MET CS 669 Database Design and Implementation for Business
Term Project

Name: Yuxiao Wu

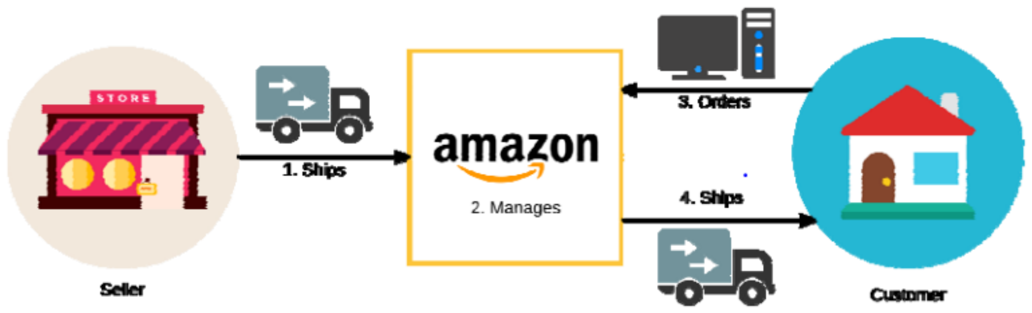
Table of Contents

Project Direction Overview

Introduction to Selling on Amazon

Are there any online marketplaces more reputable than Amazon? Probably not. Why not? For one, Amazon has successfully made use of many unique innovations. For another, Amazon has significantly systematized the selling process, structuring a generalized marketplace which virtually any seller can plug into without much difficulty. And for yet another, Amazon has become so large that it can negotiate discounts with international organizations, including significant shipping cost discounts. Simply speaking, other online marketplaces do not compete at this level. Amazon's online marketplace is exceptional.

Amazon's success notwithstanding, sellers on Amazon still need to manage, pack, and ship their own products, and provide their own customer service, right? Wrong. One of Amazon's innovations is Amazon fulfillment; Amazon handles the inventory, orders, shipping, returns, and customer service on behalf of the seller. To plug in to the marketplace, the seller only needs deliver the products to one of Amazon's warehouses. Amazon takes over from there. Essentially, with Amazon's marketplace, the roles change – sellers become suppliers, and Amazon becomes the seller. This process looks as follows.



The process makes room for other innovations, perhaps the most effective being rapid and free shipping for Amazon Prime buyers, for any of the seller's products. Individual sellers do not have the clout to make this possible by negotiating shipping discounts and arranging special shipping processes with national and international shipping companies, and they do not need to, because Amazon does so on their behalf. Buyers prize this option, making the seller's products more attractive. Why wait a week or more to receive what you have purchased when you can receive it in two days? And this is made even better by the fact that you can purchase virtually any kind of product.

The relationship between Amazon and sellers is synergistic. Amazon cannot produce the wide variety of products created by sellers worldwide yet has a superior online marketplace to sell these products effectively. Sellers cannot individually provide such an effective online marketplace but can provide the products. Both benefit and profit from this relationship.

Use Cases and Fields

New Product Use Case – This occurs when a seller plans to sell a product it has not sold before.

1.The seller searches Amazon's product list to determine if another seller is already selling the product.

2. If a different seller is already selling the product, a new listing is not required; the seller re-uses the same listing.

3.If the product is not yet sold on Amazon, a new listing is created with the product’s name, description, price, and other relevant items. Every product added is linked to a product category (all categories are predefined by Amazon), for example, “Computers”, “Electronics”, “Appliances”, and similar.

Field	Description
ProductID	This field stores the product ID, which is unique for each product. And field is used to distinguish between each product in case some product may have the same name.
ProductName	This field stores the name of the product. And it is necessary for displaying on the website for customer to search.
ProductPrice	This Field stores the price of the product. And it is necessary for customer to know how much they will spend.
ProductDescription	This field stores the description for the product. It is useful for customer to know detail information for the product.
ProductCategory	This field stores the category for the product. It is useful for customers to do categorical search and make products easier to manage.

Product Delivery Use Case – This occurs when a seller sends one or more units of a product to Amazon s o that they can be sold.
1.The seller ships one or more units of a product to Amazon’s warehouse, along with information that indicates to Amazon what the product is, how many units there are, and the condition (new, used, etc. …).
2.After Amazon receives the product(s), it updates the seller’s inventory so that customers can purchase the product.

Field	Description
ProductID	This field stores the product ID, which is unique for each product. And field is used to distinguish between each product in case some product may have the same name. And in this case, it is necessary for the warehouse to know which product did the seller shipped.
WarehouseID	This field stores the warehouse ID, and it is unique identification for each warehouse. This ID can be used to find the warehouse’s name, address, etc.
DeliveryQuantity	This Field stores the quantity of the product. And it is necessary for warehouse to know how many products the seller shipped and how to modify the inventory number.
Inventory	This field stores the inventory for a product. And it is necessary for seller and customer to know how many products are left in stock.
ProductCondition	This field stores the condition for the product. And it is useful for customer and warehouse to know the current condition for the product.

New Customer Account Use Case – This occurs when a customer signs up for an account on Amazon, so they can begin purchasing products.
1.The customer provides Amazon with basic information including a username, an address, phone number, and an email address.
2.Amazon creates an account for the customer, enabling the customer to purchase products.

Field	Description
User_id	This field stores the customer user_id. And it is the unique identification for each customer. It is necessary for Amazon to distinguish between different users.
Username	This field stores the customer username. And it is unique for each customer. It is necessary for Amazon to distinguish between different users.
CustomerAddress	This field stores the address for customer, and it is necessary for warehouse to know where to ship the product when customer place and order.
CustomerPhone	This Field stores the phone number for a customer, and it is useful for Amazon to contact the customer if some problems happened with their order.
CustomerEmail	This field stores the email address for a customer. Same as above, it is useful for Amazon to contact the customer.
CustomerFirstName	This field stores the first name for customer, it is useful for Amazon to know how to call their customer when contact them.
CustomerLastName	This field stores the last name for customer, it is useful for Amazon to know how to call their customer when contact them.

Product Purchase Use Case – This occurs when a customer purchases a product from Amazon that was provided by a seller.
1. The user logs in to Amazon under their account.
2.A customer selects one or more products on Amazon's website. When selecting a product, the customer is actually selecting a particular seller's inventory while doing so, though they might not realize this because the process is seamless on Amazon's website.
3. The customer selects a shipping speed (super saver shipping, standard shipping, two-day, one-day) and finalizes their choices.
4. Amazon decrements the seller's inventory for the products purchased.
5. Amazon creates an order which tracks which customer purchased which products from which sellers.

Field	Description
ProductID	This field stores the product ID, which is unique for each product. And field is used to distinguish between each product in case some product may have the same name. And in this case, it is necessary for the warehouse to know which product did the customer selected.
SellerID	This field stores the seller ID which is the unique identification for seller. This is necessary information on an product order to know who is the seller for this product.

Username	This Field stores the username of customer, which is the unique identification for a customer. And this information is necessary on a product order to know who the customer for this product is.
Inventory	This field stores the inventory for a product. And it is necessary for seller and customer to know how many products are left in stock. In this case, inventory for the product needs to be reduced by the amount customer purchased.
PurchaseQuantity	This field stores how many same product did the customer purchased.
ShipMethod	This field stores the shipment speed for the order. And it is necessary for the warehouse to know which shipment method they should use to ship the order.

Product Shipment Use Case – This occurs when Amazon ships the products a customer purchased.
1.Amazon packages up the purchased products, and assigns an identifier to package so that it can be tracked.
2. Amazon links the package to the customer's order.
3. Amazon ships the package to the default address linked to the customer's account.
4.Amazon notifies the customer that it has been shipped and provides the customer with the tracking ID.

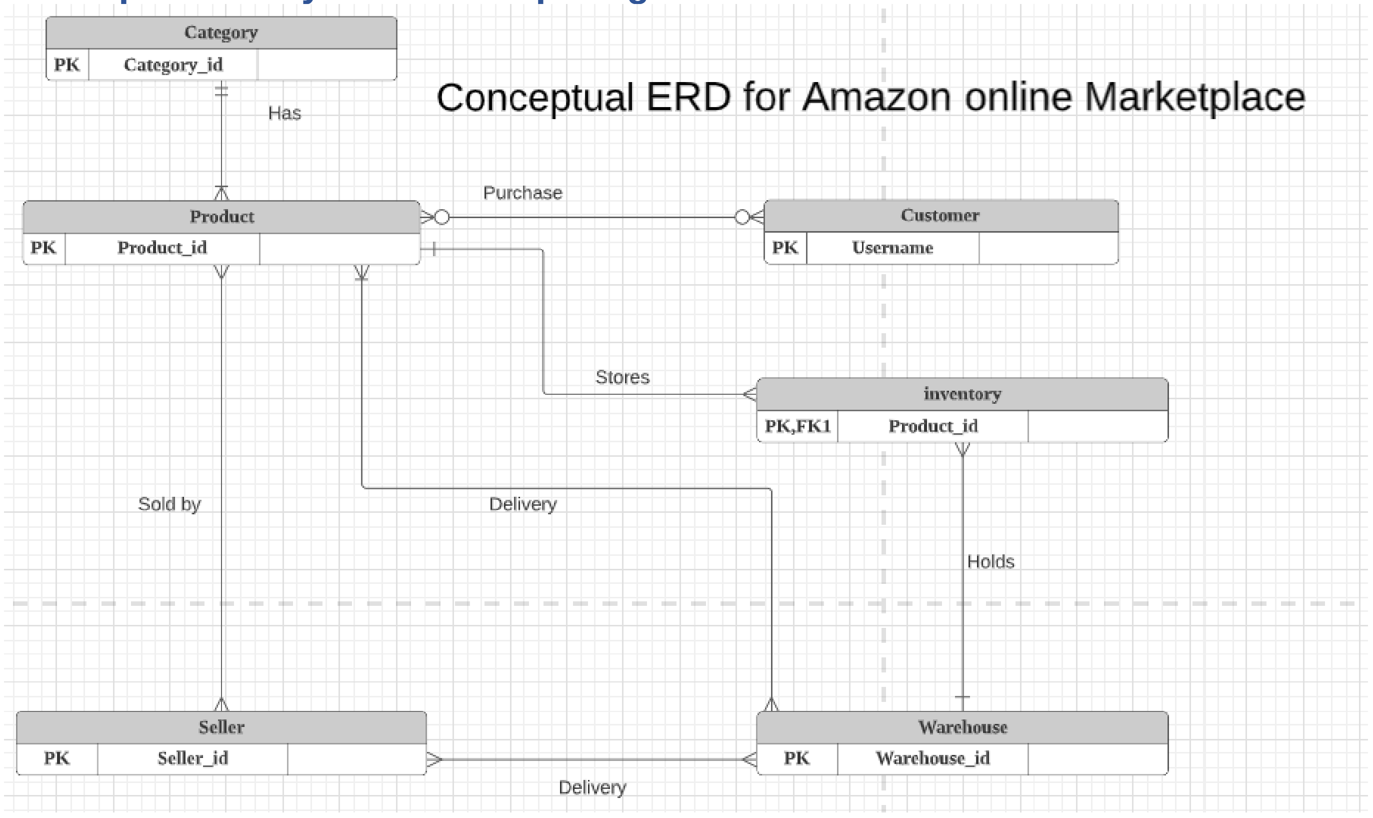
Field	Description
OrderID	This field stores the order ID, and it is the unique identification for each order. It is necessary for Amazon to distinguish between each order and to know which order the package should linked to.
Username	This field stores the username for a customer. And it is necessary in this case to help Amazon find the customer's address and contact information.
PackageID	This Field stores the package ID. It is the unique identification for each package. And it is necessary for amazon to distinguish between each package and manage them.
TrackingID	This field stores the tracking ID for the package. And it is useful for customer to know the updates for their package.

Structural Database Rules

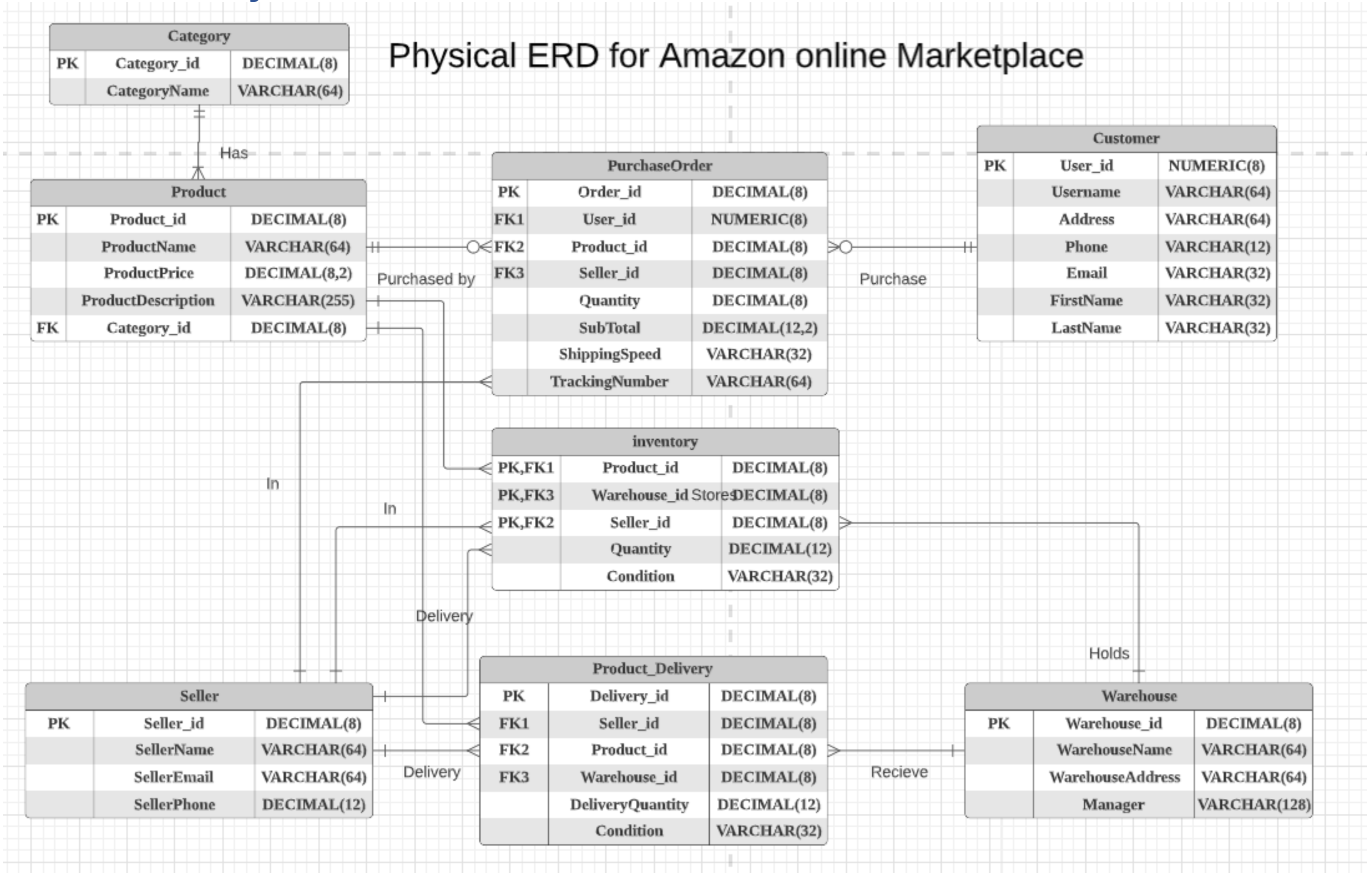
- Each Product should have one and only one category.
- Each category can have many products.
- Each Seller can sell many Products
- Each product can be sold by many sellers.
- Each seller can delivery many products to warehouse
- Each warehouse can receive products delivered by many sellers.
- Each customer can create zero or more orders
- Each order can only have one customer
- Each Product can appear in many orders

- Each order can only have one product
- Each customer's username must be unique
- Each product's name must be unique

Conceptual Entity-Relationship Diagram



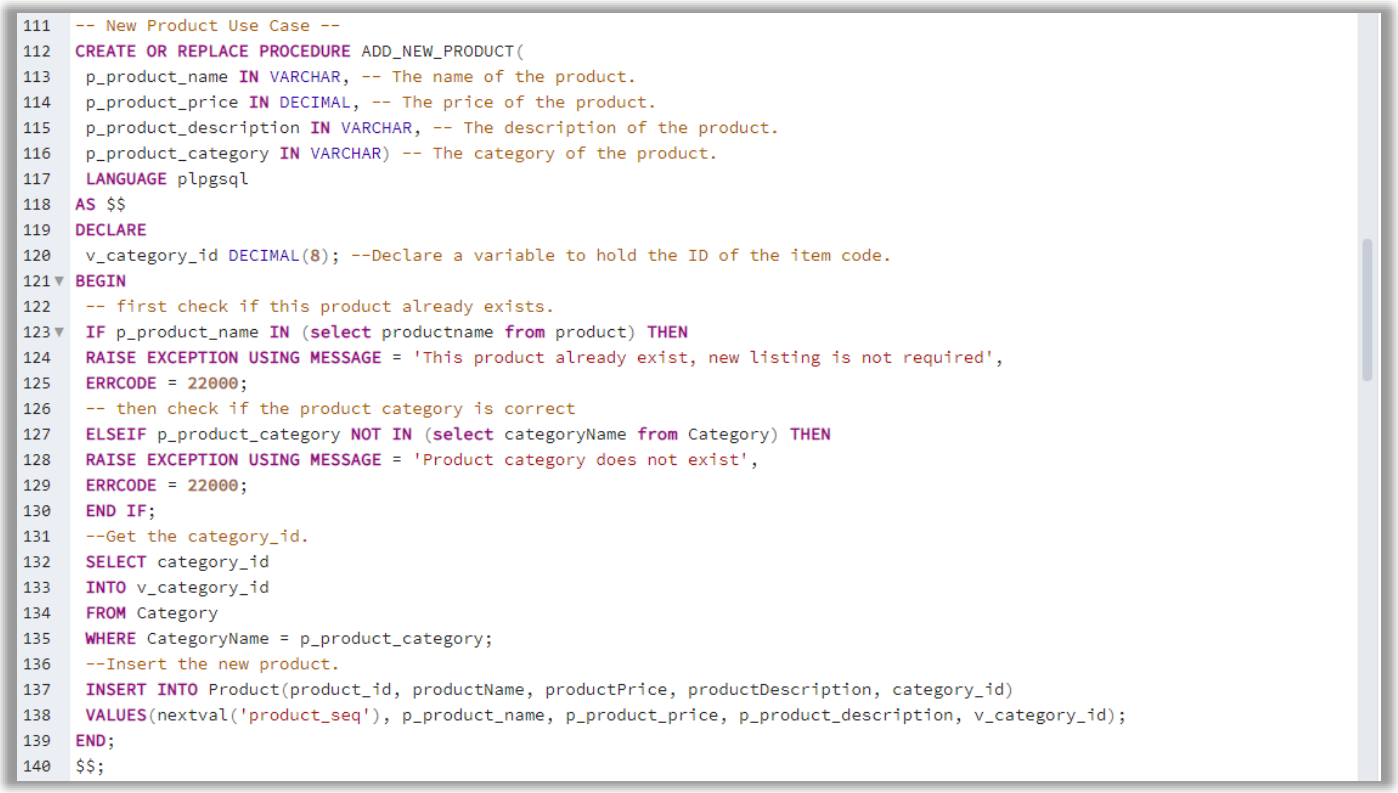
Full DBMS Physical ERD



Stored Procedure Execution and Explanations

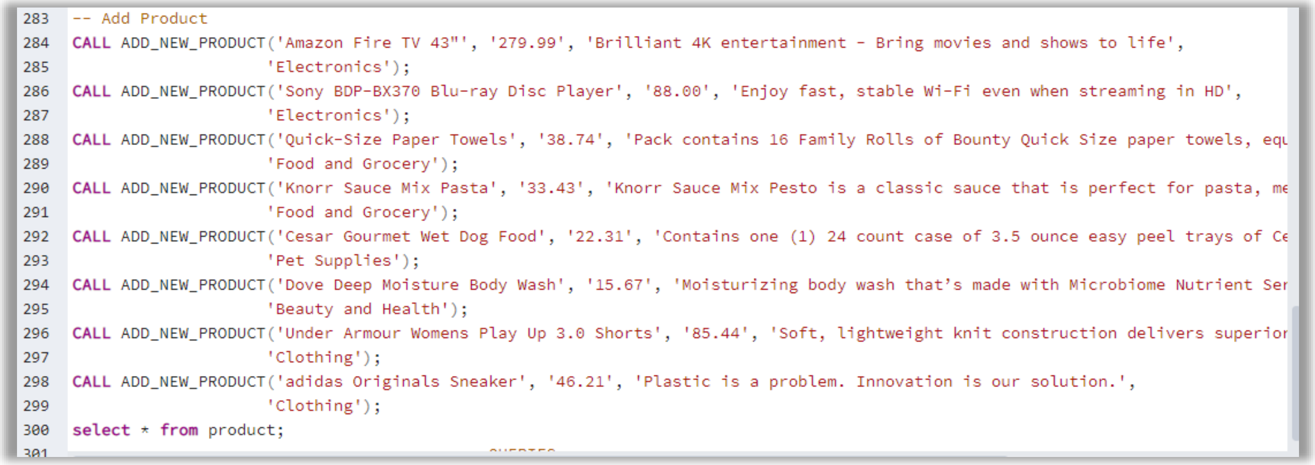
New Product Use case

The below stored procedure is used for adding a new product to Amazon's product list. Input parameters include product's name, price, description and category. If the product is already existed, then no need to add it one more time. Examples are shown below:

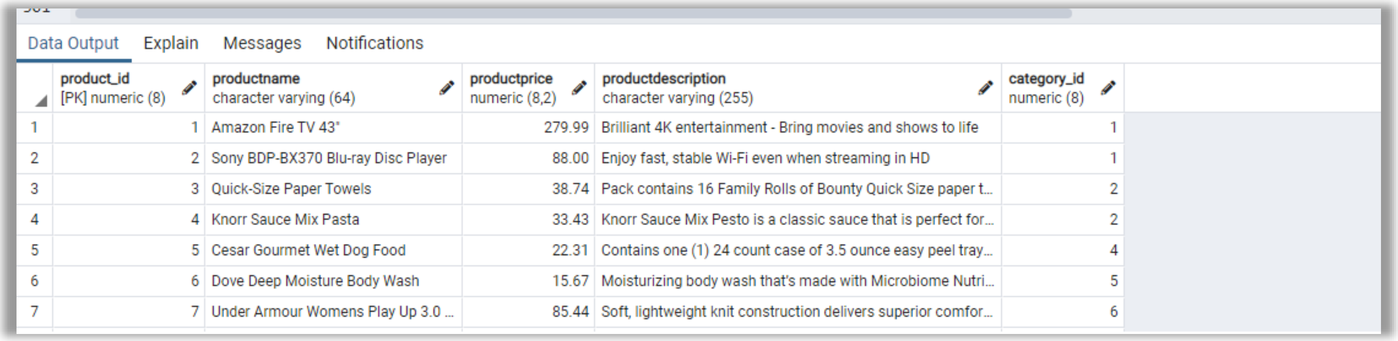


```sql
-- New Product Use Case --
CREATE OR REPLACE PROCEDURE ADD_NEW_PRODUCT(
 p_product_name IN VARCHAR, -- The name of the product.
 p_product_price IN DECIMAL, -- The price of the product.
 p_product_description IN VARCHAR, -- The description of the product.
 p_product_category IN VARCHAR) -- The category of the product.
 LANGUAGE plpgsql
AS $$
DECLARE
 v_category_id DECIMAL(8); --Declare a variable to hold the ID of the item code.
BEGIN
 -- first check if this product already exists.
 IF p_product_name IN (select productname from product) THEN
 RAISE EXCEPTION USING MESSAGE = 'This product already exist, new listing is not required',
 ERRCODE = 22000;
 -- then check if the product category is correct
 ELSEIF p_product_category NOT IN (select categoryName from Category) THEN
 RAISE EXCEPTION USING MESSAGE = 'Product category does not exist',
 ERRCODE = 22000;
 END IF;
 --Get the category_id.
 SELECT category_id
 INTO v_category_id
 FROM Category
 WHERE CategoryName = p_product_category;
 --Insert the new product.
 INSERT INTO Product(product_id, productName, productPrice, productDescription, category_id)
 VALUES(nextval('product_seq'), p_product_name, p_product_price, p_product_description, v_category_id);
END;
$$;
```

As shown below, by using the ADD NEW PRODUCT procedure, 7 new product is added to the product table.



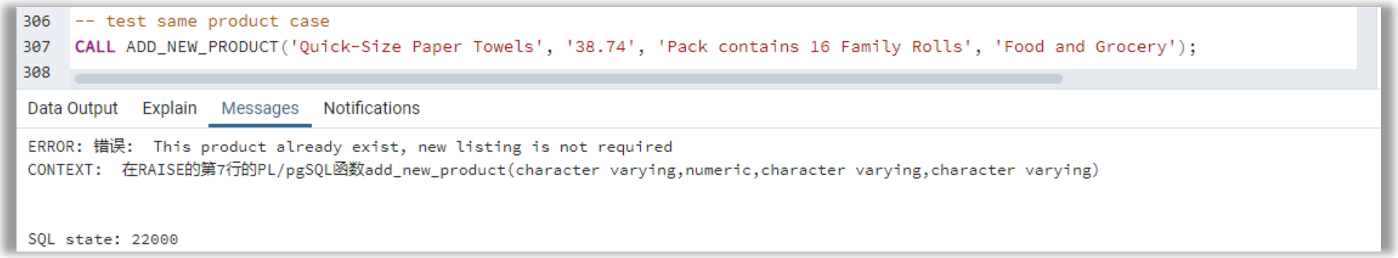
```sql
-- Add Product
CALL ADD_NEW_PRODUCT('Amazon Fire TV 43"', '279.99', 'Brilliant 4K entertainment - Bring movies and shows to life',
                    'Electronics');
CALL ADD_NEW_PRODUCT('Sony BDP-BX370 Blu-ray Disc Player', '88.00', 'Enjoy fast, stable Wi-Fi even when streaming in HD',
                    'Electronics');
CALL ADD_NEW_PRODUCT('Quick-Size Paper Towels', '38.74', 'Pack contains 16 Family Rolls of Bounty Quick Size paper towels, equ
                    'Food and Grocery');
CALL ADD_NEW_PRODUCT('Knorr Sauce Mix Pasta', '33.43', 'Knorr Sauce Mix Pesto is a classic sauce that is perfect for pasta, me
                    'Food and Grocery');
CALL ADD_NEW_PRODUCT('Cesar Gourmet Wet Dog Food', '22.31', 'Contains one (1) 24 count case of 3.5 ounce easy peel trays of Ce
                    'Pet Supplies');
CALL ADD_NEW_PRODUCT('Dove Deep Moisture Body Wash', '15.67', 'Moisturizing body wash that's made with Microbiome Nutrient Ser
                    'Beauty and Health');
CALL ADD_NEW_PRODUCT('Under Armour Womens Play Up 3.0 Shorts', '85.44', 'Soft, lightweight knit construction delivers superior
                    'Clothing');
CALL ADD_NEW_PRODUCT('adidas Originals Sneaker', '46.21', 'Plastic is a problem. Innovation is our solution.',
                    'Clothing');
select * from product;
```



Data Output Explain Messages Notifications

	product_id [PK] numeric (8)	productname character varying (64)	productprice numeric (8,2)	productdescription character varying (255)	category_id numeric (8)
1	1	Amazon Fire TV 43"	279.99	Brilliant 4K entertainment - Bring movies and shows to life	1
2	2	Sony BDP-BX370 Blu-ray Disc Player	88.00	Enjoy fast, stable Wi-Fi even when streaming in HD	1
3	3	Quick-Size Paper Towels	38.74	Pack contains 16 Family Rolls of Bounty Quick Size paper t...	2
4	4	Knorr Sauce Mix Pasta	33.43	Knorr Sauce Mix Pesto is a classic sauce that is perfect for...	2
5	5	Cesar Gourmet Wet Dog Food	22.31	Contains one (1) 24 count case of 3.5 ounce easy peel tray...	4
6	6	Dove Deep Moisture Body Wash	15.67	Moisturizing body wash that's made with Microbiome Nutri...	5
7	7	Under Armour Womens Play Up 3.0 ...	85.44	Soft, lightweight knit construction delivers superior comfor...	6

Then the below code tests the situation when same product is added. An exception will be raised, and an error message will be produced saying the product is already existing.

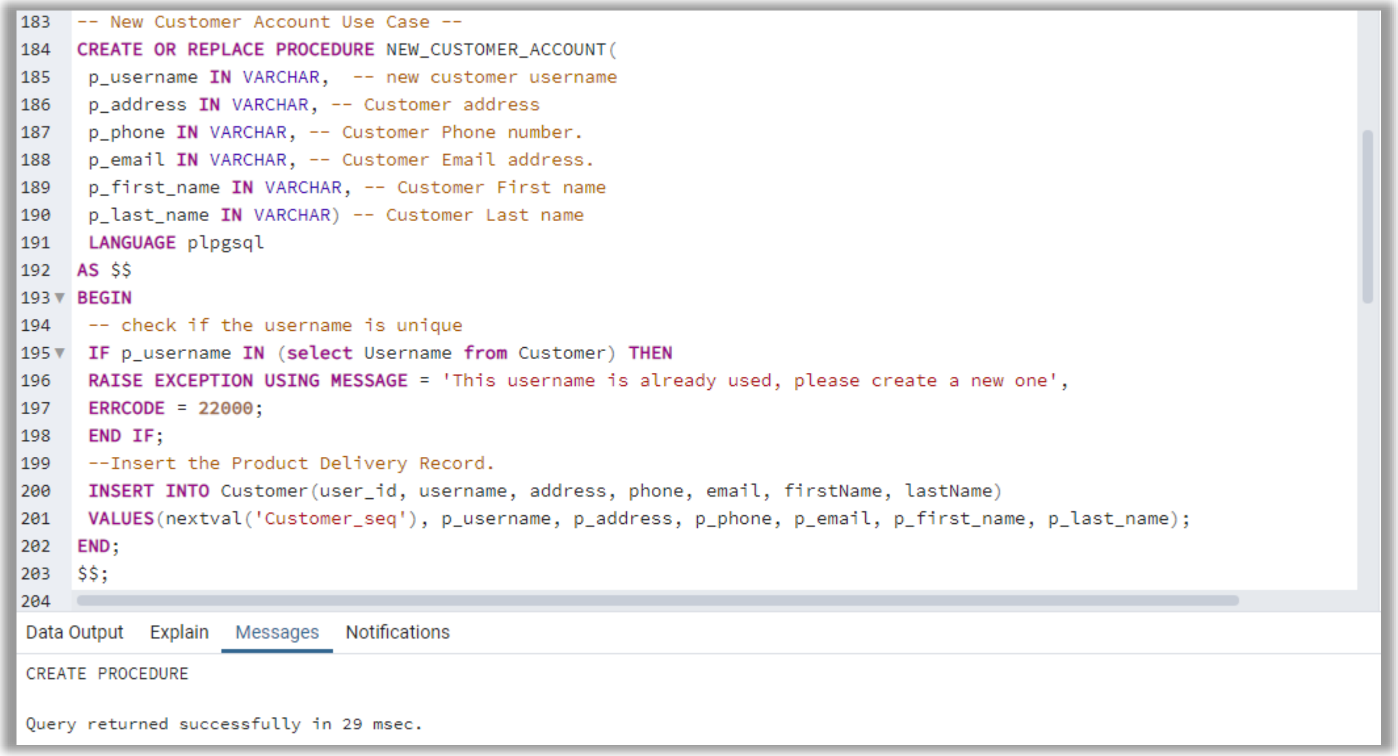

```
-- test same product case
CALL ADD_NEW_PRODUCT('Quick-Size Paper Towels', '38.74', 'Pack contains 16 Family Rolls', 'Food and Grocery');
```

```
ERROR: 错误:  This product already exist, new listing is not required
CONTEXT:  在RAISE的第7行的PL/pgSQL函数add_new_product(character varying,numeric,character varying,character varying)


SQL state: 22000
```

New Customer Use Case

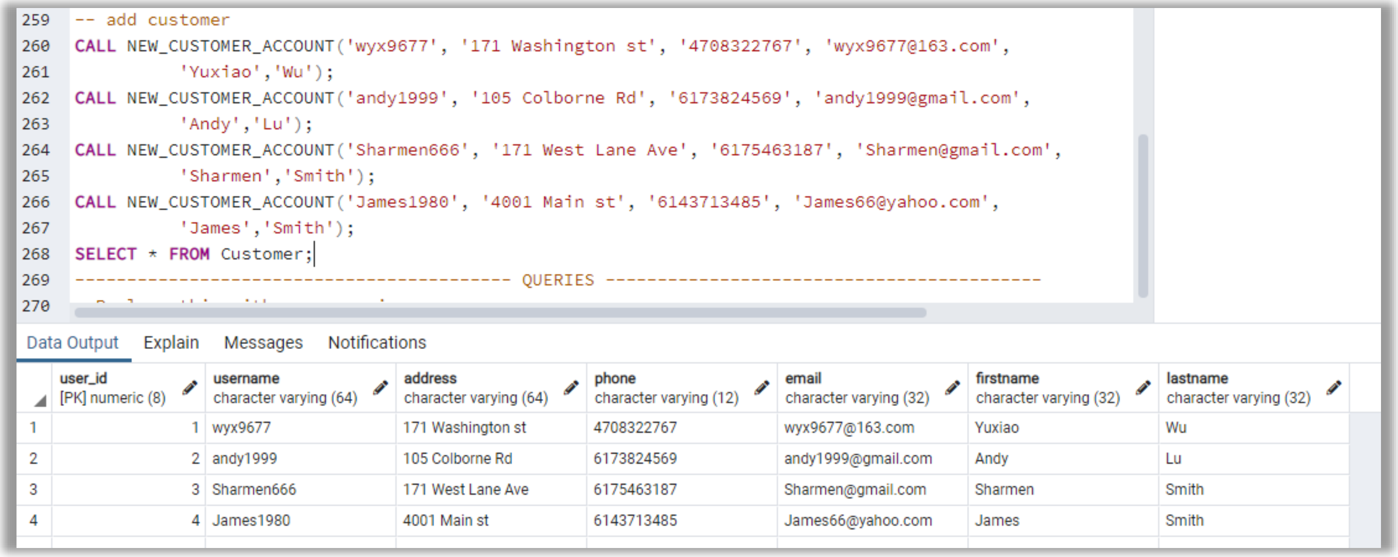
Below is the stored procedure for adding a new customer account use case. By using this procedure, amazon can add new customers into their system. And the customer's username should be unique. Customer cannot choose a username that is already been taken. Information includes customer's username, address, phone number, email, their first name and last name. all those will be stored into customer table.


```
-- New Customer Account Use Case --
CREATE OR REPLACE PROCEDURE NEW_CUSTOMER_ACCOUNT(
 p_username IN VARCHAR,  -- new customer username
 p_address IN VARCHAR, -- Customer address
 p_phone IN VARCHAR, -- Customer Phone number.
 p_email IN VARCHAR, -- Customer Email address.
 p_first_name IN VARCHAR, -- Customer First name
 p_last_name IN VARCHAR) -- Customer Last name
 LANGUAGE plpgsql
AS $$
BEGIN
 -- check if the username is unique
 IF p_username IN (select Username from Customer) THEN
 RAISE EXCEPTION USING MESSAGE = 'This username is already used, please create a new one',
 ERRCODE = 22000;
 END IF;
 --Insert the Product Delivery Record.
 INSERT INTO Customer(user_id, username, address, phone, email, firstName, lastName)
 VALUES(nextval('Customer_seq'), p_username, p_address, p_phone, p_email, p_first_name, p_last_name);
END;
$$;
```

```
CREATE PROCEDURE

Query returned successfully in 29 msec.
```

As shown below, by Calling the new customer account stored procedure, four different customers were successful added to the database.

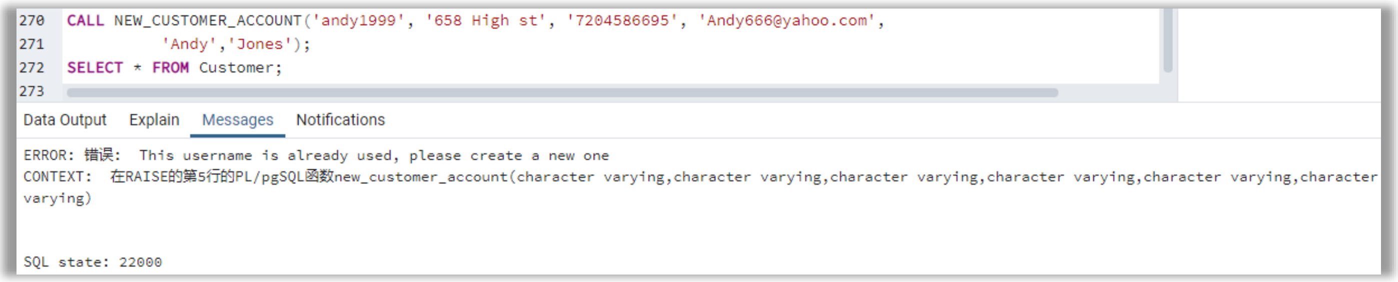


```sql
-- add customer
CALL NEW_CUSTOMER_ACCOUNT('wyx9677', '171 Washington st', '4708322767', 'wyx9677@163.com',
          'Yuxiao','Wu');
CALL NEW_CUSTOMER_ACCOUNT('andy1999', '105 Colborne Rd', '6173824569', 'andy1999@gmail.com',
          'Andy','Lu');
CALL NEW_CUSTOMER_ACCOUNT('Sharmen666', '171 West Lane Ave', '6175463187', 'Sharmen@gmail.com',
          'Sharmen','Smith');
CALL NEW_CUSTOMER_ACCOUNT('James1980', '4001 Main st', '6143713485', 'James66@yahoo.com',
          'James','Smith');
SELECT * FROM Customer;
------------------------------------------ QUERIES ------------------------------------------
```

Data Output | Explain | Messages | Notifications

	user_id [PK] numeric (8)	username character varying (64)	address character varying (64)	phone character varying (12)	email character varying (32)	firstname character varying (32)	lastname character varying (32)
1	1	wyx9677	171 Washington st	4708322767	wyx9677@163.com	Yuxiao	Wu
2	2	andy1999	105 Colborne Rd	6173824569	andy1999@gmail.com	Andy	Lu
3	3	Sharmen666	171 West Lane Ave	6175463187	Sharmen@gmail.com	Sharmen	Smith
4	4	James1980	4001 Main st	6143713485	James66@yahoo.com	James	Smith

Below shows the situation when a customer wants to create an account that has the same username as someone else. Which is not allowed. The procedure will raise an exception and error message that ask the user to choose another username. The result is shown below.

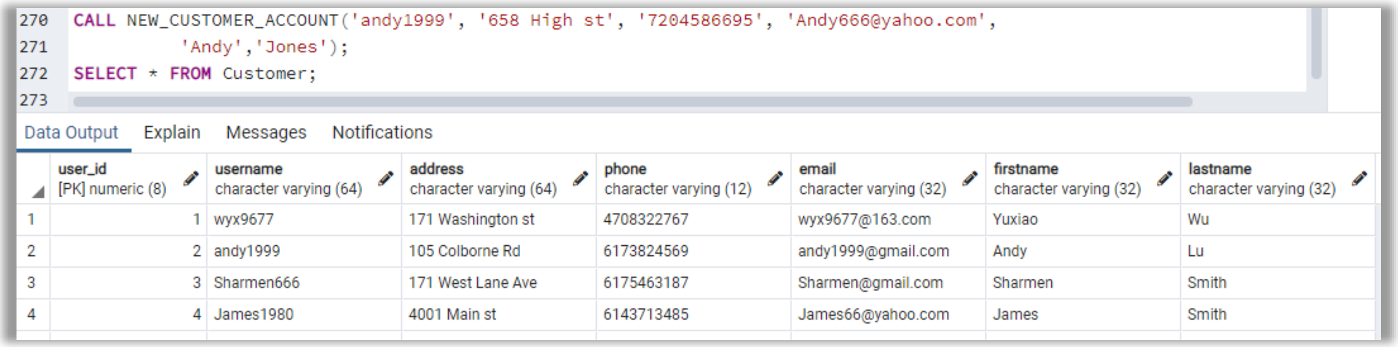


```sql
CALL NEW_CUSTOMER_ACCOUNT('andy1999', '658 High st', '7204586695', 'Andy666@yahoo.com',
          'Andy','Jones');
SELECT * FROM Customer;
```

Data Output | Explain | Messages | Notifications

```
ERROR: 错误:  This username is already used, please create a new one
CONTEXT:  在RAISE的第5行的PL/pgSQL函数new_customer_account(character varying,character varying,character varying,character varying,character varying,character
varying)


SQL state: 22000
```



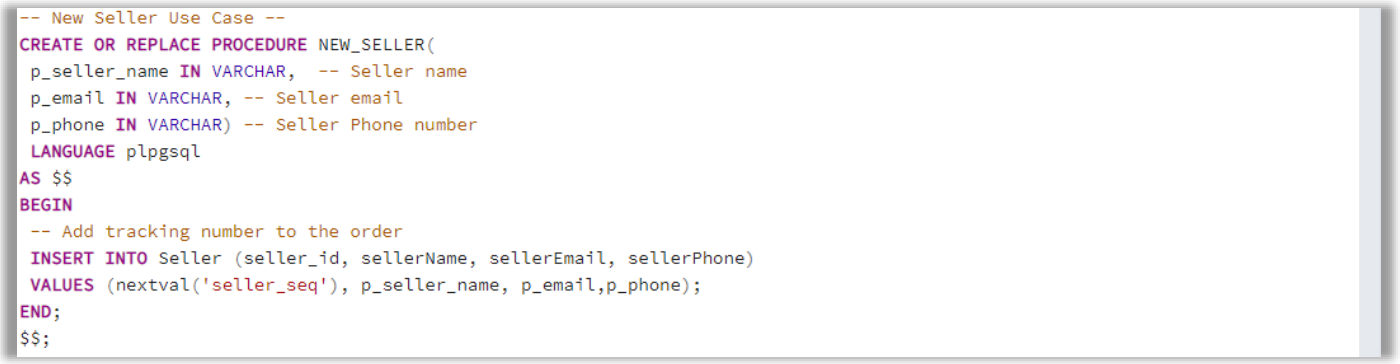
```sql
CALL NEW_CUSTOMER_ACCOUNT('andy1999', '658 High st', '7204586695', 'Andy666@yahoo.com',
          'Andy','Jones');
SELECT * FROM Customer;
```

Data Output | Explain | Messages | Notifications

	user_id [PK] numeric (8)	username character varying (64)	address character varying (64)	phone character varying (12)	email character varying (32)	firstname character varying (32)	lastname character varying (32)
1	1	wyx9677	171 Washington st	4708322767	wyx9677@163.com	Yuxiao	Wu
2	2	andy1999	105 Colborne Rd	6173824569	andy1999@gmail.com	Andy	Lu
3	3	Sharmen666	171 West Lane Ave	6175463187	Sharmen@gmail.com	Sharmen	Smith
4	4	James1980	4001 Main st	6143713485	James66@yahoo.com	James	Smith

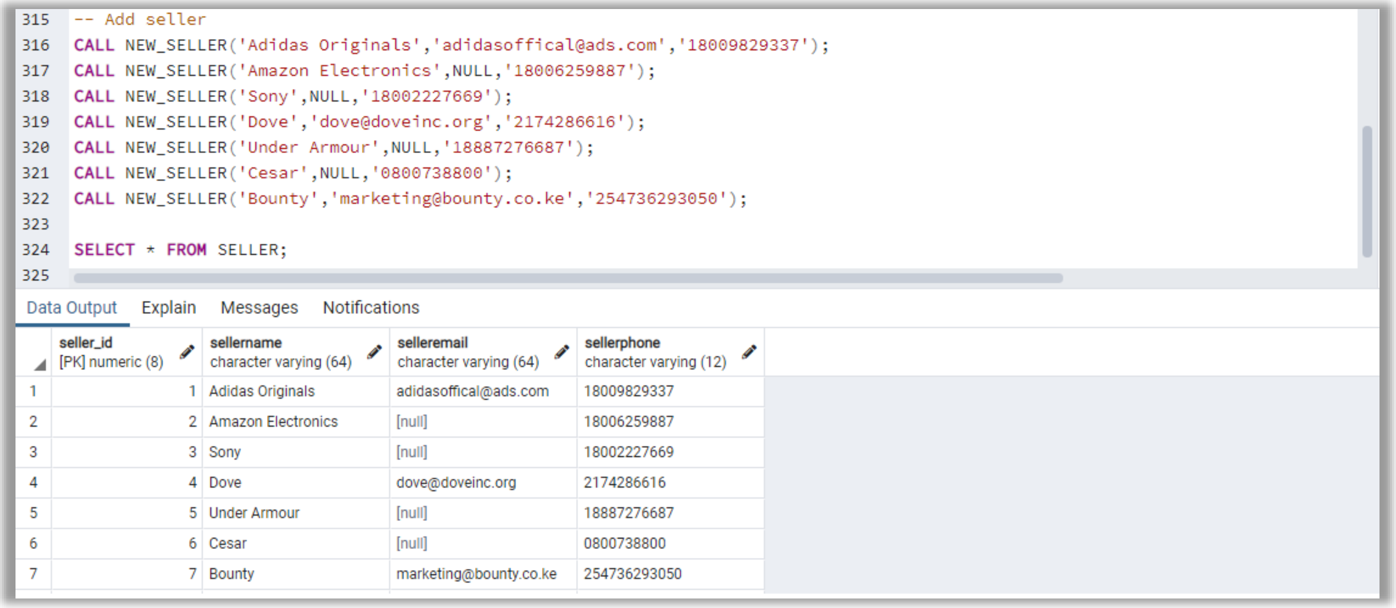
Add New Seller

The below procedure is used for adding new seller to the database. As shown below, parameters needed are seller name, email, and phone number



```
-- New Seller Use Case --
CREATE OR REPLACE PROCEDURE NEW_SELLER(
 p_seller_name IN VARCHAR,  -- Seller name
 p_email IN VARCHAR, -- Seller email
 p_phone IN VARCHAR) -- Seller Phone number
 LANGUAGE plpgsql
AS $$
BEGIN
 -- Add tracking number to the order
 INSERT INTO Seller (seller_id, sellerName, sellerEmail, sellerPhone)
 VALUES (nextval('seller_seq'), p_seller_name, p_email,p_phone);
END;
$$;
```

Below code shows how to use NEW SELLER procedure to add seller to database. And the result is shown below.



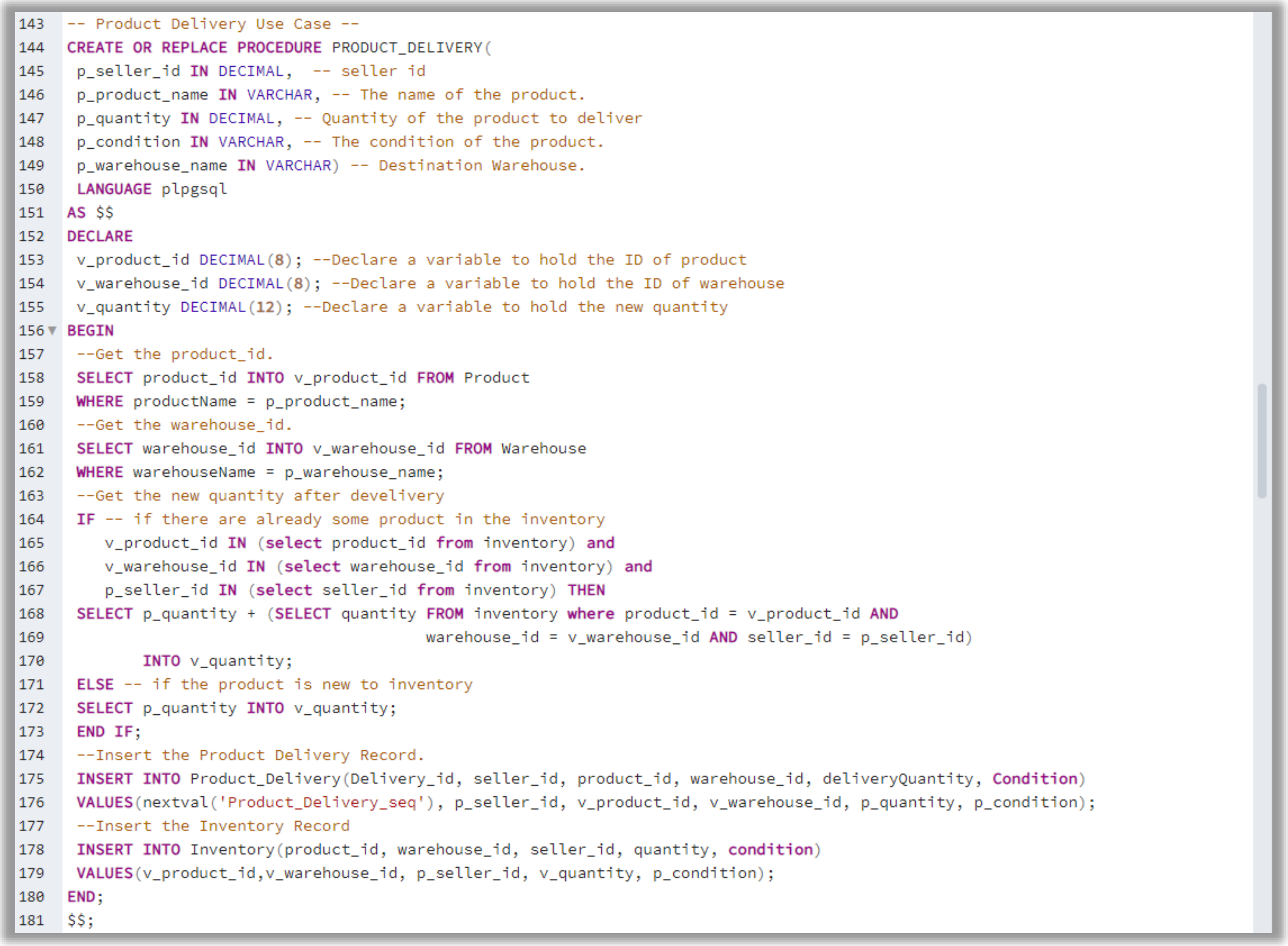
```
-- Add seller
CALL NEW_SELLER('Adidas Originals','adidasoffical@ads.com','18009829337');
CALL NEW_SELLER('Amazon Electronics',NULL,'18006259887');
CALL NEW_SELLER('Sony',NULL,'18002227669');
CALL NEW_SELLER('Dove','dove@doveinc.org','2174286616');
CALL NEW_SELLER('Under Armour',NULL,'18887276687');
CALL NEW_SELLER('Cesar',NULL,'0800738800');
CALL NEW_SELLER('Bounty','marketing@bounty.co.ke','254736293050');

SELECT * FROM SELLER;
```

	seller_id [PK] numeric (8)	sellername character varying (64)	selleremail character varying (64)	sellerphone character varying (12)
1	1	Adidas Originals	adidasoffical@ads.com	18009829337
2	2	Amazon Electronics	[null]	18006259887
3	3	Sony	[null]	18002227669
4	4	Dove	dove@doveinc.org	2174286616
5	5	Under Armour	[null]	18887276687
6	6	Cesar	[null]	0800738800
7	7	Bounty	marketing@bounty.co.ke	254736293050

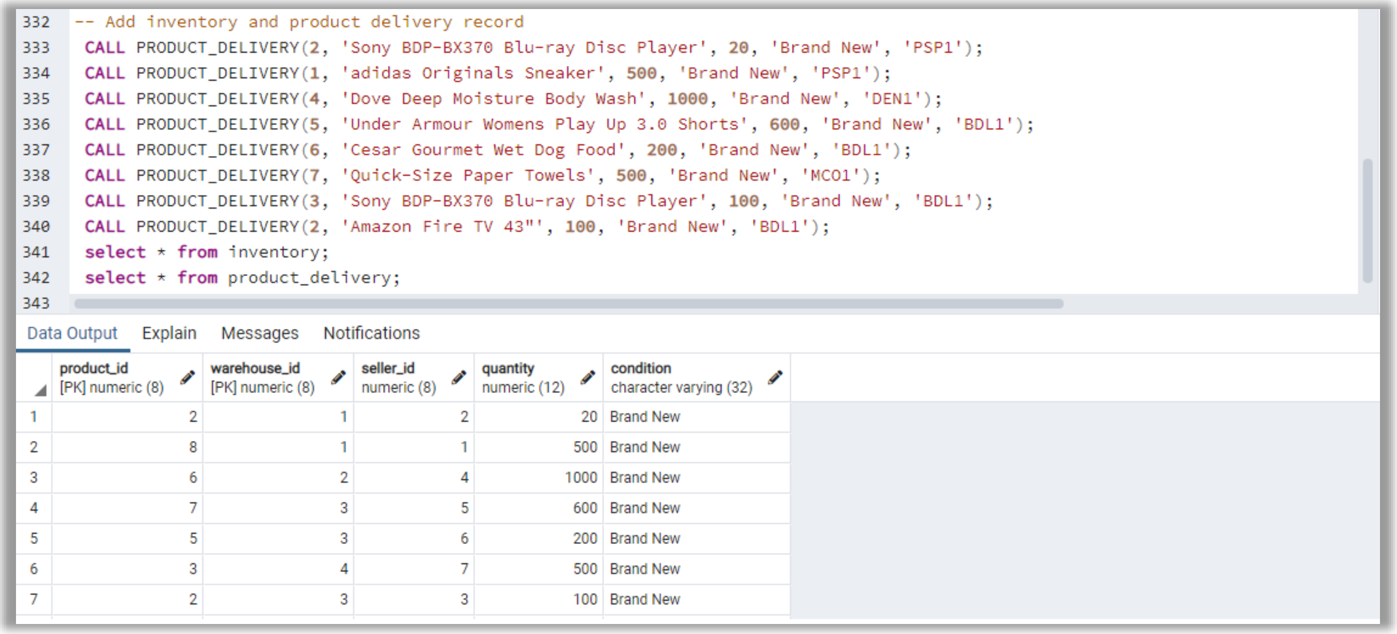
Product Delivery Use case

Below code shows the stored procedure used for seller to deliver products to warehouse use case. Parameters for this procedure includes seller id, product name, quantity delivered, product condition, and the warehouse name. by using this procedure, it will add data to both product_delivery table and inventory table. Because when a seller delivers to warehouse, a record for this delivery will be created and the inventory for that product will also change.



```
-- Product Delivery Use Case --
CREATE OR REPLACE PROCEDURE PRODUCT_DELIVERY(
 p_seller_id IN DECIMAL,  -- seller id
 p_product_name IN VARCHAR, -- The name of the product.
 p_quantity IN DECIMAL, -- Quantity of the product to deliver
 p_condition IN VARCHAR, -- The condition of the product.
 p_warehouse_name IN VARCHAR) -- Destination Warehouse.
 LANGUAGE plpgsql
AS $$
DECLARE
 v_product_id DECIMAL(8); --Declare a variable to hold the ID of product
 v_warehouse_id DECIMAL(8); --Declare a variable to hold the ID of warehouse
 v_quantity DECIMAL(12); --Declare a variable to hold the new quantity
BEGIN
 --Get the product_id.
 SELECT product_id INTO v_product_id FROM Product
 WHERE productName = p_product_name;
 --Get the warehouse_id.
 SELECT warehouse_id INTO v_warehouse_id FROM Warehouse
 WHERE warehouseName = p_warehouse_name;
 --Get the new quantity after develivery
 IF -- if there are already some product in the inventory
    v_product_id IN (select product_id from inventory) and
    v_warehouse_id IN (select warehouse_id from inventory) and
    p_seller_id IN (select seller_id from inventory) THEN
 SELECT p_quantity + (SELECT quantity FROM inventory where product_id = v_product_id AND
                                      warehouse_id = v_warehouse_id AND seller_id = p_seller_id)
        INTO v_quantity;
 ELSE -- if the product is new to inventory
 SELECT p_quantity INTO v_quantity;
 END IF;
 --Insert the Product Delivery Record.
 INSERT INTO Product_Delivery(Delivery_id, seller_id, product_id, warehouse_id, deliveryQuantity, Condition)
 VALUES(nextval('Product_Delivery_seq'), p_seller_id, v_product_id, v_warehouse_id, p_quantity, p_condition);
 --Insert the Inventory Record
 INSERT INTO Inventory(product_id, warehouse_id, seller_id, quantity, condition)
 VALUES(v_product_id,v_warehouse_id, p_seller_id, v_quantity, p_condition);
END;
$$;
```

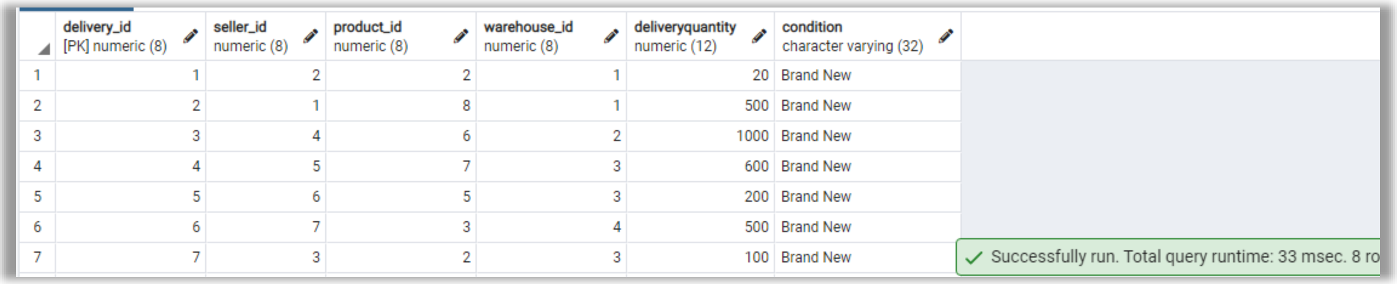
Below code shows examples for use PRODUCT DELIVERY procedure to insert values into product_delivery and inventory table. And the result table is shown below.



```
-- Add inventory and product delivery record
 CALL PRODUCT_DELIVERY(2, 'Sony BDP-BX370 Blu-ray Disc Player', 20, 'Brand New', 'PSP1');
 CALL PRODUCT_DELIVERY(1, 'adidas Originals Sneaker', 500, 'Brand New', 'PSP1');
 CALL PRODUCT_DELIVERY(4, 'Dove Deep Moisture Body Wash', 1000, 'Brand New', 'DEN1');
 CALL PRODUCT_DELIVERY(5, 'Under Armour Womens Play Up 3.0 Shorts', 600, 'Brand New', 'BDL1');
 CALL PRODUCT_DELIVERY(6, 'Cesar Gourmet Wet Dog Food', 200, 'Brand New', 'BDL1');
 CALL PRODUCT_DELIVERY(7, 'Quick-Size Paper Towels', 500, 'Brand New', 'MCO1');
 CALL PRODUCT_DELIVERY(3, 'Sony BDP-BX370 Blu-ray Disc Player', 100, 'Brand New', 'BDL1');
 CALL PRODUCT_DELIVERY(2, 'Amazon Fire TV 43"', 100, 'Brand New', 'BDL1');
 select * from inventory;
 select * from product_delivery;
```

Data Output Explain Messages Notifications

	product_id [PK] numeric (8)	warehouse_id [PK] numeric (8)	seller_id numeric (8)	quantity numeric (12)	condition character varying (32)
1	2	1	2	20	Brand New
2	8	1	1	500	Brand New
3	6	2	4	1000	Brand New
4	7	3	5	600	Brand New
5	5	3	6	200	Brand New
6	3	4	7	500	Brand New
7	2	3	3	100	Brand New

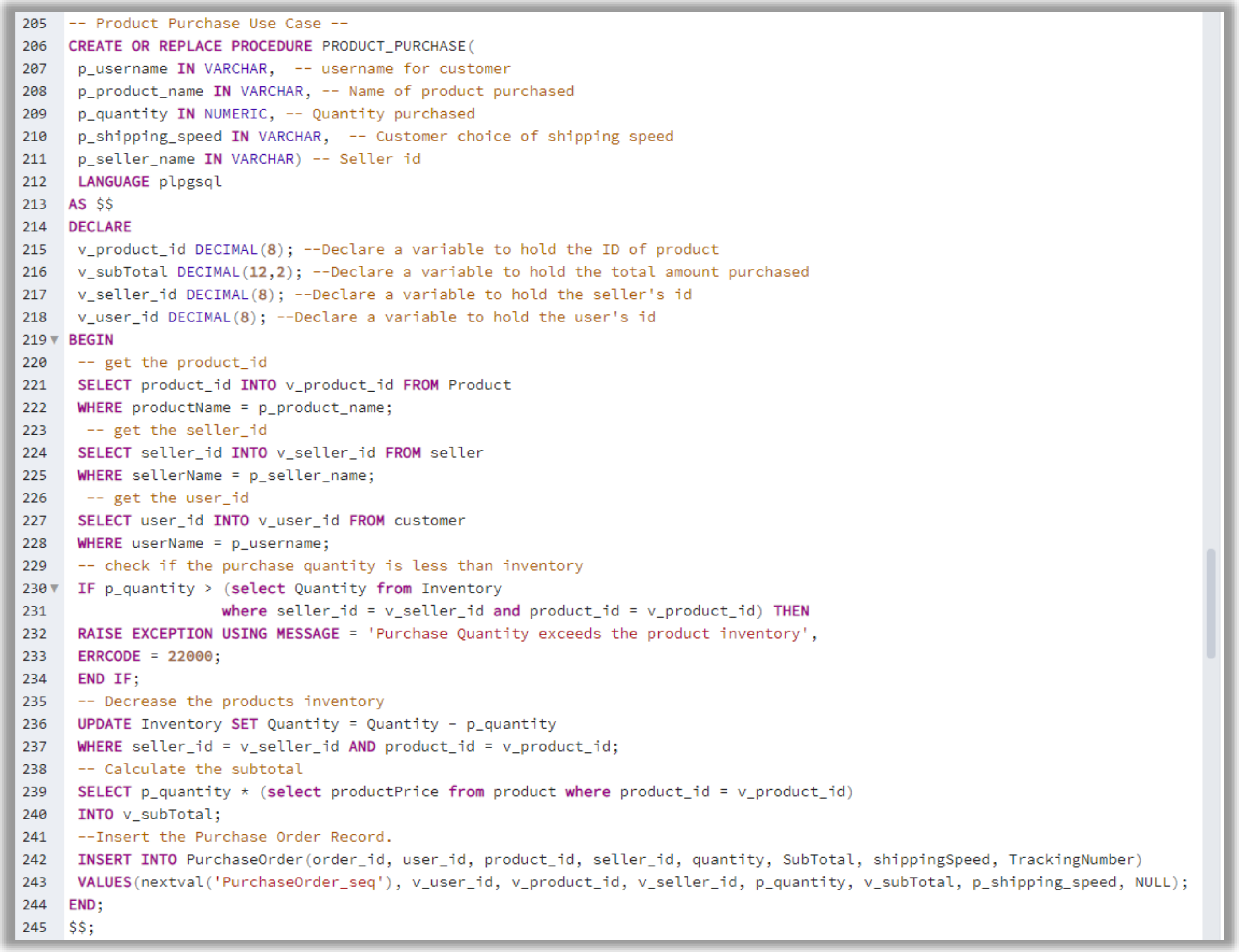


	delivery_id [PK] numeric (8)	seller_id numeric (8)	product_id numeric (8)	warehouse_id numeric (8)	deliveryquantity numeric (12)	condition character varying (32)
1	1	2	2	1	20	Brand New
2	2	1	8	1	500	Brand New
3	3	4	6	2	1000	Brand New
4	4	5	7	3	600	Brand New
5	5	6	5	3	200	Brand New
6	6	7	3	4	500	Brand New
7	7	3	2	3	100	Brand New

✓ Successfully run. Total query runtime: 33 msec. 8 ro

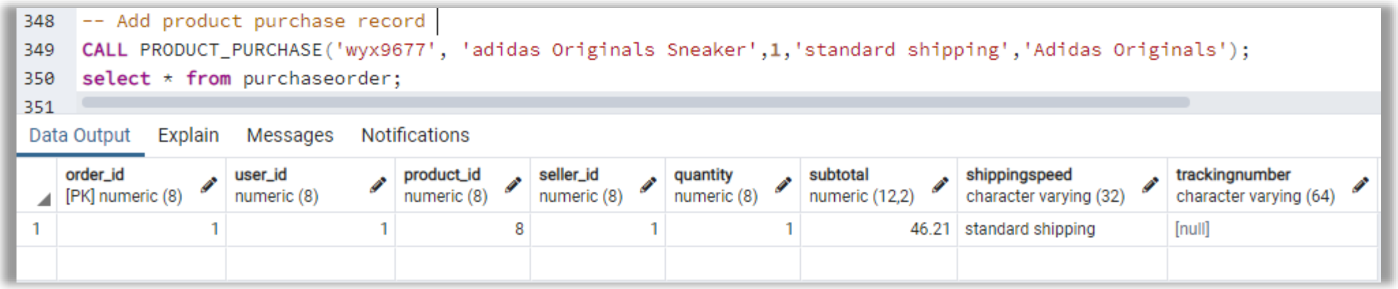
Product Purchase Use Case

Below is the code for product purchase use case procedure. When a customer places an order for a product, Amazon first check if the purchase quantity is less than the inventory, if yes, then decrease the product's inventory, and create a record in the purchase order table. Parameters needed for this procedure includes the customer's username, product name purchased, quantity purchased, choice of shipment speed and seller name.



```
-- Product Purchase Use Case --
CREATE OR REPLACE PROCEDURE PRODUCT_PURCHASE(
 p_username IN VARCHAR,  -- username for customer
 p_product_name IN VARCHAR, -- Name of product purchased
 p_quantity IN NUMERIC, -- Quantity purchased
 p_shipping_speed IN VARCHAR,  -- Customer choice of shipping speed
 p_seller_name IN VARCHAR) -- Seller id
 LANGUAGE plpgsql
AS $$
DECLARE
 v_product_id DECIMAL(8); --Declare a variable to hold the ID of product
 v_subTotal DECIMAL(12,2); --Declare a variable to hold the total amount purchased
 v_seller_id DECIMAL(8); --Declare a variable to hold the seller's id
 v_user_id DECIMAL(8); --Declare a variable to hold the user's id
BEGIN
 -- get the product_id
 SELECT product_id INTO v_product_id FROM Product
 WHERE productName = p_product_name;
  -- get the seller_id
 SELECT seller_id INTO v_seller_id FROM seller
 WHERE sellerName = p_seller_name;
  -- get the user_id
 SELECT user_id INTO v_user_id FROM customer
 WHERE userName = p_username;
 -- check if the purchase quantity is less than inventory
 IF p_quantity > (select Quantity from Inventory
                 where seller_id = v_seller_id and product_id = v_product_id) THEN
 RAISE EXCEPTION USING MESSAGE = 'Purchase Quantity exceeds the product inventory',
 ERRCODE = 22000;
 END IF;
 -- Decrease the products inventory
 UPDATE Inventory SET Quantity = Quantity - p_quantity
 WHERE seller_id = v_seller_id AND product_id = v_product_id;
 -- Calculate the subtotal
 SELECT p_quantity * (select productPrice from product where product_id = v_product_id)
 INTO v_subTotal;
 --Insert the Purchase Order Record.
 INSERT INTO PurchaseOrder(order_id, user_id, product_id, seller_id, quantity, SubTotal, shippingSpeed, TrackingNumber)
 VALUES(nextval('PurchaseOrder_seq'), v_user_id, v_product_id, v_seller_id, p_quantity, v_subTotal, p_shipping_speed, NULL);
END;
$$;
```

Below code is the example of using PRODUCT PURCHASE procedure to insert in to purchase order table. As we can see, user with username "wyx9677" purchased a adidas Originals Sneaker from adidas originals store, and choose the standard shipping speed. From the result we can see the purchase order is created and saved in the purchase order table. Also, you can notice that the tracking number is NULL because the order is not shipped yet, the tracking number will be created in the product shipment use case.

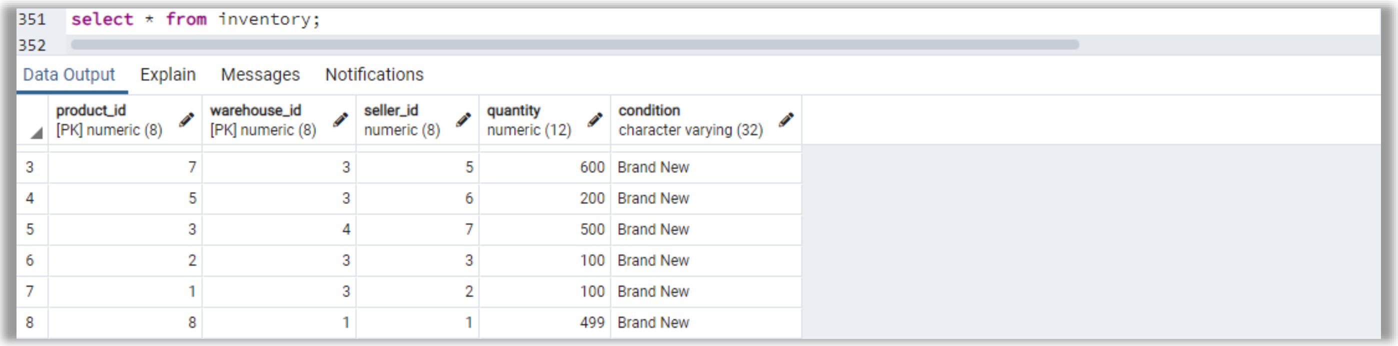


```
-- Add product purchase record
CALL PRODUCT_PURCHASE('wyx9677', 'adidas Originals Sneaker',1,'standard shipping','Adidas Originals');
select * from purchaseorder;
```

Data Output | Explain | Messages | Notifications

	order_id [PK] numeric (8)	user_id numeric (8)	product_id numeric (8)	seller_id numeric (8)	quantity numeric (8)	subtotal numeric (12,2)	shippingspeed character varying (32)	trackingnumber character varying (64)
1	1	1	8	1	1	46.21	standard shipping	[null]

Also, when a customer purchases an product, the inventory for that product should decrease by the purchase amount. As we can see from the below query, the inventory for adidas sneaker decreased by 1, from 500 to 499.

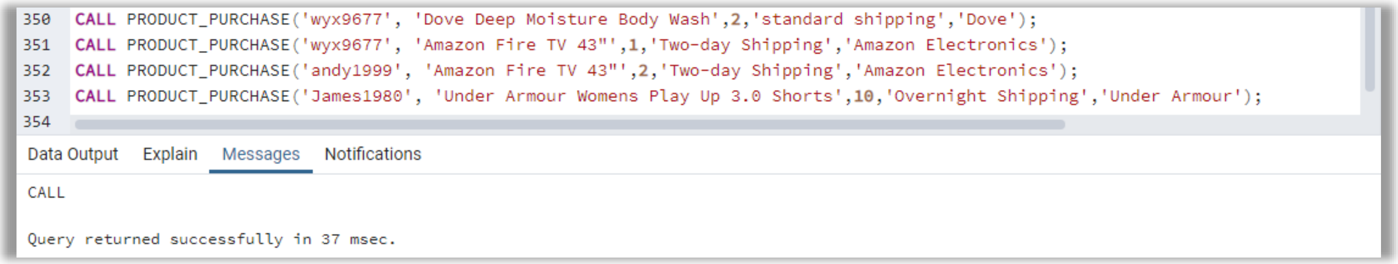


```
select * from inventory;
```

Data Output | Explain | Messages | Notifications

	product_id [PK] numeric (8)	warehouse_id [PK] numeric (8)	seller_id numeric (8)	quantity numeric (12)	condition character varying (32)
3	7	3	5	600	Brand New
4	5	3	6	200	Brand New
5	3	4	7	500	Brand New
6	2	3	3	100	Brand New
7	1	3	2	100	Brand New
8	8	1	1	499	Brand New

Below is the code for adding more purchase orders to the data base.



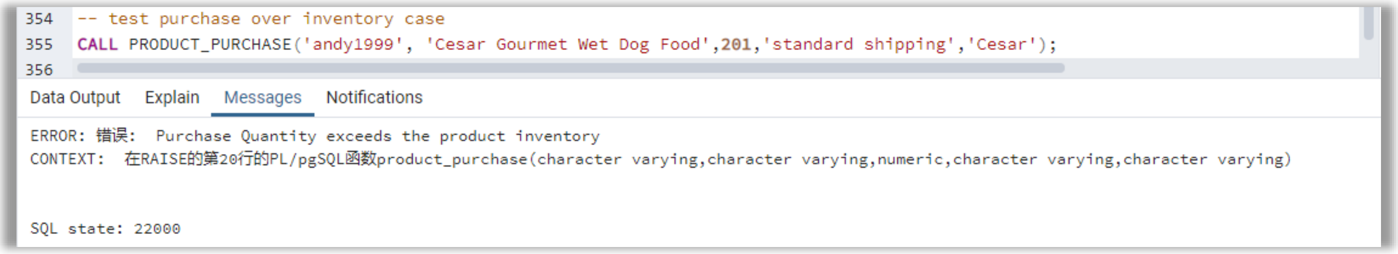
```
CALL PRODUCT_PURCHASE('wyx9677', 'Dove Deep Moisture Body Wash',2,'standard shipping','Dove');
CALL PRODUCT_PURCHASE('wyx9677', 'Amazon Fire TV 43"',1,'Two-day Shipping','Amazon Electronics');
CALL PRODUCT_PURCHASE('andy1999', 'Amazon Fire TV 43"',2,'Two-day Shipping','Amazon Electronics');
CALL PRODUCT_PURCHASE('James1980', 'Under Armour Womens Play Up 3.0 Shorts',10,'Overnight Shipping','Under Armour');
```

Data Output | Explain | Messages | Notifications

```
CALL

Query returned successfully in 37 msec.
```

Below is the situation when the purchase quantity is greater than the product's inventory. The procedure will raise an exception and through an error message saying the purchase quantity exceeds the product's inventory. No purchase order will be added, and the inventory will not be changed.



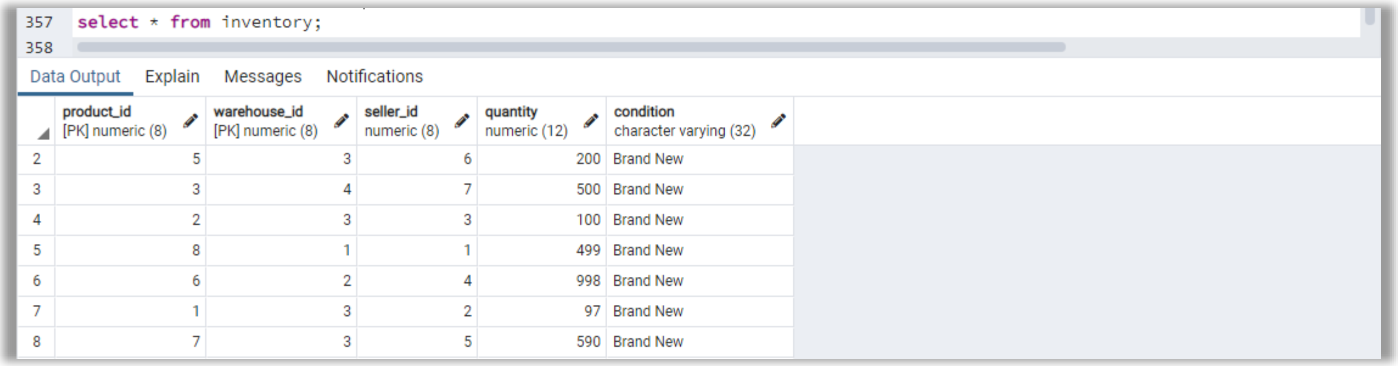
```
-- test purchase over inventory case
CALL PRODUCT_PURCHASE('andy1999', 'Cesar Gourmet Wet Dog Food',201,'standard shipping','Cesar');
```

Data Output | Explain | Messages | Notifications

```
ERROR:  错误:  Purchase Quantity exceeds the product inventory
CONTEXT:  在RAISE的第20行的PL/pgSQL函数product_purchase(character varying,character varying,numeric,character varying,character varying)


SQL state: 22000
```

Below table shows the inventory after those purchases.



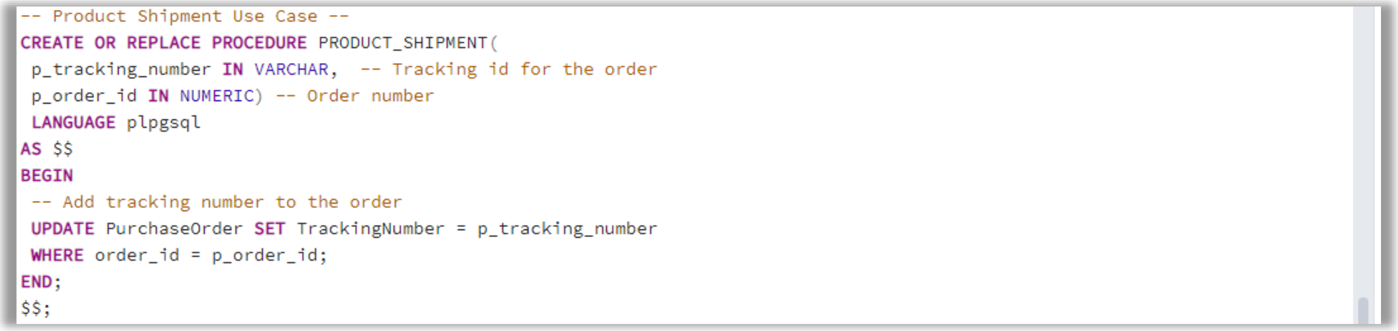
```sql
select * from inventory;
```

Data Output | Explain | Messages | Notifications

	product_id [PK] numeric (8)	warehouse_id [PK] numeric (8)	seller_id numeric (8)	quantity numeric (12)	condition character varying (32)
2	5	3	6	200	Brand New
3	3	4	7	500	Brand New
4	2	3	3	100	Brand New
5	8	1	1	499	Brand New
6	6	2	4	998	Brand New
7	1	3	2	97	Brand New
8	7	3	5	590	Brand New

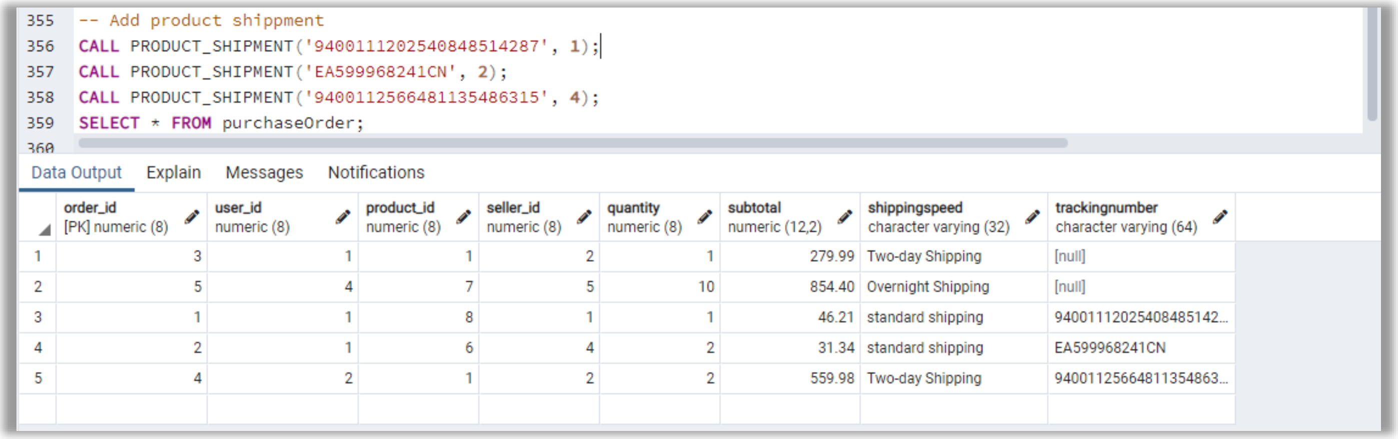
Product Shipment use case

Below is the procedure for product shipment use case, when amazon wants to ship an order, it will add a tracking number to the order so that customers can see the tracking number and track their package.



```sql
-- Product Shipment Use Case --
CREATE OR REPLACE PROCEDURE PRODUCT_SHIPMENT(
 p_tracking_number IN VARCHAR,  -- Tracking id for the order
 p_order_id IN NUMERIC) -- Order number
 LANGUAGE plpgsql
AS $$
BEGIN
 -- Add tracking number to the order
 UPDATE PurchaseOrder SET TrackingNumber = p_tracking_number
 WHERE order_id = p_order_id;
END;
$$;
```

Below code shows the example of using PRODUCT SHIPMENT procedure to add tracking id to particular order. As shown below, three orders were shipped and their tracking number is appeared in the purchase order table. The order is not shipped yet will still have a NULL value for their tracking number.



```sql
-- Add product shippment
CALL PRODUCT_SHIPMENT('9400111202540848514287', 1);
CALL PRODUCT_SHIPMENT('EA599968241CN', 2);
CALL PRODUCT_SHIPMENT('9400112566481135486315', 4);
SELECT * FROM purchaseOrder;
```

Data Output | Explain | Messages | Notifications

	order_id [PK] numeric (8)	user_id numeric (8)	product_id numeric (8)	seller_id numeric (8)	quantity numeric (8)	subtotal numeric (12,2)	shippingspeed character varying (32)	trackingnumber character varying (64)
1	3	1	1	2	1	279.99	Two-day Shipping	[null]
2	5	4	7	5	10	854.40	Overnight Shipping	[null]
3	1	1	8	1	1	46.21	standard shipping	94001112025408485142…
4	2	1	6	4	2	31.34	standard shipping	EA599968241CN
5	4	2	1	2	2	559.98	Two-day Shipping	94001125664811354863…

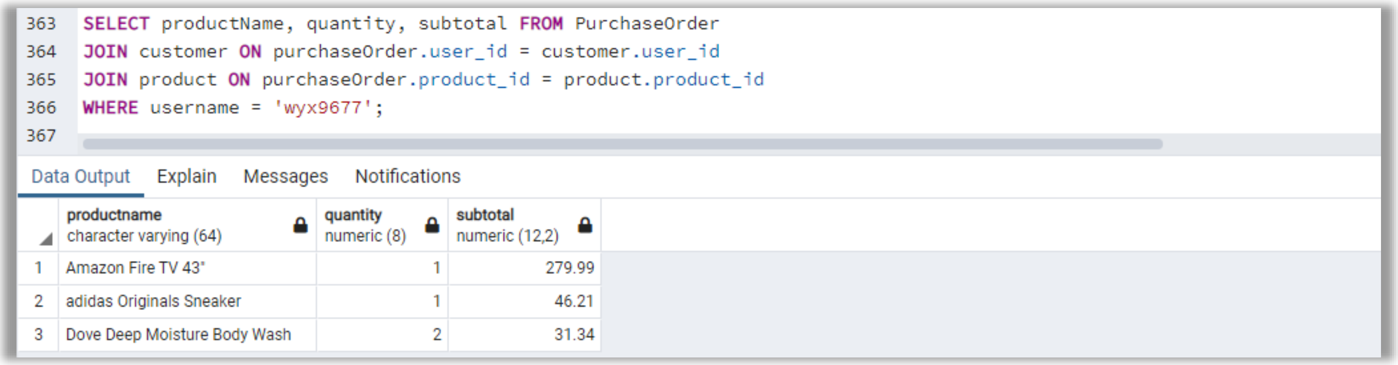
Question Identification and Explanations

Some insertion questions like how to create new user, seller, new product and the process of product purchase, product shipment and product delivery is answered in the stored procedure part. Some other useful query questions for the organization are stated below.

- If the user wants to check his/her order history, and wants to see a list of products purchased, quantity of product, total money spends on that product.
- If the user has some problem with one or more for his order, and he would like to contact the seller. So, he wants to see information cantinas the product name, seller name, seller email and seller phone.
- If Amazon wants to know which product has inventory that less than a threshold, for example 100. Along by its current inventory, product name, and warehouse name.
- If a seller wants to know the total number of purchases for each product category, create a list contains information for category name and total number of purchased order.
- If the warehouse wants to check the products and their conditions. Create a list contains the product name, product quantity, and condition in a specific warehouse.

Query Executions and Explanations

Replace this with queries answering the questions, along with screenshots and explanations.
First query is used when a customer wants to know all the products he or she purchased, along with the product's name, purchased quantity and money cost. And it was done by join the purchase order table with product and customer table to obtain the customer id and product name. in the query below, the customer has username "wyx9677" wants to check his purchase history.

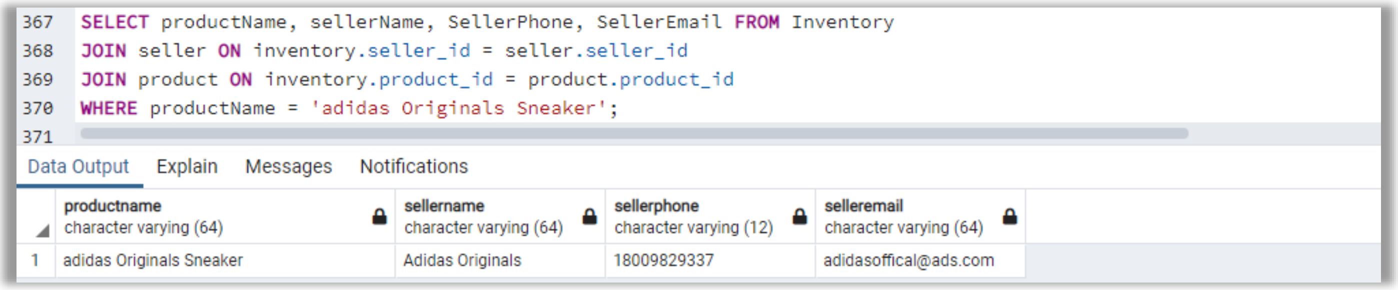


```sql
SELECT productName, quantity, subtotal FROM PurchaseOrder
JOIN customer ON purchaseOrder.user_id = customer.user_id
JOIN product ON purchaseOrder.product_id = product.product_id
WHERE username = 'wyx9677';
```

Data Output Explain Messages Notifications

	productname character varying (64)	quantity numeric (8)	subtotal numeric (12,2)
1	Amazon Fire TV 43"	1	279.99
2	adidas Originals Sneaker	1	46.21
3	Dove Deep Moisture Body Wash	2	31.34

Second query is used when a customer wants to contact the seller for a particular product. The customer wants to know the seller's phone number and email address, along with the product name and seller name. Because the customer can only see products that are in inventory, the inventory table was joined with seller and product table to get the seller's name, phone and email address. In the query below, a customer wants to know the contact information for seller of "adidas Originals Sneaker".

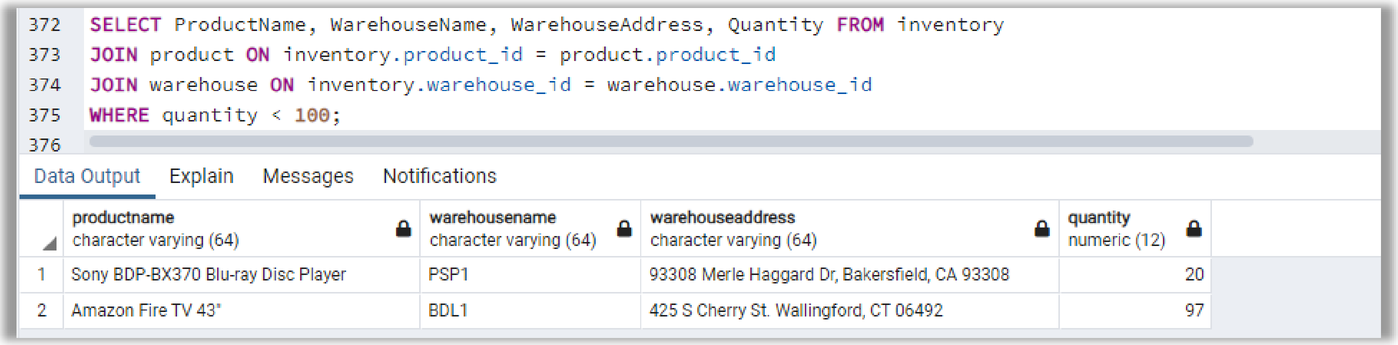


```sql
SELECT productName, sellerName, SellerPhone, SellerEmail FROM Inventory
JOIN seller ON inventory.seller_id = seller.seller_id
JOIN product ON inventory.product_id = product.product_id
WHERE productName = 'adidas Originals Sneaker';
```

Data Output Explain Messages Notifications

	productname character varying (64)	sellername character varying (64)	sellerphone character varying (12)	selleremail character varying (64)
1	adidas Originals Sneaker	Adidas Originals	18009829337	adidasoffical@ads.com

Next query is used when warehouse wants to check which product has low inventory, for example less than 100. And Amazon wants to know the product name, warehouse name, location and the current quantity of the product. This was done by join the inventory table with product and warehouse table to get the warehouse information and use a where statement to get all product that has inventory less than 100.

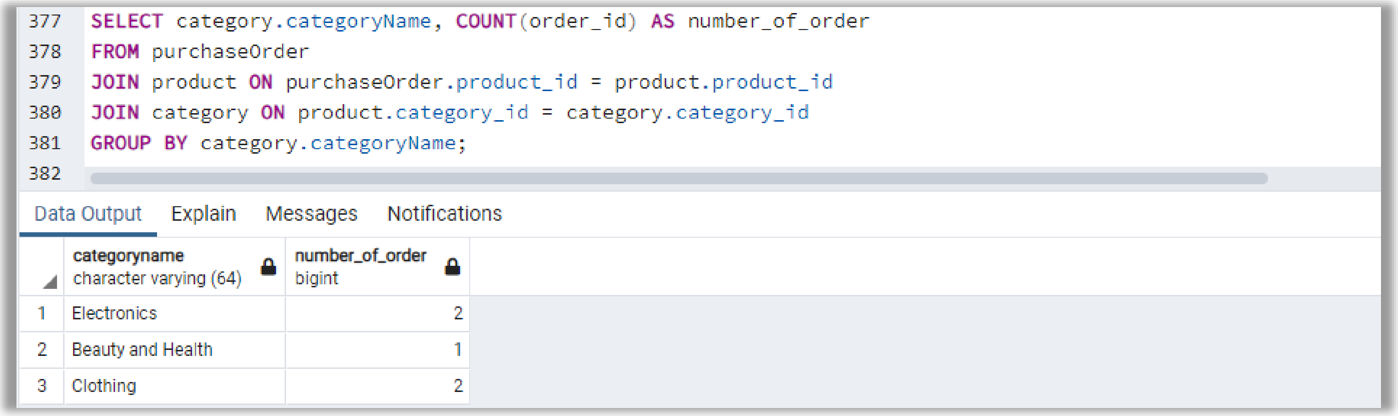


```sql
SELECT ProductName, WarehouseName, WarehouseAddress, Quantity FROM inventory
JOIN product ON inventory.product_id = product.product_id
JOIN warehouse ON inventory.warehouse_id = warehouse.warehouse_id
WHERE quantity < 100;
```

Data Output Explain Messages Notifications

	productname character varying (64)	warehousename character varying (64)	warehouseaddress character varying (64)	quantity numeric (12)
1	Sony BDP-BX370 Blu-ray Disc Player	PSP1	93308 Merle Haggard Dr, Bakersfield, CA 93308	20
2	Amazon Fire TV 43"	BDL1	425 S Cherry St. Wallingford, CT 06492	97

This query is used when someone wants to know which product category has the most purchase order. It will query the category name along with its number of orders. And this is done by using the group by statement to group the category and use count function to count number of purchases.

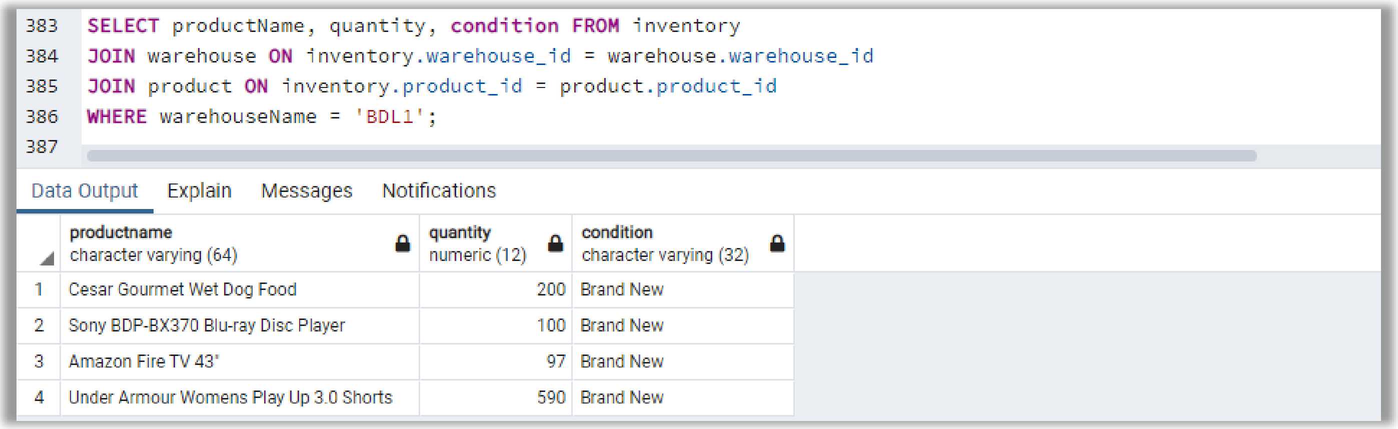


```sql
SELECT category.categoryName, COUNT(order_id) AS number_of_order
FROM purchaseOrder
JOIN product ON purchaseOrder.product_id = product.product_id
JOIN category ON product.category_id = category.category_id
GROUP BY category.categoryName;
```

Data Output Explain Messages Notifications

	categoryname character varying (64)	number_of_order bigint
1	Electronics	2
2	Beauty and Health	1
3	Clothing	2

This query is used when a warehouse wants to check its inventory, and need information include product's name, quantity and their condition. This is done by join the inventory table with warehouse and product table to get the condition and name of the product. As shown below, the example shows the inventory status in warehouse 'BDL1'.



```sql
SELECT productName, quantity, condition FROM inventory
JOIN warehouse ON inventory.warehouse_id = warehouse.warehouse_id
JOIN product ON inventory.product_id = product.product_id
WHERE warehouseName = 'BDL1';
```

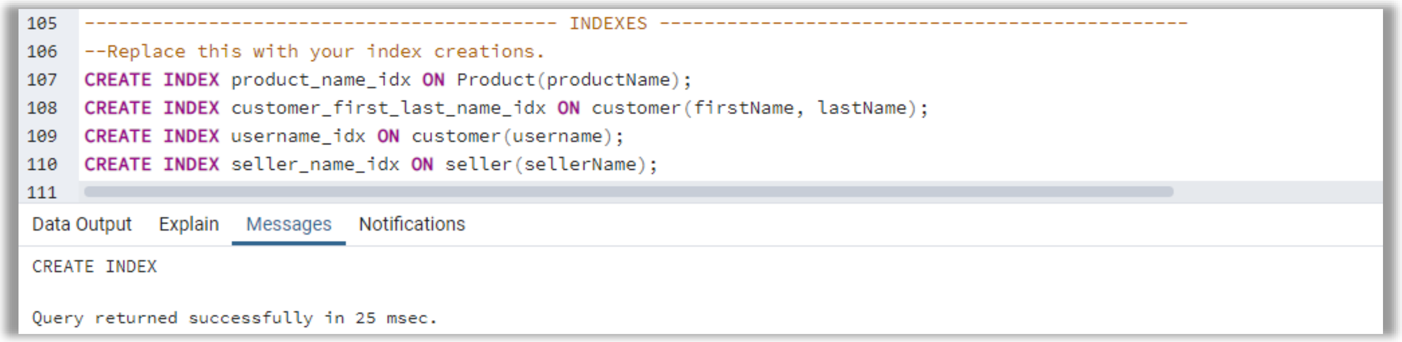
Data Output Explain Messages Notifications

	productname character varying (64)	quantity numeric (12)	condition character varying (32)
1	Cesar Gourmet Wet Dog Food	200	Brand New
2	Sony BDP-BX370 Blu-ray Disc Player	100	Brand New
3	Amazon Fire TV 43"	97	Brand New
4	Under Armour Womens Play Up 3.0 Shorts	590	Brand New

Index Identification and Creations

Some index I think are useful to create include the product name index. Product name index can be useful when perform query include product's name. Because the number of product name on Amazon can be very large. By using the product name index it can save a lot of processing time.
Next index is created on customer's first and last name. This can be useful when Amazon or seller wants to search customer's information.
Third index is created on customer's username. Same as above, this index can speed up the processing time for query related to customers.
Last index is created on seller's name. As we know there are lots of sellers on Amazon. The index is useful when customer wants to find the seller's information, such as contact information.

Below is the code for creating those indexes.



```sql
------------------------------------------- INDEXES ------------------------------------------------
--Replace this with your index creations.
CREATE INDEX product_name_idx ON Product(productName);
CREATE INDEX customer_first_last_name_idx ON customer(firstName, lastName);
CREATE INDEX username_idx ON customer(username);
CREATE INDEX seller_name_idx ON seller(sellerName);
```

```
CREATE INDEX

Query returned successfully in 25 msec.
```

Summary and Reflection

In project Iteration one, Introduction and general overview for this project has been stated, and some simple use case and their related fields has been identified. In the real world, Amazon online marketplace is a huge system, therefore, I might not be able to cover every aspect of the system. However, I am excited to continue develop the database system. It can help me to learn more about how Amazon online marketplace really works.

In project iteration 2 and 3 conceptual ERD diagram and physical ERD for Amazon online marketplace was created. The difference between conceptual and physical ERD is that all types for attributes are added to the table and the ERD should be normalized. Because each seller can sell many products and each product can be sold by many sellers, table product_seller was added to remove the M:N relationship. Because each product can be purchased by many customer and each customer can purchase many products, Purchase Order table was added to remove the M:N relationship. Because each product can be delivered to many warehouse and a warehouse can receive many products, Product_deliver table was added to remove the M:N relationship.

In project iteration 4, table structure and related sequence was created in SQL script. Tables include Category, Product, Customer, Seller, Warehouse, Inventory, Purchase order and Product delivery. The sequences are used for automatically generate primary key ID numbers for different tables, which makes them easier to manage. In last iteration, stored procedures for different use cases were created, which includes new customer account use case, new product use case, product delivery use case, product purchase use case and product shipment use case. Then those procedure were used to insert values to database. After that, some business questions were created and their corresponding Query Executions were provided.

Attribution
Amazon logo retrieved from http://amazon.com on Oct-07-2017.