DSA Assignment 10

Name : Tejas Bajirao Patil
Admission number : u24ai061

---

1. Write a Pandas program to create
a) Date time object for Jan 15 2012.
b) Specific date and time of 9:20 pm.
c) Local date and time.
d) A date without time.
e) Current date.
t) Time from a date time.
g) Current local time.

```
import pandas as pd
from datetime import datetime, date, time

# a) Date time object for Jan 15, 2012
a = pd.to_datetime("2012-01-15")
print("a) Date time object for Jan 15, 2012:", a)

# b) Specific date and time of 9:20 PM
b = pd.to_datetime("2012-01-15 21:20")
print("b) Specific date and time of 9:20 PM:", b)

# c) Local date and time (current system local datetime)
c = pd.Timestamp.now()
print("c) Local date and time:", c)

# d) A date without time
d = pd.to_datetime("2025-04-10").date()
print("d) A date without time:", d)

# e) Current date
e = date.today()
print("e) Current date:", e)
```

```
# f) Time from a date time
f_datetime = pd.to_datetime("2025-04-10 18:45:30")
f = f_datetime.time()
print("f) Time from a date time:", f)

# g) Current local time
g = datetime.now().time()
print("g) Current local time:", g)
```

Q2.Write a Pandas program to convert all the string values to upper, lower cases in a given pandas series. Also find the length of the string values. s = pd.Series (['X', 'Y', 'T', 'Aaba', 'Baca', 'CABA', None, 'bird', 'horse', 'dog'])

```
import pandas as pd

# Create the Series (with correct quotes)
s = pd.Series(['X', 'Y', 'T', 'Aaba', 'Baca', 'CABA', None, 'bird',
'horse', 'dog'])

# Convert to upper case
upper_case = s.str.upper()
print("Upper case:\n", upper_case, "\n")

# Convert to lower case
lower_case = s.str.lower()
print("Lower case:\n", lower_case, "\n")

# Find the length of each string
lengths = s.str.len()
print("Length of strings:\n", lengths)
```

Q3.After accidentally leaving an ice chest of fish and shrimp in your car for a week while you were on vacation, you're now in the market for a new vehicle. Your insurance didn't cover the loss, so you want to make sure you get a good deal on your new car.
Given a Series of car asking_prices and another Series of car fair_prices, determine which cars for sale are a good deal. In other words, identify cars whose asking price is less than their fair price.
The result should be a list of integer indices corresponding to the good deals in asking_prices.

```
import pandas as pd

# Example Series for demonstration
asking_prices = pd.Series([12000, 15000, 18000, 20000, 22000])
fair_prices = pd.Series([13000, 14000, 18500, 21000, 21500])

# Find where asking price is less than fair price
good_deals = asking_prices < fair_prices

# Get indices of good deals
good_deal_indices = asking_prices[good_deals].index.tolist()

print("Indices of good deals:", good_deal_indices)
```

Q4.Whenever your friends John and Judy visit you together, y'all have a party. Given a DataFrame with 10 rows representing the next 10 days of your schedule and whether John

and Judy are scheduled to make an appearance, insert a new column
called days_til_party that indicates how many days until the next party.
days_til_party should be 0 on days when a party occurs, 1 on days when a party doesn't
occur but will occur the next day, etc.

```
import pandas as pd

# Example schedule: random presence of John and Judy over 10 days
df = pd.DataFrame({
    'John': [True, False, True, False, True, False, True, False, False,
True],
    'Judy': [False, False, True, True, False, True, True, False, True,
True]
})

# Step 1: Identify party days
df['party'] = df['John'] & df['Judy']

# Step 2: Create the days_til_party column, filled with a placeholder
days = len(df)
days_til_party = [None] * days

# Step 3: Loop backward to fill in countdown to next party
next_party = float('inf')
for i in reversed(range(days)):
    if df.loc[i, 'party']:
        next_party = 0
    days_til_party[i] = next_party
    if next_party != float('inf'):
        next_party += 1

# Step 4: Add to DataFrame
df['days_til_party'] = days_til_party

# Optional: Drop the helper 'party' column
df.drop(columns='party', inplace=True)
```

```
print(df)
```

Q5.Given a dataset of concerts, count the number of concerts per (artist, venue), per year
month. Make the resulting table be a wide table - one row per year month with a column
for each unique (artist, venue) pair. Use the cross product of the artists and venues Series
to determine which (artist, venue) pairs to include in the result.

```
import pandas as pd
import itertools

concerts = pd.DataFrame({
    'date': pd.to_datetime([
        '2024-01-12', '2024-01-20', '2024-01-21', '2024-02-14'
    ]),
    'artist': ['Taylor', 'Drake', 'Taylor', 'Drake'],
    'venue': ['Madison', 'Barclays', 'Madison', 'Madison']
})

concerts['year_month'] = concerts['date'].dt.to_period('M')

grouped = concerts.groupby(['year_month', 'artist',
'venue']).size().reset_index(name='count')

artists = pd.Series(['Taylor', 'Drake'])
venues = pd.Series(['Madison', 'Barclays'])

artist_venue_pairs = pd.MultiIndex.from_product([artists, venues],
names=['artist', 'venue'])

all_months = concerts['year_month'].drop_duplicates().sort_values()
all_index = pd.MultiIndex.from_product([all_months, artist_venue_pairs],
names=['year_month', 'artist', 'venue'])
```

```
full = grouped.set_index(['year_month', 'artist',
'venue']).reindex(all_index, fill_value=0).reset_index()

wide_table = full.pivot(index='year_month', columns=['artist', 'venue'],
values='count')

wide_table.columns = [f'{a}_{v}' for a, v in wide_table.columns]
wide_table = wide_table.reset_index()

print(wide_table)
```
