30 Verified SQL Solutions

Notes for SHS | Topic 64 | Fully fictional Global Student Learning Resources Database

01. List all subject names alphabetically.

```
SELECT subject_name FROM Subjects ORDER BY subject_name;
```

Verified result: 8 row(s). Example: [['Astronomy'], ['Biology']]

02. Show student codes and study levels for the first five student IDs.

```
SELECT student_code,study_level FROM Students ORDER BY student_id LIMIT 5;
```

Verified result: 5 row(s). Example: [['STU-001', 'First-year'], ['STU-002', 'Secondary']]

03. Find resources labelled Beginner.

```
SELECT resource_title FROM Resources WHERE difficulty_level='Beginner' ORDER BY resource_title;
```

Verified result: 10 row(s). Example: [['Balancing Chemical Equations'], ['Functions Practice']]

04. List distinct resource types.

```
SELECT DISTINCT resource_type FROM Resources ORDER BY resource_type;
```

Verified result: 4 row(s). Example: [['Lesson'], ['Quiz']]

05. Find the five longest estimated resource durations.

```
SELECT resource_title,estimated_minutes FROM Resources ORDER BY estimated_minutes DESC,resource_title
LIMIT 5;
```

Verified result: 5 row(s). Example: [['Functions Practice', 60], ['Genetics Revision', 60]]

06. Find resources with 30–45 estimated minutes inclusive.

```
SELECT resource_title,estimated_minutes FROM Resources WHERE estimated_minutes BETWEEN 30 AND 45 ORDER
BY estimated_minutes,resource_title;
```

Verified result: 13 row(s). Example: [['Cells and Microscopes', 30], ['Data Tables and SQL', 30]]

07. List resources whose titles contain “data” (SQLite default LIKE).

```
SELECT resource_title FROM Resources WHERE resource_title LIKE '%data%' ORDER BY resource_title;
```

Verified result: 1 row(s). Example: [['Data Tables and SQL']]

08. Count recorded usage events without a measured duration.

```
SELECT COUNT(*) AS unknown_duration FROM ResourceUsage WHERE minutes_spent IS NULL;
```

Verified result: 1 row(s). Example: [[9]]

09. List the first four usage IDs with nonmissing durations.

```
SELECT usage_id,minutes_spent FROM ResourceUsage WHERE minutes_spent IS NOT NULL ORDER BY usage_id
LIMIT 4;
```

Verified result: 4 row(s). Example: [[1, 15], [2, 35]]

10. Find the minimum, maximum and average estimated resource length.

```
SELECT MIN(estimated_minutes) AS min_minutes,MAX(estimated_minutes) AS
max_minutes,ROUND(AVG(estimated_minutes),2) AS mean_minutes FROM Resources;
```

Verified result: 1 row(s). Example: [[15, 60, 37.26]]

11. Count resources by difficulty level.

```
SELECT difficulty_level,COUNT(*) AS total FROM Resources GROUP BY difficulty_level ORDER BY
difficulty_level;
```

Verified result: 3 row(s). Example: [['Advanced', 10], ['Beginner', 10]]

12. Find subjects with at least four resources.

```
SELECT s.subject_name,COUNT(r.resource_id) AS n FROM Subjects s LEFT JOIN Resources r ON
r.subject_id=s.subject_id GROUP BY s.subject_id HAVING COUNT(r.resource_id)>=4 ORDER BY n
DESC,s.subject_name;
```

Verified result: 7 row(s). Example: [['Biology', 5], ['Computer Science', 5]]

13. Match the first six resources with subject names.

```
SELECT r.resource_title,s.subject_name FROM Resources r INNER JOIN Subjects s ON
r.subject_id=s.subject_id ORDER BY r.resource_id LIMIT 6;
```

Verified result: 6 row(s). Example: [['Cells and Microscopes', 'Biology'], ['Photosynthesis Practice', 'Biology']]

14. Retain every subject, including one with no resource.

```
SELECT s.subject_name,COUNT(r.resource_id) AS n FROM Subjects s LEFT JOIN Resources r ON
s.subject_id=r.subject_id GROUP BY s.subject_id ORDER BY s.subject_id;
```

Verified result: 8 row(s). Example: [['Biology', 5], ['Chemistry', 4]]

15. Count how many distinct students used each resource, top five.

```
SELECT r.resource_title,COUNT(DISTINCT u.student_id) AS learners FROM Resources r LEFT JOIN
ResourceUsage u ON r.resource_id=u.resource_id GROUP BY r.resource_id ORDER BY learners
DESC,r.resource_title LIMIT 5;
```

Verified result: 5 row(s). Example: [['Functions Practice', 9], ['Introduction to Algorithms', 8]]

16. Count completed sessions per student; show top five.

```
SELECT st.student_code,SUM(u.completed) AS completions FROM Students st JOIN ResourceUsage u ON
st.student_id=u.student_id GROUP BY st.student_id ORDER BY completions DESC,st.student_code LIMIT 5;
```

Verified result: 5 row(s). Example: [['STU-024', 9], ['STU-002', 8]]

17. Show average measured duration by subject, excluding missing durations.

```
SELECT s.subject_name,ROUND(AVG(u.minutes_spent),1) AS avg_mins FROM Subjects s JOIN Resources r ON
s.subject_id=r.subject_id JOIN ResourceUsage u ON r.resource_id=u.resource_id GROUP BY s.subject_id
ORDER BY avg_mins DESC,s.subject_name;
```

Verified result: 7 row(s). Example: [['Geography', 61.1], ['Biology', 59.8]]

18. Find students who accessed at least four distinct resources.

```
SELECT st.student_code,COUNT(DISTINCT u.resource_id) AS n FROM Students st JOIN ResourceUsage u ON
u.student_id=st.student_id GROUP BY st.student_id HAVING COUNT(DISTINCT u.resource_id)>=4 ORDER BY n
DESC,st.student_code;
```

Verified result: 22 row(s). Example: [['STU-004', 11], ['STU-024', 9]]

19. Count all events versus distinct learners.

```
SELECT COUNT(*) AS events,COUNT(DISTINCT student_id) AS learners FROM ResourceUsage;
```

Verified result: 1 row(s). Example: [[160, 24]]

20. Count completed and incomplete events by month.

```
SELECT strftime('%Y-%m',access_date) AS month,SUM(completed=1) AS completed,SUM(completed=0) AS
incomplete FROM ResourceUsage GROUP BY month ORDER BY month;
```

Verified result: 5 row(s). Example: [['2026-01', 24, 9], ['2026-02', 25, 11]]

21. List resources longer than the overall average estimated length.

```
SELECT resource_title,estimated_minutes FROM Resources WHERE estimated_minutes>(SELECT
AVG(estimated_minutes) FROM Resources) ORDER BY estimated_minutes DESC,resource_title;
```

Verified result: 15 row(s). Example: [['Functions Practice', 60], ['Genetics Revision', 60]]

22. Use a CTE to show five most accessed subjects.

```
WITH x AS (SELECT s.subject_name,COUNT(u.usage_id) AS events FROM Subjects s LEFT JOIN Resources r ON
r.subject_id=s.subject_id LEFT JOIN ResourceUsage u ON u.resource_id=r.resource_id GROUP BY
s.subject_id) SELECT * FROM x ORDER BY events DESC,subject_name LIMIT 5;
```

Verified result: 5 row(s). Example: [['Mathematics', 32], ['Computer Science', 31]]

23. Rank resources within each subject by access count, show three rows.

```
WITH x AS (SELECT s.subject_name,r.resource_title,COUNT(u.usage_id) AS visits FROM Resources r JOIN
Subjects s ON s.subject_id=r.subject_id LEFT JOIN ResourceUsage u ON u.resource_id=r.resource_id GROUP
BY r.resource_id) SELECT subject_name,resource_title,visits,ROW_NUMBER() OVER (PARTITION BY
subject_name ORDER BY visits DESC,resource_title) AS position FROM x ORDER BY subject_name,position
LIMIT 3;
```

Verified result: 3 row(s). Example: [['Biology', 'Genetics Revision', 8, 1], ['Biology', 'Photosynthesis Practice', 6, 2]]

24. Find resources with zero access events.

```
SELECT r.resource_title FROM Resources r LEFT JOIN ResourceUsage u ON r.resource_id=u.resource_id
WHERE u.usage_id IS NULL ORDER BY r.resource_id;
```

Verified result: 1 row(s). Example: [['Market Structures']]

25. List total accesses by month.

```
SELECT strftime('%Y-%m',access_date) AS month,COUNT(*) AS visits FROM ResourceUsage GROUP BY month
ORDER BY month;
```

Verified result: 5 row(s). Example: [['2026-01', 33], ['2026-02', 36]]

26. Check duplicate student codes.

```
SELECT student_code,COUNT(*) AS n FROM Students GROUP BY student_code HAVING COUNT(*)>1;
```

Verified result: 0 row(s). Example: []

27. Check invalid negative or missing recorded minutes.

```
SELECT SUM(minutes_spent IS NULL) AS missing_minutes,SUM(minutes_spent<0) AS negative_minutes FROM
ResourceUsage;
```

Verified result: 1 row(s). Example: [[9, 0]]

28. Show top resource in each subject using ROW_NUMBER.

```
WITH x AS(SELECT s.subject_name,r.resource_title,COUNT(u.usage_id) AS n FROM Resources r JOIN Subjects
s ON s.subject_id=r.subject_id LEFT JOIN ResourceUsage u ON u.resource_id=r.resource_id GROUP BY
r.resource_id),z AS(SELECT *,ROW_NUMBER() OVER (PARTITION BY subject_name ORDER BY n
DESC,resource_title) AS rn FROM x) SELECT subject_name,resource_title,n FROM z WHERE rn=1 ORDER BY
subject_name;
```

Verified result: 7 row(s). Example: [['Biology', 'Genetics Revision', 8], ['Chemistry', 'Atomic Structure Quiz', 7]]

29. Count how many distinct students completed sessions in each subject.

```
SELECT s.subject_name,COUNT(DISTINCT CASE WHEN u.completed=1 THEN u.student_id END) AS completing_students FROM Subjects s LEFT JOIN Resources r ON s.subject_id=r.subject_id LEFT JOIN ResourceUsage u ON r.resource_id=u.resource_id GROUP BY s.subject_id ORDER BY completing_students DESC,s.subject_name;
```

Verified result: 8 row(s). Example: [['Biology', 15], ['Geography', 14]]

30. Report each subject with resources, usage events and average measured duration.

```
SELECT s.subject_name,COUNT(DISTINCT r.resource_id) AS resources,COUNT(u.usage_id) AS uses,ROUND(AVG(u.minutes_spent),1) AS avg_minutes FROM Subjects s LEFT JOIN Resources r ON r.subject_id=s.subject_id LEFT JOIN ResourceUsage u ON u.resource_id=r.resource_id GROUP BY s.subject_id ORDER BY s.subject_name;
```

Verified result: 8 row(s). Example: [['Astronomy', 0, 0, None], ['Biology', 5, 25, 59.8]]