In [1]:
```
## Required Libraries
#  pip3 install panda
#  pip3 install pymysql
#  pip3 install sqlalchemy

import pandas as pd
import pymysql
from sqlalchemy import create_engine as ce
```

In [4]:
```
## Connecting to MH6142 database,
## [NOTE] that the Password is masked
sqlengine = ce('mysql+pymysql://root:******@localhost:3306/MH6142')
```

In [5]:
```
## List all available tables from MH6142
sql_showtables = '''
    show tables
    '''
df_showtables = pd.read_sql_query(sql_showtables, sqlengine)
print(df_showtables)
```

```
    Tables_in_mh6142
0  global_superstore
```

In [5]:
```
## Load the global_superstore.xlsx into db
df_globalsuperstore = pd.read_excel("global_superstore.xlsx", "Orders",)
print(df_globalsuperstore)

df_globalsuperstore.to_sql('global_superstore', sqlengine, index=False, if_exists='fail')
```

```
       Row ID         Order ID Order Date  Ship Date       Ship Mode  \
0       32298   CA-2012-124891 2012-07-31 2012-07-31        Same Day
1       26341    IN-2013-77878 2013-02-05 2013-02-07    Second Class
2       25330    IN-2013-71249 2013-10-17 2013-10-18     First Class
3       13524  ES-2013-1579342 2013-01-28 2013-01-30     First Class
4       47221     SG-2013-4320 2013-11-05 2013-11-06        Same Day
5       22732    IN-2013-42360 2013-06-28 2013-07-01    Second Class
6       30570    IN-2011-81826 2011-11-07 2011-11-09     First Class
7       31192    IN-2012-86369 2012-04-14 2012-04-18  Standard Class
8       40155   CA-2014-135909 2014-10-14 2014-10-21  Standard Class
9       40936   CA-2012-116638 2012-01-28 2012-01-31    Second Class
10      34577   CA-2011-102988 2011-04-05 2011-04-09    Second Class
11      28879    ID-2012-28402 2012-04-19 2012-04-22     First Class
12      45794     SA-2011-1830 2011-12-27 2011-12-29    Second Class
13       4132   MX-2012-130015 2012-11-13 2012-11-13        Same Day
14      27704    IN-2013-73951 2013-06-06 2013-06-08    Second Class
15      13779  ES-2014-5099955 2014-07-31 2014-08-03    Second Class
16      36178   CA-2014-143567 2014-11-03 2014-11-06    Second Class
17      12069  ES-2014-1651774 2014-09-08 2014-09-14  Standard Class
```

In [41]:
```
## Define the basic sql queries that we want to execute later on
sql_desc_query = '''
    desc global_superstore;
'''

sql_count_query = '''
    select count(*) from global_superstore;
'''

sql_selectall_query = '''
    select * from global_superstore;
    '''
```

In [42]:
```
## Execute the defined sql queries
df_desc = pd.read_sql_query(sql_desc_query, sqlengine)
print(df_desc)
```

```
             Field        Type Null Key Default Extra
0           Row ID  bigint(20)  YES        None
1         Order ID        text  YES        None
2       Order Date    datetime  YES        None
3        Ship Date    datetime  YES        None
4        Ship Mode        text  YES        None
5      Customer ID        text  YES        None
6    Customer Name        text  YES        None
7          Segment        text  YES        None
8             City        text  YES        None
9            State        text  YES        None
10         Country        text  YES        None
11     Postal Code      double  YES        None
12          Market        text  YES        None
13          Region        text  YES        None
14      Product ID        text  YES        None
15        Category        text  YES        None
16    Sub-Category        text  YES        None
17    Product Name        text  YES        None
18           Sales      double  YES        None
19        Quantity  bigint(20)  YES        None
20        Discount      double  YES        None
21          Profit      double  YES        None
22   Shipping Cost      double  YES        None
23  Order Priority        text  YES        None
```

In [43]:
```
## Execute the defined sql queries
df_count = pd.read_sql_query(sql_count_query, sqlengine)
print(df_count)
```

```
   count(*)
0     51290
```

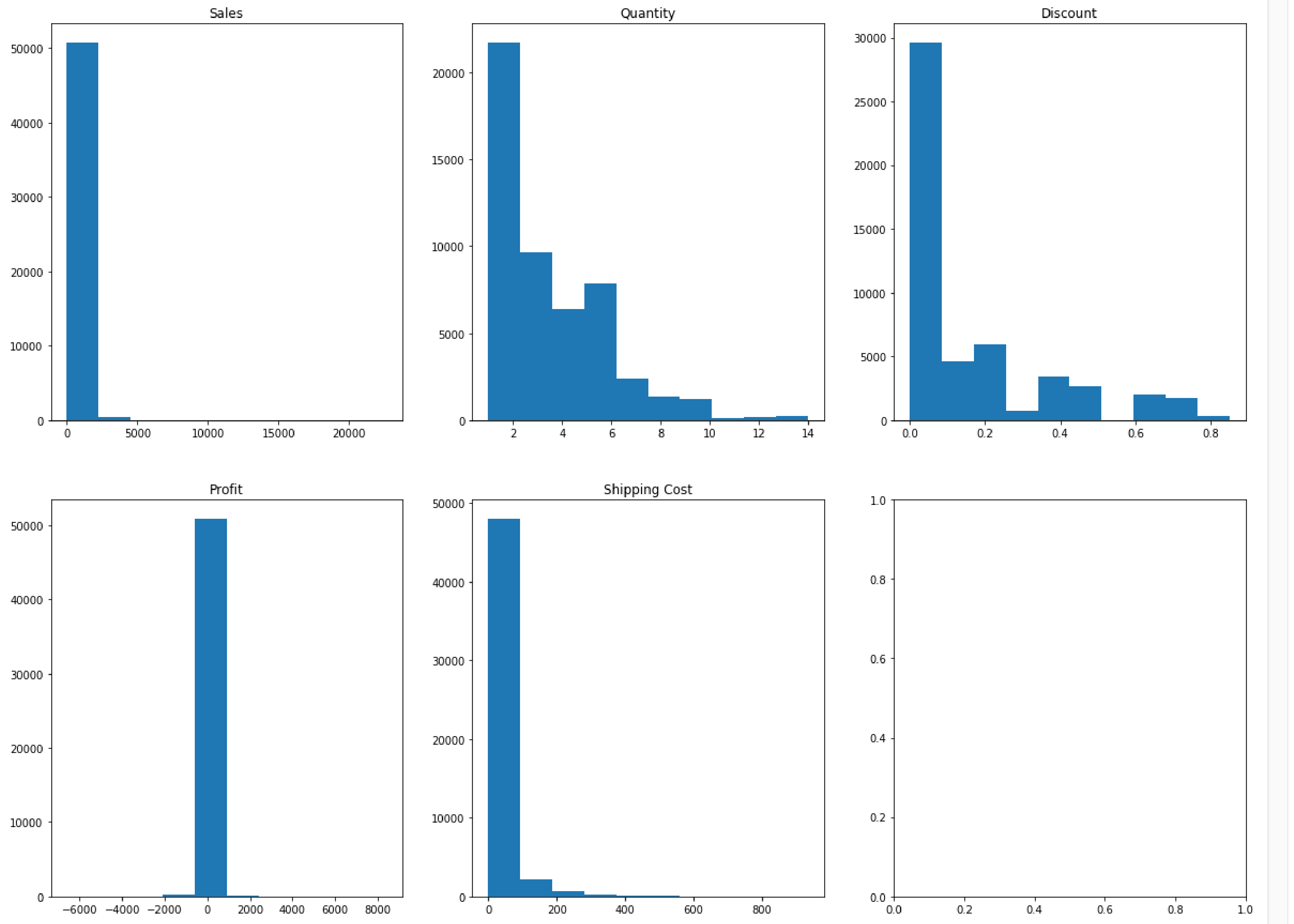
In [44]:
```
## Execute the defined sql queries
df_alldata = pd.read_sql_query(sql_selectall_query, sqlengine)
df_alldata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51290 entries, 0 to 51289
Data columns (total 24 columns):
Row ID            51290 non-null int64
Order ID          51290 non-null object
Order Date        51290 non-null datetime64[ns]
Ship Date         51290 non-null datetime64[ns]
Ship Mode         51290 non-null object
Customer ID       51290 non-null object
Customer Name     51290 non-null object
Segment           51290 non-null object
City              51290 non-null object
State             51290 non-null object
Country           51290 non-null object
Postal Code       9994 non-null float64
Market            51290 non-null object
Region            51290 non-null object
Product ID        51290 non-null object
Category          51290 non-null object
Sub-Category      51290 non-null object
Product Name      51290 non-null object
Sales             51290 non-null float64
Quantity          51290 non-null int64
Discount          51290 non-null float64
Profit            51290 non-null float64
Shipping Cost     51290 non-null float64
Order Priority    51290 non-null object
dtypes: datetime64[ns](2), float64(5), int64(2), object(15)
memory usage: 9.4+ MB
```

In [45]:
```python
## check for NA data and drop the missing data as we have plenty of records
missing_data = df_alldata.isnull().mean()*100

missing_data.sum()

df_alldata.dropna()

df_alldata.info()
```

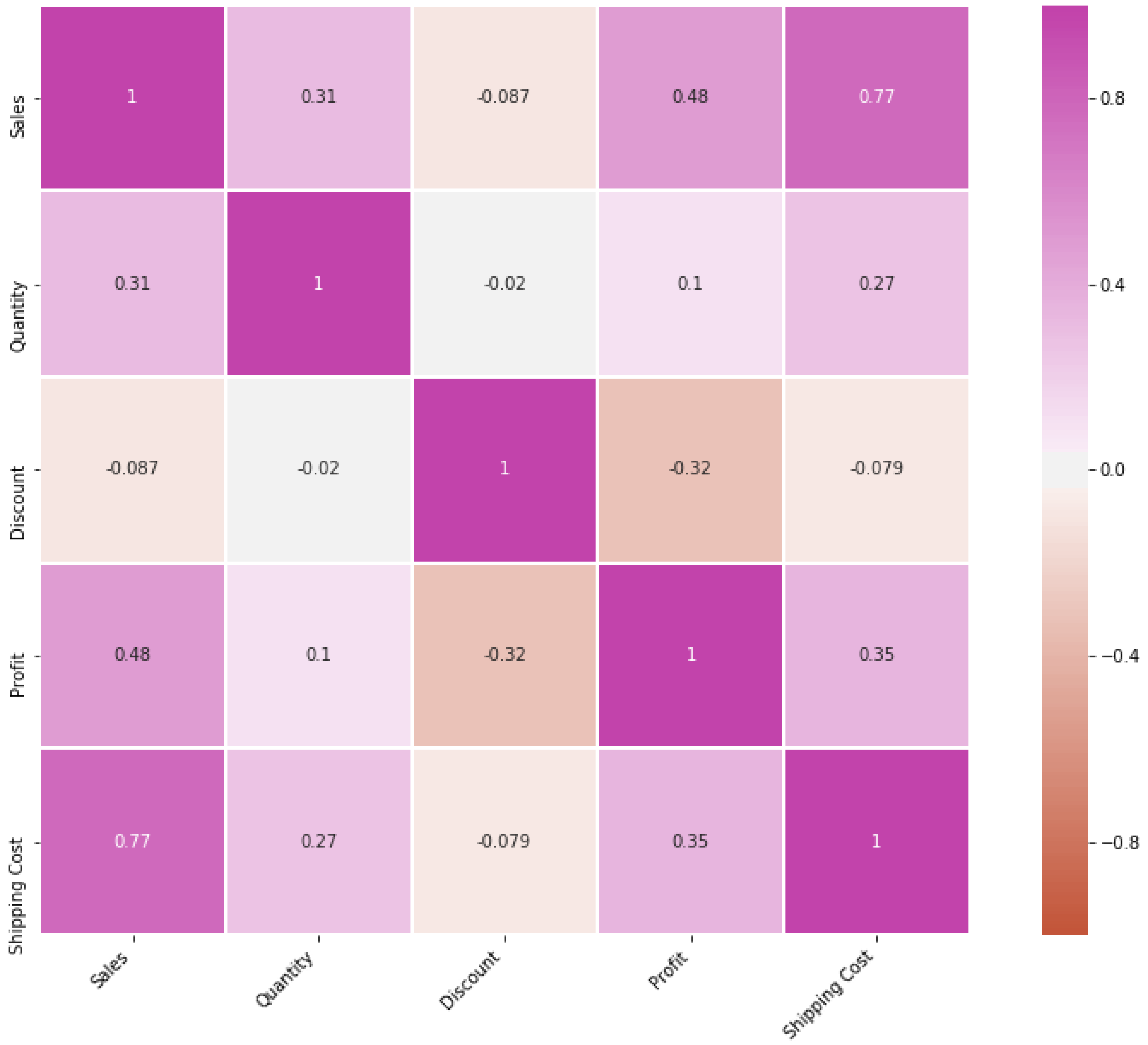
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51290 entries, 0 to 51289
Data columns (total 24 columns):
Row ID            51290 non-null int64
Order ID          51290 non-null object
Order Date        51290 non-null datetime64[ns]
Ship Date         51290 non-null datetime64[ns]
Ship Mode         51290 non-null object
Customer ID       51290 non-null object
Customer Name     51290 non-null object
Segment           51290 non-null object
City              51290 non-null object
State             51290 non-null object
Country           51290 non-null object
Postal Code       9994 non-null float64
Market            51290 non-null object
Region            51290 non-null object
Product ID        51290 non-null object
Category          51290 non-null object
Sub-Category      51290 non-null object
Product Name      51290 non-null object
Sales             51290 non-null float64
Quantity          51290 non-null int64
Discount          51290 non-null float64
Profit            51290 non-null float64
Shipping Cost     51290 non-null float64
Order Priority    51290 non-null object
dtypes: datetime64[ns](2), float64(5), int64(2), object(15)
memory usage: 9.4+ MB
```

In [46]:
```python
## Plot the continuous numuerical data to understand the distribution

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

def NumericalHistPlot(df):
    num_columns = df.columns[~(df.dtypes == 'object')]
    fig, axs = plt.subplots(2, 3, sharex=False, sharey=False, figsize=(20, 15))

    counter = 0
    for num_column in num_columns:
        trace_x = counter // 3
        trace_y = counter % 3

        axs[trace_x, trace_y].hist(df[num_column])
        axs[trace_x, trace_y].set_title(num_column)

        counter += 1

    plt.show()

df_data = df_alldata.copy()

del df_data["Row ID"]
del df_data["Postal Code"]
del df_data["Order Date"]
del df_data["Ship Date"]

NumericalHistPlot(df_data)
```


Sales
Quantity
Discount
Profit
Shipping Cost

In [47]:
```
## Check out the correlations between the continous features

import seaborn as sns

correlations = df_data.corr()

print(correlations)

plt.figure(figsize=(15,10))

ax = sns.heatmap(
    correlations,
    vmin=-1, vmax=1, center=0,
    cmap=sns.diverging_palette(20, 320, n=200),
    square=True,
    linewidths=1,
    annot=True
)

ax.set_xticklabels(
    ax.get_xticklabels(),
    rotation=45,
    horizontalalignment='right'
);
```

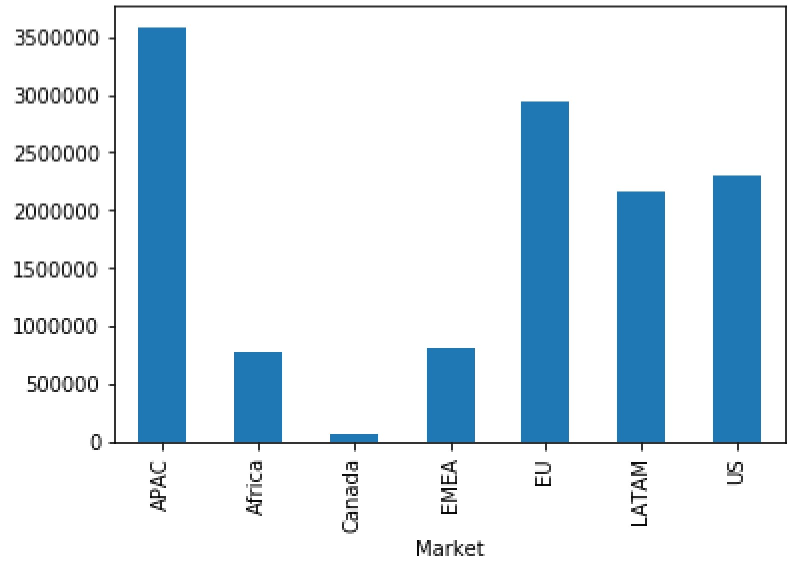
```
                  Sales  Quantity  Discount    Profit  Shipping Cost
Sales          1.000000  0.313577 -0.086722  0.484918       0.768073
Quantity       0.313577  1.000000 -0.019875  0.104365       0.272649
Discount      -0.086722 -0.019875  1.000000 -0.316490      -0.079055
Profit         0.484918  0.104365 -0.316490  1.000000       0.354441
Shipping Cost  0.768073  0.272649 -0.079055  0.354441       1.000000
```



Sales
Quantity
Discount
Profit
Shipping Cost
1
0.31
-0.087
0.48
0.77
0.31
1
-0.02
0.1
0.27
-0.087
-0.02
1
-0.32
-0.079
0.48
0.1
-0.32
1
0.35
0.77
0.27
-0.079
0.35
1
0.8
0.4
0.0
−0.4
−0.8

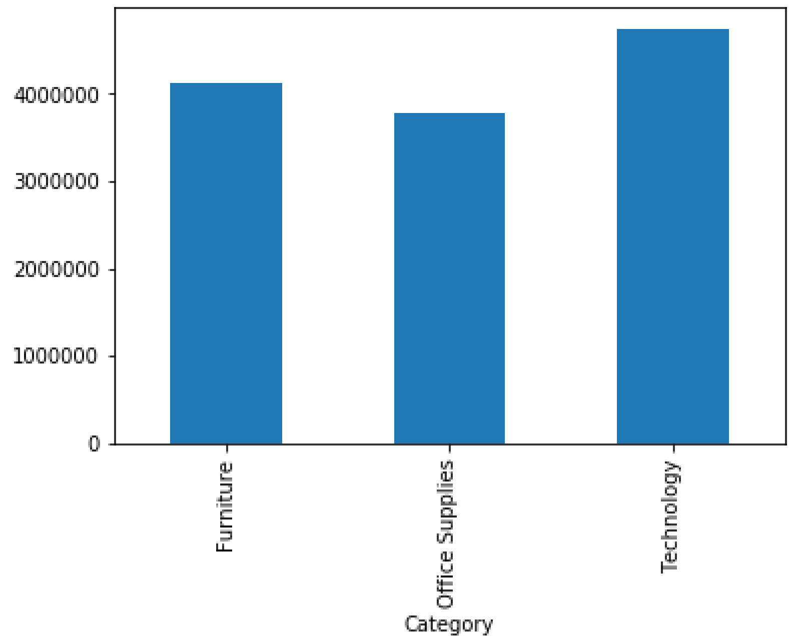
In [48]:
```
## Check out aggregate sales by Market
from matplotlib import pyplot

agg_sales_by_market = df_alldata.groupby('Market').Sales.agg('sum')
agg_sales_by_market.plot.bar()
```

Out[48]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a28ccb208>
```

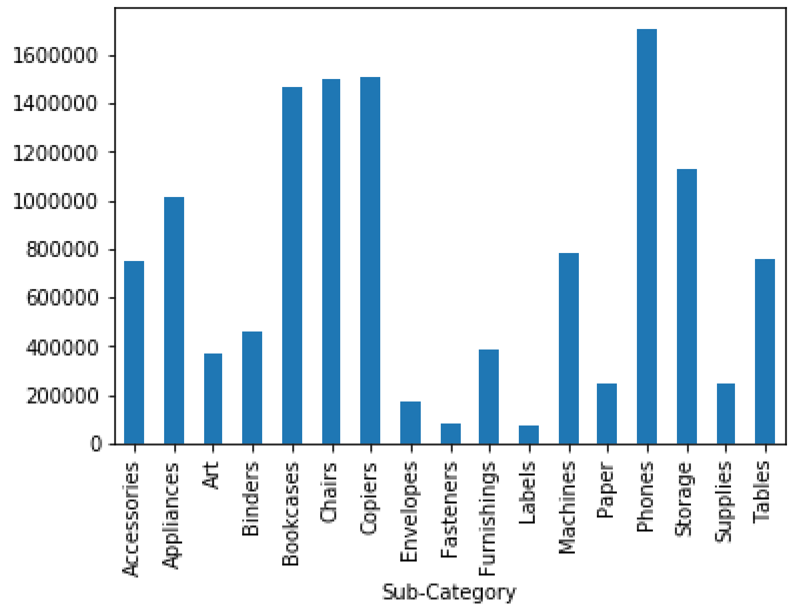


In [49]:
```
## Check out aggregate sales by Category
from matplotlib import pyplot

agg_sales_by_category = df_alldata.groupby('Category').Sales.agg('sum')
agg_sales_by_category.plot.bar()
```

Out[49]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a2898da20>
```



```
In [50]: ## Check out aggregate sales by Sub Category
         from matplotlib import pyplot

         agg_sales_by_subcategory = df_alldata.groupby('Sub-Category').Sales.agg('sum')
         agg_sales_by_subcategory.plot.bar()

Out[50]: <matplotlib.axes._subplots.AxesSubplot at 0x1a296f8c18>
```

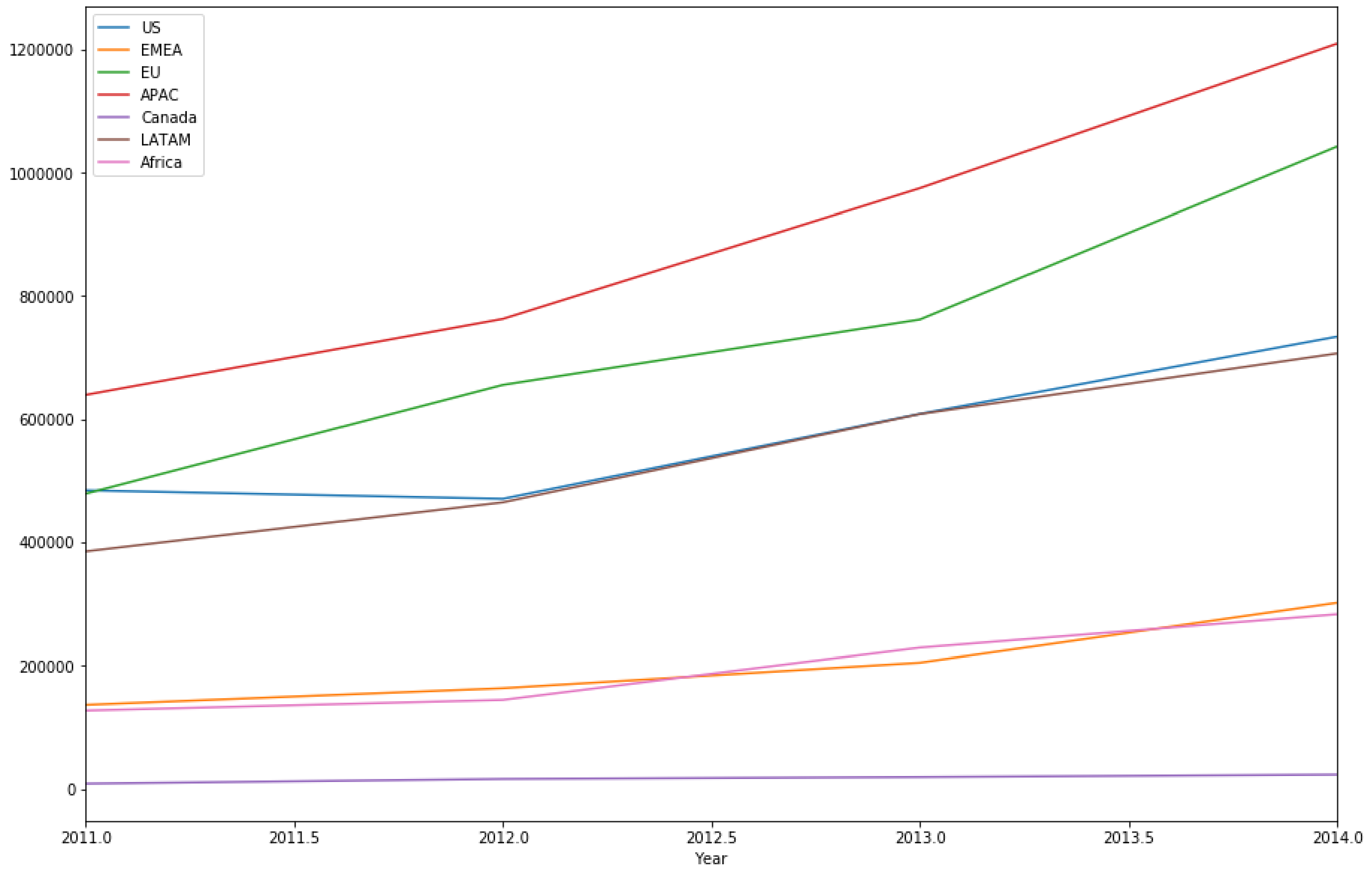


In [71]:
```python
## check trend of different market in years

df_trend_by_market = df_alldata[['Sales', 'Market', 'Order Date']]
df_trend_by_market['Year'] = df_alldata['Order Date'].apply(lambda x: x.year)

del df_trend_by_market['Order Date']
df_trend_by_market.info()

ax = plt.gca()

for mkt in set(df_trend_by_market['Market']):
    df_by_market = df_trend_by_market[df_trend_by_market['Market'] == mkt].copy()
    single_mkt_agg = df_by_market.groupby(['Year']).Sales.agg('sum')
    print(mkt)
    single_mkt_agg.plot(x='Year', y='Sales', ax=ax, figsize=(15,10), label=str(mkt))
plt.legend(loc='best')
plt.show()
```

```
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view
-versus-copy (http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy)
  after removing the cwd from sys.path.
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51290 entries, 0 to 51289
Data columns (total 3 columns):
Sales     51290 non-null float64
Market    51290 non-null object
Year      51290 non-null int64
dtypes: float64(1), int64(1), object(1)
memory usage: 1.2+ MB
US
EMEA
EU
APAC
Canada
```

```
LATAM
Africa
```



```
In [ ]:
```