GAC017 Computing III: Data Science and Web Apps

GAC Computing

What this module is, and how it is marked

GAC017 is the Level III computing module: what sits behind a website. Six units cover back-end JavaScript, databases, SQL, connecting the two, and a first look at data science.

Assessment is entirely practical — a **database** 数据库 you design and build, a **web app** 网络应用 that talks to it, a **data analysis project** 数据分析项目, and coursework — usually spread across most of the semester rather than concentrated at the end.

- The work is judged on whether it **runs and answers a question**, not on how much code there is.
- △ Design the database before you write a query. Almost every problem later is a table problem wearing a query's clothes.

Every SQL example here runs in the site's own code playground, and the SQL reference there is the fuller version of these notes.

JavaScript for a web app's back end

- The **front end** 前端 runs in the browser; the **back end** 后端 runs on a server and holds what the browser must not.
- A **server** 服务器 receives a **request** 请求 and returns a **response** 响应. That loop is the whole architecture.
- An **API** 应用程序接口 is the agreed shape of those requests and responses.
- △ Anything secret — a password, a key — belongs on the back end only. Code in a browser is readable by everyone who visits.

Back-end programming

- A **route** 路由 maps a URL to code that runs when that URL is requested.
- **Validate input** 校验输入 on the server. Checking in the browser is a convenience for honest users, not a defence.
- **Never build a query by joining strings** with user input. That is how **SQL injection** SQL 注入 happens, and it is the security failure this unit exists to prevent.
- Return an honest **status**: 200 for success, 400 for a bad request, 404 for something that is not there.

Introduction to databases

- A **relational database** 关系数据库 stores data in **tables** 表 of rows and columns.
- A **primary key** 主键 identifies a row uniquely. A **foreign key** 外键 points at another table's primary key, and that pointer is the relationship.
- **Normalisation** 规范化 removes duplicated data so one fact lives in one place.
- Design by asking what the **entities** 实体 are, then what connects them.

Worked example. A school wants to store students, courses and who takes what.

Two tables cannot do it: a student takes many courses and a course has many students. The many-to-many needs a third table — enrolments — whose rows are (student, course) pairs.

Recognising that this third table is required is the single most useful database idea in the module, and it is where most first designs go wrong.

SQL for back-end programming

- **SQL** 结构化查询语言 asks a database questions. `SELECT … FROM … WHERE …` is the core.
- **JOIN** 连接 combines rows from two tables on a matching key.
- **GROUP BY** 分组 with `COUNT`, `SUM` or `AVG` answers "how many per…" questions.
- △ `WHERE` filters rows before grouping; `HAVING` filters groups after. Using the wrong one is the classic SQL error, and it usually returns a plausible wrong answer.

Worked example. Which courses have more than 20 students?

```
SELECT c.title, COUNT(*) AS students
FROM enrolments e
JOIN courses c ON c.id = e.course_id
GROUP BY c.title
HAVING COUNT(*) > 20;
```

The count is a property of the group, so the filter is `HAVING`. Written with `WHERE` it does not run — and when a similar mistake does run, it silently answers a different question.

Connecting JavaScript with SQL

- The back end takes a request, runs a **parameterised query** 参数化查询, and returns the rows as data, usually **JSON** 数据交换格式.
- Parameterised means the values travel separately from the query text. That is what makes injection impossible rather than unlikely.
- Handle the empty case. A query that returns no rows is normal, and a page that breaks on it is not finished.

Data science

- **Data science** 数据科学 turns data into a decision, and most of the work is before the analysis: cleaning, joining, and checking what the data can support.
- **Descriptive statistics** 描述性统计 summarise; a **visualisation** 可视化 shows shape; neither proves a cause.
- **Correlation is not causation** 相关不等于因果 — the sentence every data project needs and most omit.
- State the **limitations** of your dataset. A project that names what its data cannot show scores above one that quietly overclaims.