```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from keras import layers
from keras.models import Sequential
```

Double-click (or enter) to edit

```
dataset_train = pd.read_csv('trainset.csv')
```

```
dataset_train.columns
```

```
Index(['Date', 'Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'],
dtype='object')
```

```
dataset_train.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2013-01-02	357.385559	361.151062	355.959839	359.288177	359.288177	5115500
1	2013-01-03	360.122742	363.600128	358.031342	359.496826	359.496826	4666500
2	2013-01-04	362.313507	368.339294	361.488861	366.600616	366.600616	5562800

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

```
train_set = dataset_train.iloc[:,1:2].values
```

```
type(train_set)
```

```
numpy.ndarray
```

```
train_set.shape
```

```
(1259, 1)
```

```
sc = MinMaxScaler(feature_range=(0,1))
training_set_scaled = sc.fit_transform(train_set)
```

```
training_set_scaled.shape
```

```
(1259, 1)
```

```
X_train_array = []
y_train_array = []
for i in range(60, 1259):
  X_train_array.append(training_set_scaled[i-60:i,0])
  y_train_array.append(training_set_scaled[i,0])
X_train, y_train = np.array(X_train_array), np.array(y_train_array)
X_train1 = X_train.reshape((X_train.shape[0], X_train.shape[1],1))
```

```
X_train.shape
```

```
(1199, 60)
```

```
length = 60
n_features = 1
```

```
model = Sequential([layers.SimpleRNN(50,input_shape=(60,1)),
                    layers.Dense(1)])
model.compile(optimizer='adam',loss='mse')
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204: Us
  super().__init__(**kwargs)
```

```
print("Name:  JAGAN A     Register Number: 212221230037         ")
model.summary()
```

```
Name:  JAGAN A     Register Number: 212221230037
Model: "sequential"
```

Layer (type)	Output Shape	
simple_rnn (SimpleRNN)	(None, 50)	
dense (Dense)	(None, 1)	

```
Total params: 2,651 (10.36 KB)
Trainable params: 2,651 (10.36 KB)
Non-trainable params: 0 (0.00 B)
```

```
model.fit(X_train1,y_train,epochs=20, batch_size=32)
```

```
Epoch 1/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.1057
Epoch 2/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0021
Epoch 3/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0016
Epoch 4/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0016
Epoch 5/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0015
Epoch 6/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0013
Epoch 7/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0011
Epoch 8/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0011
Epoch 9/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - loss: 0.0012
Epoch 10/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - loss: 0.0010
Epoch 11/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0011
Epoch 12/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 1s 20ms/step - loss: 9.5208e-04
Epoch 13/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 1s 19ms/step - loss: 0.0011
Epoch 14/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 0.0010
Epoch 15/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 8.9687e-04
Epoch 16/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 9.5733e-04
Epoch 17/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 8.6068e-04
Epoch 18/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 8.9019e-04
Epoch 19/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 7.6562e-04
Epoch 20/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - loss: 9.4975e-04
<keras.src.callbacks.history.History at 0x7a1e183e2b60>
```

```
dataset_test = pd.read_csv('testset.csv')
```

```
test_set = dataset_test.iloc[:,1:2].values
```

```
test_set.shape
```

```
(125, 1)
```

```
dataset_total = pd.concat((dataset_train['Open'],dataset_test['Open']),axis=0)


inputs = dataset_total.values
inputs = inputs.reshape(-1,1)
inputs_scaled=sc.transform(inputs)
X_test = []
for i in range(60,1384):
  X_test.append(inputs_scaled[i-60:i,0])
X_test = np.array(X_test)
X_test = np.reshape(X_test,(X_test.shape[0], X_test.shape[1],1))


X_test.shape
```

```
(1324, 60, 1)
```

```
predicted_stock_price_scaled = model.predict(X_test)
predicted_stock_price = sc.inverse_transform(predicted_stock_price_scaled)
```

```
42/42 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
```

```
print("Name:   JAGAN A        Register Number:  212221230037   ")
plt.plot(np.arange(0,1384),inputs, color='red', label = 'Test(Real) Google stoc
plt.plot(np.arange(60,1384),predicted_stock_price, color='blue', label = 'Predi
plt.title('Google Stock Price Prediction')
plt.xlabel('Time')
plt.ylabel('Google Stock Price')
plt.legend()
plt.show()
```

```
Name:   JAGAN A        Register Number:  212221230037
```