Data Engineer Interview Preparation SQL Questions

1. What is SQL?

It stands for Structured Query Language. A programming language used for interaction with relational database management systems (RDBMS). This includes fetching, updating, inserting, and removing data from tables.

2. What are SQL dialects? Give some examples.

The various versions of SQL, both free and paid, are also called SQL dialects. All the flavors of SQL have a very similar syntax and vary insignificantly only in additional functionality. Some examples are Microsoft SQL Server, PostgreSQL, MySQL, SQLite, T-SQL, Oracle, and MongoDB.

3. What are the main applications of SQL?

Using SQL, we can:

- create, delete, and update tables in a database
- access, manipulate, and modify data in a table
- retrieve and summarize the necessary information from a table or several tables
- add or remove certain rows or columns from a table

All in all, SQL allows querying a database in multiple ways. In addition, SQL easily integrates with other programming languages, such as Python or R, so we can use their combined power.

4. What is an SQL statement? Give some examples.

Also known as an SQL command. It's a string of characters interpreted by the SQL engine as a legal command and executed accordingly. Some examples of SQL statements are `SELECT`, `CREATE`, `DELETE`, `DROP`, `REVOKE`, and so on.

5. What types of SQL commands (or SQL subsets) do you know?

- **Data Definition Language (DDL)** – to define and modify the structure of a database.
- **Data Manipulation Language (DML)** – to access, manipulate, and modify data in a database.
- **Data Control Language (DCL)** – to control user access to the data in the database and give or revoke privileges to a specific user or a group of users.

- **Transaction Control Language (TCL)** – to control transactions in a database.
- **Data Query Language (DQL)** – to perform queries on the data in a database to retrieve the necessary information from it.

6. Give some examples of common SQL commands of each type.

- **DDL:** `CREATE`, `ALTER TABLE`, `DROP`, `TRUNCATE`, and `ADD COLUMN`
- **DML:** `UPDATE`, `DELETE`, and `INSERT`
- **DCL:** `GRANT` and `REVOKE`
- **TCL:** `COMMIT`, `SET TRANSACTION`, `ROLLBACK`, and `SAVEPOINT`
- **DQL:** – `SELECT`

7. What is a database?

A structured storage space where the data is kept in many tables and organized so that the necessary information can be easily fetched, manipulated, and summarized.

8. What is DBMS, and what types of DBMS do you know?

It stands for Database Management System, a software package used to perform various operations on the data stored in a database, such as accessing, updating, wrangling, inserting, and removing data. There are various types of DBMS, such as relational, hierarchical, network, graph, or object-oriented. These types are based on the way the data is organized, structured, and stored in the system.

9. What is RDBMS? Give some examples of RDBMS.

It stands for Relational Database Management System. It's the most common type of DBMS used for working with data stored in multiple tables related to each other by means of shared keys. The SQL programming language is particularly designed to interact with RDBMS. Some examples of RDBMS are MySQL, PostgreSQL, Oracle, MariaDB, etc.

10. What are tables and fields in SQL?

A table is an organized set of related data stored in a tabular form, i.e., in rows and columns. A field is another term for a column of a table.

11. What is an SQL query, and what types of queries do you know?

A query is a piece of code written in SQL to access the data from a database or to modify the data. Correspondingly, there are two types of SQL queries: **select** and **action** queries. The first ones are used to retrieve the necessary data (this also includes limiting, grouping, ordering the data, extracting the data from multiple tables, etc.), while the second ones are used to create, add, delete, update, rename the data, etc.

12. What is a subquery?

Also called an inner query; a query placed inside another query, or an outer query. A subquery may occur in the clauses such as `SELECT`, `FROM`, `WHERE`, `UPDATE`, etc. It's also possible to have a subquery inside another subquery. The innermost subquery is run first, and its result is passed to the containing query (or subquery).

13. What types of SQL subqueries do you know?

- **Single-row** – returns at most one row.
- **Multi-row** – returns at least two rows.
- **Multi-column** – returns at least two columns.
- **Correlated** – a subquery related to the information from the outer query.
- **Nested** – a subquery inside another subquery.

14. What is a constraint, and why use constraints?

A set of conditions defining the type of data that can be input into each column of a table. Constraints ensure data integrity in a table and block undesired actions.

15. What SQL constraints do you know?

- `DEFAULT` – provides a default value for a column.
- `UNIQUE` – allows only unique values.
- `NOT NULL` – allows only non-null values.
- `PRIMARY KEY` – allows only unique and strictly non-null values (`NOT NULL` and `UNIQUE`).
- `FOREIGN KEY` – provides shared keys between two and more tables.

16. What is a join?

A clause used to combine and retrieve records from two or multiple tables. SQL tables can be joined based on the relationship between the columns of those tables. Check out our SQL joins tutorial for more context.

17. What types of joins do you know?

- `(INNER) JOIN` – returns only those records that satisfy a defined join condition in both (or all) tables. It's a default SQL join.
- `LEFT (OUTER) JOIN` – returns all records from the left table and those records from the right table that satisfy a defined join condition.
- `RIGHT (OUTER) JOIN` – returns all records from the right table and those records from the left table that satisfy a defined join condition.
- `FULL (OUTER) JOIN` – returns all records from both (or all) tables. It can be considered as a combination of left and right joins.

18. What is a primary key?

A column (or multiple columns) of a table to which the `PRIMARY KEY` constraint was imposed to ensure unique and non-null values in that column. In other words, a primary key is a combination of the `NOT NULL` and `UNIQUE` constraints. The primary key uniquely identifies each record of the table. Each table should contain a primary key and can't contain more than one primary key.

19. What is a unique key?

A column (or multiple columns) of a table to which the `UNIQUE` constraint was imposed to ensure unique values in that column, including a possible `NULL` value (the only one).

20. What is a foreign key?

A column (or multiple columns) of a table to which the `FOREIGN KEY` constraint was imposed to link this column to the primary key in another table (or several tables). The purpose of foreign keys is to keep connected various tables of a database.

21. What is an index?

A special data structure related to a database table and used for storing its important parts and enabling faster data search and retrieval. Indexes are especially efficient for large databases, where they significantly enhance query performance.

22. What types of indexes do you know?

- **Unique index** – doesn't allow duplicates in a table column and hence helps maintain data integrity.
- **Clustered index** – defines the physical order of records of a database table and performs data searching based on the key values. A table can have only one clustered index.

- **Non-clustered index** – keeps the order of the table records that doesn't match the physical order of the actual data on the disk. It means that the data is stored in one place and a non-clustered index – in another one. A table can have multiple non-clustered indexes.

23. What is a schema?

A collection of database structural elements such as tables, stored procedures, indexes, functions, and triggers. It shows the overall database architecture, specifies the relationships between various objects of a database, and defines different access permissions for them.

24. What is a SQL comment?

A human-readable clarification on what a particular piece of code does. SQL code comments can be single-line (preceded by a double dash `--`) or span over multiple lines (as follows: `/*comment_text*/`). When the SQL engine runs, it ignores code comments. The purpose of adding SQL code comments is to make the code more comprehensive for those people who will read it in the future.

25. What is a SQL operator?

A reserved character, a combination of characters, or a keyword used in SQL queries to perform a specific operation. SQL operators are commonly used with the `WHERE` clause to set a condition (or conditions) for filtering the data.

26. What types of SQL operators do you know?

- **Arithmetic** (`+`, `-`, `*`, `/`, etc.)
- **Comparison** (`>`, `<`, `=`, `>=`, etc.)
- **Compound** (`+=`, `-=`, `*=`, `/=`, etc.)
- **Logical** (`AND`, `OR`, `NOT`, `BETWEEN`, etc.)
- **String** (`%`, `_`, `+`, `^`, etc.)
- **Set** (`UNION`, `UNION ALL`, `INTERSECT`, and `MINUS` (or `EXCEPT`))

27. What is an alias?

A temporary name given to a table (or a column in a table) while executing a certain SQL query. Aliases are used to improve the code readability and make the code more compact. An alias is introduced with the `AS` keyword:

```
SELECT col_1 AS column
```

```
FROM table_name;
```

28. What is a clause?

A condition imposed on a SQL query to filter the data to obtain the desired result. Some examples are `WHERE`, `LIMIT`, `HAVING`, `LIKE`, `AND`, `OR`, `ORDER BY`, etc.

29. What are some common statements used with the `SELECT` query?

The most common ones are `FROM`, `GROUP BY`, `JOIN`, `WHERE`, `ORDER BY`, `LIMIT`, and `HAVING`.

30. How to create a table?

Using the `CREATE TABLE` statement. For example, to create a table with three columns of predefined datatypes, we apply the following syntax:

```
CREATE TABLE table_name (col_1 datatype,

                col_2 datatype,

                col_3 datatype);
```

31. How to update a table?

Using the UPDATE statement. The syntax is:

```
UPDATE table_name

SET col_1 = value_1, column_2 = value_2

WHERE condition;
```

32. How to delete a table from a database?

Using the `DROP TABLE` statement. The syntax is: `DROP TABLE table_name;`.

33. How to get the count of records in a table?

Using the `COUNT()` aggregate function with the asterisk passed as its argument: `SELECT COUNT(*) FROM table_name;`.

34. How to sort records in a table?

Using the `ORDER BY` statement:

```
SELECT * FROM table_name

ORDER BY col_1;
```

We can specify that we need a descending order using the `DESC` keyword; otherwise, the order will be ascending by default. Also, we can sort by more than one column and specify for each one, ascending or descending order separately. For example:

```
SELECT * FROM table_name

ORDER BY col_1 DESC, col_3, col_6 DESC;
```

35. How to select all columns from a table?

Using the asterisk * with the `SELECT` statement. The syntax is: `SELECT * FROM table_name;`.

36. How to select common records from two tables?

```
Using the INTERSECT statement:

SELECT * FROM table_1

INTERSECT

SELECT * FROM table_1;
```

37. What is the DISTINCT statement and how do you use it?

This statement is used with the `SELECT` statement to filter out duplicates and return only unique values from a column of a table. The syntax is:

```
SELECT DISTINCT col_1

FROM table_name;
```

38. What are entities? Give some examples.

An entity is a real-world object, creature, place, or phenomenon for which the data can be gathered and stored in a database table. Each entity corresponds to a row in a table, while the table's columns describe its properties. Some examples of entities are bank transactions, students in a school, cars sold, etc.

39. What are relationships? Give some examples.

Relationships are the connections and correlations between entities, basically meaning how two or more tables of a database are related to one another. For example, we can find an ID of the same client in a table on sales data and in a customer table.

40. What is NULL value? How is it different from zero or a blank space?

A `NULL` value indicates the absence of data for a certain cell of a table. Instead, zero is a valid numeric value, and an empty string is a legal string of zero length.

SQL FUNCTIONS AND MORE:

1. What is a function in SQL, and why use functions?

A database object representing a set of SQL statements frequently used for a certain task. A function takes in some input parameters, performs calculations or other manipulations on them, and returns the result. Functions help improve code readability and avoid repetition of the same code snippets.

2. What types of SQL functions do you know?

- **Aggregate functions** – work on multiple, usually grouped records for the provided columns of a table, and return a single value (usually by group).
- **Scalar functions** – work on each individual value and return a single value.

On the other hand, SQL functions can be built-in (defined by the system) or user-defined (created by the user for their specific needs).

3. What aggregate functions do you know?

- `AVG()` – returns the average value
- `SUM()` – returns the sum of values
- `MIN()` – returns the minimum value
- `MAX()` – returns the maximum value
- `COUNT()` – returns the number of rows, including those with null values
- `FIRST()` – returns the first value from a column
- `LAST()`– returns the last value from a column

4. What scalar functions do you know?

- `LEN()` (in other SQL flavors – `LENGTH()`) – returns the length of a string, including the blank spaces
- `UCASE()` (in other SQL flavors – `UPPER()`) – returns a string converted to the upper case
- `LCASE()` (in other SQL flavors – `LOWER()`) – returns a string converted to the lower case
- `INITCAP()` – returns a string converted to the title case (i.e., each word of the string starts from a capital letter)
- `MID()` (in other SQL flavors – `SUBSTR()`) – extracts a substring from a string
- `ROUND()` – returns the numerical value rounded to a specified number of decimals
- `NOW()` – returns the current date and time

5. What are case manipulation functions? Give some examples.

Case manipulation functions represent a subset of character functions, and they're used to change the case of the text data. With these functions, we can convert the data into the upper, lower, or title case.

- `UCASE()` (in other SQL flavors – `UPPER()`) – returns a string converted to the upper case
- `LCASE()` (in other SQL flavors – `LOWER()`) – returns a string converted to the lower case
- `INITCAP()` – returns a string converted to the title case (i.e., each word of the string starts from a capital letter)

6. What are character manipulation functions? Give some examples.

Character manipulation functions represent a subset of character functions, and they're used to modify the text data.

- `CONCAT()` – joins two or more string values appending the second string to the end of the first one
- `SUBSTR()` – returns a part of a string satisfying the provided start and end points

- `LENGTH()` (in other SQL flavors – `LEN()`) – returns the length of a string, including the blank spaces
- `REPLACE()` – replaces all occurrences of a defined substring in a provided string with another substring
- `INSTR()` – returns the numeric position of a defined substring in a provided string
- `LPAD()` and `RPAD()` – return the padding of the left-side/right-side character for right-justified/left-justified value
- `TRIM()` – removes all the defined characters, as well as white spaces, from the left, right, or both ends of a provided string

7. What is the difference between local and global variables?

Local variables can be accessed only inside the function in which they were declared. Instead, global variables, being declared outside any function, are stored in fixed memory structures and can be used throughout the entire program.

8. What is the default data ordering with the `ORDER BY` statement, and how do you change it?

By default, the order is ascending. To change it to descending, we need to add the `DESC` keyword as follows:

```
SELECT * FROM table_name

ORDER BY col_1 DESC;
```

9. What set operators do you know?

- `UNION` – returns the records obtained by at least one of two queries (excluding duplicates)
- `UNION ALL` – returns the records obtained by at least one of two queries (including duplicates)
- `INTERSECT` – returns the records obtained by both queries
- `EXCEPT` (called `MINUS` in MySQL and Oracle) – returns only the records obtained by the first query but not the second one

10. What operator is used in the query for pattern matching?

The LIKE operator in combination with the % and _ wildcards. The % wildcard represents any number of characters including zero, while _ – strictly one character.

11. What is the difference between a primary key and a unique key?

While both types of keys ensure unique values in a column of a table, the first one identifies uniquely each record of the table, and the second one prevents duplicates in that column.

12. What is a composite primary key?

The primary key of a table, based on multiple columns.

13. What is the order of appearance of the common statements in the SELECT query?

SELECT – FROM – JOIN – ON – WHERE – GROUP BY – HAVING – ORDER BY – LIMIT

14. In which order the interpreter executes the common statements in the SELECT query?

FROM – JOIN – ON – WHERE – GROUP BY – HAVING – SELECT – ORDER BY – LIMIT

15. What is a view, and why use it?

A virtual table containing a subset of data retrieved from one or more database tables (or other views). Views take very little space, simplify complex queries, limit access to the data for security reasons, enable data independence, and summarize data from multiple tables.

16. Can we create a view based on another view?

Yes. This is also known as nested views. However, we should avoid nesting multiple views since the code becomes difficult to read and debug.

17. Can we still use a view if the original table is deleted?

No. Any views based on that table will become invalid after deleting the base table. If we try to use such a view anyway, we'll receive an error message.

18. What types of SQL relationships do you know?

- **One-to-one** – each record in one table corresponds to only one record in another table
- **One-to-many** – each record in one table corresponds to several records in another table

- **Many-to-many** – each record in both tables corresponds to several records in another table

19. What are the possible values of a BOOLEAN data field?

In some SQL flavors, such as PostgreSQL, the BOOLEAN data type exists explicitly and takes values `TRUE`, `FALSE`, or `NULL`. In other flavors, such as Microsoft SQL Server, the BIT datatype is used to store Boolean values as integers `1` (true) or `0` (false).

20. What is normalization in SQL, and why use it?

Normalization is a process of database design that includes organizing and restructuring data in a way to reduce data redundancy, dependency, duplication, and inconsistency. This leads to enhanced data integrity, more tables within the database, more efficient data access and security control, and greater query flexibility.

21. What is denormalization in SQL, and why use it?

Denormalization is the process opposite of normalization: it introduces data redundancy and combines data from multiple tables. Denormalization optimizes the performance of the database infrastructure in situations when read operations are more important than write operations since it helps avoid complex joins and reduces the time of query running.

22. What is the difference between renaming a column and giving an alias to it?

Renaming a column means permanently changing its actual name in the original table. Giving an alias to a column means giving it a temporary name while executing an SQL query, with the purpose to make the code more readable and compact.

23. What is the difference between nested and correlated subqueries?

A correlated subquery is an inner query nested in a bigger (outer) query that refers to the values from the outer query for its execution, meaning that a correlated subquery depends on its outer query. Instead, a non-correlated subquery doesn't rely on the data from the outer query and can be run independently of it.

24. What is the difference between clustered and non-clustered indexes?

While a clustered index **defines the physical order of records** of a table and performs data searching based on the key values, a non-clustered index **keeps the order of records that doesn't match the physical order of the actual data** on the disk. A table can have only one clustered index but many non-clustered ones.

25. What is the `CASE()` function?

The way to implement the *if-then-else* logic in SQL. This function sequentially checks the provided conditions in the `WHEN` clauses and returns the value from the corresponding `THEN` clause when the first condition is satisfied. If none of the conditions is satisfied, the function returns the value from the `ELSE` clause in case it's provided, otherwise, it returns `NULL`. The syntax is:

```
CASE

  WHEN condition_1 THEN value_1

  WHEN condition_2 THEN value_2

  WHEN condition_3 THEN value_3

  ...

  ELSE value

END;
```

26. What is the difference between the `DELETE` and `TRUNCATE` statements?

`DELETE` is a reversible DML (Data Manipulation Language) command used to delete one or more rows from a table based on the conditions specified in the `WHERE` clause. Instead, `TRUNCATE` is an irreversible DDL (Data Definition Language) command used to delete all rows from a table. `DELETE` works slower than `TRUNCATE`. Also, we can't use the `TRUNCATE` statement for a table containing a foreign key.

27. What is the difference between the `DROP` and `TRUNCATE` statements?

`DROP` deletes a table from the database completely, including the table structure and all the associated constraints, relationships with other tables, and access privileges. `TRUNCATE` deletes all rows from a table without affecting the table structure and constraints. `DROP` works slower than `TRUNCATE`. Both are irreversible DDL (Data Definition Language) commands.

28. What is the difference between the HAVING and WHERE statements?

The first one works on aggregated data after they are grouped, while the second one checks each row individually. If both statements are present in a query, they appear in the following order: `WHERE` – `GROUP BY` – `HAVING`. The SQL engine interprets them also in the same order.

29. How do you add a record to a table?

Using the INSERT INTO statement in combination with VALUES. The syntax is:

```
INSERT INTO table_name

VALUES (value_1, value_2, ...);
```

30. How to delete a record from a table?

Using the DELETE statement. The syntax is:

```
DELETE FROM table_name

WHERE condition;
```

In this way, we can also delete multiple records if they satisfy the provided condition.

31. How to add a column to a table?

Using the ALTER TABLE statement in combination with ADD. The syntax is:

```
ALTER TABLE table_name

ADD column_name datatype;
```

32. How to rename a column of a table?

Using the ALTER TABLE statement in combination with RENAME COLUMN ... TO ... The syntax is:

```
ALTER TABLE table_name

RENAME COLUMN old_column_name TO new_column_name;
```

33. How to delete a column from a table?

Using the ALTER TABLE statement in combination with DROP COLUMN. The syntax is:

```
ALTER TABLE table_name

DROP COLUMN column_name;
```

34. How to select all even or all odd records in a table?

By checking the remainder of the division by 2. In some SQL versions (e.g., PostgreSQL and My SQL), we use the MOD function, in the others (Microsoft SQL Server and SQLite) – the modulo operator (%). To select all even records using MOD:

```
SELECT * FROM table_name

WHERE MOD(ID_column, 2) = 0;
```

To select all even records using `%`:

```
SELECT * FROM table_name

WHERE ID_column % 2 = 0;
```

To select all odd records, the syntax is identical in both cases, only that we would use the inequality operator `<>` instead of `=`.

35. How to prevent duplicate records when making a query?

Using the `DISTINCT` statement in combination with `SELECT` or creating a unique key for that table.

36. How to insert many rows in a table?

Using the `INSERT INTO` statement in combination with `VALUES`. The syntax is:

```
INSERT INTO table_name

VALUES (value_1, value_2, ...),

    (value_3, value_4, ...),

    (value_5, value_6, ...),

    ...;
```

37. How to find the nth highest value in a column of a table?

Using the `OFFSET` clause. For example, to find the 6th highest value from a column, we would use the following syntax:

```
SELECT * FROM table_name

ORDER BY column_name DESC

LIMIT 1
```

```
OFFSET 5;
```

38. How to find the values in a text column of a table that start with a certain letter?

Using the LIKE operator in combination with the % and _ wildcards. For example, we need to find all surnames in a table that start with "A". The query is:

```
SELECT * FROM table_name

WHERE surname LIKE 'A_';
```

Here, we assume that a surname must contain at least two letters. Without this assumption (meaning that a surname can be just A), the query is as follows:

```
SELECT * FROM table_name

WHERE surname LIKE 'A%';
```

39. How to find the last id in a table?

Using the MAX() function. Otherwise, in many SQL versions, we can use the following syntax:

```
SELECT id

FROM table_name

ORDER BY id DESC

LIMIT 1;
```

or in Microsoft SQL Server:

```
SELECT TOP 1 id

FROM table_name

ORDER BY id DESC
```

40. How to select random rows from a table?

Using the RAND() function in combination with ORDER BY and LIMIT. In some SQL flavors, such as PostgreSQL, it's called RANDOM(). For example, the following code will return five random rows from a table in MySQL:

```
SELECT * FROM table_name


ORDER BY RAND()


LIMIT 5;
```

SQL JOINS:

1. What is a join?

A join is a SQ clause used to combine and retrieve records from two or multiple tables. SQL tables can be joined based on the relationship between the columns of those tables. Check out our **SQL joins** tutorial to know all the details about them.

2. What are the main types of joins?

There are six main types of joins:

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- FULL JOIN
- SELF JOIN
- CROSS JOIN

3. What is the difference between a LEFT JOIN and a RIGHT JOIN?

The LEFT JOIN includes all records from the left side and matched rows from the right table, whereas the RIGHT JOIN returns all rows from the right side and unmatched rows from the left table. Essentially, both joins will throw the same result if we exchange the table order, provided that there are only two tables involved.

4. Why are joins important in SQL management?

SQL joins are crucial in SQL management for multiple reasons, including:

- SQL JOINS are key methods to integrate multiple tables so they are easy to read and
- They provide an efficient and accessible way to access and combine information in your database.
- Using JOINS can reduce data usage and storage on the database.

5. What is an OUTER JOIN?

Outer joins are joins that return matched values and unmatched values from either or both tables. LEFT JOIN, RIGHT JOIN, AND FULL JOIN are considered outer joins.

6. What is an INNER JOIN?

An INNER JOIN returns only those records that satisfy a defined join condition in both (or all) tables. It's a default SQL join.

7. What is a CROSS JOIN?

A CROSS JOIN returns a paired combination of each row of the first table with each row of the second table. This join type is also known as cartesian join.

8. Is it possible to join a SQL table to itself?

Yes, this is normally done through a so-called self-join. A self-join is a type of JOIN used to compare rows within the same table. Unlike other SQL JOIN queries that join two or more tables, a self-join joins a table to itself.

9. What is the difference between FULL JOIN and CROSS JOIN?

A FULL JOIN returns all records from both tables. When the ON condition is not satisfied, it returns a NULL value. By contrast, a CROSS JOIN returns all possible combinations of all rows of both tables, resulting in a cartesian product between the two tables. This results in a larger table than the result of a FULL JOIN.

10. What is the purpose of using aliases in SQL JOINS?

As queries get more complex, names can get long and unwieldy. To help make things clearer, we can use aliases to assign new names to items in the query, including columns and tables. To give an alias to an object, we can use the AS clause.

10 Advanced SQL Joins Interview Questions

11. What is an EQUI JOIN?

An EQUI JOIN is a type of join operation in a database that combines rows from two or more tables based on a matching condition using the equality operator (=). It is used to retrieve data where values in specified columns are equal.

An EQUI JOIN returns the same results as an INNER JOIN with a different syntax, as shown in the following example:

```
SELECT COURSES.name, TECHNOLOGIES.name

FROM COURSES, TECHNOLOGIES

WHERE COURSES.technology_id = TECHNOLOGIES.technology_id;

SELECT COURSES.name, TECHNOLOGIES.name

FROM COURSES

INNER JOIN TECHNOLOGIES

ON COURSES.technology_id = TECHNOLOGIES.technology_id;
```

12. What is the difference between the ON and USING clauses in a join?

You will usually use the ON keyword to specify the common columns in the two tables to make the join. When the columns used to join are called equally in both tables, the USING clause can be used as a shorthand.

For example, if the tables COURSES and TECHNOLOGIES have a common column named 'technology_id', you can use the following query

```
SELECT * FROM COURSES

JOIN TECHNOLOGIES

USING (technology_id);
```

13. What is a NATURAL JOIN?

A NATURAL JOIN is used to create a JOIN based on common columns in two tables. Common columns are columns that have the same name in both tables.

Drawing on the example in the previous question, we could write the same query as follows:

```
SELECT * FROM COURSES

NATURAL JOIN TECHNOLOGIES
```

14. What is the difference between the JOIN clause and the UNION clause?

The JOIN clause is used to combine data into new columns based on the common columns of two or more tables.

By contrast, the UNION clause is used to combine data into new rows, based on the result of two or more SELECT statements. In other words, UNION is used to concatenate datasets row-wise. To prevent errors, every SELECT statement within UNION must have the same number of columns, and the columns must also have similar data types.

15. What is a semi-join?

Semi Join queries are generally executed in the form of subqueries where rows are picked up only from the first (left) table with respect to a condition (or a set of conditions) that is matched in the second table. Unlike regular joins, which include the matching rows from both tables, a semi-join only includes columns from the left table in the result.

16. What is an anti-join?

Anti-joins, also known as anti-semi-joins, are the exact opposite of semi-joins. In Anti Join, rows are picked up from the first table with respect to a condition (or a set of conditions) that is not matched in the second table.

17. What is the difference between the INTERSECT clause and an INNER JOIN?

INNER JOIN blends data from multiple tables, creating one comprehensive result comprising those rows or records that satisfy a defined join condition in both tables.

In contrast, INTERSECT focuses on common rows between SELECT statements. INNER JOIN relies on a shared column or field to connect tables, while INTERSECT works based on the structure of SELECT statements.

INNER JOIN often returns a larger dataset that combines information from different tables, while INTERSECT produces a smaller dataset featuring only shared rows.

18. What performance considerations should be taken when using CROSS JOINS?

Since CROSS JOIN returns a paired combination of each row of the first table with each row of the second table, this can result in an extremely large table, especially if the joining tables are already large. Thus, be aware when using CROSS JOINS, as they have a high potential to consume considerable resources and trigger performance issues.

19. What do you understand by conditional JOIN?

Conditional joins are a powerful technique for combining data from multiple tables based on specific conditions, allowing users to create more dynamic and flexible queries. Conditional joins help database administrators define custom queries that can include additional statements, including aggregation functions, comparison operators, and logical operators.

20. What is the difference between the WHERE and ON clauses in SQL JOINS?

The purpose of the ON clause is to specify the join conditions, in other words, to define how the tables should be joined. Specifically, you define how the records should be matched.

In contrast, the WHERE clause is used to specify the filtering conditions, that is, to define which rows should be kept in the result set. A JOIN that includes a filtering condition can be considered a conditional JOIN.