SIT225 Data Capture Technologies

Pass Task: Store data to cloud

Overview

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.

Hardware Required

i. Arduino Nano 33 IoT device, ii. USB cable, iii. LSM6DS3 module on the Arduino Nano 33 IoT for Gyroscope data.

Software Required

Python 3.

Pre-requisites: You must do the following before this task

Week 5 activities in the unit site.

Task Objective

In this week, you have learned how to use Firebase real-time database and perform operations such as add/update/query data and listen to live changes to your data. In this task, you will need to record Gyroscope readings from the built-in module of Arduino Nano board and pass data to a Python script through Serial communication (or through Arduino IoT Cloud variable synchronisation with Python). The Python script upload data to Firebase real-time database as soon as it receives any. After enough data is stored to capture interesting patterns, for no less than 10 minutes, query the database for data, format it to save in a CSV file and analyse the data. *You may reuse this data next week, so keep your CSV file handy*.

   a. WW

Submission details

Q1. Perform week 5 activities mentioned in the unit site and produce outputs.

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**: 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**: 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**:

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).

$ pip install firebase_admin

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.
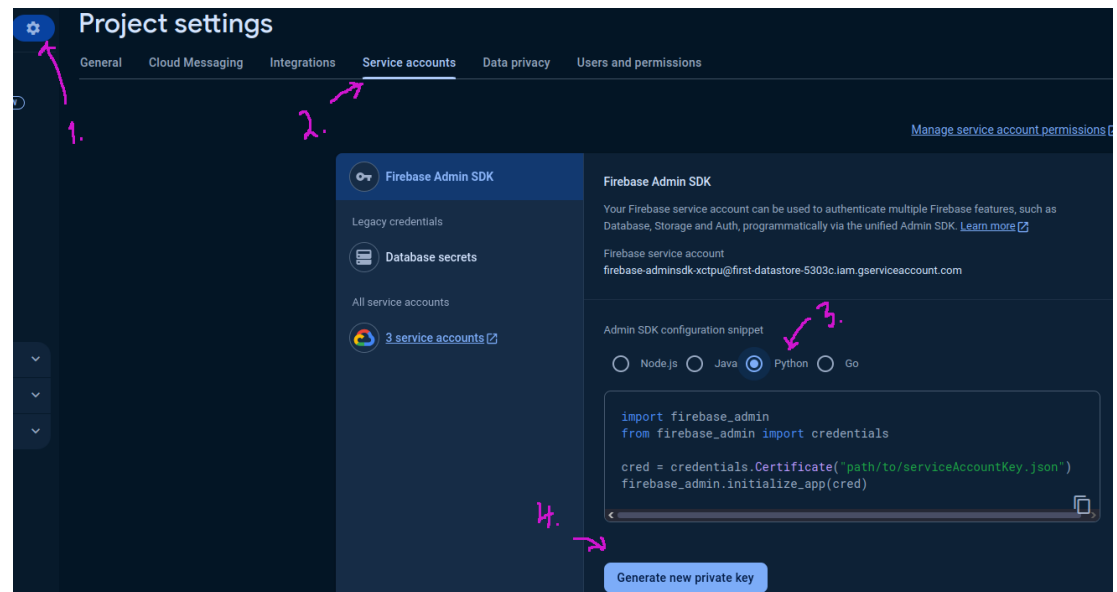

| 4 | **Connect to Firebase using Python version of Admin Database API**:
 A credential object needs to be created to initialise the Python library which can be done using the Python code below. Python notebook can be downloaded here (https://github.com/deakin-deep-dreamer/sit225/blob/main/week_5/firebase_explore.ipynb). |

```
1   import firebase_admin
2
3   databaseURL = 'https://XXX.firebasedatabase.app/'
4   cred_obj = firebase_admin.credentials.Certificate(
5       'first-datastore-5303c-firebase-adminsdk-xctpu-c9902044ac.json'
6   )
7   default_app = firebase_admin.initialize_app(cred_obj, {
8       'databaseURL':databaseURL
9       })
```

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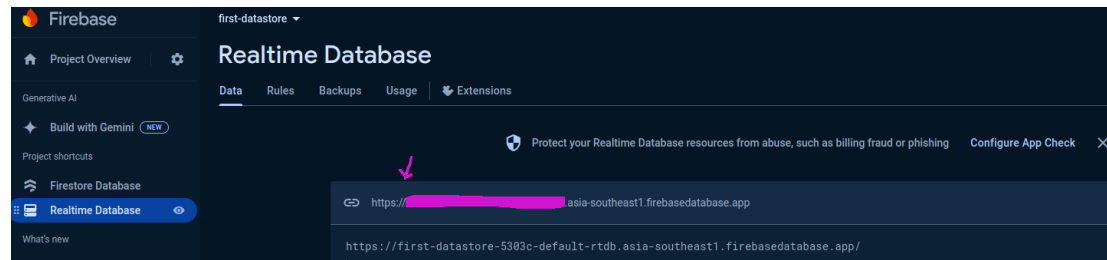
If you compile the code snippet above, it should do with no error.

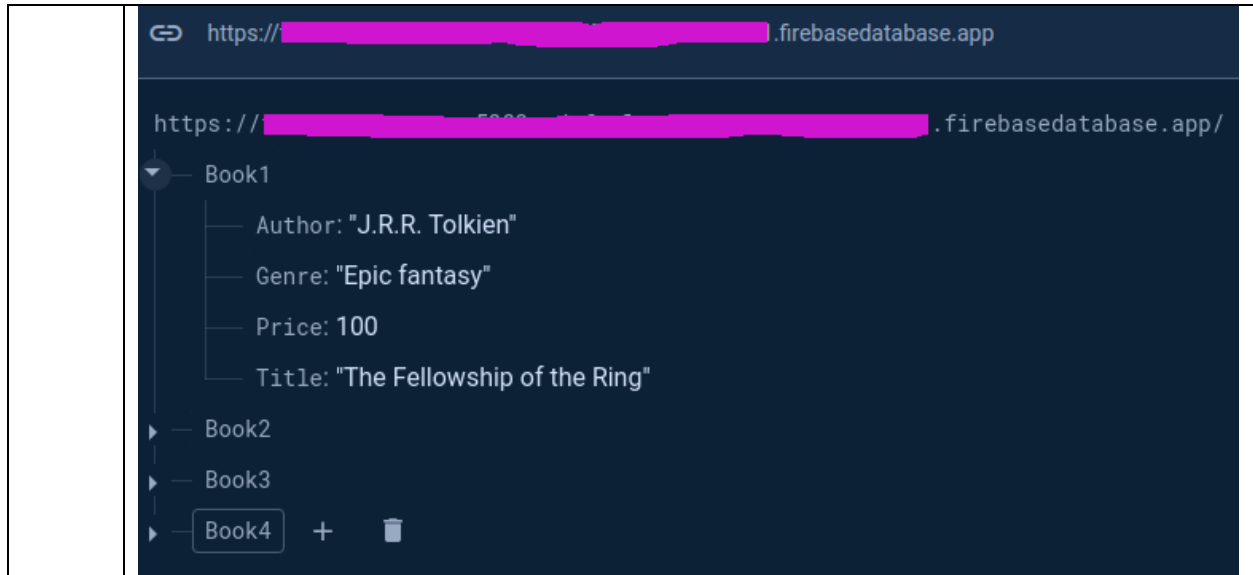| 5	**Write to database Using the set() Function**:
	We set the reference to the root of the database (or we could also set it to a key value or child key value). Data needs to be in JSON format as below.

```python
1   from firebase_admin import db
2
3   # A reference point is always needed to be set
4   # before any operation is carried out on a database.
5   #
6   ref = db.reference("/")
7
8   # JSON format data (key/value pair)
9   data = {   # Outer {} contains inner data structure
10      "Book1":
11      {
12          "Title": "The Fellowship of the Ring",
13          "Author": "J.R.R. Tolkien",
14          "Genre": "Epic fantasy",
15          "Price": 100
16      },
17      "Book2":
18      {
19          "Title": "The Two Towers",
20          "Author": "J.R.R. Tolkien",
21          "Genre": "Epic fantasy",
22          "Price": 100
23      },
24      "Book3":
25      {
26          "Title": "The Return of the King",
27          "Author": "J.R.R. Tolkien",
28          "Genre": "Epic fantasy",
29          "Price": 100
30      },
31      "Book4":
32      {
33          "Title": "Brida",
34          "Author": "Paulo Coelho",
35          "Genre": "Fiction",
36          "Price": 100
37      }
38  }
39
40  # JSON format data is set (overwritten) to the reference
41  # point set at /, which is the root node.
42  #
43  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.

You can visualise the data in the Firebase console as below -

	https:// ▓▓▓▓▓▓▓▓▓▓▓▓▓▓ .firebasedatabase.app/
	▼ Book1
	     Author: "J.R.R. Tolkien"
	     Genre: "Epic fantasy"
	     Price: 100
	     Title: "The Fellowship of the Ring"
	▶ Book2
	▶ Book3
	▶ Book4 + 🗑

6	**Read data using get() function**:
	Data can be read using get() function on the reference set beforehand, as shown below.
	
	``python
	ref = db.reference("/") # set ref point
	
	# query all data under the ref
	books = ref.get()
	print(books)
	print(type(books))
	
	# print each item separately
	for key, value in books.items():
	    print(f"{key}: {value}")
	
	# Query /Book1
	ref = db.reference("/Book1")
	books = ref.get()
	print(books)
	``
	✓ 0.3s
	
	``
	{'Book1': {'Author': 'J.R.R. Tolkien', 'Genre': 'Epic fantasy', 'Price': 100, 'Titl
	<class 'dict'>
	Book1: {'Author': 'J.R.R. Tolkien', 'Genre': 'Epic fantasy', 'Price': 100, 'Title':
	Book2: {'Author': 'J.R.R. Tolkien', 'Genre': 'Epic fantasy', 'Price': 100, 'Title':
	Book3: {'Author': 'J.R.R. Tolkien', 'Genre': 'Epic fantasy', 'Price': 100, 'Title':
	Book4: {'Author': 'Paulo Coelho', 'Genre': 'Fiction', 'Price': 100, 'Title': 'Brida
	{'Author': 'J.R.R. Tolkien', 'Genre': 'Epic fantasy', 'Price': 100, 'Title': 'The F
	``

	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**:
	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.
	
	``python
	1 # Write using push() function
	2 # Note that a set() is called on top of push()
	3 #
	4 ref = db.reference("/")
	5 ref.set({
	6 "Books":
	7 {
	8 "Best_Sellers": -1
	9 }
	10 })
	11
	12 ref = db.reference("/Books/Best_Sellers")
	13
	14 for key, value in data.items():
	15 ref.push().set(value)
	``
	✓ 2.0s
	
	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.

```python
1   # Update data
2   #
3   # Requirement: The price of the books by
4   # J. R. R. Tolkien is reduced to 80 units to
5   # offer a discount.
6   #
7   ref = db.reference("/Books/Best_Sellers/")
8   best_sellers = ref.get()
9   print(best_sellers)
10  for key, value in best_sellers.items():
11      if(value["Author"] == "J.R.R. Tolkien"):
12          value["Price"] = 90
13          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.

```
▼ — Best_Sellers
    ▼ — -O-iqpiYlui92UKRmctM
        — Author: "J.R.R. Tolkien"
        — Genre: "Epic fantasy"
        — Price: 80
        — Title: "The Fellowship of the Ring"
    ▼ — -O-iqpnK8M8wjLiw2PTX
        — Author: "J.R.R. Tolkien"
        — Genre: "Epic fantasy"
        — Price: 80
        — Title: "The Two Towers"
    ▼ — -O-iqptGIKG7WuxHdGsq
        — Author: "J.R.R. Tolkien"
        — Genre: "Epic fantasy"
        — Price: 80
        — Title: "The Return of the King"
    ▼ — -O-iqpz_nsDjhwMzLmIw
        — Author: "Paulo Coelho"
```

9	**Delete data**: 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.

```
1   # Let's delete all best seller books
2   # with J.R.R. Tolkien as the author.
3   #
4   ref = db.reference("/Books/Best_Sellers")
5
6   for key, value in best_sellers.items():
7       if(value["Author"] == "J.R.R. Tolkien"):
8           ref.child(key).set({})
```

This keeps only the other author data, as shown below.

```
▼ — Books
    ▼ — Best_Sellers
        ▼ — -O-iqpz_nsDjhwMzLmIw
                ── Author: "Paulo Coelho"
                ── Genre: "Fiction"
                ── Price: 100
                ── Title: "Brida"
```

If ref.child() not used, as shown the code below, all data will be removed.

```
1   ref = db.reference("/Books/Best_Sellers")
2   ref.set({})
```

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. 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.

Answer: **I will organize the data so that every sensor is saved under sensors/{sensor_id}, including a metadata section (sensor_name, type, location, units) and a readings section that contains all recorded values.**

Each reading will feature a shared timestamp (Unix ms) along with only the variables pertinent to that sensor — for instance, temperature and humidity for DHT22, or x, y, and z values for an accelerometer.

To facilitate continuous logging, new entries will be appended under readings using Firebase push IDs, which prevents key collisions and naturally arranges the data chronologically.

This structure reduces redundant metadata, accommodates various sensor types, simplifies time-based queries (latest values or by specific range), and guarantees efficient storage and retrieval in Firebase.

11	Question: Generate some random data for DHT22 sensor, insert data to database, query all data and screenshot the output here.
	Answer:

12	Question: Generate some random data for the SR04 Ultrasonic sensor, insert data to database, query all data and screenshot the output here.
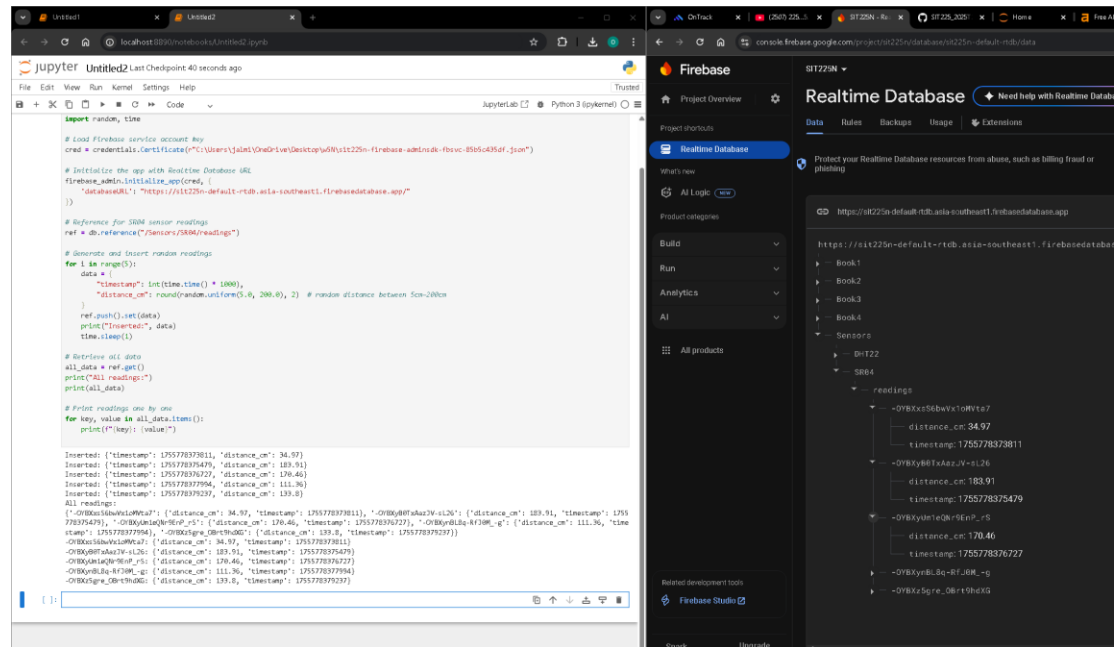
	Answer:
13	**Question**: Firebase Realtime database generates events on data operations. You can refer to section 'Handling Realtime Database events' in the document (https://firebase.google.com/docs/functions/database-events?gen=2nd). Discuss in the active learning session and summarise the idea of database events and how it is handled using Python SDK. 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. **Answer**: The Firebase Realtime Database activates events whenever data is created, updated, deleted, or written. By utilizing Cloud Functions (Gen 2), you can link triggers like onValueCreated, onValueUpdated, onValueDeleted, and onValueWritten to a specific path (for instance, /sensors/{sensorId}/readings/{pushId}), and the function will run with each modification. The handler will obtain a snapshot for create/delete events or before/after snapshots for write/update events, enabling you to validate, compute aggregates, or send alerts without altering client code. You can use path wildcards (such as {sensorId}) to handle multiple sensors, and it's wise to deploy the function in the same region as the database to reduce latency.

	In Python, the Admin SDK allows secure read/write operations but does not support realtime listeners. A monitoring application can either (1) use REST streaming (SSE) to follow a path in near-realtime, or (2) periodically check for recent readings. A suggested method is to let Cloud Functions normalize updates (for example, writing to /latest/{sensorId} or /events) and have the Python dashboard keep an eye on that single path through SSE or light polling.

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
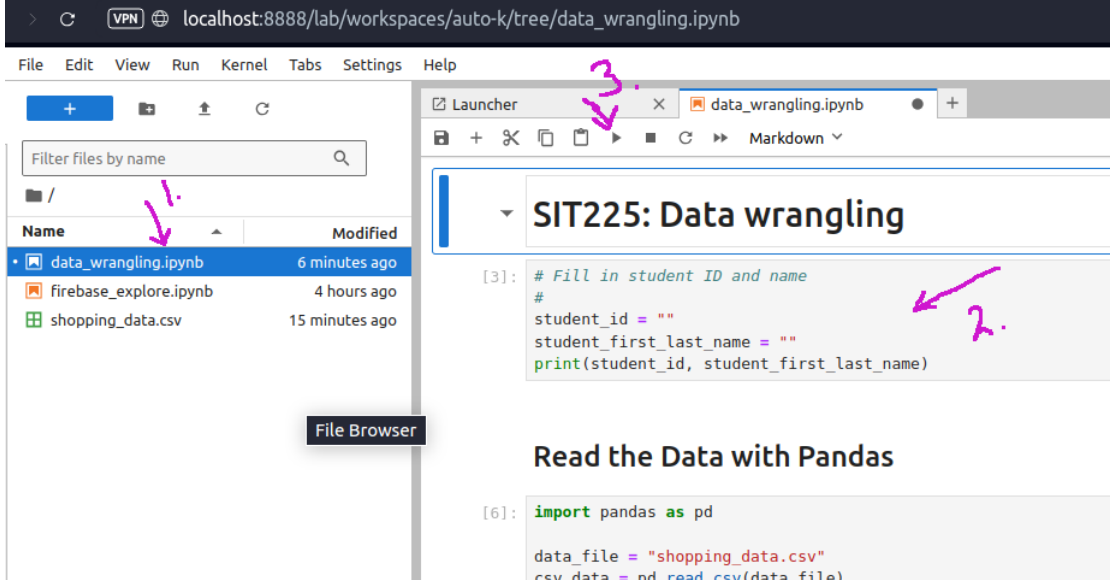
No hardware is required.

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).
	
	    $ pip install pandas
	
	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:
	
	    $ jupyter lab
	
	This will open Python Jupyter Notebook where in the left panel you can see the files (labeled as 1 in figure).
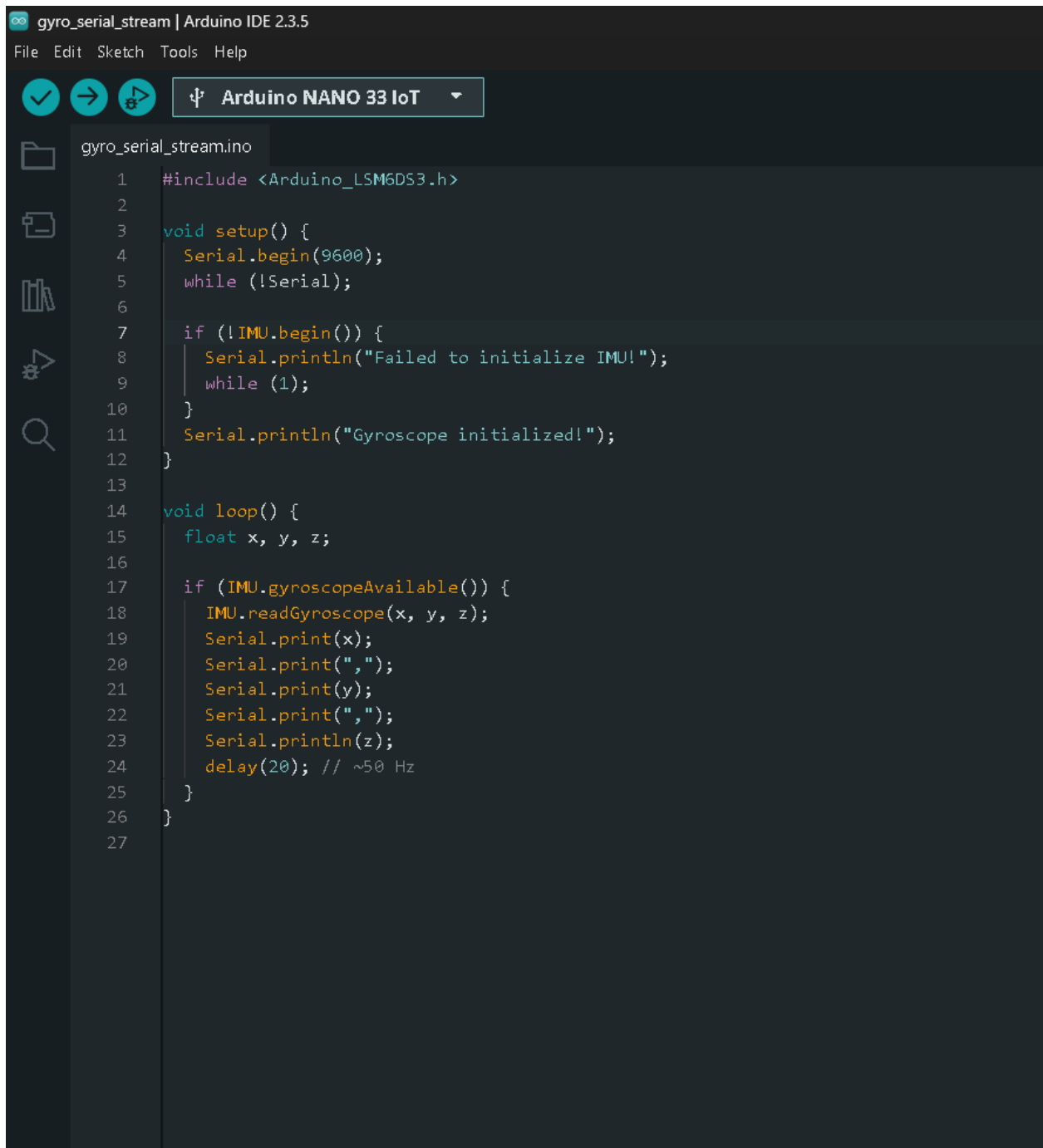
Each cell contains Python code (labeled as 2 in figure), you can run a cell by clicking on the cell, so the cursor appears in that cell and then click on the play button at the top of the panel (labeled as 3 in the figure).

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. 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? Answer: Yes, Pandas can easily handle sensor CSV files with read_csv(). The main tweaks I would make are cleaning and preparing the data: converting timestamps into proper datetime format and using them as the index, standardizing column names and units (e.g., temperature, humidity, distance_cm), handling missing values with fill or interpolation, dropping duplicates, and filtering unrealistic readings such as humidity over 100% or negative distance values.

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. **Answer**: I checked out the "Handling Missing Value" section and took it to mean "first spot the NaNs, then pick a fill that fits the data." If there are just a few missing rows and they seem random, I'd just get rid of them; for numeric columns, I'd use the mean or median (going with the median if the data is skewed), and for categorical data, I'd go for the mode. When it comes to sensor time-series data, I'd use forward or backward filling or time-based interpolation for short gaps. Plus, if the features are linked, a simple model like KNN could work well. For example, with DHT22 temperature, I'd fill in a short gap with interpolation, for a skewed variable like Income, I'd pick the median, and for Gender, I'd choose the mode—then I'd make sure to note down the changes for clarity.

Q2. State the hypothesis you can think of out of your data. Show the graph created from the sensor data, analyse it and describe if there are any interesting patterns you can observe. Justify if your hypothesis holds, at what level; if not, then what might be the reason?

My hypothesis was that the gyroscope readings on the X, Y, and Z axes would stay close to zero when the Arduino board was stationary, but would show clear spikes whenever the board was rotated. From the graphs I collected, the results supported this idea because the values stayed stable with only small fluctuations when the board was not moving, while sharp peaks appeared in the axis that matched the direction of rotation. I also noticed that sometimes the other axes showed minor changes, which could be due to the way the sensor is aligned. Overall, the hypothesis holds true at a general level because the gyroscope correctly detected both stillness and movement, but the data was not perfectly stable. This is most likely caused by normal sensor noise, drift over time, or small calibration errors.

Q3. Paste Python and Arduino sketch and explain program steps.
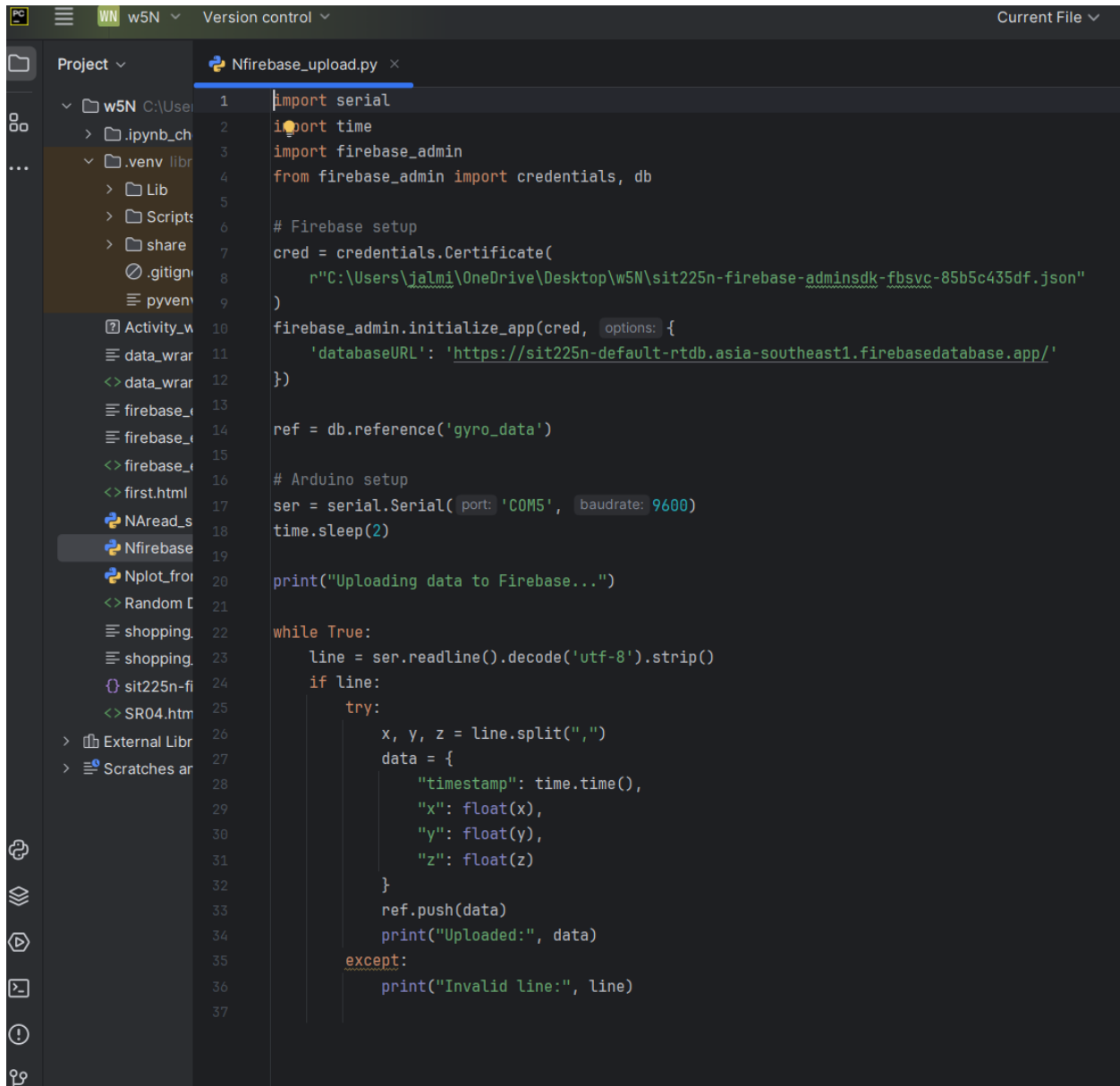
```
gyro_serial_stream | Arduino IDE 2.3.5
File  Edit  Sketch  Tools  Help

            Arduino NANO 33 IoT

gyro_serial_stream.ino
    1    #include <Arduino_LSM6DS3.h>
    2
    3    void setup() {
    4      Serial.begin(9600);
    5      while (!Serial);
    6
    7      if (!IMU.begin()) {
    8        Serial.println("Failed to initialize IMU!");
    9        while (1);
   10      }
   11      Serial.println("Gyroscope initialized!");
   12    }
   13
   14    void loop() {
   15      float x, y, z;
   16
   17      if (IMU.gyroscopeAvailable()) {
   18        IMU.readGyroscope(x, y, z);
   19        Serial.print(x);
   20        Serial.print(",");
   21        Serial.print(y);
   22        Serial.print(",");
   23        Serial.println(z);
   24        delay(20); // ~50 Hz
   25      }
   26    }
   27
```

The Arduino sketch starts by including the Arduino_LSM6DS3 library and initializing the gyroscope sensor inside the setup() function. If the sensor is successfully detected, it prints a confirmation message; otherwise, it displays an error. In the loop(), the Arduino continuously checks if

gyroscope data is available, reads the values for the X, Y, and Z axes, and then sends them over the serial port in a comma-separated format with a small delay to maintain a stable reading rate.

On the Python side, the script uses the serial library to connect to the Arduino through the correct COM port at 9600 baud rate. It then continuously reads the incoming serial data, splits each line into X, Y, and Z values, and stores them in lists along with timestamps. The program uses matplotlib to update live plots, showing the changes in each axis in real time. This way, the

Arduino is responsible for capturing and transmitting the raw sensor data, while the Python program handles storing and visualizing it.

Q4. Create a video in Panopto/CloudDeakin showing your program execution, data collection, data upload to Firebase and graph output, share the video link here.

Q5. Create a subdirectory 'week-5' under directory 'SIT225_<YYYY>T2' in your drive where you copy the Python script file, Arduino sketch file, data file and the generated graphs. Commit and push changes to GitHub. Include the link to your repository here with a GitHub page screenshot of weekly folder content. A tutor may try to access your GitHub link, if necessary. Give access to your tutor by adding the tutor's email address as a collaborator of your private repository.

Instructions

Consolidate outputs following the submission details above into a single PDF file.

Submit your work

When you are ready, login to OnTrack and submit your pdf which consolidates all the items mentioned in the submission detail section above. Remember to save and backup your work.

Complete your work

After your submission, your OnTrack reviewer (tutor) will review your submission and give you feedback in about 5 business days. Your reviewer may further ask you some questions on the weekly topics and/or about your submissions. You are required to address your OnTrack reviewer's questions as a form of task discussion. Please frequently login to OnTrack for the task *Discuss/Demonstrate* or *Resubmit* equivalent to fix your work (if needed) based on the feedback to get your task signed as *Complete*.

SIT225: Data wrangling

Run each cell to generate output and finally convert this notebook to PDF.

```
In [1]:   # Fill in student ID and name
          #
          student_id = "223737376"
          student_first_last_name = "Nawal"
          print(student_id, student_first_last_name)
```

```
223737376 Nawal
```

Read the Data with Pandas

Pandas has a dedicated function read_csv() to read CSV files.

Just in case we have a large number of data, we can just show into only five rows with head function. It will show you 5 rows data automatically.

```
In [3]:   import pandas as pd

          data_file = "shopping_data (1).csv"
          csv_data = pd.read_csv(data_file)

          print(csv_data)

          # show into only five rows with head function
          print(csv_data.head())
```

```
     CustomerID  Genre  Age  Annual Income (k$)  Spending Score (1-100)
0             1    Male   19                  15                      39
1             2    Male   21                  15                      81
2             3  Female   20                  16                       6
3             4  Female   23                  16                      77
4             5  Female   31                  17                      40
..          ...     ...  ...                 ...                     ...
195         196  Female   35                 120                      79
196         197  Female   45                 126                      28
197         198    Male   32                 126                      74
198         199    Male   32                 137                      18
199         200    Male   30                 137                      83

[200 rows x 5 columns]
   CustomerID  Genre  Age  Annual Income (k$)  Spending Score (1-100)
0           1    Male   19                  15                      39
1           2    Male   21                  15                      81
2           3  Female   20                  16                       6
3           4  Female   23                  16                      77
4           5  Female   31                  17                      40
```

Access the Column

Pandas has provided function .columns to access the column of the data source.

```
In [4]:  print(csv_data.columns)

         # if we want to access just one column, for example "Age"
         print("Age:")
         print(csv_data["Age"])
```

```
Index(['CustomerID', 'Genre', 'Age', 'Annual Income (k$)',
       'Spending Score (1-100)'],
      dtype='object')
Age:
0       19
1       21
2       20
3       23
4       31
        ..
195     35
196     45
197     32
198     32
199     30
Name: Age, Length: 200, dtype: int64
```

Access the Row

In addition to accessing data through columns, using pandas can also access using rows.
In contrast to access through columns, the function to display data from a row is the
.iloc[i] function where [i] indicates the order of the rows to be displayed where the index
starts from 0.

```
In [5]:  # we want to know what line 5 contains

         print(csv_data.iloc[5])

         print()

         # We can combine both of those function to show row and column we want.
         # For the example, we want to show the value in column "Age" at the first row
         # (remember that the row starts at 0)
         #
         print(csv_data["Age"].iloc[1])
```

```
CustomerID                6
Genre                Female
Age                      22
Annual Income (k$)       17
Spending Score (1-100)   76
Name: 5, dtype: object

21
```

Show Data Based on Range

After displaying a data set, what if you want to display data from rows 5 to 20 of a
dataset? To anticipate this, pandas can also display data within a certain range, both
ranges for rows only, only columns, and ranges for rows and columns

```
In [6]: print("Shows data to 5th to less than 10th in a row:")
        print(csv_data.iloc[5:10])
```

```
Shows data to 5th to less than 10th in a row:
   CustomerID   Genre  Age  Annual Income (k$)  Spending Score (1-100)
5           6  Female   22                  17                      76
6           7  Female   35                  18                       6
7           8  Female   23                  18                      94
8           9    Male   64                  19                       3
9          10  Female   30                  19                      72
```

Using Numpy to Show the Statistic Information

The describe() function allows to quickly find statistical information from a dataset. Those
information such as mean, median, modus, max min, even standard deviation. Don't
forget to install Numpy before using describe function.

```
In [7]: print(csv_data.describe(include="all"))
```

```
        CustomerID  Genre         Age  Annual Income (k$)  \
count   200.000000    200  200.000000          200.000000
unique         NaN      2         NaN                 NaN
top            NaN  Female         NaN                 NaN
freq           NaN    112         NaN                 NaN
mean    100.500000    NaN   38.850000           60.560000
std      57.879185    NaN   13.969007           26.264721
min       1.000000    NaN   18.000000           15.000000
25%      50.750000    NaN   28.750000           41.500000
50%     100.500000    NaN   36.000000           61.500000
75%     150.250000    NaN   49.000000           78.000000
max     200.000000    NaN   70.000000          137.000000

        Spending Score (1-100)
count               200.000000
unique                     NaN
top                        NaN
freq                       NaN
mean                 50.200000
std                  25.823522
min                   1.000000
25%                  34.750000
50%                  50.000000
75%                  73.000000
max                  99.000000
```

Handling Missing Value

In [8]:
```python
# For the first step, we will figure out if there is missing value.
print(csv_data.isnull().values.any())
print()
```

False

In [10]:
```python
# We will use another data source with missing values to practice this part.
data_missing = pd.read_csv("shopping_data_missingvalue (1).csv")
print(data_missing.head())

print()

print("Missing? ", data_missing.isnull().values.any())
```

```
   CustomerID   Genre   Age  Annual Income (k$)  Spending Score (1-100)
0           1    Male  19.0                15.0                    39.0
1           2    Male   NaN                15.0                    81.0
2           3  Female  20.0                 NaN                     6.0
3           4  Female  23.0                16.0                    77.0
4           5  Female  31.0                17.0                     NaN

Missing?  True
```

In [ ]:

Ways to deal with missing values.

Follow the tutorial (https://deepnote.com/app/rickyharyanto14-3390/Data-Wrangling-w-Python-e5d1a23e-33cf-416d-ad27-4c3f7f467442). It includes -

1. Delete data
   - deleting rows
   - pairwise deletion
   - delete column
2. imputation
   - time series problem
     - Data without trend with seasonality (mean, median, mode, random)
     - Data with trend and without seasonality (linear interpolation)
   - general problem
     - Data categorical (Make NA as multiple imputation)
     - Data numerical or continuous (mean, median, mode, multiple imputation and linear regression)

Filling with Mean Values

The mean is used for data that has a few outliers/noise/anomalies in the distribution of the data and its contents. This value will later fill in the empty value of the dataset that has a missing value case. To fill in an empty value use the fillna() function

In [11]:
```python
print(data_missing.mean())

"""
```

```
Question: This code will generate error. Can you explain why and how it can be sol
Move on to the next cell to find one way it can be solved.

Answer: The error happens because .iloc was used on a Series instead of the DataF

"""
```

```
--------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[11], line 1
----> 1 print(data_missing.mean())
      3 """
      4
      5 Question: This code will generate error. Can you explain why and how it can
be solved?
   (...)      9
     10 """

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.13_qbz5n2kfra8p0\Lo
calCache\local-packages\Python313\site-packages\pandas\core\frame.py:11700, in Data
Frame.mean(self, axis, skipna, numeric_only, **kwargs)
  11692 @doc(make_doc("mean", ndim=2))
  11693 def mean(
  11694     self,
   (...)  11698     **kwargs,
  11699 ):
> 11700     result = super().mean(axis, skipna, numeric_only, **kwargs)
  11701     if isinstance(result, Series):
  11702         result = result.__finalize__(self, method="mean")

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.13_qbz5n2kfra8p0\Lo
calCache\local-packages\Python313\site-packages\pandas\core\generic.py:12439, in ND
Frame.mean(self, axis, skipna, numeric_only, **kwargs)
  12432 def mean(
  12433     self,
  12434     axis: Axis | None = 0,
   (...)  12437     **kwargs,
  12438 ) -> Series | float:
> 12439     return self._stat_function(
  12440              , nanops.nanmean, axis, skipna, numeric_only, **kwargs
  12441     )

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.13_qbz5n2kfra8p0\Lo
calCache\local-packages\Python313\site-packages\pandas\core\generic.py:12396, in ND
Frame._stat_function(self, name, func, axis, skipna, numeric_only, **kwargs)
  12392 nv.validate_func(name, (), kwargs)
  12394 validate_bool_kwarg(skipna, "skipna", none_allowed=False)
> 12396 return self._reduce(
  12397     func, name=name, axis=axis, skipna=skipna, numeric_only=numeric_only
  12398 )

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.13_qbz5n2kfra8p0\Lo
calCache\local-packages\Python313\site-packages\pandas\core\frame.py:11569, in Data
Frame._reduce(self, op, name, axis, skipna, numeric_only, filter_type, **kwds)
  11565     df = df.T
  11567 # After possibly _get_data and transposing, we are now in the
  11568 #  simple case where we can use BlockManager.reduce
> 11569 res = df._mgr.reduce(blk_func)
  11570 out = df._constructor_from_mgr(res, axes=res.axes).iloc[0]
  11571 if out_dtype is not None and out.dtype != "boolean":

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.13_qbz5n2kfra8p0\Lo
calCache\local-packages\Python313\site-packages\pandas\core\internals\managers.py:1
500, in BlockManager.reduce(self, func)
  1498 res_blocks: list[Block] = []
  1499 for blk in self.blocks:
-> 1500     nbs = blk.reduce(func)
```

```
   1501      res_blocks.extend(nbs)
   1503 index = Index([None])  # placeholder

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.13_qbz5n2kfra8p0\Lo
calCache\local-packages\Python313\site-packages\pandas\core\internals\blocks.py:40
6, in Block.reduce(self, func)
    400 @final
    401 def reduce(self, func) -> list[Block]:
    402     # We will apply the function and reshape the result into a single-row
    403     #  Block with the same mgr_locs; squeezing will be done at a higher lev
el
    404     assert self.ndim == 2
--> 406     result = func(self.values)
    408     if self.values.ndim == 1:
    409         res_values = result

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.13_qbz5n2kfra8p0\Lo
calCache\local-packages\Python313\site-packages\pandas\core\frame.py:11488, in Data
Frame._reduce.<locals>.blk_func(values, axis)
   11486         return np.array([result])
   11487 else:
> 11488         return op(values, axis=axis, skipna=skipna, **kwds)

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.13_qbz5n2kfra8p0\Lo
calCache\local-packages\Python313\site-packages\pandas\core\nanops.py:147, in bottl
eneck_switch.__call__.<locals>.f(values, axis, skipna, **kwds)
    145         result = alt(values, axis=axis, skipna=skipna, **kwds)
    146 else:
--> 147     result = alt(values, axis=axis, skipna=skipna, **kwds)
    149 return result

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.13_qbz5n2kfra8p0\Lo
calCache\local-packages\Python313\site-packages\pandas\core\nanops.py:404, in _date
timelike_compat.<locals>.new_func(values, axis, skipna, mask, **kwargs)
    401 if datetimelike and mask is None:
    402     mask = isna(values)
--> 404 result = func(values, axis=axis, skipna=skipna, mask=mask, **kwargs)
    406 if datetimelike:
    407     result = _wrap_results(result, orig_values.dtype, fill_value=iNaT)

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.13_qbz5n2kfra8p0\Lo
calCache\local-packages\Python313\site-packages\pandas\core\nanops.py:720, in nanme
an(values, axis, skipna, mask)
    718 count = _get_counts(values.shape, mask, axis, dtype=dtype_count)
    719 the_sum = values.sum(axis, dtype=dtype_sum)
--> 720 the_sum = _ensure_numeric(the_sum)
    722 if axis is not None and getattr(the_sum, "ndim", False):
    723     count = cast(np.ndarray, count)

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.13_qbz5n2kfra8p0\Lo
calCache\local-packages\Python313\site-packages\pandas\core\nanops.py:1686, in _ens
ure_numeric(x)
   1683 inferred = lib.infer_dtype(x)
   1684 if inferred in ["string", "mixed"]:
   1685     # GH#44008, GH#36703 avoid casting e.g. strings to numeric
-> 1686     raise TypeError(f"Could not convert {x} to numeric")
   1687 try:
   1688     x = x.astype(np.complex128)

TypeError: Could not convert ['MaleMaleFemaleFemaleFemaleFemaleFemaleFemaleMaleFema
```

```
In [12]:  # Genre column contains string values and numerial operation mean fails.
          # Lets drop Genre column since for numerial calculation.
          #
          data_missing_wo_genre = data_missing.drop(columns=['Genre'])
          print(data_missing_wo_genre.head())
```

```
   CustomerID   Age  Annual Income (k$)  Spending Score (1-100)
0           1  19.0                15.0                    39.0
1           2   NaN                15.0                    81.0
2           3  20.0                 NaN                     6.0
3           4  23.0                16.0                    77.0
4           5  31.0                17.0                     NaN
```

```
In [13]:  print(data_missing_wo_genre.mean())
```

```
CustomerID              100.500000
Age                      38.939698
Annual Income (k$)       61.005051
Spending Score (1-100)   50.489899
dtype: float64
```

```
In [14]:  print("Dataset with empty values! :")
          print(data_missing_wo_genre.head(10))

          data_filling=data_missing_wo_genre.fillna(data_missing_wo_genre.mean())
          print("Dataset that has been processed Handling Missing Values with Mean :")
          print(data_filling.head(10))

          # Observe the missing value imputation in corresponding rows.
          #
```

```
Dataset with empty values! :
   CustomerID   Age   Annual Income (k$)   Spending Score (1-100)
0            1  19.0                 15.0                     39.0
1            2   NaN                 15.0                     81.0
2            3  20.0                  NaN                      6.0
3            4  23.0                 16.0                     77.0
4            5  31.0                 17.0                      NaN
5            6  22.0                  NaN                     76.0
6            7  35.0                 18.0                      6.0
7            8  23.0                 18.0                     94.0
8            9  64.0                 19.0                      NaN
9           10  30.0                 19.0                     72.0
Dataset that has been processed Handling Missing Values with Mean :
   CustomerID         Age   Annual Income (k$)   Spending Score (1-100)
0            1  19.000000            15.000000                39.000000
1            2  38.939698            15.000000                81.000000
2            3  20.000000            61.005051                 6.000000
3            4  23.000000            16.000000                77.000000
4            5  31.000000            17.000000                50.489899
5            6  22.000000            61.005051                76.000000
6            7  35.000000            18.000000                 6.000000
7            8  23.000000            18.000000                94.000000
8            9  64.000000            19.000000                50.489899
9           10  30.000000            19.000000                72.000000
```

Filling with Median

The median is used when the data presented has a high outlier. The median was chosen because it is the middle value, which means it is not the result of calculations involving outlier data. In some cases, outlier data is considered disturbing and often considered noisy because it can affect class distribution and interfere with clustering analysis.

```
In [15]:  print(data_missing_wo_genre.median())
          print("Dataset with empty values! :")
          print(data_missing_wo_genre.head(10))

          data_filling2=data_missing_wo_genre.fillna(data_missing_wo_genre.median())
          print("Dataset that has been processed Handling Missing Values with Median :")
          print(data_filling2.head(10))

          # Observe the missing value imputation in corresponding rows.
          #
```

```
CustomerID              100.5
Age                      36.0
Annual Income (k$)       62.0
Spending Score (1-100)   50.0
dtype: float64
```

Dataset with empty values! :

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
0	1	19.0	15.0	39.0
1	2	NaN	15.0	81.0
2	3	20.0	NaN	6.0
3	4	23.0	16.0	77.0
4	5	31.0	17.0	NaN
5	6	22.0	NaN	76.0
6	7	35.0	18.0	6.0
7	8	23.0	18.0	94.0
8	9	64.0	19.0	NaN
9	10	30.0	19.0	72.0

Dataset that has been processed Handling Missing Values with Median :

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
0	1	19.0	15.0	39.0
1	2	36.0	15.0	81.0
2	3	20.0	62.0	6.0
3	4	23.0	16.0	77.0
4	5	31.0	17.0	50.0
5	6	22.0	62.0	76.0
6	7	35.0	18.0	6.0
7	8	23.0	18.0	94.0
8	9	64.0	19.0	50.0
9	10	30.0	19.0	72.0

In [ ]:

In [1]:
```python
# Fill in student ID and name
#
student_id = "223737376"
student_first_last_name = "Nawal"
print(student_id, student_first_last_name)
```

223737376 Nawal

In [7]:
```python
import firebase_admin
from firebase_admin import credentials, db

databaseURL = "https://sit225n-default-rtdb.asia-southeast1.firebasedatabase.app/'

cred_obj = credentials.Certificate(
    r"C:\Users\jalmi\OneDrive\Desktop\w5N\sit225n-firebase-adminsdk-fbsvc-85b5c43!
)

default_app = firebase_admin.initialize_app(cred_obj, {
    'databaseURL': databaseURL
})

ref = db.reference("/")
print(ref.get())
```

{'Book1': {'Author': 'J.R.R. Tolkien', 'Genre': 'Epic fantasy', 'Price': 100, 'Titl
e': 'The Fellowship of the Ring'}, 'Book2': {'Author': 'J.R.R. Tolkien', 'Genre':
'Epic fantasy', 'Price': 100, 'Title': 'The Two Towers'}, 'Book3': {'Author': 'J.R.
R. Tolkien', 'Genre': 'Epic fantasy', 'Price': 100, 'Title': 'The Return of the Kin
g'}, 'Book4': {'Author': 'Paulo Coelho', 'Genre': 'Fiction', 'Price': 100, 'Title':
'Brida'}, 'Sensors': {'DHT22': {'readings': {'-OYBXF-VXr87NtXWT1ey': {'humidity': 5
1.13, 'temperature': 32.32, 'timestamp': 1755778185901}, '-OYBXFJ1MUd9OfNmX9iQ':
{'humidity': 53.59, 'temperature': 27.82, 'timestamp': 1755778187577}, '-OYBXFbiBYn
e7aUkX5xw': {'humidity': 54.19, 'temperature': 28.11, 'timestamp': 1755778188829},
'-OYBXFvFaUw6ismMB8ay': {'humidity': 64.13, 'temperature': 32.18, 'timestamp': 1755
778190082}, '-OYBXGDwIP7GRSv_9Ukd': {'humidity': 57.65, 'temperature': 25.97, 'time
stamp': 1755778191340}}}, 'SR04': {'readings': {'-OYBXxsS6bwVx1oMVta7': {'distance_
cm': 34.97, 'timestamp': 1755778373811}, '-OYBXyB0TxAazJV-sL26': {'distance_cm': 18
3.91, 'timestamp': 1755778375479}, '-OYBXyUm1eQNr9EnP_rS': {'distance_cm': 170.46,
'timestamp': 1755778376727}, '-OYBXynBL8q-RfJ0M_-g': {'distance_cm': 111.36, 'times
tamp': 1755778377994}, '-OYBXz5gre_OBrt9hdXG': {'distance_cm': 133.8, 'timestamp':
1755778379237}}}}}

In [13]:
```python
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)
```

In [11]:
```python
ref = db.reference("/")  # set ref point

# query all data under the ref
books = ref.get()
print(books)
print(type(books))

# print each item separately
for key, value in books.items():
    print(f"{key}: {value}")


# Query /Book1
ref = db.reference("/Book1")
books = ref.get()
print(books)
```

```
{'Book1': {'Author': 'J.R.R. Tolkien', 'Genre': 'Epic fantasy', 'Price': 100, 'Titl
e': 'The Fellowship of the Ring'}, 'Book2': {'Author': 'J.R.R. Tolkien', 'Genre':
'Epic fantasy', 'Price': 100, 'Title': 'The Two Towers'}, 'Book3': {'Author': 'J.R.
R. Tolkien', 'Genre': 'Epic fantasy', 'Price': 100, 'Title': 'The Return of the Kin
g'}, 'Book4': {'Author': 'Paulo Coelho', 'Genre': 'Fiction', 'Price': 100, 'Title':
'Brida'}}
<class 'dict'>
Book1: {'Author': 'J.R.R. Tolkien', 'Genre': 'Epic fantasy', 'Price': 100, 'Title':
'The Fellowship of the Ring'}
Book2: {'Author': 'J.R.R. Tolkien', 'Genre': 'Epic fantasy', 'Price': 100, 'Title':
'The Two Towers'}
Book3: {'Author': 'J.R.R. Tolkien', 'Genre': 'Epic fantasy', 'Price': 100, 'Title':
'The Return of the King'}
Book4: {'Author': 'Paulo Coelho', 'Genre': 'Fiction', 'Price': 100, 'Title': 'Brid
a'}
{'Author': 'J.R.R. Tolkien', 'Genre': 'Epic fantasy', 'Price': 100, 'Title': 'The F
ellowship of the Ring'}
```

In [15]:
```python
# Write using push() function
# Note that a set() is called on top of push()
#
ref = db.reference("/")
ref.set({
        "Books":
        {
                "Best_Sellers": -1
        }
})

ref = db.reference("/Books/Best_Sellers")

for key, value in data.items():
        ref.push().set(value)
```

In [16]:
```python
# 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})
```

```
{'-OYBgwlzSLpfRFUMlpSJ': {'Author': 'J.R.R. Tolkien', 'Genre': 'Epic fantasy', 'Pri
ce': 100, 'Title': 'The Fellowship of the Ring'}, '-OYBgwq_gBd9QGGtkdmT': {'Autho
r': 'J.R.R. Tolkien', 'Genre': 'Epic fantasy', 'Price': 100, 'Title': 'The Two Towe
rs'}, '-OYBgwv4ldf_1kfyjMCX': {'Author': 'J.R.R. Tolkien', 'Genre': 'Epic fantasy',
'Price': 100, 'Title': 'The Return of the King'}, '-OYBgwyrEyWmJPKElnxY': {'Autho
r': 'Paulo Coelho', 'Genre': 'Fiction', 'Price': 100, 'Title': 'Brida'}}
```

In [17]:
```python
# Let's delete all best seller books
# with J.R.R. Tolkien as the author.
#
ref = db.reference("/Books/Best_Sellers")
```

```
for key, value in best_sellers.items():
        if(value["Author"] == "J.R.R. Tolkien"):
                ref.child(key).set({})
```

In [18]:
```
# Delete all best_seller data.
#
ref = db.reference("/Books/Best_Sellers/")
best_sellers = ref.get()
print(best_sellers)
print(type(best_sellers))
```

```
{'-OYBgwyrEyWmJPKElnxY': {'Author': 'Paulo Coelho', 'Genre': 'Fiction', 'Price': 10
0, 'Title': 'Brida'}}
<class 'dict'>
```

In [ ]:
```
ref = db.reference("/Books/Best_Sellers")
ref.set({})
```

In [3]:
```python
# Fill in student ID and name
#
student_id = "223737376"
student_first_last_name = "Nawal"
print(student_id, student_first_last_name)
```

223737376 Nawal

In [4]:
```python
"""
    Firebase Realtime database demonstration.
"""

# Install libraries, if not yet.
! pip install firebase_admin pandas

import firebase_admin
from firebase_admin import credentials

databaseURL = "https://sit225n-default-rtdb.asia-southeast1.firebasedatabase.app/"
cred_obj = credentials.Certificate("sit225n-firebase-adminsdk-fbsvc-85b5c435df.jso

# Check if default app already exists
if not firebase_admin._apps:
    default_app = firebase_admin.initialize_app(cred_obj, {
        'databaseURL': databaseURL
    })
```

Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: firebase_admin in c:\users\jalmi\appdata\local\packa
ges\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\py
thon313\site-packages (7.1.0)
Requirement already satisfied: pandas in c:\users\jalmi\appdata\local\packages\pyth
onsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python313
\site-packages (2.3.1)
Requirement already satisfied: cachecontrol>=0.14.3 in c:\users\jalmi\appdata\local
\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packa
ges\python313\site-packages (from firebase_admin) (0.14.3)
Requirement already satisfied: google-api-core<3.0.0dev,>=2.25.1 in c:\users\jalmi
\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcac
he\local-packages\python313\site-packages (from google-api-core[grpc]<3.0.0dev,>=2.
25.1; platform_python_implementation != "PyPy"->firebase_admin) (2.25.1)
Requirement already satisfied: google-cloud-firestore>=2.21.0 in c:\users\jalmi\app
data\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\l
ocal-packages\python313\site-packages (from firebase_admin) (2.21.0)
Requirement already satisfied: google-cloud-storage>=3.1.1 in c:\users\jalmi\appdat
a\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\loca
l-packages\python313\site-packages (from firebase_admin) (3.2.0)
Requirement already satisfied: pyjwt>=2.10.1 in c:\users\jalmi\appdata\local\packag
es\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\pyt
hon313\site-packages (from pyjwt[crypto]>=2.10.1->firebase_admin) (2.10.1)
Requirement already satisfied: httpx==0.28.1 in c:\users\jalmi\appdata\local\packag
es\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\pyt
hon313\site-packages (from httpx[http2]==0.28.1->firebase_admin) (0.28.1)
Requirement already satisfied: anyio in c:\users\jalmi\appdata\local\packages\pytho
nsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python313\s
ite-packages (from httpx==0.28.1->httpx[http2]==0.28.1->firebase_admin) (4.10.0)
Requirement already satisfied: certifi in c:\users\jalmi\appdata\local\packages\pyt
honsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python313
\site-packages (from httpx==0.28.1->httpx[http2]==0.28.1->firebase_admin) (2025.8.
3)
Requirement already satisfied: httpcore==1.* in c:\users\jalmi\appdata\local\packag
es\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\pyt
hon313\site-packages (from httpx==0.28.1->httpx[http2]==0.28.1->firebase_admin) (1.
0.9)
Requirement already satisfied: idna in c:\users\jalmi\appdata\local\packages\python
softwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python313\si
te-packages (from httpx==0.28.1->httpx[http2]==0.28.1->firebase_admin) (3.10)
Requirement already satisfied: h2<5,>=3 in c:\users\jalmi\appdata\local\packages\py
thonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python31
3\site-packages (from httpx[http2]==0.28.1->firebase_admin) (4.2.0)
Requirement already satisfied: googleapis-common-protos<2.0.0,>=1.56.2 in c:\users
\jalmi\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\lo
calcache\local-packages\python313\site-packages (from google-api-core<3.0.0dev,>=2.
25.1->google-api-core[grpc]<3.0.0dev,>=2.25.1; platform_python_implementation != "P
yPy"->firebase_admin) (1.70.0)
Requirement already satisfied: protobuf!=3.20.0,!=3.20.1,!=4.21.0,!=4.21.1,!=4.21.
2,!=4.21.3,!=4.21.4,!=4.21.5,<7.0.0,>=3.19.5 in c:\users\jalmi\appdata\local\packag
es\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\pyt
hon313\site-packages (from google-api-core<3.0.0dev,>=2.25.1->google-api-core[grpc]
<3.0.0dev,>=2.25.1; platform_python_implementation != "PyPy"->firebase_admin) (6.3
1.1)
Requirement already satisfied: proto-plus<2.0.0,>=1.22.3 in c:\users\jalmi\appdata
\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local
-packages\python313\site-packages (from google-api-core<3.0.0dev,>=2.25.1->google-a
pi-core[grpc]<3.0.0dev,>=2.25.1; platform_python_implementation != "PyPy"->firebase
_admin) (1.26.1)
Requirement already satisfied: google-auth<3.0.0,>=2.14.1 in c:\users\jalmi\appdata

\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local
-packages\python313\site-packages (from google-api-core<3.0.0dev,>=2.25.1->google-a
pi-core[grpc]<3.0.0dev,>=2.25.1; platform_python_implementation != "PyPy"->firebase
_admin) (2.40.3)
Requirement already satisfied: requests<3.0.0,>=2.18.0 in c:\users\jalmi\appdata\lo
cal\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-pa
ckages\python313\site-packages (from google-api-core<3.0.0dev,>=2.25.1->google-api-
core[grpc]<3.0.0dev,>=2.25.1; platform_python_implementation != "PyPy"->firebase_ad
min) (2.32.4)
Requirement already satisfied: grpcio<2.0.0,>=1.33.2 in c:\users\jalmi\appdata\loca
l\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-pack
ages\python313\site-packages (from google-api-core[grpc]<3.0.0dev,>=2.25.1; platfor
m_python_implementation != "PyPy"->firebase_admin) (1.74.0)
Requirement already satisfied: grpcio-status<2.0.0,>=1.33.2 in c:\users\jalmi\appda
ta\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\loc
al-packages\python313\site-packages (from google-api-core[grpc]<3.0.0dev,>=2.25.1;
platform_python_implementation != "PyPy"->firebase_admin) (1.74.0)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in c:\users\jalmi\appdata\loc
al\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-pac
kages\python313\site-packages (from google-auth<3.0.0,>=2.14.1->google-api-core<3.
0.0dev,>=2.25.1->google-api-core[grpc]<3.0.0dev,>=2.25.1; platform_python_implement
ation != "PyPy"->firebase_admin) (5.5.2)
Requirement already satisfied: pyasn1-modules>=0.2.1 in c:\users\jalmi\appdata\loca
l\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-pack
ages\python313\site-packages (from google-auth<3.0.0,>=2.14.1->google-api-core<3.0.
0dev,>=2.25.1->google-api-core[grpc]<3.0.0dev,>=2.25.1; platform_python_implementat
ion != "PyPy"->firebase_admin) (0.4.2)
Requirement already satisfied: rsa<5,>=3.1.4 in c:\users\jalmi\appdata\local\packag
es\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\pyt
hon313\site-packages (from google-auth<3.0.0,>=2.14.1->google-api-core<3.0.0dev,>=
2.25.1->google-api-core[grpc]<3.0.0dev,>=2.25.1; platform_python_implementation !=
"PyPy"->firebase_admin) (4.9.1)
Requirement already satisfied: hyperframe<7,>=6.1 in c:\users\jalmi\appdata\local\p
ackages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-package
s\python313\site-packages (from h2<5,>=3->httpx[http2]==0.28.1->firebase_admin) (6.
1.0)
Requirement already satisfied: hpack<5,>=4.1 in c:\users\jalmi\appdata\local\packag
es\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\pyt
hon313\site-packages (from h2<5,>=3->httpx[http2]==0.28.1->firebase_admin) (4.1.0)
Requirement already satisfied: h11>=0.16 in c:\users\jalmi\appdata\local\packages\p
ythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python3
13\site-packages (from httpcore==1.*->httpx==0.28.1->httpx[http2]==0.28.1->firebase
_admin) (0.16.0)
Requirement already satisfied: charset_normalizer<4,>=2 in c:\users\jalmi\appdata\l
ocal\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-p
ackages\python313\site-packages (from requests<3.0.0,>=2.18.0->google-api-core<3.0.
0dev,>=2.25.1->google-api-core[grpc]<3.0.0dev,>=2.25.1; platform_python_implementat
ion != "PyPy"->firebase_admin) (3.4.3)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\jalmi\appdata\local\p
ackages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-package
s\python313\site-packages (from requests<3.0.0,>=2.18.0->google-api-core<3.0.0dev,>
=2.25.1->google-api-core[grpc]<3.0.0dev,>=2.25.1; platform_python_implementation !=
"PyPy"->firebase_admin) (2.5.0)
Requirement already satisfied: pyasn1>=0.1.3 in c:\users\jalmi\appdata\local\packag
es\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\pyt
hon313\site-packages (from rsa<5,>=3.1.4->google-auth<3.0.0,>=2.14.1->google-api-co
re<3.0.0dev,>=2.25.1->google-api-core[grpc]<3.0.0dev,>=2.25.1; platform_python_impl
ementation != "PyPy"->firebase_admin) (0.6.1)
Requirement already satisfied: numpy>=1.26.0 in c:\users\jalmi\appdata\local\packag
es\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\pyt

```
hon313\site-packages (from pandas) (2.3.2)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\jalmi\appdata\loc
al\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-pac
kages\python313\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\jalmi\appdata\local\package
s\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\pyth
on313\site-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in c:\users\jalmi\appdata\local\packa
ges\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\py
thon313\site-packages (from pandas) (2025.2)
Requirement already satisfied: msgpack<2.0.0,>=0.5.2 in c:\users\jalmi\appdata\loca
l\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-pack
ages\python313\site-packages (from cachecontrol>=0.14.3->firebase_admin) (1.1.1)
Requirement already satisfied: google-cloud-core<3.0.0,>=1.4.1 in c:\users\jalmi\ap
pdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache
\local-packages\python313\site-packages (from google-cloud-firestore>=2.21.0->fireb
ase_admin) (2.4.3)
Requirement already satisfied: google-resumable-media<3.0.0,>=2.7.2 in c:\users\jal
mi\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localc
ache\local-packages\python313\site-packages (from google-cloud-storage>=3.1.1->fire
base_admin) (2.7.2)
Requirement already satisfied: google-crc32c<2.0.0,>=1.1.3 in c:\users\jalmi\appdat
a\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\loca
l-packages\python313\site-packages (from google-cloud-storage>=3.1.1->firebase_admi
n) (1.7.1)
Requirement already satisfied: cryptography>=3.4.0 in c:\users\jalmi\appdata\local
\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packa
ges\python313\site-packages (from pyjwt[crypto]>=2.10.1->firebase_admin) (45.0.6)
Requirement already satisfied: cffi>=1.14 in c:\users\jalmi\appdata\local\packages
\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\pytho
n313\site-packages (from cryptography>=3.4.0->pyjwt[crypto]>=2.10.1->firebase_admi
n) (1.17.1)
Requirement already satisfied: pycparser in c:\users\jalmi\appdata\local\packages\p
ythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python3
13\site-packages (from cffi>=1.14->cryptography>=3.4.0->pyjwt[crypto]>=2.10.1->fire
base_admin) (2.22)
Requirement already satisfied: six>=1.5 in c:\users\jalmi\appdata\local\packages\py
thonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python31
3\site-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
Requirement already satisfied: sniffio>=1.1 in c:\users\jalmi\appdata\local\package
s\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\pyth
on313\site-packages (from anyio->httpx==0.28.1->httpx[http2]==0.28.1->firebase_admi
n) (1.3.1)
```

```python
In [5]:   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)
```

In [18]:
```
ref = db.reference("/")  # set ref point

# query all data under the ref
books = ref.get()
print(books)
print(type(books))

# print each item separately
for key, value in books.items():
    print(f"{key}: {value}")


# Query /Book1
ref = db.reference("/Book1")
books = ref.get()
print(books)
```

```
{'Book1': {'Author': 'J.R.R. Tolkien', 'Genre': 'Epic fantasy', 'Price': 100, 'Titl
e': 'The Fellowship of the Ring'}, 'Book2': {'Author': 'J.R.R. Tolkien', 'Genre':
'Epic fantasy', 'Price': 100, 'Title': 'The Two Towers'}, 'Book3': {'Author': 'J.R.
R. Tolkien', 'Genre': 'Epic fantasy', 'Price': 100, 'Title': 'The Return of the Kin
g'}, 'Book4': {'Author': 'Paulo Coelho', 'Genre': 'Fiction', 'Price': 100, 'Title':
'Brida'}}
<class 'dict'>
Book1: {'Author': 'J.R.R. Tolkien', 'Genre': 'Epic fantasy', 'Price': 100, 'Title':
'The Fellowship of the Ring'}
Book2: {'Author': 'J.R.R. Tolkien', 'Genre': 'Epic fantasy', 'Price': 100, 'Title':
'The Two Towers'}
Book3: {'Author': 'J.R.R. Tolkien', 'Genre': 'Epic fantasy', 'Price': 100, 'Title':
'The Return of the King'}
Book4: {'Author': 'Paulo Coelho', 'Genre': 'Fiction', 'Price': 100, 'Title': 'Brid
a'}
{'Author': 'J.R.R. Tolkien', 'Genre': 'Epic fantasy', 'Price': 100, 'Title': 'The F
ellowship of the Ring'}
```

In [19]:
```python
# Write using push() function
# Note that a set() is called on top of push()
#
ref = db.reference("/")
ref.set({
        "Books":
        {
                "Best_Sellers": -1
        }
})

ref = db.reference("/Books/Best_Sellers")

for key, value in data.items():
        ref.push().set(value)
```

In [20]:
```python
# 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":
        ref.child(key).update({"Price": 80})
```

```
{'-OYBU4G9oR0LVUPkt30q': {'Author': 'J.R.R. Tolkien', 'Genre': 'Epic fantasy', 'Pri
ce': 100, 'Title': 'The Fellowship of the Ring'}, '-OYBU4K5QQ11l15lr3-L': {'Autho
r': 'J.R.R. Tolkien', 'Genre': 'Epic fantasy', 'Price': 100, 'Title': 'The Two Towe
rs'}, '-OYBU4NpeyRP3koweOXe': {'Author': 'J.R.R. Tolkien', 'Genre': 'Epic fantasy',
'Price': 100, 'Title': 'The Return of the King'}, '-OYBU4RZJ-qZBBQIxgnU': {'Autho
r': 'Paulo Coelho', 'Genre': 'Fiction', 'Price': 100, 'Title': 'Brida'}}
```

In [21]:
```python
# Let's delete all best seller books
# with J.R.R. Tolkien as the author.

ref = db.reference("/Books/Best_Sellers")
best_sellers = ref.get()   # refresh the data
```

```python
for key, value in best_sellers.items():
    if value["Author"] == "J.R.R. Tolkien":
        ref.child(key).set({})
```

In [22]:
```python
# Delete all best_seller data.
ref = db.reference("/Books/Best_Sellers/")
ref.set({})
```

In [ ]:
```python
ref = db.reference("/Books/Best_Sellers")
ref.set({})
```