1. **Write complex queries to extract the results from a database using SQL.**

Complex SQL queries can be challenging to write, but there are ways to make them simpler. Here are a few pointers to help you write complex SQL queries:

1. Multiple AND OR operators: Queries that use multiple AND OR operators are called complex queries.
2. The use of subqueries and correlated subqueries: Subqueries are also complex.
3. Using Joins
4. Regular Expression Queries
5. Questions about the group by and having clauses
6. Set operators in queries

You can also simplify a complex SQL query by replacing complex subqueries with CTEs (or Common table expressions) or by identifying a better SQL feature (or combination of features) to achieve the same result in fewer lines of code (Tchigladze, 2023).

- Query to find the highest sales of a product in a given time period:

```
SELECT ProductName,
MAX(TotalSales)
FROM SalesTable
WHERE Date
BETWEEN '2022-01-01' AND '2022-12-31'
GROUP BY ProductName;
```

- Query to find the top 5 customers with the highest total purchase amount:

```
SELECT CustomerName,
SUM(TotalPurchaseAmount) as 'Total Purchase'
FROM OrdersTable
JOIN CustomersTable
ON OrdersTable.CustomerID = CustomersTable.CustomerID
GROUP BY CustomerName
ORDER BY 'Total Purchase' DESC
LIMIT 5;
```

- Query to find the average salary of employees in each department:

```
SELECT DepartmentName,
AVG(Salary) as 'Average Salary'
FROM EmployeesTable
JOIN DepartmentsTable
ON EmployeesTable.DepartmentID = DepartmentsTable.DepartmentID
GROUP BY DepartmentName;
```

- Query to find the number of orders placed by each customer in the last 30 days:

```
SELECT CustomerID,
COUNT(*) as 'Number of Orders'
FROM OrdersTable
WHERE Date >= DATEADD(day, -30, GETDATE())
GROUP BY CustomerID;
```

- Query to find the most popular products based on the number of orders:

```
SELECT ProductName,
COUNT(*) as 'Number of Orders'
FROM OrderDetailsTable
JOIN ProductsTable
ON OrderDetailsTable.ProductID = ProductsTable.ProductID
GROUP BY ProductName
ORDER BY 'Number of Orders' DESC;
```

These are just a few examples of complex SQL queries that can retrieve information from a database. The complexity of the queries is determined by the specific requirements of the data to be extracted.

2. Apply aggregate, arithmetic, date and string functions in SQL for summarised reporting.

SQL has many built-in functions that can be used to perform various data operations

(Beaulieu, 2009). For instance, here are a few of the most frequently used functions:

1. Aggregate functions:

Aggregate functions allow you to perform calculations across multiple rows of a table.

Some standard aggregate functions include:

3. **COUNT**: returns the number of rows in a table that meets a specific condition
4. **SUM**: returns the sum of the values in a column
5. **AVG**: returns the average of the values in a column
6. **MIN**: returns the minimum value in a column
7. **MAX**: returns the maximum value in a column

Example:

```
SELECT COUNT(*)
AS total_customers,
SUM(order_total) AS total_sales,
AVG(order_total) AS avg_order_total
FROM orders
WHERE order_date >= '2022-01-01';
```

2. Arithmetic functions:

Arithmetic functions allow you to perform mathematical calculations in SQL. Some common arithmetic functions include:

- +: addition
- -: subtraction
- *: multiplication
- /: division
- %: modulus (returns the remainder of a division operation)

Example:

```
SELECT product_id, unit_price, unit_price * 1.2 AS new_price
FROM products
WHERE category_id = 1;
```

3. Date functions:

Date functions allow you to work with dates and times in SQL. Some standard date functions include:

- DATEADD: adds a specific interval to a date
- DATEDIFF: calculates the difference between two dates
- DATEPART: extracts a specific part of a date (e.g., year, month, day, hour, minute, second)

Example:

```
SELECT order_id, order_date,
DATEPART(week, order_date) AS week_number
FROM orders
WHERE YEAR(order_date) = 2022;
```

4. String functions:

String functions allow you to work with text data in SQL. Some standard string functions include:

- CONCAT: combines two or more strings into one
- SUBSTRING: extracts a portion of a string
- UPPER: converts a string to uppercase
- LOWER: converts a string to lowercase

Example:

```
SELECT CONCAT(first_name, ' ', last_name) AS full_name,
UPPER(email) AS uppercase_email
FROM customers
WHERE city = 'New York';
```

These are just a few examples of how to apply aggregate, arithmetic, date, and string functions in SQL for summarised reporting. The functions you use will depend on the specific requirements of your reporting.

3. Evaluate query results.

To evaluate query results, you need to analyse the data returned and ensure that it meets the desired criteria (Team, 2021). For instance, here are some critical steps to evaluating query results:

1. **Check for errors:** When evaluating query results, the first thing to do is to look at the number of rows returned. This will assist you in ensuring that the query returns the expected results. If the number of rows returned differs significantly from what you expected, the query may need to be revised.
2. **Check for completeness**: Ensure the query has returned all the expected data. If not, review the query and make necessary adjustments.
3. **Check for accuracy**: Review the data to ensure that it is accurate and meets the expected criteria. To ensure accuracy, you may need to compare the query results with other data sources.
4. **Check for consistency**: Check that the query results are consistent with the data in the database. If there are discrepancies, you may need to investigate further to identify the cause.
5. **Check for relevancy:** Evaluate the query results regarding their relevance to the business problem or question being addressed. If the results are irrelevant, you may need to modify the query or the underlying data to get better results.

Following these steps, you can evaluate the query results to ensure they meet your criteria. This will allow you to make more informed decisions based on the information returned by the query.

References:

Tchigladze, I. (2023). *How to Simplify Complex SQL Queries (With Examples)*. [online] Available at: https://www.stratascratch.com/blog/how-to-simplify-complex-sql-queries-with-examples/ [Accessed 30 Apr. 2023].

Beaulieu, A. (2009). *Learning SQL: master SQL fundamentals*. " O'Reilly Media, Inc.".

Team, dbForge (2021). *SQL Query Optimization: 12 Useful Performance Tuning Tips and Techniques*. [online] Devart Blog. Available at: https://blog.devart.com/how-to-optimize-sql-query.html.