# DB LAB

## PL/SQL

```
DECLARE
        <declarations section>      ->      Name varchar2(20) := 'ALI'; OR <exception_name> EXCEPTION
BEGIN
        <executable command(s)>     ->      dbms_output.put_line('NAME:' || Name);
EXCEPTION
        <exception handling>        ->      WHEN <Exception Name> THEN <action>
END;
```

- 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.

To raise exception, we call RAISE <exception_name>

### Differences in PL/SQL

---

```
SELECT DEPARTMENT_ID
INTO DEPTID
FROM EMPLOYEES
WHERE EMPLOYEE_ID = 10;
```

---

```
IF (<condition>) THEN
        <action>
ELSIF (<condition>) THEN
        <action>
ELSE
        <action>
END IF;
```

---

# DB LAB

```
CASE <variable>
        WHEN x1 THEN
                <action>
        WHEN x2 THEN
                <action>
        ELSE
                <action>
END CASE;
```

---

```
FOR x IN 1..<number> LOOP
        <action>
END LOOP;

FOR x IN (SELECT EMPLOYEE_ID FROM EMPLOYEES WHERE DEPARTMENT_ID = 90) LOOP
        <action>
END LOOP;
```

---

```
CREATE OR REPLACE FUNCTION FUNC (VAR_NAME IN <type>, …)
RETURN <type>
IS
<declarations>
BEGIN
        <action>
END;
/
SELECT FUNC(80) FROM DUAL;
```

---

```
CREATE OR REPLACE TYPE EMP_OBJ_TYPE AS OBJECT (
        EMPLOYEE_ID NUMBER(6,0),
        FIRST_NAME VARCHAR(30),
        LAST_NAME VARCHAR(30),
        DEPARTMENT_ID NUMBER(4,0)
);

CREATE TYPE EMP_TBL_TYPE as TABLE OF EMP_OBJ_TYPE;
```

---

# DB LAB

```
CREATE OR REPLACE FUNCTION GETALL
RETURN EMP_TBL_TYPE
IS
        EMPLOYEE_ID NUMBER(6,0);
        FIRST_NAME VARCHAR(30);
        LAST_NAME VARCHAR(30);
        DEPARTMENT_ID NUMBER(4,0);
        EMP_DETAILS EMP_TBL_TYPE := EMP_TBL_TYPE();
BEGIN
        EMP_DETAILS.EXTEND();
        SELECT EMPLOYEE_ID,FIRST_NAME, LAST_NAME,DEPARTMENT_ID INTO
        EMPLOYEE_ID,FIRST_NAME,LAST_NAME,DEPARTMENT_ID FROM EMPLOYEES where
        EMPLOYEE_ID=100;
        EMP_DETAILS(1) := EMP_OBJ_TYPE(EMPLOYEE_ID,FIRST_NAME,LAST_NAME,DEPARTMENT_ID);
        RETURN EMP_DETAILS;
END;
/

CREATE OR REPLACE FUNCTION GETALL1
RETURN EMP_TBL_TYPE
IS
        EMPLOYEE_ID NUMBER(6,0);
        FIRST_NAME VARCHAR(30);
        LAST_NAME VARCHAR(30);
        DEPARTMENT_ID NUMBER(4,0);
        EMP_DETAILS EMP_TBL_TYPE := EMP_TBL_TYPE();
BEGIN
        EMP_DETAILS.EXTEND();
        SELECT EMP_OBJ_TYPE( EMPLOYEE_ID,FIRST_NAME, LAST_NAME,DEPARTMENT_ID) BULK COLLECT
        INTO EMP_DETAILS FROM EMPLOYEES;
        RETURN EMP_DETAILS;
END;
/
```

# DB LAB

```
CREATE OR REPLACE PROCEDURE PROC_NAME (<params (like func)>)
IS
<declarations>
BEGIN
        <action>
END;
/
EXEC PROC_NAME(<args>)
```

---

```
DECLARE
        CURSOR Cursor_EMP IS SELECT * FROM employees ORDER BY salary DESC;
        row_emp Cursor_EMP%ROWTYPE;
BEGIN
        OPEN Cursor_EMP;
        LOOP
                FETCH Cursor_EMP INTO row_emp;
                EXIT WHEN Cursor_EMP%NOTFOUND;
                DBMS_OUTPUT.PUT_LINE( 'EMPLOYEE id: ' ||row_emp.EMPLOYEE_ID || ' EMPLOYEE NAME: '
                || row_emp.FIRST_NAME || ' EMPLOYEE CONTACT: ' || row_emp.PHONE_NUMBER || '.');
        END LOOP;
        CLOSE Cursor_EMP;
END;
/
```

---

```
CREATE OR REPLACE TRIGGER TRIGGER_NAME
{BEFORE|AFTER|INSTEAD OF}
{INSERT [OR] | UPDATE [OR] | DELETE}
[OF col_name]
ON table_name
[FOR EACH ROW]
WHEN (<condition>)
BEGIN
        <actions>
END;
```

---

# DB LAB

SET TRANSACTION ISOLATION LEVEL [READ COMMITTED | READ UNCOMMITED | SERIALIZABLE | REPEATABLE READ];

SET TRANSACTION READ [WRITE | ONLY];

SET TRANSACTION NAME ‘NAME’;

SET TRANSACTION [DEFERRABLE | NOT DEFERRABLE];

SAVEPOINT SPNAME;

ROLLBACK TO SAVEPOINT SPNAME;

---

CREATE INDEX <index_name> ON <table_name> (column1, column2);

DROP INDEX <index_name>;

---

SEE MONGODB LAB