**ETL\_README.md**

**Brief Explanation of the Approach:**

* **System Setup Requirements:**

**Installed:**

Oracle Virtual-Box

Ubuntu Latest

Pyspark

Python

Git

Conda

* **Environment Setup:**

**Oracle VM settings:**

Setup Bridge network to access outside internet

Allocated 20GB of disk space

Allocated 5GB of memory

Ubuntu Iso image

Start the VM and install Ubuntu

**Ubuntu Settings:**

Wget, apt update, Dns server, network setup, hostname setup

**Ubuntu installed packages:**

pyspark

python

git

conda

**Note: I used conda for packaging and spark submitted the job. The code can be run locally or can be used in a cluster.**

**CODE APPROACH EXPLANATION:**

**TASK1 Requirements:**

**Using Apache Spark and Python, read the source data, pre-process it and persist (write) it to ensure optimal structure and performance for further processing.**

* Download the input json files to local folder and define this path in the config.ini file as inputLocation
* Define the required imports in a python file. My project file is named as hellofresh.py
* Read the json file using spark.read and store into a dataframe.

I have stored the dataframe as df

A picture containing graphical user interface

Description automatically generated



**TASK2 Requirements:**

**Using Apache Spark and Python read the processed dataset from Task 1 and:**

**1.Extract only recipes that have beef as one of the ingredients.**

To extract the recipes with beef ingredients, I filtered the ingredients column as below:

\*



1. **Calculate average cooking time duration per difficulty level.**

Steps taken to calculate the average cooking time :

* Converted the cook time into seconds using convert function and stored into a dataframe named as “dfTimeConversionCookTime”
* Converted the prep time into seconds using convert function and “dfTimeConversionCookTime” as an input and stored into a dataframe named as “dfTimeConversionPrepTime”
* Used “dfTimeConversionPrepTime” as the input to calculated the sum of two columns cooktime and preptime(which is in seconds) and stored as “dfTotalCookTimeSeconds”
* Converted “dfTotalCookTimeSeconds” to minutes by defining the userdefined function on the column TimeInMin(new column after calculating the total) and stored as “dfTImeInMin”

A computer screen capture

Description automatically generated with medium confidence

**Criteria for levels based on total cook time duration:**

* **easy - less than 30 mins**
* **medium - between 30 and 60 mins**
* **hard - more than 60 mins.**
* To add the column with difficulty based on the above criteria, I have used compare operators and added a column to the above dataframe and stored as “dfDifficulty”

Graphical user interface, text

Description automatically generated

1. **Persist dataset as CSV to the output folder.  
   The dataset should have 2 columns: difficulty,avg\_total\_cooking\_time.**

* To calculat the avg\_total\_cooking time, I have used agg(avg) on the TimeInMins column grouping by difficulty column

Text

Description automatically generated

* Using write.csv, I wrote the output to the output folder which is in csv.

**How to run the code?**

**Type1: Execute using spark-submit on local**

Spark-submit hellofresh.py

A screenshot of a computer

Description automatically generated with medium confidence

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

A picture containing text

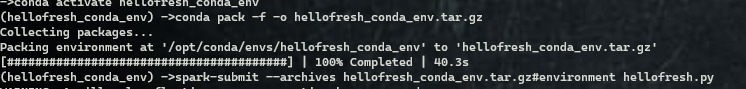
Description automatically generated

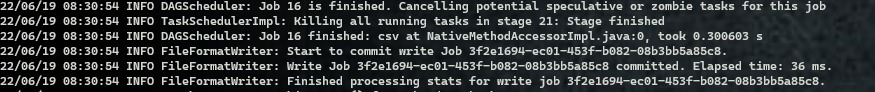
**Type2: Packaging the environment needed to run the script**

As mentioned already in requirements, conda is already installed

Text

Description automatically generated





**\*\*This code can be submitted to any of the cluster for example yarn on hadoop.**