9.2 Lesson Summary - Advanced SQL Queries

SQL provides functionality to do more than basic CRUD database transactions. You can perform mathematical calculations on your data and organize the results in ways to optimize processing once the data has been retrieved from your database. Using SQL to perform calculations and organization of your data is usually much faster than retrieving all of your data and performing similar calculations with either Python or Pandas.

Concept: SQL provides the functionality to **aggregate** data based on search criteria. The *AVG, COUNT, MIN, MAX, SUM* functions can be used to create **aggregates** of your data and be combined them with *GROUP BY, HAVING, SELECT*. To display the average rental rate of a film per rental duration you could use the following query:

*SELECT rental\_duration, AVG(rental\_rate) AS "Average rental rate"*

*FROM film*

*GROUP BY rental\_duration;*

* Activity: 02-Ins\_Aggregates, 03-Stu\_GregariousAggregates
* Suppl link: <https://www.w3schools.com/sql/sql_groupby.asp>

Concept: It is often useful to **order** the results of your SQL query by using the ORDER BY keyword. If you wanted to order the query results by average length you could use the following query:

*SELECT rental\_rate, AVG(length) AS "avg length"*

*FROM film*

*GROUP BY rental\_rate*

*ORDER BY "avg length";*

* Activity: 04-Ins\_Order\_By, 05-Stu\_Order\_By
* Suppl link: <https://www.w3schools.com/sql/sql_orderby.asp>

Concept: Queries in SQL can be nested allowing you to perform more complex **subqueries** on a database and target only the data that you need. If you wanted to find every film in your inventory that corresponded to a film titled 'EARLY HOME' in your film table you could use the following query:

*SELECT \**

*FROM inventory*

*WHERE film\_id IN*

*(*

*SELECT film\_id*

*FROM film*

*WHERE title = 'EARLY HOME'*

*);*

* Activity: 07-Stu\_Subqueries, 10-Ins\_Revist\_Subquery, 11-Stu\_Mine\_the\_Subquery

Concept: Queries that are frequently reused to access data from a database can be saved to a database as a **VIEW**. Keep in mind VIEWs don't create any new data or organize data into a new table, VIEWs are queries that can then be referenced as if they were tables in your database. To create a view of all people for a people table you could use the following query:

*CREATE VIEW all\_people AS*

*SELECT \**

*FROM people;*

You wouldn't usually create a view of all people since the all\_people view contains exactly the same data as the people table. This example illustrates that views can be used just like tables even though they contain no additional data.

* Activity: 08-Ins\_Create\_Views, 09-Stu\_View\_Room\_Queries