

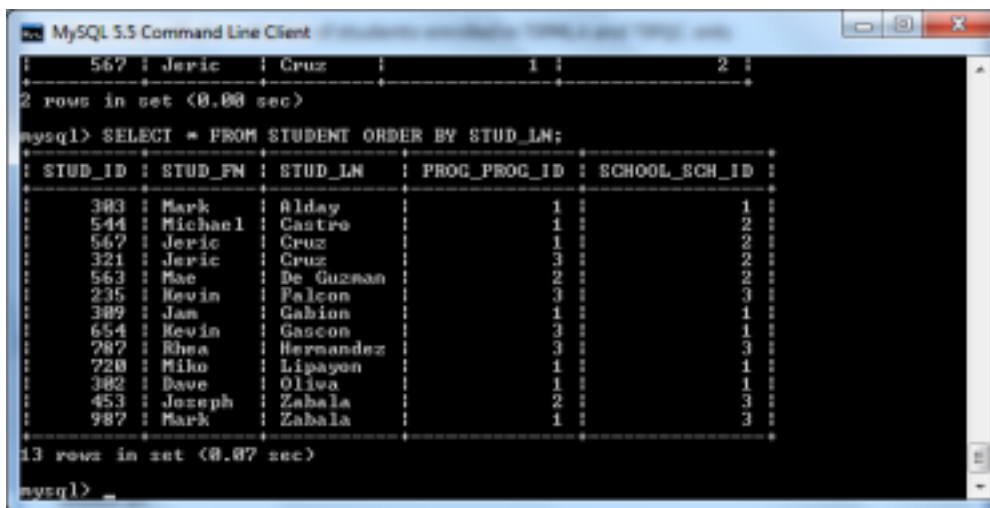
Course: CPE202	Section: CPE21S1
Name: ABALOS, JHO RAVEN	Date Performed: NOV. 6,2023
	Date Submitted: NOV. 6, 2023
	Instructor: ENGR. RUFO MARASIGAN JR.
Laboratory Activity No. 6	
DATA MANIPULATION LANGUAGE AND OTHER SQLSELECT FUNCTIONS	
1. Objective(s):	
The activity aims to introduce the concepts Data Manipulation Language in MySQL and other select functions.	
2. Intended Learning Outcomes (ILOs):	
The students shall be able to: <ol style="list-style-type: none"> 1. Identifies different Data Manipulation Language 2. Apply the knowledge of DML in manipulating databases and other select functions 	
3. Discussion	
Data on the database are the core contents which will be needed in the future use. Time to time data are been fetched left and right. As for the Database Management System it is necessary that data could be retrieved and presented. MySQL on the other hand provides a query that lets you access manipulate its content via its Data Manipulation Language (DML). It includes Select, Update, Modify and other similar functions.	
4. Resources:	
1 Desktop Computer MYSQL	
5. Procedure:	

ORDER BY

The ORDER BY keyword is used to sort the result-set.

Given the table, I want to know the list of the students which is sorted according to their last

*name. **SELECT * FROM STUDENT ORDER BY STUD_LN;***



```
mysql> SELECT * FROM STUDENT ORDER BY STUD_LN;
```

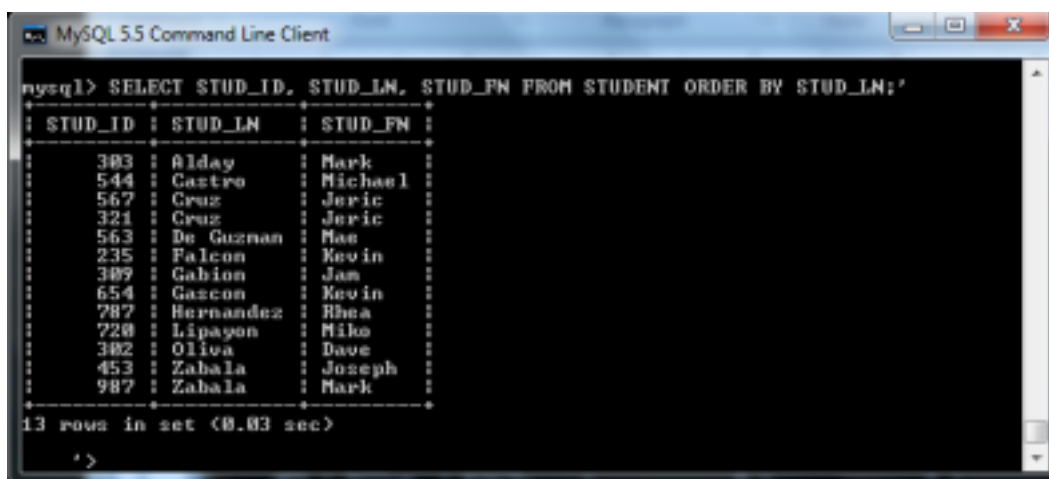
STUD_ID	STUD_FN	STUD_LN	PROG_PROG_ID	SCHOOL_SCH_ID
303	Mark	Alday	1	1
544	Michael	Castro	1	2
567	Jeric	Cruz	1	2
321	Jeric	Cruz	3	2
563	Mae	De Guzman	2	2
235	Kevin	Falcon	3	3
309	Jan	Gabion	1	1
654	Kevin	Gascon	3	1
787	Rhea	Hernandez	3	3
720	Miko	Lipayan	1	1
302	Dave	Oliva	1	1
453	Joseph	Zabala	2	3
987	Mark	Zabala	1	3

```
mysql>
```

But

I want the last name first before the first name appears and no prog_ID and School_ID

**SELECT STUD_ID, STUD_LN, STUD_FN
FROM STUDENT ORDER BY STUD_LN;**



```
mysql> SELECT STUD_ID, STUD_LN, STUD_FN FROM STUDENT ORDER BY STUD_LN;
```

STUD_ID	STUD_LN	STUD_FN
303	Alday	Mark
544	Castro	Michael
567	Cruz	Jeric
321	Cruz	Jeric
563	De Guzman	Mae
235	Falcon	Kevin
309	Gabion	Jan
654	Gascon	Kevin
787	Hernandez	Rhea
720	Lipayan	Miko
302	Oliva	Dave
453	Zabala	Joseph
987	Zabala	Mark

```
mysql>
```

IN ORDER BY FUNCTION,

--- You can add ASC OR DESC before the semi colon to sort it ascending or descending.

```
CREATE TABLE prog_tb(  
  Prog_prog_ID int not null,  
  prog_desc varchar(50),  
  PRIMARY KEY(Prog_prog_ID)  
);
```

```
CREATE TABLE SCHOOL_tb(  
  School_SCH_ID int not null,  
  SCH_name varchar(100),  
  PRIMARY KEY(School_SCH_ID)  
);
```

```
CREATE TABLE STUDENT(  
  Stu_ID int not null,  
  Stu_Fn varchar(50),  
  Stu_Ln varchar(50),  
  Prog_prog_ID int(100)not null,  
  School_SCH_ID int(100)not null,  
  PRIMARY KEY (Stu_ID),  
  FOREIGN KEY(Prog_prog_ID) REFERENCES prog_tb(Prog_prog_ID),  
  FOREIGN KEY(School_SCH_ID) REFERENCES SCHOOL_tb(School_SCH_ID)  
);
```

```
INSERT INTO prog_tb(Prog_prog_ID,prog_desc)  
VALUES (00001,"CPE");
```

```
INSERT INTO prog_tb(Prog_prog_ID,prog_desc)  
VALUES (00002,"EE");
```

```
INSERT INTO prog_tb(Prog_prog_ID,prog_desc)  
VALUES (00003,"CE");
```

```
INSERT INTO SCHOOL_tb(School_SCH_ID, SCH_name)  
VALUES(1, "TIPMANILA");
```

```
INSERT INTO SCHOOL_tb(School_SCH_ID, SCH_name)  
VALUES(2, "TIPQC");
```

```
INSERT INTO SCHOOL_tb(School_SCH_ID, SCH_name)
```

VALUES(3, "TIPLAGUNA");

```
MariaDB [LAB_6_202A_STUDENT_PROFILE_DB]> desc STUDENT;
```

Field	Type	Null	Key	Default	Extra
Stu_ID	int(11)	NO	PRI	NULL	
Stu_Fn	varchar(50)	YES		NULL	
Stu_Ln	varchar(50)	YES		NULL	
Prog_prog_ID	int(100)	NO		NULL	
School_SCH_ID	int(100)	NO		NULL	

INSERT INTO STUDENT(Stu_ID, Stu_Fn,Stu_Ln,Prog_prog_ID,School_SCH_ID)
VALUES (303,"Mark","Alday",1,1);

INSERT INTO STUDENT(Stu_ID, Stu_Fn,Stu_Ln,Prog_prog_ID,School_SCH_ID)
VALUES (544,"Michael","Castro",1,2);

INSERT INTO STUDENT(Stu_ID, Stu_Fn,Stu_Ln,Prog_prog_ID,School_SCH_ID)
VALUES (567,"Jeric","Cruz",1,2);

INSERT INTO STUDENT(Stu_ID, Stu_Fn,Stu_Ln,Prog_prog_ID,School_SCH_ID)
VALUES (321,"Jeric","Cruz",3,2);

INSERT INTO STUDENT(Stu_ID, Stu_Fn,Stu_Ln,Prog_prog_ID,School_SCH_ID)
VALUES (563,"Mae","De Guzman",2,2);

INSERT INTO STUDENT(Stu_ID, Stu_Fn,Stu_Ln,Prog_prog_ID,School_SCH_ID)
VALUES (235,"Kevin","Falcon",3,3);

INSERT INTO STUDENT(Stu_ID, Stu_Fn,Stu_Ln,Prog_prog_ID,School_SCH_ID)
VALUES (309,"Jam","Gabion",1,1);

INSERT INTO STUDENT(Stu_ID, Stu_Fn,Stu_Ln,Prog_prog_ID,School_SCH_ID)
VALUES (654,"Kevin","Gascon",3,1);

INSERT INTO STUDENT(Stu_ID, Stu_Fn,Stu_Ln,Prog_prog_ID,School_SCH_ID)
VALUES (787,"Rhea","Hernandez",3,3);

INSERT INTO STUDENT(Stu_ID, Stu_Fn,Stu_Ln,Prog_prog_ID,School_SCH_ID)
VALUES (720,"Miko","Lipayon",1,1);

INSERT INTO STUDENT(Stu_ID, Stu_Fn,Stu_Ln,Prog_prog_ID,School_SCH_ID)
VALUES (302,"Dave","Oliva",1,1);

INSERT INTO STUDENT(Stu_ID, Stu_Fn,Stu_Ln,Prog_prog_ID,School_SCH_ID)
VALUES (453,"Joseph","Zabala",2,3);

```
INSERT INTO STUDENT(Stu_ID, Stu_Fn,Stu_Ln,Prog_prog_ID,School_SCH_ID)
VALUES (987,"Mark","Zabala",1,3);
```

```
SELECT * FROM prog_tb;
```

```
MariaDB [LAB6_StudentProf_ABALOS_db]> SELECT * FROM prog_tb;
+-----+-----+
| Prog_prog_ID | prog_desc |
+-----+-----+
|          1   | CPE       |
|          2   | EEE       |
|          3   | CE        |
|          4   | CPE       |
|          5   | EE        |
|          6   | CE        |
+-----+-----+
6 rows in set (0.001 sec)
```

```
SELECT * FROM SCHOOL_tb;
```

```
SELECT * FROM STUDENT;
```

```
MariaDB [LAB_6_202A_STUDENT_PROFILE_DB]> SELECT * FROM STUDENT ORDER BY Stu_Ln;
+-----+-----+-----+-----+-----+
| Stu_ID | Stu_Fn | Stu_Ln | Prog_prog_ID | School_SCH_ID |
+-----+-----+-----+-----+-----+
|    303 | Mark  | Alday  |          1   |          1   |
|    544 | Michael | Castro |          1   |          2   |
|    567 | Jeric | Cruz   |          1   |          2   |
|    321 | Jeric | Cruz   |          3   |          2   |
|    563 | Mae   | De Guzman |          2   |          2   |
|    235 | Kevin | Falcon |          3   |          3   |
|    309 | Jam   | Gabion |          1   |          1   |
|    654 | Kevin | Gascon |          3   |          1   |
|    787 | Rhea  | Hernandez |          3   |          3   |
|    720 | Miko  | Lipayon |          1   |          1   |
|    302 | Dave  | Oliva  |          1   |          1   |
|    453 | Joseph | Zabala |          2   |          3   |
|    987 | Mark  | Zabala |          1   |          3   |
+-----+-----+-----+-----+-----+
13 rows in set (0.000 sec)
```

```
SELECT Stu_ID, Stu_Ln, Stu_Fn FROM STUDENT ORDER BY Stu_Ln;
```

```
MariaDB [LAB_6_202A_STUDENT_PROFILE_DB]> SELECT Stu_ID, Stu_Ln, Stu_Fn FROM STUDENT ORDER BY Stu_Ln;
```

Stu_ID	Stu_Ln	Stu_Fn
303	Alday	Mark
544	Castro	Michael
567	Cruz	Jeric
321	Cruz	Jeric
563	De Guzman	Mae
235	Falcon	Kevin
309	Gabion	Jam
654	Gascon	Kevin
787	Hernandez	Rhea
720	Lipayon	Miko
302	Oliva	Dave
453	Zabala	Joseph
987	Zabala	Mark

```
13 rows in set (0.000 sec)
```

SELECT DISTINCT Stu_Ln FROM STUDENT ORDER BY Stu_Ln;

```
MariaDB [LAB_6_202A_STUDENT_PROFILE_DB]> SELECT DISTINCT Stu_Ln FROM STUDENT ORDER BY Stu_Ln;
```

Stu_Ln
Alday
Castro
Cruz
De Guzman
Falcon
Gabion
Gascon
Hernandez
Lipayon
Oliva
Zabala

```
11 rows in set (0.000 sec)
```

SELECT COUNT(*) FROM STUDENT;

```
MariaDB [LAB_6_202A_STUDENT_PROFILE_DB]> SELECT COUNT(*) FROM STUDENT;
```

COUNT(*)
13

```
1 row in set (0.000 sec)
```

SELECT COUNT(DISTINCT Prog_prog_ID) FROM STUDENT;

```
MariaDB [LAB_6_202A_STUDENT_PROFILE_DB]> SELECT COUNT(DISTINCT Prog_prog_ID) FROM STUDENT;
```

COUNT(DISTINCT Prog_prog_ID)
3

```
1 row in set (0.000 sec)
```

SELECT MAX(Stu_ID) FROM STUDENT;

```

3000). Cannot delete or update a parent row: a foreign key constraint fails at line
MariaDB [LAB6_StudentProf_ABALOS_db]> SELECT MAX(Stu_ID) FROM STUDENT;
+-----+
| MAX(Stu_ID) |
+-----+
|          987 |
+-----+
1 row in set (0.001 sec)

```

SELECT MIN(Stu_ID) FROM STUDENT;

```

MariaDB [LAB6_StudentProf_ABALOS_db]> SELECT MIN(Stu_ID) FROM STUDENT;
+-----+
| MIN(Stu_ID) |
+-----+
|          235 |
+-----+
1 row in set (0.001 sec)

```

SELECT MIN(Prog_prog_ID) FROM prog_tb;

```

MariaDB [LAB6_StudentProf_ABALOS_db]> SELECT MIN(Prog_prog_ID) FROM prog_tb;
+-----+
| MIN(Prog_prog_ID) |
+-----+
|                  1 |
+-----+
1 row in set (0.001 sec)

```

SELECT Stu_Ln FROM STUDENT WHERE Stu_Ln LIKE "A%";

```

MariaDB [LAB6_StudentProf_ABALOS_db]> SELECT Stu_Ln FROM STUDENT WHERE Stu_Ln LIKE "
A%";
+-----+
| Stu_Ln |
+-----+
| Alday  |
+-----+
1 row in set (0.001 sec)

```

SELECT COUNT(DISTINCT Stu_Ln) FROM STUDENT WHERE Stu_Ln LIKE "%Z%";

```

MariaDB [LAB6_StudentProf_ABALOS_db]> SELECT COUNT(DISTINCT Stu_Ln) FROM STUDENT WHE
RE Stu_Ln LIKE "%Z%";
+-----+
| COUNT(DISTINCT Stu_Ln) |
+-----+
|                        4 |
+-----+
1 row in set (0.001 sec)

```

SELECT Stu_Fn, Stu_Ln FROM STUDENT WHERE Stu_Fn Like '%Z%' OR Stu_Ln LIKE '%Z%'
ORDER BY Stu_Fn, Stu_Ln ASC LIMIT 1;

```
MariaDB [LAB6_StudentProf_ABALOS_db]> SELECT Stu_Fn, Stu_Ln FROM STUDENT WHERE Stu_Fn
Like '%Z%' OR Stu_Ln LIKE '%Z%' ORDER BY Stu_Fn, Stu_Ln ASC LIMIT 1;
+-----+-----+
| Stu_Fn | Stu_Ln |
+-----+-----+
| Jeric  | Cruz   |
+-----+-----+
1 row in set (0.001 sec)
```

SELECT prog_desc FROM prog_tb ORDER BY prog_desc ASC LIMIT 2;

```
MariaDB [LAB6_StudentProf_ABALOS_db]> SELECT prog_desc FROM prog_tb ORDER BY prog_desc
ASC LIMIT 2;
+-----+
| prog_desc |
+-----+
| CE        |
| CE        |
+-----+
2 rows in set (0.001 sec)
```

SELECT prog_desc FROM prog_tb ORDER BY prog_desc ASC LIMIT 2;

```
MariaDB [LAB6_StudentProf_ABALOS_db]> SELECT prog_desc FROM prog_tb ORDER BY prog_desc
ASC LIMIT 2;
+-----+
| prog_desc |
+-----+
| CE        |
| CE        |
+-----+
2 rows in set (0.001 sec)
```

SELECT Stu_ID, Stu_Ln, Stu_Fn, School_SCH_ID FROM STUDENT WHERE School_SCH_ID = 3
ORDER BY Stu_Ln DESC LIMIT 3;

```
MariaDB [LAB6_StudentProf_ABALOS_db]> SELECT Stu_ID, Stu_L, Stu_Fn, School_SCH_ID FROM
STUDENT WHERE School_SCH_ID = 3 ORDER BY Stu_Ln DESC LIMIT 3;
ERROR 1054 (42S22): Unknown column 'Stu_L' in 'field list'
MariaDB [LAB6_StudentProf_ABALOS_db]> SELECT Stu_ID, Stu_Ln, Stu_Fn, School_SCH_ID FROM
STUDENT WHERE School_SCH_ID = 3 ORDER BY Stu_Ln DESC LIMIT 3;
+-----+-----+-----+-----+
| Stu_ID | Stu_Ln | Stu_Fn | School_SCH_ID |
+-----+-----+-----+-----+
| 453    | Zabala | Joseph | 3              |
| 987    | Zabala | Mark   | 3              |
| 787    | Hernandez | Rhea  | 3              |
+-----+-----+-----+-----+
3 rows in set (0.001 sec)
```


DISTINCT AND COUNT FUNCTION

If you want to know how many surnames has been saved in the database sorted from A –

Z. SELECT STUD_LN FROM STUDENT ORDER BY STUD_LN;

Question 1. How many entries has been returned in the query result set: 10

If you want to know how many surnames has been saved in the database sorted from A – Z with no repetition you needed to use the distinct

SELECT DISTINCT STUD_LN FROM STUDENT ORDER BY STUD_LN ;

Question 2. How many entries has been returned in the query result set: 10

If you want to return just the number of entries you can use the Function Count

SELECT COUNT(*) FROM table_name

Try to count how many entries in the table of student

SELECT COUNT(*) FROM STUDENT;

Or to omit repetition in a certain column we will use the distinct

SELECT COUNT(DISTINCT <COLUMN_NAME>) FROM TABLE_NAME;

we can check how many existing entries in the prog_prog_ID has been entered with no

repetition Select Count(Distinct prog_prog_id) from student;

Question 3. How many entries have been returned? 3

Question 4. What are those entries we are referring to ? District Values in the program ID

Question 5. Using Count Function write a query that can display the number of students involved in CRS_ID 2 in the STUDENT_HAS_CRS table?

The output should be 7.

FIRST and LAST FUNCTION

First function can display the first data in the table

```
SELECT FIRST(column_name) FROM table_name
```

I want to display who is the first person in the students table

```
SELECT FIRST(STUD_LN) from STUDENT;
```

Is there an error ? perhaps others may have because some SQL FUNCTION is not supported in MYSQL

So we will have a work around using LIMIT

```
SELECT * FROM STUDENT LIMIT 1;
```

If you want to show only first person with its first name and last name according to last name

```
SELECT STUD_LN, STUD_FN FROM STUDENT ORDER BY STUD_LN ASC LIMIT 1;
```

OUTPUT SHOULD BE **ALDAY MARK**

AND THE LAST PERSON since we MYSQL cannot support the LAST() Function this is the work around

```
SELECT STUD_LN, STUD_FN FROM STUDENT ORDER BY STUD_LN DESC LIMIT 1; Output
```

should be Mark Zabala.

MAX AND MIN FUNCTION

Max to get the highest value of a certain column from the table

```
SELECT MAX(column_name) FROM table_name
```

Min to get the lowest value of a certain column from a table

```
SELECT MIN(column_name) FROM table_name
```

Question 6. USING the max function who is the person who has the highest student ID

ID-787(HERNANDEZ,RHEA)

Question 7. USING the min function who is the person who has the lowest student ID

ID-253 (FALCON, KEVIN)

Question 8. Who has the highest PROF_ID

D-787 (HERNANDEZ,RHEA)

Question 9. What is the course that has the lowest course id

ID-1 (CPE)

THE SUM FUNCTION

Can compute the total of the columns specified

```
SELECT SUM(column_name) FROM table_name
```

However there are no data that we needed to compute at this point

THE TOP CLAUSE/FUNCTION

Top clause is use to get the values as TOP 10 or TOP 5 or whatever values you want .

But it is also not supported in MySQL. Thus the work around here is to set the limit again

If I want to get the top 5 according to the student ID

```
SELECT * FROM STUDENT ORDER BY STUD_ID ASC LIMIT 5
```

THE LIKE OPERATOR

```
SELECT column_name(s)  
FROM table_name  
WHERE column_name LIKE pattern
```

This is commonly used for searching

As for example, I want to know the students whose last name ends with Z

```
SELECT STUD_LN FROM STUDENT  
  
WHERE STUD_LN LIKE "%z";
```

So the % sign is use to check all characters before z but if you want to know the student who has letter z in their lastname

SELECT STUD_LN FROM STUDENT WHERE STUD_LN LIKE “%z%”;

Question 10. What will be the query to get the names of the student that has last names starting with letter A

SELECT Stu_Ln FROM STUDENT WHERE Stu_Ln LIKE “A%”;

Question 11. What will be the query that can COUNT how many student has letter z in their last name.

SELECT COUNT(DISTINCT Stu_Ln) FROM STUDENT WHERE Stu_Ln LIKE “%Z%”;

Question 12. What will be the query that can return the first person that has z in his name sorted in ascending order according to their names

**SELECT Stu_Fn, Stu_Ln FROM STUDENT WHERE Stu_Fn Like ‘%Z%’ OR Stu_Ln LIKE ‘%Z%’
ORDER BY Stu_Fn, Stu_Ln ASC LIMIT 1;**

Question 13. What is the query that can sort the list of top 2 Courses according to the name of the COURSE

SELECT prog_desc FROM prog_tb ORDER BY prog_desc ASC LIMIT 2;

Question 14. What is the query that can display the top 2 Program listed according to their names

SELECT prog_desc FROM prog_tb ORDER BY prog_desc ASC LIMIT 2;

Question 15. What will be the query so that it will display the TOP 3 Students enrolled in TIP LAGUNA sorted according to their last name in descending order.

**SELECT Stu_ID, Stu_Ln, Stu_Fn, School_SCH_ID FROM STUDENT WHERE School_SCH_ID = 3
ORDER BY Stu_Ln DESC LIMIT 3;**

JOIN OPERATORS

The JOIN keyword is used in an SQL statement to query data from two or more tables, based on a relationship between certain columns in these tables.

- **JOIN:** Return rows when there is at least one match in both tables
- **LEFT JOIN:** Return all rows from the left table, even if there are no matches in the right table
- **RIGHT JOIN:** Return all rows from the right table, even if there are no matches in the left table
- **FULL JOIN:** Return rows when there is a match in one of the tables

The join condition is generally composed of an equality comparison between the foreign key and the primary key of related tables.

Let us first review the values on our tables

The Inner Join

**SELECT column_name(s)
FROM table_name1
INNER JOIN table_name2**

ON table_name1.column_name=table_name2.column_name

The INNER JOIN keyword return rows when there is at least one match in both tables. If there are rows in "Persons" that do not have matches in "Orders", those rows will NOT be listed.

So assuming I want to concatenate the table of the PROG **FROM** the table student

```
SELECT *FROM Student
INNER JOIN prog
ON Student.prog_prog_id=prog.prog_id
ORDER BY student.stud_In
```

SELECT * FROM -> (*The original table*)

INNER JOIN prog ON Student.prog_prog_id=prog.prog_id

Question 16. What column will be check on the original and the concatenated one?

Prog_prog_ID and School_SCH_ID

If I wanted to print the list of names of the student according to their last name and also their corresponding Program

```
SELECT Student.stud_In, Student.stud_fn, prog.prog_desc
FROM Student
INNER JOIN prog
ON Student.prog_prog_id=prog.prog_id
ORDER BY student.stud_In;
```

Question 17. What will be the proper idea? Think of the table that you needed to retain or the original table that you are actually up to

1. You have a list of students.
2. You also have a list of programs.
3. You want to know which students are in which programs.

So, you use a method called INNER JOIN to combine these lists and find out which student is in which program. This way, you don't change the original student list, but you can see which programs they're a part of.

```
SELECT Student.stud_In, Student.stud_fn, prog.prog_desc
FROM Student
INNER JOIN prog
ON Student.prog_prog_id=prog.prog_id
ORDER BY student.stud_In;
```

Think of something you want to show on that table and on the other table

```
SELECT Student.stud_In, Student.stud_fn, prog.prog_desc
FROM Student
INNER JOIN prog
ON Student.prog_prog_id=prog.prog_id
ORDER BY student.stud_In;
```

If you do have the original what table will be concatenated

```
SELECT Student.stud_In, Student.stud_fn, prog.prog_desc
FROM Student
INNER JOIN proge
ON Student.prog_prog_id=prog.prog_id
ORDER BY student.stud_In;
```

Question 18. What columns are equal from the Original to the concatenated one

```
SELECT Student.stud_In, Student.stud_fn, prog.prog_desc
FROM Student
INNER JOIN prog
ON Student.prog_prog_id=prog.prog_id
ORDER BY student.stud_In;
```

Note: *Everything in the column of the original table will be looked upon on the column of the concatenated table, hence if from the original it was not found in the concatenated that data will not be listed. And those data from the concatenated columns which do not also appear in the list.*

Question 19. What instance the SQL Join operators are needed? Cite an example on a real word application and scenario where the Join Operators are needed.

7. Conclusion:

Based from the results of the activity, what general rule can you apply for?

Think of DML (Data Manipulation Language) as the tool for changing data in a database. It's like adding, editing, or deleting information in a spreadsheet.

On the other hand, SQL SELECT functions are like a magnifying glass that helps you look at and analyze the data without changing it. It's the tool for reading and understanding the data, like sorting, searching, or doing calculations on the information in your spreadsheet.

In conclusion, DML is for changing data, while SQL SELECT is for looking at and working with data without making any changes.

8. Assessment (Rubric for Laboratory Performance):

Intended Learning	Unsatisfactory	Satisfactory	Exemplary	Score
<u>Outcomes</u>	<u>1</u>	<u>2</u>		<u>3</u>
Identifies different	Unable identifies	Able to Identifies	Able to Identifies	
Data Manipulation	different Data	different Data	different Data	
Language that will	Manipulation	Manipulation	Manipulation	
be the basis of the	Language that	Language that will be	Language that will	
DFD				

	will be the basis	the basis of the DFD	be the basis of the
	of the DFD	but with errors	DFD
	Unable explain	Able to unable	Able to unable
Explain different	different	explain different	explain different
functions of MySQL	functions of	functions of MySQL	functions of
DML	<u>MySQL DML</u>	<u>DML but with errors</u>	<u>MySQL DML</u>
	Unable to apply	Apply the knowledge	Apply the
Apply the	the knowledge of	of DML in	knowledge of DML
knowledge of DML	DML in	manipulating	in manipulating
in manipulating	manipulating	databases and other	databases and
databases and	databases and	select functions but	other select
other select	other select	with errors	functions
functions	<u>functions</u>		
Other comments/observation: <u>Total Score</u>			

$$\text{RATING} = \frac{(\text{total score})}{6} \times 100\%$$

Evaluated by: Date:

_____ Printed Name and Signature of Faculty Member