Data Warehouse and Mining Lab

Lab - 5

```
Gyanendra Kr Shukla
191112040
CSE-1
```

In [1]:

```python
import pandas as pd
import math
```

In [2]:

```python
titanic_df = pd.read_excel('titanic.xls')
titanic_df.describe()
```

Out[2]:

	pclass	survived	age	sibsp	parch	fare
count	1309.000000	1309.000000	1046.000000	1309.000000	1309.000000	1308.000000
mean	2.294882	0.381971	29.881135	0.498854	0.385027	33.295479
std	0.837836	0.486055	14.413500	1.041658	0.865560	51.758668
min	1.000000	0.000000	0.166700	0.000000	0.000000	0.000000
25%	2.000000	0.000000	21.000000	0.000000	0.000000	7.895800
50%	3.000000	0.000000	28.000000	0.000000	0.000000	14.454200
75%	3.000000	1.000000	39.000000	1.000000	0.000000	31.275000
max	3.000000	1.000000	80.000000	8.000000	9.000000	512.329200

In [3]:

```python
# Step 1: Fill all of missing values in age and fare with mean values
mean_age = titanic_df['age'].mean()
titanic_df['age'].fillna(mean_age, inplace=True)

mean_fare = titanic_df['fare'].mean()
titanic_df['fare'].fillna(mean_fare, inplace=True)

titanic_df.describe()
```

Out[3]:

	pclass	survived	age	sibsp	parch	fare
count	1309.000000	1309.000000	1309.000000	1309.000000	1309.000000	1309.000000
mean	2.294882	0.381971	29.881135	0.498854	0.385027	33.295479
std	0.837836	0.486055	12.883199	1.041658	0.865560	51.738879
min	1.000000	0.000000	0.166700	0.000000	0.000000	0.000000
25%	2.000000	0.000000	22.000000	0.000000	0.000000	7.895800
50%	3.000000	0.000000	29.881135	0.000000	0.000000	14.454200
75%	3.000000	1.000000	35.000000	1.000000	0.000000	31.275000
max	3.000000	1.000000	80.000000	8.000000	9.000000	512.329200

1. For age and fare attribute, plot a singleton histogram and width=10 histogram

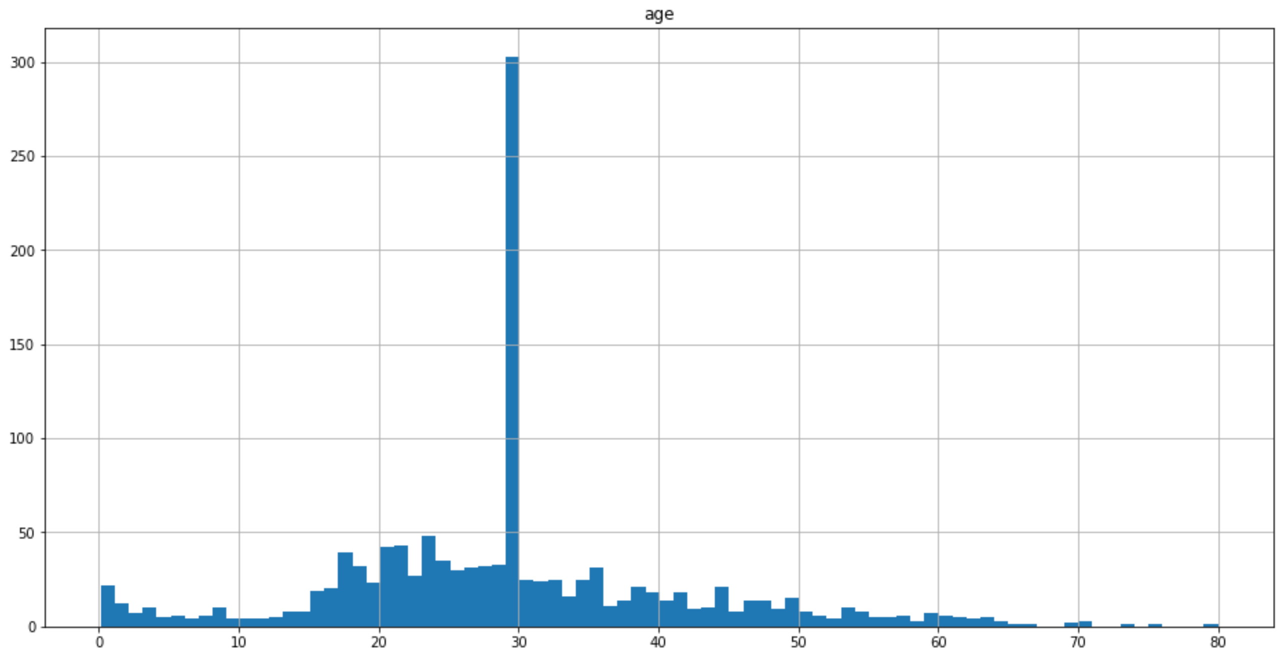
In [4]:

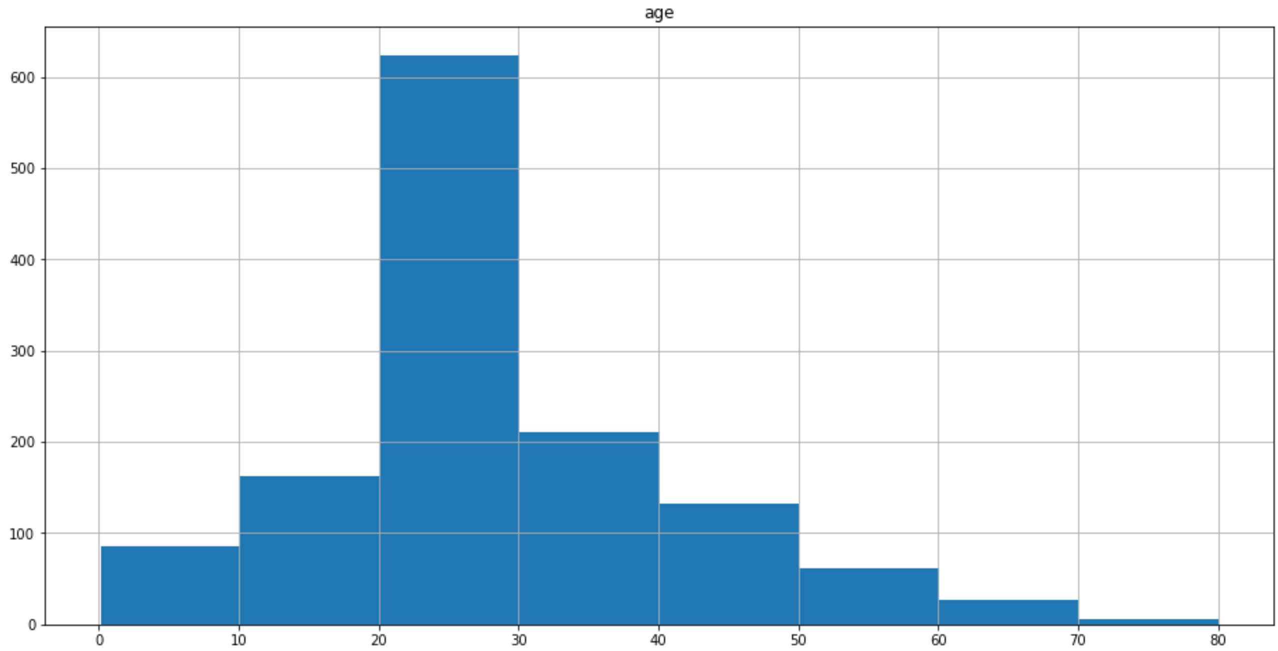
```
max_age = int(titanic_df['age'].max())
print(max_age)
titanic_df.hist(column='age', figsize=(16, 8), bins=max_age)
titanic_df.hist(column='age', figsize=(16, 8), bins=int(max_age/10))
```

```
80
```

Out[4]:

```
array([[<AxesSubplot:title={'center':'age'}>]], dtype=object)
```





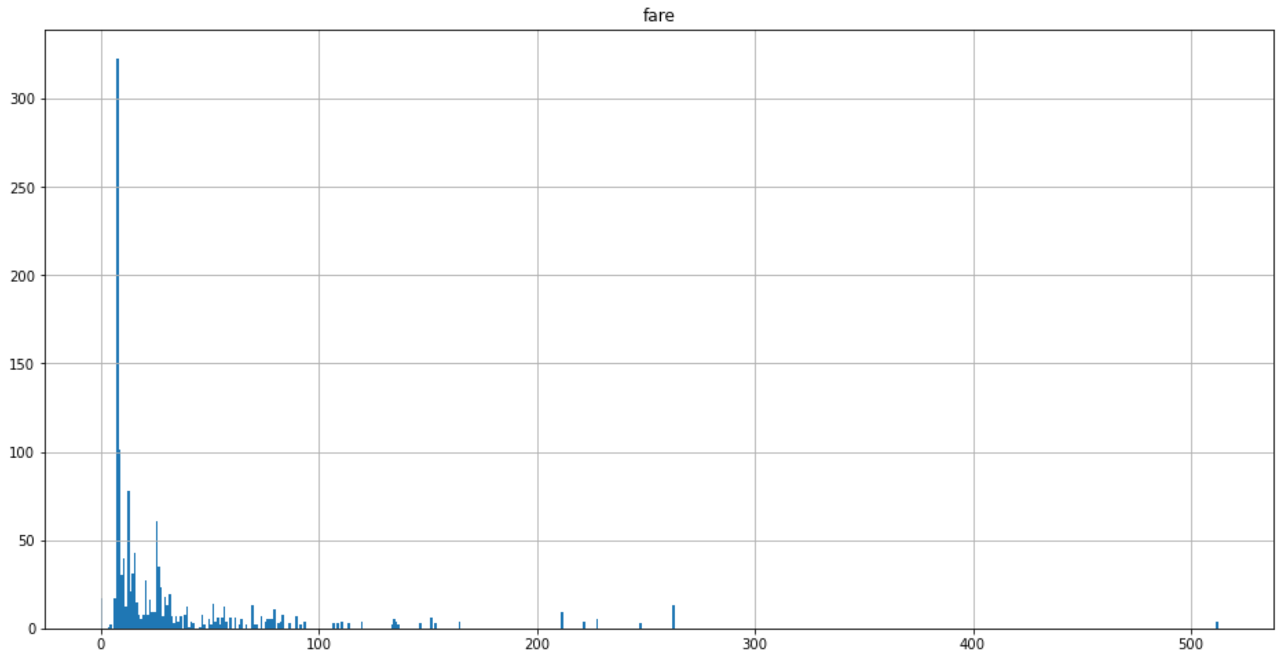
In [5]:

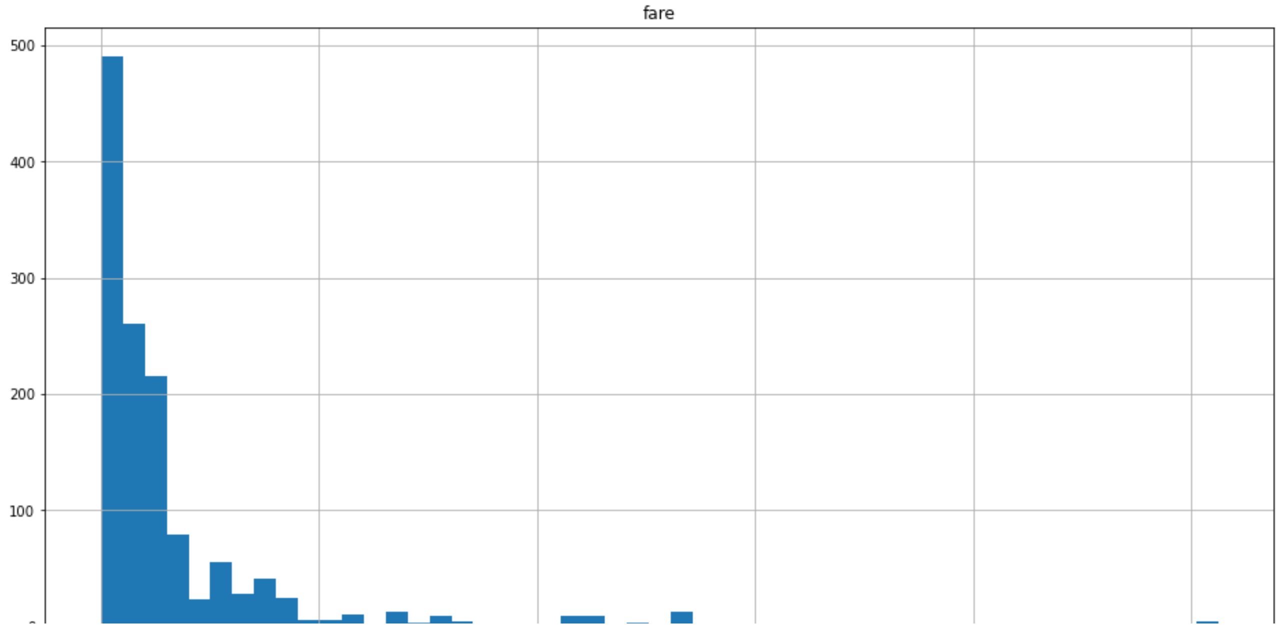
```
max_age = int(titanic_df['fare'].max())
print(max_age)
titanic_df.hist(column='fare', figsize=(16, 8), bins=max_age)
titanic_df.hist(column='fare', figsize=(16, 8), bins=int(max_age/10))
```

```
512
```

Out[5]:

```
array([[<AxesSubplot:title={'center':'fare'}>]], dtype=object)
```





2. Using the data for age attribute in the titanic dataset, wap to perform sampling techniques

select 30% samples

2.1. Simple random sampling with replacement

In [6]:

```
size_df = titanic_df['age'].size
size_sample = int(size_df*0.3)
simple_random_with_replacement = titanic_df.sample(n=size_sample, replace=True, a
simple_random_with_replacement
```

Out[6]:

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	embarked
856	3	1	Healy, Miss. Hanora "Nora"	female	29.881135	0	0	370375	7.7500	Q
714	3	1	Chip, Mr. Chang	male	32.000000	0	0	1601	56.4958	S

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	embarked
574	2	0	Turpin, Mr. William John Robert	male	29.000000	1	0	11668	21.0000	S
522	2	0	Otter, Mr. Richard	male	39.000000	0	0	28213	13.0000	S
135	1	0	Goldschmidt, Mr. George B	male	71.000000	0	0	PC 17754	34.6542	C
...	...	...	...	...	...	...	...	...	...	...
1119	3	0	Perkin, Mr. John Henry	male	22.000000	0	0	A/5 21174	7.2500	S
322	1	1	Young, Miss. Marie Grice	female	36.000000	0	0	PC 17760	135.6333	C
730	3	0	Cor, Mr. Ivan	male	27.000000	0	0	349229	7.8958	S
468	2	0	Karnes, Mrs. J Frank (Claire Bennett)	female	22.000000	0	0	F.C.C. 13534	21.0000	S
1068	3	0	Nysveen, Mr. Johan Hansen	male	61.000000	0	0	345364	6.2375	S

2.2 Simple random sampling without replacement

In [7]:

```
size_df = titanic_df['age'].size
size_sample = int(size_df*0.3)
simple_random_without_replacement = titanic_df.sample(n=size_sample, replace=Fals
simple_random_without_replacement
```

Out[7]:

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	embarked
796	3	0	Everett, Mr. Thomas James	male	40.500000	0	0	C.A. 6212	15.1000	S
765	3	1	Dean, Mrs. Bertram (Eva Georgetta Light)	female	33.000000	1	2	C.A. 2315	20.5750	S
1273	3	0	Vander Planke, Miss. Augusta Maria	female	18.000000	2	0	345764	18.0000	S
1003	3	1	McCoy, Mr. Bernard	male	29.881135	2	0	367226	23.2500	Q
521	2	1	Nye, Mrs. (Elizabeth Ramell)	female	29.000000	0	0	C.A. 29395	10.5000	S
...	...	...	...	...	...	...	...	...	...	...
808	3	0	Ford, Mr. Arthur	male	29.881135	0	0	A/5 1478	8.0500	S
905	3	1	Jonsson, Mr. Carl	male	32.000000	0	0	350417	7.8542	S

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	embarked
285	1	0	Straus, Mr. Isidor	male	67.000000	1	0	PC 17483	221.7792	S
1303	3	0	Yousseff, Mr. Gerious	male	29.881135	0	0	2627	14.4583	C
1203	3	0	Sivic, Mr.	male	40.000000	0	0	349251	7.8958	S

2.3 Stratified Sampling

In [8]:
```
size_df = titanic_df['age'].size
size_sample = int(size_df*0.3)
stratified_sample = titanic_df.groupby('age', group_keys=False).apply(lambda x: x
stratified_sample.describe()
```

Out[8]:

	pclass	survived	age	sibsp	parch	fare
count	372.000000	372.000000	372.000000	372.000000	372.000000	372.000000
mean	2.145161	0.408602	32.594281	0.586022	0.551075	40.775705
std	0.887410	0.492238	19.304962	1.096733	0.896278	61.585479
min	1.000000	0.000000	0.166700	0.000000	0.000000	0.000000
25%	1.000000	0.000000	17.000000	0.000000	0.000000	8.342725
50%	2.000000	0.000000	32.000000	0.000000	0.000000	20.550000
75%	3.000000	1.000000	48.000000	1.000000	1.000000	39.471875
max	3.000000	1.000000	80.000000	8.000000	5.000000	512.329200

2.4 Comparing mean and standard deviation of the sample with the population

In [9]:
```
def mean_and_sd(df, col, ddof=1):
    mean = df[col].mean()
    sd = df[col].std(ddof=ddof)
    return (mean, sd)

print(f"Dataframe                                  Mean                SD")
print(f"Stratified Sampling:                      {mean_and_sd(stratified_sampl
print(f"Simple Random Sampling without Replacement: {mean_and_sd(simple_random_wi
print(f"Simple Random Sampling with    Replacement: {mean_and_sd(simple_random_wi
print(f"Original Population:                      {mean_and_sd(titanic_df, 'age
```

```
Dataframe                                  Mean                SD
Stratified Sampling:                      (32.59428057140361, 19.3049620374984
6)
Simple Random Sampling without Replacement: (35.63815019852114, 12.93350566060733
4)
Simple Random Sampling with    Replacement: (34.72598236844932, 12.38233883878507
6)
Original Population:                      (29.881134512428055, 12.8782770952070
78)
```

3. Normalizing Data

3.1 wap for min max normalization onto range [0,1]

In [10]:

```
def min_max_normalize(df, col):
    normalized_df = (df[col] - df[col].min()) / (df[col].max() - df[col].min())
    return normalized_df

age_titanic_df_minmax = min_max_normalize(titanic_df, 'age')
print(age_titanic_df_minmax.describe())

print("\n")

fare_titanic_df_minmax = min_max_normalize(titanic_df, 'fare')
print(fare_titanic_df_minmax.describe())
```

```
count    1309.000000
mean        0.372206
std         0.161376
min         0.000000
25%         0.273486
50%         0.372206
75%         0.436325
max         1.000000
Name: age, dtype: float64


count    1309.000000
mean        0.064988
std         0.100988
min         0.000000
25%         0.015412
50%         0.028213
75%         0.061045
max         1.000000
Name: fare, dtype: float64
```

3.2 Write a program for z-score normalization

In [11]:

```
def z_score_normalize(df, col):
    normalized_df = (df[col] - df[col].mean()) / df[col].std(ddof=1)
    return normalized_df

age_titanic_df_zscore = z_score_normalize(titanic_df, 'age')
print(age_titanic_df_zscore.describe())

print("\n")

fare_titanic_df_zscore = z_score_normalize(titanic_df, 'fare')
print(fare_titanic_df_zscore.describe())
```

```
count    1.309000e+03
mean     1.862123e-14
std      1.000000e+00
min     -2.306448e+00
25%     -6.117374e-01
50%      1.902767e-14
75%      3.973288e-01
max      3.890250e+00
Name: age, dtype: float64


count    1.309000e+03
mean    -4.927940e-15
std      1.000000e+00
min     -6.435292e-01
25%     -4.909206e-01
50%     -3.641609e-01
75%     -3.905147e-02
max      9.258680e+00
```

```
Name: fare, dtype: float64
```

3.3 Write a program for decimal scaling

In [12]:

```
def decimal_scaling(df, col):
    max_val = df[col].max()
    digits = math.floor(math.log10(max_val)) + 1
    scaled_df = df[col]/digits
    return scaled_df

age_titanic_df_decimal = decimal_scaling(titanic_df, 'age')
print(age_titanic_df_decimal.describe())

print("\n")

fare_titanic_df_decimal = decimal_scaling(titanic_df, 'fare')
print(fare_titanic_df_decimal.describe())
```

```
count    1309.000000
mean       14.940567
std         6.441600
min         0.083350
25%        11.000000
50%        14.940567
75%        17.500000
max        40.000000
Name: age, dtype: float64


count    1309.000000
mean       11.098493
std        17.246293
min         0.000000
25%         2.631933
50%         4.818067
75%        10.425000
max       170.776400
Name: fare, dtype: float64
```

3.4 Comparing the mean and std of original data with normalized data

In [13]:

```
print(f"Dataframe [Age]                   Mean                SD")
print(f"Original Population (Age):      {mean_and_sd(titanic_df, 'age', 0)}")
print(f"Min Max Normalization:          ({age_titanic_df_minmax.mean()}, {age_titanic
print(f"Z Score Normalization:          ({age_titanic_df_zscore.mean()}, {age_titanic
print(f"Decimal Scaling:                ({age_titanic_df_decimal.mean()}, {age_titani
```

```
Dataframe [Age]                Mean                SD
Original Population (Age):   (29.881134512428055, 12.878277095207078)
Min Max Normalization:       (0.37220601569054873, 0.16131460299407732)
Z Score Normalization:       (1.86212347638089e-14, 0.9996179560510038)
Decimal Scaling:             (14.940567256214027, 6.439138547603539)
```

In [14]:

```
print(f"Dataframe [Fare]                   Mean                SD")
print(f"Original Population (Fare):  {mean_and_sd(titanic_df, 'age', 0)}")
print(f"Min Max Normalization:          ({fare_titanic_df_minmax.mean()}, {fare_tita
print(f"Z Score Normalization:          ({fare_titanic_df_zscore.mean()}, {fare_tita
print(f"Decimal Scaling:                ({fare_titanic_df_decimal.mean()}, {fare_tit
```

```
Dataframe [Fare]                Mean                SD
Original Population (Fare):   (29.881134512428055, 12.878277095207078)
Min Max Normalization:        (0.06498844743056884, 0.10094898457243737)
Z Score Normalization:        (-4.927939899792449e-15, 0.9996179560510065)
Decimal Scaling:              (11.098493093781844, 17.239704168936406)
```