

**DHAVAL GOGRI  
47444609**

**CSE 5330/7330 Fall 2017**

**Project Definition**

**Do your own work. Reference any material used.**

**(Alternative Projects will be considered by Dr. Moore. Please contact him with suggestions.)**

This is an individual project to be completed by each student. In addition, the project is to be completed in 3 phases. At the end of each phase, you are to submit that portion of the project for grading. At that time, you will receive information needed to complete the next phase of the project.

*You are to design and implement a database system to keep track of software builds performed by **Our Software Factory (OSF)**. Your user contact (and person to approach with any questions) is Dr. Moore.*

Any DBMS may be used for the project as long as a SQL command line interface is available for customer acceptance testing. The last phase of the project will require that Dr. Moore be able to test some unseen queries against your database, thus he must have access to your database. If you use the SMU implementation of Oracle or MySQL, this can easily be accomplished using appropriate security settings. If, however, you use another DBMS then you will have to provide instructions for its use or schedule a time for hands-on interaction.

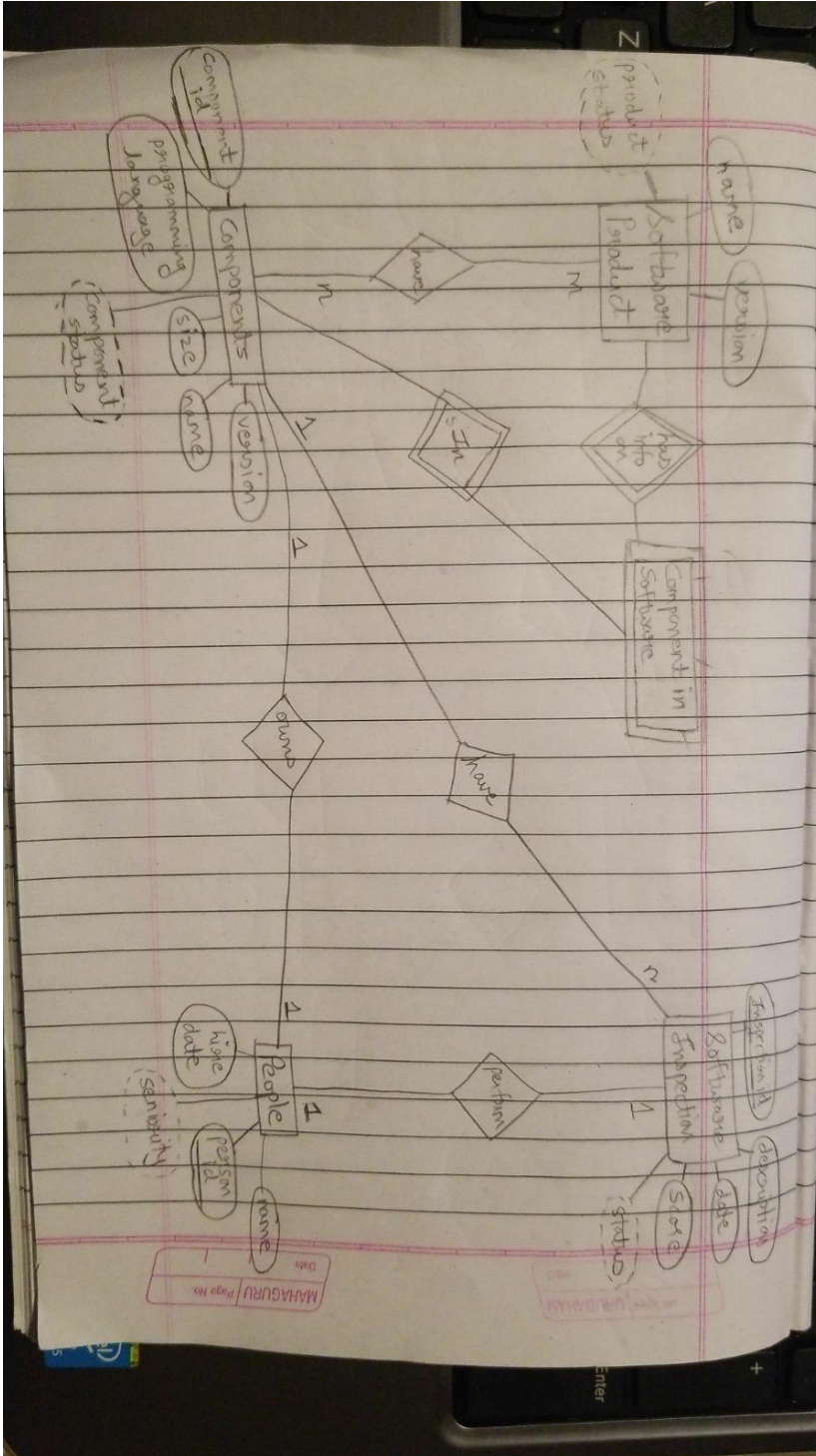
Our Software Factory develops various software products, each one requiring a software build. A software product is identified by its name and version. A software build identifies the components needed for a particular product. Components may be shared among products. The build status for a product is the lowest status of any of its components. Each component has a status of ready, usable, and not-ready. Information required about each component includes size, programming language (C, C++, C#, Java, or PHP), component name and 3 character version. Each component has one person identified as its owner. People have a unique ID (5 digit number), name (60 characters) and seniority. Components get their status as a result of a software inspection (peer review). An inspection event covers one component and results in an inspection score (0 .. 100). If the score is greater than 90, the component is considered “Ready”. If the score is less than 75, it is “not-ready”; otherwise it is “usable”. Inspection data includes what was inspected, the date of inspection, who conducted the inspection, inspection score, and textual description. The textual description can be updated later, but the score can never be changed. Components may be inspected multiple times.

The actual applications to be run against the database have not yet been determined by the user; however you have been asked to start the development of the database to get it ready. Your assignment is divided into three phases. At the end of each phase, you will be provided with any missing information needed to complete the next phase

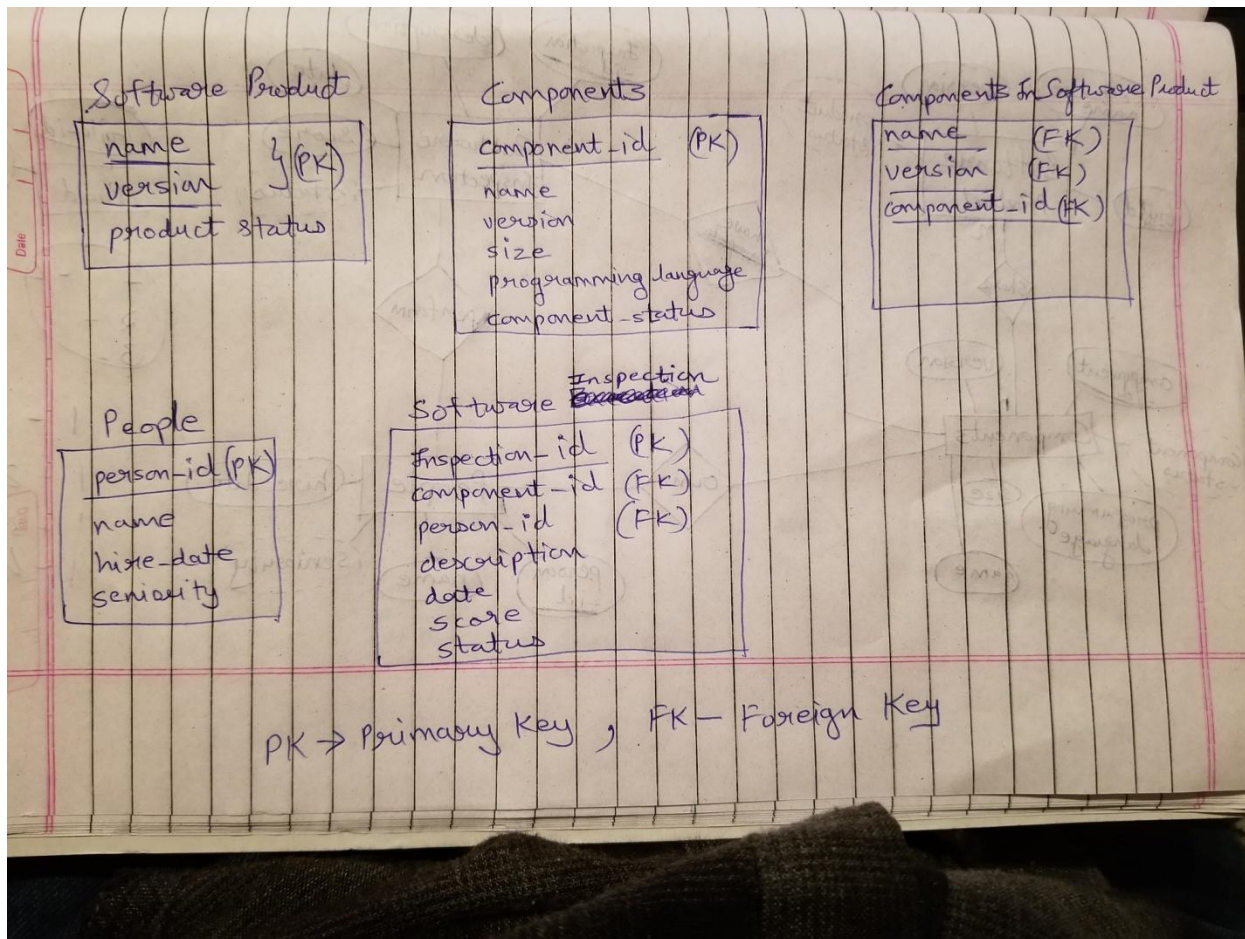
## Project Phases

### 1. ER Diagram and Initial Relational Design (20 pts; Due: 10/26):

- Construct an ER Diagram with attributes: The ER diagram you create must support all requirements stated above. If you add any restrictions or information not stated above, please explain. Indicate any design requirements that are not included in the ER diagram.



- Produce an initial Relational design: Given your ER diagram, provide an initial description of your relations, keys and foreign keys.



There would be 5 tables

1. Software Product - Contains Software Product information
2. Components - Contains Component Information
3. Components in Software Product - Contains Components information for a software product
4. People - People there in Our Software Factory
5. Software Inspection - Contains the score needed to give component its Working Status

Here, when a Software Inspection is completed and a score is given, a trigger is fired and changes are made to the component accordingly. When the status is changed for the component, all the Software Product associated with the component are also checked for updating of Product Status. The "Components in Software Product" table keeps tracks of all the components used in a particular software. When the "component\_status" changes a trigger would run and check in "Components in Software Product" for that component and associated software products to make any changes if necessary.

c. Submit for grading your ER diagram and relational schemas.

### 2.a Database Implementation (30 pts; Due: 11/16):

For phase 2 I have implemented the suggestions given by you during phase 1. I have made name and version as the primary key in Components. The table 'Components\_in\_software\_products' will have (name, version and comp\_id) as primary key.

- d. Using SQL DDL statements, create the relations as designed in phase 1. You must include any needed data constraints and keys (primary and foreign) to ensure design requirements are met.

```
create table programming_language
(
language_name varchar(20) primary key,
language_status ENUM('current', 'future') not null
)

create table software_products
(
name varchar(40),
version varchar(10),
software_status ENUM('Ready', 'not-ready', 'usable') not null default 'not-
ready',
primary key(name,version)
)

create table Employees
(
id int primary key,
name varchar(30),
hire_date timestamp,
mgr_id int,
seniority varchar(10)
)

create table Components
(
comp_id int auto_increment,
component_name varchar(40),
version varchar(10),
component_size int,
prog_language varchar(20),
comp_owner int,
component_status ENUM('Ready', 'not-ready', 'usable') not null default 'not-
ready',
primary key(component_name, version),
FOREIGN KEY (prog_language) REFERENCES programming_language(language_name),
FOREIGN KEY (comp_owner) REFERENCES Employees(id),
key (comp_id)
)

create table components_in_software_products
(
```

```

name varchar(40),
version varchar(10),
comp_id int,
primary key(name,version, comp_id),
FOREIGN KEY (name, version) REFERENCES software_products(name, version),
FOREIGN KEY (comp_id) REFERENCES Components(comp_id)
)

```

```

create table Inspection
(
inspection_id int primary key auto_increment,
component_name varchar(40),
version varchar(10),
inspection_date timestamp not null,
by_who int,
score int not null,
description varchar(4000),
status ENUM('Ready', 'not-ready', 'usable') not null default 'not-ready',
FOREIGN KEY (by_who) REFERENCES Employees(id),
FOREIGN KEY (component_name, version) REFERENCES Components(component_name,
version),
key (inspection_id)
)

```



- e. Populate the relations using data provided by the user.

```
-- Insert Programming languages
insert into programming_language values('C','current');
insert into programming_language values('C++','current');
insert into programming_language values('C#','current');
insert into programming_language values('Java','current');
insert into programming_language values('PHP','current');
insert into programming_language values('Python','Future');
insert into programming_language values('assembly','Future');

-- Insert Into Employees
insert into employees(id, name, hire_date, mgr_id) values(10100, 'Employee-1',
STR_TO_DATE( '08/11/1984', '%m/%d/%Y'), null);
insert into employees(id, name, hire_date, mgr_id) values(10200, 'Employee-2',
STR_TO_DATE( '08/11/1994', '%m/%d/%Y'),10100);
insert into employees(id, name, hire_date, mgr_id) values(10300, 'Employee-3',
STR_TO_DATE( '08/11/2004', '%m/%d/%Y'),10200);
insert into employees(id, name, hire_date, mgr_id) values(10400, 'Employee-4',
STR_TO_DATE( '01/11/2008', '%m/%d/%Y'),10200);
insert into employees(id, name, hire_date, mgr_id) values(10500, 'Employee-5',
STR_TO_DATE( '01/11/2015', '%m/%d/%Y'),10400);
insert into employees(id, name, hire_date, mgr_id) values(10600, 'Employee-6',
STR_TO_DATE( '01/11/2015', '%m/%d/%Y'),10400);
insert into employees(id, name, hire_date, mgr_id) values(10700, 'Employee-7',
STR_TO_DATE( '01/11/2016', '%m/%d/%Y'),10400);
insert into employees(id, name, hire_date, mgr_id) values(10800, 'Employee-8',
STR_TO_DATE( '01/11/2017', '%m/%d/%Y'),10200);

-- Insert into Components
insert into Components(comp_id, component_name, version, component_size,
prog_language, comp_owner) values(1, 'Keyboard Driver', 'K11', 1200, 'C', 10100);
insert into Components(comp_id, component_name, version, component_size,
prog_language, comp_owner) values(2, 'Touch Screen Driver', 'T00', 4000, 'C++',
10100);
insert into Components(comp_id, component_name, version, component_size,
prog_language, comp_owner) values(3, 'Dbase Interface', 'D00', 2500, 'C++',
10200);
insert into Components(comp_id, component_name, version, component_size,
prog_language, comp_owner) values(4, 'Dbase Interface', 'D01', 2500, 'C++',
10300);
insert into Components(comp_id, component_name, version, component_size,
prog_language, comp_owner) values(5, 'Chart generator', 'C11', 6500, 'java',
10200);
insert into Components(comp_id, component_name, version, component_size,
prog_language, comp_owner) values(6, 'Pen Driver', 'P01', 3575, 'C', 10700);
insert into Components(comp_id, component_name, version, component_size,
prog_language, comp_owner) values(7, 'Math unit', 'A01', 5000, 'C', 10200);
insert into Components(comp_id, component_name, version, component_size,
prog_language, comp_owner) values(8, 'Math unit', 'A02', 3500, 'Java', 10200);
```

```

-- Insert into Software Products
insert into software_products(name, version) values('Excel', '2010');
insert into software_products(name, version) values('Excel', '2015');
insert into software_products(name, version) values('Excel', '2018beta');
insert into software_products(name, version) values('Excel', 'secret');

-- Insert into Components in Software
insert into components_in_software_products values('Excel', '2010', 1);
insert into components_in_software_products values('Excel', '2010', 3);
insert into components_in_software_products values('Excel', '2015', 1);
insert into components_in_software_products values('Excel', '2015', 4);
insert into components_in_software_products values('Excel', '2015', 6);
insert into components_in_software_products values('Excel', '2018beta', 1);
insert into components_in_software_products values('Excel', '2018beta', 2);
insert into components_in_software_products values('Excel', '2018beta', 5);
insert into components_in_software_products values('Excel', 'secret', 1);
insert into components_in_software_products values('Excel', 'secret', 2);
insert into components_in_software_products values('Excel', 'secret', 5);
insert into components_in_software_products values('Excel', 'secret', 8);

-- insert into Inspection
insert into inspection(inspection_id, component_name, version, inspection_date,
by_who, score, description) values(1, 'Keyboard Driver', 'K11',
STR_TO_DATE('02/14/2010', '%m/%d/%Y'), 10100, 100, 'legacy code which is already
approved');
insert into inspection(inspection_id, component_name, version, inspection_date,
by_who, score, description) values(2, 'Touch Screen Driver', 'T00',
STR_TO_DATE('06/01/2017', '%m/%d/%Y'), 10200, 95, 'initial release ready for
usage');
insert into inspection(inspection_id, component_name, version, inspection_date,
by_who, score, description) values(3, 'Dbase Interface', 'D00',
STR_TO_DATE('02/22/2010', '%m/%d/%Y'), 10100, 55, 'too many hard coded
parameters, the software must be more maintainable and configurable because we
want to use this in other products. ');
insert into inspection(inspection_id, component_name, version, inspection_date,
by_who, score, description) values(4, 'Dbase Interface', 'D00',
STR_TO_DATE('02/24/2010', '%m/%d/%Y'), 10100, 78, 'improved, but only handles DB2
format');
insert into inspection(inspection_id, component_name, version, inspection_date,
by_who, score, description) values(5, 'Dbase Interface', 'D00',
STR_TO_DATE('02/26/2010', '%m/%d/%Y'), 10100, 95, 'Okay, handles DB3 format. ');
insert into inspection(inspection_id, component_name, version, inspection_date,
by_who, score, description) values(6, 'Dbase Interface', 'D00',
STR_TO_DATE('02/28/2010', '%m/%d/%Y'), 10100, 100, 'satisfied');
insert into inspection(inspection_id, component_name, version, inspection_date,
by_who, score, description) values(7, 'Dbase Interface', 'D01',
STR_TO_DATE('05/01/2011', '%m/%d/%Y'), 10200, 100, 'Okay ready for use');

```

```

insert into inspection(inspection_id, component_name, version, inspection_date,
by_who, score, description) values(8, 'Pen Driver', 'P01',
STR_TO_DATE('07/15/2017', '%m/%d/%Y'), 10300, 80, 'Okay ready for beta testing');
insert into inspection(inspection_id, component_name, version, inspection_date,
by_who, score, description) values(9, 'Math unit', 'A01',
STR_TO_DATE('06/10/2014', '%m/%d/%Y'), 10100, 90, 'almost ready');
insert into inspection(inspection_id, component_name, version, inspection_date,
by_who, score, description) values(10, 'Math unit', 'A02',
STR_TO_DATE('06/15/2014', '%m/%d/%Y'), 10100, 70, 'Accuracy problems!');
insert into inspection(inspection_id, component_name, version, inspection_date,
by_who, score, description) values(11, 'Math unit', 'A02',
STR_TO_DATE('06/30/2014', '%m/%d/%Y'), 10100, 100, 'Okay problems fixed');
insert into inspection(inspection_id, component_name, version, inspection_date,
by_who, score, description) values(12, 'Math unit', 'A02',
STR_TO_DATE('11/02/2016', '%m/%d/%Y'), 10700, 100, 're-review for new employee to
gain experience in the process.');
```



- f. Submit for grading proof of creation of the relations and their population. This could be output of 'Describe' and SELECT \* statements. Be sure to indicate your selection of DBMS and location of implementation.

describe programming\_language;

The screenshot shows a database management tool interface. The top section contains a list of SQL queries, with the last one, `describe programming_language;`, highlighted. Below the queries, the 'Result Grid' is displayed, showing the schema for the `programming_language` table. The grid has columns for Field, Type, Null, Key, Default, and Extra. The data is as follows:

Field	Type	Null	Key	Default	Extra
language name	varchar(20)	NO	PRI	NULL	
language status	enum('current','future')	NO		NULL	

At the bottom of the interface, there is an 'Output' section with a dropdown menu set to 'Action Output #'. The Windows taskbar is visible at the very bottom of the screen.

describe employees;

The screenshot shows a database management tool interface. The top section contains a list of SQL queries, with the following queries highlighted in blue:

- 416 • `select * from inspection;`
- 417 • `select * from components_in_software_products;`
- 418 • `select * from components;`
- 419 • `select * from employees;`
- 420 • `select * from software_products;`
- 421 • `select * from programming_language;`
- 422 • `describe inspection;`
- 424 • `describe components_in_software_products;`
- 425 • `describe components;`
- 426 • `describe employees;`
- 427 • `describe software_products;`
- 428 • `describe programming language;`

Below the queries, the "Result Grid" section displays the schema for the `employees` table:

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	
name	varchar(30)	YES		NULL	
hire_date	timestamp	NO		CURRENT_TIMESTAMP	on update CURRENT_TIMESTAMP
mar_id	int(11)	YES		NULL	
seniority	varchar(10)	YES		NULL	

The interface also includes a "Find" bar at the top, a "Limit to 1000 rows" dropdown, and a "Done" button. On the right side, there are buttons for "Result Grid", "Form Editor", and "Field Types". At the bottom, there is an "Output" section with a dropdown menu set to "Action Output".

describe components;

The screenshot shows a database management interface with a script editor and a table structure view. The script editor contains SQL commands for selecting and describing database tables. The table structure view displays the schema for a table with columns: comp id, component name, version, component size, prog language, comp owner, and component status.

**SQL Script:**

```
415  
416 • select * from inspection;  
417 • select * from components_in_software_products;  
418 • select * from components;  
419 • select * from employees;  
420 • select * from software_products;  
421 • select * from programming_language;  
422  
423 • describe inspection;  
424 • describe components_in_software_products;  
425 • describe components;  
426 • describe employees;  
427 • describe software_products;
```

**Table Structure:**

Field	Type	Null	Key	Default	Extra
comp id	int(11)	NO	MUL	NULL	auto increment
component name	varchar(40)	NO	PRI	NULL	
version	varchar(10)	NO	PRI	NULL	
component size	int(11)	YES		NULL	
prog language	varchar(20)	YES	MUL	NULL	
comp owner	int(11)	YES	MUL	NULL	
component status	enum('Readv','not-readv','usable')	NO			not-readv

describe inspection;

The screenshot shows a database management interface. At the top, a 'Find' bar is visible. Below it, a list of SQL queries is shown, with lines 413 to 425. The query on line 423, 'describe inspection;', is highlighted. Below the queries, a 'Result Grid' is displayed, showing the structure of the 'inspection' table. The grid has columns: Field, Type, Null, Key, Default, and Extra. The table structure is as follows:

Field	Type	Null	Key	Default	Extra
inspection id	int(11)	NO	PRI	<u>NULL</u>	auto increment
component name	varchar(40)	YES	MUL	<u>NULL</u>	
version	varchar(10)	YES		<u>NULL</u>	
inspection date	timestamp	NO		CURRENT_TIMESTAMP	on update CURRENT_TIMESTAMP
bv who	int(11)	YES	MUL	<u>NULL</u>	
score	int(11)	NO		<u>NULL</u>	
description	varchar(4000)	YES		<u>NULL</u>	
status	enum('Readv','not-readv','usable')	NO		not-readv	

Below the table, there is a 'Result 64' tab and a 'Read Only' indicator. At the bottom, there is an 'Output' section with a dropdown menu set to 'Action Output'. The Windows taskbar is visible at the very bottom of the screen.

describe software\_products;

The screenshot shows a database management interface with a SQL editor and a results pane. The SQL editor contains several queries, with the last one, `describe software_products;`, highlighted. The results pane displays the schema for the `software_products` table.

Field	Type	Null	Key	Default	Extra
name	varchar(40)	NO	PRI	NULL	
version	varchar(10)	NO	PRI	NULL	
software status	enum('Readv','not-readv','usable')	NO		not-readv	

Below the table, the results pane shows "Result 59" and "Output" sections. The "Output" section is currently empty, and the "Action Output" dropdown is set to "Action Output".

describe components\_in\_software\_products;

The screenshot shows a database management tool interface. At the top, there is a 'Find' bar. Below it, a list of SQL queries is displayed, with line numbers 414 through 426. The query 'describe components\_in\_software\_products;' is highlighted. Below the queries, there is a 'Result Grid' section. It includes a 'Filter Rows' input, an 'Export' button, and a 'Wrap Cell Content' checkbox. The 'Result Grid' table has columns: Field, Type, Null, Key, Default, and Extra. The table contains three rows of data:

Field	Type	Null	Key	Default	Extra
name	varchar(40)	NO	PRI	NULL	
version	varchar(10)	NO	PRI	NULL	
comp id	int(11)	NO	PRI	NULL	

On the right side of the interface, there are buttons for 'Result Grid', 'Form Editor', and 'Field Types'. At the bottom, there is a 'Read Only' status indicator and an 'Output' section with a dropdown menu set to 'Action Output'. The Windows taskbar is visible at the very bottom of the screen.



select \* from programming\_language;

The screenshot shows a database management interface. The top section is a SQL editor with a list of queries. The bottom section displays a result grid for the query 'select \* from programming\_language;'. The result grid has two columns: 'language\_name' and 'language\_status'. The data rows are as follows:

language_name	language_status
assemblv	future
C	current
C#	current
C++	current
Java	current
PHP	current
Pvthon	future
NULL	NULL

The interface also includes a 'Find' bar at the top, a 'Result Grid' button on the right, and a taskbar at the bottom with various application icons.

select \* from employees;

The screenshot shows a database management interface. The top section contains a list of SQL queries. The query 'select \* from employees;' is highlighted. Below the queries is a 'Result Grid' showing a table with 5 columns: id, name, hire\_date, mgr\_id, and seniority. The table contains 8 rows of employee data. The bottom section shows the 'Output' tab with 'Action Output' selected.

SQL Queries:

```
352 • delete from components_in_software_products;
353 • delete from components;
354 • delete from employees;
355 • delete from software_products;
356 • delete from programming_language;
357
358
359 • select * from inspection;
360 • select * from components_in_software_products;
361 • select * from components;
362 • select * from employees;
363 • select * from software_products;
364 • select * from programming_language;
365
366
367 -- Insert Programming Language
```

Result Grid:

id	name	hire_date	mgr_id	seniority
10100	Employee-1	1984-08-11 00:00:00	NULL	senior
10200	Employee-2	1994-08-11 00:00:00	10100	senior
10300	Employee-3	2004-08-11 00:00:00	10200	senior
10400	Employee-4	2008-01-11 00:00:00	10200	senior
10500	Employee-5	2015-01-11 00:00:00	10400	junior
10600	Employee-6	2015-01-11 00:00:00	10400	junior
10700	Employee-7	2016-01-11 00:00:00	10400	junior
10800	Employee-8	2017-01-11 00:00:00	10200	newbie
NULL	NULL	NULL	NULL	NULL

select \* from components;

The screenshot shows a database management interface. The top section contains a list of SQL queries, with the following queries visible:

```
352 • delete from components_in_software_products;
353 • delete from components;
354 • delete from employees;
355 • delete from software_products;
356 • delete from programming_language;
357
358
359 • select * from inspection;
360 • select * from components_in_software_products;
361 • select * from components;
362 • select * from employees;
363 • select * from software_products;
364 • select * from programming_language;
365
366
367
368 -- Insert Programming languages
```

The bottom section displays a "Result Grid" with the following data:

comp_id	component_name	version	component_size	prog_language	comp_owner	component_status
5	Chart oenerator	C11	6500	java	10200	not-readv
3	Dbase Interface	D00	2500	C++	10200	Readv
4	Dbase Interface	D01	2500	C++	10300	Readv
1	Keyboard Driver	K11	1200	C	10100	Readv
7	Math unit	A01	5000	C	10200	usable
8	Math unit	A02	3500	Java	10200	Readv
6	Pen Driver	P01	3575	C	10700	usable
2	Touch Screen Driver	T00	4000	C++	10100	Readv
NULL	NULL	NULL	NULL	NULL	NULL	NULL

The interface also includes a "Find" bar at the top, a "Filter Rows" section, and a "Result Grid" button on the right. The bottom status bar shows "components 52 x" and "Output".

select \* from inspection;

The screenshot shows a database management tool interface. At the top, a script editor displays a series of SQL commands. A tooltip above the script editor reads: "Execute the selected portion of the script or everything, if there is no selection". The script includes several 'delete from' statements followed by a 'select \* from inspection;' statement, which is currently selected. Below the script editor is a toolbar with options like 'Filter Rows', 'Edit', 'Export/Import', and 'Wrap Cell Content'. The main area is a 'Result Grid' showing a table with 8 columns: inspection\_id, component\_name, version, inspection\_date, by\_who, score, description, and status. The table contains 12 rows of data, with the last row showing NULL values. To the right of the grid are buttons for 'Result Grid', 'Form Editor', and 'Field Types'. At the bottom, there is an 'Output' section and a taskbar with various application icons.

Find  Done

Execute the selected portion of the script or everything, if there is no selection

```
352 • delete from components_in_software_products;
353 • delete from components;
354 • delete from employees;
355 • delete from software_products;
356 • delete from programming_language;
357
358
359 • select * from inspection;
360 • select * from components_in_software_products;
361 • select * from components;
362 • select * from employees;
363 • select * from software_products;
364 • select * from programming language;
```

Result Grid | Filter Rows:  | Edit: | Export/Import: | Wrap Cell Content:

inspection_id	component_name	version	inspection_date	by_who	score	description	status
1	Keyboard Driver	K11	2010-02-14 00:00:00	10100	100	legacy code which is already approved	Ready
2	Touch Screen Driver	T00	2017-06-01 00:00:00	10200	95	initial release ready for usage	Ready
3	Dbase Interface	D00	2010-02-22 00:00:00	10100	55	too many hard coded parameters. the software...	not-ready
4	Dbase Interface	D00	2010-02-24 00:00:00	10100	78	improved, but only handles DB2 format	usable
5	Dbase Interface	D00	2010-02-26 00:00:00	10100	95	Okay, handles DB3 format.	Ready
6	Dbase Interface	D00	2010-02-28 00:00:00	10100	100	satisfied	Ready
7	Dbase Interface	D01	2011-05-01 00:00:00	10200	100	Okay ready for use	Ready
8	Pen Driver	P01	2017-07-15 00:00:00	10300	80	Okay ready for beta testing	usable
9	Math unit	A01	2014-06-10 00:00:00	10100	90	almost ready	usable
10	Math unit	A02	2014-06-15 00:00:00	10100	70	Accuracy problems!	not-ready
11	Math unit	A02	2014-06-30 00:00:00	10100	100	Okay problems fixed	Ready
12	Math unit	A02	2016-11-02 00:00:00	10700	100	re-review for new employee to gain experience ...	Ready
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

inspection 55 x

Output

Action Output #

select \* from components\_in\_software\_products;

The screenshot shows a database management tool interface with two tabs: 'fodm\_project' and 'homework\_6'. The 'homework\_6' tab is active, displaying a list of SQL queries. The queries are as follows:

```
352 • delete from components_in_software_products;
353 • delete from components;
354 • delete from employees;
355 • delete from software_products;
356 • delete from programming_language;
357
358
359 • select * from inspection;
360 • select * from components_in_software_products;
361 • select * from components;
362 • select * from employees;
363 • select * from software_products;
364 • select * from programming_language;
```

Below the queries, there is a 'Result Grid' section. It includes a 'Filter Rows' field, an 'Edit' button, an 'Export/Import' button, and a 'Wrap Cell Content' checkbox. The result grid displays a table with the following data:

name	version	comp_id
Excel	2010	1
Excel	2015	1
Excel	2018beta	1
Excel	secret	1
Excel	2018beta	2
Excel	secret	2
Excel	2010	3
Excel	2015	4
Excel	2018beta	5
Excel	secret	5
Excel	2015	6
Excel	secret	8
NULL	NULL	NULL

At the bottom of the result grid, there is a tab labeled 'components\_in\_software\_produ...' and buttons for 'Apply' and 'Revert'. Below the result grid, there is an 'Output' section with a dropdown menu labeled 'Action Output'.

select \* from software\_products;

The screenshot shows a database management interface. The top section contains a list of SQL queries, with line numbers 352 through 368. The query at line 363, `select * from software_products;`, is highlighted. Below the queries is a toolbar with options like 'Filter Rows', 'Edit', 'Export/Import', and 'Wrap Cell Content'. The 'Result Grid' is visible, showing a table with three columns: 'name', 'version', and 'software\_status'. The table contains four rows of data, with the last row showing 'NULL' values. The bottom section of the interface shows the 'Output' pane, which is currently empty. The Windows taskbar is visible at the bottom of the screen.

name	version	software_status
Excel	2010	Readv
Excel	2015	usable
Excel	2018beta	not-readv
Excel	secret	not-readv
NULL	NULL	NULL



## 2.b Triggers (10 pts; Due: 11/16):

- g. Select one nontrivial trigger that is needed to ensure data requirements are met.

-- Trigger on Mgr Id

-- INSERT

This trigger checks for mgr\_id. For CEO i.e. for employee id 10100. Either he should be the manager himself or the field left null. For other employees the manager should be an existing employee of the company.

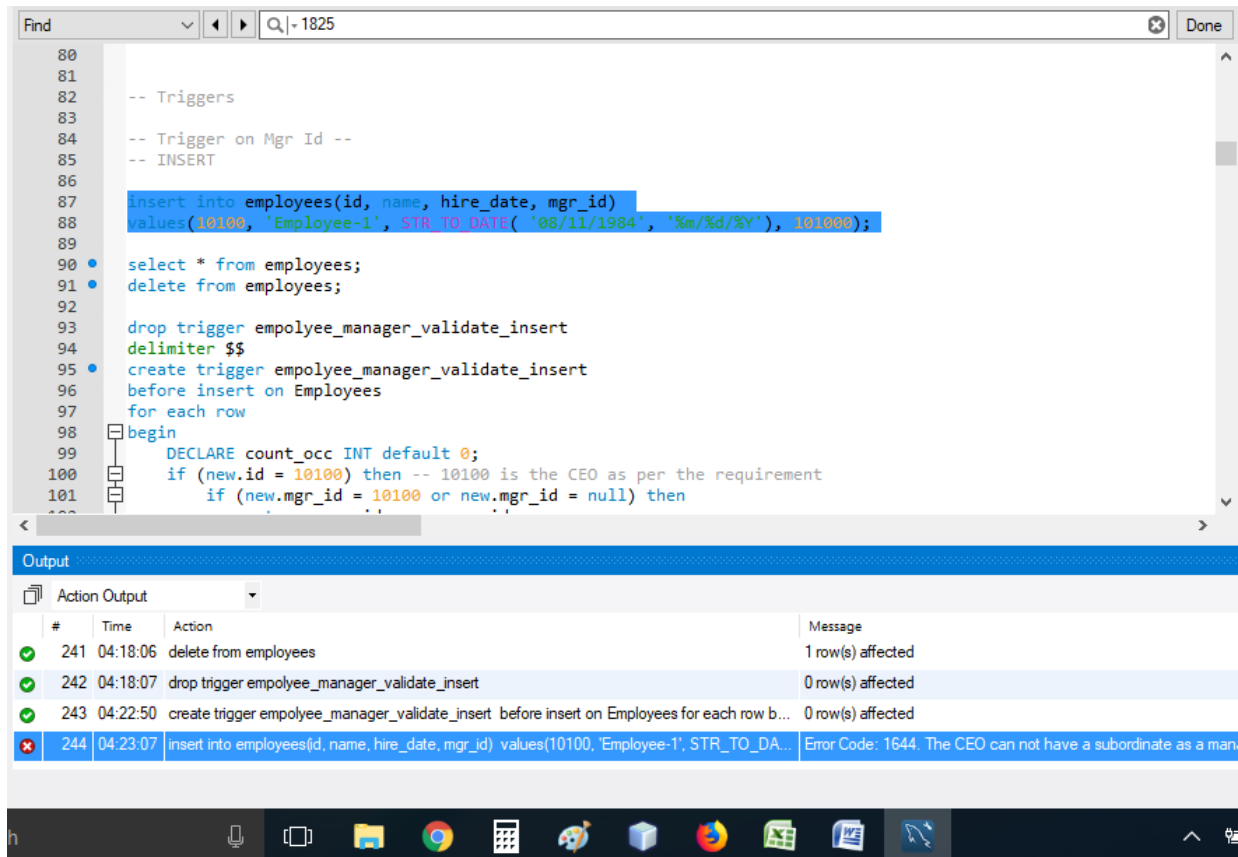
```
delimiter $$
create trigger empolyee_manager_validate_insert
before insert on Employees
for each row
begin
    DECLARE count_occ INT default 0;
    if (new.id = 10100) then -- 10100 is the CEO as per the requirement
        if (new.mgr_id = 10100 or new.mgr_id = null) then
            set new.mgr_id = new.mgr_id;
        else
            signal sqlstate '45000'
            set message_text = 'The CEO can not have a subordinate as a
manager or enter his own id or null as his manager.';
        end if;
    else
        if (new.id = new.mgr_id) then
            signal sqlstate '45000'
            set message_text = 'An employee cannot be his own
manager';
        end if;
        (select count(*) into @count_occ from Employees group by
Employees.id having Employees.id = new.id);
        if (count_occ = 0) then
            signal sqlstate '45000'
            set message_text = 'Manager should be an existing
employee';
        end if;
    end if;
end;
$$
delimiter ;
```

- h. Show the implementation of that trigger along with the results of your testing to confirm the trigger works as expected in an efficient manner. Continue to implement all required triggers.

This trigger checks for mgr\_id. For CEO i.e. for employee id 10100. Either he should be the manager himself or the field left null.

```
insert into employees(id, name, hire_date, mgr_id)
values(10100, 'Employee-1', STR_TO_DATE( '08/11/1984', '%m/%d/%Y'), 101000);
```

I enter the wrong manager id and trigger fires and doesn't allow the row to be added in the table.



```
80
81
82 -- Triggers
83
84 -- Trigger on Mgr Id --
85 -- INSERT
86
87 insert into employees(id, name, hire_date, mgr_id)
88 values(10100, 'Employee-1', STR_TO_DATE( '08/11/1984', '%m/%d/%Y'), 101000);
89
90 select * from employees;
91 delete from employees;
92
93 drop trigger empolyee_manager_validate_insert
94 delimiter $$
95 create trigger empolyee_manager_validate_insert
96 before insert on Employees
97 for each row
98 begin
99     DECLARE count_occ INT default 0;
100     if (new.id = 10100) then -- 10100 is the CEO as per the requirement
101         if (new.mgr_id = 10100 or new.mgr_id = null) then
```

Output

#	Time	Action	Message
241	04:18:06	delete from employees	1 row(s) affected
242	04:18:07	drop trigger empolyee_manager_validate_insert	0 row(s) affected
243	04:22:50	create trigger empolyee_manager_validate_insert before insert on Employees for each row b...	0 row(s) affected
244	04:23:07	insert into employees(id, name, hire_date, mgr_id) values(10100, 'Employee-1', STR_TO_DA...	Error Code: 1644. The CEO can not have a subordinate as a man...

I have also implemented several other triggers, event for updating seniority and procedures to support the triggers.

Below are the codes for those triggers, procedures and events.

```

-- Tiggers and Stored Procedure for Update in Components
-- Procedure to change the status of the component
Delimiter $$
CREATE PROCEDURE updateComponentsStatus (IN component_name varchar(20), IN
version varchar(10), IN status varchar(10))
BEGIN
    Declare id INT;
    update Components set Components.component_status = status where
Components.component_name = component_name and Components.version = version;
    set id = (select comp_id from Components where Components.component_name =
component_name and Components.version = version);
    CALL updateSoftwareProductStatus(id);
END $$
Delimiter ;

```

```

Delimiter $$
CREATE PROCEDURE updateSoftwareProductStatus (IN id int)
BEGIN

    DECLARE current_streak int;
    DECLARE rowcount int;
    DECLARE Name VARCHAR(40);
    DECLARE Version VARCHAR(10);
    DECLARE updateDone INT DEFAULT 0;
    DECLARE cur CURSOR FOR SELECT
components_in_software_products.name,components_in_software_products.version
FROM components_in_software_products where
components_in_software_products.comp_id = id;
    -- DECLARE EXIT HANDLER FOR NOT FOUND
    DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET updateDone = 1;

    set current_streak=0;
open cur;
    select FOUND_ROWS() into rowcount ;

start_loop: loop
    IF updateDone =1 THEN
        LEAVE start_loop;
    END IF;

    fetch cur into Name,Version;

    set current_streak = current_streak +1;
    if ((select count(*) from Components where
Components.component_status like 'not-ready' and Components.comp_id in (SELECT
components_in_software_products.comp_ID FROM components_in_software_products
where components_in_software_products.name = Name and
components_in_software_products.version = Version)) >0 ) then

```

```

        update software_products set software_products.software_status
= 'not-ready' where software_products.name = name and software_products.version =
version;

        else if ((select count(*) from Components where
Components.component_status like 'usable' and Components.comp_id in (SELECT
components_in_software_products.comp_ID FROM components_in_software_products
where components_in_software_products.name = Name and
components_in_software_products.version = Version)) >0 ) then
        update software_products set software_products.software_status
= 'usable' where software_products.name = name and software_products.version =
version;

        else
        update software_products set software_products.software_status
= 'ready' where software_products.name = name and software_products.version =
version;
        end if;
    end if;

    if (current_streak<=rowcount) then
        leave start_loop;
    end if;

end loop;
close cur;

END $$
Delimiter ;

```

```

-- Triggers on Status (Inspection) --> WORKING
-- INSERT

```

```

delimiter $$
create trigger inspection_status_insert
before insert on Inspection
for each row
begin
    if (new.score > 90 ) then
        set new.status = 'ready';
    else if (new.score < 75) then
        set new.status = 'not-ready';
    else
        set new.status = 'usable';
    end if;
end if;

```

```

        CALL updateComponentsStatus(new.component_name, new.version, new.status);
    end;
    $$
delimiter ;

```

```

-- Triggers on Status (Inspection) --> WORKING
-- UPDATE
delimiter $$
create trigger inspection_status_update
before update on Inspection
for each row
begin
    DECLARE score_value INT;
    SET score_value = (select score from Inspection where
Inspection.inspection_id = new.inspection_id);

    if (score_value != new.score) then
        signal sqlstate '45000'
        set message_text = 'Cannot update score';
    end if;
end;
$$
delimiter ;

```

```

-- Triggers for Employees
-- Seniority
SET GLOBAL event_scheduler = ON;
delimiter $$
CREATE EVENT seniority_update
ON SCHEDULE
EVERY 1 day
DO
BEGIN

    DECLARE current_streak int;
    DECLARE rowcount int;
    Declare hire_date timestamp;
    Declare id int;
    Declare date_diff int;
    DECLARE seniority_temp VARCHAR(10);
    DECLARE updateDone INT DEFAULT 0;
    DECLARE cur CURSOR FOR SELECT id, hire_date from employees;
    -- DECLARE EXIT HANDLER FOR NOT FOUND
    DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET updateDone = 1;

    set current_streak=0;
    open cur;
    select FOUND_ROWS() into rowcount ;

```

```

start_loop: loop
  IF updateDone =1 THEN
    LEAVE start_loop;
  END IF;

  fetch cur into id, hire_date;

  set current_streak = current_streak +1;
  set date_diff = ((UNIX_TIMESTAMP(current_date()) -
UNIX_TIMESTAMP(hire_date))/60/60/24);

  if (day_diff < 365) then
    update Employees set seniority = 'newbie' where Employees.id =
id;
  else if (day_diff > 365 and day_diff < 1825) then
    update Employees set seniority = 'junior' where Employees.id =
id;
  else if (day_diff > 1825) then
    update Employees set seniority = 'senior' where Employees.id =
id;
  end if;
  end if;
  end if;

  if (current_streak<=rowcount) then
    leave start_loop;
  end if;

end loop;
close cur;

END
$$
delimiter ;

```

```

-- Triggers on Employee Seniority --> WORKING
-- Insert

```

```

delimiter $$
create trigger employee_seniority_update
before insert on employees
for each row
begin
  DECLARE day_diff INT;
  set day_diff = ((UNIX_TIMESTAMP(current_date()) -
UNIX_TIMESTAMP(new.hire_date))/60/60/24);
  if (day_diff < 365) then
    set new.seniority = 'newbie';
  else if (day_diff > 365 and day_diff < 1825) then

```



```
        set new.seniority = 'junior';
    else if (day_diff > 1825) then
        set new.seniority = 'senior';
    end if;
end if;
end if;
end;
$$
delimiter ;
```

**3.a Applications (20 pts; Due: 12/6)**

- i. You will be provided a list of application requirements
- j. Using the relations populated in Phase 2, you are to create SQL code to implement a set of queries against the database.
- k. Submit for grading, proof of execution of these queries and documentation of any changes made to your implementation.

**3.b Final Testing (20 pts; Due: 12/6):**

- l. When you have submitted Phase 3.a for grading, Dr. Moore will perform his own testing of the database. Each student will have the same tests performed.

**NOTES:**

- 1. Requirements are subject to change at any time at the discretion of the user.
- 2. If you change details of an earlier phase implementation, please provide detail of this with your submission during the next phase.
- 3. You will receive a written grade on each phase while you are working on the next phase.