

A. Structural Business Rules

New Product

- One listing has one to many products and one product is sold on one listing.
- One product is linked to one category and one category may have many products.

Product Delivery

- One seller can provide many products and one product is provided by one seller.

New Customer Account

- One customer has one to many accounts and one account belongs to one customer.

Product Purchase

- One customer may create many orders and one order is created by one customer.
- One product is linked to one order and one order can have one to many products.

Product Shipment

- One package is linked to one order and one order has one to many packages.
- One package has one to many products and one product is packed in one package.

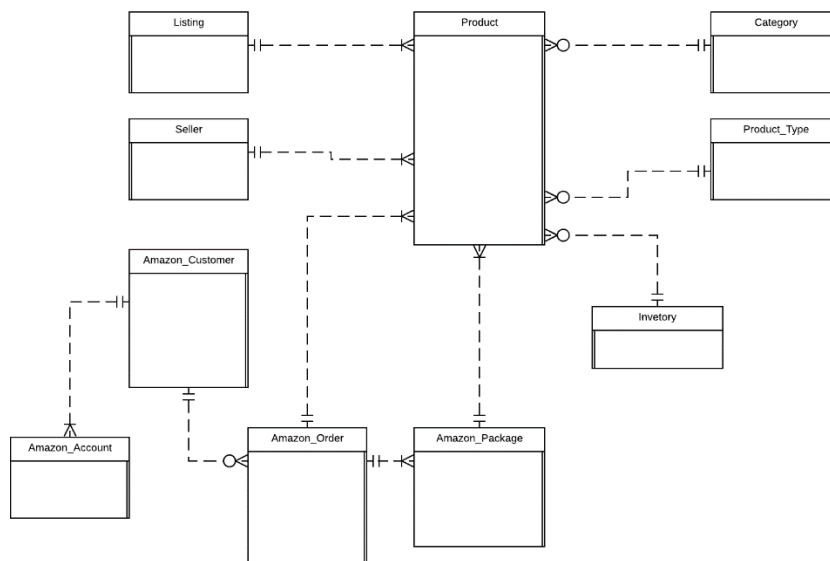
Product Type

- One product belongs to one type and one type may have many products.

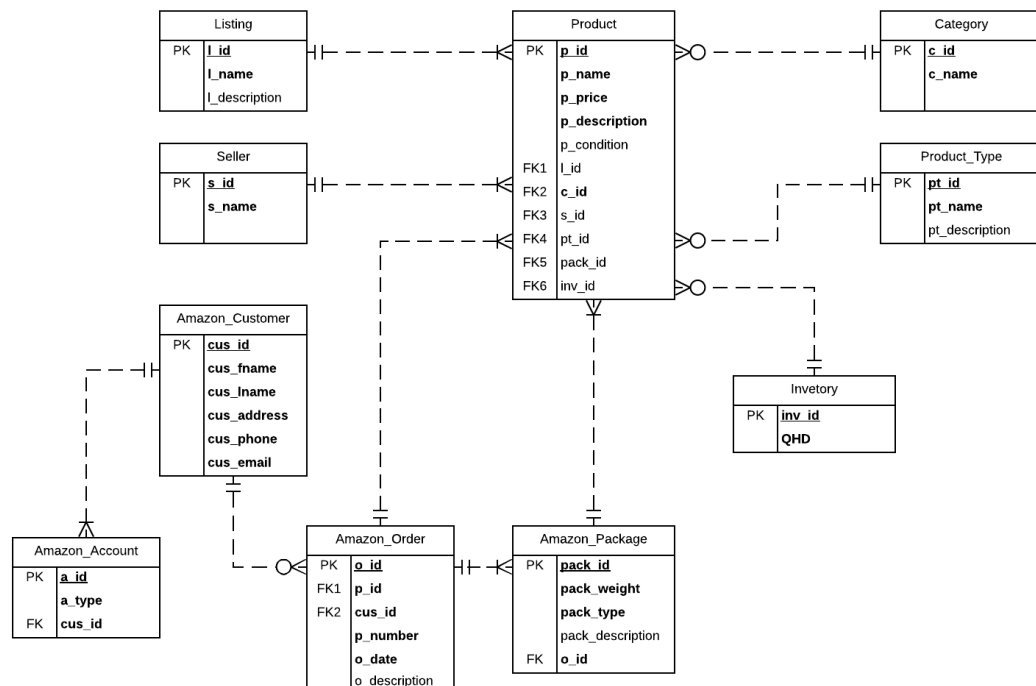
Product Inventory

- One product is stored in one inventory and one inventory may store many products

B. Conceptual ERD



C. Logical ERD



D. Aspects

Aspect1:

a. Reusable stored procedure.

The screenshot shows a SQL IDE window with a tab labeled 'myz'. The top toolbar includes icons for running, saving, and other database operations, along with a timer showing '0.50199997 seconds'. The main workspace is titled 'Query Builder' and contains the following SQL code:

```
CREATE OR REPLACE PROCEDURE ADD_PRODUCT( -- Add a new product
  p_id_arg IN DECIMAL,
  p_name_arg IN VARCHAR,
  p_price_arg IN DECIMAL,
  p_description_arg IN VARCHAR,
  c_id_arg IN DECIMAL)
IS
BEGIN
  INSERT INTO Product
  (p_id, p_name, p_price, p_description, c_id)
  VALUES (p_id_arg, p_name_arg, p_price_arg, p_description_arg, c_id_arg);
END;
```

The bottom panel is titled 'Script Output' and shows the message 'Task completed in 0.502 seconds'. Below this, the text 'Procedure ADD_PRODUCT compiled' is displayed.

b. Use of the stored procedure.

Oracle SQL Developer interface showing a SQL script in the Query Builder:

```
INSERT INTO Category VALUES (1, 'Computers');
INSERT INTO Category VALUES (2, 'Electronics');
```

The Script Output window shows the results of the execution:

```
1 row inserted.

1 row inserted.
```

Oracle SQL Developer interface showing a PL/SQL procedure in the Query Builder:

```
BEGIN
  ADD_PRODUCT(1, 'Self-driving Video Camera', 28, 'Automatically follows a subject that is being recorded.', 2);
END;
/

BEGIN
  ADD_PRODUCT(2, 'Holographic Keyboard', 28, 'Emits a three-dimensional projection of a keyboard and recognizes virtual key presses from the typist.', 1);
END;
```










The Script Output window shows the results of the execution:

```
PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.
```

c. SQL query.


Welcome Page x myz x



0.028 seconds

Worksheet






Query Builder



```
SELECT Product.p_name, Product.p_price, Category.c_name
FROM Product
INNER JOIN Category ON Product.c_id = Category.c_id
WHERE Product.p_price <= 30
AND (Category.c_name = 'Computers' OR Category.c_name = 'Electronics');
```

Script Output x

Query Result x



Task completed in 0.028 seconds

P_NAME	P_PRICE	C_NAME
Self-driving Video Camera	28	Electronics
Holographic Keyboard	28	Computers

Change to the procedure:

The screenshot shows a SQL IDE window with a tab labeled 'myz'. The main editor displays the following SQL code:

```
CREATE OR REPLACE PROCEDURE ADD_PRODUCT( -- Add a new product
p_id_arg IN DECIMAL,
p_name_arg IN VARCHAR,
p_price_arg IN DECIMAL,
p_description_arg IN VARCHAR,
c_id_arg IN DECIMAL,
pt_id_arg IN DECIMAL,
pt_name_arg IN VARCHAR)
IS
BEGIN

DECLARE
    x NUMBER:=0;
BEGIN
    SELECT nvl((SELECT 1 FROM Product_Type WHERE pt_id = pt_id_arg) , 0) INTO x FROM dual;

    IF (X = 1) THEN

        INSERT INTO Product
        (p_id,p_name,p_price,p_description,c_id,pt_id)
        VALUES (p_id_arg,p_name_arg,p_price_arg,p_description_arg,c_id_arg,pt_id_arg);

    ELSE














































        INSERT INTO Product_type(pt_id, pt_name) VALUES
        (pt_id_arg, pt_name_arg);
        INSERT INTO Product
        (p_id,p_name,p_price,p_description,c_id,pt_id)
        VALUES (p_id_arg,p_name_arg,p_price_arg,p_description_arg,c_id_arg,pt_id_arg);

    END IF;
END;
END;
```













































The bottom panel shows the 'Script Output' tab with the message: 'Task completed in 0.212 seconds' and 'Procedure ADD_PRODUCT compiled'.

Welcome Page

myz














Welcome Page x myz x



Welcome Page

myz



0.175 seconds

Worksheet

Query Builder

SELECT * FROM product_type;

Script Output

Task completed in 0.175 seconds

PT_ID	PT_NAME	PT_DESCRIPTION
1	Video Camera	

Welcome Page

myz

0.025 seconds

Worksheet

Query Builder

BEGIN

ADD_PRODUCT(3, 'Holographic Keyboard', 28, 'Emits a three-dimensional projection of a keyboard and recognizes virtual key presses from the typist.', 1, 2, 'Virtual Keyboard');

END;













!

Script Output

Task completed in 0.025 seconds

PL/SQL procedure successfully completed.





Welcome Page x myz x



WorksheetQuery Builder

```
SELECT * FROM Product_type;
```

Script Output xQuery Result x

 SQL | All Rows Fetched: 2 in 0.013 seconds

	PT_ID	PT_NAME	PT_DESCRIPTION
1	1	Video Camera	(null)
2	2	Vitual Keyboard	(null)

Welcome Page myz

Worksheet Query Builder

`SELECT * FROM Product;`

Script Output Query Result

SQL | All Rows Fetched: 5 in 0.003 seconds

P_ID	P_NAME	P_PRICE	P_DESCRIPTION	I_ID	C_ID	S_ID	PT_ID	PACK_ID	INV_ID
1	4 Self-driving Video Camera	28	Automatically follows a subject that is being recorded.	(null)	2	(null)	1	(null)	(null)
2	5 Self-driving Video Camera	28	Automatically follows a subject that is being recorded.	(null)	2	(null)	1	(null)	(null)
3	3 Holographic Keyboard	28	Emits a three-dimensional projection of a keyboard and recognizes virtual key presses from the typist.	(null)	1	(null)	2	(null)	(null)
4	1 Self-driving Video Camera	28	Automatically follows a subject that is being recorded.	(null)	2	(null)	(null)	(null)	(null)
5	2 Holographic Keyboard	28	Emits a three-dimensional projection of a keyboard and recognizes virtual key presses from the typist.	(null)	1	(null)	(null)	(null)	(null)

Aspect2:

- Reusable stored procedure.

The screenshot shows a SQL IDE window with a 'Welcome Page' and a 'myz' database connection. The 'Query Builder' tab is active, displaying the following SQL code:

```
CREATE OR REPLACE PROCEDURE ADD_INVETORY( -- Add product from seller
p_id_arg IN DECIMAL,
inv_id_arg IN DECIMAL,
QHD_arg IN DECIMAL,
p_condition_arg IN VARCHAR)
IS
BEGIN

UPDATE Invetory
SET QHD = QHD_arg
WHERE inv_id = inv_id_arg;

UPDATE Product
SET p_condition = p_condition_arg,
    inv_id = inv_id_arg
WHERE p_id = p_id_arg;

END;
```

The 'Script Output' tab at the bottom shows the result: 'Task completed in 0.018 seconds' and 'Procedure ADD_INVETORY compiled'.

b. Use of the stored procedure.

Welcome Page x myz x

Worksheet Query Builder

```
INSERT INTO Inventory(inv_id,QHD)
VALUES (1,0);

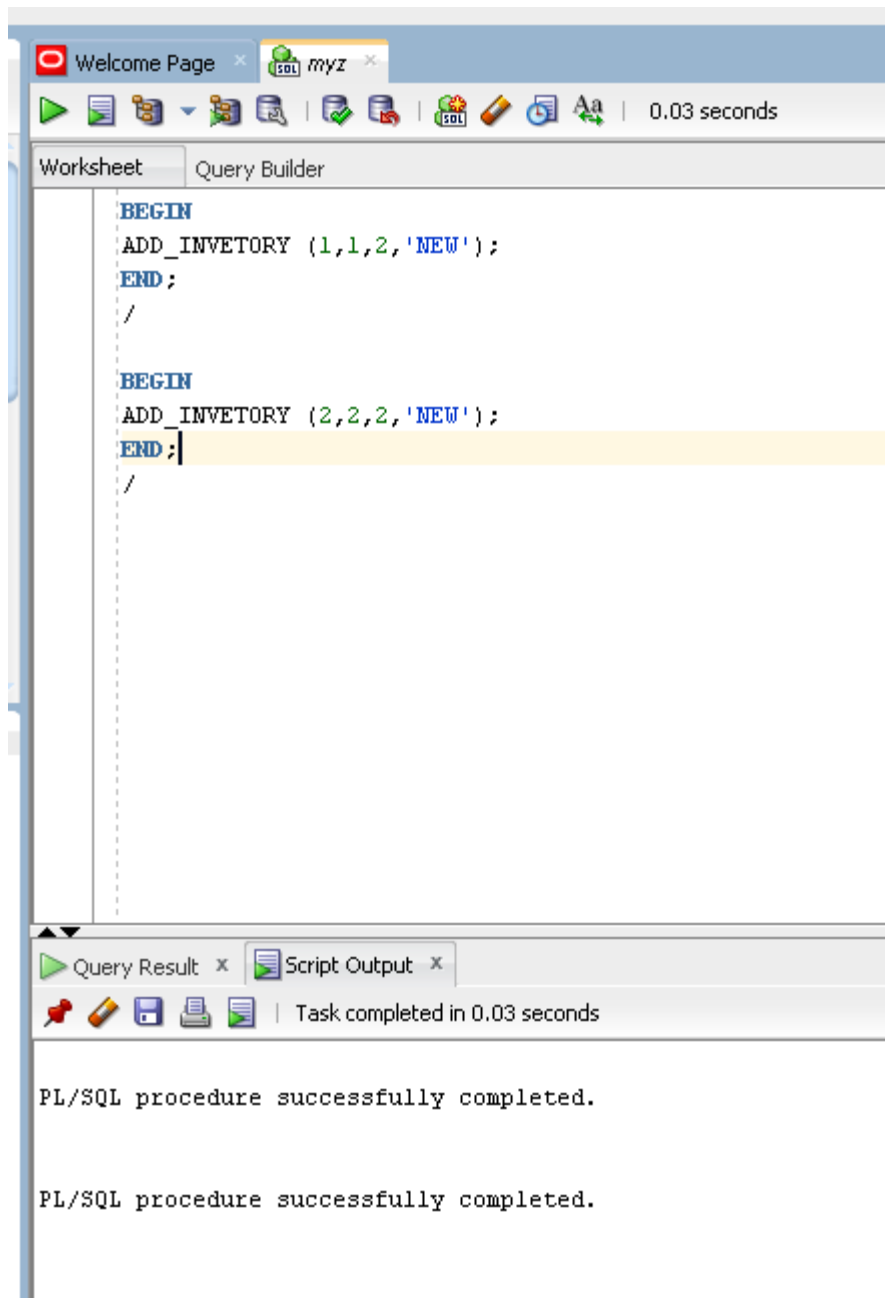
INSERT INTO Inventory(inv_id,QHD)
VALUES (2,0);
```

Query Result x Script Output x

Task completed in 0.131 seconds

1 row inserted.

1 row inserted.



c. SQL query.

Worksheet Query Builder

```
SELECT Product.p_name, Inventory.QHD
FROM Product
INNER JOIN Inventory ON Product.inv_id = Inventory.inv_id
WHERE Inventory.QHD <= 11;
```

Script Output x Query Result x

SQL | All Rows Fetched: 2 in 0.004 seconds

	P_NAME	QHD
1	Self-driving Video Camera	2
2	Holographic Keyboard	2

Aspect3:

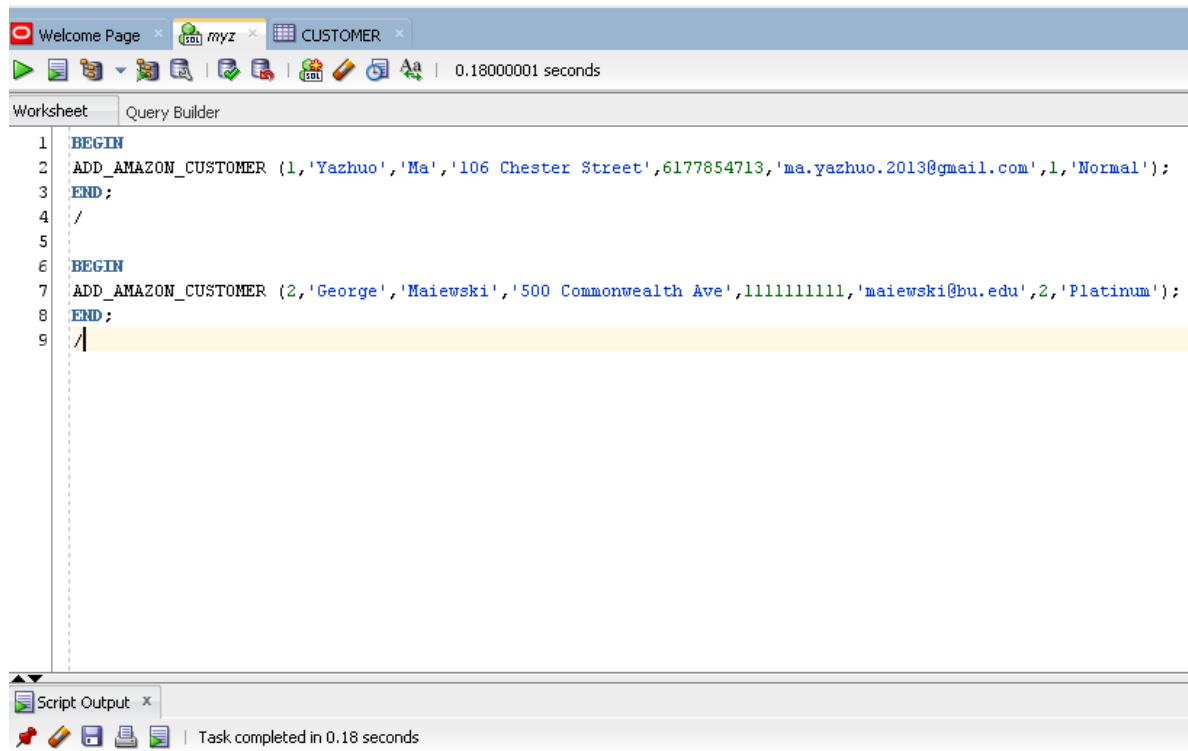
- Reusable stored procedure.

The screenshot shows a SQL IDE window with a tab titled 'CUSTOMER'. The main editor area contains the following SQL code:

```
1 CREATE OR REPLACE PROCEDURE ADD_AMAZON_CUSTOMER( -- Add product from seller
2   cus_id_arg DECIMAL,
3   cus_fname_arg VARCHAR,
4   cus_lname_arg VARCHAR,
5   cus_address_arg VARCHAR,
6   cus_phone_arg DECIMAL,
7   cus_email_arg VARCHAR,
8   a_id_arg DECIMAL,
9   a_type_arg VARCHAR)
10 IS
11 BEGIN
12
13
14 INSERT INTO Amazon_Customer
15 VALUES (cus_id_arg,cus_fname_arg,cus_lname_arg,cus_address_arg,cus_phone_arg,cus_email_arg);
16
17
18 INSERT INTO Amazon_Account
19 VALUES (a_id_arg,a_type_arg,cus_id_arg);
20
21 END;
```

Below the editor is a 'Script Output' window. It shows the message 'Task completed in 0.244 seconds' and 'Procedure ADD_AMAZON_CUSTOMER compiled'.

b. Use of the stored procedure.



PL/SQL procedure successfully completed.













PL/SQL procedure successfully completed.

c. SQL query.

Welcome Pa...

SOL ...

CUSTOM...



Worksheet

Query Builder

1

2

3

4

5






6

7

```
INSERT INTO Amazon_Account
VALUES (3,'Normal',1);
INSERT INTO Amazon_Account
VALUES (4,'Platinum',1);
INSERT INTO Amazon_Account
VALUES (5,'Normal',1);
```

Script Output

Query Result



Task completed in 0.074 seconds

1 row inserted.

1 row inserted.

1 row inserted.

The screenshot shows a SQL query builder application. The top toolbar includes icons for running queries, saving, and other database functions. The main area is divided into a 'Worksheet' tab and a 'Query Builder' tab. The 'Query Builder' tab contains a SQL query:

```
1 SELECT Amazon_Customer.cus_lname, COUNT(*)
2 FROM Amazon_Customer
3 INNER JOIN Amazon_Account ON Amazon_Customer.cus_id = Amazon_Account.cus_id
4 GROUP BY Amazon_Customer.cus_lname
5 HAVING COUNT(*) >= 4;
```

Below the query editor, there is a 'Script Output' tab and a 'Query Result' tab. The 'Query Result' tab shows the results of the query:

All Rows Fetched: 1 in 0.004 seconds

CUS_LNAME	COUNT(*)
1 Ma	4

Aspect4:

- Reusable stored procedure.

The screenshot shows a database query editor window with the title bar 'AMAZON_ORD...'. The 'Query Builder' tab is active, displaying the following SQL code:

```
CREATE OR REPLACE PROCEDURE ADD_Amazon_Order(  
    o_id_arg IN DECIMAL,  
    p_id_arg IN DECIMAL,  
    cus_id_arg IN DECIMAL,  
    p_number_arg IN DECIMAL,  
    o_date_arg IN DATE,  
    o_description_arg IN VARCHAR)  
IS  
BEGIN  
  
    INSERT INTO Amazon_Order(o_id,p_id,cus_id,p_number,o_date,o_description)  
    VALUES(o_id_arg,p_id_arg,cus_id_arg,p_number_arg,o_date_arg,o_description_arg);  
  
END;
```































Below the query editor, the 'Script Output' tab is visible, showing the message: 'Task completed in 0.137 seconds' and 'Procedure ADD_AMAZON_ORDER compiled'.

b. Use of the stored procedure

Welcome Pa...

SQL ...

AMAZON_CUSTOM...



0.13500001 seconds

Worksheet

Query Builder

1

BEGIN

2

ADD_Amazon_Order (1,1,1,1,CAST('04-DEC-2018' AS DATE), 'None');

3

END;

4

/

5

6

BEGIN

7

ADD_Amazon_Order (2,2,2,3,CAST('04-DEC-2018' AS DATE), 'None');

8

END;

9

/

10

11

12

Script Output












Task completed in 0.135 seconds

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

Welcome Page xmyz xAMAZON_ACCOUNT x












0.034 seconds

WorksheetQuery Builder

```
BEGIN
ADD_AMAZON_CUSTOMER (3,'Chenyang','Wang','107 Chester Street',6177854713,'chengyang.wang@gmail.com',6,'Golden');
END;
/

BEGIN
ADD_AMAZON_CUSTOMER (4,'James','Yu','108 Chester Street',6666666666,'james.yu@gmail.com',7,'Golden');
END;
/
```

Script Output x



Task completed in 0.034 seconds

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

Oracle SQL Developer interface showing a PL/SQL script being executed. The script contains two procedures, both of which successfully completed.

Script:

```
BEGIN
ADD_Amazon_Order (3,1,3,2,CAST('05-DEC-2018' AS DATE), 'None');
END;
/

BEGIN
ADD_Amazon_Order (4,1,4,1,CAST('05-DEC-2018' AS DATE), 'None');
END;
/
```

Script Output:

```
PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.
```

c. SQL query.

The screenshot shows a SQL query builder interface with a 'Query Builder' tab selected. The query is as follows:

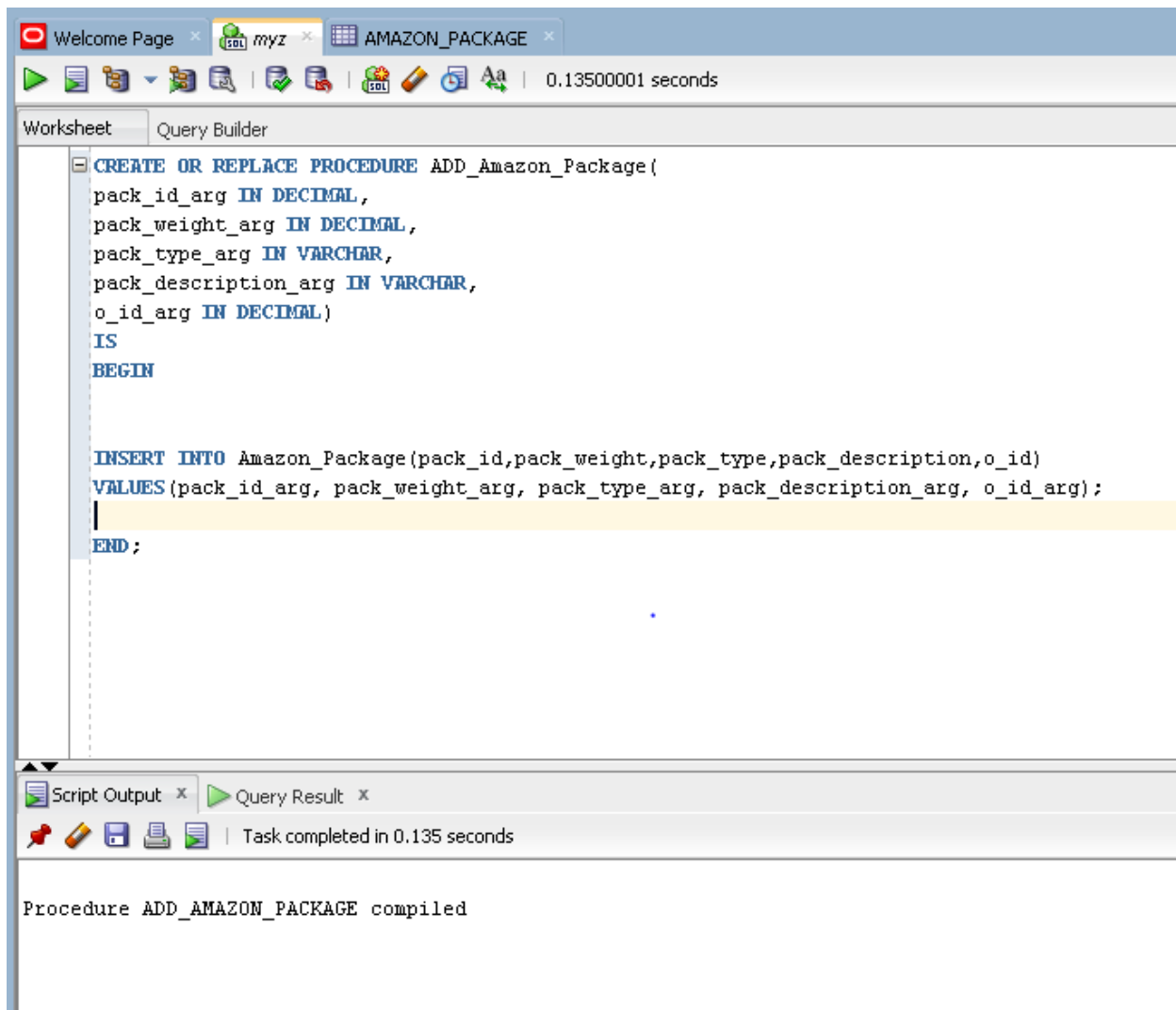
```
SELECT cus_fname, cus_lname, cus_address
FROM Amazon_Order
JOIN Amazon_Customer ON Amazon_Order.cus_id = Amazon_Customer.cus_id
WHERE Amazon_Order.p_id IN( SELECT p_id
                             FROM Amazon_Order
                             GROUP BY p_id
                             HAVING COUNT(p_id) >= 3)
```

Below the query, the 'Query Result' tab is active, displaying the results of the query. The results are shown in a table with three columns: CUS_FNAME, CUS_LNAME, and CUS_ADDRESS. The table contains three rows of data.

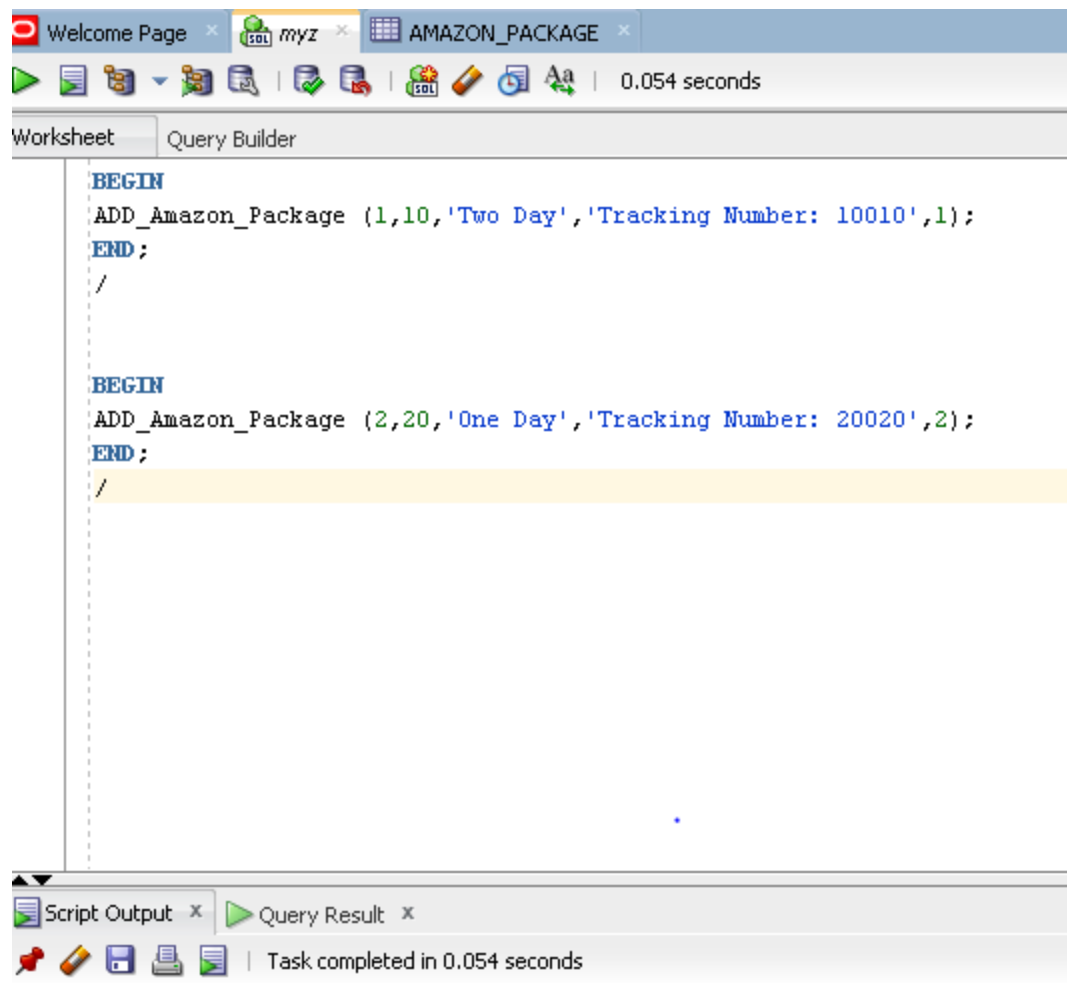
	CUS_FNAME	CUS_LNAME	CUS_ADDRESS
1	Yazhuo	Ma	106 Chester Street
2	Chenyang	Wang	107 Chester Street
3	James	Yu	108 Chester Street

Aspect5:

- Creation of the stored procedure



b. Use of the stored procedure



PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

c. SQL Query

Welcome Page x myz x AMAZON_PACKAGE x

0.024 seconds

Worksheet Query Builder

```
SELECT pack_id
FROM (SELECT pack_id, pack_weight
      FROM Amazon_Package
      WHERE pack_type = 'Two Day')
WHERE pack_weight <= 10;
```

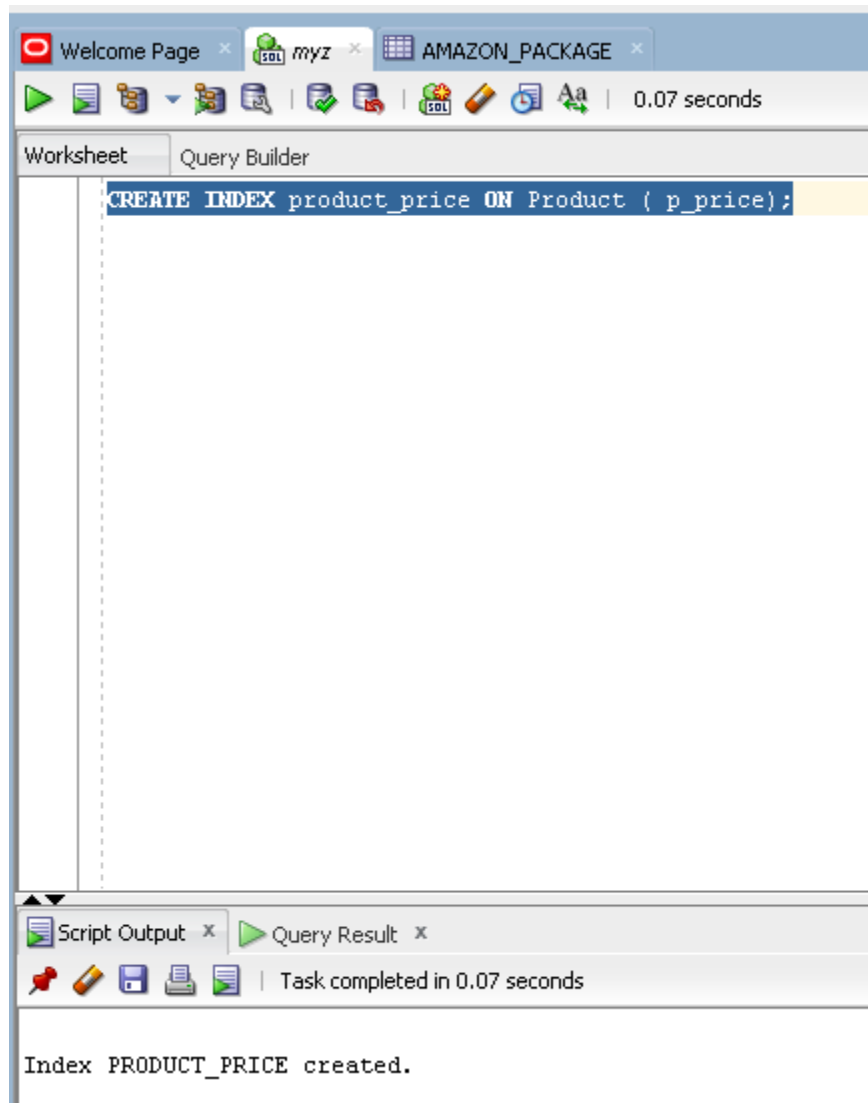
Script Output x Query Result x

Task completed in 0.024 seconds

PACK_ID

1

INDEX:



This Index can speed up the query speed of aspect 1, which will reduce the spending time to find out p_price less than or equal to 30.

