



Introduction to DBMS

Agenda

- Introduction to DBMS and RDBMS
- Key attributes
- Introduction to SQL
 - Data Types
- SQL Commands
 - DDL
 - DML
 - DQL
 - Select Statements

What is DBMS?

- DBMS stand for Database Management Systems
- These are systems to store, retrieve or ,sometimes, manipulate data
- Developed to handle large amount of data

Why DBMS?

- Consider a bank that maintains customer's account details, employee details, bank device details, etc.
- This details needs to be stored in such a way that it can be added, deleted, updated and retrieved from one place
- DBMS is a software designed for this type of operations



RDBMS

What is RDBMS?

- RDBMS stand for Relational Database Management Systems
- RDBMS allows to store, retrieve or manipulate data, but in a more efficient way than DBMS
- Apart from rows and columns the RDBMS table has following components
 - Domain
 - Instance
 - Schema
 - Keys

A database table

- A database consists of one or more tables
- A table is the most significant component in an RDBMS
- A table is where all data is stored
- A table constitutes of rows & columns
- Each column represents attributes of the entity

First Name	Last Name	Address	City	Age
Mickey	Mouse	123 Fantasy Way	Anaheim	73
Bat	Man	321 Cavern Ave	Gotham	54
Wonder	Woman	987 Truth Way	Paradise	39
Donald	Duck	555 Quack Street	Mallard	65
Bugs	Bunny	567 Carrot Street	Rascal	58
Wiley	Coyote	999 Acme Way	Canyon	61
Cat	Woman	234 Purrfect Street	Hairball	32
Tweety	Bird	543	Itottlaw	28

A record in a table

- Each row in a table is a record/tuple
- Each record is all of the information for each object, say a person or a product

First Name	Last Name	Address	City	Age
Mickey	Mouse	123 Fantasy Way	Anaheim	73
Bat	Man	321 Cavern Ave	Gotham	54
Wonder	Woman	987 Truth Way	Paradise	39
Donald	Duck	555 Quack Street	Mallard	65
Bugs	Bunny	567 Carrot Street	Rascal	58
Wiley	Coyote	999 Acme Way	Canyon	61
Cat	Woman	234 Purrfect Street	Hairball	32
Tweety	Bird	543	Itotltaw	28

A column in a table

- Each column in a table is an attribute
- This gives one piece of information about the attribute. For example, last name of a customer

First Name	Last Name	Address	City	Age
Mickey	Mouse	123 Fantasy Way	Anaheim	73
Bat	Man	321 Cavern Ave	Gotham	54
Wonder	Woman	987 Truth Way	Paradise	39
Donald	Duck	555 Quack Street	Mallard	65
Bugs	Bunny	567 Carrot Street	Rascal	58
Wiley	Coyote	999 Acme Way	Canyon	61
Cat	Woman	234 Purrfect Street	Hairball	32
Tweety	Bird	543	Itotltaw	28

Keys

- A key is a data item (a column or a set of column) to uniquely identify a record in a table
- It is used to fetch a single or a set of records from a table
- Keys can also provide several types of useful constraints. For example, a unique key constraint can help avoid enter a duplicate value

Keys

A database supports various types of keys. Some of them are

- Candidate key
- Primary key
- Foreign key
- Unique key
- Alternate key

Keys

- Candidate key: An attribute (column) of a set of attributes that uniquely identifies a record
 - eg.: Customer ID + Store ID + Location ID in a customer transaction table
- Primary key: Identifies each record in a table and must never be the same for 2 records in a table
 - eg..: Customer ID identifying a customer uniquely in a customer table

Difference between Candidate Key and Primary Key

Candidate Key	Primary Key
Candidate Key can be any column or a combination of columns that can qualify as unique key in database	A Primary Key is a column or a combination of columns that uniquely identify a record
There can be multiple Candidate Keys in one table. Each Candidate Key can qualify as Primary Key.	There is only one Primary Key in a table

Keys

- Foreign key: Foreign keys are the columns of a table that points to the primary key of another table
 - eg.: CustomerID in customer transaction table points to the CustomerID in the customer tables

Keys

- Alternate key: An alternate key is a candidate key that is not considered as a primary key
 - eg.: Store Name + Store Location in a store information table
- Unique key: An attribute or a set of attributes to uniquely identify a record in a table. This is similar to a primary key but can contain a null value.
 - eg.: Store Name in a store information table

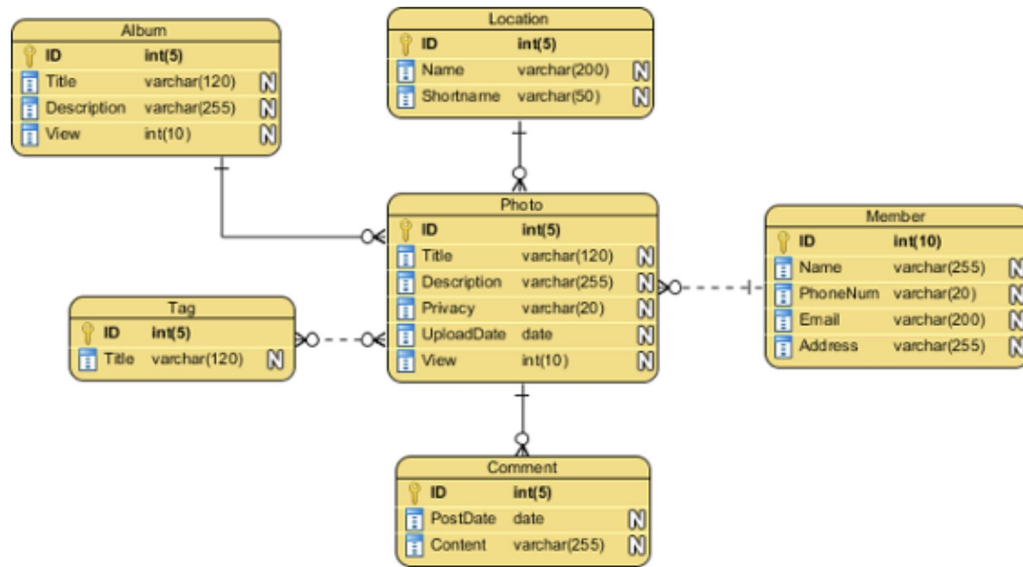
Domain

- A domain is a set of values that an attribute can take
- An attribute would not accept any value outside of its domain
- For example, in a bank customer table, the field "account_no" will only accept integer values if you give the domain of the field as integer
- Apart from data type, you can set constraints to the attributes as well
- Such combination of constraints are known as Domain Constraints

Schemas

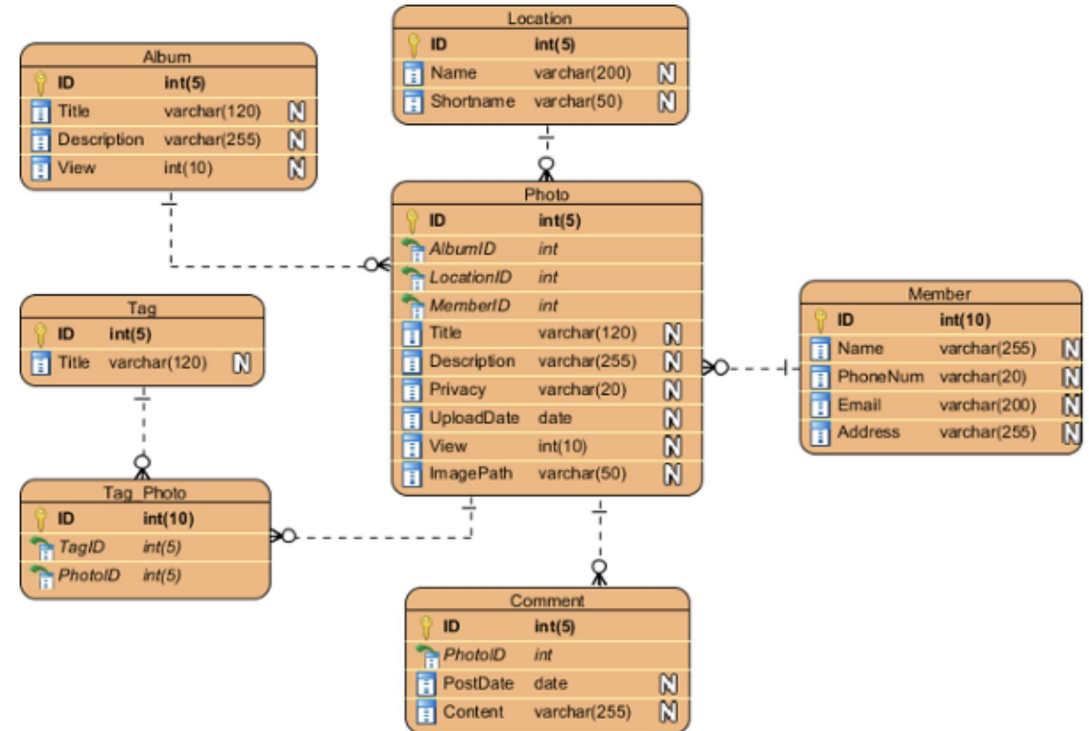
- A database schema is a blueprint that represents the logical view of the database
- It defines tables and the relationship between them
- A database schema is broadly categorized into
 - Physical Schema: How data is stored in actual storage is described at this level
 - Logical Schema: Pertains to the logical constraints (design) that need to be applied to the data stored

Logical vs Physical Schemas



Logical ERD example

Logical schema is designed based on information gathered from business requirements. This schema need not have any column types defined. But in case you do so it is with the intent to help in business analysis



Physical ERD example

Physical schema represents actual blueprint of a relational database. In this schema, the data types, primary key, foreign keys and constraints are to be designed by the database designers.

Instance

- In RDBMS, there are lot of changes taking place in a table, over time
- Data get inserted, manipulated and deleted in parallel
- The data stored in a database at a particular moment of time is called instance

Instance

- For example, consider 'customer' table in the database 'bank' has 10,000 records, so the instance of the database at this point is 10,000
- Let's say, we are going to add 1000 more records in the same table, tomorrow
- So the instance of the database will be 11,000 tomorrow



*There are many benefits to having a DBMS, but one of the most important is the **security** it provides to your business or organization.*

Using a database management system lets end users and programmers access the same data without compromising its integrity.



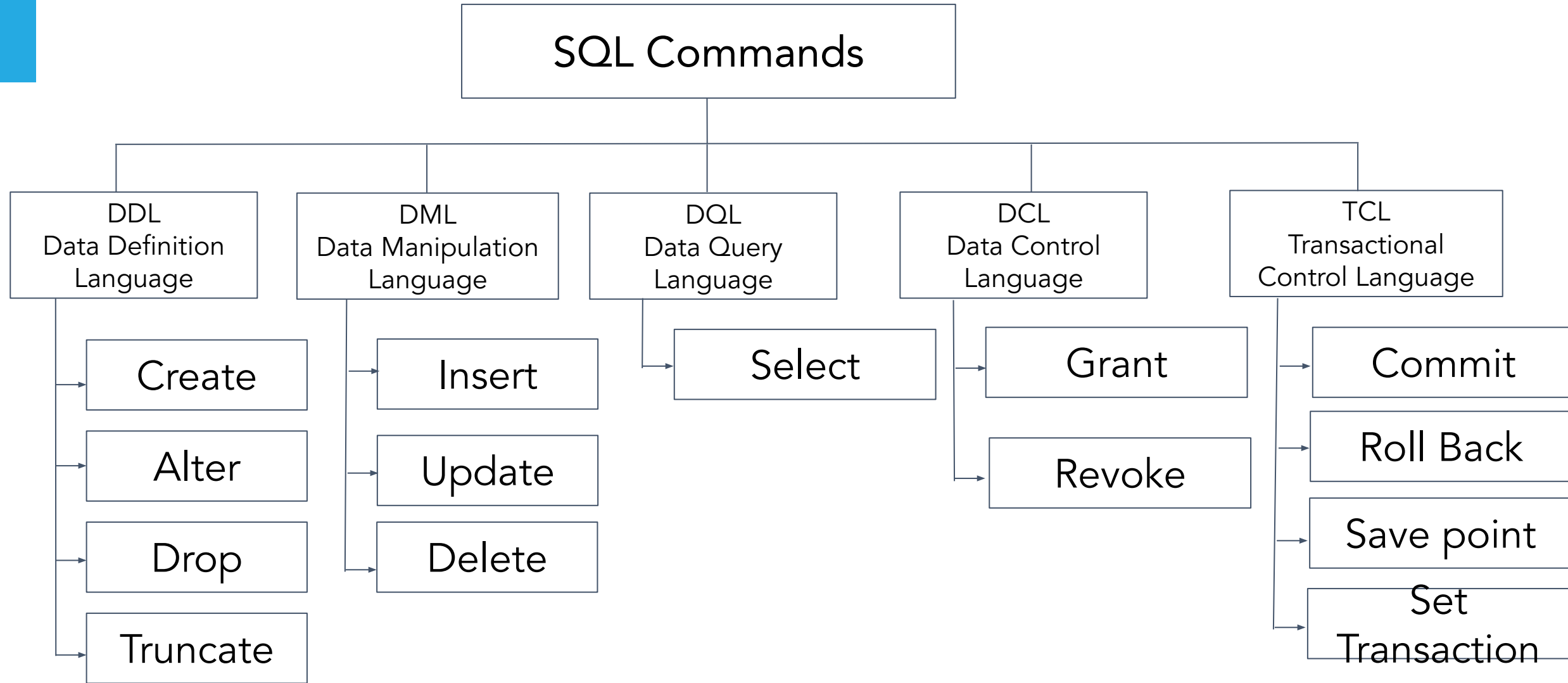
Structured Query Language (SQL) Basics

SQL Introduction

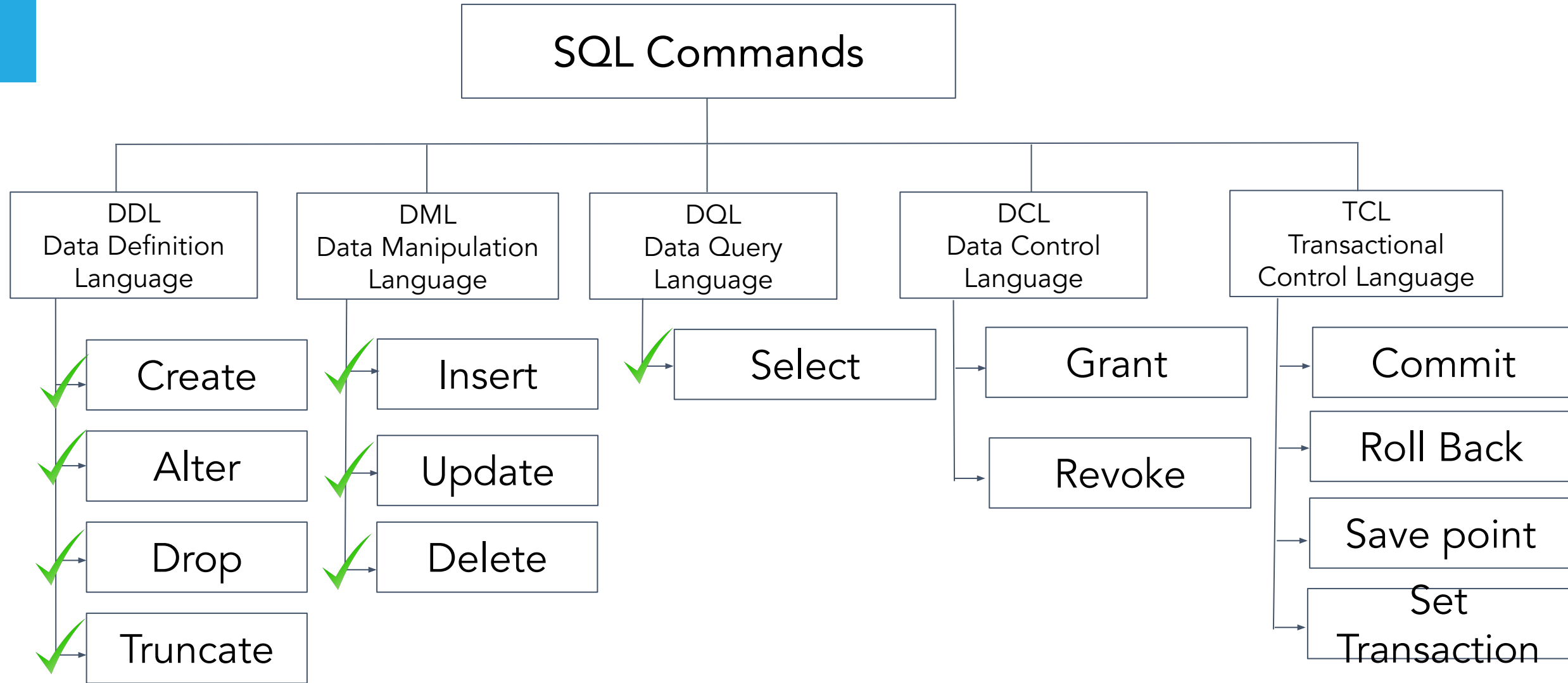
The commands available in SQL can be broadly categorised as follows:

- Data Definition Language (DDL)
- Data Manipulation Language (DML)
- Data Query Language (DQL)
- Data Control Language (DCL)
- Transactional Control Language (TCL)

Types of SQL Commands



Types of SQL Commands





Data Definition Language (DDL)

Data Definition Language (DDL)

- A database is a collection of many tables, and a database server can hold many of these databases

Database Server —> Databases —> Tables (defined by columns) —> Rows

- Databases and tables are referred to as database objects
- Any operation, such as *creating*, *modifying*, or *deleting* database objects, is called Data Definition Language (DDL)

Data Definition Language (DDL)

- DDL is used to create a new schema as well as to modify an existing schema
- The typical commands available in DDL are:
 - CREATE
 - ALTER
 - DROP
 - TRUNCATE



Create Database

DDL - CREATE DATABASE- Syntax

- The CREATE DATABASE statement is used to create a new SQL database

Syntax:

```
CREATE DATABASE databasename;
```

The semicolon character (;) is a SQL statement terminator

- To create tables in the database you need to first select the database. Use the following syntax to select the database:

```
USE databasename;
```

DDL - CREATE DATABASE - Example

- We will create a database called company

```
CREATE DATABASE company;
```

- Use **SHOW DATABASES** to check if the database *company* has been **CREATED**

Database
▶ company
information_schema
mysql
performance_schema
sys

The database *company* is now created

Default system databases



Drop Database

DDL - DROP DATABASE - Syntax

- The DROP DATABASE statement is used to drop an existing SQL database

Syntax:

```
DROP DATABASE dbname;
```



Database name

DDL - DROP DATABASE - Example

- The following SQL statement drops the existing database "company":

```
DROP DATABASE company;
```

- Use **SHOW DATABASES** to check if the database *company* has been **DROPPED**

Database
information_schema
mysql
performance_schema
sys



We see the database *company* is now dropped



Create Tables

Prerequisites for Creating Tables


- To create and maintain table, you need a database
- While defining columns in a table, you should mention:
 - the name of the columns,
 - datatype (integer, floating point, string, and so on), and
 - default value (if any)
- Let's take a look at data types before we create table



Data Types

Built - in Data Types

- Numeric: TINYINT, SMALLINT, MEDIUMINT, INT, BIGINT, and BIT
- Floating numbers: DECIMAL, FLOAT, and DOUBLE
- Strings: CHAR, VARCHAR, BINARY, VARBINARY, BLOB, TEXT, ENUM, and SET
- Date and Time: DATE, TIME, DATETIME, TIMESTAMP, and YEAR



*Every relational database has its own **maximum and minimum size limit** for different data types, you don't need to remember the limit. Idea is to have the knowledge of what data type to be used in a specific scenario*

Numeric - Data Types

Datatype	From	To
Bit	0	1
Tinyint	0	255
Smallint	-32768	32767
mediumint	-8388608	8388607
int	-2,147,483,648	2,147,483,647
bigint	-2^{63}	$2^{63}-1$

Floating Numbers

Datatype	Specification
Decimal	Max digits is 65
Float	A precision from 0 to 23 results in a 4-byte single-precision FLOAT column. A precision from 24 to 53 results in an 8-byte double-precision DOUBLE column.

Strings

Datatype	Description
CHAR	The length can be any value from 0 to 255
VARCHAR	The length can be specified as a value from 0 to 65,535
BINARY	Fixed length with maximum length of 8,000 bytes
VARBINARY	Variable length storage with maximum length of 8,000
BLOB	For binary large objects
TEXT	Variable length storage with maximum size of 1GB data

Date and Time

Datatype	Description
DATE	Stores date in the format YYYY-MM-DD
TIME	Stores time in the format HH:MI:SS
DATETIME	Stores date and time information in the format YYYY-MM-DD HH:MI:SS
TIMESTAMP	Stores number of seconds passed since the Unix epoch ('1970-01-01 00:00:00' UTC)
YEAR	Stores year in 2 digit or 4 digit format. Range 1901 to 2155 in 4-digit format. Range 70 to 69, representing 1970 to 2069.



did you know?

MySQL supports various data types. Refer to the MySQL documentation for more details

<https://dev.mysql.com/doc/refman/8.0/en/data-types.html>

CREATE TABLE - Syntax

- The CREATE TABLE statement is used to create a new table in a database

Syntax:

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

- The column parameters specify the names of the columns of the table
- The datatype parameter specifies the type of data the column can hold (e.g. varchar, integer, date, etc.).

CREATE TABLE - Example

- We will create a customers table, which will hold the customers information
- The table will have the following columns in it
 - CustomerID
 - FirstName
 - LastName and

CREATE TABLE - Example

- We will create a table "customers" in the company database. Select the database 'company' using **USE company** command

```
CREATE TABLE customers (  
  CustomerId int,  
  first_name varchar(20),  
  last_name varchar(20),  
  country varchar(20)  
);
```

It is declared as an integer since it contains only integers

`first_name`, `last_name`, and `country`: They contain strings, so they are defined as `varchar`

CREATE TABLE - Example

- The CustomerID column is of type int and will hold an integer
- 'firstName', 'lastName', and 'country' columns are of type varchar with maximum length of 20 characters
- Check the definition of the table by using: describe *customers* :

Field	Type	Null	Key	Default	Extra
CustomerId	int(11)	YES		NULL	
first_name	varchar(20)	YES		NULL	
last_name	varchar(20)	YES		NULL	
country	varchar(20)	YES		NULL	



Drop Table

DROP TABLE - Syntax

- The DROP TABLE statement is used to drop an existing table in a database

Syntax:

```
DROP TABLE table_name;
```



Be careful before dropping a table. Deleting a table will result in loss of complete information stored in the table!!!!

DROP and TRUNCATE TABLE - Example

- The following SQL statement drops the existing table "company":

```
DROP TABLE customers;
```

- The TRUNCATE TABLE statement is used to delete the data inside a table, but not the table itself

```
TRUNCATE TABLE customers;
```



Data Manipulation Language (DML)

Data Manipulation Language (DML)

- As a data analyst, the majority of your work will focus on insight generation, and you will be working with DML commands
- The typical commands available in DML are:
 - INSERT
 - UPDATE
 - DELETE
 - SELECT



INSERT

SQL INSERT - Syntax

- The INSERT INTO statement is used to insert new records in a table
- It is possible to write the INSERT INTO statement in two ways
- The first way specifies both the column names and the values to be inserted

Syntax: **INSERT INTO** *table_name* (*column1*, *column2*, *column3*,
...) **VALUES** (*value1*, *value2*, *value3*, ...);

SQL INSERT - Syntax

- Another way to insert values can be in the following manner where we do not use the column names:

Syntax:

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

- Make sure the order of the values is in the same order as the columns in the table

SQL INSERT - Example

- Here we insert values into the customers table created earlier
- The INSERT statement is used to create new records in a table

```
INSERT INTO customers (CustomerId,first_name,last_name,country)  
VALUES  
(1,'Mike', 'Christensen', 'USA'),  
(2, 'Andy', 'Hollands', 'Australia'),  
(3, 'Ravi', 'Vedantam', 'India');
```

INSERT - Example

- The "customer" table will now look like this:

CustomerId	first_name	last_name	country
1	Mike	Christensen	USA
2	Andy	Hollands	Australia
3	Ravi	Vedantam	India



UPDATE

SQL UPDATE - Syntax

- The UPDATE statement is used to modify the existing records in a table

Syntax: **UPDATE** *table_name*

SET *column1 = value1, column2 = value2, ...*

WHERE *condition;*



Be careful when updating records in a table! Notice the WHERE clause in the UPDATE statement. The WHERE clause specifies which record(s) that should be updated. If you omit the WHERE clause, all records in the table will be updated!

SQL UPDATE - Example

- We are updating the first row of the customers table
- The first_name and the last_name will be updated to John, Kent from the country 'USA'

```
UPDATE customers  
SET first_name = 'John', last_name= 'Kent'  
WHERE country = 'USA';
```

UPDATE - Example

- The "customer" table will now look like this:

CustomerId	first_name	last_name	country
1	John	Kent	USA
2	Andy	Hollands	Australia
3	Ravi	Vedantam	India



DELETE

SQL DELETE - Syntax

- The DELETE statement is used to delete existing records in a table

Syntax:

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



Be careful when deleting records in a table! Notice the WHERE clause in the DELETE statement. The WHERE clause specifies which record(s) should be deleted. If you omit the WHERE clause, all records in the table will be deleted!

SQL DELETE - Example

- We are deleting the first row where, first name is John

```
DELETE FROM customers WHERE first_name='John';
```

- The "customer" table will now look like this:

CustomerId	first_name	last_name	country
2	Andy	Hollands	Australia
3	Ravi	Vedantam	India



Data Query Language (DQL)

SELECT Statement - Syntax

- The SELECT statement is used to retrieve data from a table
- The data returned is stored in a result table, called the result-set

Syntax:

```
SELECT column1, column2, ... FROM table_name;
```

SELECT Statement - Syntax

- column1, column2, ... are the field names of the table you want to select data from
- If you want to select all the fields available in the table, use the following syntax

```
SELECT * FROM table_name;
```

SELECT Statement - Data

Below is a selection from the "Customers" table in the "company" database:

CustomerID	FirstName	LastName	country
1	Mike	Christensen	USA
2	Andy	Hollands	Australia
3	Rahul	Vedantam	India
4	Jeevan	Sharma	India

SELECT Column - Example

- The following SQL statement selects the "first_name" and "country" columns from the "Customers" table:

```
SELECT first_name, Country FROM Customers;
```

Output:

first_name	Country
Mike	USA
Andy	Australia
Ravi	India
Jeevan	India

Selecting all columns - Example

- The following SQL statement selects all the columns from the "Customers" table:

```
SELECT * FROM Customers;
```

Output:

CustomerId	first_name	last_name	country
1	Mike	Christensen	USA
2	Andy	Hollands	Australia
3	Ravi	Vedantam	India
4	Jeevan	Sharma	India

WHERE Clause- Syntax

- The WHERE clause is used to filter records
- The WHERE clause is used to extract only those records that fulfill a specified condition

Syntax:

```
SELECT column1, column2, ... FROM table_name WHERE condition;
```



The WHERE clause is not only used in SELECT statement, it is also used in UPDATE, DELETE statement, etc.!

WHERE Clause- Example

- The following SQL statement selects all the customers from the country "India", in the "Customers" table:

```
SELECT * FROM Customers WHERE Country='India';
```

Output:

CustomerId	first_name	last_name	country
3	Ravi	Vedantam	India
4	Jeevan	Sharma	India

Operators in the WHERE Clause

- The following operators can be used in WHERE clause

Operator	Description
=	Equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal

Operators in the WHERE Clause

Operator	Description
<>	Not equal. In some versions of SQL the operator may be written as !=
BETWEEN	Between a certain range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column



did you know?

By default, The SQL Keywords are case-insensitive (SELECT, FROM, WHERE, AS, etc.) but are usually written in all capitals. However, MySQL has a configuration option to enable or disable it



Compound Search Conditions

Compound Search Conditions

- The compound conditions are made up of multiple simple conditions connected by AND or OR
- There is no limit to the number of simple conditions that can be present in a single query
- They enable you to specify compound search conditions to fine tune your data retrieval requirements

Compound Search Conditions

- Use following table for further operations

distributor_id	distributor_name	city	state
1000	aakash_sales	pune	maharashtra
2000	priya_marketing	nagpur	maharashtra
3000	ramesh_sales	chandigadh	punjab
4000	bansal_electrics	indore	MP
5000	deshapndes	patna	bihar
6000	das_sales	lucknow	UP
7000	Vijay_sales	rajkot	gujrat

Compound Search Conditions

- Use of the "AND" and "OR" Conditions with the SELECT Statement

```
SELECT * FROM employeee.distributor
WHERE (state = 'maharashtra' AND
distributor_id <> 7000)
      OR (distributor_id = 1000);
```

Output:

distributor_id	distributor_name	city	state
1000	aakash_sales	pune	maharashtra
2000	priya_marketing	nagpur	maharashtra

Explanation

- Previous query will return all distributor that are in the state of Maharashtra but do not have a distributor_id equal to 7000
- The query will also return distributor whose distributor_id is equal to 1000

Compound Search Conditions

- Use of AND and OR with UPDATE statement

```
UPDATE employeee.distributor SET state =  
'rajasthan'  
WHERE distributor_id = 6000 OR  
(distributor_id > 5000 AND city <>  
'lucknow');
```

Output:

distributor_id	distributor_name	city	state
1000	aakash_sales	pune	maharashtra
2000	priya_marketing	nagpur	maharashtra
3000	ramesh_sales	chandigadh	punjab
4000	bansal_electrics	indore	MP
5000	deshapndes	patna	bihar
6000	das_sales	lucknow	rajasthan
7000	Viiav sales	raikot	raiasthan



Missing Data

Checking Missing Data

- The SQL NULL is the term used to represent a missing value
- A NULL value in a table is a value in a field that appears to be blank.
- You must use the IS NULL or IS NOT NULL operators to check for a NULL

Checking Missing Data

Consider the following Employee table having the records as shown below

ID	NAME	AGE	ADDRESS	SALARY
1	Kellie	32	California	2000
2	Pete	25	Texas	1500
3	Popy	23	Boston	2000
4	Sam	25	Florida	
5	Jhon	27	Hawaii	

Is Not Null Operator

Syntax:

```
SELECT  ID, NAME, AGE, ADDRESS, SALARY FROM Employee  
WHERE SALARY IS NOT NULL;
```

Output:

ID	NAME	AGE	ADDRESS	SALARY
1	Kellie	32	California	2000
2	Pete	25	Texas	1500
3	Popy	23	Boston	2000

Is Null Operator

Syntax:

```
SELECT  ID, NAME, AGE, ADDRESS, SALARY FROM Employee  
WHERE SALARY IS NULL;
```

Output:

ID	NAME	AGE	ADDRESS	SALARY
4	Sam	25	Florida	
5	Jhon	27	Hawaii	



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

Syntax:

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

The Alter Query - Change Clause

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;
```



Modify clause CANNOT be used to rename a column

The Alter Query - Change Clause

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

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

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