Add the following:

- Procedure which does group by information

- Function which counts the number of records

- Procedure which uses SQL%ROWCOUNT to determine the number of rows affected

- Add user-defined exception which disallows to enter title of item (e.g. book) to be less than 5 characters

- Create a trigger before insert on any entity which will show the current number of rows in the table

- Procedure which does group by information

CREATE OR REPLACE PROCEDURE group\_by\_proc IS

item\_id order\_line.itemid%TYPE;

quantity order\_line.order\_quantity%TYPE;

p\_total order\_line.total\_price%TYPE;

CURSOR c\_group\_by IS

SELECT itemid, SUM(order\_quantity) AS quantity, SUM(total\_price) AS total

FROM order\_line

GROUP BY itemid

ORDER BY itemid;

BEGIN

OPEN c\_group\_by;

LOOP

FETCH c\_group\_by INTO item\_id, quantity, p\_total;

EXIT WHEN c\_group\_by%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('ID: ' || item\_id || ', Count: ' || quantity || ' - Total: ' || p\_total);

IF quantity > 15 THEN

DBMS\_OUTPUT.PUT\_LINE('The quantity has exceeded 15, you need to order more!');

DBMS\_OUTPUT.PUT\_LINE('');

END IF;

END LOOP;

CLOSE c\_group\_by;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No data found');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLCODE || ' - ' || SQLERRM);

END;

BEGIN

group\_by\_proc;

END

- Function which counts the number of records

1️⃣ COUNT ALL ↓

CREATE OR REPLACE FUNCTION count\_records

RETURN NUMBER

IS

record\_count NUMBER;

BEGIN

SELECT COUNT(\*) INTO record\_count FROM COUNTRY;

RETURN record\_count;

EXCEPTION

WHEN OTHERS THEN

RETURN NULL;

END;

DECLARE

rec\_count NUMBER;

BEGIN

rec\_count := count\_records();

DBMS\_OUTPUT.PUT\_LINE('Number of records: ' || rec\_count);

END;

2️⃣COUNT NUMBER OF USERS WHO PAID BY CARD

CREATE OR REPLACE FUNCTION payment\_count

RETURN NUMBER

IS

count\_it NUMBER;

BEGIN

SELECT COUNT (\*) INTO count\_it FROM PAYMENT\_TYPEMETHOD

WHERE PAYMENT\_TYPE\_ID = 11;

RETURN count\_it;

EXCEPTION WHEN OTHERS THEN

RETURN NULL;

END;

DECLARE

REC\_NUM NUMBER;

BEGIN

REC\_NUM := payment\_count();

DBMS\_OUTPUT.PUT\_LINE('Number of users paid by card: ' || REC\_NUM);

END;

3️⃣COUNT NUMBER OF ITEMS IN THE BASKET WHICH ARE CHOSEN MORE THAN 1 TIME (QUANTITY IN BASKET OF THE SAME ITEM > 1)

CREATE OR REPLACE FUNCTION basket\_count

RETURN NUMBER

IS

count\_it NUMBER;

BEGIN

SELECT COUNT(BASKET\_ITEMID) INTO count\_it FROM SHOPPING\_BASKET\_ITEM

INNER JOIN SHOPPING\_BASKET

ON SHOPPING\_BASKET\_ITEM.SHOPPING\_BASKETID = SHOPPING\_BASKET.SHOPPING\_BASKETID

WHERE QUANTITY > 1;

return count\_it;

EXCEPTION WHEN OTHERS THEN

RETURN NULL;

END;

DECLARE

REC\_C NUMBER;

BEGIN

REC\_C := basket\_count();

DBMS\_OUTPUT.PUT\_LINE('Number of suppliers who serves Jewelry variations: ' || REC\_C );

END;

- Procedure which uses SQL%ROWCOUNT to determine the number of rows affected

DECLARE

total\_rows number(6);

BEGIN

UPDATE productt\_item

SET price = price + 10;

IF sql%notfound THEN

dbms\_output.put\_line('no product selected');

ELSIF sql%found THEN

total\_rows := sql%rowcount;

dbms\_output.put\_line( total\_rows || ' products selected ');

END IF;

END;

- Add user-defined exception which disallows to enter title of item (e.g. book) to be less than 5 characters

DECLARE

invalid\_title\_exception EXCEPTION;

item\_id NUMBER;

cat\_id Number;

item\_title VARCHAR2(50);

item\_des VARCHAR2(50);

BEGIN

item\_id := :item\_id;

cat\_id := :cat\_id;

item\_title := :item\_title;

item\_des := :item\_des;

IF LENGTH(item\_title) < 5 THEN

RAISE invalid\_title\_exception;

ELSE

INSERT INTO productt (productid, categoryid, productn, product\_des) VALUES (item\_id, cat\_id, item\_title, item\_des);

END IF;

EXCEPTION

WHEN invalid\_title\_exception THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Error: Item title must be at least 5 characters long.');

END;

- Create a trigger before insert on any entity which will show the current number of rows in the table

CREATE OR REPLACE TRIGGER display\_user\_count

BEFORE INSERT ON user\_info

FOR EACH ROW

WHEN (NEW.USERID > 0)

declare

cursor c is select \* from user\_info;

n number;

begin

n := 0;

for row in c loop

n := n + 1;

end loop;

dbms\_output.put\_line('Before inserting row count = ' || n);

end;