AMERICAN INTERNATIONALUNIVERSITY – BANGLADESH
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Course Title:Introduction to Database
Course Teacher: Juena Ahmed Noshin

Project Title: Product Swap Management System

Section: I
Semester:Spring2023-24
Date of Submission:15-05-2024

No	Name	ID	Program
1	Sumaiya Tasnim	23-50014-1	BSc[CSE]

INDEX

PRODUCT SWAP MANAGEMENT SYSTEM INTRODUCTION

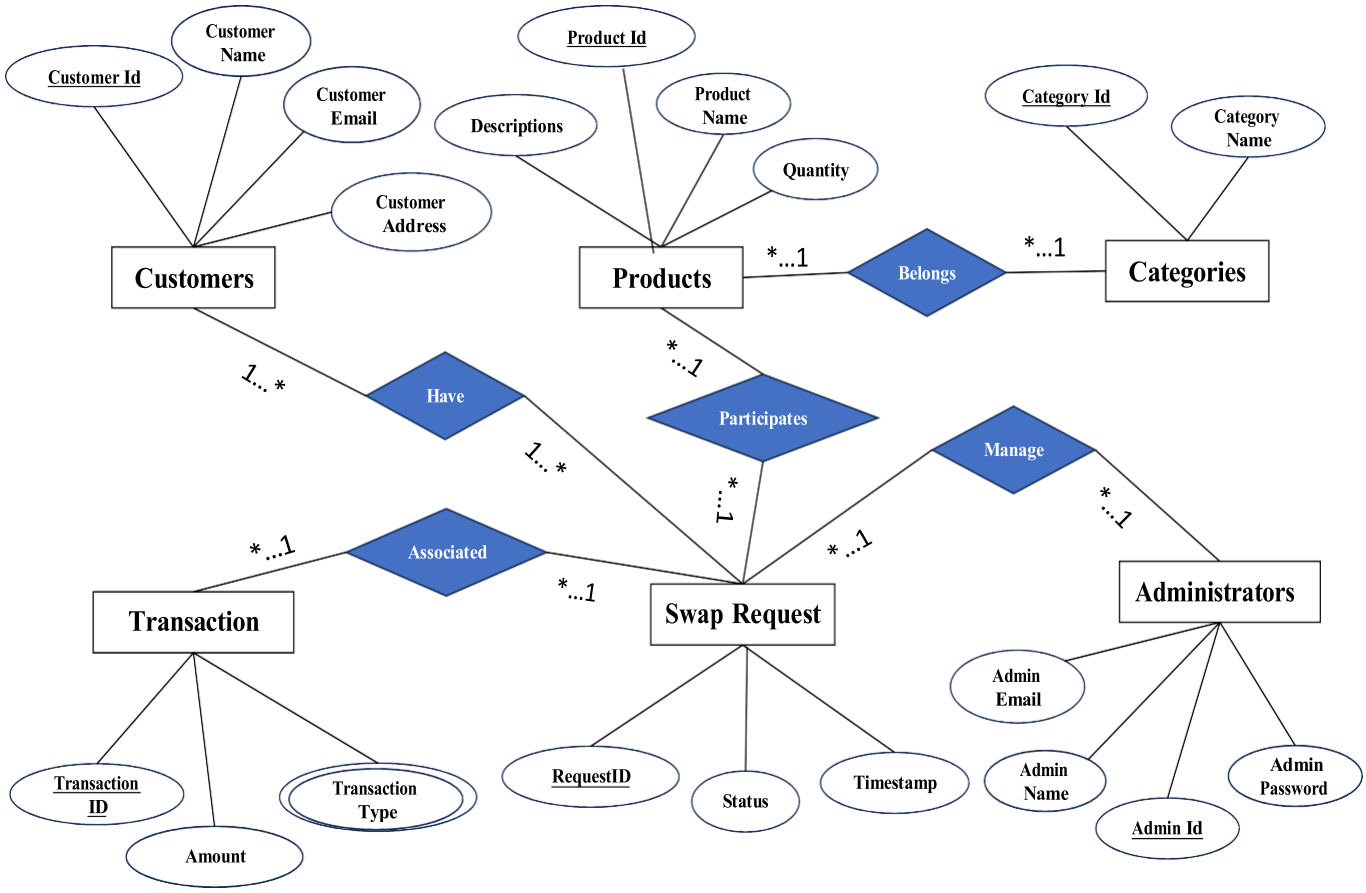
The Product Swap Management System is a comprehensive database project designed to streamline the process of swapping products between customers. This system aims to provide a user-friendly interface for both customers and administrators to facilitate efficient product exchanges, ensuring customer satisfaction and inventory management. Through this platform, users can initiate swap requests, track the status of their requests, and manage their inventory of available products for swapping.Administrators have access to advanced features for managing product listings, overseeing swap transactions, and generating insightful reports to optimize the swapping process. With its intuitive design and robust functionalities, the Product Swap Management System is poised to enhance the efficiency and effectiveness of product exchanges for businesses and consumers alike.

Product Swap Management System Scenario:

StudentID : 23-50014-1	Student Name: Sumaiya Tasnim
CO2: Understand the fundamental concepts underlying database systems and gain hands-on experience with ER diagram Case study	
PO-c2: Develop process for complex computer science and engineering problems considering cultural and societal factors.	Marks

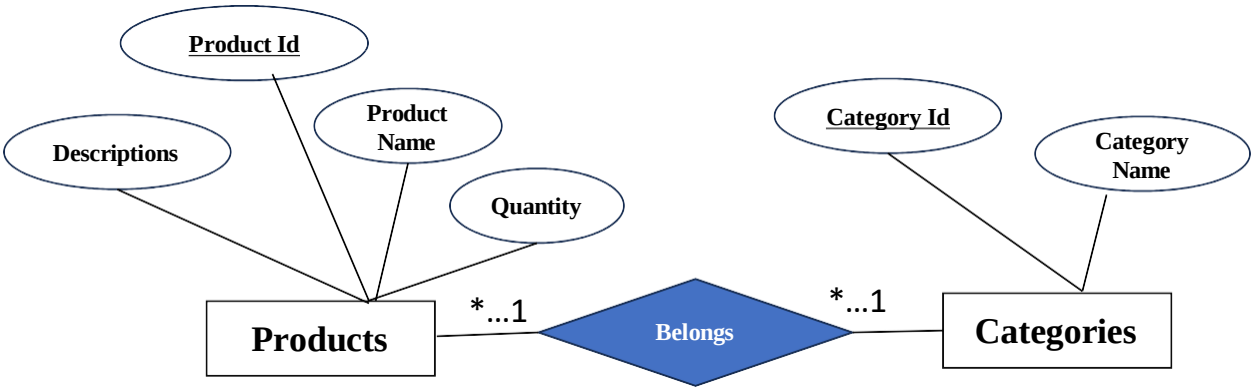
In the Product Swap Management System, customers can request swaps for various products. Eachcustomer is identified by their unique customer details such as name, email, and address. available for swapping are listed with attributes like name, description, quantity, and category. Customers can submit multiple swap requests, each detailing the product they wish to exchange and the current status of the request. Administrators oversee the system's operations and handle multiple swap requests simultaneously. Transaction history is maintained to track the details of past swaps, including the type of transaction and when it occurred. Products can participate in multiple swap requests, and the system efficiently manages product categories to aid in organizing inventory. Overall, the Product Swap Management System ensures a smooth process for customers to exchange products, improving inventory management and customer satisfaction.

Product Swap Management System – Entity Relationship Diagram(ERD)



NORMALISATION

BELONGS



UNF

Belongs(Product_id, Product_name, Quantity, Descriptions, Category_id, Category_name)

1NF

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

1.Product_id, Product_name, Quantity, Descriptions, Category_id, Category_name

2NF

1.Product_id, Product_name, Quantity, Descriptions

2.Category_id, Category_name

3NF

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

1.Product_id, Product_name, Quantity, Descriptions

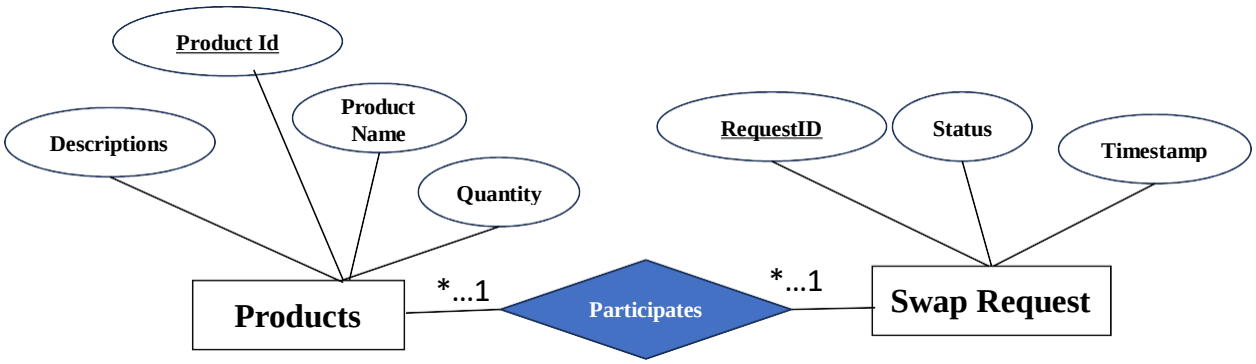
2.Category_id, Category_name

Table Creation

1.Product_id, Product_name, Quantity, Descriptions

2.Category_id, Category_name, **P_id**

PARTICIPATES



UNF

Participates (Product_id, Product_name, Quantity, Descriptions, Request_id, Status, Timestamp)

1NF

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

1.Product_id, Product_name, Quantity, Descriptions, Request_id, Status, Timestamp

2NF

1.Product_id, Product_name, Quantity, Descriptions

2.Request_id, Status, Timestamp

3NF

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

1.Product_id, Product_name, Quantity, Descriptions

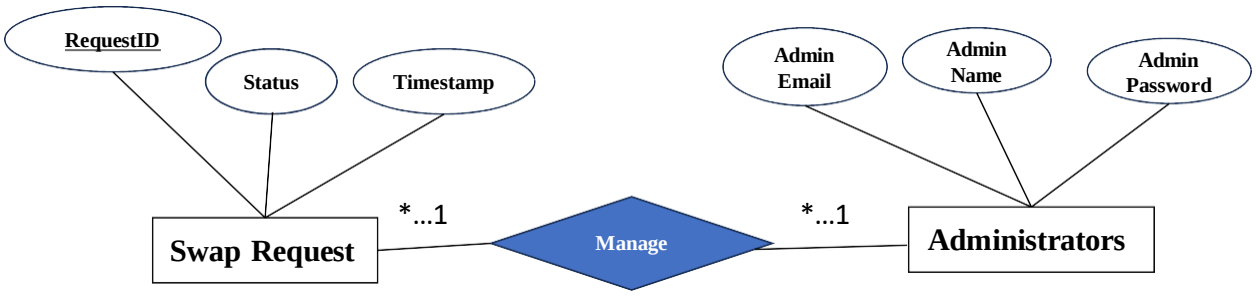
2.Request_id, Status, Timestamp

Table Creation

1.Product_id, Product_name, Quantity, Descriptions

2.Request_id, Status, Timestamp, **P_id**

MANAGE



UNF

Manage (Request_id, Status, Timestamp, Admin_id, Admin_name, Admin_password, Admin_email)

1NF

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

1. Request_id, Status, Timestamp, Admin_id, Admin_name, Admin_password, Admin_email

2NF

1. Request_id, Status, Timestamp
2. Admin_id, Admin_name, Admin_password, Admin_email

3NF

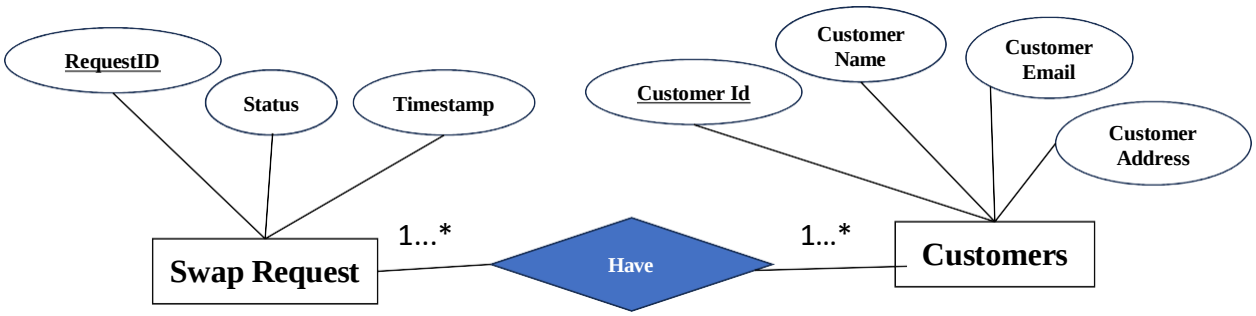
There is no transitive dependency. Relation already in 3NF.

1. Request_id, Status, Timestamp
2. Admin_id, Admin_name, Admin_password, Admin_email

Table Creation

1. Request_id, Status, Timestamp
2. Admin_id, Admin_name, Admin_password, Admin_email, **R_id**

HAVE



UNF

Have (Customer_id, Customer_name, Customer_email, Customer_address, Request_id, Status, Timestamp)

1NF

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

1.Customer_id, Customer_name, Customer_email, Customer_address, Request_id, Status, Timestamp

2NF

1.Customer_id, Customer_name, Customer_email, Customer_address

2.Request_id, Status, Timestamp

3NF

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

1.Customer_id, Customer_name, Customer_email, Customer_address

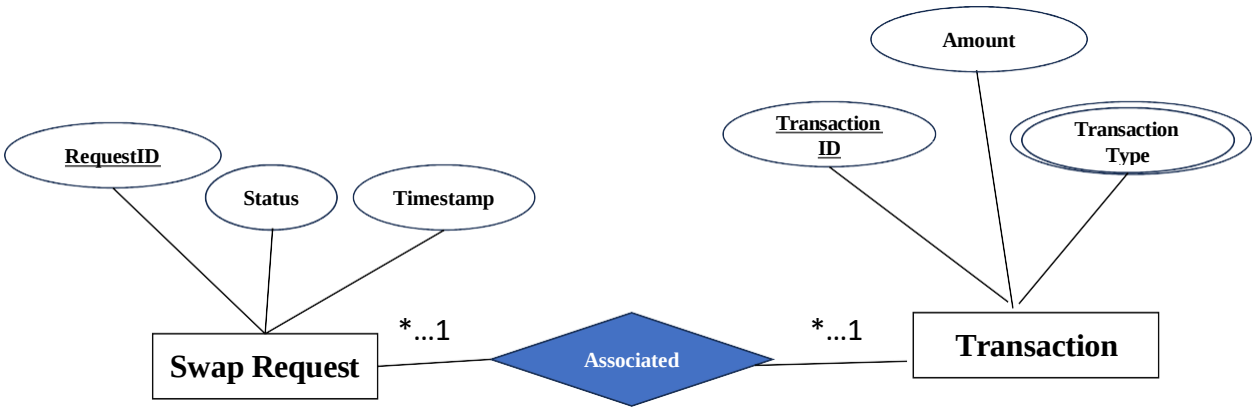
2.Request_id, Status, Timestamp

Table Creation

1.Customer_id, Customer_name, Customer_email, Customer_address

2.Request_id, Status, Timestamp, **C_id**

ASSOCIATED



UNF

Associated (Transaction_id, Transaction_type, Amount, Request_id, Status, Timestamp)

1NF

Transaction_type is a multi valued attribute.

1. Transaction_id, Transaction_type, Amount, Request_id, Status, Timestamp

2NF

1. Transaction_id, Transaction_type, Amount

2. Request_id, Status, Timestamp

3NF

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

1. Transaction_id, Transaction_type, Amount

2. Request_id, Status, Timestamp

Table Creation

1. Transaction_id, Transaction_type, Amount

2. Request_id, Status, Timestamp, **T_id**

Temporary Tables

1. Product_id, Product_name, Quantity, Descriptions
2. Category_id, Category_name, **P_id**
3. ~~Product_id, Product_name, Quantity, Descriptions~~
4. Request_id, Status, Timestamp, **P_id**
5. ~~Request_id, Status, Timestamp~~
6. Admin_id, Admin_name, Admin_password, Admin_email, **R_id**
7. Customer_id, Customer_name, Customer_email, Customer_address
8. ~~Request_id, Status, Timestamp~~, **C_id**
9. Transaction_id, Transaction_type, Amount
10. ~~Request_id, Status, Timestamp~~, **T_id**

Final tables

1. Product_id, Product_name, Quantity, Descriptions
2. Category_id, Category_name, **P_id**
3. Request_id, Status, Timestamp, **P_id**
4. Admin_id, Admin_name, Admin_password, Admin_email, **R_id**
5. Customer_id, Customer_name, Customer_email, Customer_address, **C_id**
6. Transaction_id, Transaction_type1, Transaction_type2, Transaction_type3, Amount, **T_id**

SCHEMA DIAGRAM

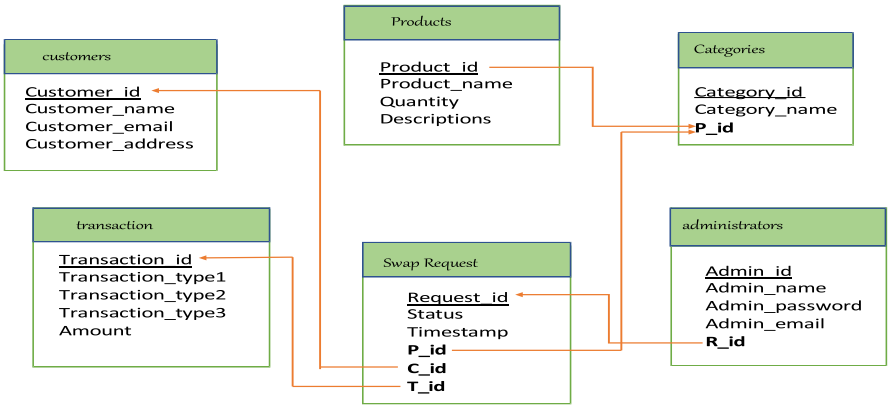
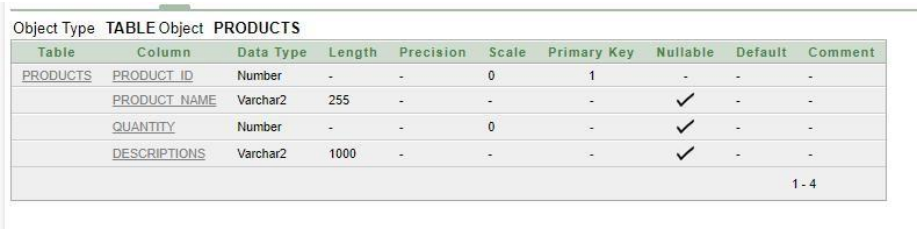


TABLE CREATION

StudentID1 : 23-50692-1 Name:Raiyan Bin Alam	StudentID3: : 23-50826-1 Name:Afifa Nujhat	
StudentID2: : 23-50695-1 Name:Monibur Rahman	StudentID4: 23-50014-1 Name: Sumaiya Tasnim	
	StudentID5: 23-50648-1 Name: AbdulAl-Abrar Shahriyar	
CO4: Creating DML, DDL using Oracle and connection with ODBC/JDBC for existing JAVA application		
PO-e-2: Use modern engineering and IT tools for prediction and modeling of complex computer science and engineering problem	Marks	



Object Type TABLE Object PRODUCTS

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PRODUCTS	PRODUCT_ID	Number	-	-	0	1	-	-	-
	PRODUCT_NAME	Varchar2	255	-	-	-	✓	-	-
	QUANTITY	Number	-	-	0	-	✓	-	-
	DESCRIPTIONS	Varchar2	1000	-	-	-	✓	-	-

1 - 4

Fig 1: Create and describe "Products" table

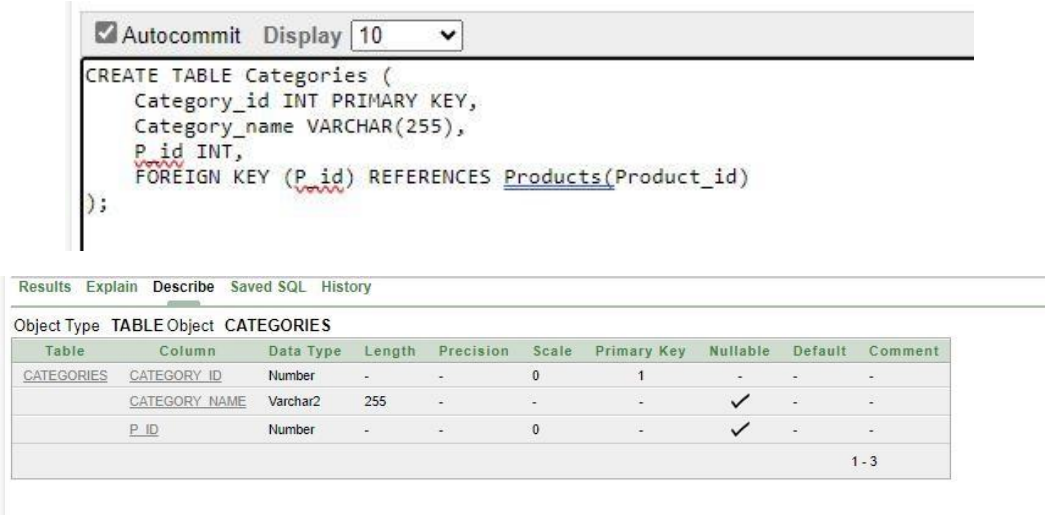


Fig2: Create and describe "Categories" table

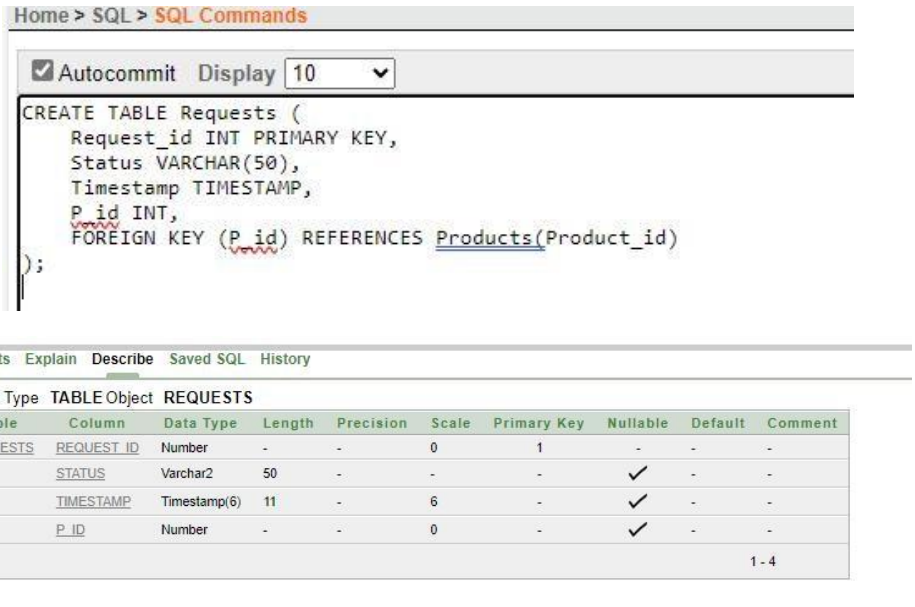


Fig3: Create and describe "Requests" table

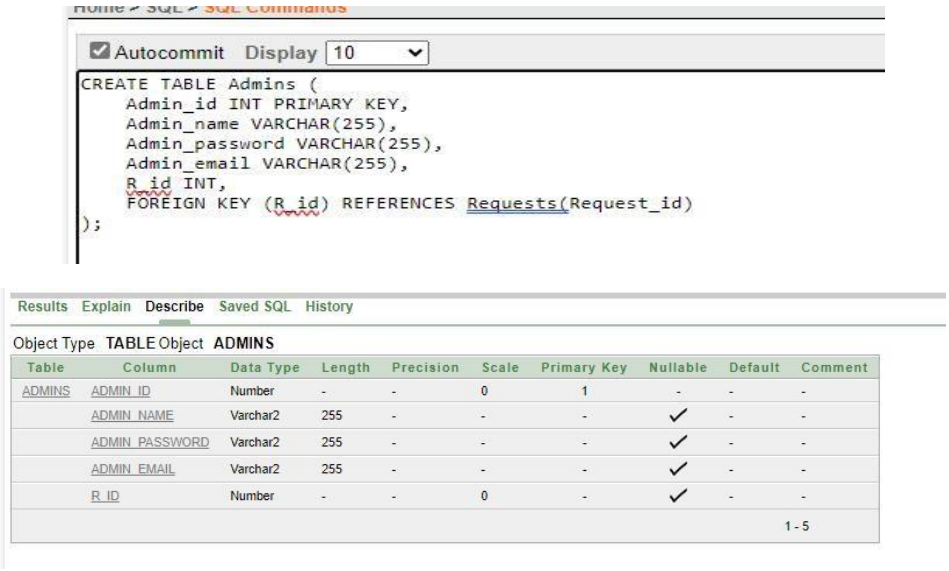


Fig4: Create and describe "Admins" table

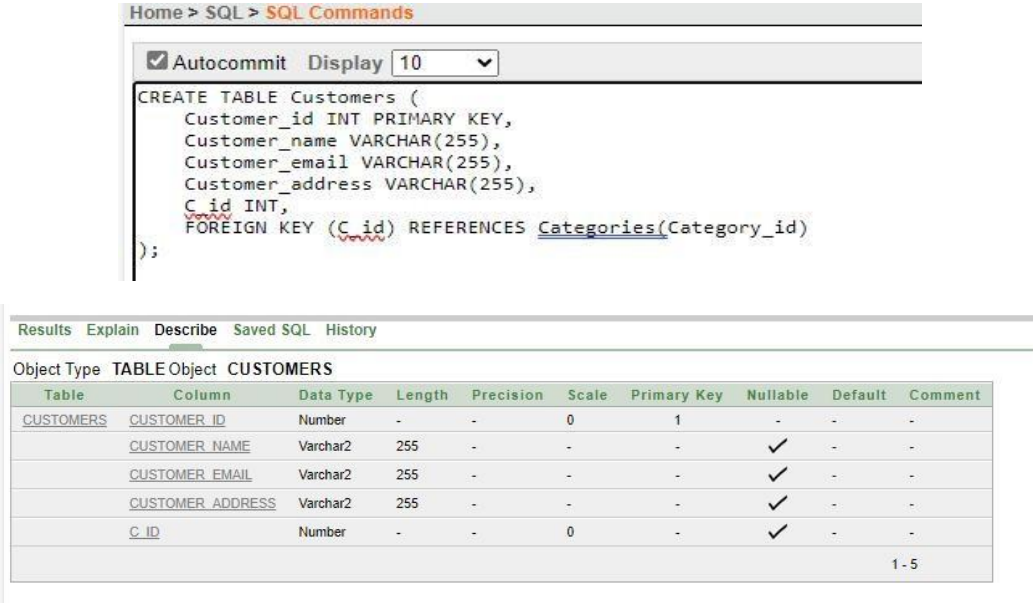


Fig5: Create and describe "Customers" table

Autocommit Display 10

```
CREATE TABLE Transactions (
    Transaction_id INT PRIMARY KEY,
    Transaction_type1 VARCHAR(50),
    Transaction_type2 VARCHAR(50),
    Transaction_type3 VARCHAR(50),
    Amount DECIMAL(10, 2),
    T_id INT,
    FOREIGN KEY (T_id) REFERENCES Transactions(Transaction_id)
);
```

Results Explain Describe Saved SQL History

Object Type TABLE Object TRANSACTIONS

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
TRANSACTIONS	TRANSACTION_ID	Number	-	-	0	1	-	-	-
	TRANSACTION_TYPE1	Varchar2	50	-	-	-	✓	-	-
	TRANSACTION_TYPE2	Varchar2	50	-	-	-	✓	-	-
	TRANSACTION_TYPE3	Varchar2	50	-	-	-	✓	-	-
	AMOUNT	Number	-	10	2	-	✓	-	-
	T_ID	Number	-	-	0	-	✓	-	-

1 - 6

Fig6: Create and describe " Transactions" table

DATA INSERTION

Insert data into the "Products" table:

```
Autocommit  Display 10
INSERT INTO Products (Product_id, Product_name, Quantity, Descriptions) VALUES (1, 'Book', 100, 'A novel about adventure');
INSERT INTO Products (Product_id, Product_name, Quantity, Descriptions) VALUES (2, 'Pen', 200, 'Blue ballpoint pen');
INSERT INTO Products (Product_id, Product_name, Quantity, Descriptions) VALUES (3, 'Laptop', 50, '15-inch laptop with SSD storage');
INSERT INTO Products (Product_id, Product_name, Quantity, Descriptions) VALUES (4, 'Headphones', 150, 'Wireless over-ear headphones');
INSERT INTO Products (Product_id, Product_name, Quantity, Descriptions) VALUES (5, 'Notebook', 300, 'A5 size ruled notebook');
```

PRODUCT_ID	PRODUCT_NAME	QUANTITY	DESCRIPTIONS
1	Book	100	A novel about adventure
2	Pen	200	Blue ballpoint pen
3	Laptop	50	15-inch laptop with SSD storage
4	Headphones	150	Wireless over-ear headphones
5	Notebook	300	A5 size ruled notebook

5 rows returned in 0.00 seconds CSV Export

Insert data into the "Categories" table:

```
INSERT INTO Categories (Category_id, Category_name, P_id) VALUES (1, 'Books', 1);
INSERT INTO Categories (Category_id, Category_name, P_id) VALUES (2, 'Stationery', 2);
INSERT INTO Categories (Category_id, Category_name, P_id) VALUES (3, 'Electronics', 3);
INSERT INTO Categories (Category_id, Category_name, P_id) VALUES (4, 'Electronics', 4);
INSERT INTO Categories (Category_id, Category_name, P_id) VALUES (5, 'Stationery', 5);
```

CATEGORY_ID	CATEGORY_NAME	P_ID
1	Books	1
2	Stationery	2
3	Electronics	3
4	Electronics	4
5	Stationery	5

5 rows returned in 0.07 seconds CSV Export

Insert data into the "Requests" table:

Autocommit Display 10

```
INSERT INTO Requests (Request_id, Status, Timestamp, P_id) VALUES (1, 'Pending', CURRENT_TIMESTAMP, 1);
INSERT INTO Requests (Request_id, Status, Timestamp, P_id) VALUES (2, 'Completed', CURRENT_TIMESTAMP, 2);
INSERT INTO Requests (Request_id, Status, Timestamp, P_id) VALUES (3, 'Processing', CURRENT_TIMESTAMP, 3);
INSERT INTO Requests (Request_id, Status, Timestamp, P_id) VALUES (4, 'Pending', CURRENT_TIMESTAMP, 4);
INSERT INTO Requests (Request_id, Status, Timestamp, P_id) VALUES (5, 'Completed', CURRENT_TIMESTAMP, 5);
```

Results Explain Describe Saved SQL History

REQUEST_ID	STATUS	TIMESTAMP	P_ID
1	Pending	12-MAY-24 11.27.42.590000 AM	1
2	Completed	12-MAY-24 11.27.48.392000 AM	2
3	Processing	12-MAY-24 11.27.54.501000 AM	3
4	Pending	12-MAY-24 11.27.59.972000 AM	4
5	Completed	12-MAY-24 11.28.05.542000 AM	5

5 rows returned in 0.00 seconds CSV Export

Insert data into the "Admins" table:

```
INSERT INTO Admins (Admin_id, Admin_name, Admin_password, Admin_email, R_id) VALUES (1, 'Admin1', 'password1', 'admin1@example.com', 1);
INSERT INTO Admins (Admin_id, Admin_name, Admin_password, Admin_email, R_id) VALUES (2, 'Admin2', 'password2', 'admin2@example.com', 2);
INSERT INTO Admins (Admin_id, Admin_name, Admin_password, Admin_email, R_id) VALUES (3, 'Admin3', 'password3', 'admin3@example.com', 3);
INSERT INTO Admins (Admin_id, Admin_name, Admin_password, Admin_email, R_id) VALUES (4, 'Admin4', 'password4', 'admin4@example.com', 4);
INSERT INTO Admins (Admin_id, Admin_name, Admin_password, Admin_email, R_id) VALUES (5, 'Admin5', 'password5', 'admin5@example.com', 5);
```

ADMIN_ID	ADMIN_NAME	ADMIN_PASSWORD	ADMIN_EMAIL	R_ID
1	Admin1	password1	admin1@example.com	1
2	Admin2	password2	admin2@example.com	2
3	Admin3	password3	admin3@example.com	3
4	Admin4	password4	admin4@example.com	4
5	Admin5	password5	admin5@example.com	5

5 rows returned in 0.05 seconds CSV Export

Insert data into the "Customers" table:

```
INSERT INTO Customers (Customer_id, Customer_name, Customer_email, Customer_address, C_id) VALUES (1, 'John Doe', 'john@example.com', '123 Main St', 1);
INSERT INTO Customers (Customer_id, Customer_name, Customer_email, Customer_address, C_id) VALUES (2, 'Jane Smith', 'jane@example.com', '456 Elm St', 2);
INSERT INTO Customers (Customer_id, Customer_name, Customer_email, Customer_address, C_id) VALUES (3, 'Bob Johnson', 'bob@example.com', '789 Oak St', 3);
INSERT INTO Customers (Customer_id, Customer_name, Customer_email, Customer_address, C_id) VALUES (4, 'Alice Brown', 'alice@example.com', '101 Maple St', 4);
INSERT INTO Customers (Customer_id, Customer_name, Customer_email, Customer_address, C_id) VALUES (5, 'Charlie Davis', 'charlie@example.com', '202 Pine St', 5);
```

CUSTOMER_ID	CUSTOMER_NAME	CUSTOMER_EMAIL	CUSTOMER_ADDRESS	C_ID
1	John Doe	john@example.com	123 Main St	1
2	Jane Smith	jane@example.com	456 Elm St	2
3	Bob Johnson	bob@example.com	789 Oak St	3
4	Alice Brown	alice@example.com	101 Maple St	4
5	Charlie Davis	charlie@example.com	202 Pine St	5

5 rows returned in 0.00 seconds CSV Export

Insert data into the "Transactions" table:

```
INSERT INTO Transactions (Transaction_id, Transaction_type1, Transaction_type2, Transaction_type3, Amount, T_id) VALUES (1, 'Sale', 'Online', 'Credit Card', 50.00, 1);
INSERT INTO Transactions (Transaction_id, Transaction_type1, Transaction_type2, Transaction_type3, Amount, T_id) VALUES (2, 'Purchase', 'In-store', 'Cash', 25.00, 2);
INSERT INTO Transactions (Transaction_id, Transaction_type1, Transaction_type2, Transaction_type3, Amount, T_id) VALUES (3, 'Refund', 'Online', 'PayPal', 10.00, 3);
INSERT INTO Transactions (Transaction_id, Transaction_type1, Transaction_type2, Transaction_type3, Amount, T_id) VALUES (4, 'Sale', 'Online', 'Credit Card', 100.00, 4);
INSERT INTO Transactions (Transaction_id, Transaction_type1, Transaction_type2, Transaction_type3, Amount, T_id) VALUES (5, 'Purchase', 'In-store', 'Cash', 75.00, 5);
```

TRANSACTION_ID	TRANSACTION_TYPE1	TRANSACTION_TYPE2	TRANSACTION_TYPE3	AMOUNT	T_ID
1	Sale	Online	Credit Card	50	1
2	Purchase	In-store	Cash	25	2
3	Refund	Online	PayPal	10	3
4	Sale	Online	Credit Card	100	4
5	Purchase	In-store	Cash	75	5

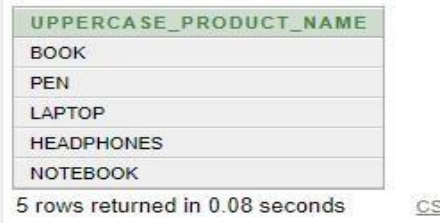
5 rows returned in 0.00 seconds CSV Export

QUERY WRITING:

Single-row functions:

Write a query to retrieve the product names in uppercase.

Ans: SELECT UPPER(Product_name) AS Uppercase_Product_Name FROM Products;

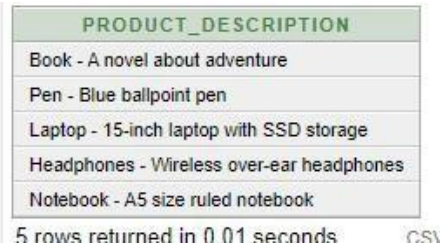


UPPERCASE_PRODUCT_NAME
BOOK
PEN
LAPTOP
HEADPHONES
NOTEBOOK

5 rows returned in 0.08 seconds CS

Write a query to concatenate the product name and description.

Ans: SELECT Product_name || ' - ' || Descriptions AS Product_Description FROM Products;



PRODUCT_DESCRIPTION
Book - A novel about adventure
Pen - Blue ballpoint pen
Laptop - 15-inch laptop with SSD storage
Headphones - Wireless over-ear headphones
Notebook - A5 size ruled notebook

5 rows returned in 0.01 seconds CSV

Group functions:

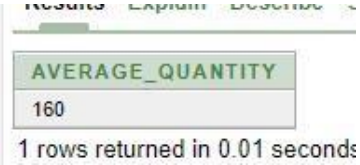
Write a query to count the total number of products.

Ans: SELECT COUNT(*) AS Total_Products FROM Products;

TOTAL_PRODUCTS
5

Write a query to find the average quantity of products.

Ans: SELECT AVG(Quantity) AS Average_Quantity FROM Products;



AVERAGE_QUANTITY
160

1 rows returned in 0.01 seconds

Subqueries:

Write a query to retrieve products with a quantity greater than the average quantity.

Ans: SELECT * FROM Products WHERE Quantity > (SELECT AVG(Quantity) FROM Products);

PRODUCT_ID	PRODUCT_NAME	QUANTITY	DESCRIPTIONS
2	Pen	200	Blue ballpoint pen
5	Notebook	300	A5 size ruled notebook

2 rows returned in 0.00 seconds CSV Export

Retrieve products with a quantity less than the average quantity.

Ans: SELECT * FROM Products WHERE Quantity < (SELECT AVG(Quantity) FROM Products);

PRODUCT_ID	PRODUCT_NAME	QUANTITY	DESCRIPTIONS
1	Book	100	A novel about adventure
3	Laptop	50	15-inch laptop with SSD storage
4	Headphones	150	Wireless over-ear headphones

3 rows returned in 0.00 seconds CSV Export

Joining:

Retrieve the names of products along with their corresponding category names using an equijoin.

Ans: SELECT Products.Product_name, Categories.Category_name FROM Products, Categories WHERE Products.Product_id = Categories.Category_id;

PRODUCT_NAME	CATEGORY_NAME
Book	Books
Pen	Stationery
Laptop	Electronics
Headphones	Electronics
Notebook	Stationery

5 rows returned in 0.03 seconds CSV Export

Retrieve the names of products along with their corresponding category names using an outer join.

Ans: SELECT Products.Product_name, Categories.Category_name FROM Products, Categories WHERE Products.Product_id(+) = Categories.Category_id;

PRODUCT_NAME	CATEGORY_NAME
Book	Books
Pen	Stationery
Laptop	Electronics
Headphones	Electronics
Notebook	Stationery

5 rows returned in 0.00 seconds CSV Export

View:

Create a view to display product names and their quantities.

Ans: CREATE VIEW Product_Quantity_View AS

SELECT Product_name, Quantity FROM Products;

SELECT * FROM Product_Quantity_View;

PRODUCT_NAME	QUANTITY
Book	100
Pen	200
Laptop	50
Headphones	150
Notebook	300

5 rows returned in 0.00 seconds CSV E

write a SQL statement to remove (drop) the view.

Ans: DROP VIEW Product_Quantity_View;

Results Explain Describe Saved SQL History

View dropped.

RELATIONAL ALGEBRA

Retrieve the names of all products.

Ans: π(Product_name)(Products)

Retrieve the names of all products along with their quantities.

Ans: π(Product_name, Quantity)(Products)

Retrieve the names of products with a quantity greater than 100.

Ans: σ(Quantity > 100)(Products)

Retrieve the names of products in category 'Electronics'.

Ans: π(Product_name)(σ(Category_name = 'Electronics')(Products))

Retrieve the names of products in category 'Electronics' with a quantity greater than 50.

Ans: π(Product_name)(σ(Category_name = 'Electronics' ∧ Quantity > 50)(Products))

CONCLUSION

Through this project, we explored various aspects of relational databases, including schema design, SQL queries, and relational algebra. We learned how to create database schemas, manipulate data using SQL, and perform operations using relational algebra. We also gained insights into database management systems (DBMS) and their practical applications in storing, retrieving, and managing data.

Future Work:

To improve this project in the future, we could consider the following:

1. Advanced SQL Topics: Explore more advanced SQL concepts such as stored procedures, triggers, and transactions.

2. Performance Optimization: Learn techniques for optimizing database performance, including indexing, query optimization, and database tuning.

3. Data Security: Enhance understanding of data security measures such as encryption, access control, and data masking to ensure data privacy and integrity.

4. Big Data and Analytics: Dive into big data technologies and analytics tools to analyze large volumes of data and derive actionable insights.

By focusing on these areas, we can further enhance our understanding of database management systems and prepare for more complex real-world scenarios in data management and analysis.