Introduction To Trigger ====>
==========

* Triggers in Oracle are blocks of PL/SQL code which Oracle engine can execute automatically based on some action or event.

* These events can be:
- DDL statements (CREATE, ALTER, DROP, TRUNCATE)
- DML statements (INSERT, SELECT, UPDATE, DELETE)
- Database operation like connecting or disconnecting to Oracle (LOGON, LOGOFF, SHUTDOWN)

Uses Of Trigger ===>
==========

* Following are use cases where using triggers proves very helpful:
- Maintaining complex constraints which is either impossible or very difficult via normal constraint(like primary, foreign, unique etc) applying technique.
- Recording the changes made on the table.
- Automatically generating primary key values.

Syntax ==>

```
CREATE OR REPLACE TRIGGER trigger_name
BEFORE | AFTER
INSERT | UPDATE | DELETE
    ON table_name
    [FOR EACH ROW]
DECLARE
    Declaration-statements
BEGIN
    Executable-statements
EXCEPTION
    Exception-handling-statements
END;
```

* Following is the explanation of statements that are present in trigger creation.
- BEFORE/ AFTER will specify the event timings.
- INSERT/UPDATE/DELETE/etc. will specify the event for which the triggerneeds to be fired.
- ON clause will specify on which object the above-mentioned event is valid. Forexample, this will be the table name on which the DML event may occur in the case of DML Trigger.
- Command "FOR EACH ROW" will specify the ROW level trigger.
- The declaration part, execution part, exception handling part is same as that of the other PL/SQL blocks. Declaration part and exception handling part are optional.

Trigger Pseudo Variables ===>
==========

* In a row level trigger, the trigger fires for each related row.

* And sometimes it is required to know the column values of the effected row before and after the DML statement.

* Oracle has provided two clauses in the ROW-level trigger to hold these values. We can use these clauses to refer to the old and new values inside the trigger body.

* ` :NEW ` – It holds a new value for the columns of the base table/view during the trigger execution

* ` :OLD ` – It holds old value of the columns of the base table/view during the trigger execution.

Rules Regarding :new And :old ==>
==========

* If the trigger is an after trigger then we cannot change any field value of :new

* We can change the field values of :new for before trigger but it is not allowed for :old

Exercise ==>

Create a trigger called checkprice that prevents users from updating or inserting any record in ALLBOOKS table with bookprice other than the range 500 to 700.

```
create or replace trigger checkprice
before insert or update
on allbooks
for each row
begin
    if :new.bookprice not between 500 and 700 then
        raise_application_error(-20001, 'Bookprice cannot be other than 500 to
700');
    end if;
end checkprice;
```

Create a trigger called checkday that prevents users from updating or inserting any record in EMP table with hiredate that falls on weekends.

```
create or replace trigger checkday
before insert or update
on emp
for each row
begin
    if to_char(:new.hiredate,'DY') in ('SAT','SUN') then
        raise_application_error(-20001,' Hiring is not done on weekends');
    end if;
end checkday;
```

Performing Audit Trail ===>
==========

* Trigger can also be used to perform audit trail on tables.

* For example , if a record is deleted from a table then we can use a trigger to save the details of the deleted record to another table.

Example ==>

```
CREATE OR REPLACE TRIGGER books_audit_trg
AFTER DELETE ON allbooks
FOR EACH ROW
BEGIN
    INSERT INTO audits (bookid, transaction_name, by_user, transaction_date)
    VALUES(:old.book_id,'DELETE' , USER, SYSDATE);
END;
```

Triggering Flags ===>
==========

* When a trigger is defined for multiple DML events, Oracle provides us three flags called INSERTING , UPDATING and DELETING.

* One of them becomes true depending on the triggering event.

* So , event-specific code can be defined using the INSERTING, UPDATING, DELETING flags.

Modify the previous trigger so that if update is issued then also it records that.

```
CREATE OR REPLACE TRIGGER books_audit_trg
AFTER UPDATE OR DELETE ON allbooks
FOR EACH ROW
DECLARE
    trans VARCHAR2(10);
BEGIN
    if UPDATING then
        trans:='UPDATE';
    else
        trans:='DELTETE';
    end if;
    INSERT INTO audits (bookid, transaction_name, by_user,
    transaction_date) VALUES(:old.book_id,trans, USER, SYSDATE);
END;
```