**Database Final Project**

MUHAMMAD DEVA ALFILA MAJID – 2502042656

**Bluejack Jewelry**

**Bluejack Jewelry** is a Jewelry Store managed by your friend, Mia. Mia manages all of the activities that belong to **Bluejack Jewelry** like **selling jewelry to a customer** and **purchasing jewelry with a vendor**.

Every staff hired by **Bluejack Jewelry** has a task to **serve a customer who wants to buy jewelry** and **purchase jewelry from a vendor**. Every staff must be following the procedures to become a staff, which are:

* Every staff hired must have personal information like name, gender, email, address, phone. Every staff has an identification number with the following format:

“STXXX”

X => number between 0 – 9

* Staff can purchase jewelry with a vendor.
* Every **purchase transaction** made with the vendor has all the information about staff, vendor, purchase date, the jewelry purchased, and the quantity of each jewelry. Every **purchase transaction** has an identification number with the following format:

“PUXXX”

X => number between 0 – 9

* Staff can also serve a customer who wants to buy jewelry.
* Every **sales transaction** made by the customer has all the information about staff, customer, sales date, jewelry sold, and the quantity of each jewelry. Every **sales transaction** has an identification number with the following format:

“SLXXX”

X => number between 0 – 9

* Every jewelry purchased from a vendor or sold to a customer has its name, weight, purchase price, sales price, and stock. Every **jewelry** has an identification number with the following format:

“JWXXX”

X => number between 0 – 9

* Every jewelry has its type data that store the information about the type name, and the type data has an identification id with the following format:

“JTXXX”

X => number between 0 – 9

Every customer that wants to buy jewelry at **Bluejack Jewelry** must be following the **sales transaction procedures**, those are:

* Every customer that wants to purchase a product must complete personal information like name, gender, email, address, phone. Every customer has an identification number with the following format:

“CUXXX”

X => number between 0 – 9

* Customer can purchase **more than one jewelry** in every transaction.

Every vendor that wants to sell their jewelry must be following the **purchase transaction procedures**, those are:

* Every vendor that wants to sell their jewelry must complete personal information like name, email, address, and phone. Every vendor has an identification number with the following format:

“VEXXX”

X => number between 0 – 9

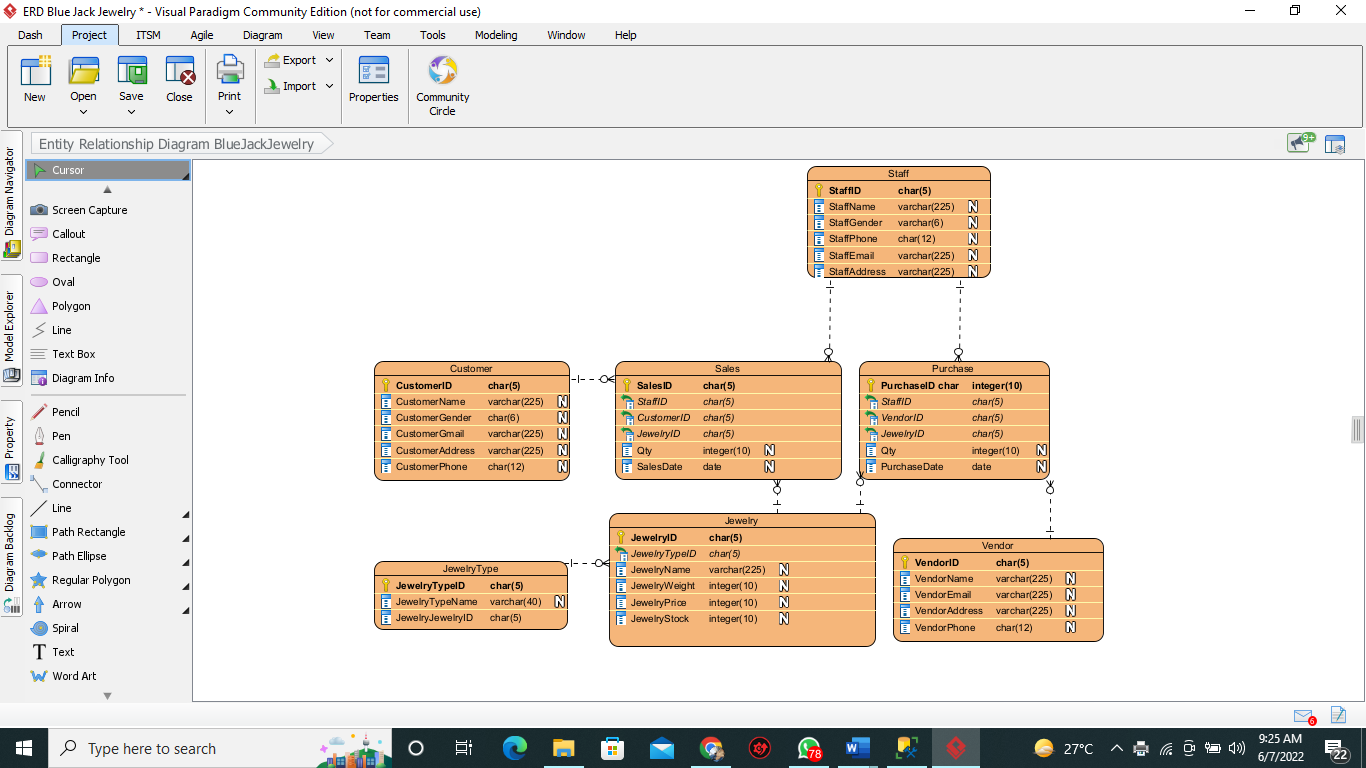
* Vendor can sell **more than one jewelry** in every transaction.

**Constraints:**

* Staff Name must be more than 3 characters.
* Staff Gender must be either “Male” or “Female” (without quote).
* Staff Email must ends with “st.com” (without quote).
* Staff Phone must starts with “08” (without quote).
* Customer Name must be more than 3 characters.
* Customer Gender must be either “Male” or “Female” (without quote).
* Customer Phone must starts with “08” (without quote).
* Vendor Name must be more than 3 characters.
* Vendor Email must ends with “ve.com” (without quote).
* Vendor Address must ends with a single character of number (0-9) and followed by ‘ street’, eg. ‘Denpasar 8 street’.
* Vendor Phone must starts with “08” (without quote).
* Jewelry Weight must be between 10 and 1000.
* Purchase and Sales Quantity must more than 0

Now **Bluejack Jewelry** still using a manual management system to maintain the **sales** and **purchase transactions**. You as his precious friend want to help **Bluejack Jewelry** to create a database system that can store data and maintain the **sales** and **purchase transactions**. The tasks that you must do are:

1. Create Entity Relationship Diagram to maintain **sales** and **purchase transactions**.



1. Create a database system using DDL syntax that is relevant to **sales** and **purchase transactions**.

/\* Create purchase table \*/

create table purchase (PurchaseID char(5) Primary Key Check (PurchaseID LIKE 'PU%'), VendorID char(5) check (VendorID LIKE 'VE%'), StaffID char(5) check (StaffID LIKE 'ST%'), JewelryID char(5) check (JewelryID like 'JW%'), Qty int, PurchaseDate Date);

/\* Create sales table \*/

create table sales (SalesID char(5) Primary Key Check (SalesID Like 'SL%'), StaffID char(5) check (StaffID LIKE 'ST%'), CustomerID char(5) check (CustomerID LIKE 'CU%'), JewelryID char(5) check (JewelryID like 'JW%'), Qty int, SalesDate Date);

1. Create query using DML syntax to fill the tables in database systems with data based on the following conditions:

* **Master** table must with more than or equals 10 data.
* **Transaction** table must be filled be filled with more than or equals 15 data.
* **Transaction detail** table must be filled with more than or equals 25 data.
* For the **Jewelry Type** table, the table must be filled with the following data:

|  |  |
| --- | --- |
| **Jewelry Type Names** | |
| Gold | Silver |
| Diamond | Bronze |
| Pearl | White Gold |
| Ruby | Green Ruby |
| Black Diamond | Blue Ruby |

/\* Insert Data into Jewelry \*/

insert into jewelry values

('JW001', 'JT001', 'Gold', '1000', '4500000', '100'),

('JW002', 'JT001', 'Diamond', '1000', '4500000', '100'),

('JW003', 'JT002', 'Pearl', '1000', '4500000', '100'),

('JW004', 'JT002', 'Ruby', '1000', '4500000', '100'),

('JW005', 'JT003', 'Black Diamond', '1000', '4500000', '100'),

('JW006', 'JT003', 'Silver', '1000', '4500000', '100'),

('JW007', 'JT004', 'Bronze', '1000', '4500000', '100'),

('JW008', 'JT004', 'White Gold', '1000', '4500000', '100'),

('JW009', 'JT005', 'Green Ruby', '1000', '4500000', '100'),

('JW010', 'JT005', 'Blue Ruby', '1000', '4500000', '100');

/\* Insert Data into Jewelry Type \*/

insert into jewelryType values

('JT001', 'Type 1'),

('JT002', 'Type 2'),

('JT003', 'Type 3'),

('JT004', 'Type 4'),

('JT005', 'Type 5');

1. Create query using DML syntax to simulate the transactions process for **sales** and **purchase transactions**.

**Note**: DML syntax to **fill database** and DML syntax to **simulate** the **transactions process** should be a **different query**.

insert into sales values

('SL001', 'ST001', 'CU003', 'JW001', '2', '07-06-2022'),

('SL002', 'ST001', 'CU003', 'JW002', '3', '08-06-2022'),

('SL003', 'ST001', 'CU003', 'JW003', '5', '09-06-2022'),

('SL004', 'ST002', 'CU002', 'JW004', '7', '09-06-2022'),

('SL005', 'ST002', 'CU002', 'JW005', '1', '10-06-2022'),

('SL006', 'ST002', 'CU002', 'JW006', '3', '10-06-2022'),

('SL007', 'ST003', 'CU001', 'JW007', '2', '10-06-2022'),

('SL008', 'ST003', 'CU001', 'JW008', '6', '11-06-2022'),

('SL009', 'ST003', 'CU002', 'JW009', '4', '11-06-2022'),

('SL010', 'ST003', 'CU003', 'JW010', '10', '12-06-2022');

/\* Insert Data into purchase \*/

insert into purchase values

('PU001', 'VE001', 'ST001', 'JW001', '10', '03-06-2022'),

('PU002', 'VE001', 'ST002', 'JW002', '10', '03-06-2022'),

('PU003', 'VE001', 'ST002', 'JW003', '10', '03-06-2022'),

('PU004', 'VE001', 'ST003', 'JW004', '10', '03-06-2022'),

('PU005', 'VE002', 'ST003', 'JW005', '10', '03-06-2022'),

('PU006', 'VE002', 'ST002', 'JW006', '10', '03-06-2022'),

('PU007', 'VE002', 'ST001', 'JW007', '10', '03-06-2022'),

('PU008', 'VE002', 'ST002', 'JW008', '10', '03-06-2022'),

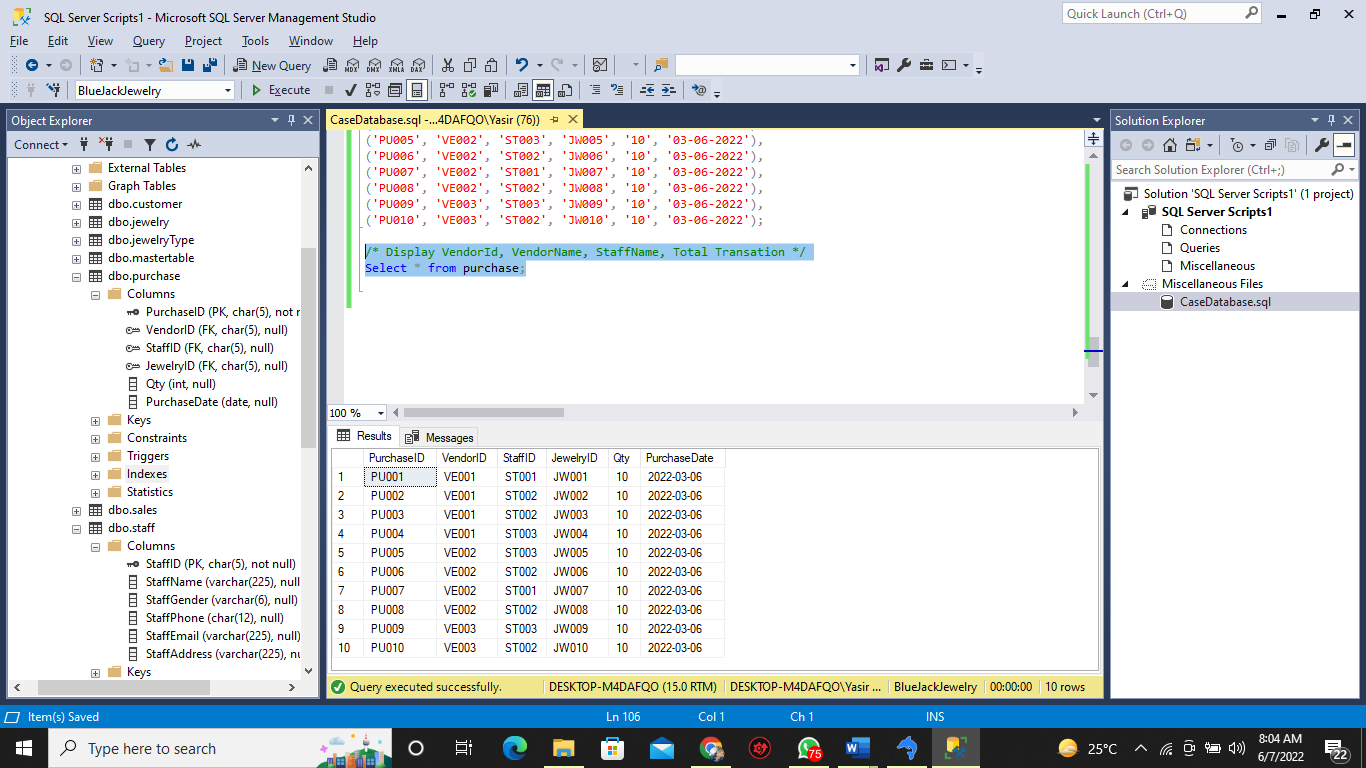
('PU009', 'VE003', 'ST003', 'JW009', '10', '03-06-2022'),

('PU010', 'VE003', 'ST002', 'JW010', '10', '03-06-2022');

1. To support database management process in **Bluejack Jewelry**,Mia asked you to provide some query that resulting important data. The requirements that asked from her are:
2. Display VendorId, VendorName, StaffName, Total Transation (obtained from the total of purchase transaction done by staff) for every purchase transaction which occurs in July and Vendor whose name is more than 1 word.

/\* Display VendorId, VendorName, StaffName, Total Transation \*/

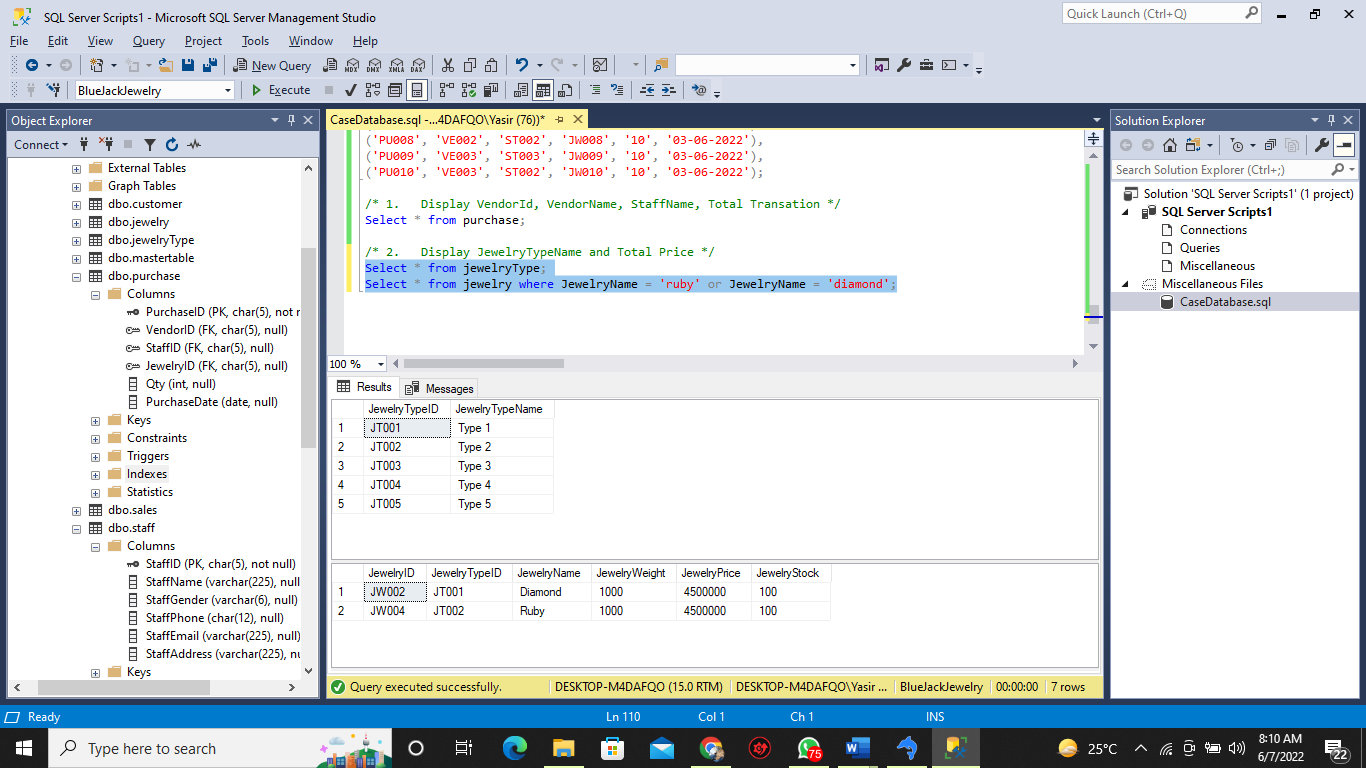
Select \* from purchase;



1. Display JewelryTypeName and Total Price (obtained from the total amount of sales price times quantity and ends with ' USD') for every type which name contains 'ruby' or 'diamond' and Total Price is greater than 40000.

Select \* from jewelryType;

Select \* from jewelry where JewelryName = 'ruby' or JewelryName = 'diamond';



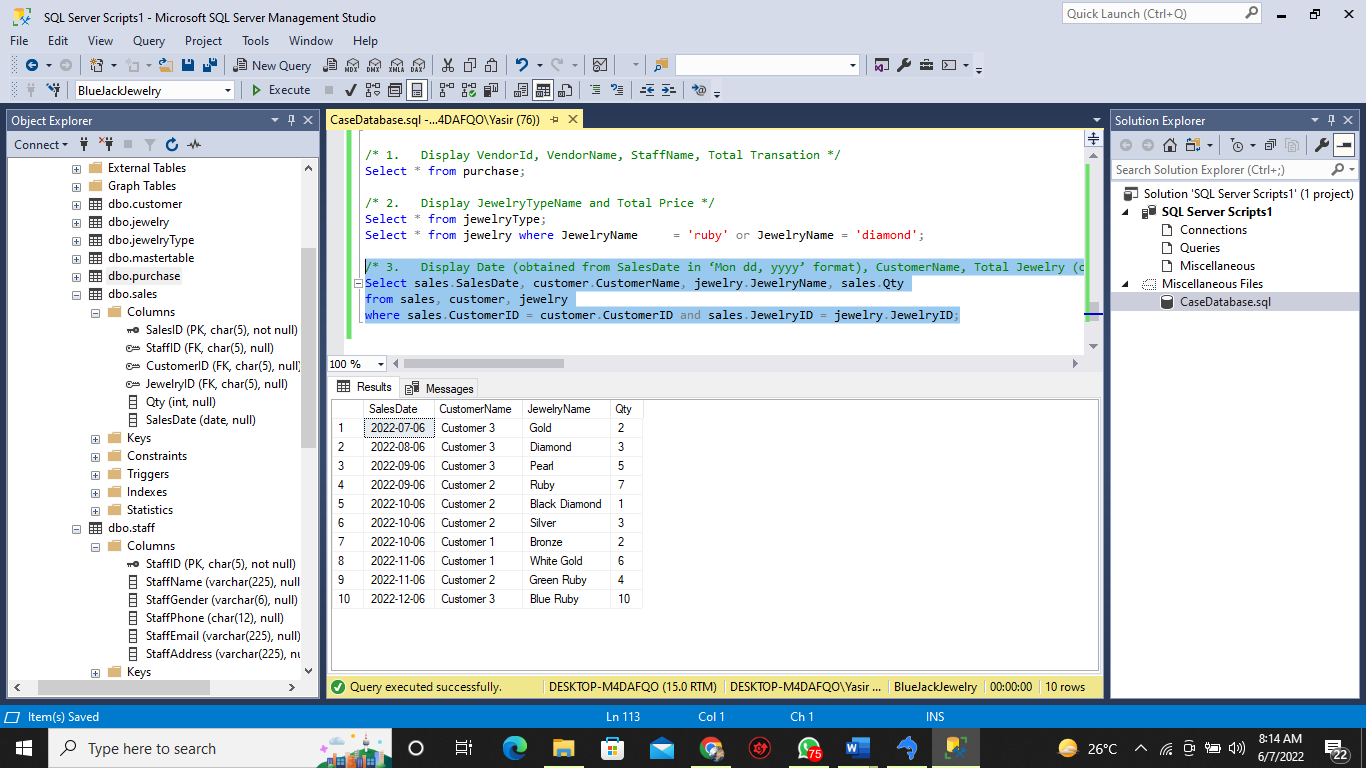
1. Display Date (obtained from SalesDate in ‘Mon dd, yyyy’ format), CustomerName, Total Jewelry (obtained from the total number of different jewelry bought by customer), and Total Quantity (obtained from the sum of the quantity of all jewelry) for every customer whose name has more than 10 characters and the transaction occurs in September.

/\* 3. Display Date (obtained from SalesDate in ‘Mon dd, yyyy’ format), CustomerName, Total Jewelry (obtained from the total number of different jewelry bought by customer), and Total Quantity \*/

Select sales.SalesDate, customer.CustomerName, jewelry.JewelryName, sales.Qty

from sales, customer, jewelry

where sales.CustomerID = customer.CustomerID and sales.JewelryID = jewelry.JewelryID;



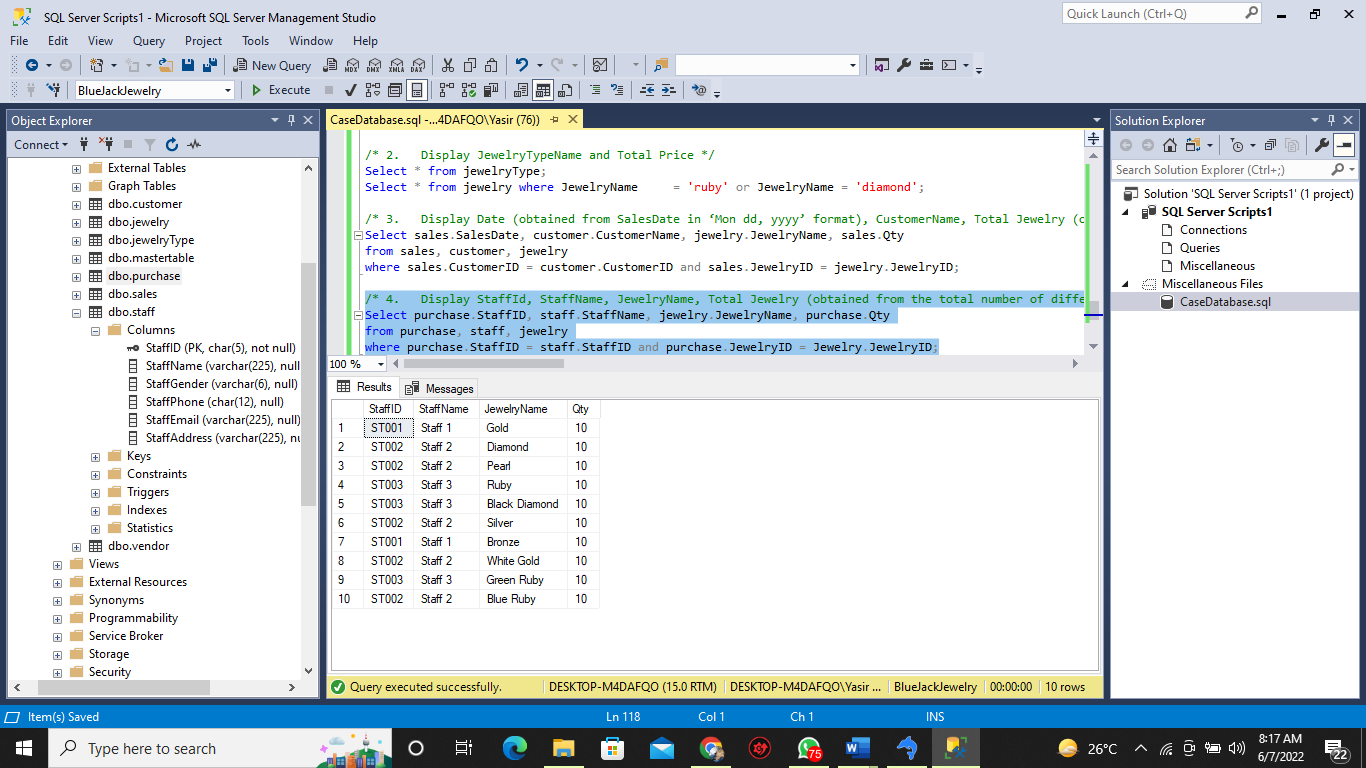
1. Display StaffId, StaffName, JewelryName, Total Jewelry (obtained from the total number of different jewelry purchase by staff), Total Price (obtained from the total amount of purchase price times quantity and ends with ' USD') for every transaction handled by a Staff whose gender is Male and Total Jewelry is more than equals 2, then sort the result based on Total Price in Descending.

/\* 4. Display StaffId, StaffName, JewelryName, Total Jewelry (obtained from the total number of different jewelry purchase by staff), Total Price \*/

Select purchase.StaffID, staff.StaffName, jewelry.JewelryName, purchase.Qty

from purchase, staff, jewelry

where purchase.StaffID = staff.StaffID and purchase.JewelryID = Jewelry.JewelryID;



1. Display CustomerName, CustomerEmail, CustomerAddress, JewelryName, and Jewelry Weight (ends with ' gram') for every Customer who buys the jewelry at a maximum JewelrySalesPrice and Customer Address contains with 'l' characters, then sort the result based on CustomerName in Ascending order. The duplicate data must be displayed once.

(**alias subquery**)

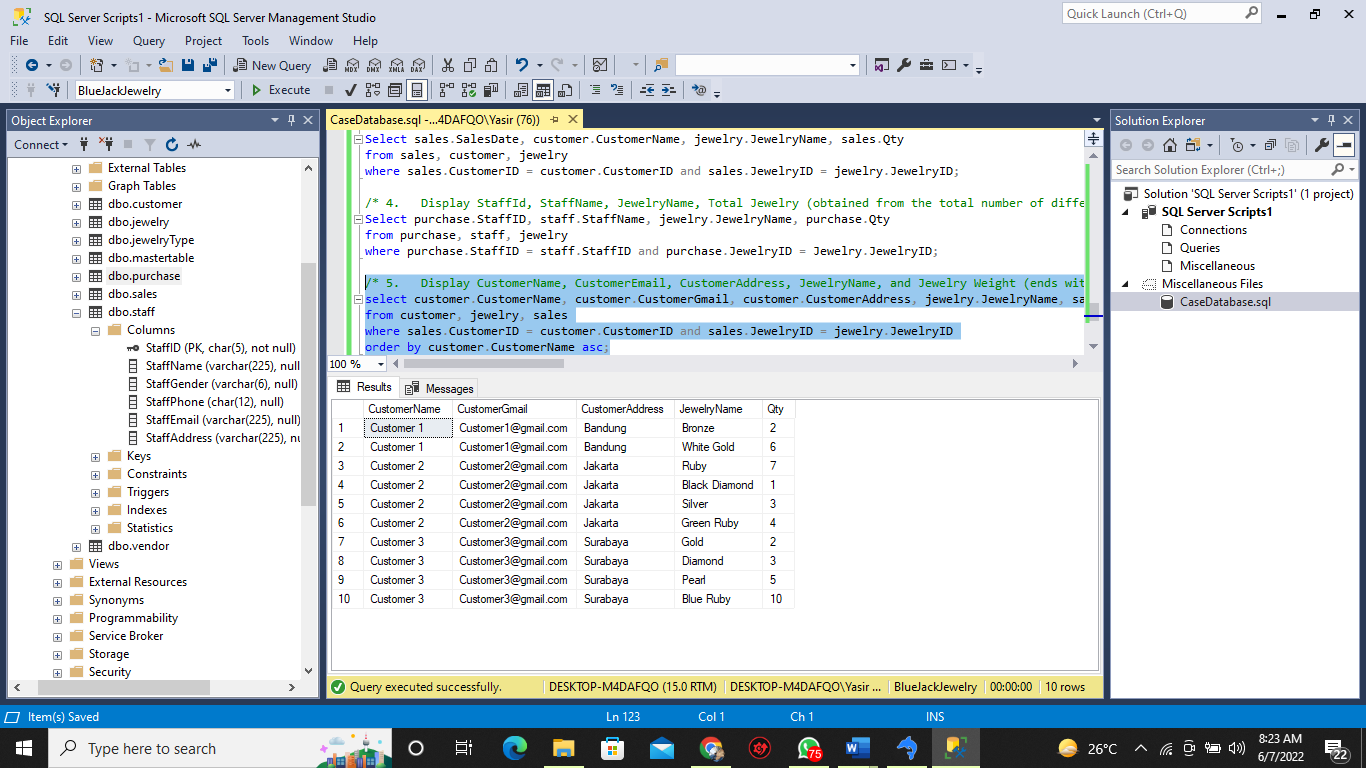
/\* 5. Display CustomerName, CustomerEmail, CustomerAddress, JewelryName, and Jewelry Weight (ends with ' gram') \*/

select customer.CustomerName, customer.CustomerGmail, customer.CustomerAddress, jewelry.JewelryName, sales.Qty

from customer, jewelry, sales

where sales.CustomerID = customer.CustomerID and sales.JewelryID = jewelry.JewelryID

order by customer.CustomerName asc;



1. Display VendorName, VendorEmail, VendorPhone, JewelryName, Jewelry Price (obtained from JewelryPurchasePrice and ends with ' USD') for every Vendor who sells the Jewelry at minimum JewelryPurchasePrice and VendorName contains 'n' characters, then sort the result based on VendorName in Descending order. The duplicate data must be displayed once.

(**alias subquery**)

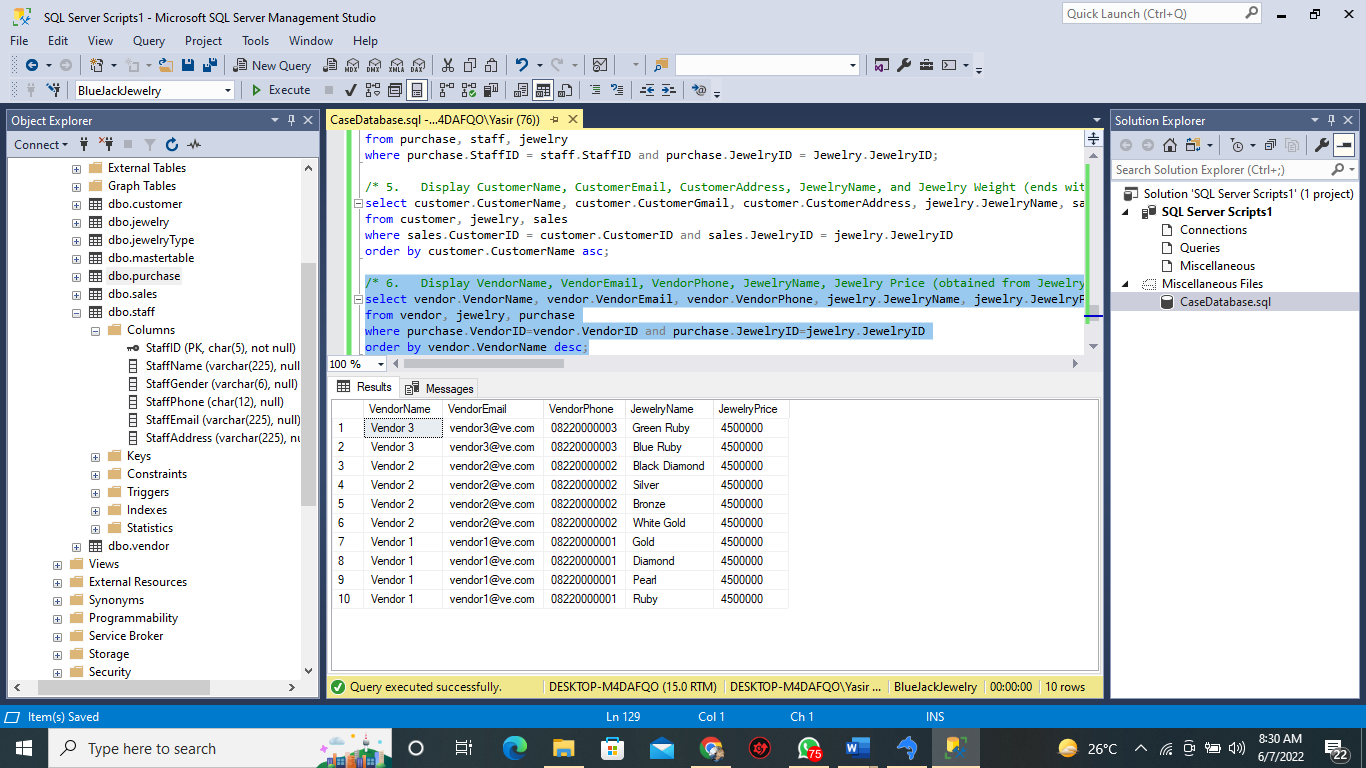
/\* 6. Display VendorName, VendorEmail, VendorPhone, JewelryName, Jewelry Price (obtained from JewelryPurchasePrice and ends with ' USD') for every Vendor who sells the Jewelry at minimum JewelryPurchasePrice \*/

select vendor.VendorName, vendor.VendorEmail, vendor.VendorPhone, jewelry.JewelryName, jewelry.JewelryPrice

from vendor, jewelry, purchase

where purchase.VendorID=vendor.VendorID and purchase.JewelryID=jewelry.JewelryID

order by vendor.VendorName desc;



1. Display Staff ID (obtained from StaffID and replace 'ST' with 'Staff '), StaffName, JewelryName, Total Price (obtained from the total amount of purchase price times quantity and ends with ' USD') for every transaction which quantity is more than the average of all purchase quantity and the transaction occurred in an odd day. Then sort the result based on StaffName in Ascending order

(**alias subquery**)

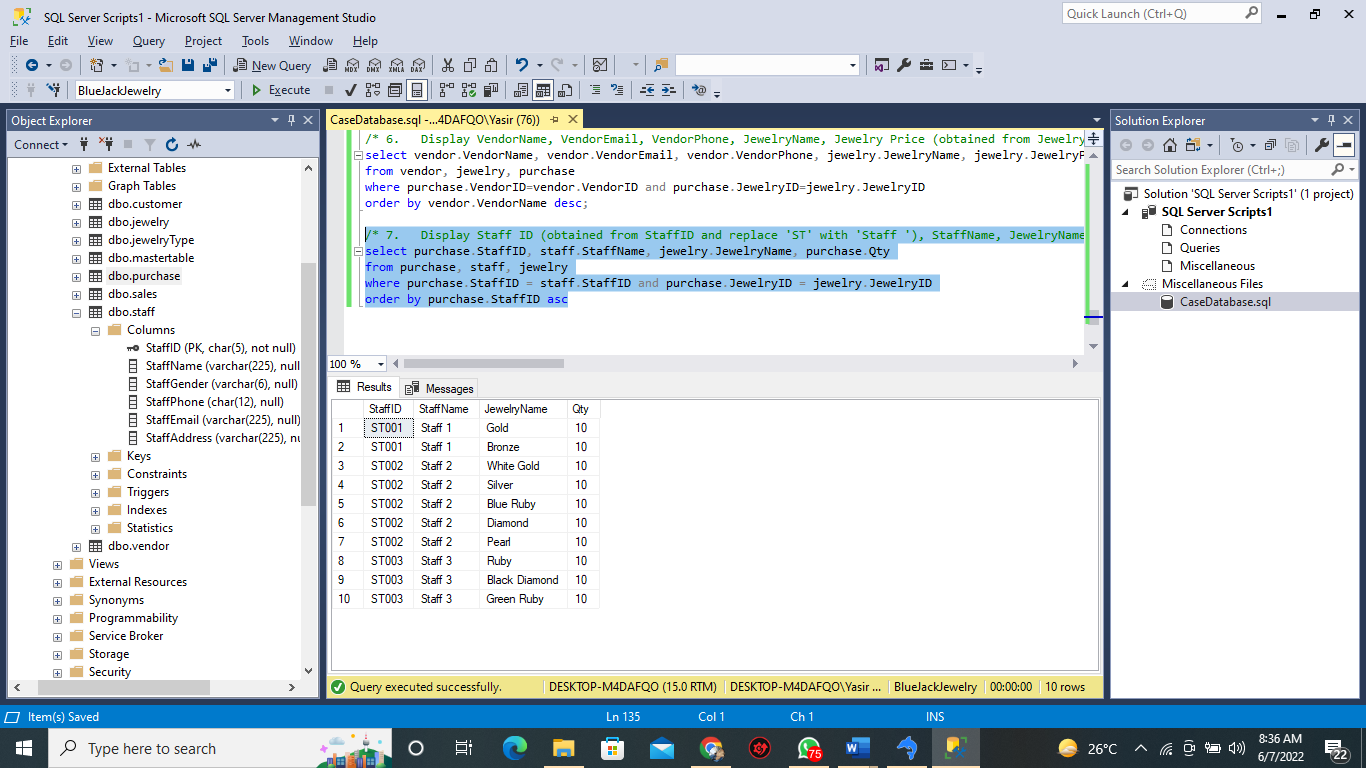
/\* 7. Display Staff ID (obtained from StaffID and replace 'ST' with 'Staff '), StaffName, JewelryName, Total Price (obtained from the total amount of purchase price times quantity and ends with ' USD') \*/

select purchase.StaffID, staff.StaffName, jewelry.JewelryName, purchase.Qty

from purchase, staff, jewelry

where purchase.StaffID = staff.StaffID and purchase.JewelryID = jewelry.JewelryID

order by purchase.StaffID asc



1. Display StaffId, Staff Name (Obtained by StaffName in Uppercase format), SalesId, and Total Price (obtained from the total amount of sales price times quantity and ends with ' USD') for every transaction with Total Price is higher than the average Total Price from every sales transaction and a Staff whose name contains 'o' character. Then sort the data based on Total Price in descending format.

(**alias subquery**)

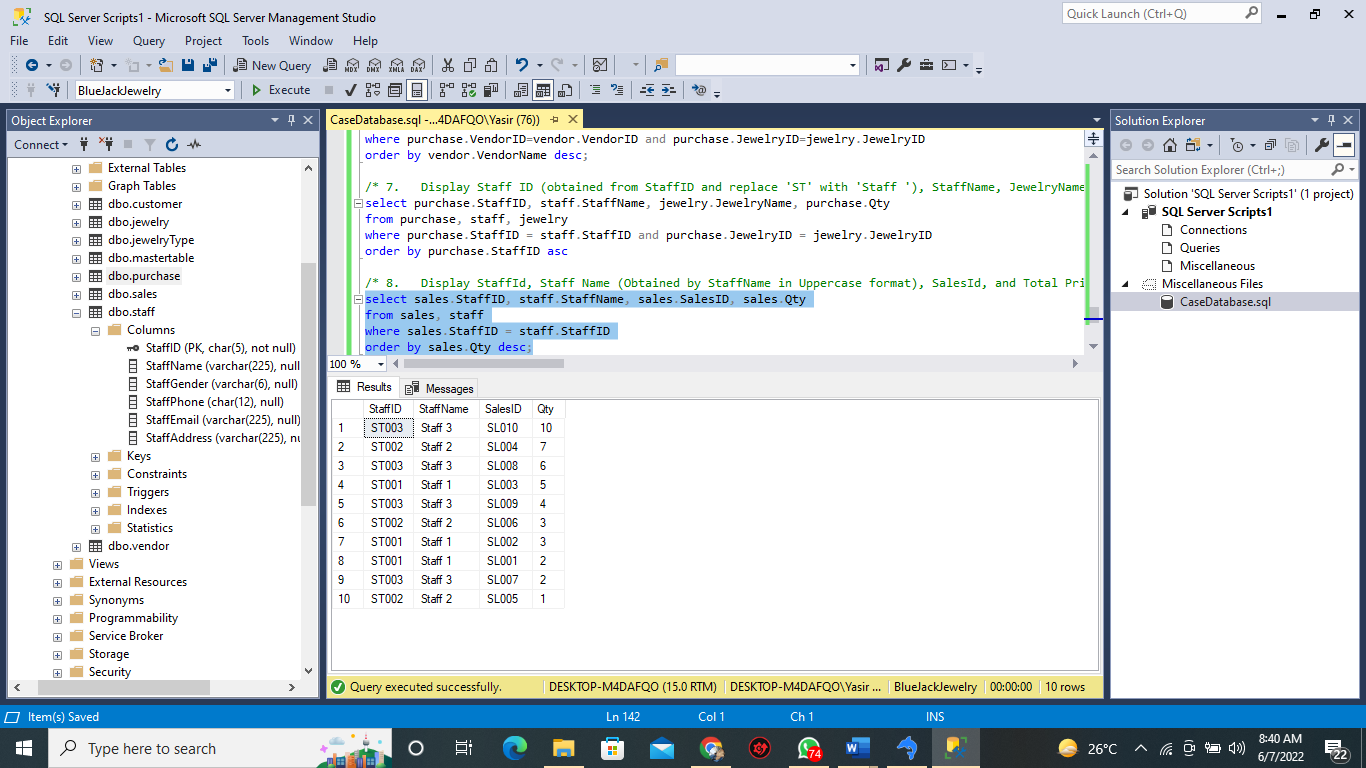
/\* 8. Display StaffId, Staff Name (Obtained by StaffName in Uppercase format), SalesId, and Total Price (obtained from the total amount of sales price times quantity and ends with ' USD') \*/

select sales.StaffID, staff.StaffName, sales.SalesID, sales.Qty

from sales, staff

where sales.StaffID = staff.StaffID

order by sales.Qty desc;



1. Create a view named [viewStaffSales] to display StaffId, Staff Name (obtained from StaffName in lowercase format), StaffEmail, Total Jewelry Sold (obtained from the total number of different jewelry sold by staff), and Average Quantity (obtained from the average of quantity) for every sales transaction which handled by staff whose name is not more than one word and more than 5 characters.

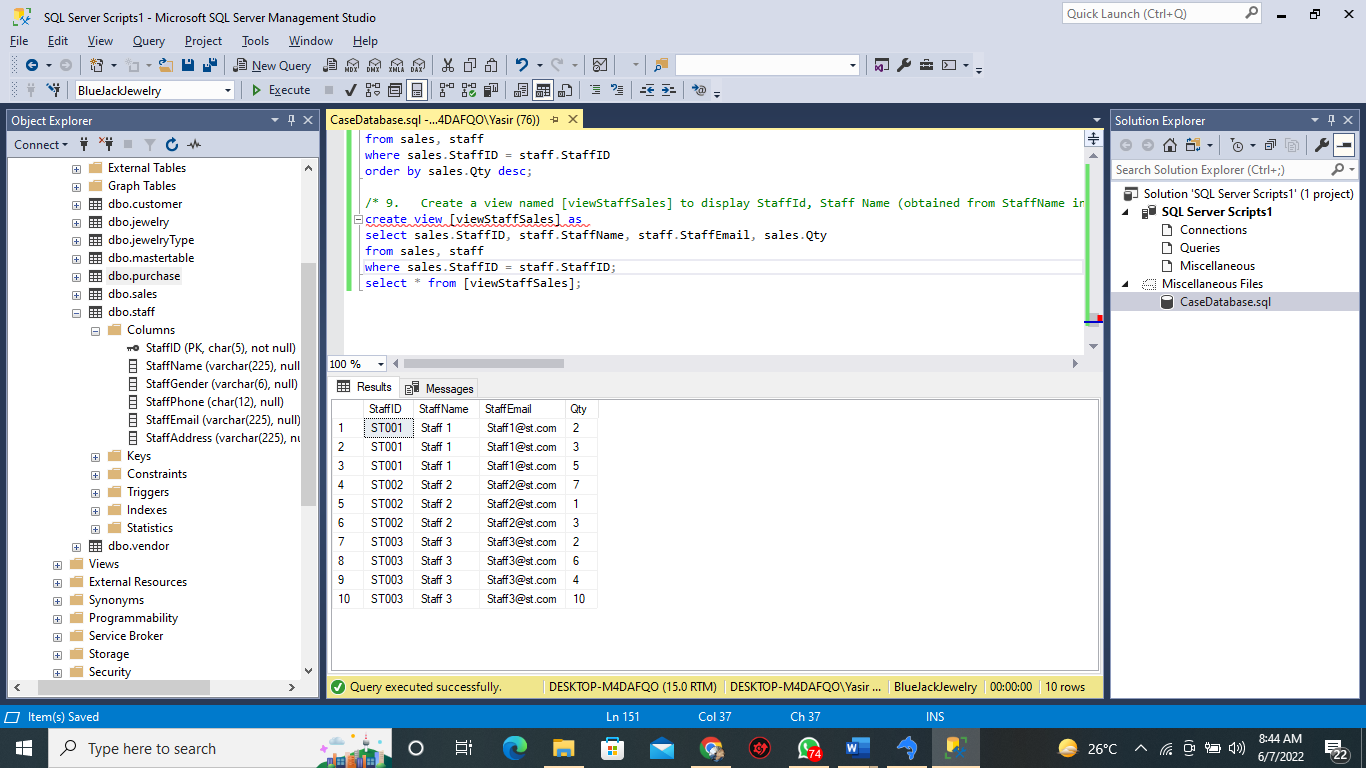
create view [viewStaffSales] as

select sales.StaffID, staff.StaffName, staff.StaffEmail, sales.Qty

from sales, staff

where sales.StaffID = staff.StaffID;

select \* from [viewStaffSales];



1. Create a view named viewVendorPurchase to display VendorId, VendorName, VendorEmail, VendorAddress, Total Jewelry Purchased (obtained from the total number of different jewelry sold by vendor), and Average Quantity (obtained from the average of quantity) for every purchase transaction done by a vendor whose VendorAddress' number is an odd number and the Total Purchased is more than 1.

/\* 10. Create a view named viewVendorPurchase to display VendorId, VendorName, VendorEmail, VendorAddress, Total Jewelry Purchased (obtained from the total number of different jewelry sold by vendor), and Average Quantity \*/

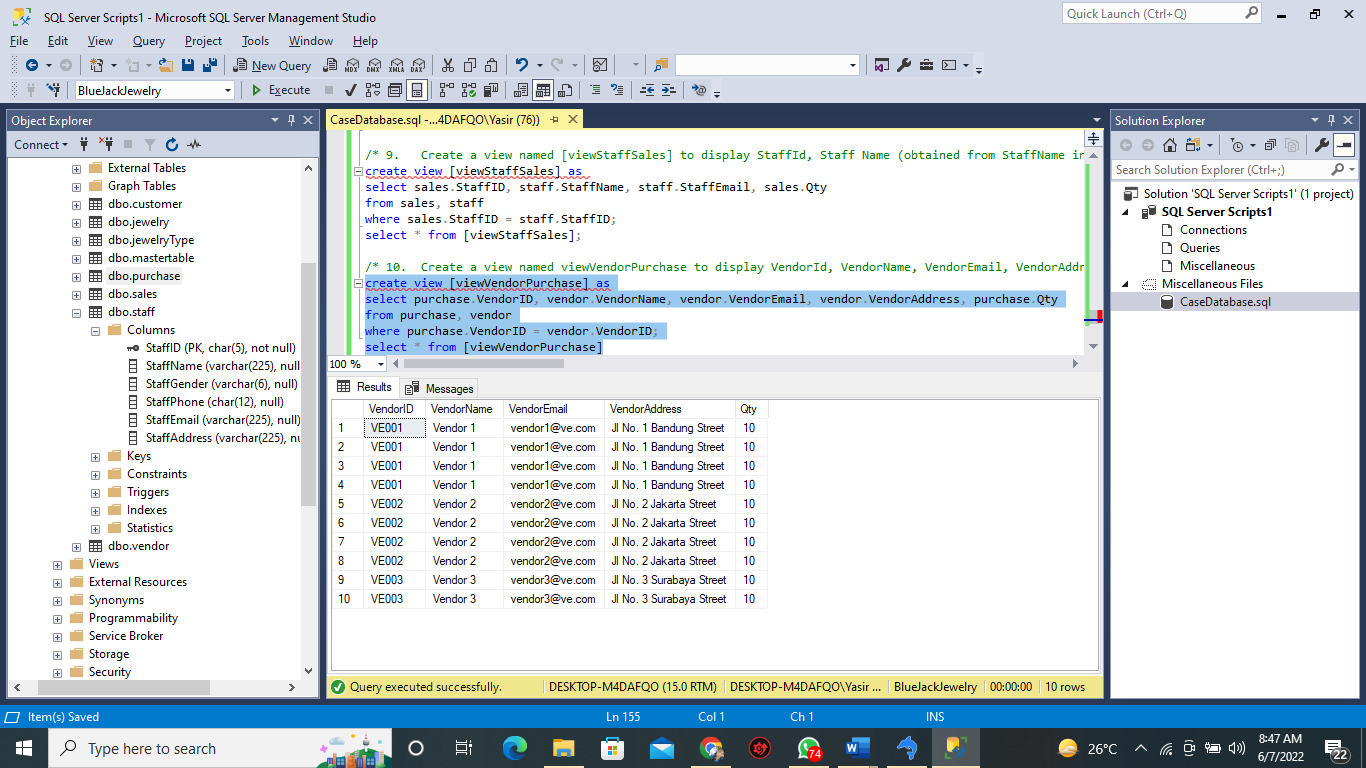
create view [viewVendorPurchase] as

select purchase.VendorID, vendor.VendorName, vendor.VendorEmail, vendor.VendorAddress, purchase.Qty

from purchase, vendor

where purchase.VendorID = vendor.VendorID;

select \* from [viewVendorPurchase]



**File that must be collected**:

1. Entity Relationship Diagram (.vpp, .png)
2. Query to create the database system. (.sql)
3. Query to insert data into tables. (.sql)
4. Query to simulate the transactions processes. (.sql)
5. Query to answer the 10 cases. (.sql)

**Here are the rules that you must follow to create your project:**

1. Use appropriate software for this subject based on **Sistem Praktikum** that can be downloaded from Binusmaya.
2. Use the techniques taught during practicum.
3. Collect appropriate files for this subject based on **Sistem Praktikum** that can be downloaded from Binusmaya.
4. Include the other files that can support your project, such as:
   * All files in your project
   * Other files (image, audio, video, etc.) used in your project
   * \*.DOC file (documentation of your project) that contains the reference links of additional files (image, audio, video, etc.) used in your project