-SQL Database table:

Table name: Employee

Column	Type	Description
Employee Code	VarChar	The unique employee code that identifies an employee. For example, 'ABC1234'
Department	VarChar	The department to which the employee belongs like 'Engineering, HR, Operations' etc
Score	Integer	A numerical score given to an employee.
Date Created	Datetime	Date and time the employee record was created in the database.

Initial Condition:

There are many employees or records in the Employee Table.

Problem Statement/Requirement:

Create a Django or any other python web framework application which will do the following:

1. Provide a JSON API with the following URL: /employees/?chunk=<chunk number> . where 'chunk' is a query parameter and 'chunk number' is a positive nonzero integer starting from 1. To see an example of this API response, go to point no:3.

2. To develop second API without any query parameter using HTTP GET. Note down the below steps to be followed in the following order:
   a. Application will query the database to fetch all the employee records. But arrangement to be done according to (b and c) conditions.

   b. Application will then return the results/employees sorted in descending order of score in JSON response. JSON response will contain an array of employee objects, for example:

```
{ 'employees':
        [{'employee_code': 'E1', 'department':'D1', 'score':100},
{'employee_code':'E2', 'department': 'D2', 'score':89}…..]
        }
```

c. In the employees' array, every 5th and 6th indices will only contain employees from specific department – 'Unthink' and every 7th and 8th indices will only contain employees created within last 14 days irrespective of their scores. For example, in the response below, the employee objects highlighted in yellow are from department 'Unthink', the employee objects highlighted in green were created within last 14 days starting today. The records satisfying these two conditions are not sorted with respect to scores.

{ 'employees':
[{'employee_code': 'E10', 'department':'D1', 'score':100},
{'employee_code':'E1', 'department': 'D2', 'score':89},
{'employee_code':'E8', 'department': 'D2', 'score':88},
{'employee_code':'E23', 'department': 'D2', 'score':88},
{'employee_code':'E36', 'department': 'Unthink', 'score':10},
{'employee_code':'E100', 'department': 'Unthink', 'score':29},
{'employee_code':'E51', 'department': 'D3', 'score':5},
{'employee_code':'E17', 'department': 'D5', 'score':17},
{'employee_code':'E91', 'department': 'D2', 'score':87},
{'employee_code':'E11', 'department': 'D2', 'score':86},
{'employee_code':'E44', 'department': 'D2', 'score':86},
{'employee_code':'E71', 'department': 'D4', 'score':85},
{'employee_code':'E14', 'department': 'Unthink', 'score':2},
{'employee_code':'E15', 'department': 'Unthink, 'score':3},
{'employee_code':'E9', 'department': 'D5', 'score':10},
{'employee_code':'E20', 'department': 'D4', 'score':30},
{'employee_code':'E32', 'department': 'D2', 'score':84},

…..
]
}

d. In the response, no employee object will be repeated in the employee objects array.

3. When the API is called with query parameter 'chunk' with a positive integer value n, the result from point 2 will be broken into chunks of 20 employee objects each and the nth chunk will be sent in the JSON response (array of 20 employee objects). For example,

If the result in 2 is [EO1, EO2, EO3,….,EO20, EO21, EO22,….EO40, EO41, EO43,…] where EO is employee object and api called is /employees/?chunk=3, then the response will be {'employees' :[EO41, EO42, EO43….EO60]}

Solution/Application Expectations:

To make it simple, you can insert the data to database in such a way that you can easily get response satisfying the above questions.

The application solution should be optimal and efficient. The API response should be quick. Mention the number of database queries made by the application to send the response in both points 2 and 3 respectively.

Please send the source code files zipped as mail attachment. Or if you have a public GitHub account, you can upload the source code there and share the URL. The code should be well commented to explain the logic used.
The solution should be provided within two-three days of receiving the assignment. Please contact us for any queries related to the assignment.