**BIG DATA MANAGEMENT**

**POST GRADUATE DIPLOMA**

**IN DATA ENGINEERING**

## ASSIGNMENT 4

**SUBMITTED BY:**

**NIRAJ BHAGCHANDANI [G23AI2087]**

****

**SUBMISSION DATE: 12th December, 2024**

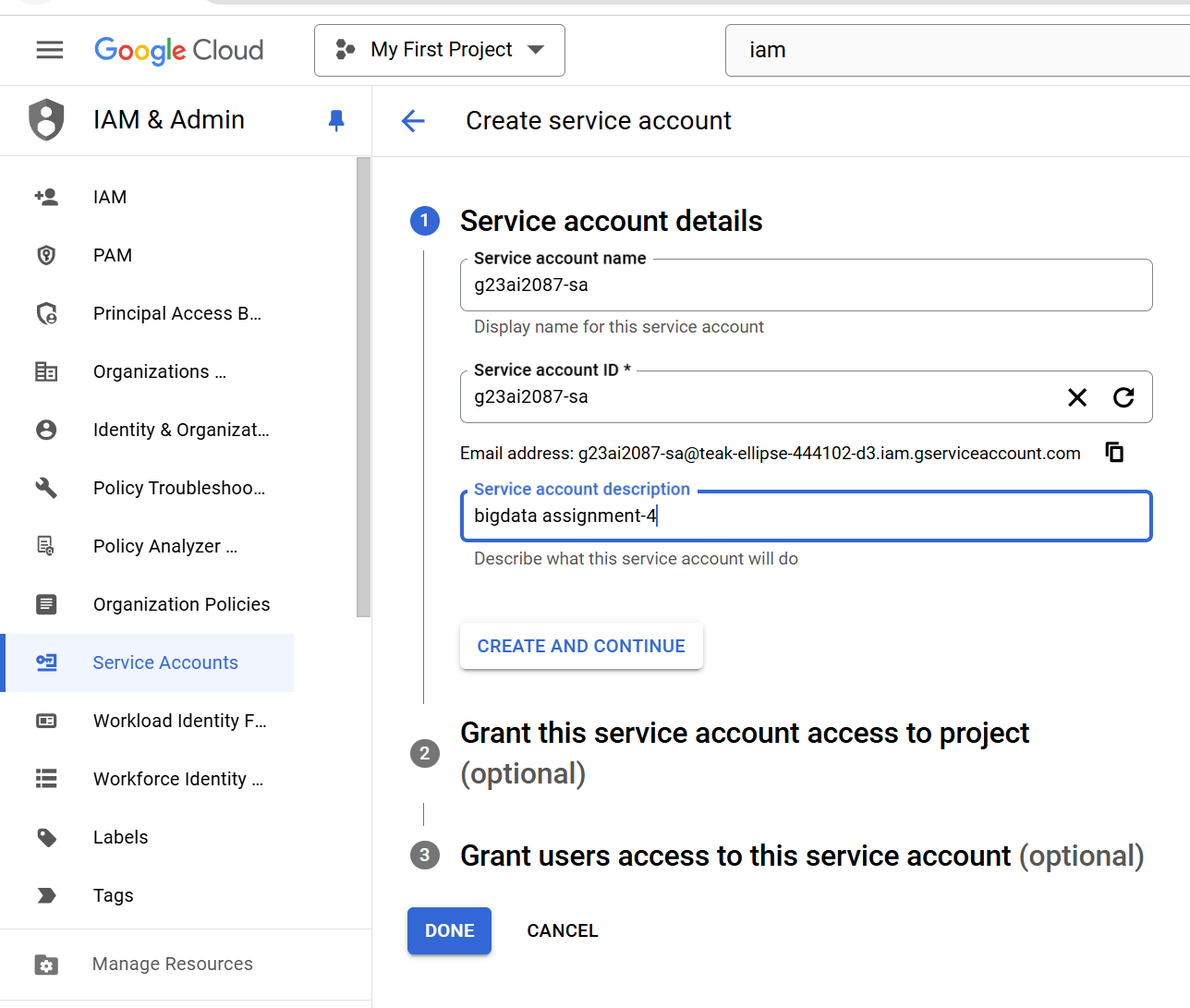
**DEPARTMENT OF AIDE**

**INDIAN INSTITUTE OF TECHNOLOGY, JODHPUR**

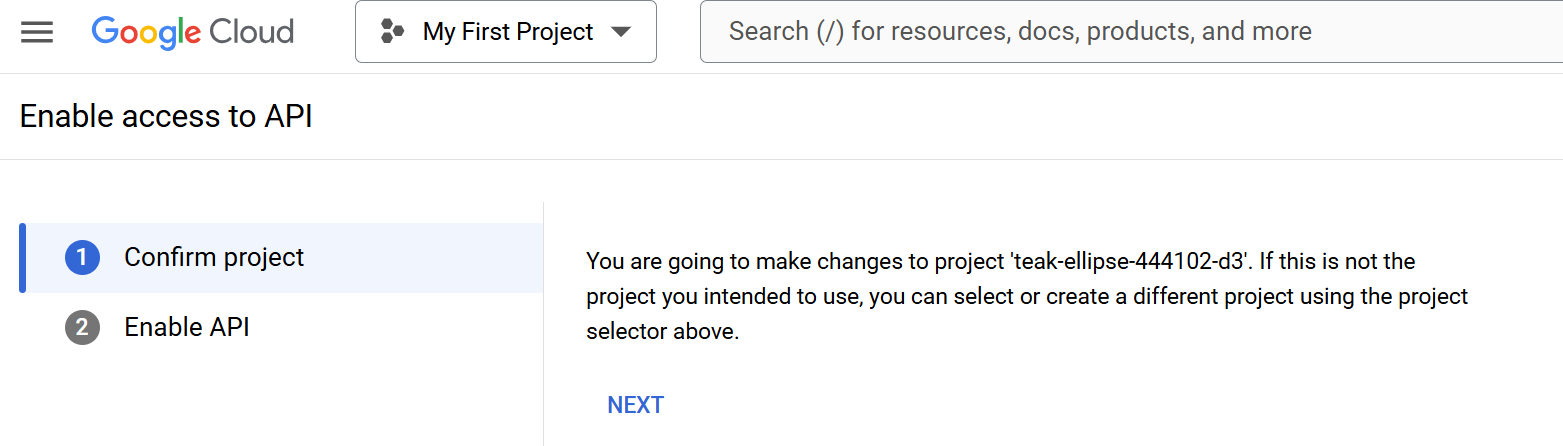
Assignment-4

* Google Big Table: 50 points
* Connecting to the Instance after setting it up as explained in class.
* Connecting to the database can be done using the cbt command-line tool or using a Bigtable client library. Google Cloud Bigtable is not a relational database and is NOT accessible using SQuirreL or other SQL tools.
* Accessing using cbt command-line tool
* The cbt command-line interface allows performing basic administrative tasks and reading/writing data from tables. There is a tutorial on cbt CLI found here:
* https://cloud.google.com/bigtable/docs/create-instance-write-data-cbt-cli?\_ga=2.111890764.-9 13511634.1664467746
* Accessing using Client Library
* The lab will use the Java client library. An example code file called HelloWorld.java shown in class. This sample creates a table, writes data, reads data, then deletes the table. There is more information on this "Hello world" example. Found here:
* https://cloud.google.com/bigtable/docs/samples-java-hello-world For setup, follow these instructions. From here:
* https://cloud.google.com/docs/authentication/provide-credentials-adc You will need install the Google Cloud CLI then run the command: gcloud auth application-default login.

**Step :1 Assign roles like BigQuery Admin or skip this step to configure permissions later.**



**Step 2: select the specific API you want to enable for your project (e.g., BigQuery API, Cloud Storage API) and click Enable to grant access.**

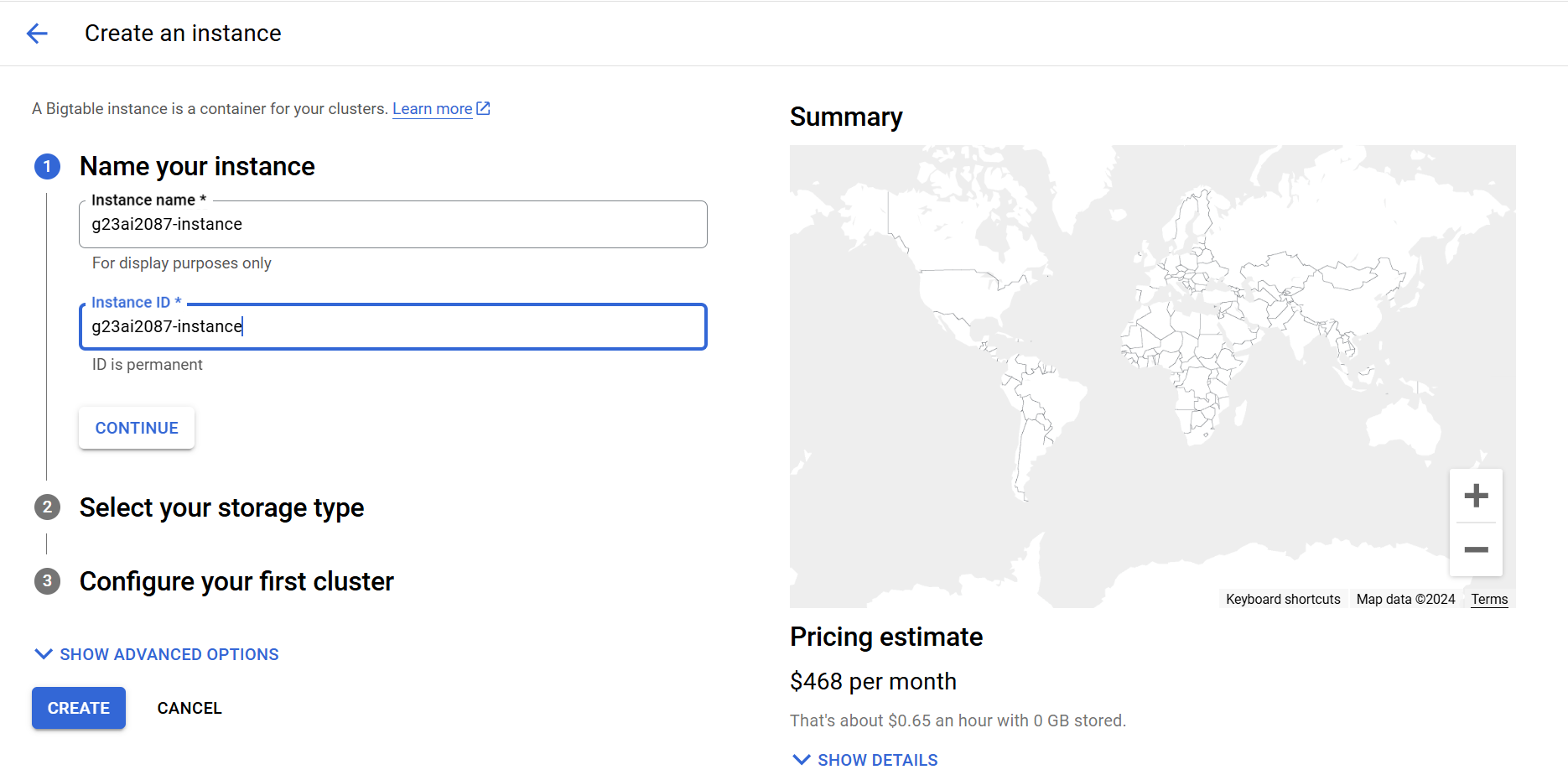


**Step 3 : Once the API is enabled, verify that the BigQuery API (or the selected API) is successfully activated for your project. You can proceed to configure or use the API as needed.**

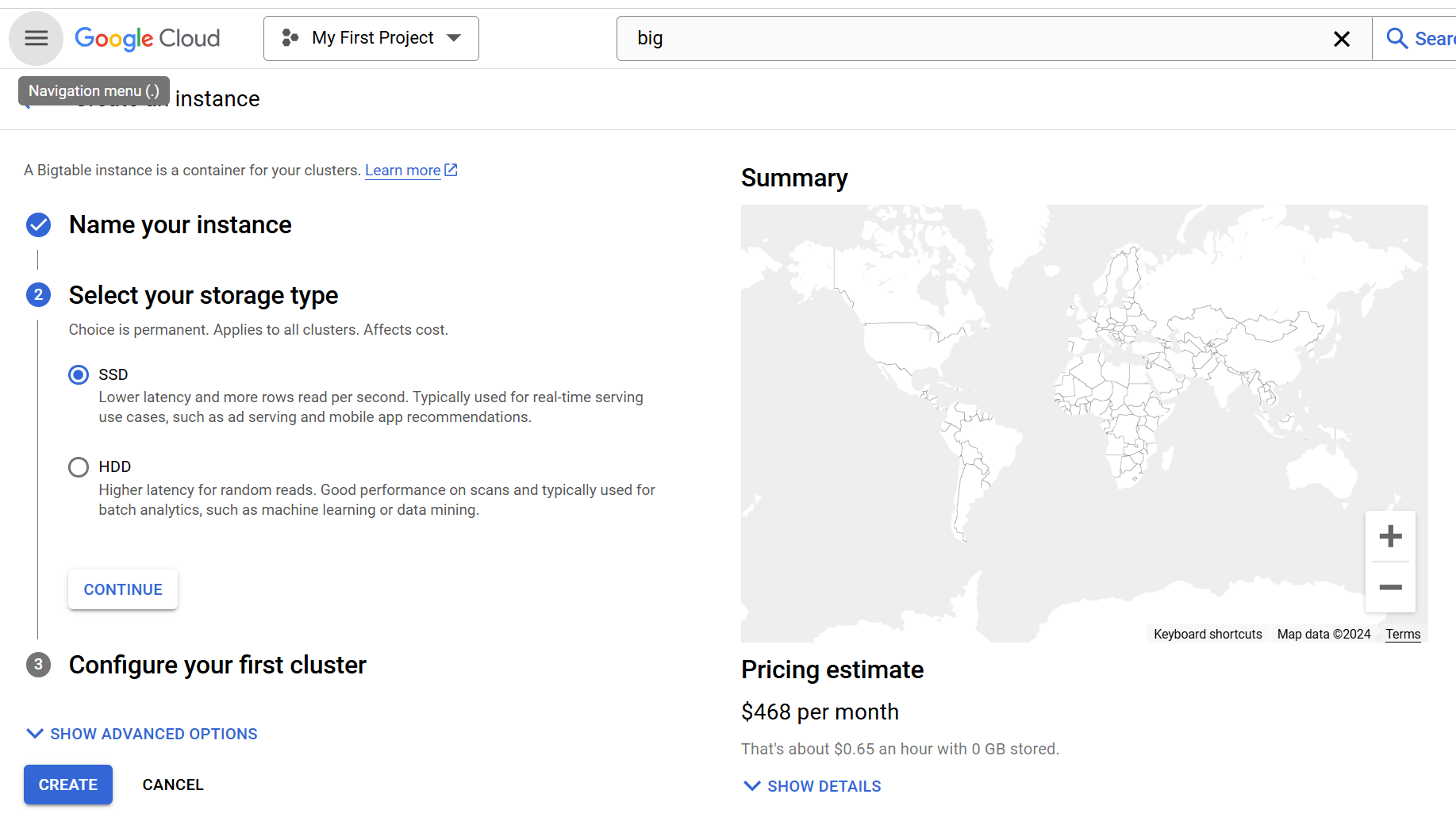
A screenshot of a computer

Description automatically generated

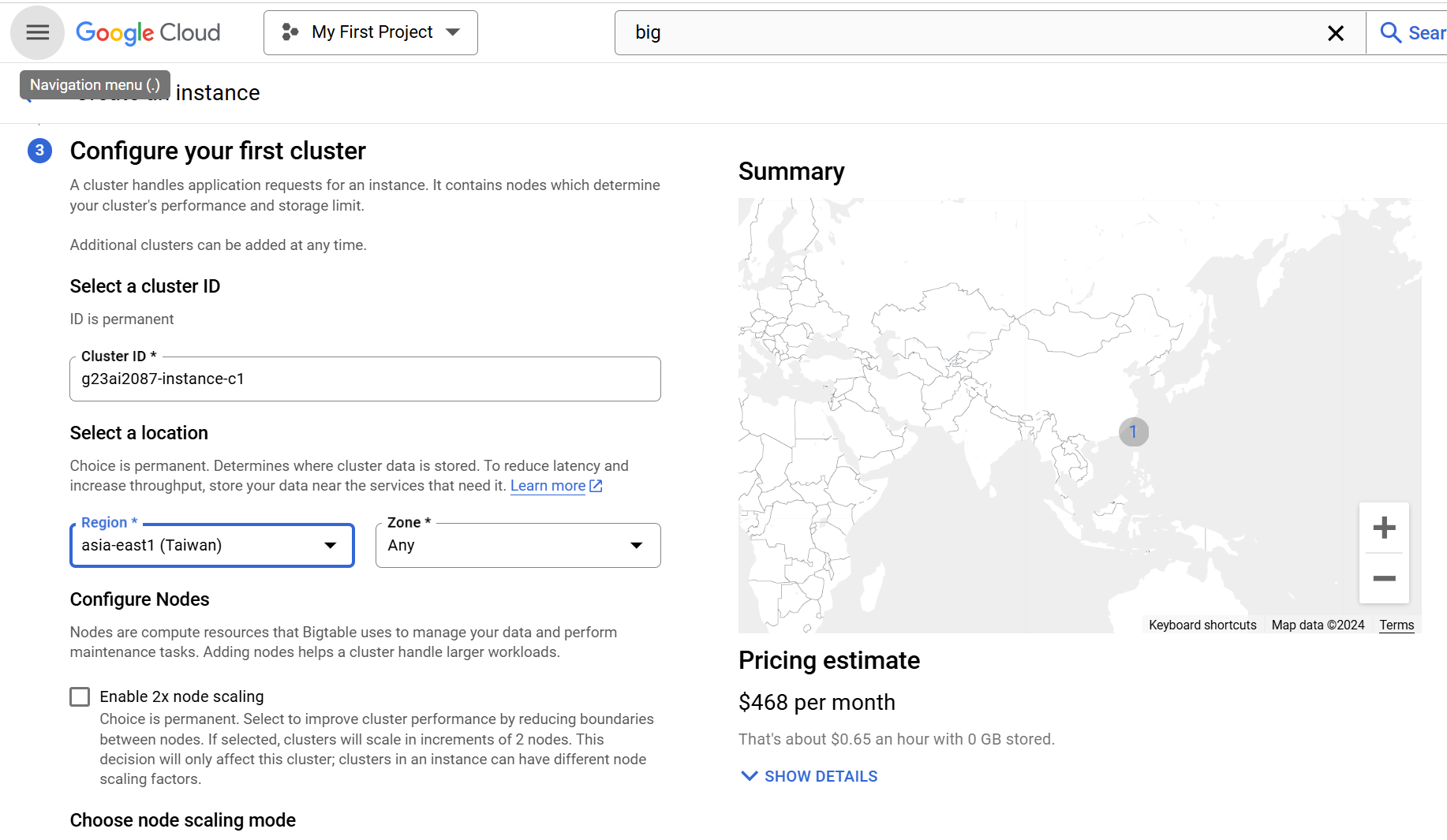
**Step 4: After naming your instance, selecting the storage type, and configuring your first cluster, review all the details and click CREATE to finalize the instance setup.**



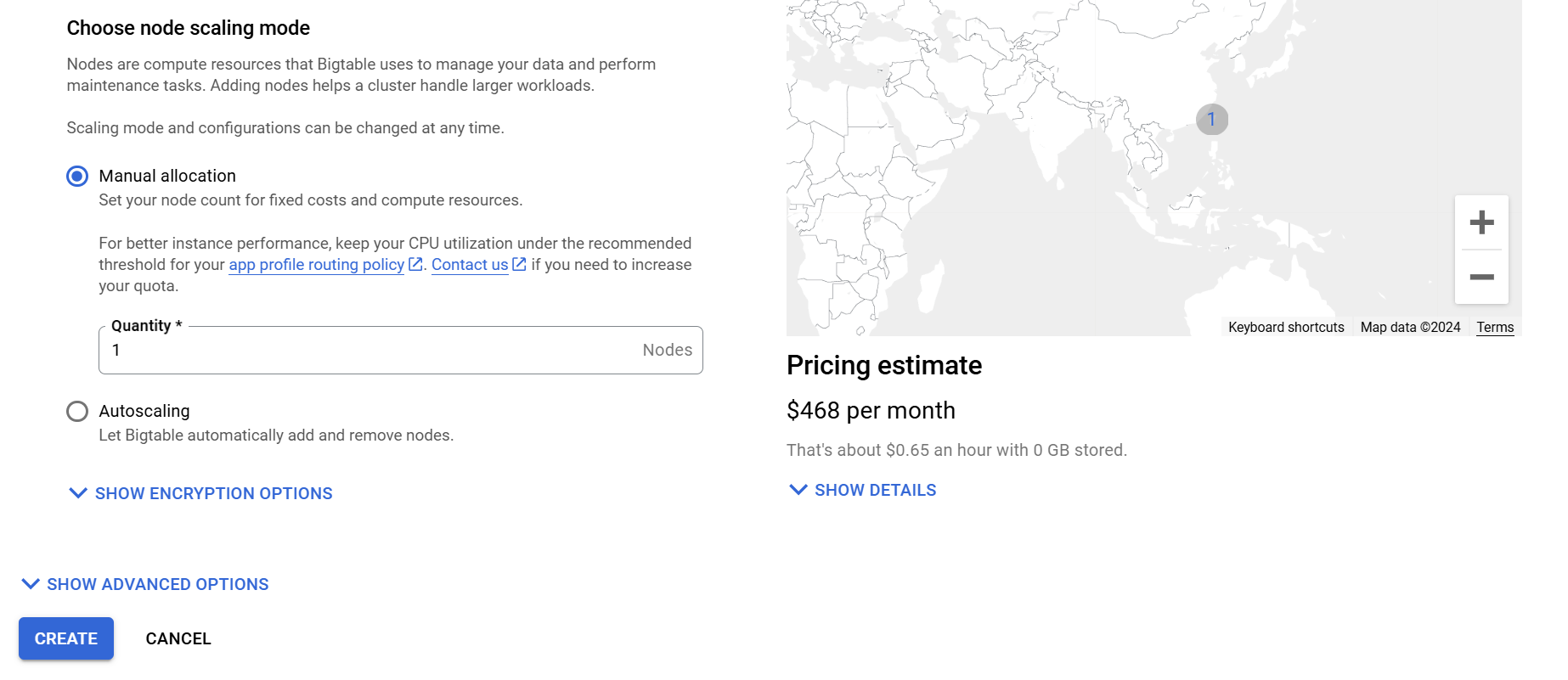
**Step 5 : Configure your first cluster by providing the cluster ID, selecting the region, and setting the number of nodes. Review the configuration and proceed by clicking CREATE to finalize your instance and cluster setup.**



**Step 6 : finalize your cluster setup by reviewing the configuration, including the cluster ID, region, and node settings. Once confirmed, click CREATE to provision the cluster and complete the setup process.**



**Step 7 : choose the node scaling mode: either Manual allocation (set a fixed number of nodes, e.g., Quantity: 1) or Autoscaling (nodes automatically scale based on workload). Once selected, review the configuration and click CREATE to complete the instance creation.**



**Full Code:**

import com.google.api.gax.rpc.NotFoundException;

import com.google.cloud.bigtable.admin.v2.BigtableTableAdminClient;

import com.google.cloud.bigtable.admin.v2.BigtableTableAdminSettings;

import com.google.cloud.bigtable.data.v2.BigtableDataClient;

import com.google.cloud.bigtable.data.v2.BigtableDataSettings;

import com.google.cloud.bigtable.data.v2.BigtableDataClient;

import com.google.cloud.bigtable.data.v2.models.Mutation;

import com.google.cloud.bigtable.data.v2.models.RowMutation;

import com.google.cloud.bigtable.admin.v2.models.CreateTableRequest;

import com.google.cloud.bigtable.admin.v2.models.ModifyColumnFamiliesRequest;

import com.google.cloud.bigtable.data.v2.BigtableDataClient;

import com.google.cloud.bigtable.data.v2.models.RowMutation;

import com.google.cloud.bigtable.data.v2.BigtableDataClient;

import com.google.cloud.bigtable.data.v2.models.Row;

import com.google.cloud.bigtable.data.v2.models.Filters;

import com.google.cloud.bigtable.data.v2.models.Query;

import com.google.cloud.bigtable.data.v2.models.RowCell;

import java.util.ArrayList;

import java.util.Scanner;

import java.io.\*;

import java.util.List;

public class Bigtable {

    public final String projectId = "teak-ellipse-444102-d3";

    public final String instanceId = "g23ai2087-instance";

    public final String COLUMN\_FAMILY = "sensor";

    public final String tableId = "weather";

    private BigtableDataClient bigtableClient;

    private BigtableDataClient dataClient;

    private BigtableTableAdminClient adminClient;

    private static final String TEMPERATURE\_COLUMN = "temperature";

    public static void main(String[] args) throws Exception {

        Bigtable bigtable = new Bigtable();

        Scanner scanner = new Scanner(System.in);

        while (true) {

            System.out.println("\n=== Bigtable Menu ===");

            System.out.println("1. Connect to Bigtable");

            System.out.println("2. Create Table");

            System.out.println("3. Delete Table");

            System.out.println("4. Load Data");

            System.out.println("5. Query 1: Temperature at Vancouver");

            System.out.println("6. Query 2: Highest Wind Speed in Portland");

            System.out.println("7. Query 3: All Readings for SeaTac");

            System.out.println("8. Query 4: Highest Temperature in Summer");

            System.out.println("9. Exit");

            System.out.print("Enter your choice: ");

            int choice = scanner.nextInt();

            switch (choice) {

                case 1:

                    bigtable.connect();

                    break;

                case 2:

                    bigtable.createTable();

                    break;

                case 3:

                    bigtable.deleteTable();

                    break;

                case 4:

                    bigtable.loadData();

                    break;

                case 5:

                    System.out.println("Temperature: " + bigtable.query1());

                    break;

                case 6:

                    System.out.println("Highest Wind Speed: " + bigtable.query2(bigtable.tableId,"Portland"));

                    break;

                case 7:

                    List<String> readings = bigtable.query3();

                    for (String reading : readings) {

                        System.out.println(reading);

                    }

                    break;

                case 8:

                    System.out.println("Highest Temperature in Summer: " + bigtable.query4());

                    break;

                case 9:

                    System.out.println("Exiting...");

                    bigtable.close();

                    scanner.close();

                    return;

                default:

                    System.out.println("Invalid choice! Please try again.");

            }

        }

    }

    public void connect() throws IOException {

        System.out.println("Establishing connection to Bigtable...");

        // Initialize data client

        BigtableDataSettings dataSettings = BigtableDataSettings.newBuilder()

                .setProjectId(projectId)

                .setInstanceId(instanceId)

                .build();

        dataClient = BigtableDataClient.create(dataSettings);

        // Initialize admin client

        BigtableTableAdminSettings adminSettings = BigtableTableAdminSettings.newBuilder()

                .setProjectId(projectId)

                .setInstanceId(instanceId)

                .build();

        adminClient = BigtableTableAdminClient.create(adminSettings);

        bigtableClient = BigtableDataClient.create(dataSettings);

        try {

            if (!adminClient.exists(tableId)) {

                System.out.println("Table '" + tableId + "' not found. Creating a new table...");

                createTable();

                System.out.println("Table '" + tableId + "' created successfully.");

            } else {

                System.out.println("Table '" + tableId + "' already exists.");

            }

        } catch (Exception e) {

            System.err.println("Error during connection setup: " + e.getMessage());

        }

        System.out.println("Connection to Bigtable established successfully.");

    }

    public void createTable() {

        try {

            CreateTableRequest createTableRequest = CreateTableRequest.of(tableId)

                    .addFamily(COLUMN\_FAMILY);

            adminClient.createTable(createTableRequest);

            System.out.println("Table " + tableId + " created successfully with column family: " + COLUMN\_FAMILY);

        } catch (Exception e) {

            System.err.println("Error creating table: " + e.getMessage());

        }

    }

    public void deleteTable() {

        System.out.println("Deleting table: " + tableId);

        try {

            adminClient.deleteTable(tableId);

            System.out.println("Table " + tableId + " deleted successfully.");

        } catch (NotFoundException e) {

            System.err.println("Table does not exist: " + e.getMessage());

        }

    }

    public void loadData() {

        System.out.println("Loading data into table...");

        String[] files = {"data/portland.csv", "data/seatac.csv", "data/vancouver.csv"};

        int batchSize = 500;

        int count = 1;

        try {

            for (String file : files) {

                System.out.println("Reading file: " + file);

                BufferedReader br = new BufferedReader(new FileReader(file));

                String line = br.readLine();

                int cnt = 1;

                String fileName = new File(file).getName();

                String city = fileName.substring(0, fileName.lastIndexOf(".csv"));

                List<RowMutation> rowMutations = new ArrayList<>();

                while ((line = br.readLine()) != null) {

                    String[] values = line.split(",");

                    String date = values[1].trim();

                    String time = values[2].trim();

                    String temperature = values[3].trim();

                    String dewpoint = values[4].trim();

                    String humidity = values[5].trim();

                    String windspeed = values[6].trim();

                    String pressure = values[7].trim();

                    String rowKey = date + "\_" + time;

                    rowMutations.add(RowMutation.create(tableId, rowKey)

                            .setCell(COLUMN\_FAMILY, "temperature", temperature)

                            .setCell(COLUMN\_FAMILY, "dewpoint", dewpoint)

                            .setCell(COLUMN\_FAMILY, "humidity", humidity)

                            .setCell(COLUMN\_FAMILY, "windspeed", windspeed)

                            .setCell(COLUMN\_FAMILY, "pressure", pressure)

                            .setCell(COLUMN\_FAMILY, "city", city));

                    System.out.println(cnt++ +")City: "+city+", Temp:" +temperature+", Dew:"+dewpoint+", humidity:"+humidity+", ws:"+windspeed+", press:"+pressure);

                    if (rowMutations.size() == batchSize) {

                        batchInsert(rowMutations);

                        rowMutations.clear();

                        System.out.println("Batch of " + batchSize + " rows inserted.");

                    }

                    System.out.println("Row: " + ++count + " Prepared!");

                }

                if (!rowMutations.isEmpty()) {

                    batchInsert(rowMutations);

                    System.out.println("Final batch for " + city + " inserted.");

                }

                br.close();

            }

            System.out.println("Data loaded successfully.");

        } catch (Exception e) {

            System.err.println("Error loading data: " + e.getMessage());

            e.printStackTrace();

        }

    }

    private void batchInsert(List<RowMutation> rowMutations) {

        for (RowMutation mutation : rowMutations) {

            try {

                bigtableClient.mutateRow(mutation);

            } catch (Exception e) {

                System.err.println("Error inserting batch: " + e.getMessage());

                e.printStackTrace();

            }

        }

    }

    public String query1() {

        String rowKey = "2022-10-01\_10:00";

        try {

            Row row = bigtableClient.readRow(tableId, rowKey);

            if (row != null && row.getCells(COLUMN\_FAMILY, TEMPERATURE\_COLUMN).size() > 0) {

                // Get the latest value for the temperature column

                return row.getCells(COLUMN\_FAMILY, TEMPERATURE\_COLUMN).get(0).getValue().toStringUtf8();

            } else {

                return "Temperature data not found for the given row key.";

            }

        } catch (Exception e) {

            System.err.println("Error querying Bigtable: " + e.getMessage());

            e.printStackTrace();

            return "Error occurred while querying.";

        }

    }

    public String query2(String tableId, String city) {

        try {

            String rowKeyPrefix = city + "\_2022-09";

            String highestWindSpeedRowKey = null;

            double maxWindSpeed = Double.MIN\_VALUE;

            Query query = Query.create(tableId).prefix(rowKeyPrefix);

            for (Row row : bigtableClient.readRows(query)) {

                List<RowCell> windspeedCells = row.getCells(COLUMN\_FAMILY, "windspeed");

                for (RowCell cell : windspeedCells) {

                    String windSpeedValue = cell.getValue().toStringUtf8();

                    double windSpeed = Double.parseDouble(windSpeedValue);

                    if (windSpeed > maxWindSpeed) {

                        maxWindSpeed = windSpeed;

                        highestWindSpeedRowKey = row.getKey().toStringUtf8();

                    }

                }

            }

            if (highestWindSpeedRowKey != null) {

                return "Highest wind speed in September 2022 in Portland: " + maxWindSpeed + " km/h";

            } else {

                return "No data found for the specified query.";

            }

        } catch (Exception e) {

            e.printStackTrace();

            return "Error occurred while querying data.";

        }

    }

    public List<String> query3() {

        System.out.println("Executing Query 3: Retrieving readings for SeaTac on October 1, 2022, at 10:00.");

        String tableId = this.tableId;

        String rowKey = "2022-10-01\_10:00";

        List<String> readings = new ArrayList<>();

        try {

            Row row = bigtableClient.readRow(tableId, rowKey);

            if (row != null) {

                StringBuilder rowData = new StringBuilder();

                rowData.append("RowKey: ").append(row.getKey().toStringUtf8()).append(", Data: {");

                // Iterate over all cells in the row

                for (RowCell cell : row.getCells()) {

                    String columnQualifier = cell.getQualifier().toStringUtf8();

                    String value = cell.getValue().toStringUtf8();

                    rowData.append(columnQualifier).append(": ").append(value).append(", ");

                }

                if (rowData.lastIndexOf(", ") > 0) {

                    rowData.delete(rowData.lastIndexOf(", "), rowData.length());

                }

                rowData.append("}");

                readings.add(rowData.toString());

            } else {

                readings.add("No data found for row key: " + rowKey);

            }

        } catch (Exception e) {

            e.printStackTrace();

            readings.add("Error occurred while querying data.");

        }

        return readings;

    }

    // Query 4

    public int query4() {

        System.out.println("Executing Query 4: Highest Temperature in Summer 2022 (July, August).");

        String tableId = "weather";

        String[] summerMonths = {"2022-07", "2022-08"};

        int highestTemperature = Integer.MIN\_VALUE;

        try {

            for (String month : summerMonths) {

                String rowKeyPrefix = month;

                Query query = Query.create(tableId).prefix(rowKeyPrefix);

                for (Row row : bigtableClient.readRows(query)) {

                    for (RowCell cell : row.getCells(COLUMN\_FAMILY, "temperature")) {

                        try {

                            int temp = Integer.parseInt(cell.getValue().toStringUtf8());

                            if (temp > highestTemperature) {

                                highestTemperature = temp; // Update the highest temperature

                            }

                        } catch (NumberFormatException e) {

                            System.err.println("Invalid temperature value: " + cell.getValue().toStringUtf8());

                        }

                    }

                }

            }

            if (highestTemperature == Integer.MIN\_VALUE) {

                System.out.println("No temperature data found for summer months.");

            } else {

                System.out.println("Highest Temperature in Summer 2022: " + highestTemperature);

            }

        } catch (Exception e) {

            e.printStackTrace();

            System.out.println("Error occurred while querying data.");

        }

        return highestTemperature;

    }

    public void close() {

        if (dataClient != null) {

            dataClient.close();

        }

        if (adminClient != null) {

            adminClient.close();

        }

        System.out.println("Closed Bigtable connections.");

    }

}

1. **Write the method connect() to create a connection. Create a Bigtable data client and admin client. See HelloWorld.java for starter code.**

**Code:**

public void connect() throws IOException {

        System.out.println("Establishing connection to Bigtable...");

        BigtableDataSettings dataSettings = BigtableDataSettings.newBuilder()

                .setProjectId(projectId)

                .setInstanceId(instanceId)

                .build();

        dataClient = BigtableDataClient.create(dataSettings);

        BigtableTableAdminSettings adminSettings = BigtableTableAdminSettings.newBuilder()

                .setProjectId(projectId)

                .setInstanceId(instanceId)

                .build();

        adminClient = BigtableTableAdminClient.create(adminSettings);

        bigtableClient = BigtableDataClient.create(dataSettings);

        try {

            if (!adminClient.exists(tableId)) {

                System.out.println("Table '" + tableId + "' not found. Creating a new table...");

                createTable();

                System.out.println("Table '" + tableId + "' created successfully.");

            } else {

                System.out.println("Table '" + tableId + "' already exists.");

            }

        } catch (Exception e) {

            System.err.println("Error during connection setup: " + e.getMessage());

        }

        System.out.println("Connection to Bigtable established successfully.");

    }

10 mark - Write the method createTable() to create a table to store the sensor data.

public void createTable() {

        try {

            CreateTableRequest createTableRequest = CreateTableRequest.of(tableId)

                    .addFamily(COLUMN\_FAMILY); // Add the column family

            adminClient.createTable(createTableRequest);

            System.out.println("Table " + tableId + " created successfully with column family: " + COLUMN\_FAMILY);

        } catch (Exception e) {

            System.err.println("Error creating table: " + e.getMessage());

        }

    }

    public void deleteTable() {

        System.out.println("Deleting table: " + tableId);

        try {

            adminClient.deleteTable(tableId);

            System.out.println("Table " + tableId + " deleted successfully.");

        } catch (NotFoundException e) {

            System.err.println("Table does not exist: " + e.getMessage());

        }

    }

**Output:**

A black screen with white text

Description automatically generated

*Fig 4.1 Command-line interface menu for Google Cloud Bigtable, showing options to manage tables and run queries. The connection to Bigtable is successfully established, with the "weather" table already existing.*

1. **Write the method load() to load the sensor data into the database. The data files are in the data folder.**

**Code:**

public void loadData() {

        System.out.println("Loading data into table...");

        String[] files = {"data/portland.csv", "data/seatac.csv", "data/vancouver.csv"};

        int batchSize = 500;

        int count = 1;

        try {

            for (String file : files) {

                System.out.println("Reading file: " + file);

                BufferedReader br = new BufferedReader(new FileReader(file));

                String line = br.readLine();

                int cnt = 1;

                String fileName = new File(file).getName();

                String city = fileName.substring(0, fileName.lastIndexOf(".csv"));

                List<RowMutation> rowMutations = new ArrayList<>();

                while ((line = br.readLine()) != null) {

                    String[] values = line.split(",");

                    String date = values[1].trim();

                    String time = values[2].trim();

                    String temperature = values[3].trim();

                    String dewpoint = values[4].trim();

                    String humidity = values[5].trim();

                    String windspeed = values[6].trim();

                    String pressure = values[7].trim();

                    String rowKey = date + "\_" + time;

                    rowMutations.add(RowMutation.create(tableId, rowKey)

                            .setCell(COLUMN\_FAMILY, "temperature", temperature)

                            .setCell(COLUMN\_FAMILY, "dewpoint", dewpoint)

                            .setCell(COLUMN\_FAMILY, "humidity", humidity)

                            .setCell(COLUMN\_FAMILY, "windspeed", windspeed)

                            .setCell(COLUMN\_FAMILY, "pressure", pressure)

                            .setCell(COLUMN\_FAMILY, "city", city));

                    System.out.println(cnt++ +")City: "+city+", Temp:" +temperature+", Dew:"+dewpoint+", humidity:"+humidity+", ws:"+windspeed+", press:"+pressure);

                    // Process batch

                    if (rowMutations.size() == batchSize) {

                        batchInsert(rowMutations);

                        rowMutations.clear();

                        System.out.println("Batch of " + batchSize + " rows inserted.");

                    }

                    System.out.println("Row: " + ++count + " Prepared!");

                }

                if (!rowMutations.isEmpty()) {

                    batchInsert(rowMutations);

                    System.out.println("Final batch for " + city + " inserted.");

                }

                br.close();

            }

            System.out.println("Data loaded successfully.");

        } catch (Exception e) {

            System.err.println("Error loading data: " + e.getMessage());

            e.printStackTrace();

        }

    }

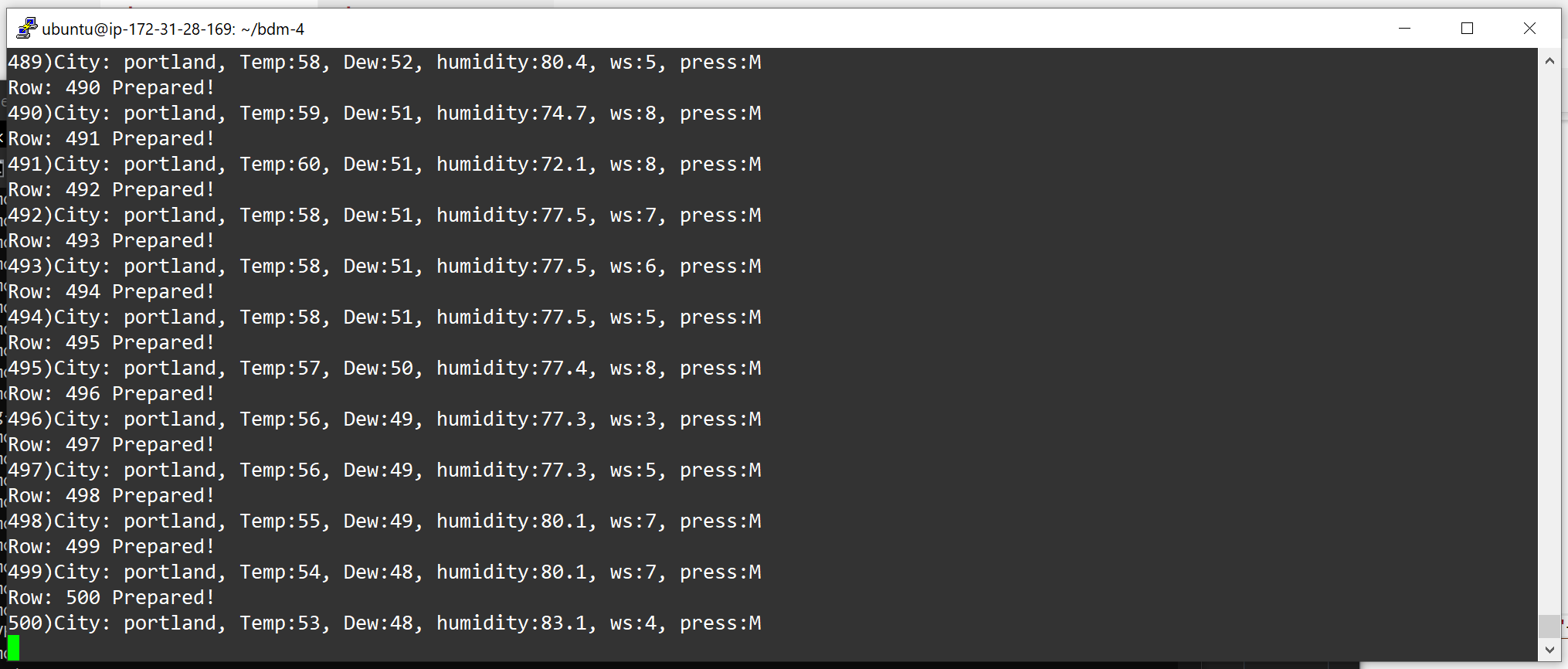
Screenshot – 1

A screenshot of a computer program

Description automatically generated

*Fig.4.2 A command-line interface for Google Cloud Bigtable showing the process of loading data into a table. Data from the file data/portland.csv is being prepared and inserted into rows with attributes like city, temperature, dew point, humidity, wind speed, and pressure.*

Screenshot - 2

  
*Fig.4.3 Rows of weather data, including attributes like temperature, dew point, humidity, wind speed, and pressure, are successfully prepared and inserted into a Bigtable instance, reaching row 500.*

Screenshot – 3

A screenshot of a computer

Description automatically generated

*Fig.4.4 Weather data for SeaTac, including temperature, dew point, humidity, wind speed, and pressure, is successfully prepared and inserted into a Bigtable instance, reaching row 3500.*

1. **Write the method query1() that returns the temperature at Vancouver on 2022-10-01 at 10 a.m.**

**Code:**

public String query1() {

        // Row key for the specific query

        String rowKey = "2022-10-01\_10:00";

        try {

            Row row = bigtableClient.readRow(tableId, rowKey);

            if (row != null && row.getCells(COLUMN\_FAMILY, TEMPERATURE\_COLUMN).size() > 0) {

                // Get the latest value for the temperature column

                return row.getCells(COLUMN\_FAMILY, TEMPERATURE\_COLUMN).get(0).getValue().toStringUtf8();

            } else {

                return "Temperature data not found for the given row key.";

            }

        } catch (Exception e) {

            System.err.println("Error querying Bigtable: " + e.getMessage());

            e.printStackTrace();

            return "Error occurred while querying.";

        }

    }

**Output:**

A screenshot of a computer

Description automatically generated

*Fig4.5 The command-line interface for Google Cloud Bigtable executes a query to retrieve the temperature at Vancouver. The connection to Bigtable is successfully established, and the queried temperature value is displayed as 52.*

1. **Write the method query2() that returns the highest wind speed in the month of September 2022 in Portland.**

**Code:**

public String query2(String tableId, String city) {

        try {

            String rowKeyPrefix = city + "\_2022-09";

            String highestWindSpeedRowKey = null;

            double maxWindSpeed = Double.MIN\_VALUE;

            Query query = Query.create(tableId).prefix(rowKeyPrefix);

            for (Row row : bigtableClient.readRows(query)) {

                // Get windspeed cells

                List<RowCell> windspeedCells = row.getCells(COLUMN\_FAMILY, "windspeed");

                for (RowCell cell : windspeedCells) {

                    String windSpeedValue = cell.getValue().toStringUtf8();

                    double windSpeed = Double.parseDouble(windSpeedValue);

                    if (windSpeed > maxWindSpeed) {

                        maxWindSpeed = windSpeed;

                        highestWindSpeedRowKey = row.getKey().toStringUtf8();

                    }

                }

            }

            if (highestWindSpeedRowKey != null) {

                return "Highest wind speed in September 2022 in Portland: " + maxWindSpeed + " km/h";

            } else {

                return "No data found for the specified query.";

            }

        } catch (Exception e) {

            e.printStackTrace();

            return "Error occurred while querying data.";

        }

**Output:**

A screenshot of a computer

Description automatically generated

*Fig 4.6 The command-line interface for Google Cloud Bigtable executes queries. Query 5 successfully retrieves the temperature at Vancouver as 52, while Query 6 for the highest wind speed in Portland returns no data for the specified query.*

1. **Write the method query3() that returns all the readings for SeaTac for October 2, 2022.**

**Code:**

public List<String> query3() {

        System.out.println("Executing Query 3: Retrieving readings for SeaTac on October 1, 2022, at 10:00.");

        String tableId = this.tableId;

        String rowKey = "2022-10-01\_10:00";

        List<String> readings = new ArrayList<>();

        try {

            Row row = bigtableClient.readRow(tableId, rowKey);

            if (row != null) {

                StringBuilder rowData = new StringBuilder();

                rowData.append("RowKey: ").append(row.getKey().toStringUtf8()).append(", Data: {");

                // Iterate over all cells in the row

                for (RowCell cell : row.getCells()) {

                    String columnQualifier = cell.getQualifier().toStringUtf8();

                    String value = cell.getValue().toStringUtf8();

                    rowData.append(columnQualifier).append(": ").append(value).append(", ");

                }

                if (rowData.lastIndexOf(", ") > 0) {

                    rowData.delete(rowData.lastIndexOf(", "), rowData.length());

                }

                rowData.append("}");

                readings.add(rowData.toString());

            } else {

                readings.add("No data found for row key: " + rowKey);

            }

        } catch (Exception e) {

            e.printStackTrace();

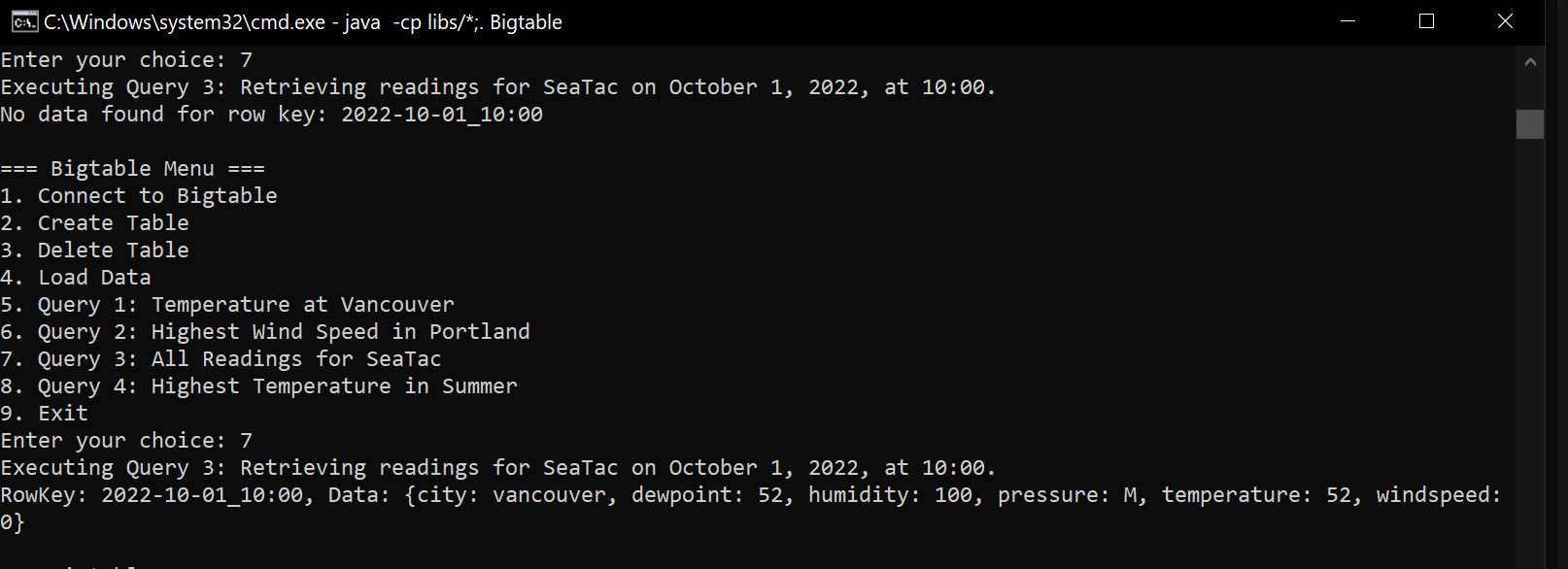
            readings.add("Error occurred while querying data.");

        }

        return readings;

    }

**Output:**



*Fig.4.7 The command-line interface for Google Cloud Bigtable executes Query 7, retrieving all readings for SeaTac on October 1, 2022, at 10:00. The data includes city (Vancouver), dew point (52), humidity (100), pressure (M), temperature (52), and wind speed (0).*

1. **Write the method query4() that returns the highest temperature at any station in the summer months of 2022 (July (7), August (8)).**

**Code:**

public int query4() {

        System.out.println("Executing Query 4: Highest Temperature in Summer 2022 (July, August).");

        String tableId = "weather";

        String[] summerMonths = {"2022-07", "2022-08"};

        int highestTemperature = Integer.MIN\_VALUE;

        try {

            for (String month : summerMonths) {

                String rowKeyPrefix = month;

                Query query = Query.create(tableId).prefix(rowKeyPrefix);

                for (Row row : bigtableClient.readRows(query)) {

                    // Iterate over all cells in the row to find the temperature column

                    for (RowCell cell : row.getCells(COLUMN\_FAMILY, "temperature")) {

                        try {

                            int temp = Integer.parseInt(cell.getValue().toStringUtf8());

                            if (temp > highestTemperature) {

                                highestTemperature = temp; // Update the highest temperature

                            }

                        } catch (NumberFormatException e) {

                            System.err.println("Invalid temperature value: " + cell.getValue().toStringUtf8());

                        }

                    }

                }

            }

            if (highestTemperature == Integer.MIN\_VALUE) {

                System.out.println("No temperature data found for summer months.");

            } else {

                System.out.println("Highest Temperature in Summer 2022: " + highestTemperature);

            }

        } catch (Exception e) {

            e.printStackTrace();

            System.out.println("Error occurred while querying data.");

        }

        return highestTemperature;

    }

**Output:**

A screen shot of a computer

Description automatically generated

*Fig.4.8 The command-line interface for Google Cloud Bigtable executes Query 8 to find the highest temperature recorded in Summer 2022 (July and August). The highest temperature retrieved is 101.*