Assignment-1 : Part-1

Question-1: (5 Marks)

Take employee id(integer), employee name(string) and salary(float) details of 4 employees from user. Calculate Bonus for the employees. It Should be calculated as follows: 1.Declare no bonus message if the salary is <5000 2.Give 25% bonus if the salary is between 5000 to 10000 3.Give 20% bonus if the salary is between 10001 to 20001 4.Give 10% bonus for all others

In [1]:
```
def calculate_bonus(salary):
    if salary < 5000:
        return "No bonus"
    elif 5000 <= salary <= 10000:
        return salary * round(0.25, 2)
    elif 10001 <= salary <= 20000:
        return salary * round(0.20, 2)
    else:
        return salary * round(0.10, 2)
employee_details = []
for i in range(1, 5):
    print(f"Enter details for employee {i}:")
    emp_id = int(input("Employee ID: "))
    emp_name = input("Employee Name: ")
    emp_salary = float(input("Employee Salary: "))
    employee_details.append((emp_id, emp_name, emp_salary))
print("\nEmployee Details with Bonus:")
for emp_id, emp_name, emp_salary in employee_details:
    bonus = calculate_bonus(emp_salary)
    print(f"Employee ID: {emp_id}, Employee Name: {emp_name}, Salary: {emp_salary}, Bonu
```

```
Enter details for employee 1:
Employee ID: 1
Employee Name: von
Employee Salary: 4000
Enter details for employee 2:
Employee ID: 2
Employee Name: von2
Employee Salary: 6000
Enter details for employee 3:
Employee ID: 3
Employee Name: von3
Employee Salary: 12000
Enter details for employee 4:
Employee ID: 4
Employee Name: von4
Employee Salary: 21000

Employee Details with Bonus:
Employee ID: 1, Employee Name: von, Salary: 4000.0, Bonus: No bonus
Employee ID: 2, Employee Name: von2, Salary: 6000.0, Bonus: 1500.0
Employee ID: 3, Employee Name: von3, Salary: 12000.0, Bonus: 2400.0
Employee ID: 4, Employee Name: von4, Salary: 21000.0, Bonus: 2100.0
```

Question 2: (5 Marks)

Write a program that accepts a sequence of whitespace separated words as input and prints the words after removing all duplicate words and sorting them alphanumerically. Input is as follows.

Amazon Black Friday Sales are already Live- here’s what to shop now

In [2]:
```
s = input("Amazon Black Friday Sales are already Live- here’s what to shop now")
words = [word for word in s.split(" ")]
print(" ".join(sorted(list(set(words)))))
```

```
Amazon Black Friday Sales are already Live- here’s what to shop nowAmazon Black Friday S
ales are already Live- here’s what to shop now
Amazon Black Friday Live- Sales already are here’s now shop to what
```

Question-3 (5 Marks)

Oxford University is offereing MBA courses with the following specializations specializationlist= ['HR','Operations','Strategy','Marketing','Finance'] Now Write a program to accept specialization preference from the user and check whether his/her choice is in the list of offered specialization or not using **conditional/loop/both statements.** Display the message "yes we found the course at postion {}" if it is in the list otherwise display "no". The execution must stop whenever the match occurs. It must also display the position of the input in the list. Even if the user provides say "hr" instead of HR it should display yes message. Means the user input needs to be case insensitive

In [3]:
```
specialization_list = ['HR', 'Operations', 'Strategy', 'Marketing', 'Finance']

input_user = input("the input string is: ")

found = False
for index, specialization in enumerate(specialization_list):
    if input_user.lower() == specialization.lower():
        print(f"Yes we found the course {input_user} at position {index + 1}")
        found = True
        break

if not found:
    print("No course hr is not matching with the user input operations")
```

```
the input string is: operations
Yes we found the course operations at position 2
```

Assignment-1: Part2

Question-1-10 Marks

Create a function email_to_fullname(email) which accepts email as a string input parameter and returns a string containing the last and first name separated by a comma. Note that output string should be uppercase

In [4]:
```
import pandas as pd

data = {'email': ['mark@example.com', 'sara@example.com', 'peter@example.com', 'lisa@exa
                  'mike@example.com', 'emily@example.com', 'james@example.com', 'linda@e
                  'chris@example.com', 'amy@example.com'],
        'name': ['Mark Miller', 'Sara Johnson', 'Peter Smith', 'Lisa White', 'Mike Davis
                 'Emily Wilson', 'James Brown', 'Linda Taylor', 'Chris Anderson', 'Amy J
table = pd.DataFrame(data)

def email_to_fullname(email):
    matching_row = table[table['email'] == email.lower()]
    if not matching_row.empty:
        name = matching_row['name'].values[0]
```

```
        first_name, last_name = full_name.split(' ', 1)
        result = f"{last_name}, {first_name}".upper()
        return result
    else:
        return "Email not found in the table"

email_input = 'mike@example.com'
output = email_to_fullname(email_input)
print(output)
```

```
DAVIS, MIKE
```

Question-2:Marks 10

Generate a dictionary which when converted to a pandas data frame, produces the output given below.

In [5]:
```
import pandas as pd

data = {
    "Adobe": ["US$2.95 billion (2019)", "US$3.27 billion (2019)", "US$11.17 billion (201
    "Amazon": ["US$11.588 billion (2019)", "US$14.541 billion (2019)", "US$280.522 billi
    "Amdocs": ["US$11.588 billion (2019)", "US$14.541 billion (2019)", "US$280.522 billi
}

financial_metrics = ["Net income", "Operating income", "Revenue", "Total assets"]

financial_table = pd.DataFrame(data, index=financial_metrics)

financial_table
```

Out[5]:

	Adobe	Amazon	Amdocs
Net income	US$2.95 billion (2019)	US$11.588 billion (2019)	US$11.588 billion (2019)
Operating income	US$3.27 billion (2019)	US$14.541 billion (2019)	US$14.541 billion (2019)
Revenue	US$11.17 billion (2019)	US$280.522 billion (2019)	US$280.522 billion (2019)
Total assets	US$20.76 billion (2019)	US$225.248 billion (2019)	US$225.248 billion (2019)

Question-3-Marks 10

Create a new text file called strategy.txt and write the following lines using file commands:

A business strategy creates a vision and direction for the whole organization. It is important that all people within a company have clear goals and are following the direction, or mission of the organization.

Then do the following operations:

Display the contents of the file

In [6]:
```
with open('strategy.txt', 'w') as file:
  file.write("A business strategy creates a vision and direction for the whole organizat
  file.write("It is important that all people within a company have clear goals and are

with open('strategy.txt', 'r') as file:
  contents = file.read()
  print(contents)
```

```
A business strategy creates a vision and direction for the whole organization.
It is important that all people within a company have clear goals and are following the
direction, or mission of the organization.
```

Now append the sentence – “This is the final statement”

In [7]:
```
with open("strategy.txt", "a") as file:
  file.write(" This is the final statement.")

with open("strategy.txt", "r") as file:
    contents = file.read()
    print(contents)
```

```
A business strategy creates a vision and direction for the whole organization.
It is important that all people within a company have clear goals and are following the
direction, or mission of the organization. This is the final statement.
```

Now find the current cursor position in the file.

In [8]:
```
with open('strategy.txt', 'r') as file:
    str_content = file.read()
    cursor_position = file.tell()
    print(f"\nThe current cursor position is: {cursor_position}")
```

```
The current cursor position is: 238
```

Now bring back the file cursor into 12th position.

In [9]:
```
with open('strategy.txt', 'r') as file:
  file.seek(12)
  cursor_position_after_move = file.tell()
  print(f"\nMove the cursor to {cursor_position_after_move}th position")
```

```
Move the cursor to 12th position
```

Question 4: Marks 20

Your task for this question is to complete the next_holiday(year,month,day) function. This function accepts three integer inputs representing a year, month, and a day and must return a two-element list containing a string with the name of the next Ontario statutory holiday and an integer representing the number of days until that holiday. The string with the name of the holiday should be all lower-case characters. The names of the holidays are given in the table below. To better understand the problem, consider the following example:

> next_holiday(2020,2,3)

In this case, the input represents the date February 3rd , 2020. The next statutory holiday after this date is Family day on February 17th. So, for this input, our function would return the following list: [‘family day’, 14] The first element in the returned list is a string with the name of the holiday. Note that all the characters in the string are lower-case. The second element in the list is the number of days until Family day. To simplify the problem, we will assume that all holidays will fall on the same date each year and we will use the 2020 dates. A list of all the statutory holidays and their corresponding date’s are given below. Condition-1: Your function should check for leap years and adjust results appropriately. A year is a leap year if it satisfies one of the following conditions:

- The year can be evenly divided by 4 and cannot be evenly divided by 100, or
- The year is evenly divisible by 400. For example, 1996 is a leap year because it satisfies condition 1, 2009 is not a leap year because it satisfies neither condition, and 2000 is a leap year because it satisfies condition 2. Condition-2: If the function is passed an invalid entry for the month or the day,

the program should return ['invalid month',-1] or ['invalid day',-1], respectively. If passed both an invalid month and an invalid day, the program should return ['invalid month',-1].

In [10]:

```python
from datetime import datetime

def is_leap_year(year):
    return (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0)

def next_holiday(year, month, day):
    holidays = {
        'new year's day': 'Jan 1',
        'family day': 'Feb 17',
        'good friday': 'Apr 10',
        'victoria day': 'May 18',
        'canada day': 'Jul 1',
        'civic holiday': 'Aug 3',
        'labor day': 'Sep 7',
        'thanksgiving day': 'Oct 12',
        'christmas day': 'Dec 25',
        'boxing day': 'Dec 26'
    }

    input_date = None

    try:
        if year > 2020 or year < 2020:
            return ['invalid year', -1]

        input_date = datetime(year, month, day)
    except ValueError:
        if month < 1 or month > 12:
            return ['invalid month', -1]
        elif day < 1 or day > 31:
            return ['invalid day', -1]

    if input_date is None:
        return ['invalid date', -1]

    for holiday, date_str in holidays.items():
        holiday_date = datetime(year, datetime.strptime(date_str, '%b %d').month, dateti
        if holiday_date >= input_date:
            days_until_holiday = (holiday_date - input_date).days
            return [holiday, days_until_holiday]


for year in range(2019, 2021):
    for month in range(1, 13):
        day = 1
        result = next_holiday(year, month, day)
        print(f"{year}-{month:02d}-{day:02d}: {result}")
```

```
2019-01-01: ['invalid year', -1]
2019-02-01: ['invalid year', -1]
2019-03-01: ['invalid year', -1]
2019-04-01: ['invalid year', -1]
2019-05-01: ['invalid year', -1]
2019-06-01: ['invalid year', -1]
2019-07-01: ['invalid year', -1]
2019-08-01: ['invalid year', -1]
2019-09-01: ['invalid year', -1]
2019-10-01: ['invalid year', -1]
2019-11-01: ['invalid year', -1]
2019-12-01: ['invalid year', -1]
2020-01-01: ['new year's day', 0]
2020-02-01: ['family day', 16]
2020-03-01: ['good friday', 40]
2020-04-01: ['good friday', 9]
2020-05-01: ['victoria day', 17]
2020-06-01: ['canada day', 30]
2020-07-01: ['canada day', 0]
2020-08-01: ['civic holiday', 2]
2020-09-01: ['labor day', 6]
2020-10-01: ['thanksgiving day', 11]
2020-11-01: ['christmas day', 54]
2020-12-01: ['christmas day', 24]
```

Assignment -1: Part-3

Question -1: Total Marks: 7

Create a dictionary 'd' using dict() function and then import it in Pandas data frame. The keys and values are as follows

'World Health Organization' -> 'Switzerland' 'UNICEF' -> 'USA' 'UNESCO'-> 'France'

Make 'Organization' column as a new index

In [11]:
```
import pandas as pd

d = dict(zip(['World Health Organization', 'UNICEF', 'UNESCO'], ['Switzerland', 'USA', '

df = pd.DataFrame(list(d.items()), columns=['Organization', 'Head Quarters'])

df.set_index('Organization', inplace=True)

df
```

Out[11]:

Organization	Head Quarters
World Health Organization	Switzerland
UNICEF	USA
UNESCO	France

Question 2: Difficulty Level-3

Each Task carries 4 Marks Total Marks: 7 * 4 -= 28

Data for this question can be found in "online-retail.xlsx" This dataset is used for all tasks in this question Assume that you are a data scientist in Amazon. Since the company is celebrating Silver Jubilee this year, it has decided to reward their customers. Your Manager handed over last 2 years retail data and asked you to do certain tasks. The tasks are as follows:

Task-1:

As you are not familiar with the data provided by your manager, you want to start exploring from InvoiceNo and Stockcode. Print the first 5 rows of InvoiceNo and Stockcode and store it in a data frame “dfnew”. The expected output is as follows:

In [12]:
```
import pandas as pd
import numpy as np

file_path = 'online-retail.xlsx'

df = pd.read_excel(file_path)

dfnew = df[['InvoiceNo', 'StockCode']].head()

dfnew['StockCode'] = dfnew['StockCode'].fillna("NaN")

dfnew
```

Out[12]:

	InvoiceNo	StockCode
0	536365	85123A
1	536365	71053
2	536365	84406B
3	536365	NaN
4	536365	NaN

Task-2: Print only non-null values from dfnew using notnull() command in python.

In [13]:
```
import pandas as pd

file_path = 'online-retail.xlsx'
df = pd.read_excel(file_path)

dfnew = df[['InvoiceNo', 'StockCode']].head()

dfnew['StockCode'] = dfnew['StockCode'].fillna("NaN")

non_null_df = dfnew[dfnew['StockCode'] != "NaN"]

non_null_df
```

Out[13]:

	InvoiceNo	StockCode
0	536365	85123A
1	536365	71053
2	536365	84406B

Task-3: Data cleaning is the crucial part thus you decided to do some data cleaning now

There is a possibility that numerical data sometimes has negative values. Thus you explored ‘Quantity’ column and found out that there are some negatives there. Count the number of negatives in ‘Quantity’ column.

In [14]:
```
import pandas as pd

file_path = 'online-retail.xlsx'
df = pd.read_excel(file_path)

dfnew = df[['InvoiceNo', 'StockCode']].head()

dfnew['StockCode'] = dfnew['StockCode'].fillna("NaN")

non_null_df = dfnew[dfnew['StockCode'] != "NaN"]

positive_count = (df['Quantity'] > 0).sum()
negative_count = (df['Quantity'] < 0).sum()

print("The number of negative values in Quantity column:")
print("False:", positive_count)
print("True:", negative_count)

quantity_column_name = 'Quantity'
quantity_column_dtype = df[quantity_column_name].dtype

print("\nName:", quantity_column_name, ", dtype:", quantity_column_dtype)
```

```
The number of negative values in Quantity column:
False: 195
True: 3

Name: Quantity , dtype: float64
```

Task-4: Now change the negative values found in ‘Quantity’ column to positive values by applying suitable python code. The final output should be the one as follows

In [15]:
```
import pandas as pd

file_path = 'online-retail.xlsx'
df = pd.read_excel(file_path)

positive_count = (df['Quantity'] > 0).sum()
negative_count = (df['Quantity'] < 0).sum()

df['Quantity'] = df['Quantity'].abs()

positive_count_updated = (df['Quantity'] > 0).sum()
negative_count_updated = (df['Quantity'] < 0).sum()

print("\nThe number of negative values in Quantity column after updating:")
print("False:", positive_count_updated)

quantity_column_name = 'Quantity'
quantity_column_dtype = df[quantity_column_name].dtype

print("\nName:", quantity_column_name, ", dtype:", quantity_column_dtype)
```

```
The number of negative values in Quantity column after updating:
False: 198

Name: Quantity , dtype: float64
```

Task-5:

To continue with the data cleaning, you want to find is there any null values in the whole data set. Apply suitable command to count the null values in the whole data set. The expected output is as follows

In [16]:
```
import pandas as pd

file_path = 'online-retail.xlsx'
df = pd.read_excel(file_path)

null_counts = df.isnull().sum()

print("Number of null values in each column:")
print(null_counts)
```

```
Number of null values in each column:
InvoiceNo      0
StockCode      5
Description    0
Quantity       1
InvoiceDate    4
UnitPrice      0
CustomerID     0
Country        0
dtype: int64
```

Task-6:

As there is only one null value in the 'Quantity' column you want to replace that null value by the average of Quantity column. Find the 'Quantity' column average and then replace the null value by that.

In [17]:
```
import pandas as pd

file_path = 'online-retail.xlsx'
df = pd.read_excel(file_path)

null_counts = df.isnull().sum()


quantity_avg = df['Quantity'].mean()

df['Quantity'].fillna(quantity_avg, inplace=True)

updated_null_counts = df['Quantity'].isnull().sum()

print("\nAverage value of 'Quantity' column is:\n", quantity_avg)
print("\nNumber of null values in 'Quantity' column after updating:\n", updated_null_cou
```

```
Average value of 'Quantity' column is:
 19.085858585858585

Number of null values in 'Quantity' column after updating:
 0
```

Task-7: Group the data based on "InvoiceNo" column. Then display the first five rows Expected result is as follows:

In [18]:
```
import pandas as pd

file_path = 'online-retail.xlsx'
df = pd.read_excel(file_path)

df = df.set_index('InvoiceNo')

df.groupby('InvoiceNo').count().reset_index()
```

```
grouped_df.head(5)
```

Out[18]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	3	7	7	7	7	7	7
1	536366	1	2	2	2	2	2	2
2	536367	12	12	12	12	12	12	12
3	536368	4	4	4	4	4	4	4
4	536369	1	1	1	1	1	1	1

In [ ]:

```
```

In [ ]:

```
```