Databases & Web Programming

Sample Paper 1 — Practice Exam

University of Birmingham

School of Computer Science

Module:	Databases & Web Programming
Exam:	Sample Paper 1 (Practice)
Duration:	2 hours
Total marks:	100

Instructions to candidates

- Answer **all three** questions.
- Marks for each part are shown in **red** on the right.
- Write SQL keywords in UPPERCASE. SQL statements must end with a semicolon.
- For JDL, follow lecture conventions (entity names PascalCase, fields camelCase).
- Where a curl command is asked for, assume the server is on `localhost:8080` unless told otherwise.

Question 1 — Databases

Total: 30 marks

(a) Anomalies and Constraints

A startup keeps its customer orders in a single spreadsheet. Each row stores the customer's name, phone number, every product they ordered (comma-separated in one cell), the unit price of each product, and the total. When a customer changes their phone number, the team must search and replace it across dozens of rows.

(i) Identify **two** different anomalies present in this design and explain each in the context of the scenario.

[4 marks]

(ii) Name **three** SQL integrity constraints you would apply when migrating this spreadsheet into a relational table. For each, give one column you would apply it to and explain why.

[6 marks]

(b) Normalisation to 3NF

Consider the following unnormalised table of student module enrolments. Each module is taught by exactly one lecturer, and the lecturer's office is determined by the lecturer (not the module).

StudentID	StudentName	ModuleCode	ModuleTitle	Lecturer	LecturerOffice	Grade
S101	Alice Khan	CS101	Intro to CS	Dr. Patel	CSB-204	72
S101	Alice Khan	CS210	Databases	Dr. Yusuf	CSB-310	68
S102	Ben Akin	CS101	Intro to CS	Dr. Patel	CSB-204	81
S102	Ben Akin	CS220	Web Prog	Dr. Patel	CSB-204	74
...	...	...	...	...	...	...

(i) Decompose the table into **3NF**. For each resulting table, list its columns and clearly mark Primary Keys with **(PK)** and Foreign Keys with **(FK)**.

[8 marks]

(ii) Identify one **transitive functional dependency** in the original table and briefly explain why it violates 3NF.

[2 marks]

(c) SQL Queries

Two tables exist in a streaming-service database:

```
Users(UserID PK, FullName, Country, JoinDate)
Streams(StreamID PK, UserID FK, ShowTitle, MinutesWatched, StreamDate)
```

(i) Write an SQL **CREATE TABLE** statement for **Streams**. Include the primary key and a foreign-key constraint referencing `Users`. `MinutesWatched` must default to `0`.

[4 marks]

(ii) Write a single SQL query that returns each user's **FullName** and the **total minutes they have watched**, but only for users whose total exceeds 500 minutes. Sort the results from highest total to lowest. Use JOIN, GROUP BY and HAVING.

[6 marks]

Question 2 — Web Development Principles

Total: 30 marks

(a) REST Design Conventions

A colleague proposes the following endpoints for a library application:

```
GET  /api/getBookById?id=42
POST /api/deleteBook/42
GET  /api/books_all
```

Identify **three** distinct REST design conventions these endpoints violate. For each, explain what is wrong and rewrite the endpoint correctly.

[9 marks]

(b) Architectures & the Persistence Layer

(i) Explain in 2-3 sentences why a **REST** architecture scales better than a classic **CGI-bin** approach for a high-traffic web app.

[4 marks]

(ii) In a Spring Data JPA Repository you find:

```
List<Book> findByAuthorAndPublishedYearGreaterThan(String author, Integer
year);
```

Explain (1) what this method returns, (2) how Spring Data JPA implements the body without you writing SQL, and (3) which annotation you would add to the `Book` class to map it to a database table called `library_book`.

[8 marks]

(c) Authentication vs Authorization

A library app distinguishes between a regular **MEMBER** and a **LIBRARIAN**. Only librarians may delete a book record.

(i) Define **authentication** and **authorization**. Make the distinction concrete using the library scenario.

[4 marks]

(ii) A member sends a `DELETE /api/books/42` request and is rejected. Which HTTP status code should the server return, and why is it different from the code returned when the request has no token at all?

[5 marks]

Question 3 — Web Development Practice

Total: 40 marks

(a) JDL Modelling

You are modelling a music-streaming app. Write JDL that:

- Defines an **Album** entity with: `title` (String, required, max 100 chars), `releaseDate` (LocalDate), `coverArt` (ImageBlob).
- Defines a **Track** entity with: `title` (String, required), `durationSeconds` (Integer, min 1), `plays` (Long).
- Each Album has many Tracks; each Track belongs to exactly one Album.
- Each Album has exactly one owning User (the uploader), linked to the built-in **User** entity.

[15 marks]

(b) cURL & HTTP Methods

Using the model from (a), write the cURL command(s) that would:

(i) **PATCH** the `plays` field of Track ID 88 to `1500`. Send JSON and include the `Authorization: Bearer <TOKEN>` header.

[5 marks]

(ii) The developer accidentally sends `GET /api/tracks/9999` for a Track that doesn't exist. What 4xx code does the server return, and how does it differ from the code returned when the user simply isn't authenticated?

[5 marks]

(c) Versioning & Deprecation

The team is releasing a v2 API which renames `plays` to `playCount`.

(i) Show how you would change the `@RequestMapping` on the new `TrackResource` so v2 lives side-by-side with v1.

[5 marks]

(ii) The team will keep v1 running for 9 months. Name this period and describe **two** concrete strategies you'd use to push existing clients to migrate before v1 is removed.

[5 marks]

(d) Security & Custom Repository Query

Only the User who uploaded an Album may delete it.

(i) In the `AlbumResource` delete handler, sketch (in Java) the lines you would add to (1) get the logged-in user's login, (2) load the Album, and (3) reject with HTTP 403 if the Album's owner login does not match.

[5 marks]

— End of Sample Paper 1 —