More Basic Operations - ALTER, DROP, RENAME

The Alter Query

- Sometimes we need to incorporate changes to an already existing tables. For example, renaming a field, changing the data-type, etc

- The *alter* command is used to make modification in an existing database/table

- Alter command is generally used with clauses such as change, modify, add, drop

The Alter Query - Change Clause

- To make changes in the column's definition we use the *Change* clause
- The change clause allows you to:
  - Change the name of the column
  - Change the column data type
  - Change column constraints

Syntax:

```
ALTER TABLE table_name CHANGE  old_column_name
new_column_name data type;
```

The Alter Query - Change Clause

Great Learning
POWER AHEAD

Changing Column Definition

- The *ALTER TABLE* command is used to specify the change in the structure of a table
- This is followed by the *CHANGE* clause that tells the MySql server that we want to change the column name
- The *CHANGE* clause is followed by an existing column name that needs to be changed
- And finally, we mention the new definition (new name, new data type, new constraint(optional))

The Alter Query - Change Clause

- Consider a table *Customer* with below fields

Field	Type	Null	Key	Default	Extra
customer_id	int(11)	YES		NULL	
First_name	varchar(10)	YES		NULL	
Second_name	varchar(10)	YES		NULL	
City	varchar(20)	YES		NULL	
Total_exp	varchar(10)	YES		NULL	

- Here, we need to rename 'Second_name' as 'last_name' with increase in the number of characters

The Alter Query - Change Clause

- Use below *alter* query to change the name of the field 'Second_name' to 'last_name'

```
ALTER TABLE Customer CHANGE Second_name last_name varchar(20);
```

- Use **describe *Customer*** to check if the column name has changed to the desired column name

Field	Type	Null	Key	Default	Extra
customer_id	int(11)	YES		NULL	
First_name	varchar(10)	YES		NULL	
last_name	varchar(20)	YES		NULL	
City	varchar(20)	YES		NULL	
Total_exp	varchar(10)	YES		NULL	

The Alter Query - Modify Clause

- The *Modify* clause allows you to:
  - Modify Column Data Type
  - Modify Column Constraints

Syntax:

```
ALTER TABLE table_name MODIFY current_column_name data
type constraint;
```

Great Learning
POWER AHEAD

Modify clause CANNOT be used to rename a column

The Alter Query - Change Clause

Great Learning
POWER AHEAD

Modifying Column Definition

- The *ALTER TABLE* command is used to specify the change in the structure of a table
- This is followed by the *MODIFY* clause that tells the MySql server that we want to modify a column
- The *MODIFY* clause is followed by an existing column name that needs to be changed
- And finally, we mention the new definition of that column (new data type, new constraint(optional))

The Alter Query - Modify Clause

- Consider a table *Customer* with below fields

Field	Type	Null	Key	Default	Extra
customer_id	int(11)	YES		NULL	
First_name	varchar(10)	YES		NULL	
Second_name	varchar(10)	YES		NULL	
City	varchar(20)	YES		NULL	
Total_exp	varchar(10)	YES		NULL	

- Here, we need to increase the width of 'First_name' field from 10 to 25

The Alter Query - Modify Clause

- Use below *alter* query to change the width of 'First_name' to varchar(25) with a NOT NULL constraint

```
ALTER TABLE Customer MODIFY First_name varchar(25) NOT NULL;
```

- Use **describe *Customer*** to check if the column name has changed to the desired column name

Field	Type	Null	Key	Default	Extra
customer_id	int(11)	YES		NULL	
First_name	varchar(25)	NO		NULL	
Second_name	varchar(10)	YES		NULL	
City	varchar(20)	YES		NULL	
Total_exp	varchar(10)	YES		NULL	

Difference between Change and Modify Clause

- If you have already created your MySQL database, and decide after the fact that one of your columns is named incorrectly, you can simply rename it using *CHANGE*

- *MODIFY* does everything *CHANGE* can, but without renaming the column

The Alter Query - Add Clause

- The *Add* clause allows you to:
  - Add a new column to an existing table
  - Add primary key constraint to an existing column

The Alter Query - Add Clause

Great Learning
POWER AHEAD

Adding a new column to a table

- To add a new column to an existing table, we use the *ADD COLUMN* clause with the *ALTER* command in the following way

Syntax:

```
ALTER TABLE table_name ADD COLUMN column_name
```

The Alter Query - Add Clause

- Consider the previously created table *Customer:*

Field	Type	Null	Key	Default	Extra
customer_id	int(11)	YES		NULL	
First_name	varchar(10)	YES		NULL	
last_name	varchar(20)	YES		NULL	
City	varchar(20)	YES		NULL	
Total_exp	varchar(10)	YES		NULL	

- Here, we add a new column 'Salary' to this table

The Alter Query - Add Clause

- Use below *alter* query with the add clause:

```
ALTER TABLE Customer ADD COLUMN Salary int;
```

- Use **describe *Customer*** to check if a new column has been added to the table

Field	Type	Null	Key	Default	Extra
customer_id	int(11)	YES		NULL	
First_name	varchar(10)	YES		NULL	
last_name	varchar(10)	YES		NULL	
City	varchar(20)	YES		NULL	
Total_exp	varchar(10)	YES		NULL	
Salary	int(11)	YES		NULL	

*By default, the ADD clause adds a column at the end of the table. Use the **AFTER** keyword to add a column at a particular position in a table*

- For example: To add a 'Date_of_Birth' column after 'last_name' column in the table Customer, use the following query :

```
ALTER TABLE Customer ADD Date_of_Birth date AFTER 'last_name';
```

*By default, the ADD clause adds a column at the end of the table. Use the **AFTER** keyword to add a column at a particular position in a table*

- Use **`describe Customer`** to check the table definition

Field	Type	Null	Key	Default	Extra
customer_id	int(11)	YES		NULL	
First_name	varchar(10)	YES		NULL	
last_name	varchar(10)	YES		NULL	
Date_of_Birth	date	YES		NULL	
City	varchar(20)	YES		NULL	
Total_exp	varchar(10)	YES		NULL	
Salary	int(11)	YES		NULL	

The Alter Query - Drop Clause

Great Learning
POWER AHEAD

Dropping a column from the table

- Suppose you no longer need a column from a table for your analysis

- In this scenario we use the *ALTER* command with the *DROP* clause to remove a column from the table

Syntax:

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

The Alter Query - Drop Clause

- Consider a table *Customer* with below fields:

Field	Type	Null	Key	Default	Extra
customer_id	int(11)	YES		NULL	
First_name	varchar(10)	NO		NULL	
last_name	varchar(10)	YES		NULL	
City	varchar(20)	YES		NULL	
Total_exp	varchar(10)	YES		NULL	
Salary	int(11)	YES		NULL	

- Here, we don't need the column 'Salary' from the table

The Alter Query - Drop Clause

- Use below *alter* query to drop the 'Salary' column from the table Customer

```
ALTER TABLE Customer DROP COLUMN Salary;
```

- Use **describe *Customer*** to check if the column has been drop from the table

Field	Type	Null	Key	Default	Extra
customer_id	int(11)	YES		NULL	
First_name	varchar(10)	NO		NULL	
last_name	varchar(10)	YES		NULL	
City	varchar(20)	YES		NULL	
Total_exp	varchar(10)	YES		NULL	

Revisiting The Drop Query

- The DROP query allows you to:
  - Delete a database
  - Delete an existing table from the database

Syntax to delete an existing database:

```
DROP DATABASE database_name
```

Syntax to delete an existing table in a database:

```
DROP TABLE table_name
```

The Rename Query

- The rename command is used to change the name of an existing database table to a new name

- Renaming a table does not make it to lose any data is contained within it

Syntax:

```
RENAME TABLE current_table_name TO new_table_name
```

The Rename Query

- Rename the current Customer table to Customer_info

	Tables_in_misc
▶	customer

You can use **`show tables`** command to retrieve the name of all the tables present in a database. 'misc' is the name of the database

The Rename Query

- Below command changes the name of the table Customer to Customer_info:

```
RENAME TABLE Customer TO Customer_info
```

- The name of the table Customer is now changed to customer_info:

	Tables_in_misc
▶	customer_info

Thank You