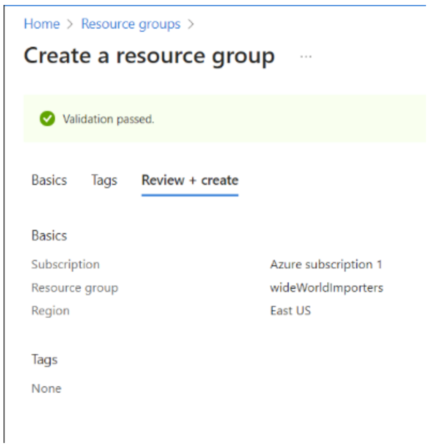
Final Project Documentation

Data Analyst Internship 2024

Stefan Ristikj

This document contains the overview and detailed documentation of the final project for the Data Analyst internship. The goal of the project is to assess my knowledge gained through the past two months, and it encompasses the Microsoft-provided database WideWorldImporters and its detailed analysis. Firstly, the database needs to be thoroughly checked for any inconsistencies, missing values and continuity in SQL Server Management Studio, then according to my area of analysis, I need to take whatever data I think is necessary and upload it to Azure services, whose environment I need to setup before that, then create views over the data that I finally import into PowerBI and create a report for a fictitious customer using various visualizations to make the report story-like and appealing for non-professional BI users.

First off, I checked the data for any anomalies. After making sure none were left (there weren't any significant changes I needed to make), I started preparing the Azure environment for importing the data. I started by creating an Azure resource group (figure 1) to keep all created resources in a centralized environment.


Home > Resource groups >

Create a resource group

Validation passed.

Basics

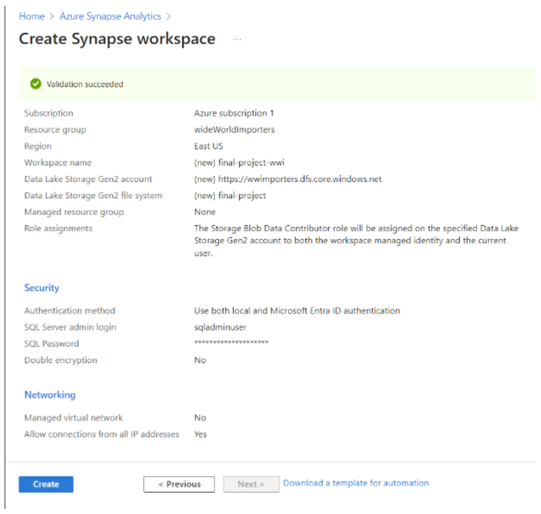
Subscription	Azure subscription 1
Resource group	wideWorldImporters
Region	East US

Tags

None

Figure 1

Then I proceeded to create a Synapse workspace (figure 2) which during initialization created a built-in serverless SQL pool automatically, to be able to work on the project. During the process of creating a workspace, I was prompted to create an Azure Data Lake Storage Gen2 account a file system name and credentials (a username, which I left to the default value of sqladminuser, and a password which I would use later to connect to my local machine).



Home > Azure Synapse Analytics >

Create Synapse workspace

Validation succeeded

Subscription	Azure subscription 1
Resource group	wideWorldImporters
Region	East US
Workspace name	(new) final-project-wwi
Data Lake Storage Gen2 account	(new) https://wwimporters.dfs.core.windows.net
Data Lake Storage Gen2 file system	(new) final-project
Managed resource group	None
Role assignments	The Storage Blob Data Contributor role will be assigned on the specified Data Lake Storage Gen2 account to both the workspace managed identity and the current user.

Security

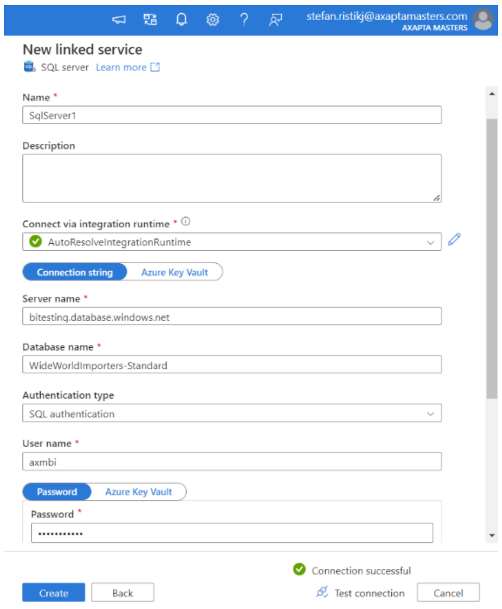
Authentication method	Use both local and Microsoft Entra ID authentication
SQL Server admin login	sqladminuser
SQL Password	********************
Double encryption	No

Networking

Managed virtual network	No
Allow connections from all IP addresses	Yes

Figure 2

After that, the next few steps were to open Synapse Studio to work in the workspace I created previously, then go to the Manage tab and Linked Services, where besides the two automatically created services to the Synapse Analytics workspace and the ADLS Gen2 account, I needed to create a linked service to the SQL server which hosts the WideWorldImporters database, so I did as in figure 3.



stefan.ristikj@axaptamasters.com
AXAPTA MASTERS

New linked service

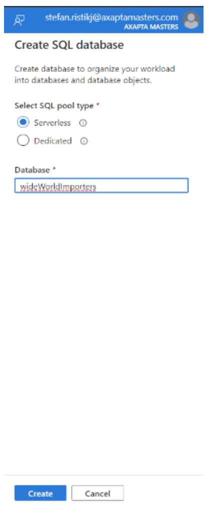
SQL server Learn more

Name *	SqlServer1
Description	
Connect via integration runtime *	AutoResolveIntegrationRuntime
Connection string / Azure Key Vault	Connection string
Server name *	bitesting.database.windows.net
Database name *	WideWorldImporters-Standard
Authentication type	SQL authentication
User name *	axmbi
Password / Azure Key Vault	Password
Password *	***********

Connection successful

Figure 3

Next up, I needed to create a SQL database on the serverless pool, to host the external tables I will create in the next steps (figure 4).



stefan.ristikj@axaptamasters.com
AXAPTA MASTERS

Create SQL database

Create database to organize your workload into databases and database objects.

Select SQL pool type *

[✓ Serverless]
Dedicated

Database *

wideWorldImporters

Figure 4

After that, I decided to create a preliminary schema (figure 5) of the two main schemas in the database part of my project, Warehouse and Sales. This schema contains every table from both schemas plus Application.People because it was needed, contains every column from the tables and my assumption on whether the tables and columns will be necessary for my final model and the fact table.

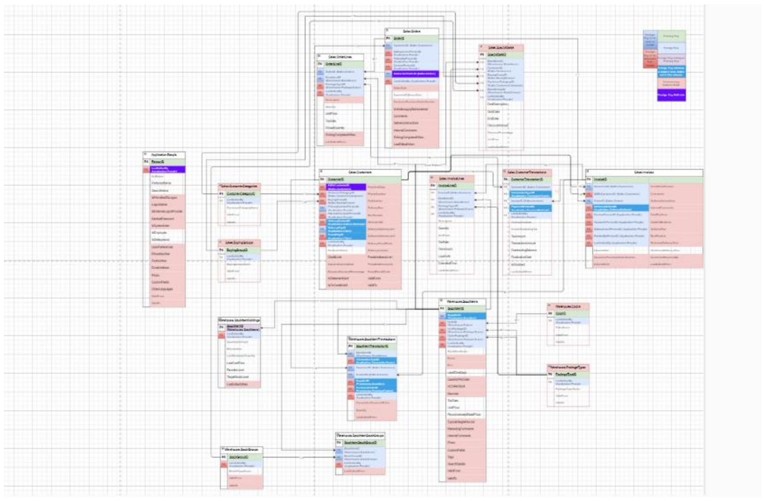
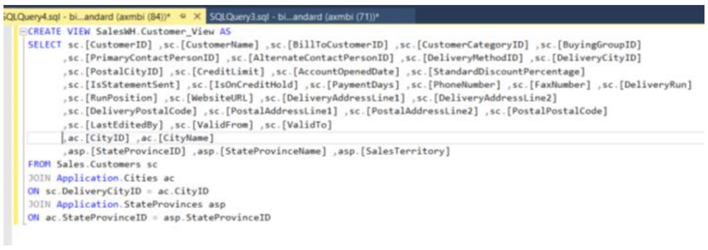


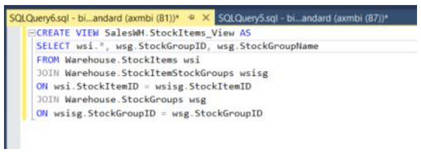
Figure 5

The next step is to create a fact table and some views on the data, which would eliminate the need for importing too many tables. In figures 6, 7 and 8 are the SSMS queries for creating views on the Sales.Customers and the Warehouse.StockItems tables, and for creating the fact table. For the Sales.Customers table, I needed information for the customers' location state-wise, so I joined the Cities and StateProvinces tables from the Application schema instead of taking them as whole tables (I also excluded a column with type geography due to a problem with these columns in Synapse). Same for the StockItems table, and I created the fact table as a joint table between the two tables that I previously identified as tables that contain facts, Sales.Invoices and Warehouse.StockItemTransactions.


SQLQuery4.sql - bi...andard (axmbi (84))*
SQLQuery3.sql - bi...andard (axmbi (71))*

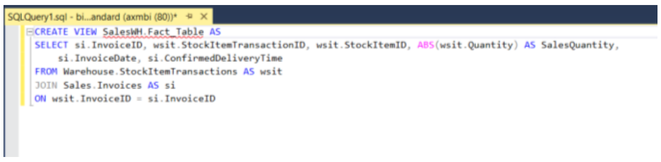
```sql
CREATE VIEW SalesWH.Customer_View AS
SELECT sc.[CustomerID] ,sc.[CustomerName] ,sc.[BillToCustomerID] ,sc.[CustomerCategoryID] ,sc.[BuyingGroupID]
      ,sc.[PrimaryContactPersonID] ,sc.[AlternateContactPersonID] ,sc.[DeliveryMethodID] ,sc.[DeliveryCityID]
      ,sc.[PostalCityID] ,sc.[CreditLimit] ,sc.[AccountOpenedDate] ,sc.[StandardDiscountPercentage]
      ,sc.[IsStatementSent] ,sc.[IsOnCreditHold] ,sc.[PaymentDays] ,sc.[PhoneNumber] ,sc.[FaxNumber] ,sc.[DeliveryRun]
      ,sc.[RunPosition] ,sc.[WebsiteURL] ,sc.[DeliveryAddressLine1] ,sc.[DeliveryAddressLine2]
      ,sc.[DeliveryPostalCode] ,sc.[PostalAddressLine1] ,sc.[PostalAddressLine2] ,sc.[PostalPostalCode]
      ,sc.[LastEditedBy] ,sc.[ValidFrom] ,sc.[ValidTo]
      ,ac.[CityID] ,ac.[CityName]
      ,asp.[StateProvinceID] ,asp.[StateProvinceName] ,asp.[SalesTerritory]
FROM Sales.Customers sc
JOIN Application.Cities ac
ON sc.DeliveryCityID = ac.CityID
JOIN Application.StateProvinces asp
ON ac.StateProvinceID = asp.StateProvinceID
```

Figure 6


SQLQuery6.sql - bi...andard (axmbi (81))*
SQLQuery5.sql - bi...andard (axmbi (87))*

```sql
CREATE VIEW SalesWH.StockItems_View AS
SELECT wsi.*, wsg.StockGroupID, wsg.StockGroupName
FROM Warehouse.StockItems wsi
JOIN Warehouse.StockItemStockGroups wsisg
ON wsi.StockItemID = wsisg.StockItemID
JOIN Warehouse.StockGroups wsg
ON wsisg.StockGroupID = wsg.StockGroupID
```

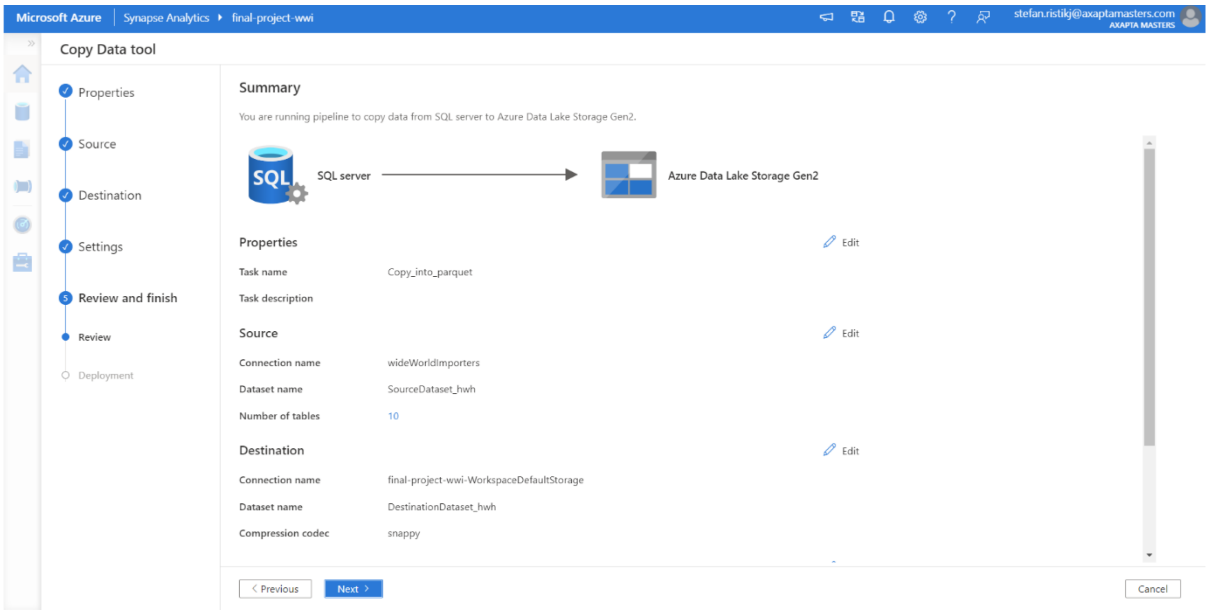
Figure 7



```sql
CREATE VIEW SalesWH.Fact_Table AS
SELECT si.InvoiceID, wsit.StockItemTransactionID, wsit.StockItemID, ABS(wsit.Quantity) AS SalesQuantity,
    si.InvoiceDate, si.ConfirmedDeliveryTime
FROM Warehouse.StockItemTransactions AS wsit
JOIN Sales.Invoices AS si
ON wsit.InvoiceID = si.InvoiceID
```

Figure 8

Next up, I imported the necessary data in Synapse through the Ingest option on the home screen (figure 9). The option is a copy data tool which creates a pipeline automatically, recognizes the tables from the database through the linked service to SQL Server, creates source and destination integration datasets and executes the pipeline. The other options pertaining to the project are file format (parquet, as needed) and the location in which these parquet files should reside. After verifying everything is in order, I executed the pipeline, creating parquet files for every necessary table.



Summary

You are running pipeline to copy data from SQL server to Azure Data Lake Storage Gen2.

Properties

Task name: Copy_into_parquet
Task description:

Source

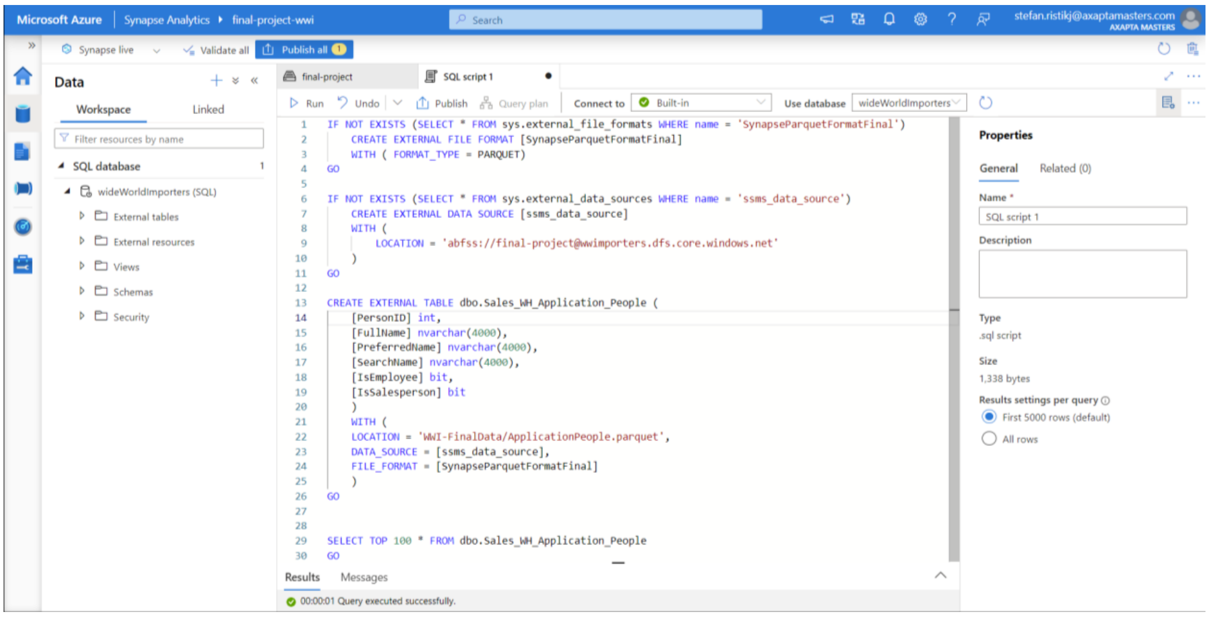
Connection name: wideWorldImporters
Dataset name: SourceDataset_hwh
Number of tables: 10

Destination

Connection name: final-project-wwi-WorkspaceDefaultStorage
Dataset name: DestinationDataset_hwh
Compression codec: snappy

Figure 9

Following this, I proceeded to create the external data source and file format, which will then be used to create the external tables. In figure 10, I give the code for creating the data source and file format, as well as an external table on the ApplicationPeople.parquet file. Every table is created using the same pattern, selecting the columns I need and using the already created data source and file format.



```sql
IF NOT EXISTS (SELECT * FROM sys.external_file_formats WHERE name = 'SynapseParquetFormatFinal')
    CREATE EXTERNAL FILE FORMAT [SynapseParquetFormatFinal]
    WITH ( FORMAT_TYPE = PARQUET)
GO

IF NOT EXISTS (SELECT * FROM sys.external_data_sources WHERE name = 'ssms_data_source')
    CREATE EXTERNAL DATA SOURCE [ssms_data_source]
    WITH (
        LOCATION = 'abfss://final-project@wwimporters.dfs.core.windows.net'
    )
GO

CREATE EXTERNAL TABLE dbo.Sales_WH_Application_People (
    [PersonID] int,
    [FullName] nvarchar(4000),
    [PreferredName] nvarchar(4000),
    [SearchName] nvarchar(4000),
    [IsEmployee] bit,
    [IsSalesperson] bit
    )
    WITH (
    LOCATION = 'WWI-FinalData/ApplicationPeople.parquet',
    DATA_SOURCE = [ssms_data_source],
    FILE_FORMAT = [SynapseParquetFormatFinal]
    )
GO


SELECT TOP 100 * FROM dbo.Sales_WH_Application_People
GO
```

```
00:00:01 Query executed successfully.
```

Figure 10

Then, I imported the external tables into PowerBI using a connection to Azure Synapse Analytics (figure 11) and reviewed the tables to make sure everything is in order, as shown in figure 12.

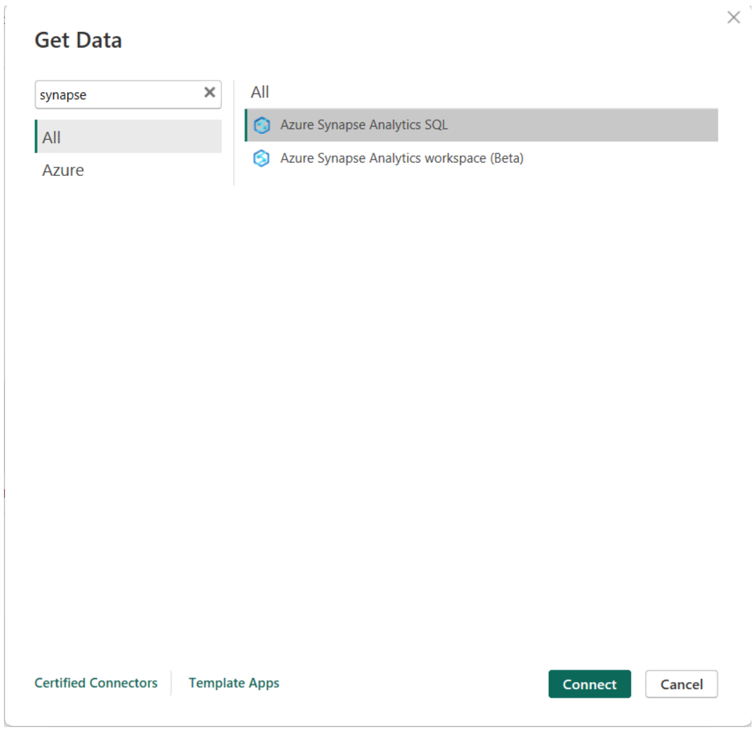
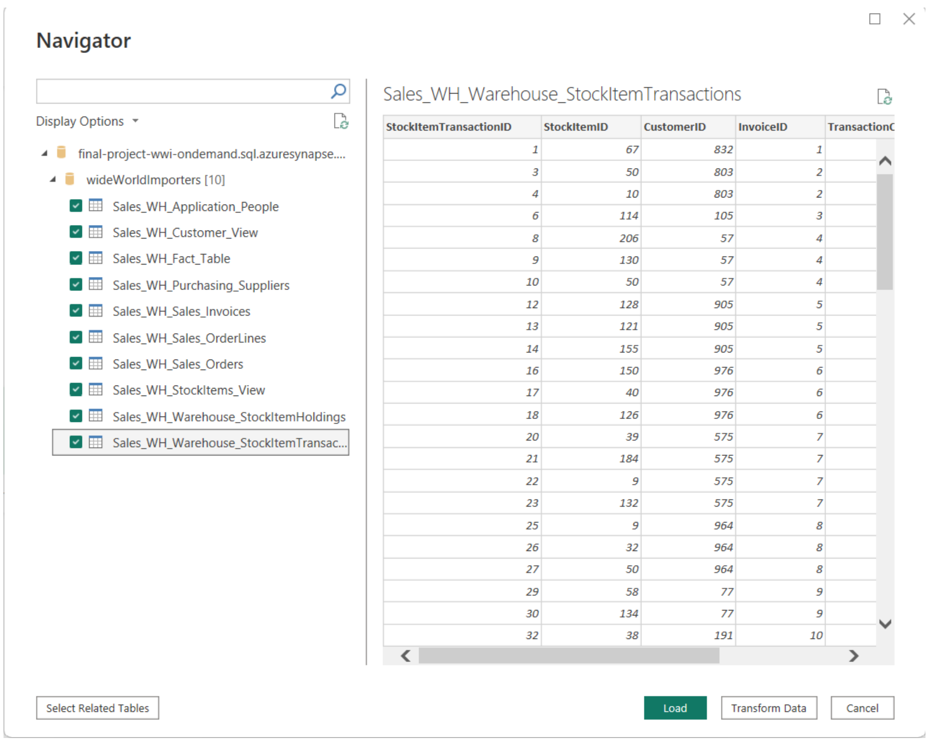


Figure 11



Navigator

Display Options

- final-project-wwi-ondemand.sql.azuresynapse....
  - wideWorldImporters [10]
    - ✓ Sales_WH_Application_People
    - ✓ Sales_WH_Customer_View
    - ✓ Sales_WH_Fact_Table
    - ✓ Sales_WH_Purchasing_Suppliers
    - ✓ Sales_WH_Sales_Invoices
    - ✓ Sales_WH_Sales_OrderLines
    - ✓ Sales_WH_Sales_Orders
    - ✓ Sales_WH_StockItems_View
    - ✓ Sales_WH_Warehouse_StockItemHoldings
    - ✓ Sales_WH_Warehouse_StockItemTransac...

Sales_WH_Warehouse_StockItemTransactions

StockItemTransactionID	StockItemID	CustomerID	InvoiceID	TransactionO
1	67	832	1	
3	50	803	2	
4	10	803	2	
6	114	105	3	
8	206	57	4	
9	130	57	4	
10	50	57	4	
12	128	905	5	
13	121	905	5	
14	155	905	5	
16	150	976	6	
17	40	976	6	
18	126	976	6	
20	39	575	7	
21	184	575	7	
22	9	575	7	
23	132	575	7	
25	9	964	8	
26	32	964	8	
27	50	964	8	
29	58	77	9	
30	134	77	9	
32	38	191	10	

Figure 12

Last but not least, I reevaluated the schema and created a final one (figure 13) that most closely represents the data model I used to create the report, as shown in my presentation.

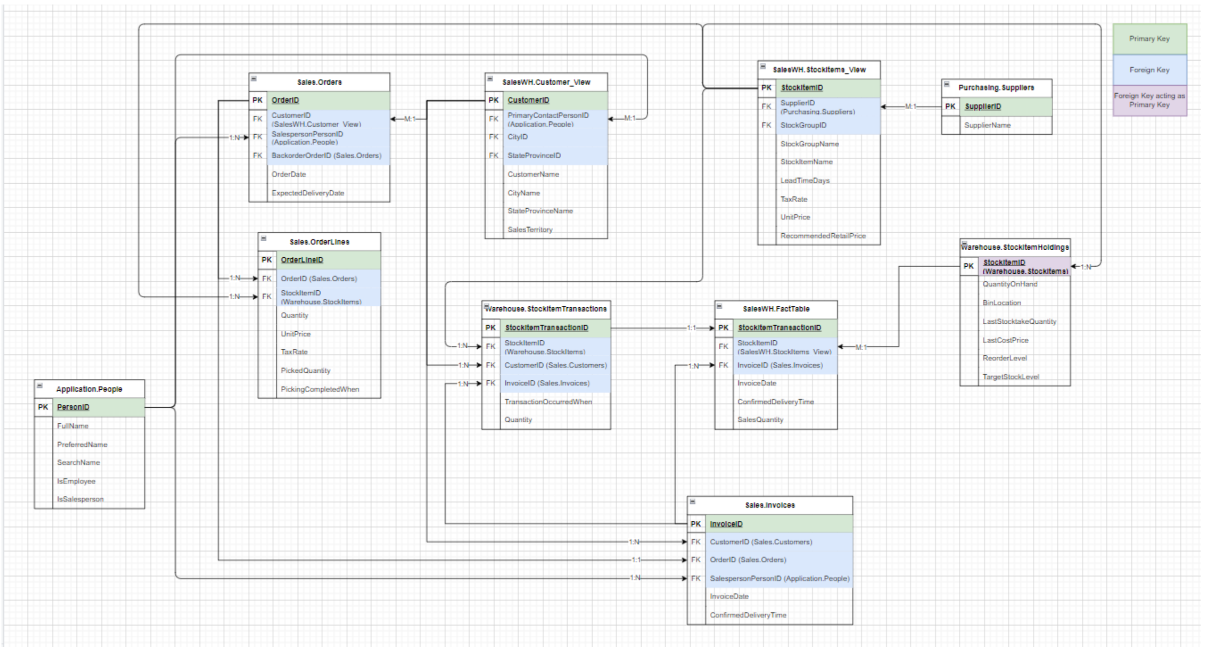


Figure 13