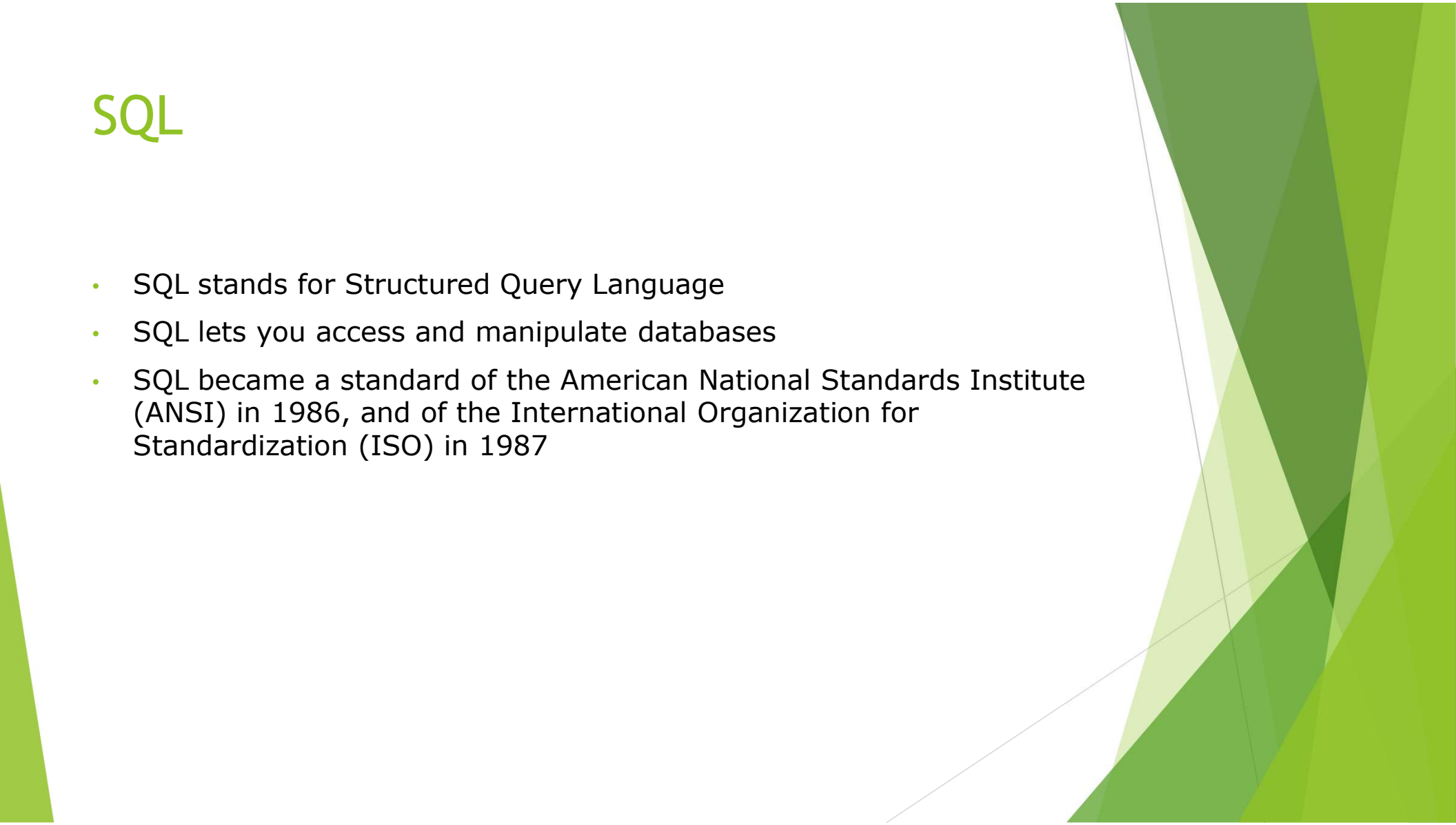
MySQL

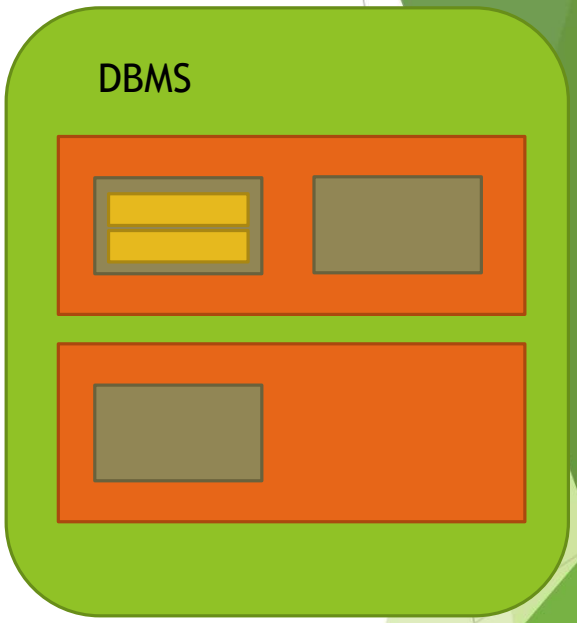
SQL

- SQL stands for Structured Query Language
- SQL lets you access and manipulate databases
- SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987



What can it do?

- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases
- SQL can create new tables in a database
- SQL can create stored procedures in a database
- SQL can create views in a database
- SQL can set permissions on tables, procedures, and views



Data Types

- Numeric
  - https://dev.mysql.com/doc/refman/8.0/en/numeric-functions.html
- String
  - https://dev.mysql.com/doc/refman/8.0/en/string-functions.html
- Date Time
  - https://dev.mysql.com/doc/refman/8.0/en/date-and-time-functions.html
- Spatial
  - Spatial data, also known as geospatial data, is a term used to describe any data related to or containing information about a specific location on the Earth's surface.
- Boolean

User Defined Variables

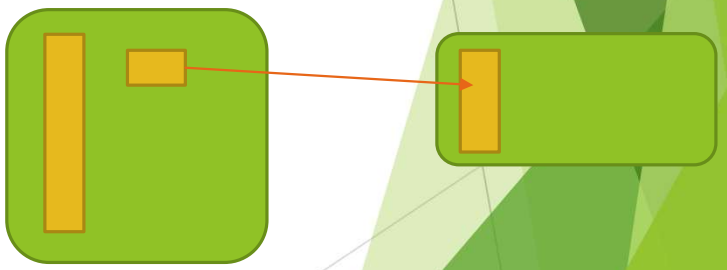
- You can store a value in a user-defined variable in one statement and refer to it later in another statement. This enables you to pass values from one statement to another.
  - SET *@var_name* = *expr* [, *@var_name* = *expr*]
  - Examples:
    - mysql> SET @v1 = X'41';
    - mysql> SET @v2 = X'41'+0;
    - mysql> SET @v3 = CAST(X'41' AS UNSIGNED);
    - mysql> SELECT @v1, @v2, @v3;

SQL Terminology

- Relational Database
  - Organized raw data in tuples
  - Tuples organized in to relations
  - It imposes structure on its contents
- DBMS
  - Software system which facilitates data organization
  - Example: MySQL, SQLite, Oracle, MS SQL Server, Mongo DB
- Schema
  - A schema is a model for describing the structure of information.
- Table
  - Table refers to data arranged in rows and columns
- View
  - a virtual table based on the result-set of an SQL statement

SQL Terminology

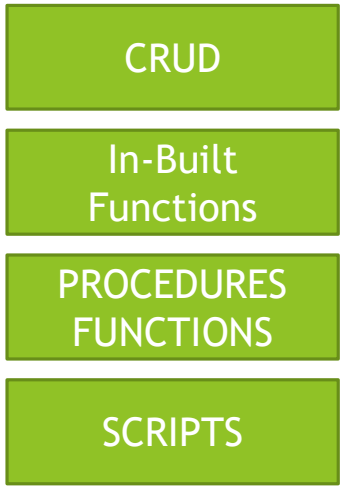
- Procedure
  - procedure is a prepared SQL code that you can save, so the code can be reused over and over again
  - Returning a value is optional
  - Can have both input and output parameters
- Function
  - A sub-programs for manipulating data. Many in-built functions available
  - Should return a value
  - Only input parameters should be present
- Primary Key
  - Attribute used to uniquely identify a row of data
  - Example: Employee ID
- Foreign Key
  - Attribute in a given database which is a primary key in another
  - Helps establish relationships



CRUD

- C -> Create
- R -> Retrieve
- U -> Update
- D -> Delete

- These are the fundamental operations that can be performed on a database



MySQL Installation and Configuration

Database: Creation, Deleting, Selecting

- Creation of a database
  - `CREATE DATABASE mydb;`
- Deleting a database
  - `DROP DATABASE mydb;`
- Creating a database does not select it for use; you must do that explicitly.
  - `USE mydb;`
- To change the overall characteristics of the database
  - `ALTER DATABASE mydb READ ONLY = 0 DEFAULT COLLATE utf8mb4_bin;`

Tables

- Creating a basic table involves naming the table and defining its columns and each column's data type.
- Example:

```
CREATE TABLE CUSTOMERS(
   ID INT  NOT NULL,
   NAME  VARCHAR (20) NOT NULL,
   AGE   INT  NOT NULL,
   ADDRESS  CHAR (25) ,
   SALARY  DECIMAL (18, 2),
   PRIMARY KEY (ID)
);
```

Inserting into tables

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

```
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00 );
```

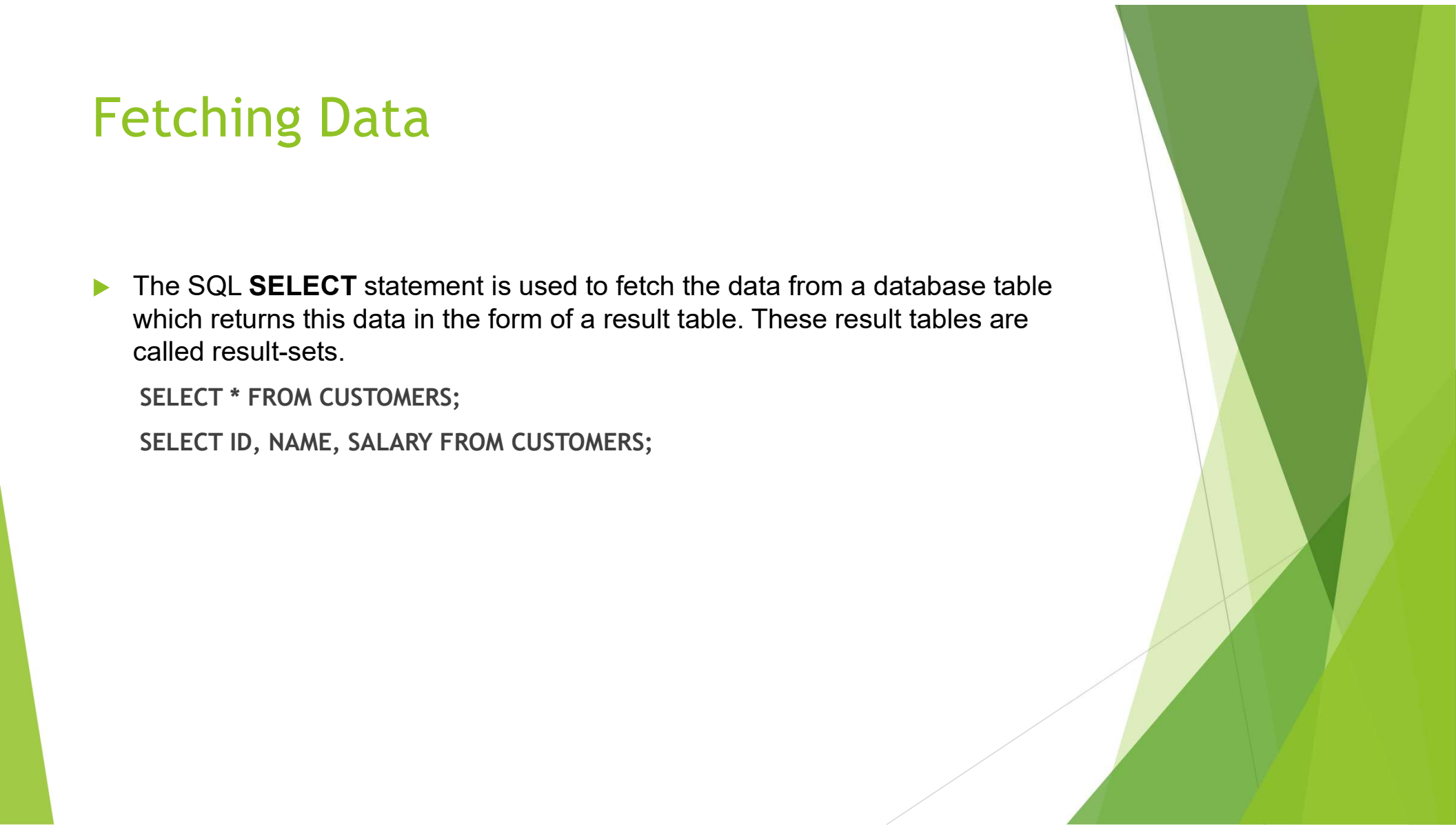
Populate the Table

```
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY) VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00);

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY) VALUES (2, 'Khilan', 25, 'Delhi', 1500.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY) VALUES (3, ‘Kaushik', 23, 'Kota', 2000.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY) VALUES (4, 'Chaitali', 25, 'Mumbai', 6500.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY) VALUES (5, 'Hardik', 27, 'Bhopal', 8500.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY) VALUES (6, 'Komal', 22, 'MP', 4500.00 );
```

Fetching Data

- 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. These result tables are called result-sets.

```
SELECT * FROM CUSTOMERS;

SELECT ID, NAME, SALARY FROM CUSTOMERS;
```



WHERE Clause

- The where clause allows you to specify conditions for fetching:

```
SELECT ID, NAME, SALARY
FROM CUSTOMERS
WHERE SALARY > 2000;
```

```
SELECT ID, NAME, SALARY
FROM CUSTOMERS
WHERE NAME = 'Hardik';
```

AND and OR Clauses

- The SQL **AND** & **OR** operators are used to combine multiple conditions to narrow data in an SQL statement. These two operators are called as the conjunctive operators.

```
SELECT ID, NAME, SALARY
FROM CUSTOMERS
WHERE SALARY > 2000 AND age < 25;
```

```
SELECT ID, NAME, SALARY
FROM CUSTOMERS
WHERE SALARY > 2000 OR age < 25;
```

UPDATE Query

- The SQL **UPDATE** Query is used to modify the existing records in a table.

```
UPDATE CUSTOMERS
SET ADDRESS = 'Pune'
WHERE ID = 6;

UPDATE CUSTOMERS
SET ADDRESS = 'Pune', SALARY = 1000.00; Updates everything
```

DELETE Query

- The SQL DELETE Query is used to delete the existing records from a table.

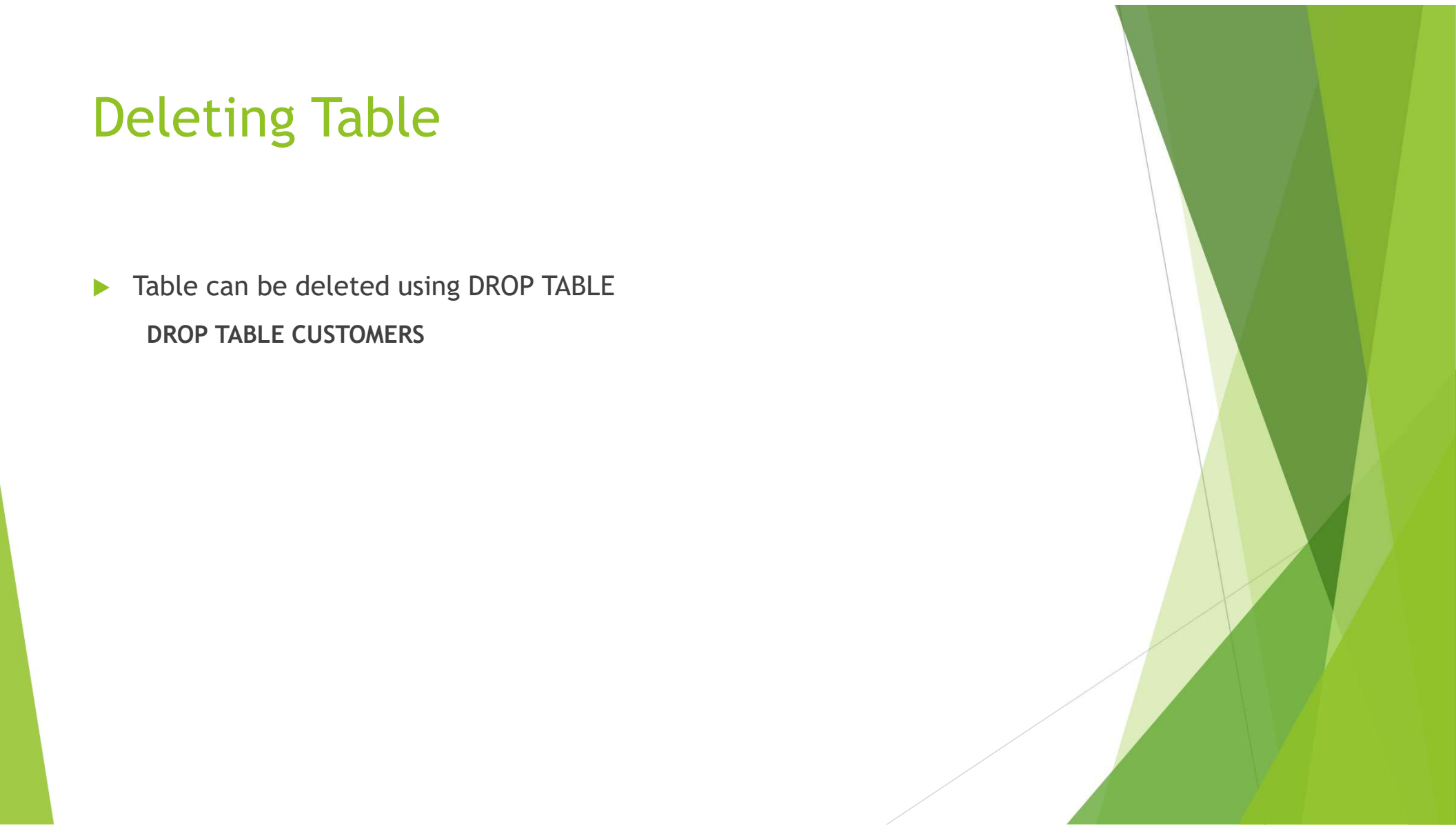
```
DELETE FROM CUSTOMERS
WHERE ID = 6;
```

```
DELETE FROM CUSTOMERS;
```

Deleting Table

- Table can be deleted using DROP TABLE

  DROP TABLE CUSTOMERS



Day #2

Prepare the Table

```
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY) VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00);

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY) VALUES (2, 'Khilan', 25, 'Delhi', 1500.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY) VALUES (3, ‘Kaushik', 23, 'Kota', 2000.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY) VALUES (4, 'Chaitali', 25, 'Mumbai', 6500.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY) VALUES (5, 'Hardik', 27, 'Bhopal', 8500.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY) VALUES (6, 'Komal', 22, 'MP', 4500.00 );
```

Transferring Data from One Table to Another

- You can populate the data into a table through the select statement over another table; provided the other table has a set of fields, which are required to populate the first table.

```
INSERT INTO Table2 (UserName,Password)
SELECT UserName,Password FROM Table1 WHERE UserName='X' AND Password='X'

DELETE FROM Table1 WHERE UserName='X' AND Password='X'
```

LIKE Clause

- The SQL **LIKE** clause is used to compare a value to similar values using wildcard operators. There are two wildcards used in conjunction with the LIKE operator.

```
SELECT * FROM CUSTOMERS
WHERE SALARY LIKE '200%';
```

No.	Statement & Description
1	**WHERE SALARY LIKE '200%'** Finds any values that start with 200.
2	**WHERE SALARY LIKE '%200%'** Finds any values that have 200 in any position.
3	**WHERE SALARY LIKE '_00%'** Finds any values that have 00 in the second and third positions.
4	**WHERE SALARY LIKE '2_%_%'** Finds any values that start with 2 and are at least 3 characters in length.
5	**WHERE SALARY LIKE '%2'** Finds any values that end with 2.
6	**WHERE SALARY LIKE '_2%3'** Finds any values that have a 2 in the second position and end with a 3.
7	**WHERE SALARY LIKE '2___3'** Finds any values in a five-digit number that start with 2 and end with 3.

Regular Expressions

- Regular expressions are templates to match patterns in given data

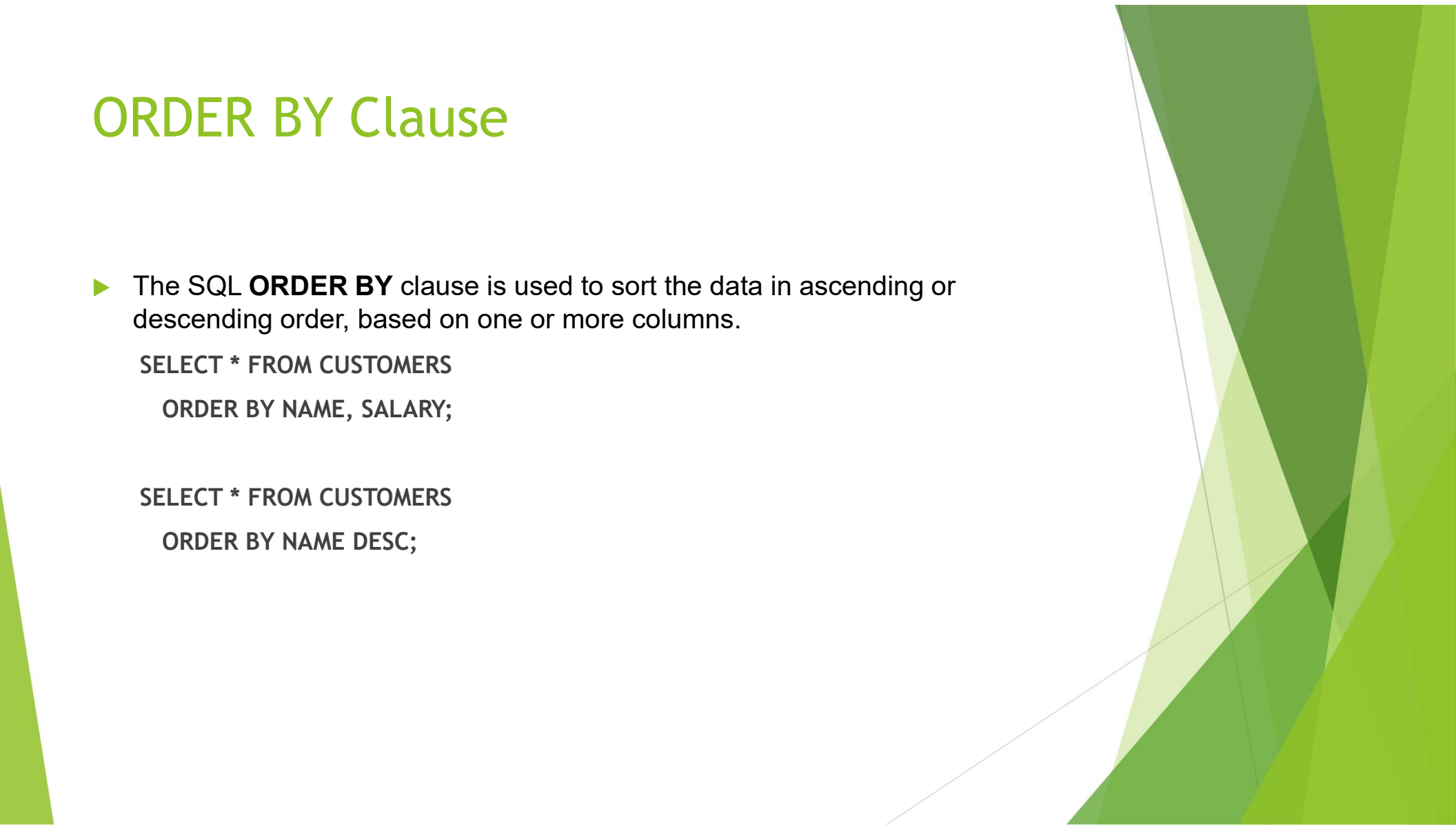
```
SELECT * FROM `movies` WHERE `title` REGEXP '^[^abcd]';
```

ORDER BY Clause

- The SQL **ORDER BY** clause is used to sort the data in ascending or descending order, based on one or more columns.

```
SELECT * FROM CUSTOMERS
   ORDER BY NAME, SALARY;
```

```
SELECT * FROM CUSTOMERS
   ORDER BY NAME DESC;
```

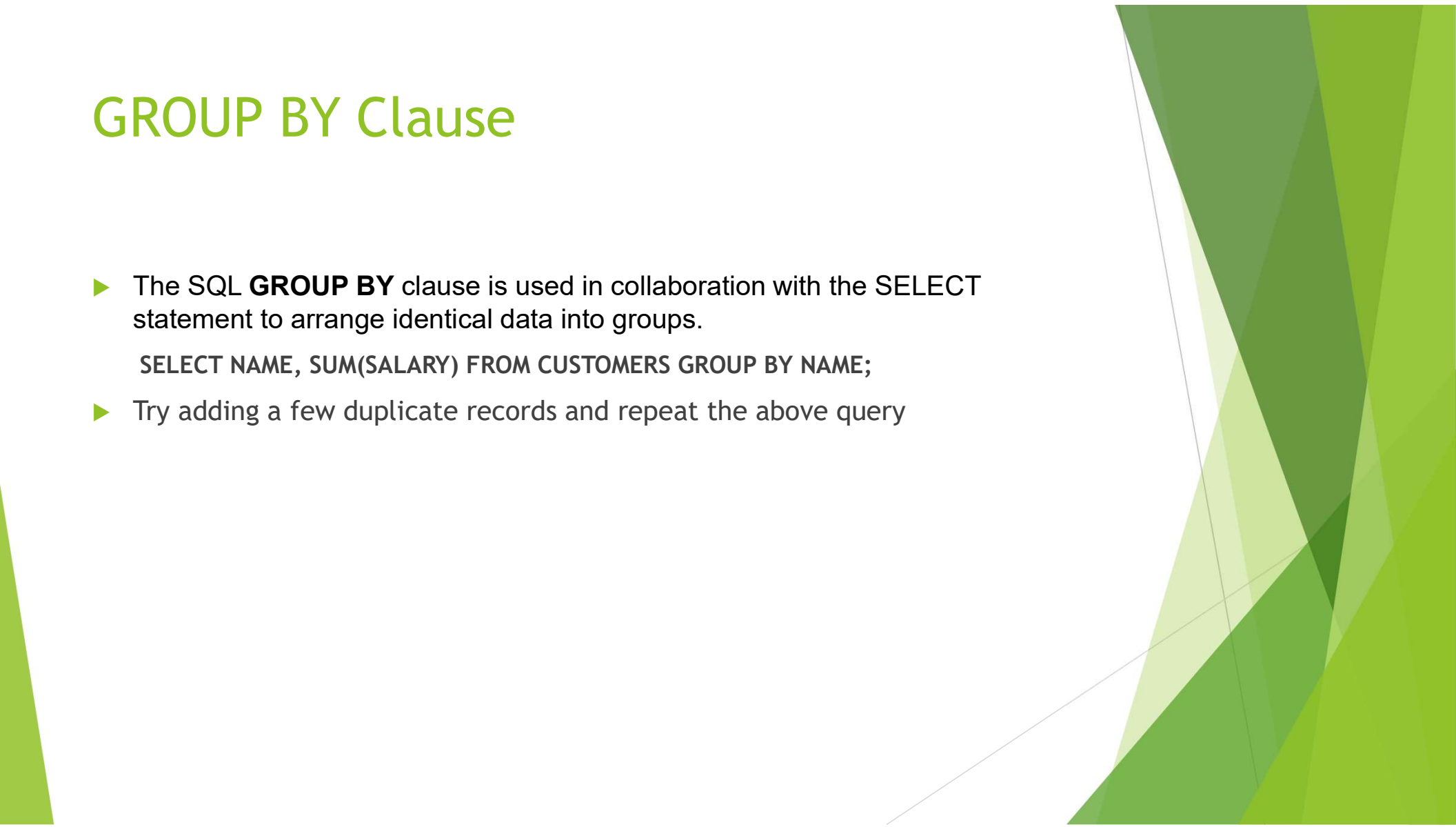


GROUP BY Clause

- The SQL **GROUP BY** clause is used in collaboration with the SELECT statement to arrange identical data into groups.

  SELECT NAME, SUM(SALARY) FROM CUSTOMERS GROUP BY NAME;

- Try adding a few duplicate records and repeat the above query



Sorting Outcomes

```
SELECT * FROM CUSTOMERS
   ORDER BY (CASE ADDRESS
   WHEN 'DELHI'       THEN 1
   WHEN 'BHOPAL'      THEN 2
   WHEN 'KOTA'        THEN 3
   WHEN 'AHMEDABAD' THEN 4
   WHEN 'MP'     THEN 5
   ELSE 100 END) ASC, ADDRESS DESC;
```

TRUNCATE TABLE

- The SQL **TRUNCATE TABLE** command is used to delete complete data from an existing table.
- **TRUNCATE TABLE CUSTOMERS;**

Eliminating Duplicates

- If you use one column after the DISTINCT operator, the database system uses that column to evaluate duplicate.

```
SELECT
    salary
FROM
    employees
ORDER BY salary DESC;


SELECT
    DISTINCT salary
FROM
    employees
ORDER BY salary DESC;
```

Eliminating Duplicates

```
SELECT
    job_id,
    salary
FROM
    employees
ORDER BY
    job_id,
    salary DESC;
```

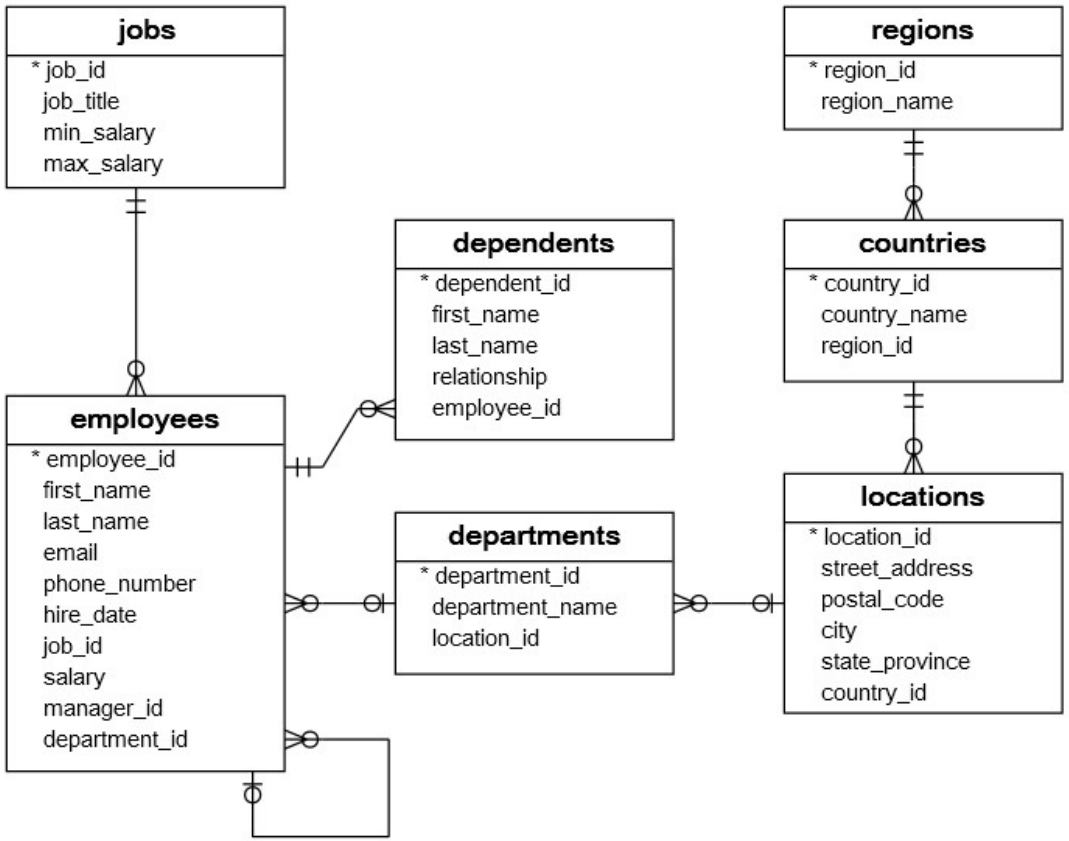
```
SELECT DISTINCT
    job_id,
    salary
FROM
    employees
ORDER BY
    job_id,
    salary DESC;
```

DROP TABLE

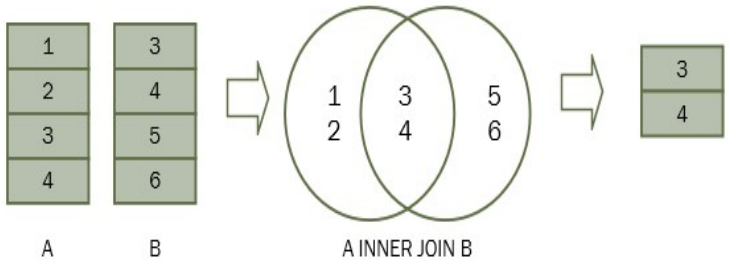
- The SQL **DROP TABLE** statement is used to remove a table definition and all the data, indexes, triggers, constraints and permission specifications for that table.

```
DROP TABLE CUSTOMERS;
```

Create new data

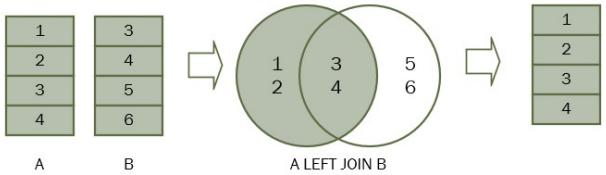


Inner Join

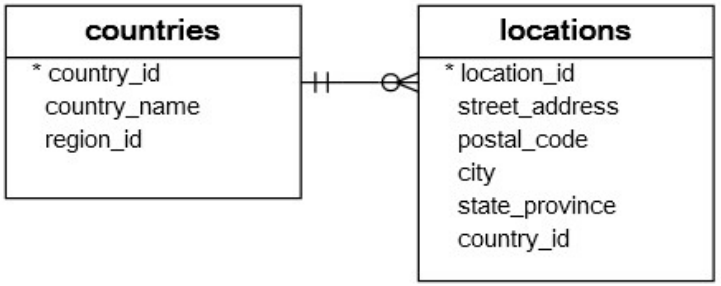



```
SELECT
    first_name,
    last_name,
    employees.department_id,
    departments.department_id,
    department_name
FROM
    employees
        INNER JOIN
    departments ON departments.department_id = employees.department_id
WHERE
    employees.department_id IN (1 , 2, 3);
```

Left Join




```
SELECT
        c.country_name,
        c.country_id,
        l.country_id,
        l.street_address,
        l.city
FROM
        countries c
LEFT JOIN locations l ON l.country_id = c.country_id
WHERE
        c.country_id IN ('US', 'UK', 'CN')
```



Aliases

- SQL alias allows you to assign a table or a column a temporary name during the execution of a query. There are two types of aliases: table alias and column alias.

```
SELECT
    employee_id,
    concat(first_name, ' ', last_name) fullname

FROM
    employees e

INNER JOIN departments d ON d.department_id = e.department_id
```

Column alias

Table alias

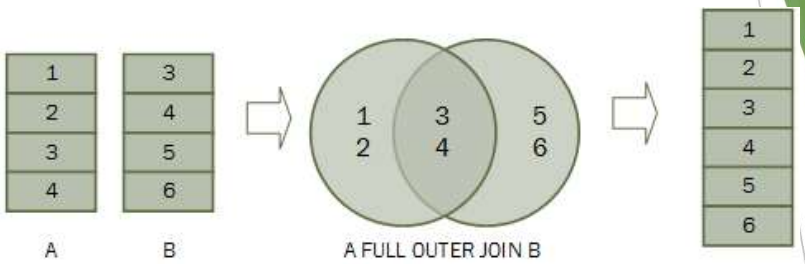
Table alias

Right Join

```
SELECT
	c.country_name,
	c.country_id,
	l.country_id,
	l.street_address,
	l.city
FROM
	countries c
RIGHT JOIN locations l ON l.country_id = c.country_id
WHERE
	c.country_id IN ('US', 'UK', 'CN')
```

Outer Join

- LEFT OUTER JOIN
- RIGHT OUTER JOIN



Grouping Records

The GROUP BY clause is an optional clause of the SELECT statement that combines rows into groups based on matching values in specified columns. One row is returned for each group.

```
SELECT
    department_id,
    COUNT(employee_id) headcount
FROM
    employees
GROUP BY
    department_id;
```

Null Values

- It is not possible to test for NULL values with comparison operators, such as =, <, or <>.
- We will have to use the IS NULL and IS NOT NULL operators instead.

```
SELECT CustomerName, ContactName, Address
FROM Customers
WHERE Address IS NULL;
```

```
SELECT CustomerName, ContactName, Address
FROM Customers
WHERE Address IS NOT NULL;
```

Alter

```
CREATE TABLE trainings (
    employee_id INT,
    course_id INT,
    taken_date DATE,
    PRIMARY KEY (employee_id , course_id)
);
```

- Once you create a new table, you may want to change its structure because business requirements change. To modify the structure of a table, you use the ALTER TABLE statement. The ALTER TABLE statement allows you to perform the following operations on an existing table:
  - Add a new column using the ADD clause.
  - Modify attribute of a column such as constraint, default value, etc. using the MODIFY clause.
  - Remove columns using the DROP clause.

Alter

```
ALTER TABLE courses ADD credit_hours INT NOT NULL;

ALTER TABLE courses
ADD fee NUMERIC (10, 2) AFTER course_name,
ADD max_limit INT AFTER course_name;

ALTER TABLE courses
MODIFY fee NUMERIC (10, 2) NOT NULL;

ALTER TABLE courses DROP COLUMN fee;
```

Sub-queries

- By definition, a subquery is a query nested inside another query such as SELECT, INSERT, UPDATE, or DELETE statement.

```
SELECT
    employee_id, first_name, last_name
FROM
    employees
WHERE
    department_id IN (SELECT
            department_id
        FROM
            departments
        WHERE
            location_id = 1700)
ORDER BY first_name , last_name;
```