Dimensional Model Documentation

1. Introduction:

During this part of the FlipKart Data Warehouse project, I created and set up a dimensional model using the standardized database that I had already made. The dimensional model uses a star schema design pattern, which is very helpful for business data and analytical processing. I turned the normalized structure into a model that works best for complicated queries, reporting, and data analysis by making smart design choices.

2. Dimensional Model Overview:

There are six dimension tables and two fact tables in the dimensional model. The facts have the business measures and foreign keys to the dimensions. The dimensions hold the main business units and descriptive attributes. This layout makes it easy to find your way around the data and runs queries quickly.

```sql
-- Verify the existence and row counts of all dimension and fact tables
SELECT
    'Dimension Tables' AS TableType,
    (SELECT COUNT(*) FROM INFORMATION_SCHEMA.TABLES
     WHERE TABLE_NAME LIKE 'dim_%' AND TABLE_TYPE = 'BASE TABLE') AS TableCount
UNION ALL
SELECT
    'Fact Tables',
    (SELECT COUNT(*) FROM INFORMATION_SCHEMA.TABLES
     WHERE TABLE_NAME LIKE 'fact_%' AND TABLE_TYPE = 'BASE TABLE')
UNION ALL
SELECT '----------------------', NULL
UNION ALL
SELECT 'dim_Customer', COUNT(*) FROM dim_Customer
UNION ALL
SELECT 'dim_Product', COUNT(*) FROM dim_Product
```

100 %

Results | Messages

	TableType	TableCount
1	Dimension Tables	6
2	Fact Tables	2
3	----------------------	NULL
4	dim_Customer	1000
5	dim_Product	1000
6	dim_Location	999
7	dim_Date	5844
8	dim_PaymentMethod	4
9	dim_ShippingMethod	3
10	----------------------	NULL
11	fact_Sales	1000
12	fact_CustomerPurchaseBehavior	1000

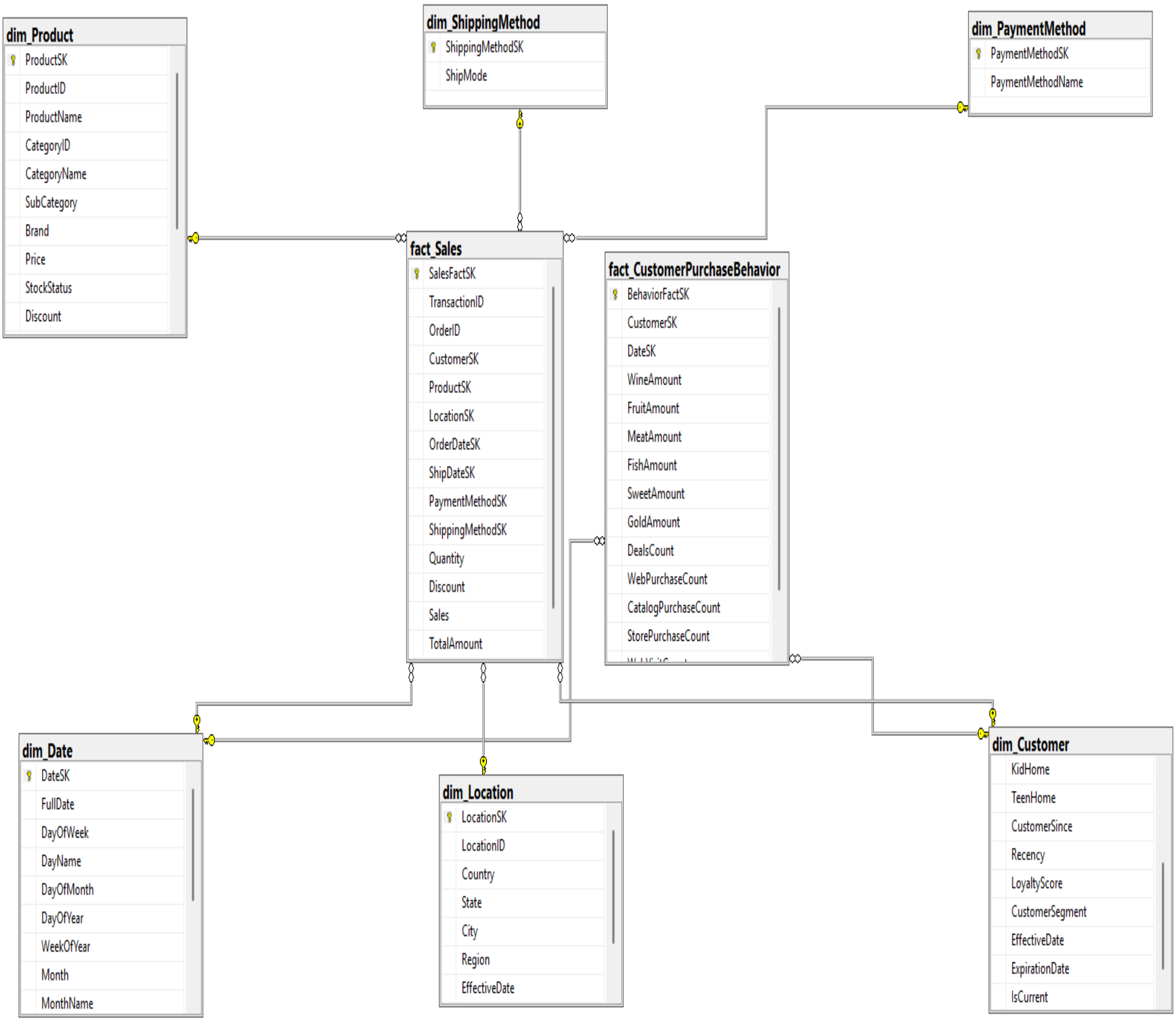
2.1 Verification of Fact Tables and Dimensions

3. Star Schema Design:

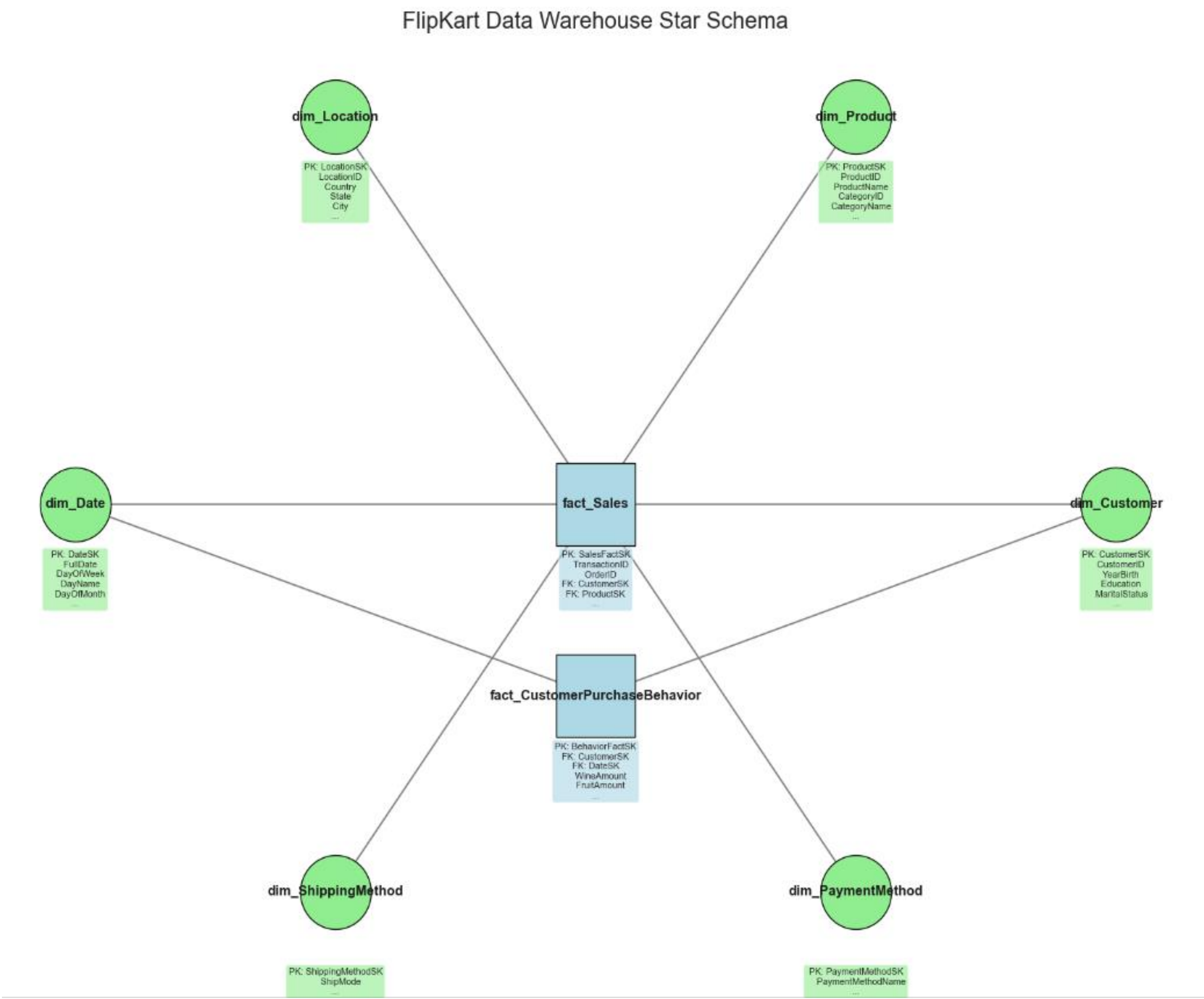
For this project, I chose a star schema method because it achieves the best balance between query speed and ease of understanding. In this plan:

- The fact tables are in the middle of the model.
- Like the points on a star, dimension tables go around the facts.
- Surrogate keys show how facts and measurements are related to each other.
- Dimensions are denormalized to improve the speed of queries.

The following database diagram makes the star system design clear:



3.1 Star Schema Diagram with Dimensions and Fact tables



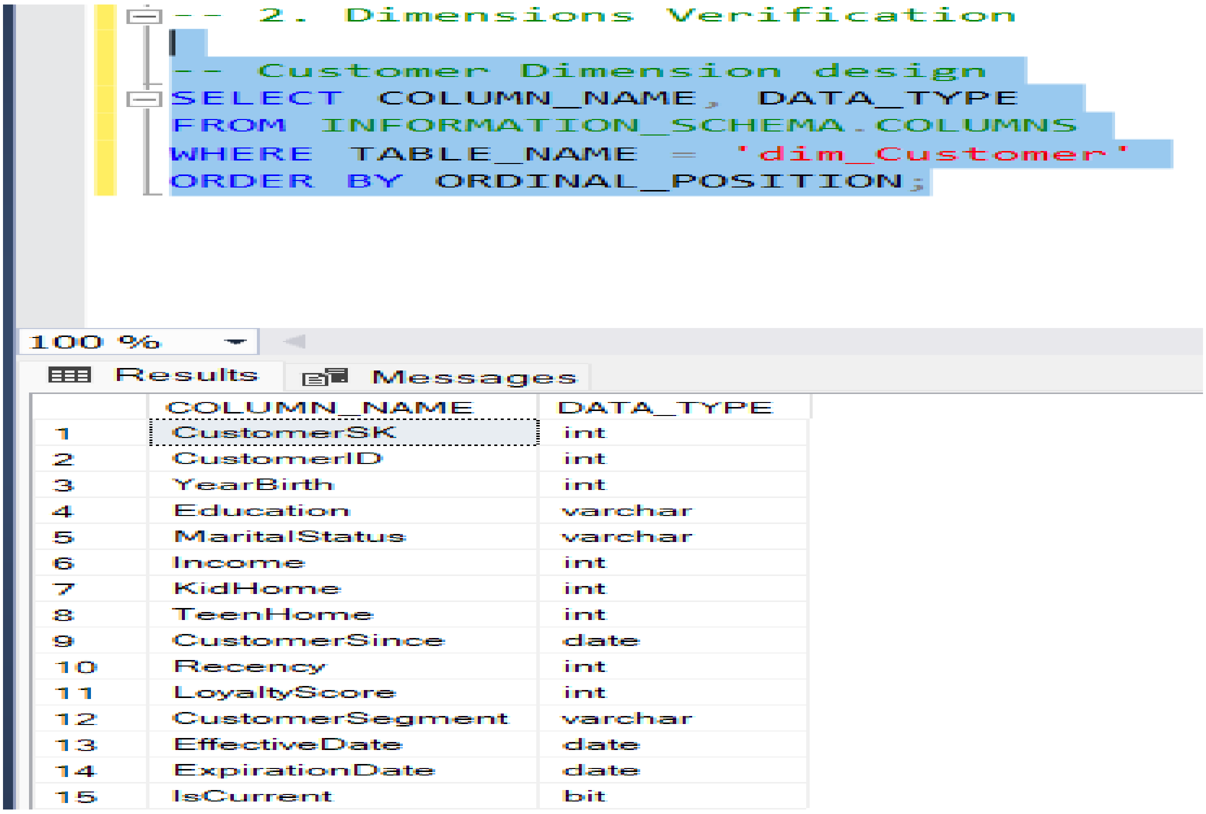
3.2 Visualized Star Schema Diagram

4. Dimension Tables Creation and Design:

It was carefully planned that each dimension table would support analytical queries while keeping the purity of the data. Through ETL processes like data transformation, enrichment, and denormalization, the dimensions were generated from the normalized database tables.

<u>a. Customer Dimension:</u>

The Customer dimension has information about the types of customers and derived attributes that can be used for business research.



```
-- 2. Dimensions Verification

-- Customer Dimension design
SELECT COLUMN_NAME, DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'dim_Customer'
ORDER BY ORDINAL_POSITION;
```

100 %

Results | Messages

	COLUMN_NAME	DATA_TYPE
1	CustomerSK	int
2	CustomerID	int
3	YearBirth	int
4	Education	varchar
5	MaritalStatus	varchar
6	Income	int
7	KidHome	int
8	TeenHome	int
9	CustomerSince	date
10	Recency	int
11	LoyaltyScore	int
12	CustomerSegment	varchar
13	EffectiveDate	date
14	ExpirationDate	date
15	IsCurrent	bit

4.1 Verifying Customer Dimension

We achieved the following:

- SCD Type 2 was put in place to keep track of changes to customer data over time.
- Based on trust scores, the derived CustomerSegment attribute sorts customers into three groups: Premium, Standard, and Basic.
- Keeps the original key (CustomerID) but adds a substitute key (CustomerSK).
- Added all the necessary customer information from the original standardized Customers table

b. Product Dimension:

Information about products and categories are put together in a denormalized way in the Product dimension.

```sql
-- Product Dimension design with derived attributes
SELECT TOP 5
      ProductSK,
      ProductID,
      CategoryName,
      StockStatus
FROM dim_Product;
```

100 %

Results | Messages

	ProductSK	ProductID	CategoryName	StockStatus
1	1	P3001	Furniture	High Stock
2	2	P3002	Electronics	Low Stock
3	3	P3003	Office Supplies	High Stock
4	4	P3004	Office Supplies	High Stock
5	5	P3005	Office Supplies	High Stock

4.2 Verifying Product Dimension

We achieved the following:

- Set up SCD Type 2 to keep track of changes to products
- Category data from the different Categories table that has been denormalized
- Added a derived StockStatus attribute to mark the amount of inventory
- Both the product key and the category key were saved for future use.

c. Location Dimension:

The Location dimension gives you a way to look at sales by area based on their location.

```sql
-- Location Dimension with derived region
SELECT TOP 5
    LocationSK,
    Country,
    State,
    City,
    Region
FROM dim_Location;
```

100 %

Results | Messages

	LocationSK	Country	State	City	Region
1	1	USA	Alabama	Austinchester	North America
2	2	USA	Alabama	Austinhaven	North America
3	3	USA	Alabama	Chrischester	North America
4	4	USA	Alabama	Dixonbury	North America
5	5	USA	Alabama	East Christopher	North America

4.3 Verifying Location Dimension

We achieved the following:

- Set up SCD Type 2 to keep track of changes in position
- Added the derived Region property to group locations by
- Maintained the order of geography (Country > State > City)

d. Date Dimension:

For time-based analysis, the Date dimension gives you characteristics that are based on time.

```
-- Date Dimension structure
SELECT
    COUNT(*) as TotalDateRecords,
    MIN(FullDate) as MinDate,
    MAX(FullDate) as MaxDate,
    COUNT(DISTINCT Year) as YearCount
FROM dim_Date;
```

100 %

Results | Messages

	TotalDateRecords	MinDate	MaxDate	YearCount
1	5844	2010-01-01	2025-12-31	16

4.4 Verifying Date Dimension

We achieved the following:

- Made a full range of dates to help with study of the past and the future
- Date traits like day of the week, month, quarter, and year have been added.
- Added special date flags for weekends and holidays
- Used a surrogate key based on dates to make joins go quickly.

Similarly, we created other 2 Dimensions: **Payment Method Dimension** and Shipping Method Dimension.

```sql
-- Payment and Shipping Method dimensions
SELECT 'dim_PaymentMethod' AS Dimension, COUNT(*) AS Count FROM dim_PaymentMethod
UNION ALL
SELECT 'dim_ShippingMethod', COUNT(*) FROM dim_ShippingMethod;
```

100 %

Results | Messages

	Dimension	Count
1	dim_PaymentMethod	4
2	dim_ShippingMethod	3

4.5 Verifying Payment Method and Shipping Method Dimension

5. Fact Tables Creation and Design:

I made two fact tables with different analytical needs in mind. Each one connects to more than one dimension using "surrogate keys."

a. Sales Fact Table:

There are transaction-level data with measures that are useful for sales research in the Sales fact table.

```sql
-- Sales Fact table structure with dimension relationships
SELECT
    'Customer Dimension' as Relationship,
    COUNT(DISTINCT fs.CustomerSK) as FactDistinctCount,
    (SELECT COUNT(*) FROM dim_Customer) as DimTotalCount
FROM fact_Sales fs
UNION ALL
SELECT 'Product Dimension', COUNT(DISTINCT fs.ProductSK), (SELECT COUNT(*) FROM dim_Product) FROM fact_Sales fs
UNION ALL
SELECT 'Location Dimension', COUNT(DISTINCT fs.LocationSK), (SELECT COUNT(*) FROM dim_Location) FROM fact_Sales fs;
```

100 %

Results | Messages

	Relationship	FactDistinctCount	DimTotalCount
1	Customer Dimension	637	1000
2	Product Dimension	615	1000
3	Location Dimension	999	999

5.1 Verifying Fact Sales Tables

We achieved the following:

- Set up a transaction grain (the most specific level)
- Measurements were given for number, sales amount, and discount.
- Connected to all factors for a full analysis
- Used more than one date (order date and ship date)

<u>b. Customer Purchase Behavior Fact Table:</u>

This fact table shows how customers usually buy different types of products.

```sql
-- Customer Purchase Behavior Fact measures
SELECT
    AVG(WineAmount) as AvgWineAmount,
    AVG(FruitAmount) as AvgFruitAmount,
    AVG(MeatAmount) as AvgMeatAmount,
    AVG(TotalSpend) as AvgTotalSpend,
    MAX(WebPurchaseCount) as MaxWebPurchases,
    MAX(StorePurchaseCount) as MaxStorePurchases
FROM fact_CustomerPurchaseBehavior;
```

100 %

Results | Messages

	AvgWineAmount	AvgFruitAmount	AvgMeatAmount	AvgTotalSpend	MaxWebPurchases	MaxStorePurchases
1	247	100	150	677.450000	10	10

5.2 Verifying Customer Purchase Behavior Fact Table

We achieved the following:

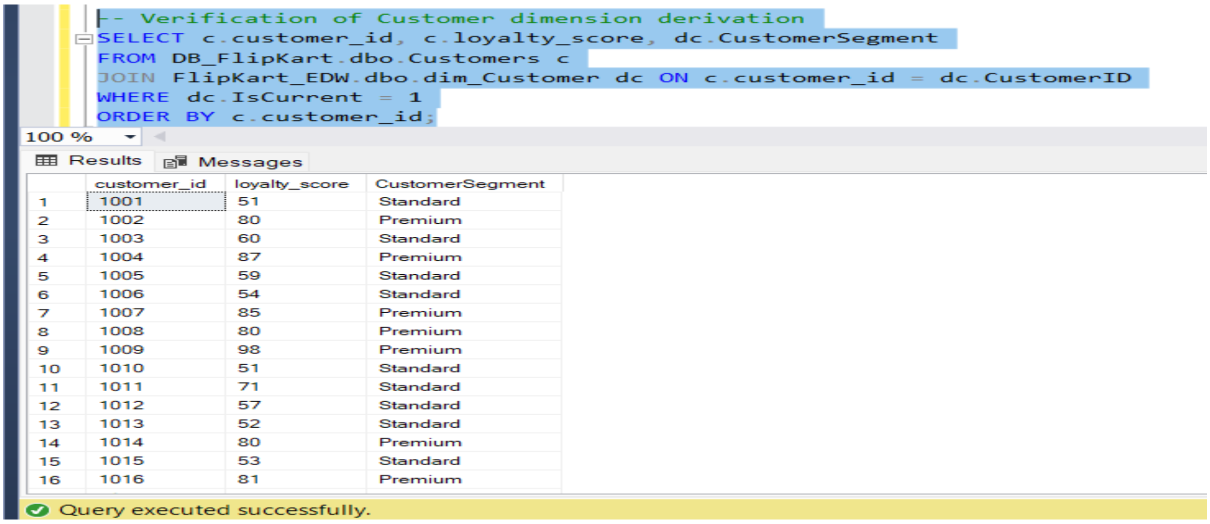
- Set up a customer grain (one row for each customer).
- Added measures of spending by type of product
- Extra steps were added for various methods of purchase
- Linked to measurements that were agreed upon (Customer and Date)

6. Explanation of How Dimensions Were Derived from Normalized Sources:

In the dimensional modeling step, the process of getting dimensions from normalized source tables was a big change. This wasn't an easy one-to-one mapping; it involved a lot of complicated changes, like making a surrogate key, denormalizing, deriving attributes, and putting in place a slowly changing dimension methodology. Let me go over this process in more depth for each measure.

A. Customer Dimension Derivation

One of the most difficult parts of our dimensional model is the Customer dimension. It mostly came from the standardized DB_FlipKart.dbo.Customers table, but it was changed in important ways:



```
-- Verification of Customer dimension derivation
SELECT c.customer_id, c.loyalty_score, dc.CustomerSegment
FROM DB_FlipKart.dbo.Customers c
JOIN FlipKart_EDW.dbo.dim_Customer dc ON c.customer_id = dc.CustomerID
WHERE dc.IsCurrent = 1
ORDER BY c.customer_id;
```

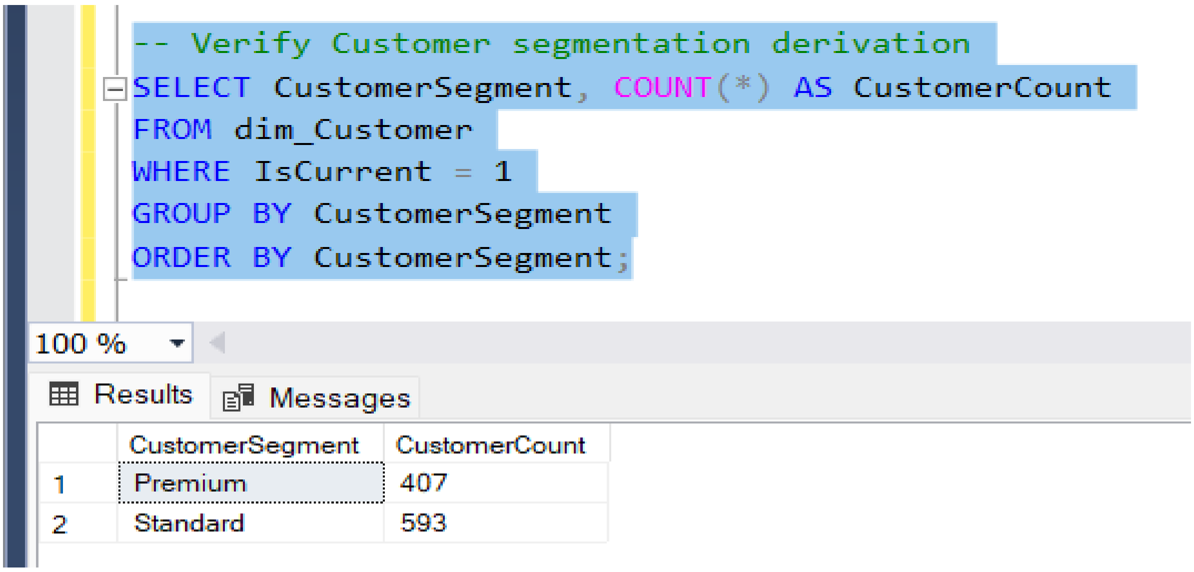
100 %

Results | Messages

	customer_id	loyalty_score	CustomerSegment
1	1001	51	Standard
2	1002	80	Premium
3	1003	60	Standard
4	1004	87	Premium
5	1005	59	Standard
6	1006	54	Standard
7	1007	85	Premium
8	1008	80	Premium
9	1009	98	Premium
10	1010	51	Standard
11	1011	71	Standard
12	1012	57	Standard
13	1013	52	Standard
14	1014	80	Premium
15	1015	53	Standard
16	1016	81	Premium

Query executed successfully.

6.1 Verify Customer Dimension Derivation



```
-- Verify Customer segmentation derivation
SELECT CustomerSegment, COUNT(*) AS CustomerCount
FROM dim_Customer
WHERE IsCurrent = 1
GROUP BY CustomerSegment
ORDER BY CustomerSegment;
```

100 %

Results | Messages

	CustomerSegment	CustomerCount
1	Premium	407
2	Standard	593

6.2 Verify Customer segmentation derivation

The transformation involved:

- Holding on to all data information from the source

- Adding tracking fields for SCD Type 2 (EffectiveDate, ExpirationDate, and IsCurrent)
- Using business logic to turn customer loyalty scores into useful groups
- Making a substitute key that is different from the normal key
- Setting up the system to keep track of changes made to customer traits over time

B. Product Dimension Derivation

The Product dimension was made by denormalizing data from two normalized tables, DB_FlipKart.dbo.Products and DB_FlipKart.dbo.Categories, and then putting them together. This change got rid of the need for joins in analytical queries while keeping the link in a hierarchy:

```sql
-- Verification of Product dimension denormalization
SELECT
    p.product_id,
    p.product_name,
    c.category_name,
    c.sub_category,
    dp.StockStatus
FROM DB_FlipKart.dbo.Products p
JOIN DB_FlipKart.dbo.Categories c ON p.category_id = c.category_id
JOIN FlipKart_EDW.dbo.dim_Product dp ON p.product_id = dp.ProductID
WHERE dp.IsCurrent = 1
ORDER BY p.product_id;
```

100 %

Results | Messages

	product_id	product_name	category_name	sub_category	StockStatus
1	P3001	Fight Desk	Furniture	Smartphones	High Stock
2	P3002	Threat Monitor	Electronics	Laptops	Low Stock
3	P3003	No Laptop	Office Supplies	Laptops	High Stock
4	P3004	Buy Monitor	Office Supplies	Laptops	High Stock
5	P3005	Describe Laptop	Office Supplies	Printers	High Stock
6	P3006	Instead Phone	Office Supplies	Chairs	High Stock
7	P3007	Which Monitor	Electronics	Chairs	High Stock
8	P3008	Less Laptop	Furniture	Printers	High Stock
9	P3009	Idea Desk	Furniture	Smartphones	High Stock
10	P3010	Civil Desk	Office Supplies	Tables	High Stock
11	P3011	Make Laptop	Office Supplies	Printers	High Stock
12	P3012	Behavior Phone	Electronics	Chairs	High Stock
13	P3013	Behavior Monitor	Electronics	Chairs	High Stock
14	P3014	No Printer	Office Supplies	Laptops	High Stock
15	P3015	Tell Laptop	Electronics	Laptops	High Stock
16	P3016	Even Monitor	Furniture	Printers	High Stock

Query executed successfully.

6.3 Verification of Product dimension denormalization

```sql
-- Verify Stock Status derivation
SELECT StockStatus, COUNT(*) AS ProductCount
FROM dim_Product
WHERE IsCurrent = 1
GROUP BY StockStatus
ORDER BY ProductCount DESC;
```

100 %

Results | Messages

	StockStatus	ProductCount
1	High Stock	821
2	Medium Stock	104
3	Low Stock	75

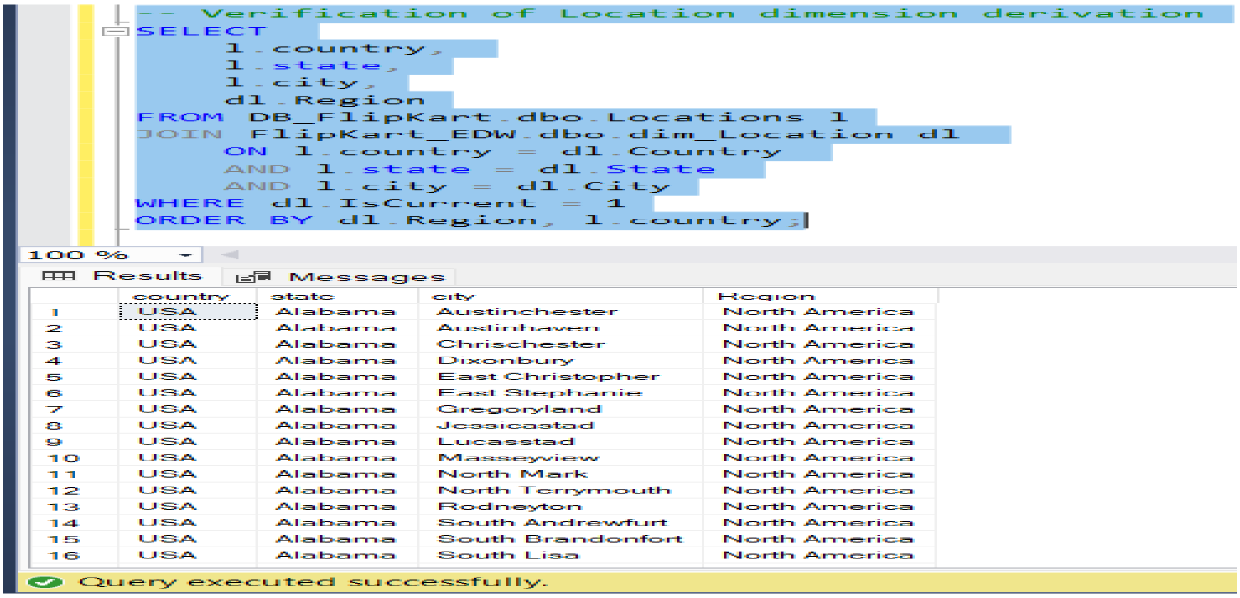
6.4 Verify Stock Status derivation

The transformation involved:

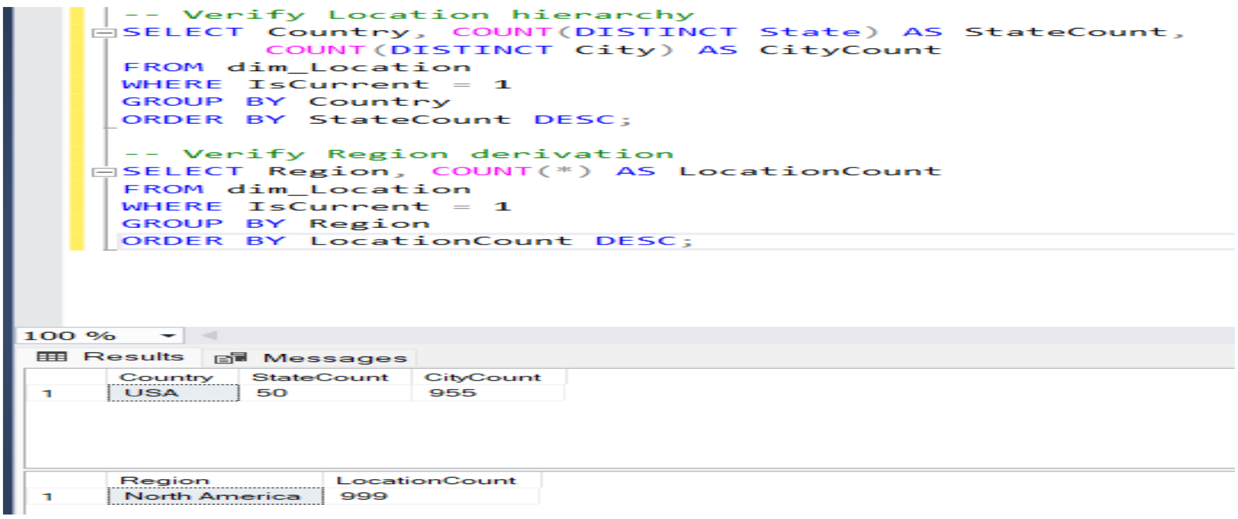
- Putting together facts about a product and its category
- Getting rid of the need for a foreign key link during query time
- Using the SCD Type 2 way to keep track of the past
- Making substitute keys for fact table joins that work better

C. Location Dimension Derivation

The Location dimension was Derived from the normalized DB_FlipKart.dbo.Locations table and shows regional information. This change was mostly about adding useful business classifications to geographic data:



6.5 Verification of Location dimension derivation



6.6 Verifying Location Hierarchy and Region Derivation

Transformation involved:

- Geographical order was kept (Country > State > City).
- Added a derived Region property that is based on how countries are grouped.
- It now has SCD Type 2 traits.
- Made a substitute key for dimension links

D. Date Dimension Creation

The Date dimension wasn't drawn from a normalized source like the others; instead, it was made using a set of steps. This method is often used for date measurements because they need certain calculations and attributes:

```sql
-- Verification of Date dimension attributes
SELECT
    Year,
    COUNT(*) AS TotalDays,
    SUM(CASE WHEN IsWeekend = 1 THEN 1 ELSE 0 END) AS WeekendDays,
    SUM(CASE WHEN IsHoliday = 1 THEN 1 ELSE 0 END) AS Holidays
FROM dim_Date
GROUP BY Year
ORDER BY Year;
```

100 %

Results | Messages

	Year	TotalDays	WeekendDays	Holidays
1	2010	365	104	3
2	2011	365	105	3
3	2012	366	105	3
4	2013	365	104	3
5	2014	365	104	3
6	2015	365	104	3
7	2016	366	105	3
8	2017	365	105	3
9	2018	365	104	3
10	2019	365	104	3
11	2020	366	104	3
12	2021	365	104	3
13	2022	365	105	3
14	2023	365	105	3
15	2024	366	104	3
16	2025	365	104	3

Query executed successfully.

6.7 Date Dimension Verification

Changes that were made:

- Date records were made for a certain set of dates.
- The day, month, quarter, and year of the date were calculated.
- Added special colors for holidays and weekends
- Made a substitute key based on dates

E. Payment and Shipping Method Dimensions

The different numbers in the OrderDetails table were used to get these dimensions.

```sql
-- Verify Payment Methods
SELECT PaymentMethodName,
       COUNT(*) AS TransactionCount
FROM dim_PaymentMethod pm
JOIN fact_Sales fs ON pm.PaymentMethodSK = fs.PaymentMethodSK
GROUP BY PaymentMethodName
```

100 %

Results | Messages

	PaymentMethodName	TransactionCount
1	Bank Transfer	263
2	PayPal	261
3	Credit Card	243
4	Debit Card	233

6.8 Verifying Payment Method Derivation

```sql
-- Verify Shipping Methods
SELECT ShipMode,
       COUNT(*) AS TransactionCount
FROM dim_ShippingMethod sm
JOIN fact_Sales fs ON sm.ShippingMethodSK = fs.ShippingMethodSK
GROUP BY ShipMode
ORDER BY TransactionCount DESC;
```

100 %

Results | Messages

	ShipMode	TransactionCount
1	Standard	341
2	Express	337
3	Same Day	322

6.9 Verifying Shipping Method Derivation

Transformation involved:

- Getting unique method values
- Made substitute keys
- Original method names were kept.

I successfully changed a normalized database that was best for transaction processing into a dimensional model that was best for analysis queries by using these changes.

Checking with the verification queries makes sure that all dimensions were correctly obtained from normalized sources while keeping the integrity of the data. Beyond the source data, each dimension now has business-meaningful characteristics that weren't there in the original normalized form. This adds a lot of analytical value.

7. Dimensional Bus Matrix

Below is the bus matrix showing the relationship between fact tables and dimensions in our FlipKart data warehouse:

Fact Table/ Dimension	Customer *	Product	Location	Date*	Payment Method	Shipping Method
fact_Sales	✓	✓	✓	✓	✓	✓
fact_CustomerPurchaseBehavior	✓			✓		

Conforming Dimensions(*):

1. Customer*: This is shared between both fact tables so that customer-centered analysis can be done across all business activities.

2. Date*: This is used by both fact tables to look at data over time and see trends.

Division-Specific Dimensions

1. Product—Used only for analyzing sales transactions
2. Location—Only supports sales research by geography
3. Payment Method: This is only used for sales.
4. Shipping Method—Only for sales deals

Hierarchical Relationships

1. **Product**: Category → Subcategory → Product
2. **Location**: Region → Country → State → City
3. **Date**: Year → Quarter → Month → Day

This bus matrix is like a plan for our dimensional model. It shows which dimensions can be used for each business process and which dimensions can be used to analyze multiple fact tables together.

8. Conclusion

From our normalized database, the dimensional model implementation met all needs and created a complex analytical structure. Six dimension tables (Customer, Product, Location, Date, PaymentMethod, ShippingMethod) and two fact tables (fact_Sales, fact_CustomerPurchaseBehavior) were connected by surrogate keys in a star structure.

The model uses Slowly Changing Dimension Type 2 for historical tracking, conformed dimensions for cross-process analysis, and business-value-added derived characteristics. The bus matrix shows dimension-fact relationships, whereas the DDL scripts create the structure. From normalized sources, all dimensions were carefully derived using transformation techniques that kept data integrity and optimized for analytical queries.

The dimension tables include customer segmentation, product stock status, and geographic regions that were not standardized. Referential integrity with dimensions is maintained in fact tables while storing relevant analysis measures. This entire transformation from operational to analytical data model provides a solid platform for business intelligence reporting and analysis of FlipKart e-commerce data.