# Database Management Systems

ICT1222

## MySQL

Department of ICT
Faculty of Technology
University of Ruhuna

Practical 03

# What we discuss Today ……..

- Review
- Answers to Exercise
- More on SELECT statement
- NULL
- Exercise

# SQL Constraints

- NOT NULL
- CHECK
- DEFAULT
- PRIMARY KEY
- AUTO_INCREMENT
- UNIQUE
- INDEX
- ENUM
- FOREIGN KEY

# NOT NULL Constraint

```
CREATE TABLE Student
(Id INT,
LastName TEXT NOT NULL,
FirstName TEXT NOT NULL,
City VARCHAR(35)
);

INSERT INTO Student VALUES(1, 'Hanks', 'P
eter', 'New York');
```

# UNIQUE Constraint

```
CREATE TABLE ShirtBrands
(Id INT,
BrandName VARCHAR(40) UNIQUE,
Size VARCHAR(30)
);

INSERT INTO ShirtBrands
(Id, BrandName, Size)
VALUES(1, 'Pantaloons', 38), (2, 'Cantabil', 40);
```

# PRIMARY KEY Constraint

```
CREATE TABLE Persons (
   ID int NOT NULL PRIMARY KEY,
   Name varchar(45) NOT NULL,
   Age int,
   City varchar(25));

INSERT INTO Persons(Id, Name, Age, City)
VALUES (1,'Robert', 15, 'Florida') ,
(2, 'Joseph', 35, 'California'),
(3, 'Peter', 40, 'Alaska');
```

# SQL FOREIGN KEY Constraint

- A foreign key is a column (or columns) that refer a column (most often the primary key) of another table.
- The purpose of the foreign key is to ensure referential integrity of the data.

# SQL FOREIGN KEY Constraint

```
CREATE TABLE table_name
  (
        Column-definitions,
        primaryKey-definition,
        foreignkey- definition
  );
```

# SQL FOREIGN KEY Constraint

```
CREATE TABLE table_name (
    column1,
    column 2,
    column3,
    PRIMARY KEY (column1),
    FOREIGN KEY (column2) REFERENCES
owner_table_name (column_name)
);
```

# Foreign Key Constraint

```
CREATE TABLE Persons (
    Person_ID int NOT NULL PRIMARY KEY,
    Name varchar(45) NOT NULL,
    Age int,
    City varchar(25)
);
```

```
CREATE TABLE Orders (
    Order_ID int NOT NULL PRIMARY KEY,
    Order_Num int NOT NULL,
    Person_ID int,
    FOREIGN KEY (Person_ID) REFERENCES P
ersons(Person_ID)
);
```

# Search and practice

- AUTO_INCREMENT Constraint
- DEFAULT Constraint
- CHECK Constraint
- ENUM Constraint
- INDEX Constraint

- Create a table named “Countries” and use following information.

| Column Name | Constraint |
|---|---|
| country<br>-ID | Primary Key, Not Null, Auto increment |
| country_Name | Give default country name as “Sri Lanka” |
| climate | Give enum values as “winter”,”summer”,”spring”, ”autumn” |

Show the table with below data.

Result Grid   Filter Rows:

| | country_Id | country_Name | cilmate |
|---|---|---|---|
| ▶ | 1 | America | Winter |
| | 3 | Sri Lanka | Autumn |
| | 4 | Sri Lanka | Autumn |
| | 5 | Sri Lanka | Summer |
| | 6 | Australia | Spring |
| * | NULL | NULL | NULL |

# SQL WHERE Clause

**SELECT *what_to_select***
**FROM *which_table***
**WHERE *conditions_to_satisfy*;**

- Operators (Use inside WHERE)
  - Comparison Operators
    - < , <= , = , != , <> , >= , >
  - Logical Operators
    - AND , OR , NOT
  - Comparison Operator for NULL value
    - IS

# SQL WHERE Clause

**SELECT *column/s***
**FROM *table***
**WHERE ‘*column_name*’ *operator* ‘*value*’;**

- Example 01

```
SELECT *
From studenet_data
WHERE reg_no = 'ICT001';
```

# SQL WHERE Clause

- Exercise
  - Display all the fields from student_data table for the student who is having Birthday on '1997-06-15'
  - Display registration no, date of birth and gender of "B.A.Jayaranga"
  - Display final marks and the grade for the student who is having registration no "ICT001"

# SQL WHERE Clause

- Example 02

  SELECT *

  From studenet_data

  WHERE dob < '1998-01-01';

- Exercise

  Insert below values to relevant columns in above created tables

  Reg_no : ICT006

  Name : H.X.Y. Saman

  Birthday : 1996/05/25

  ICT marks : 75

  Grade: A

# SQL NULL

- Try
  - Select all the records from student_data table and ict_marks table
  - Identify if there's 'NULL' values for the fields, If so identify the student/s and list down all details about them using below 'SELECT' statement
  - SELECT *

    From table

    WHERE column_name = NULL;

# More on NULL

- Did it worked ? What went wrong ?
- Try with below code segments

  - SELECT *
    From table
    WHERE column_name = ‘NULL’;

  - SELECT *
    From table
    WHERE column_name IS NULL;

# More on NULL

- Try “NOT NULL”
- SELECT 1 IS NULL, 1 IS NOT NULL;
- SELECT 1 = NULL, 1 <> NULL, 1 < NULL, 1 > NULL;
- SELECT 0 IS NULL, 0 IS NOT NULL, '' IS NULL, '' IS NOT NULL;

# SQL AND Operator

- The WHERE clause can be combined with AND operator
- The AND operator is used to filter records based on more than one condition
- The AND operator displays a record if all the conditions separated by AND is TRUE

# SQL AND Operator

**SELECT *column1*, *column2*, ...**
**FROM *table_name***
**WHERE *condition1* AND *condition2* ...;**

- Exercise
  - Display all the details of students who are having registration numbers "ICT001" and " "ICT003"
  - Any output ???

# SQL OR Operator

- The WHERE clause can be combined with OR operator
- The OR operator is used to filter records based on more than one condition
- The OR operator displays a record if any of the conditions separated by OR is TRUE

# SQL OR Operator

**SELECT *column1, column2, ...***
**FROM *table_name***
**WHERE *condition1* OR *condition2 ...;***

- Exercise
  - Display all the details of students who are "Male" or Born after "1997-01-01"

# SQL NOT Operator

- The WHERE clause can be combined with NOT operator
- The NOT operator displays a record if the condition(s) is NOT TRUE

# SQL NOT Operator

**SELECT** ***column1, column2, ...***
**FROM** ***table_name***
**WHERE NOT** ***condition;***

- Exercise
  - Display all the details of students who are not "Males"

# Combining AND, OR and NOT

- You can also combine the AND, OR and NOT operators
- Exercise
  - Display all the details of students who were born after "1997-01-01" and who are "Male" or " "Female"
  - Display all the details of students who were born after "1997-01-01" and not "Male"

# SQL LIKE Operator

- The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.
- There are two wildcards used in conjunction with the LIKE operator:
  - % - The percent sign represents zero, one, or multiple characters
  - _ - The underscore represents a single character
- The percent sign and the underscore can also be used in combinations!

# SQL LIKE Operator

**SELECT *column1, column2, ...***
**FROM *table_name***
**WHERE *columnN* LIKE *pattern*;**

- Examples :
- WHERE stu_name LIKE 'a%‘
  - Finds any values that starts with "a“
- WHERE stu_name LIKE '%a‘
  - Finds any values that ends with "a“
- WHERE stu_name LIKE '%or%‘
  - Finds any values that have "or" in any position
- WHERE stu_name LIKE '_r%‘
  - Finds any values that have "r" in the second position
- WHERE stu_name LIKE 'a_%_%‘
  - Finds any values that starts with "a" and are at least 3 characters in length
- WHERE stu_name LIKE 'a%o‘
  - Finds any values that starts with "a" and ends with "o"

# SQL IN Operator

- The IN operator allows you to specify multiple values in a WHERE clause.
- The IN operator is a shorthand for multiple OR conditions.

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1, value2, ...);
```

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (SELECT STATEMENT);
```

# SQL BETWEEN Operator

- The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.
- The BETWEEN operator is inclusive: begin and end values are included.

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

# SQL ORDER BY Keyword

- The ORDER BY keyword is used to sort the result-set in ascending or descending order
- The ORDER BY keyword sorts the records in ascending order by default.
- To sort the records in descending order, use the DESC keyword

# SQL ORDER BY Keyword

**SELECT *column1, column2, ...***
**FROM *table_name***
**ORDER BY *column1, column2, ...* ASC|DESC;**

- Exercise
  - Sort "student" table by date of birth in ascending order
  - Sort "student" table by gender in descending order
  - Sort "student" table first by date of birth ascending , then by registration number ascending and gender descending

# SQL UPDATE Statement

- The UPDATE statement is used to modify the existing records in a table

**UPDATE *table_name***
**SET *column1* = *value1*, *column2* = *value2*, ...**
**WHERE *condition*;**

- Exercise
  - Change gender of "ICT005" to "Male"
  - Change gender of "ICT003" to "FeMale" and birthday to "5/23/1997"

# SQL UPDATE Statement

- Try This

```
UPDATE student_data
SET gender ='Male';
```

Find out What happens ☺

# SQL UPDATE Statement

- Restore data in the student_data table to follow

| RegNo | StuName | DoB | Gender |
|---|---|---|---|
| ICT001 | K.M.P.Kumara | 12/25/1996 | Male |
| ICT002 | G.L.Y.Lenagala | 8/10/1996 | Female |
| ICT003 | B.A.Jayaranga | 5/23/1997 | Male |
| ICT004 | B.L.D.Lakmal | 6/15/1997 | Male |
| ICT005 | K.N.R.Nipuni | 2/18/1997 | Female |

# SQL DELETE Statement

- You are already familiar with this statement
- The DELETE statement is used to delete existing records in a table
- Try DELETE with WHERE clause
  - Hint :
    - DELETE FROM *table_name*
      WHERE *condition*;

# SQL SELECT DISTINCT Statement

- The SELECT DISTINCT statement is used to return only distinct (different) values.
- Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.

# SQL SELECT DISTINCT Statement

**SELECT DISTINCT *column1, column2, ...***
**FROM *table_name*;**

- Exercise
  - Select distinct values from gender column

- TRY
  - Retrieve How many distinct values are there in gender column
    - Hint : use COUNT AND DISTINCT TOGETHER

# SQL LIMIT Clause

- The LIMIT clause is used to specify the number of records to return.
- The LIMIT clause is useful on large tables with thousands of records.
- Returning a large number of records can impact on performance.

# SQL LIMIT Clause

```
SELECT column_name(s)
FROM table_name
WHERE condition
LIMIT number;
```

- Exercise
  - Select top 03 records from student_data table
  - Select top 02 records for Male students from student_data table

# Exercise

- Craete a “library” database in sql
- Create tables and insert given data in the “mysql_library_aid” in the resources folder
  - Consider about “Primary” and “ForeignKey” constraints

# Questions ???

# Thank You