Student name:

Student ID:

# SIT225: Data Capture Technologies

## Activity 5.1: Firebase Realtime database

The Firebase Realtime Database is a cloud-hosted NoSQL database that lets you store and sync data between your users in real-time. Data is stored as JSON and synchronized in real-time to every connected client. In this activity, you will set up and perform operations such as queries and updates on the database using Python programming language.

### Hardware Required

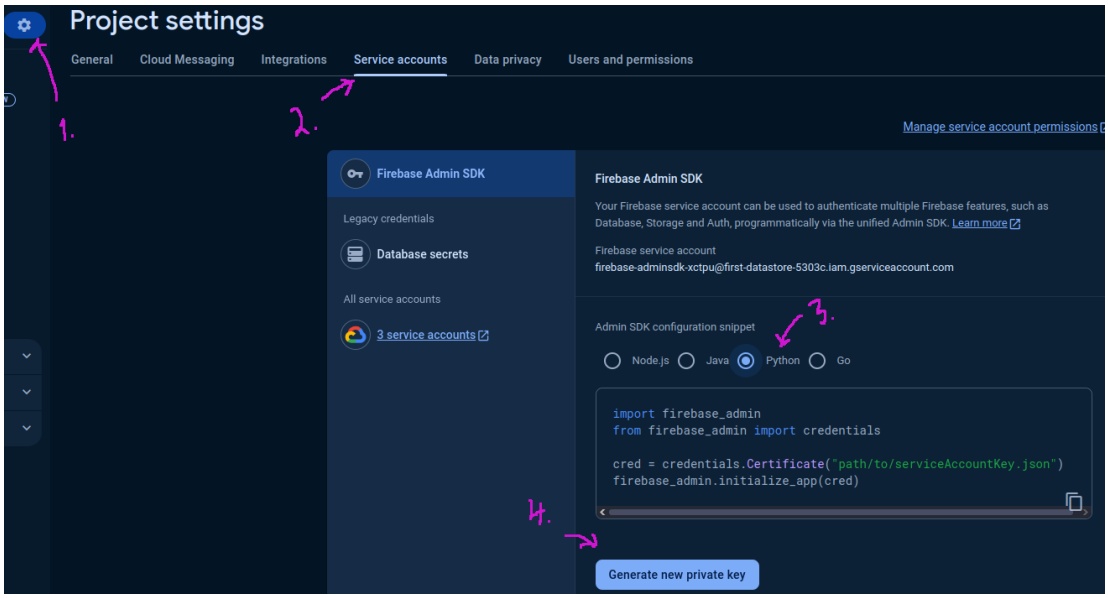
No hardware is required.

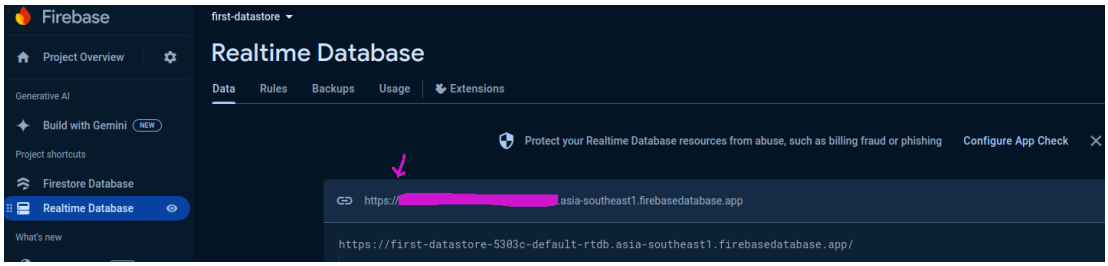
### Software Required

Firebase Realtime database
Python 3

### Steps

| Step | Action |
|---|---|
| 1 | **Create an Account**:<br>First, you will need to create an account in the Firebase console, follow instructions in the official Firebase document (https://firebase.google.com/docs/database/rest/start ). |
| 2 | **Create a Database**:<br>Follow the above Firebase document to create a database. When you click on Create Database, you have to specify the location of the database and the security rules. Two rules are available – locked mode and test mode; since we will be using the database for reading, writing, and editing, we choose test mode. |
| 3 | **Setup Python library for Firebase access**:<br>We will be using Admin Database API, which is available in *firebase_admin* library. Use the below command in the command line to install. You can follow |

| | |
|---|---|
| | a Firebase tutorial here (https://www.freecodecamp.org/news/how-to-get-started-with-firebase-using-python ).<br><br>$ pip install firebase_admin<br><br>Firebase will allow access to Firebase server APIs from Google Service Accounts. To authenticate the Service Account, we require a private key in JSON format. To generate the key, go to project settings, click Generate new private key, download the file, and place it in your current folder where you will create your Python script.<br><br>  |
| 4 | **Connect to Firebase using Python version of Admin Database API**:<br>A credential object needs to be created to initialise the Python library which can be done using the Python code below. Python notebook can be downloaded here (https://github.com/deakin-deep-dreamer/sit225/blob/main/week_5/firebase_explore.ipynb ).<br><br>`import firebase_admin`<br><br>`databaseURL = 'https://XXX.firebasedatabase.app/'`<br>`cred_obj = firebase_admin.credentials.Certificate(`<br>`'first-datastore-5303c-firebase-adminsdk-xctpu-c9902044ac.json'`<br>`)`<br>`default_app = firebase_admin.initialize_app(cred_obj, {`<br>`'databaseURL':databaseURL`<br>`})`<br><br>The databaseURL is a web address to reach your Firebase database that you have created in step 2. This URL can be found in the Data tab of Realtime Database. |




If you compile the code snippet above, it should do with no error.

5 **Write to database Using the set() Function**:

We set the reference to the root of the database (or we could also set it to a key value or child key value). Data needs to be in JSON format as below.

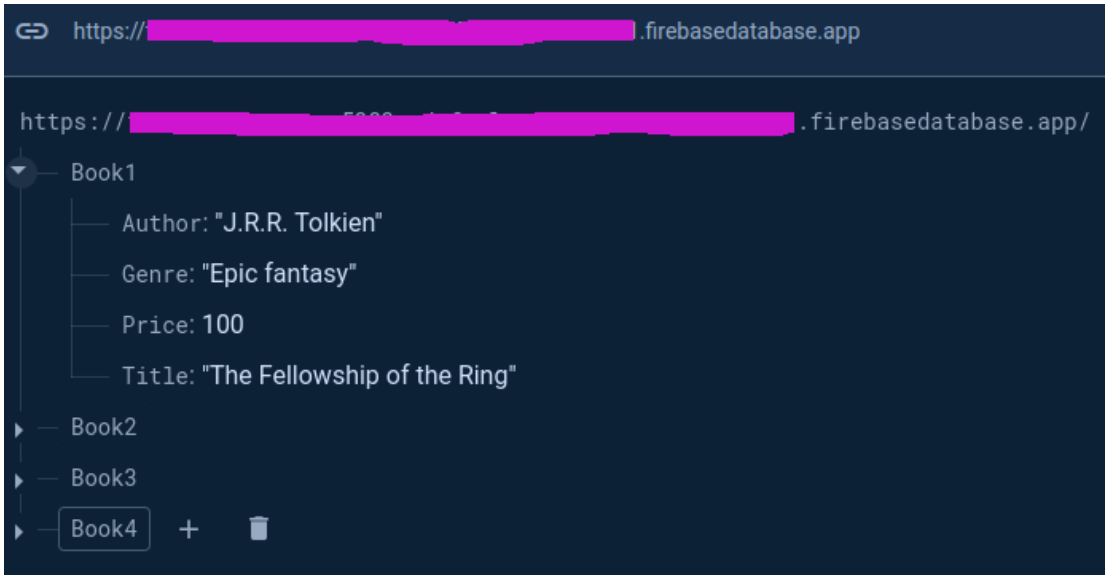
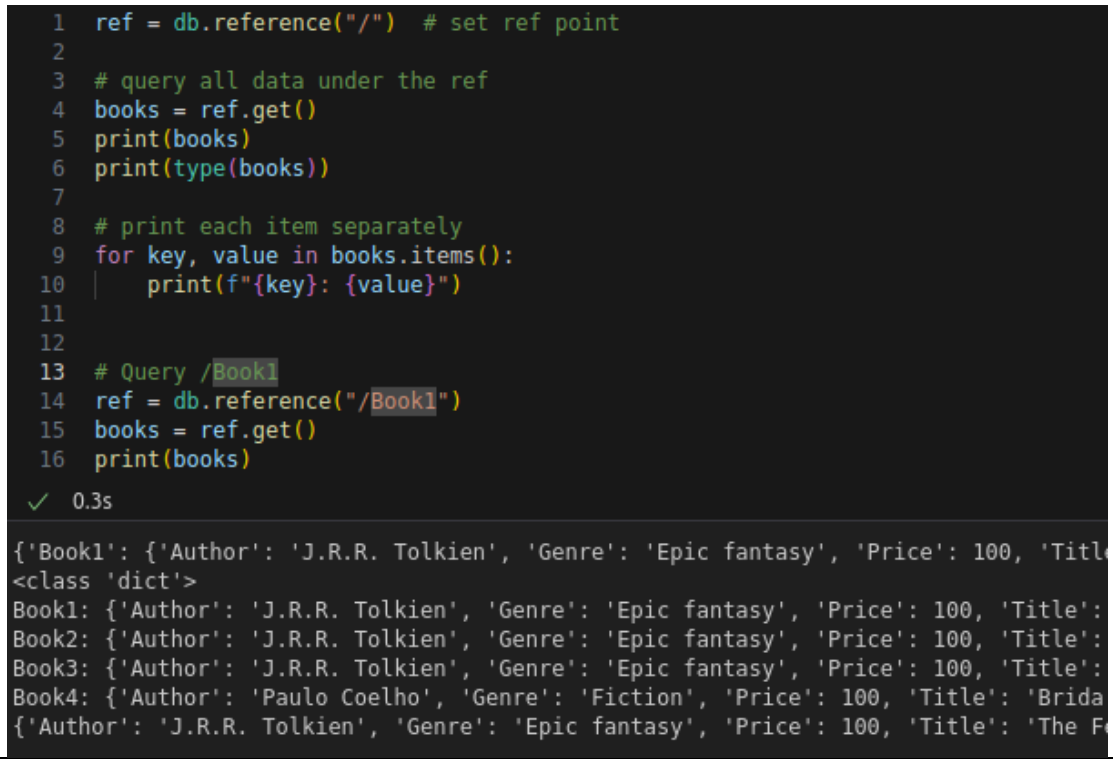
```
from firebase_admin import db

# A reference point is always needed to be set
# before any operation is carried out on a database.
#
ref = db.reference("/")

# JSON format data (key/value pair)
data = {  # Outer {} contains inner data structure
    "Book1":
    {
        "Title": "The Fellowship of the Ring",
        "Author": "J.R.R. Tolkien",
        "Genre": "Epic fantasy",
        "Price": 100
    },
    "Book2":
    {
        "Title": "The Two Towers",
        "Author": "J.R.R. Tolkien",
        "Genre": "Epic fantasy",
        "Price": 100
    },
    "Book3":
    {
        "Title": "The Return of the King",
        "Author": "J.R.R. Tolkien",
        "Genre": "Epic fantasy",
        "Price": 100
    },
    "Book4":
    {
        "Title": "Brida",
        "Author": "Paulo Coelho",
        "Genre": "Fiction",
        "Price": 100
    }
}

# JSON format data is set (overwritten) to the reference
# point set at /, which is the root node.
#
ref.set(data)
```

A reference point always needed to be set where the data read/write will take place. In the code above, the reference point is set at the root of the NoSQL Document, where consider the database is a JSON tree and / is the root node

| | |
|---|---|
| | of the tree). The set() function writes (overwrites) data at the set reference point.<br><br>You can visualise the data in the Firebase console as below -<br><br>  |
| 6 | **Read data using get() function**:<br>Data can be read using get() function on the reference set beforehand, as shown below.<br><br>  |

| | |
|---|---|
| | Consider the reference set in line 1 and the output compared to the reference set at line 14 and the bottom output line to understand the use of db.reference() and ref.get(). |
| 7 | **Write to database Using the push() Function**:<br>The push() function saves data under a *unique system generated key*. This is different than set() where you set the keys such as Book1, Book2, Book3 and Book4 under which the content (author, genre, price and title) appears. Let's try to push the same data in the root reference. Note that since we already has data under root / symbol, setting (or pushing) in the same reference point will eventually rewrite the original data.<br><br>`# Write using push() function`<br>`# Note that a set() is called on top of push()`<br>`#`<br>`ref = db.reference("/")`<br>`ref.set({`<br>`  "Books":`<br>`  {`<br>`  "Best_Sellers": -1`<br>`  }`<br>`})`<br><br>`ref = db.reference("/Books/Best_Sellers")`<br><br>`for key, value in data.items():`<br>`  ref.push().set(value)`<br>✓ 2.0s<br><br>The output will reset the previous data set in / node. The current data is shown below. |


As you can see, under /Books/Best_Sellers there are 4 nodes where the node head (or node ID) is a randomly generated key which is due to the use of push() function. When data key does not matter, the use of push() function desirable.

8

**Update data**:
Let's say the price of the books by J. R. R. Tolkien is reduced to 80 units to offer a discount. The first 3 books are written by this author, and we want to apply for a discount on all of them.

```
# Update data
#
# Requirement: The price of the books by
# J. R. R. Tolkien is reduced to 80 units to
# offer a discount.
#
ref = db.reference("/Books/Best_Sellers/")
best_sellers = ref.get()
print(best_sellers)
for key, value in best_sellers.items():
    if(value["Author"] == "J.R.R. Tolkien"):
        value["Price"] = 90
        ref.child(key).update({"Price":80})
```

✓ 0.9s

As you can see, the author name is compared and the new price is set in the best_sellers dictionary and finally, an update() function is called on the ref, however, the current ref is a '/Books/Best_Sellers/', so we need to locate the

| | |
|---|---|
| | child under the ref node, so ref.child(key) is used in line 13. The output is shown below with a discounted price.<br><br>  |
| 9 | **Delete data**:<br>Let's delete all bestseller books with J.R.R. Tolkien as the author. You can locate the node using db.reference() (line 4) and then locate specific record (for loop in line 6) and calling set() with empty data {} as a parameter, such as set({}). The particular child under the ref needs to be located first by using ref.child(key), otherwise, the ref node will be removed – BE CAREFUL. |

| | |
|---|---|
| | ```python<br># Let's delete all best seller books<br># with J.R.R. Tolkien as the author.<br>#<br>ref = db.reference("/Books/Best_Sellers")<br><br>for key, value in best_sellers.items():<br>    if(value["Author"] == "J.R.R. Tolkien"):<br>        ref.child(key).set({})<br>```<br>This keeps only the other author data, as shown below.<br><br>  <br><br>If ref.child() not used, as shown the code below, all data will be removed.<br><br>```python<br>ref = db.reference("/Books/Best_Sellers")<br>ref.set({})<br>```<br>Now in Firebase console you will see no data exists. |
| 10 | Question: Run all the cells in the Notebook you have downloaded in Step 4, fill in the student information at the top cell of the Notebook. Convert the Notebook to PDF and merge with this activity sheet PDF.<br><br>Answer: Convert the Notebook to PDF and merge with this activity sheet PDF. |
| 10 | Question: Create a sensor data structure for DHT22 sensor which contains attributes such as sensor_name, timestamp, temperature and humidity. Remember there will be other sensors with different sensor variables such as DHT22 has 2 variables, accelerometer sensor has 3. For each such sensor, you will need to gather data over time. Discuss how you are going to handle multiple data values in JSON format? Justify your design.<br><br>Answer: I should have create a data structure for specific sensor and push the data to the firebase with correct location. For example, the DHT22 sensor will have DHT22 |

| | |
|---|---|
| | for the main and push the data contain timestamp, temperature and humidity to that, since we push so that we don't need to care about the key. |
| 11 | Question: Generate some random data for DHT22 sensor, insert data to database, query all data and screenshot the output here.<br><br>Answer:<br>1]:<br>`ref = db.reference("/")`<br><br>`sensor_data = {`<br>`"DHT22": [ {`<br>`"timestamp": "2024-08-06T12:00:00Z",`<br>`"temperature": 25.6,`<br>`"humidity": 60.2`<br>`},`<br>`{`<br>`"timestamp": "2024-08-06T12:10:00Z",`<br>`"temperature": 25.8,`<br>`"humidity": 59.8`<br>`}`<br>`]`<br>`}`<br>`ref.set(sensor_data)`<br>`ref = db.reference("/DHT22/")`<br>`before = ref.get()`<br>`print(before)`<br>`ref.push().set(`<br>`{`<br>`"timestamp": "2024-08-06T12:10:00Z",`<br>`"temperature": 25.8,`<br>`"humidity": 59.8`<br>`}`<br>`)`<br>`after = ref.get()`<br>`print(after)`<br><br>`[{'humidity': 60.2, 'temperature': 25.6, 'timestamp': '2024-08-06T12:00:00Z'}, {'humidity': 59.8, 'temperature': 25.8, 'timestamp': '2024-08-06T12:10:00Z'}]`<br>`{'0': {'humidity': 60.2, 'temperature': 25.6, 'timestamp': '2024-08-06T12:00:00Z'}, '1': {'humidity': 59.8, 'temperature': 25.8, 'timestamp': '2024-08-06T12:10:00Z'}, '-O3bbkNIjHYYuJe8VWJ9': {'humidity': 59.8, 'temperature': 25.8, 'timestamp': '2024-08-06T12:10:00Z'}}`<br>]: |
| 12 | Question: Generate some random data for the SR04 Ultrasonic sensor, insert data to database, query all data and screenshot the output here.<br><br>Answer:<br>`ref = db.reference("/")`<br><br>`sensor_data = {`<br>`"HCSR04": [ {`<br>`"timestamp": "2024-08-06T12:00:00Z",`<br>`"duration": 450,`<br>`"distance": 25`<br>`},`<br>`{`<br>`"timestamp": "2024-08-06T12:10:00Z",`<br>`"duration": 600,`<br>`"distance": 40`<br>`}`<br>`]`<br>`}`<br>`ref.set(sensor_data)`<br>`ref = db.reference("/HCSR04/")`<br>`before = ref.get()`<br>`print(before)`<br>`ref.push().set(`<br>`{`<br>`"timestamp": "2024-08-06T12:10:00Z",`<br>`"duration": 500,`<br>`"distance": 30`<br>`}`<br>`)`<br>`after = ref.get()`<br>`for key, value in after.items():`<br>`print(f"{key}: {value}")`<br><br>`[{'distance': 25, 'duration': 450, 'timestamp': '2024-08-06T12:00:00Z'}, {'distance': 40, 'duration': 600, 'timestamp': '2024-08-06T12:10:00Z'}]`<br>`0: {'distance': 25, 'duration': 450, 'timestamp': '2024-08-06T12:00:00Z'}`<br>`1: {'distance': 40, 'duration': 600, 'timestamp': '2024-08-06T12:10:00Z'}`<br>`-O3bcPZcUb8xtxX_UUO8: {'distance': 30, 'duration': 500, 'timestamp': '2024-08-06T12:10:00Z'}` |

| | |
|---|---|
| | |
| 13 | Question: Firebase Realtime database generates events on data operations. You can refer to section 'Handling Realtime Database events' in the document (https://firebase.google.com/docs/functions/database-events?gen=2nd ). Discuss in the active learning session and summarise the idea of database events and how it is handled using Python SDK.<br><br>Note that these events are useful when your sensors (from Arduino script) store data directly to Firebase Realtime database and you would like to track data update actions from a central Python application such as a monitoring dashboard.<br><br>Answer: Firebase Realtime Database generates events whenever data is created, updated, deleted, or read. These events allow developers to trigger functions in response to specific database operations. Based on these operations, we can have the response to data to make change or notify . For example, On_value_written(), can be called when triggered when data is created, updated, or deleted in Realtime Database, by knowing this we can connect with Python to monitor which variable have been created at that time to handle it. Or knowing why data have been updated or deleted so that we can make a response to that. |

# Activity 5.2: Data wrangling

Data wrangling is the process of converting raw data into a usable form. The process includes collecting, processing, analyzing, and tidying the raw data so that it can be easily read and analyzed. In this activity, you will use the common library in python, "pandas".

## Hardware Required

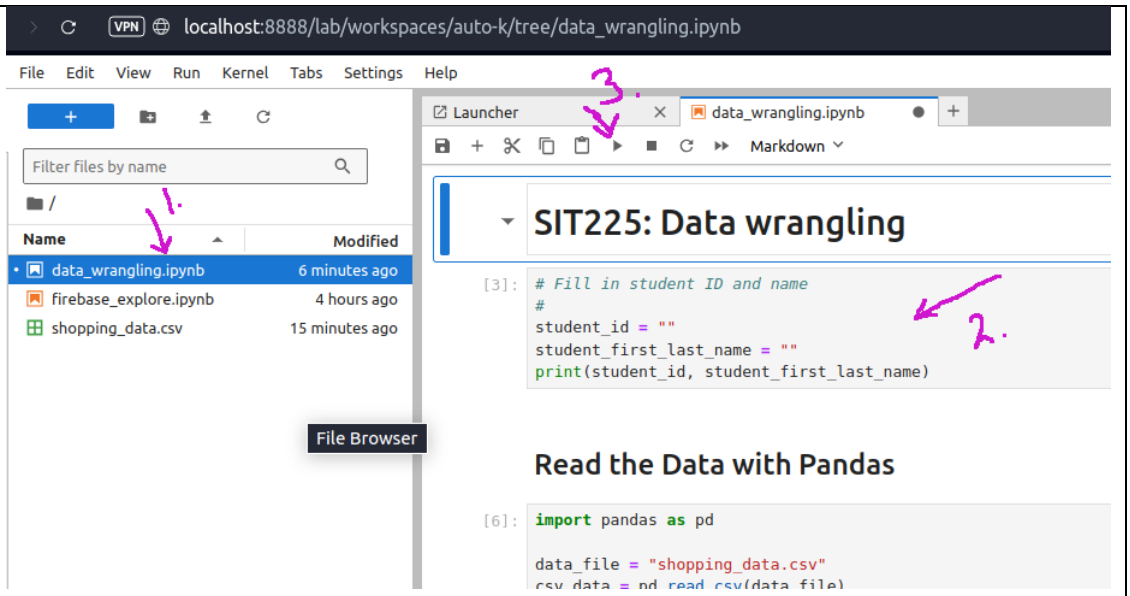
No hardware is required.

## Software Required

Python 3
Pandas Python library

## Steps

| Step | Action |
|---|---|
| 1 | Install Pandas using the command below. Most likely you already have Pandas installed if you have installed Python using Anaconda disribution (https://www.anaconda.com/download).<br><br>$ pip install pandas<br><br>A Python notebook is shared in the GitHub link (https://github.com/deakin-deep-dreamer/sit225/tree/main/week_5 ). There will be a data_wrangling.ipynb, shopping_data.csv and shopping_data_missingvalue.csv files among others. Download the week_5 folder in your computer, open a command prompt in that folder, and write the command below in the command line:<br><br>$ jupyter lab<br><br>This will open Python Jupyter Notebook where in the left panel you can see the files (labeled as 1 in figure). |

| | |
|---|---|
| |  <br>Each cell contains Python code (labeled as 2 in figure), you can run a cell by clicking on the cell, so the cursor appears in that cell and then click on the play button at the top of the panel (labeled as 3 in the figure). |
| 2 | Question: Run each cell to produce output. Follow instructions in the notebook to complete codes in some of the cells. Convert the notebook to PDF from menu File > Save and Export Notebook As > PDF. Convert this activity sheet to PDF and merge with the notebook PDF.<br><br>Answer: There is no answer to write here. You have to answer in the Jupyter Notebook. |
| 3 | Question: Once you went through the cells in the Notebook, you now have a basic understanding of data wrangling. Pandas are a powerful tool and can be used for reading CSV data. Can you use Pandas in reading sensor CSV data that you generated earlier? Describe if any modification you think necessary?<br><br>Answer: Yes we can use pandas to read any csv file. However, we should have some data wrangling steps like convert to datetime type or removing missing data. We can use read_csv() function to read the file. |
| 4 | Question: What do you understand of the Notebook section called Handling Missing Value? Discuss in group and briefly summarise different missing value imputation methods and their applicability on different data conditions.<br><br>Answer: **Time Series Data Without Trend and with Seasonality**: For time series data that exhibit seasonality but no significant trend, methods such as mean, median, mode, or random imputation can be effective. These methods help |

| | |
|---|---|
| | maintain the general pattern and seasonality without overcomplicating the imputation process.<br>**Numerical or Continuous Data**: In cases of numerical or continuous data, common imputation methods include mean, median, mode, multiple imputation, and linear regression. Mean, median, and mode imputation are straightforward and useful for general cases, while multiple imputation provides a robust approach by creating several datasets and results. Linear regression imputation uses relationships between variables to predict missing values, offering solution when relationships are strong.<br>**Categorical Data**: For datasets with categorical variables, multiple imputation can be employed to handle missing values. This approach generates several imputed datasets, accounts for uncertainty, and helps in retaining the categorical nature of the data. |

# Q2

First I have place a hypothesis that there was a slight change in the value of x, y and z as I did not touch the sensor. However the figure that I collect from 2 different sample rate I break this hypothesis:

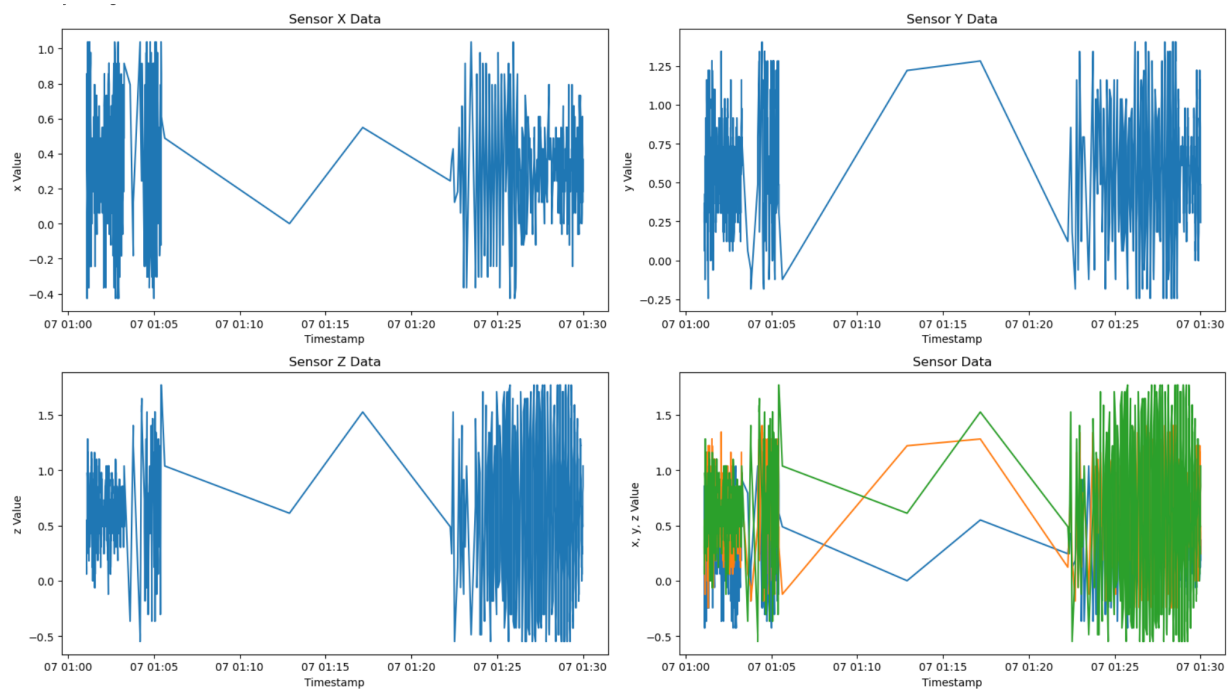



*Figure 1: The graph with sampling rate is 100Hz*

Since 100Hz is the close to the maximum that the sensor can get in Arduino so that I tried to use it. In this case, I have monitored in 30 minutes and

already remove the outliers to see the chart clearly. As we can see the value changed significantly so that the whole charts are like a spectrum. There is a clear line at middle as the value does not change rapidly. And all the value of x, y and z are nearly similar.

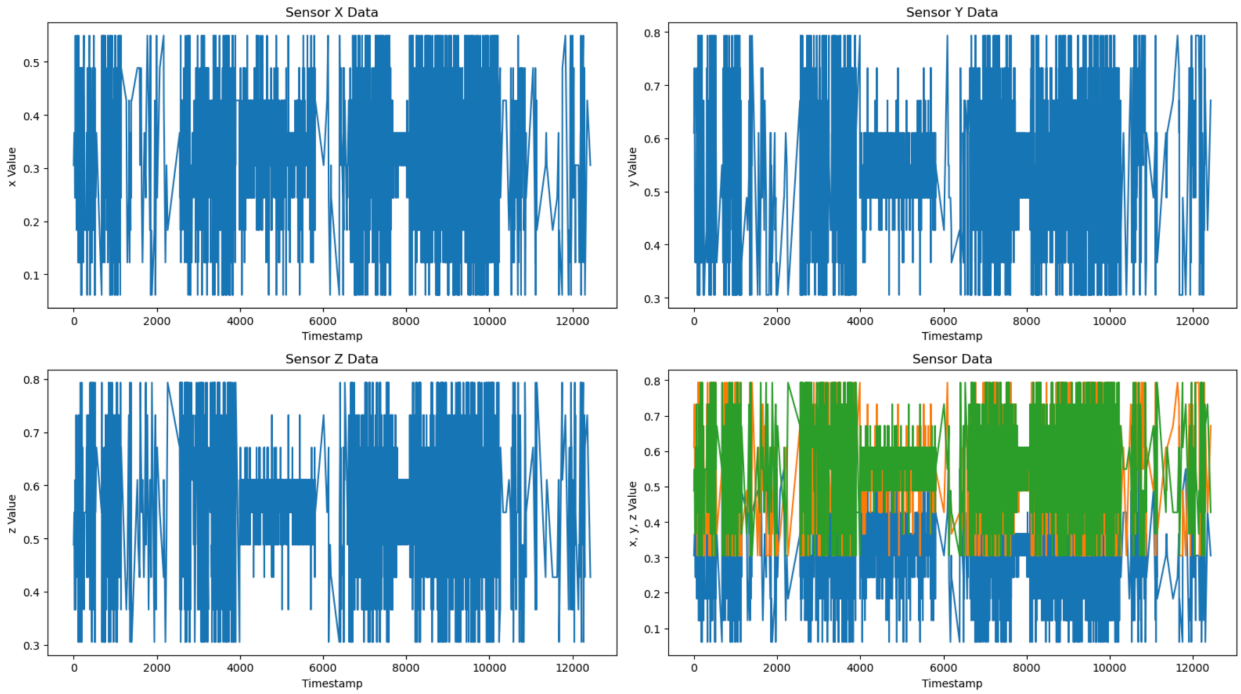



*Figure 2: The graph with sampling 10Hz.*

*I tried to reduce the sample rate to see the value more clearly. However the value still go like a spectrum like this. I think when I held it the value changing rapidly make it result in the graph like this. This can say that reducing sample rate to see the value is not a good idea, since a suitable sample rate might result in suitable graph.*

# Q3

```
#include <Arduino_LSM6DS3.h>

float x, y, z;

void setup() {
  Serial.begin(9600); // set baud rate
  while (!Serial);  // wait for port to init
  Serial.println("Started");
```

```
  if (!IMU.begin()) {
    Serial.println("Failed to initialize IMU!");
    while (1);
  }
}

void loop() {
  // read accelero data
  if (IMU.gyroscopeAvailable()) {
    IMU.readGyroscope(x, y, z);
  }
  String data = "{\"x\":" + String(x, 4) + ",\"y\":" + String(y, 4) + ",\"z\":" +
String(z, 4) + "}";
  Serial.println(data);
  delay(100);
}
```

```
import serial
import firebase_admin
from firebase_admin import credentials, db
import json
from datetime import datetime

databaseURL = 'https://sit225-2024-default-rtdb.asia-southeast1.firebasedatabase.app/'
cred_obj = firebase_admin.credentials.Certificate(
    'sit225-2024-firebase-adminsdk-8yopu-1f7163bd72.json'
)
default_app = firebase_admin.initialize_app(cred_obj, {
    'databaseURL':databaseURL
    })

# Set up serial connection
ser = serial.Serial('/dev/cu.usbmodem2101', 9600, timeout=20)  # Replace 'COM_PORT' with
your Arduino port

def upload_to_firebase(data):
    ref = db.reference('sensor_data_2')
    ref.push(data)
#Delete the previous data
ref = db.reference('sensor_data')

print("Starting data collection...")
while True:
    # Read the data from Arduino
    line = ser.readline().decode('utf-8').strip()
    # Parse the JSON data
    sensor_data = json.loads(line)

    # Record the current timestamp
```

```
    timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')

    # Prepare the data for Firebase
    data = {
        'timestamp': timestamp,
        'x': sensor_data.get('x'),
        'y': sensor_data.get('y'),
        'z': sensor_data.get('z')
    }

    # Upload data to Firebase
    upload_to_firebase(data)
    print(f"Data uploaded: {data}")

#Converting the JSON to CSV
import json
import csv

with open('sit225-2024.json') as json_file:
    datas = json.load(json_file)

with open('data_file.csv', mode='w', newline='') as file:
    writer = csv.DictWriter(file, fieldnames=['id','timestamp', 'x', 'y', 'z'])
    # Write the header
    writer.writeheader()
    for id, items in datas.items():
        for id_1, item in items.items():
            item['id'] = id_1  # Add the ID to the data dictionary
            writer.writerow(item)
data_file.close()

#Read the CSV file
import pandas as pd
df = pd.read_csv('data_file.csv')

#Remove the null value and drop the id columns and set timestamp as index
df.dropna(inplace = True)
df['timestamp'] = pd.to_datetime(df.timestamp)
df = df.drop(columns = 'id')
df.set_index('timestamp',inplace = True)
df.info()

#Remove outliers
def remove_outliers_iqr(df, columns, factor=1.5):
    # Calculate Q1 (25th percentile) and Q3 (75th percentile) for each column
    Q1 = df[columns].quantile(0.25)
    Q3 = df[columns].quantile(0.75)
```

```
    # Calculate IQR (Interquartile Range)
    IQR = Q3 - Q1

    # Define the bounds for outliers
    lower_bound = Q1 - factor * IQR
    upper_bound = Q3 + factor * IQR

    # Identify outliers
    outliers = (df[columns] < lower_bound) | (df[columns] > upper_bound)

    # Filter out outliers
    filtered_df = df[~outliers.any(axis=1)]

    return filtered_df

for _ in range(10):
    clean_df = remove_outliers_iqr(clean_df, columns=['x', 'y', 'z'])
clean_df.info()



import matplotlib.pyplot as plt


df.info()
fig, axs = plt.subplots(2,2, figsize=(16, 9))

#plot there variable in single graph
axs[0,0].plot(clean_df.index, clean_df['x'],linestyle='-')
axs[0,0].set_title('Sensor X Data')
axs[0,0].set_xlabel('Timestamp')
axs[0,0].set_ylabel('x Value')

axs[0,1].plot(clean_df.index, clean_df['y'], label='y')
axs[0,1].set_title('Sensor Y Data')
axs[0,1].set_xlabel('Timestamp')
axs[0,1].set_ylabel('y Value')

axs[1,0].plot(clean_df.index, clean_df['z'], label='z')
axs[1,0].set_title('Sensor Z Data')
axs[1,0].set_ylabel('z Value')
axs[1,0].set_xlabel('Timestamp')

axs[1,1].plot(clean_df.index, clean_df.x)
axs[1,1].plot(clean_df.index, clean_df.y)
axs[1,1].plot(clean_df.index, clean_df.z)
axs[1,1].set_title('Sensor Data')
axs[1,1].set_ylabel('x, y, z Value')
```

```
axs[1,1].set_xlabel('Timestamp')
plt.tight_layout()
plt.show()
```

## Q4

https://deakin.au.panopto.com/Panopto/Pages/Viewer.aspx?id=a840846b-a2c0-4935-a67f-b1c500efe55a

## Q5

https://github.com/namquang2910/SIT225_2024T2/tree/main/5.1P