Exam 3

fi

The video contains the following file (also available in github repo) that you will need for this test:

`widget_sales.csv`

This CSV file contains sales data for some widgets, and contains the following columns:

- `widget` : the name of the widget (string)
- `date_sold` : an epoch indicating date and time sale occurred (integer)
- `quantity_sold` : the number of widgets sold (integer)
- `unit_price` : the unit price widget was sold for (float)
- `tax` : the tax that was added to the sale price (float)
- `discount` : the discount that was subtracted from the sale price (float)

All the questions on this exam relate to the data contained in this file.

Q1

First thing is to load the data into a Pandas dataframe.

We do this by running the following code:

In [1]:

```python
import pandas as pd
df = pd.read_csv('widget_sales.csv')
```

The expected data type for the `quantity_sold` column is an integer, what are the expected and actual Numpy data types for that column in the loaded dataframe?

- a. expected = `int64` , actual = `int64`
- b. expected = `int64` , actual = `float64`
- c. expected = `float64` , actual = `object`
- d. expected = `float64` , actual = `float64`

Q2

Why is the expected and actual data type for `quantity_sold` not the same?

- a. The csv data contains some float values for that column, not just integers
- b. There's a bug in Pandas
- c. That column has some missing values

- d. The expected and actual data types are the same - nothing to see here, move along!

Q3

Inspect the data frame to determine which columns have missing (null) values.

- a. `widget` , `quantity_sold` only
- b. `date_sold` , `quantity_sold` only
- c. `quantity_sold` only
- d. there are no missing values anywhere

Q4

We want to create a new dataframe (named `df_not_null`) that does not have any missing values. We want to do this by removing all **rows** that contain null values.

Which of the following expressions will achieve this?

I.

```
df_temp = df[pd.notnull(df['widget'])]
df_not_null =
df_temp[pd.notnull(df_temp['quantity_sold'])]
```

II.

```
df_not_null = df.dropna(axis=0)
```

III.

```
df_not_null = df.dropna()
```

- a. I only
- b. II and III only
- c. none of them
- d. all of them

Q5

Assume that `df_not_null` is the result of correctly removing any rows in the original data frame that contained any null values.

Inspect this new data frame - what is the data type of `quantity_sold` ?

- a. `object`

- b. `int64`
- c. `float64`
- d. `uint64`

Q6

The data type for `quantity_sold` can be changed to be an integer since we expect all non-null values to be positive integers.

Assuming `df_not_null` is a dataframe that contains no null values (derived from our original dataframe `df`), which of the following code results in `data` being a dataframe that contains the `quantity_sold` column as an `int64` data type?

I.

```
quantity_sold =
df_not_null['quantity_sold'].astype(int)
data = pd.concat(
    [
        df_not_null[['widget', 'date_sold',
'unit_price', 'tax', 'discount']],
        quantity_sold
    ],
    axis=1
)
```

II.

```
quantity_sold =
df_not_null['quantity_sold'].astype(int)
data = df_not_null.drop('quantity_sold', axis=1)
data = pd.concat([data, quantity_sold], axis=1,
join='inner')
```

III.

```
quantity_sold =
df['quantity_sold'].dropna().astype(int)
data = df.drop('quantity_sold', axis=1)
data = pd.concat([data, quantity_sold], axis=1,
join='inner')
```

- a. I only
- b. II only
- c. III only
- d. I and II only

Q7

The net sale for each row is given by:

$$\texttt{quantity\_sold} * \texttt{unit\_price} + \texttt{tax} - \texttt{discount}$$

Calculate the total net sales of the data contained in the dataframe that has all rows with null values removed, rounded to 2 decimal points.

The answer is:

- a. `8383874.86`
- b. `7963979.82`
- c. `8380560.82`
- d. `99355.0`

Q8

Identify the date on which the **second** highest net sale for widget `AAA` occurred.

Just as before, the net sale formula is given by: $\texttt{quantity\_sold} * \texttt{unit\_price} + \texttt{tax} - \texttt{discount}$

This sale happened on this date: s

- a. 2020-01-21T16:22:07
- b. 2020-01-21T21:39:52
- c. 2020-01-26T00:30:51
- d. 2020-01-23T21:31:26

Q9

Calculate the number of rows (sales) that each widget has generated (limit your dataframe to rows that contain no null values).

Represented as a dictionary, the result is:

a.

```
{'AAA': 1646, 'BBB': 1705, 'CCC': 1653, 'DDD':
1666, 'EEE': 1668, 'FFF': 1661}
```

b.

```
{'AAA': 1647, 'BBB': 1704, 'CCC': 1652, 'DDD':
1667, 'EEE': 1667, 'FFF': 1661}
```

c.

```
{'AAA': 1666, 'BBB': 1666, 'CCC': 1666, 'DDD':
1666, 'EEE': 1666, 'FFF': 1670}
```

d.

```
{'AAA': 1647, 'BBB': 1705, 'CCC': 1652, 'DDD':
1667, 'EEE': 1667, 'FFF': 1661, NAN: 1}
```

Q10

Limiting your dataframe to rows with non-null values only, calculate the average percentage discount (rounded to 2 digits after the decimal point) of each widget.

(The percentage discount for a specific row in the dataframe is given by:

```
discount / (quantity_sold * unit_price) * 100
```

)

Represented as a dictionary, the result is:

a.

```
{'AAA': 2.44, 'BBB': 2.5, 'CCC': 2.49, 'DDD':
2.45, 'EEE': 2.48, 'FFF': 2.46}
```

b.

```
{'AAA': 2.35, 'BBB': 2.49, 'CCC': 2.42, 'DDD':
2.39, 'EEE': 2.44, 'FFF': 2.47}
```

c.

```
{'AAA': 2.14, 'BBB': 2.23, 'CCC': 2.2, 'DDD':
2.14, 'EEE': 2.24, 'FFF': 2.2}
```

d.

```
{'AAA': 2.65, 'BBB': 2.76, 'CCC': 2.71, 'DDD':
2.63, 'EEE': 2.72, 'FFF': 2.7}
```