Activity No. 4	
Advanced SELECT commands	
Course Code: CPE011	**Program: BSCPE**
Course Title: Database Management System	**Date Performed: 9/9/2022**
Section: BSCPE21S3	**Date Submitted: 9/9/2022**
Name: **Aaron Martin Parallag Caro** **Christian Efa**	**Instructor: Dr.Jonathan V. Taylar**

1. Objective(s):

This activity aims to introduce the advanced techniques for selecting records in a MySQL Database using the SQL SELECT Command.

2. Intended Learning Outcomes (ILOs):

The students should be able to:
2.1 Select data from a database based on a specific criterion
2.2 View data using select statements partnered with wildcards
2.2 Select data from a database in a specific order
2.3 View data using select statements partnered with wildcards

3. Discussion:

The SELECT Statement's basic syntax allows you to retrieve rows either all or only specific columns. In this activity you will see the different additional features that you can use with the SELECT statement through SQL Clauses and a special COUNT() function.

1. WHERE Clause

In computing and database technology, conditional statements, conditional expressions and conditional constructs are features of a programming language, which perform different computations or actions depending on whether a programmer-specified Boolean condition evaluates to true or false. Apart from the case of branch predication, this is always achieved by selectively altering the control flow based on some condition.

The SQL WHERE clause is used to specify a condition while fetching the data from single table or joining with multiple tables.

If the given condition is satisfied then only it returns specific value from the table. You would use WHERE clause to filter the records and fetching only necessary records.

The WHERE clause is not only used in SELECT statement, but it is also used in UPDATE, DELETE statement, etc., which we would examine in subsequent chapters.

```
SELECT column_name,column_name
FROM table_name
WHERE column_name operator value;
```

SQL requires single quotes around text values (most database systems will also allow double quotes). However, numeric fields should not be enclosed in quotes:

```
SELECT * FROM CustomerWHERE CustomerID=1;
```

Operators in The WHERE Clause The following operators can be used in the WHERE clause:

Operator	Description
=	Equal
<>	Not equal. **Note:** In some versions of SQL this operator may be written as !=
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

An example of an application for conditional statements are as follows:

```
SELECT * FROM Customer WHERE CustomerID> 1;
```

The LIKE operator
The LIKE operator is used in a WHERE clause to search for a specified pattern in a column. The syntax is as follows.

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

Using Wildcards
In using the LIKE pattern, wildcard characters are used with the SQL LIKE operator. SQL wildcards are used to search for data within a table. With SQL, the wildcards are:

Wildcard	Description
%	A substitute for zero or more characters
_	A substitute for a single character
[*charlist*]	Sets and ranges of characters to match
[^*charlist*] or [!*charlist*]	Matches only a character NOT specified within the brackets

Using the SQL % Wildcard
The % wildcard symbol is used to substitute any character, with any count. The following SQL statement selects all data from the student table with a program starting with "elec":

```
SELECT * FROM student
WHERE program LIKE 'elec%';
```

Any number of wildcards can be used in a single query. In the following example, two % wildcards are used. To search for data with an 'er' in between, use the following syntax.

```
SELECT * FROM student
WHERE program LIKE '%er%';
```

Using the SQL _ Wildcard
The _ wildcard is used to substitute any single character. The following SQL statement selects all customers with a City starting with any character, followed by "erlin":

```
SELECT * FROM Customers
WHERE City LIKE '_erlin';
```

The following SQL statement selects all customers with a City starting with "L", followed by any character, followed by "n", followed by any character, followed by "on":

```
SELECT * FROM Customers
WHERE City LIKE 'L_n_on';
```

Using the SQL [charlist] Wildcard

The [charlist] sets and ranges of characters to match The following SQL statement selects all customers with a City starting with "b", "s", or "p":

```
SELECT * FROM Customers
WHERE City LIKE '[bsp]%';
```

Ranges can also be utilized for the charlist wildcard. These ranges can be indicated using a hypen(-).The following SQL statement selects all customers with a City starting with "a", "b", or "c":

```
SELECT * FROM Customers
WHERE City LIKE '[a-c]%';
```

The NOT symbol, represented by an exclamation point, can be used alongside the charlist wildcard to indicate a query that does not contain any of the indicated. The following SQL statement selects all customers with a City NOT starting with "b", "s", or "p":

```
SELECT * FROM Customers
WHERE City LIKE '[!bsp]%';
```

2. ORDER BY Clause

Displaying data in a particular order

The ORDER BY keyword is used to sort the result-set by one or more columns. The ORDER BY keyword sorts the records in ascending order by default. To sort the records in a descending order, you can use the DESC keyword. The syntax for selecting based on a single column:

```
SELECT column_name, column_name
FROM table_name
ORDER BY column_name ASC|DESC;
```

For selecting data and ordering based on multiple columns, the syntax can be used:

```
SELECT column_name, column_name
FROM table_name
ORDER BY column_name ASC|DESC, column_name ASC|DESC;
```

3. LIMIT Clause

Limiting only a specific number of results

MySQL provides a LIMIT clause that is used to specify the number of records to return. The LIMIT clause makes it easy to code multi page results or pagination with SQL, and is very useful on large tables. Returning a large number of records can impact on performance. The syntax is as follows:

```
SELECT * FROM table LIMIT 30;
```

4. COUNT FUNCTION()

Counting Data

The COUNT() function returns the number of rows that matches a specified criteria. The syntax is as follows:

SELECT COUNT(column_name) FROM table_name;

Note that the count command can utilize conditions similar to the where clause. This can be seen in the following syntax below.

SELECT COUNT(CustomerID) AS OrdersFromCustomerID FROM Orders WHERE CustomerID=7;

Note that the AS command is used to create a result column that is named OrdersFromCustomerID. This name will be used as an identifier for the count value.

To count only data with unique values, use the following query:

SELECT COUNT(DISTINCT column_name) FROM table_name;

Note that the distinct values are case sensitive.

4. Materials and Equipment:

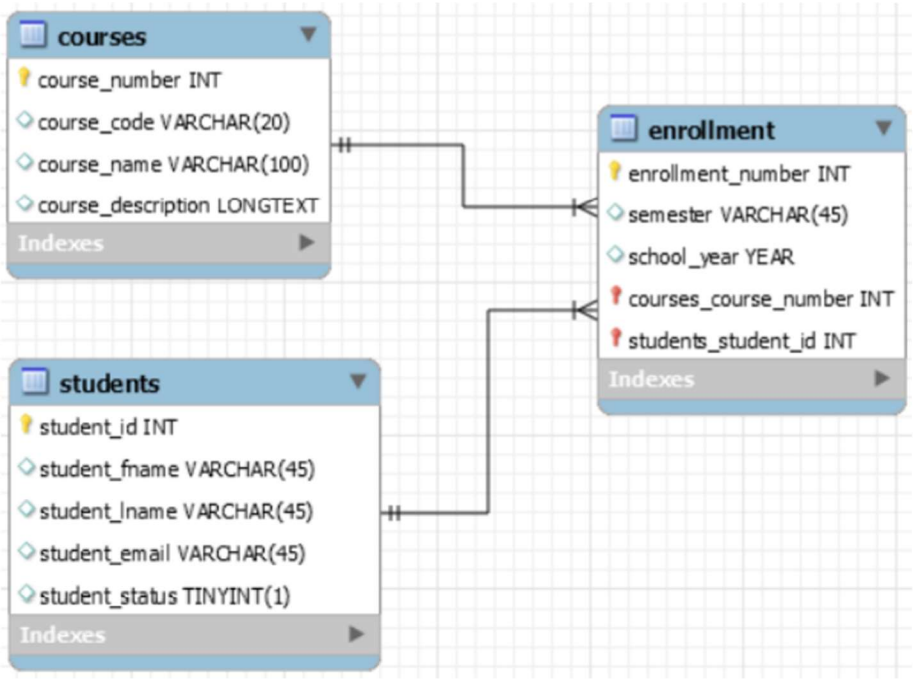
Desktop Computer
Windows Operating System
XAMPP Application

5. Procedure

Database Schema

The database that we'll be using for this activity is the simple enrollment database from the previous activity discussion. We will perform SQL commands that allow us to retrieve data based on specific requirements/conditions.

The image below shows the database schema.



Tasks

1. Create a database named enrollmentdb_<LASTNAME>
   Note: Use the techniques you've learned in the previous activity to implement a database that handles redundancy.
2. Populate the database to have at least 10 records each.
   Side activity:
   You can load pre-made values by loading data from CSV instead of manually inserting them.
   1. Copy the students.csv, courses.csv to your computer from the FTP or Canvas Instructure group. For the enrollment.csv, you will be creating your based on the pattern of the two files.
   **Take note of the order in which the table columns were created and the order of values in your csv files.

   2. Follow the pattern of the following command to load each csv files to the respective tables.

```
LOAD DATA INFILE 'C:/Users/Royce/Desktop/students.csv'
INTO TABLE students
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

   3. Take screenshots of the values in each table. Write descriptions for each image, explain the table and its values.
   4. For the succeeding steps take screenshots of each record displayed by the SELECT commands. Write your observations under each image.
   5. Select the row in course that contains the course code cpe011 along with its course description.
   6. Select the row of student number 7.
   7. Select the rows in students table whose student email ends in yahoo.com
   8. Select the rows in students table whose first name starts with d or D.
   9. Select the rows in students table who are not currently active.
   10. Select the first five rows in the courses table.
   11. Display the rows/course(s) which includes the words programming in courses table.
   12. Display the row/course which teaches VHDL.
   13. Display the courses that were enrolled during school year 2019.
   14. Display the student records in alphabetical order.
   15. Count how many students are currently not active. What is the answer? _____

6. Database Output

Copy screenshot(s) of your output with observations after completing the procedures provided in Part 5.

1. Create a database named enrollmentdb_<LASTNAME>

```
MariaDB [(none)]> create database enrollmentdb_Caro_Efa
    -> ;
Query OK, 1 row affected (0.002 sec)

MariaDB [(none)]> use enrollmentdb_Caro_Efa
Database changed
```

Note: Use the techniques you've learned in the previous activity to implement a database that handles redundancy.

2. Populate the database to have at least 10 records each.

Side activity:

You can load pre-made values by loading data from CSV instead of manually inserting them.

1. Copy the students.csv, courses.csv to your computer from the FTP or Canvas Instructure group. For the enrollment.csv, you will be creating your based on the pattern of the two files.

**Take note of the order in which the table columns were created and the order of values in your csv files.

2. Follow the pattern of the following command to load each csv files to the respective tables.

3. Take screenshots of the values in each table. Write descriptions for each image, explain the table and its values.

```
MariaDB [enrollmentdb_Caro_Efa]> create table students
    -> (student_id INT PRIMARY KEY,
    -> student_fname varchar(45),
    -> student_lname varchar(45),
    -> student_email varchar(45),
    -> student_status TINYINT(1));
Query OK, 0 rows affected (0.026 sec)

MariaDB [enrollmentdb_Caro_Efa]> describe students
    -> ;
+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| student_id     | int(11)     | NO   | PRI | NULL    |       |
| student_fname  | varchar(45) | YES  |     | NULL    |       |
| student_lname  | varchar(45) | YES  |     | NULL    |       |
| student_email  | varchar(45) | YES  |     | NULL    |       |
| student_status | tinyint(1)  | YES  |     | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
5 rows in set (0.009 sec)
```

```
MariaDB [enrollmentdb_Caro_Efa]> create table courses
    -> (course_number INT PRIMARY KEY,
    -> course_code varchar(255),
    -> course_name varchar(255),
    -> course_description LONGTEXT);
Query OK, 0 rows affected (0.017 sec)

MariaDB [enrollmentdb_Caro_Efa]> describe courses;
+--------------------+--------------+------+-----+---------+-------+
| Field              | Type         | Null | Key | Default | Extra |
+--------------------+--------------+------+-----+---------+-------+
| course_number      | int(11)      | NO   | PRI | NULL    |       |
| course_code        | varchar(255) | YES  |     | NULL    |       |
| course_name        | varchar(255) | YES  |     | NULL    |       |
| course_description | longtext     | YES  |     | NULL    |       |
+--------------------+--------------+------+-----+---------+-------+
4 rows in set (0.004 sec)
```

```
MariaDB [enrollmentdb_Caro_Efa]> create table enrollment
    -> (enrollment_number INT PRIMARY KEY,
    -> semester varchar(255),
    -> school_year YEAR,
    -> course_number INT,
    -> student_id INT,
    -> FOREIGN KEY (course_number) REFERENCES courses(course_number),
    -> FOREIGN KEY (student_id) REFERENCES students(student_id));
Query OK, 0 rows affected (0.022 sec)

MariaDB [enrollmentdb_Caro_Efa]> describe enrollment;
+-------------------+-------------+------+-----+---------+-------+
| Field             | Type        | Null | Key | Default | Extra |
+-------------------+-------------+------+-----+---------+-------+
| enrollment_number | int(11)     | NO   | PRI | NULL    |       |
| semester          | varchar(255)| YES  |     | NULL    |       |
| school_year       | year(4)     | YES  |     | NULL    |       |
| course_number     | int(11)     | YES  | MUL | NULL    |       |
| student_id        | int(11)     | YES  | MUL | NULL    |       |
+-------------------+-------------+------+-----+---------+-------+
5 rows in set (0.007 sec)
```

```
MariaDB [enrollmentdb_Caro_Efa]> LOAD DATA INFILE 'C:/Users/Aaron/Downloads/students.csv'
    -> INTO TABLE students
    -> FIELDS TERMINATED BY ','
    -> ENCLOSED BY '"'
    -> LINES TERMINATED BY '\n'
    -> IGNORE 1 ROWS;
Query OK, 10 rows affected, 10 warnings (0.008 sec)
Records: 10  Deleted: 0  Skipped: 0  Warnings: 10

MariaDB [enrollmentdb_Caro_Efa]> select * from students;
+------------+--------------+--------------+--------------------------+----------------+
| student_id | student_fname | student_lname | student_email            | student_status |
+------------+--------------+--------------+--------------------------+----------------+
|          1 | Daniel       | Oneal        | daneal@gmail.com         |              1 |
|          2 | Michael      | Inciong      | minciong12@gmail.com     |              0 |
|          3 | Paulo        | Calintong    | palintong90@gmail.com    |              1 |
|          4 | Jasmine      | Reyes        | jasmineryes@gmail.com    |              1 |
|          5 | Rommel       | Castor       | romastor@gmail.com       |              0 |
|          6 | John         | Wallis       | jowalis@gmail.com        |              1 |
|          7 | Angel        | Cruz         | angela@ymail.com         |              0 |
|          8 | mucha        | tuwa         | muwa@yahoo.com           |              0 |
|          9 | kyle         | magnifico    | kyle_magnifico@yahoo.com |              0 |
|         10 | milo         | campo        | mico123@gmail.com        |              1 |
+------------+--------------+--------------+--------------------------+----------------+
10 rows in set (0.011 sec)
```

```
MariaDB [enrollmentdb_Caro_Efa]> LOAD DATA INFILE 'C:/Users/Aaron/Downloads/courses.csv'
    -> INTO TABLE courses
    -> FIELDS TERMINATED BY ','
    -> ENCLOSED BY '"'
    -> LINES TERMINATED BY '\n'
    -> IGNORE 1 ROWS;
Query OK, 10 rows affected (0.003 sec)
Records: 10  Deleted: 0  Skipped: 0  Warnings: 0

MariaDB [enrollmentdb_Caro_Efa]> SELECT * FROM courses;
+---------------+-------------+-----------------------------------+-----------------------------------------------------------------------------+
| course_number | course_code | course_name                       | course_description                                                          |
+---------------+-------------+-----------------------------------+-----------------------------------------------------------------------------+
|             1 | CPE011      | Database Management Systems       | This course                                                                 |
|             2 | CPE007      | Programming Logic and Design      | This course                                                        |ming.
|             3 | CPE010      | Data Structures and Algorithms    | This course                                                                 |
|             4 | CPE008      | Computer Engineering as a Discipline | This course                                                      |n overview to Computer Engineering.
|             5 | CPE009      | Object-oriented programming       | This course                                                              |ienting programming.
|             6 | ECE006A     | Feedback and Control Systems      | This course                                                                 |
|             7 | CPE402      | Advanced Logic Circuits           | This course                                                     |hardware design using FPGAs and VHDL.
|             8 | CPE411      | Systems Analysis and Design       | This course                                                    | the IT Systems and how to design them.
|             9 | CPE505      | Design Project 1                  | This course                                             | the one of the final culminating courses for CPE students.
|            10 | CPE003      | Computer Aided Drafting           | This course                                                                 |
+---------------+-------------+-----------------------------------+-----------------------------------------------------------------------------+
10 rows in set (0.000 sec)
```

4. For the succeeding steps take screenshots of each record displayed by the SELECT commands. Write your observations under each image.

The values inputted in the student table is neatly placed in each columns , in contrast, the values in enrollment table to be precise beneath the course_description column the values are jumbled.

5. Select the row in course that contains the course code cpe011 along with its course description.

```
MariaDB [enrollmentdb_Caro_Efa]> select * from courses where course_code = 'CPE011';
+---------------+-------------+-----------------------+-----------------------------------+
| course_number | course_code | course_name           | course_description                |
+---------------+-------------+-----------------------+-----------------------------------+
|             1 | CPE011      | Database Management Systems | This course is all about databases. |
+---------------+-------------+-----------------------+-----------------------------------+
1 row in set (0.001 sec)
```

6. Select the row of student number 7.

```
MariaDB [enrollmentdb_Caro_Efa]> select * from students where student_id = 7;
+------------+---------------+---------------+-------------------+----------------+
| student_id | student_fname | student_lname | student_email     | student_status |
+------------+---------------+---------------+-------------------+----------------+
|          7 | Angel         | Cruz          | angela@ymail.com |              0 |
+------------+---------------+---------------+-------------------+----------------+
1 row in set (0.001 sec)
```

7. Select the rows in students table whose student email ends in yahoo.com

```
MariaDB [enrollmentdb_Caro_Efa]> Select * from students WHERE student_email like "%yahoo.com";
+------------+---------------+---------------+--------------------------+----------------+
| student_id | student_fname | student_lname | student_email            | student_status |
+------------+---------------+---------------+--------------------------+----------------+
|          8 | mucha         | tuwa          | muwa@yahoo.com           |              0 |
|          9 | kyle          | magnifico     | kyle_magnifico@yahoo.com |              0 |
+------------+---------------+---------------+--------------------------+----------------+
2 rows in set (0.002 sec)
```

8. Select the rows in students table whose first name starts with d or D.

```
MariaDB [enrollmentdb_Caro_Efa]> Select * from students WHERE student_fname like "D%" or "d%"
+------------+---------------+---------------+-------------------+----------------+
| student_id | student_fname | student_lname | student_email     | student_status |
+------------+---------------+---------------+-------------------+----------------+
|          1 | Daniel        | Oneal         | daneal@gmail.com |              1 |
+------------+---------------+---------------+-------------------+----------------+
1 row in set, 5 warnings (0.001 sec)
```

9. Select the rows in students table who are not currently active.

```
MariaDB [enrollmentdb_Caro_Efa]> SELECT * FROM students where student_status = 0;
+------------+---------------+---------------+---------------------------+----------------+
| student_id | student_fname | student_lname | student_email             | student_status |
+------------+---------------+---------------+---------------------------+----------------+
|          2 | Michael       | Inciong       | minciong12@gmail.com      |              0 |
|          5 | Rommel        | Castor        | romastor@gmail.com        |              0 |
|          7 | Angel         | Cruz          | angela@ymail.com          |              0 |
|          8 | mucha         | tuwa          | muwa@yahoo.com            |              0 |
|          9 | kyle          | magnifico     | kyle_magnifico@yahoo.com  |              0 |
+------------+---------------+---------------+---------------------------+----------------+
5 rows in set (0.001 sec)
```

10. Select the first five rows in the courses table.

```
MariaDB [enrollmentdb_Caro_Efa]> SELECT * FROM courses where course_number between 1 and 5;
+--------------+-------------+-----------------------------------+-----------------------------------------------------------------+
| course_number | course_code | course_name                       | course_description                                              |
+--------------+-------------+-----------------------------------+-----------------------------------------------------------------+
|              | | Database Management Systems          | This course is all about databases.
|            | CPE007      | Programming Logic and Design      | This course is an introduction to programming.
|                    |   | Data Structures and Algorithms    | This course is all about algorithms.
|          4 | CPE008      | Computer Engineering as a Discipline | This course is all about an overview to Computer Engineering.
|        |   5 | CPE009      | Object-oriented programming       | This course is all about object oriented programming.
+--------------+-------------+-----------------------------------+-----------------------------------------------------------------+
5 rows in set (0.003 sec)
```

11. Display the rows/course(s) which includes the words programming in courses table.

```
MariaDB [enrollmentdb_Caro_Efa]> SELECT * FROM courses WHERE course_name or course_description LIKE "%programming%";
+---------------+-------------+---------------------------+-----------------------------------------------------------+
| course_number | course_code | course_name               | course_description                                        |
+---------------+-------------+---------------------------+-----------------------------------------------------------+
|             2 | CPE007      | Programming Logic and Design | This course is an in        |n to programming.
|             5 | CPE009      | Object-oriented programming  | This course is all a |ut object oriented programming.
+---------------+-------------+---------------------------+-----------------------------------------------------------+
2 rows in set, 10 warnings (0.001 sec)
```

12. Display the row/course which teaches VHDL.

```
MariaDB [enrollmentdb_Caro_Efa]> SELECT * FROM courses WHERE course_name or course_description LIKE "%VHDL%";
+---------------+-------------+-------------------------+-----------------------------------------------------------------+
| course_number | course_code | course_name             | course_description                                              |
+---------------+-------------+-------------------------+-----------------------------------------------------------------+
|             7 | CPE402      | Advanced Logic Circuits | This course is all about hardware design using FPGAs and VHDL.  |
+---------------+-------------+-------------------------+-----------------------------------------------------------------+
1 row in set, 10 warnings (0.000 sec)
```

13. Display the courses that were enrolled during school year 2019.

```
MariaDB [enrollmentdb_Caro_Efa]> INSERT INTO enrollment value
    -> (1,'1st',2019, 1,1),
    -> (2,'1st',2019, 2,2),
    -> (3,'1st',2019, 3,3),
    -> (4,'1st',2019, 4,4),
    -> (5,'1st',2019, 5,5),
    -> (6,'2nd',2020, 6,6),
    -> (7,'2nd',2020, 7,7),
    -> (8,'2nd',2020, 8,8),
    -> (9,'2nd',2020, 9,9),
    -> (10,'2nd',2020, 10,10);
Query OK, 10 rows affected (0.004 sec)
Records: 10  Duplicates: 0  Warnings: 0

MariaDB [enrollmentdb_Caro_Efa]> SELECT * FROM enrollment where school_year = 2019;
+-------------------+----------+-------------+---------------+------------+
| enrollment_number | semester | school_year | course_number | student_id |
+-------------------+----------+-------------+---------------+------------+
|                 1 | 1st      |        2019 |             1 |          1 |
|                 2 | 1st      |        2019 |             2 |          2 |
|                 3 | 1st      |        2019 |             3 |          3 |
|                 4 | 1st      |        2019 |             4 |          4 |
|                 5 | 1st      |        2019 |             5 |          5 |
+-------------------+----------+-------------+---------------+------------+
5 rows in set (0.002 sec)
```

14. Display the student records in alphabetical order.

```
MariaDB [enrollmentdb_Caro_Efa]> SELECT student_id, student_status, student_lname, student_fname, student_email from students ORDER BY student_lname ASC;
+------------+----------------+-------------+-------------+-----------------------+
| student_id | student_status | student_lname | student_fname | student_email         |
+------------+----------------+-------------+-------------+-----------------------+
|          3 |              1 | Calintong   | Paulo       | palintong90@gmail.com |
|         10 |              1 | campo       | milo        | mico123@gmail.com     |
|          5 |              0 | Castor      | Rommel      | romastor@gmail.com    |
|          7 |              0 | Cruz        | Angel       | angela@ymail.com      |
|          2 |              0 | Inciong     | Michael     | minciong12@gmail.com  |
|          9 |              0 | magnifico   | kyle        | kyle_magnifico@yahoo.com |
|          1 |              1 | Oneal       | Daniel      | daneal@gmail.com      |
|          4 |              1 | Reyes       | Jasmine     | jasmineryes@gmail.com |
|          8 |              0 | tuwa        | mucha       | muwa@yahoo.com        |
|          6 |              1 | Wallis      | John        | jowalis@gmail.com     |
+------------+----------------+-------------+-------------+-----------------------+
10 rows in set (0.000 sec)
```

15. Count how many students are currently not active. What is the answer? 5

```
MariaDB [enrollmentdb_Caro_Efa]> SELECT COUNT(student_status) FROM students WHERE student_status = 0;
+-----------------------+
| COUNT(student_status) |
+-----------------------+
|                     5 |
+-----------------------+
1 row in set (0.003 sec)
```

7. Supplementary Activity:

Questions

1. Why do you think the SQL WHERE clause exists? Based from the activity, explain its importance.
   - When retrieving data from a single table or by joining data from many tables, a condition is specified using the SQL WHERE clause. Based on activity it is easier to use WHERE clause to determine or fetch the data on a single or multiple tables.

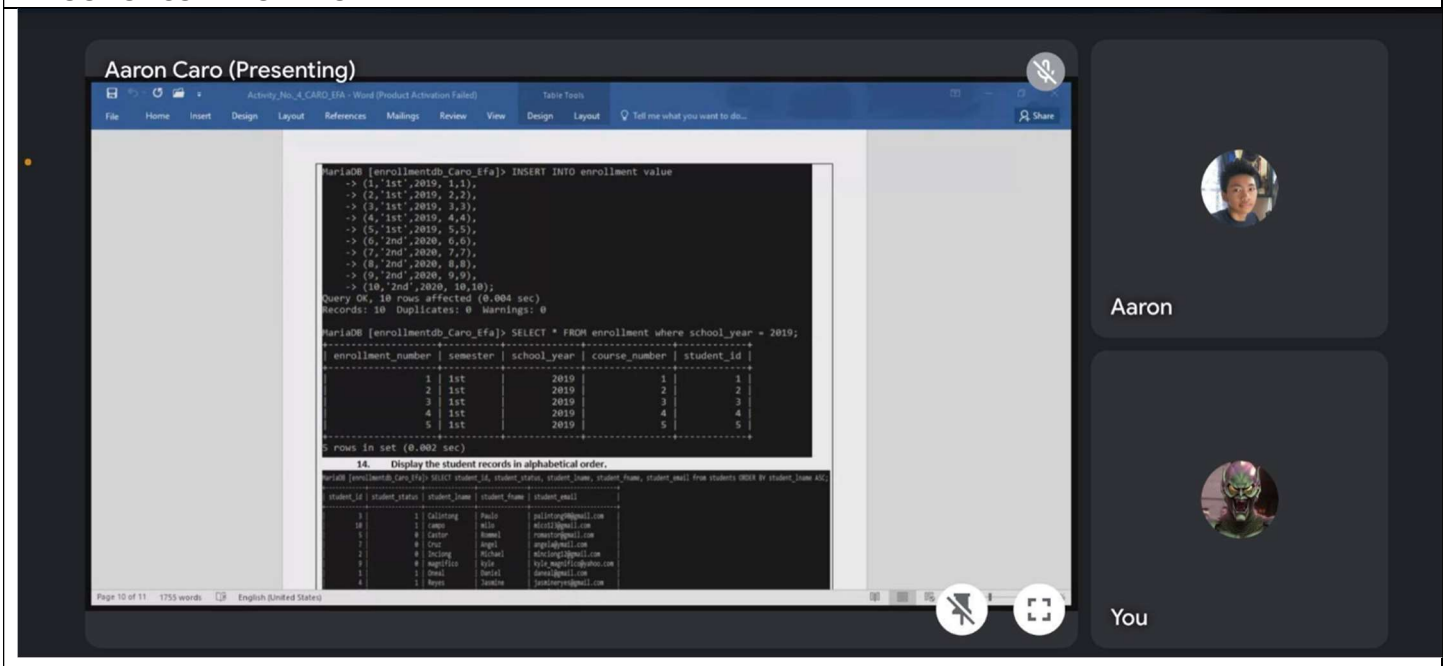
2. Where do we use SQL wildcards? Cite a use-case not based from the activity.
   - wildcard character is used to substitute one or more characters in a string.

3. What is the purpose of LIMIT clauses in database queries?
   - The number of records to return is determined by the LIMIT clause.
4. Can ORDER clause be used for multiple columns? How?
   - This clause can be used with multiple columns, To separate an order clause, you just need to use a comma operator.
5. What particular situation can the COUNT() function be used? Use an example not based in the activity.
   - to get the number of entries in a number field that is in a range or array of numbers.

8. Conclusion

We conclude that we can use many different clauses to change the behavior of the SELECT statement. The SELECT statement is probably the most important SQL command. It's used to return results from our database(s) and no matter how easy that could sound, it could be really very complex.

PROOF OF COLLABORATION



HONOR PLEDGE

"We accept responsibility for our role in ensuring the integrity of the work submitted by the group in which we participated."s