Assignment-4

Google Big Table: 50 points

> **Submit a PDF with code listing, and screenshots showing outputs of insert(), delete(), and the queries. Screenshots should be uniquely distinguishable for each submission. Be careful of plagiarism from online sources/peers.**

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.-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 to install the Google Cloud CLI then run the command:

gcloud auth application-default login.

In the given starter code, fill the functions marked as TODO

1. 10 mark - Write the method connect() to create a connection. Create a Bigtable data client and admin client. See HelloWorld.java for starter code.

2. 10 mark - Write the method createTable() to create a table to store the sensor data.

3. 5 marks - Write the method load() to load the sensor data into the database. The data files are in the data folder.

4. 10 marks - Write the method query1() that returns the temperature at Vancouver on 2022-10-01 at 10 a.m.

5. 5 marks - Write the method query2() that returns the highest wind speed in the month of September 2022 in Portland.

6. 5 marks - Write the method query3() that returns all the readings for SeaTac for October 2, 2022.

7. 5 marks - Write the method query4() that returns the highest temperature at any station in the summer months of 2022 (July (7), August (8)).

Starter Code:

```
import com.google.api.gax.rpc.NotFoundException;
import com.google.api.gax.rpc.ServerStream;
import com.google.cloud.bigtable.admin.v2.BigtableTableAdminClient;
import com.google.cloud.bigtable.admin.v2.BigtableTableAdminSettings;
import com.google.cloud.bigtable.admin.v2.models.CreateTableRequest;
import com.google.cloud.bigtable.data.v2.BigtableDataClient;
import com.google.cloud.bigtable.data.v2.BigtableDataSettings;
import com.google.cloud.bigtable.data.v2.models.BulkMutation;
import com.google.cloud.bigtable.data.v2.models.Mutation;
import com.google.cloud.bigtable.data.v2.models.Query;
import com.google.cloud.bigtable.data.v2.models.Row;
import com.google.cloud.bigtable.data.v2.models.RowCell;
import com.google.cloud.bigtable.data.v2.models.RowMutation;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

/*
 * Use Google Bigtable to store and analyze sensor data.
 */
public class Bigtable {
        // TODO: Fill in information for your database
        public final String projectId = "iitjdb";
        public final String instanceId = "ail7560";
```

```
public final String COLUMN_FAMILY = "sensor";
public final String tableId = "weather"; // TODO: Must change table name if sharing my database
public BigtableDataClient dataClient;
public BigtableTableAdminClient adminClient;

public static void main(String[] args) throws Exception {
	Bigtable testbt = new Bigtable();
	testbt.run();
}

public void connect() throws IOException {
	// TODO: Write code to create a data client and admin client to connect to
	// Google Bigtable
	// See sample code in HelloWorld.java for help

}

public void run() throws Exception {
	connect();

	// TODO: Comment or uncomment these as you proceed. Once load data, comment them
	// out.
	deleteTable();
	createTable();
	loadData();

	int temp = query1();
	System.out.println("Temperature: " + temp);

	int windspeed = query2();
	System.out.println("Windspeed: " + windspeed);

	ArrayList<Object[]> data = query3();

	StringBuffer buf = new StringBuffer();

	for (int i = 0; i < data.size(); i++) {
		Object[] vals = data.get(i);

		for (int j = 0; j < vals.length; j++)
			buf.append(vals[j].toString() + " ");
		buf.append("\n");
	}
	System.out.println(buf.toString());

	temp = query4();
	System.out.println("Temperature: " + temp);

	close();
}
```

```
/**
 * Close data and admin clients
 */
public void close() {
	dataClient.close();
	adminClient.close();
}

public void createTable() {
	// TODO: Create a table to store sensor data.

}

/**
 * Loads data into database.
 * Data is in CSV files. Note that must convert to hourly data.
 * Take the first reading in a hour and ignore any others.
 */
public void loadData() throws Exception {
	String path = "bin/data/";
	// TODO: Load data from CSV files into sensor table
	// Note: There are multiple different ways that you can decide on how to
	// organize this data into columns
	try {
		// SeaTac station id is SEA
		System.out.println("Load data for SeaTac");

		// Vancouver station id is YVR
		System.out.println("Loading data for Vancouver");

		// Portland station id is PDX
		System.out.println("Loading data for Portland");

	} catch (Exception e) {
		throw new Exception(e);
	}
}

/**
 * Query returns the temperature at Vancouver on 2022-10-01 at 10 a.m.
 *
 * @return
 *       ResultSet
 * @throws SQLException
 *                 if an error occurs
 */
public int query1() throws Exception {
	// TODO: Write query #1
	System.out.println("Executing query #1.");
```

```
        return 0;
}

/**
 * Query returns the highest wind speed in the month of September 2022 in
 * Portland.
 *
 * @return
 *      ResultSet
 * @throws SQLException
 *              if an error occurs
 */
public int query2() throws Exception {
        // TODO: Write query #2
        System.out.println("Executing query #2.");

        int maxWindSpeed = 0;

        return maxWindSpeed;
}

/**
 * Query returns all the readings for SeaTac for October 2, 2022. Return as an
 * ArrayList of objects arrays.
 * Each object array should have fields: date (string), hour (string),
 * temperature (int), dewpoint (int), humidity (string), windspeed (string),
 * pressure (string)
 *
 * @return
 *      ResultSet
 * @throws SQLException
 *              if an error occurs
 */
public ArrayList<Object[]> query3() throws Exception {
        // TODO: Write query #3
        System.out.println("Executing query #3.");

        ArrayList<Object[]> data = new ArrayList<Object[]>();

        return data;
}

/**
 * Query returns the highest temperature at any station in the summer months of
 * 2022 (July (7), August (8)).
 *
 * @return
 *      ResultSet
 * @throws SQLException
```

```
	 *                  if an error occurs
	 */
	public int query4() throws Exception {
		// TODO: Write query #4
		// Try to avoid reading the entire table. Consider using readRowRanges()
		// instead.
		System.out.println("Executing query #4.");

		int maxTemp = -100;

		return maxTemp;
	}

	/**
	 * Create your own query and test case demonstrating some different.
	 *
	 * @return
	 *        ResultSet
	 * @throws SQLException
	 *                  if an error occurs
	 */
	public int query5() throws Exception {
		// TODO: Write your own unique query and test case
		System.out.println("Executing query #5.");
		return 0;
	}

	/**
	 * Delete the table from Bigtable.
	 */
	public void deleteTable() {
		System.out.println("\nDeleting table: " + tableId);
		try {
			adminClient.deleteTable(tableId);
			System.out.printf("Table %s deleted successfully%n", tableId);
		} catch (NotFoundException e) {
			System.err.println("Failed to delete a non-existent table: " + e.getMessage());
		}
	}
}
```