Assignment 6:

Given a sample log file, write a script using grep to extract all lines containing "ERROR". Use awk to print the date, time, and error message of each extracted line.
Data Processing with sed

Sol:

```
#!/bin/bash

# Sample log file path

log_file="sample.log"


# Extract lines containing "ERROR" using grep, and then use awk to extract date, time, and error message

grep "ERROR" "$log_file" | awk '{print $1, $2, $0}' | sed -E 's/^[^ ]+ [^ ]+ ([^ ]+ [^ ]+) (.*)/\1 \2/'
```

Explanation:

1. **`grep "ERROR" "$log_file"`:**
   - `grep` is a command-line utility for searching plain-text data sets for lines matching a regular expression.
   - In this script, `grep` is used to search for lines in the log file (`$log_file`) that contain the string "ERROR".
2. **`awk '{print $1, $2, $0}'`:**
   - `awk` is a versatile programming language for working on files and generating reports.
   - In this script, `awk` is used to process the output from `grep`.
   - `{print $1, $2, $0}` tells `awk` to print the first field (date), the second field (time), and then the entire line.
3. **`sed -E 's/^[^ ]+ [^ ]+ ([^ ]+ [^ ]+) (.*)/\1 \2/'`:**
   - `sed` (stream editor) is a powerful text-processing utility.
   - `-E` enables extended regular expressions for `sed`.
   - `'s/^[^ ]+ [^ ]+ ([^ ]+ [^ ]+) (.*)/\1 \2/'` is a substitution command:
     - `^` denotes the beginning of a line.
     - `[^ ]+` matches one or more characters that are not a space.
     - `([^ ]+ [^ ]+)` captures the date and time in parentheses.
     - `(.*)` captures the remaining part of the line (the error message).

        - `\1 \2` references the captured groups (date/time and error message) for replacement.
- Essentially, this `sed` command discards the initial fields (date and time) and only retains the captured date/time and error message.