

**CSE 311L(Database Management System)**

**LAB-Week 01(Part A)**

**Objectives:

After completing this lesson, you should be able to do the following: Create database tables

Describe the data types that can be used when specifying column definition

Table naming rules & Fields Datatypes

# *Table Naming Rules

**Table names and column names: Must begin with a letter Must be 1–30 characters long

***Must contain only A–Z, a–z, 0–9, \_, $, and # Must not duplicate the name of another object owned by the same user

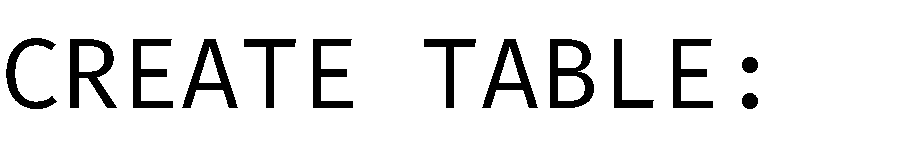
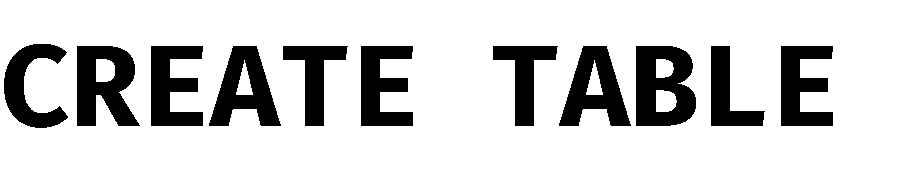
*Must not be an Oracle server reserved word

# Data Types

|  |  |
| --- | --- |
| **Data Type** | **Description** |
| CHAR(size) | A FIXED length string (can contain letters, numbers, and special characters). The size parameter specifies the column  length in characters - can be from 0 to 255. Default is 1 |
| VARCHAR(size) | A VARIABLE length string (can contain letters, numbers, and special characters). The size parameter specifies the maximum  column length in characters - can be from 0 to 65535 |
| BINARY(size) | Equal to CHAR(), but stores binary byte strings. The size  parameter specifies the column length in bytes. Default is 1 |
| TEXT(size) | Holds a string with a maximum length of 65,535 bytes. |
| BLOB(size) | For BLOBs (Binary Large Objects). Holds up to 65,535 bytes  of data |
| BIT(size) | A bit-value type. The number of bits per value is specified in size. The size parameter can hold a value from 1 to 64. The  default value for size is 1. |

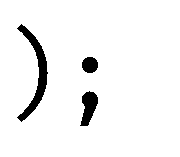
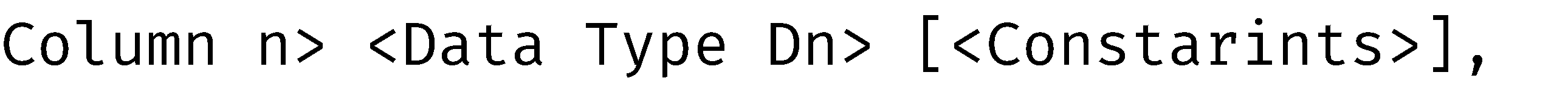
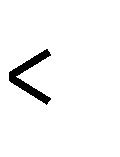
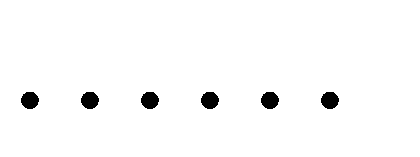
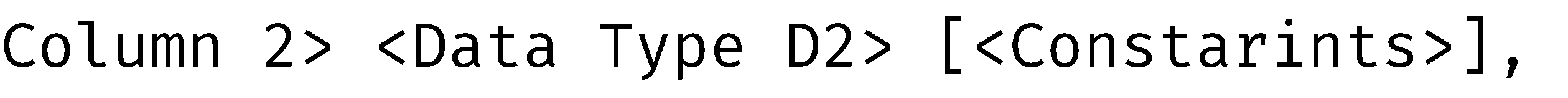
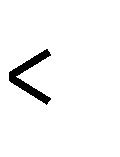
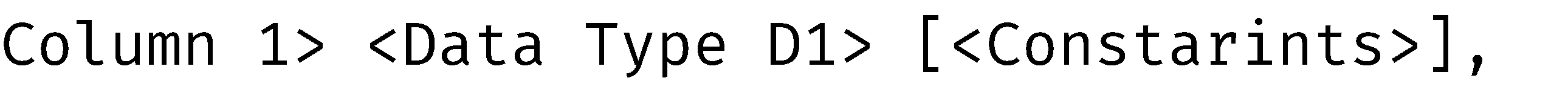
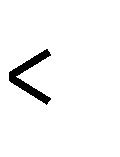
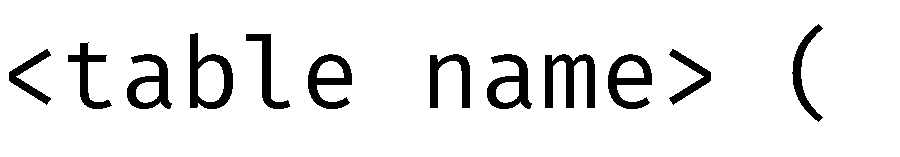
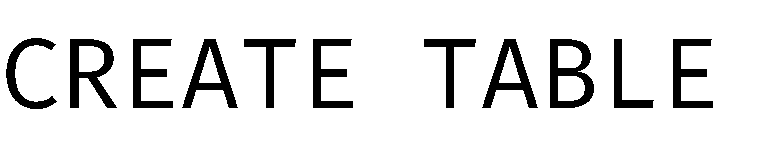
|  |  |
| --- | --- |
| TINYINT(size) | A very small integer. Signed range is from -128 to 127. Unsigned range is from 0 to 255. The size parameter specifies  the maximum display width (which is 255) |
| BOOL | Zero is considered as false, nonzero values are considered as  true. |
| BOOLEAN | Equal to BOOL |
| INT(size) | A medium integer. Signed range is from -2147483648 to 2147483647. Unsigned range is from 0 to 4294967295. The size parameter specifies the maximum display width (which is  255) |
| FLOAT(size, d) | A floating point number. The total number of digits is specified in size. The number of digits after the decimal point is specified  in the d parameter. |
| DOUBLE(size, d) | A normal-size floating point number. The total number of digits is specified in size. The number of digits after the  decimal point is specified in the d parameter |
| DECIMAL(size, d) | An exact fixed-point number. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter. The maximum number for size is 65. The maximum number for d is 30. The default value for  size is 10. The default value for d is 0. |
| DATE | A date. Format: YYYY-MM-DD. The supported range is from  '1000-01-01' to '9999-12-31' |
| DATETIME(fsp) | A date and time combination. Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'. Adding DEFAULT and ON  UPDATE in the column definition to get automatic  initialization and updating to the current date and time |
| TIMESTAMP(fsp) | A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07'  UTC. Automatic initialization and updating to the current date and time can be specified using DEFAULT CURRENT\_TIMESTAMP and ON UPDATE  CURRENT\_TIMESTAMP in the column definition. |

**Statement**



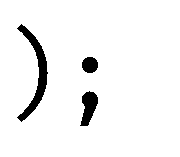
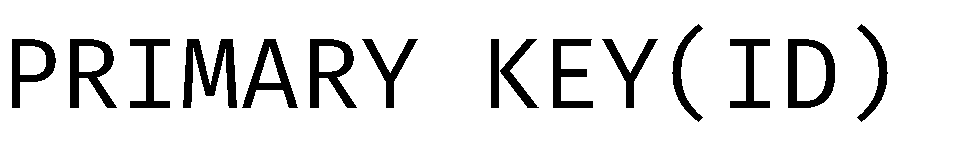
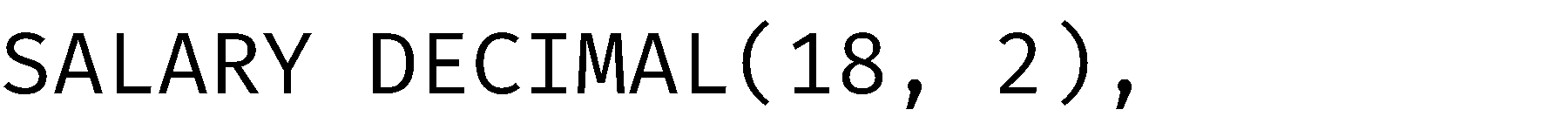
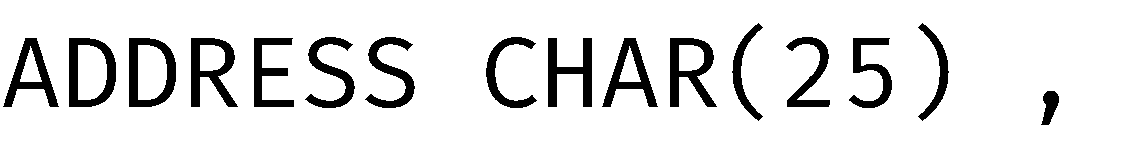
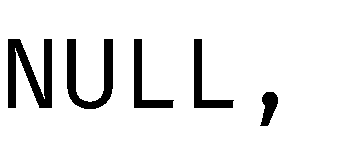
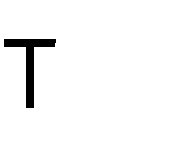
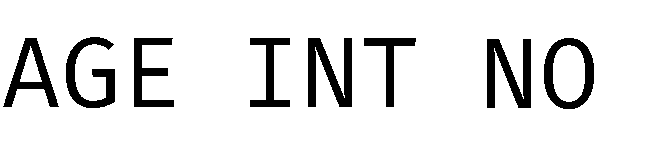
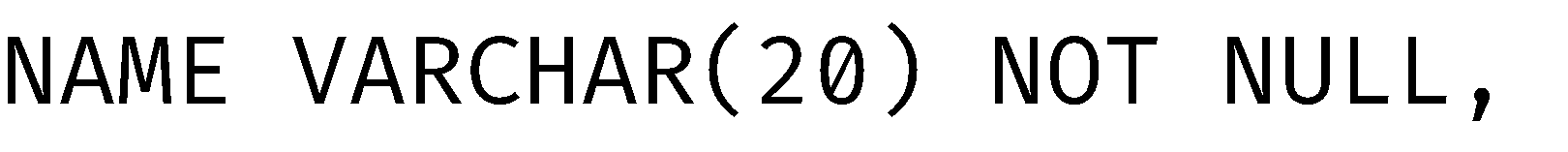
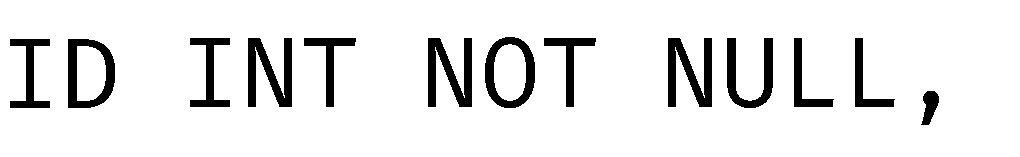
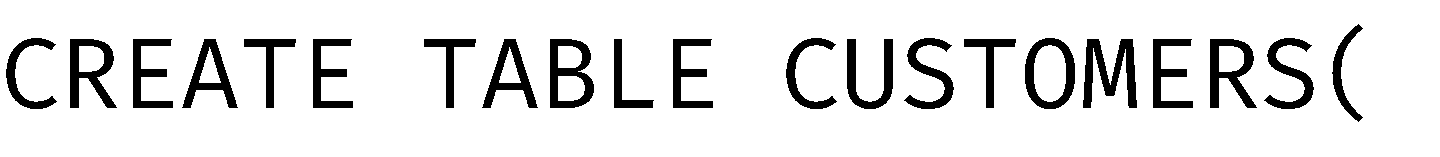
Specifies a new base relation by giving it a name, and specifying each of its attributes and their data types.

## Syntax:



* A constraint NOT NULL may be specified on an attribute.

## Example:



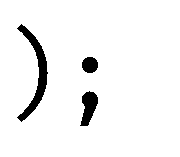
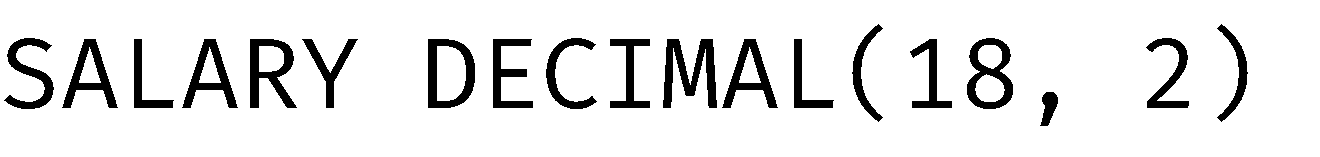
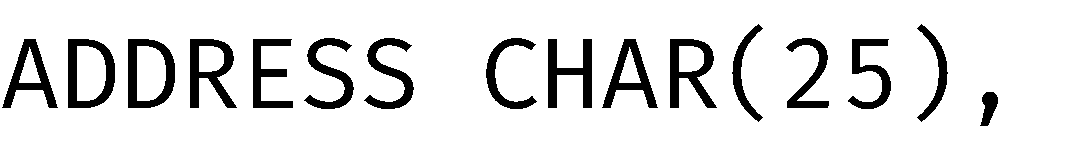
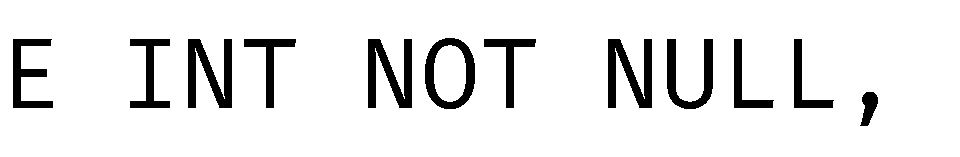
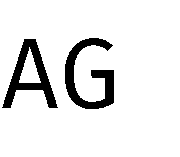
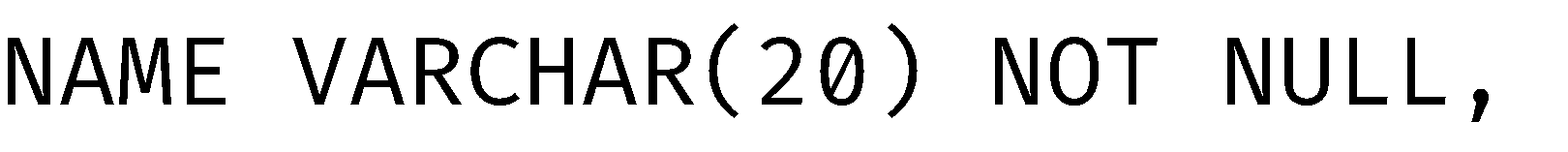
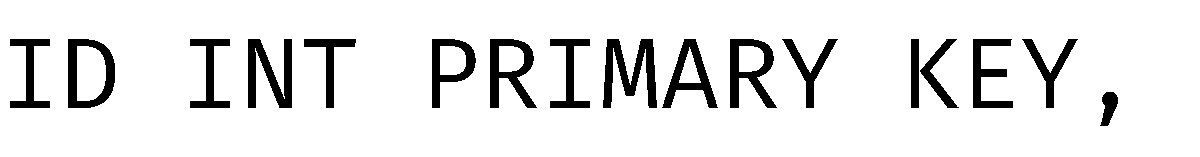
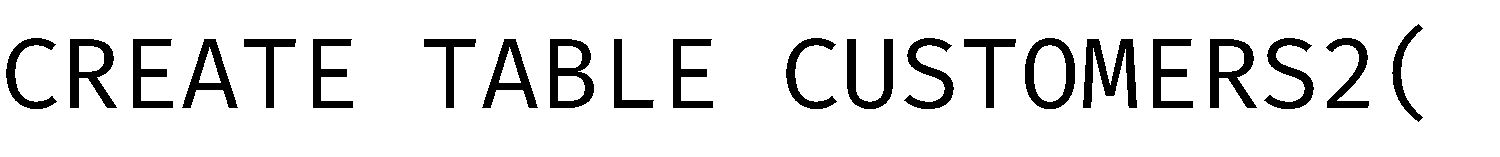
**SQL Constraints**

Constraints are the rules enforced on data columns on a table. These are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the database. Constraints can either be column level or table level. Column level constraints are applied only to one column whereas, table level constraints are applied to the entire table.

Following are some of the most commonly used constraints available in SQL.

* + NOT NULL Constraint: Ensures that a column cannot have NULL value.
  + DEFAULT Constraint: Provides a default value for a column when none is specified.
  + UNIQUE Constraint: Ensures that all values in a column are different.
  + PRIMARY Key: Uniquely identifies each row/record in a database table.
  + FOREIGN Key: Uniquely identifies a row/record in any of the given database table.
  + CHECK Constraint: The CHECK constraint ensures that all the values in a column satisfies certain conditions.
  + INDEX: Used to create and retrieve data from the database very quickly.

## Example:



**Activity 01**

Write SQL statement for create the 'Employee' table:

→

**CREATE TABLE Employee**

**(**

**Employee\_ID INT(5) NOT NULL,**

**Employee\_Name VARCHAR(30) NOT NULL,**

**Email VARCHAR(25),**

**Department\_ID INT(4) NOT null,**

**Hire\_Date DATE NOT NULL,**

**Manager\_ID INT(6),**

**Location\_ID DECIMAL(4,2)**

**)**

|  |  |  |
| --- | --- | --- |
| Name | Null? | Type |
| Employee\_ID | Not Null | INT(5) |
| Employee\_Name | Not Null | VARCHAR(30) |
| Email |  | VARCHAR(25) |
| Department\_ID | Not Null | INT(4) |
| Hire\_Date | Not Null | DATE |
| Manager\_ID |  | INT(6) |
| Location\_ID |  | DECIMAL(4,2) |



**CSE 311L(Database Management System)**

**LAB-Week 01(Part B)**

**Objectives:

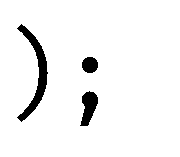
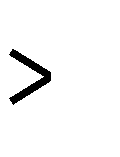
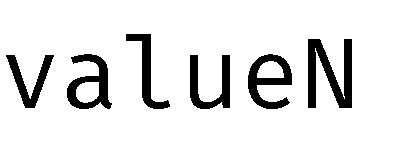
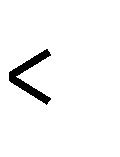
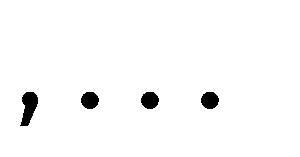
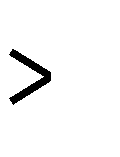
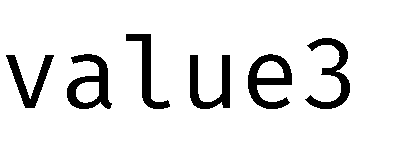
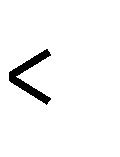
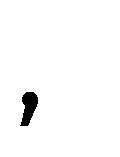
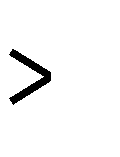
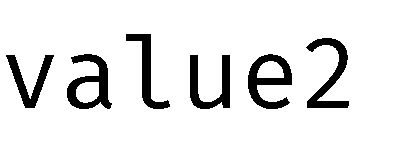
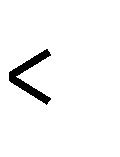
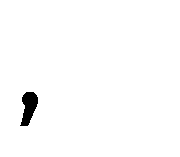
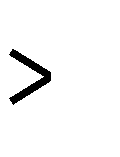
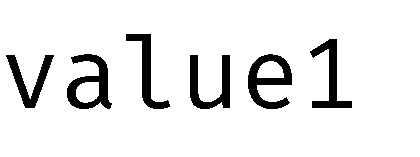
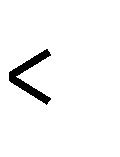
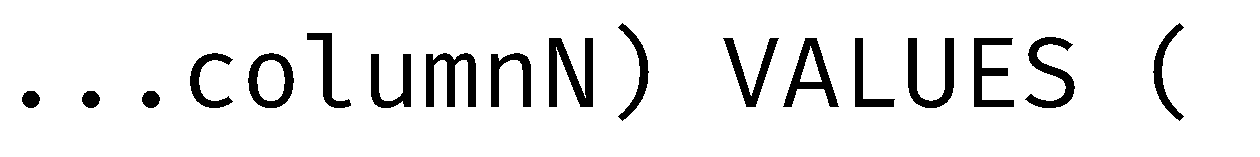
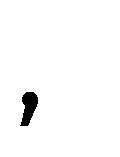
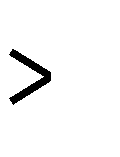
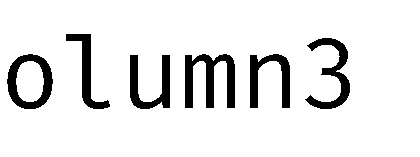
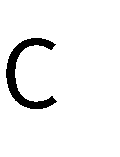
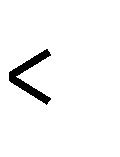
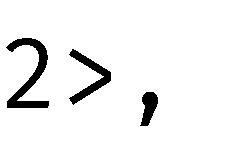
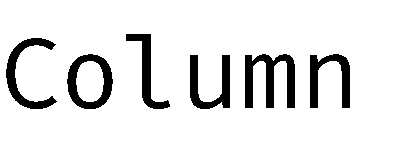
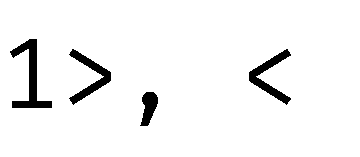
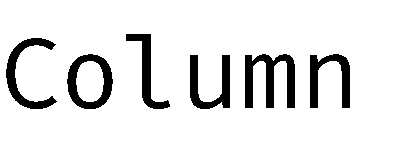
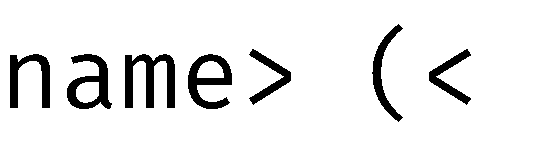
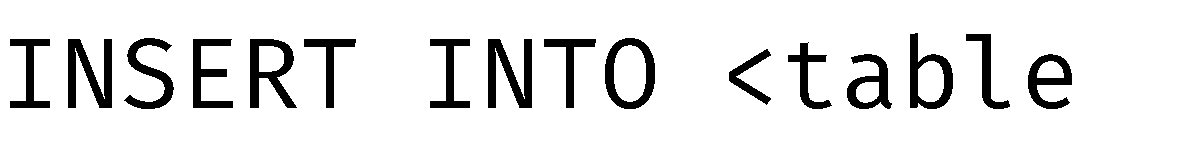
After completing this lesson, you should be able to do the following: Insert rows into the created table

Create a Department Table Execute a basic SELECT statement

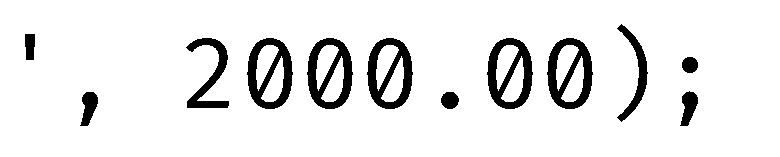
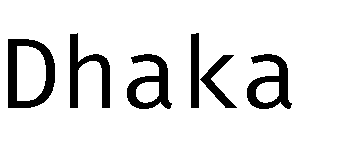
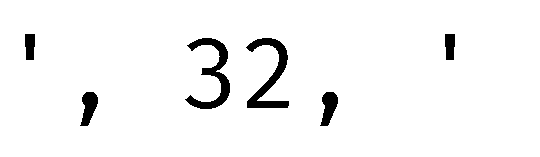
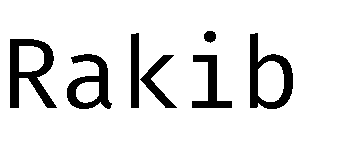
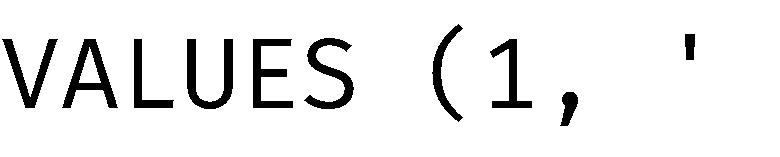
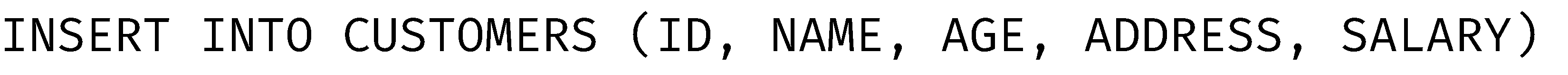
# *INSERT INTO Statement

The SQL INSERT INTO Statement is used to add new rows of data to a table in the database.

## Syntax:



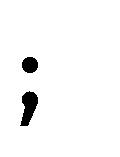
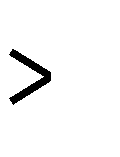
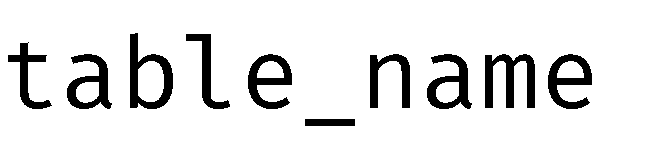
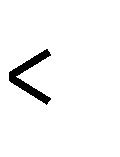
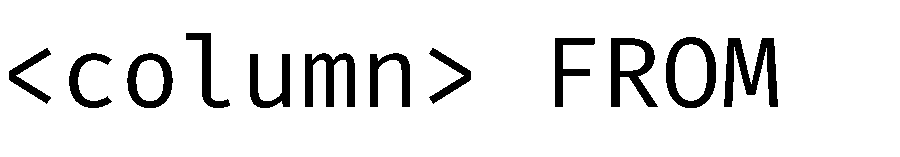
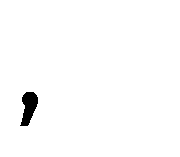
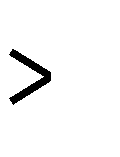
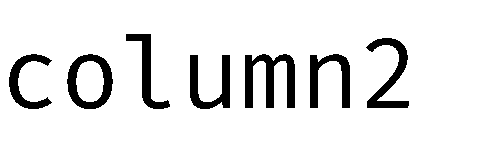
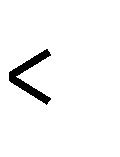
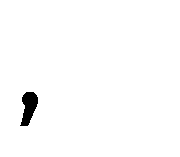
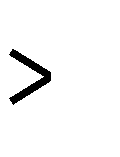
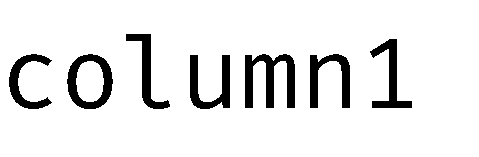
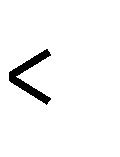
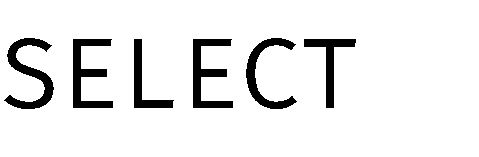
**Example:**



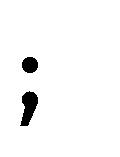
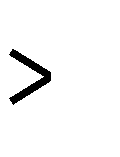
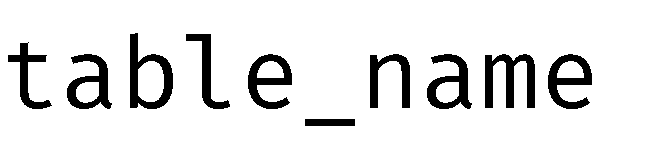
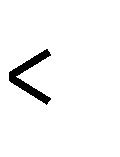
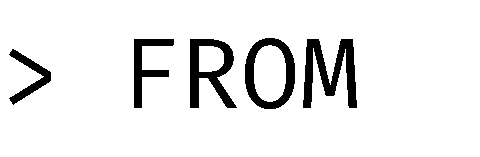
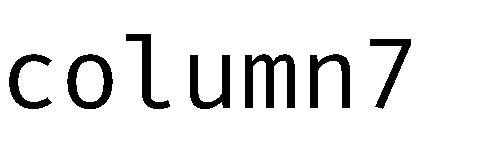
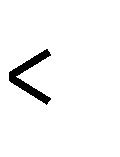
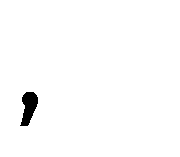
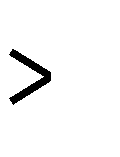
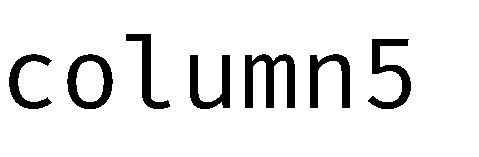
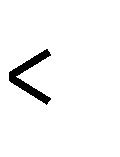
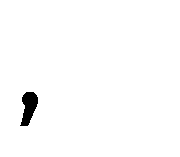
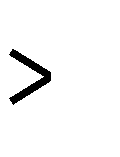
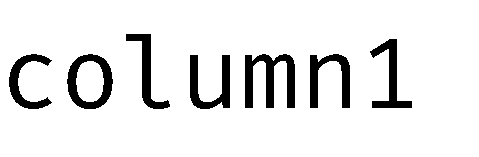
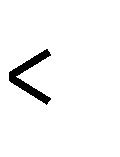
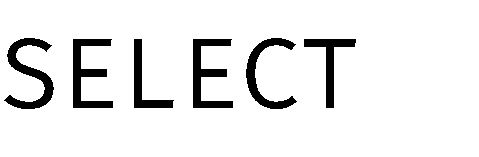
**SELECT statement**

The SQL SELECT statement is used to fetch the data from a database table which returns this data in the form of a result table.

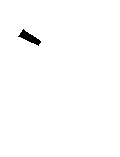
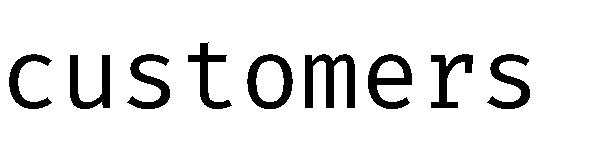
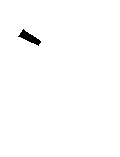
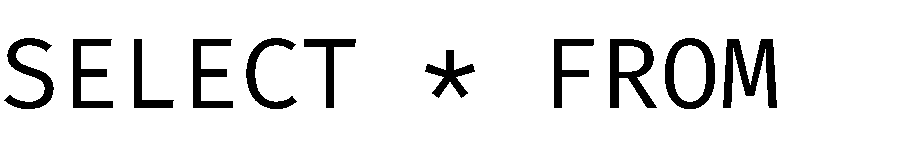
## Syntax:



**- Selecting specific columns**



**Example:**



**Activity 02**

Write SQL statement for create the 'Departments' table:

|  |  |  |
| --- | --- | --- |
| Name | Null? | Type |
| Department\_ID | Not Null | INT(4) |
| Department\_Name | Not Null | VARCHAR(30) |
| Manager\_ID |  | INT(6) |
| Location\_ID |  | DECIMAL(4,2) |

→

**CREATE TABLE Departments**

**(**

**Department\_ID INT(4) NOT NULL,**

**Department\_Name VARCHAR(30) NOT NULL,**

**Manager\_ID INT(6),**

**Location\_ID DECIMAL(4,2)**

**);**

## Activity 03

Write SQL query to insert five entries into Employee and Department table.

→

**INSERT INTO Employee (Employee.Employee\_ID, Employee.Employee\_Name, Employee.Email, Employee.Department\_ID, Employee.Hire\_Date, Employee.Manager\_ID, Employee.Location\_ID)**

**VALUES**

**(2, 'Muntashir', 'muntahir@gmail.com', 20, '1990-09-01', 200, 80.02),**

**(3, 'Munna', 'munna@gmail.com', 30, '1990-10-01', 300, 60.02),**

**(4, 'Mamun', 'mamun@gmail.com', 40, '1990-09-01', 400, 60.02),**

**(5, 'prio', 'prio@gmail.com', 50, '1990-09-01', 500, 60.02);**

## Activity 04

Write and SQL query to select all the columns from Department Table.

→ **SELECT \* FROM Employee/ Department**

## Activity 05

Write an SQL query to show Employee\_Name, Email and Location\_ID from Employee table

→ **SELECT Employee.Employee\_Name, Employee.Email, Employee.Location\_ID FROM Employee**