# Recap: Version control

## **Introduction**

When you develop software products, you’ll most likely collaborate as part of a group. In these groups, each developer works on a different aspect of the project and makes regular changes to the source code as required.

However, these changes don’t always integrate well with the existing code. Or they may not integrate well with code being developed concurrently by another developer. Instances such as these may inadvertently introduce bugs into the project. So, it’s important to maintain track of all changes made to a project and undo any changes that damage the code using version control.

In this reading, you’ll recap the basics of version control and the Git version control system in preparation for your exercise tasks.

## **Version Control**

As you know from previous courses, version control system (or VCS) is a system for maintaining consistency in changes to a project.

There are many different types of VCS, including Git, Subversion, CVS and ClearCase. These different VCS make use of different methodologies. However, the fundamental principal of each remains the same: provide a system that allows a group of individuals to collaborate on a project at the same time.

In this project, you’ll use Git as your VCS. You may be familiar with Git from projects you completed in previous courses. Git is a free and open-source software used to maintain code integrity.

## **Branching**

Version control is the idea that a record can be maintained over time of the changes and activities that have occurred in a project. To facilitate version control for individual users, as well as the team in general, Git enables branching.

Branching involves downloading a copy of the code to a local depository. The user is then free to work on this code as required. Any changes that the user makes can be saved with a meaningful title, such as “bug fix”, “typo” or “function X added”. These changes are recorded with a time stamp so that other users can tell when the changes occurred. If an error is detected, then the user can just roll back the changes to an earlier implementation.

## **Distributed System**

Collaborating on a project as part of a group means that there are many people working on a code base at the same time.

Through branching it is possible for each member of the project to have a local instance on their machine. When a collaborator is happy that their code is ready for distribution then they can add it to the main code repository. This is known as “making a merge”.

During the merging process, Git highlights the changes between the local instance of the code and the centralized version. The programmer is then asked to pick which instance of the code to keep in the central repository.

## **Conclusion**

In this reading, you recapped the basics of version control and Git. You also reviewed the importance of working collaboratively, the obstacles involved, and how you can avoid these obstacles using version control system.

# Recap: Database setup in MySQL Workbench and SQL basics

## **Introduction**

As a database engineer working with massive amounts of data, you need to build a proper database system that meets the data requirements of your business.

There are multiple database managements systems available for you to use. There are also different developments tools that you can use to create and manage your databases efficiently.

In this reading, you’ll review some of these systems, including:

* MySQL database management system,
* MySQL Workbench development tool,
* SQL basics,
* And how to set up your database environment.

This recap will support you in preparing for your exercise tasks.

## **MySQL**

MySQL is an open-source relational database management system. It’s one of the most widely used management systems for building both personal and industry relational databases.

There are several advantages offered to database engineers who work with MySQL:

* Build a secure database system,
* Offers high level scalability,
* Provides high Performance,
* 24 hours x 7 days a week uptime with cluster servers,
* Supports robust transactional database engines,
* Complete Workflow Control,
* Reduced cost,
* And facilitates easy to use and efficient database management.

## **MySQL Workbench**

MySQL Workbench is a unified visual tool for database modeling and database management. It was developed by Oracle, the company that manages MySQL.

MySQL Workbench is also free, open-source and cross-platform tool that runs on Windows, Mac and Linux. MySQL Workbench enables you to speed up the database design and development process.

Key features of MySQL Workbench include the following:

* Simplify database design and maintenance.
* Document data requirements in the form of visualized ER diagrams.
* Support model-driven database development.
* Automate the development process without the need for much coding.
* Design data models that meet the expected standards.
* Provide capabilities for creating data models in the visual designer.
* Use the Forward and the Reverse Engineer methods to create data models and databases.
* Maintain different versions of database schemas.
* Support schema synchronization and comparison functionality.
* Use the DBDoc documentation tool to document all details of your database design.
* Facilitate visual SQL editor for easy use to write, run and debug SQL statements.
* Provide complete administrative control over all aspects of MySQL server.
* Import and export mysqldump files, and export query results sets as CSV, XML and HTML files.

## **SQL Basics**

SQL (Structured Query Language) is the standard query language for accessing and manipulating relational databases. SQL provides four categories of commands to create, manipulate and manage data in a database known as DDL, DML, DCL and TCL.

* Data Definition Language (DDL) commands are used to define and modify tables in a database such as Create, Drop, Alter and Truncate commands.
* Data Manipulation Language (DML) commands are used to query, delete and update data in the database such as select, insert, update and delete commands.
* Data Control Language (DCL) commands are used to control the roles and privileges of users of a database system such as Grant and Revoke commands.
* Transaction Control Language (TCL) to manage transactions in the database such as Commit and Rollback Commands.

If you want to learn more about each category of commands and how to use them, you can review [Common SQL Commands](https://www.coursera.org/learn/introduction-to-databases/supplement/x4kOK/common-sql-commands) item in the ‘Introduction to database’ course.

## **Database setup**

There are few issues you need to consider when setting up MySQL Workbench to create and manage your databases.

* Create a new user. This is the most secure way to connect to your MySQL database, as you can manage user roles and privileges.
* Create an instance of a MySQL server. This can be done in the MySQL Workbench home screen. In the home-screen side panel, you can select MySQL Connections option to view or create instances of MySQL.
* Use connections option to load, configure, group and view information about each of your MySQL connections. You can select “Users and Privileges” under the “Management” menu to view a list of current database users.
* Use your MySQL connection to begin working with database schemas and SQL queries.

In MySQL Workbench, you can create your database in different ways. For example, you can create and use your database by writing the relevant SQL statements.

Alternatively, you can create a data model in the model designer first, then transform it into a SQL schema that can be executed and implemented directly in a MySQL server without typing any SQL code.

To learn more about this process, you can review the [Database modeling in MySQL Workbench](https://www.coursera.org/learn/advanced-data-modeling/lecture/bIYwL/database-modeling-in-mysql-workbench) item in the Advanced data modeling course.

## **Conclusion**

In this reading you learned about the various categories of SQL commands, MySQL database management system and MySQL Workbench. You are now ready to start developing your database system and complete the different tasks in this lesson.

# Recap: Creating an Entity Relationship Model in MySQL Workbench

## **Introduction**

There are several models that you can choose from to help Little Lemon create a suitable database system including an Entity Relationship Model, a Hierarchical Data Model, an Object-Oriented Data Model and a Dimensional Data Model. Each model has its advantages and disadvantages and achieves a specific objective.

In this reading you will review the Entity Relationship Model, as it is the best fit for Little Lemon’s data requirements. You will also recap the ER diagram that shows the entities, the relationships and the multiplicities. In addition, you’ll recap an overview of the three fundamental methods of normalization.

## **The Entity Relationship Model**

This model presents each table as an entity with a set of attributes in an ER diagram. This model considers all types of multiplicities in the relationships between the tables including one-to-one, one-to-many, and many-to-many relationships.

The Entity Relationship Model plays a major role helping to develop a proper structure of your database system. It is the most prominent and widely used database model that uses the ER Diagram to represent the database structure.

To help Little Lemon develop a suitable data model you should consider three levels of data modeling.

## **Conceptual data model**

This is an abstract level of a data element’s structure called entities, where you identify the various data sets and data flow in your business. The main idea of the conceptual model is to present an overall picture of the database system by recognizing the entities involved and the relationships between them. The conceptual data model represents the basis for building the logical data model.

## **Logical data model**

This model covers a more detailed level than the conceptual model. At this level you should be more specific in terms of the information and relationships as follows:

* Identify the attributes of each entity.
* Define the primary keys to identify unique records of data in each table.
* And specify the foreign keys to create relevant relationships between tables.

## **The physical data model**

This model specifies a detailed level of the database structure. It’s used to create the internal SQL schema of the database, which is implemented in the DBMS. This model involves:

* Defining each attribute with a specific data type such as VARCHAR, INT and Decimal.
* And applying relevant constraints such as NOT NULL and DEFAULT.

To learn more about the three levels of Entity Relationship data modeling, review [Overview of data modeling](https://www.coursera.org/learn/advanced-data-modeling/item/0dDnH) item in the ‘Advanced data modeling‘ course.

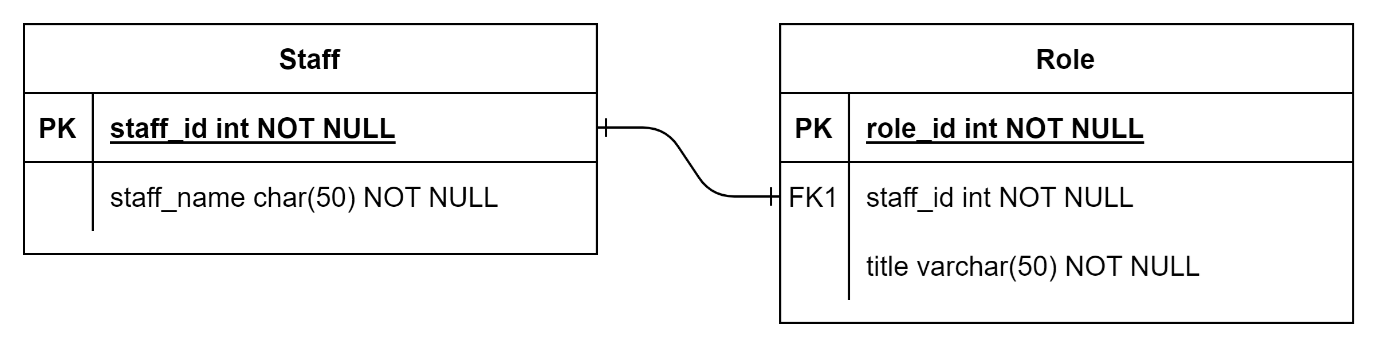
## **Relationships in ER Diagram**

There are three types of relationships between any two tables in the Entity Relationship database, summarized as follows:

### **One-to-One**

In a One-to-One relationship, each single record of one table is associated with only one single record of another table, as shown below.

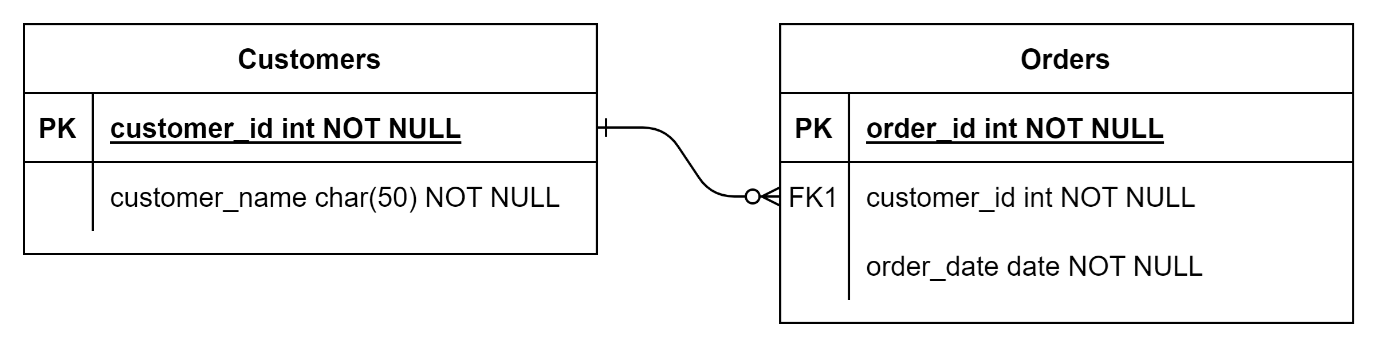
### **One to One Relationship**



### **One-to-Many**

In a One-to-Many relationship, each single record of one table is associated with one or many records of another table, as shown below.

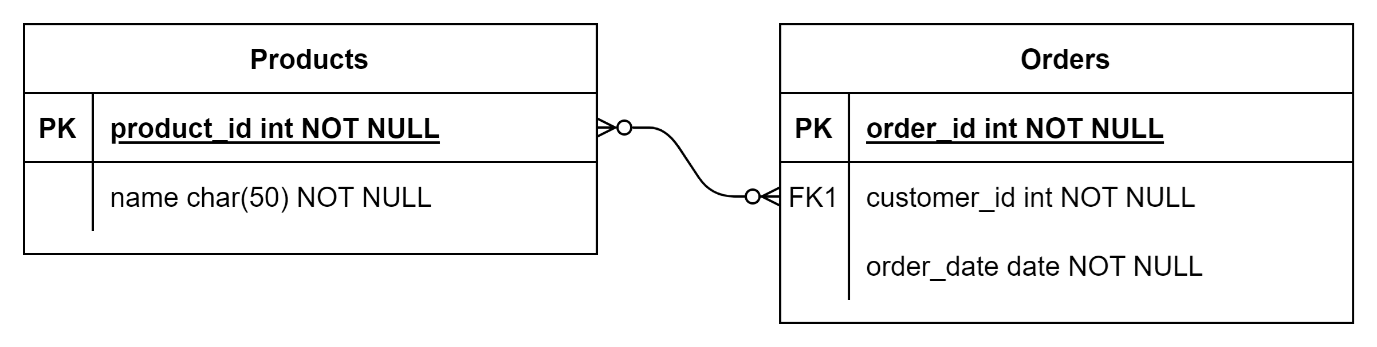
### **One to Many Relationship**



### **Many-to-Many**

In a Many-to-Many relationship, each record of one table is associated with multiple records of another table. And this relationship also works in the other direction, as shown below.

### **Many to Many Relationship**

  
If you want to learn more about these different types relationships, you can review the [Table relationships](https://www.coursera.org/learn/introduction-to-databases/lecture/b7UUl/table-relationships) item in the ‘Introduction to Databases’ course.

## **Entity Relationship Database Normalization**

In a relational database, if you want to delete or update data, you need (most likely) to do it everywhere. If you miss any field or commit a mistake, you’ll end up with incorrect or inconsistent information. These kinds of problems are referred to as the insert, update and delete anomalies.

To solve these problems, you need to apply the normalization process. This process aims to minimize data duplications, avoid errors during data modifications and simplify data queries from the database. The three fundamental normalization forms are summarized below.

### **First Normal Form (1NF)**

The 1NF form enforces the data atomicity rule and eliminates unnecessary repeating groups of data. The data atomicity rule means that you can only have one single instance value of the column attribute in any cell of the table.

### **Second Normal Form (2NF)**

For a relation in a database to be in the second normal form, it must already be in the first normal form. In addition, you need to avoid any partial dependency relationships. These occur when a non-key attribute depends only on one part of the composite primary key (a key that consists of a combination of two or more columns).

### **Third Normal Form (3NF)**

For a relation in a database to be in the third normal form, it must already be in the second normal form. In addition, it must have no transitive dependency. This means that any non-key attribute in the table may not be functionally dependent on another non-key attribute in the same table.

If you want to learn more about the three different normalization methods, you can review the [Normalization of relational DB models](https://www.coursera.org/learn/advanced-data-modeling/item/HrH2J) item in the Advanced data modeling course.

## **Conclusion**

In this reading you recapped the Entity Relationship Model, the three different levels of ER modeling, the multiplicity relationship between entities and the commonly used normalization methods. You are now ready to start developing an Entity Relationship diagram for Little Lemon and complete the different tasks in this lesson.

# Additional resources

The following resources offer additional reading materials on setting up your project:

* [IBM guide to data modeling](https://www.ibm.com/cloud/learn/data-modeling)
* [Microsoft guide to database normalization](https://learn.microsoft.com/en-us/office/troubleshoot/access/database-normalization-description)
* [Oracle tutorial on getting started with MySQL Workbench](https://docs.oracle.com/cd/E19078-01/mysql/mysql-workbench/wb-getting-started-tutorial.html)
* [GitHub "Getting started" tutorial](https://docs.github.com/en/get-started)

# Recap: MySQL virtual tables and JOINS

## **Introduction**

To manage your data in a relational database, you must have a good understanding of how to access, extract, process and manipulate information using various SQL queries.

In this reading you will recap different SQL queries with a particular focus on subqueries, virtual tables and JOINS.

## **Overview of SQL queries**

To extract all data existing in a table, you can use the primary form of SQL query as follows:

SELECT \* FROM table;

To perform more specific queries, you may need to filter data, and use operators, aliases and aggregate functions.

For instance, sometimes you need to extract specific information from your database. This involves writing SQL queries with specific filtering criteria.

For example, to obtain data from a table with specific dates you can use the following syntax.:

SELECT column\_name

FROM table\_name

WHERE date\_column > date\_value;

In some cases, existing data is labelled with long, complicated or less meaningful names, which may confuse users. In this situation, you can use an alias to display a particular column with a more meaningful name.

An alias is created by using the AS keyword as shown in the following syntax:

SELECT column\_a AS 'Column A'

FROM table\_name;

Also, using aliases can be very important when making use of JOIN statements.

You can also use the aggregate functions to summarize data in your database such as COUNT, SUM, MIN, MAX, and AVERAGE. For example, you can use the following syntax to count all values in a particular column.

SELECT COUNT(column\_name)

FROM table\_name;

To perform more sophisticated SQL queries, you need to be familiar with subqueries, JOINS and virtual tables.

## **Subqueries**

A subquery is a query within another query. You can create a subquery as an inner query and include it in another query called outer query. The inner query can be thought of as the child query and the outer query as the parent query.

Subqueries, also called nested queries, can be placed in the SELECT, FROM or WHERE clause to return relevant data. In a subquery you can use several types of operators including ANY, ALL, EXIST and NOT EXISTS.

The following syntax makes use of the EXISTS operator to test for the existence of any records in the subquery:

SELECT column\_name(s)

FROM table\_name

WHERE EXISTS (SELECT column\_name FROM table\_name WHERE condition);

To learn more about subqueries in relational database, you can review the [JOINS](https://www.coursera.org/learn/database-structures-and-management-with-mysql/lecture/4vYAK/joins) item in the ‘Database Structures and Management with MySQL’ lesson.

## **Joins**

JOINS are important for obtaining data from multiple tables in relational databases. You can use different types of JOIN to explore the relationships between different records of data that exist in different tables.

The main JOINS supported in MySQL include the INNER JOIN, the LEFT JOIN and the RIGHT JOIN.

SELECT columns

FROM table1 INNER JOIN table2

ON table1.column = table2.column;

To learn more about the different JOINS supported in MySQL, you can review the [JOINS](https://www.coursera.org/learn/database-structures-and-management-with-mysql/lecture/4vYAK/joins) item in ‘Database Structures and Management with MySQL’.

## **Virtual tables**

Virtual tables are very helpful for summarizing data. The SQL view creates a virtual table from existing tables in the database. Key advantages of virtual tables include:

* Focus on specific data in the database.
* Reduce data returned from a query.
* Include complex calculations and functions.
* Save time in querying data.
* Reduce report complexity.

The basic syntax to create a virtual table in SQL is as follows:

CREATE VIEW virtual\_table AS SELECT column FROM table;

To learn more about virtual tables, you can review the [MySQL Create View](https://www.coursera.org/learn/database-structures-and-management-with-mysql/item/t41NM) item in ‘Database Structures and Management with MySQL’.

## **Conclusion**

In this reading, you recapped the fundamentals of SQL queries with a particular focus on subqueries, virtual tables and JOINS. You are now ready to help Little Lemon query data from their database by solving different tasks and activities introduced in this lesson.

# Recap: MySQL query optimization with stored procedures and Prepared Statements

## **Introduction**

In this reading, you’ll recap the importance of query optimization. You’ll also recap techniques that can be used to optimize databases, such as stored procedures and prepared statements.

## **Query optimization**

When working with MySQL databases, the response time, or turnaround time, is extremely important. It’s particularly important in terms of how long the database takes to respond to your SQL queries. As data volumes grow, and the data requirements grow increasingly more complex, then performance becomes more important for a better end-user experience.

Database optimization is the best way to reduce database system response time. Response time is the time taken to transmit a query, process it, and transmit the response or information back to the user.

The performance of a database depends on both the software and the hardware. However, the focus of this reading is on software or database-level performance optimization. At the database level, the structure of the database, and the queries that you execute, play an important role in performance optimization.

As you should know from previous courses, you can optimize database queries using stored procedures and prepared statements. Let’s recap the basics of these techniques.

## **Stored procedures**

When working with a database, you may often want to reuse the same code of SQL statements repeatedly. In this case, you can save your block of code in the form of a stored procedure that you can use whenever needed.

When you build a stored procedure, you need to adhere to the following best practices:

* Each procedure must have a unique name.
* Each procedure may have zero or many parameters.
* Each procedure may contain one or more SQL statements.
* Each procedure may contain one or more variable and/or control statements.

Stored procedures optimize your code and make it more consistent, reusable and easier to use and maintain.

## **Create a stored procedure**

To create a stored procedure, you must use the CREATE PROCEDURE command, specify the procedure name with the parameters list and create the procedure logic with relevant SQL code.

The following syntax presents a block of code that represents the basic syntax of a stored procedure:

CREATE PROCEDURE ProcedureName()

SELECT statements;

In this syntax, the CREATE PROCEDURE command creates the stored procedure, followed by the procedure name.

The name of the procedure must be followed with a pair of parentheses ( ), which holds the list of parameters. You need to use a pair of parentheses even if you have no parameters in the procedure (in other words, even if the parameter list is empty).

The last part of the procedure contains the logic you want to implement. For example, you might require your procedure to select all data from a table in the database.

## **Stored procedures with multiple parameters**

You can also create a more sophisticated stored procedure with multiple parameters and statements. However, these procedures require additional steps to consider in the development process.

For example, when creating these procedures, you first need to change the delimiter and then use the BEGIN-END keywords to wrap the code in the body of the procedure as follows:

DELIMITER //

Changing the delimiter enables MySQL to compile the whole block of code in the BEGIN-END section of the stored procedure as one compound statement.

CREATE PROCEDURE ProcedureName (OUT parameter INT, IN parameter INT)

BEGIN

SELECT statement1\_name;

SELECT statement2\_name;

END //

In this example, the two parameters are declared as OUT and IN types, as the intention is input and to output some values.

The logic is implemented within the BEGIN and the END keywords which determines the body of the procedure.

In this case the two statements were created in the body of the procedure.

The double forward slashes // indicate the end of the entire query.

Finally, you can change the delimiter back to the default semicolon so that you can keep using MySQL.

DELIMITER;

## **Invoke a stored procedure**

Once you’ve created your stored procedure you need to invoke it. To invoke the stored procedure, you need to use the CALL command followed by the procedure name, as in the following example:

CALL ProcedureName;

To learn more about procedures, you can review the item [Stored procedures in MySQL](https://www.coursera.org/learn/database-structures-and-management-with-mysql/item/KB5Tc) in the ‘Database Structure and Management in MySQL’ course. You can also review the item [Functions and stored procedures in MySQL](https://www.coursera.org/learn/advanced-mysql-topics/lecture/l8bCl/functions-and-stored-procedures-in-mysql) in the ‘Advanced MySQL Topics’ course.

## **Prepared statements**

A prepared statement is a database optimization technique used to execute database queries faster and more securely. You can also invoke it to reuse the same SQL statements repeatedly. This leads to higher efficiency.

Creating a prepared statement in MySQL reduces the time it takes to parse a query. The query is parsed once by MySQL regardless of the number of times it is used and executed.

A prepared statement also minimizes the bandwidth usage when communicating data with a MySQL server. This is because you only need to pass the parameter’s values to the prepared statement, not the whole SQL statement.

It also provides greater database security by preventing the use of database injections to hack your database.

## **Creating a prepared statement**

A prepared statement in MySQL is essentially a template that you create with a SQL statement that includes certain unspecified values which are used as parameters.

In the following example, these values are labelled with a question mark "?" which represents four parameters that can be inserted into the table:

PREPARE statement\_Name 'INSERT INTO table1 VALUES (?, ?, ?, ?)';

## **Calling a prepared statement**

To call the prepared statement, you need to use the execute command as follows:

EXECUTE statement\_name USING @variable;

To learn more about prepared statements, you can review the item [MySQL Prepared Statement](https://www.coursera.org/learn/advanced-mysql-topics/item/Me8ob) in the ‘Advanced MySQL Topics’ course.

## **Conclusion**

In this reading, you recapped the fundamentals of stored procedures and prepared statements. You are ready now to help Little Lemon optimize their database queries.

# Additional resources

The following resources offer additional reading materials on performing database queries using procedures and prepared statements:

* [MySQLtutorial guide to MySQL prepared statements](https://www.mysqltutorial.org/mysql-prepared-statement.aspx)
* [MySQLtutorial guide to MySQL stored procedures](https://www.mysqltutorial.org/getting-started-with-mysql-stored-procedures.aspx)
* [MySQLtutorial guide to creating SQL views](https://www.mysqltutorial.org/create-sql-views-mysql.aspx)

# Recap: SQL queries and transactions

## **Introduction**

To manage your data in a relational database, you must have a good understanding of how to access, extract, process and manipulate information using various SQL queries.

In this reading you’ll recap various types of SQL queries and learn more about triggers and transactions in MySQL.

## **Overview of SQL queries**

### **Basic and complex SQL queries**

To extract all existing data from a table, you can use the primary form of a SQL query as follows:

SELECT \* FROM table;

If you need to perform more specific queries, then you may need to filter data, and use operators, alias and aggregate functions.

For instance, you may need to extract specific information from your database. In this instance, you may need to write SQL queries to obtain data from a table with specific dates as demonstrated in the following basic syntax.

SELECT column\_name

FROM table\_name

WHERE date\_column > date\_value;

### A**liases**

In some cases, you might find that existing data has been labelled with long, complicated or less meaningful names, which may confuse database users. In this case, you can use an alias to display a particular column with a more meaningful name.

The use of an alias can also be very important when working with JOIN statements.

An alias is created using the AS keyword as shown in the following syntax:

SELECT column\_a AS 'Column A' FROM table;

### **Aggregate functions**

You can also use aggregate functions to summarize data in your database such as COUNT, SUM, MIN, MAX, and AVERAGE.

For example, you can use the following syntax to count all values in a particular column.

SELECT COUNT(column\_name)

FROM table\_name;

To perform more sophisticated SQL queries, you need to be familiar with more advanced SQL concepts such as subqueries, JOINS, virtual tables, stored procedures and prepared statements.

These topics have already been covered in previous items, in addition to triggers and transactions concepts, which are important when performing complex operations.

## **Triggers**

### **Overview of triggers**

A MySQL trigger is a set of actions available in the form of a stored program. The set of actions is invoked automatically when certain events occur. You can use triggers for different types of events, like inserting, deleting and updating data.

Triggers can be used in MySQL for several purposes such as:

* Enforcing business rules,
* Ensure data integrity,
* Insert records to other tables for audit trail purposes,
* Query data for reporting purposes,
* and replicate data to different tables to achieve data consistency.

### **Trigger syntax**

To use a trigger, you first need to create it using the CREATE TRIGGER statement. Then define the trigger type. Is it an INSERT, UPDATE or DELETE trigger? And should it be executed BEFORE or AFTER the event?

You also need to define the trigger’s logic, specify which table it’s assigned to, and how it should be applied to the table.

An example of the syntax used to create a trigger in MySQL is as follows:

CREATE TRIGGER trigger\_name

{BEFORE | AFTER} {INSERT | UPDATE| DELETE}

ON table\_name FOR EACH ROW

trigger\_body;

The syntax to remove a MySQL trigger is as follows:

DROP TRIGGER [IF EXISTS] [schema\_name] trigger\_name

To learn more about Triggers, you can review the [About triggers in MySQ](https://www.coursera.org/learn/advanced-mysql-topics/item/U8zM2)L item in ‘Advanced MySQL Topics’ course.

## **Transactions**

### **Overview of transactions**

You should use MySQL transactions when carrying out critical activities, as they can be used to roll your transaction back to its original state if any failure occurs.

This is especially important when writing a series of related queries that must all work together to achieve the expected outcome. If a specific query fails, then you can use the transaction statement to undo the changes that took place.

### **Transaction syntax**

MySQL provides a set of statements to manage database transactions.

The START TRANSACTION statement is the standard SQL statement used to start a transaction process. The COMMIT statement is used to commit the current transaction and make the transaction changes permanently, as shown in the following example:

START TRANSACTION;

SQL statements

COMMIT;

If you want to roll back the current transaction and cancel the changes made to the database, you can use the ROLLBACK statement as follows:

START TRANSACTION;

SQL statements

ROLLBACK;

To learn more about transactions, you can review the [MySQL Transaction](https://www.coursera.org/learn/advanced-mysql-topics/lecture/oBz8v/mysql-transaction) item in the ‘Advanced MySQL Topics’ course.

## **Conclusion**

In this reading, you recapped the basics of SQL queries, triggers and transactions. You are now ready to help Little Lemon develop their table booking system in this lesson’s exercises.

# Additional resources

The following resources offer additional reading materials around the use of MySQL queries and transactions:

* [MySQLtutorial guide to MySQL transaction statements](https://www.mysqltutorial.org/mysql-transaction.aspx)
* [MySQLtutorial guide to MySQL triggers](https://www.mysqltutorial.org/mysql-triggers/)

# Recap: Working with Tableau

## **Introduction**

Data analytics is a complex process. You need to analyze large amounts of data to find answers to your questions. But this task is beyond the capabilities of traditional database management systems. That’s why data analysis is performed using data analytics tools. These tools make it possible for users to view and understand large amounts of data using powerful artificial intelligence.

In this reading you’ll recap the basics of using Tableau to perform visualized data analytics.

## **About Tableau**

The Tableau platform is a leading analytics platform. It supports efficient visual data analytics, data storytelling and interactive dashboards with live updates.

You can download the desktop version of Tableau on your machine from the following link:

* <https://www.tableau.com/products/desktop/download>

All the steps and technical requirements for installing Tableau are listed on the Tableau website in the following link:

* <https://www.tableau.com/products/techspecs>

## **Getting Started with Tableau**

Tableau is used to analyze data and provide answers to different questions about business performance. However, before you can start analyzing data you need to connect Tableau to your data sources.

To establish a live connection, you need to open Tableau on the connection page and carry out the following steps:

* Under the Connect tab on the left-hand side, select the required data source (for example: “Microsoft Excel”).
* This opens a dialog box that you can use to navigate to the Excel file on your machine.
* Once you open the file, the Excel file name appears on the left-hand side of the screen. The data from the Excel file is displayed in the data pane.

After you import the data you need, you can start preparing it for analysis. This process involves fixing errors in the data and shaping it so that it’s easier to understand and analyze.

To learn more about connecting to a data source, you can review the item [Data import and data preparation in Tableau](https://www.coursera.org/learn/advanced-data-modeling/item/AAMTM) in the lesson ‘Introduction to data analytics’ in the ‘Advanced data modeling’ course.

## **Data Visualization**

Database analysis isn’t just about extracting and analyzing the information you need from the database. How you present your data is also just as important. Good data visualization helps decision makers to interpret the data and make the right choices around it.

There are many different factors to consider when deciding what type of visual data to represent, summarized as follows:

* Identify your target audience.
* Specify what information your visualization includes.
* Consider how much time you want your audience to spend examining each visualized chart.
* Determine the level of accuracy your audience is looking for.

Once you’ve done this, it’s time to decide on your data visualization charts. Along this line, Tableau allows you to develop several visualized charts such as Bubble chart, Pie chart, Bar chart, Line chart, and Map chart.

Each chart has a different purpose. And each chart can be used to solve a different type of problem or communicate a different kind of message. For example, you can use a line chart to show trends, or maps to depict location-based data.

What’s most important is to select a chart that tells your audience the story of your data in the most appropriate way.

To learn more about data visualization, you can review item [Overview of data visualization](https://www.coursera.org/learn/advanced-data-modeling/item/bPNzP) in ‘Introduction to data analytics’ in the ‘Advanced data modeling’ course.

## **Data Filtering in Tableau**

You can filter data in either the source page or the worksheet.

However, filtering data directly in the data source limits your data analysis in all worksheets to the filtered criteria only.

To filter data in the source, you need to select the Add option under “Filters” in Tableau source page. This opens a dialog box that lists all fields in the data source. You select then the data fields and criteria you want to apply to your data filtering.

To filter data in a worksheet, you need to open the sheet tab at the bottom of the data source page. Then, in the worksheet, the columns from your data source are displayed as fields on the left side of the data pane. The data pane contains a variety of fields. The fields above the gray line are dimension fields. The fields below the grey line are measure fields. You can then drag the required fields into the Filter box in the Card area, where you can apply different types of data filtering.

To learn more about data filtering in Tableau, you can review the [Data Filtering and Visualization in Tableau](https://www.coursera.org/learn/advanced-data-modeling/lecture/X7DoV/data-filtering-and-visualization-in-tableau) item in the ‘Introduction to data analytics’ lesson in the ‘Advanced data modeling’ course.

## **Conclusion**

In this reading, you recapped the basics of data analytics in Tableau. You should be able now to access and make use of the Tableau work environment to visualize your data and complete all exercises in this lesson.