**1.. 1. What is RDBMS?**

**Relational Data Base Management Systems (RDBMS) are database management systems that maintain data records and indices in tables. Relationships may be created and maintained across and among the data and tables. In a relational database, relationships between data items are expressed by means of tables. Interdependencies among these tables are expressed by data values rather than by pointers. This allows a high degree of data independence. An RDBMS has the capability to recombine the data items from different files, providing powerful tools for data usage.**

**2. What are the properties of the Relational tables?**

**Relational tables have six properties:**

1. **Values are atomic.**
2. **Column values are of the same kind.**
3. **Each row is unique.**
4. **The sequence of columns is insignificant.**
5. **The sequence of rows is insignificant.**
6. **Each column must have a unique name.**

**3. What is Normalization?**

**Database normalization is a data design and organization process applied to data structures based on rules that help building relational databases. In relational database design, the process of organizing data to minimize redundancy is called normalization. Normalization usually involves dividing a database into two or more tables and defining relationships between the tables. The objective is to isolate data so that additions, deletions, and modifications of a field can be made in just one table and then propagated through the rest of the database via the defined relationships.**

**4. What is De-normalization?**

**De-normalization is the process of attempting to optimize the performance of a database by adding redundant data. It is sometimes necessary because current DBMSs implement the relational model poorly. A true relational DBMS would allow for a fully normalized database at the logical level, while providing physical storage of data that is tuned for high performance. De-normalization is a technique to move from higher to lower normal forms of database modeling in order to speed up database access.**

**5. What are different normalization forms?**

1. **1NF: Eliminate Repeating Groups Make a separate table for each set of related attributes, and give each table a primary key. Each field contains at most one value from its attribute domain.**
2. **2NF: Eliminate Redundant Data If an attribute depends on only part of a multi-valued key, remove it to a separate table.**
3. **3NF: Eliminate Columns Not Dependent On Key If attributes do not contribute to a description of the key, remove them to a separate table. All attributes must be directly dependent on the primary key.**
4. **BCNF: Boyce-Codd Normal Form If there are non-trivial dependencies between candidate key attributes, separate them out into distinct tables.**
5. **4NF: Isolate Independent Multiple Relationships No table may contain two or more 1:n or n:m relationships that are not directly related.**
6. **5NF: Isolate Semantically Related Multiple Relationships There may be practical constrains on information that justify separating logically related many-to-many relationships.**
7. **ONF: Optimal Normal Form A model limited to only simple (elemental) facts, as expressed in Object Role Model notation.**
8. **DKNF: Domain-Key Normal Form A model free from all modification anomalies is said to be in DKNF.**

**Remember, these normalization guidelines are cumulative. For a database to be in 3NF, it must first fulfill all the criteria of a 2NF and 1NF database.**

**6. What is Stored Procedure?**

**A stored procedure is a named group of SQL statements that have been previously created and stored in the server database. Stored procedures accept input parameters so that a single procedure can be used over the network by several clients using different input data. And when the procedure is modified, all clients automatically get the new version. Stored procedures reduce network traffic and improve performance. Stored procedures can be used to help ensure the integrity of the database.  
  
e.g. sp\_helpdb, sp\_renamedb, sp\_depends etc.**

**7. What is Trigger?**

**A trigger is a SQL procedure that initiates an action when an event (INSERT, DELETE or UPDATE) occurs. Triggers are stored in and managed by the DBMS. Triggers are used to maintain the referential integrity of data by changing the data in a systematic fashion. A trigger cannot be called or executed; DBMS automatically fires the trigger as a result of a data modification to the associated table. Triggers can be viewed as similar to stored procedures in that both consist of procedural logic that is stored at the database level. Stored procedures, however, are not event-drive and are not attached to a specific table as triggers are. Stored procedures are explicitly executed by invoking a CALL to the procedure while triggers are implicitly executed. In addition, triggers can also execute stored procedures.**

**8. What is Nested Trigger?**

**A trigger can also contain INSERT, UPDATE and DELETE logic within itself, so when the trigger is fired because of data modification it can also cause another data modification, thereby firing another trigger. A trigger that contains data modification logic within itself is called a nested trigger.**

**9. What is View?**

**A simple view can be thought of as a subset of a table. It can be used for retrieving data, as well as updating or deleting rows. Rows updated or deleted in the view are updated or deleted in the table the view was created with. It should also be noted that as data in the original table changes, so does data in the view, as views are the way to look at part of the original table. The results of using a view are not permanently stored in the database. The data accessed through a view is actually constructed using standard T-SQL select command and can come from one to many different base tables or even other views.**

**10. What is Index?**

**An index is a physical structure containing pointers to the data. Indices are created in an existing table to locate rows more quickly and efficiently. It is possible to create an index on one or more columns of a table, and each index is given a name. The users cannot see the indexes; they are just used to speed up queries. Effective indexes are one of the best ways to improve performance in a database application. A table scan happens when there is no index available to help a query. In a table scan SQL Server examines every row in the table to satisfy the query results. Table scans are sometimes unavoidable, but on large tables, scans have a terrific impact on performance.**

**11. What is a Linked Server?**

**Linked Servers is a concept in SQL Server by which we can add other SQL Server to a Group and query both the SQL Server dbs using T-SQL Statements. With a linked server, you can create very clean, easy to follow, SQL statements that allow remote data to be retrieved, joined and combined with local data. Stored Procedure sp\_addlinkedserver, sp\_addlinkedsrvlogin will be used add new Linked Server.**

**12. What is Cursor?**

**Cursor is a database object used by applications to manipulate data in a set on a row-by- row basis, instead of the typical SQL commands that operate on all the rows in the set at one time.   
  
In order to work with a cursor we need to perform some steps in the following order:**

1. **Declare cursor**
2. **Open cursor**
3. **Fetch row from the cursor**
4. **Process fetched row**
5. **Close cursor**
6. **Deallocate cursor**

**13. What is Collation?**

**Collation refers to a set of rules that determine how data is sorted and compared. Character data is sorted using rules that define the correct character sequence, with options for specifying case sensitivity, accent marks, kana character types and character width.**

**14. What is Difference between Function and Stored Procedure?**

**UDF can be used in the SQL statements anywhere in the WHERE/HAVING/SELECT section where as Stored procedures cannot be. UDFs that return tables can be treated as another rowset. This can be used in JOINs with other tables. Inline UDF's can be thought of as views that take parameters and can be used in JOINs and other Rowset operations.**

**15. What is sub-query? Explain properties of sub-query?**

**Sub-queries are often referred to as sub-selects, as they allow a SELECT statement to be executed arbitrarily within the body of another SQL statement. A sub-query is executed by enclosing it in a set of parentheses. Sub-queries are generally used to return a single row as an atomic value, though they may be used to compare values against multiple rows with the IN keyword.**

**A subquery is a SELECT statement that is nested within another T-SQL statement. A subquery SELECT statement if executed independently of the T-SQL statement, in which it is nested, will return a resultset. Meaning a subquery SELECT statement can standalone and is not depended on the statement in which it is nested. A subquery SELECT statement can return any number of values, and can be found in, the column list of a SELECT statement, a FROM, GROUP BY, HAVING, and/or ORDER BY clauses of a T-SQL statement. A Subquery can also be used as a parameter to a function call. Basically a subquery can be used anywhere an expression can be used.**

**16. What are different Types of Join?**

1. **Cross Join A cross join that does not have a WHERE clause produces the Cartesian product of the tables involved in the join. The size of a Cartesian product result set is the number of rows in the first table multiplied by the number of rows in the second table. The common example is when company wants to combine each product with a pricing table to analyze each product at each price.**
2. **Inner Join A join that displays only the rows that have a match in both joined tables is known as inner Join. This is the default type of join in the Query and View Designer.**
3. **Outer Join A join that includes rows even if they do not have related rows in the joined table is an Outer Join. You can create three different outer join to specify the unmatched rows to be included:**
   1. **Left Outer Join: In Left Outer Join all rows in the first-named table i.e. "left" table, which appears leftmost in the JOIN clause are included. Unmatched rows in the right table do not appear.**
   2. **Right Outer Join: In Right Outer Join all rows in the second-named table i.e. "right" table, which appears rightmost in the JOIN clause are included. Unmatched rows in the left table are not included.**
   3. **Full Outer Join: In Full Outer Join all rows in all joined tables are included, whether they are matched or not.**
4. **Self Join This is a particular case when one table joins to itself, with one or two aliases to avoid confusion. A self join can be of any type, as long as the joined tables are the same. A self join is rather unique in that it involves a relationship with only one table. The common example is when company has a hierarchal reporting structure whereby one member of staff reports to another. Self Join can be Outer Join or Inner Join.**

**17. What are primary keys and foreign keys?**

**Primary keys are the unique identifiers for each row. They must contain unique values and cannot be null. Due to their importance in relational databases, Primary keys are the most fundamental of all keys and constraints. A table can have only one Primary key. Foreign keys are both a method of ensuring data integrity and a manifestation of the relationship between tables.**

**18. What is User Defined Functions? What kind of User-Defined Functions can be created?**

**User-Defined Functions allow defining its own T-SQL functions that can accept 0 or more parameters and return a single scalar data value or a table data type.  
Different Kinds of User-Defined Functions created are:**

1. **Scalar User-Defined Function A Scalar user-defined function returns one of the scalar data types. Text, ntext, image and timestamp data types are not supported. These are the type of user-defined functions that most developers are used to in other programming languages. You pass in 0 to many parameters and you get a return value.**
2. **Inline Table-Value User-Defined Function An Inline Table-Value user-defined function returns a table data type and is an exceptional alternative to a view as the user-defined function can pass parameters into a T-SQL select command and in essence provide us with a parameterized, non-updateable view of the underlying tables.**
3. **Multi-statement Table-Value User-Defined Function A Multi-Statement Table-Value user-defined function returns a table and is also an exceptional alternative to a view as the function can support multiple T-SQL statements to build the final result where the view is limited to a single SELECT statement. Also, the ability to pass parameters into a TSQL select command or a group of them gives us the capability to in essence create a parameterized, non-updateable view of the data in the underlying tables. Within the create function command you must define the table structure that is being returned. After creating this type of user-defined function, It can be used in the FROM clause of a T-SQL command unlike the behavior found when using a stored procedure which can also return record sets.**

**19. What is Identity?**

**Identity (or AutoNumber) is a column that automatically generates numeric values. A start and increment value can be set, but most DBA leave these at 1. A GUID column also generates numbers; the value of this cannot be controlled. Identity/GUID columns do not need to be indexed.**

**20. What is DataWarehousing?**

1. **Subject-oriented, meaning that the data in the database is organized so that all the data elements relating to the same real-world event or object are linked together;**
2. **Time-variant, meaning that the changes to the data in the database are tracked and recorded so that reports can be produced showing changes over time;**
3. **Non-volatile, meaning that data in the database is never over-written or deleted, once committed, the data is static, read-only, but retained for future reporting.**
4. **Integrated, meaning that the database contains data from most or all of an organization's operational applications, and that this data is made consistent.**

**21. What are the difference between clustered and a non-clustered index?**

1. **A clustered index is a special type of index that reorders the way records in the table are physically stored. Therefore table can have only one clustered index. The leaf nodes of a clustered index contain the data pages.**
2. **A non clustered index is a special type of index in which the logical order of the index does not match the physical stored order of the rows on disk. The leaf node of a non clustered index does not consist of the data pages. Instead, the leaf nodes contain index rows.**

**22. What are the different index configurations a table can have?**

**A table can have one of the following index configurations:**

1. **No indexes**
2. **A clustered index**
3. **A clustered index and many nonclustered indexes**
4. **A nonclustered index**
5. **Many nonclustered indexes**

**23. What are different types of Collation Sensitivity?**

1. **Case sensitivity - A and a, B and b, etc.**
2. **Accent sensitivity**
3. **Kana Sensitivity - When Japanese kana characters Hiragana and Katakana are treated differently, it is called Kana sensitive.**
4. **Width sensitivity - A single-byte character (half-width) and the same character represented as a double-byte character (full-width) are treated differently than it is width sensitive.**

**24. What is OLTP (Online Transaction Processing)?**

**In OLTP - online transaction processing systems relational database design use the discipline of data modeling and generally follow the Codd rules of data normalization in order to ensure absolute data integrity. Using these rules complex information is broken down into its most simple structures (a table) where all of the individual atomic level elements relate to each other and satisfy the normalization rules.**

**25. What's the difference between a primary key and a unique key?**

**Both primary key and unique key enforces uniqueness of the column on which they are defined. But by default primary key creates a clustered index on the column, where are unique creates a nonclustered index by default. Another major difference is that, primary key doesn't allow NULLs, but unique key allows one NULL only.**

**26. What is difference between DELETE and TRUNCATE commands?**

**Delete command removes the rows from a table based on the condition that we provide with a WHERE clause. Truncate will actually remove all the rows from a table and there will be no data in the table after we run the truncate command.**

1. **TRUNCATE:**
   1. **TRUNCATE is faster and uses fewer system and transaction log resources than DELETE.**
   2. **TRUNCATE removes the data by deallocating the data pages used to store the table's data, and only the page deallocations are recorded in the transaction log.**
   3. **TRUNCATE removes all rows from a table, but the table structure, its columns, constraints, indexes and so on, remains. The counter used by an identity for new rows is reset to the seed for the column.**
   4. **You cannot use TRUNCATE TABLE on a table referenced by a FOREIGN KEY constraint. Because TRUNCATE TABLE is not logged, it cannot activate a trigger.**
   5. **TRUNCATE cannot be rolled back.**
   6. **TRUNCATE is DDL Command.**
   7. **TRUNCATE Resets identity of the table**
2. **DELETE:**
   1. **DELETE removes rows one at a time and records an entry in the transaction log for each deleted row.**
   2. **If you want to retain the identity counter, use DELETE instead. If you want to remove table definition and its data, use the DROP TABLE statement.**
   3. **DELETE Can be used with or without a WHERE clause**
   4. **DELETE Activates Triggers.**
   5. **DELETE can be rolled back.**
   6. **DELETE is DML Command.**
   7. **DELETE does not reset identity of the table.**

**Note: DELETE and TRUNCATE both can be rolled back when surrounded by TRANSACTION if the current session is not closed. If TRUNCATE is written in Query Editor surrounded by TRANSACTION and if session is closed, it can not be rolled back but DELETE can be rolled back.**

**27. When is the use of UPDATE\_STATISTICS command?**

**This command is basically used when a large processing of data has occurred. If a large amount of deletions any modification or Bulk Copy into the tables has occurred, it has to update the indexes to take these changes into account. UPDATE\_STATISTICS updates the indexes on these tables accordingly.**

**28. What is the difference between a HAVING CLAUSE and a WHERE CLAUSE?**

**They specify a search condition for a group or an aggregate. But the difference is that HAVING can be used only with the SELECT statement. HAVING is typically used in a GROUP BY clause. When GROUP BY is not used, HAVING behaves like a WHERE clause. Having Clause is basically used only with the GROUP BY function in a query whereas WHERE Clause is applied to each row before they are part of the GROUP BY function in a query.**

**29. What are the properties and different Types of Sub-Queries?**

1. **Properties of Sub-Query**
   1. **A sub-query must be enclosed in the parenthesis.**
   2. **A sub-query must be put in the right hand of the comparison operator, and**
   3. **A sub-query cannot contain an ORDER-BY clause.**
   4. **A query can contain more than one sub-query.**
2. **Types of Sub-Query**
   1. **Single-row sub-query, where the sub-query returns only one row.**
   2. **Multiple-row sub-query, where the sub-query returns multiple rows,. and**
   3. **Multiple column sub-query, where the sub-query returns multiple columns**

**menu select SQL Server Configuration Properties, and choose the Security page.**

**30. Which command using Query Analyzer will give you the version of SQL server and operating system?**

**SELECT SERVERPROPERTY ('productversion'), SERVERPROPERTY ('productlevel'), SERVERPROPERTY ('edition').**

**31. What is SQL Server Agent?**

**SQL Server agent plays an important role in the day-to-day tasks of a database administrator (DBA). It is often overlooked as one of the main tools for SQL Server management. Its purpose is to ease the implementation of tasks for the DBA, with its full- function scheduling engine, which allows you to schedule your own jobs and scripts.**

**32. Name 3 ways to get an accurate count of the number of records in a table?**

**SELECT \* FROM table1   
SELECT COUNT(\*) FROM table1   
SELECT rows FROM sysindexes WHERE id = OBJECT\_ID(table1) AND indid < 2**

**33. What is the difference between a Local and a Global temporary table?**

1. **A local temporary table exists only for the duration of a connection or, if defined inside a compound statement, for the duration of the compound statement.**
2. **A global temporary table remains in the database permanently, but the rows exist only within a given connection. When connection is closed, the data in the global temporary table disappears. However, the table definition remains with the database for access when database is opened next time.**

**34. What is PRIMARY KEY?**

**A PRIMARY KEY constraint is a unique identifier for a row within a database table. Every table should have a primary key constraint to uniquely identify each row and only one primary key constraint can be created for each table. The primary key constraints are used to enforce entity integrity.**

**35. What is UNIQUE KEY constraint?**

**A UNIQUE constraint enforces the uniqueness of the values in a set of columns, so no duplicate values are entered. The unique key constraints are used to enforce entity integrity as the primary key constraints.**

**36. What is FOREIGN KEY?**

**A FOREIGN KEY constraint prevents any actions that would destroy links between tables with the corresponding data values. A foreign key in one table points to a primary key in another table. Foreign keys prevent actions that would leave rows with foreign key values when there are no primary keys with that value. The foreign key constraints are used to enforce referential integrity.**

**37. What is CHECK Constraint?**

**A CHECK constraint is used to limit the values that can be placed in a column. The check constraints are used to enforce domain integrity.**

**38. What is NOT NULL Constraint?**

**A NOT NULL constraint enforces that the column will not accept null values. The not null constraints are used to enforce domain integrity, as the check constraints.**

**39. How to get @@ERROR and @@ROWCOUNT at the same time?**

**If @@Rowcount is checked after Error checking statement then it will have 0 as the value of @@Recordcount as it would have been reset. And if @@Recordcount is checked before the error-checking statement then @@Error would get reset. To get @@error and @@rowcount at the same time do both in same statement and store them in local variable.**

**SELECT @RC = @@ROWCOUNT, @ER = @@ERROR**

**40. What are the advantages of using Stored Procedures?**

1. **Stored procedure can reduced network traffic and latency, boosting application performance.**
2. **Stored procedure execution plans can be reused, staying cached in SQL Server's memory, reducing server overhead.**
3. **Stored procedures help promote code reuse.**
4. **Stored procedures can encapsulate logic. You can change stored procedure code without affecting clients.**
5. **Stored procedures provide better security to your data.**

**41. What is a table called, if it has neither Cluster nor Non-cluster Index? What is it used for?**

**Unindexed table or Heap. Microsoft Press Books and Book on Line (BOL) refers it as Heap. A heap is a table that does not have a clustered index and, therefore, the pages are not linked by pointers. The IAM pages are the only structures that link the pages in a table together. Unindexed tables are good for fast storing of data. Many times it is better to drop all indexes from table and then do bulk of inserts and to restore those indexes after that.**

**42. Can SQL Servers linked to other servers like Oracle?**

**SQL Server can be linked to any server provided it has OLE-DB provider from Microsoft to allow a link. E.g. Oracle has an OLE-DB provider for oracle that Microsoft provides to add it as linked server to SQL Server group.**

**.**

**43. How to implement one-to-one, one-to-many and many-to-many relationships while designing tables?**

**One-to-One relationship can be implemented as a single table and rarely as two tables with primary and foreign key relationships. One-to-Many relationships are implemented by splitting the data into two tables with primary key and foreign key relationships. Many-to-Many relationships are implemented using a junction table with the keys from both the tables forming the composite primary key of the junction table.**

**44. What does SQL Stands for?**

* **SQL stands for Structured Query Language. For more info visit**[**What is SQL?**](http://www.katieandemil.com/345)

**45. Name all SQL Clauses in correct order and briefly describe them.**

* **The answer**[**is here**](http://www.katieandemil.com/346)

**46. What is the difference between WHERE and HAVING?**

* **Where filters rows. Having filters groups.**

**47. How can you combine two queries together which contain the same structure but cover different time period.**

* **Union or Union all. You can learn more in our**[**UNION Tutorial**](http://www.katieandemil.com/1099)**. I also recommend**[**SET Operators Tutorial**](http://www.katieandemil.com/1056)**.**

**48. Name all joins types you know and describe them.**

* [**Inner, Left Outer Joins**](http://www.katieandemil.com/405)**are most popular but we also have right other, cross join, self-join and in SQL Server Full Outher, Cross apply, Outer Apply. It is also worth knowing**[**database concepts**](http://www.katieandemil.com/78)**.**

**49. What is a subquery and a derived table?**

* **Subquery allows us to get single field values (or values) inside our main query (for instance where condition or select)**

**50. What is an aggregate function?**

* **Allows to perform a calculation on a group of rows. For more info visit our**[**Aggregate Functions Free Online Class**](http://www.katieandemil.com/1097)**.**

**51. Describe all aggregate functions you know.**

* [**Sum, min, max, count, avg**](http://www.katieandemil.com/1053)**are the most common but there are more.**

**52. What is the difference between deterministic and non-deterministic functions?**

* **Deterministic means that you can work out the result and non-deterministic is that you are not able to work out the precise result.**

**53. Describe Logical Query Processing.**

* **It is a sequence in which each clause (and what's inside) is executed. For more info Visit**[**Logical Query Processing Phases**](http://www.katieandemil.com/812)**.**

**54.What is the difference between JOIN and UNION?**

**SQL JOIN allows us to “lookup” records on other table based on the given conditions between two tables. For example, if we have the department ID of each employee, then we can use this department ID of the employee table to join with the department ID of department table to lookup department names.**

**UNION operation allows us to add 2 similar data sets to create resulting data set that contains all the data from the source data sets. Union does not require any condition for joining. For example, if you have 2 employee tables with same structure, you can UNION them to create one result set that will contain all the employees from both of the tables.**

**SELECT \* FROM EMP1**

**UNION**

**SELECT \* FROM EMP2;**

**55.What is the difference between UNION and UNION ALL?**

**UNION and UNION ALL both unify for add two structurally similar data sets, but UNION operation returns only the unique records from the resulting data set whereas UNION ALL will return all the rows, even if one or more rows are duplicated to each other.**

**In the following example, I am choosing exactly the same employee from the emp table and performing UNION and UNION ALL. Check the difference in the result.**

**SELECT \* FROM EMPLOYEE WHERE ID = 5**

**UNION ALL**

**SELECT \* FROM EMPLOYEE WHERE ID = 5**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **MGR\_ID** | **DEPT\_ID** | **NAME** | **SAL** | **DOJ** |
| **5.0** | **2.0** | **2.0** | **Anno** | **80.0** | **01-Feb-2012** |
| **5.0** | **2.0** | **2.0** | **Anno** | **80.0** | **01-Feb-2012** |

**SELECT \* FROM EMPLOYEE WHERE ID = 5**

**UNION**

**SELECT \* FROM EMPLOYEE WHERE ID = 5**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **MGR\_ID** | **DEPT\_ID** | **NAME** | **SAL** | **DOJ** |
| **5.0** | **2.0** | **2.0** | **Anno** | **80.0** | **01-Feb-2012** |

**56.What is the difference between WHERE clause and HAVING clause?**

**WHERE and HAVING both filters out records based on one or more conditions. The difference is, WHERE clause can only be applied on a static non-aggregated column whereas we will need to use HAVING for aggregated columns.**

**To understand this, consider this example.   
Suppose we want to see only those departments where department ID is greater than 3. There is no aggregation operation and the condition needs to be applied on a static field. We will use WHERE clause here:**

**SELECT \* FROM DEPT WHERE ID > 3**

|  |  |
| --- | --- |
| **ID** | **NAME** |
| **4** | **Sales** |
| **5** | **Logistics** |

**Next, suppose we want to see only those Departments where Average salary is greater than 80. Here the condition is associated with a non-static aggregated information which is “average of salary”. We will need to use HAVING clause here:**

**SELECT dept.name DEPARTMENT, avg(emp.sal) AVG\_SAL**

**FROM DEPT dept, EMPLOYEE emp**

**WHERE dept.id = emp.dept\_id (+)**

**GROUP BY dept.name**

**HAVING AVG(emp.sal) > 80**

|  |  |
| --- | --- |
| **DEPARTMENT** | **AVG\_SAL** |
| **Engineering** | **90** |

**As you see above, there is only one department (Engineering) where average salary of employees is greater than 80.**

**57.What is the difference among UNION, MINUS and INTERSECT?**

**UNION combines the results from 2 tables and eliminates duplicate records from the result set.**

**MINUS operator when used between 2 tables, gives us all the rows from the first table except the rows which are present in the second table.**

**INTERSECT operator returns us only the matching or common rows between 2 result sets.**

**To understand these operators, let’s see some examples. We will use two different queries to extract data from our emp table and then we will perform UNION, MINUS and INTERSECT operations on these two sets of data.**

**UNION**

**SELECT \* FROM EMPLOYEE WHERE ID = 5**

**UNION**

**SELECT \* FROM EMPLOYEE WHERE ID = 6**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **MGR\_ID** | **DEPT\_ID** | **NAME** | **SAL** | **DOJ** |
| **5** | **2** | **2.0** | **Anno** | **80.0** | **01-Feb-2012** |
| **6** | **2** | **2.0** | **Darl** | **80.0** | **11-Feb-2012** |

**MINUS**

**SELECT \* FROM EMPLOYEE**

**MINUS**

**SELECT \* FROM EMPLOYEE WHERE ID > 2**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **MGR\_ID** | **DEPT\_ID** | **NAME** | **SAL** | **DOJ** |
| **1** |  | **2** | **Hash** | **100.0** | **01-Jan-2012** |
| **2** | **1** | **2** | **Robo** | **100.0** | **01-Jan-2012** |

**INTERSECT**

**SELECT \* FROM EMPLOYEE WHERE ID IN (2, 3, 5)**

**INTERSECT**

**SELECT \* FROM EMPLOYEE WHERE ID IN (1, 2, 4, 5)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **MGR\_ID** | **DEPT\_ID** | **NAME** | **SAL** | **DOJ** |
| **5** | **2** | **2** | **Anno** | **80.0** | **01-Feb-2012** |
| **2** | **1** | **2** | **Robo** | **100.0** | **01-Jan-2012** |

**.**

**58.How to generate row number in SQL Without ROWNUM**

**Generating a row number – that is a running sequence of numbers for each row is not easy using plain SQL. In fact, the method I am going to show below is not very generic either. This method only works if there is at least one unique column in the table. This method will also work if there is no single unique column, but collection of columns that is unique. Anyway, here is the query:**

**SELECT name, sal, (SELECT COUNT(\*) FROM EMPLOYEE i WHERE o.name >= i.name) row\_num**

**FROM EMPLOYEE o**

**order by row\_num**

|  |  |  |
| --- | --- | --- |
| **NAME** | **SAL** | **ROW\_NUM** |
| **Anno** | **80** | **1** |
| **Bhuti** | **60** | **2** |
| **Darl** | **80** | **3** |
| **Hash** | **100** | **4** |
| **Inno** | **50** | **5** |
| **Meme** | **60** | **6** |
| **Pete** | **70** | **7** |
| **Privy** | **50** | **8** |
| **Robo** | **100** | **9** |
| **Tomiti** | **70** | **10** |

**59.How to select first 5 records from a table?**

**This question, often asked in many interviews, does not make any sense to me. The problem here is how do you define which record is first and which is second. Which record is retrieved first from the database is not deterministic. It depends on many uncontrollable factors such as how database works at that moment of execution etc. So the question should really be – “how to select any 5 records from the table?” But whatever it is, here is the solution:**

**In Oracle,**

**SELECT \***

**FROM EMP**

**WHERE ROWNUM <= 5;**

**In SQL Server,**

**SELECT TOP 5 \* FROM EMP;**

**Generic solution,**

**I believe a generic solution can be devised for this problem if and only if there exists at least one distinct column in the table. For example, in our EMP table ID is distinct. We can use that distinct column in the below way to come up with a generic solution of this question that does not require database specific functions such as ROWNUM, TOP etc.**

**SELECT name**

**FROM EMPLOYEE o**

**WHERE (SELECT count(\*) FROM EMPLOYEE i WHERE i.name < o.name) < 5**

|  |
| --- |
| **name** |
| **Inno** |
| **Anno** |
| **Darl** |
| **Meme** |
| **Bhuti** |

**I have taken “name” column in the above example since “name” is happened to be unique in this table. I could very well take ID column as well.**

**In this example, if the chosen column was not distinct, we would have got more than 5 records returned in our output.**

**Do you have a better solution to this problem? If yes, post your solution in the comment.**

**60.What is the difference between ROWNUM pseudo column and ROW\_NUMBER() function?**

**ROWNUM is a pseudo column present in Oracle database returned result set prior to ORDER BY being evaluated. So ORDER BY ROWNUM does not work.**

**ROW\_NUMBER() is an analytical function which is used in conjunction to OVER() clause wherein we can specify ORDER BY and also PARTITION BY columns.**

**Suppose if you want to generate the row numbers in the order of ascending employee salaries for example, ROWNUM will not work. But you may use ROW\_NUMBER() OVER() like shown below:**

**SELECT name, sal, row\_number() over(order by sal desc) rownum\_by\_sal**

**FROM EMPLOYEE o**

|  |  |  |
| --- | --- | --- |
| **name** | **Sal** | **ROWNUM\_BY\_SAL** |
| **Hash** | **100** | **1** |
| **Robo** | **100** | **2** |
| **Anno** | **80** | **3** |
| **Darl** | **80** | **4** |
| **Tomiti** | **70** | **5** |
| **Pete** | **70** | **6** |
| **Bhuti** | **60** | **7** |
| **Meme** | **60** | **8** |
| **Inno** | **50** | **9** |
| **Privy** | **50** | **10** |

**61.What are the differences among ROWNUM, RANK and DENSE\_RANK?**

**ROW\_NUMBER assigns contiguous, unique numbers from 1.. N to a result set.**

**RANK does not assign unique numbers—nor does it assign contiguous numbers. If two records tie for second place, no record will be assigned the 3rd rank as no one came in third, according to RANK. See below:**

**SELECT name, sal, rank() over(order by sal desc) rank\_by\_sal**

**FROM EMPLOYEE o**

|  |  |  |
| --- | --- | --- |
| **name** | **Sal** | **RANK\_BY\_SAL** |
| **Hash** | **100** | **1** |
| **Robo** | **100** | **1** |
| **Anno** | **80** | **3** |
| **Darl** | **80** | **3** |
| **Tomiti** | **70** | **5** |
| **Pete** | **70** | **5** |
| **Bhuti** | **60** | **7** |
| **Meme** | **60** | **7** |
| **Inno** | **50** | **9** |
| **Privy** | **50** | **9** |

**DENSE\_RANK, like RANK, does not assign unique numbers, but it does assign contiguous numbers. Even though two records tied for second place, there is a third-place record. See below:**

**SELECT name, sal, dense\_rank() over(order by sal desc) dense\_rank\_by\_sal**

**FROM EMPLOYEE o**

|  |  |  |
| --- | --- | --- |
| **name** | **Sal** | **DENSE\_RANK\_BY\_SAL** |
| **Hash** | **100** | **1** |
| **Robo** | **100** | **1** |
| **Anno** | **80** | **2** |
| **Darl** | **80** | **2** |
| **Tomiti** | **70** | **3** |
| **Pete** | **70** | **3** |