In [1]:

```
import pandas as pd
import numpy as np
```

In [3]:

```
data = {
    "Name": ["Amit", "Priya", "Raj", "Sneha", "Vikram", "Ananya", "Rohan"
    "Gender": ["Male", "Female", "Male", "Female", "Male", "Female", "Mal
    "Marks": [85, 80, 78, 'Nan', 76, 82, 'Nan'],
    "Age": [20,21,22,23,24,25,26]
}
df = pd.DataFrame(data)
print(df)
```

```
     Name  Gender Marks  Age
0    Amit    Male    85   20
1   Priya  Female    80   21
2     Raj    Male    78   22
3   Sneha  Female   Nan   23
4  Vikram    Male    76   24
5  Ananya  Female    82   25
6   Rohan    Male   Nan   26
```

In [5]:

```
df.mean()
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
Cell In[5], line 1
----> 1 df.mean()

File ~\anaconda3\Lib\site-packages\pandas\core\frame.py:11693, in DataFram
e.mean(self, axis, skipna, numeric_only, **kwargs)
  11685 @doc(make_doc("mean", ndim=2))
  11686 def mean(
  11687     self,
   (...)
  11691     **kwargs,
  11692 ):
> 11693     result = super().mean(axis, skipna, numeric_only, **kwargs)
  11694     if isinstance(result, Series):
  11695         result = result.__finalize__(self, method="mean")

File ~\anaconda3\Lib\site-packages\pandas\core\generic.py:12420, in NDFram
e.mean(self, axis, skipna, numeric_only, **kwargs)
  12413 def mean(
  12414     self,
  12415     axis: Axis | None = 0,
   (...)
  12418     **kwargs,
  12419 ) -> Series | float:
> 12420     return self._stat_function(
  12421         "mean", nanops.nanmean, axis, skipna, numeric_only, **kwar
gs
  12422     )

File ~\anaconda3\Lib\site-packages\pandas\core\generic.py:12377, in NDFram
e._stat_function(self, name, func, axis, skipna, numeric_only, **kwargs)
  12373 nv.validate_func(name, (), kwargs)
  12375 validate_bool_kwarg(skipna, "skipna", none_allowed=False)
> 12377 return self._reduce(
  12378     func, name=name, axis=axis, skipna=skipna, numeric_only=numeri
c_only
  12379 )

File ~\anaconda3\Lib\site-packages\pandas\core\frame.py:11562, in DataFram
e._reduce(self, op, name, axis, skipna, numeric_only, filter_type, **kwds)
  11558     df = df.T
  11560 # After possibly _get_data and transposing, we are now in the
  11561 #  simple case where we can use BlockManager.reduce
> 11562 res = df._mgr.reduce(blk_func)
  11563 out = df._constructor_from_mgr(res, axes=res.axes).iloc[0]
  11564 if out_dtype is not None and out.dtype != "boolean":

File ~\anaconda3\Lib\site-packages\pandas\core\internals\managers.py:1500,
in BlockManager.reduce(self, func)
   1498 res_blocks: list[Block] = []
   1499 for blk in self.blocks:
-> 1500     nbs = blk.reduce(func)
   1501     res_blocks.extend(nbs)
   1503 index = Index([None])  # placeholder

File ~\anaconda3\Lib\site-packages\pandas\core\internals\blocks.py:404, in
Block.reduce(self, func)
```

```
    398 @final
    399 def reduce(self, func) -> list[Block]:
    400     # We will apply the function and reshape the result into a single-row
    401     #  Block with the same mgr_locs; squeezing will be done at a higher level
    402     assert self.ndim == 2
--> 404     result = func(self.values)
    406     if self.values.ndim == 1:
    407         res_values = result

File ~\anaconda3\Lib\site-packages\pandas\core\frame.py:11481, in DataFrame._reduce.<locals>.blk_func(values, axis)
  11479         return np.array([result])
  11480 else:
> 11481     return op(values, axis=axis, skipna=skipna, **kwds)

File ~\anaconda3\Lib\site-packages\pandas\core\nanops.py:147, in bottleneck_switch.__call__.<locals>.f(values, axis, skipna, **kwds)
    145         result = alt(values, axis=axis, skipna=skipna, **kwds)
    146 else:
--> 147     result = alt(values, axis=axis, skipna=skipna, **kwds)
    149 return result

File ~\anaconda3\Lib\site-packages\pandas\core\nanops.py:404, in _datetimelike_compat.<locals>.new_func(values, axis, skipna, mask, **kwargs)
    401 if datetimelike and mask is None:
    402     mask = isna(values)
--> 404 result = func(values, axis=axis, skipna=skipna, mask=mask, **kwargs)
    406 if datetimelike:
    407     result = _wrap_results(result, orig_values.dtype, fill_value=iNaT)

File ~\anaconda3\Lib\site-packages\pandas\core\nanops.py:719, in nanmean(values, axis, skipna, mask)
    716     dtype_count = dtype
    718 count = _get_counts(values.shape, mask, axis, dtype=dtype_count)
--> 719 the_sum = values.sum(axis, dtype=dtype_sum)
    720 the_sum = _ensure_numeric(the_sum)
    722 if axis is not None and getattr(the_sum, "ndim", False):

File ~\anaconda3\Lib\site-packages\numpy\core\_methods.py:49, in _sum(a, axis, dtype, out, keepdims, initial, where)
     47 def _sum(a, axis=None, dtype=None, out=None, keepdims=False,
     48          initial=_NoValue, where=True):
---> 49     return umr_sum(a, axis, dtype, out, keepdims, initial, where)

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

In [3]:

```
cat=[]
con=[]
for i in df.columns:
    if(df[i].dtypes=="object"):
        cat.append(i)
else:
    con.append(i)
df
```

Out[3]:

	Name	Gender	Marks	Age
0	Amit	Male	85	20
1	Priya	Female	80	21
2	Raj	Male	78	22
3	Sneha	Female	Nan	23
4	Vikram	Male	76	24
5	Ananya	Female	82	25
6	Rohan	Male	Nan	26

In [4]:

```
cat
```

Out[4]:

```
['Name', 'Gender', 'Marks']
```

In [5]:

```
con
```

Out[5]:

```
['Age']
```

In [6]:

```
c=avg=sum=0
for ele in df['Marks']:
    if str(ele).isnumeric():
        c+=1
        sum+=ele
if c>0:
    avg=sum/c
df=df.replace(to_replace='Nan', value=avg)
df
```

```
C:\Users\RL LAB STAFF\AppData\Local\Temp\ipykernel_4652\2556432852.py:8: F
utureWarning: Downcasting behavior in `replace` is deprecated and will be
removed in a future version. To retain the old behavior, explicitly call `
result.infer_objects(copy=False)`. To opt-in to the future behavior, set `
pd.set_option('future.no_silent_downcasting', True)`
  df=df.replace(to_replace='Nan', value=avg)
```

Out[6]:

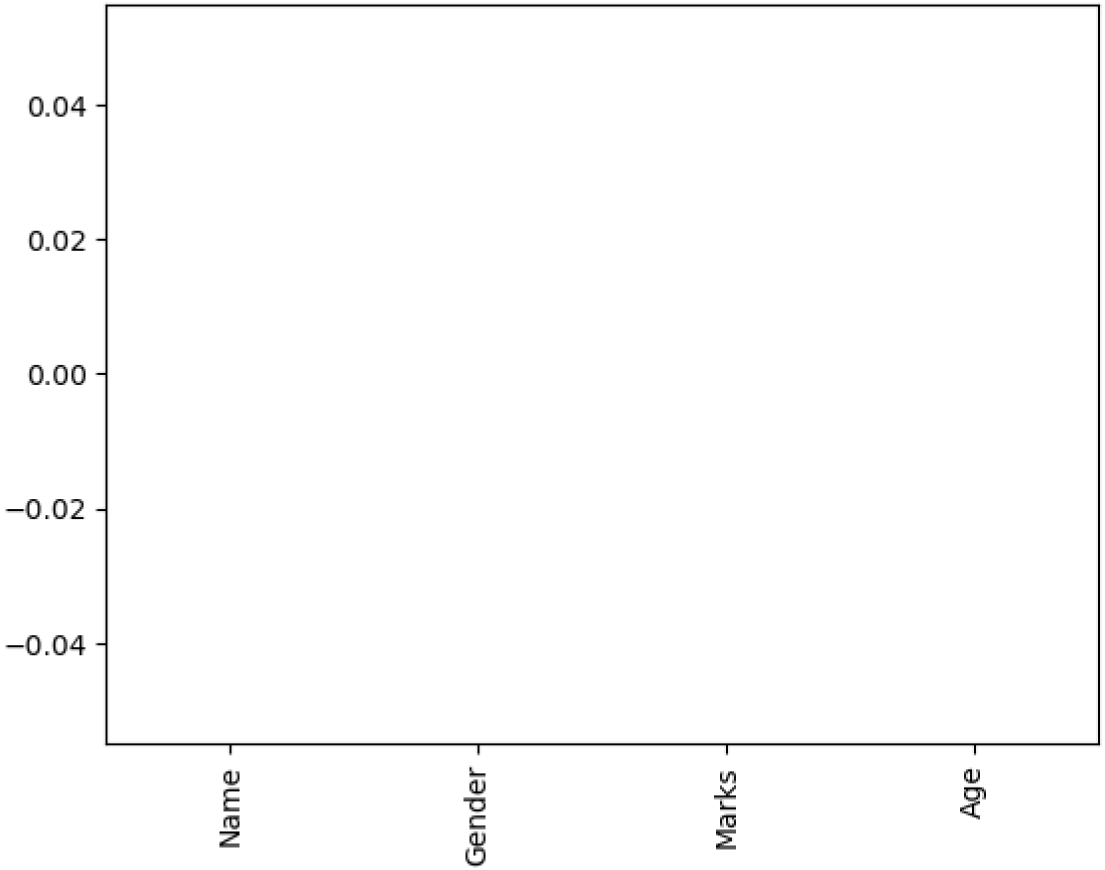
	Name	Gender	Marks	Age
0	Amit	Male	85.0	20
1	Priya	Female	80.0	21
2	Raj	Male	78.0	22
3	Sneha	Female	80.2	23
4	Vikram	Male	76.0	24
5	Ananya	Female	82.0	25
6	Rohan	Male	80.2	26

In [7]:

```
df.isna().sum().plot(kind="bar")
```

Out[7]:

```
<Axes: >
```



In [8]:

```
df['Gender']=df['Gender'].map({'Male':0,'Female':1,}).astype(int)
df
```

Out[8]:

	Name	Gender	Marks	Age
0	Amit	0	85.0	20
1	Priya	1	80.0	21
2	Raj	0	78.0	22
3	Sneha	1	80.2	23
4	Vikram	0	76.0	24
5	Ananya	1	82.0	25
6	Rohan	0	80.2	26

In [9]:

```
df=df[df['Marks']>80]
df
```

Out[9]:

	Name	Gender	Marks	Age
0	Amit	0	85.0	20
3	Sneha	1	80.2	23
5	Ananya	1	82.0	25
6	Rohan	0	80.2	26

In [10]:

```
df=df.drop(['Age'], axis=1)
df
```

Out[10]:

	Name	Gender	Marks
0	Amit	0	85.0
3	Sneha	1	80.2
5	Ananya	1	82.0
6	Rohan	0	80.2

In [11]:

```
data1 = {
    "Name": ["Amit", "Priya", "Raj", "Sneha", "Vikram", "Ananya", "Rohan"
    "Gender": ["Male", "Female", "Male", "Female", "Male", "Female", "Mal
    "Marks": [85, 80, 78, 'Nan', 76, 82, 'Nan'],
    "id": [120,121,122,123,124,125,126]
}
df1 = pd.DataFrame(data1)
print(df1)
```

```
     Name  Gender Marks   id
0    Amit    Male    85  120
1   Priya  Female    80  121
2     Raj    Male    78  122
3   Sneha  Female   Nan  123
4  Vikram    Male    76  124
5  Ananya  Female    82  125
6   Rohan    Male   Nan  126
```

In [12]:

```
data2 = {
    "Fee":[1000, 100000, 50000, 2000, 500, 70000, 30000],
       "id": [120,121,122,123,124,125,126]
}
df2 = pd.DataFrame(data2)
print(df)
```

```
     Name  Gender  Marks
0    Amit       0   85.0
3   Sneha       1   80.2
5  Ananya       1   82.0
6   Rohan       0   80.2
```

In [13]:

```
df3 = pd.merge(df1, df2)
df3
```

Out[13]:

	Name	Gender	Marks	id	Fee
0	Amit	Male	85	120	1000
1	Priya	Female	80	121	100000
2	Raj	Male	78	122	50000
3	Sneha	Female	Nan	123	2000
4	Vikram	Male	76	124	500
5	Ananya	Female	82	125	70000
6	Rohan	Male	Nan	126	30000

In [33]:

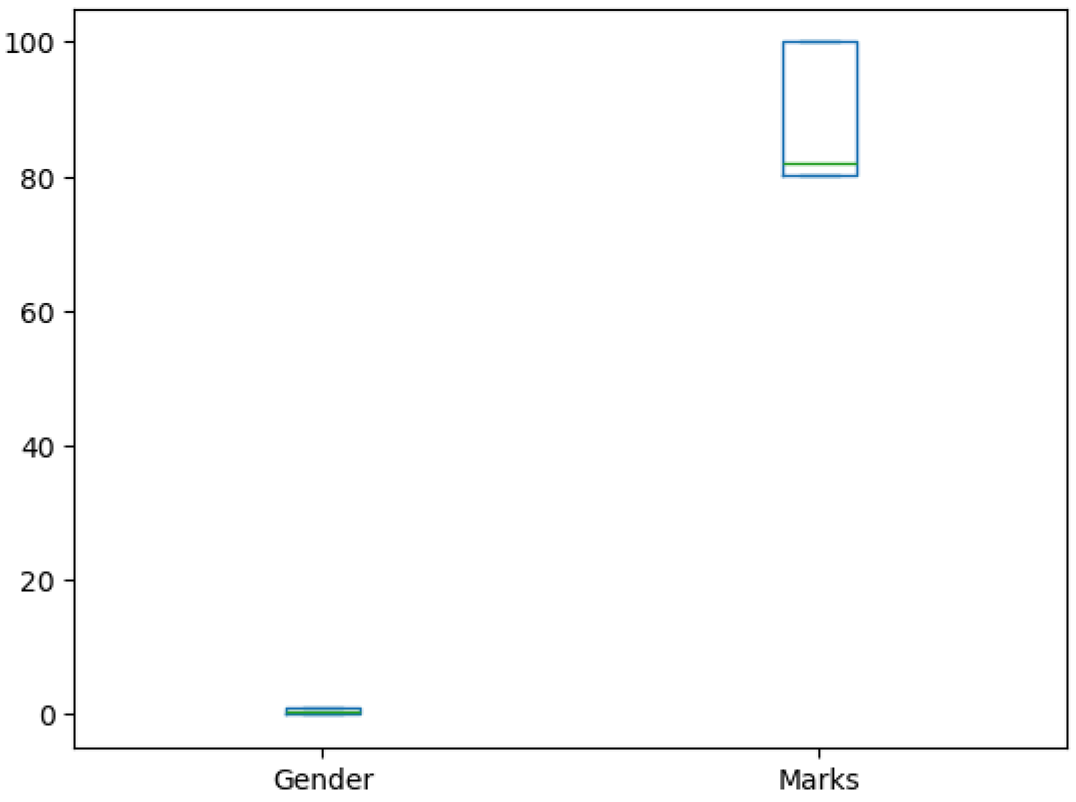
```
df.loc[0, 'Marks']= 100
print(df)
```

```
     Name  Gender  Marks
0    Amit     0.0  100.0
3   Sneha     1.0   80.2
5  Ananya     1.0   82.0
6   Rohan     0.0   80.2
2     NaN     NaN  100.0
```

In [35]:

```
df.plot.box()
```

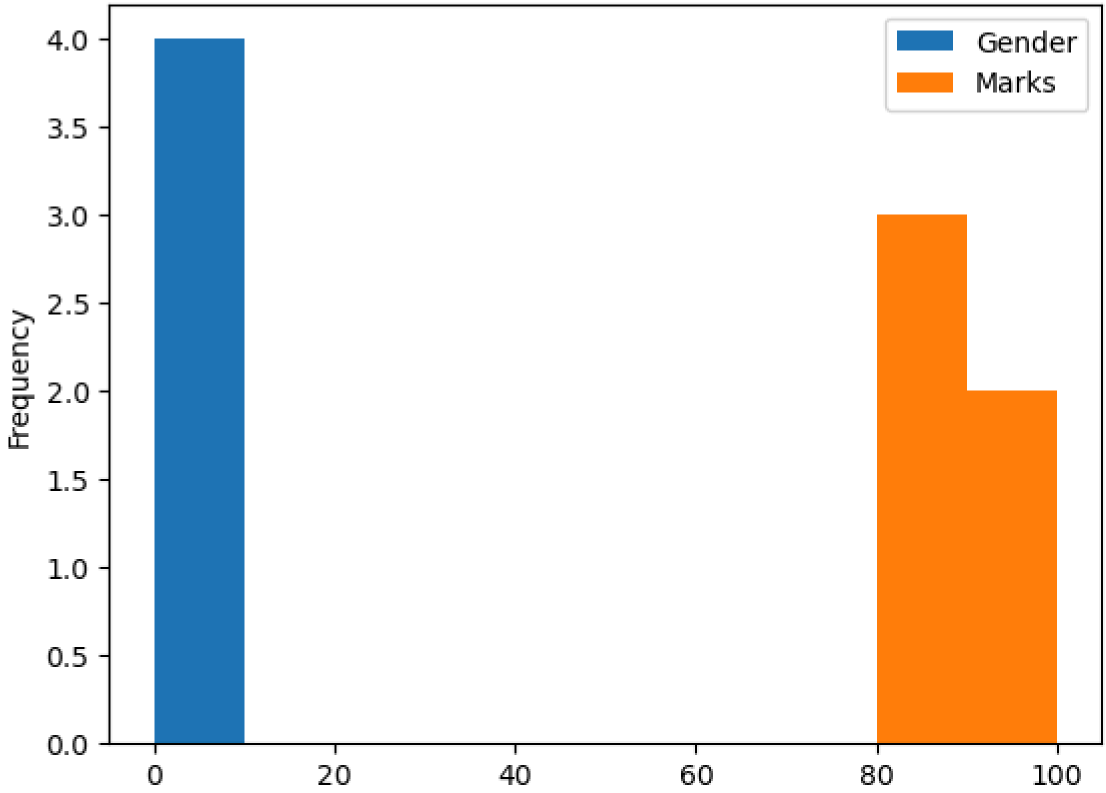
Out[35]: `<Axes: >`



In [37]:

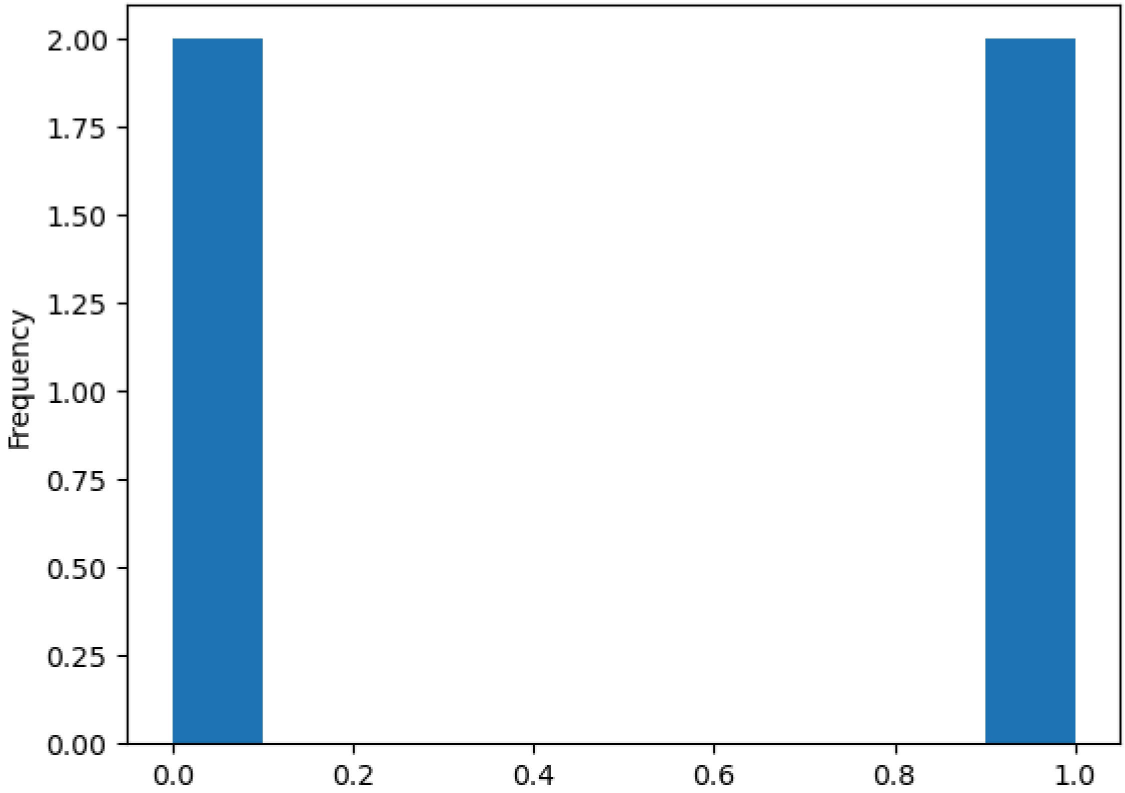
```
df.plot.hist()
```

Out[37]: `<Axes: ylabel='Frequency'>`



```
In [41]: df['Gender'].plot.hist()
```

```
Out[41]: <Axes: ylabel='Frequency'>
```



```
In [ ]:
```