INDIAN SCHOOL AL WADI AL KABIR

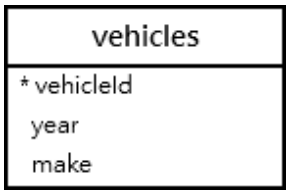
2022 -23

| Class: X | SUBJECT: COMPUTER SCIENCE |
|---|---|
| Handout-3<br>MYSQL | TOPIC: DATA DEFINITION LANGUAGE (DDL) & DATA MANIPULATION LANGUAGE (DML) |

# MySQL ALTER TABLE COMMANDS

**Sample table demonstration: vehicles**

```
CREATE TABLE vehicles (
    vehicleId INT,
    year INT,
    make VARCHAR(100),
    PRIMARY KEY(vehicleId)
);
```




## 1. MySQL ALTER TABLE – Add columns to a table

The ALTER TABLE ADD statement allows you to add one or more columns to a table.

1) Add a column to a table

To add a column to a table, you use the ALTER TABLE ADD syntax:

```
ALTER TABLE table_name
ADD
new_column_name column_definition
  [FIRST | AFTER column_name]
```

The following example uses the ALTER TABLE ADD statement to add a column at the end of the vehicles table:

**ALTER TABLE vehicles**

**ADD model VARCHAR(100) NOT NULL;**

**DESCRIBE vehicles;**

| Field | Type | Null | Key | Default | Extra |
|---|---|---|---|---|---|
| vehicleId | int(11) | NO | PRI | NULL | |
| year | int(11) | NO | | NULL | |
| make | varchar(100) | NO | | NULL | |
| model | varchar(100) | NO | | NULL | |

**2) Add multiple columns to a table**

To add multiple columns to a table, you use the following form of the ALTER TALE ADD statement:

**ALTER TABLE table_name**

**ADD new_column_name column_definition**

**[FIRST | AFTER column_name],**

**ADD new_column_name column_definition**

**[FIRST | AFTER column_name],**

**...;**

For example, this statement adds two columns color and note to the vehicles table:

**ALTER TABLE vehicles**

**ADD color VARCHAR(50),**

**ADD note VARCHAR(255);**

This statement shows the new structure of the vehicles table:

**DESCRIBE vehicles;**

| Field | Type | Null | Key | Default | Extra |
|---|---|---|---|---|---|
| vehicleId | int(11) | NO | PRI | NULL | |
| year | int(11) | NO | | NULL | |
| make | varchar(100) | NO | | NULL | |
| model | varchar(100) | NO | | NULL | |
| color | varchar(50) | YES | | NULL | |
| note | varchar(255) | YES | | NULL | |

## 2. MySQL ALTER TABLE – Modify columns

1) **Modify a column**

Here is the basic syntax for modifying a column in a table:

**ALTER TABLE table_name**

**MODIFY column_name column_definition**

**[ FIRST | AFTER column_name];**

**DESCRIBE vehicles;**

| Field | Type | Null | Key | Default | Extra |
|---|---|---|---|---|---|
| vehicleId | int(11) | NO | PRI | NULL | |
| year | int(11) | NO | | NULL | |
| make | varchar(100) | NO | | NULL | |
| model | varchar(100) | NO | | NULL | |
| color | varchar(50) | YES | | NULL | |
| note | varchar(255) | YES | | NULL | |

Then, modify the note column:

**ALTER TABLE vehicles**

**MODIFY note VARCHAR(100) NOT NULL;**

**DESCRIBE vehicles;**

| Field | Type | Null | Key | Default | Extra |
|---|---|---|---|---|---|
| vehicleId | int(11) | NO | PRI | NULL | |
| year | int(11) | NO | | NULL | |
| make | varchar(100) | NO | | NULL | |
| model | varchar(100) | NO | | NULL | |
| color | varchar(50) | YES | | NULL | |
| note | varchar(100) | NO | | NULL | |

## 3. MySQL ALTER TABLE – Rename a column in a table

To rename a column, you use the following statement:

**ALTER TABLE table_name**
**CHANGE COLUMN original_name new_name column_definition**
**[FIRST | AFTER column_name];**

**ALTER TABLE vehicles**
**CHANGE COLUMN note vehicleCondition VARCHAR(100) NOT NULL;**

**DESCRIBE vehicles;**

| Field | Type | Null | Key | Default | Extra |
|---|---|---|---|---|---|
| vehicleId | int(11) | NO | PRI | NULL | |
| year | smallint(6) | NO | | NULL | |
| make | varchar(100) | NO | | NULL | |
| color | varchar(20) | YES | | NULL | |
| model | varchar(100) | NO | | NULL | |
| vehicleCondition | varchar(100) | NO | | NULL | |

## 4. MySQL ALTER TABLE – Drop a column

To drop a column in a table, you use the ALTER TABLE DROP COLUMN statement:

**ALTER TABLE table_name**
**DROP COLUMN column_name;**

This example shows how to remove the vehicleCondition column from the vehicles table:
**ALTER TABLE vehicles**
**DROP COLUMN vehicleCondition;**

## 5. MySQL ALTER TABLE – Rename table

To rename a table, you use the ALTER TABLE RENAME TO statement:

**ALTER TABLE table_name**
**RENAME TO new_table_name;**

This example renames the vehicles table to cars:
**ALTER TABLE vehicles**
**RENAME TO cars;**

# MySQL UPDATE COMMAND

The following illustrates the basic syntax of the UPDATE statement:

```
UPDATE [LOW_PRIORITY] [IGNORE] table_name
SET
   column_name1 = expr1,
   column_name2 = expr2,
   ...
[WHERE
   condition];
```

**Sample table demonstration: employees**


```
UPDATE employees
SET
   email = 'mary.patterson@classicmodelcars.com'
WHERE
   employeeNumber = 1056;
```

# MySQL DELETE COMMAND

To delete data from a table, you use the MySQL DELETE statement. The following illustrates the syntax of the DELETE statement:

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

**Example:**

```
DELETE FROM employees
WHERE officeCode = 4;
```

To delete all rows from the employees table, you use the DELETE statement without the WHERE clause as follows:

**DELETE FROM employees;**

## MySQL DROP TABLE COMMAND

**DROP TABLE [table name];**

**Example:**
**DROP TABLE insurances;**

## MySQL DISTINCT COMMAND

When querying data from a table, you may get duplicate rows. To remove these duplicate rows, you use the DISTINCT clause in the SELECT statement.
Here's the syntax of the DISTINCT clause:

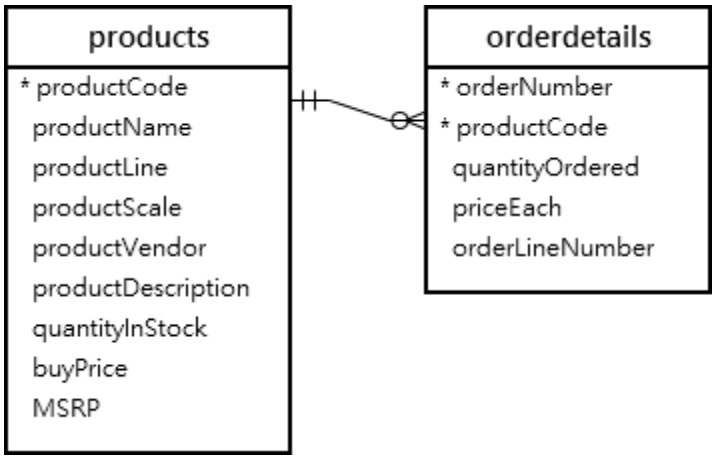
```
SELECT DISTINCT
   select_list
FROM
   table_name
WHERE
   search_condition
ORDER BY
   sort_expression;
```

**Example:**

```
SELECT DISTINCT
   state, city
FROM
   customers
WHERE
   state IS NOT NULL
ORDER BY
   state,
   city;
```

# MYSQL AGGREGATE FUNCTIONS

We will use the products and orderdetails tables for demonstration:




## 1. MySQL aggregate function – AVG() function.

The AVG() function calculates the average value of a set of values. It ignores NULL in the calculation.

AVG(expression)

For example, you can use the AVG function to calculate the average buy price of all products in the products table by using the following query:

```
SELECT
   AVG(buyPrice) average_buy_price
FROM
   products;
```

| average_buy_price |
|---|
| 54.395182 |

## 2. MySQL aggregate function – COUNT() function.

```
SELECT
   COUNT(*) AS total
FROM
   products;
```

| total |
|---|
| 110 |

## 3. MySQL aggregate function – SUM() function

```
SELECT
   productCode,
   SUM(priceEach * quantityOrdered) total
FROM
   orderDetails;
```

| productCode | total |
|---|---|
| S18_3232 | 276839.98 |
| S12_1108 | 190755.86 |
| S10_1949 | 190017.96 |
| S10_4698 | 170686.00 |
| S12_1099 | 161531.48 |
| S12_3891 | 152543.02 |
| S18_1662 | 144959.91 |
| S18_2238 | 142530.63 |
| S18_1749 | 140535.60 |
| S12_2823 | 135767.03 |
| S24_3856 | 134240.71 |
| S12_3148 | 132363.79 |

## 4. MySQL aggregate function – MAX() function

```
SELECT
    MAX(buyPrice) highest_price
FROM
    products;
```

| highest_price |
|---|
| 103.42 |

## 5. MySQL aggregate function – MIN() function

```
SELECT
   MIN(buyPrice) lowest_price
FROM
   products;
```

| lowest_price |
|---|
| 15.91 |