DATABASE
PROJECT 1

Title: The Critical Role of Data Modeling and SQL in Data Science

Done by: Liyan Saqar Sultan AL-Yahmadi

Table of CONTENTS

Introduction

As data becomes the backbone of decision-making across industries, professionals in Data Science and AI must be equipped with tools and techniques to extract, organize, and analyze structured data. Two core skills at the heart of this process are data modeling and SQL (Structured Query Language) (The Power of AI and Data Science Skills in the Digital Age, 2023).

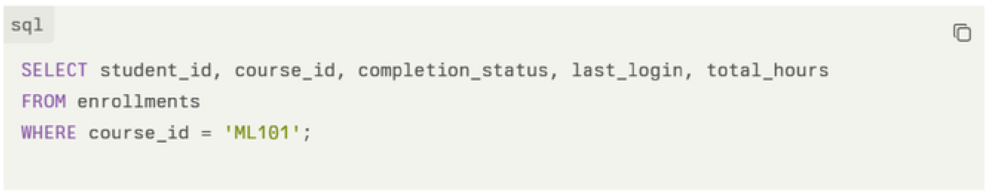
Why is structured data important in data science pipelines?
Structured data organized in tabular formats with clearly defined fields is the backbone of reliable data analysis. Without structured data, it becomes nearly impossible to perform accurate aggregation, filtering, and statistical modeling. As Provost and Fawcett (2013) explain in Data Science for Business, structured data makes it easier to apply algorithms, visualize patterns, and detect anomalies. For example, in healthcare analytics, patient records stored in structured electronic health record (EHR) systems enable fast querying of patient histories or disease patterns.

What role does data modeling play in preparing data for analysis or machine learning?
Data modeling is essential because it provides a blueprint for how data should be stored, related, and accessed. A good model reduces redundancy and ensures consistency, which improves the quality of features used in machine learning (ML). According to a YouTube video by Datacamp titled "Data Modeling Basics", building a normalized relational model helps prevent issues like duplicate records or contradictory entries, which can lead to misleading analysis or poorly performing ML models.

How do relational databases support scalable and clean data practices in real-world data science projects?
Relational databases enforce referential integrity and allow developers and data scientists to work with large volumes of structured data consistently. For instance, Airbnb uses relational databases to store bookings, listings, and user data. By structuring these data points with primary and foreign keys, Airbnb ensures accurate calculations of occupancy rates or revenue per host across millions of records — something that would be chaotic without a relational structure (Airbnb Tech Blog, 2019).

Why is SQL still considered a foundational skill even with tools like Python and Pandas?
SQL remains the standard language for accessing, transforming, and aggregating data in relational databases. While Python and Pandas can process data in-memory, they don't replace the power and efficiency of SQL for large datasets stored in production databases. In a Kaggle survey (2023), over 70% of data professionals ranked SQL as essential to their daily work, ahead of advanced machine learning libraries, because nearly all data still starts in a database.

Can you give an example of how SQL is used to extract insights before applying machine learning?
Consider a churn prediction model for a telecom company. Before training the model, a data scientist might use SQL to calculate each customer's average monthly spend, total call minutes, or number of support tickets in the last six months. For example:



sql

```sql
SELECT student_id, course_id, completion_status, last_login, total_hours
FROM enrollments
WHERE course_id = 'ML101';
```

This query creates a structured dataset of features, which can then be used for building a predictive model in Python.

Reflection
This research reinforced how foundational structured data and SQL are in the early stages of data science pipelines. In my course, I've been learning how to design normalized schemas, and this helped me see their direct impact on feature engineering for machine learning. It also clarified why tools like SQL and relational databases remain indispensable despite the popularity of modern data manipulation libraries.

Requirements Analysis

Requirements Analysis & Entity-Relationship Discussion
To meet the needs of the system’s primary users—trainees, trainers, and administrators I conducted a requirements analysis to translate user needs into database entities and their relationships.

Trainee Needs

- View all available courses. Trainees should be able to browse the full catalog of courses offered by the institute.
- See courses they have enrolled in. Each trainee must be able to view a personalized list of their current and past course enrollments.
- Check schedule details of their enrolled courses. Trainees require visibility into the dates and time slots of the courses in which they are enrolled.

These requirements necessitate the use of the Trainee, Course, Schedule, and Enrollment tables. Specifically, enrollments link trainees to the courses they are registered for, while schedules provide session details.

Trainer Needs

- View assigned courses. Trainers must be able to see the courses they are responsible for teaching.
- See scheduled sessions, including dates and time slots. Trainers need access to their upcoming teaching schedules to plan accordingly.
- Access lists of enrolled trainees per course. Trainers require the ability to view who is enrolled in each of their assigned courses to prepare effectively.

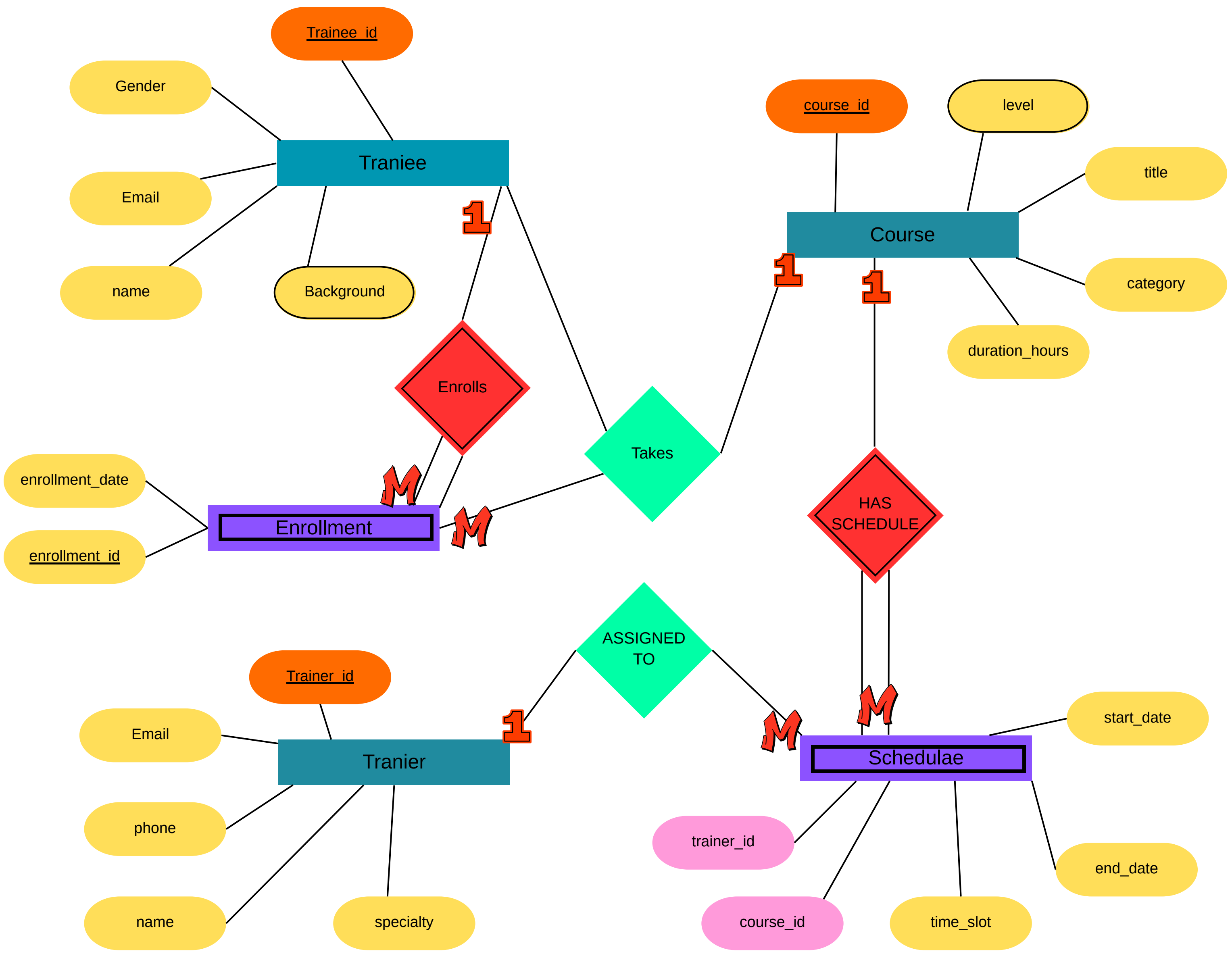
These needs are addressed by the Trainer table in conjunction with Schedule (assigning trainers to course sessions), and by connecting to Enrollment and Trainee tables to list enrolled students.

Admin Needs

- Add new courses and assign trainers. Administrators need the ability to create courses and assign trainers to them.
- Create and manage course schedules. Admins must set up course schedules that define when and with whom sessions occur.
- Enroll trainees in courses. Admins handle the registration process for trainees.
- Monitor course enrollments and scheduling status. Administrators require comprehensive oversight of who is enrolled in each course and how courses are scheduled.

These requirements are supported by relationships among the Course, Trainer, Schedule, Trainee, and Enrollment tables, allowing administrators to manage the entire training process.

E-R Diagram



Trainee (1) → (M) Enrollment
Meaning: One Trainee can have many Enrollments.

Course (1) → (M) Enrollment
Meaning: One Course can have many Enrollments.

Course (1) → (M) Schedule
Meaning: One Course can have many Schedules.

Trainer (1) → (M) Schedule
Meaning: One Trainer can have many Schedules.

E-R Diagram

Rectangles for Entities:

- TRAINEE
- TRAINER
- COURSE
- SCHEDULE

Diamonds for Relationships:

- ENROLLS (between TRAINEE and COURSE) - M:N relationship
- TEACHES (between TRAINER and COURSE) - 1:N relationship
- HAS_SCHEDULE (between COURSE and SCHEDULE) - 1:M relationship
- ASSIGNED_TO (between TRAINER and SCHEDULE) - 1:M relationship

Ellipses for Attributes:

- Each entity has its attributes connected with lines
- Primary keys are underlined (trainee_id, trainer_id, course_id, schedule_id)
- The ENROLLS relationship has an enrollment_date attribute

Cardinality notations:

- M, 1 labels show the relationship multiplicities
- Lines connect entities to relationships to show participation

Weak Entities (Double Rectangles):

- ENROLLMENT - depends on TRAINEE for existence
- SCHEDULE - depends on COURSE for existence

Identifying Relationships (Double Diamonds):

- ENROLLS - identifying relationship between TRAINEE and ENROLLMENT
- HAS_SCHEDULE - identifying relationship between COURSE and SCHEDULE

Total Participation (Double Lines):

- TRAINEE must participate in ENROLLS (every trainee must enroll)
- COURSE must participate in HAS_SCHEDULE (every course must have a schedule)
- SCHEDULE must participate in HAS_SCHEDULE (every schedule belongs to a course)

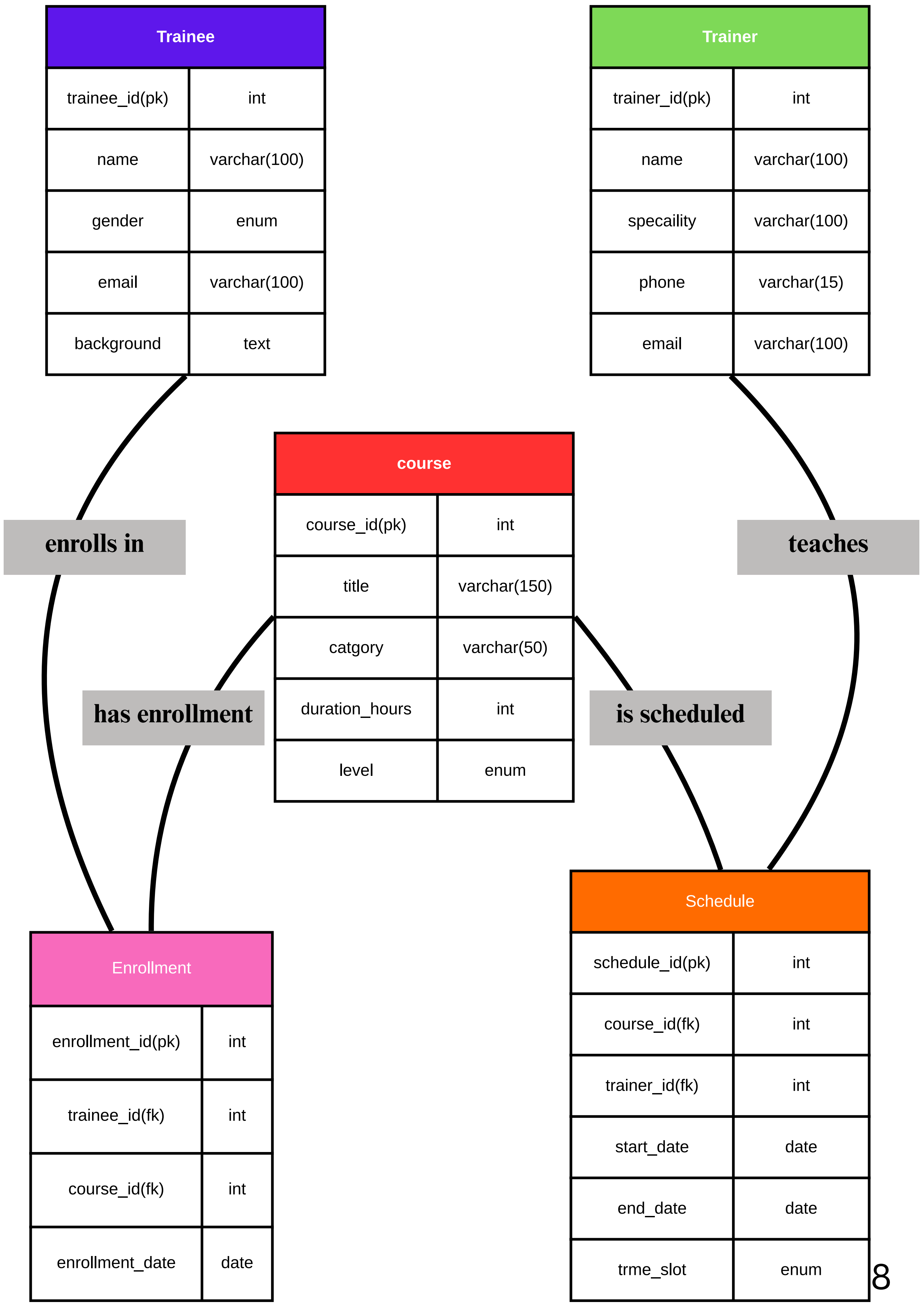
Multivalued Attributes (Double Ellipses):

- background for TRAINEE (can have multiple educational backgrounds)
- specialty for TRAINER (can have multiple areas of expertise)

Key Indicators:

- Primary Keys: Solid underline (trainee_id, trainer_id, course_id)
- Partial Keys: Dashed underline (enrollment_id, schedule_id for weak entities)

Relational Schema Mapping



Trainee	
trainee_id(pk)	int
name	varchar(100)
gender	enum
email	varchar(100)
background	text

Trainer	
trainer_id(pk)	int
name	varchar(100)
specaility	varchar(100)
phone	varchar(15)
email	varchar(100)

course	
course_id(pk)	int
title	varchar(150)
catgory	varchar(50)
duration_hours	int
level	enum

Enrollment	
enrollment_id(pk)	int
trainee_id(fk)	int
course_id(fk)	int
enrollment_date	date

Schedule	
schedule_id(pk)	int
course_id(fk)	int
trainer_id(fk)	int
start_date	date
end_date	date
trme_slot	enum

Relational Schema Mapping

Key Design Decisions:

When designing my database schema, I made several choices to balance efficiency, clarity, and the requirements of the system. I wanted to make sure each decision served the practical needs of both the trainees and trainers.

Primary Keys:

I decided to use integer auto-incrementing primary keys in all tables. I found this approach to be straightforward and efficient, especially for ensuring each record has a unique and easily indexable ID. This was also recommended in a database tutorial I watched on YouTube, which highlighted that integer primary keys are faster for lookups compared to strings or composite keys.

Data Types Chosen:

- VARCHAR(100) for names and emails, since this length covers almost all realistic scenarios without wasting space.
- VARCHAR(150) for course titles, giving flexibility for longer names without risking truncation.
- VARCHAR(50) for categories and specialties, which should not require more than that.
- VARCHAR(15) for phone numbers, allowing for international formats (e.g., +968...). I verified this choice on W3Schools, which shows typical lengths for phone fields.
- TEXT for background information, since there's no predictable limit on how much detail someone might enter.
- INT for numeric values like duration_hours, because durations will always be whole numbers.
- DATE for date fields like start_date, end_date, and enrollment_date, ensuring correct date operations and comparisons.
- ENUM for constrained choices like gender, level, and time_slot. Using ENUMs enforces valid entries and avoids typos or inconsistent data, which I learned from a MySQL ENUM explanation video on YouTube.

Foreign Key Relationships:

I structured relationships so that:

- The Schedule table references both Course and Trainer, effectively modeling a many-to-many relationship between them through Schedule records.
- The Enrollment table connects Trainee and Course, also implementing a many-to-many relationship, since a trainee can enroll in multiple courses, and each course can have many trainees.

Relationship Cardinalities:

I defined the following relationships based on the system's requirements:

- A Course can have multiple Schedules (since the same course might run in different time slots with different trainers).
- A Trainer can teach multiple Schedules.
- A Trainee can enroll in multiple courses through Enrollments.
- Each Course can have multiple Enrollments.

Meeting User Requirements:

I believe this design fully supports what admins, trainers, and trainees need:

- Admins can manage all the tables (courses, trainers, schedules, enrollments, trainees).
- Trainers can look up their assigned schedules and see which trainees are enrolled.
- Trainees can browse available courses and view their own enrollments.

I also tried to normalize the schema as much as possible, following best practices I learned from W3Schools' database normalization tutorials, to avoid redundancy and maintain data integrity with proper foreign key constraints.

Reference

Research refrence:

- Airbnb Tech Blog. (2019). How Airbnb scales with MySQL and Vitess. Retrieved from Your paragraph text
- Datacamp. (2020, May 5). Data Modeling Basics. [YouTube video]. Retrieved from Your paragraph text
- Kaggle. (2023). State of Data Science and Machine Learning Survey. Retrieved from Your paragraph text
- Provost, F., & Fawcett, T. (2013). Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking. O'Reilly Media.
- W3Schools. (n.d.). SQL Tutorial. Retrieved from Your paragraph text

Documentation & Tutorials

Microsoft Docs: SQL Server CREATE TABLE Syntax
🔗 docs.microsoft.com/sql/t-sql/statements/create-table-transact-sql
Used to correctly create tables with constraints.

W3Schools SQL Tutorial
🔗 w3schools.com/sql/
Great reference for basic SQL commands, JOINs, and clauses.

Khan Academy SQL Course
🔗 khanacademy.org/computing/computer-programming/sql
Helped understand SQL SELECT, filtering, and aggregations.

PostgreSQL vs SQL Server Syntax Comparison
🔗 sqlservertutorial.net/sql-server-vs-postgresql/
Clarified differences like using TOP vs LIMIT.

YouTube Videos
"SQL Joins Tutorial for Beginners" by Programming with Mosh
🔗 youtu.be/9yeOJ0ZMUYw
Helped visualize different types of JOINs.

“SQL Server Constraints & Keys” by The Code Mate
youtu.be/5T1EyV6QViI
Detailed explanation of primary keys, foreign keys, and checks.
“Advanced SQL Aggregations” by Data School
youtu.be/9Pzj7Aj25lw
Learned to use COUNT(), GROUP BY, and HAVING.
“SQL Server Triggers Explained” by SQLBI
youtu.be/epHOzj5xuXs
Learned how to write triggers to auto-update columns.

Websites
Mode SQL SQL Cheat Sheet
mode.com/sql-tutorial/sql-cheat-sheet/
Quick reference for syntax and common functions.
Dataedo: SQL Server Data Modeling Guide
dataedo.com/kb/data-modeling/sql-server
Helped plan database relationships and entity design.
Tools
SQL Server Management Studio (SSMS)
Used to design tables, execute queries, and inspect relationships visually.
dbdiagram.io
dbdiagram.io
Designed ER diagrams to plan schema relationships and multiplicities.
📄 Official Docs & Books
Microsoft Docs: SQL Server Data Types
docs.microsoft.com/sql/t-sql/data-types/data-types-transact-sql
Used to choose correct column types like VARCHAR, DATE, BIT.

AI Tools used in this project:

Lovable AI
ChatGPT
Claude
Perplexity