```
Practical: Data integration and Data Transformation for Data Mining

```

In [2]:

```python
#Practical: Data integration and Data Transformation for Data Mining
import pandas as pd
import numpy as np

#sample datasets for Data integration

data_1 = {
    'ID': [1,2,3,4],
    'Name' : ['Alice', 'Bob', 'Charlie', 'David'],
    'Age': [25,30,35,40]
}

data_2 = {
    'ID': [3,4,5,6],
    'Gender' : ['F', 'M', 'M', 'F'],
    'Salary': [70000,80000,50000,60000]
}

df1 = pd.DataFrame(data_1)
df2 = pd.DataFrame(data_2)
```

In [3]:

```python
#data integration

#tight coup[ling (join datasets on common key)
tight_coupling= pd.merge(df1,df2, on= 'ID', how='inner')
print("Tight Counpling result:\n", tight_coupling)
```

```
Tight Counpling result:
    ID     Name  Age Gender  Salary
0   3  Charlie   35      F   70000
1   4    David   40      M   80000
```

In [5]:

```python
#loose coupling (concatenate datasets)
loose_coupling=pd.concat([df1.set_index('ID'), df2.set_index('ID')], axis=1).reset_index()
print("Loose Coupling result:\n", loose_coupling)
```

```
Loose Coupling result:
    ID     Name   Age Gender   Salary
0   1    Alice  25.0    NaN      NaN
1   2      Bob  30.0    NaN      NaN
2   3  Charlie  35.0      F  70000.0
3   4    David  40.0      M  80000.0
4   5      NaN   NaN      M  50000.0
5   6      NaN   NaN      F  60000.0
```

In [9]:

```python
#data transformation
#smoothing (moving average for age)
loose_coupling['Smoothed_age'] = loose_coupling['Age'].rolling(window=2, min_periods=1).mean()
print("\n Smoothing:\n ", loose_coupling[['ID','Age','Smoothed_age']])
```

```

 Smoothing:
     ID   Age  Smoothed_age
0   1  25.0          25.0
1   2  30.0          27.5
2   3  35.0          32.5
3   4  40.0          37.5
4   5   NaN          40.0
5   6   NaN           NaN
```

In [10]:

```python
#aggregration (summarizing salary by gender)
aggreegration = loose_coupling.groupby('Gender')['Salary'].sum().reset_index()
print("\n Aggregration: \n", aggreegration)
```

```

 Aggregration:
   Gender    Salary
0      F  130000.0
1      M  130000.0
```

In [14]:

```
#discretization (binning age into categories)
bins=[0,20,30,40,50]
labels=['Teen','Young Adult', 'Adult','Senior']
loose_coupling['Age_Group'] = pd.cut(loose_coupling['Age'], bins=bins, labels=labels)
print("\n Discretization: \n", loose_coupling[['ID', 'Age','Age_Group']])
```

```
 Discretization:
    ID   Age    Age_Group
0   1  25.0  Young Adult
1   2  30.0  Young Adult
2   3  35.0        Adult
3   4  40.0        Adult
4   5   NaN          NaN
5   6   NaN          NaN
```

In [15]:

```
#attribute constructino (creatinf age salary ratio)
loose_coupling['Age_Salary_Ratio'] = loose_coupling['Age']/ loose_coupling['Salary']
print("\n Attribute construction: \n", loose_coupling[['ID', 'Age', 'Salary', 'Age_Salary_Ratio']])
```

```
 Attribute construction:
    ID   Age   Salary  Age_Salary_Ratio
0   1  25.0      NaN               NaN
1   2  30.0      NaN               NaN
2   3  35.0  70000.0            0.0005
3   4  40.0  80000.0            0.0005
4   5   NaN  50000.0               NaN
5   6   NaN  60000.0               NaN
```

In [17]:

```
#genralization
loose_coupling['Age_MinMax'] = (loose_coupling['Age'] - loose_coupling['Age'].min()) / (loose_coupling['Age'].max() -loo
print("\n Min Max Normalizarion: \n", loose_coupling[['ID','Age', 'Age_MinMax']] )
```

```
 Min Max Normalizarion:
    ID   Age  Age_MinMax
0   1  25.0    0.000000
1   2  30.0    0.333333
2   3  35.0    0.666667
3   4  40.0    1.000000
4   5   NaN         NaN
5   6   NaN         NaN
```

In [19]:

```
#z score normaliztion
loose_coupling["Age_Zscore"] = (loose_coupling['Age'] - loose_coupling['Age'].mean()) / loose_coupling['Age'].std()
print("\n z score normalization:" , loose_coupling[['ID','Age', 'Age_Zscore']])
```

```
 z score normalization:    ID   Age  Age_Zscore
0   1  25.0   -1.161895
1   2  30.0   -0.387298
2   3  35.0    0.387298
3   4  40.0    1.161895
4   5   NaN         NaN
5   6   NaN         NaN
```

In [20]:

```
#decimal scaling
scaling_factoe = 10 **np.ceil(np.log10(loose_coupling['Age'].abs().max()))
loose_coupling['Age_Decimal_Scaling'] = loose_coupling['Age'] / scaling_factoe
print("\n Decimal scaling:\n" , loose_coupling[['ID', 'Age', 'Age_Decimal_Scaling']])
```

```
 Decimal scaling:
    ID   Age  Age_Decimal_Scaling
0   1  25.0                 0.25
1   2  30.0                 0.30
2   3  35.0                 0.35
3   4  40.0                 0.40
4   5   NaN                  NaN
5   6   NaN                  NaN
```

In [ ]:

```

```