**Sales and Logistics Application**

**Sharath Kumar Badam**

[**https://github.com/sharathbadam/Sales-and-Logistics.git**](https://github.com/sharathbadam/Sales-and-Logistics.git)

05/01/2022

**Initial Proposal**

In this project I am going to develop a sales and logistics application. This application integrates business operations, for sales and manufacturing.At the database level information from all the modules of the application needs to be stored.

In the database, mainly data related to customer and material are stored.

* Customer data – customer information like sales, transactional units,delivery and payment are stored here.
* Material data – information related to materials and services. The data can be purchasing data, accounting data, costing data and ware house management.

The database should be accessed, shared, and maintained easily.

Sales and agreement documents are created when a customer places an order in an organization to receive services and goods. While creating a sales order, the details of the customer and goods being purchased along with the quantity and time period to deliver the goods are recorded.

This kind of sales and agreement details are used in various business processes like standard orders,returns,delivery charges, credit amount, debit amount.

The application will also be receiving data from ERP andCRM systems, which integrates business operations, for sales and manufacturing.This integration is achieved by allowing different data sources and processes using various hardware and software products. It consists of various software applications or modules.

Hopefully I am looking forward to use python as presentation layer, Microsoft Azure for cloud services and SQL Server 2019 as Database systems in this sales and logistics application

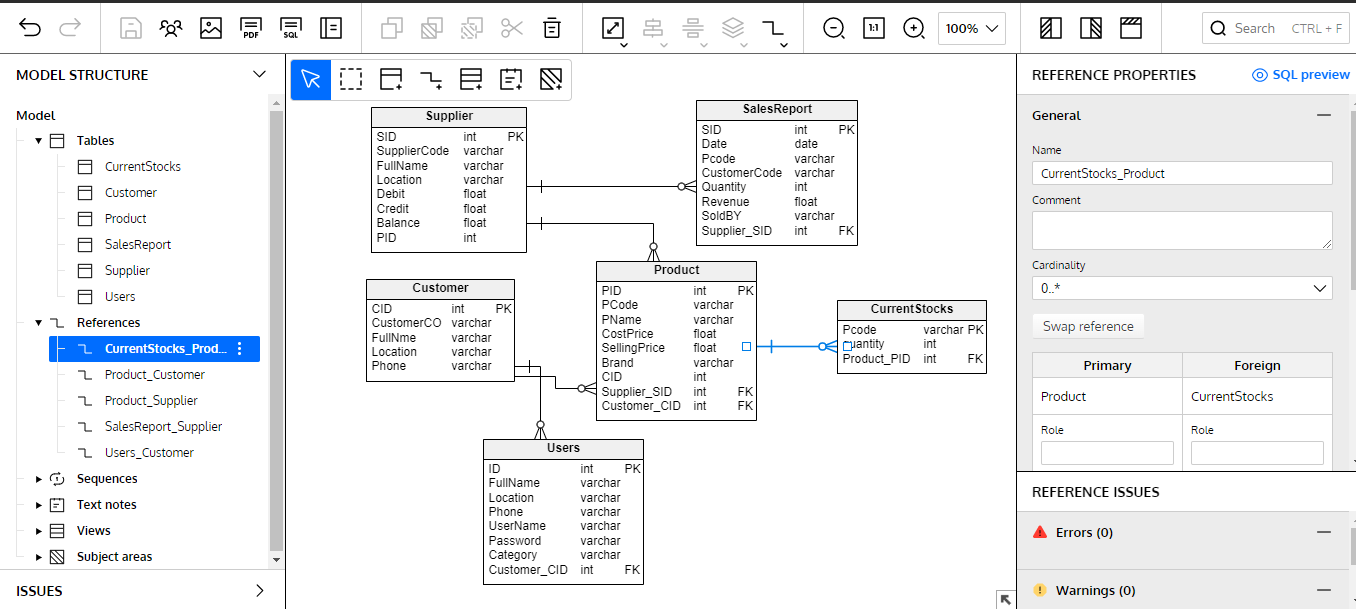
Finally, the data is useful for sales and logistics employees from different geographical locations.

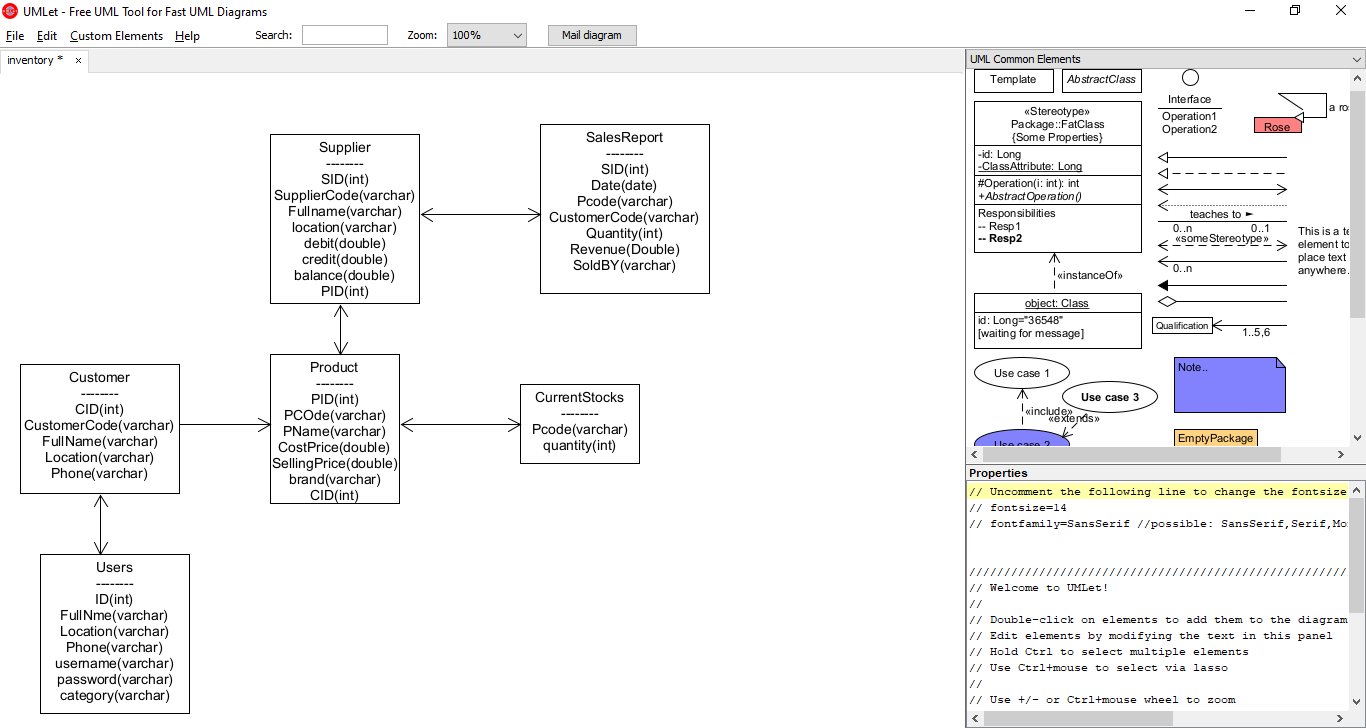
**Relational Database Design Process**

In this sales and logistics database design I am going to present few tables.

* Suppliers
* Customers
* Salesreport
* product
* currentstock
* user

ER Diagrams





DDL:

**Database Tables Definition:**

**Suppliers**

{ sid(PK) int,

suppliercode varchar,

fullname varchar,

location varchar,

debit double,

credit double,

balance double

PID int}

**Customers**

{ cid(PK) int,

CustomerCode varchar,

fullname varchar,

location varchar,

Phone varchar}

**products**

{pid(PK) int,

pcode varchar,

pname varchar,

costprice double,

sellingprice double,

brand varchar,

CID int}

**Currentstocks**

{ productcode varchar,

quantity int }

**salesreport**

{ SID int,

date varchar,

customercode varchar,

pcode varchar,

quantity int,

revenue double,

soldby varchar }

**users**

{ID int,

FullName varchar,

Location varchar,

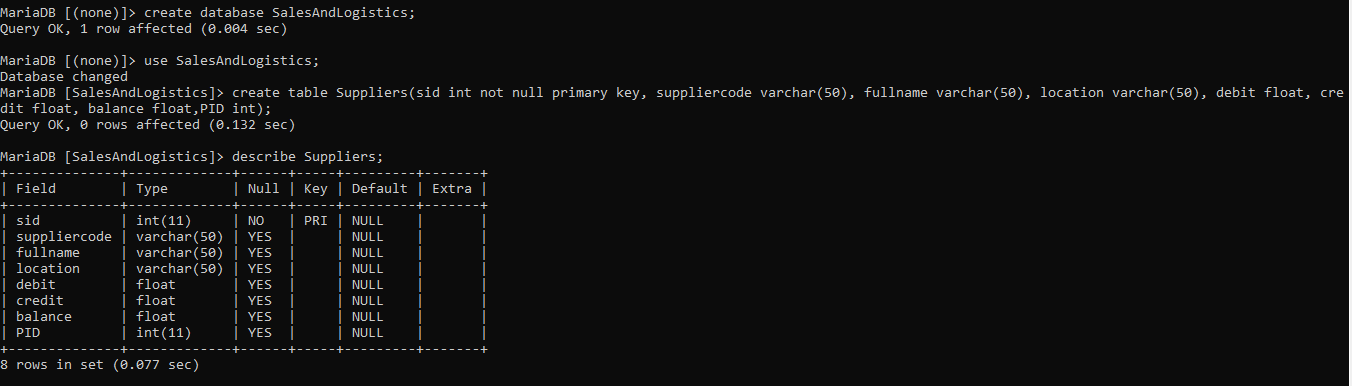
Phone varchar,

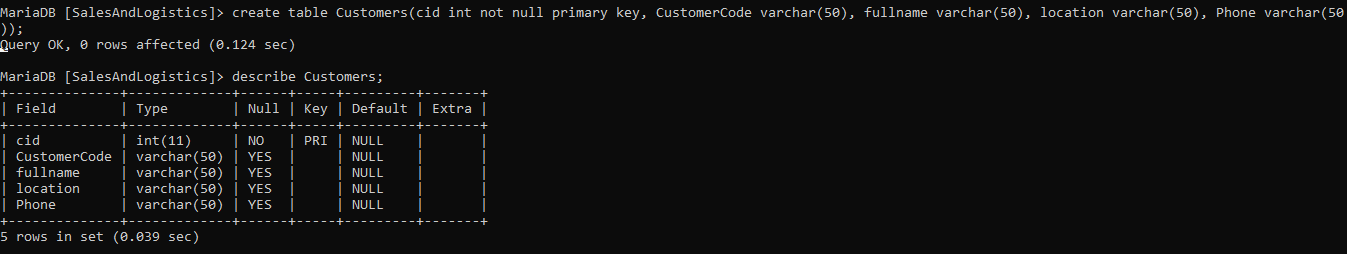
Username varchar,

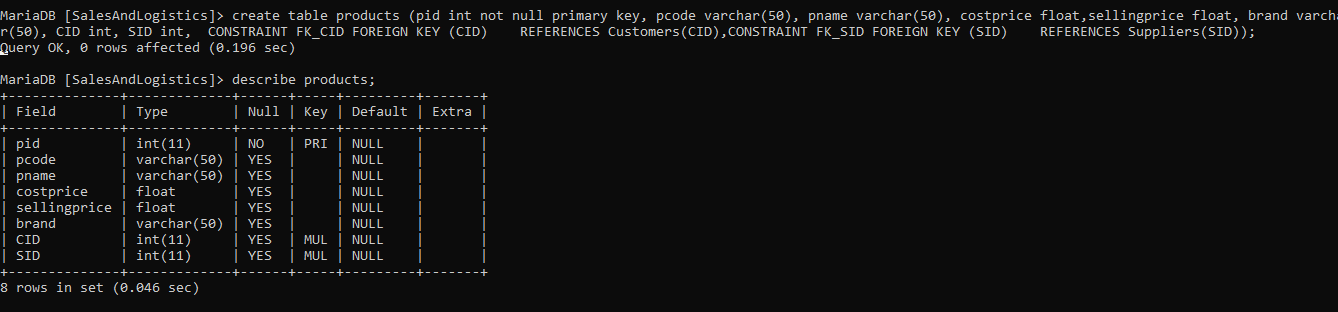
Password varchar,

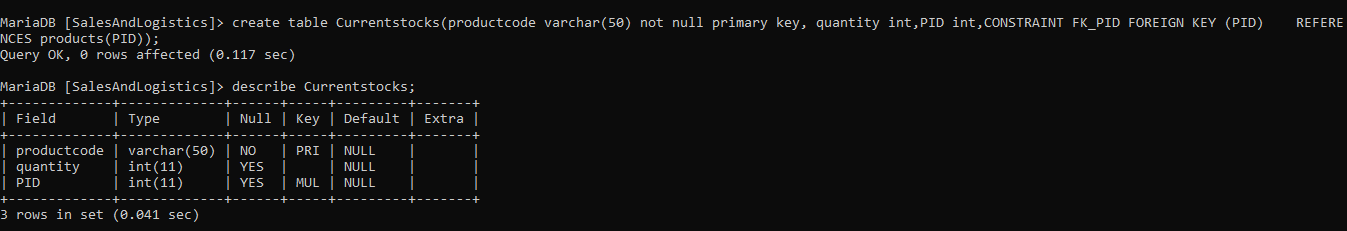
Category varchar}

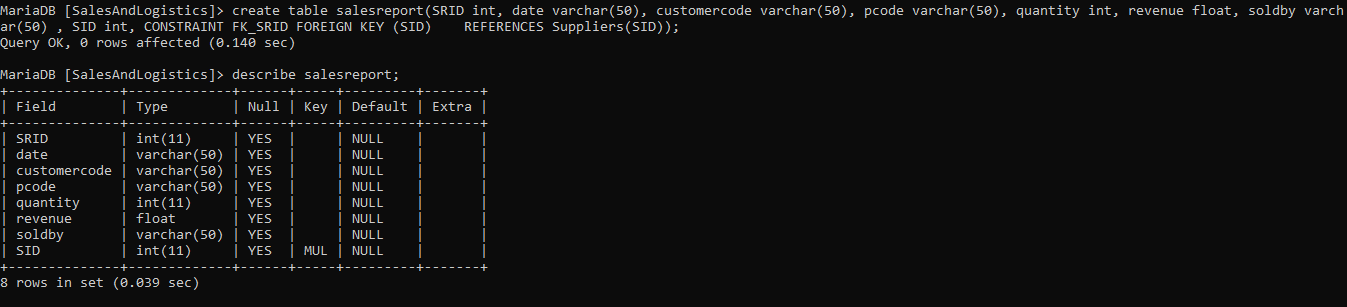
CREATING TABLES IN MARIA DB:

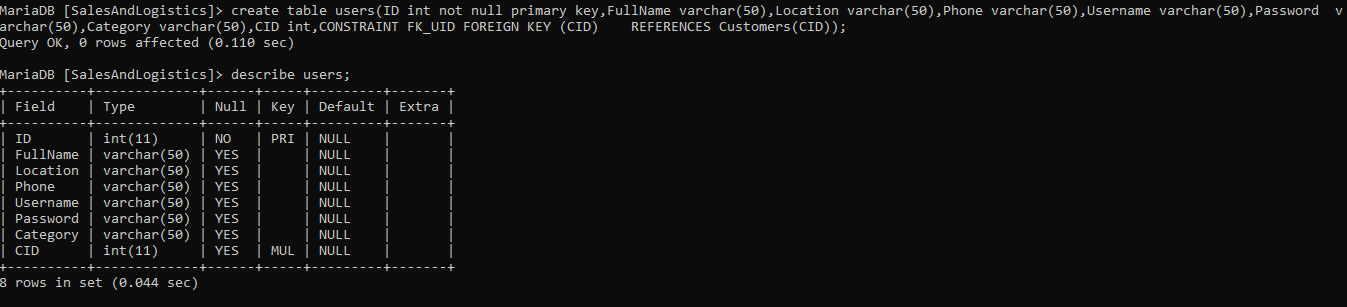




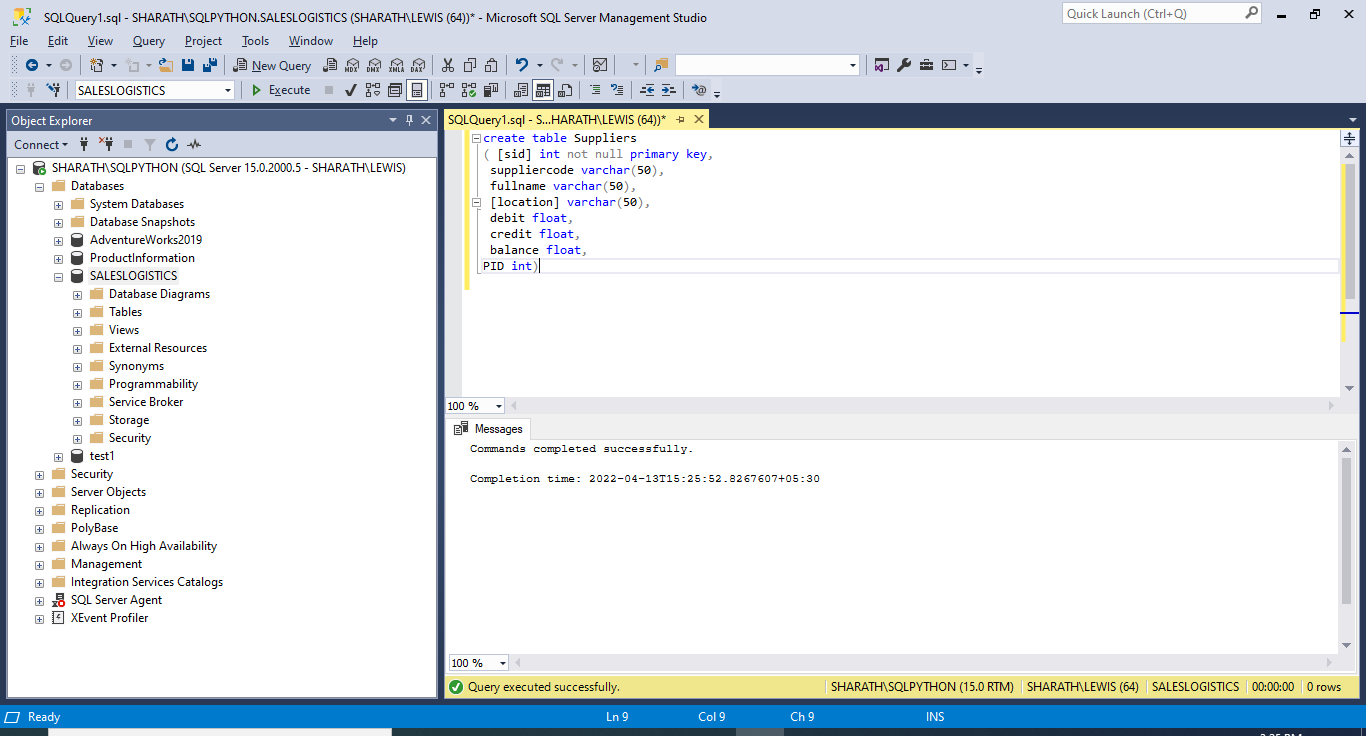




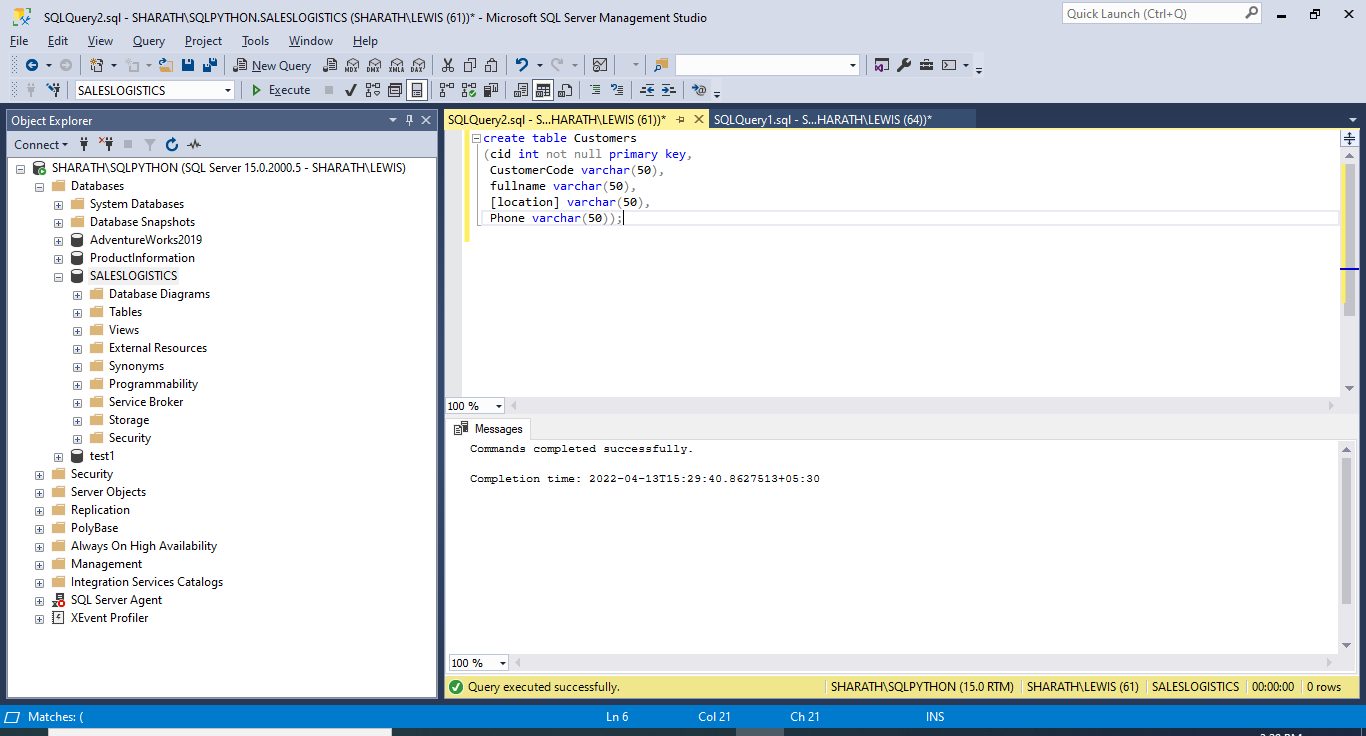




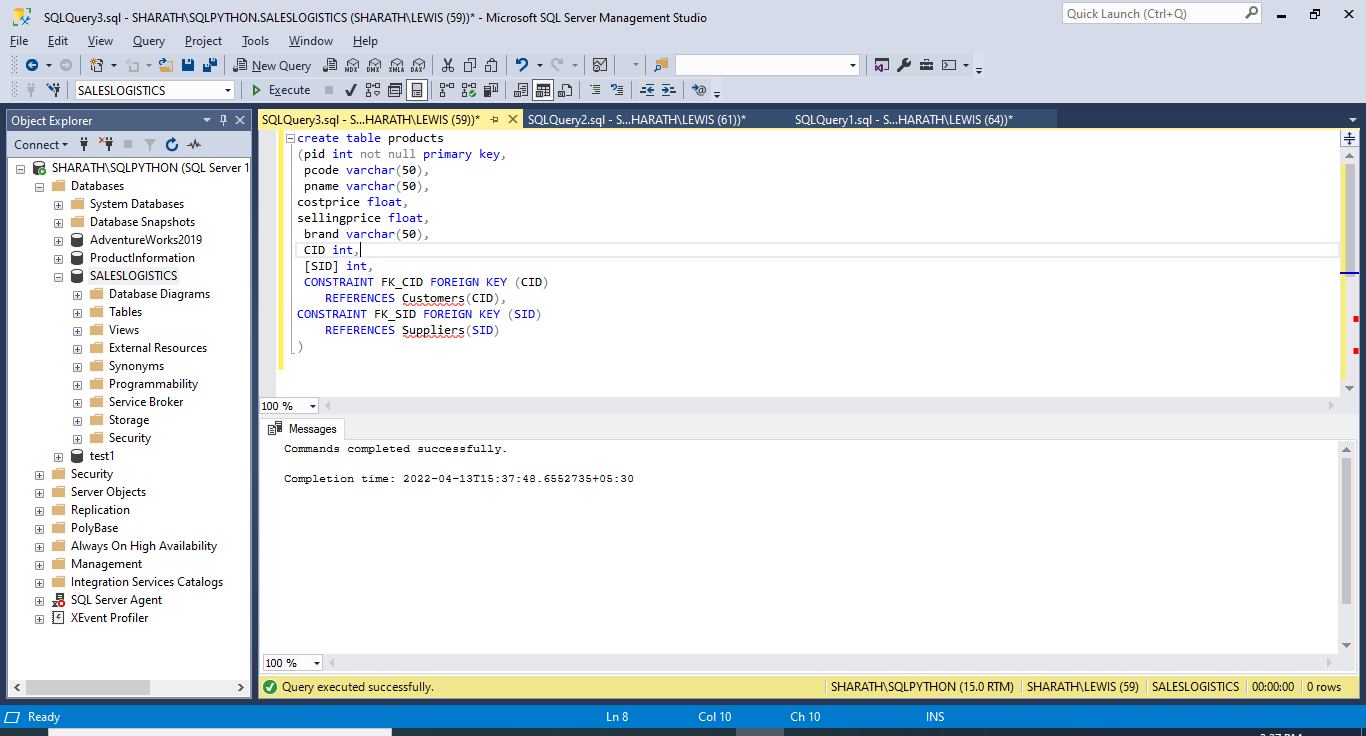
Creating supplier table in MS SQL server using create command



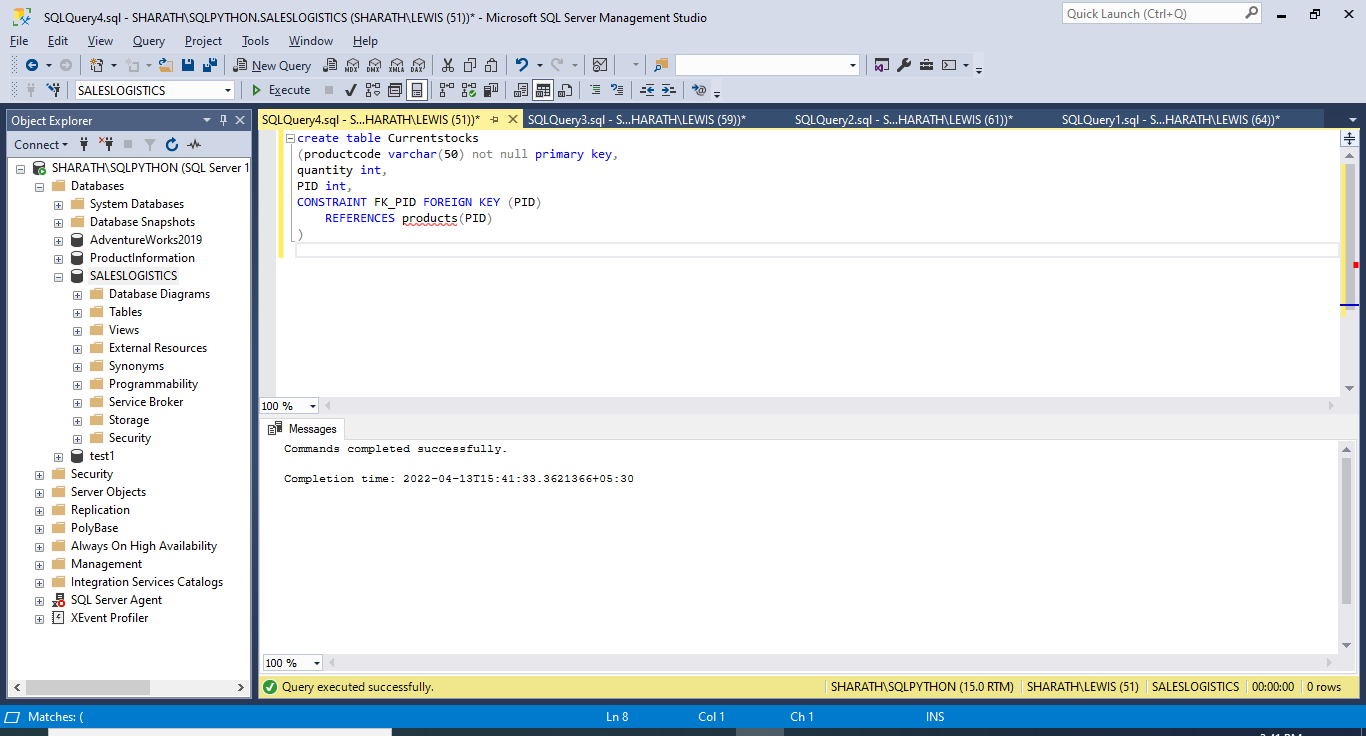
Creating customer table in MS SQL server using create command



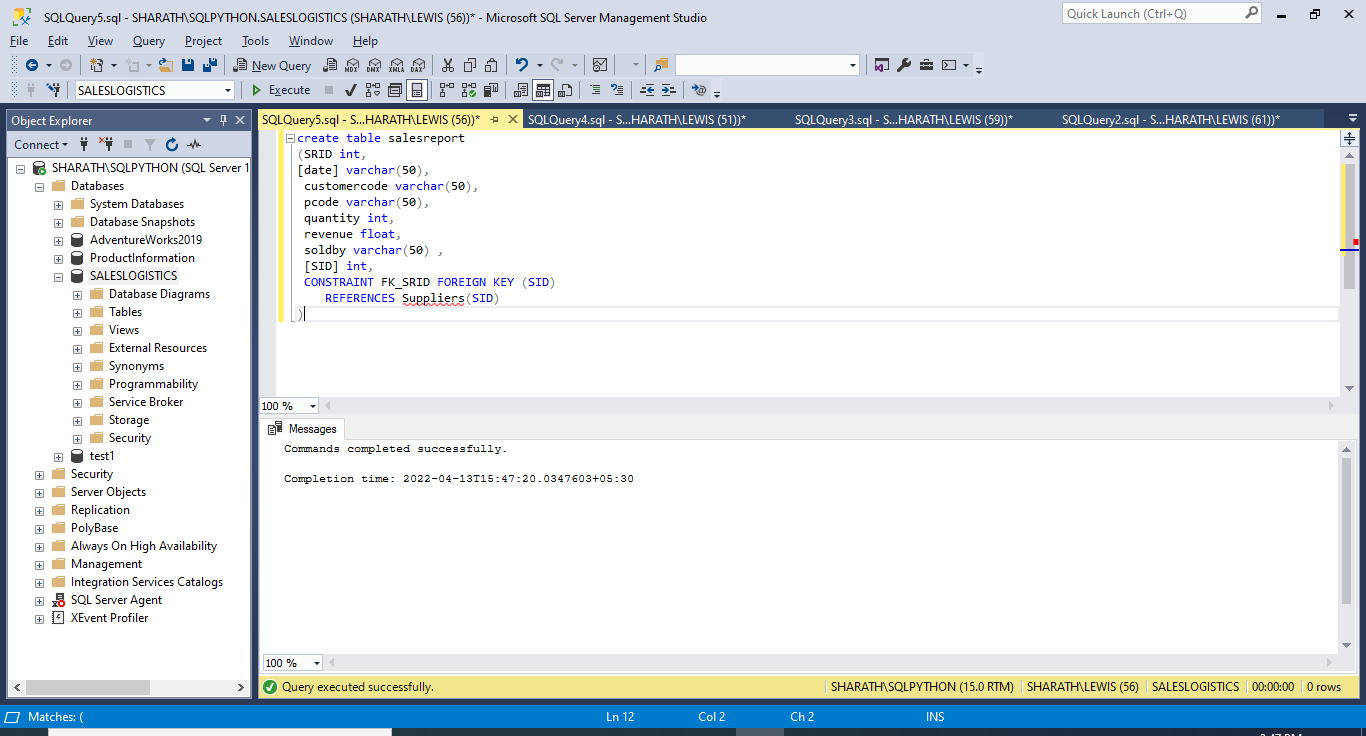
Creating product table in MS SQL server using create command



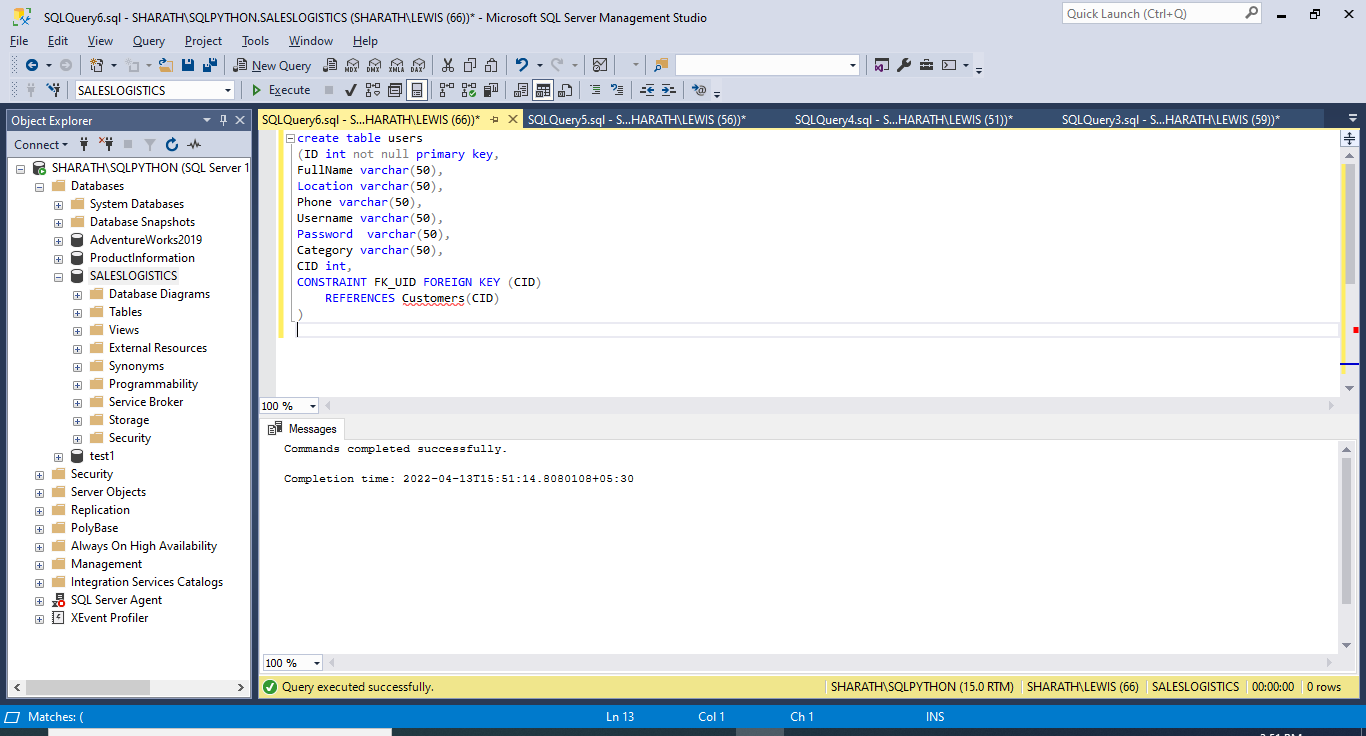
Creating current stock table in MS SQL server using create command



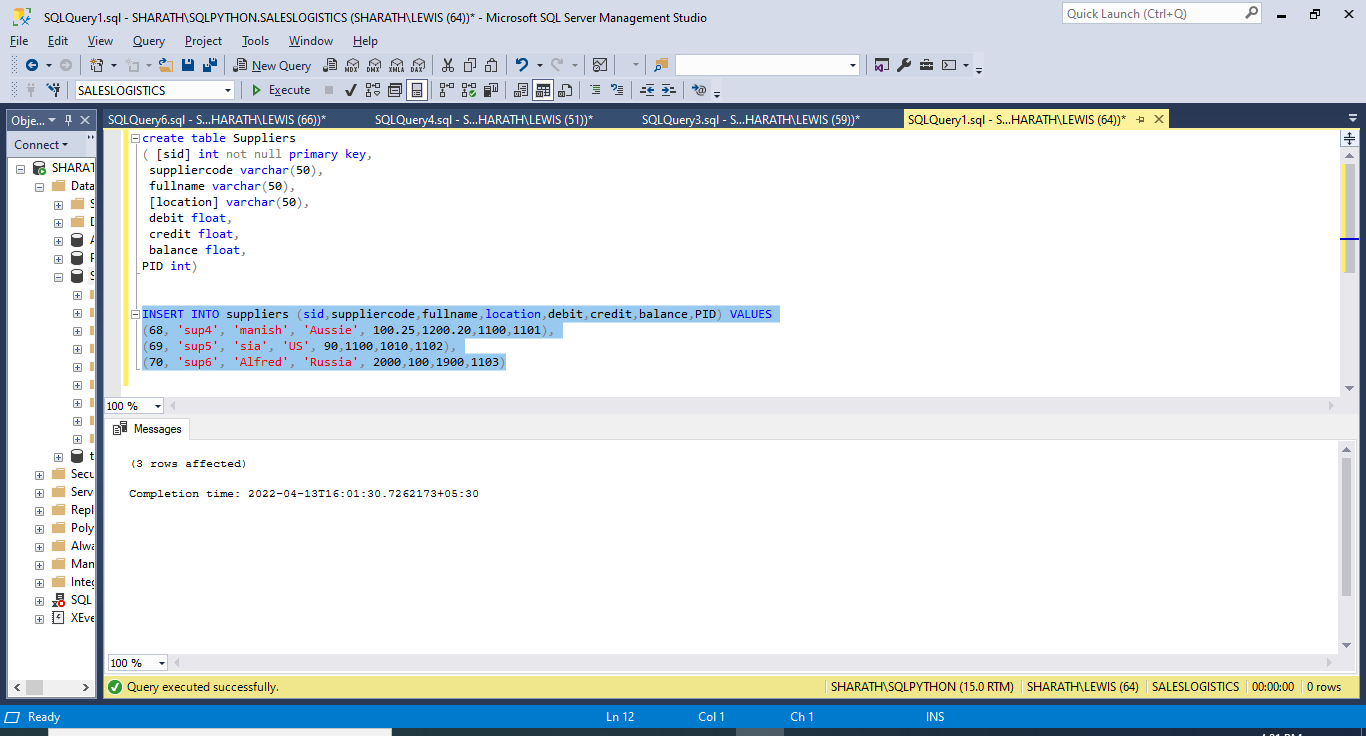
Creating salesreport table in MS SQL server using create command

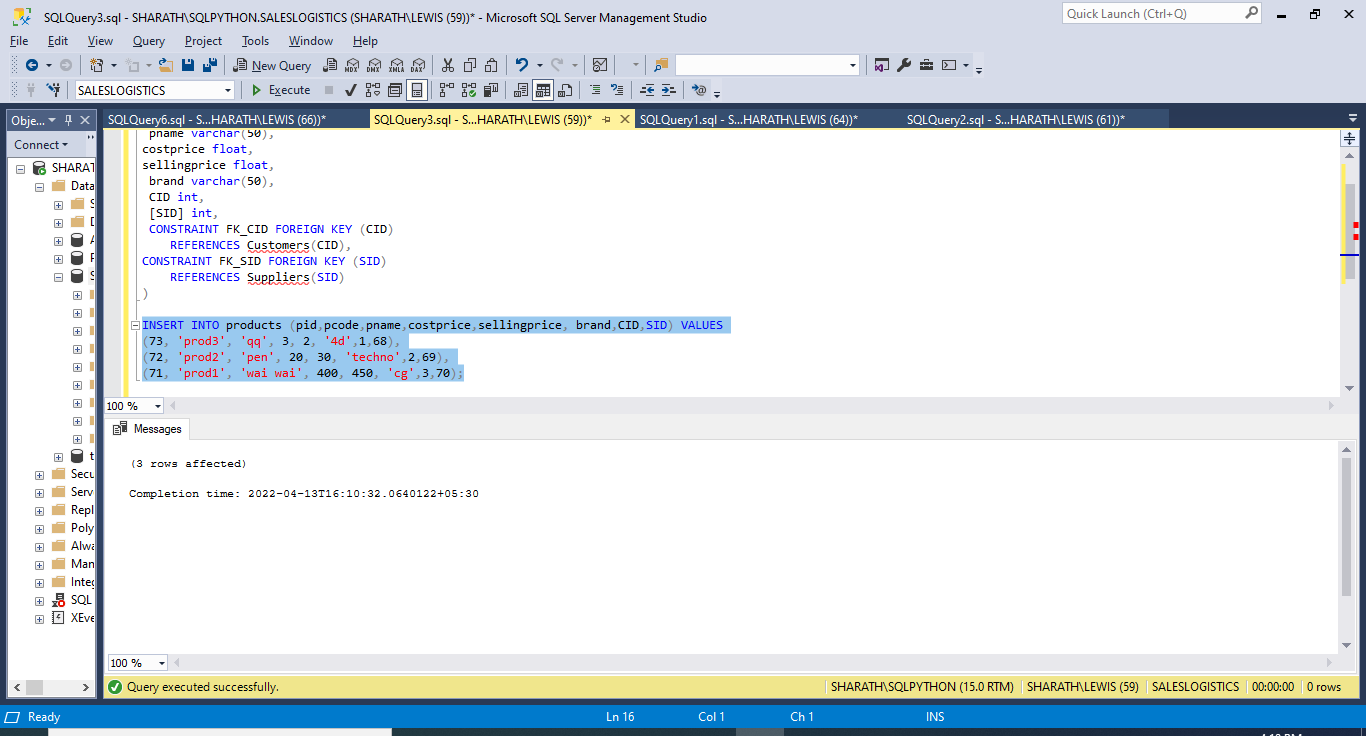


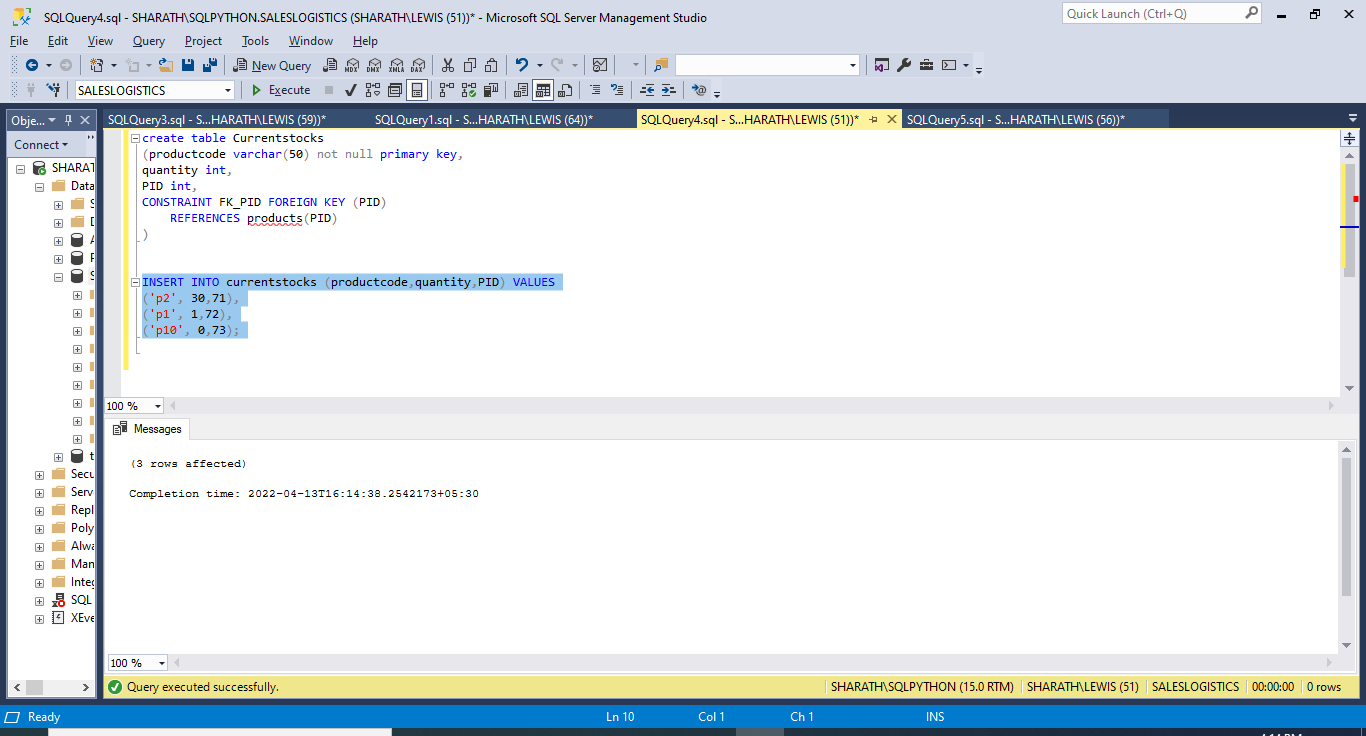
Creating users table in MS SQL server using create command

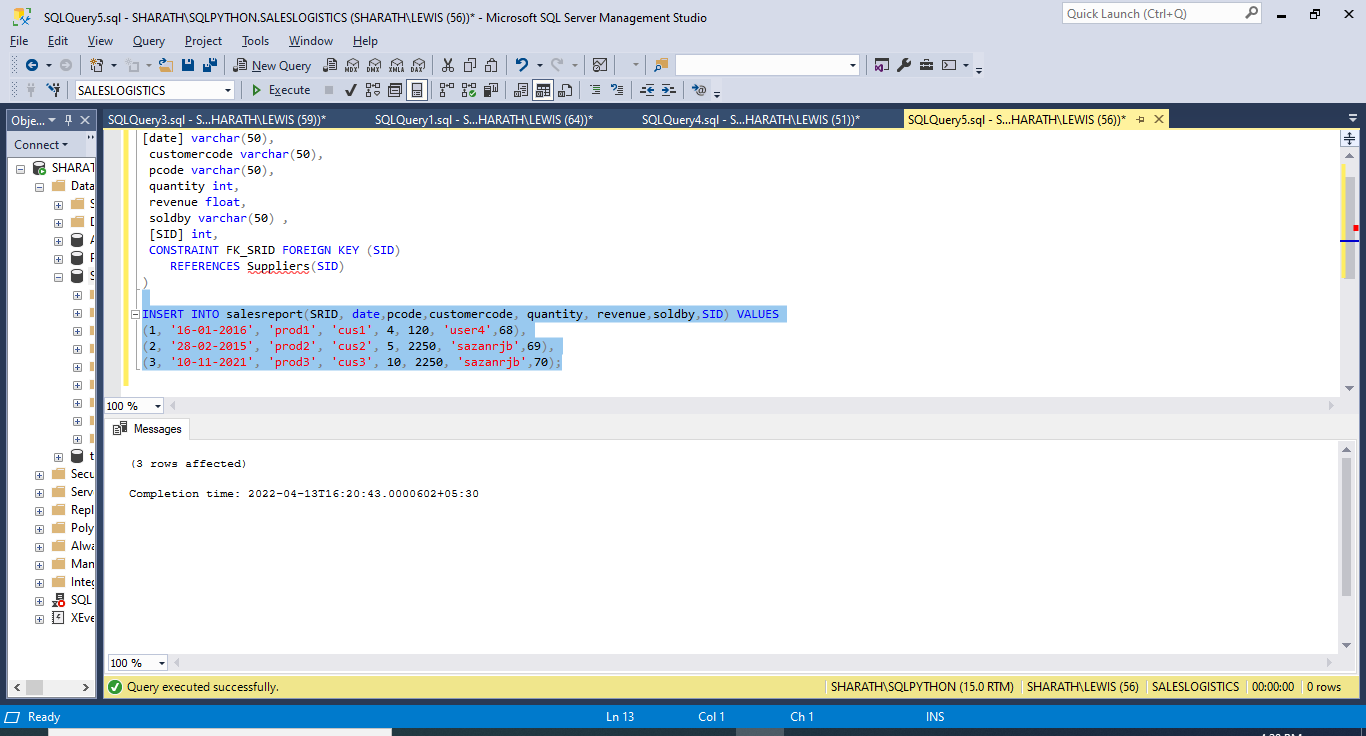


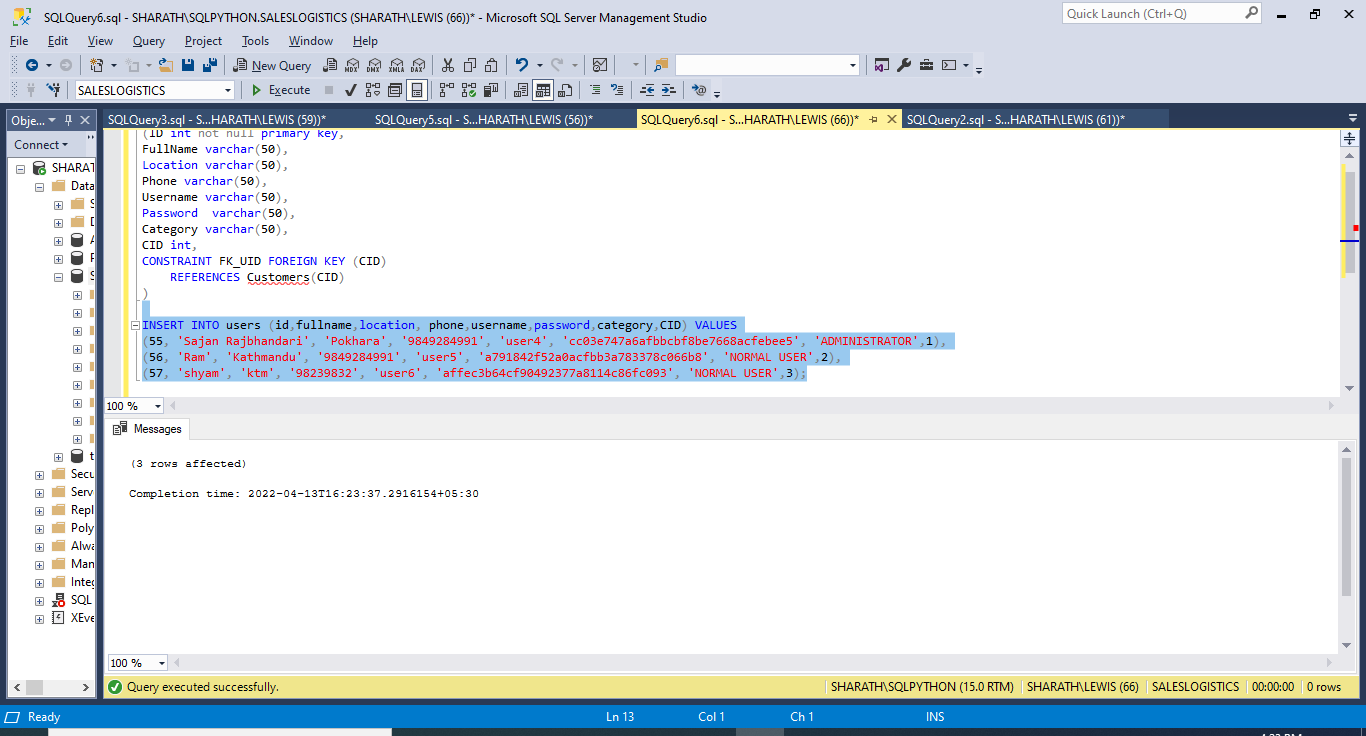
**/\* inserting values into the tables\*/**



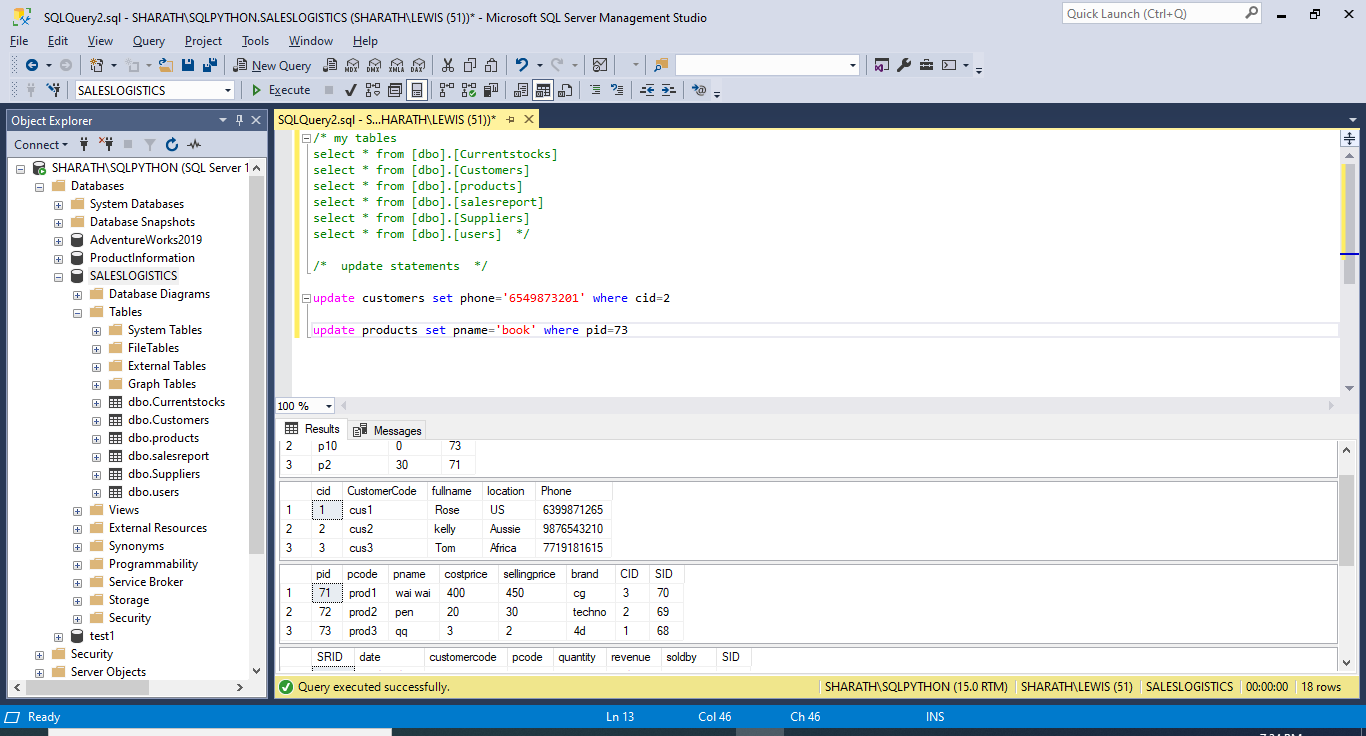


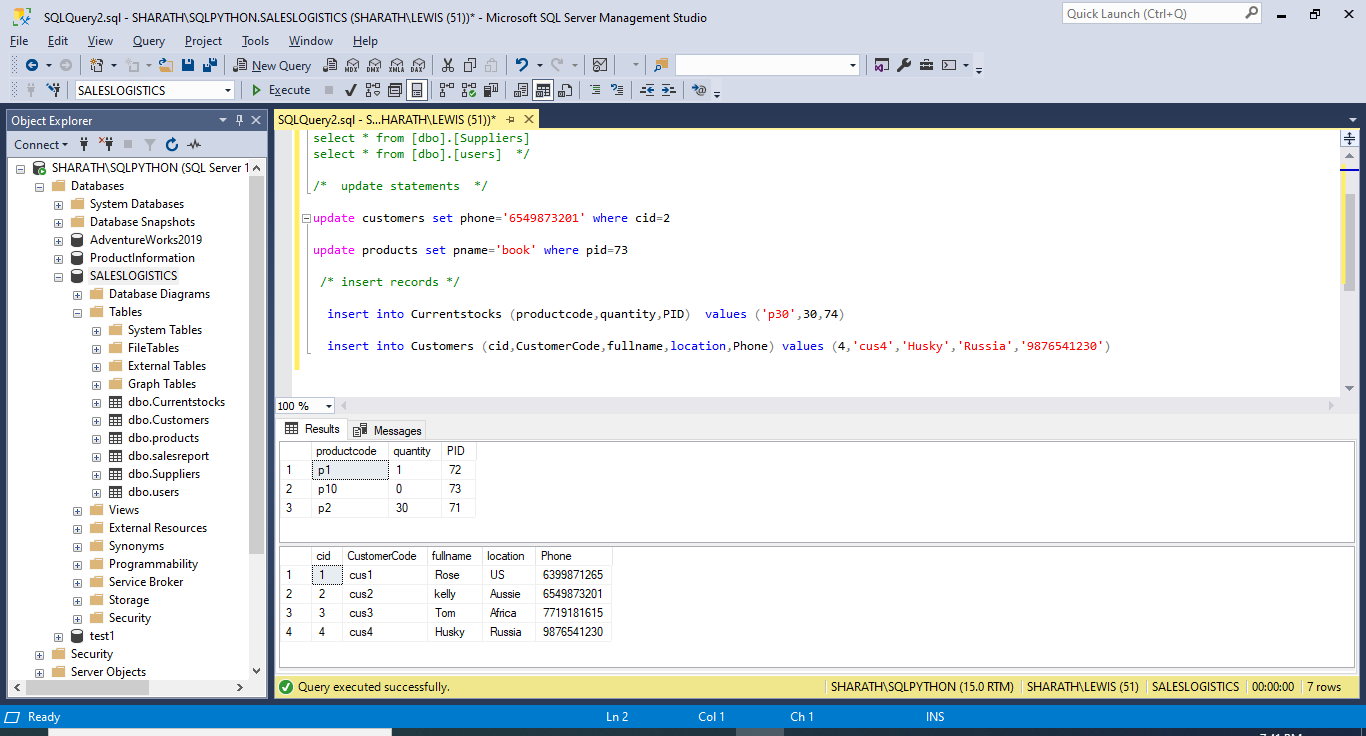


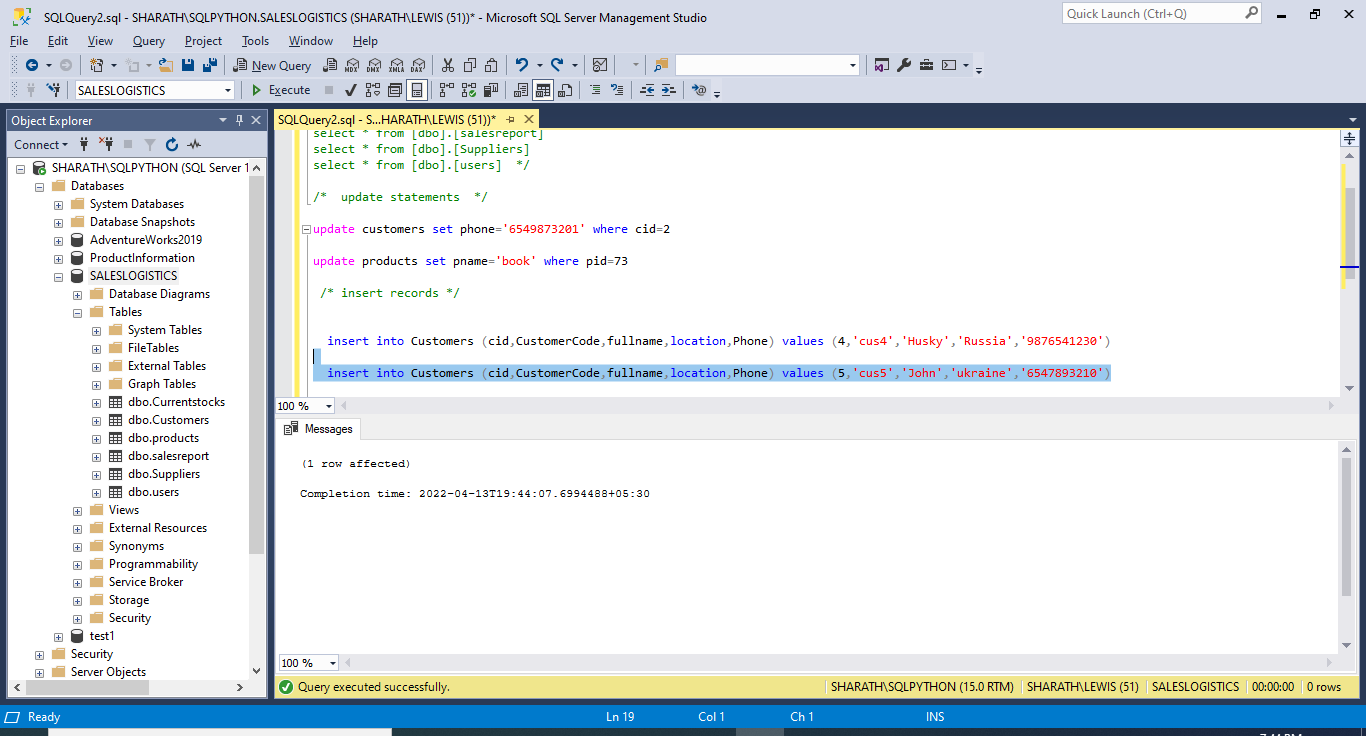


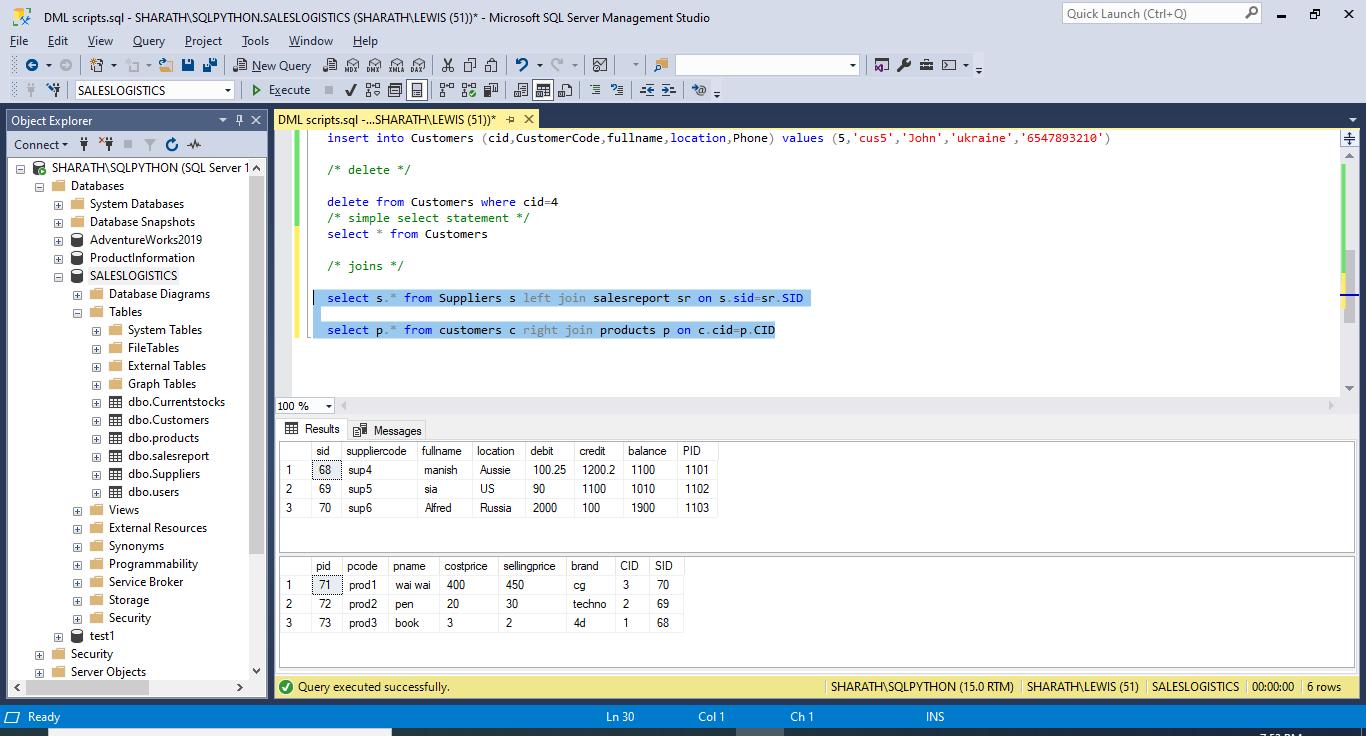


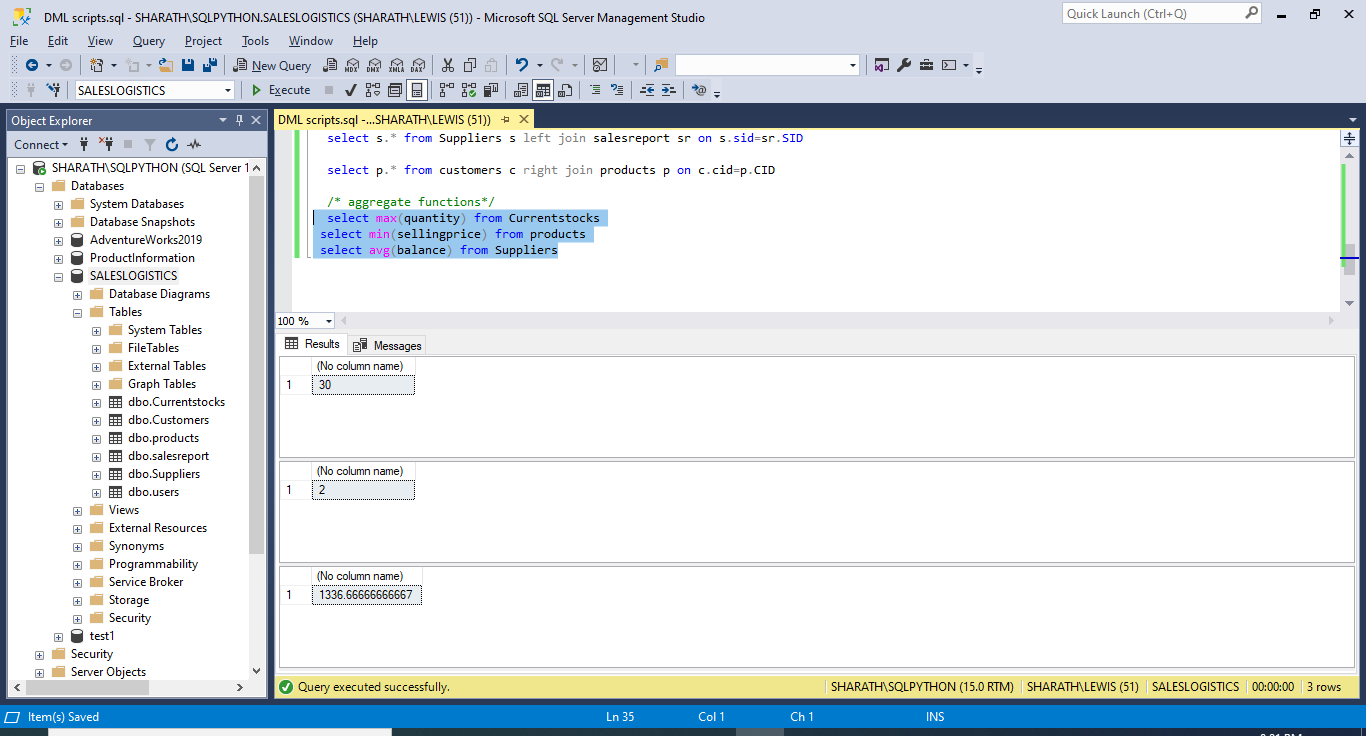
**/\* DML scripts \*/**

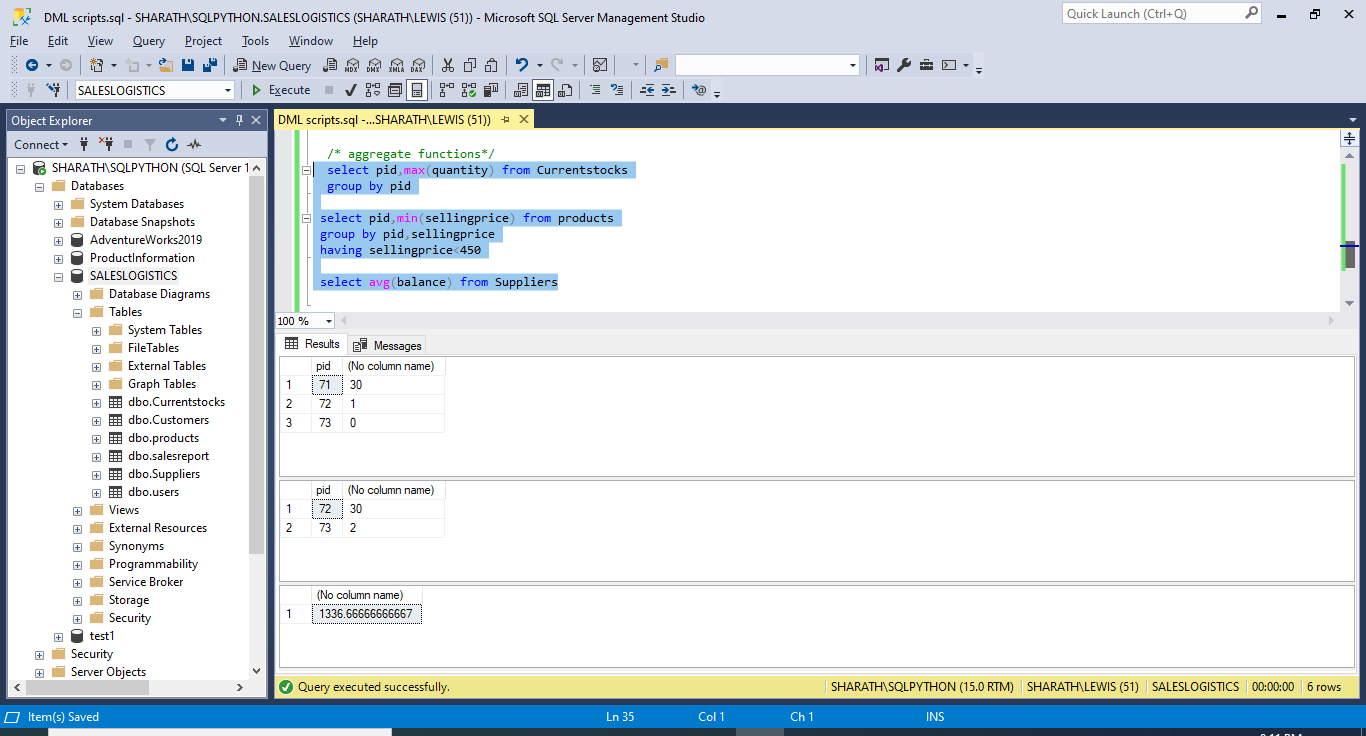


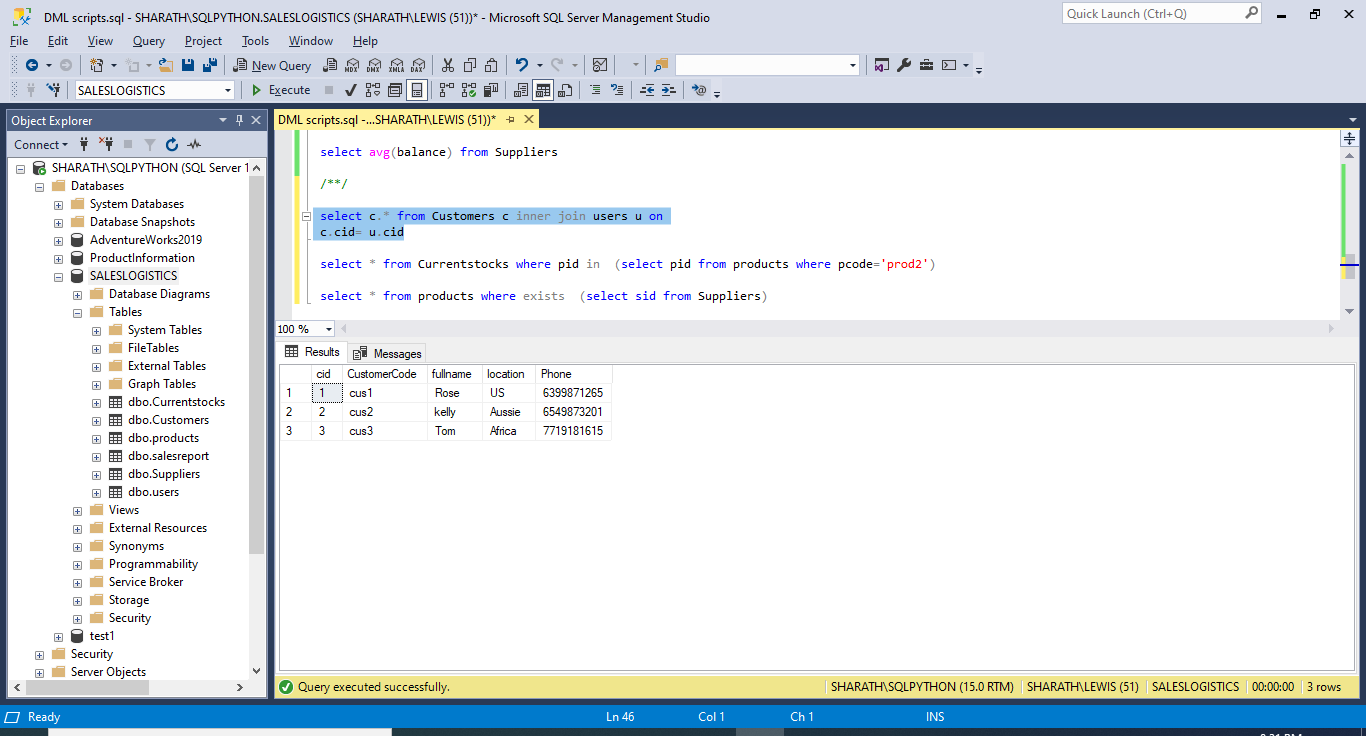


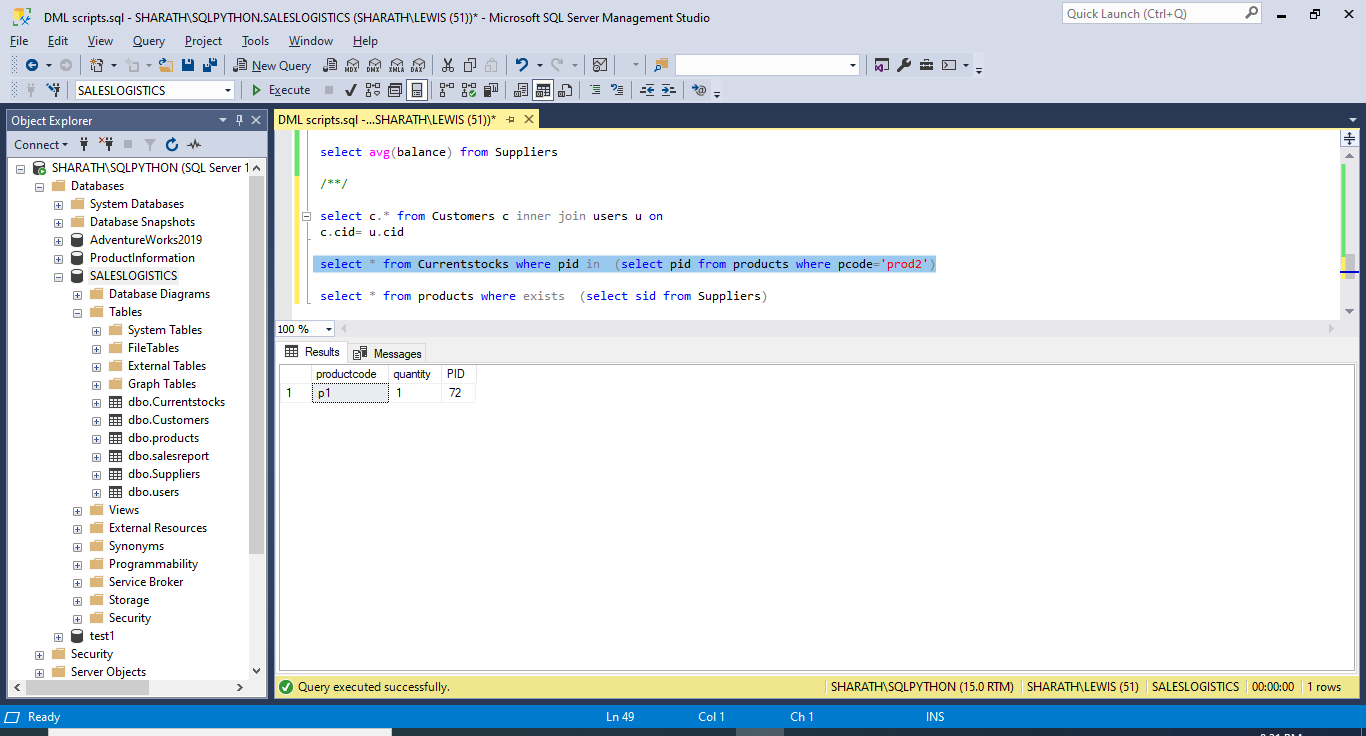


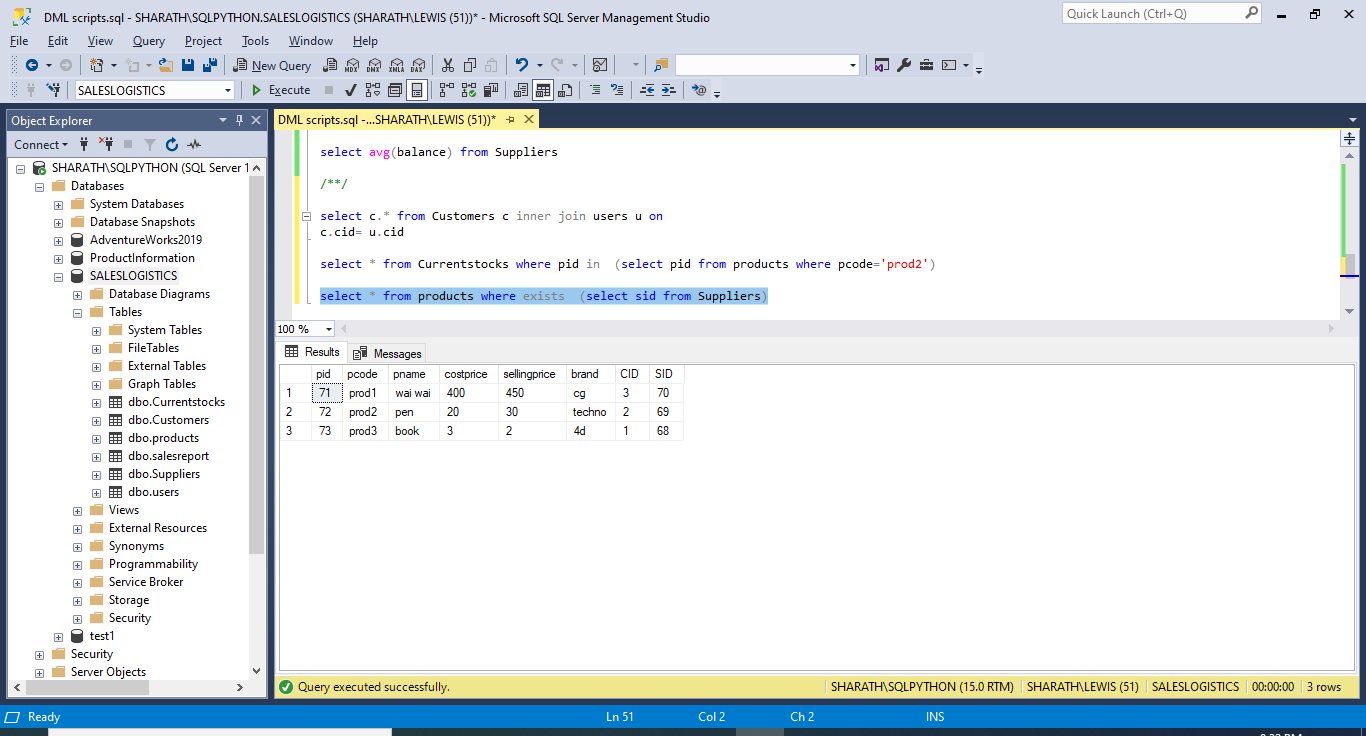












**Part-V**

**INDEXES AND VIEWS:**

Indexes are special lookup tables that the database search engine can use to speed up data retrieval. Simply put, an index is a pointer to data in a table. An index in a database is very similar to an index in the back of a book.An index helps to speed up SELECT queries and WHERE clauses, but it slows down data input, with the UPDATE and the INSERT statements. Indexes can be created or dropped with no effect on the data.

Indexes can be:

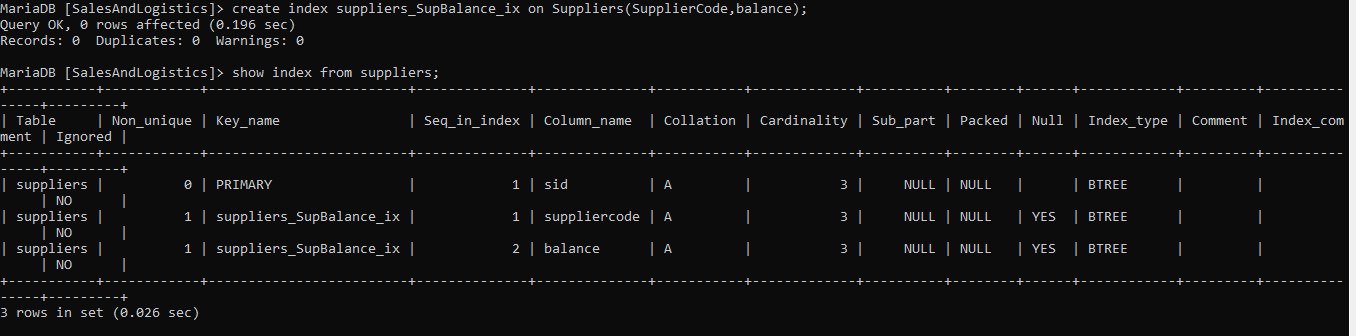
* Single-Column Indexes
* Unique Indexes
* Composite Indexes
* Implicit Indexes

A view is nothing more than a SQL statement that is stored in the database with an associated name. A view is actually a composition of a table in the form of a predefined SQL query.A view can contain all rows of a table or select rows from a table. A view can be created from one or many tables which depends on the written SQL query to create a view.Views, which are a type of virtual tables allow users to do the following −

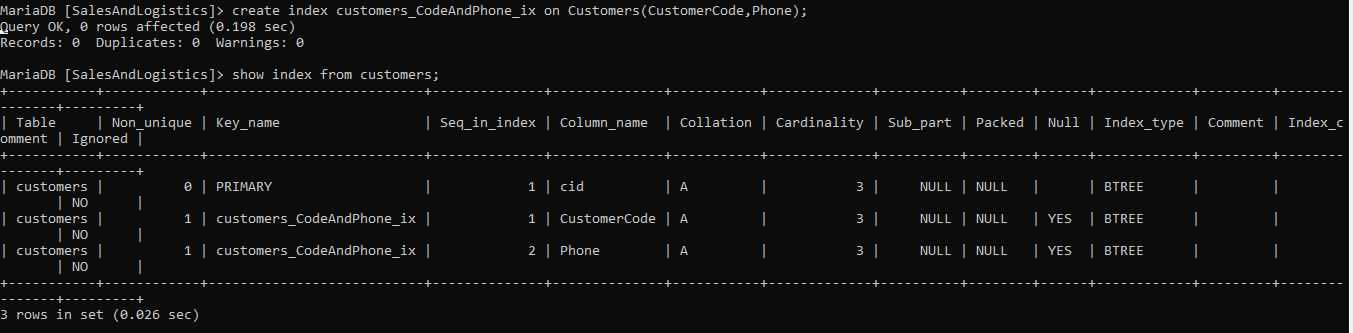
* Structure data in a way that users or classes of users find natural or intuitive.
* Restrict access to the data in such a way that a user can see and (sometimes) modify exactly what they need and no more.
* Summarize data from various tables which can be used to generate reports.

INDEXES:

The indexes I am choosing in my project are on tables Suppliers, Customers. In supplier table I have chosen SupplierCode, Balance because as Sales and Logistics person I want to know the balance with each supplier without any delay. And also these columns contains distinct values on which indexes can be created.



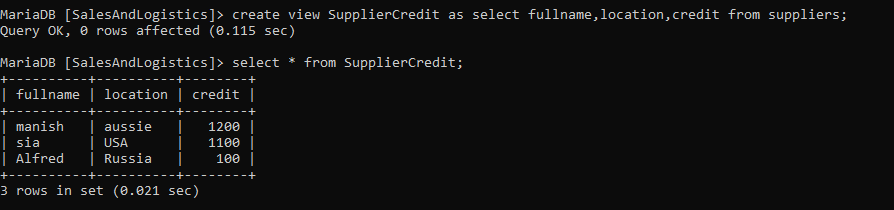
The second index is on customer table. The columns used for creating indexes are ‘CustomerCode’ and ‘Phone’. These columns are used because in the Sales department we always need to be in contact with the customer for business enhancements. And also these columns are updated infrequently.



The SQL Server Query Store feature provides us with insight on query plan choice and performance. It simplifies performance troubleshooting by helping quickly find performance differences caused by query plan changes. Query Store automatically captures a history of queries, plans, and runtime statistics, and retains these for review. It separates data by time windows so we can see database usage patterns and understand when query plan changes happened on the server.

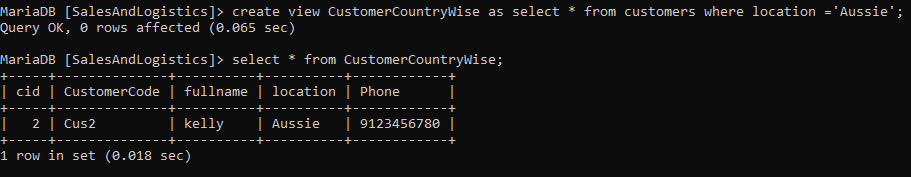
**VIEWS:**

create view SupplierCredit as select fullname,location,credit from suppliers;



The reason behind choosing this view is ‘Currency’. I want to keep track of credit Amount so that from business perspective I can collect the credit amount from the customer in case of late payments/no payments.

create view CustomerCountryWise as select \* from customers where location ='Aussie';



The reason behind choosing second view is ‘Data Independence’. I want to view data country wise so that in case of loss, we can concentrate more on the specific country as shown in the example.