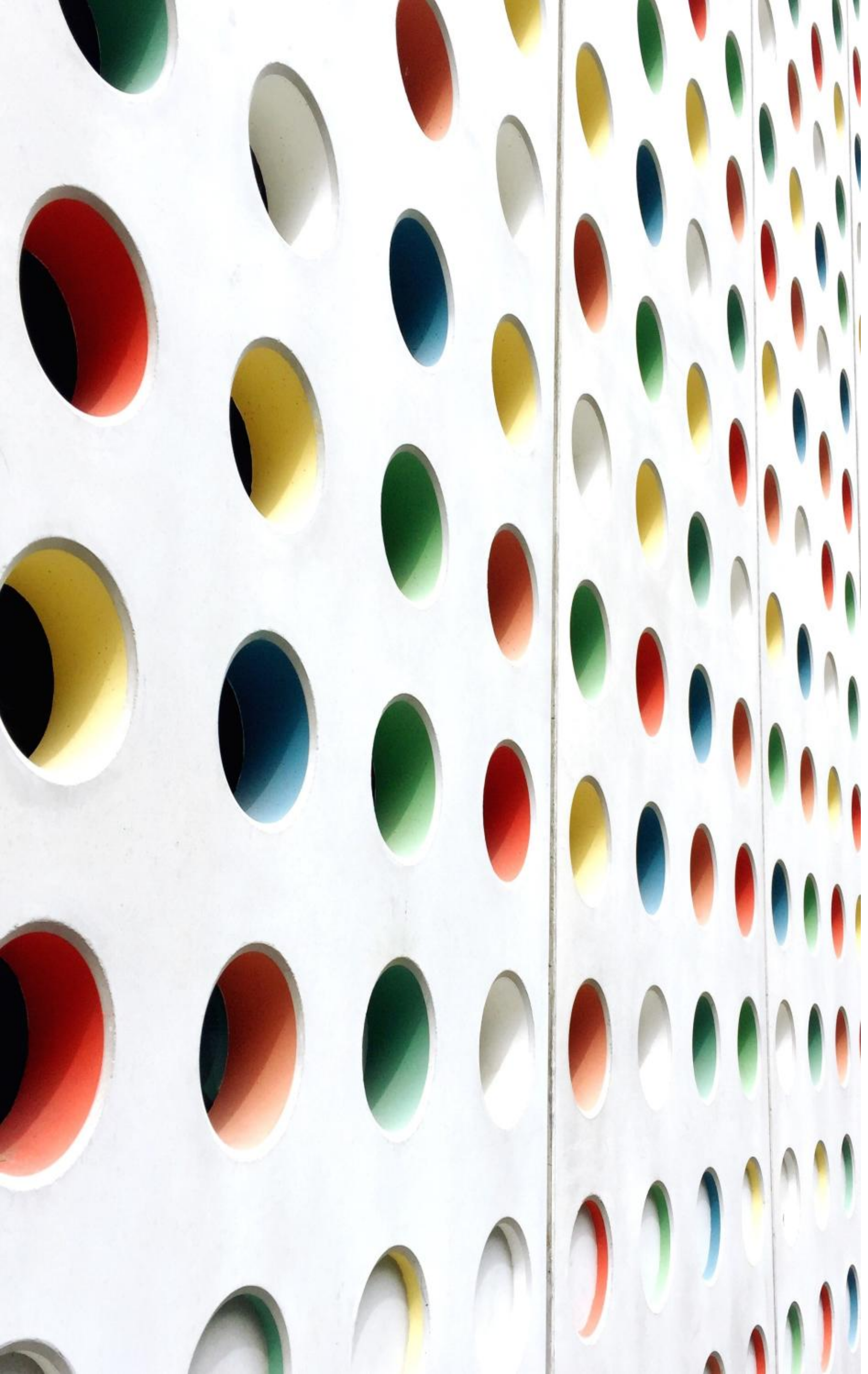


CIS 11051- PRACTICAL FOR DATABASE DESIGN

TABLE MODIFICATION

PART -1

ALTER TABLE

- The ALTER TABLE statement is used to add, delete or modify columns in an existing table.
- The ALTER TABLE statement can also used to ,
  - Add or drop a column.
  - Change a column definition.
  - Adding or dropping table constraints.

ADD COLUMN

- First, specify the name of the table which you want to add the new column.
- Second, specify the name of the column, its data type, and constraint if applicable.

```
ALTER TABLE table_name
ADD column_name data type;
```

Example:

```
ALTER TABLE Suppliers
ADD Gender VARCHAR (10);
```

As the last column of the table

```
ALTER TABLE table_name
ADD new_column definition;
```

As the first column of the table

```
ALTER TABLE table_name
ADD new_column definition FIRST;
```

After a specified column

```
ALTER TABLE table_name
ADD new_column definition
AFTER column_name;
```

Changing a Column Value

CHANGE:

CHANGE allows you to rename a column and change its definition. It is more powerful than MODIFY but can be more complex to use, as it requires both the old column name and the new column name to be specified.

MODIFY:

MODIFY allows you to change a column's definition (data type, size, etc.) but not its name.

ALTER:

ALTER is the main command used for changing table structures (e.g., adding, deleting, or modifying columns). You can use it to modify a column's default value using the SET DEFAULT syntax.

CHANGE COLUMN …

- **Syntax:**

```
ALTER TABLE table_name
CHANGE old_column_name new_column_name new_column_definition;
```

Example:

```
ALTER TABLE Employees
CHANGE EmpName Employee_Name VARCHAR(100) NOT NULL;
```

- **What this does:**
  - Renames the column EmpName to Employee_Name.
  - Changes its data type to VARCHAR(100).
  - Adds a NOT NULL constraint.

CHANGE COLUMN

- It is possible to change a columns definition as well as its position inside the table.

```
ALTER TABLE table_name
CHANGE
old_column_name
new_column_name
new_column_definition
FIRST;
```

Example:

```
ALTER TABLE Suppliers
CHANGE Sup_name Employee_name VARCHAR(30) FIRST;
```

MODIFY COLUMN

Change the data type of a column in a table.

```
ALTER TABLE table_name
MODIFY column_name new_column_definition;
```

Example:

```
ALTER TABLE Employees
MODIFY Salary DECIMAL(10,2);
```

What this does:

- Changes the column’s data type to DECIMAL(10,2)

ALTER COLUMN DEFAULT VALUE

- Syntax:

```
ALTER TABLE table_name
ALTER COLUMN column_name SET DEFAULT default_value;
```

Example:

```
ALTER TABLE Orders
ALTER COLUMN Status SET DEFAULT 'Pending';
```

What this does ?

- If no value is provided for Status, it will default to 'Pending'.

Remove Column Default Value

- Syntax:

```
ALTER TABLE table_name
ALTER COLUMN column_name DROP DEFAULT;
```

Example:

```
ALTER TABLE Orders
ALTER COLUMN Status DROP DEFAULT;
```

What this does ?

- Removes the default value for Status.

RENAMEING TABLE

- To rename a table , use the **RENAME** option of the **ALTER TABLE** statement.

```
ALTER TABLE table_name
RENAME TO new_table_name;
```

Example:

```
CREATE TABLE suppliers
RENAME TO Customers;
```

DROP COLUMN

- Used to delete a **specific column from a table**, including all its data. The rest of the table remains unchanged.

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

Example:

```
ALTER TABLE Suppliers
DROP COLUMN email;
```

DROP TABLE

- Used to delete the table

DROP TABLE table_name;

Example:

DROP TABLE student;

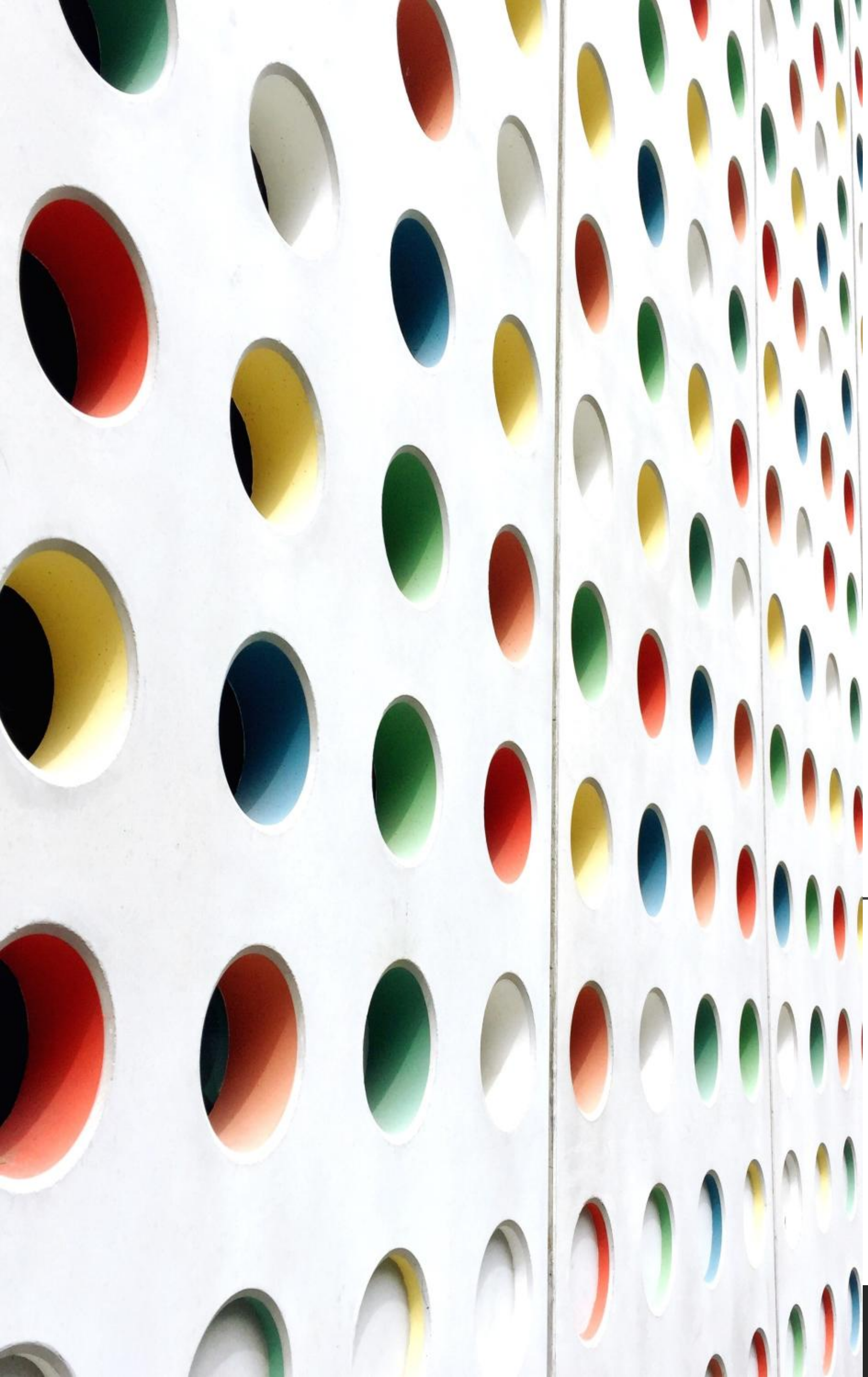
DROP DATABASE

- **Used to delete the database**

```
DROP DATABASE database_name;
```

Example:

```
DROP DATABASE store_db;
```



CIS 11051- PRACTICAL FOR DATABASE DESIGN

TABLE MODIFICATION

PART -2

UPDATING COLUMN VALUES …

- **The UPDATE statement is used to modify existing records in a table.**
- **You can update one or multiple columns at the same time.**
- **If you omit the WHERE clause, all rows will be updated, so use it carefully.**

- **Syntax:**

```
UPDATE table_name
SET
    column_name1 = expr1,
    column_name2 = expr2,
    column_name3 = expr3
WHERE condition;
```

UPDATING A SINGLE COLUMN

Suppose we have a Customers table, and we want to update the City of a customer with CustomerID = 10;

Example:

```
UPDATE Customers
SET City = 'Los Angeles’
WHERE CustomerID = 10;
```

What this does:

- Finds the row where CustomerID is 10.
- Changes the City value to 'Los Angeles'.

UPDATING MULTIPLE COLUMNS

Let’s update a customer’s City and PhoneNumber at the same time:

Example:

```
UPDATE Customers
SET
City = 'New York',
PhoneNumber = '123-456-7890’
WHERE CustomerID = 15;
```

What this does:

- **Updates both City and PhoneNumber for the customer with CustomerID = 15.**

Warning: Updating All Rows!

If you omit the WHERE clause, every row in the table will be updated:

Example:

```
UPDATE Customers
SET City = 'Chicago';
```

What this does:

- **This changes the City value to 'Chicago' for ALL customers!**

Removing Duplicates …

- To remove duplicate records from a SELECT statement and ensure unique results, it is possible to use the **DISTINCT** keyword.

```
SELECT DISTINCT field1, field2,….
FROM table_name;
```

Example Table: employees

id	name	department
1	Alice	HR
2	Bob	IT
3	Alice	HR

```
SELECT DISTINCT name, department
FROM employees;
```

Result Table:

id	name	department
1	Alice	HR
2	Bob	IT

DELETE RECORD

- The DELETE statement removes specific records (rows) from a table based on a given condition..

```
DELETE FROM table_name
WHERE condition;
```

Example:

```
DELETE FROM Customers
WHERE CustomerID = 5;
```

What this does:

- Deletes the row where CustomerID is 5.
- Other records in the table remain unchanged.

THANK YOU !!