Technical prompt answer and libraries file

Main.py

```
#main.py



#put in libraries.txt, these are the only external libraries used

#I used pandas for reading and manipulating the CSV data

#I used tkcalendar for the date picker in the GUI

import pandas as pd

import tkcalendar as tkcal

from tkcalendar import Calendar, DateEntry



#comes preloaded with python

#I used tkinter for the GUI and pop-up messages

import tkinter as tk

from tkinter import messagebox, filedialog



#path to the CSV file, change this in ordeer to point to your data file

#This is assuming that the main.py is in the same directory as the CSV file

CSVPath = "core-data.csv"



#makeReport reads the csv file and does all the processing to spit out the end report

def makeReport(startDate, endDate):



#This is because the date pickers return date objects, we need to convert them to
datetime objects for comparison

#I got into an error here before
```

```
    startDate = pd.to_datetime(startDate)

    endDate = pd.to_datetime(endDate)



#This just reads the csv file with error checking

#It also parses the "Date" column as datetime objects and in the correct format,
month/day/year hour:minute and am/pm

    try:

        dates = pd.read_csv(CSVPath,

                            parse_dates=["Date"],

                            date_format="%m/%d/%Y %I:%M %p")

    except Exception as e:

        messagebox.showerror("Error", f"Couldn't get the CSV file: {e}")

        return



#This filters the data based on the range you chose

    mask = (dates["Date"] >= startDate) & (dates["Date"] <= endDate)

    filteredDates = dates.loc[mask]



    if filteredDates.empty:

        messagebox.showinfo("No Data", "Theres is no data in the selected date
range.")

        return



#This groups the data by "Part Type", "Operation", and "Status"

#and also counts the amount of each status for each part type and operation
```

```
    returnedItems = (

        filteredDates

        .groupby(["Part Type", "Operation", "Status"])

        .size()

        .unstack(fill_value=0)

        .reset_index()

    )



    print(returnedItems)



#This makes sure that the columns "Good", "Scrap", and "Review" exist

#if they don't exist, it adds them with a default value of 0

    for col in ["Good", "Scrap", "Review"]:

        if col not in returnedItems:

            returnedItems[col] = 0



#This calculates the "Total" and "Yields" columns

    returnedItems["Total"] = returnedItems["Good"] + returnedItems["Scrap"] +
returnedItems["Review"]

    returnedItems["Yields"] = (returnedItems["Good"] / returnedItems["Total"])



    output = returnedItems[

        ["Part Type", "Operation", "Total", "Good", "Scrap", "Review", "Yields"]

    ]
```

```
#THis one opens a popup to let you choose where to save the CSV and lets you change
the name

#also checks if it actually did it

    savePath = filedialog.asksaveasfilename(

        defaultextension=".csv",

        filetypes=[("CSV files", "*.csv"), ("All files", "*.*")],

    )



    if savePath:

        try:

            output.to_csv(savePath, index=False)

            messagebox.showinfo("Success", f"Saved the report to {savePath}")

        except Exception as e:

            messagebox.showerror("Error", f"Couldn't save the report: {e}")




#Makes the GUI

def runGui():

    root = tk.Tk()

    root.title("Report Generator")

    root.geometry("300x200")

    root.resizable(True, True)



#Date checker 1 and 2

    tk.Label(root, text="Start Date:").pack(pady=5)

    startCal = DateEntry(root,
                            width=12,

                            background='darkblue',

                            foreground='white',

                            borderwidth=2, )

    startCal.pack(pady=5)



    tk.Label(root, text="End Date:").pack(pady=5)

    endCal = DateEntry(root,

                          width=12,

                          background='darkblue',

                          foreground='white',

                          borderwidth=2, )

    endCal.pack(pady=5)



#Calls makeReport when clicked

    tk.Button(root,

                text="Generate Report",

                command=lambda: makeReport(

                        startCal.get_date(),

                        endCal.get_date())

                 ).pack(pady=20)

    root.mainloop()
```

```
#runs it

if __name__ == "__main__":

    runGui()
```

Libraries.txt

External libraries used:

Pandas: for CSV data manipulation
tkcalendar: for the easy date selector