Hybrid Database System

Implementation of a Relational and an XML Database for Overdrive Car Dealers

Student:

Vinit Sawant (10527412)

Kumuditha Athukorala (10539070)

Srikanth Shilesh Pasam (10387794)

Teacher:

Dr. Shazia A Afzal

Course:

Advanced Databases (B9IS100)

Table of Contents

[Index of Figures 3](#_Toc27506547)

[Introduction 4](#_Toc27506548)

[Database 4](#_Toc27506549)

[Database Management System 4](#_Toc27506550)

[Microsoft SQL 4](#_Toc27506551)

[Hybrid Database 5](#_Toc27506552)

[Relational Database 5](#_Toc27506553)

[XML 6](#_Toc27506554)

[Scope of the System 6](#_Toc27506555)

[Business Requirements 7](#_Toc27506556)

[Business Rules 8](#_Toc27506557)

[Entity-Relationship Diagram 10](#_Toc27506558)

[Relational Schema in 3NF 10](#_Toc27506559)

[Implementation in SQL Server 12](#_Toc27506560)

[Tables with Data Diagram 12](#_Toc27506561)

[Stored Procedures 13](#_Toc27506562)

[Triggers 18](#_Toc27506563)

[Views 22](#_Toc27506564)

[Innovation 23](#_Toc27506565)

[Individual Contribution 25](#_Toc27506566)

[Vinit Sawant 25](#_Toc27506567)

[Kumuditha Athukorala 26](#_Toc27506568)

[Srikanth Shilesh Pasam 27](#_Toc27506569)

[Bibliography 29](#_Toc27506570)

[Appendix 1 30](#_Toc27506571)

[Appendix 2 34](#_Toc27506572)

[Appendix 3 39](#_Toc27506573)

[Appendix 3 56](#_Toc27506574)

[Appendix 4 58](#_Toc27506575)

# Index of Figures

[Figure 1 Unique Customer ID 7](#_Toc27505851)

[Figure 2 Unique Employee ID 7](#_Toc27505852)

[Figure 3 Incentive Trigger 8](#_Toc27505853)

[Figure 4 Log Table 8](#_Toc27505854)

[Figure 5 ER Diagram 9](#_Toc27505855)

[Figure 6 Relational Schema 10](#_Toc27505856)

[Figure 7 Data Diagram 11](#_Toc27505857)

[Figure 8 Stored Procedure 1 13](#_Toc27505858)

[Figure 9 Stored Procedure 2 13](#_Toc27505859)

[Figure 10 Stored Procedure 3 14](#_Toc27505860)

[Figure 11 Stored Procedure 4.1 15](#_Toc27505861)

[Figure 12 Stored Procedure 4.2 15](#_Toc27505862)

[Figure 13 Stored Procedure 5 16](#_Toc27505863)

[Figure 14 Stored Procedure 6 16](#_Toc27505864)

[Figure 15 Stored Procedure 7 17](#_Toc27505865)

[Figure 16 Trigger 1.1 18](#_Toc27505866)

[Figure 17 Trigger 1.2 18](#_Toc27505867)

[Figure 18 Trigger 1.3 19](#_Toc27505868)

[Figure 19 Trigger 2.1 19](#_Toc27505869)

[Figure 20 Trigger 2.2 20](#_Toc27505870)

[Figure 21 Trigger 2.3 20](#_Toc27505871)

[Figure 22 View 1 21](#_Toc27505872)

[Figure 23 View 2 22](#_Toc27505873)

[Figure 24 Azure DB Overview 24](#_Toc27505874)

[Figure 25 Azure DB Table View 25](#_Toc27505875)

# Introduction

The Hybrid Database System report explains the working of a Database system for Overdrive car dealership. The Overdrive car dealership is a medium scale business which procures cars from various manufacturers based on customer orders. It has business partnerships with five different car manufacturers currently and plans on getting more on board. The company has clear and structured hierarchy with various departments and levels within each. There is also an incentive system in place to motivate the sales associate’s performance. Whenever a customer places an order for a car, the sales associate raises an order request within the system. This request gets forwarded to the concerned department who then places an order for the same with the manufacturer. Once the order is completed successfully, the sales associate responsible for that sale will get an incentive. All this business functionalities requires a centralized database management system in order to function efficiently.

# Database

A database is a collection of information that is organized. This data can be accessed, managed and updated as per the dealership requirements. The data is indexed which makes it easier to find relevant information through queries (Rouse, no date).

## Database Management System

The data in the database is managed by a Database Management System (DBMS). A DBMS serves as an interface between the database and the Overdrive car dealership allowing to create, manipulate and store the data. DBMS is basically a software that facilitates oversight and control of databases. Few of the most popular DBMS available are MySQL, Microsoft SQL Server, Microsoft Access, Oracle Database and dBASE (*What is a database?*, no date).

## Microsoft SQL

Overdrive uses Microsoft SQL Server to create a Hybrid database. The SQL Server is a relational database system from Microsoft. Apart from storing data it also comprises of management system. There are numerous business applications of Microsoft SQL. Overdrive uses MS SQL Server to hold sensitive user information like personal details, purchase logs, manufacture details and other confidential information. The system also allows for sharing data within the dealership with increased reliability. The SQL server is used to increase speed with which data is processed allowing large operations to be performed with ease (*What is Microsoft SQL Server and What is it Used For?*, 2017).

# Hybrid Database

The SQL Server used by Overdrive stores the data in the form of a Hybrid database. The database consists of Relational and XML data types.

## Relational Database

A Relational database stores data in the form of tables. The database provides access to datapoints that are related to one another. Each row in the table is a record with a unique ID called the ‘Key’. The columns hold the data attributes. The relational databases depict the logical data structure which is different from the physical storage structure. Relational databases prioritize data accessibility and accuracy. This is accomplished in the form of data integrity rules. An example for this is where the relational database does not allow duplicate rows to be entered in the table. This helps prevent erroneous information from being stored in the database. The major application of Relational Database is in place where data points relating to each other must be managed in a secure, rules-based and consistent way (*What is a relational database?*, no date).

Overdrive chose Relation Database as one of its Hybrid database system components because of the following criteria:

* Data Accuracy Requirements – The business operates with data related to customers, car specifications, inventory management and salaries of employees. All this requires accuracy in data management and avoid any kind of duplication.
* Scalability – Relational databases have good scalability in terms of anticipated growth and ability to produce mirrored database copies. These factors enable easy expansion of the Overdrive business in future if needed.
* Concurrency – Overdrive dealers have multiple departments within the organization ranging from procurement, sales, finance and HR. All these departments need to have access to the database simultaneously. This centralized management of the database allows for easy and efficient functioning of the business. Sales agents can place customer order requests, procurement department can order for the required vehicles from the manufacture, HR can recruit new sales agent and finance team handles the salaries of the employees along with their incentives based on the employee’s individual sales record. Relational databases make this concurrent access and management of the database possible.
* Performance and Reliability Needs – The reliable performance of a system is a basic functionality expected out of any system. Overdrive dealers are highly dependent on the database management system for almost all the business operations. In this regard relational database query response performance plays a crucial role.

## XML

XML database is used for storing large volumes of data. It is a secure and persistent software system. Data in XML is queried using XQuery. XML’s functionality is highlighted in situations where the data requirements structure varies within the same element.

Overdrive uses Native XML database as a part of its hybrid database. This type of database is based on the container format. This form of hybrid database allows for certain values of XML data to be stored in relational columns while the rest can be stored in an XML column. The hybrid database yields better performance as we have better control over the indexes created in the relational columns and locking its characteristics.

# Scope of the System

1. Overdrive car dealership employs a Hybrid Database of Relational and XML within a MS SQL Server database management system.
2. The database consists of 8 tables. Each table consist of various range of attributes like ID’s of manufacturers, customers, employees, orders, manufacture details, customer details, order dates, name and addresses to name a few.
3. The car dealer stores the data about the manufacturers along with the models available within each in the manufacture and car model tables respectively. Based on this information, current market trends and estimations, certain cars are procured and stationed within the warehouse in advance.
4. The order details are tracked using the manufacture order table.
5. The car variant ordered from the manufacturer uses XML data type to enable flexibility in the details of the variant stored. As the parameters of the variant details can vary depending on the type of the car, using XML here has more relevance.
6. The list of procured cars is maintained in the inventory table which highlights all the cars available and sold by Overdrive. While this shows the backend operation of the business, on front end, the sales employees will invest the prospective customers.
7. Once a customer places an order, their details are recorded into the customer table. The customer address details are saved as an XML data type which can be updated.
8. The customer order details get recorded into the customer order table.
9. If the order is completed successfully then the sales employee will receive an incentive based on their base salary and the record is stored in the incentive table. This also helps track the performance of the employees.
10. The employee details are kept within the employee table.
11. The details of the order invoices from the manufacturer and the customer along with the mode of payment is not included in the scope of this system.
12. The profits are generated and displayed using views. This is based on the price purchased from the manufacturer and the price at which it is sold to the customer.

# Business Requirements

1. Details of all the orders placed by the customer.
2. Tracking the performance of the employees based on the cars sold.
3. 10% incentive on the base salary should be provided to the employees for each car sold.
4. All the customer order details will be stored as an XML.
5. Generating a list of all cars belonging to a particular car type.
6. Editing the employee address whenever it is required.
7. Generating a list of cars based on the seating capacity.

# Business Rules

1. Every customer should have a unique customer ID (PK constraint) in the format ‘OVDC1000000’.

A screenshot of a cell phone

Description automatically generated

Figure 1 Unique Customer ID

1. Every employee should have a unique employee ID (PK constraint) in the format ‘OVDE1000’.

A screenshot of a cell phone

Description automatically generated

Figure 2 Unique Employee ID

1. A customer order entry should have a unique order ID (PK) and should have a customer ID as well an employee ID associated with it (FK constraints).
2. A manufacturer order entry should have a unique order ID (PK) and should have car model ID as a foreign key.
3. An inventory data will have a unique inventory ID, status as ‘SOLD’ or ‘AVAILABLE’, customer order ID can be null in case of ‘AVAILABLE’ status.
4. Incentive value for the employee should be 10% on his basic salary for every car sold.

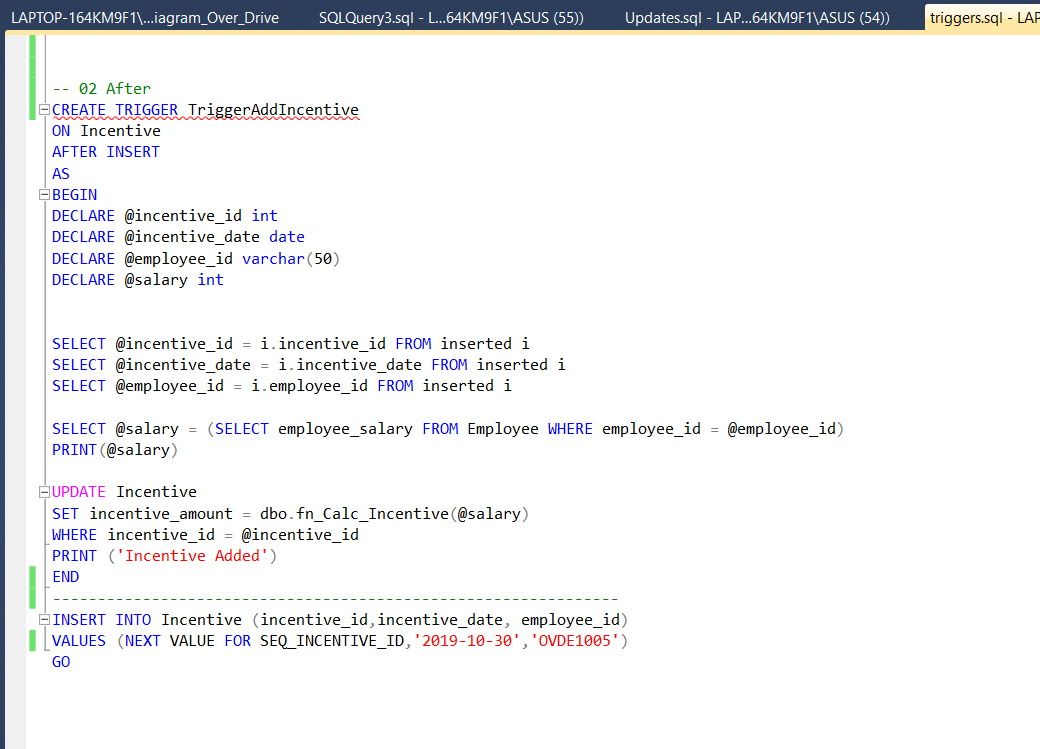
****

Figure 3 Incentive Trigger

1. Maintaining a log table for the data inserted into the customer order table.

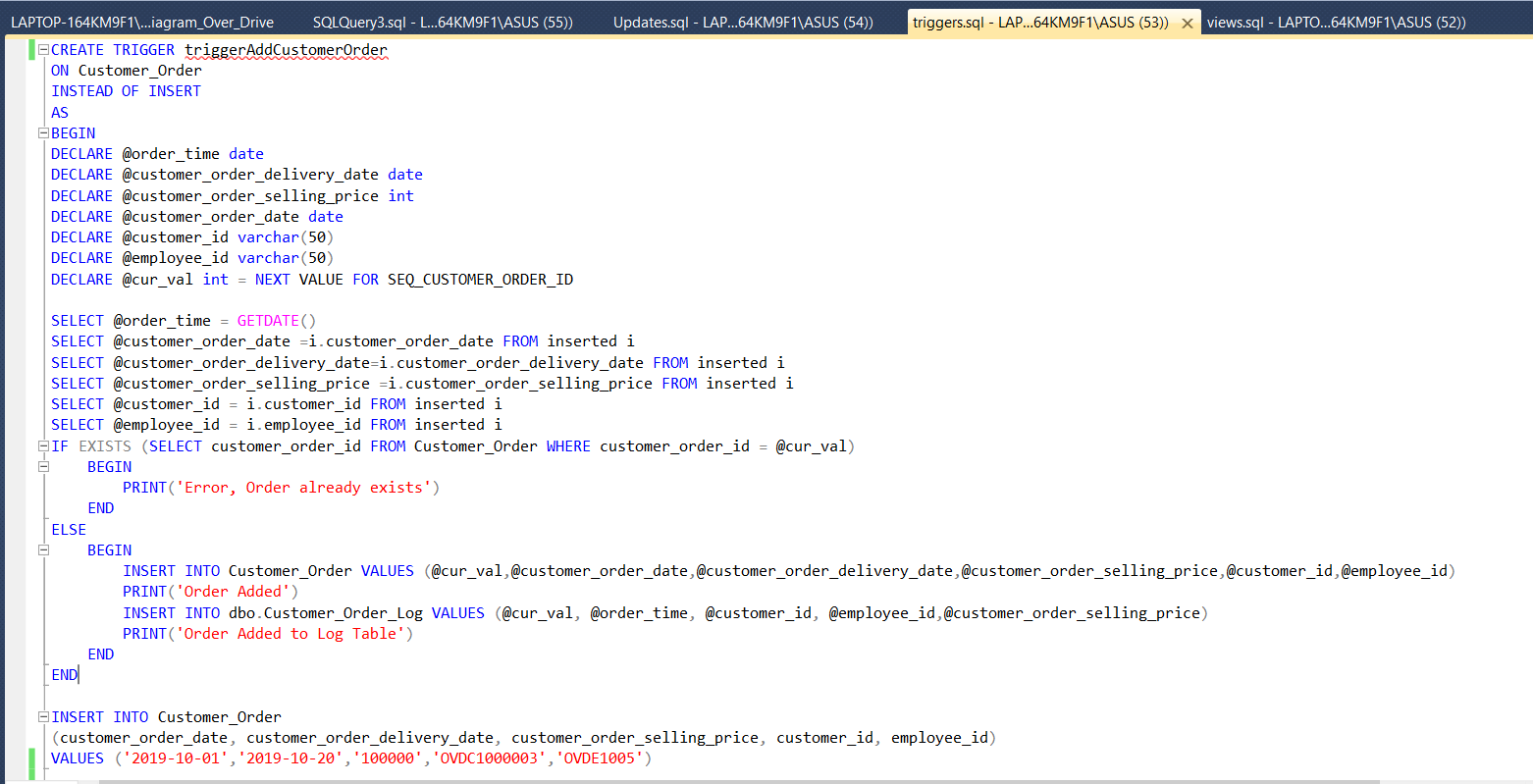


Figure 4 Log Table

# Entity-Relationship Diagram

A picture containing screenshot

Description automatically generated

Figure 5 ER Diagram

# Relational Schema in 3NF

3NF is Normal Form used to normalize a database design. This helps reduce duplication of data and improves referential integrity. This is done by making sure that the entity is in second normal form and no non-prime attribute is transitively dependent on any key. By having the Overdrives relational schema in 3NF we have achieved the following:

1. Eliminated undesirable data
2. Reduced need for restructuring
3. Made the data model more informative
4. Made the data model neutral to different kinds of statistics

The figure below shows the Relational Schema for Overdrive dealership in 3NF:

A picture containing screenshot, text

Description automatically generated

Figure 6 Relational Schema

The schema shows the hybrid database structure. While most of the attributes use relational structure, three of them are in the XML form. They are:

1. Customer address
2. Employee address
3. Car variant

The main reason behind using XML for these specific attributes is that they are semi structured data. Address fields can vary widely based on the customers location and type of residence like a building which will have flat number and building name while individual residences won’t have these fields. The other attribute is the car variant. This can vary depending on the category of the car. For example, a sports variant will have information regarding the car’s acceleration while a family van will have information regarding the seating capacity. Because of these variations in data, XML will be effective over relation data structure here. This results in Overdrives database becoming a hybrid database.

# Implementation in SQL Server

## Tables with Data Diagram

The data diagram for Overdrive car dealership is show in figure below:

A screenshot of a social media post

Description automatically generated

Figure 7 Data Diagram

According of the business requirements of OverDrive database it contains the data of vehicle manufactures, car models, customer, employee and inventory. Also, it is capable to store records of manufacturer order details, customer order details and employee incentive. From the car dealer’s perspective, it is mandatory to keep manufacturer records, car models and manufacturer order details. Manufacturer table consists all the car manufactures details and implemented manufacturer\_id to identify a particular manufacturer uniquely. Car dealers keep the records of car models, there for implemented table was Car\_Model which has car\_model\_id as the primary key. The relationship between the Manufacturer and Car\_Model is one to many and manufacturer\_id is a foreign key of Car\_Model table. Therefore, foreign key constraints have been added to the Car\_Model table. To keep the records of car dealer’s order records, Manufacturer\_Order table is implemented and the relationship between the Car\_Model table and Manufacturer\_Order is one to many because car dealer can place many orders based on one car model. Therefore car\_model\_id which is the primary key of the Car\_Model table presented in Manufacturer\_Order table as foreign key. Foreign key constraints have been added to the Manufacturer\_Order table. To represent the actual status of the cars which car dealer has purchased and sold to the customers. The relationship between the Inventory and the Manufacturer\_Order is one to and manufacturer\_order\_id and foreign key constrains has been added to the Inventory table. Based on the inventory records, car dealer is proceeded the customer orders. Assuming the one customer is purchasing one car at a time, implemented relationship was one to one in between the Customer\_Order table and Inventory table. To capture the customer orders based on car dealer inventory foreign key constraint has been added to the Inventory table and customer\_order\_id is used as foreign key in Inventory table. A customer can place many orders therefore customer\_id which is the primary key of the Customer table added as foreign key to the Customer\_Order table. Foreign key constraint has been added to the Customer\_Order table. Employee table which holds the employee details has employee\_id as primary key and an employee is involved to a particular customer order. But there can be zero or many customer order records related to an employee therefore Employee and Customer\_Order table has zero to many relationships. Employee\_id is foreign key of the Customer\_Order table and foreign key constrain has been added to the Customer\_Order table. Based on the number of orders employees are involved, car dealer is giving incentives. Incentive amount is 10% of the employee’s salary. Employee table and Incentive table has one to many relationships therefore employee\_id as primary key is implemented as foreign key in Incentive table.

## Stored Procedures

Stored procedures are used to group one or more logical SQL statements into one unit and are stored as objects in the database server. Usually stored procedures are used when we want a certain code to be reused again and again. One can also pass values to the stored procedure to make it more useful and to get more detailed outputs from the same. Stored procedures resist SQL injection. Besides stored procedures will reduce network traffic and increase performance (‘SQL Server Stored Procedures Tutorial’, no date).

For the Over Drive project, we have implemented our business requirements in terms of stored procedures.

Following are the seven business requirements that we had analyzed earlier and the screenshots of the same:

1. Use of JOIN between two or more tables as required –

The stored procedure takes the input as a Customer Id and displays the details of all orders placed by the customer as well as the details of the customer. The output also displays the employee id associated who was involved in the selling of that car. This will be helpful for the management team to determine various entities involved in a successfully placed order.

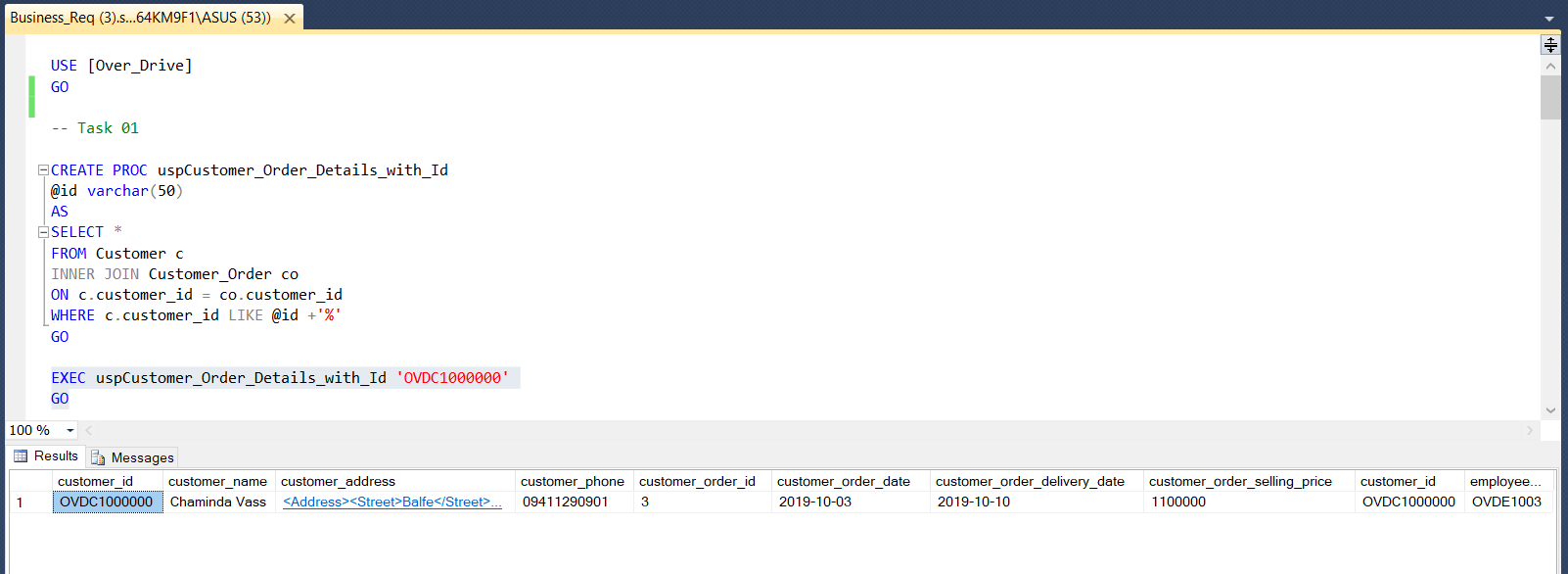


Figure 8 Stored Procedure 1

1. Use of GROUP BY with HAVING –

The stored procedure will take a cut-off price as input parameter and will display the list of all employees that were successful in selling the cars with cumulative prices above that cut-off price. So if the cut-off value is passed as 100000 then it will show all employees that have sold cars worth more than 100000. This will be helpful for the management of Over Drive to monitor the performance of the various employees.

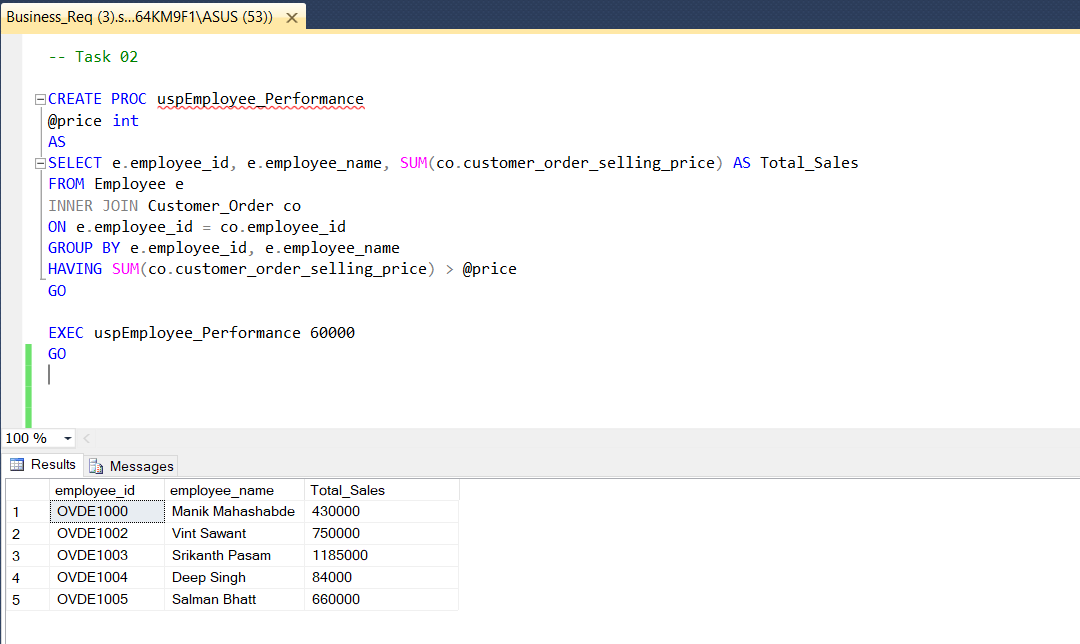


Figure 9 Stored Procedure 2

1. Use of custom SQL functions –

The stored procedure will have a custom SQL function that will calculate the incentive value (bonus) by inputting the salary of the employee. The incentive value will be returned and will be 10 % of the employee basic salary. This function will be used when giving incentives to employees after they are successful in completing an order from a customer. Such incentives are given in-order to increase the morale of employees.

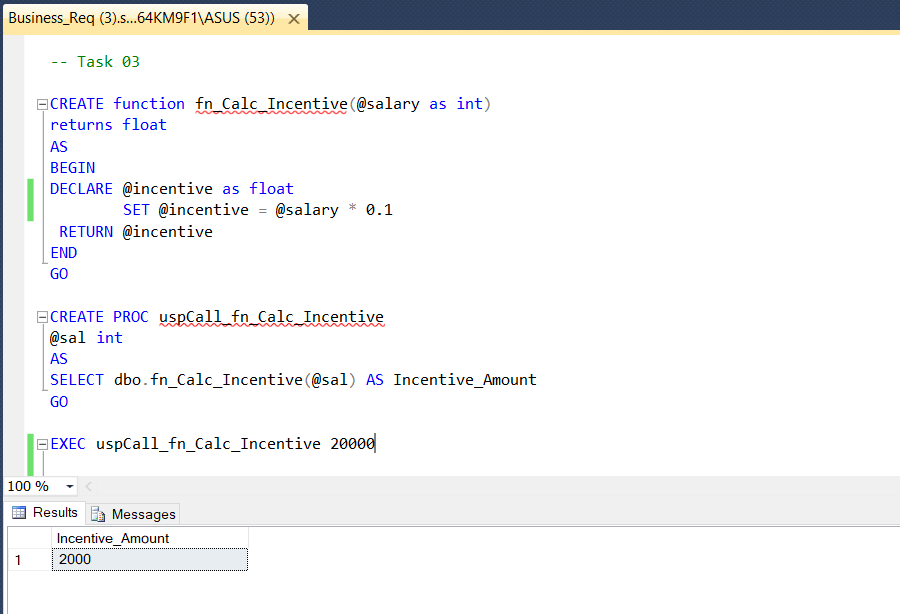


Figure 10 Stored Procedure 3

1. Developing XML with appropriate elements using relational fields –

The stored procedure will develop an XML from the relational fields. The resultant XML will have details of customer id, car model and car manufacturer, customer selling price and manufacturer order price. It will have an overall picture of the order fetched from various tables. This might be used to send the data in the form of XML to any other entities that might require it for audit purpose.

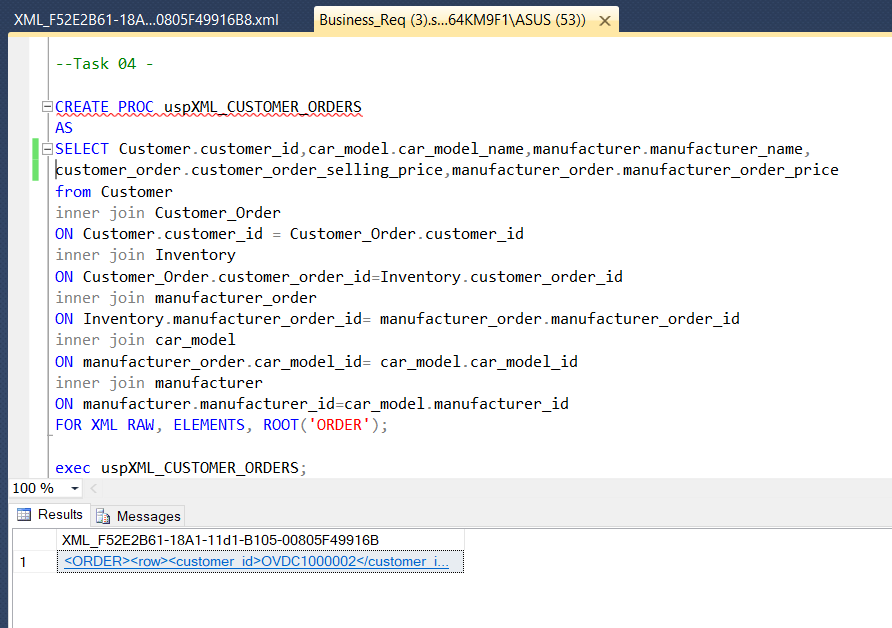


Figure 11 Stored Procedure 4.1

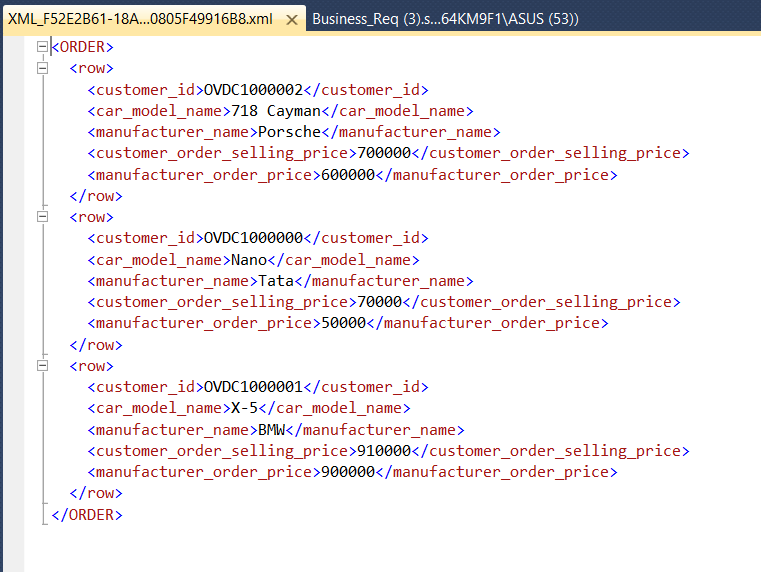
****

Figure 12 Stored Procedure 4.2

1. Retrieving data logically from a field with XML data type as well as data from fields from other data types –

The stored procedure will take an input as car type for example – Sports. The output will display all the cars that can be ordered through the Over Drive dealers. The output can be showed to the customers when they come with a specific intention of buying a particular car type. The output will have an XML data field as well which will show the car specification.

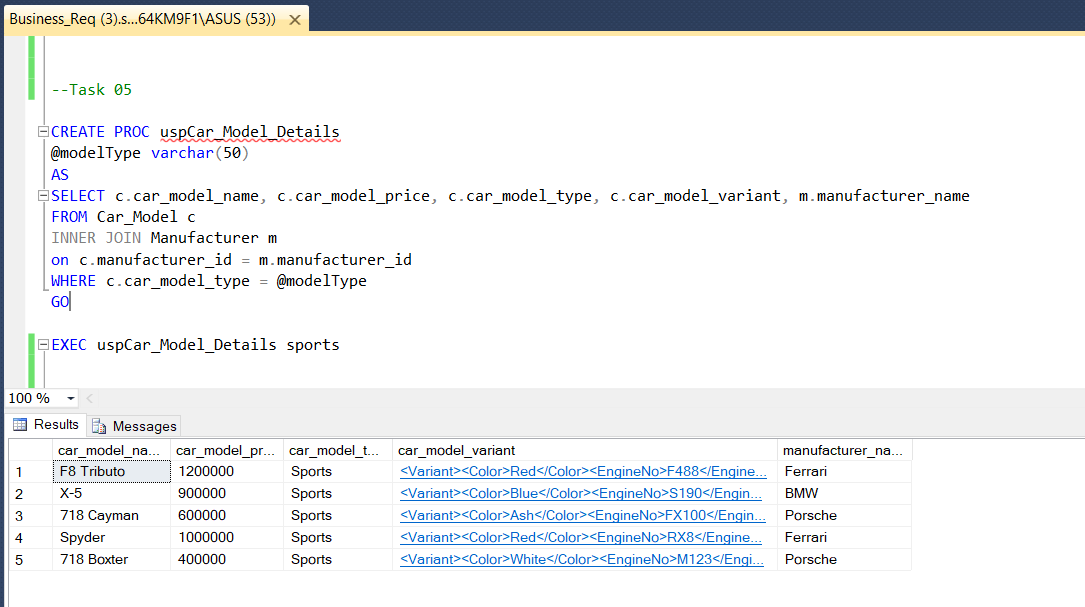


Figure 13 Stored Procedure 5

1. Modifying data in a field of XML data type –

The stored procedure will be used to update one XML field which is address in this case (street name). This will enable the Over Drive administration team to edit the employee address if there is a change in same. The Over Drive aims to have updated details of all employees at all instance.

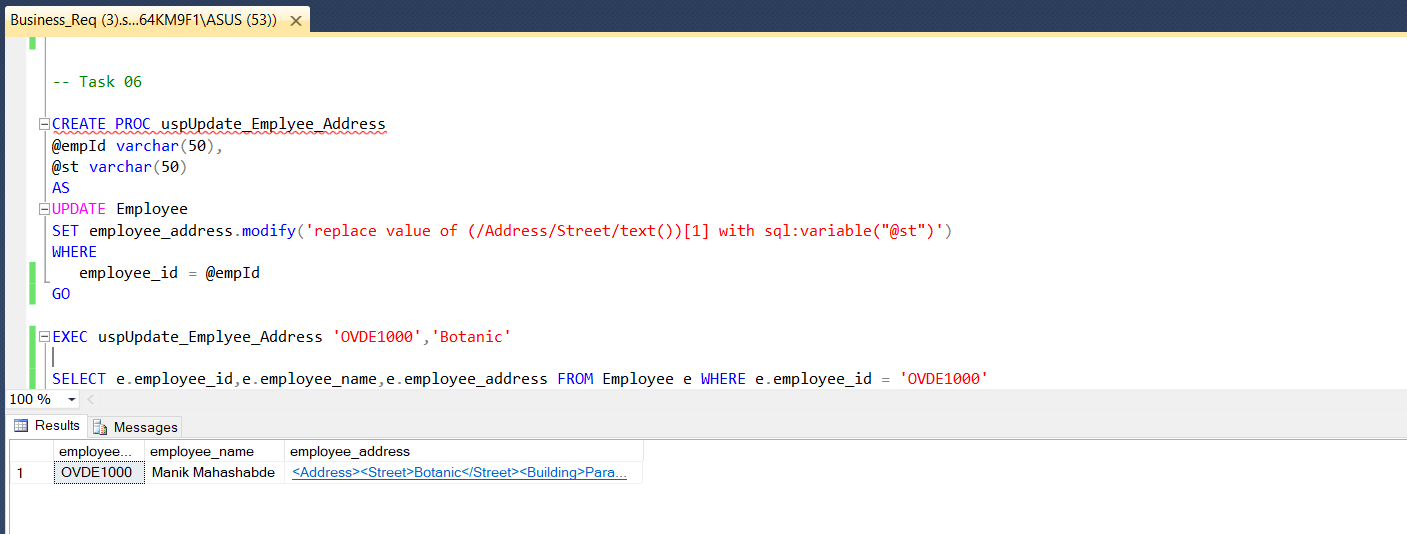


Figure 14 Stored Procedure 6

1. Searching data in a field of XML data type –

The stored procedure will be used to search a particular input from the sales department for referring when a particular customer wants to see which all cars can be ordered having a specified number of seats in them.

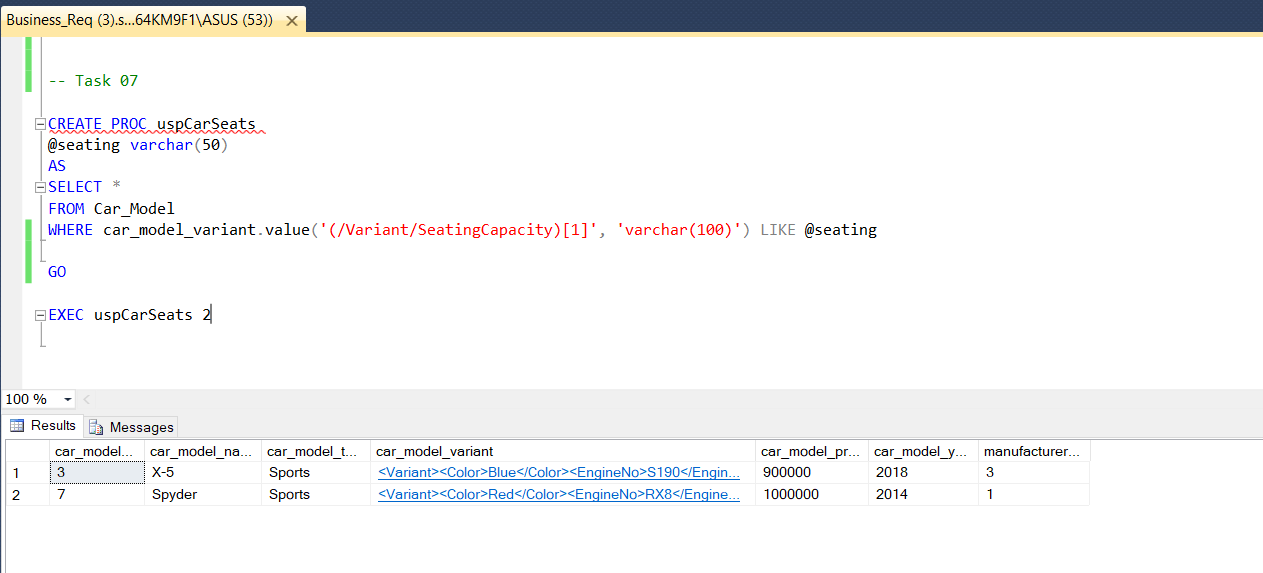


Figure 15 Stored Procedure 7

## Triggers

Trigger is kind of stored procedure which gets invoked automatically when a particular event takes place in a database. Triggers are used to maintain the referential integrity of data by changing the data in a systematic fashion. Triggers can be defined to run instead of or after DML (Data Manipulation Language) actions such as INSERT, UPDATE, and DELETE (‘SQL Trigger | Student Database’, 2018).

The Over Drive database implementation has 2 triggers implemented to enforce the business rules on the tables.

1. INSTEAD OF TRIGGER –

Since the customer order table is the most important table to be maintained for the Over Drive dealers, an instead trigger has been implemented to back the data up inside a Customer order log table.

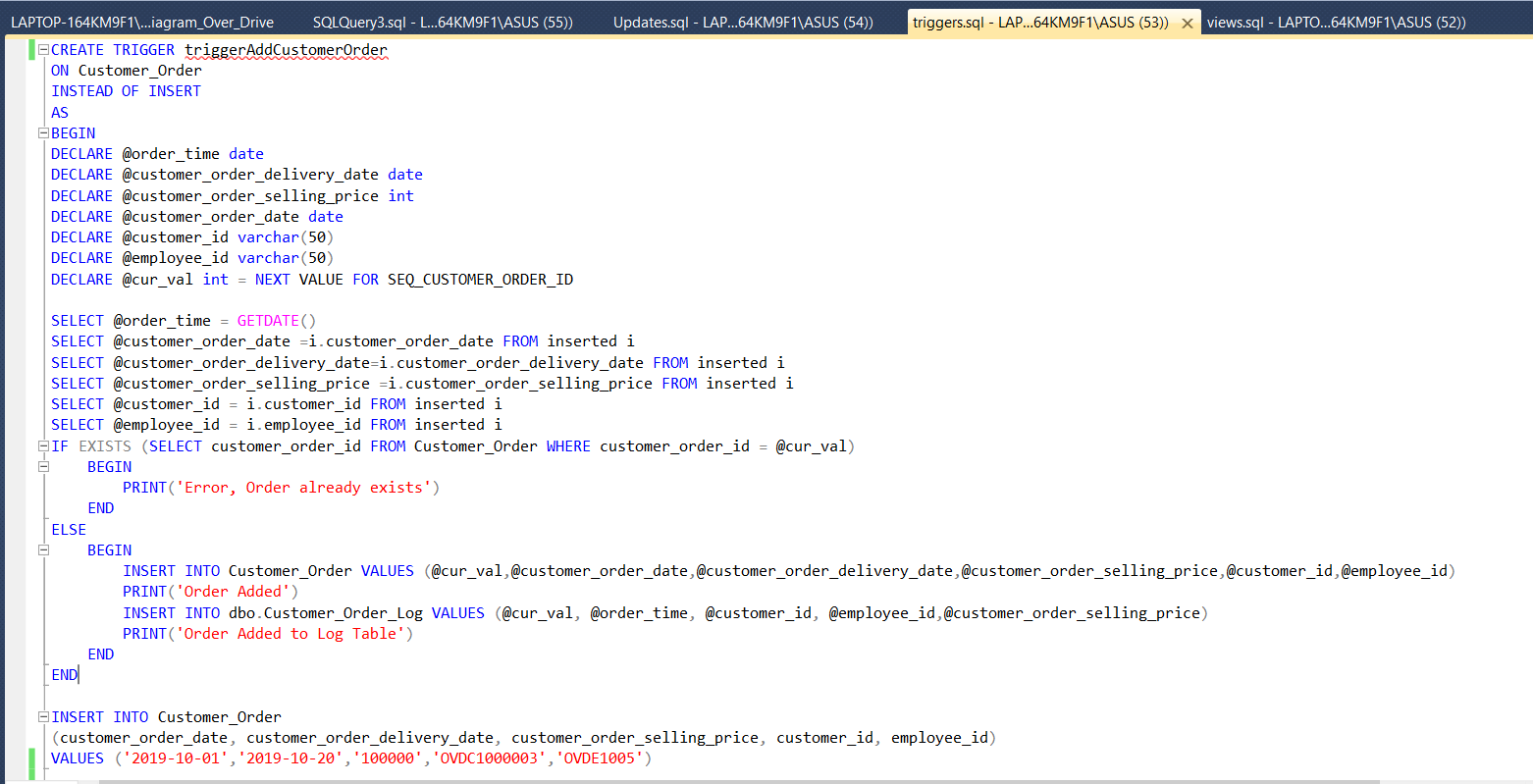


Figure 16 Trigger 1.1

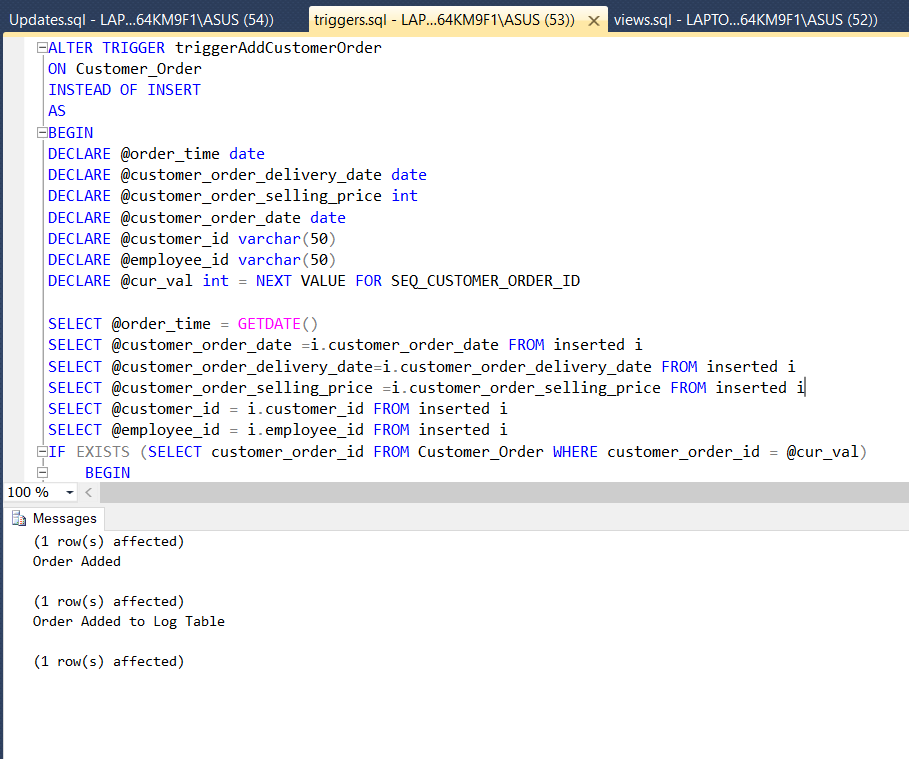


Figure 17 Trigger 1.2

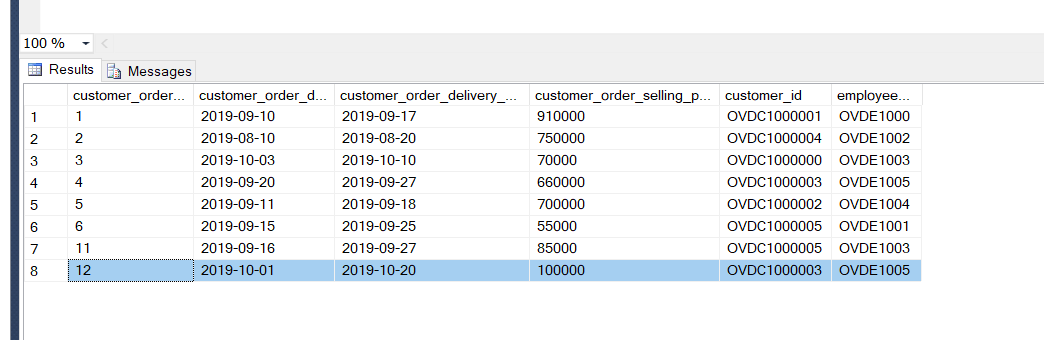


Figure 18 Trigger 1.3

1. AFTER TRIGGER –

For inserting a record in the incentive table for employees, an after trigger is implemented which calculates the incentive amount for that employee using a function and then inserts it in the respective incentive table. This ensures that the incentive amount is correct every time the insertion is made in the Incentive table.

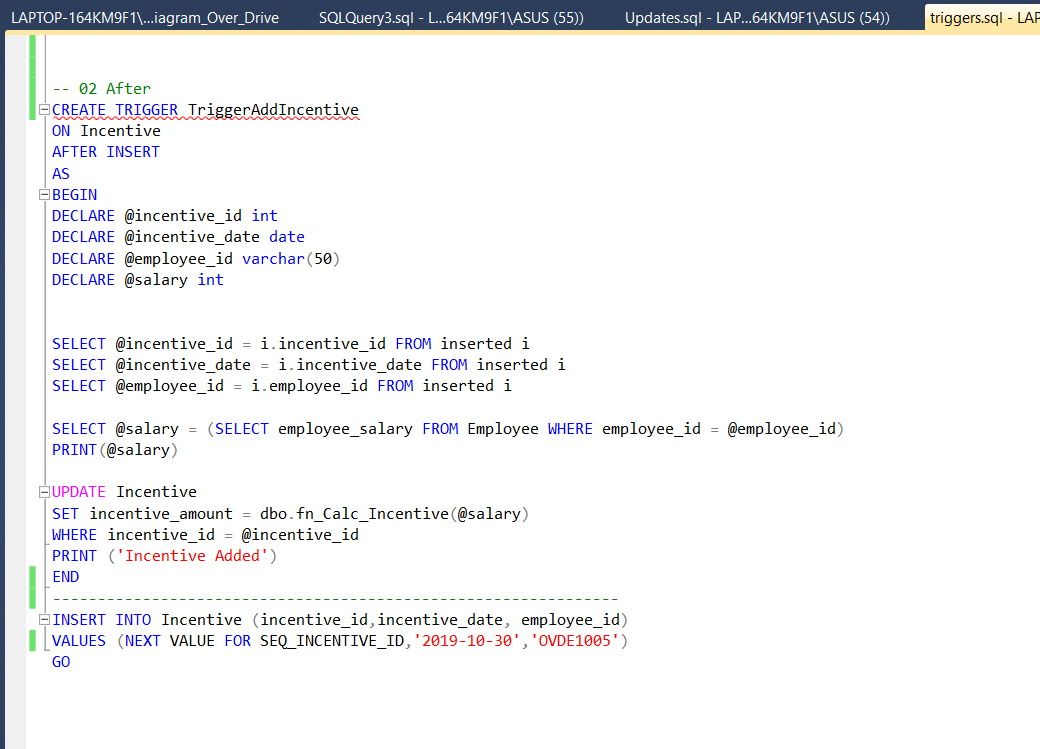
****

Figure 19 Trigger 2.1

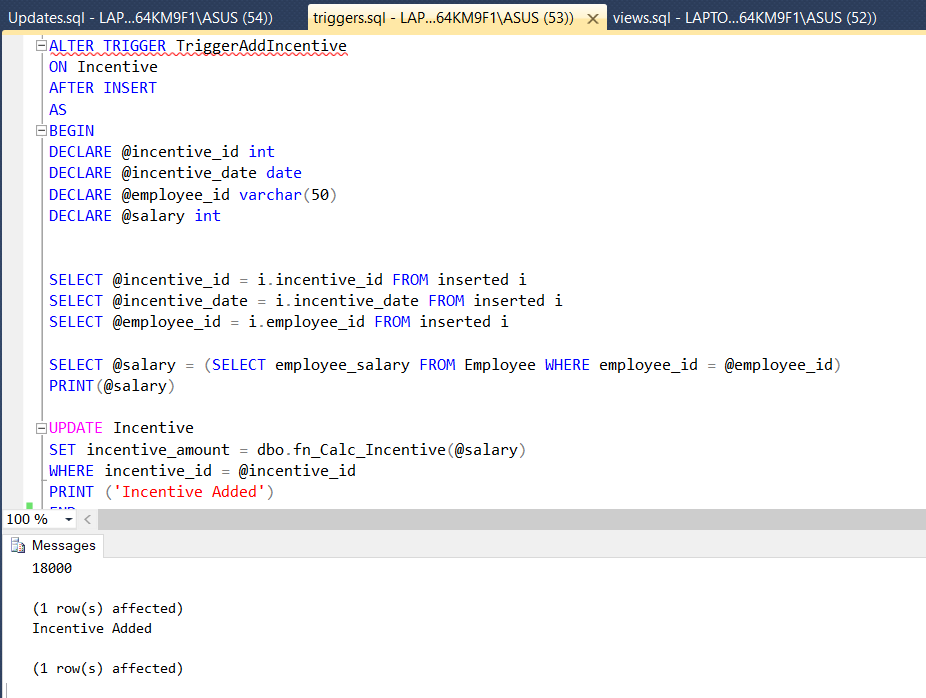
****

Figure 20 Trigger 2.2

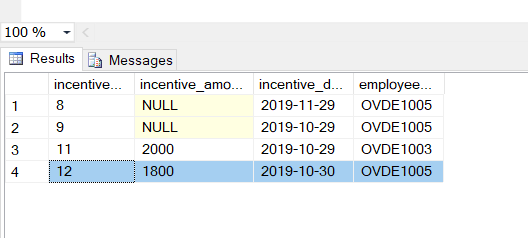
****

Figure 21 Trigger 2.3

## Views

A View in a database is kind of a searchable object. It doesn’t store data and often referred to as a virtual table. Querying a view can be same as querying a table. A view can combine data from two or more table, using joins, and also just contain a subset of information. Views are basically used to hide the complexity of the data. Views take very little place to store and it does not store the copy of the data that it represents (*What is a Relational Database View?*, 2014).

1. VIEW 1 –

In this the profit gained on every order is displayed. The profit gained will be the difference between the selling price to the customer and the purchasing price from the manufacturer. Such view can be used by the manager as well as auditors to track the profit earned on the cars.

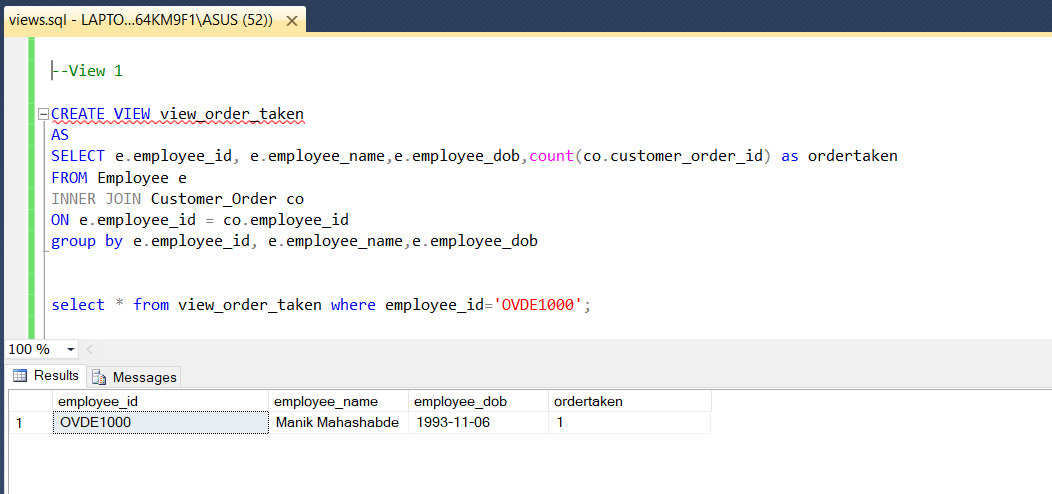
****

Figure 22 View 1

1. VIEW 2 –

In this the number of orders successfully achieved by every employee is displayed. This will help the manager to track the performing and under-performing employees at the Over Drive car dealers. That would influence his/her decision in giving promotions and incentives.

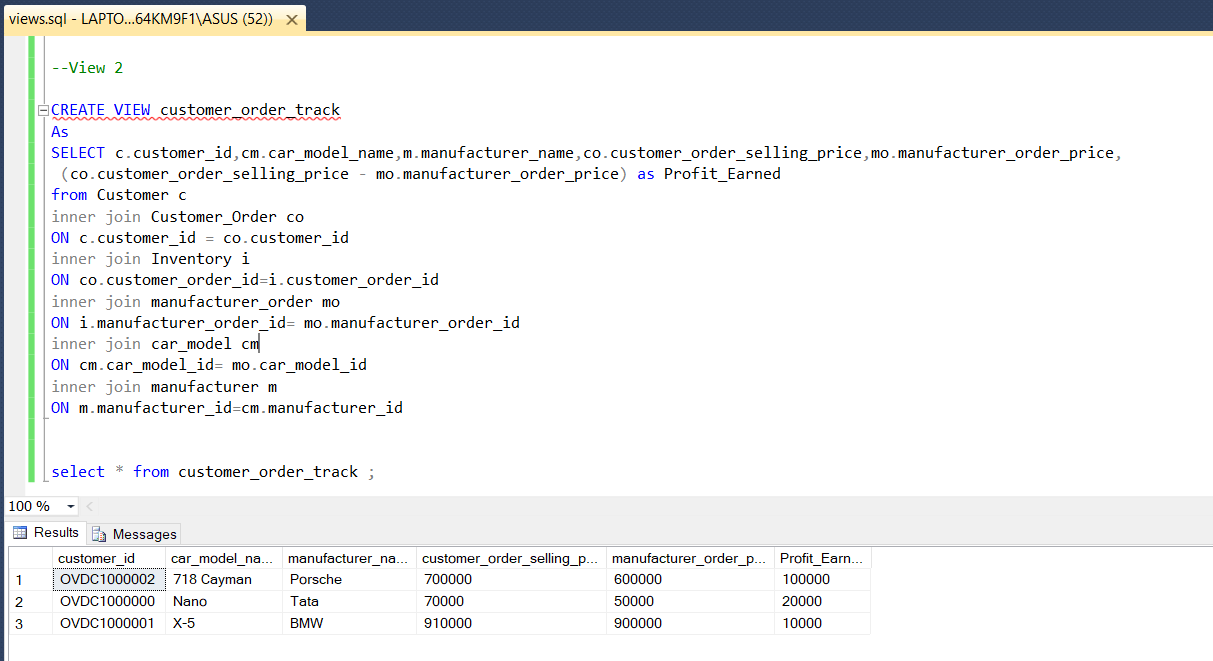
****

Figure 23 View 2

# Innovation

In the modern age, business requirements are all about accessibility, speed and scalability. These factors directly affect the growth of Overdrive car dealership. Trying to address these might require huge capital investment. Majority of the expenses during Overdrive’s business expansion goes into things like showroom, warehouse, vehicle procurement, personnel and other such things. A lot of this cost can be reduced if certain common functional elements can be centralized like for example, a common warehouse for stationing procured cars, single HR and Financial team to oversee operations across multiple branches. In order to enable this kind of collaboration among the multiple branches, a single database management system will be the best approach. This will help maintain centralized control over common data resources like employees, customers, manufacture orders, incentives among others.

Cloud technologies can help achieve this for Overdrive. With Cloud technologies progressing leaps and bounds in the last decade, the ability to scale a business has been made accessible to anyone who desires at competitive prices. Taking advantage of these features, Overdrive car dealership has come up with the innovative idea of moving their database management system to Azure SQL Database Management.

Azure SQL Databases provides the following advantages to Overdrive:

1. Database-as-a-service (DBaaS) – Azure provides SQL Database as a service platform with options ranging from general purpose machines to Hyper scalable and business critical ones. With the current business requirements, Overdrive chose to go ahead with a general purpose serverless machine. These machines provide auto-scaling options depending on the minimum and maximum resource values set during initialization.
2. Hyper scalable Database – With cloud databases, scaling the database is just a click of a button away. Within minutes Overdrive can fully scale up their database (up to 100TB) based on requirements. These reduce lot of infrastructure costs in building, maintaining and upgrading a local database within the organization.
3. Azure SQL Database Intelligent Performance – Azure SQL Database’s provides intelligent performance insights like query performance insights, automatic tuning, continuous database monitoring to prevent disruptive events. The SQL Intelligent data protection also helps Overdrive to meet the data security and compliance requirements by proactively monitoring for potential threats and vulnerabilities.
4. Reduce cost and boost productivity – The ability to use all of the above features without having to purchase and setup any of the related hardware resources or personnel greatly reduces the business cost and increases productivity of Overdrive by letting the organization focus on the actual business rather than the chores of maintaining it.

(*Three reasons Azure SQL Database is best for SQL Server migrations*, no date)

The figures below show the implementation of the Overdrive Database in Azure.

A screenshot of a computer

Description automatically generated

Figure 24 Azure DB Overview

A screenshot of a computer

Description automatically generated

Figure 25 Azure DB Table View

# Conclusion

From the screenshots we can see that Overdrive car dealership has the Hybrid Database implemented successfully with all the business requirements fulfilled. For future expansion prospects the same can also be hosted on the cloud as depicted in the innovation section.

# Individual Contribution

## Vinit Sawant

Contributed to the selection of the business case to be implemented for the CA. Decided on the entities to be taken up for the scenario and the various business requirements which the business can have and which are essential for the functioning of the Over Drive car dealer’s enterprise. Discussed with the team members about the various business rules that are essential and can be enforced, so as to maintain proper set of data in the database. Contributed in developing the ER diagrams and relational schema for the project. Discussed with the team on what fields need to be included in the various tables and what data columns should have the XML data type. Mainly those data fields were suggested by me, were semi-structured in nature and are subject to vary based on the entry like car specification as well as the address of the customer or the employee of Over Drive car dealers. I went through the various websites to understand the concept of stored procedures and the various benefits of using the stored procedures that our enterprise can get after the implementation of business requirements in same. Decided upon the exact scenarios, where the triggers can be used and before that went through the lecture slides on triggers and different types of triggers. Understood what Views are? and how they can be used to present the data in a customized manner to the end user or interested client without revealing the structure of the table or compromising on the storage of the database as views are just virtual tables. Contributed directly in coding the business requirements in the form of stored procedures, and views. Suggested the team member Kumuditha Athukorala on what needs to implement for the triggers section. Participated and lead the group meetings to discuss on the various functionalities, analysis, planning and implementation and last but not the least, content to be added in the report so as to make it a decent professional report. It was indeed a great learning experience to go through the various concepts of Advanced Databases module for implementation of the Hybrid database for the Over Drive car dealers. A real-world market kind scenario was experienced by the team members during the implementation. Understanding each and every concept from the very scratch and implementing the same in the real world was a great learning outcome and would help me surely in the future. Besides learning the technical part of the implementation, the experience to lead the team and manage the various tasks, allocate them to the team members was also a milestone that was achieved within the decided timeline of the project. Minimum planned effort from each member every week was achieved by giving small tasks to the team members and later a discussion on same used to take place if anyone faced during the execution of same. Also due to the ever-guiding nature of our module guide Dr. Shazia Afzal we were able to achieve all the functionalities of our project.

## Kumuditha Athukorala

After the discussion of CA brief three of us decided to implement hybrid database system for a car dealer named as Over Drive. From the beginning I contributed my effort in order to implement the Ovedrive database system. I have followed may web applications which are available on the internet to identify the business scenario and outline the scope of our database system. Then I initiated the business requirements collaboratively in order to proceed the ER diagrams. After identifying the entities of the system, I draw the ER diagram. There I used the Advanced Databases theories and concepts which I have learned from this program. After the feedbacks we received from the lecturer I have changed the ER diagram accordingly and develop the Database schema diagram based on that. According to the normalization theories, I have designed the schema diagram in 3NF from to draw the diagrams, I used crow’s foot notation. There we collaboratively define the data types with the length. Also, we collaboratively figure it out xml fields based on the business scenario. Then I created the Ovedrive data base with all the tables. Ovedrive data base system consists with tables, triggers, views and stored procedures. During the timetable creation for the Ovedrive database, in order to apply the referential integrity and constraints I have followed few web sites and examples which I learned from this program. After the completion of relational schema, I contributed my effort to insert sample data to the database. Then I generated the data diagram in SQL Server Management Studio. Then I have implemented most of the stored procedures in order to address the business requirements. In order to meet the business requirements, I have used join quires with the tables. Also, I have modified stored procedures with arguments. To update the xml fields of the database tables I designed the stored procedure with parameters as well, and there I had to learn about the sorted procedures, and I have followed many web sites for that. Remain sorted procedures we have implemented collaboratively based on our business requirements. In the Overdrive database we have identified triggers to implement, myself and Vinit implemented the triggers in order to fulfill the business requirement.

The reflection of this project for me is really valuable since these concepts and theories are using in the software industry. With respect to my contribution for this CA, learnings such as ER diagrams, Database schema diagrams, normalization concepts were important to me. As well as to implement stored procedures based on business requirements, learnings and the class exercises were useful for me and I learned how to implement stored procedures with arguments and how to manipulate xml values via stored procedure as well. Then I came to know how to use triggers on top of a database and how to alter these database objects if there is anything to be updated. At last but not least I convey my gratitude to Dr Shazia A Afzal regarding the given guidance.

## Srikanth Shilesh Pasam

Reflection –

The Advanced Databases CA has been one of the most indulging learning experiences for me. I come from Electronics and Communication Engineering background with no prior knowledge of the subject. My professional experience has been majorly in Teach for India, a non-government organization within the education sector. So, I have no professional experience working with databases either. Because of this I had to put in extra efforts to learn everything form the scratch. One of the most important things I am thankful for with regards to the CA is the amount of time that was given to submit it. I have been able to utilize this time to the fullest in catching up to the standard expected at master’s level. The course professor, Dr. Shazia Afzal has been extremely supportive in this regard. Her empathy towards students who were similar to me and her patience in the class by walking through every single line of a code has been of tremendous help. This was more evident when she broke down the CA into smaller deliverables making sure during every step that the project is flowing on the right track. If not for that I would have definitely been still struggling with many things.

I faced the most difficulty during the initial days of the project where I was unable to even pen down my ideas or a plot a plan of action on how to proceed. Most of this time I engrossed myself with various books from the library and online courses on Udemy. The progress was slow and whenever I heard my fellow students discuss about their progress with their CA’s it used to fill me with a lot of panic. But I kept pushing myself and through my perseverance I slowly but steadily started to make progress with my CA.

Teamwork has been the quintessential part of our group. Every single aspect of the CA has been clearly discussed and communicated with each other. A large part of the brainstorming sessions involved us questioning each other. Through techniques like 5 Why’s we pushed each other’s thinking in order to deepen our understanding. My teammates have been one of my strong pillars of support. They have been extremely patient with me and gave me the time and space to catch up to them and contribute to the CA. They have been able to address all my queries regarding the course whenever I got stuck anywhere.

Contribution –

The project has been divided into multiple parts with each of us taking up ownership for our respective topics. Apart from the combined contributions put into every aspect of the CA, my contribution has been towards the Business Requirements and innovation. While my team has been working on implementing the database on MS SQL, I have been parallelly trying to get it done in Azure SQL Database. This approach helped us gain different perspectives with regards to the various queries and table implementation logic. There were even instances where we went back to rethinking the entire business model because of the difference in perspectives. Finally, towards the end of the CA we collaborated all our work into one CA.

# Bibliography

Rouse, M. (no date) *What is a Database? - Definition from WhatIs.com*, *SearchSQLServer*. Available at: https://searchsqlserver.techtarget.com/definition/database (Accessed: 10 December 2019).

‘SQL Server Stored Procedures Tutorial’ (no date) *SQL Server Tutorial*. Available at: http://www.sqlservertutorial.net/sql-server-stored-procedures/ (Accessed: 17 December 2019).

‘SQL Trigger | Student Database’ (2018) *GeeksforGeeks*, 24 September. Available at: https://www.geeksforgeeks.org/sql-trigger-student-database/ (Accessed: 17 December 2019).

*Three reasons Azure SQL Database is best for SQL Server migrations* (no date). Available at: https://azure.microsoft.com/en-us/blog/three-reasons-azure-sql-database-is-best-for-sql-server-migration/ (Accessed: 17 December 2019).

*What is a database?* (no date). Available at: https://www.oracle.com/database/what-is-database.html (Accessed: 10 December 2019).

*What is a relational database?* (no date). Available at: https://www.oracle.com/ie/database/what-is-a-relational-database/ (Accessed: 10 December 2019).

*What is a Relational Database View?* (2014) *Essential SQL*. Available at: https://www.essentialsql.com/what-is-a-relational-database-view/ (Accessed: 17 December 2019).

*What is Microsoft SQL Server and What is it Used For?* (2017) *Infotec*. Available at: https://www.infotectraining.com/blog/what-is-microsoft-sql-server-and-what-is-it-used-for (Accessed: 10 December 2019).

# Appendix 1

CREATE TABLE Manufacturer(

manufacturer\_id int not null,

manufacturer\_name varchar(100) not null UNIQUE,

manufacturer\_address varchar(150) not null,

manufacturer\_email varchar(50) not null,

manufacturer\_phone varchar(50) not null,

CONSTRAINT manufacturer\_id\_pk PRIMARY KEY (manufacturer\_id)

)

GO

CREATE TABLE Car\_Model(

car\_model\_id int not null,

car\_model\_name varchar(50) not null UNIQUE,

car\_model\_type varchar(50) not null,

car\_model\_variant xml,

car\_model\_price int not null,

car\_model\_year int not null,

manufacturer\_id int not null,

CONSTRAINT car\_model\_id\_pk PRIMARY KEY (car\_model\_id),

CONSTRAINT manufacturer\_id\_fk FOREIGN KEY (manufacturer\_id)

REFERENCES Manufacturer(manufacturer\_id)

)

GO

CREATE TABLE Manufacturer\_Order(

manufacturer\_order\_id int not null,

manufacturer\_order\_date date not null,

manufacturer\_order\_price int not null,

car\_model\_id int not null,

CONSTRAINT manufacturer\_order\_id\_pk PRIMARY KEY (manufacturer\_order\_id),

CONSTRAINT car\_model\_id\_fk FOREIGN KEY (car\_model\_id)

REFERENCES Car\_Model(car\_model\_id)

)

GO

CREATE TABLE Customer(

customer\_id varchar(50) not null,

customer\_name varchar(100) not null,

customer\_address xml,

customer\_phone varchar(50) not null,

CONSTRAINT customer\_id\_pk PRIMARY KEY (customer\_id)

)

GO

CREATE TABLE Employee(

employee\_id varchar(50) not null,

employee\_name varchar(100) not null,

employee\_designation varchar(50) not null,

employee\_address xml,

employee\_dob date not null,

employee\_pps\_number varchar(100) not null,

employee\_salary int not null,

CONSTRAINT employee\_id\_pk PRIMARY KEY (employee\_id)

)

GO

CREATE TABLE Customer\_Order(

customer\_order\_id int not null,

customer\_order\_date date not null,

customer\_order\_delivery\_date date not null,

customer\_order\_selling\_price int not null,

customer\_id varchar(50) not null,

employee\_id varchar(50) not null,

CONSTRAINT customer\_order\_id\_pk PRIMARY KEY (customer\_order\_id),

CONSTRAINT customer\_id\_fk FOREIGN KEY (customer\_id)

REFERENCES Customer(customer\_id),

CONSTRAINT employee\_id\_fk1 FOREIGN KEY (employee\_id)

REFERENCES Employee(employee\_id)

)

GO

CREATE TABLE Incentive(

incentive\_id int not null,

incentive\_amount int,

incentive\_date date not null,

employee\_id varchar(50) not null,

CONSTRAINT incentive\_id\_pk PRIMARY KEY (incentive\_id),

CONSTRAINT employee\_id\_fk2 FOREIGN KEY (employee\_id)

REFERENCES Employee(employee\_id)

)

GO

CREATE TABLE Inventory(

inventory\_id int not null,

inventory\_date date not null,

inventory\_status varchar(10) not null,

manufacturer\_order\_id int not null,

customer\_order\_id int,

CONSTRAINT inventory\_id\_pk PRIMARY KEY (inventory\_id),

CONSTRAINT manufacturer\_order\_id\_fk FOREIGN KEY (manufacturer\_order\_id)

REFERENCES Manufacturer\_Order(manufacturer\_order\_id),

CONSTRAINT customer\_order\_id\_fk FOREIGN KEY (customer\_order\_id)

REFERENCES Customer\_Order(customer\_order\_id),

)

GO

CREATE TABLE Customer\_Order\_Log(

order\_id int not null PRIMARY KEY,

order\_time date not null,

customer\_id varchar(50) not null,

employee\_id varchar(50) not null,

customer\_order\_selling\_price int not null,

)

# Appendix 2

-- Task 01

CREATE PROC uspCustomer\_Order\_Details\_with\_Id

@id varchar(50)

AS

SELECT \*

FROM Customer c

INNER JOIN Customer\_Order co

ON c.customer\_id = co.customer\_id

WHERE c.customer\_id LIKE @id +'%'

GO

EXEC uspCustomer\_Order\_Details\_with\_Id 'OVDC1000000'

GO

-- Task 02

CREATE PROC uspEmployee\_Performance

@price int

AS

SELECT e.employee\_id, e.employee\_name, SUM(co.customer\_order\_selling\_price) AS Total\_Sales

FROM Employee e

INNER JOIN Customer\_Order co

ON e.employee\_id = co.employee\_id

GROUP BY e.employee\_id, e.employee\_name

HAVING SUM(co.customer\_order\_selling\_price) > @price

GO

EXEC uspEmployee\_Performance 60000

GO

-- Task 03

CREATE function fn\_Calc\_Incentive(@salary as int)

returns float

AS

BEGIN

DECLARE @incentive as float

SET @incentive = @salary \* 0.1

RETURN @incentive

END

GO

CREATE PROC uspCall\_fn\_Calc\_Incentive

@sal int

AS

SELECT dbo.fn\_Calc\_Incentive(@sal) AS Incentive\_Amount

GO

EXEC uspCall\_fn\_Calc\_Incentive 20000

--Task 04 -

CREATE PROC uspXML\_CUSTOMER\_ORDERS

AS

SELECT Customer.customer\_id,car\_model.car\_model\_name,manufacturer.manufacturer\_name,customer\_order.customer\_order\_selling\_price,manufacturer\_order.manufacturer\_order\_price

from Customer

inner join Customer\_Order

ON Customer.customer\_id = Customer\_Order.customer\_id

inner join Inventory

ON Customer\_Order.customer\_order\_id=Inventory.customer\_order\_id

inner join manufacturer\_order

ON Inventory.manufacturer\_order\_id= manufacturer\_order.manufacturer\_order\_id

inner join car\_model

ON manufacturer\_order.car\_model\_id= car\_model.car\_model\_id

inner join manufacturer

ON manufacturer.manufacturer\_id=car\_model.manufacturer\_id

FOR XML RAW, ELEMENTS, ROOT('ORDER');

exec uspXML\_CUSTOMER\_ORDERS;

--Task 05

CREATE PROC uspCar\_Model\_Details

@modelType varchar(50)

AS

SELECT c.car\_model\_name, c.car\_model\_price, c.car\_model\_type, c.car\_model\_variant, m.manufacturer\_name

FROM Car\_Model c

INNER JOIN Manufacturer m

on c.manufacturer\_id = m.manufacturer\_id

WHERE c.car\_model\_type = @modelType

GO

EXEC uspCar\_Model\_Details sports

--Task 06

CREATE PROC uspUpdate\_Emplyee\_Address

@empId varchar(50),

@st varchar(50)

AS

UPDATE Employee

SET employee\_address.modify('replace value of (/Address/Street/text())[1] with sql:variable("@st")')

WHERE

employee\_id = @empId

EXEC uspUpdate\_Emplyee\_Address 'OVDE1000','Botanic'

--Task 07

CREATE PROC uspCarSeats

@seating varchar(50)

AS

SELECT \*

FROM Car\_Model

WHERE car\_model\_variant.value('(/Variant/SeatingCapacity)[1]', 'varchar(100)') LIKE @seating

EXEC uspCarSeats 2

# Appendix 3

-- Insert Manufacturer

GO

CREATE SEQUENCE SEQ\_MANUFACTURER\_ID START WITH 1 INCREMENT BY 1;

GO

SELECT \* FROM Manufacturer

GO

INSERT INTO Manufacturer VALUES (NEXT VALUE FOR SEQ\_MANUFACTURER\_ID,'Ferrari','Maranello,Italy','ferraicars@gmail.com','1800553946')

INSERT INTO Manufacturer VALUES (NEXT VALUE FOR SEQ\_MANUFACTURER\_ID,'Mercedes','Stuttgrad,Germany','mercedesinfo@gmail.com','080097777777')

INSERT INTO Manufacturer VALUES (NEXT VALUE FOR SEQ\_MANUFACTURER\_ID,'BMW','Munich,Germany','bmw@gmail.com','49893820')

INSERT INTO Manufacturer VALUES (NEXT VALUE FOR SEQ\_MANUFACTURER\_ID,'Porsche','Stuttgrad,Germany','porsche@gmail.com','498756223')

INSERT INTO Manufacturer VALUES (NEXT VALUE FOR SEQ\_MANUFACTURER\_ID,'Tata','Mumbai,India','tata@gmail.com','18002582553')

INSERT INTO Manufacturer VALUES (NEXT VALUE FOR SEQ\_MANUFACTURER\_ID,'Micro','Colombo,Sri Lanka','micro@msn.com','94089554416')

-- Insert Car\_Model

GO

CREATE SEQUENCE SEQ\_CAR\_MODEL\_ID START WITH 1 INCREMENT BY 1;

INSERT INTO Car\_Model VALUES (NEXT VALUE FOR SEQ\_CAR\_MODEL\_ID,'F8 Tributo','Sports','','1200000','2016','1')

INSERT INTO Car\_Model VALUES (NEXT VALUE FOR SEQ\_CAR\_MODEL\_ID,'S-Class','Luxury','','500000','2014','2')

INSERT INTO Car\_Model VALUES (NEXT VALUE FOR SEQ\_CAR\_MODEL\_ID,'X-5','Sports','','900000','2018','3')

INSERT INTO Car\_Model VALUES (NEXT VALUE FOR SEQ\_CAR\_MODEL\_ID,'718 Cayman','Sports','','600000','2015','4')

INSERT INTO Car\_Model VALUES (NEXT VALUE FOR SEQ\_CAR\_MODEL\_ID,'Nano','Hatch back','','50000','2011','5')

INSERT INTO Car\_Model VALUES (NEXT VALUE FOR SEQ\_CAR\_MODEL\_ID,'Panda','Sedan','','50000','2015','6')

INSERT INTO Car\_Model VALUES (NEXT VALUE FOR SEQ\_CAR\_MODEL\_ID,'Spyder','Sports','','1000000','2014','1')

INSERT INTO Car\_Model VALUES (NEXT VALUE FOR SEQ\_CAR\_MODEL\_ID,'M-Class','Sedan','','400000','2012','2')

INSERT INTO Car\_Model VALUES (NEXT VALUE FOR SEQ\_CAR\_MODEL\_ID,'X-4','SUV','','700000','2017','3')

INSERT INTO Car\_Model VALUES (NEXT VALUE FOR SEQ\_CAR\_MODEL\_ID,'718 Boxter','Sports','','400000','2013','4')

INSERT INTO Car\_Model VALUES (NEXT VALUE FOR SEQ\_CAR\_MODEL\_ID,'Sumo','SUV','','60000','2006','5')

INSERT INTO Car\_Model VALUES (NEXT VALUE FOR SEQ\_CAR\_MODEL\_ID,'MX-7','Sedan','','80000','2017','6')

-- Insert Employee OVDE1000

GO

CREATE SEQUENCE SEQ\_EMPLOYEE\_ID START WITH 1000 INCREMENT BY 1;

INSERT INTO Employee VALUES (CONCAT('OVDE', NEXT VALUE FOR SEQ\_EMPLOYEE\_ID),'Manik Mahashabde','Salesman','','1993-11-06','2341897AS','20000')

INSERT INTO Employee VALUES (CONCAT('OVDE',NEXT VALUE FOR SEQ\_EMPLOYEE\_ID),'Kumuditha Athukorala','Salesman','','1990-03-05','1233198ER','10000')

INSERT INTO Employee VALUES (CONCAT('OVDE',NEXT VALUE FOR SEQ\_EMPLOYEE\_ID),'Vint Sawant','Salesman','','1994-09-11','3452789XC','15000')

INSERT INTO Employee VALUES (CONCAT('OVDE',NEXT VALUE FOR SEQ\_EMPLOYEE\_ID),'Srikanth Pasam','Salesman','','1993-12-01','8967845BD','20000')

INSERT INTO Employee VALUES (CONCAT('OVDE',NEXT VALUE FOR SEQ\_EMPLOYEE\_ID),'Deep Singh','Salesman','','1995-05-11','5674389AS','12000')

INSERT INTO Employee VALUES (CONCAT('OVDE',NEXT VALUE FOR SEQ\_EMPLOYEE\_ID),'Salman Bhatt','Salesman','','1996-06-23','4563098AS','18000')

INSERT INTO Employee VALUES (CONCAT('OVDE',NEXT VALUE FOR SEQ\_EMPLOYEE\_ID),'Salman Khan','Manager','','1986-07-23','4563098HS','80000')

-- Insert Incentive

GO

CREATE SEQUENCE SEQ\_INCENTIVE\_ID START WITH 1 INCREMENT BY 1;

INSERT INTO Incentive VALUES (NEXT VALUE FOR SEQ\_INCENTIVE\_ID,'1800','2019-11-01','OVDE1005')

INSERT INTO Incentive VALUES (NEXT VALUE FOR SEQ\_INCENTIVE\_ID,'1200','2019-11-03','OVDE1004')

INSERT INTO Incentive VALUES (NEXT VALUE FOR SEQ\_INCENTIVE\_ID,'2000','2019-11-06','OVDE1003')

INSERT INTO Incentive VALUES (NEXT VALUE FOR SEQ\_INCENTIVE\_ID,'1500','2019-11-10','OVDE1002')

INSERT INTO Incentive VALUES (NEXT VALUE FOR SEQ\_INCENTIVE\_ID,'1000','2019-10-21','OVDE1001')

INSERT INTO Incentive VALUES (NEXT VALUE FOR SEQ\_INCENTIVE\_ID,'2000','2019-10-29','OVDE1000')

-- Insert Customer

GO

CREATE SEQUENCE SEQ\_CUSTOMER\_ID START WITH 1000000 INCREMENT BY 1;

INSERT INTO Customer VALUES (CONCAT('OVDC',NEXT VALUE FOR SEQ\_CUSTOMER\_ID),'Chaminda Vass','','09411290901')

INSERT INTO Customer VALUES (CONCAT('OVDC',NEXT VALUE FOR SEQ\_CUSTOMER\_ID),'Virat Kohli','','09188134541')

INSERT INTO Customer VALUES (CONCAT('OVDC',NEXT VALUE FOR SEQ\_CUSTOMER\_ID),'VVS Laxman','','09190987609')

INSERT INTO Customer VALUES (CONCAT('OVDC',NEXT VALUE FOR SEQ\_CUSTOMER\_ID),'Sachin Tendulkar','','09112122211')

INSERT INTO Customer VALUES (CONCAT('OVDC',NEXT VALUE FOR SEQ\_CUSTOMER\_ID),'Wasim Akram','','09013454321')

INSERT INTO Customer VALUES (CONCAT('OVDC',NEXT VALUE FOR SEQ\_CUSTOMER\_ID),'Mohomad Hafeez','','09023097895')

-- Insert Customer\_Order

GO

CREATE SEQUENCE SEQ\_CUSTOMER\_ORDER\_ID START WITH 1 INCREMENT BY 1;

INSERT INTO Customer\_Order VALUES (NEXT VALUE FOR SEQ\_CUSTOMER\_ORDER\_ID,'2019-09-10','2019-09-17','430000','OVDC1000001','OVDE1000')

INSERT INTO Customer\_Order VALUES (NEXT VALUE FOR SEQ\_CUSTOMER\_ORDER\_ID,'2019-08-10','2019-08-20','750000','OVDC1000004','OVDE1002')

INSERT INTO Customer\_Order VALUES (NEXT VALUE FOR SEQ\_CUSTOMER\_ORDER\_ID,'2019-10-03','2019-10-10','1100000','OVDC1000000','OVDE1003')

INSERT INTO Customer\_Order VALUES (NEXT VALUE FOR SEQ\_CUSTOMER\_ORDER\_ID,'2019-09-20','2019-09-27','660000','OVDC1000003','OVDE1005')

INSERT INTO Customer\_Order VALUES (NEXT VALUE FOR SEQ\_CUSTOMER\_ORDER\_ID,'2019-09-11','2019-09-18','84000','OVDC1000002','OVDE1004')

INSERT INTO Customer\_Order VALUES (NEXT VALUE FOR SEQ\_CUSTOMER\_ORDER\_ID,'2019-09-15','2019-09-25','55000','OVDC1000005','OVDE1001')

-- Insert Manufacturer\_Order

GO

CREATE SEQUENCE SEQ\_MANUFACTURER\_ORDER\_ID START WITH 1 INCREMENT BY 1;

INSERT INTO Manufacturer\_Order VALUES (NEXT VALUE FOR SEQ\_MANUFACTURER\_ORDER\_ID,'2019-08-08','1200000','1')

INSERT INTO Manufacturer\_Order VALUES (NEXT VALUE FOR SEQ\_MANUFACTURER\_ORDER\_ID,'2019-09-01','50000','6')

INSERT INTO Manufacturer\_Order VALUES (NEXT VALUE FOR SEQ\_MANUFACTURER\_ORDER\_ID,'2019-08-01','600000','4')

INSERT INTO Manufacturer\_Order VALUES (NEXT VALUE FOR SEQ\_MANUFACTURER\_ORDER\_ID,'2019-09-11','50000','5')

INSERT INTO Manufacturer\_Order VALUES (NEXT VALUE FOR SEQ\_MANUFACTURER\_ORDER\_ID,'2019-08-25','500000','2')

INSERT INTO Manufacturer\_Order VALUES (NEXT VALUE FOR SEQ\_MANUFACTURER\_ORDER\_ID,'2019-09-08','900000','3')

--Insert Inventory

GO

CREATE SEQUENCE SEQ\_INVENTORY\_ID START WITH 1 INCREMENT BY 1;

INSERT INTO Inventory VALUES (NEXT VALUE FOR SEQ\_INVENTORY\_ID,'2019-08-10','AVAILABLE','1',null)

INSERT INTO Inventory VALUES (NEXT VALUE FOR SEQ\_INVENTORY\_ID,'2019-09-02','AVAILABLE','2',null)

INSERT INTO Inventory VALUES (NEXT VALUE FOR SEQ\_INVENTORY\_ID,'2019-08-05','SOLD','3','5')

INSERT INTO Inventory VALUES (NEXT VALUE FOR SEQ\_INVENTORY\_ID,'2019-09-14','SOLD','4','3')

INSERT INTO Inventory VALUES (NEXT VALUE FOR SEQ\_INVENTORY\_ID,'2019-08-29','AVAILABLE','5',null)

INSERT INTO Inventory VALUES (NEXT VALUE FOR SEQ\_INVENTORY\_ID,'2019-09-15','SOLD','6','1')

--update employee

UPDATE Employee

SET employee\_address = '<?xml version="1.0"?>

<Address>

<Street>Grafton</Street>

<Building>Paradise</Building>

<RoomNo>256</RoomNo>

<County>Dublin</County>

<AreaCode>RR82</AreaCode>

</Address>'

WHERE employee\_id = 'OVDE1000';

UPDATE Employee

SET employee\_address = '<?xml version="1.0"?>

<Address>

<Street>North Portland</Street>

<Building>Portland Villa</Building>

<RoomNo>10</RoomNo>

<County>Dublin</County>

<AreaCode>AR30</AreaCode>

</Address>'

WHERE employee\_id = 'OVDE1001';

UPDATE Employee

SET employee\_address = '<?xml version="1.0"?>

<Address>

<Street>Downtown</Street>

<Building>Downtown Court</Building>

<RoomNo>119</RoomNo>

<County>Carlow</County>

<AreaCode>CW02</AreaCode>

</Address>'

WHERE employee\_id = 'OVDE1002';

UPDATE Employee

SET employee\_address = '<?xml version="1.0"?>

<Address>

<Street>Rose Garden</Street>

<Building>Old Castle</Building>

<RoomNo>45</RoomNo>

<County>Wicklow</County>

<AreaCode>WK11</AreaCode>

</Address>'

WHERE employee\_id = 'OVDE1003';

UPDATE Employee

SET employee\_address = '<?xml version="1.0"?>

<Address>

<Street>River Lower</Street>

<Building>River Palace</Building>

<RoomNo>15</RoomNo>

<County>Dublin</County>

<AreaCode>DB22</AreaCode>

</Address>'

WHERE employee\_id = 'OVDE1004';

UPDATE Employee

SET employee\_address = '<?xml version="1.0"?>

<Address>

<Street>Old Garden</Street>

<Building>Garden Tower</Building>

<RoomNo>04</RoomNo>

<County>Athlon</County>

<AreaCode>AL77</AreaCode>

</Address>'

WHERE employee\_id = 'OVDE1005';

UPDATE Employee

SET employee\_address = '<?xml version="1.0"?>

<Address>

<Street>Upper Town</Street>

<Building>Uptown Court</Building>

<RoomNo>02</RoomNo>

<County>Carlow</County>

<AreaCode>CW11</AreaCode>

</Address>'

WHERE employee\_id = 'OVDE1006';

-- update customer

UPDATE Customer

SET customer\_address = '<?xml version="1.0"?>

<Address>

<Street>Balfe</Street>

<Building>Paradise</Building>

<RoomNo>56</RoomNo>

<County>Dublin</County>

<AreaCode>RR77</AreaCode>

</Address>'

WHERE customer\_id = 'OVDC1000000';

UPDATE Customer

SET customer\_address = '<?xml version="1.0"?>

<Address>

<Street>Riverland</Street>

<Building>River Villa</Building>

<RoomNo>1</RoomNo>

<County>Dublin</County>

<AreaCode>AR31</AreaCode>

</Address>'

WHERE customer\_id = 'OVDC1000001';

UPDATE Customer

SET customer\_address = '<?xml version="1.0"?>

<Address>

<Street>Townhall Street</Street>

<Building>Street Paradise</Building>

<RoomNo>11</RoomNo>

<County>Carlow</County>

<AreaCode>CW05</AreaCode>

</Address>'

WHERE customer\_id = 'OVDC1000002';

UPDATE Customer

SET customer\_address = '<?xml version="1.0"?>

<Address>

<Street>Folwer Garden</Street>

<Building>Castle Building</Building>

<RoomNo>48</RoomNo>

<County>Wicklow</County>

<AreaCode>WK15</AreaCode>

</Address>'

WHERE customer\_id = 'OVDC1000003';

UPDATE Customer

SET customer\_address = '<?xml version="1.0"?>

<Address>

<Street>River Liffy Rd.</Street>

<Building>River Palace</Building>

<RoomNo>15</RoomNo>

<County>Dublin</County>

<AreaCode>DB22</AreaCode>

</Address>'

WHERE customer\_id = 'OVDC1000004';

UPDATE Customer

SET customer\_address = '<?xml version="1.0"?>

<Address>

<Street>Old Garden</Street>

<Building>Garden Court</Building>

<RoomNo>06</RoomNo>

<County>Athlon</County>

<AreaCode>AL72</AreaCode>

</Address>'

WHERE customer\_id = 'OVDC1000005';

-- Car Varient

UPDATE Car\_Model

SET car\_model\_variant = '<?xml version="1.0"?>

<Variant>

<Color>Red</Color>

<EngineNo>F488</EngineNo>

<Fuel>Diesel</Fuel>

<Power>301</Power>

<ZeroToSixty>4.3</ZeroToSixty>

<SeatingCapacity>4</SeatingCapacity>

<Airbags>Yes</Airbags>

</Variant>'

WHERE car\_model\_id = 1;

UPDATE Car\_Model

SET car\_model\_variant = '<?xml version="1.0"?>

<Variant>

<Color>White</Color>

<EngineNo>M100</EngineNo>

<Fuel>Diesel</Fuel>

<Power>300</Power>

<ZeroToSixty>4.9</ZeroToSixty>

<SeatingCapacity>6</SeatingCapacity>

<Airbags>Yes</Airbags>

</Variant>'

WHERE car\_model\_id = 2;

UPDATE Car\_Model

SET car\_model\_variant = '<?xml version="1.0"?>

<Variant>

<Color>Blue</Color>

<EngineNo>S190</EngineNo>

<Fuel>Petrol</Fuel>

<Power>290</Power>

<ZeroToSixty>5.3</ZeroToSixty>

<SeatingCapacity>2</SeatingCapacity>

<Airbags>Yes</Airbags>

</Variant>'

WHERE car\_model\_id = 3;

UPDATE Car\_Model

SET car\_model\_variant = '<?xml version="1.0"?>

<Variant>

<Color>Ash</Color>

<EngineNo>FX100</EngineNo>

<Fuel>Diesel</Fuel>

<Power>295</Power>

<ZeroToSixty>5.6</ZeroToSixty>

<SeatingCapacity>4</SeatingCapacity>

<Airbags>Yes</Airbags>

</Variant>'

WHERE car\_model\_id = 4;

UPDATE Car\_Model

SET car\_model\_variant = '<?xml version="1.0"?>

<Variant>

<Color>Gray</Color>

<EngineNo>F488</EngineNo>

<Fuel>Petrol</Fuel>

<Power>278</Power>

<ZeroToSixty>6.5</ZeroToSixty>

<SeatingCapacity>4</SeatingCapacity>

<Airbags>No</Airbags>

</Variant>'

WHERE car\_model\_id = 5;

UPDATE Car\_Model

SET car\_model\_variant = '<?xml version="1.0"?>

<Variant>

<Color>White</Color>

<EngineNo>MX23</EngineNo>

<Fuel>Petrol</Fuel>

<Power>201</Power>

<ZeroToSixty>7.6</ZeroToSixty>

<SeatingCapacity>4</SeatingCapacity>

<Airbags>Yes</Airbags>

</Variant>'

WHERE car\_model\_id = 6;

------------------------------

UPDATE Car\_Model

SET car\_model\_variant = '<?xml version="1.0"?>

<Variant>

<Color>Red</Color>

<EngineNo>RX8</EngineNo>

<Fuel>Diesel</Fuel>

<Power>301</Power>

<ZeroToSixty>4.3</ZeroToSixty>

<SeatingCapacity>2</SeatingCapacity>

<Airbags>Yes</Airbags>

</Variant>'

WHERE car\_model\_id = 7;

UPDATE Car\_Model

SET car\_model\_variant = '<?xml version="1.0"?>

<Variant>

<Color>White</Color>

<EngineNo>M123</EngineNo>

<Fuel>Diesel</Fuel>

<Power>301</Power>

<ZeroToSixty>4.6</ZeroToSixty>

<SeatingCapacity>6</SeatingCapacity>

<Airbags>Yes</Airbags>

</Variant>'

WHERE car\_model\_id = 8;

UPDATE Car\_Model

SET car\_model\_variant = '<?xml version="1.0"?>

<Variant>

<Color>White</Color>

<EngineNo>F488</EngineNo>

<Fuel>Petrol</Fuel>

<Power>290</Power>

<ZeroToSixty>4.9</ZeroToSixty>

<SeatingCapacity>4</SeatingCapacity>

<Airbags>Yes</Airbags>

</Variant>'

WHERE car\_model\_id = 9;

UPDATE Car\_Model

SET car\_model\_variant = '<?xml version="1.0"?>

<Variant>

<Color>White</Color>

<EngineNo>M123</EngineNo>

<Fuel>Diesel</Fuel>

<Power>276</Power>

<ZeroToSixty>5.8</ZeroToSixty>

<SeatingCapacity>4</SeatingCapacity>

<Airbags>Yes</Airbags>

</Variant>'

WHERE car\_model\_id = 10;

UPDATE Car\_Model

SET car\_model\_variant = '<?xml version="1.0"?>

<Variant>

<Color>Green</Color>

<EngineNo>R888</EngineNo>

<Fuel>Diesel</Fuel>

<Power>175</Power>

<ZeroToSixty>8.3</ZeroToSixty>

<SeatingCapacity>4</SeatingCapacity>

<Airbags>No</Airbags>

</Variant>'

WHERE car\_model\_id = 11;

UPDATE Car\_Model

SET car\_model\_variant = '<?xml version="1.0"?>

<Variant>

<Color>Blue</Color>

<EngineNo>M192</EngineNo>

<Fuel>Petrol</Fuel>

<Power>205</Power>

<ZeroToSixty>5.8</ZeroToSixty>

<SeatingCapacity>4</SeatingCapacity>

<Airbags>Yes</Airbags>

</Variant>'

WHERE car\_model\_id = 12;

# Appendix 3

--View 1

CREATE VIEW view\_order\_taken

AS

SELECT e.employee\_id, e.employee\_name,e.employee\_dob,count(co.customer\_order\_id) as ordertaken

FROM Employee e

INNER JOIN Customer\_Order co

ON e.employee\_id = co.employee\_id

group by e.employee\_id, e.employee\_name,e.employee\_dob

--View 2

CREATE VIEW customer\_order\_track

As

SELECT c.customer\_id,cm.car\_model\_name,m.manufacturer\_name,co.customer\_order\_selling\_price,mo.manufacturer\_order\_price,

(co.customer\_order\_selling\_price - mo.manufacturer\_order\_price) as Profit\_Earned

from Customer c

inner join Customer\_Order co

ON c.customer\_id = co.customer\_id

inner join Inventory i

ON co.customer\_order\_id=i.customer\_order\_id

inner join manufacturer\_order mo

ON i.manufacturer\_order\_id= mo.manufacturer\_order\_id

inner join car\_model cm

ON cm.car\_model\_id= mo.car\_model\_id

inner join manufacturer m

ON m.manufacturer\_id=cm.manufacturer\_id

# Appendix 4

-- 01- instead

CREATE TRIGGER triggerAddCustomerOrder

ON Customer\_Order

INSTEAD OF INSERT

AS

BEGIN

DECLARE @order\_time date

DECLARE @customer\_order\_delivery\_date date

DECLARE @customer\_order\_selling\_price int

DECLARE @customer\_order\_date date

DECLARE @customer\_id varchar(50)

DECLARE @employee\_id varchar(50)

DECLARE @cur\_val int = NEXT VALUE FOR SEQ\_CUSTOMER\_ORDER\_ID

SELECT @order\_time = GETDATE()

SELECT @customer\_order\_date =i.customer\_order\_date FROM inserted i

SELECT @customer\_order\_delivery\_date=i.customer\_order\_delivery\_date FROM inserted i

SELECT @customer\_order\_selling\_price =i.customer\_order\_selling\_price FROM inserted i

SELECT @customer\_id = i.customer\_id FROM inserted i

SELECT @employee\_id = i.employee\_id FROM inserted i

IF EXISTS (SELECT customer\_order\_id FROM Customer\_Order WHERE customer\_order\_id = @cur\_val)

BEGIN

PRINT('Error, Order already exists')

END

ELSE

BEGIN

INSERT INTO Customer\_Order VALUES (@cur\_val,@customer\_order\_date,@customer\_order\_delivery\_date,@customer\_order\_selling\_price,@customer\_id,@employee\_id)

PRINT('Order Added')

INSERT INTO dbo.Customer\_Order\_Log VALUES (@cur\_val, @order\_time, @customer\_id, @employee\_id,@customer\_order\_selling\_price)

PRINT('Order Added to Log Table')

END

END

INSERT INTO Customer\_Order

(customer\_order\_date, customer\_order\_delivery\_date, customer\_order\_selling\_price, customer\_id, employee\_id)

VALUES ('2019-09-16','2019-09-27','85000','OVDC1000005','OVDE1003')

-- 02 After

CREATE TRIGGER TriggerAddIncentive

ON Incentive

AFTER INSERT

AS

BEGIN

DECLARE @incentive\_id int

DECLARE @incentive\_date date

DECLARE @employee\_id varchar(50)

DECLARE @salary int

SELECT @incentive\_id = i.incentive\_id FROM inserted i

SELECT @incentive\_date = i.incentive\_date FROM inserted i

SELECT @employee\_id = i.employee\_id FROM inserted i

SELECT @salary = (SELECT employee\_salary FROM Employee WHERE employee\_id = @employee\_id)

PRINT(@salary)

UPDATE Incentive

SET incentive\_amount = dbo.fn\_Calc\_Incentive(@salary)

WHERE incentive\_id = @incentive\_id

PRINT ('Incentive Added')

END

INSERT INTO Incentive (incentive\_id,incentive\_date, employee\_id)

VALUES (NEXT VALUE FOR SEQ\_INCENTIVE\_ID,'2019-10-29','OVDE1003')

GO