1.Read the given three CSV les to different pandas dataframes

```
import pandas as pd
x = pd.read_csv("Customers.csv")
y = pd.read_csv("Continent.csv")
z = pd.read_csv("Transaction.csv")
```

```
x.head()
```

	customer_id	region_id	start_date	end_date
0	1	3	02-01-2020	03-01-2020
1	2	3	03-01-2020	17-01-2020
2	3	5	27-01-2020	18-02-2020
3	4	5	07-01-2020	19-01-2020
4	5	3	15-01-2020	23-01-2020

```
y.head()
```

	region_id	region_name
0	1	Australia
1	2	America
2	3	Africa
3	4	Asia
4	5	Europe

```
z.head()
```

	customer_id	txn_date	txn_type	txn_amount
0	429	21-01-2020	deposit	82
1	155	10-01-2020	deposit	712
2	398	01-01-2020	deposit	196
3	255	14-01-2020	deposit	563
4	185	29-01-2020	deposit	626

2. Merge the Customers and Transaction into a single dataframe.

```
merge_df = pd.merge(x,z)
print(merge_df.head())
```

```
   customer_id  region_id  start_date    end_date    txn_date  txn_type  \
0            1          3  02-01-2020  03-01-2020  02-01-2020   deposit
1            1          3  02-01-2020  03-01-2020  05-03-2020  purchase
2            1          3  02-01-2020  03-01-2020  17-03-2020   deposit
3            1          3  02-01-2020  03-01-2020  19-03-2020  purchase
4            1          3  04-01-2020  14-01-2020  02-01-2020   deposit

   txn_amount
0         312
1         612
2         324
3         664
4         312
```

3. Calculate the duration in days between start date and end date.

```
x['start_date'] = pd.to_datetime(x['start_date'],format='%d-%m-%Y', errors='coerce')
x['end_date'] = pd.to_datetime(x['end_date'],format='%d-%m-%Y', errors='coerce')
x['duration'] = (x['end_date']- x['start_date']).dt.days
x.head()
```

	customer_id	region_id	start_date	end_date	duration
0	1	3	2020-01-02	2020-01-03	1.0
1	2	3	2020-01-03	2020-01-17	14.0
2	3	5	2020-01-27	2020-02-18	22.0
3	4	5	2020-01-07	2020-01-19	12.0
4	5	3	2020-01-15	2020-01-23	8.0

4. Drop the duplicate rows in the merged dataframe if any.

```
merge_df = merge_df.drop_duplicates()
print(merge_df.head())
```

```
   customer_id  region_id start_date   end_date    txn_date  txn_type  \
0            1          3 2020-01-02 2020-01-03  02-01-2020   deposit
1            1          3 2020-01-02 2020-01-03  05-03-2020  purchase
2            1          3 2020-01-02 2020-01-03  17-03-2020   deposit
3            1          3 2020-01-02 2020-01-03  19-03-2020  purchase
4            1          3 2020-01-04 2020-01-14  02-01-2020   deposit

   txn_amount date_difference  date_difference_in_days
0         312          1 days                      1.0
1         612          1 days                      1.0
2         324          1 days                      1.0
3         664          1 days                      1.0
4         312         10 days                     10.0
```

5. Drop rows with missing values in the merged dataframe if any

```
merge_df = merge_df.dropna()
print(merge_df.head())
```

```
   customer_id  region_id start_date   end_date    txn_date  txn_type  \
0            1          3 2020-01-02 2020-01-03  02-01-2020   deposit
1            1          3 2020-01-02 2020-01-03  05-03-2020  purchase
2            1          3 2020-01-02 2020-01-03  17-03-2020   deposit
3            1          3 2020-01-02 2020-01-03  19-03-2020  purchase
4            1          3 2020-01-04 2020-01-14  02-01-2020   deposit

   txn_amount date_difference  date_difference_in_days
0         312          1 days                      1.0
1         612          1 days                      1.0
2         324          1 days                      1.0
3         664          1 days                      1.0
4         312         10 days                     10.0
```

6. Calculate the average duration of each customer.

```
average_duration = merge_df.groupby('customer_id')['date_difference_in_days'].mean()
print(average_duration)
```

```
customer_id
1      11.500000
2      10.833333
3      13.833333
4      13.333333
5       7.500000
          ...
496     7.333333
497    18.666667
498    11.833333
499     4.333333
500    14.000000
Name: date_difference_in_days, Length: 500, dtype: float64
```

7. Display the unique transaction types

```
unique_transactions = merge_df['txn_type'].unique()
print(unique_transactions)
```

```
['deposit' 'purchase' 'withdrawal']
```

8.Display the count of each transaction type with respect to each continent.

```
transac_counts = merge_df.groupby(['region_id', 'txn_type']).size().reset_index(name='Count')
print(transac_counts)
```

```
    region_id    txn_type  Count
0           1     deposit   3474
1           1    purchase   2280
2           1  withdrawal   2052
3           2     deposit   3480
4           2    purchase   2022
```

```
5          2  withdrawal   2046
6          3     deposit   3222
7          3    purchase   2004
8          3  withdrawal   1986
9          4     deposit   3156
10         4    purchase   1842
11         4  withdrawal   1818
12         5     deposit   2694
13         5    purchase   1554
14         5  withdrawal   1578
```

9. Find out the selling cost average for packages developed in Pascal

```
software_df = pd.read_csv('Software.csv')
print(software_df.head())
```

```
     PNAME          TITLE DEVELOPIN    SCOST  DCOST  SOLD
0     MARY         README       CPP   300.00   1200    84
1    ANAND     PARACHUTES     BASIC   399.95   6000    43
2    ANAND  VIDEO TITLING    PASCAL  7500.00  16000     9
3  JULIANA      INVENTORY     COBOL  3000.00   3500     0
4   KAMALA   PAYROLL PKG.     DBASE  9000.00  20000     7
```

```
sellavg = software_df.groupby('DEVELOPIN')['SCOST'].mean()
pascal = software_df[software_df['DEVELOPIN']=='PASCAL']
pascalavg = pascal['SCOST'].mean()
print('The average selling cost for packages developed in Pascal is:',pascalavg)
```

```
The average selling cost for packages developed in Pascal is: 3066.65
```

10. Display the names of those who have done the DAP Course.

```
stu_df = pd.read_csv('Studies.csv')
print(stu_df.head())
```

```
     PNAME INSTITUTE COURSE  COURSE FEE
0    ANAND   SABHARI  PGDCA        4500
1    ALTAF      COIT    DCA        7200
2  JULIANA      BDPS    MCA       22000
3   KAMALA  PRAGATHI    DCA        5000
4     MARY   SABHARI  PGDCA        4500
```

```
stu_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13 entries, 0 to 12
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   PNAME       13 non-null     object
 1   INSTITUTE   13 non-null     object
 2   COURSE      13 non-null     object
 3   COURSE FEE  13 non-null     int64 
dtypes: int64(1), object(3)
memory usage: 544.0+ bytes
```

```
stu_df['COURSE']
```

```
0     PGDCA
1       DCA
2       MCA
3       DCA
4     PGDCA
5       DAP
6      DCAP
7      HDCA
8     PGDCA
9      DCAP
10      DCS
11      DAP
12      DCA
Name: COURSE, dtype: object
```

```
stu_df[stu_df["COURSE"]=="DAP"]
```

	PNAME	INSTITUTE	COURSE	COURSE FEE
5	NELSON	PRAGATHI	DAP	6200
11	REVATHI	SABHARI	DAP	5000

11. Display the lowest course fee.

```
min_coursefee = stu_df['COURSE FEE'].min()
min_coursefee
```

```
4500
```

12. Display the details of the packages for which development costs have been recoverd

```
earnings = software_df['SCOST']*software_df["SOLD"]
software_df[earnings>=software_df['DCOST']]
```

```
0      25200.00
1      17197.85
2      67500.00
3          0.00
4      63000.00
5      72000.00
6     103500.00
7      25200.00
8       8250.00
9      39900.00
10     52000.00
11     43796.35
12     36975.00
13     17500.00
14      2200.00
15      5400.00
dtype: float64
```

	PNAME	TITLE	DEVELOPIN	SCOST	DCOST	SOLD
0	MARY	README	CPP	300.00	1200	84
1	ANAND	PARACHUTES	BASIC	399.95	6000	43
2	ANAND	VIDEO TITLING	PASCAL	7500.00	16000	9
4	KAMALA	PAYROLL PKG.	DBASE	9000.00	20000	7
6	MARY	CODE GENERATOR	C	4500.00	20000	23
7	PATTRICK	README	CPP	300.00	1200	84
8	QADIR	BOMBS AWAY	ASSEMBLY	750.00	3000	11
9	QADIR	VACCINES	C	1900.00	3100	21
10	RAMESH	HOTEL MGMT.	DBASE	13000.00	35000	4
11	RAMESH	DEAD LEE	PASCAL	599.95	4500	73
12	REMITHA	PC UTILITIES	C	725.00	5000	51
13	REMITHA	TSR HELP PKG.	ASSEMBLY	2500.00	6000	7
15	VIJAYA	TSR EDITOR	C	900.00	700	6

13. What is the cost of the costliest software development in Basic.

```
software_df[software_df['DEVELOPIN']=='BASIC']
software_df[software_df['DEVELOPIN']=='BASIC']['DCOST'].max()
```

```
6000
```

14. how many programmers paid 5000 to 10000 for their course.

```
prog_df = pd.read_csv("Programmer.csv")
print(prog_df)
```

```
    PNAME        DOB        DOJ GENDER     PROF1     PROF2  SALARY
0   ANAND  12-Apr-66  21-Apr-92      M    PASCAL     BASIC    3200
```

1	ALTAF	02-Jul-64	13-Nov-90	M	CLIPPER	COBOL	2800
2	JULIANA	31-Jan-60	21-Apr-90	F	COBOL	DBASE	3000
3	KAMALA	30-Oct-68	02-Jan-92	F	C	DBASE	2900
4	MARY	24-Jun-70	01-Feb-91	F	CPP	ORACLE	4500
5	NELSON	11-Sep-85	11-Oct-89	M	COBOL	DBASE	2500
6	PATTRICK	10-Nov-65	21-Apr-90	M	PASCAL	CLIPPER	2800
7	QADIR	31-Aug-65	21-Apr-91	M	ASSEMBLY	C	3000
8	RAMESH	03-May-67	28-Feb-91	M	PASCAL	DBASE	3200
9	REBECCA	01-Jan-67	01-Dec-90	F	BASIC	COBOL	2500
10	REMITHA	19-Apr-70	20-Apr-93	F	C	ASSEMBLY	3600
11	REVATHI	02-Dec-69	02-Jan-92	F	PASCAL	BASIC	3700
12	VIJAYA	14-Dec-65	02-May-92	F	FOXPRO	C	3500

```
n_programmers = prog_df[(prog_df["SALARY"]>5000) & (prog_df["SALARY"]<10000)]
print("no.of programmers paid between 5000 to 10000:",len(n_programmers))
```

```
no.of programmers paid between 5000 to 10000: 0
```

15. How many programmers know either COBOL or Pascal

```
co_pas = prog_df[(prog_df['PROF1']=='COBOL')| (prog_df['PROF2']=='COBOL') | (prog_df['PROF1']=='PASC
co_pas
print("Programmers know either cobol or pascal:", len(co_pas))
```

```
Programmers know either cobol or pascal: 8
```