In [1]:
```
from plotnine import *
```

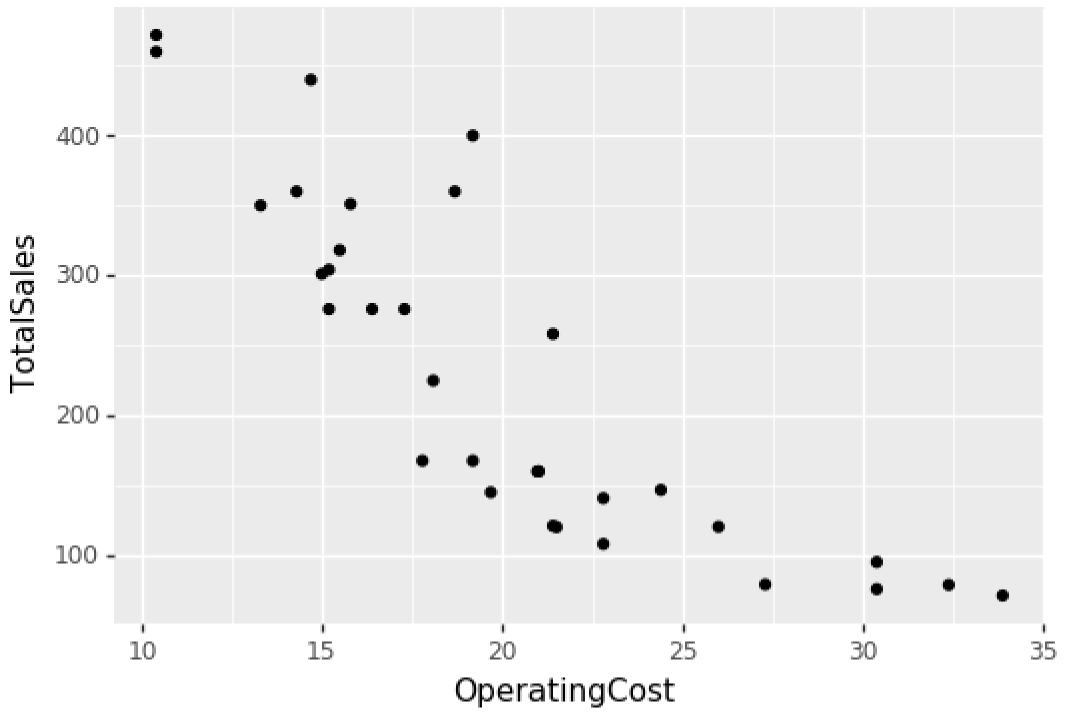
```
C:\Users\admin\Anaconda2\lib\site-packages\statsmodels\compat\pandas.py:56: Fut
ureWarning: The pandas.core.datetools module is deprecated and will be removed
in a future version. Please use the pandas.tseries module instead.
  from pandas.core import datetools
```

In [2]:
```
import pandas as pd
```

In [3]:
```
stores = pd.read_csv("C:/Users/admin/pandas/DataSets/stores.csv")
```

```
#----------------------------------------
# Aim : To get a scatterplot
#----------------------------------------
# Every ggplot2 plot has three key components:
# 1. data,
# 2. A set of aesthetic mappings between variables in the data and visual
# properties, and
# 3. At least one layer which describes how to render each observation. Layers
# are usually created with a geom function.

# Plot2 <- ggplot(stores, aes(x = OperatingCost, y = TotalSales))+ geom_point()
```

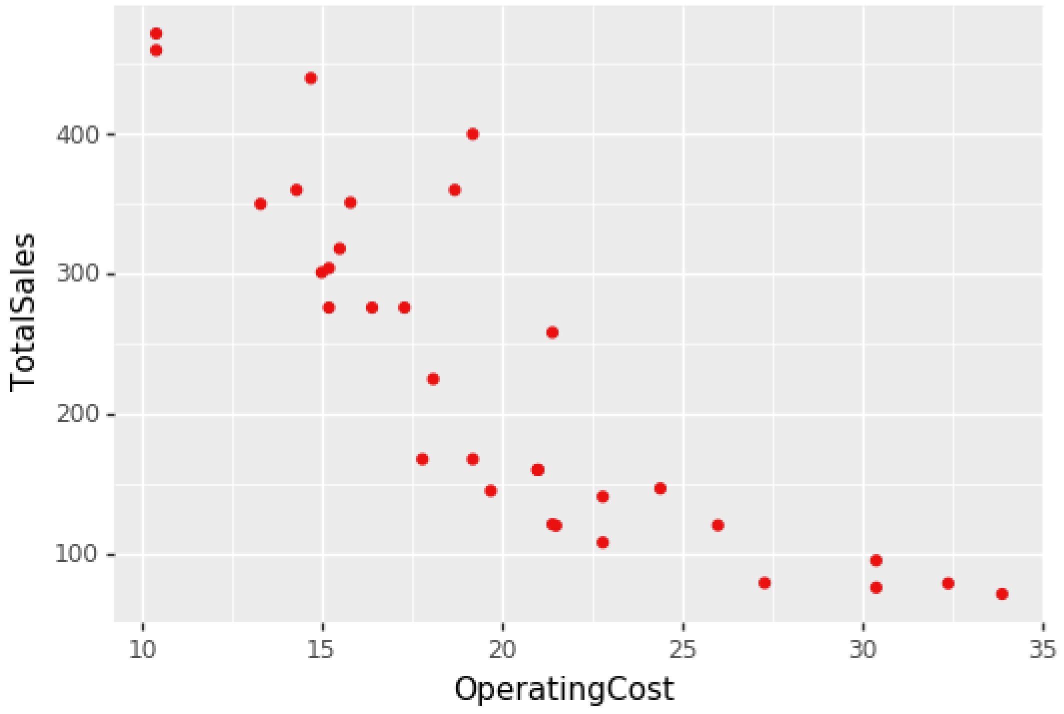
In [4]:
```
(
    ggplot(stores) +
    aes(x = "OperatingCost",y = "TotalSales") +
    geom_point()
)
```



Out[4]:
```
<ggplot: (19847561)>
```

```
# Adding colors
Plot3 <- ggplot(stores,aes(x = OperatingCost, y = TotalSales))
Plot3 <- Plot3 + geom_point(color = "green")
Plot3

# https://www.hexcolortool.com/
```

In [5]:
```
(
    ggplot(stores)
 + aes(x = 'OperatingCost', y = 'TotalSales')
 + geom_point(color = "#ea1010")
)

#ggplot(stores,aes(x = 'OperatingCost', y = 'TotalSales'))
# that's the RGBA notation for colors!!
```

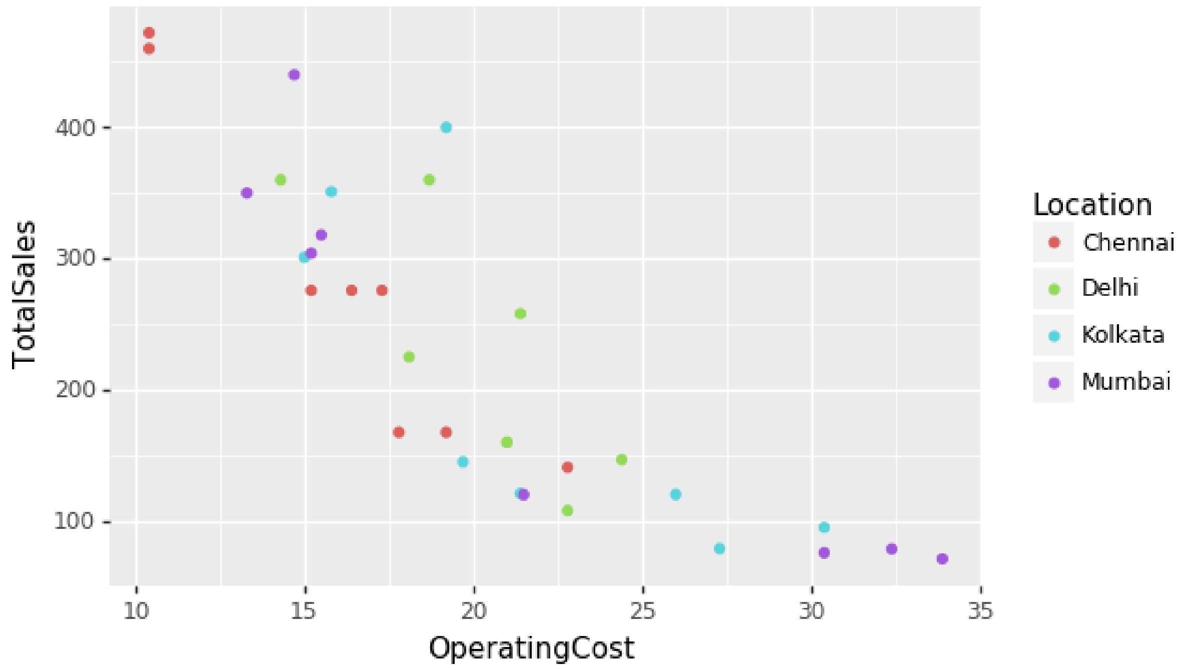


Out[5]:
```
<ggplot: (19923649)>
```

In [14]:
```
# To get colors
# https://www.hexcolortool.com/
```

```
#-------------------
# Adding more variables as colors
#-------------------
# color argument -> ideally takes a categorical variable

# different colors are filledin the graph acc to the levels
Plot5 <- ggplot(stores,aes(x = OperatingCost, y = TotalSales, color = Location))
+ geom_point()
```

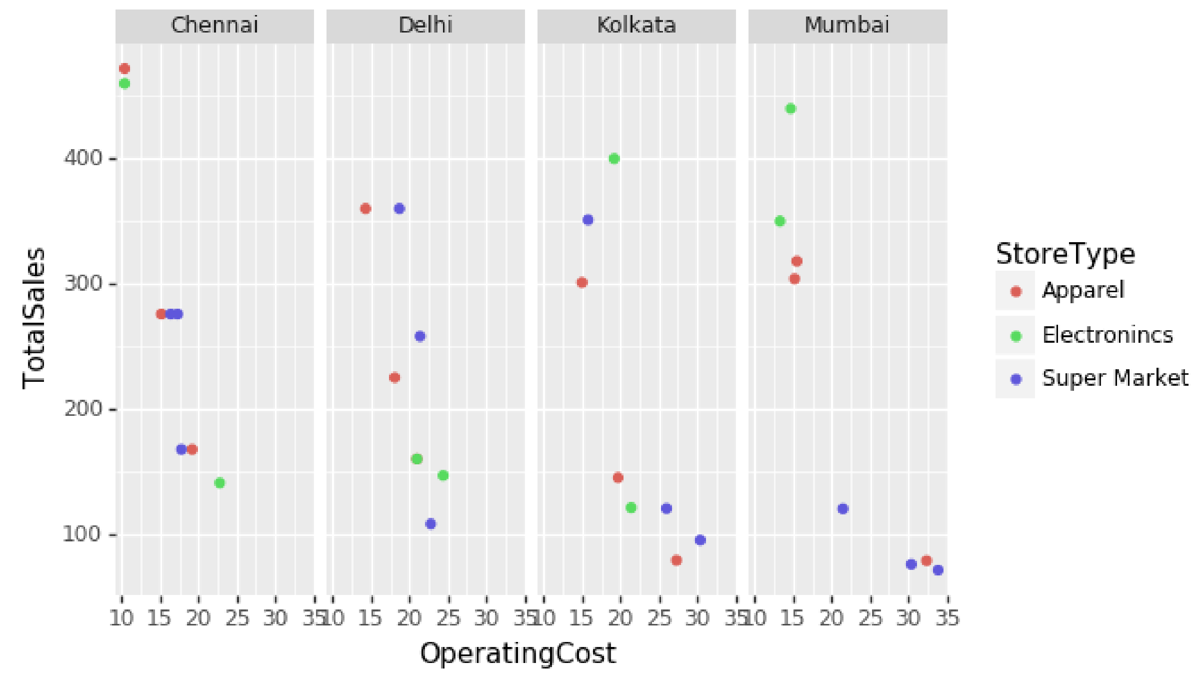
In [6]:
```
(
    ggplot(stores)
 + aes(x = 'OperatingCost', y = 'TotalSales',color = 'Location')
 + geom_point()
)
```



Out[6]:
```
<ggplot: (20075538)>
```

```
#-------------------
# Adding more variables as facets
#-------------------
Plot7 <- ggplot(stores,aes(x = OperatingCost, y = TotalSales)) +
  geom_point() +
  facet_wrap(~Location)

Plot7_1 <- ggplot(stores,aes(x = OperatingCost, y = TotalSales)) +
  geom_point() +
  facet_grid(Location ~ .)
#[r,c]
# r ~ c
# . ~ Location
# . for nothing under that section in facet
```

In [8]:
```
(ggplot(stores,aes(x = "OperatingCost", y = "TotalSales",color = "StoreType")) +
  geom_point() +
  facet_grid(". ~ Location")
)
```

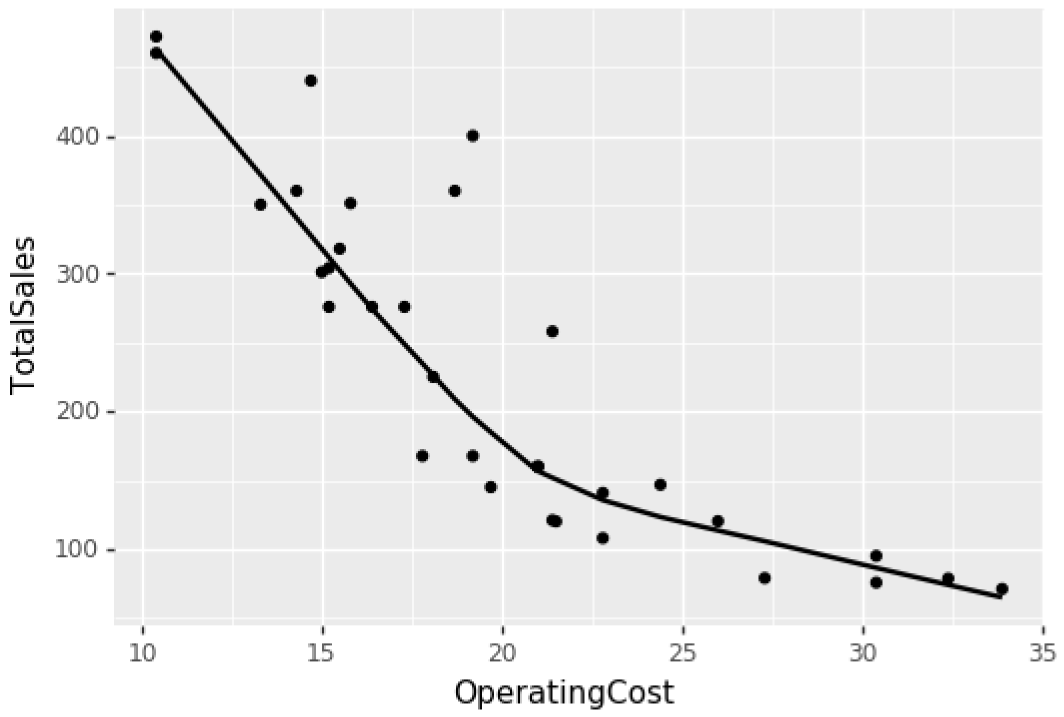


Out[8]: `<ggplot: (22290271)>`

```
#-----------------------
# More Plot geoms
#-----------------------
# 1. Smoothing Curve
ggplot(stores, aes(OperatingCost, TotalSales)) +
  geom_point() +
  geom_smooth()
```

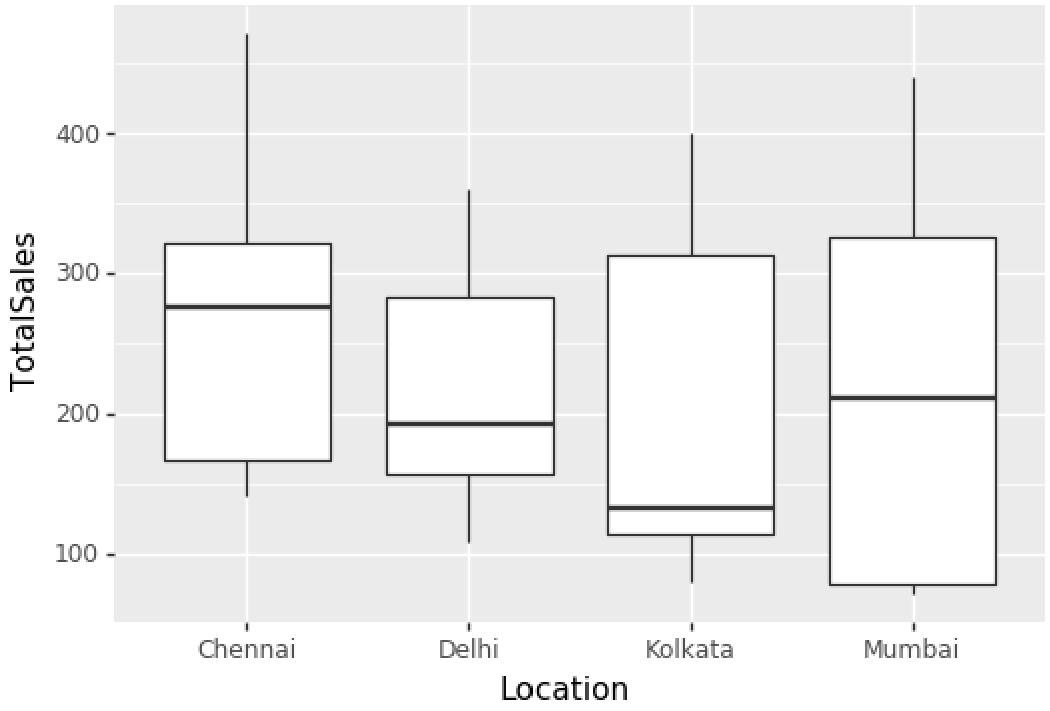
```
In [15]: (ggplot(stores, aes(x = "OperatingCost", y = "TotalSales")) +
           geom_point() +
           geom_smooth())
```

```
C:\Users\admin\Anaconda2\lib\site-packages\plotnine\stats\smoothers.py:150: Use
rWarning: Confidence intervals are not yet implementedfor lowess smoothings.
  warnings.warn("Confidence intervals are not yet implemented"
```



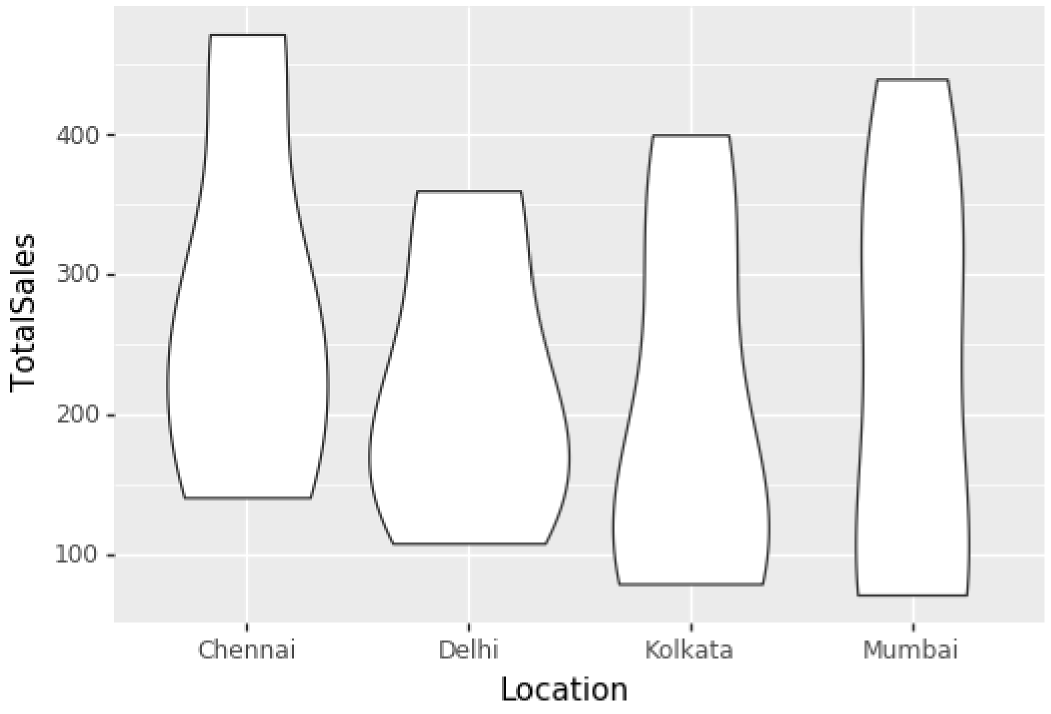
```
Out[15]: <ggplot: (24600566)>
```

```
In [20]: # 2. Whisker plots for distributions
         (ggplot(stores, aes(y = "TotalSales", x = "Location"))
          +  geom_boxplot()
         )
```



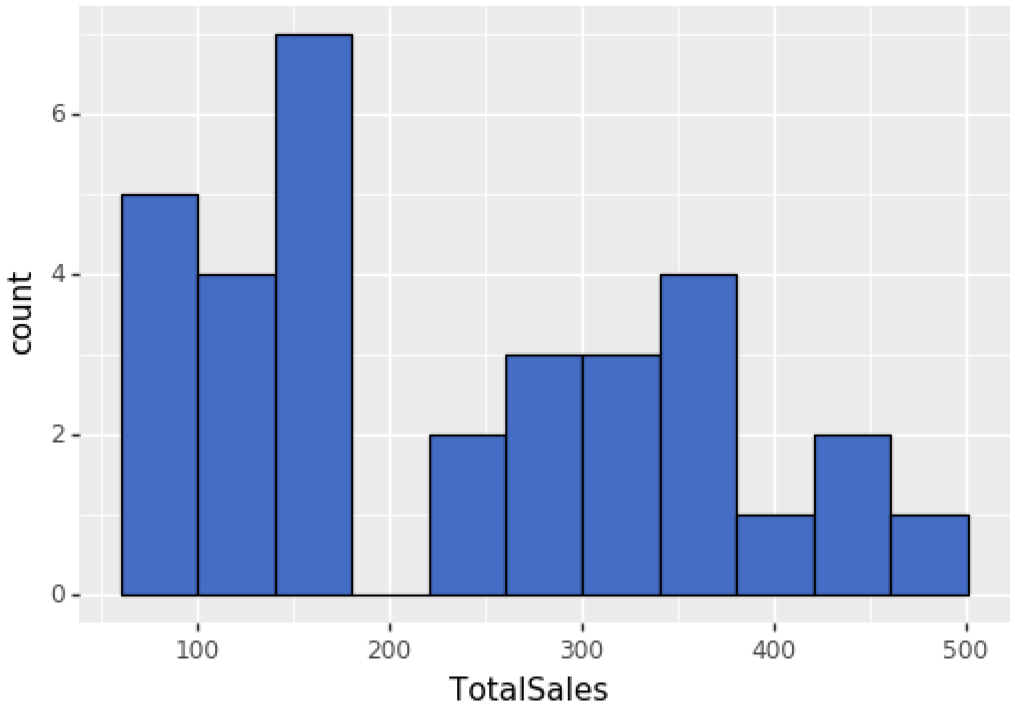
```
Out[20]: <ggplot: (20644580)>
```

In [34]:
```
# 3. Violin plots for density distribution
#   Kind of box plots but they show density -
#   areas where more data points are found

(ggplot(stores, aes(y = "TotalSales", x = "Location")) + geom_violin())
```



```
Out[34]: <ggplot: (23075900)>
```

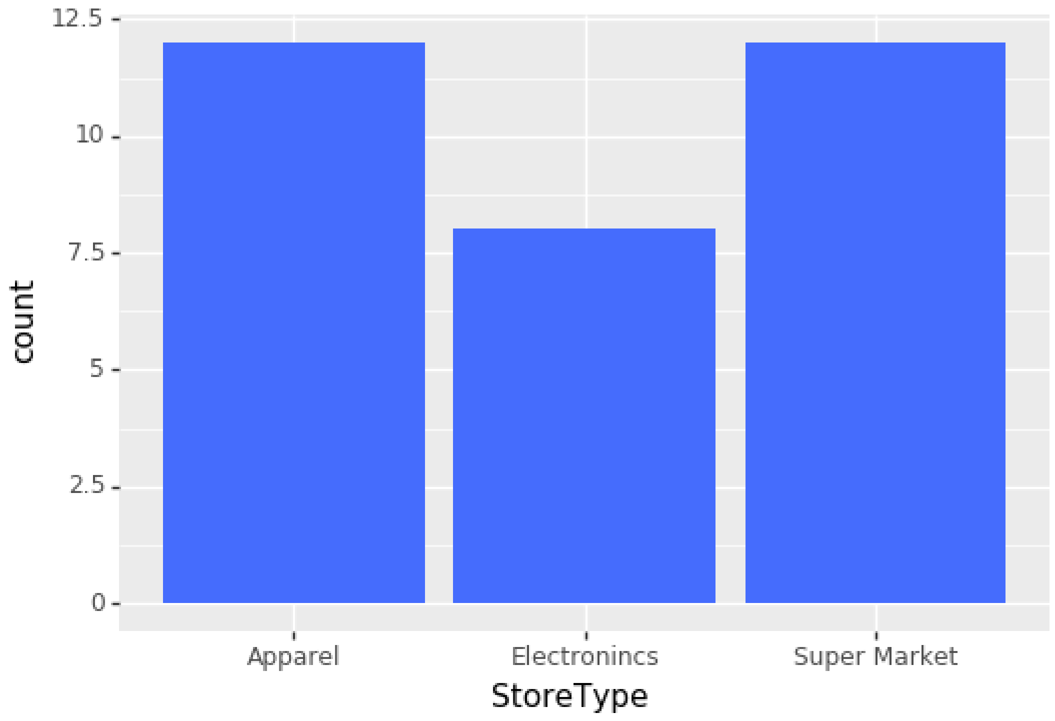
In [23]:
```
# 4. Histograms

(ggplot(stores, aes("TotalSales")) + geom_histogram(bins = 11,color = "#000000",f

# color here is for border color
# fill for the bar
```



Out[23]:
```
<ggplot: (22708758)>
```

```
#-------------------------------
# 5. Bar Plots
#-------------------------------
# by default, a geom_bar() amounts to count or stat = "bins"
```

In [71]:
```
(ggplot(stores, aes("StoreType")) +
  geom_bar(fill = "#456cfd") # some shade of blue..
)
```



Out[71]:
```
<ggplot: (23391222)>
```

```
# For pre summarized data, like a pivot kind of information,
# use stat = "identity"

td <- group_by(stores, Location)
Res <- dplyr::summarize(td, Sum.of.TotalSales = sum(TotalSales))

Bar1 <- ggplot(data = Res,aes(x = Location, y = Sum.of.TotalSales))
Bar1 <- Bar1 + geom_bar(stat = "identity")
Bar1
```

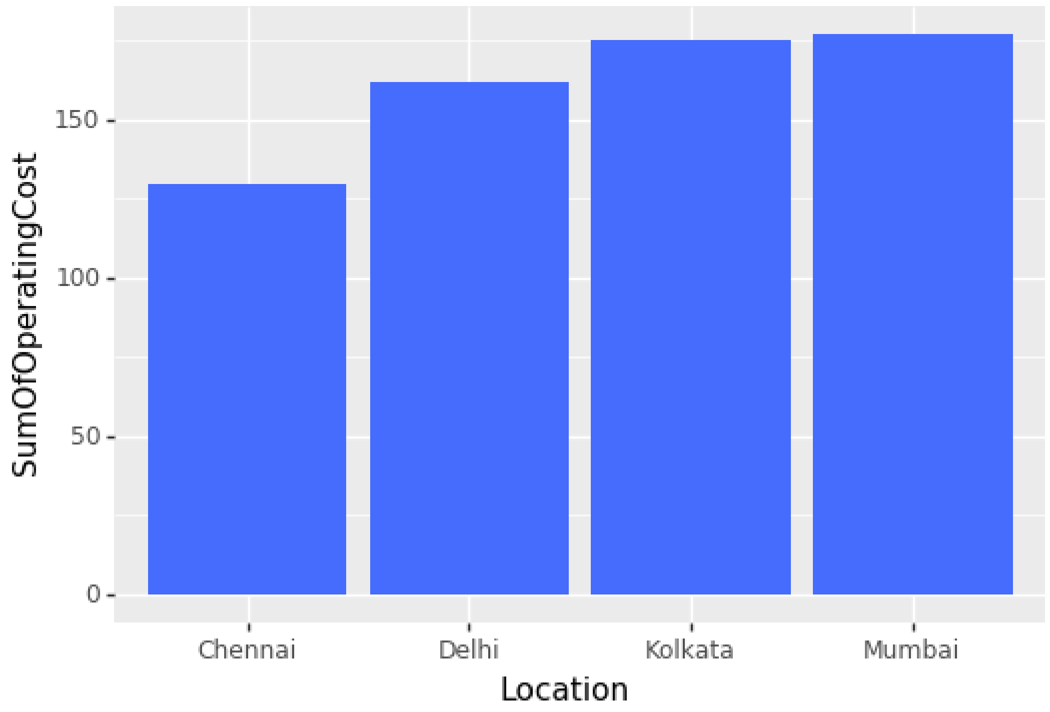
In [10]:
```
# Location based SumOfTotalSales
t = stores.groupby('Location')
Result1 = t.OperatingCost.agg({"SumOfOperatingCost":"sum"})
Result1 = Result1.reset_index()
Result1
```

```
C:\Users\admin\Anaconda2\lib\site-packages\ipykernel\__main__.py:3: FutureWarni
ng: using a dict on a Series for aggregation
is deprecated and will be removed in a future version
  app.launch_new_instance()
```

Out[10]:

	Location	SumOfOperatingCost
0	Chennai	129.5
1	Delhi	161.7
2	Kolkata	174.8
3	Mumbai	176.9

In [14]:
```
(
    ggplot(Result1) + aes(x = "Location", y = "SumOfOperatingCost") +
geom_bar(fill = "#456cfd",stat = "identity")

)

# color -> border color
# fill -> color of the bar
```



Out[14]: <ggplot: (23120434)>

```
# Add more variables through colors in aes
Res2 <- stores %>% group_by(Location,StoreType) %>%
  dplyr::summarize(Sum.of.TotalSales = sum(TotalSales))
```

```
td <- group_by(stores, Location, StoreType)
Res2 <- dplyr::summarize(td, Sum.of.TotalSales = sum(TotalSales))



# Multivariables in any bar graph can be : stacked or dodged

# 1. Stacked

Bar2 <- ggplot(Res2, aes(x = Location, y = Sum.of.TotalSales, fill = StoreType))
Bar2 <- Bar2 + geom_bar(stat = "identity")
Bar2

ggplotly(Bar2)

# 2. Dodged


Bar2d <- ggplot(Res2, aes(x = Location, y = Sum.of.TotalSales, fill =
StoreType))
Bar2d <- Bar2d + geom_bar(stat = "identity",position = "dodge")
Bar2d
```

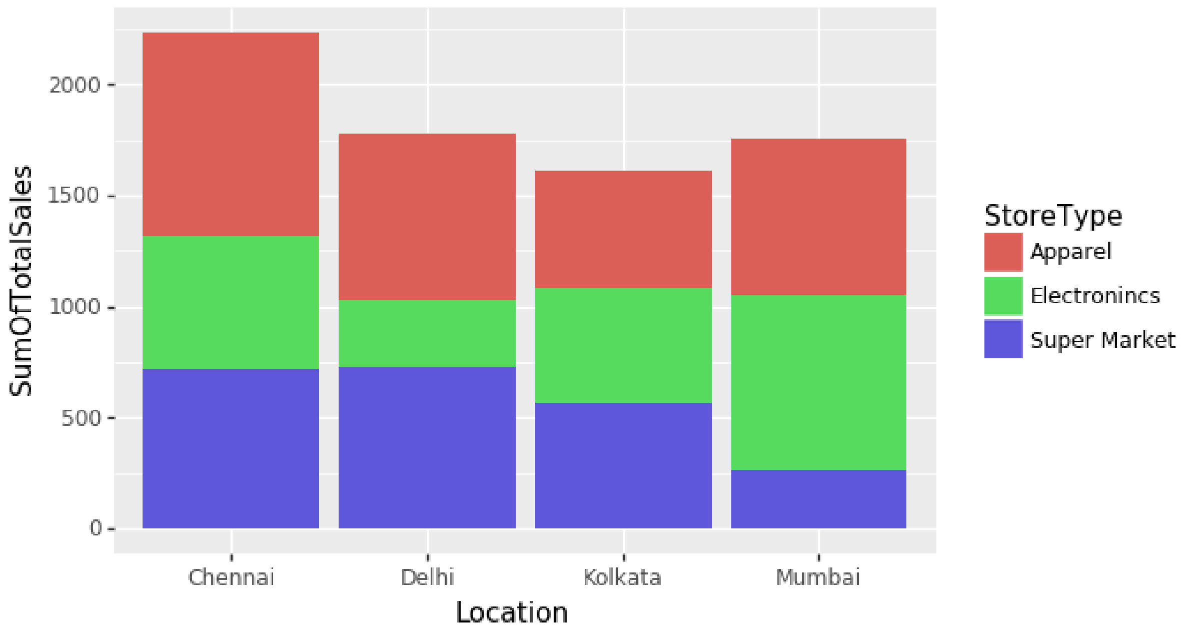
In [15]:
```
temp = stores.groupby(["Location","StoreType"])
Result2 = temp["TotalSales"].agg({"SumOfTotalSales":sum})
Result2 = Result2.reset_index()
Result2.to_excel("Result2.xlsx")
```

```
C:\Users\admin\Anaconda2\lib\site-packages\ipykernel\__main__.py:2: FutureWarni
ng: using a dict on a Series for aggregation
is deprecated and will be removed in a future version
  from ipykernel import kernelapp as app
```

Out[15]:

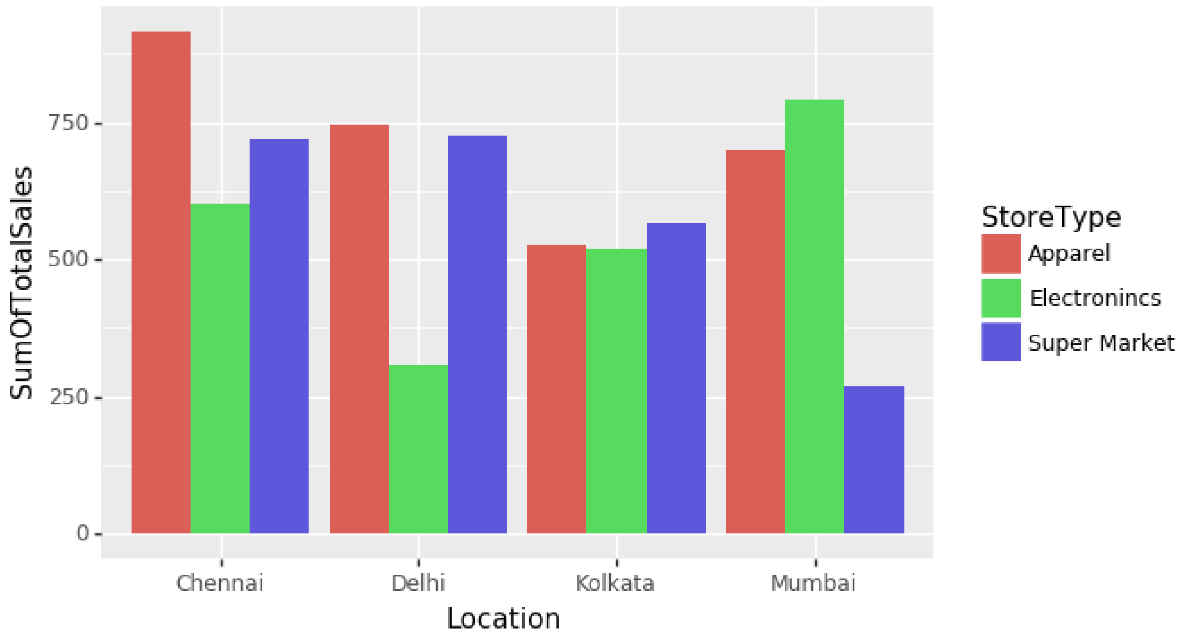
	Location	StoreType	SumOfTotalSales
0	Chennai	Apparel	915.4
1	Chennai	Electronincs	600.8
2	Chennai	Super Market	719.2
3	Delhi	Apparel	745.0
4	Delhi	Electronincs	306.7
5	Delhi	Super Market	726.0
6	Kolkata	Apparel	525.0
7	Kolkata	Electronincs	521.0
8	Kolkata	Super Market	566.4
9	Mumbai	Apparel	700.7
10	Mumbai	Electronincs	790.0
11	Mumbai	Super Market	266.9

```
In [16]: # 1. Stacked
         (ggplot(Result2, aes(x = "Location", y = "SumOfTotalSales", fill = "StoreType"))
             + geom_bar(stat = "identity")
         )
```



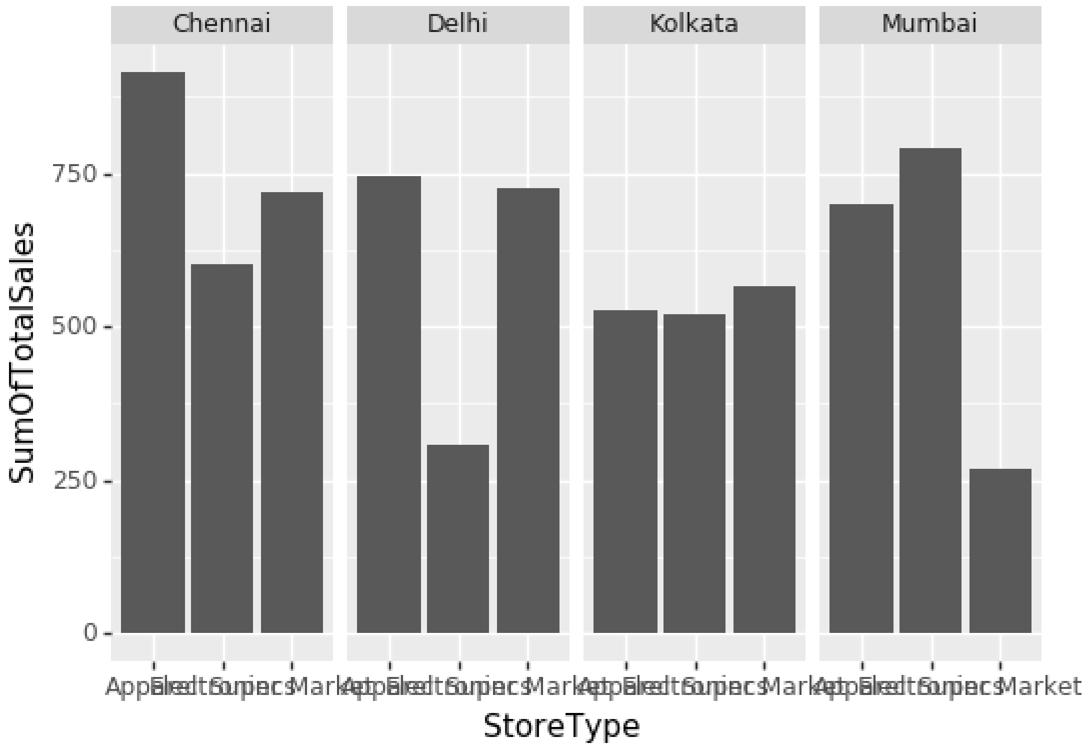
```
Out[16]: <ggplot: (23194565)>
```

```
In [17]: # 2. Dodged
         (ggplot(Result2, aes(x = "Location", y = "SumOfTotalSales", fill = "StoreType"))
         + geom_bar(stat = "identity",position = "dodge")
         )
```



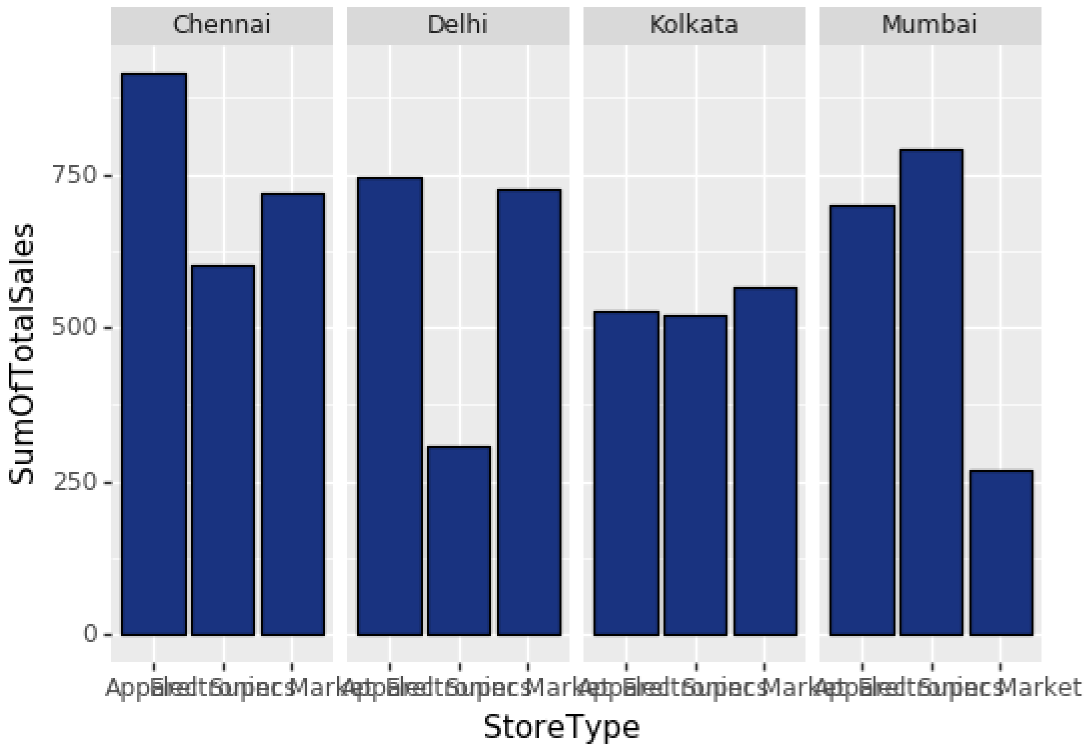
```
Out[17]: <ggplot: (23298515)>
```

In [21]:
```
(           ggplot(Result2)
            + aes(x = "StoreType", y = "SumOfTotalSales")
            + geom_bar(stat = "identity")
            + facet_grid(". ~ Location"))
```



Out[21]: `<ggplot: (23135162)>`

In [75]:
```
# Add more variables through facets


(ggplot(Result2)
 + aes(x = "StoreType",y = "SumOfTotalSales")
 + geom_bar(stat = "identity",fill = "#193380",color = "#000000")
 + facet_grid(".~Location")
)
```



Out[75]:
```
<ggplot: (22692633)>
```

```
#-----------------------------------------------
# Modifying the axes
#-----------------------------------------------

# 1. Labeling the axes

Plot9 <- ggplot(stores, aes(y = TotalSales, x = OperatingCost)) +
  geom_point() +
  xlab("Operating Cost") +
  ylab("Total Sales")
Plot9

# Remove the axis labels with NULL
ggplot(stores, aes(y = TotalSales, x = OperatingCost)) +
  geom_point() +
  xlab(NULL) +
  ylab(NULL)

# Changing the limits of the axes
# use xlim() for x
# ylim() for y
```

```
Plot9lim <- ggplot(stores, aes(y = TotalSales, x = OperatingCost)) +
  geom_point() +
  xlab("Operating Cost") +
  ylab("Total Sales") +
  xlim(min(stores$OperatingCost),50)
Plot9lim

# 2. Adding text

Plot10 <- ggplot(data = stores,aes(x = OperatingCost, y  = TotalSales))
Plot10 <- Plot10 + geom_point()
Plot10 <- Plot10 + geom_text(aes(label =
paste("\n",OperatingCost,",",TotalSales)))
Plot10
ggplotly(Plot10)


# labels : for the main labels
# family : for the font family. Serif, Mono or Sans
# fontface : bold, ita, plain
```