


Databases and SQL for Data Science with Python

30/06/2024

Why SQL is Essential for Data Scientists?

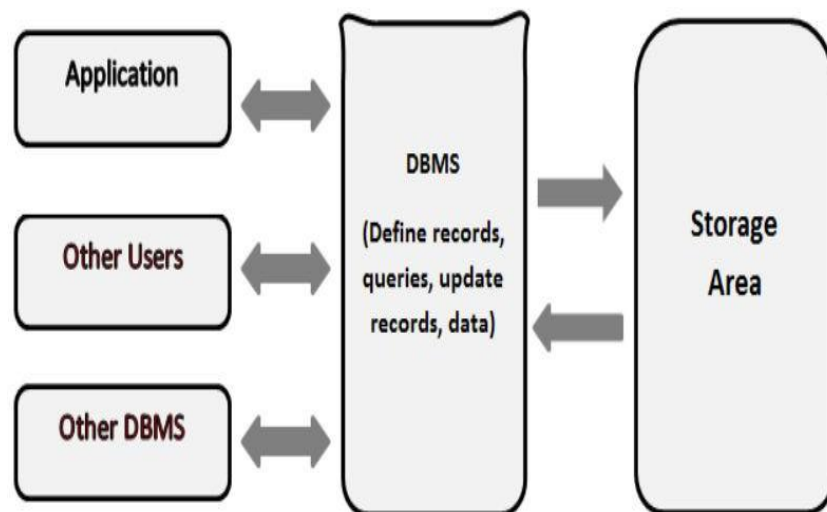
SQL = The Key to Access and Prepare Your Data 

- **Data Lives in Databases**
Most real-world data is stored in relational databases (MySQL, PostgreSQL, SQL Server).
- **SQL is the Language to Talk to Data**
Retrieve only the data you need using filtering, sorting, grouping, and joining.
- **Prepare Data at the Source**
Clean, filter, and merge data before importing it into Python, R, or BI tools.
- **Understand Data Structure**
Learn table relationships, data types, and data quality directly from the database.
- **Highly Demanded Skill**
Most Data Science job descriptions list SQL as a core requirement.

"Before you analyze data, you must know how to access it." – IBM Data Science Approach

What are Databases?

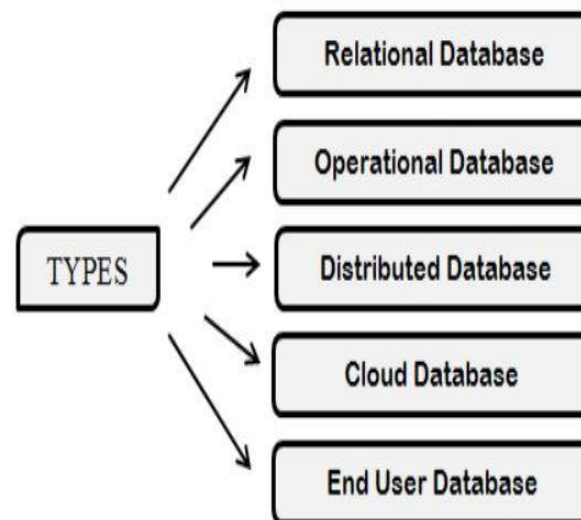
- A database is an organized collection of data, generally stored and accessed electronically from a computer system. It supports the storage and manipulation of data.
- In other words, databases are used by an organization as a method of storing, managing and retrieving information.



Types of Databases

Depending upon the usage requirements, there are following types of databases available in the market:

- Centralized database
 - Distributed database
 - Personal database
 - End-user database
 - Commercial database
 - NoSQL database
 - Operational database
 - Relational database
 - Cloud database
 - Object-oriented database
 - Graph database
- Here is a detailed article on, [Types of Database Management Systems](#).



Advantages of using Databases

There are many advantages of databases

- Reduced data redundancy
- Reduced updating errors and increased consistency
- Greater data integrity and independence from application programs
- Improved data access to users through the use of host and query languages
- Improved data security
- Reduced data entry, storage, and retrieval costs

Disadvantages of using Databases

There are many disadvantages of databases

- Although databases allow businesses to store and access data efficiently, they also have certain disadvantages
- Complexity
- Cost
- Security
- Compatibility

Some examples of Databases

Some of the most popular databases are

- Oracle Database
- Sybase
- MySQL
- IBM db2



What is SQL?

- SQL (Structured Query Language): Is used to perform operations on the records stored in the database, such as updating records, inserting records, deleting records, creating and modifying database tables, views, etc.
- SQL is not a database system, but it is a query language.

Database Transaction

- A transaction is an executing program that forms a logical unit of database actions.
- It includes one or more database access operations such as insert, delete and update.
- The database operations that form a transaction can either be embedded within an application program or they can be specified interactively via a high-level query language such as SQL.

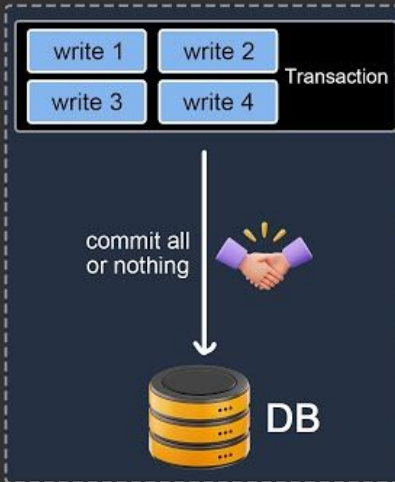
Database Transaction Properties

- Transactions should possess several properties, often called the ACID properties:
 1. Atomicity
 2. Consistency
 3. Isolation
 4. Durability

What does ACID Really Mean ?

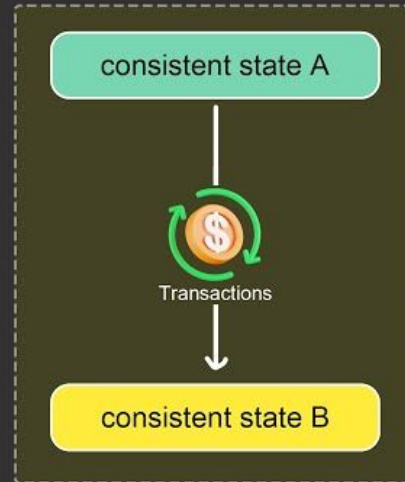
Atomicity

All or nothing



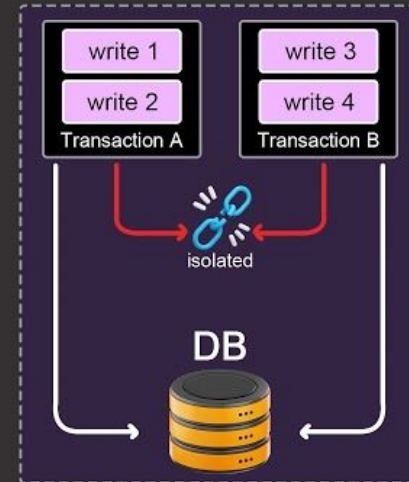
Consistency

Preserving database invariants



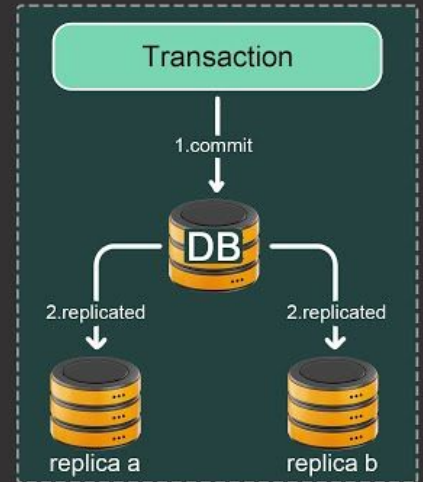
Isolation

Concurrent transactions are isolated from each other



Durability

Data is persisted after transaction is committed even in a system failure



Database Schema

A schema is a group of related objects in a database. There is one owner of a schema who has access to manipulate the structure of any object in the schema. A schema does not represent a person, although the schema is associated with a user that resides in the database.

Database Schema and Database instance

Database Schema

Database schema is the structure of a database, refers to the organization of data as a blueprint that demonstrates how the database is constructed.

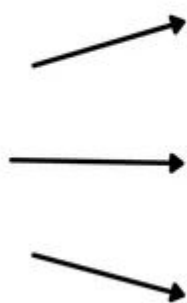
So, a database schema describes how the data may relate to other tables or other data models. However, the schema does not actually contain data.

Database instance

Database instance is a snapshot of data in a database at a single moment in time . It contains all the properties that the schema describes as data values.

Since database instances are just a snapshot at a given moment, they're likely to change over time, unlike database schemas.

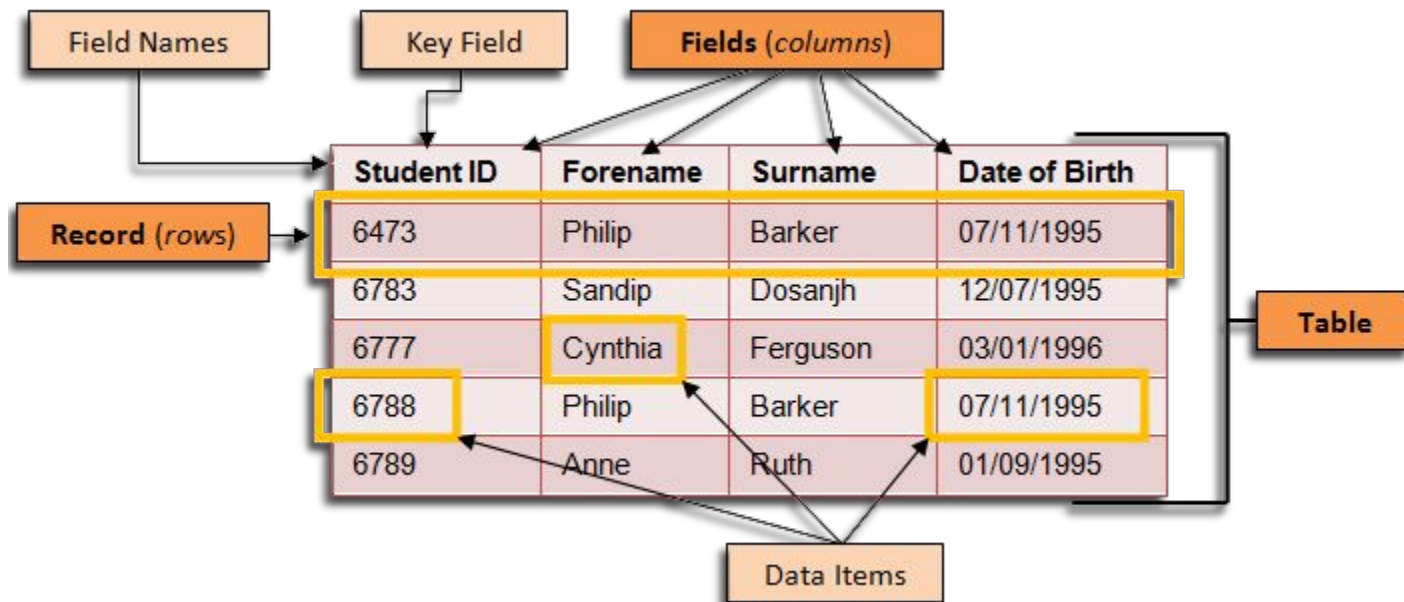
Instances



NAME	ROLL_NO	HOUSE
Akshita	20	Blue
Riya	21	Green
Tanya	22	yellow



Schema



Data types

A data type determines the type of data that can be stored in a database column. The most commonly used data types are:

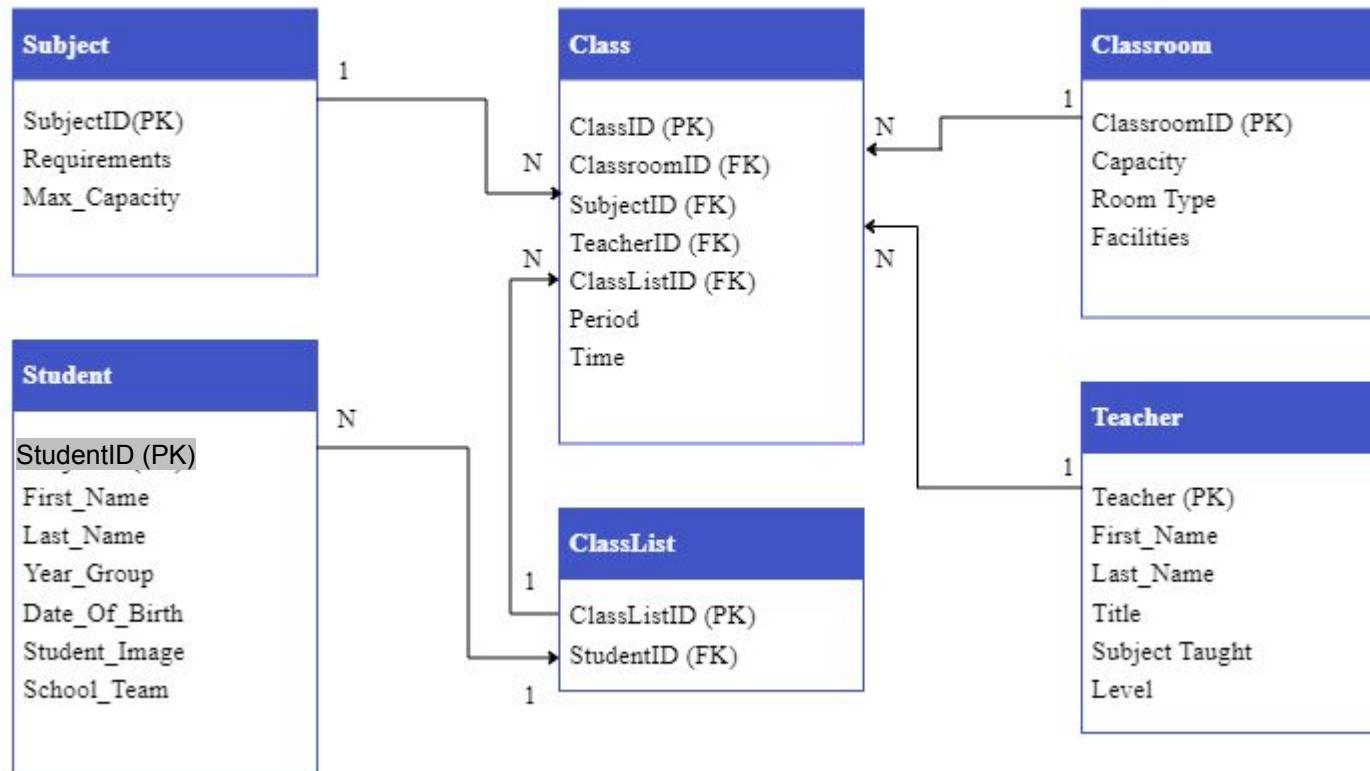
1. Alphanumeric: data types used to store characters, numbers, special characters, or nearly any combination.
2. Numeric
3. Date and Time

MySQL DATA TYPES

DATE TYPE	SPEC	DATA TYPE	SPEC
CHAR	String (0 - 255)	INT	Integer (-2147483648 to 2147483647)
VARCHAR	String (0 - 255)	BIGINT	Integer (-9223372036854775808 to 9223372036854775807)
TINYTEXT	String (0 - 255)	FLOAT	Decimal (precise to 23 digits)
TEXT	String (0 - 65535)	DOUBLE	Decimal (24 to 53 digits)
BLOB	String (0 - 65535)	DECIMAL	"DOUBLE" stored as string
MEDIUMTEXT	String (0 - 16777215)	DATE	YYYY-MM-DD
MEDIUMBLOB	String (0 - 16777215)	DATETIME	YYYY-MM-DD HH:MM:SS
LONGTEXT	String (0 - 4294967295)	TIMESTAMP	YYYYMMDDHHMMSS
LOBLOB	String (0 - 4294967295)	TIME	HH:MM:SS
TINYINT	Integer (-128 to 127)	ENUM	One of preset options
SMALLINT	Integer (-32768 to 32767)	SET	Selection of preset options
MEDIUMINT	Integer (-8388608 to 8388607)	BOOLEAN	TINYINT(1)

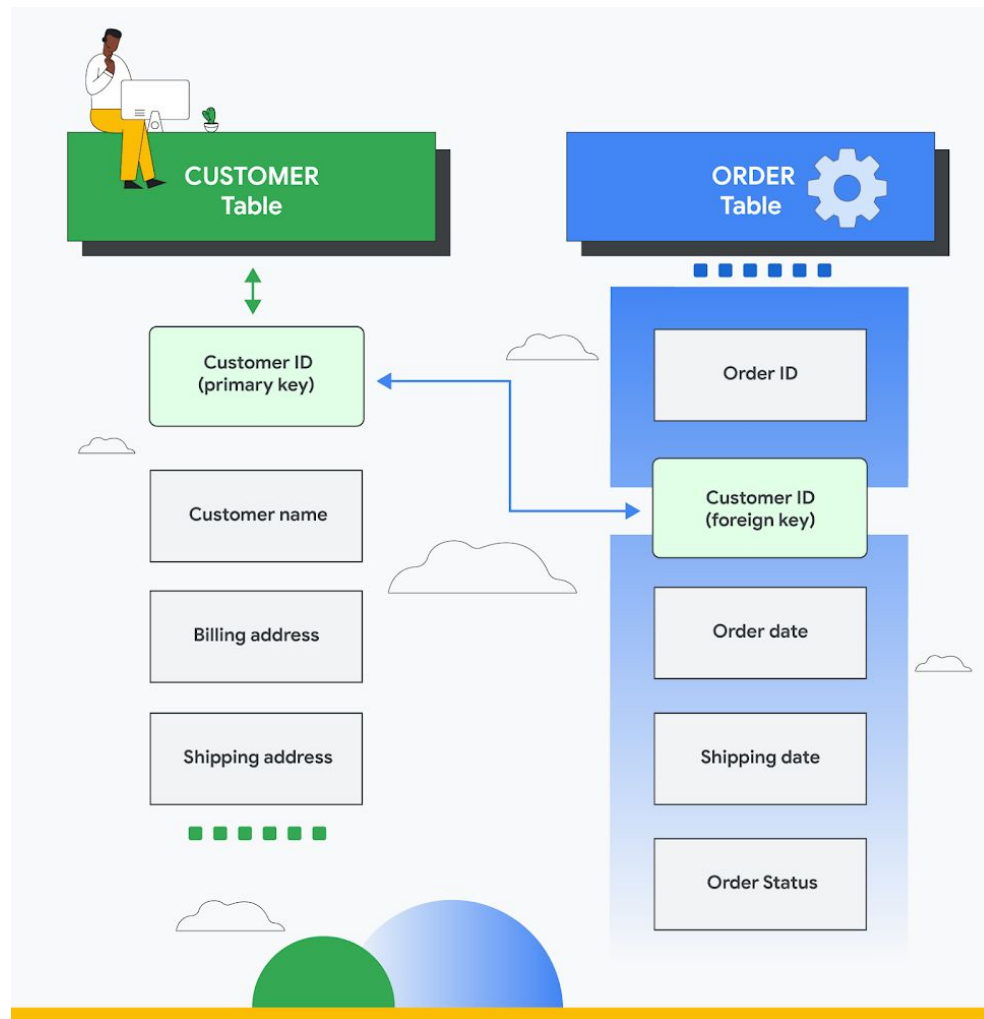
Entity-Relationship Diagram: School Classes

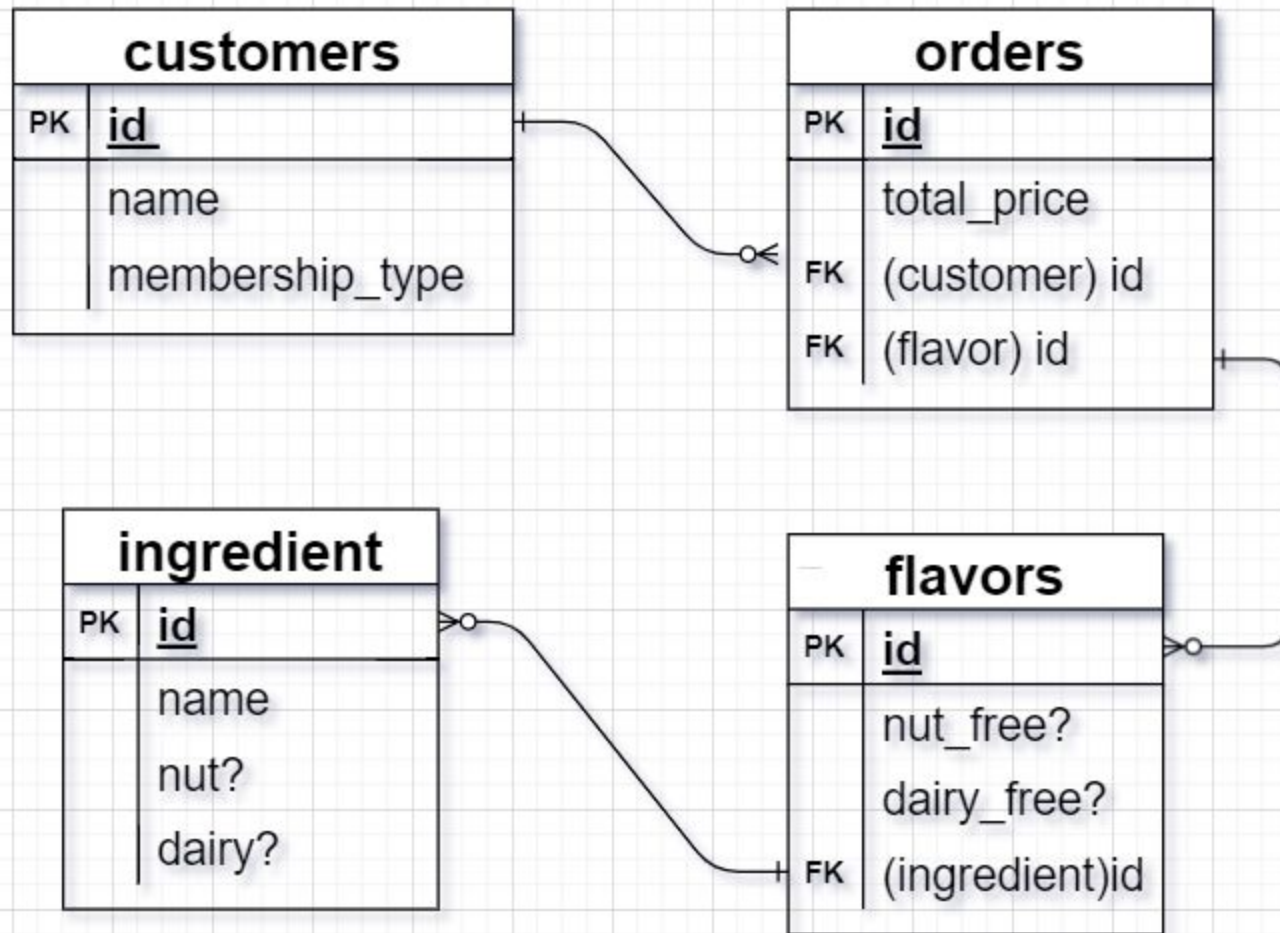
PK : Primary Key
FK : Foreign Key



Database Constraints

- Primary Key (Not Null + Unique)
- Not Null
- Unique Key
- Referential Integrity (Foreign key FK)
- Check





- **Primary key:**

- A field in a table that uniquely identifies each rows in a database table.
- Primary keys must contain unique values.
- A table can have only one Primary keys.

```
--define primary key  
CREATE TABLE students(  
    student id integer PRIMARY KEY,  
    name varchar(20));
```

- **Foreign key:**

- Reference a column in another table to define the relationship between two tables.
- The relationship between 2 tables matches the Primary Key in one of the tables with a Foreign Key in the second table.

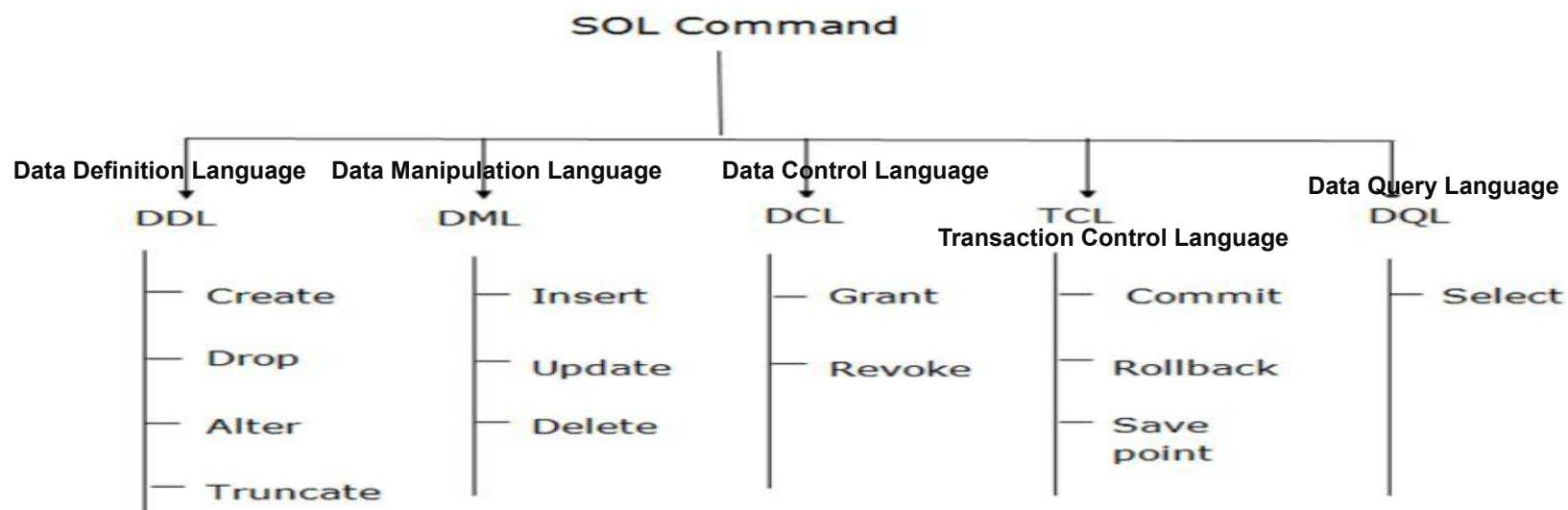
```
--define primary key
CREATE TABLE teachers (
  id integer PRIMARY KEY,
  teache_name varchar(30),
);
```

```
CREATE TABLE classes (
  id integer PRIMARY KEY,
  teacher id varchar(30)
REFERENCES teachers(id),
--foreign key is created
category varchar(40),
);
```

SQL Commands

- SQL commands are instructions. It is used to communicate with the database. It is also used to perform specific tasks, functions, and queries of data.
- SQL can perform various tasks like create a table, add data to tables, drop the table, modify the table, set permission for users.

Types of SQL Commands



Type	Primary Function	Examples
DDL	Defines database structure	CREATE , ALTER , DROP , TRUNCATE
DML	Manipulates data in tables	INSERT , UPDATE , DELETE
DCL	Manages permissions and roles	GRANT , REVOKE
TCL	Controls database transactions	COMMIT , ROLLBACK , SAVEPOINT
DQL	Retrieves data from the database	SELECT

Data Definition Language (DDL)

- DDL changes the structure of the table like creating a table, deleting a table, altering a table, etc.
- All the command of DDL are auto-committed that means it permanently save all the changes in the database.

Here are some commands that come under DDL:

- CREATE
- ALTER
- DROP
- TRUNCATE

Data Definition Language (DDL)

- CREATE It is used to create a new table in the database

```
CREATE TABLE TABLE_NAME (COLUMN_NAME DATATYPES[,....]);
```

Example

```
CREATE TABLE EMPLOYEE(Name VARCHAR2(20), Email VARCHAR2(100), DOB DATE);
```

Data Definition Language (DDL)

- DROP: It is used to delete both the structure and record stored in the table.

```
DROP TABLE table_name;
```

Example

.

```
DROP TABLE EMPLOYEE;
```

Data Definition Language (DDL)

- ALTER: It is used to alter the structure of the database. This change could be either to modify the characteristics of an existing attribute or probably to add a new attribute.

```
ALTER TABLE table_name ADD column_name COLUMN-definition;
```

Example

```
ALTER TABLE STU_DETAILS ADD(ADDRESS VARCHAR2(20));  
ALTER TABLE STU_DETAILS MODIFY (NAME VARCHAR2(20));
```

Data Definition Language (DDL)

- TRUNCATE: It is used to delete all the rows from the table and free the space containing the table.

Example

```
TRUNCATE TABLE table_name;
```

```
TRUNCATE TABLE EMPLOYEE;
```

Data Definition Language (DDL)

- TRUNCATE: It is used to delete all the rows from the table and free the space containing the table.

Example

```
TRUNCATE TABLE table_name;
```

```
TRUNCATE TABLE EMPLOYEE;
```

Data Manipulation Language (DML)

- DML commands are used to modify the database. It is responsible for all form of changes in the database.
- The command of DML is not auto-committed that means it can't permanently save all the changes in the database. They can be rollback.
- Here are some commands that come under DML:
 - INSERT
 - UPDATE
 - DELETE

Data Manipulation Language (DML)

- INSERT: The INSERT statement is a SQL query. It is used to insert data into the row of a table

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

Example:

```
INSERT INTO javatpoint (Author, Subject) VALUES ("Sonoo", "DBMS");
```

Data Manipulation Language (DML)

- UPDATE: This command is used to update or modify the value of a column in the table.

```
UPDATE table_name SET [column_name1= value1,...column_nameN = valueN] [WHERE CONDITION]
```

Example:

.

```
UPDATE students
SET User_Name = 'Sonoo'
WHERE Student_Id = '3'
```

Data Manipulation Language (DML)

- DELETE: It is used to remove one or more row from a table.

```
DELETE FROM table_name [WHERE condition];
```

Example:

```
DELETE FROM javatpoint  
WHERE Author="Sonoo";
```

Data Control Language (DCL)

- DCL commands are used to grant and take back authority from any database user.

Here are some commands that come under DCL:

- Grant
- Revoke

Data Control Language (DCL)

- Grant: It is used to give user access privileges to a database.

Example:

```
GRANT SELECT, UPDATE ON MY_TABLE TO SOME_USER, ANOTHER_USER;
```

Data Control Language (DCL)

- Revoke: It is used to take back permissions from the user.

Example:

```
REVOKE SELECT, UPDATE ON MY_TABLE FROM USER1, USER2;
```

Transaction Control Language(TCL)

- TCL commands can only use with DML commands like INSERT, DELETE and UPDATE only.
- These operations are automatically committed in the database that's why they cannot be used while creating tables or dropping them.

Here are some commands that come under TCL:

- COMMIT
- ROLLBACK
- SAVEPOINT

Transaction Control Language(TCL)

- Commit: Commit command is used to save all the transactions to the database.

```
COMMIT;
```

Example

```
DELETE FROM CUSTOMERS  
WHERE AGE = 25;  
COMMIT;
```


Transaction Control Language(TCL)

- Rollback: Rollback command is used to undo transactions that have not already been saved to the database.

```
ROLLBACK;
```

Example

```
DELETE FROM CUSTOMERS  
WHERE AGE = 25;  
ROLLBACK;
```

Transaction Control Language(TCL)

- **SAVEPOINT:** It is used to roll the transaction back to a certain point without rolling back the entire transaction.

Example

```
SAVEPOINT SAVEPOINT_NAME;
```

Relational Database Concepts

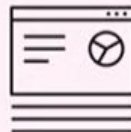
What You Will Learn



Explain the advantage of the relational model

-- 0
-- 1
-- 1
-- 0

Explain how the Entity name and attributes map to a relational database table



Describe the difference between an entity and an attribute



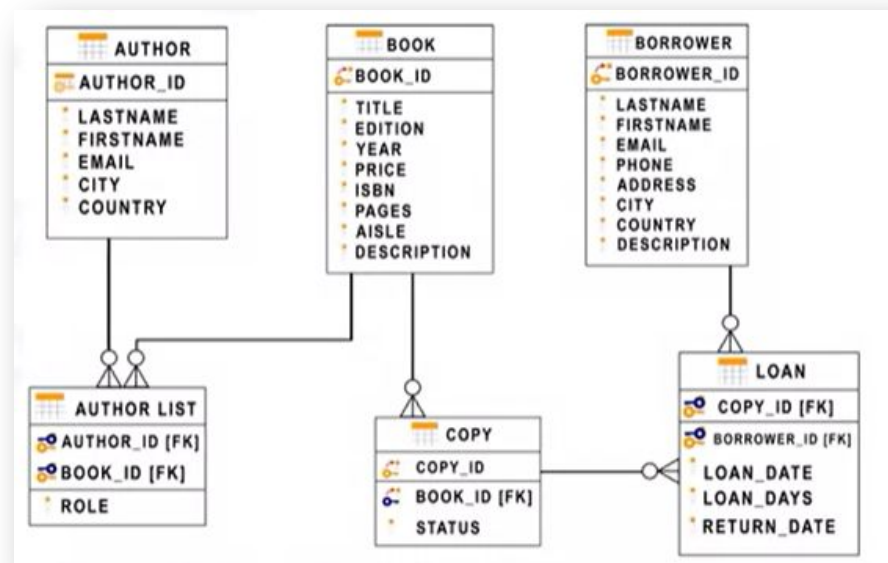
Identify some commonly used data types



Describe the function of Primary Keys

Relational Model

- Most used Data Model
- Allows for data independence
- Data is stored in tables



Logical data independence – Physical data independence – Physical storage independence

MySQL VS PostgreSQL VS SQL Server

Core Idea

SQL basics are the same across systems

- Core CRUD commands (**SELECT**, **INSERT**, **UPDATE**, **DELETE**) work almost the same in MySQL, PostgreSQL, and SQL Server.
- Learning SQL in one system makes it easy to move to another.

Strengths of Each System

- **MySQL**: Simple, widely used, great for learning and small-to-medium projects.
- **PostgreSQL**: Rich advanced features (JSONB, Arrays, GIS), strong for complex data analysis.
- **SQL Server**: Strong integration with Microsoft tools, advanced stored procedures, business analytics support.

SQL is one language, but each system has its own accent.

Master the basics first—then adapting to another system is just learning a few new words.

Differences Are in the Details and if u need

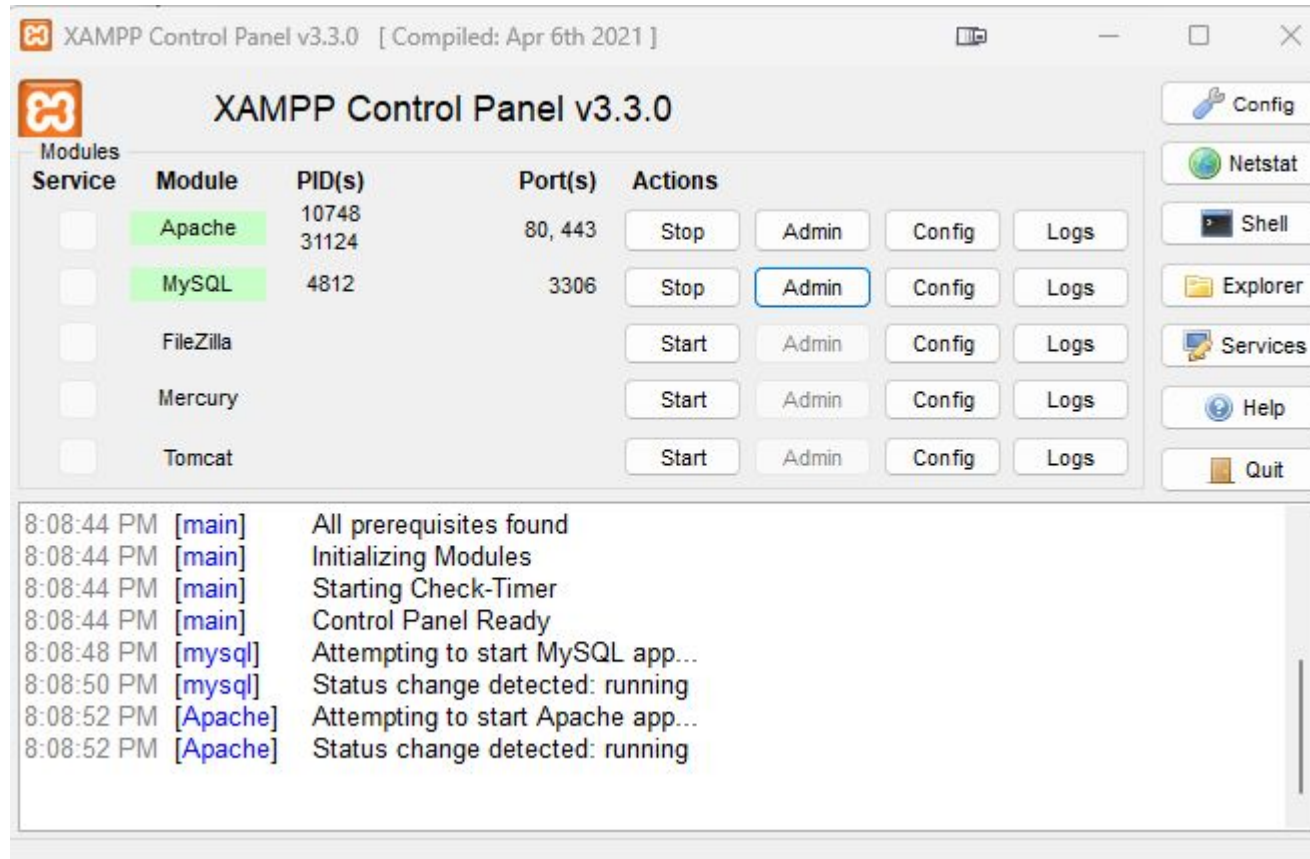
Main differences appear in:

- **Built-in Functions:**
 - Text length → `LENGTH` (MySQL), `CHAR_LENGTH` (PostgreSQL), `LEN` (SQL Server)
 - Dates → `NOW()` (MySQL/PostgreSQL), `GETDATE()` (SQL Server)
- **Pagination:**
 - `LIMIT` in MySQL/PostgreSQL
 - `TOP` or `OFFSET...FETCH` in SQL Server
- **Data Types:** Boolean, JSON, Arrays
- **Auto Increment IDs:**
 - MySQL → `AUTO_INCREMENT`
 - PostgreSQL → `SERIAL` or `GENERATED AS IDENTITY`
 - SQL Server → `IDENTITY(1,1)`
- **Advanced Features:** Full-Text Search, Window Functions, Upsert.

Task XAMPP & MySQL

1. Install XAMPP
 - a. <https://www.apachefriends.org/download.html>
2. Run Apache and MySQL
3. Create Database (usersDB)
 - a. Create Table Users (ID ,name, email)
 - b. Insert 3 users (You, Baraa, Ali)
 - c. View all users
 - d. View name column only

- **Run Apache and MySQL**



view-source:htt x | My First HTML P x | view-source:htt x | view-source:htt x | 01-Scraping ... x | Web Scraping To x | localhost / 127.0 x +

localhost/phpmyadmin/

AMIT Folder | Home | Microsoft 365 | WhatsApp | Machine-Learning-... | Technical team | Mail - Bara Abu Sall... | Jira | amitsoftware / WE -... | phpMyAdmin | Introduction to Cyb...

phpMyAdmin

Recent Favorites

- New
- information_schema
- ini_sql_pertamo_aku
- lec5
- lecture4
- lecture5
- mysql
- performance_schema
- phpmyadmin
- test
- test1111
- testdb
- toko

Server: 127.0.0.1

Databases | SQL | Status | User accounts | Export | Import | Settings | Replication | Variables | More

General settings

Server connection collation: utf8mb4_unicode_ci

[More settings](#)

Appearance settings

Language: English

Theme: pmahomme [View all](#)

Database server

- Server: 127.0.0.1 via TCP/IP
- Server type: MariaDB
- Server connection: SSL is not being used
- Server version: 10.4.32-MariaDB - mariadb.org binary distribut
- Protocol version: 10
- User: root@localhost
- Server charset: UTF-8 Unicode (utf8mb4)

Web server

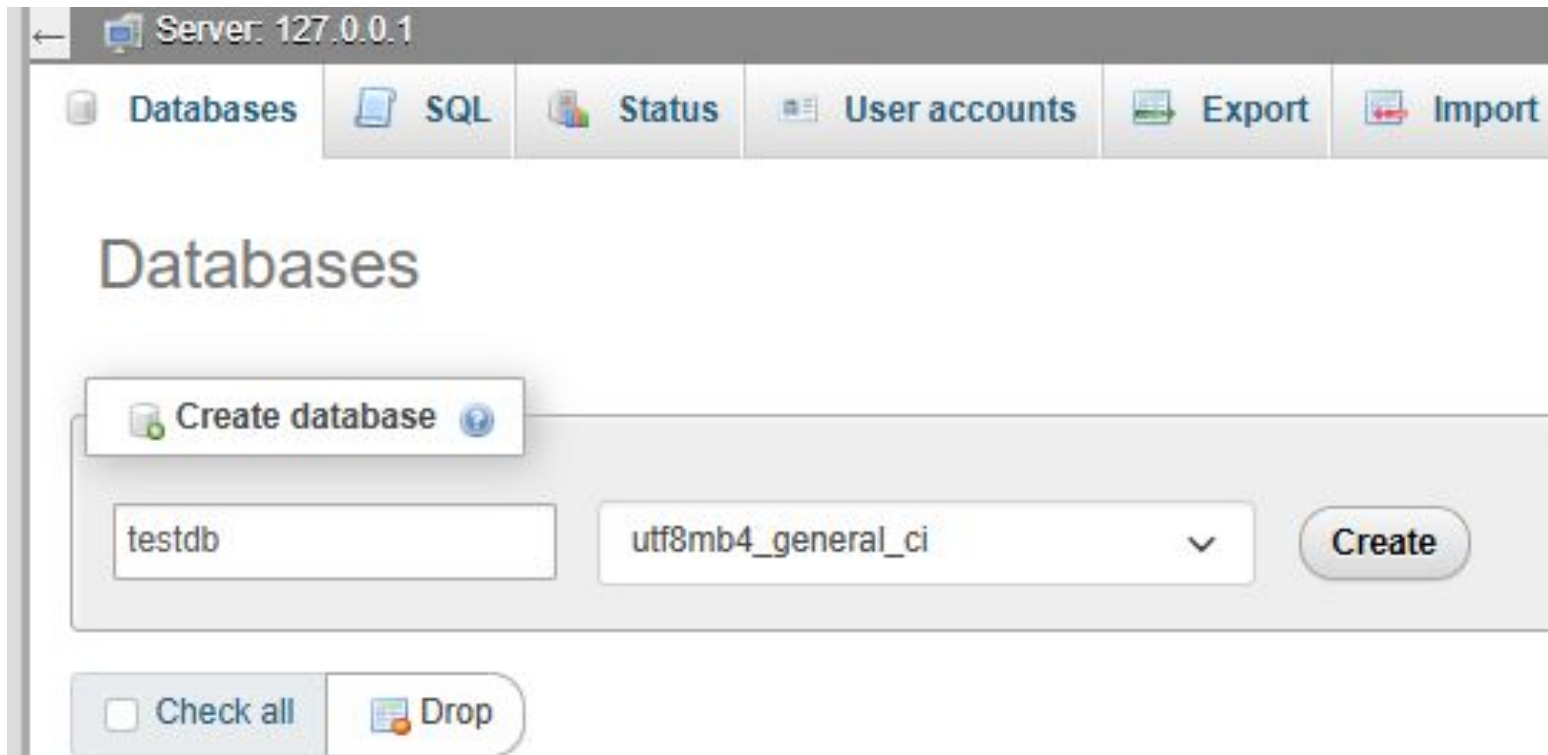
- Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.0.30
- Database client version: libmysql - mysqlnd 8.0.30
- PHP extension: mysqli curl mbstring
- PHP version: 8.0.30

phpMyAdmin

- Version information: 5.2.1 (up to date)

Console

Create Database (testDB)



The screenshot shows a web-based database management interface. At the top, a header bar displays 'Server: 127.0.0.1'. Below this is a navigation menu with buttons for 'Databases', 'SQL', 'Status', 'User accounts', 'Export', and 'Import'. The 'Databases' button is selected. The main content area is titled 'Databases'. A 'Create database' button is visible. Below it, a form for creating a new database is shown. The form has two input fields: the first contains 'testdb' and the second contains 'utf8mb4_general_ci' with a dropdown arrow. To the right of these fields is a 'Create' button. At the bottom of the form, there are two buttons: 'Check all' with an unchecked checkbox and 'Drop' with a trash icon.

Server: 127.0.0.1

Databases SQL Status User accounts Export Import

Databases

Create database ?

testdb utf8mb4_general_ci Create

☐ Check all Drop

Create Table Users (name, email)

```
1 CREATE TABLE Users (  
2     ID INT AUTO_INCREMENT PRIMARY KEY,  
3     Name VARCHAR(50),  
4     Email VARCHAR(100)  
5 );  
6 |
```

Insert 3 users (You, Baraa, Ali)

```
1 INSERT INTO Users (Name, Email)  
2 VALUES ('Ali', 'ali@example.com'),  
3         ('Sara', 'sara@example.com'),  
4         ('John', 'john@example.com');
```

View all users









✓ Showing rows 0 - 2 (3 total, Query took 0.0004 seconds.)

```
SELECT * FROM Users;
```


☐ Profiling [[Edit inline](#)] [[Edit](#)] [[Explain SQL](#)] [[Create PHP code](#)] [[Refresh](#)]

☐ Show all | Number of rows: 25 ▼ Filter rows:

Extra options

				ID	Name	Email
<input type="checkbox"/>	 Edit	 Copy	 Delete	1	Ali	ali@example.com
<input type="checkbox"/>	 Edit	 Copy	 Delete	2	Sara	sara@example.com
<input type="checkbox"/>	 Edit	 Copy	 Delete	3	John	john@example.com

View ID column only











 Showing rows 0 - 2 (3 total, Query took 0.0003 seconds.)

```
SELECT ID FROM Users;
```

☐ Profiling [[Edit inline](#)] [[Edit](#)] [[Explain SQL](#)] [[Create PHP code](#)]

☐ Show all | Number of rows: Filter rows:

Extra options

				ID
<input type="checkbox"/>	 Edit	 Copy	 Delete	1
<input type="checkbox"/>	 Edit	 Copy	 Delete	2
<input type="checkbox"/>	 Edit	 Copy	 Delete	3

Data Definition Language (DDL)

- CREATE command
- ALTER command
- DROP command
- TRUNCATE command

CREATE Command

- Syntax

```
CREATE TABLE table_name
(column1 DATA_TYPE [CONS_TYPE CONS_NAME],
column2 DATA_TYPE [CONS_TYPE CONS_NAME],... )
```

- Example

```
CREATE TABLE Students
```

```
(ID NUMBER(15) PRIMARY KEY, First_Name CHAR(50) NOT
NULL, Last_Name CHAR(50), Address CHAR(50), City
CHAR(50), Country CHAR(25), Birth_Date DATE);
```

EMPLOYEE

FNAME	MINIT	LNAME	<u>SSN</u>	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
-------	-------	-------	------------	-------	---------	-----	--------	----------	-----

DEPARTMENT

DNAME	<u>DNUMBER</u>	MGRSSN	MGRSTARTDATE
-------	----------------	--------	--------------

DEPT_LOCATIONS

<u>DNUMBER</u>	<u>DLOCATION</u>
----------------	------------------

PROJECT

PNAME	<u>PNUMBER</u>	PLOCATION	DNUM
-------	----------------	-----------	------

WORKS_ON

<u>ESSN</u>	<u>PNO</u>	HOURS
-------------	------------	-------

DEPENDENT

<u>ESSN</u>	<u>DEPENDENT_NAME</u>	SEX	BDATE	RELATIONSHIP
-------------	-----------------------	-----	-------	--------------

Figure 7.5 Schema diagram for the COMPANY relational database schema; the primary keys are underlined.

DROP Command

- Syntax

DROP TABLE table_name

- Example

DROP TABLE Students

ALTER Command

- ALTER TABLE statement is used to add or drop columns in an existing table.
- Syntax
 - ALTER TABLE table_name ADD column_name datatype
 - ALTER TABLE table_name DROP COLUMN column_name

ALTER Example

LastName	FirstName	Address
Pettersen	Kari	Storgt 20

To add a column named "City" in the "Students" table:

ALTER TABLE Students ADD City varchar(30)

Result

:

LastName	FirstName	Address	City
Pettersen	Kari	Storgt 20	

Data Manipulation Language (DML)

- INSERT Command
- UPDATE Command
- DELETE Command
- SELECT Command

INSERT Command

- Syntax

```
INSERT INTO table_name (column1, column2, ...)
VALUES (value1, value2, ...)
```

Table *Store_Information*

Column Name	Data Type
store_name	char(50)
Sales	float
Date	datetime

- Example

```
INSERT INTO Store_Information (store_name, Sales, Date)
VALUES ('Los Angeles', 900, 'Jan-10-1999')
```

INSERT Example 2

LastName	FirstName	Address	City
El-Sayed	Mohamed	Nasr City	Cairo

INSERT INTO Students VALUES ('Saleh', 'Ahmed', 'Moharam bak', 'Alex.')

Result

LastName	FirstName	Address	City
El-Sayed	Mohamed	Nasr City	Cairo
Saleh	Ahmed	Moharam bak	Alex.

INSERT Example 3

LastName	FirstName	Address	City
El-Sayed	Mohamed	Nasr City	Cairo
Saleh	Ahmed	Moharam bak	Alex.

INSERT INTO Students (LastName, City) VALUES ('Hassan', 'Assuit')

Result

LastName	FirstName	Address	City
El-Sayed	Mohamed	Nasr City	Cairo
Saleh	Ahmed	Moharam bak	Alex.
Hassan			Assuit

UPDATE Command

- Syntax

UPDATE table_name

SET column_1= new value, column_2= new value
WHERE condition

UPDATE Example

UPDATE Store_Information
SET Sales = 500
WHERE store_name = 'Los Angeles' AND Date = 'Jan-08-1999'

Before

<i>store_name</i>	<i>Sales</i>	<i>Date</i>
<i>Los Angeles</i>	<i>\$1500</i>	<i>Jan-05-1999</i>
<i>San Diego</i>	<i>\$250</i>	<i>Jan-07-1999</i>
<i>Los Angeles</i>	<i>\$300</i>	<i>Jan-08-1999</i>
<i>Boston</i>	<i>\$700</i>	<i>Jan-08-1999</i>

After

<i>store_name</i>	<i>Sales</i>	<i>Date</i>
<i>Los Angeles</i>	<i>\$1500</i>	<i>Jan-05-1999</i>
<i>San Diego</i>	<i>\$250</i>	<i>Jan-07-1999</i>
<i>Los Angeles</i>	<i>\$500</i>	<i>Jan-08-1999</i>
<i>Boston</i>	<i>\$700</i>	<i>Jan-08-1999</i>

UPDATE Example 2

LastName	FirstName	Address	City
El-Sayed	Mohamed	Nasr City	Cairo
Saleh	Ahmed	Moharam bak	Alex.

UPDATE Students SET Address = '241 El-haram', City = 'Giza'
WHERE LastName = 'El-Sayed'

Result

:

LastName	FirstName	Address	City
El-Sayed	Mohamed	241 El-haram	Giza
Saleh	Ahmed	Moharam bak	Alex.

DELETE Command

- Syntax

DELETE FROM table_name

WHERE condition

DELETE Example

DELETE FROM Store_Information
WHERE store_name = 'Los Angeles'

Before

<i>store_name</i>	<i>Sales</i>	<i>Date</i>
<i>Los Angeles</i>	<i>\$1500</i>	<i>Jan-05-1999</i>
<i>San Diego</i>	<i>\$250</i>	<i>Jan-07-1999</i>
<i>Los Angeles</i>	<i>\$300</i>	<i>Jan-08-1999</i>
<i>Boston</i>	<i>\$700</i>	<i>Jan-08-1999</i>

After

<i>store_name</i>	<i>Sales</i>	<i>Date</i>
<i>San Diego</i>	<i>\$250</i>	<i>Jan-07-1999</i>
<i>Boston</i>	<i>\$700</i>	<i>Jan-08-1999</i>

TRUNCATE Vs DELETE

TRUNCATE TABLE is functionally identical to DELETE statement with no WHERE clause

- TRUNCATE TABLE table_name
- TRUNCATE TABLE customer

Simple Queries

- Syntax

```
SELECT    <attribute list >  
FROM      <table list>  
WHERE     <condition>  
ORDER BY  <attribute list >
```

Examples

- `SELECT *`
`FROM departments;`
- `SELECT emp_id, emp_name, dept_id`
`FROM employees;`
- `SELECT dept_id, dept_name`
`FROM departments`
`WHERE location = 'Cairo';`

DISTINCT Keyword

- It's a **row** keyword that displays unique rows
- Example

Employees table

EmpNo	Name	DNo	JobID
100	Ahmed	2	Sales_Rep
200	Mai	2	IT_PROG
300	Ali	2	Sales_Rep
400	Mahmoud	3	Sales_Rep

Output

DNo
2
3

- **SELECT DISTINCT DNo**
FROM employees;

DISTINCT Keyword (cont.)

- Example

Employees table

EmpNo	Name	DNo	JobID
100	Ahmed	2	Sales_Rep
200	Mai	2	IT_PROG
300	Ali	2	Sales_Rep
400	Mahmoud	3	Sales_Rep

- SELECT **DISTINCT** DNo,JobID
FROM employees;

Output

DNo	JobID
2	Sales_Rep
2	IT_PROG
3	Sales_Rep

Questions?