

EX.NO.: 13		WORKING WITH TRIGGERS
DATE :	8/10/2024	

#### PROGRAM 1

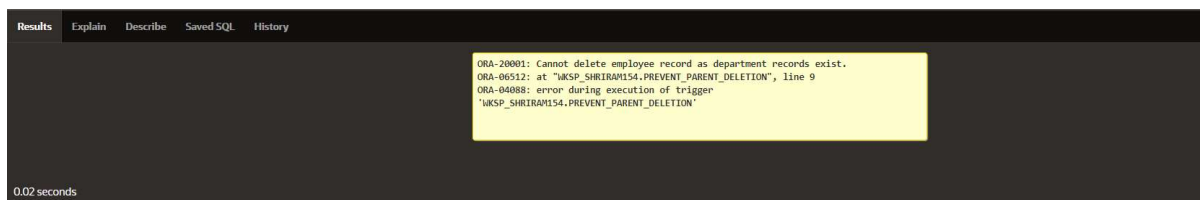
WRITE A CODE IN PL/SQL TO DEVELOP A TRIGGER THAT ENFORCES REFERENTIAL INTEGRITY BY PREVENTING THE DELETION OF A PARENT RECORD IF CHILD RECORDS EXIST.

```

CREATE OR REPLACE TRIGGER
PREVENT_PARENT_DELETION BEFORE DELETE ON
EMPLOYEES
FOR EACH ROW
DECLARE
    PL_DEPT_COUNT
NUMBER;
BEGIN
    SELECT COUNT(*)
    INTO
    PL_DEPT_COUNT
    FROM
    DEPARTMENT
    WHERE DEPT_ID =
    :OLD.EMPLOYEE_ID;
    IF
    PL_DEPT_COUNT > 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'CANNOT DELETE EMPLOYEE
RECORD AS DEPARTMENT RECORDS EXIST. ');
    END IF;
END;
```

```

DELETE FROM
EMPLOYEES WHERE
EMPLOYEE_ID = 70;
```

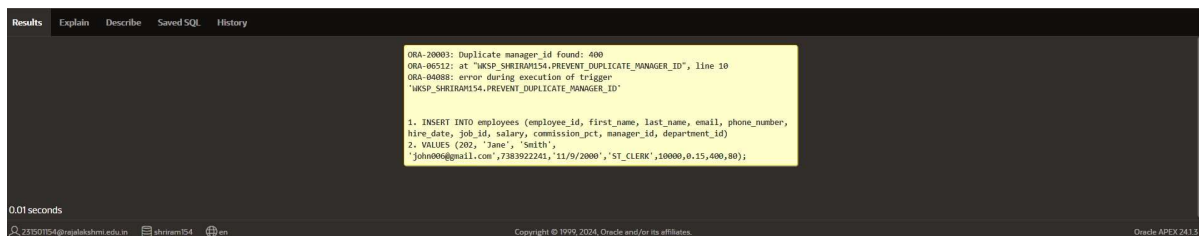


## PROGRAM 2

WRITE A CODE IN PL/SQL TO CREATE A TRIGGER THAT CHECKS FOR DUPLICATE VALUES IN A SPECIFIC COLUMN AND RAISES AN EXCEPTION IF FOUND.

```
CREATE OR REPLACE TRIGGER
PREVENT_DUPLICATE_MANAGER_ID BEFORE INSERT OR
UPDATE ON EMPLOYEES
FOR EACH ROW
DECLARE
    PL_COUNT
NUMBER; BEGIN
    SELECT COUNT(*)
    INTO PL_COUNT
    FROM
    EMPLOYEES
    WHERE      MANAGER_ID      =
:NEW.MANAGER_ID AND EMPLOYEE_ID
!= :NEW.EMPLOYEE_ID; IF PL_COUNT >
0 THEN
    RAISE_APPLICATION_ERROR(-20003, 'DUPLICATE MANAGER_ID FOUND: ' ||
:NEW.MANAGER_I
D); END IF;
END;
```

```
INSERT INTO EMPLOYEES (EMPLOYEE_ID, FIRST_NAME, LAST_NAME, EMAIL,
PHONE_NUMBER, HIRE_DATE, JOB_ID, SALARY, COMMISSION_PCT, MANAGER_ID,
DEPARTMENT_ID)
VALUES (202, 'JANE', 'SMITH',
'JOHN006@GMAIL.COM', 7383922241, '11/9/2000', 'ST_CLERK', 10000, 0.15, 400, 80);
```



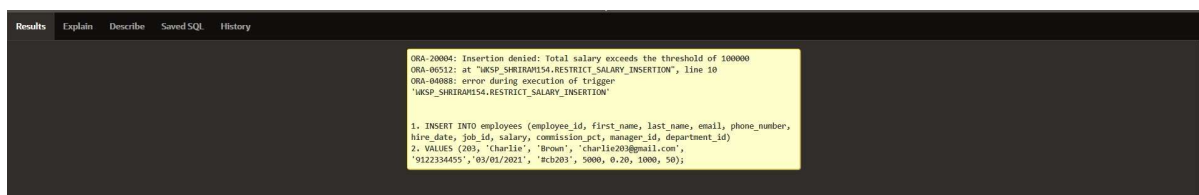
### PROGRAM 3

WRITE A CODE IN PL/SQL TO CREATE A TRIGGER THAT RESTRICTS THE INSERTION OF NEW ROWS IF THE TOTAL OF A COLUMN'S VALUES EXCEEDS A CERTAIN THRESHOLD.

```
CREATE OR REPLACE TRIGGER
RESTRICT_SALARY_INSERTION BEFORE INSERT ON
EMPLOYEES
FOR EACH ROW
DECLARE
    TOTAL_SALARY NUMBER;
    THRESHOLD NUMBER :=
    100000;
BEGIN

    SELECT
    SUM(SALARY) INTO
    TOTAL_SALARY
    FROM EMPLOYEES;
    IF (TOTAL_SALARY + :NEW.SALARY) > THRESHOLD THEN
        RAISE_APPLICATION_ERROR(-20004, 'INSERTION DENIED: TOTAL SALARY
EXCEEDS THE THRESHOLD OF ' || THRESHOLD);
    END IF;
END;
```

```
INSERT INTO EMPLOYEES (EMPLOYEE_ID, FIRST_NAME, LAST_NAME, EMAIL,
PHONE_NUMBER, HIRE_DATE, JOB_ID, SALARY, COMMISSION_PCT, MANAGER_ID,
DEPARTMENT_ID)
VALUES (203, 'CHARLIE', 'BROWN', 'CHARLIE203@GMAIL.COM',
'9122334455', '03/01/2021', '#CB203', 5000, 0.20, 1000, 50);
```



```
ORA-20004: Insertion denied: Total salary exceeds the threshold of 100000
ORA-00512: at 'WKSP_SHRIRAM154.RESTRICT_SALARY_INSERTION', line 10
ORA-04088: error during execution of trigger
'WKSP_SHRIRAM154.RESTRICT_SALARY_INSERTION'

1. INSERT INTO employees (employee_id, first_name, last_name, email, phone_number,
hire_date, job_id, salary, commission_pct, manager_id, department_id)
2. VALUES (203, 'charlie', 'Brown', 'charlie203@gmail.com',
'9122334455', '03/01/2021', '#CB203', 5000, 0.20, 1000, 50);
```



#### PROGRAM 4

WRITE A CODE IN PL/SQL TO DESIGN A TRIGGER THAT CAPTURES CHANGES MADE TO SPECIFIC COLUMNS AND LOGS THEM IN AN AUDIT TABLE.

```
CREATE OR REPLACE TRIGGER
AUDIT_CHANGES AFTER UPDATE OF SALARY,
JOB_ID ON EMPLOYEES FOR EACH ROW
BEGIN
    IF :OLD.SALARY != :NEW.SALARY OR :OLD.JOB_ID != :NEW.JOB_ID
    THEN INSERT INTO EMPLOYEE_AUDIT (
        EMPLOYEE_ID,
        OLD_SALARY,
        NEW_SALARY,
        OLD_JOB_TITLE,
        NEW_JOB_TITLE,
        CHANGE_TIMESTAMP,
        CHANGED_BY
    ) VALUES (
        :OLD.EMPLOYEE_ID,
        :OLD.SALARY,
        :NEW.SALARY,
        :OLD.JOB_ID,
        :NEW.JOB_ID,
        SYSTIMESTAMP,
        USER
    );
    END IF;
END;

UPDATE EMPLOYEES
SET SALARY = 55000, JOB_ID =
'ST_CLERK' WHERE EMPLOYEE_ID =
176;

SELECT * FROM EMPLOYEE_AUDIT;
```

AUDIT_ID	EMPLOYEE_ID	OLD_SALARY	NEW_SALARY	OLD_JOB_ID	NEW_JOB_ID	CHANGE_TIMESTAMP	CHANGED_BY
1	20	50000	55000	manager	manager	15-OCT-24 10:00:00.000000 AM	admin
2	122	60000	65000	Manager	Manager	15-OCT-24 10:15:00.000000 AM	admin
5	27	45000	47000	Analyst	Senior Analyst	15-OCT-24 10:30:00.000000 AM	user1
22	176	7500	55000	#ce005	ST_CLERK	16-OCT-24 04:25:06.252580 PM	APEX_PUBLIC_USER
3	9	70000	75000	Senior Developer	Lead Developer	15-OCT-24 10:45:00.000000 AM	user2
4	4	80000	85000	Team Lead	Project Manager	15-OCT-24 11:00:00.000000 AM	admin

6 rows returned in 0.00 seconds [Download](#)

## PROGRAM 5

WRITE A CODE IN PL/SQL TO IMPLEMENT A TRIGGER THAT RECORDS USER ACTIVITY (INSERTS, UPDATES,DELETES) IN AN AUDIT LOG FOR A GIVEN SET OF TABLES.

```
CREATE OR REPLACE TRIGGER
TRG_AUDIT_EMPLOYEES AFTER INSERT OR UPDATE
OR DELETE ON EMPLOYEES FOR EACH ROW
DECLARE
    V_OLD_VALUES
        CLOB;
    V_NEW_VALUES
        CLOB;
BEGIN
    IF INSERTING THEN
        V_OLD_VALUES := NULL;
        V_NEW_VALUES := 'EMPLOYEE_ID: ' ||
            :NEW.EMPLOYEE_ID || ', ' || 'FIRST_NAME: ' ||
            :NEW.FIRST_NAME || ', ' ||
            'SALARY: ' || :NEW.SALARY;

        INSERT INTO AUDIT_LOG (ACTION, TABLE_NAME, RECORD_ID, CHANGED_BY,
            NEW_VALUES) VALUES ('INSERT', 'EMPLOYEES', :NEW.EMPLOYEE_ID, USER,
            V_NEW_VALUES);

    ELSIF UPDATING THEN
        V_OLD_VALUES := 'EMPLOYEE_ID: ' ||
            :OLD.EMPLOYEE_ID || ', ' || 'FIRST_NAME: ' ||
            :OLD.FIRST_NAME || ', ' ||
            'SALARY: ' || :OLD.SALARY;
        V_NEW_VALUES := 'EMPLOYEE_ID: ' ||
            :NEW.EMPLOYEE_ID || ', ' || 'FIRST_NAME: ' ||
            :NEW.FIRST_NAME || ', ' ||
            'SALARY: ' || :NEW.SALARY;

        INSERT INTO AUDIT_LOG (ACTION, TABLE_NAME, RECORD_ID,
            CHANGED_BY, OLD_VALUES, NEW_VALUES)
            VALUES ('UPDATE', 'EMPLOYEES', :NEW.EMPLOYEE_ID, USER,
            V_OLD_VALUES, V_NEW_VALUES);

    ELSIF DELETING THEN
        V_OLD_VALUES := 'EMPLOYEE_ID: ' ||
            :OLD.EMPLOYEE_ID || ', ' || 'FIRST_NAME: ' ||
```

```
        :OLD.FIRST_NAME || ', ' ||  
        'SALARY: ' ||  
:OLD.SALARY;V_NEW_VALUES  
:= NULL;  
  
INSERT INTO AUDIT_LOG (ACTION, TABLE_NAME, RECORD_ID,  
CHANGED_BY, OLD_VALUES)VALUES ('DELETE', 'EMPLOYEES',  
:OLD.EMPLOYEE_ID, USER, V_OLD_VALUES);  
END IF;  
        END  
TRG_AUDIT_EMPLOYEES;
```



```
INSERT INTO EMPLOYEES (EMPLOYEE_ID, FIRST_NAME,
SALARY)VALUES (3, 'BALL', 50000);
```

Results	Explain	Describe	Saved SQL	History
1 row(s) inserted.				
0.12 seconds				

```
UPDATE
EMPLOYEESSET
SALARY = 55000
WHERE EMPLOYEE_ID = 3;
```

1 row(s) updated.				
0.06 seconds				

```
DELETE FROM
EMPLOYEESWHERE
EMPLOYEE_ID = 3;
```

```
SELECT * FROM AUDIT_LOG;
```

AUDIT_ID	ACTION	TABLE_NAME	RECORD_ID	CHANGED_BY	CHANGE_TIMESTAMP	OLD_VALUES	NEW_VALUES
1	INSERT	employees	3	APEX_PUBLIC_USER	16-OCT-24 04.39:17:957308 PM	-	employee_id: 3, first_name: Ball, salary: 50000
3	DELETE	employees	3	APEX_PUBLIC_USER	16-OCT-24 04.41:49:077471 PM	employee_id: 3, first_name: Ball, salary: 55000	-
2	UPDATE	employees	3	APEX_PUBLIC_USER	16-OCT-24 04.40:03:193035 PM	employee_id: 3, first_name: Ball, salary: 50000	employee_id: 3, first_name: Ball, salary: 55000

3 rows returned in 0.00 seconds [Download](#)

## PROGRAM 6

WRITE A CODE IN PL/SQL TO IMPLEMENT A TRIGGER THAT AUTOMATICALLY CALCULATES AND UPDATES A RUNNING TOTAL COLUMN FOR A TABLE WHENEVER NEW ROWS ARE INSERTED.

```
CREATE TABLE TRANSACTIONS (  
    TRANSACTION_ID NUMBER PRIMARY  
    KEY, AMOUNT NUMBER,  
    RUNNING_TOTAL NUMBER  
);
```

```
CREATE OR REPLACE TRIGGER  
UPDATE_RUNNING_TOTAL FOR INSERT ON  
TRANSACTIONS  
COMPOUND TRIGGER
```

```
    TYPE AMOUNT_ARRAY IS TABLE OF NUMBER INDEX BY PLS_INTEGER;  
    NEW_AMOUNTS  
    AMOUNT_ARRAY;
```

```
    BEFORE EACH ROW IS  
    BEGIN  
        NEW_AMOUNTS(:NEW.TRANSACTION_ID) :=  
        :NEW.AMOUNT; END BEFORE EACH ROW;
```

```
    AFTER STATEMENT IS  
    BEGIN  
        DECLARE  
            V_TOTAL  
            NUMBER; BEGIN  
            SELECT  
                NVL(MAX(RUNNING_TOTAL), 0)  
            INTO V_TOTAL  
            FROM TRANSACTIONS;
```

```
            FOR I IN NEW_AMOUNTS.FIRST .. NEW_AMOUNTS.LAST  
            LOOP V_TOTAL := V_TOTAL + NEW_AMOUNTS(I);  
            UPDATE TRANSACTIONS  
            SET RUNNING_TOTAL =  
            V_TOTAL WHERE  
            TRANSACTION_ID = I;  
            END LOOP;  
        END;
```

**END AFTER STATEMENT;**

**END UPDATE\_RUNNING\_TOTAL;**

**INSERT INTO TRANSACTIONS (TRANSACTION\_ID, AMOUNT)**

**VALUES (1, 10000);**

**INSERT INTO TRANSACTIONS (TRANSACTION\_ID,  
AMOUNT)VALUES (2, 20000);**

Results Explain Describe Saved SQL History		
TRANSACTION_ID	AMOUNT	RUNNING_TOTAL
1	10000	10000
2	20000	30000
2 rows returned in 0.01 seconds <a href="#">Download</a>		

## PROGRAM 7

WRITE A CODE IN PL/SQL TO CREATE A TRIGGER THAT VALIDATES THE AVAILABILITY OF ITEMS BEFORE ALLOWING AN ORDER TO BE PLACED, CONSIDERING STOCK LEVELS AND PENDING ORDERS.

```
CREATE TABLE INVENTORY (  
    ITEM_ID NUMBER PRIMARY  
    KEY,  
    ITEM_NAME  
    VARCHAR2(100),  
    STOCK_LEVEL NUMBER  
);
```

```
CREATE TABLE ORDERS (  
    ORDER_ID NUMBER PRIMARY  
    KEY, ITEM_ID NUMBER,  
    QUANTITY NUMBER,  
    ORDER_STATUS  
    VARCHAR2(20),  
    CONSTRAINT FK_ITEM FOREIGN KEY (ITEM_ID) REFERENCES INVENTORY(ITEM_ID)  
);
```

```
CREATE OR REPLACE TRIGGER  
VALIDATE_STOCK_BEFORE_ORDER BEFORE INSERT ON  
ORDERS  
FOR EACH ROW  
DECLARE  
    V_STOCK_LEVEL NUMBER;  
    V_PENDING_ORDERS  
    NUMBER;  
BEGIN  
    SELECT  
        STOCK_LEVEL INTO  
        V_STOCK_LEVEL  
    FROM INVENTORY  
    WHERE ITEM_ID =  
        :NEW.ITEM_ID; SELECT  
        NVL(SUM(QUANTITY), 0) INTO  
        V_PENDING_ORDERS
```

```
FROM ORDERS
WHERE ITEM_ID =
:NEW.ITEM_IDAND
ORDER_STATUS =
'PENDING';
IF (:NEW.QUANTITY + V_PENDING_ORDERS) > V_STOCK_LEVEL THEN
    RAISE_APPLICATION_ERROR(-20001, 'INSUFFICIENT STOCK FOR ITEM: ' ||
:NEW.ITEM_ID);
END IF;
END;
```

```
INSERT INTO ORDERS (ORDER_ID, ITEM_ID, QUANTITY,  
ORDER_STATUS)VALUES (1, 101, 5, 'PENDING');
```

```
1 row(s) inserted.
```

```
0.03 seconds
```

```
INSERT INTO ORDERS (ORDER_ID, ITEM_ID, QUANTITY,  
ORDER_STATUS)VALUES (2, 103, 20, 'PENDING');
```

```
ORA-20001: Insufficient stock for item: 103  
ORA-06512: at "WKSP_SHRIRAM154.VALIDATE_STOCK_BEFORE_ORDER", line 15  
ORA-04088: error during execution of trigger  
'WKSP_SHRIRAM154.VALIDATE_STOCK_BEFORE_ORDER'
```

```
1. INSERT INTO orders (order_id, item_id, quantity, order_status)  
2. VALUES (2, 103, 20, 'Pending');
```

ITEM_ID	ITEM_NAME	STOCK_LEVEL
101	hp_laptop	10
102	keyboard	20
103	mouse	15

5 rows returned in 0.01 seconds [Download](#)

ORDER_ID	ITEM_ID	QUANTITY	ORDER_STATUS
1	101	5	Pending

1 rows returned in 0.01 seconds [Download](#)