1. What is a stored program unit in a database?

Table Procedure

Trigger View

1. Which of the following is not a part of a procedure in a database?

Declaration section Exception section

Body section Trigger section

1. In parameter modes for procedures, which mode allows you to pass values from the calling program to the procedure and vice versa?

IN OUT

IN OUT DEFAULT

1. What is one of the advantages of using procedures in a database?

Simplified data modeling Enhanced security

Dynamic table creation Reduced data redundancy

1. Which of the following is not a type of trigger in a database?

Before Trigger After Trigger

During Trigger Instead Of Trigger

1. What is the syntax for creating a trigger in SQL?

CREATE PROCEDURE CREATE FUNCTION

CREATE TRIGGER CREATE VIEW

1. Which type of trigger is fired before the execution of a DML statement in SQL?

Before Trigger After Trigger

Instead Of Trigger Concurrent Trigger

1. What is the purpose of a package specification in a database?
   1. Contains the implementation details of a package
   2. Declares the public interface of a package
   3. Stores data for the package
   4. Executes procedures in the package
2. Which part of a package in a database contains the actual code and implementation details?

Package specification Package body

Package header Package declaration

1. What is a "bodiless package" in a database?
   1. A package without a package specification
   2. A package without a package body
   3. A package without any code or implementation
   4. A package without parameters
2. Which of the following is not an advantage of using packages in a database?

Encapsulation of code Improved performance

Simplified maintenance Increased data redundancy

1. What does a "PRAGMA AUTONOMOUS\_TRANSACTION" do in a trigger?

Executes the trigger automatically Is used to create a trigger

Starts a new transaction within the trigger Stops a running transaction

1. Which of the following is a valid parameter mode for a procedure in SQL?

IN OUT INCREMENT OUT DECREMENT

INCREMENT OUT OUT IN

1. What type of trigger is used to replace a DML statement with another statement?

Before Trigger After Trigger

Instead Of Trigger Around Trigger

1. What is the primary purpose of a trigger in a database?

To define data types To declare variables

To enforce data integrity and automate actions To create views

1. Which of the following is not a valid part of a procedure in SQL?

Parameter list Declaration section

Exception section Trigger section

1. What is the primary purpose of a package in SQL?

To store data To define triggers

To group related procedures, functions, and variables To create views

1. Which parameter mode allows a procedure to receive values from the calling program but not return values back?

IN OUT

IN OUT DEFAULT

1. Which of the following is not a type of package in SQL?

Standalone package Bodiless package

Composite package Nested package

1. What is the advantage of using triggers in a database?

Improved code encapsulation Enhanced security

Simplified package creation Dynamic table creation

1. What is the primary purpose of a transaction in a DBMS?

To retrieve data from the database To update the database

To manage database schemas To organize data into tables

1. Which of the following is not a property of a transaction in DBMS?

Isolation Atomicity

Consistency Transparency

1. Which property of transactions ensures that a transaction's changes are permanent and will survive system failures?

Atomicity Consistency

Isolation Durability

1. What does the ACID acronym stand for in the context of transactions?
   1. Atomicity, Completeness, Isolation, Durability
   2. Atomicity, Consistency, Isolation, Durability
   3. Availability, Consistency, Isolation, Durability
   4. Allotment, Concurrency, Integration, Durability
2. Which of the following is not a common concurrency problem in DBMS?
   1. Deadlock Dirty Read
   2. Lost Update Normalization
3. What is the primary goal of a concurrency control mechanism in a DBMS?

To ensure that transactions are executed concurrently without any restrictions

To improve the performance of the database system

To prevent conflicts and maintain data consistency in a multi-user environment

To reduce the storage space required for the database

1. Which of the following is a common method for achieving isolation in DBMS?

Two-Phase Locking Time-based Synchronization

Data Duplication Data Sharding

1. What is a serializable schedule in the context of concurrency control?

A schedule in which transactions are executed one after the other

A schedule that preserves the original order of transactions

A schedule that produces the same result as if transactions were executed serially

A schedule that allows concurrent execution without any restrictions

1. Which of the following is a benefit of using serializability in a DBMS?

Improved system performance Reduced data consistency

Enhanced data integrity Faster query processing

1. What is the primary purpose of a lock in a DBMS?

To restrict access to specific data items To encrypt data for security

To optimize query execution To permanently delete data

1. In a DBMS, what is a shared lock used for?

To allow multiple transactions to write to the same data item

To prevent multiple transactions from reading the same data item simultaneously

To allow multiple transactions to read the same data item simultaneously

To break deadlocks

1. Which concurrency control technique uses a timeout mechanism to resolve conflicts?

Two-Phase Locking Timestamp-Based Protocol

Optimistic Concurrency Control Strict Two-Phase Locking

1. What is a deadlock in the context of concurrency control?

A situation where two transactions are waiting for each other to release locks

A situation where a transaction is permanently blocked

A situation where transactions cannot be rolled back

A situation where transactions cannot be committed

1. What is the purpose of a deadlock detection mechanism in a DBMS?

To prevent deadlocks from occurring To identify and resolve deadlocks

To escalate conflicts between transactions To increase the isolation level

1. Which of the following is not a common deadlock prevention technique?

Wait-Die Wound-Wait

Timeout Rollback

1. What is the purpose of an intent lock in a DBMS?

To indicate the intention of a transaction to acquire a shared lock

To indicate the intention of a transaction to acquire an exclusive lock

To prevent deadlocks To release all locks

1. In a DBMS, what is the purpose of a transaction log?

To record all user queries

To store database schema information

To maintain a record of all committed and uncommitted transactions

To store backup copies of data

1. What is the primary goal of a checkpoint in a DBMS?
   1. To initiate a transaction rollback
   2. To recover from system crashes
   3. To release all locks held by a transaction
   4. To optimize query execution
2. Which of the following is an example of a conflict-serializable schedule in a DBMS?

Schedule S1: T1 → T2 → T3 Schedule S2: T2 → T1 → T3

Schedule S3: T1 → T3 → T2 Schedule S4: T3 → T1 → T2

1. What does the isolation level "Serializable" in a DBMS ensure?

It allows dirty reads

It provides the highest level of isolation

It allows transactions to write to the same data simultaneously

It does not allow any concurrency

1. Which of the following is a benefit of using a lower isolation level, such as "Read Uncommitted," in a DBMS?

Improved data integrity Higher isolation between transactions

Reduced contention for locks Faster query performance

1. In the context of locking, what is a lock mode?

The time duration for which a lock is held

The type of lock (shared or exclusive) and its compatibility with other locks

The order in which locks are acquired

The number of transactions waiting for a lock

1. Which of the following is a drawback of using a high isolation level, such as "Serializable," in a DBMS?

Increased likelihood of deadlocks Improved data consistency

Lower transaction throughput Reduced data integrity

1. What is the purpose of a transaction manager in a DBMS?

To optimize query execution To manage database schemas

To ensure the ACID properties of transactions To store backup copies of data

1. What is the primary goal of a deadlock prevention technique like "Wait-Die"?

To escalate conflicts between transactions

To prevent transactions from waiting indefinitely

To improve query performance To increase data redundancy

1. In a DBMS, what is a transaction's isolation level?

The number of locks acquired by the transaction

The duration for which a transaction is active

The level of visibility a transaction has into other transactions' changes

The number of concurrent transactions

1. Which of the following is a disadvantage of using optimistic concurrency control?

Higher contention for locks Increased likelihood of deadlocks

Slower query performance Limited data consistency

1. What is the purpose of a timestamp in a DBMS?

To record the time when a transaction started To ensure data encryption

To prevent conflicts between transactions To optimize query execution

1. What is a conflict-serializable schedule?

A schedule that contains conflicts between transactions

A schedule in which transactions are executed serially

A schedule that preserves the original order of transactions

A schedule that is equivalent to a serial schedule with the same transactions

1. Which of the following is not a common concurrency control mechanism in a DBMS?

Two-Phase Commit Optimistic Concurrency Control

Strict Two-Phase Locking Timestamp-Based Protocol

1. What is the purpose of a transaction ID in a DBMS?

To identify the user who initiated the trans

To indicate the transaction's priority

To uniquely identify and track each transaction

To store backup copies of data

1. Which of the following is a benefit of using a higher isolation level, such as "Serializable," in a DBMS?

Improved query performance Reduced likelihood of deadlocks

Higher data consistency Lower transaction throughput

1. What is the primary goal of a timeout-based deadlock prevention technique?

To prevent transactions from waiting indefinitely

To prioritize transactions based on their importance

To escalate conflicts between transactions

To increase the isolation level

1. What does a "lost update" refer to in the context of concurrency control?

A situation where a transaction is permanently blocked

A situation where a transaction is rolled back

A situation where one transaction overwrites the changes made by another transaction

A situation where transactions cannot be committed

1. Which of the following is not a common technique for deadlock detection in a DBMS?

Wait-Die Wound-Wait

Timeout Rollback

1. What is the primary purpose of a deadlock prevention technique like "Wound-Wait"?

To escalate conflicts between transactions

To prevent transactions from waiting indefinitely

To improve query performance To increase data redundancy

1. Which of the following statements about the "Repeatable Read" isolation level in a DBMS is true?

It allows dirty reads. It allows lost updates.

It prevents phantom reads. It has the lowest isolation level.

1. What is the primary goal of a transaction recovery manager in a DBMS?

To escalate conflicts between transactions To optimize query execution

To ensure the durability of transactions To manage database schemas

1. Which of the following is not a common cause of deadlocks in a DBMS?
   1. Circular Wait Resource Preemption
   2. Hold and Wait No Concurrency
2. What is the purpose of a data dictionary in a DBMS?

To store user data To maintain a log of transactions

To manage database schemas and metadata To perform data encryption

# Assuming we have a "Bank" table with the following sample data:

| account\_number | account\_holder | balance |

| | - | |

| 1001 | John Doe | 5000.00 |

| 1002 | Jane Smith | 7500.50 |

| 1003 | Alice Johnson | 3000.25 |

```sql

CREATE TABLE Bank (

account\_number NUMBER PRIMARY KEY, account\_holder VARCHAR2(100), balance NUMBER(10, 2)

);

Here's a PL/SQL code snippet based on this data:

PL/SQL Code Snippet:

```plsql DECLARE

v\_balance NUMBER;

BEGIN

SELECT balance INTO v\_balance FROM Bank

WHERE account\_holder = 'John Doe';

DBMS\_OUTPUT.PUT\_LINE('John Doe\'s Balance: $' || v\_balance); END;

```

# Based on above details gives the following question’s answer

1. What is the primary purpose of the PL/SQL code snippet?
   1. Updates John Doe's account balance
   2. Deletes John Doe's account record
   3. Retrieves and displays John Doe's account balance
   4. Inserts a new account record for John Doe

Answer: c) Retrieves and displays John Doe's account balance

1. What is the data type of the "balance" column in the "Bank" table?
   1. String
   2. Date
   3. Number
   4. Boolean

Answer: c) Number

1. Which SQL operation is performed in the PL/SQL code?
   1. INSERT
   2. DELETE
   3. SELECT
   4. UPDATE

Answer: c) SELECT

1. What is the purpose of the `INTO` clause in the code?
   1. To indicate the end of the PL/SQL block
   2. To declare a new variable
   3. To specify the source of data for the SELECT statement
   4. To define a cursor

Answer: c) To specify the source of data for the SELECT statement

1. What does the `DBMS\_OUTPUT.PUT\_LINE` statement do in the code?
   1. Updates the database records
   2. Deletes database records
   3. Retrieves data from the database
   4. Displays a message in the console

Answer: d) Displays a message in the console

# here are 10 multiple-choice questions (MCQs) based on a PL/SQL code snippet

PL/SQL Code Snippet:

```plsql DECLARE

v\_employee\_count NUMBER; BEGIN

SELECT COUNT() INTO v\_employee\_count FROM Employees;

DBMS\_OUTPUT.PUT\_LINE('Total Employees: ' || v\_employee\_count); END;

```

# MCQs:

1. What is the primary purpose of the PL/SQL code snippet?
   1. Updates employee records
   2. Deletes employee records
   3. Retrieves and displays the total number of employees
   4. Inserts a new employee record

Answer: c) Retrieves and displays the total number of employees

1. In the code snippet, what is the value stored in the `v\_employee\_count` variable?
   1. Employee names
   2. Employee IDs
   3. Total number of employees
   4. Employee salaries

Answer: c) Total number of employees

1. Which SQL operation is performed in the PL/SQL code?
   1. INSERT
   2. DELETE
   3. SELECT
   4. UPDATE

Answer: c) SELECT

1. What is the purpose of the `INTO` clause in the code?
   1. To indicate the end of the PL/SQL block
   2. To declare a new variable
   3. To specify the source of data for the SELECT statement
   4. To define a cursor

Answer: c) To specify the source of data for the SELECT statement

1. What does the `DBMS\_OUTPUT.PUT\_LINE` statement do in the code?
   1. Updates the database records
   2. Deletes database records
   3. Retrieves data from the database
   4. Displays a message in the console

Answer: d) Displays a message in the console

1. Which PL/SQL construct allows you to handle exceptions in a structured manner?
   1. TRY-CATCH
   2. EXCEPTION
   3. ERROR-HANDLER
   4. ON-ERROR

Answer: b) EXCEPTION

1. In PL/SQL, what is the primary purpose of a cursor?
   1. To define variables
   2. To loop through a result set
   3. To declare procedures
   4. To manage transactions

Answer: b) To loop through a result set

1. What is the expected output of the code snippet if there are 100 employees in the "Employees" table?
   1. Total Employees: 100
   2. Total Employees: 0
   3. Total Employees: 1
   4. Total Employees: 99

Answer: a) Total Employees: 100

1. What type of variable is `v\_employee\_count` in the code snippet?
   1. String
   2. Date
   3. Number
   4. Boolean

Answer: c) Number

1. In PL/SQL, how can you pass a parameter to a stored procedure?
   1. Using a RETURN statement
   2. Using a SELECT statement
   3. Using an IN parameter
   4. Using a WHERE clause

Answer: c) Using an IN parameter

# Here's a PL/SQL package with a "College" table and some basic code snippets :

```sql

-- Create the College table CREATE TABLE College (

student\_id NUMBER PRIMARY KEY, student\_name VARCHAR2(50), major VARCHAR2(50)

);

+ + + +

| student\_id | student\_name | major |

+ + + +

|  |  |  |
| --- | --- | --- |
| | | 1 | | John Smith | Computer Science | |
| | | 2 | | Jane Doe | Biology | |
| | | 3 | | Alice Johnson| History | |
| | | 4 | | Bob Brown | Mathematics | |
| | | 5 | | Eva Williams| Chemistry | |

+ + + +

```plsql

-- Create a PL/SQL package

CREATE OR REPLACE PACKAGE College\_Package AS

-- Function to retrieve student count by major

FUNCTION getStudentCountByMajor(major IN VARCHAR2) RETURN NUMBER; FUNCTION mcq1 RETURN VARCHAR2;

FUNCTION mcq2 RETURN NUMBER;

END College\_Package;

/

CREATE OR REPLACE PACKAGE BODY College\_Package AS

-- Function to retrieve student count by major

FUNCTION getStudentCountByMajor(major IN VARCHAR2) RETURN NUMBER IS cnt NUMBER;

BEGIN

SELECT COUNT() INTO cnt FROM College WHERE major = major; RETURN cnt;

END;

FUNCTION mcq1 RETURN VARCHAR2 IS BEGIN

RETURN 'student\_id'; END;

FUNCTION mcq2 RETURN NUMBER IS

biology\_count NUMBER; BEGIN

biology\_count := getStudentCountByMajor('Biology'); RETURN biology\_count;

END;

END College\_Package;

/

```

1 Which column is used to uniquely identify students?

1. student\_id student\_name
2. major None of the above
3. How many students are majoring in Computer Science?
4. 2

3 0

3: What is the data type of the "student\_name" column in the College table?

1. NUMBER
2. VARCHAR2
3. DATE
4. BOOLEAN

Answer: B) VARCHAR2

MCQ 4: Which PL/SQL construct is used to loop through records in a result set?

1. FOR loop
2. IF statement
3. WHILE loop
4. CASE statement Answer: A) FOR loop

MCQ 5: How many students are majoring in Chemistry?

1. 1
2. 2
3. 3
4. 0

Answer: D) 0

MCQ 6: Which PL/SQL keyword is used to declare a variable?

1. DEFINE
2. DECLARE
3. VARIABLE
4. SET

Answer: B) DECLARE

MCQ 7: What is the output of the following PL/SQL code?

```plsql DECLARE

total\_students NUMBER; BEGIN

total\_students := College\_Package.getStudentCountByMajor('Computer Science'); DBMS\_OUTPUT.PUT\_LINE('Total students in Computer Science: ' || total\_students);

END;

```

1. Total students in Computer Science: 1
2. Total students in Computer Science: 2
3. Total students in Computer Science: 3
4. Total students in Computer Science: 0

Answer: A) Total students in Computer Science: 1

MCQ 8: Which PL/SQL statement is used to raise an exception?

1. RAISE
2. THROW
3. EXCEPTION
4. ERROR

Answer: A) RAISE

MCQ 9: What is the purpose of the PRIMARY KEY constraint in the College table?

1. It enforces unique values in the "student\_name" column.
2. It enforces unique values in the "major" column.
3. It ensures that the "student\_id" column is not null.
4. It uniquely identifies each row in the table.

Answer: D) It uniquely identifies each row in the table.

MCQ 10: Which PL/SQL construct is used to handle exceptions in a controlled manner?

1. TRY...CATCH block
2. EXCEPTION block
3. ERROR block
4. HANDLE block

Answer: B) EXCEPTION block

# Here's a PL/SQL code snippet for a hypothetical "hospital" table, along with 10 multiple-choice questions (MCQs)

Let's create a PL/SQL trigger for the "hospital" table. This trigger updates the "patient\_count" column in a separate "hospital\_stats" table whenever a new patient is inserted into the "hospital" table.

```sql

-- Create the hospital\_stats table to store statistics. CREATE TABLE hospital\_stats (

total\_patients NUMBER

);

-- Create a sequence to generate unique IDs for each patient. CREATE SEQUENCE patient\_id\_seq START WITH 1;

-- Create the hospital table. CREATE TABLE hospital (

patient\_id NUMBER PRIMARY KEY, patient\_name VARCHAR2(50), admission\_date DATE,

discharge\_date DATE

);

-- Create the trigger to update patient count in hospital\_stats.

CREATE OR REPLACE TRIGGER update\_patient\_count AFTER INSERT ON hospital

FOR EACH ROW BEGIN

UPDATE hospital\_stats

SET total\_patients = total\_patients + 1; END;

/

```

# Multiple-Choice Questions (MCQs):

1. What is the purpose of the "update\_patient\_count" trigger in the "hospital" table?
   1. To automatically update all patient records.
   2. To update the total count of patients in the "hospital\_stats" table when a new patient is inserted.
   3. To prevent new records from being inserted.
   4. To calculate the average length of stay for all patients.

Correct Answer: b

1. In which event(s) will the "update\_patient\_count" trigger execute?
   1. Before inserting a new patient record.
   2. After deleting a patient record.
   3. Before updating an existing patient record.
   4. After inserting a new patient record.

Correct Answer: d

1. What does `AFTER INSERT ON hospital` mean in the trigger definition?
   1. The trigger fires before a new patient record is inserted.
   2. The trigger fires after a patient record is deleted.
   3. The trigger fires after a new patient record is inserted.
   4. The trigger fires before an existing patient record is updated.

Correct Answer: c

1. What is the purpose of the "hospital\_stats" table in the code snippet?
   1. To store patient names.
   2. To store admission and discharge dates.
   3. To store statistics related to the hospital, such as the total number of patients.
   4. To store the patient IDs.

Correct Answer: c

1. How is the "patient\_id" assigned in the "hospital" table?
   1. Manually entered by the user.
   2. Generated automatically using a sequence.
   3. Copied from the "patient\_id" in the "hospital\_stats" table.
   4. Set to a constant value.

Correct Answer: b

1. What happens if you attempt to insert a new patient record without specifying values for "patient\_name," "admission\_date," and "discharge\_date"?
   1. The trigger inserts default values.
   2. The trigger raises an error.
   3. The trigger inserts NULL values.
   4. The trigger generates random values.

Correct Answer: b

1. Which keyword is used to specify the trigger action timing in PL/SQL?
   1. WHEN
   2. BEFORE
   3. AFTER
   4. TRIGGER

Correct Answer: c

1. What is the primary purpose of the `UPDATE hospital\_stats SET total\_patients = total\_patients + 1;` statement in the trigger?
   1. To delete a patient record.
   2. To insert a new patient record.
   3. To update the "patient\_count" column in the "hospital\_stats" table.
   4. To calculate the average length of stay for all patients.

Correct Answer: c

1. Can you have multiple triggers with the same timing (e.g., AFTER INSERT) on the same table?
   1. No, only one trigger is allowed per table.
   2. Yes, but they must have different names.
   3. Yes, and they execute in a random order.
   4. No, it will result in an error.

Correct Answer: b

1. What does the `CREATE SEQUENCE patient\_id\_seq START WITH 1;` statement do in the code snippet?
   1. It creates a new table.
   2. It defines a new trigger.
   3. It creates a sequence for generating unique patient IDs.
   4. It initializes the patient ID to 1.

Correct Answer: c

A PL/SQL procedure that takes two numbers as input parameters, adds them together, and then displays the result using dbms\_output:

```sql

CREATE OR REPLACE PROCEDURE add\_numbers ( p\_num1 IN NUMBER,

p\_num2 IN NUMBER

) AS

v\_result NUMBER; BEGIN

-- Perform the addition

v\_result := p\_num1 + p\_num2;

-- Display the result

DBMS\_OUTPUT.PUT\_LINE('The sum of ' || p\_num1 || ' and ' || p\_num2 || ' is ' || v\_result); END add\_numbers;

/

```

Here's an example of how to call this procedure:

```sql DECLARE

num1 NUMBER := 10;

num2 NUMBER := 20; BEGIN

add\_numbers(num1, num2); END;

/

```

This will call the `add\_numbers` procedure with `num1` and `num2` as arguments and display the sum.

# Based on given pl sql answer the following mcq:

1. What is the purpose of the PL/SQL procedure mentioned in the code snippet?
   1. To subtract two numbers.
   2. To add two numbers and display the result.
   3. To multiply two numbers.
   4. To divide two numbers.

Answer: B

1. How many input parameters does the `add\_numbers` procedure have?
   1. None
   2. One
   3. Two
   4. Three Answer: C
2. What data type are the input parameters `p\_num1` and `p\_num2` in the `add\_numbers` procedure?
   1. VARCHAR2
   2. DATE
   3. NUMBER
   4. BOOLEAN Answer: C
3. What is the purpose of the `DBMS\_OUTPUT.PUT\_LINE` statement in the procedure?
   1. It calculates the sum of two numbers.
   2. It displays the result of the addition.
   3. It defines a new variable.
   4. It retrieves data from the database.

Answer: B

1. How is the result of the addition operation displayed in the output?
   1. Using the PRINT statement
   2. Using the RETURN statement
   3. Using the DBMS\_OUTPUT.PUT\_LINE statement
   4. Using the DISPLAY statement Answer: C
2. What should you do to call the `add\_numbers` procedure with specific numbers as arguments?
   1. Use the CALL statement.
   2. Use the SELECT statement.
   3. Use the DECLARE block.
   4. Use the EXECUTE statement.

Answer: C

1. In the example provided for calling the procedure, what are the values of `num1` and `num2`?

A. num1 = 20, num2 = 10

B. num1 = 10, num2 = 30

C. num1 = 10, num2 = 20

D. num1 = 30, num2 = 10 Answer: C

1. What is the result of calling the `add\_numbers` procedure with `num1` and `num2` as arguments in the example?
   1. 10
   2. 20
   3. The sum of 10 and 20 is 30
   4. There will be no output.

Answer: C

1. Which SQL statement is used to create a PL/SQL procedure?
   1. CREATE PROCEDURE
   2. DECLARE PROCEDURE
   3. EXECUTE PROCEDURE
   4. CALL PROCEDURE Answer: A
2. What is the purpose of the `DECLARE` block in the example?
   1. To define a new variable.
   2. To execute SQL statements.
   3. To declare and initialize variables before calling the procedure.
   4. To declare a function.

Answer: C

Ques1 - Consider the following PL/SQL function: (Difficulty level – Easy)

CREATE OR REPLACE FUNCTION calculate\_total(price NUMBER, quantity NUMBER)

RETURN NUMBER IS

total NUMBER; BEGIN

total := price \* quantity; RETURN total;

END;

What does the PL/SQL function `calculate\_total` do?

1. It calculates the average of `price` and `quantity`.
2. It calculates the sum of `price` and `quantity`.
3. It calculates the product of `price` and `quantity`.
4. It calculates the difference between `price` and `quantity`.

Correct Option: C

---

Ques2 - Consider the following PL/SQL function: (Difficulty level – Easy)

CREATE OR REPLACE FUNCTION greet(name VARCHAR2)

RETURN VARCHAR2 IS

greeting VARCHAR2(100); BEGIN

greeting := 'Hello, ' || name || '!'; RETURN greeting;

END;

\*\*What does the PL/SQL function `greet` do?\*\*

1. It calculates the length of the input string `name`.
2. It calculates the square of a numeric input.
3. It generates a greeting message with the input `name`.
4. It calculates the factorial of a numeric input.

\*\*Correct Option:\*\* C

---

Ques3 - Consider the following PL/SQL function: (Difficulty level – Easy)

```plsql

CREATE OR REPLACE FUNCTION is\_even(num NUMBER) RETURN BOOLEAN IS

BEGIN

IF MOD(num, 2) = 0 THEN RETURN TRUE;

ELSE

RETURN FALSE; END IF;

END;

```

\*\*What does the PL/SQL function `is\_even` do?\*\*

1. It checks if the input `num` is an even number and returns `TRUE` if it is, `FALSE` otherwise.
2. It checks if the input `num` is a positive number and returns `TRUE` if it is, `FALSE` otherwise.
3. It checks if the input `num` is a prime number and returns `TRUE` if it is, `FALSE` otherwise.
4. It calculates the factorial of the input `num`.

\*\*Correct Option:\*\* A

---

Ques4 - Consider the following PL/SQL function: (Difficulty level – Easy)

```plsql

CREATE OR REPLACE FUNCTION get\_employee\_salary(emp\_id NUMBER) RETURN NUMBER IS

salary NUMBER; BEGIN

*-- Retrieve the salary of the employee with the given emp\_id*

SELECT salary INTO salary FROM employees WHERE employee\_id

= emp\_id;

RETURN salary; END;

```

What does the PL/SQL function `get\_employee\_salary` do?

1. It calculates the average salary of all employees.
2. It retrieves the salary of the employee with the specified `emp\_id`.
3. It calculates the total salary of all employees.
4. It retrieves the highest salary among all employees.

\*\*Correct Option:\*\* B

---

Ques5 - Consider the following PL/SQL function: (Difficulty level – Easy)

CREATE OR REPLACE FUNCTION convert\_to\_uppercase(text VARCHAR2)

RETURN VARCHAR2 IS

upper\_text VARCHAR2(100); BEGIN

upper\_text := UPPER(text); RETURN upper\_text;

END;

What does the PL/SQL function `convert\_to\_uppercase` do?

1. It calculates the length of the input `text`.
2. It calculates the square of a numeric input.
3. It converts the input `text` to uppercase.
4. It calculates the factorial of a numeric input.

Correct Option: C

\*\*Ques6 - Consider the following PL/SQL function: (Difficulty level – Medium)\*\*

```plsql

CREATE OR REPLACE FUNCTION calculate\_tax(income NUMBER) RETURN NUMBER IS

tax NUMBER; BEGIN

IF income <= 50000 THEN tax := income \* 0.1;

ELSE

tax := 50000 \* 0.1 + (income - 50000) \* 0.2; END IF;

RETURN tax; END;

```

\*\*What does the PL/SQL function `calculate\_tax` do?\*\*

1. It calculates the total income after applying a tax rate.
2. It calculates the square root of the input number `income`.
3. It calculates the factorial of the input number `income`.
4. It calculates the tax amount based on the input income.

\*\*Correct Option:\*\* D

---

\*\*Ques7 - Consider the following PL/SQL function: (Difficulty level – Medium)\*\*

```plsql

CREATE OR REPLACE FUNCTION reverse\_and\_uppercase(input\_str VARCHAR2)

RETURN VARCHAR2 IS

reversed\_upper VARCHAR2(255); BEGIN

reversed\_upper := UPPER(REVERSE(input\_str)); RETURN reversed\_upper;

END;

```

\*\*What does the PL/SQL function `reverse\_and\_uppercase` do?\*\*

1. It calculates the length of the input string `input\_str`.
2. It calculates the square of the input number `input\_str`.
3. It reverses the characters in the input string `input\_str` and converts them to uppercase.
4. It calculates the factorial of the input number `input\_str`.

\*\*Correct Option:\*\* C

---

\*\*Ques8 - Consider the following PL/SQL function: (Difficulty level – Medium)\*\*

```plsql

CREATE OR REPLACE FUNCTION find\_largest

(numbers VARCHAR2) RETURN NUMBER IS

largest NUMBER := NULL; num\_list VARCHAR2(255); num\_str VARCHAR2(10);

BEGIN

num\_list := TRIM(BOTH ',' FROM numbers); LOOP

EXIT WHEN LENGTH(num\_list) = 0;

num\_str := TRIM(SUBSTR(num\_list, 1, INSTR(num\_list, ',') - 1));

num\_list := SUBSTR(num\_list, INSTR(num\_list, ',') + 1); IF TO\_NUMBER(num\_str) > largest OR largest IS NULL THEN

largest := TO\_NUMBER(num\_str); END IF;

END LOOP;

RETURN largest; END;

```

\*\*What does the PL/SQL function `find\_largest` do?\*\*

1. It calculates the square root of the input string `numbers`.
2. It calculates the sum of all numbers in the input string `numbers`.
3. It retrieves the largest number from a comma-separated list of numbers in the input string

`numbers`.

1. It calculates the factorial of all numbers in the input string `numbers`.

\*\*Correct Option:\*\* C

---

\*\*Ques9 - Consider the following PL/SQL function: (Difficulty level – Medium)\*\*

```plsql

CREATE OR REPLACE FUNCTION generate\_invoice(total\_amount NUMBER, customer\_id NUMBER)

RETURN VARCHAR2 IS

invoice\_text VARCHAR2(500); customer\_name VARCHAR2(255);

BEGIN

-- Retrieve the customer's name based on the customer\_id SELECT name INTO customer\_name FROM customers WHERE

customer\_id = customer\_id;

invoice\_text := 'Invoice for ' || customer\_name || ': Total Amount - $' || total\_amount;

RETURN invoice\_text; END;

```

\*\*What does the PL/SQL function `generate\_invoice` do?\*\*

1. It generates an invoice text for a customer with the specified `customer\_id` and total amount.
2. It calculates the average total amount for all customers.
3. It retrieves the customer's name based on the customer\_id.
4. It calculates the total amount for a customer with the specified `customer\_id`.

\*\*Correct Option:\*\* A

\*\*Ques10 - Consider the following PL/SQL package specification: (Difficulty level – Hard)\*\*

```plsql

CREATE OR REPLACE PACKAGE product\_recommendations AS FUNCTION recommend\_products(customer\_id NUMBER) RETURN

VARCHAR2;

FUNCTION get\_product\_rating(product\_id NUMBER) RETURN NUMBER;

FUNCTION get\_product\_reviews(product\_id NUMBER) RETURN NUMBER;

END product\_recommendations;

/

```

\*\*What does the PL/SQL package `product\_recommendations` contain?\*\*

1. It contains three PL/SQL functions, `recommend\_products`, `get\_product\_rating`, and

`get\_product\_reviews`, for providing product recommendations and retrieving product ratings and reviews.

1. It contains two PL/SQL triggers, `recommend\_products`, `get\_product\_rating`, and

`get\_product\_reviews`, for providing product recommendations and retrieving product ratings and reviews.

1. It contains four PL/SQL procedures, `recommend\_products`, `get\_product\_rating`, and

`get\_product\_reviews`, for providing product recommendations and retrieving product ratings and reviews.

1. It contains one PL/SQL function, `product\_recommendations`, and one PL/SQL procedure,

`product\_recommendations`, for providing product recommendations and retrieving product ratings and reviews.

\*\*Correct Option:\*\* A

---

\*\*Ques11 - Consider the following PL/SQL package specification: (Difficulty level – Hard)\*\*

```plsql

CREATE OR REPLACE PACKAGE inventory\_management AS FUNCTION check\_stock\_availability(product\_id NUMBER,

warehouse\_id NUMBER) RETURN BOOLEAN;

FUNCTION transfer\_product(product\_id NUMBER, source\_warehouse\_id NUMBER, destination\_warehouse\_id NUMBER, quantity NUMBER) RETURN NUMBER;

FUNCTION get\_product\_location(product\_id NUMBER) RETURN VARCHAR2;

END inventory\_management;

/

```

\*\*What does the PL/SQL package `inventory\_management` contain?\*\*

1. It contains four PL/SQL functions, `check\_stock\_availability`, `transfer\_product`, and

`get\_product\_location`, for managing inventory and retrieving product locations.

1. It contains three PL/SQL triggers, `check\_stock\_availability`, `transfer\_product`, and

`get\_product\_location`, for managing inventory and retrieving product locations.

1. It contains two PL/SQL

procedures, `check\_stock\_availability`, `transfer\_product`, and `get\_product\_location`, for managing inventory and retrieving product locations.

1. It contains one PL/SQL function, `inventory\_management`, and one PL/SQL procedure,

`inventory\_management`, for managing inventory and retrieving product locations.

\*\*Correct Option:\*\* A

---

\*\*Ques12 - Consider the following PL/SQL package specification: (Difficulty level – Hard)\*\*

```plsql

CREATE OR REPLACE PACKAGE customer\_order\_history AS FUNCTION get\_order\_count(customer\_id NUMBER) RETURN

NUMBER;

FUNCTION get\_average\_order\_value(customer\_id NUMBER) RETURN NUMBER;

FUNCTION get\_last\_order\_date(customer\_id NUMBER) RETURN DATE;

END customer\_order\_history;

/

```

\*\*What does the PL/SQL package `customer\_order\_history` contain?\*\*

1. It contains three PL/SQL functions, `get\_order\_count`, `get\_average\_order\_value`, and

`get\_last\_order\_date`, for retrieving customer order history statistics.

1. It contains three PL/SQL triggers, `get\_order\_count`, `get\_average\_order\_value`, and

`get\_last\_order\_date`, for retrieving customer order history statistics.

1. It contains three PL/SQL procedures, `get\_order\_count`, `get\_average\_order\_value`, and

`get\_last\_order\_date`, for retrieving customer order history statistics.

1. It contains one PL/SQL function, `customer\_order\_history`, and one PL/SQL procedure,

`customer\_order\_history`, for retrieving customer order history statistics.

\*\*Correct Option:\*\* A

---

\*\*Ques13 - Consider the following PL/SQL package specification: (Difficulty level – Hard)\*\*

```plsql

CREATE OR REPLACE PACKAGE employee\_performance AS

FUNCTION calculate\_performance\_rating(employee\_id NUMBER, year NUMBER) RETURN NUMBER;

FUNCTION get\_top\_performing\_employee(year NUMBER) RETURN VARCHAR2;

END employee\_performance;

/

```

\*\*What does the PL/SQL package `employee\_performance` contain?\*\*

1. It contains two PL/SQL functions, `calculate\_performance\_rating` and

`get\_top\_performing\_employee`, for calculating employee performance ratings and identifying the top-performing employee.

1. It contains one PL/SQL triggers, `calculate\_performance\_rating` and

`get\_top\_performing\_employee`, for calculating employee performance ratings and identifying the top-performing employee.

1. It contains three PL/SQL procedures, `calculate\_performance\_rating` and

`get\_top\_performing\_employee`, for calculating employee performance ratings and identifying the top-performing employee.

1. It contains four PL /SQL function, `employee\_performance`, and one PL/SQL procedure,

`employee\_performance`, for calculating employee performance ratings and identifying the top- performing employee.

\*\*Correct Option:\*\* A

\*\*Ques14 - Consider the following PL/SQL package specification: (Difficulty level – Easy)\*\*

```plsql

CREATE OR REPLACE PACKAGE employee\_info AS

FUNCTION get\_employee\_name(emp\_id NUMBER) RETURN VARCHAR2; FUNCTION get\_employee\_salary(emp\_id NUMBER) RETURN NUMBER;

END employee\_info;

/

```

\*\*What does the PL/SQL package `employee\_info` contain?\*\*

1. It contains four PL/SQL functions, `get\_employee\_name` and `get\_employee\_salary`.
2. It contains PL/SQL triggers, `get\_employee\_name` and `get\_employee\_salary`.
3. It contains two PL/SQL procedures, `get\_employee\_name` and `get\_employee\_salary`.
4. It contains one PL/SQL function, `employee\_info`, and one PL/SQL procedure, `employee\_info`.

\*\*Correct Option:\*\* A

---

\*\*Ques15 - Consider the following PL/SQL package specification: (Difficulty level – Easy)\*\*

```plsql

CREATE OR REPLACE PACKAGE math\_operations AS

FUNCTION add\_numbers(num1 NUMBER, num2 NUMBER) RETURN NUMBER;

FUNCTION subtract\_numbers(num1 NUMBER, num2 NUMBER) RETURN NUMBER;

END math\_operations;

/

```

\*\*What does the PL/SQL package `math\_operations` contain?\*\*

1. It contains two PL/SQL functions, `add\_numbers` and `subtract\_numbers`, for performing mathematical operations.
2. It contains two PL/SQL triggers, `add\_numbers` and `subtract\_numbers`, for performing mathematical operations.
3. It contains two PL/SQL procedures, `add\_numbers` and `subtract\_numbers`, for performing mathematical operations.
4. It contains one PL/SQL function, `math\_operations`, and one PL/SQL procedure,

`math\_operations`, for performing mathematical operations.

\*\*Correct Option:\*\* A

\*\*Ques1 - Consider the following PL/SQL function: (Difficulty level – Easy)\*\*

```plsql

CREATE OR REPLACE FUNCTION calculate\_area(length NUMBER, width NUMBER)

RETURN NUMBER IS

area NUMBER;

BEGIN

area := length \* width; RETURN area;

END;

```

\*\*What does the PL/SQL function `calculate\_area` do?\*\*

1. It calculates the perimeter of a rectangle.
2. It calculates the area of a rectangle.
3. It calculates the volume of a rectangle.
4. It calculates the diagonal length of a rectangle.

\*\*Correct Option:\*\* B

---

\*\*Ques7 - Consider the following PL/SQL function: (Difficulty level – Easy)\*\*

```plsql

CREATE OR REPLACE FUNCTION get\_grade(score NUMBER)

RETURN VARCHAR2 IS grade VARCHAR2(2);

BEGIN

IF score >= 90 THEN grade := 'A';

ELSIF score >= 80 THEN

grade := 'B';

ELSIF score >= 70 THEN

grade := 'C'; ELSE

grade := 'D'; END IF;

RETURN grade; END;

```

\*\*What does the PL/SQL function `get\_grade` do?\*\*

1. It calculates the square root of the input `score`.
2. It calculates the average of multiple scores.
3. It assigns a grade ('A', 'B', 'C', or 'D') based on the input `score`.
4. It calculates the factorial of the input `score`.

\*\*Correct Option:\*\* C

---

\*\*Ques2 - Consider the following PL/SQL function: (Difficulty level – Easy)\*\*

```plsql

CREATE OR REPLACE FUNCTION is\_positive(num NUMBER) RETURN BOOLEAN IS

BEGIN

IF num > 0 THEN RETURN TRUE;

ELSE

RETURN FALSE; END IF;

END;

```

\*\*What does the PL/SQL function `is\_positive` do?\*\*

1. It checks if the input `num` is a positive number and returns `TRUE` if it is, `FALSE` otherwise.
2. It checks if the input `num` is an even number and returns `TRUE` if it is, `FALSE` otherwise.
3. It calculates the square of the input `num`.
4. It calculates the factorial of the input `num`.

\*\*Correct Option:\*\* A

---

\*\*Ques3 - Consider the following PL/SQL function: (Difficulty level – Easy)\*\*

```plsql

CREATE OR REPLACE FUNCTION reverse\_string(input\_str VARCHAR2)

RETURN VARCHAR2 IS

reversed\_str VARCHAR2(255); BEGIN

SELECT REVERSE(input\_str) INTO reversed\_str FROM DUAL; RETURN reversed\_str;

END;

```

\*\*What does the PL/SQL function `reverse\_string` do?\*\*

1. It calculates the length of the input string `input\_str`.
2. It calculates the square root of the input number `input\_str`.
3. It reverses the characters in the input string `input\_str`.
4. It calculates the factorial of the input number `input\_str`.

\*\*Correct Option:\*\* C

---

\*\*Ques4 - Consider the following PL/SQL function: (Difficulty level – Easy)\*\*

```plsql

CREATE OR REPLACE FUNCTION find\_maximum(a NUMBER, b NUMBER) RETURN NUMBER IS

max\_val NUMBER; BEGIN

IF a > b THEN

max\_val := a; ELSE

max\_val := b; END IF;

RETURN max\_val; END;

```

\*\*What does the PL/SQL function `find\_maximum` do?\*\*

1. It calculates the average of two numbers.
2. It calculates the sum of two numbers.
3. It calculates the maximum value between two numbers.
4. It calculates the factorial of two numbers.

\*\*Correct Option:\*\* C

\*\*Ques5 - Consider the following PL/SQL function: (Difficulty level – Easy)\*\*

```plsql

CREATE OR REPLACE FUNCTION calculate\_discount(amount NUMBER) RETURN NUMBER IS

discount NUMBER; BEGIN

IF amount >= 1000 THEN discount := 0.1 \* amount;

ELSE

discount := 0;

END IF;

RETURN discount; END;

```

\*\*What does the PL/SQL function `calculate\_discount` do?\*\*

1. It calculates the total cost after applying a discount of 10%.
2. It calculates the total cost without any discount.
3. It calculates the total cost after applying a discount of 1%.
4. It calculates the total cost after applying a discount of 5%.

\*\*Correct Option:\*\* A

---

\*\*Ques6 - Consider the following PL/SQL function: (Difficulty level – Easy)\*\*

```plsql

CREATE OR REPLACE FUNCTION is\_vowel(character CHAR) RETURN BOOLEAN IS

BEGIN

IF character IN ('A', 'E', 'I', 'O', 'U', 'a', 'e', 'i', 'o', 'u') THEN

RETURN TRUE; ELSE

RETURN FALSE; END IF;

END;

```

\*\*What does the PL/SQL function `is\_vowel` do?\*\*

1. It checks if the input character is a consonant and returns `TRUE` if it is, `FALSE` otherwise.
2. It checks if the input character is a digit and returns `TRUE` if it is, `FALSE` otherwise.
3. It checks if the input character is a vowel and returns `TRUE` if it is, `FALSE` otherwise.
4. It calculates the square root of the input character.

\*\*Correct Option:\*\* C

---

\*\*Ques7 - Consider the following PL/SQL function: (Difficulty level – Easy)\*\*

```plsql

CREATE OR REPLACE FUNCTION find\_length(input\_str VARCHAR2) RETURN NUMBER IS

length NUMBER; BEGIN

SELECT LENGTH(input\_str) INTO length FROM DUAL; RETURN length;

END;

```

\*\*What does the PL/SQL function `find\_length` do?\*\*

1. It calculates the factorial of the length of the input string `input\_str`.
2. It calculates the square root of the length of the input string `input\_str`.
3. It retrieves the length of the input string `input\_str`.
4. It checks if the length of the input string `input\_str` is even and returns `TRUE` if it is, `FALSE` otherwise.

\*\*Correct Option:\*\* C

---

\*\*Ques8 - Consider the following PL/SQL function: (Difficulty level – Easy)\*\*

```plsql

CREATE OR REPLACE FUNCTION calculate\_average(num1 NUMBER, num2 NUMBER)

RETURN NUMBER IS

average NUMBER; BEGIN

average := (num1 + num2) / 2; RETURN average;

END;

```

\*\*What does the PL/SQL function `calculate\_average` do?\*\*

1. It calculates the sum of two numbers.
2. It calculates the product of two numbers.
3. It calculates the average of two numbers.
4. It calculates the square root of two numbers.

\*\*Correct Option:\*\* C

---

\*\*Ques9 - Consider the following PL/SQL function: (Difficulty level – Easy)\*\*

```plsql

CREATE OR REPLACE FUNCTION is\_positive\_or\_zero(num NUMBER) RETURN BOOLEAN IS

BEGIN

IF num >= 0 THEN RETURN TRUE;

ELSE

RETURN FALSE; END IF;

END;

```

\*\*What does the PL/SQL function `is\_positive\_or\_zero` do?\*\*

1. It checks if the input `num` is a positive number and returns `TRUE` if it is, `FALSE` otherwise.
2. It checks if the input `num` is an even number and returns `TRUE` if it is, `FALSE` otherwise.
3. It checks if the input `num` is a non-negative number and returns `TRUE` if it is, `FALSE` otherwise.
4. It calculates the factorial of the input `num`.

\*\*Correct Option:\*\* C

---

\*\*Ques10 - Consider the following PL/SQL function: (Difficulty level – Easy)\*\*

```plsql

CREATE OR REPLACE FUNCTION generate\_greeting(name VARCHAR2)

RETURN VARCHAR2 IS

greeting VARCHAR2(100); BEGIN

greeting := 'Hi there, ' || name || '!'; RETURN greeting;

END;

```

\*\*What does the PL/SQL function `generate\_greeting` do?\*\*

1. It calculates the length of the input string `name`.
2. It calculates the square of a numeric input.
3. It generates a friendly greeting message with the input `name`.
4. It calculates the factorial of a numeric input.

\*\*Correct Option:\*\* C

\*\*Ques11 - Consider the following PL/SQL function: (Difficulty level – Medium)\*\*

```plsql

CREATE OR REPLACE FUNCTION calculate\_factorial(n NUMBER) RETURN NUMBER IS

result NUMBER := 1; BEGIN

IF n < 0 THEN RETURN NULL;

ELSIF n = 0 THEN RETURN 1;

ELSE

FOR i IN 1..n LOOP

result := result \* i; END LOOP;

END IF;

RETURN result; END;

```

\*\*What does the PL/SQL function `calculate\_factorial` do?\*\*

1. It calculates the factorial of a non-negative integer `n`.
2. It calculates the square root of the input number `n`.
3. It calculates the average of multiple numbers.
4. It calculates the sum of all integers from 1 to `n`.

\*\*Correct Option:\*\* A

---

\*\*Ques12 - Consider the following PL/SQL function: (Difficulty level – Medium)\*\*

```plsql

CREATE OR REPLACE FUNCTION calculate\_fibonacci(n NUMBER) RETURN NUMBER IS

a NUMBER := 0; b NUMBER := 1;

result NUMBER := 0;

BEGIN

IF n <= 0 THEN RETURN 0;

ELSIF n = 1 THEN RETURN 1;

ELSE

FOR i IN 2..n LOOP

result := a + b; a := b;

b := result; END LOOP;

END IF;

RETURN result; END;

```

\*\*What does the PL/SQL function `calculate\_fibonacci` do?\*\*

1. It calculates the sum of the first `n` Fibonacci numbers.
2. It calculates the square root of the input number `n`.
3. It calculates the factorial of the input number `n`.
4. It calculates the `n`-th Fibonacci number.

\*\*Correct Option:\*\* D

---

\*\*Ques13 - Consider the following PL/SQL function: (Difficulty level – Medium)\*\*

```plsql

CREATE OR REPLACE FUNCTION calculate\_power(base NUMBER, exponent NUMBER)

RETURN NUMBER IS

result NUMBER := 1; BEGIN

IF exponent < 0 THEN RETURN NULL;

ELSE

FOR i IN 1..exponent LOOP result := result \* base;

END LOOP; END IF;

RETURN result; END;

```

\*\*What does the PL/SQL function `calculate\_power` do?\*\*

1. It calculates the product of `base` and `exponent`.
2. It calculates the square root of `base` raised to the power of `exponent`.
3. It calculates the factorial of `exponent`.
4. It calculates `base` raised to the power of `exponent`.

\*\*Correct Option:\*\* D

---

\*\*Ques14 - Consider the following PL/SQL function: (Difficulty level – Medium)\*\*

```plsql

CREATE OR REPLACE FUNCTION is\_palindrome(word VARCHAR2) RETURN BOOLEAN IS

reversed\_word VARCHAR2(255); BEGIN

reversed\_word := REVERSE(word); IF word = reversed\_word THEN

RETURN TRUE; ELSE

RETURN FALSE; END IF;

END;

```

\*\*What does the PL/SQL function `is\_palindrome` do?\*\*

1. It checks if the input string `word` is a palindrome (reads the same forwards and backwards) and returns `TRUE` if it is, `FALSE` otherwise.
2. It calculates the length of the input string `word`.
3. It calculates the square root of the input number `word`.
4. It checks if the input string `word` contains any digits and returns `TRUE` if it does, `FALSE` otherwise.

\*\*Correct Option:\*\* A

---

\*\*Ques25 - Consider the following PL/SQL function: (Difficulty level – Medium)\*\*

```plsql

CREATE OR REPLACE FUNCTION get\_employee\_salary(emp\_id NUMBER) RETURN NUMBER IS

salary NUMBER; BEGIN

-- Retrieve the salary of the employee with the given emp\_id

SELECT salary INTO salary FROM employees WHERE employee\_id

= emp\_id;

IF SQL%FOUND THEN

RETURN salary; ELSE

RETURN NULL; END IF;

END;

```

\*\*What does the PL/SQL function `get\_employee\_salary` do?\*\*

1. It calculates the average salary of all employees.
2. It retrieves the salary of the employee with the specified `emp\_id`.
3. It calculates the total salary of all employees.
4. It retrieves the highest salary among all employees.

\*\*Correct Option:\*\* B

---

\*\*Ques15 - Consider the following PL/SQL function: (Difficulty level – Medium)\*\*

```plsql

CREATE OR REPLACE FUNCTION count\_words(sentence VARCHAR2) RETURN NUMBER IS

word\_count NUMBER := 0; BEGIN

FOR i IN 1..LENGTH(sentence) LOOP

IF SUBSTR(sentence, i, 1) = ' ' THEN word\_count := word\_count + 1;

END IF; END LOOP;

-- Add one to count the last word word\_count := word\_count + 1; RETURN word\_count;

END;

```

\*\*What does the PL/SQL function `count\_words` do?\*\*

1. It calculates the number of characters in the input sentence.
2. It calculates the number of words in the input sentence.
3. It calculates the number of vowels in the input sentence.
4. It calculates the number of digits in the input sentence.

\*\*Correct Option:\*\* B

---

2 mark questions –

\*\*Ques1 - Consider the following PL/SQL package specification: (Difficulty level – Medium)\*\*

```plsql

CREATE OR REPLACE PACKAGE product\_discounts AS

FUNCTION calculate\_discount(product\_id NUMBER, quantity NUMBER) RETURN NUMBER;

FUNCTION apply\_discount\_to\_order(order\_id NUMBER) RETURN BOOLEAN;

END product\_discounts;

/

```

\*\*What does the PL/SQL package `product\_discounts` contain?\*\*

1. It contains two PL/SQL functions, `calculate\_discount` and `apply\_discount\_to\_order`, for calculating and applying product discounts.
2. It contains two PL/SQL triggers, `calculate\_discount` and `apply\_discount\_to\_order`, for calculating and applying product discounts.
3. It contains two PL/SQL procedures, `calculate\_discount` and `apply\_discount\_to\_order`, for calculating and applying product discounts.
4. It contains one PL/SQL function, `product\_discounts`, and one PL/SQL procedure,

`product\_discounts`, for calculating and applying product discounts.

\*\*Correct Option:\*\* A

---

\*\*Ques2 - Consider the following PL/SQL package specification: (Difficulty level – Medium)\*\*

```plsql

CREATE OR REPLACE PACKAGE order\_processing AS

FUNCTION process\_order(order\_id NUMBER) RETURN BOOLEAN;

FUNCTION validate\_payment(order\_id NUMBER) RETURN BOOLEAN; END order\_processing;

/

```

\*\*What does the PL/SQL package `order\_processing` contain?\*\*

1. It contains two PL/SQL functions, `process\_order` and `validate\_payment`, for processing orders and validating payments.
2. It contains two PL/SQL triggers, `process\_order` and `validate\_payment`, for processing orders and validating payments.
3. It contains two PL/SQL procedures, `process\_order` and `validate\_payment`, for processing orders and validating payments.
4. It contains one PL/SQL function, `order\_processing`, and one PL/SQL procedure,

`order\_processing`, for processing orders and validating payments.

\*\*Correct Option:\*\* A

---

\*\*Ques3 - Consider the following PL/SQL package specification: (Difficulty level – Medium)\*\*

```plsql

CREATE OR REPLACE PACKAGE employee\_management AS FUNCTION hire\_employee(name VARCHAR2, salary NUMBER)

RETURN NUMBER;

FUNCTION terminate\_employee(employee\_id NUMBER) RETURN BOOLEAN;

END employee\_management;

/

```

\*\*What does the PL/SQL package `employee\_management` contain?\*\*

1. It contains two PL/SQL functions, `hire\_employee` and `terminate\_employee`, for hiring and terminating employees.
2. It contains two PL/SQL triggers, `hire\_employee` and `terminate\_employee`, for hiring and terminating employees.
3. It contains two PL/SQL procedures, `hire\_employee` and `terminate\_employee`, for hiring and terminating employees.
4. It contains one PL/SQL function, `employee\_management`, and one PL/SQL procedure,

`employee\_management`, for hiring and terminating employees.

\*\*Correct Option:\*\* A

---

\*\*Ques4 - Consider the following PL/SQL package specification: (Difficulty level – Medium)\*\*

```plsql

CREATE OR REPLACE PACKAGE order\_management AS

FUNCTION create\_order(customer\_id NUMBER, total\_amount NUMBER) RETURN NUMBER;

FUNCTION cancel\_order(order\_id NUMBER) RETURN BOOLEAN; END order\_management;

/

```

\*\*What does the PL/SQL package `order\_management` contain?\*\*

1. It contains two PL/SQL functions, `create\_order` and `cancel\_order`, for creating and canceling orders.
2. It contains two PL/SQL triggers, `create\_order` and `cancel\_order`, for creating and canceling orders.
3. It contains two PL/SQL procedures, `create\_order` and `cancel\_order`, for creating and canceling orders.
4. It contains one PL/SQL function, `order\_management`, and one PL/SQL procedure,

`order\_management`, for creating and canceling orders.

\*\*Correct Option:\*\* A

\*\*Ques5 - Consider the following PL/SQL package specification: (Difficulty level – Hard)\*\*

```plsql

CREATE OR REPLACE PACKAGE order\_tracking AS

FUNCTION track\_order(order\_id NUMBER) RETURN VARCHAR2; FUNCTION estimate\_delivery\_time(order\_id NUMBER) RETURN

NUMBER;

FUNCTION get\_order\_status(order\_id NUMBER) RETURN VARCHAR2;

END order\_tracking;

/

```

\*\*What does the PL/SQL package `order\_tracking` contain?\*\*

1. It contains three PL/SQL functions, `track\_order`, `estimate\_delivery\_time`, and

`get\_order\_status`, for tracking orders and estimating delivery times.

1. It contains three PL/SQL triggers, `track\_order`, `estimate\_delivery\_time`, and `get\_order\_status`, for tracking orders and estimating delivery times.
2. It contains three PL/SQL procedures, `track\_order`, `estimate\_delivery\_time`, and

`get\_order\_status`, for tracking orders and estimating delivery times.

1. It contains one PL/SQL function, `order\_tracking`, and one PL/SQL procedure, `order\_tracking`, for tracking orders and estimating delivery times.

\*\*Correct Option:\*\* A

---

---

\*\*Ques6 - Consider the following PL/SQL package specification: (Difficulty level – Hard)\*\*

```plsql

CREATE OR REPLACE PACKAGE project\_management AS

FUNCTION allocate\_resources(project\_id NUMBER, resource\_id NUMBER, hours NUMBER) RETURN BOOLEAN;

FUNCTION get\_project\_status(project\_id NUMBER) RETURN VARCHAR2;

END project\_management;

/

```

\*\*What does the PL/SQL package `project\_management` contain?\*\*

1. It contains two PL/SQL functions, `allocate\_resources` and `get\_project\_status`, for resource allocation and project status retrieval.
2. It contains two PL/SQL triggers, `allocate\_resources` and `get\_project\_status`, for resource allocation and project status retrieval.
3. It contains two PL/SQL procedures, `allocate\_resources` and `get\_project\_status`, for resource allocation and project status retrieval.
4. It contains one PL/SQL function, `project\_management`, and one PL/SQL procedure,

`project\_management`, for resource allocation and project status retrieval.

\*\*Correct Option:\*\* A

---

\*\*Ques7 - Consider the following PL/SQL package specification: (Difficulty level – Hard)\*\*

```plsql

CREATE OR REPLACE PACKAGE student\_grading AS FUNCTION calculate\_final\_grade(student\_id NUMBER,

course\_id NUMBER) RETURN CHAR;

FUNCTION get\_student\_ranking(course\_id NUMBER) RETURN NUMBER;

END student\_grading;

/

```

\*\*What does the PL/SQL package `student\_grading` contain?\*\*

1. It contains two PL/SQL functions, `calculate\_final\_grade` and `get\_student\_ranking`, for calculating student grades and retrieving student rankings in a course.
2. It contains two PL/SQL triggers, `calculate\_final\_grade` and `get\_student\_ranking`, for calculating student grades and retrieving student rankings in a course.
3. It contains two PL/SQL procedures, `calculate\_final\_grade` and `get\_student\_ranking`, for calculating student grades and retrieving student rankings in a course.
4. It contains one PL/SQL function, `student\_grading`, and one PL/SQL procedure, `student\_grading`, for calculating student grades and retrieving student rankings in a course.

\*\*Correct Option:\*\* A

---

\*\*Ques8 - Consider the following PL/SQL package specification: (Difficulty level – Hard)\*\*

```plsql

CREATE OR REPLACE PACKAGE medical\_records AS

FUNCTION get\_patient\_history(patient\_id NUMBER) RETURN CLOB;

FUNCTION analyze\_patient\_data(patient\_id NUMBER) RETURN CLOB;

END medical\_records;

/

```

\*\*What does the PL/SQL package `medical\_records` contain?\*\*

1. It contains two PL/SQL functions, `get\_patient\_history` and `analyze\_patient\_data`, for retrieving patient medical history and analyzing patient data.
2. It contains two PL/SQL triggers, `get\_patient\_history` and `analyze\_patient\_data`, for retrieving patient medical history and analyzing patient data.
3. It contains two PL/SQL procedures, `get\_patient\_history` and `analyze\_patient\_data`, for retrieving patient medical history and analyzing patient data.
4. It contains one PL/SQL function, `medical\_records`, and one PL/SQL procedure, `medical\_records`, for retrieving patient medical history and analyzing patient data.

\*\*Correct Option:\*\* A

\*\*Ques9 - Consider the following PL/SQL package specification: (Difficulty level – Hard)\*\*

```plsql

CREATE OR REPLACE PACKAGE order\_tracking AS

FUNCTION track\_order(order\_id NUMBER) RETURN VARCHAR2; FUNCTION estimate\_delivery\_time(order\_id NUMBER) RETURN

NUMBER;

FUNCTION get\_order\_status(order\_id NUMBER) RETURN VARCHAR2;

END order\_tracking;

/

```

\*\*What does the PL/SQL package `order\_tracking` contain?\*\*

1. It contains three PL/SQL functions, `track\_order`, `estimate\_delivery\_time`, and

`get\_order\_status`, for tracking orders and estimating delivery times.

1. It contains three PL/SQL triggers, `track\_order`, `estimate\_delivery\_time`, and `get\_order\_status`, for tracking orders and estimating delivery times.
2. It contains three PL/SQL procedures, `track\_order`, `estimate\_delivery\_time`, and

`get\_order\_status`, for tracking orders and estimating delivery times.

1. It contains one PL/SQL function, `order\_tracking`, and one PL/SQL procedure, `order\_tracking`, for tracking orders and estimating delivery times.

\*\*Correct Option:\*\* A

---

---

\*\*Ques10 - Consider the following PL/SQL package specification: (Difficulty level – Hard)\*\*

```plsql

CREATE OR REPLACE PACKAGE project\_management AS

FUNCTION allocate\_resources(project\_id NUMBER, resource\_id NUMBER, hours NUMBER) RETURN BOOLEAN;

FUNCTION get\_project\_status(project\_id NUMBER) RETURN VARCHAR2;

END project\_management;

/

```

\*\*What does the PL/SQL package `project\_management` contain?\*\*

1. It contains two PL/SQL functions, `allocate\_resources` and `get\_project\_status`, for resource allocation and project status retrieval.
2. It contains two PL/SQL triggers, `allocate\_resources` and `get\_project\_status`, for resource allocation and project status retrieval.
3. It contains two PL/SQL procedures, `allocate\_resources` and `get\_project\_status`, for resource allocation and project status retrieval.
4. It contains one PL/SQL function, `project\_management`, and one PL/SQL procedure,

`project\_management`, for resource allocation and project status retrieval.

\*\*Correct Option:\*\* A

---

\*\*Ques11 - Consider the following PL/SQL package specification: (Difficulty level – Hard)\*\*

```plsql

CREATE OR REPLACE PACKAGE student\_grading AS FUNCTION calculate\_final\_grade(student\_id NUMBER,

course\_id NUMBER) RETURN CHAR;

FUNCTION get\_student\_ranking(course\_id NUMBER) RETURN NUMBER;

END student\_grading;

/

```

\*\*What does the PL/SQL package `student\_grading` contain?\*\*

1. It contains two PL/SQL functions, `calculate\_final\_grade` and `get\_student\_ranking`, for calculating student grades and retrieving student rankings in a course.
2. It contains two PL/SQL triggers, `calculate\_final\_grade` and `get\_student\_ranking`, for calculating student grades and retrieving student rankings in a course.
3. It contains two PL/SQL procedures, `calculate\_final\_grade` and `get\_student\_ranking`, for calculating student grades and retrieving student rankings in a course.
4. It contains one PL/SQL function, `student\_grading`, and one PL/SQL procedure, `student\_grading`, for calculating student grades and retrieving student rankings in a course.

\*\*Correct Option:\*\* A

---

\*\*Ques12 - Consider the following PL/SQL package specification: (Difficulty level – Hard)\*\*

```plsql

CREATE OR REPLACE PACKAGE medical\_records AS

FUNCTION get\_patient\_history(patient\_id NUMBER) RETURN CLOB;

FUNCTION analyze\_patient\_data(patient\_id NUMBER) RETURN CLOB;

END medical\_records;

/

```

\*\*What does the PL/SQL package `medical\_records` contain?\*\*

1. It contains two PL/SQL functions, `get\_patient\_history` and `analyze\_patient\_data`, for retrieving patient medical history and analyzing patient data.
2. It contains two PL/SQL triggers, `get\_patient\_history` and `analyze\_patient\_data`, for retrieving patient medical history and analyzing patient data.
3. It contains two PL/SQL procedures, `get\_patient\_history` and `analyze\_patient\_data`, for retrieving patient medical history and analyzing patient data.
4. It contains one PL/SQL function, `medical\_records`, and one PL/SQL procedure, `medical\_records`, for retrieving patient medical history and analyzing patient data.

\*\*Correct Option:\*\* A

\*\*Ques13 - Consider the following SQL cursor declaration: (Difficulty level – Easy)\*\*

```sql

DECLARE

emp\_cursor CURSOR FOR

SELECT employee\_name FROM employees;

```

\*\*What does the SQL cursor `emp\_cursor` do?\*\*

1. It retrieves all columns from the `employees` table.
2. It retrieves the `employee\_name` column from the `employees` table.
3. It updates the `employee\_name` column in the `employees` table.
4. It deletes records from the `employees` table.

\*\*Correct Option:\*\* B

---

\*\*Ques14 - Consider the following SQL cursor declaration: (Difficulty level – Easy)\*\*

```sql

DECLARE

product\_cursor CURSOR FOR

SELECT product\_name, product\_price FROM products;

```

\*\*What does the SQL cursor `product\_cursor` do?\*\*

1. It retrieves all columns from the `products` table.
2. It retrieves the `product\_name` and `product\_price` columns from the `products` table.
3. It updates the `product\_name` and `product\_price` columns in the `products` table.
4. It deletes records from the `products` table.

\*\*Correct Option:\*\* B

\*\*Ques15 - Consider the following SQL cursor declaration: (Difficulty level – Easy)\*\*

```sql

DECLARE

order\_cursor CURSOR FOR

SELECT order\_id, order\_date FROM orders;

```

\*\*What does the SQL cursor `order\_cursor` do?\*\*

1. It retrieves all columns from the `orders` table.
2. It retrieves the `order\_id` and `order\_date` columns from the `orders` table.
3. It updates the `order\_id` and `order\_date` columns in the `orders` table.
4. It deletes records from the `orders` table.

\*\*Correct Option:\*\* B

\*\*Ques1 - Consider the following SQL cursor declaration: (Difficulty level – Easy)\*\*

```sql

DECLARE

customer\_cursor CURSOR FOR

SELECT customer\_name FROM customers;

```

\*\*What does the SQL cursor `customer\_cursor` do?\*\*

1. It retrieves all columns from the `customers` table.
2. It retrieves the `customer\_name` column from the `customers` table.
3. It updates the `customer\_name` column in the `customers` table.
4. It deletes records from the `customers` table.

\*\*Correct Option:\*\* B

---

\*\*Ques2 - Consider the following SQL cursor declaration: (Difficulty level – Easy)\*\*

```sql

DECLARE

employee\_cursor CURSOR FOR

SELECT employee\_id, employee\_name FROM employees;

```

\*\*What does the SQL cursor `employee\_cursor` do?\*\*

1. It retrieves all columns from the `employees` table.
2. It retrieves the `employee\_id` and `employee\_name` columns from the `employees` table.
3. It updates the `employee\_id` and `employee\_name` columns in the `employees` table.
4. It deletes records from the `employees` table.

\*\*Correct Option:\*\* B

---

\*\*Ques3 - Consider the following SQL cursor declaration: (Difficulty level – Easy)\*\*

```sql

DECLARE

product\_cursor CURSOR FOR

SELECT product\_id FROM products WHERE product\_price >

100;

```

\*\*What does the SQL cursor `product\_cursor` do?\*\*

1. It retrieves all columns from the `products` table.
2. It retrieves the `product\_id` column from the `products` table for products with a price greater than 100.
3. It updates the `product\_id` column in the `products` table.
4. It deletes records from the `products` table.

\*\*Correct Option:\*\* B

---

\*\*Ques4 - Consider the following SQL cursor declaration: (Difficulty level – Easy)\*\*

```sql

DECLARE

order\_cursor CURSOR FOR

SELECT order\_date FROM orders WHERE order\_status = 'Shipped';

```

\*\*What does the SQL cursor `order\_cursor` do?\*\*

1. It retrieves all columns from the `orders` table.
2. It retrieves the `order\_date` column from the `orders` table for orders with a status of 'Shipped'.
3. It updates the `order\_date` column in the `orders` table.
4. It deletes records from the `orders` table.

\*\*Correct Option:\*\* B

---

\*\*Ques5 - Consider the following SQL cursor declaration: (Difficulty level – Easy)\*\*

```sql

DECLARE

customer\_cursor CURSOR FOR

SELECT customer\_id FROM customers WHERE registration\_date >= '2023-01-01';

```

\*\*What does the SQL cursor `customer\_cursor` do?\*\*

1. It retrieves all columns from the `customers` table.
2. It retrieves the `customer\_id` column from the `customers` table for customers registered on or after January 1, 2023.
3. It updates the `customer\_id` column in the `customers` table.
4. It deletes records from the `customers` table.

\*\*Correct Option:\*\* B

---

\*\*Ques6 - Consider the following SQL cursor declaration: (Difficulty level – Easy)\*\*

```sql

DECLARE

employee\_cursor CURSOR FOR

SELECT department\_id, COUNT(\*) FROM employees GROUP BY department\_id;

```

\*\*What does the SQL cursor `employee\_cursor` do?\*\*

1. It retrieves all columns from the `employees` table.
2. It retrieves the `department\_id` and the count of employees in each department from the

`employees` table.

1. It updates the `department\_id` and employee counts in the `employees` table.
2. It deletes records from the `employees` table.

\*\*Correct Option:\*\* B

\*\*Ques7 - Consider the following SQL cursor declaration: (Difficulty level – Easy)\*\*

```sql

DECLARE

product\_cursor CURSOR FOR

SELECT product\_name, product\_category FROM products WHERE product\_category = 'Electronics';

```

\*\*What does the SQL cursor `product\_cursor` do?\*\*

1. It retrieves all columns from the `products` table.
2. It retrieves the `product\_name` and `product\_category` columns from the `products` table for products in the 'Electronics' category.
3. It updates the `product\_name` and `product\_category` columns in the `products` table.
4. It deletes records from the `products` table.

\*\*Correct Option:\*\* B

\*\*Ques8 - Consider the following SQL cursor declaration: (Difficulty level – Hard)\*\*

```sql

DECLARE

employee\_cursor CURSOR FOR

SELECT employee\_id, employee\_name, department\_id FROM employees

WHERE salary > (SELECT AVG(salary) FROM employees);

```

\*\*What does the SQL cursor `employee\_cursor` do?\*\*

1. It retrieves all columns from the `employees` table.
2. It retrieves the `employee\_id`, `employee\_name`, and `department\_id` columns from the

`employees` table for employees with salaries above the average salary in the company.

1. It updates the `employee\_id`, `employee\_name`, and `department\_id` columns in the

`employees` table.

1. It deletes records from the `employees` table.

\*\*Correct Option:\*\* B

---

\*\*Ques9 - Consider the following SQL cursor declaration: (Difficulty level – Hard)\*\*

```sql

DECLARE

order\_cursor CURSOR FOR

SELECT order\_id, customer\_id, order\_date FROM orders

WHERE EXISTS (SELECT 1 FROM order\_items WHERE

order\_items.order\_id = orders.order\_id);

```

\*\*What does the SQL cursor `order\_cursor` do?\*\*

1. It retrieves all columns from the `orders` table.
2. It retrieves the `order\_id`, `customer\_id`, and `order\_date` columns from the `orders` table for orders that have associated order items.
3. It updates the `order\_id`, `customer\_id`, and `order\_date` columns in the `orders` table.
4. It deletes records from the `orders` table.

\*\*Correct Option:\*\* B

---

\*\*Ques10 - Consider the following SQL cursor declaration: (Difficulty level – Hard)\*\*

```sql

DECLARE

customer\_cursor CURSOR FOR

SELECT customer\_id, COUNT(\*) AS order\_count FROM orders

GROUP BY customer\_id HAVING COUNT(\*) > 5;

```

\*\*What does the SQL cursor `customer\_cursor` do?\*\*

1. It retrieves all columns from the `orders` table.
2. It retrieves the `customer\_id` and the count of orders placed by each customer from the `orders` table for customers who have placed more than 5 orders.
3. It updates the `customer\_id` and order counts in the `orders` table.
4. It deletes records from the `orders` table.

\*\*Correct Option:\*\* B

---

\*\*Ques11 - Consider the following SQL cursor declaration: (Difficulty level – Hard)\*\*

```sql

DECLARE

product\_cursor CURSOR FOR

SELECT product\_id, product\_name, product\_price FROM products

WHERE product\_id IN (SELECT product\_id FROM order\_items GROUP BY product\_id HAVING COUNT(\*) >= 10);

```

\*\*What does the SQL cursor `product\_cursor` do?\*\*

1. It retrieves all columns from the `products` table.
2. It retrieves the `product\_id`, `product\_name`, and `product\_price` columns from the `products` table for products that have been ordered at least 10 times.
3. It updates the `product\_id`, `product\_name`, and `product\_price` columns in the `products` table.
4. It deletes records from the `products` table.

\*\*Correct Option:\*\* B

---

\*\*Ques12 - Consider the following SQL cursor declaration: (Difficulty level – Hard)\*\*

```sql

DECLARE

order\_cursor CURSOR FOR

SELECT order\_id, order\_date, SUM(order\_total) AS total\_amount

FROM orders

WHERE order\_status = 'Shipped' GROUP BY order\_id, order\_date HAVING SUM(order\_total) > 1000;

```

\*\*What does the SQL cursor `order\_cursor` do?\*\*

1. It retrieves all columns from the `orders` table.
2. It retrieves the `order\_id`, `order\_date`, and total order amount columns from the `orders` table for shipped orders with a total amount greater than 1000.
3. It updates the `order\_id`, `order\_date`, and total order amount columns in the `orders` table.
4. It deletes records from the `orders` table.

\*\*Correct Option:\*\* B

\*\*Ques13 - Consider the following SQL trigger: (Difficulty level – Hard)\*\*

```sql

CREATE OR REPLACE TRIGGER update\_salary\_trigger BEFORE UPDATE ON employees

FOR EACH ROW BEGIN

IF :NEW.salary > :OLD.salary THEN

INSERT INTO salary\_history (employee\_id, old\_salary, new\_salary, change\_date)

VALUES (:OLD.employee\_id, :OLD.salary, :NEW.salary, SYSDATE);

END IF; END;

/

```

\*\*What does the SQL trigger `update\_salary\_trigger` do?\*\*

1. It updates the salary of all employees in the `employees` table.
2. It inserts a record into the `salary\_history` table whenever an employee's salary is increased.
3. It deletes records from the `employees` table whenever an employee's salary is updated.
4. It calculates the average salary of all employees.

\*\*Correct Option:\*\* B

---

\*\*Ques14 - Consider the following SQL trigger: (Difficulty level – Hard)\*\*

```sql

CREATE OR REPLACE TRIGGER audit\_employee\_delete AFTER DELETE ON employees

FOR EACH ROW BEGIN

INSERT INTO audit\_log (event\_type, event\_date, username, details)

VALUES ('Employee Deletion', SYSDATE, USER, 'Employee ID: ' || :OLD.employee\_id);

END;

/

```

\*\*What does the SQL trigger `audit\_employee\_delete` do?\*\*

1. It updates employee records in the `employees` table.
2. It inserts a record into the `audit\_log` table whenever an employee is deleted.
3. It inserts a record into the `employees` table whenever an employee is deleted.
4. It calculates the total number of employees in the `employees` table.

\*\*Correct Option:\*\* B

---

\*\*Ques15 - Consider the following SQL trigger: (Difficulty level – Hard)\*\*

```sql

CREATE OR REPLACE TRIGGER calculate\_avg\_salary AFTER INSERT OR DELETE ON employees

FOR EACH ROW BEGIN

DECLARE

total\_salary NUMBER; num\_employees NUMBER;

BEGIN

SELECT SUM(salary), COUNT(\*) INTO total\_salary, num\_employees FROM employees;

IF num\_employees > 0 THEN

INSERT INTO salary\_stats (average\_salary, total\_employees, calculation\_date)

VALUES (total\_salary / num\_employees, num\_employees, SYSDATE);

END IF; END;

END;

/

```

\*\*What does the SQL trigger `calculate\_avg\_salary` do?\*\*

1. It updates the salary of all employees in the `employees` table.
2. It calculates the average salary and total number of employees whenever a new employee is inserted or an employee is deleted.
3. It inserts a record into the `salary\_stats` table whenever an employee is deleted.
4. It calculates the total number of employees in the `employees` table.

\*\*Correct Option:\*\* B

\*\*Ques1 - Consider the following SQL trigger: (Difficulty level – Hard)\*\*

```sql

CREATE OR REPLACE TRIGGER prevent\_salary\_reduction BEFORE UPDATE ON employees

FOR EACH ROW BEGIN

IF :NEW.salary < :OLD.salary THEN RAISE\_APPLICATION\_ERROR (-20001, 'Salary reduction is

not allowed.'); END IF;

END;

/

```

\*\*What does the SQL trigger `prevent\_salary\_reduction` do?\*\*

1. It updates the salary of all employees in the `employees` table.
2. It prevents any attempt to reduce an employee's salary and raises a custom application error if such an update is detected.
3. It inserts a record into the `salary\_history` table whenever an employee's salary is increased.
4. It calculates the average salary of all employees.

\*\*Correct Option:\*\* B

---

\*\*Ques2 - Consider the following SQL trigger: (Difficulty level – Medium)\*\*

```sql

CREATE OR REPLACE TRIGGER audit\_table\_changes AFTER INSERT OR UPDATE OR DELETE ON employees DECLARE

change\_description VARCHAR2(500); BEGIN

change\_description := 'Table "employees" was '; IF INSERTING THEN

change\_description := change\_description || 'inserted into.';

ELSIF UPDATING THEN

change\_description := change\_description || 'updated.'; ELSIF DELETING THEN

change\_description := change\_description || 'deleted from.';

END IF;

INSERT INTO audit\_log (event\_type, event\_date, details) VALUES ('Table Change', SYSDATE, change\_description);

END;

/

```

\*\*What does the SQL trigger `audit\_table\_changes` do?\*\*

1. It updates the `employees` table whenever a change is made to it.
2. It inserts a record into the `audit\_log` table whenever a change (insert, update, or delete) is made to the `employees` table, including a description of the change.
3. It calculates the total number of employees in the `employees` table.
4. It deletes records from the `employees` table whenever a change is made to it.

\*\*Correct Option:\*\* B

---

\*\*Ques3 - Consider the following SQL trigger: (Difficulty level – Easy)\*\*

```sql

CREATE OR REPLACE TRIGGER enforce\_manager\_approval BEFORE INSERT ON purchase\_orders

FOR EACH ROW BEGIN

IF :NEW.total\_amount > 1000 AND :NEW.manager\_approval IS NULL THEN

RAISE\_APPLICATION\_ERROR (-20002, 'Manager approval is required for purchase orders over $1000.');

END IF; END;

/

```

\*\*What does the SQL trigger `enforce\_manager\_approval` do?\*\*

1. It inserts records into the `purchase\_orders` table.
2. It updates records in the `purchase\_orders` table.
3. It prevents the insertion of purchase orders with a total amount over $1000 if they don't have manager approval, raising a custom application error if such an insert is attempted.
4. It calculates the total amount of all purchase orders.

\*\*Correct Option:\*\* C

---

\*\*Ques4 - Consider the following SQL trigger: (Difficulty level – Hard)\*\*

```sql

CREATE OR REPLACE TRIGGER calculate\_total\_order\_amount AFTER INSERT OR UPDATE ON order\_items

FOR EACH ROW DECLARE

total\_amount NUMBER; BEGIN

total\_amount := 0;

SELECT SUM(quantity \* unit\_price) INTO total\_amount FROM order\_items WHERE order\_id = :NEW.order\_id;

UPDATE orders SET total\_amount = total\_amount WHERE order\_id = :NEW.order\_id;

END;

/

```

\*\*What does the SQL trigger `calculate\_total\_order\_amount` do?\*\*

1. It inserts records into the `order\_items` table.
2. It updates records in the `order\_items` table.
3. It calculates the total order amount for an order whenever a new order item is inserted or an existing order item is updated, and updates the `total\_amount` in the `orders` table.
4. It calculates the average order amount.

\*\*Correct Option:\*\* C

---

\*\*Ques5 - Consider the following SQL trigger: (Difficulty level –Easy)\*\*

```sql

CREATE OR REPLACE TRIGGER prevent\_duplicate\_records BEFORE INSERT ON employees

FOR EACH ROW BEGIN

IF EXISTS (SELECT 1 FROM employees WHERE employee\_id =

:NEW.employee\_id) THEN

RAISE\_APPLICATION\_ERROR (-20003, 'Employee ID must be unique.');

END IF; END;

/

```

\*\*What does the SQL trigger `prevent\_duplicate\_records` do?\*\*

1. It inserts records into the `employees` table.
2. It updates records in the `employees` table.
3. It prevents the insertion of duplicate employee records with the same `employee\_id`, raising a custom application error if such an insert is attempted.
4. It calculates the total number of employees in the `employees` table.

\*\*Correct Option:\*\* C

---

\*\*Ques6 - Consider the following SQL trigger: (Difficulty level – Hard)\*\*

```sql

CREATE OR REPLACE TRIGGER calculate\_sales\_bonus AFTER INSERT OR UPDATE ON sales

FOR EACH ROW BEGIN

DECLARE

bonus\_amount NUMBER; BEGIN

IF :NEW.sale\_amount > 10000 THEN bonus\_amount := :NEW.sale\_amount \* 0.05;

UPDATE sales SET bonus = bonus\_amount WHERE sale\_id

= :NEW.sale\_id;

END IF; END;

END;

/

```

\*\*What does the SQL trigger `calculate\_sales\_bonus` do?\*\*

1. It inserts records into the `sales` table.
2. It updates records in the `sales` table.
3. It calculates a sales bonus for sales with an amount over $10,000 and updates the `bonus` field in the `sales` table whenever a new sale is inserted or an existing sale is updated.
4. It calculates the average sale amount.

\*\*Correct Option:\*\* C

\*\*Ques7 - Consider the following PL/SQL function: (Difficulty level – Easy)\*\*

```plsql

CREATE OR REPLACE FUNCTION get\_last\_name(full\_name VARCHAR2) RETURN VARCHAR2 IS

last\_name VARCHAR2(50); BEGIN

last\_name := SUBSTR(full\_name, INSTR(full\_name, ' ')+1); RETURN last\_name;

END;

```

\*\*What does the PL/SQL function `get\_last\_name` do?\*\*

1. It calculates the average length of all words in the input `full\_name`.
2. It calculates the length of the last word in the input `full\_name`.
3. It retrieves the last name from the input `full\_name`.
4. It checks if the input `full\_name` contains any digits and returns `TRUE` if it does, `FALSE` otherwise.

\*\*Correct Option:\*\* C

---

\*\*Ques8 - Consider the following PL/SQL function: (Difficulty level – Easy)\*\*

```plsql

CREATE OR REPLACE FUNCTION square\_number(num NUMBER) RETURN NUMBER IS

square NUMBER; BEGIN

square := num \* num; RETURN square;

END;

```

\*\*What does the PL/SQL function `square\_number` do?\*\*

1. It calculates the square root of the input number `num`.
2. It calculates the sum of two numbers.
3. It calculates the square of the input number `num`.
4. It calculates the factorial of the input number `num`.

\*\*Correct Option:\*\* C

---

\*\*Ques9 - Consider the following PL/SQL function: (Difficulty level – Easy)\*\*

```plsql

CREATE OR REPLACE FUNCTION is\_prime(number NUMBER) RETURN BOOLEAN IS

BEGIN

IF number <= 1 THEN RETURN FALSE;

END IF;

FOR i IN 2..number-1 LOOP

IF MOD(number, i) = 0 THEN RETURN FALSE;

END IF; END LOOP; RETURN TRUE;

END;

```

\*\*What does the PL/SQL function `is\_prime` do?\*\*

1. It checks if the input `number` is a prime number and returns `TRUE` if it is, `FALSE` otherwise.
2. It calculates the square root of the input `number`.
3. It calculates the factorial of the input `number`.
4. It checks if the input `number` is even and returns `TRUE` if it is, `FALSE` otherwise.

\*\*Correct Option:\*\* A

---

\*\*Ques10 - Consider the following PL/ SQL function: (Difficulty level – Easy)\*\*

```plsql

CREATE OR REPLACE FUNCTION get\_day\_of\_week(date\_value DATE)

RETURN VARCHAR2 IS

day\_of\_week VARCHAR2(15); BEGIN

SELECT TO\_CHAR(date\_value, 'Day') INTO day\_of\_week FROM DUAL;

RETURN day\_of\_week; END;

```

\*\*What does the PL/SQL function `get\_day\_of\_week` do?\*\*

1. It calculates the day of the week for the input `date\_value` and returns it as a string.
2. It calculates the square root of the input `date\_value`.
3. It calculates the average of multiple dates.
4. It retrieves the month of the input `date\_value`.

\*\*Correct Option:\*\* A

\*\*Ques11 - Consider the following SQL cursor declaration: (Difficulty level – Hard)\*\*

```sql

DECLARE

product\_cursor CURSOR FOR

SELECT product\_id, product\_name FROM products

WHERE product\_id NOT IN (SELECT DISTINCT product\_id FROM order\_items);

```

\*\*What does the SQL cursor `product\_cursor` do?\*\*

1. It retrieves all columns from the `products` table.
2. It retrieves the `product\_id` and `product\_name` columns from the `products` table for products that have not been ordered.
3. It updates the `product\_id` and `product\_name` columns in the `products` table.
4. It deletes records from the `products` table.

\*\*Correct Option:\*\* B

---

\*\*Ques12 - Consider the following SQL cursor declaration: (Difficulty level – Hard)\*\*

```sql

DECLARE

customer\_cursor CURSOR FOR

SELECT customer\_id, MAX(order\_date) AS last\_order\_date FROM orders

GROUP BY customer\_id

HAVING MAX(order\_date) < TO\_DATE('2023-01-01', 'YYYY- MM-DD');

```

\*\*What does the SQL cursor `customer\_cursor` do?\*\*

1. It retrieves all columns from the `orders` table.
2. It retrieves the `customer\_id` and the last order date columns from the `orders` table for customers whose last order date is before January 1, 2023.
3. It updates the `customer\_id` and last order date columns in the `orders` table.
4. It deletes records from the `orders` table.

\*\*Correct Option:\*\* B

---

\*\*Ques13 - Consider the following SQL cursor declaration: (Difficulty level – Hard)\*\*

```sql

DECLARE

employee\_cursor CURSOR FOR

SELECT employee\_id, employee\_name, department\_id FROM employees

WHERE department\_id = (SELECT department\_id FROM departments WHERE department\_name = 'Engineering');

```

\*\*What does the SQL cursor `employee\_cursor` do?\*\*

1. It retrieves all columns from the `employees` table.
2. It retrieves the `employee\_id`, `employee\_name`, and `department\_id` columns from the

`employees` table for employees in the 'Engineering' department.

1. It updates the `employee\_id`, `employee\_name`, and `department\_id` columns in the

`employees` table.

1. It deletes records from the `employees` table.

\*\*Correct Option:\*\* B

---

\*\*Ques14 - Consider the following SQL cursor declaration: (Difficulty level – Hard)\*\*

```sql

DECLARE

product\_cursor CURSOR FOR

SELECT product\_id, product\_name FROM products

WHERE product\_price = (SELECT MAX(product\_price) FROM products);

```

\*\*What does the SQL cursor `product\_cursor` do?\*\*

1. It retrieves all columns from the `products` table.
2. It retrieves the `product\_id` and `product\_name` columns from the `products` table for products with the highest product price.
3. It updates the `product\_id` and `product\_name` columns in the `products` table.
4. It deletes records from the `products` table.

\*\*Correct Option:\*\* B

---

\*\*Ques15 - Consider the following SQL cursor declaration: (Difficulty level – Hard)\*\*

```sql

DECLARE

order\_cursor CURSOR FOR

SELECT order\_id, order\_date FROM orders

WHERE order\_id = (SELECT MAX(order\_id) FROM orders);

```

\*\*What does the SQL cursor `order\_cursor` do?\*\*

1. It retrieves all columns from the `orders` table.
2. It retrieves the `order\_id` and `order\_date` columns from the `orders` table for the order with the highest order ID.
3. It updates the `order\_id` and `order\_date` columns in the `orders` table.
4. It deletes records from the `orders` table.

\*\*Correct Option:\*\* B

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **SR** | **Questions** | **Option 1** | **Option 2** | **Option 3** | **Option 4** | **Correct**  **Answer** |
| 1 | Which of the following is NOT a benefit of using transactions? | Data integrity | High availability | Data consistency | Data durablity | B |
| 2 | A transaction that violates the consistency property is considered to be: | Serializable | Inconsistent | Isolate | Error | B |
| 3 | Can you change the parameter values of a cursor after it has been declared and opened? | Yes, parameter values can be modified at any time. | No, parameter values are fixed once the cursor is declared and opened. | Parameter values can only be changed during cursor declaration. | Cursors cannot have parameter values. | B |
| 4 | Can you declare a cursor without specifying the SELECT statement immediately? | No, a SELECT statement must always be specified. | Yes, a SELECT  statement can be added later in the code. | Cursors cannot be declared in PL/SQL. | Cursors are automatically generated in PL/SQL. | A |
| 5 | Can you declare multiple cursors with the same name but different parameters in the same PL/SQL block? | Yes, as long as the cursor names are unique. | No, cursor names must be unique regardless of the parameters. | Multiple cursors are not allowed in the same PL/SQL block. | Cursors with parameters cannot have the same name. | B |
| 6 | Can you declare multiple cursors within the same PL/SQL block? If so, how do you differentiate them? | No, only one cursor is allowed per block. | Yes, multiple cursors can be declared, and they are differentiated by their data types. | Yes, multiple cursors can be declared, and they are differentiated by their names. | Multiple cursors cannot be used in PL/SQL. | C |
| 7 | Can you fetch data from a cursor into individual variables or into a record type? Explain. | Data can only be fetched into individual variables. | Data can only be fetched into a record type. | Data can be fetched into both individual variables and a record type. | Data cannot be fetched from a cursor. | C |
| 8 | Can you nest a Cursor FOR Loop inside another Cursor FOR Loop? If so, why might you do so? | No, nesting Cursor FOR Loops is not allowed. | Yes, you can nest Cursor FOR Loops to perform complex data processing and handle related data hierarchies. | Cursor FOR Loops can only be used individually, not nested. | Nesting Cursor FOR Loops results in performance issues. | B |
| 9 | Can you use a Cursor FOR Loop to update or delete records in a database table? Explain. | No, Cursor FOR Loops are read-only. | Yes, Cursor FOR Loops can update or delete records using the UPDATE and DELETE statements. | Cursor FOR Loops can only insert records, not update or delete them. | Cursor FOR Loops can only be used for reporting purposes. | B |
| 10 | Describe the differences between an implicit cursor and an explicit cursor in PL/SQL. | Implicit cursors are used for data modeling, while explicit cursors are used for data manipulation. | Implicit cursors are automatically created for DML statements, while explicit cursors are user-defined. | Implicit cursors are used for database connections, while explicit cursors are used for loop control. | Implicit cursors are used for hardware design, while explicit cursors are used for web development. | B |
| 11 | Describe the purpose of PL/SQL collections, and provide examples of their types. | PL/SQL collections are used for defining variables. | PL/SQL collections are used for database connections. | PL/SQL collections are used for storing multiple values of the same data type. | PL/SQL collections are used for creating triggers. | C |
| 12 | Explain how cursor parameters can be used to create dynamic cursors. | Cursor parameters have no role in creating dynamic cursors. | By allowing parameterization of the WHERE clause in the cursor's SELECT statement, you can create dynamic cursors that retrieve specific data based on different criteria. | Cursor parameters can only be used with static cursors. | Cursor parameters can be used to create triggers. | B |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 13 | Explain the concept of triggers in a database context. How are they used in PL/SQL? | Triggers are used for creating web applications. | Triggers are used for hardware design. | Triggers are used for automatically executing PL/SQL code in response to database events. | Triggers are used for data modeling. | C |
| 14 | Explain the difference between declaring a cursor and opening a cursor. | Declaring a cursor retrieves data; opening a cursor defines its structure. | Declaring a cursor defines its structure; opening a cursor retrieves data. | Declaring a cursor and opening a cursor are the same. | Declaring a cursor is not a PL/SQL concept. | B |
| 15 | Explain the importance of transactions in PL/SQL and how they are managed. | Transactions are used for web development. | Transactions are used for data modeling. | Transactions ensure data consistency and are managed using COMMIT and ROLLBACK  statements. | Transactions are not supported in PL/SQL. | C |
| 16 | Explain the purpose of a PL/SQL package and its components. | PL/SQL packages are used for web development. | PL/SQL packages are used for encapsulating procedures and functions. | PL/SQL packages are used for data modeling. | PL/SQL packages are used for hardware design. | B |
| 17 | How can you pass parameters to a PL/SQL procedure or function? | Parameters are passed using the CALL statement. | Parameters are not supported in PL/SQL. | Parameters are passed as input and output variables. | Parameters are passed using the DECLARE statement. | C |
| 18 | How can you resolve a deadlock in a database system? | By terminating one of the transactions involved in the deadlock. | By rolling back all transactions involved in the deadlock. | By increasing the isolation level. | Deadlocks cannot be resolved. | A |
| 19 | How do you create and manipulate PL/SQL associative arrays (index-by tables)? | Associative arrays are created using the ARRAY keyword. | Associative arrays are created using the INDEX keyword. | Associative arrays are not supported in PL/SQL. | Associative arrays are created using the TYPE keyword. | D |
| 20 | How do you declare a cursor, and what are the required components? | Cursors are automatically declared in PL/SQL. | Cursors are declared using the DECLARE CURSOR statement and require a SELECT statement. | Cursors are declared using the DECLARE keyword. | Cursors are declared using the OPEN statement. | B |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 21 | How do you declare a variable in PL/SQL, and what are the data types supported for variables? | Variables are declared using the DECLARE keyword, and PL/SQL supports only one data type. | Variables are declared using the VAR keyword, and PL/SQL supports multiple data types. | Variables are declared using the VARIABLE keyword, and PL/SQL supports multiple data types. | Variables are not supported in PL/SQL. | B |
| 22 | How do you define and use PL/SQL records and record types? | Records are used for creating tables in PL/SQL. | Records are defined using the DECLARE RECORD statement. | Records are used to hold data in a structured format. | Records are not supported in PL/SQL. | C |
| 23 | How do you ensure that you've fetched all available data from a cursor? | By using the CLOSE statement. | By using the OPEN statement. | By checking the cursor attribute %NOTFOUND. | Cursors automatically fetch all available data. | C |
| 24 | How do you handle database connections and transactions in PL/SQL? | Database connections and transactions are automatically managed by the PL/SQL engine. | Database connections and transactions are not supported in PL/SQL. | Database connections are established using the CONNECT statement, and transactions are managed using COMMIT and ROLLBACK  statements. | Database connections are established using the DECLARE statement. | C |
| 25 | How do you handle exceptions in PL/SQL? Provide an example. | Exceptions are handled using the IF-ELSE statement. | Exceptions are handled using the TRY-CATCH block. | Exceptions are handled using the EXCEPTION block. | Exceptions are not supported in PL/SQL. | C |
| 26 | How do you handle exceptions that may occur when working with cursors that have parameters? | By using the FETCH statement. | By ignoring exceptions and proceeding with the cursor operations. | By using exception handling techniques such as WHEN OTHERS and specific exception handlers for cursor- related errors. | Cursors with parameters do not raise exceptions. | C |
| 27 | How do you name a cursor, and what are some best practices for naming conventions? | Cursors are named automatically. | Cursors can be named using any random string. | Cursors should have meaningful names following naming conventions such as prefixing with CUR\_. | Cursors cannot have names in PL/SQL. | C |
| 28 | How do you open a cursor to make it ready for data retrieval? | Use the DECLARE CURSOR statement. | Use the OPEN CURSOR statement. | Use the FETCH statement. | Cursors are automatically opened in PL/SQL. | B |
| 29 | How do you pass values to the cursor parameters when opening the cursor? | Use the FETCH statement to provide parameter values. | Use the SET PARAMETER  statement. | Use a separate ASSIGN statement to assign values to parameters before opening the cursor. | Cursor parameters do not require values when opening. | C |
| 30 | How does a Cursor FOR Loop handle exceptions compared to explicit cursor processing? | Cursor FOR Loops do not support exception handling. | Cursor FOR Loops handle exceptions more gracefully by providing built-in error handling mechanisms. | Exception handling in Cursor FOR Loops is the same as in explicit cursor processing. | Cursor FOR Loops handle exceptions less efficiently than explicit cursors. | B |
| 31 | How does a cursor with parameters differ from a cursor without parameters in terms of flexibility? | Cursors with parameters are less flexible. | Cursors with parameters are more flexible because they can retrieve data based on varying conditions. | There is no difference in flexibility between the two types of cursors. | Cursors with parameters are slower. | B |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 32 | How is the declaration of a cursor different from a regular SQL query? | Cursors cannot be used to retrieve data. | Cursors have a SELECT statement, while regular SQL queries are standalone. | Regular SQL queries cannot be used in PL/SQL. | There is no difference; they are the same. | B |
| 33 | In a multi-user database system, what does optimistic concurrency control aim to achieve? | It aims to prevent transactions from running concurrently. | It aims to avoid blocking and allow transactions to proceed concurrently, only checking for conflicts at the end. | It aims to lock all records to avoid conflicts. | It aims to roll back all transactions. | B |
| 34 | In database recovery, what is the difference between forward recovery and backward recovery? | Forward recovery restores the database to a previous state, while backward recovery recovers the database to its current state. | Forward recovery is the same as database backup, while backward recovery restores the database to a previous state. | Forward recovery involves log analysis, while backward recovery involves restoring database backups. | There is no difference; the terms are used interchangeably. | C |
| 35 | In database recovery, what is the purpose of a database log file? | To store user data. | To record changes made to the database for recovery purposes. | To create database backups. | To store database metadata. | B |
| 36 | used? | View | Commit | Rollback | Flashback | C |
| 37 | In SQL, what is the role of the ROLLBACK statement? | To save pending changes. | To begin a new transaction. | To undo all changes made during the current transaction. | To release locks on database records. | C |
| 38 | Is it necessary to declare a cursor inside a PL/SQL block, or can it be declared globally in a package? | Cursors can only be declared globally. | Cursors can only be declared inside a PL/SQL block. | Cursors can be declared both globally and inside a PL/SQL block. | Cursors are not supported in PL/SQL. | C |
| 39 | What are database triggers, and when might you use them in PL/SQL? | Database triggers are used for hardware design. | Database triggers are used for declaring variables. | Database triggers are used to automatically respond to database events and can be used for auditing or enforcing business rules. | Database triggers are used for creating tables. | C |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 40 | What are some common use cases for using cursor parameters in PL/SQL? | Cursor parameters are rarely used in practice. | Common use cases include generating reports with different filter criteria, processing data based on user inputs, and customizing data retrieval based on changing conditions. | Cursor parameters are mainly used for database administration tasks. | Cursor parameters are only used in triggers. | B |
| 41 | What are the advantages of using explicit cursors over implicit cursors in PL/SQL? | Explicit cursors are faster in performance. | Implicit cursors are more flexible. | Explicit cursors are easier to use and provide more control. | Implicit cursors are automatically managed by the database. | C |
| 42 | What are the benefits of using PL/SQL for database programming compared to using SQL alone? | PL/SQL allows for creating web applications. | PL/SQL provides procedural capabilities for better control and encapsulation of logic in the database. | PL/SQL is used for hardware design. | PL/SQL is primarily used for data modeling. | B |
| 43 | What does the isolation level READ COMMITTED mean? | Reads data as it was when the transaction started. | Reads uncommitted changes made by other transactions. | Prevents any reads until the transaction is committed. | Reads data from committed transactions only. | A |
| 44 | What does the SAVEPOINT statement do in SQL? | Marks a point in a transaction to be rolled back to later. | Commits the transaction. | Opens a new transaction. | Locks the database. | A |
| 45 | What happens when you fetch data from a cursor that has no more rows to retrieve? | An error occurs. | The cursor is automatically closed. | The cursor remains open and ready for the next fetch. | Cursors always have more rows to retrieve. | A |
| 46 | What is a cursor in PL/SQL, and why is it used? | A cursor is a database table. | A cursor is used for looping through query results. | A cursor is a data type in PL/SQL. | A cursor is used for creating triggers. | B |
| 47 | What is a database checkpoint? | A physical location where the database is stored. | A marker indicating the point in time up to which transactions are considered safe and can be recovered. | A log file containing SQL statements. | A password for accessing the database. | B |
| 48 | What is a database lock in the context of concurrency control? | A mechanism to block all database transactions. | A mechanism to prevent data corruption. | A mechanism to prevent multiple transactions from accessing the same data simultaneously. | A mechanism to unlock databases. | C |
| 49 | What is a database restore operation? | A process that erases all data from the database. | A process that removes the database log files. | A process that brings a database back to a previous state by applying database backups and log files. | A process that upgrades the database to a new version. | C |
| 50 | What is a database transaction? | A single SQL statement. | A sequence of related SQL statements that are executed as a unit. | A database schema. | A database table. | B |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 51 | What is a deadlock in the context of concurrency control? | A situation where a transaction is rolled back. | A situation where two or more transactions are waiting for each other to release locks. | A situation where a transaction is terminated. | A situation where a transaction is committed. | B |
| 52 | What is a distributed transaction in database management? | A transaction that involves multiple databases. | A transaction that is committed automatically. | A transaction with a large number of SQL statements. | A transaction without a COMMIT. | A |
| 53 | What is a full database backup? | A backup that includes only a subset of the database. | A backup that includes all the data and structures in the database. | A backup that contains only log files. | A backup that is encrypted for security. | B |
| 54 | What is a nested transaction in SQL? | A transaction inside another transaction. | A transaction without any nested SQL statements. | A transaction that cannot be rolled back. | A transaction with a SAVEPOINT. | A |
| 55 | What is a PL/SQL function, and how does it differ from a procedure? | A function is used for controlling database transactions. | A function is used for encapsulating reusable logic and returns a value. | A procedure is used for data modeling. | A procedure is used for creating tables. | B |
| 56 | What is concurrency control in database systems? | Managing multiple database transactions simultaneously. | Controlling access to the database using passwords. | Rolling back transactions in case of errors. | Creating indexes for database tables. | A |
| 57 | What is cursor positioning, and how does it relate to fetching data? | Cursor positioning determines the cursor's name. | Cursor positioning is the process of opening a cursor. | Cursor positioning refers to the current position of the cursor relative to the result set, affecting the next fetch operation. | Cursor positioning is not relevant in PL/SQL. | C |
| 58 | What is database recovery in the context of database management systems? | Backing up the database to prevent data loss. | The process of restoring a database to a previous state after a failure. | Increasing the database size to accommodate more data. | Encrypting database files for security. | B |
| 59 | What is dynamic SQL, and why might you use it in PL/SQL? | Dynamic SQL is used for creating triggers in PL/SQL. | Dynamic SQL allows you to generate and execute SQL statements at runtime. | Dynamic SQL is used for web development. | Dynamic SQL is used for hardware design. | B |
| 60 | What is PL/SQL, and how does it differ from SQL? | PL/SQL is a markup language for web development. | PL/SQL is a procedural extension of SQL. | PL/SQL is a data modeling language. | PL/SQL is a hardware description language. | B |
| 61 | What is the ACID property in the context of database transactions? | Atomicity, Consistency, Isolation, Durability | Aggregation, Continuity, Integrity, Durability | Affinity, Consistency, Isolation, Durability | Atomicity, Cancellation, Isolation, Division | A |
| 62 | What is the advantage of using a Cursor FOR Loop over traditional cursor processing? | Cursor FOR Loops are slower than traditional cursors. | Cursor FOR Loops offer less control. | Cursor FOR Loops simplify cursor processing by handling cursor declaration, opening, fetching, and closing automatically. | Cursor FOR Loops are not recommended in PL/SQL. | C |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 63 | What is the benefit of using cursor parameters when working with data retrieval? | Cursor parameters make cursor declaration simpler. | Cursor parameters allow for dynamic queries and customization of data retrieval based on varying conditions. | Cursor parameters improve cursor performance. | Cursor parameters are not useful in PL/SQL. | B |
| 64 | What is the default behavior of a Cursor FOR Loop if there are no rows to process? | It raises an error. | It skips the loop and continues with the next statement. | It automatically exits the loop. | It waits for rows to be available. | C |
| 65 | What is the difference between a PL/SQL procedure and a PL/SQL function? | A procedure returns a value, while a function does not. | A procedure does not return a value, while a function does. | A procedure and a function are the same. | A procedure and a function are not supported in PL/SQL. | B |
| 66 | What is the opposite of a COMMIT statement in SQL? | ROLLBACK | BEGIN TRANSACTION | SAVEPOINT | LOCK | A |
| 67 | What is the primary drawback of using pessimistic concurrency control in a database system? | It can lead to data inconsistency. | It can result in excessive locking and reduced concurrency. | It is slower than optimistic concurrency control. | It requires frequent COMMIT statements. | B |
| 68 | What is the purpose of a differential backup in database recovery? | To recover the database to a specific point in time. | To restore only the data that has changed since the last full backup, reducing the recovery time. | To make a copy of the entire database. | To compress the database backup files. | B |
| 69 | What is the purpose of declaring a cursor in PL/SQL? | To insert data into a table. | To define a variable in PL/SQL. | To retrieve and manipulate query results in a controlled manner. | To create a trigger in PL/SQL. | C |
| 70 | What is the purpose of fetching data from a cursor in PL/SQL? | To insert data into a table. | To define a variable in PL/SQL. | To retrieve and manipulate query results row by row. | To create a trigger in PL/SQL. | C |
| 71 | What is the purpose of the COMMIT statement in SQL? | To roll back a transaction. | To save all pending changes permanently to the database. | To lock database records. | To create a new transaction. | B |
| 72 | What is the purpose of the FOR loop in PL/SQL, and how is it used? | The FOR loop is used for declaring variables. | The FOR loop is used for defining exceptions. | The FOR loop is used for iterative processing. | The FOR loop is used for database connections. | C |
| 73 | What is the purpose of the LOCK TABLE statement in SQL? | To unlock a table. | To create a new table. | To specify the locking mode for a table explicitly. | To commit a transaction. | C |
| 74 | What is the purpose of the WHEN OTHERS exception handler in PL/SQL? | It is used for declaring variables. | It is used for defining custom exceptions. | It is used to catch and handle unexpected exceptions. | It is used for database connections. | C |
| 75 | What is the role of a database backup in recovery? | It serves as a temporary storage location. | It records changes made to the database. | It provides a copy of the database that can be used to restore data in case of data loss or corruption. | It ensures data consistency during concurrent transactions. | C |
| 76 | What is the role of the FETCH statement in cursor processing? | It defines the cursor's name. | It specifies the number of rows to fetch. | It retrieves rows from the cursor into variables or records. | It opens the cursor for data retrieval. | C |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 77 | What is the significance of the %ROWTYPE attribute when declaring a cursor? | It defines the cursor's name. | It specifies the number of rows the cursor can fetch. | It defines the structure of the result set the cursor will hold. | It determines the data type of cursor variables. | C |
| 78 | What is the significance of the recovery point objective (RPO) in database recovery planning? | It defines the maximum number of recovery points allowed. | It specifies the desired point in time to which a database should be recovered after a failure. | It defines the number of database logs to retain. | It measures the database's performance. | B |
| 79 | Which ACID property ensures that a transaction is completed in its entirety or not at all? | Atomicity | Consistency | Isolation | Durability | A |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 80 | Which concurrency control technique allows conflicts to be detected and resolved only at the commit time? | Validation-based protocol | Timestamp ordering | Two-phase locking | Three-phase locking | A |
| 81 | Which database recovery model allows for point-in-time recovery to any arbitrary moment? | Simple recovery model | Full recovery model | Bulk-logged recovery model | Incremental recovery model | B |
| 82 | Which isolation level in SQL provides the highest level of isolation but can lead to concurrency issues? | READ COMMITTED | SERIALIZABLE | READ UNCOMMITTED | REPEATABLE READ | B |
| 83 | Which of the following is not a concurrency control mechanism in DBMS? | Locking | Timestamp ordering | Multiversion concurrency control | Rollback and recovery | D |
| 84 | Which of the following recovery techniques is based on maintaining multiple copies  of the database at different points in time? | Replication | Deferred update | Redo logging | Undo logging | A |
| 85 | Which SQL statement is used to set a SAVEPOINT within a transaction? | BEGIN SAVEPOINT | SAVEPOINT | SET SAVEPOINT | CREATE SAVEPOINT | B |
| 86 | Which technique allows concurrent transactions to access different parts of a database without conflicts? | Pessimistic concurrency control | Optimistic concurrency control | Exclusive locking | Distributed transactions | B |
| 87 | [ON table\_name] specifies the name of the table associated with the trigger. | Yes | No | Can be yes or no | None of the above | A |
| 88 | some events occur. | Procedure | Triggers | Collection | Transaction | B |
| 89 | A consists of a sequence of query and/or update statements. | Transaction | Commit | Rollback | Flashback | A |
| 90 | A is a special kind of a store procedure that executes in response to certain action on the table like insertion, deletion or updation of data. | Procedures | Triggers | Functions | None of the mentioned | B |
| 91 | A stored procedure in SQL is a | Block of functions | Group of Transact- SQL statements compiled into a single execution plan. | Group of distinct SQL statements. | None of the mentioned | B |
| 92 | A view is actually a? | composition of a table | decomposition of a table | associated to a table | None of the above | A |
| 93 | All objects placed in the specification are called objects. | private | protected | public | None of the above | B |
| 94 | Any subprogram not in the package specification but coded in the package body is called a object. | protected | private | self | public | B |
| 95 | Boyce-Codd Normal Form (BCNF) is an extension of which normal form? | A) First Normal Form (1NF) | B) Second Normal Form (2NF) | C) Third Normal Form (3NF) | D) Fourth Normal Form (4NF) | B |
| 96 | Consider the following action:  TRANSACTION.....  Commit; ROLLBACK;  What does Rollback do? | Undoes the transactions before commit | Clears all transactions | Redoes the transactions before commit | No action | D |
| 97 | Consider the following cursor declaration in SQL:  DECLARE cursor1 CURSOR FOR  SELECT FirstName, LastName FROM Employees WHERE Department = 'Sales'  If you want to open and fetch rows from this cursor, what SQL statement should | FETCH NEXT FROM  cursor1; | **OPEN cursor1;** | CLOSE cursor1; | DECLARE cursor1 CURSOR FOR ... | B |
| 98 | S = r2(X); r1(X); r2(Y); w1(X); r1(Y); w2(X); a1; a2  where ri(Z) denotes a read operation by transaction Ti on a variable Z, wi(Z) denotes a write operation by Ti on a variable Z and ai denotes an abort by transaction Ti .  Which one of the following statements about the above schedule is TRUE? | S is non-recoverable | S is recoverable, but has a cascading abort | S does not have a cascading abort | S is strict | C |
| 99 | Consider the following stored procedure in SQL:  CREATE PROCEDURE CalculateTotalPrice @ProductID INT,  @Quantity INT AS  BEGIN  DECLARE @Price DECIMAL(10, 2)  SELECT @Price = UnitPrice FROM Products WHERE ProductID = @ProductID PRINT 'Total Price: ' + CAST(@Price \* @Quantity AS VARCHAR)  END  If you call this stored procedure with @ProductID = 101 and @Quantity = 5, what | Total Price: 505 | **Total Price: 25** | Total Price: 101 | Total Price: 5 | B |
| 100 | Create function dept count(dept\_name varchar(20)) begin  declare d count integer; select count(\*) into d count from instructor  where instructor.dept\_name= dept\_name return d count;  end  Find the error in the the above statement. | Return type missing | Dept\_name is mismatched | Reference relation is not mentioned | All of the mentioned | A |
| 101 | CREATE OR REPLACE FUNCTION calculate\_gpa( student\_id NUMBER  ) RETURN NUMBER IS  total\_points NUMBER := 0; total\_credits NUMBER := 0; gpa NUMBER;  BEGIN  -- Calculate GPA for a student  -- Assume the grade scale: A=4, B=3, C=2, D=1, F=0  -- Credits per subject: 3 credits  -- GPA = (Total Points) / (Total Credits) RETURN gpa;  END; | Deletes students by age. | Updates student's name. | Calculates a student's GPA based on their grades. | Retrieves students in a subject above average. | C |
| 102 | CREATE OR REPLACE FUNCTION calculate\_student\_gpa( student\_id NUMBER  ) RETURN NUMBER IS gpa NUMBER;  BEGIN  -- Calculate GPA for a student based on their grades and credit hours RETURN gpa;  END; | Deletes students by age. | Updates student's name. | Calculates a student's GPA based on their grades and credit hours. | Transfers students from one batch to another. | C |
| 103 | CREATE OR REPLACE FUNCTION calculate\_subject\_average( subject\_name VARCHAR2  ) RETURN NUMBER IS  avg\_grade NUMBER; BEGIN  SELECT AVG(grade) INTO avg\_grade FROM student  WHERE subject = subject\_name; RETURN avg\_grade;  END; | Deletes students by age. | Updates student information. | Calculates the average grade in a specific subject. | Returns a list of students with above-average grades in a subject. | C |
| 104 | CREATE OR REPLACE FUNCTION calculate\_total\_students RETURN NUMBER IS total\_students NUMBER;  BEGIN  SELECT COUNT(\*) INTO total\_students FROM student;  RETURN total\_students;  END; | Deletes students by name. | Updates student information. | Calculates the total number of students. | Returns a list of students by age range. | C |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 105 | CREATE OR REPLACE FUNCTION count\_students\_by\_age( age NUMBER  ) RETURN NUMBER IS  student\_count NUMBER; BEGIN  SELECT COUNT(\*) INTO student\_count FROM student  WHERE age = age;  RETURN student\_count; END; | Retrieves students by age. | Returns the total count of students by age. | Enrolls students in multiple subjects. | Deletes students by subject. | B |
| 106 | CREATE OR REPLACE FUNCTION count\_students\_in\_subject( subject\_name VARCHAR2  ) RETURN NUMBER IS  student\_count NUMBER; BEGIN  SELECT COUNT(\*) INTO student\_count FROM student  WHERE subject = subject\_name; RETURN student\_count;  END; | Retrieves students by subject count. | Returns a list of students with the maximum number of subjects. | Deletes students by age. | Updates a student's batch. | A |
| 107 | SYS\_REFCURSOR IS  students\_cursor SYS\_REFCURSOR; BEGIN  OPEN students\_cursor FOR SELECT student\_id  FROM student  WHERE age = (SELECT MAX(age) FROM student); RETURN students\_cursor;  END; | Retrieves students with the lowest age. | Returns a list of students with the highest age. | Deletes students by age. | Awards scholarships to deserving students. | B |
| 108 | SYS\_REFCURSOR IS  students\_cursor SYS\_REFCURSOR; BEGIN  OPEN students\_cursor FOR SELECT student\_id  FROM (  SELECT student\_id, COUNT(DISTINCT subject) AS subject\_count FROM student  GROUP BY student\_id  ORDER BY subject\_count DESC  )  WHERE ROWNUM = 1;  RETURN students\_cursor; END; | Retrieves students with the highest grades. | Returns a list of subjects with above- average grades. | Enrolls students in multiple subjects. | Deletes students by subject. | B |
| 109 | CREATE OR REPLACE FUNCTION find\_students\_with\_subject\_count( subject\_count NUMBER  ) RETURN SYS\_REFCURSOR IS  students\_cursor SYS\_REFCURSOR; BEGIN  OPEN students\_cursor FOR SELECT student\_id  FROM (  SELECT student\_id, COUNT(DISTINCT subject) AS subject\_count FROM student  GROUP BY student\_id  )  WHERE subject\_count = subject\_count; RETURN students\_cursor;  END; | Retrieves students by subject count. | Returns a list of students with a specific subject count. | Deletes students by age. | Updates a student's subject and grade. | A |
| 110 | SYS\_REFCURSOR IS  subjects\_cursor SYS\_REFCURSOR; BEGIN  OPEN subjects\_cursor FOR SELECT subject  FROM (  SELECT subject, COUNT(\*) AS student\_count FROM student  GROUP BY subject  ORDER BY student\_count ASC  )  WHERE ROWNUM = 1;  RETURN subjects\_cursor; END; | Retrieves subjects with the highest grades. | Returns a list of subjects with the lowest number of students. | Enrolls students in multiple subjects. | Deletes students by subject. | B |
| 111 | CREATE OR REPLACE FUNCTION get\_highest\_grade\_by\_subject( subject\_name VARCHAR2  ) RETURN NUMBER IS  highest\_grade NUMBER; BEGIN  SELECT MAX(grade) INTO highest\_grade FROM student  WHERE subject = subject\_name; RETURN highest\_grade;  END; | Returns the highest grade of all students. | Enrolls students in a subject. | Deletes students by subject. | Returns the highest grade in a specific subject. | D |
| 112 | CREATE OR REPLACE FUNCTION get\_student\_count\_by\_subject( subject\_name VARCHAR2  ) RETURN NUMBER IS  student\_count NUMBER; BEGIN  SELECT COUNT(\*) INTO student\_count FROM student  WHERE subject = subject\_name; RETURN student\_count;  END; | Deletes a student by subject. | Returns the count of students in a specific subject. | Enrolls a student in a subject. | Updates the student's subject. | B |
| 113 | subject\_name VARCHAR2  ) RETURN SYS\_REFCURSOR IS  students\_cursor SYS\_REFCURSOR; BEGIN  OPEN students\_cursor FOR SELECT student\_id  FROM student  WHERE subject = subject\_name AND grade > (SELECT AVG(grade) FROM student WHERE subject = subject\_name);  RETURN students\_cursor;  END; | Retrieves all students. | Returns a list of students with above- average grades in a specific subject. | Enrolls students in a subject. | Deletes students by subject. | B |
| 114 | CREATE OR REPLACE FUNCTION get\_students\_by\_age\_range( min\_age NUMBER,  max\_age NUMBER  ) RETURN SYS\_REFCURSOR IS  students\_cursor SYS\_REFCURSOR; BEGIN  OPEN students\_cursor FOR SELECT student\_id  FROM student  WHERE age BETWEEN min\_age AND max\_age; RETURN students\_cursor;  END; | Retrieves students by age range. | Returns a list of students with above- average grades in a specific subject. | Enrolls students in multiple subjects. | Deletes students by subject. | A |
| 115 | CREATE OR REPLACE FUNCTION get\_students\_by\_grade\_range( min\_grade NUMBER,  max\_grade NUMBER  ) RETURN SYS\_REFCURSOR IS  students\_cursor SYS\_REFCURSOR; BEGIN  OPEN students\_cursor FOR SELECT student\_id  FROM student  WHERE grade BETWEEN min\_grade AND max\_grade; RETURN students\_cursor;  END; | Retrieves students by grade range. | Returns the total count of students by grade range. | Awards scholarships to deserving students. | Transfers students from one batch to another. | A |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 116 | CREATE OR REPLACE FUNCTION get\_students\_by\_subject\_and\_grade( subject\_name VARCHAR2,  min\_grade NUMBER  ) RETURN SYS\_REFCURSOR IS  students\_cursor SYS\_REFCURSOR; BEGIN  OPEN students\_cursor FOR SELECT student\_id  FROM student  WHERE subject = subject\_name AND grade >= min\_grade; RETURN students\_cursor;  END; | Retrieves students by age range. | Returns a list of students with a specific subject and minimum grade. | Enrolls students in multiple subjects. | Deletes students by subject. | B |
| 117 | IS  students\_cursor SYS\_REFCURSOR; BEGIN  OPEN students\_cursor FOR SELECT student\_id  FROM student  WHERE batch\_id IS NOT NULL; RETURN students\_cursor;  END; | Retrieves students in a specific batch. | Returns a list of students with the highest age. | Enrolls students in multiple subjects. | Deletes students by subject. | A |
| 118 | subject\_name VARCHAR2  ) RETURN SYS\_REFCURSOR IS  students\_cursor SYS\_REFCURSOR; BEGIN  OPEN students\_cursor FOR SELECT student\_id  FROM student  WHERE subject = subject\_name  AND grade > (SELECT AVG(grade) FROM student WHERE subject = subject\_name);  RETURN students\_cursor;  END; | Retrieves all students. | Returns a list of students with above- average grades in a specific subject. | Enrolls students in a subject. | Updates student information. | B |
| 119 | SYS\_REFCURSOR IS  students\_cursor SYS\_REFCURSOR; BEGIN  OPEN students\_cursor FOR SELECT student\_id  FROM student  WHERE grade = (SELECT MAX(grade) FROM student); RETURN students\_cursor;  END; | Retrieves students with the lowest grade. | Returns students with the highest grade. | Enrolls students in a batch. | Deletes students by age. | B |
| 120 | SYS\_REFCURSOR IS  students\_cursor SYS\_REFCURSOR; BEGIN  OPEN students\_cursor FOR SELECT student\_id  FROM student  WHERE grade = (SELECT MIN(grade) FROM student); RETURN students\_cursor;  END; | Retrieves students with the highest grade. | Returns students with the lowest grade. | Enrolls students in a batch. | Updates student information. | B |
| 121 | RETURN SYS\_REFCURSOR IS  students\_cursor SYS\_REFCURSOR; BEGIN  OPEN students\_cursor FOR SELECT student\_id  FROM (  SELECT student\_id, MAX(grade) AS max\_grade FROM student  GROUP BY student\_id  );  RETURN students\_cursor; END; | Retrieves students with the subject wise lowest grades . | Returns students with the subject wise highest grades. | Enrolls students in multiple subjects. | Deletes students by subject. | B |
| 122 | CREATE OR REPLACE FUNCTION get\_subjects\_by\_student( student\_id NUMBER  ) RETURN SYS\_REFCURSOR IS  subjects\_cursor SYS\_REFCURSOR; BEGIN  OPEN subjects\_cursor FOR SELECT DISTINCT subject FROM student  WHERE student\_id = student\_id; RETURN subjects\_cursor;  END; | Deletes a student record. | Updates a student's information. | Returns a list of distinct subjects for a specific student. | Enrolls students in a subject. | C |
| 123 | SYS\_REFCURSOR IS  subjects\_cursor SYS\_REFCURSOR; BEGIN  OPEN subjects\_cursor FOR SELECT subject  FROM (  SELECT subject, MAX(grade) AS max\_grade FROM student  GROUP BY subject  );  RETURN subjects\_cursor; END; | Retrieves subjects with the highest grades. | Returns a list of subjects with above- average grades. | Enrolls students in multiple subjects. | Deletes students by subject. | A |
| 124 | RETURN SYS\_REFCURSOR IS  subjects\_cursor SYS\_REFCURSOR; BEGIN  OPEN subjects\_cursor FOR SELECT subject  FROM (  SELECT subject, AVG(grade) AS avg\_grade FROM student  GROUP BY subject  )  WHERE avg\_grade > (SELECT AVG(grade) FROM student); RETURN subjects\_cursor;  END; | Retrieves subjects with students above average grades. | Returns a list of subjects with students below average grades. | Enrolls students in multiple subjects. | Deletes students by subject. | A |
| 125 | CREATE OR REPLACE PROCEDURE archive\_student\_records IS BEGIN  -- Write logic to archive old student records  END; | Deletes students by age. | Updates student information. | Archives old student records. | Awards scholarships to deserving students. | C |
| 126 | CREATE OR REPLACE PROCEDURE assign\_student\_grade( student\_id NUMBER,  subject\_name VARCHAR2, grade NUMBER  ) IS BEGIN  -- Write logic to assign a specific grade to a student in a subject  END; | Deletes students by age. | Updates a student's name. | Assigns a specific grade to a student. | Promotes students to the next grade. | C |
| 127 | CREATE OR REPLACE PROCEDURE assign\_student\_subjects\_and\_grades( student\_id NUMBER,  subject\_grades SYS.ODCINUMBERLIST  ) IS BEGIN  -- Write logic to assign subjects and grades to a student  END; | Deletes students by age. | Updates student information. | Assigns subjects and grades to a student. | Promotes students to the next grade. | C |
| 128 | CREATE OR REPLACE PROCEDURE delete\_student( student\_id NUMBER  ) IS BEGIN  DELETE FROM student  WHERE student\_id = student\_id; COMMIT;  END; | Retrieve student details by ID. | Update student information. | Enroll a new student. | Delete a student record. | D |
| 129 | CREATE OR REPLACE PROCEDURE delete\_students\_by\_age( max\_age NUMBER  ) IS BEGIN  DELETE FROM student WHERE age > max\_age; COMMIT;  END; | Deletes students by name. | Updates student age. | Deletes students older than a specified age. | Calculates students' GPA. | C |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 130 | CREATE OR REPLACE PROCEDURE delete\_students\_by\_batch( batch\_id NUMBER  ) IS BEGIN  DELETE FROM student WHERE batch\_id = batch\_id; COMMIT;  END; | Deletes students by age. | Updates student information. | Deletes students by batch. | Assigns subjects and grades to a student. | C |
| 131 | CREATE OR REPLACE PROCEDURE delete\_students\_by\_subject( subject\_name VARCHAR2  ) IS BEGIN  DELETE FROM student  WHERE subject = subject\_name; COMMIT;  END; | Deletes students by age. | Updates student information. | Deletes students by subject. | Enrolls students in multiple subjects. | C |
| 132 | student\_name VARCHAR2, student\_age NUMBER, subject VARCHAR2, student\_grade NUMBER  ) IS BEGIN  INSERT INTO student(student\_id, name, age, subject, grade) VALUES(student\_sequence.NEXTVAL, student\_name, student\_age, subject, student\_grade);  COMMIT;  END; | Deletes a student record. | Updates a student's information. | Enrolls a new student with provided details. | Returns the count of students in a specific subject. | C |
| 133 | CREATE OR REPLACE PROCEDURE enroll\_student\_in\_multiple\_subjects( student\_name VARCHAR2,  student\_age NUMBER, subjects VARCHAR2, student\_grades VARCHAR2  ) IS BEGIN  -- Write logic to enroll a student in multiple subjects with corresponding grades  END; | Retrieve student details by ID. | Update student information. | Enroll students in multiple subjects. | Delete students by subject. | C |
| 134 | CREATE OR REPLACE PROCEDURE increase\_student\_grades( grade\_increase NUMBER  ) IS BEGIN  UPDATE student  SET grade = grade + grade\_increase; COMMIT;  END; | Deletes students by age. | Updates student information. | Increases student grades. | Awards scholarships to deserving students. | C |
| 135 | CREATE OR REPLACE PROCEDURE print\_student\_details( student\_id NUMBER  ) IS  student\_name VARCHAR2(100); student\_age NUMBER; subject\_name VARCHAR2(50); student\_grade NUMBER;  BEGIN  SELECT name, age, subject, grade  INTO student\_name, student\_age, subject\_name, student\_grade FROM student  WHERE student\_id = student\_id;  DBMS\_OUTPUT.PUT\_LINE('Student ID: ' || student\_id); DBMS\_OUTPUT.PUT\_LINE('Name: ' || student\_name); DBMS\_OUTPUT.PUT\_LINE('Age: ' || student\_age); | Deletes a student record. | Updates student information. | Prints details of a student by ID. | Returns the count of students in a specific subject. | C |
| 136 | CREATE OR REPLACE PROCEDURE print\_student\_transcript( student\_id NUMBER  ) IS BEGIN  -- Write logic to print the student's transcript  END; | Deletes students by age. | Updates student information. | Prints a student's transcript. | Returns students with the lowest grade. | C |
| 137 | student\_id NUMBER  ) IS BEGIN  FOR rec IN (SELECT subject, grade FROM student WHERE student\_id = student\_id) LOOP  DBMS\_OUTPUT.PUT\_LINE('Subject: ' || rec.subject || ', Grade: ' || rec.grade); END LOOP;  END; | Deletes a student record. | Updates student information. | Prints subjects and grades of a student by ID. | Returns a list of subjects in which a student is enrolled. | C |
| 138 | CREATE OR REPLACE PROCEDURE transfer\_student( student\_id NUMBER,  new\_batch\_id NUMBER  ) IS BEGIN  -- Write logic to transfer a student to a new batch  END; | Deletes students by age. | Updates student information. | Transfers a student to a new batch. | Calculates the average grade in a subject. | C |
| 139 | CREATE OR REPLACE PROCEDURE update\_student\_age( student\_id NUMBER,  new\_age NUMBER  ) IS BEGIN  UPDATE student SET age = new\_age  WHERE student\_id = student\_id; COMMIT;  END;  / | Deletes a student record. | Updates a student's age. | Enrolls a new student. | Returns the count of students in a specific subject. | B |
| 140 | CREATE OR REPLACE PROCEDURE update\_student\_name( student\_id NUMBER,  new\_name VARCHAR2  ) IS BEGIN  UPDATE student  SET name = new\_name  WHERE student\_id = student\_id; COMMIT;  END; | Deletes students by age. | Updates a student's name. | Enrolls a new student. | Returns the count of students in a specific subject. | B |
| 141 | CREATE OR REPLACE PROCEDURE update\_student\_subject\_and\_grade( student\_id NUMBER,  subject\_name VARCHAR2, new\_grade NUMBER  ) IS BEGIN  -- Write logic to update a student's subject and grade  END; | Deletes a student's subject and grade. | Updates a student's subject and grade. | Enrolls a new student. | Returns a student's subject and grade. | B |
| 142 | Create procedure dept\_count proc(in dept name varchar(20), out d count integer)  begin  select count(\*) into d count from instructor  where instructor.dept name= dept count proc.dept name end  Which of the following is used to call the procedure given above ? | Declare d\_count integer; | Declare d\_count integer;  call dept\_count proc(’Physics’, d\_count); | Declare d\_count integer; call dept\_count  proc(’Physics’); | Declare d\_count;  call dept\_count proc(’Physics’, d\_count); | B |
| 143 | Declare out of classroom seats condition  DECLARE exit handler FOR OUT OF classroom seats BEGIN  SEQUENCE OF statements END  The above statements are used for | Calling procedures | Handling Exception | Handling procedures | All of the mentioned | B |
| 144 | internal database error. | Yes | No | 1 or 2 | All of the above | A |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 145 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to calculate the average grade for each subject. Which of the following code snippets accomplishes this task? | DECLARE  CURSOR subject\_cursor IS  SELECT DISTINCT  subject  FROM student; BEGIN  FOR subject\_rec IN subject\_cursor LOOP  -- Calculate and print the average grade for each subject  END LOOP; END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; avg\_grade NUMBER; BEGIN  FOR subject\_rec IN subject\_cursor LOOP  -- Calculate and store the average grade for each subject in avg\_grade  DBMS\_OUTPUT.PUT\_  LINE('Subject: ' || subject\_rec.subject  || ', Avg Grade: ' || avg\_grade);  END LOOP; END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; subject\_recordstudent% ROWTYPE;  BEGIN  FOR subject\_rec IN subject\_cursor LOOP  -- Calculate and print the average grade for each subject  END LOOP; END; | DECLARE  CURSOR subject\_cursor IS SELECT DISTINCT subject FROM student;  BEGIN  FOR subject\_rec IN subject\_cursor LOOP  -- Calculate and print the average grade for each subject END LOOP;  END; | B |
| 146 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to fetch all students who have a grade greater than or equal to 90 in the subject "Mathematics." Which of the following code snippets accomplishes this task? | DECLARE CURSOR  high\_math\_grades IS SELECT student\_id FROM student WHERE subject =  'Mathematics' AND grade  >= 90; BEGIN  FOR rec IN high\_math\_grades LOOP  DBMS\_OUTPUT.PUT\_LINE  ('Student ID: ' || rec.student\_id); END LOOP; END; | DECLARE  CURSOR  high\_math\_grades IS SELECT student\_id FROM student WHERE subject =  'Mathematics' AND grade >= 90; student\_recordstude nt%ROWTYPE; BEGIN  OPEN  high\_math\_grades; LOOP  FETCH  high\_math\_grades INTO student\_record;  EXIT WHEN  high\_math\_grades% NOTFOUND;  DBMS\_OUTPUT.PUT\_  LINE('Student ID: ' || student\_record.stude nt\_id);  END LOOP; CLOSE  high\_math\_grades;  END; | DECLARE CURSOR  high\_math\_grades IS SELECT student\_id FROM student WHERE subject =  'Mathematics' AND grade >= 90; student\_id NUMBER; BEGIN  OPEN  high\_math\_grades; LOOP  FETCH  high\_math\_grades INTO student\_id;  EXIT WHEN  high\_math\_grades%NO TFOUND;  DBMS\_OUTPUT.PUT\_LI  NE('Student ID: ' || student\_id);  END LOOP; CLOSE  high\_math\_grades;  END; | DECLARE  CURSOR high\_math\_grades(student\_id NUMBER) IS SELECT student\_id  FROM student  WHERE subject = 'Mathematics' AND grade >= 90; BEGIN  FOR rec IN high\_math\_grades(90) LOOP DBMS\_OUTPUT.PUT\_LINE('Student ID: ' ||  rec.student\_id); END LOOP; END; | B |
| 147 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to find and print the student with the highest grade for each subject. Which of the following code snippets accomplishes this task? | DECLARE  CURSOR subject\_cursor IS  SELECT DISTINCT  subject  FROM student; BEGIN  FOR subject\_rec IN subject\_cursor LOOP  -- Find and print the student with the highest grade for each subject END LOOP;  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; highest\_grade NUMBER;  BEGIN  FOR subject\_rec IN subject\_cursor LOOP  -- Calculate and store the highest grade for each subject in highest\_grade  DBMS\_OUTPUT.PUT\_  LINE('Subject: ' || subject\_rec.subject  || ', Highest Grade: '  || highest\_grade); END LOOP;  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; student\_recordstudent  %ROWTYPE; BEGIN  FOR subject\_rec IN subject\_cursor LOOP  -- Find and print the student with the highest grade for each subject END LOOP;  END; | DECLARE  CURSOR subject\_cursor IS SELECT DISTINCT subject FROM student;  BEGIN  FOR subject\_rec IN subject\_cursor LOOP  -- Find and print the student with the highest grade for each subject  END LOOP; END; | A |
| 148 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to find and print the students who have improved their grades in at least one subject compared to the previous year. Which of the following code snippets accomplishes this task? | DECLARE  CURSOR student\_cursor IS  SELECT DISTINCT  student\_id FROM student;  BEGIN  -- Find and print students who have improved their grades  END; | DECLARE CURSOR  student\_cursor IS SELECT DISTINCT  student\_id FROM student;  improved\_students VARCHAR2(4000); BEGIN  -- Calculate and store the list of improved students in improved\_students  DBMS\_OUTPUT.PUT\_  LINE('Improved Students: ' || improved\_students); END; | DECLARE CURSOR  student\_cursor IS SELECT DISTINCT  student\_id FROM student;  student\_recordstudent  %ROWTYPE; BEGIN  -- Find and print students who have improved their grades END; | DECLARE  CURSOR student\_cursor IS SELECT DISTINCT student\_id FROM student;  BEGIN  -- Find and print students who have improved their grades  END; | B |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 149 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to find and print the students who have not improved their grades in any subject compared to the previous year. Which of the following code snippets accomplishes this task? | DECLARE  CURSOR student\_cursor IS  SELECT DISTINCT  student\_id FROM student;  BEGIN  -- Find and print students who have not improved their grades  END; | DECLARE CURSOR  student\_cursor IS SELECT DISTINCT  student\_id FROM student;  not\_improved\_stude nts VARCHAR2(4000); BEGIN  -- Calculate and store the list of students who have not improved their grades in not\_improved\_stude nts  DBMS\_OUTPUT.PUT\_  LINE('Students Who Have Not Improved Their Grades: ' || not\_improved\_stude nts);  END; | DECLARE CURSOR  student\_cursor IS SELECT DISTINCT  student\_id FROM student;  student\_recordstudent  %ROWTYPE; BEGIN  -- Find and print students who have not improved their grades END; | DECLARE  CURSOR student\_cursor IS SELECT DISTINCT student\_id FROM student;  BEGIN  -- Find and print students who have not improved their grades  END; | B |
| 150 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to find and print the students who have not improved their grades in any subject compared to the previous year. Which of the following code snippets accomplishes this task? | DECLARE  CURSOR student\_cursor IS  SELECT DISTINCT  student\_id FROM student;  BEGIN  -- Find and print students who have not improved their grades  END; | DECLARE CURSOR  student\_cursor IS SELECT DISTINCT  student\_id FROM student;  not\_improved\_stude nts VARCHAR2(4000); BEGIN  -- Calculate and store the list of students who have not improved their grades in not\_improved\_stude nts  DBMS\_OUTPUT.PUT\_  LINE('Students Who Have Not Improved Their Grades: ' || not\_improved\_stude nts);  END; | DECLARE CURSOR  student\_cursor IS SELECT DISTINCT  student\_id FROM student;  student\_recordstudent  %ROWTYPE; BEGIN  -- Find and print students who have not improved their grades END; | DECLARE  CURSOR student\_cursor IS SELECT DISTINCT student\_id FROM student;  BEGIN  -- Find and print students who have not improved their grades  END; | B |
| 151 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to find and print the students who have not improved their grades in any subject compared to the previous year. Which of the following code snippets accomplishes this task? | DECLARE  CURSOR student\_cursor IS  SELECT DISTINCT  student\_id FROM student;  BEGIN  -- Find and print students who have not improved their grades  END; | DECLARE CURSOR  student\_cursor IS SELECT DISTINCT  student\_id FROM student;  not\_improved\_stude nts VARCHAR2(4000); BEGIN  -- Calculate and store the list of students who have not improved their grades in not\_improved\_stude nts  DBMS\_OUTPUT.PUT\_  LINE('Students Who Have Not Improved Their Grades: ' || not\_improved\_stude nts);  END; | DECLARE CURSOR  student\_cursor IS SELECT DISTINCT  student\_id FROM student;  student\_recordstudent  %ROWTYPE; BEGIN  -- Find and print students who have not improved their grades END; | DECLARE  CURSOR student\_cursor IS SELECT DISTINCT student\_id FROM student;  BEGIN  -- Find and print students who have not improved their grades  END; | B |
| 152 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to find and print the students who have not improved their grades in any subject compared to the previous year. Which of the following code snippets accomplishes this task? | DECLARE  CURSOR student\_cursor IS  SELECT DISTINCT  student\_id FROM student;  BEGIN  -- Find and print students who have not improved their grades  END; | DECLARE CURSOR  student\_cursor IS SELECT DISTINCT  student\_id FROM student;  not\_improved\_stude nts VARCHAR2(4000); BEGIN  -- Calculate and store the list of students who have not improved their grades in not\_improved\_stude nts  DBMS\_OUTPUT.PUT\_  LINE('Students Who Have Not Improved Their Grades: ' || not\_improved\_stude nts);  END; | DECLARE CURSOR  student\_cursor IS SELECT DISTINCT  student\_id FROM student;  student\_recordstudent  %ROWTYPE; BEGIN  -- Find and print students who have not improved their grades END; | DECLARE  CURSOR student\_cursor IS SELECT DISTINCT student\_id FROM student;  BEGIN  -- Find and print students who have not improved their grades  END; | B |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 153 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to find and print the students who have not improved their grades in any subject compared to the previous year. Which of the following code snippets accomplishes this task? | DECLARE  CURSOR student\_cursor IS  SELECT DISTINCT  student\_id FROM student;  BEGIN  -- Find and print students who have not improved their grades  END; | DECLARE CURSOR  student\_cursor IS SELECT DISTINCT  student\_id FROM student;  not\_improved\_stude nts VARCHAR2(4000); BEGIN  -- Calculate and store the list of students who have not improved their grades in not\_improved\_stude nts  DBMS\_OUTPUT.PUT\_  LINE('Students Who Have Not Improved Their Grades: ' || not\_improved\_stude nts);  END; | DECLARE CURSOR  student\_cursor IS SELECT DISTINCT  student\_id FROM student;  student\_recordstudent  %ROWTYPE; BEGIN  -- Find and print students who have not improved their grades END; | DECLARE  CURSOR student\_cursor IS SELECT DISTINCT student\_id FROM student;  BEGIN  -- Find and print students who have not improved their grades  END; | B |
| 154 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to find and print the students who have not improved their grades in any subject compared to the previous year. Which of the following code snippets accomplishes this task? | DECLARE  CURSOR student\_cursor IS  SELECT DISTINCT  student\_id FROM student;  BEGIN  -- Find and print students who have not improved their grades  END; | DECLARE CURSOR  student\_cursor IS SELECT DISTINCT  student\_id FROM student;  not\_improved\_stude nts VARCHAR2(4000); BEGIN  -- Calculate and store the list of students who have not improved their grades in not\_improved\_stude nts  DBMS\_OUTPUT.PUT\_  LINE('Students Who Have Not Improved Their Grades: ' || not\_improved\_stude nts);  END; | DECLARE CURSOR  student\_cursor IS SELECT DISTINCT  student\_id FROM student;  student\_recordstudent  %ROWTYPE; BEGIN  -- Find and print students who have not improved their grades END; | DECLARE  CURSOR student\_cursor IS SELECT DISTINCT student\_id FROM student;  BEGIN  -- Find and print students who have not improved their grades  END; | B |
| 155 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to find and print the students who have not improved their grades in any subject compared to the previous year. Which of the following code snippets accomplishes this task? | DECLARE  CURSOR student\_cursor IS  SELECT DISTINCT  student\_id FROM student;  BEGIN  -- Find and print students who have not improved their grades  END; | DECLARE CURSOR  student\_cursor IS SELECT DISTINCT  student\_id FROM student;  not\_improved\_stude nts VARCHAR2(4000); BEGIN  -- Calculate and store the list of students who have not improved their grades in not\_improved\_stude nts  DBMS\_OUTPUT.PUT\_  LINE('Students Who Have Not Improved Their Grades: ' || not\_improved\_stude nts);  END; | DECLARE CURSOR  student\_cursor IS SELECT DISTINCT  student\_id FROM student;  student\_recordstudent  %ROWTYPE; BEGIN  -- Find and print students who have not improved their grades END; | DECLARE  CURSOR student\_cursor IS SELECT DISTINCT student\_id FROM student;  BEGIN  -- Find and print students who have not improved their grades  END; | B |
| 156 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to find and print the students who have scored the highest grade in at least one subject and the lowest grade in at least one subject. Which of the following code snippets accomplishes this task? | DECLARE  CURSOR student\_cursor IS  SELECT DISTINCT  student\_id FROM student;  BEGIN  -- Find and print students with highest and lowest grades in subjects  END; | DECLARE CURSOR  student\_cursor IS SELECT DISTINCT  student\_id FROM student;  students\_with\_extre me\_grades VARCHAR2(4000); BEGIN  -- Calculate and store the list of students with extreme grades in students\_with\_extre me\_grades  DBMS\_OUTPUT.PUT\_  LINE('Students with Extreme Grades: ' || students\_with\_extre me\_grades);  END; | DECLARE CURSOR  student\_cursor IS SELECT DISTINCT  student\_id FROM student;  student\_recordstudent  %ROWTYPE; BEGIN  -- Find and print students with highest and lowest grades in subjects  END; | DECLARE  CURSOR student\_cursor IS SELECT DISTINCT student\_id FROM student;  BEGIN  -- Find and print students with highest and lowest grades in subjects  END; | B |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 157 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to find and print the students who have scored the highest grade in each subject. Which of the following code snippets accomplishes this task? | DECLARE  CURSOR subject\_cursor IS  SELECT DISTINCT  subject  FROM student; BEGIN  -- Find and print students with the highest grade in each subject  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; highest\_grade NUMBER;  BEGIN  -- Calculate and store the highest grade for each subject in highest\_grade  DBMS\_OUTPUT.PUT\_  LINE('Students with Highest Grade in Each Subject: ' || highest\_grade);  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; student\_recordstudent  %ROWTYPE; BEGIN  -- Find and print students with the highest grade in each subject  END; | DECLARE  CURSOR subject\_cursor IS SELECT DISTINCT subject FROM student;  BEGIN  -- Find and print students with the highest grade in each subject  END; | B |
| 158 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to find and print the students who have scored the highest grade in each subject. Which of the following code snippets accomplishes this task? | DECLARE  CURSOR subject\_cursor IS  SELECT DISTINCT  subject  FROM student; BEGIN  -- Find and print students with the highest grade in each subject  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; highest\_grade NUMBER;  BEGIN  -- Calculate and store the highest grade for each subject in highest\_grade  DBMS\_OUTPUT.PUT\_  LINE('Students with Highest Grade in Each Subject: ' || highest\_grade);  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; student\_recordstudent  %ROWTYPE; BEGIN  -- Find and print students with the highest grade in each subject  END; | DECLARE  CURSOR subject\_cursor IS SELECT DISTINCT subject FROM student;  BEGIN  -- Find and print students with the highest grade in each subject  END; | B |
| 159 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to find and print the students who have scored the highest grade in each subject. Which of the following code snippets accomplishes this task? | DECLARE  CURSOR subject\_cursor IS  SELECT DISTINCT  subject  FROM student; BEGIN  -- Find and print students with the highest grade in each subject  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; highest\_grade NUMBER;  BEGIN  -- Calculate and store the highest grade for each subject in highest\_grade  DBMS\_OUTPUT.PUT\_  LINE('Students with Highest Grade in Each Subject: ' || highest\_grade);  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; student\_recordstudent  %ROWTYPE; BEGIN  -- Find and print students with the highest grade in each subject  END; | DECLARE  CURSOR subject\_cursor IS SELECT DISTINCT subject FROM student;  BEGIN  -- Find and print students with the highest grade in each subject  END; | B |
| 160 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to find and print the students who have scored the highest grade in each subject. Which of the following code snippets accomplishes this task? | DECLARE  CURSOR subject\_cursor IS  SELECT DISTINCT  subject  FROM student; BEGIN  -- Find and print students with the highest grade in each subject  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; highest\_grade NUMBER;  BEGIN  -- Calculate and store the highest grade for each subject in highest\_grade  DBMS\_OUTPUT.PUT\_  LINE('Students with Highest Grade in Each Subject: ' || highest\_grade);  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; student\_recordstudent  %ROWTYPE; BEGIN  -- Find and print students with the highest grade in each subject  END; | DECLARE  CURSOR subject\_cursor IS SELECT DISTINCT subject FROM student;  BEGIN  -- Find and print students with the highest grade in each subject  END; | B |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 161 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to find and print the students who have scored the highest grade in each subject. Which of the following code snippets accomplishes this task? | DECLARE  CURSOR subject\_cursor IS  SELECT DISTINCT  subject  FROM student; BEGIN  -- Find and print students with the highest grade in each subject  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; highest\_grade NUMBER;  BEGIN  -- Calculate and store the highest grade for each subject in highest\_grade  DBMS\_OUTPUT.PUT\_  LINE('Students with Highest Grade in Each Subject: ' || highest\_grade);  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; student\_recordstudent  %ROWTYPE; BEGIN  -- Find and print students with the highest grade in each subject  END; | DECLARE  CURSOR subject\_cursor IS SELECT DISTINCT subject FROM student;  BEGIN  -- Find and print students with the highest grade in each subject  END; | B |
| 162 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to find and print the students who have scored the lowest grade in each subject. Which of the following code snippets accomplishes this task? | DECLARE  CURSOR subject\_cursor IS  SELECT DISTINCT  subject  FROM student; BEGIN  -- Find and print students with the lowest grade in each subject  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; lowest\_grade NUMBER;  BEGIN  -- Calculate and store the lowest grade for each subject in lowest\_grade  DBMS\_OUTPUT.PUT\_  LINE('Students with Lowest Grade in Each Subject: ' || lowest\_grade);  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; student\_recordstudent  %ROWTYPE; BEGIN  -- Find and print students with the lowest grade in each subject  END; | DECLARE  CURSOR subject\_cursor IS SELECT DISTINCT subject FROM student;  BEGIN  -- Find and print students with the lowest grade in each subject  END; | B |
| 163 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to find and print the students who have scored the lowest grade in each subject. Which of the following code snippets accomplishes this task? | DECLARE  CURSOR subject\_cursor IS  SELECT DISTINCT  subject  FROM student; BEGIN  -- Find and print students with the lowest grade in each subject  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; lowest\_grade NUMBER;  BEGIN  -- Calculate and store the lowest grade for each subject in lowest\_grade  DBMS\_OUTPUT.PUT\_  LINE('Students with Lowest Grade in Each Subject: ' || lowest\_grade);  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; student\_recordstudent  %ROWTYPE; BEGIN  -- Find and print students with the lowest grade in each subject  END; | DECLARE  CURSOR subject\_cursor IS SELECT DISTINCT subject FROM student;  BEGIN  -- Find and print students with the lowest grade in each subject  END; | B |
| 164 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to find and print the students who have scored the same grade in all subjects. Which of the following code snippets accomplishes this task? | DECLARE  CURSOR student\_cursor IS  SELECT DISTINCT  student\_id FROM student;  BEGIN  -- Find and print students who have scored the same grade in all subjects END; | DECLARE CURSOR  student\_cursor IS SELECT DISTINCT  student\_id FROM student;  students\_with\_same  \_grade VARCHAR2(4000); BEGIN  -- Calculate and store the list of students with the same grade in all subjects in students\_with\_same  \_grade  DBMS\_OUTPUT.PUT\_  LINE('Students with Same Grade in All Subjects: ' || students\_with\_same  \_grade); END; | DECLARE CURSOR  student\_cursor IS SELECT DISTINCT  student\_id FROM student;  student\_recordstudent  %ROWTYPE; BEGIN  -- Find and print students who have scored the same grade in all subjects  END; | DECLARE  CURSOR student\_cursor IS SELECT DISTINCT student\_id FROM student;  BEGIN  -- Find and print students who have scored the same grade in all subjects  END; | B |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 165 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to find and print the students who have scored the same grade in all subjects. Which of the following code snippets accomplishes this task? | DECLARE  CURSOR student\_cursor IS  SELECT DISTINCT  student\_id FROM student;  BEGIN  -- Find and print students who have scored the same grade in all subjects END; | DECLARE CURSOR  student\_cursor IS SELECT DISTINCT  student\_id FROM student;  students\_with\_same  \_grade VARCHAR2(4000); BEGIN  -- Calculate and store the list of students with the same grade in all subjects in students\_with\_same  \_grade  DBMS\_OUTPUT.PUT\_  LINE('Students with Same Grade in All Subjects: ' || students\_with\_same  \_grade); END; | DECLARE CURSOR  student\_cursor IS SELECT DISTINCT  student\_id FROM student;  student\_recordstudent  %ROWTYPE; BEGIN  -- Find and print students who have scored the same grade in all subjects  END; | DECLARE  CURSOR student\_cursor IS SELECT DISTINCT student\_id FROM student;  BEGIN  -- Find and print students who have scored the same grade in all subjects  END; | B |
| 166 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to find and print the subject with the highest overall grades (sum of grades for all students). Which of the following code snippets accomplishes this task? | DECLARE  CURSOR subject\_cursor IS  SELECT DISTINCT  subject  FROM student; BEGIN  -- Find and print the subject with the highest overall grades  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; highest\_subject VARCHAR2(100); BEGIN  -- Calculate and store the subject with the highest overall grades in highest\_subject  DBMS\_OUTPUT.PUT\_  LINE('Subject with Highest Overall Grades: ' || highest\_subject); END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; subject\_recordstudent% ROWTYPE;  BEGIN  -- Find and print the subject with the highest overall grades  END; | DECLARE  CURSOR subject\_cursor IS SELECT DISTINCT subject FROM student;  BEGIN  -- Find and print the subject with the highest overall grades  END; | B |
| 167 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to find and print the subject with the lowest average grade. Which of the following code snippets accomplishes this task? | DECLARE  CURSOR subject\_cursor IS  SELECT DISTINCT  subject  FROM student; BEGIN  -- Find and print the subject with the lowest average grade  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; lowest\_avg\_grade NUMBER;  BEGIN  -- Calculate and store the lowest average grade for a subject in lowest\_avg\_grade  DBMS\_OUTPUT.PUT\_  LINE('Subject with Lowest Average Grade: ' || lowest\_avg\_grade); END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; subject\_recordstudent% ROWTYPE;  BEGIN  -- Find and print the subject  with the lowest average grade  END; | DECLARE  CURSOR subject\_cursor IS SELECT DISTINCT subject FROM student;  BEGIN  -- Find and print the subject with the lowest average grade  END; | B |
| 168 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to find and print the subjects in which all students have scored above a specified threshold (e.g., 60). Which of the following code snippets accomplishes this task? | DECLARE  CURSOR subject\_cursor IS  SELECT DISTINCT  subject  FROM student; BEGIN  -- Find and print subjects with all scores above the threshold  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; subjects\_with\_all\_hig h\_scores VARCHAR2(4000); BEGIN  -- Calculate and store the list of subjects with all high scores in subjects\_with\_all\_hig h\_scores  DBMS\_OUTPUT.PUT\_  LINE('Subjects with All High Scores: ' || subjects\_with\_all\_hig h\_scores);  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; subject\_recordstudent% ROWTYPE;  BEGIN  -- Find and print subjects with all scores above the threshold END; | DECLARE  CURSOR subject\_cursor IS SELECT DISTINCT subject FROM student;  BEGIN  -- Find and print subjects with all scores above the threshold  END; | B |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 169 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to find and print the subjects in which all students have scored above a specified threshold (e.g., 70). Which of the following code snippets accomplishes this task? | DECLARE  CURSOR subject\_cursor IS  SELECT DISTINCT  subject  FROM student; BEGIN  -- Find and print subjects with all scores above the threshold  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; subjects\_with\_all\_hig h\_scores VARCHAR2(4000); BEGIN  -- Calculate and store the list of subjects with all high scores in subjects\_with\_all\_hig h\_scores  DBMS\_OUTPUT.PUT\_  LINE('Subjects with All High Scores: ' || subjects\_with\_all\_hig h\_scores);  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; subject\_recordstudent% ROWTYPE;  BEGIN  -- Find and print subjects with all scores above the threshold END; | DECLARE  CURSOR subject\_cursor IS SELECT DISTINCT subject FROM student;  BEGIN  -- Find and print subjects with all scores above the threshold  END; | B |
| 170 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to find and print the subjects in which at least one student has scored below a specified threshold (e.g., 40). Which of the following code snippets accomplishes this task? | DECLARE  CURSOR subject\_cursor IS  SELECT DISTINCT  subject  FROM student; BEGIN  -- Find and print subjects with at least one score below the threshold  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; subjects\_with\_low\_s cores VARCHAR2(4000); BEGIN  -- Calculate and store the list of subjects with at least one low score in subjects\_with\_low\_s cores  DBMS\_OUTPUT.PUT\_  LINE('Subjects with Low Scores: ' || subjects\_with\_low\_s cores);  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; subject\_recordstudent% ROWTYPE;  BEGIN  -- Find and print subjects with at least one score below the threshold  END; | DECLARE  CURSOR subject\_cursor IS SELECT DISTINCT subject FROM student;  BEGIN  -- Find and print subjects with at least one score below the threshold  END; | A |
| 171 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to find and print the subjects in which at least one student has scored below a specified threshold (e.g., 50). Which of the following code snippets accomplishes this task? | DECLARE  CURSOR subject\_cursor IS  SELECT DISTINCT  subject  FROM student; BEGIN  -- Find and print subjects with at least one score below the threshold  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; subjects\_with\_low\_s cores VARCHAR2(4000); BEGIN  -- Calculate and store the list of subjects with at least one low score in subjects\_with\_low\_s cores  DBMS\_OUTPUT.PUT\_  LINE('Subjects with Low Scores: ' || subjects\_with\_low\_s cores);  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; subject\_recordstudent% ROWTYPE;  BEGIN  -- Find and print subjects with at least one score below the threshold  END; | DECLARE  CURSOR subject\_cursor IS SELECT DISTINCT subject FROM student;  BEGIN  -- Find and print subjects with at least one score below the threshold  END; | B |
| 172 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to find and print the subjects in which at least one student has scored below a specified threshold (e.g., 50). Which of the following code snippets accomplishes this task? | DECLARE  CURSOR subject\_cursor IS  SELECT DISTINCT  subject  FROM student; BEGIN  -- Find and print subjects with at least one score below the threshold  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; subjects\_with\_low\_s cores VARCHAR2(4000); BEGIN  -- Calculate and store the list of subjects with at least one low score in subjects\_with\_low\_s cores  DBMS\_OUTPUT.PUT\_  LINE('Subjects with Low Scores: ' || subjects\_with\_low\_s cores);  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; subject\_recordstudent% ROWTYPE;  BEGIN  -- Find and print subjects with at least one score below the threshold  END; | DECLARE  CURSOR subject\_cursor IS SELECT DISTINCT subject FROM student;  BEGIN  -- Find and print subjects with at least one score below the threshold  END; | A |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 173 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to find and print the subjects in which at least one student has scored below a specified threshold (e.g., 55). Which of the following code snippets accomplishes this task? | DECLARE  CURSOR subject\_cursor IS  SELECT DISTINCT  subject  FROM student; BEGIN  -- Find and print subjects with at least one score below the threshold  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; subjects\_with\_low\_s cores VARCHAR2(4000); BEGIN  -- Calculate and store the list of subjects with at least one low score in subjects\_with\_low\_s cores  DBMS\_OUTPUT.PUT\_  LINE('Subjects with Low Scores: ' || subjects\_with\_low\_s cores);  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; subject\_recordstudent% ROWTYPE;  BEGIN  -- Find and print subjects with at least one score below the threshold  END; | DECLARE  CURSOR subject\_cursor IS SELECT DISTINCT subject FROM student;  BEGIN  -- Find and print subjects with at least one score below the threshold  END; | B |
| 174 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to find and print the subjects in which at least one student has scored below a specified threshold (e.g., 60). Which of the following code snippets accomplishes this task? | DECLARE  CURSOR subject\_cursor IS  SELECT DISTINCT  subject  FROM student; BEGIN  -- Find and print subjects with at least one score below the threshold  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; subjects\_with\_low\_s cores VARCHAR2(4000); BEGIN  -- Calculate and store the list of subjects with at least one low score in subjects\_with\_low\_s cores  DBMS\_OUTPUT.PUT\_  LINE('Subjects with Low Scores: ' || subjects\_with\_low\_s cores);  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; subject\_recordstudent% ROWTYPE;  BEGIN  -- Find and print subjects with at least one score below the threshold  END; | DECLARE  CURSOR subject\_cursor IS SELECT DISTINCT subject FROM student;  BEGIN  -- Find and print subjects with at least one score below the threshold  END; | B |
| 175 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to find and print the subjects in which every student has scored above a specified threshold (e.g., 85). Which of the following code snippets accomplishes this task? | DECLARE  CURSOR subject\_cursor IS  SELECT DISTINCT  subject  FROM student; BEGIN  -- Find and print subjects with all scores above the threshold  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; subjects\_with\_all\_hig h\_scores VARCHAR2(4000); BEGIN  -- Calculate and store the list of subjects with all high scores in subjects\_with\_all\_hig h\_scores  DBMS\_OUTPUT.PUT\_  LINE('Subjects with All High Scores: ' || subjects\_with\_all\_hig h\_scores);  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; subject\_recordstudent% ROWTYPE;  BEGIN  -- Find and print subjects with all scores above the threshold END; | DECLARE  CURSOR subject\_cursor IS SELECT DISTINCT subject FROM student;  BEGIN  -- Find and print subjects with all scores above the threshold  END; | B |
| 176 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to find and print the subjects in which every student has scored above a specified threshold (e.g., 90). Which of the following code snippets accomplishes this task? | DECLARE  CURSOR subject\_cursor IS  SELECT DISTINCT  subject  FROM student; BEGIN  -- Find and print subjects with all scores above the threshold  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; subjects\_with\_all\_hig h\_scores VARCHAR2(4000); BEGIN  -- Calculate and store the list of subjects with all high scores in subjects\_with\_all\_hig h\_scores  DBMS\_OUTPUT.PUT\_  LINE('Subjects with All High Scores: ' || subjects\_with\_all\_hig h\_scores);  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; subject\_recordstudent% ROWTYPE;  BEGIN  -- Find and print subjects with all scores above the threshold END; | DECLARE  CURSOR subject\_cursor IS SELECT DISTINCT subject FROM student;  BEGIN  -- Find and print subjects with all scores above the threshold  END; | B |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 177 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to find and print the subjects in which no student has scored below a specified threshold (e.g., 60). Which of the following code snippets accomplishes this task? | DECLARE  CURSOR subject\_cursor IS  SELECT DISTINCT  subject  FROM student; BEGIN  -- Find and print subjects with no scores below the threshold  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; subjects\_with\_no\_lo w\_scores VARCHAR2(4000); BEGIN  -- Calculate and store the list of subjects with no low scores in subjects\_with\_no\_lo w\_scores  DBMS\_OUTPUT.PUT\_  LINE('Subjects with No Low Scores: ' || subjects\_with\_no\_lo w\_scores);  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; subject\_recordstudent% ROWTYPE;  BEGIN  -- Find and print subjects with no scores below the threshold END; | DECLARE  CURSOR subject\_cursor IS SELECT DISTINCT subject FROM student;  BEGIN  -- Find and print subjects with no scores below the threshold  END; | B |
| 178 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to find and print the subjects in which the average grade is above a specified threshold (e.g., 75). Which of the following code snippets accomplishes this task? | DECLARE  CURSOR subject\_cursor IS  SELECT DISTINCT  subject  FROM student; BEGIN  -- Find and print subjects with an average grade above the threshold  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; subjects\_above\_thre shold VARCHAR2(4000); BEGIN  -- Calculate and store the list of subjects with an average grade above the threshold in subjects\_above\_thre shold  DBMS\_OUTPUT.PUT\_  LINE('Subjects with Average Grade Above Threshold: ' || subjects\_above\_thre shold);  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; subject\_recordstudent% ROWTYPE;  BEGIN  -- Find and print subjects with an average grade above the threshold  END; | DECLARE  CURSOR subject\_cursor IS SELECT DISTINCT subject FROM student;  BEGIN  -- Find and print subjects with an average grade above the threshold  END; | B |
| 179 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to find and print the subjects in which the average grade is above a specified threshold (e.g., 75). Which of the following code snippets accomplishes this task? | DECLARE  CURSOR subject\_cursor IS  SELECT DISTINCT  subject  FROM student; BEGIN  -- Find and print subjects with an average grade above the threshold  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; subjects\_above\_thre shold VARCHAR2(4000); BEGIN  -- Calculate and store the list of subjects with an average grade above the threshold in subjects\_above\_thre shold  DBMS\_OUTPUT.PUT\_  LINE('Subjects with Average Grade Above Threshold: ' || subjects\_above\_thre shold);  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; subject\_recordstudent% ROWTYPE;  BEGIN  -- Find and print subjects with an average grade above the threshold  END; | DECLARE  CURSOR subject\_cursor IS SELECT DISTINCT subject FROM student;  BEGIN  -- Find and print subjects with an average grade above the threshold  END; | B |
| 180 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to find and print the subjects in which the average grade is above a specified threshold (e.g., 80). Which of the following code snippets accomplishes this task? | DECLARE  CURSOR subject\_cursor IS  SELECT DISTINCT  subject  FROM student; BEGIN  -- Find and print subjects with an average grade above the threshold  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; subjects\_above\_thre shold VARCHAR2(4000); BEGIN  -- Calculate and store the list of subjects with an average grade above the threshold in subjects\_above\_thre shold  DBMS\_OUTPUT.PUT\_  LINE('Subjects with Average Grade Above Threshold: ' || subjects\_above\_thre shold);  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; subject\_recordstudent% ROWTYPE;  BEGIN  -- Find and print subjects with an average grade above the threshold  END; | DECLARE  CURSOR subject\_cursor IS SELECT DISTINCT subject FROM student;  BEGIN  -- Find and print subjects with an average grade above the threshold  END; | B |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 181 | Given a "student" table with columns `student\_id`, `subject`, and `grade`, write a PL/SQL block that uses a cursor to find and print the subjects in which the average grade is below a specified threshold (e.g., 70). Which of the following code snippets accomplishes this task? | DECLARE  CURSOR subject\_cursor IS  SELECT DISTINCT  subject  FROM student; BEGIN  -- Find and print subjects with an average grade below the threshold  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; subjects\_below\_thre shold VARCHAR2(4000); BEGIN  -- Calculate and store the list of subjects with an average grade below the threshold in subjects\_below\_thre shold  DBMS\_OUTPUT.PUT\_  LINE('Subjects with Average Grade Below Threshold: ' || subjects\_below\_thre shold);  END; | DECLARE CURSOR  subject\_cursor IS SELECT DISTINCT  subject  FROM student; subject\_recordstudent% ROWTYPE;  BEGIN  -- Find and print subjects with an average grade below the threshold  END; | DECLARE  CURSOR subject\_cursor IS SELECT DISTINCT subject FROM student;  BEGIN  -- Find and print subjects with an average grade below the threshold  END; | B |
| 182 | Given a "student" table with columns `student\_id`, `subject`, and `grade`,write a PL/SQL block that uses a cursor to find and print the students who have scored the same grade in all subjects. Which of the following code snippets accomplishes this task? | DECLARE  CURSOR student\_cursor IS  SELECT DISTINCT  student\_id FROM student;  BEGIN  -- Find and print students who have scored the same grade in all subjects END; | DECLARE CURSOR  student\_cursor IS SELECT DISTINCT  student\_id FROM student;  students\_with\_same  \_grade VARCHAR2(4000); BEGIN  -- Calculate and store the list of students with the same grade in all subjects in students\_with\_same  \_grade  DBMS\_OUTPUT.PUT\_  LINE('Students with Same Grade in All Subjects: ' || students\_with\_same  \_grade); END; | DECLARE CURSOR  student\_cursor IS SELECT DISTINCT  student\_id FROM student;  student\_recordstudent  %ROWTYPE; BEGIN  -- Find and print students who have scored the same grade in all subjects  END; | DECLARE  CURSOR student\_cursor IS SELECT DISTINCT student\_id FROM student;  BEGIN  -- Find and print students who have scored the same grade in all subjects  END; | B |
| 183 | How can concurrent access to shared data lead to data inconsistency in a DBMS? | A) By preventing data updates | B) By enforcing data integrity rules | C) By allowing simultaneous updates | D) By reducing query performance | C |
| 184 | How can you remove a package from the database in PL/SQL? | By using the DROP PACKAGE statement | By removing all the procedures and functions from the package body | By using the DELETE PACKAGE statement | By using the TRUNCATE PACKAGE statement | A |
| 185 | How many mandatory parts packages has? | 1 | 2 | 3 | 4 | B |
| 186 | In a DBMS, a package is typically used to: | Store and organize tables and views | Group related procedures, functions, and variables | Define user roles and permissions | Execute ad-hoc SQL queries | B |
| 187 | In a DBMS, what is a benefit of using stored procedures for business logic? | It allows for dynamic SQL execution. | **It centralizes and secures the business logic.** | It simplifies data retrieval operations. | It eliminates the need for data validation. | B |
| 188 | In a DBMS, what is a database trigger? | A procedure that runs when the database is created | A statement that rolls back database changes | A set of rules for data validation | **A piece of code that automatically executes in response to a specific event** | D |
| 189 | In a DBMS, what is the main purpose of a database trigger? | To enforce referential integrity constraints | To encapsulate business logic for data processing | To automatically generate primary keys | To record changes to data in response to events | D |
| 190 | In a DBMS, what is the primary purpose of a "SERIALIZABLE" isolation level? | To maximize concurrency | To prevent transactions from acquiring locks | To ensure that transactions execute in a specific order | **To provide the highest level of data consistency and isolation** | D |
| 191 | In a DBMS, what is the primary purpose of a database package body? | A) To define package variables | B) To provide information about the package's procedures and functions | C) To specify package triggers | D) To implement the actual code for package procedures | D |
| 192 | In a DBMS, what is the primary purpose of using "SELECT FOR UPDATE"? | To retrieve data for reporting purposes | **To lock rows, preventing other transactions from modifying them** | To retrieve data without acquiring locks | To roll back a transaction | B |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 193 | In a DBMS, what is the primary purpose of using a "READ COMMITTED" isolation level? | To minimize data redundancy | To ensure that transactions execute in a specific order | To prevent transactions from acquiring locks | **To balance data consistency and concurrency** | D |
| 194 | In a DBMS, what is the primary purpose of using an "AFTER UPDATE" trigger? | To prevent any updates from occurring | To execute before any update operation occurs | **To log changes made to a table after an update** | To execute after a row is updated in a table | C |
| 195 | In a DBMS, what is the primary purpose of using triggers? | To simplify data retrieval operations | **To enforce data integrity constraints** | To create temporary tables | To improve query performance | B |
| 196 | In a DBMS, what is the purpose of a "SAVE TRANSACTION" statement? | To save a transaction for later execution | To create a new transaction | **To save the current state of a transaction as a savepoint** | To commit the transaction | C |
| 197 | In a DBMS, what is the purpose of a savepoint? | To commit a transaction | To roll back a transaction | **To create a point within a transaction to which you can later roll back** | To lock a table temporarily | C |
| 198 | In a DBMS, what is the purpose of the "COMMIT" statement? | To start a new transaction | **To save all changes made within the current transaction** | To roll back all changes made within the current transaction | To create a new savepoint | B |
| 199 | In a DBMS, what is the purpose of the "ROLLBACK TO SAVEPOINT" statement? | To roll back an entire transaction | To create a new savepoint | **To roll back to a specific savepoint within a transaction** | To commit a transaction | C |
| 200 | In a DBMS, what is the purpose of the "SET TRANSACTION" statement? | **To define transaction isolation levels** | To start a new transaction | To commit a transaction | To roll back a transaction | A |
| 201 | In a DBMS, what is the purpose of the procedure header? | A) To declare the procedure's input and output variables | B) To specify the operations to be performed by the procedure | C) To implement the actual code for the procedure | D) To define the package specification | A |
| 202 | In a DBMS, what is the term for a package that only contains the package specification without a package body? | A) A minimal package | B) A comprehensive package | C) A bodiless package | D) A complete package | C |
| 203 | In a DBMS, what is the typical process of developing a package? | A) Write the package body first, then the specification | B) Write the package specification first, then the body | C) Develop procedures and triggers independently of packages | D) Develop triggers before procedures | B |
| 204 | In a DBMS, which clause is used to specify the action to be taken when a trigger event occurs? | EXECUTE | FOR EACH ROW | **WHEN** | INSTEAD OF | C |
| 205 | In a DBMS, which type of cursor is used to fetch and process one row at a time? | Static cursor | Dynamic cursor | **Forward-only cursor** | Scrollable cursor | C |
| 206 | In a locking-based concurrency control system, what does a "lock" prevent other transactions from doing? | A) Accessing the locked data | B) Aborting the transaction | C) Executing queries | D) Creating database triggers | A |
| 207 | In a multi-user DBMS, what issue can occur without proper concurrency control? | A) Faster query execution | B) Data consistency problems | C) Reduced code duplication | D) Increased code modularity | B |
| 208 | In a multi-user DBMS, what problem can arise when transactions are executed concurrently without control? | A) Faster data retrieval | B) Enhanced data consistency | C) Reduced code duplication | D) Increased code modularity | B |
| 209 | In Oracle PL/SQL, can a package contain both procedures and functions? | Yes, but they must all have the same name. | No, a package can only contain either procedures or functions, not both. | **Yes, a package can contain a mix of procedures and functions.** | Yes, but they must all be stored in separate packages. | C |
| 210 | In Oracle PL/SQL, what is a benefit of using packages over standalone procedures? | Packages are easier to write and debug. | **Packages allow you to encapsulate related code and data.** | Packages execute faster than standalone procedures. | Packages are not dependent on any database schema. | B |
| 211 | In Oracle PL/SQL, which statement is used to raise an exception explicitly within a stored procedure? | RAISE EXCEPTION | SIGNAL | **RAISE\_APPLICATION\_ER ROR** | THROW | C |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 212 | In PL/SQL, the CREATE TABLESPACE is used | To create a place in the database for storage of scheme objects, rollback segments, and naming the data files to comprise the table- space | To create a database trigger | To add/rename data files, to change storage | All of the above | A |
| 213 | In PL/SQL, which of the following statements accurately describes a view? | A view is a virtual table based on the result of a SELECT query. | A view is a physical table that stores data permanently. | A view is a temporary table used for transactional purposes. | A view is a table used only for indexing purposes. | A |
| 214 | In PL/SQL, which trigger event is fired when a column value is updated to NULL? | AFTER UPDATE NULL | BEFORE UPDATE NULL | AFTER UPDATE OF  column\_name | BEFORE UPDATE OF column\_name | C |
| 215 | In the context of a DBMS, what is the purpose of a stored procedure? | To store data in the database | To define the structure of a table | To encapsulate a series of SQL statements for reuse | To retrieve data from external sources | C |
| 216 | In the context of database transactions, what is a deadlock? | **A situation where two or more transactions are waiting for each other to release locks, preventing progress.** | A situation where a transaction reads uncommitted data. | A situation where a transaction violates integrity constraints. | A situation where a transaction is aborted due to a rollback request. | A |
| 217 | In the context of isolation levels in a DBMS, what does the "Read Uncommitted" isolation level allow? | Transactions can read uncommitted changes made by other transactions. | Transactions cannot read any data until all changes are committed. | Transactions can only read committed data. | Transactions can read their own uncommitted changes. | A |
| 218 | In the context of isolation levels in a DBMS, what does the "Read Uncommitted" isolation level allow? | **Transactions can read uncommitted changes made by other transactions.** | Transactions cannot read any data until all changes are committed. | Transactions can only read committed data. | Transactions can read their own uncommitted changes. | A |
| 219 | In the context of transactions, what does "serializability" mean? | A) The ability to serialize data | B) The ability to execute transactions concurrently | C) The ability to recover from failures | D) The ability to perform queries efficiently | A |
| 220 | OLD and NEW references are not available for table-level triggers. | TRUE | FALSE | Can be true or false | None of the above | A |
| 221 | and subprograms. | Yes | No | Can be yes or no | none of the above | A |
| 222 | PL/SQL controls the context area through a cursor. | TRUE | FALSE | Can be true or false | All of the above | A |
| 223 | the program | Try | Throw | Catch | Exception | D |
| 224 | Repeat  sequence of statements;    end repeat  Fill in the correct option : | While Condition | Until variable | Until boolean expression | Until 0 | C |
| 225 | Repeat  sequence of statements;    end repeat  Fill in the correct option : | While Condition | Until variable | Until boolean expression | Until 0 | C |
| 226 | Suppose a database system crashes again while recovering from a previous crash. Assume checkpointing is not done by the database either during the transactions or during recovery.  Which of the following statements is/are correct? | The same undo and redo list will be used while recovering again. | The database will become inconsistent. | All the transactions that are already undone and redone will not be recovered again. | The system cannot recover any further. | A |
| 227 | Temporary stored procedures are stored in database. | Master | Model | User specific | Tempdb | D |
| 228 | The Statement is used for creating the package body. | CREATE | CREATE PACKAGE | CREATE BODY | CREATE PACKAGE BODY | D |
| 229 | The Statement is used for creating the package body. | CREATE PACKAGE BODY | CREATE | CREAT BODY | CREATE BODY | A |
| 230 | The constructs of a procedure, function or a package are . | Variables and Constants | Cursors | Exceptions | All of the above | D |
| 231 | The CREATE TRIGGER statement is used to create the trigger. THE clause specifies the table name on which the trigger is to be attached. The specifies that this is an AFTER INSERT trigger. | for insert, on | On, for insert | For, insert | None of the mentioned | B |
| 232 | specifies the table name on which the trigger is to be attached. The specifies that this is an AFTER INSERT trigger. | for insert, on | On, for insert | For, insert | None of the mentioned | A |
| 233 | The format for compound statement is | Begin ……. end | Begin atomic……. end | Begin ……. repeat | Both Begin ……. end and Begin atomic……. end | D |
| 234 | The package specification is the interface to the package. | TRUE | FALSE | Nither TRUE NOR FALSE | none of the above | A |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 235 | The parameters can be passed as default also to the procedures and the functions. | TRUE | FALSE | Nither TRUE NOR FALSE | None of he above | A |
| 236 | The property of a schedule that states that the result of executing concurrent transactions is the same as executing them serially is known as: | Consistency | Atomicity | Serializability | Durability | C |
| 237 | The technique used to detect and resolve conflicts among concurrent transactions is called: | Two-phase locking | Timestamp ordering | Deadlock detection | Deadlock prevention | C |
| 238 | Triggers can be defined on the? | table | view | schema | All of the above | D |
| 239 | Triggers can be defined on the? | DDL | DML | Database Operation | All of the above | D |
| 240 | What does "recoverability" encompass in the context of transactions and concurrency control? | A) The ability to recover from system failures | B) The ability to execute transactions concurrently | C) The ability to lock database tables | D) The ability to perform efficient queries | A |
| 241 | What does DBA stand for in the context of databases? | A) Database Backup Administrator | B) Data Business Analyst | C) Database Architect | D) Database Administrator | D |
| 242 | What does the concept of "recoverability" in concurrency control refer to? | A) The ability to lock data | B) The ability to recover from system failures | C) The ability to perform database recovery | D) The ability to execute queries efficiently | B |
| 243 | What does the concept of "serializability" in concurrency control refer to? | A) The ability to lock data | B) The ability to execute transactions in parallel | C) The ability to perform database recovery | D) The ability to execute queries efficiently | B |
| 244 | What does the term "serializability" imply in the context of transaction execution? | A) Transactions occur in sequence | B) Transactions can execute concurrently | C) Transactions are aborted | D) Transactions are isolated | A |
| 245 | What does the term "transaction isolation" refer to in the context of concurrency control? | A) A transaction's lifespan | B) A transaction's ability to update data | C) A transaction's isolation level | D) A transaction's recovery | C |
| 246 | What is a "bodiless" package in a DBMS context? | A) A package without a body | B) A package with excessive code | C) A package with only triggers | D) A package with minimal documentation | A |
| 247 | What is a "bodiless" package in a DBMS context? | A) A package without a body | B) A package with excessive code | C) A package with only triggers | D) A package with minimal documentation | A |
| 248 | What is a "transaction" in the context of a database management system (DBMS)? | A) A data dictionary | B) A single unit of work | C) A database schema | D) A database connection | B |
| 249 | What is a common approach to resolving deadlocks in a DBMS? | Rolling back one of the transactions involved in the deadlock | Killing all transactions to release locks | Preventing transactions from acquiring locks | Using deadlock detection and resolution algorithms | D |
| 250 | What is a common drawback of "pessimistic" locking in concurrency control systems? | A) Increased code modularity | B) Reduced data consistency | C) Reduced query performance | D) Optimized query execution | C |
| 251 | What is a common use of a database trigger in a DBMS? | A) To define package specifications | B) To encapsulate related procedures and functions | C) To monitor and respond to database events | D) To create database packages | C |
| 252 | What is a database schema? | A) A collection of tables in a database | B) A diagram representing the structure of a database | C) A set of rules that define the database structure | D) A description of the database structure, including tables, fields, and relationships | D |
| 253 | What is a package body in a DBMS? | A) A part of a package that contains package variables | B) A part of a package that specifies the package's procedures and functions | C) A part of a package that defines package triggers | D) A part of a package that implements the actual code for package procedures | D |
| 254 | What is a package specification in a DBMS? | A) A part of a package that contains package variables | B) A part of a package that specifies the package's procedures and functions | C) A part of a package that defines package triggers | D) A part of a package that implements the actual code for package procedures | D |
| 255 | What is a parameterized stored procedure in a DBMS? | **A procedure that accepts parameters and returns a result set** | A procedure that uses a parameter as its name | A procedure that cannot accept any input parameters | A procedure that can only accept integer parameters | A |
| 256 | What is a transaction in a DBMS? | A database schema | A series of SQL statements | **A logical unit of work that is either fully completed or fully undone** | A data dictionary | C |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 257 | What is correct a PL/SQL program that create Trigger to update the "salary" of an employee to 80000 if the "department" is  changed to 'Management' | CREATE OR REPLACE TRIGGER  trg\_department\_update BEFORE UPDATE OF  department ON employee FOR EACH ROW  BEGIN  IF :NEW.department = 'Management' THEN  :NEW .salary := 80000;  END IF; END;  / | CREATE OR REPLACE TRIGGER  trg\_department\_updat e  BEFORE UPDATE  OF department ON employee  FOR EACH ROW IF  :NEW.department = 'Management' THEN  :NEW .salary := 80000;  END IF; END;  / | CREATE OR REPLACE TRI  trg\_department\_update BEFORE UPDATE OF  department ON employee  FOR EACH BEGIN  IF :NEW.department = 'Management' THEN  :NEW .salary := 80000;  END IF; END;  / | CREATE OR REPLACE TRIGGER trg\_department\_update UPDATE OF department ON employee  ROW BEGIN  IF :NEW.department = 'Management' THEN  :NEW .salary := 80000; END IF;  END;  / | A |
| 258 | What is correct a procedure that calculates and displays the total salary of employees in a given department.  The department name is an optional parameter with a default value of 'HR'. | CREATE OR REPLACE PROCEDURE  total\_salary\_by\_departme nt(p\_department\_name IN VARCHAR2 DEFAULT 'HR') AS  v\_total\_salary NUMBER; BEGIN  SELECT SUM(salary) INTO v\_total\_salary  FROM employee  WHERE department = p\_department\_name;  DBMS\_OUTPUT.PUT\_LIN  E('Total Salary for Department ' || p\_department\_name || ': '  || v\_total\_salary); EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LIN  E('An error occurred.'); END;  / | CREATE OR REPLACE PROCEDURE  total\_salary\_by\_depar tment(p\_department\_ name IN VARCHAR2 ') AS  v\_total\_salary NUMBER; BEGIN  SELECT  SUM(salary) INTO v\_total\_salary  FROM employee WHERE  department = p\_department\_name;  DBMS\_OUTPUT.PUT  \_LINE('Total Salary for Department ' || p\_department\_name  || ': ' || v\_total\_salary); EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT  \_LINE('An error occurred.'); END;  / | CREATE OR REPLACE PROCEDURE  total\_salary\_by\_departm ent(p\_department\_name IN VARCHAR2 DEFAULT 'HR') AS  v\_total\_salary NUMBER;  SELECT SUM(salary) INTO v\_total\_salary  FROM employee  WHERE department = p\_department\_name;  DBMS\_OUTPUT.PUT\_L  INE('Total Salary for Department ' || p\_department\_name || ': ' || v\_total\_salary); EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_L  INE('An error occurred.'); END;  / | CREATE OR REPLACE PROCEDURE  total\_salary\_by\_department(DEFAULT 'HR') AS v\_total\_salary NUMBER;  BEGIN  SELCT SUM(salary) INTO v\_total\_salary FROM employee  WHERE department = p\_department\_name;  DBMS\_OUTPUT.PUT\_LINE('Total Salary for Department ' || p\_department\_name || ': ' || v\_total\_salary); EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('An error occurred.'); END;  / | A |
| 259 | What is one of the advantages of using procedures in a DBMS? | A) Increased code duplication | B) Slower query performance | C) Enhanced security vulnerabilities | D) Reduced code redundancy | D |
| 260 | What is one of the advantages of using triggers in a DBMS? | A) Increased code modularity | B) Reduced control over data changes | C) Enhanced query performance | D) Automated enforcement of data integrity rules | D |
| 261 | What is Output-- Insert sample records into the "employee" table  INSERT INTO employee (employee\_id, first\_name, last\_name, department, salary) VALUES (1, 'John', 'Doe', 'HR', 50000);  INSERT INTO employee (employee\_id, first\_name, last\_name, department, salary) VALUES (2, 'Jane', 'Smith', 'Finance', 60000);  INSERT INTO employee (employee\_id, first\_name, last\_name, department, salary) VALUES (3, 'Michael', 'Johnson', 'IT', 70000);  INSERT INTO employee (employee\_id, first\_name, last\_name, department, salary) VALUES (4, 'Merry', 'Agarwal', 'IT', 50000);  CREATE OR REPLACE PROCEDURE delete\_employee\_by\_id(p\_employee\_id NUMBER) AS  BEGIN  DELETE FROM employee  WHERE employee\_id = p\_employee\_id;  IF SQL%ROWCOUNT > 0 THEN  DBMS\_OUTPUT.PUT\_LINE('Employee deleted successfully.'); ELSE  DBMS\_OUTPUT.PUT\_LINE('Employee ID not found. No employee deleted.'); END IF;  EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('An error occurred.'); END;  /  EXECUTE delete\_employee\_by\_id(6); | An error occurred. | Employee ID not found. No employee deleted. | Employee deleted successfully. | EXECUTE delete\_employee\_by\_id(2); | B |
| 262 | What is the ACID property that ensures that transactions are performed correctly and completely? | A) Atomicity | B) Consistency | C) Isolation | D) Durability | A |
| 263 | What is the key difference between stored procedures and functions in DBMS? | Stored procedures can return multiple values, while functions can only return a single value | Stored procedures can be executed by users, while functions can only be executed by the database administrator | Stored procedures are used for data manipulation, while functions are used for data retrieval | Stored procedures can be called from within other procedures, while functions cannot | A |
| 264 | What is the Oracle Error Code for ACCESS\_INTO\_NULL? | 6592 | 6531 | 1722 | 6530 | D |
| 265 | What is the output of the following program? DECLARE  A NUMBER :=2; BEGIN  FOR I IN 1..3 LOOP A := A\*2;  END LOOP; DBMS\_OUTPUT.PUT\_LINE(A);  END; | 4 | 8 | 16 | 32 | D |
| 266 | What is the primary advantage of using packages in a DBMS? | Improved query performance | Enhanced data security | Better code organization and reusability | Simplified database design | D |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 267 | What is the primary difference between a row-level trigger and a statement-level trigger? | Row-level triggers are executed before statement- level triggers. | Row-level triggers are fired once for each affected row, while statement-level triggers are fired once for each SQL statement. | Statement-level triggers can be defined on tables, whereas row-level triggers cannot. | Row-level triggers can be recursive, while statement-level triggers cannot. | B |
| 268 | What is the primary drawback of "optimistic" concurrency control in a DBMS? | A) Increased code modularity | B) Slower query performance | C) Risk of transaction conflicts | D) Enhanced data integrity | C |
| 269 | What is the primary goal of concurrency control in a DBMS? | To maximize data redundancy | To minimize database access | To ensure data consistency and integrity in a multi- user environment | To eliminate the need for indexing | C |
| 270 | What is the primary goal of concurrency control in a DBMS? | To maximize data redundancy | To minimize database access | **To ensure data consistency and integrity in a multi-user environment** | To eliminate the need for indexing | C |
| 271 | What is the primary purpose of "deadlock detection" mechanisms in a DBMS that uses locking for concurrency control? | A) To prevent transaction conflicts | B) To optimize query performance | C) To eliminate transactions | D) To create database triggers | C |
| 272 | What is the primary purpose of a "BEFORE DELETE" trigger in a DBMS? | **To execute before any delete operation occurs** | To execute after a row is deleted from a table | To prevent any delete operations from taking place | To execute only if a delete operation fails | A |
| 273 | What is the primary purpose of a cursor in a stored procedure? | To store the results of a query | **To iterate through the records returned by a query** | To create a temporary table | To enforce data integrity constraints | B |
| 274 | What is the primary purpose of a database package in a DBMS? | A) To define database triggers | B) To encapsulate and group related procedures, functions, and variables | C) To establish database connections | D) To optimize query performance | B |
| 275 | What is the primary purpose of a stored procedure in DBMS? | To store and organize data in a database | To retrieve data from the database | To define the structure of the database | To encapsulate a series of database operations | D |
| 276 | What is the primary purpose of an "AFTER INSERT" trigger in a DBMS? | To execute before any insert operation occurs | **To execute after a new row is inserted into a table** | To prevent any insert operations from taking place | To execute only if an insert operation fails | B |
| 277 | What is the primary purpose of locking in concurrency control? | A) To eliminate transactions | B) To optimize query performance | C) To manage data access | D) To create database triggers | C |
| 278 | What is the primary purpose of transaction management in a database system? | A) To optimize query performance | B) To ensure data consistency | C) To define database triggers | D) To establish database connections | B |
| 279 | What is the primary purpose of using packages in a DBMS? | To store data in tables | **To organize related procedures, functions, and variables** | To manage user permissions | To enforce data integrity constraints | B |
| 280 | What is the primary syntax for creating a trigger in a DBMS? | A) CREATE PROCEDURE | B) CREATE FUNCTION | C) CREATE TRIGGER | D) CREATE TABLE | C |
| 281 | What is the purpose of "recoverability" in the context of database transactions? | A) To ensure all transactions recover | B) To prevent data recovery issues | C) To recover from system failures | D) To lock database tables | C |
| 282 | What is the purpose of declaring a cursor in SQL? | To define a new table in the database | To specify the database connection string | **To define a result set for query execution** | To create a new user account | C |
| 283 | What is the purpose of the "ROLLBACK" statement in a stored procedure? | To commit all changes made within the procedure | **To undo all changes made within the procedure** | To restart the execution of the procedure from the beginning | To create a new savepoint within the procedure | B |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 284 | What is the purpose of the internal schema in a database system? | A) To define the logical view of the database for users | B) To specify the access controls and security settings for the database | C) To represent the physical storage structure of the database | D) To define the user views and queries for the database | C |
| 285 | What is the purpose of the JOIN operation in a relational database? | A) To add new records to a table | B) To remove records from a table | C) To combine data from multiple tables based on a related column | D) To modify existing records in a table | C |
| 286 | What is the purpose of the SAVEPOINT statement in DBMS? | To define the start of a transaction | To create a temporary table | To define a point within a transaction to which you can roll back | To release a lock on a database object | C |
| 287 | What is the role of a "transaction log" in a DBMS with respect to concurrency control? | A) To manage database locks | B) To record transaction history | C) To optimize query performance | D) To create database triggers | B |
| 288 | What is the role of a "transaction manager" in a DBMS? | A) To design the database | B) To manage database connections | C) To coordinate transaction execution | D) To create database triggers | C |
| 289 | What is the role of a package specification in a database package? | A) To define package variables | B) To provide information about the package's procedures and functions | C) To specify package triggers | D) To establish database connections | B |
| 290 | What is the role of the FETCH statement in SQL cursor operations? | It declares a new cursor. | **It retrieves rows from the result set and moves the cursor to the next row.** | It closes an open cursor. | It defines the structure of a cursor. | B |
| 291 | What is the role of the FETCH statement in SQL cursor operations? | It declares a new cursor. | **It retrieves rows from the result set and moves the cursor to the next row.** | It closes an open cursor. | It defines the structure of a cursor. | B |
| 292 | What is the syntax of User-defined exceptions? | DECLARE my-exception EXCEPTION; | DECLARE EXCEPTION; | DECLARE my-exception; | EXCEPTION; | A |
| 293 | What is the typical sequence of steps for developing a package in a DBMS? | A) Develop triggers first, then procedures | B) Develop procedures and package specification simultaneously | C) Write the package specification first, then the package body | D) Write the package body first, then the specification | C |
| 294 | What is the typical sequence of steps for developing a package in a DBMS? | A) Develop triggers first, then procedures | B) Develop procedures and package specification simultaneously | C) Write the package specification first, then the package body | D) Write the package body first, then the specification | C |
| 295 | What part of a procedure in a DBMS is responsible for declaring the input and output variables? | A) Procedure header | B) Procedure specification | C) Procedure body | D) Procedure parameters | B |
| 296 | What type of trigger is executed automatically after the triggering event? | A) After Trigger | B) Before Trigger | C) Instead of Trigger | D) Compound Trigger | A |
| 297 | What type of trigger is executed automatically before a specific event, such as an INSERT or UPDATE operation? | A) After Trigger | B) Before Trigger | C) Instead of Trigger | D) Compound Trigger | B |
| 298 | When executing a stored procedure, what keyword is commonly used to return a  result set to the calling application? | RESULT | RESULTSET | **OUTPUT** | RETURN | C |
| 299 | and the data doesn't get back, another transaction tries to access the updated  database item. | Rolled | Committed | Aborted | None | A |
| 300 | Which ACID property ensures that a transaction's effects on the database are permanent? | A) Atomicity | B) Consistency | C) Isolation | D) Durability | D |
| 301 | Which attribute is used to raise exception? | Open | Select | Raise | Try | C |
| 302 | Which attribute returns TRUE if an INSERT, UPDATE, or DELETE statement affected one or more rows? | %NOTFOUND | %ISOPEN | %ROWCOUNT | %FOUND | D |
| 303 | Which clause is used to create trigger on a view? | BEFORE | INSTEAD OF | AFTER | None of the above | B |
| 304 | Which component of a package in DBMS defines the interface and public entities? | Package body | Package Signature | Package Constructor | Package specification | D |
| 305 | Which concurrency control technique allows multiple transactions to read data simultaneously but enforces write locks to prevent data conflicts? | Two-Phase Locking (2PL) | Time-stamp Ordering | Multi-Version Concurrency Control (MVCC) | Optimistic Concurrency Control | A |
| 306 | Which concurrency control technique allows multiple transactions to read data simultaneously but enforces write locks to prevent data conflicts? | **Two-Phase Locking (2PL)** | Time-stamp Ordering | Multi-Version Concurrency Control (MVCC) | Optimistic Concurrency Control | A |
| 307 | Which cursor attribute can be used to determine the total number of rows returned by a cursor in PL/SQL? | %ROWCOUNT | %FOUND | %ISOPEN | %NOTFOUND | A |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 308 | Which database system component is responsible for managing transactions and ensuring data integrity? | A) Database schema | B) Data dictionary | C) Concurrency control manager | D) Query optimizer | C |
| 309 | Which isolation level allows only committed data to be read? | Read Uncommitted | Read committed | Serializable | Read Update | B |
| 310 | Which isolation level allows the highest concurrency but may result in non- repeatable reads? | Read Uncommitted | Read Committed | Repeatable Read | Serializable | A |
| 311 | Which isolation level provides the highest level of data consistency but can lead to reduced concurrency? | Read Uncommitted | Read Committed | Repeatable Read | Serializable | D |
| 312 | Which isolation level provides the highest level of data consistency but can lead to reduced concurrency? | Read Uncommitted | Read Committed | Repeatable Read | **Serializable** | D |
| 313 | Which keyword is used to create a new package body in PL/SQL? | BODY | IMPLEMENTATION | DEFINE | CREATE | A |
| 314 | Which keyword is used to define an exception handler in PL/SQL? | EXCEPTION | CATCH | TRY | HANDLE | A |
| 315 | Which of the following actions can be performed by a "BEFORE INSERT" trigger in a DBMS? | Rollback the entire transaction | **Modify data before it is inserted** | Create a new table | Terminate the DBMS session | B |
| 316 | Which of the following are benefits of Triggers? | Generating some derived column values automatically | Enforcing referential integrity | Event logging and storing information on table access | All of the above | D |
| 317 | Which of the following are the advantages of PL/SQL Packages? | Modularity | Easier Application Design | Information Hiding | All of the above | D |
| 318 | Which of the following clause does not comes in the syntax while raising an exception? | DECLARE | WHEN | CLOSE | END | C |
| 319 | Which of the following database languages is used to define the structure and organization of a database? | A) Data Manipulation Language (DML) | B) Data Definition Language (DDL) | C) Data Control Language (DCL) | D) Data Query Language (DQL) | B |
| 320 | Which of the following describes the ACID properties of transactions? | Atomicity, Consistency, Isolation, Durability | Atomicity, Consistency, Isolation, Dileang | Atomicity, Consistency, Isolate, Durability | Atomic, Consistency, Isolation, Durability | A |
| 321 | Which of the following is a benefit of using procedures in a DBMS? | A) Increased code duplication | B) Reduced security | C) Improved code organization and maintenance | D) Limited code reuse | C |
| 322 | Which of the following is a benefit of using the "SAVEPOINT" statement in a DBMS? | It allows you to commit a transaction. | It allows you to roll back the entire database. | It enables you to create nested transactions. | **It provides a way to roll back to a specific point within a transaction.** | D |
| 323 | Which of the following is a benefit of using the "SAVEPOINT" statement in a DBMS? | It allows you to commit a transaction. | It allows you to roll back the entire database. | It enables you to create nested transactions. | **It provides a way to roll back to a specific point within a transaction.** | D |
| 324 | Which of the following is a characteristic of an "AUTOCOMMIT" transaction mode in a DBMS? | Each SQL statement is treated as a separate transaction. | **Transactions are automatically committed after each SQL statement.** | Transactions cannot be rolled back. | It is a read-only mode. | B |
| 325 | Which of the following is a commonly used technique for concurrency control in a DBMS? | A) Optimistic concurrency control | B) Serializability control | C) Slower query performance | D) Increased code duplication | B |
| 326 | Which of the following is a drawback of strict two-phase locking (S2PL) | Increased concurrency | Increased deadlock probability | Reduced consistency | Reduced durability | B |
| 327 | Which of the following is a property of a transaction in a database system? | A) Increased code duplication | B) Slower query performance | C) Atomicity | D) Reduced code redundancy | C |
| 328 | Which of the following is an advantage of using packages in a DBMS? | A) Limited code organization | B) Increased code redundancy | C) Enhanced code isolation | D) Improved code modularity and reusability | D |
| 329 | Which of the following is an advantage of using procedures in a DBMS? | A) Increased code redundancy | B) Slower execution of queries | C) Improved security vulnerabilities | D) Code reusability and maintainability | D |
| 330 | Which of the following is an advantage of using stored procedures in a DBMS? | Reduced security | Increased data redundancy | **Improved performance** | Decreased data integrity | C |
| 331 | Which of the following is an advantage of using triggers in a DBMS? | A) Limited code organization | B) Increased code redundancy | C) Enhanced code isolation | D) Improved code modularity and reusability | D |
| 332 | Which of the following is an example of a parameterized trigger in a DBMS? | A trigger that fires after any insert operation | A trigger that fires after a specific date and time | A trigger that fires when a specific condition is met | **A trigger that accepts parameters passed from the calling application** | D |
| 333 | Which of the following is an example of an optimistic concurrency control technique in a DBMS? | Two-Phase Locking (2PL) | **Multi-Version Concurrency Control (MVCC)** | Time-stamp Ordering | Rollback Segments | B |
| 334 | Which of the following is not a level of data abstraction in a database system? | A) Physical level | B) Logical level | C) External level | D) Semantic level | D |
| 335 | Which of the following is NOT a property of a transaction in DBMS? | Atomicity | Consistency | Durability | Isolation | B |
| 336 | Which of the following is not a type of trigger in DBMS? | Insert trigger | Update trigger | Delete trigger | Search trigger | D |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 337 | Which of the following is not an advantage of trigger? | Various column values are automatically generated by triggers | Maintains the integrity of referential | Tables are replicated asynchronously | Validating transactions and preventing them from being invalid | C |
| 338 | Which of the following is NOT an Oracle-supported trigger? | BEFORE | DURING | AFTER | INSTEAD OF | B |
| 339 | Which of the following is the correct format for if statement? | If boolean expression then statement or compound statement elseif boolean expression  then statement or compound statement else statement or compound statement end if | If boolean expression  then statement or compound statement  elsif boolean expression  then statement or compound statement  else statement or compound statement  end if | If boolean expression then statement or compound statement elif boolean expression then statement or compound statement else statement or compound statement end if | If boolean expression  then statement or compound statement else  statement or compound statement else statement or compound statement end if | A |
| 340 | Which of the following is true about compound triggers? | They can only be defined for tables, not views | They are fired once for each row affected by the triggering event | They cannot contain any SQL statements | They are not supported in DBMS | B |
| 341 | Which of the following is true about recursive triggers? | They are triggered by other triggers | They can only be fired once per event | They can cause an infinite loop if not handled properly | They are not supported in DBMS | C |
| 342 | Which of the following is true about stored procedures? | They can only return a single scalar value. | They can contain control-of-flow statements like IF and LOOP. | They cannot accept input parameters. | They are always automatically executed when the database starts. | B |
| 343 | Which of the following is TRUE about User-defined exceptions? | Users can explicitly raise an exception by using a RAISE statement | RAISE\_APPLICATIO  N\_ERROR can be used to raise a user- defined exception explicitly | both 1 and 2 | None of the above | C |
| 344 | Which of the following is used to input the entry and give the result in a variable in a procedure? | Put and get | Get and put | Out and In | In and out | D |
| 345 | Which of the following makes the transaction permanent in the database? | View | Commit | Rollback | Flashback | B |
| 346 | Which of the following specifies when the trigger will be executed? | BEFORE | AFTER | INSTEAD OF | All of the above | D |
| 347 | Which of the following statements best defines database recovery in DBMS? | The process of restoring data from backup tapes | The process of ensuring that the database remains secure | The process of restoring the database to a consistent state after a failure | The process of recovering deleted data from the Recycle Bin | C |
| 348 | Which of the following statements is true about First Normal Form (1NF)? | A) It allows for multivalued dependencies. | B) It allows for partial dependencies. | C) It eliminates repeating groups and ensures atomicity of data. | D) It enforces referential integrity constraints. | C |
| 349 | Which of the following statements is true about stored procedures? | Stored procedures cannot have input parameters | Stored procedures cannot return values | Stored procedures can be reused and shared by multiple applications | Stored procedures can only be executed by the database administrator | C |
| 350 | Which of the following statements is true about the Two-tier architecture? | A) It allows for better scalability than the Three- tier architecture. | B) It is easier to maintain and modify compared to the Three-tier architecture. | C) It requires less network traffic than the Three-tier architecture. | D) It provides better security and data isolation compared to the Three-tier architecture. | C |
| 351 | Which of the following statements is true regarding stored procedures? | Stored procedures always return a single value. | Stored procedures are not allowed to contain conditional statements. | **Stored procedures are precompiled and stored in the database for reuse.** | Stored procedures can only be executed by database administrators. | C |
| 352 | Which package lets PL/SQL programs read and write operating system (OS) text files? | UTL\_HTTP | UTL\_FILE | UTL\_SMTP | UTL\_FMT | B |
| 353 | Which part of a procedure in a DBMS is responsible for specifying the operations to be performed? | A) Procedure header | B) Procedure specification | C) Procedure body | D) Procedure parameters | C |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 354 | Which property of a transaction ensures that it does not interfere with other transactions while executing? | A) Atomicity | B) Consistency | C) Isolation | D) Durability | C |
| 355 | Which property of a transaction ensures that it does not interfere with other transactions while executing? | A) Atomicity | B) Consistency | C) Isolation | D) Durability | C |
| 356 | Which property of a transaction ensures that it does not violate integrity constraints? | Isolation | Atomicity | Consistency | Durability | C |
| 357 | Which property of a transaction ensures that it either completes in its entirety or has no effect at all? | A) Atomicity | B) Optimistic concurrency control | C) Slower query performance | D) Data redundancy | A |
| 358 | Which property of a transaction ensures that the database remains in a consistent state after transaction execution? | A) Atomicity | B) Consistency | C) Isolation | D) Durability | B |
| 359 | Which recovery technique uses backward recovery to undo the changes made by a  failed transaction? | Undo logging | Redo logging | Deferred update | Immediate update | A |
| 360 | Which specifies the column name that will be updated? | For col\_name | ON col\_name | OF col\_name | WHEN col\_name | C |
| 361 | Which type of database constraint ensures that a foreign key value matches a primary key value in another table? | A) Unique constraint | B) Primary key constraint | C) Foreign key constraint | D) Not null constraint | C |
| 362 | Which type of database trigger in SQL is executed before the triggering event occurs? | AFTER trigger | INSTEAD OF trigger | BEFORE trigger | FOR EACH ROW trigger | C |
| 363 | Which type of error occurs when the database crashes while a transaction is being executed? | System error | Media error | Transaction error | Operator error | A |
| 364 | Which type of trigger in a DBMS can be used to prevent changes to a table? | BEFORE trigger | AFTER trigger | **INSTEAD OF trigger** | FOR EACH ROW trigger | C |
| 365 | Which type of trigger in a DBMS is fired after a triggering event and can be used for  auditing purposes? | BEFORE trigger | **AFTER trigger** | INSTEAD OF trigger | FOR EACH ROW trigger | B |
| 366 | Which type of view in PL/SQL allows you to update data directly through the view? | Materialized View | Read-Only View | Updatable View | Join View | C |
| 367 | Why is "concurrency control" important in a multi-user database environment? | A) To increase query performance | B) To ensure data consistency | C) To eliminate transactions | D) To optimize database storage | B |
| 368 | Why is "concurrency" a concern in a multi-user DBMS environment? | A) To simplify data retrieval | B) To ensure data consistency | C) To reduce query performance | D) To create redundant data | B |
| 369 | Why is concurrency control needed in a database management system (DBMS)? | A) To increase data redundancy | B) To slow down query execution | C) To ensure data consistency | D) To reduce code duplication | C |
| 370 | Write a PL/SQL function named `get\_student\_average\_grade` that takes a student ID as input and returns the average grade of the specified student across all subjects. Which of the following code snippets correctly defines this function? | CREATE OR REPLACE FUNCTION  get\_student\_average\_gra de(student\_id NUMBER) RETURN NUMBER IS BEGIN  -- Function logic here END; | CREATE OR REPLACE PROCEDURE  get\_student\_average  \_grade(student\_id NUMBER) IS BEGIN  -- Procedure logic here  END; | CREATE OR REPLACE FUNCTION  get\_student\_average\_gr ade(student\_id NUMBER, avg\_grade OUT NUMBER) IS  BEGIN  -- Function logic here END; | CREATE OR REPLACE FUNCTION  get\_student\_average\_grade(student\_id NUMBER) RETURN TABLE IS  BEGIN  -- Function logic here END; | A |
| 371 | Write a PL/SQL function named `get\_student\_count\_by\_subject` that takes a subject as input and returns the count of students enrolled in that subject. Which of the following code snippets correctly defines this function? | CREATE OR REPLACE FUNCTION  get\_student\_count\_by\_su bject(subject VARCHAR2) RETURN NUMBER IS BEGIN  -- Function logic here END; | CREATE OR REPLACE FUNCTION  get\_student\_count\_b y\_subject(subject VARCHAR2) RETURN NUMBER IS  student\_count NUMBER; BEGIN  -- Function logic here  END; | CREATE OR REPLACE FUNCTION  get\_student\_count\_by\_ subject(subject VARCHAR2,  student\_count OUT NUMBER) IS  BEGIN  -- Function logic here END; | CREATE OR REPLACE PROCEDURE  get\_student\_count\_by\_subject(subject VARCHAR2) IS BEGIN  -- Procedure logic here END; | a |
| 372 | Write a PL/SQL function named `get\_student\_grade\_in\_subject` that takes a student ID and a subject as input and returns the grade of the specified student for the given subject. Which of the following code snippets correctly defines this function? | CREATE OR REPLACE FUNCTION  get\_student\_grade\_in\_su bject(student\_id NUMBER, subject VARCHAR2) RETURN NUMBER IS  BEGIN  -- Function logic here END; | CREATE OR REPLACE PROCEDURE  get\_student\_grade\_i n\_subject(student\_id NUMBER, subject VARCHAR2) IS  BEGIN  -- Procedure logic here  END; | CREATE OR REPLACE FUNCTION  get\_student\_grade\_in\_s ubject(student\_id NUMBER, subject VARCHAR2, grade OUT NUMBER) IS  BEGIN  -- Function logic here END; | CREATE OR REPLACE FUNCTION  get\_student\_grade\_in\_subject(student\_id NUMBER, subject VARCHAR2) RETURN TABLE IS  BEGIN  -- Function logic here END; | A |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 373 | Write a PL/SQL function named `get\_student\_grade\_in\_subject` that takes a student ID and a subject as input and returns the grade of the specified student for the given subject. Which of the following code snippets correctly defines this function? | CREATE OR REPLACE FUNCTION  get\_student\_grade\_in\_su bject(student\_id NUMBER, subject VARCHAR2) RETURN NUMBER IS  BEGIN  -- Function logic here END; | CREATE OR REPLACE PROCEDURE  get\_student\_grade\_i n\_subject(student\_id NUMBER, subject VARCHAR2) IS  BEGIN  -- Procedure logic here  END; | CREATE OR REPLACE FUNCTION  get\_student\_grade\_in\_s ubject(student\_id NUMBER, subject VARCHAR2, grade OUT NUMBER) IS  BEGIN  -- Function logic here END; | CREATE OR REPLACE FUNCTION  get\_student\_grade\_in\_subject(student\_id NUMBER, subject VARCHAR2) RETURN TABLE IS  BEGIN  -- Function logic here END; | A |
| 374 | Write a PL/SQL function named `get\_student\_grade\_in\_subject` that takes a student ID and a subject as input and returns the grade of the specified student for the given subject. Which of the following code snippets correctly defines this function? | CREATE OR REPLACE FUNCTION  get\_student\_grade\_in\_su bject(student\_id NUMBER, subject VARCHAR2) RETURN NUMBER IS  BEGIN  -- Function logic here END; | CREATE OR REPLACE PROCEDURE  get\_student\_grade\_i n\_subject(student\_id NUMBER, subject VARCHAR2) IS  BEGIN  -- Procedure logic here  END; | CREATE OR REPLACE FUNCTION  get\_student\_grade\_in\_s ubject(student\_id NUMBER, subject VARCHAR2, grade OUT NUMBER) IS  BEGIN  -- Function logic here END; | CREATE OR REPLACE FUNCTION  get\_student\_grade\_in\_subject(student\_id NUMBER, subject VARCHAR2) RETURN TABLE IS  BEGIN  -- Function logic here END; | A |
| 375 | Write a PL/SQL function named `get\_student\_grade\_in\_subject` that takes a student ID and a subject as input and returns the grade of the specified student for the given subject. Which of the following code snippets correctly defines this function? | CREATE OR REPLACE FUNCTION  get\_student\_grade\_in\_su bject(student\_id NUMBER, subject VARCHAR2) RETURN NUMBER IS  BEGIN  -- Function logic here END; | CREATE OR REPLACE PROCEDURE  get\_student\_grade\_i n\_subject(student\_id NUMBER, subject VARCHAR2) IS  BEGIN  -- Procedure logic here  END; | CREATE OR REPLACE FUNCTION  get\_student\_grade\_in\_s ubject(student\_id NUMBER, subject VARCHAR2, grade OUT NUMBER) IS  BEGIN  -- Function logic here END; | CREATE OR REPLACE FUNCTION  get\_student\_grade\_in\_subject(student\_id NUMBER, subject VARCHAR2) RETURN TABLE IS  BEGIN  -- Function logic here END; | A |
| 376 | Write a PL/SQL function named `get\_student\_grade\_in\_subject` that takes a student ID and a subject as input and returns the grade of the specified student for the given subject. Which of the following code snippets correctly defines this function? | CREATE OR REPLACE FUNCTION  get\_student\_grade\_in\_su bject(student\_id NUMBER, subject VARCHAR2) RETURN NUMBER IS  BEGIN  -- Function logic here END; | CREATE OR REPLACE PROCEDURE  get\_student\_grade\_i n\_subject(student\_id NUMBER, subject VARCHAR2) IS  BEGIN  -- Procedure logic here  END; | CREATE OR REPLACE FUNCTION  get\_student\_grade\_in\_s ubject(student\_id NUMBER, subject VARCHAR2, grade OUT NUMBER) IS  BEGIN  -- Function logic here END; | CREATE OR REPLACE FUNCTION  get\_student\_grade\_in\_subject(student\_id NUMBER, subject VARCHAR2) RETURN TABLE IS  BEGIN  -- Function logic here END; | A |
| 377 | Write a PL/SQL function named `get\_student\_grade` that takes a student ID and a subject as input and returns the grade of the specified student for the given subject. Which of the following code snippets correctly defines this function? | CREATE OR REPLACE FUNCTION  get\_student\_grade(stude nt\_id NUMBER, subject VARCHAR2) RETURN NUMBER IS  BEGIN  -- Function logic here END; | CREATE OR REPLACE PROCEDURE  get\_student\_grade(st udent\_id NUMBER, subject VARCHAR2) IS BEGIN  -- Procedure logic here  END; | CREATE OR REPLACE FUNCTION  get\_student\_grade(stud ent\_id NUMBER, subject VARCHAR2, grade OUT NUMBER) IS  BEGIN  -- Function logic here END; | CREATE OR REPLACE FUNCTION  get\_student\_grade(student\_id NUMBER, subject VARCHAR2) RETURN TABLE IS  BEGIN  -- Function logic here END; | A |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 378 | Write a PL/SQL function named `get\_student\_info` that takes a student ID as input and returns the student's name, age, and the number of subjects they are enrolled in. Which of the following code snippets correctly defines this function? | CREATE OR REPLACE FUNCTION  get\_student\_info(student  \_id NUMBER) RETURN VARCHAR2 IS  BEGIN  -- Function logic here END; | CREATE OR REPLACE PROCEDURE  get\_student\_info(stu dent\_id NUMBER) IS BEGIN  -- Procedure logic here  END; | CREATE OR REPLACE FUNCTION  get\_student\_info( student\_id NUMBER, student\_name OUT VARCHAR2,  student\_age OUT NUMBER,  subject\_count OUT NUMBER  ) IS BEGIN  -- Function logic here END; | CREATE OR REPLACE FUNCTION  get\_student\_info(student\_id NUMBER) RETURN TABLE IS BEGIN  -- Function logic here END; | C |
| 379 | Write a PL/SQL function named `get\_student\_info` that takes a student ID as input and returns the student's name, age, and the number of subjects they are enrolled in. Which of the following code snippets correctly defines this function? | CREATE OR REPLACE FUNCTION  get\_student\_info(student  \_id NUMBER) RETURN VARCHAR2 IS  BEGIN  -- Function logic here END; | CREATE OR REPLACE PROCEDURE  get\_student\_info(stu dent\_id NUMBER) IS BEGIN  -- Procedure logic here  END; | CREATE OR REPLACE FUNCTION  get\_student\_info( student\_id NUMBER, student\_name OUT VARCHAR2,  student\_age OUT NUMBER,  subject\_count OUT NUMBER  ) IS BEGIN  -- Function logic here END; | CREATE OR REPLACE FUNCTION  get\_student\_info(student\_id NUMBER) RETURN TABLE IS BEGIN  -- Function logic here END; | C |
| 380 | Write a PL/SQL function named `get\_student\_info` that takes a student ID as input and returns the student's name, age, and the number of subjects they are enrolled in. Which of the following code snippets correctly defines this function? | CREATE OR REPLACE FUNCTION  get\_student\_info(student  \_id NUMBER) RETURN VARCHAR2 IS  BEGIN  -- Function logic here END; | CREATE OR REPLACE PROCEDURE  get\_student\_info(stu dent\_id NUMBER) IS BEGIN  -- Procedure logic here  END; | CREATE OR REPLACE FUNCTION  get\_student\_info( student\_id NUMBER, student\_name OUT VARCHAR2,  student\_age OUT NUMBER,  subject\_count OUT NUMBER  ) IS BEGIN  -- Function logic here END; | CREATE OR REPLACE FUNCTION  get\_student\_info(student\_id NUMBER) RETURN TABLE IS BEGIN  -- Function logic here END; | C |
| 381 | Write a PL/SQL function named `get\_student\_info` that takes a student ID as input and returns the student's name, age, and the number of subjects they are enrolled in. Which of the following code snippets correctly defines this function? | CREATE OR REPLACE FUNCTION  get\_student\_info(student  \_id NUMBER) RETURN VARCHAR2 IS  BEGIN  -- Function logic here END; | CREATE OR REPLACE PROCEDURE  get\_student\_info(stu dent\_id NUMBER) IS BEGIN  -- Procedure logic here  END; | CREATE OR REPLACE FUNCTION  get\_student\_info( student\_id NUMBER, student\_name OUT VARCHAR2,  student\_age OUT NUMBER,  subject\_count OUT NUMBER  ) IS BEGIN  -- Function logic here END; | CREATE OR REPLACE FUNCTION  get\_student\_info(student\_id NUMBER) RETURN TABLE IS BEGIN  -- Function logic here END; | C |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 382 | Write a PL/SQL function named `get\_student\_info` that takes a student ID as input and returns the student's name, age, and the number of subjects they are enrolled in. Which of the following code snippets correctly  defines this function? | CREATE OR REPLACE FUNCTION  get\_student\_info(student  \_id NUMBER) RETURN VARCHAR2 IS  BEGIN  -- Function logic here END; | CREATE OR REPLACE PROCEDURE  get\_student\_info(stu dent\_id NUMBER) IS BEGIN  -- Procedure logic here  END; | CREATE OR REPLACE FUNCTION  get\_student\_info( student\_id NUMBER, student\_name OUT VARCHAR2,  student\_age OUT NUMBER,  subject\_count OUT NUMBER  ) IS BEGIN  -- Function logic here END; | CREATE OR REPLACE FUNCTION  get\_student\_info(student\_id NUMBER) RETURN TABLE IS BEGIN  -- Function logic here END; | C |
| 383 | Write a PL/SQL function named `get\_student\_subject\_scores` that takes a student ID as input and returns a cursor containing the subject and grade for all subjects in which the student is enrolled. Which of the following code snippets correctly defines this function? | CREATE OR REPLACE FUNCTION  get\_student\_subject\_scor es(student\_id NUMBER) RETURN CURSOR IS  BEGIN  -- Function logic here END; | CREATE OR REPLACE PROCEDURE  get\_student\_subject  \_scores(student\_id NUMBER) IS  BEGIN  -- Procedure logic here  END; | CREATE OR REPLACE FUNCTION  get\_student\_subject\_sc ores(  student\_id NUMBER, subject\_scores OUT SYS\_REFCURSOR  ) IS BEGIN  -- Function logic here END; | CREATE OR REPLACE FUNCTION  get\_student\_subject\_scores(student\_id NUMBER) RETURN TABLE IS  BEGIN  -- Function logic here END; | C |
| 384 | Write a PL/SQL function named `get\_student\_subjects` that takes a student ID as input and returns a list of subjects that the student is enrolled in. Which of the following code snippets correctly defines this function? | CREATE OR REPLACE FUNCTION  get\_student\_subjects(stu dent\_id NUMBER) RETURN VARCHAR2 IS BEGIN  -- Function logic here END; | CREATE OR REPLACE FUNCTION  get\_student\_subjects (student\_id NUMBER, subjects OUT VARCHAR2) IS  BEGIN  -- Function logic here  END; | CREATE OR REPLACE PROCEDURE  get\_student\_subjects(st udent\_id NUMBER, subjects OUT VARCHAR2) IS  BEGIN  -- Procedure logic here END; | CREATE OR REPLACE PROCEDURE  get\_student\_subjects(student\_id NUMBER) IS BEGIN  -- Procedure logic here END; | A |
| 385 | Write a PL/SQL function named `get\_student\_subjects` that takes a student ID as input and returns a list of subjects that the student is enrolled in. Which of the following code snippets correctly defines this function? | CREATE OR REPLACE FUNCTION  get\_student\_subjects(stu dent\_id NUMBER) RETURN VARCHAR2 IS BEGIN  -- Function logic here END; | CREATE OR REPLACE FUNCTION  get\_student\_subjects (student\_id NUMBER, subjects OUT VARCHAR2) IS  BEGIN  -- Function logic here  END; | CREATE OR REPLACE PROCEDURE  get\_student\_subjects(st udent\_id NUMBER, subjects OUT VARCHAR2) IS  BEGIN  -- Procedure logic here END; | CREATE OR REPLACE PROCEDURE  get\_student\_subjects(student\_id NUMBER) IS BEGIN  -- Procedure logic here END; | A |
| 386 | Write a PL/SQL function named `get\_student\_subjects` that takes a student ID as input and returns a list of subjects that the student is enrolled in. Which of the following code snippets correctly defines this function? | CREATE OR REPLACE FUNCTION  get\_student\_subjects(stu dent\_id NUMBER) RETURN VARCHAR2 IS BEGIN  -- Function logic here END; | CREATE OR REPLACE FUNCTION  get\_student\_subjects (student\_id NUMBER, subjects OUT VARCHAR2) IS  BEGIN  -- Function logic here  END; | CREATE OR REPLACE PROCEDURE  get\_student\_subjects(st udent\_id NUMBER, subjects OUT VARCHAR2) IS  BEGIN  -- Procedure logic here END; | CREATE OR REPLACE PROCEDURE  get\_student\_subjects(student\_id NUMBER) IS BEGIN  -- Procedure logic here END; | A |
| 387 | Write a PL/SQL function named `get\_student\_subjects` that takes a student ID as input and returns a list of subjects that the student is enrolled in. Which of the following code snippets correctly defines this function? | CREATE OR REPLACE FUNCTION  get\_student\_subjects(stu dent\_id NUMBER) RETURN VARCHAR2 IS BEGIN  -- Function logic here END; | CREATE OR REPLACE FUNCTION  get\_student\_subjects (student\_id NUMBER, subjects OUT VARCHAR2) IS  BEGIN  -- Function logic here  END; | CREATE OR REPLACE PROCEDURE  get\_student\_subjects(st udent\_id NUMBER, subjects OUT VARCHAR2) IS  BEGIN  -- Procedure logic here END; | CREATE OR REPLACE PROCEDURE  get\_student\_subjects(student\_id NUMBER) IS BEGIN  -- Procedure logic here END; | A |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 388 | Write a PL/SQL function named `get\_student\_subjects` that takes a student ID as input and returns a list of subjects that the student is enrolled in. Which of the following code snippets correctly defines this function? | CREATE OR REPLACE FUNCTION  get\_student\_subjects(stu dent\_id NUMBER) RETURN VARCHAR2 IS BEGIN  -- Function logic here END; | CREATE OR REPLACE FUNCTION  get\_student\_subjects (student\_id NUMBER, subjects OUT VARCHAR2) IS  BEGIN  -- Function logic here  END; | CREATE OR REPLACE PROCEDURE  get\_student\_subjects(st udent\_id NUMBER, subjects OUT VARCHAR2) IS  BEGIN  -- Procedure logic here END; | CREATE OR REPLACE PROCEDURE  get\_student\_subjects(student\_id NUMBER) IS BEGIN  -- Procedure logic here END; | A |
| 389 | Write a PL/SQL function named `get\_subjects\_by\_student` that takes a student ID as input and returns a list of subjects that the student is enrolled in. Which of the following code snippets correctly defines this function? | CREATE OR REPLACE FUNCTION  get\_subjects\_by\_student( student\_id NUMBER) RETURN VARCHAR2 IS BEGIN  -- Function logic here END; | CREATE OR REPLACE PROCEDURE  get\_subjects\_by\_stu dent(student\_id NUMBER) IS  BEGIN  -- Procedure logic here  END; | CREATE OR REPLACE FUNCTION  get\_subjects\_by\_studen t(student\_id NUMBER, subjects OUT VARCHAR2) IS  BEGIN  -- Function logic here END; | CREATE OR REPLACE FUNCTION  get\_subjects\_by\_student(student\_id NUMBER) RETURN TABLE IS  BEGIN  -- Function logic here END; | A |
| 390 | Write a PL/SQL procedure named `add\_student\_subject` that takes a student ID, subject, and grade as input and adds a new subject enrollment record for the specified student in the "student" table. Which of the following code snippets correctly defines this procedure? | CREATE OR REPLACE FUNCTION  add\_student\_subject( student\_id NUMBER, subject VARCHAR2, grade NUMBER  ) RETURN NUMBER IS BEGIN  -- Function logic here END; | CREATE OR REPLACE PROCEDURE  add\_student\_subject (  student\_id NUMBER, subject VARCHAR2, grade NUMBER  ) IS BEGIN  -- Procedure logic here  END; | CREATE OR REPLACE PROCEDURE  add\_student\_subject( student\_id NUMBER, subject VARCHAR2, grade NUMBER  ) AS BEGIN  -- Procedure logic here END; | CREATE OR REPLACE PROCEDURE add\_student\_subject( student\_id NUMBER,  subject VARCHAR2, grade NUMBER  ) IS  -- Declare variables here BEGIN  -- Procedure logic here END; | B |
| 391 | Write a PL/SQL procedure named `add\_student\_subject` that takes a student ID, subject, and grade as input and adds a new subject enrollment record for the specified student in the "student" table. Which of the following code snippets correctly defines this procedure? | CREATE OR REPLACE FUNCTION  add\_student\_subject( student\_id NUMBER, subject VARCHAR2, grade NUMBER  ) RETURN NUMBER IS BEGIN  -- Function logic here END; | CREATE OR REPLACE PROCEDURE  add\_student\_subject (  student\_id NUMBER, subject VARCHAR2, grade NUMBER  ) IS BEGIN  -- Procedure logic here  END; | CREATE OR REPLACE PROCEDURE  add\_student\_subject( student\_id NUMBER, subject VARCHAR2, grade NUMBER  ) AS BEGIN  -- Procedure logic here END; | CREATE OR REPLACE PROCEDURE add\_student\_subject( student\_id NUMBER,  subject VARCHAR2, grade NUMBER  ) IS  -- Declare variables here BEGIN  -- Procedure logic here END; | B |
| 392 | Write a PL/SQL procedure named `calculate\_avg\_grade` that takes a student ID as input and calculates the average grade of that student across all subjects. Which of the following code snippets correctly defines this procedure? | CREATE OR REPLACE PROCEDURE  calculate\_avg\_grade(stud ent\_id NUMBER) IS BEGIN  -- Procedure logic here END; | CREATE OR REPLACE PROCEDURE  calculate\_avg\_grade( student\_id NUMBER, avg\_grade OUT NUMBER) IS  BEGIN  -- Procedure logic here  END; | CREATE OR REPLACE PROCEDURE  calculate\_avg\_grade(stu dent\_id NUMBER) AS avg\_grade NUMBER; BEGIN  -- Procedure logic here END; | CREATE OR REPLACE FUNCTION  calculate\_avg\_grade(student\_id NUMBER) RETURN NUMBER IS  BEGIN  -- Function logic here END; | B |
| 393 | Write a PL/SQL procedure named `calculate\_student\_average` that calculates the average grade for a specific student identified by their `student\_id` and stores it in a variable. Which of the following code snippets correctly defines this procedure? | CREATE OR REPLACE PROCEDURE  calculate\_student\_averag e(student\_id NUMBER) AS BEGIN  -- Procedure logic here END; | CREATE OR REPLACE PROCEDURE  calculate\_student\_av erage(student\_id NUMBER, avg\_grade OUT NUMBER) IS BEGIN  -- Procedure logic here  END; | CREATE OR REPLACE FUNCTION  calculate\_student\_avera ge(student\_id NUMBER) RETURN NUMBER IS BEGIN  -- Function logic here END; | CREATE OR REPLACE FUNCTION  calculate\_student\_average(student\_id NUMBER, avg\_grade OUT NUMBER) RETURN NUMBER IS BEGIN  -- Function logic here END; | B |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 394 | Write a PL/SQL procedure named `delete\_student\_record` that takes a student ID as input and deletes the corresponding student record from the "student" table. Which of the following code snippets correctly defines this procedure? | CREATE OR REPLACE FUNCTION  delete\_student\_record(st udent\_id NUMBER) RETURN NUMBER IS BEGIN  -- Function logic here END; | CREATE OR REPLACE PROCEDURE  delete\_student\_recor d(student\_id NUMBER) IS  BEGIN  -- Procedure logic here  END; | CREATE OR REPLACE PROCEDURE  delete\_student\_record( student\_id NUMBER) AS BEGIN  -- Procedure logic here END; | CREATE OR REPLACE PROCEDURE  delete\_student\_record(student\_id NUMBER) IS  -- Declare variables here BEGIN  -- Procedure logic here END; | B |
| 395 | Write a PL/SQL procedure named `delete\_student\_record` that takes a student ID as input and deletes the corresponding student record from the "student" table. Which of the following code snippets correctly defines this procedure? | CREATE OR REPLACE FUNCTION  delete\_student\_record(st udent\_id NUMBER) RETURN NUMBER IS BEGIN  -- Function logic here END; | CREATE OR REPLACE PROCEDURE  delete\_student\_recor d(student\_id NUMBER) IS  BEGIN  -- Procedure logic here  END; | CREATE OR REPLACE PROCEDURE  delete\_student\_record( student\_id NUMBER) AS BEGIN  -- Procedure logic here END; | CREATE OR REPLACE PROCEDURE  delete\_student\_record(student\_id NUMBER) IS  -- Declare variables here BEGIN  -- Procedure logic here END; | B |
| 396 | Write a PL/SQL procedure named `delete\_student\_record` that takes a student ID as input and deletes the corresponding student record from the "student" table. Which of the following code snippets correctly defines this procedure? | CREATE OR REPLACE FUNCTION  delete\_student\_record(st udent\_id NUMBER) RETURN NUMBER IS BEGIN  -- Function logic here END; | CREATE OR REPLACE PROCEDURE  delete\_student\_recor d(student\_id NUMBER) IS  BEGIN  -- Procedure logic here  END; | CREATE OR REPLACE PROCEDURE  delete\_student\_record( student\_id NUMBER) AS BEGIN  -- Procedure logic here END; | CREATE OR REPLACE PROCEDURE  delete\_student\_record(student\_id NUMBER) IS  -- Declare variables here BEGIN  -- Procedure logic here END; | B |
| 397 | Write a PL/SQL procedure named `delete\_student\_record` that takes a student ID as input and deletes the corresponding student record from the "student" table. Which of the following code snippets correctly defines this procedure? | CREATE OR REPLACE FUNCTION  delete\_student\_record(st udent\_id NUMBER) RETURN NUMBER IS BEGIN  -- Function logic here END; | CREATE OR REPLACE PROCEDURE  delete\_student\_recor d(student\_id NUMBER) IS  BEGIN  -- Procedure logic here  END; | CREATE OR REPLACE PROCEDURE  delete\_student\_record( student\_id NUMBER) AS BEGIN  -- Procedure logic here END; | CREATE OR REPLACE PROCEDURE  delete\_student\_record(student\_id NUMBER) IS  -- Declare variables here BEGIN  -- Procedure logic here END; | B |
| 398 | Write a PL/SQL procedure named `delete\_student\_record` that takes a student ID as input and deletes the corresponding student record from the "student" table. Which of the following code snippets correctly defines this procedure? | CREATE OR REPLACE FUNCTION  delete\_student\_record(st udent\_id NUMBER) RETURN NUMBER IS BEGIN  -- Function logic here END; | CREATE OR REPLACE PROCEDURE  delete\_student\_recor d(student\_id NUMBER) IS  BEGIN  -- Procedure logic here  END; | CREATE OR REPLACE PROCEDURE  delete\_student\_record( student\_id NUMBER) AS BEGIN  -- Procedure logic here END; | CREATE OR REPLACE PROCEDURE  delete\_student\_record(student\_id NUMBER) IS  -- Declare variables here BEGIN  -- Procedure logic here END; | B |
| 399 | Write a PL/SQL procedure named `delete\_student` that takes a student ID as input and deletes the corresponding student record from the "student" table. Which of the following code snippets correctly defines this procedure? | CREATE OR REPLACE FUNCTION  delete\_student(student\_i d NUMBER) RETURN NUMBER IS  BEGIN  -- Function logic here END; | CREATE OR REPLACE PROCEDURE  delete\_student(stude nt\_id NUMBER) IS BEGIN  -- Procedure logic here  END; | CREATE OR REPLACE PROCEDURE  delete\_student(student  \_id NUMBER) AS BEGIN  -- Procedure logic here END; | CREATE OR REPLACE PROCEDURE  delete\_student(student\_id NUMBER) IS  -- Declare variables here BEGIN  -- Procedure logic here END; | B |
| 400 | Write a PL/SQL procedure named `enroll\_student\_in\_subject` that takes a student ID, subject, and grade as input and inserts a new enrollment record for the specified student in the "student" table. Which of the following code snippets correctly defines this procedure? | CREATE OR REPLACE FUNCTION  enroll\_student\_in\_subject (  student\_id NUMBER, subject VARCHAR2, grade NUMBER  ) RETURN NUMBER IS BEGIN  -- Function logic here END; | CREATE OR REPLACE PROCEDURE  enroll\_student\_in\_su bject(  student\_id NUMBER, subject VARCHAR2, grade NUMBER  ) IS BEGIN  -- Procedure logic here  END; | CREATE OR REPLACE PROCEDURE  enroll\_student\_in\_subje ct(  student\_id NUMBER, subject VARCHAR2, grade NUMBER  ) AS BEGIN  -- Procedure logic here END; | CREATE OR REPLACE PROCEDURE  enroll\_student\_in\_subject( student\_id NUMBER, subject VARCHAR2,  grade NUMBER  ) IS  -- Declare variables here BEGIN  -- Procedure logic here END; | B |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 401 | Write a PL/SQL procedure named `enroll\_student` that takes student details (name, age, subject, grade) as input and inserts a new record into the "student" table. Which of the following code snippets correctly defines this procedure? | CREATE OR REPLACE FUNCTION  enroll\_student( student\_name VARCHAR2,  student\_age NUMBER, subject VARCHAR2, student\_grade NUMBER  ) RETURN NUMBER IS BEGIN  -- Function logic here END; | CREATE OR REPLACE PROCEDURE  enroll\_student( student\_name VARCHAR2,  student\_age NUMBER,  subject VARCHAR2, student\_grade NUMBER  ) IS BEGIN  -- Procedure logic here  END; | CREATE OR REPLACE PROCEDURE  enroll\_student( student\_name VARCHAR2,  student\_age NUMBER, subject VARCHAR2, student\_grade NUMBER  ) AS BEGIN  -- Procedure logic here END; | CREATE OR REPLACE PROCEDURE enroll\_student( student\_name VARCHAR2,  student\_age NUMBER, subject VARCHAR2, student\_grade NUMBER  ) IS  -- Declare variables here BEGIN  -- Procedure logic here END; | b |
| 402 | Write a PL/SQL procedure named `insert\_student\_record` that takes student details (name, age, subject, grade) as input and inserts a new record into the "student" table. Additionally, it should return the newly generated student ID. Which of the following code snippets correctly defines this procedure? | CREATE OR REPLACE FUNCTION  insert\_student\_record( student\_name VARCHAR2,  student\_age NUMBER, subject VARCHAR2, student\_grade NUMBER  ) RETURN NUMBER IS BEGIN  -- Function logic here END; | CREATE OR REPLACE PROCEDURE  insert\_student\_recor d(  student\_name VARCHAR2,  student\_age NUMBER,  subject VARCHAR2, student\_grade NUMBER,  student\_id OUT NUMBER  ) IS BEGIN  -- Procedure logic here  END; | CREATE OR REPLACE PROCEDURE  insert\_student\_record( student\_name VARCHAR2,  student\_age NUMBER, subject VARCHAR2, student\_grade NUMBER,  student\_id OUT NUMBER  ) AS BEGIN  -- Procedure logic here END; | CREATE OR REPLACE FUNCTION insert\_student\_record( student\_name VARCHAR2,  student\_age NUMBER, subject VARCHAR2, student\_grade NUMBER  ) RETURN NUMBER IS  student\_id NUMBER; BEGIN  -- Function logic here END; | B |
| 403 | Write a PL/SQL procedure named `insert\_student\_record` that takes student details (name, age, subject, grade) as input and inserts a new record into the "student" table. Additionally, it should return the newly generated student ID. Which of the following code snippets correctly defines this procedure? | CREATE OR REPLACE FUNCTION  insert\_student\_record( student\_name VARCHAR2,  student\_age NUMBER, subject VARCHAR2, student\_grade NUMBER  ) RETURN NUMBER IS BEGIN  -- Function logic here END; | CREATE OR REPLACE PROCEDURE  insert\_student\_recor d(  student\_name VARCHAR2,  student\_age NUMBER,  subject VARCHAR2, student\_grade NUMBER,  student\_id OUT NUMBER  ) IS BEGIN  -- Procedure logic here  END; | CREATE OR REPLACE PROCEDURE  insert\_student\_record( student\_name VARCHAR2,  student\_age NUMBER, subject VARCHAR2, student\_grade NUMBER,  student\_id OUT NUMBER  ) AS BEGIN  -- Procedure logic here END; | CREATE OR REPLACE FUNCTION insert\_student\_record( student\_name VARCHAR2,  student\_age NUMBER, subject VARCHAR2, student\_grade NUMBER  ) RETURN NUMBER IS  student\_id NUMBER; BEGIN  -- Function logic here END; | B |
| 404 | Write a PL/SQL procedure named `insert\_student` that takes student details (name, age, subject, grade) as input and inserts a new record into the "student" table. Additionally, it should return the newly generated student ID. Which of the following code snippets correctly defines this procedure? | CREATE OR REPLACE FUNCTION  insert\_student( student\_name VARCHAR2,  student\_age NUMBER, subject VARCHAR2, student\_grade NUMBER  ) RETURN NUMBER IS BEGIN  -- Function logic here END; | CREATE OR REPLACE PROCEDURE  insert\_student( student\_name VARCHAR2,  student\_age NUMBER,  subject VARCHAR2, student\_grade NUMBER,  student\_id OUT NUMBER  ) IS BEGIN  -- Procedure logic here  END; | CREATE OR REPLACE PROCEDURE  insert\_student( student\_name VARCHAR2,  student\_age NUMBER, subject VARCHAR2, student\_grade NUMBER,  student\_id OUT NUMBER  ) AS BEGIN  -- Procedure logic here END; | CREATE OR REPLACE FUNCTION insert\_student( student\_name VARCHAR2,  student\_age NUMBER, subject VARCHAR2, student\_grade NUMBER  ) RETURN NUMBER IS  student\_id NUMBER; BEGIN  -- Function logic here END; | B |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 405 | Write a PL/SQL procedure named `update\_student\_grade` that takes a student ID, subject, and a new grade as input and updates the grade of the specified student for the given subject. Which of the following code snippets correctly defines this procedure? | CREATE OR REPLACE FUNCTION  update\_student\_grade( student\_id NUMBER, subject VARCHAR2, new\_grade NUMBER  ) RETURN NUMBER IS BEGIN  -- Function logic here END; | CREATE OR REPLACE PROCEDURE  update\_student\_grad e(  student\_id NUMBER, subject VARCHAR2, new\_grade NUMBER  ) IS BEGIN  -- Procedure logic here  END; | CREATE OR REPLACE PROCEDURE  update\_student\_grade( student\_id NUMBER, subject VARCHAR2, new\_grade NUMBER  ) AS BEGIN  -- Procedure logic here END; | CREATE OR REPLACE PROCEDURE update\_student\_grade( student\_id NUMBER,  subject VARCHAR2, new\_grade NUMBER  ) IS  -- Declare variables here BEGIN  -- Procedure logic here END; | B |
| 406 | Write a PL/SQL procedure named `update\_student\_record` that takes a student ID as input and updates the student's name, age, and grade in the "student" table.  Which of the following code snippets correctly defines this procedure? | CREATE OR REPLACE FUNCTION  update\_student\_record(st udent\_id NUMBER) RETURN NUMBER IS BEGIN  -- Function logic here END; | CREATE OR REPLACE PROCEDURE  update\_student\_reco rd(student\_id NUMBER) IS  BEGIN  -- Procedure logic here  END; | CREATE OR REPLACE PROCEDURE  update\_student\_record( student\_id NUMBER) AS BEGIN  -- Procedure logic here END; | CREATE OR REPLACE PROCEDURE  update\_student\_record(student\_id NUMBER) IS  -- Declare variables here BEGIN  -- Procedure logic here END; | B |
| 407 | Write a PL/SQL procedure named `update\_student\_record` that takes a student ID as input and updates the student's name, age, and grade in the "student" table.  Which of the following code snippets correctly defines this procedure? | CREATE OR REPLACE FUNCTION  update\_student\_record(st udent\_id NUMBER) RETURN NUMBER IS BEGIN  -- Function logic here END; | CREATE OR REPLACE PROCEDURE  update\_student\_reco rd(student\_id NUMBER) IS  BEGIN  -- Procedure logic here  END; | CREATE OR REPLACE PROCEDURE  update\_student\_record( student\_id NUMBER) AS BEGIN  -- Procedure logic here END; | CREATE OR REPLACE PROCEDURE  update\_student\_record(student\_id NUMBER) IS  -- Declare variables here BEGIN  -- Procedure logic here END; | B |
| 408 | Write a PL/SQL procedure named `update\_student\_record` that takes a student ID as input and updates the student's name, age, and grade in the "student" table.  Which of the following code snippets correctly defines this procedure? | CREATE OR REPLACE FUNCTION  update\_student\_record(st udent\_id NUMBER) RETURN NUMBER IS BEGIN  -- Function logic here END; | CREATE OR REPLACE PROCEDURE  update\_student\_reco rd(student\_id NUMBER) IS  BEGIN  -- Procedure logic here  END; | CREATE OR REPLACE PROCEDURE  update\_student\_record( student\_id NUMBER) AS BEGIN  -- Procedure logic here END; | CREATE OR REPLACE PROCEDURE  update\_student\_record(student\_id NUMBER) IS  -- Declare variables here BEGIN  -- Procedure logic here END; | B |
| 409 | Write a PL/SQL procedure named `update\_student\_record` that takes a student ID as input and updates the student's name, age, and grade in the "student" table.  Which of the following code snippets correctly defines this procedure? | CREATE OR REPLACE FUNCTION  update\_student\_record(st udent\_id NUMBER) RETURN NUMBER IS BEGIN  -- Function logic here END; | CREATE OR REPLACE PROCEDURE  update\_student\_reco rd(student\_id NUMBER) IS  BEGIN  -- Procedure logic here  END; | CREATE OR REPLACE PROCEDURE  update\_student\_record( student\_id NUMBER) AS BEGIN  -- Procedure logic here END; | CREATE OR REPLACE PROCEDURE  update\_student\_record(student\_id NUMBER) IS  -- Declare variables here BEGIN  -- Procedure logic here END; | B |
| 410 | Write a PL/SQL procedure named `update\_student\_subject\_grade` that takes a student ID, subject, and a new grade as input and updates the grade of the specified student for the given subject. Which of the following code snippets correctly defines this procedure? | CREATE OR REPLACE FUNCTION  update\_student\_subject\_ grade(  student\_id NUMBER, subject VARCHAR2, new\_grade NUMBER  ) RETURN NUMBER IS BEGIN  -- Function logic here END; | CREATE OR REPLACE PROCEDURE  update\_student\_subj ect\_grade( student\_id NUMBER, subject VARCHAR2, new\_grade NUMBER  ) IS BEGIN  -- Procedure logic here  END; | CREATE OR REPLACE PROCEDURE  update\_student\_subject  \_grade(  student\_id NUMBER, subject VARCHAR2, new\_grade NUMBER  ) AS BEGIN  -- Procedure logic here END; | CREATE OR REPLACE PROCEDURE  update\_student\_subject\_grade( student\_id NUMBER,  subject VARCHAR2, new\_grade NUMBER  ) IS  -- Declare variables here BEGIN  -- Procedure logic here END; | B |
| 411 | You have a stored procedure named UpdateEmployeeSalary that accepts an employee ID and a salary value as parameters and updates the employee's salary in the database. Which SQL statement would you use to execute this stored procedure with employee ID 101 and a new salary of 55000? | **EXEC**  **UpdateEmployeeSalary 101, 55000;** | CALL  UpdateEmployeeSala ry(101, 55000); | RUN  UpdateEmployeeSalary( 101, 55000); | UPDATE EmployeeSalary(101, 55000); | A |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 412 | You have a stored procedure that calculates the average salary of employees in a specific department. Which SQL statement do you use to execute this stored procedure and retrieve the result? | **EXEC**  **GetAverageSalaryForDep artment 101;** | CALL  GetAverageSalaryFor Department(101); | EXEC  GetAverageSalaryForDe partment @DepartmentID = 101; | EXEC GetAverageSalaryForDepartment @DepartmentID = 101, @Result = OUTPUT; | A |
| 413 | want to create a trigger that updates the "LastPurchaseDate" column to the current date whenever a new purchase is made by a customer. What type of  trigger should you create? | **AFTER INSERT** | BEFORE INSERT | AFTER UPDATE | INSTEAD OF INSERT | A |
| 414 | You have a table named "Inventory" with a "Quantity" column. You want to create  a trigger that automatically updates the "Quantity" column to zero when a product | **AFTER UPDATE** | BEFORE UPDATE | AFTER INSERT | INSTEAD OF UPDATE | A |
| 415 | You want to create a stored procedure that inserts a new customer record into the "Customers" table. The customer's name, email, and phone number will be passed as parameters. Which SQL statement creates this stored procedure? | **CREATE PROCEDURE**  **InsertCustomer @Name VARCHAR(50), @Email VARCHAR(100), @Phone VARCHAR(20) AS**  **BEGIN**  **INSERT INTO Customers (Name, Email, Phone) VALUES (@Name, @Email, @Phone)**  **END** | CREATE PROCEDURE  InsertCustomer AS BEGIN INSERT INTO  Customers (Name, Email, Phone) VALUES (@Name, @Email, @Phone) END | CREATE PROCEDURE  InsertCustomer @CustomerData VARCHAR(MAX) AS BEGIN INSERT INTO  Customers (Name, Email, Phone) VALUES (@CustomerData) END | CREATE PROCEDURE InsertCustomer @Name VARCHAR(50), @Email VARCHAR(100), @Phone VARCHAR(20) AS BEGIN INSERT INTO Customers (Name,  Email, Phone) VALUES (@Name, @Email, @Phone) END | A |