

Data Engineering
LAB 10
Mukul Shingwani
B20AI023

Q1)A)

Attached below is the screenshot which shows the required 3 tables created in PHPMyAdmin or Sql serve management studio

BOOK table detail

Author_ID	Book_ID	Book
Da_001	Da001_Sel	Self Comes to Mind
Mi_009	Mi009_Emo	Emotion Machine
Mi_009	Mi009_Soc	Society of Mind
Ra_001	Ra001_Pha	Phantoms in the Brain
Ro_015	Ro015_Fan	Fantastic Beasts and Where to Find Them
Ro_015	Ro015_Gob	Goblet of Fire_Harry Potter
Ro_015	Ro015_Phi	Philosophers Stone_Harry Potter
Ro_015	Ro015_Pri	Prisoner of Azkaban_Harry Potter
Sa_001	Sa001_Voy	Voyage of the Turtle
Sa_001	Sa001_Wha	What Animals Think
To_015	To015_Fel	Fellowship of the Rings_Lord of the Rings
Wo_015	Wo015_Wod	Wodehouse at the Wicket

AUTHOR table details

Author_ID	Author_Name
Da_001	Damasio
Mi_009	Minsky
Ra_001	Ramachandran
Ro_015	Rowling
Ru_021	Russel
Sa_001	Safina
Ta_001	Tagore
To_015	Tolkien
Wo_015	Wodehouse

BOOK PURCHASE table details

Book_ID	Purchase_Dt	Copies	Price	Purchase_Price	Book_Value
Da001_Sel	Sep 1, 2021	1	\$50	NULL	NULL
Da001	Sep 7, 2021	5	\$50	NULL	NULL
Mi009_Emo	Sep2, 2021	2	\$50	NULL	NULL
Mi009_Soc	Sep 1, 2021	2	\$50	NULL	NULL
Ra001_Pha	Sep 2, 2021	2	\$50	NULL	NULL
Ro015_Fan	Sep 1, 2021	3	\$50	NULL	NULL
Ro015_Fan	Sep 9, 2021	12	\$35	NULL	NULL
Ro015_Gob	Sep 1, 2021	3	\$50	NULL	NULL
Ro015_Phi	Sep 1, 2021	3	\$50	NULL	NULL
Ro015_Phi	Sep 10, 2021	20	\$75	NULL	NULL
Ro015_Pri	Sep 1, 2021	3	\$50	NULL	NULL
Sa001_Voy	Sep 2, 2021	2	\$50	NULL	NULL
Sa001_Wha	Sep 2, 2021	2	\$50	NULL	NULL
To015_Fel	Sep 1, 2021	3	\$50	NULL	NULL
To015_Fel	Sep 12, 2021	9	\$55	NULL	NULL
Wo015_Wod	Sep 5, 2021	1	\$50	NULL	NULL

The SQL query written in C++ for a view comprising Author_name, Book_Name and Copies; Group by Book_Name is given below :-

```
CREATE VIEW author_book_copies AS(SELECT author_details.Author_Name,book_details.Book,
SUM(book_purchase_details.Copies) AS `Total_Copies` FROM
`author_details`,`book_details`,`book_purchase_details` WHERE book_details.Book_ID =
book_purchase_Details.Book_ID AND author_details.Author_ID = book_details.Author_ID GROUP BY
book_details.Book)
```

```
snprintf(query9, 512, "CREATE VIEW `%s` AS(SELECT
author_details.Author_Name,book_details.Book, SUM(book_purchase_details.Copies) AS
`Total_Copies` FROM `author_details`,`book_details`,`book_purchase_details` WHERE
book_details.Book_ID = book_purchase_Details.Book_ID AND author_details.Author_ID =
book_details.Author_ID GROUP BY book_details.Book)", viewName);
int createTableStatus9 = mysql_query(conn, query9);
if (createTableStatus9 != 0) {
    cout<<"Error while creating table: "<<mysql_error(conn)<<endl;
}
```

This is the view generated by executing the above query with appropriate parameters.

Author_Name	Book	Total_Copies
Minsky	Emotion Machine	2
Rowling	Fantastic Beasts and Where to Find Them	15
Tolkien	Fellowship of the Rings_Lord of the Rings	12
Rowling	Goblet of Fire_Harry Potter	3
Ramachandran	Phantoms in the Brain	2
Rowling	Philosophers Stone_Harry Potter	23
Rowling	Prisoner of Azkaban_Harry Potter	3
Damasio	Self Comes to Mind	1
Minsky	Society of Mind	2
Safina	Voyage of the Turtle	2
Safina	What Animals Think	2
Wodehouse	Wodehouse at the Wicket	1

Q1)B We know that the SELECT query takes less time than the VIEW creation + VIEW retrieval, but we also know that views can be inserted/deleted/updated by themselves. When we have to deal with multiple queries, views are a more straightforward and better way to execute than the post-execution of the SQL.

On comparing only the retrieval time for the queries over the original database and on the created view, we observe that almost all the data retrieval queries using these views are faster than the executing queries over the main database. Because we have a smaller

tuple to look through every time and get the number of copies of a particular book instantly.

If we compare the total timing of the creation of the views as well as the execution of the query then the retrieval time will be higher than the normal select and join operations on the original database. If we have a smaller set of the database then simple SELECT and JOIN operations may be effective but for developing a sustainable model where query executions are frequent, if we have built a view then the retrieval time will be less as the 'throughput' increases.

Testing : I had executed the query for the total number of copies of any or all books available both on views and the original database. I got the following results for the execution time of queries - 0.0027 for views and 0.0031 for the original database.

Q1)C) The view also gets updated once we insert/delete tuples from the main database according to the value which was inserted/deleted. This happens only if the update query which does the insertion/deletion suits the conditions/criteria with the matching view.

Deletion

If we delete the Book "Voyage of the Turtle" from the book_purchase table which has a book id = 'Sa001_Voy'. If we delete the tuple associated with this ID, then details corresponding to this book must also be deleted in the view. This is because the 'Copies' of this book have been deleted in the book_purchase_details table and hence the details corresponding to the Copies in the view must also be deleted thereby deleting the whole row.

The SQL Query written in C++ to do the same is as follows:

```
snprintf(query10, 512, "DELETE FROM `book_purchase_details` WHERE  
`Book_ID` = 'Sa001_Voy';", tableName);  
int createTableStatus10 = mysql_query(conn, query10);  
if (createTableStatus10 != 0) {  
    cout<<"Error while creating table: "<<mysql_error(conn)<<endl;  
}
```

Author_Name	Book	Total_Copies
Minsky	Emotion Machine	2
Rowling	Fantastic Beasts and Where to Find Them	15
Tolkien	Fellowship of the Rings_Lord of the Rings	12
Rowling	Goblet of Fire_Harry Potter	3
Ramachandran	Phantoms in the Brain	2
Rowling	Philosophers Stone_Harry Potter	23
Rowling	Prisoner of Azkaban_Harry Potter	3
Damasio	Self Comes to Mind	1
Minsky	Society of Mind	2
Safina	What Animals Think	2
Wodehouse	Wodehouse at the Wicket	1

Screenshot of the view after deletion takes place

Insertion

Now let's insert the details of the same Book in the book_purchase_details. The SQL query written in C++ to do the same is as follows:

```

snprintf(query11, 512, "INSERT INTO `book_purchase_details`(`Book_ID`,
`Purchase Dt`,`Copies`,`Price`,`Purchase_Price`,`Book_Value`) VALUES('Sa001_Voy', 'Sep 2, 2021', 2,
'$50', NULL, NULL);");

int createTableStatus11 = mysql_query(conn, query11);
if (createTableStatus11 != 0) {
    cout<<"Error while creating table: "<<mysql_error(conn)<<endl;
}

```

The screenshot of the view after we execute the SQL query which inserts the row corresponding to the 'Book_ID' of Voyage of the Turtle in the book_purchase_details is shown below:

Author_Name	Book	Total_Copies
Minsky	Emotion Machine	2
Rowling	Fantastic Beasts and Where to Find Them	15
Tolkien	Fellowship of the Rings_Lord of the Rings	12
Rowling	Goblet of Fire_Harry Potter	3
Ramachandran	Phantoms in the Brain	2
Rowling	Philosophers Stone_Harry Potter	23
Rowling	Prisoner of Azkaban_Harry Potter	3
Damasio	Self Comes to Mind	1
Minsky	Society of Mind	2
Safina	Voyage of the Turtle	2
Safina	What Animals Think	2
Wodehouse	Wodehouse at the Wicket	1

Q1)D)

Let us check if we can insert/delete any tuple in the view. In the previous question, we did the insertion/deletion in the database which got automatically updated in the view. We did not directly execute the SQL query on the view. The view got updated as we updated the database. Let's now try to directly update the view by inserting/deleting a tuple.

Let's try and delete the tuple where Author_Name='Minsky'

```
snprintf(query12, 512, "DELETE FROM `author_book_copies` WHERE  
`Author_Name` = 'Minsky';");  
int createTableStatus12 = mysql_query(conn, query12);  
if (createTableStatus12 != 0) {  
    cout<<"Error while creating table: "<<mysql_error(conn)<<endl;  
}
```

We get this as the output -> ERROR !!

```
Connected Successfully!  
Error while creating table: The target table author_book_copies of the DELETE is not updatable
```

This is because the view here created consists of the SQL commands like WHERE and GROUP BY. That's why it's **not possible to insert or delete** entries **directly** in the above **view**. We can however update the view, as seen in the previous question, by updating the entries in the main database which automatically updates the information regarding the same entity in the view.

Q1)E)

The query to be executed in order to create a trigger to calculate the Purchase_Price of the books on the date of purchase is as follows:

```
CREATE TRIGGER purchase_trigger BEFORE INSERT ON book_purchase_Details FOR EACH ROW SET  
new.Purchase_Price = new.Price*new.Copies
```

```
snprintf(query13, 512, "CREATE TRIGGER purchase_trigger BEFORE INSERT ON  
book_purchase_Details FOR EACH ROW SET new.Purchase_Price = new.Price*new.Copies;");  
int createTableStatus13 = mysql_query(conn, query13);  
if (createTableStatus13 != 0) {  
    cout<<"Error while creating table: "<<mysql_error(conn)<<endl;  
}
```

After this query is executed, the trigger 'purchase_triger' starts executing and calculates the prices of the newly added information as Purchase_Price =

Price*Copies. Let's insert the following data into the table
`book_purchase_details`:

```
snprintf(query14, 512, "INSERT INTO  
`book_purchase_details`(`Book_ID`,`Purchase_Dt`,`Copies`,`Price`,`Purchase_Price`,`Book_Value`)  
VALUES('Mi009_Soc','Sep 16, 2021', 5,75,NULL,NULL);");  
int createTableStatus14 = mysql_query(conn, query14);  
if (createTableStatus14 != 0) {  
    cout<<"Error while creating table: "<<mysql_error(conn)<<endl;  
}
```

Now, the `book_purchase_details` table looks as follows:

Book_ID	Purchase_Dt	Copies	Price	Purchase_Price	Book_Value
Da001_Sel	Sep 1, 2021	1	50	NULL	NULL
Da001	Sep 7, 2021	5	50	NULL	NULL
Mi009_Emo	Sep 2, 2021	2	50	NULL	NULL
Mi009_Soc	Sep 1, 2021	2	50	NULL	NULL
Ra001_Pha	Sep 2, 2021	2	50	NULL	NULL
Ro015_Fan	Sep 1, 2021	3	50	NULL	NULL
Ro015_Fan	Sep 9, 2021	12	35	NULL	NULL
Ro015_Gob	Sep 1, 2021	3	50	NULL	NULL
Ro015_Phi	Sep 1, 2021	3	50	NULL	NULL
Ro015_Phi	Sep 10, 2021	20	75	NULL	NULL
Ro015_Pri	Sep 1, 2021	3	50	NULL	NULL
Sa001_Wha	Sep 2, 2021	2	50	NULL	NULL
To015_Fel	Sep 1, 2021	3	50	NULL	NULL
To015_Fel	Sep 12, 2021	9	55	NULL	NULL
Wo015_Wod	Sep 5, 2021	1	50	NULL	NULL
Sa001_Voy	Sep 2, 2021	2	50	NULL	NULL
Mi009_Soc	Sep 16, 2021	5	75	375	NULL

The trigger worked on the latest insertion into the table and we hence get the Purchase_Price of the same.

Q1)F)

Yes, we can create triggers on views.

The CREATE TRIGGER statement is used to add triggers to the database schema.

Triggers are database operations that are automatically performed when a specified database event occurs. Triggers may be created on the original database as well as on the views. For creating triggers on the views we can do so by specifying INSTEAD OF

in the CREATE TRIGGER statement but we cannot create a trigger on any original table by the use of INSTEAD OF. Now if there are one or more ON INSERT, ON DELETE or ON UPDATE triggers defined on a view, then without encountering an error we can execute the INSERT, DELETE or UPDATE statement on the view. Executing an INSERT, DELETE or UPDATE operation on views causes the associated trigger to execute .

In this case, we cannot create a trigger on the view as the view is non-updateable. (created using GROUP BY statement). We can check this by using an example.

```
CREATE VIEW book_view_2 AS(SELECT book_purchase_details.Book_ID,  
SUM(book_purchase_details.Copies) AS `Total_Copies`, book_purchase_details.Book_Value FROM  
`book_purchase_details` GROUP BY book_purchase_details.Copies)
```

```
snprintf(query15, 512, "CREATE VIEW `%s` AS(SELECT book_purchase_details.Book_ID,  
SUM(book_purchase_details.Copies) AS `Total_Copies`, book_purchase_details.Book_Value FROM  
`book_purchase_details` GROUP BY book_purchase_details.Copies);", viewName2);  
int createTableStatus15 = mysql_query(conn, query15);  
if (createTableStatus15 != 0) {  
cout<<"Error while creating table: "<<mysql_error(conn)<<endl;  
}
```

The view generated is shown below:

Book	Total_Copies	Book_Value
Self Comes to Mind	2	NULL
Emotion Machine	10	NULL
Fantastic Beasts and Where to Find Them	15	NULL
Self Comes to Mind	5	NULL
Fellowship of the Rings_Lord of the Rings	9	NULL
Fantastic Beasts and Where to Find Them	12	NULL
Philosophers Stone_Harry Potter	20	NULL

Now, let's generate a trigger on this view which does the function Book_Value = Copies*10.

```
CREATE TRIGGER book_trigger_fo_val INSTEAD OF INSERT ON book_purchase_details FOR EACH ROW  
SET new.Book_Value = new.Copies*10
```

The C++ code to generate this trigger is as follows:

```
snprintf(query16, 512, "CREATE TRIGGER book_trigger_fo_val INSTEAD OF INSERT ON
book_purchase_details FOR EACH ROW SET new.Book_Value = new.Copies*10;");
int createTableStatus16 = mysql_query(conn, query16);
if (createTableStatus16 != 0) {
    cout<<"Error while creating table: "<<mysql_error(conn)<<endl;
}
```

We can see that hereby, the trigger doesn't get created in this particular example because the view has been created by using a GROUP BY statement. Let's demonstrate this. This trigger doesn't get generated and gives the following syntax error:

```
Connected Successfully!
Error while creating table: You have an error in your SQL syntax; check the manual that corresponds to
your MariaDB server version for the right syntax to use near 'INSTEAD OF INSERT ON expected_trigger FOR
EACH ROW (SET new.Book Value = new....' at line 1
```

Q1G)

DDL trigger: A DDL trigger executes in response to a change to the structure of the database. For example, CREATE, ALTER, DROP etc.

DML trigger: A DML trigger executes in response to a change in data. For example, INSERT, UPDATE, DELETE etc

We can create a DDL trigger over the database for the given scenario. Whenever DROP_TABLE is tried to be executed it revokes a trigger which first needs to be disabled to provide safety against some errors. Also, If we try to set Purchase_Price field as not null and put a compulsory constraint on the field then we can create a DDL trigger which gets executed on ALTER command on Book_purchase_Details table and restricts the alter and displays a warning message: Table alteration is not allowed.

Q1H)

```
CREATE OR REPLACE VIEW total_copies_book AS(SELECT author_details.Author_Name,
book_details.Book, book_purchase_details.Book_Value FROM author_details, book_details,
book_purchase_details WHERE book_details.Book_ID = book_purchase_details.Book_ID AND
author_details.Author_ID = book_details.Author_ID)
```

The SQL query to create the view using C++ is as follows:

```
snprintf(query18, 512, "CREATE OR REPLACE VIEW `%s` AS(SELECT
author_details.Author_Name, book_details.Book, book_purchase_details.Book_Value FROM
author_details, book_details, book_purchase_details WHERE book_details.Book_ID =
book_purchase_details.Book_ID AND author_details.Author_ID = book_details.Author_ID);",
viewName3);
```



```
int createTableStatus18 = mysql_query(conn, query18);
if (createTableStatus18 != 0) {
    cout<<"Error while creating table: "<<mysql_error(conn)<<endl;
}
```

For Trigger - ASSUMPTION : Let's assume that the book is rented in order to get 50% profit, so there will be an addition of 50% to the actual price.

CREATE TRIGGER profitable_books **BEFORE INSERT ON** book_purchase_details **FOR EACH ROW SET**
new.Book_Value = new.Purchase_Price + 0.5*Purchase_Price

```
snprintf(query19, 512, "CREATE TRIGGER profitable_books BEFORE INSERT ON
book_purchase_details FOR EACH ROW SET new.Book_Value = new.Purchase_Price +
0.5*Purchase_Price;");
int createTableStatus19 = mysql_query(conn, query19);
if (createTableStatus19 != 0) {
    cout<<"Error while creating table: "<<mysql_error(conn)<<endl;
}
```

Q1)I)

In order to calculate the total purchase price of respective books we can create a trigger on the INSERT operation, which updates the view on each INSERT operation. So this will be a DML trigger.

```
SET price_total = 0;
CREATE TRIGGER price_each_book
BEFORE INSERT ON Book_Purchase_Details
FOR EACH ROW
UPDATE purchase_value
SET price_total = price_total + NEW.purchase_price
WHERE purchase_value.Book_ID = NEW.Book_ID
```

We can also create a view containing Book_ID and the sum of Purchase_Price which is GROUP BY Book_ID

Q1)J)

Yes, both BEFORE and AFTER triggers can be used alternatively in each other's place but for a particular trigger we cannot use both the triggers together in the same query at the same place.

```
CREATE TRIGGER trigger
BEFORE INSERT ON `Book_Purchase_Details`
FOR EACH ROW
```

```
SET new.Purchase_Price = new.Price * new.Copies;
```

```
CREATE TRIGGER `trigger`  
AFTER UPDATE ON Book_Purchase_Details  
FOR EACH ROW  
SET new.Purchase_Price = new.Price * new.Copies
```

Q2)

Given schema is:

APPLICATION (SID, Date, State)
COURSE (CID, Title, Credits)
RANK (SID, Avg, Credits, Rank)
EXAM (CID, SID, Date, Grade)

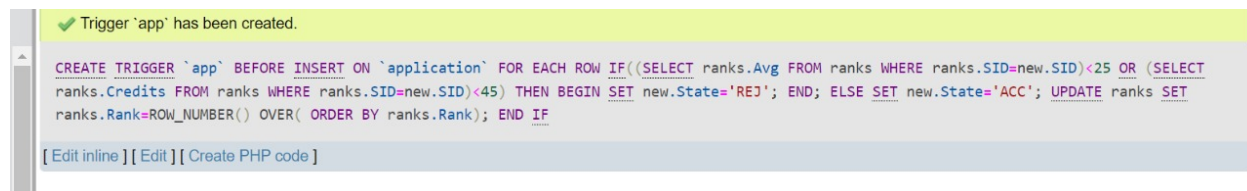
Q2)A)

In case of acceptance, the student is automatically assigned a position in the ranking, determined by the average of the grades; in case of equality of media, we consider first the greatest number of credits incurred at the date of application and finally the insertion order of the applications.

If a student renounces the scholarship (State is changed to "dropout"), the ranking is updated

REJ represents: rejected, and ACC represents accepted

```
CREATE TRIGGER `app` BEFORE INSERT ON `application` FOR EACH ROW IF((SELECT ranks.Avg FROM ranks WHERE ranks.SID=new.SID)<25 OR (SELECT ranks.Credits FROM ranks WHERE ranks.SID=new.SID)<45) THEN BEGIN SET new.State='REJ'; END; ELSE SET new.State='ACC'; UPDATE ranks SET ranks.Rank=ROW_NUMBER() OVER( ORDER BY ranks.Rank);END IF;
```



Trigger 'app' has been created.

```
CREATE TRIGGER `app` BEFORE INSERT ON `application` FOR EACH ROW IF((SELECT ranks.Avg FROM ranks WHERE ranks.SID=new.SID)<25 OR (SELECT ranks.Credits FROM ranks WHERE ranks.SID=new.SID)<45) THEN BEGIN SET new.State='REJ'; END; ELSE SET new.State='ACC'; UPDATE ranks SET ranks.Rank=ROW_NUMBER() OVER( ORDER BY ranks.Rank); END IF
```

[Edit inline] [Edit] [Create PHP code]

☐ **app**

BEFORE INSERT

Edit

Export

Drop

```

snprintf(query2, 512, "CREATE TRIGGER `app` BEFORE INSERT ON `application`
FOR EACH ROW IF((SELECT ranks.Avg FROM ranks WHERE ranks.SID=new.SID)<25 OR
(SELECT ranks.Credits FROM ranks WHERE ranks.SID=new.SID)<45) THEN BEGIN SET
new.State='REJ'; END; ELSE SET new.State='ACC'; UPDATE ranks SET
ranks.Rank=ROW_NUMBER() OVER( ORDER BY ranks.Rank);END IF;");
int insertStatus2 = mysql_query(conn, query2);
if (insertStatus2 != 0) {
    cout<<"Error while inserting: "<<mysql_error(conn)<<endl;
}

```

In case of dropout following will be helpful

```

Create trigger Dropout
after update of State on `application`
for each row
when new.State = "dropout"
begin
delete * from ranks where SID = new.SID;
End

```

Q2)B)

Although the above trigger could handle this as well, but a dedicated answer would be

```

Create trigger updating_ranks_1
after insert on ranks
for each row
begin
POS:=select count(*) --count the number of students with greater
-- average grade or same average but more
credits
from ranks
where(Average > new.Average) OR (Average = new.Average AND Credits > new.Credits) OR (Average =
new.Average AND Credits = new.Credits);
update ranks set Rank = Rank + 1 where Rank > POS;
-- we move the following one down by one position (+1 in the ranking)
update ranks set Rank = POS + 1 where SID = new.SID;
-- we attribute the «new» student the rank (pos + 1)
End

```

- The insertion order is implicitly considered, in fact when the average and the credits are both the same, the «older» applications are privileged.
- The rank is below the others with the same average and credits.
- POS is the number of preceding students in the ranking

```

Create trigger updating_ranks_2
after delete From ranks
for each row
begin
    --update the position of the one that follows moving them up by one
    update ranks set Rank = Rank - 1
    where Rank > old.Rank;
End

```

Q3)A)

Query for creating the trigger:

```

CREATE TRIGGER `friend_del` AFTER DELETE ON `friendship` FOR EACH ROW IF (select * from friendship
where id1 = old.id2 and id2 = old.id1) THEN delete from friendship where (id1 = old.id2 and id2 = old.id1);
END IF;

```

✓ Trigger 'friend_del' has been created.

```

CREATE TRIGGER `friend_del` AFTER DELETE ON `friendship` FOR EACH ROW IF (select * from friendship where id1 = old.id2 and id2 = old.id1)
THEN delete from friendship where (id1 = old.id2 and id2 = old.id1); END IF;

```

[Edit inline] [Edit] [Create PHP code]



friend_del AFTER DELETE



Edit



Export



Drop

```

sprintf(query3, 512, "CREATE TRIGGER `friend_del` AFTER DELETE ON
`friendship` FOR EACH ROW IF (select * from friendship where id1 = old.id2
and id2 = old.id1) THEN delete from friendship where (id1 = old.id2 and
id2 = old.id1); END IF;");
int insertStatus3 = mysql_query(conn, query3);
if (insertStatus3 != 0) {
    cout<<"Error while inserting: "<<mysql_error(conn)<<endl;
}

```

Q3)B)

```

CREATE TRIGGER `grade_check` AFTER UPDATE ON `student` FOR EACH ROW IF (new.class>12) THEN
DELETE FROM student WHERE id=new.id; END IF;

```


✔ Trigger 'grade_check' has been created.







```
CREATE TRIGGER `grade_check` AFTER UPDATE ON `student` FOR EACH ROW IF (new.class>12) THEN DELETE FROM student WHERE id=new.id; END IF;
```

[Edit inline] [Edit] [Create PHP code]

Triggers ⓘ

☐ Check all  Export  Drop

 Create new trigger

	Name	Time	Event	
<input type="checkbox"/>	Update_all_friends	AFTER	UPDATE	 Edit  Export  Drop
<input type="checkbox"/>	del_grad	AFTER	UPDATE	 Edit  Export  Drop

C++ query

```
snprintf(query4, 512, "CREATE TRIGGER `grade_check` AFTER UPDATE ON
`student` FOR EACH ROW IF (new.class>12) THEN DELETE FROM student WHERE
id=new.id; END IF;");
int insertStatus4 = mysql_query(conn, query4);
if (insertStatus4 != 0) {
    cout<<"Error while inserting: "<<mysql_error(conn)<<endl;
}
```




Q3)C)

```
CREATE TRIGGER `update_all_friends` AFTER UPDATE ON `student` FOR EACH ROW IF
(new.class=old.class+1) THEN UPDATE student SET class=class+1 WHERE id in (SELECT id2 FROM
friendship where id1=new.id); END IF;
```

✔ Trigger 'Update_all_friends' has been created.

```
CREATE TRIGGER `Update_all_friends` AFTER UPDATE ON `student` FOR EACH ROW IF (new.class=old.class+1) THEN UPDATE student SET
class=class+1 WHERE id in (SELECT id2 FROM friendship where id1=new.id); END IF;
```

[Edit inline] [Edit] [Create PHP code]

	Name	Time	Event	
<input type="checkbox"/>	Update_all_friends	AFTER	UPDATE	 Edit  Export  Drop

C++ query

```
snprintf(query5, 512, "CREATE TRIGGER `update_all_friends` AFTER UPDATE ON
`student` FOR EACH ROW IF (new.class=old.class+1) THEN UPDATE student SET
```

```
class=class+1 WHERE id in (SELECT id2 FROM friendship where id1=new.id);
END IF;");
int insertStatus5 = mysql_query(conn, query5);
if (insertStatus5 != 0) {
    cout<<"Error while inserting: "<<mysql_error(conn)<<endl;
}
```

-----END-----