DATA MANAGEMENT FOR INFORMATION TECHNOLOGY

FINAL TERM PROJECT:

Introduction to “Selling on Amazon”

MAHIM DHUNGEL

Table of Contents:

Introduction:

As of June 2022, Amazon, a behemoth in the online world holds roughly 37.8% of the total market share in North America while accounting for roughly 9.2% market share globally. It is also estimated, but rather hard to prove, that amazon accounts for almost 1% of all sales worldwide that includes sales from both ecommerce and from local vendors (including from countries like India, Haiti and Nepal where it's hard to estimate the real sales).

Amazon could pull this feat by heavily investing on technological infrastructures; part of which is the Database.

In this project I have implemented the necessary designs and models to accommodate the inventory, orders, shipping, and customer knowledge on behalf of the seller. In Amazon Fulfillment there are features such as: the seller only needs to deliver the products to one of Amazon's warehouses and Amazon takes over from there or Amazon creating a generalized marketplace where any seller can start without much barrier to entry, in this project I have implemented similar concepts as well.

Since it is practically impossible to design and build a full production-capable database for a large enterprise such as Amazon by a single person because the database would have thousands of tables and business rules that need to be implemented. This project is a small snapshot of one of the core functionalities of amazon: Amazon Fulfillment.

However, I wanted to demonstrate that Amazon Fulfillment could be applied in a business sense of my own for which I have made some business assumptions and added business rules which have not been included in the schema.

Schema Overview:

Below you can see the overview of the Schema for Amazon Fulfillment:

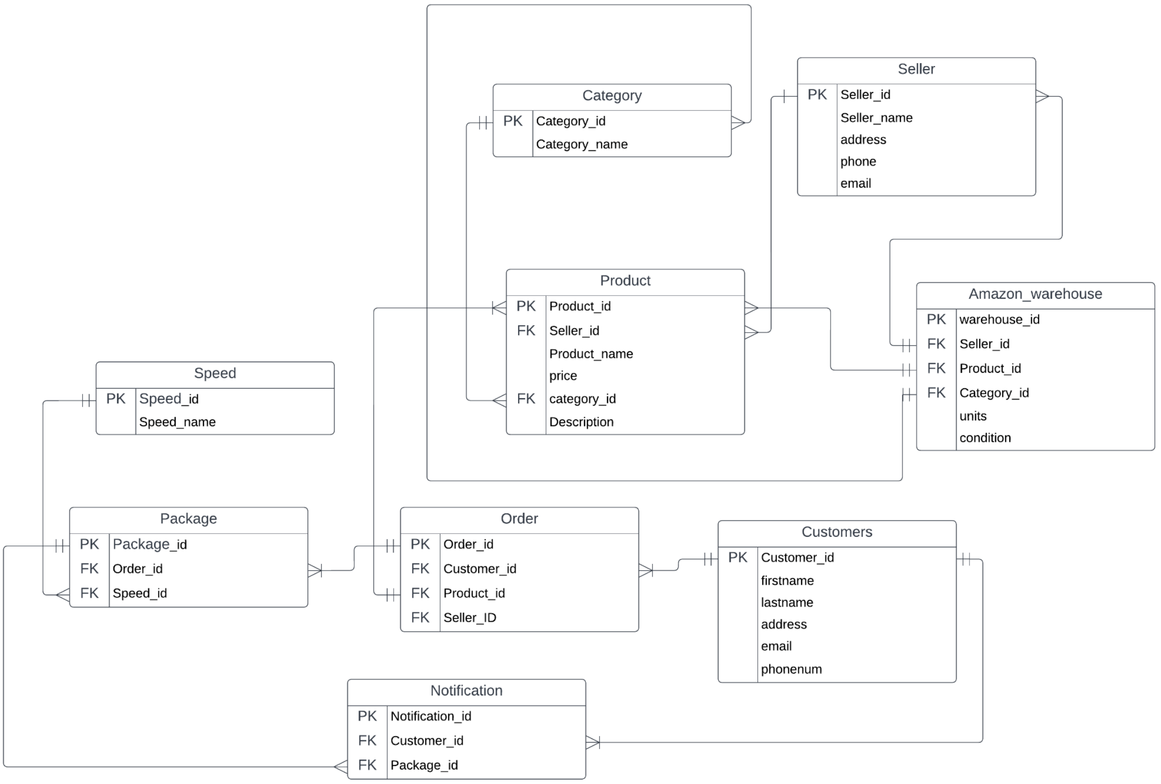
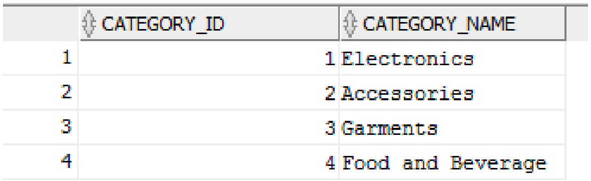


Fig: Diagram made from LucidChart

Table Overview:

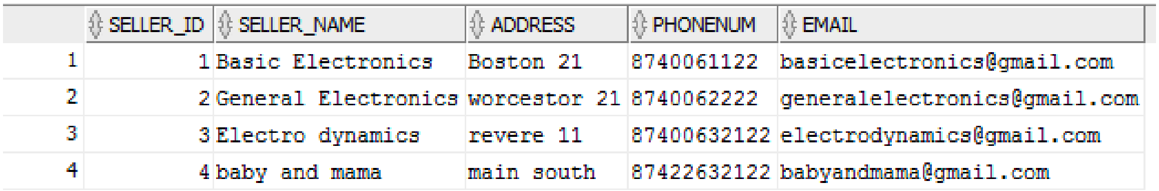
The table is populated with dummy data for Proof of Concept.

Category Table:



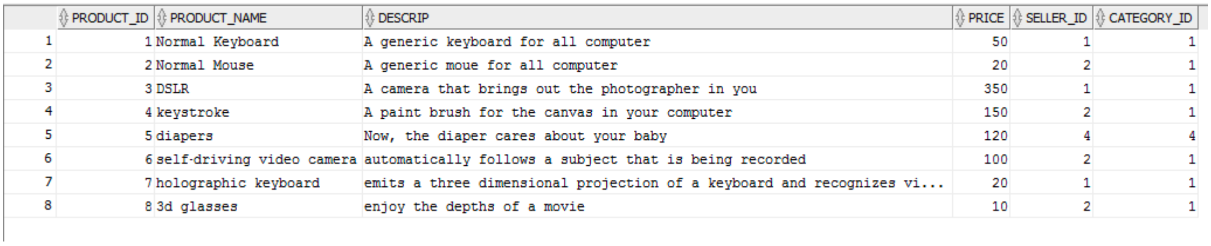
	CATEGORY_ID	CATEGORY_NAME
1	1	Electronics
2	2	Accessories
3	3	Garments
4	4	Food and Beverage

Seller Table:



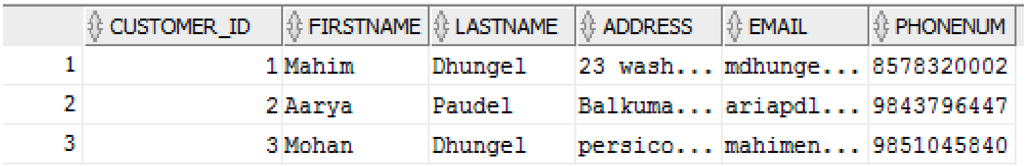
	SELLER_ID	SELLER_NAME	ADDRESS	PHONENUM	EMAIL
1	1	Basic Electronics	Boston 21	8740061122	basicelectronics@gmail.com
2	2	General Electronics	worcestor 21	8740062222	generalelectronics@gmail.com
3	3	Electro dynamics	revere 11	87400632122	electrodynamics@gmail.com
4	4	baby and mama	main south	87422632122	babyandmama@gmail.com

Products Table:



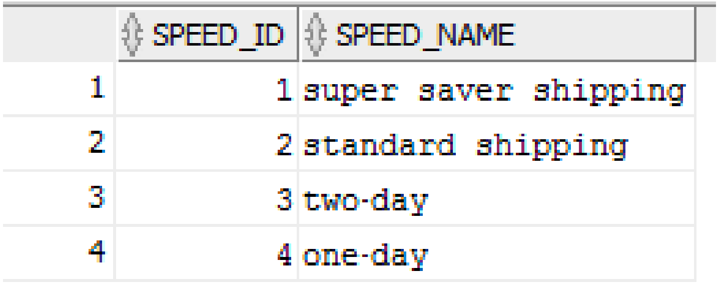
	PRODUCT_ID	PRODUCT_NAME	DESCRIP	PRICE	SELLER_ID	CATEGORY_ID
1	1	Normal Keyboard	A generic keyboard for all computer	50	1	1
2	2	Normal Mouse	A generic moue for all computer	20	2	1
3	3	DSLR	A camera that brings out the photographer in you	350	1	1
4	4	keystroke	A paint brush for the canvas in your computer	150	2	1
5	5	diapers	Now, the diaper cares about your baby	120	4	4
6	6	self-driving video camera	automatically follows a subject that is being recorded	100	2	1
7	7	holographic keyboard	emits a three dimensional projection of a keyboard and recognizes vi...	20	1	1
8	8	3d glasses	enjoy the depths of a movie	10	2	1

Customer Table:



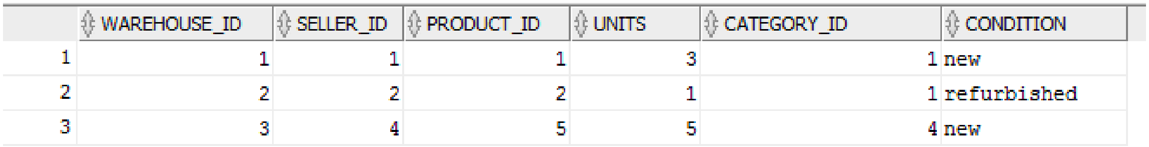
	CUSTOMER_ID	FIRSTNAME	LASTNAME	ADDRESS	EMAIL	PHONENUM
1	1	Mahim	Dhungel	23 wash...	mdhunge...	8578320002
2	2	Aarya	Paudel	Balkuma...	ariapdl...	9843796447
3	3	Mohan	Dhungel	persico...	mahimen...	9851045840

Speed Table:



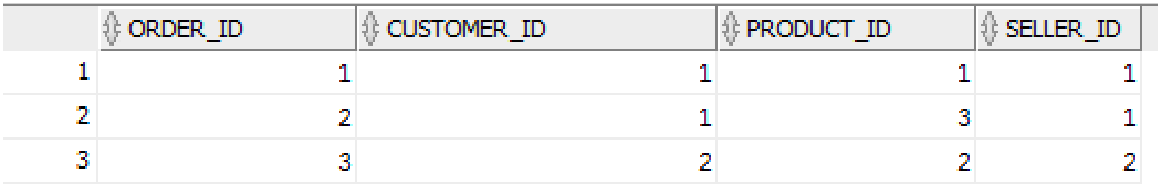
	SPEED_ID	SPEED_NAME
1	1	super saver shipping
2	2	standard shipping
3	3	two-day
4	4	one-day

Warehouse Table:



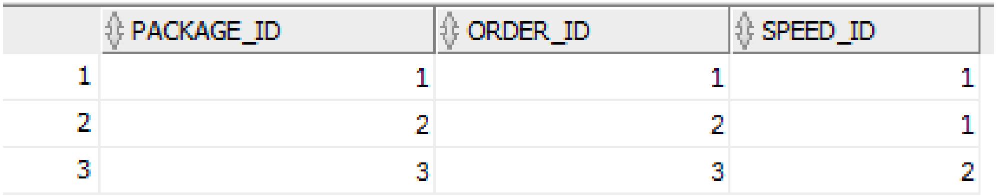
	WAREHOUSE_ID	SELLER_ID	PRODUCT_ID	UNITS	CATEGORY_ID	CONDITION
1	1	1	1	3	1	new
2	2	2	2	1	1	refurbished
3	3	4	5	5	4	new

Order Table:



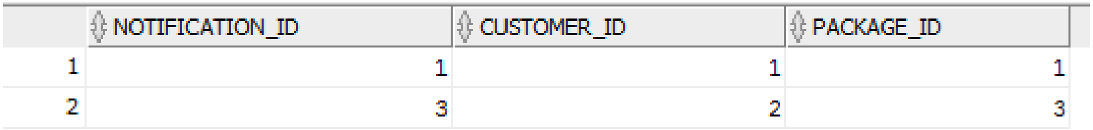
	ORDER_ID	CUSTOMER_ID	PRODUCT_ID	SELLER_ID
1	1	1	1	1
2	2	1	3	1
3	3	2	2	2

Package Table:



	PACKAGE_ID	ORDER_ID	SPEED_ID
1	1	1	1
2	2	2	1
3	3	3	2

Notification Table:



	NOTIFICATION_ID	CUSTOMER_ID	PACKAGE_ID
1	1	1	1
2	3	2	3

Structural Business Rules:

The structural business rules for the Schema defined above are as follows:

- A customer can browse and buy the product on Amazon.
- Multiple sellers can sell the same product.
- A customer can have only one account; multiple customer accounts are under the platform.
- A 3rd party seller can have only one account; multiple 3rd party sellers are under the platform.
- An inventory holds multiple products from multiple sellers.
- Once the product is sold from the platform, the inventory decreases.
- The product can be uploaded by multiple sellers at the same time.
- The products can only be shipped under one shipping detail; a shipping detail can have multiple orders.
- Products can have only one purchase detail; a purchase detail can have multiple products.

Below are a series of use cases that describe the operations the database of Renter's Den would support ideally.

Business Assumptions as Mentioned:

The assumptions I’m making for the database design of an ecommerce website that I call “Renter’s Den”(acronym for renting over the internet) are as following:

Since my assumption over what I have observed in the world right now is that a lot of young people are renting items rather than buying them outright or people shop over Instagram, I decided to add features from different existing services. So Renter’s Den is a combination of some functionalities among Rent the runway, Amazon and Instagram.

+ +

Amazon
Fulfillment

Rent the
runway renting

Instagram
shopping

Therefore Renter’s Den is an E-commerce platform for renting clothes in the modern sharing economy in the world.

Business Use Case 1:

New garment listed and updated by Renter’s Den platform itself:

- Since the garments are sold both on the platform as well as on Instagram, the garments providing all the necessary details are uploaded on both the platforms simultaneously.
- Since this an ecommerce site specifically for selling and renting clothes, every new garment added is linked to a garment category which are predefined by Renter’s Den (“Tshirt”, “Pants”, “Jackets” etc)
- If the garment is already sold, then Renter’s Den updates the description caption of the post informing the other potential customers that the item is no longer available in the inventory.

Business Use Case 2:

New garment listed and updated By 3rd party seller.

- The seller who is uploading a new listing will have their listing on both the platforms or can choose not to upload any one platform they desire.
- When the seller has a new garment, it is updated with its description, size, quality, material made with and price.
- The garment also appears on the Shopping section of instagram for visibility.
- If the garment is already sold, the seller updates the description caption of the post informing the other potential customers that the item is no longer available in the inventory.

Business Use Case 3:

New 3rd party seller Account.

This occurs when a Customer signs up for a new account on Renter’s Den, so they can begin purchasing garments:

- The 3rd party seller provides Renter’s Den with basic information including a business name, business address, phone number, and a business email address, types of garments they want to sell, inventory details, billing address etc.
- Renter’s Den creates an account for the seller, enabling the customer to purchase garments from them

Business Use Case 4:

Garment bought or rented by the customer [from both the website and Instagram]

Sub Case 1: This occurs when a customer purchases or rents a garment from Renter's Den that was provided by the platform itself.

- The customer logs in to the website with their credentials.
- The customer selects one or more garments. When selecting a garment, the customer is actually selecting from the inventory of the website itself.
- Renter's Den decreases their inventory for the garment purchased or rented. ● Renter's Den creates an order with the order id and shipping details.

Sub Case 2: Under Instagram.

- The customer selects a garment and can purchase it by chatting with the representative of the website after which the representative will decrease the item from the inventory or can directly purchase or rent using Instagram Checkout which will also decrease the item from the inventory.
- Renter's Den creates an order with the order id and shipping details.

Business Use Case 5:

Sub Case 1: This occurs when a customer purchases or rents a garment from Renter's Den that was provided by a 3rd party seller:

- The customer logs in to the website with their credentials.
- The customer selects one or more garments. When selecting a garment, the customer is actually selecting a particular seller's inventory.
- Renter's Den decreases the seller's inventory for the garment purchased or rented ● Renter's Den creates an order with the order id and shipping details.

Sub Case 2: Under Instagram.

- The customer selects a garment and can purchase or rent it by chatting with the 3rd party seller directly after which the 3rd party seller will decrease the item from the inventory or can directly purchase or rent using Instagram Checkout which will also decrease the item from the inventory.

- Renter’s Den creates an order with the order id and shipping details.

Business Use Case 6:

New Customer Account

This occurs when a Customer signs up for a new account on Renter’s Den, so they can begin purchasing garments:

- The new customer provides Renter’s Den with basic information including a username, an address, phone number, and an email address.
- After the validation of the email address is done, the site creates an account for the customer.
- Renter’s Den creates an account for the customer, enabling the customer to purchase garments.

Business Use Case 7:

Wishlist:

The customer can add a garment that they like to their wishlist which they can come back to provided the garment is still in the inventory. If the garment has already been sold, then the wish list becomes empty but if the garment is just rented out, the update on the wishlist reads:

“*Currently rented out. Please wait till it is available.*”

Business Use Case 8:

Dry Cleaning:

Since the idea is heavily influenced by Rent the runway, once the garment is returned, it needs to go through the cleaning process before it is rented out again. The cleaning process requires the garment type, garment id, cleaning process name (because every garment needs to be cleaned differently with different processes)

Business Use Case 9:

Payment:

Once the garment is rented or sold, the payment is processed. It can be under 4 different types:

a. Card (debit and credit)

b. Mobile Payment

c. Bank transfer

d. Cash on delivery.

The payment details are then generated accordingly.

Business Use Case 10:

Product Shipment by Renter’s Den.

This occurs when Renter’s Den ships the product that consumer has purchased:

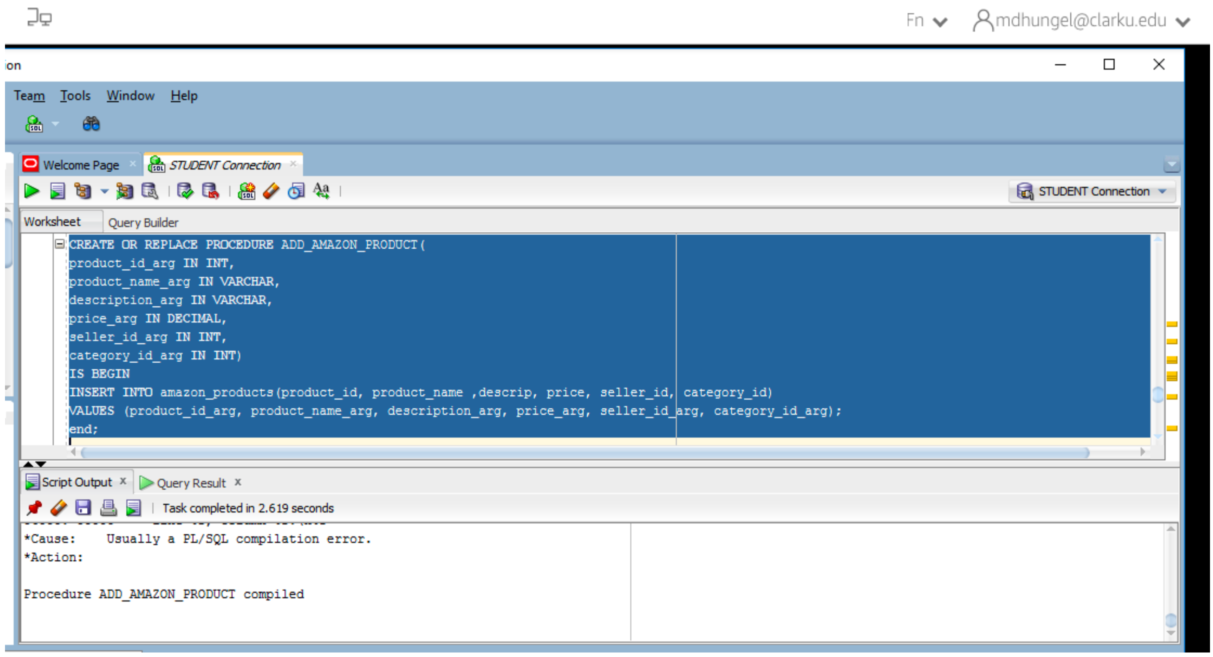
- Renter’s Den packages the purchased products and assigns a tracking id.
- Renter’s Den links the package to the customer’s order.
- Renter’s Den ships the package to the address linked to the customer’s account. However if the 3rd party has chosen not to use the inventory service, they can ship the item themselves.
- Renter’s Den notifies the customer that it has been shipped and provides the customer with the tracking ID.
- After the order is ready to be delivered, the delivery person takes it to the customer. However, if the customer wants to pick up the item by themselves, they can do so by visiting the seller’s store or the Renter’s Den as well.

Use Case Driven Aspects:

(All table creations and insertions images are in the zip file.)

Aspect 1: New Product Created by Seller

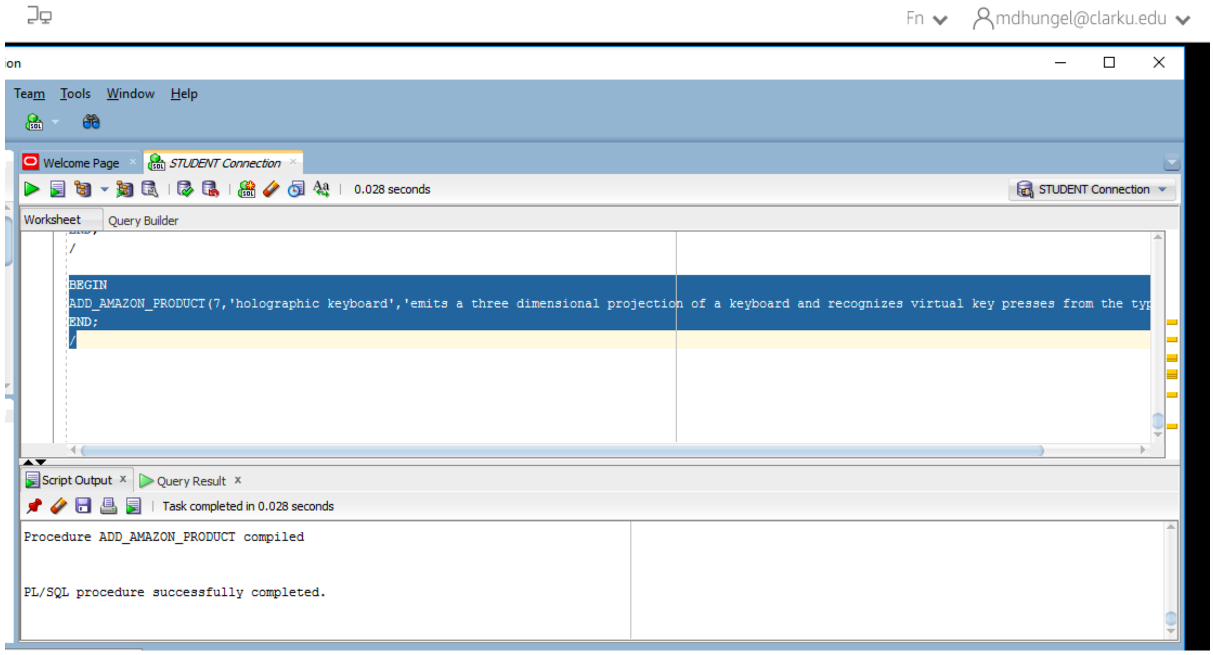
Creation of parameterized stored procedure to add a product by a seller.



```
CREATE OR REPLACE PROCEDURE ADD_AMAZON_PRODUCT(
product_id_arg IN INT,
product_name_arg IN VARCHAR,
description_arg IN VARCHAR,
price_arg IN DECIMAL,
seller_id_arg IN INT,
category_id_arg IN INT)
IS BEGIN
INSERT INTO amazon_products(product_id, product_name ,descrip, price, seller_id, category_id)
VALUES (product_id_arg, product_name_arg, description_arg, price_arg, seller_id_arg, category_id_arg);
end;
```

```
*Cause:    Usually a PL/SQL compilation error.
*Action:

Procedure ADD_AMAZON_PRODUCT compiled
```

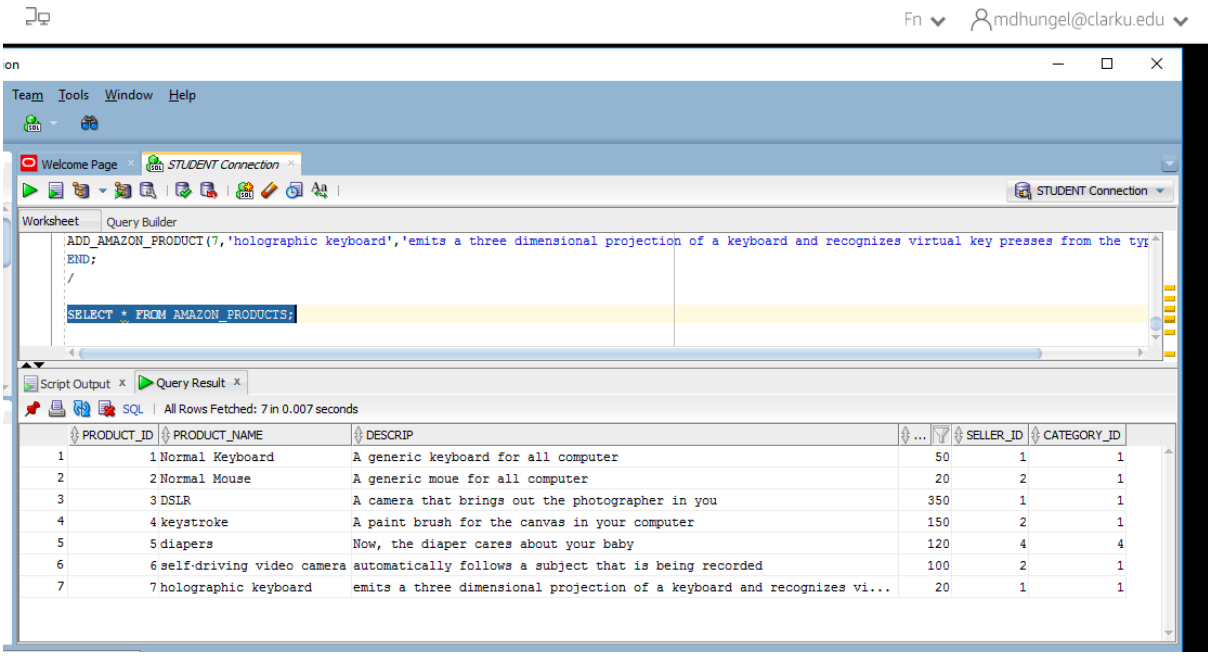
Use of the stored procedure to add a new product.



```
/

BEGIN
ADD_AMAZON_PRODUCT(7,'holographic keyboard','emits a three dimensional projection of a keyboard and recognizes virtual key presses from the typ
END;
/
```

```
Procedure ADD_AMAZON_PRODUCT compiled


PL/SQL procedure successfully completed.
```

SQL Query.



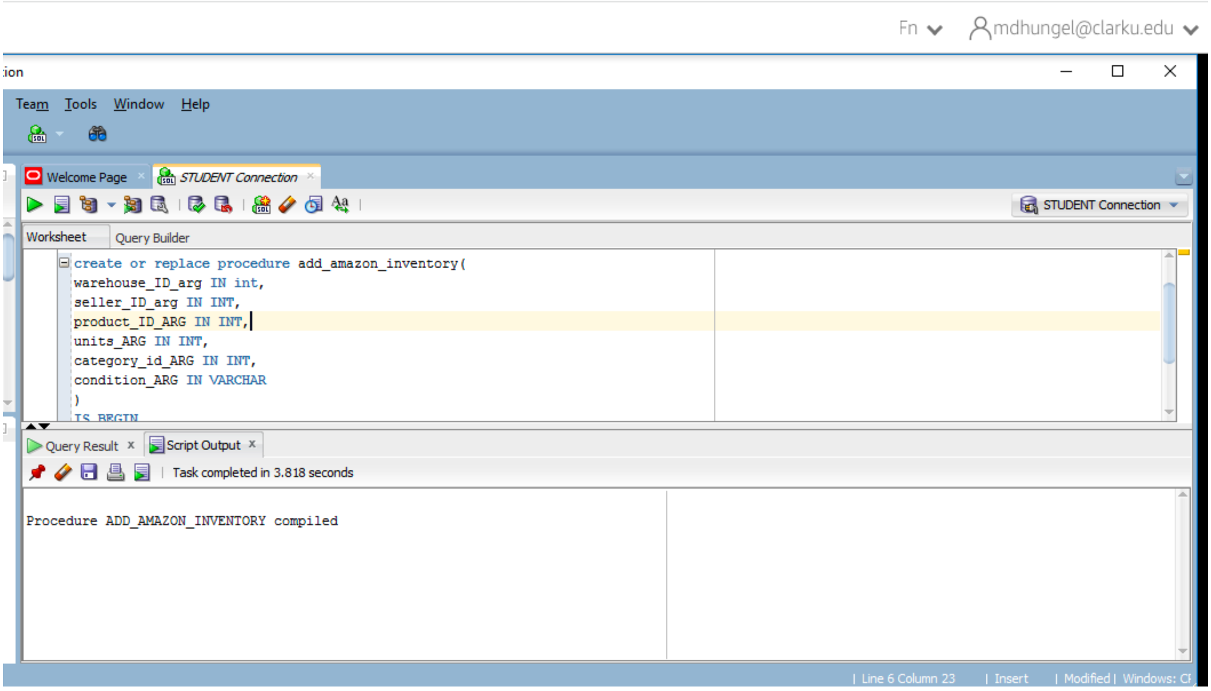
```
ADD_AMAZON_PRODUCT(7,'holographic keyboard','emits a three dimensional projection of a keyboard and recognizes virtual key presses from the typ
END;
/

SELECT * FROM AMAZON_PRODUCTS;
```

All Rows Fetched: 7 in 0.007 seconds

	PRODUCT_ID	PRODUCT_NAME	DESCRIP	...	SELLER_ID	CATEGORY_ID
1	1	Normal Keyboard	A generic keyboard for all computer	50	1	1
2	2	Normal Mouse	A generic moue for all computer	20	2	1
3	3	DSLR	A camera that brings out the photographer in you	350	1	1
4	4	keystroke	A paint brush for the canvas in your computer	150	2	1
5	5	diapers	Now, the diaper cares about your baby	120	4	4
6	6	self-driving video camera	automatically follows a subject that is being recorded	100	2	1
7	7	holographic keyboard	emits a three dimensional projection of a keyboard and recognizes vi...	20	1	1

Aspect 2: Amazon Receipt of Product from Seller

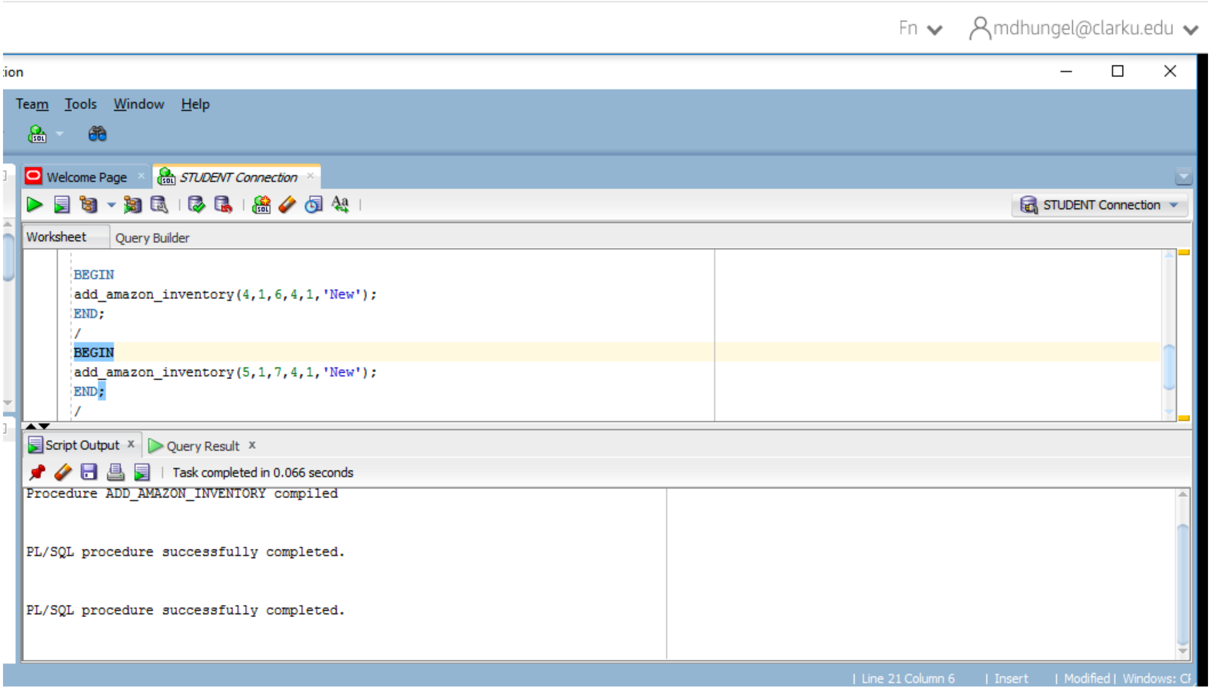
Creation of parameterized stored procedure that is used when any seller delivers any product to Amazon's warehouse.



```
create or replace procedure add_amazon_inventory(
warehouse_ID_arg IN int,
seller_ID_arg IN INT,
product_ID_ARG IN INT,
units_ARG IN INT,
category_id_ARG IN INT,
condition_ARG IN VARCHAR
)
IS BEGIN
```

```
Procedure ADD_AMAZON_INVENTORY compiled
```

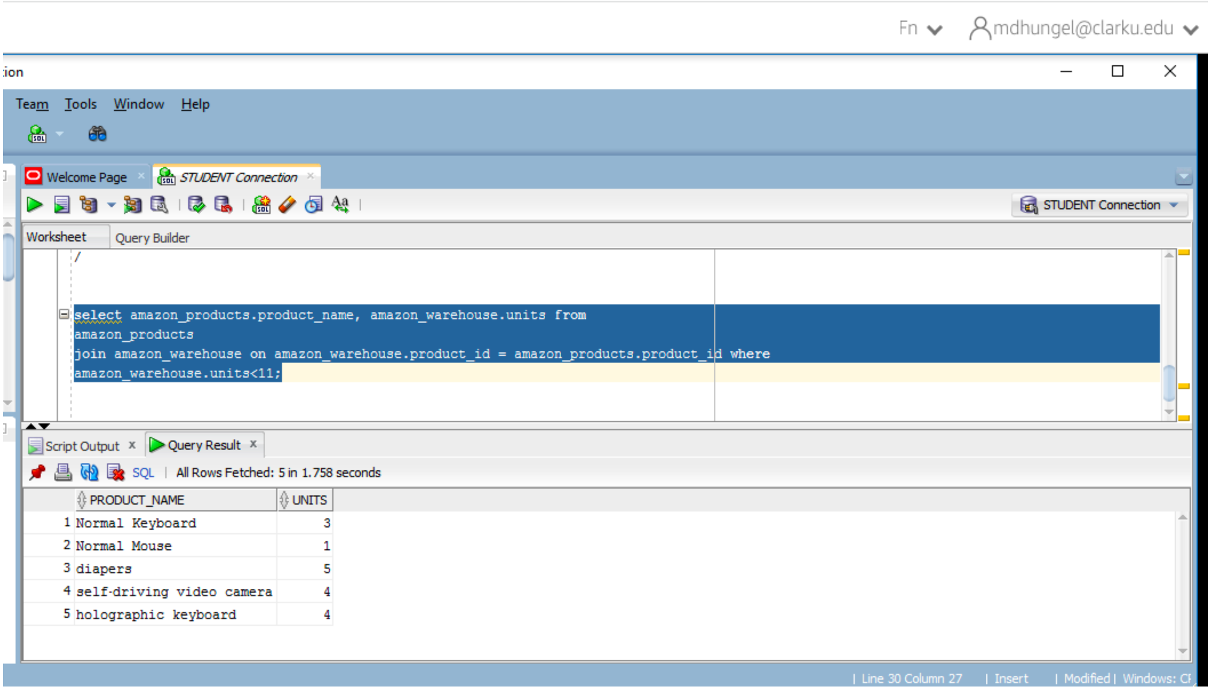
Use of the parameterized stored procedure to add a new products twice to update the inventory of these products for a seller



```
BEGIN
add_amazon_inventory(4,1,6,4,1,'New');
END;
/
BEGIN
add_amazon_inventory(5,1,7,4,1,'New');
END;
/
```

```
Procedure ADD_AMAZON_INVENTORY compiled


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.
```

Execution of a single query that provides a listing of all of its products that have an inventory of 11 or less.

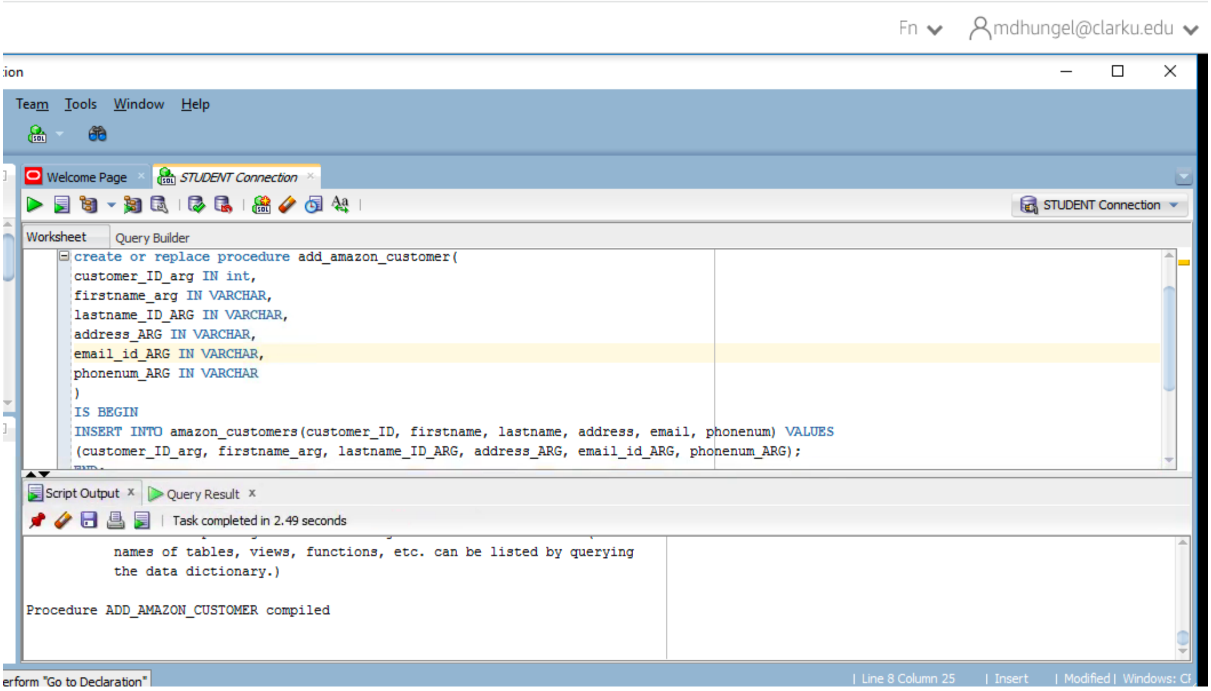


```
/


select amazon_products.product_name, amazon_warehouse.units from
amazon_products
join amazon_warehouse on amazon_warehouse.product_id = amazon_products.product_id where
amazon_warehouse.units<11;
```

	PRODUCT_NAME	UNITS
1	Normal Keyboard	3
2	Normal Mouse	1
3	diapers	5
4	self-driving video camera	4
5	holographic keyboard	4

Aspect 3: New Consumer Account

Creation of parameterized stored procedure that is used when any new customer signs up for a new account on Amazon.

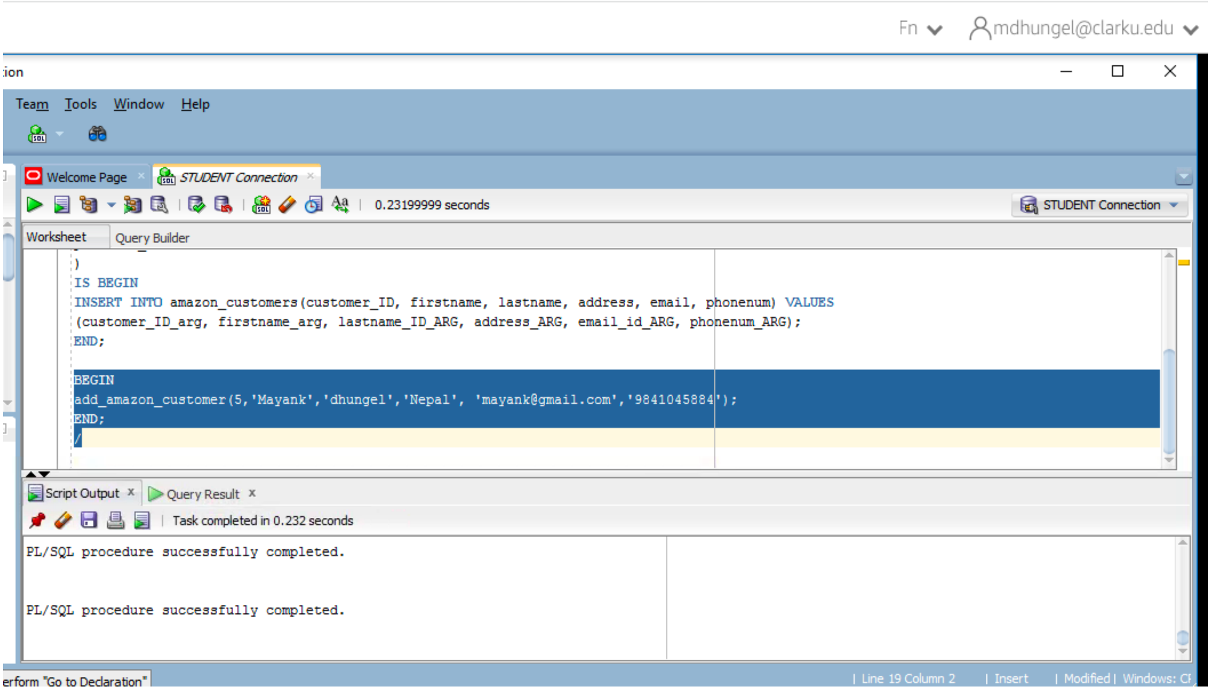


```
create or replace procedure add_amazon_customer(
customer_ID_arg IN int,
firstname_arg IN VARCHAR,
lastname_ID_ARG IN VARCHAR,
address_ARG IN VARCHAR,
email_id_ARG IN VARCHAR,
phonenum_ARG IN VARCHAR
)
IS BEGIN
INSERT INTO amazon_customers(customer_ID, firstname, lastname, address, email, phonenum) VALUES
(customer_ID_arg, firstname_arg, lastname_ID_ARG, address_ARG, email_id_ARG, phonenum_ARG);
```

Task completed in 2.49 seconds

```
           names of tables, views, functions, etc. can be listed by querying
           the data dictionary.)

Procedure ADD_AMAZON_CUSTOMER compiled
```

Use of the parameterized stored procedure to add a new customers



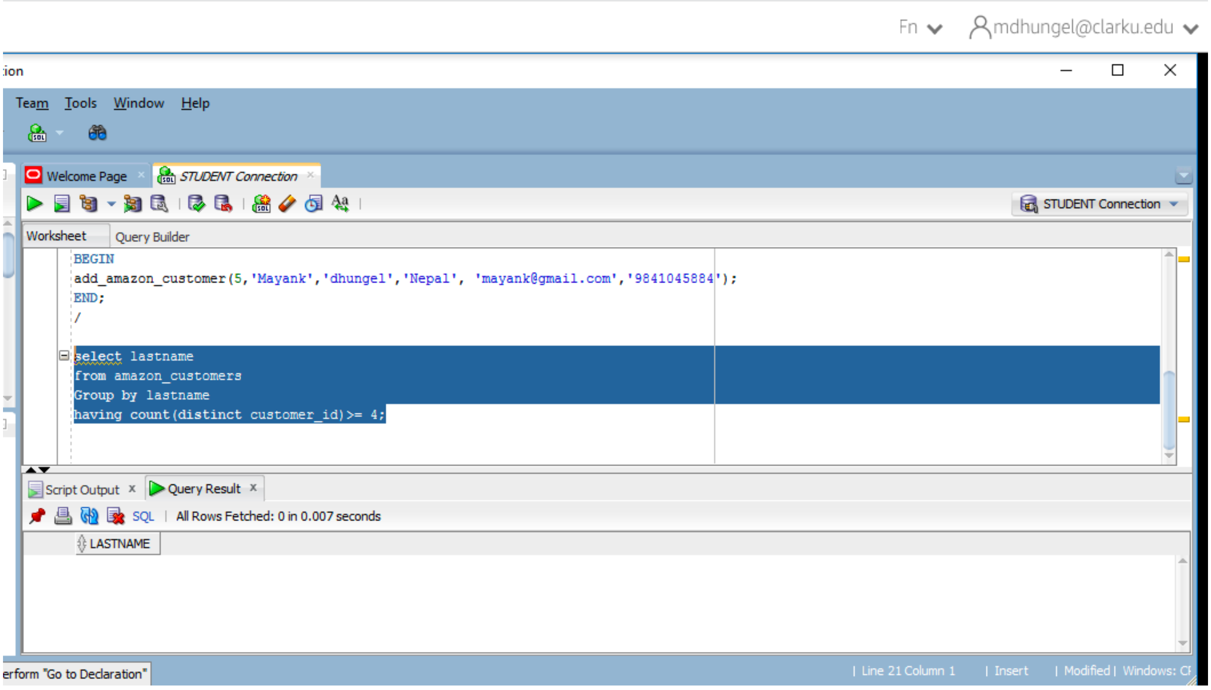
0.23199999 seconds

```
)
IS BEGIN
INSERT INTO amazon_customers(customer_ID, firstname, lastname, address, email, phonenum) VALUES
(customer_ID_arg, firstname_arg, lastname_ID_ARG, address_ARG, email_id_ARG, phonenum_ARG);
END;

BEGIN
add_amazon_customer(5,'Mayank','dhungel','Nepal', 'mayank@gmail.com','9841045884');
END;
/
```

Task completed in 0.232 seconds

```
PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.
```

Execution of a single query that is requested by Amazon to provide the last names of consumers where there are least 4 accounts associated with the last name.



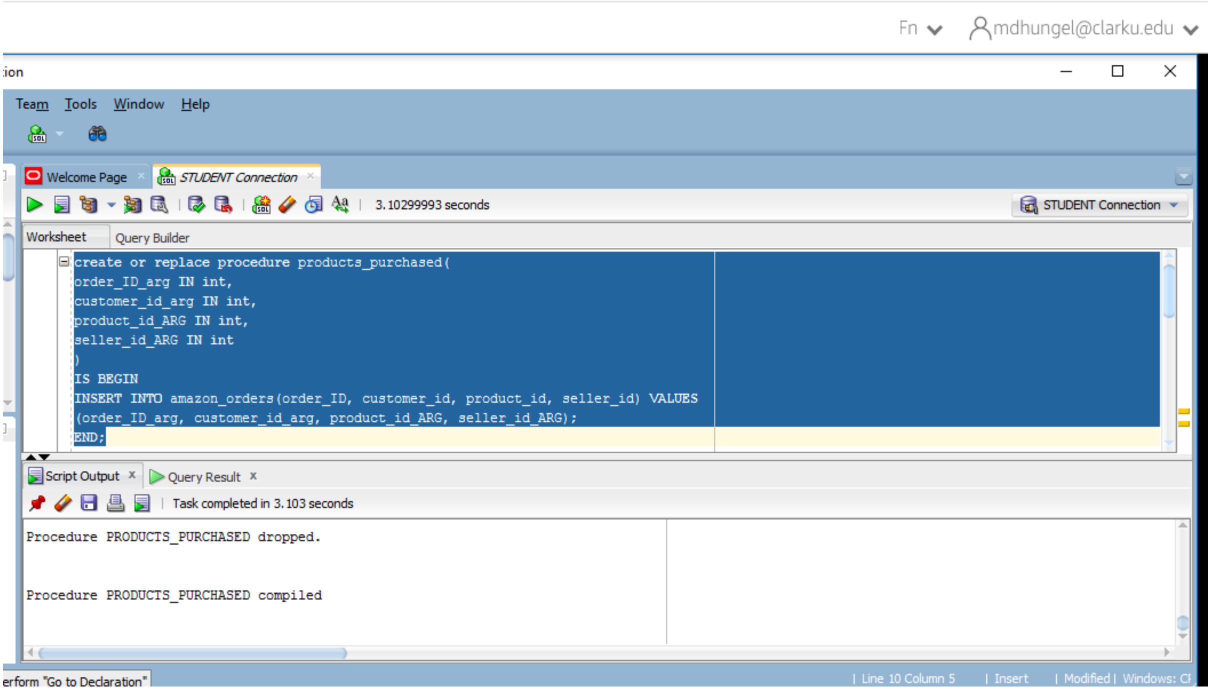
```
BEGIN
add_amazon_customer(5,'Mayank','dhungel','Nepal', 'mayank@gmail.com','9841045884');
END;
/

select lastname
from amazon_customers
Group by lastname
having count(distinct customer_id)>= 4;
```

All Rows Fetched: 0 in 0.007 seconds

LASTNAME

Aspect 4: Product Purchase by Consumer

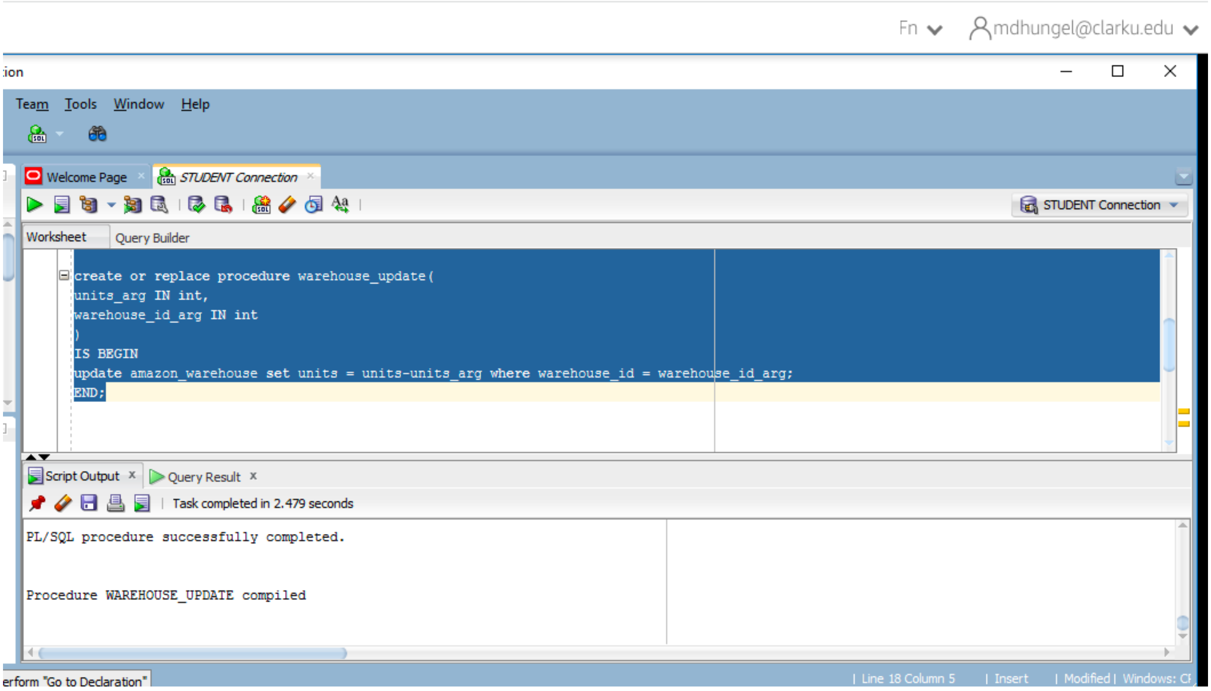
Creation of parameterized stored procedure that is used when any customer purchases any product.



```
create or replace procedure products_purchased(
order_ID_arg IN int,
customer_id_arg IN int,
product_id_ARG IN int,
seller_id_ARG IN int
)
IS BEGIN
INSERT INTO amazon_orders(order_ID, customer_id, product_id, seller_id) VALUES
(order_ID_arg, customer_id_arg, product_id_ARG, seller_id_ARG);
END;
```

```
Procedure PRODUCTS_PURCHASED dropped.


Procedure PRODUCTS_PURCHASED compiled
```

Creation of parameterized stored procedure that is used to update the units in the warehouse.

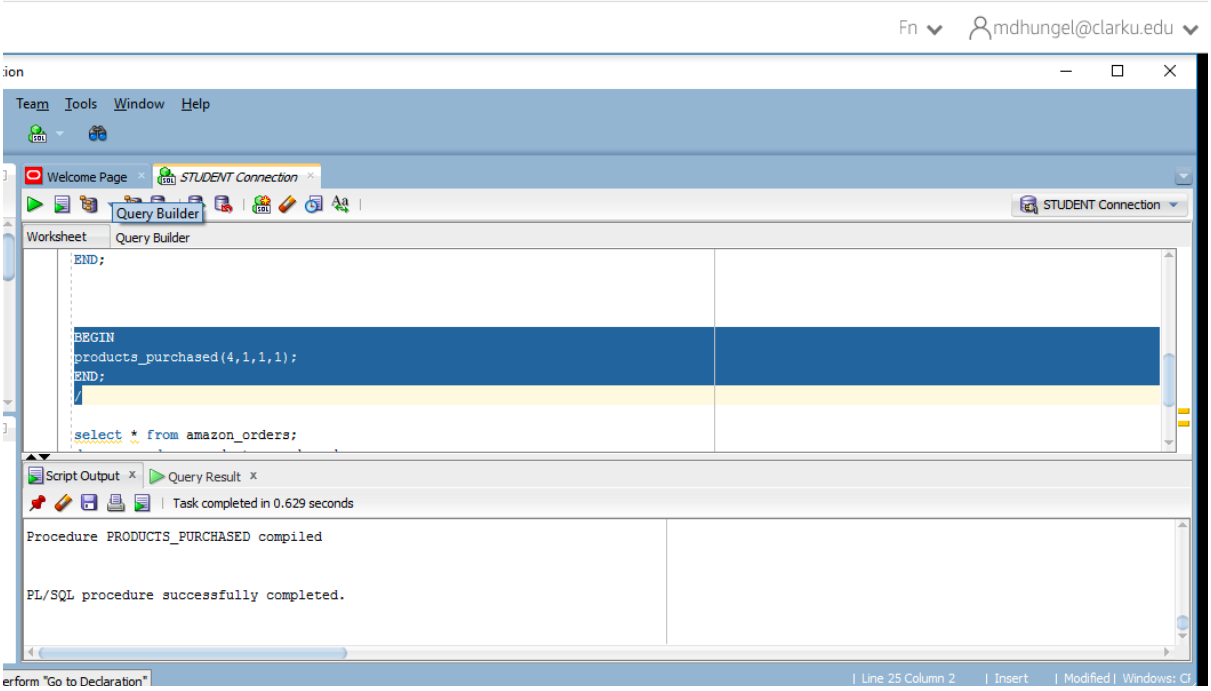


```
create or replace procedure warehouse_update(
units_arg IN int,
warehouse_id_arg IN int
)
IS BEGIN
update amazon_warehouse set units = units-units_arg where warehouse_id = warehouse_id_arg;
END;
```

Task completed in 2.479 seconds

```
PL/SQL procedure successfully completed.


Procedure WAREHOUSE_UPDATE compiled
```

Use of the parameterized stored procedure to add a new to purchase history

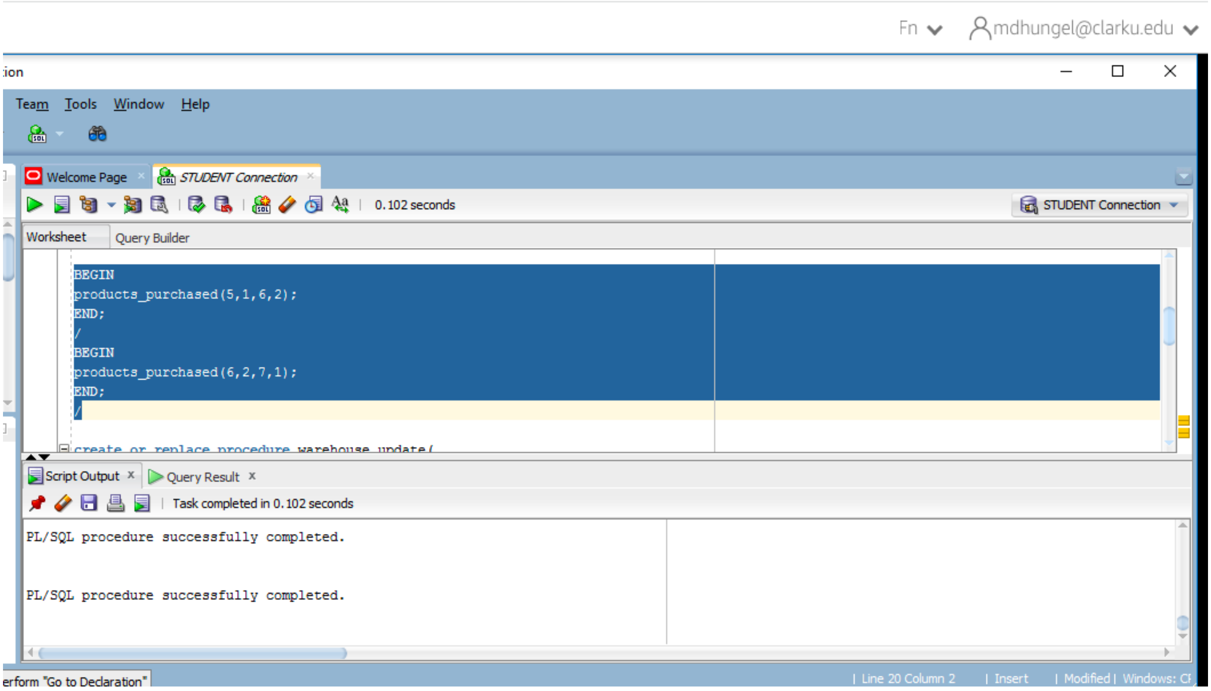


```
END;


BEGIN
products_purchased(4,1,1,1);
END;
/

select * from amazon_orders;
```

Task completed in 0.629 seconds

```
Procedure PRODUCTS_PURCHASED compiled


PL/SQL procedure successfully completed.
```

Use of the parameterized stored procedure to add a self-driving video camera, and your facilitator purchases holographic keyboards.

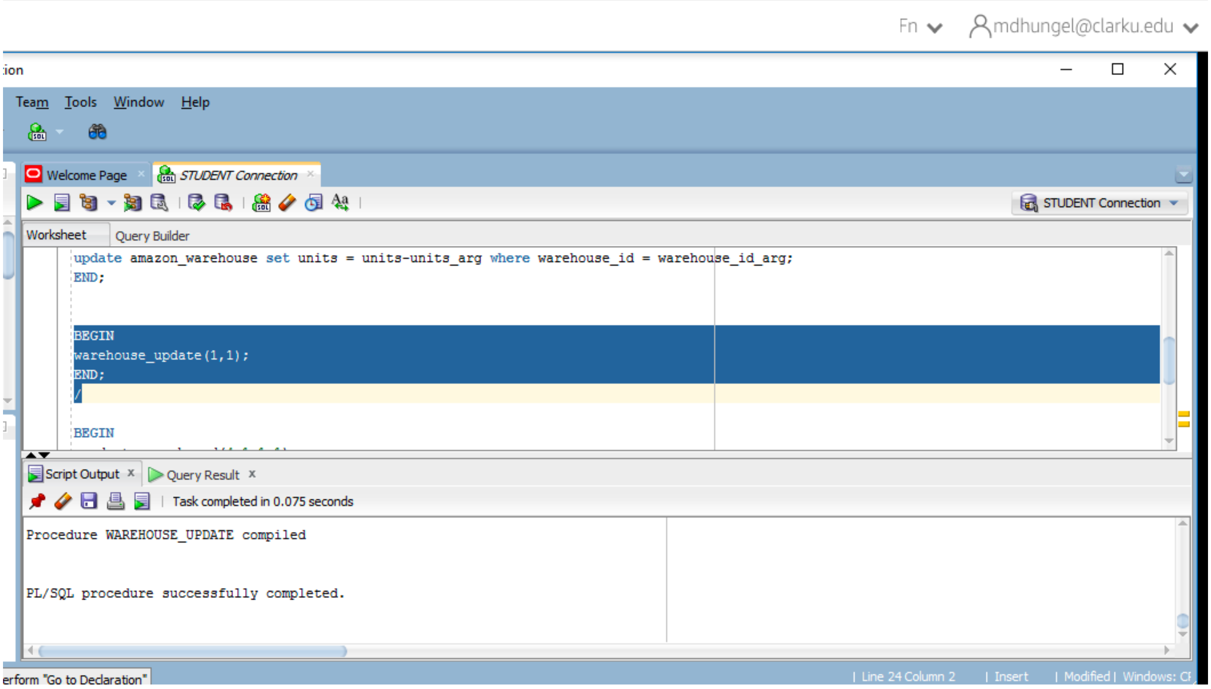


```
BEGIN
products_purchased(5,1,6,2);
END;
/
BEGIN
products_purchased(6,2,7,1);
END;
/
create or replace procedure warehouse_update(
```

Task completed in 0.102 seconds

```
PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.
```

Use of the parameterized stored procedure to update the units

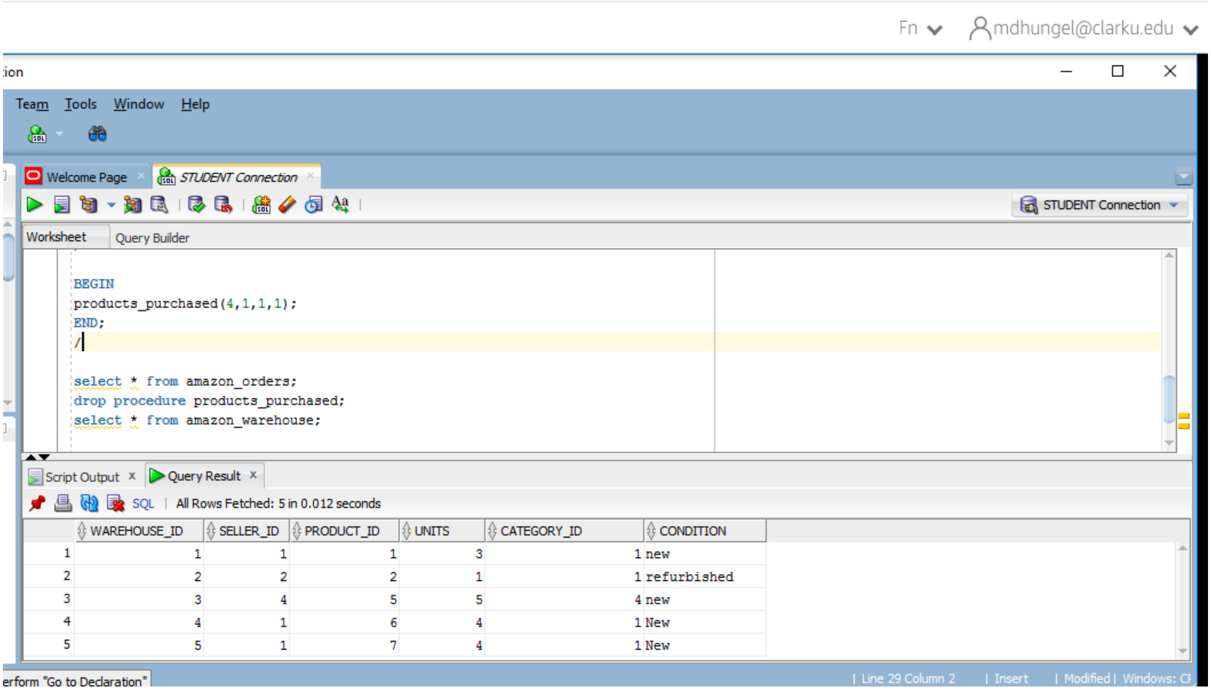


```
update amazon_warehouse set units = units-units_arg where warehouse_id = warehouse_id_arg;
END;


BEGIN
warehouse_update(1,1);
END;
/

BEGIN
```

Task completed in 0.075 seconds

```
Procedure WAREHOUSE_UPDATE compiled


PL/SQL procedure successfully completed.
```

As we can see before the invocation of the procedure the number of units is 3 for the product id 1 with warehouse id 1.

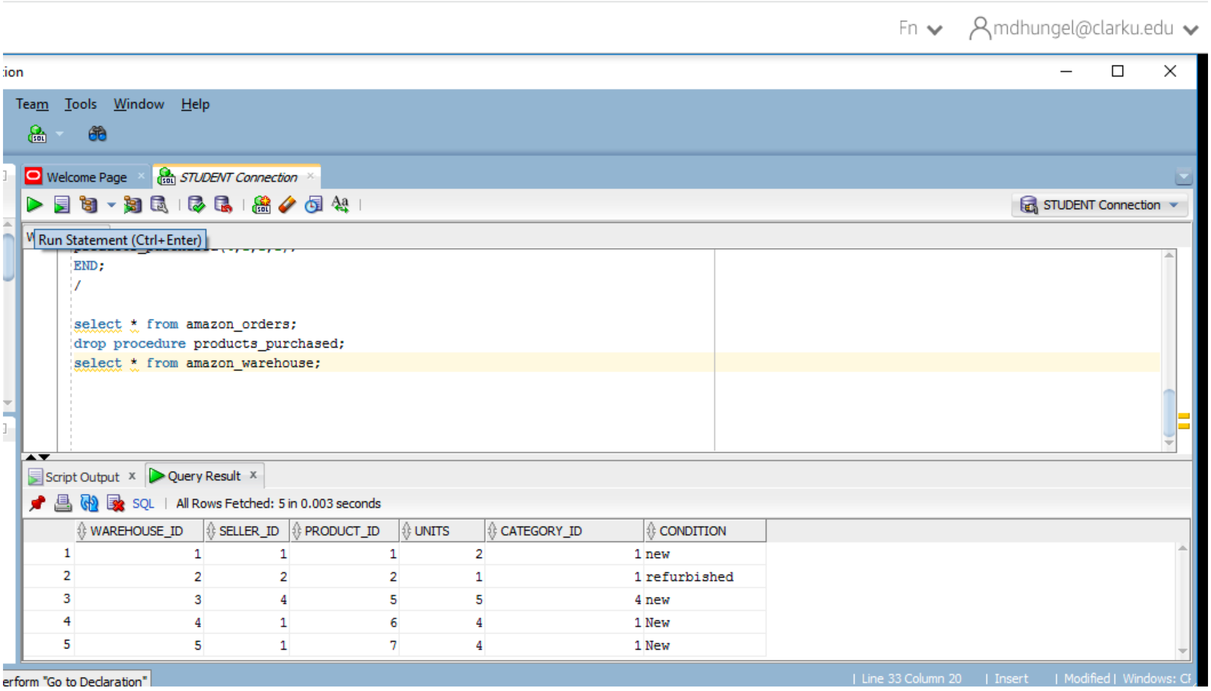


```
BEGIN
products_purchased(4,1,1,1);
END;
/

select * from amazon_orders;
drop procedure products_purchased;
select * from amazon_warehouse;
```

All Rows Fetched: 5 in 0.012 seconds

	WAREHOUSE_ID	SELLER_ID	PRODUCT_ID	UNITS	CATEGORY_ID	CONDITION
1	1	1	1	3	1	new
2	2	2	2	1	1	refurbished
3	3	4	5	5	4	new
4	4	1	6	4	1	New
5	5	1	7	4	1	New

As we can see after the invocation of the procedure the number of units is 2 for the product id 1 with warehouse id 1.



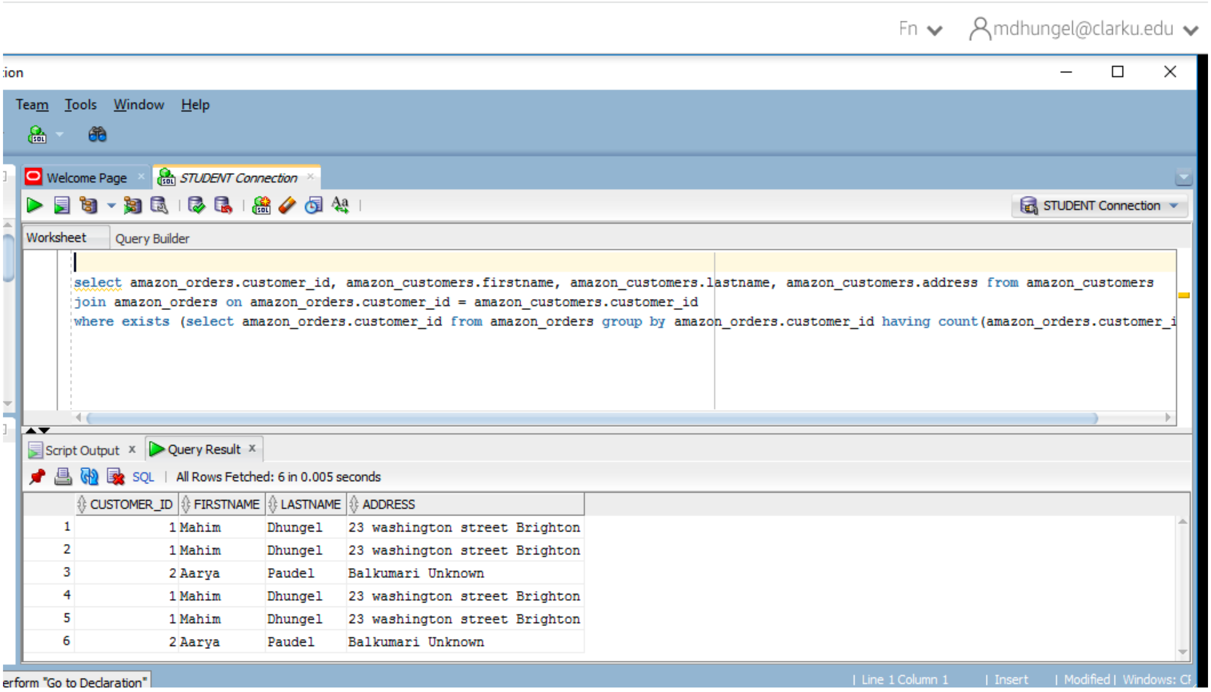
```
END;
/

select * from amazon_orders;
drop procedure products_purchased;
select * from amazon_warehouse;
```

All Rows Fetched: 5 in 0.003 seconds

	WAREHOUSE_ID	SELLER_ID	PRODUCT_ID	UNITS	CATEGORY_ID	CONDITION
1	1	1	1	2	1	new
2	2	2	2	1	1	refurbished
3	3	4	5	5	4	new
4	4	1	6	4	1	New
5	5	1	7	4	1	New

Query that the marketing department at Amazon wants to use to reach out to consumers who buy popular products. The department requests the names and addresses of all consumers who bought any product that was purchased by at least three different people.

(In my case I used 2 because the values aren't populated so much but the basic query remains the same. The query is available in the aspect folder of the zip file)



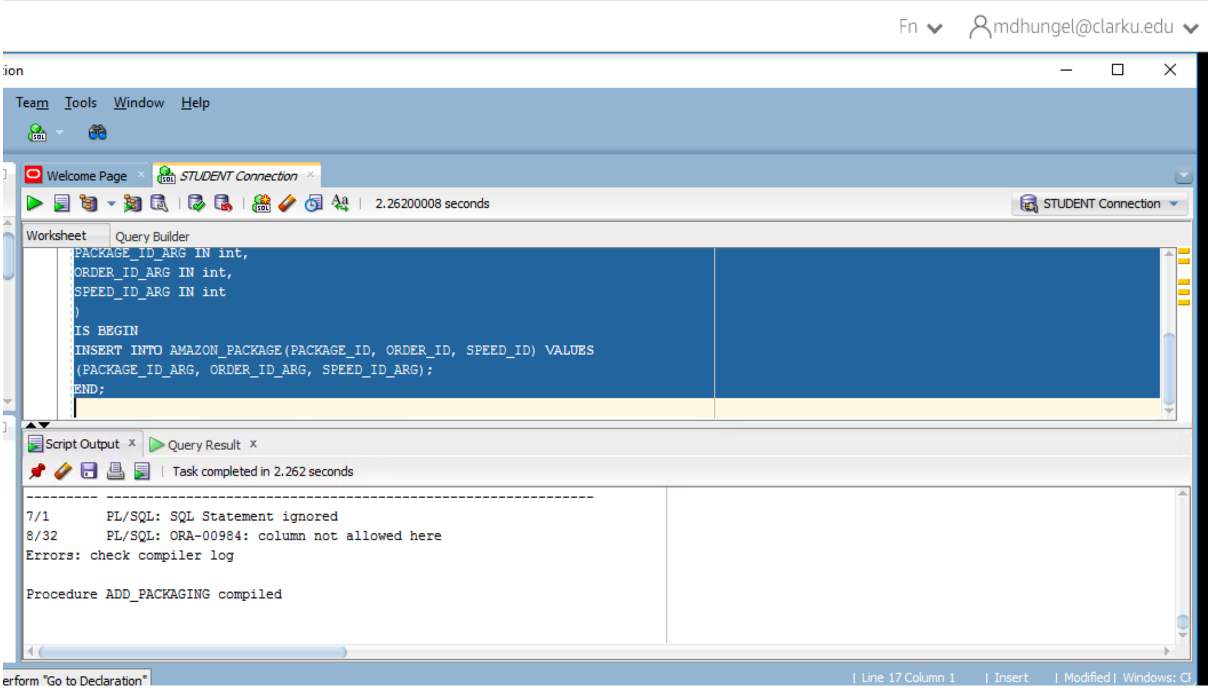
```
select amazon_orders.customer_id, amazon_customers.firstname, amazon_customers.lastname, amazon_customers.address from amazon_customers
join amazon_orders on amazon_orders.customer_id = amazon_customers.customer_id
where exists (select amazon_orders.customer_id from amazon_orders group by amazon_orders.customer_id having count(amazon_orders.customer_i
```

All Rows Fetched: 6 in 0.005 seconds

	CUSTOMER_ID	FIRSTNAME	LASTNAME	ADDRESS
1	1	Mahim	Dhungel	23 washington street Brighton
2	1	Mahim	Dhungel	23 washington street Brighton
3	2	Aarya	Paudel	Balkumari Unknown
4	1	Mahim	Dhungel	23 washington street Brighton
5	1	Mahim	Dhungel	23 washington street Brighton
6	2	Aarya	Paudel	Balkumari Unknown

Aspect 5: Product Shipment by Amazon.

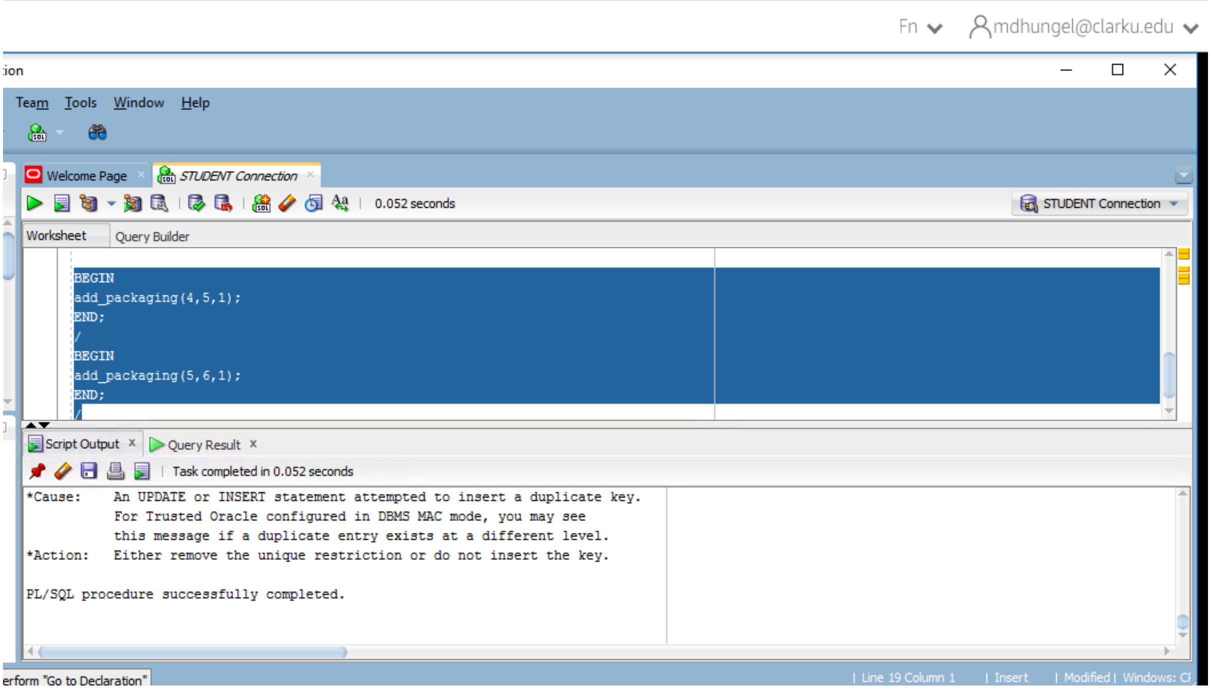
Creation of parameterized stored procedure that is used when Amazon ships any order.



```
PACKAGE_ID_ARG IN int,
ORDER_ID_ARG IN int,
SPEED_ID_ARG IN int
)
IS BEGIN
INSERT INTO AMAZON_PACKAGE(PACKAGE_ID, ORDER_ID, SPEED_ID) VALUES
(PACKAGE_ID_ARG, ORDER_ID_ARG, SPEED_ID_ARG);
END;
```

```
--------- -------------------------------------------------------------
7/1       PL/SQL: SQL Statement ignored
8/32      PL/SQL: ORA-00984: column not allowed here
Errors: check compiler log

Procedure ADD_PACKAGING compiled
```

Use of the parameterized stored procedure to ships the orders listed



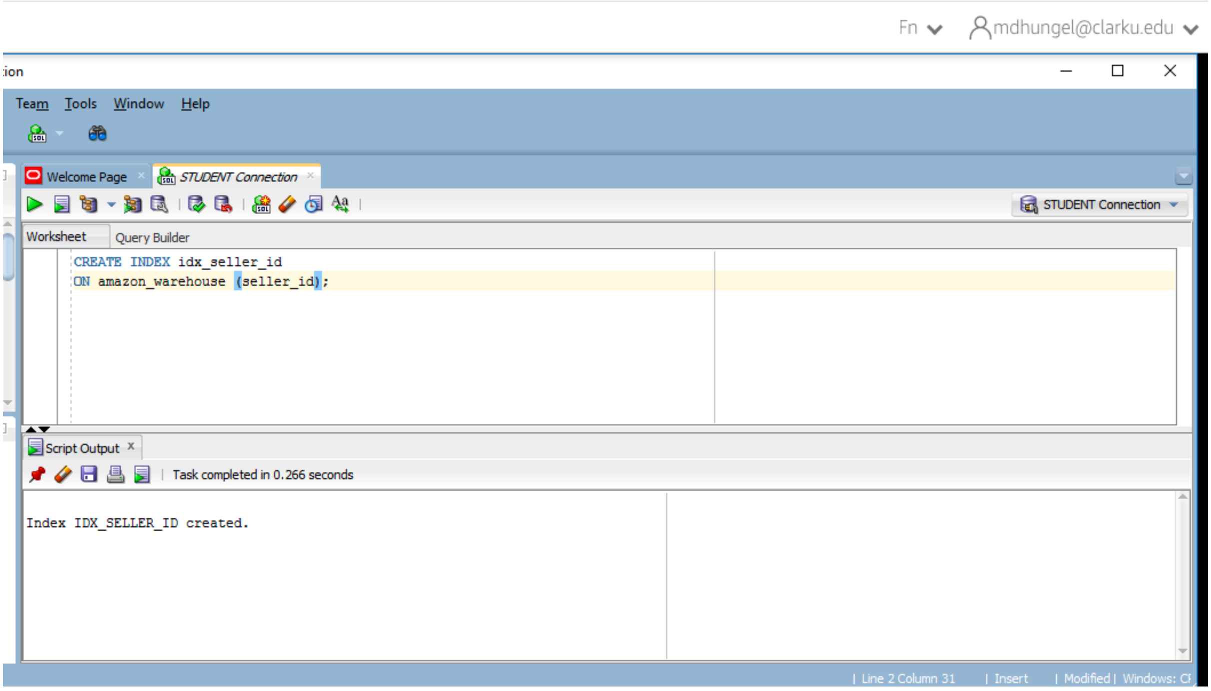
```
BEGIN
add_packaging(4,5,1);
END;
/
BEGIN
add_packaging(5,6,1);
END;
/
```

```
*Cause:    An UPDATE or INSERT statement attempted to insert a duplicate key.
           For Trusted Oracle configured in DBMS MAC mode, you may see
           this message if a duplicate entry exists at a different level.
*Action:   Either remove the unique restriction or do not insert the key.

PL/SQL procedure successfully completed.
```

Index.1 Justification and Creation:

Justification: Since this project is Amazon Fulfillment, the majority of the sellers will have their inventory on the amazon warehouse so it makes sense if the warehouse is indexed via the seller id for faster data retrieval.

Index created on Seller id of Amazon Warehouse
Since Inventory management in Amazon largely consists of sellers.



```
CREATE INDEX idx_seller_id
ON amazon_warehouse (seller_id);
```

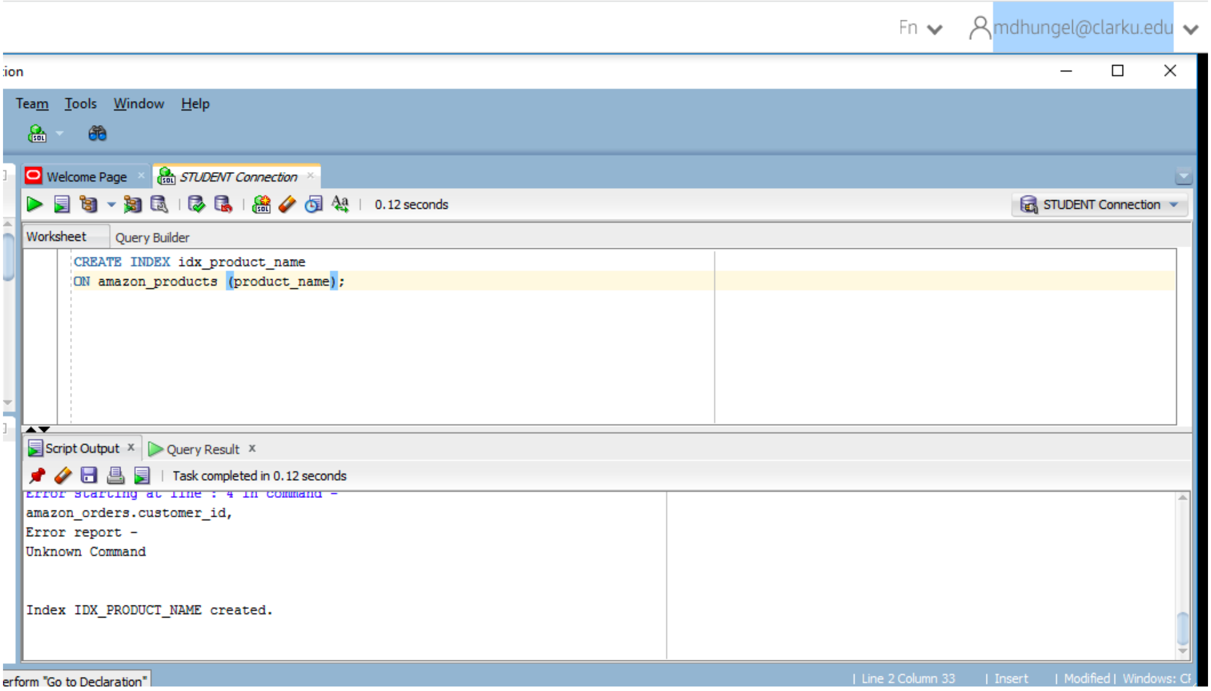
Task completed in 0.266 seconds

```
Index IDX_SELLER_ID created.
```

Index.2 Justification and Creation:

Justification: We all know by now that most customers search items by name and not by category or ids. Therefore creating an index on the product name on the product table is justified because it makes for a better customer experience as data is displayed faster helping with customer retention.

Index created on product name of product table



```
CREATE INDEX idx_product_name
ON amazon_products (product_name);
```

```
Error starting at line : 4 in command -
amazon_orders.customer_id,
Error report -
Unknown Command


Index IDX_PRODUCT_NAME created.
```