IMPORTING NECESSARY LIBRARIES

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline
```

LOAD THE DATA FROM EACH SHEETS

```
f1 = pd.read_excel('Inventory_data.xlsx', sheet_name='MB51')
f2 = pd.read_excel('Inventory_data.xlsx', sheet_name='MC.9')
f3 = pd.read_excel('Inventory_data.xlsx', sheet_name='MB51(Backflush)-Summary')
f4 = pd.read_excel('Inventory_data.xlsx', sheet_name='MC.9(Stock)-Summary')
```

```
f1.to_csv('MB51.csv', index=False)
f2.to_csv('MC.9.csv', index=False)
f3.to_csv('MB51(Backflush)-Summary.csv', index=False)
f4.to_csv('MC.9(Stock)-Summary.csv', index=False)
```

```
df1 = pd.read_csv('MB51.csv')
df2 = pd.read_csv('MC.9.csv')
df3 = pd.read_csv('MB51(Backflush)-Summary.csv')
df4 = pd.read_csv('MC.9(Stock)-Summary.csv')
```

```
<ipython-input-331-14d2b0863be4>:1: DtypeWarning: Columns (3,4,5,7,9,10,12,13,14,16,17,18,20,21,22,25,27,28,30,31,32,33,35,37,38,39
  df1 = pd.read_csv('MB51.csv')
```

```
df1.shape
```

```
(110398, 68)
```

```
df2.shape
```

```
(6709, 12)
```

```
df3.shape
```

```
(260, 26)
```

```
df4.shape
```

```
(307, 46)
```

```
df1.head()
```

	Year	Month	Pstng Date	Time	Vendor	Material	Description	Reference	Doc. Date	Quantity	...
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	..
1	2,021	2021-01-01	31.01.2021	19:20:03	NaN	100300630	Limiting & Quick Release Valve	NaN	31.01.2021	10	..
2	2,021	2021-01-01	31.01.2021	19:19:50	NaN	100300630	Limiting & Quick Release Valve	NaN	31.01.2021	10	..
3	2,021	2021-01-01	31.01.2021	13:38:15	NaN	100302910	Q.S.P.Valve (New)-Tata	NaN	31.01.2021	15	..
4	2,021	2021-01-01	31.01.2021	17:29:39	NaN	100304920	Quick Relase Valve (Voss)	NaN	31.01.2021	20	..

```
df2.head()
```

	Part	Material	Value of valuated stock	INR	Valuated stock	UOM	ValCl	Last cons.	LstReceipt	Month	Year	De
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

```
df3.head()
```

	Sum of Quantity	Unnamed: 1	Column Labels	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9
0	NaN	NaN	2021	2021	2021	2021	2021	2021	2021	2021
1	Row Labels	Description	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug
2	100250040	Low Pressure Indicator Switch	335	365	1220	1310	100	245	135	510
3	100250050	Air Cylinder	NaN	NaN	182	295	NaN	NaN	NaN	56
4	100250150	Stop Light Switch-(M15) Assy	3216	4150	1750	2685	1425	2340	1390	1645

5 rows × 26 columns

```
df4.head()
```

	Unnamed: 0	Unnamed: 1	Column Labels	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9
0	NaN	NaN	2021	2021	2021	2021	2021	2021	2021	2021
1	NaN	NaN	Jan	Jan	Feb	Feb	Mar	Mar	Apr	Apr
2	Row Labels	Part - Description	Valuated stock Qty	Value of valuated stock	Valuated stock Qty	Value of valuated stock	Valuated stock Qty	Value of valuated stock	Valuated stock Qty	Value of valuated stock
3	-	-	NaN	NaN	0	0	NaN	NaN	NaN	NaN
4	100250040	Low Pressure Indicator Switch	0	0	150	7648.5	400	20396	800	40792

5 rows × 46 columns

SUMMARY OF THE DATASETS

```
df1.describe()
```

	Year
count	110,337
mean	2,021
std	0
min	2,021
25%	2,021
50%	2,021
75%	2,022
max	2,022

```
df2.describe()
```

	Year
count	6,646
mean	2,021
std	0
min	2,021
25%	2,021
50%	2,021
75%	2,022

```
df3.describe()
```

	Sum of Quantity	Unnamed: 1	Column Labels	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed
count	259	258	177	185	201	193	171	180	194	18
unique	259	76	139	143	163	147	112	135	152	13
top	Row Labels	Foot brake valve	10	8	10	40	5	80	24	
freq	1	32	6	7	4	5	8	4	6	

4 rows × 26 columns

```
df4.describe()
```

	Unnamed: 0	Unnamed: 1	Column Labels	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed
count	305	304	306	306	307	307	306	306	306	30
unique	305	98	57	107	59	130	73	148	75	16
top	Row Labels	Foot brake valve	0	0	0	0	0	0	0	
freq	1	32	200	200	177	177	158	158	147	14

4 rows × 46 columns

MISSING VALUES

```
df1.isnull().sum()
```

```
Year             61
Month            61
Pstng Date       41
Time             41
Vendor       110378
              ...
Crcy             41
GR/GI Sl     110378
Aut          110378
OrLi         110378
Multi AA     110378
Length: 68, dtype: int64
```

```
df2.isnull().sum()
```

```
Part                        1
Material                   44
Value of valuated stock    44
INR                        65
     Valuated stock        44
UOM                        65
ValCl                      44
Last cons.                604
LstReceipt                419
Month                      63
```

```
    Year                          63
    Part - Description             1
    dtype: int64
```

DUPLICATE ROWS

```
df1.duplicated().sum()
```

```
    57
```

```
df2.duplicated().sum()
```

```
    60
```

```
df3.duplicated().sum()
```

```
    0
```

```
df4.duplicated().sum()
```

```
    0
```

```
df1.dtypes
```

```
    Year           float64
    Month           object
    Pstng Date      object
    Time            object
    Vendor          object
                     ...
    Crcy            object
    GR/GI Sl        object
    Aut             object
    OrLi            object
    Multi AA        object
    Length: 68, dtype: object
```

```
df2.dtypes
```

```
    Part                           object
    Material                       object
    Value of valuated stock        object
    INR                            object
         Valuated stock            object
    UOM                            object
    ValCl                          object
    Last cons.                     object
    LstReceipt                     object
    Month                          object
    Year                          float64
    Part - Description             object
    dtype: object
```

```
df1.tail()
```

	Year	Month	Pstng Date	Time	Vendor	Material	Description	Reference	Doc. Date	Quantity
110393	2,022	2022-10-01	01.10.2022	1	NaN	9718990000	Inversion Relay Valve	NaN	01.10.2022	1
110394	2,022	2022-10-01	01.10.2022	1	NaN	9718990000	Inversion Relay Valve	NaN	01.10.2022	1
110395	2,022	2022-10-01	01.10.2022	0	NaN	9718990000	Inversion Relay Valve	NaN	01.10.2022	8
110396	2,022	2022-10-01	01.10.2022	1	NaN	9718990000	Inversion Relay Valve	NaN	01.10.2022	5
110397	2,022	2022-10-01	01.10.2022	1	NaN	9718990000	Inversion Relay Valve	NaN	01.10.2022	1

5 rows × 68 columns

```
df2.tail()
```

	Part	Material	Value of valuated stock	INR	Valuated stock	UOM	ValCl	Last cons.	LstReceipt	Month	Year
6704	9718990110	9718990110 Inversion Relay Valve	0	INR	0	NOS	7921	22.09.2022	22.09.2022	2022-10-01	2,022
6705	9718991200	9718991200 Inversion	0	INR	0	NOS	7921	13.08.2022	25.01.2021	2022-	2,022

PARTS CONTRIBUTED MORE

```
df1.columns
```

```
Index(['Year', 'Month', 'Pstng Date', 'Time', 'Vendor', 'Material',
       'Description', 'Reference', 'Doc. Date', '   Quantity', 'Mat. Doc.',
       'Batch', 'PO', 'CoCd', '  Amount in LC', 'User name', 'Plnt', 'MvT',
       'OPU', 'Entry Date', 'Reas.', 'Customer', 'MatYr', 'HeaderText', 'SLoc',
       'G/L Acct', 'Mvt Type Text', 'Reserv.No.', 'Cost Ctr', 'D/C', 'Cns',
       'Rec', 'Sales Ord.', 'SO item', 'OUn', 'Order', 'Name 1', 'S', 'Item',
       ' Qty in UnE', 'EUn', 'Asset', 'SNo.', 'Counter', 'Plan no.',
       'Qty OPUn', 'Quantity in', 'Val. Type', 'Smart No.', 'Item.1',
       'ExtAmnt LC', 'Sales Val.', 'Sales Ord..1', 'Sch.', 'SO Item', 'Mvt',
       'BUn', 'Network', 'OpAc', 'WBS Elem.', 'Itm', 'TETy', 'SV inc VAT',
       'Crcy', 'GR/GI Sl', 'Aut', 'OrLi', 'Multi AA'],
      dtype='object')
```

```
# format the quantity
def format_qty(qty):
    if qty >= 1000000000:
        return f'{qty/1000000000:,.2f}B'
    elif qty >= 1000000:
        return f'{qty/1000000:,.2f}M'
    elif qty >= 1000:
        return f'{qty/1000:,.2f}K'
    else:
        return f'{qty:,.2f}'
```

```
total_qty = df1.groupby(['Material', 'Description'])['   Quantity'].sum()
total_qty = pd.to_numeric(total_qty, errors='coerce')

sorted_qty = total_qty.sort_values(ascending=False)

sorted_qty = sorted_qty.apply(format_qty)

print(sorted_qty.head((4)))
```

```
Material    Description
4611000040  Foot brake valve                 181,010,105,153,510,110,785,625,121,051,656,27...
9618990410  Foot control valve w/treadle     31,173,134,232,520,200,234,359,446,010,477,906...
4214292650  Exhaust Brake assembly           4,522,222,222,222,222,626,394,212,482,296,600,...
100830600   E 1 Brake Valve                  88,551,768,687,282,086,377,635,893,677,791,162...
Name:    Quantity, dtype: object
```

The output above shows the total quantity for each material and part description.The quantity column is formatted with commas to make it more readable.The material are sorted in descending order based on the total quantity.

Trend impacting the Inventory

```
print(df2.columns)
```

```
Index(['Part', 'Material', 'Value of valuated stock', 'INR',
       '     Valuated stock', 'UOM', 'ValCl', 'Last cons.', 'LstReceipt',
       'Month', 'Year', 'Part - Description'],
      dtype='object')
```

```
print(df2['Value of valuated stock'].dtype)
```

```
object
```

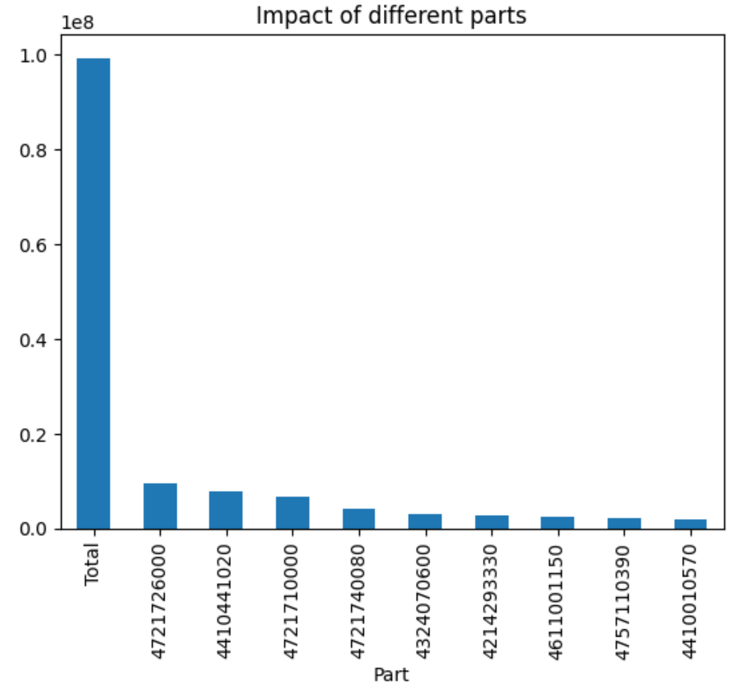
```
df2['Value of valuated stock'] = pd.to_numeric(df2['Value of valuated stock'], errors='coerce')
```

```
grouped_df = df2.groupby('Part')

value_by_part = grouped_df['Value of valuated stock'].sum()
value_by_part = value_by_part.sort_values(ascending=False)

value_by_part.head(10).plot(kind='bar', title='Impact of different parts')
```

```
<Axes: title={'center': 'Impact of different parts'}, xlabel='Part'>
```



To analyze the impact of different parts/assemblies on the inventory, we can use the data from the second sheet and group the data based on the Part or Material column. We can then calculate the sum of Value of valuated stock or Valuated stock to determine which parts/assemblies are contributing more.

```
# Replace non-finite values with -1
df2['Year'] = df2['Year'].replace([np.inf, -np.inf, np.nan], -1)

# Cast 'Year' column to int data type
df2['Year'] = df2['Year'].astype(int)
```

```
df2['Value of valuated stock'] = df2['Value of valuated stock'].astype(float)
```

```
df2['Valuated stock'] = df2['     Valuated stock'].str.strip()
```

```
df2.dtypes
```

```
Part                        object
Material                    object
Value of valuated stock    float64
INR                         object
     Valuated stock         object
UOM                         object
ValCl                       object
Last cons.                  object
LstReceipt                  object
Month                       object
Year                         int64
Part - Description          object
Valuated stock              object
dtype: object
```

```
df2['Valuated stock'] = df2['Valuated stock'].str.replace('Valuated stock', '0').astype(float)
```

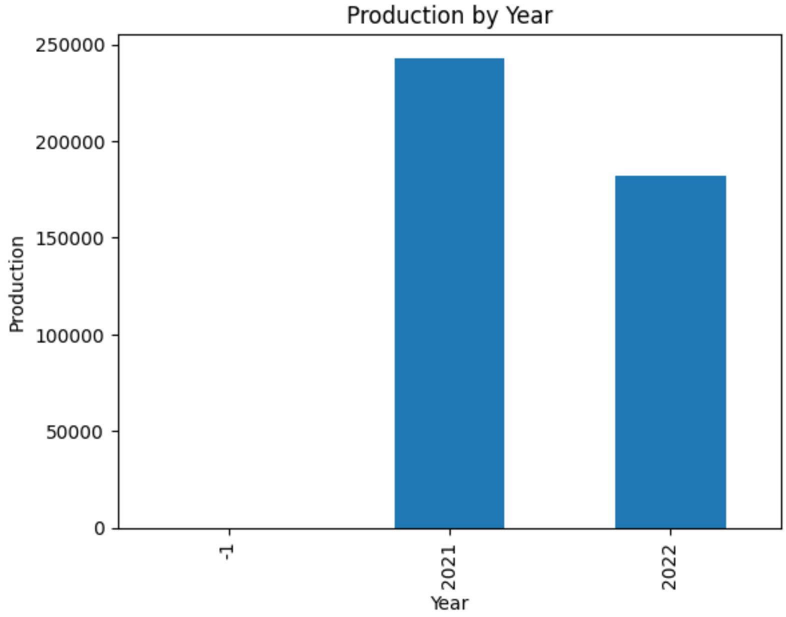
```
# Group data based on Year
grouped_df = df2.groupby('Year')

# Calculate the sum of Valuated stock column for each group
production_by_year = grouped_df['Valuated stock'].sum()
```

```
# Plot the production by year
production_by_year.plot(kind='bar', title='Production by Year')

# Set axis labels
plt.xlabel('Year')
plt.ylabel('Production')
```

```
    Text(0, 0.5, 'Production')
```



The resulting graph shows the total production of each year based on the 'Valuated stock' column.

The Minimum Inventory required

```
df1['Year'] = pd.to_datetime(df1['Year'], format='%Y')
df1['Month'] = pd.to_datetime(df1['Month'], format='%Y-%m-%d').dt.month

df1['   Quantity'] = pd.to_numeric(df1['   Quantity'], errors='coerce')


monthly_demand = df1.groupby(['Year', 'Month', 'Material'])['   Quantity'].sum().reset_index()
monthly_demand['Demand Rate'] = monthly_demand['   Quantity'] / 30


df1['   Quantity'] = pd.to_numeric(df1['   Quantity'], errors='coerce')
df1['Pstng Date'] = pd.to_numeric(df1['Pstng Date'], errors='coerce')


df1['Doc. Date'] = pd.to_numeric(df1['Doc. Date'], errors='coerce')


df1['Pstng Date'] = pd.to_datetime(df1['Pstng Date'])
df1['Doc. Date'] = pd.to_datetime(df1['Doc. Date'])


df1['Lead Time'] = (df1['Pstng Date'] - df1['Doc. Date']).dt.days


df1['Lead Time'] = (df1['Pstng Date'] - df1['Doc. Date']).dt.days
service_level = 0.95
safety_factor = 1.64



df1['Year'] = pd.to_datetime(df1['Year'], format='%Y')
df1['Month'] = pd.to_datetime(df1['Month'], format='%Y-%m-%d').dt.month

df1['   Quantity'] = pd.to_numeric(df1['   Quantity'], errors='coerce')

monthly_demand = df1.groupby(['Year', 'Month', 'Material'])['   Quantity'].sum().reset_index()
monthly_demand['Demand Rate'] = monthly_demand['   Quantity'] / 30

df1['Lead Time'] = (df1['Pstng Date'] - df1['Doc. Date']).dt.days

service_level = 0.95
safety_factor = 1.64  # for 95% service level
```

```
df1['Demand during lead time'] = monthly_demand['Demand Rate'] * df1['Lead Time']
stddev_lead_time = df1.groupby('Material')['Demand during lead time'].std()

reorder_point = (monthly_demand['Demand Rate'].mean() * (stddev_lead_time * safety_factor).fillna(0) +
                 monthly_demand['Demand Rate'].mean() * df1['Lead Time'].mean())
safety_stock = stddev_lead_time * safety_factor
```

+ Code + Text

```
df1['Year'] = pd.to_datetime(df1['Year'], format='%Y')
df1['Month'] = pd.to_datetime(df1['Month'], format='%Y-%m-%d').dt.month

df1['   Quantity'] = pd.to_numeric(df1['   Quantity'], errors='coerce')

monthly_demand = df1.groupby(['Year', 'Month', 'Material'])['   Quantity'].sum().reset_index()
monthly_demand['Demand Rate'] = monthly_demand['   Quantity'] / 30

df1['Lead Time'] = (df1['Pstng Date'] - df1['Doc. Date']).dt.days

df1 = df1.merge(monthly_demand, on=['Year', 'Month', 'Material'], how='left')

df1['Demand during lead time'] = df1['Demand Rate'] * df1['Lead Time']
stddev_lead_time = df1.groupby('Material')['Demand during lead time'].std()

service_level = 0.95
safety_factor = 1.64  # for 95% service level

reorder_point = (monthly_demand['Demand Rate'].mean() * (stddev_lead_time * safety_factor).fillna(0) +
                 monthly_demand['Demand Rate'].mean() * df1['Lead Time'].mean())
safety_stock = stddev_lead_time * safety_factor
```

```
print(monthly_demand.columns)
```

```
Index(['Year', 'Month', 'Material', '   Quantity', 'Demand Rate',
       'Average monthly demand', 'Lead Time', 'Demand during lead time'],
      dtype='object')
```

```
monthly_demand["Average monthly demand"] = monthly_demand["Demand Rate"].rolling(window=3).mean()
```

```
# Add the Lead Time column to monthly_demand
monthly_demand = monthly_demand.merge(df1[['Material', 'Lead Time']].drop_duplicates(), on='Material')
```

```
monthly_demand["Demand during lead time"] = monthly_demand["Average monthly demand"] * monthly_demand["Lead Time"]
```

✓ 0s completed at 7:00 PM ● ×