Table Of Contents

Exercise

Question 1:

- **Database Name**: Lab3
- **Table Name**: Supplier
- **Primary Key**: SupplierID

SupplierID	Name	Address
S01	ABC Company	Penang
S02	XYZ Company	Johor
S03	HJK Company	Selangor
S04	PQR Company	Selangor

Attributes	Data Type
SupplierID	nvarchar(50)
Name	nvarchar(50)
Address	nvarchar(50)

- **Table Name**: Product
- **Primary Key**: ProductID

ProductID	Name	Price_RM	QuantityInStock
P01	Keyboard	103.55	60
P02	Mouse	30.90	70
P03	Monitor	200.00	80
P04	Pendrive	40.30	50

Attributes	Data Type
ProductID	nvarchar(50)
Name	nvarchar(50)

Price_RM	decimal(10,2)
QuantityInStock	integer

- **Table Name**: Supplies
- **Primary Key**: SuppliesID
- **Foreign Key:** SupplierID, ProductID

SuppliesID	SupplierID	ProductID	SuppliedDate	QuantitySupplied
001	S01	P01	11/1/17	100
002	S01	P02	22/2/2017	200
003	S01	P03	NULL	300
004	S02	P03	30/4/2017	400

Attributes	Data Type
SuppliesID	nvarchar(50)
SupplierID	nvarchar(50)
ProductID	nvarchar(50)
SuppliedDate	date
QuantitySupplied	integer

a. Using MS SQL Server, create a new database Lab3

b. Write a query to create the tables given above

c. Write a query to add each row of data to the tables

Answer a.:

```
CREATE DATABASE Lab3
```

Answer b.:

Create Table Supplier(SupplierID nvarchar(50) PRIMARY KEY, Name nvarchar(50), Address nvarchar(50))	Create Table Product(ProductID nvarchar(50) PRIMARY KEY, Name nvarchar(50), Price_RM decimal(10,2), QuantityInStock integer)
Create Table Supplies(SuppliesID nvarchar(50) PRIMARY KEY, SupplierID nvarchar(50) FOREIGN KEY REFERENCES Supplier(SupplierID), ProductID nvarchar(50) FOREIGN KEY REFERENCES Product(ProductID), SuppliedDate date, QuantitySupplied integer)	

Answer c.:

INSERT INTO Supplier (SupplierID, Name, Address) VALUES ('S01','ABC Company','Penang'), ('S02','XYZ Company','Johor'), ('S03','HJK Company','Selangor'), ('S04','PQR Company','Selangor')	INSERT INTO Product (ProductID, Name, Price_RM, QuantityInStock) VALUES ('P01','Keyboard',103.55,60), ('P02','Mouse',30.90,70), ('P03','Monitor',200.00,80), ('P04','Pendrive',40.30,50)
INSERT INTO Supplies (SuppliesID, SupplierID, ProductID, SuppliedDate, QuantitySupplied) VALUES ('001','S01','P01','2017-01-11',100), ('002','S01','P02','2017-02-22',200), ('003','S01','P03',NULL,300), ('004','S02','P03','2017-04-30',400)	

SQL Comparison Operators:

- < smaller than
- = equal
- <= smaller and equal to
- > greater than
- >= greater and equal to
- != not equal
- <> not equal

Try Yourself:

- SQL> SELECT * FROM PRODUCT WHERE Price_RM < 50
- SQL> SELECT * FROM PRODUCT WHERE Price_RM <= 50
- SQL> SELECT * FROM PRODUCT WHERE Price_RM > 50
- SQL> SELECT * FROM PRODUCT WHERE Price_RM >= 50
- SQL> SELECT * FROM PRODUCT WHERE Name = 'MONITOR'
- SQL> SELECT * FROM PRODUCT WHERE Name != 'MONITOR'
- SQL> SELECT * FROM PRODUCT WHERE Name <> 'MONITOR'
- SQL> SELECT * FROM PRODUCT WHERE Price_RM < 50 AND Name = 'MONITOR'
- SQL> SELECT * FROM PRODUCT WHERE Price_RM < 50 OR Name = 'MONITOR'

SQL Logical Operator:

- ALL - TRUE if all of the subquery values meet the condition
- AND - TRUE if all the conditions separated by AND is TRUE
- ANY - TRUE if any of the subquery values meet the condition
- BETWEEN - TRUE if the operand is within the range of comparisons
- EXISTS - TRUE if the subquery returns one or more records
- IN - TRUE if the operand is equal to one of a list of expressions
- LIKE - TRUE if the operand matches a pattern
- NOT - Displays a record if the condition(s) is NOT TRUE
- OR - TRUE if any of the conditions separated by OR is TRUE
- SOME - TRUE if any of the subquery values meet the condition

Try Yourself:

- TRUE OR FALSE OR FALSE ⟶ TRUE

- FALSE AND FALSE AND TRUE ⟶ FALSE

- FALSE OR TRUE AND FALSE ⟶ FALSE

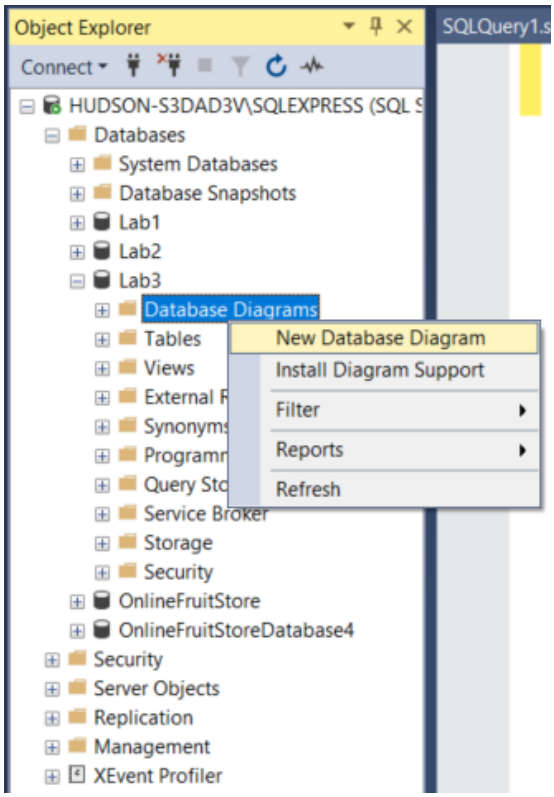
- **TRUE OR TRUE AND FALSE** ⟶ TRUE (AND has higher priority)
    - **Evaluate the inner part** (**FIRST**): "TRUE **AND** FALSE" evaluates to FALSE because both conditions must be true for the AND operation to be true.
    - **Evaluate the outer part**: "TRUE OR FALSE" evaluates to TRUE because only one condition needs to be true for the OR operation to be true.

- (TRUE OR TRUE) AND FALSE ⟶ FALSE (Parenthesis has higher priority)

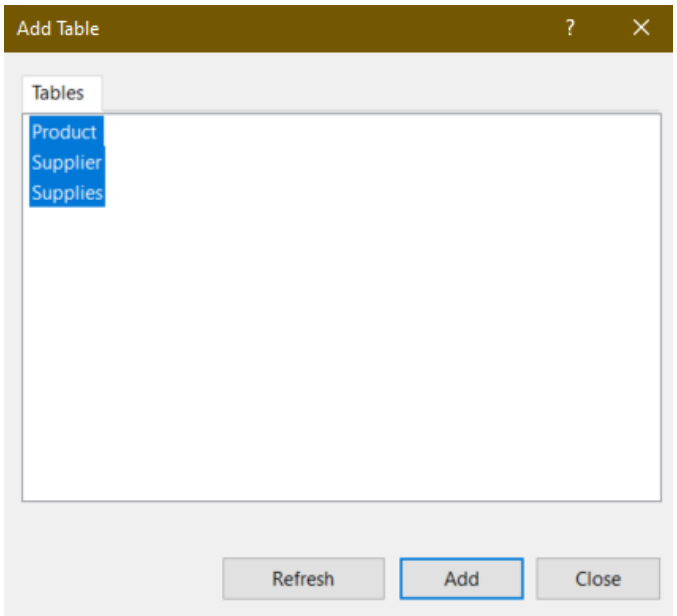
Question 2: Generate The Database Diagram

If you are using Lab PC (or if you're not using the Admin account), execute the statement below (no need to execute if you're using your own PC/using an admin account), this is to grant Admin access to the functions to create a database diagram.

Step 1:

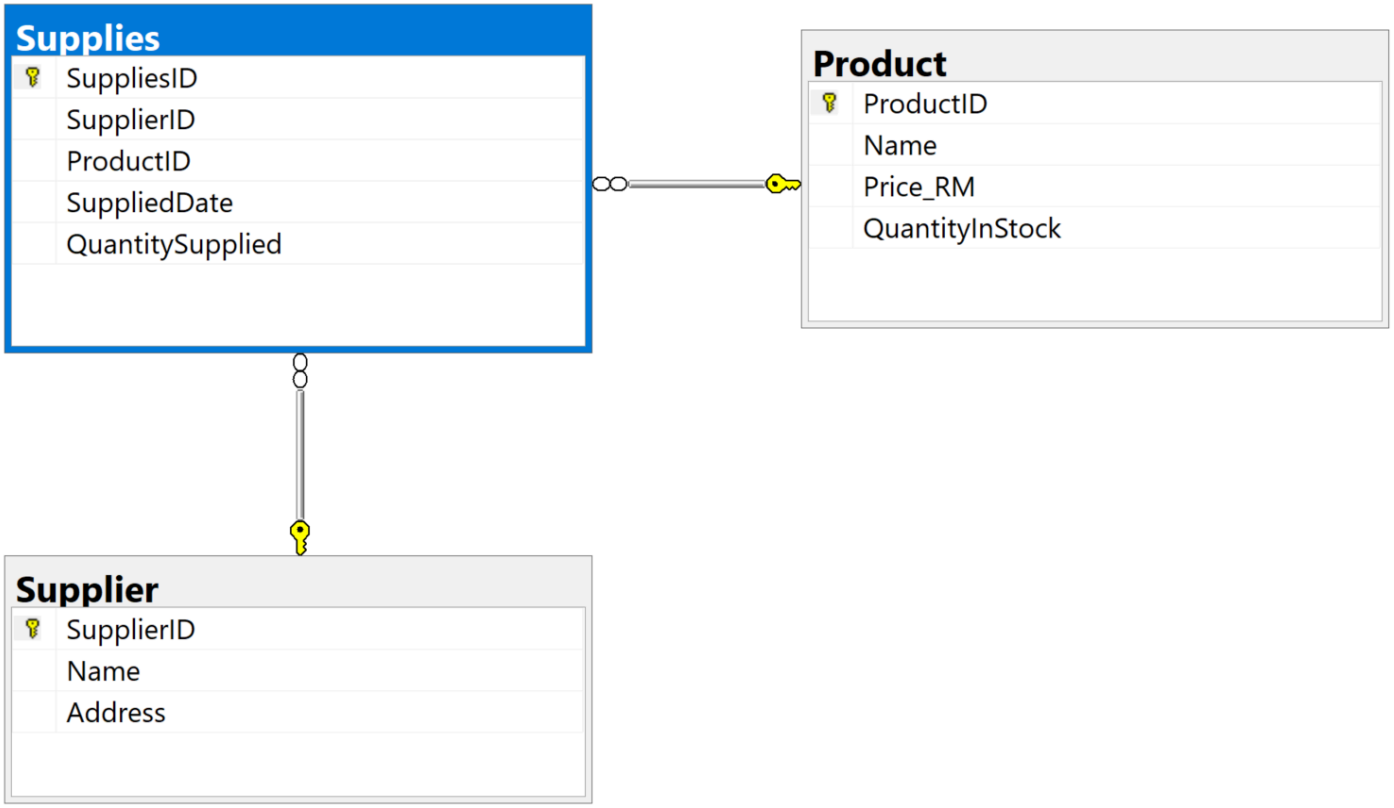


Step 2: Highlight All The Table → Click "Add"



Step 3: Database Diagram Appeared

- The Relation Type is One-To-Many



Question 3: Using SQL 'Select':

a. Display data from all columns from the supplier table

b. Display only product name and price from the product table

Answer a.:

SQL> SELECT * FROM Supplier

Answer b.:

SQL> SELECT Name,Price_RM FROM Product

Question 4: Using SQL 'Select Distinct':

Note:

- The SELECT DISTINCT statement is used in SQL to return only distinct (different) values. If there are duplicate records in the result set, DISTINCT ensures that only one copy of each distinct value is returned.

Key Concepts:

- **DISTINCT:** Ensures that duplicate rows in the result set are removed, so each row is unique. This is useful when you want to eliminate redundancy in your results.
- **Combination of Columns:** When DISTINCT is applied to multiple columns, it considers the combination of values across those columns. If the combination is the same across different rows, those rows are considered duplicates and will be filtered out.
- SELECT DISTINCT is used to eliminate duplicate rows from the result set.
- When applied to multiple columns, DISTINCT returns only unique combinations of those columns.
- It is particularly useful in situations where you need to ensure that the results do not contain redundant data.

a. Display only the DISTINCT values from the "Address" column in the Supplier table

Answer a.:

SQL> SELECT DISTINCT Address FROM Supplier

This query will return only the distinct values in the Address column, ignoring duplicates. The result set for this query would be:

	Address
1	Johor
2	Penang
3	Selangor

In this case, the Address column is considered independently, so only the unique addresses are returned, with "Selangor" appearing only once, even though it was in two different rows in the original table.

b. Display only the DISTINCT values from the "Address" & "Name" column in the Supplier table

This query will return a list of distinct pairs of Address and Name. Since both columns are considered together, only unique combinations of Address and Name will be returned. Based on your data, the result would be:

	Address	Name
1	Johor	XYZ Company
2	Penang	ABC Company
3	Selangor	HJK Company
4	Selangor	PQR Company

Here, even though "Selangor" appears twice in the Address column, the combination of Address and Name is different for each row, so both are included in the result.

Question 5: Using SQL 'IS NULL', 'IS NOT NULL'

a. Display all records from the Supplies table where SuppliedDate is a null

b. Display all records from the Supplies table where SuppliedDate is not a null

Answer a.:

SQL> SELECT * FROM Supplies WHERE SuppliedDate IS NULL

Answer b.:

SQL> SELECT * FROM Supplies WHERE SuppliedDate IS NOT NULL