Import the Libraries

In [1]:
```
# importing the required libraries
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn import metrics
```

Read the Data

In [2]:
```
# reading the data
data = pd.read_csv("rainfall in india 1901-2015.csv")
data.head()
```

Out[2]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV
0	ANDAMAN & NICOBAR ISLANDS	1901	49.2	87.1	29.2	2.3	528.8	517.5	365.1	481.1	332.6	388.5	558.2
1	ANDAMAN & NICOBAR ISLANDS	1902	0.0	159.8	12.2	0.0	446.1	537.1	228.9	753.7	666.2	197.2	359.0
2	ANDAMAN & NICOBAR ISLANDS	1903	12.7	144.0	0.0	1.0	235.1	479.9	728.4	326.7	339.0	181.2	284.4
3	ANDAMAN & NICOBAR ISLANDS	1904	9.4	14.7	0.0	202.4	304.5	495.1	502.0	160.1	820.4	222.2	308.7
4	ANDAMAN & NICOBAR ISLANDS	1905	1.3	0.0	3.3	26.9	279.5	628.7	368.7	330.5	297.0	260.7	25.4

Data Exploration and Pre-Processing

In [3]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4116 entries, 0 to 4115
Data columns (total 19 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   SUBDIVISION  4116 non-null   object 
 1   YEAR         4116 non-null   int64  
 2   JAN          4112 non-null   float64
 3   FEB          4113 non-null   float64
 4   MAR          4110 non-null   float64
 5   APR          4112 non-null   float64
 6   MAY          4113 non-null   float64
 7   JUN          4111 non-null   float64
 8   JUL          4109 non-null   float64
 9   AUG          4112 non-null   float64
 10  SEP          4110 non-null   float64
 11  OCT          4109 non-null   float64
 12  NOV          4105 non-null   float64
 13  DEC          4106 non-null   float64
 14  ANNUAL       4090 non-null   float64
 15  Jan-Feb      4110 non-null   float64
 16  Mar-May      4107 non-null   float64
 17  Jun-Sep      4106 non-null   float64
 18  Oct-Dec      4103 non-null   float64
dtypes: float64(17), int64(1), object(1)
memory usage: 611.1+ KB
```

In [4]:
```
data.isnull().sum()
```

Out[4]:
```
SUBDIVISION     0
YEAR            0
JAN             4
FEB             3
MAR             6
APR             4
MAY             3
JUN             5
JUL             7
AUG             4
SEP             6
OCT             7
NOV            11
DEC            10
ANNUAL         26
Jan-Feb         6
Mar-May         9
Jun-Sep        10
Oct-Dec        13
dtype: int64
```

In [5]:
```
data.duplicated().sum()
```

Out[5]:
```
0
```

In [6]:
```
data['SUBDIVISION'].value_counts()
```

Out[6]:
```
JHARKHAND                            115
EAST UTTAR PRADESH                   115
VIDARBHA                             115
SOUTH INTERIOR KARNATAKA             115
HARYANA DELHI & CHANDIGARH           115
HIMACHAL PRADESH                     115
COASTAL ANDHRA PRADESH               115
KERALA                               115
PUNJAB                               115
MATATHWADA                           115
EAST MADHYA PRADESH                  115
UTTARAKHAND                          115
GUJARAT REGION                       115
WEST MADHYA PRADESH                  115
TELANGANA                            115
ORISSA                               115
CHHATTISGARH                         115
KONKAN & GOA                         115
SUB HIMALAYAN WEST BENGAL & SIKKIM   115
NORTH INTERIOR KARNATAKA             115
WEST UTTAR PRADESH                   115
BIHAR                                115
TAMIL NADU                           115
ASSAM & MEGHALAYA                    115
WEST RAJASTHAN                       115
COASTAL KARNATAKA                    115
EAST RAJASTHAN                       115
MADHYA MAHARASHTRA                   115
SAURASHTRA & KUTCH                   115
NAGA MANI MIZO TRIPURA               115
GANGETIC WEST BENGAL                 115
JAMMU & KASHMIR                      115
RAYALSEEMA                           115
LAKSHADWEEP                          114
ANDAMAN & NICOBAR ISLANDS            110
ARUNACHAL PRADESH                     97
Name: SUBDIVISION, dtype: int64
```

In [7]:
```
data.mean()
```

Out[7]:
```
YEAR       1958.218659
JAN          18.957320
FEB          21.805325
MAR          27.359197
APR          43.127432
MAY          85.745417
JUN         230.234444
JUL         347.214334
AUG         290.263497
SEP         197.361922
OCT          95.507009
NOV          39.866163
DEC          18.870580
ANNUAL     1411.008900
Jan-Feb      40.747786
Mar-May     155.901753
Jun-Sep    1064.724769
Oct-Dec     154.100487
dtype: float64
```

In [8]:
```
# filling na values with mean
data = data.fillna(data.mean())
```

In [9]:
```
data.head(3)
```

Out[9]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV
0	ANDAMAN & NICOBAR ISLANDS	1901	49.2	87.1	29.2	2.3	528.8	517.5	365.1	481.1	332.6	388.5	558.2
1	ANDAMAN & NICOBAR ISLANDS	1902	0.0	159.8	12.2	0.0	446.1	537.1	228.9	753.7	666.2	197.2	359.0
2	ANDAMAN & NICOBAR ISLANDS	1903	12.7	144.0	0.0	1.0	235.1	479.9	728.4	326.7	339.0	181.2	284.4

In [10]:
```
data.isnull().any()
```

Out[10]:
```
SUBDIVISION    False
YEAR           False
JAN            False
FEB            False
MAR            False
APR            False
MAY            False
JUN            False
JUL            False
AUG            False
SEP            False
OCT            False
NOV            False
DEC            False
ANNUAL         False
Jan-Feb        False
Mar-May        False
Jun-Sep        False
Oct-Dec        False
dtype: bool
```

In [11]:
```
data.YEAR.unique()
```

Out[11]:
```
array([1901, 1902, 1903, 1904, 1905, 1906, 1907, 1908, 1910, 1911, 1912,
       1913, 1914, 1915, 1916, 1917, 1918, 1919, 1920, 1921, 1922, 1923,
       1924, 1925, 1926, 1927, 1928, 1929, 1930, 1931, 1932, 1933, 1934,
       1935, 1936, 1937, 1938, 1939, 1940, 1941, 1942, 1946, 1947, 1949,
       1950, 1951, 1952, 1953, 1954, 1955, 1956, 1957, 1958, 1959, 1960,
       1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969, 1970, 1971,
       1972, 1973, 1974, 1975, 1976, 1977, 1978, 1979, 1980, 1981, 1982,
       1983, 1984, 1985, 1986, 1987, 1988, 1989, 1990, 1991, 1992, 1993,
       1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004,
       2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015,
       1943, 1944, 1945, 1948, 1909], dtype=int64)
```

In [12]:
```
data.describe()
```

Out[12]:

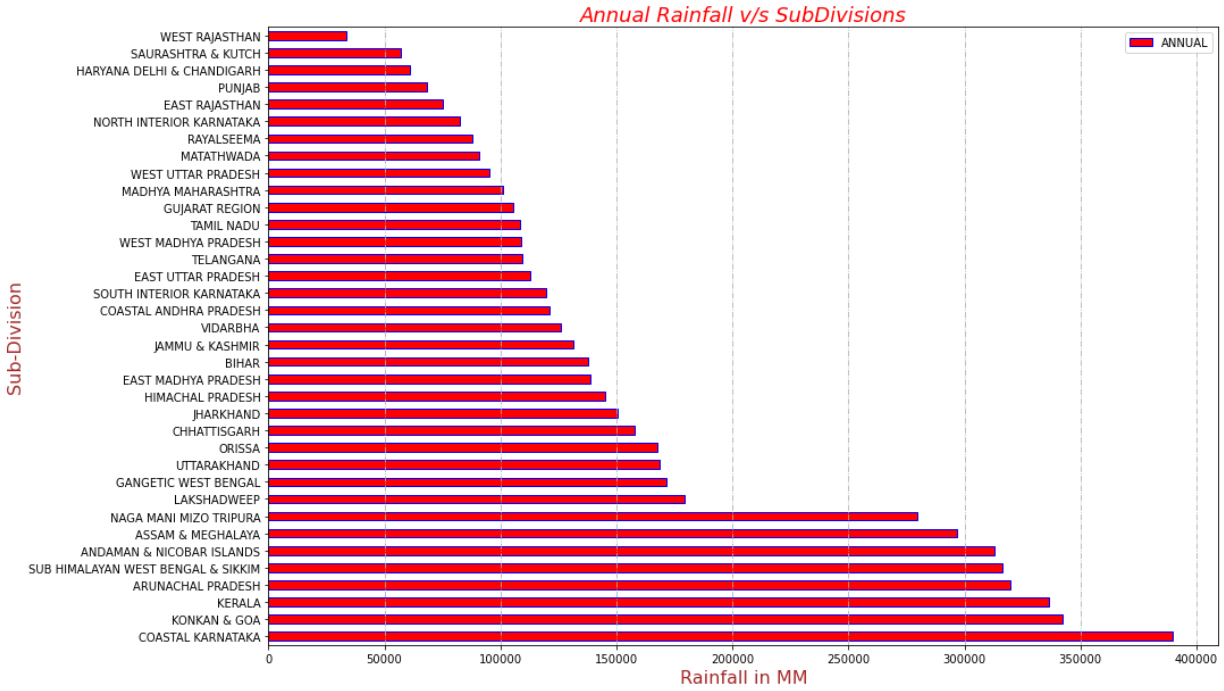
	YEAR	JAN	FEB	MAR	APR	MAY	JUN
count	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000
mean	1958.218659	18.957320	21.805325	27.359197	43.127432	85.745417	230.234444
std	33.140898	33.569044	35.896396	46.925176	67.798192	123.189974	234.568120
min	1901.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.400000
25%	1930.000000	0.600000	0.600000	1.000000	3.000000	8.600000	70.475000
50%	1958.000000	6.000000	6.700000	7.900000	15.700000	36.700000	138.900000
75%	1987.000000	22.125000	26.800000	31.225000	49.825000	96.825000	304.950000
max	2015.000000	583.700000	403.500000	605.600000	595.100000	1168.600000	1609.900000

In [13]:
```
data.shape
```

Out[13]: (4116, 19)

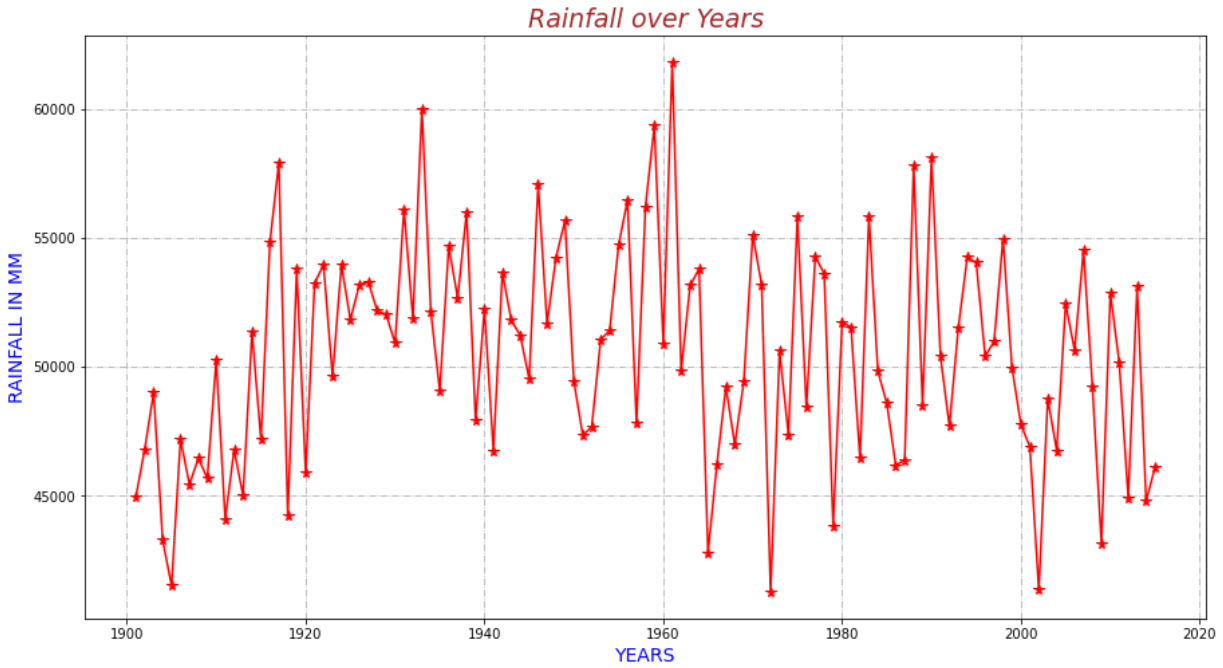
Data Visualization

In [14]:
```
data[["SUBDIVISION","ANNUAL"]].groupby("SUBDIVISION").sum().sort_values(by='ANNUA
plt.xlabel("Rainfall in MM",size=16, color='brown')
plt.ylabel("Sub-Division",size=16, color='brown')
plt.title("Annual Rainfall v/s SubDivisions", size=18, color='red', style='obliqu
plt.grid(axis="x",linestyle="-.")
plt.show()
```



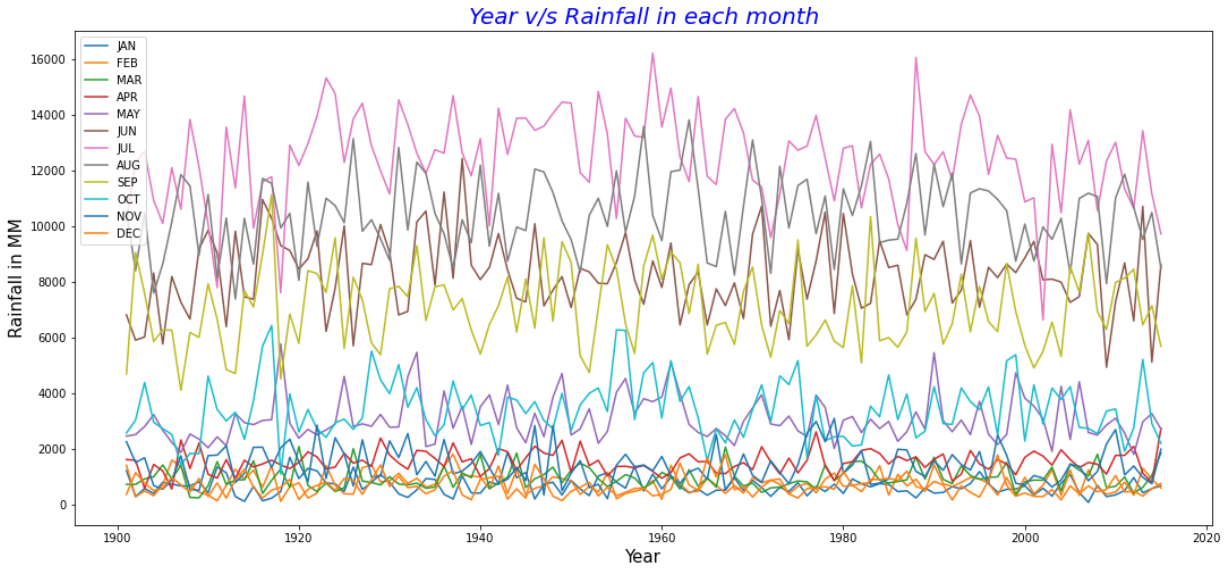
Above plot shows the total rainfall from year 1901-2015 in each subdivision,
from the above plot we note that:-
1)west Rajasthan,punjab,delhi have recieved the least rainfall
2)Rainfall was more in western states tripura,meghalaya,Arunachal Pradesh,kerala,Goa,karnatka,and also in Andaman

In [15]:
```
plt.figure(figsize=(15,8))
data.groupby("YEAR").sum()['ANNUAL'].plot(kind="line",color="red",marker="*", mar
plt.xlabel("YEARS",size=14,color='blue')
plt.ylabel("RAINFALL IN MM",size=14, color='blue')
plt.grid(axis="both",linestyle="-.")
plt.title("Rainfall over Years",size=19, color='brown',style='oblique')
plt.show()
```



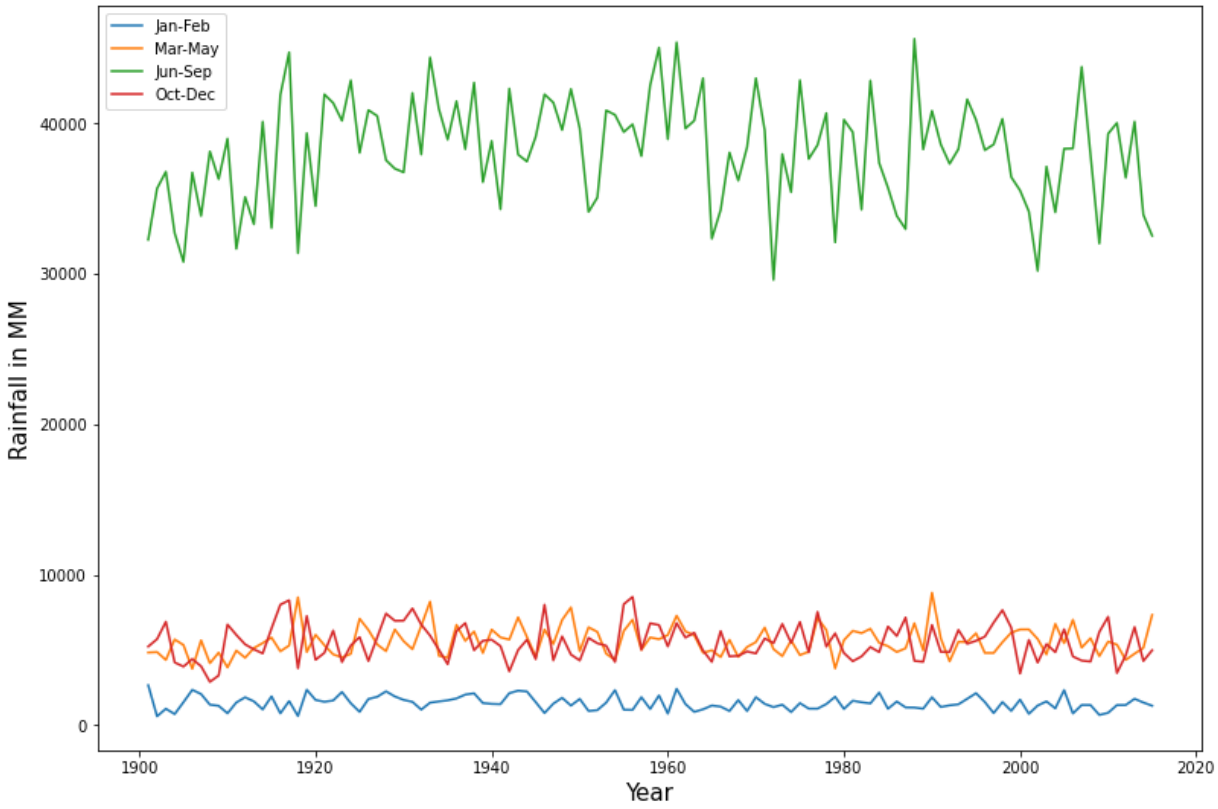
Above graph shows the Rainfall from 1901-2015 in India,
we observe that :-
1)The maximum Rainfall was in 1950s

In [16]:
```
data[['YEAR', 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL','AUG', 'SEP',
      'OCT', 'NOV', 'DEC']].groupby("YEAR").sum().plot(kind="line",figsize=(18,8)
plt.xlabel("Year",size=15)
plt.ylabel("Rainfall in MM",size=15)
plt.title("Year v/s Rainfall in each month",size=20, color='b',style='oblique')
plt.show()
```



Above plot shows the Year vs Rainfall in each month,
we observe that:-
1)July has heavy rainfall
2)Feb has least rainfall

In [17]:
```
data[['YEAR','Jan-Feb', 'Mar-May',
       'Jun-Sep', 'Oct-Dec']].groupby("YEAR").sum().plot(figsize=(13,9))
plt.xlabel("Year",size=15)
plt.ylabel("Rainfall in MM",size=15)
plt.show()
```

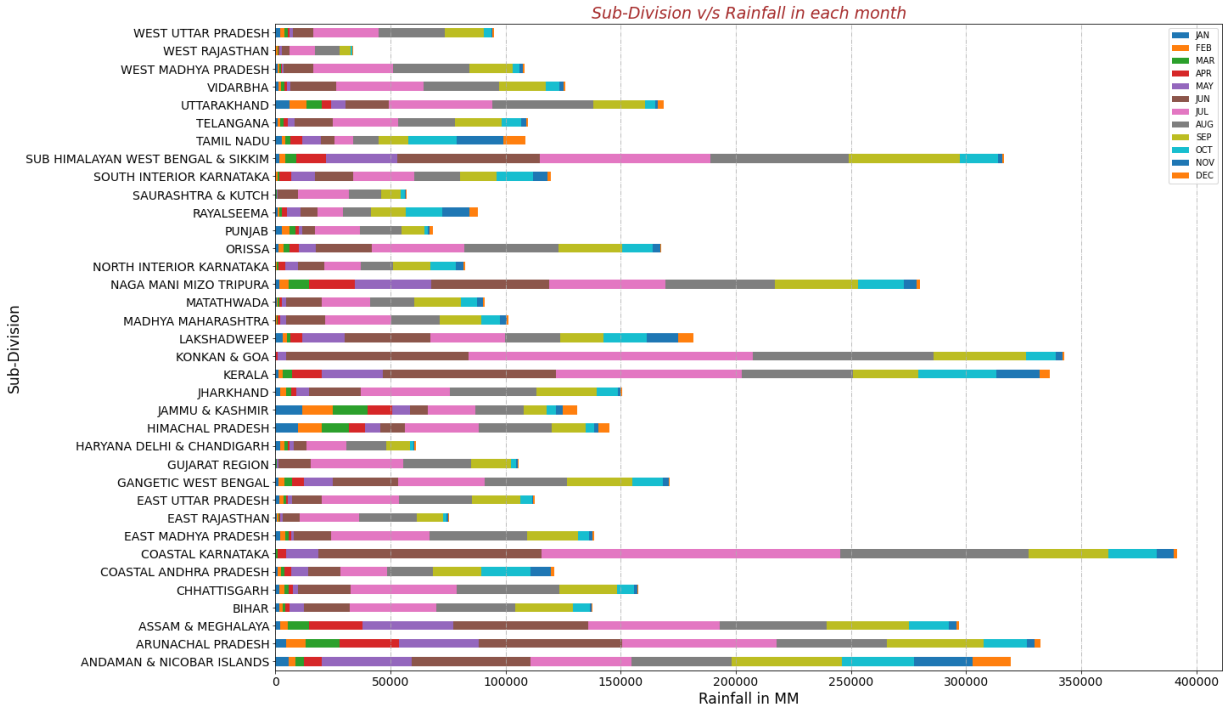


From Above Graph we observe that :-

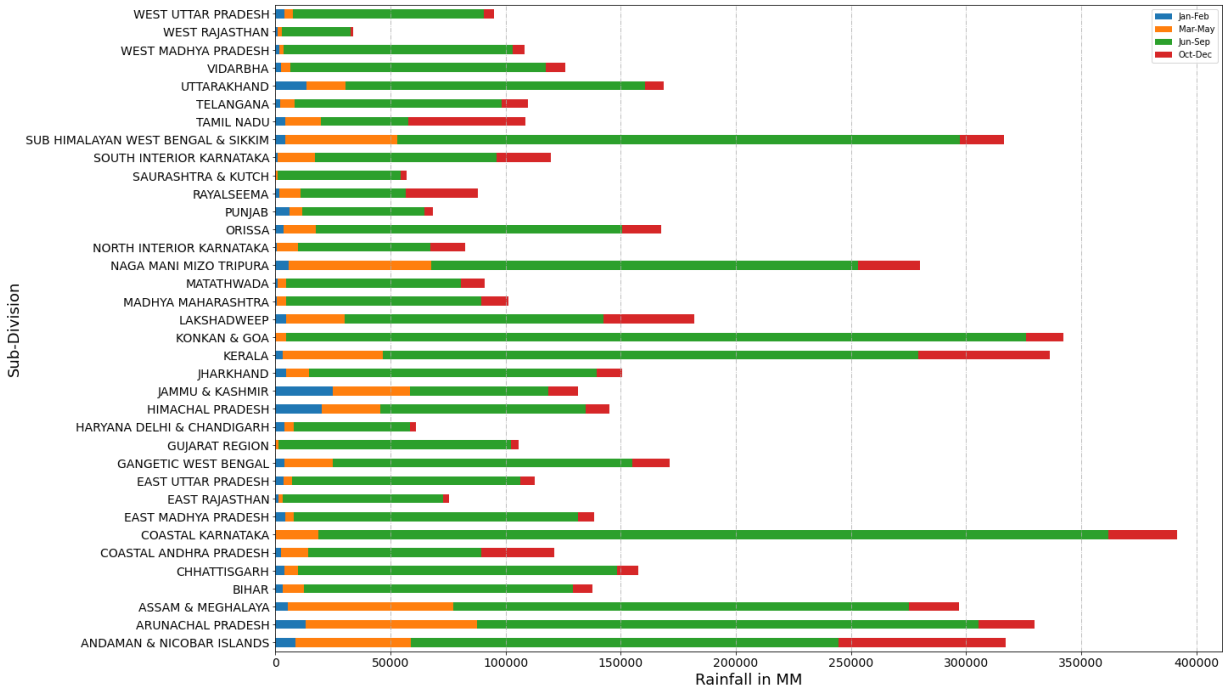
1)combined jun,july,august,sept recieves huge rainfall

2)combined jan,feb recieves least Rainfall

In [18]:
```
data[['SUBDIVISION', 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL',
       'AUG', 'SEP', 'OCT', 'NOV', 'DEC']].groupby("SUBDIVISION").sum().plot(kind
plt.title("Sub-Division v/s Rainfall in each month",size=19, color='brown',style=
plt.xlabel("Rainfall in MM",size=17)
plt.ylabel("Sub-Division",size=17)
plt.grid(axis="x",linestyle="-.")
plt.show()
```



In [19]:
```
data[['SUBDIVISION', 'Jan-Feb', 'Mar-May',
       'Jun-Sep', 'Oct-Dec']].groupby("SUBDIVISION").sum().plot(kind="barh",stack
plt.xlabel("Rainfall in MM",size=18)
plt.ylabel("Sub-Division",size=18)
plt.grid(axis="x",linestyle="-.")
plt.show()
```



From the above two graph we observe that:-

1)eastern states have good amont of rainfall in march,april,may

In [20]:
```
# Analysis of rainfall data of tamil nadu
TN = data.loc[((data['SUBDIVISION'] == 'TAMIL NADU'))]
TN.head(4)
```

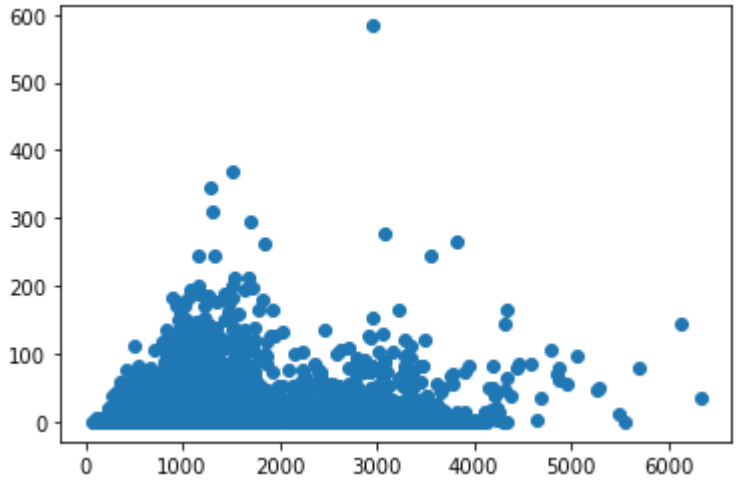
Out[20]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NO
3427	TAMIL NADU	1901	24.5	39.1	21.7	36.0	74.0	41.8	49.3	67.9	191.1	122.3	212
3428	TAMIL NADU	1902	67.2	9.8	25.1	21.9	84.7	39.3	55.1	113.8	98.6	282.2	174
3429	TAMIL NADU	1903	19.3	7.8	1.7	18.2	128.5	58.5	72.6	115.0	210.4	128.1	200
3430	TAMIL NADU	1904	35.2	0.1	0.7	19.5	121.9	34.9	89.0	40.4	85.7	163.2	23

In [21]:
```
print("Scatter plot of annual and january attributes")
plt.scatter(data.ANNUAL,data.JAN)
```

```
Scatter plot of annual and january attributes
```

Out[21]:
```
<matplotlib.collections.PathCollection at 0x115188917c0>
```



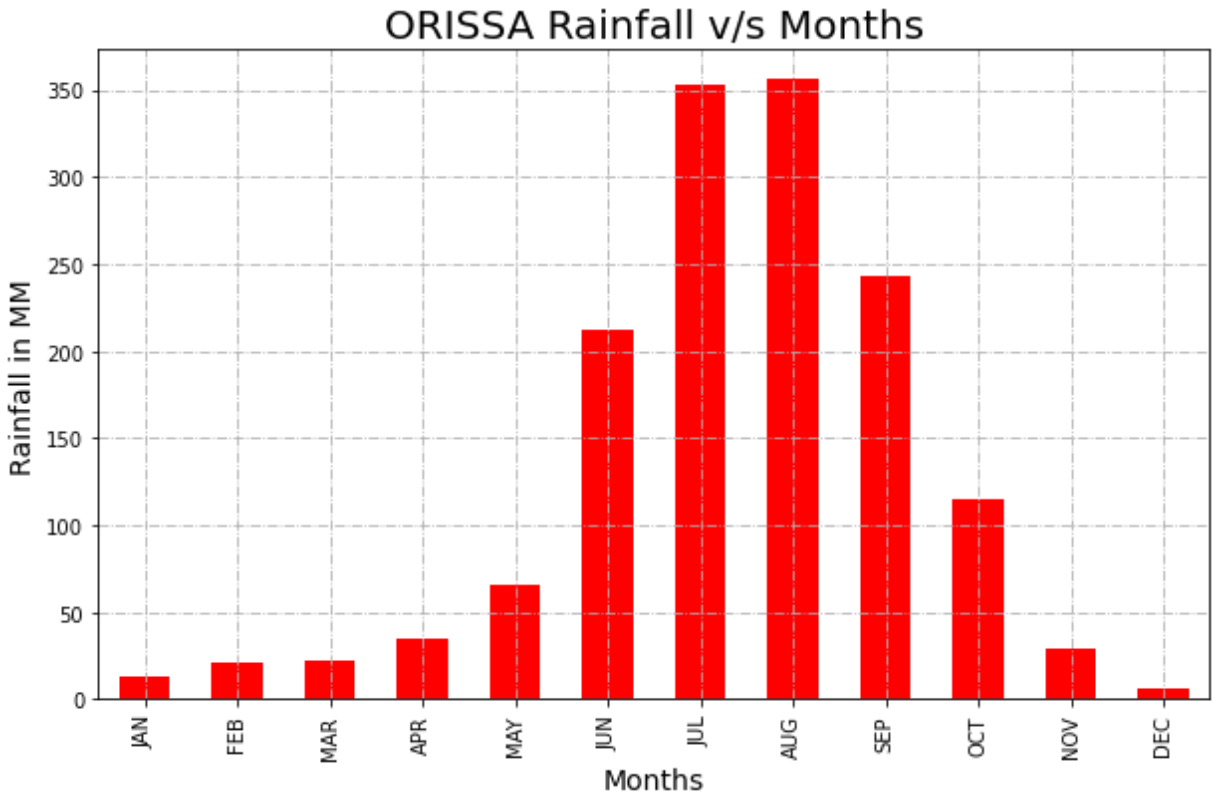
In [22]:
```
# Analysis of rainfall data of Orissa
OR = data.loc[((data['SUBDIVISION'] == 'ORISSA'))]
OR.head(4)
```

Out[22]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV
667	ORISSA	1901	39.5	65.1	16.1	51.6	79.0	78.2	288.4	307.7	185.3	76.6	96.7
668	ORISSA	1902	3.4	0.2	14.2	101.1	56.7	108.3	437.4	349.1	202.7	33.2	13.0
669	ORISSA	1903	19.7	18.9	10.5	34.6	73.3	154.3	410.4	295.2	265.6	228.5	46.2
670	ORISSA	1904	0.2	12.2	20.6	10.1	100.2	342.9	336.7	350.4	227.8	111.8	0.0

In [23]:
```
plt.figure(figsize=(10,6))
OR[['JAN', 'FEB', 'MAR', 'APR','MAY', 'JUN','JUL','AUG', 'SEP', 'OCT','NOV','DEC'
plt.title("ORISSA Rainfall v/s Months",size=20)
plt.xlabel("Months",size=14)
plt.ylabel("Rainfall in MM",size=14)
plt.grid(axis="both",linestyle="-.")
plt.show()
```

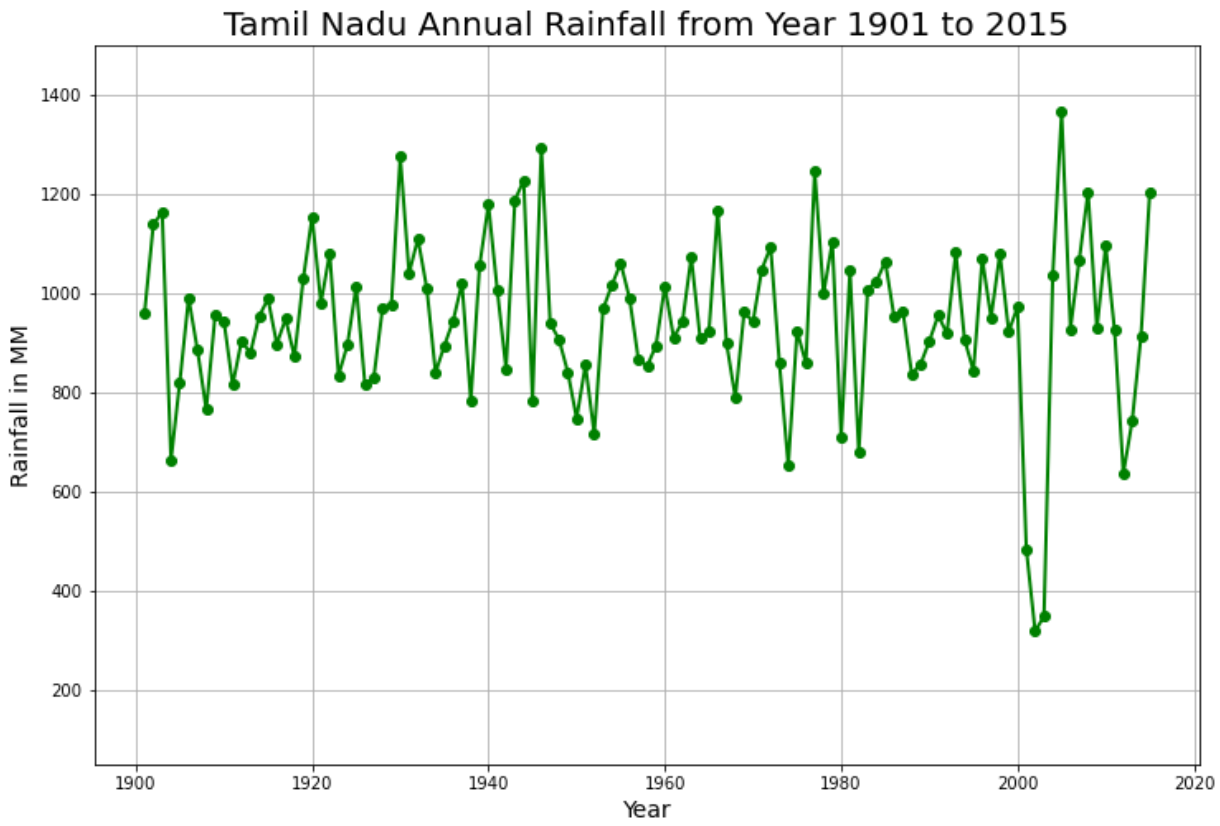
```
C:\Users\malik\anaconda3\lib\site-packages\pandas\plotting\_matplotlib\core.py:
1373: MatplotlibDeprecationWarning: Case-insensitive properties were deprecated
in 3.3 and support will be removed two minor releases later
  return ax.bar(x, y, w, bottom=start, log=log, **kwds)
```



From the above graph we observe that:-

1)Tamil Nadu has good amount of rainfall in oct and nov

In [26]:
```
TN.groupby("YEAR").sum()['ANNUAL'].plot(ylim=(50,1500),color='green',marker='o',l
plt.xlabel('Year',size=14)
plt.ylabel('Rainfall in MM',size=14)
plt.title('Tamil Nadu Annual Rainfall from Year 1901 to 2015',size=20)
plt.grid()
plt.show()
```

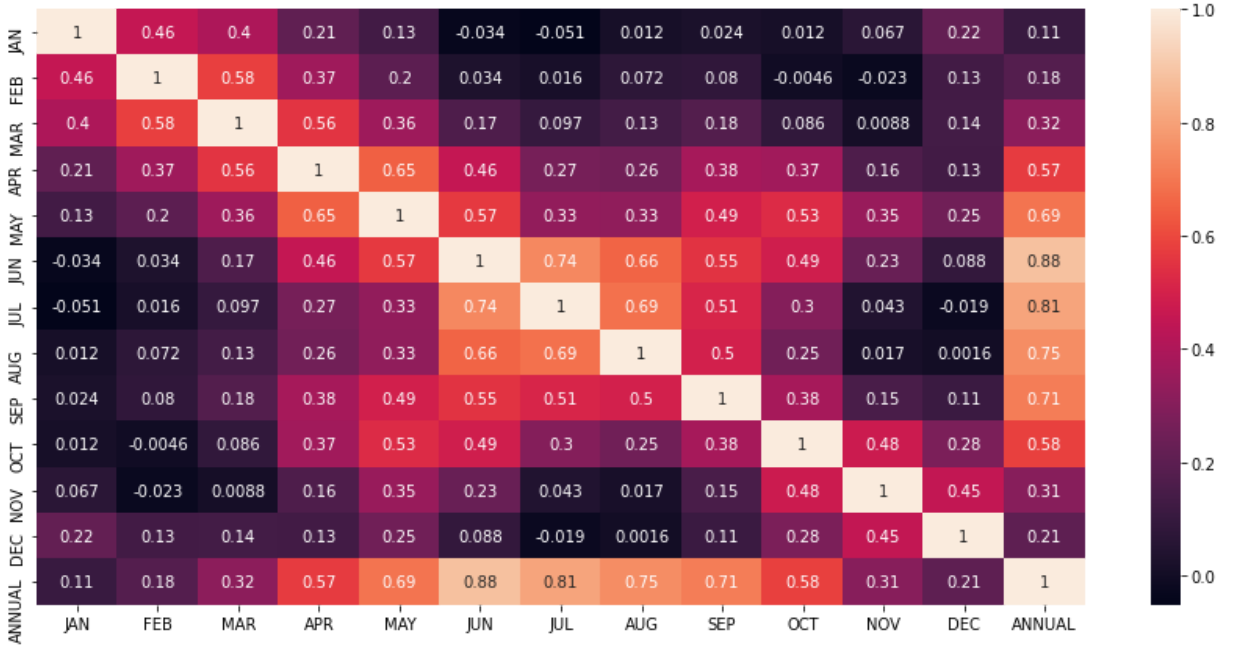


From the Above graph we observe that:

1)The lowest rainfall in Tamil Nadu was noted in 2002-2003

2)and, The highest was Rainfall was noted in 2005

In [30]:
```
# correlation b/w each numeric attribute
plt.figure(figsize=(15,7))
sns.heatmap(data[['JAN','FEB','MAR','APR','MAY','JUN','JUL','AUG','SEP','OCT','NO
plt.show()
```



The above heatmap shows the coorelation between different features in the dataset

Modelling

In [31]:
```
data["SUBDIVISION"].nunique()
```

Out[31]: 36

In [32]:
```
group = data.groupby('SUBDIVISION')['YEAR','JAN','FEB','MAR','APR','MAY','JUN','J
data=group.get_group(('ORISSA'))
data.head()
```

```
<ipython-input-32-27f63946a9eb>:1: FutureWarning: Indexing with multiple keys
(implicitly converted to a tuple of keys) will be deprecated, use a list instea
d.
  group = data.groupby('SUBDIVISION')['YEAR','JAN','FEB','MAR','APR','MAY','JU
N','JUL','AUG','SEP','OCT','NOV','DEC']
```

Out[32]:

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
667	1901	39.5	65.1	16.1	51.6	79.0	78.2	288.4	307.7	185.3	76.6	96.7	0.0
668	1902	3.4	0.2	14.2	101.1	56.7	108.3	437.4	349.1	202.7	33.2	13.0	29.6
669	1903	19.7	18.9	10.5	34.6	73.3	154.3	410.4	295.2	265.6	228.5	46.2	11.0
670	1904	0.2	12.2	20.6	10.1	100.2	342.9	336.7	350.4	227.8	111.8	0.0	1.9
671	1905	24.3	17.2	66.3	56.9	107.5	92.0	330.1	281.4	344.1	36.4	0.7	0.4

In [33]:
```
df=data.melt(['YEAR']).reset_index()
df.head()
```

Out[33]:

	index	YEAR	variable	value
0	0	1901	JAN	39.5
1	1	1902	JAN	3.4
2	2	1903	JAN	19.7
3	3	1904	JAN	0.2
4	4	1905	JAN	24.3

In [34]:
```
df= df[['YEAR','variable','value']].reset_index().sort_values(by=['YEAR','index']
df.head()
```

Out[34]:

	index	YEAR	variable	value
0	0	1901	JAN	39.5
115	115	1901	FEB	65.1
230	230	1901	MAR	16.1
345	345	1901	APR	51.6
460	460	1901	MAY	79.0

In [35]:
```
df.YEAR.unique()
```

Out[35]:
```
array([1901, 1902, 1903, 1904, 1905, 1906, 1907, 1908, 1909, 1910, 1911,
       1912, 1913, 1914, 1915, 1916, 1917, 1918, 1919, 1920, 1921, 1922,
       1923, 1924, 1925, 1926, 1927, 1928, 1929, 1930, 1931, 1932, 1933,
       1934, 1935, 1936, 1937, 1938, 1939, 1940, 1941, 1942, 1943, 1944,
       1945, 1946, 1947, 1948, 1949, 1950, 1951, 1952, 1953, 1954, 1955,
       1956, 1957, 1958, 1959, 1960, 1961, 1962, 1963, 1964, 1965, 1966,
       1967, 1968, 1969, 1970, 1971, 1972, 1973, 1974, 1975, 1976, 1977,
       1978, 1979, 1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987, 1988,
       1989, 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999,
       2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010,
       2011, 2012, 2013, 2014, 2015], dtype=int64)
```

In [36]:
```
df.columns=['Index','Year','Month','Avg_Rainfall']
```

In [37]:
```
df.head()
```

Out[37]:

	Index	Year	Month	Avg_Rainfall
0	0	1901	JAN	39.5
115	115	1901	FEB	65.1
230	230	1901	MAR	16.1
345	345	1901	APR	51.6
460	460	1901	MAY	79.0

In [38]:
```
Month_map={'JAN':1,'FEB':2,'MAR' :3,'APR':4,'MAY':5,'JUN':6,'JUL':7,'AUG':8,'SEP'
   'OCT':10,'NOV':11,'DEC':12}
df['Month']=df['Month'].map(Month_map)
df.head(12)
```

Out[38]:

	Index	Year	Month	Avg_Rainfall
0	0	1901	1	39.5
115	115	1901	2	65.1
230	230	1901	3	16.1
345	345	1901	4	51.6
460	460	1901	5	79.0
575	575	1901	6	78.2
690	690	1901	7	288.4
805	805	1901	8	307.7
920	920	1901	9	185.3
1035	1035	1901	10	76.6
1150	1150	1901	11	96.7
1265	1265	1901	12	0.0

In [39]:
```
df.drop(columns="Index",inplace=True)
```

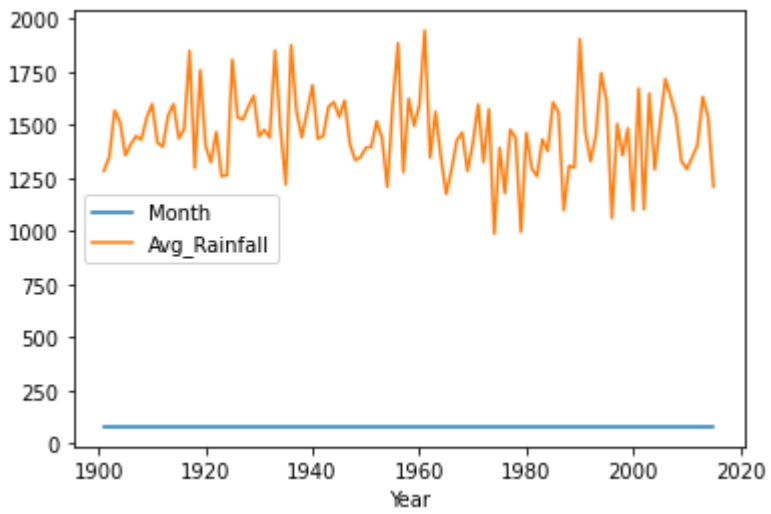
In [40]:
```
df.head(2)
```

Out[40]:

	Year	Month	Avg_Rainfall
0	1901	1	39.5
115	1901	2	65.1

In [41]:
```
plt.figure(figsize=(50,12))
df.groupby("Year").sum().plot()
plt.show()
```

```
<Figure size 3600x864 with 0 Axes>
```



In [42]:
```
X=np.asanyarray(df[['Year','Month']]).astype('int')
y=np.asanyarray(df['Avg_Rainfall']).astype('int')
print(X.shape)
print(y.shape)
```

```
(1380, 2)
(1380,)
```

In [43]:
```
# splitting the dataset into training and testing
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_s
```

Linear Regression Model

In [44]:
```
from sklearn.linear_model import LinearRegression
LR = LinearRegression()
LR.fit(X_train,y_train)
```

Out[44]:
```
LinearRegression()
```

In [45]:
```
# predicting
y_train_predict=LR.predict(X_train)
y_test_predict=LR.predict(X_test)
```

In [46]:
```
print("-------Test Data--------")
print('MAE:', metrics.mean_absolute_error(y_test, y_test_predict))
print('MSE:', metrics.mean_squared_error(y_test, y_test_predict))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_test_predict)))

print("\n-------Train Data--------")
print('MAE:', metrics.mean_absolute_error(y_train,y_train_predict))
print('MSE:', metrics.mean_squared_error(y_train, y_train_predict))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_train, y_train_predict)))

print("\n-----Training Accuracy-------")
print(round(LR.score(X_train,y_train),3)*100)
print("-----Testing Accuracy--------")
print(round(LR.score(X_test,y_test),3)*100)
```

```
-------Test Data--------
MAE: 111.40064171312682
MSE: 18400.0755527868
RMSE: 135.64687815348645

-------Train Data--------
MAE: 108.99562619101339
MSE: 18038.815538997766
RMSE: 134.30865772167394

-----Training Accuracy-------
5.8999999999999995
-----Testing Accuracy--------
4.6
```

Lasso Model

In [47]:
```
from sklearn.linear_model import Lasso
from sklearn.model_selection import GridSearchCV

# create a lasso object
lasso = Lasso(max_iter=100000)

# check for best alpha value using GridSearch
parameter={'alpha':[1e-15,1e-10,1e-8,1e-3,1e-2,1,5,1e1,1e2,1e3,1e4,1e5,1e6,1e7]}
lasso_regressor=GridSearchCV(
    lasso,parameter,
    scoring='neg_mean_squared_error',
    cv=5
    )
```

In [48]:
```
lasso_regressor.fit(X_train,y_train)
```

```
C:\Users\malik\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_des
cent.py:529: ConvergenceWarning: Objective did not converge. You might want to
increase the number of iterations. Duality gap: 6123370.262930247, tolerance: 1
502.740079172057
  model = cd_fast.enet_coordinate_descent(
C:\Users\malik\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_des
cent.py:529: ConvergenceWarning: Objective did not converge. You might want to
increase the number of iterations. Duality gap: 6953069.799215374, tolerance: 1
526.727245278137
  model = cd_fast.enet_coordinate_descent(
C:\Users\malik\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_des
cent.py:529: ConvergenceWarning: Objective did not converge. You might want to
increase the number of iterations. Duality gap: 6439031.091742183, tolerance: 1
445.2750716688226
  model = cd_fast.enet_coordinate_descent(
```

Out[48]:
```
GridSearchCV(cv=5, estimator=Lasso(max_iter=100000),
             param_grid={'alpha': [1e-15, 1e-10, 1e-08, 0.001, 0.01, 1, 5, 10.
0,
                                   100.0, 1000.0, 10000.0, 100000.0, 1000000.0,
                                   10000000.0]},
             scoring='neg_mean_squared_error')
```

In [49]:
```
print("Best Parameter for Lasso:",lasso_regressor.best_estimator_)
```

```
Best Parameter for Lasso: Lasso(alpha=5, max_iter=100000)
```

In [50]:
```
lasso=Lasso(alpha=100.0,max_iter=100000)

# fit into the object
lasso.fit(X_train,y_train)
```

Out[50]:
```
Lasso(alpha=100.0, max_iter=100000)
```

In [51]:
```
# predicting
y_train_predict=lasso.predict(X_train)
y_test_predict=lasso.predict(X_test)
```

In [52]:
```
from sklearn import metrics
print("-------Test Data--------")
print('MAE:', metrics.mean_absolute_error(y_test, y_test_predict))
print('MSE:', metrics.mean_squared_error(y_test, y_test_predict))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_test_predict)))

print("\n-------Train Data--------")
print('MAE:', metrics.mean_absolute_error(y_train,y_train_predict))
print('MSE:', metrics.mean_squared_error(y_train, y_train_predict))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_train, y_train_predict)))

print("\n-----Training Accuracy-------")
print(round(lasso.score(X_train,y_train),3)*100)
print("-----Testing Accuracy--------")
print(round(lasso.score(X_test,y_test),3)*100)
```

```
-------Test Data--------
MAE: 117.72544935771658
MSE: 19035.16288049466
RMSE: 137.96797773575815

-------Train Data--------
MAE: 115.87878230221885
MSE: 18874.24709777946
RMSE: 137.38357652128386

-----Training Accuracy-------
1.6
-----Testing Accuracy--------
1.3
```

Ridge Model

In [53]:
```
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV

ridge=Ridge()
parameters={'alpha':[1e-15,1e-10,1e-8,1e-3,1e-2,1,5,10,20,30,35,40,45,50,55,100]}
ridge_regressor=GridSearchCV(ridge,parameters,scoring='neg_mean_squared_error',cv
ridge_regressor.fit(X_train,y_train)

print(ridge_regressor.best_params_)
print(ridge_regressor.best_score_)
```

```
{'alpha': 100}
-18120.236827294655
```

In [54]:
```
print("Best Parameter for Ridge:",ridge_regressor.best_estimator_)
```

```
Best Parameter for Ridge: Ridge(alpha=100)
```

In [55]:
```
ridge=Ridge(alpha=100.0)

# fit into the object
ridge.fit(X_train,y_train)
```

Out[55]:
```
Ridge(alpha=100.0)
```

In [56]:
```
# predicting the train and test values
y_train_predict=ridge.predict(X_train)
y_test_predict=ridge.predict(X_test)
```

In [57]:
```
from sklearn import metrics
print("-------Test Data--------")
print('MAE:', metrics.mean_absolute_error(y_test, y_test_predict))
print('MSE:', metrics.mean_squared_error(y_test, y_test_predict))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_test_predict)))

print("\n-------Train Data--------")
print('MAE:', metrics.mean_absolute_error(y_train,y_train_predict))
print('MSE:', metrics.mean_squared_error(y_train, y_train_predict))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_train, y_train_predict)))

print("\n-----Training Accuracy-------")
print(round(ridge.score(X_train,y_train),3)*100)
print("-----Testing Accuracy--------")
print(round(ridge.score(X_test,y_test),3)*100)
```

```
-------Test Data--------
MAE: 111.45418498474254
MSE: 18398.21415841806
RMSE: 135.64001680336838

-------Train Data--------
MAE: 109.05576772283105
MSE: 18038.898541943272
RMSE: 134.30896672204454

-----Training Accuracy-------
5.8999999999999995
-----Testing Accuracy--------
4.6
```

Svm Model

In [58]:
```
from sklearn import preprocessing
from sklearn import svm

svm_regr = svm.SVC(kernel='rbf')
svm_regr.fit(X_train, y_train)
```

Out[58]:
```
SVC()
```

In [59]:
```
y_test_predict = svm_regr.predict(X_test)
y_train_predict = svm_regr.predict(X_train)
```

In [60]:
```
from sklearn import metrics
print("-------Test Data--------")
print('MAE:', metrics.mean_absolute_error(y_test, y_test_predict))
print('MSE:', metrics.mean_squared_error(y_test, y_test_predict))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_test_predict)))

print("\n-------Train Data--------")
print('MAE:', metrics.mean_absolute_error(y_train,y_train_predict))
print('MSE:', metrics.mean_squared_error(y_train, y_train_predict))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_train, y_train_predict)))


print("\n-----Training Accuracy-------")
print(round(svm_regr.score(X_train,y_train),3)*100)
print("-----Testing Accuracy--------")
print(round(svm_regr.score(X_test,y_test),3)*100)
```

```
-------Test Data--------
MAE: 123.64251207729468
MSE: 34570.913043478264
RMSE: 185.93254971488523

-------Train Data--------
MAE: 119.98654244306418
MSE: 33570.43581780538
RMSE: 183.22236713296056

-----Training Accuracy-------
8.9
-----Testing Accuracy--------
10.100000000000001
```

Random Forest Model

In [61]:
```
from sklearn.ensemble import RandomForestRegressor
random_forest_model = RandomForestRegressor(max_depth=100, max_features='sqrt', m
                      min_samples_split=10, n_estimators=800)
random_forest_model.fit(X_train, y_train)
```

Out[61]:
```
RandomForestRegressor(max_depth=100, max_features='sqrt', min_samples_leaf=4,
                      min_samples_split=10, n_estimators=800)
```

In [62]:
```
y_train_predict=random_forest_model.predict(X_train)
y_test_predict=random_forest_model.predict(X_test)
```

In [63]:
```
print("-------Test Data--------")
print('MAE:', metrics.mean_absolute_error(y_test, y_test_predict))
print('MSE:', metrics.mean_squared_error(y_test, y_test_predict))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_test_predict)))

print("\n-------Train Data--------")
print('MAE:', metrics.mean_absolute_error(y_train,y_train_predict))
print('MSE:', metrics.mean_squared_error(y_train, y_train_predict))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_train, y_train_predict)))
```

```
-------Test Data--------
MAE: 41.33033981895136
MSE: 3359.8826065498442
RMSE: 57.96449436120222

-------Train Data--------
MAE: 32.872926542755934
MSE: 2336.435139549825
RMSE: 48.33668523543816
```

In [64]:
```
print("-----------Training Accuracy------------")
print(round(random_forest_model.score(X_train,y_train),3)*100)
print("-----------Testing Accuracy------------")
print(round(random_forest_model.score(X_test,y_test),3)*100)
```

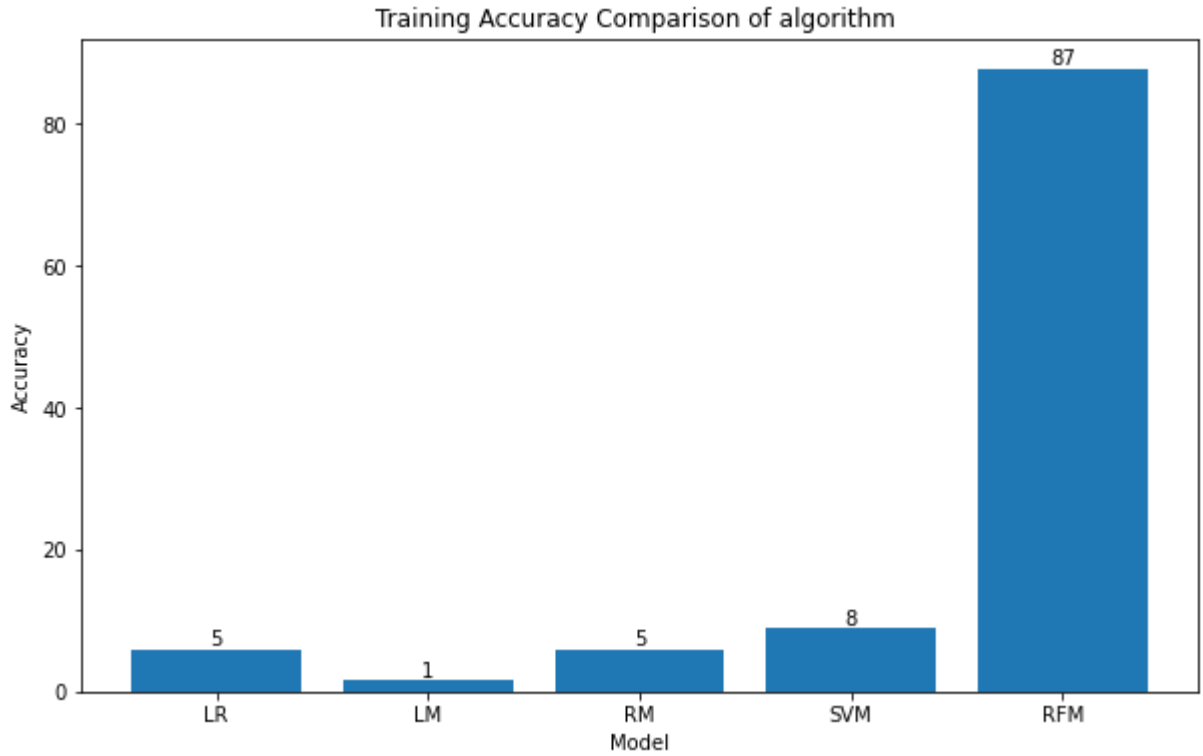
```
-----------Training Accuracy------------
87.8
-----------Testing Accuracy------------
82.6
```

In [65]:
```
predicted = random_forest_model.predict([[2016,11]])
```

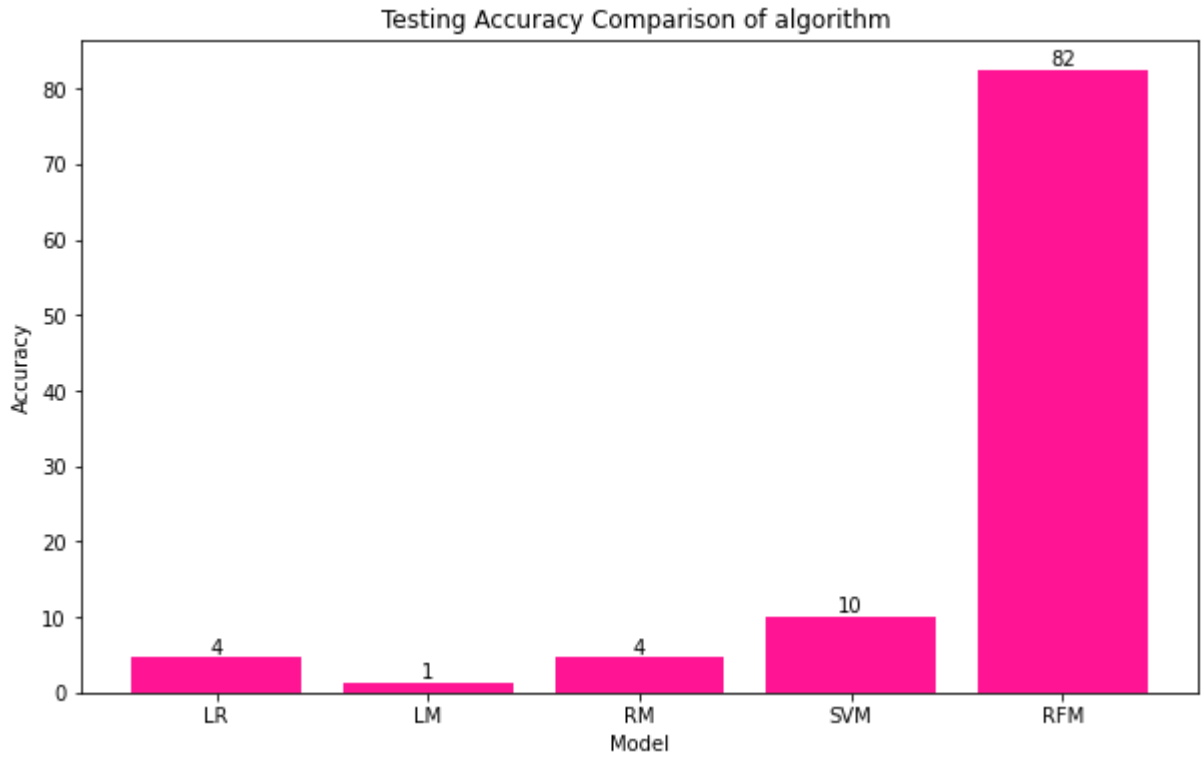
In [66]:
```
predicted
```

Out[66]:
```
array([65.18018043])
```

In [67]:
```
import matplotlib.pyplot as plt

Model=['LR', 'LM', 'RM', 'SVM', 'RFM']
Accuracy=[5.8999999999999995, 1.6, 5.8999999999999995, 8.9, 87.6]

plt.figure(figsize=(10,6))
#plot the bar graph
plot=plt.bar(Model, Accuracy)

for value in plot:
    height = value.get_height()
    plt.text(value.get_x() + value.get_width ()/2.,
            1.002*height, '%d' %int(height), ha='center',va='bottom')

#Add labels and title
plt.title("Training Accuracy Comparison of algorithm ")
plt.xlabel("Model")
plt.ylabel("Accuracy")

plt.show()
```



In [68]:
```
import matplotlib.pyplot as plt

Model=['LR', 'LM', 'RM', 'SVM', 'RFM']
Accuracy=[4.6, 1.3, 4.6, 10.100000000000001, 82.39999999999999]

plt.figure(figsize=(10,6))
#plot the bar graph
plot=plt.bar(Model, Accuracy,color = 'deeppink')

for value in plot:
    height = value.get_height()
    plt.text(value.get_x() + value.get_width ()/2.,
            1.002*height, '%d' %int(height), ha='center',va='bottom')

#Add labels and title
plt.title("Testing Accuracy Comparison of algorithm ")
plt.xlabel("Model")
plt.ylabel("Accuracy")

plt.show()
```



In [ ]: