Triggers:

Applications:

1. DB Security → Control User Access (Can manage DDL, DML Changes)
2. DB Auditing →
3. Referential Integrity → When Foreign Key is defined
4. Monitoring DB Events
    a. LOG OFF
    b. LOG ON
    c. STARTUP
    d. SHUTDOWN
    e. Alert.log file → DBA can trace complete activity of user like who was logged on, what happened and what was the issue and how to solve it.
5. Replication → To populate views and snapshots after changing table such that we don't have to ask for updated data from table → Now we are using MVs for this

Triggers Type:

1. DB Level
    a. DML
    b. DDL
    c. INSTEAD OF
2. Application Level
    a. If you want to block something at application level for a database.

- ## Another Type:

1. Row level:
    a. To Create Row Level trigger, we have to use key word: **FOR EACH ROW.** When you don't use this keyword by default it will be taken as Statement Level triggers.
2. Statement Level
    a. You can specify when you want use trigger in statements **using BEFORE or AFTER ROW** Statements.
    b. Example,
        i. BEFORE STATEMENT
        ii. AFTER STATEMENT

Note that Triggers are Implicitly Invoked. Triggers are also an object.

Statement-Level Triggers Versus Row-Level Triggers

Statement-Level Triggers	Row-Level Triggers
Is the default when creating a trigger	Use the FOR EACH ROW clause when creating a trigger.
Fires once for the triggering event	Fires once for each row affected by the triggering event
Fires once even if no rows are affected	Does not fire if the triggering event does not affect any rows

- When a row-level trigger fires, the PL/SQL run-time engine creates and populates two data structures:
  - OLD: Stores the original values of the record processed by the trigger
  - NEW: Contains the new values
- NEW and OLD have the same structure as a record declared using the %ROWTYPE on the table to which the trigger is attached.

Data Operations	Old Value	New Value
INSERT	NULL	Inserted value
UPDATE	Value before update	Value after update
DELETE	Value before delete	NULL

successfully.

```
CREATE OR REPLACE TRIGGER mytrg
  BEFORE INSERT ON mytable FOR EACH ROW
  DISABLE
BEGIN
  :New.ID := my_seq.Nextval;
. . .
END;
/
```

Viewing Trigger Information

You can view the following trigger information:

Data Dictionary View	Description
USER_OBJECTS	Displays object information
USER/ALL/DBA_TRIGGERS	Displays trigger information
USER_ERRORS	Displays PL/SQL syntax errors for a trigger

Timing-Point Sections of a Table Compound Trigger

A compound trigger defined on a table has one or more of the following timing-point sections. Timing-point sections must appear in the order shown in the table.

Timing Point	Compound Trigger Section
Before the triggering statement executes	BEFORE statement
After the triggering statement executes	AFTER statement
Before each row that the triggering statement affects	BEFORE EACH ROW
After each row that the triggering statement affects	AFTER EACH ROW