|  |
| --- |
| **Overview of the Lab** |
| The purpose of this lab is to gently get you started using the Structured Query Language (SQL), by teaching you the most fundamental commands and concepts step-by-step. SQL is the de-facto query language for modern relational database management systems (RDBMS). All major, modern RDBMS support SQL. Database developers, administrators, and even software applications access RDBMS through the use of SQL. Familiarity with SQL is essential for working with and understanding modern RDBMS.  In this lab, you will learn how to:   * create and drop a table. * insert, update, delete, and select a row in a table. * use strings, dates, and numbers. * use a WHERE clause to limit the number of rows affected by the SELECT, UPDATE, and DELETE commands. * SELECT only a subset of columns in a result set. * add a NOT NULL constraint to a table column. * add a PRIMARY KEY constraint to a table column. * Insert a NULL value into a table row. * demonstrate anomalies resulting from data redundancy. * explore alternative file representation for data and its limitations. |

|  |
| --- |
| **Lab 1 Explanations** |
| It is important to read through the Lab 1 Explanation document to successfully complete this lab. It is available in the assignment inbox alongside this lab. The explanation document illustrates how to correctly execute each SQL construct step-by-step, and explains important theoretical and practical details. |

|  |
| --- |
| **Required Software** |
| The examples in this lab will execute in modern versions of Oracle, Microsoft SQL Server, and PostgreSQL as is. If you have been approved to use a different RDBMS, you may need to modify the SQL for successful execution, though the SQL should execute as is if your RDBMS is ANSI compliant.  The screenshots in this lab display execution of SQL in the default SQL clients supported in the course – Oracle SQL Developer, SQL Server Management Studio, and pgAdmin. You are welcome, however, to use a SQL client other than these defaults if you prefer. |

|  |
| --- |
| **Preparing for the Lab** |
| You will need to install a RDBMS prior to completing this lab. If you are using Oracle, it is highly recommended that you create and login as a non-system user, to avoid damaging the database. You can create a user with the following commands:  CREATE USER *username* IDENTIFIED BY *password* DEFAULT TABLESPACE users TEMPORARY TABLESPACE temp;  GRANT connect, resource TO *username*;  You will then be able to login as the new user.    If you are using Microsoft SQL Server, it is highly recommended that you create and use a database other than the Master database. You can do so with the following commands:  CREATE DATABASE *database\_name*;  GO;  USE *database\_name*;  If you are using PostgreSQL you can use the UI wizard to create a database or use the following script from the default databse created at time of installation:  CREATE DATABASE database\_name; |

|  |
| --- |
| **Saving Your Data** |
| If you choose to perform portions of the lab in different sittings, it is important to *commit* your data at the end of each session. This way, you will be sure to make permanent any data changes you have made in your current session, so that you can resume working without issue in your next session. To do so, simply issue this command:  COMMIT;  We will learn more about committing data in future weeks. For now, it is sufficient to know that data changes in one session will only be visible only in that session, unless they are committed, at which time the changes are made permanent in the database. |

**Section One – Absolute Fundamentals**

**Section Background**

In this section, you learn the absolute fundamentals of SQL – creating and dropping a table, getting data into the table, listing the data in the table, and deleting and updating the data. You will be working with a Car dealership table that has basic information about the cars they have for sale. When you have completed some steps in the section, the Cars table will look as illustrated below.

*Cars Table*

|  |  |  |  |
| --- | --- | --- | --- |
| CarBrand: VARCHAR(64) | CarModel: VARCHAR(50) | AcquiredDate: DATE | Price: DECIMAL(10,2) |
| Ford | Econoline Full-Size Van | 15-AUG-2021 | 29,995.00 |

You will create this table and try out SQL commands using the table.

Do not worry if you do not recognize the structure and datatypes in the table above. The Lab 1 Explanation document and supporting lecture and textbook readings give you the information you need. Start reading the explanation document first, then iteratively complete the steps below. Each step below has an accompanying explanation in the explanation document.

For each step that requires SQL, *make sure to capture a screenshot of the command and the results of its execution.* Submissions that do not contain screenshots will be returned to you. A screenshot is more legible if you use one of the many free tools to capture only the relevant portion of the screen, rather than capturing the entire application window. A few steps ask for explanations rather than SQL; no screenshot is needed for such steps.

**Section Steps**

1. *Creating a Table –* Create the Cars table. As a reminder, make sure to follow along in the Lab 1 Explanations document as it shows you how to create tables and complete the other steps.
2. *Inserting a Row –* Insert the first row where the Car Brand name is “Ford”, the Car Model is “Econoline Full-Size Van”, the acquisition date for the car is August 15,2021, and the price is $29,995.00.
3. *Selecting All Rows –* Select all rows in the table to view the row you inserted.
4. *Updating All Rows –* Update the price of the row in the table to $28,000, then select all rows in the table to view the row you updated.
5. *Deleting All Rows –* Remove all rows from the table, then select all rows in the table to verify there are no rows.
6. *Dropping a Table –* Drop the Cars table, then select all rows in the table to verify the table doesn’t exist. Explain how you would use the error message, in conjunction with the SELECT command, to diagnose the error.

**Section Two – More Precise Data Handling**

**Section Background**

In this section, you enhance your skills by more precisely working with data. In the prior section, you learned to work with all rows in the table. In this section, you add to that by learning to pinpoint specific rows to be retrieved, modified, or deleted. You also learn how to add SQL constraints to your table, and to work with nulls.

You will work with an Apartments table, which will ultimately look like the below when all steps have been completed.

*Apartments Table*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ApartmentNum: DECIMAL  Primary Key | ApartmentName: VARCHAR(64)  NOT NULL | Description: VARCHAR(64)  NULL | CleanedDate: DATE  NOT NULL | AvailableDate: DATE  NOT NULL |
| 498 | Deer Creek Crossing | Great view of Riverwalk | 19-APR-2022 | 25-APR-2022 |
| 128 | Town Place Apartments | Convenient walk to Parking | 20-MAY-2022 | 25-MAY-2022 |
| 316 | Paradise Palms |  | 02-JUN-2021 | 08-JUN-2021 |

**Section Steps**

1. *Table Setup –* Create the Apartments table with its columns, datatypes, and constraints.
2. *Table Population –* Insert the rows illustrated in the figure above. Note that the description for Apartment 316 at Paradise Palms is null. Then select all rows from the Apartments table to show that the inserts were successful.
3. *Invalid Insertion –* The following values leave the Apartment Name with no value.

**ApartmentNum** = 252

**ApartmentName** = NULL

**Description** = Close to Downtown shops

**CleanedDate** = 17-JUL-2020  
**AvailableDate** = 13-JUL-2020

a. In your own words, explain what a null value is.

b. In your own words, explain what a NOT NULL constraint is.

c. Attempt to insert the values as listed and explain how the database handles this attempt.

Explain how you would interpret the error message conclude that the location column is missing a required value.

1. *Valid Insertion –* Now insert the row with the Apartment Name intact, with the following values.

**ApartmentNum** = 252

**ApartmentName** = The Glenn

**Description** = Close to Downtown shops

**CleanedDate** = 17-JUL-2020  
**AvailableDate** = 13-JUL-2020

1. *Filtered Results –* Retrieve only the Apartment Name and the Description for Deer Creek Crossing, using the primary key as the column that determines which row is retrieved.

Explain why it is useful to limit the number of rows and columns returned from a SELECT statement.

1. *Targeted Update –* The Paradise Palms apartment has no description. Update the row so that its description says “A mile walk to the beach”.

Select all rows in the table to show that the update was successful.

1. *Updating to Null –* Update the Town Place Apartments so that it no longer has a description (i.e., its description is null).

Select all rows in the table to show that the update was successful.

1. *Targeted Deletion –* Delete all rows where the Cleaned date is greater than April 1, 2022, by using the Cleaned Date column as the determinant of which rows are deleted.

Select all rows in the table to show the delete was successful.

**Section Three – Data Anomalies and Formats**

**Section Background**

When the same data is repeated multiple times, anomalies can result. In this section, you demonstrate and explore three such anomalies in a relational database – insert, update, and delete anomalies.

Databases provide several advantages over files. In this section, you explore putting the same data in a relational table and in a file, then compare the two.

**Section Steps**

1. *Data Anomalies –* In this step you demonstrate anomalies that can occur in improperly designed tables.
   1. Create a table of your choosing that has at least three columns.
   2. Using the table, demonstrate an anomaly that occurs when the same data is inserted multiple times with different values, and explain what the anomaly means for data integrity.
   3. Using the table, demonstrate a deletion anomaly with SQL, and explain what the anomaly means for data integrity.
2. *File and Database Table Comparison* – In this step you compare the table created in #15 with a file that contains all the same information.
   1. Create a file in any format you’d like that contains all the same columns and at least 4 rows of information as the table you created in #15. There are many formats you can use. Some examples include XML, flat file, binary, text, and JSON; this list is not exhaustive. All columns and at least 4 rows should be present in the file in its new format. Make sure to provide the file or a screenshot of the file and to explain your choices.
   2. With a few paragraphs, compare what it’s like to access data in the table versus in the file. You may need to first research how applications typically access data in this type of file. Make sure to at least use these comparison points:
      1. Efficiency – If there were millions of rows of data, would it be more efficient to access a single record in the relational table, or the file, and why?
      2. Security – Imagine you needed to restrict access to one specific row/record, allowing only one person to access it, while the rest of the rows could be accessed by many people. Would it be easier or more difficult to secure this row in the relational table compared to the file, and why?
      3. Structural Independence – Imagine the table structure was modified by adding or taking away columns, and equivalent changes were made to the file. Would these changes affect an app using the table differently than an app using the file, and why?

**Evaluation**

Your lab will be reviewed by your facilitator or instructor with the criteria outlined in the table below. Note that the grading process:

* involves the grader assigning an appropriate letter grade to each criterion.
* uses the following letter-to-number grade mapping – A+=100,A=96,A-=92,B+=88,B=85,B-=82,C+=88,C=85,C-=82,D=67,F=0.
* provides an overall grade for the submission based upon the grade and weight assigned to each criterion.
* allows the grader to apply additional deductions or adjustments as appropriate for the submission.
* applies equally to every student in the course.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Criterion** | **What it Measures** | **A+ Excellent** | **B Good** | **C Fair/Satisfactory** | **D Insufficient** | **F Failure** |
| **Section 1: Quality (30%)** | For section 1, this measures the correctness of the SQL results, the appropriateness of the SQL constructs used, how well the section is presented, and the quality of the supporting explanations. | Entirely correct results All constructs appropriate Excellent presentation Excellent supporting explanations | Mostly correct results Most constructs appropriate Good presentation Good supporting explanations | Somewhat correct results Some constructs appropriate Satisfactory presentation Satisfactory supporting explanations | Mostly incorrect results Inappropriate constructs Insufficient presentation Insufficient supporting explanations | Entirely incorrect results Inappropriate constructs Insufficient presentation Insufficient or missing supporting explanations |
| **Section 2: Quality (30%)** | For section 2, this measures the correctness of the SQL results, the appropriateness of the SQL constructs used, how well the section is presented, and the quality of the supporting explanations. | Entirely correct results All constructs appropriate Excellent presentation Excellent supporting explanations | Mostly correct results Most constructs appropriate Good presentation Good supporting explanations | Somewhat correct results Some constructs appropriate Satisfactory presentation Satisfactory supporting explanations | Mostly incorrect results Inappropriate constructs Insufficient presentation Insufficient supporting explanations | Entirely incorrect results Inappropriate constructs Insufficient presentation Insufficient or missing supporting explanations |
| **Section 3: Anomaly Demonstration and Explanation (20%)** | This measures how accurately and completely both anomaly types are demonstrated with SQL and justified with supporting explanations. | SQL demonstrations accurate and complete Explanations accurate and complete | SQL demonstrations mostly accurate and complete Explanations mostly accurate and complete | SQL demonstrations somewhat accurate and complete Explanations somewhat accurate and complete | SQL demonstrations mostly inaccurate and incomplete Explanations mostly inaccurate and incomplete | SQL demonstrations missing or entirely inaccurate and incomplete Explanations missing or entirely inaccurate and incomplete |
| **Section 3: File Soundness (20%)** | This measures how well the file represents the data from the relational table, how well the section is presented, and how well the explanations compare the structural independence, efficiency, and security between the table and file. | Completely accurate representation Excellent presentation Excellent explanations | Mostly accurate representation Good presentation Good explanations | Somewhat accurate representation Satisfactory presentation Satisfactory explanations | Mostly inaccurate representation Insufficient presentation Insufficient explanations | File missing or entirely inaccurate representation Insufficient presentation Insufficient or missing explanations |
|  |  | **Preliminary Grade:** |  | **Lateness Deduction** 5points per day4 days maximumContact your facilitator for any exceptions |  | **Lab Grade:** |

Use the **Ask the Teaching Team Discussion Forum** if you have any questions regarding how to approach this lab. Make sure to include your name in the filename and submit it in the *Assignments* section of the course.