

ACIT1630 Database Systems Fall 2023: Lab 07

Structured Query Language (SQL)

ABOUT THE LABS

There are 12 labs for you to complete during this course. They are all designed to be self-paced, so you should be able to complete them with little or no assistance. If you do find yourself having trouble with the work, make sure you ask your instructor before the next lab so that you don't fall behind. To get the most out of the labs you should be up to date with the reading and lecture material.

The lab exercises are assessable; however, completing them will assist you with both the practical and theoretical aspects of the unit, with the project and quizzes. Both phases of the project in this course assume that you have attempted and completed the lab exercises.

AIMS OF THE GUIDE

- Demonstrate competence in designing SELECT queries that incorporate subqueries to retrieve specific and relevant data from the database.
- Create SET queries and be able to determine the UNION COMPATIBILITY of a query.
- Add records to a table using INSERT.
- Update records in a table using UPDATE.
- Delete records from a table using DELETE.

ViewRidgeDatabase

In this lab, we are contemplating the utilization of the same database that has been our focus since the beginning of week 1.

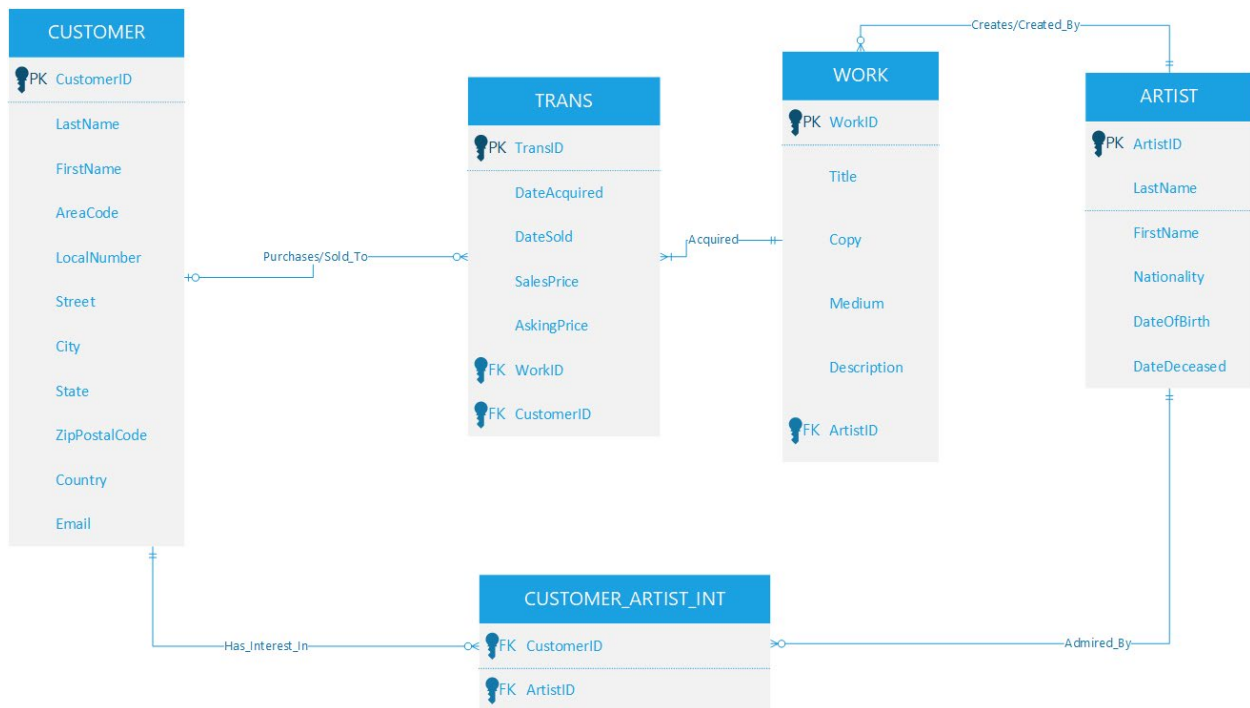
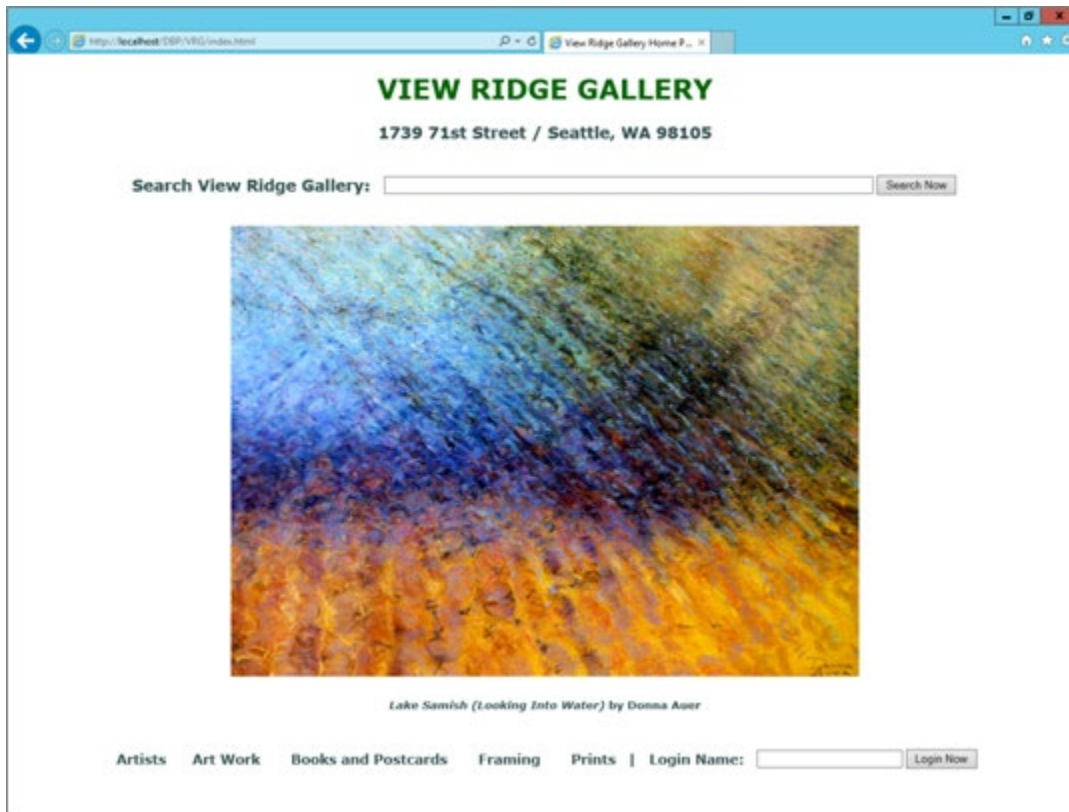


Image drawn from Kroenke, D.M., and Auer, D.J., 2016, Database Processing: Fundamentals, Design and Implementation, 14th Edition, Pearson, Boston.

1. More Queries

So far, you've learned about the process of choosing specific rows and columns from both single tables and multiple tables through the use of joins. In the upcoming section, we'll delve into more intricate SELECT queries. It's important to work through the provided examples to ensure a solid grasp of their functionality. These more advanced query concepts will be revisited in later lab exercises.

1.1 Subqueries

Subqueries are essentially entire SELECT queries that are nested within the WHERE clause of another query. Visualize the subquery as generating a temporary table of results that the outer query can tap into and incorporate. Subqueries seamlessly integrate with various operators like =, <, >, IN, NOT IN, EXISTS, NOT EXISTS, ANY, ALL, and SOME.

- ➔ Execute the following query to display the details of artworks crafted by Spanish artists:

```
SELECT * FROM WORK
WHERE ArtistID IN
  (SELECT ArtistID FROM ARTIST
   WHERE Nationality = 'Spanish');
```

- ➔ Subqueries employing IN or EXISTS offer an alternative approach to constructing a join. For instance, this query could have been alternatively expressed as:

```
SELECT W.*
FROM WORK W, ARTIST A
WHERE W.ArtistID = A. ArtistID
AND Nationality = 'Spanish';
```

- ➔ Run the query to verify that both forms yield identical results.

Nevertheless, it's essential to recognize that subqueries cannot always replace a join. When the columns in the SELECT statement come from multiple tables, a join must be utilized. For instance, if the query mentioned above also included the title of the work in the SELECT statement, it would necessitate being expressed as a join.

2. Set Queries

In Topic 2, you were introduced to relational algebra set operations like UNION, INTERSECTION, and MINUS. However, it's important to note that **MySQL does not support these specific operators**. Unlike some other database systems, MySQL doesn't provide built-in support for these set operations. In MySQL, you'll need to use other methods or queries to achieve similar outcomes when combining the results of two separate statements, such as **subqueries**. The key consideration remains the same: the result tables from these statements must have compatible columns in terms of number and data type, just as in relational algebra.

Question 1: Develop an SQL query that will retrieve the following:



Submit
Screenshot

1. Return the first and last names of the artists who are French, OR who were born BEFORE 1870, OR BOTH.
2. Return the first and last names of the artists who are French AND were born BEFORE 1870.
3. Return the first and last names of the artists who are French and were NOT born before 1870.

Question 2: Now, Solve the following activities using the View Ridge Gallery

database.



Submit
Screenshot

Try to solve these queries yourself using the tables Artist, Work and Trans.

4. List the full name (i.e., First Name, Last Name) of all Artists and their date of birth, sorted so that the older an artist is, the nearer they will be to the top of the list.
5. List the Titles of works of art created by Artists who are French; do this as both a JOIN and a SUBQUERY.
6. List the Titles and the name of the artist, of works of art created by Artists who are French.
7. List the artist's name, the title, copy and medium of the work of art, and the date that it was sold.
8. List the title, copy and medium of all works of art, and, if they have been sold, the date they were sold.

3. Altering Database Content: INSERT, UPDATE, DELETE

Three SQL commands are employed to make changes to the data within database tables: INSERT, DELETE, and UPDATE. Their usage will be demonstrated in the following section.

Please note that these modifications can be made on the existing table or you can create a duplicate table (see Other clauses).

3.1 Modifying Records Using INSERT

To add records, you use the INSERT statement in SQL. The values you want to insert should be listed in the same order as the fields in the table, and they must match the data types accordingly. Character and date fields need to be enclosed in single quotes, while numeric fields should not have quotes. If a field has no value, you can enter "NULL" (without quotes) to represent that. The fundamental syntax of the INSERT command is as follows:

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

If data values are being inserted into ALL columns, you don't need to specify the column list, but the data values must align with the order in which the columns are defined (you can verify this order using the DESCRIBE tablename command).

➔ To insert a new record into the Customer table, use the following query:

```
INSERT INTO customer
VALUES (
1059, 'Doe', 'John', 'johndoe@example.com', 'hashedpassword', '123 Main St', 'Los Angeles', 'CA', '90001', 'USA', '123', '4567890'
);
```

➔ Use the following query to insert a record into the Customer table, including values exclusively for the ID, FirstName, LastName, and Email address columns:

```
INSERT INTO CUSTOMER (CustomerID, FirstName, LastName, EmailAddress)
VALUES (1068, 'Alice', 'Johnson', 'alice@alicejohnson.com');
```

Now, attempt to execute the identical command once more to insert another record with the same CustomerID. What outcome do you observe? Which constraint is being breached in this case?

As previously noted, when you create a table using CREATE TABLE AS, it inherits the NOT NULL constraint but not other constraints like primary key or foreign key. You must add these constraints separately.

You can also copy records for insertion from another table by utilizing a SELECT query to fetch the desired records:

```

INSERT INTO destination_table (column1, column2, column3, ...)
    SELECT column1, column2, column3, ...
    FROM source_table
    WHERE condition;

```

To test this, you can create a table EXTRACUSTOMER using this command:

```

CREATE TABLE EXTRACUSTOMER(
    CustomerID int AUTO_INCREMENT,
    LastName char(25),
    FirstName char(25),
    EmailAddress varchar(100),
    EncryptedPassword varchar(50),
    Street char(30),
    City char(35),
    State char(2),
    ZIPorPostalCode char(9),
    Country char(50),
    AreaCode char(3),
    PhoneNumber char(8),
    CONSTRAINT ExtraCustPK PRIMARY KEY (CustomerID));

```

```

INSERT INTO extracustomer
(LastName, FirstName, EmailAddress, EncryptedPassword, Street, City, State, ZIPorPostalCode, Country, AreaCode, PhoneNumber)
VALUES
('Doe', 'John', 'johndoe@example.com', 'hashedpassword1', '123 Main St', 'Los Angeles', 'CA', '90001', 'USA', '123', '4567890'),
('Smith', 'Alice', 'alicesmith@example.com', 'hashedpassword2', '456 Elm St', 'New York', 'NY', '10001', 'USA', '456', '7891234'),
('Johnson', 'Fred', 'fredjohnson@example.com', 'hashedpassword3', '789 Oak St', 'Chicago', 'IL', '60601', 'USA', '789', '2345678');

```

➔ Some more records to introduce some diversity.

```
INSERT INTO extracustomer
```

```
(LastName, FirstName, EmailAddress, EncryptedPassword, Street, City, State, ZIPorPostalCode, Country, AreaCode, PhoneNumber)
```

```
VALUES
```

```
('Williams', 'Emily', 'emily@example.com', 'hashedpassword7', '111 Oak St', 'London', 'ENG', 'SW1A 1AA', 'United Kingdom', '111', '9998887'),  
( 'Martinez', 'Carlos', 'carlos@example.com', 'hashedpassword8', '222 Maple St', 'Paris', 'FRA', '75001', 'France', '222', '8887776'),  
( 'Gonzalez', 'Sofia', 'sofia@example.com', 'hashedpassword9', '333 Elm St', 'Berlin', 'DEU', '10115', 'Germany', '333', '7776665');
```

If you encounter the "You are in safe mode" error, you need to disable safe mode by setting the value to 0, and then re-enable it by changing the value to 1.

```
SET SQL_SAFE_UPDATES = 0;
```

If you encounter an error related to truncation, you are required to alter the table. The warnings you're seeing indicate that there's a data truncation issue for the 'State' column in your INSERT statements. The 'State' column seems to be defined with a size of '2' characters, and the data you're trying to insert exceeds this size for the rows with 'ENG', 'FRA', and 'DEU' as the states.

To resolve this issue, you can either increase the size of the 'State' column in your table definition to accommodate longer state codes or adjust the data you're trying to insert to fit within the defined size. Here's an example of how to change the 'State' column size to '3' character.

```
ALTER TABLE extracustomer  
MODIFY COLUMN State char(3);
```

Try to Insert the newly inserted record into the CUSTOMER. Is there any constraint that is violated? Or will be violated if we try to insert more record into EXTRACUSTOMER and try to copy them to CUSTOMER?

3.2 Modifying Records Using UPDATE.

The UPDATE command allows you to modify the contents of existing records. Here's the basic syntax of the UPDATE command:

```
UPDATE tablename  
  
SET columnName1 = dataValue1 [, columnName2 = dataValue2 ...]  
  
[WHERE searchCondition];
```

For the upcoming exercises, we'll use a table named " EXTRACUSTOMER".

- ➔ To view the structure of the "EXTRACUSTOMER" table and its data, use the DESCRIBE command to familiarize yourself with it.
- ➔ In an UPDATE query, if there's no "WHERE" condition, all rows will be affected. To test it, you can create a copy of the artist table and name as ARTISTTEST (refer to other clauses section).
- ➔ Check the record of ARTISTTEST, which is unique.
- ➔ Now try to test the update clause without the where condition.

```
UPDATE ARTISTTEST
SET LastName = 'Walker', FirstName = 'Susan', Nationality = 'French';
```

- ➔ Check the records of ARTISTTEST table.

Is there any difference you have noticed before and after using the update without clauses?

- ➔ To update ALL customer's information, execute the following query as an example:

```
UPDATE EXTRACUSTOMER
SET LastName = 'Anderson', FirstName = 'Linda', EmailAddress = 'linda.anderson@gmail.com'
WHERE CustomerID = 4;
```

If you want to update only specific records, you can use a WHERE clause to identify them.

- ➔ To update the state of customers with a specific condition (e.g., those from United Kingdom), enter the following query:

```
UPDATE extracustomer
SET State = 'UK'
WHERE Country = 'United Kingdom';
```

You can similarly update other attributes like LastName, FirstName, or any other attribute based on your requirements.

After running these queries, verify that they have worked.

To change specific attribute values back to their original values, enter queries to revert the changes as needed.

4. Deleting records using DELETE

To delete records from the "EXTRACUSTOMER" table using the DELETE command, you can follow a syntax given below:

```
DELETE FROM TABLENAME  
WHERE condition;
```

- ➔ Delete records from the "EXTRACUSTOMER" table where specific conditions are met: For example, if you want to delete customers with a specific LastName, you can use:

```
DELETE FROM extracustomer  
WHERE LastName = 'Brown';
```

- ➔ To delete all records from the "extracustomer" table:

```
DELETE FROM extracustomer;
```

This will delete all records in the "extracustomer" table without any conditions.

- ➔ To check if there are no rows left in the "extracustomer" table:

```
SELECT COUNT(*) FROM extracustomer;
```

This query will return the count of rows in the "extracustomer" table. If it returns 0, it means there are no rows left in the table.

Remember to be cautious when using the DELETE command, especially without a WHERE condition, as it will delete all records from the specified table. Always ensure that you have backups of your data before performing any deletion operation (Refer to other clauses to learn how to backup your data).

5. Other Clauses:

5.1 Create Duplicate Table:

Generate a duplicate of the CUSTOMER by utilizing CREATE TABLE AS for utilization in the upcoming exercises.

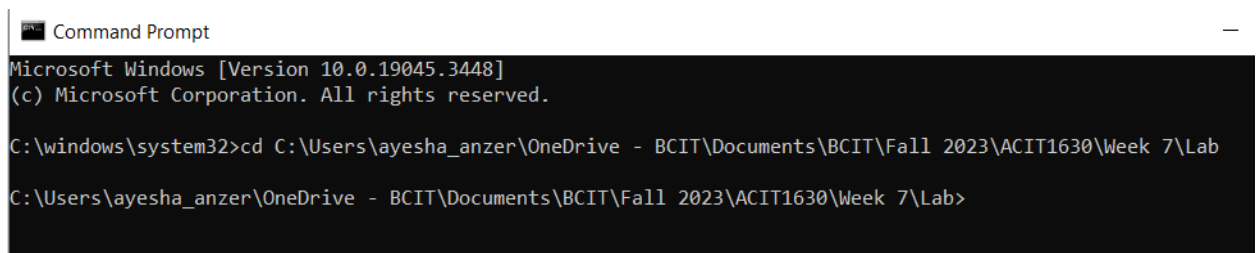
```
CREATE TABLE CUSTOMER AS  
SELECT *  
FROM CUSTOMER;
```

5.2 Create a Backup

In MySQL, you can create a backup of a table using various methods. One common approach is to use the mysqldump utility, which allows you to create a SQL dump file containing the table's data and structure.

Here's how you can **create a backup of a table** using mysqldump:

- ➔ Open your command prompt or terminal as administrator.
- ➔ Navigate to folder where you want to save your backup file using cd command, such as,



```
Command Prompt  
Microsoft Windows [Version 10.0.19045.3448]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\windows\system32>cd C:\Users\ayesha_anzer\OneDrive - BCIT\Documents\BCIT\Fall 2023\ACIT1630\Week 7\Lab  
C:\Users\ayesha_anzer\OneDrive - BCIT\Documents\BCIT\Fall 2023\ACIT1630\Week 7\Lab>
```

- ➔ Use the following command to create a backup of a specific table in a database:

```
mysqldump -u [username] -p [database_name] [table_name] > [backup_file.sql]
```

Replace [username] with your MySQL username, [database_name] with the name of the database containing the table, [table_name] with the name of the table you want to backup, and [backup_file.sql] with the desired name of the backup SQL file (e.g., backup.sql).

- ➔ Use the following command to create a backup of the entire database:

```
mysqldump -u username -p database_name > backup_file.sql
```

Replace [username] with your MySQL username, [database_name] with the name of the database, and [backup_file.sql] with the desired name of the backup SQL file (e.g., backup.sql).

- ➔ After running the command, you will be prompted to enter your MySQL password.
- ➔ Once you've entered the password, the mysqldump utility will create an SQL dump file containing the table's data and structure or database in the specified file.
- ➔ You can find the backup file in the current directory of your command prompt or terminal.

5.3 Restore the table from the backup file.

To restore the table or database from the backup, you can use the mysql command-line tool or a MySQL client to execute the SQL dump file.

- ➔ Here's an example of how to restore a table or database from a backup file:

```
mysql -u [username] -p [database_name] < [backup_file.sql]
```

Replace [username], [database_name], and [backup_file.sql] with your MySQL username, the database name, and the backup SQL file, respectively.

Remember to replace placeholders with your specific values and ensure that you have appropriate permissions to perform these operations in your MySQL environment.

What to submit:

1. Submit your SQL file. Rename your file as Lab7_firstname_lastname23.sql. Please note make sure to change firstname and lastname to your name, such as Lab7_Ayesha_Anzer23.sql
2. You have been given a template named "ACIT1630 2023 Lab 7 Template.docx" that needs to be completed. Please download it from the learning hub. Ensure that when saving your file, you rename it in the format Lab7_firstname_lastname23.docx. For example, Lab7_Ayesha_Anzer23.docx". The screenshot should include both the query and its output, with a visible success message in the output. The description should provide comprehensive information about the clauses you have utilized.
3. **Do not zip the files! Submit 2 files separately.**