



MAG 28 C

# SALES TRANSACTION APPLICATION DATABASE DESIGN

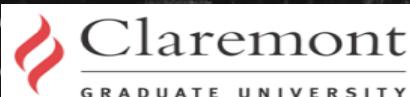
---

## IST 302: Databases

(Fall 2022)

Claremont Graduate University

- ❖ Kasturi Gavali
- ❖ Maria Komugabe
- ❖ Pankaj Chaudhari



# 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.....	4
Sales Transactions Applications .....	4
Users.....	4
Section 2: Data Understanding.....	6
Overview .....	6
Findings.....	6
Summary .....	13
Section 3: Conceptual Design .....	15
Overview .....	15
Section 4: Security Requirements .....	17
Section 5: Logical Design.....	18
Tables with Design .....	18
Section 6: Physical Design.....	22
DDL Scripts .....	22
Database Model .....	27
Section 7: Database Implementation.....	28
DML Scripts for Populating Data.....	28
Test Integrity Constraints.....	30
Business View's and Stored Procedures .....	33
Security .....	39
Execution Plan and Query Performance .....	40
Database Deliverables.....	45
Section 8: Conclusion .....	46
<i>References</i> .....	47
<i>Appendix 1: Source LIY26 Database Tables</i> .....	48

# 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 key 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.

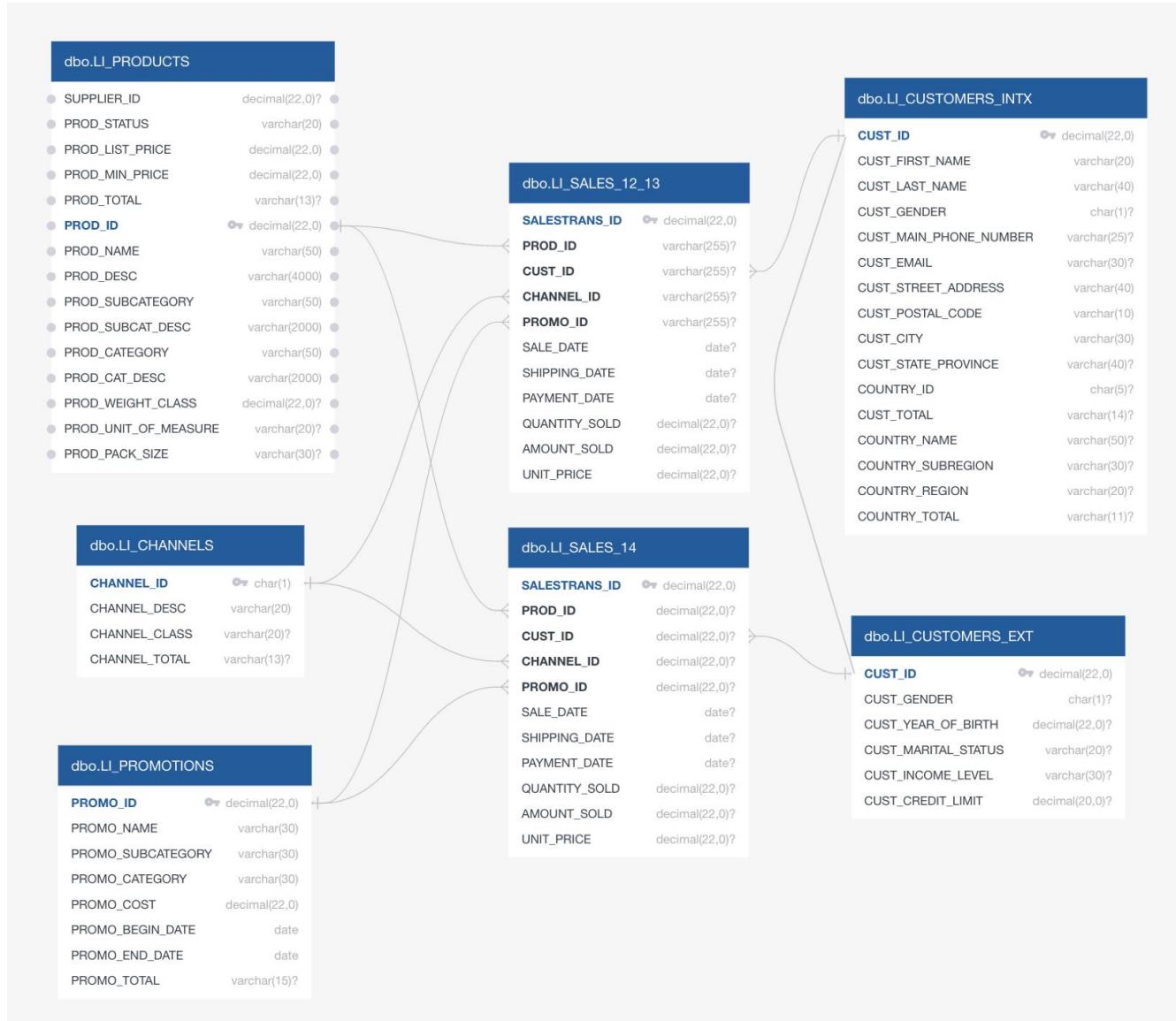
Identified key users have the following roles and would be interested in associated business queries are shown in table below.

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	<ul style="list-style-type: none"> <li>• Sales revenue analysis (monthly, quarterly, yearly)</li> <li>• Find the most popular products sorted by Year.</li> <li>• Find out maximum number of days it takes to ship each product.</li> <li>• Understand the product purchase trend by customer genders, marital status, and geography.</li> </ul>
U02 (Marketing)	Creates and retires promotions by analyzing sales, aiming to tailor promotions based on recorded and potential selling patterns	<ul style="list-style-type: none"> <li>• View Promotions resulted in highest sales</li> <li>• Find out maximum number of days it takes to ship each product.</li> <li>• Understand the product purchase trend by customer genders, marital status, and geography.</li> </ul>
U03 (Manager)	Analyzes and aims to optimize and incentivize performance of sales and job productivity	<ul style="list-style-type: none"> <li>• Sales revenue analysis (monthly, quarterly, yearly and Region)</li> <li>• Find the most popular products sorted by Year.</li> <li>• View Promotions resulted in highest sales</li> <li>• View Most used channels in sales transaction.</li> <li>• Find out maximum number of days it takes to ship each product.</li> <li>• Understand the product purchase trend by customer genders, marital status, and geography.</li> </ul>

# 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	<b>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.</b>	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_PROVINCE
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_SUBCATEG ORY	Varchar (50), NOT NULL	Yes	Yes	Yes, 21	No
PROD_SUBCAT_D ESC	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_SUBCATEG ORY
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_CL ASS	Decimal (22, 0), NULL	No, should be int.	No		No
PROD_UNIT_OF_M EASURE	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_SUBCATEG ORY	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
SALETRANS_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.

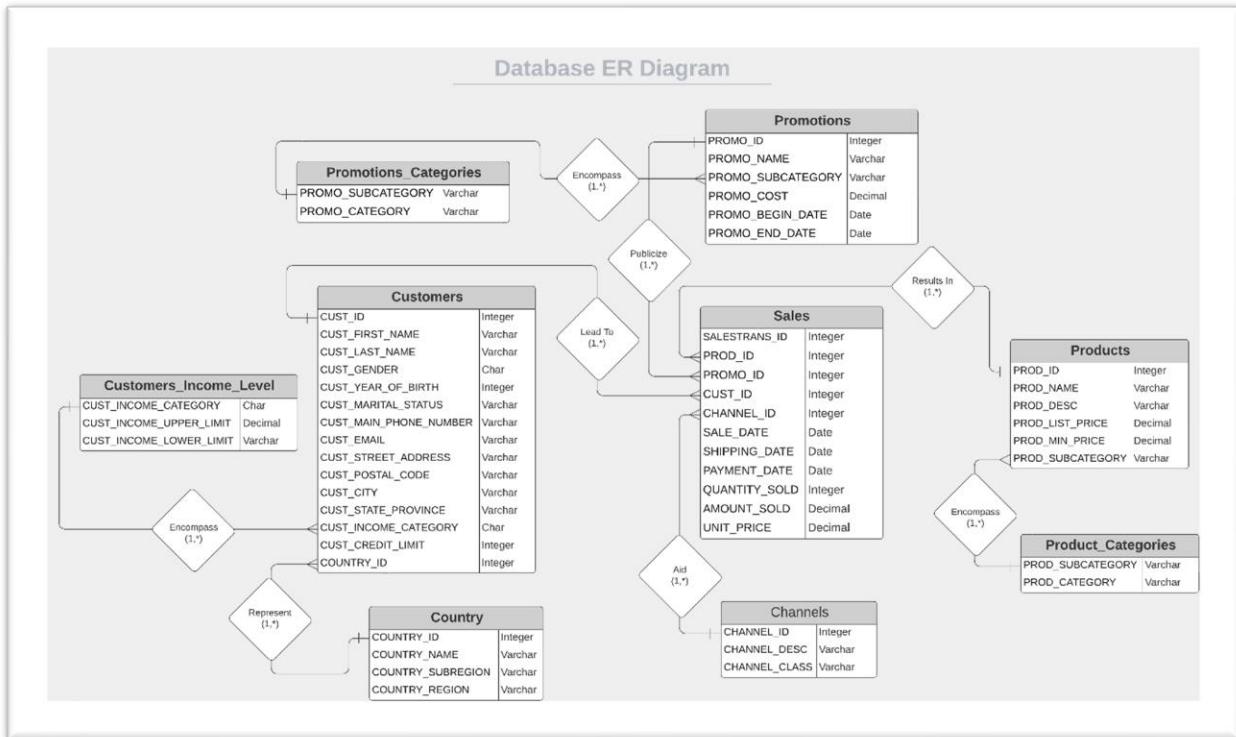
- Promotions
- Products
- Country
- Channel
- Customers
- Sales

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  $(\text{SUM}(\text{UNIT\_PRICE})/\text{SUM}(\text{QUANTITY\_SOLD}))$ . This view will be used as sales executive to see sales revenue by month and year but only for their assigned region. This view will be implemented as stored procedure which will accept 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
SALETRANS_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.

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer on the left, several databases are listed, including 2020Fall\_Group\_1\_DB, 2020Fall\_Pankaj\_Chaudhan, 2019W\_Yiu\_DB, 2019W\_Yiu\_DB, 2020Fall\_CenEvel\_DB, 2020Fall\_Aman\_Sinha\_DB, 2020Fall\_Billy\_Chen\_DB, 2020Fall\_Buki\_Jemre\_DB, 2020Fall\_Carolina\_Salgado\_DB, 2020Fall\_Dev\_Phillips\_DB, 2020Fall\_Francis\_Asumming\_DB, and 2020Fall\_Group\_1\_DB. The central pane displays a multi-line T-SQL script with line numbers from 188 to 214. The script contains multiple ALTER TABLE statements adding constraints to the dbo.SALES table. The right pane shows the results of the execution, including the completion time (2022-12-12T13:46:09.2470768-08:00) and a message indicating the command completed successfully.

```

188 189 ALTER TABLE dbo.SALES ADD CONSTRAINT PK_SALESTRAINS_ID PRIMARY KEY (SALESTRAINS_ID)
190 GO
191 192 ALTER TABLE dbo.SALES ADD CONSTRAINT FK_PROD_ID FOREIGN KEY (PROD_ID) REFERENCES PRODUCTS (PROD_ID)
193 GO
194 195 ALTER TABLE dbo.SALES ADD CONSTRAINT FK_CUST_ID FOREIGN KEY (CUST_ID) REFERENCES CUSTOMERS (CUST_ID)
196 GO
197 198 ALTER TABLE dbo.SALES ADD CONSTRAINT FK_CHANNEL_ID FOREIGN KEY (CHANNEL_ID) REFERENCES CHANNELS (CHANNEL_ID)
199 GO
200 201 ALTER TABLE dbo.SALES ADD CONSTRAINT FK_PROMO_ID FOREIGN KEY (PROMO_ID) REFERENCES PROMOTIONS (PROMO_ID)
202 GO
203 204 ALTER TABLE dbo.SALES ADD CONSTRAINT CK_SHIPPING_DATE CHECK (SHIPPING_DATE >= SALE_DATE OR SHIPPING_DATE IS NULL)
205 GO
206 207 ALTER TABLE dbo.SALES ADD CONSTRAINT CK_PAYMENT_DATE CHECK (PAYMENT_DATE >= SALE_DATE OR PAYMENT_DATE IS NULL)
208 GO
209 210 ALTER TABLE dbo.SALES ADD CONSTRAINT CK_QUANTITY SOLD CHECK (QUANTITY SOLD > 0)
211 GO
212 213 ALTER TABLE dbo.SALES ADD CONSTRAINT CK_UNIT PRICE CHECK (UNIT PRICE > 0)
214 GO

```

Completion time: 2022-12-12T13:46:09.2470768-08:00

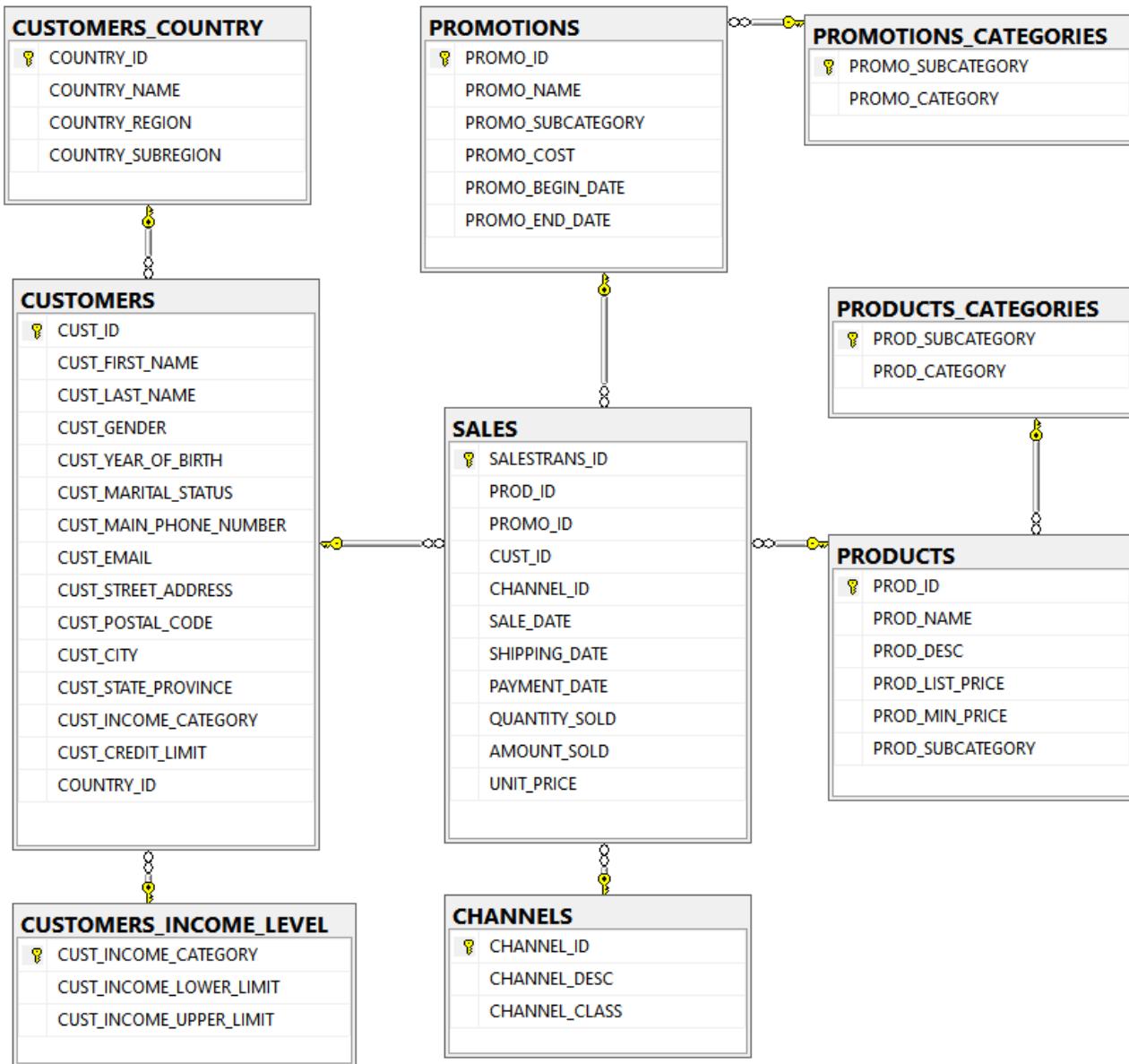
Messages

Commands completed successfully.

Query executed successfully.

## 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

```

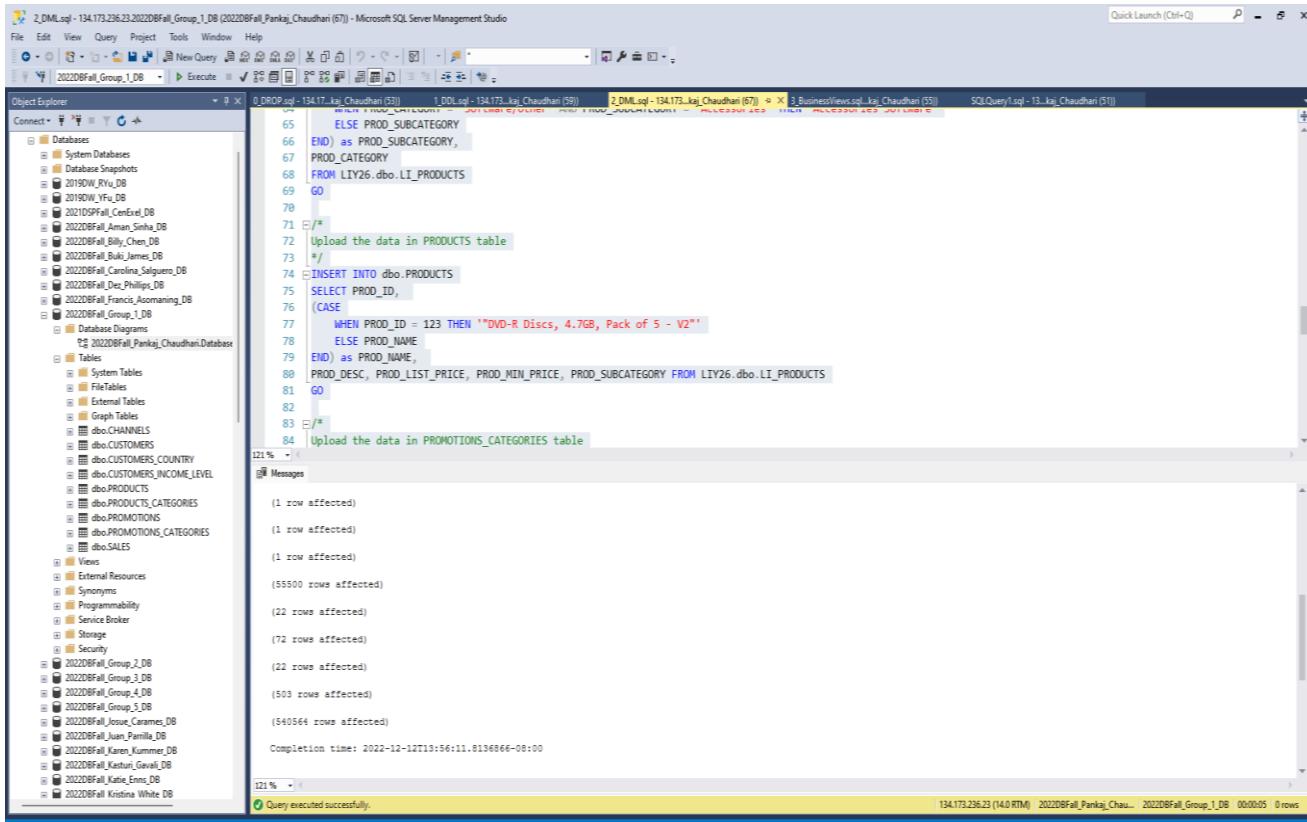
## *POPULATE PROMOTIONS TABLE*

```
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

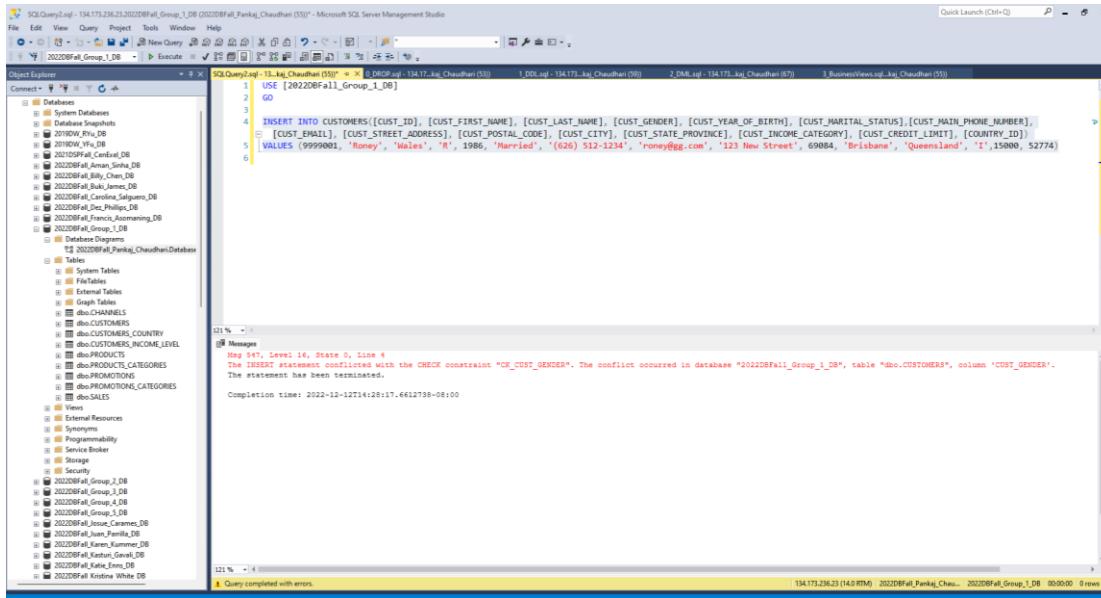
```
SELECT * FROM CUSTOMERS
```

--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))
```

The screenshot shows the SQL Server Management Studio interface. In the Object Explorer, the database '2022DBFall\_Group\_1\_DB' is selected, displaying tables like CHANNELS, CUSTOMERS, and PROMOTIONS. In the center pane, a query window titled 'SQLQuery2.sql - 134.173.236.23,2022DBFall\_Group\_1\_DB (2022DBFall\_Pankaj\_Chaudhari (55))' contains the following SQL code:

```
1 USE [2022DBFall_Group_1_DB]
2 GO
3
4 INSERT INTO PROMOTIONS (PROMO_ID, PROMO_NAME, PROMO_SUBCATEGORY, PROMO_COST, PROMO_BEGIN_DATE, PROMO_END_DATE)
5 VALUES (892, 'sms promotion', 'ad', 77000, CAST('2022-12-01' as date), CAST('2020-11-16' as date))
6
```

In the Messages pane, an error message is displayed:

```
Msg 547, Level 16, State 0, Line 4
The INSERT statement conflicted with the CHECK constraint "CK_PROMO_END_DATE". The conflict occurred in database "2022DBFall_Group_1_DB", table "dbo.PROMOTIONS".
The statement has been terminated.
```

Completion time: 2022-12-12T14:58:37.2466083-08:00

At the bottom, it says 'Query completed with errors.'

## 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)
```

```

SQLQuery2.sql - 134.173.236.23[2022DBFall_Group_1_DB (2022DBFall_Pankaj_Chaudhari (55))] - Microsoft SQL Server Management Studio
Edit View Query Project Tools Window Help
2022DBFall_Group_1_DB Execute
1 USE [2022DBFall_Group_1_DB]
2 GO
3
4 INSERT Into SALES (SALETRANS_ID, PROD_ID, PROMO_ID, CUST_ID, CHANNEL_ID, SALE_DATE, SHIPPING_DATE, PAYMENT_DATE, QUANTITY SOLD, AMOUNT SOLD,
5     UNIT PRICE)
VALUES (918846, 148, 999, 43773, 2, '2022-12-04', '2022-12-01', '2022-12-04', 21.00, 4.00, 2)
6
7
8
121 %
Messages
Msg 547, Level 16, State 0, Line 4
The INSERT statement conflicted with the CHECK constraint "CK_SHIPPING_DATE". The conflict occurred in database "2022DBFall_Group_1_DB", table "dbo.SALES".
The statement has been terminated.

Completion time: 2022-12-12T15:07:22.7936158-08:00
134.173.236.23 (14.0 RTM) 2022DBFall_Pankaj_Cha... 2022DBFall_Group_1_DB 00:00:00 0 rows
121 %
Query completed with errors.
Line 1 Col 1 Ch 1 INS
134.173.236.23 (14.0 RTM) 2022DBFall_Pankaj_Cha... 2022DBFall_Group_1_DB 00:00:00 0 rows

```

## 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
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].[PROPOPULAR_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.

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists various databases and objects. The central pane displays a script window with the following content:

```

USE [2020BFall_Group_1_DB]
GO
/*
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
/*
Business View 2 - Revenue Analysis Average
*/
CREATE OR ALTER VIEW [dbo].[REVENUE_ANALYSIS_AVG_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
Completion time: 2022-12-12T15:14:26.4565182-08:00

```

The status bar at the bottom indicates "Commands completed successfully." and "Completion time: 2022-12-12T15:14:26.4565182-08:00".

## BUSINESS 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
```

Product ID	Year	Month	Region	Total Amount Sold	Total Quantity Sold	Average Sale Price
13	2012	1	Asia	1097.00	3	365.67
15	2012	1	Asia	3341.00	4	835.25
18	2012	1	Asia	4834.00	18	268.78
23	2012	1	Asia	70.00	8	8.75
26	2012	1	Asia	351.00	4	88.25
31	2012	1	Asia	30.00	4	7.50
32	2012	1	Asia	160.00	3	53.33
37	2012	1	Asia	72.00	5	28.57
40	2012	1	Asia	314.00	9	34.89
42	2012	1	Asia	276.00	68	4.06
45	2012	1	Asia	274.00	12	22.83
47	2012	1	Asia	96.00	49	1.76
113	2012	1	Asia	50.00	3	16.67
114	2012	1	Asia	69.00	8	8.63
116	2012	1	Asia	53.00	50	1.06
119	2012	1	Asia	32.00	20	1.60
127	2012	1	Asia	71.00	4	17.75
130	2012	1	Asia	452.00	13	34.77
132	2012	1	Asia	103.00	73	1.41

```
/*
Business View 2
*/
```

```
SELECT * FROM [dbo].[PROPOPULAR_PRODUCTS_VW]
```

PROD_ID	PROD_NAME	SALES_YEAR	TOTAL_QUANTITY SOLD	REGION	
100	"DVD-R Disc with Jewel Case, 4.7 GB"	2015	155557	Americas	
120	"DVD-R Disc with Jewel Case, 4.7 GB"	2014	155200	Americas	
24	PCMCIA modem/fax 28200 baud	2015	67588	Americas	
40	"DVD-R Disc with Jewel Case, 4.7 GB"	2015	62462	Europe	
50	120	"DVD-R Disc with Jewel Case, 4.7 GB"	2014	50734	Europe
6132	Model C98ZTB Cordless Phone Battery	2015	50470	Americas	
47	Deluxe Mouse	2015	45414	Americas	
33	PCMCIA modem/fax 19200 baud	2015	45314	Americas	
132	Model C98ZTB Cordless Phone Battery	2014	40103	Americas	
116	"CD-RW, High Speed Pack of 5"	2014	36938	Americas	

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

The screenshot shows the SQL Server Management Studio interface with the following details:

- Object Explorer:** Shows the database structure, including tables like Aman\_Sinha\_DB, Billy\_Chen\_DB, Buji\_James\_DB, Carolina\_Salgueiro\_DB, Dez\_Philips\_DB, Francis\_Asonamning\_DB, Group\_1\_DB, and various system and external resources.
- Execution Plan:** Shows the execution plan for the query.
- Results:** Displays the output of the query:
 

PROD_ID	PROD_NAME	MAX_DAYS_TO_SHIP
13	SMP Telephone Digital Camera	3
2	"17" LCD w/builtin HDTV Tuner"	3
3	Envoy 250MB-40GB	3
4	Envoy External Keyboard	8
5	"My DV Camcorder with 3.5" Swivel LCD"	3
6	Envoy Ambassador	3
7	Laptop carrying case	3
8	Home Theatre Package with DVD-Audio/Media Play	3
9	"18" Ret Panel Graphics Monitor"	3
10	Envoy External Keyboard	3
11	External 101 key keyboard	3
12	PCMCIA modem/fax 28800 baud	3
13	SIMM-8MB PCMCIA card	3
14	SIMM-16MB PCMCIA card	8
15	"Multimedia speakers 3" cones"	8
16	Usr/Windows User pack	3
17	8.3 Mhz Speaker	3
18	Mouse Pad	10
19	"14MM External 3.5" Diskette"	3
- Messages:** Shows the message "Query executed successfully."
- Status Bar:** Shows the session ID (134.173.236.23), the database (202DBFall\_Group\_1\_DB), and the time (00:00:00).

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

The screenshot shows the SQL Server Management Studio interface with the following details:

- Object Explorer:** Shows the database structure, including tables like Aman\_Sinha\_DB, Billy\_Chen\_DB, Buji\_James\_DB, Carolina\_Salgueiro\_DB, Dez\_Philips\_DB, Francis\_Asonamning\_DB, Group\_1\_DB, and various system and external resources.
- Execution Plan:** Shows the execution plan for the query.
- Results:** Displays the output of the query:
 

PROMO_NAME	SALES_YEAR	TOTAL_QUANTITY SOLD	REGION
internet promotion #29-350	2012	5511	Americas
internet promotion #29-350	2012	1027	Asia
internet promotion #29-350	2012	1657	Europe
internet promotion #29-350	2012	79	Oceania
internet promotion #29-350	2014	31	Africa
internet promotion #29-350	2014	2213	Americas
internet promotion #29-350	2014	60	Asia
internet promotion #29-350	2014	80	Europe
internet promotion #29-350	2015	295	Americas
internet promotion #29-350	2015	35	Asia
internet promotion #29-350	2015	116	Europe
internet promotion #29-350	2015	10	Oceania
post promotion #20-33	2012	5	Americas
post promotion #20-33	2012	6344	Americas
post promotion #20-33	2012	50	Asia
post promotion #20-33	2012	300	Europe
post promotion #20-33	2012	39	Oceania
TV promotion #13-351	2015	33105	Americas
TV promotion #13-351	2015	4996	Asia
- Messages:** Shows the message "Query executed successfully."
- Status Bar:** Shows the session ID (134.173.236.23), the database (202DBFall\_Group\_1\_DB), and the time (00:00:00).

```
/*
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'
```

PRODUCT NAME	TOTAL QUANTITY SOLD
"1.44MB External 3.5" Diskette"	118
"1.44MB External 3.5" Floppy"	35
"18" Flat Panel Graphics Monitor"	278
"3 1/2" Bulk deletes. Box of 100"	36
"3 1/2" Bulk deletes. Box of 50"	48
"CD-R with Jewel Case, Pack OF 12"	296
"CD-R Professional Grade, Pack of 10"	110
"CD-RW, High Speed Pack of 5"	800
"CD-RW, High Speed, Pack of 10"	71
"DVD-R Disc with Jewel Case, 4.7 GB"	2166
"DVD-R Disc, 4.7GB, Pack of 5-V2"	437
"DVD-R Disc, 4.7GB, Pack of 5"	210
"DVD-RAM Jewel Case, Double-Sided, 9.4GB"	114
"DVD-RW Disc, 4.7GB, Pack of 3"	465
"Mens DV Camcorder with 3.5" Swivel LCD"	273
"Multimedia speakers 5" cones"	145
"Multimedia speakers 5" cones"	67
"OakMusic CD-R, Pack of 10"	171
128MB Memory Card	246

Query executed successfully.

## 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')

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].[PROPOPULAR_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”, “MARITAL\_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

```

ExecutionPlan.sql - 134.173.236.23.2022DBFall_Group_1_DB (2022DBFall_Pankaj_Chaudhuri (70)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
2022DBFall_Group_1_DB | Execute | 
Object Explorer
Connect + 
2022DBFall_Aman_Sinha_DB
2022DBFall_Billy_Chen_DB
2022DBFall_Bulk_James_DB
2022DBFall_Carolina_Salgueiro_DB
2022DBFall_Der_Philiips_DB
2022DBFall_Group_1_DB
2022DBFall_Group_2_DB
Database Diagrams
Tables
Views
System Views
    dbo.CUSTOMER_TRENDS_VW
    dbo.PRODUCTS_SHPOADS_VW
    dbo.PROPOPULAR_PRODUCTS_VW
    dbo.REVENUE_ANALYSIS_AVG_VW
    dbo.REVENUE_ANALYSIS_VW
    dbo.SALES_PROMOTIONS_VW
External Resources
Synonyms
Availability Groups
Service Broker
Storage
Security
2022DBFall_Group_2_DB
2022DBFall_Group_3_DB
2022DBFall_Group_4_DB
2022DBFall_Group_5_DB
2022DBFall_Jesse_Carmes_DB
2022DBFall_Juan_Perilla_DB
2022DBFall_Karen_Kummer_DB
2022DBFall_Kasturi_Gavali_DB
2022DBFall_Kathy_Ernes_DB
2022DBFall_Laura_Wilson_DB
2022DBFall_Maria_Karampoglu_DB
2022DBFall_Mark_Suttor_DB
2022DBFall_Nader_Moslem_DB
2022DBFall_Navya_Natani_DB
2022DBFall_Oke_Omwukwe_DB
2022DBFall_Oluigbenega_Emoila_DB
2022DBFall_Pankaj_Chaudhuri_DB
2022DBFall_Patrick_Wilkman_DB
2022DBFall_Phuong_Vu_DB
2022DBFall_Raad_Alighandi_DB
2022DBFall_Sanjana_Shashidhara_DB
2022DBFall_Yuri_Yu_DB
ExecutionPlan.sql - 134.173.236.23.2022DBFall_Group_1_DB (2022DBFall_Pankaj_Chaudhuri (70)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
2022DBFall_Group_1_DB | Execute | 
ExecutionPlan.sql - 134.173.236.23.2022DBFall_Group_1_DB (2022DBFall_Pankaj_Chaudhuri (70)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
2022DBFall_Group_1_DB | Execute | 
4 SET STATISTICS TIME ON
5 GO
6
7 SET SHOWPLAN_ALL ON
8 GO
9
10 SELECT s.PROD_ID AS [PRODUCT ID],
11     YEAR(s.SALE_DATE) AS [YEAR],
12     MONTH(s.SALE_DATE) AS [MONTH],
13     SUM(s.AMOUNT SOLD) AS [Total Amount Sold],
14     SUM(s.QUANTITY SOLD) AS [Total Quantity Sold],
15     cc.COUNTRY_REGION AS REGION,
16     COUNT_BIG(*) AS [Count]
17 FROM dbo.SALES AS s
18     JOIN dbo.CUSTOMERS AS cust
19         ON cust.CUST_ID = s.CUST_ID
20     JOIN dbo.CUSTOMERS_COUNTRY AS cc
21         ON cust.COUNTRY_ID = cc.COUNTRY_ID
22 GROUP BY s.PROD_ID, YEAR(s.SALE_DATE), MONTH(s.SALE_DATE), cc.COUNTRY_REGION
23 GO
121 %
121 %
Results Messages
SQL Server parse and compile time:
CPU time = 7 ms, elapsed time = 7 ms.

(10987 rows affected)

SQL Server Execution Times:
CPU time = 31 ms, elapsed time = 179 ms.

Completion time: 2022-12-12T15:22:52.9599210-08:00
|
121 %
121 %
Query executed successfully.
134.173.236.23 (14.0 RTM) 2022DBFall_Pankaj_Cha... 2022DBFall_Group_1_DB 00:00:00 10,987 rows

```

## Business View 1 - Plan 2

```

ExecutionPlan.sql - 134.173.236.23.2022DBFall_Group_1_DB (2022DBFall_Pankaj_Chaudhuri (70)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
2022DBFall_Group_1_DB | Execute | 
ExecutionPlan.sql - 134.173.236.23.2022DBFall_Group_1_DB (2022DBFall_Pankaj_Chaudhuri (70)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
2022DBFall_Group_1_DB | Execute | 
ExecutionPlan.sql - 134.173.236.23.2022DBFall_Group_1_DB (2022DBFall_Pankaj_Chaudhuri (70)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
2022DBFall_Group_1_DB | Execute | 
10 SELECT s.PROD_ID AS [PRODUCT ID],
11     YEAR(s.SALE_DATE) AS [YEAR],
12     MONTH(s.SALE_DATE) AS [MONTH],
13     SUM(s.AMOUNT SOLD) AS [Total Amount Sold],
14     SUM(s.QUANTITY SOLD) AS [Total Quantity Sold],
15     cc.COUNTRY_REGION AS REGION,
16     COUNT_BIG(*) AS [Count]
17 FROM dbo.SALES AS s
18     JOIN dbo.CUSTOMERS AS cust
19         ON cust.CUST_ID = s.CUST_ID
20     JOIN dbo.CUSTOMERS_COUNTRY AS cc
21         ON cust.COUNTRY_ID = cc.COUNTRY_ID
22     GROUP BY s.PROD_ID, YEAR(s.SALE_DATE), MONTH(s.SALE_DATE), cc.COUNTRY_REGION
23 GO
24
25
26 SELECT * FROM [dbo].[REVENUE_ANALYSIS_AVG_VW]
27 GO
28
29 SELECT p.PROD_NAME AS [PRODUCT NAME],
30     AVG(p.AVG_SALE) AS [AVG_SALE]
31 FROM [dbo].[REVENUE_ANALYSIS_AVG_VW] p
32 WHERE p.PROD_ID = s.PROD_ID
33 GROUP BY p.PROD_NAME
34
35
36 Results Messages
SQL Server parse and compile time:
CPU time = 0 ms, elapsed time = 7 ms.

(10987 rows affected)

SQL Server Execution Times:
CPU time = 16 ms, elapsed time = 41 ms.

Completion time: 2022-12-12T15:23:33.1309542-08:00
|
121 %
121 %
Query executed successfully.
134.173.236.23 (14.0 RTM) 2022DBFall_Pankaj_Cha... 2022DBFall_Group_1_DB 00:00:00 10,987 rows

```

## Business View 5 - Plan 1

```

28
29   SELECT p.PROD_NAME AS [PRODUCT_NAME],
30         SUM(s.QUANTITY SOLD) AS [TOTAL QUANTITY SOLD],
31         c.CUST_GENDER AS [GENDER],
32         c.CUST_MARITAL_STATUS AS [MARITAL STATUS],
33         cc.COUNTRY_REGION AS REGION,
34         COUNT_BIG() AS [COUNT]
35     FROM dbo.CUSTOMERS AS c
36     JOIN dbo.SALES AS s
37       ON c.CUST_ID = s.CUST_ID
38     JOIN dbo.CUSTOMERS_COUNTRY AS cc
39       ON c.COUNTRY_ID = cc.COUNTRY_ID
40     JOIN dbo.PRODUCTS AS p
41       ON p.PROD_ID = s.PROD_ID
42     GROUP BY p.PROD_NAME, c.CUST_GENDER, c.CUST_MARITAL_STATUS, cc.COUNTRY_REGION
43   GO
44
45   SELECT * FROM [dbo].[CUSTOMER_TREND_IN_VW]
46   GO
47
48   SET SHOWPLAN_ALL OFF
49
50   SET STATISTICS TIME OFF
51   GO

```

121 % 134.173.236.23 (14.0 RTM) 2022DBFall\_Pankaj\_Chaudhuri 2022DBFall\_Group\_1\_DB 00:00:00 2,577 rows

Query executed successfully.

## Business View 5 - Plan 2

```

37
38   JOIN dbo.CUSTOMERS_COUNTRY AS cc
39     ON c.COUNTRY_ID = cc.COUNTRY_ID
40   JOIN dbo.PRODUCTS AS p
41     ON p.PROD_ID = s.PROD_ID
42   GROUP BY p.PROD_NAME, c.CUST_GENDER, c.CUST_MARITAL_STATUS, cc.COUNTRY_REGION
43   GO
44
45   SELECT * FROM [dbo].[CUSTOMER_TREND_IN_VW]
46   GO
47
48   SET SHOWPLAN_ALL OFF
49
50   SET STATISTICS TIME OFF
51   GO

```

121 % 134.173.236.23 (14.0 RTM) 2022DBFall\_Pankaj\_Chaudhuri 2022DBFall\_Group\_1\_DB 00:00:00 2,577 rows

Query executed successfully.

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

The screenshot shows the SQL Server Management Studio interface with the following details:

- Object Explorer:** Shows the database structure, including tables like `SALES`, `CUSTOMERS`, and `COUNTRIES`.
- SQL Query Editor:** Contains the T-SQL code for the query:
 

```

USE [2022DBFall_Group_1_DB]
GO

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
      
```
- Execution Plan:** A detailed diagram showing the flow of data through various operators like Parallelism, Hash Match, and Compute Scalar. The estimated cost is 12.61.
- Status Bar:** Shows the execution time as 14.0 RTM and the number of rows as 0 rows.

## Business View 1 - Plan 2

The screenshot shows the SQL Server Management Studio interface with the following details:

- Object Explorer:** Shows the database structure, including tables like `SALES`, `CUSTOMERS`, and `COUNTRIES`.
- SQL Query Editor:** Contains the T-SQL code for the query:
 

```

USE [2022DBFall_Group_1_DB]
GO

SELECT * FROM [dbo].[REVENUE_ANALYSIS_AVG_WW]
      
```
- Execution Plan:** A simplified diagram showing a single step: a `Compute Scalar` followed by a `Clustered Index Scan` on the `REVENUE\_ANALYSIS\_AVG\_WW` view. The estimated cost is 0.078.
- Status Bar:** Shows the execution time as 14.0 RTM and the number of rows as 0 rows.

## Business View 5 - Plan 1

SQLQuery2.log - 134.173.236.23|2022DBFall\_Group\_1\_DB (2022DBFall\_Pankaj\_Chaudhari (55)) - Microsoft SQL Server Management Studio

```

1 USE [2022DBFall_Group_1_DB]
2 GO
3
4 SELECT p.PROD_NAME AS [PRODUCT NAME],
5     SUM(s.QUANTITY SOLD) AS [TOTAL QUANTITY SOLD],
6     c.CUST_GENDER AS [GENDER],
7     c.CUST_MARITAL_STATUS AS [MARITAL STATUS],
8     cc.COUNTRY_REGION AS REGION,
9     COUNT_BIG(*) AS [COUNT]
10    FROM dbo.CUSTOMERS AS c
11    JOIN dbo.SALES AS s
12        ON c.CUST_ID = s.CUST_ID
13    JOIN dbo.CUSTOMERS_COUNTRY AS cc
14        ON c.COUNTRY_ID = cc.COUNTRY_ID
15    JOIN dbo.PRODUCTS AS p
16        ON p.PROD_ID = s.PROD_ID
17    GROUP BY p.PROD_NAME, c.CUST_GENDER, c.CUST_MARITAL_STATUS, cc.COUNTRY_REGION
18
19
20

```

Estimated Execution Plan:

The execution plan shows a parallel query with four parallelism paths. It starts with a hash match between Sales and Customers, followed by joins with Customers\_Country and Products. The final group by operation is also parallel.

Execution Plan Statistics:

Statistic	Value
Cached plan size	160 KB
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	10.6033
Estimated Number of Rows for All Executions	0
Estimated Number of Rows Per Execution	4320

## Business View 5 - Plan 2

SQLQuery2.log - 134.173.236.23|2022DBFall\_Group\_1\_DB (2022DBFall\_Pankaj\_Chaudhari (55)) - Microsoft SQL Server Management Studio

```

1 USE [2022DBFall_Group_1_DB]
2 GO
3
4 SELECT * FROM [dbo].[CUSTOMER_TREND_IN_VW]
5
6
7

```

Estimated Execution Plan:

The execution plan shows a simple SELECT statement from a view named CUSTOMER\_TREND\_IN\_VW. The cached plan size is 24 KB.

Execution Plan Statistics:

Statistic	Value
Cached plan size	24 KB
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	0.022413
Estimated Number of Rows for All Executions	0
Estimated Number of Rows Per Execution	4320

Message Log:

Query executed successfully.

## 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, 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	SALETRANS_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	SALETRANS_ID, PROD_ID, CUST_ID, CHANNEL_ID, PROMO_ID, SALE_DATE, SHIPPING_DATE, PAYMENT_DATE, QUANTITY SOLD, AMOUNT SOLD, UNIT PRICE