**ASSIGNMENT**

GROUP: 01

Student name: Đỗ minh chiến | HE186520

Student name: hoàng văn đại | HE180375

Student name: ngô quốc huy | HE186268

Student name: trần minh đức | HE186954

anh

INSTRUCTOR: BUI THI THU TRANG

DBI202 – DATABASE SYSTEM FOR CAMERA STORE

# TABLE OF CONTENTS

## INTRODUCE THE PROBLEM…………………………………………………………………………………………….

### Describe the problem……………………………………………………………………………………………………….

### Management objectives…………………………………………………………………………………………………..

## entity – relationship – er…………………………………………………………………………………………

### difinITION entity – attributE…………………………………………………………………………………………..

### set-up entity – relationship…………………………………………………………………………………………….

## data dictionary………………………………………………………………………………………………………….

### DATABASE AND TABLE……………………………………………………………………………………….………………….

## ENTITY RELATIONSHIP DIAGRAM (ERD)…………………………………………………………………………

### customer……………………………………………………………………………………………………………………………

### products…………………………………………………………………………………………………………………………….

### orders………………………………………………………………………………………………………………………………..

### orderdetail……………………………………………………………………………………………………………………….

### cAtegories…………………………………………………………………………………………………………………………

### warranty……………………………………………………………………………………………………………………………

### company…………………………………………………………………………………………………………………………….

### Supplier……………………………………………………………………………………………………………………………...

### employee ……………………………………………………………………………………………………………………………

## SQL COMMAND

### QUERY USING ORDER BY………………………………………………………………………………………………………

### QUERY USING JOIN……………………………………………………………………………………………………

### QUERY USING AGGREGATE FUNCTIONS…………………………………………………………………………………

### QUERY USING THE GROUP BY AND HAVING row……………………………………………………………

### QUERY THAT USES A SUB-QUERY AS A RELATION………………………………………………………………….

### QUERY THAT USES PARTIAL MATCHING IN THE WHERE CLAUSE…………………………………………….

### QUERY THAT USES A LELF-JOIN……………………………………………………………………………………………..

### STORE PROCEDURE……………………………………………………………………………………………………………….

### trigger……………………………………………………………………………………………………………………………….

# **introduce the problem**

## **describe the problem**

Currently, your store specializes in selling various cameras but lacks a database system to effectively manage sales, inventory, and customer information. After conducting a thorough analysis, the following information has been gathered:

* Store must provide information about each camera such as name, price, suppliers, type of camera
* The shop classifies the camera depending on lens, features, installation location, v.v....
* Each camera can belong to one or more categories.
* An order must have at least one camera and no limit.
* An order stores id of product, id of customer and the order date
* Order detail provide more information about order such as total amount, number of each camera
* The shop must store the customer’s information such as name, phone number, address for other orders. A customer can order more than one order.
* Employees will be responsible for each order they undertake. One employee can response for many orders.
* Employees have personal information such as name, phone number, email to contact in case of problem.
* Suppliers provide the products for shops to sell. Suppliers have names, phone number, email to contact if the product sources got trouble.
* Each supplier can sell and buy products from many companies.
* Companies are the product resources. Companies’ response for the quality of products and repair if camera is damaged.
* When a camera is damaged and needs to repair, customers can bring it to the shop and the shop will forward the camera back to company to repair.

**Request:**

* Daily, caculate number of orders, number of warrenty and the total money.
* Monthly, shop need cacule total amount collected.
* Manager can check information each shipment.
* Monthly, manager need to count and analyse amount of customers.
* Manager can check performance of each employees per month.

## **management objectives**

**Product Management:**

* Effectively manage detailed product and accessory information.
* Continuously monitor stock levels and automatically update inventory upon changes.

**Customer Management:**

* Efficiently manage customer information and purchase history.
* Establish and maintain positive customer relationships through effective after-sales care.

**Sales Management:**

* Effectively manage and track sales transactions.
* Ensure prompt and accurate order processing.

**Inventory Management:**

* Continuously track and manage inventory supplier information.
* Update stock levels promptly upon any changes.

**Specific Requirements:**

* **Daily:** Update stock levels and order payment status.
* **Monthly:** Summarize revenue and profit, conduct inventory checks and updates, evaluate sales performance, and generate sales reports.

**Product Management:**

* Monitor stock status, update product information when new products arrive, prices change.

**Customer Management:**

* Update customer information and purchase history, track and manage after-sales customer care.

**Sales Management:**

* Create and process new orders, check and update order payment and shipping status.

**Inventory Management:**

* Track and manage inventory inflow information, update stock levels upon changes.

**Database Design:**

* Create tables to store products, customer, order, inventory, and supplier information.
* Implement constraints and indexes to optimize query performance.

**Application Development:**

* Develop a user-friendly interface for managing information and transactions.
* Integrate functionalities such as product search, order management, inventory tracking, and customer management.

**Deployment and Testing:**

* Deploy the system to a production environment and conduct thorough testing to ensure system stability and adherence to requirements.

# **introduce the problem**

## **DIFINITION ENTITY – ATTRIBUTE**

Base on the problem description and management objectives, we can present several entities and attributes of the entity as follow:

Categories: **CategoryID**, CategoryName

Supplier: **SupplierID**, SupplierName, Address, Email, PhoneNumber

Products: **ProductID**, ProductName, Price, SupplierID

Customers: **CustomerID**, CustomerName, Email, Address, PhoneNumber

Orders: **OrderID**, CustomerID, ProductID, OrderDate, PurchaseDate

Company: **CompanyID**, CompanyName, Email

CompanyOfProduct: CompanyID, SupplierID

Warranty: **WarrantyID**, ProductID, WarrantyDate, TypeOfWarranty, CompanyID Employee: **EmployeeID**, OrderID, EmployeeName, PhoneNumber, Email

OrderDetail: **OrderDetailID**, TotalAmount, ProductUnitPrice, OrderID, Quantity

## **SETUP ENTITY – RELATIONSHIP**

A diagram of a company

Description automatically generated

# **Data dictionary**

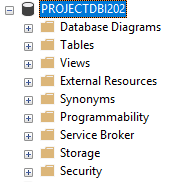
Just for example on some tables (other tables are similar, you must define all the tables in your database). Note: to run the query you must define the table 1 first then go to the side tables much

1. **DATABASE AND TABLE**

### cREATE DATABASE **PROJECTDBI202**

--create database

CREATE DATABASE PROJECTDBI202



1. CREATE TABLE **Categories**

| Column Name | Data Type | Default | Check | Key/ Index/ Constraint |
| --- | --- | --- | --- | --- |
| CategoryID | NVARCHAR(200) |  |  | Not null |
| CategoryName | NVARCHAR(200) |  |  | Not null |

***Code:***

CREATE TABLE Categories (

CategoryID NVARCHAR(20) NOT NULL PRIMARY KEY,

CategoryName NVARCHAR(200) not null

);

***Example:***

|  |  |
| --- | --- |
| CategoryID | CategoryName |
| C001 | DSLR Cameras |
| C002 | Mirrorless Cameras |
| C003 | Point and Shot Cameras |
| C004 | Action Cameras |
| C005 | 360-degree Cameras |

1. CREATE TABLE **Supplier**

| Column Name | Data Type | Default | Check | Key/ Index/ Constraint |
| --- | --- | --- | --- | --- |
| SupplierID | NVARCHAR(20) |  |  | Not null |
| SupplierID | NVARCHAR(50) |  |  | Not null |
| SupplierName | NVARCHAR(50) |  |  | Not null |
| Address | NVARCHAR(225) |  |  | Not null |
| Email | NVARCHAR(100) |  | '%\_@\_\_%.\_\_%' | Not null |
| PhoneNumber | CHAR(11) |  | 0 - 9 | Not null |

***Code:***

CREATE TABLE Supplier(

SupplierID NVARCHAR(20) NOT NULL PRIMARY KEY,

SupplierName NVARCHAR(50) NOT NULL,

Address NVARCHAR(225) NOT NULL,

Email NVARCHAR(100) CHECK (Email LIKE '%\_@\_\_%.\_\_%'),

PhoneNumber CHAR(11) NOT NULL CHECK (

PhoneNumber LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'

OR PhoneNumber LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]')

);

***Example:***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SupplierID | SupplierName | Address | Email | PhoneNumber |
| SUP001 | Công ty ABC | 123 Đường Lê Lợi, Quận 1, TP.HCM | contact@abc.vn | 01234567890 |
| SUP002 | Công ty XYZ | 456 Đường Nguyễn Huệ, Quận 1, TP.HCM' | support@xyz.vn | 09876543210 |
| SUP003 | Công ty DEF | 789 Đường Điện Biên Phủ, Quận 3, TP.HCM | info@def.vn' | 01122334455 |
| SUP004 | Nhà cung cấp GHI | 101 Đường Hai Bà Trưng, Quận 3, TP.HCM | service@ghi.vn | 02233445566 |
| SUP005 | Công ty JKL | 202 Đường Phạm Ngũ Lão, Quận 1, TP.HCM | sales@jkl.vn | 03344556677 |

1. CREATE TABLE **Products**

| Column Name | Data Type | Default | Check | Key/ Index/ Constraint |
| --- | --- | --- | --- | --- |
| ProductID | NVARCHAR(200) |  |  | Not null |
| ProductName | NVARCHAR(200) |  |  | Not null |
| Price | INT |  | Price>0 | Not null |
| CategoryID | NVARCHAR(200) |  |  | Not null |
| SupplierID | SupplierID |  |  | Not null |

***Code:***

CREATE TABLE Products (

ProductID CHAR(20) NOT NULL PRIMARY KEY,

ProductName NVARCHAR(50) NOT NULL,

Price INT CHECK (Price>0),

CategoryID NVARCHAR(20) NOT NULL,

SupplierID NVARCHAR(20) NOT NULL,

FOREIGN KEY (SupplierID) REFERENCES Supplier(SupplierID),

FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID)

);

***Example:***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ProductID | ProductName | Price | CategoryID | SupplierID |
| P001 | Nikon D3500 DSLR | 450 | C001 | SUP001 |
| P002 | Canon EOS Rebel SL3 | 600 | C001 | SUP002 |
| P003 | Sony Alpha A6100 | 750 | C002 | SUP003 |
| P004 | Fujifilm X-T200 | 700 | C002 | SUP004 |
| P005 | Olympus OM-D E-M10 Mark III | 550 | C002 | SUP005 |

1. **CREATE TABBLE** **Customers**

| Column Name | Data Type | Default | Check | Key/ Index/ Constraint |
| --- | --- | --- | --- | --- |
| CustomerID | NVARCHAR(20) |  |  | Not null |
| CustomerName | CHAR(20) |  |  | Not null |
| Email | NVARCHAR(100) |  | %\_@\_\_%.\_\_% | Not null |
| Address | NVARCHAR(100) |  | 0 - 9 | Not null |
| PhoneNumber | CHAR(11) |  | 0 - 9 | Not null |

***Code:***

CREATE TABLE Customers (

CustomerID CHAR(20) NOT NULL PRIMARY KEY,

CustomerName NVARCHAR(255) NOT NULL,

Email NVARCHAR(100) CHECK (Email LIKE '%\_@\_\_%.\_\_%'),

Address nvarchar(100) not null,

PhoneNumber CHAR(11) NOT NULL CHECK (

PhoneNumber LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'

OR PhoneNumber LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]')

);

***Example:***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CustomerID | CustomerName | Email | Address | PhoneNumber |
| CU001 | John Doe | johndoe@email.com | 100 Main St, Anytown | 01234567890 |
| CU002 | Jane Smith | janesmith@email.com | 101 First Ave, Othertown | 09876543211 |
| CU003 | Ella Morrison | ella.morrison@example.com | 203 Oak St, Lakeview | 02133445566 |
| CU004 | James Wilson | james.wilson@example.com | 204 Pine St, Hilltown | 02244556677 |
| CU005 | Charlotte Brown | charlotte.brown@example.com | 205 Maple St, Rivertown | 02355667788 |

1. **CREATE TABLE Orders**

| Column Name | Data Type | Default | Check | Key/ Index/ Constraint |
| --- | --- | --- | --- | --- |
| OrderID | CHAR(20) |  |  | Not null |
| CustomerID | CHAR(20) |  |  | Not null |
| ProductID | CHAR(20) |  | %\_@\_\_%.\_\_% | Not null |
| OrderDate | date |  | 0 - 9 | Not null |
| purchaseDate | date |  | 0 - 9 | Not null |

### ***Code:***

create table Orders(

OrderID char(20) not null primary key,

CustomerID char(20) not null,

ProductID char(20) not null,

OrderDate date not null,

purchaseDate date not null,

foreign key(CustomerID) REFERENCES dbo.Customers(CustomerID),

foreign key (ProductID) references dbo.Products(ProductID)

);

***Example:***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OrderID | CustomerID | ProductID | OrderDate | purchaseDate |
| O001 | CU001 | P001 | 2023-07-01 | 2023-07-02 |
| O001 | CU002 | P002 | 2024-07-01 | 2024-07-03 |
| O001 | CU003 | P003 | 2024-07-10 | 2024-07-11 |
| O001 | CU004 | P004 | 2024-07-10 | 2024-07-11 |
| O001 | CU005 | P005 | 2024-07-11 | 2024-07-12 |

1. **CREATE TABLE** **OrderDetail**

| Column Name | Data Type | Default | Check | Key/ Index/ Constraint |
| --- | --- | --- | --- | --- |
| OrderDetailID | NVARCHAR(20) |  |  | Not null |
| OrderID | NVARCHAR(20) |  |  | Not null |
| ProductID | NVARCHAR(20) |  |  | Not null |
| Quantity | INT |  | Quantity > 0 | Not null |
| TotalAmount | INT |  |  | Not null |
| ProductUnitPrice | INT |  | ProductUnitPrice>0 |  |

***Code:***

CREATE TABLE OrderDetail(

OrderDetailID CHAR(20) NOT NULL PRIMARY KEY,

OrderID CHAR(20) NOT NULL,

ProductID CHAR(20) NOT NULL,

Quantity INT NOT NULL CHECK (Quantity > 0),

TotalAmount INT NOT NULL,

ProductUnitPrice INT NOT NULL CHECK (ProductUnitPrice>0),

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),

FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);

***Example:***

| OrderDetailID | OrderID | ProductID | Quantity | TotalAmount | ProductUnitPrice |
| --- | --- | --- | --- | --- | --- |
| OD001 | O001 | P001 | 2 | 900 | 450 |
| OD002 | O002 | P002 | 1 | 600 | 600 |
| OD003 | O003 | P003 | 3 | 2250 | 750 |
| OD004 | O004 | P004 | 2 | 1400 | 700 |
| OD005 | O005 | P005 | 1 | 550 | 550 |

1. Create table **Company**

| Column Name | Data Type | Default | Check | Key/ Index/ Constraint |
| --- | --- | --- | --- | --- |
| CompanyID | NVARCHAR(20) |  |  | Not null |
| CompanyName | NVARCHAR(225) |  |  | Not null |
| Email | NVARCHAR(100) |  | Email LIKE '%\_@\_\_%.\_\_%') | Not null |

***Code:***

create table Company(

CompanyID nvarchar(20) not null primary key,

CompanyName nvarchar(225) not null,

Email NVARCHAR(100) NOT NULL CHECK (Email LIKE '%\_@\_\_%.\_\_%')

);

***Example:***

| CompanyID | CompanyName | Email |
| --- | --- | --- |
| CO001 | Ultimate Camera Inc. | support@ultimatecam.com |
| CO002 | Global Optics Ltd. | contact@globaloptics.com |
| CO003 | Visionary Imaging Solutions | service@visionaryimaging.com |
| CO004 | SkyHigh Drones Inc. | contact@skyhighdrones.com |
| CO005 | Rapid Development Co. | support@rapiddev.com |

1. Create table **CompanyOfProduct**

| Column Name | Data Type | Default | Check | Key/ Index/ Constraint |
| --- | --- | --- | --- | --- |
| CompanyID | NVARCHAR(20) |  |  | Not null |
| SupplierID | NVARCHAR(20) |  |  | Not null |

***Code:***

create table CompanyOfProduct(

CompanyID nvarchar(20) not null,

SupplierID nvarchar(20) not null,

foreign key (CompanyID) references Company(CompanyID),

foreign key (SupplierID) references Supplier(SupplierID)

);

***Example:***

| CompanyID | SupplierID |
| --- | --- |
| CO001 | SUP001 |
| CO001 | SUP002 |
| CO003 | SUP003 |
| CO004 | SUP004 |
| CO005 | SUP005 |

1. CREATE TABLE **Warranty**

| Column Name | Data Type | Default | Check | Key/ Index/ Constraint |
| --- | --- | --- | --- | --- |
| WarrantyID | CHAR(20) |  |  | Not null |
| ProductID | CHAR(20) |  |  | Not null |
| WarrantyDate | date |  |  | Not null |
| TypeOfWarranty | NVARCHAR(255) |  |  | Not null |
| CompanyID | NVARCHAR(20) |  |  | Not null |

***Code:***

CREATE TABLE Warranty (

WarrantyID CHAR(20) NOT NULL PRIMARY KEY,

ProductID char(20) not null,

WarrantyDate DATE NOT NULL,

TypeOfWarranty NVARCHAR(255) NOT NULL,

CompanyID nvarchar(20) not null,

foreign key (ProductID) references Products(ProductID),

foreign key (CompanyID) references Company(CompanyID)

);

***Example:***

| WarrantyID | ProductID Type | WarrantyDate | TypeOfWarranty | CompanyID |
| --- | --- | --- | --- | --- |
| W003 | P003 | 2023-07-07 | Extended 5-Year Warranty | CO003 |
| W004 | P004 | 2023-07-08 | Standard 2-Year Warranty | CO004 |
| W005 | P005 | 2023-07-09 | Extended 5-Year Warranty | CO005 |
| W008 | P008 | 2023-07-12 | Standard 2-Year Warranty | CO008 |
| W009 | P009 | 2023-07-13 | Extended 5-Year Warranty | CO009 |

1. CREATE TABLE Employee

| Column Name | Data Type | Default | Check | Key/ Index/ Constraint |
| --- | --- | --- | --- | --- |
| EmployeeID | CHAR(20) |  |  | Not null |
| OrderID | CHAR(20) |  |  | Not null |
| EmployeeName | NVARCHAR(225) |  |  | Not null |
| PhoneNumber | CHAR(11) |  | 0 - 9 | Not null |
| Email | NVARCHAR(100) |  | %\_@\_\_%.\_\_% | Not null |

***Code:***

CREATE TABLE Employee(

EmployeeID CHAR(20) NOT NULL PRIMARY KEY,

OrderID char(20) not null,

EmployeeName NVARCHAR(225) NOT NULL,

PhoneNumber CHAR(11) NOT NULL,

Email NVARCHAR(100),

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)

);

***Example:***

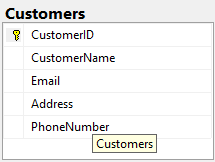
| EmployeeID | OrderID | EmployeeName | PhoneNumber | Email |
| --- | --- | --- | --- | --- |
| E001 | O001 | Alice Johnson | 01234567891 | alice.j@camstore.com |
| E002 | O002 | Bob Smith | 09876543210 | bob.s@camstore.com |
| E019 | O001 | Ava Taylor | 01234567890 | ava.taylor@example.com |
| E020 | O002 | Liam Brown | 01234567891 | liam.brown@example.com |
| E021 | O003 | Olivia Smith | 01234567892 | olivia.smith@example.com |

# IV. entity relationship diagram (erd)

1. CUSTOMERS:

A diagram of a customer

Description automatically generated



This is the Customer entity of the ERD for the Camera store program.

The Customers entity has 5 attributes.

The attribute CustomerID is also the primary key of this entity.

Each Customer will have a name and contact details.

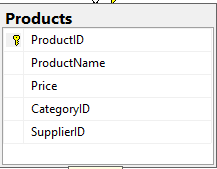
The name is CustomerName.

The contact details of the Customer include Email ,Phone, and Address.

1. PRODUCT:

A diagram of a product

Description automatically generated



This is the Product entity of the ERD for the Camera store program.

The Product entity has 3 attributes.

The attribute ProductID is the primary key of this entity.

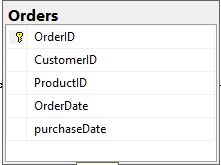
Each Product will have a ProductName and Price.

Additionally, it can be linked to the Supplier entity through SupplierID.

1. ORDERS:

A diagram of order and order

Description automatically generated

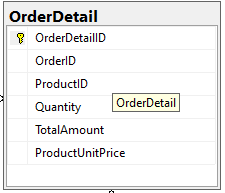


This is the Orders entity of the ERD for the Camera store program. The Orders entity has 2 attributes, which are OrderID and PurchaseDate, where OrderID is the primary key of this entity.

1. ORDER DETAIL:

A diagram of a company

Description automatically generated

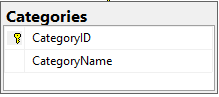


This is the OrderDetail entity of the ERD for the Camera store program. The OrderDetail entity has 4 attributes, which are OrderDetailID and TotalAmount, where OrderDetailID is the primary key of this entity. TotalAmount is used to calculate the total cost of an order.

1. CATEGORIES:

A diagram of categories

Description automatically generated

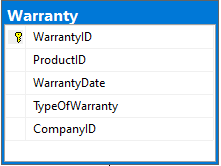


This is the Categories entity of the ERD for the Camera store program. The Categories entity has 2 attributes, which are CategoryID and CategoryName, where CategoryID is the primary key of this entity. Each Category includes the name and ID of each type.

1. WARRANTY:

A diagram of a warranty

Description automatically generated

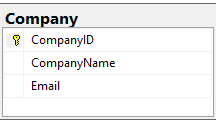


This is the Warranty entity of the ERD for the Camera store program. The Warranty entity has 3 attributes, which are WarrantyID, WarrantyDate, and TypeOfWarranty, where WarrantyID is the primary key of this entity. The warranty of each Product is indirectly linked to the Supplier through Product.

1. COMPANY:

A diagram of company

Description automatically generated

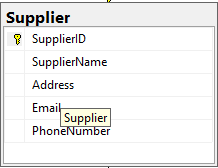


This is the Company entity of the ERD for the Camera store program. The Company entity has 3 attributes, which are CompanyID, CompanyName, and Email, where CompanyID is the primary key of this entity. Each Company will have a specific name and contact information for Supplier and Customer to understand clearly.

1. SUPPLIER:

A diagram of a company

Description automatically generated

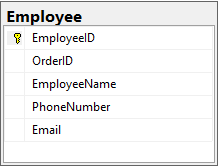


This is the Supplier entity of the ERD for the Camera store program. The Supplier entity has 5 attributes, which are SupplierID, SupplierName, Address, PhoneNumber, and Email, where SupplierID is the primary key of this entity. Each Supplier will have a name, ID, and specific contact details (Email, Address, PhoneNumber).

1. EMPLOYEE:

A diagram of a company

Description automatically generated



This is the Employee entity of the ERD for the Camera store program. The Employee entity has 4 attributes, which are EmployeeID, EmployeeName, PhoneNumber, and Email, where EmployeeID is the primary key of this entity. The name and contact details (PhoneNumber, Email) are clearly defined in terms of data. It ensures the connectivity of tables through common attributes.

# V. sql command

I using Microsoft SQL Server 2016, this server build intelligent, mission-critical applications using a scalable, hybrid database platform that has everything built in—from in-memory performance and advanced security to in-database analytics.

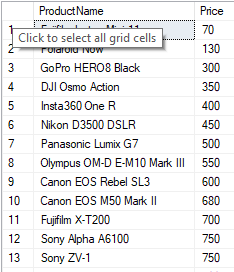
## query using order by

***Code:***

SELECT ProductName, Price FROM Products

ORDER BY Price ASC;

***Result:***

This query retrieves the ProductName and Price from the Products table and orders the results by the Price column in ascending order. This means that the products with the lowest prices will be listed first..

## query using inner join

***Code:***

SELECT Products.ProductName, Products.Price, Supplier.SupplierName

FROM Products

JOIN Supplier ON Products.SupplierID = Supplier.SupplierID;

***Result:***

This query joins the Products and Supplier tables on the SupplierID column. It then selects the ProductName and Price from the Products table and the SupplierName from the Supplier table. This will return a list of products, along with the name of the supplier that supplies each product.

## query using aggregate functions

***Code:***

SELECT AVG(Price) AS AveragePrice FROM Products;

***Result:***

******

This query selects the average price of all products from the Products table. The AVG() function is used to calculate the average value of the Price column.

## query using the group by and having clauses

***Code:***

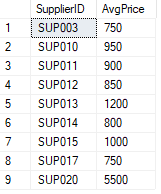
SELECT SupplierID, AVG(Price) AS AvgPrice

FROM Products

GROUP BY SupplierID

HAVING AVG(Price) > 700;

***Result:***

******

This query groups the products by SupplierID and calculates the average price for each supplier using the AVG() function. It then filters the results to only include suppliers with an average price greater than 700.

## query that uses a sub-query as a relation

***Code:***

SELECT \* FROM Products

WHERE Price = (SELECT MAX(Price) FROM Products);

***Result:***

******

This query selects all information from the Products table where the Price is equal to the maximum price of all products. The MAX() function is used to find the maximum value of the Price column.

## query that uses partial matching in the where clause

***Code:***

SELECT \* FROM Products

WHERE Price BETWEEN 500 AND 1000;

***Result:***

****

This query selects all information from the Products table where the Price is between 500 and 1000. The BETWEEN keyword is used to specify a range of values

## query that uses a lEFT – join

***Code:***

SELECT Products.ProductName, Orders.OrderID

FROM Products

LEFT JOIN Orders ON Products.ProductID = Orders.ProductID;

***Result:***

******

This query performs a left join between the Products and Orders tables on the ProductID column. This means that all products will be returned, even if they do not have any corresponding orders. The OrderID from the Orders table is included in the results for each product.

## query that uses a trigger

***Code:***

CREATE TRIGGER CheckOrderDates

ON Orders

AFTER INSERT

AS

BEGIN

IF EXISTS (

SELECT 1

FROM inserted

WHERE OrderDate <> PurchaseDate

)

BEGIN

UPDATE OrderDetail

SET TotalAmount = 0

FROM OrderDetail

JOIN inserted ON OrderDetail.OrderID = inserted.OrderID

END

END;

***----TESTING THE TRIGGER----***

INSERT INTO Orders (OrderID, CustomerID, ProductID, OrderDate, PurchaseDate)

VALUES ('O021', 'CU001', 'P001', '2024-07-20', '2024-07-21');

--SELECT Inserted OrderID

SELECT \* FROM OrderDetail WHERE OrderID = 'O021';

INSERT INTO OrderDetail (OrderDetailID, OrderID, ProductID, Quantity, TotalAmount, ProductUnitPrice)

VALUES ('OD100', 'O021', 'P001', 2, 900, 450);

***Result:***

****

This query creates a trigger called CheckOrderDates that fires after an insert is made to the Orders table. The trigger checks if the OrderDate and PurchaseDate columns are equal. If they are not equal, the trigger updates the TotalAmount column in the OrderDetail table to 0 for the corresponding order.

## query that uses a procedure

***Code:***

***--CREATE PROCEDURE--***

CREATE PROCEDURE GetProductDetails

@ProductID CHAR(20)

AS

BEGIN

SELECT \* FROM Products

WHERE ProductID = @ProductID;

END;

***--TESTING PROCEDURE--***

EXEC GetProductDetails @ProductID = 'P001';

***Result:***



This query creates a stored procedure called GetProductDetails that takes a ProductID parameter. The procedure selects all information from the Products table where the ProductID matches the parameter value

THE END