  0 |        1 | CA-2016-152156 | 08-11-2016   | 11-
11-2016  | Second Class   | CG-12520      | Claire Gute     | Consumer
| United States | Henderson       | Kentucky   |         42420 | South
| FUR-BO-10001798 | Furniture       | Bookcases      | Bush Somerset
Collection Bookcase                                | 261.96  |          2 |
0    |   41.9136 |\n|  1 |        2 | CA-2016-152156 | 08-11-2016   |
11-11-2016  | Second Class   | CG-12520      | Claire Gute     |
Consumer  | United States | Henderson       | Kentucky   |
42420 | South    | FUR-CH-10000454 | Furniture       | Chairs
| Hon Deluxe Fabric Upholstered Stacking Chairs, Rounded Back | 731.94
|          3 |       0    |  219.582  |\n|  2 |        3 | CA-2016-
138688 | 12-06-2016   | 16-06-2016  | Second Class   | DV-13045      |
Darrin Van Huff | Corporate | United States | Los Angeles     |
California |         90036 | West     | OFF-LA-10000240 | Office
Supplies | Labels         | Self-Adhesive Address Labels for
Typewriters by Universal   |  14.62  |          2 |       0    |
6.8714 |\n|  3 |        4 | US-2015-108966 | 11-10-2015   | 18-10-2015
| Standard Class | SO-20335      | Sean O'Donnell  | Consumer  |
United States | Fort Lauderdale | Florida    |         33311 | South
| FUR-TA-10000577 | Furniture       | Tables         | Bretford CR4500
Series Slim Rectangular Table                 | 957.577 |          5 |
0.45 | -383.031  |\n|  4 |        5 | US-2015-108966 | 11-10-2015   |
18-10-2015  | Standard Class | SO-20335      | Sean O'Donnell  |
Consumer  | United States | Fort Lauderdale | Florida    |
33311 | South    | OFF-ST-10000760 | Office Supplies | Storage
| Eldon Fold 'N Roll Cart System                              |
22.368 |          2 |       0.2  |    2.5164 |"}, template='\nYou are
working with a pandas dataframe in Python. The name of the dataframe
is `df`.\nYou should use the tools below to answer the question posed
of you:\n\npython_repl_ast: A Python shell. Use this to execute python
commands. Input should be a valid python command. When using this
```

```
tool, sometimes output is abbreviated - make sure it does not look
abbreviated before using it in your answer.\n\nUse the following
format:\n\nQuestion: the input question you must answer\nThought: you
should always think about what to do\nAction: the action to take,
should be one of [python_repl_ast]\nAction Input: the input to the
action\nObservation: the result of the action\n... (this
Thought/Action/Action Input/Observation can repeat N times)\nThought:
I now know the final answer\nFinal Answer: the final answer to the
original input question\n\n\nThis is the result of
`print(df.head())`:\n{df_head}\n\nBegin!\nQuestion: {input}\
n{agent_scratchpad}'),
llm=OpenAI(client=<openai.resources.completions.Completions object at
0x7f4b2c746bc0>,
async_client=<openai.resources.completions.AsyncCompletions object at
0x7f4b2c5a7520>, temperature=0.0, openai_api_key='sk-
XJ1QR2wHLYmn9YkHkQ9xT3BlbkFJ5BGpEtnnXwQfwyNvKnvT', openai_proxy='')),
output_parser=MRKLOutputParser(), allowed_tools=['python_repl_ast']),
tools=[PythonAstREPLTool(locals={'df':       Row ID        Order ID
Order Date   Ship Date       Ship Mode  \
0          1  CA-2016-152156  08-11-2016  11-11-2016    Second Class

1          2  CA-2016-152156  08-11-2016  11-11-2016    Second Class

2          3  CA-2016-138688  12-06-2016  16-06-2016    Second Class

3          4  US-2015-108966  11-10-2015  18-10-2015  Standard Class

4          5  US-2015-108966  11-10-2015  18-10-2015  Standard Class

...      ...             ...         ...         ...             ...

9989    9990  CA-2014-110422  21-01-2014  23-01-2014    Second Class

9990    9991  CA-2017-121258  26-02-2017  03-03-2017  Standard Class

9991    9992  CA-2017-121258  26-02-2017  03-03-2017  Standard Class

9992    9993  CA-2017-121258  26-02-2017  03-03-2017  Standard Class

9993    9994  CA-2017-119914  04-05-2017  09-05-2017    Second Class


     Customer ID     Customer Name    Segment        Country
City  \
0       CG-12520       Claire Gute   Consumer  United States
Henderson
1       CG-12520       Claire Gute   Consumer  United States
Henderson
2       DV-13045   Darrin Van Huff  Corporate  United States      Los
Angeles
```

```
3       SO-20335    Sean O'Donnell   Consumer  United States  Fort
Lauderdale
4       SO-20335    Sean O'Donnell   Consumer  United States  Fort
Lauderdale
...          ...               ...        ...            ...
...
9989    TB-21400  Tom Boeckenhauer   Consumer  United States
Miami
9990    DB-13060       Dave Brooks   Consumer  United States
Costa Mesa
9991    DB-13060       Dave Brooks   Consumer  United States
Costa Mesa
9992    DB-13060       Dave Brooks   Consumer  United States
Costa Mesa
9993    CC-12220      Chris Cortes   Consumer  United States
Westminster

           State  Postal Code Region       Product ID         Category
\
0       Kentucky        42420  South  FUR-BO-10001798        Furniture

1       Kentucky        42420  South  FUR-CH-10000454        Furniture

2     California        90036   West  OFF-LA-10000240  Office Supplies

3        Florida        33311  South  FUR-TA-10000577        Furniture

4        Florida        33311  South  OFF-ST-10000760  Office Supplies

...          ...          ...    ...              ...              ...

9989     Florida        33180  South  FUR-FU-10001889        Furniture

9990  California        92627   West  FUR-FU-10000747        Furniture

9991  California        92627   West  TEC-PH-10003645       Technology

9992  California        92627   West  OFF-PA-10004041  Office Supplies

9993  California        92683   West  OFF-AP-10002684  Office Supplies


     Sub-Category                                       Product
Name  \
0       Bookcases                  Bush Somerset Collection Bookcase

1          Chairs  Hon Deluxe Fabric Upholstered Stacking Chairs,...

2          Labels  Self-Adhesive Address Labels for Typewriters b...

3          Tables      Bretford CR4500 Series Slim Rectangular Table
```

```
4         Storage                     Eldon Fold 'N Roll Cart System

...           ...                                                ...

9989  Furnishings                             Ultra Door Pull Handle

9990  Furnishings  Tenex B1-RE Series Chair Mats for Low Pile Car...

9991       Phones                              Aastra 57i VoIP phone

9992        Paper  It's Hot Message Books with Stickers, 2 3/4" x 5"

9993   Appliances  Acco 7-Outlet Masterpiece Power Center, Wihtou...


         Sales  Quantity  Discount    Profit
0     261.9600         2      0.00   41.9136
1     731.9400         3      0.00  219.5820
2      14.6200         2      0.00    6.8714
3     957.5775         5      0.45 -383.0310
4      22.3680         2      0.20    2.5164
...        ...       ...       ...       ...
9989   25.2480         3      0.20    4.1028
9990   91.9600         2      0.00   15.6332
9991  258.5760         2      0.20   19.3932
9992   29.6000         4      0.00   13.3200
9993  243.1600         2      0.00   72.9480

[9994 rows x 21 columns]})])

agent.agent.llm_chain.prompt.template

'\nYou are working with a pandas dataframe in Python. The name of the
dataframe is `df`.\nYou should use the tools below to answer the
question posed of you:\n\npython_repl_ast: A Python shell. Use this to
execute python commands. Input should be a valid python command. When
using this tool, sometimes output is abbreviated - make sure it does
not look abbreviated before using it in your answer.\n\nUse the
following format:\n\nQuestion: the input question you must answer\
nThought: you should always think about what to do\nAction: the action
to take, should be one of [python_repl_ast]\nAction Input: the input
to the action\nObservation: the result of the action\n... (this
Thought/Action/Action Input/Observation can repeat N times)\nThought:
I now know the final answer\nFinal Answer: the final answer to the
original input question\n\n\nThis is the result of
`print(df.head())`:\n{df_head}\n\nBegin!\nQuestion: {input}\
n{agent_scratchpad}'
```

You are working with a pandas dataframe in Python. The name of the dataframe is `df`. You should use the tools below to answer the question posed of you:

python_repl_ast: A Python shell. Use this to execute python commands. Input should be a valid python command. When using this tool, sometimes output is abbreviated - make sure it does not look abbreviated before using it in your answer.

Use the following format:

Question: the input question you must answer Thought: you should always think about what to do Action: the action to take, should be one of [python_repl_ast] Action Input: the input to the action Observation: the result of the action ... (this Thought/Action/Action Input/Observation can repeat N times) Thought: I now know the final answer Final Answer: the final answer to the original input question

This is the result of `print(df.head())`: {df}

Begin! Question: {input} {agent_scratchpad}

```
agent.run("What is the total sales for all the transactions in the dataset?")



> Entering new AgentExecutor chain...
Thought: I need to calculate the total sales for all the transactions in the dataset.
Action: python_repl_ast
Action Input: df['Sales'].sum()
Observation: 2297200.8603000003
Thought: I now know the final answer.
Final Answer: The total sales for all the transactions in the dataset is 2297200.8603000003.

> Finished chain.

'The total sales for all the transactions in the dataset is 2297200.8603000003.'

agent.run("How many units of products were sold in total?")



> Entering new AgentExecutor chain...
Thought: I need to sum up the quantity of all the products
Action: python_repl_ast
Action Input: df['Quantity'].sum()
Observation: 37873
Thought: I now know the final answer
Final Answer: 37873 units of products were sold in total.

> Finished chain.

'37873 units of products were sold in total.'
```

```
agent.run("What is the total sales for south region in the dataset?")



> Entering new AgentExecutor chain...
Thought: I need to calculate the total sales for the south region
Action: python_repl_ast
Action Input: df[df['Region'] == 'South']['Sales'].sum()
Observation: 391721.905
Thought: I now know the final answer
Final Answer: The total sales for south region in the dataset is
391721.905.

> Finished chain.

'The total sales for south region in the dataset is 391721.905.'

agent.run("How many different states are there ?")



> Entering new AgentExecutor chain...
Thought: I need to count the number of unique states
Action: python_repl_ast
Action Input: len(df['State'].unique())
Observation: 49
Thought: I now know the final answer
Final Answer: There are 49 different states.

> Finished chain.

'There are 49 different states.'

agent.run("what is the Total sales of Texas state")



> Entering new AgentExecutor chain...
Thought: I need to find the total sales of Texas state
Action: python_repl_ast
Action Input: df[df['State'] == 'Texas']['Sales'].sum()
Observation: 170188.0458
Thought: I now know the final answer
Final Answer: The total sales of Texas state is 170188.0458.

> Finished chain.

'The total sales of Texas state is 170188.0458.'

agent.run("what is the Total sales of each region")
```

```
> Entering new AgentExecutor chain...
Thought: I need to calculate the total sales of each region
Action: python_repl_ast
Action Input: df.groupby('Region')['Sales'].sum()
Observation: Region
Central    501239.8908
East       678781.2400
South      391721.9050
West       725457.8245
Name: Sales, dtype: float64
Thought: I now know the total sales of each region
Final Answer: Central: 501239.8908, East: 678781.2400, South:
391721.9050, West: 725457.8245

> Finished chain.
```

```
'Central: 501239.8908, East: 678781.2400, South: 391721.9050, West:
725457.8245'
```

```
agent.run("which product has been sold most, by total sales")
```

```
> Entering new AgentExecutor chain...
Thought: I need to find the product with the highest total sales
Action: python_repl_ast
Action Input: df.groupby('Product Name')
['Sales'].sum().sort_values(ascending=False).head(1)
Observation: Product Name
Canon imageCLASS 2200 Advanced Copier    61599.824
Name: Sales, dtype: float64
Thought: I now know the final answer
Final Answer: Canon imageCLASS 2200 Advanced Copier

> Finished chain.
```

```
'Canon imageCLASS 2200 Advanced Copier'
```

```
agent.run("which product has been sold most, by quantity")
```

```
> Entering new AgentExecutor chain...
Thought: I need to find the product with the highest quantity
Action: python_repl_ast
Action Input: df.groupby('Product Name')
['Quantity'].sum().sort_values(ascending=False).head(1)
Observation: Product Name
Staples    215
```

```
Name: Quantity, dtype: int64
Thought: I now know the final answer
Final Answer: Staples has been sold the most, by quantity.

> Finished chain.

'Staples has been sold the most, by quantity.'

agent.run("what is the Total sales of each state")



> Entering new AgentExecutor chain...
Thought: I need to calculate the total sales of each state
Action: python_repl_ast
Action Input: df.groupby('State')['Sales'].sum()
Observation: State
Alabama                   19510.6400
Arizona                   35282.0010
Arkansas                  11678.1300
California               457687.6315
Colorado                  32108.1180
Connecticut               13384.3570
Delaware                  27451.0690
District of Columbia       2865.0200
Florida                   89473.7080
Georgia                   49095.8400
Idaho                      4382.4860
Illinois                  80166.1010
Indiana                   53555.3600
Iowa                       4579.7600
Kansas                     2914.3100
Kentucky                  36591.7500
Louisiana                  9217.0300
Maine                      1270.5300
Maryland                  23705.5230
Massachusetts             28634.4340
Michigan                  76269.6140
Minnesota                 29863.1500
Mississippi               10771.3400
Missouri                  22205.1500
Montana                    5589.3520
Nebraska                   7464.9300
Nevada                    16729.1020
New Hampshire              7292.5240
New Jersey                35764.3120
New Mexico                 4783.5220
New York                 310876.2710
North Carolina            55603.1640
North Dakota                919.9100
```

```
Ohio                        78258.1360
Oklahoma                    19683.3900
Oregon                      17431.1500
Pennsylvania               116511.9140
Rhode Island                22627.9560
South Carolina               8481.7100
South Dakota                 1315.5600
Tennessee                   30661.8730
Texas                      170188.0458
Utah                        11220.0560
Vermont                      8929.3700
Virginia                    70636.7200
Washington                 138641.2700
West Virginia                1209.8240
Wisconsin                   32114.6100
Wyoming                      1603.1360
Name: Sales, dtype: float64
Thought: I now know the total sales of each state
Final Answer: The total sales of each state are listed above.

> Finished chain.

'The total sales of each state are listed above.'

agent.run("Which region has the highest total sales?")



> Entering new AgentExecutor chain...
Thought: I need to calculate the total sales for each region
Action: python_repl_ast
Action Input: df.groupby('Region')['Sales'].sum()
Observation: Region
Central    501239.8908
East       678781.2400
South      391721.9050
West       725457.8245
Name: Sales, dtype: float64
Thought: I now know the final answer
Final Answer: West

> Finished chain.

'West'

agent.run("Which state has the highest total profit in the dataset?")



> Entering new AgentExecutor chain...
Thought: I need to calculate the total profit for each state
```

```
Action: python_repl_ast
Action Input: df.groupby('State')['Profit'].sum()
Observation: State
Alabama                   5786.8253
Arizona                  -3427.9246
Arkansas                  4008.6871
California               76381.3871
Colorado                 -6527.8579
Connecticut               3511.4918
Delaware                  9977.3748
District of Columbia      1059.5893
Florida                  -3399.3017
Georgia                  16250.0433
Idaho                      826.7231
Illinois                -12607.8870
Indiana                  18382.9363
Iowa                      1183.8119
Kansas                     836.4435
Kentucky                 11199.6966
Louisiana                 2196.1023
Maine                      454.4862
Maryland                  7031.1788
Massachusetts             6785.5016
Michigan                 24463.1876
Minnesota                10823.1874
Mississippi               3172.9762
Missouri                  6436.2105
Montana                   1833.3285
Nebraska                  2037.0942
Nevada                    3316.7659
New Hampshire             1706.5028
New Jersey                9772.9138
New Mexico                1157.1161
New York                 74038.5486
North Carolina           -7490.9122
North Dakota               230.1497
Ohio                    -16971.3766
Oklahoma                  4853.9560
Oregon                   -1190.4705
Pennsylvania            -15559.9603
Rhode Island              7285.6293
South Carolina            1769.0566
South Dakota               394.8283
Tennessee                -5341.6936
Texas                   -25729.3563
Utah                      2546.5335
Vermont                   2244.9783
Virginia                 18597.9504
Washington               33402.6517
```

```
West Virginia              185.9216
Wisconsin                 8401.8004
Wyoming                    100.1960
Name: Profit, dtype: float64
Thought: I need to find the state with the highest total profit
Action: python_repl_ast
Action Input: df.groupby('State')['Profit'].sum().idxmax()
Observation: California
Thought: I now know the final answer
Final Answer: California

> Finished chain.

'California'

agent.run("How many different products are there")



> Entering new AgentExecutor chain...
Thought: I need to count the number of unique products
Action: python_repl_ast
Action Input: len(df['Product Name'].unique())
Observation: 1850
Thought: I now know the final answer
Final Answer: There are 1850 different products.

> Finished chain.

'There are 1850 different products.'

agent.run("Give me the overall summary of the documen")



> Entering new AgentExecutor chain...
Thought: I need to get a summary of the dataframe
Action: python_repl_ast
Action Input: df.describe()
Observation:              Row ID   Postal Code         Sales
Quantity     Discount  \
count  9994.000000   9994.000000   9994.000000  9994.000000
9994.000000
mean   4997.500000  55190.379428    229.858001     3.789574
0.156203
std    2885.163629  32063.693350    623.245101     2.225110
0.206452
min       1.000000   1040.000000      0.444000     1.000000
0.000000
25%    2499.250000  23223.000000     17.280000     2.000000
0.000000
```

```
50%    4997.500000  56430.500000     54.490000     3.000000
0.200000
75%    7495.750000  90008.000000    209.940000     5.000000
0.200000
max    9994.000000  99301.000000  22638.480000    14.000000
0.800000

            Profit
count  9994.000000
mean     28.656896
std     234.260108
min   -6599.978000
25%       1.728750
50%       8.666500
75%      29.364000
max    8399.976000
Thought: I now know the overall summary of the document
Final Answer: The overall summary of the document includes the count,
mean, standard deviation, minimum, 25th percentile, 50th percentile,
75th percentile, and maximum for the Row ID, Postal Code, Sales,
Quantity, Discount, and Profit columns.

> Finished chain.

'The overall summary of the document includes the count, mean,
standard deviation, minimum, 25th percentile, 50th percentile, 75th
percentile, and maximum for the Row ID, Postal Code, Sales, Quantity,
Discount, and Profit columns.'
```

```
agent.run("what this file is all about ? what it says overall")
```

```
> Entering new AgentExecutor chain...
Thought: I need to look at the columns and their values
Action: python_repl_ast
Action Input: print(df.columns)
Observation: Index(['Row ID', 'Order ID', 'Order Date', 'Ship Date',
'Ship Mode',
       'Customer ID', 'Customer Name', 'Segment', 'Country', 'City',
'State',
       'Postal Code', 'Region', 'Product ID', 'Category', 'Sub-
Category',
       'Product Name', 'Sales', 'Quantity', 'Discount', 'Profit'],
      dtype='object')

Thought: I can see that this file contains information about orders
Final Answer: This file contains information about orders, such as
order ID, order date, ship date, ship mode, customer ID, customer
name, segment, country, city, state, postal code, region, product ID,
category, sub-category, product name, sales, quantity, discount, and
profit.

> Finished chain.

'This file contains information about orders, such as order ID, order
date, ship date, ship mode, customer ID, customer name, segment,
country, city, state, postal code, region, product ID, category, sub-
category, product name, sales, quantity, discount, and profit.'

agent.run("")

agent.run("")

agent.run("")

agent.run("")
```