

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 SquirrelL 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.-913511634.1664467746
- 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.

The screenshot shows the Google Cloud IAM & Admin console. The left sidebar lists various IAM and Admin tools, with 'Service Accounts' highlighted. The main content area displays the 'Create service account' wizard. Step 1, 'Service account details', is active, showing fields for 'Service account name' (g23ai2087-sa), 'Service account ID' (g23ai2087-sa), and 'Service account description' (bigdata assignment-4). The email address is g23ai2087-sa@teak-ellipse-444102-d3.iam.gserviceaccount.com. A 'CREATE AND CONTINUE' button is visible. Steps 2 and 3 are listed as optional: 'Grant this service account access to project' and 'Grant users access to this service account'. At the bottom, there are 'DONE' and 'CANCEL' buttons.

Google Cloud My First Project iam

IAM & Admin Create service account

1 Service account details

Service account name
g23ai2087-sa

Display name for this service account

Service account ID *
g23ai2087-sa

Email address: g23ai2087-sa@teak-ellipse-444102-d3.iam.gserviceaccount.com

Service account description
bigdata assignment-4

Describe what this service account will do

CREATE AND CONTINUE

2 Grant this service account access to project (optional)

3 Grant users access to this service account (optional)

DONE CANCEL

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.

My First Project ▼

Search (/) for resources, docs, products, and more

Enable access to API

1 Confirm project

2 Enable API

You are going to make changes to project 'teak-ellipse-444102-d3'. If this is not the project you intended to use, you can select or create a different project using the project selector above.

[NEXT](#)

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.

My First Project ▼

Search (/) for resources, docs, products, and more

Enable access to API

✓ Confirm project

2 Enable API

You are about to enable 'BigQuery API'.

[ENABLE](#)

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.

[←](#) Create an instance

A Bigtable instance is a container for your clusters. [Learn more](#)

1 Name your instance

Instance name *

g23ai2087-instance

For display purposes only

Instance ID *

g23ai2087-instance

ID is permanent

CONTINUE

2 Select your storage type

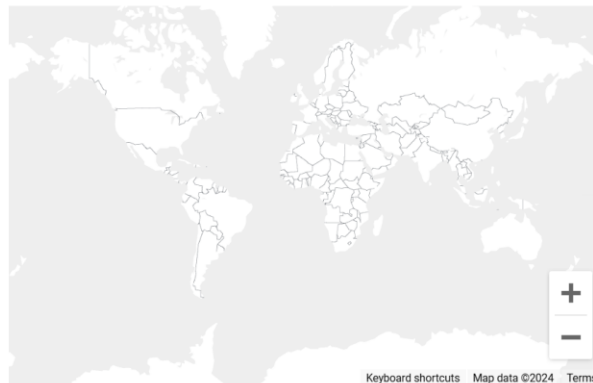
3 Configure your first cluster

SHOW ADVANCED OPTIONS

CREATE

CANCEL

Summary



Pricing estimate

\$468 per month

That's about \$0.65 an hour with 0 GB stored.

SHOW DETAILS

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.

Google Cloud

My First Project

big

X

Search

Navigation menu () instance

A Bigtable instance is a container for your clusters. [Learn more](#)

✓ Name your instance

2 Select your storage type

3 Configure your first cluster

SHOW ADVANCED OPTIONS

CREATE CANCEL

Choice is permanent. Applies to all clusters. Affects cost.

SSD

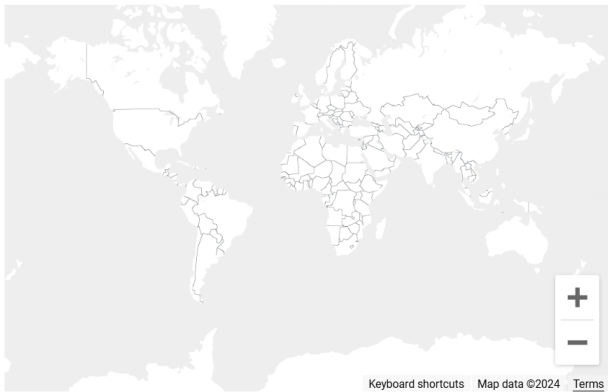
Lower latency and more rows read per second. Typically used for real-time serving use cases, such as ad serving and mobile app recommendations.

HDD

Higher latency for random reads. Good performance on scans and typically used for batch analytics, such as machine learning or data mining.

CONTINUE

Summary



Pricing estimate

\$468 per month

That's about \$0.65 an hour with 0 GB stored.

SHOW DETAILS

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.

Google Cloud

My First Project

big

X

Search

Navigation menu () instance

3 Configure your first cluster

A cluster handles application requests for an instance. It contains nodes which determine your cluster's performance and storage limit.

Additional clusters can be added at any time.

Select a cluster ID

ID is permanent

Cluster ID *

g23ai2087-instance-c1

Select a location

Choice is permanent. Determines where cluster data is stored. To reduce latency and increase throughput, store your data near the services that need it. [Learn more](#)

Region *

asia-east1 (Taiwan)

Zone *

Any

Configure Nodes


Nodes are compute resources that Bigtable uses to manage your data and perform maintenance tasks. Adding nodes helps a cluster handle larger workloads.

Enable 2x node scaling

Choice is permanent. Select to improve cluster performance by reducing boundaries between nodes. If selected, clusters will scale in increments of 2 nodes. This decision will only affect this cluster; clusters in an instance can have different node scaling factors.

Choose node scaling mode

Summary



Pricing estimate

\$468 per month

That's about \$0.65 an hour with 0 GB stored.

SHOW DETAILS

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.

Choose node scaling mode

Nodes are compute resources that Bigtable uses to manage your data and perform maintenance tasks. Adding nodes helps a cluster handle larger workloads.

Scaling mode and configurations can be changed at any time.

☒ **Manual allocation**

Set your node count for fixed costs and compute resources.

For better instance performance, keep your CPU utilization under the recommended threshold for your [app profile routing policy](#). [Contact us](#) if you need to increase your quota.

Quantity * Nodes

☐ **Autoscaling**

Let Bigtable automatically add and remove nodes.

▼ SHOW ENCRYPTION OPTIONS

▼ SHOW ADVANCED OPTIONS

CREATE

CANCEL

**Pricing estimate**

\$468 per month

That's about \$0.65 an hour with 0 GB stored.

▼ SHOW DETAILS

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.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:



```

C:\Windows\system32\cmd.exe - java -cp libs/*;. Bigtable
=== Bigtable Menu ===
1. Connect to Bigtable
2. Create Table
3. Delete Table
4. Load Data
5. Query 1: Temperature at Vancouver
6. Query 2: Highest Wind Speed in Portland
7. Query 3: All Readings for SeaTac
8. Query 4: Highest Temperature in Summer
9. Exit
Enter your choice: 1
Establishing connection to Bigtable...
Table 'weather' already exists.
Connection to Bigtable established successfully.

```

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.

2. 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

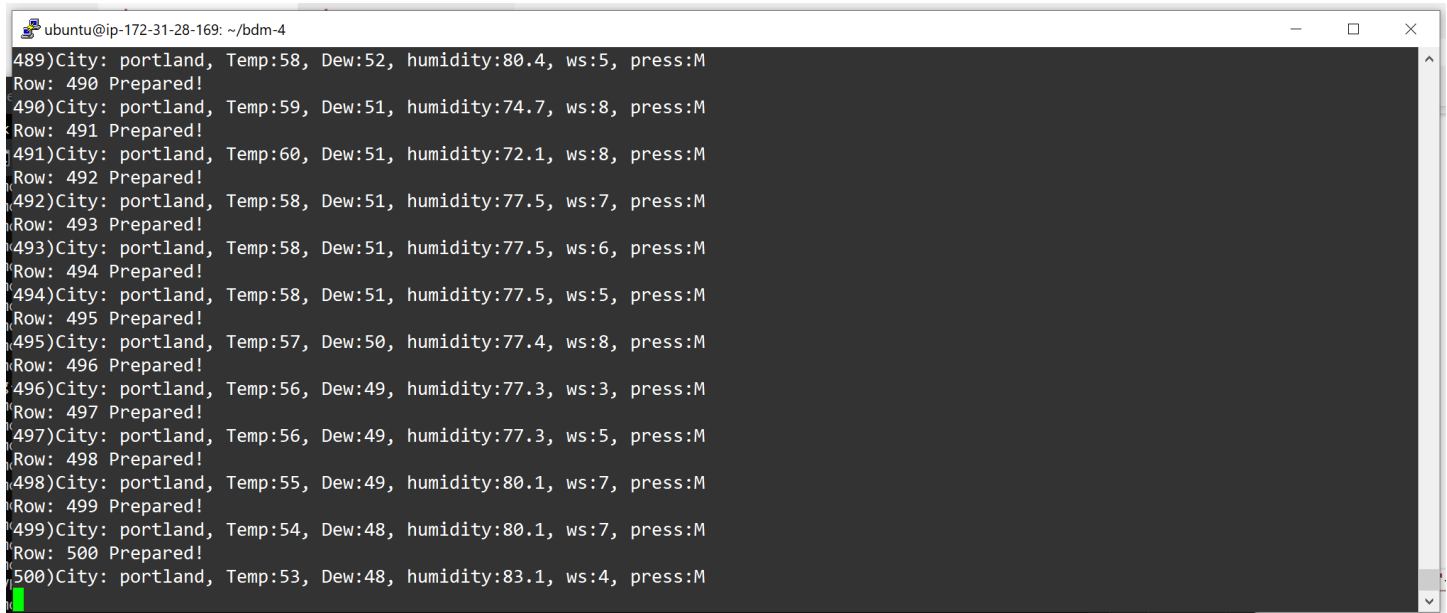
```

=== Bigtable Menu ===
1. Connect to Bigtable
2. Create Table
3. Delete Table
4. Load Data
5. Query 1: Temperature at Vancouver
6. Query 2: Highest Wind Speed in Portland
7. Query 3: All Readings for SeaTac
8. Query 4: Highest Temperature in Summer
9. Exit
Enter your choice: 4
Loading data into table...
Reading file: data/portland.csv
1)City: portland, Temp:Temperature, Dew:Dewpoint, humidity:Relhum, ws:Speed, press:Gust
Row: 2 Prepared!
2)City: portland, Temp:64, Dew:53, humidity:67.4, ws:4, press:M
Row: 3 Prepared!
3)City: portland, Temp:67, Dew:53, humidity:60.7, ws:3, press:M
Row: 4 Prepared!
4)City: portland, Temp:68, Dew:52, humidity:56.5, ws:3, press:M
Row: 5 Prepared!

```


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



```

ubuntu@ip-172-31-28-169: ~/bdm-4
489)City: portland, Temp:58, Dew:52, humidity:80.4, ws:5, press:M
Row: 490 Prepared!
490)City: portland, Temp:59, Dew:51, humidity:74.7, ws:8, press:M
Row: 491 Prepared!
491)City: portland, Temp:60, Dew:51, humidity:72.1, ws:8, press:M
Row: 492 Prepared!
492)City: portland, Temp:58, Dew:51, humidity:77.5, ws:7, press:M
Row: 493 Prepared!
493)City: portland, Temp:58, Dew:51, humidity:77.5, ws:6, press:M
Row: 494 Prepared!
494)City: portland, Temp:58, Dew:51, humidity:77.5, ws:5, press:M
Row: 495 Prepared!
495)City: portland, Temp:57, Dew:50, humidity:77.4, ws:8, press:M
Row: 496 Prepared!
496)City: portland, Temp:56, Dew:49, humidity:77.3, ws:3, press:M
Row: 497 Prepared!
497)City: portland, Temp:56, Dew:49, humidity:77.3, ws:5, press:M
Row: 498 Prepared!
498)City: portland, Temp:55, Dew:49, humidity:80.1, ws:7, press:M
Row: 499 Prepared!
499)City: portland, Temp:54, Dew:48, humidity:80.1, ws:7, press:M
Row: 500 Prepared!
500)City: portland, Temp:53, Dew:48, humidity:83.1, ws:4, press:M

```

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

```

ubuntu@ip-172-31-28-169: ~/bdm-4
3489)City: seatac, Temp:41, Dew:40, humidity:96.2, ws:0, press:M
Row: 13879 Prepared!
3490)City: seatac, Temp:41, Dew:41, humidity:100, ws:3, press:M
Row: 13880 Prepared!
3491)City: seatac, Temp:41, Dew:41, humidity:100, ws:0, press:M
Row: 13881 Prepared!
3492)City: seatac, Temp:41, Dew:39, humidity:92.5, ws:6, press:M
Row: 13882 Prepared!
3493)City: seatac, Temp:41, Dew:40, humidity:96.2, ws:6, press:M
Row: 13883 Prepared!
3494)City: seatac, Temp:40, Dew:39, humidity:96.2, ws:0, press:M
Row: 13884 Prepared!
3495)City: seatac, Temp:39, Dew:39, humidity:100, ws:0, press:M
Row: 13885 Prepared!
3496)City: seatac, Temp:40, Dew:39, humidity:96.2, ws:0, press:M
Row: 13886 Prepared!
3497)City: seatac, Temp:40, Dew:39, humidity:96.2, ws:3, press:M
Row: 13887 Prepared!
3498)City: seatac, Temp:40, Dew:39, humidity:96.2, ws:0, press:M
Row: 13888 Prepared!
3499)City: seatac, Temp:40, Dew:40, humidity:100, ws:0, press:M
Row: 13889 Prepared!
3500)City: seatac, Temp:40, Dew:40, humidity:100, ws:0, press:M

```

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.

3. 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:

```

2 C:\Windows\system32\cmd.exe - java -cp libs\*. Bigtable
9. Exit
Enter your choice: 1
Establishing connection to Bigtable...
Table 'weather' already exists.
Connection to Bigtable established successfully.

=== Bigtable Menu ===
1. Connect to Bigtable
2. Create Table
3. Delete Table
4. Load Data
5. Query 1: Temperature at Vancouver
6. Query 2: Highest Wind Speed in Portland
7. Query 3: All Readings for SeaTac
8. Query 4: Highest Temperature in Summer
9. Exit
Enter your choice: 5
Temperature: 52

```

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.

4. 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:

```

C:\Windows\system32\cmd.exe - java -cp libs/*;. Bigtable
7. Query 3: All Readings for SeaTac
8. Query 4: Highest Temperature in Summer
9. Exit
Enter your choice: 5
Temperature: 52

=== Bigtable Menu ===
1. Connect to Bigtable
2. Create Table
3. Delete Table
4. Load Data
5. Query 1: Temperature at Vancouver
6. Query 2: Highest Wind Speed in Portland
7. Query 3: All Readings for SeaTac
8. Query 4: Highest Temperature in Summer
9. Exit
Enter your choice: 6
Highest Wind Speed: No data found for the specified query.

```

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.

5. 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:

```

=== Bigtable Menu ===
1. Connect to Bigtable
2. Create Table
3. Delete Table
4. Load Data
5. Query 1: Temperature at Vancouver
6. Query 2: Highest Wind Speed in Portland
7. Query 3: All Readings for SeaTac
8. Query 4: Highest Temperature in Summer
9. Exit
Enter your choice: 7
Executing Query 3: Retrieving readings for SeaTac on October 1, 2022, at 10:00.
RowKey: 2022-10-01_10:00, Data: {city: vancouver, dewpoint: 52, humidity: 100, pressure: M, temperature: 52, windspeed:
0}

```

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).

6. 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:


```

C:\Windows\system32\cmd.exe - java -cp libs/*;. Bigtable
Enter your choice: 1
Establishing connection to Bigtable...
Table 'weather' already exists.
Connection to Bigtable established successfully.

=== Bigtable Menu ===
1. Connect to Bigtable
2. Create Table
3. Delete Table
4. Load Data
5. Query 1: Temperature at Vancouver
6. Query 2: Highest Wind Speed in Portland
7. Query 3: All Readings for SeaTac
8. Query 4: Highest Temperature in Summer
9. Exit
Enter your choice: 8
Executing Query 4: Highest Temperature in Summer 2022 (July, August).
Highest Temperature in Summer 2022: 101
Highest Temperature in Summer: 101

```

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.