Name: Adir Ben Ishay

Student number: 667039

Git link: https://github.com/Adir667/INH-PDP

**Diplomacy done on Apache Pig [#2]**

A screenshot of a computer

Description automatically generated with low confidence

1. **Introduction and explanation about Apache pig:**

Apache Pig is a platform for analyzing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs. The salient property of Pig programs is that their structure is amenable to substantial parallelization, which in turns enables them to handle very large data sets.

At the present time, Pig's infrastructure layer consists of a compiler that produces sequences of Map-Reduce programs, for which large-scale parallel implementations already exist (e.g., the Hadoop subproject). Pig's language layer currently consists of a textual language called Pig Latin, which has the following key properties:

Ease of programming. It is trivial to achieve parallel execution of simple, "embarrassingly parallel" data analysis tasks. Complex tasks comprised of multiple interrelated data transformations are explicitly encoded as data flow sequences, making them easy to write, understand, and maintain.

Optimization opportunities. The way in which tasks are encoded permits the system to optimize their execution automatically, allowing the user to focus on semantics rather than efficiency.

Extensibility. Users can create their own functions to do special-purpose processing.

[https://pig.apache.org/]

Apache Pig was developed at Yahoo! Research as an open-source project and became a top-level Apache Software Foundation project in 2007. It was designed to simplify the data processing tasks on large datasets, leveraging the power of Apache Hadoop.

The motivation behind creating Pig was to address the challenges faced by data analysts and developers when working with big data. Traditional approaches for data processing, such as writing MapReduce jobs in Java, often required significant effort and expertise. Pig aimed to provide a higher-level language and a framework to make data processing more accessible and efficient.

Pig Latin, the language used by Pig, was inspired by scripting languages like SQL, Python, and dataflow languages. It is a procedural language that allows users to express data transformations and queries using a concise and expressive syntax. Pig Latin focuses on the flow of data and abstracts away the complexity of distributed computing, allowing users to focus on the logic of their data processing tasks.

Under the hood, Pig optimizes and translates Pig Latin scripts into a series of MapReduce jobs that can run on a Hadoop cluster. It performs various optimizations, such as algebraic simplification, predicate pushdown, and join optimization, to improve the efficiency and performance of the execution.

Pig integrates with the Hadoop ecosystem, including HDFS, YARN, and various data storage systems, making it compatible with different data formats and environments. It also provides extensibility through User-Defined Functions (UDFs), allowing users to incorporate their own custom logic and leverage existing libraries.

Over the years, Pig has gained popularity among data analysts, data scientists, and developers working with big data. It provides a higher level of abstraction, making it easier to express and execute data processing tasks, especially for those who are not proficient in writing low-level MapReduce code.

Pig continues to evolve and improve, with new versions released to introduce new features, enhancements, and bug fixes. The Pig community actively contributes to its development, providing support, sharing knowledge, and expanding its capabilities.

[https://www.youtube.com/watch?v=Hve24pRW\_Ps&ab\_channel=Simplilearn]

1. **Check the setup of pig**

pig --version to see if installed properly:

A screenshot of a computer

Description automatically generated with low confidence

1. **Pig scripts**

Writing a script in pig is an easy way to interact with the program, that is very similar to working with other programming languages like Python, with a syntax that is also very close to SQL.

In order to understand the columns in the dataset, I used the diplomacy.sql file that describes the data structure and relations.

This file is very large, so in order to show only what I wish you see I used the following linux command in cli: “cat diplomacy.sql | grep 'CREATE' -A 10”

This way I can capture the first 10 lines after every “CREATE” statement in the sql file, to see the structure of each table and understand the columns.

For the first script, we are using the orders.csv file (also, the orders sql table).

A screen shot of a computer code

Description automatically generated with low confidence

1. Script - Summary of moves to “Holland”

The following script is generating a “grouped” list of all the moves from any location to “Holland”, and shows how many times this move was performed.

**A screenshot of a computer

Description automatically generated**

Result:

*("Adriatic Sea","Holland",1)*

*("Aegean Sea","Holland",5)*

*("Albania","Holland",1)*

*("Armenia","Holland",1)*

*("Baltic Sea","Holland",326)*

*("Barents Sea","Holland",38)*

*("Belgium","Holland",35134)*

*("Berlin","Holland",1282)*

*("Black Sea","Holland",3)*

*("Bohemia","Holland",5)*

*("Brest","Holland",32)*

*("Budapest","Holland",1)*

*("Bulgaria","Holland",2)*

*("Burgundy","Holland",1153)*

*("Clyde","Holland",19)*

*("Constantinople","Holland",4)*

*("Denmark","Holland",4051)*

*("Eastern Mediterranean","Holland",4)*

*("Edinburgh","Holland",3023)*

*("English Channel","Holland",1231)*

*("Finland","Holland",2)*

*("Galicia","Holland",1)*

*("Gascony","Holland",6)*

*("Greece","Holland",3)*

*("Gulf of Bothnia","Holland",2)*

*("Gulf of Lyons","Holland",3)*

*("Helgoland Bight","Holland",9107)*

*("Holland","Holland",325)*

*("Ionian Sea","Holland",11)*

*("Kiel","Holland",44658)*

*("Liverpool","Holland",90)*

*("Livonia","Holland",2)*

*("London","Holland",3519)*

*("Marseilles","Holland",2)*

*("Mid-Atlantic Ocean","Holland",30)*

*("Moscow","Holland",2)*

*("Munich","Holland",2297)*

*("Naples","Holland",1)*

*("North Africa","Holland",1)*

*("North Sea","Holland",16250)*

*("Norway","Holland",740)*

*("Norwegian Sea","Holland",233)*

*("Paris","Holland",10)*

*("Picardy","Holland",501)*

*("Portugal","Holland",5)*

*("Prussia","Holland",8)*

*("Rome","Holland",3)*

*("Ruhr","Holland",22142)*

*("Rumania","Holland",2)*

*("Serbia","Holland",4)*

*("Silesia","Holland",4)*

*("Skagerrack","Holland",164)*

*("Smyrna","Holland",1)*

*("Spain (South Coast)","Holland",1)*

*("Spain","Holland",5)*

*("St. Petersburg (North Coast)","Holland",10)*

*("St. Petersburg","Holland",24)*

*("Sweden","Holland",73)*

*("Syria","Holland",1)*

*("Tunis","Holland",2)*

*("Tyrolia","Holland",4)*

*("Tyrrhenian Sea","Holland",11)*

*("Venice","Holland",2)*

*("Vienna","Holland",2)*

*("Wales","Holland",37)*

*("Warsaw","Holland",1)*

*("Western Mediterranean","Holland",13)*

*("Yorkshire","Holland",2882)*

Screenshot:

**A screen shot of a computer program

Description automatically generated with low confidence**

1. Script - Winning countries

For this script we need to examine the “players” table:

**A picture containing text, screenshot, font

Description automatically generated**

Filtering in this script is easier performed when names are given to the columns. Instead of $n.

A screenshot of a computer

Description automatically generated with medium confidence

Result:

("A",3008)

("E",2960)

("F",3305)

("G",3439)

("I",2013)

("R",4110)

("T",4457)

Screenshot:

A screenshot of a computer program

Description automatically generated with medium confidence