



Welcome to this session: Skills Bootcamp - Introduction to Databases and SQL

The session will start shortly...

Questions? Drop them in the chat.
We'll have dedicated moderators
answering questions.



Skills Bootcamp Data Science Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(Fundamental British Values: Mutual Respect and Tolerance)**
- 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. We will be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: **Questions**

Skills Bootcamp Data Science Housekeeping

- For all **non-academic questions**, please submit a query: www.hyperiondev.com/support
- Report a safeguarding incident: www.hyperiondev.com/safeguardreporting
- We would love your feedback on lectures: [Feedback on Lectures.](#)
- Find all the lecture **content** in your [Lecture Backpack](#) on GitHub.
- If you are hearing impaired, kindly use your computer's function through Google chrome to enable captions.

Safeguarding & Welfare

We are committed to all our students and staff feeling safe and happy; we want to make sure there is always someone you can turn to if you are worried about anything.

If you are feeling upset or unsafe, are worried about a friend, student or family member, or you feel like something isn't right, speak to our safeguarding team:



Ian Wyles
Designated Safeguarding
Lead



Simone Botes



Nurhaan Snyman



Rafiq Manan



Ronald Munodawafa



Tevin Pitts

Scan to report a
safeguarding concern



or email the Designated
Safeguarding Lead:
Ian Wyles

safeguarding@hyperiondev.com

Skills Bootcamp Progression Overview

✓ Criterion 1 - Initial Requirements

Specific achievements **within the first two weeks** of the program.

To meet this criterion, students need to, by no later than **01 December 2024 (C11)** or **22 December 2024 (C12)**:

- **Guided Learning Hours (GLH):** Attend a **minimum of 7-8 GLH per week** (lectures, workshops, or mentor calls) for a total minimum of **15 GLH**.
- **Task Completion:** Successfully complete the **first 4 of the assigned tasks**.

✓ Criterion 2 - Mid-Course Progress

Progress through the successful completion of tasks **within the first half** of the program.

To meet this criterion, students should, by no later than **12 January 2025 (C11)** or **02 February 2025 (C12)**:

- **Guided Learning Hours (GLH):** Complete at least **60 GLH**.
- **Task Completion :** Successfully complete the **first 13 of the assigned tasks**.

Skills Bootcamp Progression Overview

✓ Criterion 3 – End-Course Progress

Showcasing students' progress nearing the completion of the course.

To meet this criterion, students should:

- **Guided Learning Hours (GLH):** Complete the **total minimum required GLH**, by the **support end date**.
- **Task Completion : Complete all mandatory tasks**, including any necessary resubmissions, by the end of the bootcamp, **09 March 2025 (C11)** or **30 March 2025 (C12)**.

✓ Criterion 4 - Employability

Demonstrating progress to find employment.

To meet this criterion, students should:

- **Record an Interview Invite:** Students are required to record proof of invitation to an interview by **30 March 2025 (C11)** or **04 May 2025 (C12)**.
 - **South Holland Students** are required to proof and interview by **17 March 2025**.
- **Record a Final Job Outcome :** Within 12 weeks post-graduation, students are required to record a job outcome.

Learning Outcomes

- ❖ **Describe basic database concepts** and the role of SQL in data management.
- ❖ **Execute CRUD operations using SQL** in a relational database system.
- ❖ **Construct SQL queries** to filter, aggregate, and manipulate datasets effectively.
- ❖ **Explain the concept of table joins** and write a basic INNER JOIN query to combine data from two related tables.
- ❖ **Implement database interactions** within Python applications, showcasing practical connection and data manipulation.
- ❖ **Design simple relational database schemas**, emphasizing the importance of normalization and efficient data organization.

Lecture Overview

- Databases
- SQL

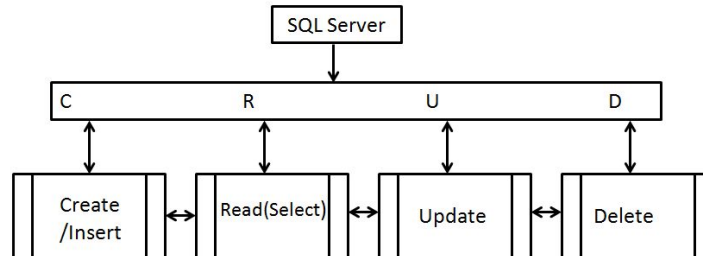


Which of the following is NOT a basic CRUD operation in database management?

- A. Create
- B. Retrieve
- C. Update
- D. Discard

Correct Answer: D

- ❖ **CRUD Operations** are the 4 basic operations which act as the foundation of any computer programming language.
- ❖ CRUD stands for: Create, Retrieve, Update and **Delete**.
- ❖ It is important to understand **what** CRUD operations are and **how** they are implemented in SQL Statements.





Which SQL keyword is used to retrieve data from a database?

- A. SELECT
- B. UPDATE
- C. DELETE
- D. INSERT

Correct Answer: A

- ❖ All four options are valid SQL statements.
- ❖ It is important to know the difference between the different SQL statements and what they accomplish.
- ❖ **SELECT** is a statement which retrieves data from one or more tables.
- ❖ **UPDATE** updates data in a table, **DELETE** deletes data from a table and **INSERT** inserts data into a table.



Which of the following is NOT a valid SQL data type?

- A. INT
- B. STRING
- C. DATE
- D. BOOLEAN

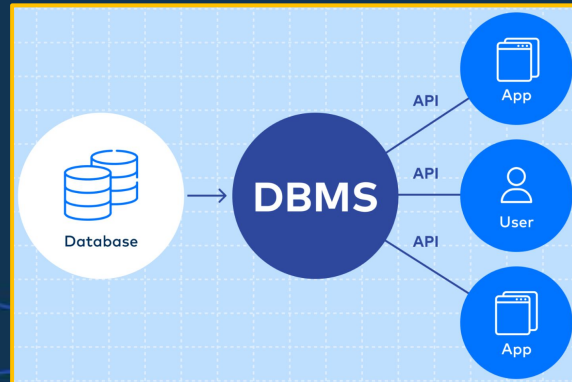
Correct Answer: B

- ❖ Strings of data are stored in SQL in different ways and there are a number of different String Data Types in SQL but **STRING is not** a type in SQL.
- ❖ Strings can be of the **CHAR(n)**, **VARCHAR(n)** or **TEXT** type, both of which can store strings.

Databases

An organized collection of structured information, or data.

- ❖ A database is usually controlled by a database engine, commonly known as a **Database Management System (DBMS)**.
- ❖ DBMSs serve as a **tool** between a user and their data, **organising** and **cataloging** the data for **quick and easy retrieval**.
- ❖ The data and the DBMS, and the applications associated with them are referred to as a **database system**, usually shortened to **database**.





Relational Databases

Any database system that allows data to be associated and grouped by common attributes.

- ❖ Relational databases are comprised of a number of tables (**relations**), within each are:
 - Rows also known as records or tuples
 - Columns also known as attributes or fields
- ❖ Each record is identified with a **unique key**, known as the **primary key**.
- ❖ Records from one table can be references in other tables using their key, in this case they are called **foreign keys**.
- ❖ Each table/relation represents one “**entity type**”.



Tables

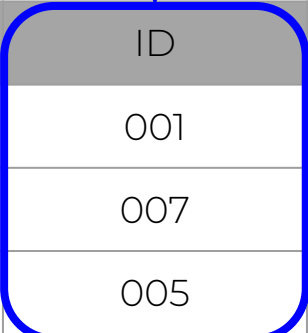
row or record

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



NoSQL Databases



- ❖ The performance of relational databases degrades as the volume of data increases.
 - ❖ Web applications usually have to store massive amounts of data, so NoSQL databases were developed to improve performance.
 - ❖ NoSQL databases have the following characteristics:
 - Not based on the relational model.
 - Support distributed database architectures.
 - High scalability, high availability and fault tolerance.
 - Support large amounts of sparse data.
 - Geared toward performance rather than transactional consistency
- 

RDMS Example

Consider the following scenario:

You are the captain of a spaceship and you would like to store all of the data pertaining to the ship in one place so you can easily access and update the data.

Some examples of the data you need to store is: information on each of the crewmates, information about each of the areas of the ship and the shift schedules which places each crew member at different areas of the ship at different times.

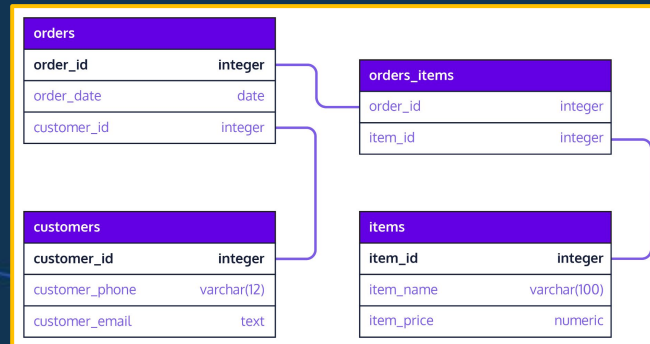
RDMS Example

crewID	firstName	lastName	roomNum	shiftTime	shiftArea
0	Zahra	Mohamed	7	19:00	Hull
1	Moumita	Aich	12	19:00	Engine
2	Anri	Lombard	4	19:00	Cafeteria
3	Julien	Nyambal	3	19:00	Stock
0	Zahra	Mohamed	7	20:00	Engine
1	Moumita	Aich	12	20:00	Stock

Database Schema

Defines how data is organised within a relational database.

- ❖ Considered to be the “**blueprint**” of a database, which **describes** how the data may **relate to other tables** or other data models.
- ❖ It defines any **logical constraints** such as table names, fields, data types and the relationships between these entities.
- ❖ Commonly use **visual representations** to illustrate database architecture.
- ❖ The process of designing data schemas is known as **data modelling**.



Let's Breathe!

Let's take a small break
before moving on to
the next topic.



SQL

Structures Query Language is a programming language used for communicating with relational databases.

- ❖ **SQL** is used by nearly all relational databases to **query, manipulate, and define data**.
- ❖ It was first developed at **IBM** in the **1970s** and is still widely used today although new programming languages are beginning to appear.
- ❖ SQL is divided into the following categories:
 - **Data Definition Language:** Defines the data objects within the database based on the defined data model.
 - **Data Manipulation Language:** Used to insert, update, or delete.
 - **Data Query Language:** Used to query or select data

SQL: RDBMSs

- ❖ We use SQL to interact with Relational Database Management Systems and it **facilitates several operations**.
- ❖ Different DBMSs use different **dialects** of SQL.
- ❖ There are many different RDBMSs which we could work with. We will talk about two in this lecture:
 - **SQLite:** Server-less database which is self-contained, best used for small, standalone apps and basic development and testing.
 - **MySQL:** Requires a server to run and requires a client-server architecture to interact over a network. This is best used for web-based applications, larger apps with multiple user access.

CRUD Operations

Create, Read, Update and Delete

- ❖ These are the 4 basic operations which act as the **foundation** of any computer programming language.
- ❖ We need to understand CRUD in SQL to interact with databases.
 1. **Create:** To add or insert data into the SQL tables.
 - a. **CREATE** TableName (ColumnName1 Datatype, ...);
 - b. **INSERT INTO** TableName (ColumnName1, ...) **VALUES** (Value 1, ... Value N), ... , (Value 1, ... Value N);
 2. **Read:** To retrieve or fetch data from the SQL tables.
 - a. **SELECT** * **FROM** TableName **WHERE** Condition;

CRUD Operations

3. Update: To make updates to the records in the SQL tables.

- a. **UPDATE** TableName **SET** ColumnName = Value
WHERE Condition;

4. Delete: To remove or delete records from the SQL tables.

- a. **DELETE FROM** TableName **WHERE** Condition;

CREATE

❖ SYNTAX:

CREATE TABLE tableName (column1 datatype constraint, column2 datatype constraint, ... , columnN datatype constraint)

❖ Constraints (these are not compulsory):

- NOT NULL - column cannot have a NULL value
- UNIQUE - all values in a column are different
- PRIMARY KEY - combines NOT NULL and UNIQUE
- FOREIGN KEY - prevents actions that would destroy table links
- CHECK - limit the value range that can be placed in a column
- DEFAULT - sets default value if none is specified
- AUTO_INCREMENT - field is automatically given a new value

CREATE Example

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

Can you describe the table that this code would create?

INSERT INTO

❖ SYNTAX:

```
INSERT INTO tableName (column1, column2, ... , columnN) VALUES [  
(value1, value2, ..., valueN), ... , (value1, value2, ..., valueN)]
```

```
INSERT INTO Students (student_id, name, age)  
VALUES (1, 'Alice', 20);
```

SELECT

❖ SYNTAX:

SELECT column1, column2, ... , columnN) **FROM** tableName **WHERE** Condition (**GROUP BY** column **HAVING** filter **ORDER BY** column)

- ❖ * can be used when selecting all columns.
- ❖ **Group By:** Groups rows with the same values, which allows us to perform data aggregation with aggregate functions like COUNT(), MAX(), MIN() SUM() AVG()
- ❖ **Having:** Allows us to set conditions when we use aggregate functions
- ❖ **Order By:** Allows us to sort by specific columns in ASC or DESC order

SELECT Example

```
SELECT * FROM Students;
```

```
SELECT * FROM Students  
ORDER BY age;
```

Where Clause

- ❖ Filters the results of our search
- ❖ We can pass logical and comparison operators
- ❖ Comparison Operators
 - < (Less than)
 - > (Greater than)
 - = (Equal to)
 - <> (Equality)

```
SELECT * FROM Students  
WHERE name = "Alice";
```

Logical Operators

- ❖ We can use logical operators to check different things in our WHERE clause
 - **AND** - Used like a WHERE if you want to check more than one condition.
 - **OR** - Checks if one of two conditions is true
 - **BETWEEN** - Checks if a value falls between a given range
 - **IN** - Checks if a value is present in a list of values.
 - **NOT** - Checks the opposite for a condition.

String Comparison

- ❖ **LIKE:** Checks if a value exists in a list by using '%' or '_' to determine what is being looked for
 - % - Any value and the specified characters
 - **%hello** - looks for values ending in hello
 - **hello%** - looks for values starting with hello
 - _ - There can be any character at the specified position
 - **_im** - Any letter + im, eg. Jim, Kim, Him
 - **__m** - eg. Gym, sum, hum

UPDATE

❖ SYNTAX:

```
UPDATE tableName SET column1 = value1, ... columnN = valueN  
WHERE Condition
```

```
UPDATE Students
```

```
SET age = 21
```

```
WHERE student_id = 1;
```

DELETE

❖ SYNTAX:

```
DELETE FROM tableName WHERE Condition
```

```
DELETE FROM Students
```

```
WHERE student_id = 1;
```

Aliasing

- ❖ Allows us to give **different names** to the values we are working with.
- ❖ We can alias **tables and columns**
- ❖ We use the **AS keyword** for aliasing
- ❖ Keep the order of execution in mind because we won't be able to use the column aliases for a lot of operations

```
SELECT s.name, t.name  
  
FROM Students AS s, Teachers AS t  
  
WHERE s.teacher_id = t.teacher_id;
```


Joins

- ❖ When working with relational databases, we like to **break our data up** into different tables to make our data more consistent.
- ❖ Storing data in different tables is good for **storage and consistency**, but when viewing data, a single table doesn't always have all of the information that we need
- ❖ JOINS allow us to **combine data from multiple tables** into a single output.

INNER JOIN

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

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

ID_ play er	Name_Ga me	Name_pla yer
012	Pro Evolution Soccer 6	Kazuki Ito
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_pl ayer	Name
012	Pro Evolution Soccer 6
020	Super Smash Bros. Melee

ID_ play er	Name_Ga me	Name_pla yer
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_pl ayer	Name
012	Pro Evolution Soccer 6
060	Rainbow Six
020	Super Smash Bros. Melee

ID_ play er	Name_Ga me	Name_pla yer
012	PES 6	Kazuki Ito
060	Rainbow Six	Null
020	S.S.B.M	Captain Falcon



Which of the following is an advantage of using a relational database?

- A. Flexibility in data storage
- B. High scalability for large datasets
- C. Data consistency and integrity
- D. Low cost of implementation

Correct Answer: C

- ❖ Relational Databases have many advantages but the main one is **data consistency and integrity**, this is due to the validations and rules defined for each RDMS.
- ❖ **Non-relational databases** are more commonly known for their **flexibility** specifically when it comes to the way data is stored.
- ❖ Relational databases are only **vertically scalable** but are not flexible enough to be horizontally scalable.
- ❖ Cost refers here to cost in resources (space and time) but this is not an advantage for RDMSs.



What is the purpose of the **GROUP BY** clause in SQL?

- A. Filter rows based on a specified condition
- B. Sort result set in ascending/descending order
- C. Group rows that have the same values into summary rows
- D. Perform arithmetic operations on numeric columns

Correct Answer: C

- ❖ **Group By:** Groups rows with the same values, which allows us to perform data aggregation with aggregate functions like COUNT(), MAX(), MIN() SUM() AVG()
- ❖ We use HAVING or WHERE to filter rows
- ❖ We use ORDER BY to sort the result set

Summary

- ★ **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.

CoGrammar

Q & A SECTION

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

Thank you for attending



CoGrammar



Department
for Education