AMERICAN INTERNATIONAL UNIVERSITY OF BANGLADESH

REPORT COVER PAGE

Submitted to

Juena Ahmed Noshin

Course Name: Advance Database Management System

Section: A

Project Name: Courier Management System

Submitted by

NAME	ID	CONTRIUBUTION
MARUF, IMRAN HOSSAIN	19-41806-3	30%
RAFI, MUSHFIQUR RAHMAN	20-43053-1	30%
RAIYAN,ASHFI MUHAMMED	22-46894-1	40%

Content

Introduction

The effective management of courier services has become crucial in the modern company environment for encouraging growth and ensuring client satisfaction. We offer a comprehensive system called the Courier Management System that aims to simplify and improve the complex process of product delivery. Our solution aims to promote seamless communication between retailers, distribution facilities, delivery staff, and customers. We want to improve the end-to-end courier experience by utilizing technology, guaranteeing that packages arrive at their destinations promptly and securely. Our system offers an integrated platform that precisely orchestrates the whole delivery workflow in order to meet the different needs of both customers and merchants.

Project Proposal

By offering a cutting-edge framework that streamlines and improves the entire delivery process, our courier management system seeks to transform the logistics industry. The primary feature centers on merchants' simple order placement. An order is placed, and then a well planned series of actions ensures that it travels smoothly from the merchant to the recipient.

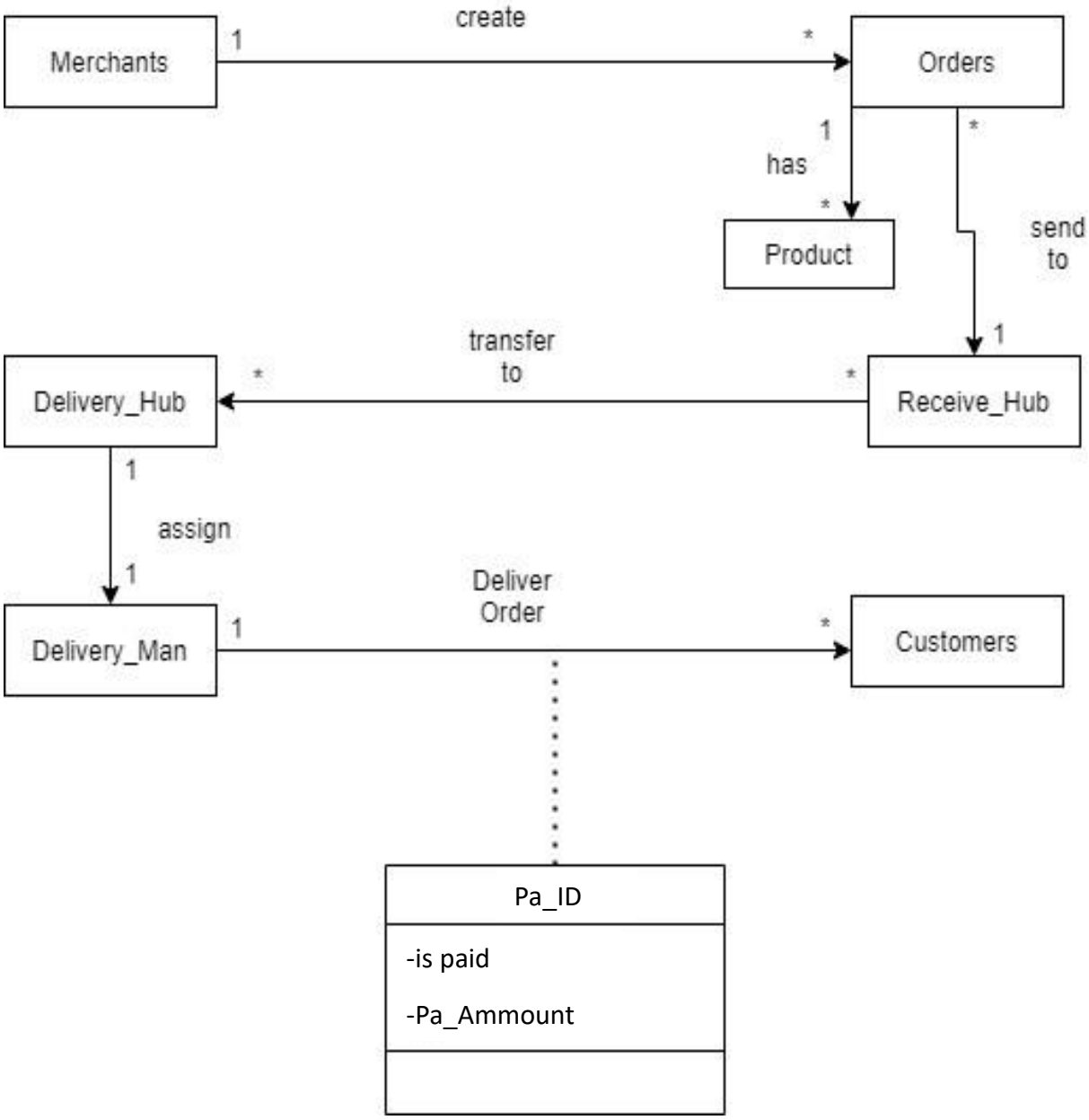
When placing an order, merchants can easily enter crucial information like their name and contact information. Throughout the delivery lifecycle, this information acts as a vital communication connection. Each order is given a special Order ID that serves as the main point of tracking and management.

The order travels via important nodes starting with the receiver hub as it moves forward. The system logs important data at this point, including the recipient's name, location, and expected arrival time. The system can maintain exact control over order movement and delivery schedules thanks to this data.

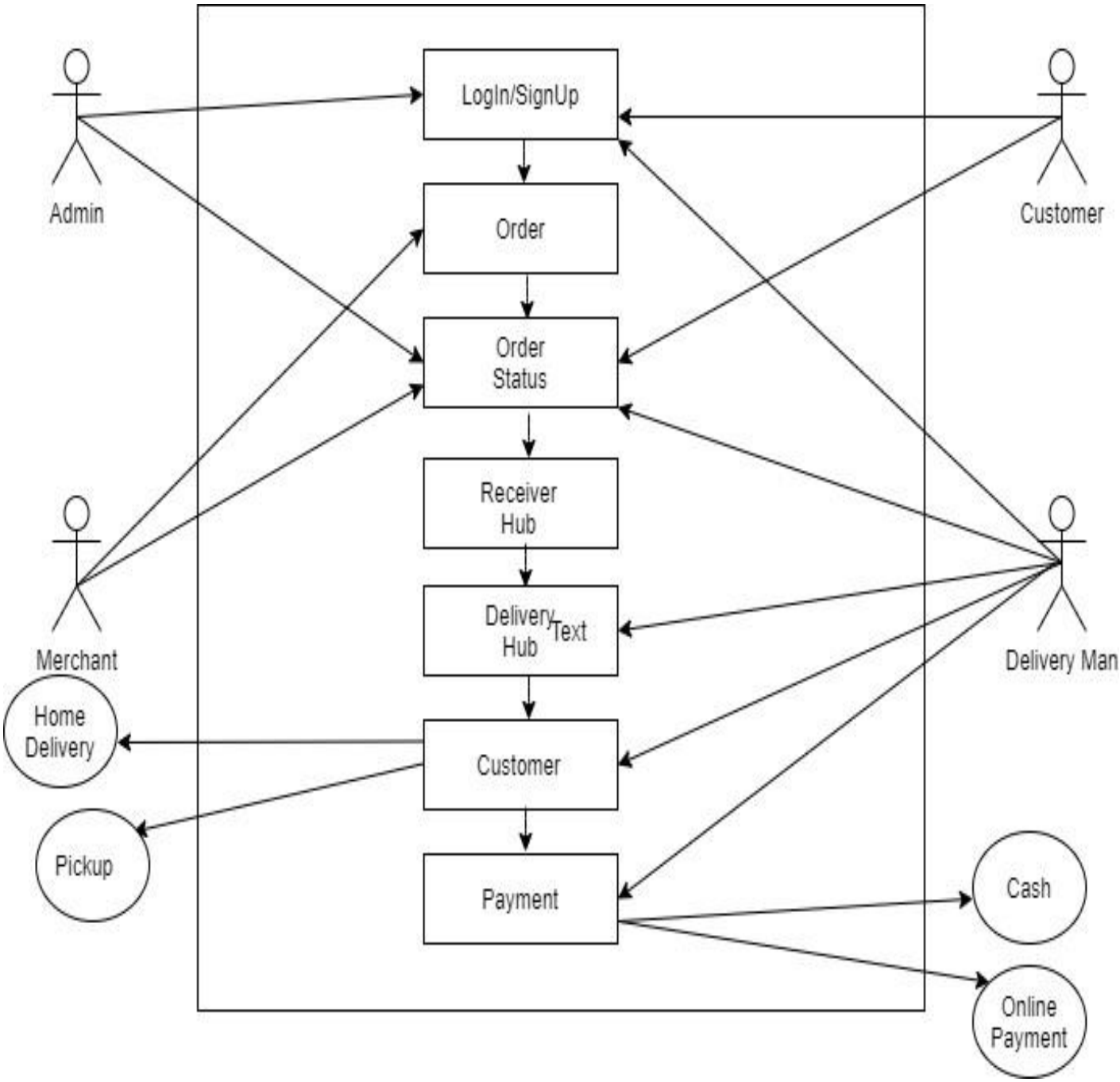
The system then designates a dedicated delivery team so they can oversee and carry out the last leg of the process. Transporting the order from the receiver hub to the customer's specified location is under the purview of the designated delivery person. In order to ensure an accurate and effective last-mile delivery, this phase involves gathering crucial information such as the delivery location and projected delivery time.

In conclusion, our Courier Management System transforms the courier sector by providing a streamlined and transparent procedure for placing, monitoring, and completing orders. Our solution enhances the delivery experience for merchants, customers, and all associated parties by merging technological innovation and a thorough approach.

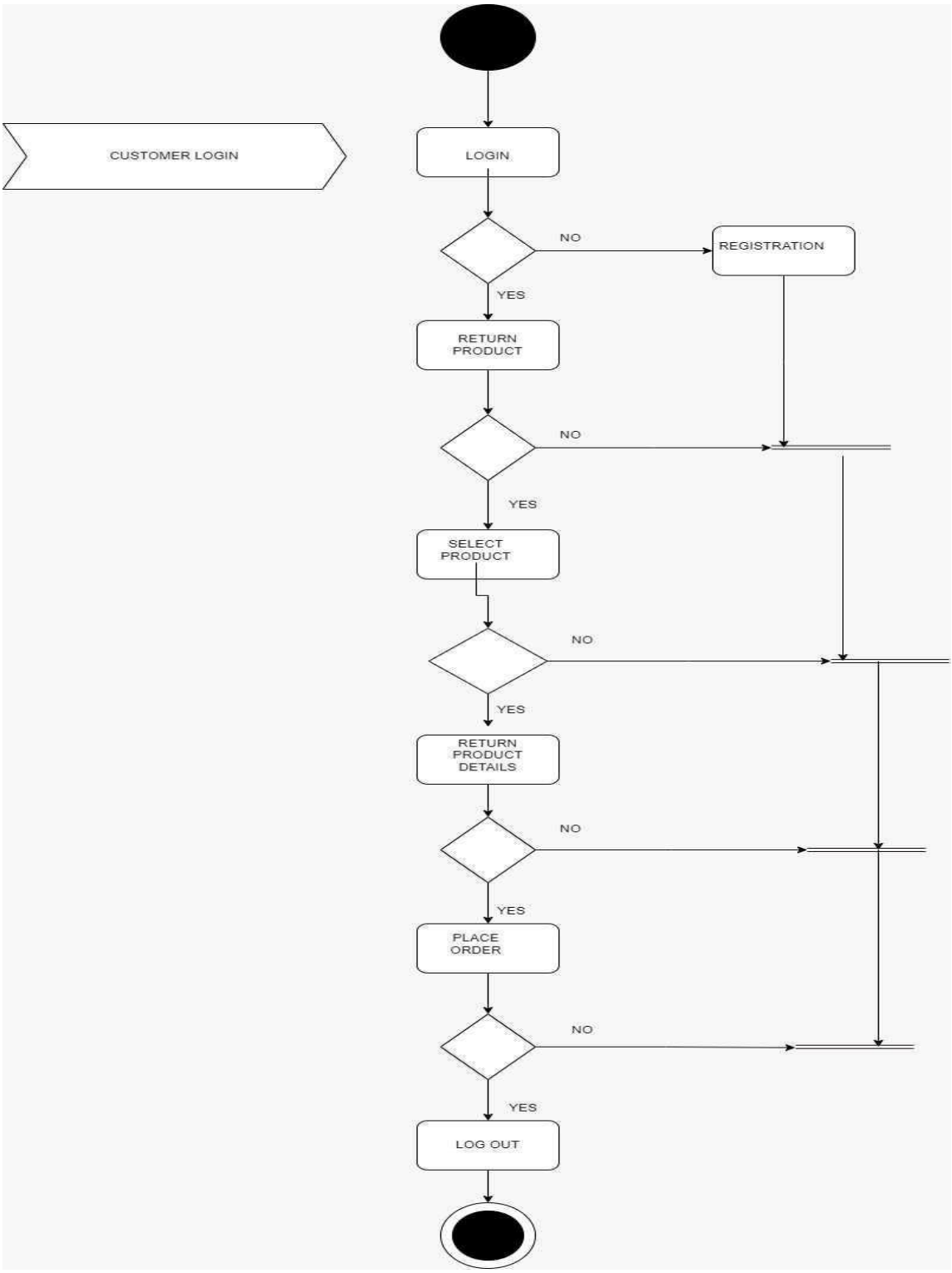
Class Diagram

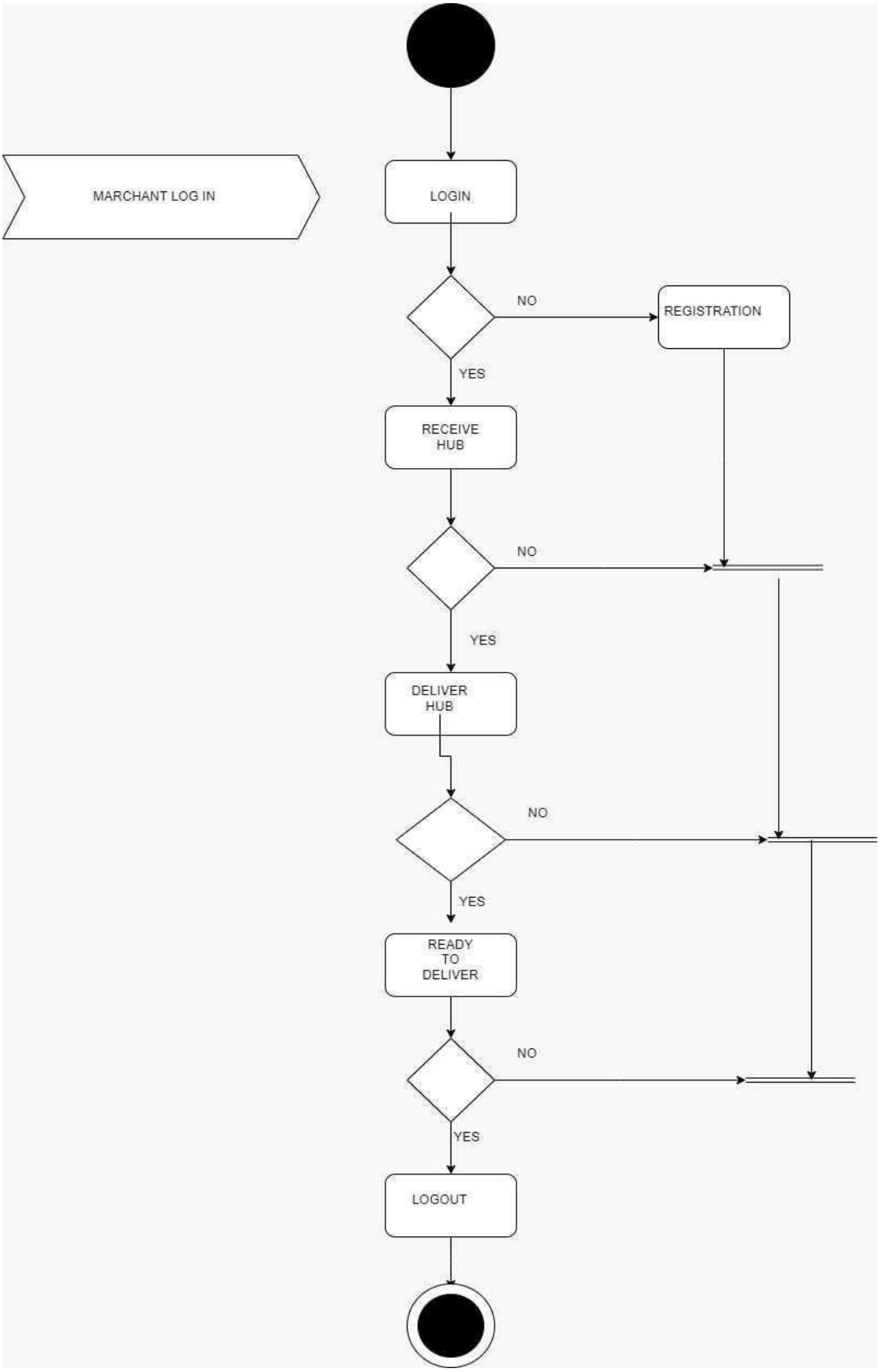


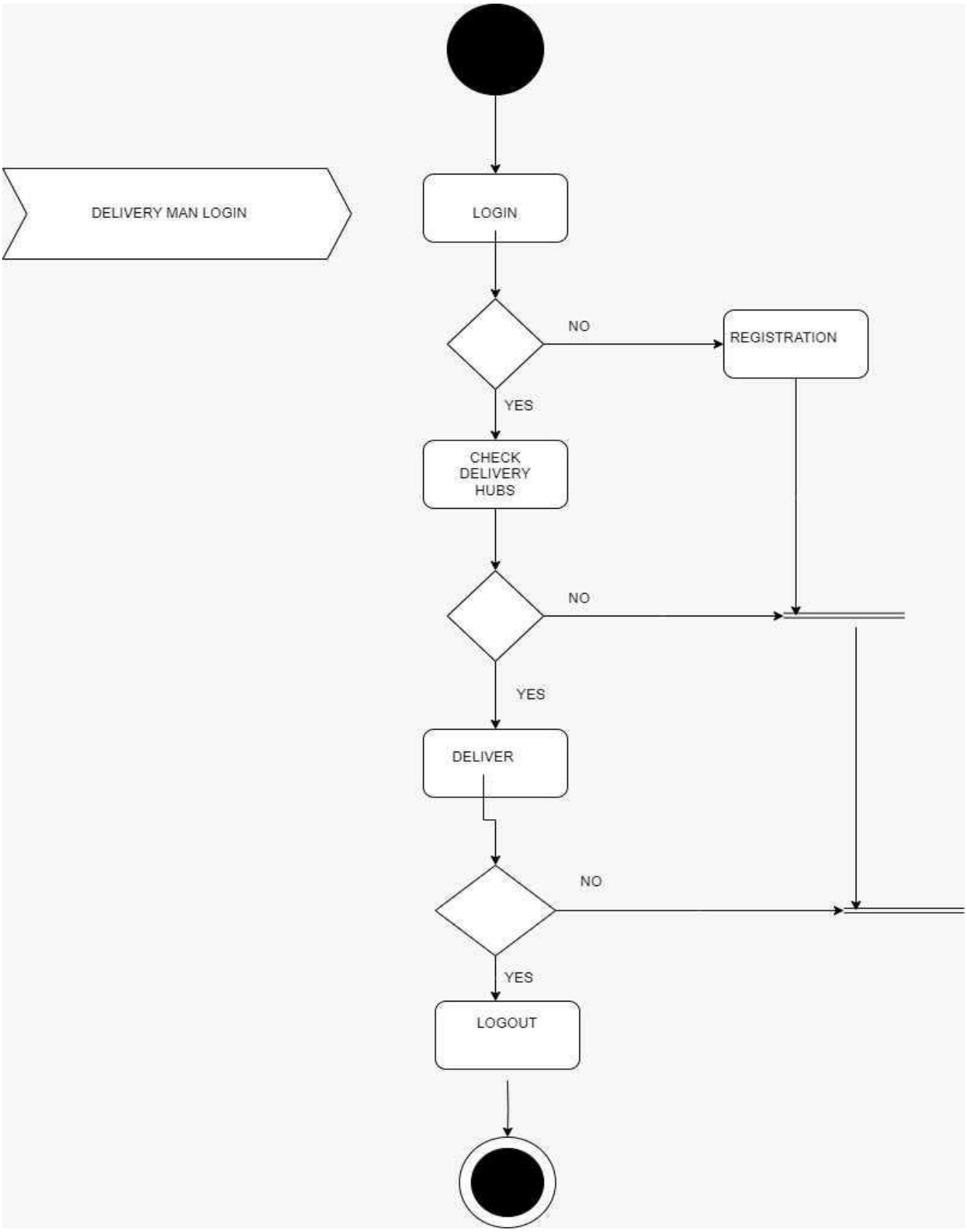
Use Case Diagram



Activity Diagram:

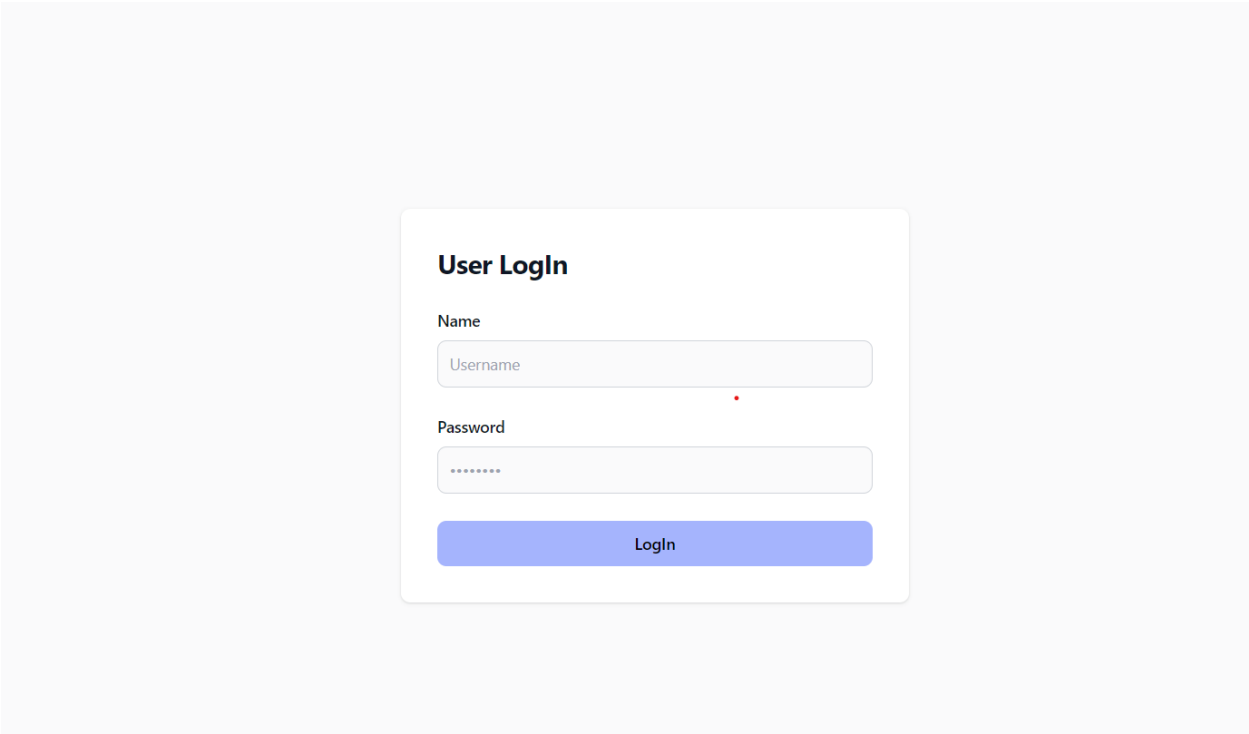



MARCHANT LOG IN
LOGIN
NO
REGISTRATION
YES
RECEIVE HUB
NO
YES
DELIVER HUB
NO
YES
READY TO DELIVER
NO
YES
LOGOUT

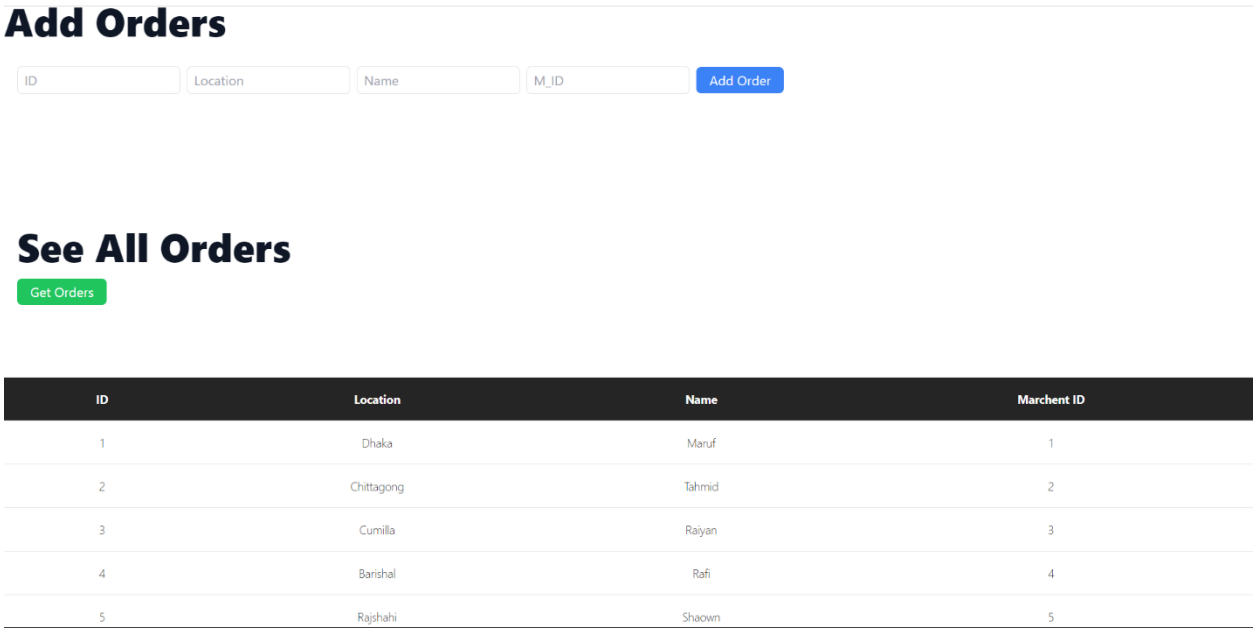

DELIVERY MAN LOGIN
LOGIN
NO
REGISTRATION
YES
CHECK DELIVERY HUBS
NO
YES
DELIVER
NO
YES
LOGOUT

User Interface:

Login Page



Order Status Page

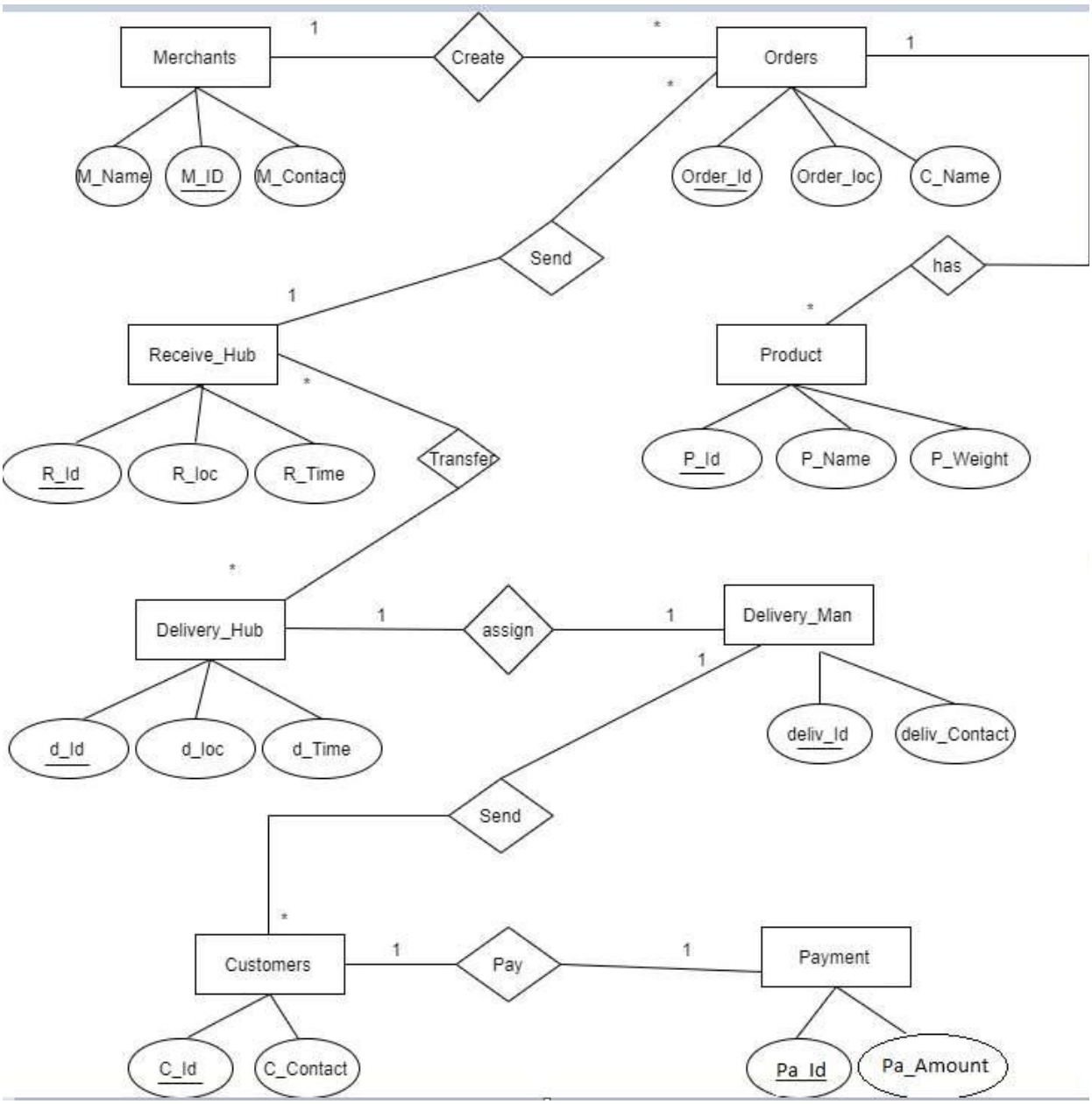


ID	Location	Name	Marchent ID
1	Dhaka	Maruf	1
2	Chittagong	Tahmid	2
3	Cumilla	Raiyan	3
4	Barishal	Rafi	4
5	Rajshahi	Shaown	5

Scenario Description:

This is a courier service management system. In this system one Merchant can create multiple orders where a merchant must assign M_ID, M_Name , M_Contact in merchant Table and in Order Table it's need to include Order_ID, O_Loc , C_Name . These multiple Orders send to one Receive_Hub where Receive_Hub has R_id, R-loc , R_Time attributes . Multiple Receiver_Hub Transfer these orders to multiple Delivery_Hub that includes d_id, d_loc, d_Time in table. Each Delivery_Hub assign one Delivery_Man which needs to include delive_id and delive_contact. One Delivery_Man send the orders to multiple Customer. Customer has C_id and C_Contact .One Customer can pay one Payment. Payment must include Pa_id and Pa_Ammount.

ER Diagram



Normalization

Create:

UNF

Create(M_ID, M_Name, M_Contact, Order_ID , Order_Loc, C_Name)

1NF

There is no multi valued attribute. Relation already in 1NF.

1. M_ID, M_Name, M_Contact, Order_ID , Order_Loc, C_Name

2NF

1. M_ID, M_Name, M_Contact
2. Order_ID , Order_Loc, C_Name, **M_ID**

3NF

There is no transitive dependency. Relation already in 3NF.

1. M_ID,M_Name ,M_Contact
2. Order_ID ,Order_Loc ,C_Name ,**M_ID**

Table Creation:

1: M_ID,M_Name ,M_Contact

2: Order_ID, Order_Loc ,C_Name ,**M_ID**

Has:

UNF

Has(Order_ID , Order_Loc, C_Name, P_ID , P_Name , P_Weight)

1NF:

There is no multi valued attribute. Relation already in 1NF.
1: Order_ID , Order_Loc, C_Name, P_ID , P_Name , P_Weight

2NF:

1: Order_ID , Order_Loc, C_Name

2: P_ID , P_Name , P_Weight , **Order_ID**

3NF:

There is no transitive dependency. Relation already in 3NF.
1: Order_ID , Order_Loc, C_Name

2: P_ID , P_Name , P_Weight , **Order_ID**

Table Creation

1: Order_ID , Order_Loc, C_Name

2: P_ID , P_Name , P_Weight , **Order_ID**

Send1

UNF

Send1 (Order_ID , Order_Loc, C_Name , R_ID , R_Loc, R_time)

1NF:

There is no multi valued attribute. Relation already in 1NF.
1: Order_ID , Order_Loc, C_Name , R_ID , R_Loc, R_time

2NF:

1: Order_ID , Order_Loc, C_Name

2: R_ID , R_Loc, R_time, **Order_ID**

3NF:

There is no transitive dependency. Relation already in 3NF.
1: Order_ID , Order_Loc, C_Name

2: R_ID , R_Loc, R_time, **Order_ID**

Table Creation

1: Order_ID , Order_Loc, C_Name

2: R_ID , R_Loc, R_time, **Order_ID**

Transfer

UNF

Transfer (R_ID , R_Loc, R_time ,D_ID , D_Loc, D_Time)

1NF:

There is no multi valued attribute. Relation already in 1NF.
1: R_ID , R_Loc, R_time ,D_ID , D_Loc, D_Time

2NF:

1: R_ID , R_Loc, R_time, **D_ID**

2: D_ID , D_Loc, D_Time

3NF:

There is no transitive dependency. Relation already in 3NF.
1: R_ID , R_Loc, R_time, **D_ID**

2: D_ID , D_Loc, D_Time

Table Creation

1: R_ID , R_Loc, R_time, **D_ID**

2: D_ID , D_Loc, D_Time

Assign

UNF:
Assign(D_ID , D_Loc, D_Time, Deliv_ID , Deliv_Contact
1NF

There is no multi valued attribute. Relation already in 1NF.
1: D_ID , D_Loc, D_Time, Deliv_ID , Deliv_Contact
2NF
1: D_ID , D_Loc, D_Time **, Deliv_ID**
2: Deliv_ID , Deliv_Contact
3NF

There is no transitive dependency. Relation already in 3NF.
1: D_ID , D_Loc, D_Time , **Deliv_ID**
2: Deliv_ID , Deliv_Contact

Table Creation

1: D_ID , D_Loc, D_Time , **Deliv_ID**
2: Deliv_ID , Deliv_Contact

Send2

UNF
Send2(C_ID , C_Contact, Deliv_ID ,Deliv_Contact)
1NF

There is no multi valued attribute. Relation already in 1NF.
1: C_ID , C_Contact, Deliv_ID ,Deliv_Contact
2NF:
1: C_ID , C_Contact
2: Deliv_ID ,Deliv_Contact, **C_ID**

3NF:

There is no transitive dependency. Relation already in 3NF.
1: C_ID , C_Contact
2: Deliv_ID ,Deliv_Contact, **C_ID**

Table Creation

1: C_ID , C_Contact
2: Deliv_ID ,Deliv_Contact, **C_ID**

Pay

UNF

Pay(Pa_ID, Pa_Ammount , C_ID, C_Contact)

1NF

There is no multi valued attribute. Relation already in 1NF.
1: Pa_ID, Pa_Ammount , C_ID, C_Contact

2NF:

1: Pa_ID, Pa_Ammount , **C_ID**
2: C_ID, C_Contact

3NF:

There is no transitive dependency. Relation already in 3NF.
1: Pa_ID, Pa_Ammount , **C_ID**
2: C_ID, C_Contact

Table Creation

1: Pa_ID, Pa_Ammount , **C_ID**
2: C_ID, C_Contact

Temporary Tables

1: M_ID,M_Name ,M_Contact

2: Order_ID, Order_Loc ,C_Name ,**M_ID**

~~**3:** Order_ID , Order_Loc, C_Name~~

4: P_ID , P_Name , P_Weight , **Order_ID**

~~**5:** Order_ID , Order_Loc, C_Name~~

6: R_ID , R_Loc, R_time, **Order_ID**

~~**7:** R_ID , R_Loc, R_time, **D_ID**~~

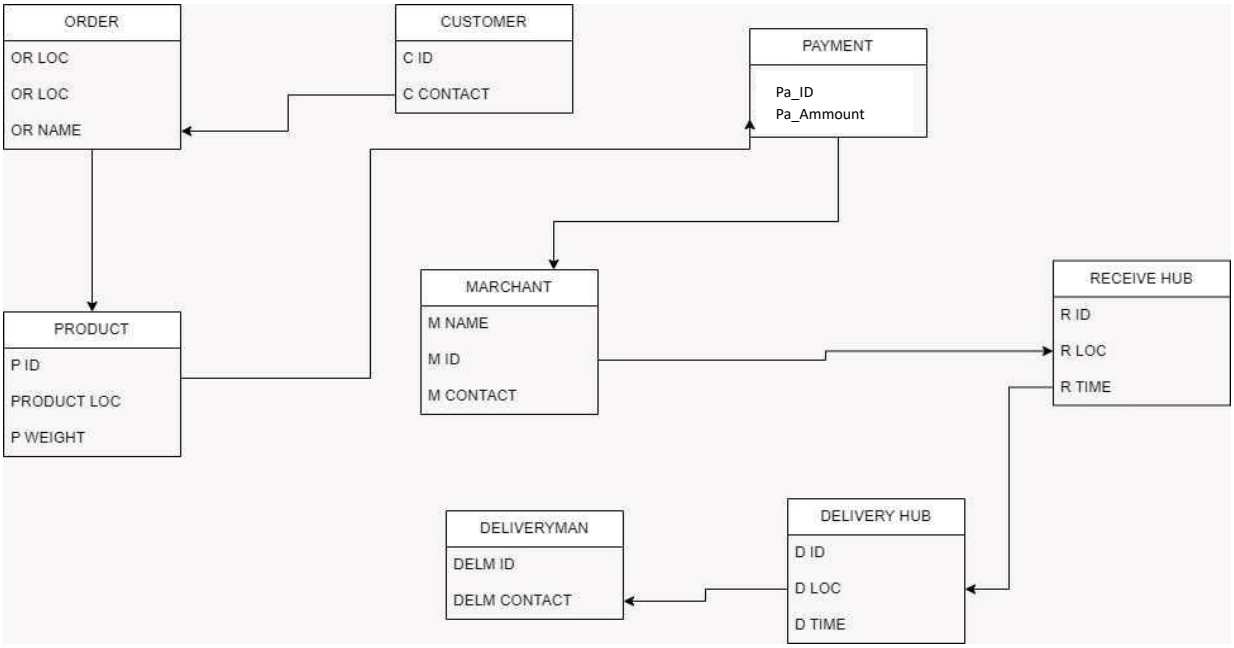
~~**8:** D_ID , D_Loc, D_Time~~

9: D_ID , D_Loc, D_Time , **Deliv_ID**
~~10: Deliv_ID , Deliv_Contact~~
11: C_ID , C_Contact
12: Deliv_ID ,Deliv_Contact, **C_ID**
13: Pa_ID, Pa_Ammount , **C_ID**
~~14: C_ID, C_Contact~~

Final Table:

1: M_ID,M_Name ,M_Contact
2: Order_ID, Order_Loc ,C_Name ,**M_ID**
3: P_ID , P_Name , P_Weight , **Order_ID**
4: R_ID , R_Loc, R_time, **Order_ID**
5: D_ID , D_Loc, D_Time , **Deliv_ID**
6: C_ID , C_Contact
7: Deliv_ID ,Deliv_Contact, **C_ID**
8: Pa_ID, Pa_Ammount , **C_ID**

Schema Diagram



Create Table

User Account Creation

CREATE USER courier IDENTIFIED BY service;
GRANT CONNECT, RESOURCE TO courier ;
ALTER USER courier QUOTA UNLIMITED ON users;

Marchent Table

Table Creation

CREATE TABLE Marchent (
M_ID INT PRIMARY KEY,
M_Name VARCHAR(255),
M_Contact VARCHAR(255)
);

Object Type **TABLE** Object **MARCHENT**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
MARCHENT	M_ID	Number	-	-	0	1	-	-	-
	M_NAME	Varchar2	255	-	-	-	✓	-	-
	M_CONTACT	Varchar2	255	-	-	-	✓	-	-
									1 - 3

Index

CREATE INDEX idx_marchent_name ON Marchent (M_Name);
SELECT index_name, column_name, column_position
FROM all_ind_columns
WHERE table_name = 'MARCHENT';

INDEX_NAME	COLUMN_NAME	COLUMN_POSITION
SYS_C006471	M_ID	1
IDX_MARCHENT_NAME	M_NAME	1

2 rows returned in 0.00 seconds CSV Export

Order Table

Table Creation

CREATE TABLE Orders (
Order_ID INT PRIMARY KEY,
Order_Loc VARCHAR(255),
C_Name VARCHAR(255),
M_ID INT,
FOREIGN KEY (M_ID) REFERENCES Marchent(M_ID)
);
Describe orders;

Object Type **TABLE** Object **ORDERS**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
ORDERS	ORDER_ID	Number	-	-	0	1	-	-	-
	ORDER_LOC	Varchar2	255	-	-	-	✓	-	-
	C_NAME	Varchar2	255	-	-	-	✓	-	-
	M_ID	Number	-	-	0	-	✓	-	-
									1 - 4

INDEX

CREATE INDEX idx_orders_cname ON Orders (C_Name);
SELECT index_name, table_name
FROM user_indexes
WHERE table_name = 'ORDERS';

INDEX_NAME	TABLE_NAME
SYS_C006472	ORDERS
IDX_ORDERS_CNAME	ORDERS

2 rows returned in 0.01 seconds CSV Export

Product Table

Create Table

CREATE TABLE Product (
P_ID INT PRIMARY KEY,
P_Name VARCHAR(255),
P_Weight DECIMAL(10, 2),
Order_ID INT,
FOREIGN KEY (Order_ID) REFERENCES Orders(Order_ID)
);
Describe product;

Object Type **TABLE** Object **PRODUCT**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PRODUCT	P_ID	Number	-	-	0	1	-	-	-
	P_NAME	Varchar2	255	-	-	-	✓	-	-
	P_WEIGHT	Number	-	10	2	-	✓	-	-
	ORDER_ID	Number	-	-	0	-	✓	-	-
									1 - 4

Index

CREATE INDEX idx_product_order_id ON Product (Order_ID);
SELECT index_name, column_name FROM all_ind_columns WHERE table_name = 'PRODUCT';

INDEX_NAME	COLUMN_NAME
SYS_C006474	P_ID
IDX_PRODUCT_ORDER_ID	ORDER_ID

2 rows returned in 0.05 seconds CSV Export

Receive_Hub Table

Table Creation

```
CREATE TABLE Receive_Hub (
R_ID INT PRIMARY KEY,
R_Loc VARCHAR(255),
R_time TIMESTAMP,
Order_ID INT,
FOREIGN KEY (Order_ID) REFERENCES Orders(Order_ID)
);
Describe Receive_Hub;
```

Object Type **TABLE** Object **RECEIVE_HUB**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
RECEIVE_HUB	R_ID	Number	-	-	0	1	-	-	-
	R_LOC	Varchar2	255	-	-	-	✓	-	-
	R_TIME	Timestamp(6)	11	-	6	-	✓	-	-
	ORDER_ID	Number	-	-	0	-	✓	-	-

1 - 4

INDEX

```
CREATE INDEX idx_receivehub_oid ON Receive_Hub (Order_ID);
SELECT index_name, table_name
FROM user_indexes
WHERE index_name = 'IDX_RECEIVEHUB_OID' AND table_name = 'RECEIVE_HUB';
```

INDEX_NAME	TABLE_NAME
IDX_RECEIVEHUB_OID	RECEIVE_HUB

1 rows returned in 0.02 seconds CSV Export

Delivery_Hub Table

Table Creation

```
CREATE TABLE Delivery_Hub (
D_ID INT PRIMARY KEY,
D_Loc VARCHAR(255),
 D_Time TIMESTAMP
);
```

Object Type **TABLE** Object **DELIVERY_HUB**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DELIVERY_HUB	D_ID	Number	-	-	0	1	-	-	-
	D_LOC	Varchar2	255	-	-	-	✓	-	-
	D_TIME	Timestamp(6)	11	-	6	-	✓	-	-

1 - 3

INDEX

```
CREATE INDEX idx_deliveryhub_loc ON Delivery_Hub (D_Loc);
SELECT index_name, table_name
FROM user_indexes
WHERE index_name = 'IDX_DELIVERYHUB_LOC' AND table_name = 'DELIVERY_HUB';
```

INDEX_NAME	TABLE_NAME
IDX_DELIVERYHUB_LOC	DELIVERY_HUB

1 rows returned in 0.02 seconds CSV Export

Delivery_man Table

Table Creation

```
CREATE TABLE Delivery_man (
Deliv_ID INT PRIMARY KEY,
Deliv_Contact VARCHAR(255),
D_ID INT,
FOREIGN KEY (D_ID) REFERENCES Delivery_Hub(D_ID)
);
describe Delivery_man;
```

Object Type **TABLE** Object **DELIVERY_MAN**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DELIVERY_MAN	DELIV_ID	Number	-	-	0	1	-	-	-
	DELIV_CONTACT	Varchar2	255	-	-	-	✓	-	-
	D_ID	Number	-	-	0	-	✓	-	-
									1 - 3

INDEX

```
SELECT index_name, table_name
FROM user_indexes
WHERE index_name = 'IDX_DELIVERYMAN_DID' AND table_name = 'DELIVERY_MAN';
```

INDEX_NAME	TABLE_NAME
IDX_DELIVERYMAN_DID	DELIVERY_MAN

1 rows returned in 0.02 seconds CSV Export

Customer Table

Table Creation

```
CREATE TABLE Customer (
C_ID INT PRIMARY KEY,
C_Contact VARCHAR(255)
);
```

Object Type **TABLE** Object **CUSTOMER**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
CUSTOMER	C_ID	Number	-	-	0	1	-	-	-
	C_CONTACT	Varchar2	255	-	-	-	✓	-	-
									1 - 2

INDEX

CREATE INDEX idx_customer_contact ON Customer (C_Contact);
SELECT index_name, table_name
FROM user_indexes
WHERE index_name = 'IDX_CUSTOMER_CONTACT' AND table_name = 'CUSTOMER';

INDEX_NAME	TABLE_NAME
IDX_CUSTOMER_CONTACT	CUSTOMER

1 rows returned in 0.00 seconds CSV Export

Payment Table

Table Creation

CREATE TABLE Payment (
Pa_ID INT PRIMARY KEY,
Pa_Ammount DECIMAL(10, 2),
C_ID INT,
FOREIGN KEY (C_ID) REFERENCES Customer(C_ID)
);

Object Type **TABLE** Object **PAYMENT**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PAYMENT	PA_ID	Number	-	-	0	1	-	-	-
	PA_AMMOUNT	Number	-	10	2	-	✓	-	-
	C_ID	Number	-	-	0	-	✓	-	-
									1 - 3

INDEX

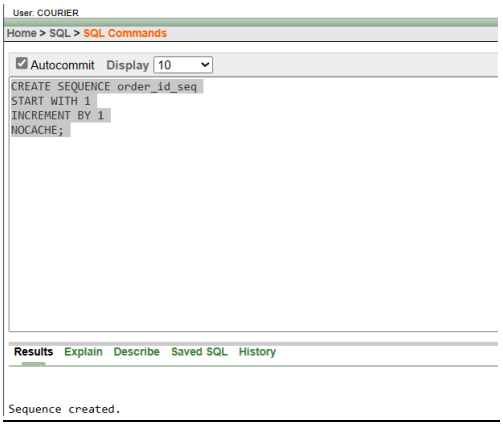
CREATE INDEX idx_payment_cid ON Payment (C_ID);
SELECT index_name, table_name
FROM user_indexes
WHERE index_name = 'IDX_PAYMENT_CID' AND table_name = 'PAYMENT';

INDEX_NAME	TABLE_NAME
IDX_PAYMENT_CID	PAYMENT

1 rows returned in 0.00 seconds CSV Export

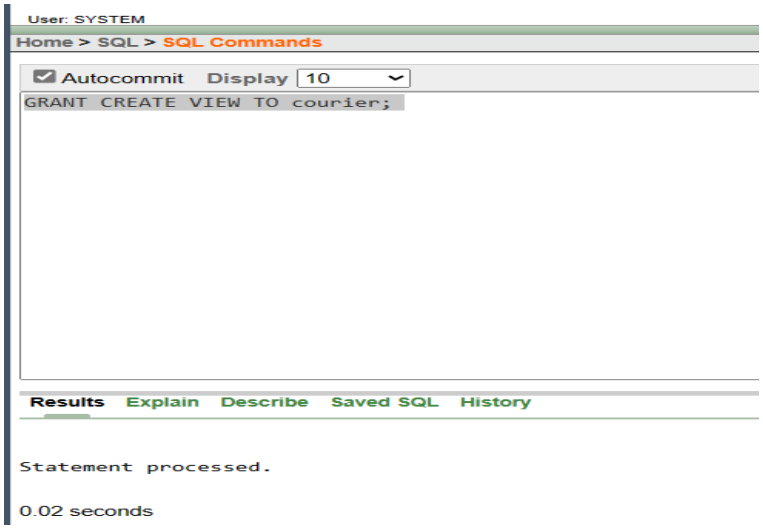
Create Sequence

CREATE SEQUENCE order_id_seq
START WITH 1
INCREMENT BY 1
NOCACHE;

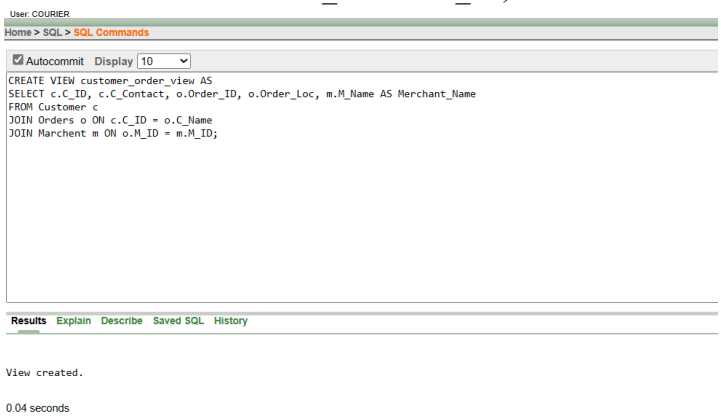


Create View

Grant Create view to courier;

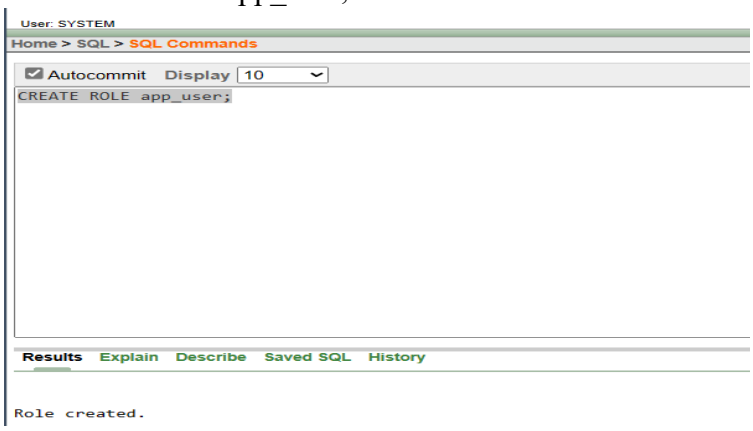


CREATE VIEW customer_order_view AS
SELECT c.C_ID, c.C_Contact, o.Order_ID, o.Order_Loc, m.M_Name AS Merchant_Name
FROM Customer c
JOIN Orders o ON c.C_ID = o.C_Name
JOIN Marchent m ON o.M_ID = m.M_ID;



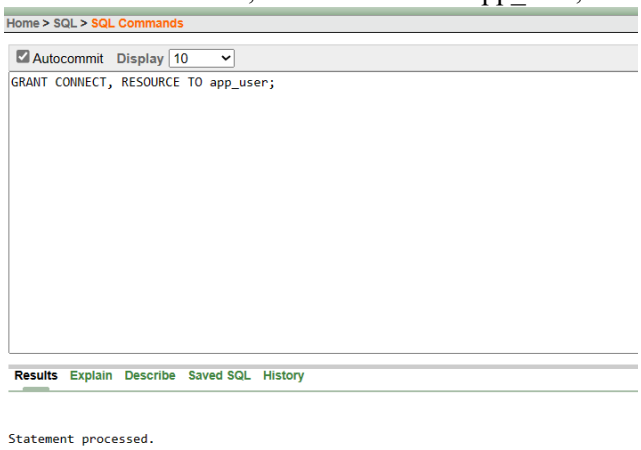
Role Create

CREATE ROLE app_user;



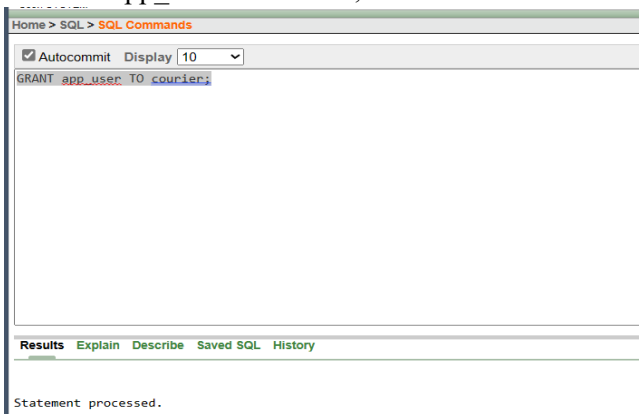
Grant privileges to the role

GRANT CONNECT, RESOURCE TO app_user;



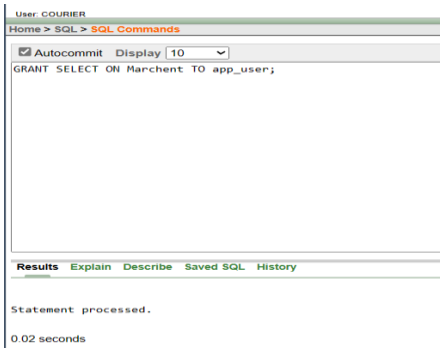
Assign the role to the user

GRANT app_user TO courier;

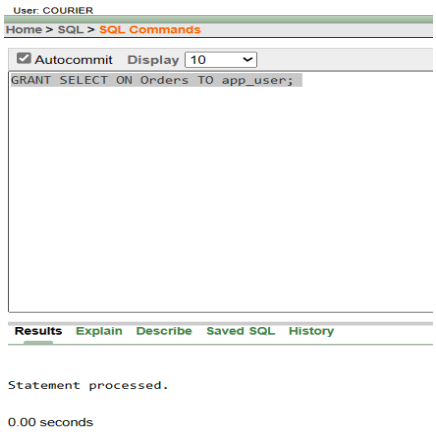


Grant Privileges

GRANT SELECT ON Marchent TO app_user;



GRANT SELECT ON Orders TO app_user;



Value Insertion

Value Insertion on Marchent Table

INSERT INTO Marchent (M_ID, M_Name, M_Contact) VALUES(1, 'Merchant 1', '125628');
INSERT INTO Marchent (M_ID, M_Name, M_Contact) VALUES(2, 'Merchant 2', '978965');
INSERT INTO Marchent (M_ID, M_Name, M_Contact) VALUES(3, 'Merchant 3', '285686');
INSERT INTO Marchent (M_ID, M_Name, M_Contact) VALUES(4, 'Merchant 4', '897514');
INSERT INTO Marchent (M_ID, M_Name, M_Contact) VALUES(5, 'Merchant 5', '894678');
select * from marchent;

M_ID	M_NAME	M_CONTACT
1	Merchant 1	125628
2	Merchant 2	978965
3	Merchant 3	285686
4	Merchant 4	897514
5	Merchant 5	894678

5 rows returned in 0.00 seconds

Value Insertion on Order Table

INSERT INTO Orders (Order_ID, Order_Loc, C_Name, M_ID)VALUES(1, 'Dhaka', 'Maruf', 1);
INSERT INTO Orders (Order_ID, Order_Loc, C_Name, M_ID)VALUES(2, 'Chittagong', 'Tahmid', 2);
INSERT INTO Orders (Order_ID, Order_Loc, C_Name, M_ID)VALUES(3, 'Cumilla', 'Raiyan', 3);
INSERT INTO Orders (Order_ID, Order_Loc, C_Name, M_ID)VALUES(4, 'Barishal', 'Rafi', 4);
INSERT INTO Orders (Order_ID, Order_Loc, C_Name, M_ID)VALUES(5, 'Rajshahi', 'Shaown', 5);

ORDER_ID	ORDER_LOC	C_NAME	M_ID
1	Dhaka	Maruf	1
2	Chittagong	Tahmid	2
3	Cumilla	Raiyan	3
4	Barishal	Rafi	4
5	Rajshahi	Shaown	5

5 rows returned in 0.01 seconds CSV Export

Value Insertion on Product table

INSERT INTO Product (P_ID, P_Name, P_Weight, Order_ID)VALUES(1, 'Shirt', 2.5, 1);
INSERT INTO Product (P_ID, P_Name, P_Weight, Order_ID)VALUES(2, 'Pant', 1.8, 1);
INSERT INTO Product (P_ID, P_Name, P_Weight, Order_ID)VALUES(3, 'Blazer', 5.0, 2);
INSERT INTO Product (P_ID, P_Name, P_Weight, Order_ID)VALUES(4, 'Trimmer', 3.2, 2);
INSERT INTO Product (P_ID, P_Name, P_Weight, Order_ID)VALUES(5, 'T-Shirt', 0.9, 3);

Results Explain Describe Saved SQL History

P_ID	P_NAME	P_WEIGHT	ORDER_ID
1	Shirt	2.5	1
2	Pant	1.8	1
3	Blazer	5	2
4	Trimmer	3.2	2
5	T-Shirt	.9	3

5 rows returned in 0.00 seconds CSV Export

Value Insertion on Receive_Hub Table

INSERT INTO Receive_Hub (R_ID, R_Loc, R_time, Order_ID)VALUES(1, 'Khulna', TIMESTAMP '2023-08-25 10:00:00', 1);
INSERT INTO Receive_Hub (R_ID, R_Loc, R_time, Order_ID)VALUES(2, 'Dhaka', TIMESTAMP '2023-08-25 11:30:00', 2);
INSERT INTO Receive_Hub (R_ID, R_Loc, R_time, Order_ID)VALUES(3, 'Chittagong', TIMESTAMP '2023-08-25 12:45:00', 3);
INSERT INTO Receive_Hub (R_ID, R_Loc, R_time, Order_ID)VALUES(4, 'Rajshahi', TIMESTAMP '2023-08-25 14:15:00', 4);
INSERT INTO Receive_Hub (R_ID, R_Loc, R_time, Order_ID)VALUES(5, 'Barishal', TIMESTAMP '2023-08-25 15:30:00', 5);
select * from Receive_Hub;

R_ID	R_LOC	R_TIME	ORDER_ID
1	Khulna	25-AUG-23 10.00.00.000000 AM	1
2	Dhaka	25-AUG-23 11.30.00.000000 AM	2
3	Chittagong	25-AUG-23 12.45.00.000000 PM	3
4	Rajshahi	25-AUG-23 02.15.00.000000 PM	4
5	Barishal	25-AUG-23 03.30.00.000000 PM	5

5 rows returned in 0.02 seconds CSV Export

Value Insertion on Delivery_Hub Table

INSERT INTO Delivery_Hub (D_ID, D_Loc, D_Time)VALUES(1, 'Dhaka', TIMESTAMP '2023-08-27 08:00:00');
INSERT INTO Delivery_Hub (D_ID, D_Loc, D_Time)VALUES(2, 'Chittagong', TIMESTAMP '2023-08-26 09:30:00');
INSERT INTO Delivery_Hub (D_ID, D_Loc, D_Time)VALUES(3, 'Cumilla', TIMESTAMP '2023-08-28 11:15:00');
INSERT INTO Delivery_Hub (D_ID, D_Loc, D_Time)VALUES(4, 'Barishal', TIMESTAMP '2023-08-29 13:00:00');
INSERT INTO Delivery_Hub (D_ID, D_Loc, D_Time)VALUES (5, 'Rajshahi', TIMESTAMP '2023-08-30 15:30:00');
Select * from Delivery_Hub;

D_ID	D_LOC	D_TIME
1	Dhaka	27-AUG-23 08.00.00.000000 AM
2	Chittagong	26-AUG-23 09.30.00.000000 AM
3	Cumilla	28-AUG-23 11.15.00.000000 AM
4	Barishal	29-AUG-23 01.00.00.000000 PM
5	Rajshahi	30-AUG-23 03.30.00.000000 PM

5 rows returned in 0.00 seconds CSV Export

Value Insertion on Delivery_Man Table

INSERT INTO Delivery_man (Deliv_ID, Deliv_Contact, D_ID)VALUES(1, '87921', 1);
INSERT INTO Delivery_man (Deliv_ID, Deliv_Contact, D_ID)VALUES(2, '59519', 2);
INSERT INTO Delivery_man (Deliv_ID, Deliv_Contact, D_ID)VALUES(3, '89716', 3);
INSERT INTO Delivery_man (Deliv_ID, Deliv_Contact, D_ID)VALUES(4, '78914', 4);
INSERT INTO Delivery_man (Deliv_ID, Deliv_Contact, D_ID)VALUES(5, '12378', 5);
select * from Delivery_man ;

DELIV_ID	DELIV_CONTACT	D_ID
1	87921	1
2	59519	2
3	89716	3
4	78914	4
5	12378	5

5 rows returned in 0.00 seconds CSV Export

Value Insertion on Customer Table

INSERT INTO Customer (C_ID, C_Contact)VALUES(1, '87945');
INSERT INTO Customer (C_ID, C_Contact)VALUES(2, '89721');
INSERT INTO Customer (C_ID, C_Contact)VALUES(3, '12478');
INSERT INTO Customer (C_ID, C_Contact)VALUES(4, '12783');
INSERT INTO Customer (C_ID, C_Contact)VALUES(5, '88888');
select * from Customer;

C_ID	C_CONTACT
1	87945
2	89721
3	12478
4	12783
5	88888

5 rows returned in 0.00 seconds

Value Insertion on Payment Table

INSERT INTO Payment (Pa_ID, Pa_Ammount, C_ID)VALUES(1, 100.50, 1);
INSERT INTO Payment (Pa_ID, Pa_Ammount, C_ID)VALUES(2, 75.20, 2);
INSERT INTO Payment (Pa_ID, Pa_Ammount, C_ID)VALUES(3, 200.00, 3);
INSERT INTO Payment (Pa_ID, Pa_Ammount, C_ID)VALUES(4, 50.75, 4);
INSERT INTO Payment (Pa_ID, Pa_Ammount, C_ID)VALUES(5, 150.00, 5);
select * from Payment;

PA_ID	PA_AMMOUNT	C_ID
1	100.5	1
2	75.2	2
3	200	3
4	50.75	4
5	150	5

5 rows returned in 0.00 seconds CSV Export

SQL

Single Row Function

1. Retrieve the length of each merchant's name.

   Answer:

   SELECT M_Name, LENGTH(M_Name) AS Name_Length
   FROM Marchent;

M_NAME	NAME_LENGTH
Merchant 1	10
Merchant 2	10
Merchant 3	10
Merchant 4	10
Merchant 5	10

5 rows returned in 0.05 seconds CSV Export

2. Display the merchant names in lowercase and with leading/trailing spaces removed.

   Answer:

   SELECT M_Name, TRIM(LOWER(M_Name)) AS Cleaned_Name
   FROM Marchent;

M_NAME	CLEANED_NAME
Merchant 1	merchant 1
Merchant 2	merchant 2
Merchant 3	merchant 3
Merchant 4	merchant 4
Merchant 5	merchant 5

5 rows returned in 0.00 seconds CSV Export

3. Calculate the square root of the product weight for each product.

   Answer:

   SELECT P_Name, P_Weight, SQRT(P_Weight) AS Weight_SquareRoot
   FROM Product;

P_NAME	P_WEIGHT	WEIGHT_SQUAREROOT
Shirt	2.5	1.581138830084189665999446772216359266 86
Pant	1.8	1.341640786499873817845504201238765741 26
Blazer	5	2.236067977499789696409173668731276235 44
Trimmer	3.2	1.788854381999831757127338934985020988 35
T-Shirt	.9	.948683298050513799599668063329815560116

5 rows returned in 0.02 seconds CSV Export

Group Function

Question: Calculate the total weight of all products.
Answer:
SELECT SUM(P_Weight) AS Total_Weight
FROM Product;

TOTAL_WEIGHT
13.4

1 rows returned in 0.02 seconds CSV Export

Question: Calculate the average order ID.
Answer:
SELECT AVG(Order_ID) AS Average_Order_ID
FROM Orders;

Results Explain Describe Saved SQL History

AVERAGE_ORDER_ID
3

1 rows returned in 0.00 seconds CSV Export

Question: Find the maximum and minimum order IDs in the "Orders" table.
Answer:
SELECT MAX(Order_ID) AS Max_Order_ID, MIN(Order_ID) AS Min_Order_ID
FROM Orders;

MAX_ORDER_ID	MIN_ORDER_ID
5	1

1 rows returned in 0.00 seconds CSV Export

Sub Query

Question: Retrieve all orders placed by customers with a specific contact number.
Answer:
```
SELECT *
FROM Orders
WHERE C_Name IN
   (SELECT C_Name FROM Customer
    WHERE C_Contact = '87945');
```

ORDER_ID	ORDER_LOC	C_NAME	M_ID
1	Dhaka	Maruf	1
2	Chittagong	Tahmid	2
3	Cumilla	Raiyan	3
4	Barishal	Rafi	4
5	Rajshahi	Shaown	5

5 rows returned in 0.00 seconds CSV Export

Question: Find merchants whose IDs match those of orders.
Answer:
```
SELECT *
FROM Marchent
WHERE M_ID IN
(SELECT M_ID FROM Orders);
```

ORDER_ID	ORDER_LOC	C_NAME	M_ID
1	Dhaka	Maruf	1
2	Chittagong	Tahmid	2
3	Cumilla	Raiyan	3
4	Barishal	Rafi	4
5	Rajshahi	Shaown	5

5 rows returned in 0.00 seconds CSV Export

Question: Retrieve the names of merchants who have received orders from customer 'Raiyan'.
Answer:
```
SELECT M_Name
FROM Marchent
WHERE M_ID IN (
   SELECT M_ID
   FROM Orders
   WHERE C_Name = 'Raiyan'
);
```

M_NAME
Merchant 3

1 rows returned in 0.00 seconds

Joining

Question: Get Merchant Name and Total Order Count for Merchants with Orders
Answer:

```
SELECT m.M_Name, COUNT(o.Order_ID) AS Total_Orders
FROM Marchent m
JOIN Orders o ON m.M_ID = o.M_ID
GROUP BY m.M_Name;
```

M_NAME	TOTAL_ORDERS
Merchant 5	1
Merchant 4	1
Merchant 1	1
Merchant 2	1
Merchant 3	1

5 rows returned in 0.00 seconds CSV Export

Question: Get the Merchant Names and Order Locations, including orders without associated merchants.
Answer:

```
SELECT Marchent.M_Name, Orders.Order_Loc
FROM Marchent
LEFT JOIN Orders ON Marchent.M_ID = Orders.M_ID;
```

M_NAME	ORDER_LOC
Merchant 1	Dhaka
Merchant 2	Chittagong
Merchant 3	Cumilla
Merchant 4	Barishal
Merchant 5	Rajshahi

5 rows returned in 0.00 seconds CSV Export

Question: Display the Merchant Names and Receiving Locations for all data, even if there are no matches between Marchent and Receive_Hub.
Answer:

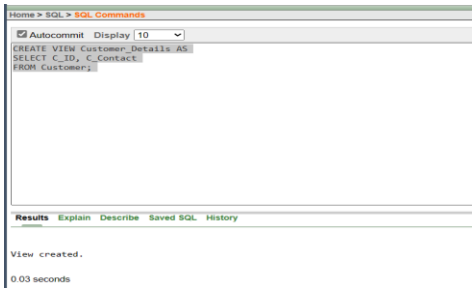
```
SELECT Marchent.M_Name, Receive_Hub.R_Loc
FROM Marchent
FULL OUTER JOIN Receive_Hub ON Marchent.M_ID = Receive_Hub.Order_ID;
```

ORDER_ID	R_LOC
1	Khulna
2	Dhaka
3	Chittagong
4	Rajshahi
5	Barishal

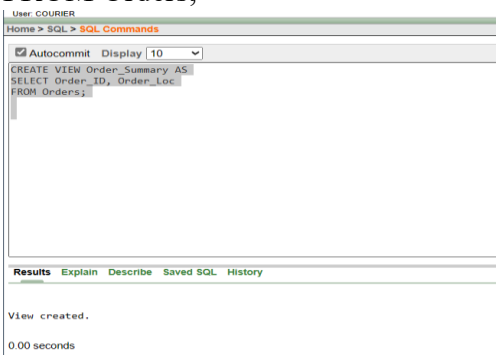
5 rows returned in 0.00 seconds CSV Export

View

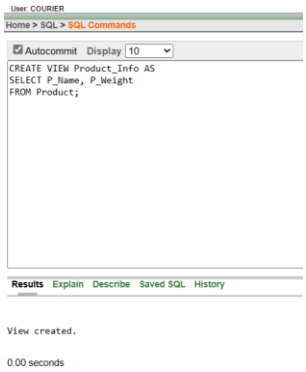
Question: Create a view that displays the names and contact numbers of customers.
Answer:
CREATE VIEW Customer_Details AS
SELECT C_ID, C_Contact
FROM Customer;



Question: Create a view that shows the order IDs and their respective order locations.
Answer:
CREATE VIEW Order_Summary AS
SELECT Order_ID, Order_Loc
FROM Orders;



Question: Create a view that provides the names and weights of products.
Answer:
CREATE VIEW Product_Info AS
SELECT P_Name, P_Weight
FROM Product;



Synonym

Grant create synonym to COURIER;

```
Grant create synonym to COURIER;
```

Results Explain Describe Saved SQL History

```
Statement processed.
```

Question: Create a synonym named "OrderLocationSyn" for the "Order_Loc" column in the "Orders" table.

Answer: CREATE SYNONYM OrderLocationSyn FOR Orders.Order_Loc;

Autocommit Display 10

```
CREATE SYNONYM OrderLocationSyn FOR Orders.Order_Loc;
```

Results Explain Describe Saved SQL History

```
Synonym created.
```

0.01 seconds

Question: Create a synonym named "CustomerTableSyn" for the "Customer" table.

Answer: CREATE SYNONYM CustomerTableSyn FOR Customer;

Autocommit Display 10

```
CREATE SYNONYM CustomerTableSyn FOR Customer;
```

Results Explain Describe Saved SQL History

```
Synonym created.
```

0.00 seconds

PL/SQL
Function

Question: Create function to calculate the total weight of products in a specific order.

Answer:

```
CREATE OR REPLACE FUNCTION calculate_total_weight(order_id INT) RETURN
DECIMAL IS
  total_weight DECIMAL := 0;
BEGIN
  FOR product_rec IN (SELECT P_Weight FROM Product WHERE Order_ID = order_id)
LOOP
    total_weight := total_weight + product_rec.P_Weight;
  END LOOP;
  RETURN total_weight;
END;
/
```

Autocommit Display 10

```
CREATE OR REPLACE FUNCTION calculate_total_weight(order_id INT) RETURN DECIMAL IS
  total_weight DECIMAL := 0;
BEGIN
  FOR product_rec IN (SELECT P_Weight FROM Product WHERE Order_ID = order_id) LOOP
    total_weight := total_weight + product_rec.P_Weight;
  END LOOP;

  RETURN total_weight;
END;
/
```

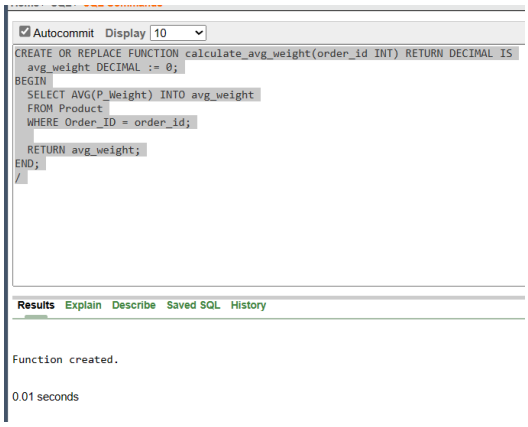
Results Explain Describe Saved SQL History

```
Function created.
```

0.08 seconds

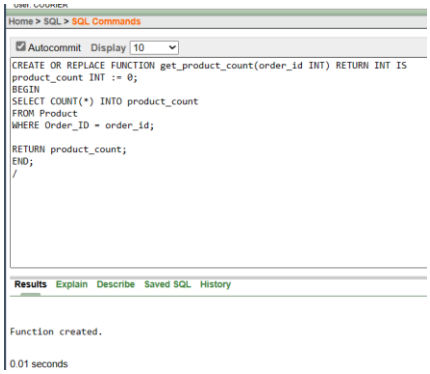
Question: Create Function to get Average Weight of Products in an Order
Answer:

```
CREATE OR REPLACE FUNCTION calculate_avg_weight(order_id INT) RETURN
DECIMAL IS
 avg_weight DECIMAL := 0;
BEGIN
SELECT AVG(P_Weight) INTO avg_weight
FROM Product
WHERE Order_ID = order_id;
RETURN avg_weight;
END;
/
```



Question: Create Function Get Product Count
Answer:

```
CREATE OR REPLACE FUNCTION get_product_count(order_id INT) RETURN INT IS
product_count INT := 0;
BEGIN
SELECT COUNT(*) INTO product_count
FROM Product
WHERE Order_ID = order_id;
RETURN product_count;
END;
/
```

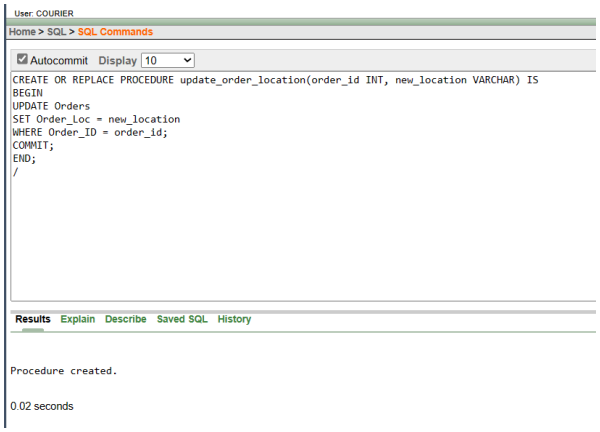


Procedure

Question: Create or Replace Procedure to Update Order Location

Answer:

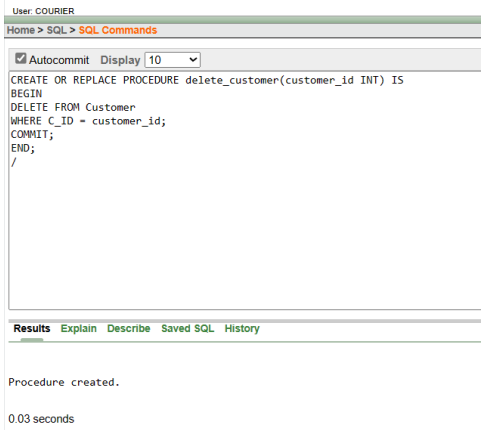
```
CREATE OR REPLACE PROCEDURE update_order_location(order_id INT, new_location
VARCHAR) IS
BEGIN
  UPDATE Orders
  SET Order_Loc = new_location
  WHERE Order_ID = order_id;
  COMMIT;
END;
/
```



Question: Create or Replace Procedure to Delete a Customer

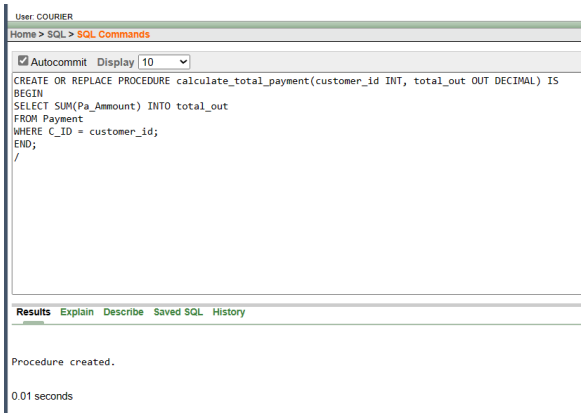
Answer:

```
CREATE OR REPLACE PROCEDURE delete_customer(customer_id INT) IS
BEGIN
DELETE FROM Customer
WHERE C_ID = customer_id;
COMMIT;
END;
/
```



Question: Create or Replace Procedure to Calculate Total Payment
Answer:

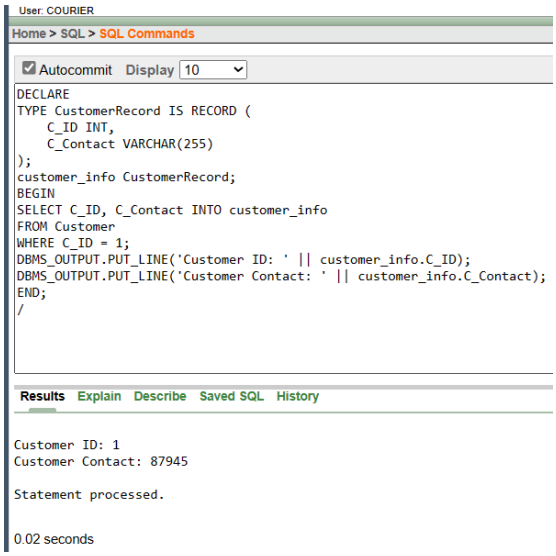
```
CREATE OR REPLACE PROCEDURE calculate_total_payment(customer_id INT,
total_out OUT DECIMAL) IS
BEGIN
SELECT SUM(Pa_Ammount) INTO total_out
FROM Payment
WHERE C_ID = customer_id;
END;
/
```



Records

Question: Record for Customer Details
Answer:

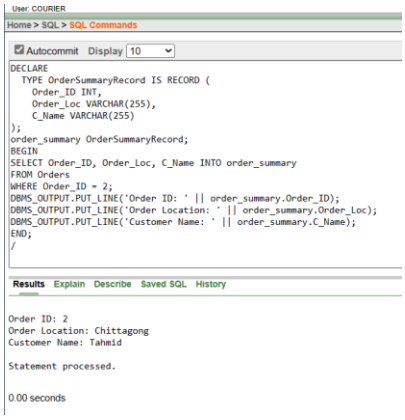
```
DECLARE
TYPE CustomerRecord IS RECORD (
   C_ID INT,
   C_Contact VARCHAR(255)
);
customer_info CustomerRecord;
BEGIN
SELECT C_ID, C_Contact INTO customer_info
FROM Customer
WHERE C_ID = 1;
DBMS_OUTPUT.PUT_LINE('Customer ID: ' || customer_info.C_ID);
DBMS_OUTPUT.PUT_LINE('Customer Contact: ' || customer_info.C_Contact);
END;
/
```



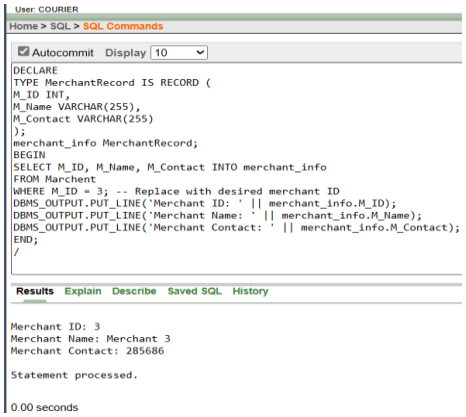
Question:**Record for Order Summary**

Answer:

```
DECLARE
  TYPE OrderSummaryRecord IS RECORD (
    Order_ID INT,
    Order_Loc VARCHAR(255),
    C_Name VARCHAR(255)
);
order_summary OrderSummaryRecord;
BEGIN
SELECT Order_ID, Order_Loc, C_Name INTO order_summary
FROM Orders
WHERE Order_ID = 2; -- Replace with desired order ID
DBMS_OUTPUT.PUT_LINE('Order ID: ' || order_summary.Order_ID);
DBMS_OUTPUT.PUT_LINE('Order Location: ' || order_summary.Order_Loc);
DBMS_OUTPUT.PUT_LINE('Customer Name: ' || order_summary.C_Name);
END;
/
```



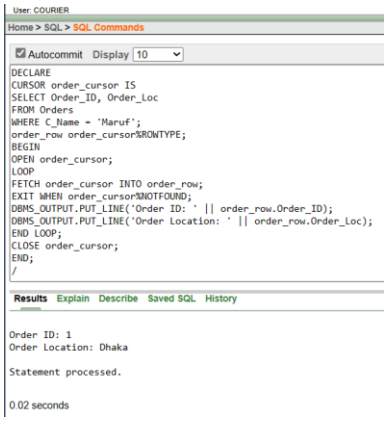
Question:**Record for Merchant Information**
Answer: DECLARE
TYPE MerchantRecord IS RECORD (
M_ID INT,
M_Name VARCHAR(255),
M_Contact VARCHAR(255)
);
merchant_info MerchantRecord;
BEGIN
SELECT M_ID, M_Name, M_Contact INTO merchant_info
FROM Marchent
WHERE M_ID = 3;
DBMS_OUTPUT.PUT_LINE('Merchant ID: ' || merchant_info.M_ID);
DBMS_OUTPUT.PUT_LINE('Merchant Name: ' || merchant_info.M_Name);
DBMS_OUTPUT.PUT_LINE('Merchant Contact: ' || merchant_info.M_Contact);
END;
/



Cursor

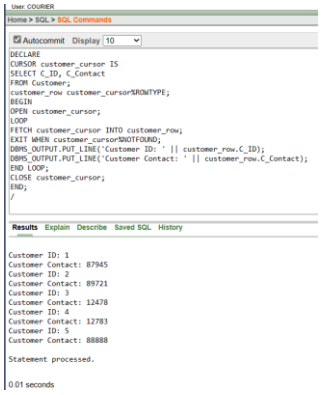
Question: Create a cursor that fetches the order IDs and locations of all orders placed by a specific customer.
Answer: DECLARE
CURSOR order_cursor IS
SELECT Order_ID, Order_Loc
FROM Orders
WHERE C_Name = 'Maruf';
order_row order_cursor%ROWTYPE;
BEGIN
OPEN order_cursor;
LOOP
FETCH order_cursor INTO order_row;
EXIT WHEN order_cursor%NOTFOUND;

```
DBMS_OUTPUT.PUT_LINE('Order ID: ' || order_row.Order_ID);
DBMS_OUTPUT.PUT_LINE('Order Location: ' || order_row.Order_Loc);
END LOOP;
CLOSE order_cursor;
END;/
```



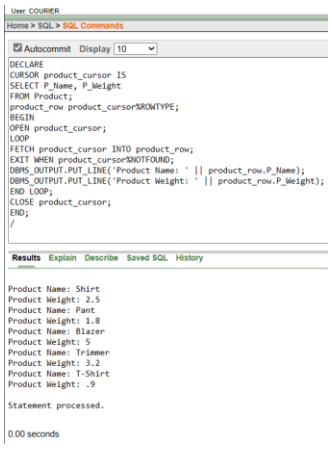
Question:Create a cursor that retrieves the names and contact numbers of all customers.

Answer:
```
DECLARE
CURSOR customer_cursor IS
SELECT C_ID, C_Contact
FROM Customer;
customer_row customer_cursor%ROWTYPE;
BEGIN
OPEN customer_cursor;
LOOP
FETCH customer_cursor INTO customer_row;
EXIT WHEN customer_cursor%NOTFOUND;
DBMS_OUTPUT.PUT_LINE('Customer ID: ' || customer_row.C_ID);
DBMS_OUTPUT.PUT_LINE('Customer Contact: ' || customer_row.C_Contact);
END LOOP;
CLOSE customer_cursor;
END;
/
```



Question: Create a cursor that fetches the names and weights of all products.
Answer:

```
DECLARE
CURSOR product_cursor IS
SELECT P_Name, P_Weight
FROM Product;
product_row product_cursor%ROWTYPE;
BEGIN
OPEN product_cursor;
LOOP
FETCH product_cursor INTO product_row;
EXIT WHEN product_cursor%NOTFOUND;
DBMS_OUTPUT.PUT_LINE('Product Name: ' || product_row.P_Name);
DBMS_OUTPUT.PUT_LINE('Product Weight: ' || product_row.P_Weight);
END LOOP;
CLOSE product_cursor;
END;
/
```

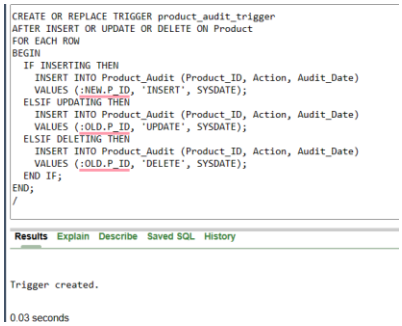


Trigger

Question: Create a trigger that logs changes in order status to an audit table whenever an order's status is updated to "Shipped."
Answer:

```
CREATE OR REPLACE TRIGGER product_audit_trigger
AFTER INSERT OR UPDATE OR DELETE ON Product
FOR EACH ROW
BEGIN
  IF INSERTING THEN
    INSERT INTO Product_Audit (Product_ID, Action, Audit_Date)
    VALUES (:NEW.P_ID, 'INSERT', SYSDATE);
  ELSIF UPDATING THEN
    INSERT INTO Product_Audit (Product_ID, Action, Audit_Date)
    VALUES (:OLD.P_ID, 'UPDATE', SYSDATE);
```

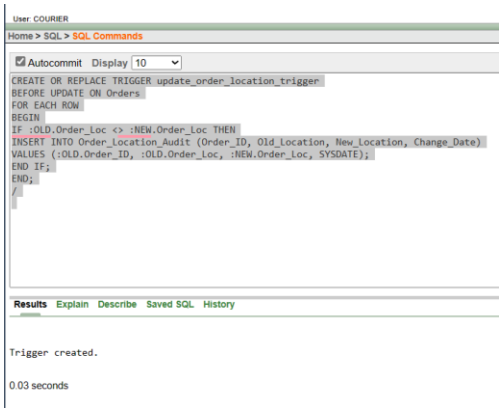
```
  ELSIF DELETING THEN
    INSERT INTO Product_Audit (Product_ID, Action, Audit_Date)
VALUES (:OLD.P_ID, 'DELETE', SYSDATE);
END IF;
END;
/
```



Question:Describe the flow of events when the UPDATE operation is executed on the Orders table for the order with Order_ID = 01. Include the roles of the :OLD and :NEW keywords in the trigger execution.

Answer:

```
CREATE OR REPLACE TRIGGER update_order_location_trigger
BEFORE UPDATE ON Orders
FOR EACH ROW
BEGIN
IF :OLD.Order_Loc <> :NEW.Order_Loc THEN
INSERT INTO Order_Location_Audit (Order_ID, Old_Location, New_Location, Change_Date)
VALUES (:OLD.Order_ID, :OLD.Order_Loc, :NEW.Order_Loc, SYSDATE);
END IF;
END;
/
```

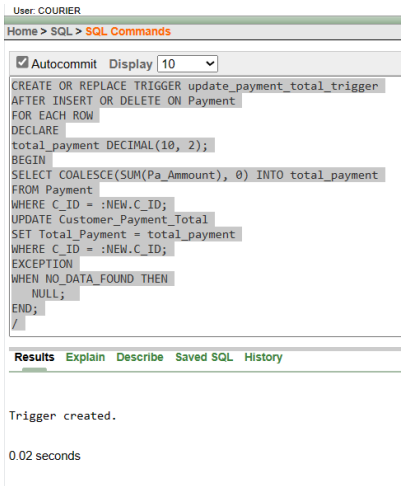


Question: Explain the purpose of the update_payment_total_trigger trigger. How does it contribute to maintaining accurate total payment records for customers?

Answer:

CREATE OR REPLACE TRIGGER update_payment_total_trigger

```
AFTER INSERT OR DELETE ON Payment
FOR EACH ROW
DECLARE
total_payment DECIMAL(10, 2);
BEGIN
SELECT COALESCE(SUM(Pa_Ammount), 0) INTO total_payment
FROM Payment
WHERE C_ID = :NEW.C_ID;
UPDATE Customer_Payment_Total
SET Total_Payment = total_payment
WHERE C_ID = :NEW.C_ID;
EXCEPTION
WHEN NO_DATA_FOUND THEN
  NULL;
END;
/
```

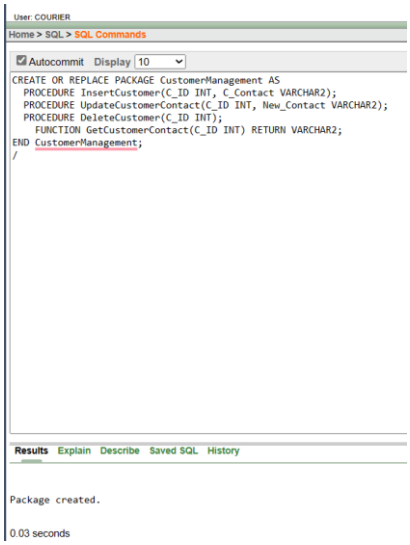


Package

Question:

Answer: Explain the purpose of the CustomerManagement package. What are the main tasks it facilitates in managing customer information?

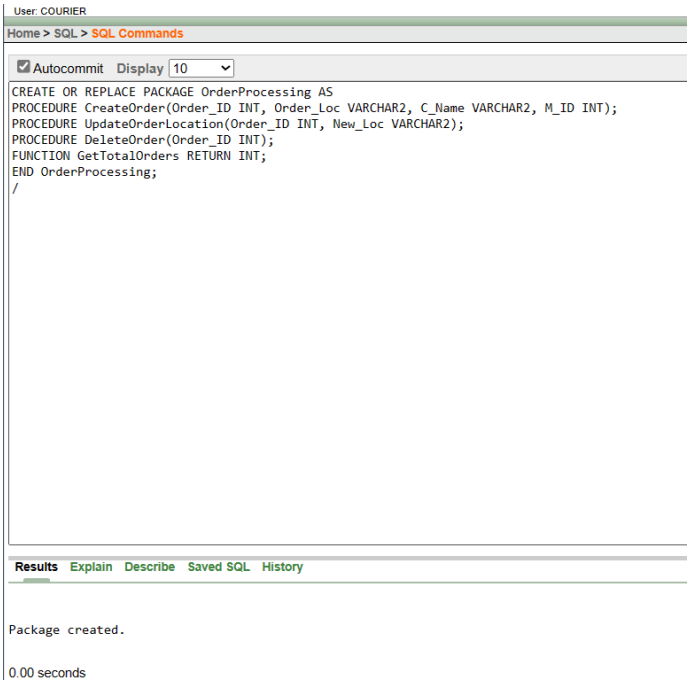
```
CREATE OR REPLACE PACKAGE CustomerManagement AS
PROCEDURE InsertCustomer(C_ID INT, C_Contact VARCHAR2);
PROCEDURE UpdateCustomerContact(C_ID INT, New_Contact VARCHAR2);
PROCEDURE DeleteCustomer(C_ID INT);
FUNCTION GetCustomerContact(C_ID INT) RETURN VARCHAR2;
END CustomerManagement;
/
```



Question: Explain the purpose of the OrderProcessing package. What are the main functionalities it offers in the context of order management?

Answer:

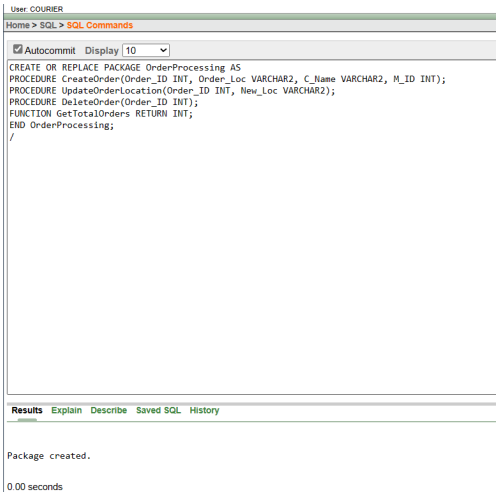
CREATE OR REPLACE PACKAGE OrderProcessing AS
PROCEDURE CreateOrder(Order_ID INT, Order_Loc VARCHAR2, C_Name VARCHAR2, M_ID INT);
PROCEDURE UpdateOrderLocation(Order_ID INT, New_Loc VARCHAR2);
PROCEDURE DeleteOrder(Order_ID INT);
FUNCTION GetTotalOrders RETURN INT;
END OrderProcessing;
/



Question: Describe the parameters of the CreateOrder procedure included in the package. How would you use this procedure to add a new order entry into the system?

Answer:

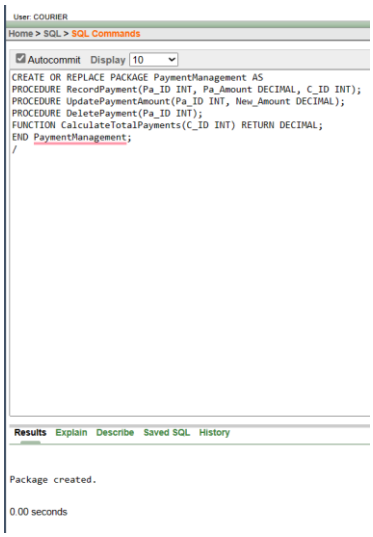
```
CREATE OR REPLACE PACKAGE OrderProcessing AS
PROCEDURE CreateOrder(Order_ID INT, Order_Loc VARCHAR2, C_Name VARCHAR2,
M_ID INT);
PROCEDURE UpdateOrderLocation(Order_ID INT, New_Loc VARCHAR2);
PROCEDURE DeleteOrder(Order_ID INT);
FUNCTION GetTotalOrders RETURN INT;
END OrderProcessing;
/
```



Question: The package contains a procedure called RecordPayment. Describe the purpose of this procedure and how it is used to capture payment information in the system.

Answer:

```
CREATE OR REPLACE PACKAGE PaymentManagement AS
PROCEDURE RecordPayment(Pa_ID INT, Pa_Amount DECIMAL, C_ID INT);
PROCEDURE UpdatePaymentAmount(Pa_ID INT, New_Amount DECIMAL);
PROCEDURE DeletePayment(Pa_ID INT);
FUNCTION CalculateTotalPayments(C_ID INT) RETURN DECIMAL;
END PaymentManagement;
/
```



Relational Algebra

Selection

How can we retrieve the details of orders with an Order_ID of 3 using the selection operation in relational algebra?

Answer: The corresponding relational algebra expression is

$$\sigma_{(Order\_ID=3)}(Orders).$$

Projection

If we want to extract only the Order_ID and Order_Loc attributes from the "Orders" table, which relational algebra operation should we use?

Answer: The required operation is the projection operation. The relational algebra expression is

$$\pi_{(Order\_ID,\ Order\_Loc)}(Orders).$$

Union

How can we obtain a combined list of distinct Order_Loc values from both the "Orders" and "Receive_Hub" tables using relational algebra?

Answer: To achieve this, we perform the union operation on the Order_Loc attributes of both tables. The relational algebra expression is

$$Orders_{(Order\_Loc)} \cup Receive\_Hub_{(Order\_Loc)}.$$

Cartesian

In order to generate all possible combinations of Order_Loc values from the "Orders" table and R_Loc values from the "Receive_Hub" table, which relational algebra operation would be appropriate?

Answer: To find all combinations, we use the Cartesian product operation. The corresponding relational algebra expression is

$$Orders_{(Order\_Loc)} \times Receive\_Hub_{(R\_Loc)}.$$

Rename

How can we create a new relation with columns named M_ID_New, M_Name_New, and M_Contact_New based on the "Marchent" table using relational algebra?

Answer: The appropriate operation here is the rename operation. We can achieve this by renaming the columns of the "Marchent" table. The relational algebra expression is

$$\rho_{(M\_ID\_New,\ M\_Name\_New,\ M\_Contact\_New)}(\text{Marchent}).$$

Conclusion:

Our project findings highlight the successful implementation of a Courier Management System, streamlining the process of order placement, routing, and delivery. Future work will focus on enhancing user experience through the development of a user-friendly interface, real-time order tracking, and integration with third-party logistics services for expanded coverage. Additionally, we aim to implement advanced analytics to optimize delivery routes, minimize delays, and offer predictive insights to merchants and customers, thereby further enhancing the efficiency and effectiveness of the system.