


PL/SQL Messages and Codes_

00000-01000: PL/SQL

This help file created via:
Oracle Book to Microsoft Help Version 1.0w
© 1994 Oracle Corporation Inc.
Authored by **KM** .

 **00000-01000: PL/SQL**

This section lists messages that might arise from using PL/SQL. Each message code has the format PLS- *nnnn*, where *nnnn* is an integer. Each message description lists possible causes for the message and suggests corrective action.

The messages listed here **are** duplicated in the [Oracle7 Server Messages and Codes Manual](#).

PLS-00102

PLS-00102 parser stack overflow because nesting is too deep

Cause: The parser, which checks the syntax of PL/SQL statements, uses a data structure called a stack; the number of levels of nesting in your PL/SQL block exceeded the stack capacity.

Action: Reorganize your block structure to avoid nesting at too deep a level. For example, move the lowest-level sub-block to a higher level.

PLS-00103

PLS-00103 found '*str*' but expected one of the following: '*str*'

Cause: This message is from the parser. It found a token (language element) that is inappropriate in this context.

Action: Check previous tokens as well as the one given in the message. The line and column numbers given in the following message refer to the end of the faulty language construct.

PLS-00104

PLS-00104 empty argument list in call of procedure '*name*' must be omitted

Cause: In a subprogram call, the name of the subprogram was followed by an empty parameter list. For example, procedure P was called as P(). This is not allowed.

Action: Remove the empty parameter list. In the example, change the procedure call to P.

PLS-00108

PLS-00108 declarative units must be a single variable declaration

Cause: While checking a declarative unit (a top-level declare block without the BEGIN...END), PL/SQL found that there was more than one item declared or that the item was not a variable declaration. A table is a common variable declaration at the unit level. To define a TABLE you can compile a DECLARE compilation unit, but only one at a time is allowed.

Action: Declare variables in separate declarative units.

PLS-00109

PLS-00109 unknown exception name '*name*' in PRAGMA EXCEPTION_INIT

Cause: No declaration for the exception name was found within its scope.

Action: Make sure the pragma follows the exception declaration and is within the same scope.

PLS-00110

PLS-00110 bind variable '*name*' not allowed in this context

Cause: A bind variable, that is, an identifier prefixed with a colon, was found in an inappropriate context.

Action: Remove the colon or replace the bind variable with the appropriate object.

PLS-00111

PLS-00111 end-of-file in comment

Cause: A comment had a comment initiator (/*), but before the comment terminator (*/) was found, an end-of-file marker was encountered.

Action: Remove the comment initiator or add a comment terminator. The line and column numbers given in the following message refer to the beginning of the last legal token before the comment initiator.

PLS-00112

PLS-00112 end-of-line in quoted identifier

Cause: A quoted identifier had a beginning quote ("), but before the ending quote (") was found, an end-of-line marker was encountered.

Action: Remove the beginning quote or add the ending quote. The line and column numbers given in the message refer to the beginning of the quoted identifier.

PLS-00113

PLS-00113 END identifier '*name1*' must match '*name2*' at line *num*, column *num*

Cause: Following the keyword END, which terminates some language constructs (such as loops, blocks, functions, and procedures), you can optionally place the name of that construct. For example, at the end of the definition of loop L you might write END L.

This message occurs when the optional name does not match the name given to the language construct. It is usually caused by a misspelled identifier or by faulty block structure.

Action: Make sure the spelling of the END identifier matches the name given to the language construct and that the block structure is correct.

PLS-00114

PLS-00114 bind variable '*name*' exceeds implementation length

Cause: The name of a PL/SQL variable is longer than 30 characters. Legal identifiers (including quoted identifiers) have a maximum length of 30 characters. You might have mistakenly enclosed a string literal in double quotes instead of single quotes, in which case PL/SQL considers it a quoted identifier.

Action: Shorten the name to 30 characters or less, or if you are using a string literal, replace the double quotes with single quotes.

PLS-00115

PLS-00115 this PRAGMA must follow the declaration of 'name'

Cause: The pragma refers to a PL/SQL object that was not declared or is not within the scope of the reference. Identifiers must be declared before they are used in a pragma; forward references are *not* allowed.

Action: Check your spelling and declaration of the identifier. Also confirm that the declaration is placed correctly in the block structure.

PLS-00116

PLS-00116 duplicate WHERE clause in table expression

Cause: Two or more WHERE clauses were found in a DELETE, SELECT, or UPDATE statement. The WHERE clause specifies a condition under which rows in a table are processed. The condition can contain several logical expressions connected by AND or OR, but a statement can contain only one WHERE clause.

Action: Remove one of the WHERE clauses and, if necessary, connect logical expressions by AND or OR.

PLS-00117

PLS-00117 duplicate CONNECT BY clause in table expression

Cause: Two or more CONNECT BY clauses were found in a SELECT statement. The CONNECT BY clause defines a relationship used to return rows in a hierarchical order. The relationship can contain two expressions separated by a relational operator (such as = or !=), but a statement can contain only one CONNECT BY clause.

Action: Remove one of the CONNECT BY clauses and, if necessary, separate expressions by a relational operator.

PLS-00118

PLS-00118 duplicate GROUP BY clause in table expression

Cause: Two or more GROUP BY clauses were found in a SELECT statement. The GROUP BY clause lists column expressions used to form a summary row for each group of selected rows. The list can contain several column expressions separated by commas, but a statement can contain only one GROUP BY clause.

Action: Remove one of the GROUP BY clauses and, if necessary, separate column expressions by commas.

PLS-00119

PLS-00119 duplicate HAVING clause in table expression

Cause: Two or more HAVING clauses were found in a SELECT statement. The HAVING clause specifies a condition under which groups of rows (formed by the GROUP BY clause) are included in the result. The condition can include several logical expressions connected by AND or OR, but a statement can contain only one HAVING clause.

Action: Remove one of the HAVING clauses and, if necessary, connect logical expressions by AND or OR.

PLS-00120

PLS-00120 inappropriate argument in OPEN statement

Cause: The *cursor_name* parameter in an OPEN statement is misspelled or does not refer to a legally declared cursor.

Action: Check your spelling of the *cursor_name* parameter. Make sure the cursor was declared properly.

PLS-00123

PLS-00123 program too large

- Cause:** PL/SQL was designed primarily for robust transaction processing. One consequence of the special-purpose design is that the PL/SQL compiler imposes a limit on block size. The limit depends on the mix of statements in your PL/SQL block. Blocks that exceed the limit cause generate this message.
- Action:** The best solution is to modularize your program by defining subprograms, which can be stored in an ORACLE database. Another solution is to break your program into two sub-blocks. Have the first block INSERT any data the second block needs into a temporary database table. Then, have the second block SELECT the data from the table.

PLS-00124

PLS-00124 name of exception expected for first argument in EXCEPTION_INIT pragma

Cause: The first argument passed to the EXCEPTION_INIT pragma was something other than an exception name. The first argument must be the name of a legally declared exception.

Action: Replace the first argument with the name of a legally declared exception.

PLS-00201

PLS-00201 identifier '*name*' must be declared

Cause: You tried to reference an undefined variable, exception, procedure, function, or other object. Either you failed to declare the identifier or it is not within the scope of the reference.

Action: Check your spelling and declaration of the identifier. Also confirm that the declaration is placed correctly in the block structure.

PLS-00202

PLS-00202 **type '*name*' must be declared**

Cause: You tried to reference an undefined type. Either you failed to declare the type identifier or it is not within the scope of the reference.

Action: Check your spelling and declaration of the type identifier. Also confirm that the declaration is placed correctly in the block structure.

PLS-00203

PLS-00203 function DECODE must be called with at least 3 non-Boolean arguments

Cause: Less than three arguments were passed to the built-in function DECODE. Though DECODE takes a variable number of (non-Boolean) arguments, you must pass it at least three.

Action: Call DECODE with three or more arguments.

PLS-00204

PLS-00204 function or pseudo-column '*name*' may be used inside a SQL statement only

Cause: A pseudo-column or proscribed function was used in a procedural statement. The SQL pseudo-columns (CURRVAL, LEVEL, NEXTVAL, ROWID, ROWNUM) can be used only in SQL statements. Likewise, certain functions such as DECODE and the SQL group functions (AVG, MIN, MAX, COUNT, SUM, STDDEV, VARIANCE) can be used only in SQL statements.

Action: Remove the pseudo-column reference or function call from the procedural statement. Or, replace the procedural statement with a SELECT INTO statement; for example, replace

```
bonus := DECODE(rating,1,5000,2,2500,...);
```

with the following statement:

```
SELECT DECODE(rating,1,5000,2,2500,...) INTO bonus FROM dual;
```

PLS-00205

PLS-00205 aggregate not allowed here

Cause: An aggregate, that is, a parenthesized list of values such as (7788, 'SCOTT', 20), was found in an inappropriate context.

Action: Remove or relocate the aggregate.

PLS-00206

PLS-00206 %TYPE must be applied to a variable or column, not '*name*'

Cause: The program object declared using the %TYPE datatype attribute is not of the appropriate class. It must be a variable, column, record component, subprogram formal parameter, or other object to which values can be assigned.

Action: Declare an object of the appropriate class or define the datatype in another way (for example, use %ROWTYPE).

PLS-00207

PLS-00207 identifier '*name*', applied to implicit cursor SQL, is not a legal cursor attribute

Cause: An identifier that is not a cursor attribute was applied to the identifier SQL. For example, this error occurs if the cursor attribute is misspelled.

Action: Check your spelling of the cursor attribute name. Make sure the attribute is one of these: %NOTFOUND, %FOUND, %ROWCOUNT, %ISOPEN.

PLS-00208

PLS-00208 identifier '*name*' is not a legal cursor attribute

Cause: An identifier not declared as a cursor attribute was applied to an identifier declared as a cursor. For example, this error occurs if the cursor attribute is misspelled.

Action: Check your spelling of the cursor attribute name. Make sure the attribute is one of these: %NOTFOUND, %FOUND, %ROWCOUNT, %ISOPEN.

PLS-00209

PLS-00209 table '*name*' is not in FROM clause

Cause: In a query, a table referenced by the select list is not named in the FROM clause.

Action: Check your spelling of the table names, make sure each column in the select list refers to a table in the FROM clause, then retry the query.

PLS-00210

PLS-00210 an OTHERS clause is required in this CASE statement

Cause: Not in Release 2.0.

PLS-00211

PLS-00211 CASE labels or ranges must not be duplicated in different WHEN clauses

Cause: Not in Release 2.0.

PLS-00212

PLS-00212 could not obtain enough memory to compile CASE statement

Cause: Not in Release 2.0.

PLS-00213

PLS-00213 package STANDARD not accessible

Cause: The PL/SQL compiler could not find library unit STANDARD in the current ORACLE database and so could not access package STANDARD, which is stored in library unit STANDARD. To compile a program, PL/SQL needs package STANDARD.

Action: Make sure that library unit STANDARD is available in the current ORACLE database, then retry the operation.

PLS-00214

PLS-00214 BEGIN...END block nesting is too deep

Cause: The number of levels of nesting in your PL/SQL block is too large. You can nest blocks up to 255 levels deep, depending on the availability of system resources such as memory.

Action: Reorganize your block structure to avoid nesting at too deep a level. For example, move the lowest-level sub-block to a higher level.

PLS-00215

PLS-00215 string length constraints must be in range (1 .. 32767)

Cause: When declaring a character variable, you specified a length that is outside the legal range. For example, the following declarations are illegal:

```
flag CHAR(0);          -- illegal; zero length
name VARCHAR2(-10);   -- illegal; negative length
```

Action: Change the length constraint, making sure that it lies in the range 1 .. 32767.

PLS-00216

PLS-00216 NUMBER precision constraint must be in range (1 .. 38)

Cause: You declared a NUMBER variable with a precision that is outside the legal range. Declarations such as N NUMBER(800) or N NUMBER(123,10) are not supported.

Action: Change the illegal NUMBER precision constraint, making sure that it lies in the range 1 .. 38.

PLS-00217

PLS-00217 NUMBER scale constraint must be in range (-84 .. 127)

Cause: You declared a NUMBER variable with a scale that is outside the legal range. Declarations such as N NUMBER(10,345) or N NUMBER(10,-100) are not supported.

Action: Change the illegal NUMBER scale constraint, making sure that it lies in the range -84 .. 127.

PLS-00218

PLS-00218 a variable declared NOT NULL must have an initialization assignment

Cause: In general, variables that have no initialization clause in their declaration are automatically initialized to NULL. This is illogical for NOT NULL variables, and therefore an initialization clause is required.

Action: Add an initialization clause to the variable declaration.

PLS-00219

PLS-00219 label '*name*' reference is out of scope

Cause: A block or loop label was used to qualify a variable (as in *outer_block.date*) that was not declared or is not within the scope of the label. The variable name might be misspelled, its declaration might be faulty, or the declaration might be placed incorrectly in the block structure.

Action: Check your spelling and declaration of the variable name. Also confirm that the declaration is placed correctly in the block structure.

PLS-00220

PLS-00220 simple name required in this context

Cause: A qualified name such as A.B or A.B.C is not permitted here.

Action: Use a simple name such as A instead.

PLS-00221

PLS-00221 *'name'* is not a procedure or is undefined

Cause: The named identifier is being referenced as a procedure, but the identifier was not declared or actually represents another object (for example, it might have been declared as a function).

Action: Check your spelling and declaration of the identifier. Also confirm that the declaration is placed correctly in the block structure.

PLS-00222

PLS-00222 no function with name '*name*' exists in this scope

Cause: The named identifier is being referenced as a function, but the identifier was not declared or actually represents another object (for example, it might have been declared as a procedure).

Action: Check your spelling and declaration of the identifier. Also confirm that the declaration is placed correctly in the block structure.

PLS-00223

PLS-00223 parameterless procedure '*name*' used as function

Cause: The named identifier is being referenced as a parameterless function, but the identifier actually represents a procedure.

Action: Check your spelling and declaration of the identifier. Also confirm that the declaration is placed correctly in the block structure. If necessary, change the declaration of the identifier or change the reference so that it does not require a return value.

PLS-00224

PLS-00224 object '*name*' must be of type function or array to be used this way

Cause: The named identifier is being referenced as a function or an array, but the identifier actually represents an object (a number or date, for example) that cannot be referenced in this way.

Action: Check your spelling and declaration of the identifier. Also confirm that the declaration is placed correctly in the block structure.

PLS-00225

PLS-00225 subprogram or cursor '*name*' reference is out of scope

Cause: A subprogram or cursor references a variable that was not declared or is not within the scope of the subprogram or cursor. The variable name might be misspelled, its declaration might be faulty, or the declaration might be placed incorrectly in the block structure.

Action: Check your spelling and declaration of the variable name. Also confirm that the declaration is placed correctly in the block structure.

PLS-00226

PLS-00226 package '*name*' used as variable reference

Cause: A package was referenced in an expression as if it were a variable or function. Either the name of the variable or function is misspelled or the reference is not fully qualified. For example, to call the function *my_function*, which is stored in package *my_package*, you must use dot notation as follows:

```
... my_package.my_function ...
```

Action: Correct your spelling of the variable or function name or use dot notation to reference the packaged variable or function.

PLS-00227

PLS-00227 subprogram IN formal '*name*' is not yet denotable

Cause: When declaring the formal parameters of a subprogram, you used one parameter to initialize another, as in

```
PROCEDURE my_proc (j NUMBER, k NUMBER := j) IS ...
```

The first parameter has no value until run time, so it cannot be used to initialize another parameter.

Action: Remove the illegal formal parameter reference.

PLS-00229

PLS-00229 attribute expression within SQL expression

Cause: You used an attribute expression such as SQL%NOTFOUND in a SQL statement, but attribute expressions are allowed only in procedural statements.

Action: To work around this limitation, assign the value of the attribute expression to a variable, then use the variable in the SQL statement. For example, replace the statement

```
INSERT INTO audits VALUES (c1%ROWCOUNT, ...);
```

with the following statements:

```
row_count := c1%ROWCOUNT;
```

```
INSERT INTO audits VALUES (row_count, ...);
```

PLS-00230

PLS-00230 OUT and IN OUT formal parameters may not have default expressions

Cause: When declaring the formal parameters of a procedure, you initialized an OUT or IN OUT parameter to a default value, as in

```
PROCEDURE calc_bonus (bonus OUT REAL := 0, ...) IS ...
```

However, only IN parameters can be initialized to default values.

Action: Remove the illegal default expression.

PLS-00231

PLS-00231 function '*name*' may not be used in SQL

Cause: You used a proscribed function in a SQL statement. Certain functions such as SQLCODE and SQLERRM can be used only in procedural statements.

Action: Remove the function call from the SQL statement. Or, replace the function call with a local variable. For example, the following statement is illegal:

```
INSERT INTO errors VALUES (SQLCODE, SQLERRM);
```

However, you can assign the values of SQLCODE and SQLERRM to local variables, then use the variables in the SQL statement, as follows:

```
err_num := SQLCODE;  
err_msg := SQLERRM;  
INSERT INTO errors VALUES (err_num, err_msg);
```

PLS-00232

PLS-00232 nested packages not permitted

Cause: You declared a package inside another package, but package declarations are allowed only at the top level. In other words, you cannot nest packages.

Action: Move the package declaration outside the enclosing package.

PLS-00233

PLS-00233 function name used as an exception name in WHEN clause

Cause: The WHEN clause in an exception handler contains a function call instead of an exception name. A valid exception handler consists of a WHEN clause, which must specify an exception, followed by a sequence of statements to be executed when that exception is raised.

Action: Check your spelling of the identifier in the WHEN clause, then replace the function call with an exception name.

PLS-00302

PLS-00302 component '*name*' must be declared

Cause: In a reference to a component (for example, in the name A.B, B is a component of A), the component was not declared. The component might be misspelled, its declaration might be faulty, or the declaration might be placed incorrectly in the block structure.

Action: Check your spelling and declaration of the component. Also confirm that the declaration is placed correctly in the block structure.

PLS-00303

PLS-00303 qualifier '*name*' must be declared

Cause: In a name such as A.B, A is a qualifier, and B is a component of the qualifier. This error occurs when no declaration for the qualifier is found. The qualifier might be misspelled, its declaration might be faulty, or the declaration might be placed incorrectly in the block structure.

Action: Check your spelling and declaration of the qualifier. Also confirm that the declaration is placed correctly in the block structure.

PLS-00304

PLS-00304 cannot compile body of 'name' without its specification

Cause: The compiled package specification needed to compile a package body could not be found. Some possible causes follow:

- the package name is misspelled
- the package specification was never compiled
- the compiled package specification is not accessible

You must compile the package specification before compiling the package body, and the compiler must have access to the compiled specification.

Action: Check your spelling of the package name. Compile the package specification before compiling the package body. And, make sure the compiler has access to the compiled specification.

PLS-00305

PLS-00305 previous use of '*name*' conflicts with this use

Cause: While looking for prior declarations of a cursor, procedure, function, or package, the compiler found another object with the same name in the same scope.

Action: Check your spelling of the cursor, procedure, function, or package name. Also check the names of all constants, variables, parameters, and exceptions declared in the same scope. Then, remove or rename the object with the duplicate name.

PLS-00306

PLS-00306 wrong number or types of arguments in call to '*name*'

Cause: This error occurs when the named subprogram call cannot be matched to any declaration for that subprogram name. The subprogram name might be misspelled, a parameter might be of the wrong datatype, the declaration might be faulty, or the declaration might be placed incorrectly in the block structure. For example, this error occurs if you call the built-in square root function SQRT with a misspelled name or with a parameter of the wrong datatype.

Action: Check your spelling and declaration of the subprogram name. Also confirm that its call is correct, its parameters are of the right datatype, and, if it is not a built-in function, that its declaration is placed correctly in the block structure.

PLS-00307

PLS-00307 too many declarations of '*name*' match this call

Cause: The declaration of a subprogram name is ambiguous because there was no exact match between the declaration and the call and more than one declaration matched the call when implicit conversions of the parameter datatypes were used. The subprogram name might be misspelled, its declaration might be faulty, or the declaration might be placed incorrectly in the block structure.

Action: Check your spelling and declaration of the subprogram name. Also confirm that its call is correct, its parameters are of the right datatype, and, if it is not a built-in function, that its declaration is placed correctly in the block structure.

PLS-00308

PLS-00308 this construct is not allowed as the origin of an assignment

Cause: The construct or expression does not designate a value that can be assigned to a variable. For example, the datatype name NUMBER cannot appear on the right hand side of an assignment statement as in X := NUMBER.

Action: Correct the illegal assignment statement.

PLS-00309

PLS-00309 with %LAST attribute, '*name*' must be a variable of an enumerated type

Cause: Not in Release 2.0.

PLS-00310

PLS-00310 with %ROWTYPE attribute, 'name' must name a cursor or table

Cause: The %ROWTYPE attribute must be applied to an identifier declared as a cursor or table. This error occurs when %ROWTYPE follows some identifier that has not been so declared.

Action: Change the declaration or do not apply the %ROWTYPE attribute to the identifier.

PLS-00311

PLS-00311 the declaration of the type of 'name' is incomplete or malformed

Cause: This occurrence of the identifier cannot be compiled because its datatype has not been properly defined.

Action: Correct the faulty datatype declaration.

PLS-00312

PLS-00312 a positional parameter association may not follow a named association

Cause: When passing a list of parameters to a subprogram or cursor, if you use both positional and named associations, you must place all positional associations in their declared order and before all named associations, which can be in any order.

Action: Reorder the parameter list to meet the requirements or use named association only.

PLS-00313

PLS-00313 *'name'* not declared in this scope

Cause: There is no declaration for the given identifier within the scope of reference. The identifier might be misspelled, its declaration might be faulty, or the declaration might be placed incorrectly in the block structure.

Action: Check your spelling and declaration of the identifier. Also confirm that the declaration is placed correctly in the block structure.

PLS-00314

PLS-00314 TABLE declarations are not allowed as PL/SQL local variables

Cause: In a precompiled program, you mistakenly used the DECLARE TABLE statement inside an embedded PL/SQL block. If an embedded PL/SQL block refers to a database table that does not yet exist, you can use the DECLARE TABLE statement to tell the precompiler what the table will look like. But, DECLARE TABLE statements are allowed only in the host program.

Action: Move the DECLARE TABLE statement outside the embedded PL/SQL block. If you want a variable that can store an entire row of data selected from a database table or fetched by a cursor, use the %ROWTYPE attribute.

PLS-00315

PLS-00315 PL/SQL TABLE declarations must currently use BINARY_INTEGER indexes

Cause: In the INDEX BY clause of a PL/SQL table declaration, you specified a datatype other than BINARY_INTEGER. PL/SQL tables can have one column and a primary key. The column can belong to any scalar type, but the primary key must belong to type BINARY_INTEGER.

Action: Change the datatype specifier to BINARY_INTEGER.

PLS-00316

PLS-00316 PL/SQL TABLE declarations must currently use a single index

Cause: In the INDEX BY clause of a PL/SQL table declaration, you specified a composite primary key. PL/SQL tables must have a simple, unnamed primary key of type BINARY_INTEGER.

Action: Change the faulty clause to INDEX BY BINARY_INTEGER.

PLS-00319

PLS-00319 subquery in an IN or NOT IN clause must contain exactly one column

Cause: You used an invalid expression such as:

```
a IN (SELECT x, y, z FROM ... )
```

When a [NOT]IN clause is used with a subquery, it does not test for set membership. The number of expressions in the [NOT]IN clause and the subquery select list must match. So, in the example above, the subquery must specify at most one column.

Action: Change the subquery to select only one column.

PLS-00320

PLS-00320 the declaration of the type of this expression is incomplete or malformed

Cause: In a declaration, the name of a variable or cursor is misspelled or the declaration makes a forward reference. Forward references are not allowed in PL/SQL. You must declare a variable or cursor *before* referencing it in other statements, including other declarative statements. For example, the following declaration of DEPT_REC raises this exception because it refers to a cursor not yet declared:

```
DECLARE
```

```
dept_rec dept_cur%ROWTYPE;
```

```
CURSOR dept_cur IS SELECT ...
```

```
...
```

Action: Check your spelling of all identifiers in the declaration. If necessary, move the declaration so that it makes no forward references.

PLS-00321

PLS-00321 expression 'str' is inappropriate as the left hand side of an assignment statement

Cause: The expression does not designate a variable that can have a value assigned to it. For example, the function SYSDATE cannot appear on the left hand side of an assignment statement such as:

```
SYSDATE := '01-JAN-1990';
```

Action: Correct the illegal assignment statement.

PLS-00322

PLS-00322 declaration of a constant '*name*' must contain an initialization assignment

Cause: A constant declaration is lacking the assignment of the constant value. For example, in the following declaration " := 3.14159" is the initialization assignment:

```
pi CONSTANT NUMBER := 3.14159;
```

Action: Correct the constant declaration by supplying the missing initialization assignment.

PLS-00323

PLS-00323 subprogram '*name*' is declared in a package specification and must be defined in the package body

Cause: You placed a subprogram specification in a package specification, but neglected to place the corresponding subprogram body in the package body. The package body implements the package specification. So, the package body must contain the definition of every subprogram declared in the package specification.

Action: Check your spelling of the subprogram name. If necessary, add the missing subprogram body to the package body.

PLS-00324

PLS-00324 cursor attribute may not be applied to non-cursor '*name*'

Cause: This error occurs when a cursor attribute (%FOUND, %NOTFOUND, %ROWCOUNT, or %ISOPEN) appears following an identifier not declared as a cursor. It occurs, for example, if the variable name *my_cur* in *my_cur*%FOUND was not properly declared as a cursor or if the variable declaration was placed incorrectly in the block structure.

Action: Check your spelling and declaration of the identifier. Also confirm that the declaration is placed correctly in the block structure.

PLS-00325

PLS-00325 non-integral numeric literal *num* is inappropriate in this context

Cause: A non-integral numeric literal was used in a context that requires an integer (a number with no fractional part).

Action: Replace the inappropriate literal with an integral literal.

PLS-00326

PLS-00326 IN clause must contain same number of expressions as subquery

Cause: The number of expressions in an IN clause did not equal the number of expressions in a corresponding subquery select list. For example, the following statement is invalid because the IN clause contains two expressions, but the subquery select list contains just one:

```
... WHERE (ename, sal) IN (SELECT sal FROM emp);
```

Action: Check the number of expressions in each set, then revise the statement to make the numbers equal.

PLS-00328

PLS-00328 a subprogram body must be defined for the forward declaration of *'name'*

Cause: You declared a subprogram specification, but failed to define the corresponding subprogram body. You can write the subprogram specification and body as a unit. Or, you can separate the specification from its body, which is necessary when you want to define mutually recursive subprograms or group subprograms in a package.

Action: Check your spelling of the subprogram name. If necessary, supply the missing subprogram body.

PLS-00341

PLS-00341 declaration of cursor '*name*' is incomplete or malformed

Cause: A cursor declaration is improper or an identifier referenced in the cursor declaration was not properly declared. You might have specified a return type (%ROWTYPE) that does not refer to an existing database table or previously declared cursor. For example, the following cursor declaration is illegal because *c1* is not yet fully defined:

```
CURSOR c1 RETURN c1%ROWTYPE IS SELECT ... -- illegal
```

In this case, you need not specify a return type because it is implicit.

Action: Check your spelling and declaration of the cursor name and any identifiers referenced in the cursor declaration. Also confirm that the declaration is placed correctly in the block structure. If you specified a return type, make sure that it refers to an existing database table or previously declared cursor.

PLS-00351

PLS-00351 not logged on to database 'name'

Cause: You tried to access an ORACLE database without being logged on. Probably, an invalid username or password was entered.

Action: Log on to ORACLE with a correctly spelled username and password before trying to access the database.

PLS-00352

PLS-00352 unable to access another database 'name'

Cause: You tried to reference an object in a database other than the current local or remote ORACLE database.

Action: Correct your reference and make sure the object is in the current ORACLE database.

PLS-00353

PLS-00353 *'name'* must name a user in the database

Cause: This error occurs when the username was misspelled or when the user does not exist in the database.

Action: Check your spelling of the username and make sure the user exists.

PLS-00354

PLS-00354 username must be a simple identifier

Cause: A qualified username such as SCOTT.ACCTS is not permitted in this context.

Action: Specify a simple username such as SCOTT instead.

PLS-00356

PLS-00356 *'name'* must name a table to which the user has access

Cause: The named table is not accessible to the user. This error occurs when the table name or username was misspelled, the table and/or user does not exist in the database, or the user was not granted the necessary privileges.

Action: Check your spelling of the table name and username. Also confirm that the table and user exist and that the user has the necessary privileges.

PLS-00357

PLS-00357 table, view or sequence reference '*name*' not allowed in this context

Cause: A reference to database table, view, or sequence was found in an inappropriate context. Such references can appear only in SQL statements or (excluding sequences) in %TYPE and %ROWTYPE declarations. Some valid examples follow:

```
SELECT ename, emp.deptno, dname INTO my_ename, my_deptno, my_dept
```

```
FROM emp, dept WHERE emp.deptno = dept.deptno;
```

```
DECLARE
```

```
last_name emp.ename%TYPE;
```

```
dept_rec dept%ROWTYPE;
```

Action: Remove or relocate the illegal reference.

PLS-00358

PLS-00358 column '*name*' exists in more than one table; use qualifier

Cause: Your statement is ambiguous because it specifies two or more tables having the same column name. For example, the following statement is ambiguous because DEPTNO is a column in both tables:

```
SELECT deptno, loc INTO my_deptno, my_loc FROM emp, dept;
```

Action: Precede the column name with the table name (as in EMP.DEPTNO) so that the column reference is unambiguous.

PLS-00359

PLS-00359 assignment target in 'str' must have components

Cause: An assignment target was not declared to have the required components. For example, this error occurs if you try to assign a row of column values to a variable instead of a record, as follows:

```
DECLARE
```

```
dept_rec dept%ROWTYPE;  
my_deptno dept.deptno%TYPE;  
...
```

```
BEGIN
```

```
SELECT deptno, dname, loc INTO my_deptno -- invalid  
FROM dept WHERE ...  
...
```

Action: Check your spelling of the names of the assignment target and all its components. Make sure the assignment target is declared with the required components and that the declaration is placed correctly in the block structure.

PLS-00360

PLS-00360: cursor declaration without body needs return type

Cause: A cursor declaration lacks either a body (SELECT statement) or a return type (%ROWTYPE). If you want to separate a cursor specification from its body, you must supply a return type, as in

```
CURSOR c1 RETURN emp%ROWTYPE;
```

Action: Add a SELECT statement or return type to the cursor declaration.

PLS-00363

PLS-00363 expression '*str*' cannot be used as an assignment target

Cause: You mistakenly used a literal, constant, IN parameter, loop counter, or function call as the target of an assignment. For example, the following statement is illegal because the assignment target, 30, is a literal:

```
SELECT deptno INTO 30 FROM dept WHERE ... -- illegal
```

Action: Correct the statement by using a valid assignment target.

PLS-00364

PLS-00364 loop index variable '*name*' use is invalid

Cause: A reference to a loop counter was found in an inappropriate context. For example, the following statement is illegal because the loop counter is used as the terminal value in its own range expression:

```
FOR j IN 1 .. j LOOP ... -- illegal
```

Action: Change the loop range expression so that it does not reference the loop counter. If you want to refer in the range expression to another variable with the same name as the loop counter, change either name or qualify the variable name with a label.

PLS-00365

PLS-00365 '*name*' is an OUT parameter and cannot be read

Cause: You tried to assign the value of an OUT parameter to another parameter or variable. Inside a procedure, an OUT parameter acts like an uninitialized variable; therefore, its value cannot be read. For example, the following assignments are illegal:

```
PROCEDURE calc_wages (bonus OUT REAL, ...) IS
    rating REAL;
    wages REAL;
BEGIN
    ...
    IF rating > 90 THEN
        bonus := bonus * 2; -- illegal
    END IF;
    SELECT sal + bonus INTO wages FROM emp ... -- illegal
    ...
END calc_bonus;
```

Action: Use an IN OUT parameter instead of the OUT parameter. Inside a procedure, an IN OUT parameter acts like an initialized variable; therefore, its value can be read.

PLS-00366

PLS-00366 subtype of a NOT NULL type must also be NOT NULL

Cause: Not in Release 2.0.

PLS-00367

PLS-00367 a RAISE statement with no exception name must be inside an exception handler

Cause: A RAISE statement not followed by an exception name was found outside an exception handler.

Action: Delete the RAISE statement, relocate it to an exception handler, or supply the missing exception name.

PLS-00368

PLS-00368 in RAISE statement, '*name*' must be an exception name

Cause: The identifier in a RAISE statement is not a valid exception name.

Action: Make sure the identifier in the RAISE statement was declared as an exception and is correctly placed in the block structure. If you are using one of the PL/SQL predefined exception names, check its spelling.

PLS-00369

PLS-00369 no choices may appear with choice OTHERS in an exception handler

Cause: A construct of the form

```
WHEN excep1 OR OTHERS =>
```

was encountered in the definition of an exception handler. OTHERS must appear by itself as the last exception handler in a block.

Action: Remove the identifier that appears with OTHERS or write a separate exception handler for that identifier.

PLS-00370

PLS-00370 OTHERS handler must be last among the exception handlers of a block

Cause: One or more exception handlers appear after an OTHERS handler. However, the OTHERS handler must be the last handler in a block or subprogram because it acts as the handler for all exceptions not named specifically.

Action: Move the OTHERS handler so that it follows all specific exception handlers.

PLS-00371

PLS-00371 at most one declaration for '*name*' is permitted in the declaration section

Cause: A reference to an identifier is ambiguous because there are conflicting declarations for it in the declaration section of a block, procedure, or function. At most one declaration of the identifier is permitted in a declaration section.

Action: Check your spelling of the identifier. If necessary, remove all but one declaration of the identifier.

PLS-00372

PLS-00372 in a procedure, RETURN statement cannot contain an expression

Cause: In a procedure, a RETURN statement contains an expression, which is not allowed. In functions, a RETURN statement *must* contain an expression because its value is assigned to the function identifier. However, in procedures, a RETURN statement simply lets you exit before the normal end of the procedure is reached.

Action: Remove the expression from the RETURN statement, or redefine the procedure as a function.

PLS-00373

PLS-00373 EXIT label '*name*' must label a LOOP statement

Cause: The statement with the named label is not a loop. An EXIT statement does not require a label operand, but if you specify one (as in EXIT *my_label*), it must be the label of a loop statement.

Action: Make sure the label name is spelled correctly and that it refers to a loop statement.

PLS-00374

PLS-00374 illegal EXIT statement; it must appear inside the loop labeled 'name'

Cause: An EXIT statement does not require a label operand, but if you specify one (as in EXIT *my_label*), the EXIT statement must be inside the loop designated by that label.

Action: Make sure the label name is spelled correctly, placed properly, and refers to the appropriate EXIT statement.

PLS-00375

PLS-00375 illegal GOTO statement; this GOTO cannot branch to label '*name*'

Cause: The line and column numbers accompanying the error message refer to a GOTO that branches from outside a construct (a loop or exception handler, for example) containing a sequence of statements to a label inside that sequence of statements. Such a branch is not allowed.

Action: Either move the GOTO statement inside the sequence of statements or move the labeled statement outside the sequence of statements.

PLS-00376

PLS-00376 illegal EXIT statement; it must appear inside a loop

Cause: An EXIT statement was found outside of a loop construct. The EXIT statement is used to exit prematurely from a loop and so must always appear within a loop.

Action: Either remove the EXIT statement or place it inside a loop.

PLS-00377

PLS-00377 internal type PLS_INTEGER is not included in this release of PL/SQL

Cause: In a declaration, you mistakenly specified the datatype PLS_INTEGER. PLS_INTEGER is a PL/SQL internal datatype used within package STANDARD.

Action: Specify the datatype BINARY_INTEGER instead of PLS_INTEGER.

PLS-00378

PLS-00378 invalid compilation unit for this release of PL/SQL

Cause: A *compilation unit* is a file containing PL/SQL source code that is passed to the compiler. Only compilation units containing blocks, declarations, statements, and subprograms are allowed. This error occurs when some other language construct is passed to the compiler.

Action: Make sure the compilation unit contains only blocks, declarations, statements, and subprograms.

PLS-00379

PLS-00379 CASE statements are not included in this release of PL/SQL

Cause: The unit being compiled contains a CASE statement. However, the current release of PL/SQL does not support CASE statements.

Action: Remove the CASE statement from the compilation unit.

PLS-00381

PLS-00381 type mismatch found at '*name*' between column and variable in subquery or INSERT

Cause: The datatypes of a column and a variable do not match. The variable was encountered in a subquery or INSERT statement.

Action: Change the variable datatype to match that of the column.

PLS-00382

PLS-00382 expression is of wrong type

Cause: The given expression variable is of the wrong datatype for the context in which it was found.

Action: Change the datatype of the expression. You might want to use datatype conversion functions.

PLS-00383

PLS-00383 type mismatch found at '*name*' inside an IN or NOT IN clause

Cause: In a test for set membership such as `X NOT IN (SELECT Y ...)`, the expressions X and Y do not match in datatype, and it is unclear which implicit conversion is needed to correct the mismatch.

Action: Change the expressions so that their datatypes match. You might want to use datatype conversion functions in the select list.

PLS-00384

PLS-00384 type mismatch found at '*name*' in UPDATE's SET clause

Cause: The column to the left of the equal sign in the SET clause of an UPDATE statement does not match in datatype with the column, expression, or subquery to the right of the equal sign, and it is unclear which implicit conversion is needed to correct the mismatch.

Action: Change the expressions so that their datatypes match. You might want to use datatype conversion functions in the SET clause.

PLS-00385

PLS-00385 type mismatch found at '*name*' in SELECT...INTO statement

Cause: The expressions to the left and right of the INTO clause in a SELECT...INTO statement do not match in datatype, and it is unclear which implicit conversion is needed to correct the mismatch.

Action: Change the expressions so that their datatypes match. You might want to use datatype conversion functions in the select list.

PLS-00386

PLS-00386 type mismatch found at '*name*' between FETCH cursor and INTO variables

Cause: An assignment target in the INTO list of a FETCH statement does not match in datatype with the corresponding column in the select list of the cursor declaration, and it is unclear which implicit conversion is needed to correct the mismatch.

Action: Change the cursor declaration or change the datatype of the assignment target. You might want to use datatype conversion functions in the select list of the query associated with the cursor.

PLS-00387

PLS-00387 INTO variable cannot be a database object

Cause: An item in the INTO list of a FETCH or SELECT statement was found to be a database object. INTO introduces a list of user-defined variables to which output values are assigned. Therefore, database objects cannot appear in the INTO list.

Action: Check your spelling of the INTO list item. If necessary, remove the item from the INTO list or replace it with a user-defined output variable.

PLS-00388

PLS-00388 undefined column '*name*' in subquery

Cause: A subquery contains a column name that was not defined for the specified table.

Action: Change the expression to specify a column that was defined.

PLS-00389

PLS-00389 undefined column '*name*' in left-hand-side expression

Cause: A left-hand-side expression in a SQL statement refers to an undefined column.

Action: Check your spelling of the column name, then change the expression so that it refers only to defined columns.

PLS-00390

PLS-00390 undefined column '*name*' in INSERT statement

Cause: An INSERT statement refers to a column not defined for the table or view into which data is being INSERTed.

Action: Check your spelling of the column name, then revise the statement so that it refers only to defined columns.

PLS-00391

PLS-00391 undefined column '*name*' in UPDATE statement

Cause: An UPDATE statement refers to a column not defined for the table or view being UPDATED.

Action: Check your spelling of the column name, then revise the statement so that it refers only to defined columns.

PLS-00392

PLS-00392 type mismatch in arguments to BETWEEN

Cause: In a comparison such as X BETWEEN Y AND Z, the expressions X, Y, and Z do not match in datatype, and it is unclear which implicit conversion is needed to correct the mismatch.

Action: Change the expressions so that their datatypes match. You might want to use datatype conversion functions.

PLS-00393

PLS-00393 wrong number of columns in SELECT...INTO statement

Cause: The number of columns selected by a SELECT...INTO statement does not match the number of variables in the INTO clause.

Action: Change the number of columns in the select list or the number of variables in the INTO clause so that the numbers match.

PLS-00394

PLS-00394 wrong number of values in the INTO list of a FETCH statement

Cause: The number of variables in the INTO clause of a FETCH statement does not match the number of columns in the cursor declaration.

Action: Change the number of variables in the INTO clause or the number of columns in the cursor declaration so that the numbers match.

PLS-00395

PLS-00395 wrong number of values in VALUES clause of INSERT statement

Cause: The number of columns in an INSERT statement does not match the number of values in the VALUES clause. For example, the following statement is faulty because no column is specified for the value 20:

```
INSERT INTO emp (empno, ename) VALUES (7788, 'SCOTT', 20);
```

Action: Change the number of items in the column list or the number of items in the VALUES list so that the numbers match.

PLS-00396

PLS-00396 INSERT statement's subquery yields wrong number of columns

Cause: The number of columns in an INSERT statement does not match the number of columns in a subquery select list. For example, the following statement is faulty because no corresponding column is specified for *col3*:

```
INSERT INTO emp (ename, empno) SELECT col1, col2, col3 FROM ...
```

Action: Change the number of items in the column list of the INSERT statement or the number of items the select list so that the numbers match.

PLS-00397

PLS-00397 type mismatch in arguments to IN

Cause: In a test for set membership such as `X IN (Y, Z)`, the expressions `X`, `Y`, and `Z` do not match in datatype, and it is unclear which implicit conversion is needed to correct the mismatch.

Action: Change the expressions so that their datatypes match. You might want to use datatype conversion functions.

PLS-00398

PLS-00398 wrong number of columns in UNION, INTERSECT, or MINUS expression

Cause: The SELECT clauses to the left and right of a UNION, INTERSECT, or MINUS expression do not select the same number of columns. For example, the following statement is faulty because the select lists do not contain the same number of items:

```
CURSOR my_cur IS SELECT ename FROM emp
```

```
INTERSECT SELECT ename, empno FROM emp;
```

Action: Change the select lists so that they contain the same number of items.

PLS-00399

PLS-00399 different types of columns in UNION, INTERSECT, or MINUS expression

Cause: The select lists to the left and right of a UNION, INTERSECT, or MINUS expression select at least one column that is mismatched in datatype. For example, the following statement is faulty because the constant 3 is of datatype NUMBER, whereas SYSDATE is of datatype DATE:

```
CURSOR my_cur IS SELECT 3 FROM emp
```

```
INTERSECT SELECT SYSDATE FROM emp;
```

Action: Change the select lists so that they match in datatype. You might want to use datatype conversion functions in the select list of one or more queries.

PLS-00400

PLS-00400: different number of columns between cursor SELECT statement and return value

Cause: In a cursor declaration, you specified a return type (such as RETURN emp %ROWTYPE), but the number of returned column values does not match the number of select-list items.

Action: Change the cursor return type or the select list so that the number of returned column values matches the number of select-list items.

PLS-00401

PLS-00401: different column types between cursor SELECT statement and return value found at 'name'

Cause: In a cursor declaration, you specified a return type (such as RETURN emp %ROWTYPE), but a returned column value and its corresponding select-list item belong to different datatypes.

Action: Change the cursor return type or the select list so that each returned column value and its corresponding select-list item belong to the same datatype.

PLS-00402

PLS-00402 alias required in SELECT list of cursor to avoid duplicate column names

Cause: A cursor was declared with a SELECT statement that contains duplicate column names. Such references are ambiguous.

Action: Replace the duplicate column name in the select list with an alias.

PLS-00403

PLS-00403 INTO list of FETCH statement contains illegal assignment target

Cause: A FETCH statement was unable to assign a value to an assignment target in its INTO list because the target is not a legally formed and declared variable. For example, the following assignment is illegal because 'Jones' is a character string, not a variable:

```
FETCH my_cur INTO 'Jones';
```

Action: Check your spelling and declaration of the assignment target. Make sure you followed the rules for forming variable names.

PLS-00404

PLS-00404 cursor '*name*' must be declared with FOR UPDATE to use with CURRENT OF

Cause: The use of the CURRENT OF *name* clause is legal only if *name* was declared with a FOR UPDATE clause.

Action: Add a FOR UPDATE clause to the definition of the cursor or do not use the CURRENT OF '*name*' clause.

PLS-00405

PLS-00405 subquery not allowed in this context

Cause: A subquery was used in an inappropriate context, such as:

```
if (SELECT deptno FROM emp WHERE ... ) = 20 then ...
```

Subqueries are allowed only in SQL statements.

Action: You can get the same result by using a temporary variable, as in:

```
SELECT deptno INTO temp_var FROM emp WHERE ...;
```

```
IF temp_var = 20 THEN ...
```

PLS-00406

PLS-00406 length of SELECT list in subquery must match number of assignment targets

Cause: A query select list is not the same length as the list of targets that will receive the returned values. For example, the following statement is faulty because the subquery returns two values for one target:

```
UPDATE emp SET ename =
```

```
(SELECT ename, empno FROM emp WHERE ename = 'SMITH') ...
```

Action: Change one of the lists so that they contain the same number of items.

PLS-00407

PLS-00407 '*' not allowed here; a list of columns is required

Cause: An asterisk (*) was used as an abbreviation for a list of column names. However, in this context the column names must be written out explicitly.

Action: Replace the asterisk with a list of column names.

PLS-00408

PLS-00408 duplicate column '*name*' not permitted in INSERT or UPDATE

Cause: An UPDATE or INSERT statement has a column list that contains duplicate column names.

Action: Check your spelling of the column names, then eliminate the duplication.

PLS-00409

PLS-00409 duplicate variable '*name*' in INTO list is not permitted

Cause: The same variable appears twice in the INTO list of a SELECT or FETCH statement.

Action: Remove one of the variables from the INTO clause.

PLS-00410

PLS-00410 duplicate fields in record or table are not allowed

Cause: When declaring a user-defined record, you gave the same name to two fields. Like column names in a database table, field names in a user-defined record must be unique.

Action: Check your spelling of the field names, then remove the duplicate.

PLS-00412

PLS-00412 list of values not allowed as argument to this function or procedure

Cause: A parenthesized list of values separated by commas (that is, an aggregate) was used in the wrong context. For example, the following usage is *invalid*:

```
WHERE (col1, col2) > (SELECT col3, col4 FROM my_table ...)
```

However, an equal sign can take a list of values and a subquery as left- and right-hand-side arguments respectively, so the following usage is valid:

```
WHERE (col1, col2) = (SELECT col3, col4 FROM my_table ...)
```

Action: Rewrite the expression. For example, the clause

```
WHERE (col1, col2) > (SELECT col3, col4 FROM my_table ...)
```

can be rewritten as

```
WHERE col1 > (SELECT col3 FROM my_table ...) AND  
col2 > (SELECT col4 FROM my_table ...)
```

PLS-00413

PLS-00413 identifier in CURRENT OF clause is not a cursor name

Cause: The identifier in a CURRENT OF clause names an object other than a cursor.

Action: Check your spelling of the identifier. Make sure that it names the cursor in the DELETE or UPDATE statement and that it names the cursor itself, not a FOR-loop variable.

PLS-00414

PLS-00414 no column '*name*' in table

Cause: A table name or alias was used to qualify a column reference, but the column was not found in that table. Either the column was never defined or the column name is misspelled.

Action: Confirm that the column was defined and check your spelling of the column name.

PLS-00415

PLS-00415 *'name'* is an **OUT** parameter and cannot appear in a function

Cause: When declaring the formal arguments of a function, you specified the OUT or IN OUT parameter mode. Procedures can take IN, OUT, and IN OUT parameters, but functions can take only IN arguments.

Action: Remove the OUT or IN OUT argument from the formal argument list, or redefine the function as a procedure.

PLS-00450

PLS-00450 a variable of this private type cannot be declared here

Cause: Not in Release 2.0.

PLS-00483

PLS-00483 exception '*name*' may appear in at most one exception handler in this block

Cause: The same exception appears in two different exception handlers within the same EXCEPTION section. That is not allowed.

Action: Remove one of the duplicate exception handlers.

PLS-00484

PLS-00484 exceptions '*name*' and '*name*' have same ORACLE error number and must appear in same exception handler

Cause: Using PRAGMA EXCEPTION_INIT, you initialized different exceptions to the same ORACLE error number, then referred to them in different exception handlers within the same EXCEPTION section. Such references conflict.

Action: Remove one of the exceptions or initialize it to a different ORACLE error number.

PLS-00485

PLS-00485 in exception handler, 'name' must be an exception name

Cause: An identifier not declared as an exception appears in an exception handler WHEN clause. Only the name of an exception is valid in a WHEN clause.

Action: Check your spelling of the exception name and make sure the exception was declared properly.

PLS-00486

PLS-00486 SELECT list cannot be enclosed in parentheses

Cause: In a SELECT statement, the select list was enclosed in parentheses, as in:

```
SELECT (deptno, dname, loc) FROM dept INTO ...
```

This breaks the rules of SQL syntax. Parentheses are not needed because the keywords SELECT and FROM delimit the select list.

Action: Remove the parentheses enclosing the select list.

PLS-00487

PLS-00487 invalid reference to variable '*name*'

Cause: A variable was referenced in a way that is inconsistent with its datatype. For example, you might have mistakenly referenced a scalar variable as a record, as follows:

```
DECLARE
```

```
    CURSOR emp_cur IS SELECT empno, ename, sal FROM emp;
```

```
    emp_rec emp_cur%ROWTYPE;
```

```
    my_sal  NUMBER(7,2);
```

```
BEGIN
```

```
    ...
```

```
    total_sal := total_sal + my_sal.sal; -- invalid
```

```
    ...
```

Action: Check your spelling of the variable name. Make sure the variable was declared properly and that the declaration and reference are consistent with regard to datatype.

PLS-00488

PLS-00488 invalid variable declaration: object '*name*' must be a type or subtype

Cause: The datatype specifier in a variable declaration does not designate a legal type. For example, you might have neglected to add the %TYPE attribute to a declaration, as in

```
DECLARE
```

```
my_sal    emp.sal%TYPE;
```

```
my_ename  emp.ename;    -- missing %TYPE
```

```
...
```

When declaring a constant or variable, to provide the datatype of a column automatically, you must use the %TYPE attribute. Likewise, when declaring a record, to provide the datatypes of a row automatically, you must use the %ROWTYPE attribute.

Action: Make sure the datatype specifier designates a legal type. Remember to use the %TYPE and %ROWTYPE attributes when necessary.

PLS-00489

PLS-00489 invalid table reference: 'name' must be a column in this expression

Cause: In a query, a select-list item refers to a table in the FROM clause but not to a database column.

Action: Check your spelling of the column names, make sure each column in the select list refers to a table in the FROM clause, then retry the query.

PLS-00503

PLS-00503 RETURN <value> statement required for this return from function

Cause: In a function body, you used a RETURN statement that contains no expression. In procedures, a RETURN statement contains no expression because the statement simply returns control to the caller. However, in functions, a RETURN statement must contain an expression because its value is assigned to the function identifier.

Action: Add an expression to the RETURN statement.

PLS-00504

PLS-00504 type '*name*'_BASE may not be used outside of package STANDARD

Cause: In a declaration, you mistakenly specified (for example) the datatype NUMBER_BASE. CHAR_BASE, DATE_BASE, MLSLABEL_BASE, and NUMBER_BASE are PL/SQL internal datatypes used within package STANDARD.

Action: Specify (for example) the datatype NUMBER instead of NUMBER_BASE.

PLS-00505

PLS-00505 user-defined types may only be defined as PL/SQL tables or records

Cause: You tried to define a datatype derived from some base type other than RECORD or TABLE. User-defined types must be derived from the RECORD or TABLE type.

Action: Remove the faulty type definition, or define a RECORD or TABLE type.

PLS-00506

PLS-00506 user-defined constrained subtypes are disallowed

Cause: You tried to define a subtype (a subtype associates a base type with a constraint and so defines a subset of values). User-defined subtypes are not allowed in this release of PL/SQL. For example, the following type definition is illegal:

```
SUBTYPE Acronym IS VARCHAR2(5); -- illegal
```

However, future versions of PL/SQL will allow you to define subtypes.

Action: Remove the illegal type definition.

PLS-00507

PLS-00507 PL/SQL tables may not be defined in terms of records or other tables

Cause: In a TABLE type definition, you mistakenly specified a composite datatype (RECORD or TABLE) for the column. The single, unnamed column must belong to a scalar datatype such as CHAR, DATE, or NUMBER.

Action: Remove the TABLE type definition, or replace the composite datatype specifier with a scalar datatype specifier.

PLS-00700

PLS-00700 PRAGMA EXCEPTION_INIT of '*name*' must follow declaration of its exception in same block

Cause: A PRAGMA EXCEPTION_INIT was not declared in the same block as its exception. They must be declared in the proper order in the same block, with the PRAGMA EXCEPTION_INIT declaration following the exception declaration.

Action: Place the PRAGMA EXCEPTION_INIT declaration directly after the declaration of the exception referenced by the pragma.

PLS-00701

PLS-00701 illegal ORACLE error number *num* for PRAGMA EXCEPTION_INIT

Cause: The error number passed to a PRAGMA EXCEPTION_INIT was out of range. The error number must be in the range -9999 .. -1 (excluding -100) for ORACLE errors or in the range -20000 .. -20999 for user-defined errors.

Action: Use a valid error number.

PLS-00702

PLS-00702 second argument to PRAGMA EXCEPTION_INIT must be a numeric literal

Cause: The second argument passed to a PRAGMA EXCEPTION_INIT was something other than a numeric literal (a variable, for example). The second argument must be a numeric literal in the range -9999 .. -1 (excluding -100) for ORACLE errors or in the range -20000 .. -20999 for user-defined errors.

Action: Replace the second argument with a valid error number.

PLS-00703

PLS-00703 multiple instances of named argument in list

Cause: Two or more actual parameters in a subprogram call refer to the same formal parameter.

Action: Remove the duplicate actual parameter.

PLS-00704

PLS-00704 '*name*' must be declared as an exception

Cause: The *exception_name* parameter passed to PRAGMA EXCEPTION_INIT is misspelled or does not refer to a legally declared exception. Or, the pragma is misplaced; it must appear in the same declarative section, somewhere after the exception declaration.

Action: Check your spelling of the *exception_name* parameter. Then, check the exception declaration, making sure the exception name and the keyword EXCEPTION are spelled correctly. Also make sure the pragma appears in the same declarative section somewhere after the exception declaration.

PLS-00705

PLS-00705 exceptions not allowed in an expression

Cause: You mistakenly referred to an exception within an expression. Exceptions have names but not values and therefore cannot contribute values to an expression. For example, the following RETURN statement is illegal:

```
FUNCTION credit_limit (cust_no INTEGER) RETURN NUMBER IS
    limit      NUMBER;
    over_limit EXCEPTION;
    ...
BEGIN
    ...
    RETURN over_limit; -- illegal
END;
```

Action: Check your spelling of the identifiers in the expression, then rewrite the expression so that it does not refer to an exception.

PLS-00900

PLS-00900 can't find body of unit '*name*'

Cause: At run time, the body of a program unit could not be found. This happens, for example, if you reference a procedure for which a specification but no body exists. (No compile-time errors were generated because the specification exists.)

Action: Define a body for the program unit.

PLS-00901

PLS-00901 the datatype of column '*name*' of table '*name*' is not supported

Cause: A column in a database table belongs to a datatype that is not supported by the current release of PL/SQL.

Action: Remove the offending column from the table or copy the desired columns to another table.

PLS-00902

PLS-00902 a READ ONLY bind variable used in OUT or IN OUT context

Cause: A host variable that is protected from update was used in a context that allows an update.

Action: Check the context and change your use of the host variable, or assign the value of the host variable to a PL/SQL local variable, then use the local variable instead.

PLS-00904

PLS-00904 insufficient privilege to access object '*name*'

Cause: You tried to operate on a database object without the required privilege. This error occurs, for example, if you try to UPDATE a table for which you have only SELECT privileges.

Action: Ask your DBA to perform the operation or to grant you the required privilege.

PLS-00905

PLS-00905 object '*name*' is invalid

Cause: You referenced an invalid package specification or stored subprogram. A package specification or stored subprogram is invalid if its source code or any database object it references has been DROPPed, REPLACed, or ALTERed since it was last compiled.

Action: Find out what invalidated the package specification or stored subprogram, then make sure that ORACLE can recompile it without errors.

PLS-00995

PLS-00995 unhandled exception # 'num'

Cause: An exception was raised for which no handler was found. If it cannot find a handler for a raised exception, PL/SQL returns an unhandled exception to the host environment. The number in the message refers to the *ORA-num* listed in Chapter2 of this Manual.

Action: Look up *ORA-num* in Chapter2 of this Manual. Fix the condition that raised the exception, write an appropriate exception handler, or use the OTHERS handler. If there is an appropriate handler in the current block, the exception was raised in a declaration or exception handler. (An exception raised in a declaration or exception handler propagates immediately to the enclosing block.) You can avoid unhandled exceptions by coding an OTHERS handler at the topmost level of every PL/SQL block and subprogram.

PLS-00996

PLS-00996 out of memory

Cause: A request from PL/SQL for more memory failed.

Action: Make sure that you are not referencing the wrong row in a PL/SQL table and that your program is not recursing too deeply.

