Data Warehouse and Mining Lab

Lab - 6

```
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

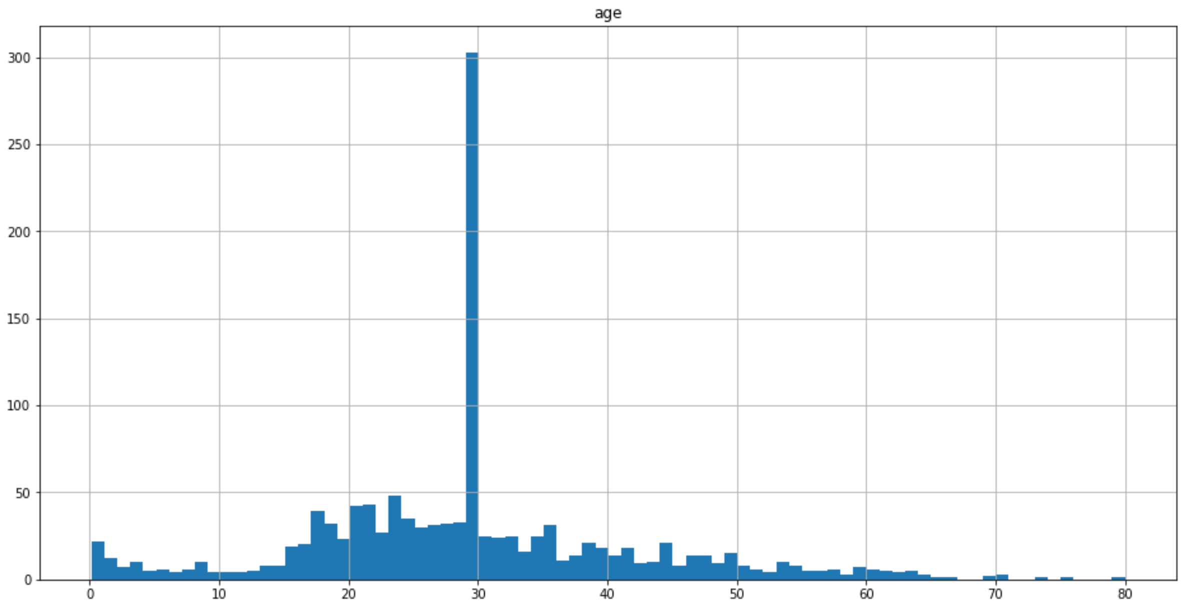
In [4]:

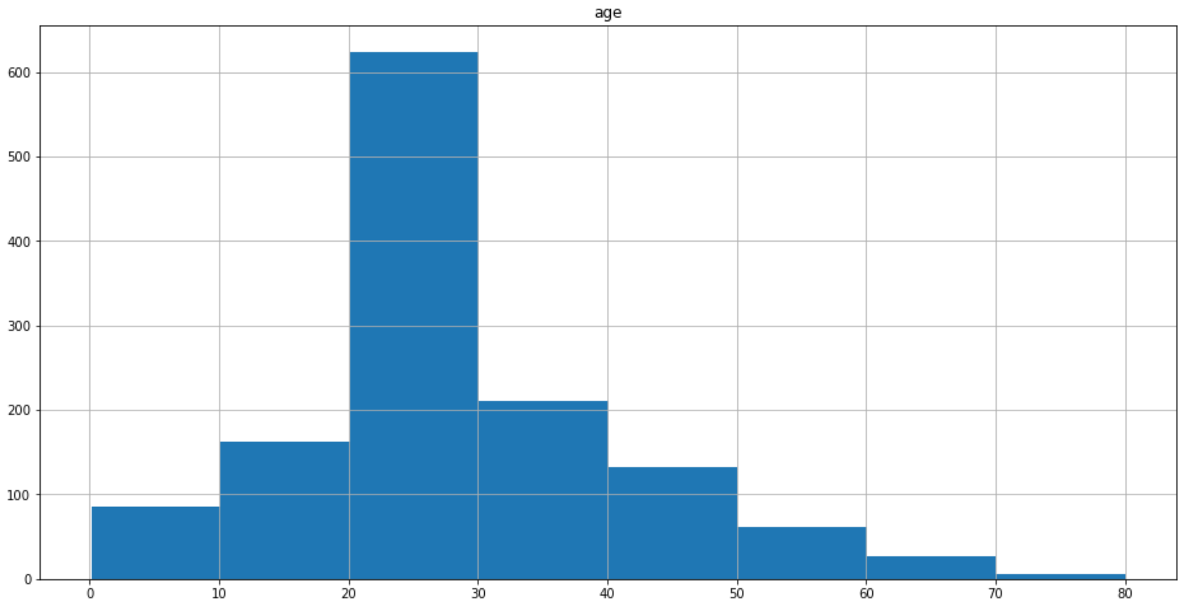
```
max_age = int(titanic_df['age'].max())
print(max_age)
titanic_df.hist(column='age', figsize=(16, 8), bins=max_age)
titanic_df.hist(column='age', figsize=(16, 8), bins=int(max_age/10))
```

```
80
```

Out[4]:

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





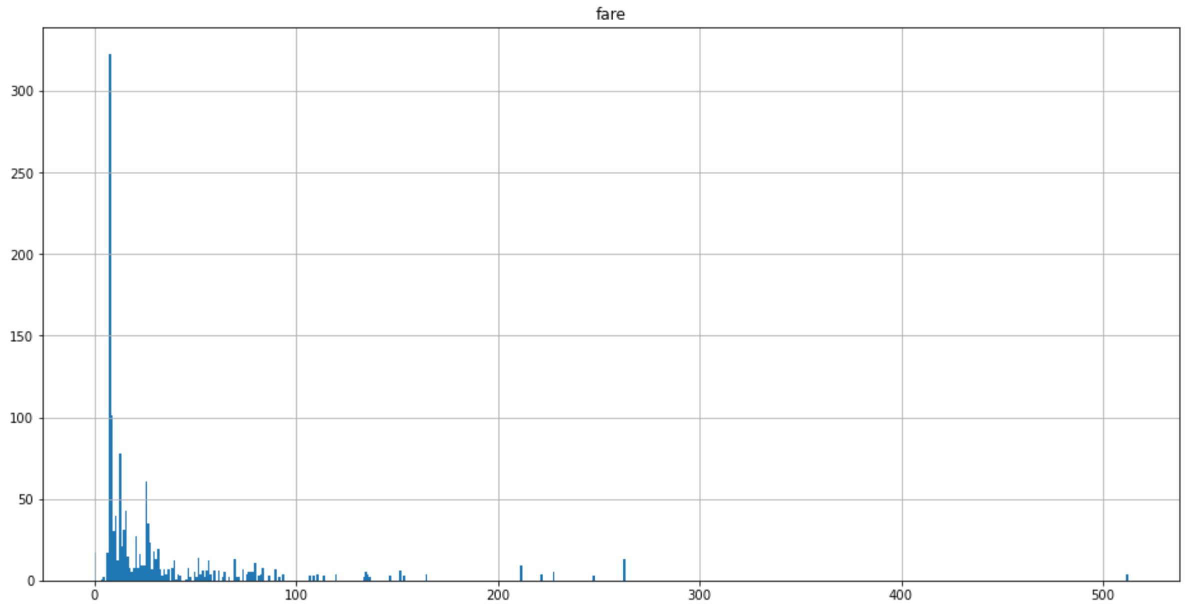
In [5]:

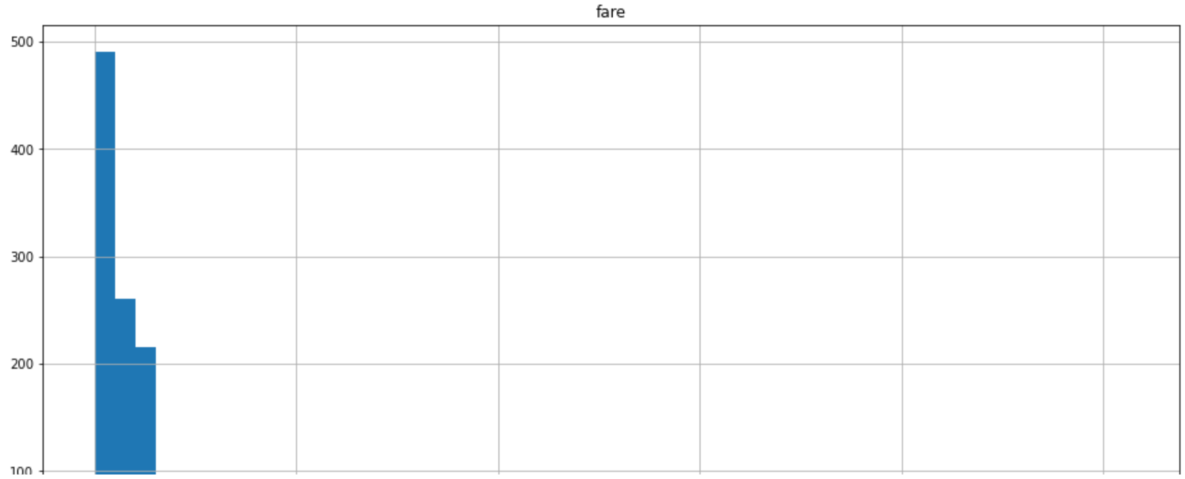
```
max_age = int(titanic_df['fare'].max())
print(max_age)
titanic_df.hist(column='fare', figsize=(16, 8), bins=max_age)
titanic_df.hist(column='fare', figsize=(16, 8), bins=int(max_age/10))
```

```
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=T
simple_random_with_replacement
```

Out[6]:

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	embarked
118	1	0	Franklin, Mr. Thomas Parham	male	29.881135	0	0	113778	26.5500	S
757	3	1	Davison, Mrs. Thomas Henry (Mary E Finck)	female	29.881135	1	0	386525	16.1000	S

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	embarked
170	1	1	Ismay, Mr. Joseph Bruce	male	49.000000	0	0	112058	0.0000	S
306	1	0	White, Mr. Percival Wayland	male	54.000000	0	1	35281	77.2875	S
1039	3	1	Mullens, Miss. Katherine "Katie"	female	29.881135	0	0	35852	7.7333	Q
...	...	...	...	...	...	...	...	...	...	...
1028	3	1	Moran, Miss. Bertha	female	29.881135	1	0	371110	24.1500	Q
1158	3	0	Rosblom, Mrs. Viktor (Helena Wilhelmina)	female	41.000000	0	2	370129	20.2125	S
238	1	1	Robert, Mrs. Edward Scott (Elisabeth Walton Mc...	female	43.000000	0	1	24160	211.3375	S
281	1	1	Stengel, Mrs. Charles Emil Henry (Annie May Mo...	female	43.000000	1	0	11778	55.4417	C
171	1	0	Jones, Mr. Charles	male	46.000000	0	0	694	26.0000	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, replac
simple_random_without_replacement
```

Out[7]:

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	embar
693	3	1	Buckley, Mr. Daniel	male	21.000000	0	0	330920	7.8208	
1211	3	0	Skoog, Mrs. William (Anna Bernhardina Karlsson)	female	45.000000	1	4	347088	27.9000	
1291	3	0	Willer, Mr. Aaron ("Abi Weller")	male	29.881135	0	0	3410	8.7125	

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	embar
546	2	1	Reynaldo, Ms. Encarnacion	female	28.000000	0	0	230434	13.0000	
78	1	1	Compton, Mrs. Alexander Taylor (Mary Eliza Ing...	female	64.000000	0	2	PC 17756	83.1583	
...	...	...	...	...	...	...	...	...	...	
598	2	1	Wright, Miss. Marion	female	26.000000	0	0	220844	13.5000	
634	3	0	Angheloff, Mr. Minko	male	26.000000	0	0	349202	7.8958	
822	3	0	Goldsmith, Mr. Nathan	male	41.000000	0	0	SOTON/O.Q. 3101263	7.8500	
1243	3	0	Thomas, Mr. Tannous	male	29.881135	0	0	2684	7.2250	
793	3	0	Elsbury, Mr. William James	male	47.000000	0	0	A/5 3902	7.2500	

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(lambc
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	2.201613	0.413978	32.594281	0.637097	0.545699	37.365882
std	0.846555	0.493208	19.304962	1.098894	0.902562	57.636342
min	1.000000	0.000000	0.166700	0.000000	0.000000	0.000000
25%	1.000000	0.000000	17.000000	0.000000	0.000000	8.662500
50%	2.000000	0.000000	32.000000	0.000000	0.000000	20.231250
75%	3.000000	1.000000	48.000000	1.000000	1.000000	34.865650
max	3.000000	1.000000	80.000000	8.000000	6.000000	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
print(f"Stratified Sampling:                        {mean_and_sd(stratifiec
print(f"Simple Random Sampling without Replacement: {mean_and_sd(simple_rar
print(f"Simple Random Sampling with    Replacement: {mean_and_sd(simple_rar
print(f"Original Population:                        {mean_and_sd(titanic_df
```

```
Dataframe                                       Mean                SD
Stratified Sampling:                        (32.59428057140361, 19.30496203
749846)
Simple Random Sampling without Replacement: (35.3145990291002, 13.299809238
865967)
Simple Random Sampling with    Replacement: (34.08701435327003, 12.30962665
2263514)
Original Population:                        (29.881134512428055, 12.8782770
95207078)
```

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].mi
    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_t
print(f"Z Score Normalization:      ({age_titanic_df_zscore.mean()}, {age_t
print(f"Decimal Scaling:            ({age_titanic_df_decimal.mean()}, {age_
```

```
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()}, {far
print(f"Z Score Normalization:       ({fare_titanic_df_zscore.mean()}, {far
print(f"Decimal Scaling:             ({fare_titanic_df_decimal.mean()}, {fa
```

```
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)
```