2025

Database Capstone Project

SQL & DATA MODELING SPRINT

ARWA AL RAHBI

Table of Contents

Introduction

Before any charts glow or models crunch numbers, the data itself has to behave. That starts with structured tables—clean rows and columns that keep every value in the right place. Thoughtful data modeling then sketches how those pieces fit together, stripping out redundancy and clarifying relationships. A relational database enforces those rules at scale, so the data stays accurate no matter how fast it grows. And SQL gives you a quick, precise way to sift, join, and shape that information, handing Python or Pandas a dataset that's already reliable and ready for machine-learning insight.

Why is structured data important in data science pipelines?

Data science pipelines require structured data since it is tidy in standardized formats, such as rows and columns, and they can be analyzed easily, modeled, and cleaned. It facilitates effective querying, filtering and transformation all of which are crucial part of readying data to be visualized, analyzed with statistics or to be training examples to a machine learner. Structured formats provide coherence, reproducibility, connectivity with tools (e.g., SQL, Pandas, TensorFlow), and code (e.g., Python) on the pipeline.

What role does data modeling play in preparing data for analysis or machine learning?

Data modeling determines the data organizational, relation, and access pattern. It plays a very important role during the initial part of any project since it will decide the quality as well as the usability of the data. Supposedly it is a good data model with very little redundancy, logical consistency, and entity relationship. This preparation step allows fluent feature engineering, increases query performance, and guarantees aligning the data to analysis or machine learning models, calorimetrically.

How do relational databases support scalable and clean data practices in real-world data science projects?

To provide structure, relational databases use schemas, relationships and constraints (e.g. foreign keys, data types). This makes the data integrity, gets rid of duplication and makes the data normalized that is easily scalable. In practice with the kind of data science projects that need huge and varied data sets (e.g., customer data, transactions, logs), relational databases enable effective joins, aggregation and historical tracking necessary to large-scale correct and repeatable operations.

Why is SQL still considered a foundational skill even with tools like Python and Pandas?

SQL is universal in the sense that it is engineered to manipulate really big records without requiring all data to be in memory via queries with the source of the information (SQL PostgreSQL, MySQL, SQL Server, etc.). When Python and Pandas are powerful tools to manipulate data in memory, SQL is preferable to extract, filter, group and aggregate the data - which can be the first step in any analysis. The ability to work with SQL-related data warehouses, data lakes, and ETL processes enables data scientists to be efficient with raw data, and high-level analysis, and it is a goal in itself to know SQL.

Can you give an example of how SQL is used to extract insights before applying machine learning?

Yes. Suppose a company wants to predict customer churn. Before training a machine learning model, a data scientist might use SQL to extract features such as:

```
SELECT
   customer_id,
   COUNT(order_id) AS total_orders,
   AVG(order_amount) AS avg_spending,
   MAX(order_date) AS last_order_date
FROM Orders
GROUP BY customer_id;
```

The following SQL applies aggregation on the historic transaction data, to generate even new feature (such as total orders and average spending per customer). The information is then fed to a machine learning model (e.g. logistic regression or random forest) to classify the risk of churn. SQL would not be used to retrieve and prepare this data at scale efficiently and without error.

Reflection

The development and execution of this relational database system demonstrate a practical understanding of real-world training management needs. Through careful requirements analysis, the system addresses the distinct roles of trainee, trainer, and admin, ensuring each user can interact with relevant data efficiently. The ER model accurately maps the complex many-to-many and one-to-many relationships between trainees, trainers, courses, and schedules using junction tables like Enrollment and Schedule. The queries created serve operational goals: trainees can view and track their courses and schedules; trainers can manage their assigned sessions and monitor trainees; and admins can add, assign, and oversee the training process in real-time. Each SQL command — from simple selections to advanced JOINs and aggregations — reflects the thoughtful design of a scalable, normalized system. This implementation not only satisfies functional requirements but also lays the foundation for extending the system with role-based access control, automation, and analytics features.

Requirements Analysis

Trainee Needs

- View all available courses → needs access to the **Course** table.
- See courses they have enrolled in → via the **Enrollment** table (links Trainee ↔ Course).
- Check schedule details of their enrolled courses → use **Schedule**, join with **Enrollment** and **Course**.

Trainer Needs

- View courses they are assigned to → via **Schedule** (Trainer ↔ Course).
- See scheduled sessions (dates and time slots) → **Schedule** table.
- Access list of enrolled trainees per course → **Enrollment** + **Course** + **Trainee**, filtered by trainer's courses.

Admin Needs

- Add new courses → insert into **Course** table.
- Assign trainers → via **Schedule** table (connects Course ↔ Trainer).
- Create and manage schedules → insert/update in **Schedule**.
- Enroll trainees → insert into **Enrollment** table.
- Monitor course enrollments and scheduling → JOINs across **Course**, **Schedule**, **Enrollment**.

Entities and Their Relationships

1. Trainee

- trainee_id (PK)
- Enrolls in many courses → **many-to-many** with **Course** via **Enrollment**

2. Trainer

- trainer_id (PK)
- Assigned to multiple courses via **Schedule**
- Can teach multiple courses; a course can have one trainer per schedule

3. Course

- course_id (PK)
- Enrolled in by many trainees
- Scheduled by one or more trainers (depending on schedule)

4. Schedule

- Connects **Course** and **Trainer**
- Contains session details: start_date, end_date, time_slot

5. Enrollment

- Connects **Trainee** and **Course**
- Stores enrollment date

Entity-Relationship Summary

Entity A	Relationship	Entity B	Type
Trainee	Enrolls in	Course	Many-to-Many (via Enrollment)
Trainer	Teaches	Course	One-to-Many (via Schedule)
Course	Has	Schedule	One-to-Many
Schedule	Belongs to	Course, Trainer	Many-to-One

ER Model

This system models how trainees, trainers, courses, schedules, and enrollments interact. It follows normalized relational database principles and uses clear one-to-many (1:M) and many-to-many (M:N) relationships.

Normalization: Data is not duplicated; relationships are handled through foreign keys.

Scalability: You can add more schedules, trainees, or courses without affecting existing data.

Query Power: This setup supports rich SQL queries — like showing all trainees in a session, checking trainer workload, or identifying the most popular course.

Flexibility: It allows changes such as assigning a new trainer to an existing schedule, or enrolling more trainee's mid-way.

Relationships

Trainee (M) ->(M) Enrollment

Many trainee can enroll in multiple courses (via multiple rows in Enrollment).

Course (M) ->(M) Enrollment

A course can have multiple trainees enrolled.

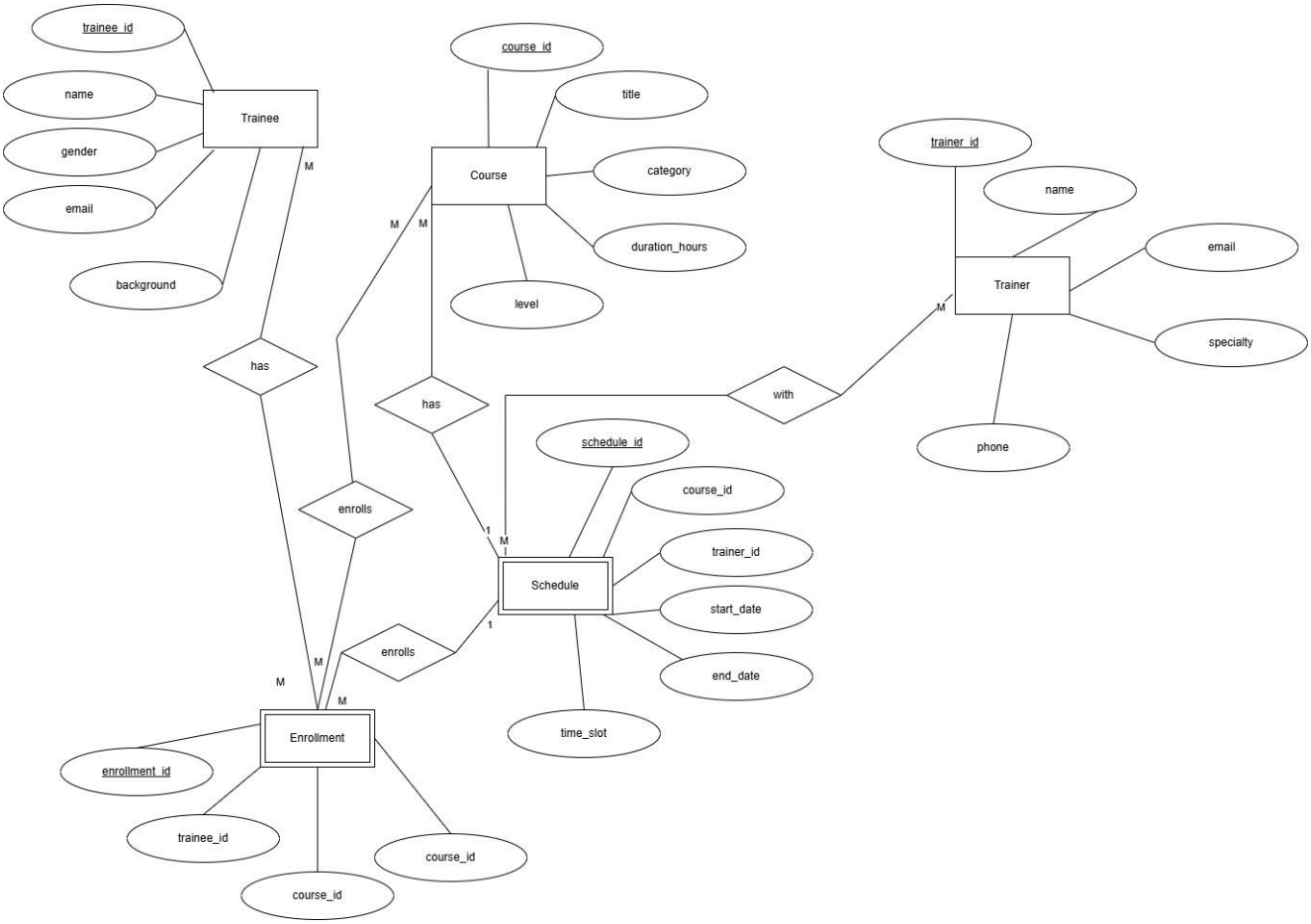
Trainer (M) ->(M) Schedule

Many trainer can handle multiple course schedules.

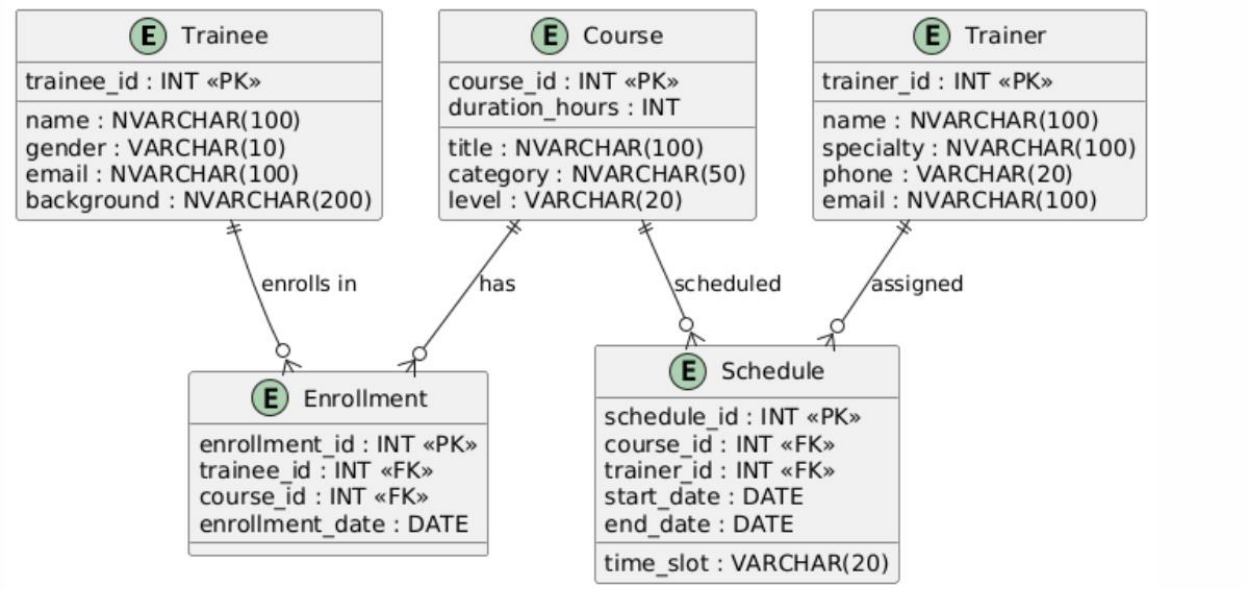
Course (M) ->(1) Schedule

A course can be scheduled multiple times (different dates/time slots).

Enrollment(M) ->(1) Schedule
Many enrollments in one schedule



Relational Schema Mapping



Query

1. Select all beginner-level courses

```sql
-- Get all courses where the level is Beginner
SELECT course_id, title, category, level
FROM Course
WHERE level = 'Beginner';
```

100 %

Results | Messages

	course_id	title	category	level
1	1	Database Fundamentals	Databases	Beginner
2	2	Web Development Basics	Web	Beginner

2. View all courses a specific trainee is enrolled in (with schedule)

```sql
-- Show the courses and schedule details for a trainee named 'Aisha Al-Harthy'
SELECT
    t.name AS trainee_name,
    c.title AS course_title,
    s.start_date,
    s.end_date,
    s.time_slot
FROM Enrollment AS e
JOIN Trainee AS t ON e.trainee_id = t.trainee_id
JOIN Course AS c ON e.course_id = c.course_id
LEFT JOIN Schedule AS s ON s.course_id = c.course_id
WHERE t.name = 'Aisha Al-Harthy';
```

100 %

Results | Messages

	trainee_name	course_title	start_date	end_date	time_slot
1	Aisha Al-Harthy	Database Fundamentals	2025-07-01	2025-07-10	Morning
2	Aisha Al-Harthy	Advanced SQL Queries	2025-07-15	2025-07-22	Morning

3. List all trainers and the courses they teach

```sql
-- List trainers along with the courses they are scheduled to teach
SELECT
    tr.name AS trainer_name,
    c.title AS course_title,
    s.time_slot
FROM Schedule AS s
JOIN Trainer AS tr ON s.trainer_id = tr.trainer_id
JOIN Course AS c ON s.course_id = c.course_id
ORDER BY tr.name;
```

100 %

Results | Messages

	trainer_name	course_title	time_slot
1	Khalid Al-Maawali	Database Fundamentals	Morning
2	Khalid Al-Maawali	Advanced SQL Queries	Morning
3	Noura Al-Kindi	Web Development Basics	Evening
4	Salim Al-Harthy	Data Science Introduction	Weekend

4. Count how many trainees are enrolled in each course

```sql
-- Count total enrollments per course
SELECT
    c.title AS course_title,
    COUNT(e.trainee_id) AS total_enrollments
FROM Course AS c
LEFT JOIN Enrollment AS e ON c.course_id = e.course_id
GROUP BY c.title
ORDER BY total_enrollments DESC;
```

100 %

Results | Messages

	course_title	total_enrollments
1	Data Science Introduction	2
2	Database Fundamentals	2
3	Web Development Basics	1
4	Advanced SQL Queries	1

5. Display all course schedules ordered by start date

```sql
-- Display course schedules with trainer and time slot, ordered by start date
SELECT
    c.title AS course_title,
    tr.name AS trainer_name,
    s.start_date,
    s.end_date,
    s.time_slot
FROM Schedule AS s
JOIN Course AS c ON s.course_id = c.course_id
JOIN Trainer AS tr ON s.trainer_id = tr.trainer_id
ORDER BY s.start_date;
```

100 %

Results | Messages

	course_title	trainer_name	start_date	end_date	time_slot
1	Database Fundamentals	Khalid Al-Maawali	2025-07-01	2025-07-10	Morning
2	Web Development Basics	Noura Al-Kindi	2025-07-05	2025-07-20	Evening
3	Data Science Introduction	Salim Al-Harthy	2025-07-10	2025-07-25	Weekend
4	Advanced SQL Queries	Khalid Al-Maawali	2025-07-15	2025-07-22	Morning

6. Find all trainees enrolled in 'Data Science Introduction'

```sql
-- Get trainees enrolled in the course 'Data Science Introduction'
SELECT
    t.name AS trainee_name,
    t.email,
    c.title AS course_title,
    e.enrollment_date
FROM Enrollment AS e
JOIN Trainee AS t ON e.trainee_id = t.trainee_id
JOIN Course AS c ON e.course_id = c.course_id
WHERE c.title = 'Data Science Introduction';
```

100 %

Results | Messages

	trainee_name	email	course_title	enrollment_date
1	Omar Al-Balushi	omar@example.com	Data Science Introduction	2025-06-04
2	Fatma Al-Hinai	fatma@example.com	Data Science Introduction	2025-06-05

7. List all courses along with assigned trainers

```sql
-- Show all courses and the trainers assigned to them
SELECT
    c.title AS course_title,
    tr.name AS trainer_name,
    tr.specialty
FROM Schedule AS s
JOIN Course AS c ON s.course_id = c.course_id
JOIN Trainer AS tr ON s.trainer_id = tr.trainer_id
ORDER BY c.title;
```

100 %

Results | Messages

	course_title	trainer_name	specialty
1	Advanced SQL Queries	Khalid Al-Maawali	Databases
2	Data Science Introduction	Salim Al-Harthy	Data Science
3	Database Fundamentals	Khalid Al-Maawali	Databases
4	Web Development Basics	Noura Al-Kindi	Web Development

Trainee Perspective Query Challenges

1. Show all available courses (title, level, category)

```sql
-- Retrieve all course titles with their level and category
SELECT
    title,
    level,
    category
FROM Course;
```

00 %

Results | Messages

	title	level	category
1	Database Fundamentals	Beginner	Databases
2	Web Development Basics	Beginner	Web
3	Data Science Introduction	Intermediate	Data Science
4	Advanced SQL Queries	Advanced	Databases

2. View beginner-level Data Science courses

```sql
-- Get only beginner-level courses that belong to the 'Dat
SELECT
    title,
    level,
    category
FROM Course
WHERE level = 'Beginner' AND category = 'Data Science';
```

100 %

Results | Messages

title	level	category

3. Show courses this trainee is enrolled in

```sql
-- List course titles the trainee with ID 1 is enrolled in
SELECT
    c.title
FROM Enrollment AS e
JOIN Course AS c ON e.course_id = c.course_id
WHERE e.trainee_id = 1;
```

100 %

Results | Messages

	title
1	Database Fundamentals
2	Advanced SQL Queries

4. View the schedule (start_date, time_slot) for the trainee's enrolled courses

```sql
-- Show schedule (start date and time slot) for courses the t
SELECT
    c.title AS course_title,
    s.start_date,
    s.time_slot
FROM Enrollment AS e
JOIN Course AS c ON e.course_id = c.course_id
JOIN Schedule AS s ON c.course_id = s.course_id
WHERE e.trainee_id = 1;
```

100 %

Results | Messages

	course_title	start_date	time_slot
1	Database Fundamentals	2025-07-01	Morning
2	Advanced SQL Queries	2025-07-15	Morning

5. Count how many courses the trainee is enrolled in

```sql
-- Count the number of enrollments for trainee ID 1
SELECT
    COUNT(*) AS total_courses
FROM Enrollment
WHERE trainee_id = 1;
```

100 %

Results | Messages

	total_courses
1	2

6. Show course titles, trainer names, and time slots the trainee is attending

```sql
-- Show course title, trainer name, and time slot for each co
SELECT
    c.title AS course_title,
    tr.name AS trainer_name,
    s.time_slot
FROM Enrollment AS e
JOIN Course AS c ON e.course_id = c.course_id
JOIN Schedule AS s ON c.course_id = s.course_id
JOIN Trainer AS tr ON s.trainer_id = tr.trainer_id
WHERE e.trainee_id = 1;
```

100 %

Results | Messages

	course_title	trainer_name	time_slot
1	Database Fundamentals	Khalid Al-Maawali	Morning
2	Advanced SQL Queries	Khalid Al-Maawali	Morning

Trainer Perspective Query Challenges

1. List all courses the trainer is assigned to

```sql
-- Get all courses assigned to trainer with ID 1
SELECT
    c.title AS course_title
FROM Schedule AS s
JOIN Course AS c ON s.course_id = c.course_id
WHERE s.trainer_id = 1;
```

100 %

Results | Messages

	course_title
1	Database Fundamentals
2	Advanced SQL Queries

2. Show upcoming sessions (with dates and time slots)

```sql
-- Show sessions (start/end date and time slot) scheduled in
SELECT
    c.title AS course_title,
    s.start_date,
    s.end_date,
    s.time_slot
FROM Schedule AS s
JOIN Course AS c ON s.course_id = c.course_id
WHERE s.trainer_id = 1 AND s.start_date > GETDATE();
```

100 %

Results | Messages

	course_title	start_date	end_date	time_slot
1	Database Fundamentals	2025-07-01	2025-07-10	Morning
2	Advanced SQL Queries	2025-07-15	2025-07-22	Morning

3. See how many trainees are enrolled in each of your courses

```sql
-- Count enrolled trainees per course taught by trainer ID 1
SELECT
    c.title AS course_title,
    COUNT(e.trainee_id) AS total_trainees
FROM Schedule AS s
JOIN Course AS c ON s.course_id = c.course_id
LEFT JOIN Enrollment AS e ON c.course_id = e.course_id
WHERE s.trainer_id = 1
GROUP BY c.title;
```

100 %

Results | Messages

	course_title	total_trainees
1	Advanced SQL Queries	1
2	Database Fundamentals	2

4. List names and emails of trainees in each of your courses

```sql
-- List names and emails of trainees in courses taught by tra
SELECT
    c.title AS course_title,
    t.name AS trainee_name,
    t.email AS trainee_email
FROM Schedule AS s
JOIN Course AS c ON s.course_id = c.course_id
JOIN Enrollment AS e ON c.course_id = e.course_id
JOIN Trainee AS t ON e.trainee_id = t.trainee_id
WHERE s.trainer_id = 1
ORDER BY c.title;
```

00 %

Results | Messages

	course_title	trainee_name	trainee_email
1	Advanced SQL Queries	Aisha Al-Harthy	aisha@example.com
2	Database Fundamentals	Aisha Al-Harthy	aisha@example.com
3	Database Fundamentals	Sultan Al-Farsi	sultan@example.com

5. Show the trainer's contact info and assigned courses

```sql
-- Display phone, email, and course titles for trainer ID 1
SELECT
    tr.name AS trainer_name,
    tr.phone,
    tr.email,
    c.title AS course_title
FROM Trainer AS tr
JOIN Schedule AS s ON tr.trainer_id = s.trainer_id
JOIN Course AS c ON s.course_id = c.course_id
WHERE tr.trainer_id = 1;
```

100 %

Results | Messages

	trainer_name	phone	email	course_title
1	Khalid Al-Maawali	96891234567	khalid@example.com	Database Fundamentals
2	Khalid Al-Maawali	96891234567	khalid@example.com	Advanced SQL Queries

6. Count the number of courses the trainer teaches

```sql
-- Count the number of unique courses assigned to trainer ID
SELECT
    COUNT(DISTINCT course_id) AS course_count
FROM Schedule
WHERE trainer_id = 1;
```

100 %

Results | Messages

	course_count
1	2

Admin Perspective Query Challenges

1. Add a new course (INSERT)

```
-- Add a new course to the Course table
INSERT INTO Course (title, category, duration_hours, level)
VALUES ('Python Programming Basics', 'Programming', 40, 'Beginner');
```

100 %

Messages

```
(1 row affected)

Completion time: 2025-06-29T22:01:02.2206588+05:00
```

2. Create a new schedule for a trainer (INSERT)

```
-- Create a schedule assigning trainer 2 to course 1 starting on August 1
INSERT INTO Schedule (course_id, trainer_id, start_date, end_date, time_slot)
VALUES (1, 2, '2025-08-01', '2025-08-15', 'Evening');
```

100 %

Messages

```
(1 row affected)

Completion time: 2025-06-29T22:15:45.9737500+05:00
```

3. View all trainee enrollments with course title and schedule info

```sql
-- Show which trainees are enrolled in which courses, along with schedule d
SELECT
    t.name AS trainee_name,
    c.title AS course_title,
    s.start_date,
    s.end_date,
    s.time_slot
FROM Enrollment AS e
JOIN Trainee AS t ON e.trainee_id = t.trainee_id
JOIN Course AS c ON e.course_id = c.course_id
LEFT JOIN Schedule AS s ON c.course_id = s.course_id
ORDER BY t.name;
```

100 %

Results | Messages

	trainee_name	course_title	start_date	end_date	time_slot
1	Aisha Al-Harthy	Database Fundamentals	2025-07-01	2025-07-10	Morning
2	Aisha Al-Harthy	Database Fundamentals	2025-08-01	2025-08-15	Evening
3	Aisha Al-Harthy	Advanced SQL Queries	2025-07-15	2025-07-22	Morning
4	Fatma Al-Hinai	Data Science Introduction	2025-07-10	2025-07-25	Weekend
5	Mariam Al-Saadi	Web Development Basics	2025-07-05	2025-07-20	Evening
6	Omar Al-Balushi	Data Science Introduction	2025-07-10	2025-07-25	Weekend
7	Sultan Al-Farsi	Database Fundamentals	2025-07-01	2025-07-10	Morning
8	Sultan Al-Farsi	Database Fundamentals	2025-08-01	2025-08-15	Evening

4. Show how many courses each trainer is assigned to

```sql
-- Count the number of courses assigned to each trainer
SELECT
    tr.name AS trainer_name,
    COUNT(DISTINCT s.course_id) AS total_courses
FROM Trainer AS tr
LEFT JOIN Schedule AS s ON tr.trainer_id = s.trainer_id
GROUP BY tr.name;
```

100 %

Results | Messages

	trainer_name	total_courses
1	Khalid Al-Maawali	2
2	Noura Al-Kindi	2
3	Salim Al-Harthy	1

5. List all trainees enrolled in "Data Basics"

```sql
-- Get names and emails of trainees enrolled in the course titled "Data Basic
SELECT
    t.name AS trainee_name,
    t.email
FROM Enrollment AS e
JOIN Trainee AS t ON e.trainee_id = t.trainee_id
JOIN Course AS c ON e.course_id = c.course_id
WHERE c.title = 'Data Basics';
```

100 %

Results | Messages

trainee_name	email

6. Identify the course with the highest number of enrollments

```sql
-- Find the course with the most enrollments
SELECT TOP 1
    c.title AS course_title,
    COUNT(e.enrollment_id) AS total_enrollments
FROM Enrollment AS e
JOIN Course AS c ON e.course_id = c.course_id
GROUP BY c.title
ORDER BY total_enrollments DESC;
```

100 %

Results | Messages

	course_title	total_enrollments
1	Data Science Introduction	2

7. Display all schedules sorted by start date

```sql
-- List all schedules ordered by the start date
SELECT
    s.schedule_id,
    c.title AS course_title,
    s.start_date,
    s.end_date,
    s.time_slot
FROM Schedule AS s
JOIN Course AS c ON s.course_id = c.course_id
ORDER BY s.start_date ASC;
```

00 %

Results | Messages

	schedule_id	course_title	start_date	end_date	time_slot
1	1	Database Fundamentals	2025-07-01	2025-07-10	Morning
2	2	Web Development Basics	2025-07-05	2025-07-20	Evening
3	3	Data Science Introduction	2025-07-10	2025-07-25	Weekend
4	4	Advanced SQL Queries	2025-07-15	2025-07-22	Morning
5	5	Database Fundamentals	2025-08-01	2025-08-15	Evening

References

1. **Elmasri, R., & Navathe, S. B.** (2016). *Fundamentals of Database Systems* (7th ed.). Pearson Education.
   – A foundational book for understanding ER models, relational schema mapping, and normalization.
2. **Silberschatz, A., Korth, H. F., & Sudarshan, S.** (2020). *Database System Concepts* (7th ed.). McGraw-Hill.
   – Comprehensive coverage of SQL, relational databases, and query optimization.
3. **W3Schools**. (n.d.). *SQL Tutorial*. Retrieved from https://www.w3schools.com/sql/
   – Used for syntax examples and practical demonstrations of `SELECT`, `JOIN`, `GROUP BY`, and `INSERT`.
4. **Microsoft Learn**. (n.d.). *T-SQL Reference for SQL Server*. Retrieved from https://learn.microsoft.com/en-us/sql/t-sql
   – Reference for SQL Server-specific syntax like `TOP`, `GETDATE()`, `IDENTITY`, and constraints.

AI tools Used:

1. Chatgpt
2. ClaudeAi