

Databases

IGCSE Computer Science 0478 · Topic 9 revision notes

Everything from the 4 lessons of this topic on a few pages: the key idea of each lesson, the diagram to remember, the points to learn and the traps examiners set. Read it, cover it, test yourself.

- | | |
|--|---|
| 1. <u>Single-table databases: fields, records and data types</u> | 3. <u>SQL: SELECT, FROM, WHERE and ORDER BY</u> |
| 2. <u>Primary keys</u> | 4. <u>SQL: SUM, COUNT, AND and OR</u> |

thelucidstem.com

1 Single-table databases: fields, records and data types

Syllabus 9 (1-3)

A **database** is a structured collection of data that allows people to extract information. In a **single-table database** all the data is in one **table**. Each **field** (column) holds one item of data, such as Name or Weight. Each **record** (row) holds all the data about one person or thing. Every field is given a **data type**: **text/alphanumeric**, **character**, **Boolean**, **integer**, **real** or **date/time**. **Validation** rules check that data entered into a field is sensible.

table PETS

PetID	Name	Species	Weight	Vaccinated
P01	Bella	Dog	12.5	TRUE
P02	Tom	Cat	4.2	FALSE
P03	Nibbles	Rabbit	1.8	TRUE
P04	Max	Dog	30.1	TRUE

a field (column): one item of data

a record (row): all the data about one pet

Fig. 1: the PETS table has 5 fields and 4 records

Worked example: Phone numbers

Suggest a data type for a field that stores a telephone number such as 01632 960 123, and justify it.

1. The number starts with a 0, and an integer would drop the leading zero.
2. It may contain spaces or a + sign, and no arithmetic is ever done on it.

Answer: Text / alphanumeric

Columns and rows.

- ✓ **Field:** a column, one item of data (e.g. Weight). **Record:** a row, all the data about one item.
- ✓ **Counting:** do not count the row of field names as a record.
- ✓ **Data types:** text/alphanumeric, character, Boolean, integer, real, date/time. Never write "number".
- ✓ **Justify:** whole number = integer; decimal = real; yes/no = Boolean; leading zero or symbols = text.

Watch out!

Do not mix the two words up. A **field** is a column (one kind of data); a **record** is a row (one item). Swapping them loses the mark.

"Number" and "string" are not accepted as answers. Give the exact type: integer or real, and text/alphanumeric.

A range check on Weight would still accept 21.5 when the dog really weighs 12.5. Validation checks that data is *sensible*, not that it is *accurate*.

2 Primary keys

Syllabus 9 (1-3)

A **primary key** is a field that **uniquely identifies each record** in a table. Its value must be **different for every record** and must **never be empty**. Names, towns and dates are poor choices because values can repeat. When no existing field is guaranteed to be unique, the database designer adds an **ID field**, such as MemberID or BookID, to act as the primary key.

which field could be the primary key?

MemberID	FirstName	Surname	Town
M001	Ana	Silva	Leeds
M002	Ben	Silva	York
M003	Ana	Khan	Leeds
M004	Chen	Wu	York

✓ unique

X values repeat, so not unique

a primary key value is never repeated and never empty

Fig. 1: only MemberID has a different value in every record, so it is the primary key

Worked example: Members table

Identify the most suitable primary key for the table above and explain why each other field is not suitable.

1. FirstName repeats (Ana twice), Surname repeats (Silva twice) and Town repeats (Leeds and York twice each).
2. DateJoined happens to be different in these four records, but two people could join on the same day, so it is not guaranteed unique.
3. MemberID is different for every member and is given to every member when they join.

Answer: MemberID

Can it ever repeat?

- ✓ **Purpose:** a primary key uniquely identifies each record.
- ✓ **Test each field:** could two records ever share a value? Could a value be missing?
- ✓ **Poor keys:** names, towns, dates, prices, genres: they repeat.
- ✓ **Good keys:** ID fields (StudentID, OrderNo) and unique codes (ISBN, car registration).

Watch out!

"A primary key is the first field" or "the most important field" earns nothing. The mark is for **uniquely identifies each record**.

Do not choose a field just because its values are all different in the sample shown. It must be unique for every record that could ever be added.

An email address may look unique, but people change addresses and some share family accounts. In an exam, an ID field is the safest choice when one is given.

3 SQL: SELECT, FROM, WHERE and ORDER BY

Syllabus 9 (4)

SQL (structured query language) is used to search a database. **SELECT** lists the **fields** to show. **FROM** names the **table**. **WHERE** gives a condition, so only the **records** that meet it are shown. **ORDER BY** sorts the results by a field, **ASC** (ascending: A to Z, smallest first) or **DESC** (descending: Z to A, largest first). To find the output, check every record against the **WHERE** condition, keep the matching ones, show only the **SELECTed** fields, then sort.

the four parts of a query

SELECT	Title, Price	-----	which fields to show
FROM	Games	-----	which table
WHERE	Genre = 'Racing'	-	which records
ORDER BY	Price DESC;	-----	in what order

SELECT and FROM are always needed; WHERE and ORDER BY are optional

Fig. 1: each line of a query does one job

Worked example: Text condition

Give the output of this query on the **GAMES** table.

```
SELECT Title, Price
FROM Games
WHERE Genre = 'Racing';
```

1. Check each record's Genre: G01 Racing (yes), G02 Adventure (no), G03 Puzzle (no), G04 Racing (yes), G05 Adventure (no), G06 Puzzle (no).
2. Keep G01 and G04, and show only Title and Price.

Answer: Star Racer 25.00; Speed Kings 35.00

Four clauses, one order.

- ✓ **SELECT** fields, **FROM** table, **WHERE** condition, **ORDER BY** field ASC or DESC.
- ✓ **Output:** test every record against **WHERE**, one row at a time, and tick the ones that match.
- ✓ **Then** show only the **SELECTed** fields, and sort if there is an **ORDER BY**.
- ✓ **Watch** < versus <=, and put text values in quotes.

Watch out!

SELECT names fields, not records. "SELECT Racing" is wrong: the word Racing is a value in the Genre field, so it belongs in a **WHERE** condition.

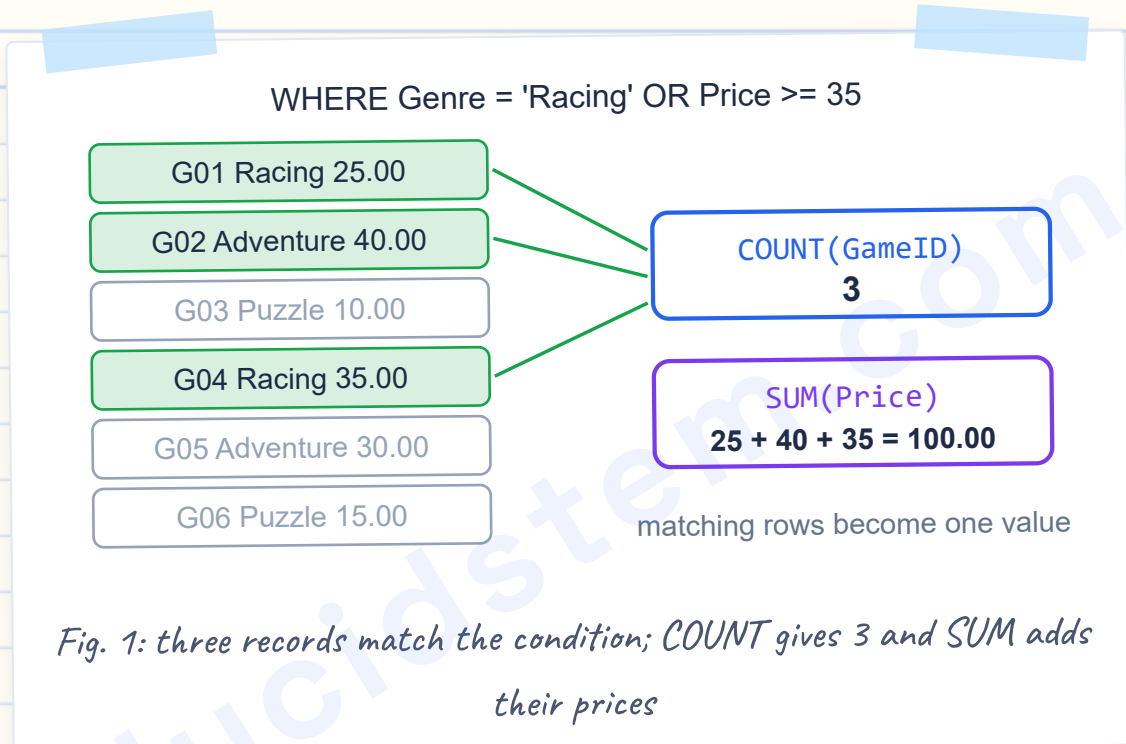
Read < and <= carefully. **WHERE** Price > 30 would leave out Dragon Isle at exactly 30.00; **WHERE** Price >= 30 includes it.

The keywords must be in this order: **SELECT**, **FROM**, **WHERE**, **ORDER BY**. And **ORDER BY** Price **DESC** sorts the output, it does not choose which records appear.

4 SQL: SUM, COUNT, AND and OR

Syllabus 9 (4)

SUM(field) adds up the values in a numeric field for every record that matches the WHERE condition. **COUNT(field)** gives the **number of records** that match. Both return a single value, not a list. In a WHERE clause, **AND** keeps a record only if **both** conditions are true; **OR** keeps a record if **at least one** condition is true. A record that meets both OR conditions is still output only once.



Worked example: SUM

Give the output of this query.

```
SELECT SUM(Price)
FROM Games
WHERE Genre = 'Adventure';
```

1. Find the Adventure records: G02 (40.00) and G05 (30.00).
2. Add their prices: 40.00 + 30.00.

Answer: 70.00

One record at a time.

- ✓ **SUM(field)** adds the values; **COUNT(field)** counts the matching records. Each gives one value.
- ✓ **AND:** both conditions true. **OR:** at least one true; a record is never output twice.
- ✓ **Full conditions:** Price > 10 AND Price < 30, never Price > 10 AND < 30.
- ✓ **Method:** tick each record against each condition, then apply SUM, COUNT or ORDER BY.

Watch out!

SUM adds the values; COUNT counts the records. COUNT(Price) on the whole GAMES table gives 6, not 155.00.

AND gives the same or fewer records than either condition alone; OR gives the same or more. If your AND answer has more rows than one of its conditions, recheck it.

When asked for the output of a SUM or COUNT query, give one value. Listing the matching records instead earns no mark.