Technical test for Rystad Energy (data) Analyst

This exercise is a simplified version of a real-life workflow. You will download, process and interpret some production figures from the Gulf of Mexico, straight from public data sources. Partial solutions will not disqualify you automatically; if you are unable or do not wish to spend too much time on any particular question or sub-task, please indicate (as a comment in your deliverables, as detailed as possible) how you would have proceeded if you had more time at hand.

Expected completion time: 2-3 hours assuming sufficient familiarity with SQL, Python web crawling, and basic Excel, and likely some more hours in case of lack of significant python web scrapping experience.

Content

Section A: python, crawling
Section B: SQL
Section C: Excel and interpretation of results

Deliverables: 3 files, zipped in a single file.

- The python script code as .txt document (for security reasons our IT system will block attachments with .py extension). There are some qualitative questions in the task description: please provide your answers as commented sections in your code. No need to excessively document the script. Feel free to use any open-source libraries; the more common they are, the better.
- A text or Word file containing the SQL queries from part B. Any comments can be added there.
- An Excel workbook as described in part C.

Part A – Python

Write a python script that upon execution (from an arbitrary location) navigates to the BSEE website, checks for latest updates on certain datasets, downloads only necessary files, and generates/refreshes a clean output file. It is suggested to proceed as follows

1. The script should first download zipped well-production data for **2019 and 2020 in the Gulf of Mexico** from the following website: https://www.data.bsee.gov/Main/OGOR-A.aspx
   - You may choose to work with either the fixed-length or delimited datasets. Clicking on "OGOR-A Production 2020" will show you the field definitions.
   - The script should write to a sub-folder (e.g. "Download") of the working directory where the python script is executed. That is, the script **should work independently of the exact path where the final user decides to run it from**. If this Download folder does not exist, the script should create it.

2. **Unzip the compressed archives**, saves the "raw" txt content files into the folder, and delete the zip files. Notice that the website indicates when each data file was "last updated". Make sure you keep track of this timestamp in the unzipped file: you can encode it in the name of the file, e.g. ogora2019_20220601. Notice each individual file may have a different "last updated" timestamp, even if this is not the case on the date you visit the website.

3. **Convert the "raw" downloaded files into a single clean flat file, where the column separators are vertical bars "|",** while rows are delimited by new line symbol. Note there are 19 columns in this dataset, but we are only interested in the following: **lease_number; production_date, product_code, monthly oil and gas volumes, operator_num.** The script should save this file in the same folder, call it "production.csv". Do clean a bit the data if necessary (provide a comment in the script if you feel this is the case). At this stage the contents of the "Download" folder should be a single clean flat file containing both 2019-2020 production (with 6 columns) plus the 2 timestamped "raw data" txt files.

4. Imagine this script is somehow run periodically and automatically, with the intention of re-downloading any data file that has been refreshed on the BSEE website. **Modify step 1 so that the script ONLY downloads and processes a particular file if the "last updated" flag on the website suggest the data has been refreshed**. If at least one file has changed, make sure the clean output file is also refreshed. Otherwise do nothing. For example, if no files are present (e.g. the first run), it should still run and download everything, and if it is run for example twice in a row, the second run should do nothing.

Keep the following design guidelines in mind: in "real life", a workflow of this kind would need to be scaled, for example to include all years. Hence try to keep the code generic enough: your final script should still only deal with 2019-2020, but can I download 2008-2015 instead by just changing a single line or parameter in your code? Would it be able, upon light modification, to include pick up all files, including the most recent one, even if run a few years from now?

Part B- SQL

For this part, you do not need to actually work with an SQL server or engine. This only intends to test your conceptual ability to write queries. If you are unable to write answer a question with a single query (eventually nested), it is OK to provide a sequence of queries (eventually with #temporary tables).

Start with the clean 2019-2020 file generated in the previous step (the format nicely fits what could be used to dump it into a SQL database). If you have no experience in Python, it is OK to skip part A, but you will need to download manually the files described in part A and generate a clean version of it (you will also need this for part C).

1. Imagine this dataset is in fact an SQL table, called `GoM_Production`. It has seven columns. **Explore your clean file first. What would be the ideal data structure? Describe this by writing a fictive query that would create an empty table GoM_Production with 6 named columns with adequate data types.**

```
CREATE TABLE Gom_Production
([Lease_Number] float,--
[Production_date] varchar(100) --etc
)
--N.B: these datatypes are not correct for this particular table
```

2. Imagine you have another table, called `Lease_info`, that contains some information about the lease: here is small slice of actual data from BSEE. Assume there are no NULLS in the first column, water depth is int, and the other two columns are text.

Lease_number	Lease_Expiration_Date	Block_Max_Water_Depth	Area_Code
G20671	20000602	181	HI
00104	19570623	53	HI
G05368	NULL	79	EC
G01192	NULL	109	SM
G12438	19950930	69	MI
G09487	19930531	561	EC
G04063	NULL	145	MU

2a. **You suspect `Lease_info` may contain more than one row per lease: write a query to check this.** Assume from now onwards there is a single row per lease_number.

2b. **You are interested in leases in Area "EC". Write a query to obtain the total oil production from this lease in year 2019.**

2c. **You care about leases in "HI" and "SM" regions where the block's max water depth is between 100 and 200 feet. Write a query that would tell us how many such leases are not expired as of December 2020 (treat NULLs as unexpired leases).**

3. Finally, you have a third table called `Calendar` which contains, for each month, the number of days in the month. Assume int and no nulls

Date	Days
201901	31
201902	28
201903	31
201904	30

a. **Write a query to display total oil production volume per Area, year, and month.**
b. Oil and gas production is typically thought of in "barrels per day" figures instead of absolute volumes. For example 28,000 barrels in February and 31,000 barrels in March means no change in monthly production rate. **Modify your previous query to obtain values expressed in thousand barrels per day, by performing a join on this table.**
c. **Write a query that returns the number of months between 2019 and 2020, if there are any, where gas production fell below 2 billion cubic feet per day** (gas production in the downloaded BSEE data is reported in thousand cubic feet).
d. **Can you write a query that returns the list of leases that produced some oil volumes at least 10 out of 12 months in 2020, AND whose average production rate in 2019 was above 10,000 barrels per day?**

Part C - Excel

Create an Excel file (preferred, but open-source alternatives are acceptable) file with the following data:

- The clean data source from part A.3 as input, you may remove irrelevant rows or columns to keep file size reasonable.
- As a separate tab, include data linking operator number and operator name, which you can download from here: https://www.data.bsee.gov/Company/Files/compalldelimit.zip
- A pivot table, which will help you answer the following 3 questions

1. Question 3.c from part B by using Excel instead of SQL: **During how many months between 2019 and 2020, if any, was gas production rate on average below 2 billion cubic feet per day?**
2. Speculate a little bit on **what may have driven these production swings**.
3. Which were the top operators in term of oil production in the Gulf of Mexico, during 2019-2020?

Ideally, a single pivot table should be enough, but feel free to proceed as you feel more comfortable.

Your deliverable is the excel file, with the above 3, together with 2 additional tabs: in the first one, include a graph showing how gas production (in million cubic feet per day) changed over 2019-2020; add a small text box with 2-3 sentences answering question 1 and commenting on point 2. Add another tab with a clean chart, that in your opinion clearly illustrates graphically the answer to question 3.

Feel free to manipulate your “clean” input file as you see fit, import additional data from other sources or tabs if there is a need for it.