- **Project is built using MySQL on MySQL workbench**

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

a. 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.

b. Populate the relations using data provided by the user.

c. 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.

Tables =>

select * from software_products;

Result Grid | Filter Rows:

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

describe software_products;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

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

select * from Components;

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

comp_id	component_name	version	component_size	prog_language	comp_owner	comp_status
5	Chart generator	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

describe Components;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
comp_id	int(11)	NO	MUL	NULL	auto_increment
component_name	varchar(30)	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	
comp_status	enum('Readv','not-readv','usable')	NO		not-readv	

select * from programming_languages;

Result Grid | Filter Rows:

language_name	language_status
assembly	future
C	current
C#	current
C++	current
Java	current
PHP	current
Python	future
NULL	NULL

describe programming_languages;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
language_name	varchar(20)	NO	PRI	NULL	
language_status	enum('current','future')	YES		NULL	

select * from software_product_built;

391 describe softwar

Result Grid | Filter Rows:

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

describe software_product_built;

Result Grid | Filter Rows: | Export:

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

select * from employees;

Result Grid | Filter Rows: | Edit:

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

describe employees;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

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

select * from inspection;

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	not-ready
2	Touch Screen Driver	T00	2017-06-01 00:00:00	10200	95	initial release ready for usage	not-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	not-ready
5	Dbase Interface	D00	2010-02-26 00:00:00	10100	95	Okay. handles DB3 format.	not-ready
6	Dbase Interface	D00	2010-02-28 00:00:00	10100	100	satisfied	not-ready
7	Dbase Interface	D01	2011-05-01 00:00:00	10200	100	Okay ready for use	not-ready
8	Pen Driver	P01	2017-07-15 00:00:00	10300	80	Okay ready for beta testing	not-ready
9	Math unit	A01	2014-06-10 00:00:00	10100	90	almost ready	not-ready
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	not-ready
12	Math unit	A02	2016-11-02 00:00:00	10700	100	re-review for new employee to gain experience ...	not-ready
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

describe inspection;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
inspection_id	int(11)	NO	PRI	NULL	auto_increment
component_name	varchar(30)	YES	MUL	NULL	
version	varchar(10)	YES		NULL	
inspection_date	timestamp	NO		CURRENT_TIMESTAMP	on update CURRENT_TIMESTAMP
by_who	int(11)	YES	MUL	NULL	
score	int(11)	NO		NULL	
description	varchar(4000)	YES		NULL	
status	enum('Ready','not-ready','usable')	NO		not-ready	

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

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

Triggers & Procedures =>

```
Delimiter $$
CREATE PROCEDURE employeeManagerValidation(in mgr_id int,in emp_id int)
begin
        DECLARE count_occ INT;
    if (emp_id = 10100) then
                set mgr_id = mgr_id;
            if (mgr_id != 10100 or mgr_id != null) then
                        signal sqlstate '45000'
                        set message_text = 'The CEO can not have a subordinate as a manager or enter
his own id or null as his manager.';
            end if;
    else
                begin
                        if (emp_id = mgr_id) then
                                signal sqlstate '45000'
                                set message_text = 'An employee cannot be his own manager';
                end if;
                        SET count_occ = (select count(id) from Employees where Employees.id = mgr_id
group by Employees.id);
                if (count_occ = 0) then
                                signal sqlstate '45000'
                                set message_text = 'Manager should be an existing employee';
                end if;
                end;
    end if;
end;
$$
```

```
delimiter ;

delimiter $$
create trigger employee_manager_validation_insert
BEFORE INSERT on Employees
for each row
begin
	call employeeManagerValidation(mgr_id,new.emp_id);
end;
$$
delimiter ;


delimiter $$
create trigger employee_manager_validation_update
BEFORE Update on Employees
for each row
begin
	call employeeManagerValidation(mgr_id,new.emp_id);
end;
$$
delimiter ;
```

Result Grid | Filter Rows: | Edit:

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

```
Delimiter $$
CREATE PROCEDURE updateComponentsStatus (IN component_name varchar(30), IN version
varchar(10), IN status varchar(10),IN score int)
BEGIN

	DECLARE id int;
if (score >90 ) then
		set status = 'ready';
	else if (score < 75) then
		set status = 'not-ready';
```

```
        else
                set status = 'usable';
        end if;
    end if;
    update Components set Components.comp_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 trigger inspection_status_insert
after insert on Inspection
for each row
begin

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

delimiter $$
create trigger inspection_status_update
after update on Inspection
for each row
begin
        CALL updateComponentsStatus(new.component_name, new.version, new.status, new.score);
end;
$$
delimiter ;
```

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

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

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

```
BEGIN

        DECLARE current_streak int;
    DECLARE rowcount int;
        DECLARE Name VARCHAR(40);
    DECLARE Version VARCHAR(40);
    DECLARE updatedone int default 0;
        DECLARE cur CURSOR FOR SELECT
software_product_built.name,software_product_built.version FROM software_product_built where
comp_id = id;
    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.comp_status like 'not-ready'
and Components.comp_id in (SELECT software_product_built.comp_id    FROM software_product_built
where software_product_built.name = Name and software_product_built.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.comp_status like 'usable'
and Components.comp_id in (SELECT software_product_built.comp_id    FROM software_product_built
where software_product_built.name = Name and software_product_built.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 ;
```

Result Grid | Filter Rows:

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

```
SET GLOBAL event_scheduler = ON;

-- Triggers for Employees
-- Seniority
drop event seniority_update;
delimiter $$
CREATE EVENT seniority_update
ON SCHEDULE
EVERY 1 second
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 emp_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.emp_id = id;
                else if (day_diff > 365 and day_diff < 1825) then
                        update Employees set seniority = 'junior' where Employees.emp_id = id;
                else if (day_diff > 1825) then
                        update Employees set seniority = 'senior' where Employees.emp_id = id;
                end if;
                end if;
                end if;


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

        end loop;
        close cur;

END
$$
delimiter ;

select * from employees;



-- Triggers on Employee Seniority
-- Insert
drop trigger employee_seniority_insert;
delimiter $$
create trigger employee_seniority_insert
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 ;

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

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

a. 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.

before seniority trigger ==========➔

Result Grid | Filter Rows: | Edit:

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

After seniority trigger ==========➔

```
351 • truncate employees;
352 •        *      employees;
353   -- Insert Into Employees
354 • insert into employees(emp_id, name, hire_date, mgr_id) values(10100, 'Employee-1', STR_TO_DATE( '08/11/1984', '%m/%d/%Y'),
355 • insert into employees(emp_id, name, hire_date, mgr_id) values(10200, 'Employee-2', STR_TO_DATE( '08/11/1994', '%m/%d/%Y'),1
356 • insert into employees(emp_id, name, hire_date, mgr_id) values(10300, 'Employee-3', STR_TO_DATE( '08/11/2004', '%m/%d/%Y'),1
357 • insert into employees(emp_id, name, hire_date, mgr_id) values(10400, 'Employee-4', STR_TO_DATE( '01/11/2008', '%m/%d/%Y'),1
```

Result Grid

emp_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

employees 6

Output

Action Output

#	Time	Action	Message
1	07:19:30	insert into employees(emp_id, name, hire_date, mgr_id) values(10100, 'Employee-1', STR_TO_DATE('08/11/19...	1 row(s) affected
2	07:19:30	insert into employees(emp_id, name, hire_date, mgr_id) values(10200, 'Employee-2', STR_TO_DATE('08/11/19...	1 row(s) affected
3	07:19:30	insert into employees(emp_id, name, hire_date, mgr_id) values(10300, 'Employee-3', STR_TO_DATE('08/11/20...	1 row(s) affected
4	07:19:30	insert into employees(emp_id, name, hire_date, mgr_id) values(10400, 'Employee-4', STR_TO_DATE('01/11/20...	1 row(s) affected
5	07:19:30	insert into employees(emp_id, name, hire_date, mgr_id) values(10500, 'Employee-5', STR_TO_DATE('01/11/20...	1 row(s) affected
6	07:19:30	insert into employees(emp_id, name, hire_date, mgr_id) values(10600, 'Employee-6', STR_TO_DATE('01/11/20...	1 row(s) affected
7	07:19:30	insert into employees(emp_id, name, hire_date, mgr_id) values(10700, 'Employee-7', STR_TO_DATE('01/11/20...	1 row(s) affected
8	07:19:30	insert into employees(emp_id, name, hire_date, mgr_id) values(10800, 'Employee-8', STR_TO_DATE('01/11/20...	1 row(s) affected
9	07:19:48	select * from employees LIMIT 0, 1000	8 row(s) returned