|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| Sunday |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## Team meeting - Connect with King & Hammed:

### Points to discuss

\* Living in Ontario / GTA ?

\* Possibility for Zoom meeting

\* WhatsApp group

\* Any other alternative

\* Background for each team member

\* Areas each team member feel comfortable with

\* Plan:

- Strategy/way of working : divide the tasks or work in parallel ?

- Frequency of connecting/meeting

- Create plan for how to approach the problem

- Next Day/time to connect

### Plan:

\* Monday

\* Tuesday

\* Wednesday

\* Thursday

\* Friday

\* Saturday

\* Sunday

### The assignment

- Review the PDF for the assignment

- Skills Covered: Apache NiFi, Hive/Trino, Spark, Kafka, Cassandra, and Superset

### Other material and areas to improve

1- NiFi, Kafka, Spark streaming, and Cassandra

Resources:

\* Review the professor’s correct solution for the NiFi HW & do it

===>\* Review the professor comments for the past assignment

\* Review the Workshops/Tutorials that were no covered

\* Review the Workshops that were not referencing any of the Tutorials.

2- HDFS, Spark Trino & Superset

Resources:

\* Review the professor’s correct solution for the first two Homeworks

\* Review the Workshops/Tutorials that were no covered

\* Review the Workshops that were not referencing any of the Tutrials

#### Other tasks:

~~\* Check the site for the API based on their email for the quota?~~

* Checked and already sent an email to the support team of AviationStack.

~~\* Send email to Khaled for the typo in HW4~~

~~\*\*\* aviationstack API allows you to get access to data quickly so you can focus on building the features we need. The data is published through a REST API that returns a JSON-encoded response which contains market stocks data (chart) for a particular stock(s) in a date range.~~

* Already reached out to Kahled and he confirmed that there is a typo/mistake – to be corrected.

## Tasks

|  |  |
| --- | --- |
| Read / Understand the tasks of the assignment |  |
| Divide into multiple subtasks |  |
| Review Previous NiFi HW – Tannir’s Solution |  |
| Review Previous NiFi HW – Tannir’s Comments |  |
| Cassandra |  |
| Plan for each task |  |
| Team Meetings |  |
| Integrate solution |  |
| Workshops - |  |
| MySql |  |
| Other |  |

## Problem Statement and Requirements:

|  |  |
| --- | --- |
| **Requirement** | **Simplified Version** |
| **READING JSON FLIGHTS DATA (5 pts)**   * Build an Apache NiFi flow to ingest flights **limited** to **YUL** using the REST API. * Implement pagination using a loop to increment the offset (do not clone InvokeHTTP processors). Limit (in the loop) each **API request to 100 rows**. The free plan is limited to **100 rows** per call. * Avoid hardcoding sensitive or variable information. * Use saved JSON files during development and testing. | **Reading JSON Flights Data (5 points)**   1. **Create a NiFi Flow to Read Flight Data**    * Set up a NiFi flow that gets flight data from the REST API.    * Filter the data to only include flights related to **YUL airport**. 2. **Handle Pagination with a Loop**    * Use a loop in NiFi to **increase the offset** for each API call.    * **Do not duplicate or clone InvokeHTTP processors**—use the loop instead.    * Each API request should be limited to **100 results per call**, as required by the free plan. 3. **Avoid Hardcoding**    * Don’t hardcode sensitive data (like API keys) or values that might change.    * Use variables, parameters, or controller services to store this information securely and flexibly. 4. **Use Sample Data While Testing**   During development and testing, use **saved JSON files** instead of calling the live API every time. |
| **SPLITING/EVALUATING FLIGHTS DATA (4 pts)**   * Use Jolt/JSLT to extract **Live** data and send it to a Kafka topic. * Save the complete dataset to HDFS as compressed Parquet/Snappy files. * Ensure filenames follow: flights-<timestamp>.parquet and partition the data by year/month/day. | **✈️ Splitting and Processing Flights Data**   1. **Extract Live Flight Info and Send to Kafka**    * Use **Jolt or JSLT** to pull out only the **live flight data** (e.g., currently in the air).    * Send this live data to a **Kafka topic**. 2. **Save All Data to HDFS**    * Store the full flights dataset in **HDFS**.    * Save it in **Parquet format** with **Snappy compression**. 3. **File Naming and Organization**    * Name each file like this: flights-<timestamp>.parquet (e.g., flights-20250720.parquet).    * **Organize (partition)** the files by **year/month/day** for better structure and performance. |
| **STORING HISTORICAL DATA (3 pts)**  Use Jolt/JSLT to extract the **Arrival** section. Store this information in a MySQL table as historical flight data. | **✈️ Storing Historical Data (3 points)**   * Use **Jolt** or **JSLT** in NiFi to extract only the **"arrival"** part from the flight data. * Save this **arrival information** into a **MySQL table**. * This table will be used to keep a record of **past flight arrivals** (historical data). |
| **DATA ANALYSIS (6 pts)**  Create a Hive/Trino table to query the Parquet data. Provide:   * First 5 rows of data. * Count of flights per airline for the current date. * First and last scheduled flights for the current date. | **📊 Data Analysis (6 points)**   * Create a **Hive or Trino table** to read the **Parquet flight data**. * Then, run the following queries:   1. **Show the first 5 rows** from the table.   2. **Count how many flights** each **airline** has for **today’s date**.   3. Find the **first** and **last scheduled flights** for **today’s date**. |
| **CONSUMING REAL TIME DATA (9 pts)**   * Use Spark Structured Streaming to read the Kafka topic and output results to Cassandra. * Use the Haversine Formula (provided) to calculate distance from each plane to YUL. * Display results on the console (in Zeppelin) and write output to a Cassandra table.   **Hint**: You might need at least the following columns in Cassandra. (You can add more columns if you need to)  departure airport, airport iata, airline name, flight iata, arrival estimated. | **🔄 Consuming Real-Time Data (9 points)**   * Use **Spark Structured Streaming** to read live flight data from a **Kafka topic**. * Send the processed results to a **Cassandra** database. * Use the **Haversine Formula** (provided) to calculate how far each plane is from **YUL airport**. * Show the results on the **Zeppelin console** and also **save them into a Cassandra table**. * Your output should include:   + The **number of flights per airline** for **today**.   + The **first and last scheduled flights** for **today**.   **Hint**: You might need at least the following columns in Cassandra. (You can add more columns if you need to)  departure airport, airport iata, airline name, flight iata, arrival estimated. |
| **DATA VISUALIZATION (3 pts)**  Build a Superset dashboard with at least three visualizations. Choose any relevant metrics or KPIs based on the ingested data. | |
| **Development Guidelines**   * Use GetHTTP/InvokeHTTP in NiFi sparingly to avoid consuming your entire API quota in one call. * Use process groups to organize your dataflow and avoid hardcoding. * Save test JSON responses from browser calls and build workflows offline when possible. * A sample dataset is provided to allow team members to work concurrently.   Note: This case study focuses only on the data ingestion process and not on the flights data analysis. | |

**Meta Information**

**YUL Airport Coordinates:** Latitude: 45.4690 Longitude: -73.7378

**Cassandra Configuration**: - Host: localhost:9042 - Keyspace: training - Auth: None - Consistency Level: LOCAL\_QUORUM

**Troubleshooting**

Please use carefully the API and look at the requests limit. Otherwise, you will be blocked as you are using the Free Plan.

In case you are blocked, and you need the data to move forward, I can provide you with some data but with **5 pts penalty**.