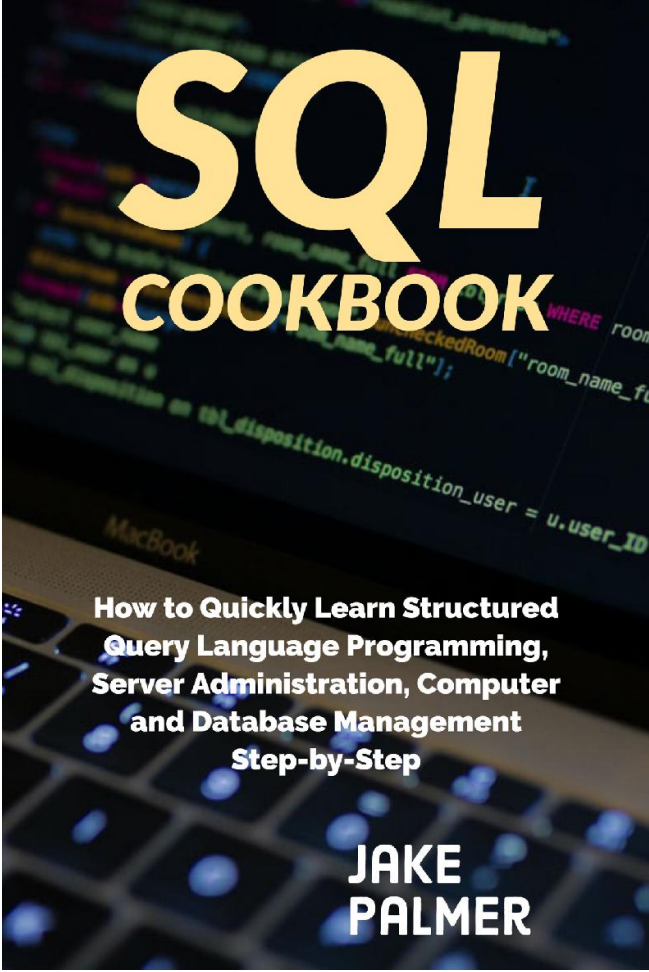


# SQL COOKBOOK

**How to Quickly Learn Structured  
Query Language Programming,  
Server Administration, Computer  
and Database Management  
Step-by-Step**

**JAKE  
PALMER**



# SQL COOKBOOK

**How to Quickly Learn Structured  
Query Language Programming,  
Server Administration, Computer  
and Database Management  
Step-by-Step**

**JAKE  
PALMER**

# SQL Cookbook

How to Quickly Learn Structured Query Language  
Programming, Server Administration, Computer  
and Database Management Step-by-Step

---

**[Jake Palmer]**

### **Text Copyright © [Jake Palmer]**

All rights reserved. No part of this guide may be reproduced in any form without permission in writing from the publisher except in the case of brief quotations embodied in critical articles or reviews.

### **Legal & Disclaimer**

The information contained in this book and its contents is not designed to replace or take the place of any form of medical or professional advice; and is not meant to replace the need for independent medical, financial, legal or other professional advice or services, as may be required. The content and information in this book has been provided for educational and entertainment purposes only.

The content and information contained in this book has been compiled from sources deemed reliable, and it is accurate to the best of the Author's knowledge, information and belief. However, the Author cannot guarantee its accuracy and validity and cannot be held liable for any errors and/or omissions. Further, changes are periodically made to this book as and when needed. Where appropriate and/or necessary, you must consult a professional (including but not limited to your doctor, attorney, financial advisor or such other professional advisor) before using any of the suggested remedies, techniques, or information in this book.

Upon using the contents and information contained in this book, you agree to hold harmless the Author from and against any damages, costs, and expenses, including any legal fees potentially resulting from the application of any of the information provided by this book. This disclaimer applies to any loss, damages or injury caused by the use and application, whether directly or indirectly, of any advice or information presented, whether for breach of contract, tort, negligence, personal injury, criminal intent, or under any other cause of action.

You agree to accept all risks of using the information presented inside this book.

You agree that by continuing to read this book, where appropriate and/or necessary, you shall consult a professional (including but not limited to your doctor, attorney, or financial advisor or such other advisor as needed)

before using any of the suggested remedies, techniques, or information in  
this book.

# Table of Contents

## Book

### Introduction

What is SQL?

Installing and configuring MySql on your system

### Chapter 1: Working with Internet-based database systems

Benefits of Working with SQL

High speed

Standards that are well defined

You Do Not Need to Code

Object-oriented DBMS

You Can Earn A Lot of Money

All Types of Technology Uses SQL

Employers Look for SQL Skills

You Always Obtain an Answer

The Size of a File Never limits you

Reports Are Easy to Create

### Chapter 2: SQL Joins and Union

### Chapter 3: Data Types in SQL

### Chapter 4: The Syntax of SQL

### Chapter 5: SQL Transactions

### Chapter 6: Sub Queries That Are Done Inside of SQL

### Chapter 7: Four Tips That Make Using SQL Easier!

### Chapter 8: How to Manage Objects

What is the schema?

[How to create a new table](#)

[How to create a table with one that already exists](#)

[How to drop tables](#)

[\*\*Chapter 9: MS SQL Server\*\*](#)

[\*\*Chapter 10: Database Operators\*\*](#)

[\*\*Chapter 11: Database Security\*\*](#)

[\*\*Chapter 12: Real-World Uses\*\*](#)

[SQL in an Application](#)

[\*\*Chapter 13: How SQL Injections Can Destroy Databases\*\*](#)

[\*\*Chapter 14: Additional Pointers in Using SQL\*\*](#)

[\*\*Chapter 15: SQL Quiz\*\*](#)

[\*\*Conclusion\*\*](#)

# Introduction

Anything that stores data records is called a database. It can be a file, CD, hard disk, or any number of storage solutions. From a programming point of view, a database is a methodically structured repository of indexed data information that can be easily accessed by the users for creating, retrieving, updating and deleting information. Data can be stored in many forms. Most applications require a database for storing information.

A database can be of two types: (1) flat database and (2) relational database. As, the name suggests a flat database has a two dimensional structure that has data fields and records stored in one large table. It is not capable of storing complex information, which creates a need for relational databases. A relational database stores data in several tables that are related to each other.

Let's take the example of a school. A school will have to maintain data for several students. To find information for a student, we will first ask the class name. After the class name, we will ask for the first name. However, if there are two children with the same first name, then we will ask for the surname. If there are two children with identical names, we can still discriminate the information related to them based on their student id, parents name, date of birth, siblings in same school, etc. This is all related information. When all of this information is stored on paper, it takes a lot of time to retrieve it. Relational database allow easy access to all of this information.

Let's suppose Alex is not feeling well in school and the teacher wants to call his parents to come and pick him up. In the traditional way of maintaining information on paper, if the teacher loses the file with Alex's mom's number, she would not be able to contact her. However, if this information is stored in a database, she just needs to go to the administrator. The administrator will search for Alex's home records and within a matter of seconds, the teacher will have the contact details. This will also free her from the burden of maintaining separate records for each child, allowing her to focus more time on other teaching related activities.



## What is SQL?

SQL is a very popular programming language in the world of software development. It allows direct communication with the relational database management system. It acts as a bridge between the application, and the database. If you want to transfer money from your bank to another bank, you log onto the website and fill in the details, then press the Submit button for the work to get processed. Once the button is pressed, the SQL commands (embedded in the application by the developer) are carried out to do the needed tasks. These commands locate your data on the database, and if all the details are correct, the transaction would be carried out.

SQL basics are easy to learn. If you want to learn it for advanced purposes, like setting up relational database management systems, then you will have to learn how these commands can be used to carry out complex functions.

## Installing and configuring MySql on your system

If you already have MySql server installed on your server then you can skip this section.

Here, I am going to provide you the steps required to install MySQL 6.0 on Windows computer. I have Windows 8.1. Here is the link to the software <https://dev.mysql.com/downloads/mysql/6.0.html> . If you are not using Windows then under the heading “MySQL Community Server 5.7.19” you will see the “Select Operating System:” dropdown. Select your Operating system and you will be provided with all the options.

If you are learning on a standalone system and you are a beginner, I would recommend that you install Windows Essentials from <https://mysql-essential.en.uptodown.com/windows> .

While looking for information on MySql you will come across terms like MySQL community server, MySql installer and MySql essentials. As the name suggests Essentials provides just what is essential without any additional components and ideal to start learning. Both installer and community server have full server features, but the community server is only installed online. For the installer, you have to download the package and then install it offline.

Back to MySQL Essential. Once you click the download button, the installer package can be found in the downloads folder by the name of “mysql-essential-6.0.0.msi”. Follow the steps given below for installing the software.

1. Double click on the mysql-essential-6.0.0.msi package to start the installation process. You would see a MySQL Server 6.0 – Setup Wizard Window. To continue with the installation process click on “Next button”.
2. You will now have to select the Setup Type. Setup can be one of three types:
  - 1. Typical for general use**
  2. Complete for installation of all program features
  - 3. Custom to select which programs you want to install.**

You can go ahead with Typical installation. If you want to change the path where the software is installed, you will have to opt for custom installation.

1. For this book, I went with custom installation because I wanted to change the path of installation. Click Next.
2. The next screen will prompt you to login or create a MySQL account. Signing up is not mandatory. You can create an account or click on the “Skip Sign-Up” radio button and click the Next button.
3. You have reached the last screen of the installation process. Before pressing the Finish button ensure that the “Configure the MySQL Server now” check box is clicked.

We now move on to configuration of MySQL. After clicking the “Finish” button, you will be presented with “MySQL Server Instance Configuration Wizard”. If the window does not pop up on its own, click the Start button on the desktop. Look for the Wizard and click on it. Click “Next” on the first screen. Now follow the steps given below:

1. You must first select whether you want to go for detailed or standard configuration. Select “Detailed Configuration”.
2. Next, you have to select the server type. Your selection will influence memory, disk and CPU usage. Go for Server if you are planning to work on a server that is hosting other applications. Click Next.
3. If you have selected Server in the above step, you will be presented a screen where you are asked to set a path for InnoDBdatafile to enable InnoDB database engine. Without making any modifications click on Next.
4. On this screen, you will have to set the approximate number of concurrent connections to the server. Ideally, for general purpose usage, it is best to go with the first option .i.e., “Decision Support(DSS)/OLAP”. Click Next after making your selection.
5. On the next screen, you are presented with two options: (1) Enable TCP/IP Networking and (2) Enable Strict Mode. Check the “Enable TCP/IP Networking” option. The second option “Enable Strict Mode” will be checked by default. Most applications do not prefer this option, so if you do not have a good reason for using it in Strict mode, uncheck this option and click the Next button.
6. You will now be asked to set the default character set used by MySQL. Select “Standard Character Set” and click on Next.
7. On this step, you must set window options. You will see three check boxes : “Install as Windows Service”, “Launch the MySQL Server automatically” and “Include Bin

Directory in Windows PATH” . Check all three boxes and click Next.

8. You will now have to set the root password for the account. Check “Modify Security Settings” and provide your password details. If your server is on the internet then avoid checking “Enable root access from remote machines”. Also, it is not recommended to check the “Create An Anonymous Account” option. Please save the password at a safe place.
9. This window will show you how the configuration is processing. Once the processing is over, the “Finish” button will be enabled, and you can click on it.

Congratulations!! You have successfully installed and configured MySQL on your machine. It is time for action now.

# Chapter 1: Working with Internet-based database systems

The server and client technology are popular for many different businesses in the world. This works well for some companies, but some other things need to be taken care of because of the changes in technology. Many companies allow a user to access an online database from their system. Customers that have an account on the company website can use this database system to update or change the information. These customers can even pay online, check their orders, make their purchases and much more.

Since more companies are setting up their websites, you must ensure that you develop a good website that will give the customer a chance to check the information out. There may be times when you want to include some security information like passwords when a customer enters his or her personal information. Many companies will require a customer to do this, but these companies always give the information out for free.

This system is easy to handle, but there are a few different things that will happen behind the scenes to ensure that the queries work correctly. The customer can use the system and check this information out, but there is a lot of information which the server will need to piece together to ensure that the information will show up properly on the screen. This will also help to enhance the user's experience.

For example, you may notice that the web browser you currently use will use an SQL code or a program that is similar to it. This program is to figure out what type of data the user hopes to see. The browser will use the SQL interface to reach the database once the customer has put in the information they are looking for. The SQL system will look at the query and will then bring the information back onto the website. This information will show up on the web browser, and only the right information will show up on the screen if the system works properly.

## **Benefits of Working with SQL**

Now that you know about the different database systems that you can work with, let us learn about the advantages of using the Structured Query Language. In addition to a variety of coding languages, you also have some database languages, and each of these languages is different when compared to the other. You may also wonder why you should use SQL over other languages. Therefore, it is important that you know the benefits of SQL.

## **High speed**

You should use the SQL system if you want to go through volumes of information quickly. The SQL system can find you a lot of information within a few seconds. You will also find the information that you need. If you work with volumes of data, you should use SQL since it is one of the most efficient options.

## **Standards that are well defined**

If you want to ensure that your database is secure and strong, you should use SQL since it is constantly updated. These updates help to keep the SQL system strong. Other database tools will not have the same standards like SQL, which will affect your analyses. The features in other databases will also make it difficult for you to store all the necessary information.

## **You Do Not Need to Code**

You do not need to code to use the SQL tool. All you need to do is remember a few syntaxes, which we will cover in the later chapters of the book. You, however, do not have to master coding in SQL if you want to use it to perform any analyses on some data.

## **Object-oriented DBMS**

Since you will use a database management system when you work with SQL, it makes it easier for you to find all the information you need, store that information and perform the necessary analysis to make informed decisions.

## **You Can Earn A Lot of Money**

You always want to earn more when you work for an organization. You can certainly get a better salary if you know how to use SQL. You can do this by either nurturing your programming skills in SQL or by learning how to maintain a system and keep it running effectively and efficiently. You can also work as an SQL analyst and provide information and insights for a business; it will help the seniors make better decisions. This will help to maximize the profits for any business.

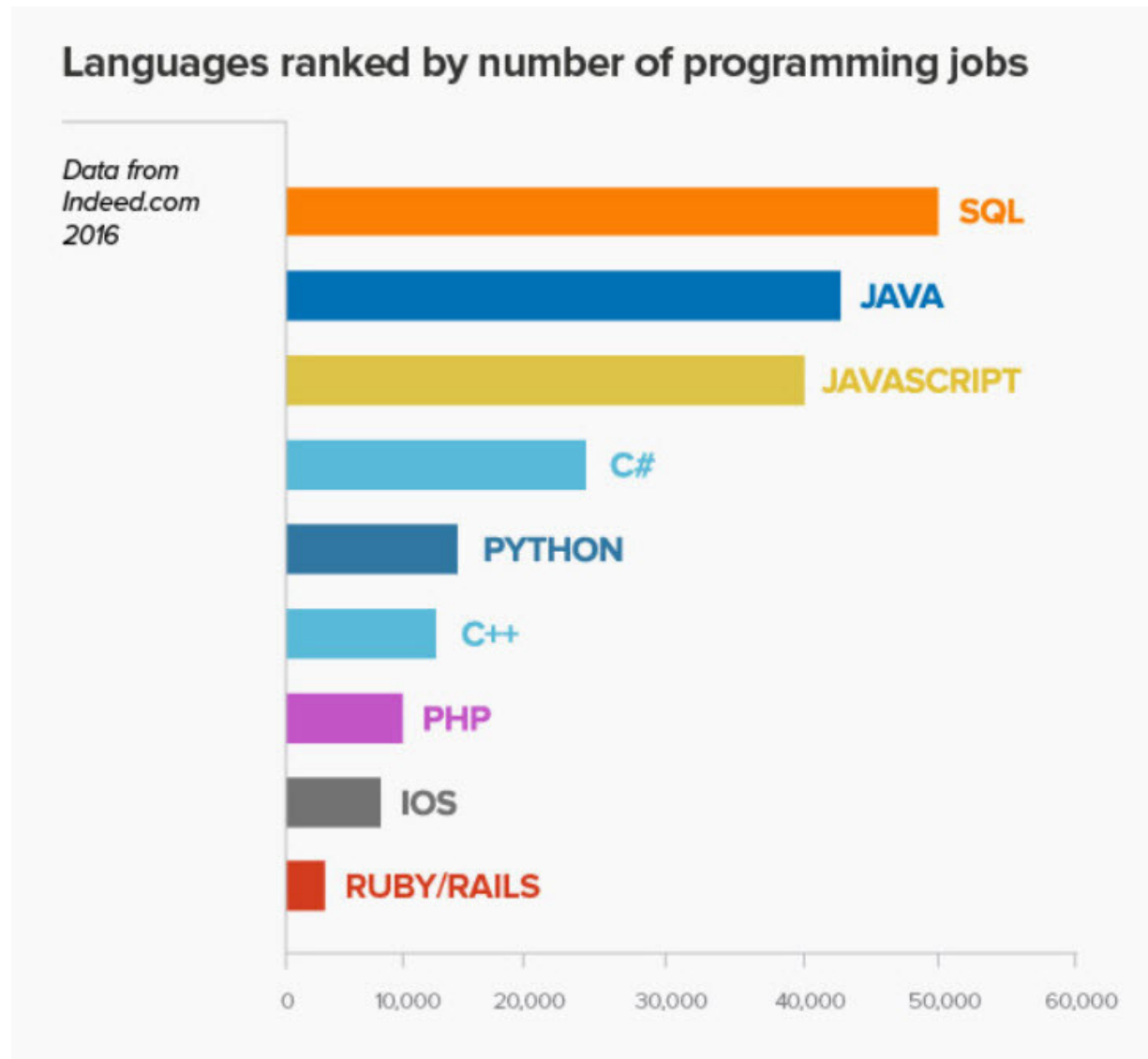
## **All Types of Technology Uses SQL**

Most businesses use database tools and technologies like MySQL, Microsoft SQL Server, and PostgreSQL. You should also remember that most people use SQL at some point in their lives. If you are not aware, you also use SQL on your smartphone.



## Employers Look for SQL Skills

Most employers actively look for people who know how to use SQL. Yes, an employer is willing to pay you more, but he or she also is aware of the benefits of hiring an individual who is skilled at using SQL. If you want to move jobs or change your area of work, you should learn how to code in SQL. You will be one of the most sought-after candidate for the position



## **You Always Obtain an Answer**

You should think about the different questions you have about the data. You may want to know more about the sales from last year. Were your customers satisfied with your product? Have your expenses reduced since last year? You can answer all these questions using SQL. When you identify the database that provides this information, you can use SQL to explore that data to answer your questions. SQL allows you to analyze data in different ways. You do not have to rely only on simple reports or even contact your employers to obtain data. You can become an independent employee when you use SQL.

## **The Size of a File Never limits you**

Have you ever had a problem where a spreadsheet has crashed because you have hundreds or thousands of rows filled with data? You can opt for a relational database to save the data since these databases can save millions of rows and columns of data with ease. SQL will allow you to perform different operations on the data and use that information to make decisions. It is true that MS Excel is a great tool to use, but this tool is not equipped to perform multiple operations on millions of rows of data. You should try to use a relational database to store the information and use SQL to perform any analysis on the data.

## Reports Are Easy to Create

You can save SQL queries easily and re-use them whenever necessary. You can also make changes to the query any time you want to. You can also use comments in the SQL code that makes it easier for you and for anybody else to understand the query. If you work only on spreadsheets, you will need to include long multi-step processes. You will first need to import the data into excel from a report, tabulate it, sort it, and then filter or use some values depending on your need.

If you use SQL, you only need to write the code once, save that code and re-open it whenever you need to produce a report. You will save many hours and days. These are some of the benefits you can enjoy when you work with SQL. Some people claim that the SQL interface is slightly hard to use, and there are some features that you may need to purchase from a third party. In simple words, SQL has many features that make it easy for one to use the language. Since this language has multiple benefits, it is one of the best languages to use to perform analyses on different data sets that you collect for your business.

You may say that it is hype when I ask you to learn SQL, but you should remember that numbers never lie. SQL is an invaluable tool that many employers desire. Since most of the information is now digital, you have more data available to you. All this information is either stored in a database or a data warehouse. If you want to manage these databases or warehouses, you need to learn SQL. If you read a business journal, you will see that most businesses are looking for business intelligence analysts. If an organization wants to do more with the data it has, it should have individuals who know how to access and analyze data. You can do this through SQL.

## Chapter 2: SQL Joins and Union

The select statement can be made to process more than one table at the same time in a single select statement with the use of the logical operators ('OR' and 'AND'). It can sometimes be more efficient to use the Left, right and the inner join operator for more efficient processing.

### 1. SQL INNER JOIN

The SQL 'INNER JOIN' can also be used to process more than one data table in an SQL query or statement. However, the data tables must have relating columns (the primary and its associated foreign key columns). The "INNER JOIN" keyword returns records from two data tables if a match is found between columns in the affected tables. The syntax of usage is as described below: SELECT column-1, column-2... column-n FROM data\_table-1

INNER JOIN data\_table-2

ON data\_table-1.keycolumn = data\_table-2.foreign\_keycolumn For example, to select customer acct\_no, first\_name, surname, sex and account\_bal data from across the two tables 'customer\_accounts' (table 4.1) and 'customer\_personal\_info' (table 4.2) based on matching columns 'cust\_id', we use the following SQL INNER JOIN query:

acct_no	cust_id	acct_bal	acct_type	acct_opening_date
0411003	03231	2540.33	100	2000-11-04
0412007	03146	10350.02	200	2000-09-13
0412010	03347	7500.00	200	2002-12-05

Table 4.1 customer\_accounts

--	--	--	--	--	--	--

cust_id	first_nm	second_nm	lastname	sex	Date_of_birth	addr
03051	Mary	Ellen	Brown	Female	1980-10-19	Coventry
03231	Juan		Maslow	Female	1978-11-18	York
03146	John	Ken	Pascal	Male	1983-07-12	Liverpool
03347	Alan		Basal	Male	1975-10-09	Easton

Table 4.2 customer\_personal\_info SELECT a.acct\_no, b.first\_nm AS first\_name, b. lastname AS surname, b.sex, a.acct\_bal FROM customer\_accounts AS a INNER JOIN customer\_personal\_info AS b ON b.cust\_id = a.cust\_id<sup>[5]</sup>

The above SQL query would produce the result set in table 4.3 below:

acct_no	first_name	surname	sex	acct_bal
0411003	Juan	Maslow	Female	2540.33
0412007	John	Pascal	Male	10350.02
0412010	Alan	Basal	Male	7500.00

Table 4.3

## 2. SQL RIGHT JOIN

When used with the SQL select statement, The “RIGHT JOIN” keyword includes all records in the right data table even when no matching records are found in the left data table. The syntax of usage is as described below: SELECT column-1, column-2... column-n FROM left-data\_table RIGHT

JOIN right-data\_table ON left-data\_table.keyColumn = right-data\_table.foreign\_keycolumn For example, to select customers' acct\_no, first\_name, surname, sex and account\_bal details across the two tables, customer\_accounts and customer\_personal\_info display customer personal information whether they have an active account or not. We use the following SQL 'RIGHT JOIN' query: SELECT a.acct\_no, b.first\_nm AS first\_name, b.lastname AS surname, b.sex, a.acct\_bal FROM customer\_accounts AS a RIGHT JOIN customer\_personal\_info AS b ON b.cust\_id = a.cust\_id<sup>[6]</sup>

The output of execution of the above SQL RIGHT JOIN query is presented in table 4.4 below:

acct_no	first_name	surname	sex	acct_bal
0411003	Juan	Maslow	Female	2540.33
0412007	John	Pascal	Male	10350.02
0412010	Alan	Basal	Male	7500.00
	Mary	Brown	Female	

Table 4.4

### 3. SQL LEFT JOIN

The 'LEFT JOIN' is used with the SQL select statement to return all records in the left data table (first table) even if there were no matches found in the relating right table (second table). The syntax of usage is as described below: SELECT column-1, column-2... column-n FROM left-data\_table LEFT JOIN right-data\_table ON left-data\_table.keyColumn = right-data\_table.foreign\_keycolumn For example, to display all active accounts holders' details across the 'customer\_accounts and 'customer\_personal\_info' tables, we use the following SQL 'LEFT JOIN' query: SELECT a.acct\_no, b.first\_nm AS first\_name, b.lastname AS surname, b.sex, a.acct\_bal FROM customer\_accounts AS a LEFT JOIN customer\_personal\_info AS b ON b.cust\_id = a.cust\_id<sup>[7]</sup>

The above SQL LEFT JOIN query would produce the result set in table 4.5 below:

---

acct_no	first_name	surname	sex	acct_bal
0411003	Juan	Maslow	Female	2540.33
0412007	John	Pascal	Male	10350.02
0412010	Alan	Basal	Male	7500.00

Table 4.5

## 4. SQL UNION Command

Two or more SQL select statements' result sets can be joined with the 'UNION' command. The syntax of usage is as described below: SELECT column-1, column-2... column-n FROM data\_table-1

UNION

SELECT column-1, column-2... column-n FROM data\_table-2

Source<sup>[8]</sup>

For example, to display a list of our phantom bank's customers in UK and US branches, you may display one record for customers that have accounts in both branches according to the following tables 4.6 and 4.7:

acct_no	first_name	surname	sex	acct_bal
0411003	Juan	Maslow	Female	2540.33
0412007	John	Pascal	Male	10350.02
0412010	Alan	Basal	Male	7500.00

Table 4.6 London\_Customers

acct_no	first_name	surname	sex	acct_bal
0413112	Deborah	Johnson	Female	4500.33
0414304	John	Pascal	Male	13360.53
0414019	Rick	Bright	Male	5500.70
0413014	Authur	Warren	Male	220118.02

Table 4.7 Washington\_Customers We use the following SQL query:  
SELECT first\_name, surname, sex, acct\_bal FROM London\_Customers  
UNION

SELECT first\_name, surname, sex, acct\_bal FROM  
Washington\_Customers<sup>[9]</sup>

The following is the result set from the execution of the above SQL query:

first_name	surname	sex	acct_bal
Juan	Maslow	Female	2540.33
John	Pascal	Male	10350.02
Alan	Basal	Male	7500.00
Deborah	Johnson	Female	4500.33
Rick	Bright	Male	5500.70
Authur	Warren	Male	220118.02

Table 4.8 SQL UNION

Note that the record for a customer (John Pascal) is only listed once, even if he is a customer of the two branches. This is because the UNION command lists only distinct records across tables. So, to display all records across associated tables, use 'UNION ALL' instead.

## 5. SQL UNION ALL Command

The UNION ALL command is basically the same as the UNION command, except that its displays all records across unionized tables, as explained earlier. The syntax of usage is as described below: SELECT column-1, column-2... column-n FROM data\_table-1

UNION

SELECT column-1, column-2... column-n FROM data\_table-2

Apply the 'UNION ALL' command on the data of the two tables 'London\_Customers' and 'Washington\_Customers' with the SQL query



below:   SELECT   first\_name,   surname,   sex,   acct\_bal   FROM  
London\_Customers UNION ALL  
SELECT       first\_name,       surname,       sex,       acct\_bal       FROM  
Washington\_Customers<sup>[10]</sup>

The records would then be displayed in two tables, as shown below:

first_name	surname	sex	acct_bal
Juan	Maslow	Female	2540.33
John	Pascal	Male	10350.02
Alan	Basal	Male	7500.00
Deborah	Johnson	Female	4500.33
John	Pascal	Male	13360.53
Rick	Bright	Male	5500.70
Authur	Warren	Male	220118.02

Table 4.9

## Chapter 3: Data Types in SQL

Data is at the core of SQL. After all, SQL was created to make it easier to manipulate the data stored inside a database. It ensures that you do not have to sort through large chunks of data manually in search of what you want. Now, there are various types of data that can be stored in the database depending on the platform used. As such, you now need to learn about the data types available in SQL.

**Data Types: An Introduction** The data type specifies the kind of data which can be stored in a column of a database table. When creating a table, it is important to decide the data type which is going to be used for defining the columns. Data types can also be used for defining variables and for storing procedure output and input parameters. The data type will instruct SQL to expect a particular kind of data in each column.

A data type must be selected for each variable or column which is appropriate for the kind of data which is to be stored in that column or variable. Additionally, storage requirements must be considered. You need to select data types which ensure efficiency in the storage.

Selecting the correct data type for the variables, tables and stored procedures will improve the performance greatly as it will ensure that the execution plan is correct. At the same time, it will be a great improvement on data integrity as it ensures that the right data has been stored inside a database.

**The List of Data Types** There are several data types used in SQL. The following table lists all of them along with a short description of what they are. Mark

this table as it will prove to be invaluable as a reference guide on data types when you are learning and even later.

Before you start, you should take a moment to understand what precision and scale are. Precision is the total number of digits that is present in a number. Scale is the total number of digits located on the right side of the decimal point of a number. In the case of a number like 123.4567, the precision is 7 while the scale is 4.

INTEGER (p)	Integer numerical (no decimal). Precision p
SMALLINT	Integer numerical (no decimal). Precision 5
INTEGER	Integer numerical (no decimal). Precision 10
BIGINT	Integer numerical (no decimal). Precision 19
DECIMAL (p,s)	Exact numerical, precision p, scale s. Example: decimal(5,2) is a number that has 3 digits before the decimal and 2 digits after the decimal
NUMERIC (p,s)	Exact numerical, precision p, scale s. (Same as DECIMAL)
FLOAT(p)	Approximate numerical, mantissa precision p. A floating number in base 10 exponential notation. The size argument for this type consists of a single number specifying the minimum precision
REAL	Approximate numerical, mantissa precision 7
FLOAT	Approximate numerical, mantissa precision 16
DOUBLE PRECISION	Approximate numerical, mantissa precision 16
DATE	Stores year, month, and day values

TIME	Stores hour, minute, and second values
TIMESTAMP	Stores year, month, day, hour, minute, and second values
INTERVAL	Composed of a number of integer fields, representing a period of time, depending on the type of interval
ARRAY	A set-length and ordered collection of elements
MULTISET	A variable-length and unordered collection of elements
XML	Stores XML data

**Variations among Database Platforms** The data types given above are common to all database platforms. However, there are some which are known by different names in different database platforms. This can be a source of confusion. As such, take a look at the variations in the table given below.

<b>Data Type</b>	<b>Access</b>	<b>SQLServer</b>	<b>Oracle</b>	<b>MySQL</b>	<b>PostgreSQL</b>
<i>boolean</i>	Yes/No	Bit	Byte	N/A	Boolean
<i>integer</i>	Number (integer)	Int	Number	Int Integer	Int Integer
<i>float</i>	Number (single)	Float Real	Number	Float	Numeric
<i>currency</i>	Currency	Money	N/A	N/A	Money
<i>string (fixed)</i>	N/A	Char	Char	Char	Char
<i>string (variable)</i>	Text (<256) Memo (65k+)	Varchar	Varchar Varchar 2	Varchar	Varchar
<i>binary object</i>	OLE Object Memo	Binary (fixed up to 8K) Varbinary (<8K) Image (<2GB)	Long Raw	Blob Text	Binary Varbinary

Data Types for Different Databases Each database platform tends to have its own range of data types. As such, you need to know what those data types are in order to use those platforms effectively. Here, we shall be focusing on the most popular database platforms: MySQL, SQL Server and Microsoft Access. A short description has been provided with each data type.

## Data Types in Microsoft Access

Data Type	Description
Text	Use for text or combinations of text and numbers. 255 characters' maximum
Memo	Memo is used for larger amounts of text. Stores up to 65,536 characters. Note: You cannot sort a memo field. However, they are searchable
Byte	Allows whole numbers from 0 to 255. Storage is 1 byte.
Integer	Allows whole numbers between -32,768 and 32,767. Storage is 2 bytes.
Long	Allows whole numbers between -2,147,483,648 and 2,147,483,647. Storage is 4 bytes.
Single	Single precision floating-point. Will handle most decimals. Storage is 4 bytes.
Double	Double precision floating-point. Will handle most decimals. Storage is 8 bytes.
Currency	Use for currency. Holds up to 15 digits of whole dollars, plus 4 decimal places. Tip: You can choose which country's currency to use. Storage is 8 bytes.
AutoNumber	AutoNumber fields automatically give each record its own number, usually starting at 1. Storage is 4 bytes.
Data/Time	Use for dates and times. Storage is 8 bytes.
Yes/No	A logical field can be displayed as Yes/No, True/False, or On/Off. In code, use the constants True and False (equivalent to -1 and 0). Note: Null values are not allowed in Yes/No fields. Storage is 1 bit.
Ole Object	Can store pictures, audio, video, or other BLOBs (Binary Large Objects). Storage is up to 1GB.
Hyperlink	Contain links to other files, including web pages.
Lookup Wizard	Let you type a list of options, which can then be chosen from a drop-down list. Storage is 4 bytes

## Data Types in MySQL

In MySQL, the data types available can be widely classified into 3 categories. They are text, date/time and number. We shall take a look at the available data types in each category in the following tables.

### MySQL Data Types



<b>Data Type (Text)</b>	<b>Description</b>
CHAR(size)	Holds a fixed length string (can contain letters, numbers, and special characters). The fixed size is specified in parenthesis. Can store up to 255 characters.
VARCHAR(size)	Holds a variable length string (can contain letters, numbers, and special characters). The maximum size is specified in parenthesis. Can store up to 255 characters. Note: If you put a greater value than 255 it will be converted to a TEXT type
TINYTEXT	Holds a string with a maximum length of 255 characters
TEXT	Holds a string with a maximum length of 65,535 characters
BLOB	For BLOBs (Binary Large Objects). Holds up to 65,535 bytes of data
MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
MEDIUMBLOB	For BLOBs (Binary Large Objects). Holds up to 16,777,215 bytes of data
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters
LOBLOB	For BLOBs (Binary Large Objects). Holds up to 4,294,967,295 bytes of data
ENUM(x,y,z,etc.)	Let you enter a list of possible values. You can list up to 65535 values in an ENUM list. If a value is inserted that is not in the list, a blank value will be inserted. Note: The values are sorted in the order you enter them. You enter the possible values in this format: ENUM('X','Y','Z')
SET	Similar to ENUM except that SET may contain up to 64 list items and can store more than one choice

## Integers:

In MySQL, integer data types will have an extra option known as UNSIGNED. Generally, an integer will move to a positive value from a negative one. However, the addition of the UNSIGNED attribute is going to move up that range. As a result, the integer will start at zero and not a negative number.



<b>Data Type (Number)</b>	<b>Description</b>
TINYINT(size)	-128 to 127 normal. 0 to 255 UNSIGNED*. The maximum number of digits may be specified in parenthesis
SMALLINT(size)	-32768 to 32767 normal. 0 to 65535 UNSIGNED*. The maximum number of digits may be specified in parenthesis
MEDIUMINT(size)	-8388608 to 8388607 normal. 0 to 16777215 UNSIGNED*. The maximum number of digits may be specified in parenthesis
INT(size)	-2147483648 to 2147483647 normal. 0 to 4294967295 UNSIGNED*. The maximum number of digits may be specified in parenthesis
BIGINT(size)	-9223372036854775808 to 9223372036854775807 normal. 0 to 18446744073709551615 UNSIGNED*. The maximum number of digits may be specified in parenthesis
FLOAT(size,d)	A small number with a floating decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter
DOUBLE(size,d)	A large number with a floating decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter
DECIMAL(size,d)	A DOUBLE stored as a string, allowing for a fixed decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter

**Date/Time:** The two data types, **TIMESTAMP** and **DATETIME** may return the same format. However, they work in different ways. **TIMESTAMP** will set itself automatically to the current time and date in an **UPDATE** or **INSERT** query. Different formats are accepted by **TIMESTAMP** as well.



<b>Data Type (Date)</b>	<b>Description</b>
DATE()	A date. Format: YYYY-MM-DD Note: The supported range is from '1000-01-01' to '9999-12-31'
DATETIME()	*A date and time combination. Format: YYYY-MM-DD HH:MI:SS Note: The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'
TIMESTAMP()	*A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD HH:MI:SS Note: The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC
TIME()	A time. Format: HH:MI:SS Note: The supported range is from '-838:59:59' to '838:59:59'
YEAR()	A year in two-digit or four-digit format. Note: Values allowed in four-digit format: 1901 to 2155. Values allowed in two-digit format: 70 to 69, representing years from 1970 to 2069

**Data Types in SQL Server Like MySQL, the data types in SQL Server can also be classified into different categories. We shall be looking at each one in the following tables.**

## String:

Data Type (String)	Description	Storage
char(n)	Fixed width character string. Maximum 8,000 characters	Defined width
varchar(n)	Variable width character string. Maximum 8,000 characters	2 bytes + number of chars
varchar(max)	Variable width character string. Maximum 1,073,741,824 characters	2 bytes + number of chars
text	Variable width character string. Maximum 2GB of text data	4 bytes + number of chars
nchar	Fixed width Unicode string. Maximum 4,000 characters	Defined width x 2
nvarchar	Variable width Unicode string. Maximum 4,000 characters	
nvarchar(max)	Variable width Unicode string. Maximum 536,870,912 characters	
ntext	Variable width Unicode string. Maximum 2GB of text data	
bit	Allows 0, 1, or NULL	
binary(n)	Fixed width binary string. Maximum 8,000 bytes	
varbinary	Variable width binary string. Maximum 8,000 bytes	
varbinary(max)	Variable width binary string. Maximum 2GB	
image	Variable width binary string. Maximum 2GB	

## Number:



<b>Data Type (Number)</b>	<b>Description</b>	<b>Storage</b>
tinyint	Allows whole numbers from 0 to 255	Stores up to 1 byte
smallint	Allows whole numbers between -32,768 and 32,767	Stores up to 2 bytes
int	Allows whole numbers between -2,147,483,648 and 2,147,483,647	Stores up to 4 bytes
bigint	Allows whole numbers between -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807	Stores up to 8 bytes
decimal(p,s)	<p>Fixed precision and scale numbers.</p> <p>Allows numbers from -<math>10^{38} + 1</math> to <math>10^{38} - 1</math>.</p> <p>The p parameter indicates the maximum total number of digits that can be stored (both to the left and to the right of the decimal point). p must be a value from 1 to 38. Default is 18.</p> <p>The s parameter indicates the maximum number of digits stored to the right of the decimal point. s must be a value from 0 to p. Default value is 0</p>	Stores between 5-17 bytes

numeric(p,s)	<p>Fixed precision and scale numbers. Allows numbers from -<math>10^{38} + 1</math> to <math>10^{38} - 1</math>.</p> <p>The p parameter indicates the maximum total number of digits that can be stored (both to the left and to the right of the decimal point). p must be a value from 1 to 38. Default is 18.</p> <p>The s parameter indicates the maximum number of digits stored to the right of the decimal point. s must be a value from 0 to p. Default value is 0</p>	Stores between 5-17 bytes
smallmoney	Monetary data from -214,748.3648 to 214,748.3647	Stores up to 4 bytes
money	Monetary data from -922,337,203,685,477.5808 to 922,337,203,685,477.5807	Stores up to 8 bytes
float(n)	<p>Floating precision number data from <math>-1.79E + 308</math> to <math>1.79E + 308</math>.</p> <p>The n parameter indicates whether the field should hold 4 or 8 bytes. float(24) holds a 4-byte field and float(53) holds an 8-byte field. Default value of n is 53.</p>	Stores up to either 4 or 8 bytes depending
real	Floating precision number data from $-3.40E + 38$ to $3.40E + 38$	Stores up to 4 bytes

**Date/Time:**



<b>Data Type (Date)</b>	<b>Description</b>	<b>Storage</b>
datetime	From January 1, 1753 to December 31, 9999 with an accuracy of 3.33 milliseconds	Stores up to 8 bytes
datetime2	From January 1, 0001 to December 31, 9999 with an accuracy of 100 nanoseconds	Stores between 6-8 bytes
smalldatetime	From January 1, 1900 to June 6, 2079 with an accuracy of 1 minute	Stores up to 4 bytes
date	Store a date only. From January 1, 0001 to December 31, 9999	Stores up to 3 bytes
time	Store a time only to an accuracy of 100 nanoseconds	Stores between 3-5 bytes
datetimeoffset	The same as datetime2 with the addition of a time zone offset	Stores between 8-10 bytes
timestamp	Stores a unique number that gets updated every time a row gets created or modified. The timestamp value is based upon an internal clock and does not correspond to real time. Each table may have only one timestamp variable.	

The above tables will certainly prove to be of immense help in your work. You should certainly go through them more than once to ensure that you have a thorough idea of the data types in use in SQL in different database platforms. In fact, it will be a good idea to mark down the chapter so that you can quickly refer to it whenever you need to refresh your knowledge.

## Chapter 4: The Syntax of SQL

Like all computer languages, SQL, too, is governed by a set of guidelines and rules as to how the commands are to be used. You will learn about the syntax of basic SQL commands in this chapter.

There are standards in place that enable SQL to be usable in different databases. Despite the presence of standards, SQL code is unable to enjoy complete portability among the various database systems. Instead, adjustments are necessary to make the code accessible in various databases.

On the other hand, there are certain features which remain applicable irrespective of the database using the code. All statements in SQL will need to start with any one of the following keywords.

- **SELECT**
- **UPDATE**
- **INSERT**
- **ALTER**
- **DELETE**
- **CREATE**
- **DROP**
- **SHOW**
- **USE**

Another important point you must note is that there is no case sensitivity in SQL. Therefore, 'INSERT' and 'insert' will have the same result in a SQL Statement. On the other hand, MySQL will recognize case sensitivity for the table names. You need to remember this point when you are working in MySQL.

### Language Elements in SQL

There are quite a few elements in SQL. Knowing and understanding them will be essential to getting a good grip on the language.

**Clauses:** These are the elemental components of queries and statements. They can be optional in certain cases.

**Expressions:** These components are capable of producing scalar values. Alternatively, they can also produce tables that contain rows and columns of data.

**Predicates:** These elements can specify the conditions which can be evaluated as per the three-valued logic of SQL. As per this logic, known in short as 3VL, there are only three values possible: true, false or unknown. They are used for limiting the effects of queries and statements. Alternatively, they can be used for changing the program flow.

**Queries:** Queries are responsible for retrieving the data based on the criteria specified. They are one of the most important elements in SQL.

**Statements:** They may be used for having a constant effect on data and schemata. Alternatively, they may be used for controlling program flow, transactions, sessions, diagnostics or connections.

In SQL, a statement must also include a semicolon to denote that the statement has been terminated. While this is not necessary for all platforms, it is a standard rule in SQL syntax.

Insignificant whitespace will typically be ignored when present in queries and statements in SQL. This makes it easier to format the code in SQL to enhance readability.

The following illustrates the syntax for a range of common elements in SQL. Note this table as it will be a useful quick reference guide when you are working on your own SQL code.



SQL Statement	Syntax
AND / OR	SELECT column_name(s) FROM table_name WHERE condition AND/OR condition
ALTER TABLE	ALTER TABLE table_name ADD column_name datatype or ALTER TABLE table_name DROP COLUMN column_name
AS (alias)	SELECT column_name AS column_alias FROM table_name or SELECT column_name FROM table_name AS table_alias
BETWEEN	SELECT column_name(s) FROM table_name WHERE column_name BETWEEN value1 AND value2
CREATE DATABASE	CREATE DATABASE database_name
CREATE TABLE	CREATE TABLE table_name (column_name1 data_type, column_name2 data_type, column_name3 data_type, ...)
CREATE INDEX	CREATE INDEX index_name ON table_name (column_name) or CREATE UNIQUE INDEX index_name ON table_name (column_name)
CREATE VIEW	CREATE VIEW view_name AS SELECT column_name(s) FROM table_name WHERE condition
DELETE	DELETE FROM table_name WHERE some_column=some_value or DELETE FROM table_name



	(Note: Deletes the entire table!!) DELETE * FROM table_name (Note: Deletes the entire table!!)
DROP DATABASE	DROP DATABASE database_name
DROP INDEX	DROP INDEX table_name.index_name (SQL Server) DROP INDEX index_name ON table_name (MS Access) DROP INDEX index_name (DB2/Oracle) ALTER TABLE table_name DROP INDEX index_name (MySQL)
DROP TABLE	DROP TABLE table_name
EXISTS	IF EXISTS (SELECT * FROM table_name WHERE id = ?) BEGIN --do what needs to be done if exists END ELSE BEGIN --do what needs to be done if not END
IN	SELECT column_name, aggregate_function(column_name) FROM table_name WHERE column_name operator value GROUP BY column_name SELECT column_name(s) FROM table_name WHERE column_name IN (value1,value2,..)
HAVING	SELECT column_name, aggregate_function(column_name) FROM table_name WHERE column_name operator value GROUP BY column_name HAVING aggregate_function(column_name) operator value
GROUP BY	SELECT column_name, aggregate_function(column_name) FROM table_name WHERE column_name operator value GROUP BY column_name

INSERT INTO	INSERT INTO table_name VALUES (value1, value2, value3,...) or INSERT INTO table_name (column1, column2, column3,...) VALUES (value1, value2, value3,...)
INNER JOIN	SELECT column_name(s) FROM table_name1 INNER JOIN table_name2 ON table_name1.column_name=table_name2.column_name
LEFT JOIN	SELECT column_name(s) FROM table_name1 LEFT JOIN table_name2 ON table_name1.column_name=table_name2.column_name
RIGHT JOIN	SELECT column_name(s) FROM table_name1 RIGHT JOIN table_name2 ON table_name1.column_name=table_name2.column_name
FULL JOIN	SELECT column_name(s) FROM table_name1 FULL JOIN table_name2 ON table_name1.column_name=table_name2.column_name
LIKE	SELECT column_name(s) FROM table_name WHERE column_name LIKE pattern
ORDER BY	SELECT column_name(s) FROM table_name ORDER BY column_name [ASC DESC]
SELECT	SELECT column_name(s) FROM table_name
SELECT *	SELECT * FROM table_name
SELECT DISTINCT	SELECT DISTINCT column_name(s) FROM table_name
SELECT INTO	SELECT * INTO new_table_name [IN externaldatabase] FROM old_table_name or SELECT column_name(s) INTO new_table_name [IN externaldatabase] FROM old_table_name

SELECT TOP	SELECT TOP number percent column_name(s) FROM table_name
TRUNCATE TABLE	TRUNCATE TABLE table_name
UNION	SELECT column_name(s) FROM table_name1 UNION SELECT column_name(s) FROM table_name2
UNION ALL	SELECT column_name(s) FROM table_name1 UNION ALL SELECT column_name(s) FROM table_name2
UPDATE	UPDATE table_name SET column1=value, column2=value,... WHERE some_column=some_value
WHERE	SELECT column_name(s) FROM table_name WHERE column_name operator value

You need to become familiar with the syntax of SQL if you want to become capable of writing your own code. It is fundamental to your success in this language. As such, you must keep working on strengthening your foundation in the syntax. Getting the syntax incorrect is bound to have an impact on your code, making it completely unusable. You may end up spending hours correcting the syntax. Go through the chapter again if you have any doubts. Keep working on the syntax and make it a second habit.

## Chapter 5: SQL Transactions

Transactions are units that will be used against what the data base intended for it to be used to. These units are meant to accomplish a function in an order that makes sense to the user or to the data base.

Transactions are the propagation of changes that you are making in the data base. So, whenever you do something such as update a record then you are performing a transaction on that table and it is changing it in the data base. You need to make sure that you can control your transactions so that you can make sure that the data that is inside of the table that you are working with still has its integrity and that the data base is going to be able to handle any errors that you might get.

In the end, you are clubbing your SQL searches into a group so that they can be done all at once instead of doing them individually.

Properties of transactions There are four properties that are often known as ACID

**Atomicity:** all the operations in your equations are going to be successful. At any point should your equation fail, the process is going to be terminated and all the previous operations are returned to their former state.

**Consistency:** the data base will be watched to make sure that any changes made are made properly but only when there has been a successful transaction.

**Isolation:** transactions are going to be given the ability to run apart from other transactions and will be transparent from each other.

**Durability:** the result of all your transactions are going to need to survive, even if the system ends up failing.

Control of the transaction You will use four different commands to ensure that a transaction is controlled properly.

**Commit:** changes will be saved.

**Rollback:** all your changes will be reverted back to what there previously were.

**Savepoint:** this is a point where you will save your transaction so if you need to, it can be rolled back.

**Set transaction:** the title of the transaction will be placed.

The control commands can only be used with the insert, delete, and update commands only. You cannot use them to create or drop tables since those commands are already in the data base.

**Commit command** The commit command is going to be the command that you will use when you need to save changes based on a transaction that was made to your data base.

All your transactions will be saved from the last commit or rollback command that was entered into the system.

Code:

Commit;

Example

Think back to the family table we used in our last example. To delete the phone number from the table and use the commit function so that you can ensure your changes are saved in your data base, Your code will be: Delete from family Where phone number = 485-9876; Commit;

Now your table does not have that phone number while the changes are saved in your data base.

**Rollback command** Your transactions will be undone as long as the transaction has not been saved to the data base. They are only going to be able to undo everything back to the last commit command that was done.

Code

Rollback; Since in the last example the commit command was used, you are going to be able to use the rollback command so that you can undo what it was that you did.

Delete from family Where phone number = 485-9876; Rollback; If done properly then you are going to get the phone number back in your table.

Savepoint Once this command has been put into the system, then you are going to be able to roll back to this point. You need to make sure that you have everything the way that you are wanting it to be before you use the savepoint command because if you need to go back to something that is before this savepoint, you are not going to get it back unless you input it in manually.

Code:

Savepoint savepoint\_title; Once a savepoint has been created, the rollback command is only going to be able to undo everything up to this point.

To use a rollback with the savepoint command the code is: Rollback to savepoint\_title; Example

Remember the family table. You are going to create a savepoint that will not only save all of your changes that you have done, but will be the point that you will rollback to when you need to rollback your code.

Your script for constructa save point is going to be longer and appear to be more complex than any other command that you have entered in thus far.

Savepoint SP 1; Save was created.

Delete family phone numbers Phone numbers deleted Savepoint SP 2

Save was created Delete family country Country deleted Savepoint SP 3

Save created Delete family age Age deleted As you can see, you deleted what may be some important information from your family table. You are now going to use the rollback command to go back to the last save point which will be save point 3.

Rollback to point 3

Your rollback was successful!

You now have any information that was deleted after the third rollback was created.

Releasing the save point command When you release your savepoint, you are going to be removing the savepoint of your choosing.

Code

`RELEASE SAVEPOINT SAVEPOINT_TITLE;` After you have released the savepoint you are not going to be able to use the rollback code to go back to the last transaction that was performed because there is nothing that has been saved due to the fact that the system has now undone that save point.

Set transaction command This command is going to be used to indicate a transaction that was made on the data base. This command is going to need to specify the different characteristics from the transaction.

So if you are only wanting to read your transaction, you are going to need to specify that.

Code

`SET TRANSACTION [READ WRITE | READ ONLY ] ;`



## Chapter 6: Sub Queries That Are Done Inside of SQL

A sub query is also known as an inner query or a nested query. In essence, it is a query that is going to be inside of another query that has been embedded inside of a where clause.

Sub queries are going to return data that came from the query that initiated the process due to a condition that is set to restrict data that will be displayed.

You will use update, delete, select, and insert for your sub queries while also using the basic math operators and the in and between operators.

When you are using subqueries you will have some rules that you will need to follow:

1. You need to enclose your subqueries in parentheses
2. There can only be one column inside of the select clause. That is unless you have selected multiple columns when you made your main query.
3. The order by command is not going to be able to be used in a subquery, but it can be used in the main one.
- 4. A subquery has the ability to return more than one row so that it can be used with a variety of operators.**
5. Select list cannot be used for any reference to the values that are being evaluated.
6. You cannot enclose a subquery in a set function
7. The between operator does not have the ability to be used in a subquery.



Subqueries with a select statement Subqueries are going to be used with select statements.

Code

```
SELECT column_title [, column_title ]  
FROM table 1 [, table 2 ]  
WHERE column_title OPERATOR  
(SELECT column_title [, column_title ]  
FROM table 1 [ , table 2 ]  
([ WHERE ])
```

## Example

If you have your employees titles, ages, countries of birth, and their hire date all in a table, you will code to get it to where you are going to only see a set of data that you are trying to locate.

So if you are trying to see everyone that is under the age of 56: Select

From employees

Where age in ( select age

From employees

Where age > 45) ;

Subqueries using an insert statement You can also use a subquery in an insert statement. This statement is going to use the data that you get and place it into a different table. This data that comes from the subquery is going to be able to be modified with the characters, dates, or number functions.

Code

```
INSERT INTO table_title [ (column 1 [, column 2 ] ) }
```

```
SELECT [ * | column 1 [ , column 2 ]
```

```
FROM table 1 [, table 2]
```

```
[WHERE VALUE OPERATOR ]
```

## Example

Going back to the example we just used, you are going to have a table that is going to be similar to the table that you were just working with, all you are going to do is copy the table in order to get a new table.

Subqueries and the update statement You can use your subquery with your update statements. You are either going to be using multiple or single columns that are located inside of the table that needs to be updated through the use of a subquery.

Code



UPDATE table

SET column\_title = new\_value [ WHERE OPERATOR [ VALUE ]

(SELECT COLUMN\_TITLE

FROM TABLE\_TITLE)

[ WHERE ) ]

Subqueries and the delete statement Subqueries are able to be used when you are using a delete statement just like any of the statements that have been mentioned above.

Code

```
DELETE FROM TABLE_TITLE  
[WHERE OPERATOR [ VALUE ]  
(SELECT COLUMN_TITLE  
FROM TABLE_TITLE)  
[WHERE) ]
```

## Chapter 7: Four Tips That Make Using SQL Easier!

- Changing the language on the user interface: close out the program if you have it open and then go to the installation folder. You will right click on the short cut that is on your desk top and open the file location. From there you will open the SQL developer folder and then the first folder that is listed will need to be opened nexted. The next thing that you are going to click on is the SQL developer.conf. You can use any text editor that you are comfortable with using to change the language. You are going to be adding in a new setting inside of the text that is already there to change the language to what it is that you are wanting to see. You can put this new setting anywhere. Putting a comment in the code is going to be a good idea so that you know what you have done if you have to get back into it at a later date. You will AddVMOption before adding in the Duser.lanaguage and you can set it to any language that you are wanting. Now reopen your SQL developer and it will be in the language that you want it in.
- Constructdata base connections: right click on the connection on the left of the screen and click on new connection. You will need to title the connection whatever it is that you want. You will need to enter the usertitle and password for it. You should change the color if you are going to be working with multiple connections at once. In the role you are going to change the role if you are using a system connection title. You can leave the home host alone if you are using your home computer. However, if you are using a different location, you will need to input the IP address for where the system is going to be running. Leave your part alone and xe should be left alone as well unless you are not working with an express edition of SQL. You can test the connection and if

it is successful, you can close the connection down and you have created your connection. If everything is correct it is going to open with no errors and you are going to be able to put in SQL code.

- Disabling features: there are a lot of features that SQL offers and if you do not use them, then you should disable them so that they are not slowing down the developer. You will go to the tools menu and go down to the features option. Each feature has different folders, it is up to you to decide which features you want to keep running and which ones you want to disable. You can expand each folder down so that you are able to see what each folder contains. All you are going to do is uncheck the feature and it will turn that feature off and cause the system to start to run faster. Be sure that you are going to apply the changes so that they are not turning themselves back on without you turning them on yourself.
- Executing commands and scripts: use the tool bar that is at the top of the developer and press the play button. Make sure that you have added in your semi colon. You can also use ctrl and enter so that you are not having to pull your hand off the keyboard. To run a script, you are going to you can use the toolbar again just select run scripts so you run both commands. Or, press the F5 key if that is easier for you. Should your file be external use the at sign and the path file to import it and run it.

## Chapter 8: How to Manage Objects

This guidebook has taken a lot of time to look over many different topics when it comes to SQL, such as which data types you should use to work with your database in SQL. We even spent some time looking at the various commands that you would be able to choose when you are in SQL so you can initiate queries and search around inside the database.

Now that you have some of these basics down, it is time to move on to learn about the steps that you should take when you want to manage the objectivity that comes in the database. There are many things that we would be able to touch upon with regards to this in the guidebook, but some of the ones that we will spend our time on include tables, views, clusters, sequences, and synonyms. Let's start looking at these right now to help you understand how all of them will work in your code.

### What is the schema?

When you are working with what is known as a 'schema' inside of SQL, you should always think of it as using a set of objects that are already found inside of the database, but which will be linked to just one user on the database, rather than being linked to all of the users. The user who has the access will be the one who is the owner of the schema, and they will be the ones you can set the objects. These objects will then be linked directly back to the username that the owner picked. The user will have the power to generate their objects, and then when this is done, they can generate their own schema. This allows the user to have a ton of control over what is found in their databases and they could have the control to change it as much as they want.

You will find that this can be helpful in several ways. Let's say that your users just want to place an order with you. If they have their own account and schema, they would be able to make an order, and then they would be able to change or delete that order if they so choose.

Another example of this is when the user is trying to set up their own account in your store. This is something that they can sign up for and then they will have an account for your store. This is something that they can

choose to do, and then they will go through and pick out the username and the password that they would like to use. You, as the administrator of the site, can approve these.

After the user has been able to set up their own account, they will have access to all parts of the database that pertain to them. They can make changes as well, such as updating their address, changing their payment options, and even making changes to the orders that they placed. In addition, any time that the user would like to be able to get into their account, all they need to do is use the username and password that they picked the first time and log in to mess around on the database.

Let's take a better look at how this will work by bringing out an example. Let's say that you are the person who has the credentials that are needed to log in. For this example, we will use the username 'PERSON1.' You can decide what you would like to place inside this database and you can even create a brand new table, for this one we will call it 'EMPLOYEES\_TBL.' When you then go into the records, you will notice that for this new table, it will be called PERSON1 EMPLOYEES\_TBL. This is how others will see the table name as well so they know who created the table. The schema will be the same for each person who created this table and owns it.

When you or your user would like to access their own schema, one that is prepared already, you will not have to list out the exact name of the schema. Instead, you would simply need to pull up the name that you gave it. So, for the example that we went through before, you would be able to call up EMPLOYEES\_TBL. Remember that this is just with schemas that are in your own account. If you would like to be able to pull up schemas that are present somewhere else, you must add the username ahead of it.

## **How to create a new table**

There are many times when you are creating something new in a database, and you will need to bring out a table. These tables are nice because they can store and present the information that you would want to use. You will find that SQL makes it easy to create tables, and you will then be able to add information as needed. Whenever you want to create a new table, you just have to use the simple command of 'CREATE TABLE.' This command will allow you to bring up the table and start using it, but if you would like

to fill it in and make the table look a certain way, there will be a few more steps that you will need to accomplish.

It is important to think about what you would like to have in the table, how you would like the table to look, how big it should be, and other information about the table to ensure that it is made properly. Almost all of the versions of SQL will provide you with characters that will make it easy to submit or terminate a statement to the server. With ORACLE, the semicolon will be the option that you would use, but with the Transact-SQL version, it is better to work with the GO command. But for most of these versions, you would be able to use the CREATE TABLE command and then when you are ready, you can start filling them out.

### **How to create a table with one that already exists**

There will be times that when you are working with SQL where you will want to take the information that you have from one table and then use that information to create a new table. This is something that you can do with SQL, you just need to learn the right commands to make sure that it works right. The commands that work the best for making this happen include the 'CREATE TABLE' and the 'SELECT' commands. Once you have been able to use these two commands, you will see that it worked to create a new table that will have the same parameters and definitions as your older table. This is a good feature to use when you would like to create a new table that you can customize, but it would have the information that you need from an older table.

There is a little bit of coding to make all of this work for your needs. If you would like to take one of your older tables and use it as the basis of your new table, you would be able to use the following syntax:

```
CREATE TABLE NEW_TABLE_NAME AS  
SELECT [ "[COLUMN1, COLUMN2]  
FROM TABLE_NAME  
[ WHERE];
```

As you take a look at this syntax, you should be able to see that the new syntax will use the keyword SELECT. This is the right keyword to use here

because it is something that you can bring out any time that you would like to work on a query for that particular database. This SELECT keyword will help you to work on your new table, even while you are creating it, with the help of your search results.

## How to drop tables

The next thing that we will work on doing with the SQL system is how to drop tables. If you use a new keyword, the keyword 'RESTRICT' and then you reference a particular table by using the view or the constraint that is set up, the command 'DROP' will be used, but it will give you a message alerting you that there is an error in the system. It is also possible to add in the 'CASCADE' command along with the DROP command. This will make sure that the DROP command will work properly and that all the views and the constraints that are inside of your table will be dropped. To ensure that all of this will work out well for you, you can use the following syntax to drop a new table:

```
DROP TABLE TABLE_NAME [ RESTRICT | CASCADE]
```

Any time that you are interested in dropping your new table inside of the SQL database, you should make sure that you are telling the program who is the owner of this new table that you are working on. This is not always necessary, but it is a good habit to get into. This will ensure that you do not drop the wrong table and it will often help to prevent loss of information inside of the table. If you have access to some of the other accounts inside of your database which is not your own, it is important that you check that you are working inside the account that you want so that you do not change the wrong things and have to fix that mess later on.

Since you will work with a lot of information and databases when you work with SQL, it makes sense that these tables are an important part of working inside the SQL system. These tables will help you to gather information and present it in a way that you are easily able to read. The tables will then be able to take the information, or perhaps the products that you would like to sell, and they can present them to you in a way that is easier for you to look through. Or, you can set up the table so that it is easier for the user to look at it when they are on your website. As you can see, creating these tables is not something that has to be too difficult to work with, but they will



certainly help you to keep the information in your database as safe as possible.

## Chapter 9: MS SQL Server

Microsoft SQL server is so far the leading database provider, helping databases to communicate with computers and other programs. It is a software program mandated to store and recover data as demanded by a range of software applications that run on the same computer or even on a different computer, all connected by a shared network, including the Internet.

Microsoft SQL server has already been named the best database server in the entire market. The first edition was released in 1989, and by 2010, the database server edition had reached 10.5. There are a dozen different Microsoft SQL Server editions that are aimed for different audiences and for different kinds of workloads that range from small single computer applications to large internet based applications that are in use by many connected users.

**How to log into MS SQL server** To begin with for the first time, you will need to have an SQL server account. Once the account is created, access the server page through the start button, to the programs and then to SQL server.

You will need your account details, which include the name of the server, the name with which you will login and your chosen password to gain access to the SQL page.

You can always change your password to ensure that you are the only one accessing your account for obvious reasons.

**Microsoft SQL Server Editions** There are diverse editions available for Microsoft SQL server. Each of them have different features from the other, and they target different users. They are mainly categorized into Mainstream editions and specialized editions as explained below: a) **Mainstream Editions: Datacenter** : this is the full-featured edition of the SQL server. It is designed for data centers that require high levels of scalability and application support. The mainstream editions

**support up to 256 processors as well as virtually unlimited memory. They also come with Stream Insight premium edition.**

**Standard** : this includes the central database engine plus the stand alone services. Active instances receive less support with this edition. It lacks high availability functions too like hot-add memory and parallel indexes.

**Enterprise** : this includes the central database engine as well as the add-on services. It has a wide range of tools as well, useful for creating and managing a SQL server cluster. It supports 8 physical processors and can manage a good number of databases.

**Web** : this is a low TCO-option for web hosting.

**Workgroup** : this includes the central database functionality but leaves out the additional services.

**Business Intelligence** : this includes the standard edition capabilities as well as business intelligence tools for instance Master data services, Power View, Data Quality services, Power Pivot, the BI Semantic Model, among others.

**Express** : this is a lower, free edition of MS SQL Server. It includes the central database engine. It is restricted to use only one processor, 10GB database files, and 1GB memory, but the number of databases and users it can support is unlimited.

b) Specialized editions: **Developer**: this includes the identical features of SQL Enterprise edition. However, it can only be used as a development or a test system and not as a production server. Students can download this program for free.

**Azure** : this is the cloud-based version of the Microsoft SQL server. On Microsoft Azure, this appears as a service offering platform.

**Compact** : this is an embedded database edition. It is based on SQL mobile; it was initially meant to be used with hand held devices. It is small sized and so its features are reduced when compared to all the other editions. It is only limited to a 4 GB maximum database size.

**Embedded** : this can only be accessed by certain windows services. It is specially configured and named an instance of the SQL Express database

engine.

**Fast Track** : this is explicitly designed for business intelligence processing as well as enterprise-scale data warehousing storage. It runs on reference architecture hardware that is specially made for Fast Track.

**Evaluation** : this is the trial edition of the SQL server. It contains the complete features of enterprise edition, but it can only be used for 180 days after which the server services will stop.

**LocalDB** : it's a minimal, on demand version. It is mainly designed for application developers. It is also useful as an embedded database.

**Data warehouse appliance edition** : this is usually pre-installed and configured as a section of an appliance together with HP and Dell based on the Fast Track Architecture. It lacks SQL server integration services, reporting services and analysis services.

**Analytics platform system** : this is a massive parallel processing appliance that is meant for large-scale data warehousing.

Among the reasons why Microsoft SQL server has been a preference of many people is:

- It is affordable when compared to its competitors. Its ownership costs are much less and more affordable when compared to other servers.
- It is user-friendly. Its features are better performing, easy to understand and use in order to get the results that you anticipate every time.
- It has a front-end graphical user interface which is much better than any other server out there.
- It creates more robust database applications than its competitors. This makes it considerably better for your business than any server around.
- MS SQL Server is very easy to install. It is easily installed through a set up wizard. Your installer will

always detect, download and install any required updates to make your server perform better all the time.

- MS SQL has superior security features compared to many other servers. The safety of your database is very crucial, which is why you have to choose a secure server at all times.

**Learning SQL Fast Learning a structured query language is not something that can happen overnight. Learning step by step is better since there are a lot of things you have to learn. Start by mastering the basics and proceed bit by bit to the advanced levels of the use of SQL server.**

After the basics, you can start learning the advanced SQL techniques like user functions, stored procedures, unions, nested queries, and so many others. The most important start point is at the basics level though as with the basic knowledge, you can begin experiencing SQL power. These easy steps will get you started:

- Begin by downloading the required tools. If you are using a Windows operating system, the fastest way is to download the SQL server express. If you are using other platforms, MySQL package will be a better option. It is inclusive of the developer tools. You can start learning SQL from Microsoft access too if you have it on your machine. Oracle offers its Express version for free but for SQL developer, you will download it separately so as to use and manage it.
- Once you have all the tools you need, use a tutorial to get started. Try SQL-Tutorial.net. Ensure that the tutorial that you will be learning from is one that is easy for you to follow. If there is something that you will not understand, be free to present your questions

online so as to ensure that you are mastering the required core competencies well.

- If your tutorial asks you to do some exercises, do them. This is the easiest way to master the skills that you learned from the tutorials. Exercises make it easy for you to understand the concept that has been explained better. They will take a significant portion of your time, but you will benefit a lot from the exercises. You may not master any computer language until you write down some codes, therefore, do as many exercises as you can.
- If you can, read as many books as well. There are books that have been written for starters like you, and they will help you a lot in learning as much as you need to learn about MS SQL server. With the right book, you will get to learn the basics faster and also learn what to expect as you proceed to the advanced levels of learning and use of SQL server.

# Chapter 10: Database Operators

In this chapter you will learn what an operator is and the available database operators in SQL. You will also be introduced on how to properly use various operators in your data querying needs.

An SQL operator is a reserved word or character used mainly within the SELECT command statement's WHERE clause. Its primary function is to perform a given operation, such as a comparison or an arithmetic process. It can also be used to specify a certain SQL statement condition or serve as a conjunction for multiple conditions. In this chapter, we will focus on the following categories of operators: mathematical (or arithmetic), comparison and logical operators.

## Mathematical or Arithmetic Operators

Like in any other computer languages, the main job of a mathematical or arithmetic operator is to perform an SQL mathematical function. Below is a table of the five core mathematical operators in SQLite:

OPERATOR	DESCRIPTION	EXAMPLE
+	Addition (adds two numbers)	a + b
-	Subtraction (subtracts two numbers)	a - b
*	Multiplication (multiplies two numbers)	a * b
/	Division (divides two numbers)	a / b
%	Division with remainder (divides two numbers but returns the remainder)	a % b

These mathematical operators can be combined with one another. Just remember the rules of precedence: multiplication and division operations are performed first and then followed by addition and subtraction. Also take note that a mathematical expression surrounded by parentheses should be evaluated as a block.

## Comparison Operators

The role of a comparison operator is to test single values in an SQL statement and will result to a true (1) or false (0) value, depending on the comparative evaluation. Let us assume that a = 10 and b = 20. The table below summarizes how the comparison operators can be used and the corresponding results.

OPERATOR	DESCRIPTION	EXAMPLE	RESULT
= and ==	Equality (checks if two values are equal)	a = b	0 (false)
!= and <>	Non-Equality (checks if two values are not equal)	a != b a <> b	1 (true)
>	Greater Than (checks if the value on the left is greater than the value on the right)	a > b	0 (false)
<	Less Than (checks if the value on the left is less than the value on the right)	a < b	1 (true)
>=	Greater Than or Equal To (checks if the value on the left is greater than or equal to the value on the right)	a >= b	0 (false)
<=	Less Than or Equal To (checks if the value on the left is less than or equal to the value on the right)	a <= b	1 (true)

The greater than and the less than operators are the two most commonly used comparison operators. They can be combined with the equal sign as shown in the last two rows of the table above.

## Logical Operators

Logical operators use different SQL keywords to perform comparisons instead of symbols. They also allow Boolean expressions to be combined, which means they can perform more conditional operations.



# Chapter 11: Database Security

In this chapter you will learn the basics of database security - its definition, importance and how to manage the users who are allowed to access the system. Security factors should be taken into consideration even at the start of conceptualizing the database model. It becomes a full-time job when administering privileges and security has started, which is often performed by database administrators (DBAs). However, it becomes a complex issue to address especially when excessive security policies create bureaucracy in an organization. That is why you need to have a clear picture of the answers to the following security questions:

- Who is allowed to access the database system?
- Is the database system critical to the organization's operations?
- What are the backup plans available in the event of a database failure?
- Are there security measures implemented in case the database will be available for web applications?
- Should changes to the tables be logged into the system?

## Database Security Definition

Database security is basically defined as the process of protecting data from unauthorized access, which also includes the act of implementing protocols to prevent unauthorized connectivity and manage distribution of user privileges. Bear in mind that many users exist who need to connect and use the database system - creators, administrators, programmers and end users, among others. Generally, the DBA assigns each user a particular user account and password that will give him the access to the database. The given password should be changed immediately by the user since that will be his confidential and personal information. Also, the DBA does not need to know what the user's password is.

## Database Security Importance

Managing security at different levels of user access is essential since databases usually contain sensitive information that should not be available to everyone in an organization. The DBA controls which database operations can be performed and what part of the database can be accessed by each authorized user. Thus, the most important reason for implementing database security is data protection against intentional or unintentional threats. Such data loss or corruption could possibly bring about drastic effects to the daily operations of an organization. Just imagine if a bank would compromise thousands of customers' credit card information, it would surely be devastating for the company's credibility and stability.

Another aspect of database security is to prevent data tampering or modification in a distributed environment, where data continuously flows. In the case of a modification attack, an unauthorized user on a database network will intercept the data being transferred and changes it before it is re-transmitted. An example is changing the amount of an electronic bank transfer from \$100 to \$1,000. How much more risk is at stake, if a certain business organization, transfers thousands, or even millions of dollars?

Also, it becomes more feasible for a database user to falsify an identity in a network environment just to gain access to confidential information. Such culprits can attempt to steal personal information like another person's social security and driver's license numbers, and then setup a fake credit card account in someone else's name. In this case, the DBA should be able to monitor the activities of the different database users and ensure that they will be held responsible for their actions.

### Managing Database Privileges

To clearly define privileges, these are authority levels used to gain database access that further give permission to access the available objects, manipulate data and perform various administrative functions within the database. Like what was explained before, the GRANT command issues such privileges while the REVOKE command takes them away.

When a user is able to connect to a database, it does not mean that he can automatically access the data within the database. It is through these two types of database privileges that a user will be able to determine if he is allowed to perform data manipulation:

## System Privileges

These are the type of privileges that permit database users to perform administrative functions within the database, which could result in drastic repercussions if not monitored carefully. System privileges also vary greatly among the different database software vendors, so you have to check first that they will be correctly implemented. The database administrative functions include the following:

- Create and drop databases
- Create and drop user accounts
- Drop and alter the state of database objects

## Object Privileges

These are the type of privileges that give users permission to perform certain operations on database objects. The owner of the existing database objects are the ones who could grant object privileges to the other system users. If you are an owner of an object, you are automatically granted all the privileges that are associated to that object you have created. The operations that can be performed on database objects include:

- Use a specific domain
- Access a specific table
- Insert data into a column or all the columns of a specific table
- Update a column or all the columns of a specific table
- Reference a column or all the columns of a specific table

In the next chapter you will learn how to enhance your query processes using all the available SQLite command statements

## Chapter 12: Real-World Uses

We have seen how we can use SQL in isolation. For instance, we went through different ways to create tables and what operations you can perform on those tables to retrieve the required answers. If you only wish to learn how SQL works, you can use this type of learning, but this is not how SQL is used.

The syntax of SQL is close to English, but it is not an easy language to master. Most computer users are not familiar with SQL, and you can assume that there will always be individuals who do not understand how to work with SQL. If a question about a database comes up, a user will almost never use a SELECT statement to answer that question. Application developers and systems analysts are probably the only people who are comfortable with using SQL. They do not make a career out of typing queries into a database to retrieve information. They instead develop applications that write queries.

If you intend to reuse the same operation, you should ensure that you never have to rebuild that operation from scratch. Instead, write an application to do the job for you. If you use SQL in the application, it will work differently.

### SQL in an Application

You may believe that SQL is an incomplete programming language. If you want to use SQL in an application, you must combine SQL with another procedural language like FORTRAN, Pascal, C, Visual Basic, C++, COBOL, or Java. SQL has some strengths and weaknesses because of how the language is structured. A procedural language that is structured differently will have different strengths and weaknesses. When you combine the two languages, you can overcome the weaknesses of both SQL and the procedural language.

You can build a powerful application when you combine SQL and a procedural language. This application will have a wide range of capabilities. We use an asterisk to indicate that we want to include all the columns in the table. If this table has many columns, you can save a lot of

time by typing an asterisk. Do not use an asterisk when you are writing a program in a procedural language. Once you have written the application, you may want to add or delete a column from the table when it is no longer necessary. When you do this, you change the meaning of the asterisk. If you use the asterisk in the application, it may retrieve columns which it thinks it is getting.

This change will not affect the existing program until you need to recompile it to make some change or fix a bug. The effect of the asterisk wildcard will then expand to current columns. The application could stop working if it cannot identify the bug during the debugging process. Therefore, when you build an application, refer to the column names explicitly in the application and avoid using the asterisk.

Since the replacement of paper files stored in a physical file cabinet, relational databases have given way to new ground. Relational database management systems, or RDBMS for short, are used anywhere information is stored or retrieved, like a login account for a website or articles on a blog. Speaking of which, this also gave a new platform to and helped leverage websites like Wikipedia, Facebook, Amazon, and eBay. Wikipedia, for instance, contains articles, links, and images, all of which are stored in a database behind-the-scene. Facebook holds much of the same type of information, and Amazon holds product information and payment methods, and even handles payment transactions.

With that in mind, banks also use databases for payment transactions and to manage the funds within someone's bank account. Other industries, like retail, use databases to store product information, inventory, sales transactions, price, and so much more. Medical offices use databases to store patient information, prescription medication, appointments, and other information.

To expand further, using the medical office for instance, a database gives permission for numerous users to connect to it at once and interact with its information. Since it uses a network to manage connections, virtually anyone with access to the database can access it from just about anywhere in the world.

The digital database helps leverage mobile applications and provides new opportunities for software, or any other platforms that use databases on a daily basis, to be developed. One app that comes to mind would be an email app, as it's storing the emails on a server somewhere in a data center and allowing you to view and send emails.

These types of databases have also given way to new jobs and have even expanded the tasks and responsibilities of current jobs. Those who are in finance, for instance, now have the ability to run reports on financial data; those in sales can run reports for sales forecasts, and so much more!

In practical situations, databases are often used by multiple users at the same time. A database that can support many users at once has a high level of concurrency. In some situations, concurrency can lead to loss of data or the reading of non-existent data. SQL manages these situations by using transactions to control atomicity, consistency, isolation, and durability. These elements comprise the properties of transactions. A transaction is a sequence of T-SQL statements that combine logically and complete an operation that would otherwise introduce inconsistency to a database. Atomicity is a property that acts as a container for transaction statements. If the statement is successful, then the total transaction completes. If any part of a transaction is unable to process fully, then the entire operation fails, and all partial changes roll back to a prior state. Transactions take place once a row, or a page-wide lock is in place. Locking prevents modification of data from other users taking effect on the locked object. It is akin to reserving a spot within the database to make changes. If another user attempts to change data under lock, their process will fail, and an alert communicates that the object in question is barred and unavailable for modification. Transforming data using transactions allows a database to move from one consistent state to a new consistent state. It's critical to understand that transactions can modify more than one database at a time. Changing data in a primary key or foreign key field without simultaneously updating the other location, creates inconsistent data that SQL does not accept. Transactions are a big part of changing related data from multiple table sources all at once. Transactional transformation reinforces isolation, a property that prevents concurrent transactions from interfering with each other. If two simultaneous transactions take place at the same time, only one of them will be successful. Transactions are invisible until they are

complete. Whichever transaction completes first will be accepted. The new information displays upon completion of the failed transaction, and at that point, the user must decide if the updated information still requires modification. If there happened to be a power outage and the stability of the system fails, data durability would ensure that the effects of incomplete transactions rollback. If one transaction completes and another concurrent transaction fails to finish, the completed transaction is retained. Rollbacks are accomplished by the database engine using the transaction log to identify the previous state of data and match the data to an earlier point in time.

There are a few variations of a database lock, and various properties of locks as well. Lock properties include mode, granularity, and duration. The easiest to define is duration, which specifies a time interval where the lock is applied. Lock modes define different types of locking, and these modes are determined based on the type of resource being locked. A shared lock allows the data reads while the row or page lock is in effect. Exclusive locks are for performing data manipulation (DML), and they provide exclusive use of a row or page for the execution of data modification. Exclusive locks do not take place concurrently, as data is being actively modified; the page is then inaccessible to all other users regardless of permissions. Update locks are placed on a single object and allow for the data reads while the update lock is in place. They also allow the database engine to determine if an exclusive lock is necessary once a transaction that modifies an object is committed. This is only true if no other locks are active on the object in question at the time of the update lock. The update lock is the best of both worlds, allowing reading of data and DML transactions to take place at the same time until the actual update is committed to the row or table. These lock types describe page-level locking, but there are other types beyond the scope of this text. The final property of a lock, the granularity, specifies to what degree a resource is unavailable. Rows are the smallest object available for locking, leaving the rest of the database available for manipulations. Pages, indexes, tables, extents, or the entire database are candidates for locking. An extent is a physical allocation of data, and the database engine will employ this lock if a table or index grows and more disk space is needed. Problems can arise from locks, such as lock escalation or deadlock, and we highly encourage readers to pursue a deeper understanding of how these function.

It is useful to mention that Oracle developed an extension for SQL that allows for procedural instruction using SQL syntax. This is called PL/SQL, and as we discussed at the beginning of the book, SQL on its own is unable to provide procedural instruction because it is a non-procedural language. The extension changes this and expands the capabilities of SQL. PL/SQL code is used to create and modify advanced SQL concepts such as functions, stored procedures, and triggers. Triggers allow SQL to perform specific operations when conditional instructions are defined. They are an advanced functionality of SQL, and often work in conjunction with logging or alerts to notify principals or administrators when errors occur. SQL lacks control structures, the for looping, branching, and decision making, which are available in programming languages such as Java. The Oracle corporation developed PL/SQL to meet the needs of their database product, which includes similar functionality to other database management systems, but is not limited to non-procedural operations. Previously, user-defined functions were mentioned but not defined. T-SQL does not adequately cover the creation of user-defined functions, but using programming, it is possible to create functions that fit neatly within the same scope as system-defined functions. A user-defined function (UDF) is a programming construct that accepts parameters, performs tasks capable of making use of system defined parameters, and returns results successfully. UDFs are tricky because Microsoft SQL allows for stored procedures that often can accomplish the same task as a user-defined function. Stored procedures are a batch of SQL statements that are executed in multiple ways and contain centralized data access logic. Both of these features are important when working with SQL in production environments.



## Chapter 13: How SQL Injections Can Destroy Databases

Your databases can be destroyed easily by malicious persons with SQL injections. These thoughtless hackers can insert a code into your SQL statements through the web and damage your databases.

Hence, if you have plans of creating your own website, you must be aware of this so that you can avoid or prevent it from happening. The most common method is to hack into passwords and user names or IDs.

Hackers usually insert a number or a symbol that can change the returns of your SQL query.

**Here's how you can minimize this incident from happening.**

1. Create a BLACKLIST of words that are not allowed in the SQL statements submitted. If you are a website administrator, you will be screening out and minimizing the hacking of your website.

This is not a full proof protection though, because most of the characters and symbols in SQL statements or queries are commonly used.

2. Utilize SQL PARAMETERS after your SQL query has been submitted. The symbol of PARAMETERS is @ .

Example #1: `SELECT * FROM Visitors WHERE VisitorId = @0`;

Example #2: `INSERT INTO Students (StudentName, Age, City) Values (@1, @2, @3)`;

In the examples above, a malicious code (number) could not be inserted because the parameters have limited any addition of a new number.

3. Double check your SQL queries or syntax, so that this cannot be easily tampered with.
4. If you are a database user, submit your queries to websites that are proven legitimate.
5. If you are unsure of your SQL syntax, don't input it or be nonchalant about it. When uncertain, don't use it.

Although, not all SQL users will be concerned with hacking, it pays to know these basic prevention skills and be on the lookout for these malicious codes.

## Chapter 14: Additional Pointers in Using SQL

There are significant pointers that you must know as a SQL beginner. Take note of them and enrich your SQL learning process.

1. **Be aware that there may be other versions of the SQL queries** . SQL has been extraordinarily helpful in modifying tables that many developers had created SQL syntax for their own applications. So, be open to new SQL statements.
2. **Ensure that your computer is durable, if you plan to establish your own database** . A computer that is not protected from viruses may have all of its databases corrupted.
3. **Be patient in learning.** There are no shortcuts to success; you have to go through the ups and downs.
4. **Persistence is the key** . No matter how hard the task is, if you are persistent and you persevere, you will never fail. As the cliché goes: A quitter never wins, and a winner NEVER quits.”
5. **Be positive.** An optimistic attitude will help you learn more quickly. This is because you are motivated in searching for the positive things that await you.
6. **Learn one chapter at a time.** No one is keeping tab of the time, so take your time. Munch the information slowly until your mind has digested the information sufficiently.
7. **Learn the limitations of your DBMS (database management system).** It is only in knowing this that you can modify your tables effectively.

8. **Avoid taxing your SQL server by creating unnecessary queries.** If you don't need all the columns, refrain from using \* and select only the columns that you need. Submitting useless queries will only slow down your system.
9. **There are certain SQL statements that can be easier to use in fetching tables.** Remember the queries and apply whichever is more preferable for you.
10. **Ascertain that your 'table\_names' are unique and have a fully qualified name .** You would not want retrieving two tables when all you needed is one. It is time consuming. Time is gold.
11. **Practice makes perfect;** therefore, do not be afraid to practice on some SQL databases or tables.
12. **Invite a friend to learn SQL with you.** Many beginners prefer to learn with someone. This is due to the fact that they can challenge and motivate each other. But, of course, if you prefer learning alone, then that would not be a problem.
13. **Keep a logbook of your learning activities .** Through this method, you can monitor your progress and assess your knowledge.
14. **Apply what you have learned.** Learning can only be verified when you are able to apply it, so don't be afraid to create and tweak your own databases.
15. **Share your knowledge with others .** As you share it, you are also mastering and retaining the information more effectively.

These are simple tips that can help you optimize your SQL learning process. Apply them and be a master SQL expert eventually.

## Chapter 15: SQL Quiz

Okay, so are you ready now to test how much you have learned about basic SQL?

Here are 10 questions to test your basic knowledge about SQL syntax or queries.

All SQL statements should be based on this table:

EmployeesSalary

Names	Age	Salary	City
Williams Michael	22	30000.00	<b>Casper</b>
Colton Jean	24	37000.00	<b>San Diego</b>
Anderson Ted	30	45000.00	<b>Laramie</b>
Dixon Allan	27	43000.00	<b>Chicago</b>
Clarkson Tim	25	35000.00	<b>New York</b>
Alaina Ann	32	41000.00	<b>Ottawa</b>
Rogers David	29	50000.00	<b>San Francisco</b>
Lambert Jancy	38	47000.00	<b>Los Angeles</b>
Kennedy Tom	27	34000.00	<b>Denver</b>
Schultz Diana	40	46000.00	<b>New York</b>

**Answer the questions first before looking at the correct answers. Here goes:**

1. Transcribe SQL.
2. What is the keyword in creating tables?
3. What is the SQL syntax in selecting tables?
4. What is the keyword in deleting tables?

5. What is the SQL statement if you want to display only the names and the city of the table above?
6. What is the SQL statement if you want to retrieve only the data of employees who are 25 years old and above?
7. What is the SQL command if you want to arrange the names in an ascending order?
8. What is the SQL query if you want to fetch the data of employees, who have a salary of more than 20000.00?
9. What is the SQL command if you want to select only the employees coming from Denver?
10. What is the SQL if you want to change the Name of Lambert Jancy to Walker Jean?

Easy? Oh yes! You should know the answers, as a beginner.

Try to answer all of them first, before checking on the correct answers.

**Now, let's check if you have the correct answers.**

### **ANSWERS:**

1. STRUCTURED QUERY LANGUAGE
2. CREATE TABLE
3. SELECT“column\_name1”,“column\_name2  
FROM“table\_name”;

(Remember to remove the double quotes when substituting the names of your columns and tables.)

4. DELETE TABLE

5. SELECT Names, City FROM EmployeesSalary;
6. SELECT \* FROM EmployeesSalary  
WHERE Age >= 25;
7. SELECT \* FROM EmployeesSalary  
ORDER BY Names ASC;
8. SELECT \* FROM EmployeesSalary  
WHERE Salary >20000;
9. SELECT \* FROM EmployeesSalary  
WHERE City ='Denver';
10. UPDATE EmployeesSalary  
SET Names ='Walker Jean'  
WHERE Names ='Lambert Jancy';

So, how many correct answers did you get?

They are easy questions. I hope you got a perfect score. If not, then no sweat! Just review the chapters again and repeat the quiz.

Take note that the keywords such as, UPDATE, goes hand in hand with SET and WHERE.

UPDATE, SET WHERE. This is only one item, so create your own 'MEMORY NOTES'.

## Conclusion

Learning the SQL language can be laborious and tedious, but if you have genuine interest in learning a new language and updating your skills, it could be relatively easy.

In this book, all the basic information that you need to learn as a beginner are presented. All you have to do is to apply them.

Now that you have read the book, you can go back for the details that you may forget along the way.

If you have properly studied the contents of this book, you could construct your basic SQL syntax easily.