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

1 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;
```

2 COUNT 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;
```

3COUNT 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 + 20
 where productid in (select productid
                  from productt, product_category
                  where productt.categoryid = product_category.categoryid and categoryn =
'Jewelry');


 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;
```