```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

```
df=pd.read_excel("/content/DURGESH_MILK_DATA.xlsx")
df
```

	Date	Shift	Kgs	Ltrs	Fat%	Snf%	Kg Fat	Total Amt
0	2024-07-21	AM	18.6	18.1	6.8	9	1.26	1011.84
1	2024-07-21	PM	17.3	16.8	7.0	9	1.21	968.80
2	2024-07-22	AM	14.7	14.3	6.9	9	1.01	811.44
3	2024-07-22	PM	16.1	15.6	7.1	9	1.14	914.48
4	2024-07-23	AM	10.7	10.4	7.7	9	0.82	659.12
...	...	...	...	...	...	...	...	...
95	2024-10-08	PM	18.6	18.1	6.6	9	1.23	957.53
96	2024-10-09	AM	17.7	17.2	7.1	9	1.26	980.23
97	2024-10-09	PM	17.9	17.4	6.7	9	1.20	935.45
98	2024-10-10	AM	19.9	19.3	7.0	9	1.39	1086.54
99	2024-10-10	PM	16.6	16.1	6.5	9	1.08	841.62

100 rows × 8 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.shape
```

```
(100, 8)
```

```
df.columns
```

```
Index(['Date', 'Shift', 'Kgs', 'Ltrs', 'Fat%', 'Snf%', 'Kg Fat', 'Total Amt'], dtype='object')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 8 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       100 non-null    datetime64[ns]
 1   Shift      100 non-null    object
 2   Kgs        100 non-null    float64
 3   Ltrs       100 non-null    float64
 4   Fat%       100 non-null    float64
 5   Snf%       100 non-null    int64
 6   Kg Fat     100 non-null    float64
 7   Total Amt  100 non-null    float64
dtypes: datetime64[ns](1), float64(5), int64(1), object(1)
memory usage: 6.4+ KB
```

```
df.describe()
```

	Date	Kgs	Ltrs	Fat%	Snf%	Kg Fat	Total Amt
count	100	100.000000	100.000000	100.000000	100.0	100.000000	100.00000
mean	2024-08-27 16:48:00	16.561000	16.078000	6.831000	9.0	1.133900	973.42530
min	2024-07-21 00:00:00	9.200000	8.900000	6.200000	9.0	0.590000	459.26000
25%	2024-08-03 00:00:00	15.050000	14.650000	6.500000	9.0	1.020000	802.54250
50%	2024-08-15 12:00:00	16.700000	16.200000	6.800000	9.0	1.140000	901.59500
75%	2024-09-28 00:00:00	18.225000	17.725000	7.100000	9.0	1.245000	980.79500
max	2024-10-10 00:00:00	20.400000	19.800000	8.500000	9.0	1.490000	9393.12000
std	NaN	2.063923	2.005879	0.370829	0.0	0.172591	861.01409

```
# checking for null values
df.isna().sum().sort_values(ascending = False)
```

	0
Date	0
Shift	0
Kgs	0
Ltrs	0
Fat%	0
Snf%	0
Kg Fat	0
Total Amt	0

dtype: int64

```
df['Date'].unique()
```

```
<DatetimeArray>
['2024-07-21 00:00:00', '2024-07-22 00:00:00', '2024-07-23 00:00:00',
 '2024-07-24 00:00:00', '2024-07-25 00:00:00', '2024-07-26 00:00:00',
 '2024-07-27 00:00:00', '2024-07-28 00:00:00', '2024-07-29 00:00:00',
 '2024-07-30 00:00:00', '2024-08-11 00:00:00', '2024-08-12 00:00:00',
 '2024-08-13 00:00:00', '2024-08-14 00:00:00', '2024-08-15 00:00:00',
 '2024-08-16 00:00:00', '2024-08-17 00:00:00', '2024-08-18 00:00:00',
 '2024-08-19 00:00:00', '2024-08-20 00:00:00', '2024-08-01 00:00:00',
 '2024-08-02 00:00:00', '2024-08-03 00:00:00', '2024-08-04 00:00:00',
 '2024-08-05 00:00:00', '2024-08-06 00:00:00', '2024-08-07 00:00:00',
 '2024-08-08 00:00:00', '2024-08-09 00:00:00', '2024-08-10 00:00:00',
 '2024-09-21 00:00:00', '2024-09-22 00:00:00', '2024-09-23 00:00:00',
 '2024-09-24 00:00:00', '2024-09-25 00:00:00', '2024-09-26 00:00:00',
 '2024-09-27 00:00:00', '2024-09-28 00:00:00', '2024-09-29 00:00:00',
 '2024-09-30 00:00:00', '2024-10-01 00:00:00', '2024-10-02 00:00:00',
 '2024-10-03 00:00:00', '2024-10-04 00:00:00', '2024-10-05 00:00:00',
 '2024-10-06 00:00:00', '2024-10-07 00:00:00', '2024-10-08 00:00:00',
 '2024-10-09 00:00:00', '2024-10-10 00:00:00']
Length: 50, dtype: datetime64[ns]
```

```
df['Date'].value_counts()
```

Date	count
2024-07-21	2
2024-09-28	2
2024-08-08	2
2024-08-09	2
2024-08-10	2
2024-09-21	2
2024-09-22	2
2024-09-23	2
2024-09-24	2
2024-09-25	2
2024-09-26	2
2024-09-27	2
2024-09-29	2
2024-07-22	2
2024-09-30	2
2024-10-01	2
2024-10-02	2
2024-10-03	2
2024-10-04	2
2024-10-05	2
2024-10-06	2
2024-10-07	2
2024-10-08	2
2024-10-09	2
2024-08-07	2
2024-08-06	2
2024-08-05	2
2024-08-04	2
2024-07-23	2
2024-07-24	2
2024-07-25	2
2024-07-26	2
2024-07-27	2
2024-07-28	2
2024-07-29	2
2024-07-30	2
2024-08-11	2
2024-08-12	2
2024-08-13	2
2024-08-14	2
2024-08-15	2
2024-08-16	2
2024-08-17	2
2024-08-18	2
2024-08-19	2

2024-08-20	2
2024-08-01	2
2024-08-02	2
2024-08-03	2
2024-10-10	2

dtype: int64

```
for i in df.columns:
  print(df[i].value_counts())
```

```
1.03    4
1.17    3
1.39    3
1.02    3
0.94    3
1.08    3
0.93    2
1.38    2
0.86    2
1.27    2
1.31    2
1.10    2
1.21    2
1.07    2
1.13    2
1.40    2
0.91    2
0.95    2
1.05    2
1.19    2
1.43    1
1.36    1
1.30    1
1.44    1
1.32    1
1.29    1
1.33    1
1.41    1
1.49    1
1.15    1
0.90    1
0.59    1
0.81    1
0.85    1
0.87    1
1.06    1
0.83    1
0.92    1
0.97    1
0.96    1
0.99    1
0.82    1
1.01    1
1.35    1
Name: count, dtype: int64
Total Amt
731.64     2
885.77     2
937.95     2
1011.84    1
1074.37    1
          ..
733.82     1
718.85     1
725.01     1
846.30     1
841.62     1
Name: count, Length: 97, dtype: int64
```

```
df.isnull()
```

	Date	Shift	Kgs	Ltrs	Fat%	Snf%	Kg Fat	Total Amt
0	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...
95	False	False	False	False	False	False	False	False
96	False	False	False	False	False	False	False	False
97	False	False	False	False	False	False	False	False
98	False	False	False	False	False	False	False	False
99	False	False	False	False	False	False	False	False

100 rows × 8 columns

```
df.drop_duplicates()
```

	Date	Shift	Kgs	Ltrs	Fat%	Snf%	Kg Fat	Total Amt
0	2024-07-21	AM	18.6	18.1	6.8	9	1.26	1011.84
1	2024-07-21	PM	17.3	16.8	7.0	9	1.21	968.80
2	2024-07-22	AM	14.7	14.3	6.9	9	1.01	811.44
3	2024-07-22	PM	16.1	15.6	7.1	9	1.14	914.48
4	2024-07-23	AM	10.7	10.4	7.7	9	0.82	659.12
...	...	...	...	...	...	...	...	...
95	2024-10-08	PM	18.6	18.1	6.6	9	1.23	957.53
96	2024-10-09	AM	17.7	17.2	7.1	9	1.26	980.23
97	2024-10-09	PM	17.9	17.4	6.7	9	1.20	935.45
98	2024-10-10	AM	19.9	19.3	7.0	9	1.39	1086.54
99	2024-10-10	PM	16.6	16.1	6.5	9	1.08	841.62

100 rows × 8 columns

```
for i in df.columns:
  print(f"{i} are : ",df[i].unique())
```

```
Date are :  <DatetimeArray>
['2024-07-21 00:00:00', '2024-07-22 00:00:00', '2024-07-23 00:00:00',
 '2024-07-24 00:00:00', '2024-07-25 00:00:00', '2024-07-26 00:00:00',
 '2024-07-27 00:00:00', '2024-07-28 00:00:00', '2024-07-29 00:00:00',
 '2024-07-30 00:00:00', '2024-08-11 00:00:00', '2024-08-12 00:00:00',
 '2024-08-13 00:00:00', '2024-08-14 00:00:00', '2024-08-15 00:00:00',
 '2024-08-16 00:00:00', '2024-08-17 00:00:00', '2024-08-18 00:00:00',
 '2024-08-19 00:00:00', '2024-08-20 00:00:00', '2024-08-01 00:00:00',
 '2024-08-02 00:00:00', '2024-08-03 00:00:00', '2024-08-04 00:00:00',
 '2024-08-05 00:00:00', '2024-08-06 00:00:00', '2024-08-07 00:00:00',
 '2024-08-08 00:00:00', '2024-08-09 00:00:00', '2024-08-10 00:00:00',
 '2024-09-21 00:00:00', '2024-09-22 00:00:00', '2024-09-23 00:00:00',
 '2024-09-24 00:00:00', '2024-09-25 00:00:00', '2024-09-26 00:00:00',
 '2024-09-27 00:00:00', '2024-09-28 00:00:00', '2024-09-29 00:00:00',
 '2024-09-30 00:00:00', '2024-10-01 00:00:00', '2024-10-02 00:00:00',
 '2024-10-03 00:00:00', '2024-10-04 00:00:00', '2024-10-05 00:00:00',
 '2024-10-06 00:00:00', '2024-10-07 00:00:00', '2024-10-08 00:00:00',
 '2024-10-09 00:00:00', '2024-10-10 00:00:00']
Length: 50, dtype: datetime64[ns]
Shift are :  ['AM' 'PM']
Kgs are :  [18.6 17.3 14.7 16.1 10.7 16.5 15.7 16.  14.8 18.  15.5 14.9 15.2 16.4
 15.9 14.6 14.  14.1 13.6 14.2 14.3 14.4 15.6 16.3 18.5 19.  17.7 17.2
 16.8 16.7 13.  15.8 13.2 12.5 13.5  9.2 14.5 15.1 19.2 17.4 20.4 18.8
 19.4 18.3 17.6 18.1 18.4 17.8 18.7 17.1 19.1 19.3 19.6 16.9 18.9 18.2
 16.6 17.5 17.9 19.9]
Ltrs are :  [18.1 16.8 14.3 15.6 10.4 16.  15.2 15.5 14.4 17.5 15.  14.5 14.8 15.9
 15.4 14.2 13.6 13.7 13.2 13.8 13.9 14.  15.1 15.8 18.  18.4 17.2 16.7
 16.3 16.2 12.6 15.3 12.8 12.1 13.1  8.9 14.1 14.7 18.6 16.9 19.8 18.3
```

```
 18.8 17.8 17.1 17.6 17.9 17.3 18.2 16.6 18.5 18.7 19.  16.4 17.7 16.1
 17.  17.4 19.3]
Fat% are :  [6.8 7.  6.9 7.1 7.7 8.5 7.2 7.4 6.5 7.3 6.7 6.4 6.3 6.6 6.2 7.5]
Snf% are :  [9]
Kg Fat are :  [1.26 1.21 1.01 1.14 0.82 1.4  1.13 1.05 1.19 1.08 1.17 1.09 1.07 0.99
 1.03 1.02 0.95 0.94 0.96 0.91 0.97 0.93 0.92 1.2  1.24 1.15 1.1  0.83
 1.06 0.87 0.85 0.86 0.81 0.59 0.9  1.44 1.49 1.38 1.33 1.27 1.29 1.23
 1.31 1.39 1.43 1.3  1.41 1.36 1.32 1.35]
Total Amt are :  [1011.84  968.8   811.44  914.48  659.12 1122.    904.32  908.8   840.64
  953.12  866.64  936.    934.4   868.    854.54  792.28  826.88  839.68
  826.56  814.08  740.22  793.73  731.64  747.86  710.74  756.76  742.09
  846.3   725.01  718.85  733.82  803.09  728.83  851.84  798.72  836.55
  937.95  963.95  938.81  898.87  838.66 9393.12  846.69  939.12  859.72
  885.77  648.96  825.71  851.76  679.54  658.94  668.3   633.75  673.92
  459.26  712.53  800.9   701.22 1123.2   855.04 1161.58  982.49 1074.37
  970.63 1037.4   892.32  988.54  986.54 1004.64  958.    966.42 1021.02
  960.34 1087.55  963.61 1114.   1081.86 1016.5  1073.59  944.58 1100.74
  909.56 1093.25  961.58 1061.42 1030.07 1022.11  925.7   988.26 1055.81
  880.46  955.5   957.53  980.23  935.45 1086.54  841.62]
```

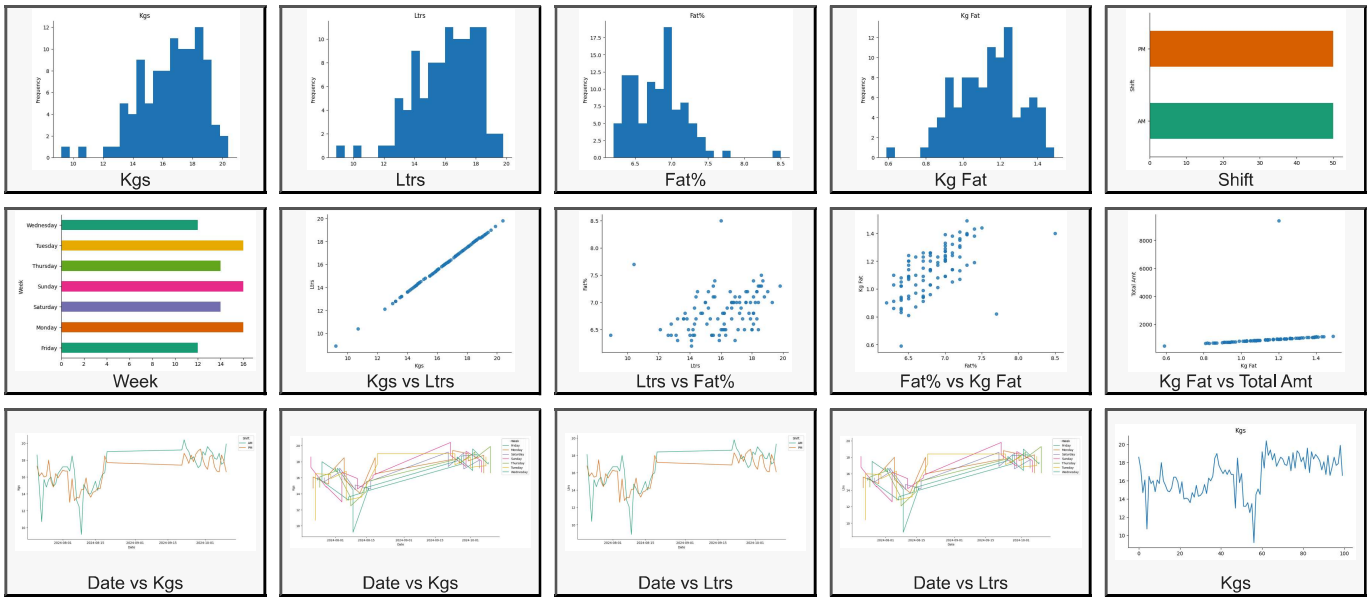
```
df['Week'] = df['Date'].dt.day_name()
```

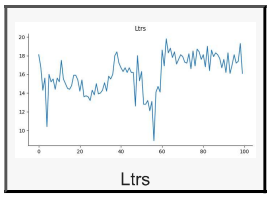
```
df
```

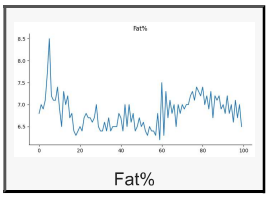
	Date	Shift	Kgs	Ltrs	Fat%	Snf%	Kg Fat	Total Amt	Week
0	2024-07-21	AM	18.6	18.1	6.8	9	1.26	1011.84	Sunday
1	2024-07-21	PM	17.3	16.8	7.0	9	1.21	968.80	Sunday
2	2024-07-22	AM	14.7	14.3	6.9	9	1.01	811.44	Monday
3	2024-07-22	PM	16.1	15.6	7.1	9	1.14	914.48	Monday
4	2024-07-23	AM	10.7	10.4	7.7	9	0.82	659.12	Tuesday
...	...	...	...	...	...	...	...	...	...
95	2024-10-08	PM	18.6	18.1	6.6	9	1.23	957.53	Tuesday
96	2024-10-09	AM	17.7	17.2	7.1	9	1.26	980.23	Wednesday
97	2024-10-09	PM	17.9	17.4	6.7	9	1.20	935.45	Wednesday
98	2024-10-10	AM	19.9	19.3	7.0	9	1.39	1086.54	Thursday
99	2024-10-10	PM	16.6	16.1	6.5	9	1.08	841.62	Thursday

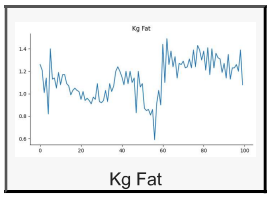
100 rows × 9 columns

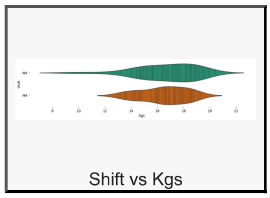
Next steps: Generate code with df | View recommended plots | New interactive sheet

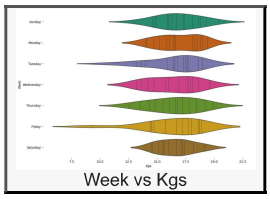


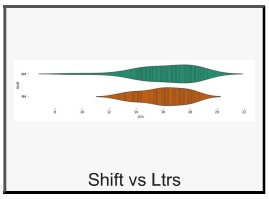


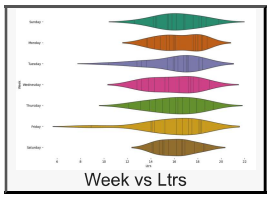













```
df1=df.drop('Date',axis=1)
df1
```

	Shift	Kgs	Ltrs	Fat%	Snf%	Kg Fat	Total Amt	Week
0	AM	18.6	18.1	6.8	9	1.26	1011.84	Sunday
1	PM	17.3	16.8	7.0	9	1.21	968.80	Sunday
2	AM	14.7	14.3	6.9	9	1.01	811.44	Monday
3	PM	16.1	15.6	7.1	9	1.14	914.48	Monday
4	AM	10.7	10.4	7.7	9	0.82	659.12	Tuesday
...	...	...	...	...	...	...	...	...
95	PM	18.6	18.1	6.6	9	1.23	957.53	Tuesday
96	AM	17.7	17.2	7.1	9	1.26	980.23	Wednesday
97	PM	17.9	17.4	6.7	9	1.20	935.45	Wednesday
98	AM	19.9	19.3	7.0	9	1.39	1086.54	Thursday
99	PM	16.6	16.1	6.5	9	1.08	841.62	Thursday

100 rows × 8 columns

Next steps: Generate code with df1 | View recommended plots | New interactive sheet

```
g1=df1.groupby(['Week','Shift'],as_index=False).mean()
g1
```

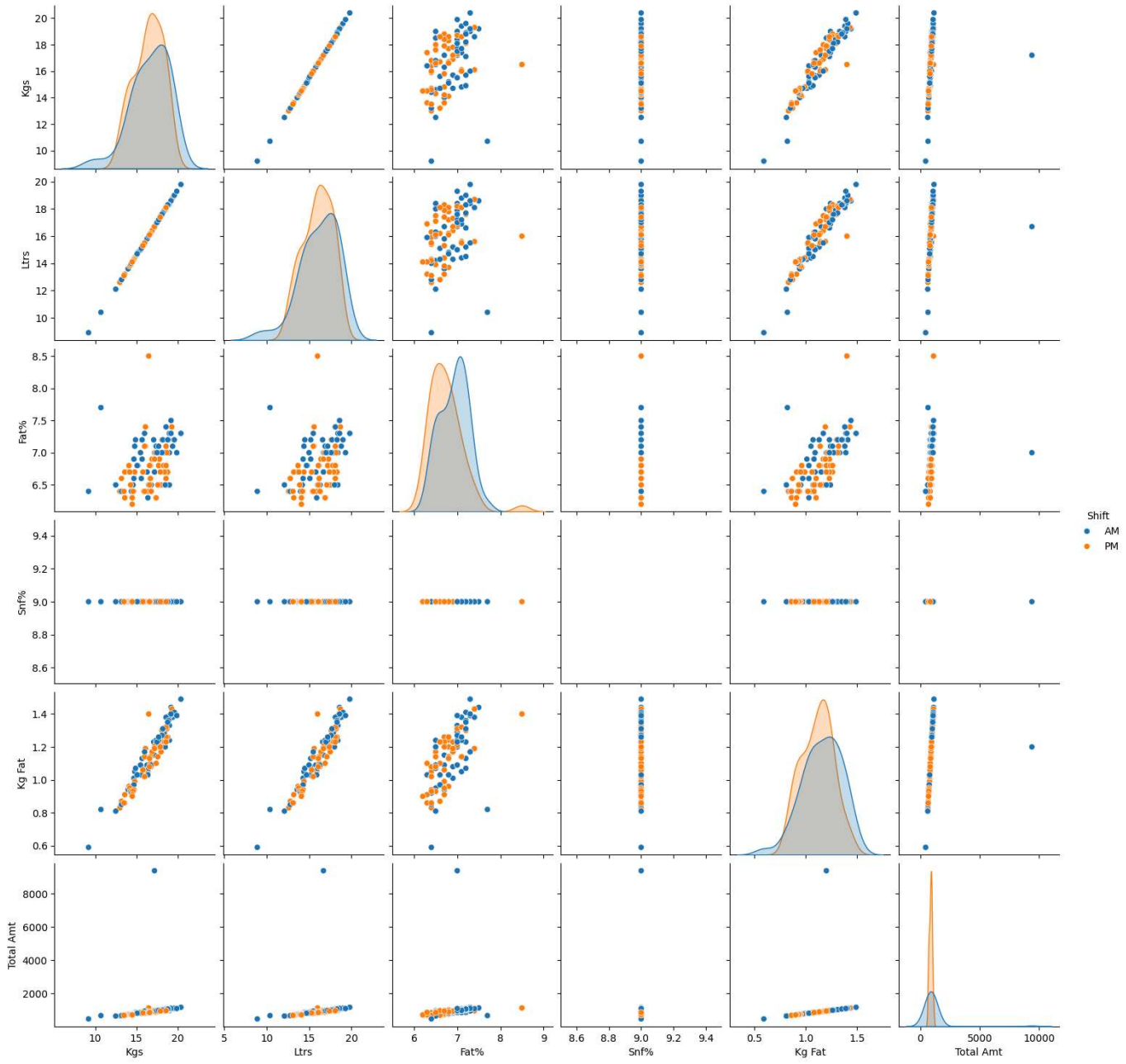
	Week	Shift	Kgs	Ltrs	Fat%	Snf%	Kg Fat	Total Amt
0	Friday	AM	15.516667	15.050000	6.816667	9.0	1.065000	2244.285000
1	Friday	PM	16.866667	16.400000	6.633333	9.0	1.123333	880.021667
2	Monday	AM	17.012500	16.525000	6.875000	9.0	1.171250	919.562500
3	Monday	PM	16.737500	16.250000	6.787500	9.0	1.135000	891.658750
4	Saturday	AM	16.914286	16.414286	7.085714	9.0	1.201429	940.451429
5	Saturday	PM	16.428571	15.942857	6.671429	9.0	1.100000	860.865714
6	Sunday	AM	17.125000	16.637500	6.925000	9.0	1.188750	932.207500
7	Sunday	PM	16.475000	15.987500	6.725000	9.0	1.112500	873.093750
8	Thursday	AM	16.785714	16.285714	6.942857	9.0	1.168571	914.855714
9	Thursday	PM	16.214286	15.742857	6.685714	9.0	1.085714	850.272857
10	Tuesday	AM	16.500000	16.012500	6.887500	9.0	1.132500	887.440000
11	Tuesday	PM	16.300000	15.825000	6.887500	9.0	1.122500	882.450000
12	Wednesday	AM	16.500000	16.016667	6.916667	9.0	1.148333	898.255000
13	Wednesday	PM	16.200000	15.733333	6.750000	9.0	1.096667	858.456667

Next steps: Generate code with g1 | View recommended plots | New interactive sheet

Data Visualization

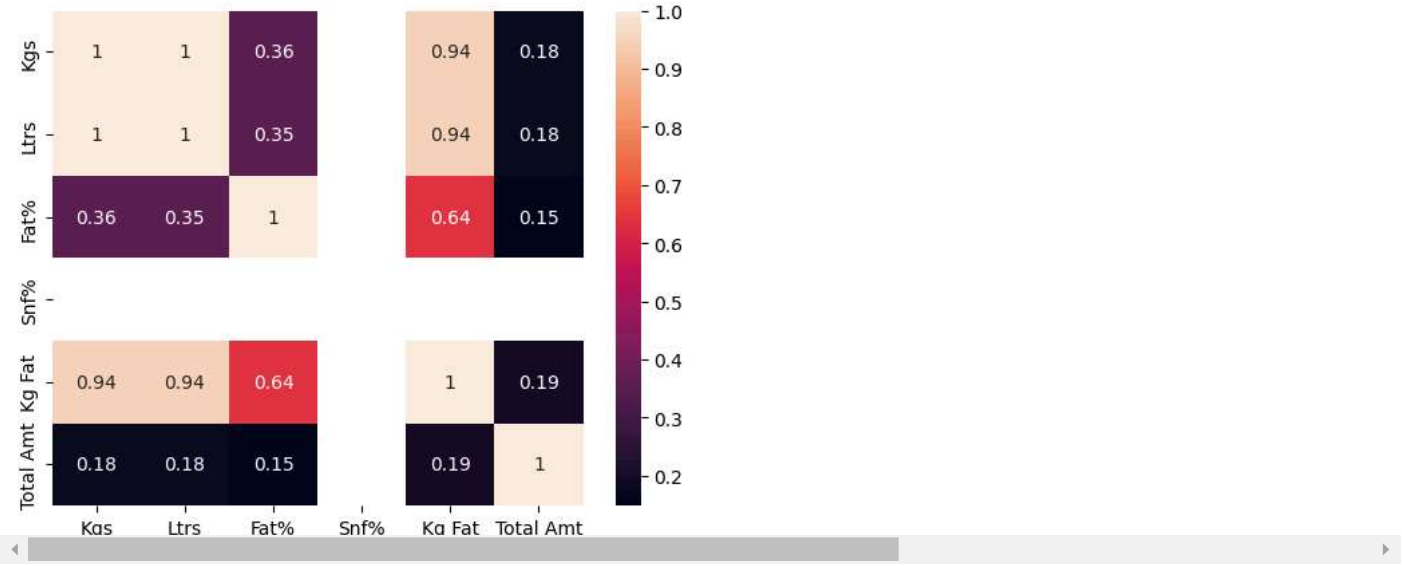
```
sns.pairplot(df,hue='Shift')
```

```
<seaborn.axisgrid.PairGrid at 0x7a017f40b070>
```



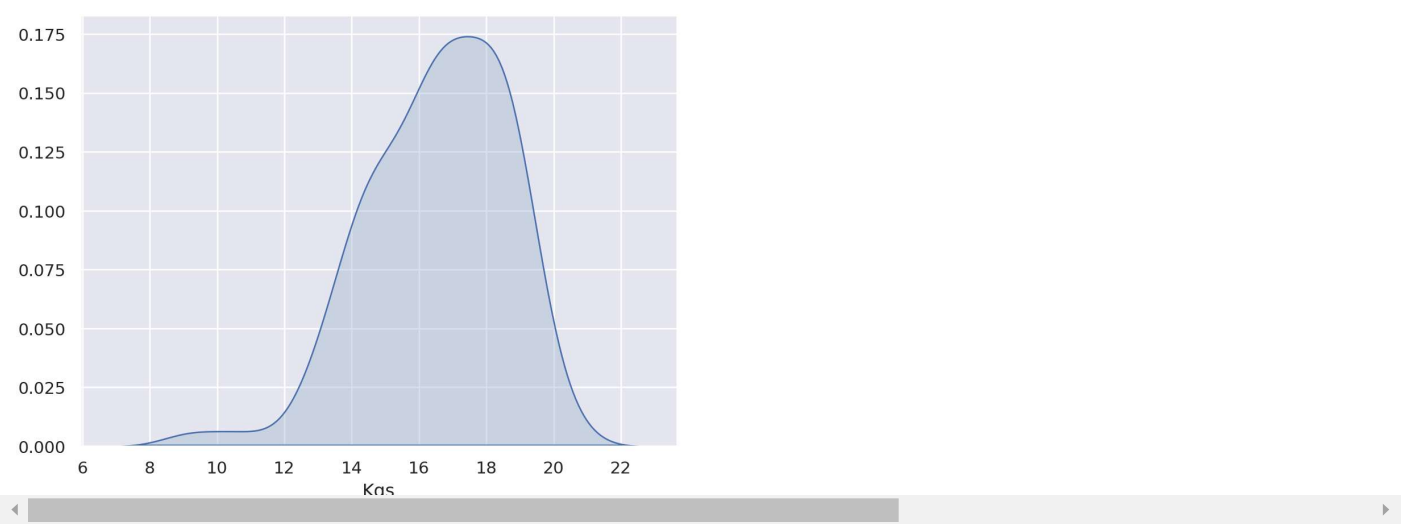

```
df2=df.drop(['Date','Shift','Week'],axis=1)
sns.heatmap(df2.corr(),annot=True)
```

<Axes: >

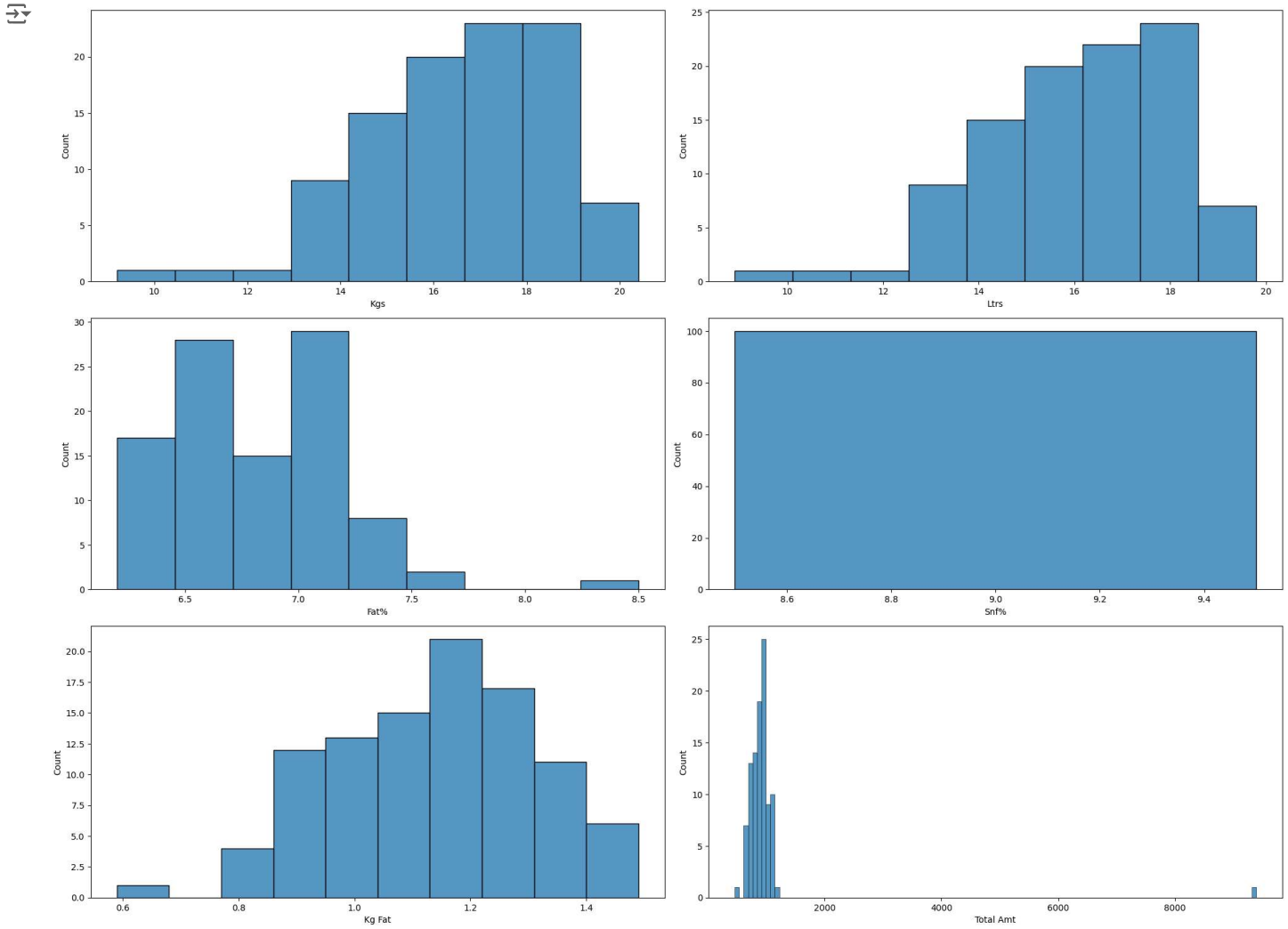



```
import seaborn.objects as so
```

```
p = so.Plot(df, x="Kgs")
p.add(so.Area(), so.KDE())
```

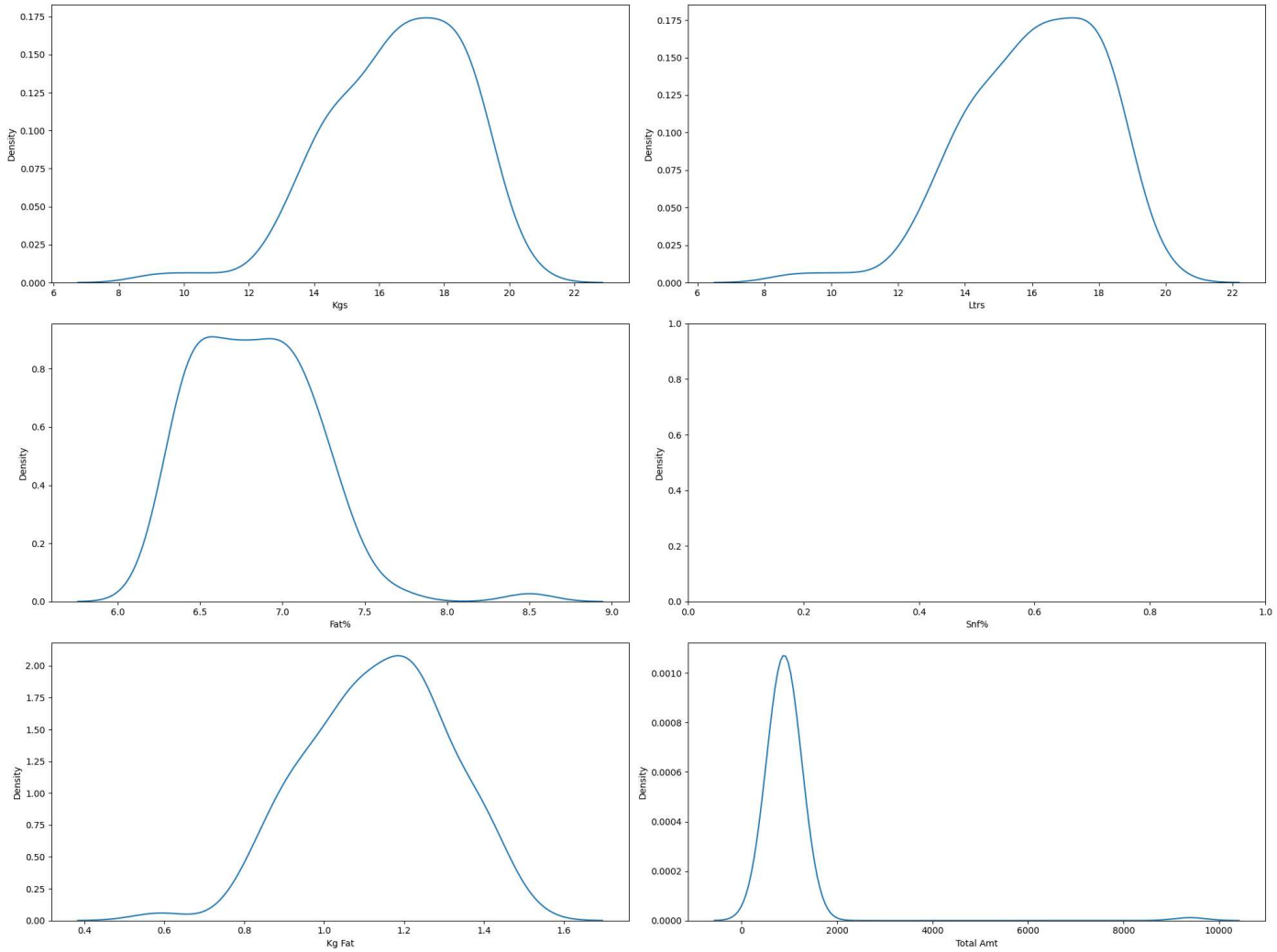



```
# checking numerical features distribution

plt.figure(figsize = (20, 15))
plotnumber = 1
num_cols=['Kgs','Ltrs','Fat%','Snf%','Kg Fat','Total Amt']

for column in num_cols:
    if plotnumber <= len(num_cols):
        ax = plt.subplot(3,2, plotnumber)
        sns.histplot(df[column])
        plt.xlabel(column)

    plotnumber += 1

plt.tight_layout()
plt.show()
```

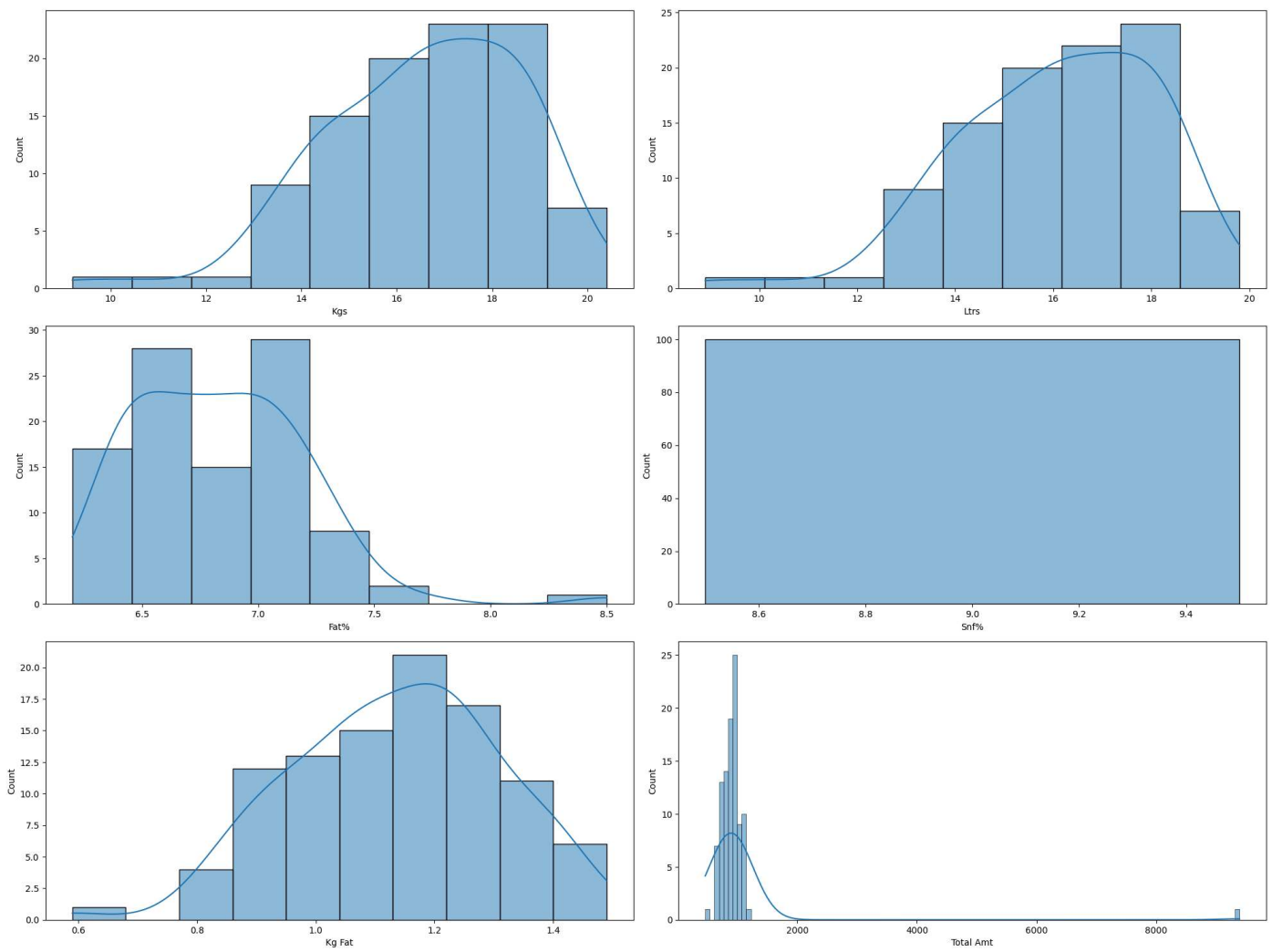



```
# checking numerical features distribution

plt.figure(figsize = (20, 15))
plotnumber = 1
num_cols=['Kgs','Ltrs','Fat%','Snf%','Kg Fat','Total Amt']

for column in num_cols:
    if plotnumber <= len(num_cols):
        ax = plt.subplot(3,2, plotnumber)
        sns.kdeplot(df[column])
        plt.xlabel(column)

    plotnumber += 1

plt.tight_layout()
plt.show()
```

```
<ipython-input-23-8ad742622ad2>:10: UserWarning: Dataset has 0 variance; skipping density estimate. Pass `warn_singular=False` to disabl
  sns.kdeplot(df[column])
```

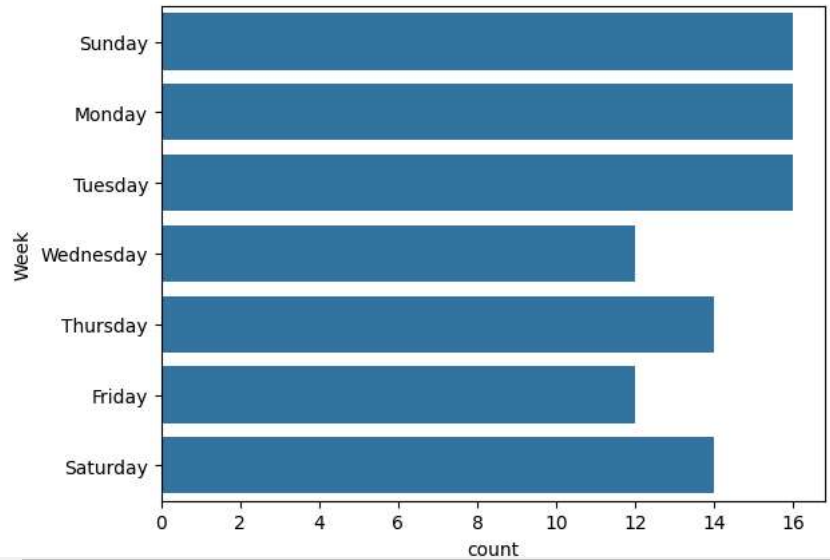



```
# checking numerical features distribution

plt.figure(figsize = (20, 15))
plotnumber = 1
num_cols=['Kgs','Ltrs','Fat%','Snf%','Kg Fat','Total Amt']

for column in num_cols:
    if plotnumber <= len(num_cols):
        ax = plt.subplot(3,2, plotnumber)
        sns.histplot(df[column],kde=True)
        plt.xlabel(column)

    plotnumber += 1

plt.tight_layout()
plt.show()
```



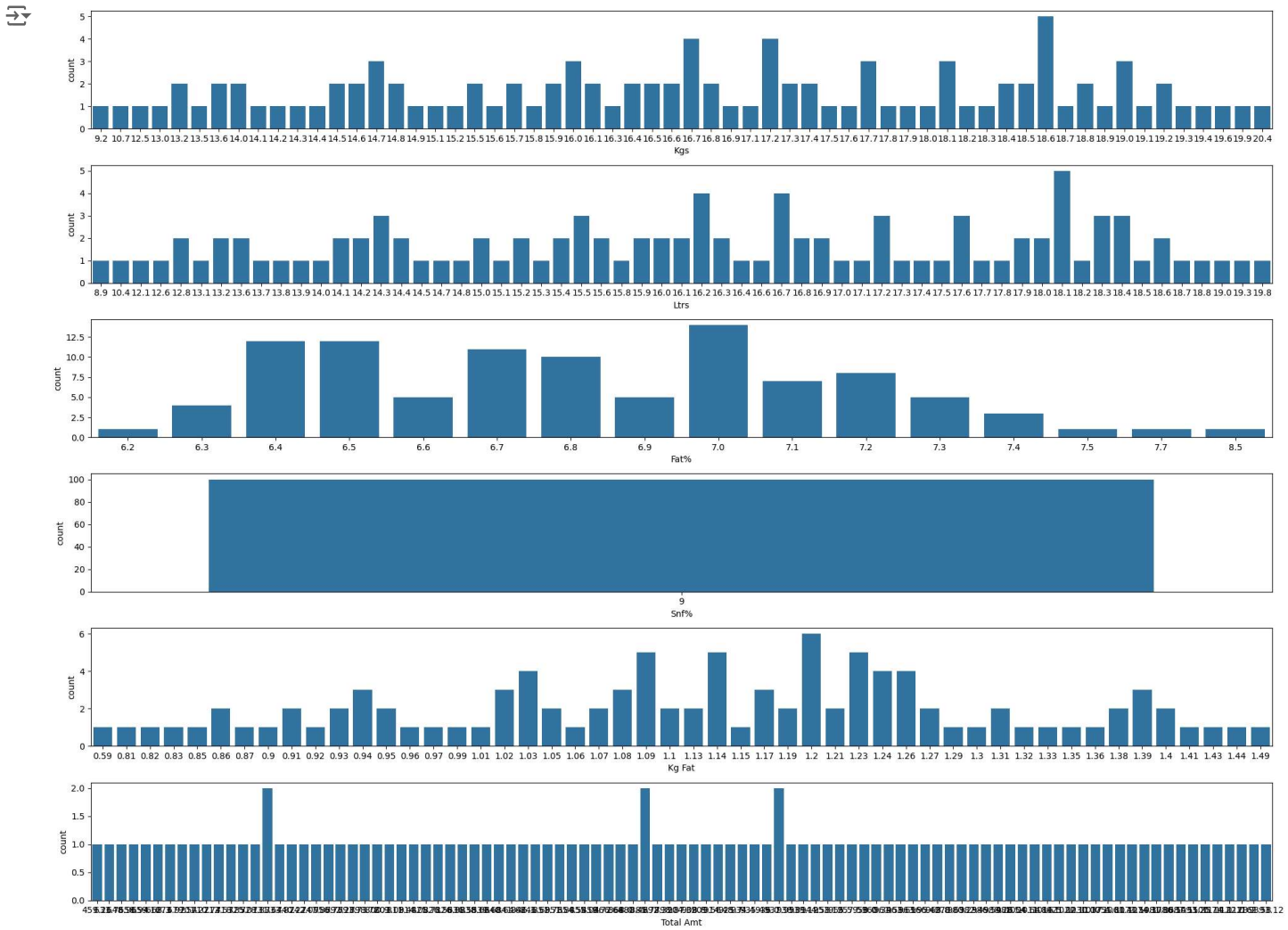

```
sns.countplot(df, y="Week")
```

```
<Axes: xlabel='count', ylabel='Week'>
```

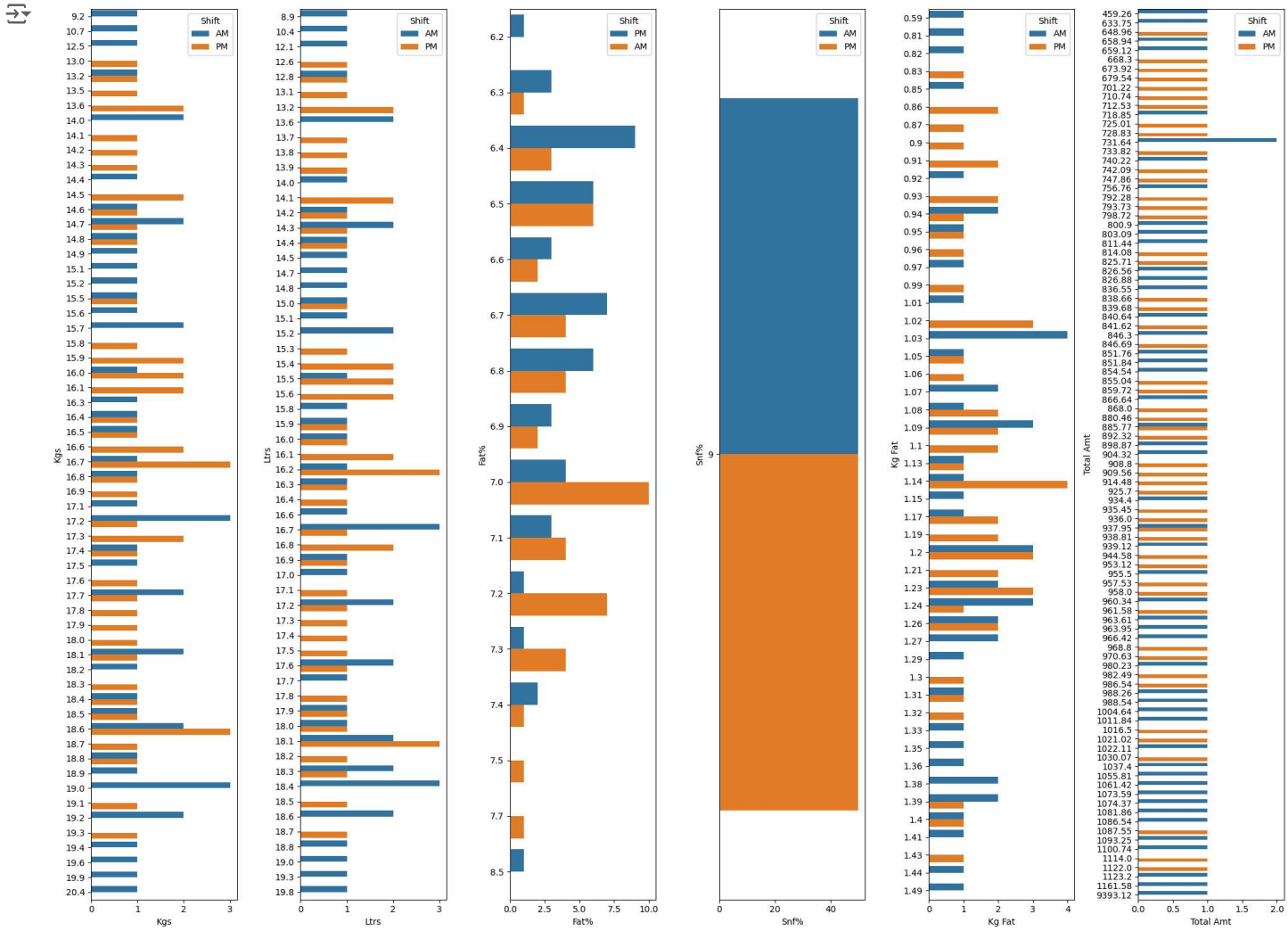



```
# checking numerical features distribution

plt.figure(figsize = (20, 15))
plotnumber = 1
```

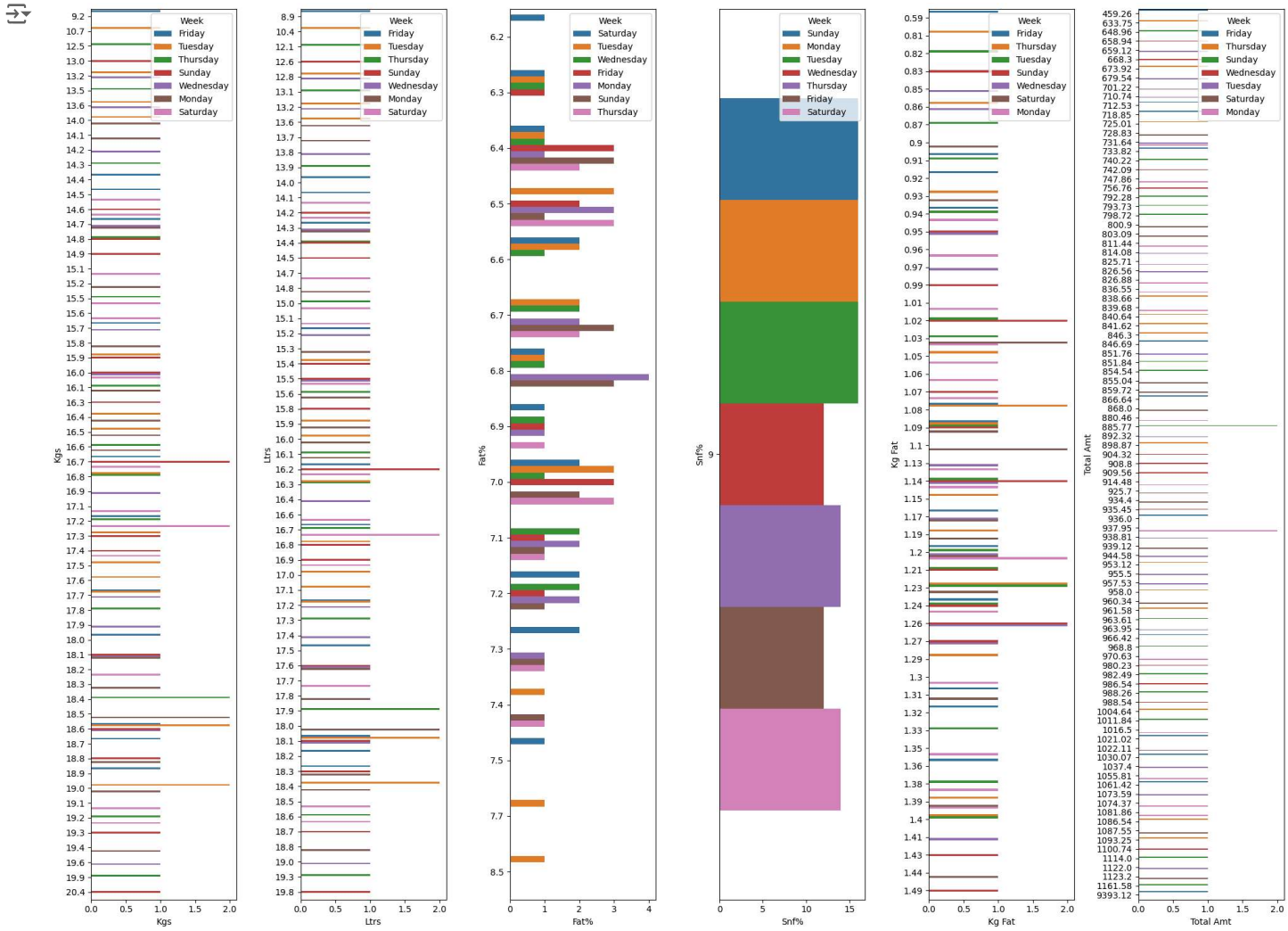
```
num_cols=['Kgs','Ltrs','Fat%','Snf%','Kg Fat','Total Amt']

for column in num_cols:
    if plotnumber <= len(num_cols):
        ax = plt.subplot(6,1, plotnumber)
        sns.countplot(df,x=column)
        plt.xlabel(column)

    plotnumber += 1

plt.tight_layout()
plt.show()
```



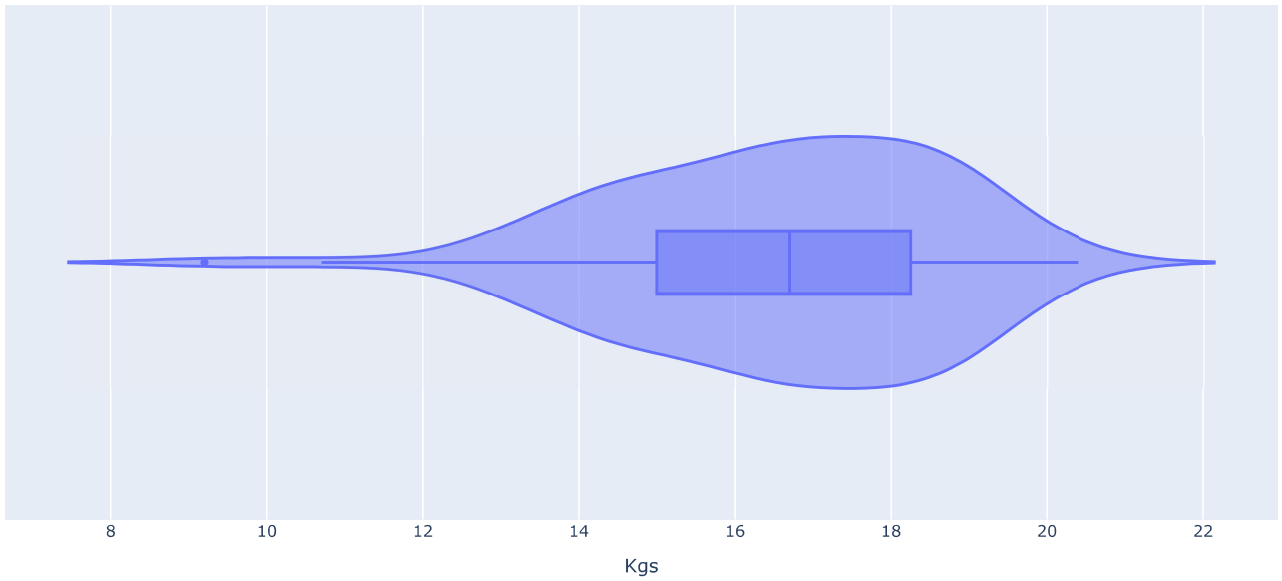

```
# checking numerical features distribution

plt.figure(figsize = (20, 15))
plotnumber = 1
num_cols=['Kgs','Ltrs','Fat%','Snf%','Kg Fat','Total Amt']

for column in num_cols:
    if plotnumber <= len(num_cols):
        ax = plt.subplot(1,6, plotnumber)
        sns.countplot(df,y=column,hue='Shift')
        plt.xlabel(column)

    plotnumber += 1

plt.tight_layout()
plt.show()
```

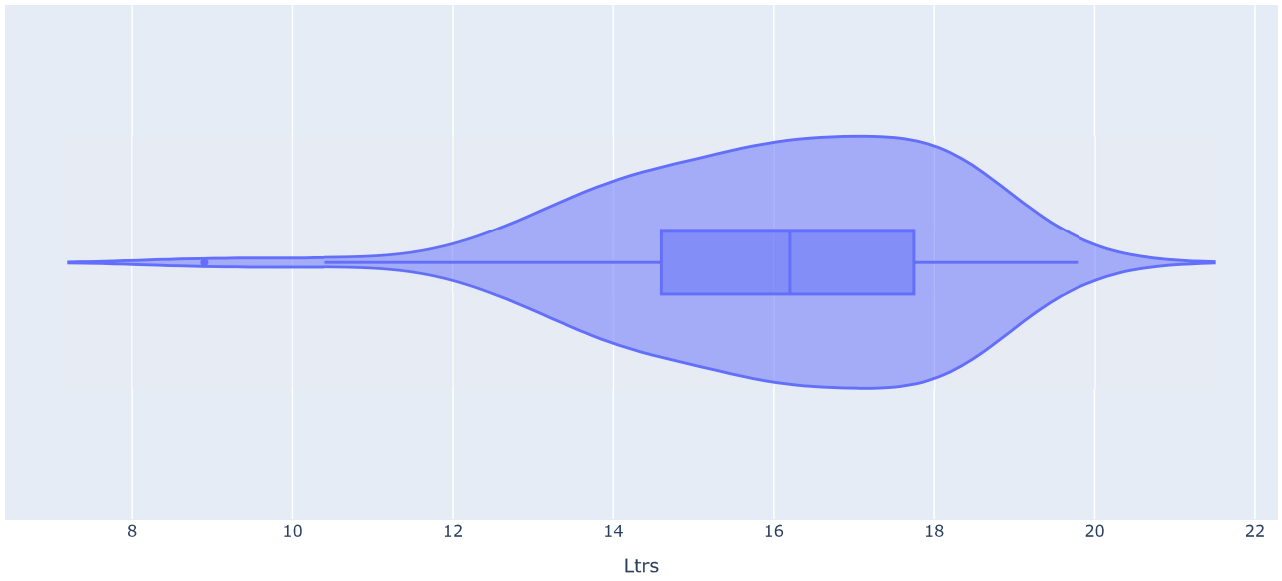


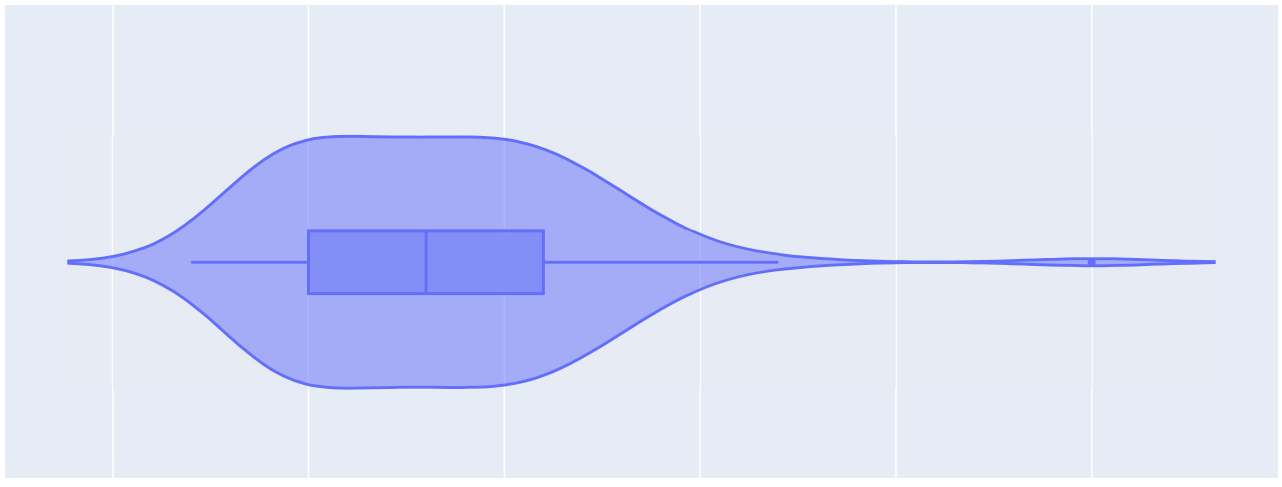

```
# checking numerical features distribution

plt.figure(figsize = (20, 15))
plotnumber = 1
num_cols=['Kgs','Ltrs','Fat%','Snf%','Kg Fat','Total Amt']

for column in num_cols:
    if plotnumber <= len(num_cols):
        ax = plt.subplot(1,6, plotnumber)
        sns.countplot(df,y=column,hue='Week')
        plt.xlabel(column)

    plotnumber += 1

plt.tight_layout()
plt.show()
```

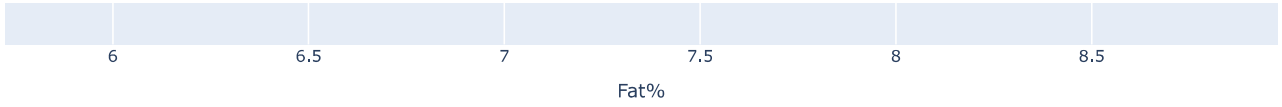



```
for i in num_cols:
  fig=px.violin(df,x=i,box=True)
  fig.show()
```

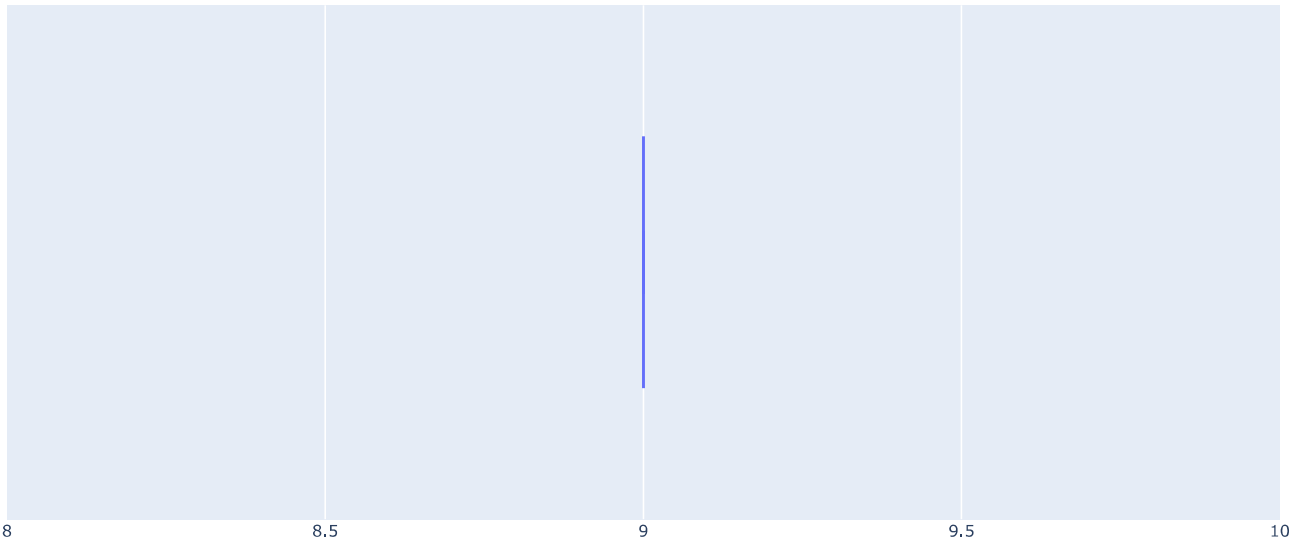

8
10
12
14
16
18
20
22
Kgs


8
10
12
14
16
18
20
22
Ltrs

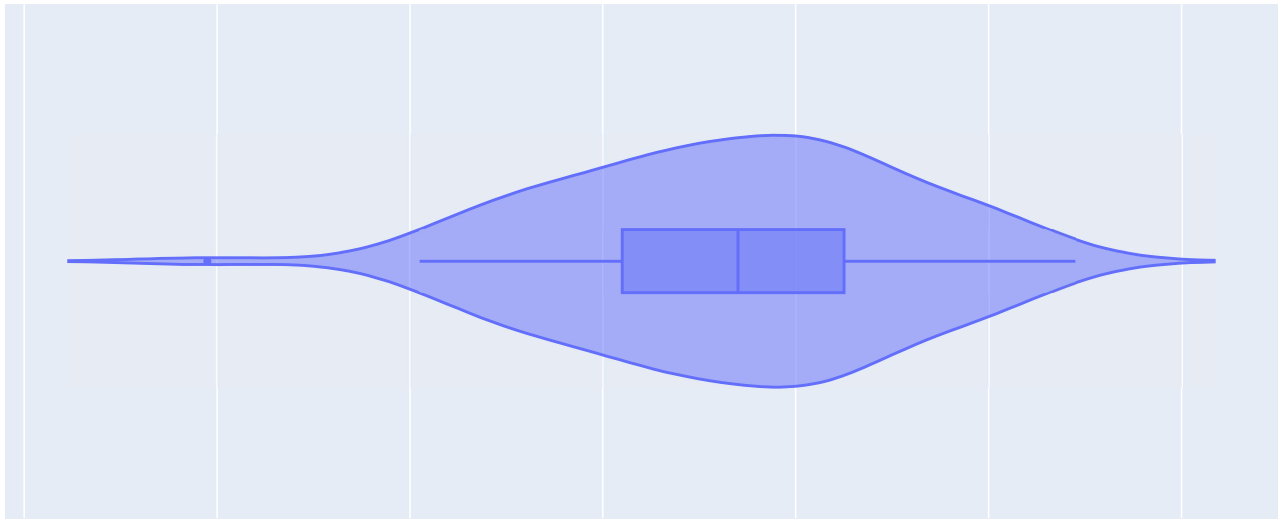




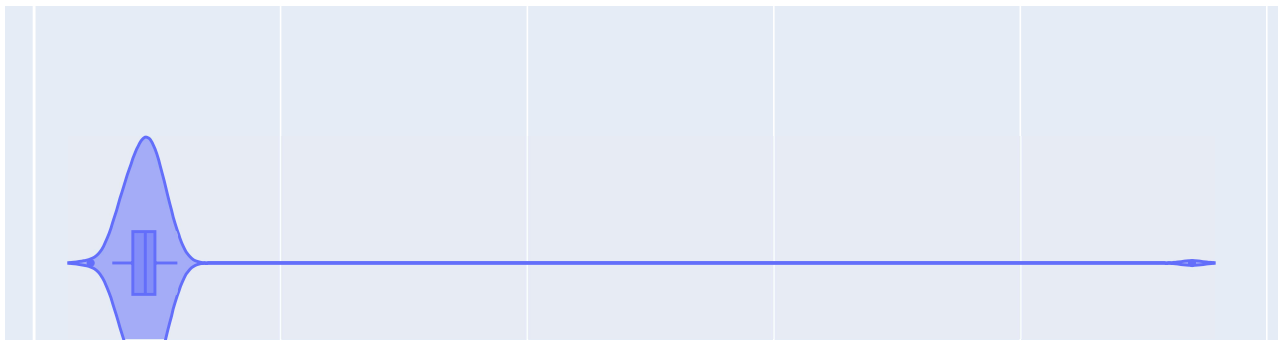
Fat%

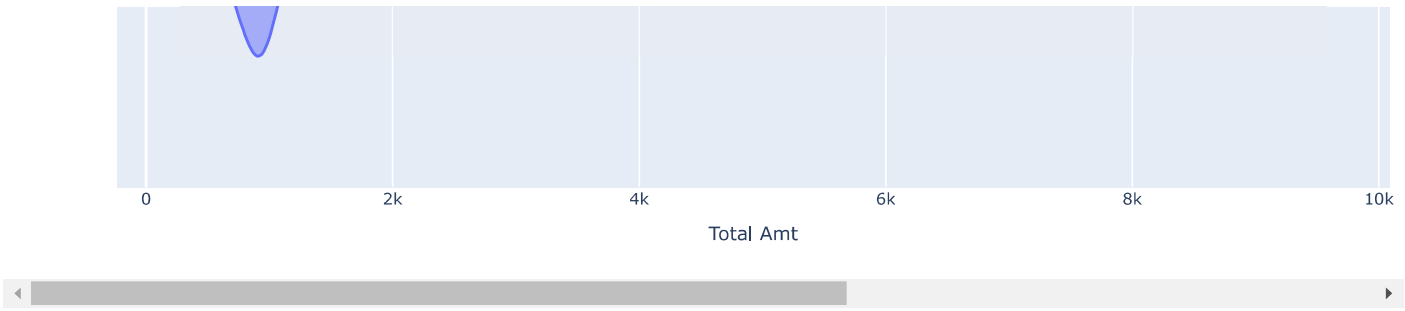


Snf%

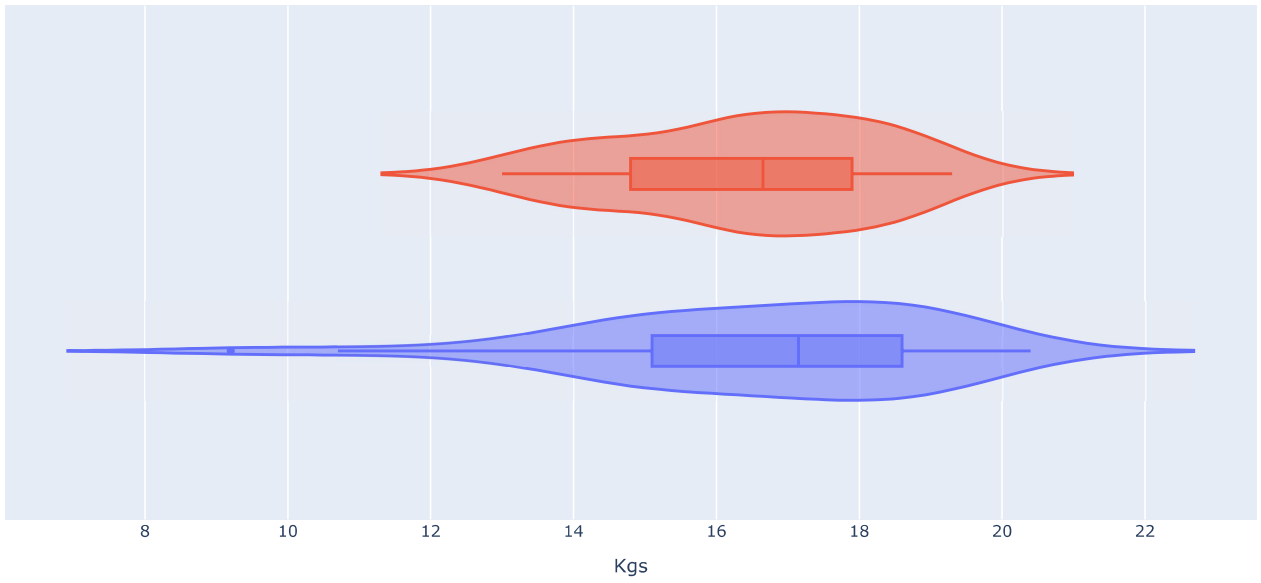


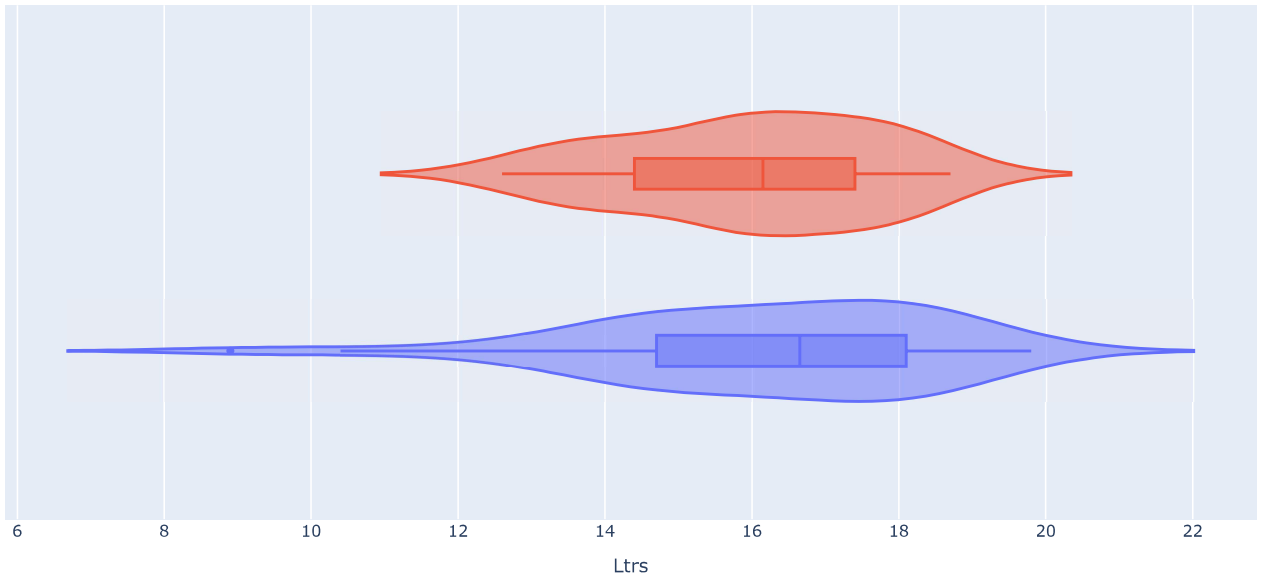
Kg Fat

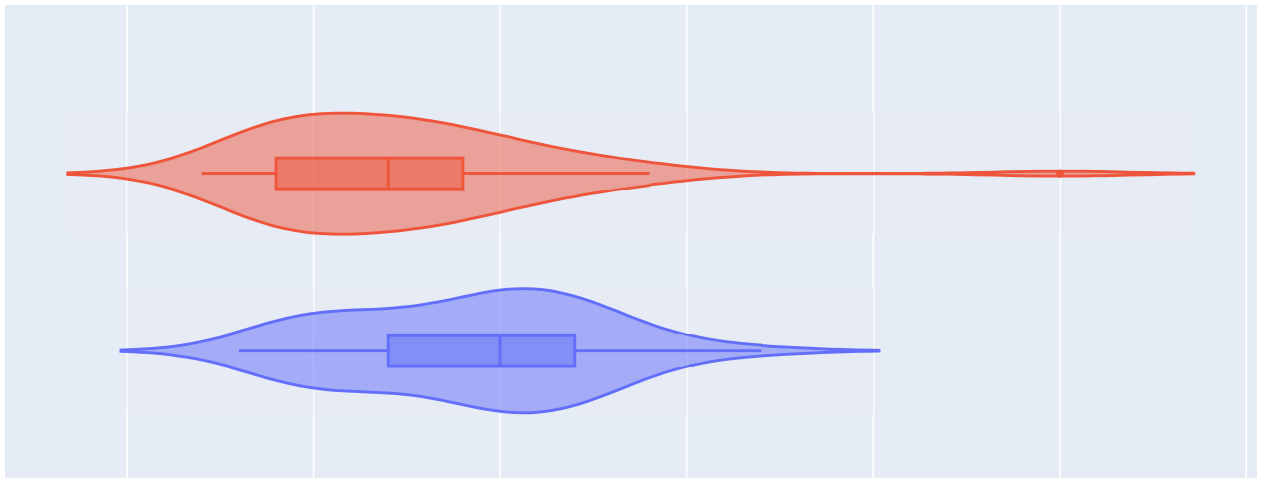



0
2k
4k
6k
8k
10k
Total Amt

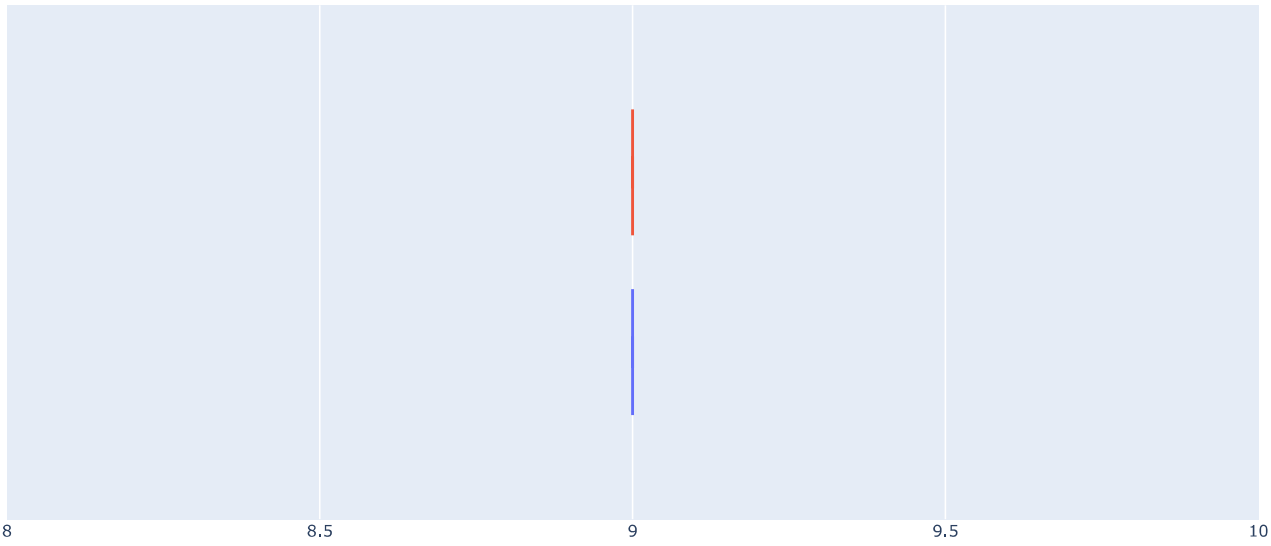
```
for i in num_cols:
  fig=px.violin(df,x=i,color='Shift',box=True)
  fig.show()
```


8
10
12
14
16
18
20
22
Kgs

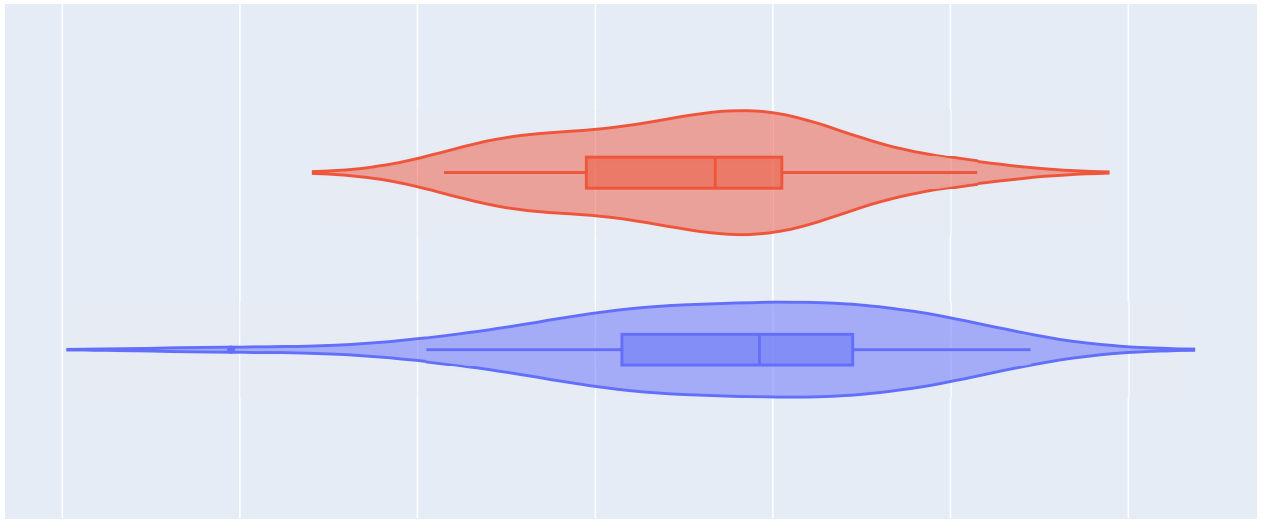

6
8
10
12
14
16
18
20
22
Ltrs



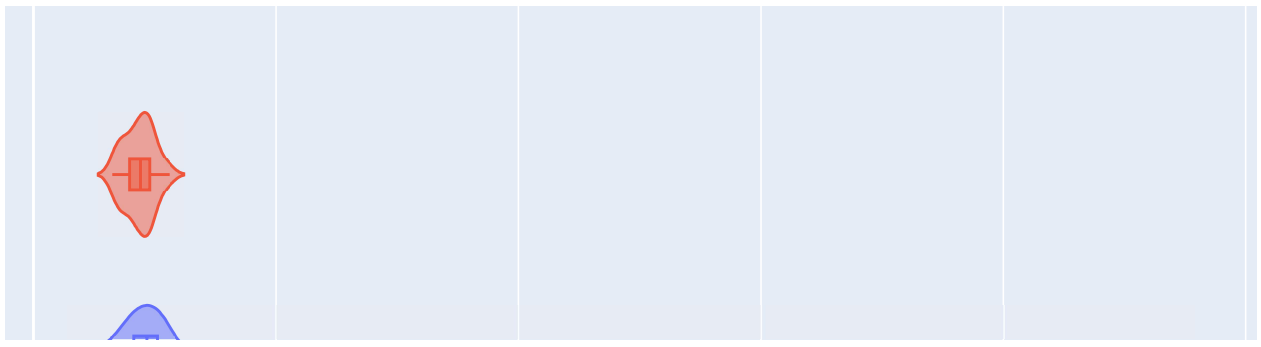
Fat%

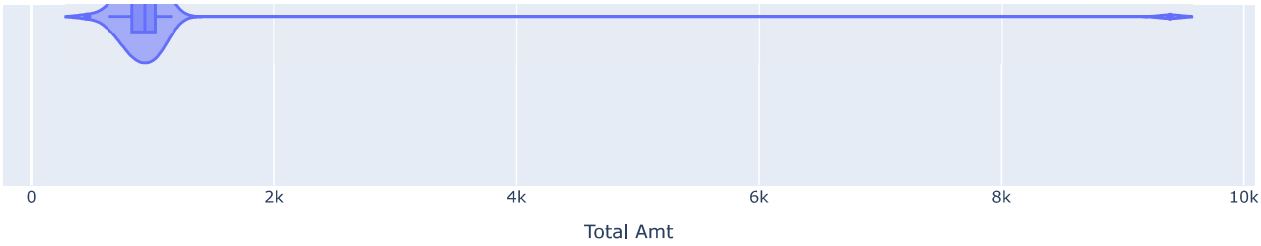


Snf%



Kg Fat

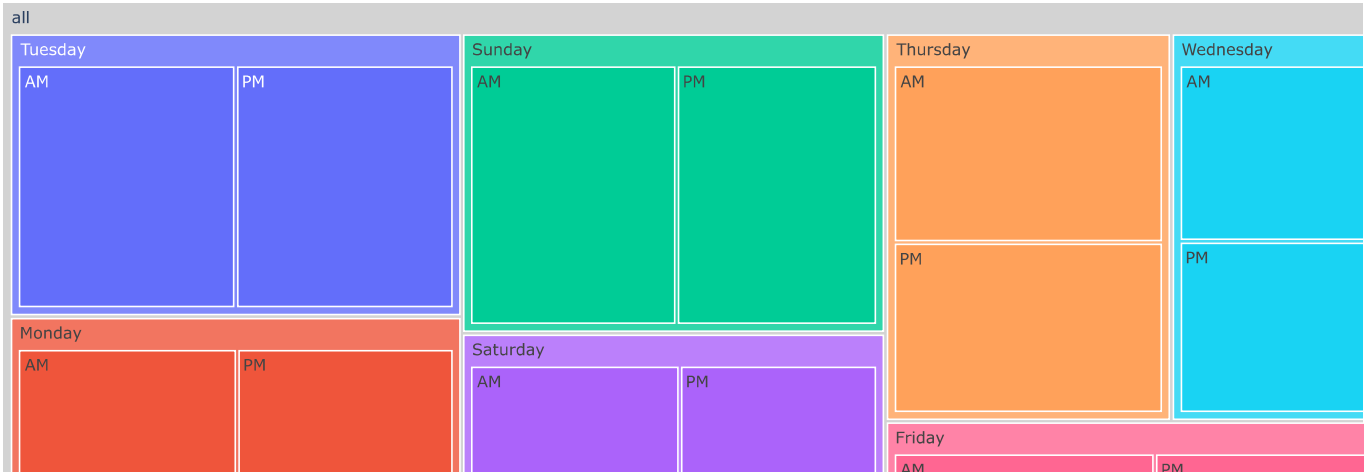



0
2k
4k
6k
8k
10k
Total Amt

```
fig = px.treemap(df, path=[px.Constant("all"), 'Week', 'Shift'], values='Total Amt')
fig.update_traces(root_color="lightgrey")
fig.update_layout(margin = dict(t=50, l=25, r=25, b=25))
fig.show()
```


```
fig = px.treemap(df, path=[px.Constant("all"), 'Week', 'Shift'], values='Kgs')
fig.update_traces(root_color="lightgrey")
fig.update_layout(margin = dict(t=50, l=25, r=25, b=25))
fig.show()
```


```
fig = px.treemap(df, path=[px.Constant("all"), 'Week', 'Shift'], values='Fat%')
fig.update_traces(root_color="lightgrey")
fig.update_layout(margin = dict(t=50, l=25, r=25, b=25))
fig.show()
```


all
Tuesday
AM
PM
Monday
AM
PM
Sunday
AM
PM
Saturday
AM
PM
Thursday
AM
PM
Wednesday
AM
PM
Friday