DATA MINING

ASSESSMENT 01

ABHIRUPA MITRA
17BCE0437

QUESTION:

Consider the Sales Transaction Dataset given below in this link in CSV format.

Perform the following tasks:-

a) Read and clean the data first
b) Try removing all the rows from the dataset that contain missing values (0) for the product transaction per week. How many of the dataset are left? Find out what percent of the dataset is missing? Print the values.
c) Use Min-Max Normalization to normalize the transaction data for all the weeks
d) Use the appropriate plot to display the highest sold product (ID) in the first 3 weeks.

In [43]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
header = ['Product_code','w0','w1','w2','w3','w4','w5','w6','w7','w8','w9','w10','w11','w12','w13','w14','w15','w16','w
df = pd.read_csv("Sales_Transactions_Dataset_Weekly.csv", header=None, names=header)
```

In [44]:

```
print(df.head(10))
row=df.shape[0]
```

```
   Product_code  w0  w1  w2  w3  w4  w5  w6  w7  w8 ...   w44  w45  w46  w47  \
0  Product_Code  W0  W1  W2  W3  W4  W5  W6  W7  W8 ...   W44  W45  W46  W47
1            P1  11  12  10   8  13  12  14  21   6 ...     8   10   12    3
2            P2   7   6   3   2   7   1   6   3   3 ...     5    1    1    4
3            P3   7  11   8   9  10   8   7  13  12 ...     5    5    7    8
4            P4  12   8  13   5   9   6   9  13  13 ...     3    4    6    8
5            P5   8   5  13  11   6   7   9  14   9 ...     7   12    6    6
6            P6   3   3   2   7   6   3   8   6   6 ...     4    3    6    5
7            P7   4   8   3   7   8   7   2   3  10 ...     3    6    2    6
8            P8   8   6  10   9   6   8   7   5  10 ...     4    8    8    6
9            P9  14   9  10   7  11  15  12   7  13 ...    13    3    7    7

   w48  w49  w50  w51  min  max
0  W48  W49  W50  W51  MIN  MAX
1    7    6    5   10    3   21
2    5    1    6    0    0   10
3   14    8    8    7    3   14
4   14    8    7    8    2   19
5    5   11    8    9    3   18
6    3    3   10    6    0   11
7    2    4    2    1    0   10
8    7    4    9    9    3   15
9   10   12    7   13    3   18

[10 rows x 55 columns]
```

In [45]:

```
df.describe()
# df.dtypes
```

Out[45]:

	Product_code	w0	w1	w2	w3	w4	w5	w6	w7	w8	...	w44	w45	w46	w47	w48	w49	w50	w51	min	max
count	812	812	812	812	812	812	812	812	812	812	...	812	812	812	812	812	812	812	812	812	812
unique	812	51	50	55	54	55	54	54	55	54	...	47	47	51	48	49	46	45	43	26	63
top	P436	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	2
freq	1	256	259	258	260	247	250	248	262	251	...	205	190	184	204	176	178	167	154	445	100

4 rows × 55 columns

In [46]:

```
df = df.iloc[1:]
df[['w0','w1','w2','w3','w4','w5','w6','w7','w8','w9','w10','w11','w12','w13','w14','w15','w16','w17','w18','w19','w20'
print(df)
```

```
    Product_code  w0  w1  w2  w3  w4  w5  w6  w7  w8 ...   w44  w45  w46  w47  \
1             P1  11  12  10   8  13  12  14  21   6 ...     8   10   12    3
2             P2   7   6   3   2   7   1   6   3   3 ...     5    1    1    4
3             P3   7  11   8   9  10   8   7  13  12 ...     5    5    7    8
4             P4  12   8  13   5   9   6   9  13  13 ...     3    4    6    8
5             P5   8   5  13  11   6   7   9  14   9 ...     7   12    6    6
6             P6   3   3   2   7   6   3   8   6   6 ...     4    3    6    5
7             P7   4   8   3   7   8   7   2   3  10 ...     3    6    2    6
8             P8   8   6  10   9   6   8   7   5  10 ...     4    8    8    6
9             P9  14   9  10   7  11  15  12   7  13 ...    13    3    7    7
10           P10  22  19  19  29  20  16  26  20  24 ...    11   24   13   16
11           P11  15   7  15  14  17   7  10  16  11 ...    12   11   13    8
12           P12   3   4   1   6   4   3   7   3   5 ...     4    5    1    3
13           P13  12  10   9   6  10  11  18   8  10 ...     5    8    7   19
14           P14  14  12   9  11  13  12   8  12  13 ...     3    9   15    8
15           P15  19  45  47  42  29  44  43  36  25 ...    33   30   37   30
16           P16  30  27  27  43  29  32  49  41  49 ...    33   39   42   45
17           P17  49  40  40  28  40  47  44  45  39 ...    36   35   43   28
18           P18  40  38  39  38  39  33  28  44  36 ...    29   41   35   22
```

In [48]:
```
df=df[(df != 0).all(1)]
```

In [50]:
```
rowN=df.shape[0]
# rowN
print("Total number of rows:", rowN)
print("Percentage of dataset missing:", (rowN/812)*100,"%")
```

```
Total number of rows: 367
Percentage of dataset missing: 45.19704433497537 %
```

In [90]:
```
# Min Max normalization using libraries from sklearn
from sklearn import preprocessing
df_n=df[['Product_code']]
df_n=df_n.reset_index(drop=True)
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled= min_max_scaler.fit_transform(df[['w0','w1','w2','w3','w4','w5','w6','w7','w8','w9','w10','w11','w12','w13','w
df_m = pd.DataFrame(x_scaled)
print(df_m)
print(df_n)
# frames=[df_n, df_m]
result = pd.concat([df_n, df_m], axis=1, sort=False)
result.head(200)
```

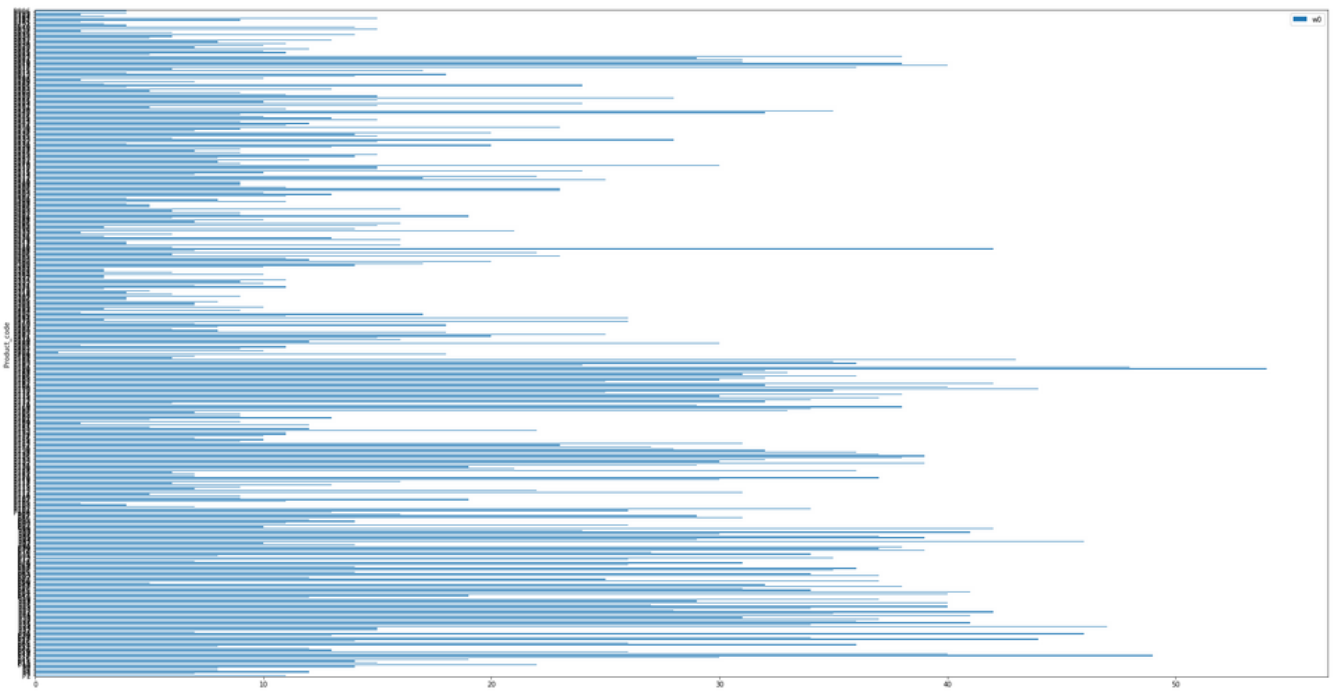
	0	1	2	3	4	5	6	\
0	0.188679	0.211538	0.163636	0.120690	0.200000	0.215686	0.236364	
1	0.113208	0.192308	0.127273	0.137931	0.150000	0.137255	0.109091	
2	0.207547	0.134615	0.218182	0.068966	0.133333	0.098039	0.145455	
3	0.132075	0.076923	0.218182	0.172414	0.083333	0.117647	0.145455	
4	0.132075	0.096154	0.163636	0.137931	0.083333	0.137255	0.109091	
5	0.245283	0.153846	0.163636	0.103448	0.166667	0.274510	0.200000	
6	0.396226	0.346154	0.327273	0.482759	0.316667	0.294118	0.454545	
7	0.264151	0.115385	0.254545	0.224138	0.266667	0.117647	0.163636	
8	0.245283	0.211538	0.145455	0.172414	0.200000	0.215686	0.127273	
9	0.339623	0.846154	0.836364	0.706897	0.466667	0.843137	0.763636	
10	0.547170	0.500000	0.472727	0.724138	0.466667	0.607843	0.872727	
11	0.905660	0.750000	0.709091	0.465517	0.650000	0.901961	0.781818	
12	0.735849	0.711538	0.690909	0.637931	0.633333	0.627451	0.490909	
13	0.471698	0.576923	0.800000	0.603448	0.500000	0.529412	0.490909	
14	0.226415	0.307692	0.181818	0.155172	0.100000	0.196078	0.290909	
15	0.207547	0.076923	0.145455	0.120690	0.133333	0.098039	0.163636	
16	0.132075	0.250000	0.127273	0.137931	0.266667	0.098039	0.290909	
17	0.660377	0.788462	0.472727	0.551724	0.650000	0.921569	0.672727	

In [96]:
```
import matplotlib.pyplot as plt
# fig, axes = plt.subplots(3, 4, figsize=(14, 8))
df.plot(x="Product_code", y=["w0"], kind="barh",figsize=(34, 18))
# df.plot.scatter(x="w0", y="Product_code");
m0=result[0].argmax()
max0=result.iloc[[m0]]['Product_code']
print("Product Code of item sold the most", max0)

```

```
/home/abhirupa/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:5: FutureWarning: 'argmax' is deprecated, us
e 'idxmax' instead. The behavior of 'argmax'
will be corrected to return the positional maximum in the future.
Use 'series.values.argmax' to get the position of the maximum now.
  """
```

```
Product Code of item sold the most 169    P190
Name: Product_code, dtype: object
```



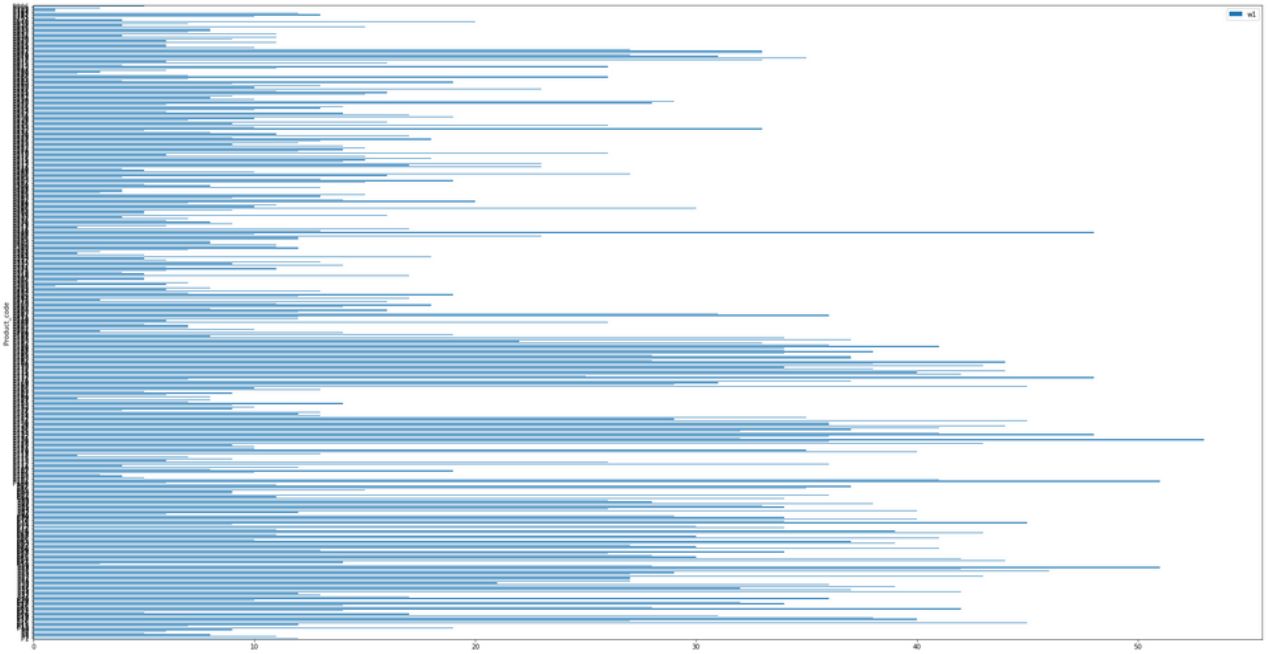
In [11]:

```
import matplotlib.pyplot as plt
# fig, axes = plt.subplots(3, 4, figsize=(14, 8))
df.plot(x="Product_code", y=["w1"], kind="barh",figsize=(34, 18))
# df.plot.scatter(x="w0", y="Product_code");
m0=result[1].argmax()
max0=result.iloc[[m0]]['Product_code']
print("Product Code of item sold the most", max0)

```

```
/home/abhirupa/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:5: FutureWarning: 'argmax' is deprecated, us
e 'idxmax' instead. The behavior of 'argmax'
will be corrected to return the positional maximum in the future.
Use 'series.values.argmax' to get the position of the maximum now.
  """

Product Code of item sold the most 115    P130
Name: Product_code, dtype: object
```



In [12]:

```
import matplotlib.pyplot as plt
# fig, axes = plt.subplots(3, 4, figsize=(14, 8))
df.plot(x="Product_code", y=["w2"], kind="barh",figsize=(34, 18))
# df.plot.scatter(x="w0", y="Product_code");
m0=result[2].argmax()
max0=result.iloc[[m0]]['Product_code']
print("Product Code of item sold the most", max0)

```

```
/home/abhirupa/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:5: FutureWarning: 'argmax' is deprecated, us
e 'idxmax' instead. The behavior of 'argmax'
will be corrected to return the positional maximum in the future.
Use 'series.values.argmax' to get the position of the maximum now.
  """

Product Code of item sold the most 83    P92
Name: Product_code, dtype: object
```