HyperionDev

Lecture – Housekeeping

- ❏ The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly.
  - ❏ Please review Code of Conduct (in Student Undertaking Agreement) if unsure
- ❏ No question is daft or silly - **ask them!**
- ❏ There are Q&A sessions midway and at the end of the session, should you wish to ask any follow-up questions.
- ❏ Should you have any questions after the lecture, please schedule a mentor session.
- ❏ For all non-academic questions, please submit a query: www.hyperiondev.com/support

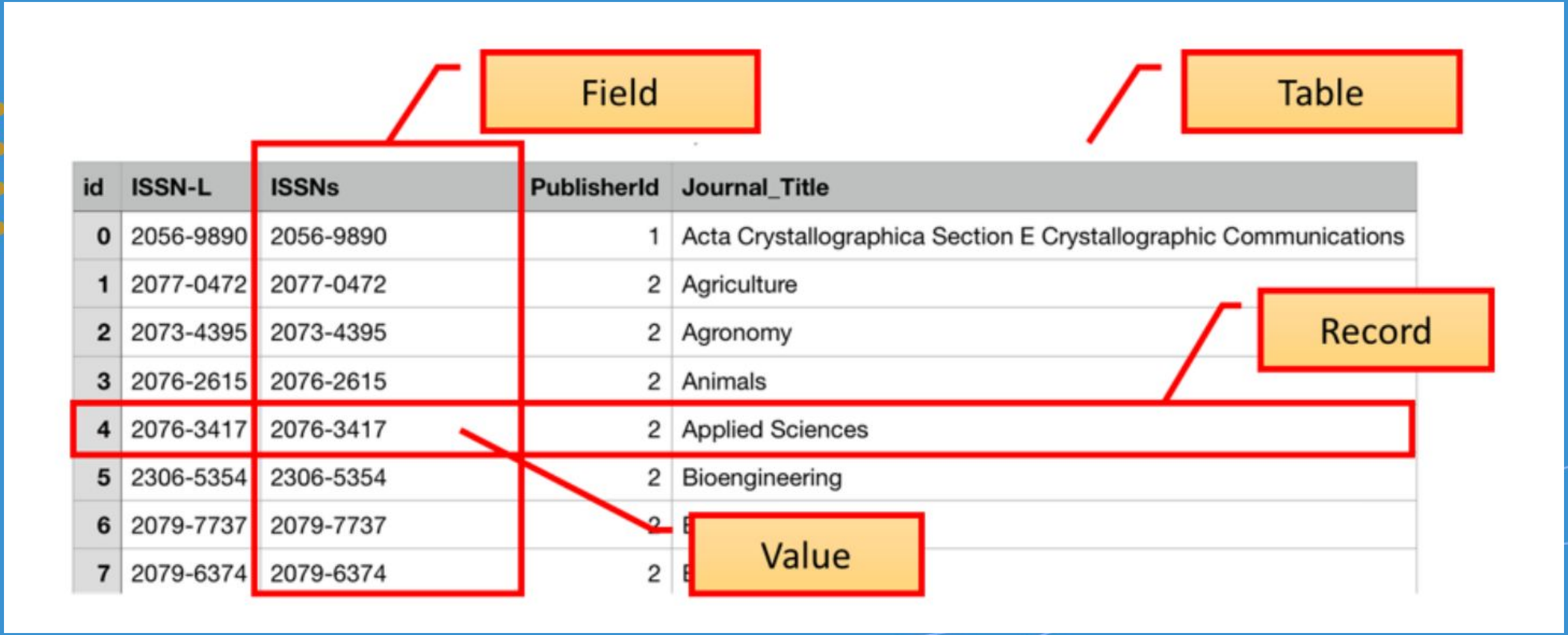
Lecture Objectives

1. Look at how SQL, the language of relational databases, can be used for performing CRUD (create, read, update, delete) operations
2. Dive into set theory
3. Discuss primary and foreign keys
4. An overview of integrating file-based and server-based databases with your application.

What is a Database

- A database is made up of a collection of tables that stores a specific set of structured data
- Base of data: well-organized electronic filing cabinet.
- Shared, integrated computer structures that store end-user data, raw facts, metadata.
- In essence, it is a collection of tables.

Database


Field
Table
Record
Value

id	ISSN-L	ISSNs	PublisherId	Journal_Title
0	2056-9890	2056-9890	1	Acta Crystallographica Section E Crystallographic Communications
1	2077-0472	2077-0472	2	Agriculture
2	2073-4395	2073-4395	2	Agronomy
3	2076-2615	2076-2615	2	Animals
4	2076-3417	2076-3417	2	Applied Sciences
5	2306-5354	2306-5354	2	Bioengineering
6	2079-7737	2079-7737	2	B
7	2079-6374	2079-6374	2	B

Types of Databases

- Users supported at a time:
    - Single-user DB (Desktop files)
    - Multi-user DB (Web applications)
- **Operational** - designed for day-to-day activities for companies.
- **Analytical** - stores historical data and business metrics.
- **Relational** - recognizes relations between objects.

SQL Language

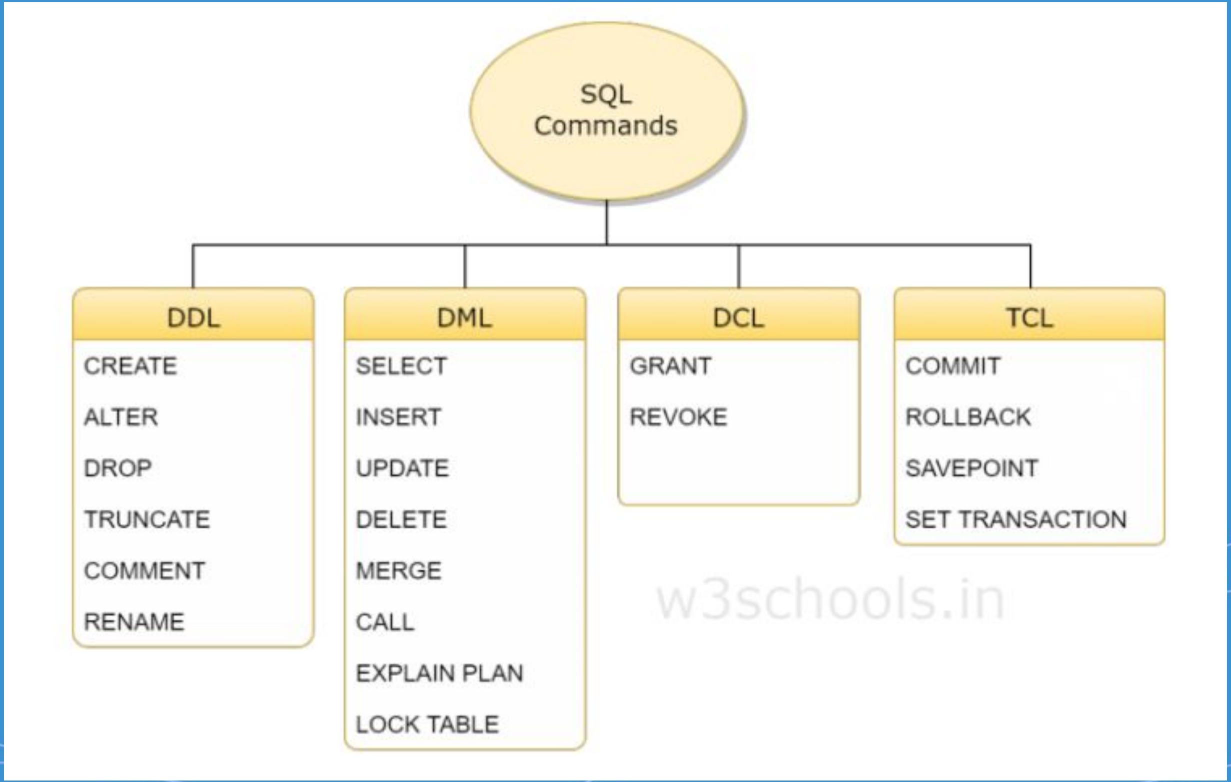
- SQL - Structured Query Language
- It is a means of communicating with a relational database that allows you to define, query, modify, and control the data

SQL Commands

SQL commands are divided into 4 categories:

1. Data Manipulation Language (DML)
2. Data Definition Language (DDL)
3. Data Control Language (DCL)
4. Transaction Control Language (TCL)

SQL Commands



SQL Commands

Common DDL statements include:

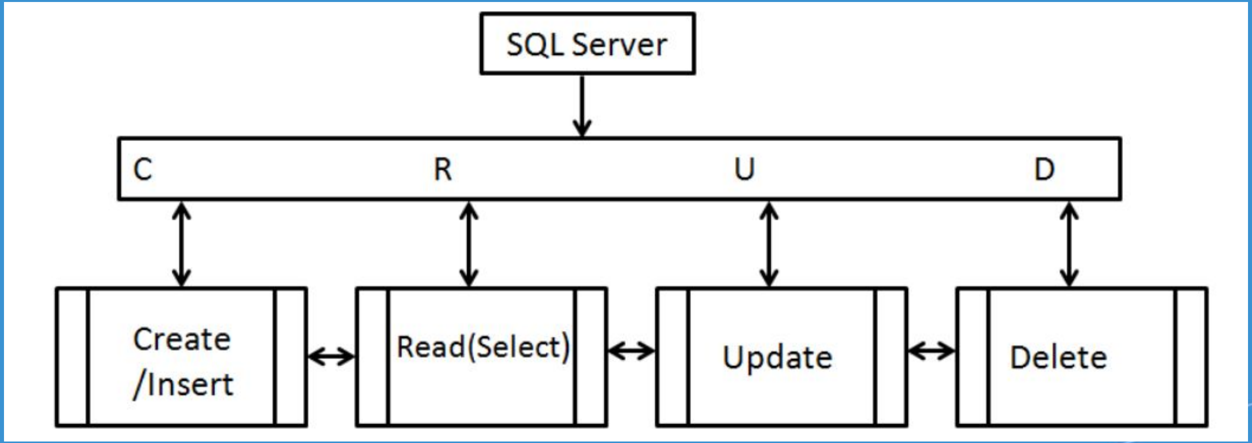
CREATE: generates a new table
ALTER: alters table
DROP: removes a table from the database
RENAME: rename an object
TRUNCATE: remove all records from a table, including all spaces allocated for records are removed

SQL Language

Common DML statements include:

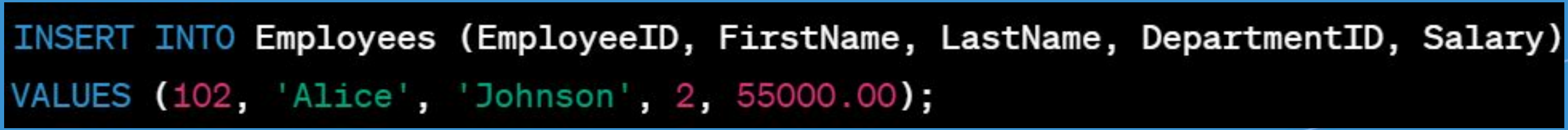
- **SELECT**: Read certain fields from records.
- **INSERT**: Add records to a table.
- **UPDATE**: Change already-existing records.
- **DELETE**: Delete certain records.

CRUD - Create Read Update Delete



Create (INSERT)

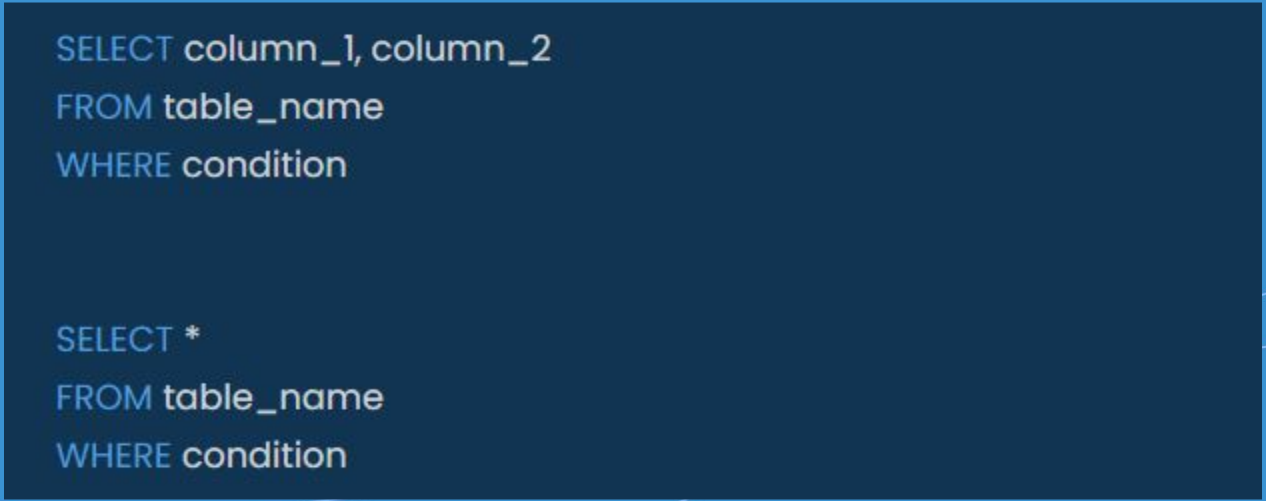
Let’s say we have a new employee, Alice Johnson. We want to insert a record for "Alice Johnson" into the "Employees" table with specific values for columns "EmployeeID," "FirstName," "LastName," "DepartmentID," and "Salary."



```sql
INSERT INTO Employees (EmployeeID, FirstName, LastName, DepartmentID, Salary)
VALUES (102, 'Alice', 'Johnson', 2, 55000.00);
```

Read (SELECT)

We can specify what columns we want to retrieve using SELECT and further specify what exactly we are looking for within those columns by setting the condition.

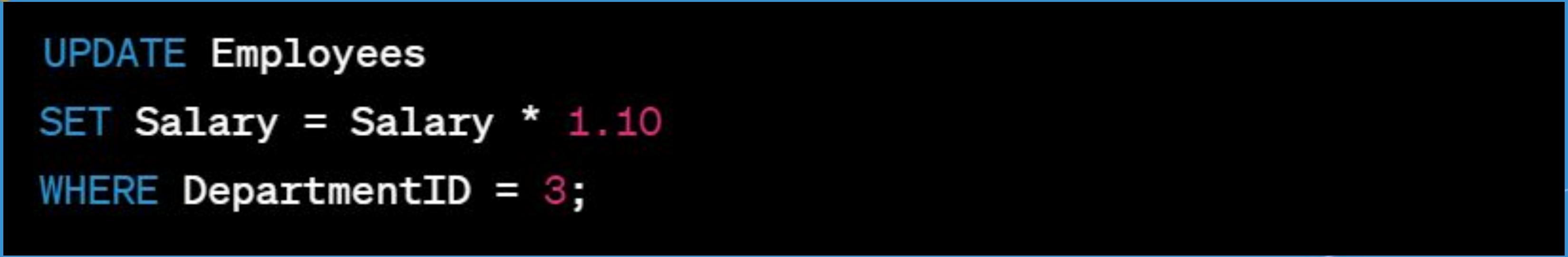


```
SELECT column_1, column_2
FROM table_name
WHERE condition


SELECT *
FROM table_name
WHERE condition
```

Update (UPDATE)

Say you want to increase the salary of all employees in the “Sales” department by 10%. The “Sales” department has a DepartmentID of 3.



```sql
UPDATE Employees
SET Salary = Salary * 1.10
WHERE DepartmentID = 3;
```

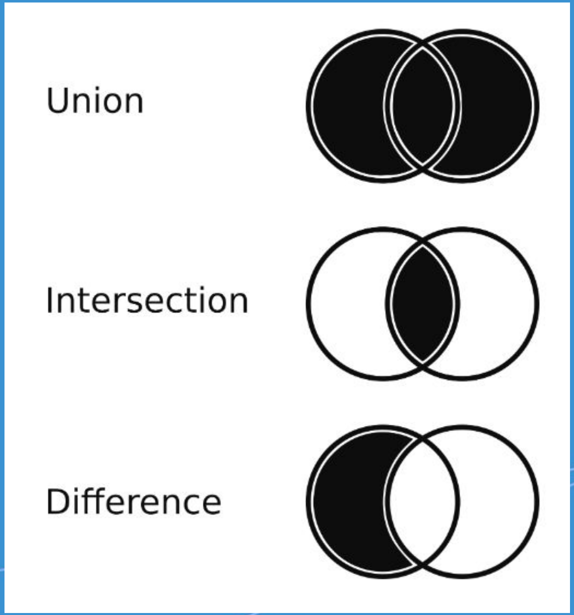
Delete (DELETE)

Say you have a table called "Products" and there is a specific item, identified with "ProductID" of 456, that you no longer stock.

```sql
DELETE FROM Products
WHERE ProductID = 456;
```

Set Theory

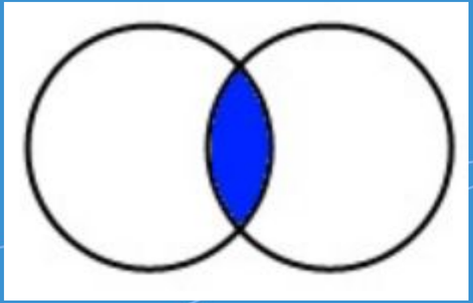
In SQL Server we have 3 important operators at our disposal – UNION (ALL), INTERSECT, and EXCEPT.



INNER JOIN

Let's say you have two tables, "Customers" and "Orders," and you want to retrieve a list of customers who have placed orders.

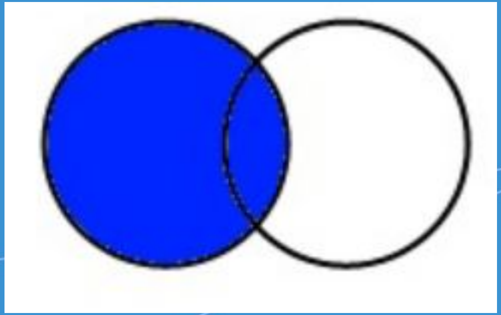
```sql
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
INNER JOIN Orders ON Customers.CustomerID = Orders.CustomerID;
```



LEFT JOIN

Let’s say you have two tables, "Employees" and "Departments," and you want to see a list of all employees along with their department names (including employees without assigned departments).

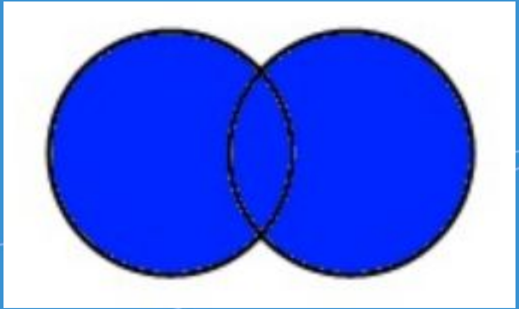
```sql
SELECT Employees.EmployeeName, Departments.DepartmentName
FROM Employees
LEFT JOIN Departments ON Employees.DepartmentID = Departments.DepartmentID;
```



FULL JOIN

Suppose you have two tables, "Students" and "Courses," and you want to list all students and the courses they are enrolled in (including students not enrolled in any courses and courses with no students).

```sql
SELECT Students.StudentName, Courses.CourseName
FROM Students
FULL JOIN Courses ON Students.StudentID = Courses.StudentID;
```

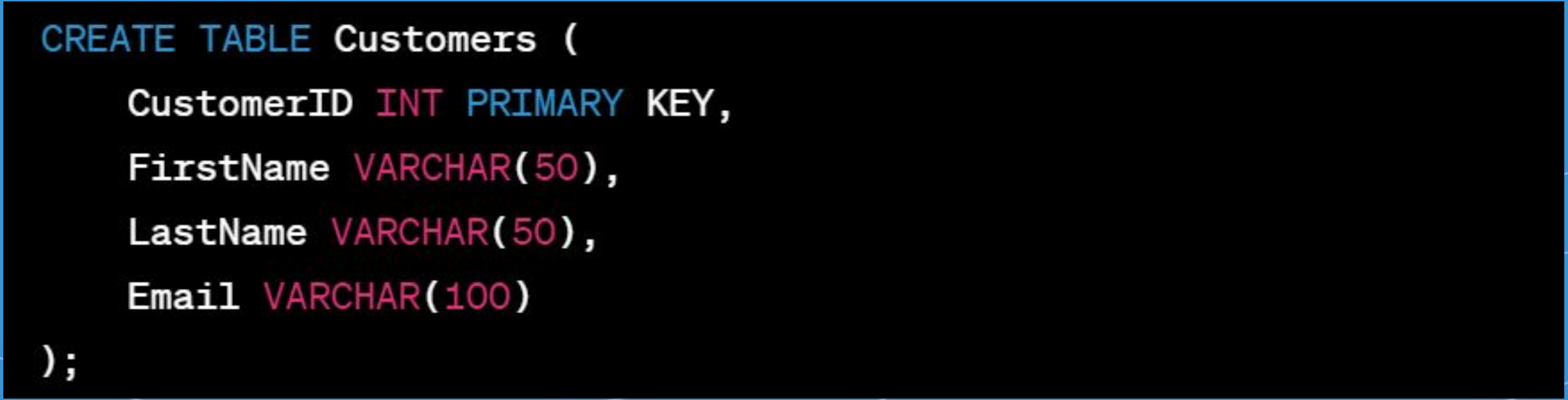


Primary & Foreign keys

- Primary keys serve as unique identifiers for each row in a database table.
- Foreign keys link data in one table to the data in another table.
- A foreign key column in a table points to a column with unique values in another table (often the primary key column) to create a way of cross-referencing the two tables

Primary & Foreign keys

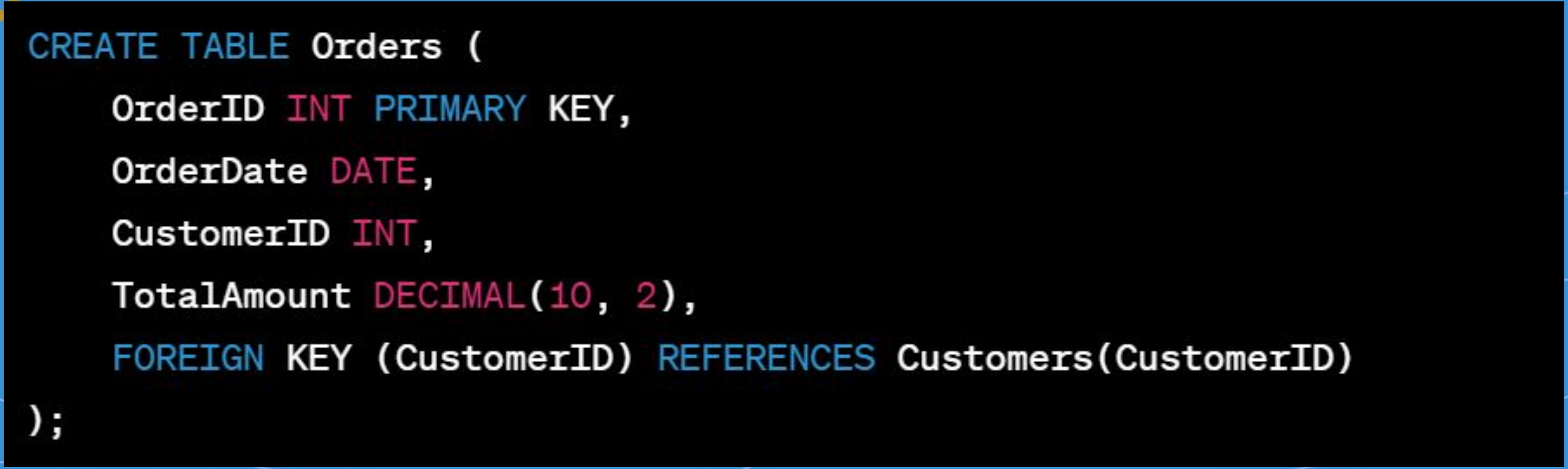
Example: Let’s say we have a "Customers" table, and you want to ensure that each customer has a unique identifier. You can use the "CustomerID" column as the primary key



```
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    Email VARCHAR(100)
);
```

Primary & Foreign keys

Now, let’s say you also have an "Orders" table that needs to reference customers from the "Customers" table (to know what they ordered). You can use the "CustomerID" column in the "Orders" table as a foreign key:



```
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    OrderDate DATE,
    CustomerID INT,
    TotalAmount DECIMAL(10, 2),
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);
```

Primary & Foreign keys

- So, the "CustomerID" column in the "Orders" table is a foreign key that references the primary key "CustomerID" in the "Customers" table.
- We do this cross-referencing to ensure that each order is associated with a valid customer, maintaining referential integrity.
- In summary, primary keys uniquely identify records within a table, while foreign keys establish relationships between tables by referencing the primary key of another table.
- Thus, we are able to maintain data integrity and support the structure of a relational database.

Integrating file-based & server-based databases

Firstly, what is database integration?

- ➢ The process of combining data from multiple databases or data sources into a unified coherent system.

Why is this of importance?

- ➢ Use of different data sources together in one place - you can combine spreadsheets or text files with structured databases (relational database management systems like SQLite)
- ➢ You can analyse all your data together which is essential for being able to make good decisions on all your available information
- ➢ Many more reasons, explore! :)

Integrating file-based & server-based databases

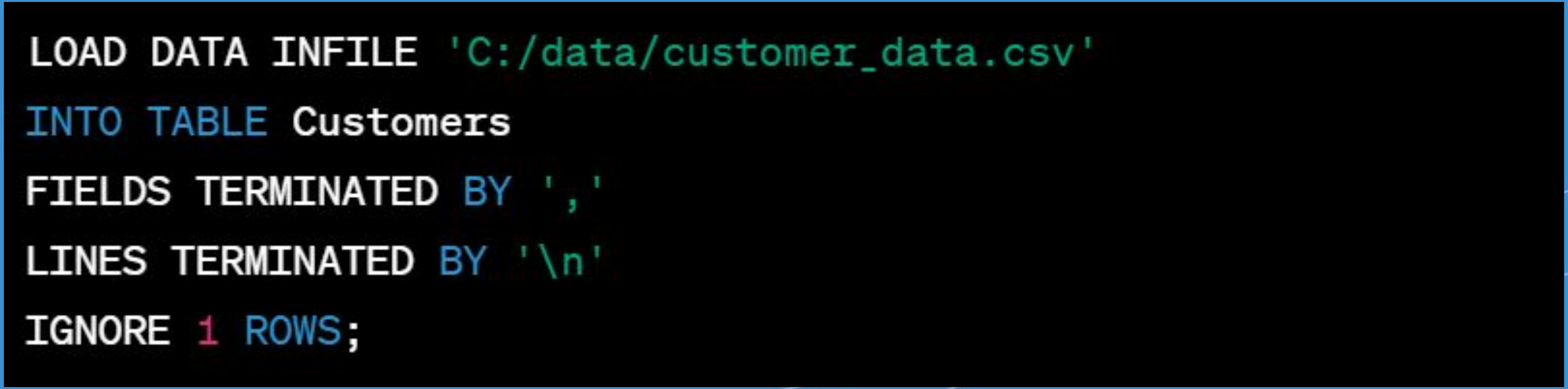
Here is a simple example. Let’s say you have data about your customers stored in two sources:

- A MySQL database (a server-based database) that contains a “Customers” table containing customer information.
- A CSV file (a file-based database) also containing information about customers.

And you need both data sources for your analysis.

Integrating file-based & server-based databases

We can use MySQL's 'LOAD DATA INFILE' statement to import the CSV file into our MySQL "Customers" table so that we have them in one place - the "Customers" table.



```
LOAD DATA INFILE 'C:/data/customer_data.csv'
INTO TABLE Customers
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

Questions and Answers

Questions around SQL