# Advanced Databases - MotoGP CouchDB Project

Submitted By: Darragh Elbel T00211193

Computing with Software Development

Date Submitted: dd/mm/yyyy

## Summary:

## Content:

## Introduction:

## Sections:

### Dataset

For my dataset, I chose “Moto GP World Championship(1949-2022)” available at the following link <https://www.kaggle.com/datasets/alrizacelk/moto-gp-world-championship19492022?select=riders-finishing-positions.csv> . The original data consisted of 6 separate CSV files.

The first step I took towards importing this data into CouchDB was to process the original CSV file into JSON, to do this I looped through all the files in my CSV directory and parsed them into JSON using the ‘csv-parser’ npm module.

fs.readdir(inputFilePath, (err, files) => {

  files.forEach((file) => {

    let fileName = file.split(".")[0];

    let jsonArray = [];

    fs.createReadStream(`${inputFilePath}/${file}`)

      .pipe(csv())

      .on("data", (row) => {

        jsonArray.push(row);

      })

      .on("end", () => {

        const jsonContent = JSON.stringify(jsonArray, null, 2);

        fs.writeFile(

          `${outPutFilePath}/${fileName}.json`,

          jsonContent,

          (err) => {

            if (err) {

              console.error(`Error writing ${fileName}.json:`, err);

            } else {

              console.log(`${fileName}.json has been saved`);

            }

          }

        );

      });

  });

});

After reviewing the parsed data I decided it would be best to focus on further converting one of the six CSV files into JSON, ‘riders-finishing-positions.json’. The original CSV file contained a row for each rider, a data shape that would not be optimized for a file based database. To remedy I altered the data shape to have a unique object for each country code, and interested an array of riders into this object.

{

    "Country": "IT",

    "Riders": [

      {

        "name": "Giacomo Agostini",

        "Victories": "30",

        "NumberofSecond": "24",

        "NumberofThird": "19",

        "Numberof4th": "16",

        "Numberof5th": "16",

        "Numberof6th": "15"

      }

    ]

  }

And the code that was used to do this conversion.

fs.readFile("./riders-finishing-positions.json", "utf8", (err, data) => {

  if (err) {

    console.error("Error reading input file:", err);

    return;

  }

  const inputData = JSON.parse(data);

  // Process the data (similar to the previous code)

  const result = inputData.reduce((acc, entry) => {

    const country = entry.Country;

    const riderInfo = {

      name: entry.Rider,

      Victories: entry.Victories,

      NumberofSecond: entry.NumberofSecond,

      NumberofThird: entry.NumberofThird,

      Numberof4th: entry.Numberof4th,

      Numberof5th: entry.Numberof5th,

      Numberof6th: entry.Numberof6th,

    };

    if (!acc[country]) {

      acc[country] = {

        Country: country,

        Riders: [],

      };

    }

    acc[country].Riders.push(riderInfo);

    return acc;

  }, {});

  // Convert the result object into an array

  const finalResult = Object.values(result);

  // Write the result to a new file

  const outputJson = JSON.stringify(finalResult, null, 2);

  fs.writeFile("output.json", outputJson, "utf8", (err) => {

    if (err) {

      console.error("Error writing output file:", err);

      return;

    }

    console.log("Result written to output.json");

  });

});

I then split each country into its own JSON document before importing into my CouchDB database.

fs.readFile("output.json", "utf8", (err, data) => {

  const outputData = JSON.parse(data);

  outputData.forEach((countryData) => {

    const country = countryData.Country;

    const filename = `${country}.json`;

    fs.writeFile(

      filename,

      JSON.stringify(countryData, null, 2),

      "utf8",

      (err) => {

        if (err) {

          console.error(`Error writing ${filename}:`, err);

        } else {

          console.log(`Created ${filename}`);

        }

      }

    );

  });

});

After this splitting I looped through each of the 21 unique files and using a library called “node-couchdb” sent a put request for each file and added them to the database, a subset of these documents can be seen within Fauxton below.

A screenshot of a phone

Description automatically generated

CRUD Operations:

Create:

A screenshot of a computer program

Description automatically generated

Read:

A screenshot of a computer

Description automatically generated

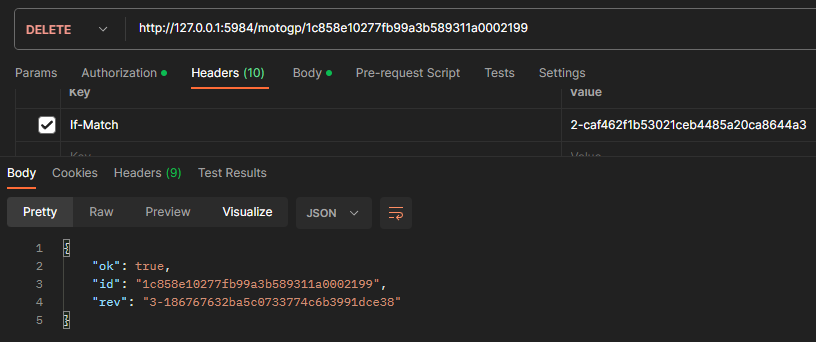
Figure 1Retrieving Document with GET Request

Update:

A screenshot of a computer program

Description automatically generated

Delete:



### MapReduce Functions

### Java or other client for CouchDB

### Replication and mobile app using JavaScript

### DBaaS investigation

## Conclusions:

## References:

## Bibliography: