Somya Singh CSE 7330 Fall 2017
Email: somyas@smu.edu
Student ID: 47304053 (Distance Education)

**Term Project**

Term Project Phase II

## Table of Contents



## Database Used

For this project, I am using sqlite3 to design my project. I am using SQLiteStudio (3.1.1) to work with sqlite.

Somya Singh CSE 7330 Fall 2017
Email: somyas@smu.edu
Student ID: 47304053 (Distance Education)

## Database Schema Definition

The tables used for my project implementation are;

1. Programming_language: The table is to store programming languages used for application

```
16
17 Pragma table_info(Programming_language);
```

Grid view | Form view

Total rows loaded: 2

| | cid | name | type | notnull | dflt_value | pk |
|---|---|---|---|---|---|---|
| 1 | 0 | Language_Status | Varchar (10) | 0 | NULL | 0 |
| 2 | 1 | Language | VARCHAR (30) | 0 | NULL | 1 |

```
19 select * from Programming_language;
20
```

Grid view | Form view

Total rows loaded: 7

| | Language_Status | Language |
|---|---|---|
| 1 | Current | C |
| 2 | Current | C++ |
| 3 | Current | C# |
| 4 | Current | Java |
| 5 | Current | PHP |
| 6 | Future | Python |
| 7 | Future | assembly |

   a. Columns
      i. Language_Status: Varchar (10) with values either current or future
      ii. Language: Varchar (30) containing language name

   b. Index
      i. idx_language: Index built on language.

Somya Singh CSE 7330 Fall 2017
Email: somyas@smu.edu
Student ID: 47304053 (Distance Education)

2. Employee: The table is used to store the employee details

```
5
6 Pragma table_info(Employee);
```

Grid view | Form view

Total rows loaded: 4

| | cid | name | type | notnull | dflt_value | pk |
|---|---|---|---|---|---|---|
| 1 | 0 | Employee_ID | INTEGER | 0 | NULL | 1 |
| 2 | 1 | Employee_Name | VARCHAR (60) | 1 | NULL | 0 |
| 3 | 2 | Hire_date | DATE | 1 | NULL | 0 |
| 4 | 3 | Manager_ID | INTEGER | 0 | NULL | 0 |

```
8 Select * from Employee;
```

Grid view | Form view

Total rows loaded: 8

| | Employee_ID | Employee_Name | Hire_date | Manager_ID |
|---|---|---|---|---|
| 1 | 10100 | Employee-1 | 1984-11-08 | NULL |
| 2 | 10200 | Employee-2 | 1994-11-08 | 10100 |
| 3 | 10300 | Employee-3 | 2004-11-08 | 10200 |
| 4 | 10400 | Employee-4 | 2008-11-01 | 10200 |
| 5 | 10500 | Employee-5 | 2015-11-01 | 10400 |
| 6 | 10600 | Employee-6 | 2015-11-01 | 10400 |
| 7 | 10700 | Employee-7 | 2016-11-01 | 10400 |
| 8 | 10800 | Employee-8 | 2017-11-01 | 10200 |

   a. Columns
      i. Employee_ID: Integer type. Its defined as Primary Key. It is used to store Employee ID. It will be unique for each employee.
      ii. Employee_Name: Varchar (60) not null. Used to store employee name.

iii. Hire_date: Date format not null. To store employee hire date. This is used to calculate Seniority.
iv. Manager_ID: Integer type. To store Manager id for the employee. This is assigned as Foreign Key referencing itself to Employee_ID. This is named as "fk_employee_mgrid" with On Update set as "Cascade" and On delete set as "Set NULL".

b. Index
   i. idx_Employee_ID: Unique index using "Employee_ID".

c. Triggers
   i. trig_insert_hire_date / trig_update_hire_date: Since sqlite only takes a date in limited format I created these triggers to force user only enter or update hire date in "YYYY-MM-DD" format. If user insert or update the hire date in any other format user will be thrown an error to enter date in correct format.
   ii. trig_insert_hire_date_in_past / trig_update_hire_date_in_past: Since hire date can't be a future date this trigger was created to limit user entering a hire date only from present or past.

3. Component: The table is to store the component details

```
19 Pragma table_info(Component);
```

Grid view | Form view

Total rows loaded: 7

| | cid | name | type | notnull | dflt_value | pk |
|---|---|---|---|---|---|---|
| 1 | 0 | Component_ID | integer | 0 | NULL | 1 |
| 2 | 1 | Component_Name | varchar (30) | 1 | NULL | 0 |
| 3 | 2 | Component_Version | varchar (20) | 1 | NULL | 0 |
| 4 | 3 | Component_Size | integer | 1 | NULL | 0 |
| 5 | 4 | Component_Language | VARCHAR (30) | 0 | NULL | 0 |
| 6 | 5 | Owner | integer | 0 | NULL | 0 |
| 7 | 6 | Component_Status | Varchar (20) | 0 | 'Not-Ready' | 0 |

Somya Singh CSE 7330 Fall 2017
Email: somyas@smu.edu
Student ID: 47304053 (Distance Education)

```
17 select * from Component;
18
19 Pragma table_info(Component);
```

Grid view | Form view

Total rows loaded: 8

| | Component_ID | Component_Name | Component_Version | Component_Size | Component_Language | Owner | Component_Status |
|---|---|---|---|---|---|---|---|
| 1 | 1 | Keyboard Driver | K11 | 1200 | C | 10100 | Ready |
| 2 | 2 | Touch Screen Driver | T00 | 4000 | C++ | 10100 | Ready |
| 3 | 3 | Dbase Interface | D00 | 2500 | C++ | 10200 | Ready |
| 4 | 4 | Dbase Interface | D01 | 2500 | C++ | 10300 | Ready |
| 5 | 5 | Chart generator | C11 | 6500 | Java | 10200 | Not-Ready |
| 6 | 6 | Pen driver | P01 | 3575 | C | 10700 | Usable |
| 7 | 7 | Math unit | A01 | 5000 | C | 10200 | Usable |
| 8 | 8 | Math unit | A02 | 3500 | Java | 10200 | Ready |

a. Columns
   i. Component_ID: integer type and primary key. Its set to auto-increment.
   ii. Component_Name: varchar (30) and not null. This is used to store the component name
   iii. Component_Version: varchar (20) and not null. It's used to store component version.
   iv. Component_Size: integer and not null. It's used to store size of the component
   v. Component_language: varchar (30) Foreign key referencing Programming_language (Language)
   vi. Owner: integer Foreign Key Referencing Employee (Employee_ID)
   vii. Component_Status: varchar (20) default value "Not-Ready". This is derived value from Inspection table score.

b. Index
   i. Idx_component: Unique index using Component_name and Componet_version.

c. Triggers
   i. trig_delete_build: This is to ensure that every product inserted has at-least one software build and component. This will ensure that there is one component tied to a product in build table.
   ii. trig_insert_ComponentStatus / trig_update_ComponentStatus: The trigger is to ensure that the status if the column "Component_Status" is only one of "Ready", "Usable" and "Not-Ready"

Somya Singh CSE 7330 Fall 2017
Email: somyas@smu.edu
Student ID: 47304053 (Distance Education)

4. Software_Product: To store product information.

```
19 Pragma table_info(Software_Product);
20
21
```

Grid view | Form view

Total rows loaded: 4

| | cid | name | type | notnull | dflt_value | pk |
|---|---|---|---|---|---|---|
| 1 | 0 | Software_ID | INTEGER | 0 | *NULL* | 1 |
| 2 | 1 | Software_Name | VARCHAR (20) | 1 | *NULL* | 0 |
| 3 | 2 | Software_Version | VARCHAR (20) | 1 | *NULL* | 0 |
| 4 | 3 | Product_Status | VARCHAR (20) | 0 | 'Not-Ready' | 0 |

```
17 select * from Software_Product;
18
```

Grid view | Form view

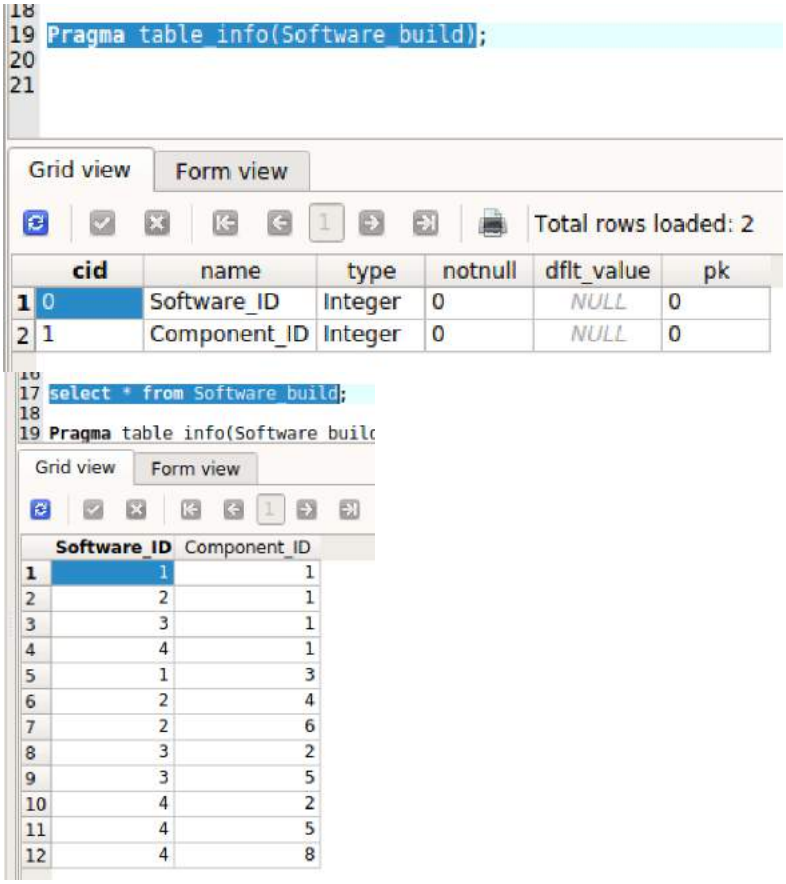
Total rows loaded: 4

| | Software_ID | Software_Name | Software_Version | Product_Status |
|---|---|---|---|---|
| 1 | 1 | Excel | 2010 | Ready |
| 2 | 2 | Excel | 2015 | Usable |
| 3 | 3 | Excel | 2018beta | Not-Ready |
| 4 | 4 | Excel | secret | Not-Ready |

   a. Colums
      i. Software_ID: integer and primary key. It's set to auto-increment. This is unique identifier for Product.
      ii. Software_Name: varchar (20) and not null. It's used to store name of the software product. It's also the partial key with software_version.
      iii. Software_version: varchar (20) and not null. It's used to record software version. It's the second half of the partial key with Software_name.
      iv. Product_status: varchar (20 and default value set to "Not-Ready". This is derived from Inspection and Component table.

b. Triggers
   i. trig_insert_ProductStatus / trig_update_ProductStatus: The trigger is created to ensure the status of product inserted is one of “Ready”, “Usable” and “Not-Ready”
   ii. trig_insert_Software_Product: The trigger is to ensure that every Software product has a build associated with it.

5. Software_Build: To store build information for each product and corresponding component.




a. Columns
   i. Software_ID: integer Foreign Key Referencing Software_Product (Software_ID).
   ii. Component_ID: integer Foreign key referencing Component (Component_ID)

b. Index

i. Idx_softbuild: the index is unique to identify using both software_id and component_id.

6. Inspection: The table stores inspection for component

```
19 Pragma table_info(Inspection);
20
```

Grid view | Form view

Total rows loaded: 5

| | cid | name | type | notnull | dflt_value | pk |
|---|---|---|---|---|---|---|
| 1 | 0 | Inspected_Component | INTEGER | 1 | NULL | 3 |
| 2 | 1 | Inspection_date | DATE | 1 | NULL | 1 |
| 3 | 2 | Own_BY | INTEGER | 1 | NULL | 2 |
| 4 | 3 | Score | INTEGER | 1 | NULL | 0 |
| 5 | 4 | Description | TEXT | 0 | NULL | 0 |

```
17 select * from Inspection;
18
19 Pragma table_info(Inspection);
20
```

Grid view | Form view

Total rows loaded: 12

| | Inspected_Component | Inspection_date | Own_BY | Score | Description |
|---|---|---|---|---|---|
| 1 | 1 | 2010-02-14 | 10100 | 100 | legacy code which is already approved |
| 2 | 2 | 2017-06-01 | 10200 | 95 | initial release ready for usage |
| 3 | 3 | 2010-02-22 | 10100 | 55 | too many hard coded parameters, the software must be more maintainable and configurable be... |
| 4 | 3 | 2010-02-24 | 10100 | 78 | improved, but only handles DB2 format |
| 5 | 3 | 2010-02-26 | 10100 | 95 | Okay, handles DB3 format. |
| 6 | 4 | 2011-05-01 | 10200 | 100 | Okay ready for use |
| 7 | 7 | 2014-06-10 | 10100 | 90 | almost ready |
| 8 | 8 | 2014-06-15 | 10100 | 70 | Accuracy problems! |
| 9 | 8 | 2014-06-30 | 10100 | 100 | Okay problems fixed |
| 10 | 8 | 2016-11-02 | 10700 | 100 | re-review for new employee to gain experience in the process. |
| 11 | 3 | 2010-02-28 | 10100 | 100 | satisifed |
| 12 | 6 | 2017-07-15 | 10300 | 80 | Okay ready for beta testing |

a. Columns
   i. Inspected_Component: Integer to store the component id that needs to be inspected. It's not null and referencing component (component_id). It's set to Cascade on delete.
   ii. Inspection_date: date to store inspection date. It's part of primary key for table.
   iii. Own_by: Integer to store Owner id. The column reference to Employee table (Employee_id). It's also set to cascade on delete for Foreign key. It's the third component of Primary key for table.

iv. Score: Integer to store the score of inspection. The value can only be between 0 and 100 and cannot be null.
v. Description: Text to store comments about inspection.

b. Trigger
   i. trig_update_inspection: the trigger is to ensure the user can only update description for a row. No other details for the inspection can be edited once inserted.
   ii. trig_insert_ProdStatus_NotReady / trig_insert_ProdStatus_Ready / trig_insert_ProdStatus_Usable: This trigger is to update the Product status based on inspection. This is using the view "SoftwareProduct_status_vw".
   iii. trig_insert_CompStatus_NotReady / trig_insert_CompStatus_Ready / trig_insert_CompStatus_Usable: This trigger is to update the Component status based on inspection. This is using the view "SoftwareComponent_status_vw".
   iv. trig_delete_ProdStatus_NotReady / trig_delete_ProdStatus_Ready / trig_delete_ProdStatus_Usable: This trigger is to update the Product status if any inspection is deleted. This is using the view "SoftwareProduct_status_vw".
   v. trig_delete_CompStatus_NotReady / trig_delete_CompStatus_Ready / trig_delete_CompStatus_Usable: This trigger is to update the Component status if any inspection is deleted. This is using the view "SoftwareComponent_status_vw".

7. SoftwareComponent_status_vw: This is a view which maintains the component final status based on inspection.

```
18
19 Pragma table_info(SoftwareComponent_status_vw);
20
```

Grid view | Form view

Total rows loaded: 2

| | cid | name | type | notnull | dflt_value | pk |
|---|---|---|---|---|---|---|
| 1 | 0 | ComponentID | integer | 0 | NULL | 0 |
| 2 | 1 | FinalScore | | 0 | NULL | 0 |

Email: somyas@smu.edu
Student ID: 47304053 (Distance Education)

```
select * from SoftwareComponent_status_vw;

Pragma table_info(SoftwareComponent_status_vw);
```

Grid view | Form view

Total rows loaded: 8

| | ComponentID | FinalScore |
|---|---|---|
| 1 | 1 | 100 |
| 2 | 2 | 95 |
| 3 | 3 | 100 |
| 4 | 4 | 100 |
| 5 | 5 | 0 |
| 6 | 6 | 80 |
| 7 | 7 | 90 |
| 8 | 8 | 100 |

Select comp.Component_ID AS ComponentID, MIN(ifNULL(insp.Score,0)) AS FinalScore
from Component comp
Left Join Inspection insp ON (insp.Inspected_Component = comp.Component_ID AND insp.Inspection_date IN (select MAX(Inspection_date) from Inspection group by Inspected_Component))
GROUP BY comp.Component_ID
ORDER BY comp.Component_ID

8. SoftwareProduct_status_vw: This is a view which maintains the product final status based on inspection and component.

```
Pragma table_info(SoftwareProduct_status_vw);
```

Grid view | Form view

Total rows loaded: 3

| | cid | name | type | notnull | dflt_value | pk |
|---|---|---|---|---|---|---|
| 1 | 0 | SoftwareID | Integer | 0 | NULL | 0 |
| 2 | 1 | ComponentID | Integer | 0 | NULL | 0 |
| 3 | 2 | FinalScore | | 0 | NULL | 0 |

Email: somyas@smu.edu
Student ID: 47304053 (Distance Education)

```
17 select * from SoftwareProduct_status_vw;
18
19 Pragma table_info(SoftwareProduct_status_vw);
20
21
```

Grid view | Form view

Total rows lo

| | SoftwareID | ComponentID | FinalScore |
|---|---|---|---|
| 1 | 1 | 1 | 100 |
| 2 | 2 | 6 | 80 |
| 3 | 3 | 5 | 0 |
| 4 | 4 | 5 | 0 |

Select swb.Software_ID AS SoftwareID,swb.Component_ID AS ComponentID,
MIN(ifNULL(insp.Score,0)) AS FinalScore
from Component comp
Join Software_Build swb ON swb.Component_ID = comp.Component_ID
Left Join Inspection insp ON (insp.Inspected_Component = comp.Component_ID AND
insp.Inspection_date IN (select MAX(Inspection_date) from Inspection group by
Inspected_Component))
GROUP BY swb.Software_ID
ORDER BY swb.Software_ID

9. Employee_Seniority_VW: This is a view which Calculates the Seniority of the Employee based on hire date.

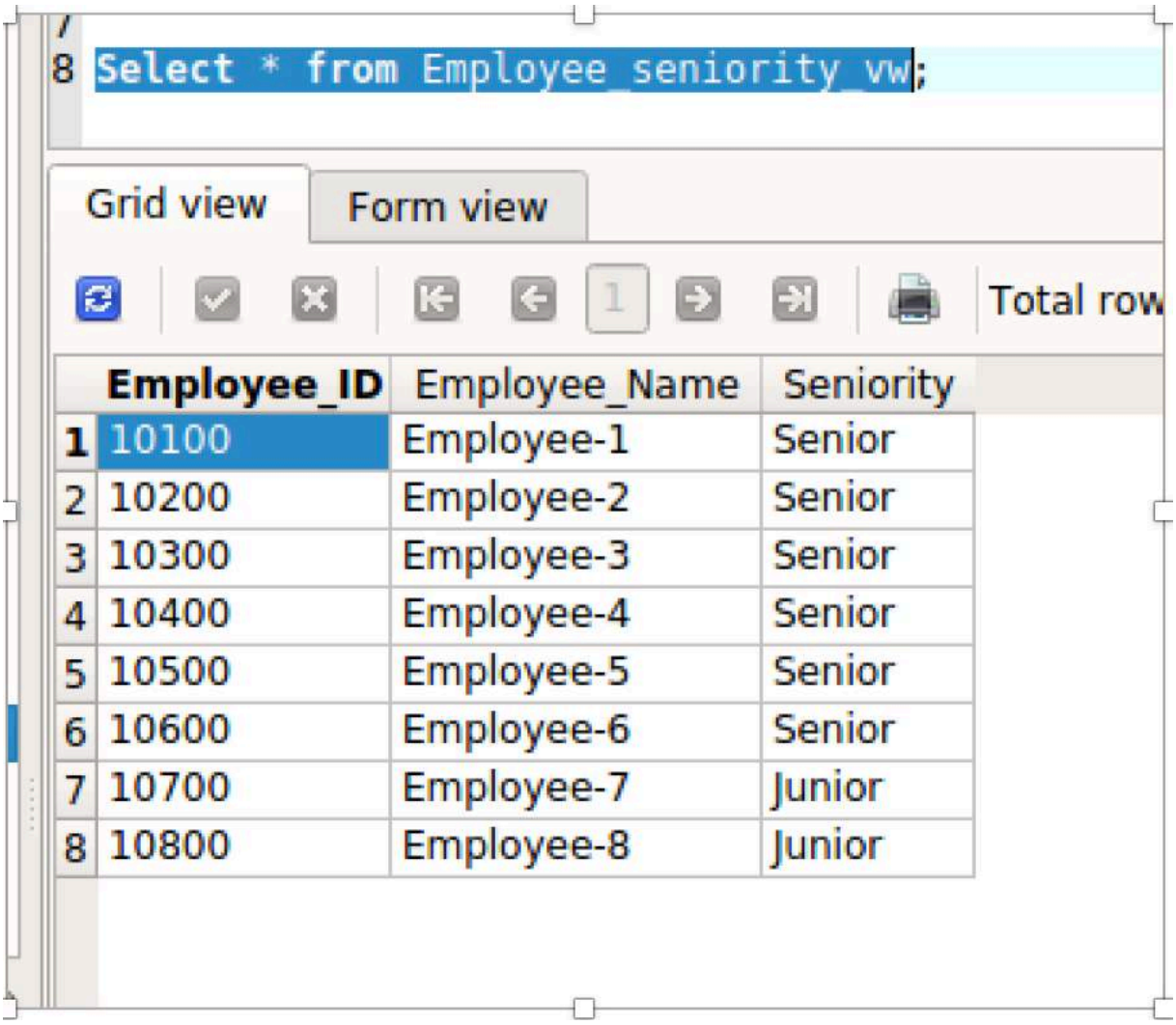
```
5
6 Pragma table_info(Employee_seniority_vw);
7
8 Select * from Employee;
```

Grid view | Form view

Total rows loaded: 3

| | cid | name | type | notnull | dflt_value | pk |
|---|---|---|---|---|---|---|
| 1 | 0 | Employee_ID | INTEGER | 0 | NULL | 0 |
| 2 | 1 | Employee_Name | VARCHAR (60) | 0 | NULL | 0 |
| 3 | 2 | Seniority | | 0 | NULL | 0 |

Somya Singh CSE 7330 Fall 2017
Email: somyas@smu.edu
Student ID: 47304053 (Distance Education)




| | Employee_ID | Employee_Name | Seniority |
|---|---|---|---|
| 1 | 10100 | Employee-1 | Senior |
| 2 | 10200 | Employee-2 | Senior |
| 3 | 10300 | Employee-3 | Senior |
| 4 | 10400 | Employee-4 | Senior |
| 5 | 10500 | Employee-5 | Senior |
| 6 | 10600 | Employee-6 | Senior |
| 7 | 10700 | Employee-7 | Junior |
| 8 | 10800 | Employee-8 | Junior |

```
Select
Employee_ID,
Employee_Name,
CASE
        WHEN ((julianday(date('now')) - (julianday(Hire_date))) < 365) THEN 'Newbie'
        WHEN ((julianday(date('now')) - (julianday(Hire_date))) > 1825) THEN 'Senior'
        WHEN (365 <= (julianday(date('now')) - (julianday(Hire_date))) <= 1825) THEN
'Junior'
END Seniority
FROM
Employee E
```

Somya Singh CSE 7330 Fall 2017
Email: somyas@smu.edu
Student ID: 47304053 (Distance Education)

# Test Cases for the Database

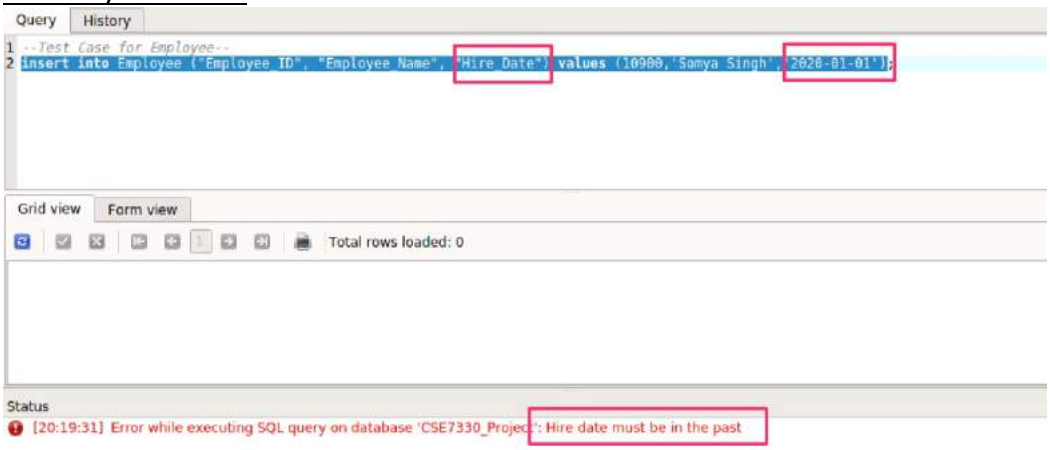
Below are test cases I created for table to make sure the data requirements and user expectation are met.

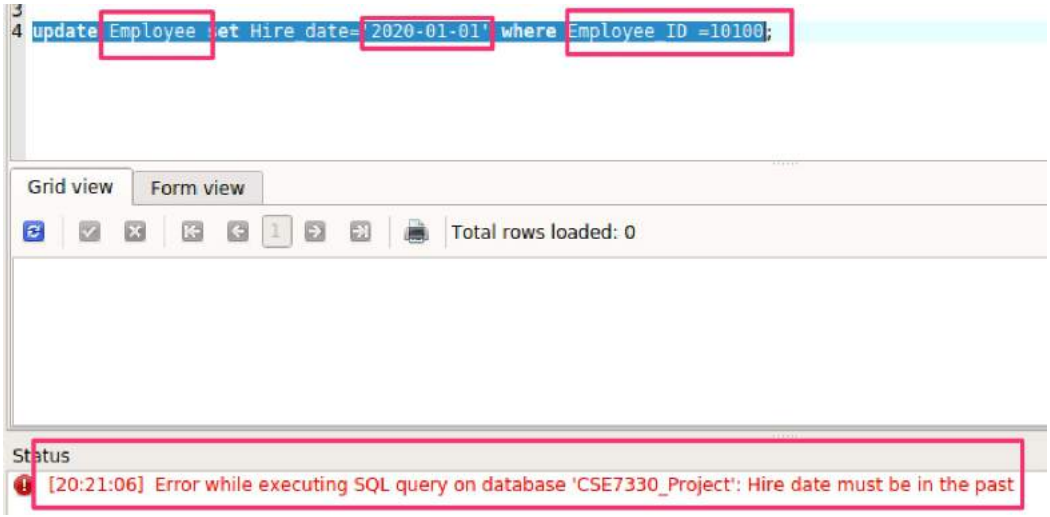
## 1. Employee Table and Employee_Seniority_VW

Key Requirement summarized on this table;

a Hire date cannot be a future date
b Based on hire date seniority should be updated to Newbie (< 1 year), Junior (<5 years) and Senior (> 5 years)
c Date can only be in format "YYYY-MM-DD"

To verify Hire Date:



Somya Singh CSE 7330 Fall 2017
Email: somyas@smu.edu
Student ID: 47304053 (Distance Education)

Insert Employee Record

```
Insert into Employee ('Employee_ID','Employee_Name','Hire_date') values (10900,'Employee-9','2017-11-15');
Insert into Employee ('Employee_ID','Employee_Name','Hire_date') values (11000,'Employee-10','2014-11-15');
Insert into Employee ('Employee_ID','Employee_Name','Hire_date') values (11100,'Employee-11','2007-11-15');

Select * from Employee;

slect * from Employee_seniority_vw
```

Grid view | Form view

Total rows loaded: 11

| | Employee_ID | Employee_Name | Hire_date | Manager_ID |
|---|---|---|---|---|
| 1 | 10100 | Employee-1 | 1984-11-08 | NULL |
| 2 | 10200 | Employee-2 | 1994-11-08 | 10100 |
| 3 | 10300 | Employee-3 | 2004-11-08 | 10200 |
| 4 | 10400 | Employee-4 | 2008-11-01 | 10200 |
| 5 | 10500 | Employee-5 | 2015-11-01 | 10400 |
| 6 | 10600 | Employee-6 | 2015-11-01 | 10400 |
| 7 | 10700 | Employee-7 | 2016-11-01 | 10400 |
| 8 | 10800 | Employee-8 | 2017-11-01 | 10200 |
| 9 | 10900 | Employee-9 | 2017-11-15 | NULL |
| 10 | 11000 | Employee-10 | 2014-11-15 | NULL |
| 11 | 11100 | Employee-11 | 2007-11-15 | NULL |

```
select * from Employee_seniority_vw;
```

Grid view | Form view

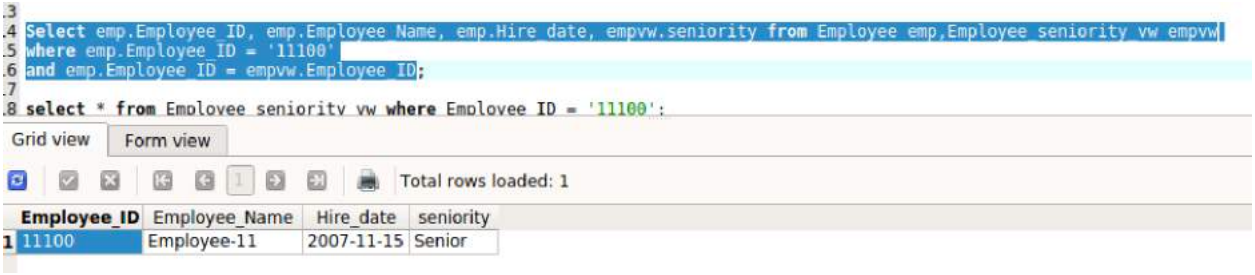
Total rows loade

| | Employee_ID | Employee_Name | Seniority |
|---|---|---|---|
| 1 | 10100 | Employee-1 | Senior |
| 2 | 10200 | Employee-2 | Senior |
| 3 | 10300 | Employee-3 | Senior |
| 4 | 10400 | Employee-4 | Senior |
| 5 | 10500 | Employee-5 | Junior |
| 6 | 10600 | Employee-6 | Junior |
| 7 | 10700 | Employee-7 | Junior |
| 8 | 10800 | Employee-8 | Newbie |
| 9 | 10900 | Employee-9 | Newbie |
| 10 | 11000 | Employee-10 | Junior |
| 11 | 11100 | Employee-11 | Senior |

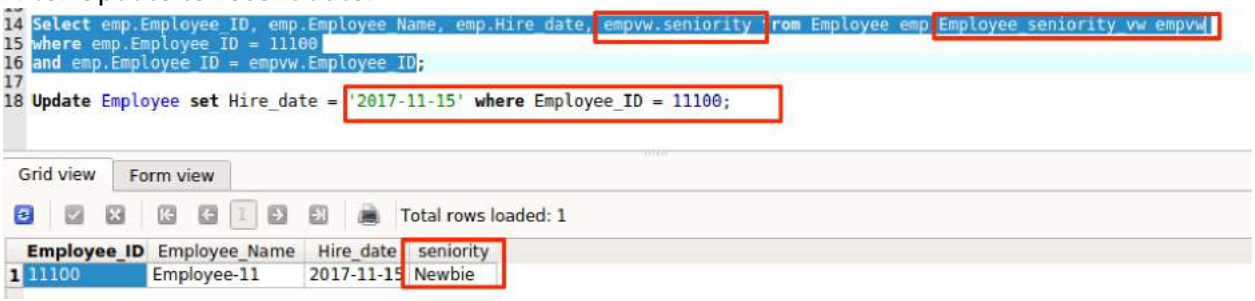
Somya Singh CSE 7330 Fall 2017
Email: somyas@smu.edu
Student ID: 47304053 (Distance Education)

Update Employee Record

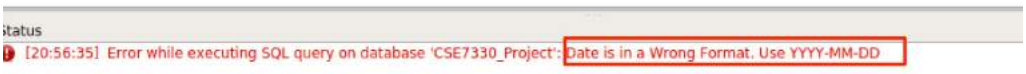
Before Update:




After Update to recent date:




Date Format


## 2. Software_Product

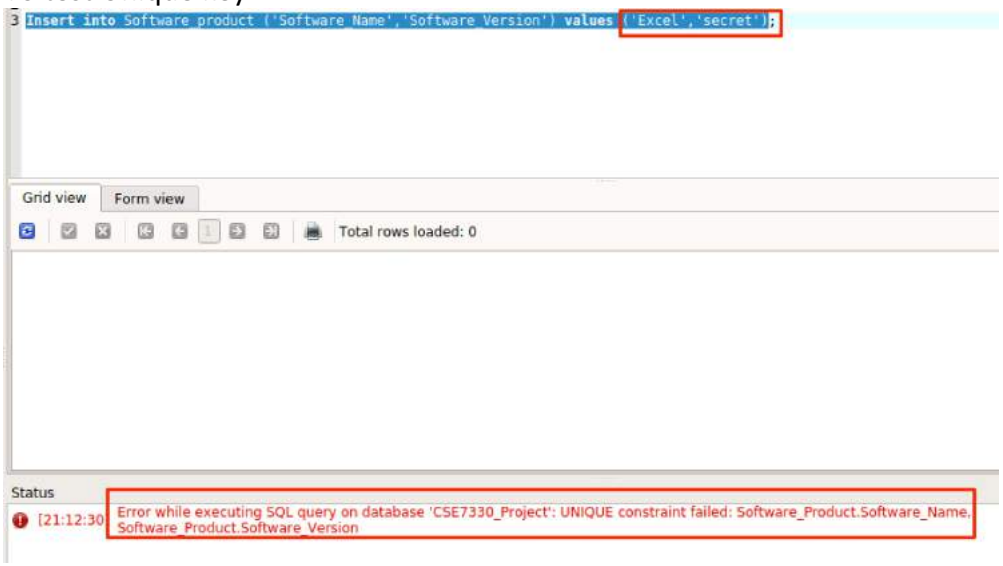
Key Requirement summarized on this table;

a. Unique Key for a software with Name and version key
b. Insert in Product table will enter a default record in build table tied to first component in component table. This is to ensure the statement that every product will require a build
c. Product Status is calculated based on inspection and status of Component.

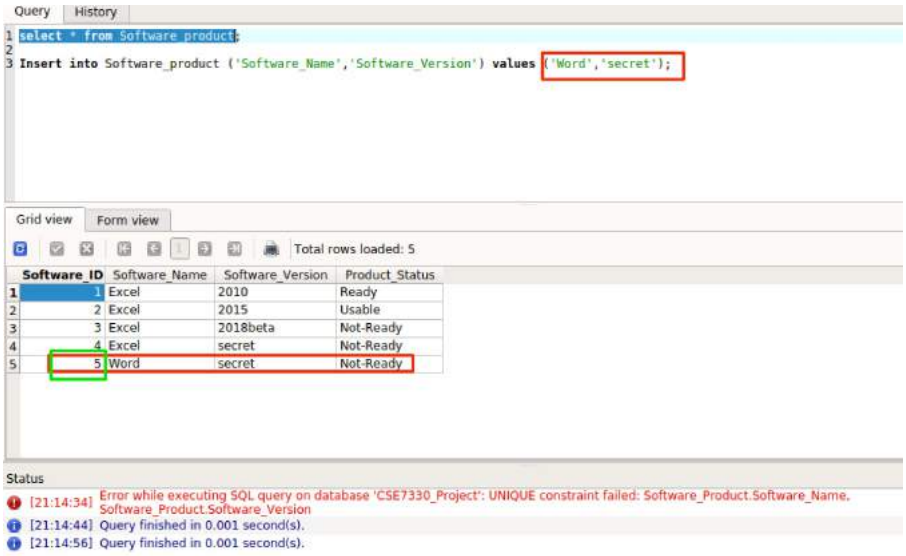
Email: somyas@smu.edu
Student ID: 47304053 (Distance Education)

d. Product Status can only be “Ready”, “Not-Ready” and “Usable”.

To test Unique key




To test and make sure build has record for Product.



Somya Singh CSE 7330 Fall 2017
Email: somyas@smu.edu
Student ID: 47304053 (Distance Education)

```
select * from Software_build where Software_ID = 5;
```

Grid view | Form view

Total rows loaded: 1

| | Software_ID | Component_ID |
|---|---|---|
| 1 | 5 | 1 |

The Product status is the lowest status of the component its associated

```
Select sp.Software_ID, sp.Product_Status, comp.Component_ID, comp.Component_Status
from Component comp, Software_product sp, Software_build sb
where sp.software_ID = sb.software_ID
AND sb.Component_ID = comp.Component_ID
AND sp.Software_ID = 4;
```

Grid view | Form view

Total rows loaded: 4

| Software_ID | Product_Status | Component_ID | Component_Status |
|---|---|---|---|
| 4 | Not-Ready | 1 | Ready |
| 4 | Not-Ready | 2 | Ready |
| 4 | Not-Ready | 5 | Not-Ready |
| 4 | Not-Ready | 8 | Ready |

Invalid Product Status.

```
Insert into Software_product ('Software_Name','Software_Version','Product_status') values ('Word','secret','Somya');
```

Grid view | Form view

Total rows loaded: 0

Status

[21:52:17] Error while executing SQL query on database 'CSE7330_Project': Wrong type of Product_Status

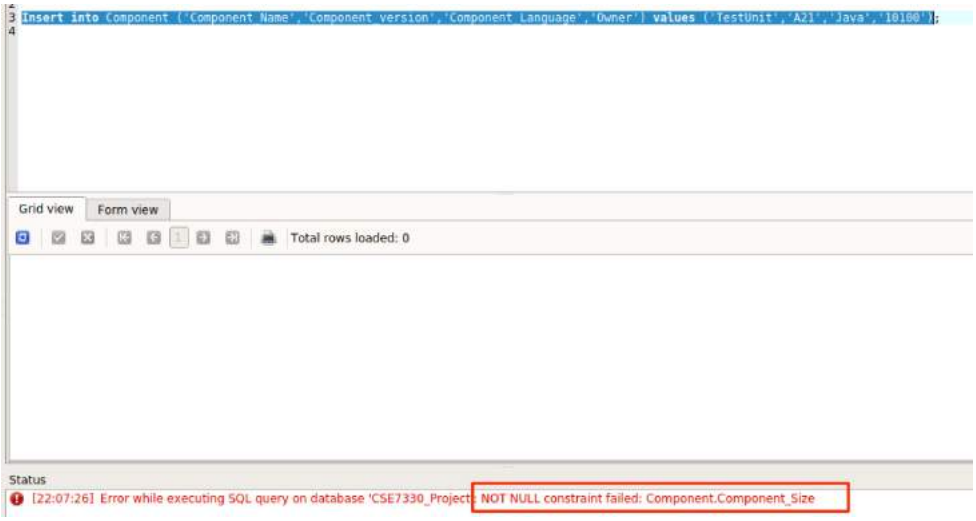
Somya Singh CSE 7330 Fall 2017
Email: somyas@smu.edu
Student ID: 47304053 (Distance Education)

## 3. Component

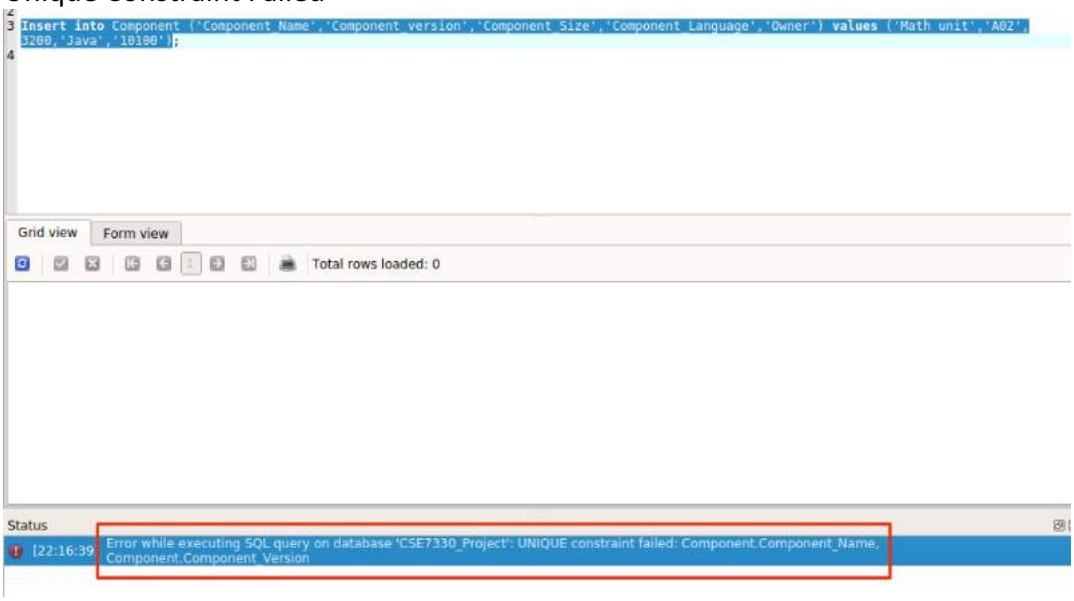
Key requirement identified for this table are;

a. Component_name, component_version and component_size cannot be null
b. Component_name and component_version makes unique index
c. Since Every product needs to have a build and each build does need a component, this means we cannot delete all the components associated with a Product. There has to be one component tied to product through software build table
d. Component Status is derived from Inspection table.

Not Null validation

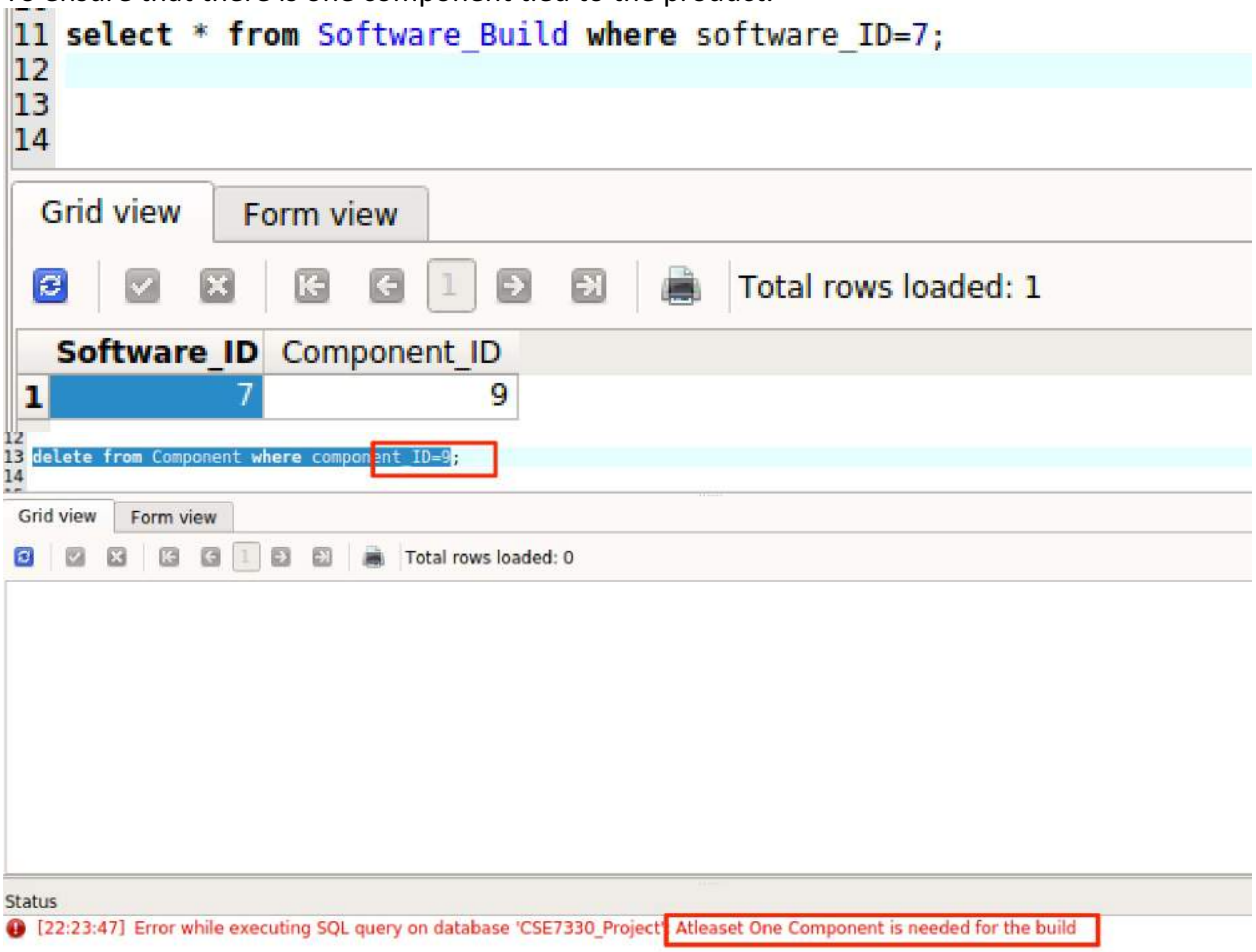



Unique Constraint Failed



Somya Singh CSE 7330 Fall 2017
Email: somyas@smu.edu
Student ID: 47304053 (Distance Education)

To ensure that there is one component tied to the product.




Based on latest Inspection score of component Is being updated.
For e.g component_id 3 had multiple test but the latest has 100 and which means its ready

Email: somyas@smu.edu
Student ID: 47304053 (Distance Education)

```
select * from Inspection where Inspected_component= 3;
```

Grid view | Form view

Total rows loaded: 4

| | Inspected_Component | Inspection_date | Own_BY | Score | Description |
|---|---|---|---|---|---|
| 1 | 3 | 2010-02-22 | 10100 | 55 | too many hard coded parameters, the software must be more maintainable and configurable be... |
| 2 | 3 | 2010-02-24 | 10100 | 78 | improved, but only handles DB2 format |
| 3 | 3 | 2010-02-26 | 10100 | 95 | Okay, handles DB3 format. |
| 4 | 3 | 2010-02-28 | 10100 | 100 | satisfied |

For a component with no inspection the status has to be Not-Ready

```
select * from Component where Component_ID = 5;

select * from Inspection where Inspected_component= 3;
```

Grid view | Form view

Total rows loaded: 1

| | Component_ID | Component_Name | Component_Version | Component_Size | Component_Language | Owner | Component_Status |
|---|---|---|---|---|---|---|---|
| 1 | 5 | Chart generator | C11 | 6500 | Java | 10200 | Not-Ready |

## 4. Inspection

Key Requirements are;

a. Score can be between 0 and 100 only
b. Once a record is inserted only text can be updated
c. On Delete or insert of an inspection record the Status of Component and Product gets updated

Score is less than 100

```
insert into Inspection ('Inspected_Component','Inspection_date','OWN_by','Score','Description') values (3,'2017-02-02',
10100,'600','Testing Inspection table');
```

Grid view | Form view

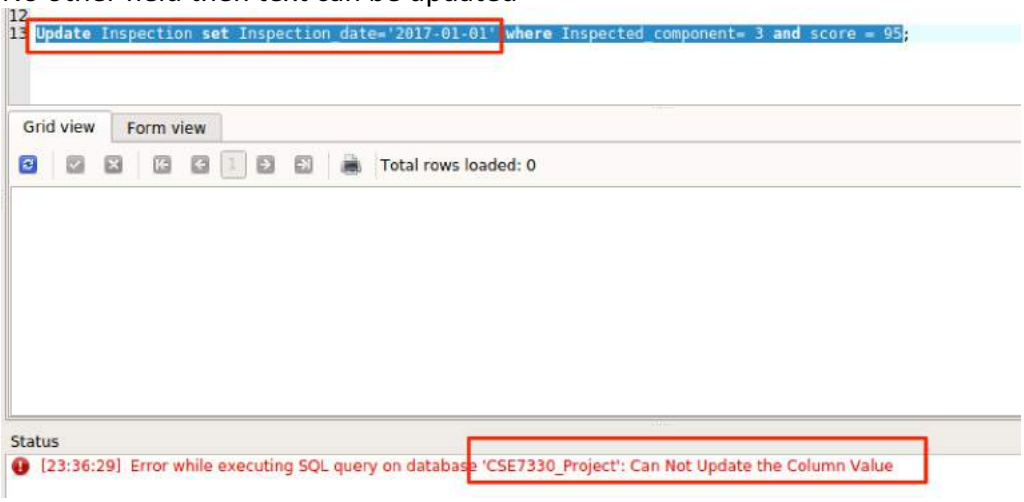
Total rows loaded: 0

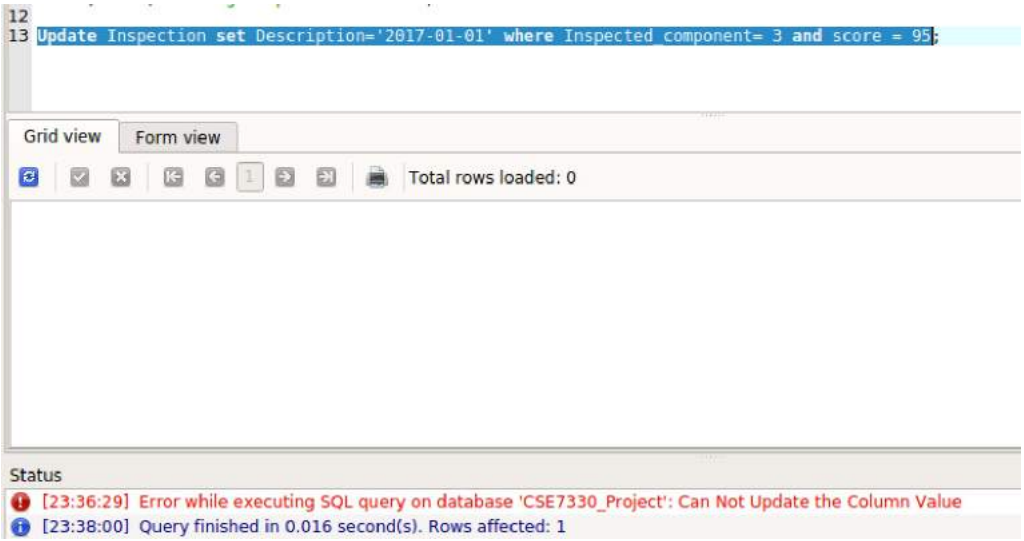
Status

[23:34:30] Error while executing SQL query on database 'CSE7330_Project': CHECK constraint failed: Inspection

Email: somyas@smu.edu
Student ID: 47304053 (Distance Education)

No other field then text can be updated



Somya Singh CSE 7330 Fall 2017
Email: somyas@smu.edu
Student ID: 47304053 (Distance Education)

# Testing Status update of Component and Product

The status of Component is derived from Inspection table. The Latest score in inspection table for a component will derive the status of component. If score is over 90 the Component becomes "Ready" and under 75 it will be "Not-Ready". Anything between it is "Usable". The lowest status of component will be status of product.
In order to implement this, I created two views and created triggers on Inspection table to update the status of component and Product based on these views.

**SoftwareComponent_status_vw**

```
Select comp.Component_ID AS ComponentID,MIN(ifNULL(insp.Score,0)) AS FinalScore
from Component comp
Left Join Inspection insp ON (insp.Inspected_Component = comp.Component_ID AND
insp.Inspection_date IN (select MAX(Inspection_date) from Inspection group by
Inspected_Component))
GROUP BY comp.Component_ID
ORDER BY comp.Component_ID
```

**Triggers on Inspection to update status of component to ready if score is over 90 for a new inspection.**

```
CREATE TRIGGER trig_insert_CompStatus_Ready
AFTER INSERT
ON Inspection
FOR EACH ROW
WHEN (new.Score > 90)
BEGIN
        UPDATE Component
        SET Component_Status = 'Ready'
        WHERE Component_ID = new.Inspected_Component;
END;
```

**Triggers on Inspection to update status of component to ready if last score is 90 after the inspection deleted.**

```
CREATE TRIGGER trig_delete_CompStatus_Ready
AFTER DELETE
ON Inspection
FOR EACH ROW
WHEN ( (
        SELECT FinalScore
        FROM SoftwareComponent_status_vw
        WHERE ComponentID = old.Inspected_Component
        )
```

Email: somyas@smu.edu
Student ID: 47304053 (Distance Education)

```
        > 90)
BEGIN
        UPDATE Component
        SET Component_Status = 'Ready'
        WHERE Component_ID = (
        SELECT ComponentID
        FROM SoftwareComponent_status_vw
        WHERE ComponentID = old.Inspected_Component
        );
END;
```

**SoftwareProduct_status_vw**

```
Select swb.Software_ID AS SoftwareID,swb.Component_ID AS
ComponentID,MIN(ifNULL(insp.Score,0)) AS FinalScore
from Component comp
Join Software_Build swb ON swb.Component_ID = comp.Component_ID
Left Join Inspection insp ON (insp.Inspected_Component = comp.Component_ID AND
insp.Inspection_date IN (select MAX(Inspection_date) from Inspection group by
Inspected_Component))
GROUP BY swb.Software_ID
ORDER BY swb.Software_ID
```

**Triggers on Inspection to update status of Product to Usable if score is over 75 and under 90 for a new inspection.**

```
CREATE TRIGGER trig_insert_ProdStatus_Usable
AFTER INSERT
ON Inspection
FOR EACH ROW
WHEN (74 < (
        SELECT FinalScore
        FROM SoftwareProduct_status_vw
        WHERE ComponentID = new.Inspected_Component
        )
        < 91)
BEGIN
        UPDATE Software_Product
        SET Product_Status = 'Usable'
        WHERE Software_ID = (
        SELECT SoftwareID
        FROM SoftwareProduct_status_vw
        WHERE ComponentID = new.Inspected_Component
        );
END;
```

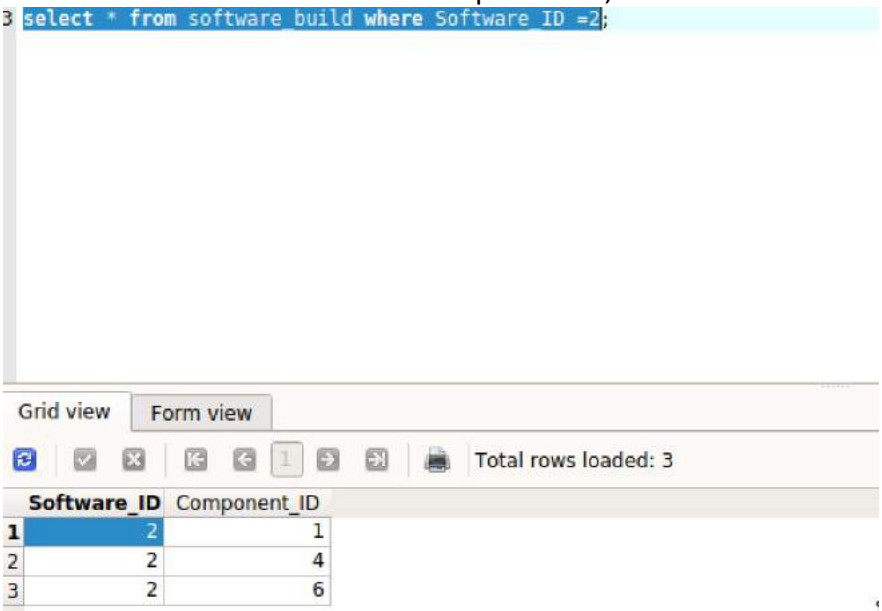
Somya Singh CSE 7330 Fall 2017
Email: somyas@smu.edu
Student ID: 47304053 (Distance Education)

**Triggers on Inspection to update status of Product to Usable if score is over 75 and under 90 for a deleted inspection.**

```
CREATE TRIGGER trig_delete_ProdStatus_NotReady
AFTER DELETE
ON Inspection
FOR EACH ROW
WHEN ( (
        SELECT FinalScore
        FROM SoftwareProduct_status_vw
        WHERE ComponentID = old.Inspected_Component
        )
        < 75)
BEGIN
        UPDATE Software_Product
        SET Product_Status = 'Not-Ready'
        WHERE Software_ID = (
        SELECT SoftwareID
        FROM SoftwareProduct_status_vw
        WHERE ComponentID = old.Inspected_Component
        );
END;
```

**For e.g**

Let pick the software product with Software_ID=2 which based on data currently provided is Usable. This software has three components 1, 4 and 6.



Somya Singh CSE 7330 Fall 2017
Email: somyas@smu.edu
Student ID: 47304053 (Distance Education)

For the components since 1 and 4 is ready and 6 is usable this results in Product to be Usable only.

Inspection

```
select * from Inspection where Inspected Component in (select Component ID from software build where Software ID =2);
```

Grid view | Form view

Total rows loaded: 3

| | Inspected_Component | Inspection_date | Own_BY | Score | Description |
|---|---|---|---|---|---|
| 1 | 1 | 2010-02-14 | 10100 | 100 | legacy code which is already approved |
| 2 | 4 | 2011-05-01 | 10200 | 100 | Okay ready for use |
| 3 | 6 | 2017-07-15 | 10300 | 80 | Okay ready for beta testing |

Component

```
select * from Component where Component ID in (select Component ID from software build where Software ID =2);
```

Grid view | Form view

Total rows loaded: 3

| | Component_ID | Component_Name | Component_Version | Component_Size | Component_Language | Owner | Component_Status |
|---|---|---|---|---|---|---|---|
| 1 | 1 | Keyboard Driver | K11 | 1200 | C | 10100 | Ready |
| 2 | 4 | Dbase Interface | D01 | 2500 | C++ | 10300 | Ready |
| 3 | 6 | Pen driver | P01 | 3575 | C | 10700 | Usable |

Software Product

Grid view | Form view

Total rows loaded: 1

| | Software_ID | Software_Name | Software_Version | Product_Status |
|---|---|---|---|---|
| 1 | 2 | Excel | 2015 | Usable |