Assignment Questions

For all the questions, please provide minimal reproducible code which I can run locally and check the output
Be sure not to hardcode absolute file paths in the code and ensure that it runs irrespective of the system platform.
For the sql queries, make sure it is readable by adding proper formatting wherever necessary.
Please submit the assignment in a jupyter notebook

Question 1.

Given the sample.csv, convert to to give the expected output format

Sample CSV

	Name	Value
0	M_ID	30001006
1	P_PART	Test Part
2	M_STATE	red
3	NOZZLE_TEMP	25
4	ZONE_TEMP	200
5	C_TIME	65.6
6	POWER	3245.67
7	TS	2021-04-20T12:00:00

Expected Output

```
[
    {"param_name": "machineId", "param_value": "300010006", "timestamp": "2021-04-20T17:30:00+0530"},
    {"param_name": "partNumber", "param_value": "Test Part", "timestamp": "2021-04-20T17:30:00+0530"},
    {"param_name": "machineStatus", "param_value": "red", "timestamp": "2021-04-20T17:30:00+0530"},
    {"param_name": "nozzleTemp", "param_value": "25", "timestamp": "2021-04-20T17:30:00+0530"},
    {"param_name": "zoneTemp", "param_value": "200", "timestamp": "2021-04-20T17:30:00+0530"},
    {"param_name": "cycleTime", "param_value": "65.6", "timestamp": "2021-04-20T17:30:00+0530"},
    {"param_name": "powerConsumption", "param_value": "3245.67", "timestamp": "2021-04-20T17:30:00+0530
}",
]
```

Name Mapping

```
M_ID        = machineId
P_PART      = partNumber
M_STATE     = machineStatus
NOZZLE_TEMP = nozzleTemp
ZONE_TEMP   = zoneTemp
C_TIME      = cycleTime
POWER       = powerConsumption
TS          = timestamp
```

Things to note:
- TS will be in UTC which needs to be converted to the corresponding IST in the requested format
- All param values must in string type irrespective of the input data type
- Use only standard libraries (use of pandas to load the csv is discouraged)

Your code should take in a csv in the sample format as input and output a list in the requested format.

Question 2

In a normal hash table, a key is hashed then linear search is performed in the bucket where the key will be found if it exists in the hash table. A successful search terminates as soon as the key is found, on average half-way through the bucket, but an unsuccessful search requires that every item in the bucket be examined.

Donald Knuth proposed that instead of keeping keys in a bucket in random order they be kept in increasing order, so that a search can stop as soon as the search passes the value of the key. That means inserts take longer; you have to find the right place in the bucket to put the key instead of just putting it at the beginning of the bucket. But lookups a re quicker, as an unsuccessful search can terminate half-way through the bucket, on average. This change also mean s that the data type of the hash table changes, as now the comparison function is less-than rather than equal-to.

Your task is to write a small library of ordered hash tables; you should provide lookup, insert and delete functions, at least.

Things to note:
- Not much emphasis on the time and space complexity, more interested in the logic

Question 3

SQL Assignment

Please use the IMDB dataset provided and write queries to the solve the following problems -

a. Find the film(s) with the largest cast. Return the movie title and the size of the cast. Note that "cast size" means the number of distinct actors that played in that movie: if an actor played multiple roles, or if it simply occurs multiple times in casts, it should still count her/him only once.
b. For each year in the dataset, count the number of movies in that particular year that had only female actors.

Things to note
- Go through the db schema to understand the layout of the data
- Use of pandas to load the dataset is allowed for this question (use sqllite3 to avoid any dependency issue)
- Make sure all the functions used in the queries are supported in sqllite3
- Make sure to pre-process the data wherever necessary, for example:
  Removing leading and trailing spaces in strings.
  For year, you would have to extract the exact numeric year from it before making any year related comparisons in your queries.