|  |
| --- |
| **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; |

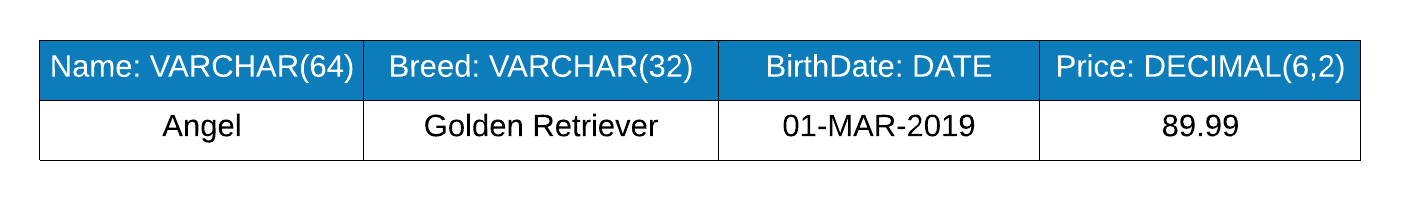
|  |
| --- |
| **Saving Your Data** |
| If you choose to perform portions of the lab in different sittings, it is important to *commit* your data at the end of each session. This way, you will be sure to make permanent any data changes you have made in your current session, so that you can resume working 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 PetStore table that has basic information about dogs as pets. When you have completed some steps in the section, the PetStore table will look as illustrated below.

*PetStore Table*

**

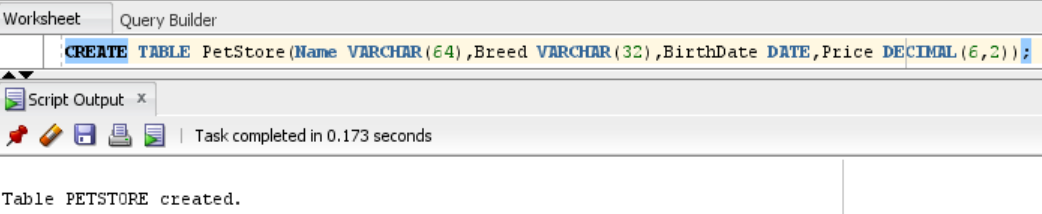
You will create this table and try out SQL commands using the table.

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 PetStore 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.

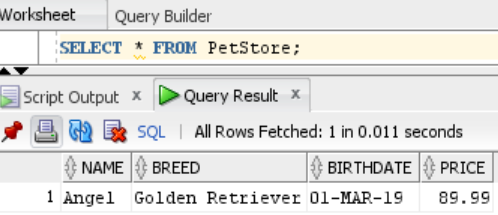


1. *Inserting a Row –* Insert the first row where the title is “Angel”, the breed is “Golden Retriever”, the birth date is 3/1/2019, and the price is $89.99.

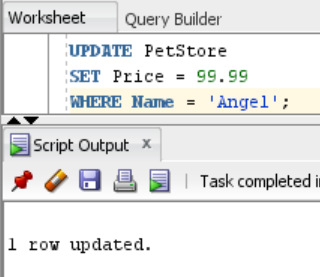
Graphical user interface, text, application, email

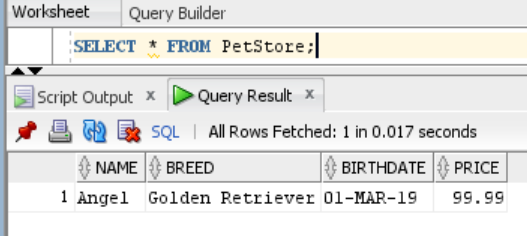
Description automatically generated

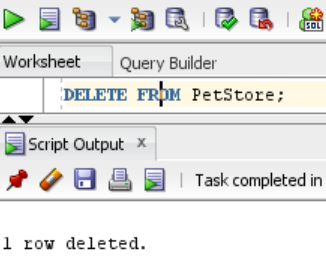
1. *Selecting All Rows –* Select all rows in the table to view the row you inserted.



1. *Updating All Rows –* Update the price of the row in the table to $99.99, then select all rows in the table to view the row you updated.



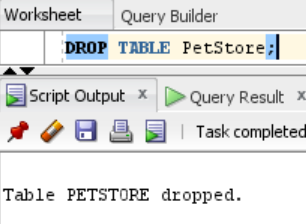


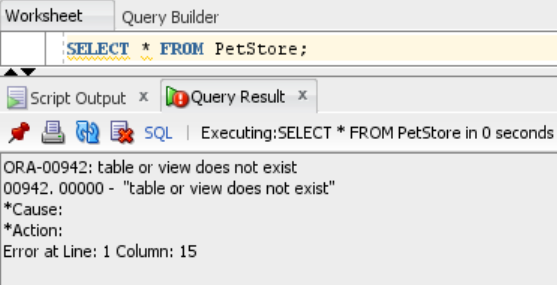
1. *Deleting All Rows –* Remove all rows from the table, then select all rows in the table to verify there are no rows.  
   

Graphical user interface, text, application

Description automatically generated

1. *Dropping a Table –* Drop the PetStore 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.





From the picture above, the error message “table or view does not exist” refers to the “PetStore” table that no longer exists because it was dropped/ deleted prior to executing the SELECT command. The source of error is said to be on Line 1 and Column 15. “PetStore” in the SELECT command starts at Line 1, Column 15, hence this is the source of the error. Since the “PetStore” table was dropped, executing a statement to select all rows from the deleted table will clearly throw the said error.

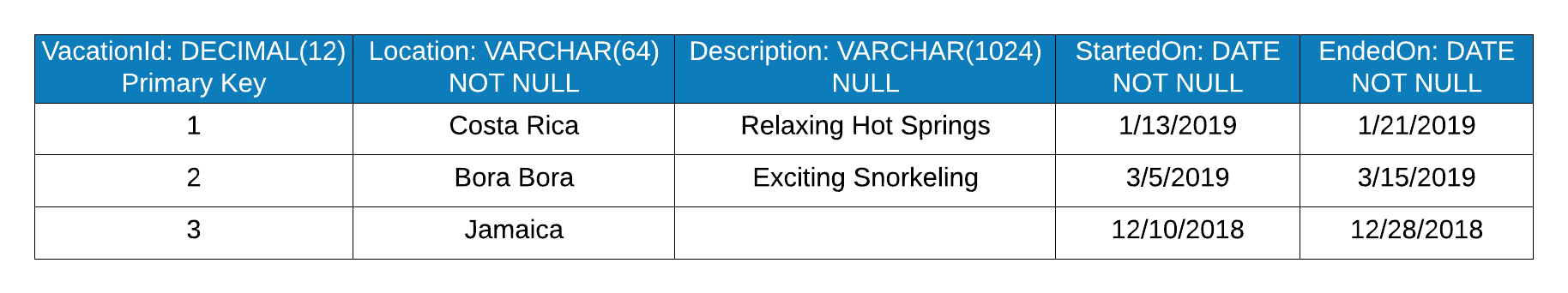
**Section Two – More Precise Data Handling**

**Section Background**

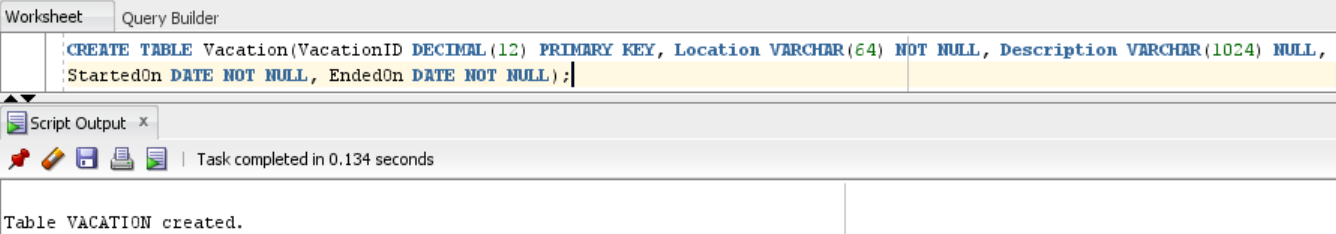
In this section, you enhance your skills by more precisely working with data. In the prior section, you learned to work with all rows in the table. In this section, you add to that by learning to pinpoint specific rows to be retrieved, modified, or deleted. You also learn how to add SQL constraints to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

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

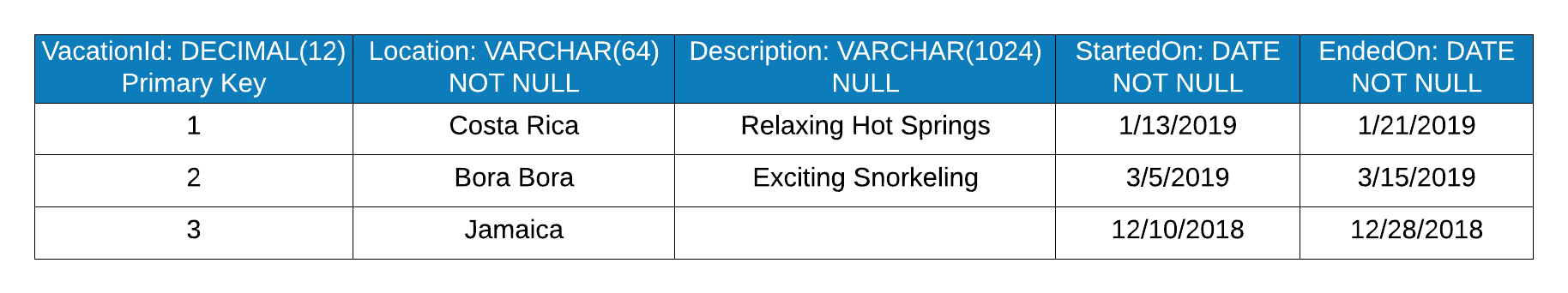
*Vacation Table*

**

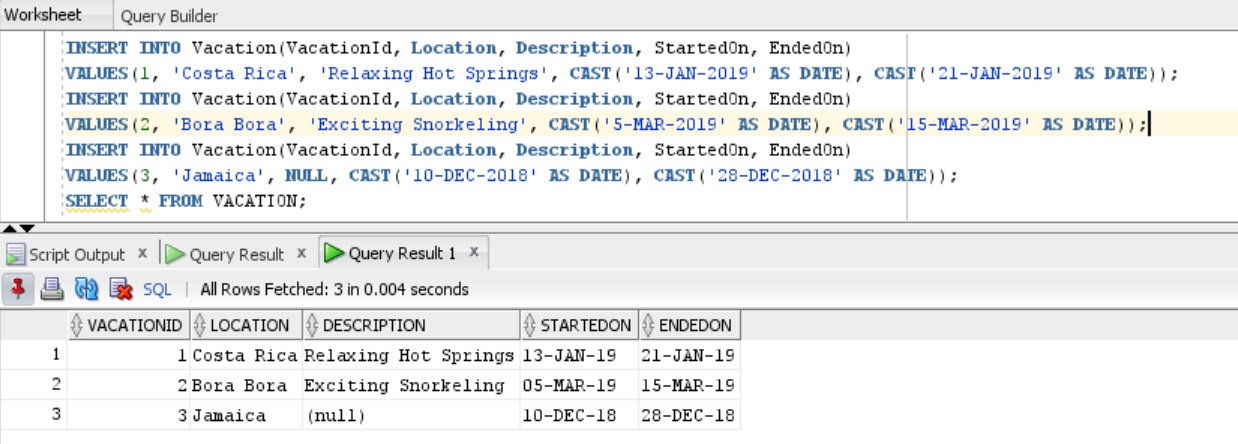
**Section Steps**

1. *Table Setup –* Create the Vacation table with its columns, datatypes, and constraints.   
   
2. *Table Population –* Insert the rows illustrated in the figure below. Note that the description for Jamaica is null.

*Vacation Table*

**

Select all rows from the Vacation table to show that the inserts were successful.



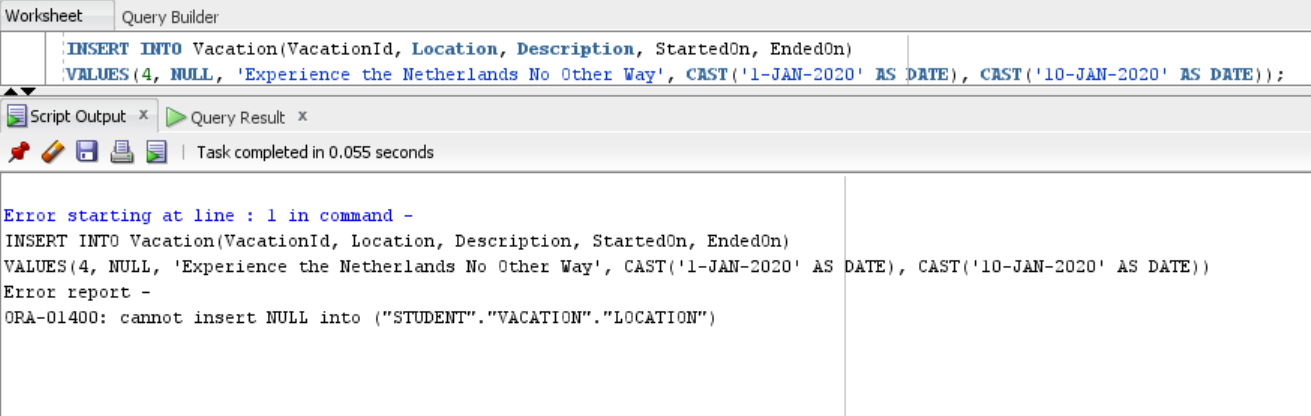
1. *Invalid Insertion –* Attempt to insert a row with the following values. The insert command will fail because the location column must have a value.  
     
   **VacationId** = 4

**Location** = NULL

**Description** = Experience the Netherlands No Other Way

**StartedOn** = 1/1/2020  
**EndedOn** = 1/10/2020

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

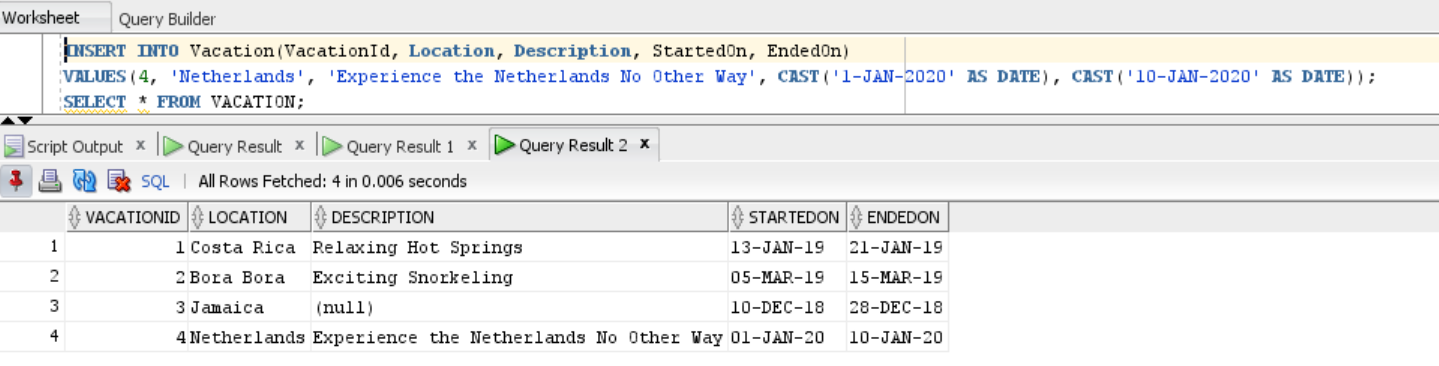


The screenshot above shows the error report “ORA-01400: cannot insert NULL into ("STUDENT"."VACATION"."LOCATION")”. VACATION.LOCATION is refers to the “Location” field within the “Vacation” table. It is understood that the “Location” field cannot be null, so the value inserted for this column should not be a null value. The value must be replaced with a NOT NULL value which is a character datatype for the “Location” field. So, inserting a NULL value or irrelevant datatype value will lead to errors.

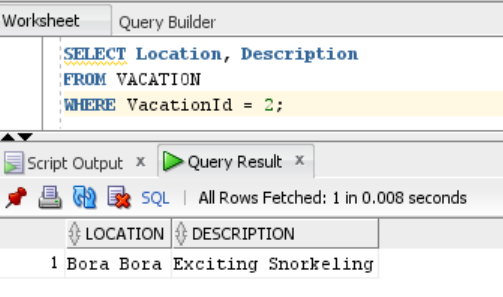
1. *Valid Insertion –* Now insert the row with a location intact, with the following values.  
     
   **VacationId** = 4

**Location** = Netherlands

**Description** = Experience the Netherlands No Other Way

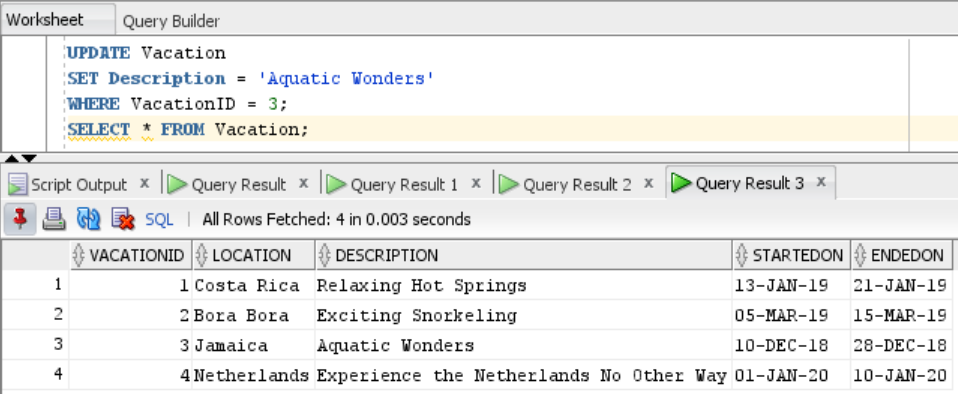
**StartedOn** = 1/1/2020  
**EndedOn** = 1/10/2020  


1. *Filtered Results –* Retrieve only the location and description for the Bora Bora vacation, 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.

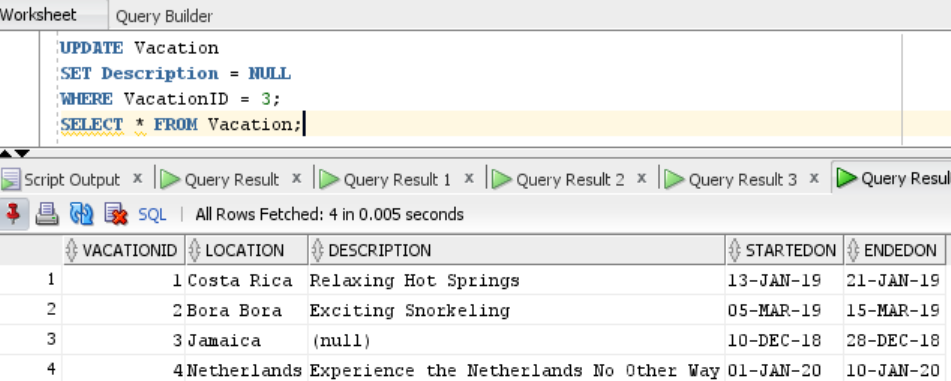


To increase efficiency and resourcefulness, it is useful to limit the number of rows and columns. It is not always ideal to retrieve all rows and columns in a table; especially when there are billions of rows and plenty of columns in a table. It can be very time-consuming to retrieve all the rows and columns. For example, when a customer of a bank only needs to know his/her balance, it may be ideal to retrieve columns like *CustomerName*, and *BalanceAmount* with *CustomerID* or *AccountNumber* as the primary key. Other fields like *TransactionID*, *DepositDate*, *DepositAmount*, *WithdrawalDate*, and *WithdrawalAmount* may not be needed.

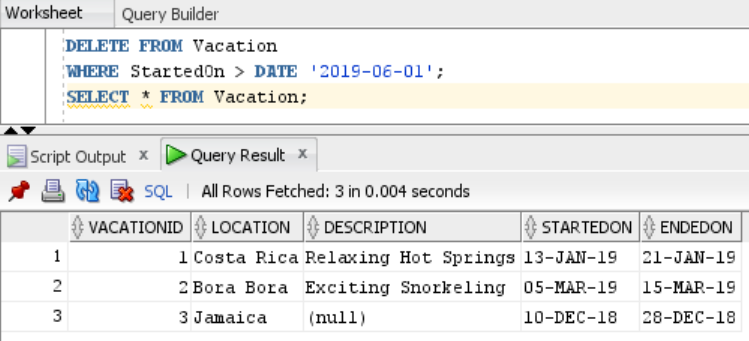
1. *Updating a Column –* The Jamaica vacation has no description. Update the row so that its description is “Aquatic Wonders”. Select all rows in the table to show that the update was successful.



1. *Updating to Null –* Update the Jamaica vacation so that it no longer has the 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 vacation started on a date greater than June 1st, 2019, by using the StartedOn column as the determinant of which rows are deleted. Select all rows in the table to show the delete was successful.



**Section Three – Concept Demonstration**

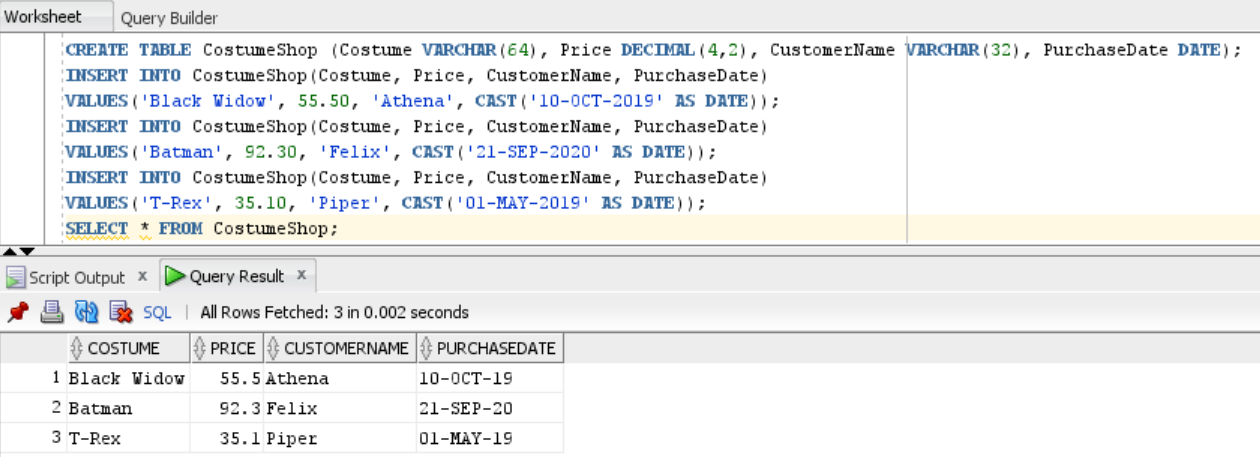
**Section Background**

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.

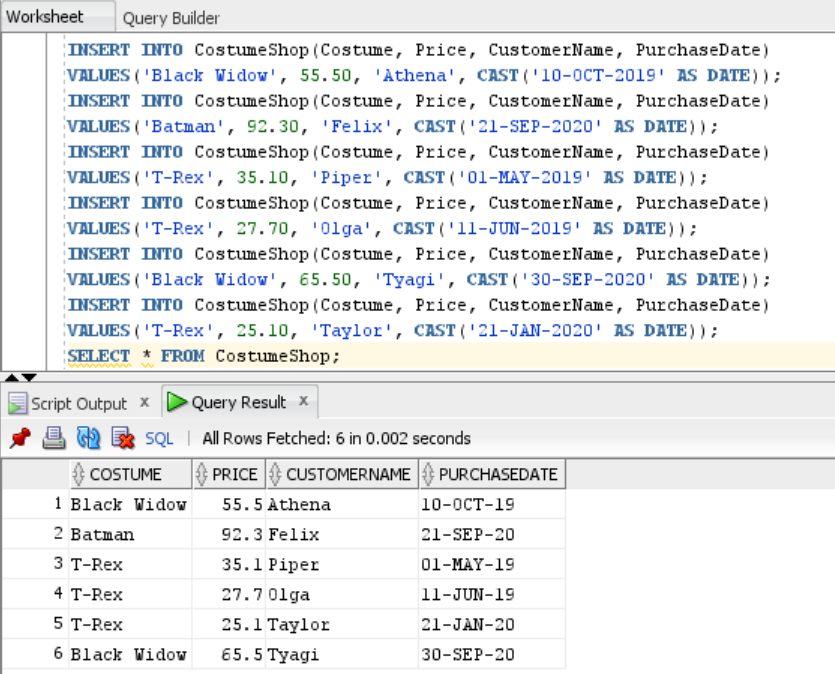
**Section Steps**

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.

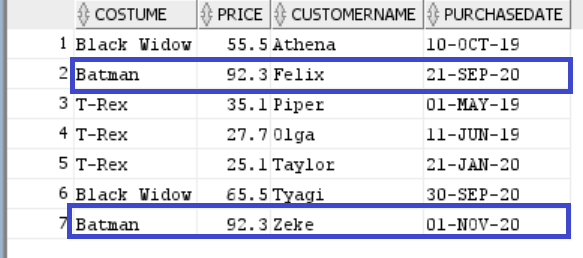


* 1. Using the table, demonstrate an anomaly that occurs when the same data is inserted multiple times with different values, and explain what the anomaly means for data integrity.

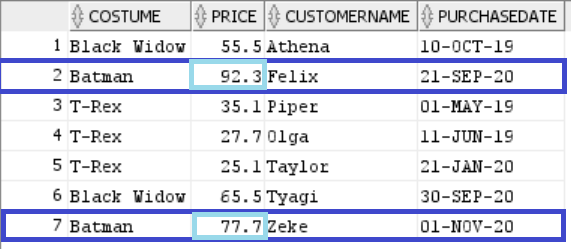
The following table has three different values for the T-Rex costume and two different values for the Black Widow costume. It is hard to know the actual price of these costumes as the price is different in each entry. It is hard to tell if the price of the T-Rex costume was lowered as time passed or these customers received a discount. Basically, Costume, Customer, and Purchase should have been three different tables, but these are clubbed as a single table. For this table, it is hard to determine the actual price of the T-Rex and Black Widow costume. Any given costume may be purchased again and there will be more duplicate entries.



Updating a table can also cause anomalies by disagreeing with some of the values in the table. Suppose a new row for the Batman costume was inserted by the shopkeeper with the same price for both customers. There is no anomaly even if the data is duplicated.

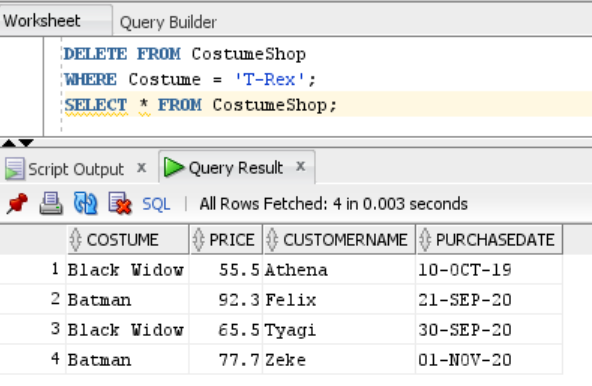


If the shopkeeper realizes that he sold the Batman costume at a discount for the customer named Zeke, he’ll update the table with the correct price. This leads to a data anomaly as there are two different prices for the Batman costume.



* 1. Using the table, demonstrate a deletion anomaly with SQL, and explain what the anomaly means for data integrity.

Deleting a record that may contain attributes that should not be deleted leads to a deletion anomaly. Suppose the customers Olga, Piper, and Taylor placed an online order for the T-Rex costume and decided to return the costume after trying it on in-store. In this case, the costume shop refunded them, and the shop owner deletes the purchases for the T-Rex costume. This action deletes all the records of the T-Rex costume along with the price and customer information.



1. In this step you compare the table created in #15 with a file that contains all of 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 #17. There are many formats you can use. Some examples include XML, flatfile, 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.

The following information was generated using Eclipse. The first tag <?xml version=*"1.0"* encoding=*"UTF-8"*?> is a standard tag that indicates the XML version as well as the encoding being used. XML structure is generally as follows:

<root>

<child>

<sub-child>.....</sub-child>

</child>

</root>

XML usually has only one root element, which is <CostumeShop> in the given example. The child element contains a row, which is <Purchase> containing purchase information in each row. The sub-child element contains all the columns that contain information on Costume, Price, CustomerName, and PurchaseDate.

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<CostumeShop>

<Purchase>

<Costume>Black Widow</Costume>

<Price>55.5</Price>

<CustomerName>Athena</CustomerName>

<PurchaseDate>10-OCT-2019</PurchaseDate>

</Purchase>

<Purchase>

<Costume>Batman</Costume>

<Price>92.3</Price>

<CustomerName>Felix</CustomerName>

<PurchaseDate>21-SEP-2020</PurchaseDate>

</Purchase>

<Purchase>

<Costume>Black Widow</Costume>

<Price>65.5</Price>

<CustomerName>Tyagi</CustomerName>

<PurchaseDate>30-SEP-2020</PurchaseDate>

</Purchase>

<Purchase>

<Costume>Batman</Costume>

<Price>77.7</Price>

<CustomerName>Zeke</CustomerName>

<PurchaseDate>01-NOV-2020</PurchaseDate>

</Purchase>

</CostumeShop>

* 1. 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?

There are a couple of ways to read an XML file. Using Eclipse, a project must be created and then a class file which will include the code in Java. To read and parse the file, *DOM Parser* must be downloaded, placed in the Lib folder that shall be created under the project, and the path must be set. Then, finally, run the project.

The application to access the file should know the path of the XML file. The application may malfunction if the file is relocated to a different path. If the application were to access the table from a database, the relocation of the file will not break or stop the application from accessing it.

The database is faster in retrieving data compared to the XML file. The XML file consumes time to read and parse a single record. A database is much more scalable and can support additional tables. The database can retrieve a record in seconds when there are billions of records, while retrieving a record using XML may take several hours or days.

A file system provides security at the file level; either a user has access to the file or they don’t. To provide access only to a piece of information within a file is not possible, however, the database makes it possible for a user to access only a particular table or a record within a table.

.

**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** | **A** | **B** | **C** | **D** | **F** |
| **Section 1: Quality (30%)** | The results for all steps in Section 1 are complete and correct. Appropriate SQL constructs have been used for all steps, and supporting explanations are present and accurate. All screenshots in Section 1 are legible. The section is well organized. All supporting explanations are clear and understandable. | The results for most steps in Section 1 are complete and correct. The appropriate SQL constructs have been used for most steps, and supporting explanations are mostly present and accurate. Most screenshots in Section 1 are legible. The section is organized. Most supporting explanations are clear and understandable. | The results for some steps in Section 1 are complete and correct. Appropriate SQL constructs have been used for some steps, and some supporting explanations are present and accurate. Some screenshots in Section 1 are legible. Some supporting explanations are clear and understandable. | The results for most steps in Section 1 are incomplete or incorrect. Appropriate SQL constructs have not been used for most steps. The screenshots in Section 1 are mostly illegible or missing. Most supporting explanations are unclear or missing. The section is disorganized. | The results for virtually all steps in Section 1 are incomplete or incorrect. Appropriate SQL constructs have not been used. Virtually all screenshots in Section 1 are illegible or missing. Virtually all supporting explanations are unclear or missing. The section is disorganized. |
| **Section 2: Quality (30%)** | The results for all steps in Section 2 are complete and correct. Appropriate SQL constructs have been used for all steps, and supporting explanations are present and accurate. All screenshots in Section 2 are legible. The section is well organized. All supporting explanations are clear and understandable. | The results for most steps in Section 2 are complete and correct. The appropriate SQL constructs have been used for most steps, and supporting explanations are mostly present and accurate. Most screenshots in Section 2 are legible. The section is organized. Most supporting explanations are clear and understandable. | The results for some steps in Section 2 are complete and correct. Appropriate SQL constructs have been used for some steps, and some supporting explanations are present and accurate. Some screenshots in Section 2 are legible. Some supporting explanations are clear and understandable. | The results for most steps in Section 2 are incomplete or incorrect. Appropriate SQL constructs have not been used for most steps. The screenshots in Section 2 are mostly illegible or missing. Most supporting explanations are unclear or missing. The section is disorganized. | The results for virtually all steps in Section 2 are incomplete or incorrect. Appropriate SQL constructs have not been used. Virtually all screenshots in Section 2 are illegible or missing. Virtually all supporting explanations are unclear or missing. The section is disorganized. |
| **Section 3: Anomaly Demonstration and Explanation (20%)** | Both anomaly types are accurately and completely demonstrated with SQL.The explanations for both anomaly types are accurate and complete. | Both anomaly types are demonstrated with SQL, in a way that is mostly complete and accurate. The explanations for both anomaly types are mostly accurate and complete. | Some of the anomaly types are demonstrated with SQL, in a way that is somewhat accurate. The explanations for some anomaly types are somewhat accurate. | The demonstration of both anomaly types with SQL is mostly incomplete and inaccurate. The explanations for both anomaly types are mostly inaccurate and incomplete. | The demonstration of both anomaly types is missing or completely innacurate. The explanations for both anomaly types are missing or completely inaccurate. |
| **Section 3: File Soundness (20%)** | The file contains all of the rows and columns as the relational table, in a form native to its format. The screenshot(s) of the file are legible and complete. The explanations comparing the structural independence, efficiency, and security between the table and file are accurate and complete. | The file contains most of the rows and columns as the relational table, in a form native to its format. The screenshot(s) of the file are mostly legible and complete. The explanations comparing the structural independence, efficiency, and security between the table and file are mostly accurate and complete. | The file contains some of the rows or columns as the relational table, in a form native to its format. The screenshot(s) of the file are somewhat legible and complete. The explanations comparing the structural independence, efficiency, and security between the table and file are somewhat accurate. | Most of the rows and columns in the relational table are missing from the file. The screenshot(s) of the file may be hard to read. The explanations comparing the structural independence, efficiency, and security between the table and file are mostly inaccurate or incomplete. | Virtually none fo the rows and columns in the relational table are present in the file. The screenshot(s) may be missing or illegible. The explanations comparing the structural independence, efficiency, and security between the table and file are missing or entirely inaccurate and incomplete. |

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.