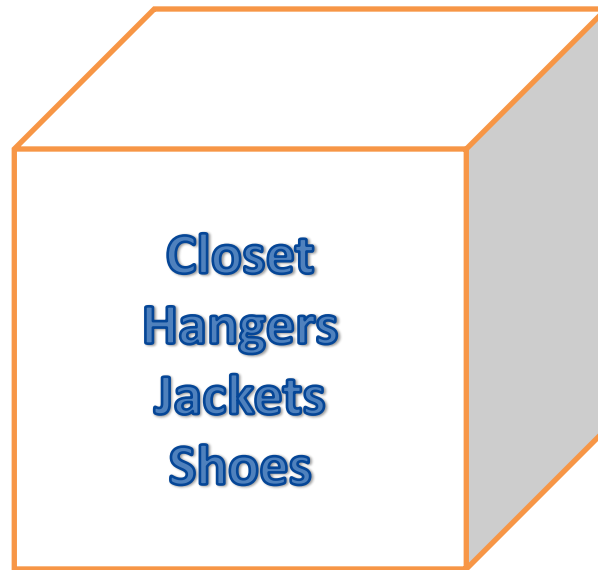


# Introduction to Databases and SQL

(Structured Query Language)

# Why Have a Database?

- A moving story



# Moving Lists

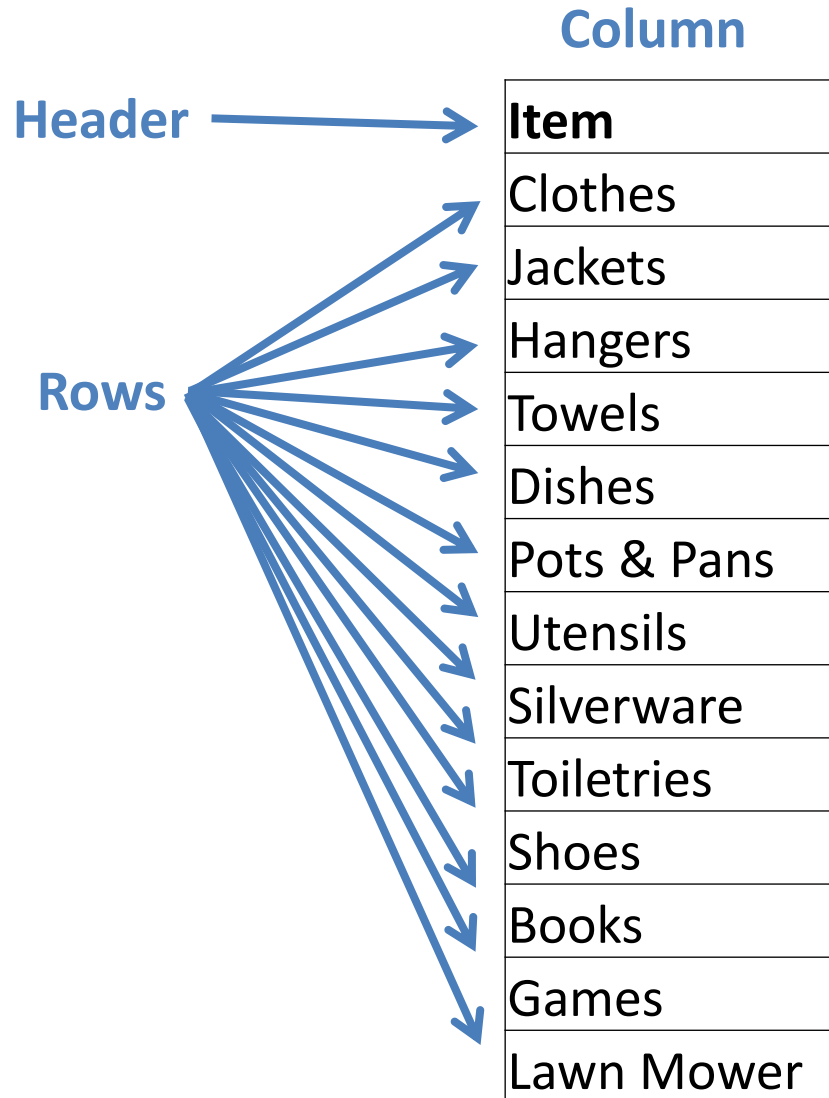
- Rooms
  - Kitchen
  - Garage
  - Family Room
  - Dining Room
  - Bedroom
  - Bathroom
  - Closet
  - Patio
- Items
  - Clothes
  - Jackets
  - Hangers
  - Towels
  - Dishes
  - Pots & Pans
  - Utensils
  - Silverware
  - Toiletries
  - Shoes
  - Books
  - Games
  - Lawn Mower

# Database Table Rooms

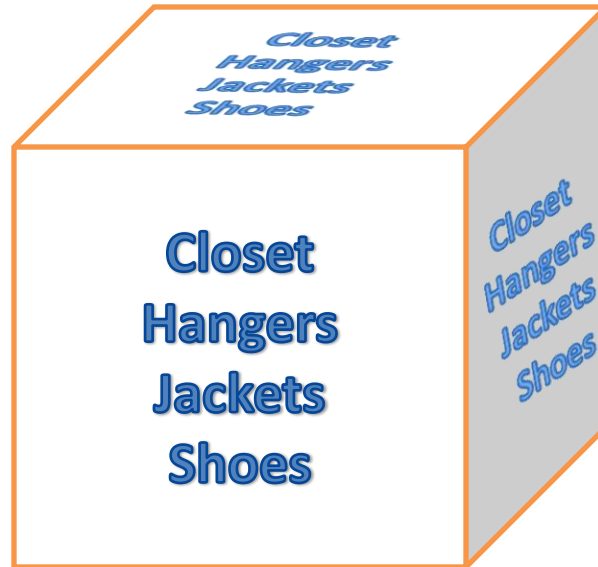
The diagram illustrates a database table structure. A blue arrow labeled "Header" points to the first row of the table, which contains the word "Room". A blue arrow labeled "Rows" points to the remaining eight rows of the table, which contain the names of different rooms: Kitchen, Garage, Family Room, Dining Room, Bedroom, Bathroom, Closet, and Patio. The word "Column" is written in blue above the table.

Room
Kitchen
Garage
Family Room
Dining Room
Bedroom
Bathroom
Closet
Patio

# Database Table Items



# Box Contents



# Database Table Boxes

The diagram illustrates a database table structure. The word "Columns" is positioned above the table, with five blue arrows pointing to each of the five columns. The word "Header" is to the left of the first row, with a blue arrow pointing to the "Box #" header cell. The word "Rows" is to the left of the first three data rows, with three blue arrows pointing to each row. The table itself has five columns and four rows. The first row is the header row, and the following three rows are data rows.

Box #	Room	Item 1	Item 2	Item 3
1	Closet	Hangers	Jackets	Shoes
2	Bathroom	Towels	Toiletries	
3	Kitchen	Towels	Dishes	Silverware

# Database Table Boxes

**But wait! Box #3 also has Utensils!**

The diagram illustrates a database table structure. The word "Columns" is positioned above the table with five arrows pointing to each of the five columns. The word "Header" is to the left of the first row with an arrow pointing to the "Box #" column. The word "Rows" is to the left of the table with three arrows pointing to the three data rows.

Box #	Room	Item 1	Item 2	Item 3
1	Closet	Hangers	Jackets	Shoes
2	Bathroom	Towels	Toiletries	
3	Kitchen	Towels	Dishes	Silverware



# Database Table Boxes

Diagram illustrating a Database Table structure with annotations:

- Columns:** Indicated by blue arrows pointing to the header row (Box #, Room, Item 1, Item 2, Item 3, Item 4).
- Header:** Indicated by a blue arrow pointing to the first row (Box #, Room, Item 1, Item 2, Item 3, Item 4).
- Rows:** Indicated by blue arrows pointing to the data rows (1, 2, 3).

Box #	Room	Item 1	Item 2	Item 3	Item 4
1	Closet	Hangers	Jackets	Shoes	
2	Bathroom	Towels	Toiletries		
3	Kitchen	Towels	Dishes	Silverware	Utensils

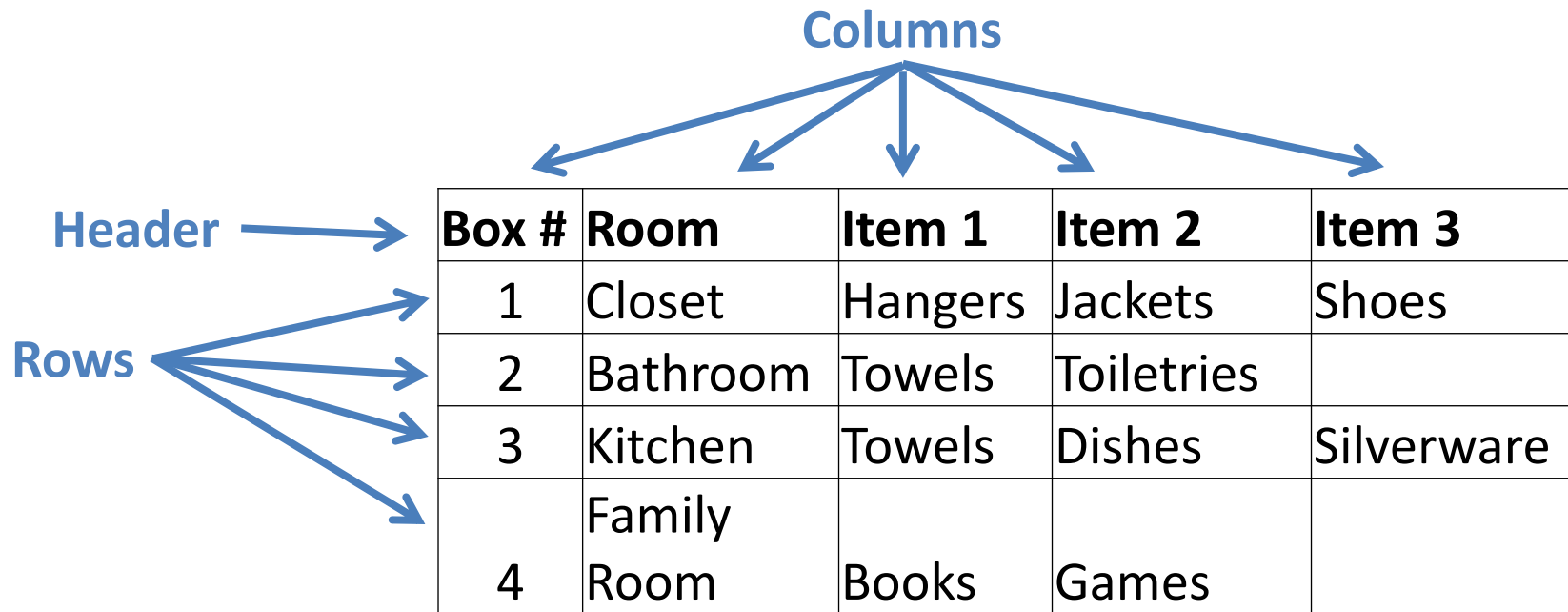
# Database Table Boxes

The diagram illustrates a database table structure. The word "Columns" is positioned above the table with five arrows pointing to each of the six columns. The word "Header" is to the left of the first row with an arrow pointing to it. The word "Rows" is to the left of the table with four arrows pointing to each of the four rows.

Box #	Room	Item 1	Item 2	Item 3	Item 4
1	Closet	Hangers	Jackets	Shoes	
2	Bathroom	Towels	Toiletries		
3	Kitchen	Towels	Dishes	Silverware	Utensils
4	Family Room	Books	Games		

# Database Table Boxes

**But wait! Box #4 has Books from two rooms!**

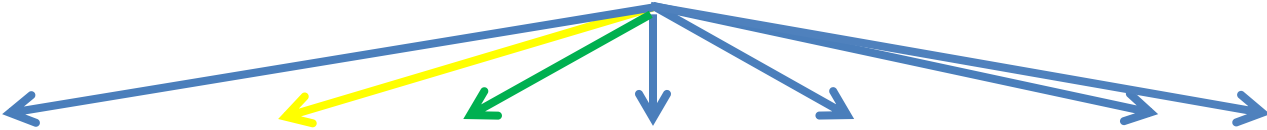


The diagram illustrates a database table structure. A table with 5 columns and 4 rows is shown. Above the table, the word "Columns" is written in blue, with five blue arrows pointing to each of the five columns. To the left of the table, the word "Header" is written in blue, with a blue arrow pointing to the first column. Below the table, the word "Rows" is written in blue, with four blue arrows pointing to each of the four rows.

Box #	Room	Item 1	Item 2	Item 3
1	Closet	Hangers	Jackets	Shoes
2	Bathroom	Towels	Toiletries	
3	Kitchen	Towels	Dishes	Silverware
4	Family Room	Books	Games	

# Database Table Boxes

Columns



Box #	Room 1	Room 2	Item 1	Item 2	Item 3	Item 4
1	Closet		Hangers	Jackets	Shoes	
2	Bathroom		Towels	Toiletries		
3	Kitchen		Towels	Dishes	Silverware	Utensils
4	Family Room	Bedroom	Books	Games		

# Database Table Boxes

**But wait! Box #5 has multiple items  
from multiple rooms!**

Box #	Room 1	Room 2	Item 1	Item 2	Item 3	Item 4
1	Closet		Hangers	Jackets	Shoes	
2	Bathroom		Towels	Toiletries		
3	Kitchen		Towels	Dishes	Silverware	Utensils
4	Family Room	Bedroom	Books	Games		
5	<b>Miscellaneous</b>					

# There Must be a Better Way!

Let's talk about relationships.

- A Box can have:
  - One item in it (books)
  - Many items in it (hangers, jackets, shoes)
- Likewise a Box can have items from:
  - One room (kitchen)
  - Many rooms (family room, bedroom)

# Relationships and Cardinality

- Relationship
  - How 2 or more things are connected
- Cardinality
  - The number of things in a set

# Cardinality

- Cardinality examples
  - Zero things
  - One thing
  - Many things

How many is Many?



# Relationships and Cardinality

- A **Box** can have:
  - **One item** in it (books)
  - **Many items** in it (hangers, jackets, shoes)
- Likewise a **Box** can have items from:
  - **One room** (kitchen)
  - **Many rooms** (family room, bedroom)

# Large Items?

- Items that will not be boxed
  - Patio
    - Lawn Mower
  - Family Room
    - Couch
    - Chair
    - Tables
    - Etc.

# Relationships and Cardinality

- An **Item** belongs to **One Room**
  - Clothes
- An **Item** may go into **One Box**, **Many Boxes** , or **No Boxes**
  - Silverware, Shoes, Lawn mower

# Empty Boxes?

- Do we care about empty boxes?
  - Yes
    - We'll keep them and move them too.
  - No
    - We'll sell them back or trash them.

# Zero Cardinality or Optionality

- Zero Cardinality (Optionality) examples
  - An **Item** may not go in **One Box**
    - Lawn mower, couch, table, etc.
  - A **Room** may not have anything that goes into a **Box**
  - A **Box** may not have anything in it.

# Data Model Notation

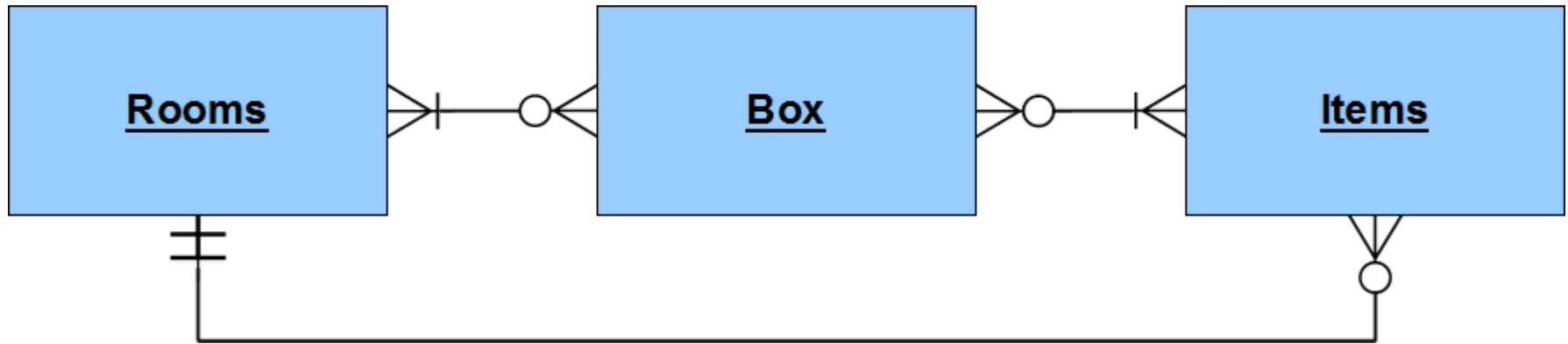


Thing A has (is related to) One or More of Thing B

Thing B has (is related to) One and Only One Thing A

Notation	Zero or One	One and Only One	Zero or Many	One or Many
Crow's Feet (IDEF1X)				

# Moving Data Model



Notation	Zero or One	One and Only One	Zero or Many	One or Many
Crow's Feet (IDEF1X)				

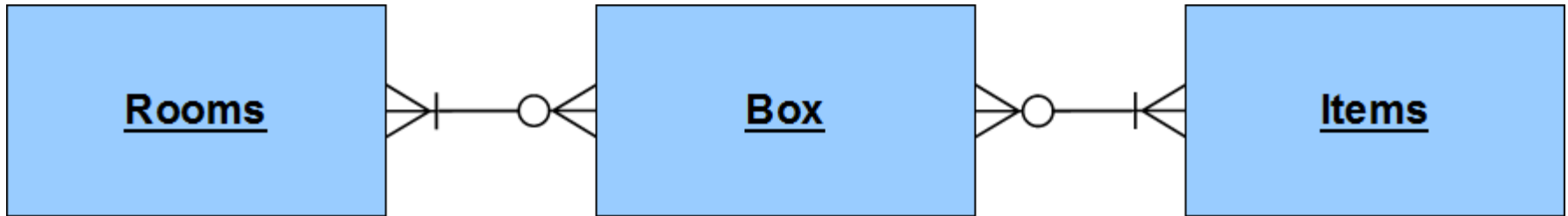
# Database Table Boxes

Back to our problem

Box	Room 1	Room 2	Item 1	Item 2	Item 3	Item 4
1	Closet		Hangers	Jackets	Shoes	
2	Bathroom		Towels	Toiletries		
3	Kitchen		Towels	Dishes	Silverware	Utensils
4	Family Room	Bedroom	Books	Games		
5	<b>Miscellaneous</b>					

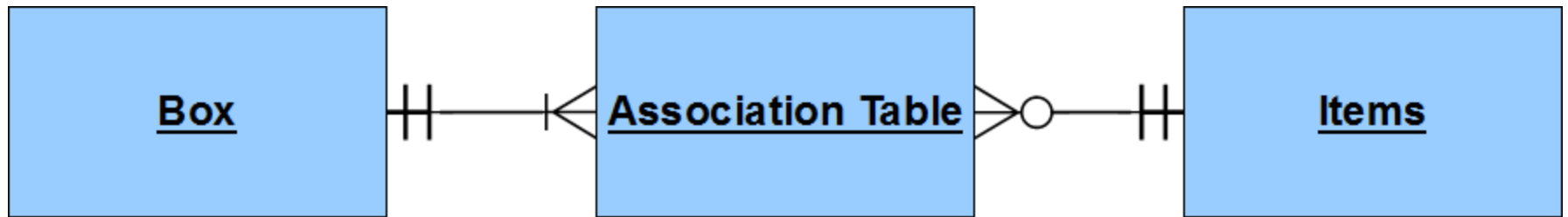
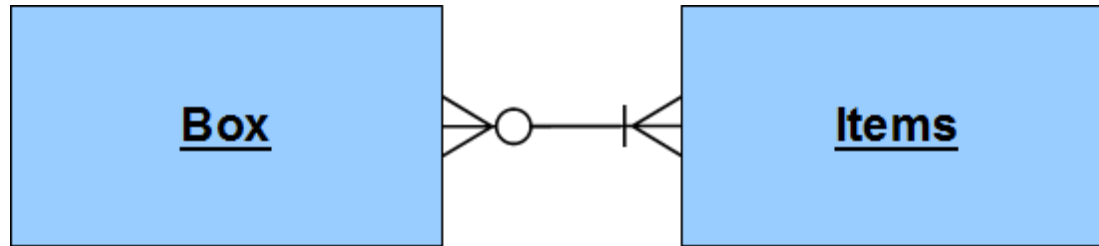


# Database Table Boxes



Box	Room 1	Room 2	Item 1	Item 2	Item 3	Item 4
1	Closet		Hangers	Jackets	Shoes	
2	Bathroom		Towels	Toiletries		
3	Kitchen		Towels	Dishes	Silverware	Utensils
4	Family Room	Bedroom	Books	Games		
5	<b>Miscellaneous</b>					

# Association Table



# Association Table Solution

Box #	Room	Item
1	Closet	Hangers
1	Closet	Jackets
1	Closet	Shoes
2	Barthroom	Towels
2	Bathroom	Toiletries
4	Family Room	Games
4	Bedroom	Books
5	Closet	hanger
5	Closet	Shoes
5	Garbage	Shoes
5	Kitchen	Books

# But Wait... Typos!

Box #	Room	Item
1	Closet	Hangers
1	Closet	Jackets
1	Closet	Shoes
2	Barthroom	Towels
2	Bathroom	Toiletries
4	Family Room	Games
4	Bedroom	Books
5	Closet	hanger
5	Closet	Shoes
5	Garbage	Shoes
5	Kitchen	Books

# Database Table Rooms - Updated

The diagram illustrates the structure of a database table. The word "Columns" is positioned above the table with two arrows pointing to the "Room ID" and "Room" headers. The word "Header" is to the left of the first row with an arrow pointing to the "Room ID" header. The word "Rows" is to the left of the data rows with eight arrows pointing to each row from the second row down to the eighth row.

Room ID	Room
1	Kitchen
2	Garage
3	Family Room
4	Dining Room
5	Bedroom
6	Bathroom
7	Closet
8	Patio

# Database Table Items - Updated

The diagram illustrates a database table structure. The word "Columns" is positioned above the table with two arrows pointing to the "Item ID" and "Item" headers. The word "Header" is to the left of the first row with an arrow pointing to the "Item ID" cell. The word "Rows" is to the left of the data rows with multiple arrows pointing to each row from the second row down to the last row.

Item ID	Item
1	Clothes
2	Jackets
3	Hangers
4	Towels
5	Dishes
6	Pots & Pans
7	Utensils
8	Silverware
9	Toiletries
10	Shoes
11	Books
12	Games
13	Lawn Mower

# Association Table - Updated

**No Typos!**

Box	Room ID	Item ID
1	7	3
1	7	2
1	7	10
2	6	4
2	6	9
4	3	12
4	5	11
5	7	3
5	7	10
5	2	10
5	1	11

**But what's in  
the Box?**

# SQL Statements

- SELECT – Retrieve (SELECT) data from a table.
- INSERT – Add (INSERT) data into a table.
- DELETE – Remove (DELETE) data from a table.
- CREATE – Add (CREATE) a table, or other object.
- DROP – Remove (DROP) a table, or other object.
- EXEC – Run (EXECUTE) a Stored Procedure.



# Anatomy of a SELECT Statement

SELECT

<column,...>

FROM <table>

WHERE <condition>

ORDER BY <column,...>

# Anatomy of a SELECT Statement

SELECT

<column,...>

*A list of columns or just use \* for all columns*

---

SELECT

\*

*Get all columns*

SELECT

Name, Zip, Phone

*Get only 3 columns*

# Anatomy of a SELECT Statement

FROM <table>

*Name of a table*

---

---

FROM Rooms	FROM Items	FROM Boxes
<i>Rooms table</i>	<i>Items table</i>	<i>Boxes table</i>

# Anatomy of a SELECT Statement

WHERE <condition>

*Criteria or filter conditions*

---

WHERE Item = 'Books'

*Only get rows where the  
Item is Books*

WHERE ItemID >= 4

*Only get rows where the  
value of the Item ID is  
greater than or equal to*

# Anatomy of a SELECT Statement

ORDER BY <column,...>

*Sort by a column or columns,  
Ascending (ASC) is the default,  
descending (DESC) is an option.*

---

ORDER BY Item

*Sort by the Item  
(ascending)*

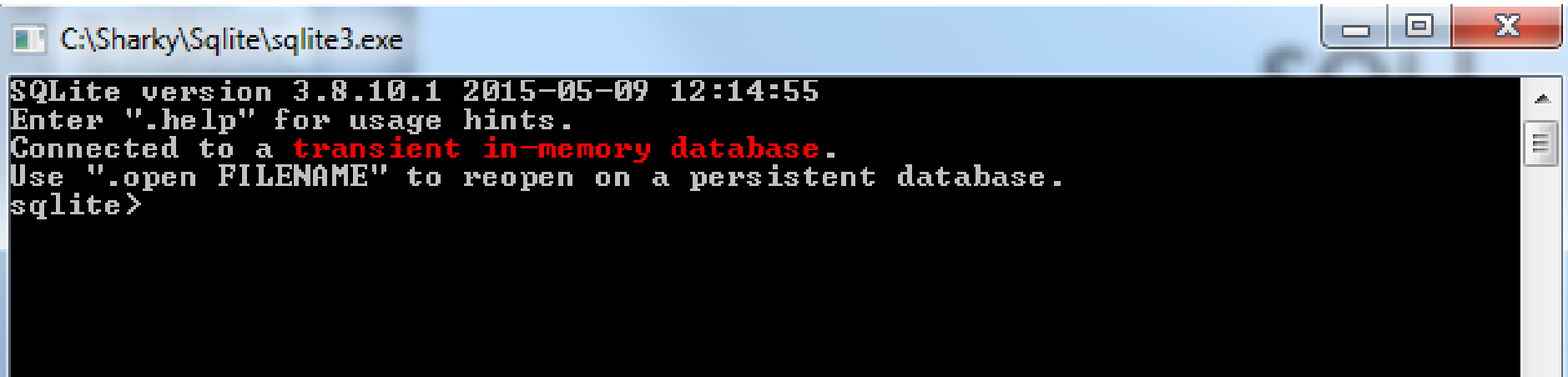
|  
|  
|  
|

ORDER BY ItemID DESC

*Sort by the ItemID  
descending*

# Time to Play!

## Open sqlite3

A screenshot of a Windows command prompt window titled "C:\Sharky\Sqlite\sqlite3.exe". The window has standard Windows window controls (minimize, maximize, close) in the top right corner. The command prompt shows the following text: "SQLite version 3.8.10.1 2015-05-09 12:14:55", "Enter ".help" for usage hints.", "Connected to a transient in-memory database.", "Use ".open FILENAME" to reopen on a persistent database.", and the prompt "sqlite>".

```
C:\Sharky\Sqlite\sqlite3.exe
SQLite version 3.8.10.1 2015-05-09 12:14:55
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite>
```

# Sqlite housekeeping

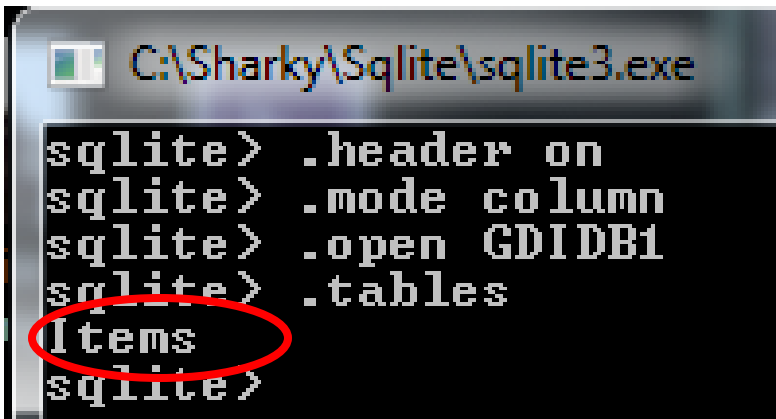
```
sqlite> .header on
```

```
sqlite> .mode column
```

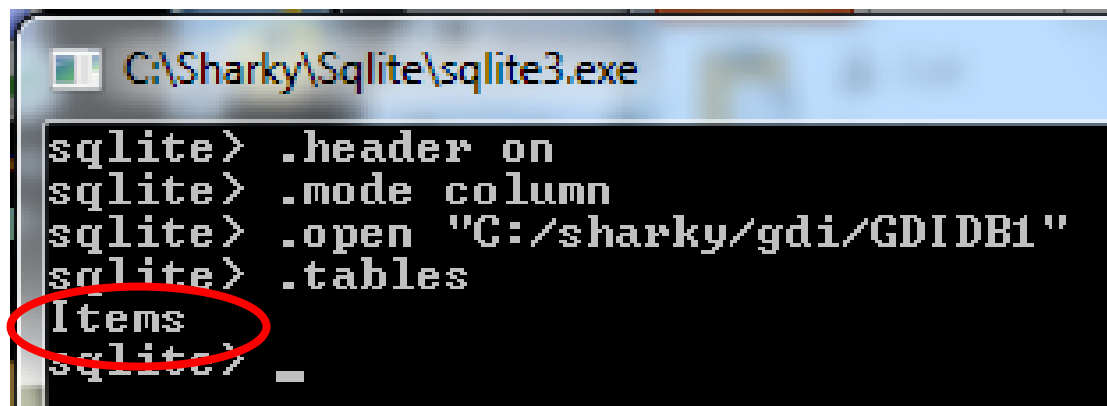
```
sqlite> .open GDIDB1
```

```
sqlite> .tables
```

To use full path - use '/' and must be in quotes!  
.  
open "C:/sharky/gdi/GDIDB1"



```
C:\Sharky\Sqlite\sqlite3.exe
sqlite> .header on
sqlite> .mode column
sqlite> .open GDIDB1
sqlite> .tables
Items
sqlite>
```



```
C:\Sharky\Sqlite\sqlite3.exe
sqlite> .header on
sqlite> .mode column
sqlite> .open "C:/sharky/gdi/GDIDB1"
sqlite> .tables
Items
sqlite> _
```

# Basic SELECT Statement

SELECT

<column,...> *(use \* for all columns)*

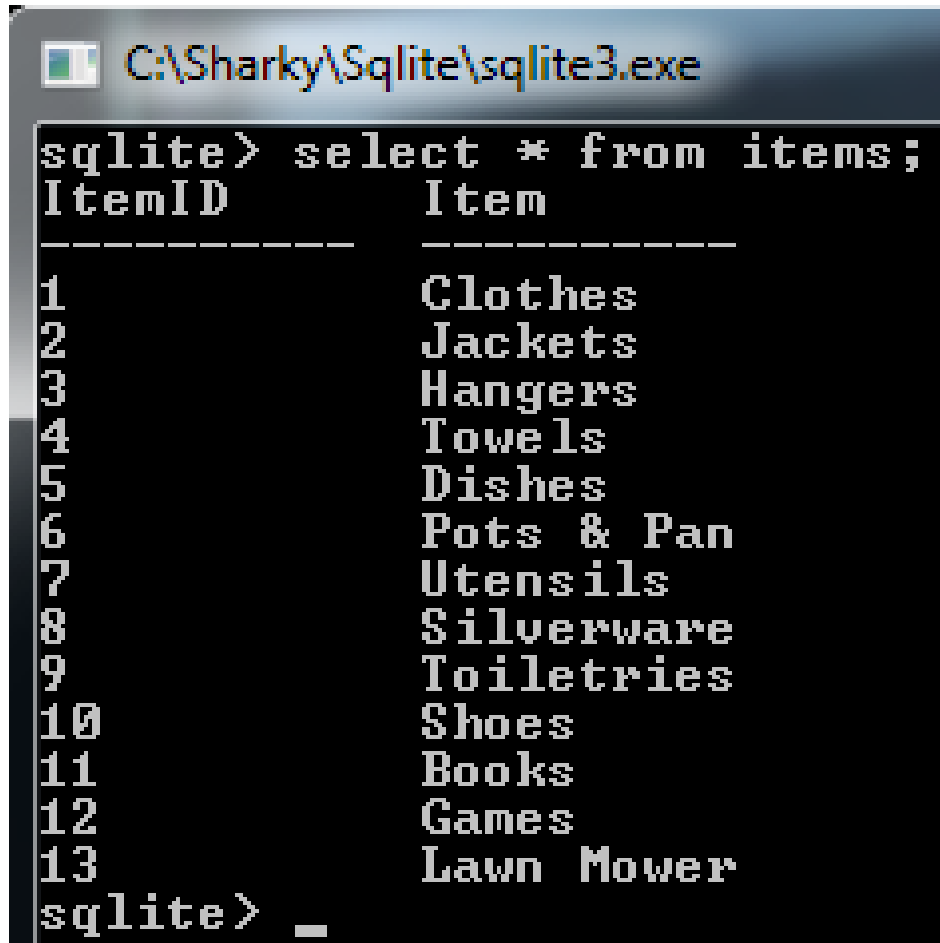
FROM <table>



# SQL!

## Single line

sqlite> **SELECT \* FROM Items;**



The screenshot shows a command prompt window titled "C:\Sharky\Sqlite\sqlite3.exe". The prompt is "sqlite>". The user has entered the command "select \* from items;". The output is a table with two columns: "ItemID" and "Item". The data is as follows:

ItemID	Item
1	Clothes
2	Jackets
3	Hangers
4	Towels
5	Dishes
6	Pots & Pan
7	Utensils
8	Silverware
9	Toiletries
10	Shoes
11	Books
12	Games
13	Lawn Mower

The prompt is now "sqlite> \_".

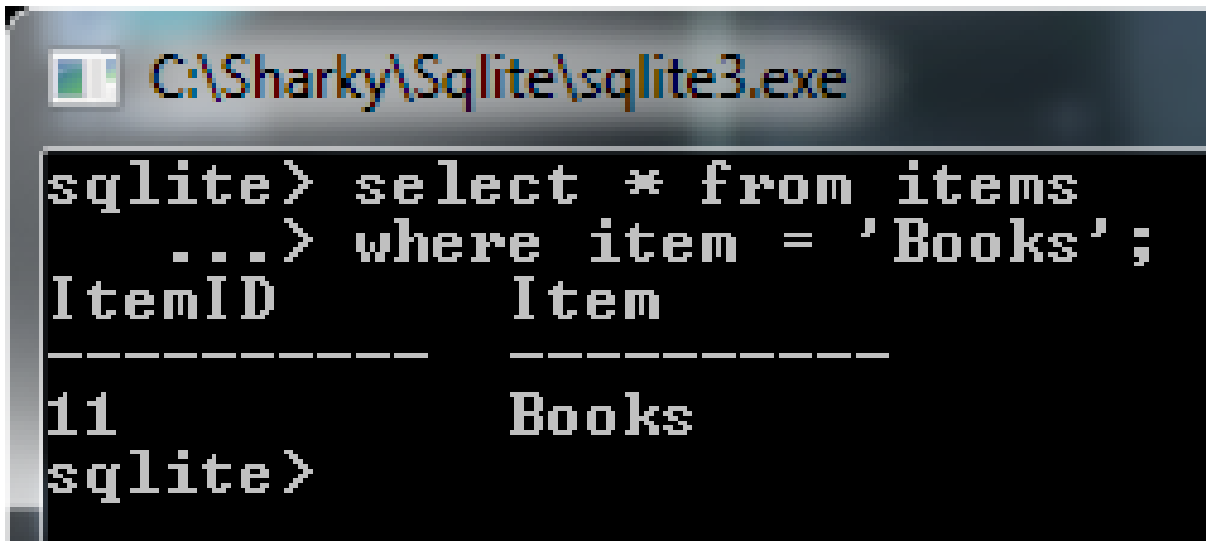
## Multi-line

sqlite> **SELECT \***  
...> **FROM Items**  
...> **;**

# SQL Where Clause

sqlite> **SELECT \* FROM Items**

**...> WHERE Item = 'books';**



The screenshot shows a Windows command prompt window titled "C:\Sharky\Sqlite\sqlite3.exe". The prompt is "sqlite>". The user enters the command "select \* from items" followed by a new line and "...> where item = 'Books';". The output is a table with two columns: "ItemID" and "Item". The first row of data shows "11" for ItemID and "Books" for Item. The prompt returns to "sqlite>".

```
sqlite> select * from items
...> where item = 'Books';
ItemID      Item
-----
11          Books
sqlite>
```

# SQL Where Clause

sqlite> **SELECT \* FROM Items**

**...> WHERE ItemID >= 4;**

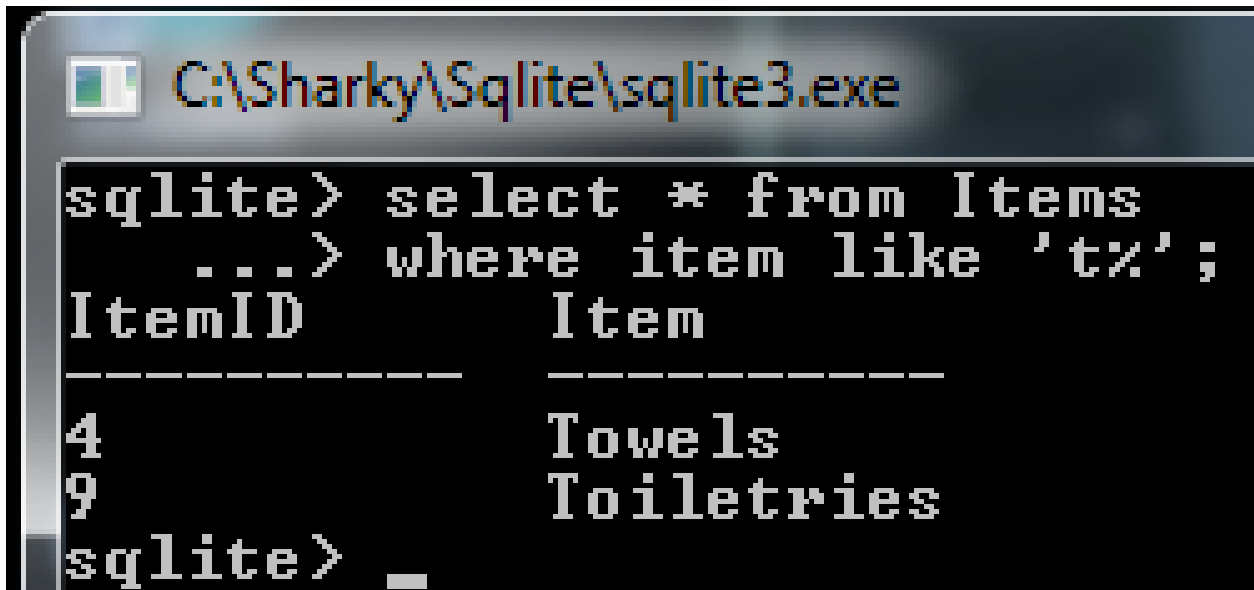
A screenshot of a SQLite command prompt window. The title bar shows the file path 'C:\Sharky\Sqlite\sqlite3.exe'. The command prompt shows the user entering the SQL query 'select \* from Items' followed by a new line and 'where itemID >= 4;'. The output is a table with two columns: 'ItemID' and 'Item'. The table contains 10 rows of data, starting from ItemID 4. The items listed are Towels, Dishes, Pots & Pan, Utensils, Silverware, Toiletries, Shoes, Books, Games, and Lawn Mower.

```
C:\Sharky\Sqlite\sqlite3.exe
sqlite> select * from Items
...> where itemID >= 4;
ItemID      Item
-----
4           Towels
5           Dishes
6           Pots & Pan
7           Utensils
8           Silverware
9           Toiletries
10          Shoes
11          Books
12          Games
13          Lawn Mower
sqlite>
```

# SQL Where Clause

```
sqlite> SELECT * FROM Items  
...> WHERE Item like 't%';
```

% is a wildcard  
't%' – starts with 't'  
'%t' – ends with 't'  
'%t%' - has a 't'  
somewhere

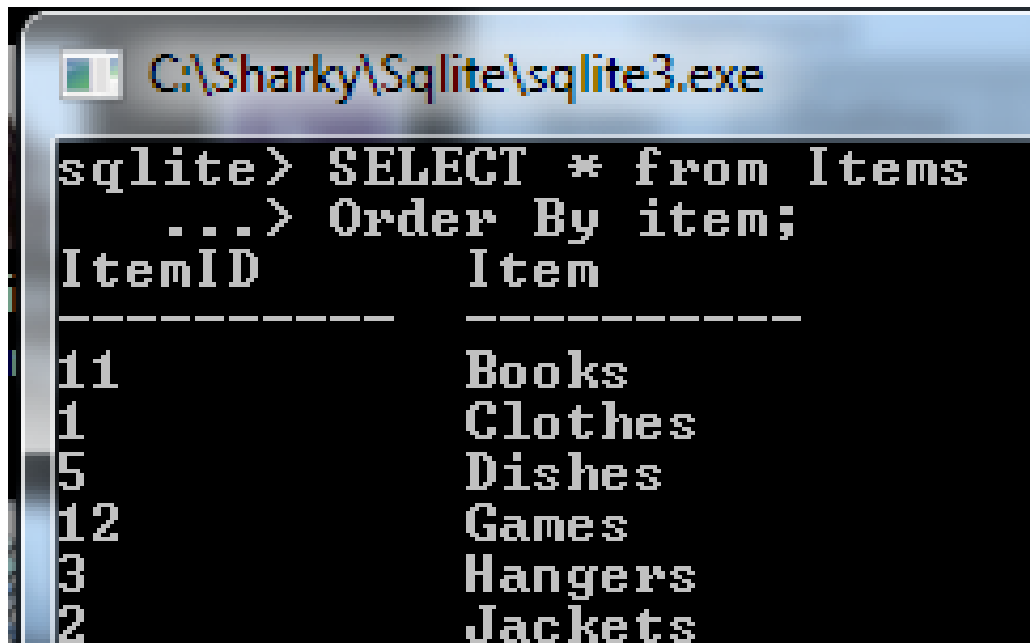


The screenshot shows a SQLite command prompt window with the title bar 'C:\Sharky\Sqlite\sqlite3.exe'. The command prompt displays the following text:

```
sqlite> select * from Items  
...> where item like 't%';  
ItemID      Item  
-----  
4           Towels  
9           Toiletries  
sqlite> _
```

# SQL Order By Clause

```
sqlite> SELECT * FROM Items  
...> Order By item;
```



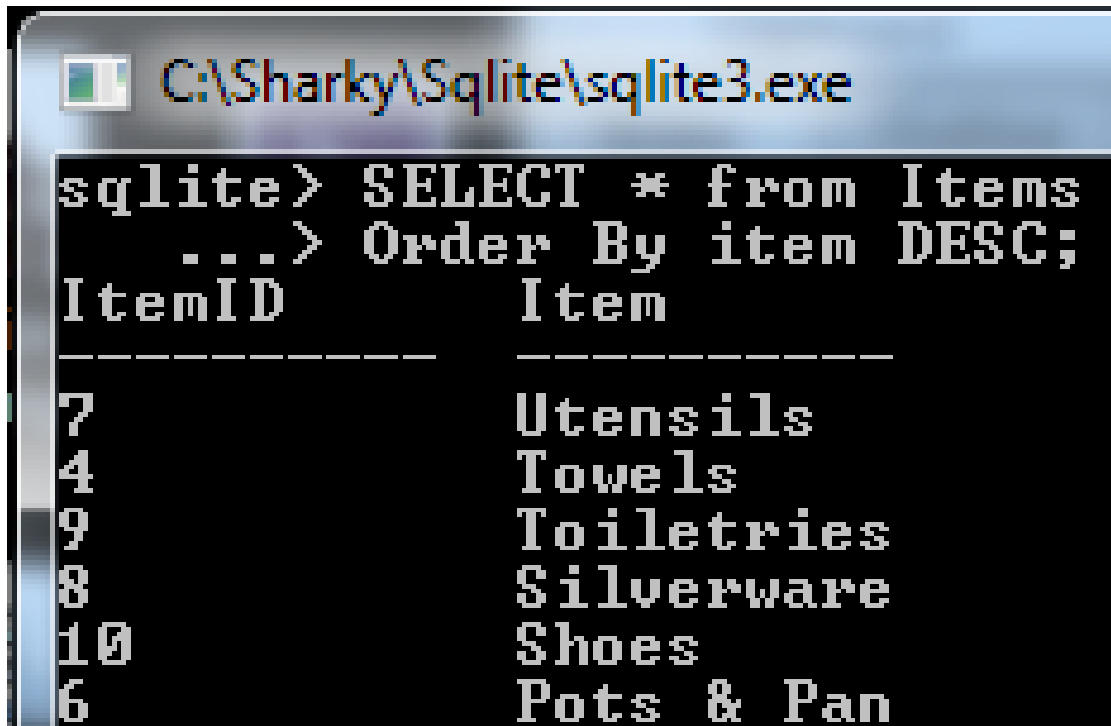
The screenshot shows a Windows command prompt window titled "C:\Sharky\Sqlite\sqlite3.exe". The prompt is "sqlite>". The user has entered the SQL query "SELECT \* from Items" and then "Order By item;". The output is a table with two columns: "ItemID" and "Item". The data is sorted by the "Item" column in ascending order. The output is as follows:

ItemID	Item
11	Books
1	Clothes
5	Dishes
12	Games
3	Hangers
2	Jackets

# SQL Order By Clause

sqlite> **SELECT \* FROM Items**

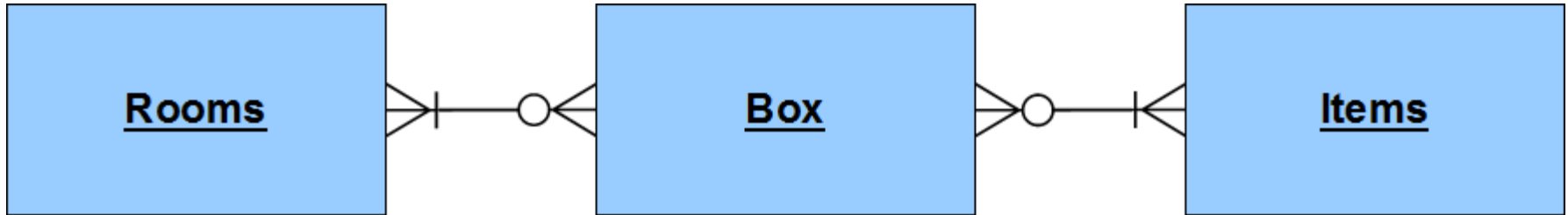
**...> Order By item DESC;**



A screenshot of a SQLite command prompt window. The title bar shows the file path 'C:\Sharky\Sqlite\sqlite3.exe'. The command prompt displays the following text:

```
sqlite> SELECT * from Items
...> Order By item DESC;
ItemID      Item
-----
7           Utensils
4           Towels
9           Toiletries
8           Silverware
10          Shoes
6           Pots & Pan
```

# Multiple Related Tables



Box	Room ID	Item ID
1	7	3
1	7	2
1	7	10
2	6	4
2	6	9
4	3	12
4	5	11
5	7	3
5	7	10
5	2	10
5	1	11

Room ID	Room
1	Kitchen
2	Garage
3	Family Room
4	Dining Room
5	Bedroom
6	Bathroom
7	Closet
8	Patio

Item ID	Item
1	Clothes
2	Jackets
3	Hangers
4	Towels
5	Dishes
6	Pots & Pans
7	Utensils
8	Silverware
9	Toiletries
10	Shoes
11	Books
12	Games
13	Lawn Mower

# Anatomy of a SELECT JOIN Statement

SELECT

    <column,...>

FROM <table1>

    JOIN <table2>

        ON <table1.column> = <table2.column>

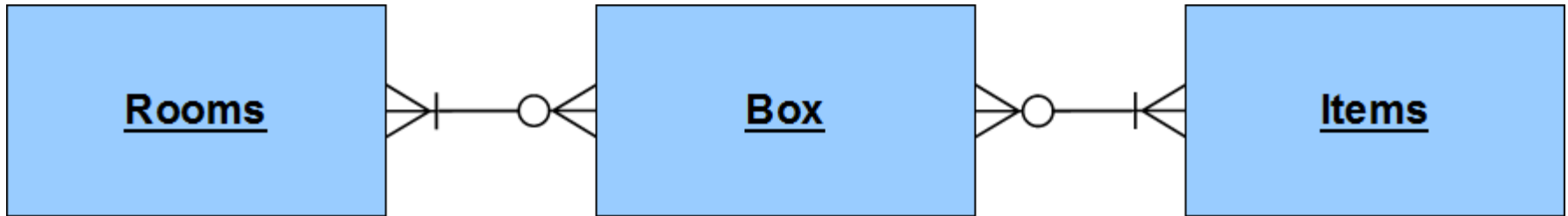
...

WHERE <condition>

ORDER BY <column,...>



# Multiple Related Tables = SQL JOIN



Box	Room ID	Item ID
1	7	3
1	7	2
1	7	10
2	6	4
2	6	9
4	3	12
4	5	11
5	7	3
5	7	10
5	2	10
5	1	11

Room ID	Room
1	Kitchen
2	Garage
3	Family Room
4	Dining Room
5	Bedroom
6	Bathroom
7	Closet
8	Patio

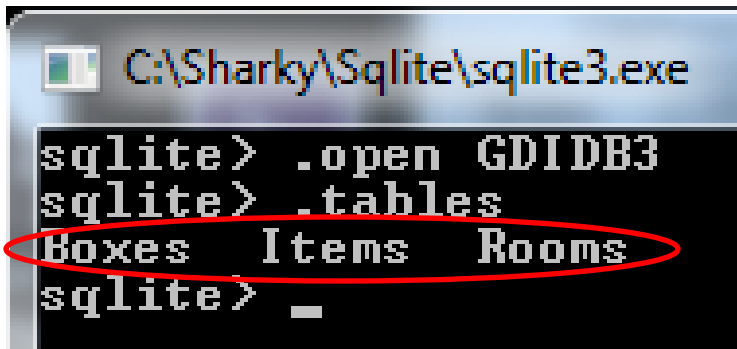
Item ID	Item
1	Clothes
2	Jackets
3	Hangers
4	Towels
5	Dishes
6	Pots & Pans
7	Utensils
8	Silverware
9	Toiletries
10	Shoes
11	Books
12	Games
13	Lawn Mower

# Sqlite housekeeping

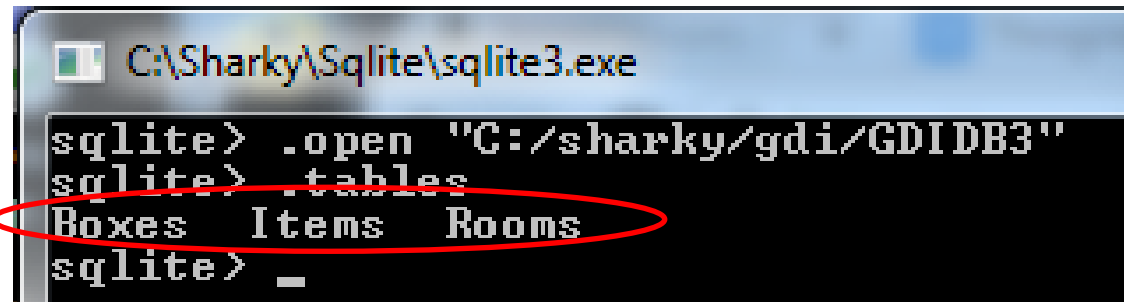
```
sqlite> .open <path>GDIDB3
```

```
sqlite> .tables
```

To use full path - use '/' and must be in quotes!  
.open "C:/sharky/gdi/GDIDB3"



```
C:\Sharky\Sqlite\sqlite3.exe
sqlite> .open GDIDB3
sqlite> .tables
Boxes Items Rooms
sqlite> _
```



```
C:\Sharky\Sqlite\sqlite3.exe
sqlite> .open "C:/sharky/gdi/GDIDB3"
sqlite> .tables
Boxes Items Rooms
sqlite> _
```

# SQL JOIN Clause

sqlite> **SELECT \* FROM Boxes b**

**...> JOIN Rooms r ON b.roomID = r.roomID;**

C:\Sharky\Sqlite\sqlite3.exe

```
sqlite> SELECT * from Boxes b
...> JOIN Rooms r on b.roomID = r.roomID;
```

Box	RoomID	ItemID	RoomID	Room
1	7	3	7	Closet
1	7	2	7	Closet
1	7	10	7	Closet
2	6	4	6	Bathroom
2	6	9	6	Bathroom
4	3	12	3	Family Roo
4	5	11	5	Bedroom
5	7	3	7	Closet
5	7	10	7	Closet
5	2	10	2	Garage
5	1	11	1	Kitchen

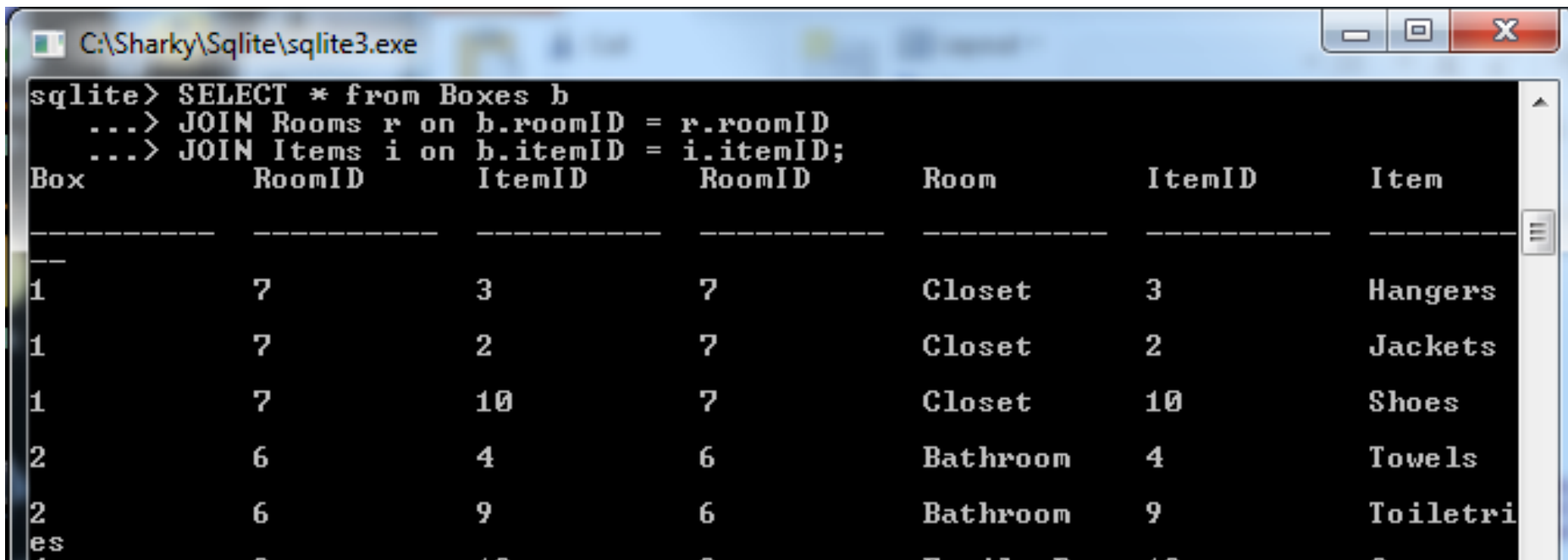
sqlite> \_

# SQL JOIN Clause

sqlite> **SELECT \* FROM Boxes b**

**...> JOIN Rooms r ON b.roomID = r.roomID**

**...> JOIN Items i ON b.itemID = i.itemID;**



The screenshot shows a SQLite command window titled "C:\Sharky\Sqlite\sqlite3.exe". The command prompt shows the following SQL query being executed:

```
sqlite> SELECT * from Boxes b
...> JOIN Rooms r on b.roomID = r.roomID
...> JOIN Items i on b.itemID = i.itemID;
```

The results are displayed in a table with the following columns: Box, RoomID, ItemID, Room, ItemID, and Item. The data is as follows:

Box	RoomID	ItemID	Room	ItemID	Item
1	7	3	Closet	3	Hangers
1	7	2	Closet	2	Jackets
1	7	10	Closet	10	Shoes
2	6	4	Bathroom	4	Towels
2	6	9	Bathroom	9	Toiletri

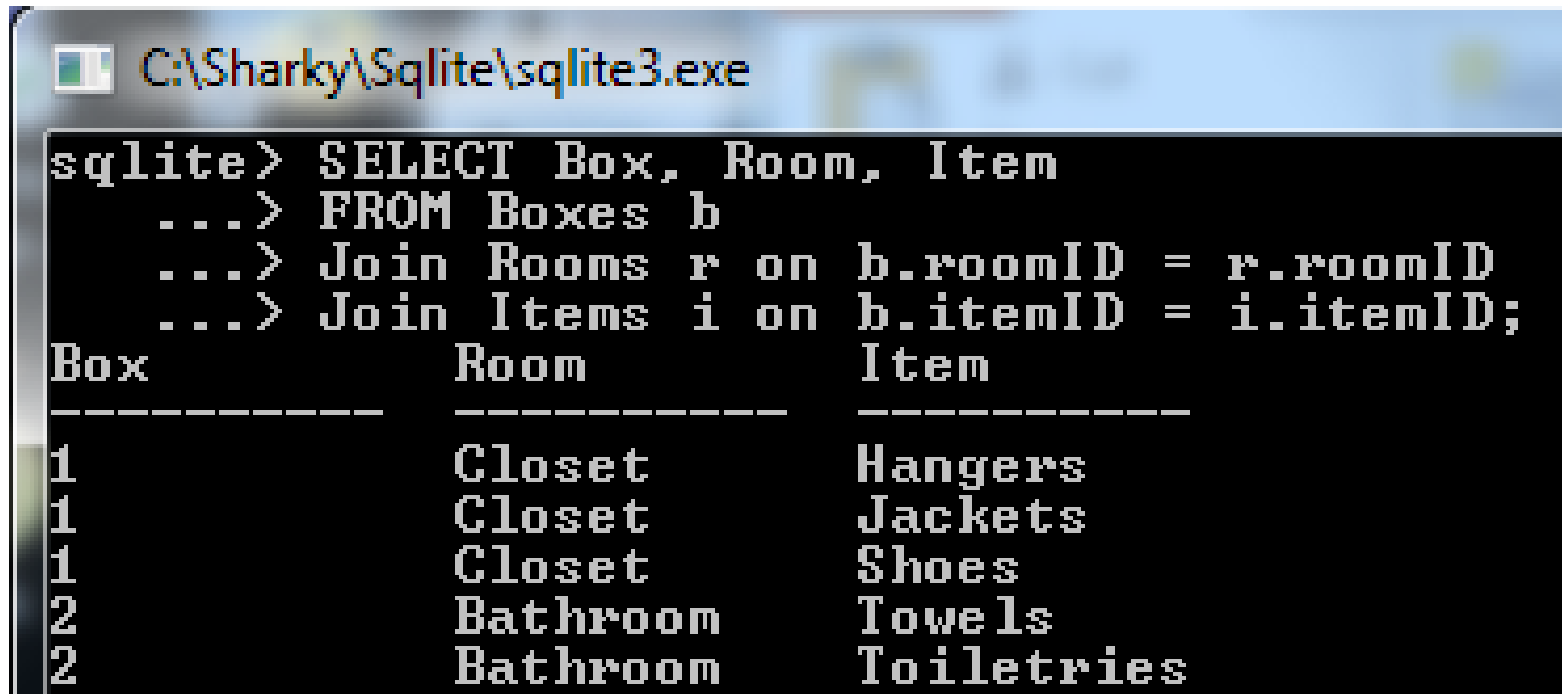
# SQL JOIN Clause

sqlite> **SELECT Box, Room, Item**

**...> FROM Boxes b**

**...> JOIN Rooms r ON b.roomID = r.roomID**

**...> JOIN Items i ON b.itemID = i.itemID;**



The screenshot shows a SQLite3 command prompt window titled "C:\Sharky\Sqlite\sqlite3.exe". The user has entered the following SQL query:

```
sqlite> SELECT Box, Room, Item
...> FROM Boxes b
...> Join Rooms r on b.roomID = r.roomID
...> Join Items i on b.itemID = i.itemID;
```

The results of the query are displayed in a table format with three columns: Box, Room, and Item. The data is as follows:

Box	Room	Item
1	Closet	Hangers
1	Closet	Jackets
1	Closet	Shoes
2	Bathroom	Towels
2	Bathroom	Toiletries

# Anatomy of a CREATE TABLE Statement

```
CREATE TABLE <table name> (  
    <column name> <data type> <constraint>,  
    ...  
);
```

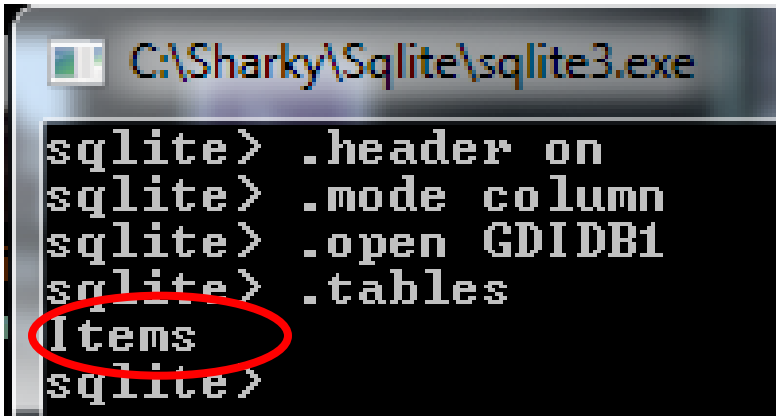
# Sqlite housekeeping

```
sqlite> .open GDIDB1
```

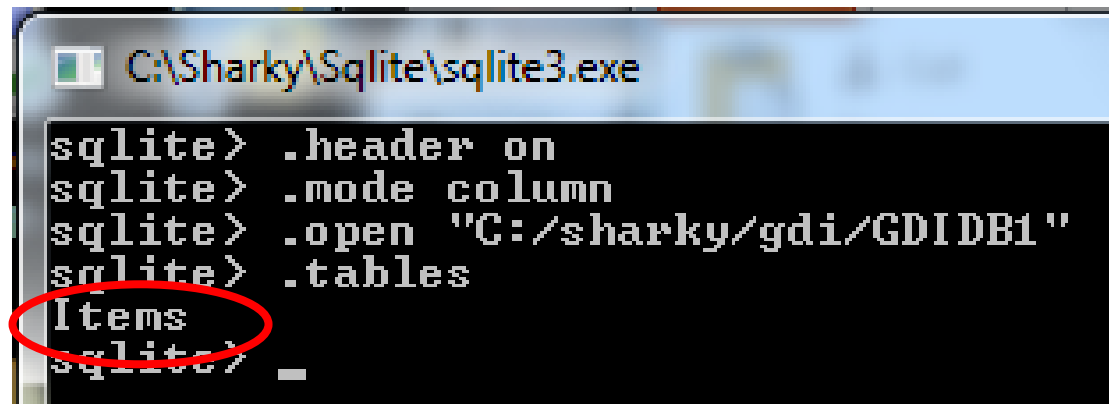
```
sqlite> .tables
```

To use full path - use '/' and must be in quotes!

```
.open "C:/sharky/gdi/GDIDB1"
```



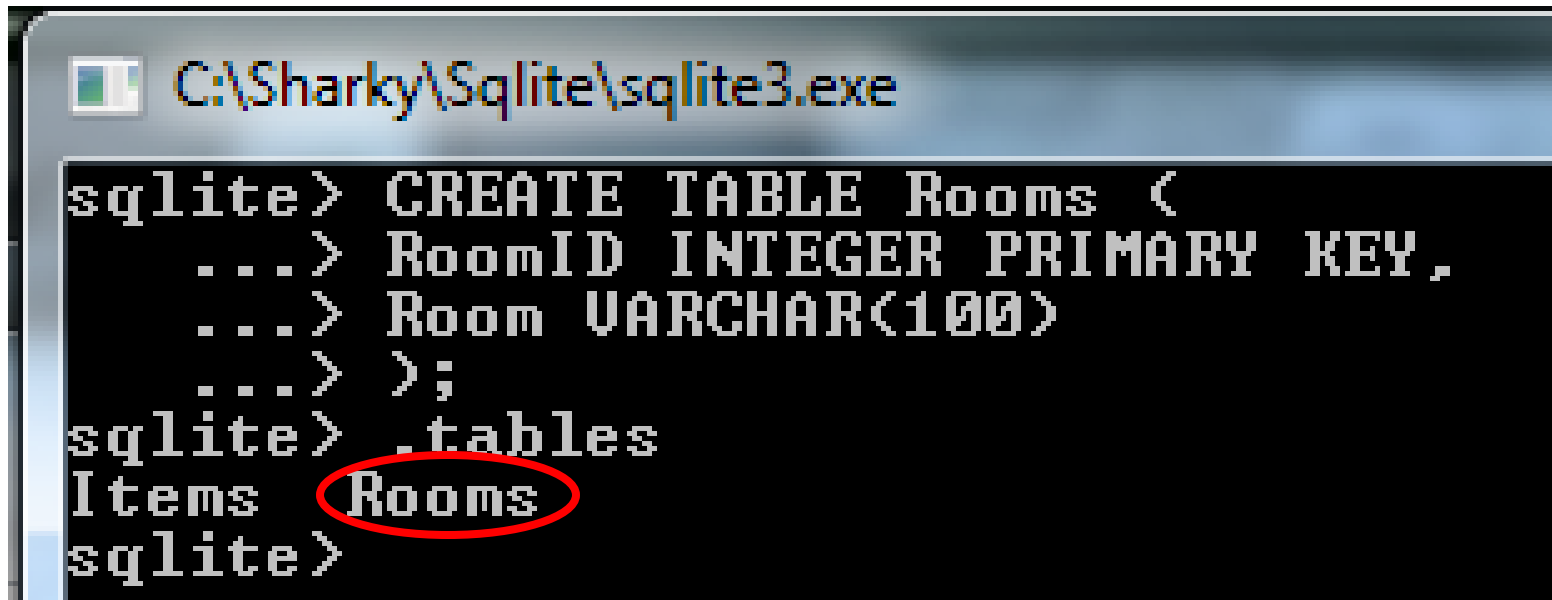
```
C:\Sharky\Sqlite\sqlite3.exe
sqlite> .header on
sqlite> .mode column
sqlite> .open GDIDB1
sqlite> .tables
Items
sqlite>
```



```
C:\Sharky\Sqlite\sqlite3.exe
sqlite> .header on
sqlite> .mode column
sqlite> .open "C:/sharky/gdi/GDIDB1"
sqlite> .tables
Items
sqlite> _
```

# CREATE TABLE Statement

```
sqlite> CREATE TABLE Rooms (  
...> RoomID INTEGER PRIMARY KEY,  
...> Room VARCHAR(100)  
...> );
```



```
C:\Sharky\Sqlite\sqlite3.exe  
sqlite> CREATE TABLE Rooms (  
...> RoomID INTEGER PRIMARY KEY,  
...> Room VARCHAR(100)  
...> );  
sqlite> .tables  
Items Rooms  
sqlite>
```

The screenshot shows a SQLite command prompt window. The title bar indicates the path is C:\Sharky\Sqlite\sqlite3.exe. The command history shows the successful execution of the CREATE TABLE statement for the 'Rooms' table. The subsequent '.tables' command output lists 'Items' and 'Rooms', with 'Rooms' circled in red to highlight its successful creation.



# Anatomy of an INSERT Statement

```
INSERT INTO <table name> (column,...)  
VALUES (values,...);
```

*Data for a single row only*

# INSERT Statement

```
sqlite> INSERT INTO Rooms (Room)
```

```
...> VALUES ('Kitchen');
```

```
sqlite> SELECT * FROM Rooms;
```



```
C:\Sharky\Sqlite\sqlite3.exe

sqlite> INSERT INTO Rooms (Room)
...> VALUES ('Kitchen');
sqlite> SELECT * FROM Rooms;
RoomID      ROOM
-----
1           Kitchen
sqlite>
```

# SQL Statements

- ✓ SELECT – Retrieve (SELECT) data from a table.
- ✓ INSERT – Add (INSERT) data into a table.
- DELETE – Remove (DELETE) data from a table.
- ✓ CREATE – Add (CREATE) a table, or other object.
- DROP – Remove (DROP) a table, or other object.
- EXEC – Run (EXECUTE) a Stored Procedure.

