Job Description

1. Ability to write and understand the complex PL/SQL objects like procedures, packages, functions, triggers, cursors etc.
2. Strong knowledge to debug the code (anonymous block/stored procedure/function) etc.
3. Ability to write the procedures / functions which will be consumed by any APIs in real time scenarios.
4. Excellent understanding of performance tuning and optimization also able to understand the explain plan / AWR / DDR report.
5. Proven experience in to work with Oracle Database, SQL and PL/SQL.
6. Strong knowledge of database management and design with SDLC understanding.
7. Proficient to understand the basis DDL, DML, TCL, languages and its relevant use.
8. Fundamental of SQL and PL/SQL should be very clear.
9. Ability to understand the business need and write the code to meet the desired result set.
10. Familiar to work in any version management tool like SVN, Git-lab etc.

Typical Interview

Initial 10 Minutes: Discuss Work Experience on SQL

(In this segment, you will mostly keep these two questions only.)

1. **Question:** Can you briefly describe your experience working with SQL, specifically with Oracle DB?
   Answer: In my previous role, I have extensively worked with Oracle DB for over three years. My responsibilities included writing complex queries, optimizing existing ones, and developing PL/SQL procedures and functions. I also created and managed triggers for data validation and consistency, and was involved in schema design and data modeling.

2. **Question:** Can you share a challenging SQL problem you encountered and how you resolved it?
   Answer: One challenging issue was optimizing a report generation query that was running slow due to large data volumes. After analyzing the execution plan, I identified inefficient joins and lack of proper indexing. I resolved it by restructuring the query, adding necessary indexes, and using hints to optimize execution paths. This reduced the query execution time by 70%.

Next 20 Minutes: Verbal Questions

(Target to ask 5 questions in this section. Compelete answer of 3 questions and 2 partials is good enough to proceed the candidate.)

1. **Question:** Explain the difference between a procedure and a function in PL/SQL.
   Answer: In PL/SQL, a procedure is a subprogram that performs a specific action and does not return a value, while a function is a subprogram that returns a single value. Procedures can return multiple values through OUT parameters, whereas functions can be used in SQL statements and are generally designed to return a single computed value.

2. **Question:** How do you create a trigger in Oracle DB to enforce data integrity?
   Answer: To create a trigger in Oracle DB, you define it using the `CREATE TRIGGER` statement, specifying the timing (BEFORE or AFTER) and the event (INSERT, UPDATE, DELETE). For example:

```
CREATE TRIGGER check_salary
BEFORE INSERT OR UPDATE OF salary
ON employees
FOR EACH ROW
BEGIN
  IF :NEW.salary < 0 THEN
    RAISE_APPLICATION_ERROR(-20001, 'Salary cannot be negative');
  END IF;
END;
```

This trigger ensures that the salary cannot be negative.

3. **Question:** Can you explain DDL, DML, and TCL with examples?
   Answer:
   - **DDL (Data Definition Language):** Commands that define the database schema. Examples include `CREATE TABLE`, `ALTER TABLE`, `DROP TABLE`.
   - **DML (Data Manipulation Language):** Commands that manipulate data. Examples include `SELECT`, `INSERT`, `UPDATE`, `DELETE`.
   - **TCL (Transaction Control Language):** Commands that manage transactions. Examples include `COMMIT`, `ROLLBACK`, `SAVEPOINT`.

4. **Question:** How do you write a PL/SQL block to calculate the factorial of a number?
   Answer:

```
DECLARE
  v_num NUMBER := 5;  -- Example number
  v_fact NUMBER := 1;
BEGIN
  FOR i IN 1..v_num LOOP
    v_fact := v_fact * i;
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('Factorial of ' || v_num || ' is ' || v_fact);
END;
```

5. **Question:** Describe how you would design a stored procedure that an API can call to retrieve employee details by department.
   Answer:

```
CREATE OR REPLACE PROCEDURE get_employees_by_department(
  p_department_id IN employees.department_id%TYPE,
  p_cursor OUT SYS_REFCURSOR
) IS
BEGIN
  OPEN p_cursor FOR
  SELECT * FROM employees
  WHERE department_id = p_department_id;
END;
```

Last 30 Minutes: Scenario-Based Design and Queries

(He must solve the first design problem, must be able to write the query asked, should be able to suggest improvements on his design if any required or be able to justify if his/her DB/Query is the best and should be able to either write a PL/SQL procedure or the block.)

Scenario: Design a simple database for a library system with tables for books, authors, and loans.

Questions:

1. **Design the database schema for the library system.**
   Answer:

```sql
CREATE TABLE authors (
  author_id NUMBER PRIMARY KEY,
  name VARCHAR2(100)
);

CREATE TABLE books (
  book_id NUMBER PRIMARY KEY,
  title VARCHAR2(200),
  author_id NUMBER,
  FOREIGN KEY (author_id) REFERENCES authors(author_id)
);

CREATE TABLE loans (
  loan_id NUMBER PRIMARY KEY,
  book_id NUMBER,
  loan_date DATE,
  return_date DATE,
  FOREIGN KEY (book_id) REFERENCES books(book_id)
);
```

2. **Write a query to find all books currently on loan.**
   Answer:

```sql
SELECT b.title, l.loan_date
FROM books b
JOIN loans l ON b.book_id = l.book_id
WHERE l.return_date IS NULL;
```

3. **Optimize the design to improve query performance for frequently loaned books.**
   Answer:
   - Add indexes on `loan_date` and `return_date` in the `loans` table.
   - Create a composite index on `books(author_id, title)` to speed up joins and searches.

4. **Write a stored procedure to add a new loan record.**
   Answer:

```sql
CREATE OR REPLACE PROCEDURE add_loan(
  p_book_id IN loans.book_id%TYPE,
  p_loan_date IN loans.loan_date%TYPE
) IS
BEGIN
  INSERT INTO loans (book_id, loan_date)
  VALUES (p_book_id, p_loan_date);
END;
```

5. **Write a PL/SQL block to find and display all overdue books (assuming a loan period of 30 days).**
   Answer:

```sql
DECLARE
  CURSOR overdue_books IS
    SELECT b.title, l.loan_date
    FROM books b
    JOIN loans l ON b.book_id = l.book_id
    WHERE l.return_date IS NULL AND l.loan_date < SYSDATE - 30;
BEGIN
  FOR book IN overdue_books LOOP
    DBMS_OUTPUT.PUT_LINE('Overdue Book: ' || book.title || ' Loaned on: ' || book.loan_date);
  END LOOP;
END;
```

QUESTION BANK

Verbal Questions

Sure! Here are 20 verbal questions you can ask during the second segment of the interview to assess the candidate's knowledge in SQL, PL/SQL, and database design.

1. **Question:** Can you explain the difference between a primary key and a unique key in Oracle?
   Answer: A primary key uniquely identifies each record in a table and does not allow NULL values. A table can have only one primary key. A unique key also ensures uniqueness of the values in one or more columns, but it can contain NULL values and a table can have multiple unique keys.

2. **Question:** What are the different types of joins in SQL, and when would you use each type?
   Answer: The different types of joins include:
   - **INNER JOIN:** Returns rows when there is a match in both tables.
   - **LEFT JOIN (LEFT OUTER JOIN):** Returns all rows from the left table and matched rows from the right table.
   - **RIGHT JOIN (RIGHT OUTER JOIN):** Returns all rows from the right table and matched rows from the left table.
   - **FULL JOIN (FULL OUTER JOIN):** Returns rows when there is a match in one of the tables.
   - **CROSS JOIN:** Returns the Cartesian product of the two tables.
     Each type is used based on the requirement for matching records between tables.

3. **Question:** How would you handle exceptions in PL/SQL?
   Answer: In PL/SQL, exceptions are handled using the `EXCEPTION` block. It allows you to catch and handle runtime errors. Here is an example:

   ```
   BEGIN
     -- code that might cause an exception
   EXCEPTION
     WHEN NO_DATA_FOUND THEN
       -- handle no data found exception
     WHEN OTHERS THEN
       -- handle all other exceptions
   END;
   ```

4. **Question:** What is the purpose of an index in a database, and how does it improve performance?
   Answer: An index is used to speed up the retrieval of rows by using a pointer. It improves performance by reducing the amount of data that needs to be scanned to find the desired rows. Indexes are particularly useful for columns that are frequently used in search conditions and join conditions.

5. **Question:** Can you explain the concept of a cursor in PL/SQL?
   Answer: A cursor in PL/SQL is a pointer to a context area where a query result is stored. Cursors allow row-by-row processing of query results. There are two types of cursors: implicit cursors (automatically created by Oracle for single-row SELECT statements) and explicit cursors (defined by the programmer for multi-row SELECT statements).

6. **Question:** Describe how you would use a PL/SQL package and its advantages.
   Answer: A PL/SQL package is a collection of related procedures, functions, variables, and other PL/SQL constructs grouped together. Advantages include modularity, easier maintenance, improved performance (because the entire package is loaded into memory), and encapsulation of related operations.

7. **Question:** What is the difference between DELETE and TRUNCATE in SQL?
   Answer: `DELETE` removes rows one at a time and logs each deletion, allowing for a rollback. `TRUNCATE` removes all rows from a table without logging individual row deletions and cannot be rolled back. `TRUNCATE` is faster but less flexible than `DELETE`.

8. **Question:** How would you optimize a slow-running query in Oracle?
   Answer: To optimize a slow-running query, you can:
   - Analyze and update statistics with `ANALYZE` or `DBMS_STATS`.

- Use indexes appropriately.
- Rewrite the query to avoid full table scans.
- Use execution plans to identify bottlenecks.
- Consider partitioning large tables.
- Optimize joins and subqueries.

9. **Question:** Explain the use of the `MERGE` statement in SQL.
   Answer: The `MERGE` statement is used to perform `INSERT`, `UPDATE`, or `DELETE` operations in a single statement based on the conditions specified. It is useful for synchronizing two tables.

10. **Question:** What is a materialized view, and how does it differ from a regular view?
    Answer: A materialized view is a database object that contains the results of a query and can be refreshed periodically. Unlike regular views, which are virtual and run the query each time they are accessed, materialized views store the query result physically, improving performance for complex queries.

11. **Question:** Describe the concept of a foreign key and its importance in relational databases.
    Answer: A foreign key is a column or a set of columns in one table that refers to the primary key in another table. It enforces referential integrity by ensuring that the value in the foreign key column corresponds to a valid primary key value in the referenced table.

12. **Question:** How do you create and use a sequence in Oracle?
    Answer: A sequence is a database object that generates unique numeric values. It is created using the `CREATE SEQUENCE` statement. Example:

    ```
    CREATE SEQUENCE emp_seq
    START WITH 1
    INCREMENT BY 1;
    ```

    You can use the sequence to generate values:

    ```
    INSERT INTO employees (id, name) VALUES (emp_seq.NEXTVAL, 'John Doe');
    ```

13. **Question:** What is a composite key, and when would you use it?
    Answer: A composite key is a primary key that consists of two or more columns. It is used when a single column is not sufficient to uniquely identify a row. For example, in a table storing order items, a combination of order ID and product ID could serve as a composite key.

14. **Question:** Explain the difference between `IN` and `EXISTS` in SQL.
    Answer: `IN` is used to check if a value exists within a list of values or a subquery result set. `EXISTS` is used to check if a subquery returns any rows. `EXISTS` is typically faster for subqueries that return large result sets because it stops processing as soon as it finds a match.

15. **Question:** How would you create a PL/SQL function to calculate the total salary of all employees in a department?
    Answer:

    ```
    CREATE OR REPLACE FUNCTION total_salary_by_dept(
      p_dept_id IN employees.department_id%TYPE
    ) RETURN NUMBER IS
      v_total_salary NUMBER;
    BEGIN
      SELECT SUM(salary)
      INTO v_total_salary
      FROM employees
      WHERE department_id = p_dept_id;
      RETURN v_total_salary;
    END;
    ```

16. **Question:** What are the benefits of using stored procedures in a database?
    Answer: Stored procedures offer several benefits, including improved performance (since they are precompiled), enhanced security (by restricting direct access to data), modularity and reusability of code, and easier maintenance.

17. **Question:** How do you implement error logging in PL/SQL?
    Answer: Error logging in PL/SQL can be implemented by creating an error logging table and using the `EXCEPTION` block to capture and insert error details into this table. For example:

```
CREATE TABLE error_log (
  error_time TIMESTAMP,
  error_code NUMBER,
  error_message VARCHAR2(4000)
);

BEGIN
  -- code that might cause an exception
EXCEPTION
  WHEN OTHERS THEN
    INSERT INTO error_log (error_time, error_code, error_message)
    VALUES (SYSTIMESTAMP, SQLCODE, SQLERRM);
END;
```

18. **Question:** Can you explain the difference between a view and a table in Oracle?
    Answer: A table is a database object that stores data in rows and columns. A view is a virtual table created by a query that retrieves data from one or more tables. Views do not store data themselves but provide a way to simplify complex queries and enhance security by restricting data access.

19. **Question:** Describe how you would use a temporary table in Oracle.
    Answer: Temporary tables are used to store intermediate results and are session-specific or transaction-specific. They are created using the `CREATE GLOBAL TEMPORARY TABLE` statement. Example:

```
CREATE GLOBAL TEMPORARY TABLE temp_employees (
  id NUMBER,
  name VARCHAR2(50)
) ON COMMIT DELETE ROWS;
```

20. **Question:** How do you handle NULL values in SQL, and what are some functions used to manage them?
    Answer: NULL values represent missing or unknown data. You can handle NULL values using functions like `NVL` (to replace NULL with a specified value), `COALESCE` (to return the first non-NULL value from a list), and `NULLIF` (to return NULL if two expressions are equal). Example:

```
SELECT NVL(commission_pct, 0) FROM employees;
```

21. **Question:** Explain the concept of data normalization and its importance.
    Answer: Data normalization is the process of organizing data in a database to reduce redundancy and improve data integrity. The primary goals are to eliminate duplicate data, ensure data dependencies make sense, and make the database more efficient to query and maintain. Normalization involves dividing large tables into smaller ones and defining relationships between them.

22. **Question:** What is a cursor in SQL and how do you use it in PL/SQL?
    Answer: A cursor is a database object used to retrieve a set of rows one at a time. In PL/SQL, cursors allow row-by-row processing of query results. You declare a cursor, open it, fetch rows, and then close it. Example:

```
DECLARE
  CURSOR emp_cursor IS SELECT id, name FROM employees;
  v_id employees.id%TYPE;
  v_name employees.name%TYPE;
BEGIN
  OPEN emp_cursor;
  LOOP
    FETCH emp_cursor INTO v_id, v_name;
    EXIT WHEN emp_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('ID: ' || v_id || ' Name: ' || v_name);
  END LOOP;
  CLOSE emp_cursor;
END;
```

23. **Question:** How do you use the `CASE` statement in SQL?
    Answer: The `CASE` statement allows for conditional logic in SQL queries. It evaluates conditions and returns a value when the first condition is met. Example:

```sql
SELECT name,
       CASE
         WHEN salary < 3000 THEN 'Low'
         WHEN salary BETWEEN 3000 AND 7000 THEN 'Medium'
         ELSE 'High'
       END AS salary_range
FROM employees;
```

24. **Question:** What is a common table expression (CTE) and how do you use it?
    Answer: A CTE is a temporary result set defined within the execution scope of a single `SELECT`, `INSERT`, `UPDATE`, or `DELETE` statement. CTEs are defined using the `WITH` keyword. Example:

```sql
WITH employee_sales AS (
  SELECT employee_id, SUM(sales) AS total_sales
  FROM sales
  GROUP BY employee_id
)
SELECT e.name, es.total_sales
FROM employees e
JOIN employee_sales es ON e.id = es.employee_id;
```

25. **Question:** Explain the difference between `INNER JOIN` and `OUTER JOIN`.
    Answer: `INNER JOIN` returns rows when there is a match in both tables being joined. `OUTER JOIN` includes rows even if there is no match. `LEFT OUTER JOIN` includes all rows from the left table and matched rows from the right, while `RIGHT OUTER JOIN` includes all rows from the right table and matched rows from the left. `FULL OUTER JOIN` includes rows when there is a match in one of the tables.

26. **Question:** What is partitioning in a database, and why is it useful?
    Answer: Partitioning divides a large table or index into smaller, more manageable pieces called partitions. Each partition can be managed and accessed independently, improving performance and manageability. Partitioning is useful for enhancing query performance, especially in large databases, and for improving maintenance tasks like backups and archiving.

27. **Question:** How do you use the `ROWNUM` pseudo-column in Oracle?
    Answer: `ROWNUM` assigns a unique number to each row returned by a query. It can be used to limit the number of rows returned or for pagination. Example:

```sql
SELECT *
FROM (SELECT employee_id, name FROM employees ORDER BY salary DESC)
WHERE ROWNUM <= 10;
```

28. **Question:** Describe what a synonym is in Oracle and its purpose.
    Answer: A synonym is an alias for a database object such as a table, view, sequence, or another synonym. Synonyms simplify SQL statements by allowing you to reference an object by a simpler name and can be used to provide location transparency for remote objects. Example:

```sql
CREATE SYNONYM emp_syn FOR employees;
SELECT * FROM emp_syn;
```

29. **Question:** What are the differences between `UNION` and `UNION ALL`?
    Answer: `UNION` combines the result sets of two or more SELECT statements and removes duplicate rows. `UNION ALL` also combines the result sets but includes all duplicates. `UNION` is slower due to the duplicate elimination process.

30. **Question:** How do you handle transactions in PL/SQL?
    Answer: Transactions in PL/SQL are managed using `COMMIT`, `ROLLBACK`, and `SAVEPOINT`. `COMMIT` saves changes made during the transaction.

`ROLLBACK` undoes changes. `SAVEPOINT` sets a point within a transaction to which you can roll back. Example:

```
BEGIN
  INSERT INTO employees (id, name) VALUES (1, 'John');
  SAVEPOINT sp1;
  INSERT INTO employees (id, name) VALUES (2, 'Doe');
  ROLLBACK TO sp1;
  COMMIT;
END;
```

31. **Question:** What is a dual table in Oracle and how is it used?
    Answer: The `DUAL` table is a special one-row, one-column table provided by Oracle for selecting a value without needing an actual table. It is commonly used for selecting system functions or pseudo-columns. Example:

```
SELECT SYSDATE FROM DUAL;
```

32. **Question:** Explain the difference between `NVL` and `COALESCE` functions.
    Answer: `NVL` takes two arguments and returns the first if it is not NULL; otherwise, it returns the second. `COALESCE` can take multiple arguments and returns the first non-NULL value among them. `COALESCE` is more versatile and follows the ANSI SQL standard.

33. **Question:** What are database triggers, and what are some common use cases?
    Answer: Triggers are PL/SQL blocks executed automatically in response to certain events on a table or view, such as `INSERT`, `UPDATE`, or `DELETE`. Common use cases include enforcing complex business rules, auditing changes, and maintaining derived values.

34. **Question:** How do you use the `DECODE` function in Oracle?
    Answer: The `DECODE` function allows you to perform IF-THEN-ELSE logic within a query. It compares an expression to each search value and returns the corresponding result. Example:

```
SELECT name,
       DECODE(department_id, 10, 'HR', 20, 'Sales', 30, 'Finance', 'Other')
FROM employees;
```

35. **Question:** Describe the purpose of the `EXPLAIN PLAN` statement.
    Answer: `EXPLAIN PLAN` shows the execution plan of a SQL statement, detailing how Oracle will execute the statement, including join methods, access paths, and the order of operations. It helps in analyzing and optimizing query performance.

36. **Question:** What is a pragma in PL/SQL, and can you give an example?
    Answer: A pragma is a compiler directive that provides additional information to the compiler. An example is `PRAGMA EXCEPTION_INIT` to associate a user-defined exception with an Oracle error number. Example:

```
DECLARE
  my_exception EXCEPTION;
  PRAGMA EXCEPTION_INIT(my_exception, -20001);
BEGIN
  -- code that might cause an exception
EXCEPTION
  WHEN my_exception THEN
    DBMS_OUTPUT.PUT_LINE('Custom error message');
END;
```

37. **Question:** What is the `CONNECT BY` clause in Oracle, and when would you use it?
    Answer: The `CONNECT BY` clause is used for hierarchical queries to retrieve data in a tree-like structure. It is typically used with `START WITH` and `LEVEL` pseudo-columns. Example:

```
SELECT employee_id, manager_id, LEVEL
FROM employees
START WITH manager_id IS NULL
CONNECT BY PRIOR employee_id = manager_id;
```

38. **Question:** How do you use the `CASE` expression in PL/SQL?
    Answer: The `CASE` expression allows you to perform conditional logic in PL/SQL code. Example:

```
DECLARE
  v_grade CHAR(1) := 'B';
  v_desc VARCHAR2(20);
BEGIN
  v_desc := CASE v_grade
              WHEN 'A' THEN 'Excellent'
              WHEN 'B' THEN 'Good'
              WHEN 'C' THEN 'Fair'
              ELSE 'Poor'
            END;
  DBMS_OUTPUT.PUT_LINE('Grade description: ' || v_desc);
END;
```

39. **Question:** Explain the use of `FOR UPDATE` and `WHERE CURRENT OF` in PL/SQL cursors.
    Answer: `FOR UPDATE` locks the selected rows for update, preventing other transactions from modifying them. `WHERE CURRENT OF` references the current row in the cursor, allowing you to update or delete it. Example:

```
DECLARE
  CURSOR emp_cursor IS SELECT id, salary FROM employees FOR UPDATE;
BEGIN
  OPEN emp_cursor;
  LOOP
    FETCH emp_cursor INTO v_id, v_salary;
    EXIT WHEN emp_cursor%NOTFOUND;
    UPDATE employees SET salary = salary * 1.1 WHERE CURRENT OF emp_cursor;
  END LOOP;
  CLOSE emp_cursor;
END;
```

40.

Question: How do you use the `INSERT ALL` statement in Oracle?
Answer: The `INSERT ALL` statement allows you to insert multiple rows into one or more tables in a single statement. Example:

```
sql     INSERT ALL       INTO employees (id, name) VALUES (1, 'John Doe')       INTO employees (id, name) VALUES (2, 'Jane Smith')
```

41. **Question:** What is the purpose of the `DBMS_OUTPUT` package in PL/SQL?
    Answer: The `DBMS_OUTPUT` package is used to display output from PL/SQL blocks, procedures, and functions. It is commonly used for debugging purposes. Example:

```
BEGIN
  DBMS_OUTPUT.PUT_LINE('Hello, World!');
END;
```

42. **Question:** Explain the difference between `EXISTS` and `IN` in a subquery.
    Answer: `EXISTS` returns true if the subquery returns one or more rows, stopping as soon as a row is found. `IN` checks if a value is present in the result set of a subquery. `EXISTS` is typically faster for large subquery result sets because it can short-circuit.

43. **Question:** How do you implement a bulk collect in PL/SQL?
    Answer: `BULK COLLECT` is used to retrieve multiple rows into a collection in a single fetch, improving performance. Example:

```
DECLARE
  TYPE emp_table IS TABLE OF employees%ROWTYPE;
  emp_data emp_table;
BEGIN
  SELECT * BULK COLLECT INTO emp_data FROM employees;
  FOR i IN 1..emp_data.COUNT LOOP
    DBMS_OUTPUT.PUT_LINE('ID: ' || emp_data(i).id || ' Name: ' || emp_data(i).name);
  END LOOP;
END;
```

44. **Question:** What is the `RETURNING` clause in SQL, and how is it used?
    Answer: The `RETURNING` clause retrieves values from rows affected by `INSERT`, `UPDATE`, or `DELETE` statements without the need for a subsequent `SELECT`. Example:

```
DECLARE
  v_id employees.id%TYPE;
  v_name employees.name%TYPE;
BEGIN
  INSERT INTO employees (id, name) VALUES (1, 'John') RETURNING id, name INTO v_id, v_name;
  DBMS_OUTPUT.PUT_LINE('Inserted ID: ' || v_id || ' Name: ' || v_name);
END;
```

45. **Question:** Explain the concept of data denormalization and when it might be used.
    Answer: Data denormalization involves combining normalized tables into larger tables to improve read performance at the expense of write performance and data redundancy. It is used in scenarios where read operations are more frequent than write operations, such as in data warehousing.

46. **Question:** How do you create and use a user-defined type (UDT) in Oracle?
    Answer: A UDT is a custom data type defined by the user. It can be used to define the structure of columns in a table or variables in PL/SQL. Example:

```
CREATE TYPE address_type AS OBJECT (
  street VARCHAR2(50),
  city VARCHAR2(50),
  postal_code VARCHAR2(10)
);

CREATE TABLE customers (
  id NUMBER,
  name VARCHAR2(50),
  address address_type
);
```

47. **Question:** What is the purpose of the `ORA-00001` error code, and how do you handle it?
    Answer: The `ORA-00001` error code indicates a unique constraint violation. It occurs when an attempt is made to insert or update a value that already exists in a column with a unique constraint. You can handle it using an exception block:

```
BEGIN
  INSERT INTO employees (id, name) VALUES (1, 'John');
EXCEPTION
  WHEN DUP_VAL_ON_INDEX THEN
    DBMS_OUTPUT.PUT_LINE('Duplicate value found');
END;
```

48. **Question:** Describe the use of the `WITH CHECK OPTION` clause in a view.
    Answer: The `WITH CHECK OPTION` clause ensures that all `INSERT` or `UPDATE` operations performed through the view adhere to the view's defining query. It prevents modifications that would result in rows not meeting the view's criteria.

49. **Question:** What is an inline view, and how is it different from a regular view?
    Answer: An inline view is a subquery in the `FROM` clause of a SQL statement that is treated as a table. It differs from a regular view in that it is not stored in the database schema but is defined and used within a single query. Example:

```
SELECT name, total_sales
FROM (SELECT employee_id, SUM(sales) AS total_sales FROM sales GROUP BY employee_id) sales_summary
JOIN employees ON sales_summary.employee_id = employees.id;
```

50. **Question:** Explain the use of the `TO_DATE` function in Oracle.
    Answer: The `TO_DATE` function converts a string to a date data type, based on the specified format. It is used to ensure that date strings are correctly interpreted by Oracle. Example:

```
SELECT TO_DATE('2024-05-17', 'YYYY-MM-DD') FROM DUAL;
```

Scenarios

Scenario 1: Online Retail Store Database Design

Scenario Description:
You need to design a database for an online retail store that keeps track of customers, products, orders, and order details.

Questions:

1. **Question:** Design the database schema with appropriate tables and relationships for the online retail store.
   Answer:

```
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY,
    FirstName VARCHAR2(50),
    LastName VARCHAR2(50),
    Email VARCHAR2(100),
    Address VARCHAR2(255)
);

CREATE TABLE Products (
    ProductID INT PRIMARY KEY,
    ProductName VARCHAR2(100),
    Price DECIMAL(10, 2),
    Stock INT
);

CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    CustomerID INT,
    OrderDate DATE,
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);

CREATE TABLE OrderDetails (
    OrderDetailID INT PRIMARY KEY,
    OrderID INT,
    ProductID INT,
    Quantity INT,
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
    FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
);
```

2. **Question:** Write a query to find the total number of orders placed by each customer.
   Answer:

```sql
SELECT C.CustomerID, C.FirstName, C.LastName, COUNT(O.OrderID) AS TotalOrders
FROM Customers C
LEFT JOIN Orders O ON C.CustomerID = O.CustomerID
GROUP BY C.CustomerID, C.FirstName, C.LastName;
```

3. **Question:** Optimize the design to include indexes that improve query performance.
   Answer:

```sql
CREATE INDEX idx_customers_email ON Customers (Email);
CREATE INDEX idx_orders_customerid ON Orders (CustomerID);
CREATE INDEX idx_orderdetails_orderid ON OrderDetails (OrderID);
CREATE INDEX idx_orderdetails_productid ON OrderDetails (ProductID);
```

4. **Question:** Write a PL/SQL procedure to add a new order with multiple order details.
   Answer:

```sql
CREATE OR REPLACE PROCEDURE AddOrder(
    p_CustomerID IN INT,
    p_OrderDate IN DATE,
    p_OrderDetails IN OrderDetailsType
) AS
BEGIN
    INSERT INTO Orders (CustomerID, OrderDate) VALUES (p_CustomerID, p_OrderDate);
    FOR i IN 1 .. p_OrderDetails.COUNT LOOP
        INSERT INTO OrderDetails (OrderID, ProductID, Quantity)
        VALUES (SEQ_OrderID.CURRVAL, p_OrderDetails(i).ProductID, p_OrderDetails(i).Quantity);
    END LOOP;
END;
```

5. **Question:** Write a query to retrieve all orders along with the customer details and the products ordered.
   Answer:

```sql
SELECT O.OrderID, C.FirstName, C.LastName, P.ProductName, OD.Quantity, O.OrderDate
FROM Orders O
JOIN Customers C ON O.CustomerID = C.CustomerID
JOIN OrderDetails OD ON O.OrderID = OD.OrderID
JOIN Products P ON OD.ProductID = P.ProductID;
```

Scenario 2: Library Management System Database Design

Scenario Description:
Design a database for a library management system to track books, members, and book loans.

Questions:

1. **Question:** Design the database schema with appropriate tables and relationships for the library management system.
   Answer:

```
CREATE TABLE Members (
    MemberID INT PRIMARY KEY,
    FirstName VARCHAR2(50),
    LastName VARCHAR2(50),
    Email VARCHAR2(100),
    Address VARCHAR2(255)
);

CREATE TABLE Books (
    BookID INT PRIMARY KEY,
    Title VARCHAR2(100),
    Author VARCHAR2(100),
    ISBN VARCHAR2(20),
    PublishedYear INT
);

CREATE TABLE Loans (
    LoanID INT PRIMARY KEY,
    MemberID INT,
    BookID INT,
    LoanDate DATE,
    ReturnDate DATE,
    FOREIGN KEY (MemberID) REFERENCES Members(MemberID),
    FOREIGN KEY (BookID) REFERENCES Books(BookID)
);
```

2. **Question:** Write a query to find the total number of books borrowed by each member.
   Answer:

```
SELECT M.MemberID, M.FirstName, M.LastName, COUNT(L.LoanID) AS TotalLoans
FROM Members M
LEFT JOIN Loans L ON M.MemberID = L.MemberID
GROUP BY M.MemberID, M.FirstName, M.LastName;
```

3. **Question:** Optimize the design to include indexes that improve query performance.
   Answer:

```
CREATE INDEX idx_members_email ON Members (Email);
CREATE INDEX idx_loans_memberid ON Loans (MemberID);
CREATE INDEX idx_loans_bookid ON Loans (BookID);
```

4. **Question:** Write a PL/SQL procedure to record a new loan for a member.
   Answer:

```
CREATE OR REPLACE PROCEDURE AddLoan(
    p_MemberID IN INT,
    p_BookID IN INT,
    p_LoanDate IN DATE
) AS
BEGIN
    INSERT INTO Loans (MemberID, BookID, LoanDate) VALUES (p_MemberID, p_BookID, p_LoanDate);
END;
```

5. **Question:** Write a query to retrieve all loans along with member and book details.
   Answer:

```
SELECT L.LoanID, M.FirstName, M.LastName, B.Title, L.LoanDate, L.ReturnDate
FROM Loans L
JOIN Members M ON L.MemberID = M.MemberID
JOIN Books B ON L.BookID = B.BookID;
```

Scenario 3: Employee Management System Database Design

Scenario Description:

Design a database for an employee management system to track employees, departments, and salaries.

Questions:

1. **Question:** Design the database schema with appropriate tables and relationships for the employee management system.
   Answer:

```
CREATE TABLE Departments (
    DepartmentID INT PRIMARY KEY,
    DepartmentName VARCHAR2(50)
);

CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    FirstName VARCHAR2(50),
    LastName VARCHAR2(50),
    Email VARCHAR2(100),
    DepartmentID INT,
    HireDate DATE,
    FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID)
);

CREATE TABLE Salaries (
    SalaryID INT PRIMARY KEY,
    EmployeeID INT,
    SalaryAmount DECIMAL(10, 2),
    SalaryDate DATE,
    FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID)
);
```

2. **Question:** Write a query to find the total salary paid to each employee.
   Answer:

```
SELECT E.EmployeeID, E.FirstName, E.LastName, SUM(S.SalaryAmount) AS TotalSalary
FROM Employees E
JOIN Salaries S ON E.EmployeeID = S.EmployeeID
GROUP BY E.EmployeeID, E.FirstName, E.LastName;
```

3. **Question:** Optimize the design to include indexes that improve query performance.
   Answer:

```
CREATE INDEX idx_employees_email ON Employees (Email);
CREATE INDEX idx_salaries_employeeid ON Salaries (EmployeeID);
```

4. **Question:** Write a PL/SQL function to calculate the total salary paid to an employee within a given date range.
   Answer:

```
CREATE OR REPLACE FUNCTION GetTotalSalary(
    p_EmployeeID IN INT,
    p_StartDate IN DATE,
    p_EndDate IN DATE
) RETURN DECIMAL IS
    v_TotalSalary DECIMAL(10, 2);
BEGIN
    SELECT SUM(SalaryAmount)
    INTO v_TotalSalary
    FROM Salaries
    WHERE EmployeeID = p_EmployeeID
      AND SalaryDate BETWEEN p_StartDate AND p_EndDate;
    RETURN v_TotalSalary;
END;
```

5. **Question:** Write a query to retrieve all employees along with their department names and total salaries.
   Answer:

```
SELECT E.EmployeeID, E.FirstName, E.LastName, D.DepartmentName, SUM(S.SalaryAmount) AS TotalSalary
FROM Employees E
JOIN Departments D ON E.DepartmentID = D.DepartmentID
JOIN Salaries S ON E.EmployeeID = S.EmployeeID
GROUP BY E.EmployeeID, E.FirstName, E.LastName, D.DepartmentName;
```

Scenario 4: School Management System Database Design

Scenario Description:
Design a database for a school management system to track students, courses, enrollments, and grades.

Questions:

1. **Question:** Design the database schema with appropriate tables and relationships for the school management system.
   Answer:

```
CREATE TABLE Students (
    StudentID INT PRIMARY KEY,
    FirstName VARCHAR2(50),
    LastName VARCHAR2(50),
    Email VARCHAR2(100),
    EnrollmentDate DATE
);

CREATE TABLE Courses (
    CourseID INT PRIMARY KEY,
    CourseName VARCHAR2(100),
    Credits INT
);

CREATE TABLE Enrollments (
    EnrollmentID INT PRIMARY KEY,
    StudentID INT,
    CourseID INT,
    Grade VARCHAR2(2),
    FOREIGN KEY (StudentID) REFERENCES Students(StudentID),
    FOREIGN KEY (CourseID) REFERENCES Courses(CourseID)
);
```

2. **Question:** Write a query to find the total number of courses each student is enrolled in.
   Answer:

```
SELECT S.StudentID, S.FirstName, S.LastName, COUNT(E.EnrollmentID) AS TotalCourses
FROM Students S
LEFT JOIN Enrollments E ON S.StudentID = E.StudentID
GROUP BY S.StudentID, S.FirstName, S.LastName;
```

3. **Question:** Optimize the design to include indexes that improve query performance.
   Answer:

```
CREATE INDEX idx_students_email ON Students (Email);
CREATE INDEX idx_enrollments_studentid ON Enrollments (StudentID);
CREATE INDEX idx_enrollments_courseid ON Enrollments (CourseID);
```

4. **Question:** Write a PL/SQL procedure to enroll a student in a course.
   Answer:

```
CREATE OR REPLACE PROCEDURE EnrollStudent(
    p_StudentID IN INT,
    p_CourseID IN INT,
    p_Grade IN VARCHAR2
) AS
BEGIN
    INSERT INTO Enrollments (StudentID, CourseID, Grade) VALUES (p_StudentID, p_CourseID, p_Grade);
END;
```

5. **Question:** Write a query to retrieve all students along with their enrolled courses and grades.
   Answer:

```
SELECT S.StudentID, S.FirstName, S.LastName, C.CourseName, E.Grade
FROM Students S
JOIN Enrollments E ON S.StudentID = E.StudentID
JOIN Courses C ON E.CourseID = C.CourseID;
```

Scenario 5: Hotel Reservation System Database Design

Scenario Description:
Design a database for a hotel reservation system to track guests, rooms, reservations, and payments.

Questions:

1. **Question:** Design the database schema with appropriate tables and relationships for the hotel reservation system.
   Answer:

```
CREATE TABLE Guests (
    GuestID INT PRIMARY KEY,
    FirstName VARCHAR2(50),
    LastName VARCHAR2(50),
    Email VARCHAR2(100),
    Phone VARCHAR2(15)
);

CREATE TABLE Rooms (
    RoomID INT PRIMARY KEY,
    RoomNumber VARCHAR2(10),
    RoomType VARCHAR2(50),
    Price DECIMAL(10, 2)
);

CREATE TABLE Reservations (
    ReservationID INT PRIMARY KEY,
    GuestID INT,
    RoomID INT,
    CheckInDate DATE,
    CheckOutDate DATE,
    FOREIGN KEY (GuestID) REFERENCES Guests(GuestID),
    FOREIGN KEY (RoomID) REFERENCES Rooms(RoomID)
);

CREATE TABLE Payments (
    PaymentID INT PRIMARY KEY,
    ReservationID INT,
    Amount DECIMAL(10, 2),
    PaymentDate DATE,
    FOREIGN KEY (ReservationID) REFERENCES Reservations(ReservationID)
);
```

2. **Question:** Write a query to find the total amount paid by each guest.
   Answer:

```sql
SELECT G.GuestID, G.FirstName, G.LastName, SUM(P.Amount) AS TotalAmountPaid
FROM Guests G
JOIN Reservations R ON G.GuestID = R.GuestID
JOIN Payments P ON R.ReservationID = P.ReservationID
GROUP BY G.GuestID, G.FirstName, G.LastName;
```

3. **Question:** Optimize the design to include indexes that improve query performance.
   Answer:

```sql
CREATE INDEX idx_guests_email ON Guests (Email);
CREATE INDEX idx_reservations_guestid ON Reservations (GuestID);
CREATE INDEX idx_reservations_roomid ON Reservations (RoomID);
CREATE INDEX idx_payments_reservationid ON Payments (ReservationID);
```

4. **Question:** Write a PL/SQL procedure to make a new reservation for a guest.
   Answer:

```sql
CREATE OR REPLACE PROCEDURE MakeReservation(
    p_GuestID IN INT,
    p_RoomID IN INT,
    p_CheckInDate IN DATE,
    p_CheckOutDate IN DATE
) AS
BEGIN
    INSERT INTO Reservations (GuestID, RoomID, CheckInDate, CheckOutDate)
    VALUES (p_GuestID, p_RoomID, p_CheckInDate, p_CheckOutDate);
END;
```

5. **Question:** Write a query to retrieve all reservations along with guest and room details.
   Answer:

```sql
SELECT R.ReservationID, G.FirstName, G.LastName, RM.RoomNumber, R.CheckInDate, R.CheckOutDate
FROM Reservations R
JOIN Guests G ON R.GuestID = G.GuestID
JOIN Rooms RM ON R.RoomID = RM.RoomID;
```