Elias Stillahn

Merchandise Database Project

Database Info

The name of my database is MerchandiseEVS which is based on the topic of Merchandise as you can guess from the name already. There are three tables in my database: Product, Orders, and Order_Details. The Product table stores the Product_ID and Product_Name columns. The Orders table stores the Order_ID and OrderDate columns. Finally, the Order_Details tables stores the Order_ID, Product_ID, and Price columns.

The Primary keys are as follows.

Product table: Product_ID.

Orders table: OrderID.

Order_Details: Order_ID and Product_ID.

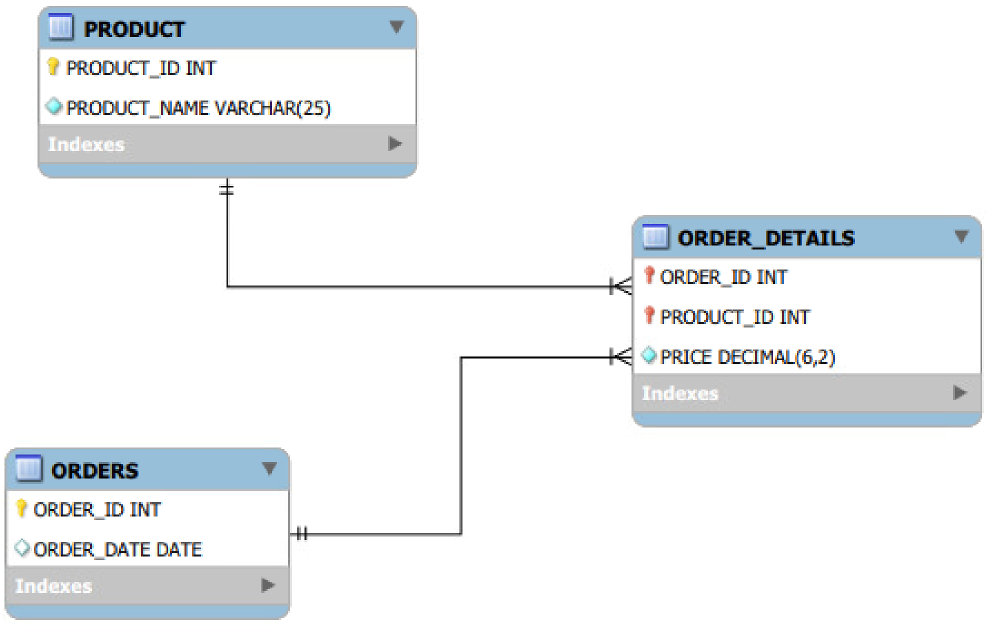
The Foreign key are as follows.

Product table: None

Orders table: None

Order_Details: Order_ID Referencing Orders Table and Product_ID Referencing Product table.

The Order_Details table has the many-to-one relationships to the other table since it draws both other tables.



Database Project Data Dictionary

Table Name	column Name	DataType(length)	Required (Y/N)	Primary Key (Y/N)	Foreign Key (Y/N)	References	Attribute
PRODUCT	PRODUCT_ID	INT	Y	Y	NO		AUTO_INCREMENT
PRODUCT	PRODUCT_NAME	VARCHAR(25)	Y	No	No		
ORDERS	ORDER_ID	INT	Y	Y	NO		AUTO_INCREMENT
ORDERS	ORDER_DATE	Date	N	N	n		DEFAULT NULL
ORDER_DETAILS	ORDER_ID	INT	Y	Y	Y	ORDERS.ORDER_ID	
ORDER_DETAILS	PRODUCT_ID	INT	Y	Y	Y	PRODUCTS.PRODUCT_ID	
ORDER_DETAILS	PRICE	DECIMAL(6,2)	Y	N	N		

MySQL DDL Script

```
USE sys;

DROP DATABASE IF EXISTS merchandise_es;

CREATE DATABASE merchandise_es;

USE merchandise_es;

CREATE TABLE product (

product_id INT NOT NULL auto_increment,

product_name VARCHAR(25) NOT NULL,

CONSTRAINT product_id_pk PRIMARY KEY (product_id)

);

INSERT INTO product

(product_id, product_name)

VALUES

(1, "football"), (2, "soccer"), (3, "baseball"), (4, "bat"), (5,
"laptop"), (6, "mouse"), (7, "headphones"), (8, "phone"), (9, "cards"),
(10, "charger"), (11, "monitor"), (12, "book"), (13, "textbook"), (14,
"umbrella"), (15, "pen");

CREATE TABLE orders (

order_id INT NOT NULL auto_increment,

order_date DATE DEFAULT NULL,

CONSTRAINT order_id_pk PRIMARY KEY (order_id)
);


INSERT INTO orders

(order_id, order_date)

VALUES

(1, "2009-04-21"), (2, "2013-11-03"), (3, "2014-04-29"), (4, "2014-07-
02"), (5, "2014-08-16"), (6, "2015-04-12"), (7, "2015-05-07"), (8,
"2016-01-22"), (9, "2017-04-04"), (10, "2018-10-31"), (11, "2019-12-
25"), (12, "2020-02-20"), (13, "2020-06-01"), (14, "2022-09-20"), (15,
"2024-01-27");

CREATE TABLE order_details (

order_id INT NOT NULL,

product_id INT NOT NULL,

price DECIMAL(6,2) NOT NULL,

CONSTRAINT order_id_fk foreign key (order_id) REFERENCES
orders(order_id),

CONSTRAINT product_id_fk foreign key (product_id) REFERENCES
product(product_id),

CONSTRAINT composite_pk PRIMARY KEY (product_id, order_id)

);

INSERT INTO order_details

(order_id, product_id, price)

VALUES

(1, 1, 5.00), (2, 1, 6.00), (3, 1, 3.00), (4, 1, 20.00), (5, 5, 999.00),

(6, 6, 60.00), (7, 7, 60.00), (8, 8, 799.99), (9, 9, 18.00), (10, 10,
5.99), (11, 11, 200.00), (12, 12, 35.00), (13, 13, 112.00), (14, 14,
12.99), (15, 15, 1.99);
```

SQL Server DDL Script

```
USE master;

GO

DROP DATABASE IF EXISTS MerchandiseEVS;

GO

CREATE DATABASE MerchandiseEVS;

GO

USE MerchandiseEVS;

GO

CREATE TABLE Product (

ProductID INT NOT NULL IDENTITY,

ProductName VARCHAR(25) NOT NULL,

CONSTRAINT PK_Product_ProductID PRIMARY KEY (ProductID)

);


CREATE TABLE Orders (

OrderID INT NOT NULL IDENTITY,

OrderDate DATE DEFAULT NULL,

CONSTRAINT PK_Orders_OrderID PRIMARY KEY (OrderID)

);


CREATE TABLE OrderDetails (

OrderID INT NOT NULL,
ProductID INT NOT NULL,

Price DECIMAL(6,2) NOT NULL,

CONSTRAINT FK_Order_OrderID foreign key (OrderID) REFERENCES
Orders(OrderID),

CONSTRAINT FK_Product_ProductID foreign key (ProductID) REFERENCES
Product(ProductID),

CONSTRAINT PK_OrderDetails_ProductID_OrderID PRIMARY KEY (ProductID,
OrderID)

);

GO



INSERT INTO Product (ProductName)

VALUES

('football'), ('soccer'), ('baseball'), ('bat'), ('laptop'), ('mouse'),
('headphones'), ('phone'), ('cards'), ('charger'), ('monitor'),
('book'), ('textbook'), ('umbrella'), ('pen');

INSERT INTO Orders (OrderDate)

VALUES

('2009-04-21'), ('2013-11-03'), ('2014-04-29'), ('2014-07-02'), ('2014-
08-16'), ('2015-04-12'), ('2015-05-07'), ('2016-01-22'), ('2017-04-04'),
('2018-10-31'), ('2019-12-25'), ('2020-02-20'), ('2020-06-01'), ('2022-
09-20'), ('2024-01-27');

INSERT INTO OrderDetails

(OrderID, ProductID, Price)

VALUES

(1, 1, 6.00), (2, 1, 7.00), (3, 1, 5.99), (4, 1, 6.00), (5, 5, 999.00),
(6, 6, 60.00), (7, 7, 60.00), (8, 8, 799.99), (9, 9, 18.00), (10, 10,
5.99), (11, 11, 200.00), (12, 12, 35.00), (13, 13, 112.00), (14, 14,
12.99), (15, 15, 1.99);
```

SELECT QUERIES

```
/* OUTER join */

/*Q1 What is the total revenue?*/

SELECT SUM(d.Price) AS TotalRevenue

FROM Product p

LEFT OUTER JOIN OrderDetails d ON p.productID = d.productID;

/*Results
```

TotalRevenue
2329.95

```
*/

/* TOP values INNER join*/

/*Q2 What are the top five most expensive items you can buy?*/

SELECT TOP 5

p.ProductName, d.Price

FROM Product p

INNER JOIN OrderDetails d ON p.productID = d.productID

ORDER BY d.Price DESC;

/*Results
```

ProductName	Price
laptop	999.00
phone	799.99
monitor	200.00
textbook	112.00
mouse	60.00

```
*/
```

/* Aggregate Function with GROUP BY */

/*Q3 What is the most popular product?*/

```
SELECT TOP 1 p.ProductName, COUNT(d.OrderID) AS totalOrders

FROM Product p

JOIN OrderDetails d ON p.productID = d.productID

GROUP BY p.productName

ORDER BY totalOrders DESC;

/*Result
```

ProductName	totalOrders
football	4

```
*/
```

/* Aggregate Function with GROUP BY and HAVING */

/*Q4 What is the avg cost of a football based of the prices from previous orders?*/

```
SELECT p.ProductName, AVG(d.Price) AS AveragePrice

FROM Product p

JOIN OrderDetails d ON p.productID = d.productID

GROUP BY p.productName

HAVING p.productName = 'football';

ORDER BY totalOrders DESC;

/*Result
```

ProductName	AveragePrice
football	6.247500

*/

/* String Expression */

/* Q5 What are all of the products and their average prices? (will not be listed if item was not sold before)*/

```
SELECT p.ProductID, CONCAT('The avg price of ', p.ProductName, ' is $', AVG(d.Price))

FROM Product p

JOIN OrderDetails d ON p.productID = d.productID

GROUP BY p.ProductID, p.ProductName;
```

/*Result

ProductID	avgPrice
1	The avg price of football is $6.247500
5	The avg price of laptop is $999.000000
6	The avg price of mouse is $60.000000
7	The avg price of headphones is $60.000000
8	The avg price of phone is $799.990000
9	The avg price of cards is $18.000000
10	The avg price of charger is $5.990000
11	The avg price of monitor is $200.000000
12	The avg price of book is $35.000000
13	The avg price of textbook is $112.000000
14	The avg price of umbrella is $12.990000
15	The avg price of pen is $1.990000

*/

/* DATEDIFF Function */

/*Q6 How long ago was the last purchase made?*/

```
SELECT DATEDIFF(day, (SELECT MAX(OrderDate) FROM Orders), GetDate()) AS
DaysSinceLastPurchase;
```

```
/*Result
```

DaysSinceLastPurchase
454

```
*/

/* Subquery */

/*Q7 What are the products that have never been sold?*/

SELECT p.ProductName

FROM Product p

WHERE p.ProductID NOT IN(

      SELECT d.ProductID

      FROM OrderDetails d);

/*Results
```

ProductName
soccer
baseball
bat

```
*/
```

VIEWS

```
USE MerchandiseEVS;

GO
```

/*Cast or Convert USING INNER JOIN*/

/*A view to display a recipt. This will show the orderID, Date, and Total using an inner join to combine tow tables. The Date will be changed from a date type to varchar*/

```
DROP VIEW IF EXISTS VW_Order_Receipt;

GO


CREATE VIEW VW_Order_Receipt AS

SELECT d.OrderID, d.ProductID, d.Price, CAST(o.OrderDate AS
varchar) AS varcharDate

FROM OrderDetails d

JOIN Orders o ON o.OrderID = d.OrderID;

GO


SELECT *

FROM VW_Order_Receipt;
```

OrderID	ProductID	Price	varcharDate
1	1	6.00	2009-04-21
2	1	7.00	2013-11-03
3	1	5.99	2014-04-29
4	1	6.00	2014-07-02
5	5	999.00	2014-08-16
6	6	60.00	2015-04-12
7	7	60.00	2015-05-07
8	8	799.99	2016-01-22
9	9	18.00	2017-04-04
10	10	5.99	2018-10-31
11	11	200.00	2019-12-25
12	12	35.00	2020-02-20

13	13	112.00	2020-06-01
14	14	12.99	2022-09-20
15	15	1.99	2024-01-27

/*Two String Functions*/

/*This view will display the prices that items sold at from different orders since not sales ended with the same price for an item.*/

```
DROP VIEW IF EXISTS VW_Product_Details;

GO


CREATE VIEW VW_Product_Details AS

SELECT CONCAT(UPPER(p.ProductName),  ' ', d.OrderID, ' was $',
d.Price) AS 'Product Sales Info'

FROM Product p

JOIN OrderDetails d ON p.ProductID = d.ProductID;

GO

SELECT *

FROM VW_Product_Details;
```

Product Sales Info
FOOTBALL 1 was $6.00
FOOTBALL 2 was $7.00
FOOTBALL 3 was $5.99
FOOTBALL 4 was $6.00
LAPTOP 5 was $999.00
MOUSE 6 was $60.00
HEADPHONES 7 was $60.00
PHONE 8 was $799.99
CARDS 9 was $18.00
CHARGER 10 was $5.99
MONITOR 11 was $200.00
BOOK 12 was $35.00
TEXTBOOK 13 was $112.00
UMBRELLA 14 was $12.99
PEN 15 was $1.99

/*Substring function SUBQUERY*/

/*This view will display every order where a football was sold that was either greater than or equal to 6 dollars.*/

```
DROP VIEW IF EXISTS VW_Order_Date_Info;

GO


CREATE VIEW VW_Order_Date_Info AS

SELECT o.OrderID, p.ProductName

FROM  OrderDetails o

JOIN Product p ON p.ProductID = o.ProductID

WHERE Price IN (

SELECT Price

FROM Orders

WHERE Price >= 6)

AND SUBSTRING(p.ProductName, 1, 8) = 'Football'

;

GO

SELECT *

FROM VW_Order_Date_Info;
```

OrderID	ProductName
1	football
2	football
4	football

/*Round function with calc value or aggregate function*/

/* This view displays the average price for each product being sold and if the item was not sold it will not be listed in this chart.*/

```
DROP VIEW IF EXISTS VW_Orders_AveragePrice;

GO


CREATE VIEW VW_Orders_AveragePrice AS

SELECT p.ProductName, CONCAT('$', ROUND(AVG(o.Price), 2)) AS
AveragePrice

FROM OrderDetails o

RIGHT OUTER JOIN Product p ON p.ProductID = o.ProductID

WHERE o.Price IS NOT NULL

GROUP BY p.ProductName;

GO

SELECT *

FROM VW_Orders_AveragePrice;
```

ProductName	AveragePrice
book	$35.000000
cards	$18.000000
charger	$5.990000
football	$6.250000
headphones	$60.000000
laptop	$999.000000
monitor	$200.000000
mouse	$60.000000
pen	$1.990000
phone	$799.990000
textbook	$112.000000
umbrella	$12.990000

/*DATEDIFF Function*/

/*The point of this view is to show the date of the last order received and display how many days ago it was.*/

```
DROP VIEW IF EXISTS VW_Oldest_Order;
GO

CREATE VIEW VW_Oldest_Order AS
SELECT TOP 1
OrderDate AS 'Last Date Sold',
DATEDIFF(day, OrderDate, GetDate()) AS 'Days ago'
FROM Orders
ORDER BY OrderDate DESC;
GO

SELECT *
FROM VW_Oldest_Order;
```

Last Date Sold	Days ago
2024-01-27	464

STORED PROCEDURES

```sql
USE MerchandiseEVS;

GO


/*Stored Procedure 1 - CASE Function*/

/*This stored procedure gets the input from the user asking for the OrderID,

After inputing the OrderID it determines if your order was inexpensive or expensive based

off of the CASE function.*/

DROP PROC IF EXISTS sp_OrderDetails_Price;

GO


CREATE PROC sp_OrderDetails_Price

@OrderID int

AS

BEGIN

SELECT Price,

     CASE

     WHEN Price < 20 THEN 'Your order was inexpensive.'

     Else 'Your order was expensive.'

     END AS PriceStatus

FROM OrderDetails

WHERE OrderID = @OrderID;

END;

GO


EXEC sp_OrderDetails_Price 1;

GO
```

Price	PriceStatus
6.00	Your order was inexpensive.

```
/*Stored Procedure - 2 - Input and Output Parameters*/

/*This stored procedure grabs the input of Product ID 10 and then grabs the price and outputs it

to a variable called ProductPrice which is cast into a new data type of varchar and printed out.

The price is also rounded to the nearest dollar.*/

DROP PROC IF EXISTS sp_OrderDetails_IDToPrice;

GO


CREATE PROC sp_OrderDetails_IDToPrice

@ProductID int, /*input*/

@ProductPrice decimal(6,2) OUTPUT

AS

BEGIN

SELECT @ProductPrice = ROUND(Price, 0)

FROM OrderDetails

WHERE ProductID = @ProductID;

END;

GO


DECLARE @ProductPrice decimal(6,2);

EXEC sp_OrderDetails_IDToPrice @ProductID = 10, @ProductPrice = @ProductPrice OUTPUT;

PRINT 'The Product Price is $' + CAST(@ProductPrice AS Varchar(10));

GO
```

```
/*OUTPUT*/

The Product Price is $6.00
```

Completion time: 2025-05-12T18:41:23.4743899-05:00