# Stage 2 – Intermediate SQL: Sample SQL for Testing

## CREATE Samples

**Assume we have two tables like the following:**

CREATE TABLE Book (

bookId int PRIMARY KEY,

title varchar(30),

pages int,

authorId int,  
editorial varchar(30)

)

CREATE TABLE Author (

authorId int PRIMARY KEY,  
 name varchar(30),  
 nationality varchar(30)  
)

## INSERT Samples

Please refer to files **author.sql** and **book2.sql**.

## SELECT Samples

**We need to perform the following simple SQL queries on the two tables Author and Book from the INSERT statements in files book2.sql and author.sql:**

SELECT bookId, title, pages, authorId, editorial  
FROM Book;

This query should return:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **bookId** | **title** | **pages** | **authorId** | **editorial** |
| 1 | Bible | 500 | 1 | Prentice Hall |
| 2 | Computer Science | 200 | 2 | Barron’s |
| 3 | Study Guide | 140 | 1 | Prentice Hall |
| 4 | Jurassic Park | 450 | 3 | Penguin Books |
| 5 | Congo | 500 | 3 | Bauhaus |
| 6 | Romeo and Juliet | 300 | 4 | English Books |
| 7 | The Merchant of Venice | 350 | 4 | English Books |
| 8 | Network Programming | 79 | 5 | Prentice Hall |
| 9 | Star Wars | 1320 | 6 | Penguin Books |
| 10 | VPN Architectures | 450 | 2 | Barron’s |

SELECT \*  
FROM Author;

This query should return:

|  |  |  |
| --- | --- | --- |
| **authorId** | **name** | **nationality** |
| 1 | Jim Chen | Taiwan |
| 2 | John Goodman | Zaire |
| 3 | Michael Crichton | USA |
| 4 | Shakespeare | England |
| 5 | Tim Chang | Taiwan |
| 6 | George Lucas | USA |
| 7 | Garcia Marquez | Colombia |
| 8 | Katsu Moto | Japan |
| 9 | Confucius | China |
| 10 | Jesus | Nazareth |

SELECT title  
FROM Book  
WHERE bookId = 1;

This query should return:

|  |
| --- |
| **title** |
| Bible |

SELECT b.title  
FROM Book AS b  
WHERE pages > 100 AND editorial = 'Prentice Hall';

This query should return:

|  |
| --- |
| **b.title** |
| Bible |
| Study Guide |

SELECT \*  
FROM Book  
WHERE authorId = 1 OR pages < 200;

This query should return:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **bookId** | **title** | **pages** | **authorId** | **editorial** |
| 1 | Bible | 500 | 1 | Prentice Hall |
| 3 | Study Guide | 140 | 1 | Prentice Hall |
| 8 | Network Programming | 79 | 5 | Prentice Hall |

**Now some inner join queries:**

SELECT b.\*  
FROM Book AS b, Author AS a  
WHERE b.authorId = a.authorId AND a.name = 'Michael Crichton';

This query should return:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **b.bookId** | **b.title** | **b.pages** | **b.authorId** | **b.editorial** |
| 4 | Jurassic Park | 450 | 3 | Penguin Books |
| 5 | Congo | 500 | 3 | Bauhaus |

SELECT bookId, title, pages, name  
FROM Book, Author  
WHERE Book.authorId = Author.authorId;

This query should return:

|  |  |  |  |
| --- | --- | --- | --- |
| **bookId** | **title** | **pages** | **name** |
| 1 | Bible | 500 | Jim Chen |
| 2 | Computer Science | 200 | John Goodman |
| 3 | Study Guide | 140 | Jim Chen |
| 4 | Jurassic Park | 450 | Michael Crichton |
| 5 | Congo | 500 | Michael Crichton |
| 6 | Romeo and Juliet | 300 | Shakespeare |
| 7 | The Merchant of Venice | 350 | Shakespeare |
| 8 | Network Programming | 79 | Tim Chang |
| 9 | Star Wars | 1320 | George Lucas |
| 10 | VPN Architectures | 450 | John Goodman |

SELECT a.name, title  
FROM Book, Author AS a  
WHERE Book.authorId = a.authorId AND Book.pages > 200;

This query should return:

|  |  |
| --- | --- |
| **a.name** | **title** |
| Jim Chen | Bible |
| Michael Crichton | Jurassic Park |
| Michael Crichton | Congo |
| Shakespeare | Romeo and Juliet |
| Shakespeare | The Merchant of Venice |
| George Lucas | Star Wars |
| John Goodman | VPN Architectures |

SELECT a.name  
FROM Author AS a, Book AS b  
WHERE a.authorId = b.authorId AND b.title = 'Star Wars';

This query should return:

|  |
| --- |
| **a.name** |
| George Lucas |

SELECT a.name, b.title  
FROM Author AS a, Book AS b  
WHERE a.authorId = b.authorId AND a.nationality <> 'Taiwan';

This query should return:

|  |  |
| --- | --- |
| **a.name** | **b.title** |
| John Goodman | Computer Science |
| Michael Crichton | Jurassic Park |
| Michael Crichton | Congo |
| Shakespeare | Romeo and Juliet |
| Shakespeare | The Merchant of Venice |
| George Lucas | Star Wars |
| John Goodman | VPN Architectures |

**And now some aggregation functions:**

SELECT COUNT(\*)  
FROM Book;

This query should return:

|  |
| --- |
| **COUNT(\*)** |
| 10 |

SELECT COUNT(editorial)  
FROM Book;

This query should return:

|  |
| --- |
| **COUNT(editorial)** |
| 10 |

SELECT COUNT(\*)  
FROM Author  
WHERE nationality = 'Taiwan';

This query should return:

|  |
| --- |
| **COUNT(\*)** |
| 2 |

SELECT SUM(pages)  
FROM Book  
WHERE authorId = 2;

This query should return:

|  |
| --- |
| **SUM(pages)** |
| 650 |

**And finally some possible incorrect SELECT and INSERT statements:**

SELECT authorId  
FROM Author, Book  
WHERE Author.authorId = Book.authorId AND Book.title = 'Star Wars';

This SQL statement would cause an error because the attribute “authorId” is ambiguous, as it appears in both the Author and Book tables.

SELECT \*  
FROM Author  
WHERE authorId = 'John';

This SQL statement would cause an error because the attribute authorId is of type integer, not char or varchar.

SELECT Book.\*  
FROM Book, Author  
WHERE Book.authorId = Author.name;

This SQL statement would cause an error because the attribute Book.authorId and Author.name are of different types and cannot be compared.