**Function:**

CREATE or REPLACE FUNCTION count\_consumer\_rows

RETURN NUMBER

IS

row\_count NUMBER;

BEGIN

SELECT COUNT(\*) INTO row\_count FROM Consumer;

RETURN row\_count;

EXCEPTION

WHEN OTHERS THEN

RETURN NULL;

END;

**Procedures:**

1)create or replace PROCEDURE count\_unsent\_rows

IS

v\_count NUMBER;

BEGIN

SELECT COUNT(\*)

INTO v\_count

FROM DELIVERATION

WHERE status = 'unsent';

DBMS\_OUTPUT.PUT\_LINE('Number of rows with status "unsent": ' v\_count);

END;

BEGIN

count\_unsent\_rows;

END;

2)

create or replace PROCEDURE show\_cash\_total AS

cash\_total NUMBER;

BEGIN

SELECT SUM(total\_price) INTO cash\_total

FROM Payment

WHERE method\_payment = 'Cash';

DBMS\_OUTPUT.PUT\_LINE('Total money paid with cash: ' cash\_total);

END;

BEGIN

show\_cash\_total;

END;

**Triggers:**

1)

create or replace TRIGGER show\_row\_count\_payment

AFTER INSERT ON PAYMENT

DECLARE

row\_count NUMBER;

BEGIN

SELECT COUNT(\*) INTO row\_count FROM PAYMENT;

DBMS\_OUTPUT.PUT\_LINE('Current number of rows in PAYMENT: ' row\_count);

END;

/

2)

create or replace TRIGGER update\_aff\_income

AFTER INSERT ON Payment

FOR EACH ROW

BEGIN

UPDATE All\_for\_football

SET income = income + :NEW.total\_price;

END;

/

3)

create or replace TRIGGER update\_total\_payment

AFTER INSERT ON Payment

FOR EACH ROW

BEGIN

UPDATE Consumer SET total\_payment = total\_payment + :NEW.total\_price

WHERE ConID = :NEW.conid;

END;

/

**Row count:**

CREATE OR REPLACE PROCEDURE update\_boot\_price(p\_size IN NUMBER, p\_price IN NUMBER)

IS

BEGIN

UPDATE Boots

SET prcie = prcie + p\_price

WHERE size\_boots = p\_size;

DBMS\_OUTPUT.PUT\_LINE('Number of rows updated: ' SQL%ROWCOUNT);

END;

BEGIN

update\_boot\_price(41, 150);

END;