Software Engineering Bootcamp

Hyperiondev

Relational Databases and SQL

Lecture - Housekeeping

- ❑ The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly.
- ❑ No question is daft or silly - ask them!
- ❑ There are Q/A sessions at the end of the session, should you wish to ask any follow-up questions.
- ❑ For all non-academic questions, please submit a query: www.hyperiondev.com/support
- ❑ Report a safeguarding incident: http://hyperiondev.com/safeguardreporting

Objectives

1. Define and explain the concept of a relational database, including its key components such as tables, rows, columns, and primary keys.

2. Distinguish between different types of databases and articulate the advantages of using relational databases for structured data storage.

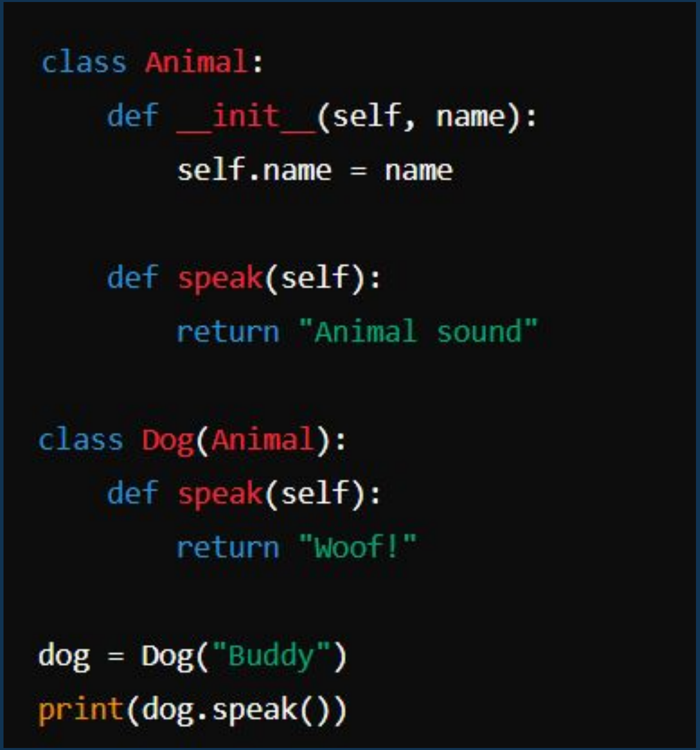
3. Write basic SQL commands to create tables, insert data, and perform simple queries, including SELECT statements with WHERE clauses and ORDER BY.

4. Explain the concept of table joins and write a basic INNER JOIN query to combine data from two related tables.

5. Identify real-world applications of relational databases and explain how they support various systems and services in everyday life.

Poll

What will be the output of the following code?

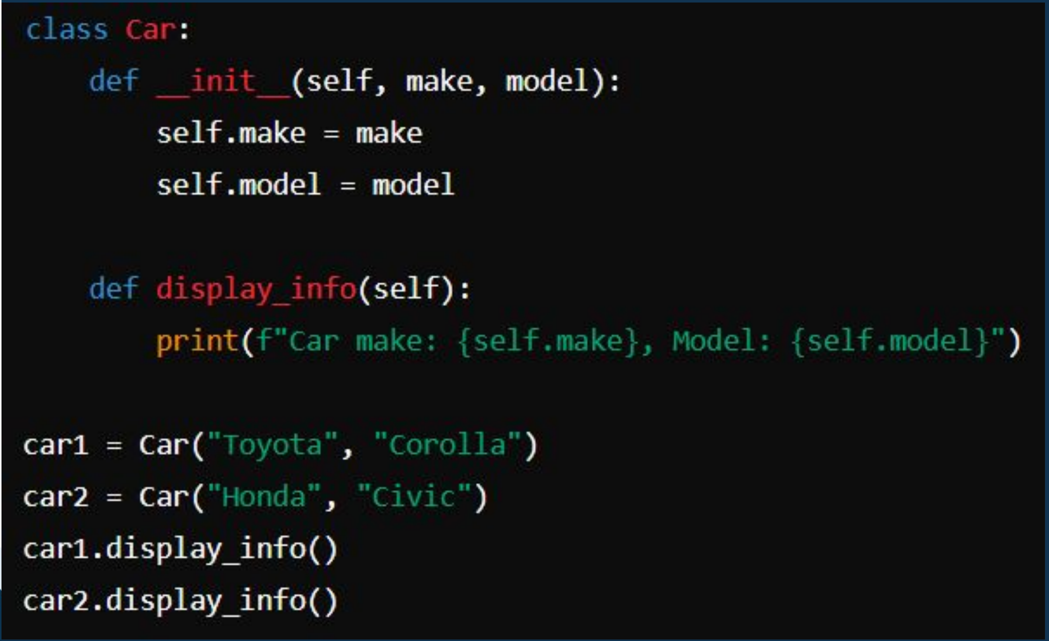


```
class Animal:
    def __init__(self, name):
        self.name = name

    def speak(self):
        return "Animal sound"

class Dog(Animal):
    def speak(self):
        return "Woof!"

dog = Dog("Buddy")
print(dog.speak())
```

- Animal sound
- Buddy
- Woof!
- Nothing

Poll

What will be the output of the following code snippet?



```
class Car:
    def __init__(self, make, model):
        self.make = make
        self.model = model

    def display_info(self):
        print(f"Car make: {self.make}, Model: {self.model}")

car1 = Car("Toyota", "Corolla")
car2 = Car("Honda", "Civic")
car1.display_info()
car2.display_info()
```

- The code will throw an error because display_info method is not defined
- The code will only print the make and model of car 1
- The code will only print the make and model of car 2
- The code will print the make and model of both cars

Understanding Databases in Everyday Life

Hyperiondev

Introduction to Databases

Definition of a database

- A **database** is a structured collection of data that is organized in a way that allows for efficient storage, retrieval, and management of information. It acts as a digital filing system where data is stored in tables, similar to a spreadsheet, with rows representing individual records and columns representing data fields.

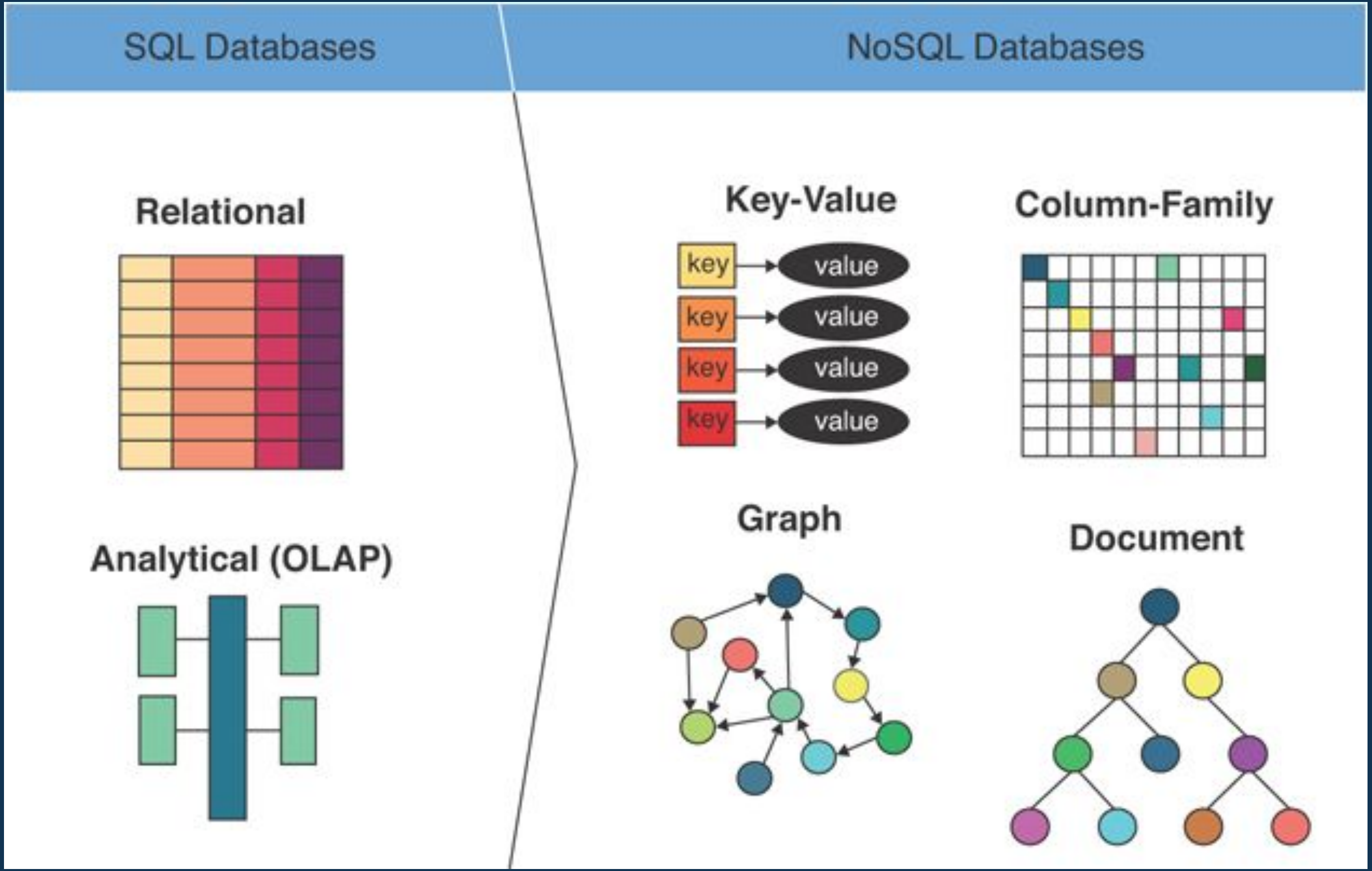
Types of databases

- **Relational Databases** (SQL):
  - Data is organized into **tables** (relations) with **rows** and **columns**.
  - Tables are linked by **keys** (**Primary** and **Foreign** Keys).
  - Commonly used for structured data with clear **relationships**.
  - Examples: **MySQL**, **PostgreSQL**, **SQLite**.

Types of databases

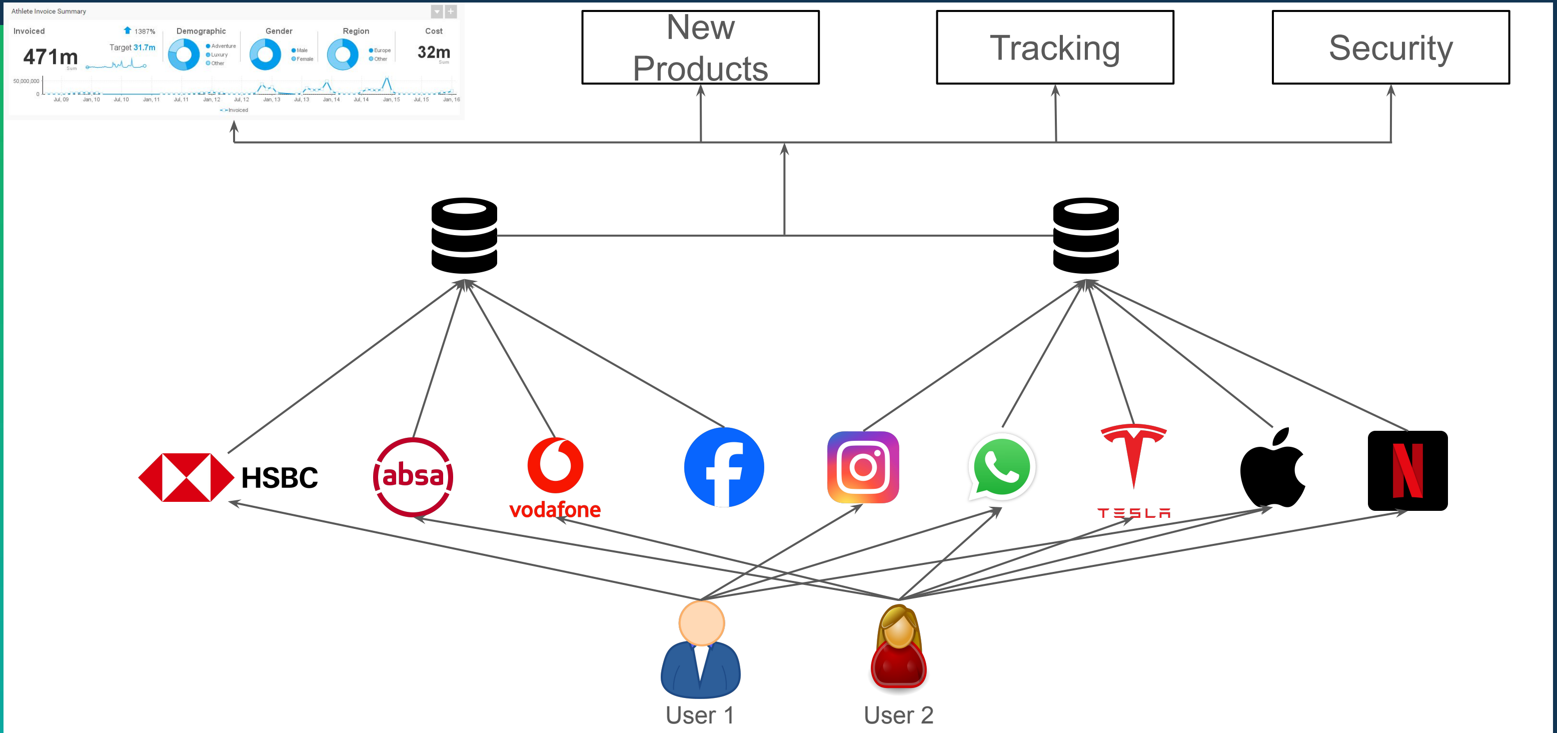
- **Non-Relational Databases** (NoSQL):
    - Data is stored in a flexible format, such as documents, key-value pairs, or graphs.
    - No fixed schema; ideal for unstructured or semi-structured data.
    - Handles large volumes of diverse data types.
    - Examples: MongoDB, Cassandra, Redis.

Types of databases



Source

Use cases of databases in everyday applications



Hyperiondev

Relational Databases

What is a Relational Database?

- A **relational database** organizes data into tables and uses keys (Primary and Foreign) to define relationships between these tables.

What is a Relational Database?

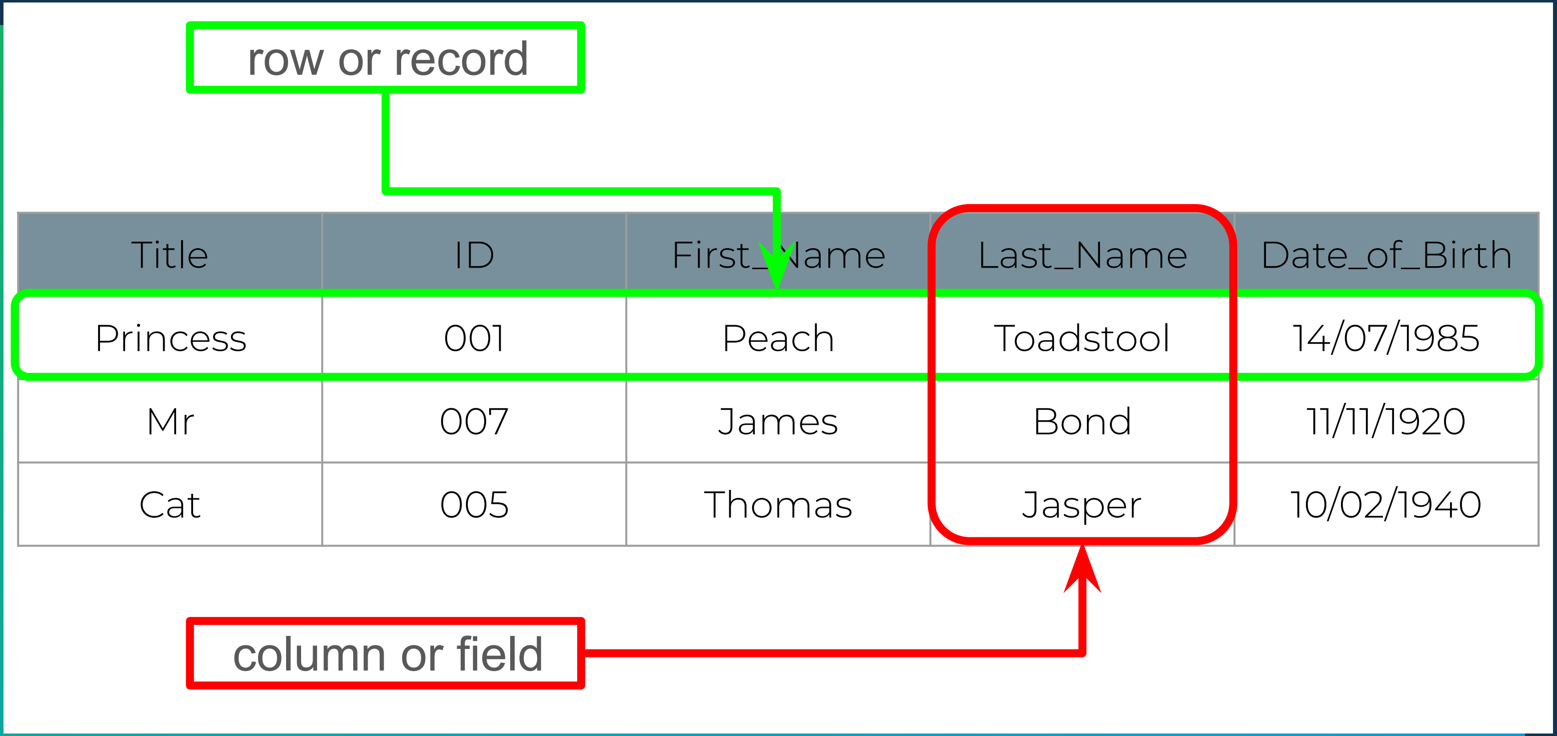
- **Advantages**:
    - Structured Data Management:
    - Data is organized in a clear, tabular format, making it easy to manage and navigate.
    - Data Integrity and Consistency:
    - Relationships and keys ensure that data remains accurate and consistent across tables.
    - Efficient Querying Using SQL:
    - Powerful querying capabilities allow for quick and precise data retrieval.

What is a Relational Database?

- **Example**:
  - Students Table: Stores student information.
  - Courses Table: Lists available courses.
  - Enrollments Table: Links students to the courses they’re enrolled in, demonstrating how tables relate to each other in a relational database.

Tables and Keys

Tables

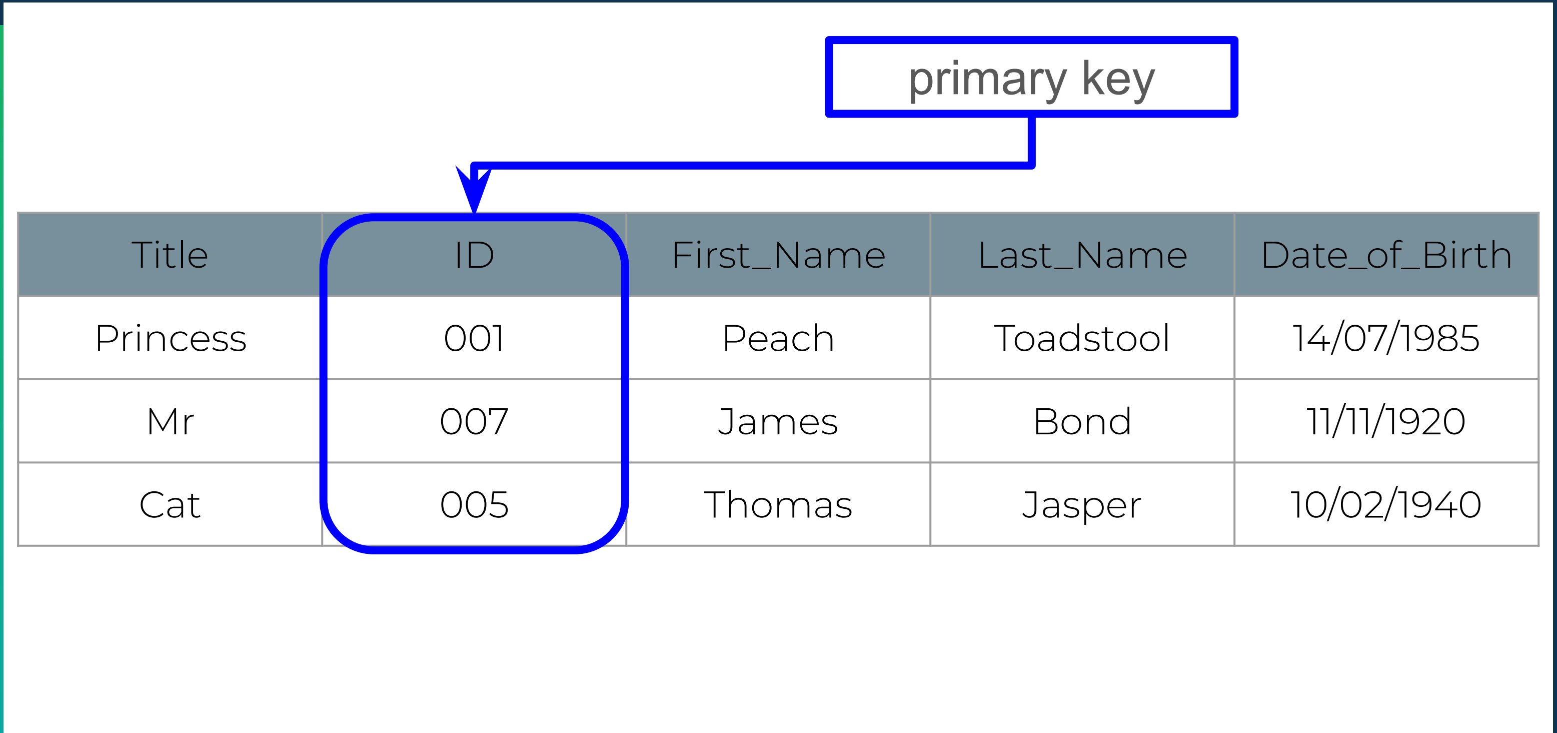


row or record

Title	ID	First_Name	Last_Name	Date_of_Birth
Princess	001	Peach	Toadstool	14/07/1985
Mr	007	James	Bond	11/11/1920
Cat	005	Thomas	Jasper	10/02/1940

column or field

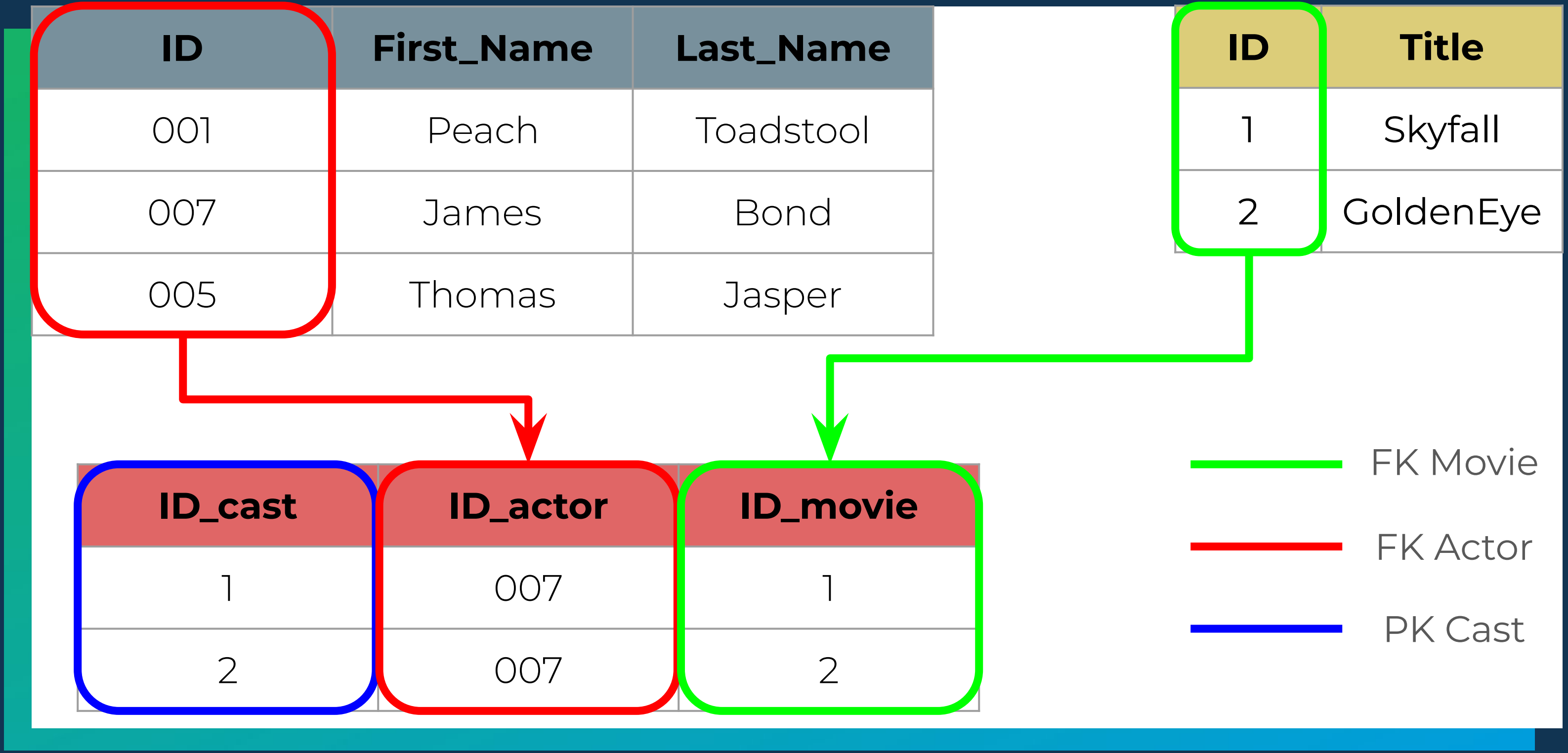
Primary Key



primary key

Title	ID	First_Name	Last_Name	Date_of_Birth
Princess	001	Peach	Toadstool	14/07/1985
Mr	007	James	Bond	11/11/1920
Cat	005	Thomas	Jasper	10/02/1940

Foreign Key



ID	First_Name	Last_Name
001	Peach	Toadstool
007	James	Bond
005	Thomas	Jasper

ID	Title
1	Skyfall
2	GoldenEye

ID_cast	ID_actor	ID_movie
1	007	1
2	007	2

🟩 FK Movie

🟥 FK Actor

🟦 PK Cast

Basics of Joins

INNER JOIN

ID	Name
001	Peach Toadstool
007	Mario
010	Sam Fisher
012	Kazuki Ito
020	Captain Falcon

ID_player	Name
012	Pro Evolution Soccer 6
060	Rainbow Six
020	Super Smash Bros. Melee

ID_player	Name_Game	Name_player
012	Pro Evolution Soccer 6	Peach Toadstool
020	Super Smash Bros. Melee	Captain Falcon

LEFT JOIN

ID	Name
001	Peach Toadstool
007	Mario
010	Sam Fisher
012	Kazuki Ito
020	Captain Falcon

ID_player	Name
012	Pro Evolution Soccer 6
020	Super Smash Bros. Melee

ID_player	Name_Game	Name_player
001	Null	Null
007	Null	Null
010	Null	Null
012	PES 6	Kazuki Ito
020	S.S.B.M	Captain Falcon

RIGHT JOIN

ID	Name
012	Kazuki Ito
020	Captain Falcon

ID_player	Name
012	Pro Evolution Soccer 6
060	Rainbow Six
020	Super Smash Bros. Melee

ID_player	Name_Game	Name_player
012	PES 6	Kazuki Ito
060	Rainbow Six	Null
020	S.S.B.M	Captain Falcon

Introduction to SQL

SQL

Definition	**SQL** (Structured Query Language) is a standard language used to manage and manipulate data in a relational database. It allows for performing various operations on the data.
CREATE	Define new tables and structure.
READ	Retrieve specific data from one or more tables.
UPDATE	Modify existing data within tables.
DELETE	Remove data from tables.

Basic SQL Commands

- **CREATE**

```sql
CREATE TABLE Students (
    student_id INT PRIMARY KEY,
    name VARCHAR(100),
    age INT
);
```

Basic SQL Commands

- **INSERT INTO**

```
INSERT INTO Students (student_id, name, age)

VALUES (1, 'Alice', 20);
```

Basic SQL Commands

- **SELECT**

```sql
SELECT * FROM Students;
```

Basic SQL Commands

- **ORDER BY**

```
SELECT * FROM Students

ORDER BY Grades;
```

Basic SQL Commands

- **UPDATE**

```
UPDATE Students

SET age = 21

WHERE student_id = 1;
```

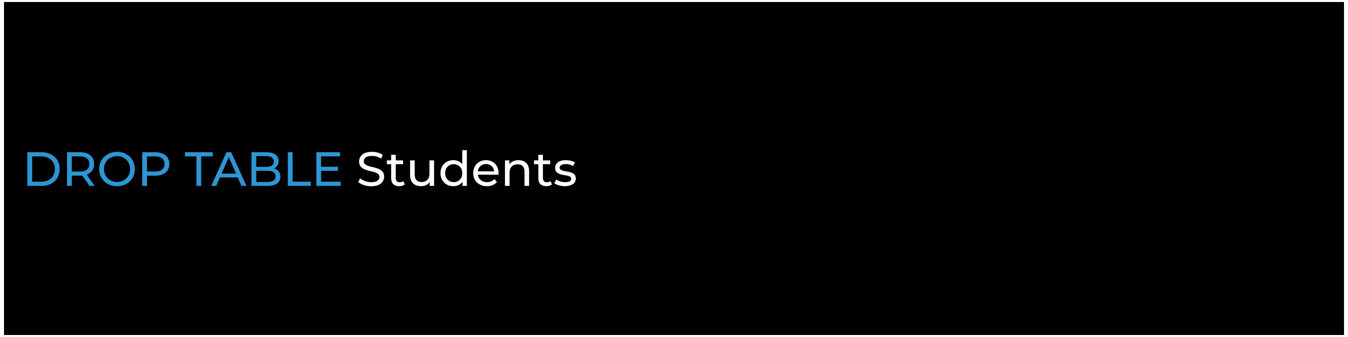
Basic SQL Commands

- **DELETE**

```
DELETE FROM Students

WHERE student_id = 1;
```

Basic SQL Commands

- **DROP**



```
DROP TABLE Students
```

SQL Databases: Accessible from Anywhere

Accessible from Anywhere

- **Remote Accessibility**:
  - SQL databases are often hosted on servers, allowing them to be accessed remotely by various applications, including web and mobile apps.
- **Multiple Access Points**:
  - Tools like DBeaver, DB Browser for SQLite, and other software can all interact with the same SQL database, regardless of where the software is running.

Accessible from Anywhere

- Real-World Examples:
    - Web Applications:
        - Websites like e-commerce platforms use SQL databases to store product information, user accounts, and transaction records.
    - Mobile Apps:
        - Banking apps use SQL databases to manage account balances, transaction histories, and user profiles.

Lesson Conclusion and Recap

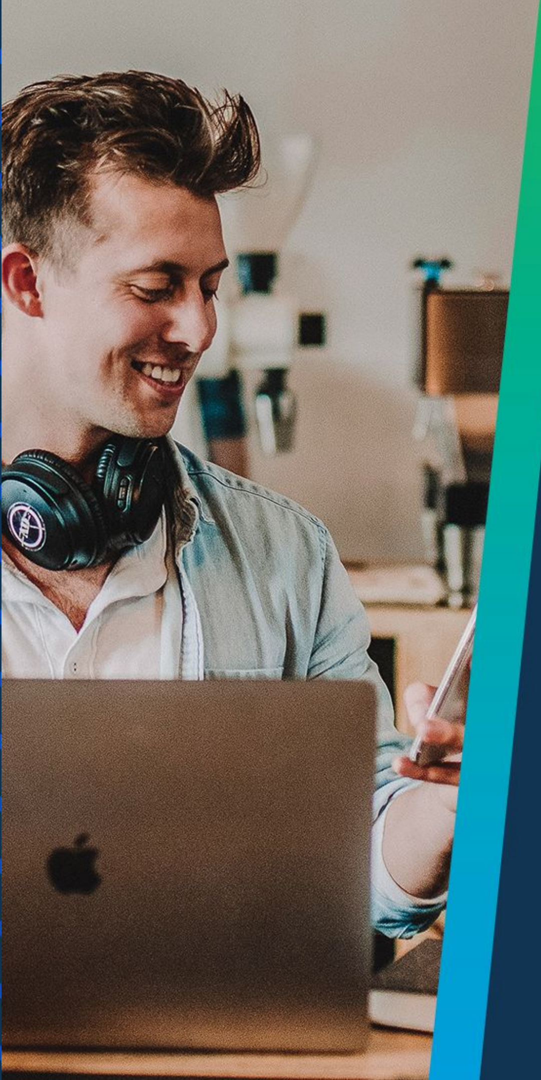
Recap the key concepts and techniques covered during the lesson.

- **Databases and Tables**: Understanding what a database is and how data is structured within tables, including rows and columns.
- **Primary and Foreign Keys**: The role of primary keys in uniquely identifying records and foreign keys in linking related tables.
- **Basic SQL Queries**: Techniques for creating tables, inserting data, and retrieving records using SQL commands.
- **Joins**: How to use INNER JOIN and LEFT JOIN to combine and retrieve data from multiple related tables.
- **Relational Database Design**: The importance of designing and managing a relational database with well-structured tables and defined relationships.

Hyperiondev

Q & A Section

Please use this time to ask any questions relating to the topic explained, should you have any



Hyperiondev

Thank you for joining us

Take regular breaks.
Stay hydrated.
Avoid prolonged screen time.
Remember to have fun :)

Some useful links

- Readings in Database Systems, 5th Edition: Readings in Database Systems, 5th Edition Peter Bailis, Joseph M. Hellerstein, Michael Stonebraker
- Databases Normalisation