In [36]:

```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import Regression
from scipy.stats import f
import sys
from Regression import SWEEPOperator
from scipy.stats import chi2, norm
```

In [37]:

```python
df = pd.read_excel("WeightDiary.xlsx")
df["Month"] = df["Date"].dt.month_name()
df["DayOfWeek"] = df['Date'].dt.day_name()
df = df[["Month","DayOfWeek","Weight"]]
df
```

Out[37]:

	Month	DayOfWeek	Weight
0	March	Saturday	2.0970000e+02
1	March	Saturday	2.1240000e+02
2	March	Sunday	2.1000000e+02
3	March	Sunday	2.1430000e+02
4	March	Monday	2.0910000e+02
...	...	...	...
1039	December	Friday	2.1780000e+02
1040	December	Saturday	2.1560000e+02
1041	December	Sunday	2.1450000e+02
1042	December	Monday	2.1470000e+02
1043	December	Tuesday	2.1250000e+02

1044 rows × 3 columns

Q1(a). Provide a frequency table for the Month, and another frequency table for the DayOfWeek.

In [38]:

```
month = df['Month'].astype('category')
month_reorder = month.cat.reorder_categories(['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August
df_month_reorder = pd.DataFrame(month_reorder)
df_month_reorder.join(df[['Weight']]).groupby("Month").agg(len)
```

Out[38]:

	Weight
Month	
January	6.0000000e+01
February	5.6000000e+01
March	1.1500000e+02
April	1.1200000e+02
May	1.0500000e+02
June	9.8000000e+01
July	8.2000000e+01
August	7.2000000e+01
September	8.5000000e+01
October	8.6000000e+01
November	8.3000000e+01
December	9.0000000e+01

In [39]:

```
dow = df['DayOfWeek'].astype('category')
dow_reorder = dow.cat.reorder_categories(['Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','Sunday'])
df_dow_reorder = pd.DataFrame(dow_reorder)
df_dow_reorder.join(df[['Weight']]).groupby("DayOfWeek").agg(len)
```

Out[39]:

	Weight
DayOfWeek	
Monday	1.4800000e+02
Tuesday	1.5300000e+02
Wednesday	1.5100000e+02
Thursday	1.5400000e+02
Friday	1.4500000e+02
Saturday	1.4600000e+02
Sunday	1.4700000e+02

Q1(b). What is the Residual Sum of Squares for this model Weight ~ Intercept? Give your answer using the ".7E" scientific notation

In [40]:

```
np.set_printoptions(precision = 7, threshold = sys.maxsize)
np.set_printoptions(linewidth = np.inf)

pd.set_option('display.max_columns', None)
pd.set_option('display.expand_frame_repr', False)
pd.set_option('max_colwidth', None)
pd.set_option('precision', 10)
pd.options.display.float_format = '{:,.7e}'.format
```

In [41]:
```
df = df_month_reorder.join(df_dow_reorder).join(df["Weight"])
df_dummies = pd.get_dummies(df_month_reorder.join(df_dow_reorder)).join(df[["Weight"]])
X = df_dummies.drop("Weight",axis=1)
X.insert(0, 'Intercept', 1.0)
y = df_dummies["Weight"]
month_cols= ['Intercept','Month_April', 'Month_August', 'Month_December','Month_February', 'Month_January', 'Month_Ju
dayofweek_cols = ['Intercept','DayOfWeek_Monday','DayOfWeek_Saturday', 'DayOfWeek_Sunday', 'DayOfWeek_Thursday','Day(
param_name = X.columns
```

In [42]:
```
b, residual_SS_b , XtX_Ginv, aliasParam, nonAliasParam = Regression.RegModel(X[["Intercept"]], y)
print('Residual Sum of Squares_b =', "{:.7e}".format(residual_SS_b))
```

```
Residual Sum of Squares_b = 2.2360230e+04
```

Q1(c). What is the Residual Sum of Squares for this model Weight ~ Intercept + Month? Give your answer using the ".7E" scientific notation

In [43]:
```
# Intercept + Month
b_c, residual_SS_c, XtX_Ginv_c, aliasParam_c, nonAliasParam_c = Regression.RegModel(X[month_cols], y)
print('Residual Sum of Squares_c =', "{:.7e}".format(residual_SS_c))
```

```
Residual Sum of Squares_c = 1.7776054e+04
```

Q1(d). What is the Residual Sum of Squares for this model Weight ~ Intercept + DayOfWeek? Give your answer using the ".7E" scientific notation

In [44]:
```
# Intercept + DayofWeek
b_d, residual_SS_d, XtX_Ginv_d, aliasParam_d, nonAliasParam_d = Regression.RegModel(X[dayofweek_cols], y)
print('Residual Sum of Squares_d =', "{:.7e}".format(residual_SS_d))
```

```
Residual Sum of Squares_d = 2.2239170e+04
```

Q1(e). What is the generalized inverse that the SWEEP Operator gives for this model Weight ~ Intercept + DayOfWeek? Give your answer

using the ".7E" scientific notation

In [45]:
```
b_e, residual_SS_e, XtX_Ginv_e, aliasParam_e, nonAliasParam_e = Regression.RegModel(X[dayofweek_cols], y)
print('Generalized Inverse of XtX')
print(XtX_Ginv_e)
```

```
Generalized Inverse of XtX
[[ 0.0068966 -0.0068966 -0.0068966 -0.0068966 -0.0068966 -0.0068966 -0.0068966]
 [-0.0068966  0.0136533  0.0068966  0.0068966  0.0068966  0.0068966  0.0068966]
 [-0.0068966  0.0068966  0.0137459  0.0068966  0.0068966  0.0068966  0.0068966]
 [-0.0068966  0.0068966  0.0068966  0.0136993  0.0068966  0.0068966  0.0068966]
 [-0.0068966  0.0068966  0.0068966  0.0068966  0.0133901  0.0068966  0.0068966]
 [-0.0068966  0.0068966  0.0068966  0.0068966  0.0068966  0.0134325  0.0068966]
 [-0.0068966  0.0068966  0.0068966  0.0068966  0.0068966  0.0068966  0.0135191]]
```

Q1(f). What is the Residual Sum of Squares for this model Weight ~ Intercept + Month + DayOfWeek? Give your answer using the ".7E" scientific notation.

In [46]:
```
b_f, residual_SS_f, XtX_Ginv_f, aliasParam_f, nonAliasParam_f = Regression.RegModel(X, y)
print('Residual Sum of Squares_f =', "{:.7e}".format(residual_SS_f))
```

```
Residual Sum of Squares_f = 1.7665566e+04
```

Q1(g). Which model yields the smallest Residual Sum of Squares

In [47]:
```
print('Weight ~ Intercept + Month + DayOfWeek yields the smallest Residual Sum of Squares, which is',"{:.7e}".format(
```

```
Weight ~ Intercept + Month + DayOfWeek yields the smallest Residual Sum of Squares, which is 1.7665566e+04
```

Q1(h). How many regression parameters (including the aliased parameters) are in this model Weight ~ Intercept + Month + DayOfWeek?

In [48]:
```python
b_h, residual_SS_h, XtX_Ginv_h, aliasParam_h, nonAliasParam_h = Regression.RegModel(X, y)
len(nonAliasParam_h)+len(aliasParam_h)
```

Out[48]:
```
20
```

Q1(i). What are the regression coefficients (including the aliased parameters) of this model Weight ~ Intercept + Month + DayOfWeek? Give your answer using the ".7E" scientific notation.

In [49]:
```python
b_i, residual_SS_i, XtX_Ginv_i, aliasParam_i, nonAliasParam_i = Regression.RegModel(X, y)
beta_i = pd.Series(b_i, index = param_name)
print('Parameter Estimates_i')
print(beta_i)
```

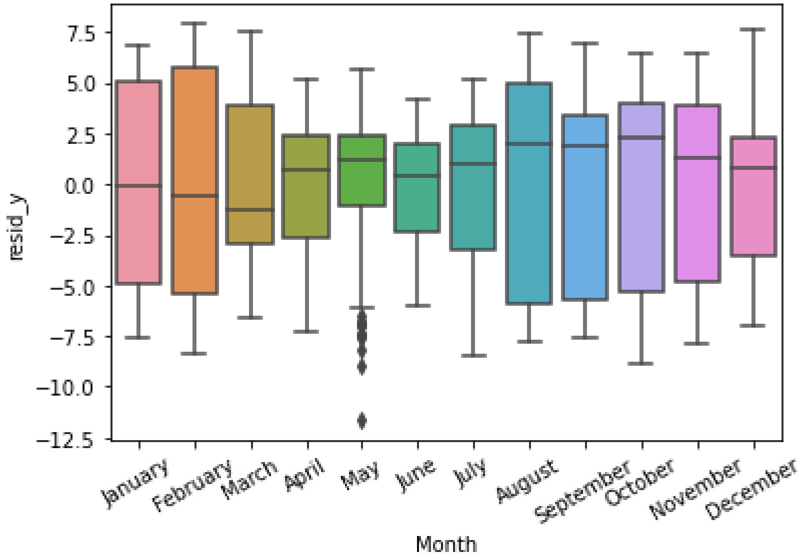
```
Parameter Estimates_i
Intercept              2.1162201e+02
Month_January         -4.4054252e+00
Month_February        -4.8239659e+00
Month_March           -2.8253761e+00
Month_April           -4.2731582e+00
Month_May             -6.4393219e+00
Month_June            -7.1583571e+00
Month_July            -7.1115251e+00
Month_August          -4.8223766e+00
Month_September       -4.0327408e+00
Month_October         -3.3379674e+00
Month_November        -1.5751820e+00
Month_December         0.0000000e+00
DayOfWeek_Monday       2.7295639e-01
DayOfWeek_Tuesday     -3.2245416e-01
DayOfWeek_Wednesday   -5.9404519e-01
DayOfWeek_Thursday    -7.0507218e-01
DayOfWeek_Friday      -5.6643717e-01
DayOfWeek_Saturday    -3.9628664e-01
DayOfWeek_Sunday       0.0000000e+00
dtype: float64
```

Q2(a). Let us focus on this model Weight ~ Intercept + Month + DayOfWeek. Generate a Boxplot of the residuals versus Month. The residuals are on the vertical axis and the Month categories are on the horizontal axis. Also, generate another Boxplot of the residuals

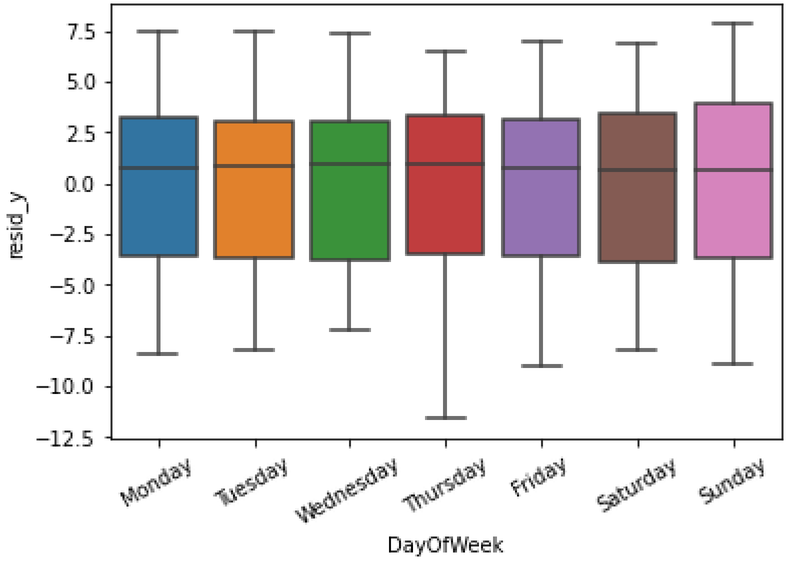
residuals are on the vertical axis and the Month categories are on the horizontal axis. Also, generate another Boxplot of the residuals versus DayOfWeek. Comment on the evidence of heteroskedasticity of the residuals.**

In [50]:

```
predicted_y = np.matmul(X, b_i)
resid_y = predicted_y - y
df["resid_y"] = resid_y

sns.boxplot(x = "Month", y = "resid_y",data = df);
plt.xticks(rotation=30);

### we can see the resiudals of y is not constant, which is the evidence of heteroskedasticity
```



In [51]:
```
predicted_y = np.matmul(X, b_i)
resid_y = predicted_y - y
df["resid_y"] = resid_y

sns.boxplot(x = "DayOfWeek", y = "resid_y",data = df);
plt.xticks(rotation=30);
```



Q2(b). Calculate the Anderson-Darling Test statistic and generate a Normality Q-Q Plot for the residuals. Comment on the evidence of normality (or non-normality) of the residuals

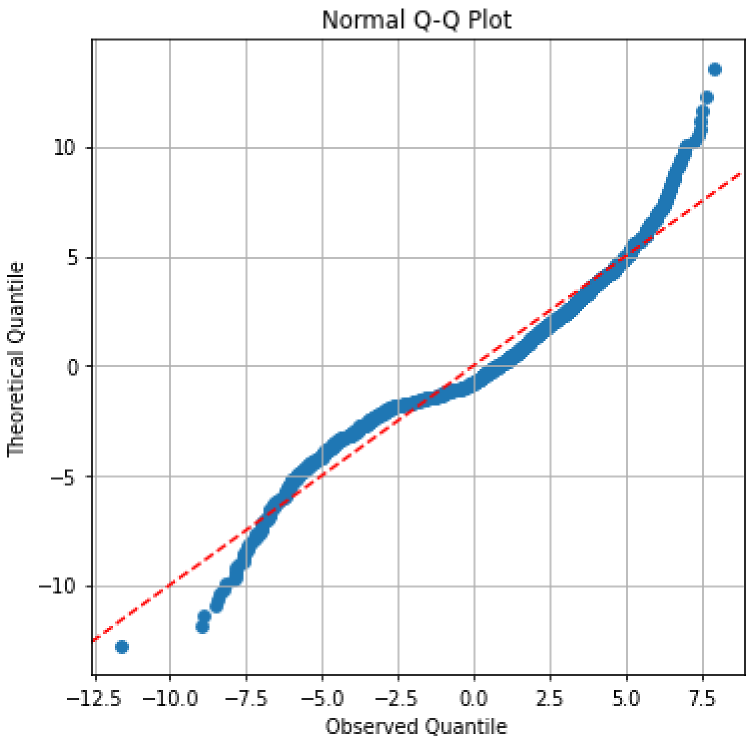
In [52]:

```python
from scipy.stats import anderson

anderson_test = anderson(resid_y, dist = 'norm')
print('  Anderson Test = ', anderson_test[0])
print('Critical Values = ', anderson_test[1])
print('       p-values = ', anderson_test[2]/100.0)

## Since the test statistic is higher than all the critical values, null hypothesis is rejected at all significance l
## i.e there is evidence to suggest that the population doesn’t follow the norm distribution
```

```
  Anderson Test =  15.636691540376887
Critical Values =  [0.574 0.654 0.784 0.915 1.088]
       p-values =  [0.15  0.1   0.05  0.025 0.01 ]
```

In [26]:

```python
y_new = pd.Series(resid_y * resid_y, name = 'Square_Residual')
n_obs = len(y_new)

obs_quantile = np.sort(resid_y)
z_p = np.array(range(n_obs))
z_p = (1.0 + z_p) / (n_obs + 0.5)
z_quantile = norm.ppf(z_p, loc = np.mean(obs_quantile), scale = np.std(obs_quantile))

fig, ax00 = plt.subplots(1, 1, dpi = 70, figsize = (6,6))
ax00.scatter(obs_quantile, z_quantile)
ax00.set_title('Normal Q-Q Plot')
ax00.set_xlabel('Observed Quantile')
ax00.set_ylabel('Theoretical Quantile')
ax00.axline((0,0), slope = 1, color = 'red', linestyle = '--')
ax00.grid(axis = 'both')
plt.show()

## data points not really perfectly lie in a straight line, so not normally distributed
```



Q2(c). Perform the Breusch-Pagan Test and the White Test of Heteroskedasticity. Provide the Chi-square statistics, the degrees of freedom, and the significance values. Comment on the evidence of non-homogenous variance

In [28]:

```python
y_new = pd.Series(resid_y * resid_y, name = 'Square_Residual')
n_obs = len(y_new)
b, SSE0, XtX_Ginv, aliasParam, nonAliasParam = Regression.RegModel(X[['Intercept']], y_new)
b, SSE1, XtX_Ginv, aliasParam, nonAliasParam = Regression.RegModel(X, y_new)

r_squared = 1.0 - (SSE1 / SSE0)

breusch_test = n_obs * r_squared
breusch_df = len(nonAliasParam) - 1
breusch_pvalue = chi2.sf(breusch_test, breusch_df)
print("breusch_test:","{:.7e}".format(breusch_test))
print("breusch_df:","{:.7e}".format(breusch_df))
print("breusch_pvalue:","{:.7e}".format(breusch_pvalue))

## p-value < 0.01 which is very small, null hypothesis is rejected, so it's not non-homogenous
```

```
breusch_test: 2.0935170e+02
breusch_df: 1.7000000e+01
breusch_pvalue: 3.7450746e-35
```

In [31]:

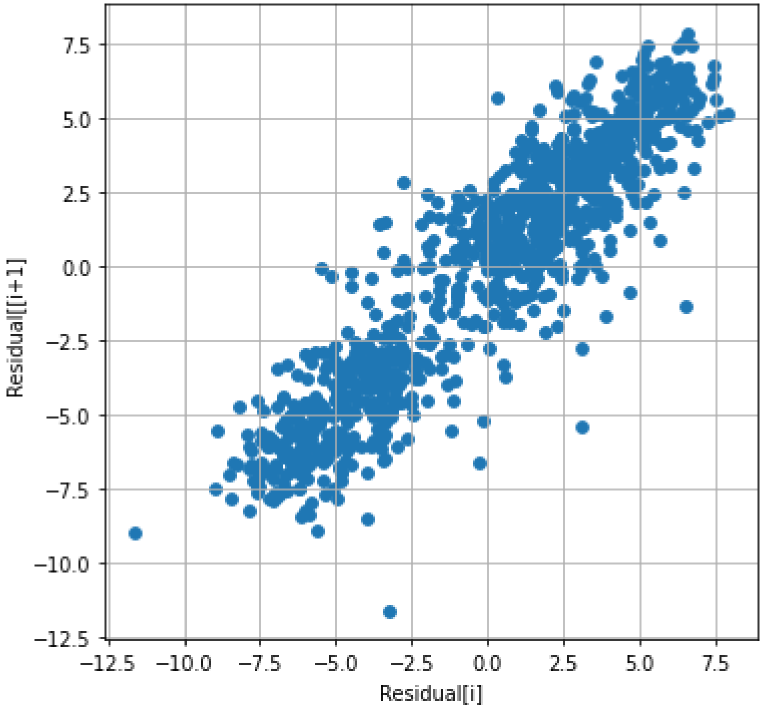
```python
X_new = X

for col1 in month_cols:
    for col2 in dayofweek_cols:
        X_new[col1 + ' ' + col2] = X[col1].multiply(X[col2])

b, SSE0, XtX_Ginv, aliasParam, nonAliasParam = Regression.RegModel(X_new[['Intercept']], y_new)
b, SSE1, XtX_Ginv, aliasParam, nonAliasParam = Regression.RegModel(X_new, y_new)

r_squared = 1.0 - (SSE1 / SSE0)

white_test = n_obs * r_squared
white_df = len(nonAliasParam) - 1
white_pvalue = chi2.sf(white_test, white_df)
print("white_test:","{:.7e}".format(white_test))
print("white_df:","{:.7e}".format(white_df))
print("white_pvalue:","{:.7e}".format(white_pvalue))

# p-value < 0.01 which is very small, null hypothesis is rejected, so it's not non-homogenous
```

```
white_test: 2.3555502e+02
white_df: 8.3000000e+01
white_pvalue: 1.5537170e-16
```

Q2(d). Calculate the Durbin-Watson Test statistic. Comment on the evidence of autocorrelation among observations.

In [32]:

```python
z1 = resid_y[0:(n_obs-1)].to_numpy()
z2 = resid_y[1:n_obs].to_numpy()

fig, ax00 = plt.subplots(1, 1, dpi = 70, figsize = (6,6))
ax00.scatter(z1, z2)
ax00.set_xlabel('Residual[i]')
ax00.set_ylabel('Residual[[i+1]')
ax00.grid(axis = 'both')
plt.show()

z12_corr = np.corrcoef(z1, z2)
print('Autocorrelation = ', '{:.7e}'.format(z12_corr[0,1]))

durbin_watson_test = np.sum((z1-z2)**2) / np.sum(resid_y**2)
print('Durbin-Watson Test', '{:.7e}'.format(durbin_watson_test))


## the DW value is 0.16, indicates positive correlation, and not common(definite cause for concern).
## the scatter plot also shows strongly correlated pattern.
```




```
Autocorrelation =  9.1995391e-01
Durbin-Watson Test 1.6007799e-01
```

Q2(e). Calculate the Shapley values of the two predictors Month and DayOfWeek. Also, provide the Percent Shapley values of the two predictors. Among these two predictors, which one influence the weight more?

In [66]:

```
all_possible_subset = pd.DataFrame({'Index':(['00','10','01','11']),
                                    'Model':('Weight ~ Intercept', 'Weight ~ Intercept + Month',
                                             'Weight ~ Intercept + DayOfWeek',
                                             'Weight ~ Intercept + Month + DayOfWeek'),
                                             'Residual Sum of Squares':(residual_SS_b, residual_SS_c, residual_SS_d,
                                   })


all_possible_subset['Coefficient of Determination (R2)'] =1-all_possible_subset.iloc[:,2]/all_possible_subset.iloc[0,
all_possible_subset
```

Out[66]:

	Index	Model	Residual Sum of Squares	Coefficient of Determination (R2)
0	00	Weight ~ Intercept	2.2360230e+04	0.0000000e+00
1	10	Weight ~ Intercept + Month	1.7776054e+04	2.0501468e-01
2	01	Weight ~ Intercept + DayOfWeek	2.2239170e+04	5.4140346e-03
3	11	Weight ~ Intercept + Month + DayOfWeek	1.7665566e+04	2.0995596e-01

In [78]:

```
grand_coalition = pd.DataFrame({'Pred Sequence': ('Month,DayOfWeek', 'DayOfWeek,Month')})
grand_coalition['Predictor 1'] = [2.0501468e-01, 5.4140346e-03]
grand_coalition['Predictor 2'] = [2.0995596e-01, 2.0995596e-01]
grand_coalition['Month Contributes'] = (grand_coalition.iloc[0,1], grand_coalition.iloc[1,2] - grand_coalition.iloc[1
grand_coalition['DayOfWeek Contributes'] = (grand_coalition.iloc[0,2] - grand_coalition.iloc[0,1], grand_coalition.il
grand_coalition
```

Out[78]:

	Pred Sequence	Predictor 1	Predictor 2	Month Contributes	DayOfWeek Contributes
0	Month,DayOfWeek	2.0501468e-01	2.0995596e-01	2.0501468e-01	4.9412800e-03
1	DayOfWeek,Month	5.4140346e-03	2.0995596e-01	2.0454193e-01	5.4140346e-03

In [79]:

```
M_mean = grand_coalition['Month Contributes'].mean()
D_mean = grand_coalition['DayOfWeek Contributes'].mean()
total_mean = M_mean + D_mean
```

In [80]:

```python
Shap = pd.DataFrame({'Predictor': ('Month', 'DayOfWeek')})
Shap['Shapley Value'] = (M_mean, D_mean)
Shap.loc[0, 'Percent Shapley Value'] = '{:.3f}%'.format(M_mean / total_mean * 100)
Shap.loc[1, 'Percent Shapley Value'] = '{:.3f}%'.format(D_mean / total_mean * 100)
Shap

## month weights more
```

Out[80]:

	Predictor	Shapley Value	Percent Shapley Value
0	Month	2.0477830e-01	97.534%
1	DayOfWeek	5.1776573e-03	2.466%