MySQL Tutorial

SQL stands for Structured Query Language

It has Data Definition Language instructions (DDL)

It has Data Manipulation Language instructions (DML)

MySQL is a Relational Database Management System (RDBMS)

DDL

Create

```
CREATE DATABASE StarTrekShips;
```

Drop

```
DROP DATABASE StarTrekShips;
```

Create Table

```
CREATE TABLE Ships (
    ShipID INT AUTO_INCREMENT PRIMARY KEY, -- Unique identifier for each ship
    ShipName VARCHAR(100) NOT NULL,        -- Name of the ship
    ShipClass VARCHAR(100) NOT NULL,       -- Class of the ship
    CaptainName VARCHAR(100),              -- Name of the ship's captain
    YearCommissioned YEAR,                 -- Year the ship was commissioned
    MaxWarpSpeed DECIMAL(2, 1),            -- Maximum warp speed
    Notes TEXT                             -- Additional notes or details
);
```

Drop Table

```
DROP TABLE Ships
```

Alter Table

```
ALTER TABLE Ships
ADD RegistryNumber VARCHAR(50);

ALTER TABLE Ships
DROP COLUMN Notes;

ALTER TABLE Ships
MODIFY MaxWarpSpeed DECIMAL(5, 3);
```

Constraints

```
ALTER TABLE Ships
ADD RegistryNumber VARCHAR(50) UNIQUE;
```

Not Null

```
ALTER TABLE Ships
MODIFY YearCommissioned YEAR NOT NULL;
```

Unique

```
ALTER TABLE Ships
ADD RegistryNumber VARCHAR(50) UNIQUE;
```

Primary Key

```
ALTER TABLE Ships
ADD PRIMARY KEY (ShipID);
```

Foreign Key

```
CREATE TABLE ShipTypes (
    ShipTypeID INT AUTO_INCREMENT PRIMARY KEY, -- Unique identifier for each ship type
    ShipID INT,                                -- Foreign key to link to Ships table
    TypeName VARCHAR(100) NOT NULL UNIQUE,     -- Name of the ship type
    Description TEXT,                          -- Description of the ship type
    FOREIGN KEY (ShipID) REFERENCES Ships(ShipID) -- Establish the relationship
);
```

Check

```
ALTER TABLE Ships
ADD CONSTRAINT CHK_MaxWarpSpeed_Range CHECK (MaxWarpSpeed BETWEEN 0 AND 10);
```

Default

```
ALTER TABLE ShipTypes
MODIFY Description TEXT DEFAULT 'No description available';
```

Create Index

```
CREATE INDEX IDX_ShipName ON Ships (ShipName);
```

Auto Increment

```
ALTER TABLE Ships
MODIFY ShipID INT AUTO_INCREMENT PRIMARY KEY;
```

Views

```
CREATE VIEW ShipNamesAndCaptains AS
SELECT ShipName, Captain
FROM Ships;
```

DML

SELECT

```
SELECT * FROM Ships;
```

WHERE

```
SELECT ShipName, YearCommissioned
FROM Ships
WHERE YearCommissioned > 2300;
```

AND, OR, NOT

```
SELECT ShipName, TypeName, MaxWarpSpeed
FROM Ships
WHERE (TypeName = 'Constitution' OR MaxWarpSpeed > 8.0)
AND NOT ShipName = 'Enterprise';
```

ORDER BY

```
SELECT ShipName, YearCommissioned
FROM Ships
ORDER BY YearCommissioned DESC;
```

INSERT INTO

```
INSERT INTO Ships (ShipName, ShipClass, MaxWarpSpeed, YearCommissioned)
VALUES ('USS Voyager', 'Intrepid', 9.975, 2371);
```

NULL

```
SELECT ShipName, Captain
FROM Ships
WHERE Captain IS NULL;
```

UPDATE

```
UPDATE Ships
SET Captain = 'Kathryn Janeway'
WHERE ShipName = 'USS Voyager';

UPDATE Ships
SET MaxWarpSpeed = 9.0, YearCommissioned = 2350
WHERE ShipID IN (1, 2, 3);
```

DELETE

```
DELETE FROM Ships
WHERE ShipName = 'USS Voyager';
```

LIMIT

```
SELECT ShipName, ShipClass
FROM Ships
LIMIT 5;
```

MIN and MAX

```
SELECT MIN(MaxWarpSpeed) AS MinWarp, MAX(MaxWarpSpeed) AS MaxWarp
FROM Ships;
```

COUNT, AVG, SUM

```
SELECT COUNT(*) AS TotalShips, AVG(MaxWarpSpeed) AS AverageWarpSpeed
FROM Ships
WHERE YearCommissioned > 2300;
```

LIKE

```
SELECT ShipName
FROM Ships
WHERE ShipName LIKE 'E%';  -- Starts with E

SELECT ShipName
FROM Ships
WHERE ShipName LIKE '%E';  -- Ends with E

SELECT ShipName
FROM Ships
WHERE ShipName LIKE '%E%';   -- Contains E
```

Wildcards

```
SELECT ShipName
FROM Ships
WHERE ShipName LIKE '%NCC%';

SELECT ShipName
FROM Ships
WHERE ShipName LIKE 'USS_';
```

IN

```
SELECT ShipName, TypeName
FROM Ships
WHERE TypeName IN ('Constitution', 'Galaxy');
```

BETWEEN

```
SELECT ShipName, YearCommissioned
FROM Ships
WHERE YearCommissioned BETWEEN 2300 AND 2400;
```

Aliases

```
SELECT ShipName AS Name, YearCommissioned AS Year
FROM Ships;
```

INNER JOIN

The INNER JOIN is used when you want to retrieve only the records that have matching values in both tables being joined.

```
SELECT ShipName, TypeName
FROM Ships
INNER JOIN ShipTypes ON Ships.ShipID = ShipTypes.ShipID;
```

LEFT JOIN

The LEFT JOIN returns all records from the left table and the matched records from the right table.

```
SELECT ShipName, TypeName
FROM Ships
LEFT JOIN ShipTypes ON Ships.ShipID = ShipTypes.ShipID;
```

RIGHT JOIN

The RIGHT JOIN is like the LEFT JOIN, but it returns all records from the right table and the matching records from the left table.

```
SELECT ShipName, TypeName
FROM Ships
RIGHT JOIN ShipTypes ON Ships.ShipID = ShipTypes.ShipID;
```

Self Join

A SELF JOIN is when a table is joined with itself. It's typically used when you want to compare rows within the same table. You give the table two different aliases to distinguish between the two instances of the same table.

Example: To find ships that belong to the same class, a self join would be used to compare each ship to others in the same class.

```
SELECT s1.ShipName AS Ship1, s2.ShipName AS Ship2
FROM Ships s1
JOIN Ships s2 ON s1.ShipClass = s2.ShipClass
WHERE s1.ShipID != s2.ShipID;
```

UNION

UNION removes duplicates: If a record appears in both SELECT statements, it will only appear once in the result.

```
SELECT ShipName
FROM Ships
WHERE YearCommissioned > 2300
UNION
SELECT ShipName
FROM Ships
WHERE MaxWarpSpeed > 9.0;
```

GROUP BY

```
SELECT TypeName, AVG(MaxWarpSpeed) AS AverageWarp
FROM Ships
GROUP BY TypeName;

SELECT ShipName, YearCommissioned
FROM Ships
ORDER BY YearCommissioned ASC;

SELECT ShipName, YearCommissioned, MaxWarpSpeed
FROM Ships
ORDER BY YearCommissioned ASC, MaxWarpSpeed DESC;
```

HAVING

The HAVING clause is used when you want to filter groups after performing aggregation (such as calculating averages or sums).

The WHERE clause is used for filtering individual rows before any aggregation.

```
SELECT TypeName, AVG(MaxWarpSpeed) AS AverageWarp
FROM Ships
GROUP BY TypeName
HAVING AVG(MaxWarpSpeed) > 8.0;
```

EXISTS

The EXISTS operator in SQL is used to check if a subquery returns any results. It returns true if the subquery returns one or more rows, and false if it returns no rows.

```
SELECT ShipName
FROM Ships
WHERE EXISTS (
    SELECT 1
    FROM ShipTypes
    WHERE TypeName = 'Galaxy'
);
```

ANY, ALL

```
SELECT ShipName
FROM Ships
WHERE MaxWarpSpeed > ANY (
    SELECT MaxWarpSpeed
    FROM Ships
    WHERE YearCommissioned > 2300
);
```

INSERT SELECT

```
INSERT INTO NewShips (ShipName, YearCommissioned)
SELECT ShipName, YearCommissioned
FROM Ships
WHERE YearCommissioned > 2300;
```

CASE

```
SELECT ShipName, MaxWarpSpeed,
    CASE
       WHEN MaxWarpSpeed >= 9.0 THEN 'Fast'
       WHEN MaxWarpSpeed >= 8.0 THEN 'Medium'
       ELSE 'Slow'
    END AS SpeedCategory
FROM Ships;
```

Completed 50 of 50 Exercises:

- MySQL Select ✔
- MySQL Where ✔
- MySQL Order By ✔
- MySQL Insert ✔
- MySQL Null ✔
- MySQL Update ✔
- MySQL Delete ✔
- MySQL Functions ✔
- MySQL Like ✔
- MySQL Wildcards ✔
- MySQL In ✔
- MySQL Between ✔
- MySQL Alias ✔
- MySQL Join ✔
- MySQL Group By ✔
- MySQL Database ✔