References: https://www.toptal.com/sql/interview-questions

Queries:

https://www.complexsql.com/complex-sql-queries-examples-with-answers/

https://www.techbeamers.com/sql-query-questions-answers-for-practice/

https://bytescout.com/blog/20-important-sql-queries.html

Basic:

1. SELECT and WHERE for filtering and selection
2. COUNT, SUM, MAX, GROUP BY, HAVING for aggregating data
3. DISTINCT, COUNT DISTINCT for producing useful distinct lists and distinct aggregates
4. OUTER (e.g. LEFT) and INNER JOIN when/where to use them
5. Working with time and strings (like expressions, basic manipulations like substring and date conversion)
6. Subqueries
7. UNION and UNION ALL.

Intermediate:

1. DML/DDL/DCL concepts
2. Handling NULLs creatively (e.g. with COALESCE)
3. Subqueries and the impact of subqueries on efficiency of the query
4. Basic query tuning (not necessarily optimization)
5. CAST variables and data types
7. CASE statements
8. Temporary tables
9. Self joins
10. Window functions like PARTITION, LEAD, LAG, NTILE
11. UDFs (user defined functions)
12. Use of indexes in querying to make operations faster.

Advanced:

1. Common Table Expressions (CTEs) and recursive CTEs

2. Dynamic SQL generation

3. Query optimization

4. Materialized views

5. Clustered index

6. Cursors

7. Minimizing network, I/O

8. Profilers and execution plans

SQL > Python https://www.kaggle.com/dorianlazar/pandas-equivalent-of-10-useful-sql-queries

1. What is SQL and Why SQL?

SQL is a Structured Query language.

It is used for storing, manipulating and retrieving data stored in a relational database.

SQL is widely popular because it offers various advantages −

· Allows users to **create and drop databases** and tables.

· Allows users to create **view, stored procedure**, functions in a database.

· Allows users to **set permissions on tables, procedures and views**. This is done using Grant and Revoke commands.

2. What are different types of SQL

· DDL (Data Definition Language) – It allows you to perform various operations on the database such

CREATE, ALTER, DROP objects. CAD

· DML (Data Manipulation Language) – It allows you to access and manipulate data.

SELECT, **INSERT, UPDATE, DELETE** data from the database. SIUD

· DCL (Data Control Language) – It allows you to control access to the database.

GRANT, REVOKE access permissions. GR

· TCL (Transaction Control Language) commands are used to manage transactions in the database. These are used to manage the changes made by DML-statements. It also allows statements to be grouped together into logical transactions.

COMMIT: Commit command is used to permanently save any transaction into the database.

ROLLBACK: This command restores the database to the last committed state. It is also used with the savepoint command to jump to a savepoint in a transaction.

SAVEPOINT: Savepoint command is used to temporarily save a transaction so that you can roll back to that point whenever necessary.

3. DBMS and RDBMS

A database is a structured collection of data.

A **Database Management System** (**DBMS**) is a software application that interacts with the user, applications and the database itself to capture and analyze data.

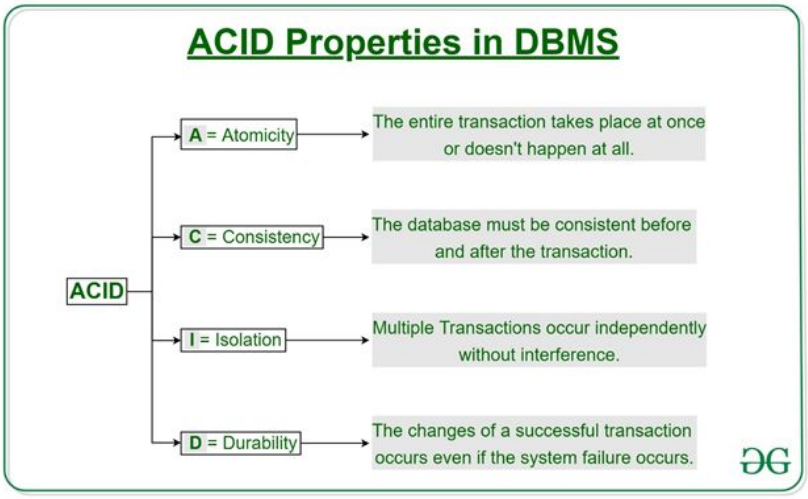
There are two types of DBMS:

· Relational Database Management System: The data is stored in relations (tables). Example – MySQL.

· Non-Relational DBMS: There is no concept of relations, tuples and attributes. E.g. – Mongo

Properties of a database:

Consistency: This means that integrity constraints must be maintained so that the database is consistent before and after the transaction.



4. What are Relationships and Entities ?

Entity: An entity can be a real-world object. For example, in a college database, students, professors, workers, departments, and projects can be referred to as entities. Each entity has some associated properties that provide it an identity.

Relationships: Relations or links between entities that have something to do with each other. For example - The employees table in a company's database can be associated with the salary table in the same database.

There are 4 types of relations

One to one
One to many and many to one
Many to many

Self: This is used when a table needs to define a relationship with itself.

5. Flat Files vs RDBMS

Flat file databases are typically plain text files that store one record per line, with record fields delimited by whitespace or a delimiting character.

6. Table vs Fields

A table is a set of data that are organized in a model with Columns and Rows. Columns can be categorized as vertical, and Rows are horizontal. A table has a specified number of columns called fields but can have any number of rows which is called record.

Table: Student Information

Field: Stu Id, Stu Name, Stu Marks

7. Char vs Varchar

CHAR

1. Used to store character string value of fixed length.
2. The maximum no. of characters the data type can hold is 8000 characters.
3. It's 50% **faster** than VARCHAR.
4. Uses static memory allocation.

VARCHAR

1. Used to store variable length alphanumeric data.
2. The maximum this data type can hold is up to 8000 characters.
3. It's **slower** than CHAR.
4. Uses dynamic memory allocation.

8. What's a Null value?

A NULL value in a table is a value in a field that appears to be **blank**, which means a field with a NULL value is a field with no value.

It is very important to understand that a **NULL value is different from a zero value or a field that contains spaces**. A field with a NULL value is the one that has been left blank during a record creation.

9. SQL Constraints

Constraints are the **rules enforced on data** columns on a table. This ensures the accuracy and reliability of the data in the database.

Constraints can either be **column level or table level**. Column level constraints are applied only to one column whereas table level constraints are applied to the entire table.

Following are some of the most commonly used constraints available in SQL −

- NOT NULL Constraint − Ensures that a column cannot have a NULL value.
- DEFAULT Constraint − Provides a default value for a column when none is specified.
- UNIQUE Constraint − Ensures that all the values in a column are different.
- PRIMARY Key − Uniquely identifies each row/record in a database table.
- FOREIGN Key − Uniquely identifies a row/record in any other database table.
- CHECK Constraint − The CHECK constraint ensures that all values in a column satisfy certain conditions.
- INDEX − Used to create and retrieve data from the database very quickly.

```
CREATE TABLE table_name (
    column1 datatype constraint,
    column2 datatype constraint,
    column3 datatype constraint,
    ....
);
```

10. Primary Key vs Foreign key

	Primary Key	Foreign Key
1	Primary Key Can Not Accept Null Values.	Foreign Key Can Accept Multiple Null Values.
2	Only One Primary Key in a Table.	More than One Foreign Key in a Table.
3	Primary Key Uniquely Identify a Record in the Table	Foreign Key is a Field in the Table that is Primary Key in Another Table
4	Primary Key is Clustered Index. Ex: `CREATE TABLE [country] ( [id] INT IDENTITY (1, 1) NOT NULL, [name] VARCHAR (50) NOT NULL, UNIQUE NONCLUSTERED ([name], CONSTRAINT [PK_country] PRIMARY KEY CLUSTERED ([id]) );`	Foreign Key is Non-Clustered Index. Ex: `CREATE TABLE [reg] ( [country] VARCHAR (50) NOT NULL, [name] VARCHAR (50) NOT NULL, CONSTRAINT [FK_reg_country] FOREIGN KEY ([name]) REFERENCES [dbo].[country] ([name]) );`

A clustered index defines the order in which data is physically stored in a table. Table data can be sorted in only one way, therefore, there can be only one clustered index per table.

A non-clustered index doesn't sort the physical data inside the table. In fact, a non-clustered index is stored at one place and table data is stored in another place.

11. Data Integrity

Data integrity refers to the **accuracy, consistency, and reliability** of data that is stored in the database. Both database designers and database developers are responsible for implementing data integrity within one or a set of related databases

There are four types of data integrity:

1. Row integrity

Row integrity refers to the requirement that all rows in a table must have a unique identifier that can be used to tell apart each record. This unique identifier is normally known as the Primary Key of the table. A Primary Key can be formed by a single column or a combination of multiple columns.

2. Column integrity

Column integrity refers to the requirement that data stored in a column must adhere to the same format and definition. This includes data type, data length, default value of data, range of possible values, whether duplicate values are allowed, or whether null values are allowed.

3. Referential integrity

RI requires that a foreign key must have a matching primary key or it must be null. This constraint is specified between two tables (parent and child); it maintains the correspondence between rows in these tables. It means the reference from a row in one table to another table must be valid.

12. Data normalization

Database Normalization is a technique of organizing the data in the database. It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables.

Normalization is used for mainly two purposes,

· Eliminating redundant(useless) data.

· Ensuring data dependencies make sense i.e. data is logically stored.

First Normal Form (1NF)

For a table to be in the First Normal Form, it should follow the following 4 rules:

1. It should only have **single(atomic) valued** columns. 1 data point or block should have only one value.
2. Values stored in a column should be of the **same domain**.
3. Make sure there's a **primary key.**
4. Remove groups of columns that have similar data (e.g. Customer 1, customer 2.....)

Second Normal Form (2NF)

For a table to be in the Second Normal Form,

1. It should be in the First Normal form.

2. All **non-key columns should be dependent on the primary key**.

Third Normal Form (3NF)

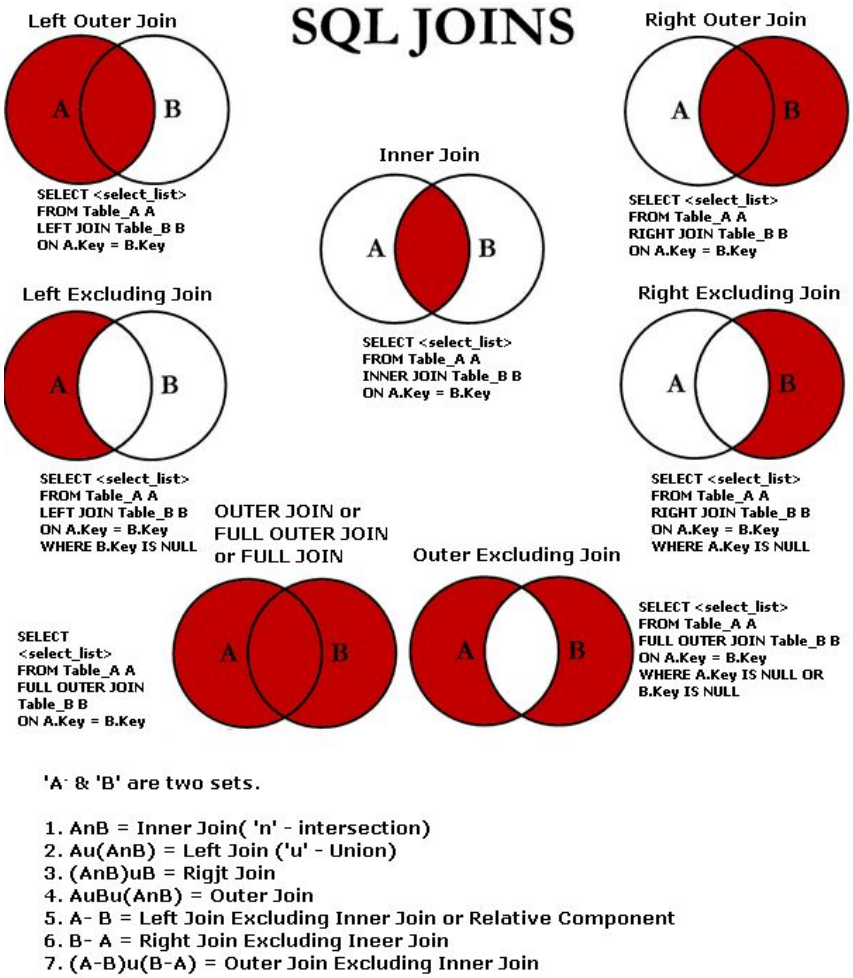
A table is said to be in the Third Normal Form when,

1. It is in the Second Normal form.
2. And, it doesn't have Transitive Dependency.

13. SQL Joins

The SQL **Joins** clause is used to combine records from two or more tables in a database. A JOIN is a means for combining fields from two tables by using values common to each.

- INNER JOIN − returns rows when there is a match in both tables.
- LEFT JOIN − returns all rows from the left table, even if there are no matches in the right table.
- RIGHT JOIN − returns all rows from the right table, even if there are no matches in the left table.
- FULL JOIN − returns rows when there is a match in one of the tables.
- SELF JOIN − is used to join a table to itself as if the table were two tables
- CARTESIAN − returns the Cartesian product of the sets of records from the two or more joined tables.



'A' & 'B' are two sets.

1. AnB = Inner Join('n' - intersection)
2. Au(AnB) = Left Join ('u' - Union)
3. (AnB)uB = Rigjt Join
4. AuBu(AnB) = Outer Join
5. A- B = Left Join Excluding Inner Join or Relative Component
6. B- A = Right Join Excluding Ineer Join
7. (A-B)u(B-A) = Outer Join Excluding Inner Join

14. Subqueries

A subquery is a query inside another query where a query is defined to retrieve data or information back from the database. In a subquery, the outer query is called as the main query whereas the inner query is called subquery.

Correlated and uncorrelated subqueries

A subquery can contain a reference to an object defined in a parent statement. This is called an **outer reference**. A subquery that contains an outer reference is called a **correlated subquery**. Correlated subqueries cannot be evaluated independently of the outer query because the subquery uses the values of the parent statement. That is, the subquery is performed for each row in the parent statement. Thus, results of the subquery are dependent upon the active row being evaluated in the parent statement.

For example, the subquery in the statement below returns a value dependent upon the active row in the Products table:

```
#Correlated subquery
SELECT Name, Description FROM Products
WHERE Quantity < 2 * (
   SELECT AVG( Quantity )
   FROM SalesOrderItems
   WHERE Products.ID = SalesOrderItems.ProductID );
```

A subquery that does not contain references to objects in a parent statement is called an **uncorrelated subquery**. In the example below, the subquery calculated exactly one value: the average quantity from the SalesOrderItems table. In evaluating the query, the database server computes this value once, and compares each value in the Quantity field of the Products table to it to determine whether to select the corresponding row.

```
#uncorrelated subquery
SELECT Name, Description
FROM Products
WHERE Quantity <  2 * (
   SELECT AVG( Quantity )
   FROM SalesOrderItems );
```

15. Joins vs subquery

Joins and subqueries are both used to combine data from different tables into a single result.
Joins are mostly used when the combining tables have common columns amongst them.
Subqueries can be used to return either a scalar (single) value or a row set; whereas, joins are used to return rows.

16. Index in SQL

Different types of indexes are:

- B-Tree index
- Bitmap index
- Clustered index
- Covering index
- Non-unique index
- Unique index

Indexes are used to speed-up query processes in SQL Server, resulting in high performance.

Without indexes, a DBMS has to go through all the records in the table in order to retrieve the desired results. This process is called table-scanning and is extremely slow. On the other hand, if you create indexes, the database goes to that index first and then retrieves the corresponding table records directly.

Clustered Index

A clustered index defines the order in which data is physically stored in a table. Table data can be sorted in only way, therefore, there can be only one clustered index per table.

In SQL Server, the **primary key** constraint automatically creates a clustered index on that particular column.

```
CREATE CLUSTERED INDEX IX_tblStudent_Gender_Score ON
student (gender ASC, total_score DESC)
```

The above script creates a clustered index named "IX_tblStudent_Gender_Score" on the student table. This index is created on the "gender" and "total_score" columns. An index that is created on more than one column is called a "**composite index**".

The above index first sorts all the records in the ascending order of the gender. If gender is the same for two or more records, the records are sorted in the descending order of the values in their "total_score" column. You can create a clustered index on a single column as well.

Note: If a table has no clustered index, its data rows are stored in an unordered structure called a **heap**.

Non clustered Index

A non-clustered index **doesn't sort the physical data inside the table**. In fact, a non-clustered index is stored at one place and table data is stored in another place. This is similar to a textbook where the book content is located in one place and the index is located in another. This allows for more than one non-clustered index per table.

It is important to mention here that inside the table the data will be sorted by a clustered index. However, inside the non-clustered index, data is stored in the specified order. The index contains column values on which the index is created and the address of the record that the column value belongs to.

Both clustered and nonclustered indexes can be unique. This means no two rows can have the same value for the index key.

Ordered Indexing is of two types −

- Dense Index
- Sparse Index

In a dense index, there is an index record for every search key value (pk column or any such other column) in the database. This makes searching faster but requires more space to store index records itself.

In sparse index, index records are not created for every search key. An index record here contains a search key and an actual pointer to the data on the disk.

Clustered indexes could be used as **dense or sparse** but one at a time. **If the key attribute is unique it will be dense or else sparse.**

They need **less memory and are faster.**

Unclustered indexes are dense.

They need **more memory and slower**

https://www.youtube.com/watch?v=lg8S2s_yTh4

17. Sets in SQL

SQL supports few Set operations which can be performed on the table data. These are used to get meaningful results from data stored in the table, under different special conditions.

UNION is used to combine the results of two or more `SELECT` statements. However, it will **eliminate duplicate** rows from its result set. In case of union, number of columns and datatype must be the same in both the tables, on which UNION operation is being applied.

```
SELECT * FROM First
UNION
SELECT * FROM Second;
```

UNION ALL

This operation is similar to Union. But it also shows the duplicate rows.

```
SELECT * FROM  First
UNION ALL
SELECT * FROM Second;
```

INTERSECT

Intersect operation is used to combine two SELECT statements, but it only returns the records which are common from both SELECT statements. In case of **Intersect** the number of columns and datatype must be the same.

```
SELECT * FROM First
INTERSECT
SELECT * FROM Second;
```

MINUS

The Minus operation combines results of two SELECT statements and returns only those in the final result, which belongs to the first set of the result.

```
SELECT * FROM First
MINUS
SELECT * FROM Second;
```

18. View vs Table

A view is a virtual table. A view consists of rows and columns just like a table.
If data is changing in the underlying table, the same **change is reflected in the view**. A view can be built on top of a single table or multiple tables. It can also be built on top of another view

Views offer the following advantages:

1. Ease of use: A view hides the complexity of the database tables from end users. Essentially, we can think of views as a layer of abstraction on top of the database tables.

2. Space savings: Views **take very little space** to store, since they do not store actual data.

3. Additional data security: Views can include only certain columns in the table so that only the non-sensitive columns are included and exposed to the end user. In addition, some databases allow views to have different security settings, thus hiding sensitive data from prying eyes.

19. Cursor

SQL works based on a set e.g., SELECT statement returns a set of rows which is called a result set. However, sometimes, you may want to process a data set on a **row by row basis**. This is where cursors come into play.
A database cursor is an object that enables traversal over the rows of a result set. It allows you to process individual rows returned by a query.

20. Window Function

https://learnsql.com/blog/sql-window-functions-examples/

https://drill.apache.org/docs/sql-window-functions-introduction/

Window functions operate on a **set of rows** and **return a single value for each row** from the underlying query. The term window describes the set of rows on which the function operates. A window function uses values from the rows in a window to calculate the returned values.When you use a window function in a query, define the window using the OVER() clause. The OVER() clause (window definition) differentiates window functions from other analytical and reporting functions. A query can include multiple window functions with the same or different window definitions.

The OVER() clause has the following capabilities:

· Defines window partitions to form groups of rows. (PARTITION BY clause) – basically group by. You gotta specify which column to work on

· Orders rows within a partition. (ORDER BY clause)

Aggregate Window Functions

SUM(), MAX(), MIN(), AVG(). COUNT()

Ranking Window Functions

RANK(), DENSE_RANK(), ROW_NUMBER(), NTILE()

Value Window Functions

LAG(), LEAD(), FIRST_VALUE(), LAST_VALUE()

a. Rownumber

i. The row number ranking window function returns a unique sequential number for each row within the partition of the specified window.

ii. If the PARTITION BY clause is specified, the ranking row number will be reset for each partition.

iii. SELECT *, ROW_NUMBER () OVER (PARTITION BY class ORDER BY StudentScore DESC) as rowNum.

iv. USE – When you want **a unique number for each row**.

b. Rank

i. The RANK() ranking window function returns a unique rank number for each distinct row within the partition according to a specified column value, starting at 1 for the first row in each partition, with the same rank for duplicate values and leaving gaps between the ranks; this gap appears in the sequence after the duplicate values.

ii. In other words, the RANK() ranking window function behaves like the ROW_NUMBER() function except for the rows with equal values, where it will rank with the same rank ID and generate a gap after it.

iii. SELECT *, RANK() OVER(PARTITION BY class ORDER BY StudentScore DESC) as rnk.

iv. USE – When you want **to keep track of the total number of people involved in the ranking.**

c. Dense Rank

i. The DENSE_RANK() ranking window function is similar to the RANK() function by generating a unique rank number for each distinct row within the partition according to a specified column value, starting at 1 for the first row in each partition, ranking the rows with equal values with the same rank number, except that it does not skip any rank, leaving no gaps between the ranks.

ii. SELECT *, DENSE_RANK OVER(PARTITION BY class ORDER BY StudentScore DESC) as dnsRnk.

iii. USE – **When pure ranking is important**.

d. Ntile

i. The NTILE(N) ranking window function is used to distribute the rows in the rows set into a specified number of groups, providing each row in the row set with a unique group number, starting with the number 1 that shows the group this row belongs to, where N is a positive number, which defines the number of groups you need to distribute the rows set into.

Rank vs Dense_Rank

Both are used to retrieve ranked data. But Rank skips missed ranks dense rank does not.

Rank – 1 1 3 4 4 6

Dense Rank – 1 1 2 3 3 4

Let's say you want to know the top 25 salesmen and their sales in your company. Now here rank doesn't matter what matters is salesman performance. So, you use Rank.
But if you want to know the top 25 students in a class whom you are going to award a scholarship, here rank matters. So, you use dense_rank.

ROW_NUMBER : Returns a unique number for each row starting with 1. For rows that have duplicate values,numbers are arbitrarily assigned.

Lag and lead (eg. Find time to reach next stop)

The LAG function has the ability to fetch data from a previous row, while LEAD fetches data from a subsequent row.

The SQL **NTILE()** is a window function that allows you to break the result set into a specified number of approximately equal groups, or buckets. It assigns each group a bucket number starting from one.

ROWID	ROWNUM
1.ROWID is nothing but Physical memory allocation	1. ROWNUM is nothing but the sequence which is allocated to that data retreival bunch.
2.ROWID is permanant to that row which identifies the address of that row.	2. ROWNUM is tempararily allocated sequence to the rows.
3.ROWID is 16 digit Hexadecimal number which is uniquely identifies the rows.	3.ROWNUM is numeric sequence number allocated to that row temporarily.
4.ROWID returns PHYSICAL ADDRESS of that row.	4.ROWNUM returns the sequence number to that row.
5. ROWID is automatically generated unique id of a row and it is generated at the time of insertion of row.	5. ROWNUM is an dynamic value automatically retrieved along with select statement output.

21. Stored Procedures – Triggers

A stored procedure is a **user defined piece of code** written in the local version of PL/SQL, which may return a value (making it a function) that is invoked by calling it explicitly. A **trigger** is a **stored procedure** that runs automatically when various events happen (e.g. update, insert, delete). A stored procedure which calls by itself until it reaches some boundary condition. This recursive function or procedure helps programmers to use the same set of code any number of times.

Pros:
- they don't have to go through the **compile process** over and over again.
- offer some additional **security** aspects.

Cons: **slow speed**

22. Drop vs Truncate vs Delete

DELETE: (dml)

1. This command will **temporarily** delete the data from the database.
2. Before commit operation, if we roll back, we will get the data.
3. It won't re-initialize the table.
4. command: SQL>delete from <table_name>

5. It is a data manipulation language command.

use:

we can delete a single column from the table.

Delete can have conditions unlike truncate.

TRUNCATE: (ddl)

1. This command will **permanently** delete the data from the database. But the table is not deleted.
2. If we roll back also, we won't get the data.
3. It will re-initialize the table.
4. command: sql>truncate table <table_name>

use:

we can delete the entire table.

DROP: (ddl)

1. This command will completely **destroy the table** from the database.
2. command: sql>drop table <table_name>

use:

We can remove tables from the database.

23. What is Collation? different types of Collation Sensitivity?

Collation refers to a **set of rules** that determine how data is sorted and compared. Rules defining the correct character sequence are used to sort the character data. It incorporates options for specifying case-sensitivity, accent marks, kana character types and character width. Below are the different types of collation sensitivity:

Case sensitivity: A and a are treated differently.

Accent sensitivity: a and á are treated differently.

Kana sensitivity: Japanese kana characters Hiragana and Katakana are treated differently.

Width sensitivity: Same character represented in single-byte (half-width) and double-byte (full-width) are treated differently.

24. What are Multidimensional schemas?

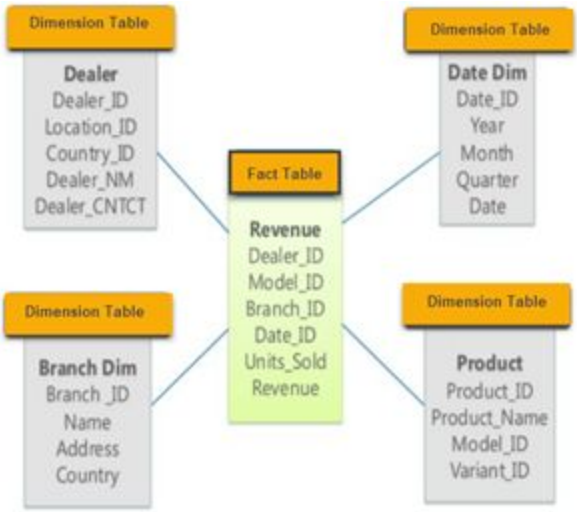
Multidimensional schema is especially designed to model data warehouse systems. The schemas are designed to address the unique needs of very large databases designed for the analytical purpose (OLAP).

Types of Data Warehouse Schema:

Following are 3 chief types of multidimensional schemas each having its unique advantages.

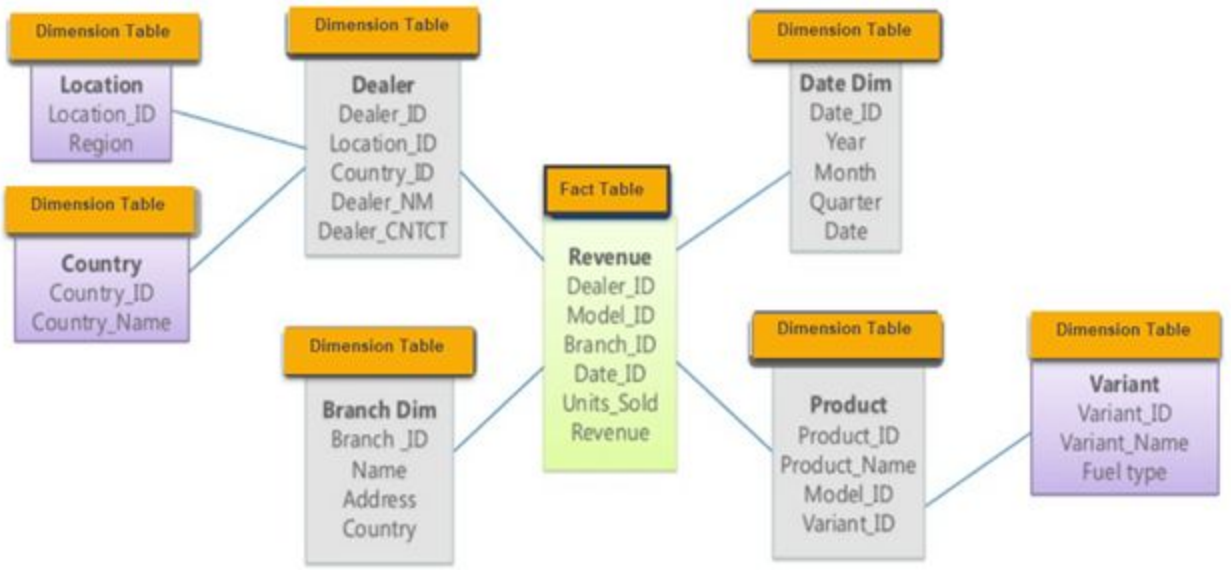
- Star Schema
- Snowflake Schema
- Galaxy Schema

What is a Star Schema?



In the **Star Schema**, the center of the star can have one fact table and a number of associated dimension tables. It is known as star schema as its structure resembles a star. The star schema is the simplest type of Data Warehouse schema. It is also known as Star Join Schema and is optimized for querying large data sets.

What is a Snowflake Schema?



A Snowflake Schema is an extension of a Star Schema, and it adds additional dimensions. It is called snowflake because its diagram resembles a Snowflake.

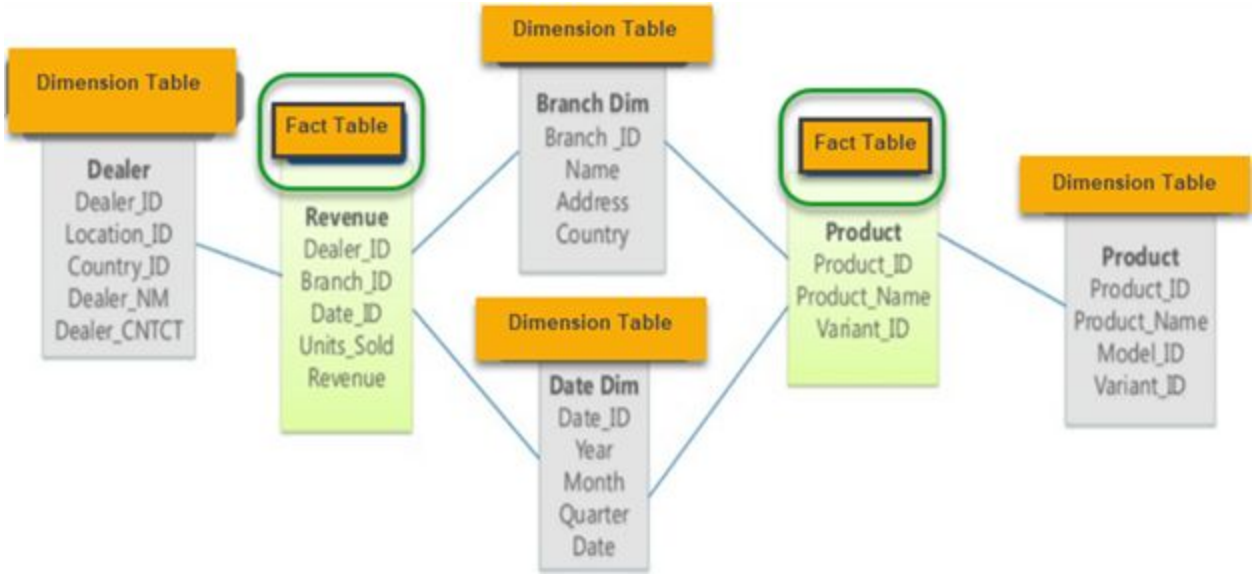
The **dimension tables are normalized which splits data into additional tables**. In the following example, Country is further normalized into an individual table.

Star Vs Snowflake Schema: Key Differences

Star Schema	SnowFlake Schema
Hierarchies for the dimensions are stored in the dimensional table.	Hierarchies are divided into separate tables.
It contains a fact table surrounded by dimension tables.	One fact table surrounded by dimension table which are in turn surrounded by dimension table
In a star schema, only a single join creates the relationship between the fact table and any dimension tables.	A snowflake schema requires many joins to fetch the data.

Simple DB Design.	Very Complex DB Design.
Denormalized Data structure and query also run faster.	Normalized Data Structure.
High level of Data redundancy	Very low-level data redundancy
Single Dimension table contains aggregated data.	Data Split into different Dimension Tables.
Cube processing is faster.	Cube processing might be slow because of the complex j
Offers higher performing queries using Star Join Query Optimization. Tables may be connected with multiple dimensions.	The Snow Flake Schema is represented by a centralized fact table which is unlikely connected with mu dimensions.

What is a Galaxy schema?



A Galaxy Schema contains **two** **fact tables that shares dimension tables**. It is also called Fact Constellation Schema. The schema is viewed as a collection of stars hence the name Galaxy Schema.

As you can see in above figure, there are two facts table

1. Revenue
2. Product.

In Galaxy schema shares dimensions are called Conformed Dimensions.

Summary:

· Multidimensional schema is especially designed to model data warehouse systems

· The star schema is the simplest type of Data Warehouse schema. It is known as star schema as its structure resembles a star.

· A Snowflake Schema is an extension of a Star Schema, and it adds additional dimensions. It is called snowflake because its diagram resembles a Snowflake.

· In a star schema, only a single join creates the relationship between the fact table and any dimension tables.

· Star schema contains a fact table surrounded by dimension tables.

· Snowflake schema is surrounded by dimension table which are in turn surrounded by dimension table

· A snowflake schema requires many joins to fetch the data.

· A Galaxy Schema contains two fact tables that shares dimension tables. It is also called Fact Constellation Schema.

· Star cluster schema contains attributes of Star schema and Snowflake schema.

25. Define Aggregate functions

Functions which operate against a collection of values and returning single value is called aggregate functions

SQL aggregate functions aggregate functions return a single value, calculated from values in a column.

Useful aggregate functions:

· AVG() - Returns the average value

- COUNT() - Returns the number of rows
- FIRST() - Returns the first value
- LAST() - Returns the last value
- MAX() - Returns the largest value
- MIN() - Returns the smallest value
- SUM() - Returns the sum

26. Define Scalar functions

Scalar function is dependent on the argument given and returns sole value.

SQL scalar functions: SQL scalar functions return a single value, based on the input value.

Useful scalar functions:

- UCASE() - Converts a field to uppercase
- LCASE() - Converts a field to lowercase
- MID() - Extract characters from a text field
- LEN() - Returns the length of a text field
- ROUND() - Rounds a numeric field to the number of decimals specified
- NOW() - Returns the current system date and time
- FORMAT() - Formats how a field is to be displayed

27. What restrictions can you apply when you are creating views?

Restrictions that are applied are:

- Only the current database can have views.
- You are not liable to change any computed value in any particular view.

- Integrity constraints decide the functionality of INSERT and DELETE.
- Full-text index definitions cannot be applied.
- Temporary views cannot be created.
- Temporary tables cannot contain views.
- No association with DEFAULT definitions.
- Triggers such as INSTEAD OF are associated with views.

28. Cliche Questions

How to remove duplicate rows from the table?

```
1. By using Group By and Count

SELECT id, COUNT(*) as Count
FROM Orders
GROUP BY  id
HAVING COUNT(*) > 1


2. Using row number

With CTE_Duplicates as
   (select empid,name , row_number() over(partition by empid,name
order by empid,name ) rownumber
   from EmpDup )
   delete from CTE_Duplicates where rownumber!=1
```

Q. If you are a SQL Developer, how can you delete duplicate records in a table with no primary key?

We can delete the duplicate records in the table with no primary key by using the SET ROWCOUNT command. It limits the number of records affected by a command. For example, if you have 2 duplicate rows, you would SET ROWCOUNT 1, execute DELETE command and then SET ROWCOUNT 0

Q. How to select random rows from a table?
Using a SAMPLE clause we can select random rows.

```
FROM table_name SAMPLE(10)
```

29. Join vs Merge vs Union

Join:
Combines data from multiple tables based on a certain matching condition.
Adds new columns.
Indexes are not destroyed.

Merge:
Same as Join but indexes are destroyed.

Union:
Combines results of two queries
Adds new rows
Queries should have same column

30. What do you know about databases?

- What database are for?
- Basic CRUD Operations
- SELECT queries with JOINs
- Normalization
- Basic Indexing
- Basic Constraints

31. Coalesce function

The COALESCE function in SQL returns the first non-NULL expression among its arguments.

Name	Business_Phone	Cell_Phone	Home_Phone
Jeff	531-2531	622-7813	565-9901
Laura	NULL	772-5588	312-4088
Peter	NULL	NULL	594-7477

and we want to find out the best way to contact each person according to the following rules:

1. If a person has a business phone, use the business phone number.

2. If a person does not have a business phone and has a cell phone, use the cell phone number.

3. If a person does not have a business phone, does not have a cell phone, and has a home phone, use the home phone number.

We can use the **COALESCE** function to achieve our goal:

```
SELECT Name, COALESCE (Business_Phone, Cell_Phone, Home_Phone) Contact_Phone
FROM Contact_Info;
```

Result:

```
Name Contact_Phone
Jeff       531-2531
Laura      772-5588
Peter      594-7477
```

NVL vs NVL2

Both the NVL(exp1, exp2) and NVL2(exp1, exp2, exp3) functions check the value exp1 to see if it is null.

With the NVL(exp1, exp2) function, if exp1 is not null, then the value of exp1 is returned; otherwise, the value of exp2 is returned, but case to the **same data type** as that of exp1.

With the NVL2(exp1, exp2, exp3) function, if exp1 is not null, then exp2 is returned; otherwise, the value of exp3 is returned.

32. WITH Clause

- The clause is used for defining a temporary relation such that the output of this temporary relation is available and is used by the query that is associated with the WITH clause.
- Queries that have an associated WITH clause can also be written using nested sub-queries but doing so add more complexity to read/debug the SQL query.

- The WITH clause is not supported by all database systems.

```
WITH table1 AS (
          SELECT *
          FROM web_events),

     table2 AS (
          SELECT *
          FROM accounts)


SELECT *
FROM table1
JOIN table2
ON table1.account_id = table2.id;
```

33. Linked List Data Structure

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers as shown in the below image:

In simple words, a linked list consists of nodes where each node contains a data field and a reference(link) to the next node in the list.

34. Difference between Exists and In operators

The Exist operators are used to check sub- query conditions if it satisfies all conditions then exist operators provide true results , however In operator is used for adding multiple or queries.

For exist operator:-

select column_name from table_name
where exist (select column_name from table_name where condition);

****

select productName from Products

where Exist(select productPrice from products
where procustID.products =ProductId.customers);
For In operator:-

Select column_name from table_name
where column_name In(value1, value2);

****
Select productName from products
where country In ('India', 'USA', 'Brazil');

35. What are some tips to improve the performance of SQL queries?

Ans. Optimizing SQL queries can bring substantial positive impact on the performance. It also depends on the level of RDBMS knowledge you have. Let's now go over some of the tips for tuning SQL queries.

Use: Views, Stored Procedures, Indexes

Avoid: Unnecessary Joins, Cursors, Distinct clause, Having clause

1. The UNION ALL clause responds faster than UNION. It doesn't look for duplicate rows whereas the UNION statement does that regardless of whether they exist or not.

2. It's a good practice to return the required column instead of all the columns of a table.

36. DBA Questions

What is Database White Box Testing?

Answer: Database White Box testing involves:

- Database Consistency and ACID properties
- Database triggers and logical views
- Decision Coverage, Condition Coverage, and Statement Coverage
- Database Tables, Data Model, and Database Schema

- Referential integrity rules

Q. What is Database Black Box Testing?

Database Black Box testing involves:

- Data Mapping
- Data stored and retrieved
- Use of Black Box testing techniques such as Equivalence Partitioning and Boundary Value Analysis (BVA)

Q Which TCP/IP port does SQL Server run?

Answer: By default SQL Server runs on port 1433.

37. Difference between alter and update ?

ALTER SQL command is a DDL (Data Definition Language) statement. ALTER is used to update the **structure** of the table in the database (like add, delete, modify the attributes of the tables in the database).

The SQL command is a DML (Data manipulation Language) statement. It is used to manipulate the **data** of any existing column. But can't change the table's definition.