MODULE N · OCR J277 2.2.3 · CAMBRIDGE 0478 9.1-9.4 (WITH 7.5(A))

Databases and SQL

Tables, records and fields, **data types**, **validation**, **primary keys** and **SQL**: SELECT, FROM, WHERE, then ORDER BY, SUM, COUNT, AND and OR. **OCR J277: you only need cards 01, 05 and 07**, without the IGCSE-only rows. Examples use the made-up table PRODUCT (6 fields, 8 records). Spec versions as checked in November 2026.

01 Tables, records, fields (N1) · both boards

Table Stores data about one type of item, in rows and columns.

Record One row: all the data about one item (e.g. product P04).

Field One column: one item of data for every record (e.g. Price). Every record has the same fields.

Counting PRODUCT: 6 fields, 8 records. Heading row is not a record. Add a field: 7 fields, still 8 records.

IGCSE only Define a single table from data storage requirements: field names (no spaces), data types, validation, primary key.

02 Data types (N2) IGCSE only

Text/alphanumeric Letters, digits, symbols: Name, ProductID, phone number.

Character One character: Size 'S', 'M' or 'L'.

Boolean Two values: Vegan TRUE/FALSE.

Integer / real Whole number: Stock. Decimal: Price.

Date/time A date and/or time: DateBaked.

Phone number Text: keeps the leading 0, may hold spaces, no calculations. Never "string" for a database field.

03 Validation (N2) IGCSE only

Purpose Automatic check that data entered is reasonable and meets a rule; rejected if not. Cannot prove it is correct.

Range Between limits: Stock 0 to 100 inclusive.

Length Number of characters: Name no more than 30.

Type Right data type: Price must be a number.

Presence Not left empty.

Format Follows a pattern: P then two digits (P04).

Check digit Last digit calculated from the others; recalculated on entry and compared; no match: rejected.

04 Primary keys (N3) IGCSE only

Primary key A field that uniquely identifies each record. No two records share a value.

Purpose Find, update or refer to one record without mixing it up with another.

Choosing Values that can never repeat, for every record that could ever be stored: ProductID.

Unsuitable Names, categories, prices, dates: two records could share them (Bread appears 3 times).

None unique? Add an ID field. A key can hold letters: P04 is fine.

05 SELECT, FROM, WHERE (N4) · both boards

SELECT Lists the fields to show, separated by commas. `SELECT *` = all fields.

FROM Names the table to search.

WHERE The condition a record must meet: = <> < > <= >=

Quotes Text in quotes: `Category = 'Bread'`. Numbers not: `Stock >= 20`.

GCSE only Only these three are listed. SQL tests equality with =, not ERL's ==.

IGCSE only End with ; and state the output values exactly, in SELECT order, no commas or quotes.

06 ORDER BY, SUM, COUNT (N5) IGCSE only

ORDER BY `ORDER BY Stock ASC` smallest first; `DESC` largest first. Ascending is the default. Changes order only, not which records.

SUM `SUM(Stock)` adds up a numeric field for the matching records.

COUNT `COUNT(ProductID)` gives the number of matching records.

AND / OR AND: both conditions true. OR: at least one. (OCR WHERE conditions can use AND and OR too.)

Order SELECT, FROM, WHERE, ORDER BY.

07 2D arrays and records (N1) GCSE only

Records A way to store data in a program: related items of data about one thing kept together.

2D array Emulates a database table: each row is a record, each column a field.

Field type OCR can ask the best data type for a field: integer, real, Boolean, character or string (J277 2.2.2).

Indexing From 0, [row, column]: `products[3,1]` is "Lemon tart" (record P04, Name field).

Search Loop over every row; check one column; output another: Bread rows give Sourdough loaf, Seeded rye, Bagel.

08 Worked outputs on PRODUCT

Both boards `SELECT Name, Price FROM PRODUCT WHERE Price < 2.00;` gives Cheese scone 1.60, Bagel 1.10, Fruit flapjack 1.90.

IGCSE only Sweet items `ORDER BY Stock DESC`: Cinnamon bun 24, Fruit flapjack 18, Lemon tart 15.

IGCSE only `COUNT(ProductID) ... WHERE Vegan = TRUE` gives 4. `SUM(Stock) ... WHERE Category = 'Bread'` gives 12 + 8 + 40 = 60.

01
A **record is a row**, a **field is a column**. Do not count the heading row.

02
Separate SELECT fields with **commas**, never AND. Put **text in quotes**.

03
IGCSE: a primary key must be unique for **every record that could ever be stored**.

04
IGCSE: **COUNT counts** records; **SUM adds** values. ORDER BY never filters.