Introduction to SQL

History

- SQL was initially developed at IBM by Donald D. Chamberlin and Raymond F. Boyce in the early 1970s. The programming language was named as SEQUEL at the time when it was developed

- In June 1979, Relational Software Inc. introduced the first commercially available implementation of SQL, Oracle V2 (Version2) for VAX computers

Basic Concepts

- SQL stands for Structured Query Language
- SQL is a standard language for accessing and manipulating databases
- The user can issue a **query** when he/she wants to retirieve information from a database
- A query is a user-generated request to retrieve data or set of information with a certain condition

Categories of SQL

SQL functions fit into three broad categories:

- **Data definition language (DDL)** which deals with database schemas and descriptions. It includes different commands such as create, alter, drop, truncate, comment, rename etc.
- **Data Manipulation Language (DML)** which deals with data manipulation. It includes different commands such as select, insert, update, delete, merge etc.
- **Data Control Language (DCL)** which deals with the rights and permissions of database system. It includes commands such as grant and revoke

Advantages of SQL

- SQL Queries can be used to retrieve large amounts of records from a database quickly and efficiently
- Using standard SQL, it is easier to manage database systems without having to write substantial amount of code
- SQL databases use long-established standards which is being adopted by ANSI & ISO

Disadvantages of SQL

- **Difficulty in Interfacing :** Interfacing an SQL database is more complex than adding a few lines of code

- **More Features Implemented in Proprietary way :** Although SQL databases conform to ANSI & ISO standards, some databases go for proprietary extensions to standard SQL to ensure vendor lock-in

What is Excel?

proschool
An ims Initiative

- Microsoft Excel is an application used to build and format spreadsheets
- It is good for operations on flat files - summarizing, cross-validating, visualizing, and building pivot tables
- It is a very powerful application that has the ability to do many things such as calculating loan, etc.

Advantages of Excel

- It’s easy to browse data
- It’s easy to manually enter and edit data
- Formulas make it a living document
- It has a built-in suite of helpers for charts, comments, spellchecking, etc.
- It’s relatively easy to learn

Disadvantages of Excel

- It is not suitable for working with multiple datasets in combination
- Lacks data integrity
- It doesn’t scale. As the amount of data increases, performance suffers
- If you have two people editing data in Excel, you can expect three copies of the final spreadsheet
- It uses a bunch of filters and sorting for finding some data in a spreadsheet

Why SQL is suited for Data Analysis ?

proschool
An ims Initiative

- It is used for accessing, cleaning and analysing data that is stored in a databases
- It is semantically easy to understand and learn
- It can be used to access large amounts of data directly where it's stored, analysts don't have to copy data into other applications
- Compared to spreadsheet tools, data analysis done in SQL is easy to audit and replicate
- SQL is great for performing the types of aggregations that you might normally do in an Excel pivot table—sums, counts, minimums and maximums, etc.—but over much larger datasets and on multiple tables at the same time

SQL vs Excel

proschool
An ims Initiative

- If you need to work with a lot of data, you can do it with one or two commands. You don't need to worry about skipping the last row or having one incorrect formula out of 1,000

- For searching some data, you can do it with one query instead of using a bunch of filters and sorts or a manual selection in database

- If you have two people editing data in Excel, you can expect three copies of the final spreadsheet. This is manageable with a little data, but it becomes time-consuming and error prone with more. Databases are designed to handle multiple users

- Even if you're just storing data, Excel has a hard limit of one million rows per sheet. A good database will store and process as much data as your hard drive can hold

MySQL Installation

For Windows Using MySQL Installer

- To download MySQL installer, go to the following link **http://dev.mysql.com/downloads/installer/.** There are two files are available. If you are connecting to the internet while installing MySQL, you can choose the online installation version **mysql-installer-web-community.exe** . If you want to install MySQL offline, you can download the **mysql-installer-community.exe** file
- After downloading the MySQL installer, double click on it and then follow the next steps
- Windows configures MySQL Installer
- A welcome screen provides several options. Choose the first option: Install MySQL Products
- MySQL installer checks and downloads the latest MySQL products including MySQL server, MySQL Workbench, etc.

MySQL Installation (Cont.)

- Click Next button to continue
- Next there are several setup types available. Choose the Full option to install all MySQL products and features
- It will check for the requirements
- MySQL Installer downloads all selected products. It will take a while, depending on which products that you selected and the speed of your internet connection
- Next it will show the downloading progress
- Once the downloading is over, click on next button to continue
- Click Next button to configure MySQL Database Server
- Choose Config Type and MySQL port (3306 by default) and click Next button to continue

MySQL Installation (Cont.)

- Choose a password for the root account. Please note the password download and keep it securely if you are install MySQL database server in a production server. If you want to add more MySQL user, you can do it in this step
- Choose Windows service details including Windows Service Name and account type, then click Next button to continue.
- MySQL Installer is configuring MySQL database server. Wait until it is done and click Next button to continue
- Once Done. Click the Next button to continue
- MySQL Installer installs sample databases and sample models
- The installation completes. Click finish button to close the installation wizard and launch the MySQL Workbench

For Reference：**http://www.mysqltutorial.org/install-mysql/**

MySQL Installation- Error Handling

- Error handling
  - Check the MySQL error log
  - If there is an error for Installing Visual Code Distributable
  - Install it from this location
  - Install both the 32bit and the 64bit system
  - https://www.microsoft.com/en-in/download/details.aspx?id=40784

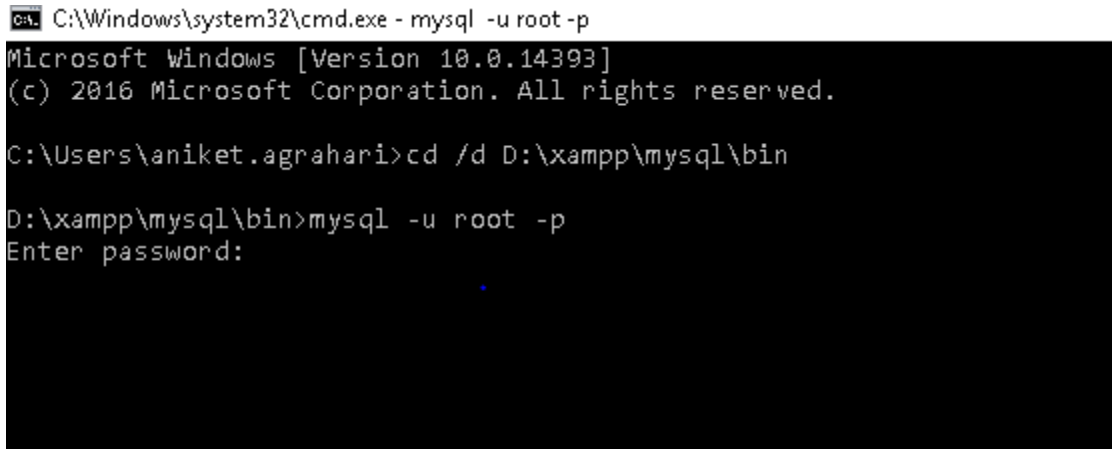
MySQL Installation (Cont.)

For Linux (Ubuntu)

- Firstly login to your Linux server using SSH
- Then run below commands for installing MySQL
  - **sudo apt-get install mysql-server** for downloading the mysql package
  - During the installation process, you will be prompted to set a password for the MySQL root user. Choose a strong password and keep it in a safe place for future reference
  - **sudo mysql_secure_installation** script to address several security concerns in a default MySQL installation
  - You will be given the choice to change the MySQL root password, remove anonymous user accounts, disable root logins outside of localhost, and remove test databases. It is recommended that you answer yes to these options

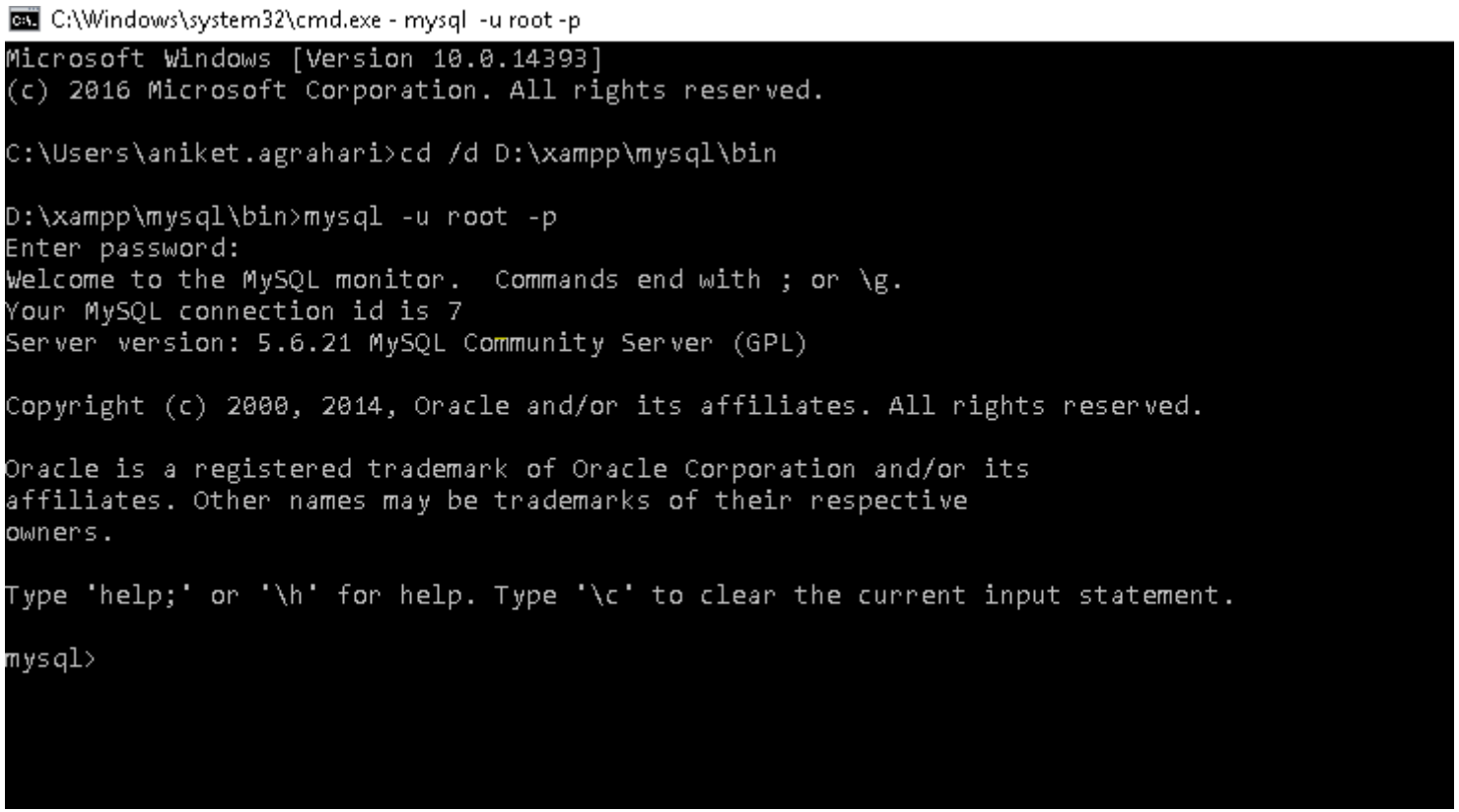
MySQL Installation (Cont.)

- After installation, mysql client is accessed through a terminal
- To login to the MySQL as the root user:
  - Run this command, **mysql -u root –p**
  - When prompted, enter the root password you assigned when the **mysql_secure_installation** script was run



MySQL Installation (Cont.)

- After entering the correct password, you will get the access to mysql server and can start running the query from terminal

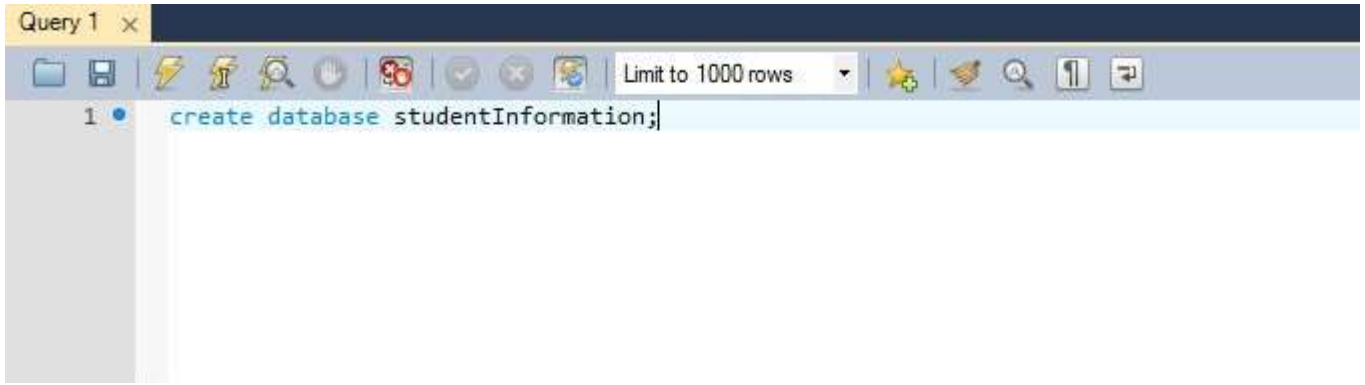


What is Database?

- A database is a tool for collecting and organizing information.
- Database can store information about people, products, orders or anything else.
- Databases make data management easy. Let's discuss few examples
  - An online telephone directory would definitely use database to store data pertaining to people, phone numbers, other contact details, etc.
  - Your electricity service provider is using a database to manage billing , client related issues, to handle fault data, etc.

Creating the Database

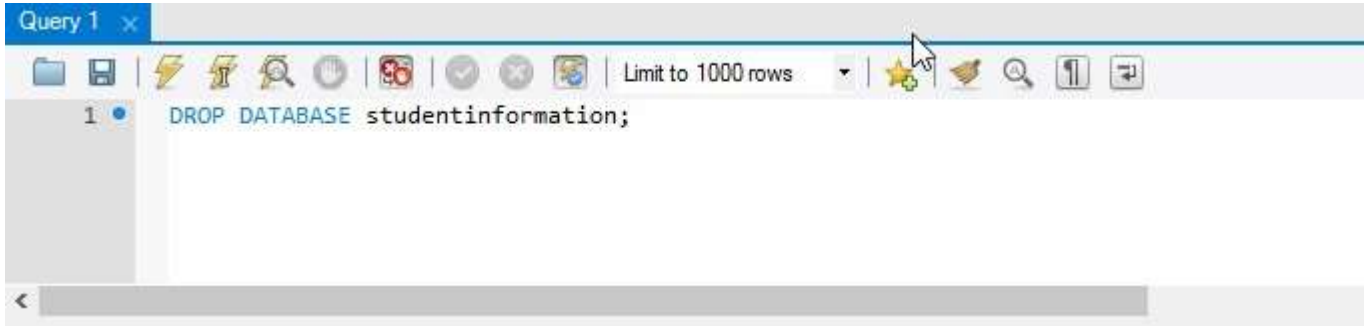
- Database is a systematic collection of data. Database support storage and manipulation of data
- Below is the query for creating the database



- After successfully running the above query, the database is been created

Dropping the Database

- The DROP DATABASE statement is used to drop an existing SQL database
- Below is the query for dropping the database



- After successfully running the above query, the above mentioned database is dropped

SQL Constraints

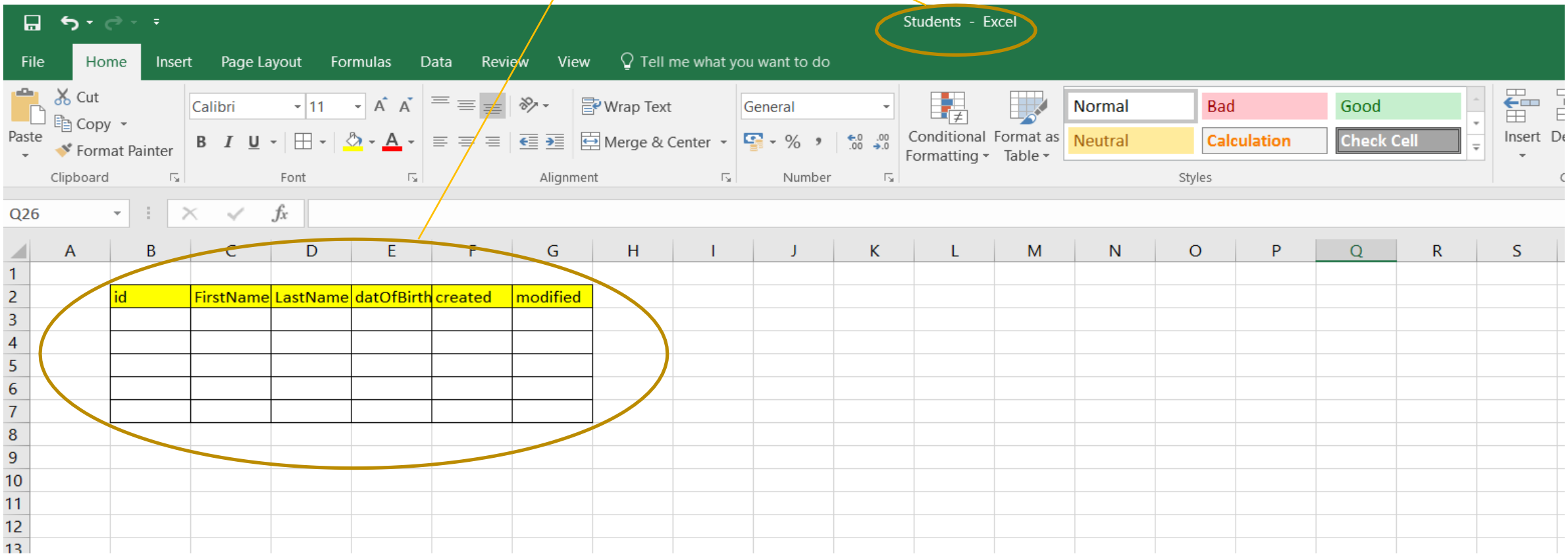
- Constraints are the rules enforced on the data columns of a table. These are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the database
- Constraints could be either on a column level or a table level. The column level constraints are applied only to one column, whereas the table level constraints are applied to the whole table
- Following are some of the most commonly used constraints available in SQL
  - **NOT NULL Constraint:** Ensures that a column cannot have NULL value.
  - **DEFAULT Constraint:** Provides a default value for a column when none is specified.
  - **UNIQUE Constraint:**Ensures that all values in a column are different.
  - **PRIMARY Key:** Uniquely identifies each row/record in a database table.
  - **FOREIGN Key:** Uniquely identifies a row/record in any of the given database table
  - **CHECK Constraint:** The CHECK constraint ensures that all the values in a column satisfies certain conditions

Data types in SQL

- Characters:
  - CHAR(20) -- fixed length
  - VARCHAR(40) -- variable length
- Numbers:
  - BIGINT, INT, SMALLINT, TINYINT
  - DOUBLE, FLOAT -- differ in precision
- Times and dates:
  - DATE -- stores date value in Y-M-D format
  - DATETIME -- stores date and time in Y-M-D HH:MM:SS format

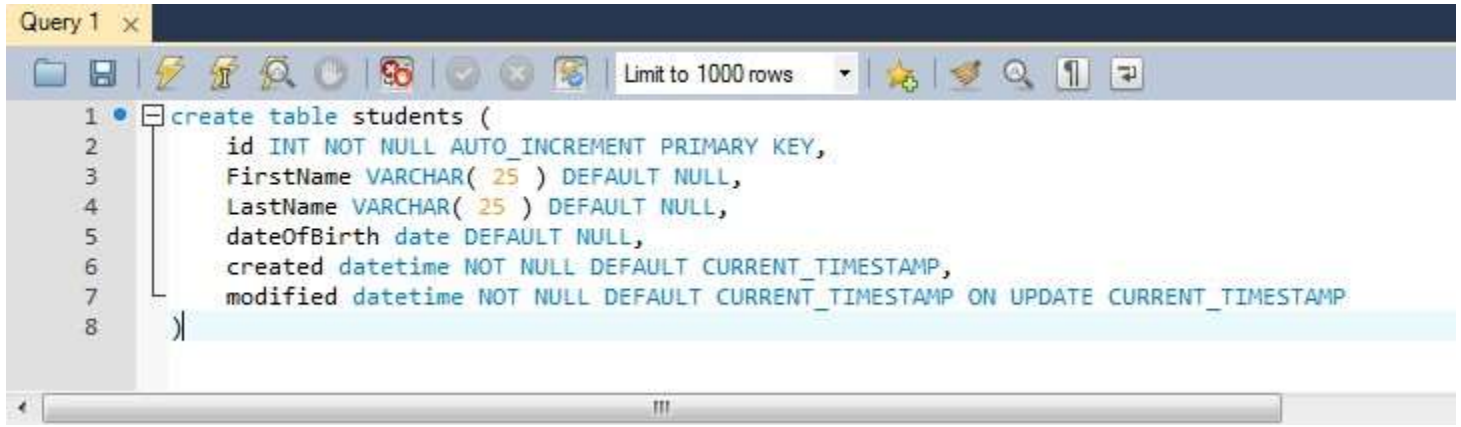
Creating the Table

- A table is a collection of related data held in a structured format within a database. It consists of columns and rows.
- Lets create the below mentioned Excel table (Students) in SQL



Creating the Table

- A table is a collection of related data held in a structured format within a database. It consists of columns and rows
- Below is the query for creating the table



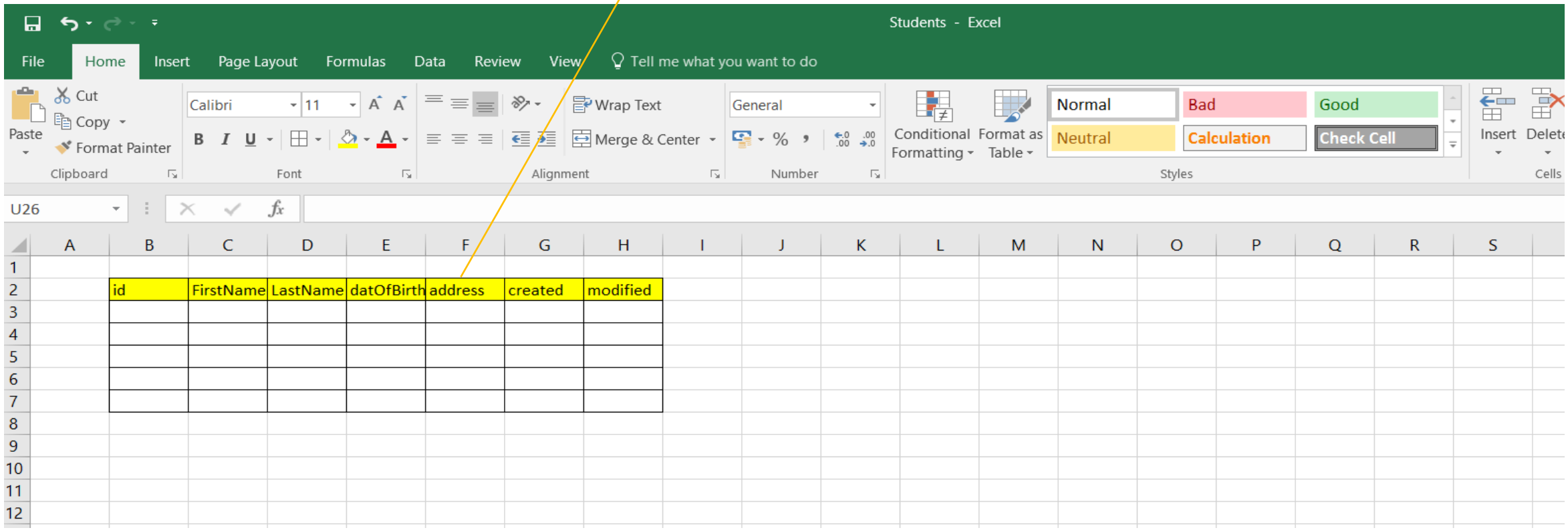
- After successfully running the above query, the **students** table is been created

Table Structure Explained

- The **id** column is of type int and will hold an integer value with maximum length of 11 digits. It is also an auto increment value which means at a new insertion of record, a new auto generated value would be generated and inserted. It is also primary key which can contain only unique values. A primary key column **cannot have NULL** values
- The **firstName** and **lastName** columns are of type varchar and will hold characters, and the maximum length for these fields is 25 characters and can also hold default NULL value in case of no data.
- The **dateOfBirth** column is of type date which will hold data in format of Y-M-D and can also hold default NULL value in case of no data
- The **created** column is of type datetime which will hold data in format of Y-M-D HH:MM:SS and at the time of new record insertion the value would be current timestamp
- The **modified** column is of type datetime which will hold data in format of Y-M-D HH:MM:SS and at the time of new record insertion the value would be current timestamp and at the time of updation of old record, the value would be updated as per current timestamp

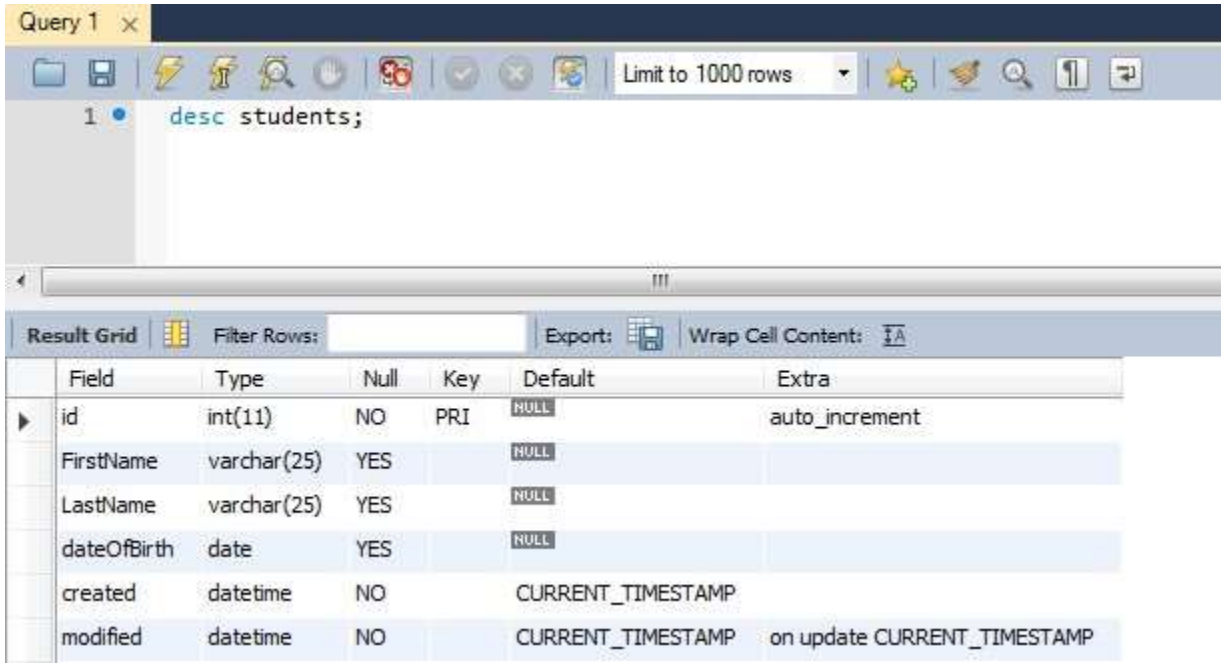
Altering the Table

- If you wish to add a new column viz. address as shown in the Excel table



Altering the Table (Cont.)

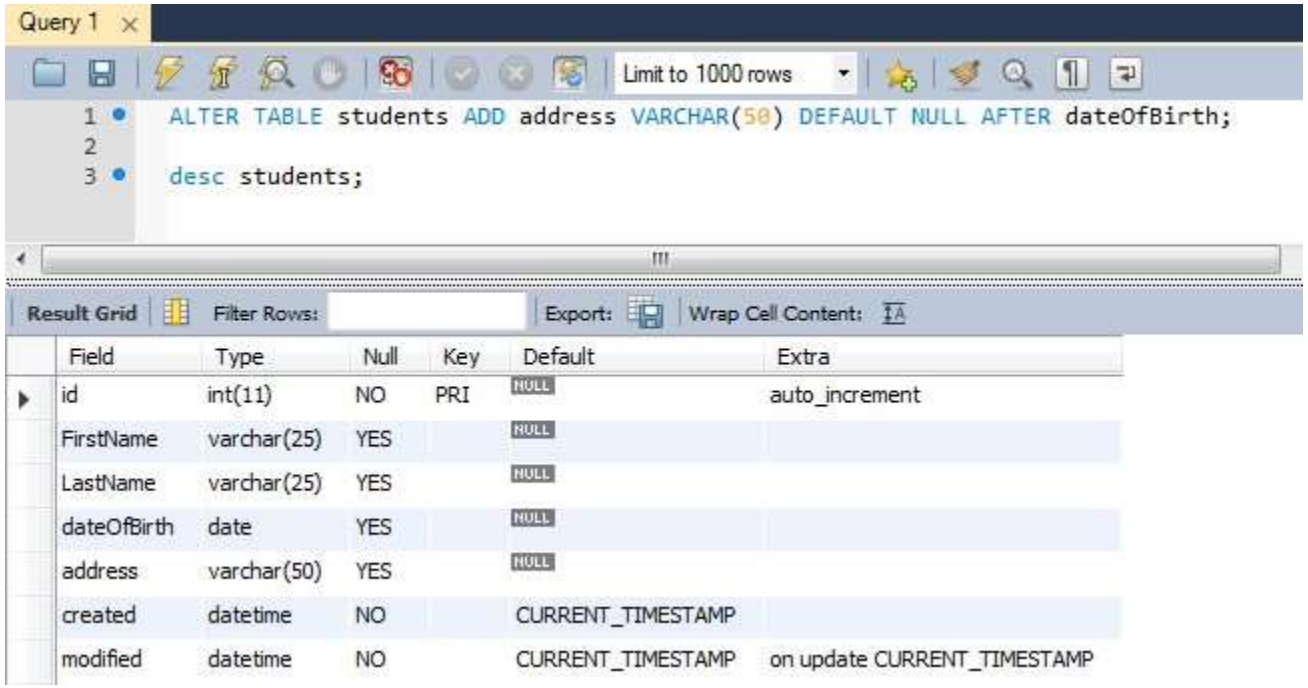
- The ALTER TABLE statement is used to add, delete, or modify columns in an existing table
- The ALTER TABLE statement is also used to add and drop various constraints on an existing table
- Before altering the table structure:



Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
FirstName	varchar(25)	YES		NULL	
LastName	varchar(25)	YES		NULL	
dateOfBirth	date	YES		NULL	
created	datetime	NO		CURRENT_TIMESTAMP	
modified	datetime	NO		CURRENT_TIMESTAMP	on update CURRENT_TIMESTAMP

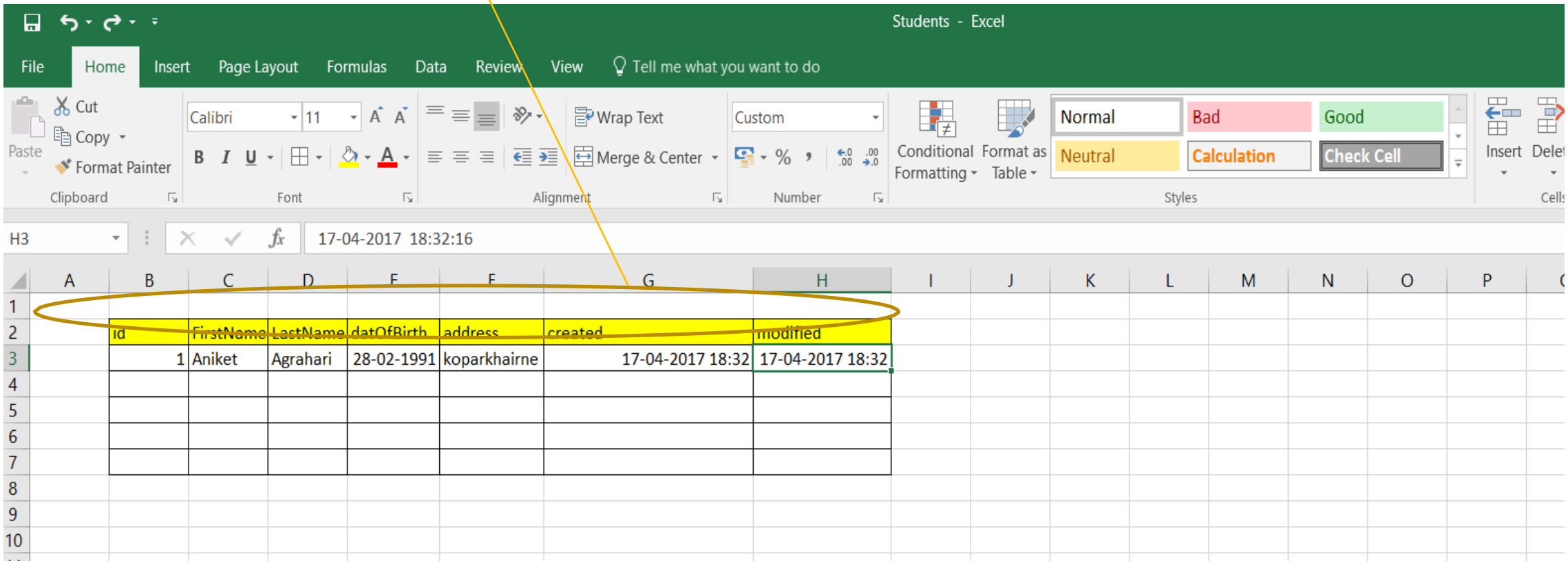
Altering the Table(Cont.)

- Below is the query for adding a new column i.e. address with it’s data type as varchar(50) after dateOfBirth column



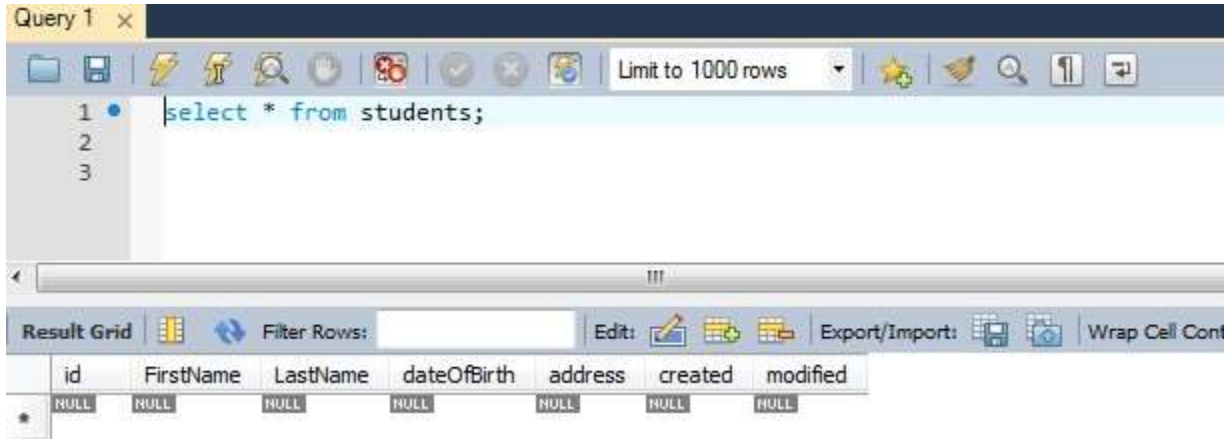
Inserting a Record

- If you wish to add a new record as shown in the Excel table



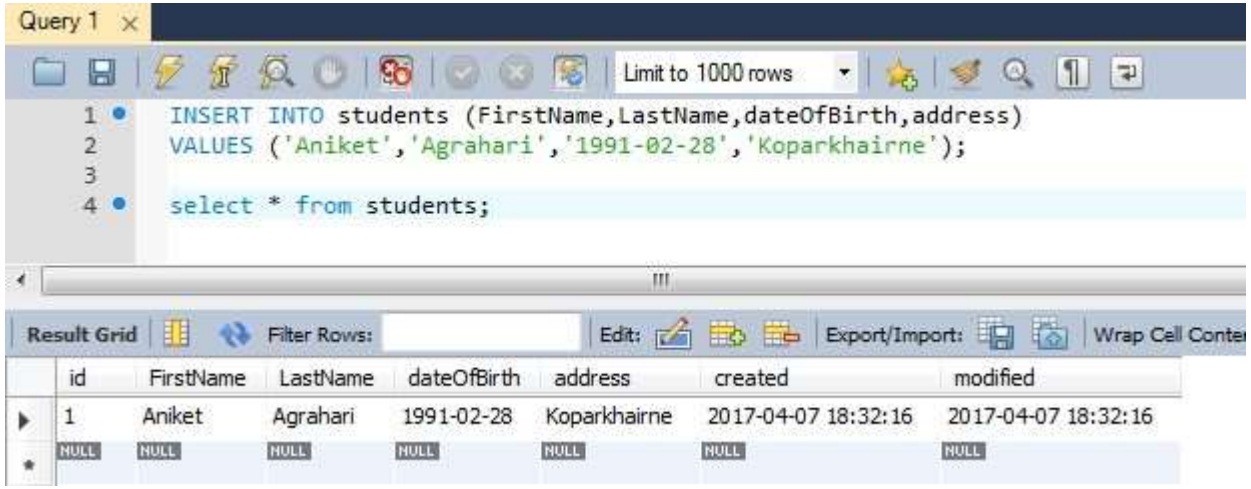
Inserting a Record

- Used to insert a new record into table
- There are two ways to write INSERT INTO statement:
  - INSERT INTO table_name (column1, column2, column3) VALUES (value1, value2, value3);
  - INSERT INTO table_name VALUES (value1, value2, value3);
- Before inserting a new record:



Inserting a Record (Cont.)

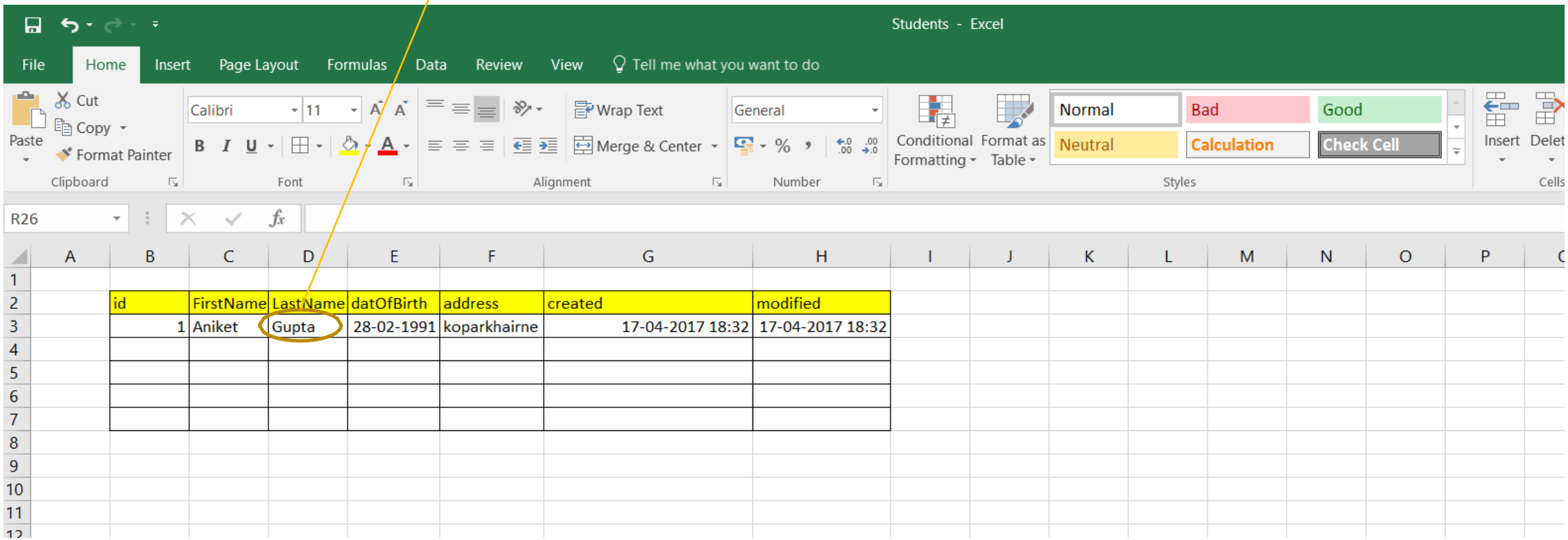
- Below is the query for inserting a new record into the table



- In the above query, we have not included three columns i.e. id, created, modified as explained in the previous slide

Updating a Record

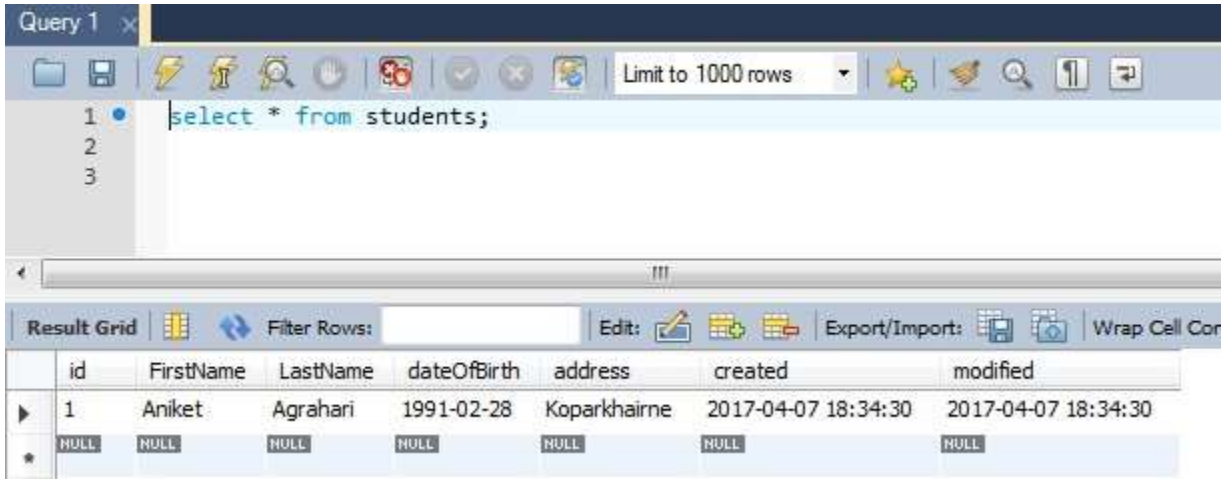
If you wish to change the surname to Gupta i.e update the record as shown in the Excel table



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1																
2		id	FirstName	LastName	datOfBirth	address	created	modified								
3		1	Aniket	Gupta	28-02-1991	koparkhairne	17-04-2017 18:32	17-04-2017 18:32								
4																
5																
6																
7																
8																
9																
10																
11																

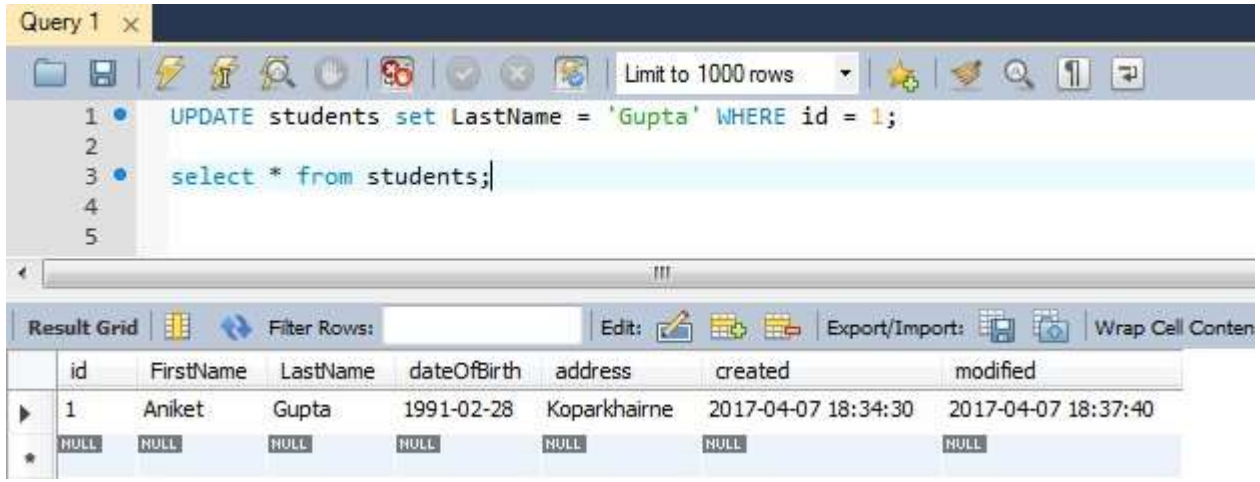
Updating a Record

- Used to update an existing record into table
- Syntax for updating a record is **“UPDATE table_name SET column1 = value1, column2 = value2 WHERE condition;”**
- Before updating an old record:-



Updating a Record (Cont.)

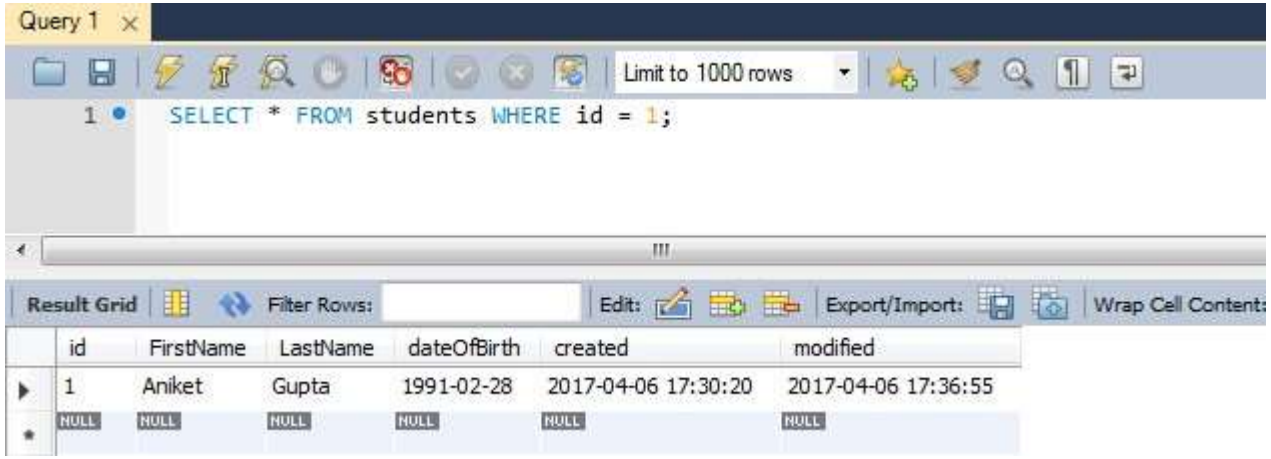
- Below is the query for updating a lastName column value of a record into the table



- In the above query, we updated the lastName of a record with id as 1

Selecting a Record

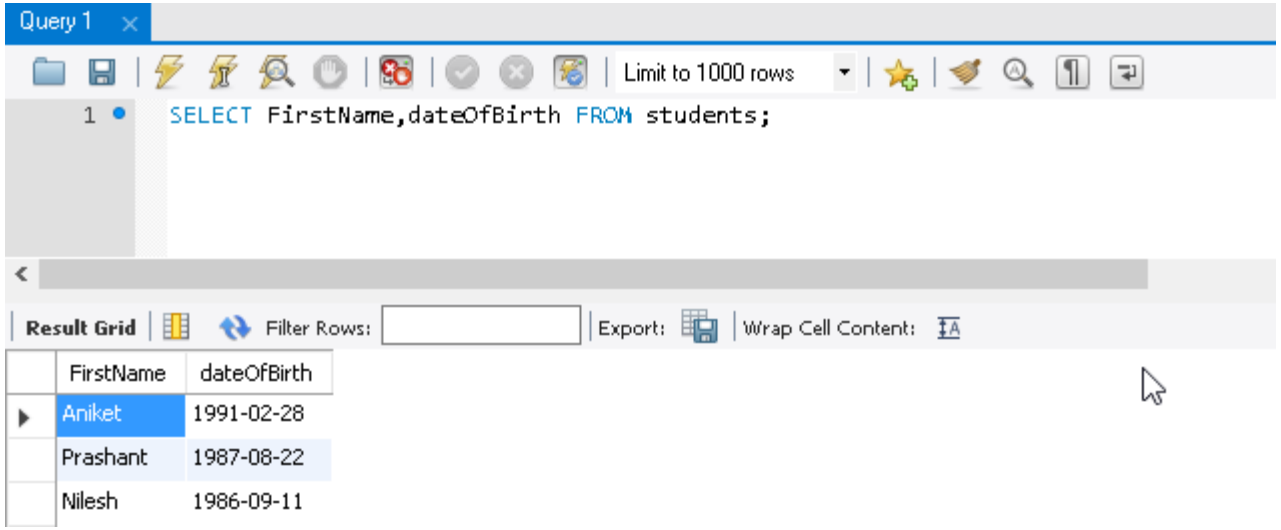
- Used to retrieve an existing records from the table
- Below is the query for retrieving a single record from the table



- For retrieving all the records, **“SELECT * FROM students;”**

Selecting a Record (Cont.)

- Below is the query for retrieving only FirstName and dateOfBirth from the student table

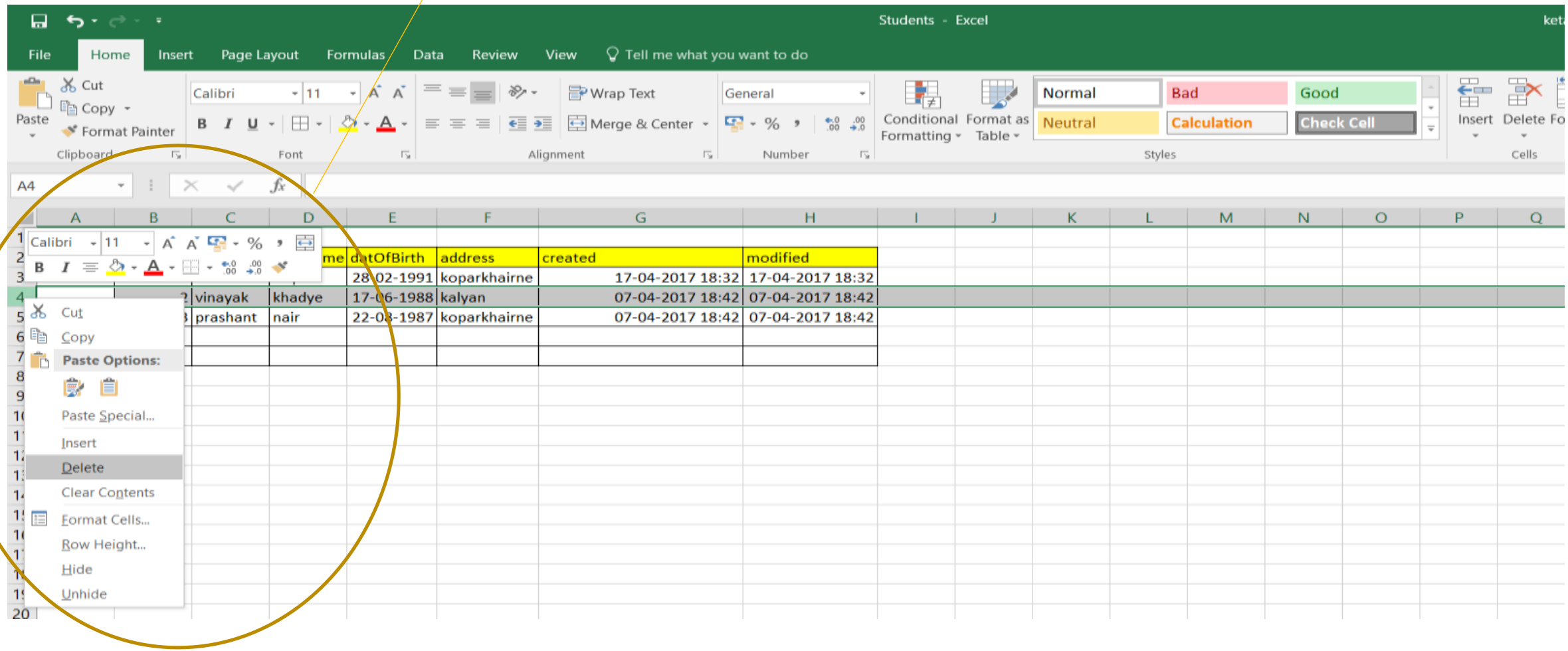


Selecting a Distinct Record

- Below is the query for retrieving distinct names from students table

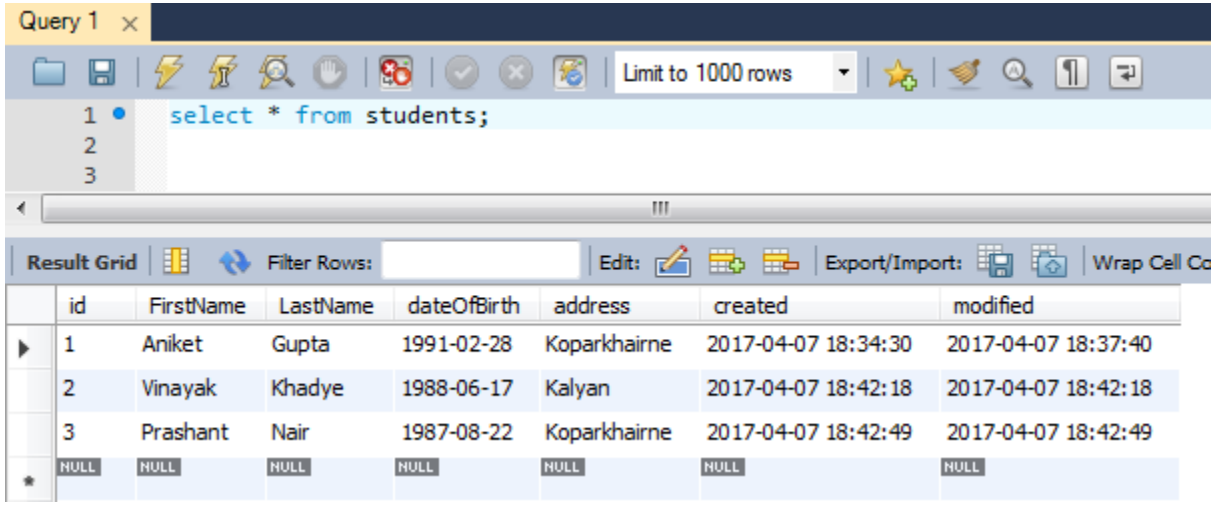
Deleting a Record

- If you wish to delet2 the id=2 record as shown in the Excel table



Deleting a Record

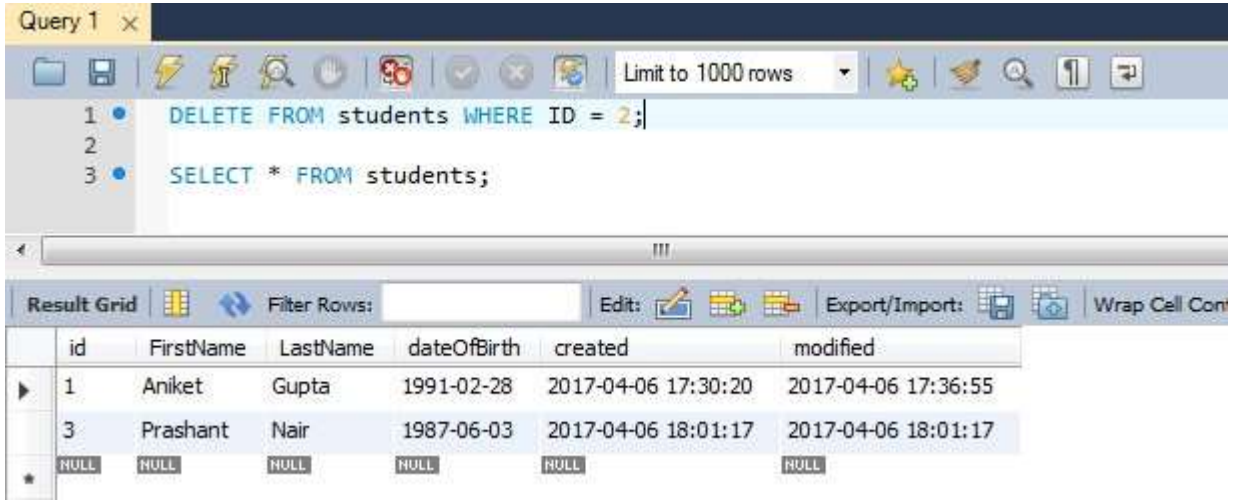
- Used to delete an existing records from the given table
- Syntax for updating a record is **“DELETE FROM *table_name* WHERE *condition*;”**.
- Before deleting a record:



id	FirstName	LastName	dateOfBirth	address	created	modified
1	Aniket	Gupta	1991-02-28	Koparkhairne	2017-04-07 18:34:30	2017-04-07 18:37:40
2	Vinayak	Khadye	1988-06-17	Kalyan	2017-04-07 18:42:18	2017-04-07 18:42:18
3	Prashant	Nair	1987-08-22	Koparkhairne	2017-04-07 18:42:49	2017-04-07 18:42:49
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Deleting A Record (Cont.)

- Below is the query for deleting a record from the table:

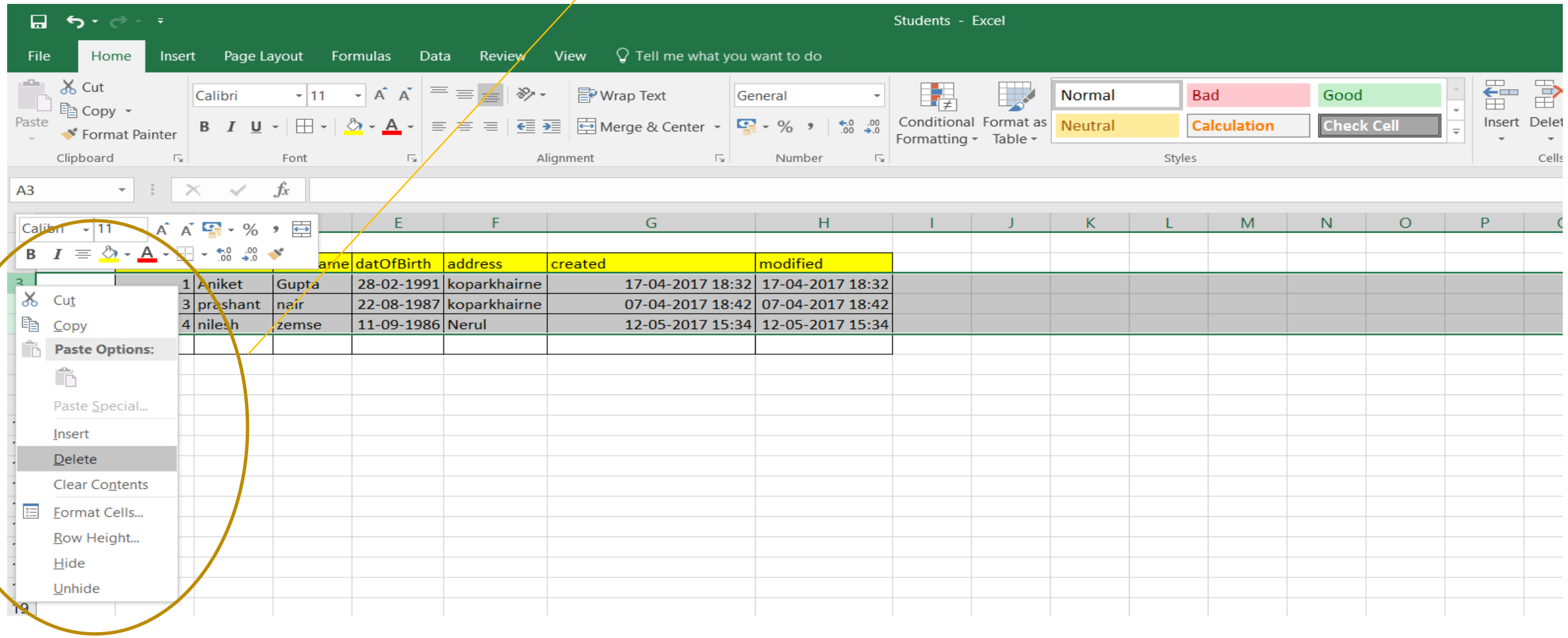


- Above we have deleted a record with id as 2 and then we have retrieved all the records from that table, if we want to delete all the records we can run **“DELETE * FROM students;”**

Truncating the Table

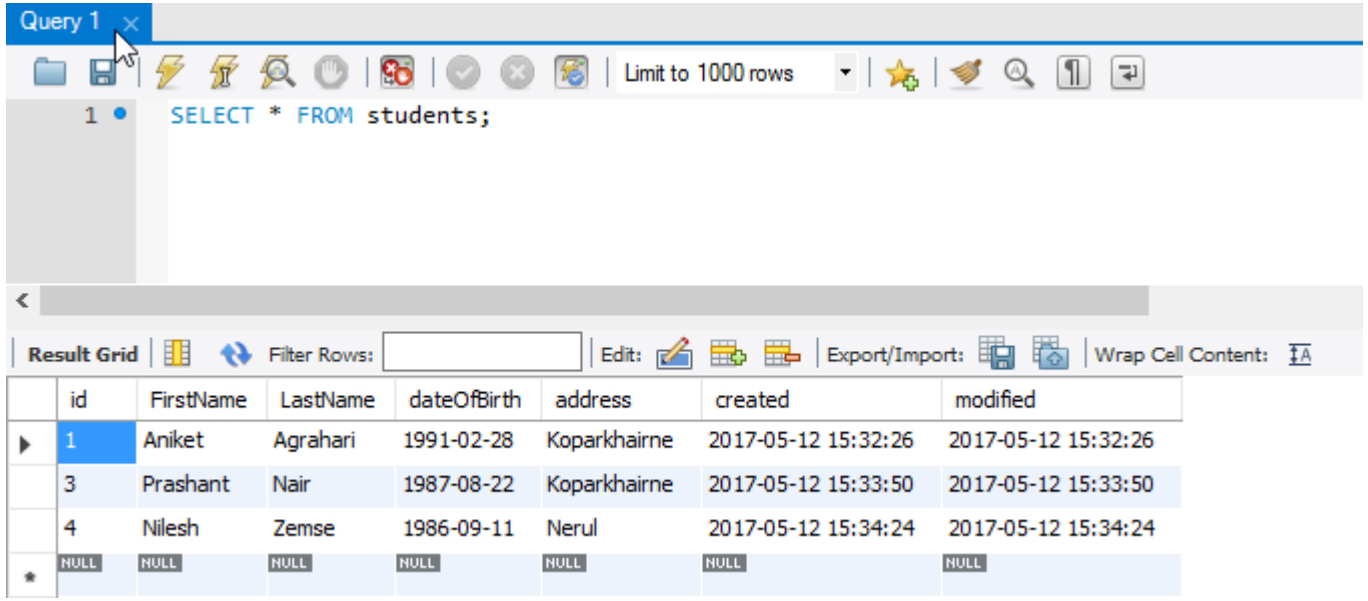
proschool
An ims Initiative

- To Delete all record in Excel i.e to Truncate



Truncating the Table

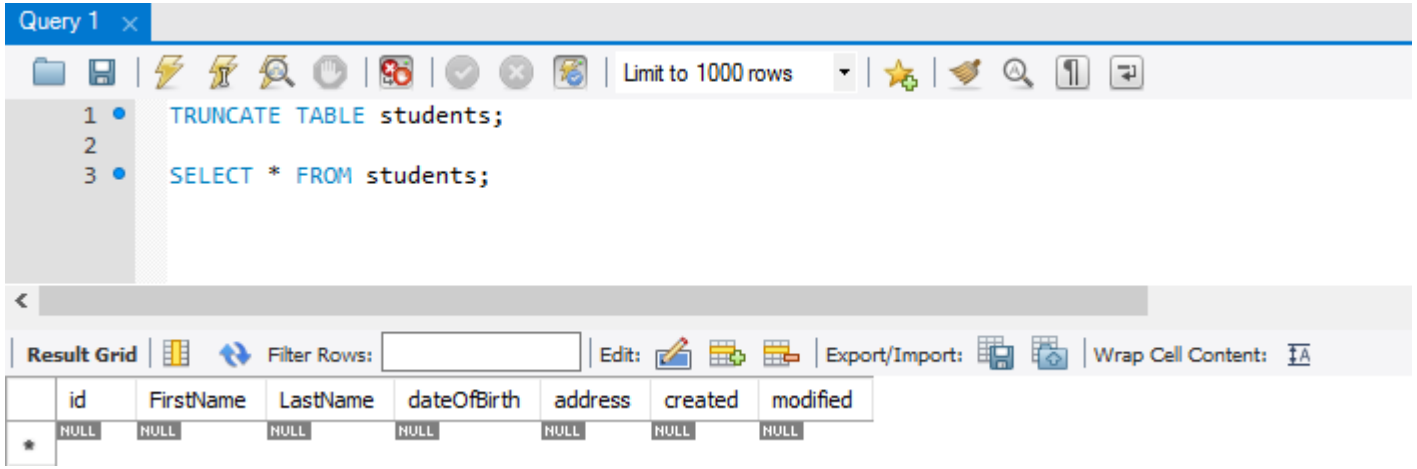
- The **TRUNCATE TABLE** statement is used to delete the data inside a table, but not the table itself
- Before truncating the table:



id	FirstName	LastName	dateOfBirth	address	created	modified
1	Aniket	Agrahari	1991-02-28	Koparkhairne	2017-05-12 15:32:26	2017-05-12 15:32:26
3	Prashant	Nair	1987-08-22	Koparkhairne	2017-05-12 15:33:50	2017-05-12 15:33:50
4	Nilesh	Zemse	1986-09-11	Nerul	2017-05-12 15:34:24	2017-05-12 15:34:24
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Truncating the Table (Cont.)

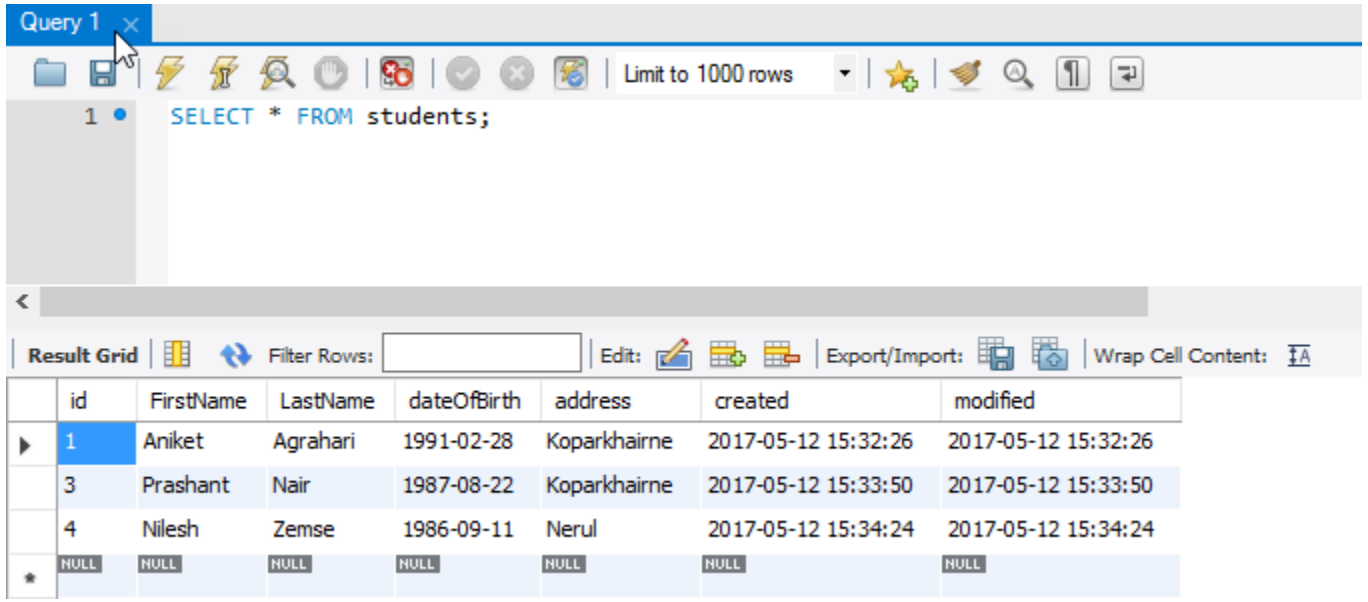
- Below is the query for truncating the table



- After successfully running the above query, all the records from the table gets deleted

Dropping the Table

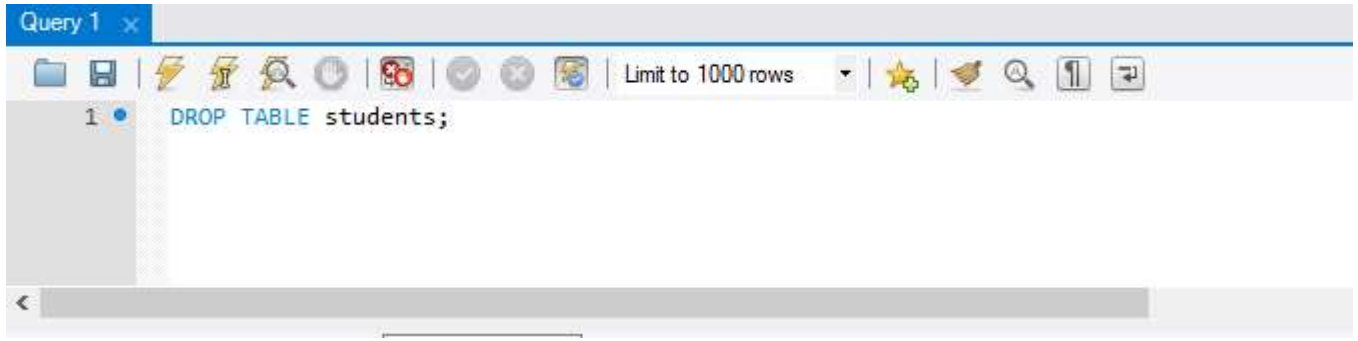
- The **DROP TABLE** statement is used to drop an existing table in a database
- Before dropping the table:



id	FirstName	LastName	dateOfBirth	address	created	modified
1	Aniket	Agrahari	1991-02-28	Koparkhairne	2017-05-12 15:32:26	2017-05-12 15:32:26
3	Prashant	Nair	1987-08-22	Koparkhairne	2017-05-12 15:33:50	2017-05-12 15:33:50
4	Nilesh	Zemse	1986-09-11	Nerul	2017-05-12 15:34:24	2017-05-12 15:34:24
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Dropping the Table (Cont.)

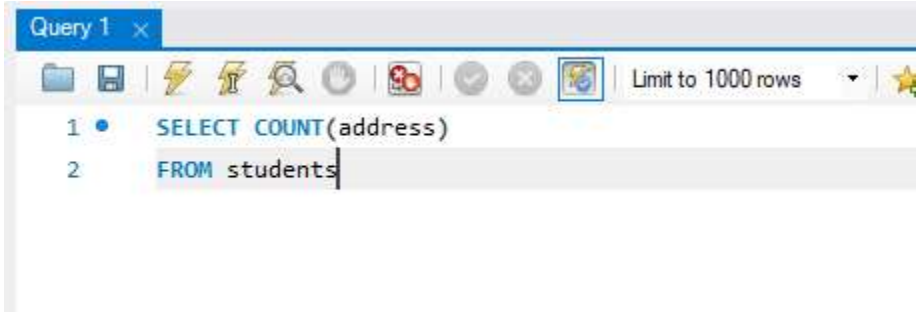
- Below is the query for dropping the table



- After successfully running the above query, the above mentioned table is dropped

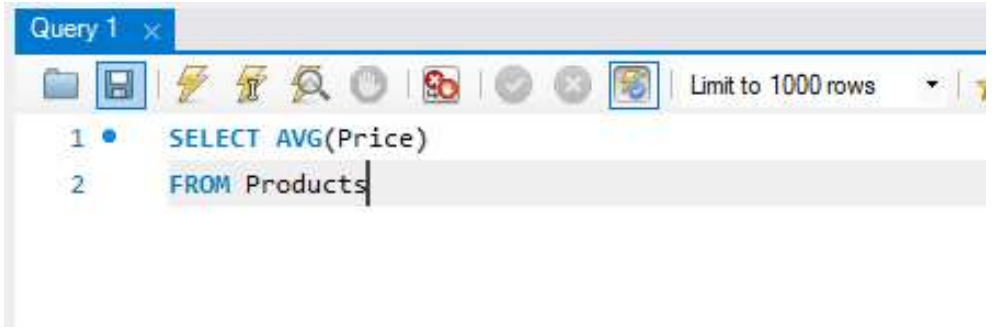
Count in SQL

- Count function in SQL is used to count total number of observations in a column
- Below is the syntax used to count number of observations in address column



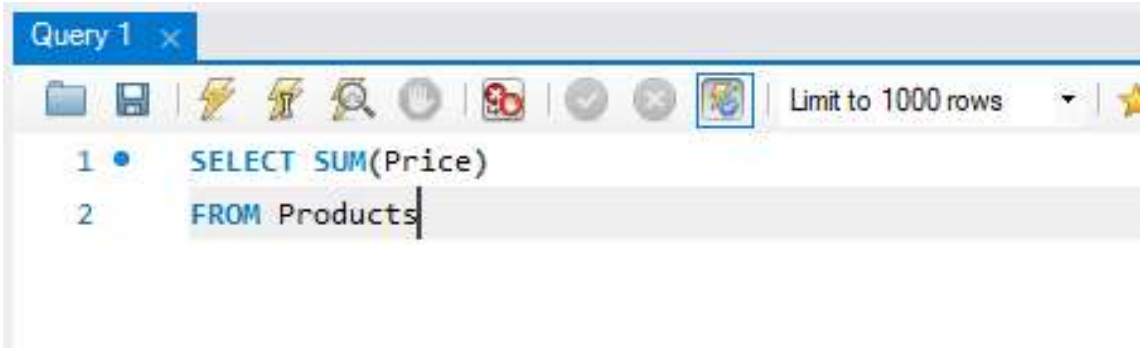
Average in SQL

- Average function in SQL is used to get an average of observations in a column
- Below is the syntax used to get an average of observations in Price column



Sum in SQL

- Sum function in SQL is used to get a sum of observations in a column
- Below is the syntax used to get a sum of observations in Price column



Operators in SQL

- An operator is a reserved word or a character used in an SQL statement's WHERE clause to perform operations, such as comparisons and arithmetic operations

- Below are the list of some operators:
  - Arithmetic operators
  - Comparison operators
  - Logical operators

SQL Arithmetic Operators

proschool
An ims Initiative

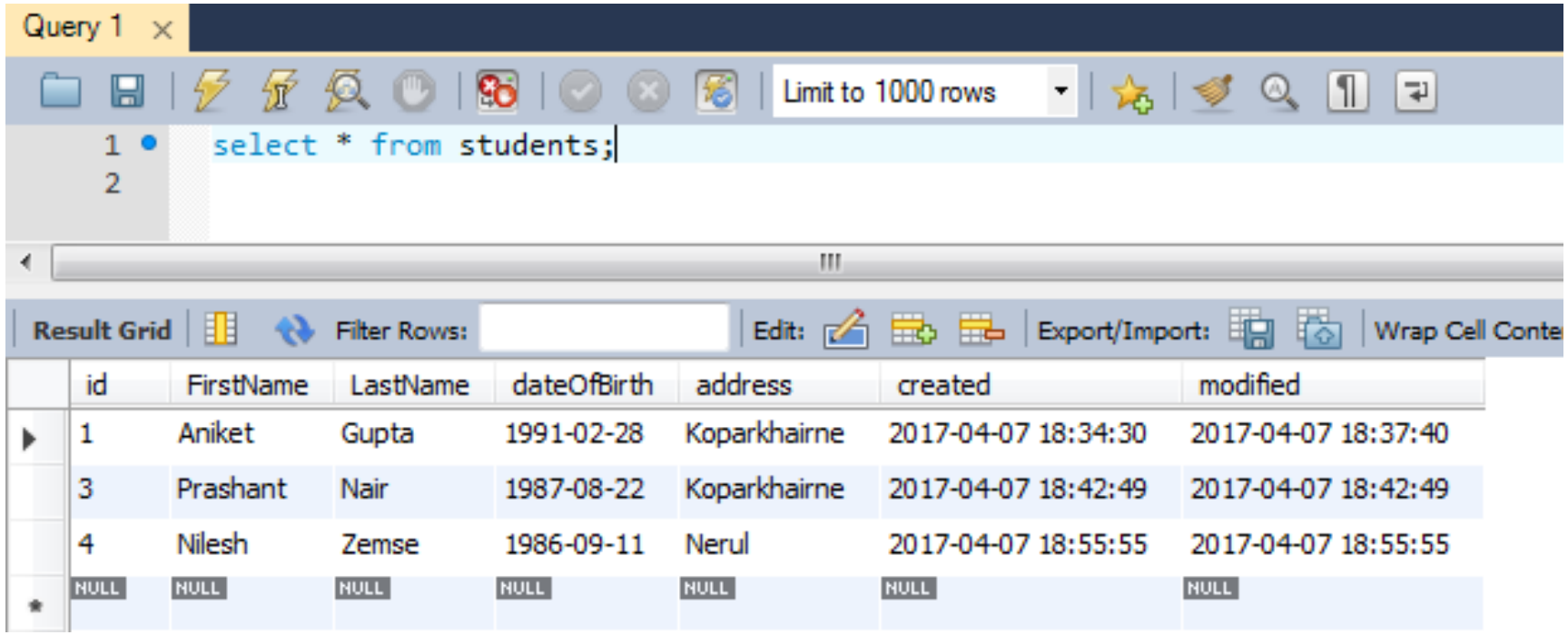
Operator	Description	Example
+	Addition - Adds values on either side of the operator	a + b
-	Subtraction - Subtracts right hand operand from left hand operand	a - b
*	Multiplication - Multiplies values on either side of the operator	a * b
/	Division - Divides left hand operand by right hand operand	b / a
%	Modulus - Divides left hand operand by right hand operand and returns remainder	b % a

SQL Comparison Operators

Operator	Description	Example
=	Checks if the values of two operands are equal or not, if yes then condition becomes true.	(a = b)
!=	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(a != b)
<>	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(a <> b)
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(a > b)
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(a < b)

SQL Comparison Operators (Cont.)

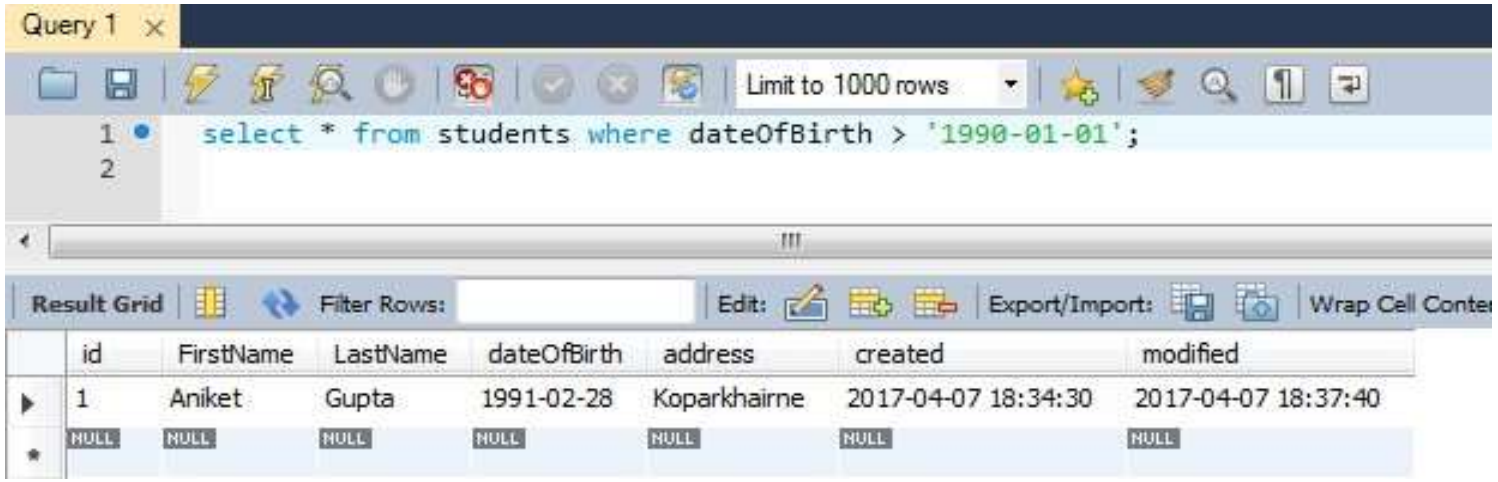
- Below is the current table data:



id	FirstName	LastName	dateOfBirth	address	created	modified
1	Aniket	Gupta	1991-02-28	Koparkhairne	2017-04-07 18:34:30	2017-04-07 18:37:40
3	Prashant	Nair	1987-08-22	Koparkhairne	2017-04-07 18:42:49	2017-04-07 18:42:49
4	Nilesh	Zemse	1986-09-11	Nerul	2017-04-07 18:55:55	2017-04-07 18:55:55
NULL	NULL	NULL	NULL	NULL	NULL	NULL

SQL Comparison Operators (Cont.)

- Below is the query for finding students whose date of birth is greater than 1990-01-01.



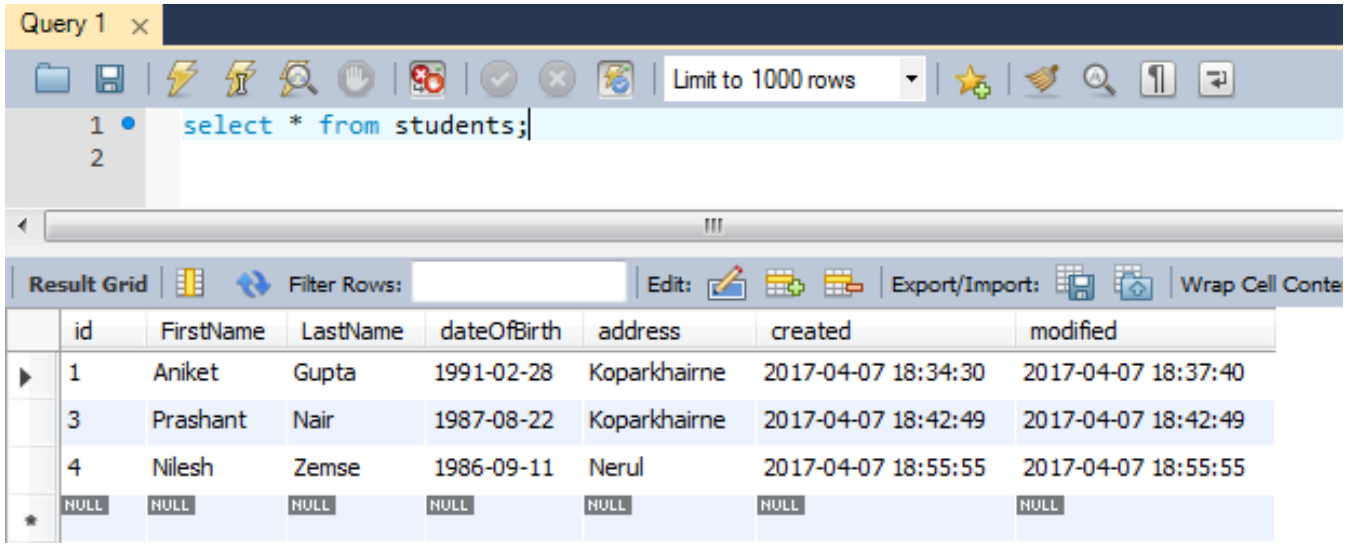
- Same way we can use other comparison operators defined in previous slide as per our need.

SQL Logical Operators

Operator	Description
ALL	The ALL operator is used to compare a value to all values in another value set.
AND	The AND operator allows the existence of multiple conditions in an SQL statement's WHERE clause.
ANY	The ANY operator is used to compare a value to any applicable value in the list according to the condition.
BETWEEN	The BETWEEN operator is used to search for values that are within a set of values, given the minimum value and the maximum value.
IN	The IN operator is used to compare a value to a list of literal values that have been specified.

SQL Logical Operators (Cont.)

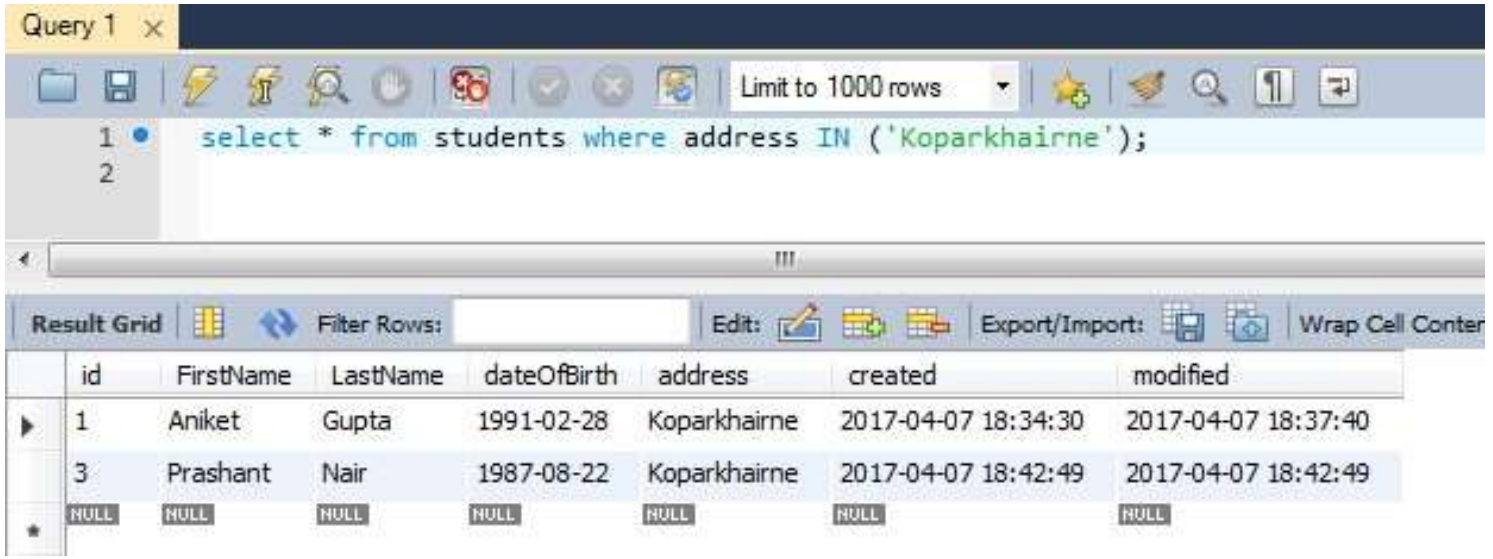
- Below is the current table data:



id	FirstName	LastName	dateOfBirth	address	created	modified
1	Aniket	Gupta	1991-02-28	Koparkhairne	2017-04-07 18:34:30	2017-04-07 18:37:40
3	Prashant	Nair	1987-08-22	Koparkhairne	2017-04-07 18:42:49	2017-04-07 18:42:49
4	Nilesh	Zemse	1986-09-11	Nerul	2017-04-07 18:55:55	2017-04-07 18:55:55
NULL	NULL	NULL	NULL	NULL	NULL	NULL

SQL Logical Operators(Cont.)

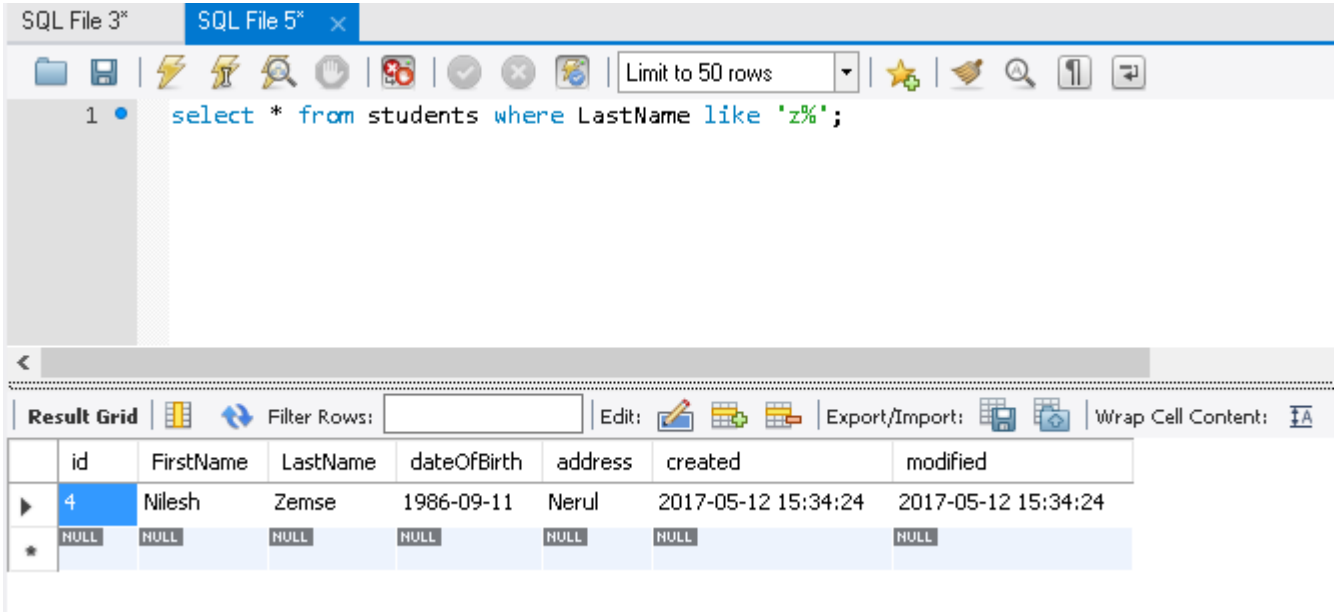
- Below is the query for finding students who are staying in Koparkhairne.



id	FirstName	LastName	dateOfBirth	address	created	modified
1	Aniket	Gupta	1991-02-28	Koparkhairne	2017-04-07 18:34:30	2017-04-07 18:37:40
3	Prashant	Nair	1987-08-22	Koparkhairne	2017-04-07 18:42:49	2017-04-07 18:42:49
NULL	NULL	NULL	NULL	NULL	NULL	NULL

SQL Logical Operators (Cont.)

- Below is the query for finding students whose last name starts with “Z”



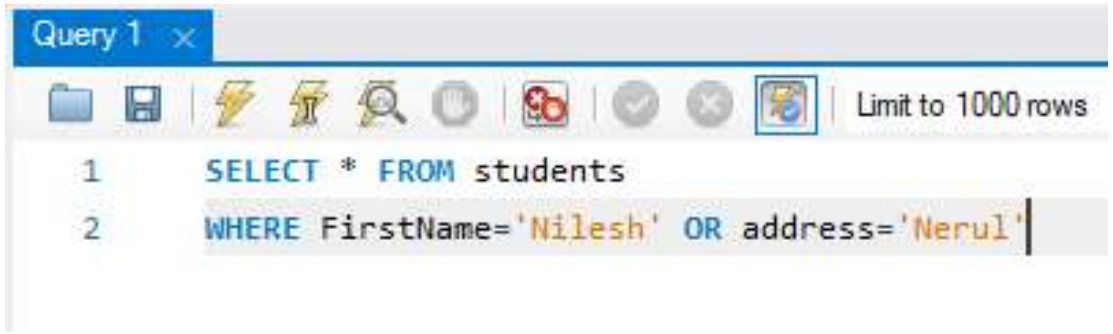
- Same way we can use other logical operators defined in previous slide as per our need

SQL Logical Operators (Cont.)

- Below is the query for finding students whose first name is “Nilesh” and address is “Nerul”

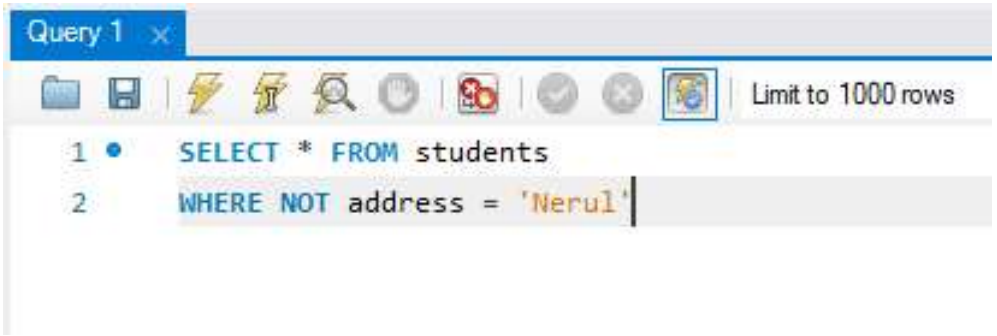
SQL Logical Operators (Cont.)

- Below is the query for finding students whose first name is “Nilesh” or students whose address is “Nerul”



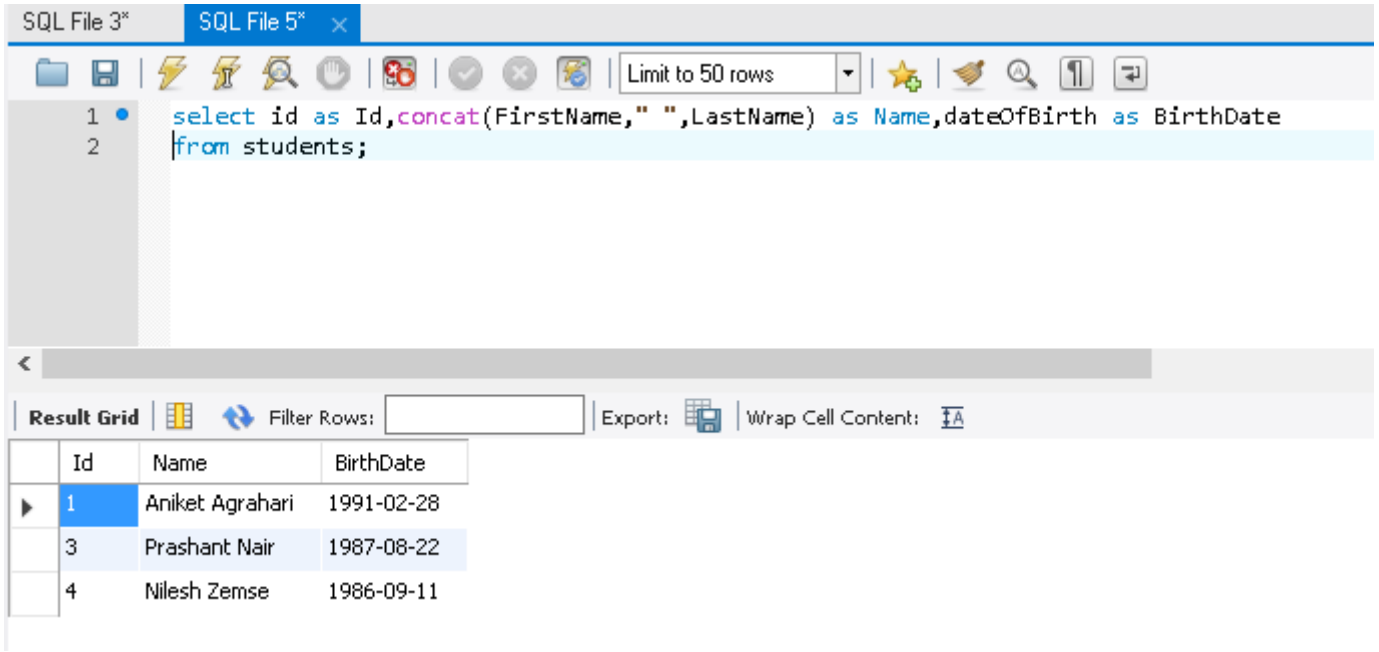
SQL Logical Operators (Cont.)

- Below is the query for finding students whose address is not “Nerul”



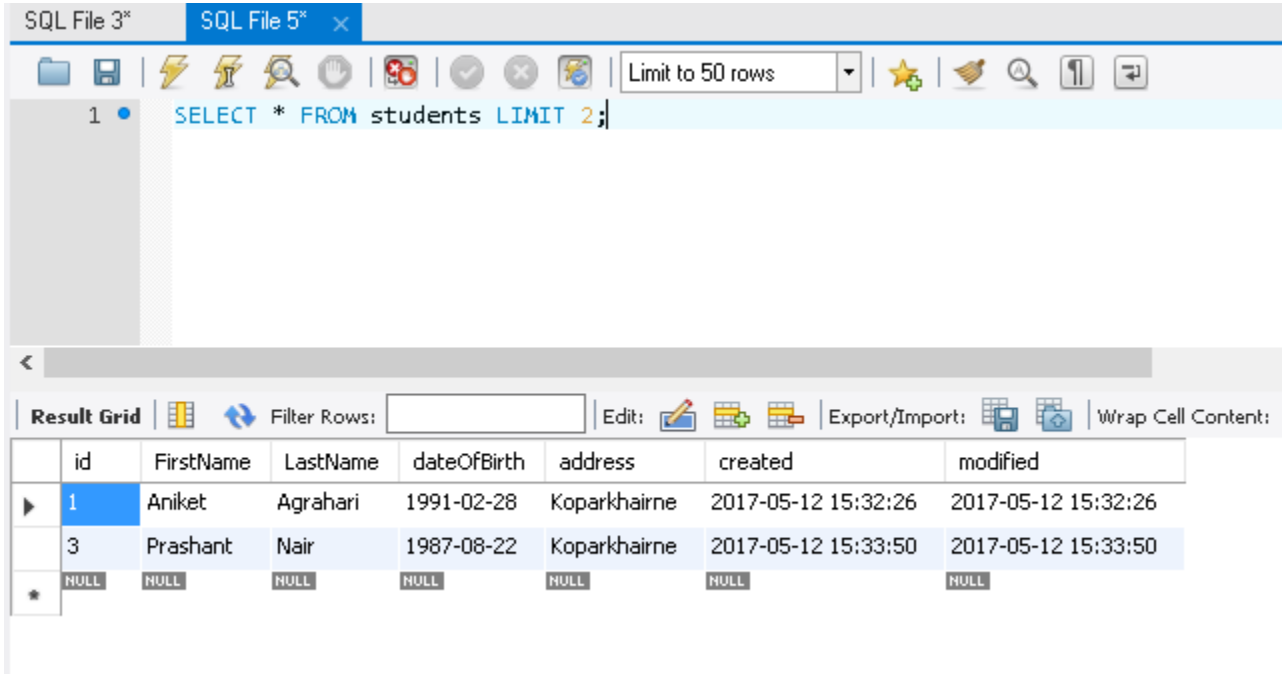
SQL Aliases

- SQL aliases are used to give a table, or a column in a table, a temporary name
- Aliases are often used to make column names more readable
- An alias only exists for the duration of the query



SQL LIMIT Clause

- The **SELECT LIMIT** statement is used to retrieve records from one or more tables in a database and limit the number of records returned based on a limit value
- Below is the query which retrieves only two records from students table

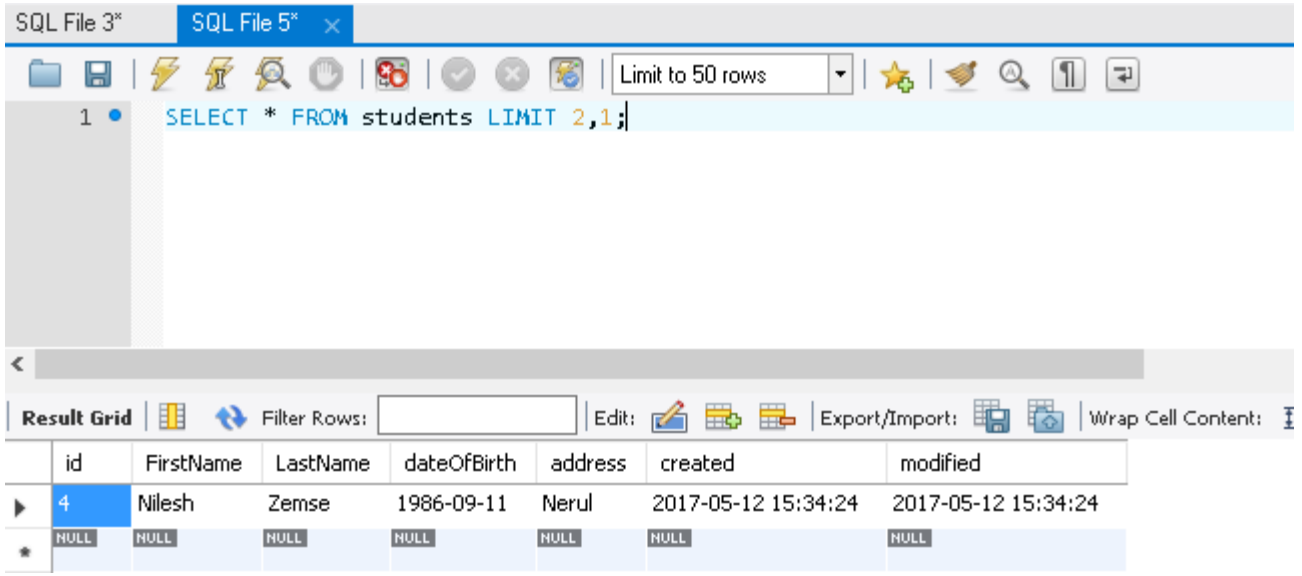


SQL LIMIT Clause (Cont.)

- When we use the **LIMIT clause** with one argument, this argument will be used to determine the maximum number of rows to return from the beginning of the result set
- The **LIMIT clause** accepts one or two arguments. The values of both arguments must be zero or positive integers
- Below is the syntax with two arguments:
  - SELECT column1,column2,... FROM table LIMIT **offset** , **count**;
  - The **offset** specifies the offset of the first row to return. The offset of the first row is 0, not 1
  - The **count** specifies the maximum number of rows to return

SQL LIMIT Clause (Cont.)

- Below is the query which retrieves only third record from students table:



SQL NULL Values

proschool
An ims Initiative

- A field with a NULL value is a field with no value

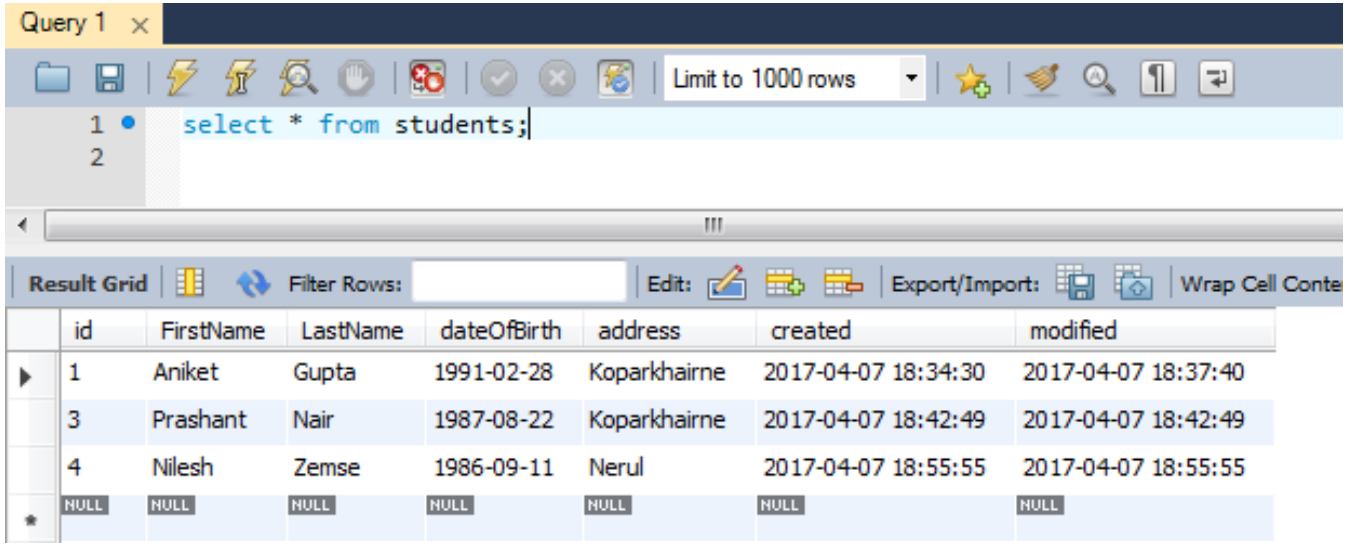
- If a field in a table is optional, it is possible to insert a new record or update a record without adding a value to this field. Then, the field will be saved with a NULL value

- It is very important to understand that a NULL value is different from a zero value or a field that contains spaces. A field with a NULL value is one that has been left blank during record creation

- We will have to use IS NULL and IS NOT NULL operators for checking if a column value is NULL or NOT NULL

SQL NULL Values (Cont.)

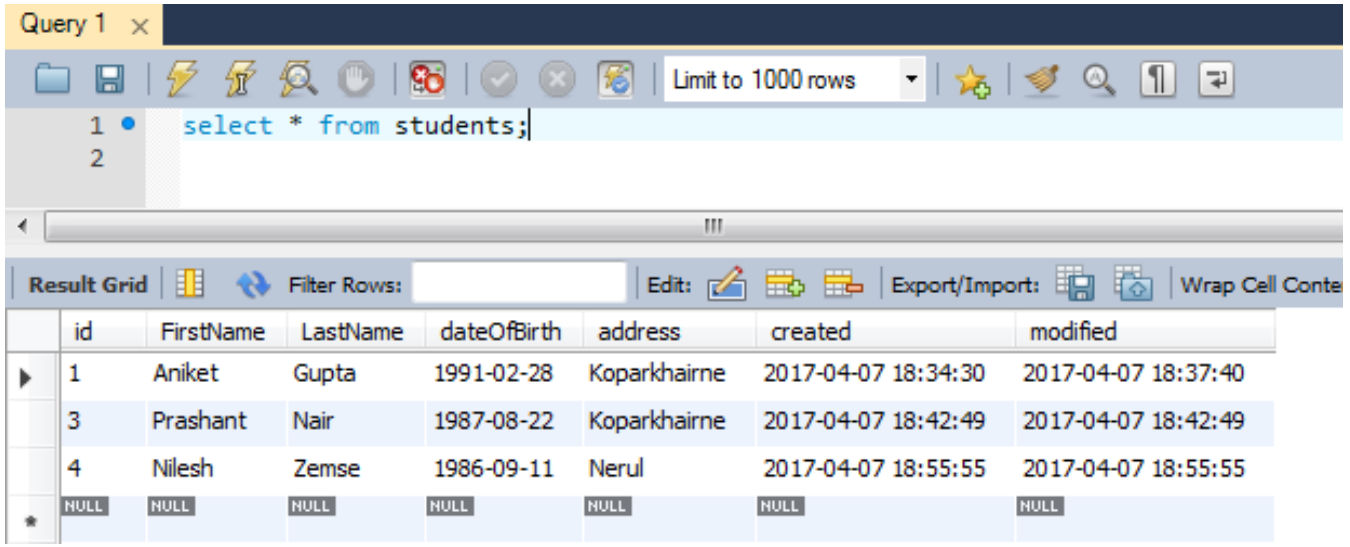
- The **SELECT DISTINCT** statement is used to return only distinct (different) values
- Inside a table, a column often contains many duplicate values and sometimes you only want to list the different (distinct) values
- Below is the current table data:



id	FirstName	LastName	dateOfBirth	address	created	modified
1	Aniket	Gupta	1991-02-28	Koparkhairne	2017-04-07 18:34:30	2017-04-07 18:37:40
3	Prashant	Nair	1987-08-22	Koparkhairne	2017-04-07 18:42:49	2017-04-07 18:42:49
4	Nilesh	Zemse	1986-09-11	Nerul	2017-04-07 18:55:55	2017-04-07 18:55:55
NULL	NULL	NULL	NULL	NULL	NULL	NULL

SELECT DISTINCT Statement

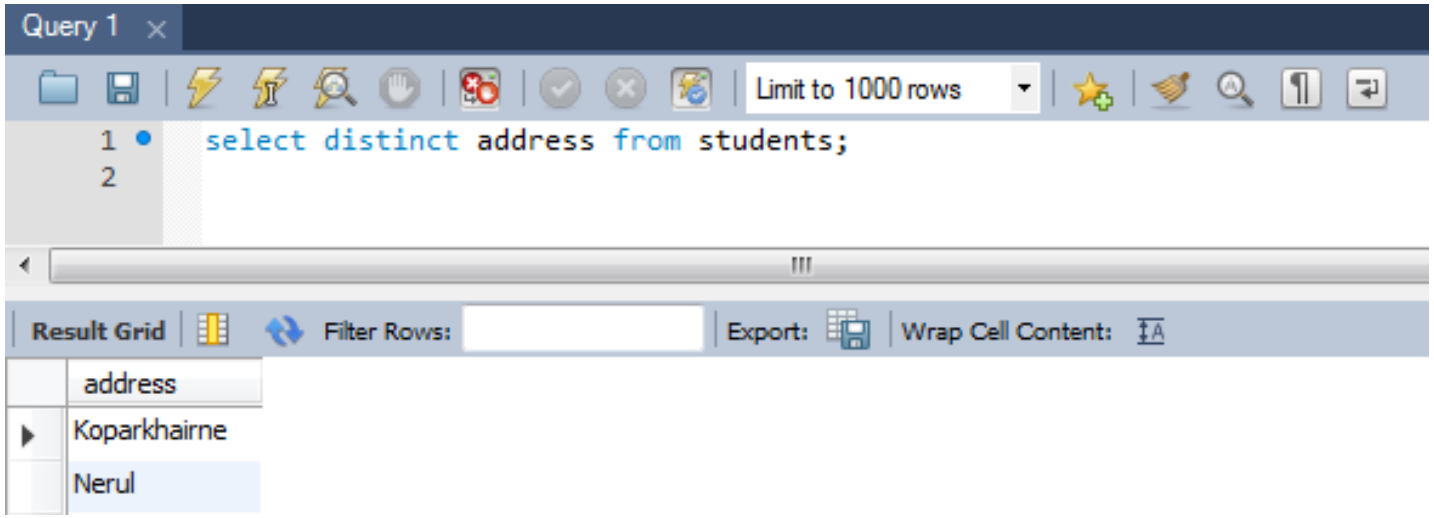
- The **SELECT DISTINCT** statement is used to return only distinct (different) values.
- Inside a table, a column often contains many duplicate values and sometimes you only want to list the different (distinct) values.
- Below is the current table data:



id	FirstName	LastName	dateOfBirth	address	created	modified
1	Aniket	Gupta	1991-02-28	Koparkhairne	2017-04-07 18:34:30	2017-04-07 18:37:40
3	Prashant	Nair	1987-08-22	Koparkhairne	2017-04-07 18:42:49	2017-04-07 18:42:49
4	Nilesh	Zemse	1986-09-11	Nerul	2017-04-07 18:55:55	2017-04-07 18:55:55
NULL	NULL	NULL	NULL	NULL	NULL	NULL

SELECT DISTINCT Statement (Cont.)

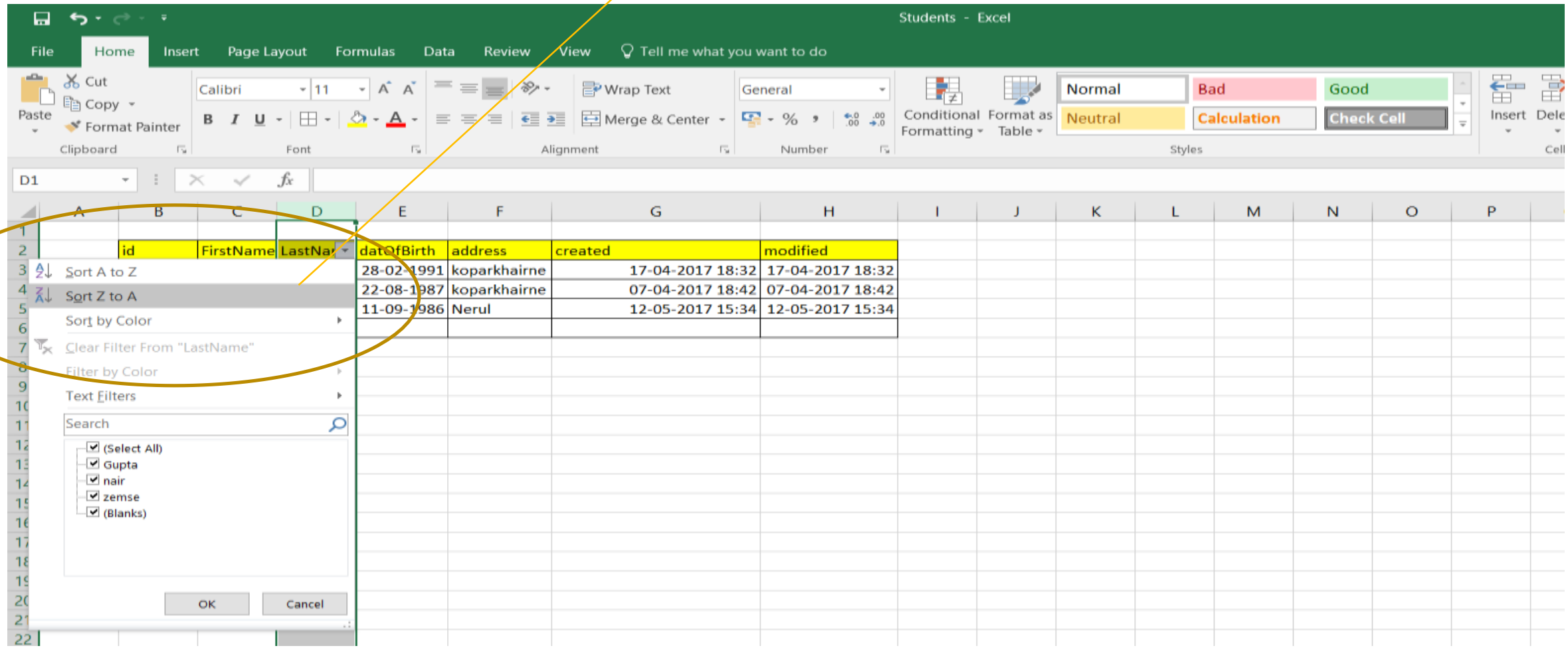
- If we want to find out different address from our previous shown students table. Below would be the query for it



SQL ORDER by Keyword

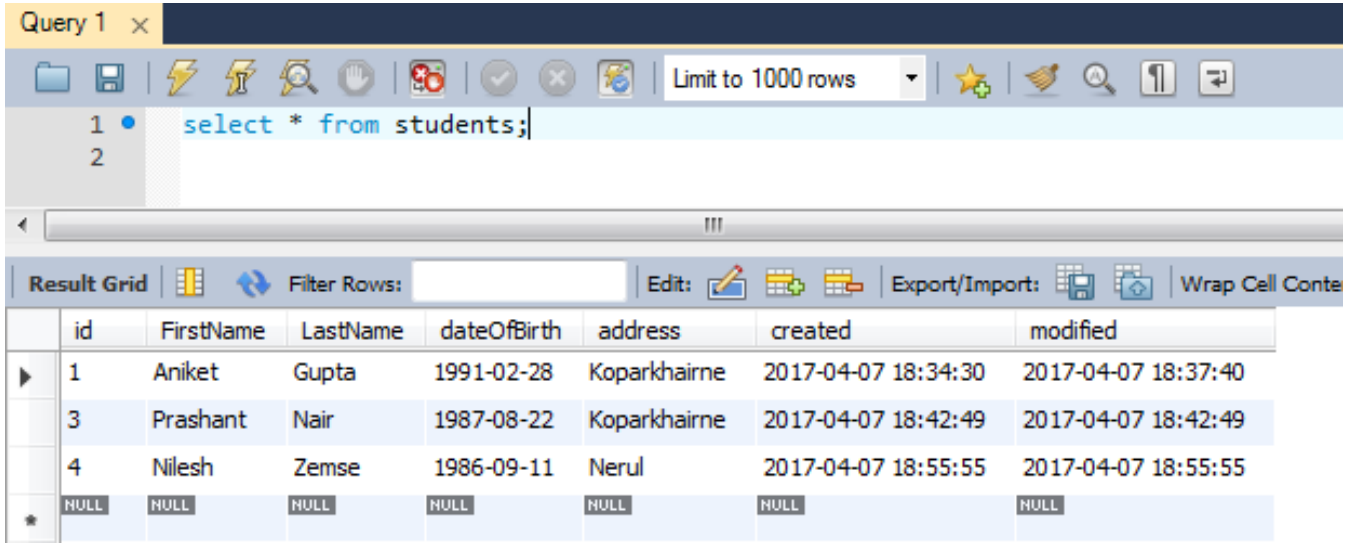
proschool
An ims Initiative

- If you want to sort lastName column in descending order in Excel



SQL ORDER by Keyword

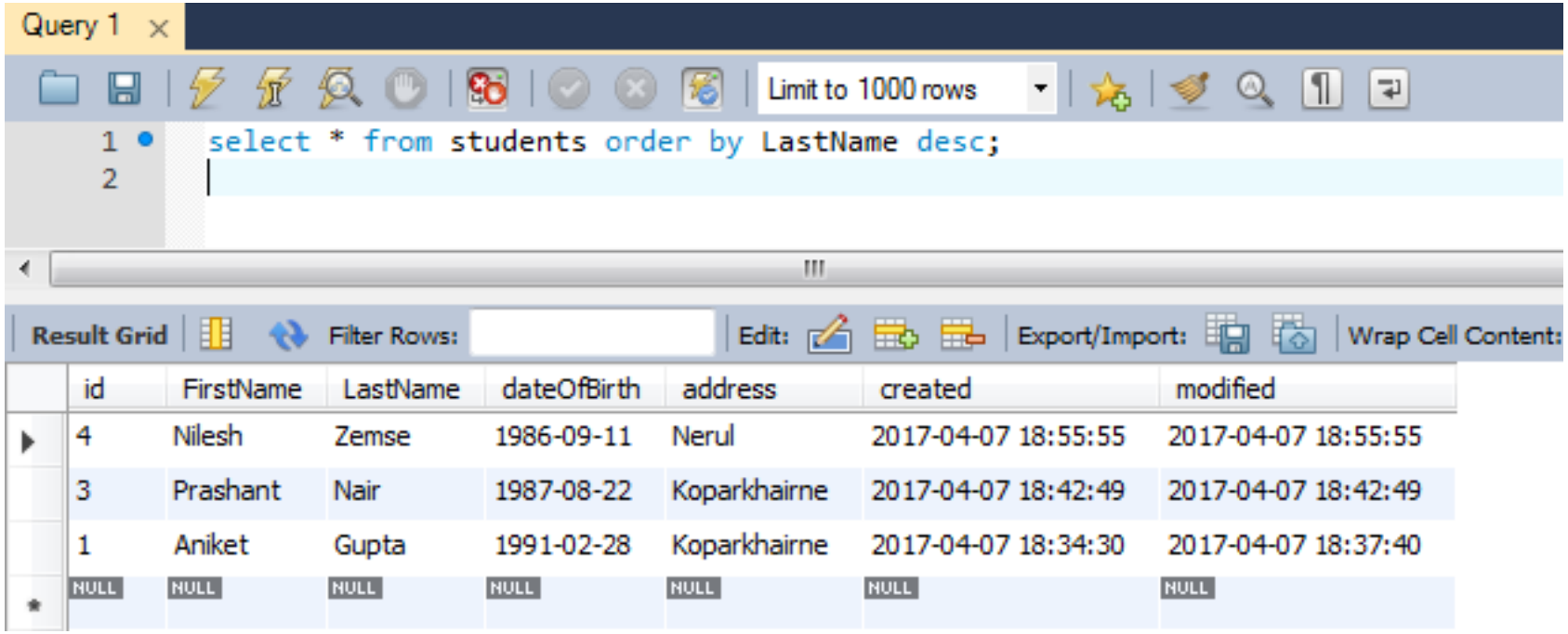
- The **ORDER BY** keyword is used to sort the result-set in ascending or descending order
- The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword
- Our current data set is:



id	FirstName	LastName	dateOfBirth	address	created	modified
1	Aniket	Gupta	1991-02-28	Koparkhairne	2017-04-07 18:34:30	2017-04-07 18:37:40
3	Prashant	Nair	1987-08-22	Koparkhairne	2017-04-07 18:42:49	2017-04-07 18:42:49
4	Nilesh	Zemse	1986-09-11	Nerul	2017-04-07 18:55:55	2017-04-07 18:55:55
NULL	NULL	NULL	NULL	NULL	NULL	NULL

SQL ORDER by Keyword (Cont.)

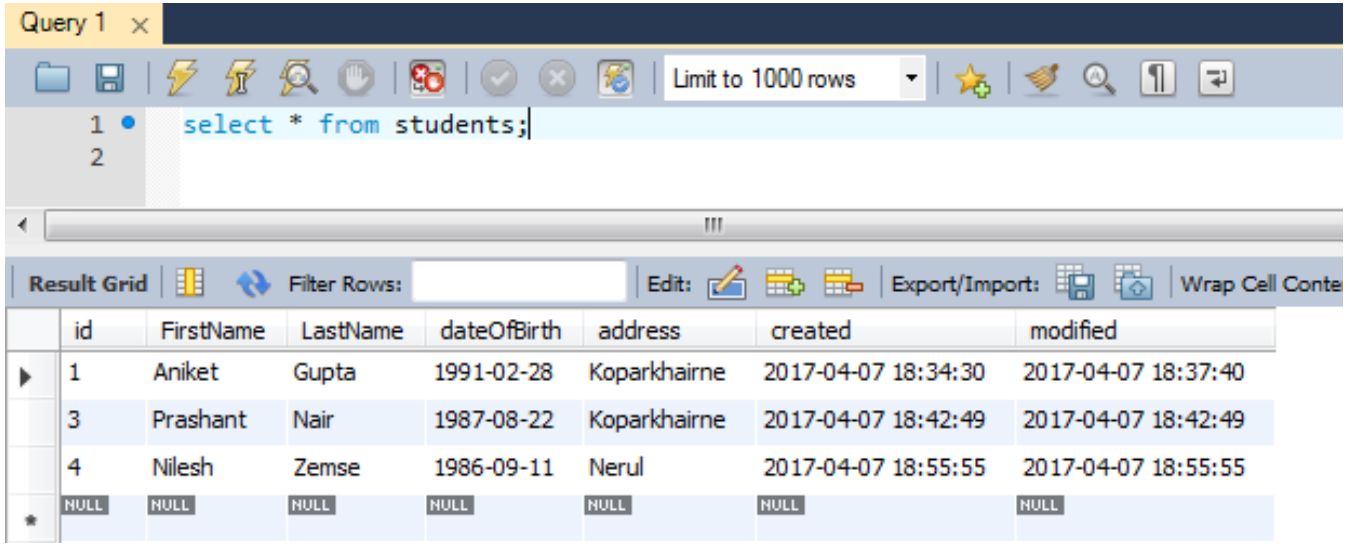
- If we want to find out all the records from students table sorted by lastName column in descending order. This can be stated as



id	FirstName	LastName	dateOfBirth	address	created	modified
4	Nilesh	Zemse	1986-09-11	Nerul	2017-04-07 18:55:55	2017-04-07 18:55:55
3	Prashant	Nair	1987-08-22	Koparkhairne	2017-04-07 18:42:49	2017-04-07 18:42:49
1	Aniket	Gupta	1991-02-28	Koparkhairne	2017-04-07 18:34:30	2017-04-07 18:37:40
NULL	NULL	NULL	NULL	NULL	NULL	NULL

SQL WHERE Clause

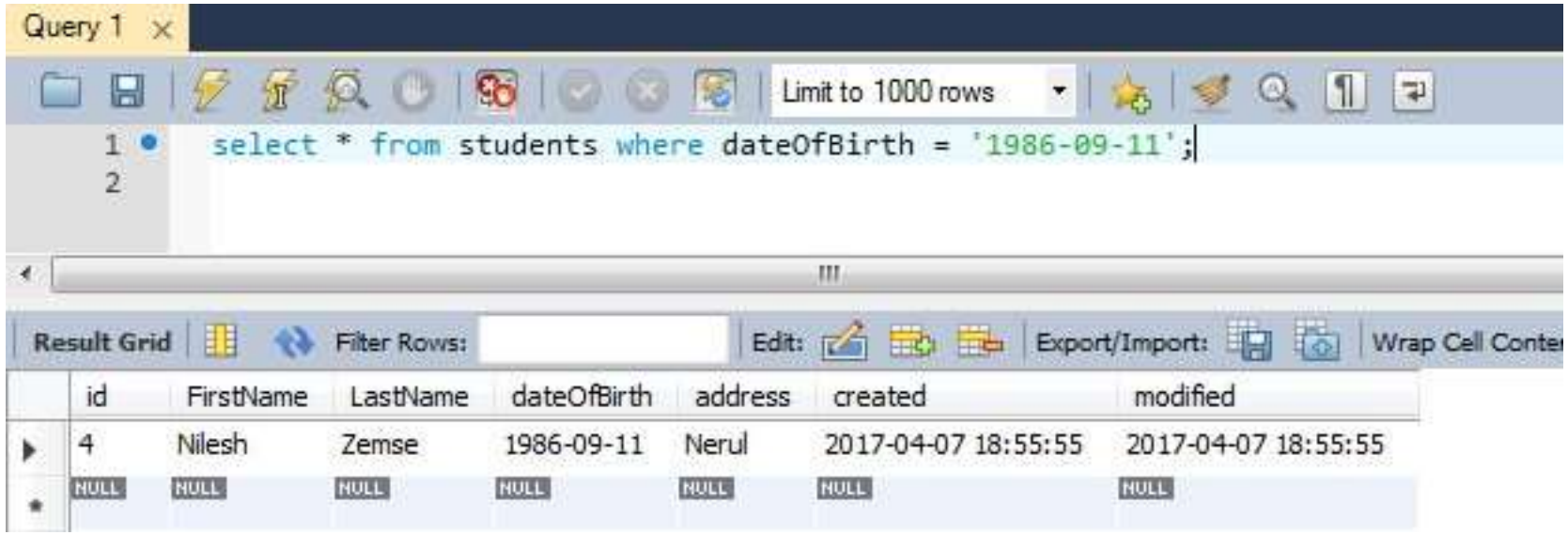
- The WHERE clause is used to filter records
- The SQL WHERE clause is used to specify a condition while fetching the data from single table or joining with multiple tables
- The WHERE clause is not only used in SELECT statement, but it is also used in UPDATE, DELETE statement, etc.
- Below is the current table data:



id	FirstName	LastName	dateOfBirth	address	created	modified
1	Aniket	Gupta	1991-02-28	Koparkhairne	2017-04-07 18:34:30	2017-04-07 18:37:40
3	Prashant	Nair	1987-08-22	Koparkhairne	2017-04-07 18:42:49	2017-04-07 18:42:49
4	Nilesh	Zemse	1986-09-11	Nerul	2017-04-07 18:55:55	2017-04-07 18:55:55
NULL	NULL	NULL	NULL	NULL	NULL	NULL

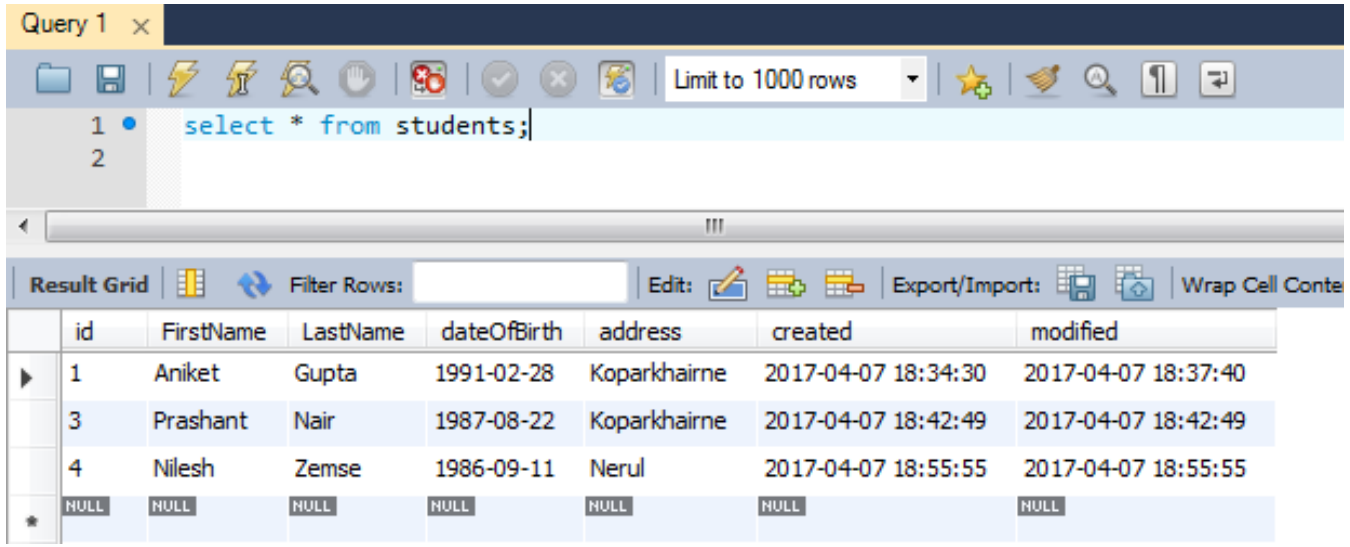
SQL WHERE Clause (Cont.)

- If we want to find out all the details of a student whose date of birth is ‘1986-09-11’. Below would be the query for it:



SQL GROUP BY statement

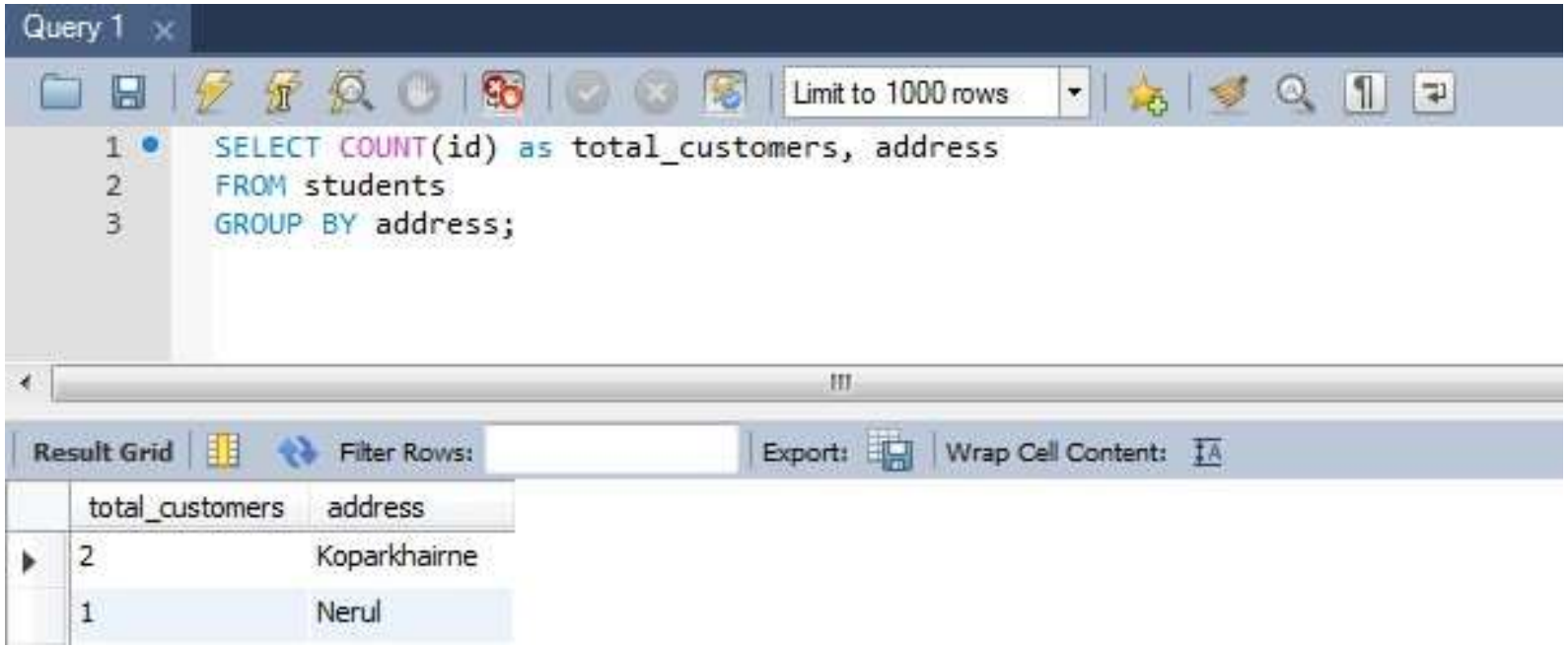
- The GROUP BY statement is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns
- The GROUP BY clause follows the WHERE clause in a SELECT statement and precedes the ORDER BY clause
- Below is the current table data:



id	FirstName	LastName	dateOfBirth	address	created	modified
1	Aniket	Gupta	1991-02-28	Koparkhairne	2017-04-07 18:34:30	2017-04-07 18:37:40
3	Prashant	Nair	1987-08-22	Koparkhairne	2017-04-07 18:42:49	2017-04-07 18:42:49
4	Nilesh	Zemse	1986-09-11	Nerul	2017-04-07 18:55:55	2017-04-07 18:55:55
NULL	NULL	NULL	NULL	NULL	NULL	NULL

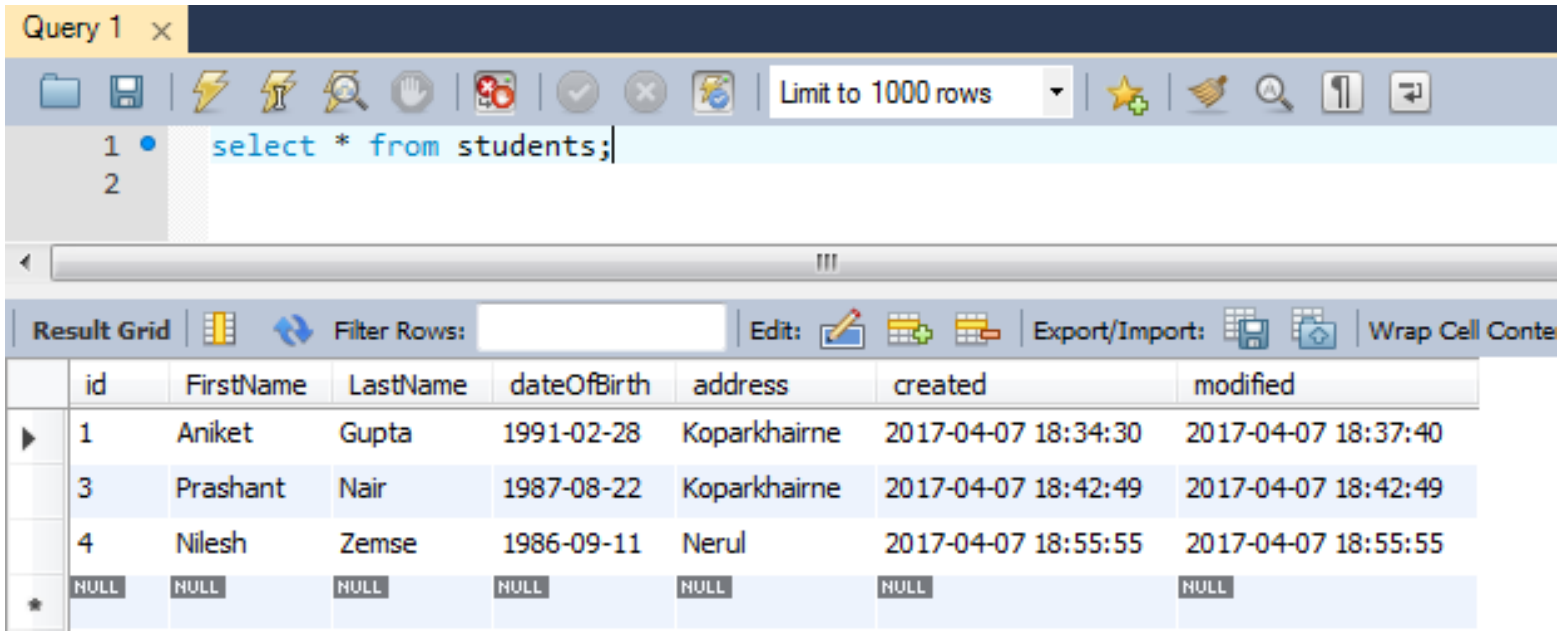
SQL GROUP BY statement (Cont.)

- The below query would list the total number of customers in each address



SQL - Having Clause

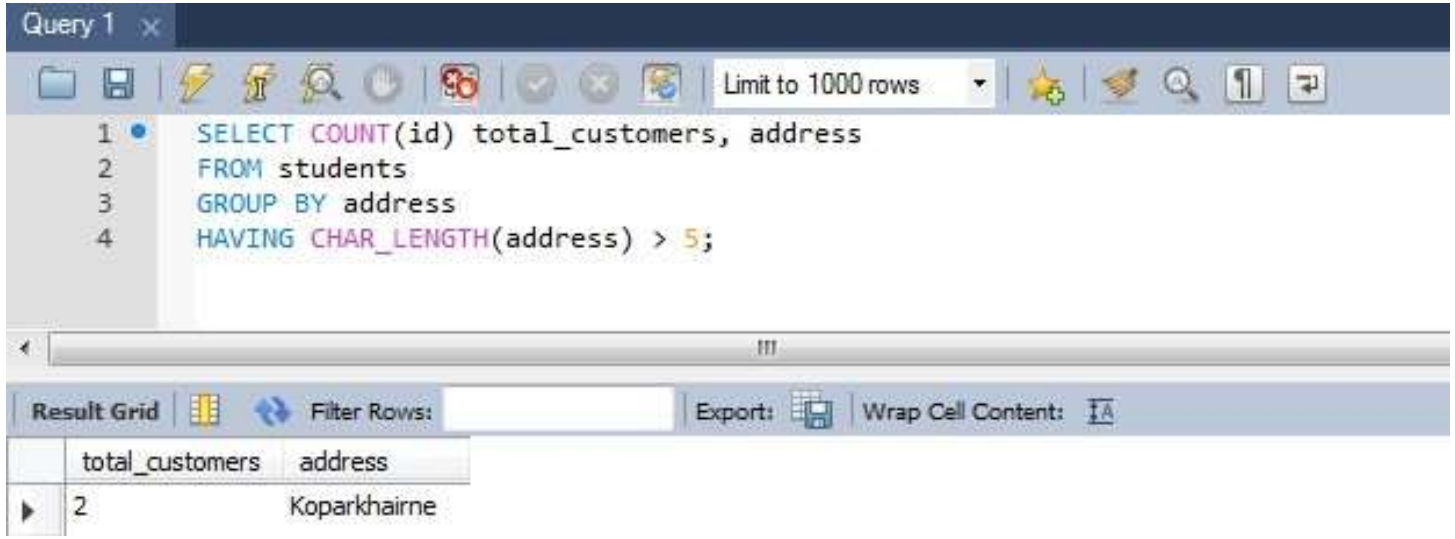
- The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions
- The WHERE clause places conditions on the selected columns, whereas the HAVING clause places conditions on groups created by the GROUP BY clause
- Below is the current table data:



id	FirstName	LastName	dateOfBirth	address	created	modified
1	Aniket	Gupta	1991-02-28	Koparkhairne	2017-04-07 18:34:30	2017-04-07 18:37:40
3	Prashant	Nair	1987-08-22	Koparkhairne	2017-04-07 18:42:49	2017-04-07 18:42:49
4	Nilesh	Zemse	1986-09-11	Nerul	2017-04-07 18:55:55	2017-04-07 18:55:55
NULL	NULL	NULL	NULL	NULL	NULL	NULL

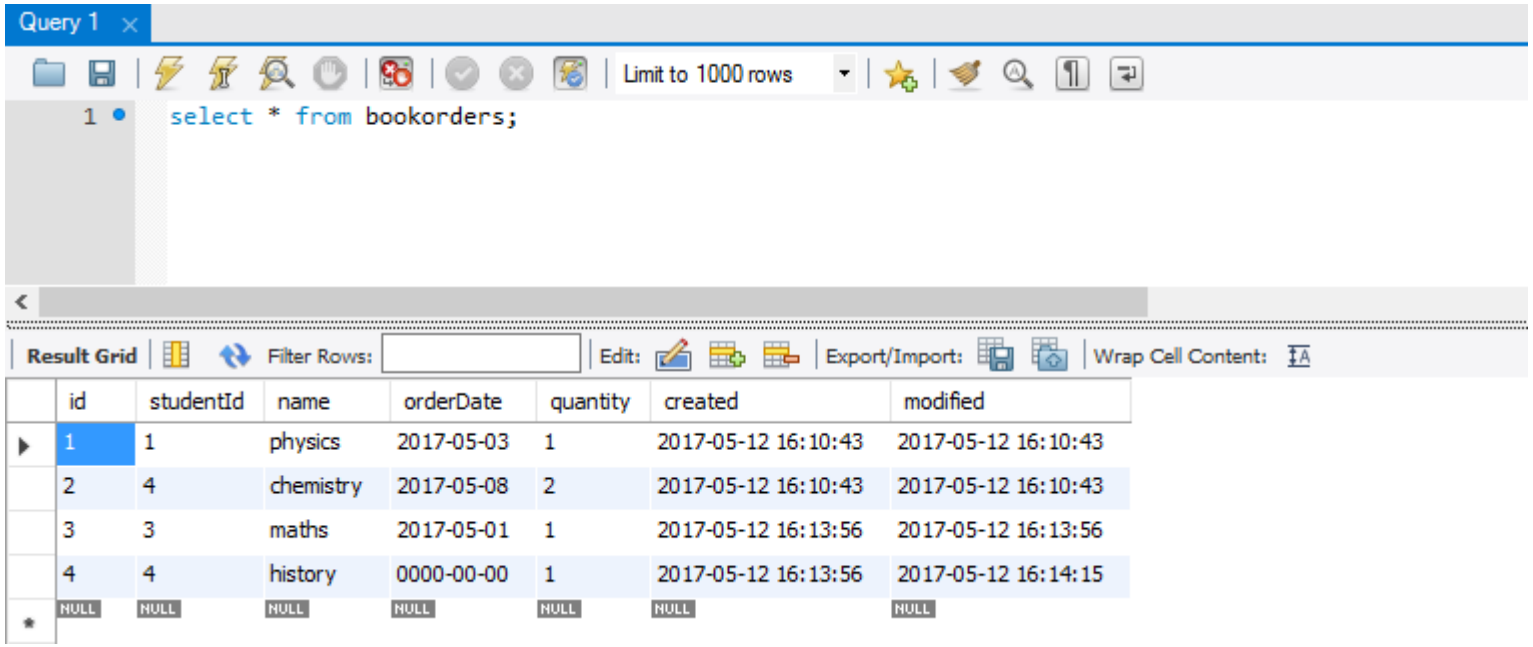
SQL - Having Clause (Cont.)

- The below mentioned SQL statement lists the number of students in each address. Only include address with character length more than 5:



SQL – Using Joins

- A JOIN clause is used to combine rows from two or more tables, based on a related column between them
- We will create a new table as “Books Order” from which we will fetch data of students who have placed books orders using joins command
- Below is the books order table:



id	studentId	name	orderDate	quantity	created	modified
1	1	physics	2017-05-03	1	2017-05-12 16:10:43	2017-05-12 16:10:43
2	4	chemistry	2017-05-08	2	2017-05-12 16:10:43	2017-05-12 16:10:43
3	3	maths	2017-05-01	1	2017-05-12 16:13:56	2017-05-12 16:13:56
4	4	history	0000-00-00	1	2017-05-12 16:13:56	2017-05-12 16:14:15
NULL	NULL	NULL	NULL	NULL	NULL	NULL

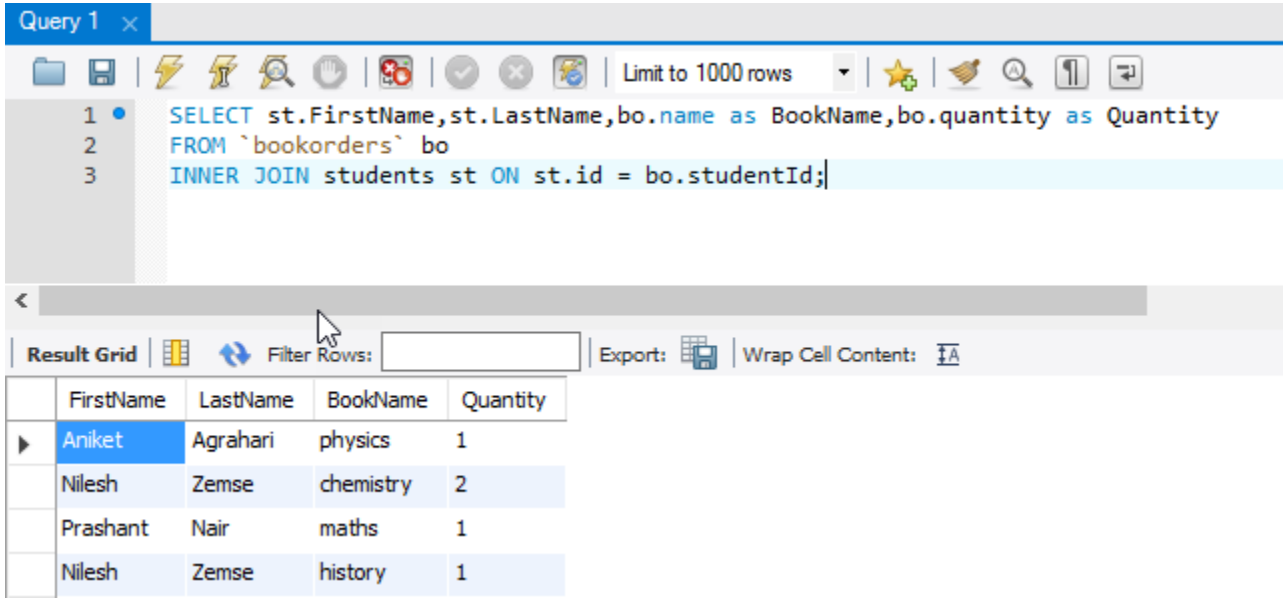
SQL – Using Joins (Cont.)

- Notice that the “studentId" column in the “Book Orders" table refers to the “id" in the “students" table. The relationship between the two tables above is the “studentId" column
- **Different Types of SQL JOINs**
  - **(INNER) JOIN:** Returns records that have matching values in both tables
  - **LEFT (OUTER) JOIN**: Return all records from the left table, and the matched records from the right table
  - **RIGHT (OUTER) JOIN**: Return all records from the right table, and the matched records from the left table
  - **SELF JOIN:** It is used to join a table to itself as if the table were two tables, temporarily renaming at least one table in the SQL statement
  - **FULL (OUTER) JOIN**: Return all records when there is a match in either left or right table

SQL – Inner Join (Cont.)

- The **INNER JOIN** keyword selects records that have matching values in both tables

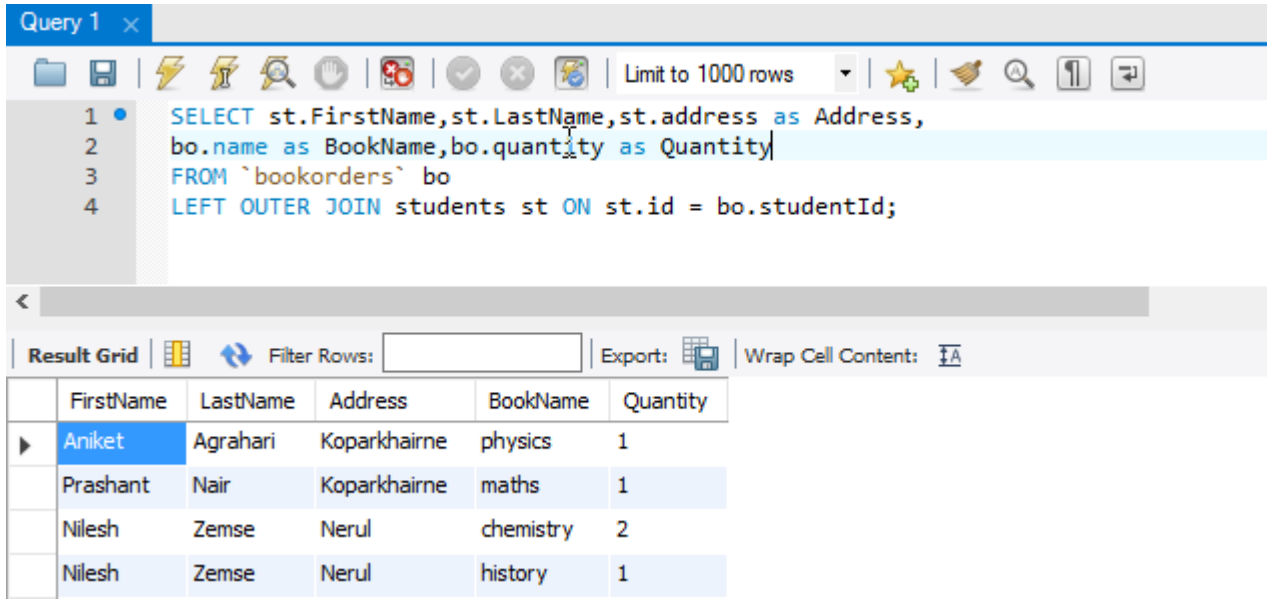
- The **INNER JOIN** keyword selects all rows from both tables as long as there is a match between the columns. If there are records in the “Book Orders" table that do not have matches in “Students", the orders will not be shown



SQL – Left Join (Cont.)

- The **LEFT JOIN** keyword returns all records from the left table (table1), and the matched records from the right table (table2). The result is NULL from the right side, if there is no match

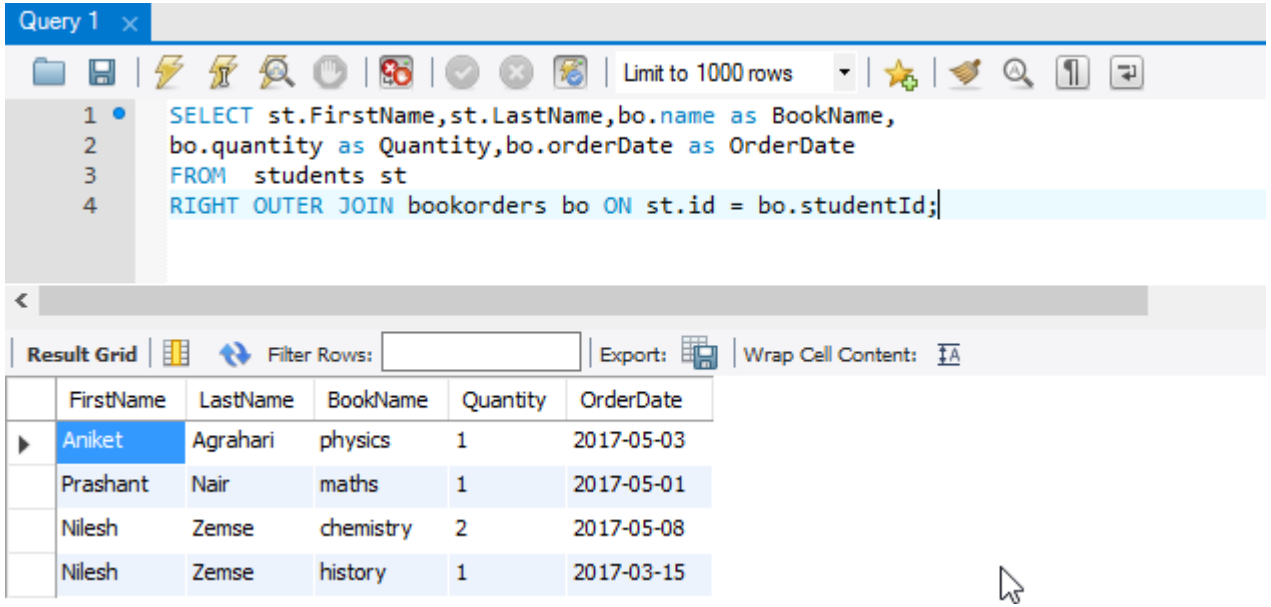
- Below query will select all students, address and the orders



SQL – Right Join (Cont.)

- The **RIGHT JOIN** keyword returns all records from the right table (table2), and the matched records from the left table (table1). The result is NULL from the left side, when there is no match

- Below query will select all students, order date and any orders specified



SQL – Self Join (Cont.)

- A self join is a join in which a table is joined with itself

- Below query will select all workers, which are from the same department

SQL File 3* | SQL File 5* | SQL File 6*

Limit to 50 rows

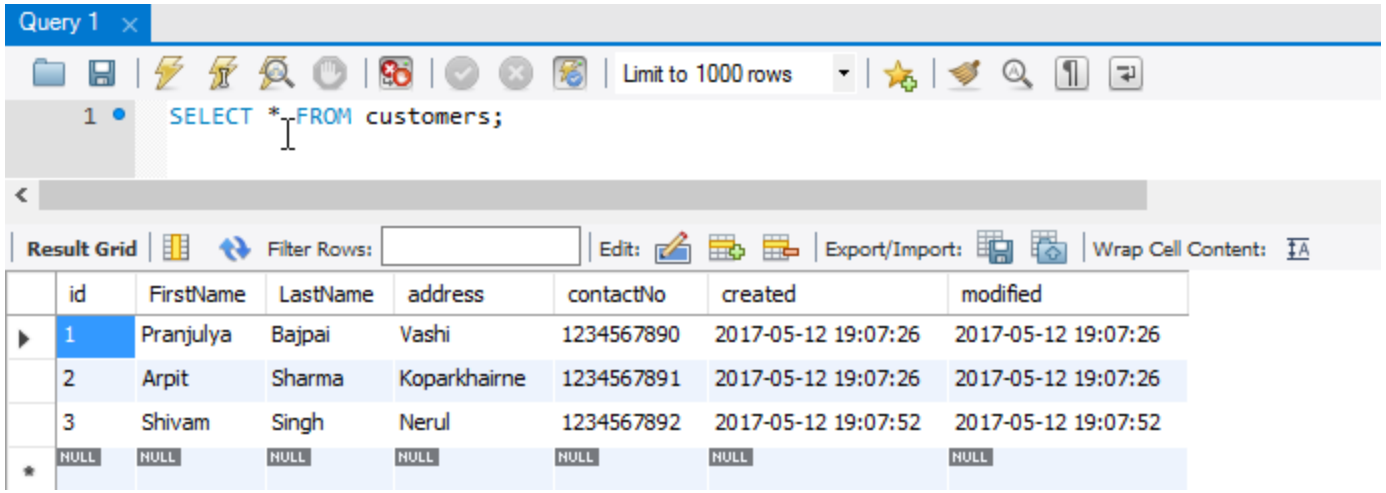
```
1  SELECT w1.FIRST_NAME AS worker1, w2.FIRST_NAME AS worker2, w1.DEPARTMENT
2  FROM workers w1, workers w2
3  WHERE w1.WORKER_ID <> w2.WORKER_ID
4  AND w1.DEPARTMENT = w2.DEPARTMENT
5  ORDER BY w1.DEPARTMENT;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

worker1	worker2	DEPARTMENT
Vipul	Satish	Account
Satish	Vipul	Account
Vivek	Niharika	Admin
Vivek	Geetika	Admin
Amitabh	Vivek	Admin
Vivek	Amitabh	Admin
Geetika	Niharika	Admin
Niharika	Geetika	Admin
Geetika	Vivek	Admin

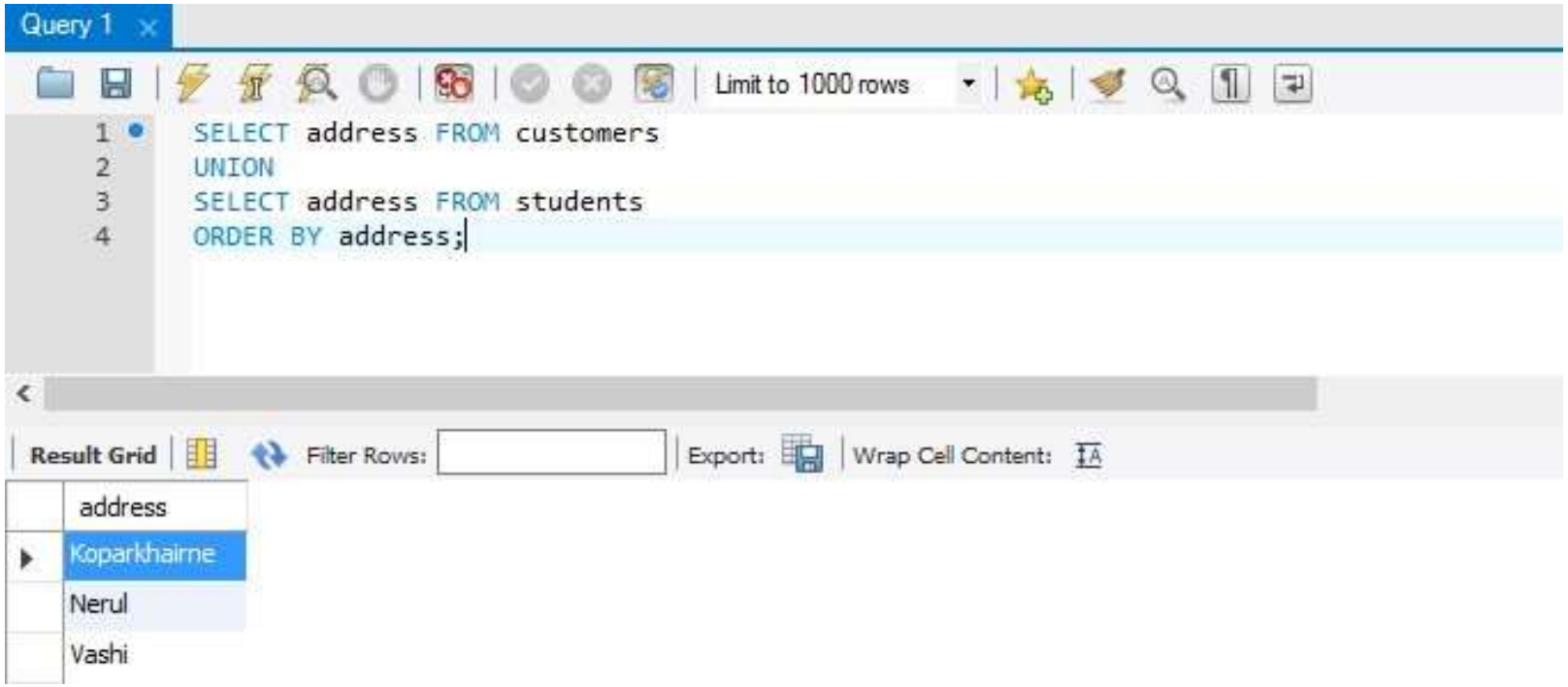
SQL – Union Operator

- The **UNION operator** is used to combine the result-set of two or more SELECT statements
  - Each SELECT statement within UNION must have the same number of columns
  - The columns must also have similar data types
  - The columns in each SELECT statement must also be in the same order
  - It only select distinct values
- We will create a new table as “Customers” from which we will fetch data of customers



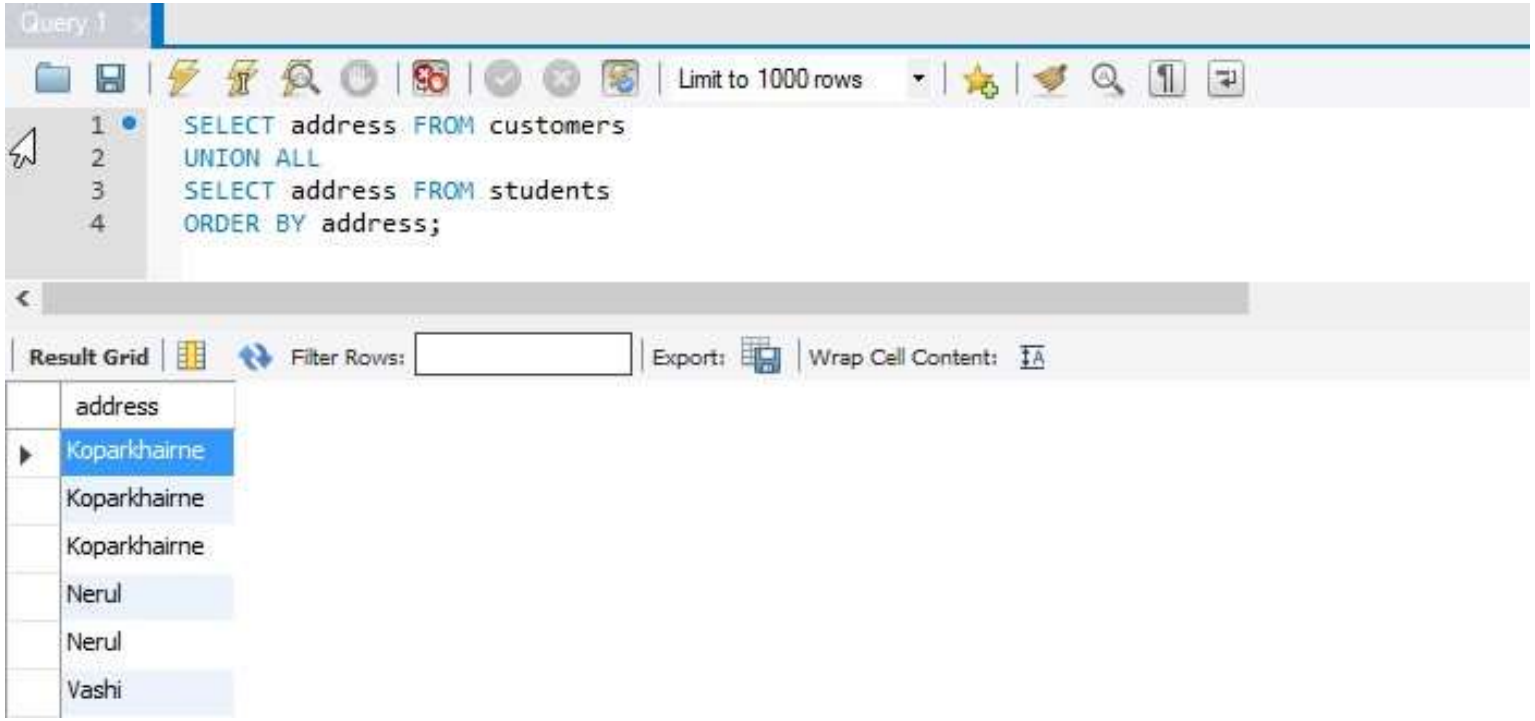
SQL – Union Operator (Cont.)

- The below mentioned SQL statement lists distinct address from both students and customers table



SQL – Union All Operator

- The **UNION ALL** operator is used to combine the results of two SELECT statements **including duplicate rows**
- The below mentioned SQL statement lists all address from both students and customers table



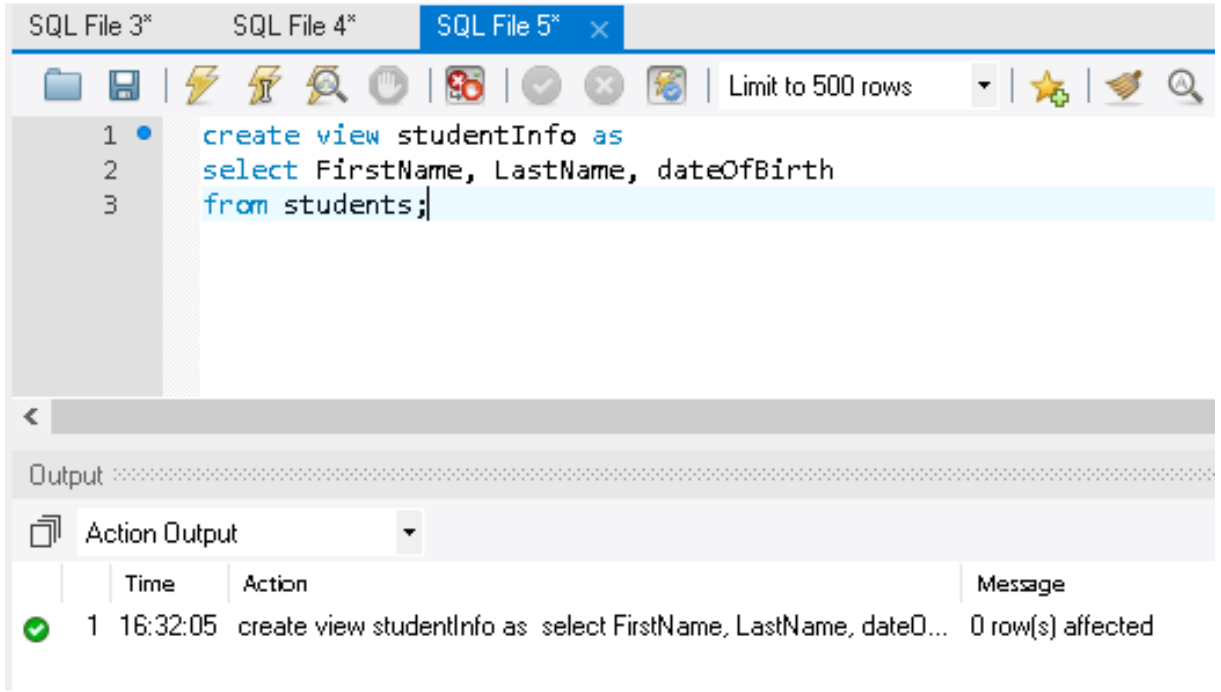
SQL – Views

- A view is a virtual table based on the result-set of an SQL statement
- A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database
- You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table
- Below is the syntax for creating a view:

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

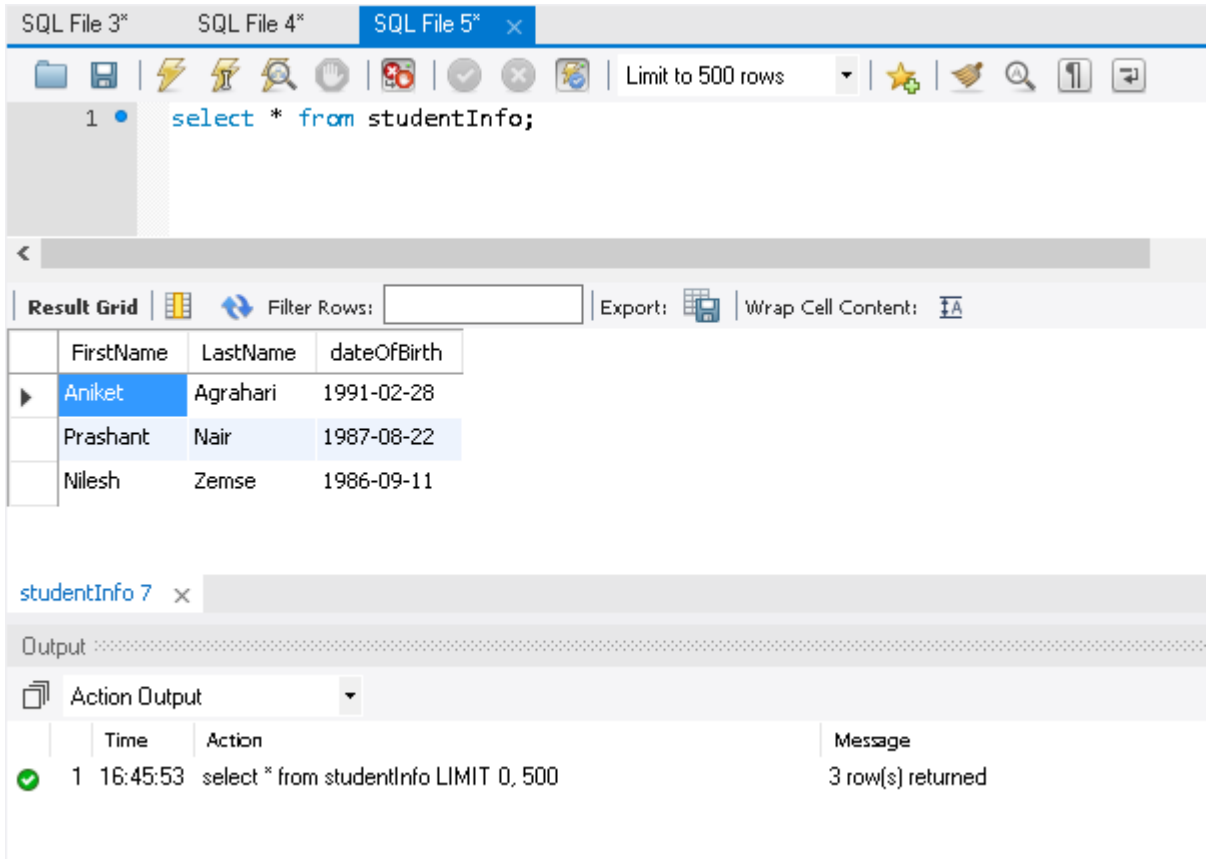
SQL – Views (Cont.)

- **A view always shows up-to-date data! The database engine recreates the data, using the view's SQL statement, every time a user queries a view**
- We have created a view named **studentInfo,** which stores only FirstName, LastName, dateOfBirth of students table



SQL – Views (Cont.)

- Once we have created the view, we can query the view with the following SQL interface:



SQL – Views (Cont.)

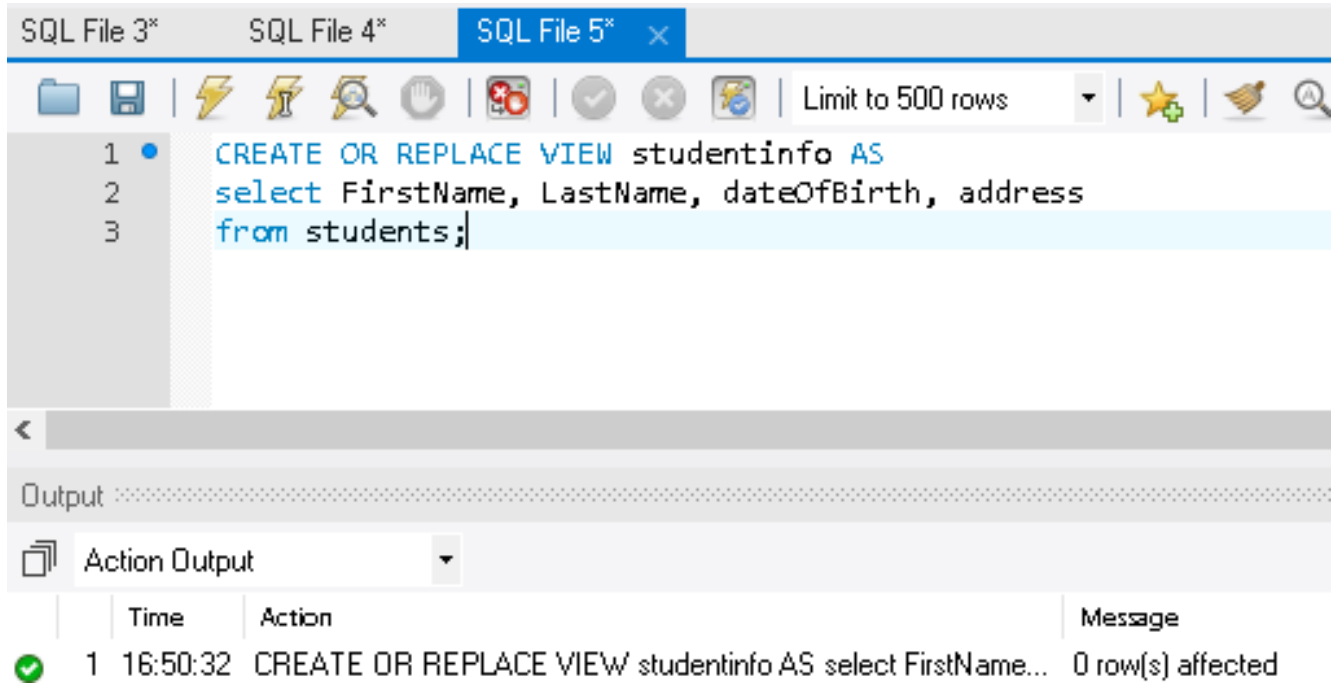
- We can also update a view by using the following syntax:

```
CREATE OR REPLACE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

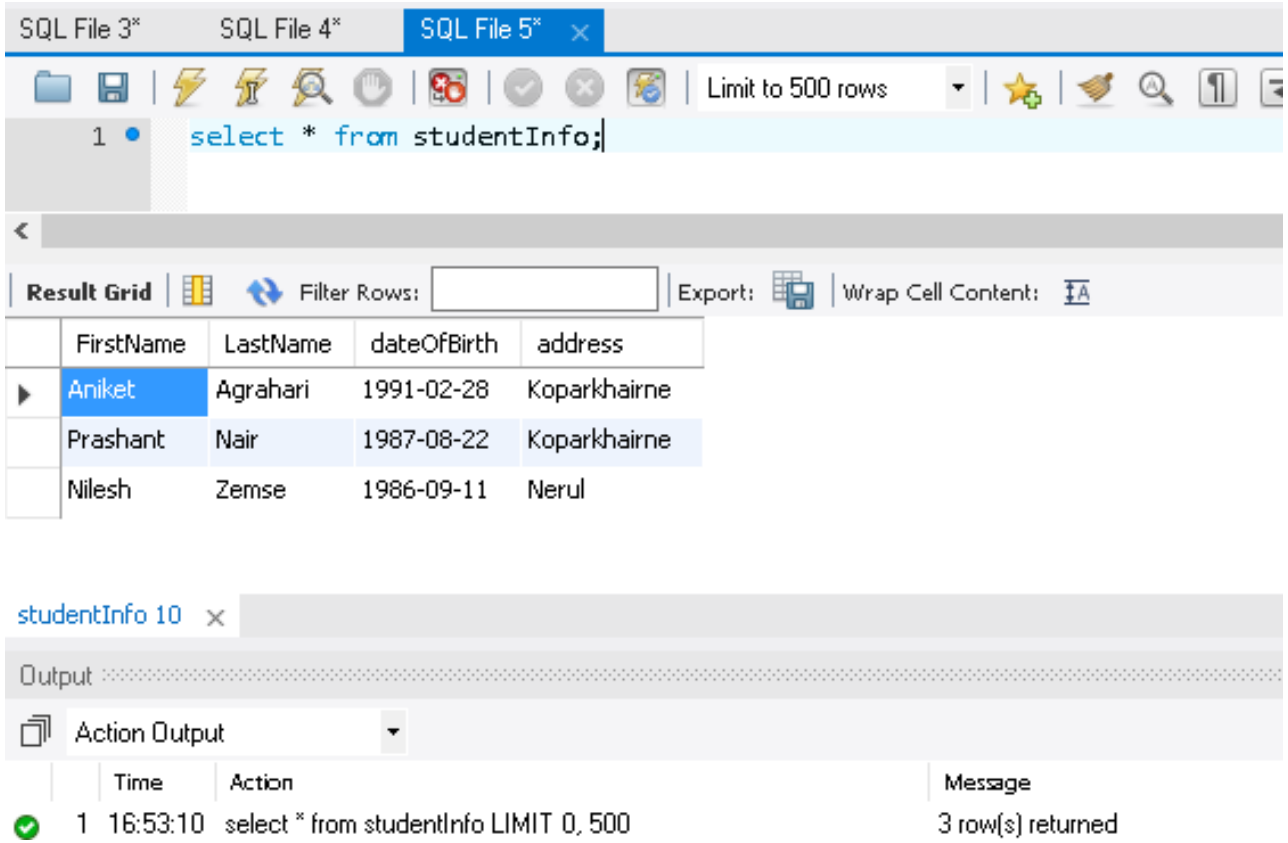
- Now we want to add the “address” column to the “studentInfo” view. We will update the view with The related SQL command

SQL – Views (Cont.)

- Now once we have updated the view we can see the updated result as follows:



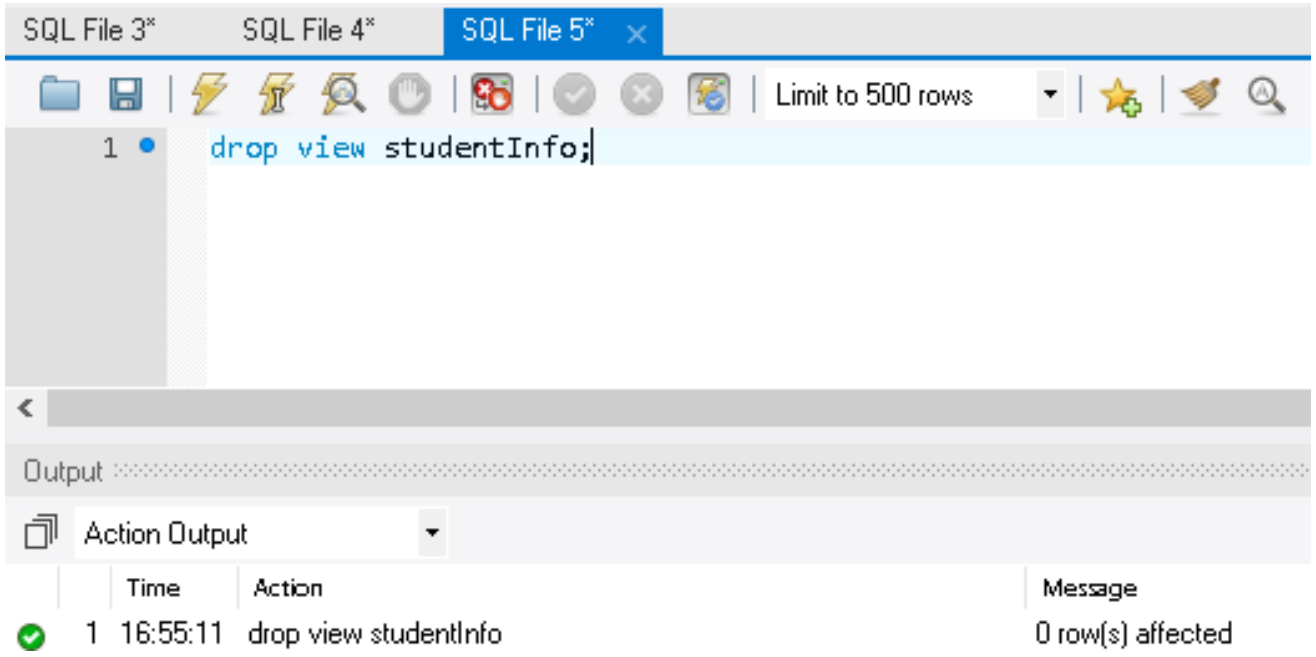
SQL – Views (Cont.)



SQL – Views (Cont.)

- You can delete a view with the DROP VIEW command.

DROP VIEW view_name;



SQL – Temporary Tables

proschool
An ims Initiative

- A TEMPORARY table is visible only within the current session, and is dropped automatically when the session is closed

- This means that two different sessions can use the same temporary table name without conflicting with each other

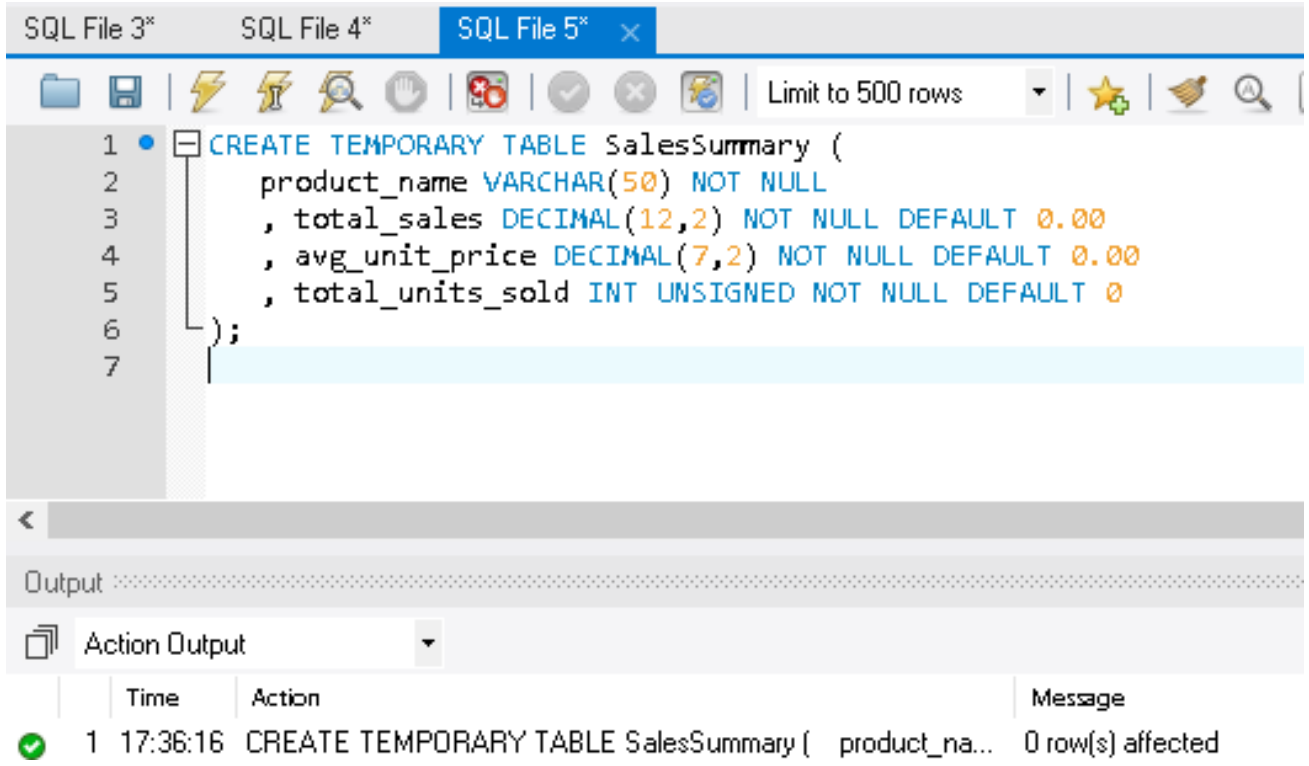
- The temporary tables could be very useful in some cases to keep temporary data.
- Temporary tables were added in the MySQL Version 3.23

- If you are connected to the MySQL database server through the MySQL client program, then the temporary table will exist until you close the client or manually destroy the table

- To create a temporary table, you must have the **CREATE TEMPORARY TABLES** privilege

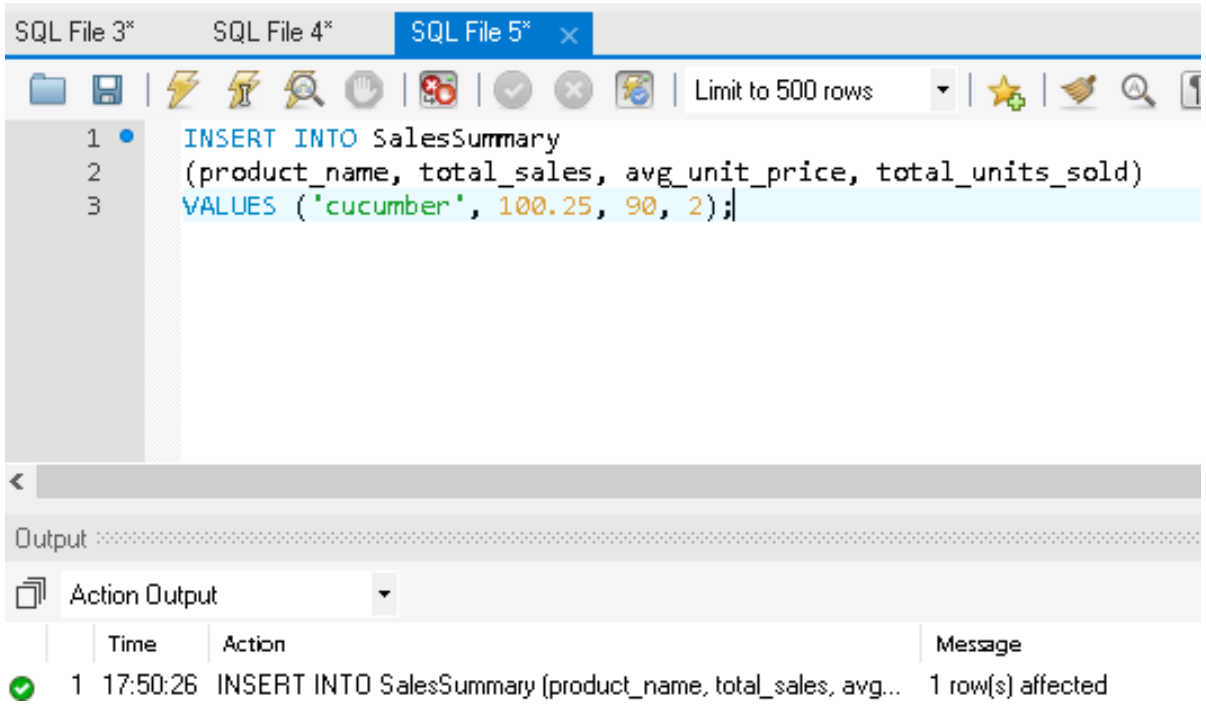
SQL – Temporary Tables (Cont.)

- Below we have created the temporary table named as “SalesSummary”:



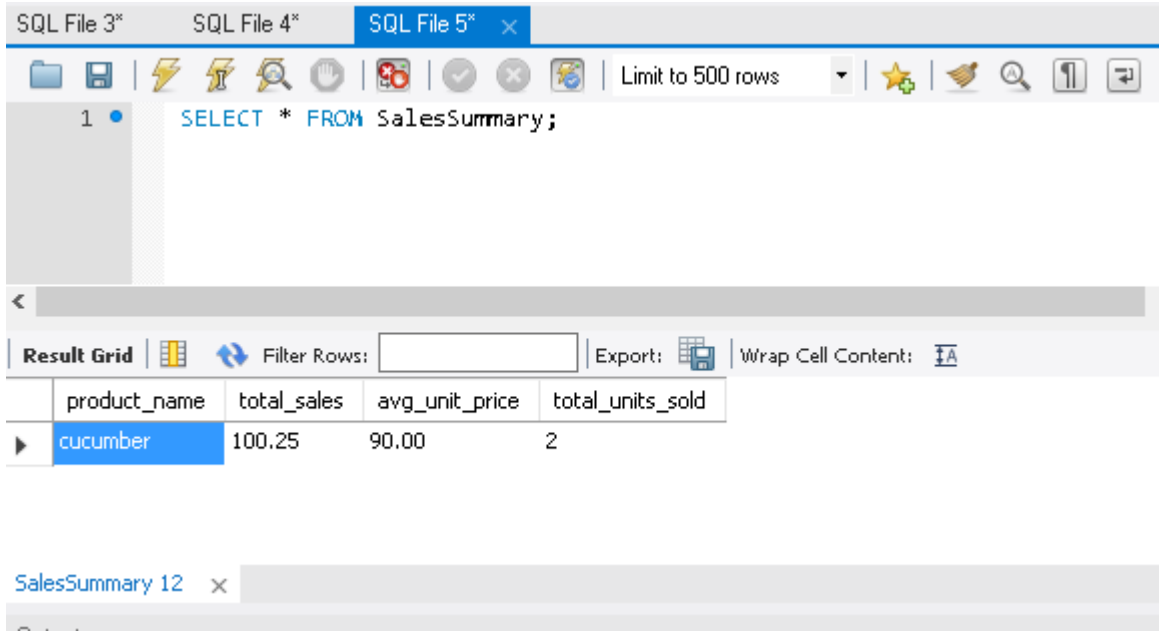
SQL – Temporary Tables(Cont.)

- Next we have inserted a record in a table:



SQL – Temporary Tables (Cont.)

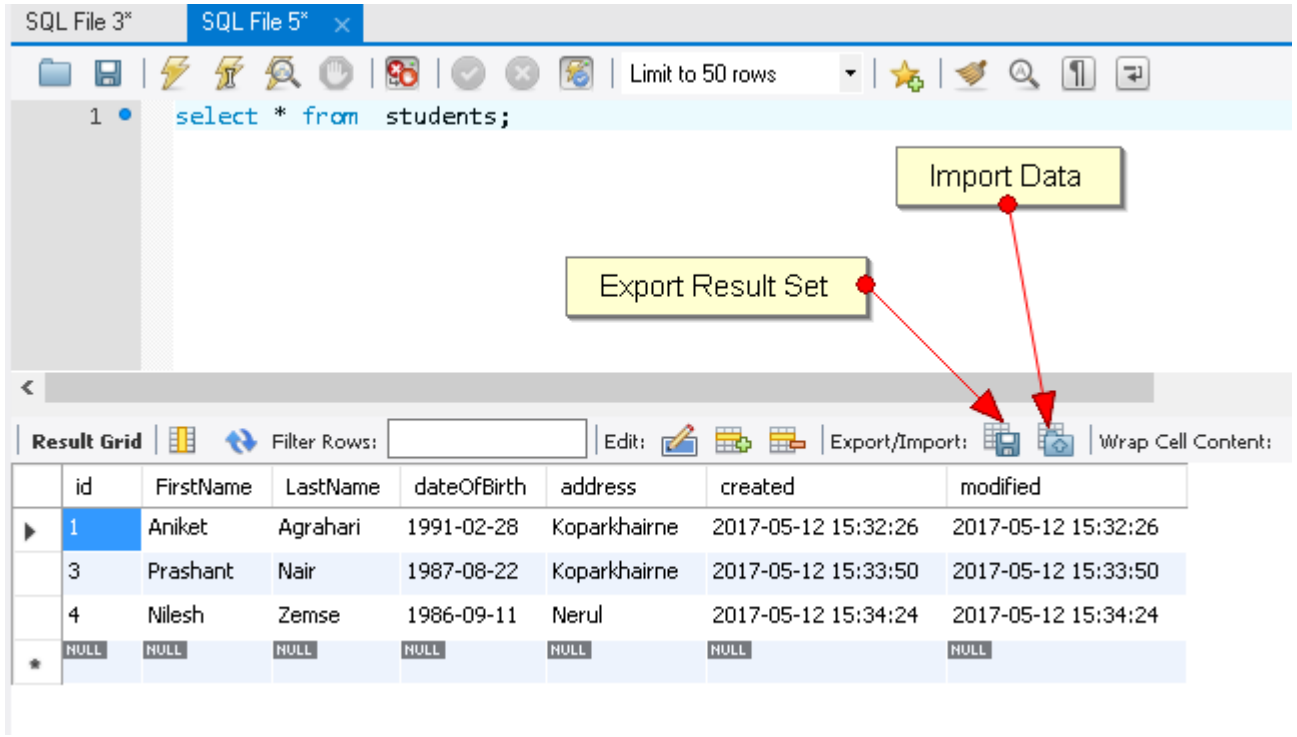
- Now when we select the data from the table, we get one record



- By default, all the temporary tables are deleted by MySQL when your database connection gets terminated. Still if you want to delete them in between, then you can do it by issuing the **DROP TABLE** command

SQL – Import/Export

- **Export a Result Set**
  - A result set in the workbench can be exported to common file formats including CSV, JSON, HTML, and XML
- **Import a Result Set**
  - Records from a CSV file can be imported into the result set of the workbench



Stored Procedures in SQL

proschool
An ims Initiative

- A procedure is a PL/SQL block which performs one or more specific tasks stored in a database
- The procedure contains a header and a body
  - **Header**: The header contains the name of the procedure and the parameters or variables passed to the procedure
  - **Body**: The body contains a declaration section, execution section and exception section similar to a general PL/SQL block
- When you want to create a procedure, you have to define parameters. There is three ways to pass parameters in procedure:
  - **IN parameters**: The IN parameter can be referenced by the procedure. The value of the parameter cannot be overwritten by the procedure
  - **OUT parameters**: The OUT parameter cannot be referenced by the procedure, but the value of the parameter can be overwritten by the procedure

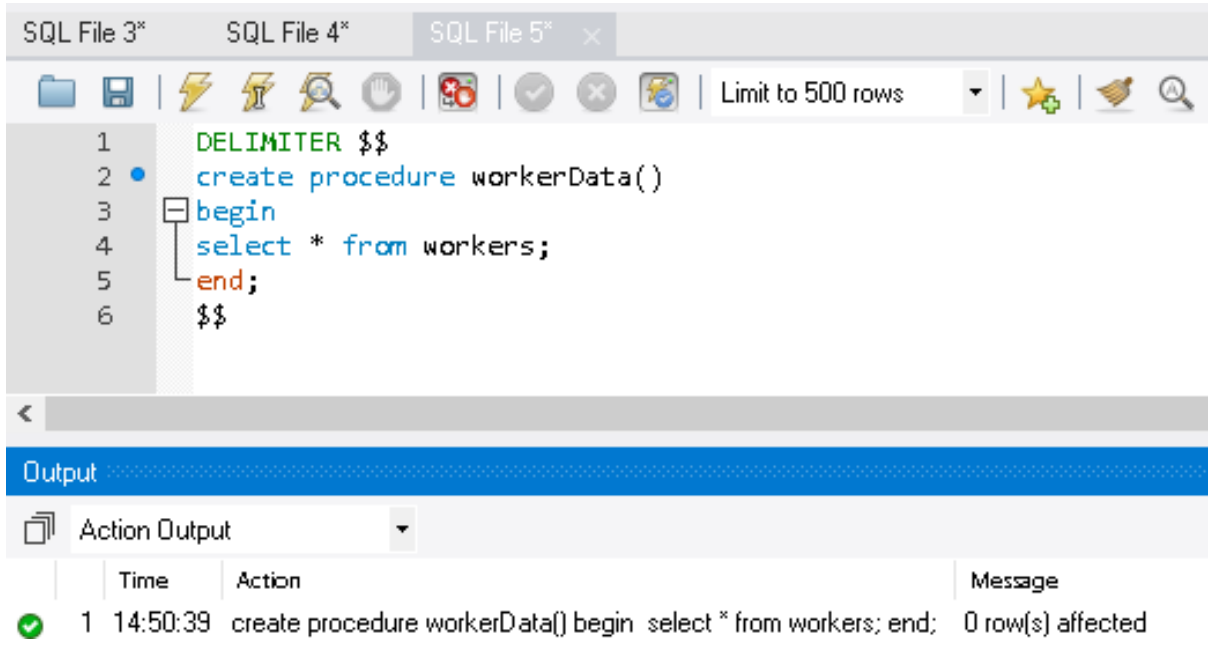
Stored Procedures in SQL (Cont.)

- **INOUT parameters**: The INOUT parameter can be referenced by the procedure and the value of the parameter can be overwritten by the procedure
- **A procedure may or may not return any value**
- Below is the syntax for creating the procedure:

```
CREATE [OR REPLACE] PROCEDURE procedure_name
                [ (parameter [,parameter]) ]
        IS
                [declaration_section]
        BEGIN
                executable_section
        [EXCEPTION
                exception_section]
        END [procedure_name];
```

Stored Procedures in SQL (Cont.)

- We will create a simple procedure called **workerData**, when we will execute the procedure it will display all the data from “workers" tables

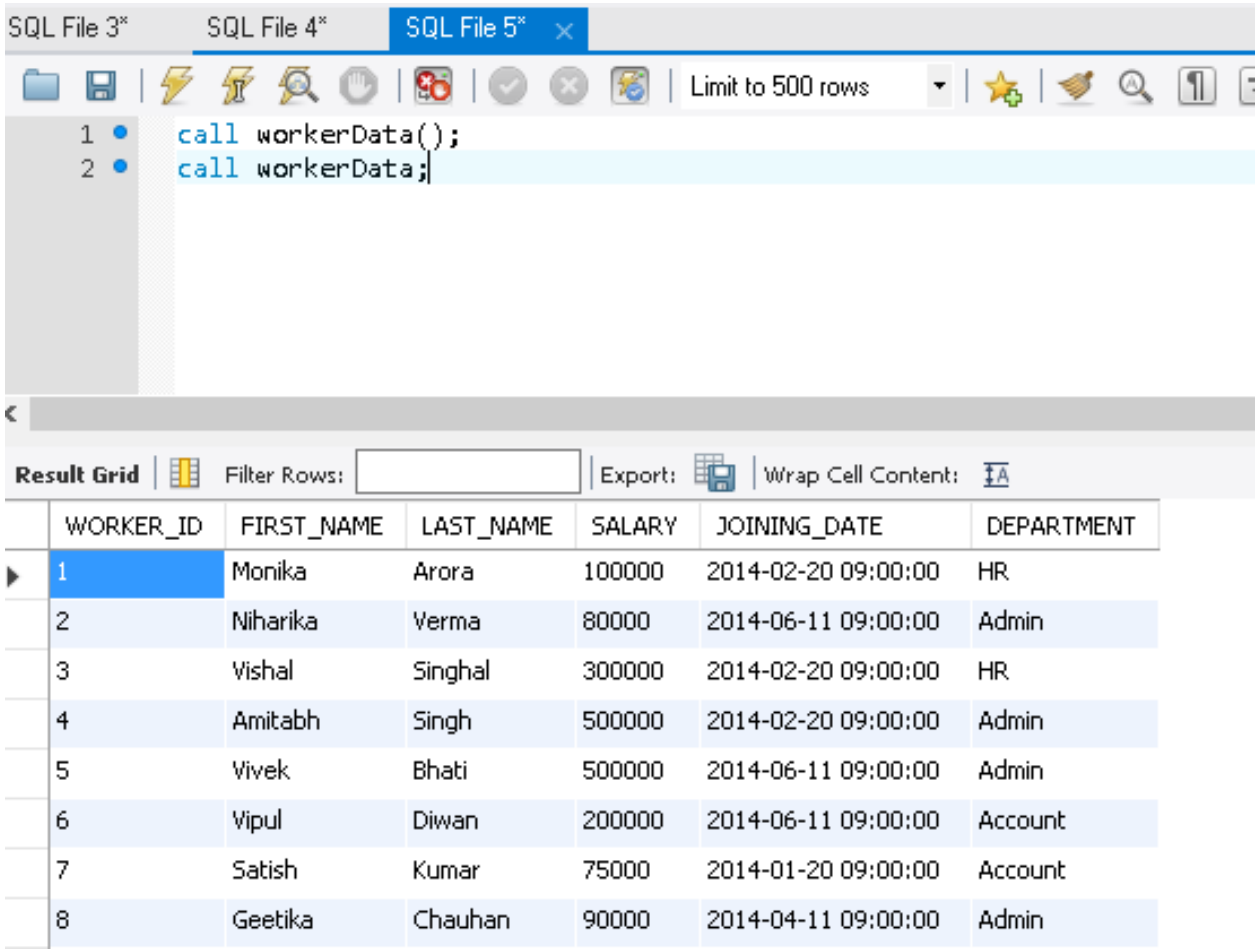


Stored Procedures in SQL (Cont.)

- The delimiter is the character or string of characters which is used to complete an SQL statement. By default we use semicolon (;) as a delimiter. But this causes problem in stored procedure because a procedure can have many statements, and everyone must end with a semicolon. So for your delimiter, pick a string which is rarely occur within statement or within procedure. Here we have used double dollar sign i.e. $$.You can use whatever you want
- We can call the procedure using **CALL statement**. The CALL statement is used to invoke a procedure that is stored in a DATABASE. Here is the syntax :

  CALL sp_name([parameter[,...]])
- Stored procedures which do not accept arguments can be invoked without parentheses. Therefore CALL **workerData**() and CALL **workerData** are equivalent

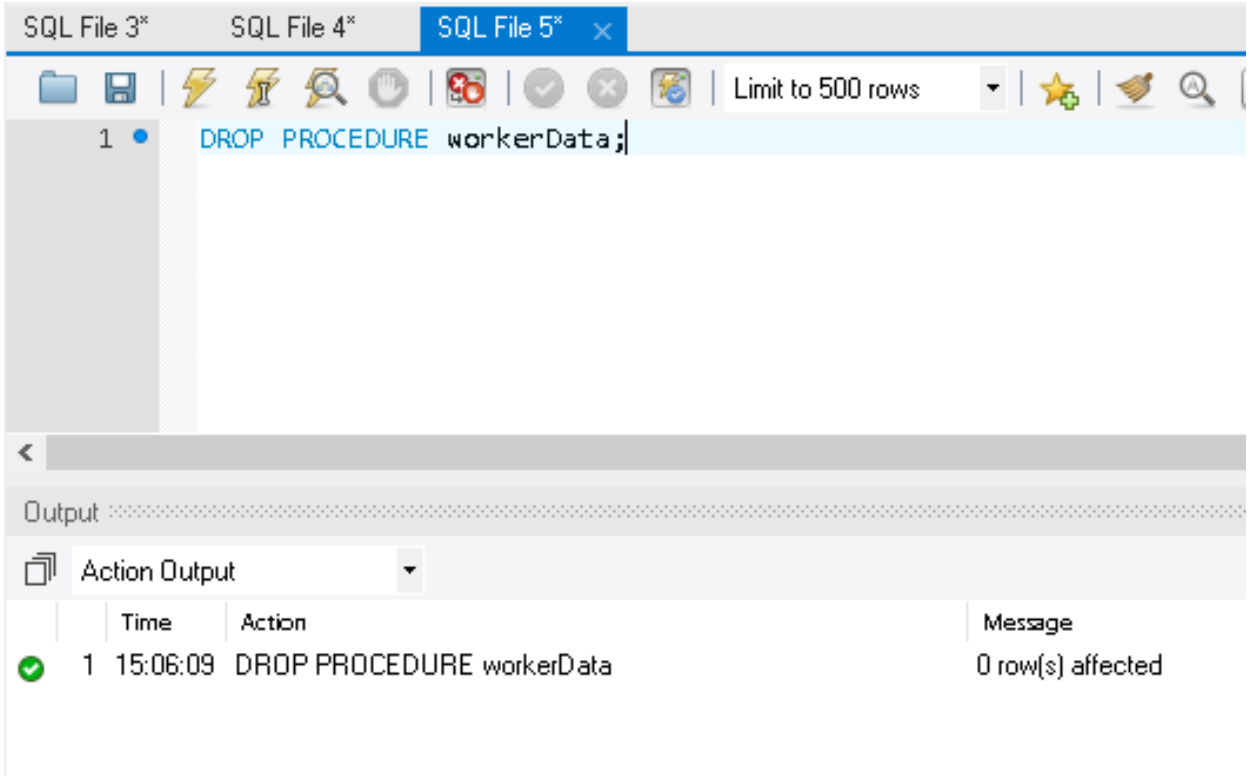
Stored Procedures in SQL (Cont.)



WORKER_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
1	Monika	Arora	100000	2014-02-20 09:00:00	HR
2	Niharika	Verma	80000	2014-06-11 09:00:00	Admin
3	Vishal	Singhal	300000	2014-02-20 09:00:00	HR
4	Amitabh	Singh	500000	2014-02-20 09:00:00	Admin
5	Vivek	Bhati	500000	2014-06-11 09:00:00	Admin
6	Vipul	Diwan	200000	2014-06-11 09:00:00	Account
7	Satish	Kumar	75000	2014-01-20 09:00:00	Account
8	Geetika	Chauhan	90000	2014-04-11 09:00:00	Admin

Stored Procedures in SQL (Cont.)

- **DROP PROCEDURE :**
  - This statement is used to drop a stored procedure
  - DROP {PROCEDURE} [IF EXISTS] sp_name
  - The IF EXISTS clause is a MySQL extension. It prevents an error from occurring if the procedure does not exist.



Triggers

- A trigger is a set of actions that are run automatically when a specified change operation (SQL INSERT, UPDATE, or DELETE statement) is performed on a specified table
- Triggers are useful for tasks such as enforcing business rules, validating input data, and keeping an audit trail
- **Benefits of using triggers in business:**
  - **Faster application development**. Because the database stores triggers, you do not have to code the trigger actions into each database application
  - **Easier maintenance**. If a business policy changes, you need to change only the corresponding trigger program instead of each application program
  - **Improve performance in client/server environment**. All rules run on the server before the result returns
- We can create the trigger with the following syntax: (refer to the next slide)

Triggers (Cont.)

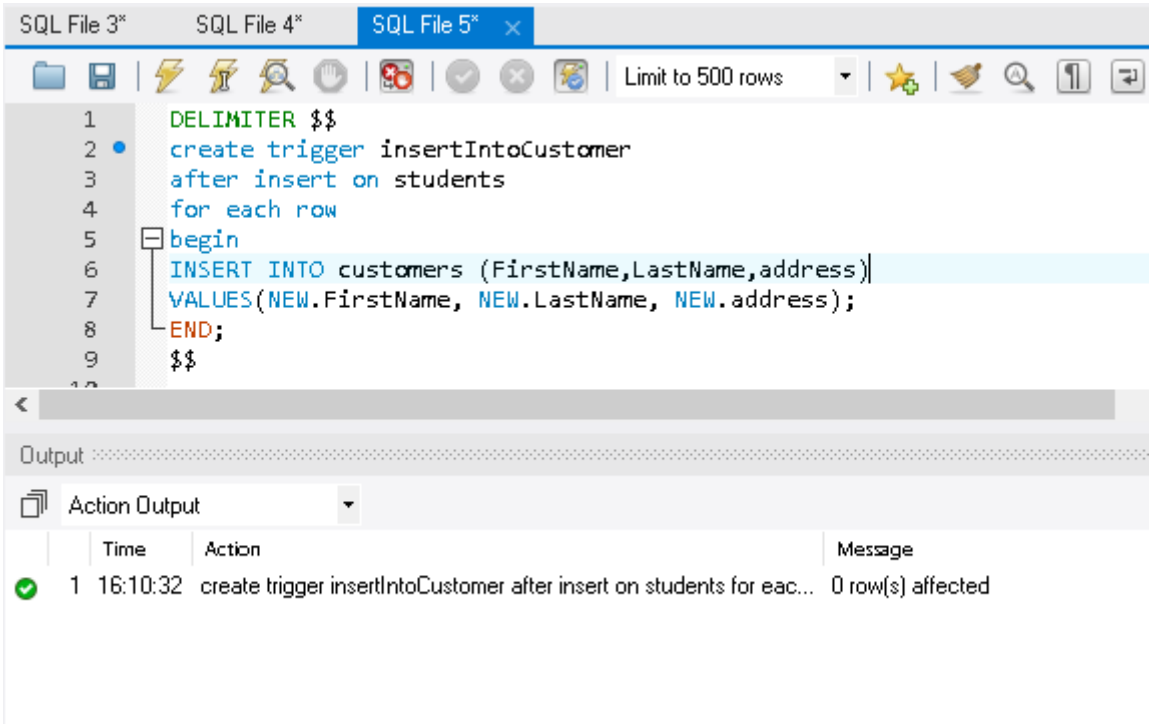

```
CREATE [OR REPLACE ] TRIGGER trigger_name
{BEFORE | AFTER | INSTEAD OF }
{INSERT [OR] | UPDATE [OR] | DELETE}
[OF col_name]   ON table_name
[REFERENCING OLD AS o NEW AS n]
[FOR EACH ROW]
WHEN (condition)
DECLARE
  Declaration-statements
BEGIN
  Executable-statements
EXCEPTION
  Exception-handling-statements
END;
```

Triggers (Cont.)

- **CREATE [OR REPLACE] TRIGGER trigger_name**: It creates or replaces an existing trigger with the trigger_name
- **{BEFORE | AFTER | INSTEAD OF}** : This specifies when the trigger would be executed. The INSTEAD OF clause is used for creating trigger on a view
- **{INSERT [OR] | UPDATE [OR] | DELETE}**: This specifies the DML operation
- **[OF col_name]**: This specifies the column name that would be updated
- **[ON table_name]**: This specifies the name of the table associated with the trigger
- **[REFERENCING OLD AS o NEW AS n]**: This allows you to refer new and old values for various DML statements, like INSERT, UPDATE, and DELETE
- **[FOR EACH ROW]**: This specifies a row level trigger, i.e., the trigger would be executed for each row being affected. Otherwise the trigger will execute just once when the SQL statement is executed, which is called a table level trigger
- **WHEN (condition)**: This provides a condition for rows for which the trigger would fire. This clause is valid only for row level triggers

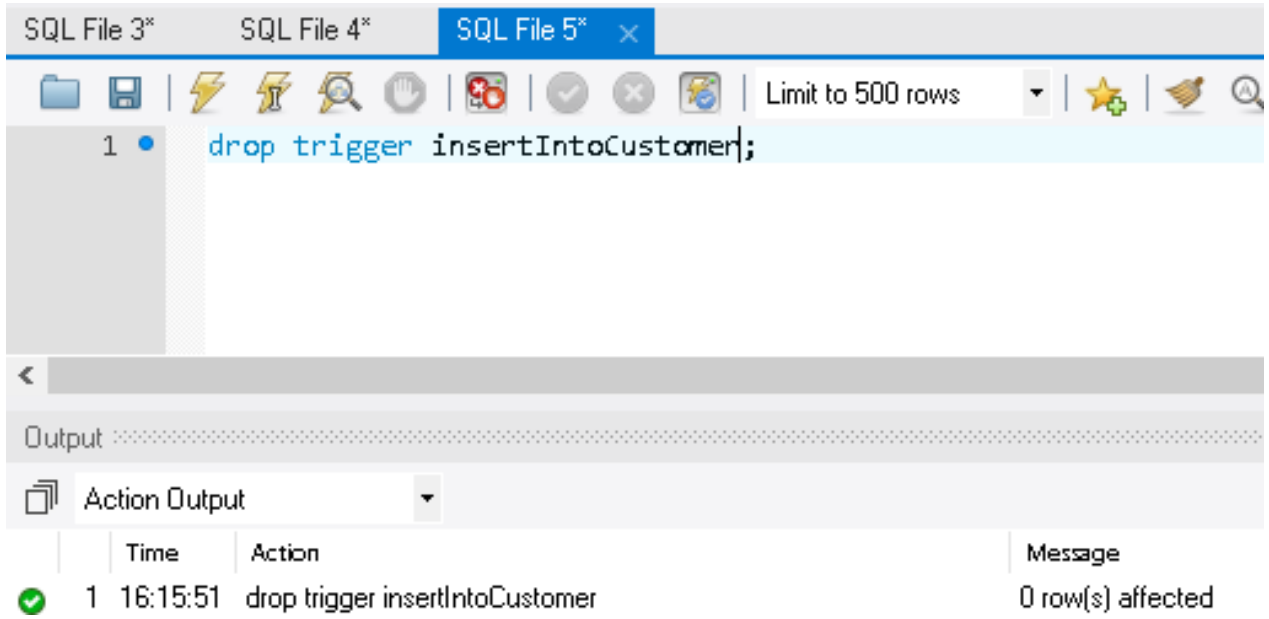
Triggers (Cont.)

- We will create a trigger called **insertIntoCustomer**, when there is any insertion in students table then, we will insert it in customers table as well



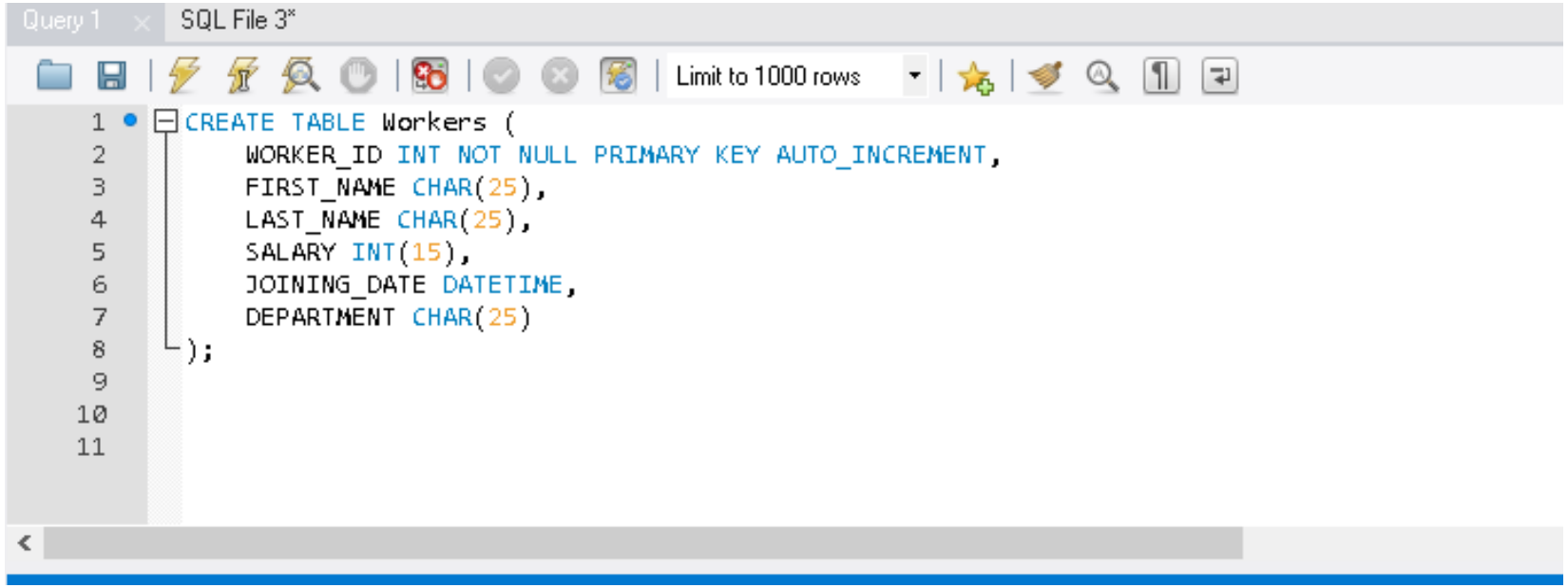
Triggers (Cont.)

- **DROP TRIGGER :**
  - This statement is used to drop a trigger
  - DROP {TRIGGER} [IF EXISTS] trigger_nam
  - The IF EXISTS clause is a MySQL extension. It prevents an error from occurring if the trigger does not exist



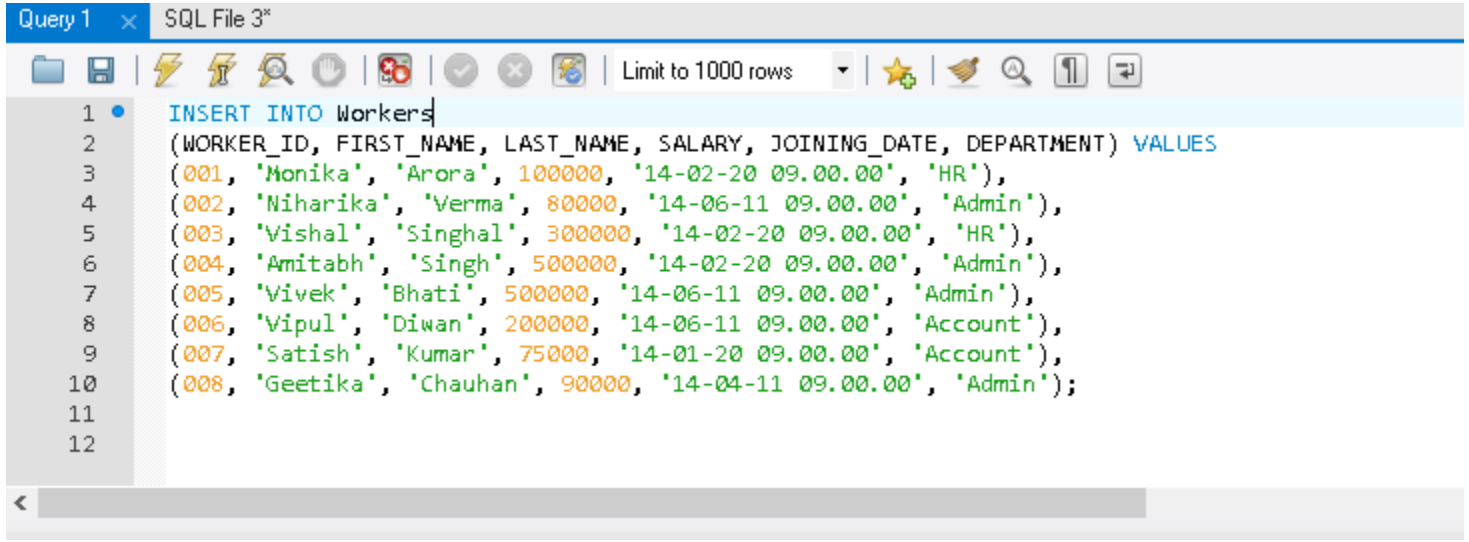
SQL – Assignment

- Suppose we have a table named **Workers** which contains workers information with the below structure and below data in it



SQL – Assignment (Cont.)

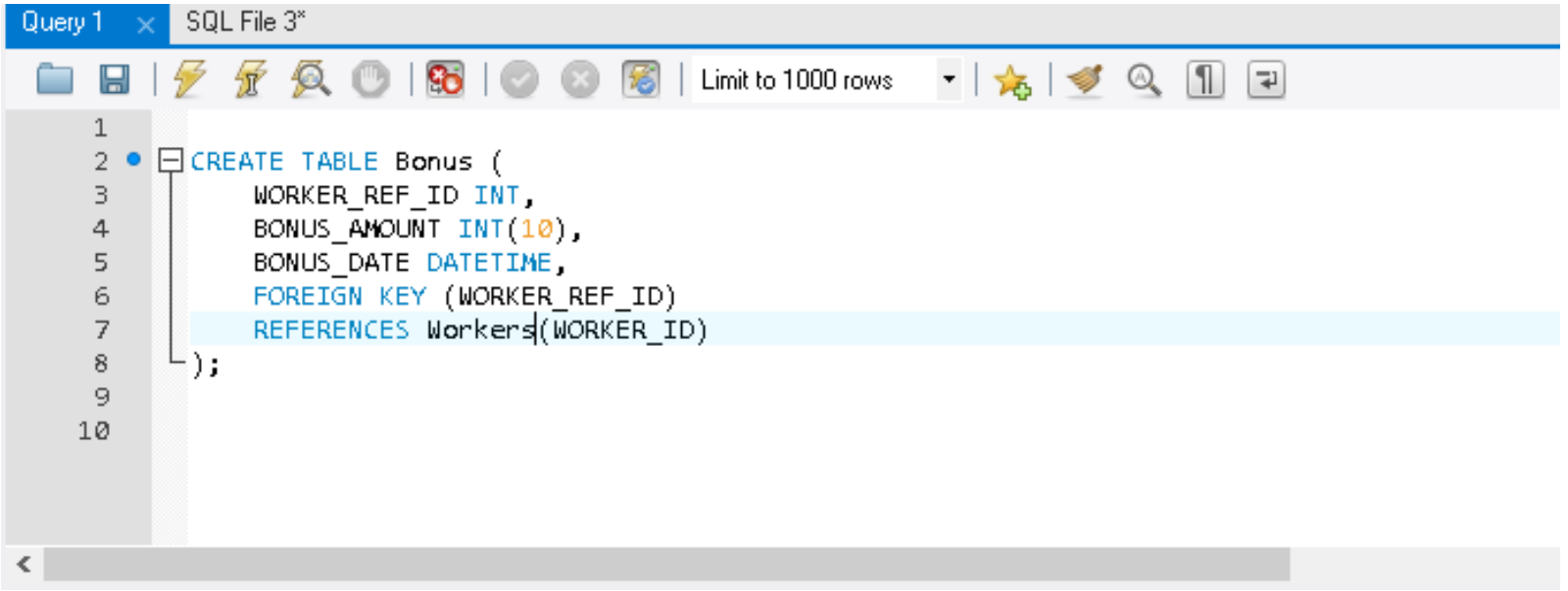
- Below is the insert queries:



- We also have another table named as **Bonus** which stores the bonus information of **Workers** with below structure and data

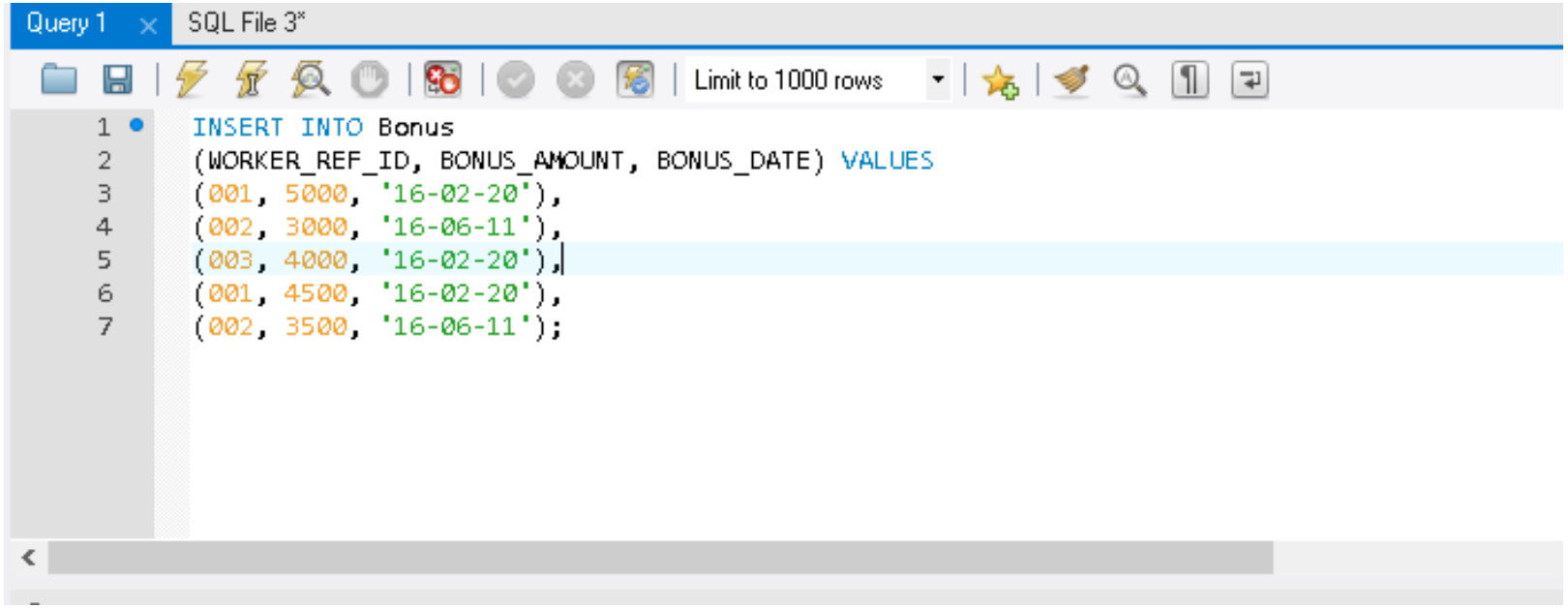
SQL – Assignment (Cont.)

- Below is the structure of **Bonus** table:



SQL – Assignment (Cont.)

- Below is the insert queries in the Bonus table:



SQL – Assignment (Cont.)

• Write an SQL query to fetch FIRST_NAME from workers table in upper case

```
Select upper(FIRST_NAME) from workers;
```

• Write an SQL query to print the FIRST_NAME And LAST_NAME from workers table into a single column COMPLETE_NAME. A space char should separate them where DEPARTMENT is Account

```
Select CONCAT(FIRST_NAME, ' ', LAST_NAME) AS 'COMPLETE_NAME' from workers where DEPARTMENT = “Account”;
```

SQL – Assignment (Cont.)

- Write the difference between a primary key and a foreign key?

Item	Primary Key	Foreign Key
Consist of One or More Columns	Yes	Yes
Duplicate Values Allowed	No	Yes
Null Values Allowed	No	Yes
Uniquely Identify Rows In a Table	Yes	Maybe
Number allowed per table	One	One or More
Indexed	Automatically Indexed	No Index Automatically created

SQL – Assignment (Cont.)

•Write an SQL query to print details of the workers whose FIRST_NAME ends with ‘H’ and contains six alphabets

```
Select * from workers where FIRST_NAME like '_____h';
```

• Write an SQL query to print the FIRST_NAME And LAST_NAME from workers table into a single column NAME which have BONUS_AMOUNT between 3000 and 4000 and in ascending order according to BONUS_AMOUNT

```
Select CONCAT(FIRST_NAME, ' ', LAST_NAME) AS 'NAME' from workers where WORKER_ID IN
(select WORKER_REF_ID from bonus where BONUS_AMOUNT Between 3000 and 4000 order
by BONUS_AMOUNT);
```

SQL – Assignment (Cont.)

•Write an SQL query to update the BONUS_AMOUNT as 7000 where DEPARTMENT is Admin and LAST_NAME is ‘Verma’

```
Update bonus set BONUS_AMOUNT = '7000' where WORKER_REF_ID IN
(select WORKER_ID from workers where DEPARTMENT = 'Admin' and LAST_NAME = 'Verma');
```

• Write an SQL query to print details of the workers who has joined in Feb’2014

```
Select * from workers where year(JOINING_DATE) = 2014 and month(JOINING_DATE) = 2;
```

SQL – Assignment (Cont.)

- Write the difference between a primary key and a unique key?

	Primary Key	Unique Key
1	Primary Key Can't Accept Null Values.	Unique Key Can Accept Only One Null Value
2	Creates Clustered Index	Creates Non-Clustered Index
3	Only One Primary key in a Table	More than One Unique Key in a Table.
4	Primary Key Can be Made Foreign Key Into Another Table. Ex: `CREATE TABLE [country] ( [id] VARCHAR (50) NOT NULL, [country] VARCHAR (50) NOT NULL, CONSTRAINT [PK_country] PRIMARY KEY CLUSTERED ([id]));`	SQL Server, Unique Key Can be Made Foreign Key Into Another Table. Ex: `CREATE TABLE [country] ( [name] VARCHAR (50) NOT NULL, [country] VARCHAR (50) NOT NULL, UNIQUE NONCLUSTERED ([name]));`

SQL – Assignment (Cont.)

- Write an SQL query to fetch unique values of DEPARTMENT from workers table

  Select distinct DEPARTMENT from workers;

•Write an SQL query to print the DEPARTMENT, total number of workers in a DEPARTMENT and total salary with respect to a DEPARTMENT from workers table order by total salary descending

```
Select DEPARTMENT, count(WORKER_ID), sum(SALARY) as TOTAL_SALARY FROM workers group
by DEPARTMENT order by TOTAL_SALARY DESC;
```

SQL – Assignment (Cont.)

- Write an SQL query to print details of the workers whose FIRST_NAME ends with ‘A’.

```
Select * from workers where FIRST_NAME like '%a';
```

- Write an SQL query to print FIRST_NAME and BONUS_AMOUNT of workers order by FIRST_NAME Ascending and BONUS_AMOUNT descending

```
Select wk.FIRST_NAME, bs.BONUS_AMOUNT from workers wk INNER JOIN bonus bs ON
(wk.WORKER_ID = bs.WORKER_REF_ID) order by wk.FIRST_NAME ASC,bs.BONUS_AMOUNT DESC;
```

SQL – Assignment (Cont.)

proschool An ims Initiative

• Write an SQL query to print the FIRST_NAME and LAST_NAME of workers who received BONUS_AMOUNT more than once

```
Select CONCAT(wk.FIRST_NAME, ' ', wk.LAST_NAME) AS 'NAME' from workers wk INNER JOIN bonus bs ON (wk.WORKER_ID = bs.WORKER_REF_ID) group by wk.WORKER_ID HAVING count(bs.BONUS_AMOUNT) > 1;
```

• Write an SQL query to fetch FIRST_NAME and monthly salary of workers from workers table where DEPARTMENT is HR

```
Select FIRST_NAME, (SALARY/12) as MONTHLY_SALARY from workers where DEPARTMENT = 'HR';
```

SQL – Assignment (Cont.)

•Write an SQL query to change datatype of LAST_NAME column in workers table from char to varchar of length 25

```
ALTER TABLE workers MODIFY COLUMN LAST_NAME varchar(25);
```

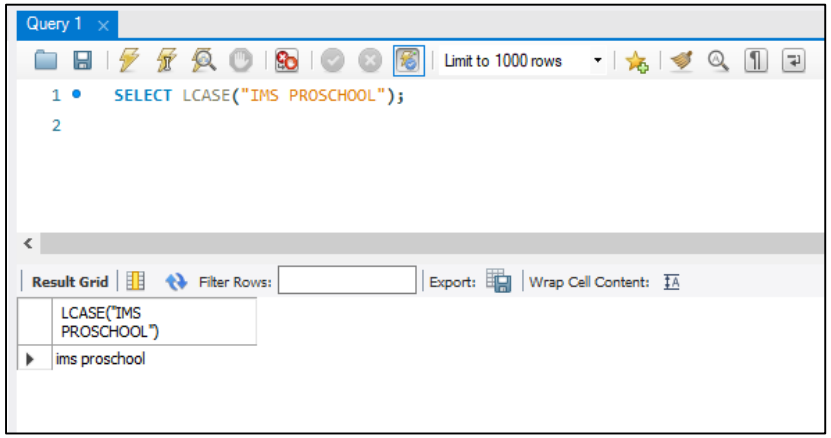
• Write an SQL query to print the names of workers who do not received BONUS_AMOUNT using joins

```
Select CONCAT(wk.FIRST_NAME, ' ', wk.LAST_NAME) AS 'NAME' from workers wk LEFT JOIN bonus bs ON (wk.WORKER_ID = bs.WORKER_REF_ID) WHERE bs.BONUS_AMOUNT is NULL;
```

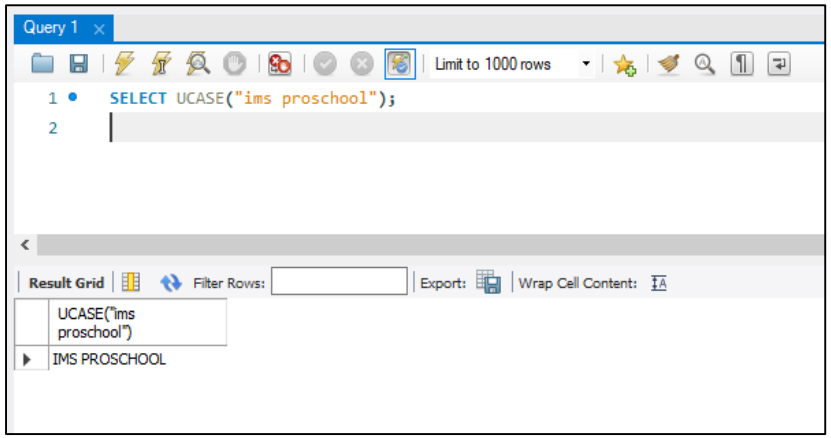
SQL Functions – String Functions

SQL String Functions are used to manipulate string data types as per required
Few string operations are:

- Lcase - Converts a string to lower-case
- Syntax - SELECT LCASE(“IMS PROSCHOOL")

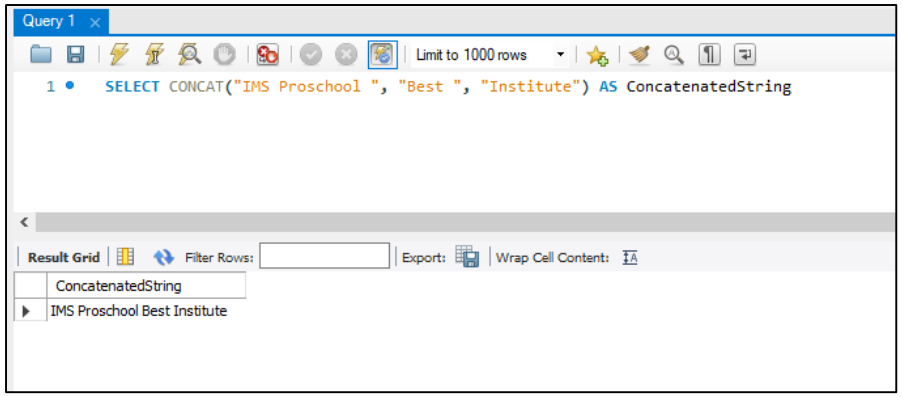


- Ucase - Converts a string to lower-case
- Syntax - SELECT LCASE(“ims proschool")

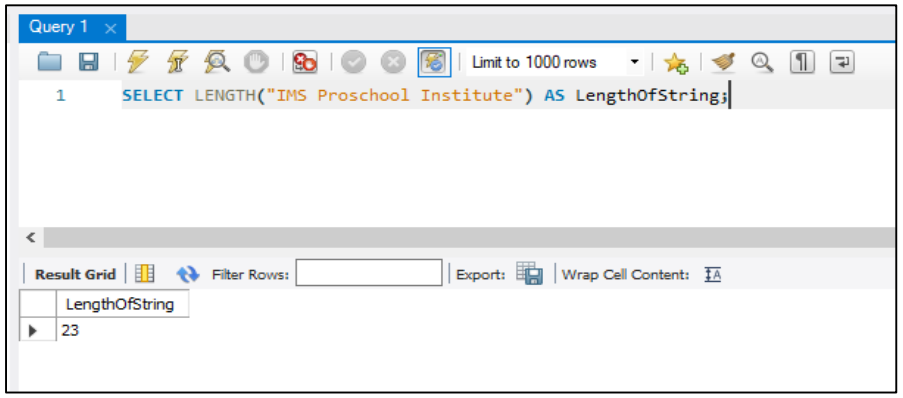


SQL Functions – String Functions

- Concat - Function used to add two or more expressions together
- Syntax - SELECT CONCAT("IMS Proschool ", "Best ", "Institute") AS ConcatenatedString

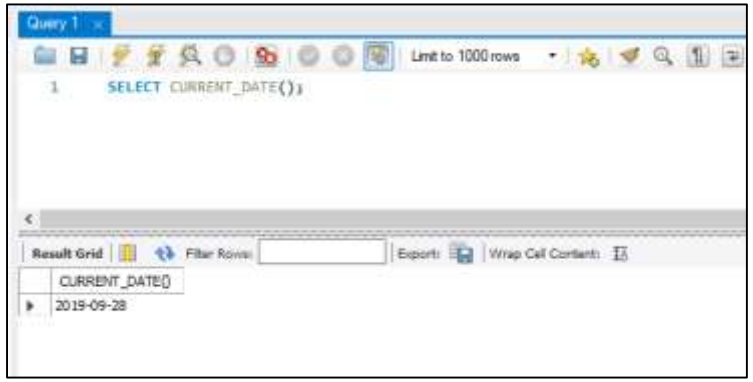


- Length - Function returns the length of a string (in bytes)
- Syntax - SELECT LENGTH("IMS Proschool Institute") AS LengthOfString;

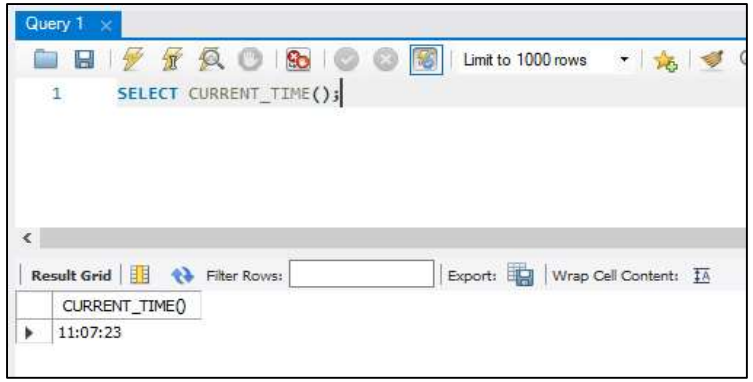


SQL Functions – Date Functions

- Current_Date – Returns current date in "YYYY-MM-DD" (string) or as YYYYMMDD (numeric)
- Syntax - SELECT CURRENT_DATE()
- Also equals function - CURDATE

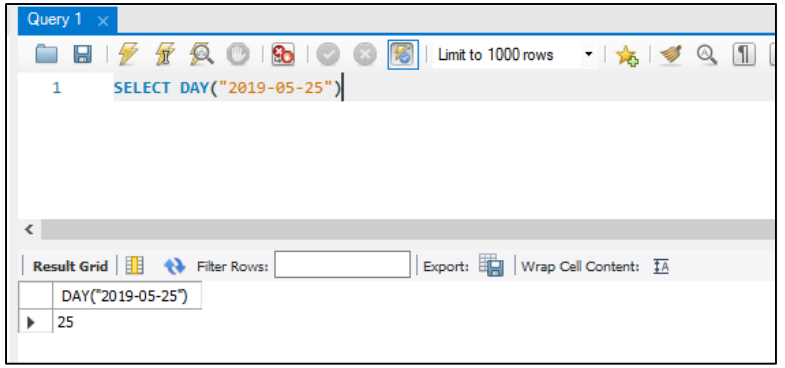


- Current_Time – Returns current time as "HH-MM-SS" (string) or as HHMMSS.uuuuuu (numeric)
- Syntax - SELECT CURRENT_TIME()
- Also equals function - CURTIME

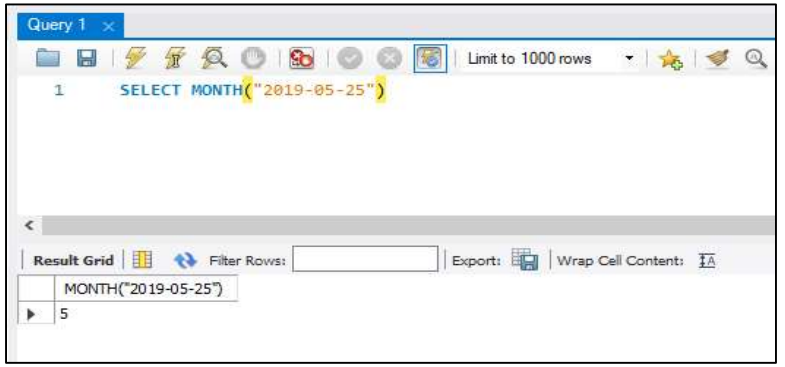


SQL Functions – Date Functions

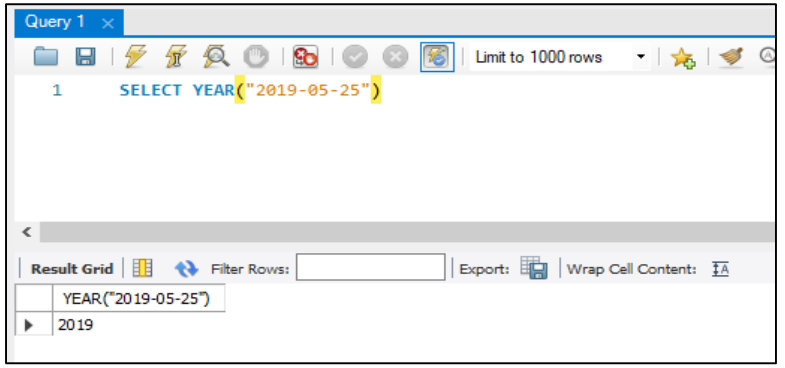
- Day – Returns day of the month from a date
- Syntax - SELECT DAY("2019-05-25")



- Month – Returns the month part from a date
- Syntax - SELECT MONTH("2019-05-25")



- Year – Returns the year part from a date
- Syntax - SELECT YEAR("2019-05-25")



Thank You.