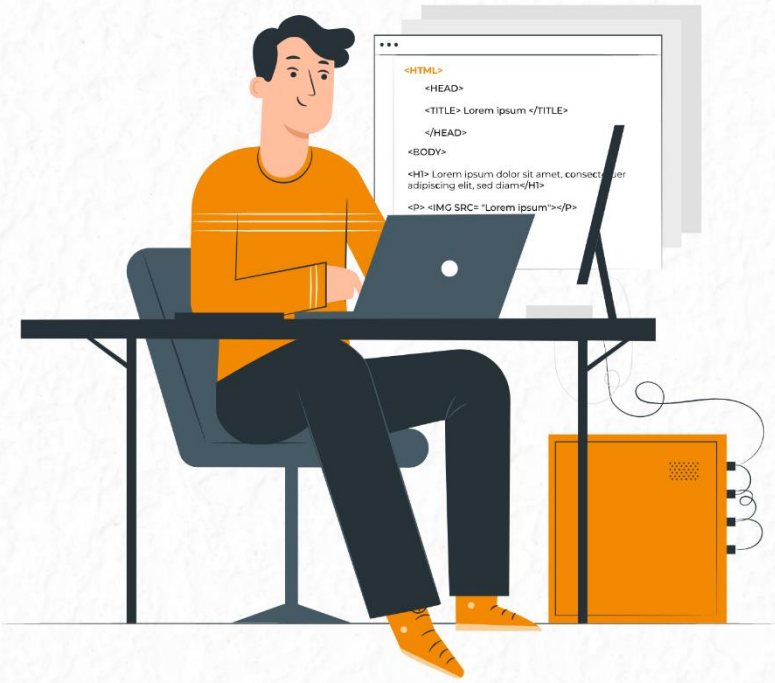
iNeuron



Python

Interview Questions

Q1. What is the use of package body statements while working on a complex database system with PL/SQL code?

The package body statement in PL/SQL is used to develop complex database systems with procedural logic. A package body is a collection of related functions and procedures whose code must be defined for the program to run properly, and it can also serve as a wrapper around several other elements, like any variables and type definitions it needs.

With this form of encapsulation, developers can better organize their programs into logical groupings based on what they do, making them easier to read while maintaining their performance when executing tasks.

Q2. What is a cursor variable, and how can it be used in PL/SQL?

A cursor variable is a pointer or reference to the memory address of an Oracle server-side cursor that enables stored procedures and functions in PL/SQL to access data from relational databases.

It can be used to retrieve rows from one or more database tables by using SQL queries within PL/SQL code. The returned result set is assigned into a record structure that remains accessible for the duration of the program execution.

Cursor Variable in PL/SQL

Q3. How do you check for duplicate records within an outer query in PL/SQL?

In PL/SQL, you can check for duplicate records within an outer query using the SELECT DISTINCT clause. The SELECT DISTINCT clause ensures that only the specified columns' distinct (unique) values are returned in a record set.

When combined with other clauses such as GROUP BY and ORDER BY, it can detect duplicate rows based on specific criteria or group them using various column combinations.

Additionally, aggregate functions such as COUNT, MAX and MIN can be used to calculate statistics on the returned individual records.

Q4. Describe the process of creating nested tables in a relational database management system?

Nested tables are collection objects in an Oracle database that allow storing and retrieving multiple records within one data structure. They can store complex data sets such as arrays, multidimensional collections, or entire tabular structures.

Creating nested tables typically involves creating an SQL table with columns defined to support the type and number of elements stored in each row, then inserting values using either Insert Statements or Select statements into that newly created nested table.

Q5. How does one create objects at the schema level using PL/SQL code?

In PL/SQL, one can create objects at the schema level by executing a CREATE statement. This needs to be followed by specifying the name and type of object that will be created within the database.
For example, consider a situation where PL/SQL code can be used to build a table in the Oracle Database. In that case, all required columns must be defined with data types and primary key constraints before any INSERT statements are issued against this newly created table.
Other objects, such as sequences or triggers, may also have various constraints associated with them; for instance, an AUTONOMOUS_TRANSACTION trigger will not fire while COMMIT occurs due to its PRAGMA clause settings that must first be set accordingly before execution of CREATE TRIGGER statement.

Q6. Explain what Cursor-based records mean and how can they be managed through PL/SQL commands?

Cursor-based records are the result sets that can be retrieved by performing a query using cursors in PL/SQL. A cursor is an object used to point to and process individual rows at a time from a returned record set when data retrieval is done either through standard SELECT statements or complex stored procedures and functions. To work with such cursor-based records,
You must first Declare your Cursor variable name along with all necessary parameters associated with the given SQL query;
Open said Variable once declared correctly - this allocates memory resources needed for processing the underlying dataset;
Fetch particular values out from opened variables using ensuing fetch commands;
Perform updates/deletions upon certain conditions (if any) depending upon your application's logic;
Close finally reallocating corresponding used database resources enabling other programs waiting in line.

Q7. Can EXECUTE keyword be used to execute anonymous blocks in PL/SQL?

Yes, the EXECUTE keyword can be used to execute anonymous blocks in PL/SQL. An anonymous block is a set of valid SQL and PL/SQL statements grouped together as one segment that does not have an assigned name.

They are typically typed directly into command-line interfaces such as Oracle SQL*Plus or used within application programming sources to interact with underlying relational databases without having actual subprogram definitions created for the same operations every time they need to be executed again later on.

Q8. What is the return type of a PL/SQL function? Explain each return type.

The return type of a PL/SQL function can be any valid data type such as NUMBER, VARCHAR2, BOOLEAN, and DATE.

The NUMBER type is used to hold numeric values such as integers, real numbers, and floating point numbers.
The VARCHAR2 data type stores character (or string) values of fixed or variable length up to 4 gigabytes.
BOOLEAN datatype holds a boolean "true" or "false." It can be represented by 0 for false and 1 for true value.
DATE types are used to store date and time information in the Oracle Database, which includes the year, month, day, hours, minutes, etc.; each DATE column occupies 7 bytes on disk and has maximum precision up to 9 decimal places.

Q9. How do you use the PL/SQL function return statement?

The return statement is used in a PL/SQL function to define the value of the data type declared in the RETURN clause and end execution of that particular function. In other words, it specifies what type of data should be returned by a given function when called.

For example, A simple mathematical expression may use multiple functions within one larger assembling program (e.g., parametric equation), with each component's values being taken from different variables or external sources.

The entire sub-program can then be compiled into one callable unit by using returning statements for each part which will read as follows: RETURN x + y, where x & y are two separate numerical parameters defined prior. Then this assembled program can, later on, be utilized like any other normal stand-alone utility!

Q10. Name two concatenation operators used in PL/SQL programming language.

The two concatenation operators used in PL/SQL programming language are the period (.) and double vertical bar (||) operators. The period operator is used to combine a column name with its table or view. In contrast, the double vertical bars combine strings together, typically within an expression or as part of a larger statement. For example, 'Hello' || 'World' will result in Hello World being printed out when executed on the database console.

Q10. Describe the control structures used in PL/SQL programming language.

PL/SQL control structure helps to design a program in an orderly manner. It provides the flexibility and functionality of large programs by allowing blocks of code to execute conditionally or repeatedly based on user input, conditions, etc. All programming languages have their implementation of these structures; in PL/SQL, there are three main categories: statements, loops, and branches.

Statements: A statement is used when no further decision-making needs to be done – it simply allows you to perform one action at a time (e.g., assign values).
Loops: Loops allow you to set up instructions that will repeat until some criteria are reached - for example, iterate through records returned from SQL query or looping over data defined within your block itself (using 'for' loops).
Branches: Branches let you specify different sets of instructions depending on certain factors – such as variables being equal/not-equal - if the result is true, then branch down this route; else, take another path. This gives great flexibility when designing complex functions inside your application that require multiple outcomes to fulfill specific results.

Q11. What are duplicate tables, and what purpose do they serve when working with databases?
Duplicate tables are used to store copies of data from another table. They can be particularly useful in database management systems where a single mistake in the original table could have disastrous effects if not corrected quickly.

Duplicate tables can provide an extra layer of protection by providing access to unburdened versions of the same information that allows for any changes made in one version without affecting another or even reversing any mistakes before they affect other users. This helps organizations avoid costly mistakes and reduces time spent troubleshooting them when they do happen.

Q12. How do triggers work in Oracle PL/SQL?

Triggers in Oracle PL/SQL are stored programs that fire when a specific event occurs, such as data modification. They are primarily used to enforce business rules and maintain database integrity by preventing unauthorized access or changes to data. Triggers can also be used for auditing purposes, and they typically take the form of SQL statements that execute when a table is changed. The trigger usually contains an IF statement that executes commands based on whether certain conditions have been met; this allows triggers to selectively process only those rows that need additional processing beyond what happens during normal inserts or updates on tables.

Q13. What are the different types of joins in Oracle PL/SQL?

There are four different types of joins in Oracle PL/SQL:

Oracle Simple JOIN: This type of join is also known as an inner or equi-join. It displays all the records from both tables where there is a match with the specified columns in each table.
Oracle LEFT JOIN: The left join returns all rows that exist in the left side table and optionally any matching rows from the right side table if it exists. If there are no matches, NULL values will be returned for the right column's results.
Oracle RIGHT JOIN: The Right outer join performs similarly to Left Outer join but reverses roles between the two tables, returning information from the second table (right) and matching info from the first one (left). Unmatched entries containing Nulls will be included for Right Table data fields only, while on the Left side - the usual query output and result set will appear.
Oracle FULL JOIN: A full outer join combines elements of a left outer join and a right outer join into one statement, thus giving us access to every row existing within either joined relation regardless of if there was/was not a matched value present when comparing keys across participating relations

Q14.What is the purpose of using a DUAL table in Oracle PL/SQL?

The purpose of the DUAL table is to provide developers with a single-row, single-column table used in SELECT statements to validate your code. It can be particularly useful when entering complex SQL queries that use Oracle functions and operators such as NVL(), DECODE() or CONCAT(). It can also be used to test SQL statements without creating and populating a table.

Q15. What is the syntax for declarations of cursors in PL/SQL?

The syntax for declaring of cursors in PL/SQL is: CURSOR <cursor_name> [(<parameter list>)] IS SELECT statement;
Example: CURSOR student_cursor IS SELECT * FROM students;

Q16. How can we use single quotes in Oracle PL/SQL?

We can use single quotes in Oracle PL/SQL by escaping them using a backslash (\).
For example: SELECT * FROM mytable WHERE name = 'John\'s Place';

Q17. Name and explain some implicit cursor attributes associated with cursors in PL/SQL.
The implicit cursor attributes associated with cursors in PL/SQL are:

%FOUND: This attribute is a Boolean variable that returns true if the last fetch from the cursor was successful and false otherwise. If no rows were found, an exception would be raised instead of returning false for this value.
%NOTFOUND: This attribute is also a Boolean variable that returns true when nothing has been returned by the previous fetch statement or query execution but can return NULL when no data operation has occurred yet on that particular context area (i.e., after opening the cursor).
%ISOPEN: A built-in function that checks whether or not your defined SQL Cursor is open, and if so, it's True; else, False.
%ROWCOUNT: It keeps track of how many records have been fetched from within its loop to return multiple rows as part of its results set as per user requirement.
%BULK_ROWCOUNT: This attribute returns the total number of rows processed in a bulk operation, such as when using FORALL or BULK COLLECT operations to insert multiple rows at once into an Oracle table with PL/SQL code (i.e., "bulk DML").

Q18. What are some of the cursor operations that can be performed on a created using PL/SQL?
The most common cursor operations that can be performed on a PL/SQL cursor include:

Opening the cursor.
Fetching rows from the result set one at a time or in bulk amounts.
Closing the cursor when finished processing.
Bind Variables for passing parameters to query execution.
Explicit Cursor Attributes such as %ISOPEN, %ROWCOUNT and %NOTFOUND that take values based on query resultset information

Q19. What are some of the list of parameters used to define a cursor inside an anonymous block or stored procedure?

The list of parameters used to define a cursor inside an anonymous block or stored procedure includes:

CURSOR_NAME: This is the name you assign to the cursor.
SELECT_STATEMENT: This is the SQL query that forms your result set.
ORDER BY: An optional clause that can be used to sort your data in ascending/descending order based on one or more columns.
FOR UPDATE [OF column_list]: Specify this option if you want other users (in another session) to be blocked from updating rows included in the resultset while it's being processed by the current user's open cursor statement.
OPEN, CLOSE and FETCH statements for fetching records from the database into program variables.
NO_PARSE: Specifying this clause will force the engine to parse your SQL query once instead of parsing it every time when fetched.
USING Clause: This is an optional clause used to pass values from program variables in a PL/SQL block while opening the cursor or during fetching records.

Q20. What kinds of variables should you declare when using integers in range functions like BETWEEN etc.?

When using range functions like BETWEEN in PL/SQL, you should declare two variables: one for the lower bound of the range and another for the upper bound. Both variables should be declared as integers. For example:

```
DECLARE
lower_bound INTEGER;
upper_bound INTEGER;
BEGIN

SELECT * FROM customers WHERE customer_number BETWEEN
:lower_bound AND :upper_Bound; END;
```