There will be two sections in the code exam. The first one is python and the second is SQL.

Python Challenges:

Please find attached example.json file. You have to understand JSON file and solve the following problems. Please solve the below questions without using any third-party python packages like Pandas or NumPy.

1. **Read the JSON file and print the output in the following format.**

   Expected Output:

   Type of JSON Object: <class 'dict'>

   Name: John

   Phone Number: +61 2 3617 9451

   Age: 19

   Address:

   10/365 Pacific Highway, Hornsby,

   Sydney, New South Wales,

   Australia - 2077.

   Name: Ravi

   Phone Number: +91 9972354015

   Age: 21

   Address:

   110 New Vora House,

   Koramangala

   Bengaluru, Karnataka

   India - 560078.

2. **Count the number of lines needed to store the addresses of the students. Store these values in a list along with the person's name, and print the list.**

Expected Output:

Person vs Number of Address Lines:

   John 3

   Ravi 4

3. **Create a new JSON file that stores only the mappings between the students' names and their phone numbers.**

Expected Output:

{"John": "+61 2 3617 9451", "Ravi": "+91 9972354015"}

4. **Modify the example.json file to store the student addresses in a single string instead of a list of address lines.**

   Expected Output:


```
{
   "John": {
      "number": "+61 2 3617 9451",
      "age": 19,
      "address": "10/365 Pacific Highway, Hornsby, Sydney, New South Wales, Australia - 2077."
   },
   "Ravi": {
      "number": "+91 9972354015",
      "age": 21,
      "address": "110 New Vora House, Koramangala Bengaluru, Karnataka India - 560078."
   }
}
```


SQL Challenges:

You will be given a schema definition and you have to write SQL query against it. Sample output will also be provided.

1. **Write an SQL query to report the first name, last name, city, and state of each person in the Person table. If the address of a personId is not present in the Address table, report null instead.**

   Table: Person

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| personId    | int     |
| lastName    | varchar |
| firstName   | varchar |
+-------------+---------+
```

   Table: Address

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| addressId   | int     |
```

```
| personId    | int     |
| city        | varchar |
| state       | varchar |
+-------------+---------+
```

Input:

Person table:

```
+----------+----------+-----------+
| personId | lastName | firstName |
+----------+----------+-----------+
| 1        | Wang     | Allen     |
| 2        | Alice    | Bob       |
+----------+----------+-----------+
```

Address table:

```
+-----------+----------+---------------+------------+
| addressId | personId | city          | state      |
+-----------+----------+---------------+------------+
| 1         | 2        | New York City | New York   |
| 2         | 3        | Leetcode      | California |
+-----------+----------+---------------+------------+
```

Output:

```
+-----------+----------+---------------+----------+
| firstName | lastName | city          | state    |
+-----------+----------+---------------+----------+
| Allen     | Wang     | Null          | Null     |
| Bob       | Alice    | New York City | New York |
+-----------+----------+---------------+----------+
```

2. Write an SQL query to find the employees who earn more than their managers.

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| id          | int     |
| name        | varchar |
| salary      | int     |
| managerId   | int     |
+-------------+---------+
```

Input:

Employee table:

```
+----+-------+--------+-----------+
| id | name  | salary | managerId |
+----+-------+--------+-----------+
| 1  | Joe   | 70000  | 3         |
| 2  | Henry | 80000  | 4         |
| 3  | Sam   | 60000  | Null      |
| 4  | Max   | 90000  | Null      |
+----+-------+--------+-----------+
```

Output:

```
+----------+
| Employee |
+----------+
| Joe      |
+----------+
```

3. **Write an SQL query to report all the duplicate emails.**

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| id          | int     |
| email       | varchar |
+-------------+---------+
```

Input:
Person table:

```
+----+---------+
| id | email   |
+----+---------+
| 1  | a@b.com |
| 2  | c@d.com |
| 3  | a@b.com |
+----+---------+
```

Output:

```
+---------+
| Email   |
+---------+
| a@b.com |
+---------+
```

4. **Write an SQL query to report the first login date for each player.**

Input:
Activity table:

```
+-----------+-----------+------------+--------------+
| player_id | device_id | event_date | games_played |
+-----------+-----------+------------+--------------+
| 1         | 2         | 2016-03-01 | 5            |
| 1         | 2         | 2016-05-02 | 6            |
| 2         | 3         | 2017-06-25 | 1            |
| 3         | 1         | 2016-03-02 | 0            |
| 3         | 4         | 2018-07-03 | 5            |
+-----------+-----------+------------+--------------+
```

Output:

```
+-----------+-------------+
| player_id | first_login |
+-----------+-------------+
| 1         | 2016-03-01  |
| 2         | 2017-06-25  |
| 3         | 2016-03-02  |
+-----------+-------------+
```