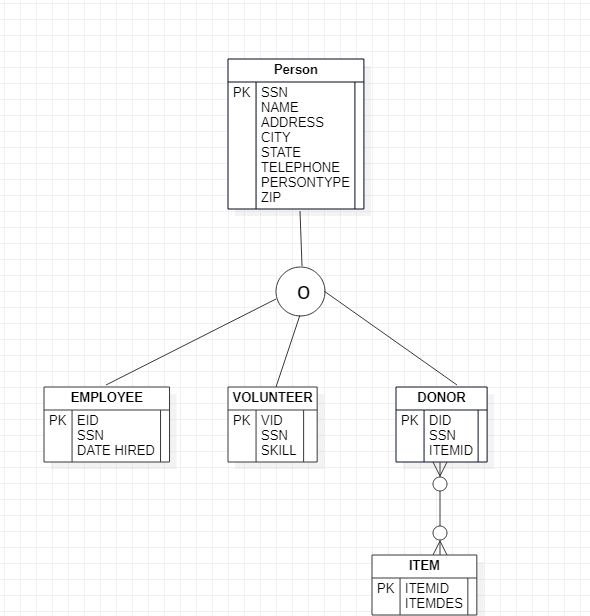
**CPSC6127 Contemporary Issues in Database Management Systems Spring 2022 Project1**

**(Harika Bhemidi)**

**Problem Statement:**

A nonprofit organization depends on a number of different types of persons for its successful operation. The organization is interested in the following attributes for all of these persons: SSN, Name, Address, City/State/Zip, and Telephone. Three types of persons are of greatest interest: employees, volunteers, and donors. Employees have only a Date Hired attribute, and volunteers have only a Skill attribute. Donors have only a relationship (named Donates) with an Item entity type. A donor must have donated one or more items, and an item may have no donors, or one or more donors. There are persons other than employees, volunteers, and donors who are of interest to the organization so that a person need not belong to any of these three groups. On the other hand, at a given time a person may belong to two or more of these groups (e.g., employee and donor).

1. Draw an EER diagram for the above problem using this text’s EER notation, the Visio notation, or the subtypes inside supertypes notation, as specified by your instructor to captures all requirements.



1. A full DDL statements that includes all primary keys, unique keys, foreign keys, and, check constraints.

CREATE TABLE project1\_harika\_db.PERSON

(

SSN numeric(10),

NAME varchar(50) NOT NULL,

ADDRESS varchar(50),

CITY varchar(50),

STATE varchar(50),

TELEPHONE numeric(10) UNIQUE,

PERSONTYPE CHAR(10),

ZIP numeric(38),

CONSTRAINT PKSSN PRIMARY KEY (SSN)

);

CREATE TABLE project1\_harika\_db.EMPLOYEE

(

EID numeric(10),

SSN numeric(10),

DATEHIRED DATE,

CONSTRAINT PKEID PRIMARY KEY (EID),

CONSTRAINT FKSSN FOREIGN KEY (SSN) REFERENCES PERSON(SSN)

);

CREATE TABLE project1\_harika\_db.VOLUNTEER

(

VID numeric(10),

SSN numeric(10),

SKILL varchar(50),

CONSTRAINT PKVID PRIMARY KEY (VID),

CONSTRAINT FKSSNV FOREIGN KEY (SSN) REFERENCES PERSON(SSN)

);

CREATE TABLE project1\_harika\_db.ITEM

(

ITEMID numeric(8),

ITEMDESC varchar(20),

CONSTRAINT PKID PRIMARY KEY (ITEMID)

);

CREATE TABLE project1\_harika\_db.DONOR

(

DID numeric(10),

SSN numeric(10),

ITEMID numeric(8),

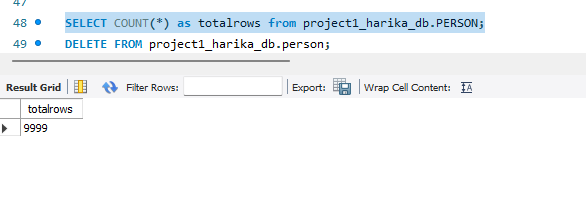
CONSTRAINT PKDID PRIMARY KEY (DID),

CONSTRAINT FKSSND FOREIGN KEY (SSN) REFERENCES PERSON(SSN),

CONSTRAINT FKIDD FOREIGN KEY (ITEMID) REFERENCES ITEM(ITEMID)

);

1. You should fill your tables with many rows (main transaction tables at least 100 records). It is allowed to use import data utilities(like excel ….etc), or writing a transcript insert statements to create random data.



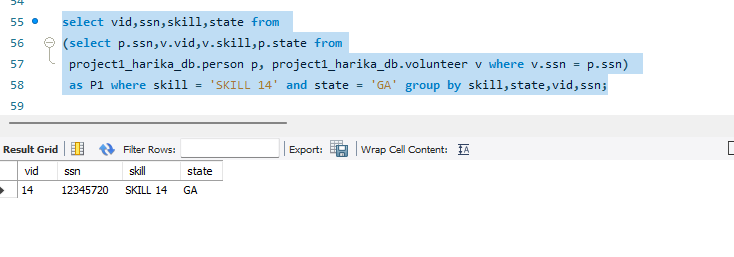
1. Writing 5 complex quires to retrieve main reports you find they important in the system.(The complex queries means it should include at least one of or more of the following: aggregate function, inline queries, correlated inner query, different types of joins, set operations etc).
2. Retrieve the details of Volunteer ID and their SSN details of volunteers who belongs to state AL with Skill level SKILL 14?

select vid,ssn,skill,state from

(select p.ssn,v.vid,v.skill,p.state from

project1\_harika\_db.person p, project1\_harika\_db.volunteer v where v.ssn = p.ssn)

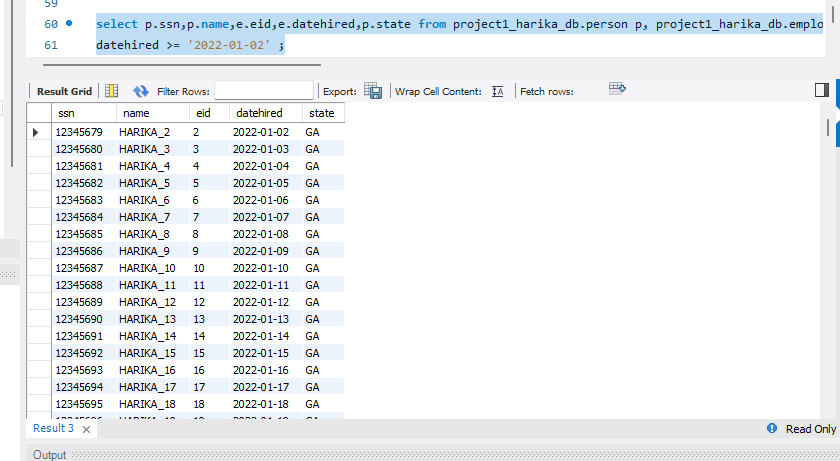
as P1 where skill = 'SKILL 14' and state = 'GA' group by skill,state,vid,ssn;



1. Retrieve the details of employee who hired on or after 02/01/2022?

select p.ssn,p.name,e.eid,e.datehired,p.state from project1\_harika\_db.person p, project1\_harika\_db.employee e where e.ssn = p.ssn and

datehired >= '2022-01-02' ;



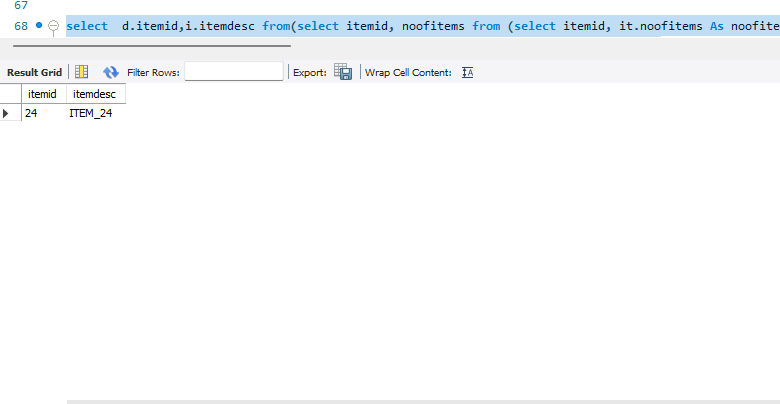
1. Retrieve the item details, that are donated mostly.

select d.itemid,i.itemdesc from(select itemid, noofitems from (select itemid, it.noofitems As noofitems, Row\_Number() Over (Order By noofitems desc) As RowNum from

(select itemid,count(\*) as noofitems from project1\_harika\_db.donor group by itemid

order by noofitems desc) as it) as it2 where RowNum<=1) as d inner join

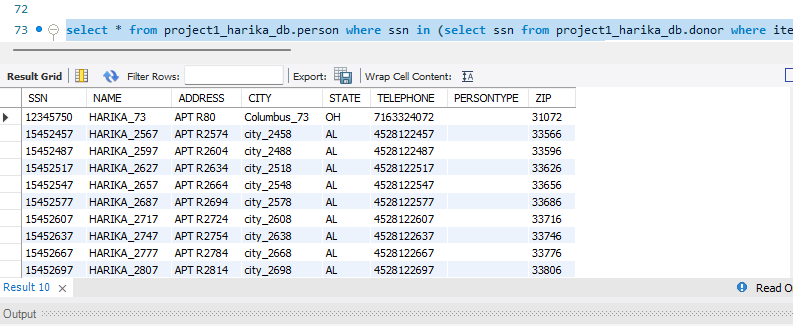
project1\_harika\_db.item i on d.itemid = i.itemid;



* Retrieve the donar details, who donated most number of times.

select \* from project1\_harika\_db.person where ssn in (select ssn from project1\_harika\_db.donor where itemid in (select itemid from (select itemid, Row\_Number() Over (Order By itemid desc) As RowNum from (select itemid,count(\*) from project1\_harika\_db.donor group by itemid order by count(\*) desc)

as t1 ) as t2 where RowNum<=1) );



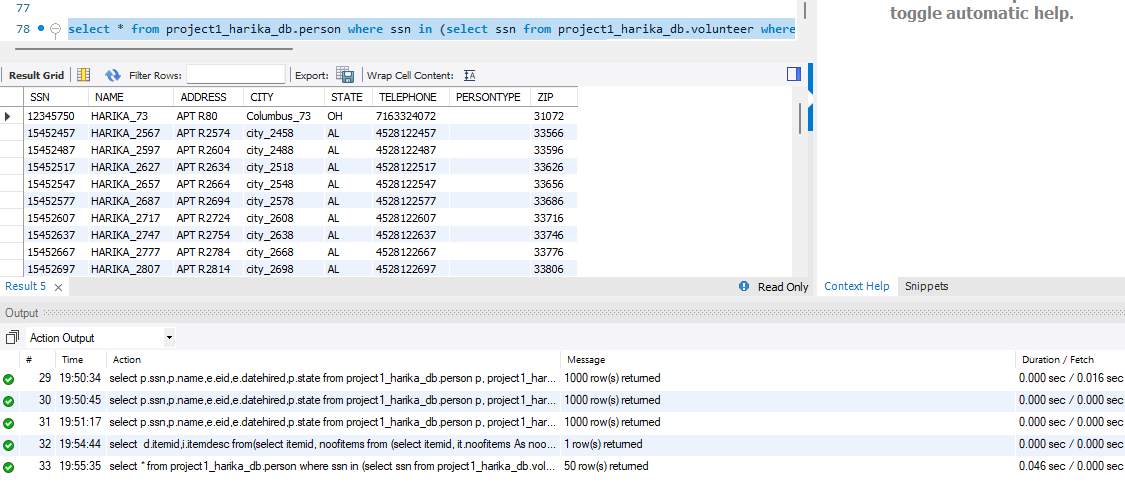
* Details of donors who are also volunteers who donated the most popular item?

select \* from project1db.person where ssn in (select ssn from project1db.volunteer where ssn in (select ssn from project1db.donor where itemid in

(select itemid from (select itemid, Row\_Number() Over (Order By itemid desc)

As RowNum from (select itemid,count(\*) from project1db.donor group by itemid order by count(\*) desc)

as t1 ) as t2 where RowNum<=1) ));



1. Analysis the performance of the suggested queries and apply different techniques to enhance quires like: indexing, De-normalization.

Index helps to speed data access. By creating an [index](https://docs.oracle.com/cd/E11882_01/server.112/e40540/glossary.htm#i432409) on one or more columns of a table, you gain the ability in some cases to retrieve a small set of randomly distributed rows from the table. Indexes are one of many means of reducing disk input out. As it consume more space on Database, we need think about it when we have restriction about storage.

Indexes can slow INSERT, UPDATE, and DELETE queries, because the index and the table need to be updated whenever these statements are run.

I have consider the below query for index analysis, ie

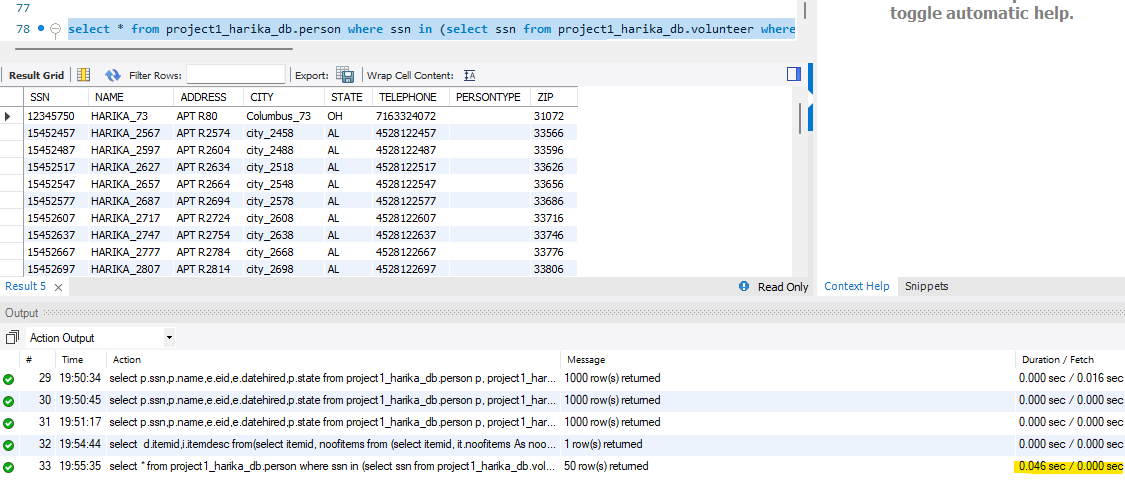
Retrieving the details of Donor who donated the most popular Item?

select \* from project1\_harika\_db.person where ssn in (select ssn from project1\_harika\_db.donor where itemid in (select itemid from (select itemid, Row\_Number() Over (Order By itemid desc) As RowNum from (select itemid,count(\*) from project1\_harika\_db.donor group by itemid order by count(\*) desc)

as t1 ) as t2 where RowNum<=1) );

* Before Indexing:

It took 0.045 sec to fetch the data

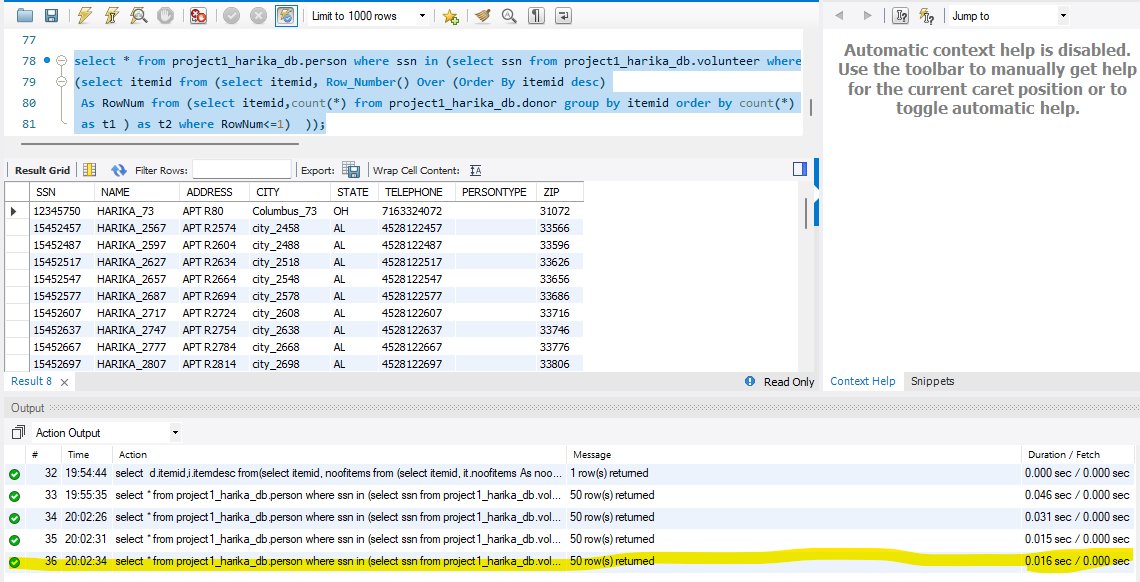


After Indexing:

create project1\_harika\_db.index inx\_per\_item on project1\_harika\_db.PERSON(ssn,telephone,city);

create project1\_harika\_db.index inx\_per\_item1 on project1\_harika\_db .item(itemid,itemdesc);

It reduced the processing time to 0.015 sec to fetch the data

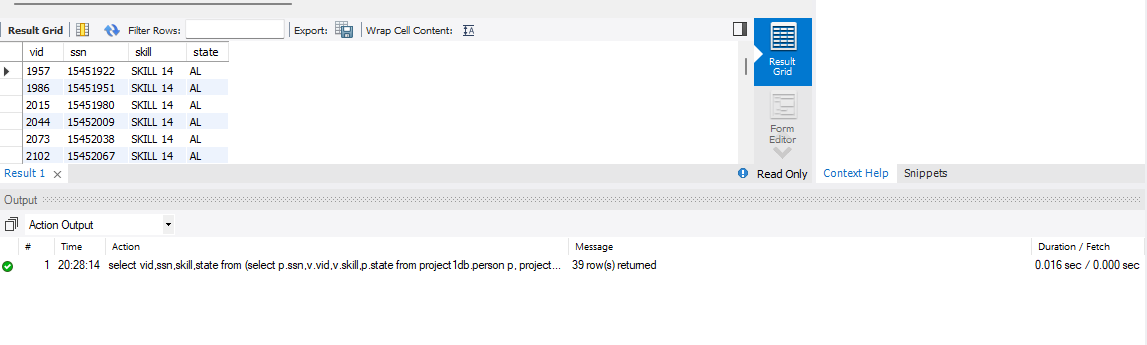


**Denormalization:**

Denormalization is used to increase performance. *Denormalization* is the process of reversing the transformations made during [normalization](https://mariadb.com/kb/en/database-normalization/) for performance reasons. Denormalization usually speeds retrieval but can slow updates. Denormalization is always application-specific and needs to be re-evaluated if the application changes.Denormalization can increase the size of tables.

**Before Denormalization:**

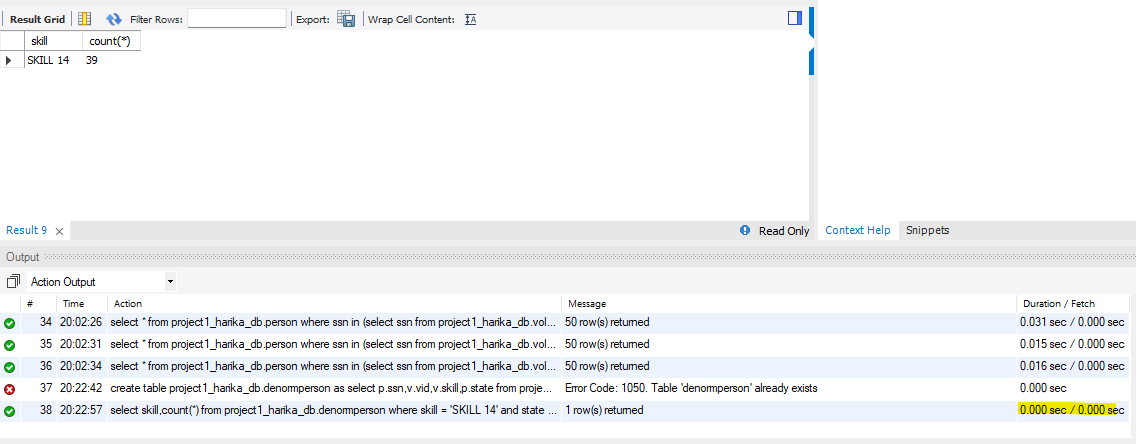
select vid,ssn,skill,state from (select p.ssn,v.vid,v.skill,p.state from project1db.person p, project1db.volunteer v where v.ssn = p.ssn) as P1 where skill = 'SKILL 14' and state = 'AL' group by skill,state,vid,ssn;



**After Denormalization:**

create table project1\_harika\_db.denormperson as select p.ssn,v.vid,v.skill,p.state from project1\_harika\_db.person p, project1\_harika\_db .volunteer v where v.ssn = p.ssn;

create index inx\_denorm on project1\_harika\_db.denormperson (SSN,VID,SKILL,STATE); select skill,count(\*) from project1\_harika\_db.denormperson where skill = 'SKILL 14' and state = 'AL' group by skill;



Re-measure performance after applying different techniques and criticize the drawbacks of each techniques and the obtained enhancement.

It shows it improves the processing time after indexing. Before indexing it is 0.045 Sec.

After indexing it reduced to 0.016 Sec.

It shows it improves the processing time after denormalization. Before denormalization it is 0.015 Sec.

After indexing it reduced to 0.0001 Sec.