Q1: Time Series Analysis:

```
import pandas as pd
from pandas import read_csv

data = read_csv('https://raw.githubusercontent.com/ourcodingclub/CC-time-series/master/monthly_milk.csv',
                index_col ='month',
                parse_dates = True)
```

a. Find first five values of time series data (Monthly milk)

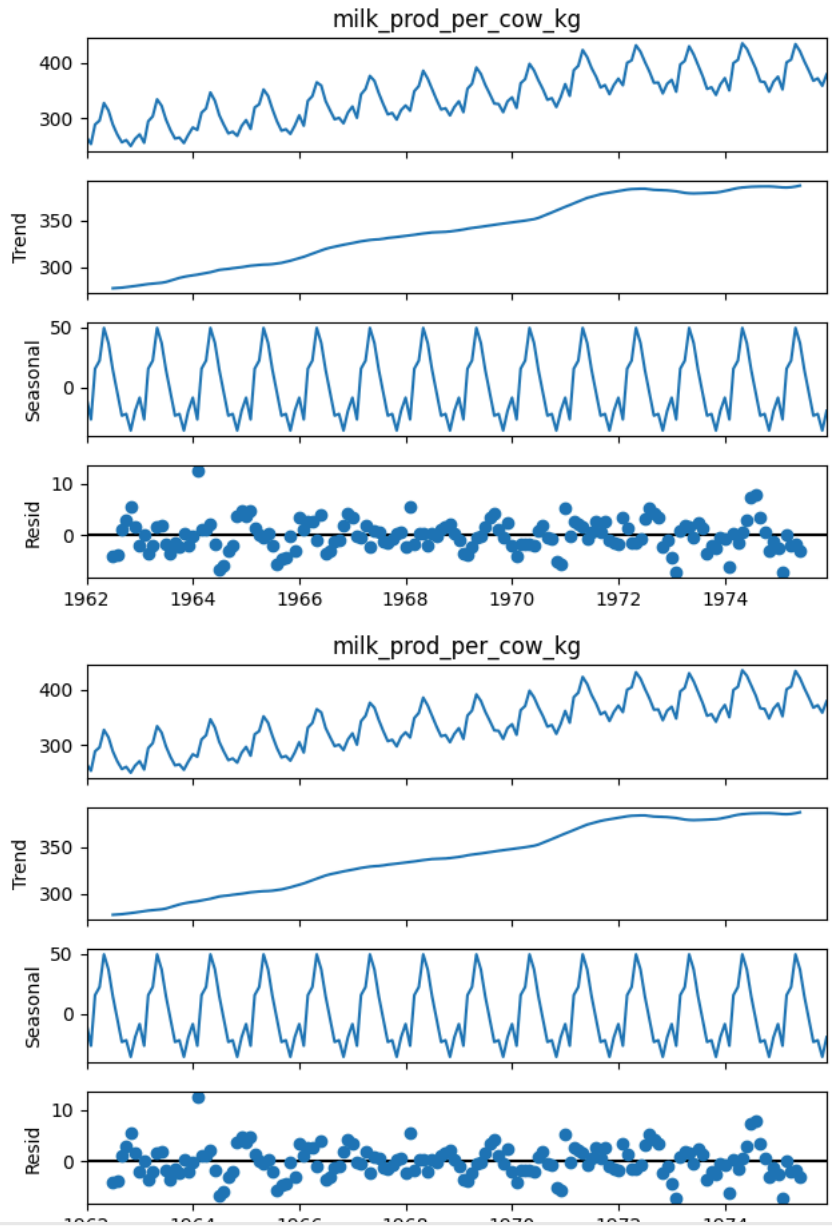
```
data.head()
```

month	milk_prod_per_cow_kg
1962-01-01	265.05
1962-02-01	252.45
1962-03-01	288.00
1962-04-01	295.20
1962-05-01	327.15

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

b. Fit ARIMA model in the time series data set and predict for 2 years.

```
from statsmodels.tsa.seasonal import seasonal_decompose

# ETS Decomposition
result = seasonal_decompose(data['milk_prod_per_cow_kg'],
                            model ='Additive')

# ETS plot
result.plot()
```




```
!pip install pmdarima
```

Show hidden output

```
import matplotlib.pyplot as plt
from pmdarima import auto_arima

import warnings
warnings.filterwarnings("ignore")

stepwise_fit = auto_arima(data['milk_prod_per_cow_kg'],
                          start_p = 1,
                          start_q = 1,
                          max_p = 3,
                          max_q = 3,
                          m = 12,
                          start_P = 0,
                          seasonal = True,
                          d = None,
```

```
                           D = 1,
                           trace = True,
                           error_action ='ignore',
                           suppress_warnings = True,
                           stepwise = True)

stepwise_fit.summary()
```

```
Performing stepwise search to minimize aic
 ARIMA(1,0,1)(0,1,1)[12] intercept   : AIC=823.289, Time=0.66 sec
 ARIMA(0,0,0)(0,1,0)[12] intercept   : AIC=1075.573, Time=0.05 sec
 ARIMA(1,0,0)(1,1,0)[12] intercept   : AIC=843.201, Time=0.38 sec
 ARIMA(0,0,1)(0,1,1)[12] intercept   : AIC=962.793, Time=0.38 sec
 ARIMA(0,0,0)(0,1,0)[12]             : AIC=1203.299, Time=0.06 sec
 ARIMA(1,0,1)(0,1,0)[12] intercept   : AIC=870.000, Time=0.19 sec
 ARIMA(1,0,1)(1,1,1)[12] intercept   : AIC=825.170, Time=0.77 sec
 ARIMA(1,0,1)(0,1,2)[12] intercept   : AIC=825.147, Time=1.42 sec
 ARIMA(1,0,1)(1,1,0)[12] intercept   : AIC=839.342, Time=0.53 sec
 ARIMA(1,0,1)(1,1,2)[12] intercept   : AIC=inf, Time=7.51 sec
 ARIMA(1,0,0)(0,1,1)[12] intercept   : AIC=826.588, Time=0.38 sec
 ARIMA(2,0,1)(0,1,1)[12] intercept   : AIC=824.908, Time=0.80 sec
 ARIMA(1,0,2)(0,1,1)[12] intercept   : AIC=824.202, Time=0.71 sec
 ARIMA(0,0,0)(0,1,1)[12] intercept   : AIC=1075.819, Time=0.29 sec
 ARIMA(0,0,2)(0,1,1)[12] intercept   : AIC=918.974, Time=0.64 sec
 ARIMA(2,0,0)(0,1,1)[12] intercept   : AIC=822.921, Time=0.53 sec
 ARIMA(2,0,0)(0,1,0)[12] intercept   : AIC=869.597, Time=0.16 sec
 ARIMA(2,0,0)(1,1,1)[12] intercept   : AIC=824.788, Time=0.76 sec
 ARIMA(2,0,0)(0,1,2)[12] intercept   : AIC=824.764, Time=1.25 sec
 ARIMA(2,0,0)(1,1,0)[12] intercept   : AIC=838.597, Time=0.51 sec
 ARIMA(2,0,0)(1,1,2)[12] intercept   : AIC=inf, Time=7.31 sec
 ARIMA(3,0,0)(0,1,1)[12] intercept   : AIC=824.888, Time=0.66 sec
 ARIMA(3,0,1)(0,1,1)[12] intercept   : AIC=823.661, Time=1.42 sec
 ARIMA(2,0,0)(0,1,1)[12]             : AIC=827.433, Time=0.50 sec

Best model:  ARIMA(2,0,0)(0,1,1)[12] intercept
Total fit time: 27.898 seconds
```

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	168
Model:	SARIMAX(2, 0, 0)x(0, 1, [1], 12)	**Log Likelihood**	-406.460
Date:	Fri, 15 Nov 2024	**AIC**	822.921
Time:	17:36:35	**BIC**	838.170
Sample:	01-01-1962	**HQIC**	829.114
	- 12-01-1975		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	0.7228	0.407	1.776	0.076	-0.075	1.520
ar.L1	0.7309	0.084	8.741	0.000	0.567	0.895
ar.L2	0.1888	0.080	2.359	0.018	0.032	0.346
ma.S.L12	-0.6167	0.073	-8.405	0.000	-0.761	-0.473
sigma2	10.2694	0.995	10.318	0.000	8.319	12.220

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	42.73
Prob(Q):	0.97	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.83	**Skew:**	0.78
Prob(H) (two-sided):	0.52	**Kurtosis:**	5.03

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step)

Using SARIMAX

```
from statsmodels.tsa.statespace.sarimax import SARIMAX

model_SARIMAX = SARIMAX(data['milk_prod_per_cow_kg'],
                order = (2,0,0),
                seasonal_order =(0, 1, 1, 12))

result = model_SARIMAX.fit()
result.summary()
```

SARIMAX Results

Dep. Variable:	milk_prod_per_cow_kg	**No. Observations:**	168
Model:	SARIMAX(2, 0, 0)x(0, 1, [1], 12)	**Log Likelihood**	-409.717
Date:	Fri, 15 Nov 2024	**AIC**	827.433
Time:	17:44:00	**BIC**	839.633
Sample:	01-01-1962 - 12-01-1975	**HQIC**	832.388

Covariance Type: opg

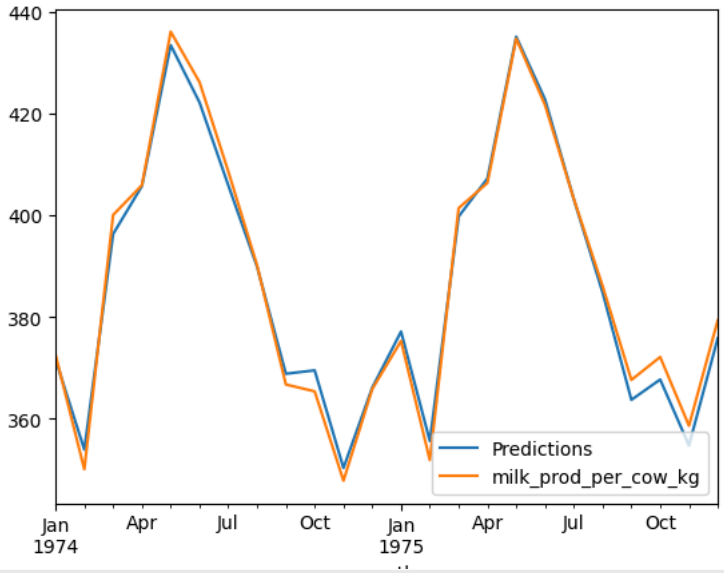
	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.7699	0.078	9.879	0.000	0.617	0.923
ar.L2	0.2221	0.076	2.904	0.004	0.072	0.372
ma.S.L12	-0.6145	0.071	-8.616	0.000	-0.754	-0.475
sigma2	10.6326	1.030	10.321	0.000	8.613	12.652

Ljung-Box (L1) (Q):	0.02	**Jarque-Bera (JB):**	37.20
Prob(Q):	0.90	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.80	**Skew:**	0.74
Prob(H) (two-sided):	0.43	**Kurtosis:**	4.87

Warnings:

```
train = data.iloc[:len(data)-24]
test = data.iloc[len(data)-24:]

start = len(train)
end = len(train) + len(test) - 1

predictions = result.predict(start, end,
                             typ = 'levels').rename("Predictions")

predictions.plot(legend = True)
test['milk_prod_per_cow_kg'].plot(legend = True)
```

```
<Axes: xlabel='month'>
```

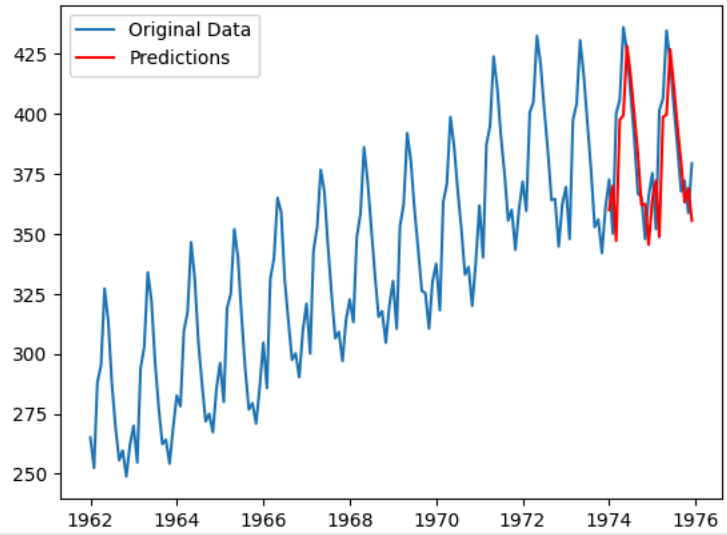


Using ARIMA

```
from statsmodels.tsa.arima.model import ARIMA
from pandas import DataFrame
from matplotlib import pyplot

model_ARIMA = ARIMA(data['milk_prod_per_cow_kg'], order=(2,0,0))
```

```
model_fit = model_ARIMA.fit()

predictions = model_fit.predict(start, end)
print(predictions)

pyplot.plot(data['milk_prod_per_cow_kg'], label='Original Data')
pyplot.plot(predictions, color='red', label='Predictions')
pyplot.legend()
pyplot.show()
```

```
1974-01-01    360.024701
1974-02-01    369.946099
1974-03-01    347.110642
1974-04-01    397.533788
1974-05-01    399.328535
1974-06-01    428.245168
1974-07-01    416.243120
1974-08-01    399.918757
1974-09-01    383.315574
1974-10-01    361.960158
1974-11-01    362.474123
1974-12-01    345.481393
1975-01-01    364.389984
1975-02-01    372.189651
1975-03-01    348.653247
1975-04-01    398.708330
1975-05-01    399.661421
1975-06-01    426.894740
1975-07-01    411.964482
1975-08-01    395.009526
1975-09-01    379.791948
1975-10-01    363.153587
1975-11-01    368.980023
1975-12-01    355.475742
Freq: MS, Name: predicted_mean, dtype: float64
```



Q2 : Linear Regression Problem:

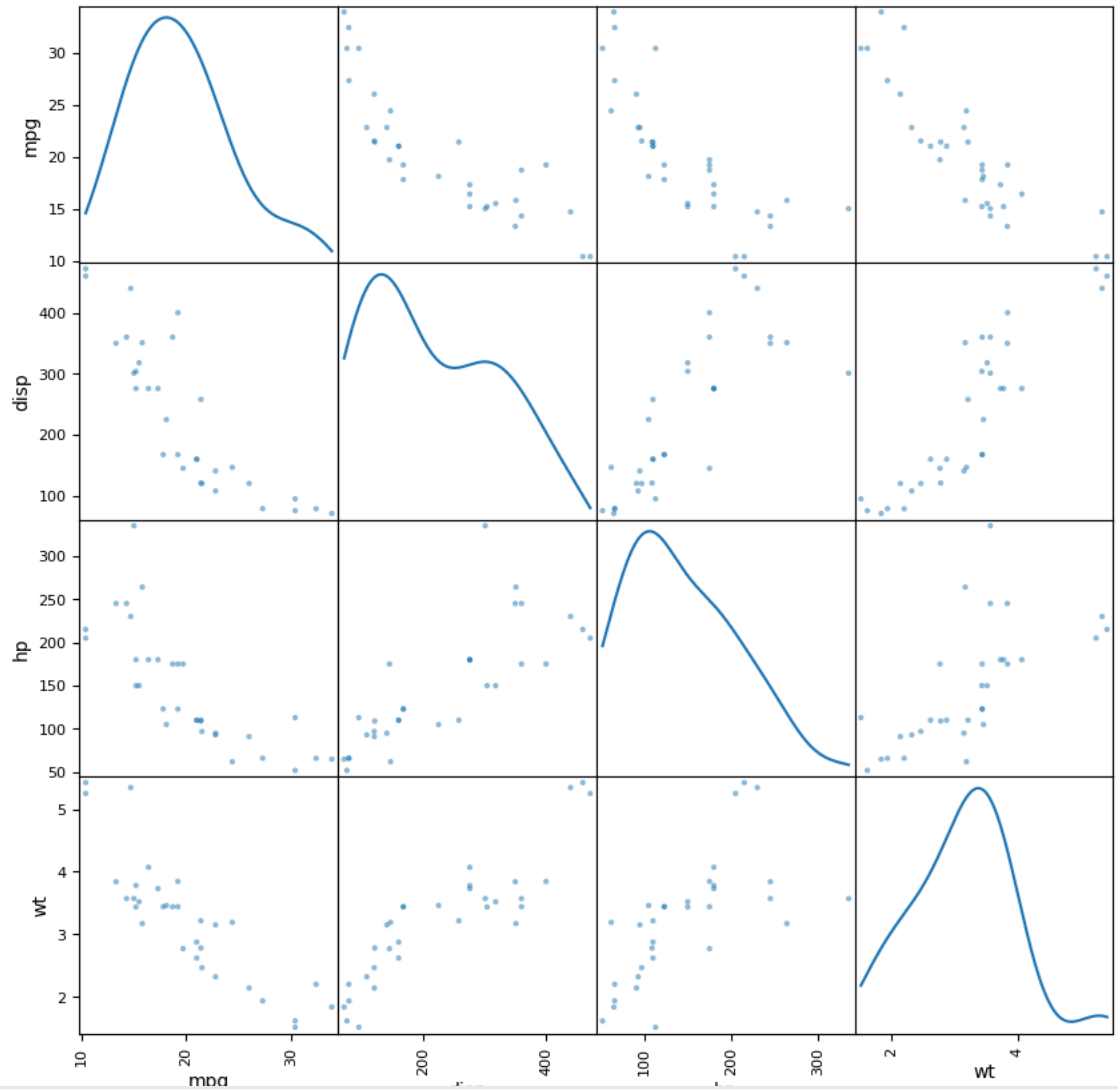
```
df = pd.read_csv('mtcars.csv')

df.head()
```

	model	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1

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

a. Create a subset of the variables mpg, disp, hp and wt from the mtcars data. Construct the scatter plot for these attributes.

```
import matplotlib.pyplot as plt

subset = df[['mpg', 'disp', 'hp', 'wt']]

pd.plotting.scatter_matrix(subset, figsize=(10, 10), diagonal='kde')
plt.show()
```



b. Fit a multiple linear regression model with mpg as the response variable and disp and hp as the input variables

```
import statsmodels.formula.api as sm

# Define the model formula
formula = 'mpg ~ disp + hp'

# Fit the model
model = sm.ols(formula=formula, data=df).fit()

print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    mpg   R-squared:                       0.748
Model:                            OLS   Adj. R-squared:                  0.731
Method:                 Least Squares   F-statistic:                     43.09
Date:                Fri, 15 Nov 2024   Prob (F-statistic):           2.06e-09
Time:                        17:58:04   Log-Likelihood:                -80.309
No. Observations:                  32   AIC:                             166.6
Df Residuals:                      29   BIC:                             171.0
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     30.7359      1.332     23.083      0.000      28.013      33.459
disp          -0.0303      0.007     -4.098      0.000      -0.045      -0.015
hp            -0.0248      0.013     -1.856      0.074      -0.052       0.003
==============================================================================
Omnibus:                        3.082   Durbin-Watson:                   1.370
Prob(Omnibus):                  0.214   Jarque-Bera (JB):                2.788
Skew:                           0.680   Prob(JB):                        0.248
Kurtosis:                       2.508   Cond. No.                         733.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

c. Check the p-values to evaluate the significance of the input variables.

```
# From the 'P>|t|' column in the output of model.summary().

# p value for 'disp': 0.000, less that 0.05. There's enough evidence to reject the null hy

# p value for 'hp': 0.074, greater than 0.05. There's not enough evidence to reject the nu
```

d. Which independent variable is the least significant? Why?