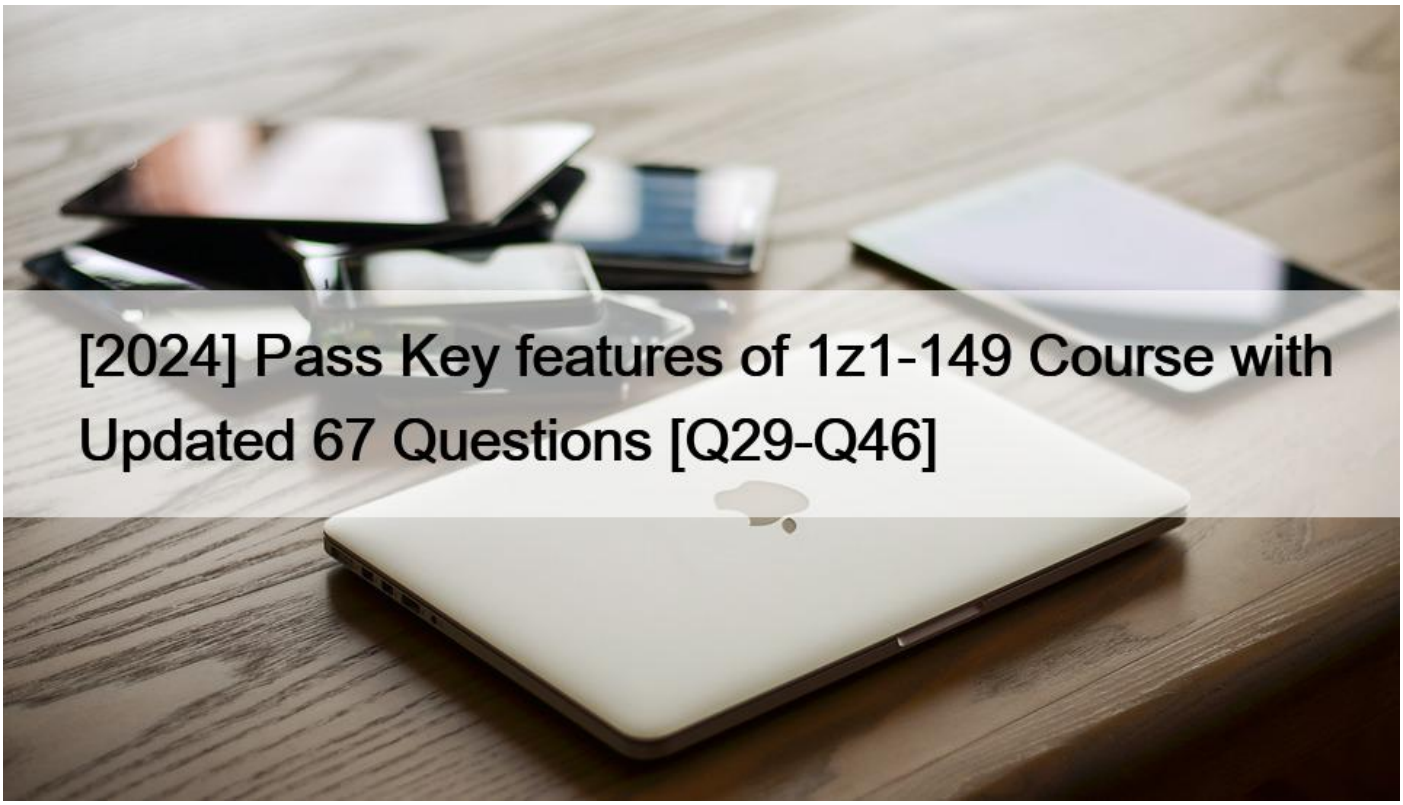
[2024 Pass Key features of 1z1-149 Course with Updated 67 Questions [Q29-Q46



[2024] Pass Key features of 1z1-149 Course with Updated 67 Questions
1z1-149 Sample Practice Exam Questions 2024 Updated Verified

Candidates who pass the Oracle 1Z0-149 exam will have demonstrated their understanding of PL/SQL programming concepts and their ability to write effective PL/SQL code. They will be able to create and manage program units, use collections to manipulate data, and implement error handling and transactions. Oracle Database 19c: Program with PL/SQL certification is recognized by industry professionals and is a valuable addition to any programmer's resume.

NEW QUESTION 29

Examine these statements and output:

```
CONNECT ora1/ora1@pdb1

CREATE OR REPLACE PROCEDURE proc1( v1 OUT NUMBER) is
BEGIN
DBMS_OUTPUT.PUT_LINE(v1*10);
END;
/
```


The procedure is created successfully.

```
GRANT EXECUTE ON proc1 TO ora2;
 grant succeeded.
```

User ora2 has password ora2 in pdb1.

Which script will execute successfully?

*

```
CONNECT ora2/ora2@pdb1
SET SERVEROUTPUT ON
/
BEGIN
  EXEC ora1.proc1;
END;
```


*

```
CONNECT ora2/ora2@pdb1
SET SERVEROUTPUT ON
DECLARE
x NUMBER:=5;
BEGIN
  ora1.proc1(x);
END;
```

*

```
CONNECT ora2/ora2@pdb1
SET SERVEROUTPUT ON
DECLARE
x NUMBER:=5;
BEGIN
  x:=proc1;
DBMS_OUTPUT.PUT_LINE(x)
END;
```


*

```
CONNECT ora2/ora2@pdb1
SET SERVEROUTPUT ON
/
BEGIN
  EXEC proc1;
END;
```


NEW QUESTION 30

Which three are true about the procedure overloading feature? (Choose three.)
* Each procedure can be a nested subprogram.
* Each procedure’s formal parameters can differ in data type or name.
* Each procedure must use named notation to specify the corresponding actual parameters.
* Each procedure’s formal parameters must differ in name.
* Each procedure can be a packaged subprogram.
* Each procedure must use positional notation to specify the corresponding actual parameters.
* Each procedure can be a standalone subprogram.

NEW QUESTION 31

Examine these statements from a block of code:

```
CURSOR c1 IS
  SELECT * FROM products
    FOR UPDATE OF price;

UPDATE products
  SET price = price * 1.05
    WHERE CURRENT OF c1;
```


Which two are true? (Choose two.)

* The lock acquired when executing the OPEN c1 command will be released only after a COMMIT or ROLLBACK statement is issued.
* Only the PRICE column can be updated in the PRODUCTS table.
* The FOR UPDATE OF clause can be used only if the WHERE CURRENT OF clause is used in the executable part of the block.
* The WHERE CURRENT OF clause can be used only if the FOR UPDATE clause is used in the cursor definition.
* An OPEN c1 command will acquire a lock only on the PRICE column in the PRODUCTS table.

NEW QUESTION 32

Examine these statements which execute successfully:

```
CREATE TABLE t (a INT, b INT, c INT INVISIBLE);
 INSERT INTO t (a, b, c) VALUES (1, 2, 3);
 COMMIT;
```

Which anonymous block executes successfully?

*

```
DECLARE
 t_rec t%ROWTYPE;
BEGIN
 t_rec.c := t_rec.a;
 SELECT * INTO t_rec FROM t WHERE ROWNUM < 2;
 DBMS_OUTPUT.PUT_LINE('c = ' || t_rec.c);
END;
```


*

```
DECLARE
 t_rec t%ROWTYPE;
BEGIN
 t_rec.a := t_rec.b;
 SELECT * INTO t_rec FROM t WHERE ROWNUM < 2;
 DBMS_OUTPUT.PUT_LINE('a = ' || t_rec.a);
END;
```


*

```
DECLARE
 t_rec t%ROWTYPE;
BEGIN
 t_rec.b := t_rec.c;
 SELECT * INTO t_rec FROM t WHERE ROWNUM < 2;
 DBMS_OUTPUT.PUT_LINE('b = ' || t_rec.b);
END;
```


*

```
DECLARE
 t_rec t%ROWTYPE;
BEGIN
 t_rec.c := NULL;
 SELECT * INTO t_rec FROM t WHERE ROWNUM < 2;
 DBMS_OUTPUT.PUT_LINE('c = ' || t_rec.c);
END;
```


NEW QUESTION 33

Which three are true about the NOCOPY hint, the PARALLEL ENABLE hint, and the DETERMINISTIC clause? (Choose three.)

* The PARALLEL_ENABLE clause can be used only in the CREATE FUNCTION statement.
* The NOCOPY hint asks the compiler to pass the actual parameters by reference.

* A deterministic function’s results always depend on the state of session variables.
* The NOCOPY hint asks the compiler to pass the actual parameters by value.
* A function is deterministic if it always returns the same result for a specific combination of input values.
* The PARALLEL_ENABLE clause can be specified for a nested function.
* A function defined with the PARALLEL_ENABLE clause may be executed in parallel in a SELECT statement or a subquery in a DML statement.

NEW QUESTION 34

Which two are true about exception handling? (Choose two.)
* Internally defined exceptions can be handled only by the OTHERS exception handler.
* All declared exceptions are raised implicitly by the runtime system.
* User-defined exceptions can be defined in the declarative part of any PL/SQL anonymous block, subprogram, or package.
* Only predefined exceptions and user-defined exceptions can have a user-declared name associated with them.
* Predefined exceptions are globally declared in the standard package.

NEW QUESTION 35

Which three are true about functions and procedures? (Choose three.)
* The ACCESSIBLE BY clause can be used only for procedures.
* In a function, every execution path must lead to a RETURN statement.
* Both can have only constants as actual parameters for IN mode parameters.
* Both can be invoked from within SQL statements.
* In a procedure the RETURN statement cannot specify an expression.
* In a function every RETURN statement must specify an expression.

NEW QUESTION 36

Examine this code:

ALTER SESSION SET plsql_warnings=’ENABLE:ALL’;

/

You compile this function:

```
CREATE OR REPLACE FUNCTION check_values(a NUMBER, b NUMBER)
RETURN NUMBER
AUTHID CURRENT_USER
IS
BEGIN
IF a>b then RETURN 1;
ELSIF a<b then return 2;
ELSE RETURN 3;
END IF;
RETURN 100;
END check values;
```


What happens when the function is created with PLSQL_WARNINGS set to ‘ENABLE: ALL’?
* It fails compilation.
* There are no compilation warnings or errors.

* A severe compilation warning is generated.
* A performance compilation warning is generated.
* An information compilation warning is generated.

NEW QUESTION 37

User ORA41 executes these statements successfully:

Now, examine this statement which is executed successfully by user ORA61 after a successful login:

EXECUTE ora41.update_emp_proc(100,25000);

Which two are true? (Choose two.)
* The salary will be changed for employee 100 in the EMPLOYEES table owned by ORA41.
* No update happens even though the procedure executes successfully.
* The salary will be changed for employee 100 in the EMPLOYEES table owned by ORA61.
* The UPDATE privilege on ORA41.EMPLOYEES is not inherited by ORA61 through the procedure.
* ORA61 will have been granted the UPDATE privilege explicitly on ORA41.EMPLOYEES before executing the statement.

NEW QUESTION 38

Which three are true regarding code based access control (CBAC)? (Choose three.)
* In a multitenant environment, the DELEGATE option of CBAC cannot be used.
* CBAC roles can be granted to a program unit only if they are directly granted to its owner.
* CBAC roles can be granted to a program unit only if they are the predefined roles automatically defined by the standard scripts as part of database creation.
* You can use CBAC to attach database roles to a PL/SQL function or procedure only.
* In CBAC, the ADMIN and DELEGATE options cannot both be granted to the same user.
* You can use CBAC to attach database roles to a PL/SQL function, procedure, or package.
* CBAC cannot be used to secure definer’s rights.

NEW QUESTION 39

Examine these facts:

Table EMP exists in schema USERA with columns SALARY and EMP_ID.

EMP_ID is the primary key with values ranging from 1 to 100.

USERA now executes these statements successfully:

```
conn userA/userA@pdb1
/
create or replace procedure myproc
is
eRec emp%rowtype;
begin
select * into eRec from userA.emp where emp_id=50;
dbms_output.put_line(eRec.Salary);
end;
/
```


USERA then grants execute privilege on procedure MYPROC to USERB.

USERB exists in the database identified by pdb1 but does not have select privilege on USERA.EMP.

USERB now executes these statements:

conn userB/userB@pdb1

execute userA.myproc;

Which is true?
* It results in an error because Authid Current_User is missing from MYPROC.
* It results in an error because Authid Definer is missing from MYPROC.
* It results in an error because USERB doesn’t have select privilege on USERA.EMP.
* It executes successfully.

NEW QUESTION 40

Which two are valid MODIFIER values for the PLSQL_WARNINGS parameter? (Choose two.)
* DISABLE
* ENABLE
* ERROR
* ALL
* SEVERE

NEW QUESTION 41

Examine this anonymous block of code:

```
DECLARE
   v_raise   number(5);
BEGIN
   UPDATE employees
   SET salary = salary + v_raise;
END;
```


Which two statements are true about the results of executing it? (Choose two.)
* It will set all salaries to 0 if it executes successfully.
* It will always return a compile time error because it lacks an EXCEPTION section.
* It might return a run time error depending on who invokes it.
* It will always automatically initialize v_raise.
* It will set all salaries to NULL if it executes successfully.
* It will always return a run time error because v_raise is not initialized.

NEW QUESTION 42

Examine this DECLARE section:

```
1 DECLARE
2  v_join_date DATE := SYSDATE - 10;
3  v_flag BOOLEAN NOT NULL DEFAULT TRUE;
4  v_char VARCHAR2 := NULL;
5  v_bonus_pct CONSTANT REAL(2) := 8.25;
6  v_zip_code VARCHAR2(80) := SUBSTR('Oracle Corporation', 24, 0);
```


Which line will cause an error upon execution?
* line 5
* line 3
* line 2
* line 4
* line 6

NEW QUESTION 43

Which two blocks of code display a numerical zero? (Choose two.)
*

```
CREATE OR REPLACE PROCEDURE calc_price IS
 price NUMBER := 0;
BEGIN
  DECLARE
    price NUMBER;
  BEGIN
    price := calc_price.price;
    DBMS_OUTPUT.PUT_LINE(price);
  END;
END;
/
BEGIN
  calc_price;
END;
/
```

*
```
<<outer>>
DECLARE
  price NUMBER := 0;
  PROCEDURE calc_price AS
    BEGIN
      DBMS_OUTPUT.PUT_LINE(price);
  END;
BEGIN
  calc_price;
END;
/
```

*
```
<<outer>>
<<inner>>
DECLARE
  price NUMBER := 0;
BEGIN
<<inner>>
  DECLARE
    price NUMBER := NULL;
  BEGIN
    price := inner.price;
    DBMS_OUTPUT.PUT_LINE(price);
  END;
END;
/
```

*
```
<<outer>>
DECLARE
  price NUMBER;
BEGIN
<<inner>
  DECLARE
    price NUMBER;
  BEGIN
    price := 0;
  END;
  DBMS_OUTPUT.PUT_LINE(price);
END;
/
```

NEW QUESTION 44

Which block of code displays the error message “Incorrect price value”?

*
```
DECLARE
  price CONSTANT NUMBER(4) := 10000;
BEGIN
  NULL;
EXCEPTION
  WHEN VALUE_ERROR THEN
    DBMS_OUTPUT.PUT_LINE('Incorrect price value');
END;
/
```

*
```
BEGIN
DECLARE
  price CONSTANT NUMBER(4) := 50000;
BEGIN
  NULL;
END;
EXCEPTION
  WHEN VALUE_ERROR THEN
    DBMS_OUTPUT.PUT_LINE('Incorrect price value');
END;
/
```

*

```
BEGIN
DECLARE
  error_detected EXCEPTION;
  PRAGMA EXCEPTION_INIT(error_detected, -20001);
  price CONSTANT NUMBER(4) := 10000;
BEGIN
  NULL;
END;
EXCEPTION
  WHEN error_detected THEN
    DBMS_OUTPUT.PUT_LINE('Incorrect price value');
END;
/
```

*

```
DECLARE
  price CONSTANT NUMBER(4) := 10000;
BEGIN
  NULL;
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Incorrect price value');
END;
/
```


NEW QUESTION 45

Which is the correct method to implement a local subprogram in an anonymous block?
*

```
DECLARE
fnam VARCHAR2(10) := 'King';
lnam VARCHAR2(12) := 'Cobra';
BEGIN
FUNCTION full_name ( A VARCHAR2, B VARCHAR2) RETURN VARCHAR2 AS
C VARCHAR2(20);
BEGIN
C := A || ';' || B;
RETURN C;
END full_name;
DBMS_OUTPUT.PUT_LINE(full_name (fnam, lnam));
END;
```

*

```
BEGIN
DECLARE
fnam VARCHAR2(10) := 'King';
lnam VARCHAR2(12) := 'Cobra';
FUNCTION full_name ( A  VARCHAR2, B  VARCHAR2) RETURN VARCHAR2 AS
C VARCHAR2(20);
BEGIN
C := A || ';' || B;
RETURN C;
END full_name;
BEGIN
DBMS_OUTPUT.PUT_LINE('And the output is...');
END;
DBMS_OUTPUT.PUT_LINE(full_name (fnam, lnam));
END;
```

*

```
BEGIN
DECLARE
fnam VARCHAR2(10) := 'King';
lnam VARCHAR2(12) := 'Cobra';
BEGIN
FUNCTION full_name ( A  VARCHAR2, B  VARCHAR2) RETURN VARCHAR2 AS
C VARCHAR2(20);
BEGIN
C := A || ' ' || B;
RETURN C;
END full_name;
DBMS_OUTPUT.PUT_LINE('And the output is...');
END;
DBMS_OUTPUT.PUT_LINE(full_name (fnam, lnam));
END;
```

*

```
DECLARE
fnam VARCHAR2(10) := 'King';
lnam VARCHAR2(12) := 'Cobra';
FUNCTION full_name ( A  VARCHAR2, B  VARCHAR2) RETURN VARCHAR2 AS
C VARCHAR2(20);
BEGIN
C := A || ';' || B ;
RETURN C;
END full_name;
BEGIN
DBMS_OUTPUT.PUT_LINE(full_name (fnam, lnam));
END;
```


NEW QUESTION 46

Examine this statement which executes successfully:

SQL> SET SERVEROUTPUT ON;

Now, examine this code which is executed:

```
SQL> DECLARE
  2  v_hiredate DATE := '12-June-2020';
  3  v_location VARCHAR2(13);
  4  v_deptno NUMBER(2) NOT NULL;
  5  v_comm CONSTANT NUMBER := 5;
  6  BEGIN
  7  DBMS_OUTPUT.PUT_LINE ('Hire date '   ||v_hiredate);
  8  DBMS_OUTPUT.PUT_LINE ('Location '    ||v_location);
  9  DBMS_OUTPUT.PUT_LINE ('Department ');
 10  DBMS_OUTPUT.PUT_LINE ('Commission ' ||v_comm);
 11 END;
 12 /
```


What is true about the result?

* It returns an error in line 2.
* It returns an error in line 4.
* It returns an error in line 9.
* It executes and displays output.