**Interview Questions of Different Software Companies**

=> 2nd highest salary in the database

To find the second highest salary in the table, we will use the concept of subquery, which means that firstly, we will find the highest salary in the table and then we will nest that query to a subquery to find the second highest salary in SQL.

To find the highest salary in the table, write the following query.

**SELECT MAX(SALARY) FROM Employee**;

This will give you the output as the highest salary in the table.

Now, to find the second highest salary, we nest the above query into another query as written below.

**SELECT MAX(SALARY) FROM Employee**

**WHERE SALARY < (SELECT MAX(SALARY) FROM Employee);**

This query will give you the desired output, which is the second highest salary.

---------------------------------------------------------------------------------------------------------------------

=> Query optimization

Query optimization is of great importance for the performance of a relational database, especially for the execution of complex SQL statements. A query optimizer decides the best methods for implementing each query.

There is the various principle of Query Optimization are as follows

* **Understand how your database is executing your query**
* **Retrieve as little data as possible**
* **Store intermediate results**

There are various query optimization strategies are as follows

* **Use Index**
* **Aggregate Table**
* **Vertical Partitioning**
* **Horizontal Partitioning**
* **De-normalization**− The process of de-normalization combines multiple tables into a single table. This speeds up query implementation because fewer table joins are required.

------------------------------------------------------------------------------------------------------------------

=> Relationships

One-to-Many Relationship in DBMS

Many-to-One Relationship in DBMS

When each entry in one table may be linked to one or more records in the other table, this is known as a one-to-many relationship.

For example, it is possible for more than one student to work on a project. A group of five college students is given a project that they must accomplish in a month’s time. It shows a Many-to-One relationship.

## One-to-One Relationship in DBMS

When each record in one table is linked to only one record in the other table, this is known as a One-to-One relationship.

This type of relationship is, nonetheless, used for security reasons. We can easily save the **passport** id in the ‘Person’ database. However, we can create a separate table for the ‘Passport’ because the passport number is potentially sensitive information that should be kept from other users. For that reason, by creating a separate table, we can add an extra layer of protection by limiting access to just particular database users

## Many-to-Many Relationship in DBMS?

This type of relationship exists when each of the records of the first table can be associated with one or more records of the second table, as well as a single record of the second table may be related to one or more records of the first table.

A Many-to-Many relationship is formed **by two one-to-many relationships** that are connected by an ‘associate table’ or ‘linking table.’ By having fields that are the primary keys of the other two tables, the bridging table connects two tables

---------------------------------------------------------------------------------------------------------------------

=> how to resolve/handle many to many relationships.

* Many-to-many (m:n) relationships add complexity and confusion to your model and to the application development process.
* The key to resolve m:n relationships is to **separate the two entities and create two one-to-many (1:n) relationships between them with a third intersect entity**.
* **Junction table**.
  + When you need to establish a many-to-many relationship between two or more tables, the simplest way is to use a Junction Table. A Junction table in a database, also referred to as a **Bridge table or Associative Table**, bridges the tables together by referencing **the primary keys** of each data table

---------------------------------------------------------------------------------------------------------------------

# **=> ACID Properties in DBMS**

The **ACID** properties, provide a mechanism to ensure the correctness and consistency of a database in a way such that each transaction is a group of operations that acts as a single unit, produces consistent results, acts in isolation from other operations, and updates that it makes are durably stored

A [**transaction**](https://www.geeksforgeeks.org/sql-transactions/) is a single logical unit of work that accesses and possibly modifies the contents of a database. Transactions access data using read and write operations. In order to maintain consistency in a database, before and after the transaction, certain properties are followed. These are called **ACID** properties.

### **Atomicity**

### **Consistency**

### **Isolation**

### **Durability**

### **Atomicity:**

By this, we mean that either the entire transaction takes place at once or doesn’t happen at all. There is no midway i.e. transactions do not occur partially. Each transaction is considered as one unit and either runs to completion or is not executed at all. It involves the following two operations.   
—**Abort**: If a transaction aborts, changes made to the database are not visible.   
—**Commit**: If a transaction commits, changes made are visible.   
Atomicity is also known as the ‘All or nothing rule’.

### Consistency:

### This means that integrity constraints must be maintained so that the database is consistent before and after the transaction. It refers to the correctness of a database.

### Isolation:

This property ensures that multiple transactions can occur concurrently without leading to the inconsistency of the database state. Transactions occur independently without interference. Changes occurring in a particular transaction will not be visible to any other transaction until that particular change in that transaction is written to memory or has been committed. This property ensures that the execution of transactions concurrently will result in a state that is equivalent to a state achieved these were executed serially in some order.

### Durability:

This property ensures that once the transaction has completed execution, the updates and modifications to the database are stored in and written to disk and they persist even if a system failure occurs. These updates now become permanent and are stored in non-volatile memory. The effects of the transaction, thus, are never lost.

In Simple Example:

| **Property** | **Responsibility for maintaining properties** |
| --- | --- |
| Atomicity | Transaction Manager |
| Consistency | Application programmer |
| Isolation | Concurrency Control Manager |
| Durability | Recovery Manager |

---------------------------------------------------------------------------------------------------------------------

=> normalization 1st - 3rd form

This is a standardized method of structuring relational data that leads to a more efficient database. The efficiency comes from both **minimizing redundancy and ensuring consistency of data.**

**1NF**

The first normal form, otherwise known as 1NF, has the following rules:

* **Uniquely Identify each row (with a primary key)**
* **Multiple value columns must be separated, so that there is only one value in each column per row**
* **Consistent data type must be enforced for each column**

To identify each row, we need to have some sort of unique user ID for each row. To separate multiple values, we simply create another row for each additional item. By doing that we also will resolve our data consistency issue.

**2NF**

Second normal form, or 2NF, has only one rule:

* **Every non-key column in a table must depend on the value of the key**

What exactly does this mean? This means that every attribute must be describing only our user. So let’s go through each attribute and determine whether or not they belong on the table. If they don’t we have to create another table for them. i.e.

Name: ✓ This attribute describes our user, so we keep it

Item: ✗ This doesn’t describe our user, so we’ll move this to another table

Phone#: ✓ Describes our user

Subscription Level: ✓ Describes our user

Manufacturer: ✗ Doesn’t describe our user, this describes our item

Manufacturer Phone#: ✗ Doesn’t describe our user, describes item

Price: ✗ Doesn’t describe our user, describes item

**3NF**

Third normal form, or 3NF, again only has one rule:

* **Every non-key column must ONLY depend on the value of the key**

So this looks like a repeat of 2NF at first glance. However, this further qualifies that every column must ONLY describe the item.

-------------------------------------------------------------------------------------------------------------

**=> writing queries, aggregate functions, having clause, group by**

* Aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG() )
* The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.
* The GROUP BY statement is often used with aggregate functions to group the result-set by one or more columns.
* You can apply a HAVING clause only to columns that also appear in the GROUP BY clause or in an aggregate function.
* Example

SELECT COUNT(CustomerID),Country  
FROM Customers  
GROUP BY Country  
HAVING COUNT(CustomerID) > 5  
ORDER BY COUNT(CustomerID) DESC;

---------------------------------------------------------------------------------------------------------------------

=> **primary, foreign, composite keys ma difference with example**

## Primary Key

First, a primary key uniquely identifies each record in a database table. Any individual key that does this can be called a candidate key, but only one can be chosen by database engineers as a primary key.

## Composite Key

Next, there's the composite key, which is composed/combined of two or more attributes that collectively uniquely identify each record.

For example: the combination of house number and street might qualify as a composite key, given that the market listings are local. If so, then when someone searches using both the house number and the street, they should only get one single record returned.

## Foreign Key

Meanwhile, if there is a key in a linked table, such as a buyer’s table that references the primary key, that will be a foreign key.

While a primary key and a composite key might do the same things, the primary key will consist of one column, where the composite key will consist of two or more columns.

The relationship between a primary key and a foreign key is quite different. The key thing to understand here is that the primary key in one database table becomes a foreign key in another database table, and vice versa.

A foreign key in a database table is taken from some other table and applied in order to link database records back to that foreign table.

The foreign key in the database table where it resides is actually the primary key of the other table.

--------------------------------------------------------------------------------------------------------------------