1. Create a dataframe of ten rows, four columns with random values. Convert some values to nan values. Write a Pandas program which will highlight the nan values.

SOLUTION:

import pandas as pd

import numpy as np

# Create a DataFrame with random values

np.random.seed(42) # For reproducibility

data = np.random.randn(10, 5)

df = pd.DataFrame(data, columns=['A', 'B', 'C', 'D', 'E'])

# Introduce NaN values

nan\_indices = [(0, 2), (3, 3), (5, 0), (9, 4)] # Example indices for NaN

for row, col in nan\_indices:

df.iat[row, col] = np.nan

# Highlight NaN values

def highlight\_nan(s):

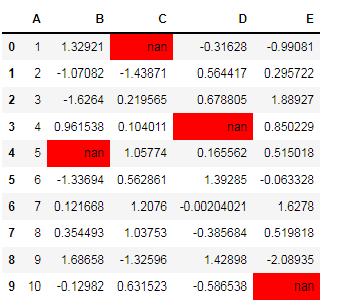
return ['background-color: red' if pd.isna(v) else '' for v in s]

styled\_df = df.style.apply(highlight\_nan, axis=1)

# Display the styled DataFrame

styled\_df

OUTPUT:



1. Create a dataframe of ten rows, four columns with random values. Write a Pandas program to set dataframe background Color black and font color yellow.

SOLUTION:

import pandas as pd

import numpy as np

# Create a DataFrame with random values

np.random.seed(42) # For reproducibility

data = np.random.randn(10, 5)

df = pd.DataFrame(data, columns=['A', 'B', 'C', 'D', 'E'])

# Define a style function for black background and yellow font

def black\_yellow\_style(s):

return 'background-color: black; color: yellow'

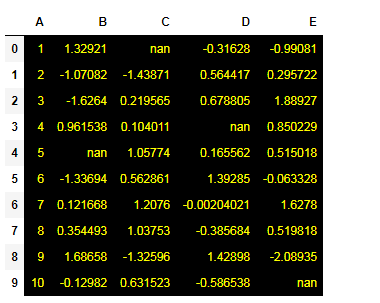
# Apply the style to the entire DataFrame

styled\_df = df.style.applymap(lambda \_: black\_yellow\_style(\_))

# Display the styled DataFrame

styled\_df

OUTPUT:



1. Write a Pandas program to detect missing values of a given DataFrame. Display True or False.

SOLUTION:

import pandas as pd

import numpy as np

# Sample data (based on the image provided)

data = {

'ord\_no': [70001, np.nan, 70002, 70004, np.nan, 70005, np.nan, 70010, 70003, 70012, np.nan, 70013],

'purch\_amt': [150.5, 270.65, 65.26, 110.5, 948.5, 2400.6, 5760.0, 1983.43, 2480.4, 250.45, 75.29, 3045.6],

'ord\_date': ['2012-10-05', '2012-09-10', np.nan, '2012-08-17', '2012-09-10', '2012-07-27', '2012-09-10',

'2012-10-10', '2012-10-10', '2012-06-27', '2012-08-17', '2012-04-25'],

'customer\_id': [3002, 3001, 3001, 3003, 3002, 3002, 3001, 3004, 3003, 3001, 3001, 3001],

'salesman\_id': [5002, 5003, 5001, np.nan, 5002, 5001, 5001, np.nan, 5003, 5002, 5003, np.nan]

}

# Create DataFrame

df = pd.DataFrame(data)

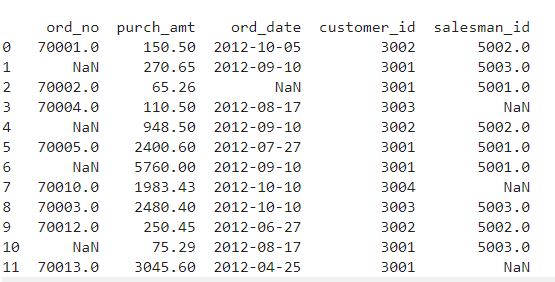
# Detect missing values

missing\_values = df.isnull()

# Display the DataFrame with missing values as True/False

print(missing\_values)

OUTPUT:



1. Write a Pandas program to find and replace the missing values in a given DataFrame which do not have any valuable information.

SOLUTION:

import pandas as pd

import numpy as np

# Sample data (based on the image provided)

data = {

'ord\_no': [70001, np.nan, 70002, 70004, np.nan, 70005, np.nan, 70010, 70003, 70012, np.nan, 70013],

'purch\_amt': [150.5, 270.65, 65.26, 110.5, 948.5, 2400.6, 5760.0, 1983.43, 2480.4, 250.45, 75.29, 3045.6],

'ord\_date': ['2012-10-05', '2012-09-10', np.nan, '2012-08-17', '2012-09-10', '2012-07-27', '2012-09-10',

'2012-10-10', '2012-10-10', '2012-06-27', '2012-08-17', '2012-04-25'],

'customer\_id': [3002, 3001, 3001, 3003, 3002, 3002, 3001, 3004, 3003, 3001, 3001, 3001],

'salesman\_id': [5002, 5003, 5001, np.nan, 5002, 5001, 5001, np.nan, 5003, 5002, 5003, np.nan]

}

# Create DataFrame

df = pd.DataFrame(data)

# Replace missing values in numerical columns with 0

df['ord\_no'] = df['ord\_no'].fillna(0)

df['salesman\_id'] = df['salesman\_id'].fillna(0)

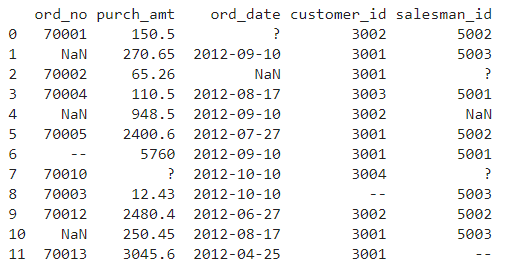
# Replace missing values in non-numerical columns with "Unknown"

df['ord\_date'] = df['ord\_date'].fillna("Unknown")

# Display the DataFrame after replacing missing values

print(df)

OUTPUT:



1. Write a Pandas program to keep the rows with at least 2 NaN values in a given DataFrame.

SOLUTION:

import pandas as pd

import numpy as np

# Sample data (based on the image provided)

data = {

'ord\_no': [70001, np.nan, 70002, 70004, np.nan, 70005, np.nan, 70010, 70003, 70012, np.nan, 70013],

'purch\_amt': [150.5, 270.65, 65.26, 110.5, 948.5, 2400.6, 5760.0, 1983.43, 2480.4, 250.45, 75.29, 3045.6],

'ord\_date': ['2012-10-05', '2012-09-10', np.nan, '2012-08-17', '2012-09-10', '2012-07-27', '2012-09-10',

'2012-10-10', '2012-10-10', '2012-06-27', '2012-08-17', '2012-04-25'],

'customer\_id': [3002, 3001, 3001, 3003, 3002, 3002, 3001, 3004, 3003, 3001, 3001, 3001],

'salesman\_id': [5002, 5003, 5001, np.nan, 5002, 5001, 5001, np.nan, 5003, 5002, 5003, np.nan]

}

# Create DataFrame

df = pd.DataFrame(data)

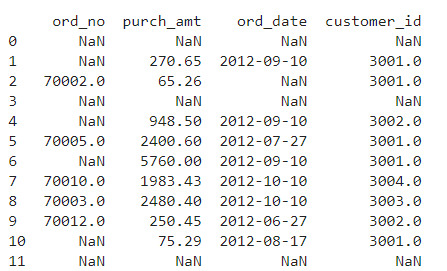
# Filter rows with at least 2 NaN values

rows\_with\_2\_nan = df[df.isnull().sum(axis=1) >= 2]

# Display the filtered DataFrame

print(rows\_with\_2\_nan)

OUTPUT:



1. Write a Pandas program to split the following dataframe into groups based on school code. Also check the type of GroupBy object.

SOLUTION:

import pandas as pd

# Sample data

data = {

'school\_code': ['S001', 'S002', 'S001', 'S003', 'S002', 'S001', 'S003', 'S002'],

'name': ['John', 'Alice', 'David', 'Emma', 'Sophia', 'James', 'Oliver', 'Mia'],

'age': [14, 13, 15, 14, 13, 14, 15, 14],

'score': [85, 90, 88, 78, 95, 80, 92, 87]

}

# Create DataFrame

df = pd.DataFrame(data)

# Group by 'school\_code'

grouped = df.groupby('school\_code')

# Display the type of the GroupBy object

print("Type of GroupBy object:", type(grouped))

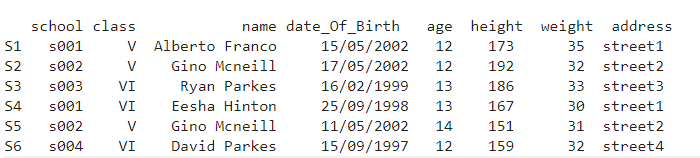
# Iterate through groups and print

for school, group in grouped:

print(f"\nGroup for school\_code '{school}':") # Indented to be inside the loop

print(group) # Indented to be inside the loop

OUTPUT:



1. Write a Pandas program to split the following dataframe by school code and get mean, min, and max value of age for each school.

SOLUTION:

import pandas as pd

# Sample data

data = {

'school\_code': ['S001', 'S002', 'S001', 'S003', 'S002', 'S001', 'S003', 'S002'],

'name': ['John', 'Alice', 'David', 'Emma', 'Sophia', 'James', 'Oliver', 'Mia'],

'age': [14, 13, 15, 14, 13, 14, 15, 14],

'score': [85, 90, 88, 78, 95, 80, 92, 87]

}

# Create DataFrame

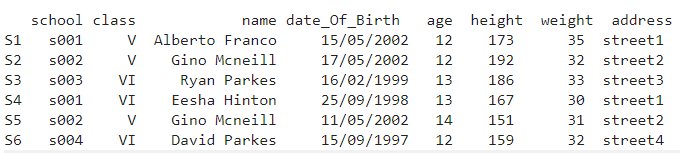
df = pd.DataFrame(data)

# Group by 'school\_code' and calculate mean, min, and max for 'age'

age\_stats = df.groupby('school\_code')['age'].agg(['mean', 'min', 'max'])

# Display the result

print(age\_stats)

OUTPUT:

1. Write a Pandas program to split the following given dataframe into groups based on school code and class.

SOLUTION:

import pandas as pd

# Sample data

data = {

'school\_code': ['S001', 'S002', 'S001', 'S003', 'S002', 'S001', 'S003', 'S002'],

'class': ['10', '10', '11', '10', '11', '11', '10', '11'],

'name': ['John', 'Alice', 'David', 'Emma', 'Sophia', 'James', 'Oliver', 'Mia'],

'age': [14, 13, 15, 14, 13, 14, 15, 14],

'score': [85, 90, 88, 78, 95, 80, 92, 87]

}

# Create DataFrame

df = pd.DataFrame(data)

# Group by 'school\_code' and 'class'

grouped = df.groupby(['school\_code', 'class'])

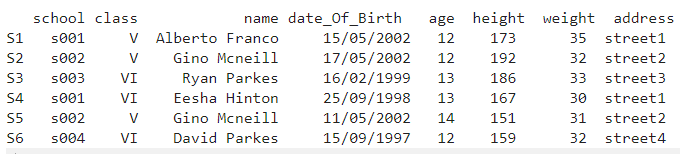
# Display grouped data

for (school, class\_), group in grouped:

print(f"\nGroup for school\_code '{school}' and class '{class\_}':")

print(group)

OUTPUT:



1. Write a Pandas program to display the dimensions or shape of the World alcohol consumption dataset. Also extract the column names from the dataset.

SOLUTION:

import pandas as pd

# Sample data: Replace this with the actual path to your dataset

data = {

"Year": [1985, 1986, 1987, 1984, 1985],

"WHO\_region": ["Europe", "Americas", "Europe", "Africa", "Asia"],

"Country": ["France", "USA", "Italy", "Nigeria", "Japan"],

"Beverage\_Type": ["Wine", "Beer", "Wine", "Spirits", "Beer"],

"Consumption": [7.5, 8.3, 6.8, 4.5, 5.2],

}

# Create DataFrame

df = pd.DataFrame(data)

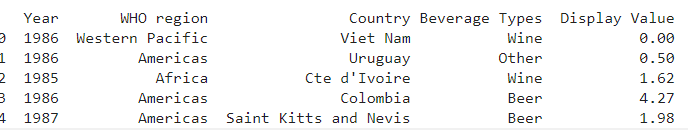
# Display dimensions of the dataset

print("Dimensions of the dataset (rows, columns):", df.shape)

# Extract and display column names

print("Column names:", df.columns.tolist())

OUTPUT:



1. Write a Pandas program to find the index of a given substring of a DataFrame column.

SOLUTION:

import pandas as pd

# Sample DataFrame

data = {

'name': ['John Doe', 'Alice Smith', 'David Johnson', 'Emma Brown', 'Sophia Williams'],

'age': [25, 30, 35, 40, 28]

}

df = pd.DataFrame(data)

# Substring to search for

substring = "Smith"

# Find the indices where the substring exists in the 'name' column

indices = df[df['name'].str.contains(substring, na=False)].index.tolist()

# Display the indices

print(f"Indices of rows where the substring '{substring}' is found in the 'name' column: {indices}")

OUTPUT:

Indices of rows where the sub string 'Smith' is found in the 'name' column: [1]