```
"""VIII. Exersises
1. Sales
Fill in the TODO cells in sales.ipynb notebook.
✓ Fix column datatypes.
✓ Drop if duplicated or null.
✓ Sanity check for value ranges and to check assumptions.
✓ Use regular expression and lambda function to parse data.
"""
```

```
'VIII. Exersises\n1. Sales\nFill in the TODO cells in sales.ipynb notebook.\n✓ Fi
x column datatypes.\n✓ Drop if duplicated or null.\n✓ Sanity check for value ran
ges and to check assumptions.\n✓ Use regular expression and lambda function to pa
```

```
import pandas as pd
df = pd.read_csv('sales.csv')
df.head()
```

	order_id	name	ordered_at	price	quantity	line_total
0	10000	"ICE CREAM" Peanut Fudge	2018-01-01 11:30:00	$3.50	3.0	$10.50
1	10000	"ICE CREAM" Peanut Fudge	2018-01-01 11:30:00	$3.50	1.0	$3.50
2	10001	"SORBET" Raspberry	2018-01-01 12:14:54	$2.50	2.0	$5.00
			2018-01-01			

```
#✓ Fix column datatypes.


df['name'] = df['name'].str.replace('"', '')
df['ordered_at'] = pd.to_datetime(df['ordered_at'])
df['price'] = df['price'].replace('[$,]', '', regex=True).astype(float)
df['line_total'] = df['line_total'].replace('[$,]', '', regex=True).astype(float)
print(df.dtypes)
```

```
order_id               int64
name                  object
ordered_at    datetime64[ns]
price                float64
quantity             float64
line_total           float64
dtype: object
```

```
#✓ Drop if duplicated or null.


df.drop_duplicates(inplace=True)
df.dropna(inplace=True)
df.head()
```

	order_id	name	ordered_at	price	quantity	line_total
0	10000	ICE CREAM Peanut Fudge	2018-01-01 11:30:00	3.5	3.0	10.5
1	10000	ICE CREAM Peanut Fudge	2018-01-01 11:30:00	3.5	1.0	3.5
2	10001	SORBET Raspberry	2018-01-01 12:14:54	2.5	2.0	5.0

```
#✓ Sanity check for value ranges and to check assumptions.

print(df['quantity'].describe())
print(df['price'].describe())
print(df['line_total'].describe())

print(df['name'].value_counts())

print(df['ordered_at'].min())
print(df['ordered_at'].max())
```

```
count    15098.000000
mean         1.995761
std          0.820828
min          1.000000
25%          1.000000
50%          2.000000
75%          3.000000
max          3.000000
Name: quantity, dtype: float64
count    15098.000000
mean         2.469466
std          1.174679
min         -4.000000
25%          1.500000
50%          2.500000
75%          3.500000
max          4.000000
Name: price, dtype: float64
count    15098.000000
mean         4.935025
std          3.277588
min        -12.000000
25%          2.500000
50%          4.500000
75%          7.500000
max         12.000000
Name: line_total, dtype: float64
SORBET Blood Orange             594
ICE CREAM Mint Chip             591
CONE Cookie Cone                590
ICE CREAM Dark Chocolate        589
SORBET Lemon                    588
ICE CREAM Candied Bacon         586
ICE CREAM Maple Brown Sugar     581
BEVERAGE Iced Coffee            581
SORBET Raspberry                571
SORBET Watermelon               569
CONE Sugar Cone                 564
CONE Dipped Waffle Cone         562
ICE CREAM Double Fudge Chunk    559
ICE CREAM Rocky Road            556
ICE CREAM Wildberry             553
```

```
CONE Brownie Cone               553
BEVERAGE Tea                    550
SORBET Lychee                   547
ICE CREAM Strawberry            545
ICE CREAM Dulce De Leche        543
ICE CREAM Peanut Fudge          540
ICE CREAM Vanilla Bean          539
CONE Waffle Cone                535
ICE CREAM Matcha                534
BEVERAGE Espresso               531
MISC Ice Cream Cake             527
ICE CREAM Earl Gray             520
Name: name, dtype: int64
2018-01-01 11:30:00
2018-04-23 00:57:34
```

```
#✓ Use regular expression and lambda function to parse data.
```

```
df.head()
```

	order_id	name	ordered_at	price	quantity	line_total
0	10000	ICE CREAM Peanut Fudge	2018-01-01 11:30:00	3.5	3.0	10.5
1	10000	ICE CREAM Peanut Fudge	2018-01-01 11:30:00	3.5	1.0	3.5
2	10001	SORBET Raspberry	2018-01-01 12:14:54	2.5	2.0	5.0
3	10001	CONE Dipped Waffle	2018-01-01	2.5	1.0	2.5

```
"""2. Job Market
Given the job market data in csv file. Create your own jupyter notebook and explore the data by:
✓ Load the data using Pandas.
✓ Visualize top 10 first rows
✓ Fix column datatypes.
✓ Check and clean the data.
✓ Load the data using Pandas.
"""
```

```
'2. Job Market\nGiven the job market data in csv file. Create your own jupyter no
tebook and explore the data by:\n✓ Load the data using Pandas.\n✓ Visualize top
10 first rows\n✓ Fix column datatypes.\n✓ Check and clean the data.\n✓ Load the
```

```
data = pd.read_csv('job-market (1).csv')
data.head(10)
```

```
data.info()
data.dropna(inplace=True)
data.drop_duplicates(inplace=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40789 entries, 0 to 40788
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
```

```
 0   Id                 10099 non-null  float64
 1   Title              10099 non-null  object
 2   Company            9483 non-null   object
 3   Date               10099 non-null  object
 4   Location           10099 non-null  object
 5   Area               6392 non-null   object
 6   Classification     10099 non-null  object
 7   SubClassification  10099 non-null  object
 8   Requirement        10099 non-null  object
 9   FullDescription    9843 non-null   object
 10  LowestSalary       10099 non-null  float64
 11  HighestSalary      10099 non-null  float64
 12  JobType            9852 non-null   object
dtypes: float64(3), object(10)
memory usage: 4.0+ MB
```

```
import re

def clean_string(text):
    if text:
        # Remove HTML tags
        clean = re.compile('<.*?>')
        text = re.sub(clean, '', text)
        # Replace ' ' and '\n * ' with a space
        text = re.sub(r'( |\n * )', ' ', text)
        # Remove unwanted characters
        text = re.sub(r'\n *|\*', '', text)

        return text if text else None


data['Id'] = data['Id'].astype(int)
data['LowestSalary'] = data['LowestSalary'].astype(int)
data['HighestSalary'] = data['HighestSalary'].astype(int)
data['Date'] = pd.to_datetime(data['Date'], format='%Y-%m-%dT%H:%M:%S.%fZ')
data['LowestSalary'] = pd.to_numeric(data['LowestSalary'], errors='coerce')
data['HighestSalary'] = pd.to_numeric(data['HighestSalary'], errors='coerce')
data['FullDescription'] = data['FullDescription'].apply(clean_string)


#Check and clean the data


print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5898 entries, 121 to 10098
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Id                 5898 non-null   int64
 1   Title              5898 non-null   object
 2   Company            5898 non-null   object
 3   Date               5898 non-null   datetime64[ns]
 4   Location           5898 non-null   object
 5   Area               5898 non-null   object
 6   Classification     5898 non-null   object
 7   SubClassification  5898 non-null   object
 8   Requirement        5898 non-null   object
```

```
 9   FullDescription    5898 non-null   object
 10  LowestSalary       5898 non-null   int64
 11  HighestSalary      5898 non-null   int64
 12  JobType            5898 non-null   object
dtypes: datetime64[ns](1), int64(3), object(9)
memory usage: 645.1+ KB
None
```

```
print(data.describe())
```

```
                 Id  LowestSalary  HighestSalary
count  5.898000e+03   5898.000000    5898.000000
mean   3.739050e+07     24.560868      42.390641
std    2.780707e+04     20.621646      11.656813
min    3.679829e+07      0.000000      30.000000
25%    3.738720e+07      0.000000      30.000000
50%    3.739687e+07     30.000000      40.000000
75%    3.740149e+07     40.000000      50.000000
max    3.740440e+07     50.000000      60.000000
```

```
print(data.isnull().sum())
```

```
Id                   0
Title                0
Company              0
Date                 0
Location             0
Area                 0
Classification       0
SubClassification    0
Requirement          0
FullDescription      0
LowestSalary         0
HighestSalary        0
JobType              0
dtype: int64
```

```
data.head(10)
```

	Id	Title	Company	Date	Location	Area	Classif
121	37404238	Fabricator/Installer	WORKPLACE ACCESS & SAFETY	2018-10-07	Melbourne	Bayside & South Eastern Suburbs	T
122	37404195	Boilermaker	RPM Contracting QLD P/I	2018-10-07	Brisbane	Southern Suburbs & Logan	T
125	37404288	Casual Childcare Positions | Bondi Junction	anzuk Education	2018-10-07	Sydney	CBD, Inner West & Eastern Suburbs	Edu
126	37404267	Technician	Zoom Recruitment & Training	2018-10-07	Sydney	South West & M5 Corridor	Eng
127	37404230	Systems Engineer	Humanised Group	2018-10-07	Brisbane	CBD & Inner Suburbs	Inforr Commu Tec
129	37404237	SENIOR MARKETING & PRODUCT MANAGER	Credit Repair Australia Pty Ltd	2018-10-07	Sydney	South West & M5 Corridor	Mar Commur
130	37404370	Operations Delivery Manager	Woolworths Group	2018-10-07	Sydney	CBD, Inner West & Eastern Suburbs	Inforr Commu Tec
131	37404228	General Manager	Multiple Sclerosis SA and NT Inc	2018-10-07	Sydney	Inner West & Eastern Suburbs	CEO & Mana
132	37404226	General Manager	Multiple Sclerosis SA and NT Inc	2018-10-07	Melbourne	CBD & Inner Suburbs	CEO & Mana
133	37404174	Technical Support Executive - $70K + Super - C	Command Group	2018-10-07	Sydney	CBD, Inner West & Eastern	Inforr Commu

✓ 0s completed at 10:41 PM