

# **E-commerce Marketplace Database Management System**

Abbas Ibrahim

With this project, I set out on the thrilling adventure of creating, putting into practice, and making use of our very own database system, which was motivated by the complexities of selling on Amazon, an e-commerce platform. By carefully examining the theoretical ideas, my goal is to build a solid database that contains all the essential features required to facilitate the selling process on an ever-changing internet marketplace. The project is guided by a set of use cases that delineate the essential functions our database needs to facilitate, ranging from sellers' generation of new products to Amazon's fulfilment of those products.

#### **THE NAMES OF ALL ENTITIES:**

- Products
- Category
- Warehouse
- Seller
- Accounts
- Product\_Order
- Shipping

#### **RELATIONSHIP BETWEEN THE ENTITIES:**

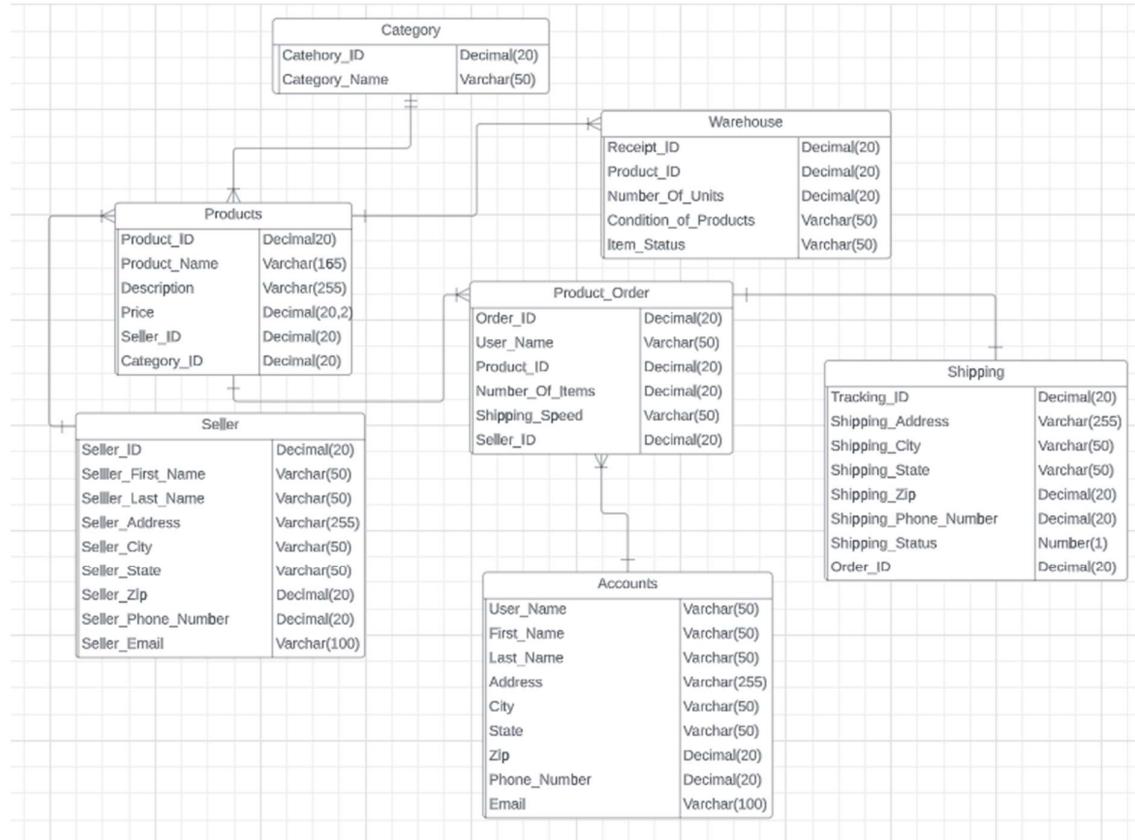
- Products to Category (one to many relationship)
- Category to Products (one to one relationship)
- Products to Seller (one to many relationship)
- Seller to Products (one to one relationship)
- Products to Warehouse (one to one relationship)
- Warehouse to Products (one to many relationship)
- Products to Product\_Order (one to one relationship)
- Product\_Order to Products (one to many relationship)
- Accounts to Product\_Order (one to one relationship)
- Product\_Order to Accounts (one to many relationship)
- Product\_Order to Shipping (one to one relationship)
- Shipping to Product\_Order (one to one relationship)

#### **BUSINESS RULES:**

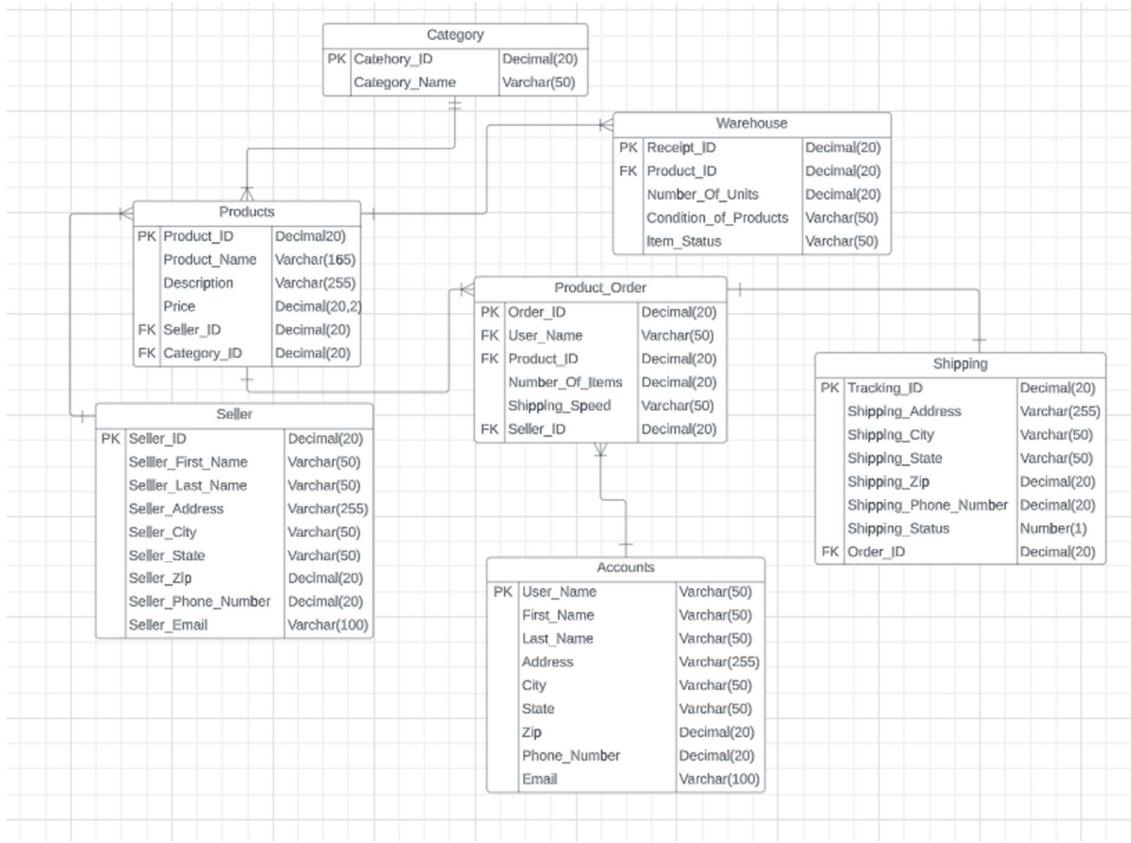
1. One or many products can fall under one category.
2. At least one product should fall under one category.
3. Each product should be listed uniquely (by using a unique identifier).
4. Sellers can list their products in relevant categories.
5. Many products can have one seller.
6. All products should have some quantity mentioned in the warehouse and their status whether the particulars stock is received or not and how many.
7. Whenever the seller ships the product the number units must be updated.

8. A seller must be associated with each product.
9. Per order should have all the product details, the number of quantities selected by the customer, shipping speed.
10. Many products can be placed in one order.
11. One order should have at least one product in it.
12. Each order should have a shipping or tracking ID.
13. Each shipping ID should have an updated shipping status.
14. User account should have all the necessary details filled, that is it cannot hold null values for the mandatory columns like username, Email, Phone number, address.
15. Orders can be placed only if the users have an account or registered with an account.
16. Products can be added only if it has a seller.

### CONCEPTUAL DATA MODEL:



## LOGICAL DATA MODEL:



## ASPECT 1: NEW PRODUCT CREATED BY SELLER

- Creating the tables, constraints, and data needed to support new products as described in the use case.

The screenshot shows the Oracle SQL Developer interface with a Worksheet tab active. The code area contains the following SQL statements:

```
CREATE TABLE Products (Product_ID DECIMAL(20) PRIMARY KEY,
Product_Name VARCHAR(165),
Description VARCHAR(255),
Price DECIMAL(20,2),
Seller_ID DECIMAL(20),
Category_ID DECIMAL(20)
);

CREATE TABLE Category (Category_ID DECIMAL(20) PRIMARY KEY,
Category_Name VARCHAR(50)
);

CREATE TABLE Seller (Seller_ID DECIMAL(20) PRIMARY KEY,
Seller_First_Name VARCHAR(50),
Seller_Last_Name VARCHAR(50),
Seller_Address VARCHAR(255),
Seller_City VARCHAR(50),
Seller_State VARCHAR(50),
Seller_Zip DECIMAL(20),
Seller_Phone_Number DECIMAL(20),
Seller_Email VARCHAR(100)
);
```

The Script Output window shows the results of the table creations:

```
Table PRODUCTS created.

Table CATEGORY created.

Table SELLER created.
```

At the bottom right, status indicators show Line 23 Column 1, Insert, Modified, and Windows: C.

The screenshot shows the Oracle SQL Developer interface with a Worksheet tab active. The code area contains the following SQL statements:

```
ALTER TABLE Products
ADD CONSTRAINT Seller_ID_FK
FOREIGN KEY(Seller_ID) REFERENCES Seller(Seller_ID);

ALTER TABLE Products
ADD CONSTRAINT Category_ID_FK
FOREIGN KEY(Category_ID) REFERENCES Category(Category_ID);
```

The Script Output window shows the results of the table alterations:

```
Table PRODUCTS altered.

Table PRODUCTS altered.
```

At the bottom right, status indicators show Line 69 Column 1, Insert, Modified, and Windows: C.

- b. Developing a parameterized stored procedure that is used when a seller needs to add any new product.

```

CREATE OR REPLACE PROCEDURE ADD_CATEGORY(
    Category_ID_arg IN DECIMAL,
    Category_Name_arg IN VARCHAR)
IS
BEGIN
  INSERT INTO Category(Category_ID,
    Category_Name)
  VALUES(Category_ID_arg,
    Category_Name_arg);
END;

```

Procedure ADD\_CATEGORY compiled

```

CREATE OR REPLACE PROCEDURE ADD_SELLER(Seller_ID_arg IN DECIMAL,
    Seller_First_Name_arg IN VARCHAR,
    Seller_Last_Name_arg IN VARCHAR,
    Seller_Address_arg IN VARCHAR,
    Seller_City_arg IN VARCHAR,
    Seller_State_arg IN VARCHAR,
    Seller_Zip_arg IN DECIMAL,
    Seller_Phone_Number_arg IN DECIMAL,
    Seller_Email_arg IN VARCHAR)
IS
BEGIN
  INSERT INTO Seller(Seller_ID,
    Seller_First_Name,
    Seller_Last_Name,
    Seller_Address,
    Seller_City,
    Seller_State,
    Seller_Zip,
    Seller_Phone_Number,
    Seller_Email)
  VALUES(Seller_ID_arg,
    Seller_First_Name_arg,
    Seller_Last_Name_arg,
    Seller_Address_arg,
    Seller_City_arg,
    Seller_State_arg,
    Seller_Zip_arg,
    Seller_Phone_Number_arg,
    Seller_Email_arg);
END;

```

Procedure ADD\_SELLER compiled

CREATE OR REPLACE PROCEDURE ADD\_PRODUCT(  
Product\_ID\_arg IN DECIMAL,  
Product\_Name\_arg IN VARCHAR,  
Description\_arg IN VARCHAR,  
Price\_arg IN DECIMAL,  
Seller\_ID\_arg IN DECIMAL,  
Category\_ID\_arg IN DECIMAL)  
IS BEGIN  
INSERT INTO Products(Product\_ID, Product\_Name, Description, Price, Seller\_ID, Category\_ID)  
VALUES (Product\_ID\_arg, Product\_Name\_arg, Description\_arg, Price\_arg, Seller\_ID\_arg, Category\_ID\_arg);  
END;

Procedure ADD\_PRODUCT compiled

| Line 136 Column 9 | Insert | Modified | Windows: C|

```
BEGIN ADD_CATEGORY(1, 'Computers');  
END;  
/  
BEGIN ADD_CATEGORY(2, 'Electronics');  
END;  
/  
BEGIN ADD_CATEGORY(3, 'Appliances');  
END;  
/  
BEGIN ADD_CATEGORY(4, 'Computers');  
END;  
/  
BEGIN ADD_CATEGORY(5, 'Shoes');  
END;  
/  
PL/SQL procedure successfully completed.  
PL/SQL procedure successfully completed.  
PL/SQL procedure successfully completed.  
PL/SQL procedure successfully completed.  
PL/SQL procedure successfully completed.
```

| Line 155 Column 2 | Insert | Modified | Windows: C|

```
BEGIN ADD_CATEGORY(6, 'Clothing');  
END;  
/  
BEGIN ADD_CATEGORY(7, 'Jewelry');  
END;  
/  
BEGIN ADD_CATEGORY(8, 'Plants');  
END;  
/  
BEGIN ADD_CATEGORY(9, 'Games');  
END;  
/  
BEGIN ADD_CATEGORY(10, 'Toys');  
END;  
/  
PL/SQL procedure successfully completed.  
PL/SQL procedure successfully completed.  
PL/SQL procedure successfully completed.  
PL/SQL procedure successfully completed.
```

| Line 162 Column 2 | Insert | Modified | Windows: C|

Welcome Page | VarsaSriMadhusudhanOracle | Worksheet | Query Builder

```
select * from category;
```

Script Output | Query Result | All Rows Fetched: 10 in 0.072 seconds

CATEGORY_ID	CATEGORY_NAME
1	Computers
2	Electronics
3	Appliances
4	Computers
5	Shoes
6	Clothing
7	Jewelry
8	Plants
9	Games
10	Toys

SQL History | Line 174 Column 1 | Insert | Modified | Windows: C

Welcome Page | VarsaSriMadhusudhanOracle | Worksheet | Query Builder

```
BEGIN ADD_SELLER(1000, 'John', 'Smith', 'A Road', 'Worcester', 'Massachusetts', '161310', '7894561230', 'John@gmail.com');
/
BEGIN ADD_SELLER(1001, 'Jerry', 'Wahn', 'B Road', 'Beverly Hills', 'California', '191310', '4561237890', 'Jerry@gmail.com');
/
BEGIN ADD_SELLER(1002, 'Ravi', 'Kumar', 'C Road', 'Boston', 'Massachusetts', '181310', '8520741963', 'Ravi@gmail.com');
/
BEGIN ADD_SELLER(1003, 'Mac', 'Joey', 'D Road', 'Dayton', 'Ohio', '131310', '7531598520', 'Mac@gmail.com');
/
BEGIN ADD_SELLER(1004, 'Jennifer', 'Cooper', 'E Road', 'Buffalo', 'New York', '141310', '1234567890', 'Cooper@gmail.com');
/

```

Script Output | Query Result | Task completed in 0.299 seconds

PL/SQL procedure successfully completed.

SQL History | Line 175 Column 33 | Insert | Modified | Windows: C

Welcome Page | VarsaSriMadhusudhanOracle | Worksheet | Query Builder

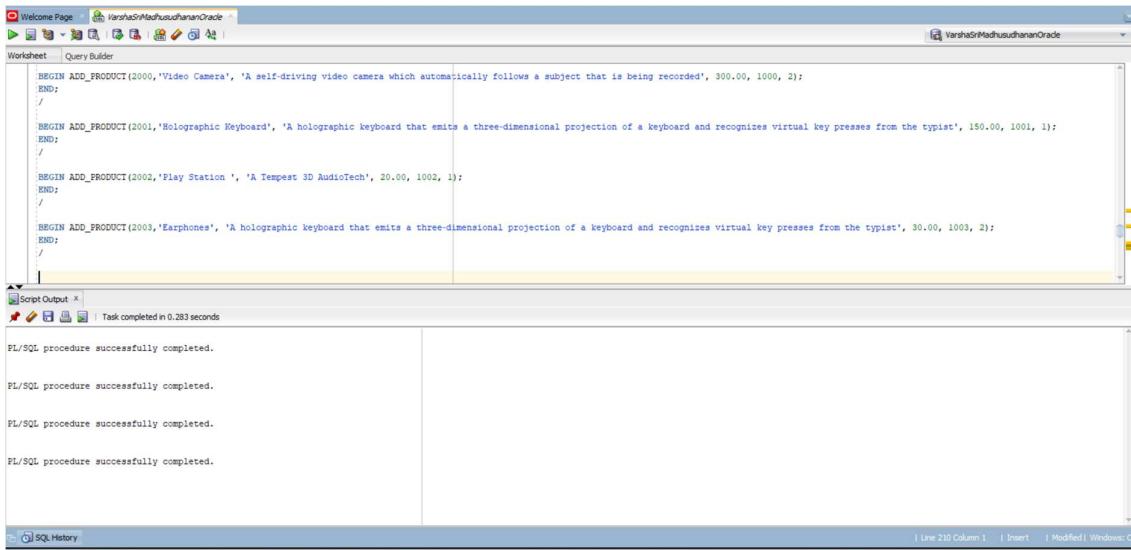
```
SELECT * FROM SELLER;
```

Script Output | Query Result | All Rows Fetched: 5 in 0.032 seconds

SELLER_ID	SELLER_FIRST_NAME	SELLER_LAST_NAME	SELLER_ADDRESS	SELLER_CITY	SELLER_STATE	SELLER_ZIP	SELLER_PHONE_NUMBER	SELLER_EMAIL	
1	1000	John	Smith	A Road	Worcester	Massachusetts	161310	7894561230	John@gmail.com
2	1001	Jerry	Wahn	B Road	Beverly Hills	California	191310	4561237890	Jerry@gmail.com
3	1002	Ravi	Kumar	C Road	Boston	Massachusetts	181310	8520741963	Ravi@gmail.com
4	1003	Mac	Joey	D Road	Dayton	Ohio	131310	7531598520	Mac@gmail.com
5	1004	Jennifer	Cooper	E Road	Buffalo	New York	141310	1234567890	Cooper@gmail.com

SQL History | Line 191 Column 22 | Insert | Modified | Windows: C

- c. A seller adds two new products. The first is a self-driving video camera which automatically follows a subject that is being recorded. The second is a holographic keyboard that emits a three-dimensional projection of a keyboard and recognizes virtual key presses from the typist. Invoking the stored procedure twice to add these products, which have at a minimum a name, description, price, and category in its attributes.



```

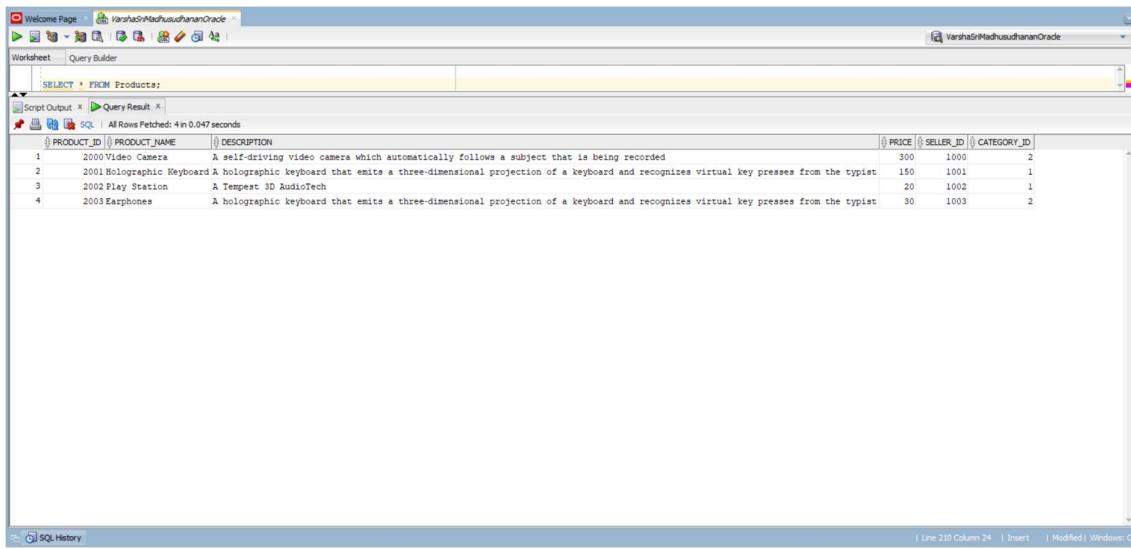
BEGIN ADD_PRODUCT(2000,'Video Camera', 'A self-driving video camera which automatically follows a subject that is being recorded', 300.00, 1000, 2);
END;
/
BEGIN ADD_PRODUCT(2001,'Holographic Keyboard', 'A holographic keyboard that emits a three-dimensional projection of a keyboard and recognizes virtual key presses from the typist', 150.00, 1001, 1);
END;
/
BEGIN ADD_PRODUCT(2002,'Play Station ', 'A Tempest 3D AudioTech', 20.00, 1002, 1);
END;
/
BEGIN ADD_PRODUCT(2003,'Earphones', 'A holographic keyboard that emits a three-dimensional projection of a keyboard and recognizes virtual key presses from the typist', 30.00, 1003, 2);
END;
/

```

Script Output: X | Task completed in 0.283 seconds

PL/SQL procedure successfully completed.  
PL/SQL procedure successfully completed.  
PL/SQL procedure successfully completed.  
PL/SQL procedure successfully completed.

SQL History | Line 210 Column 1 | Insert | Modified | Windows: O



```

SELECT * FROM Products;

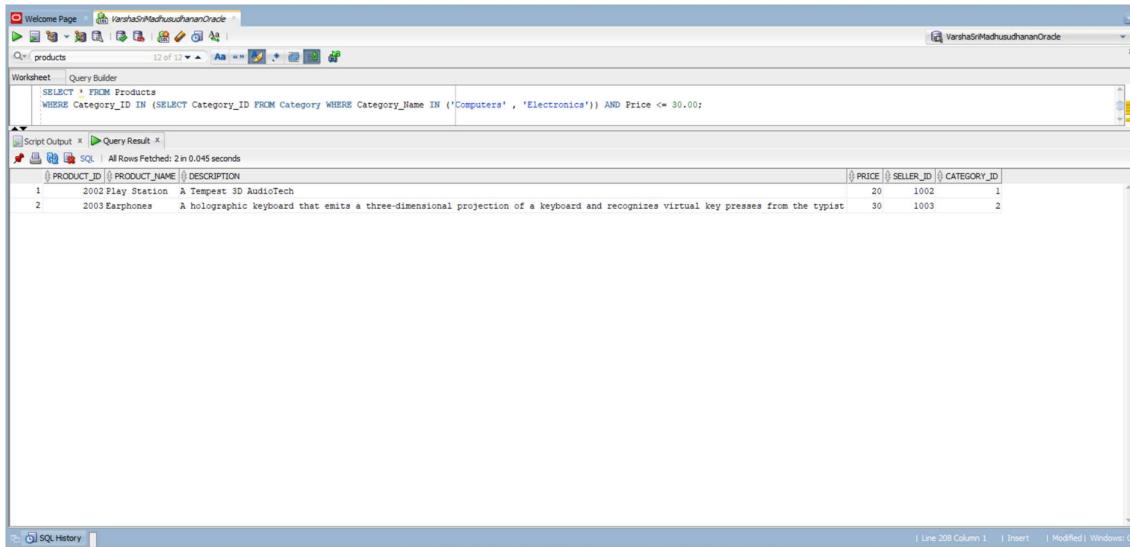
```

Script Output: X | Query Result: X | All Rows Fetched: 4 in 0.047 seconds

PRODUCT_ID	PRODUCT_NAME	DESCRIPTION	PRICE	SELLER_ID	CATEGORY_ID
1	2000 Video Camera	A self-driving video camera which automatically follows a subject that is being recorded	300	1000	2
2	2001 Holographic Keyboard	A holographic keyboard that emits a three-dimensional projection of a keyboard and recognizes virtual key presses from the typist	150	1001	1
3	2002 Play Station	A Tempest 3D AudioTech	20	1002	1
4	2003 Earphones	A holographic keyboard that emits a three-dimensional projection of a keyboard and recognizes virtual key presses from the typist	30	1003	2

SQL History | Line 210 Column 24 | Insert | Modified | Windows: O

- d. A seller is considering developing a new electronic product and requests a list of existing products in the “Computers” or “Electronics” categories that cost \$30 or less.



The screenshot shows the Oracle SQL Developer interface. The query window contains the following SQL code:

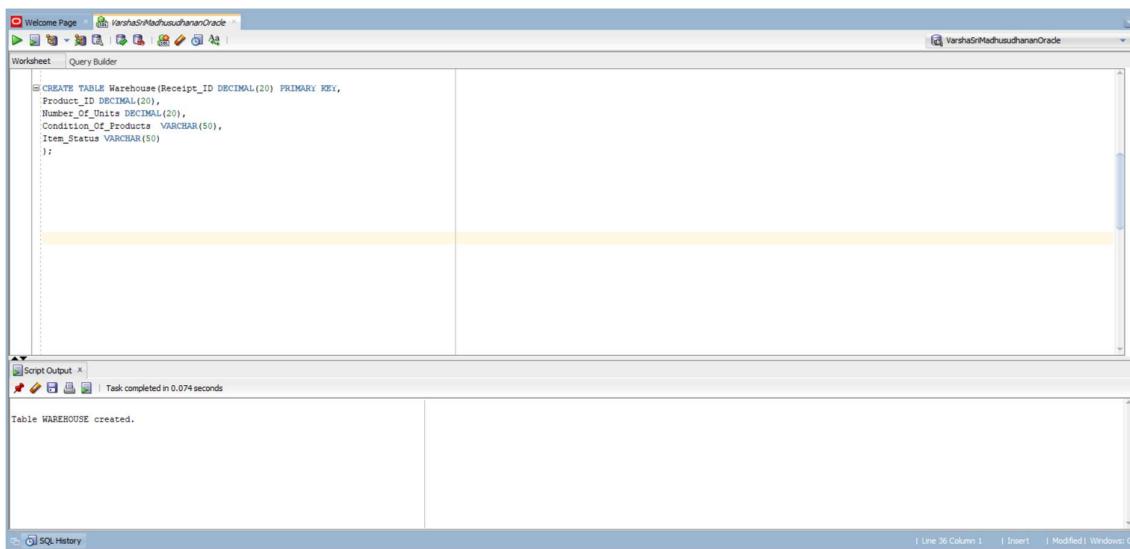
```
SELECT * FROM Products
WHERE Category_ID IN (SELECT Category_ID FROM Category WHERE Category_Name IN ('Computers', 'Electronics')) AND Price <= 30.00;
```

The results pane displays the following data:

PRODUCT_ID	PRODUCT_NAME	DESCRIPTION	PRICE	SELLER_ID	CATEGORY_ID
1	2002 Play Station	A Tempest 3D AudioTech	20	1002	1
2	2003 Earphones	A holographic keyboard that emits a three-dimensional projection of a keyboard and recognizes virtual key presses from the typist	30	1003	2

## ASPECT 2: AMAZON RECEIPT OF PRODUCT FROM SELLER

- a. Creating the tables, constraints, and data needed to support product delivery.



The screenshot shows the Oracle SQL Developer interface. The query window contains the following SQL code for creating a table:

```
CREATE TABLE Warehouse(Receipt_ID DECIMAL(20) PRIMARY KEY,
Product_ID DECIMAL(20),
Number_Of_Units DECIMAL(20),
Condition_Of_Products VARCHAR(50),
Item_Status VARCHAR(50)
);
```

The results pane shows the message: "Table WAREHOUSE created." and indicates the task completed in 0.074 seconds.

The screenshot shows the Oracle SQL Developer interface. In the top-left pane, there is a script editor window titled 'Worksheet' containing the following SQL code:

```
ALTER TABLE Warehouse
ADD CONSTRAINT Product_ID_FK
FOREIGN KEY (Product_ID) REFERENCES Products (Product_ID);
```

In the bottom-left pane, the 'Script Output' window displays the result of the execution:

```
Table WAREHOUSE altered.
```

The status bar at the bottom right indicates 'Line 74 Column 1'.

- b. Developing a parameterized stored procedure that is used when any seller delivers any product to Amazon's warehouse.

The screenshot shows the Oracle SQL Developer interface. In the top-left pane, there is a script editor window titled 'Worksheet' containing the following PL/SQL code:

```
CREATE OR REPLACE PROCEDURE ADD_PRODUCT_WAREHOUSE(
    Receipt_ID_arg IN DECIMAL,
    Product_ID_arg IN DECIMAL,
    Number_of_Units_arg IN DECIMAL,
    Condition_of_Products_arg IN VARCHAR,
    Item_Status_arg IN VARCHAR)
IS BEGIN
    INSERT INTO Warehouse(Receipt_ID, Product_ID, Number_of_Units, Condition_of_Products, Item_Status)
    VALUES(Receipt_ID_arg, Product_ID_arg, Number_of_Units_arg, Condition_of_Products_arg, Item_Status_arg);
END;
```

In the bottom-left pane, the 'Script Output' window displays the result of the compilation:

```
Procedure ADD_PRODUCT_WAREHOUSE compiled
```

The status bar at the bottom right indicates 'Line 213 Column 26'.

- c. A seller delivers four of each of the two new products added in Aspect 1 (the self-driving video camera and the holographic keyboard). Invoking the stored procedure twice to update the inventory of these products for a seller.

```

Welcome Page  VarshaSriMadhusudhanOrade
Worksheet  Query Builder
BEGIN ADD_PRODUCT_WAREHOUSE(3000,2000, 3, 'New', 'Delivered');
END;
/
BEGIN ADD_PRODUCT_WAREHOUSE(3001, 2001, 5, 'New', 'Delivered');
END;
/
BEGIN ADD_PRODUCT_WAREHOUSE(3002, 2002, 11, 'Used', 'Delivered');
END;
/
BEGIN ADD_PRODUCT_WAREHOUSE(3003, 2003, 15, 'Used', 'Delivered');
END;
/

```

Script Output: Task completed in 0.256 seconds

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

SQL History

```

Welcome Page  VarshaSriMadhusudhanOrade
Worksheet  Query Result
SELECT * FROM Warehouse;

```

Script Output: All Rows Fetched: 4 in 0.036 seconds

RECEIPT_ID	PRODUCT_ID	NUMBER_OF_UNITS	CONDITION	ITEM_STATUS
1	3000	2000	3 New	Delivered
2	3001	2001	5 New	Delivered
3	3002	2002	11 Used	Delivered
4	3003	2003	15 Used	Delivered

SQL History

- d. The seller from b above requests a listing of all of its products that have an inventory of 11 or less. (the self-driving video camera and holographic keyboard are included in the listed).

The screenshot shows the Oracle SQL Developer interface. The top pane displays a SQL query:

```
SELECT a.Seller_ID, w.Product_ID, p.Product_Name, p.Description, p.Price, p.Category_ID, w.Number_of_Units FROM Products p
INNER JOIN Warehouse w ON w.Product_ID = p.Product_ID
INNER JOIN Seller s ON s.Seller_ID = p.Seller_ID
WHERE Number_of_Units <= 11;
```

The bottom pane shows the results of the query:

SELLER_ID	PRODUCT_ID	PRODUCT_NAME	DESCRIPTION	PRICE	CATEGORY_ID	NUMBER_OF_UNITS
1	1000	2000 Video Camera	A self-driving video camera which automatically follows a subject that is being recorded	300	2	3
2	1001	2001 Holographic Keyboard	A holographic keyboard that emits a three-dimensional projection of a keyboard and recognizes virtual key presses from the typist	150	1	5
3	1002	2002 Play Station	A Tempest 3D AudioTech	20	1	11

### ASPECT 3: NEW CONSUMER ACCOUNT

- a. Creating the tables, constraints, and data needed to support customer accounts.

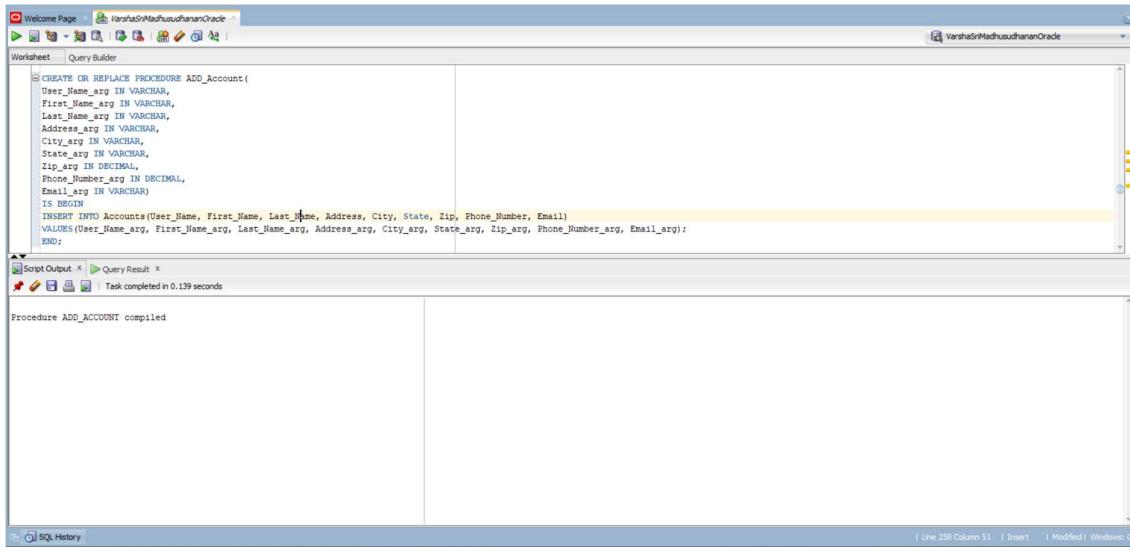
The screenshot shows the Oracle SQL Developer interface. The top pane displays a SQL script for creating a table:

```
CREATE TABLE Accounts(User_Name VARCHAR(50) PRIMARY KEY,
First_Name VARCHAR(50),
Last_Name VARCHAR(50),
Address VARCHAR(255),
City VARCHAR(50),
State VARCHAR(50),
Zip DECIMAL(120),
Phone_Number DECIMAL(20),
Email VARCHAR(100)
);
```

The bottom pane shows the results of the table creation:

Table ACCOUNTS created.

- b. Developing a parameterized stored procedure that is used when any new customer signs up for a new account on Amazon.

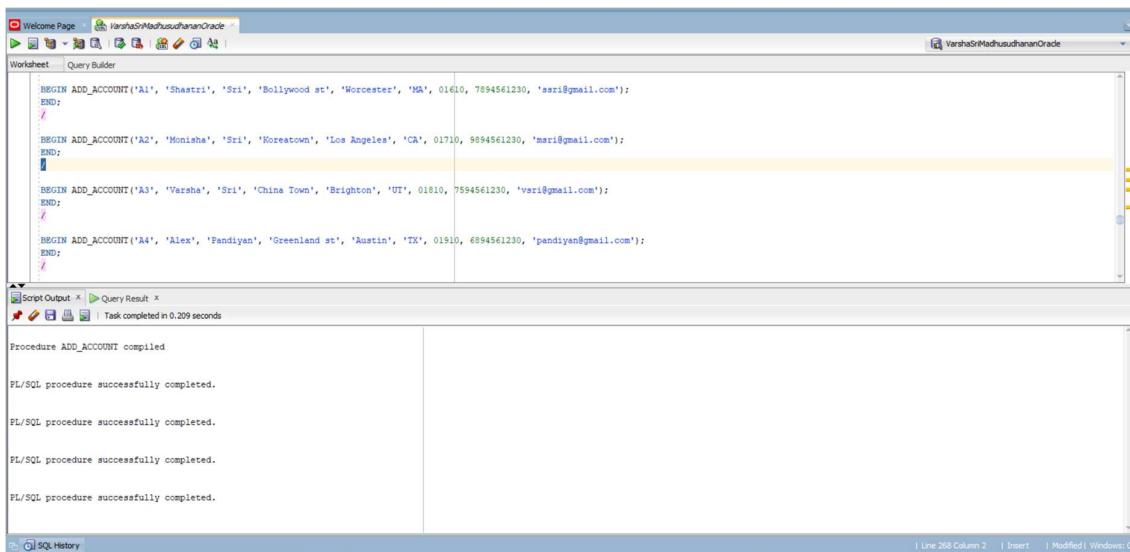


```

CREATE OR REPLACE PROCEDURE ADD_Account(
    User_Name_arg IN VARCHAR,
    First_Name_arg IN VARCHAR,
    Last_Name_arg IN VARCHAR,
    Address_arg IN VARCHAR,
    City_arg IN VARCHAR,
    State_arg IN VARCHAR,
    Zip_arg IN DECIMAL,
    Phone_Number_arg IN DECIMAL,
    Email_arg IN VARCHAR)
IS BEGIN
    INSERT INTO Accounts(User_Name, First_Name, Last_Name, Address, City, State, Zip, Phone_Number, Email)
    VALUES(User_Name_arg, First_Name_arg, Last_Name_arg, Address_arg, City_arg, State_arg, Zip_arg, Phone_Number_arg, Email_arg);
END;
  
```

Procedure ADD\_ACCOUNT compiled

- c. You and your facilitator sign up for new accounts on Amazon. Invoking the stored procedure twice to add me and my facilitator as customers.



```

BEGIN ADD_ACCOUNT('A1', 'Shashri', 'Sri', 'Bollywood st', 'Worcester', 'MA', 01610, 7894561230, 'sri@gmail.com');
END;
/

BEGIN ADD_ACCOUNT('A2', 'Monisha', 'Sri', 'Koreatown', 'Los Angeles', 'CA', 01710, 9894561230, 'msri@gmail.com');
END;
/

BEGIN ADD_ACCOUNT('A3', 'Varsha', 'Sri', 'China Town', 'Brighton', 'UT', 01810, 7594561230, 'varsi@gmail.com');
END;
/

BEGIN ADD_ACCOUNT('A4', 'Alex', 'Pandian', 'Greenland st', 'Austin', 'TX', 01910, 6894561230, 'pandian@gmail.com');
END;
  
```

Procedure ADD\_ACCOUNT compiled

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

Welcome Page | VarshaSriMadhusudhananOracle

Worksheet | Query Builder

```
BEGIN ADD_ACCOUNT('A5', 'Markin', 'Luther', 'Woodland st', 'Chicago', 'MI', 01110, 2894561230, 'luther@gmail.com');
END;
/
BEGIN ADD_ACCOUNT('A6', 'Penny', 'Sri', 'Mason st', 'Provincetown', 'RI', 01210, 8894561230, 'perri@gmail.com');
END;
/

```

Script Output | Query Result | Task completed in 0.126 seconds

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

SQL History | Line 280 Column 2 | Insert | Modified | Windows

Welcome Page | VarshaSriMadhusudhananOracle

Worksheet | Query Builder

```
SELECT * FROM Accounts;
```

Script Output | Query Result | All Rows Fetched: 6 in 0.041 seconds

	USER_NAME	FIRST_NAME	LAST_NAME	ADDRESS	CITY	STATE	ZIP	PHONE_NUMBER	EMAIL
1	A1	Shastri	Sri	Bollywood st	Worcester	MA	1610	7894561230	sri@gmail.com
2	A2	Monisha	Sri	Koreatown	Los Angeles	CA	1710	9894561230	moni@gmail.com
3	A3	Varsha	Sri	China Town	Brighton	UT	1810	7594561230	vrsi@gmail.com
4	A4	Alex	Pandian	Greenland st	Austin	TX	1910	6694561230	pandian@gmail.com
5	A5	Markin	Luther	Woodland st	Chicago	MI	1110	2894561230	luther@gmail.com
6	A6	Penny	Sri	Mason st	Provincetown	RI	1210	8894561230	perri@gmail.com

SQL History | Line 285 Column 1 | Insert | Modified | Windows

- d. For research purposes, Amazon requests the last names of consumers where there are at least 4 accounts associated with the last name. Amazon would like to see the actual number of accounts associated with those last names. Developing and executing a single query that helps the above request.

The screenshot shows the Oracle SQL Developer interface. In the 'Worksheet' tab, a query is run:

```
SELECT COUNT(Last_Name) AS Last_Name_Sri FROM Accounts
WHERE Last_Name = 'Sri';
```

The results are displayed in the 'Query Result' tab:

Last_Name_Sri
4

Below the results, the status bar indicates: All Rows Fetched: 1 in 0.029 seconds.

## ASPECT 4: PRODUCT PURCHASE BY CONSUMER

- a. Creating the tables, constraints, and data needed to support product purchases.

The screenshot shows the Oracle SQL Developer interface. In the 'Worksheet' tab, a CREATE TABLE statement is run:

```
= CREATE TABLE Product_Order(Order_ID DECIMAL(20) PRIMARY KEY,
User_Name VARCHAR(50),
Product_ID DECIMAL(20),
Number_Of_Items DECIMAL(20),
Shipping_Speed VARCHAR(50),
Seller_ID DECIMAL(20)
);
```

The results are displayed in the 'Script Output' tab:

Table PRODUCT\_ORDER created.

Below the results, the status bar indicates: Task completed in 0.082 seconds.

```

ALTER TABLE Product_Order
ADD CONSTRAINT Product_Order_Product_ID_FK
FOREIGN KEY(Product_ID) REFERENCES Products(Product_ID);

ALTER TABLE Product_Order
ADD CONSTRAINT Product_Order_Seller_ID_FK
FOREIGN KEY(Seller_ID) REFERENCES Seller(Seller_ID);

ALTER TABLE Product_Order
ADD CONSTRAINT User_Name_Order_FK
FOREIGN KEY(User_Name) REFERENCES Accounts(User_Name);

```

Script Output: Task completed in 0.196 seconds

Table PRODUCT\_ORDER altered.  
Table PRODUCT\_ORDER altered.  
Table PRODUCT\_ORDER altered.

- b. Developing a parameterized stored procedure that is used when any customer purchases any product.

```

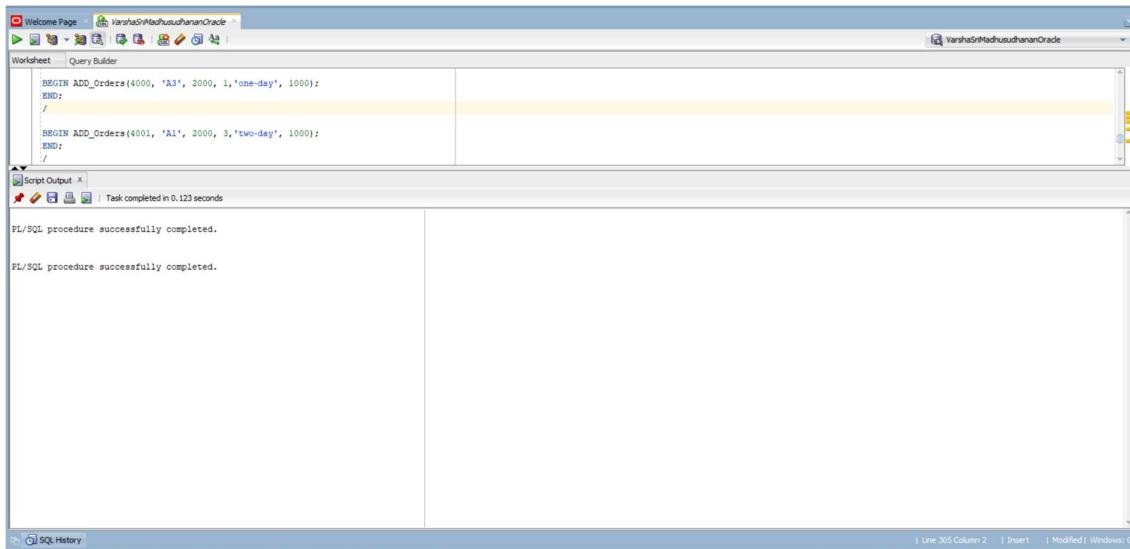
CREATE OR REPLACE PROCEDURE ADD_Orders(
    Order_ID_arg IN DECIMAL,
    User_Name_arg IN VARCHAR,
    Product_id_arg IN DECIMAL,
    Number_of_Items_arg IN DECIMAL,
    Shipping_Speed_arg IN VARCHAR,
    Seller_ID_arg IN DECIMAL)
IS BEGIN
    INSERT INTO Product_Order(Order_ID, User_Name, Product_id, Number_of_Items, Shipping_Speed, Seller_ID)
    VALUES(Order_ID_arg, User_Name_arg, Product_id_arg, Number_of_Items_arg, Shipping_Speed_arg, Seller_ID_arg);
END;

```

Script Output: Task completed in 0.15 seconds

Procedure ADD\_ORDERS compiled

- c. Purchasing a self-driving video camera (from Aspect 1), and , my facilitator purchases three holographic keyboards. Invoking the stored procedure twice, once for each purchase.



The screenshot shows the Oracle SQL Developer interface. In the top-left pane, there is a worksheet titled "Query Builder" containing the following PL/SQL code:

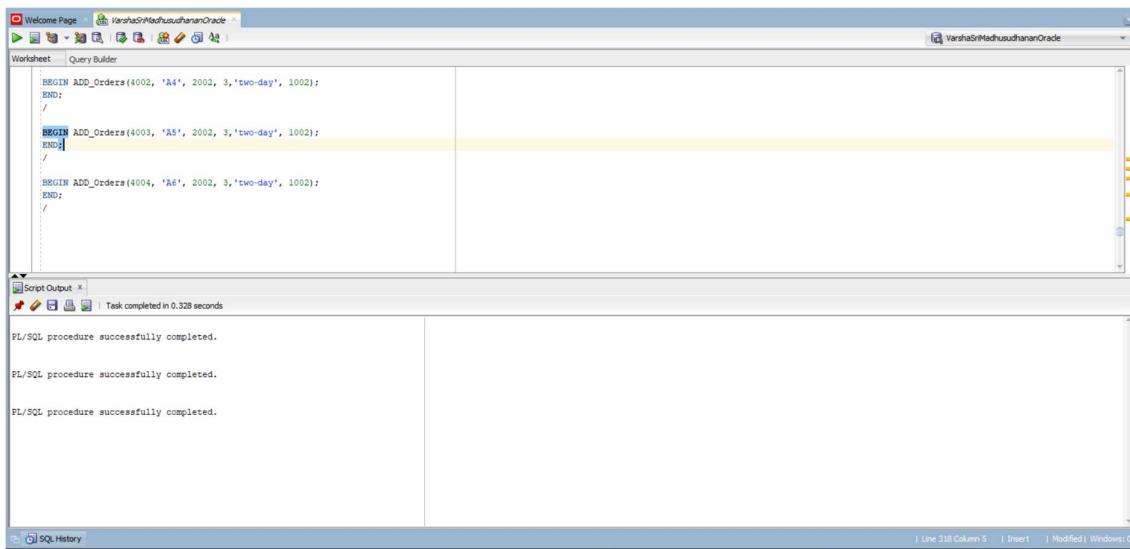
```
BEGIN ADD_Orders(4000, 'A3', 2000, 1,'one-day', 1000);
END;
/
BEGIN ADD_Orders(4001, 'A1', 2000, 3,'two-day', 1000);
END;
/
```

In the bottom-left pane, the "Script Output" window displays the results of the execution:

```
PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.
```

- d. The marketing department at Amazon wants 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.



The screenshot shows the Oracle SQL Developer interface. In the top-left pane, there is a worksheet titled "Query Builder" containing the following PL/SQL code:

```
BEGIN ADD_Orders(4002, 'A4', 2002, 3,'two-day', 1002);
END;
/
BEGIN ADD_Orders(4003, 'A5', 2002, 3,'two-day', 1002);
END;
/
BEGIN ADD_Orders(4004, 'A6', 2002, 3,'two-day', 1002);
END;
/
```

In the bottom-left pane, the "Script Output" window displays the results of the execution:

```
PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.
```

```

SELECT a.User_Name, a.First_Name, a.Last_Name, a.Address, a.City, a.State, a.Zip, a.Phone_Number, a.Email, o.Product_ID FROM Accounts a
INNER JOIN Product_Order o ON o.User_Name = a.User_Name
WHERE o.Product_ID IN (SELECT Product_ID FROM Product_Order
GROUP BY Product_ID
HAVING COUNT(DISTINCT User_Name) >= 3);

```

Script Output X | Query Result X

All Rows Fetched: 3 in 0.036 seconds

	User_Name	First_Name	Last_Name	Address	City	State	Zip	Phone_Number	Email	Product_ID
1 A4	Alex	Pandian	Greenland st	Austin	Austin	1910	6894561230	pandian@gmail.com	2002	
2 A5	Markin	Luther	Woodland st	Chicago	Chicago	1110	2894561230	luther@gmail.com	2002	
3 A6	Penny	Sri	Mason st	Provincetown	Provincetown	1210	8894561230	pari@gmail.com	2002	

## ASPECT 5: PRODUCT SHIPMENT BY AMAZON

- Creating the tables, constraints, and data needed to support product shipments.

```

CREATE TABLE Shipping(Tracking_ID DECIMAL(20) PRIMARY KEY,
Shipping_Address VARCHAR(255),
Shipping_City VARCHAR(50),
Shipping_State VARCHAR(50),
Shipping_Zip DECIMAL(20),
Shipping_Phone_Number DECIMAL(20),
Shipping_Status NUMBER(1),
Order_ID DECIMAL(20)
);

```

Script Output X

Table SHIPPING created.

Task completed in 0.085 seconds

The screenshot shows the Oracle SQL Developer interface. In the top-left corner, there's a 'Welcome Page' button and a connection icon labeled 'VarshaSriMadhusudhananOracle'. Below the connection icon, the tabs 'Worksheet' and 'Query Builder' are visible. The main area contains a code editor with the following SQL command:

```
ALTER TABLE Shipping
ADD CONSTRAINT Order_ID_FK
FOREIGN KEY(Order_ID) REFERENCES Product_Order(Order_ID);
```

Below the code editor, the 'Script Output' tab is selected, showing the message 'Table SHIPPING altered.' and a note 'Task completed in 0.078 seconds'.

- b. Developing a parameterized stored procedure that is used when Amazon ships any order.

The screenshot shows the Oracle SQL Developer interface. In the top-left corner, there's a 'Welcome Page' button and a connection icon labeled 'VarshaSriMadhusudhananOracle'. Below the connection icon, the tabs 'Worksheet' and 'Query Builder' are visible. The main area contains a code editor with the following PL/SQL code:

```
CREATE OR REPLACE PROCEDURE ADD_Shipping(
    Tracking_ID_arg IN DECIMAL,
    Shipping_Address_arg IN VARCHAR,
    Shipping_City_arg IN VARCHAR,
    Shipping_State_arg IN VARCHAR,
    Shipping_Zip_arg IN DECIMAL,
    Shipping_Phone_Number_arg IN DECIMAL,
    Shipping_Status_arg IN NUMBER,
    Order_ID_arg IN DECIMAL)
IS
BEGIN
    INSERT INTO Shipping(Tracking_ID, Shipping_Address, Shipping_City, Shipping_State, Shipping_Zip, Shipping_Phone_Number, Shipping_Status, Order_ID)
    VALUES(Tracking_ID_arg, Shipping_Address_arg, Shipping_City_arg, Shipping_State_arg, Shipping_Zip_arg, Shipping_Phone_Number_arg, Shipping_Status_arg, Order_ID_arg);
END;
```

Below the code editor, the 'Script Output' tab is selected, showing the message 'Procedure ADD\_SHIPPING compiled.' and a note 'Task completed in 0.185 seconds'.

- c. Amazon ships the orders listed in Aspect 4, one to you and the other to your facilitator. Invoking the stored procedure twice, once for each order.

```

BEGIN ADD_Shipping($000, 'Cambridge st', 'Boston', 'MA', 56454, 4596871236, 1, 4000);
END;
/
BEGIN ADD_Shipping($001, 'Stanford st', 'San Francisco', 'CA', 58754, 7896871236, 1, 4001);
END;
/

```

PL/SQL procedure successfully completed.  
PL/SQL procedure successfully completed.

```

SELECT * FROM Shipping;

```

	TRACKING_ID	SHIPPING_ADDRESS	SHIPPING_CITY	SHIPPING_STATE	SHIPPING_ZIP	SHIPPING_PHONE_NUMBER	SHIPPING_STATUS	ORDER_ID
1	5000 Cambridge st	Boston	MA	56454	4596871236	1	4000	
2	5001 Stanford st	San Francisco	CA	58754	7896871236	1	4001	

d. Calculating total shipped items for each user.

The screenshot shows the Oracle SQL Developer interface. The top window is a Worksheet titled 'shipping' with the following SQL code:

```
SELECT a.User_Name, COUNT(s.Tracking_ID) AS TotalShippedItems
FROM Accounts a
INNER JOIN Product_Order o ON a.User_Name = o.User_Name
INNER JOIN Shipping s ON o.Order_ID = s.Order_ID
GROUP BY a.User_Name;
```

The bottom window is a Script Output tab showing the results of the query:

USER_NAME	TOTALSHIPPEDITEMS
A3	1
A1	1

#### 4: Index Justification and Creation

The screenshot shows the Oracle SQL Developer interface. The top window is a Worksheet titled 'shipping' with the following SQL code:

```
CREATE INDEX index_last_name ON Accounts (Last_Name);
```

The bottom window is a Script Output tab showing the confirmation message:

```
Index INDEX_LAST_NAME created.
```

The screenshot shows the Oracle SQL Developer interface. In the top-left pane, a query is written:

```
SELECT COUNT(Last_Name) AS Last_Name_Sri FROM Accounts
WHERE Last_Name = 'Sri';
```

In the bottom-right pane, the results are displayed in a table:

Last_Name_Sri
4

## EXPLANATION:

Here, I have created an index for `Last_Name` in the `Accounts` table. An index is used to access the values from the table quickly and to increase the performance speed while executing. In the above scenario, the index can improve the performance of the search based on the user's last name in the query. When searching for a specific user (identified by last name), having an index on the `Last_Name` column allows the database engine to quickly locate the relevant rows in the `Accounts` table.

For example, the above-given query is used in Aspect 3 to retrieve the `Last_Name` of the users whose name is repeated more than 3 times, compiler acts quickly while the index is added. The previous query in Aspect 3 it takes 4 milliseconds after adding index it only takes 1 or 2 milliseconds.

## INSERTING VALUES:

**Note:** these are not part of the aspects, if wanted to insert additional values these statements can be used or even the stored procedures can be invoked again.

The screenshot shows the Oracle SQL Developer interface. In the top-left pane, several `INSERT` statements are written:

```
INSERT INTO Category(Category_ID, Category_Name)
VALUES(11, 'Kids');

INSERT INTO Seller(Seller_ID, Seller_First_Name, Seller_Last_Name, Seller_Address, Seller_City, Seller_State, Seller_Zip, Seller_Phone_Number, Seller_Email)
VALUES(1005, 'John', 'Way', 'F Road', 'San diego', 'California', '156310', '7894781230', 'Johnway@gmail.com');

INSERT INTO Products(Product_ID, Product_Name, Description, Price, Seller_ID, Category_ID)
VALUES(2004, 'Airpods', 'An earphones sold by Apple Inc.', $20.00, 1003, 2);
```

In the bottom-right pane, the results of the insertions are shown:

```
1 row inserted.

1 row inserted.

1 row inserted.
```

SQL Worksheet

```
INSERT INTO Warehouse(Receipt_ID, Product_ID, Number_Of_Units, Condition_of_Products, Item_Status)
VALUES(3004, 2004, 30, 'New', 'Delivered');

INSERT INTO Accounts(User_Name, First_Name, Last_Name, Address, City, State, Zip, Phone_Number, Email)
VALUES('A7', 'Penny', 'Richard', 'Tason st', 'Quincy', 'MA', 07610, 8845561230, 'prichard@gmail.com');

INSERT INTO Product_Order(Order_ID, User_Name, Product_id, Number_of_Items, Shipping_Speed, Seller_ID)
VALUES(4005, 'AE', 2002, 11, 'standard shipping', 1002);

INSERT INTO Shipping(Tacking_ID, Shipping_Address, Shipping_City, Shipping_State, Shipping_Zip, Shipping_Phone_Number, Shipping_Status, Order_ID)
VALUES(5002, 'Williams st', 'San Jose', 'CA', 58784, 789875636, 1, 4002);
```

Script Output | Query Result | Task completed in 0.494 seconds

1 row inserted.

SQL History | Line 126 Column 1 | Insert | Modified | Windows: C