**Required Viewing**

Before starting this assignment, you must watch two videos:

1. **Week 6 Fundamentals Lecture Video**
   * Explains HBase architecture, the importance of row key design, and how HBase fits into the NoSQL ecosystem.
   * Provides the background needed to understand what you are learning.
   * Link: <https://youtu.be/DT-kisp9Zkw>
2. **Week 6 Assignment Walkthrough Video**
   * Demonstrates step by step how to complete the tasks, including commands and expected outputs.
   * It is not enough to simply run commands. You must verify that your commands execute correctly. Incorrect or incomplete results will lose points.

Watching both videos is mandatory. The **fundamentals video** explains what you are learning and why it matters, while the **walkthrough video** shows how to complete the assignment.

**Submission Guidelines**

* Submit your work as a **single Word or PDF document** (no raw screenshots or multiple files).
* Include the following in your submission:
  + Screenshots of each required step.
  + A short explanation for each screenshot:
    - The command/action you ran.
    - What the output shows.
    - Whether the result matched your expectation.
* Organize your work in the **same order as the assignment guide** so it is easy to follow.
* This is a **master’s level course** – professionalism and clarity are expected. Well-structured submissions demonstrate your ability to communicate technical work effectively.

**Week 6 Assignment – Objectives and Points**

* **Objective 1 – Conceptual Foundations (HBase Overview)**: 8 pts
* **Objective 2 – HBase Table Creation and Management**: 10 pts
* **Objective 3 – HBase Data Manipulation (Insert and Query)**: 12 pts
* **Objective 4 – Composite Row Keys with HBase**: 12 pts
* **Objective 5 – Data Generation in HBase**: 10 pts
* **Objective 6 – Advanced Data Manipulation in HBase**: 36 pts

**Total: 88 points**

# Week 6 Assignment: Hands-on with HBase

In this assignment, you will gain hands-on experience with **Apache HBase**, a distributed, scalable, and NoSQL database designed for large-scale data storage. HBase allows you to manage unstructured and semi-structured data, and it excels at handling sparse data with variable schema. You will explore core functionalities such as table creation, data manipulation, and querying in HBase’s interactive shell. You’ll also generate, modify, and retrieve data using HBase’s powerful features like composite row keys and efficient scanning mechanisms.

By the end of this assignment, you will:

* Understand how to create and manage HBase tables.
* Gain experience with basic and advanced data manipulation using HBase commands.
* Explore the concept of composite row keys for optimized data access.
* Generate and modify large datasets programmatically, making use of HBase’s distributed storage.

# Objective 1 – Conceptual Foundations (8 points)

Before beginning the assignment, watch the instructor-led fundamentals video, which introduces and explains the key concepts for this week:  
<https://youtu.be/DT-kisp9Zkw>

**Deliverable:** Write a 3–4 paragraph summary that demonstrates your understanding of the fundamentals video. Your write-up should explain the main ideas in your own words, highlight why these concepts are important, and connect them to the technologies used in this assignment.

# Objective 2 – HBase Table Creation and Management (10 points)

#### **1. Environment Initialization**

* Start by navigating to the required directory and initiating the Docker containers:

cd dsc650-infra/bellevue-bigdata/hadoop-hive-spark-hbase

docker-compose up -d

* Access the master container:

docker-compose exec master bash

! STOP! Wait for HBase to load. You need to be able to successfully access the HBase UI before you proceed.

<http://localhost:16010>

#### **2. Introduction to HBase**

You will begin by accessing the **HBase interactive shell**, where you can issue commands to create and manage HBase tables. The HBase shell provides a command-line interface for interacting with the HBase database.

* Enter the HBase interactive shell:

hbase shell

#### **3. Table Creation and Management**

In this section, you will create an HBase table named students, which will store information about students. This introduces you to the basics of HBase table creation, where each table contains **column families**, such as details in this case.

**Exercise 1:** Create a table named ‘students’ with a column family ‘details’.

create 'students', 'details'

**Deliverable 1:** Screenshot of the table creation command and its output, plus 1–2 sentences explaining what the output confirms.

**Exercise 2:** Verify that the table has been created.

list

**Deliverable 2:** Screenshot of the tables listed in HBase, plus a short explanation of what this proves (that your table was created successfully).

# Objective 3 – HBase Data Manipulation (12 points)

#### **1. Data Manipulation in HBase**

You will now add data to the students table using the put command, which inserts rows into an HBase table. This introduces the concept of storing data in key-value pairs, with each student being identified by a unique row key (their ID).

**Exercise 3:** Add data to the ‘students’ table. Let’s assume each student has a unique ID, a first name, and a last name.

put 'students', '1', 'details:firstName', 'John'  
put 'students', '1', 'details:lastName', 'Doe'

**Deliverable 1:** Screenshot of the commands used to add data and their outputs, plus a short explanation of what was inserted.

**Exercise 4:** Query the data from the ‘students’ table to retrieve the details of the student with ID ‘1’.

get 'students', '1'

**Deliverable 2:** Screenshot of the query retrieving student ID “1” and its output, plus 1–2 sentences explaining what the result shows.

# Objective 4 – Composite Row Keys with HBase (12 points)

#### **1. Advanced HBase Features: Composite Row Key**

In this section, you’ll create a new table named orders that uses **composite row keys**. Composite row keys allow you to combine multiple fields (like customer ID and order date) to uniquely identify each row, optimizing your ability to query data efficiently.

**Exercise 5:** Create a table named ‘orders’ to store data about customer orders. Assume each order is uniquely identified by a composite key formed by combining the customer ID and order date (in the format YYYYMMDD).

create 'orders', 'orderDetails'

**Deliverable 1:** Screenshot of the table creation command and its output, plus a short explanation of why a composite row key is useful.

**Exercise 6:** Add sample data to the ‘orders’ table using the composite key:

put 'orders', '101:20230806', 'orderDetails:item', 'Laptop'  
put 'orders', '102:20230806', 'orderDetails:item', 'Smartphone'

**Deliverable 2:** Screenshot of sample data inserted with composite keys, plus 1–2 sentences describing what the keys represent.

**Exercise 7:** Query the ‘orders’ table to retrieve details of all orders placed by the customer with ID ‘101’.

scan 'orders', {STARTROW => '101:', ENDROW => '101:~'}

This command will scan rows starting from ‘101:’ to before ‘101:~’ (tilde ‘~’ is the next ASCII character after colon ‘:’).

**Deliverable 3:** Screenshot of the scan query and its output, plus a short explanation of how the scan filtered results by customer ID.

# Objective 5 – Data Generation in HBase (10 points)

#### **1. Data Generation for HBase**

You will now automate the insertion of data into the students table using a Ruby loop. This demonstrates how HBase can handle large amounts of data efficiently, and how programmatic data generation can be applied.

**Exercise 8:** Generate random data for the ‘students’ table.

|  |
| --- |
| (2..100).each do |i|  first\_name = "Student#{i}"  last\_name = "LastName#{i}"  put 'students', "#{i}", 'details:firstName', first\_name  put 'students', "#{i}", 'details:lastName', last\_name  end |

**Deliverable 1:** Screenshot of the Ruby loop used to generate student data, plus a short explanation of what the loop is doing.

**Exercise 9:** Scan the ‘students’ table to verify data insertion.

scan 'students'

**Deliverable 2:** Screenshot of the scan verifying data insertion, plus 1–2 sentences describing what the output confirms.

# Objective 6 – Advanced Data Manipulation in HBase (36 points)

**Exercise 10: HBase Data Manipulation**

In this final section, you will modify the data in the students table and practice using various data manipulation techniques, including updating, adding columns, and bulk deletion.

**Tasks:**

1. **Update First Names:**
   * For students with IDs from 2 to 50, change the first name prefix from Student to Scholar. For instance, Student3 should become Scholar3.

**Deliverable 1:** Screenshot of the update commands (changing first names for IDs 2–50), plus a short explanation of how the update was applied.

1. **Add a Middle Name:**
   * For students with IDs from 51 to 75, add a middle name column under the details column family. The middle name should follow the pattern MidName#{i}.

**Deliverable 2:** Screenshot of the commands adding middle names for IDs 51–75, plus an explanation of what was added and why.

1. **Modify Last Names:**
   * For students with IDs from 76 to 100, append \_Modified to the last name. So, LastName76 should be updated to LastName76\_Modified.

**Deliverable 3:** Screenshot of the commands modifying last names for IDs 76–100, plus 1–2 sentences describing the change.

1. **Bulk Delete:**
   * Delete all the details for students with IDs from 90 to 100.

**Deliverable 4:** Screenshot of the bulk delete for IDs 90–100, plus a short explanation of what data was removed.

1. **Data Retrieval:**
   * After all modifications, retrieve and display the details for students with IDs 40, 60, 80, and 90 to verify changes.

**Deliverable 5:** Screenshot of the retrieval queries for IDs 40, 60, 80, and 90, plus a short explanation confirming that all modifications were applied correctly.

**Deliverable 6:** Screenshot of the final scan of the table, plus 1–2 sentences summarizing the overall results of your advanced manipulations.

## Shutting Down

Ensure all Docker containers are turned off with docker-compose down for each directory. If you’re using google cloud, please shut down your virtual machine to preserve cloud costs.