Exception Handling

1. Exception Handling:

Exception is an abnormal situation occurs during PL/SQL execution, it means exception is an error situation occurs during PL/SQL. Whenever a predefined error occurs in a program, PL/SQL raises an exception and terminates the PL/SQL execution. For example if you try to divide a number by zero then PL/SQl raise exception called ZERO_DIVIDE.

When PL/SQL raises a predefined exception, program is aborted by displaying error message. To handle exception raised by PL/SQL, You have to use exception handler part of PL/SQL block. When exception handler is used, control is transferred to exception handler whenever there is an exception. There are two type of exception

1. Predefine exceptions or system exceptions.
2. User-defined exceptions.

2. Predefine Exception:

Oracle has defined certain common errors and givens names to these errors which are called as predefined exception.

The following is the list of predefined exception:

Error No	Exception Name	Description
ORA_01476	ZERO_DIVIDE	It raise when number is divided by zero
ORA-65111	CURSOR_ALREADY_OPEN	It raise when you try to open an already open cursor.
ORA-00001	DUP_VAL_ON_INDEX	It raise, when you try store duplicate values in database column that is defined as unique index.
ORA-01001	INVALID_CURSOR	It raise if you try an illegal cursor operation.
ORA-01722	LOGIN_DENIED	It raise if user name or password is not specified correctly.
ORA-01017	NO_DATA_FOUND	It raise when implicit cursor statement returns no rows or f you reference an un initialize row in a PL/SQl
ORA-01403	NOT_LOGGED_ON	This exception is raised when user is not connected to oracle and tries to perform an operation.
ORA-01012	PROGRAM_ERROR	This exception is raised when PL/SQL block has an internal error.
ORA-06504	STORAGE_ERROR	This exception is raised when PL/SQL have insufficient memory to execute the PL/SQL block.
ORA-06500	TIMEOUT_ON_RESOURCE	It raised when oracle is not responding within a specified time limit
ORA-01422	TOO_MANY_ROWS	This exception is raised when a implicit cursor (selection into statement) returns multiple rows
ORA-06502	VALUES_ERROR	This exception is raised whenever the user encounters an arithmetic, conversion or constraints error.

PL1	Write a PL/SQL block to perform division of two number , manage appropriate exception

	declare
	no1 number(3);
	no2 number(3);
	ans number(8,2);
	begin
	no1 := &n1;
	no2 := &n2;
	ans := no1/no2;
	dbms_output.put_line(' ANS = ' || ans);
	exception
	when zero_divide then
	dbms_output.put_line(' Divide Zerro Error');
	end;

EXP	In these program if value of no2 is zero, it will raise exception automatically and execution control is automatically transfer to exception part on execution of ans:=no1/no2 statement and it will handle using ZERO-DIVIDE exception and display appropriate message.

PL2	Write a PL/SQL block to display the student name for given Rollno, manage appropriate exception situation.

	declare
	mrollno student.rollno%type;
	mname student.name%type;
	begin
	mrollno := &roll;
	select name into mname from student where rollno = mrollno;
	dbms_output.put_line(' Name of student is ' || mname);
	exception
	when no_data_found then
	dbms_output.put_line('Data Not Found ');
	when too_many_rows then
	dbms_output.put_line(' More than one record selected ');
	end;

EXP	In these program if select command is successfully fetch one and only one row it will be display name of student, if there is no corresponding data then exception no_data_found will be raised , if more than one records are selected than it will raise too_many_rows exceptin will be raised.

3. SQLCODE and SQLERRM

The SQLCODE function return the error code and SQLERRM returns the corresponding error message.

Both SQLCODE and SQLERRM get values after exception.

You can use this function in exception part as given bellow;

When zero_devide
Dbms_output.put_line(SQLcode || ‘ ‘ || SQLerrm);

4. User Define Exception:

Predefine exceptions are used to manage system errors, where user define exceptions are use to handle the error situation occurs due to contravention of business rules. (e.g. students paying fees installment less than decided by institution.

Unlike predefine exception , user define exception are to be raised explicitly using RAISE command.

The following are steps to work with user define exception:

1. Declare exception in declaration section.

   Syntax: <exception_name> exception;

2. Raise exception using RAISE command.

   Syntax: RAISE exception_name;

3. Define exception in exception section.

PL3	Write a PL/SQL block that explain use of user define exception

	declare
	less_then_5000 exception;
	fees_paid fees.fees_amount%type;
	begin
	select fees_amount into fees_paid from fees where rollno = &rno;
	if fees_paid < 5000 then
	raise less_then_5000;
	end if;
	dbms_output.put_line(' Fees paid ');
	exception
	when less_then_5000 then
	dbms_output.put_line('Fees Not paid');
	when no_data_found then
	dbms_output.Put_line('Studen not found');
	end;

EXP	In above block , there are two exception less_then_5000 is a user define exception where as no_data_found is a system exception , if student has paid fees less than 5000 then it will raise user define exception less_then_5000.

5. RAISE_APPLICATION_ERROR PROCEDURE:

The RAISE_APPLICATION_ERROR procedure, is one of the Oracle utilities, which helps the user to manage the error condition in the application by specifying user defined error number and message.

The RAISE_APPLICATION_ERROR takes two input parameters; the error number (between –20000 and –20999) and the error message that can be up to 2048 bytes long to be displayed.

RAISE_APPLICATION_ERROR stop the execution of corresponding block, rollback all database changes, and give error message.

RAISE_APPLICATION_ERROR used in pl/sql block, procedure, function and triggers. It is mostly used in database trigger.

Syntax: RAISE_APPLICATION_ERROR(<ERROR_NO>, <MESSAGE>);

PL4	Write a PL/SQL block that explain use of raise_application_error.

	declare
	fees_less_than_3000 exception;
	mroll fees.rollno%type;
	mamount fees.fees_amount%type;
	mdate fees.fees_date%type;
	begin
	mroll := &rno;
	mamount := &amt;
	mdate := &dt;
	insert into fees values(mroll,mamount,mdate);
	if mamount < 3000 then
	raise fees_less_than_3000;
	end if;
	exception
	when fees_less_than_3000 then
	raise_application_error(-20001,'FEES LESS THAN 3000');
	end;

Exp	In above block data input by use are inserted into table. if value of fees_amount is less_than_3000 exception is raised. This raised an error using raise_appliation_error and rollback the inserted data.

Review Question

1.	What is Exception?
2.	Explain predefine exception? Give example of various predefine exception.
3.	Explain SQLCODE and SQLERRM
4.	Explain User Define Exception? Give example of user define exception.
5.	Explain RAISE_APPLICATION_ERROR PROCEDURE.