Homework 1: Distributed RDBMS and NoSQL

Form:	Theoretical - Written report (Word) Practical – Jupyter notebook
Language:	English or Hebrew
Requirements:	Theoretical - The report should be up to 2 pages Practical - The code should run without errors
Submission:	Course Website – moodle
Contact:	yotamdery@mail.tau.ac.il
Deadline for submission:	**14/04/2022**

Students will form teams of three people each and submit a single homework for each team. The same score for the homework will be given to each member of the team. Please write all group members ids in the document and Jupyter notebook.

The goal of this homework is to test your understanding of the concepts presented in the first three lectures and the first lab. If your answers are based on a material you have read elsewhere, please make sure to avoid plagiarism (i.e., cite the source and phrase each answer with your own words).

Theoretical

Question 1: Describe the four ACID properties and for each of them provide an example that shows how the property can be broken when no mechanism is used to ensure it.

Question 2: Describe replication and fragmentation in the context of a distributed RDBMS. Provide some of the advantages and disadvantages of both. Describe two real-world examples representing the benefits in using vertical and horizontal fragmentation.

Question 3: Read the paper: “Dynamo: Amazon’s Highly Available Key-value Store” and describe the main advantages and limitations of the Dynamo platform (your answer should not exceed half a page!).

Practical

The homework assignment should be submitted in a Jupyter notebook **(including outputs)**. Use Markdown or comments for the important code lines. The Jupyter notebook should run without errors or bugs by order top to bottom.

Scenario: Develop a website that allows companies to post jobs and candidates can apply for these jobs. For simplicity, you can assume that the generated data has the following structure:

- Company: Company Name, Company Description, Jobs List
- Jobs: Job Id, Job Title, Location, Requirements List, Publish Date, Status, Applications List
- Applications: Candidate Name, Email, LinkedIn URL, Skills List, Application Date

The system will comprise two components: persistent storage and an in-memory cache. The persistent storage will be implemented using MongoDB and will contain all the information that must be stored for the correct functioning of the website, as well as for analytical purposes. Note that for the purposes of the exercise you are not allowed to use more than **one collection**. The in-memory cache will be implemented using Redis and will take care of the operational side including fast operations.

Create the following functions:

Operation 1: Add a new company. Create the function *add_company(company)* that adds the *company* (a python dictionary) to the system. The Company Name should be unique, so you cannot accept more than one Company with the same Company Name. An example of a call to this function is as follows:

```
add_company({'company_name':'TAU', 'company_description':'University'})
```

Operation 2: Add a new job position. Create the function add_job(job, company_name) that adds the Job (a python dictionary) to the company. The Job Id should be unique for a given company and automatically generated when a new job is added to the company. An example of a call to this function:

```
add_job({'job_title':'bi developer', 'location': 'Tel Aviv',
'requirements':['python','big data','mongodb'],
'status':'open','publish_date':'01-02-2020'},'TAU')
```

Operation 3: Add a new application. Create the function *new_application(candidate, application_time, job_id, company_name)* that adds the *candidate* (a python dictionary) to the Job. A candidate can apply when the job Status is ‘open’ and only once to the same job. The Candidate Email should be unique for a given candidate. An example of a call to this function:

```
new_application({'candidate_name':'laura', 'email':'laura@gmail.com',
'linkedin':'https://www.linkedin.com/in/laura/', skills: ['python','sql']},
'01-02-2020 15:00:00', '1','TAU')

#note that this means that Laura applied for job number 1 in Company TAU at
'01-02-2020 15:00:00'
```

Operation 4: Create the function *update_job_status(company_name, job_id, new_status)* that updates the job status. An example of a call to this function:

```
update_job_status('TAU','1','close')

update_job_status('TAU','1','open')
```

please note – the following operations (both 5 & 6) should have a run time complexity of O(1)

Operation 5: Given a candidate, show the latest 10* (unique) companies that the candidate applied to. Define and implement the function *show_latest_10_companies(candidate_email).* An example of a call to this function:

```
show_latest_10_companies(‘lebron@gmail.com’)
```

* Should there are less than 10 companies, show all of them.

Operation 6: Create the function *show_number_of_jobs(location, job_title)* that returns the number of jobs with the given job_title and location having the status ‘open’. An example of a call to this function:

```
show_number_of_jobs('Tel Aviv','bi developer')
```

alongside to the operations above, you’re asked to produce the following reports:

Report 1: Define and implement the function *count_jobs_by_company()* that counts how many jobs with status ‘open’ and how many jobs with status ‘closed’ are there, for each company name. Order the output by jobs with status ‘open’, than by jobs with status ‘closed’, in descending order.

Report 2: Create the function *count_candidates_by_job()* that returns the number of candidates for each Job ID during the last 20 days.

Recovery: Create the function *recovery()* that should be executed whenever the Redis server restarts (for example after a server crash like power failure). This function will use the persistent data (stored on MongoDB) to reload data to the cache (Redis).

Execution: Create the function *execute()* that call all functions with the arguments given in the examples. All the function should return an output. Print all the outputs in a readable and nice displayed.

Evaluation

Special attention will be given to the data model that you defined and to the efficient implementation of the different operations/reports, so you should make sure to include in the notebook also a description of your model and motivations of your implementation choices. The code should be mainly based on built-in functions of MongoDB and Redis. The code should be organized and the outputs should be nice displayed.

Good Luck!