PART 3: Database Operations

Environment Setup

- **MySQL Community Server Download: https://dev.mysql.com/downloads/mysql/**
- **POP SQL Download: https://popsql.com/**

Online SQL Editor

- **SQL Fiddle: http://sqlfiddle.com/**

PART 3: Database Operations

CRUD Operations

In most applications, it should allow user to

- Create or Add new entries
- Read, Retrieve, search or view existing entries
- Update or Edit existing entries
- Delete, deactivate, or remove existing entries

SQL CRUD Operations

- Create: To create table, the command is CREATE TABLE statement
- Read: To read data from a table, we can use SELECT statement
- Update: To make changes to a table, UPDATE or ALTER statement is used.
- Delete: To delete data, we can use DELETE or DROP statement.

CRUD	SQL Statement
C - Create	INSERT INTO … VALUES…
R - Read	SELECT … FROM … WHERE …
U - Update	UPDATE … SET … WHERE …
D - Delete	DELETE FROM … WHERE …

PART 3: Database Operations

Creating Table

First step to store data in database is to add a table. We use **CREATE TABLE** command in SQL

CREATE TABLE Statement

- The column parameters specify the names of the columns of the table; The datatype parameter specify the type of data the column can have, such as varchar, integer, date, etc.
- If there already exists a table with the same name, the SQL implementation will usually throw an error, so to reduce the error and skip creating a table if one exists, you can use **IF NOT EXISTS** clause.

```
CREATE TABLE IF NOT EXISTS mytable (
    column1 dataType,
    column2 dataType,
    …
);
```

- You can create a table using an existing table. The new table gets same column definitions.

```
CREATE TABLE new_table_name AS
    SELECT column1, column2,...
    FROM existing_table_name
    WHERE ....;
```

Syntax

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....
);
```

Example

```
CREATE TABLE IF NOT EXISTS `Shippers` (
  `ShipperID` INTEGER PRIMARY KEY,
  `ShipperName` VARCHAR(100) NOT NULL,
  `Phone` VARCHAR(100) NOT NULL
);
```

Result

- shippers (TABLE)
  - ShipperID INT(10)
  - ShipperName VARCHAR(100)
  - Phone VARCHAR(100)

PART 3: Database Operations

Creating Table

First step to store data in database is to add a table. We use **CREATE TABLE** command in SQL

Table Column Data Type

- Different databases support different data types, but common types include numeric, string, dates. Below table summarizes MySQL data types.

MySQL DATA TYPES

DATE TYPE	SPEC	DATA TYPE	SPEC
CHAR	String (0 - 255)	INT	Integer (-2147483648 to 214748-3647)
VARCHAR	String (0 - 255)	BIGINT	Integer (-9223372036854775808 to 9223372036854775807)
TINYTEXT	String (0 - 255)	FLOAT	Decimal (precise to 23 digits)
TEXT	String (0 - 65535)	DOUBLE	Decimal (24 to 53 digits)
BLOB	String (0 - 65535)	DECIMAL	"DOUBLE" stored as string
MEDIUMTEXT	String (0 - 16777215)	DATE	YYYY-MM-DD
MEDIUMBLOB	String (0 - 16777215)	DATETIME	YYYY-MM-DD HH:MM:SS
LONGTEXT	String (0 - 4294967295)	TIMESTAMP	YYYYMMDDHHMMSS
LONGBLOB	String (0 - 4294967295)	TIME	HH:MM:SS
TINYINT	Integer (-128 to 127)	ENUM	One of preset options
SMALLINT	Integer (-32768 to 32767)	SET	Selection of preset options

Create Constraints

- Constraints are rules enforced on data columns of a table. It is to limit the type of data that go to a table, ensures the accuracy and reliability of data.
- Below are the commonly used constraints.

Constraint	Description
PRIMARY KEY	A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
UNIQUE	Ensures that all values in a column are different
NOT NULL	Ensures that a column cannot have a NULL value
CHECK (expression)	Ensures that all values in a column satisfies a specific condition
FOREIGN KEY	This is a consistency check which ensures that each value in this column corresponds to another value in a column in another table.
DEFAULT	Sets a default value for a column when no value is specified

PART 3: Database Operations

Inserting Rows

Adding values to the table can use **INSERT INTO** statement. Make sure the order of values is the same order as table columns.

INSERT INTO Statement

- INSERT INTO can contain values for some or all of its columns.

```
INSERT INTO Customers (CustomerName, City, Country)
VALUES ('Cardinal', 'Stavanger', 'Norway');
```

- INSERT INTO can be combined with a SELECT to insert records.

```
INSERT INTO table_name1 (column1, column2,...)
SELECT column1, column2,...
  FROM table_name2
 WHERE condition
```

- The order of values should be in the same order as columns in the table.

Syntax

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

Example

```
INSERT INTO `Shippers` (`ShipperID`, `ShipperName`, `Phone`) VALUES
(1, 'Speedy Express', '(503) 555-9831'),
(2, 'United Package', '(503) 555-3199'),
(3, 'Federal Shipping', '(503) 555-9931');
```

Result

ShipperID	ShipperName	Phone
1	Speedy Express	(503) 555-9831
2	United Package	(503) 555-3199
3	Federal Shipping	(503) 555-9931

PART 3: Database Operations

Updating Table

To update existing rows, use the **UPDATE** command.

UPDATE Statement

Similar to INSERT statement, you need to specify exactly which table, which columns and rows to update.
This requires three pieces of information:

1. The name of the table and column to update
2. The new value of the column
3. Which row(s) to update

Syntax

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

Example

```
UPDATE Customers
SET ContactName='Juan'
WHERE Country='Mexico';
```

Result

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Alfred Schmidt	Obere Str. 57	Frankfurt	12209	Germany
2	Ana Trujillo Emparedados y helados	Juan	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Juan	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

PART 3: Database Operations

Deleting Rows

DELETE statement is used to delete existing records in a table.

DELETE Statement

- DELETE all records if WHERE clause is omitted.
- DELETE only deletes the records, but the table structure, attributes and index will be intact, which is different from DROP statement.
- It's recommended that you run the constraint in a SELECT query before you DELETE rows, to avoid irrevocably removing data.

Syntax

```
DELETE FROM table_name WHERE condition;

DELETE FROM table_name;
```

Example

```
DELETE FROM Customers;
```

Result

Your Database:

Tablename	Records
Customers	91
Categories	8
Employees	10
OrderDetails	518
Orders	196
Products	77
Shippers	3
Suppliers	29

Your Database:

Tablename	Records
Customers	0
Categories	8
Employees	10
OrderDetails	518
Orders	196
Products	77
Shippers	3
Suppliers	29

PART 3: Database Operations

Dropping Tables

DROP TABLE statement removes entire table including all its data and attributes.

DROP Statement

- Be careful before dropping a table. DELETE only removes the rows, but the table attributes and indexes are intact. However, DROP will remove a table and it cannot be rolled back from the database.

Syntax

```
DROP TABLE IF EXISTS table_name
```

Example

```
DROP TABLE Customers
```

Result

Your Database:

Tablename	Records
Customers	91
Categories	8
Employees	10
OrderDetails	518
Orders	196
Products	77
Shippers	3
Suppliers	29

Your Database:

Tablename	Records
Categories	8
Employees	10
OrderDetails	518
Orders	196
Products	77
Shippers	3
Suppliers	29

PART 3: Database Operations

Altering Table

To add, remove, modify columns and table constraints, we can use ALTER TABLE statement.

ALTER TABLE Statement

- You can **ADD** columns by specifying the data type of the columns along with table constraints.

```
ALTER TABLE table_name
ADD column_name datatype;
```

- **DROP** columns. Notice that some databases don't support this feature. Instead you need to create a new table and migrate the data over.

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

- You can use **RENAME TO** clause to rename tables

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

Syntax

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

Example

```
ALTER TABLE Customers
RENAME TO CustomerInfo
```

Result

Your Database:

Tablename	Records
Customers	91
Categories	8
Employees	10
OrderDetails	518
Orders	196
Products	77
Shippers	3
Suppliers	29

Your Database:

Tablename	Records
CustomerInfo	91
Categories	8
Employees	10
OrderDetails	518
Orders	196
Products	77
Shippers	3
Suppliers	29