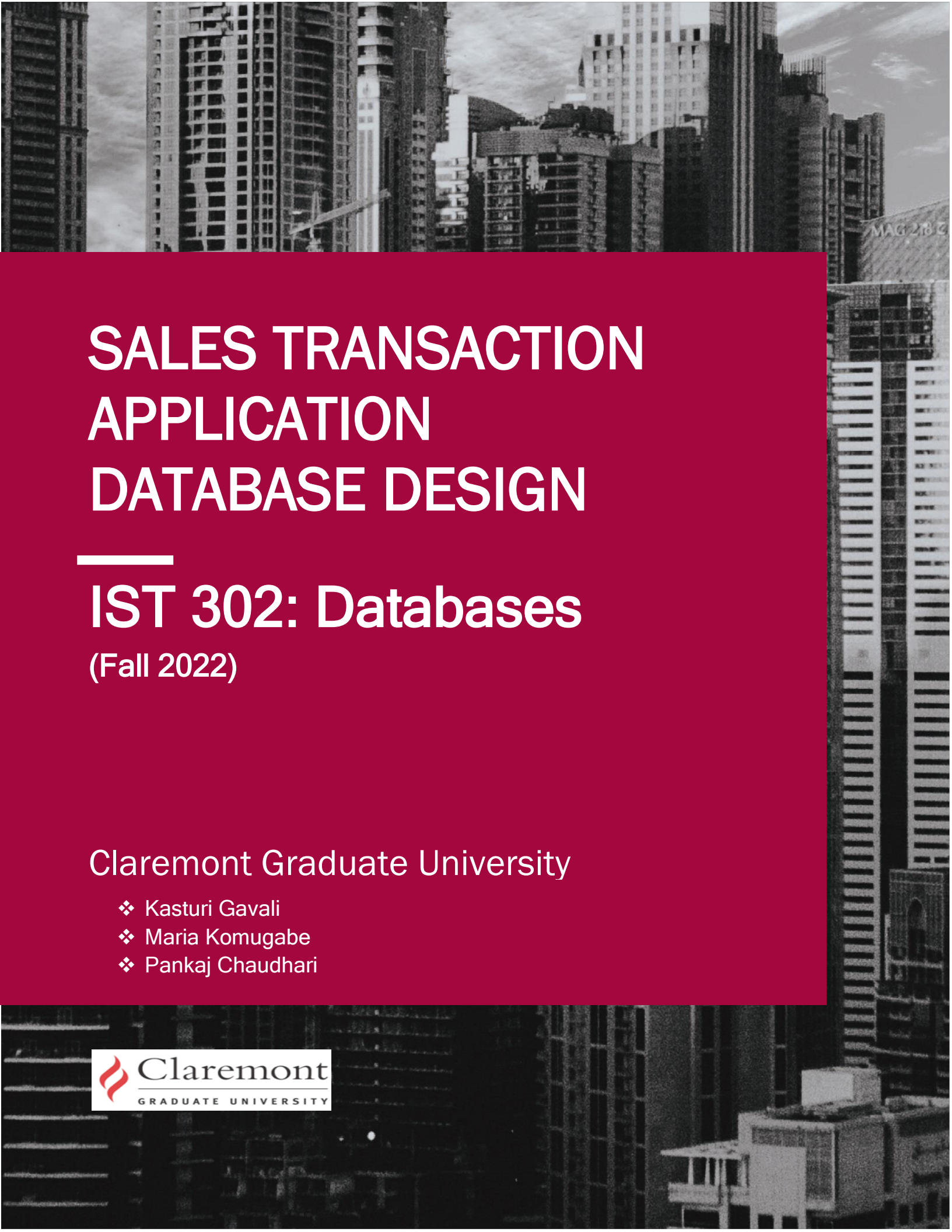


SALES TRANSACTION APPLICATION DATABASE DESIGN

IST 302: Databases

(Fall 2022)

Claremont Graduate University

- Kasturi Gavali
- Maria Komugabe
- Pankaj Chaudhari

Claremont GRADUATE UNIVERSITY

Executive Summary

In this project we have taken over an existing database to understand its structure and model to create a better and refined database as required. We have improvised and recreated our own database by looking at existing database and applying database optimization techniques like normalization and looking at execution plan for bottlenecks. The data have been migrated to new schema without any loss of functionality or loss of any data from source database.

In the coming sections, we have given a detailed business understanding, data understanding, logical understanding and a few more details about how we have taken over a complex database and made it easier for our users. The target users in our case are:

- Sales Executive
- Marketing Analysts
- Manager (upper management)

We have taken in consideration for our users and their needs and created the database views to give database access to the business entities in secured manner.

Table of Contents

Section 1: Business Understanding

Sales Transactions Applications

Understanding sales transaction data is essential for a business to thrive in today's fast-paced competitive market. Based on current trends the ecommerce sphere has become the main domain of sales for retail and is a growing domain for other branches of Point-of-Sale transactions [1]. Online sales transactions capture data about products sold, quantities, promotions, and customer information and store it in a database.

By having fully functional and optimized database would enable client to interpret and analyze the sales data better, understand buying patterns of customers, analyze their promotion campaigns etc. By analyzing the data systematically, will help clients to improve employees job performance and increase the company's bottom line profit. During the preliminary analysis of the data, we confirmed that this was indeed a fully online business, using credit card as the form of payment in the Sales transaction. Current database has the adequate information about customers, products, sales, and promotions. Considering this, we have identified key users for database and how they are going to use database is described in section below.

Users

Considering the business requirements, we have identified three hey database users listed below.

- Sales Executive
- Marketing Analyst
- Manager (Upper Management)

In addition to these key players, the database administrator and customers are also vital members in sales transaction application. The database administrator has full access to database to maintains its integrity, security, manage backup and recovery etc. The customer would have access to the external sales application.

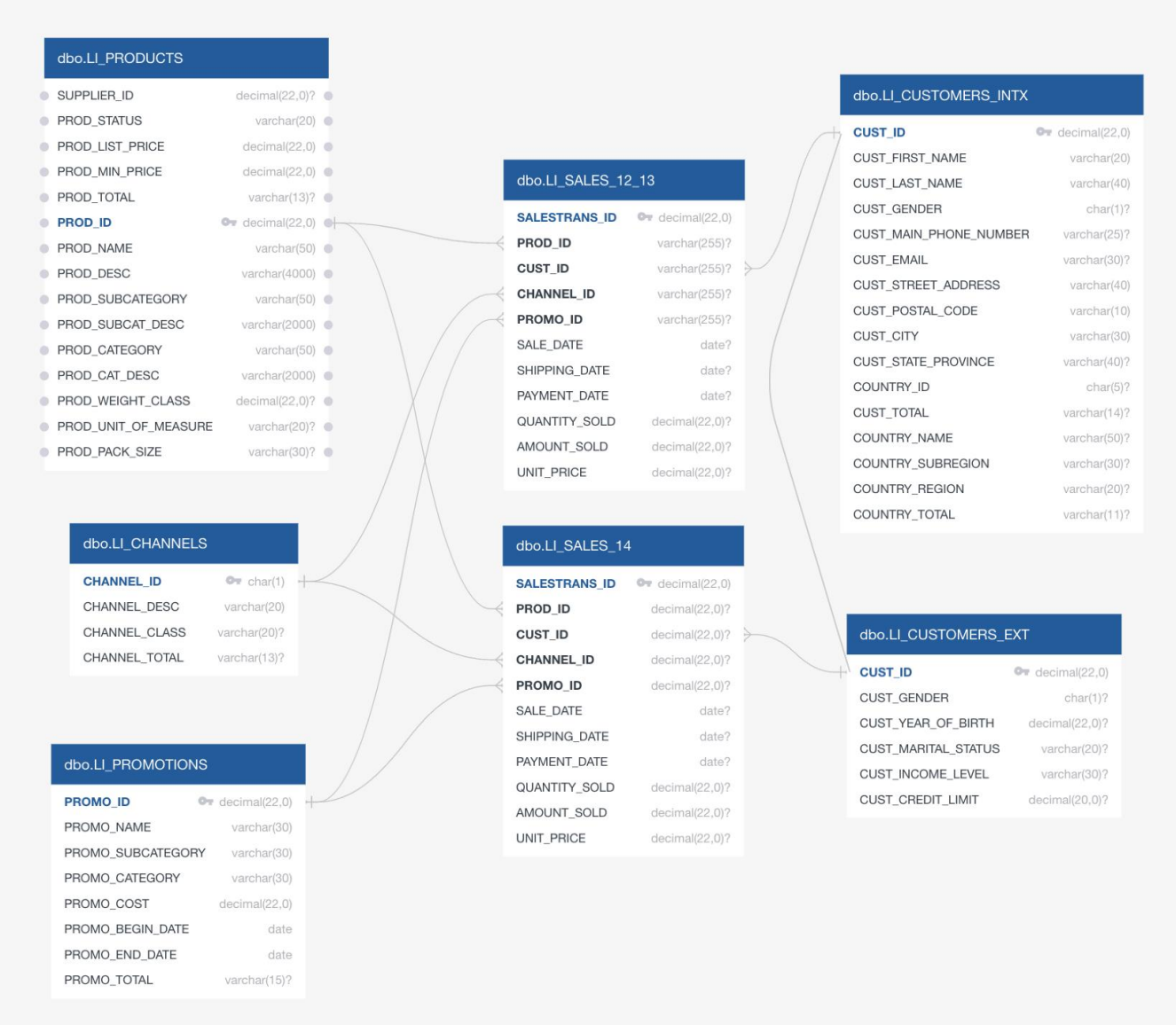
Identified key users have the following roles and would be interested in associated business queries are shown in table below.

Business View		
Users	**Description**	**Business Queries**
U01 (Sales)	Knowledge of product stock and past sales; including product costs, product sale price, product information, and most importantly, available promotions	• Sales revenue analysis (monthly, quarterly, yearly) • Find the most popular products sorted by Year. • Find out maximum number of days it takes to ship each product. • Understand the product purchase trend by customer genders, marital status, and geography.
U02 (Marketing)	Creates and retires promotions by analyzing sales, aiming to tailor promotions based on recorded and potential selling patterns	• View Promotions resulted in highest sales • Find out maximum number of days it takes to ship each product. • Understand the product purchase trend by customer genders, marital status, and geography.
U03 (Manager)	Analyzes and aims to optimize and incentivize performance of sales and job productivity	• Sales revenue analysis (monthly, quarterly, yearly and Region) • Find the most popular products sorted by Year. • View Promotions resulted in highest sales • View Most used channels in sales transaction. • Find out maximum number of days it takes to ship each product. • Understand the product purchase trend by customer genders, marital status, and geography.

Section 2: Data Understanding

Overview

Current state of LIY26 (based on appendix 1 of this document) schema is depicted in diagram below.



Findings

After carefully studying the current database in depth, we made numerous discoveries on how we can change current state of database to satisfy the business requirements mentioned in this document. This section describes each table including necessary modifications needed for each field of the table with appropriate justification.

LI_CHANNELS

Attribute	Data Type	Data Type Consistency	Useful	Low Cardinality	Functional Dependency
CHANNEL_ID	CHAR (1) NOT NULL	**No, it should be INT. All attribute values in this column are of type INT and this column is primary key.**	Yes		PK, CHANNEL_DESC, CHANNEL_CLASS
CHANNEL_DESC	VARCHAR (20) NOT NULL	Yes	Yes	Yes, 5	No
CHANNEL_CLASS	VARCHAR (20) NULL	Yes	Yes	Yes, 3	No
CHANNEL_TOTAL	VARCHAR (13) NULL	Yes	**No, content is the same for all rows**		No

LI_CUSTOMERS_EXT

Attribute	Data Type	Data Type Consistency	Useful	Low Cardinality	Functional Dependency
CUST_ID	DECIMAL (22,0), NOT NULL	**No, it should be INT. All attribute values in this column are of type INT and this column is primary key.**	Yes		CUST_ID, CUST_GENDER, CUST_YEAR_OF_BIRTH, CUST_MARITAL_STATUS, CUST_INCOME_LEVEL, CUST_CREDIT_LIMIT
CUST_GENDER	CHAR (1), NULL	Yes	Yes, but repetitive data	Yes, 2	No
CUST_YEAR_OF_BIRTH	DECIMAL (22,0), NULL	**No, it should be INT. All attribute values in this column are of type INT.**	Yes		No
CUST_MARITAL_STATUS	VARCHAR (20), NULL	Yes	Yes	Yes,11 but should be specified	No
CUST_INCOME_LEVEL	VARCHAR (30), NULL	Yes	**Yes, but this column needs to split, and need to include only income category. Separate table needs to be created defining upper and lower limit for each income level category.**	Yes, 13	No
CUST_CREDIT_LIMIT	DECIMAL (20,0), NULL	**No, it should be INT. All attribute values in this column are of type INT.**	Yes		No

LI_CUSTOMERS_INTX

Attribute	Data Type	Data Type Consistency	Useful	Low Cardinality	Functional Dependency
CUST_ID	Decimal (22, 0), NOT NULL	**No, it should be INT. All attribute values in this column are of type INT and this column is primary key.**	Yes		PK, CUST_FIRST_NAME, CUST_LAST_NAME, CUST_GENDER, CUST_MAIN_PHONE_NUMBER, CUST_EMAIL, CUST_STREET_ADDRESS, CUST_POSTAL_CODE, CUST_CITY, CUST_STATE_PROVINC
CUST_FIRST_NAME	Varchar (20), NOT NULL	Yes	Yes		No
CUST_LAST_NAME	Varchar (40), NOT NULL	Yes	Yes		No
CUST_GENDER	Char (1), NULL	Yes	Yes, but repetitive data	Yes, 2	No
CUST_MAIN_PHONE_NUMBER	Varchar (25), NULL	Yes	Yes		No
CUST_EMAIL	Varchar (30), NULL	Yes	Yes		No
CUST_STREET_ADDRESS	Varchar (40), NOT NULL	Yes	Yes		No
CUST_POSTAL_CODE	Varchar (10), NOT NULL	Yes	Yes		No
CUST_CITY	Varchar (30), NOT NULL	Yes	Yes		No
CUST_STATE_PROVINCE	Varchar (40), NULL	Yes	Yes		No
COUNTRY_ID	Char (5), NULL	**No, it should be INT. All attribute values in this column are of type INT.**	Yes		COUNTRY_NAME, COUNTRY_SUBREGION, COUNTRY_REGION, COUNTRY_TOTAL
CUST_TOTAL	Varchar (14), NULL	Yes	**No, same content for all rows.**		No

Attribute	Data Type	Data Type Consistency	Useful	Low Cardinality	Functional Dependency
COUNTRY_NAME	Varchar (50), NULL	Yes	Yes	Yes, 19	No
COUNTRY_SUBREGION	Varchar (30), NULL	Yes	Yes	Yes, 10	No
COUNTRY_REGION	Varchar (20), NULL	Yes	Yes	Yes, 5	No
COUNTRY_TOTAL	Varchar (11), NULL	Yes	**No, same content for all rows.**		No

LI_PRODUCTS

Attribute	Data Type	Data Type Consistency	Useful	Low Cardinality	Functional Dependency
SUPPLIER_ID	Decimal (22, 0), NULL	**No, it should be INT.**	**No, same content for all rows.**		
PROD_STATUS	Varchar (20), NOT NULL	Yes	**No, same content for all rows.**		No
PROD_LIST_PRICE	Decimal (22, 0), NOT NULL	Yes	Yes		No
PROD_MIN_PRICE	Decimal (22, 0), NOT NULL	Yes	Yes,		No
PROD_TOTAL	Varchar (13), NULL	Yes	**No, same content for all rows.**		No
PROD_ID	Decimal (22, 0), NOT NULL	**No, it should be INT. All attribute values in this column are of type INT and this column is primary key.**	Yes		PK, PROD_NAME, PROD_DESC, PROD_CATEGORY, PROD_WEIGHT_CLASS, PROD_UNIT_OF_MEASURE, PROD_PACK_SIZE, PROD_STATUS, PROD_LIST_PRICE, PROD_MIN_PRICE
PROD_NAME	Varchar (50), NOT NULL	Yes	Yes, two PROD_IDs can have the same PROD_NAME		No
PROD_DESC	Varchar (4000), NOT NULL	**No, length 4000 is too large for this column.**	Yes		No

Attribute	Data Type	Data Type Consistency	Useful	Low Cardinality	Functional Dependency
		Recommended column length is 1000.			
PROD_SUBCATEGORY	Varchar (50), NOT NULL	Yes	Yes	Yes, 21	No
PROD_SUBCAT_DESC	Varchar (2000), NOT NULL	**No, length 2000 is too large for this column. Recommended column length is 500.**	**No, duplicate data**		No
PROD_CATEGORY	Varchar (50), NOT NULL	Yes	Yes	Yes, 5	PROD_SUBCATEGORY
PROD_CAT_DESC	Varchar (2000), NOT NULL	**No, length 2000 is too large for this column. Recommended column length is 500.**	**No, duplicate data**		No
PROD_WEIGHT_CLASS	Decimal (22, 0), NULL	**No, should be int.**	**No**		No
PROD_UNIT_OF_MEASURE	Varchar (20), NULL	**No, should be char.**	**No, same content for all rows.**		No
PROD_PACK_SIZE	Varchar (30), NULL	**No, should be char.**	**No, same content for all rows.**		No

LI_PROMOTIONS

Attribute	Data Type	Data Type Consistency	Useful	Low Cardinality	Functional Dependency
PROMO_ID	Decimal (22, 0), NOT NULL	**No, it should be INT.**	Yes		PK, PROMO_NAME, PROMO_CATEGORY, PROMO_COST, PROMO_BEGIN_DATE, PROMO_END_DATE
PROMO_NAME	Varchar (30), NOT NULL	Yes	Yes		No
PROMO_SUBCATEGORY	Varchar (30), NOT NULL	Yes	Yes	Yes, 22	No

Attribute	Data Type	Data Type Consistency	Useful	Low Cardinality	Functional Dependency
PROMO_CATEGORY	Varchar (30), NOT NULL	Yes	Yes	Yes, 9	PROMO_SUBCATEGORY
PROMO_COST	Decimal (22, 0), NOT NULL	Yes	Yes		No
PROMO_BEGIN_DATE	Date, NOT NULL	Yes	Yes		No
PROMO_END_DATE	Date, NOT NULL	Yes	Yes		No
PROMO_TOTAL	Varchar (15), NULL	Yes	**No, same content for all rows**		No

LI_SALES_12_13

Attribute	Data Type	Data Type Consistency	Useful	Low Cardinality	Functional Dependency
SALESTRANS_ID	Decimal (22, 0), NOT NULL	**No, it should be INT. All attribute values in this column are of type INT and this column is primary key.**	Yes		PK, SALE_DATE, SHIPPING_DATE, PAYMENT_DATE, QUANTITY_SOLD, AMOUNT_SOLD, UNIT_PRICE
PROD_ID	Varchar (255), NULL	**No, it should be INT.**	Yes		No
CUST_ID	Varchar (255), NULL	**No, it should be INT.**	Yes		No
CHANNEL_ID	Varchar (255), NULL	**No, it should be INT.**	Yes		No
PROMO_ID	Varchar (255), NULL	**No, it should be INT.**	Yes		No
SALE_DATE	Date, NULL	Yes	Yes		No
SHIPPING_DATE	Date, NULL	Yes	Yes		No
PAYMENT_DATE	Date, NULL	Yes	Yes		No
QUANTITY_SOLD	Decimal (22, 0), NULL	**No, it should be INT.**	Yes		No
AMOUNT_SOLD	Decimal (22, 0), NULL	**No, it should allow 2 digits after decimal.**	Yes		No

Attribute	Data Type	Data Type Consistency	Useful	Low Cardinality	Functional Dependency
UNIT_PRICE	Decimal (22, 0), NULL	**No, it should allow 2 digits after decimal.**	Yes		No

LI_SALES_14

Attribute	Data Type	Data Type Consistency	Useful	Low Cardinality	Functional Dependency
SALESTRANS_ID	Decimal (22, 0), NOT NULL	**No, it should be INT. All attribute values in this column are of type INT and this column is primary key.**	Yes		PK, SALE_DATE, SHIPPING_DATE, PAYMENT_DATE, QUANTITY_SOLD, AMOUNT_SOLD, UNIT_PRICE
PROD_ID	Varchar (255), NULL	**No, it should be INT.**	Yes		No
CUST_ID	Varchar (255), NULL	**No, it should be INT.**	Yes		No
CHANNEL_ID	Varchar (255), NULL	**No, it should be INT.**	Yes		No
PROMO_ID	Varchar (255), NULL	**No, it should be INT.**	Yes		No
SALE_DATE	Date, NULL	Yes	Yes		No
SHIPPING_DATE	Date, NULL	Yes	Yes		No
PAYMENT_DATE	Date, NULL	Yes	Yes		No
QUANTITY_SOLD	Decimal (22, 0), NULL	**No, it should be INT.**	Yes		No
AMOUNT_SOLD	Decimal (22, 0), NULL	**No, it should allow 2 digits after decimal.**	Yes		No
UNIT_PRICE	Decimal (22, 0), NULL	**No, it should allow 2 digits after decimal.**	Yes		No

Summary

"LI_SALES_12_13" and "LI_SALES_14" tables represent same entity. Both tables have same column structure. Data from the table can be combined into single table. "LI_CUSTOMERS_EXT" and "LI_ CUSTOMERS_INTX" table represent customer information with different columns in each table. Both tables can be combined into single table by joining information based on "CUST_ID" column.

In summary, based on findings discovered, above mentioned tables need to be restructured in primary tables as listed below.

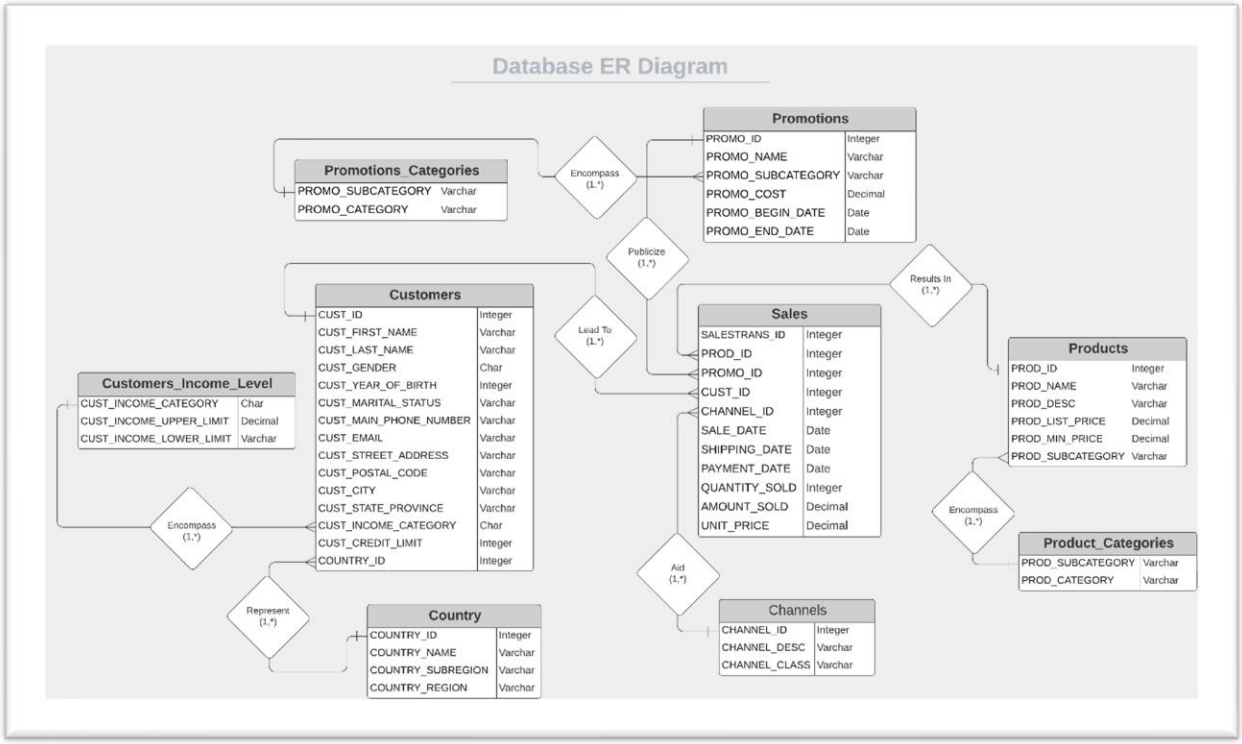
- Promotions
- Products
- Country
- Channel
- Customers
- Sales

Data from existing tables needs to reload into above table structure. Restructuring of tables allows us to make informed queries satisfying all business requirements.

Section 3: Conceptual Design

Overview

The conceptual database model is depicted in diagram below.



The conceptual model is defined as “provide concepts that are close to the way many users perceive data” by utilizing “entities, attributes and relationships”” (Elmasri & Navathe, 2017) [2]. The conceptual model provides general view of the database to be more understandable by the users. However, the conceptual model does not include the primary or foreign keys. The square shape represents the tables in the database and the diamond shape represent how the entity relates to one another. Furthermore, the one-to-many relationships represents the impact one has to the others. For example, the sales table has information about the sale’s date, shipping date, payment date, quantity sold and unit price. So, when products are sold, we need information about the product list price, minimum price, and product name. Thus, products need to be listed in the Sales table and is done by adding “Product_ID” to Sales table creating the one-to-many relation.

From observing the tables in the original database and the literature, the relationships are determined. “LI_CUSTOMERS_EXT” and “LI_CUSTOMERS_INTX” tables have combined, also the “LI_SALES_12_13” and “LI_SALES_14” have been combined. Those table share the same primary keys and relevant types of data for users. For example, the sales tables have the same columns but for different years. Countries information from table “LI_CUSTOMERS_INTX” is separated into “Country” table. Similarly, product categories and promotions categories are separated from tables “LI_PROMOTIONS” and “PRODUCTS” into

sperate tables. Customers income information also separated into another table by introducing new column “CUST_INCOME_CATEGORY” into “Customers” table.

Section 4: Security Requirements

To ensure data integrity and security we are using business views primarily for user access. Three users we identified in business understanding study will have “SELECT” privileges to appropriate views as defined in table below. The user we identified are high level business users for decision making purposes and not expected manipulate any data (“UPDATE”, “DELETE”, and “INSERT”) by themselves.

Business Users	**Business Views**				
	Revenue_Analysis_VW	*Popular_Products_VW*	*Products_Shipdate_VW*	*Sales_Promotions_VW*	*Customer_Trend_VW*
Manager (Upper Management)	X	X	X	X	X
Sales Executive	X*	X	X		X
Marketing Analyst			X	X	X

*: Restricted access to view for regions they have assigned to

Above table highlights the business view for Manager (Upper Management), Sales executive and Marketing analyst.

- “**Revenue_Analysis_VW**” view will be used by Manager to see sales revenue by month, year and region (by country). Average sales price will be calculated as (SUM(UNIT_PRICE)/SUM(QUANTITY_SOLD). This view will be used as sales executive to see sales revenue by month and year but only for their assigned region. This view wll be implemented as stored procedure which will accepts input parameter like Year, Month (Optional), and Region.
- “**Popular_Products_VW**” will be used by Manager and Sales executive to view products with highest sale sorted by Year.
- “**Products_Shipdate_VW**” will be used by Manager, Sales executive, and Marketing analyst to analyze maximum days to ship each product.
- “**Sales_Promotions_VW**” will be used by Manager and Marketing analyst to view promotions resulted in highest sale sorted by Year.
- “**Customer_Trend_VW**” will be used by Manager, Sales executive, and Marketing analyst to understand the product purchase trend by customer genders, marital status, and geography. This view will be implemented as stored procedure which will accept Customer Gender, Customer Marital Status and Region as an input parameter.

Section 5: Logical Design

Based on business understanding, findings discovered in existing "LIY26" database (Stated in section 2 of this document), and conceptual design, we have derived logical design as described below.

Tables with Design

CHANNELS

Attribute Name	Data Type	NULL/ NOT NULL	Primary Key Constraint	Referential Integrity Constraint	Other Constraints
CHANNEL_ID	Integer	NOT NULL	YES		
CHANNEL_DESC	Varchar (20)	NOT NULL			
CHANNEL_CLASS	Varchar (20)	NOT NULL			

CUSTOMERS

Attribute Name	Data Type	NULL/ NOT NULL	Primary Key Constraint	Referential Integrity Constraint	Other Constraints
CUST_ID	Integer	NOT NULL	YES		
CUST_FIRST_NAME	Varchar (20)	NOT NULL			
CUST_LAST_NAME	Varchar (20)	NOT NULL			
CUST_GENDER	Char (1)	NOT NULL			Check (Fixed set of attribute values)
CUST_YEAR_OF_BIRTH	Integer	NULL			
CUST_MARITAL_STATUS	Varchar (20)	NOT NULL			Check (Fixed set of attribute values)
CUST_MAIN_PHONE_NUMBER	Varchar (12)	NULL			
CUST_EMAIL	Varchar (30)	NULL			
CUST_STREET_ADDRESS	Varchar (40)	NOT NULL			
CUST_POSTAL_CODE	Varchar (10)	NOT NULL			
CUST_CITY	Varchar (30)	NOT NULL			
CUST_STATE_PROVINCE	Varchar (40)	NULL			
CUST_INCOME_CATEGORY	Char	NOT NULL		YES	
CUST_CREDIT_LIMIT	Integer	NOT NULL			
COUNTRY_ID	Integer	NOT NULL		YES	

CUSTOMERS_INCOME_LEVEL

Attribute Name	Data Type	NULL/ NOT NULL	Primary Key Constraint	Referential Integrity Constraint	Other Constraints
CUST_INCOME_CATEGORY	Char	NOT NULL	YES		
CUST_INCOME_LOWER_LIMIT	Decimal (22,0)	NOT NULL			
CUST_INCOME_UPPER_LIMIT	Decimal (22,0)	NULL			

CUSTOMERS_COUNTRY

Attribute Name	Data Type	NULL/ NOT NULL	Primary Key Constraint	Referential Integrity Constraint	Other Constraints
COUNTRY_ID	Char	NOT NULL	YES		
COUNTRY_NAME	Varchar (50)	NOT NULL			
COUNTRY_REGION	Varchar (20)	NOT NULL			Check (Fixed set of attribute values)
COUNTRY_SUBREGION	Varchar (30)	NOT NULL			Check (Fixed set of attribute values)

PRODUCTS

Attribute Name	Data Type	NULL/ NOT NULL	Primary Key Constraint	Referential Integrity Constraint	Other Constraints
PROD_ID	Integer	NOT NULL	YES		
PROD_NAME	Varchar (50)	NOT NULL			Unique (Product name must be unique)
PROD_DESC	Varchar (1000)	NOT NULL			
PROD_LIST_PRICE	Decimal (22, 0)	NOT NULL			
PROD_MIN_PRICE	Decimal (22, 0)	NOT NULL			Check (Min price must be less than list price)
PROD_SUBCATEGORY	Varchar (50)	NOT NULL		YES	

PRODUCTS_CATEGORIES

Attribute Name	Data Type	NULL/ NOT NULL	Primary Key Constraint	Referential Integrity Constraint	Other Constraints
PROD_SUBCATEGORY	Varchar (50)	NOT NULL	YES		
PROD_CATEGORY	Varchar (50)	NOT NULL			

PROMOTIONS

Attribute Name	Data Type	NULL/ NOT NULL	Primary Key Constraint	Referential Integrity Constraint	Other Constraints
PROMO_ID	Integer	NOT NULL	YES		
PROMO_NAME	Varchar (30)	NOT NULL			
PROMO_SUBCATEGORY	Varchar (30),	NOT NULL		YES	
PROMO_COST	Decimal (22, 0)	NOT NULL			
PROMO_BEGIN_DATE	Date,	NOT NULL			
PROMO_END_DATE	Date,	NOT NULL			Check (Promo end date must be equal to or greater than Promo begin date)

PROMOTIONS_CATEGORIES

Attribute Name	Data Type	NULL/ NOT NULL	Primary Key Constraint	Referential Integrity Constraint	Other Constraints
PROMO_SUBCATEGORY	Varchar (30)	NOT NULL	YES		
PROMO_CATEGORY	Varchar (30)	NOT NULL			

SALES

Attribute Name	Data Type	NULL/ NOT NULL	Primary Key Constraint	Referential Integrity Constraint	Other Constraints
SALESTRANS_ID	Integer	NOT NULL	YES		
PROD_ID	Integer	NOT NULL		YES	
PROMO_ID	Integer	NOT NULL		YES	
CUST_ID	Integer	NOT NULL		YES	
CHANNEL_ID	Integer	NOT NULL		YES	
SALE_DATE	Date	NOT NULL			
SHIPPING_DATE	Date	NOT NULL			Check (Should be greater than or equal to sales date)
PAYMENT_DATE	Date	NOT NULL			Check (Should be greater than or equal to sales date)
QUANTITY_SOLD	Integer	NOT NULL			Check (Should be greater zero)
AMOUNT_SOLD	Decimal (22, 2)	NOT NULL			Check (Should be greater zero)
UNIT_PRICE	Decimal (22, 2)	NOT NULL			Check (Should be greater zero)

Section 6: Physical Design

Based on logical design described in section 5, we have defined primary key constraint, referential integrity constraint and other constraints and created the physical design as described below.

DDL Scripts

CREATE CHANNELS TABLE AND ADD CONSTRAINT

```
CREATE TABLE dbo.CHANNELS (
	CHANNEL_ID INT NOT NULL,
	CHANNEL_DESC VARCHAR (20) NOT NULL,
	CHANNEL_CLASS VARCHAR (20) NOT NULL
)
GO

ALTER TABLE CHANNELS Add CONSTRAINT PK_CHANNEL_ID PRIMARY KEY (CHANNEL_ID)
GO
```

CREATE CUSTOMERS TABLE

```
CREATE TABLE dbo.CUSTOMERS (
	CUST_ID INT NOT NULL,
	CUST_FIRST_NAME Varchar(20) NOT NULL,
	CUST_LAST_NAME Varchar(20) NOT NULL,
	CUST_GENDER   Char(1) NOT NULL,
	CUST_YEAR_OF_BIRTH   INT NULL,
	CUST_MARITAL_STATUS  Varchar(20)   NOT NULL,
	CUST_MAIN_PHONE_NUMBER      Varchar(20)   NULL,
	CUST_EMAIL    Varchar(30)   NULL,
	CUST_STREET_ADDRESS  Varchar(40)   NOT NULL,
	CUST_POSTAL_CODE     Varchar(10)   NOT NULL,
	CUST_CITY     Varchar(30)   NOT NULL,
	CUST_STATE_PROVINCE  Varchar(40)   NULL,
	CUST_INCOME_CATEGORY Char   NOT NULL,
	CUST_CREDIT_LIMIT    INT    NOT NULL,
	COUNTRY_ID    INT    NOT NULL,
)
GO
```

CREATE CUSTOMERS_COUNTRY TABLE

```
CREATE TABLE dbo.CUSTOMERS_COUNTRY (
	COUNTRY_ID    INT    NOT NULL,
	COUNTRY_NAME  Varchar(50)   NOT NULL,
	COUNTRY_REGION       Varchar(20)   NOT NULL,
	COUNTRY_SUBREGION    Varchar(30)   NOT NULL,
)
GO
```

CREATE CUSTOMERS_INCOME_LEVEL TABLE

```
CREATE TABLE dbo.CUSTOMERS_INCOME_LEVEL(
	CUST_INCOME_CATEGORY Char   NOT NULL,
```

```
	CUST_INCOME_LOWER_LIMIT	Decimal(22,0) NOT NULL,
	CUST_INCOME_UPPER_LIMIT	Decimal(22,0) NULL
)
GO
```

ADD CONSTRAINTS ON CUSTOMERS_COUNTRY, CUSTOMERS_INCOME_LEVEL AND CUSTOMERS TABLES

```
/*
Add additional constraints on CUSTOMERS_COUNTRY, CUSTOMERS_INCOME_LEVEL and CUSTOMERS table
*/
ALTER TABLE dbo.CUSTOMERS_COUNTRY ADD CONSTRAINT PK_COUNTRY_ID PRIMARY KEY (COUNTRY_ID)
GO

ALTER TABLE dbo.CUSTOMERS_COUNTRY ADD CONSTRAINT CK_COUNTRY_REGION CHECK (COUNTRY_REGION IN
('Oceania', 'Europe', 'Americas', 'Africa', 'Asia'))
GO

ALTER TABLE CUSTOMERS_COUNTRY ADD CONSTRAINT CK_COUNTRY_SUBREGION CHECK (COUNTRY_SUBREGION IN
('East Asia', 'West Africa', 'North Asia', 'Southern America', 'SouthWestern Europe',
'NorthWestern Europe', 'South Asia', 'Southern Africa', 'Northern America', 'Australia'))
GO

ALTER TABLE dbo.CUSTOMERS_INCOME_LEVEL ADD CONSTRAINT PK_CUST_INCOME_CATEGORY PRIMARY KEY
(CUST_INCOME_CATEGORY)
GO

ALTER TABLE dbo.CUSTOMERS ADD CONSTRAINT PK_CUST_ID PRIMARY KEY (CUST_ID)
GO

ALTER TABLE dbo.CUSTOMERS ADD CONSTRAINT FK_COUNTRY_ID FOREIGN KEY (COUNTRY_ID) REFERENCES
CUSTOMERS_COUNTRY (COUNTRY_ID)
GO

ALTER TABLE dbo.CUSTOMERS ADD CONSTRAINT FK_CUST_INCOME_CATEGORY FOREIGN KEY
(CUST_INCOME_CATEGORY) REFERENCES CUSTOMERS_INCOME_LEVEL (CUST_INCOME_CATEGORY)
GO

ALTER TABLE dbo.CUSTOMERS ADD CONSTRAINT CK_CUST_GENDER CHECK (CUST_GENDER IN ('F','M'))
GO

ALTER TABLE dbo.CUSTOMERS ADD CONSTRAINT CK_CUST_MARITAL_STATUS CHECK (CUST_MARITAL_STATUS IN
('Single' , 'Married' , 'Divorced', 'Widowed','Never married' ,'Prefer not to say'))
GO
```

CREATE PRODUCTS TABLE

```
CREATE TABLE dbo.PRODUCTS (
	PROD_ID INT	NOT NULL,
	PROD_NAME	Varchar(50) NOT NULL,
	PROD_DESC	Varchar(1000) NOT NULL,
	PROD_LIST_PRICE	Decimal(22, 0)	NOT NULL,
	PROD_MIN_PRICE	Decimal(22, 0)	NOT NULL,
	PROD_SUBCATEGORY	Varchar(50)	NOT NULL
)
GO
```

CREATE PRODUCTS_CATEGORIES TABLE

```
CREATE TABLE dbo.PRODUCTS_CATEGORIES (
	PROD_SUBCATEGORY	Varchar(50)	NOT NULL,
	PROD_CATEGORY Varchar(50)	NOT NULL
)
GO
```

ADD CONSTRAINTS ON PRODUCTS_CATEGORIES AND PRODUCTS TABLES

```
/*
Add constraints on PRODUCTS and PRODUCTS_CATEGORIES table
*/

ALTER TABLE dbo.PRODUCTS ADD CONSTRAINT PK_PROD_ID PRIMARY KEY (PROD_ID)
GO

ALTER TABLE dbo.PRODUCTS ADD CONSTRAINT UQ_PROD_NAME UNIQUE (PROD_NAME)
GO

ALTER TABLE dbo.PRODUCTS_CATEGORIES ADD CONSTRAINT PK_PROD_SUBCATEGORY PRIMARY KEY
(PROD_SUBCATEGORY)
GO

ALTER TABLE dbo.PRODUCTS ADD CONSTRAINT FK_PROD_SUBCATEGORY FOREIGN KEY (PROD_SUBCATEGORY)
REFERENCES PRODUCTS_CATEGORIES (PROD_SUBCATEGORY)
GO

ALTER TABLE dbo.PRODUCTS ADD CONSTRAINT CK_MIN_PRICE CHECK (PROD_MIN_PRICE <= PROD_LIST_PRICE)
GO
```

CREATE PROMOTIONS TABLE

```
CREATE TABLE dbo.PROMOTIONS (
	PROMO_ID	INT	NOT NULL,
	PROMO_NAME	Varchar(30)	NOT NULL,
	PROMO_SUBCATEGORY	Varchar(30)	NOT NULL,
	PROMO_COST	Decimal(22, 0)	NOT NULL,
	PROMO_BEGIN_DATE	Date	NOT NULL,
	PROMO_END_DATE	Date	NOT NULL
)
GO
```

CREATE PROMOTIONS_CATEGORIES TABLE

```
CREATE TABLE dbo.PROMOTIONS_CATEGORIES (
	PROMO_SUBCATEGORY	Varchar(30)	NOT NULL,
	PROMO_CATEGORY	Varchar(30)	NOT NULL
)
GO
```

ADD CONSTRAINTS ON PROMOTIONS_CATEGORIES AND PROMOTIONS TABLES

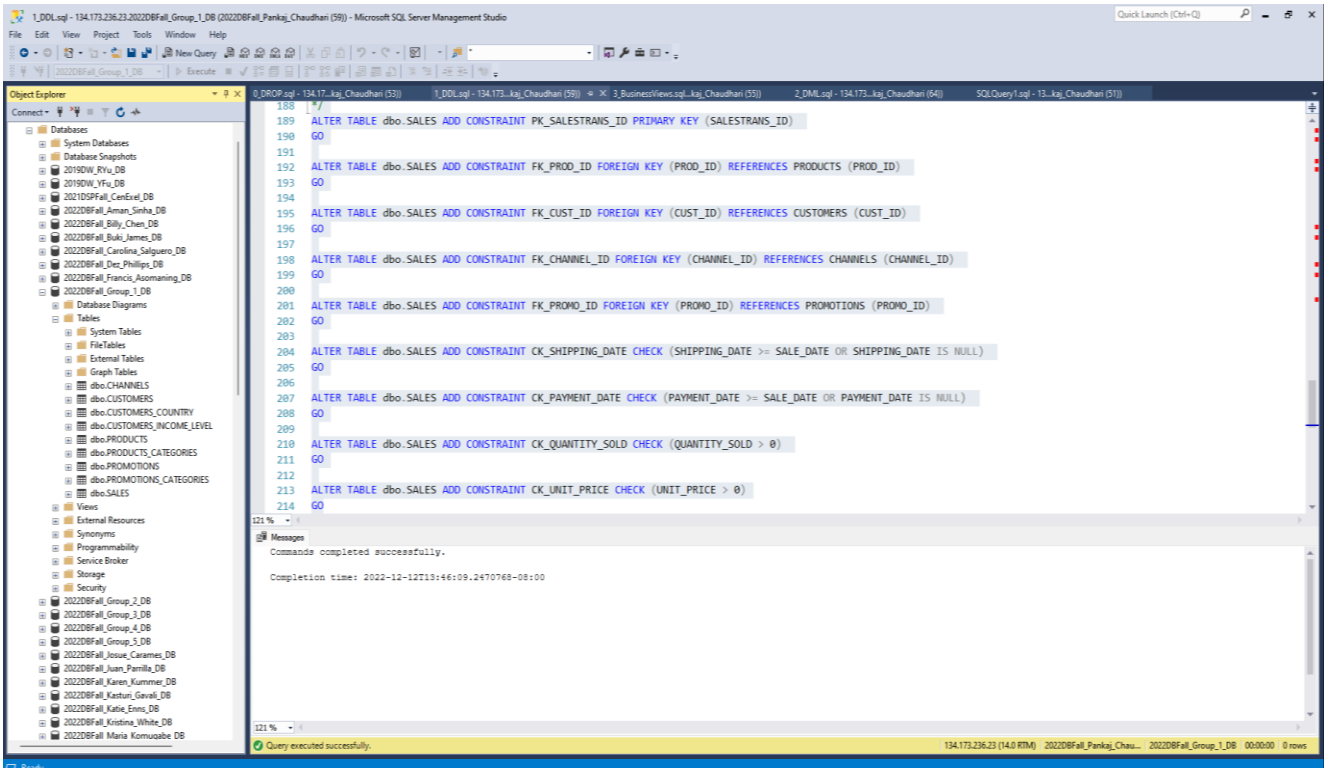
```
/*
Add constraints on PROMOTIONS and PROMOTIONS_CATEGORIES tables
*/
ALTER TABLE dbo.PROMOTIONS ADD CONSTRAINT PK_PROMO_ID PRIMARY KEY (PROMO_ID)
GO

ALTER TABLE dbo.PROMOTIONS_CATEGORIES ADD CONSTRAINT PK_PROMO_SUBCATEGORY PRIMARY KEY
(PROMO_SUBCATEGORY)
GO

ALTER TABLE dbo.PROMOTIONS ADD CONSTRAINT FK_PROMO_SUBCATEGORY_ID FOREIGN KEY
(PROMO_SUBCATEGORY) REFERENCES PROMOTIONS_CATEGORIES (PROMO_SUBCATEGORY)
GO

ALTER TABLE dbo.PROMOTIONS ADD CONSTRAINT CK_PROMO_END_DATE CHECK
(PROMO_END_DATE>=PROMO_BEGIN_DATE)
GO
```

CREATE SALES TABLE AND ADD CONSTRAINTS

```
CREATE TABLE dbo.SALES (
       SALESTRANS_ID  INT   NOT NULL,
       PROD_ID       INT    NOT NULL,
       PROMO_ID      INT    NOT NULL,
       CUST_ID       INT    NOT NULL,
       CHANNEL_ID    INT    NOT NULL,
       SALE_DATE Date       NOT NULL,
       SHIPPING_DATE Date   NOT NULL,
       PAYMENT_DATE Date    NOT NULL,
       QUANTITY_SOLD INT NOT NULL,
       AMOUNT_SOLD Decimal(22, 2)  NOT NULL,
       UNIT_PRICE    Decimal(22, 2)       NOT NULL
)
GO

/*
Add constraints on SALES tables
*/
ALTER TABLE dbo.SALES ADD CONSTRAINT PK_SALESTRANS_ID PRIMARY KEY (SALESTRANS_ID)
GO

ALTER TABLE dbo.SALES ADD CONSTRAINT FK_PROD_ID FOREIGN KEY (PROD_ID) REFERENCES PRODUCTS
(PROD_ID)
GO

ALTER TABLE dbo.SALES ADD CONSTRAINT FK_CUST_ID FOREIGN KEY (CUST_ID) REFERENCES CUSTOMERS
(CUST_ID)
GO

ALTER TABLE dbo.SALES ADD CONSTRAINT FK_CHANNEL_ID FOREIGN KEY (CHANNEL_ID) REFERENCES CHANNELS
(CHANNEL_ID)
GO

ALTER TABLE dbo.SALES ADD CONSTRAINT FK_PROMO_ID FOREIGN KEY (PROMO_ID) REFERENCES PROMOTIONS
(PROMO_ID)
GO

ALTER TABLE dbo.SALES ADD CONSTRAINT CK_SHIPPING_DATE CHECK (SHIPPING_DATE >= SALE_DATE)
GO
```

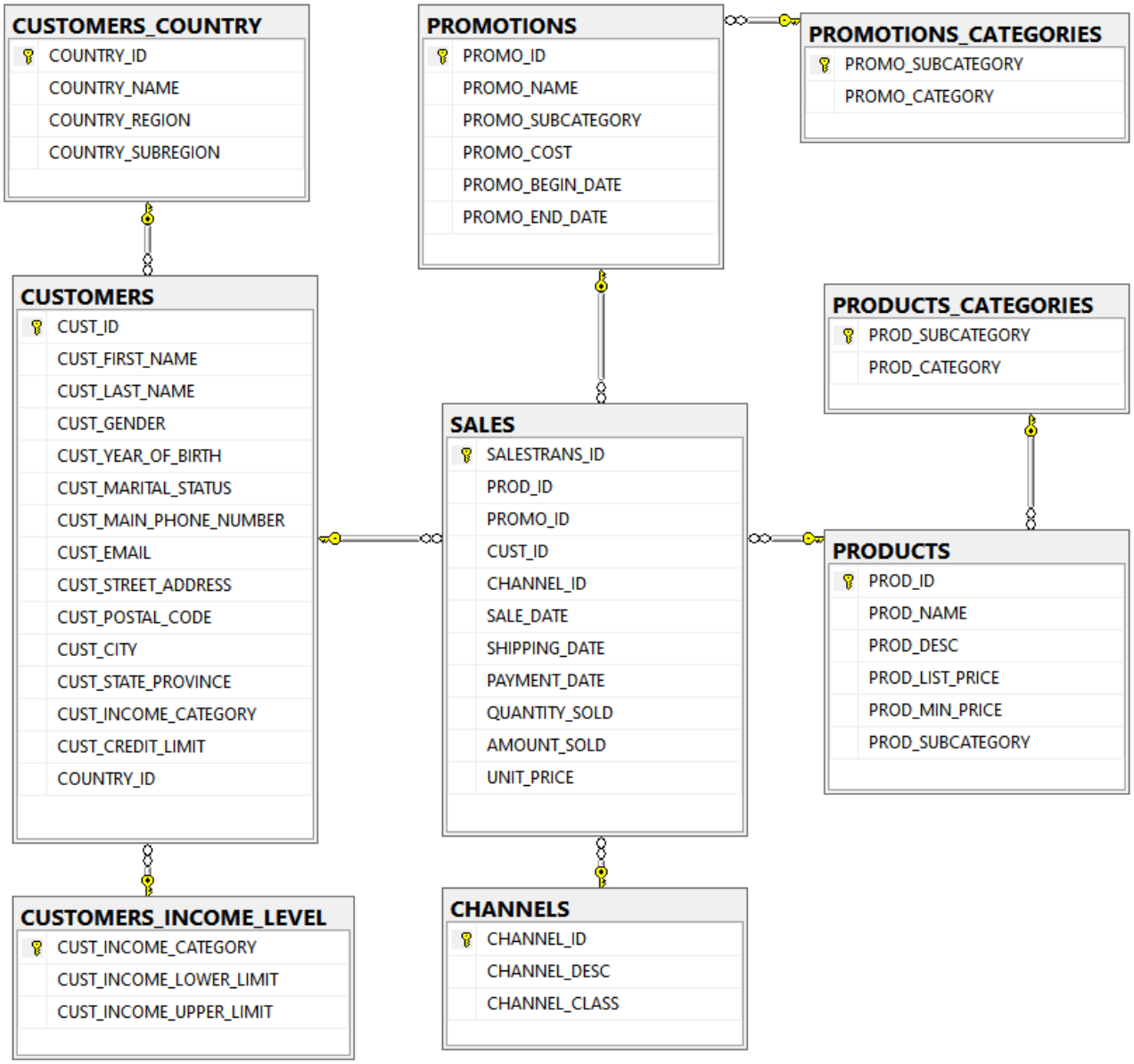
```
ALTER TABLE dbo.SALES ADD CONSTRAINT CK_PAYMENT_DATE CHECK (PAYMENT_DATE >= SALE_DATE)
GO

ALTER TABLE dbo.SALES ADD CONSTRAINT CK_QUANTITY_SOLD CHECK (QUANTITY_SOLD > 0)
GO

ALTER TABLE dbo.SALES ADD CONSTRAINT CK_AMOUNT_SOLD CHECK (AMOUNT_SOLD > 0)
GO

ALTER TABLE dbo.SALES ADD CONSTRAINT CK_UNIT_PRICE CHECK (UNIT_PRICE > 0)
GO
```

We have clubbed all above DDL scripts into single file and executed file. Below is screenshot for successful execution of DDL scripts.



Database Model

After creating all tables and constraints mentioned in DDL statements, below is our final database schema model. Diagram below is generated from SQL Server Management Studio Database Diagrams section.



Section 7: Database Implementation

With the tables and constraints created it we will now populate the tables with the existing records. This section contains all the DML scripts needed to populate the tables, along with integrity constraint tests, access structure, indexes, views, and stored procedures.

DML Scripts for Populating Data

POPULATE CHANNELS TABLE

```
INSERT INTO dbo.CHANNELS SELECT CHANNEL_ID ,CHANNEL_DESC, CHANNEL_CLASS FROM
LIY26.dbo.LI_CHANNELS
GO
```

POPULATE CUSTOMERS_COUNTRY TABLE

```
INSERT INTO dbo.CUSTOMERS_COUNTRY SELECT DISTINCT(COUNTRY_ID), COUNTRY_NAME, COUNTRY_REGION,
COUNTRY_SUBREGION FROM LIY26.dbo.LI_CUSTOMERS_INTX
GO
```

POPULATE CUSTOMERS_INCOME_LEVEL TABLE

```
INSERT INTO dbo.CUSTOMERS_INCOME_LEVEL (CUST_INCOME_CATEGORY, CUST_INCOME_LOWER_LIMIT,
CUST_INCOME_UPPER_LIMIT) VALUES ('G',130.000,149.999)
INSERT INTO dbo.CUSTOMERS_INCOME_LEVEL (CUST_INCOME_CATEGORY, CUST_INCOME_LOWER_LIMIT,
CUST_INCOME_UPPER_LIMIT) VALUES ('A',0.000,30.000)
INSERT INTO dbo.CUSTOMERS_INCOME_LEVEL (CUST_INCOME_CATEGORY, CUST_INCOME_LOWER_LIMIT,
CUST_INCOME_UPPER_LIMIT) VALUES ('J',190.000,249.999)
INSERT INTO dbo.CUSTOMERS_INCOME_LEVEL (CUST_INCOME_CATEGORY, CUST_INCOME_LOWER_LIMIT,
CUST_INCOME_UPPER_LIMIT) VALUES ('K',250.000,299.999)
INSERT INTO dbo.CUSTOMERS_INCOME_LEVEL (CUST_INCOME_CATEGORY, CUST_INCOME_LOWER_LIMIT,
CUST_INCOME_UPPER_LIMIT) VALUES ('L',300.000,NULL)
INSERT INTO dbo.CUSTOMERS_INCOME_LEVEL (CUST_INCOME_CATEGORY, CUST_INCOME_LOWER_LIMIT,
CUST_INCOME_UPPER_LIMIT) VALUES ('F',110.000,129.999)
INSERT INTO dbo.CUSTOMERS_INCOME_LEVEL (CUST_INCOME_CATEGORY, CUST_INCOME_LOWER_LIMIT,
CUST_INCOME_UPPER_LIMIT) VALUES ('D',70.000,89.999)
INSERT INTO dbo.CUSTOMERS_INCOME_LEVEL (CUST_INCOME_CATEGORY, CUST_INCOME_LOWER_LIMIT,
CUST_INCOME_UPPER_LIMIT) VALUES ('E',90.000,109.999)
INSERT INTO dbo.CUSTOMERS_INCOME_LEVEL (CUST_INCOME_CATEGORY, CUST_INCOME_LOWER_LIMIT,
CUST_INCOME_UPPER_LIMIT) VALUES ('C',50.000,69.999)
INSERT INTO dbo.CUSTOMERS_INCOME_LEVEL (CUST_INCOME_CATEGORY, CUST_INCOME_LOWER_LIMIT,
CUST_INCOME_UPPER_LIMIT) VALUES ('I',170.000,189.999)
INSERT INTO dbo.CUSTOMERS_INCOME_LEVEL (CUST_INCOME_CATEGORY, CUST_INCOME_LOWER_LIMIT,
CUST_INCOME_UPPER_LIMIT) VALUES ('B',30.000,49.999)
INSERT INTO dbo.CUSTOMERS_INCOME_LEVEL (CUST_INCOME_CATEGORY, CUST_INCOME_LOWER_LIMIT,
CUST_INCOME_UPPER_LIMIT) VALUES ('H',150.000,169.999)
INSERT INTO dbo.CUSTOMERS_INCOME_LEVEL (CUST_INCOME_CATEGORY, CUST_INCOME_LOWER_LIMIT,
CUST_INCOME_UPPER_LIMIT) VALUES ('M',0.000,NULL)
GO
```

POPULATE CUSTOMERS TABLE

```
INSERT INTO dbo.CUSTOMERS SELECT e.CUST_ID,i.CUST_FIRST_NAME ,i.CUST_LAST_NAME
,i.CUST_GENDER,e.CUST_YEAR_OF_BIRTH ,
CASE
              WHEN e.CUST_MARITAL_STATUS IN ('married','Mar-AF')  THEN 'Married'
              WHEN e.CUST_MARITAL_STATUS IN ('single','Separ.') THEN 'Single'
              WHEN e.CUST_MARITAL_STATUS IN ('Widowed','widow') THEN 'Widowed'
              WHEN e.CUST_MARITAL_STATUS IN ('Divorc.','divorced') THEN 'Divorced'
              WHEN e.CUST_MARITAL_STATUS IN ('NeverM') THEN 'Never married'
              ELSE 'Prefer not to say'
END AS CUST_MARITAL_STATUS,
i.CUST_MAIN_PHONE_NUMBER,i.CUST_EMAIL,i.CUST_STREET_ADDRESS,
i.CUST_POSTAL_CODE,i.CUST_CITY,i.CUST_STATE_PROVINCE,
CASE
       WHEN e.CUST_INCOME_LEVEL = '' THEN 'M'
       ELSE SUBSTRING(e.CUST_INCOME_LEVEL, 2,1)
END AS CUST_INCOME_CATEGORY,
e.CUST_CREDIT_LIMIT ,i.COUNTRY_ID
FROM LIY26.dbo.LI_CUSTOMERS_INTX as i
       JOIN LIY26.dbo.LI_CUSTOMERS_EXT AS e
              ON i.CUST_ID = e.CUST_ID
GO
```

POPULATE PRODUCTS_CATEGORIES TABLE

```
INSERT INTO dbo.PRODUCTS_CATEGORIES
SELECT
DISTINCT(CASE
       WHEN PROD_CATEGORY = 'Software/Other' AND PROD_SUBCATEGORY = 'Accessories' THEN
'Accessories-Software'
       ELSE PROD_SUBCATEGORY
END) as PROD_SUBCATEGORY,
PROD_CATEGORY
FROM LIY26.dbo.LI_PRODUCTS
GO
```

POPULATE PRODUCTS TABLE

```
INSERT INTO dbo.PRODUCTS
SELECT PROD_ID,
(CASE
       WHEN PROD_ID = 123 THEN '"DVD-R Discs, 4.7GB, Pack of 5 - V2"'
       ELSE PROD_NAME
END) as PROD_NAME,
PROD_DESC, PROD_LIST_PRICE, PROD_MIN_PRICE, PROD_SUBCATEGORY FROM LIY26.dbo.LI_PRODUCTS
GO
```

POPULATE PROMOTIONS_CATEGORIES TABLE

```
INSERT INTO dbo.PROMOTIONS_CATEGORIES SELECT DISTINCT(PROMO_SUBCATEGORY), PROMO_CATEGORY FROM
LIY26.DBO.LI_PROMOTIONS
GO
```

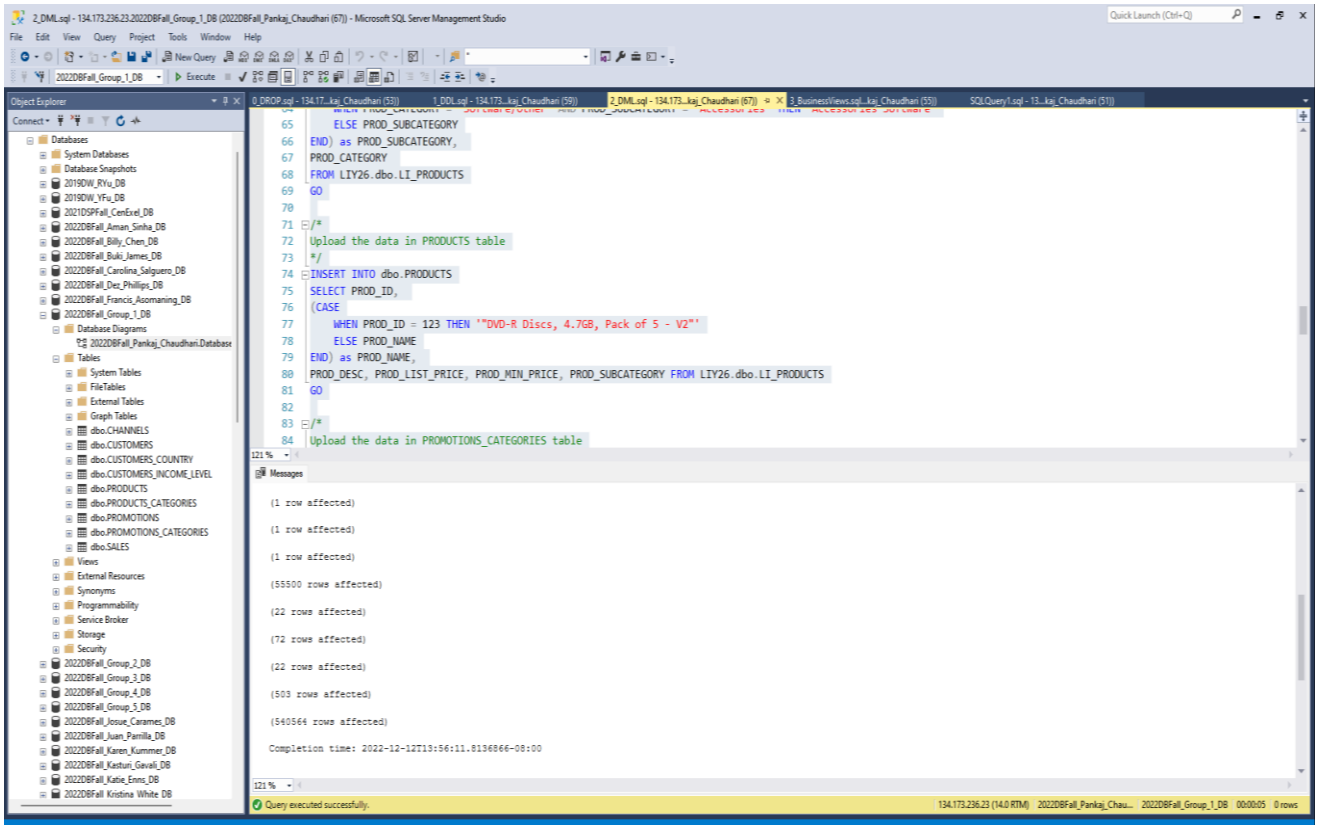
POPULATE PROMOTIONS TABLE

```
INSERT INTO dbo.PROMOTIONS SELECT PROMO_ID, PROMO_NAME, PROMO_SUBCATEGORY, PROMO_COST,
PROMO_BEGIN_DATE, PROMO_END_DATE FROM LIY26.DBO.LI_PROMOTIONS
GO
```

POPULATE SALES TABLE

```
INSERT INTO dbo.SALES
SELECT SALESTRANS_ID, PROD_ID, PROMO_ID, CUST_ID, CHANNEL_ID, SALE_DATE, SHIPPING_DATE,
PAYMENT_DATE, QUANTITY_SOLD, AMOUNT_SOLD, UNIT_PRICE FROM LIY26.dbo.LI_SALES_12_13
UNION
SELECT SALESTRANS_ID, PROD_ID, PROMO_ID, CUST_ID, CHANNEL_ID, SALE_DATE, SHIPPING_DATE,
PAYMENT_DATE, QUANTITY_SOLD, AMOUNT_SOLD, UNIT_PRICE FROM LIY26.dbo.LI_SALES_14
GO
```

We have clubbed all above DML scripts into single file and executed file. Below is screenshot for successful execution of DML scripts.

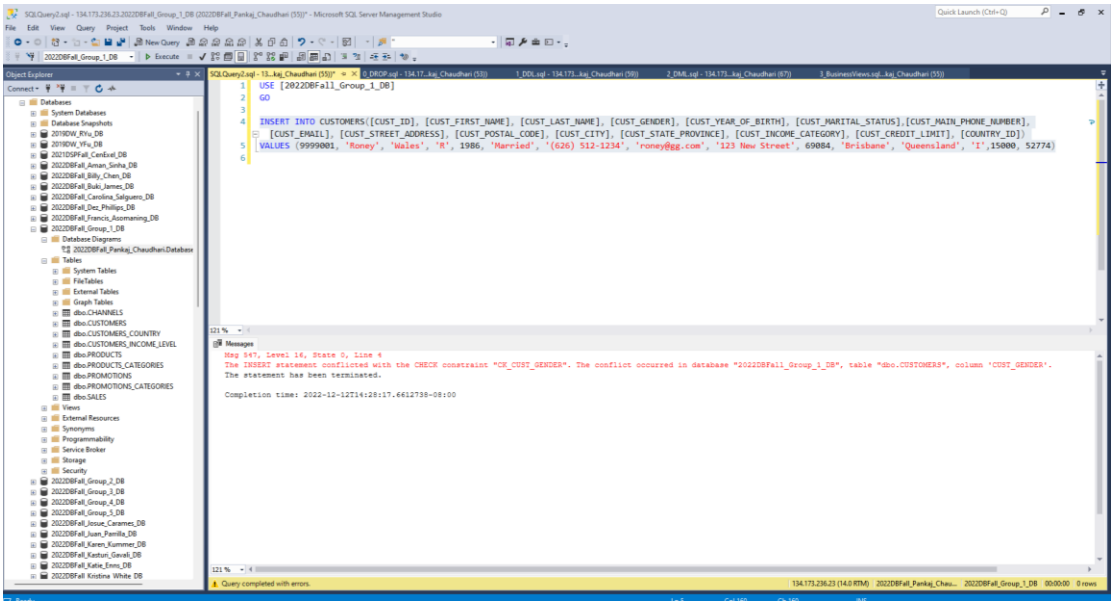


Test Integrity Constraints

We have created primary key constraint on every table and foreign key constraint on the referential tables. Noting down the attributes with low cardinality in few tables we have added check constraints on them. Following are the statements to test only some of the integrity constraints.

CONSTRAINT CHECK ON CUSTOMERS TABLE

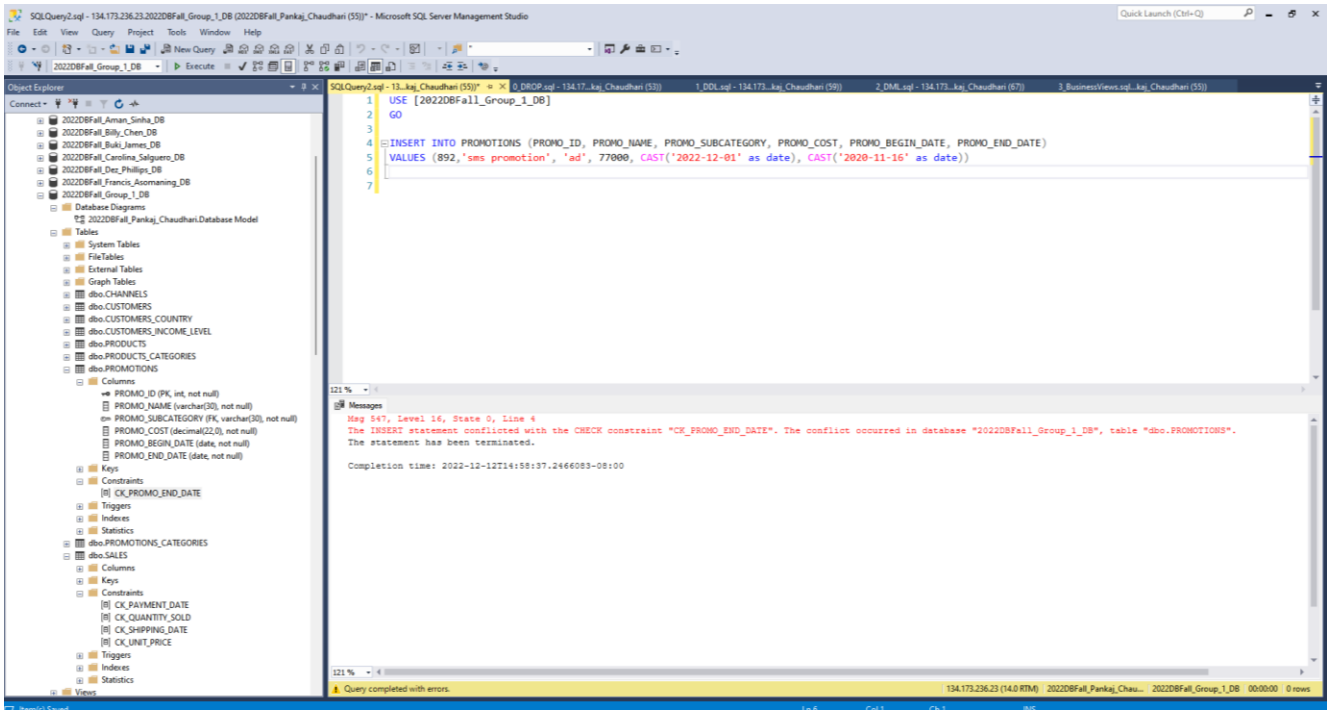
```
The CUSTOMERS table has integrity constraint as:
--1. CUST_GENDER = F, M or NULL

SELECT * FROM CUSTOMERS

--This is a Positive check:
INSERT INTO CUSTOMERS([CUST_ID], [ CUST_FIRST_NAME], [CUST_LAST_NAME], [CUST_GENDER],
[CUST_YEAR_OF_BIRTH], [CUST_MARITAL_STATUS],[ CUST_MAIN_PHONE_NUMBER], [ CUST_EMAIL],
[CUST_STREET_ADDRESS], [CUST_POSTAL_CODE], [CUST_CITY], [CUST_STATE_PROVINCE],
[CUST_INCOME_CATEGORY], [CUST_CREDIT_LIMIT], [COUNTRY_ID)
VALUES (9999001, 'Roney', 'Wales', 'M', 1986, 'Married', '(626) 512-1234', 'roney@gg.com',
'123 New Street', 69084, 'Brisbane', 'Queensland', "I"',15000, 52774)


--This is a Negative check:
INSERT INTO CUSTOMERS([CUST_ID], [CUST_FIRST_NAME], [CUST_LAST_NAME], [CUST_GENDER],
[CUST_YEAR_OF_BIRTH], [CUST_MARITAL_STATUS],[CUST_MAIN_PHONE_NUMBER], [CUST_EMAIL],
[CUST_STREET_ADDRESS], [CUST_POSTAL_CODE], [CUST_CITY], [CUST_STATE_PROVINCE],
[CUST_INCOME_CATEGORY], [CUST_CREDIT_LIMIT], [COUNTRY_ID])
VALUES (9999001, 'Roney', 'Wales', 'R', 1986, 'Married', '(626) 512-1234', 'roney@gg.com', '123
New Street', 69084, 'Brisbane', 'Queensland', 'I',15000, 52774)
```



CONSTRAINT CHECK ON PROMOTIONS TABLE

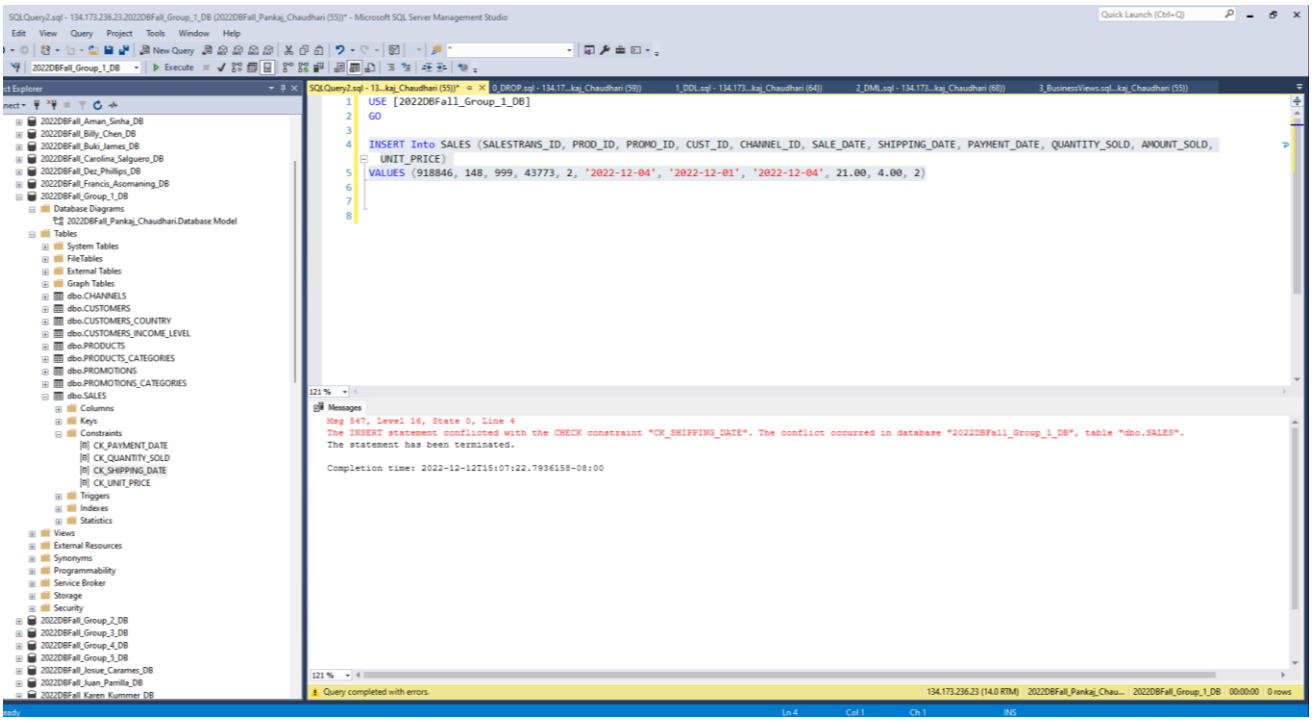
```
-- First table is PROMOTIONS where it has two integrity constraints:
-- 1. PROMO_END_DATE must be NULL or >= promo_begin_date

SELECT * FROM PROMOTIONS

-- This is a Positive check:
INSERT INTO PROMOTIONS (PROMO_ID, PROMO_NAME, PROMO_SUBCATEGORY,
PROMO_COST, PROMO_BEGIN_DATE, PROMO_END_DATE)
VALUES (888,'sms promotion', 'ad', 77000, '01/12/2022', '31/12/2022')
```

```
-- This is a Negative check:
INSERT INTO PROMOTIONS (PROMO_ID, PROMO_NAME, PROMO_SUBCATEGORY, PROMO_COST, PROMO_BEGIN_DATE,
PROMO_END_DATE)
VALUES (892,'sms promotion', 'ad', 77000, CAST('2022-12-01' as date), CAST('2020-11-16' as date))
```



CONSTRAINT CHECK ON SALES TABLE

```
The SALES table has four integrity constraints:
-- 1. shipping_date must be NULL or >sale_date
-- 2. payment_date must be NULL or >sale_date
-- 3. quantity_sold must be >0
-- 4. unit_price must be >0


SELECT * FROM SALES

--Positive Check Sales
INSERT Into SALES (SALESTRANS_ID, PROD_ID,
PROMO_ID, CUST_ID, CHANNEL_ID, SALE_DATE, SHIPPING_DATE, PAYMENT_DATE, QUANTITY_SOLD, AMOUNT_SOLD,
UNIT_PRICE)
VALUES (918844, 148, 999, 43773, 2, '2022-12-04', '2022-12-10', '2022-12-05', 21.00, 4.00, 4.00)

--Negative Check Sales
INSERT Into SALES (SALESTRANS_ID, PROD_ID, PROMO_ID, CUST_ID, CHANNEL_ID, SALE_DATE,
SHIPPING_DATE, PAYMENT_DATE, QUANTITY_SOLD, AMOUNT_SOLD, UNIT_PRICE)
VALUES (918846, 148, 999, 43773, 2, '2022-12-04', '2022-12-01', '2022-12-04', 21.00, 4.00, 2.00)
```



Business View's and Stored Procedures

As per our business requirements, we have created five business views out of which two of them are indexed view and implemented as stored procedure.

BUSINESS VIEW 1 - REVENUE ANALYSIS

```
CREATE OR ALTER VIEW [dbo].[REVENUE_ANALYSIS_VW] WITH SCHEMABINDING
AS
	SELECT s.PROD_ID as [PRODUCT ID],
		YEAR(s.SALE_DATE) AS [YEAR],
		MONTH(s.SALE_DATE) AS [MONTH],
		SUM(s.AMOUNT_SOLD) AS [Total Amount Sold],
		SUM(s.QUANTITY_SOLD) AS [Total Quantity Sold],
		cc.COUNTRY_REGION AS REGION,
		COUNT_BIG(*) AS [Count]
	FROM dbo.SALES as s
		JOIN dbo.CUSTOMERS as cust
			ON cust.CUST_ID = s.CUST_ID
		JOIN dbo.CUSTOMERS_COUNTRY as cc
			ON cust.COUNTRY_ID = cc.COUNTRY_ID
	GROUP BY s.PROD_ID, YEAR(s.SALE_DATE), MONTH(s.SALE_DATE), cc.COUNTRY_REGION
GO

CREATE UNIQUE CLUSTERED INDEX IDX_REVENUE_ANALYSIS ON [dbo].[REVENUE_ANALYSIS_VW] ([PRODUCT
ID], [YEAR], [MONTH], [REGION])
GO

CREATE OR ALTER VIEW [dbo].[REVENUE_ANALYSIS_AVG_VW] WITH SCHEMABINDING
AS
	SELECT
		[Product ID],
		[Year],
```

```
		[Month],
		[Region],
		[Total Amount Sold],
		[Total Quantity Sold],
		CAST([Total Amount Sold]/[Total Quantity Sold] AS DECIMAL (10,2)) AS [Average
Sale Price]
	FROM [dbo].[REVENUE_ANALYSIS_VW]
GO

CREATE OR ALTER PROCEDURE [dbo].[usp_revenue_analysis]
	@Year INT,
	@Month INT = 0,
	@Region varchar(20) = NULL
AS
BEGIN
	SET NOCOUNT ON
	IF @Month = 0 AND @Region IS NULL
		BEGIN
			SELECT [Product ID], [Year], [Month], [Region], [Total Amount Sold],
[Total Quantity Sold], [Average Sale Price] FROM [dbo].[REVENUE_ANALYSIS_AVG_VW] WHERE
[Year]=@Year
		END
	ELSE IF @Month = 0 AND @Region IS NOT NULL
		BEGIN
			SELECT [Product ID], [Year], [Month], [Region], [Total Amount Sold],
[Total Quantity Sold], [Average Sale Price] FROM [dbo].[REVENUE_ANALYSIS_AVG_VW]WHERE
[Year]=@Year AND [Region]=@Region
		END
	ELSE IF @Month > 0 AND @Region IS NULL
		BEGIN
			SELECT [Product ID], [Year], [Month], [Region], [Total Amount Sold],
[Total Quantity Sold], [Average Sale Price] FROM [dbo].[REVENUE_ANALYSIS_AVG_VW] WHERE
[Year]=@Year AND [Month]=@Month
		END
	ELSE
		BEGIN
			SELECT [Product ID], [Year], [Month], [Region], [Total Amount Sold],
[Total Quantity Sold], [Average Sale Price] FROM [dbo].[REVENUE_ANALYSIS_AVG_VW] WHERE
[Year]=@Year AND [Month]=@Month AND [Region]=@Region
		END
END
```

This view is an indexed view and implemented as stored procedure.

BUSINESS VIEW 2 - LIST TOP 10 POPULAR PRODUCTS

```
CREATE OR ALTER VIEW [dbo].[PROPULAR_PRODUCTS_VW] WITH SCHEMABINDING AS
	SELECT TOP 10 s.PROD_ID, p.PROD_NAME, YEAR(s.SALE_DATE) AS SALES_YEAR,
		SUM(s.QUANTITY_SOLD) AS TOTAL_QUANTITY_SOLD,
		cc.COUNTRY_REGION AS REGION
		FROM dbo.SALES as s
			JOIN dbo.CUSTOMERS as cust
				ON cust.CUST_ID = s.CUST_ID
			JOIN dbo.CUSTOMERS_COUNTRY as cc
				ON cust.COUNTRY_ID = cc.COUNTRY_ID
			JOIN dbo.PRODUCTS as p
				ON s.PROD_ID = p.PROD_ID
	GROUP BY s.PROD_ID, p.PROD_NAME, YEAR(s.SALE_DATE), cc.COUNTRY_REGION
	ORDER BY TOTAL_QUANTITY_SOLD DESC
GO
```

BUSINESS VIEW 3 - LIST MAXIMUM NUMBER OF DAYS TO SHIP EACH PRODUCT

```
CREATE OR ALTER VIEW [dbo].[PRODUCTS_SHIPDAYS_VW] WITH SCHEMABINDING AS
	SELECT s.PROD_ID, p.PROD_NAME,
		MAX(DATEDIFF(day,SALE_DATE,SHIPPING_DATE))  AS MAX_DAYS_TO_SHIP
	FROM dbo.SALES as s
		JOIN dbo.PRODUCTS AS p
			ON s.PROD_ID = p.PROD_ID
	GROUP BY s.PROD_ID , p.PROD_NAME
GO
```

BUSINESS VIEW 4 - LIST PROMOTIONS RESULTED IN HIGHEST SALES

```
CREATE OR ALTER VIEW [dbo].[SALES_PROMOTIONS_VW] WITH SCHEMABINDING AS
	SELECT p.PROMO_NAME,YEAR(s.SALE_DATE) AS SALES_YEAR,
		SUM (s.QUANTITY_SOLD) AS TOTAL_QUANTITY_SOLD,
		cc.COUNTRY_REGION AS REGION
	FROM dbo.SALES as s
		JOIN dbo.CUSTOMERS as cust
			ON cust.CUST_ID = s.CUST_ID
		JOIN dbo.CUSTOMERS_COUNTRY as cc
			ON cust.COUNTRY_ID = cc.COUNTRY_ID
		JOIN dbo.PROMOTIONS as p
			ON s.PROMO_ID = p.PROMO_ID
	WHERE p.PROMO_NAME NOT IN ('NO PROMOTION #')
	GROUP BY p.PROMO_NAME,YEAR(s.SALE_DATE),cc.COUNTRY_REGION
GO
```

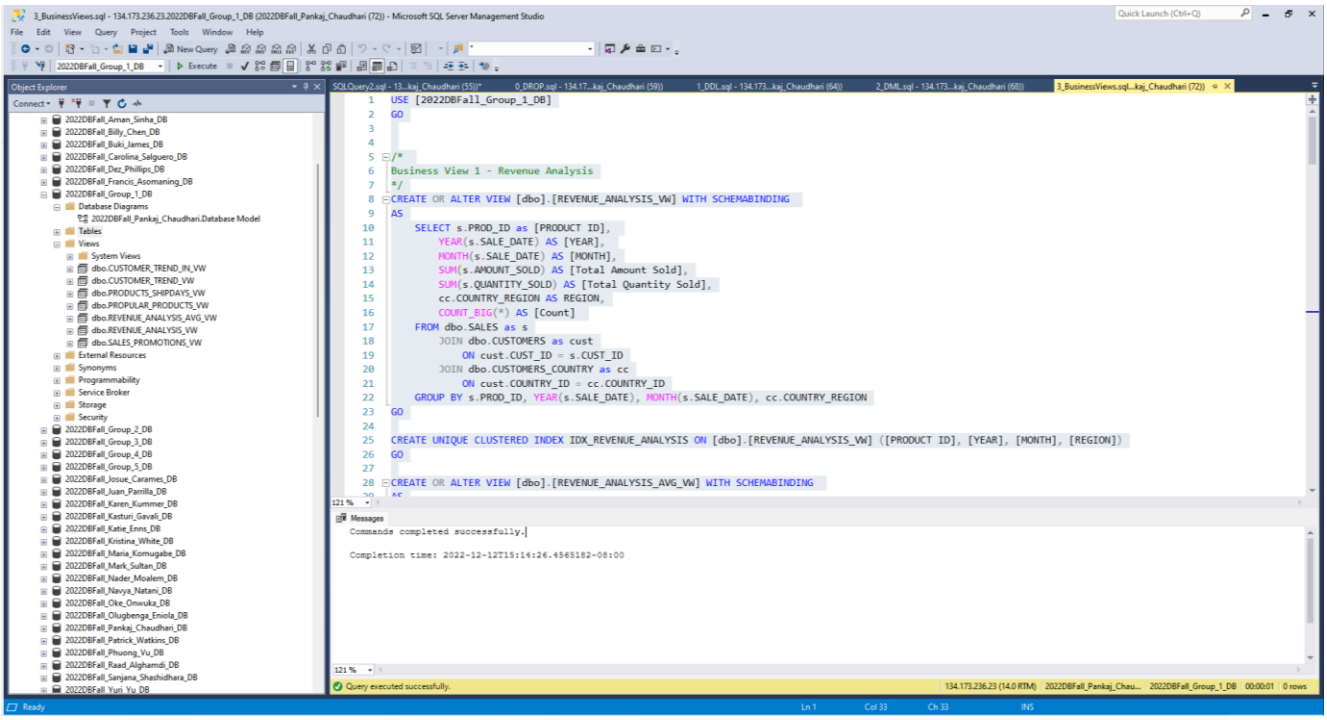
BUSINESS VIEW 5 - ANALYZE CUSTOMER TREND

```
CREATE OR ALTER VIEW [dbo].[CUSTOMER_TREND_VW] WITH SCHEMABINDING AS
	SELECT p.PROD_NAME AS [PRODUCT NAME],
		SUM (s.QUANTITY_SOLD) AS [TOTAL QUANTITY SOLD],
		c.CUST_GENDER AS [GENDER],
		c.CUST_MARITAL_STATUS AS [MARITAL STATUS],
		cc.COUNTRY_REGION AS REGION,
		COUNT_BIG(*) AS [COUNT]
	FROM dbo.CUSTOMERS as c
		JOIN dbo.SALES AS s
			ON c.CUST_ID = s.CUST_ID
		JOIN dbo.CUSTOMERS_COUNTRY as cc
			ON c.COUNTRY_ID = cc.COUNTRY_ID
		JOIN dbo.PRODUCTS as p
			ON p.PROD_ID = s.PROD_ID
	GROUP BY p.PROD_NAME , c.CUST_GENDER , c.CUST_MARITAL_STATUS, cc.COUNTRY_REGION
GO

CREATE UNIQUE CLUSTERED INDEX IDX_CUSTOMER_TREND ON [dbo].[CUSTOMER_TREND_VW] ([PRODUCT NAME],
[GENDER], [MARITAL STATUS], [REGION])
GO

CREATE OR ALTER VIEW [dbo].[CUSTOMER_TREND_IN_VW] WITH SCHEMABINDING AS
	SELECT [PRODUCT NAME],
```

```
            [TOTAL QUANTITY SOLD],
            [GENDER],
            [MARITAL STATUS],
            [REGION]
      FROM [dbo].[CUSTOMER_TREND_VW]
GO

CREATE OR ALTER PROCEDURE [dbo].[usp_customer_trends]
      @Gender Char (1),
      @MaritalStatus varchar(20),
      @Region varchar(20)
AS
BEGIN
      SELECT [PRODUCT NAME],
            [TOTAL QUANTITY SOLD]
      FROM [dbo].[CUSTOMER_TREND_VW]
      WHERE [GENDER] = @Gender AND
            [MARITAL STATUS] = @MaritalStatus AND
            [REGION] = @Region
      ORDER BY [PRODUCT NAME]
END
GO
```

This view is an indexed view and implemented as stored procedure.

We have clubbed all above business views scripts into single file and executed file. Below is screenshot for successful execution of business views scripts.

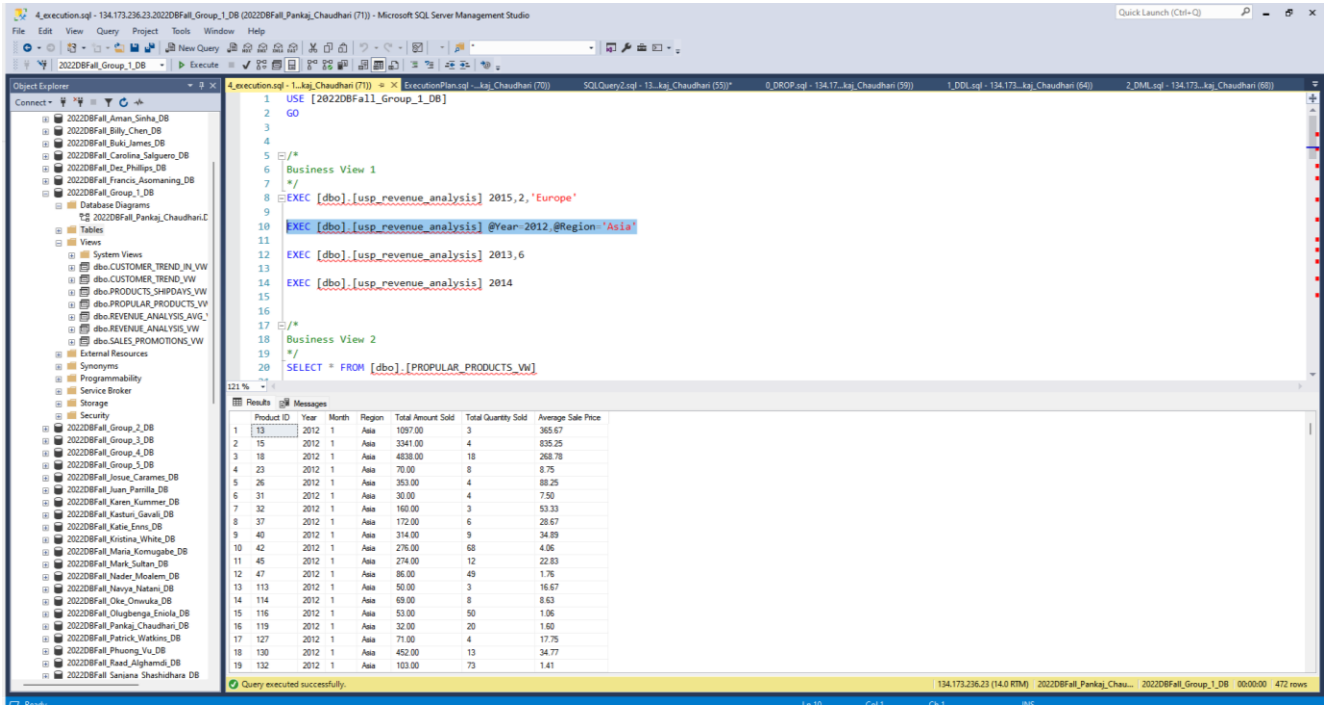


BUSNESS VIEW EXECUTION

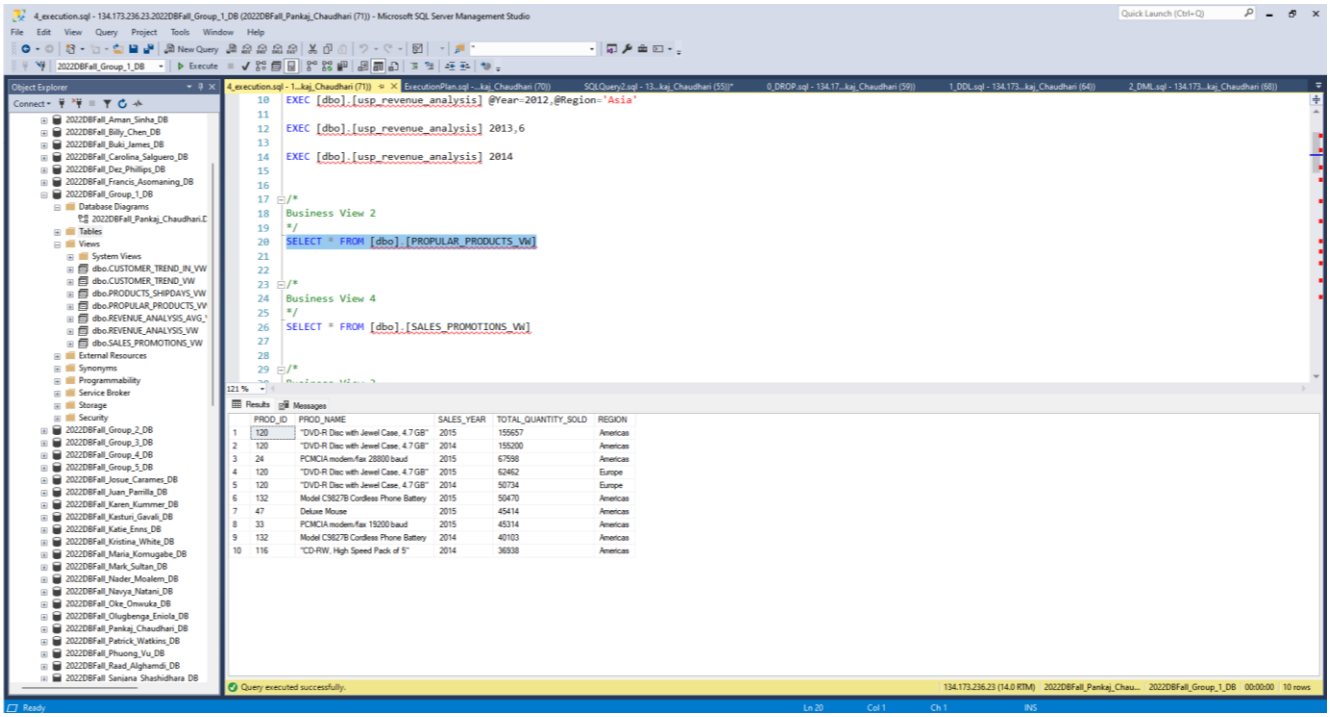
The above business views can be executed with following statements.

```
/*
Business View 1
*/
EXEC [dbo].[usp_revenue_analysis] 2015,2,'Europe'
```

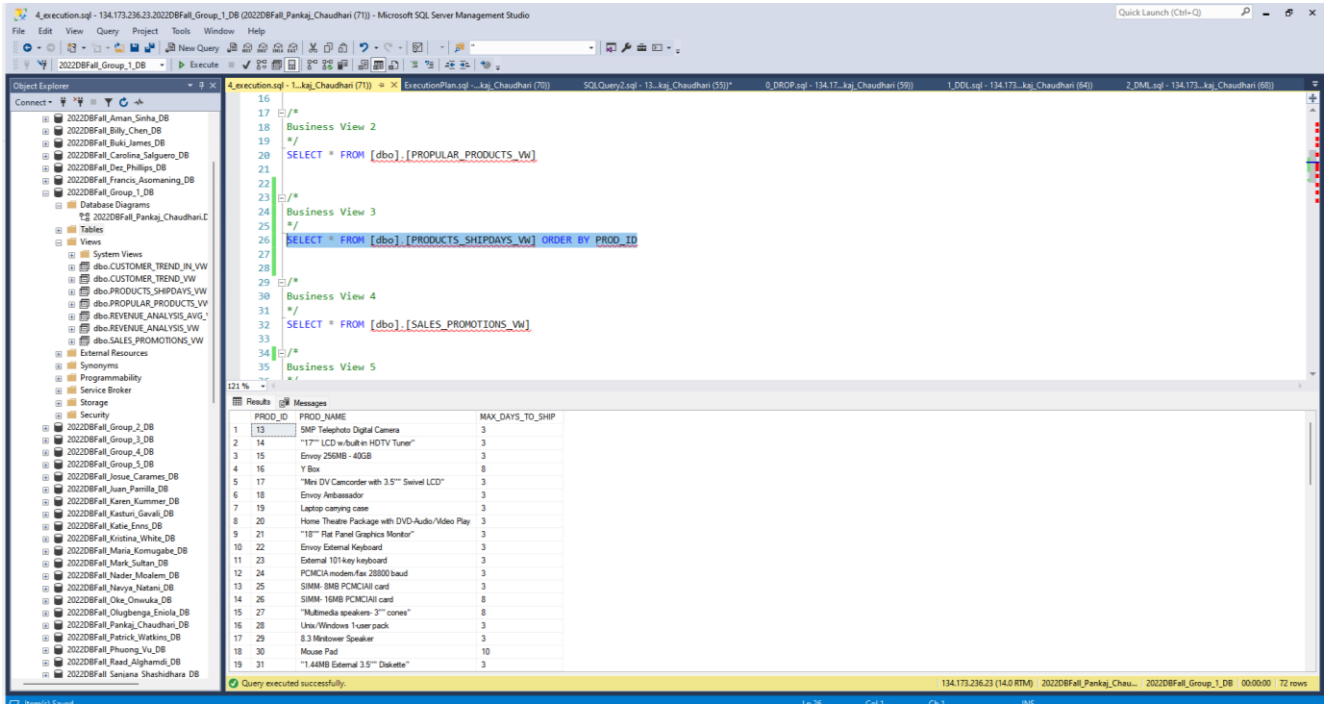
```
EXEC [dbo].[usp_revenue_analysis] @Year=2012,@Region='Asia'

EXEC [dbo].[usp_revenue_analysis] 2013,6

EXEC [dbo].[usp_revenue_analysis] 2014
```



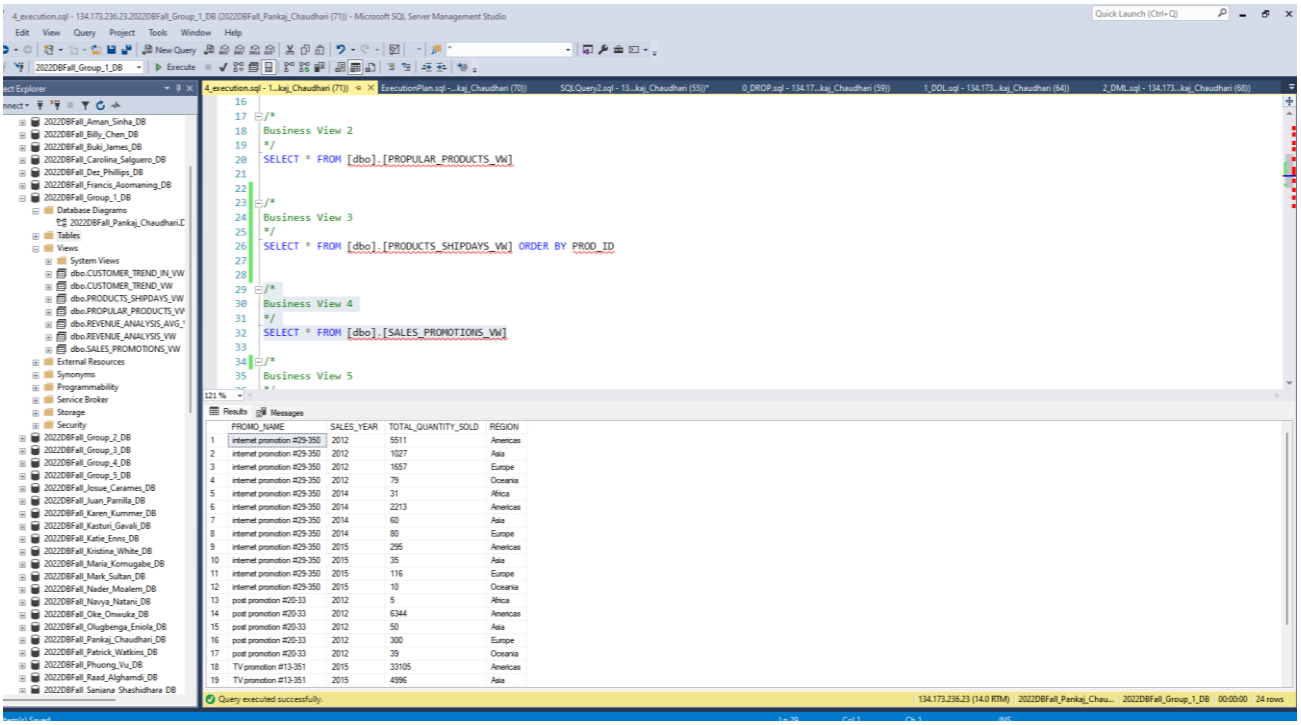
```
/*
Business View 2
*/
SELECT * FROM [dbo].[PROPULAR_PRODUCTS_VW]
```



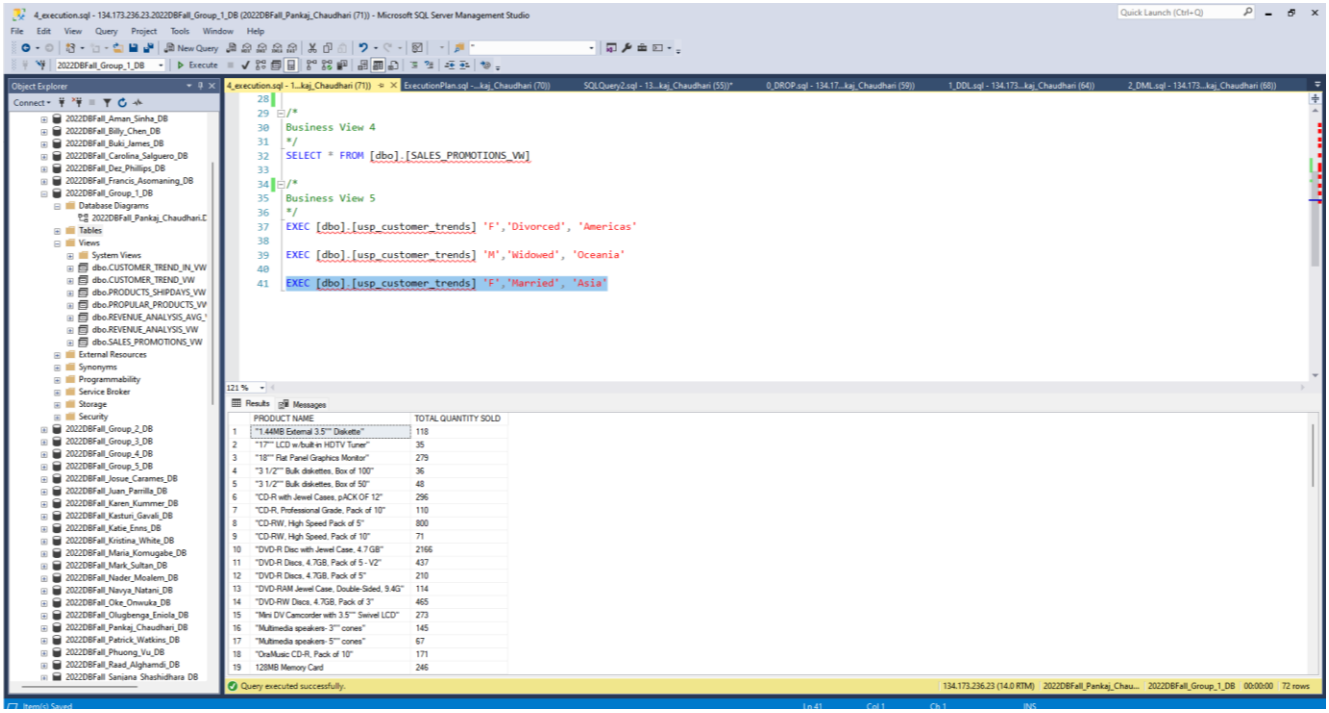
```
/*
Business View 3
*/
SELECT * FROM [dbo].[PRODUCTS_SHIPDAYS_VW] ORDER BY PROD_ID
```



```
/*
Business View 4
*/
SELECT * FROM [dbo].[SALES_PROMOTIONS_VW]
```



```
/*
Business View 5
*/
EXEC [dbo].[usp_customer_trends] 'F','Divorced', 'Americas'

EXEC [dbo].[usp_customer_trends] 'M','Widowed', 'Oceania'

EXEC [dbo].[usp_customer_trends] 'F','Married', 'Asia'
```



Security

We have defined three users Sales Executive, Marketing Analyst and Manager, considering their authority and the security requirements as above we designed the following grant statements for the business view created above.

```
/*
Business View 1 - Revenue Analysis
*/
GRANT SELECT ON [dbo].[REVENUE_ANALYSIS_VW] TO MANAGER
GRANT SELECT ON [dbo].[REVENUE_ANALYSIS_AVG_VW] TO MANAGER

CREATE OR ALTER VIEW [dbo].[REVENUE_ANALYSIS_AVG_SALES_VW] WITH SCHEMABINDING
AS
       SELECT *
       FROM [dbo].[REVENUE_ANALYSIS_AVG_VW]
       WHERE [REGION] IN ('Americas')
GO

GRANT EXECUTE ON OBJECT::dbo.usp_revenue_analysis
    TO [DEPARTMENT];
GO

GRANT SELECT ON [dbo].[REVENUE_ANALYSIS_AVG_SALES_VW] TO SALES
```

```
/*
Business View 2 -List top 10 popular products
*/
GRANT SELECT ON [dbo].[PROPULAR_PRODUCTS_VW] TO MANAGER, SALES


/*
Business View 3 - List maximum number of days to ship each product
*/
GRANT SELECT ON [dbo].[PRODUCTS_SHIPDAYS_VW] TO MANAGER, SALES, MARKETTING


/*
Business View 4 - List promotions resulted in highest sale
*/
GRANT SELECT ON [dbo].[SALES_PROMOTIONS_VW] TO MANAGER, MARKETTING


/*
Business View 5 - Analyse customer trend
*/
GRANT SELECT ON [dbo].[CUSTOMER_TREND_VW] TO MANAGER, SALES, MARKETTING
GRANT SELECT ON [dbo].[CUSTOMER_TREND_IN_VW] TO MANAGER, SALES, MARKETTING
GRANT EXECUTE ON OBJECT::dbo.usp_customer_trends
    TO [DEPARTMENT];
GO
```

Note: Don't try to execute above scripts. These are reference purpose only.

Execution Plan and Query Performance

After creating business views, we tried creating index on columns participating in JOIN or part of order by clause. Having indexes does result in any performance improvement for business views 2, 3 and 4. Hence in this section, we are more focused on business view 1 and 5, where we had significant improvement in query performance after indexing.

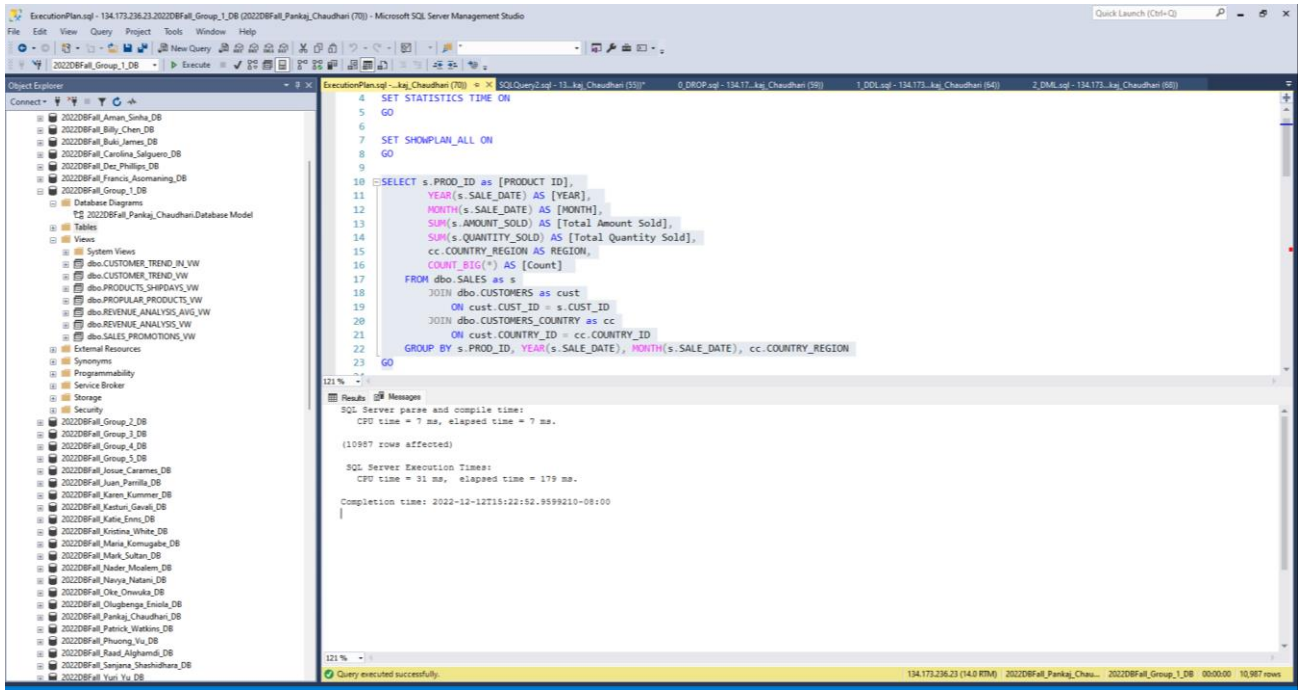
For business view 1 and business view, we created 2 plans (With and Without indexing) to measure the query performance and calculate query cost. In case of plan 2, for business view 1 unique clustered indexes are created on columns "PRODUCT_ID", "YEAR", "MONTH" and "REGION" column. For business view 5, unique clustered indexes are created on columns "PRODUCT_NAME", "GENDER", "MARITIAL_STATUS" and "REGION".

Below is query performance statistics summary for business views before and after indexing.

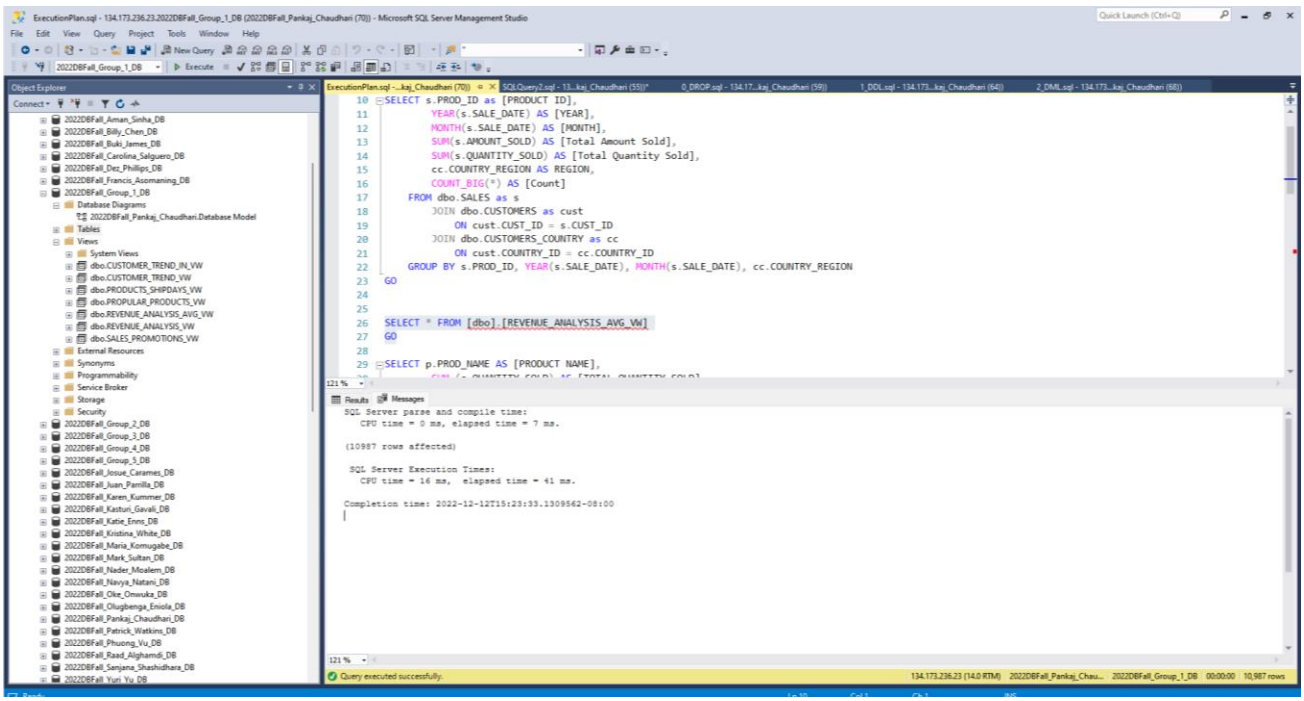
Business Views	Plan 1 - Query Performance (without Index) in ms	Plan 2 - Query Performance (with Index) in ms	Best Plan
BV 1	179	41	Plan 2
BV 5	44	16	Plan 2

Below are statistics screenshots for each view and each plan.

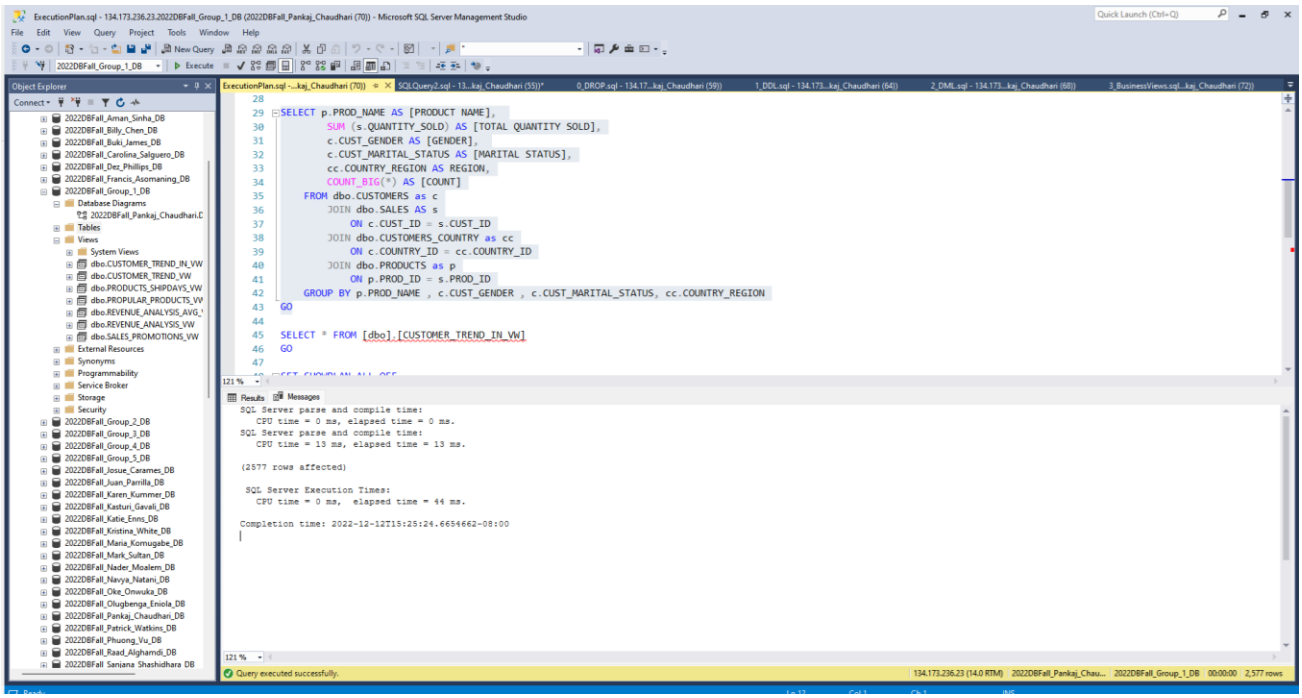
Business View 1 – Plan 1



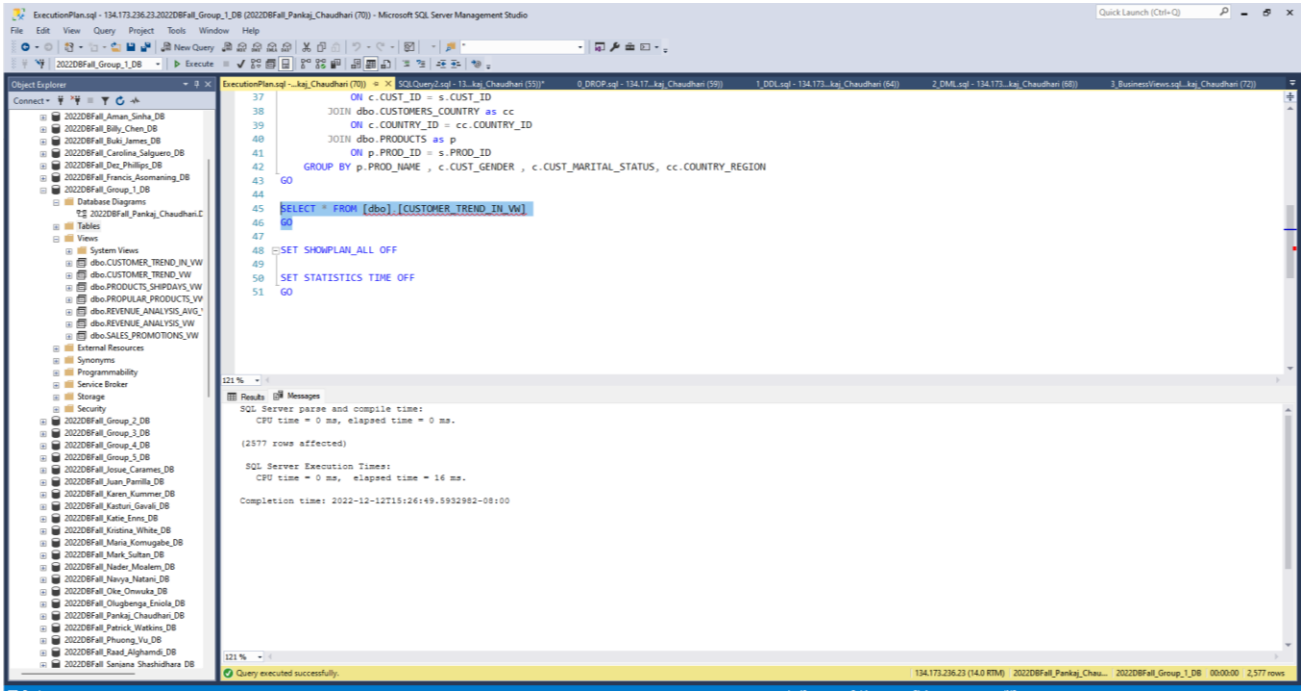
Business View 1 – Plan 2



Business View 5 - Plan 1



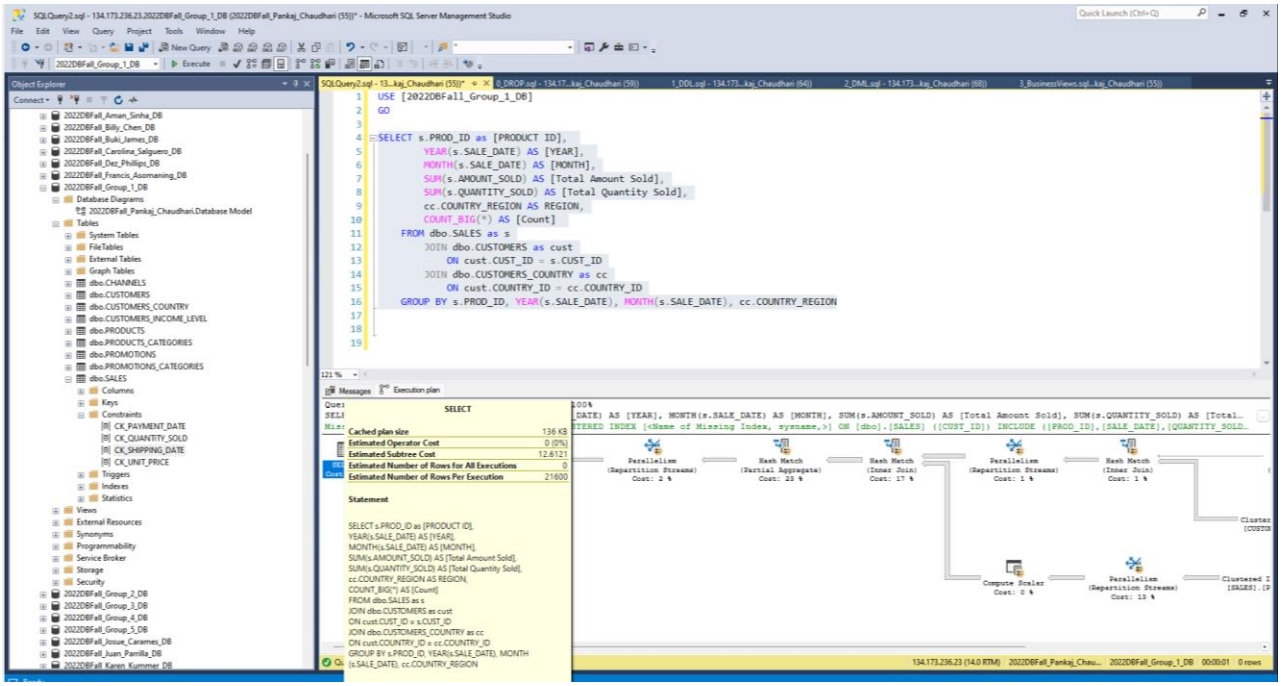
Business View 5 - Plan 2



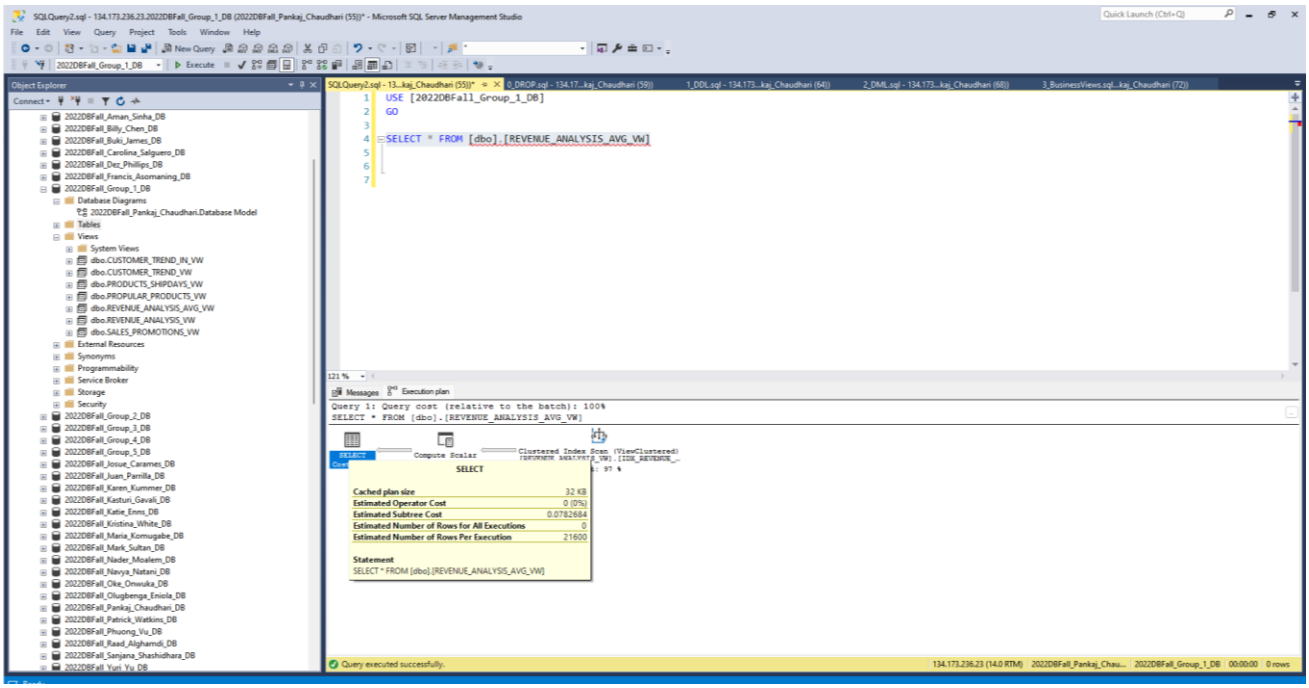
Below is query cost table for business views before and after indexing.

Business Views	Plan 1 - (without Index) Cost	Plan 2 - (with Index) Cost	Best Plan
BV 1	12.61	0.078	Plan 2
BV 5	10.60	0.022	Plan 2

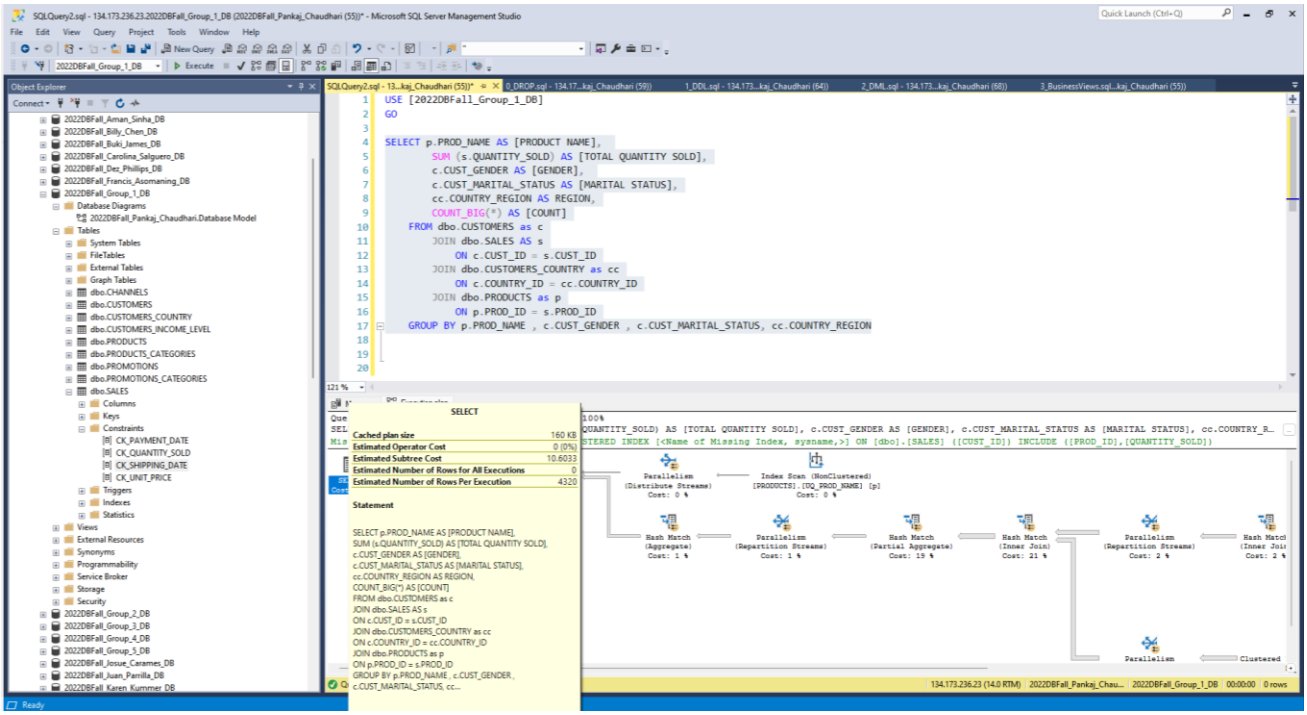
Business View 1 - Plan 1



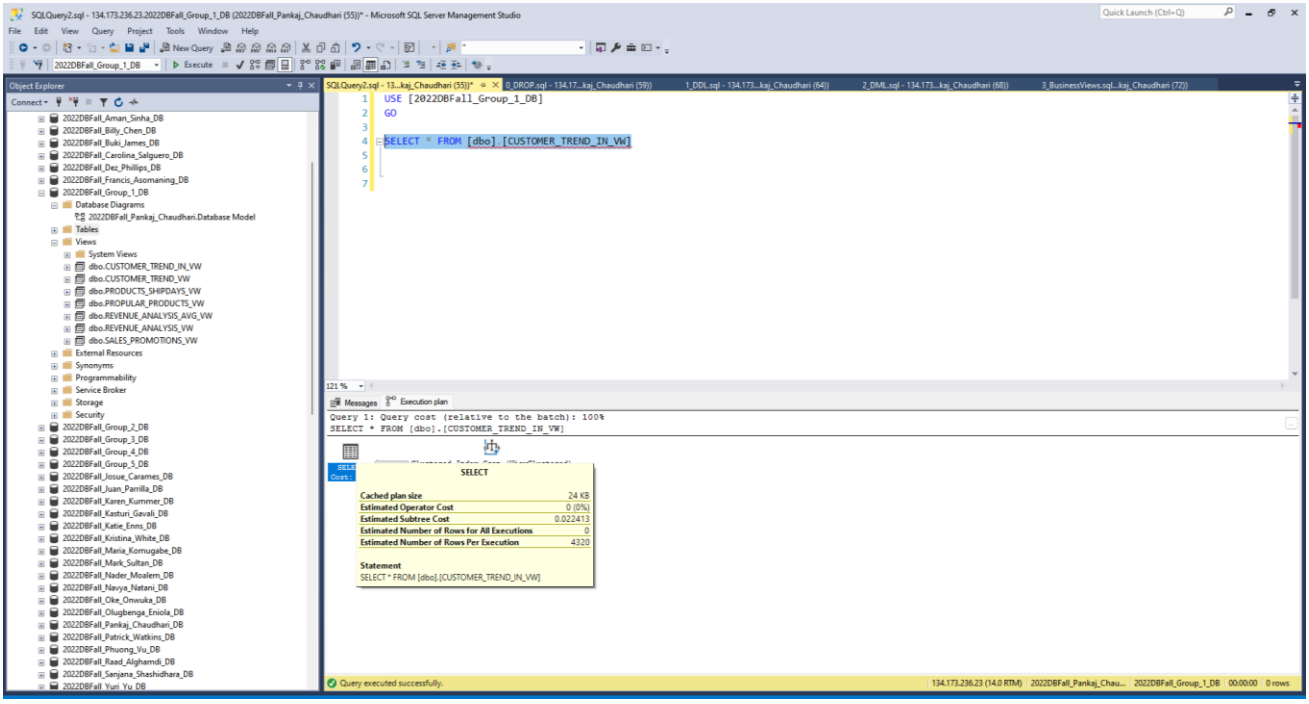
Business View 1 - Plan 2



Business View 5 - Plan 1



Business View 5 - Plan 2



Database Deliverables

Database scripts is our deliverable for this project. You can use script below. Make sure you have access to “LIY26” database.

project_scripts.zip

You can copy paster above zip file on your machine. You must execute above scripts in following sequence. Make sure to change the database name in each script (usually specified at beginning) with appropriate database name you are using.

1. 0_Drop.sql
2. 1_DDL.sql
3. 2_DML.sql
4. 3_BusinessViews.sql
5. 4_execution.sql

Section 8: Conclusion

We were assigned with this project to design a database system for a Sales transaction application. We started this project by understanding about sales transaction application. We learned how important this application is for businesses to thrive. To make the best use of the available sales data it is important to understand the data. We took a deep dive in understanding the existing data and came up with better and more clean and useful database design. We observed the dependencies of tables in this database and how we can leverage those to help the business executives.

Once the data started making sense to us, we then brainstormed on what types of business queries as views would help the business executives to make better decision and hence be beneficial to grow the business. We designed the views keeping in mind the requirements and interests of the business executives or users. We learned that it is important to consider the user needs and how to make it more functional and easier for them.

We created stored procedures to fit the requirements of the users and assigned grant statements to users for security purpose. We created unique clustered indexes to optimize the query performances. We learned that indexes could optimize the query executions and take less time to run the query. We hope that we have provided the users with better decision-making business queries that will help them with their business.

References

[1] Ingram, D. (n.d.). Point of sale vs. transaction processing systems. Small Business Chron.
https://smallbusiness.chron.com/point-sale-vs-transaction-processing-systems-17548.html

[2] Elmsari, R. and Navathe, S.B. (2016). Additional database topics: security. Fundamentals of database systems (pp. 1121 - 1158). Boston: Pearson.

Appendix 1: Source LIY26 Database Tables

Database Tables Table Name	Description	Columns
LI_CHANNELS	Data on Distribution Channels	**CHANNEL_ID**, CHANNEL_DESC, CHANNEL_CLASS, CHANNEL_TOTAL
LI_CUSTOMERS_INTX	Data on Customers obtained from internal source	**CUST_ID**, CUST_FIRST_NAME, CUST_LAST_NAME, CUST_GENDER ,CUST_MAIN_PHONE_NUMBER ,CUST_EMAIL, CUST_STREET_ADDRESS ,CUST_POSTAL_CODE ,CUST_CITY ,CUST_STATE_PROVINCE ,C.COUNTRY_ID ,CUST_TOTAL, COUNTRY_NAME ,COUNTRY_SUBREGION ,COUNTRY_REGION, COUNTRY_TOTAL
LI_CUSTOMERS_EXT	Data on Customers obtained from external sources	**CUST_ID**, CUST_GENDER, CUST_YEAR_OF_BIRTH, CUST_MARITAL_STATUS, CUST_INCOME_LEVEL, CUST_CREDIT_LIMIT
LI_PRODUCTS	Product data	**PROD_ID**, PROD_NAME, PROD_DESC, PROD_SUBCATEGORY, PROD_SUBCAT_DESC, PROD_CATEGORY, PROD_CAT_DESC, PROD_WEIGHT_CLASS, PROD_UNIT_OF_MEASURE, PROD_PACK_SIZE, SUPPLIER_ID, PROD_STATUS, PROD_LIST_PRICE, PROD_MIN_PRICE, PROD_TOTAL
LI_PROMOTIONS	Promotions data	**PROMO_ID**, PROMO_NAME, PROMO_SUBCATEGORY, PROMO_CATEGORY, PROMO_COST, PROMO_BEGIN_DATE, PROMO_END_DATE, PROMO_TOTAL
LI_SALES_12_13	Sales transaction data for 2012 through 2013	**SALESTRANS_ID,** *PROD_ID, CUST_ID, CHANNEL_ID, PROMO_ID, SALE_DATE*, SHIPPING_DATE, PAYMENT_DATE, QUANTITY_SOLD, AMOUNT_SOLD, UNIT_PRICE
LI_SALES_14	Sales transaction data for 2014	**SALESTRANS_ID,** *PROD_ID, CUST_ID, CHANNEL_ID, PROMO_ID, SALE_DATE*, SHIPPING_DATE, PAYMENT_DATE, QUANTITY_SOLD, AMOUNT_SOLD, UNIT_PRICE