**A**

**1.**

**UPDATE** request

**SET** title = title || **' from '**|| start\_date|| **' to '** || end\_date;

**2.** For each request, find volunteers whose skill assignments match the

requesting skills. List these volunteers from those with the most matching

skills to those with the least (even 0 matching skills). Only consider

volunteers who applied to the request and have a valid application

**SELECT** *va*.volunteer\_id

**FROM** request *r*

**JOIN** volunteer\_application *va* **ON** *va*.request\_id = *r*.id

**LEFT** **JOIN** request\_skill *rs* **ON** *rs*.request\_id = *r*.id -- joins requests with skills, if there are no requested skills assigned, then it is assigned to be NULL

**LEFT** **JOIN** skill\_assignment *sa* **ON** *sa*.volunteer\_id = *va*.volunteer\_id **AND** *sa*.skill\_name = *rs*.skill\_name -- left join tries to include volunteers skills by matching the volunteer id. The sa.skill\_name will be kept as it is, if its matches with request skills, otherwise it will contain NULL values

**WHERE** va.is\_valid = TRUE -- condition to select only valid applications

**GROUP** **BY** *va*.volunteer\_id -- groups the rows by volutneer id

**ORDER** **BY** **COUNT**(*sa*.skill\_name) **DESC**; -- ordering from highest to lowest number of matching skills per volunteer. Since column of skill name contains null values whenever it does not match the requested skill, the function count (does not count NULL values) can be used to count the number of matching skills

**3.** For each request, show the missing number of volunteers needed per

skill (minimum needed of that skill). Assume a volunteer fulfills the need for all

the skills they possess.

**select** *rs*.request\_id, *rs*.skill\_name, (*rs*.min\_need - **COUNT**(**DISTINCT** *sa*.volunteer\_id)) **AS** *missing\_skills\_count*

-- reporting request id for each skill (thus each request id will have as many rows as there is the number of assigned skills to it)

-- reporting the number of people that are still needed

**from** request\_skill *rs*

**join** volunteer\_application *va* **on** *rs*.request\_id =*va*.request\_id -- joining volunteer applications with requests for skills

**join** skill\_assignment *sa* **on** *sa*.volunteer\_id = *va*.volunteer\_id -- joining volunteer skills by volunteer id

**where** *rs*.skill\_name =*sa*.skill\_name **OR** *rs*.skill\_name **IS** **null** -- selecting rows where there is a match between the skills, of the volunteer who applied, and the listed skill needed

**group** **by** *rs*.request\_id,*rs*.skill\_name, *rs*.min\_need -- group by request id

**HAVING**

(*rs*.min\_need - **COUNT**(**DISTINCT** *sa*.volunteer\_id)) > 0; -- filter to show only the requests and the related skills for which there are volunteers still missing

-- reducing the number of volunteers needed for the skills of the specific request by each distinct volunteer that applied and matches the skill

**4.** Sort requests and the beneficiaries who made them by the highest

number of priority (requestʼs priority value) and the closest 'register by date'.

**SELECT**

*r*.title , *b*."name"

**FROM** request *r*

**JOIN** beneficiary *b* **ON** *r*.beneficiary\_id = *b*.id -- Join request and beneficiary by matching the beneficiary id

**where** *r*.register\_by\_date > ***current\_date*** -- removing those requests that have already passed the registration deadline

**ORDER** **BY**

*r*.priority\_value **DESC**, -- Ordering by request's priority from the highest

*r*.register\_by\_date **ASC**; -- Sort by register by the closest register date - the most urgent

**5.**

For each volunteer, list requests that are within their volunteer range and

match at least 2 of their skills (also include requests that donʼt require any

skills)

**SELECT** *vr*.volunteer\_id, *rl*.request\_id,

**from** request *r*

**JOIN** request\_location *rl* **ON** *r*.id = *rl*.request\_id --joining request and range by id

**join** volunteer\_range *vr* **on** *vr*.city\_id = *rl*.city\_id -- joining volunteer range by city

**JOIN** skill\_assignment *sa* **ON** *vr*.volunteer\_id = *sa*.volunteer\_id -- joining volunteer skills by matching volunteer ids

**LEFT** **JOIN** request\_skill *rs* **ON** *rs*.request\_id = *rl*.request\_id -- joining requested skills by request id; left join ensures that if no request skills are for the given request, then the skills column is null

**and** *rs*.skill\_name = *sa*.skill\_name -- Adjusted LEFT JOIN condition, the matching is only if the requested and volunteer skills match

**GROUP** **BY** *vr*.volunteer\_id, *rl*.request\_id -- grouping

**HAVING**

**COUNT**(\*) >= 2 --for each volunteer and request, count those that have more than 2 rows (ie fulfilling the conditions of matching and range)

**6.**

#adding a new empty column in request

**ALTER** **TABLE** request **ADD** **COLUMN** normalized\_title **VARCHAR**(255);

#the column filled up with edited title to be same format as interest\_name (selecting string until ‘need’ - to remove the dates and include only info that is indicated in volunteers’ assignment of interest; first letter of word in caps and removing space between words)

**update** request

**SET** normalized\_title =

**REGEXP\_REPLACE**(

**INITCAP**(

**CASE**

**WHEN** **POSITION**(**'needed'** **IN** title) > 0 **THEN**

**LEFT**(title, **POSITION**(**'needed'** **IN** title) - 1)

**ELSE**

title

**end**),**'[^a-zA-Z0-9]+'**, **''**, **'g'**);

# For each volunteer, listing all the requests where the title matches their area

of interest and are still available to register.

**select** *r*.title

**from** interest\_assignment *ia*

**JOIN** request *r* **ON** *ia*.interest\_name **LIKE** *r*.***normalized\_title***

**WHERE** *r*.register\_by\_date > ***CURRENT\_DATE***;

**7.**

Listing the request ID and the volunteers who applied to them (name and

email) but are not within the location range of the request. Order volunteers

by readiness to travel.

**select** *r*.id, *v*."name",*v*.email -- listing request ID, volunteer name and email

**from** request *r* -- first table is request

**join** request\_location *rl* **on** *rl*.request\_id = *r*.id -- joining info on the location at which the request is needed, it can be at multiple distinct locations

**inner** **join** volunteer\_application *va* **on** *r*.id =*va*.request\_id -- joining request table and volunteer application only for rows where the request id is matching. One request has often many applications, even from the same person

**join** volunteer *v* **on** *v*.id = *va*.volunteer\_id -- joining volunteer table by matching volunteer id with volunteer application table. One volunteer, with specific name, email, and travel\_readiness, usually submits many applications to different requests

**join** volunteer\_range *vr* **on** *vr*.volunteer\_id = *va*.volunteer\_id -- adding information on the volunteer location

**where** *vr*.city\_id != *rl*.city\_id -- selecting only the rows where the volunteer and request locations do not match

**group** **by** *r*.id, *va*.request\_id, *va*.id, *v*."name",*v*.email, *v*.travel\_readiness -- Each volunteer can have many locations and each request can have many locations. Therefore, for each request, all the possible combinations, where these don’t match, are listed. To not list them separately by locations, we group them.

**order** **by** *v*.travel\_readiness ; -- ordering the list by travel\_readiness which is specific for each volunteer

**8.**

Order the skills overall (from all requests) in the most prioritized to least prioritized (average the importance value)

**SELECT** *rs*.skill\_name -- listing the skills in order

**FROM** request\_skill *rs* -- from table request\_skill, which already contains all the skills requested by beneficiaries

**GROUP** **BY** *rs*.skill\_name -- we group it by skills, because the skills are repated in many requests

**ORDER** **BY** **avg**(*rs*.value) **DESC**; -- and for each skill we calculate the average val

B. ADVANCED

a)

**2.** Create a view that lists beneficiaries from highest number of fully fitting applications for it (agreed on skills, interest, validity, application wasnt modified after deadline) that have passed the registration date

(the script assuming there has been the ‘normalized\_title’ created in request, in Query no.6, which defines the interest area in the same format as interest\_assignment)

**CREATE** **VIEW** Beneficiaries\_With\_most\_match **AS**

**SELECT** b.**"name"**, **count**(**distinct** va.id) **as** matching\_applications -- listing beneficiaries from the highest to lowest number of failed applications

**FROM** beneficiary b

**JOIN** request r **ON** b.id = r.beneficiary\_id -- joining beneficiaries and requests by matching beneficiary id

**JOIN** volunteer\_application va **ON** r.id = va.request\_id -- joining volunteer applications by matching request ids

**join** request\_skill rs **on** rs.request\_id = r.id -- joining requested skills by matching request ids

**join** skill\_assignment sa **on** sa.volunteer\_id = va.volunteer\_id -- joining volunteer skills by matching volunteer id

**join** interest\_assignment ia **on** ia.volunteer\_id = va.volunteer\_id

**WHERE** r.register\_by\_date < **current\_date**

**and** r.register\_by\_date >= va.modified

**and** sa.skill\_name =rs.skill\_name

**and** ia.interest\_name = r.normalized\_title

**and** va.is\_valid = TRUE-- selecting rows for the registration date has passed, the application wasnt modified after the deadline, the skills matched, the applicant's interests matched, the application was valid,

**GROUP** **BY** b.**name**

**ORDER** **BY** **count**(**distinct** va.id) **DESC**;

c) Transaction

1.

**CREATE** **TABLE** volunteer\_assigned (

assignment\_id SERIAL **PRIMARY** **KEY**,

request\_id **INT**,

volunteer\_id **INT**,

volunteer\_skill **text**,

requested\_skill **text**);

**BEGIN** **TRANSACTION**;

-- extracting the number of volunteers needed for the request and the registration deadline

**SELECT** r.number\_of\_volunteers, r.register\_by\_date

**INTO** total\_needed, deadline

**FROM** request r

**WHERE** r.id = **:request\_id**; -- dynamic assignment of the request id of interest

**WITH** prioritizing\_skills\_order -- prioritize skills by the value of importance; listing request id, skill name, minimum number of people needed

**AS** (**SELECT**

rs.request\_id,

rs.skill\_name,

rs.min\_need,

**ROW\_NUMBER**() **OVER** (**PARTITION** **BY** rs.request\_id **ORDER** **BY** rs.value **DESC**) **AS** skill\_priority\_request -- within request id ordering the skills by importance

**FROM** request\_skill rs),

applicant\_skills\_order -- matching volunteers with skills and requests and ordering by the skills importance

**AS** (**SELECT**

va.volunteer\_id,

va.request\_id,

sa.skill\_name,

**ROW\_NUMBER**() **OVER** (**PARTITION** **BY** rs.request\_id **ORDER** **BY** rs.value **DESC**) **AS** skill\_usefulness -- for each request, if the application is valid, the volunteers's skills are ordered by value of importance for the request

**FROM** volunteer\_application va

**JOIN** skill\_assignment sa **ON** va.volunteer\_id = sa.volunteer\_id

**JOIN** request\_skill rs **ON** va.request\_id = rs.request\_id **AND** sa.skill\_name = rs.skill\_name

**WHERE** va.is\_valid = **TRUE**),

chosen -- matching the volunteers with the requests in order of how well their skills meet the needs

**AS** (**SELECT**

aso.volunteer\_id,

pso.request\_id,

aso.skill\_name

**FROM** applicant\_skills\_order aso

**JOIN** prioritizing\_skills\_order pso **ON** aso.request\_id = pso.request\_id **AND** siao.skill\_name = siro.skill\_name

**WHERE** aso.skill\_usefulness <= pso.min\_need) -- the rows are combined for each request as long as the integer in the column 'skill\_usefulness' is lower or equal to the number of people needed for the skills

-- Insert the information from 'chosen' (assigned volunteers) into the previously created volunteer\_assigned table

**INSERT** **INTO** volunteer\_assigned (request\_id, volunteer\_id, volunteer\_skill, requested\_skill)

**SELECT** **CAST**(*c*.request\_id **AS** **INT**),

**CAST**(*c*.volunteer\_id **AS** **INT**),

*c*.***skill\_name***

**FROM** ***chosen*** *c*;

-- saving how many volunteers match the needed skills, from the data saved in the new table

**SELECT** **COUNT**(\*) **AS** volunteer\_count,

**INTO** volunteer\_count

**FROM** volunteer\_assigned

**WHERE** va.request\_id = **:request\_id**;

-- If the conditions are met, commit the transaction, else roll back

**IF** (volunteer\_count < total\_needed **AND** **CURRENT\_TIMESTAMP** < deadline) **THEN**

**ROLLBACK**;

**ELSE**

**COMMIT**;

**END** **IF**;

**END** **TRANSACTION**;

d)Analysis   
1. In python – file updated on teams