FINRA Engineer Questionnaire

Within this document are four different questions. Each question is structured in the following manner:

1. Premise  
    - Contains any needed background information
2. Request  
   - The actual question, what you are to solve
3. Notes  
   - A space if you feel like including notes of any kind for the given question

Please place your answer for each question in a separate file, following this naming convention:

FINRA\_Q**n**.txt, where **n** = the question number (i.e., 1, 2 ...). So the file for the first question should be named ‘FINRA\_Q1.txt’

Reponses can be code or pseudocode, with higher preference given to actual coded solutions. If a specific language is not requested by the question, please indicate the chosen language within a comment at the top of your solution. If you feel it necessary, add comments to parts of your solutions.

When complete, please package everything together and send email responses to the designated POCs.

**Premise:**You have a table with one column, `original\_date`, of datatype string

|  |
| --- |
| ORIGINAL\_DATE |
| 20190825 |
| 20190826 |
| 20190827 |
| 20190828 |
| 20190829 |
| 20190830 |
| 20190831 |
| 20190901 |

**Question:**  
Write a (preferably hive) **SQL** query to calculate two more columns –

1. `end\_of\_week ` - the date of the next Sunday from `original\_date`. If `original\_date` is already a Sunday, this field should be the same value
2. `end\_of\_month ` - the value of the end of month date

An acceptable solution is one which works for any valid date in the string format of `original\_date`.

Desired Result:

|  |  |  |
| --- | --- | --- |
| ORIGINAL\_DATE | END\_OF\_WEEK | END\_OF\_MONTH |
| 20190825 | **20190825** | **20190831** |
| 20190826 | **20190901** | **20190831** |
| 20190827 | **20190901** | **20190831** |
| 20190828 | **20190901** | **20190831** |
| 20190829 | **20190901** | **20190831** |
| 20190830 | **20190901** | **20190831** |
| 20190831 | **20190901** | **20190831** |
| 20190901 | **20190901** | **20190930** |

Additional Info:

20190825 is a Sunday, so the `end\_of\_week` for that value is still that same date.

20190827 is a Tuesday, and the next Sunday is on 20190901

**Notes:**

**Premise:**You have a table `Activity`, with fields `ID`, `start\_time`, and `end\_time`. These fields are all of datatype string.

|  |  |  |
| --- | --- | --- |
| ID | START\_TIME | END\_TIME |
| 100 | 10:00 | 12:00 |
| 100 | 10:15 | 12:30 |
| 100 | 12:15 | 12:45 |
| 100 | 13:00 | 14:00 |
| 200 | 10:15 | 10:30 |
| … | … | … |

**Question:**Write a (preferably hive) **SQL** query to create a new field `group\_id` which identifies records within each `ID` that have overlapping `start\_time` and `end\_time` intervals. An acceptable solution will have a unique `group\_id` for each `ID` and overlapping set of intervals.

Desired Example:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | START\_TIME | END\_TIME | GROUP\_ID |
| 100 | 10:00 | 12:00 | **1** |
| 100 | 10:15 | 12:30 | **1** |
| 100 | 12:15 | 12:45 | **1** |
| 100 | 13:00 | 14:00 | **2** |
| 200 | 10:15 | 10:30 | **3** |

Additional Info: For a given `ID`, if any of its intervals overlap then the corresponding records belong to the same group, and thus should have the same `group\_id`. A record *A* overlaps another record *B* when *A*’s `start\_time` and/or `end\_time` is between *B*’s `start\_time` and `end\_time`.

In the example, `ID` = 100 has four intervals. The first three overlap => the second record overlaps with the first (the `start\_time` of 10:15 is between the `start\_time` and `end\_time` of 10:00 to 12:00) and the third overlaps with the second (the `start\_time` of 12:15 is between the `start\_time` and `end\_time` of 10:15 to 12:30). Because of this, they all have the same `group\_id` of 1. The fourth interval for `ID` = 100 does not overlap any of the other intervals within that `ID`, and so it becomes its own group with a new `group\_id`. The last record has a completely different `ID` and so it starts a third group also with a new `group\_id`.

**Notes:**

**Premise:**

Tracy has a file that contains a list of actors and the movies in which they acted. She wants to know the top 10 actors from her list whom have acted in the most movies.

|  |  |
| --- | --- |
| ACTOR\_NAME | MOVIE\_NAME |
| Leonardo DiCaprio | The Revenant |
| Samuel L. Jackson | Pulp Fiction |
| Tom Cruise | Mission Impossible |
| Leonardo DiCaprio | The Great Gatsby |
| … | … |

**Question:**Write code (or pseudocode) in your **favorite programming language** to display the 10 actors appearing in the most movies, and the count of movies in which they have acted. If there are less than 10 actors in her list, display all of them.

Consider all scenarios - such as, if two actors have acted in the same number of movies, they will have the same rank.

**Notes:**

**Premise:**Adam is so good at playing arcade games that he will win at every game he plays. One fine day as he was walking on the street, he discovers an arcade store that pays real cash for every game that the player wins - however, the store will only pay out once per game. The store has 10 games for which they will pay winners, and each game has its own completion time and payout rate. Thrilled at the prospect of earning money for his talent, Adam walked into the store only to realize that the store closes in 2 hours (exactly 120 minutes). Knowing that he cannot play all the games in that time, he decides to pick the games that maximize his earnings

|  |  |  |
| --- | --- | --- |
| GAME | COMPLETION\_TIME (in minutes) | PAYOUT\_RATE |
| Pac-man | 75 | $250 |
| Speed Racer | 45 | $280 |
| Pump it Up | 30 | $150 |
| Space Invaders | 35 | $120 |
| Mario Bros | 30 | $200 |
| Mortal Kombat | 15 | $100 |
| Atari Breakout | 60 | $300 |
| Super Tetris | 90 | $350 |
| Star Wars | 20 | $110 |
| Street Fighter II | 10 | $90 |

**Question:**Write code (or pseudocode) in your **favorite programming language** to help Adam pick the sequence(s) of games that earn him the most money?.  
  
Then, assume you have a variable list of games and their payout rates. What is the best way to pick the games that earn you the most?

An acceptable solution is a workable solution that accounts for all the different scenarios in a variable list of games with their completion times and payout rates.  
  
**Notes:**