SQL FOR BEGINNERS

The Simplified Guide to Managing, Analyzing Data With SQL

DAN PARK

SQL FOR BEGINNERS

The Simplified Guide to Managing, Analyzing Data With SQL

DAN PARK

SQL for Beginners

The Simplified Guide to Managing, Analyzing Data With SQL

By

DAN PARK

Table of Contents

Introduction

Learning SQL Programming for Beginners

- Brief History of SQL
- Design and Syntax
- Features of SQL
- Common SQL Database Management Systems
- Key SQL Statements SELECT

Creating a Database in SQL Server

The SQL Structure

- The SQL Structure
- SQL Fundamental Features

Database administration

- Setting up a Protection Plan in SQL Server
- Setting up Database Mail
- SQL Server Agent

Structure of the SELECT statement

- The SELECT Clause
- The TOP Clause
- The CASE Conditional Expression

SQL Data Types

- Data Definition
- SQL Data Types

Cursors: Read Only, Forward Only, Fast Forward, Static, Scroll

- Dynamic, Optimistic

Preparation

- Database Architecture
- Database Versions
- Connecting

ODBC Admin Main Screen
SQL Server Data Source
Working with Filters
WHERE Clause
HAVING Clause
Evaluating a Condition
Usage of Parentheses
The NOT Operator
Sequences
SQL Subqueries
Subqueries with SELECT Statement
Subqueries with INSERT Statement
Subqueries with UPDATE Statement
Subqueries with DELETE Statement
Database Components
Database Tables
Schemas
Columns
Rows and NULL values
Primary Keys
Foreign Keys
Constraints
Views
Stored Procedures
Triggers
Deadlocks in SQL
Managing Database Objects
What is a Database Object?
Altering the Elements of a Database Table
Modifying Fields/Columns

How to Drop Tables
The Integrity Constraints
How to Drop Constraints
Conclusion

Introduction

SQL for Beginners guide.

When most people pay attention programming, they think it is something complicated which demand adequate know-how to learn. That isn't the case; however, gaining knowledge of SQL programming is one of the short methods to gain sufficient experience in computer languages starting doorways to learning other standards of programming.

The following chapters will discuss important factors of a way to study SQL programming greater so on a character with constrained or no experience in the location of laptop study. The eBook will start with a brief records of SQL and allow you to understand the layout and syntaxes used. As understanding the basics before everything is vital, you may learn about the characteristic capabilities of SQL as well as the database management systems commonly used today.

That said, the chapters will then spotlight the important SQL statements and their examples followed by means of the syntaxes and capabilities used to query records in tables. As database systems usually have multiple tables, there is a want to learn how to question this records to in shape a given criterion. As such, the e book, therefore, discusses the filters paintings in SQL programming, especially in tables within beneath database systems.

There are lots of books on this concern at the market, thanks again for selecting this one! Every effort become made to make certain it is complete of as a great deal useful records as possible, please revel in it!

Learning SQL Programming for Beginners

When one mentions approximately SQL, which stands for Structured Query Language, in computing, a few learners or those with little knowledge approximately the device will tend to suppose that it's far an in depth field of study. As such, many will discover it difficult, and therefore, fail to engage in the creation of databases. However, that isn't always the case as SQL programming language is well-known software with an honest concept, not like other pc programming tools. Besides, it's miles amongst the most original programming languages to examine in computing, acting as a doorway to interact in different pc languages, for instance, JavaScript and Python.

As programming is typically extensive, the identical may be stated with SQL because it also takes one-of-a-kind pathways for the successful improvement of an operational database gadget. Some might also time period SQL as a laptop programming language as it takes the shape of commands, which consists of commands for the machine to engage in a particular action like maximum programming tools. On the other hand, others talk over with it as a records management gadget because it entails the introduction of a couple of tables organized under one or extra databases critical for the garage of statistics. Therefore, scientists and developers finish that SQL is a standard pc language which helps in the communique between exceptional database systems.

Structured Query Language or SQL can, therefore, be described as a standard pc programming language that enables programmers to paintings with data from extraordinary sources. Despite the use of one of a kind markup within the database, SQL primarily involves tables which usually constitute the general gadget. Tables commonly consist of columns and rows which constitute the sort of statistics fed, and the number can also vary as well. For example, a library can be used as a conventional storage and control unit, which may be fast become a database system. When a table is created, the information fed into each field may additionally constitute differently. That is, the e-book ID might also be referred to as an integer in databases, even as the yr. is specified as date and title, author, genre, and language supplied specific char values.

Today, nearly all corporations around the arena have was digital coping with of facts, which keeps developing daily. Ranging from small on line business shops to big groups along with Fortune 500 utilize databases as a way to store and manage this information. This way, SQL has gained reputation over the years with call for greater database administrators increasing every passing day. Therefore, turning into an expert or developer in SQL may additionally play a widespread function in making you very lucrative. In maximum cases, you ought to have ok understanding in SQL while you want to make a career in web and app development. However, taking a route in SQL programming allows you to start your way into attractive in more complex laptop languages.

Brief History of SQL

The first SQL model was developed by Ted Code, who brought the potential of communication between specific storage units. His concept become later received via Donald Chamberlin and Raymond Boyce and evolved the version within the 1970s. However, the model at the time become referred to as SEQUEL (Structured English Query Language) specifically designed to help in information garage and retrieval. More changes were made over time in IBM, with the primary being accomplished in a laboratory in San Jose, California, United States, where the primary subscript notation became delivered. After trying out the effectiveness of the software program in customer take a look at websites, SQL turned into then developed for use commercially after its usefulness and practicality determined.

Initially, the call SEQUEL became changed to SQL because it resembled a dynamic engineering business enterprise within the United Kingdom. Over the years, however, exclusive groups commenced the development of database software program, for instance, Rational Software, which commenced inside the overdue 1970s. Due to its advantages in coping with statistics in tables form, control, and retrieval, one-of-a-kind sectors utilized the usual Database Language SQL. The recognition of the software program rose quick, and the developing employer had to make modifications in keeping with customer enjoy and feedbacks. As such, variations of SQL commenced trickling with the primary delivered in 1989 with others following in 1992, 1996, 1999, 2003, 2006, 2008, 2011, and 2012.

Design and Syntax

When SQL is as compared to different comparable concepts, it substantially deviates, in particular in its theoretical foundation. The layout is unique and includes tables and queries with datasets written in rows and columns. Other tuples along with tuple calculus normally consist of tables which are set, which is not the case in SQL. Due to its uniqueness in the sector, many argue that SQL need to be altered to create a language that permits for a right away return to the original format however with limited research. As a programming tool, SQL comprises exceptional syntaxes classified into clauses, expressions, predicates, queries, statements, and insignificant whitespaces.

Clauses usually encompass factors of statements and queries which from time to time may not be present while insignificant whitespaces do now not include them but allow for code readability in SQL. Expressions are essential inside the manufacturing of scalar values or rows and columns utilized in tables. Queries help in data retrieval while following particular set parameters. Statements manage the overall impact of the facts, together with connections, operations, schemata, and diagnostics. Predicates, on the opposite hand, are crucial within the specification of conditions evaluated by using SQL in distinctive functions, which includes three-valued good judgment and Boolean truth value.

Features of SQL

- Increased overall performance for excessive transactional, utilization of database systems, and handling of substantial operations
- High compatibility with different databases inclusive of Microsoft SQL Server, MySQL, SAP Adaptive Server, and MS Access amongst others
- Highly stable and brief get entry to tables and views for directors, therefore, retaining the statistics secure from threats It gives stable transactional assistance as it handles a massive amount of facts and helps within the control of several transactions
- Easy to scale in addition to flexible mainly while creating, enhancing and deleting of tables
- Straightforward management systems such as SELECT,
- UPDATE, DELETE and CREATE
- Used across more than one areas henceforth imparting a complete application improvement benefits

- It is one in every of the nice open-supply programming languages whilst it comes to dealing with information the usage of relational database systems

Common SQL Database Management Systems

Database control systems are critical as it enables the interaction among the builders and database programs. These systems come with templates, in addition to builders and other equipment which enable builders to maneuver thru databases and tables readily. As such, it simplifies programmers' life, for example, in database gadget cleaning. The rating of a database gadget is normally carried out by using the frequency of seek engines, professional networks profile, and social network relevance. Conventional SQL database structures include:

Oracle database - it's far used industrially in both warehouse and online processing and the main SQL database device within the world

MySQL - in contrast to oracle, MySQL is by and large utilized in small organizations as well as startups and unfastened for personal use. It is also one of the leading in open supply database machine

Microsoft SQL Server - this is a Microsoft SQL version database device designed to run throughout Windows operating systems. It is generally utilized in web servers and accommodates of a widespread user base PostgreSQL - that is also every other unfastened SQL database control system used by generation small groups and startups. It is compatible with all operating systems including macOS, Windows, and Linux and comes with enormous change tools, not like other database structures.

Key SQL Statements SELECT

SELECT is one in every of the vital statements in SQL as it facilitates the method of database querying and records retrieval. SQL, like other computer languages, involves more than two approaches when searching for to gain particular processes. Similarly, when you need to go approximately the SELECT assertion, you can either use the commands or maneuver in the database management studio. When writing the command, one-of-a-kind clauses are involved with some, consisting of quotes inside the technique. Besides, the command line is often observed by way of different statements together with FROM, LIKE, and WHERE, which facilitate the manner of

building the function for precise instructions. SELECT is also vital while dealing with particular statistics as properly as records with similarities inside tables under one or more databases.

The SELECT declaration takes the format select * from (tabl_name), Which without problems selects records from the specified desk. Besides, the choice statement can also either consist of situations or not depending at the user set parameters. For example, if you want to choose the file of students with a score of more than 5 with a table name ‘scholar scores,' the format used will be ‘SELECT (table_name) FROM student scores WHERE score>5. In this example, students with a score of extra than five will be selected even as deselecting those with a lower price.

INSERT

Creating a brand new database and desk does not always mean which you are thru with SQL functions in facts garage and retrieval. That is, you are required to feed the information into the table within the fields that intersect rows and columns. SQL permits customers to input values without delay into the desk as nicely as the use of codes to give commands that provide direct facts entry. The announcement used in this situation is INSERT INTO or INSERT VALUE, where the fee is inserted into the system with minimal difficulty. For example, when you upload values inside the table, you can mistake a few values, therefore, compromising the general information. Thus, the use of INSERT statements allows for short and direct data access techniques essential for avoiding common errors springing up from information entry errors.

UPDATE

Data updates are regularly made in the machine as new information arises every day greater so in organizations. In some cases, administrators may also pick to update the records with respect to sure conditions. All the equal, there exists a factor where facts have to be altered. Therefore, SQL accompanies an assertion UPDATE, which enables you to with no trouble get access for your tables and make the necessary adjustments. The function is implemented in already present tables and may additionally impact one or extra rows or columns without causing adjustments to accidental fields. You can also use the command option, on the way to henceforth update the facts as such, preserving it updated as needed.

DELETE

Deleting is regularly used for the elimination of unwanted or other irrelevant facts within the distinctive sections in a computer machine. The equal is same in SQL, as some information may be removed to allow area for other crucial information or dispose of preceding records. Therefore, SQL uses the statement DELETE to remove individual documents in either particular fields or the general data without deleting the desk. In some cases, a few users may use the DELETE feature to replace, modify, and make a correction while errors are made during statistics access. This is done by way of following positive criterions which are set by way of conditions, so when the command is run, it deletes the meant sections specified.

When you operate the DELETE statement, the function used is DELETE * FROM (table_name) WHERE (column_name). conditions used on this assertion allow the consumer to quickly dispose of certain information at the same time as following the necessary conditions for the elimination of the intended information. For example, if you want to remove or delete a character student call from the table, you have to realize his or her call first. If the call is Michael with the table name Records, the layout used will be DELETE * FROM Records WHERE StudentName = ‘Michael.' The statement will ensure that the information deleted will be of a scholar mentioned Michael alone without affecting other data. More so, the user may additionally specify different names also if he or she wants to do away with other information of the students from the table.

Creating a Database in SQL Server

SQL databases are among the maximum used databases throughout the world. This is due to a number of reasons, for instance it is very easy to create. What you want is a graphical consumer interface application that comes freely like a SQL Server Management. With that during place, creating a database is straightforward and you may begin getting into your facts very quickly at all. Here is how:

Start by installing the software (SQL Server Management Studio) on your computer

This is software program this is freely available for Microsoft. It will permit you to advantage get right of entry to and additionally to work along with your SQL server from a graphical interface apart from using a command line for the same. The software will additionally assist you to advantage get right of entry to a faraway request of an SQL server. If not this one, you will require a similar software program.

There are other interfaces that are to be had for other structures like Mac as an example SQuirreL SQL. Such interfaces may additionally differ however all of them work the same.

You also can create a database the usage of the tools available in command line.

Once the software program has been installed, start it up.

After the installation, you could now begin your application. You can be required to select if you need to connect with a sure server. If there may be a server already that is already set and working and you have all the permissions connect access it, just input its address and the authentication facts.

But if you want to build your very own nearby database, you will create the Database name and the type of authentication under the Windows Authentication.

Now locate your database folder

After a connection has been made to the server, whether it is a nearby connection or a remote one, the Object Explorer will now open on the left hand facet of your display. Right on the top a part of your Object Explorer

diagram, you may see the server that you are the use of to. If it has not been expanded, click at the "+" icon that is following it and it'll expand.

You can now create a fresh database

Spot the database folder and right click on it. Click on New database choice from the list to be able to come up. This will come up with a brand new window so one can allow you to arrange your database earlier than you begin growing it. First of all, you need to present your database a brand new and unique name, which will make it clean in order to discover it. The different settings may be left just the way they may be at default settings unless there's a critical change that you need to make. When you give your database a call, there are two different additional documents so as to be formed automatically, which might be log and records files. The information record will be the one with a purpose to host all of your facts in your database and the log record could be the one with the intention to track all of the adjustments that you may make at the database. When satisfied, you can hit OK so that you can create your database. Your newly created database will now appear in the prolonged database Folder, with a cylindrical icon, it may be smooth to spot it.

Start growing your table

You need to give you a shape where you'll begin storing your facts and this will be your table. With a desk, you can preserve all manner of records and records that you want stored inside the database. This is an important part before you could move on. To do this, you amplify the brand new database this is for your database folder and then right click on the desk’s icon to select a New Table option. Windows thereafter opens the whole lot else on your screen to will let you to work for your new table as plenty as you need.

It’s time for the primary key

Primary keys are very important, therefore it's miles crucial to let them be the first entry on the first column of your SQL table. They act as the ID variety or the best range that enables you quickly do not forget what you have positioned in file in that table. In order to create your number one keys, enter ID on the sphere that has the Column Name and enter INT into the field marked Data Type. As of the Allow Nulls, ensure that they're all unchecked. Now hit the important thing icon for your toolbar so that it will make this column your number one key. With this, you may no longer have null values

but if you want to have a null value as your most important access, you may check to Allow Nulls.

Scroll down the column residences to find the choice Identity Specification.

Expanding this option and setting it to a YES will ensure that the values on the ID column increases routinely on each access that you may make.

With this, all of your new entries can be successfully numbered within the right order.

It’s time to apprehend how tables are designed.

This is a critical part to be able to find it easy to input information in your database. With tables, you will get different columns or fields and each column denotes an aspect of each database entry that you may make. If you've got a database for human beings in an enterprise for instance, your may has a FirsName column access, LastName column entry, Address, Phone Number and such like entries.

The different columns

When all the fields of the Primary Key have been crammed in, different fields will routinely form below it. These can be the fields where all of your other data might be entered. You are now loose to enter records in the ones fields the manner you need to. The proper statistics type must be chosen although if you want to suit the statistics which you have filled in that column. nchar(#) Represents the sort of information that must be used for the text as an instance addresses, names amongst others. In the parenthesis can be quite a number which is the highest wide variety on the way to be allowed in that field. You can set the limit so as to allow the dimensions of your database to remain manageable. You can as an example use this format for the smartphone numbers for you to make it difficult for you to perform mathematical function at the numbers.

Int however represents information in complete numbers. This is the one that is used inside the discipline marked ID.

Decimal(x,y) will shop your numbers in a decimal format. The wide variety in the parenthesis will signify the overall wide variety of numerals and the opposite number of digits as a way to comply with the decimals respectively.

When all that is done, keep the table

First store the table then you could start entering data for your columns. To do this, click at the Save button in your toolbar, then enter the call for your table. It is important to have a completely unique and easy to recognize call for your table so that you may be able to tell what the desk is all about with out going through the facts in it. This can be very beneficial specifically as soon as you start using large databases that have such a lot of tables.

The SQL Structure

The SQL Structure

In this chapter you'll examine the fundamental functions of the SQL language and an overview of its programming aspect. In addition, you will be offered with a step-through-step instruction on wherein and how to down load SQLite, a model of the SQL software program with a purpose to be used all in the course of the discussion of this guide.

SQL Fundamental Features

SQL is a flexible computer language that you may deploy in unique methods to talk with relational databases. This software has a few distinct capabilities that differentiates it from other programming applications. First and foremost, SQL is a nonprocedural language. Most laptop programs (e.G., C, C++ and Java) solve problems by means of following a sequence of commands that is known as a procedure. In this case, one precise operation is done after another until the required assignment has been accomplished. The drift of operation can either be a linear collection or a looping one, depending on what the programmer had detailed. This isn't the equal for SQL. In the use of this utility, you may just have to specify the output that you want, not how you need to generate the output. From the CUSTOMER TABLE, if you need to create a separate listing of contacts whose enterprise are placed in Texas then you have to retrieve the rows where the STATE column contains “TX” as its value. In writing the SQL command, you don’t have to indicate how the facts need to be retrieved. It is the number one function of the database management device to look at the database and determine a way to generate the results you wanted.

Learning the SQL syntax is like know-how the English language shape. Its command language, made out of a restrained variety of statements, plays three primary information functions - definition, manipulation and control. The SQL programming language also consists of reserved words that are handiest for use for specific purposes. Thus, you cannot use these words as names for variables, tables and columns; or in any different way other than their supposed use.

If you suspect that an SQL database is only a series of tables, then you are wrong. There are additional systems that want to be designated to hold the

integrity of your data, which includes schemas, domain names and constraints.

Schema – This is also referred to as the conceptual view or the whole logical view that defines the entire database shape and presents overall desk organization. Such schema is taken into consideration a metadata – saved in tables and a part of the database (just like tables that include regular statistics).

Domain – This specifies the set of all finite facts values you may keep in a selected table column or characteristic. For example, in our previous CUSTOMER TABLE the STATE column can best contain the values “TX”, “NY”, “CA” and “NV” if you handiest offer merchandise and services inside the states of Texas, New York, California and Nevada respectively. So these four kingdom abbreviations are the area of the STATE characteristic.

Constraint – Often disregarded however considered one of the essential database components, this sets down the regulations that pick out what statistics values a selected desk characteristic can include. Incorporating tight constraints assures that database users most effective enter valid facts into a particular column. Together with defined table characteristics, column constraints decide its area. Using the equal STATE column as an example with the given constraint of handiest the four values, if a database person enters “NJ” for New Jersey, then the entry will not be accepted. The system will no longer proceed until a valid value is entered for the STATE attribute, until the database shape needs to be updated due to sudden business changes.

ROLLBACK [WORK];

In the previous command line, the keyword WORK is optional.

SAVEPOINT – This announcement works with the ROLLBACK command, wherein it creates sections or factors within companies of transactions in which you'll be appearing the ROLLBACK command. Its syntax is:

SAVEPOINT SAVEPOINT_NAME;

SQLite Installation Instructions and Database Features

Before you begin overwhelming yourself with diverse database answers and SQL command lines, you need to decide first your cause why you are growing a database. This will further determine other database design considerations together with size, complexity, kind of device in which the

utility will run, garage medium and more. When you start thinking of your database requirements, you want to recognize as much as what degree of detail ought to be taken into consideration in your design. Too much detail will result to a very complicated design that further wastes time and effort, and even your pc's garage space. Too little will lead to a negative acting, corrupt and nugatory database. Once you are carried out with the layout phase, then you may decide which database software program you can download to begin your SQL experience.

For the sake of this guide discussion, SQLite, a simple software

program library, can be used as a starter database engine to layout, build and set up applications. A loose and stand-by myself database software program that is brief to download and clean to administer, SQLite was developed with the aid of Richard Hipp and his group of programmers. It is being designed so that it could be without difficulty configured and implemented, which does no longer require any client-server setup at all. Thus, SQLite is taken into consideration as certainly one of the most broadly used database software applications within the world.

Stated beneath are some of the major features of SQLite:

- Transactions are atomic, consistent, isolated and durable
- Compilation is simple and easy
- System crashes and power failures are supported
- Full SQL implementation with a stand-alone command line interface client
- Code footprint is significantly small
- Adaptable and adjustable to larger projects
- Self-contained with no external dependencies
- Portable and supports other platforms like Windows, Android, iOS, Mac, Solaris and more In the use of SQLite, you need to down load SQLite Studio as your database manager and editor. With its intuitive interface, this software program is very light yet rapid and powerful. You don't even want to put in it, just download, unpack and run the utility. Follow these easy steps in downloading SQLite Studio on a Windows 10 computer:
- Go to http://sqlitestudio.Pl/?Act=about. You ought to get the subsequent page:

_ Check the version of your computer's operating system then click the suitable link to start downloading the software.

After downloading the software program, go to the folder wherein the application changed into saved (commonly the Downloads Folder in Windows). Click at the Extract tab on top then pick the Extract all choice.

You will get the Extract Compressed (Zipped) Folders conversation box. Change the vacation spot folder to C:SQL then click the Extract button. This will be the folder where all your SQLite files could be saved.

_ Once all the files were extracted, you may have the SQLite Studio subfolder.

_ Find the software application named SQLite Studio inside the subfolder. To create a shortcut on your desktop (so you can q uick launch the software), right-click the filename, choose Send to alternative then pick out Desktop (create shortcut).

_ When you double-click on the SQLite Studio icon to your desktop,

_ You should get the subsequent screen:

Database administration

Once you have got your database up and walking with tables and queries it is up you to maintain the production database going for walks smoothly. The database will ought to be regularly checked out so as to ensure that it keeps to carry out as at the start intended. If a database is poorly maintained it is able to easily result in a website linked to it performing poorly or worse still bring about down time or even data loss. There is commonly a person precise to appearance after the database and their job is titled Database Administrator or DBA. However, it's commonly a non-DBA man or woman who needs help with the database.

There are some of one-of-a-kind tasks which you could carry out while sporting out maintenance which consist of the following:

- **Database Integrity** : When you test the integrity of the database you are running tests at the information to make certain that each the bodily and logical shape of the database is steady and accurate.
- **Index Reorganization** : Once you start to insert and delete statistics on your database there is going to be fragmentation (or a scattering) of indexes. Reorganizing the index will bring everything returned together again and growth speed.
- **Rebuild Index** : You don't should carry out an index reorganization, you may drop an index and then recreate them.
- **Database Backup** : One of the most vital tasks to carry out. There are some of specific ways in which you may returned up the database, these include: Full which backs up the database entirely, Differential which backs up the database since the last full backup and Transaction log which best backs up the transactional log.
- **Check Database Statistics** : You can take a look at the facts of the database which might be kept on queries. If you update the records, that may get out of date, you could assist resource the queries being run.
- **Data and Log File** : In general, ensure the records and log files are kept separate from each other. These files will grow whilst your database is getting used and its first-class to allocate them the correct size going forward (and not just permit them to grow).

Depending in your database a few obligations can be more useful than others.

Apart from database backup which in all likelihood obligatory if it’s in production you can pick out through the opposite duties relying on the nation of the database.

For example, need to the fragmentation of the database be under 30% then you may choose to carry out an index reorganization. However, if the database fragmentation is more than 30% then you should rebuild the index. You can rebuild the index on a weekly basis or more often if feasible.

You can run a protection plan on SQL Server thru its Server Agent relying on database requirements. It’s important to set the times right no longer while your application is anticipated to be busy. You can choose a time or you may run it when the server CPU is not busy. Choosing to run while the server isn't always busy is a more desired choice for large databases than selecting a selected time as there may be no assured time which the CPU can be idle. However, it's far typically handiest an issue if your software is quite large and has a lot of requests.

When you do rebuild the indexes, it is critical that you have the outcomes taken care of in tempdb. When the usage of tempdb the antique indexes are stored until new ones are added. Normally rebuilding the indexes makes use of the fixed area which the database become allocated. So, in case you run out of disk space then you definitely would now not be able to finish the rebuilding of indexes. It’s feasible to use the tempdb and now not need to growth the database disk size. The database preservation can be run each synchronous (wait for undertaking completion) or asynchronous (together) to speed things up however you ought to make certain that the duties run in the right order.

Setting up a Protection Plan in SQL Server

To installation a maintenance plan in SQL Server you first need to get the server to reveal superior options. This is accomplished by jogging the subsequent code in a new question in SQL Server:

```
sp_configure 'show advanced options', 1

GO

RECONFIGURE GO

sp_configure 'Agent XPs', 1
```

GO

RECONFIGURE

GO

SQL Server will now display the advanced alternatives. Left click the + icon to the left of Management which is on the left-hand side of SQL Server Management Studio. Now left click on Maintenance Plans and then right click on Maintenance Plans. Select New Maintenance Plan Wizard.

Enter the appropriate maintenance plan call and description. From here you could either run one or all obligations in a single plan and feature as many plans as you need. After you have given a name, pick out unmarried schedule and click on subsequent.

You will see a number of options which you can choose for your protection including: Check Database Integrity, Shrink Database, Reorganize Index, Rebuild Index, Update Statistics, Clean up History, Execute SQL Server

Agent Job, Back Up – full, differential or transaction log and Maintenance

Cleanup Task. Select which you want to carry out (on this example choose all) This wizard will bring you through every of the gadgets you have got selected to great track them.

Once you select the items you want for your plan click next, you could now rearrange them within the order you wish them to complete. It's high-quality to have Database Backup first in case of electricity failure, so select it and pass it to the top of the list. Click next.

Define Back Up Database (Full) Task

This display lets in you to pick out which full database backup you desire to perform it on. Best practice is to hold one plan per database, pick one database and choose next.

Define Database Check Integrity Task

This screen – the integrity challenge is a SQL Server command which exams the integrity of the database to peer if everything is not corrupt and stable. Select a database and click next.

Define Shrink Database Task

You can now configure to reduce the database in order to unfastened up space in the subsequent screen. It will only shrink area if available but should you want area within the destiny you'll ought to re allocate it. However, this step will assist backup speeds. Most developers don’t use this selection that much. Click next after selecting a database to reduce.

Define Reorganize Index Task

The next display is the Define Reorganize Index Tag display screen. When you add, alter and delete indexes you will, like tables, need to reorganize them. The technique is the same as a tough disk wherein you have there are fragmented files and space scattered across the disk. Best practice is to carry out this venture once consistent with week for a busy database. You can pick out to compact huge object which compacts any index which has massive binary item facts. Click subsequent to continue to the following display.

Define Rebuild Index Task

This display screen covers individual index rows. As cited both reorganize or reindexing. Doing both collectively in a single plan is pointless. Depending for your fragmentation level pick one or the other. In this example select your database and kind results in tempdb. Click subsequent to proceed.

Define Update Statistics Task

The update information task allows developer maintain song of facts retrieval as its created, modified and deleted. You can maintain the facts updated by performing this plan. Both facts for index and information for individual columns are kept. Select your database and click on next to proceed.

Define History Cleanup Task

You must now see the Define preservation cleanup challenge display screen which specifies the historical facts to delete. You can specify a shorter time body to hold the backup and recovery, agent job records and maintenance place for at the drop down. Click next to proceed. Define Back up Database (Differential) Task

This display lets in you to back up each page within the database which has been changed considering the fact that the last complete backup. Select a database you wish to use and click subsequent.

Define Back Up Database (Transaction Log) Task

The transaction log backup backs up all the log records because the last backup. You can pick a folder to shop it. Performing this kind of backup is the least useful resource in depth backup. Select a database and storage vicinity and click subsequent.

Define Execute SQL Server Agent Job Task

The SQL Server Agent Job Task deals with jobs which can be outside the wizard, as an instance it could be to test for nulls, check whether the database meets specified requirements etc. Any jobs which can be laid out in SQL Server Agent Job Task are listed here. Click next to proceed.

Define Maintenance Cleanup Task

This display defines the clean-up movement of the upkeep venture i.e. To ensure that the they are no longer taking over unnecessary area, so that you can specify wherein to store them. You can delete specific backup files. Click next to continue.

Report Options

The next display screen covers where you need to keep the file of the preservation plan. Make a note of wherein you'll shop it. You want to have e mail installation on SQL Server with a view to electronic mail it. Click subsequent to continue.

Complete the Wizard

The very last display is a whole overview of the wizard. You can assessment the summary of the plan and which options have been selected. Clicking finishes ends the wizard and creates the plan. You must now see a success display with the duties completed.

Running the upkeep plan

Once you successfully whole the maintenance wizard the following step is to run the plan you created. In order to get the plan to run you need to have the SQL Server Agent running. It is visible down from wherein Management is on SQL Server Management Studio. You can have left click on SQL Server Agent and then right click on and select Start.

Also, you may press the windows key + and press the letter r, then kind in services. MSc and hit return. Once Services appear scroll down and search for SQL Server Agent (MSSQLEXPRESS). SQL Server Express was set up in this EBook but you may pick the other variations like (MSSQLSERVER) if you installed that. Left click it, then right click on it and select Start.

You can go lower back to SSMS and proper click on the protection plan you created below preservation plans and then choose Execute. This will now run your plan. One successful crowning glory of the plan click adequate and close the communicate box. You can view the reports by right clicking the upkeep plan you created and selecting View history. On the left-hand side are all of the exceptional plans in SQL Server even as at the proper is the consequences of the unique plan.

Emailing the reviews.

A lot of DBA's want to get their database reports through email. What you need to do is to set up a database mail before you may fireplace off emails and then set up a Server agent to ship the e mail.

Setting up Database Mail

The first step is to proper click Database mail in SSMS and choose configure database mail. A wizard screen will seem, click next. Now choose the primary choice – installation Database Mail and click on next. Enter a profile call optional description of the profile. Now click on at the Add button to the right.

This will bring you to an add New Database Mail Account – SMTP. You need to enter the STMP information for an e mail account. Maybe you may installation a new e-mail account for this service. You can search online for SMTP information; Gmail works quite well (server name: smtp. Gmail.Com, port quantity 587, SSL required, tick simple authentication & affirm password). Click on adequate. Click next, click on public (important: so it can be utilized by the relaxation of the database). Set it as default profile, click on next, click subsequent again. You must now get an achievement screen. Click near.

SQL Server Agent

To ship off the database email you need to set up a Server Agent. Start by right clicking on SQL Server Agent – New – Operator. Give the operator a

name like Maintenance Plan Operator and enter in the electronic mail address you want to send the file to and click on good enough.

Now right the preservation plan which you have efficiently finished and pick modify. The renovation plan design display screen will seem at the right hand facet where you may see a few pics of the tasks completed in it. Now click on Reporting and Logging – it's far an icon situated at the menu bar of the layout plan - to the left of Manage Connections…

The Reporting and Logging window will appear. Select the tick box – Send report to an email recipient and select the Maintenance plan operator you just created. The next time you run the plan an email will be sent to the electronic mail deal with.

Structure of the SELECT statement

The SELECT Clause

The SELECT clause is the most effective re q uired clause in a SELECT statement, all the other clauses are optional. The SELECT columns can be literals (constants), expressions, desk columns and even subqueries. Lines may be commented with "--".

```
SELECT 15 * 15;                                    -- 225
SELECT Today = convert(DATE, getdate());           -- 2016-07-27
SELECT                  Color,
ProdCnt                 = COUNT(*),
AvgPrice                =
FORMAT(AVG(ListPrice),'c','en-US')
FROM AdventureWorks2016.Production.Product p
WHERE Color is not null
GROUP BY Color   HAVING count(*) > 10
ORDER BY AvgPrice DESC;
GO
Color  ProdCnt   AvgPrice
Yellow- 36   $959.09
Blue - 26      $923.68
Silver- 43     $850.31
Black- 93 $725.12
Red- 38 $1,401.95
-- E q  uivalent with column aliases on the right
SELECT                  Color,
COUNT(*)                                    AS
ProdCnt,
```

```
FORMAT(AVG(ListPrice),'c','en-US')              AS
AvgPrice
FROM AdventureWorks2016.Production.Product p
WHERE Color is not null  GROUP BY Color
HAVING count(*) > 10
ORDER BY AvgPrice DESC;
GO
```

SELECT with Search Expression

SELECT statement can have complicated expressions for text or numbers as demonstrated inside the subsequent T-SQL query for finding the road call in AddressLine1 column.

```
SELECT              AddressID,
SUBSTRING(AddressLine1, CHARINDEX(' ',
AddressLine1+' ', 1) +1,
CHARINDEX(' ', AddressLine1+' ', CHARINDEX(' ', AddressLine1+' ', 1)
+1) -
CHARINDEX(' ', AddressLine1+' ', 1)
-1)                                                AS StreetName,
AddressLine1,
City
FROM AdventureWorks2016.Person.Address
WHERE ISNUMERIC (LEFT(AddressLine1,1))=1
AND City = 'Seattle'
ORDER BY AddressLine1; -- -- (141 row(s) affected)- Partial results.
```

AddressID	StreetName	AddressLine1	City
13079	boulevard	081, boulevard du Montparnasse	Seattle

859	Oak	1050 Oak Street	Seattle
110	Slow	1064 Slow Creek Road	Seattle
113	Ravenwood	1102 Ravenwood	Seattle
95	Bradford	1220 Bradford Way	Seattle
32510	Steven	1349 Steven Way	Seattle
118	Balboa	136 Balboa Court	Seattle
32519	Mazatlan	137 Mazatlan	Seattle
25869	Calle	1386 Calle Verde	Seattle
114	Yorba	1398 Yorba Linda	Seattle
15657	Book	151 Book Ct	Seattle
105	Stillman	1619 Stillman Court	Seattle
18002	Carmel	1635 Carmel Dr	Seattle
19813	Acardia	1787 Acardia Pl.	Seattle
16392	Orchid	1874 Orchid Ct	Seattle
18053	Green	1883 Green View Court	Seattle
13035	Mt.	1887 Mt. Diablo St	Seattle
29864	Valley	1946 Valley Crest Drive	Seattle
13580	Hill	2030 Hill Drive	Seattle
106	San	2144 San Rafael	Seattle

SELECT Statement with Subquery

Two Northwind category images, Beverages & Dairy Products, from the dbo.Categories table.

The following SELECT statement entails a subquery which is called a derived desk. It additionally demonstrates that INNER JOIN may be carried out with a GROUP BY subquery as nicely not handiest with another table or view.

USE Northwind;

```
SELECT          c.CategoryName

Category,
cnum.NoOfProducts                       AS
CatProdCnt,
p.ProductName
Product,
FORMAT(p.UnitPrice,'c', 'en-
US')                    AS UnitPrice
FROM    Categories c
INNER JOIN Products p
ON c.CategoryID =
p.CategoryID
INNER JOIN
(       SELECT          c.CategoryID,
NoO
= count(* )
FROM    Categories c
INNER JOIN
Products p
ON
c.CategoryID = p.CategoryID
GROUP BY
c.CategoryID
)
cnum                                    -- derived table
ON c.CategoryID =
```

cnum.CategoryID ORDER BY Category, Product;

-- (77 row(s) affected) - Partial results.

Category	CatProdCnt	Product	UnitPrice
Dairy Products	10	Mozzarella di Giovanni	$34.80
Dairy Products	10	Queso Cabrales	$21.00
Dairy Products	10	Queso Manchego La Pastora	$38.00
Dairy Products	10	Raclette Courdavault	$55.00
Grains/Cereals	7	Filo Mix	$7.00
Grains/Cereals	7	Gnocchi di nonna Alice	$38.00
Grains/Cereals	7	Gustaf's Knäckebröd	$21.00
Grains/Cereals	7	Ravioli Angelo	$19.50
Grains/Cereals	7	Singaporean Hokkien Fried Mee	$14.00
Grains/Cereals	7	Tunnbröd	$9.00

Creating Delimited String List (CSV) with XML PATH

The XML PATH clause , the text() characteristic and correlated subquery is used to create a comma delimited string inside the SELECT columns. Note: it cannot be carried out the usage of traditional (without XML) SQL unmarried statement, it could be done with multiple SQL statements only. STUFF() string characteristic is carried out to update the leading comma with an empty string

```
USE AdventureWorks;
SELECT              Territory           = st.[Name],
SalesYTD =   FORMAT(floor(SalesYTD), 'c', 'en-US'), --
currency format
SalesStaffAssignmentHistory =
STUFF((SELECT CONCAT(', ', c.FirstName, SPACE(1),
c.LastName)        AS [text()]
FROM   Person.Contact c
INNER JOIN Sales.SalesTerritoryHistory sth
ON c.ContactID = sth.SalesPersonID
WHERE  sth.TerritoryID
=   st.TerritoryID
ORDER  BY StartDate
FOR XML Path ('')), 1, 1, SPACE(0))
FROM   Sales.SalesTerritory st
ORDER  BY SalesYTD DESC;
GO
```

Territory	SalesYTD	SalesStaffAssignmentHistory
Southwest	$8,351,296.00	Shelley Dyck, Jauna Elson
Canada	$6,917,270.00	Carla Eldridge, Michael Emanuel, Gail Erickson
Northwest	$5,767,341.00	Shannon Elliott, Terry Eminhizer, Martha Espinoza
Central	$4,677,108.00	Linda Ecoffey, Maciej Dusza
France	$3,899,045.	Mark Erickson

	00	
Northeast	$3,857,163.00	Maciej Dusza, Linda Ecoffey
United Kingdom	$3,514,865.00	Michael Emanuel
Southeast	$2,851,419.00	Carol Elliott
Germany	$2,481,039.00	Janeth Esteves
Australia	$1,977,474.00	Twanna Evans

Logical Processing Order of the SELECT Statement

The consequences from the previous step will be to be had to the following step. The logical processing order for a SELECT declaration is the following. Actual processing by means of the database engine can be different because of overall performance and other considerations.

1.	FROM
2.	ON
3.	JOIN
4.	WHERE
5.	GROUP BY
6.	WITH CUBE or WITH ROLLUP
7	HAVING

.	
8.	SELECT
9.	DISTINCT
10.	ORDER BY
11.	TOP

As an example, it's far logical to filter with the WHERE clause previous to applying GROUP BY. It is also logical to sort while the very last end result set is to be had.

SELECT Color, COUNT(*) AS ColorCount FROM AdventureWorks2016.Production.Product

WHERE Color is not NULL GROUP BY Color ORDER BY ColorCount

DESC;

Color	ColorCount
Black	93
Silver	43
Red	38
Yellow	36
Blue	26
Multi	8
Silver/Black	7

White	4
Grey	1

The TOP Clause

The TOP clause filters effects according the sorting laid out in an ORDER BY clause, otherwise random filtering takes place.

Simple TOP usage to return 10 rows only.

SELECT TOP 10 SalesOrderID, OrderDate, TotalDue

FROM AdventureWorks2016.Sales.SalesOrderHeader ORDER BY TotalDue DESC;

SalesOrderID	OrderDate	TotalDue
51131	2007-07-01 00:00:00.000	187487.825
55282	2007-10-01 00:00:00.000	182018.6272
46616	2006-07-01 00:00:00.000	170512.6689
46981	2006-08-01 00:00:00.000	166537.0808
47395	2006-09-01 00:00:00.000	165028.7482
47369	2006-09-01 00:00:00.000	158056.5449
47355	2006-09-01 00:00:00.000	145741.8553
51822	2007-08-01 00:00:00.000	145454.366
44518	2005-11-01 00:00:00.000	142312.2199
51858	2007-08-01 00:00:00.000	140042.1209

51858 2007-08-01 00:00:00.000 140042.1209

Complex TOP function usage: not known in advance how many rows will be returned due to "TIES".

SELECT TOP 1 WITH TIES coalesce(Color,

'N/A') AS Color,

FORMAT(ListPrice, 'c', 'en-

```
US')                                    AS ListPrice,
Name
ProductName,
ProductID
FROM     AdventureWorks2016.Production.Product
ORDER BY ROW_NUMBER()  OVER(PARTITION BY Color ORDER
BY ListPrice DESC);
```

Color	ListPrice	ProductName	ProductID
N/A	$229.49	HL Fork	804
Black	$3,374.99	Mountain-100 Black, 38	775
Red	$3,578.27	Road-150 Red, 62	749
Silver	$3,399.99	Mountain-100 Silver, 38	771
Blue	$2,384.07	Touring-1000 Blue, 46	966
Grey	$125.00	Touring-	842
		Panniers, Large	
Multi	$89.99	Men's BibShorts, S	855
Silver/Black	$80.99	HL Mountain Pedal	937
White	$9.50	Mountain Bike Socks, M	709
Yellow	$2,384.07	Touring-1000 Yellow, 46	954

The DISTINCT Clause to Omit Duplicates

The DISTINCT clause returns only unique results, omitting duplicates in the result set.

USE AdventureWorks2016; SELECT DISTINCT Color FROM Production.Product

WHERE Color is not NULL

ORDER BY Color;

GO

Color
Black
Blue
Grey
Multi
Red
Silver
Silver/Black
White
Yellow

SELECT DISTINCT ListPrice

FROM Production.Product

WHERE ListPrice > 0.0

ORDER BY ListPrice DESC;

GO

-- (102 row(s) affected) - Partial results.

ListPrice
3578.27
3399.99

3374.99
2443.35

-- Using DISTINCT in COUNT - NULL is counted

SELECT COUNT(*) AS

TotalRows,

COUNT(DISTINCT Color) AS

ProductColors,

COUNT(DISTINCT Size) AS

ProductSizes

FROM AdventureWorks2016.Production.Product;

TotalRows ProductColors ProductSizes

504 9 1

The CASE Conditional Expression

The CASE conditional expression evaluates to a single value of the same data type, therefore it can be used anywhere in a q uery where a single value is required.

SELECT CASE ProductLine

WHEN 'R' THEN 'Road'

WHEN 'M' THEN 'Mountain'

WHEN 'T' THEN 'Touring'

WHEN 'S' THEN 'Other'

ELSE 'Parts'

END AS

Category,

Name AS

ProductName,

ProductNumber

```
FROM AdventureWorks2016.Production.Product
ORDER BY ProductName;
GO
-- (504 row(s) affected) - Partial results.
```

Category	ProductName	ProductNumber
Touring	Touring-3000 Blue, 62	BK-T18U-62
Touring	Touring-3000 Yellow, 44	BK-T18Y-44
Touring	Touring-3000 Yellow, 50	BK-T18Y-50
Touring	Touring-3000 Yellow, 54	BK-T18Y-54
Touring	Touring-3000 Yellow, 58	BK-T18Y-58
Touring	Touring-3000 Yellow, 62	BK-T18Y-62
Touring	Touring-Panniers, Large	PA-T100
Other	Water Bottle - 30 oz.	WB-H098
Mountain	Women's Mountain Shorts, L	SH-W890-L

Query to return different result sets for repeated execution due to newid().

```
SELECT   TOP 3 CompanyName,     City=CONCAT(City, ', ',
Country),               PostalCode,
[IsNumeric] =   CASE              WHEN PostalCode like '[0-9] [0-9][0-9][0-9][0-9]'
THEN '5-Digit Numeric'     ELSE 'Other'  END
FROM     Northwind.dbo.Suppliers
ORDER BY NEWID();                                  -- random sort
GO
```

CompanyName	City	PostalCode	IsNumeric

PB Knäckebröd AB	Göteborg, Sweden	S-345 67	Other
Gai pâturage	Annecy, France	74000	5-Digit Numeric
Heli Süßwaren GmbH & Co. KG	Berlin, Germany	10785	5-Digit Numeric

```
SELECT ROW_NUMBER() OVER (ORDER BY Name) AS RowNo,
CASE ProductLine
WHEN 'R' THEN 'Road'
WHEN 'M' THEN 'Mountain'
WHEN 'T' THEN 'Touring'
WHEN 'S' THEN 'Other'
ELSE 'Parts'
END
Category,
Name A
ProductName,
CASE WHEN Color is null THEN 'N/A'
ELSE Color
END AS Color,
ProductNumber
FROM Production.Product ORDER BY ProductName;
-- (504 row(s) affected) - Partial results.
```

Row	Categor	ProductNam	Col	ProductNumbe

No	y	e	or	r
1	Parts	Adjustable Race	N/A	AR-5381
2	Mountain	All-Purpose Bike Stand	N/A	ST-1401
3	Other	AWC Logo Cap	Multi	CA-1098
4	Parts	BB Ball Bearing	N/A	BE-2349
5	Parts	Bearing Ball	N/A	BA-8327
6	Other	Bike Wash Dissolver	N/A	CL-9009
7	Parts	Blade	N/A	BL-2036
8	Other	Cable Lock	N/A	LO-C100
9	Parts	Chain	Silver	CH-0234
10	Parts	Chain Stays	N/A	CS-2812

Testing PostalCode with ISNUMERIC and generating a flag with CASE expression.

```
SELECT TOP (4) AddressID, City,
PostalCode AS Zip,
CASE WHEN ISNUMERIC(PostalCode) = 1 THEN
'Y' ELSE 'N' END AS IsZipNumeric
FROM AdventureWorks2008.Person.Address ORDER BY NEWID();
```

SQL Data Types

n this chapter you will research the position of facts in a database model, how it's far described, its characteristics and the numerous types that the SQL software program supports. There are general statistics types that are in addition classified into unique subtypes. It is advisable which you use described data sorts to make certain the portability and comprehensibility of the database model.

Data Definition

Data is the saved facts in a database that you could manipulate anytime that you want. If you may remember the calling card instance in its database version is a group of customers' names, contact numbers, business enterprise addresses, process titles and so on. When regulations are furnished on the way to write and store records, then you need to have a clear expertise of the one of a kind information kind. You need to take into consideration the duration or space allocated by the database for each table column and what facts values it must include whether it's far just all letters or all numbers, combination or alphanumeric, graphical, date or time. By defining what facts type is stored in every area at some stage in the design phase, statistics access errors can be prevented. This is the sector definition process, a form of validation that controls how incorrect facts is to be entered into the database.

When a certain database field does no longer have any statistics gadgets at all, then the price is unknown or what is referred to as a null value. This is completely unique from the numeric 0 or the blank man or woman value, considering the fact that zeroes and blanks are nevertheless considered particular values. Check out the subsequent eventualities when you may have a null price:

- Even if the information value could likely exist, you don't recognize what it's far yet.
- The cost does no longer clearly exist yet.
- The price could be out of range.
- The area isn't appropriate for a specific row.

SQL Data Types

These are the general sorts of SQL information types and their subtypes.

Numeric – The price described by means of this records kind is both a specific or an approximate number.

Exact Numeric

INTEGER – This is composed of nice and poor whole numbers without any decimal nor a fractional part. The INTEGER records fee levels from terrible 2,147,483,648 to fantastic 2,147,483,647, with a most storage size of 4 bytes.

SMALLINT – This replaces integers whilst you want to save some storage space. However, its precision cannot be larger than that of an integer. Precision in laptop programming is the maximum general of great digits a sure quantity may have. The SMALLINT data fee tiers from poor 32,768 to positive 32,767, with a maximum storage length of two bytes.

BIGINT – This is the opposite of SMALLINT, wherein the minimal precision is the same or extra than that of an INTEGER. The BIGINT records cost tiers from poor

9,223,372,036,854,775,808 to fantastic

9,223,372,036,854,775,807, with a maximum storage length of 8 bytes.

NUMERIC (p, s) – This data kind incorporates an integer component and a fractional component that indicates the precision and scale of the records price. Scale is the wide variety of digits reserved inside the fractional a part of the information value (positioned at the right aspect of the decimal factor). In NUMERIC (p, s), ‘p’ specifies the precision whilst ‘s’ specifies the scale. For instance, NUMERIC (6, 3) approach that the variety has a complete of 6 sizeable digits with 3 digits following the decimal point. Therefore, its absolute fee will only be as much as 999.999.

DECIMAL (p, s) – This also has a fractional aspect where you may specify each the statistics cost’s precision and scale, but allows for greater precision. For instance, DECIMAL (6, 3) can include values up to 999.999 however the database will still be given values large than 999.999 with the aid of rounding off the number. Let us say you entered the range 123.4564, the cost a good way to be stored is 123.456. Thus, the precision given specifies the allotted storage length for this statistics kind.

Approximate Numeric

REAL (s) – This is a single-precision, floating-point number where the decimal factor can "float" inside the stated range. This gives a countless precision and a scale of variable lengths for the statistics type's decimal price. For instance, the values for π (pi) can encompass 3.1, three.14 and three.14159 (every fee has its very own precision). This records type's precision degrees from 1 as much as 21, with a most storage length of 4 bytes.

DOUBLE PRECISION (p, s) – As what the name suggests, that is a double-precision, floating-factor quantity with a storage capability of twice the REAL information type. This facts type is appropriate when you require more unique numbers, such as in maximum scientific subject of disciplines. This facts kind's precision tiers from 22 up to 53 digits, with a maximum storage length of 8 bytes.

FLOAT (p, s) – This facts type helps you to specify the value's precision and the computer makes a decision whether it will be a single or a double-precision quantity. It will permit both the precision of REAL and DOUBLE PRECISION information types. Such capabilities make it easier to move the database from one laptop platform to another.

String – Considered as the maximum normally used data type, this stores alphanumeric statistics.

CHARACTER (n) or CHAR (n) – Known as a fixed-duration string or a constant man or woman, this information kind consists of strings which have the same length (represented through 'n', that is the maximum quantity of characters allotted for the described area). For example, placing the column's records type to CHAR (23) means the most period of the records to be stored in that area is 23 characters. If its period is less than 23, then the remaining areas are packed with blanks by SQL. However, this turns into the drawback of using fixed-period strings because storage space is definitely wasted. On the alternative hand, if the duration is not specified, then SQL assumes a length of just one individual. The CHARACTER facts kind will have a maximum duration of 254 characters.

CHARACTER VARYING (n) or VARCHAR (n) – This facts type is for entries which have exceptional lengths, however the remaining areas will no longer be filled by way of areas. This approach that the exact number of characters entered will be saved within the database to avoid area wastage.

The maximum length for this information kind is 32,672 characters with no default cost.

CHARACTER LARGE OBJECT (CLOB) – This was added in SQL:1999 where the variable-period information type is used, which contains a Unicode, person-based statistics. Such facts are too large to be saved as a CHARACTER type, simply like big documents, and the most price is up to 2,147,483,647 characters long.

Date and Time – This statistics type handles statistics related to dates and times.

DATE – This presents a storage area for the date's year, month and day values (in that particular order). The cost for the 12 months is expressed in four digits (represented with the aid of values ranging from 0001 as much as 9999), while the month and day values are both represented by any two digits. The layout of this data kind is: 'yyyy-mm-dd.'

TIME – This stores and presentations time values the usage of an hour-minute-2d format ("HH:MM:SS").

DATETIME – This carries each date and time information displayed the usage of the "YYYY-MM-DD HH:MM:SS" format. The range of this information type is from "1000-01-01 00:00:00" to "9999-12-31 23:59:59".

TIMESTAMP – Similar to the DATETIME information kind, this degree from "1970-01-01 00:00:01" UTC to "203801-19 03:14:07" UTC.

Boolean – This records type is used for comparing facts and based totally from the outcomes they could go back TRUE, FALSE, or NULL values. If all the conditions for a given query are met, then Boolean price returns TRUE. Otherwise, the value is both FALSE or NULL.

User-Defined Data Type

We will now discuss consumer-described records sorts or truly UDT's. By the call itself, the person defines or specifies the data values primarily based at the existing statistics types. This allows customization to satisfy other person necessities and maximize the available storage space. Moreover, programmers enjoy the flexibility they convey in growing database applications. UDT's make it viable when you need to save the identical kind of data in a column that will additionally be described in several tables. The CREATE TYPE assertion is used to outline UDT's.

For example, if you need to apply two specific currencies in your database like the US dollar and the United Kingdom pound, you can create and outline the following UDT's:

CREATE TYPE US Dollar AS DECIMAL (9, 2);

CREATE TYPE Pound AS DECIMAL (9, 2);

Data is the stored records in a database that a user can outline and control.

There are one-of-a-kind popular SQL statistics kinds, specifically numeric, string, date and time, and Boolean.

If you need to outline extra specific records sorts when designing your database model, you may use the exclusive subtypes below each trendy SQL data type.

UDT's or consumer-described information kinds are custom designed and created by means of the person primarily based at the existing statistics types, which offers flexibility in developing numerous database applications.

In the following chapter you'll research the common SQL commands that are used to create, control and retrieve statistics from a database, in a green and effective way.

Cursors: Read Only, Forward Only, Fast Forward, Static, Scroll

Dynamic, Optimistic

Cursors are database items which are used to iterate over a set of rows and usually perform some extra logical operations or others on each row of fetched statistics. The process of cursor operation entails following tasks:

- Declare the variables to be used
- Define the Cursor and kind of cursor
- Populate Cursor with values the usage of SELECT
- Open Cursor that turned into declared & populated above
- Fetch the values from Cursor in to declared variables
- While loop to fetch next row and loop till no longer rows exist
- Perform any data processing on that row inside at the same time as loop
- Close Cursor to gracefully un-lock tables in any
- Remove Cursor from memory

However, Cursors are typically no longer recommended due to row-by-row fetching and processing of statistics, consumption of memory (due to allocation of temporary desk and filling it with the result set) and occasionally locking tables in unpredictable ways. Thus, alternate tactics like the usage of at the same time as loop wishes to be considered before the use of cursors.

LOCAL/GLOBAL CURSOR

This cursor specifies the scope and whether this scope allows domestically to a stored procedure or trigger or maybe a batch OR if the scope of the cursor is applicable globally for the connection. Cursor is valid most effective within the scope defined.

FORWARD_ONLY CURSOR

FORWARD_ONLY Cursor specifies that rows can most effective be scrolled from first to the end row. Thus, this precludes transferring to earlier or final row and fetch subsequent is simplest choice available. Below is an example of FORWARD_ONLY Cursor:

```
-- ============================================
-- Author:		Neal Gupta
-- Create date: 12/01/2013
-- Description: Create a Cursor for displaying customer info
-- ============================================DECLARE
@CustomerID INT
,@FirstName VARCHAR(50)
,@LastName VARCHAR(50)
,@City VARCHAR(50)
,@State VARCHAR(10)
,@ZipCode VARCHAR(10)
-- Create a Cursor
DECLARE curTblCustomer CURSOR FORWARD_ONLY
FOR
SELECT
CustomerID, FirstName, LastName, City, [State], ZipCode
FROM [IMS].[dbo].[TblCustomer]
ORDER BY CustomerID ASC;
-- Open the Cursor
OPEN curTblCustomer
-- Get the first Customer
FETCH NEXT FROM curTblCustomer INTO @CustomerID, @FirstName,
@LastName, @City, @State,@ZipCode
PRINT 'Customer Details:'
-- Loop thru all the customers
WHILE @@FETCH_STATUS = 0
BEGIN
```

```
-- Display customer details
PRINT CAST(@CustomerID AS VARCHAR(50)) + ' ' +
@FirstName + ' ' + @LastName + ' '+  @City + ' '+ @State + ' '+
@ZipCode
-- Get the next customer
FETCH NEXT FROM curTblCustomer INTO @CustomerID,
@FirstName, @LastName, @City, @State, @ZipCode
END
-- Close Cursor
CLOSE curTblCustomer
-- Remove Cursor from memory of temp database
DEALLOCATE curTblCustomer
```

Cursor through default is FORWARD_ONLY, if STATIC, KEYSET or DYNAMIC alternatives are not noted and cursor works as a DYNAMIC one if those three keywords are not certain.

READ_ONLY CURSOR

This cursor is similar to above FORWARD_ONLY cursor, besides that updates on the current fetched row cannot be performed.

FAST_FORWARD CURSOR

This cursor is actually a combination of FAST_FORWARD (#1) and READ_ONLY (#2) along with performance optimizations. Since, it's miles a fast forward cursor, it precludes scrolling to earlier or closing row and being read most effective cursor also, prevents update of modern-day fetched row. However, due to these 2 restrictions, they help SQL server to optimize the overall cursor performance.

```
-- ============================================
-- Description: Create a FAST_FORWARD Cursor
============================================DECLARE
@CustomerID INT
```

```
,@FirstName VARCHAR(50)
,@LastName VARCHAR(50)
,@City VARCHAR(50)
,@State VARCHAR(10)
,@ZipCode VARCHAR(10)
-- Create a Cursor
DECLARE curTblCustomer CURSOR FAST_FORWARD
FOR
SELECT
CustomerID
,FirstName
,LastName
,City
,[State]
,ZipCode
FROM
[IMS].[dbo].[TblCustomer]
ORDER BY
CustomerID ASC;
-- Open the Cursor
OPEN curTblCustomer
-- Get the first Customer
FETCH NEXT FROM curTblCustomer INTO @CustomerID, @FirstName,
@LastName, @City, @State, @ZipCode
PRINT 'Customer Details:'
-- Loop thru all the customers
WHILE @@FETCH_STATUS = 0 BEGIN
```

```
-- Display customer details
PRINT CAST(@CustomerID AS VARCHAR(50)) + ' ' +
@FirstName + ' ' + @LastName + ' '+  @City + ' '+ @State + ' '+
@ZipCode
-- Get the next customer
FETCH NEXT FROM curTblCustomer INTO @CustomerID, @FirstName,
@LastName, @City, @State, @ZipCode
END
-- Close Cursor
CLOSE curTblCustomer
-- Remove Cursor from memory of temp database
DEALLOCATE curTblCustomer
```

STATIC CURSOR

If a cursor is precise as STATIC, SQL server takes a image of the information and places into brief desk in tempdb database. So, when the cursor fetches next row, information comes from this brief desk and therefore, if some thing is modified within the original desk, it is not reflected within the transient desk. This makes the performance of cursor quicker in comparison to dynamic cursor (explained underneath) since next row of statistics is already pre-fetched in temp database.

DYNAMIC CURSOR

As the call suggests, whilst the cursor is scrolling to subsequent row, information for that row is dynamically added from the original desk, and if there has been any change, it is reflected within the records fetched as well.

SCROLL CURSOR

This cursor allows scrolling of rows: FIRST, LAST, PRIOR, NEXT and if a cursor is not specified as SCROLL, it can simplest perform FETCH subsequent row. If the cursor is FAST_FORWARD, SCROLL option can't be used.

OPTIMISTIC/SCROLL_LOCKS CURSOR

Below is a cursor declaration the use of a number of the above alternatives:

LOCAL,

FORWARD_ONLY, STATIC and READ_ONLY:

DECLARE curTblCustomerOp1 CURSOR

LOCAL

FORWARD_ONLY

STATIC

READ_ONLY

FOR

-- Rest of SQL remains same as used in FORWARD_ONLY Cursor

Another cursor declaration could use following options: GLOBAL, SCROLL, DYNAMIC, OPTIMISTIC:

DECLARE curTblCustomerOp2 CURSOR

GLOBAL -- OR USE LOCAL

SCROLL -- OR USE FORWARD_ONLY

DYNAMIC -- OR USE FAST_FORWARD/STATIC/KEYSET

OPTIMISTIC -- OR USE READ_ONLY/SCROLL_LOCKS FOR

-- Rest of SQL remains same as used in FORWARD_ONLY Cursor

NESTED CURSOR

As the name suggest, cursors can be nested, that means one cursor could have another inner cursor and so on. In below instance we will use one nested cursor.

-- ==========

-- Description: Create a Nested Cursor for displaying Orders

-- and products ordered

-- ========== DECLARE

@OrderID INT

,@ProductID INT

```
,@OrderQty INT
,@OrderDate DATETIME
,@Name VARCHAR(50)
,@Manufacturer VARCHAR(50)
,@Price DECIMAL(9,2)
PRINT '***** Orders Details *****'
--- First, declare OUTER Cursor
DECLARE curTblOrder CURSOR
LOCAL
FORWARD_ONLY
STATIC
READ_ONLY
TYPE_WARNING
FOR
SELECT
OrderID
,ProductID
,OrderQty
,OrderDate
FROM
[IMS].[dbo].[TblOrder]
ORDER BY
OrderID
-- Open OUTER Cursor
OPEN curTblOrder
-- Fetch data from cursor and populate into variables
```

```
FETCH NEXT FROM curTblOrder INTO @OrderID, @ProductID,
@OrderQty, @OrderDate
WHILE @@FETCH_STATUS = 0
BEGIN
PRINT '*** Order: ' + ' ' + CAST(@OrderID AS
VARCHAR(10))
-- Now, declare INNER Cursor
DECLARE curTblProduct CURSOR
FOR
SELECT Name, Manufacturer, Price
FROM [IMS].[dbo].[TblProduct] P
WHERE P.ProductID = @ProductID
-- Open INNER Cursor
OPEN curTblProduct
FETCH NEXT FROM curTblProduct INTO @Name, @Manufacturer,
@Price
-- Loop for INNER Cursor
WHILE @@FETCH_STATUS = 0 BEGIN
PRINT 'Product: ' + @Name + ' ' + @Manufacturer + ' ' + CAST(@Price AS
VARCHAR(15))
FETCH NEXT FROM curTblProduct INTO @Name,
@Manufacturer, @Price
END
-- Close INNER Cursor first and deallocate it from temp database
CLOSE curTblProduct
DEALLOCATE curTblProduct
-- Fetch next Order
```

```
FETCH NEXT FROM curTblOrder INTO @OrderID,
@ProductID, @OrderQty,
@OrderDate
END
-- Finally, close OUTER Cursor and deallocate it from temp database
CLOSE curTblOrder
DEALLOCATE curTblOrder
```

FOR UPDATE CURSOR

This cursor lets in updating the column values inside the fetched row for the specified columns simplest, however, if the columns are not specified, then, all of the columns may be updated for the row beneath consideration the use of WHERE CURRENT OF clause.

ALTERNATIVE APPROACH

Note that in above examples, we used cursors to demonstrate the capability of different types of cursor, however, we could have used alternative approach, like using WHILE loop and counter approach to perform similar task, as became done in #2 above, as under:

```
-- ================================================
-- Description: Alternative Approach
-- using WHILE loop and Counter Method
-- ================================================
DECLARE @Customers TABLE
RowID INT IDENTITY(1,1) PRIMARY KEY
,CustomerID INT
,FirstName VARCHAR(50)
,LastName VARCHAR(50)
,City VARCHAR(50)
,[State] VARCHAR(25)
,ZipCode VARCHAR(10) DECLARE
```

```
@StartCount INT = 1             -- First Row Count
,@EndCount INT                  -- Total Row Counts
,@CustomerID INT
,@FirstName VARCHAR(50)
,@LastName VARCHAR(50)
,@City VARCHAR(50)
,@State VARCHAR(50)
,@ZipCode VARCHAR(10)
-- Bulk Insert all the customers into temp table: @Customers
INSERT INTO @Customers (CustomerID,FirstName,LastName,City,
[State],ZipCode)
SELECT
CustomerID
,FirstName
,LastName
,City
,[State]
,ZipCode
FROM
[IMS].[dbo].[TblCustomer] WITH (NOLOCK)
-- EndCount is set to total of all rows fetched in above SELECT
SELECT @EndCount = @@ROWCOUNT
-- Loop thru all the rows
WHILE @StartCount <= @EndCount
BEGIN
SELECT
@CustomerID = CustomerID
```

```
,@FirstName = FirstName
,@LastName = LastName
,@City = City
,@State = [State]
,@ZipCode = ZipCode
FROM @Customers
WHERE
RowID = @StartCount
PRINT 'Fetched Row#: ' + CAST(@StartCount AS VARCHAR(5)) + ' from
TblCustomer table. Details below: '
PRINT 'CustomerID = '+ CAST(@CustomerID
AS VARCHAR(5)) + '
FirstName = ' + @FirstName + ' LastName = ' + @LastName + ' City
= ' + @City + ' State = ' + @State + ' ZipCode = ' +
@ZipCode
SELECT @StartCount += 1
END
```

Preparation

To efficaciously hook up with and paintings with a SQL Server Database out of your Windows PC (laptop or notebook), there are several things that ought to be properly setup or configured. None of these are very complex, but failure to deal with those items can and will bring about both problems, warnings and/or errors. So whilst this chapter may be brief, the basic standards (and their mental images) are nonetheless very critical on your success. So it might be smart to study and learn this chapter's material well.

Database Architecture

The SQL Server database itself will most usually are living upon a server somewhere inside your organization. While you could each run and get admission to the SQL Server database on most PC's these days, the uncooked performance, tight security and excessive availability requirements alone generally require a secure, centrally controlled database server.

The critical object of note is that both the client and the server need some supporting SQL Server network library files in order for communication between your application and the database to occur. Because it's not necessary to recognize something about TDS to hook up with a SQL Server database, we won't delve any further into the specifics approximately it.

What is essential if you need to have the ones community library documents on your PC for database connections to function well. Since Windows 2000, all versions of Windows have at least a basic set of those community library files already pre-installed. SQL Server 2005 added a new set of network files known as the SQL Server Native Client which provides additional capability based totally on new capabilities that was added to that model of SQL Server. The features are no longer essential to retrieve records from a SQL Server database however middle around safety and management functionality which includes the ability to address passwords just like the way Windows does. However, in case you don't use those additional features, chances are you already have the entirety you want to connect with the SQL Server database.

Database Versions

The SQL Server database, like maximum software program, has distinctive variations – which includes

SQL Server 2000, 2005, 2008, and 2008 R2. These are really the advertising and marketing names for the preliminary or base releases. In addition, there are various patches available, usually referred to as provider packs or cumulative updates. If you're experiencing an issue with something not working the manner you'd assume it to, you might take a look at along with your machine administrator or database administrator to peer if this type of is needed. But typically they are most effective required in case you hooked up precise purchaser software from the SQL Server CDs/DVDs.

Connecting

Thus far we've primarily covered terminology and that is a prerequisite for correctly running along with your SQL Server databases. Now it's time to carry out your very first SQL Server database task – connecting on your database. That may seem like an anticlimactic task but it's the first step in utilizing the information within those databases.

Think of creating the database connection like making a telephone name. If you don't have the proper equipment, a service plan, the number of whom you're calling and expertise of the way to dial that range, then you can't provoke a smartphone call and thus cannot hold a meaningful conversation. The equal is true for databases. You should efficaciously connect before you could retrieve, insert or update your statistics.

Connecting Via ODBC

A lot of programs connect with SQL Server the use of a method referred to as ODBC. Either the application will gift an interface where you could create your connection or it will ask you for a current ODBC connection. Chances are that if it offers an interface, the steps may be similar to while you create your own ODBC connection. In fact, loads of times programs will actually re-use Windows' very own tool for managing those connections. So let's observe how to do that.

Control Panel icon

You'll need to click on it and with the intention to bring up the Control Panel. If you're in Classic View, you have to see an icon for Administrative Tools. If you're in Category View, you'll must double-click on the Performance and Maintenance icon first. Once you notice the Administrative Tools icon, doubleclick on it and you have to have a new listing of alternatives. What

you're seeking out is the icon for Data Sources (ODBC). Double-click on it and it's going to bring up the tool ODBC Administrator where you'll be able to configure a new ODBC connection.

There are two sorts of statistics sources referred to as statistics set names (i.e. DSN): user and gadget. User ones can best be seen and utilized by the modern Windows consumer, whereas system ones may be seen and utilized by any Windows person on that equal PC.

ODBC Admin Main Screen

Here we have an awesome range of choices, however usually you'll just want the only that announces SQL Server. In this example, the entire SQL Server equipment are mounted for several variations of SQL Server, that is why you furthermore mght see picks for SQL Server Native Client. You in all likelihood won't have the SQL Server Native Client selections except you need them. Simply select SQL Server and click on Finish.

SQL Server Data Source

In the event that you'll one day want to connect to a distinctive database server (Oracle, DB2, etc.) usually speaking you will have the most reliable effects and fastest performance the use of the ODBC driving force from the database vendor and it in all likelihood can be named so you easily understand it.

Choose the Name and SQL Server

You'll then need to choose how you connect to the SQL Server within the sense of how does SQL Server understand who you are. Most of the time, you'll connect the use of the user account you logged on to Windows with. If you need a unique SQL Server login, your device or database administrator should offer it ahead of time, along with an appropriate password. In both case, make the perfect selections for the next screen,

Tell SQL Server Who You Are

Next, you'll need to specify any other alternatives for the connection. Here you'll want to test the checkbox to change the default database and then choose the right one from the drop down list. If you need to get a mistakes here, it approaches both the SQL Server you specified within the previous display is not available, or you mistyped its call, or you don't have

permission to connect. If that's the case, go lower back to the previous display, test the name of the SQL Server, and in case you trust it's proper, observe up along with your system or database administrator. It may be down or there may be another motive as to why you can't connect with it.

Specify the Database

This is the database I recognize consists of the sales information I need to get entry to. You'll want to recognize each the server and the database name to get at the facts saved at the SQL Server.

Click the Next button to visit the remaining configuration display screen, in which you probably won't need to make any changes, then click Finish to create your ODBC connection. You have to be provided with a display like in - be sure to check your connection to make sure everything is excellent before clicking OK.

In this chapter we reviewed the simple ideas and processes you want to recognize and perform so that it will begin effectively running along with your SQL Server databases. We additionally reviewed covered the prerequisite know-how to tackle the first and most important database task – connecting. Much like a phone call – we entered the records necessary to dial in to the database and then positioned the call.

Working with Filters

WHERE Clause

WHERE is the most widely used clause. It helps retrieve exactly what you require. The following example table shows the STUDENT STRENGTH in various Engineering courses in a college:

ENGINEERING_STUDENTS

ENGG_ID	ENGG_NAME	STUDENT_STRENGTH
1	Electronics	150
2	Software	250
3	Genetic	75
4	Mechanical	150
5	Biomedical	72
6	Instrumentation	80
7	Chemical	75
8	Civil	60
9	Electronics & Com	250
10	Electrical	60

Now, if you need to recognize how many publications have over 200 in STUDENT STRENGTH, then you may simplify your search by means of passing on an easy statement:

```
SELECT ENGG_NAME, STUDENT_STRENGTH
FROM ENGINEERING_STUDENTS
WHERE STUDENT_STRENGTH > 200;
```

ENGG_NAME	STUDENT_STRENGTH
Software	250
Electronics & Com	250

HAVING Clause

HAVING is every other clause used as a clear out in SQL. At this point, it's miles crucial to understand the distinction among the WHERE and HAVING clauses. WHERE specifies a circumstance, and simplest that set of data that passes the condition can be fetched and displayed in the end result set. HAVING clause is used to filter grouped or summarized information. If a SELECT question has each WHERE and HAVING clauses, then while WHERE is used to clear out rows, the result is aggregated so the HAVING clause may be applied. You will get a better know-how whilst you see an example.

For an explanation, every other desk by using the call of Dept_Data has been created, and it is defined as follows:

Field	Type	Null	Key	Default	Extra
Dept_ ID	Bigint (20)	NO	PRI	NULL	auto_ increment
HOD	Varchar (35)	NO			
NO_ OF_ Prof	Varchar (35)	YE S		NULL	
ENGG_ ID	Smallint (6)	YE S	MUL	NULL	

Now, let’s have a take a look at the information to be had in this table:

Where Dept_ID is set to 100.

Dept_ID	HOD	NO_OF_Prof	ENGG_ID
100	Miley Andrews	7	1
101	Alex Dawson	6	2
102	Victoria Fox	7	3

103	Anne Joseph	5	4
104	Sophia Williams	8	5
105	Olive Brown	4	6
106	Joshua Taylor	6	7
107	Ethan Thomas	5	8
108	Michael Anderson	8	9
109	Martin Jones	5	10

There are a few easy differences among the WHERE and HAVING clauses. The WHERE clause can be used with SELECT, UPDATE, and DELETE clauses, however the HAVING clause does now not experience that privilege; it is handiest used within the SELECT question. The WHERE clause can be used for character rows, however HAVING is carried out on grouped records. If the WHERE and HAVING clauses are used together, then the WHERE clause can be used before the GROUP BY clause, and the HAVING clause may be used after the GROUP BY clause. Whenever WHERE and HAVING clauses are used

together in a query, the WHERE clause is carried out first on each row to filter out the results and make sure a group is created. After that, you may observe the HAVING clause on that group.

Now, based totally on our preceding tables, let's see which departments have extra than five professors:

SELECT * FROM Dept_Data WHERE NO_OF_Prof > 5;

Look at the WHERE clause here. It will check each and every row to see which record has NO_OF_Prof > 5.

Dept_ID	HOD	NO_OF_Prof	ENGG_ID
100	Miley Andrews	7	1
101	Alex Dawson	6	2
102	Victoria Fox	7	3

104	Sophia Williams	8	5
106	Joshua Taylor	6	7
108	Michael Anderson	8	9

Now, let's locate the names of the Engineering courses for the above records:

```
SELECT e. ENGG_NAME, e.STUDENT_STRENGTH,
d.HOD,d.NO_OF_Prof,d.Dept_ID
FROM ENGINEERING_STUDENTS e, Dept_Data d
WHERE d.NO_OF_Prof > 5 AND e.ENGG_ID = d.ENGG_ID;
```

The result set will be as follows:

ENGG_NAME	STUDENT_ STRENGTH	HOD	NO _ OF_ Prof	Dept_ID
Electronics	150	Miley Andrews	7	100
Software	250	Alex Dawson	6	101
Genetic	75	Victoria Fox	7	102
Biomedical	72	Sophia Williams	8	104

Chemical	75	Joshua Taylor	6	106
Electronics & Com	250	Michael Anderson	8	108

Next, we GROUP the data as shown below:

```
SELECT e.ENGG_NAME, d.HOD,d.NO_OF_Prof,d.Dept_ID
FROM ENGINEERING_STUDENTS e, Dept_Data d
WHERE d.NO_OF_Prof > 5 AND e.ENGG_ID = d.ENGG_ID
GROUP BY ENGG_NAME;
```

ENGG_NAME	STUDENT_ STRENGTH	HOD	NO_OF_Prof	Dept_ID
Biomedical	72	Sophia Williams	8	104
Chemical	75	Joshua Taylor	6	106
Electronics	150	Miley Andrews	7	100
Electronics & Com	250	Michael Anderson	8	108
Genetic	75	Victoria Fox	7	102

Software	250	Alex Dawson	6	101

Let's see which departments from this institution have greater than 100 students:

```
SELECT e. ENGG_NAME, e.STUDENT_STRENGTH,
d.HOD,d.NO_OF_Prof,d.Dept_ID
FROM ENGINEERING_STUDENTS e, Dept_Data d
WHERE d.NO_OF_Prof > 5 AND e.ENGG_ID = d.ENGG_ID
GROUP BY e.ENGG_NAME HAVING e.STUDENT_STRENGTH > 100;
```

ENGG_NAME	STUDENT_ STRENGTH	HOD	NO_ OF _Prof	Dept_ID
Electronics	150	Miley Andrews	7	100
Electronics & Com	250	Michael Anderson	8	108
Software	250	Alex Dawson	6	101

Evaluating a Condition

A WHERE clause can evaluate more than one situation in which each condition is separated by using the AND operator. Let's take a take a look at the instance below:

```
SELECT e. ENGG_NAME, e.STUDENT_STRENGTH,
```

d.HOD,d.NO_OF_Prof,d.Dept_ID

FROM ENGINEERING_STUDENTS e, Dept_Data d WHERE d.NO_OF_Prof > 5 AND

e.ENGG_ID = d.ENGG_ID AND

100 < e.STUDENT_STRENGTH < 250;

ENGG_NAME	STUDENT_ STRENGTH	HOD	NO_OF_ Prof	Dept_ ID
Electronics	150	Miley Andrews	7	100
Software	250	Alex Dawson	6	101
Genetic	75	Victoria Fox	7	102
Biomedical	72	Sophia Williams	8	104
Chemical	75	Joshua Taylor	6	106
Electronics & Com	250	Michael Anderson	8	108

There is one aspect you must recognize with the WHERE clause. It is most effective whilst all situations end up true for a row that it is included in the result set. If even one condition seems to be false, the row will no longer be included in the result set.

The end result set may be exceptional if you update any or all AND operators within the above declaration with the OR operator. Say you need to find out which department has either fewer than 100 college students OR fewer than 5 professors. Have a have a look at the following statement:

SELECT e. ENGG_NAME,e.STUDENT_STRENGTH,

d.HOD,d.NO_OF_Prof,d.Dept_ID,e.ENGG_ID

FROM ENGINEERING_STUDENTS e, Dept_Data d

WHERE e.ENGG_ID = d.ENGG_ID AND

(e.STUDENT_STRENGTH < 100 OR d.NO_OF_Prof < 5);

AND Operator

Condition	Outcome
Where True AND True	True
Where False AND True	False
Where True AND False	False
Where False AND False	False

OR Operator

Condition	Outcome
Where True OR True	True
Where False OR True	True
Where True OR False	True
Where False OR False	False

Usage of Parentheses

In the last example, we had three conditions, and we put one condition in parentheses. If the parentheses are missing, the outcomes could be incorrect and confusing. For a change, let's try and see what takes place if this occurs:

SELECT e.ENGG_NAME,e.STUDENT_STRENGTH,

d.HOD,d.NO_OF_Prof,d.Dept_ID,e.ENGG_ID

FROM ENGINEERING_STUDENTS e, DEPT_DATA d

WHERE e.ENGG_ID = d.ENGG_ID AND

e.STUDENT_STRENGTH < 100 OR d.NO_OF_Prof < 5;

ENGG_ NAME	STUDEN T_	HOD	N O _	Dept_ ID	ENGG_ ID

	STRENGTH		OF_Prof		
Genetic	75	Victoria Fox	7	102	3
Biomedical	72	Sophia Williams	8	104	5
Electronics	150	Olive Brown	4	105	1
Software	250	Olive Brown	4	105	2
Genetic	75	Olive Brown	4	105	3
Mechanical	150	Olive Brown	4	105	4
Biomedical	72	Olive Brown	4	105	5
Instrumentation	80	Olive Brown	4	105	6
Chemical	75	Olive Brown	4	105	7

Civil	60	Olive Brown	4	105	8
Electronics & Com	250	Olive Brown	4	105	9
Electrical	60	Olive Brown	4	105	10
Chemical	75	Joshua Taylor	6	106	7
Civil	60	Ethan Thomas	5	107	8
Electrical	60	Martin Jones	5	109	10

One look at the table above and you recognize the outcomes are all incorrect and misleading. The reason is that the instructions aren't clean to the statistics server. Now, reflect onconsideration on what we want to accomplish; we want to know which department has fewer than one hundred college students OR fewer than 5 professors. Now, witness the magic of the parentheses. The parentheses convert three conditions to two, well-defined conditions.

WHERE e.ENGG_ID = d.ENGG_ID AND

(e.STUDENT_STRENGTH < 100 OR d.NO_OF_Prof < 5);

This follows the following format: Condition1 AND (Condition2 OR Condition3).

Now, this is how the condition could be calculated:

Cond1	Cond2	Cond3	Cond2 OR Cond3	Cond1 AND (Cond2 OR Cond3)
T	T	T	T	T
T	T	F	T	T
T	F	T	T	T
T	F	F	F	F
F	T	T	T	F
F	T	F	T	F
F	F	T	T	F
F	F	F	F	F

By putting e.STUDENT_STRENGTH < 100 OR d.NO_OF_Prof < 5 into parentheses, we convert it into one situation, and the first circumstance will remember the output of the parentheses for the very last end result. Out of eight feasible situations from the table above, there are simplest three scenarios where the final condition is True.

The NOT Operator

To understand using the NOT operator, let's update AND with AND NOT and find out what output we receive:

```
SELECT e. ENGG_NAME,e.STUDENT_STRENGTH,
d.HOD,d.NO_OF_Prof,d.Dept_ID,e.ENGG_ID
FROM ENGINEERING_STUDENTS e, Dept_Data d
WHERE e.ENGG_ID = d.ENGG_ID AND NOT
(e.STUDENT_STRENGTH < 100 OR d.NO_OF_Prof < 5);
```

ENGG_ NAME	STUDENT_ STRENGTH	HOD	NO_ OF _Prof	Dept_ ID	ENGG_ ID
Electronics	150	Miley Andrews	7	100	1
Software	250	Alex Dawson	6	101	2
Mechanical	150	Anne Joseph	5	103	4
Electronics & Com	250	Michael Anderson	8	108	9

Remember that our desired effects discover which department has fewer than 100 students or fewer than five professors. After applying the NOT operator before the parentheses, our results look for both extra than five professors, or extra than a hundred students.

Is it feasible to gain the identical effects while the usage of the NOT assertion? Yes! Just update '>' with '

SELECT e.ENGG_NAME,e.STUDENT_STRENGTH,

d.HOD,d.NO_OF_Prof,d.Dept_ID,e.ENGG_ID

FROM ENGINEERING_STUDENTS e, Dept_Data d

WHERE e.ENGG_ID = d.ENGG_ID AND NOT (e.STUDENT_STRENGTH ≥ 100 AND d.NO_OF_Prof ≥ 5);

ENGG_	STUDENT	HOD	N	Dep	ENG

NAME	_ STRENGTH		O _ O F_ Prof	t_ ID	G_ ID
Genetic	75	Victoria Fox	7	102	3
Biomedical	72	Sophia Williams	8	104	5
Instrumentation	80	Olive Brown	4	105	6
Chemical	75	Joshua Taylor	6	106	7
Civil	60	Ethan Thomas	5	107	8
Electrical	60	Martin Jones	5	109	10

Here is the output for applying AND NOT:

Cond1	Cond2	Cond3	Cond2 AND Cond3	Cond1 AND NOT (Cond2 AND Cond3)
T	T	T	T	F

T	T	F	F	T
T	F	T	F	T
T	F	F	F	T
F	T	T	T	T
F	T	F	F	T
F	F	T	F	T
F	F	F	F	T

However, use the NOT operator best whilst required. It can aid in the legibility of the statement, however if you use NOT whilst it may be avoided, it is able to unnecessarily complicate matters for the developer.

Sequences

A series refers to a set of numbers that has been generated in a exact order on demand. These are popular in databases. The reason behind this is that sequences provide an easy way to have a completely unique fee for every row in a distinct column. This section explains the use of sequences in SQL.

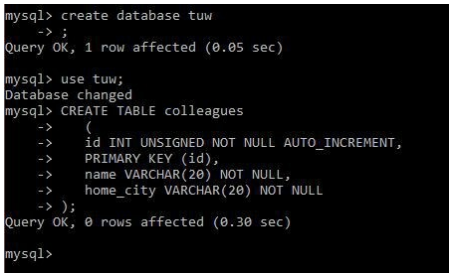
AUTO_INCREMENT Column

This provides you with the easiest manner of developing a chain in MySQL. You only have to outline the column as auto_increment and depart MySQL to take care of the rest. To show the way to use this belongings, we are able to create a simple desk and insert some statistics into the table.

The following command will assist us create the desk:

```
CREATE TABLE colleagues
(
id INT UNSIGNED NOT NULL AUTO_INCREMENT,
PRIMARY KEY (id),
name VARCHAR(20) NOT NULL,
home_city VARCHAR(20) NOT NULL
);
```

The command have to create the table effectively, as proven below:



We have created a table named colleagues. This table has 3 columns: identification, name, and home_city. The first column is an integer information kind even as the relaxation are varchars (variable characters). We have introduced the auto_increment property to the identity column, so the column values may be incremented routinely. When entering records into the desk, we don't need to specify the cost of this column. It will begin at 1 by default then increment the values robotically for each report you insert into the desk.

Let us now insert some statistics into the table:

INSERT INTO colleagues

VALUES (NULL, "John", "New Delhi");

INSERT INTO colleagues

VALUES (NULL, "Joel", "New Jersey");

INSERT INTO colleagues

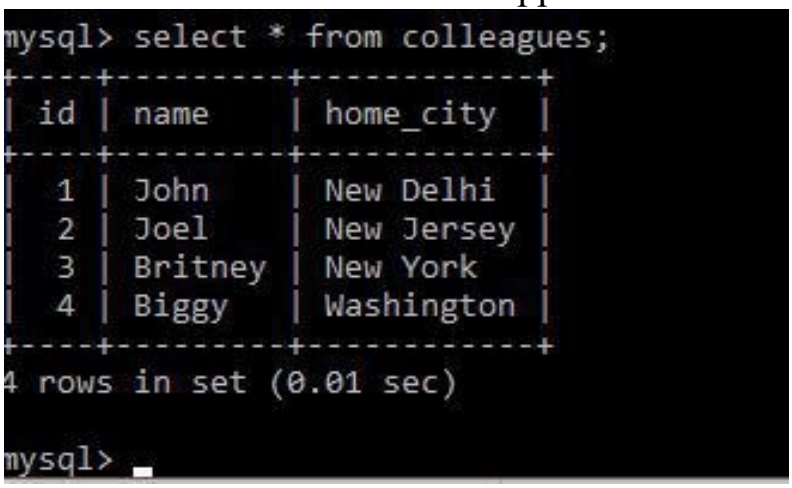
VALUES (NULL, "Britney", "New York");

INSERT INTO colleagues

VALUES (NULL, "Biggy", "Washington");

The instructions need to run efficaciously, as proven below:

Now, we can run the select statement in opposition to the desk and notice its contents:



We see that the identity column has additionally been populated with values starting from 1. Each time you enter a document, the fee of this column is increased by 1.

We have efficaciously created a series.

Renumbering a Sequence

You notice that while you delete a report from a sequence such as the only we've got created above, the data will not be renumbered. You won't be impressed via this sort of numbering. However, it's far possible so that you can resequence the facts. This only involves a single trick, but make sure to check whether or not the desk has a join with another table or not.

If you find you have to re-series your data, the great manner to do it's far by way of losing the column after which including it. Here, we'll display how to drop the identity column of the colleagues desk.

The desk is as follows for now:

```
mysql> select * from colleagues;
+----+---------+------------+
| id | name    | home_city  |
+----+---------+------------+
|  1 | John    | New Delhi  |
|  2 | Joel    | New Jersey |
|  3 | Britney | New York   |
|  4 | Biggy   | Washington |
+----+---------+------------+
4 rows in set (0.01 sec)

mysql>
```

Let us drop the identification column by means of strolling the subsequent command: ALTER TABLE colleagues DROP identification;

```
mysql> ALTER TABLE colleagues DROP id;
Query OK, 4 rows affected (0.40 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql>
```

To affirm whether the deletion has taken place, let's take a look at the table records:

```
mysql> select * from colleagues;
+---------+------------+
| name    | home_city  |
+---------+------------+
| John    | New Delhi  |
| Joel    | New Jersey |
| Britney | New York   |
| Biggy   | Washington |
+---------+------------+
4 rows in set (0.00 sec)

mysql>
```

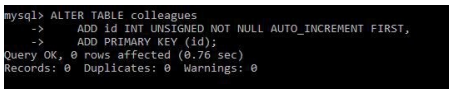
The deletion became successful. We combined the ALTER TABLE and the DROP instructions for the deletion of the column. Now, allow us to re-add the column to the table:

ALTER TABLE colleagues

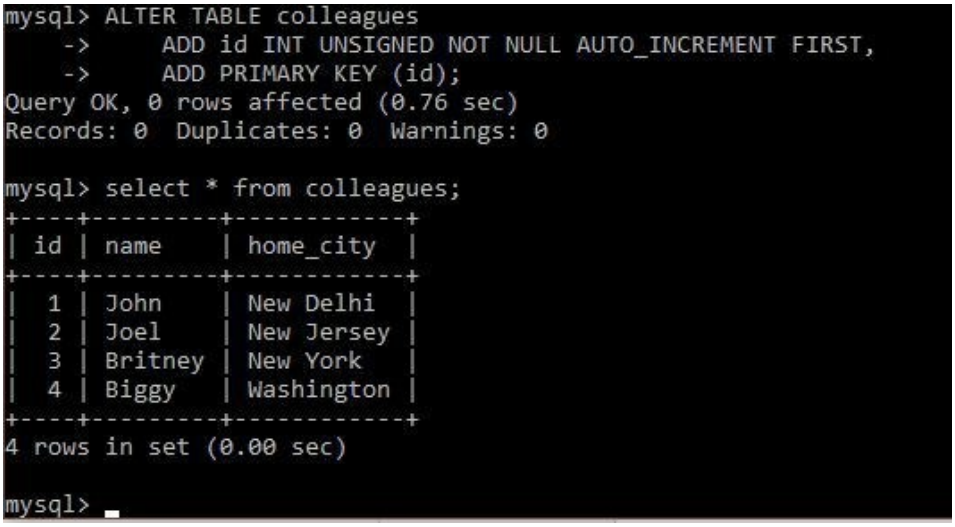
ADD id INT UNSIGNED NOT NULL AUTO_INCREMENT FIRST,

ADD PRIMARY KEY (id);

The command should run as follows:



We started out with the ALTER TABLE command to specify the call of the table we want to change. The ADD command has then been used to feature the column and set it as the primary key for the table. We have also used the auto_increment property inside the column definition. We can now question the desk to look what has happened:

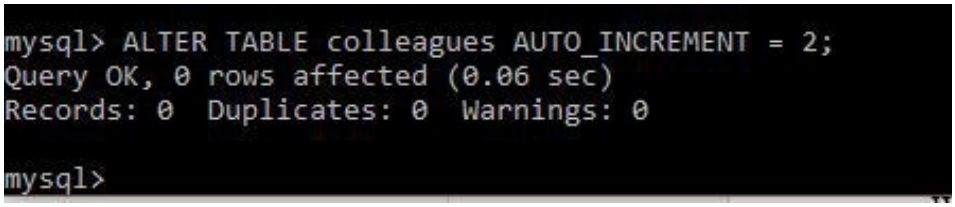


The id column became added effectively. The sequence has additionally been numbered correctly.

MySQL starts offevolved the sequence at index 1 by using default. However, it's miles possible so that it will customise this when you are developing the desk. You can set the restrict or amount of the increment every time a document is created. Like within the desk named colleagues, we will regulate the desk for the auto_increment to be carried out at durations of 2. This is achieved via the code below:

ALTER TABLE colleagues AUTO_INCREMENT = 2;

The command need to run successfully, as proven below:



We can specify wherein the auto_increment will start on the time of the introduction of the table. The following example shows this:

```
CREATE TABLE colleagues2
(
id INT UNSIGNED NOT NULL AUTO_INCREMENT = 10, PRIMARY KEY (id),
name VARCHAR(20) NOT NULL, home_city VARCHAR(20) NOT NULL
);
```

From the above instance, we set the auto_increment belongings on the identity column, and the initial price for the column will be 10.

SQL Subqueries

A sub-query refers to a question embedded interior every other query, and that is done within the WHERE clause. It is also known as an Inner question or Nested query.

We use a sub-query to go back the statistics so one can be used in our fundamental query and as a circumstance for proscribing the statistics we are to retrieve. We can add subqueries to SELECT, UPDATE, INSERT, and DELETE statements and integrate them with operators like =, =, IN, BETWEEN and many others.

The following are the rules that govern the usage of sub-queries:

· A sub-query need to be added within parenthesis.

· A sub-question should go back most effective one column, which means that the SELECT * cannot be used inside a sub-query unless where the table has handiest an unmarried column. If your goal is to carry out row comparison, you can create a subquery to be able to go back a couple of columns.

· One can only create sub-queries that return over one row with multiple cost operators like IN and NOT IN operators.

· You can't use a UNION in a sub-query. You are most effective allowed to apply one SELECT assertion.

· Your SELECT list must now not include a connection with values trying out to a

BLOB, CLOB, ARRAY, or NCLOB.

· You cannot right away enclose a sub-query within a hard and fast function.

· You can't use the BETWEEN operator with a sub-query. However, you can use this operator inside a sub-question.

Subqueries with SELECT Statement

In maximum cases, we use subqueries with the SQL's SELECT announcement.

We do that the usage of the syntax given underneath:

SELECT columnName [, columnName]

```
FROM   table_1 [, table_2 ]
WHERE  columnName OPERATOR
(SELECT columnName  [,columnName ]
FROM table_1 [, table_2 ]
[WHERE])
```

Consider the college students table with the statistics given underneath:

Let us now create a query with a sub-query as shown below:

```
SELECT *
FROM students
WHERE regno IN (SELECT regno
FROM students
WHERE age >= 19);
```

The command should return the result given below:

We have used the following subquery:

```
SELECT regno
FROM students
WHERE age >= 19;
```

In the principle query, we are the use of the consequences of the above subquery to go back our very last effects. This method that the main question will only return the information of college students whose age is nineteen and above. That is what the above output suggests.

Subqueries with INSERT Statement

We can also use Sub-queries with the INSERT assertion. What happens, in this case, is that the insert declaration will use the records that has been returned with the aid of the question to insert it into every other desk. It is feasible for us to trade the facts that the sub-query returns the usage of date, person or variety functions.

The following syntax shows how we can upload a sub-question to the INSERT declaration:

INSERT INTO tableName [(column_1 [, column_2])]

SELECT [*|column_1 [, column_2]

FROM table_1 [, table_2]

[WHERE A VALUE OPERATOR]

Consider a scenario in which we have the desk students2 that's a precise replica of the college students' desk. First, let us create this desk:

CREATE TABLE students2 LIKE college students;

The table doesn’t have even a unmarried record. We want to populate it with facts from the students' desk. We then run the following query to perform this task:

INSERT INTO students2

SELECT * FROM students

WHERE regno IN (SELECT regno

FROM students);

The code ought to run effectively as shown underneath:

We have used the regno column of the desk named college students to pick all the facts in that desk and populate them into the desk named students2. Let us question the desk to peer its contents:

It is clear that the desk turned into populated successfully.

Subqueries with UPDATE Statement

We can combine the replace announcement with a sub-question. With this, we will replace either unmarried or more than one columns in a table.

We can do that the usage of the following syntax:

UPDATE table

SET columnName = newValue

[WHERE OPERATOR [VALUE]

(SELECT COLUMNNAME

FROM TABLENAME)

[WHERE)]

We need to reveal the use of the college students' desk. The table has the information given below:

The table students2 is a duplicate of the college students' desk. We need to

boom the ages of the students by way of 1. This method we run the following command:

UPDATE students

SET age= age + 1

WHERE regno IN (SELECT regno FROM students2); The command need to run successfully as shown below:

Since the 2 tables have precisely the same statistics, all the facts within the college students' desk were matched to the sub-question. This way that all the facts had been updated. To affirm this, we run a choose announcement in opposition to the college students' desk:

The above figure indicates that the values of age had been increased by 1.

Subqueries with DELETE Statement

We can use sub-queries with the DELETE assertion much like the opposite SQL statements. The following syntax indicates how we are able to do that:

DELETE FROM TABLENAME

[WHERE OPERATOR [VALUE]

(SELECT COLUMNNAME

FROM TABLENAME)

[WHERE)]

Suppose we want to make a deletion on the students table. The students desk has the following facts:

The students2 table is an precise copy of the above table as shown below:

We want to carry out a deletion on the students table by using going for walks the following command:

DELETE FROM students

WHERE AGE IN (SELECT age FROM students2

WHERE AGE < 20);

The command ought to run correctly as shown under:

The document wherein the value of age is under 20 can be deleted. This is because this is what the subqueries go back. We can verify this via going for walks a choose statement on the students' table as shown beneath:

The above output suggests that the file changed into deleted efficiently.

Database Components

Now that you know more about a database's use inside the real global and why you may need to learn SQL, we will dive into the additives of the database.

These components or objects inside a database are usually mentioned as "items". These items can range from tables, to columns, to indexes and even triggers. Essentially, these are all of the pieces of the facts puzzle that make up the database itself.

The examples and screenshots used in the following sections are from the AdventureWorks2016 database, which you'll be running with later on in this book.

Database Tables

Within the database are tables, which preserve the information itself. A table consists of columns that are the headers of the table, like First_Name and Last_Name, for example. There are also rows, that are taken into consideration an entry in the table. The point to wherein the column and row intersect is called a mobile.

The mobile is where the facts are shown, like someone's actual first call and last call. Some cells won't continually have data, which is taken into consideration NULL in the database. This just manner that no statistics exists in that cell.

In Microsoft's SQL Server, a desk may have as much as 1,024 columns, however could have any wide variety of rows.

Schemas

A schema is taken into consideration a logical container for the tables. It's essentially, a way to institution tables together based totally on the kind of statistics that they hold. It won't have an effect on a stop person who interacts with the database, like someone who runs reports. But one that works directly with the database, like a database administrator or a developer, will see the available schemas.

Consider a sensible instance of several tables containing data for Sales records. There can be several tables named Sales_Order, Sales_History or

Sales_Customer. You can have placed all of these Sales tables into a “Sales” schema to higher discover them while you work immediately with the database.

Columns

Remember that you can handiest have as much as 1,024 columns in a given table in SQL Server!

Rows and NULL values

A row is considered an access in a desk. The row could be one line across, and normally have statistics within each column. Though, in a few cases, there may be a NULL value in a single or many cells.

Back to our example of names, most human beings have first and last names, but no longer every person has a middle name. In that case, a row could have values in the first and closing name columns, but not the center call column, like shown underneath.

Primary Keys

A number one secret is a constraint on a column that forces every value in that column to be precise. By forcing area of expertise on values in that column, it helps preserve the integrity of the records and helps prevent any future facts issues.

A realistic instance of a primary key would be an employee ID or a sales file ID. You wouldn’t want to have two of the equal employee ID’s for two different humans, nor could you need to have two or greater of the same income file ID’s for different income transactions. That would be a nightmare while seeking to save and retrieve records!

You can see within the underneath instance that every value for BusinessEntityID is unique for each person.

Foreign Keys

Another key much like the primary key is a foreign key. These fluctuate from primary keys by way of no longer usually being specific and act as a hyperlink between two or extra tables.

Below is an instance of an overseas key that exists inside the AdventureWorks2016 database. The overseas secret's ProductID on this desk

(Sales.SalesOrderDetail):

The ProductID inside the above table is linking to the ProductID (primary key) in the Production.Product desk:

Essentially, overseas keys will check its link to the other table to see if that value exists. If now not, then you definitely will turn out to be receiving an error when looking to insert records into the table where the overseas key is.

Constraints

Primary keys and foreign keys are referred to as constraints within the database.

Constraints are “rules” which are set in location as some distance as the kinds of information that can be entered. There are numerous others which are used aside from primary keys and foreign keys that help hold the integrity of the facts.

UNIQUE – enforces all values in that column to be different. An example of this may be implemented to the Production.Product table. Each product have to be different, since you wouldn’t need to store the equal product call a couple of times.

NOT NULL – guarantees that no cost in that column is NULL. This may also be applied to the equal table as above. In this case, the ProductNumber can not have a NULL price, as each Product have to have its personal corresponding ProductNumber.

DEFAULT – sets a default fee in a column whilst a cost isn't always provided. A exceptional example of this would be the ListPrice column. When a fee isn’t specified while being added to this table, the price will default to 0.00. If this cost had been to be calculated in another table and be a NULL value (like a income desk where income from the employer are made), then it would be impossible to calculate primarily based on a NULL price on the grounds that it’s not a variety. Using a default value of zero.00 is a better approach.

INDEXES – Indexes are constraints that are created on a column that hurries up the retrieval of records. An index will essentially compile all the values in a column and treat them as specific values, even supposing they’re now

not. By treating them as precise values, it allows the database engine to improve its search based on that column.

Indexes are pleasant used on columns that:

Do not have a completely unique constraint

Are now not a primary key

Or aren't a overseas key

The reason for not applying an index to a column that satisfies any of the above three conditions, is that those are naturally quicker for retrieving information due to the fact that they are constraints.

As an instance, an index might be fine used on something like a date column in a sales table. You may be filtering certain transaction dates from January thru March as part of your quarterly reports, but see many purchases on the same day between the ones months. By treating it as a completely unique column, even the equal or comparable values can nevertheless be observed much faster by means of the database engine.

Views

A view is a virtual desk that’s constructed from one or extra columns from one or more tables. It is created the usage of a SQL question and the authentic code used to create the view is recompiled while a user queries that table.

In addition, any updates to records made inside the originating tables (i.e. The tables and columns that make up the view) might be pulled into the view to show current records. This is another motive that perspectives are top notch for reporting functions, as you can pull actual-time information without touching the foundation tables.

For nice practices, DO NOT replace any records in a view. If you want to update the statistics for any reason, perform that within the originating table(s).

To extend a little bit on why a view would be used is the following:

- To hide the raw factors of the database from the quit-person in order that they handiest see what they want to. You can also make it more cryptic for the cease-user.

- An alternative to queries that are regularly run within the database, like reporting purposes as an example.

These are only some motives as to why you will use a view. However, depending in your situation, there could be other reasons why you would use a view as opposed to writing a query to immediately obtain facts from one or more tables.

To better illustrate the concept of a view, the underneath instance has two tables: ‘People’ and ‘Locations’. These tables are combined right into a view that is called ‘People and Locations’ only for simplicity. These are also joined on a commonplace field, i.e. The LocationID.

Stored Procedures

Stored procedures are pre-compiled SQL syntax that may be used again and again once more by using executing its name in SQL Server. If there’s a positive query which you’re jogging regularly and writing it from scratch or saving the document someplace and then opening it if you want to run it, then it can be time to take into account growing a stored procedure out of that query.

Just like with SQL syntax that you’d write from scratch and passing in a fee in your WHERE clause, you may do the equal with a stored system. You have the ability to pass in sure values to obtain the end result which you’re looking for. Though, you don’t usually have to skip a parameter right into a stored system.

As an example, let’s say that as part of the HR branch, you should run a query once a month to verify which personnel are income and non-salary, in compliance with labor legal guidelines and organization policy.

Instead of beginning a record often or writing the code from scratch, you could certainly name the stored procedure which you saved within the database, to retrieve the records for you. You might just specify the proper price (in which 1 is TRUE and 0 is FALSE on this case).

EXEC HumanResources.SalariedEmployees @SalariedFlag = 1 In the end result set below, you can see some of the employees who work in a profits type position:

Triggers

A cause within the database is a stored manner (pre-compiled code) so that it will execute when a certain occasion occurs to a desk. Generally, these triggers will fire off whilst data is brought, up to date or deleted from a desk.

Below is an example of a cause that prints a message when a new branch is created inside the HumanResources.Department desk. --Creates a notification declaring that a new branch has been created

--when an INSERT statement is executed against the Department table

CREATE TRIGGER NewDepartment

ON HumanResources.Department

AFTER INSERT

AS RAISERROR ('A new department has been created.', 10, 9)

To increase on this a little more, you specify the name of your trigger after CREATE TRIGGER. After ON, you'll specify the table name that this is associated with.

Next, you can specify which sort of movement will hearth this cause (you could also use UPDATE and/or DELETE), that is called a DML trigger in this case.

Last, I'm printing a message that a new branch has been created and using some variety codes in SQL Server for configuration.

To see this cause inside the works, here's the INSERT statement I'm using to create a brand new department. There are 4 columns in this desk, DepartmentID, Name, GroupName and ModifiedDate. I'm skipping the DepartmentID column within the INSERT assertion due to the fact a new ID is mechanically generated through the database engine.

--Adding a new department to the Department's table

INSERT INTO HumanResources.Department

(Name, GroupName, ModifiedDate)

VALUES

('Business Analysis', 'Research and Development', GETDATE()) - GETDATE() gets the current date and time, depending on the data type being used in the table

The trigger will prompt a message after the new record has been successfully inserted.

A new department has been created.

(1 row(s) affected)

If I have been to run a query against this table, I can see that my department became effectively introduced as well.

Deadlocks in SQL

In most of the cases, a couple of users get entry to database programs simultaneously, because of this that multiple transactions are being finished on database in parallel. By default, when a transaction plays an operation on a database aid inclusive of a desk, it locks the resource. During that period, no different transaction can access the locked resource. Deadlocks occur when or greater than strategies try to get entry to resources which can be locked by means of the alternative techniques participating inside the impasse.

Deadlocks are satisfactory defined with the assist of an example. Consider a state of affairs where a few transactionA has done an operation on tableA and has obtained lock at the desk. Similarly, there's any other transaction named transactionB this is executing in parallel and plays a few operations on tableB. Now, transactionA desires to perform a few operations on table which is already locked via transactionB. Similarly, transactionB wants to carry out an operation on tableA, however it is already locked through transactionA. This effects in an impasse in view that transactionA is ready on an aid locked via transactionB, that's ready on an aid locked by transactionA. In this bankruptcy we will see a realistic instance of deadlocks. Then we are able to see how we can analyze and clear up deadlocks.

Dummy Data Creation

For the sake of this bankruptcy, we will create a dummy database. This database may be used inside the impasse instance that we will in next section. Execute the following script

```
CREATE DATABASE dldb;

GO

USE dldb;
```

```
CREATE TABLE tableA
(
id INT IDENTITY PRIMARY KEY, patient_name NVARCHAR(50)
)
INSERT INTO tableA VALUES ('Thomas')
CREATE TABLE tableB
(
id INT IDENTITY PRIMARY KEY, patient_name NVARCHAR(50)
)
INSERT INTO table2 VALUES ('Helene')
```

The above script creates database named “dldb”. In the database we create tables: tableA and tableB. We then insert one record each inside the both tables.

Practical Example of Deadlock

Let’s write a script that creates impasse. Open instances of SQL server management studio. To simulate simultaneous facts get right of entry to, we will run our queries in parallel in these two instances.

Now, open the first times of SSMS, and write the following script. Do now not execute this script on the moment.

Instance1 Script

```
USE dldb;
BEGIN TRANSACTION transactionA
-- First update statement
UPDATE tableA SET patient_name = 'Thomas - TransactionA'
WHERE id  = 1
-- Go to the second instance and execute
-- first update statement
UPDATE tableB SET patient_name = 'Helene - TransactionA'
```

```
WHERE id = 1
-- Go to the second instance and execute
-- second update statement
COMMIT TRANSACTION
```

In the second one instance, reproduction and paste the following script. Again, do no longer run the Script. Instance 2 Script

```
USE dldb;
BEGIN TRANSACTION transactionB
-- First update statement
UPDATE tableB SET patient_name = 'Helene - TransactionB'
WHERE id  = 1
-- Go to the first instance and execute
-- second update statement
UPDATE tableA SET patient_name = 'Thomas - TransactionB'
WHERE id = 1
COMMIT TRANSACTION
```

Now we've got our scripts ready in both the transaction.

Open both the instances of SSMS side by way of aspect as proven inside the following figure:

To create a deadlock we should comply with little by little approach. Go to the first instance of SQL Server control studio(SSMS) and execute the following strains from the script:

```
USE dldb;
BEGIN TRANSACTION transactionA
-- First update statement
UPDATE tableA SET patient_name = 'Thomas - TransactionA' WHERE id
= 1
```

In the above script, transactionA updates the tableAby setting the name of the patient with id one to'Thomas –TransactionA'. At this point of time,

transactionA acquires lock on tableA.

Now, execute the following script from the second one instance of SSMS.

```
USE dldb;

BEGIN TRANSACTION transactionB

-- First update statement

UPDATE tableB SET patient_name = 'Helene - TransactionB' WHERE id = 1
```

The above script executes transactionB which updates tableB with the aid of putting the name of affected person with id one to'Helene – TransactionB', obtaining lock on tableB.

Now come back once more to first instance of SSMS. Execute the subsequent piece of script:

```
UPDATE tableB SET patient_name = 'Helene - TransactionA'

WHERE id = 1
```

Here transactionA tries to replace tableB that's locked via transactionB. Hence transactionA goes to ready state.

Go to the second instance of SSMS again and execute the following piece of script.

```
UPDATE tableA SET patient_name = 'Thomas - TransactionB'

WHERE id = 1
```

In the above script, transactionB tries to replace tableA which is locked by using transactionA. Hence transactionA also is going to ready state.

At this point of time, transactionA is awaiting a aid locked with the aid of transactionB. Similarly, transactionB is expecting the aid locked via TransactionA. Hence impasse happens here.

By default, SSMS selects one of the transactions concerned within the impasse as deadlock victim. The transaction decided on as deadlock sufferer is rolled back, permitting the opposite transaction to complete its execution. You will see that when few second, the transaction in one of the instances will entire its execution while an errors will appear in different instance.

In the example that we just saw, transactionA was allowed to finish its execution even as transactionB turned into decided on as impasse victim. Your result can be different. This is proven within the following figure:

You can see the message “1 row affected” in the instance at the left this is running transactionA. On the other hand within the left instance an mistakes is displayed that reads:

Msg 1205, Level 13, State 51, Line 12

Transaction (Process ID 54) became deadlocked on lock resources with some other method and has been selected as the impasse victim. Rerun the transaction.

The mistakes says that the Transaction with manner ID 54 turned into worried in a deadlock and therefore chosen as sufferer of the deadlock.

Deadlock Analysis and Prevention

In the previous section we generated impasse ourselves, therefore we've got statistics about the approaches worried within the deadlock. In the actual global scenarios, this is not the case. Multiple users get admission to the database simultaneously, which regularly outcomes in deadlocks. However, in such instances we cannot inform which transactions and resources are involved inside the deadlock. We want a mechanism that permits us to analyze deadlocks in detail so that we can see what transactions and sources are worried and decide how to resolve the deadlocks. One such ways is thru SQL Server errors logs.

Reading Deadlock information through SQL Server Error Log

The SQL Server provides simplest little information about the deadlock. You can get detailed records approximately the impasse via SQL error log. However to log impasse information to blunders log, first you need to use a trace flag 1222. You can turn trace flag 1222 on worldwide in addition to consultation degree. To turn on hint flag 1222 on, execute the following script:

DBCC Traceon(1222, -1)

The above script turns trace flag on global degree. If you do now not pass the second one argument, the trace flag is turned on session degree. To see if

trace flag is sincerely became on, execute the subsequent query: DBCC TraceStatus(1222)

The above announcement results inside the following output:

TraceFlag Status Global Session

1222 1 1 0

Here status fee 1 suggests that trace flag 1222 is on. The 1 for Global column implies that hint flag has been became on globally.

Now, try and generate a deadlock via following the steps that we achieved inside the ultimate segment. The detailed deadlock facts will be logged in the error log. To view sq. server blunders log, you need to execute the following stored manner.

executesp_readerrorlog

The above stored manner will retrieve detailed mistakes log a snippet of which is shown below:

Your blunders log might be different depending upon the databases to your database. The statistics approximately all of the deadlocks to your database starts with log text “deadlock-list”. You may need to scroll down a chunk to find this row.

The deadlock statistics logged with the aid of the SQL server blunders log has three main parts.

- The deadlock Victim
- Process List

The procedure list is the list of all the strategies worried in a deadlock. In the impasse that we generated, procedures had been involved. In the tactics listing you can see details of each of these methods. The id of the first process is highlighted in red whereas the id of the second technique is highlighted in green. Notice that inside the technique list, the first process is the procedure that has been selected as impasse sufferer too.

Apart from manner id, there you may also see different data approximately the procedures. For instance, you may discover login statistics of the manner, the isolation degree of the technique etc. You can see the script that the process was trying to run. For instance if you have a look at the first manner

in the method list, you may find that it changed into seeking to update the patient_name column of the table tableA, while the impasse occurred. 3- Resource List

The aid listing contains data approximately the sources that have been involved in the deadlock. In our instance, tableA and tableB had been the best two resources worried inside the deadlock. You can both of those tables highlighted in blue within the aid listing of the log within the desk above.

Some hints for Deadlock Avoidance

From the mistake log we can get detailed records approximately the impasse. However we will minimize the chance of impasse occurrence if we follow these pointers:

- Execute transactions in a unmarried batch and keep them short
- Release assets routinely after a positive time period
- Sequential useful resource sharing
- Not permitting person to have interaction with the utility when transactions are being executed.

This bankruptcy provided a brief assessment to deadlocks. In the following bankruptcy, we shall see some other extremely useful concept, i.e. Cursors.

Managing Database Objects

This section will speak database gadgets: their nature, behaviors, garage requirements, and interrelatedness. Basically, databases objects are the spine of relational databases. You use these objects to save records (i.e. they may be logical units located interior a database). For this reason, these gadgets are also referred to as back-cease databases.

What is a Database Object?

Database objects are the defined gadgets inside a database utilized to store or retrieve facts. Here are numerous examples of database items: views, clusters, tables, indexes, synonyms, and sequences.

The Schema

A schema is a hard and fast of database gadgets linked to a certain database user. This user, known as the "schema owner," owns the set of gadgets related to his/her username. Simply put, any person who generates an item has simply generated his/her personal schema. That approach customers have manipulate over database items that are generated, deleted, and manipulated.

Let's expect which you obtained login credentials (i.e. Username and password) from a database administrator. The username is PERSON1. Let's say you accessed the database and created a table named

EMPLOYEES_TBL. In the database's information, the file's actual name is

PERSON1.EMPLOYEES_TBL. The desk's "schema call" is PERSON1, which is likewise the creator/proprietor of that table.

When getting access to a schema which you very own, you aren't required to apply the schema name. That approach you have two approaches of accessing the imaginary report given above. These are:

PERSON1.EMPLOYEES_TBL EMPLOYEES_TBL

As you may see, the second alternative entails fewer characters. This is the purpose why schema owners select this method of having access to their files. If other customers need to view the document, however, they should consist of the schema in their database query.

The screenshot below indicates schemas within a database.

Tables – The Main Tool for Storing Data

Modern database customers bear in mind tables as the principle storage device. In general, a table is formed via row/s and column/s. Tables take up space within a database and can be temporary or permanent.

Fields/Columns

Fields, referred to as columns when running with a relational database, are components of a table wherein a particular statistics type is assigned to. You need to call a discipline so that it fits the statistics type it will be used with. You might also specify fields as NULL (i.e. Nothing should be entered) or NOT NULL (i.e. something wishes to be entered).

Each table need to have as a minimum one subject. Fields are the factors inner a table that store certain kinds of statistics (e.G. Names, addresses, cellphone numbers, etc.). For instance, you'll locate a "client name" column whilst checking a database desk for client records.

Rows

Rows are records of statistics within a table. For instance, a row in a client database table would possibly hold the name, fax variety, and identity number of a certain customer. Rows are composed of fields that preserve statistics from a single file inside the desk.

SQL Statement – CREATE TABLE

"CREATE TABLE" is an SQL assertion used to generate a table. Even though you may create tables quickly and easily, you have to spend time and effort in planning the structures of your new table. That means you have to do some research and making plans before issuing this SQL announcement.

Here are a number of the questions you have to answer while creating tables:

What kind of statistics am I running on?

What name must I select for this table?

What column will form the principle key?

What names have to be assigned to the fields/columns?

What type of facts may be assigned to those columns?

Which columns may be empty?

What is the most period for each column?

Once you've got replied those questions, using the CREATE TABLE command becomes simple.

Here’s the syntax to generate a new desk:

The final man or woman of that declaration is a semicolon. Almost all SQL implementations use positive characters to terminate statements or put up statements to the server. MySQL and Oracle use semicolons to perform those functions. Transact-SQL, on the opposite hand, makes use of the “GO” command. To make this eBook consistent, statements can be terminated or submitted using a semicolon.

The STORAGE Clause

Some SQL implementations offer STORAGE clauses. These clauses assist you in assigning the table sizes. That way you can use them whilst growing tables. MySQL uses the subsequent syntax for its STORAGE clause:

The Naming Conventions

When naming database gadgets, specifically columns and tables, you must choose names that mirror the statistics they may be used for. For instance, you could use the call EMPLOYEES_TBL for a table used to maintain worker statistics. You want to name columns using the equal principle. A column

used to save the phone range of personnel may be named

PHONE_NUMBER.

SQL Command - ALTER TABLE

You can use ALTER TABLE, an effective SQL command, to modify current database tables. You may add fields, do away with columns, alternate subject definitions, consist of or exclude constraints, and, in sure SQL implementations, alternate the table’s STORAGE values. Here’s the syntax for this command:

Altering the Elements of a Database Table

A column’s “attributes” discuss with the legal guidelines and behaviors of statistics interior that column. You may also alternate a column’s attributes using ALTER TABLE. Here, the term “attributes” refers to:

The type of facts assigned to a column.

The scale, period, or precision of a column.

Whether you could input NULL values into a column.

In the screenshot below, ALTER TABLE is used at the EMPLOYEE_TBL to exchange the attributes of a column named EMP_ID:

If you are using MySQL, you'll get the following statement:

Adding Columns to a Database Table

You ought to recollect positive regulations while including columns to present database tables. One of the rules is this: You can't upload a NOT NULL column if the table has information in it. Basically, you ought to use NOT NULL to suggest that the column ought to preserve some price for every information row within the desk. If you'll upload a NOT NULL column, you will go in opposition to this new constraint if the current facts rows don't have precise values for the added column.

Modifying Fields/Columns

Here are the rules you ought to follow whilst altering existing columns:

You can always increase a column's period.

You can lower a column's length only if the highest fee for the column is decrease than or same to the desired length. You can always boom the quantity of digits for numeric data sorts.

You can simplest lower the quantity of digits for numeric records sorts if the fee of the largest amount of digits inside the column is decrease than or same to the preferred amount of digits.

You can growth or decrease the amount of decimal places for numeric records types.

You can easily alternate the records sort of any column.

Important Note: Be extremely cautious whilst changing or losing tables. You would possibly lose valuable facts if you'll commit typing or logical mistakes whilst executing these SQL statements. How to Create New Tables from Existing Ones

You may additionally replica a present desk the use of those SQL statements: (1)

CREATE TABLE and (2) SELECT. After executing these statements, you'll get a new desk whose column definitions are same to that of the vintage one. This feature is customizable: you may copy all of the columns or simply those you need. The columns generated the use of this pair of statements will assume the size needed to save the facts. Here's the main syntax for generating a table from a current one:

This syntax entails a new keyword (i.e. SELECT). This keyword permits you to carry out database queries. In cutting-edge database systems, SELECT can help you generate tables the usage of search results.

How to Drop Tables

You can drop tables easily. If you used the RESTRICT statement and referenced the table using a view/constraint, the DROP command will give you an error message. If you used CASCASE, however, DROP will be triumphant and all constraints and/or views can be dropped. The syntax for losing a desk is:

Important Note: When dropping a database table, specify the owner or schema name of the desk you are running on. This is essential since losing the wrong table can end result to loss of statistics. If you may access multiple database accounts, make sure that you are logged in to the right account prior to losing any desk.

The Integrity Constraints

You can use integrity constraints to make certain the consistency and accuracy of records inside a database. In general, database users take care of integrity concerns thru a concept known as "Referential Integrity." In this phase, you'll study the integrity constraints that you could use in SQL.

Primary Key

A number one key is used to decide columns that make information rows unique. You can form primary keys the usage of one or extra columns. For instance, either the product's name or an assigned reference number can serve as a primary key for a product table. The purpose is to provide every document with a unique detail or primary key. In general, you can assign a number one key during table creation.

In the instance below, the table’s number one secret's the column named EMP_ID.

You can assign number one keys this way whilst growing a brand new desk. In this example, the desk’s number one secret is an implicit condition. As an alternative, you may specify number one keys as explicit situations at the same time as creating a table. Here’s an example:

- In the instance given above, the primary key's given after the comma list.
- If you want to form a number one key the use of multiple columns, you operate this approach:

Unique Column Constraint

Unique column constraints are just like number one keys: the column ought to have a completely unique value for each row. While you want to vicinity a primary key in a single column, you can area specific constraints on one of a kind column. Here’s an example:

In the instance above, EMP_ID serves as the primary key. That manner the column for worker identification numbers is getting used to assure the distinctiveness of each report. Users often reference primary key columns for database queries, especially while merging tables. The EMP_PHONE column has a completely unique cost, because of this each employee has a unique smartphone number.

Foreign Key

You can use this key at the same time as working on figure and baby tables. Foreign keys are columns in an infant desk that points to a number one key in the parent table. This kind of key serves as the number one tool in enforcing referential integrity within a database. You can also use a foreign key column to reference a primary key from a one of a kind desk.

In the instance below, you’ll discover ways to create an overseas key:

Here, EMP_ID serves as an overseas key for a desk named

EMPLOYEE_PAY_TBL. This key factors to the EMP_ID section of another desk (i.e. The EMPLOYEE_TBL desk). With this key, the database administrator can ensure that each EMP_ID within the

EMPLOYEE_PAY_TBL has a corresponding access in EMPLOYEE_TBL.

SQL practitioners call this the “figure/child relationship.”

Study the subsequent figure. This will assist you to understand the connection between baby tables and discern tables.

How to Drop Constraints

You can use the option “DROP CONSTAINT” to drop the constraints (e.G. primary key, overseas key, particular column, etc.) you applied to your tables. For instance, if you need to cast off the primary key in a desk named “EMPLOYEES”, you may use this command:

Some SQL implementations provide shortcuts for removing constraints. Oracle, for instance, makes use of this command to drop a number one key constraint:

On the alternative hand, sure SQL implementations permit users to deactivate constraints. Rather than dropping constraints permanently, you may disable them temporarily. This way, you can reactivate the constraints you'll need inside the future.

Conclusion

Thanks once more for getting ***SQL for Beginners*** .

SQL is one of the most commonplace programming languages used within the web improvement area today. It is a general-purpose, high-level, interpreted programming language. It is designed in a manner that it focuses on code readability through the great whitespace. Python programming makes use of an object-oriented and language constructs that concentrate on supporting programmers write logical and clean codes. Programmers can write code for both small and large–scale projects.

SQL is typed dynamically and rubbish-collected. It offers help for more than one paradigms in programming. Such models include practical, procedural, and object-orientated programming. Python language is generally defined as a battery that covered programming language. It is due to its considerable popular library.

SQL is taken into consideration as a multi-paradigm programming language. Structured and object-oriented programming are completely supported through SQL's interface. These are including meta-items and metaprogramming. Other programming paradigms are supported thru extensions which include good judgment and design by using settlement programming.

SQL uses dynamic typing combined with reference counting and garbage collectors that are cycle-detecting. They work in this way for the efficient management of memory. SQL is likewise characterized by way of dynamic call resolution, also referred to as past due binding. The name resolution facilitates in binding variable names and techniques during the execution of programs. Besides, SQL design also lets in for the guide of practical programming within the Lisp tradition.