

# SQL PROGRAMMING FOR BEGINNERS

THE ULTIMATE BEGINNERS GUIDE TO ANALYZE  
AND MANIPULATE DATA WITH SQL



EUGENE GATES

# **SQL PROGRAMMING FOR BEGINNERS:**

**THE ULTIMATE BEGINNERS GUIDE  
TO *ANALYZE AND MANIPULATE*  
*DATA WITH SQL***

**Eugene Gates**

# **Table of Contents**

[Introduction](#)

[Chapter 1:SQL Basics](#)

[Chapter 2:Relational Database](#)

[Chapter 3:Data Types](#)

[Chapter 4:Exploring a Database](#)

[Chapter 5:Creating a Database](#)

[Chapter 6:Getting Started with Queries](#)

[Chapter 7:Turning Data into Information](#)

[Chapter 8:Working with Multiple Tables](#)

[Chapter 9:Using Functions](#)

[Chapter 10:\\_\\_\\_\\_\\_Subqueries](#)

[Chapter 11:\\_\\_\\_\\_\\_SQL Views and Transactions](#)

[Conclusion](#)

# **Introduction**

Network programs have become larger and more flexible.

In many cases, the fundamental scheme of operations is mainly a mix of scripts that handle the command of a database.

When you look at the structure of any business, you will see that it generates, holds, and then uses data.

Because of the different ways that the company will need to handle this data, they will need to find some method of storing the information.

In the traditional methods, known as Database Management System or DBMS, business organizations would have all the data in one place to help them out.

These systems are pretty simple to use, but modern technology has forced about some changes.

Even the most essential or basic data management systems have changed, and now they are more powerful than before.

This can be an advantage to some companies that have a large amount of data to keep track of or who may need to be careful with some sensitive information.

Out of all this, there was a new breed of data management that has been implemented known as the Relational Database Management System or RDBMS.

This was derived from the renown traditional DBMS, but it is going to have some more to do with the web as well as server and client technologies.

This basically means that they are going to help various companies with the management of data.

One of these new relational databases that will help to store the data in an easy and simple to use a method that also keeps it all safe is SQL.

## **What is SQL?**

It is best to start at the beginning. SQL stands for 'Structured Query Language,' and it is a simple language to learn, considering it will allow interaction to occur between various databases found in a particular system.

The original version was established in the 1970s.

This continued to progress in 1979 until IBM released a new prototype, the Relational Software Inc. that published one of the first SQL tools in the world.

This tool was at first called ORACLE, and it gained so much success that the company was able to split off from IBM and create the Oracle Corporation.

Even today, ORACLE is one of the leaders thanks to being able to use the SQL language.

The SQL is a set of instructions that you can use to interact with your relational database.

While there are a lot of languages that you can use to do this, SQL is the only language that most databases can understand.

Whenever you are ready to interact with one of these databases, the software can go in and translate the commands that you are given, whether you are giving them in form entries or mouse clicks.

These will be translated into SQL statements that the database will already be able to interpret.

If you have ever worked with a software program that is database-driven, then it is likely that you have used some form of SQL in the past.

It is likely that you didn't even know that you were doing this, though.

For example, there are a lot of dynamic web pages that are database driven.

These will take some user input from the forms and clicks that you are making and then will use this information to compose a SQL query.

This query will then go through and retrieve the information from the database to perform the action, such as switch over to a new page.

To illustrate how this functions, think about a simple online catalog that allows you to search.

The search page will often contain a form that will just have a text box.

You can enter the name of the item that you would like to search using the form, and then you would simply need to click on the search button.

As soon as you click on the search button, the web server will go through and search through the database to find anything related to that search term.

It will bring those back to create a new web page that will go along with your

specific request.

For those who have not spent that much time at all learning a programming language and who would not consider themselves programmers, the commands that you would use in SQL are not too hard to learn.

Take your time in reading this book for this will be very helpful in guiding you as a beginner to SQL Programming.

# Chapter 1: SQL Basics

The SQL (the Structured Query Language, Structured Query Language) is a special language used to define data, provide access to data and their processing. The SQL language refers to nonprocedural languages - it only describes the necessary components (for example, tables) and the desired results, without specifying how these results should be obtained. Each SQL implementation is an add-on on the database engine, which interprets SQL statements and determines the order of accessing the database structures for the correct and effective formation of the desired result.

## SQL to Work with Databases?

To process the request, the database server translates SQL commands into internal procedures. Due to the fact that SQL hides the details of data processing, it is easy to use.

**You can use SQL to help out in the following ways:**

- SQL helps when you want to create tables based on the data you have.
- SQL can store the data that you collect.
- SQL can look at your database and retrieves the information on there.
- SQL allows you to modify data.
- SQL can take some of the structures in your database and change them up.
- SQL allows you to combine data.
- SQL allows you to perform calculations.
- SQL allows data protection.

Traditionally, many companies would choose to work with the 'Database Management System,' or the DBMS to help them to keep organized and to keep track of their customers and their products. This was the first option that was on the market for this kind of organization, and it does work well. But over the years there have been some newer methods that have changed the way that companies can sort and hold their information. Even when it comes

to the most basic management system for data that you can choose, you will see that there is a ton more power and security than you would have found in the past.

Big companies will be responsible for holding onto a lot of data, and some of this data will include personal information about their customers like address, names, and credit card information. Because of the more complex sort of information that these businesses need to store, a new 'Relational Database Management System' has been created to help keep this information safe in a way that the DBMS has not been able to.

Now, as a business owner, there are some different options that you can pick from when you want to get a good database management system. Most business owners like to go with SQL because it is one of the best options out there. The SQL language is easy to use, was designed to work well with businesses, and it will give you all the tools that you need to make sure that your information is safe. Let's take some more time to look at this SQL and learn how to make it work for your business.

## **How This Works with Your Database**

If you decide that SQL is the language that you will work on for managing your database, you can take a look at the database. You will notice that when you look at this, you are basically just looking at groups of information. Some people will consider these to be organizational mechanisms that will be used to store information that you, as the user, can look at later on, and it can do this as effectively as possible. There are a ton of things that SQL can help you with when it comes to managing your database, and you will see some great results.

There are times when you are working on a project with your company, and you may be working with some kind of database that is very similar to SQL, and you may not even realize that you are doing this. For example, one database that you commonly use is the phone book. This will contain a ton of information about people in your area including their name, what business they are in, their address, and their phone numbers. And all this information is found in one place so you won't have to search all over to find it.

This is kind of how the SQL database works as well. It will do this by looking through the information that you have available through your company database. It will sort through that information so that you are better



able to find what you need the most without making a mess or wasting time.

## **Relational Databases**

First, we need to take a look at the relational databases. This database is the one that you will want to use when you want to work with databases that are aggregated into logical units or other types of tables, and then these tables have the ability to be interconnected inside of your database in a way that will make sense depending on what you are looking for at the time. These databases can also be good to use if you want to take in some complex information, and then get the program to break it down into some smaller pieces so that you can manage it a little bit better.

The relational databases are good ones to work with because they allow you to grab on to all the information that you have stored for your business, and then manipulate it in a way that makes it easier to use. You can take that complex information and then break it up into a way that you and others are more likely to understand. While you might be confused by all the information and how to break it all up, the system would be able to go through this and sort it the way that you need in no time. You are also able to get some more security so that if you place personal information about the customer into that database, you can keep it away from others, in other words, it will be kept completely safe from people who would want to steal it.

## **Client and Server Technology**

In the past, if you were working with a computer for your business, you were most likely using a mainframe computer. What this means is that the machines were able to hold onto a large system, and this system would be good at storing all the information that you need and for processing options.

Now, these systems were able to work, and they got the job done for a very long time. If your company uses these and this is what you are most comfortable with using, it does get the work done. But there are some options on the market that will do a better job. These options can be found in the client-server system.

These systems will use some different processes to help you to get the results that are needed. With this one, the main computer that you are using, which would be called the 'server,' will be accessible to any user who is on the

network. Now, these users must have the right credentials to do this, which helps to keep the system safe and secure. But if the user has the right information and is on your network, they can reach the information without a lot of trouble and barely any effort. The user can get the server from other servers or from their desktop computer, and the user will then be known as the 'client' so that the client and server are easily able to interact through this database.

## **How to Work With Databases That Are Online**

There are a lot of business owners who will find that the client and server technology is the one that works for them. This system is great for many companies, but there are some things that you will need to add or take away at times because of how technology has been changing lately. There are some companies that like the idea that their database will do better with the internet so that they can work on this database anywhere they are located, whether they are at home or at the office. There are even times when a customer will have an account with the company, and they will need to be able to access the database online as well. For example, if you have an account with Amazon, you are a part of their database, and you can gain access to certain parts through this.

As the trend continues for companies to move online, it is more common to see that databases are moving online as well and that you must have a website and a good web browser so that the customer can come in and check them out. You can always add in usernames and passwords to make it more secure and to ensure that only the right user can gain access to their information. This is a great idea to help protect personal and payment information of your customers. Most companies will require that their users pick out security credentials to get on the account, but they will offer the account for free.

Of course, this is a system that is pretty easy to work with, but there will be a number of things going on behind the scenes to make sure that the program will work properly. The customer can simply go onto the system and check the information with ease, but there will be a lot of work for the server to do to make sure that the information is showing up on the screen in the right way, and to ensure that the user will have a good experience and actually see their own account information on the screen.

For example, you may be able to see that the web browser that you are using uses SQL or a program that is similar to it, to figure out the user that your data is hoping to see.

## **Why is SQL So Great?**

Now that we have spent some time talking about the various types of database management systems that you can work with; it is time to discuss why you would want to choose SQL over some of the other options that are out there. You not only have the option of working with other databases but also with other coding languages, and there are benefits to choosing each one. So, why would you want to work with SQL in particular? Some of the great benefits that you can get from using SQL as your database management system includes:

### **Incredibly Fast**

If you would like to pick out a management system that can sort through the information quickly and will get the results back in no time, then SQL is one of the best programs to use for this. Just give it a try, and you will be surprised at how much information you can get back, and how quickly it will come back to you. In fact, out of all the options, this is the most efficient one that you can go with.

### **Well-Defined Standards**

The database that comes with SQL is one that has been working well for a long time. In addition, it has been able to develop some good standards that ensure the database is strong and works the way that you want. Some of the other databases that you may want to work with will miss out on these standards, and this can be frustrating when you use them.

## Chapter 2: Relational Database

A relational database management system is a management system based on the relational database model. Its most common representation is as a table that contains rows and columns.

First introduced by E.F Codd, A RDBMS has several key components.

They include:

- Table
- Record/Tuple
- Field/ Column
- Instance
- Schema
- Keys

We will cover some of these components as we progress further into the SQL lessons covered in the book. For now, let us look at tables and columns.

### SQL Tables

Tables are objects used to store data in a Relational Database Management System. Tables contain a group of related data entries with numeric columns and rows.

Tables are the simplest form of data storage for relational databases. They are also a convenient representation of relations. However, a table can contain duplicate rows while a relation cannot contain any duplicate. The following is an example of a DVD rental database table:

Data Output

Explain

Messages

Notifications

	customer_id [PK] integer	store_id smallint	first_name character varying (45)	last_name character varying (45)
1	524	1	Jared	Ely
2	1	1	Mary	Smith
3	2	1	Patricia	Johnson
4	3	1	Linda	Williams
5	4	2	Barbara	Jones
6	5	1	Elizabeth	Brown
7	6	2	Jennifer	Davis
8	7	1	Maria	Miller

The table shows information about customers in a DVD rental database.

## SQL Fields

Tables break down further into smaller entities referred to as fields. Fields are columns in tables designed to hold information about records in the specified table. For example, in the table above, we can see fields that contain Customer\_id, store\_id, first\_name and last\_name.

## SQL Records

In a database, we use Records or Rows to refer to a single entry in a database table. It represents a collection or related data. We can also regard it as the horizontal entity in a database table. For example, in the above table, we have five records or tuples.

1	524	1	Jared	Ely
2	1	1	Mary	Smith
3	2	1	Patricia	Johnson
4	3	1	Linda	Williams

**SQL**

## Columns

In a database table, we use a column to refer to the vertical entity containing data related to specific fields in a table. For example, in the above table, a column could be first\_name or last\_name.

<b>first_name</b> character varying (45)
Jared
Mary
Patricia
Linda
Barbara
Elizabeth
Jennifer
Maria

## NULL Values

A null value is a value assigned to a blank field. This means that a field with a value NULL has no value. In SQL, a zero or a space is not a NULL value. NULL value mainly refers to a field that is blank in a record.

## SQL Constraints

Constraints are rules applied to columns in a database table. We mainly use them to govern the type of data stored within the table. This ensures that the database is accurate and reliable thus minimizing errors.

We can set constraints on columnar or tabular level. Columnar level rules apply only to the specified column while tabular level rules apply to the entire database level.

SQL contains various rules applied to the stored data.

They include:

- **NOT NULL:** This rule ensures that a table or column does not contain null value, which means during record creation, you must enter either a zero, Space, or another value
- **UNIQUE:** Ensures each value is unique and no duplicates are available
- **DEFAULT:** We use this constraint rule to set a default value for a column where we have an unspecified value

- **PRIMARY KEY:** Identifies each record distinctively in a table. We create this rule by combining NOT NULL and UNIQUE constraints
- **FOREIGN KEY:** We use this constraint to identify (distinctively) a record in another database table
- **CHECK constraint:** We use this constraint to confirm if the values in a column fulfill defined conditions
- **INDEX constraint:** We use this to create and retrieve data from a database

## **Data Integrity**

Data integrity refers to the consistency and accuracy of the data. When creating databases, you **MUST** pay attention to data integrity. A good database must ensure reinforcement of data integrity as much as possible. Data integrity must also remain maintained during manipulation and update of the database.

Various factors can lead to compromised data integrity within a database. These factors include:

- Connection failure during transfer of data between databases
- Data input that is outside the range.
- Deletion of wrong database records
- Hacking and malicious attacks
- Database backup failures
- Updating of primary key value with the presence of foreign key in a related table
- Using test data in the database

These primary factors are often the ones that lead to loss of data integrity. To avoid compromised data integrity, it is good practice to back up the database before any operation.

Let us look at the types of data integrity available: These data integrity types exist to each RDBMS:

- Entity Data Integrity
- Referential Data Integrity
- Domain Data Integrity
- User-Defined Integrity

## **Entity Data Integrity**

Entity data integrity ensures that rows in a database table are unique and thus, no row can be the same within the same database table. The best way to achieve this is by primary key definition. The field in which the primary key is stored contains a unique identifier thus no row can contain the same identifier.

## **Referential Data Integrity**

Referential Data integrity means consistency and accuracy of data between a relationship. A database relationship refers to the data link within 2 or more tables. In a relationship, we use a certain foreign key to reference another primary key of a certain table. Due to this referencing, we always need to observe data integrity between database relationships.

Hence, Referential Data Integrity requires that the use of a foreign key must reference to an existing and valid primary key.

### **Referential integrity prevents:**

- Addition of records to a related table if no record is available in the parent table
- Deletion of records in a parent table if records are matching a related table
- Changing values in a primary table resulting in orphaned records in the child table

## **Domain Data Integrity**

Domain data integrity is the consistency and accuracy of data within a column. We achieve this by selecting the suitable data type for a column. Other steps to preserve this type of data integrity could include setting suitable constraints and defining the data forms and range restrictions.



## **User-Defined Data Integrity**

This type of data integrity is custom defined by the database administrator. It allows the administrator to define rules that are not available using any of the other type of data integrity.

## **Normalization**

Normalization refers to the process of organizing databases to improve data integrity and reduce data redundancy. This process aids the simplification of the database design. Database normalization offers a range of advantages including:

- Removes the null values available in the databases
- It helps in query simplification
- Helps in speed up operations such as searching and sorting the database indexes
- Helps to clean and optimize database structure
- Removes data redundancy
- Helps in achieving compact databases

## **Database Normalization Levels**

Normalization occurs in various levels where each level builds upon the previous levels. Databases must satisfy all the rules of the lower levels to attain a specific level.

Let us discuss the types of database normalization levels. These levels appear arranged in the order of their strength from the strongest to the weakest.

### **1: Domain-Key Normal Form**

In this type of normalization level, the relation is to Domain-Key Normal Form (DKNF) when every constraint in the relation follows a logical sequence of the definition keys and domains. This removes the probability of non-temporal anomalies as the domain restraint and enforcing keys roots all the constraints to be met.

### **2: Sixth Normal Form**

We say a database relation is in the sixth form normalization when every join dependency of the relation is said to be trivial. We classify a join dependency

as trivial if only one of the components is equivalent to the related heading in its total.

### **3: Fifth Normal Form**

We say a database relation is in fifth normalization level if non-trivial join dependency in the stated table is indirect by candidate keys.

### **4: Essential Tuple Normal Form**

A database relational schema is in ENTF normalization level if it is in Boyce-Codd form and components of every openly declared join dependency are a super key.

### **5: Fourth Normal Form**

We say that a database table or relation is in Fourth Normal form if all of its non-trivial multivalued dependencies are super-keys.

NOTE: We have many other database normalization levels not covered in this book. They include:

- Unnormalized Form
- First Normal Form
- Second Normal Form
- Third Normal Form
- Elementary-Key Normal Form
- Boyce-Codd Normal Form

## **DBMS VS RDBMS**

We have already covered RDBMS. DBMS is not different either. DBMS is a software package we use to create and manipulate databases.

DBMS and RDBMS give programmers and users an organized way of working with databases. DBMS and RDBMS are not very different since both provide a physical database storage.

With the above noted, some relevant difference between them are worth mentioning.

They include:

	RDBMS	DBMS
--	-------	------

	Supports Normalization	Normalization is not available in a DBMS
	Supports Distributed Database System	Does not support Distributed Database System
	Stores data in a table format	Stores data is a normal computer form
	Integrity constraints are defined for the purpose of ACID property	Security for data manipulation is not applied by DBMS
	Can handle large amounts of data such as a Company database	Designed to small scale data and personal use
	It supports multiple users	Supports a single user
	Cases of data redundancy are close to none	Data redundancy is a very common scenario
	Stores data in tabular form and utilizes primary keys	Mainly stores data in hierarchical or navigational format
	Requires complex software and high-performance hardware to implement	Low software and hardware requirements for implementation
	Supports client-server architecture	Client-side architectures is not supported
	They include PostgreSQL, MySQL and Oracle etc.	Include file system files, xml files, and windows registry.

At this point in the guide, you know enough to start working with SQL and databases.

## Chapter 3: Data Types

### The Basic Types of Data

Data types are attributes of the information itself, whose characteristics are placed inside a table. For instance, you may require that a field should hold numeric values only, stopping any user from entering alphanumeric data. By assigning data types to certain fields in a database, you can minimize the possibility of errors in data entry.

❖ Important Note: Each version of SQL has its own array of data types. Nowadays, you should use version-specific data types if you want to manage your database properly. The basics, however, are the same for all SQL versions.

Here are the basic types of data:

- Numeric strings
- Character strings
- Time and date values

### The Fixed-Length Characters

Constant characters, or strings that have constant length, are saved via fixed length data types. Here is the typical data type for a fixed-length character in SQL:

#### **CHARACTER(n)**

“n” is the assigned (or maximum) length of the field you are working on.

Some SQL implementations utilize the “CHAR” data type to save fixed-length information. You can use this data type to store alphanumeric information. State abbreviations serve as excellent example for this: each state abbreviation is composed of two characters.

When using fixed data types, database users often add spaces to fill excess fields. That means if the assigned length was 15 and the data you entered filled only 11 places, you should fill the remaining four places with spaces. This method helps you to make sure that each value is fixed-length.

**Important Note:**

❖ If you're working on fields that may hold values of different lengths (e.g. usernames), make sure that you are not using fixed-length data types. If you used this type incorrectly, you may encounter problems related to data accuracy and storage space.

## **The Variable Characters**

SQL also allows you to use varying-length strings (i.e. strings whose length may change from one data unit to another). Here's the standard notation for varying-length characters:

`CHARACTER VARYING (n)`

"n" is a number that identifies the maximum or assigned field length.

`VARCHAR2` and `VARCHAR` are two of the well-known variable-length data types. ANSI (American National Standards Institute) considers `VARCHAR` as the standard data type for variable-length characters. Because of this, popular services such as MySQL and Microsoft SQL Server use it for their regular operations. Oracle, a top-notch database system provider, uses `VARCHAR2` and `VARCHAR`. Character-defined columns may hold alphanumeric data: you can enter letters and numbers into these columns.

Varying-length data types don't need spaces to fill excess fields. For example, if the assigned length for a field is 15, and you enter a string of 11 characters, the overall length of that value is just 11. You don't have to use spaces to populate empty places.

### **Important Note:**

❖ If you are working with variable character strings, you should use a varying-length data type. This allows you to maximize database space.

## **The Numeric Values**

A numeric value is stored in a field defined as a number. Numeric values are commonly referred to as `REAL`, `NUMBER`, `DECIMAL`, and `INTEGER`.

Here are the SQL standards for numeric values:

`INTEGER`

`FLOAT(p)`

`BIT(n)`

`REAL(s)`

DECIMAL(p, s)

BIT VARYING(n)

DOUBLE PRECISION(P)

“p” is a number that shows the maximum or assigned field length.

“s” is a number located at the right side of a decimal point (e.g. 15.ss).

## **The Decimal Values**

A decimal value is a numeric value that contains a decimal point. Here is the SQL standard for decimal values. Remember that p represents precision and s represents the scale of the decimal.

DECIMAL(p, s)

Important Notes:

- ❖ Precision is the overall length of a numeric value. For this value: (5.3), 5 is the precision. It is the length assigned to a numeric value.
- ❖ Scale refers to the total number of digits found on the decimal point's right side..

## **The Integers**

Integers are numeric values that don't involve a decimal point. That means integers are whole numbers (regardless if they are negative or positive). Here are some examples of valid integers:

2

0

-3

99

-199

200

## **The Floating-Point Decimals**

A floating-point decimal is a decimal value whose scale and precision have varying lengths. Virtually, floating-point decimals have unlimited character lengths. That means any scale and precision is valid. The data type called “REAL” assigns a column that holds single-precision, floating-point

decimals. The data type called “DOUBLE PRECISION,” on the other hand, assigns a column that holds double-precision, floating-point decimals.

A floating-point number is only considered as single-precision if its precision is 1 to 21. A floating-point number will be considered double-precision if its precision is 22 to 53.

## **Time and Dates**

Obviously, these data types are used to record data regarding time and dates. Typical SQL distributions support data types known as DATETIME. Here are some of the popular members of this category:

TIME

DATE

TIMESTAMP

INTERVAL

When working with these data types, you’ll encounter the following elements:

Day

Year

Hour

Second

Minute

Month

## **The Literal Strings**

Literal strings are series of characters (e.g. names, phone numbers, etc.) that are specified by a program or database user. In general, a literal string is composed of data with similar characteristics. The value of the entire string is identified. The column’s value, on the other hand, is often unknown since different values exist between data columns and data rows.

When using literal strings, you don’t really specify the data types. You are just specifying the strings. The following list shows some literal strings:

‘Morning’

50000

“50000”

5.60

‘August 1, 1991’

Alphanumeric strings require quotation marks (either single or double). Number strings, on the other hand, can be stored without any quotation mark.

## **The Null Values**

Basically, null values are missing values or columns in a data row that hasn't received a value yet. These values play an important role in almost every aspect of SQL. You will use null values in creating tables, assigning search conditions, and entering literal strings.

If you need to reference null values, you may use the following methods:

‘ ’ (i.e. single quotation marks with a space between them)

Enter “NULL” (i.e. the word NULL itself)

When working with a null value, you should know that data doesn't have to be entered in any field. If the fields you are working on require data, use a data type followed by NOT NULL. If there's a possibility that a field doesn't require data, you should use the null data type.

## **The Boolean Values**

Boolean values are values that can be NULL, TRUE, OR FALSE. You should use BOOLEAN values when comparing data units. For instance, when you specify parameters for a search, every condition results to either a FALSE, TRUE, OR NULL. If all of the parameters return the BOOLEAN value of NULL or FALSE, data might not be retrieved. If the value is TRUE, however, data is retrieved.

Here's a simple example:

```
WHERE NAME = 'SMITH'
```

A database user may have used this line to perform a search. The system will evaluate this line for each data row. If NAME's value is equal to SMITH in one of the data rows, the search gives TRUE as the result. Afterward, the user



will get the data linked with that search result.

## Chapter 4: Exploring a Database

A database is properly organized and collected data that are stored and can be accessed through electronic means from a particular computer. Formal design and modeling methods are essential in developing a database. Before a designer develops a database, his first assignment is to create a conceptual data model that reflects the type of information to be kept in the database.

The database management system provides the access to the stored data. The DBMS interacts with application users, and the database to analyze and capture the stored data. It also allows end users have seamless interaction with more than one database. Although there may be restrictions in accessing some data, however this will be based on the security level made by the programmer or database owner. The functions of a database are diverse; it allows entry of information, storage of data and also retrieving large files of information. Aside from doing all of these, it also provides ways in which the data stored can be managed and organized effectively.

### The Functions of a Database Management Systems

A database is known as the collection of data and a common example is a spreadsheet or card index form. The DBMSs manages the data stored in the below four categories.

**Data Definition:** Data definition consists of defining the structure of your data, creation and modification of your data.

**Update:** After you have saved your data on your database, the DBMS gives you the permission to insert new information, alter or modify the former information, or even delete any of the information stored.

**Retrieval:** DBMS helps you provide information in a readable and usable form. The stored data when retrieved may be available in the exact format it was saved, or a new format.

**Administration:** For every data stored, there is an end-user, data and a future need for the data. Every data stored needs to be secured properly, and DBMS makes it easy for the application administrator to register and monitor the data effectively, enforces data security to prevent unauthorized personnel from accessing the data, to maintain the integrity and value of the data without any corruption.

### Benefits of Database

Database helps to facilitate the development of a new application, improves factors like data security, data access to enabled users, promotes data integrity and also reduces data errors and redundancy.

## **Classification of Database**

Database can be classified using the following criteria; type of the contents stored, application area and technicality of the database, such as the structure of the database. The classification you are about to read is made based on database distribution, and there are four major distribution systems for DBMS.

**Centralized Systems:** Databases that are stored in a single location so that the data can be accessed and used by other systems are stored with a centralized database system.

**Distributed Database System:** The actual databases in this system are distributed evenly from different locations, which are connected to a computer network.

**Homogenous Distributed Database System:** The same DBMS is used in this database system, though they are from different location.

**Heterogenous Distributed Database System:** In this database system, different DBMS are used by different sites, however there is a presence of additional software to support easy exchange of data.

## **Types of Databases**

The type of database you want is based on the usage requirements. You are about to learn the available database in the market today.

### **Centralized database**

Centralized database is abbreviated as CDB, it is a database that is stored, located and maintained in a single location, and this location can be a database system or in the central computer. Centralized database is often used by organizations like a business company or a university.

The centralized database has a good number of advantages compared to other types of databases. CBD maximizes the quality of a data properly, helping the data to maintain its integrity, accuracy, and reliability. For the fact that there is only one storage location, there is a higher level of security, and better data preservation. Although CBD is easy to maintain and change at any

time, given it needs to be updated, yet it is highly dependent on network connectivity.

The disadvantages of CBD are few, but they are however detrimental to the organization. If for any reason the unexpected occurs, all data will be lost and it may be difficult to retrieve the data, because it was saved in one location.

### **Distributed database**

When information is stored on multiple storage devices, or locations, it is called distributed database. This information may be stored on different computers in the same physical place, but they are not stored on the same processor. Distributed database can be subdivided into two groups namely, homogenous database and heterogeneous database.

Distributed database involves two processes, the replication and duplication.

Replication is the use of specific software that looks for changes in the distributed database. The replication process is complex and time consuming.

While duplication identifies one of the stored databases as the master and duplicates what it does. However, both replication and duplication can keep the integrity of the data in all distributed locations.

The distributed database should not be confused for centralized database. Although the distributed database has a multiple storage location, it still relies on the central DBMS which manages the storage devices remotely. The benefits of having a distributed database will increase the reliability and availability of your data, easy expansion improved performance and it is also a means of saving cost.

DBAs are very complex, which makes it a disadvantage because it needs extra effort to ensure that all data are evenly distributed to all locations. It also needs experienced hands to be able to manage the data, and provide maximum security on the stored data.

### **Personal database**

A personal database is a stored data on a personal computer, it is created to be used, organized and managed by one person. It can be created by a computer programming language like Practical Extraction and Reporting Language (PERL). A personal database only supports one application, makes use of one computer.

## **End-user database**

An end user is not usually concerned about the about the kind of transactions and operations done, but it is only concerned about the software or application. The end user database is specially meant for the end user, who can be a manager who summarizes the collected information in a database.

## **Commercial database**

Commercial databases are information collected and presented electronically, which are managed by commercial entities. These databases are stored, and maintained for customers' services, they are available to customers for commercial purposes. There are 5 reliable sources where you can access a commercial database; a commercial database retailer, internet sources, library or places with site license, a help of a professional, and personal subscription source.

Commercial databases have advantages and disadvantage, like every other database.

The commercial database requires no physical space, since it is online, unlike printed materials that need library or bookshelf. Aside from this part, the database can be easily monitored, and can be accessed by multiple users at the same time.

The disadvantages of commercial database are that it needs constant network connectivity and cost of management can be expensive for both the administrator and user.

## **NoSQL database**

A NoSQL database was originally referred to as a non-relational database or a non-SQL. It is used in managing big data, and real time web applications. It is a database that is built to provide mechanism for retrieval and storage of data that are modeled in other means used in such relational database. There are different classifications, and sub-categories of NoSQL database.

## **The Basic Classification of NoSQL Based On Data Model**

### **Column**

Column is the lowest level NoSQL object in a key space, which consists of a name, value and timestamp. The name is a unique factor that is used to reference the column. On other hand, the content of the column is referred to

as the value and the timestamp is used to determine the validity of the content.

- HBase is an open source non-relational database written in Java language and modelled after Google's Bigtable
- Cassandra is a Java based system that is designed to manage large data across commodity servers, and provide availability without any iota of error or failure.
- Scylla is an open source distributed database that was designed to work with the Casandra database.
- Accumulo is a popular NoSQL database that is written in Java language. It is ranked as the third most popular column database.

### **Document**

This is a computer program that was designed for the storage and retrieval of document oriented information. The document NoSQL category is also subdivided into fourteen.

- Apache
- CouchDB
- ArangoDB
- BaseX
- Clusterpoint
- Couchbase
- Cosmos
- DB

### **Operational database**

An operational database is used to manage stored data in real time. Organizations make use of operational database because things like clients' information, payroll records and employees bio-data

### **Relational database**

A relational database consists of set of tables where data can be accessed in various ways without an enabled user having to reorganize the tables. Relational database allows users to access and categorize stored data that can

later be filtered to retrieve data. Relational database shares few similarities with a database; however, the major difference between a database and a relational database is that the later stores data in rows and columns while the former stores data in files or document.

### **Cloud database**

This is a database that is kept and managed on a cloud application. The benefits of cloud database include fast automated recovery for lost files, increased accessibility, maintenance of in-house hardware, and a better performance. Despite all of these advantages, cloud databases have their own drawbacks such as security issues and potential loss of data in case of bankruptcy or disaster.

### **Object-oriented database**

This is a database management system that represents information as objects.

### **Graph database**

Graph database (GDB) is a database that uses graph structures for semantic queries that have nodes and edges to store data. It is a part of NoSQL database, created for the purpose of addressing limitations attached to the existence of relational databases. GDB is designed to allow a fast retrieval of complex data.

There are notable graph database that have used for various purpose and here are their descriptions.

- AllegroGraph was designed as a metadata data model, and its programming language is C, C#, Common Lisp, Java and Python.
- Amazon Neptune is a graph database that is managed Amazon. Notable users of Amazon Neptune databases include Samsung Electronics, Intuit, FINRA, Siemens, AstraZeneca, LifeOmic, Blackfynn and Amazon Alexa.
- AnzoGraph was developed to interact with sets of semantic data. Its programming is C, C++
- ArangoDB is a multi-model database that supports key, documents and graphs which are the three important data models. It uses languages like C++, JavaScript, .NET, Java, Python, Ruby, Node.
- DataStax is a distributed real time database that supports Cassandra. It is a

Java only database.

- Microsoft SQL Server is software developed by Microsoft, to store and retrieve data requested by other software applications.



## Chapter 5: Creating a Database

Before you can be able to do anything on your data, create a database. My assumption is that you have installed either MySQL or SQL Server in your computer.

To create a database in SQL, we use the CREATE DATABASE statement. This statement takes the syntax given below:

CREATE DATABASE database\_name;

First, login to MySQL by running the following command:

```
mysql -u root -p
```

Now you have logged into the MySQL database, it is time for you to create a new database. In the command given below, we are creating a database named school:

CREATE DATABASE school;

```
mysql> CREATE DATABASE school;
Query OK, 1 row affected (0.00 sec)

mysql>
```

The output shows that the database was created successfully. However, it will be good for you to confirm whether or not the database was created. To do this, use the SHOW command as shown below:

SHOW databases;

```
mysql> SHOW databases;
+-----+
| Database |
+-----+
| information_schema |
| company1 |
| easydrive |
| library_system |
| movies |
| mysql |
| performance_schema |
| school |
| sys |
| wordpress |
+-----+
10 rows in set (0.00 sec)

mysql>
```

The above output shows that the school database was created successfully. The above command returns the list of databases you have in your system.

An attempt to create a database that already exists generates an error. To confirm this, try to recreate the school database by running the following command:

```
CREATE DATABASE school;
```

```
mysql> CREATE DATABASE school;  
ERROR 1007 (HY000): Can't create database 'school'; database exists  
mysql>
```

The above output shows an error because the database already exists. To avoid this error, we can use the optional clause IF NOT EXISTS. This is showed below:

```
mysql> CREATE DATABASE IF NOT EXISTS school;  
Query OK, 1 row affected, 1 warning (0.00 sec)  
  
mysql>
```

The statement executed without returning an error.

Once you have created a database, it doesn't mean that you have select it for use. The target database must be selected using the USE statement. To select the school database, for example, run the following command:

```
USE school;
```

After running the above command, the school database will receive all the commands you execute.

## RENAME Database

Sometimes, you may need changing the name of a database. This is after you realize that the name you have given to the database is not much relevant to it. You may also need giving the database a database name. This can be done using the SQL RENAME DATABASE command.

The command takes the syntax given below:

```
RENAME DATABASE old_database_name TO new_database_name;
```

For example, in my case, I have the following list of databases:

```
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| company1 |
| easydrive |
| library_system |
| movies |
| mysql |
| performance_schema |
| school |
| sys |
| wordpress |
+-----+
10 rows in set (0.01 sec)

mysql>
```

Let us rename the database named movies by giving it the name movies\_db. This means we run the following command:

```
RENAME DATABASE movies TO movies_db;
```

The database should be renamed successfully.

## Database Backup

It is always important to back up your database. This is because an unexpected and unforeseen event may happen to the database. Examples of such unforeseen events include Cyber-criminality and natural disasters. In case of such an occurrence, it will be impossible for you to recover your database if you had not backed it up. However, if your database had been backed up, it will be easy to recover the database and resume normal operations. You also need to back up your database to prevent the loss of your data. SQL provides you with an easy way of creating a backup of your database.

To create a database backup, you use the BACKUP DATABASE command. This command takes the syntax given below:

```
BACKUP DATABASE database_name
```

```
TO DISK = 'file_path';
```

The database\_name parameter denotes the name of the database you need to back up. The file\_path parameter denotes the file leading to the directory

where you need to back up your database. The above command should be done when you need to back up the database from the beginning.

However, the differential command can be used if you need to create a differential backup. When you do this, the backup will only be created from the time you did your last full backup of the database. To do this, you must change the command to:

```
BACKUP DATABASE database_name  
TO DISK = 'file_path'  
WITH DIFFERENTIAL;
```

We have changed the command by adding the WITH DIFFERENTIAL statement. This means that the command will perform a differential backup on the database.

Suppose we need to create a full backup of the database named school. We will store the backup file in the local disk D.

The following command will help us accomplish this:

```
BACKUP DATABASE school  
TO DISK = 'D:\schoolDB.bak';
```

If you need to create a differential backup of the database, just run the following command:

```
BACKUP DATABASE school  
TO DISK = 'D:\schoolDB.bak'  
WITH DIFFERENTIAL;
```

We have just added the WITH DIFFERENTIAL statement to the command.

Note that with a differential backup, the backup will be created within a short time. This is because you are only backing up changes that have occurred within a short period. However, a full backup will take a longer time to complete.

## **Creating Tables**

To create a table, we have to define the table name, its column names and their corresponding data types.

We create tables by running the CREATE TABLE command. The tables are

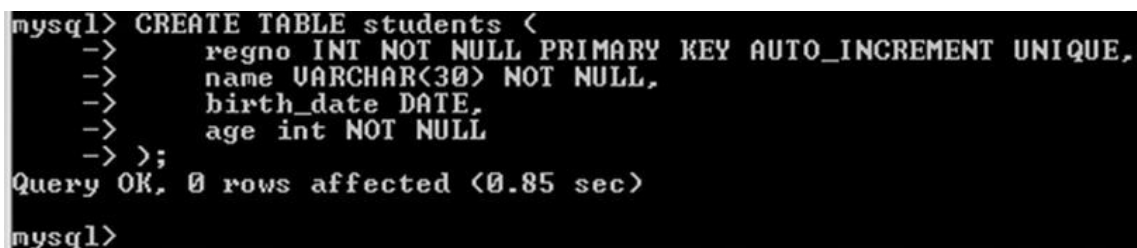
created within the database, meaning that one database can have many tables. The tables hold the data. The table stores the data in the form of rows and columns. The CREATE TABLE command takes the syntax given below:

```
CREATE TABLE tableName (  
    column1_name datatype constraints,  
    column2_name datatype constraints,  
    ....  
);
```

To make it the above syntax easy to understand, we need to create a table within the school database. Just run the following command from the MySQL command line.

-- Syntax for MySQL Database

```
CREATE TABLE students (  
    regno INT NOT NULL PRIMARY KEY AUTO_INCREMENT  
    UNIQUE,  
    name VARCHAR(30) NOT NULL,  
    birth_date DATE,  
    age int NOT NULL  
);
```



```
mysql> CREATE TABLE students (  
->     regno INT NOT NULL PRIMARY KEY AUTO_INCREMENT UNIQUE,  
->     name VARCHAR(30) NOT NULL,  
->     birth_date DATE,  
->     age int NOT NULL  
-> );  
Query OK, 0 rows affected (0.85 sec)  
mysql>
```

In the above example, we have created a table named students with 4 columns. These columns include regno, name, birth\_date, and age. Note that the name of each column has been followed by a data type representation, which dictates the type of data that the column will accept for storage. For the case of some data types, we can define the length of data we will store in it. This means you will not store more characters that exceed what you have defined.

If you attempt to create a table that already exists, you will get an error. Let us show this by trying to recreate the students' table:

```
mysql> CREATE TABLE students (<
->     regno INT NOT NULL PRIMARY KEY AUTO_INCREMENT UNIQUE,
->     name VARCHAR(30) NOT NULL,
->     birth_date DATE,
->     age int NOT NULL
-> );
ERROR 1050 (42S01): Table 'students' already exists
mysql>
```

The statement generates an error as shown above.

To avoid this, the IF NOT EXISTS statement can be used. This will create the table if doesn't exist and exit without an error if the table exists. It can be used as follows:

-- Syntax for MySQL Database

```
CREATE TABLE IF NOT EXISTS students (
    regno INT NOT NULL PRIMARY KEY AUTO_INCREMENT
    UNIQUE,
    name VARCHAR(30) NOT NULL,
    birth_date DATE,
    age int NOT NULL
)
```

The statement runs without an error as shown below:

```
mysql> CREATE TABLE IF NOT EXISTS students (<
->     regno INT NOT NULL PRIMARY KEY AUTO_INCREMENT UNIQUE,
->     name VARCHAR(30) NOT NULL,
->     birth_date DATE,
->     age int NOT NULL
-> );
Query OK, 0 rows affected, 1 warning (0.00 sec)
mysql>
```

## RENAME TABLE

Sometimes, you may need to change the name of your database table. This could be after you realize that the current name of the table is not meaningful. It then becomes necessary for you to change the name of the table. This is possible by the use of the RENAME TABLE command. The command takes the following syntax:

```
ALTER TABLE old_table_name
```

```
RENAME TO new_table_name;  
;
```

For example, to change a table named employees to workers, we run the following command:

```
ALTER TABLE employees  
RENAME TO workers;
```

The table will be renamed successfully.

**ALTER TABLE**

This is the statement we use any time we need to change the columns of a table. The modification in this can mean the need to add, change or delete a table column. This command can also help you add or drop the various constraints that may have been imposed on an existing table.

### **Adding a Column**

To add a new column to a table, use the ALTER TABLE command with the following syntax:

```
ALTER TABLE tableName ADD columnName column-definition;
```

However, you may sometimes to add multiple columns to the table. In such a case, use the command with the following syntax:

```
ALTER TABLE tableName  
ADD (column1 column-definition,  
      column2 column-definition,  
      ....  
      column_n column-definition);
```

Let us show this with an example. Suppose we have the workers table with the following columns:

```
mysql> desc workers;
```

Field	Type	Null	Key	Default	Extra
ID	int(11)	NO	PRI	NULL	
NAME	varchar(15)	NO		NULL	
ADDRESS	char(20)	YES		NULL	
AGE	int(11)	NO		NULL	
SALARY	decimal(16,2)	YES		NULL	

```
5 rows in set (0.00 sec)
```

We need to add a column named home\_town in which we will add the hometown for each worker. We can do this by running the following command:

```
mysql> ALTER TABLE workers ADD home_town varchar(15);
Query OK, 6 rows affected (0.38 sec)
Records: 6 Duplicates: 0 Warnings: 0

mysql>
```

ALTER TABLE workers ADD home\_town varchar(15);

We have added a column named home\_town of the varchar data type. To confirm whether this happened, we can describe the table again:

```
mysql> desc workers;
```

Field	Type	Null	Key	Default	Extra
ID	int(11)	NO	PRI	NULL	
NAME	varchar(15)	NO		NULL	
ADDRESS	char(20)	YES		NULL	
AGE	int(11)	NO		NULL	
SALARY	decimal(16,2)	YES		NULL	
home_town	varchar(15)	YES		NULL	

```
6 rows in set (0.03 sec)

mysql>
```

It is very clear that the table was added successfully.

## Modifying a Column

Sometimes, you may need changing a particular column table. This can be done using the ALTER TABLE command with the following syntax:

ALTER TABLE tableName MODIFY column\_name column\_type;

If you need to change over one table columns, use the command with the following syntax:

ALTER TABLE tableName



```
MODIFY (column1 column_type,  
        column2 column_type,  
        ....  
        column_n column_type);
```

Consider the workers table with the following data:

```
mysql> desc workers;  
+-----+-----+-----+-----+-----+-----+  
| Field | Type          | Null | Key | Default | Extra |  
+-----+-----+-----+-----+-----+-----+  
| ID    | int(11)       | NO   | PRI | NULL    |       |  
| NAME  | varchar(15)   | NO   |     | NULL    |       |  
| ADDRESS | char(20)      | YES  |     | NULL    |       |  
| AGE   | int(11)       | NO   |     | NULL    |       |  
| SALARY | decimal(16,2) | YES  |     | NULL    |       |  
| home_town | varchar(15)  | YES  |     | NULL    |       |  
+-----+-----+-----+-----+-----+-----+  
6 rows in set (0.03 sec)  
  
mysql>
```

We need to change the datatype of the column named home\_town from varchar(15) to varchar(20). To do this, we run the command given below:

```
ALTER TABLE workers MODIFY home_town char(20);
```

The command should run successfully as follows:

```
mysql> ALTER TABLE workers MODIFY home_town char(20);  
Query OK, 6 rows affected (0.95 sec)  
Records: 6  Duplicates: 0  Warnings: 0  
  
mysql>
```

To confirm whether the change happened, we can describe the table as follows:

```
mysql> desc workers;  
+-----+-----+-----+-----+-----+-----+  
| Field | Type          | Null | Key | Default | Extra |  
+-----+-----+-----+-----+-----+-----+  
| ID    | int(11)       | NO   | PRI | NULL    |       |  
| NAME  | varchar(15)   | NO   |     | NULL    |       |  
| ADDRESS | char(20)      | YES  |     | NULL    |       |  
| AGE   | int(11)       | NO   |     | NULL    |       |  
| SALARY | decimal(16,2) | YES  |     | NULL    |       |  
| home_town | char(20)     | YES  |     | NULL    |       |  
+-----+-----+-----+-----+-----+-----+  
6 rows in set (0.08 sec)  
  
mysql>
```

The above output shows that the change was successful.

## Renaming a Column

You may need to change the name of a certain table column. This could be when you need to give the table column a more meaningful name. To make this change, use the ALTER TABLE command with the following syntax:

```
ALTER TABLE "tableName"
```

```
Change "column_1" "column_2" ["Data Type"];
```

The parameter column\_1 should be the old column of the table while the parameter column\_2 should be the new name you need to assign to the column. If you don't specify the old column name correctly, you will get an error since the column won't be found.

Consider the workers' table with the following columns:

```
mysql> desc workers;
```

Field	Type	Null	Key	Default	Extra
ID	int(11)	NO	PRI	NULL	
NAME	varchar(15)	NO		NULL	
ADDRESS	char(20)	YES		NULL	
AGE	int(11)	NO		NULL	
SALARY	decimal(16,2)	YES		NULL	
home_town	char(20)	YES		NULL	

```
6 rows in set (0.30 sec)

mysql>
```

Suppose we need to change the column named home\_town to home\_city. We can achieve this by running the following command:

```
ALTER TABLE workers
```

```
Change home_town home_city varchar(20);
```

The command should run successfully as shown below:

```
mysql> ALTER TABLE workers
-> Change home_town home_city varchar(20);
Query OK, 6 rows affected (0.23 sec)
Records: 6 Duplicates: 0 Warnings: 0
```

We can describe the table to see whether or not the change was reflected:

```
mysql> desc workers;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| ID    | int(11)       | NO   | PRI | NULL    |       |
| NAME  | varchar(15)   | NO   |     | NULL    |       |
| ADDRESS | char(20)      | YES  |     | NULL    |       |
| AGE   | int(11)       | NO   |     | NULL    |       |
| SALARY | decimal(16,2) | YES  |     | NULL    |       |
| home_city | varchar(20)  | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.09 sec)

mysql>
```

## Dropping Columns

Sometimes, you may need to reduce the number of columns you have in a certain table. In such a case, it will be necessary for you to drop or delete some table columns. It is possible with SQL.

To delete a table column, we combine the ALTER TABLE and the DROP statements. This is done using the syntax given below:

ALTER TABLE tableName DROP COLUMN column\_name;

Again, we will use the workers table to show how to use the above command. The table has the following set of columns:

```
mysql> desc workers;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| ID    | int(11)       | NO   | PRI | NULL    |       |
| NAME  | varchar(15)   | NO   |     | NULL    |       |
| ADDRESS | char(20)      | YES  |     | NULL    |       |
| AGE   | int(11)       | NO   |     | NULL    |       |
| SALARY | decimal(16,2) | YES  |     | NULL    |       |
| home_city | varchar(20)  | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.04 sec)

mysql>
```

We need to delete the last column of the table, which is home\_city. To delete it, we run the command given below:

ALTER TABLE workers DROP COLUMN home\_city;

The command should run successfully as shown below:

```
mysql> ALTER TABLE workers DROP COLUMN home_city;
Query OK, 6 rows affected (0.29 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql>
```

To confirm whether or not the column was deleted, we run the describe command against the table as shown below:

```
mysql> desc workers;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| ID     | int(11)       | NO   | PRI | NULL    |       |
| NAME   | varchar(15)   | NO   |     | NULL    |       |
| ADDRESS | char(20)      | YES  |     | NULL    |       |
| AGE    | int(11)       | NO   |     | NULL    |       |
| SALARY | decimal(16,2) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.05 sec)

mysql>
```

The above figure shows that the column deleted from the table successfully.

It is important to know the various data types supported in SQL. SQL data types specify the data for every object. Every variable, column and expression in SQL has an associated data type.

# **Chapter 6: Getting Started with Queries**

## **How Do Queries Work?**

Queries are a principal working flow for SQL servers. Without querying operations, it is technically impossible to maintain or regulate database operations.

For successful querying database checks the following essentials:

- 1) Does the authorized user have permissions to execute this statement? If not terminate the statement and show Authorization error.
- 2) If the user is authorized then the SQL server will check whether the authorized user has permission to retrieve the data he is asking for? If not terminate the procedure and display data retrieval errors for the user.
- 3) If the author is authorized and has permission to retrieve the data then the SQL server will check the SQL syntax that the user has entered. If SQL syntax is correct then the querying process will start and display results to the end-user. If the SQL syntax is not right then syntax error will be shown.

If the statement entered satisfies these three complexities then your statement will be sent to a query optimizer to analyze the syntax in detail.

## **What is a Query optimizer?**

The query optimizer is a system that helps you to analyze the syntactical structure of the syntax given. The optimizer will first check at all the from clauses that are present and then will look at all of the indexes that are supported. After analyzing all the clauses optimizer will choose the best way to query the operation provided. According to the resources provided it will proceed.

## **What Happens Next?**

After an execution procedure is selected by the Query optimizer it is then sent to the tool that you are using. This tool is the SQL application itself. Then the SQL application will conduct the procedure and display results for the query operation. If the operation succeeds the result will be shown if not then an error will appear. There is also a chance for the query process with no results.

Before knowing about the SELECT function it is necessary to know about

the importance of the FROM clause.

## **FROM**

FROM is a simple SQL clause that helps you to point out an instance. When using the SELECT statement it is mandatory to look out at an instance for selecting the columns. This is where the FROM clause helps you to select the columns from the databases.

An SQL statement is here:

```
FROM {instance name}
```

Here the instance name belongs to either database or table name. You can also individually enter the column or primary key values to extract information.

As we have sufficient knowledge about the FROM statement now, we will discuss the SELECT clause in detail.

## **SELECT**

The SELECT statement is usually used to select the columns provided using the logical entities that are given. The select clause can often be simple but sometimes it may lead to complex query operations.

Usually, select statements are performed at the end of the query operation by the SQL server software because it is technically not possible to select the things without having an update and changed table. For this reason, SELECT statements are usually performed at the end and often take much longer time than the other query operations.

Here is the SQL statement:

```
SELECT {Table or column values}
```

```
-> FROM {Instance name}
```

Now, we will discuss the parameters that are present in the syntax.

1) Table/ Column values - In this instance you need to enter all the columns that you are going or willing to select by this query. You can use logical expressions to denote the columns you need to select automatically. If you want to select all the columns present then you can use an asterisk '\*'.

2) Instance names - In this, we need to enter the database name we need to retrieve the columns from. Without mentioning FROM statement, it is

technically not possible to select the columns.

To say in a single sentence, the Select clause is used to determine all the columns for a query operation.

What else can you do with a SELECT statement?

- 1) You can use any numerical information such as the primary key id for selecting it.
- 2) You can enter various logical and conditional expressions to expect automated results.
- 3) You can sometimes also use inbuilt or scalar functions that SQL offers for its users.
- 4) If there is no system function satisfying for your requirement then you can build a user-defined function from scratch and use it for selecting.

## **ALIASES**

Aliases are column names that are generated for the queries. Usually, SQL generates custom column names for the query results. However, they are often clumsy and it is essential to generate our columns for both query results and function results. To make your column names usually aliases are used.

Aliases can also be used to return column names that are generated using logical expressions.

Here is the SQL statement:

ALIAS {Column name} {Logical expression}

## **Creating Indexes**

When querying large amounts of data, indexing technology can be applied. An index is a special type of database object that holds the sort structure of one or more columns in a data table. Effective use of indexes can improve the efficiency of the data query. This part focuses on the creation of indexes and related maintenance work.

An index is a separate, physical database structure. In SQL Server, an index is a decentralized storage structure created to speed up the retrieval of data rows in a table. It is built for a table, and the rows in each index page contain logical pointers to physical locations in the data table to speed up retrieval of physical data.

Therefore, whether to create indexes on the columns in the table will have a great impact on the query speed. The storage of a table is composed of two parts, one part is used to store the data pages of the table and the other part is used to store the index pages.

Usually, index pages are much smaller for data pages. In data retrieval, the system first searches the index page to find the pointer of the required data, and then directly reads the data from the data page through the pointer, thus improving the query speed.

The way the database uses indexes is very similar to the catalog of books. Indexes allow a chance of viewing through a complete book. In a database, indexes can be used for introspecting a table without having to scan the entire table.

After understanding the basic concepts of indexes, the following describes the advantages and disadvantages of using indexes and the conditions for using indexes.

### **1. Advantages of using indexes**

- (1) Create a unique index to ensure the uniqueness of each row of data in the database table.
- (2) The creation of indexes is directly proportional to the increase in the speed of the system.
- (3) The accelerator and the table are of special significance in realizing the extra advantages of the data.
- (4) When using GROUP BY and ORDER BY you should be aware that the grouping and ordering time in the query can also be reduced.

### **2. Disadvantages of Using Index**

- (1) It is a very high turmoil to look at indexes and maintain indexes, which increases with the increase in data volume.
- (2) The index needs a lot of storage indication. In addition to the data table occupying data space, each index also occupies a certain amount of physical space. If cluster indexes are to be established, the space required will be larger.
- (3) When adding, deleting and modifying the data irrespective of circumstances, the index is used in a separate mechanism such that it reduces



the overall capability of the system.

## **Classification of Indexes**

The data in the database page of SQL Server can be divided into two types according to the storage structure: Clustered Index and Non clustered Index.

### **1. Cluster index**

Cluster index (also called "clustered index") means that the physical storage order of data rows in the table is the same as the index order. Cluster index consists of upper and lower layers: the upper layer is an index page, which contains index pages in the table and is used for data retrieval; The lower layer is the data page, which contains the actual data page and stores the data in the table.

When creating a clustered index in a column of a table, the data in the table will be reordered according to the index order of the columns and the table will be modified. Therefore, only one cluster index can be created in each table. Cluster indexes are created on columns that are frequently searched or accessed sequentially.

### **2. Non-cluster index**

The non-clustered index (also called "non-clustered index") does not change the physical storage location of the data rows in the table. Data is stored separately from the index and is linked to the data in the table through the pointer carried by the index. Non-clustered indexes are similar to those in textbooks.

The data is present in one place, the index is present in another place, and the index has a pointer to the storage location of the data. The items in the index are stored in the order of index key values, while the information in the table is stored in another order (non-clustered indexes can specify this).

A table can contain multiple non-clustered indexes, and a non-clustered index can be created for each column commonly used when looking up data in the table.

The method of creating a clustered index is similar to that of the non-cluster index, both uses CREATE INDEX statement, except that the CLUSTERED clause needs to be specified to create a clustered index.

## **How to Create An Index?**

All you need to do is enter the following syntax for creating indexes. As explained above clustered index needs a separate parameter for creation.

Here is the syntax:

```
CREATE INDEX {Parameters}
```

In these parameters, we will define whether it is a clustered or non-clustered index.

You can also use SQL server management software to create indexes with a click.

### MySQL Functions

Functions are an easy way to repeat a task. Functions consist of a set of code that can be used in logical execution and expressions. SQL provides a lot of scalar and aggregation functions for the database users.

### **Why are Functions Used?**

For example, you can use SQL functions to find maximum or minimum values for your column values. You can also use functions created by yourself from scratch to repeat tasks automatically.

## Chapter 7: Turning Data into Information

Through this part, when we bring up the topic of the operators that are inside of the database, we are going to talk about the characters and the words that we have reserved, and any that can be used along with the WHERE clause of the statements we write. Operators are helpful because we are able to use them to perform operations in the statements, such as comparisons and any of the equations that we need in math, but they can also come in to make it easier to set up some of the parameters that we want to see around our statements. And we need always to remember that they are there to help us connect together two or more parameters that we have in the same statement if this is something that our code needs.

During this process, we are going to need to remember that there are a few options of operators that you are able to use in these statements. The four most common options that we are able to work with include:

Operators that are used to help negate conditions.

- Logical operators
- Comparison operators
- Arithmetic operators.

### Logical Operators

The first operator type that we are going to spend some time focusing on in this guidebook will be the logical operators. These are going to be the kinds of operators that we are able to use with the keywords of our statements, and they are going to be easier to form with comparisons inside of the statements. Some of the different logical operators that we are able to add into our statements will include:

- In—this operator will allow you to compare the value of specified literal values that are set. You will get a true if one or more of the specified values is equal to the value that you test.
- Like—the like operator is going to make it so that you can compare a value against others that are similar. You will usually combine it with the “\_” or the “%” sign to get more done. The underscore is used to represent a number or a character and the % is

used for several characters, zero, or one.

- Unique—with the unique operator, you will be able to take a look at one or more of your data rows and see if they are unique or not.
- Exists—you will need to use this operator to find the data rows that will meet the criteria that you put down. It basically allows you to put down some criteria and see if there are any rows that exist that meet with this.
- Between—you can use this operator in order to find values that will fall into a specific range. You will need to assign the maximum and the minimum values to make this work.
- Is null—the is null operator will allow you to compare the value of your choosing with a NULL one.
- Any and all—any and all values often go together. Any operator is going to compare a value against all of the values on a list. The list of values is to be set up with predetermined conditions. ALL, on the other hand, will compare the values that you select against the values that are in a different value set.

These are good operators to learn how to use because they will help us to make some good comparisons and to help us look at a few of the points of data that are already inside of our database that we are working with. You are able to work with a few of these logical operators and see how they can work with some of the different statements that you want to work with, and see what will be the best for you.

## **Comparison Operators**

The second operator that we are able to work with inside of the SQL language is going to be the comparison operators. These are going to be the ones that you would choose to use when you want to check on some of the single values that are found in the statement here. This category is going to be composed of a few mathematical signs, some that you are even familiar with so it is not going to be too hard for us to figure out. Some of the best comparison operators that we are able to work within this database will include:

- Non-equality—if you are testing the non-equality operator, you

will use the “<>” signs. The operation is going to give you the result of TRUE if the data does not equal and FALSE if the data does equal. You can also use the “!=” to check for this as well.

- Equality—you will be able to use this operator in order to test out some single values in your statement. You will simply need to use the “=” sign to check for this. You are only going to get a response if the data that you chose has identical values. If the values aren’t equal, you will get a false and if they are equal, you will get a true.
- Greater-than values and Less-than values—these two are going to rely on the “<” and “>” signs to help you out. They will work as stand-alone operators if you would like but often, they are more effective when you combine them together with some of the other operators.

These are going to help us to see some of the single values that are found in any of the statements of our database, and sometimes they are able to bring out some unique things like the less than and great than options, resulting in a few true and false statements that add in some power to the code we are writing and can make our program stronger than ever before.

## **Arithmetic Operators**

The third type of operator that we need to focus on is going to be the arithmetic operator. These are going to be the most useful when we are trying to add in some of the different mathematical operations that are useful in the SQL language. There are going to be four main arithmetic operators that we are able to use inside of these equations, and we are able to combine a few of them together in the same statement if that is what is needed for our code. The four main types of operators that we are able to focus on here will include:

- Addition—you will just need to use the “+” sign to get started on using addition. For this example, we are going to add up the values for the overhead column and the materials column. The statement that works for this is:
- `SELECT        MATERIALS        +        OVERHEAD        FROM  
                 PRODUCTION_COST_TBL.`

- Subtraction—you can also do some basic subtraction when it comes to your SQL statements. You will simply need to use the “-” sign to do this.
- Multiplication—it is possible to do multiplication from your tables as well. You would need to use the “\*” sign to do this.
- Division—this one will just need the “/” sign to get started.

With the arithmetic operators, you will be able to gain a lot of freedom in some of the codes that you are writing. You can combine a few of these operators together and even end up with your own equation inside of the statement. For example, you could have one statement with a bit of multiplication along with some division and addition if you would like. You can also have it so that you are just using a few multiplication signs in the same statement as well. Whatever is needed to bring the code together and make it work in the way that you want is available when it comes to these arithmetic operators.

One thing that we need to remember when we decide to add in more than one of these operators to our statement at a time is that we need to work with something known as the principles of precedence to make sure that we get it right. This means that the syntax is going to be able to take care of the arithmetic operators in a certain way to help us get the right answer.

When we are doing this kind of principle in SQL, the syntax is going to take care of everything that we need to multiply first, and then it will pay attention to everything that we need to divide, and then it will do those that should be added together, before finishing up with the things that should be subtracted from one another. And the syntax is going to do this in a manner that goes from left to right, along with going through the symbols in the order that we did above. This makes sure that we are going to get the right answers, and is similar to what we should remember from basic algebra classes in the past.

## **Working with the Conjunctive Operators**

Another type of operator that we need to spend some of our time on here is the conjunctive operators. When you are doing some work with your statements, you may find that there will be some times when new criteria will be needed in order to make sure that the command is going to behave the way that we want.

For example, if you are sending out a database search and your results that might be a bit more confusing, you may find that adding in some different criteria to the mix to get the results you want is a much better idea. You can also use these when you would like to be able to combine together a few different types of criteria within the single statement, and then you would create a brand-new conjunctive operator that works for your needs.

Even though you are able to create one of the conjunctive operators that you want technically, there are going to be a few of these that are already predefined inside of the SQL language, and you are able to use them as you need. Some of these operators are going to include:

- OR—you will use this statement to combine the conditions of the WHERE clause. Before this statement can take action, the criteria need to be separated by the “OR” or it should be TRUE.
- AND—this operator is going to make it easier to include multiple criteria into the WHERE section of your statement. The statement will then only take action when the criteria have been segregated by the “AND” and they are all true.

These operators are all important and each of the categories is going to make a big difference in what we are able to do with some of our own statements. You are able to use the arithmetic operators to help with mathematical equations, work on how to compare different parts of the database, and so much more. You will not have to work longer with the SQL language in order to see all of the cool things that you are able to do with this language in no time at all.

## **Chapter 8: Working with Multiple Tables**

Databases usually consist of different tables with different column values and types. However, some tables in the database consist of the same entities and can be used to query two database columns with precision. To retrieve data using this technique we use JOINS in SQL.

### **What are Joins?**

Usually, we can query columns and rows using the SELECT statement. However, SELECT statements only work for a single table and so is often not useful for complex query operations. At this Joins come into operative where they can query data from two or more tables with the help of common columns present in the tables.

### **What are Unions?**

Unions also are used to retrieve data from two or more tables. The only difference between Joins and Unions is that Joins use only SELECT statement whereas Unions use a different number of SELECT statement to achieve the results.

### **How to Implement Joins?**

Joins are usually implemented using two types of syntaxes. Explicit syntax deals with the cartesian product of two tables. Whereas implicit syntax deals with Where clauses. It is always recommended to use explicit functionality because it is easy for queries to function and operate.

### **Cartesian product**

Before going to learn about the advanced functionalities of Joins it is important to know about the simple cartesian product.

Cartesian product is a simple mathematical operation that multiplies with any entity without restrictions. By using the cartesian product you can combine with any column of the other table.

To understand the cartesian product in layman terms it is just keeping two tables side by side. For example, if Table A has 10 rows and Table B has 5 rows then the cartesian product will have a total of 50 rows.

Cartesian product is often under-discussed even though after being simple is because it gives large output tables.



This cartesian product from SQL is explicitly known as cross join. However, with the time SQL server has restricted cross join operations for its huge data taking.

**Here is the SQL statement for your reference:**

```
SELECT * [ CROSS JOIN] {Enter parameters here}
```

## **SQL Inner Join**

If cross join talks about the cartesian product of two tables natural join discusses natural join of tables.

### **What is a Natural Join?**

To explain natural join, we need to be aware of the tables we are trying to target. There will be two tables known as the Source table and Target table. The source table is which we use a SELECT statement to extract columns and the Target table is the table that we use after the JOIN statement.

To understand Natural joins in a better way we will discuss this concept in three scenarios. Follow along!

#### **Scenario 1:**

If there is only one same column in both of the tables

#### **Explanation:**

To demonstrate this example, we will introduce two tables known as College and Place.

College table has 4 columns namely - BRANCH, BRANCH\_NAME, HOD, STREAM

Place table has 6 columns namely - PLACE\_ID, STREET, PINCODE, TOWN, COUNTY, STREAM

Now, if we carefully observe you can notice that both tables contain a similar column known as STREAM. This column can be used as a Foreign key to perform a natural join.

*Here is the SQL statement to perform Natural join:*

```
SELECT {enter columns here} FROM { Source table} JOIN { target table}
```

Note:

Always enter all the columns you want to retrieve in the second parameter

position. For the table mentioned above, we will enter Stream as a parameter for join to function.

### **Scenario 2:**

If there are more than two identical columns

#### **Explanation:**

If you have observed more than two identical columns in the target and source table then it is important to enter only one column for retrieving user desired columns. This type of join is known as pure natural join. Entering one column can recognize all other columns and will give results.

Here is the SQL statement:

```
SELECT {enter columns here } FROM { Source table} JOIN { target table}  
AND { Linked columns}
```

### **Scenario 3:**

Retrieve columns by USING clause

#### **Explanation:**

If there are two columns and you need only one column you can use the USING column to retrieve the column you are wishing for.

Here is the SQL statement:

```
SELECT {enter columns here } FROM { Source table} JOIN { target table}  
USING { Required columns}
```

With this, we have given a thorough and complete introduction to inner join and in the next part, we will discuss Left and Right joins using a few examples. But before discussing these joins let us discuss equijoins and non-equijoins for better understanding of these concepts. Although not solely responsible for Right and Left joins these concepts are important for an overall understanding of the subject. Follow along!

### **What is Equijoin?**

Equijoin refers to comparing connected columns using an operator equal to "=" in the connection condition. The query result will list all columns in the connected table, including duplicate columns. The types of connection columns in connection conditions must be comparable, but not necessarily the same. For example, it can be both character type or both date type; It can

also be an integer type and a real type because they are both numeric types.

## **SQL RIGHT & LEFT JOIN**

SQL Right and Left join works on the concept of external join. The inner join compares each column of records FROM the table in the FROM clause and returns all records that satisfy the join condition.

However, sometimes it is necessary to display all records in the table, including those records that do not meet the connection conditions. In this case, the external connection is required. Using outer join can easily query other records in a table in the join result. The query result of the external connection is an extension of the query result of the internal connection.

An obvious feature of external connection is that some data that do not meet the connection conditions are also output in the connection results. External connection takes the specified data table as the main body and outputs the data in the main body table that do not meet the connection conditions. According to the different rows saved by external connections, external connections can be divided into the following 3 types of connections.

- (1) Left outer join: indicates that the results include data in the left table that do not meet the conditions.
- (2) right join: indicates that the results include data in the right table that do not meet the conditions.
- (3) Total External Connection: Data that do not meet the conditions in the left table and the right table appear in the results.

In the JOIN statement, the left side of the join keyword represents the left table and the right side represents the right table.

## **SQL Self-connection**

Self-connection refers to the connection between the same table and itself. All databases process only one row in a table at a time. Users can access all columns in one row, but only in one row. If information of two different rows in a table is needed at the same time, the table needs to be connected with itself.

To better understand self-joining, one table can be thought of as two independent tables. In the FROM clause, the tables are listed twice. To distinguish, each table must be given an alias to distinguish the two copies.

## **SQL UNION**

In SQL Server, combined queries are completed through the UNION operator. The UNION operator can be displayed by combining data FROM multiple tables, but unlike joins, UNION is not implemented by adding multiple tables in the from clause and specifying join conditions, but by combining the results of multiple queries.

The following points should be noted when using the UNION operator:

- (1) The number of columns in the two query statements is required to be compatible with the data type of the columns.
- (2) The column name in the final result set comes from the column name of the first SELECT statement.
- (3) In combination query, duplicate rows will be deleted from the final result set by default unless the ALL keyword is used.
- (4) The query results will automatically sort the columns in the SELECT list from left to right, regardless of the position of the query relative to the UNION operator.

Here is the syntax:

```
SELECT {UNION}
```

Merging multiple result sets through UNION is explained below by using a server management studio.

The operation steps are as follows:

- (1) Select Start → All Programs → SQL Server Management Studio in turn. In the pop-up connection dialog box, select "SQL Server Authentication", login name is "Your choice" and password is empty.
- (2) In the "Microsoft SQL Server Management Studio" window, click the "New Query" button, then enter the corresponding code in the code editing area, and click the Run button on the toolbar. The results will be displayed in the window.

```
SQL UNION {Enter parameters} Merge {Enter column names here}
```

## **SQL UNION ALL**

When the UNION statement is used to merge the result sets in the query statement, the UNION statement will automatically delete duplicate rows and

automatically sort the result sets. The UNION ALL statement does not delete duplicate rows in the result set and does not automatically sort. In most cases, UNION statements are used for merging, and UNION ALL statements are only used in some special cases.

For example:

- (1) Know that there are duplicate lines and want to keep the duplicate lines.
- (2) Know that there can be no repetition.
- (3) I don't care if there are duplicate lines.

The UNION ALL statement will only be used in the above cases, and the UNION statement is better for other cases.

### **Use UNION ALL to Preserve Duplicate Rows**

The operation steps are as follows:

- (1) Select Start → All Programs → SQL Server Management Studio in turn. In the pop-up connection dialog box, select "SQL Server Authentication", login name is "your choice " and password is empty.
- (2) In the "Microsoft SQL Server Management Studio" window, click the "New Query" button, then enter the corresponding code in the code editing area, and click the Run button on the toolbar. The results will be displayed in the window.

SQL UNION {Enter parameters} Duplicate {Enter column names here}

### ***Improving the Readability of Query Results through UNION Statements***

UNION statement can not only merge the query results of the SELECT statement but also increase the readability of query results. That is, by creating aliases for the column names in the results, the meaning of the query results can be better reflected.

### **Determine the Source of Data by Text in UNION**

In UNION, the source of data is determined by the text, mainly by adding a new column to the result set and determining the source of data in the table through the new column. This method can ensure that duplicate rows in two tables are not deleted, and can show that duplicate rows come from different tables.

## Chapter 9: Using Functions

In regard to SQL, a built-in function can be defined as a portion of programming that accepts zero or any other input and returns an answer. There are different roles of built-in functions. These include the performance of calculations, obtaining of the system value, and application in textual data manipulation. This part aims at examining the various SQL built-in functions, categories of functions, pros and cons of functions and types of built-in functions.

### Types of SQL Functions

The SQL functions are grouped into two groups: aggregate and scalar function. Working on Group by, the aggregate functions run on different records and deliver a summary. Scalar functions, on the other hand, run on different records independently.

These are as follows:

1. **Single Row Functions**-They provide a one-row return for any queried table. They are found in select lists, START WITH, WHERE CLAUSE and others. Examples of single-row functions include numeric\_, data\_mining, Datetime\_, conversion\_ and XML\_functions.
2. **Aggregate Function**-When you apply this kind of function, you see a single row returns based on different rows. The aggregate function exists in Select lists, ORDER BY and HAVING CLAUSE. They go hand in hand with Group by Clause and SELECT statements. Many of them do not take attention to null values. Those that are usually used include AVG, EANK, MIN, SUM and others.
3. **Analytic Function**-They are used to compute an aggregate value that are found on specific groups of rows. When you run this function, it delivers many rows for every group. The analytic functions are the last one to be run in a query. Examples of analytic functions include analytic\_clause and Order-by-Clause.
4. **Model Functions**-These are found in SELECT statements. Examples of model functions include CV, present and previous.
5. **User-Defined Function**-They are used in PL/SQL to offer functions that are not found in SQL. They are mostly used in sections where expressions occur. For instance, you can find them in select list of Select statement.

6. SQL COUNT Function-It is used to provide the number of rows in a table.

## **Categories of Functions**

Functions are classified according to the role they play on the SQL database. The following are some of the function categories available:

1. Aggregate Functions-They do a calculation on a specific set of values and deliver a single value. The aggregate values are used in the SELECT LIST and HAVING clause. The aggregate functions are referred to as being deterministic. This means that they return the same value when running on the same input value.

2. Analytic Function-They calculate an aggregate value according to a group of rows. They return many rows for different groups. The analytic functions can be used to perform different computations like running totals, percentages and others.

3. Ranking Functions-They provide a ranking value for each portioned row. These kinds of functions are regarded as nondeterministic.

4. Rowset Functions- They're used to return an object that can be applied.

5. Scalar Functions-They work on a single value to return the same. There are various kinds of scalar values. These include configuration function, conversion function and others.

a) Configuration Functions-They offer information about the present configuration.

b) Conversion Functions-They support data changing.

c) Cursor Functions-They provide information concerning cursors.

d) Date and Time Data Type-They are concerned with operations as regards date and time.

6. Function Determinism-Functions that are found in SQL servers are either deterministic or nondeterministic. Deterministic functions provide the same answers when they are used. Nondeterministic functions, on the other hand, offer different results when they are applied.

## **SQL Calculations**

There are various mathematical functions build-in the SQL server. The

functions can be classified into 4 main groups, including Scientific and Trig Functions, rounding functions, signs and random numbers. Although there are numerous mathematical functions within each class, not all of them are used regularly. The various classes are highlighted and explained below:

1. Scientific and Trig Functions-Under this category, there are various subclasses found under it. These include PI, SQRT, and SQUARE. PI function is used to compute the circumference and area in circles. How it works: `SELECT 2 *10`. SQRT connotes square root. This function is used most of the time. The function recognizes any number that can be changed to float datatype. Example: `SELECT SQRT (36)` Returns 6.SQUARE means that you multiply any number by itself. The concept of Pythagoras theorem is useful here. This means that  $A^2+B^2=C^2$ . This can be performed as `SELECT SQRT (SQUARE (A) + SQUARE(B)) as C`.

2. Rounding Functions- Under this class, there are various subcategories which include the CEILING and FLOOR. The ceiling function is helpful when dealing with a float or decimal number. Your interest is to find out the highest or lowest integer. Whereas the CEILING is the best highest integer, the floor represents the lowest integer. The ROUND function is applied when you want to round a number to the nearest specific decimal place. This is expressed as `ROUND (value, number of decimal places)`.

3. Signs- There are occasions that require that you obtain an absolute figure of a number. For instance, the absolute value of -7 is 7. The absolute number doesn't contain any sign. To assist you with this task, it's essential to utilize the ABS function.

4. COS (X)-This function recognizes an angle expressed as radian as the parameter. After an operation, you get a cosine value.

5. SIN (X)-This function notices a radian angle. After computation, it gives back a sine value.

6. Sign-You can use a sign function when you want a negative, positive, or zero value.

## **The Importance of SQL Built-In Functions and Mathematical Applications**

The build-in functions are sub-programs that help users to achieve different results when handling SQL database. These applications are used several



times when you want to manipulate or process data. The SQL functions provide tools that are applied when creating, processing, and manipulating data. The benefits of SQL in-built and maths functions are as follows:

1. **Manipulation of Data**-The in-built tools and maths functions play a significant role in data manipulation. Manipulating massive data may be difficult if you do it manually. This is especially when the data is massive. Therefore, these functions play a significant role in ensuring that your data is manipulated fast as per your demands.
2. **Assist in The Processing of Data**-To benefit from data; you must process it. You may never have the ability to process big data manually. Therefore, the built-in SQL functions and maths applications assist you in processing your database.
3. **Simplifies Tasks**-In case you're a programmer, you can attest to the fact that these functions and formulas make your work ease. You can work fast when you apply these in-built functions and formulas. Due to these tools, you'll accomplish many projects within a short time.
4. **Increase Your Productivity**-Using the in-built functions enhance your productivity as a programmer. This is because the functions enable you to work quickly on different projects. In case you were to handle data manually, you may take much time before you accomplish a task which ultimately injures your productivity. However, the built-in functions and calculations allow quick execution of tasks.
5. **Time Saving**-Because functions are written once and used several times, and they save much time. Besides timesaving, the functions offer support to modular programming.
6. **They Enhance Performance**- When you apply functions; you enhance the performance of your database. This is because the functions are prepared and inserted prior to usage.
7. **Enhances Understanding of Complicated Logic**-Handling of SQL database is a complex task. Therefore, functions enable you to decompose data into smooth and manageable functions. In this way, you find it easy to maintain your database.
8. **Cost Effective**-Because the functions are in-built in the SQL database; you can use them many times without the need to invest in new ones. In this

connection, therefore, they reduce the cost of operating and maintaining your SQL database.

## **Downsides of In-Built Functions**

The SQL in-built functions have various limitations, including:

1. Testability- When using the in-built functions, it's challenging to test their business philosophy. This is a big challenge, especially when you want the functions to support your business philosophy. You may never understand whether the functions are in line with your business vision and mission.
2. Versioning-It's challenging to establish the kind of version that is used in the SQL built-functions. You need to understand whether there're any new versions that can probably provide the best service.
3. Errors-In case there are errors within the in-built functions which you don't know, they may disrupt the program. This may prove costly and time-wasting.
4. Fear of Change-In case there is change; you may not understand how it will affect your SQL built-in functions. The world of technology keeps changes, and this change may affect the in-built functions.

The SQL in-built functions and calculations are essential as they enable a programmer to execute a task fast with minimal errors. The calculations in the in-built database makes it possible to process and manipulate data.

# Chapter 10: Subqueries

## Definition

In SQL, subqueries are queries within queries. The subqueries usually use the WHERE clause. Also called nested query or internal query, subqueries can also restrict the data being retrieved.

Creating subqueries are more complex than creating simple queries. You have to use essential key words such as, SELECT, DELETE, UPDATE, INSERT and the operators such as, BETWEEN (used only WITHIN a subquery and not WITH a subquery), IN, =, < =, > =, >, <, < >, and similar symbols.

In composing a subquery, you have to remember these pointers.

- It must be enclosed with an open and close parenthesis.
- It can be used in several ways.
- It is recommended that a subquery can run on its own.
- It can ascribe column values for files.
- It can be found anywhere in the main query. You can identify it because it is enclosed in parentheses.
- If it displays more than one row in response to an SQL command, this can only be accepted when there are multiple value operators. Example is the IN operator.
- In a subquery, the GROUP BY is used instead of the ORDER BY, which is used in the main statement or query.
- When creating subqueries, do not enclose it immediately in a set function.
- To create subqueries, it is easier to start with a FROM statement.
- Subqueries should also have names for easy identification.
- When using the SELECT key word, only one column should be included in your subquery. The exception is when the main query has selected many columns for comparison.
- Values that refer to a National Character Large Object (NCLOB), Binary Large Object (BLOB), Character Large Object (CLOB)

and an Array, which is part of a collection data in variable sizes, should NOT be included in your SELECT list.

## **Working with the Queries**

When you do set up the query that you would like to use, you will find that you are basically sending out an inquiry to the database that you already set up. You will find that there are a few methods to do this, but the SELECT command is going to be one of the best options to make this happen, and can instantly bring back the information that we need from there, based on our search.

For example, if you are working with a table that is going to hold onto all of the products that you offer for sale, then you would be able to use the command of SELECT in order to find the best selling products or ones that will meet another criterion that you have at that time. The request is going to be good on any of the information on the product that is stored in the database, and you will see that this is done pretty normally when we are talking about work in a relational database.

## **Working with the SELECT Command**

Any time that you have a plan to go through and query your database, you will find that the command of SELECT is going to be the best option to make this happen. This command is important because it is going to be in charge of starting and then executing the queries that you would like to send out. In many cases, you will have to add something to the statement as just sending out SELECT is not going to help us to get some of the results that you want. You can choose the product that you would like to find along with the command, or even work with some of the features that show up as well.

Whenever you work with the SELECT command on one of your databases inside of the SQL language, you will find that there are four main keywords that we are able to focus on. These are going to be known as the four classes that we need to have present in order to make sure that we are able to complete the command that we want and see some good results. These four commands are going to include:

- **SELECT**—this command will be combined with the FROM command in order to obtain the necessary data in a format that is readable and organized. You will use this to help determine the

data that is going to show up. The SELECT clause is going to introduce the columns that you would like to see out of the search results and then you can use the FROM in order to find the exact point that you need.

- FROM—the SELECT and the FROM commands often go together. It is mandatory because it takes your search from everything in the database, down to just the things that you would like. You will need to have at least one FROM clause for this to work. A good syntax that would use both the SELECT and the FROM properly includes:
- WHERE—this is what you will use when there are multiple conditions within the clause. For example, it is the element in the query that will display the selective data after the user puts in the information that they want to find. If you are using this feature, the right conditions to have along with it are the AND OR operators. The syntax that you should use for the WHERE command includes:

```
SELEC [ * | ALL | DISTINCT COLUMN1, COLUMN2 ]  
FROM TABLE1 [ , TABLE2];  
WHERE [ CONDITION1 | EXPRESSION 1 ]  
[ AND CONDITION2 | EXPRESSION 2 ]
```

- ORDER BY—you are able to use this clause in order to arrange the output of your query. The server will be able to decide the order and the format that the different information comes up for the user after they do their basic query. The default for this query is going to be organizing the output going from A to Z, but you can make changes that you would like. The syntax that you can use for this will be the same as the one above, but add in the following line at the end:

```
ORDER BY COLUMN 1 | INTEGER [ ASC/DESC ]
```

You will quickly see that all of these are helpful and you can easily use them instead of the SELECT command if you would like. They can sometimes pull out the information that you need from the database you are working in a

more efficient manner than you will see with just the SELECT command. But there are going to be many times when you will find that the SELECT command will be plenty to help you get things done when it is time to search your database as well.

## **A Look at Case Sensitivity**

Unlike some of the other coding languages that are out there and that you may be tempted to use on your database searches, you may find that the case sensitivity in SQL is not going to be as important as it is in some of those other ones. You are able to use uppercase or lowercase words as you would like, and you can use either typing of the word and still get the part that you need out of the database. It is even possible for us to go through and enter in some clauses and statements in uppercase or lowercase, without having to worry too much about how these commands are going to work for our needs.

However, there are a few exceptions to this, which means there are going to be times when we need to worry about the case sensitivity that is going to show up in this language a bit more than we may want to. One of the main times for this is when we are looking at the data objects. For the most part, the data that you are storing should be done with uppercase letters. This is going to be helpful because it ensures that there is some consistency in the work that you are doing and can make it easier for us to get the results that we want.

For example, you could run into some issues down the road if one of the users is going through the database and typing in JOHN, but then the next person is typing in John, and then the third person is going through and typing in john to get the results. If you make sure that there is some consistency present, you will find that it is easier for all of the users to get the information that they want, and then you can make sure that you are able to provide the relevant information back when it is all done.

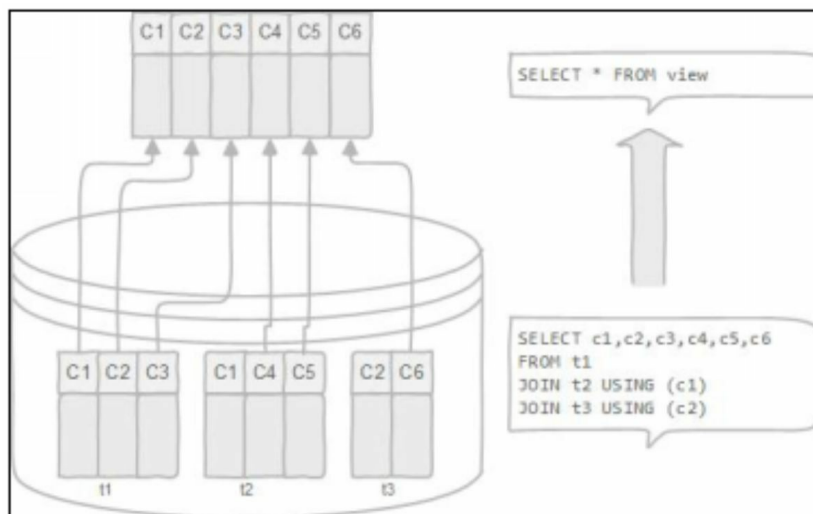
In this case, working with letters in uppercase is often one of the easiest ways to work with this because it is going to make it easier and the user is going to see that this is the norm in order options as well. If you choose to not go with uppercase in this, then you should try to find some other method that is going to keep the consistency that you are looking for during the whole thing. This allows the user a chance to figure out what you are doing, and will help them

to find what they need with what is inside of their queries.

## Chapter 11: SQL Views and Transactions

A Database View in SQL is defined as a “virtual or logical table” described as the SELECT statements containing join function. As a Database View is like a table in the database consisting of rows and columns, you will be able to run queries on it easily. Many DBMSs, including MySQL, enable users to modify information in the existing tables using Database View by meeting certain prerequisites, as shown in the picture below.

A SQL database View can be deemed dynamic since there is no connection between the SQL View to the physical system. The database system will store SQL Views in the form on SELECT statements using JOIN clause. When the information in the table is modified, the SQL View will also reflect that modification.



### Pros of Using SQL View

- A SQL View enables simplification of complicated queries: a SQL View is characterized by a SQL statement which is associated with multiple tables. To conceal the complexity of the underlying tables from the end-users and external apps, SQL View is extremely helpful. You only need to use straightforward SQL statements instead of complicated ones with multiple JOIN clauses using the SQL View.
- A SQL View enables restricted access to information depending on the user requirements. Perhaps you would not like all users to be able to query a subset of confidential information. In such cases SQL



View can be used to expose non-sensitive information to a targeted set of users selectively.

- The SQL View offers an additional layer of safety. Data security is a key component of every DBMS. It ensures additional security for the DBMS. It enables generation of a read only view to display read only information for targeted users. In read only view, users are able only to retrieve data and are not allowed to update any information.
- The SQL View is used to enable computed columns. The table in the database is not capable of containing computed columns but a SQL View can easily contain computed column. Assume in the OrderDetails table there is the quantity Order column for the amount of products ordered and priceEach column for the price of each item. But the OrderDetails table cannot contain a calculated column storing total sales for every product from the order. If it could, the database schema may have never been a good design. In such a situation, to display the calculated outcome, you could generate a computed column called total, which would be a product of quantityOrder and priceEach columns. When querying information from the SQL View, the calculated column information will be calculated on the fly.
- A SQL View allows for backward compatibility. Assume that we have a central database that is being used by multiple applications. Out of the blue you have been tasked to redesign the database accommodating the new business needs. As you delete some tables and create new ones, you would not want other applications to be affected by these modifications. You could generate SQL Views in such situations, using the identical schematic of the legacy tables that you are planning to delete.

## **Cons of Using SQL View**

In addition to the pros listed above, the use of SQL View may have certain disadvantages such as:

- Performance: Executing queries against SQL View could be slow, particularly if it is generated from another SQL View.
- Table dependencies: Since a SQL View is created from the underlying tables of the database. Anytime the tables structure

connected with SQL View is modified, you also need to modify the SQL View.

## **Create SQL View with JOIN Clause**

The MySQL database allows you to create SQL View using JOIN clause. For instance, the query below with the INNER JOIN clause can be used to create a view containing order no, client name, and totl sale per order:

```
CREATE VIEW clientOrdrs AS
SELECT
x.orderNo,
clientName,
SUM(qtyOrdrd * pricEch) total
FROM
orderDetail d
INNER JOIN
order r ON r.orderNo = x.orderNo
INNER JOIN
clients y ON y.clientNo = x.clientNo
GROUP BY x.orderNo
ORDER BY total DESC;
```

By executing the syntax below, the desired data can be retrieved from the client order view:

```
SELECT *FROM
clientOrder;
```

The result set is displayed in the picture below:

orderNumber	customerName	total
10165	Dragon Souvenirs, Ltd.	67392.84
10287	Vida Sport, Ltd	61402
10310	Tome Spezialitäten, Ltd	61234.66
10212	Euro+ Shopping Channel	59830.54
10207	Diecast Collectables	59265.14
10127	Muscle Machine Inc	58841.35
10204	Muscle Machine Inc	58793.53
10126	Comde Auto Replicas, Ltd	57131.93

## Create SQL View with a Subquery

The query below can be used to generate a SQL View with a subquery, containing products with purchase prices greater than the average product prices.

```
CREATE VIEW abvAvgPdcts AS
```

```
SELECT
```

```
pdctCode, pdctName, buyPric
```

```
FROM
```

```
pdcts
```

```
WHERE
```

```
buyPric >
```

```
(SELECT
```

```
AVG(buyPric)
```

```
FROM
```

```
pdcts)
```

```
ORDER BY buyPric DESC;
```

The query to extract data from the aboveAvgPdcts view is even more straightforward, as shown in the picture below:

```
SELECT
```

```
    *
```

```
FROM
```

```
abvAvgPdcts;
```

The result set is displayed in the picture below:

	productCode	productName	buyPrice
▶	S10_4962	1962 LanciaA Delta 16V	103.42
	S18_2238	1998 Chrysler Plymouth Prowler	101.51
	S10_1949	1952 Alpine Renault 1300	98.58
	S24_3856	1956 Porsche 356A Coupe	98.3
	S12_1108	2001 Ferrari Enzo	95.59
	S12_1099	1968 Ford Mustang	95.34
	S18_1984	1995 Honda Civic	93.89
	S18_4027	1970 Triumph Spitfire	91.92

## Check if an Existing View is Updatable

By running a query against the `is_updtble` column from the view in the `info_schema` database, you should verify whether a view in the database is updatable or not.

You can use the query below to display all the views from the `classicmodeldb` and check which for views that can be updated:

```
SELECT
tab_name,
is_updtble
FROM
info_schema.view
WHERE
tab_schema = 'classicmodel';
```

The result set is displayed in the picture below:

	table_name	is_updatable
▶	aboveavgproducts	YES
	customerorders	NO
	officeinfo	YES
	saleperorder	NO

## Dropping Rows Using SQL View

To understand this concept, execute the syntax below first to create a table called `itms`, use the `INSERT` statements to add records into this table and then use the `CREATE` clause to generate a view containing items with prices higher than 701.

--- creating new tbl called itms

```
CREATE TABLE itms (  
    identity INT AUTO_INCREMENT PRIMARY KEY,  
    nam VARCHAR (110) NOT NULL,  
    pric DECIMAL (10 , 1 ) NOT NULL  
)
```

-- adding records into itmstbl

```
INSERT INTO itms (nam,pric)  
VALUES ('Comp', 600.54), ('Laptop', 799.99),('Tablet', 699.50) ;
```

--- creating views based on itmstbl

```
CREATE VIEW LxryItms AS  
SELECT
```

\*

FROM

itms

WHERE

pric > 899;

--- retrieve records from the LxryItms view

```
SELECT
```

\*

FROM

LxryItms;

The result set is displayed in the picture below:

	id	name	price
▶	1	Laptop	700.56
	3	iPad	700.50

Now, using the DELETE clause record with identity value 3 can be dropped.  
DELETE FROM LxryItms

WHERE

id = 3;

After you run the query above, you will receive a message stating 1 row(s) affected.

Now, to verify the data with the view use the query below:

SELECT

\*

FROM

LxryItms;

The result set is displayed in the picture below:

	id	name	price
▶	1	Laptop	700.56

Finally, use the syntax below to retrieve desired records from the underlying table to confirm that the “DELETE” statement in fact removed the record:

SELECT

\*

FROM

itms;

The result set is displayed in the picture below, which confirms that the record with identity value 3 has been deleted from the items table:

	id	name	price
▶	1	Laptop	700.56
	2	Desktop	699.99

## Modification of SQL View

In MySQL, you can use ALTER VIEW and CREATE OR REPLACE VIEW statements to make changes to views that have already been created.

### Using ALTER VIEW Statement

The syntax for ALTER VIEW works a lot like the CREATE VIEW statement that you learned earlier, the only difference being that the ALTER keyword is used instead of the CREATE keyword, as shown below:

```
ALTER
[ALGRTHM = {MERG | TEMPTBL | UNDEFND}]
VIEW [db_nam]. [vw_nam]
AS
[SELECT statemnt]
```

The query below will change the organization view by incorporating the email column in the table:

```
ALTER VIEW org
AS
SELECT CONCAT (x.lastname,x.firstname) AS Emplie,
x.emailAS emplieEmail,
CONCAT(y.lstname, y.fstname) AS Mgr
FROM emplyes AS x
INNER JOIN emplyes AS y
ON y.emplyeNo = x.ReprtsTo
ORDER BY Mgr;
```

You may run the code below against the org view to verify the modification:

```
SELECT
*
FROM
Org;
```

The result set is displayed in the picture below:

	Employee	employeeEmail	Manager
►	JonesBarry	bjones@classicmodelcars.com	BondurGerard
	HernandezGerard	ghernande@classicmodelcars.com	BondurGerard
	BottLarry	lbott@classicmodelcars.com	BondurGerard
	GerardMartin	mgerard@classicmodelcars.com	BondurGerard
	BondurLoui	lbondur@classicmodelcars.com	BondurGerard
	CastilloPamela	pcastillo@classicmodelcars.com	BondurGerard
	VanaufGeorge	gvanauf@classicmodelcars.com	BowAnthony

## Using CREATE OR REPLACE VIEW Statement

These statements can be used to replace or generate a SQL View that already exists in the database. For all existing views, MySQL will easily modify the view but if the view is non-existent, it will create a new view based on the query.

The syntax below can be used to generate the contacts view on the basis of the employees table:

```
CREATE OR REPLACE VIEW cntcts AS
```

```
    SELECT
```

```
    firstName, lstName, extnsn, eml
```

```
    FROM
```

```
    emplyes;
```

The result set is displayed in the picture below:

	firstName	lastName	extension	email
►	Diane	Murphy	x5800	dmurphy@classicmodelcars.com
	Mary	Patterson	x4611	mpatterso@classicmodelcars.com
	Jeff	Firrelli	x9273	jfirrelli@classicmodelcars.com
	William	Patterson	x4871	wpatterson@classicmodelcars.com
	Gerard	Bondur	x5408	gbondur@classicmodelcars.com
	Anthony	Bow	x5428	abow@classicmodelcars.com
	Leslie	Jennings	x3291	ljennings@classicmodelcars.com

Now, assume that you would like to insert the jobtitl column to the cntcts view. You can accomplish this with the syntax below:

```
CREATE OR REPLACE VIEW cntcts AS
```

```
    SELECT
```



firstName, lastName, extension, email, jobtitle

FROM

employees;

The result set is displayed in the picture below:

	firstName	lastName	extension	email	jobtitle
▶	Diane	Murphy	x5800	dmurphy@classicmodelcars.com	President
	Mary	Patterson	x4611	mpatterso@classicmodelcars.com	VP Sales
	Jeff	Firrelli	x9273	jfirrelli@classicmodelcars.com	VP Marketing
	William	Patterson	x4871	wpatterson@classicmodelcars.com	Sales Manager (APAC)
	Gerard	Bondur	x5408	gbondur@classicmodelcars.com	Sale Manager (EMEA)
	Anthony	Bow	x5428	abow@classicmodelcars.com	Sales Manager (NA)
	Leslie	Jennings	x3291	ljennings@classicmodelcars.com	Sales Rep

## Dropping a SQL View

The DROP VIEW statement can be utilized to delete an existing view from the database, using the syntax below:

```
DROP VIEW [IF EXISTS] [db_name]. [vw_name]
```

The "IF EXISTS" clause is not mandatory in the statement above and is used to determine if the view already exists in the database. It prevents you from mistakenly removing a view that does not exist in the database.

You may, for instance, use the DROP VIEW statement as shown in the syntax below to delete the organization view:

```
DROP VIEW IF EXISTS org;
```

## SQL TRANSACTIONS

Any actions that are executed on a database are called as transactions. These are actions that are executed logically, either manually by a user or automatically using by the database program.

Or simply put, they are the spread of one or more database modifications. For instance, every time you create a row, update a row, or delete a row a transaction is being executed on that table. To maintain data integrity and address database errors, it is essential to regulate these transactions.

Basically, to execute a transaction, you must group several SQL queries and run them at the same time.

## **Conclusion**

SQL in full is the Structured Query Language and is a kind of ANSI computer language that has been specially designed to access, manipulate, and update database systems. SQL has different uses; the most significant of them is managing data in database systems that can store data in table forms. Additionally, SQL statements have been used regularly in updating and retrieving data from databases.

What we have thought since childhood is the best to learn a new concept is by practicing it. It is no exception; you have to do various equations related to SQL as a way of learning about them. I hope you can apply what you have learned from this book to be successful in your future projects.

## **Projects in SQL Programming**

Below are random projects that you can encounter in SQL programming. They have been picked and randomly, and I believe if you practice them out, you will be better equipped in handling such situations. Before we list the projects, we first should get to understand the differences between exercises and projects. Well, logically, a task is more of a quick test you can do without a lot of complications: it is less complicated compared to projects. On the other hand, projects are a little more complicated and sophisticated. It requires advanced skills as well as data research to do it.

Having learned that, let's discuss the potential projects you can encounter in SQL programming.

### **i) Interviews**

When making software for a particular company, you will need knowledge of an existing system that belongs to that company. For such purposes, you will have to carry out interviews for some individuals working in that company and collect critical information. You will have to do interviews on people that are aware of that software. Such people could be working as hostel wardens or trainers.

### **ii) Discussions (Groups)**

They can be a kind of group discussion that has occurred between employees of the company you are working on. For a start, a good number of ideas might appear clustered together or filled by concepts that already exist. Such ideologies might be brought on board by programmers.

Additionally, it can be done through online observation. It is a procedure of obtaining more essential details about the existing software or web apps from the web. The primary purpose of this project is getting as close as possible to the system. SQL programming plays a critical role in ensuring the systems are up and running as recommended.

## **Application of SQL**

The self-variable option lets you carry out the joining process on the same table, saving you the time you spend organizing the final table. There are, however, a few situations where this can be a good option. Imagine the chart you created earlier has the columns consisting of country and continent.

When faced with a task of listing countries located on the same continent, a clearly outlined set below should give you a glimpse of the results expected. SQL variable can further be subdivided into three different types: the left join, the right join as well as the full outer join. The outer join primary role is returning all the identified rows from a single table, and once the joining target is archived, it includes the columns from another table. Outer joins are different from inner joins in the essence that an inner join cannot involve the unmatched rows in the final set of results.

When using an order entry, for instance, you may be faced with situations where it is inevitable to list every employee regardless of the location, they put customer orders. In such a case, this kind of joins is beneficial. When you opt to use this kind of join, all employees, including those that have been given marching orders, will be included in the final result.

This is a kind of outer join that is responsible for returning each row from the first left side of the table and those row that match from the right side of the table. In case there are no matches on the right side, left join returns a null value for each of those columns. A type of outer join has the task of returning each row from the right side of the table and merging with the other side (left) of the table. Again, if there aren't any values for those digits in the column, the join returns null values for them.

It has the task of returning rows from their initial location in the inner join, and in case there is no match found, this join returns null results for those tables.

This is a kind of variable that is essentially a product of Cartesian elements that have been expressed in the SQL set up. Picture this; you require a whole

set of combinations available between both tables or even in just one meal. You will have to use the cross join to achieve that technique. To help you understand this join better, you can go back to the two tables we created at the beginning of the article. Look at both the columns and try to compare the impact each one of them has to the final result. The cross join plays an essential in ensuring accuracy during merging. You ought to note that there are apparent differences between cross joins and outer joins despite the fact that the description makes them almost look similar.

Similarly, MySQL system has a slot that allows you to announce more than one set of variables that has a common type of data. Again, most of the technicians have had issues with how this command relays information to related databases. In the various methods of storing variances and variables, this one has proven to be more secure than others. Consequently, it has been known to be the most popular of them all.

Variables can be applied in mathematical expressions, for example, adding values altogether or combining and holding texts. This can be used as a section of the general information. For your information, variables are also applied in storing information so as one can participate in a kind of calculations. Additionally, variables can be part of the parameters and are used in procedural assessments. This is two in one method that not only lets you declare a variable but also setting it up with values that have a similar data type. Going back to the examples we gave earlier; we can affirm that varchar is a kind of data that lets you sustain more than one kind of character in just a single string.

Up to this point, you should be in a position to understand the kind of SQL Exercises and Programs as well as the various types in existence. This will not only let you be in an excellent place to tackle errors in case they occur and prevent them from happening as well. When Mark Suaman, a renown data scientist and a graduate of Havard University, first used varchar, he recommended it for being efficient and accurate. He rated it among the best types of data set in the market today. It does not have an allocation for potential occurrences of errors. It is hard to interfere with such a highly secure kind of data type.

## **Applications of SQL**

Since its introduction in the computing world, SQL has played a significant

role in revolutionizing data storage in a systematic manner as well as direct retrieval. As the digital world continues to grow steadily, the amount of data stored quickly escalates, making organizations and personal data piling up. Therefore, SQL acts as a platform where these data are stored while offering direct access without the emergence of limitations. As such, SQL is used in different sectors, including telecommunication, finance, trade, manufacturing, institutional, and transport. Its presence primarily deals with data but also accompanies other significant benefits on its application.

### **Data Integration**

Across the sectors mentioned above, one of its main applications of SQL is the creation of data integration scripts commonly done by administrators and developers. SQL databases comprise of several tables which may contain different data. When these data are integrated, it creates a new experience essential for the provision of meaningful information, therefore, increasing productivity. Data integration scripts are crucial in any given organization, including the government, as it offers trusted data which can be utilized to promote the achievement of the set goals.

### **Analytical Queries**

Data analysts regularly utilize Structured Query Language to smoothen their operations more so when establishing and executing queries. SQL comprises multiple tables that consist of different datasets. When these data are combined, it brings out more comprehensive information critical for any individual or organization. The same is also applicable for data analysts as they use a similar aspect. As they use an analytical query structure, queries, and tables from SQL are fed into the structure to deliver crucial results from varying sources. In this case, data analysts can readily acquire different queries and customize them to have a more comprehensive data to depend on as solutions.

### **Data Retrieval**

This is another important application of SQL to retrieve data from different subsets within a database with big data. This is essential in financial sectors and analytics as to the use of numerical values typically consists of mixed statistical data. The most commonly used SQL elements are create, select, delete, and update, among others. The technique is made possible when the user quickly can search the database and acquire the needed data as SQL sieves the information to bring out the desired data. In some cases, the

language may deliver similar or related data when the required data is missing. This is crucial as one can compare the results as well as make changes where the need arises.