

```
CREATE TABLE Books (  
    book_id INT PRIMARY KEY,  
    title VARCHAR(100),  
    author VARCHAR(100),  
    genre VARCHAR(50),  
    price DECIMAL(6,2),  
    stock INT  
);
```

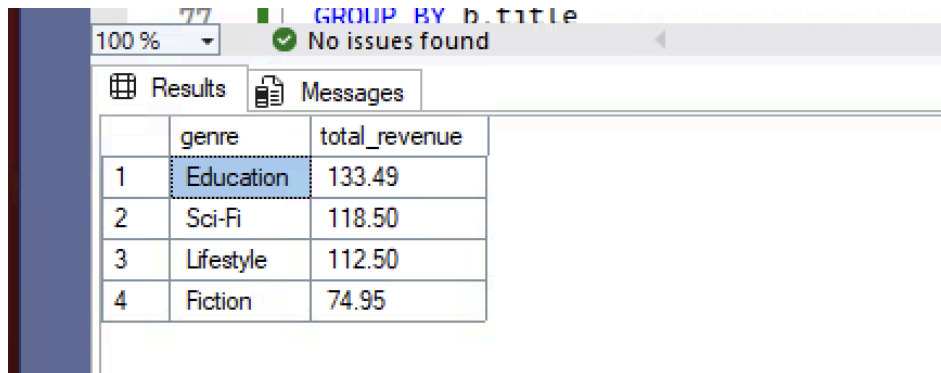
```
INSERT INTO Books (book_id, title, author, genre, price, stock)  
VALUES  
(1, 'The Silent Forest', 'Laura Green', 'Fiction', 14.99, 120),  
(2, 'Data Science 101', 'Sam Curtis', 'Education', 34.50, 60),  
(3, 'Journey to Mars', 'Alan Peters', 'Sci-Fi', 19.75, 80),  
(4, 'Mastering SQL', 'Janet Cole', 'Education', 29.99, 40),  
(5, 'Healthy Living', 'Mia Stone', 'Lifestyle', 22.50, 100);
```

```
CREATE TABLE Sales (  
    sale_id INT PRIMARY KEY,  
    book_id INT,  
    quantity INT,  
    sale_date DATE,  
    FOREIGN KEY (book_id) REFERENCES Books(book_id)  
);
```

```
INSERT INTO Sales (sale_id, book_id, quantity, sale_date) VALUES  
(1, 1, 3, '2025-01-03'),  
(2, 2, 1, '2025-01-04'),  
(3, 3, 4, '2025-01-04'),  
(4, 1, 2, '2025-01-05'),  
(5, 4, 1, '2025-01-06'),  
(6, 3, 2, '2025-01-06'),  
(7, 5, 5, '2025-01-07'),  
(8, 2, 2, '2025-01-08');
```

1) Total revenue by Genre

```
SELECT b.genre,  
       SUM(s.quantity * b.price) AS total_revenue  
FROM Sales s  
JOIN Books b ON s.book_id = b.book_id  
GROUP BY b.genre  
ORDER BY total_revenue DESC;
```

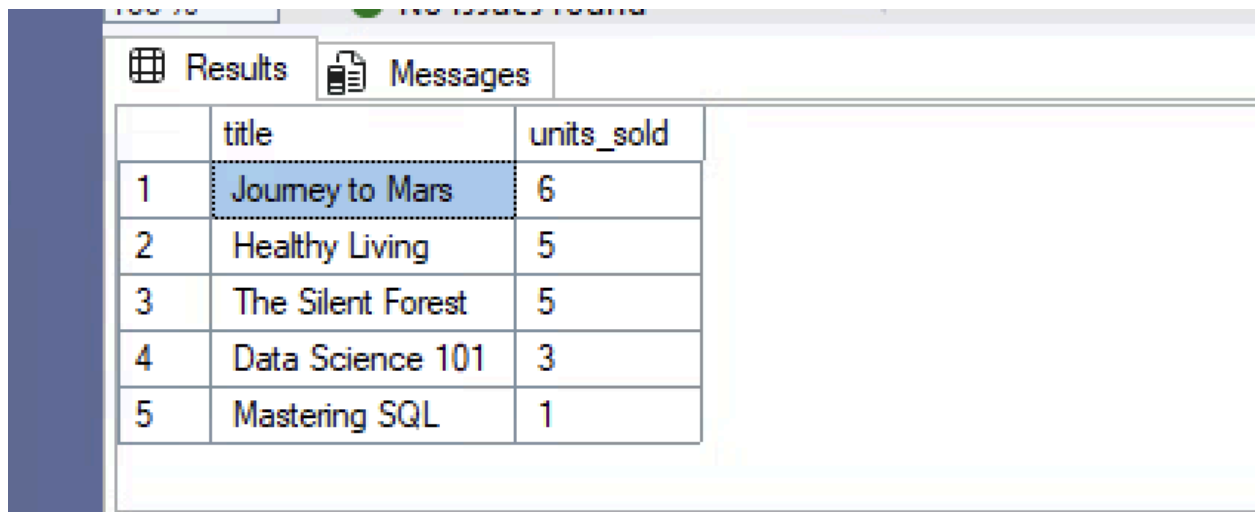


The screenshot shows a database query result in a web interface. The query is: `SELECT b.genre, SUM(s.quantity * b.price) AS total_revenue FROM Sales s JOIN Books b ON s.book_id = b.book_id GROUP BY b.genre ORDER BY total_revenue DESC;`. The result is displayed in a table with two columns: 'genre' and 'total_revenue'. The table is sorted by total revenue in descending order. The first row is 'Education' with a total revenue of 133.49. The second row is 'Sci-Fi' with a total revenue of 118.50. The third row is 'Lifestyle' with a total revenue of 112.50. The fourth row is 'Fiction' with a total revenue of 74.95. The interface also shows a 'Results' tab and a 'Messages' tab, with a status bar indicating 'No issues found'.

	genre	total_revenue
1	Education	133.49
2	Sci-Fi	118.50
3	Lifestyle	112.50
4	Fiction	74.95

2) Bestselling books

```
SELECT b.title,  
       SUM(s.quantity) AS units_sold  
FROM Sales s  
JOIN Books b ON s.book_id = b.book_id  
GROUP BY b.title  
ORDER BY units_sold DESC;
```



The screenshot shows a database query result in a web interface. The query is: `SELECT b.title, SUM(s.quantity) AS units_sold FROM Sales s JOIN Books b ON s.book_id = b.book_id GROUP BY b.title ORDER BY units_sold DESC;`. The result is displayed in a table with two columns: 'title' and 'units_sold'. The table is sorted by units sold in descending order. The first row is 'Journey to Mars' with 6 units sold. The second row is 'Healthy Living' with 5 units sold. The third row is 'The Silent Forest' with 5 units sold. The fourth row is 'Data Science 101' with 3 units sold. The fifth row is 'Mastering SQL' with 1 unit sold. The interface also shows a 'Results' tab and a 'Messages' tab, with a status bar indicating 'No issues found'.

	title	units_sold
1	Journey to Mars	6
2	Healthy Living	5
3	The Silent Forest	5
4	Data Science 101	3
5	Mastering SQL	1

3) Total revenue per book

```
SELECT b.title,  
       SUM(s.quantity * b.price) AS revenue  
FROM Sales s  
JOIN Books b ON s.book_id = b.book_id  
GROUP BY b.title  
ORDER BY revenue DESC;
```

Results		Messages
	title	revenue
1	Journey to Mars	118.50
2	Healthy Living	112.50
3	Data Science 101	103.50
4	The Silent Forest	74.95
5	Mastering SQL	29.99

4) Books by Revenue

```
SELECT title, revenue,  
       RANK() OVER (ORDER BY revenue DESC) AS ranking  
FROM (  
    SELECT b.title,  
           SUM(s.quantity * b.price) AS revenue  
    FROM Sales s  
    JOIN Books b ON s.book_id = b.book_id  
    GROUP BY b.title  
) t;
```

Results		Messages	
	title	revenue	ranking
1	Journey to Mars	118.50	1
2	Healthy Living	112.50	2
3	Data Science 101	103.50	3
4	The Silent Forest	74.95	4
5	Mastering SQL	29.99	5