**Types of Exceptions in PL/SQL**

Predefined exceptions are built-in exceptions provided by Oracle PL/SQL to handle common error conditions. Some of the commonly used predefined exceptions include:

* NO\_DATA\_FOUND: Raised when a SELECT INTO statement doesn't return any rows.
* TOO\_MANY\_ROWS: Raised when a SELECT INTO statement returns more than one row.
* ZERO\_DIVIDE: Raised when attempting to divide by zero.
* VALUE\_ERROR: Raised when a conversion or arithmetic operation fails.
* INVALID\_CURSOR: Raised when attempting operations on a closed or invalid cursor.
* LOGIN\_DENIED: Raised when a login to the database fails.
* STORAGE\_ERROR: Raised when there's insufficient memory or storage.
* PROGRAM\_ERROR: Raised for unexpected program errors.
* TIMEOUT\_ON\_RESOURCE: Raised when an operation times out.
* OTHERS: A catch-all exception that can be used to catch any unhandled exception.

**How to handle an exception(**predefined**)**

**Declare**

< variables>

**begin**

<logic>

**exception**

when <Exception name> then

<action>

**end**

**User Defined exceptions**

**DECLARE**

<exception\_name> **EXCEPTION**

**BEGIN**

<logic>

**IF** <condition> **THEN**

RAISE <exception\_name>

**END IF**

**EXCEPTION**

**WHEN** <exception\_name> **THEN**

-- Handle your user-defined exception here

**END.**

**Tasks(neeche wale sare)**

**Using exception handling**

Using the template above, write a PL/SQL code to ask for 2 integers from the user. If the sum of these numbers is even, throw an exception to print “Even Sum”. Also, add a custom exception for division by zero.

**Using Objects**

* Write a PL/SQL to extract and print employee\_id, first\_name and last\_name combined, department\_name, salary incremented by 5%, and job title of employee where employee\_id is 140.
* Write a PL/SQL to extract & print the same details of employees belonging to department 90.

**Using Cursor**

* Write a PL/SQL to extract and print some details of employee, job and department details of employee having employee\_id 120.
* Write a PL/SQL to extract details of employees having employee\_id in range 101-145.

**Triggers**

* Create 2 tables Sellers and Buyers having attributes of both.
* Create 2 more tables: **statements** & **queries**.
* Create another table for transactions.
* Write a PL/SQL code to create triggers that are fired before any of the following statements are executed on either statements/queries table and add details of the given statement to the statements table.
  + Insert
  + Update
  + Delete

**e.g:** For Insert, add a message, “Insert statement executed”.

* Write a PL/SQL code to create triggers that are fired after any of the following statements are executed at row level on buyers OR sellers table. add old data(if delete), new data(if insert), and old & new data (if update) of the given row to the queries table.
  + Insert
  + Update
  + Delete

**e.g:** if the sellers table had data (1, 2500) and you run a query to update unit price to 3000, a trigger should be added on price(sellers table) in such a way that it should add both (2500, 3000) to the queries table.

* Write a PL/SQL code that asks the user for **seller\_id**, **buyer\_id,** and **Price**. Use these inputs to fill both the sellers & buyers table, and add a trigger on insert query in such a way that it adds the newly added details from both of these tables to the transaction table.
* Create a function named getDetails that accepts employee\_id as parameter and returns the full name of that employee.   
  Create a procedure called getID that asks the user for an employee\_id and then makes a call to the getDetails function, if the getDetails function is unable to find data for the given employee\_id, then the details of the first employee(in the database) are returned.

**Example:**

If I pass 5 as the value or any value less than 110, then it should return **John Chen**, as he’s the first employee in the employees table.

But, if I pass 111, it should return **Ismael Sciarra.**

**P.S:**  You can’t use an **IF** check to see if the emp\_id < 110 and then pass 110 as parameter. You have to make the first getDetails call with initial values, and then do the rest later.