Summary

MySQL

MySQL is a **relational database** structuring data into **rows** and **columns** where:

- *Rows* are the data
- *Columns* are the attributes

Why Databases

Databases are used to store/manage **huge amounts of data** which is preferred compared using something like *excel* because:

1. **Scalability** -> Increase in size will not affect efficiency
2. **Accuracy** -> Reliable
3. **Flexibility** -> Many people could upload/edit data at the same time
4. **Data Security** -> Privileges to restrict access

To **access** our database, we need to write **queries** because **MySQL** is a **Structured Query Language**

Query Syntax

Overview

1. Create a Database
2. Use/Access to the specific Database
3. Creating a table
4. Modifying the Database
5. Constraints and Operators
6. Inserting in data
7. View Data

Step 1: **Creating** a Database:

First, we need to know about the *Data Types*:

- **Characters**
  - `char(s)` where *s* is the size of characters (Max: 255 Characters)
  - `varchar(s)` where *s* is the size of characters (Max: 255 Characters)
    - Different from char because the **size** is **dynamic**
  - `text` (Max: 65,535 Characters)
- **Dates**
  - `date` (YYYY-MM-DD)
  - `datetime` (YYYY-MM-DD HH:MM)
- **Integers**

    - `int`
    - `bigint`
    - `decimals(m,d)` where *m* is the max digit and *d* is the numbers after the decimal point
        - `decimal(10, 2)` can store numbers like 12345678.90, 123456.78, etc.
    - `float(m, d)` where *m* is the max digit and *d* is the numbers after the decimal point
        - `float(7, 3)` can store numbers like 1234.567, 123.456, etc.

Query Syntax to create a Database: `CREATE DATABASE database_name;` --> `CREATE DATABASE coffee`

You could view all your databases with: `SHOW DATABASES;`

Step 2: You must **Use** a Database in order to access it

`USE database_name;` --> `USE coffee`

Step 3: **Creating** a table within the Database

When we create a **table** you need to know the **attributes** otherwise known as the **columns** and their **data types**

Example:

Coffee Orders: order id --> integer **primary key** customer name --> character (string) Cannot be null price --> integer Default value: 0 coffee type --> character (string) Cannot be null

Code Template

```
CREATE TABLE tablename(
    id INT PRIMARY KEY,
    col1 VARCHAR(30),
    col2 INT,
    col3 DATE
)
```

Coffee Orders

```
CREATE TABLE coffee_order(
    id INT PRIMARY KEY,
    customer_name VARCHAR(50),
    price float(4,2),
    coffee_type VARCHAR(30)
)
```

Every Database Table has a **Primary Key** which is just an **ID** for each row.

- It **CANNOT** be the same
- We Should **Auto Increment** this field so that everytime we add data, this will *automatically* add

Since we created the database already we need to modify...

Step 4: **Modifying** the database (if you must)

To **Auto Increment** the primary key field we use the `ALTER` keyword:

Code Template

```
ALTER TABLE table_name
MODIFY colum_name data_type AUTO_INCREMENT;
```

Coffee Orders

```
ALTER TABLE coffee_order
MODIFY id INT AUTO_INCREMENT;
```

NOTE make sure there's no `;` after the `ALTER TABLE tablename` the `;` should be at the end of the `modify colum_name data_type AUTO_INCREMENT;` statement

You could also **add** columns using this `ALTER` keyword:

Code Template

```
ALTER TABLE table_name
ADD column_name data_type;
```

Coffee Orders

```
ALTER TABLE coffee_order
ADD order_date DATE;
```

Step 5: Modifying the database with **Constraints** (if needed)

NOT NULL constraint:

We don't want a column to be *empty* so we put a **NOT NULL** constriant

Code Template

```
ALTER TABLE table_name
MODIFY column_name data_type NOT NULL;
```

Coffee Orders

```
ALTER TABLE coffee_order
MODIFY customer_name VARCHAR(50) NOT NULL,
MODIFY coffee_type VARCHAR(30) NOT NULL;
```

DEFAULT Constraint

We may want to set a **default option** for some cell in case the user *hasn't entered* any data.

Code Template

```
ALTER TABLE table_name
MODIFY column_name data_type DEFAULT default_val;
```

Coffee Orders

```
ALTER TABLE coffee_order
MODIFY price int DEFAULT 0,
MODIFY order_date DATE DEFAULT (current_date());
```

Unique constraint

If you don't want values in that column having the **same data** as other *cells* (within the same column)

Code Template

```
ALTER TABLE table_name
MODIFY column_name data_type UNIQUE;
```

Coffee Orders

```
ALTER TABLE coffee_order
MODIFY id INT UNIQUE,
-- You could also add a column that's unique
ADD ticket_number int UNIQUE;
```

CHECK constraint

This will **check** all the rows within the column (including existing ones) for a **condition**

This could be accomplished in **two ways**:

Add method

```
ALTER TABLE table_name
ADD CONSTRAINT constraint_name CHECK (column_name < condition);
```

Modify method

```
ALTER TABLE table_name
MODIFY column_name data_type CHECK (column_name < condition);
```

Coffee Orders

```
ALTER TABLE coffee_order
-- Remember that the date should be in: 'YYYY-MM-DD'
ADD CONSTRAINT date_check CHECK (order_date > '2024-08-03');
```

Operators just allows us to use operations like +, -, *, / with our **columns**

Step 6: **Inserting** in data

We could insert add into our table using the `INSERT` and `VALUES()` keyword

Code Template

```
INSERT INTO table_name(col1,col2,col3)
VALUES(val1,val2,val3);
```

Coffee Orders

```
INSERT INTO coffee_order(customer_name, price, coffee_type, ticket_number)
VALUES('Justin',1.51,'espresso', 243);
```

Step 7: **Viewing** the data

We could look at certain columns or everything within our table using the `SELECT` keyword with `FROM`

Code Template

```
SELECT col1,col3,col8 FROM table_name;

SELECT * FROM table_name;
```

Coffee Orders

```
SELECT ticket_number, customer_name, price  FROM coffee_order;

SELECT * FROM coffee_order
```

The * acts as a **wildcard** to select everything that **exists** within the database.

If you want to check your database to see **more details** just run the `DESCRIBE` keyword:

`DESCRIBE database_name;` -> `DESCRIBE coffee_order;`

Coffee Database Script

```
-- Creatinng the Database
CREATE DATABASE coffee;

-- Using the Database
USE coffee;

-- Creating a table
CREATE TABLE coffee_order(
    id INT PRIMARY KEY,
    customer_name VARCHAR(50),
    price float(4,2),
    coffee_type VARCHAR(30)
);

-- Alter to auto-increment our id
ALTER table coffee_order
modify id int AUTO_INCREMENT;

-- Alter to add a column
ALTER table coffee_order
ADD order_date DATE;

-- Alter for NOT NULL constraint
ALTER TABLE coffee_order
MODIFY customer_name VARCHAR(50) NOT NULL,
MODIFY coffee_type VARCHAR(30) NOT NULL;

-- Alter for DEFAULT constraint
ALTER TABLE coffee_order
MODIFY price int DEFAULT 0,
MODIFY order_date DATE DEFAULT (current_date());

-- Alter with a UNIQUE constraint
ALTER TABLE coffee_order
MODIFY id INT UNIQUE,
ADD ticket_number int UNIQUE;

-- Alter with Check Constraint
ALTER TABLE coffee_order
-- Remember that the date should be in: YYYY-MM-DD
ADD CONSTRAINT date_check CHECK (order_date > '2024-08-03');


-- Initial Inserting Data
INSERT INTO coffee_order(id, customer_name, price, coffee_type, ticket_number)
VALUES(1,'Justin',1.51,'espresso', 243);

-- But since our id auto-increment
```

```sql
INSERT INTO coffee_order(customer_name, price, coffee_type, ticket_number)
VALUES('Thy',1.53,'Iced Coffee', 244);

-- But since our id auto-increment
INSERT INTO coffee_order(customer_name, price, coffee_type, ticket_number)
VALUES('Muffin',3.53,'americano', 245);

-- Viewing Data
SELECT * FROM coffee_order;
SELECT ticket_number, customer_name, price  FROM coffee_order;

-- Checking More Details
DESCRIBE coffee_order;
```
