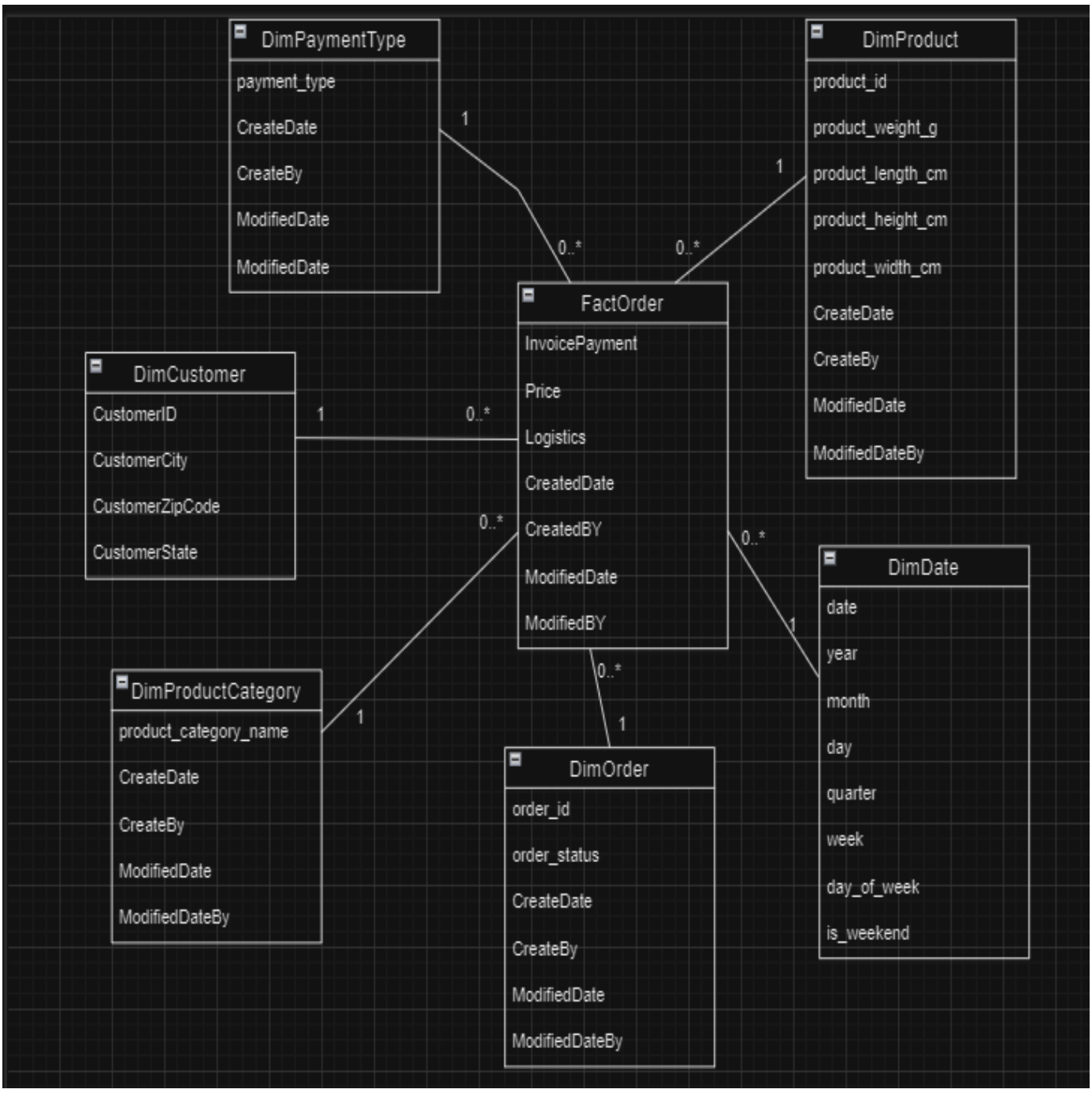
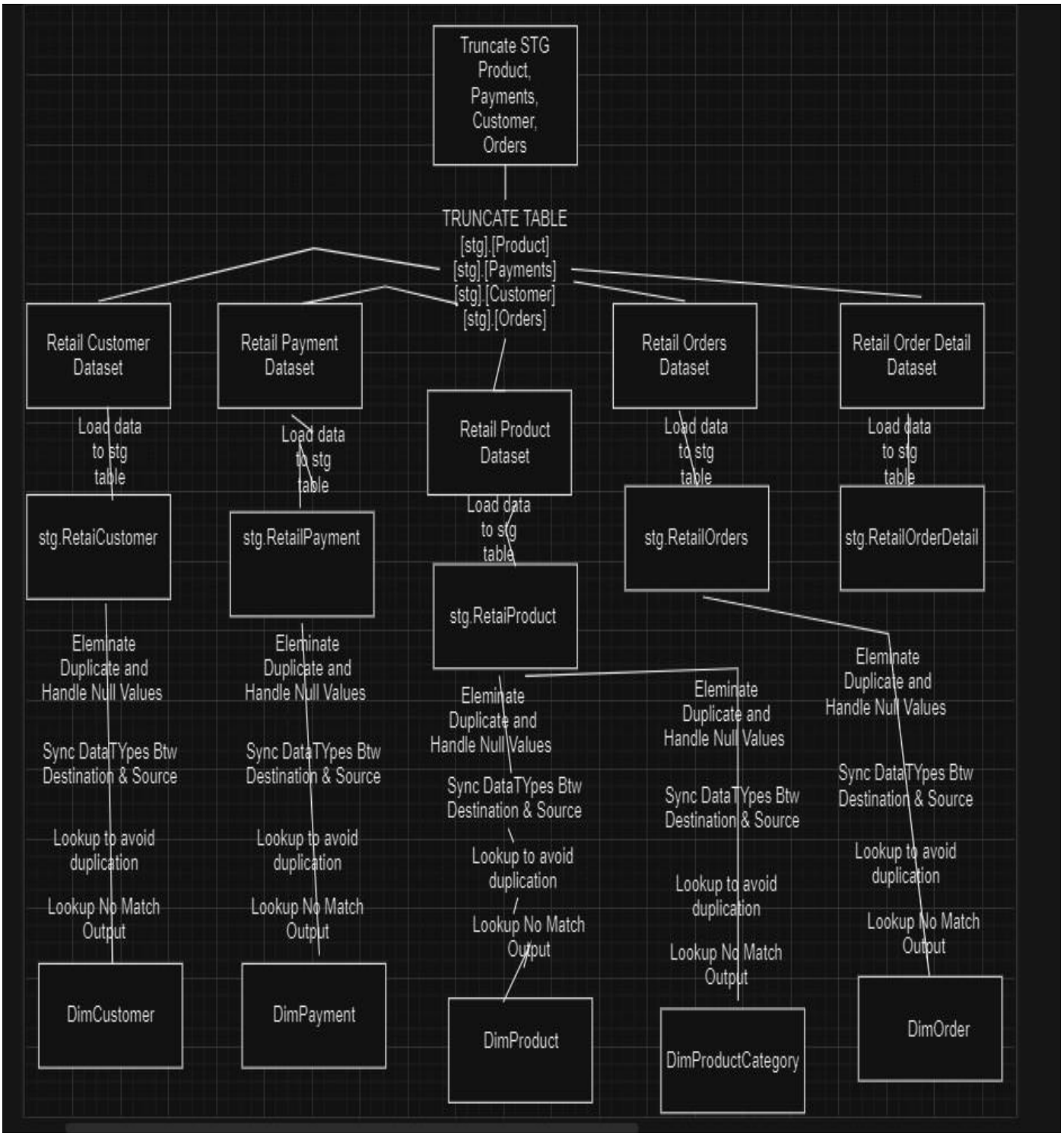
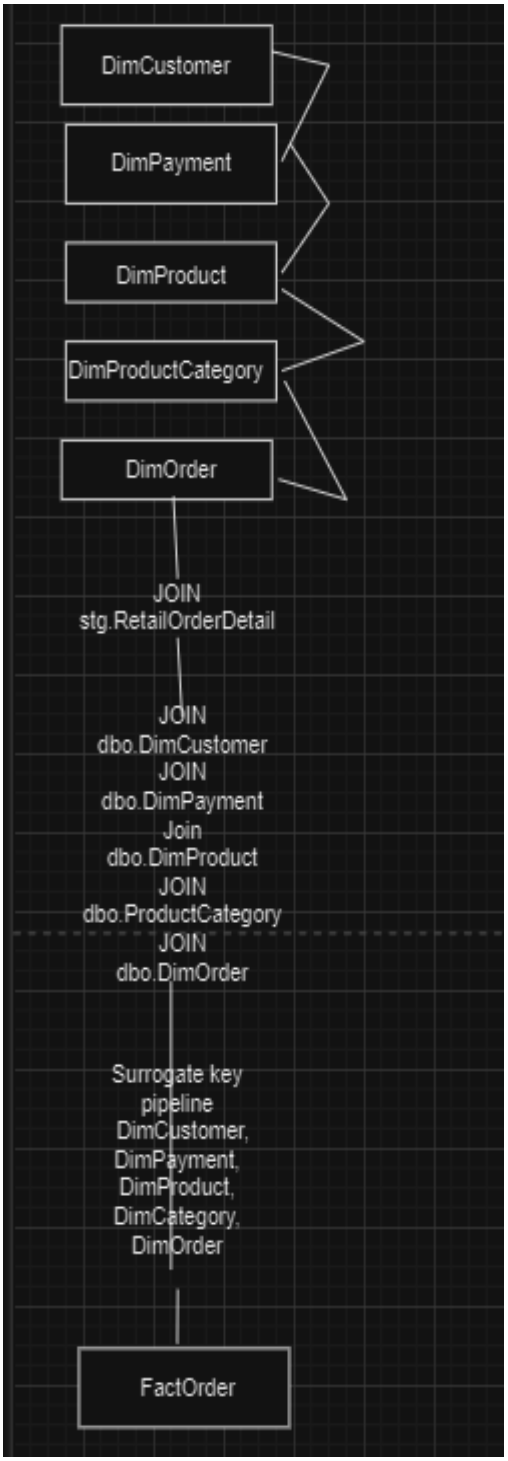
1.1 DIMENSIONAL SCHEMA – DIAGRAM



1.2 ETL MAP


Truncate STG Product, Payments, Customer, Orders
TRUNCATE TABLE [stg].[Product] [stg].[Payments] [stg].[Customer] [stg].[Orders]
Retail Customer Dataset
Retail Payment Dataset
Retail Product Dataset
Retail Orders Dataset
Retail Order Detail Dataset
Load data to stg table
Load data to stg table
Load data to stg table
Load data to stg table
Load data to stg table
stg.RetaiCustomer
stg.RetailPayment
stg.RetailProduct
stg.RetailOrders
stg.RetailOrderDetail
Eleminate Duplicate and Handle Null Values
Eleminate Duplicate and Handle Null Values
Eleminate Duplicate and Handle Null Values
Eleminate Duplicate and Handle Null Values
Eleminate Duplicate and Handle Null Values
Sync DataTYpes Btw Destination & Source
Sync DataTYpes Btw Destination & Source
Sync DataTYpes Btw Destination & Source
Sync DataTYpes Btw Destination & Source
Sync DataTYpes Btw Destination & Source
Lookup to avoid duplication
Lookup to avoid duplication
Lookup to avoid duplication
Lookup to avoid duplication
Lookup to avoid duplication
Lookup No Match Output
Lookup No Match Output
Lookup No Match Output
Lookup No Match Output
Lookup No Match Output
DimCustomer
DimPayment
DimProduct
DimProductCategory
DimOrder

Burak Kaya Report-2



2.1.1 TARGET DATABASE CREATION SCRIPT

```
-- Orders
CREATE TABLE stg.Orders (
    order_id                  VARCHAR(512),
    customer_id               VARCHAR(512),
    order_status              VARCHAR(512),
    order_purchase_timestamp  VARCHAR(50),
    order_approved_at         VARCHAR(50),
    order_delivered_timestamp VARCHAR(50),
    order_estimated_delivery_date VARCHAR(50)
);

-- OrderDetail
CREATE TABLE stg.OrderDetail (
    order_id          VARCHAR(512),
    order_item_id     VARCHAR(512),
    product_id        VARCHAR(512),
    seller_id         VARCHAR(512),
    price             DECIMAL(18,2),
    shipping_charges  DECIMAL(18,2)
);

-- Customers
CREATE TABLE stg.Customer (
    customer_id               VARCHAR(512),
    customer_zip_code_prefix  INT,
    customer_city             VARCHAR(512),
    customer_state            VARCHAR(512)
);

-- Payments
CREATE TABLE stg.Payments (
    order_id             VARCHAR(512),
    payment_sequential   VARCHAR(512),
    payment_type         VARCHAR(512),
    payment_installments VARCHAR(512),
    payment_value        DECIMAL(18,2)
);

-- Products
```

Burak Kaya Report-2

```
CREATE TABLE stg.Product (
   product_id          VARCHAR(512),
   product_category_name VARCHAR(512),
   product_weight_g      VARCHAR(50),
   product_length_cm     VARCHAR(50),
   product_height_cm     VARCHAR(50),
   product_width_cm      VARCHAR(50)
);
CREATE TABLE dbo.DimDate (
   DateKey INT PRIMARY KEY,
   date DATE,
   year INT,
   month INT,
   day INT,
   quarter INT,
   week INT,
   day_of_week INT,
   is_weekend INT
);
--Fact Table


CREATE TABLE [dbo].[FactOrder](
        [FactOrderKey] [int] IDENTITY(1,1) NOT NULL,
        [Customerkey] [int] NOT NULL,
        [Orderkey] [int] NOT NULL,
        [ProductKey] [int] NOT NULL,
        [ProductCategorykey] [int] NOT NULL,
        [PaymentTypeKey] [int] NOT NULL,
        [PurchaseDateKey] [int] NOT NULL,
        [DeliveredDateKey] [int] NOT NULL,
        [InvoicePayment] [decimal](18, 2) NULL,
        [Price] [decimal](18, 2) NULL,
        [Logistics] [decimal](18, 2) NULL,
        [CreatedDate] [datetime] DEFAULT GETUTCDATE() NOT NULL,
        [CreatedBY] [nvarchar](4000) DEFAULT ORIGINAL_LOGIN() NOT NULL,
        [ModifiedDate] [datetime] DEFAULT GETUTCDATE() NOT NULL,
        [ModifiedBY] [nvarchar](4000) DEFAULT ORIGINAL_LOGIN() NOT NULL,
 CONSTRAINT [PK_FactOrder] PRIMARY KEY CLUSTERED
(
        [FactOrderKey] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]) ON [PRIMARY]  GO
```

Burak Kaya Report-2

```
------------------------------PK Constraints
--Customer
IF NOT EXISTS (
   SELECT 1
   FROM sys.key_constraints
   WHERE [parent_object_id] = OBJECT_ID('DimCustomer')
   AND [type] = 'PK'
)
BEGIN
            ALTER TABLE [dbo].[DimCustomer]
            ADD CONSTRAINT pk_DimCustomer PRIMARY KEY ([Customerkey]);
END;

--DimOrder

IF NOT EXISTS (
   SELECT 1
   FROM sys.key_constraints
   WHERE [parent_object_id] = OBJECT_ID('DimOrder')
   AND [type] = 'PK'
)
BEGIN

            ALTER TABLE [dbo].[DimOrder]
            ADD CONSTRAINT pk_DimOrder PRIMARY KEY ([Orderkey]);
END;



--DimPaymentType

IF NOT EXISTS (
   SELECT 1
   FROM sys.key_constraints
   WHERE [parent_object_id] = OBJECT_ID('DimPaymentType')
   AND [type] = 'PK'
)
BEGIN
            ALTER TABLE [dbo].[DimPaymentType]
            ADD CONSTRAINT pk_DimPaymentType PRIMARY KEY ([PaymentTypekey]);
END;
```

Burak Kaya Report-2

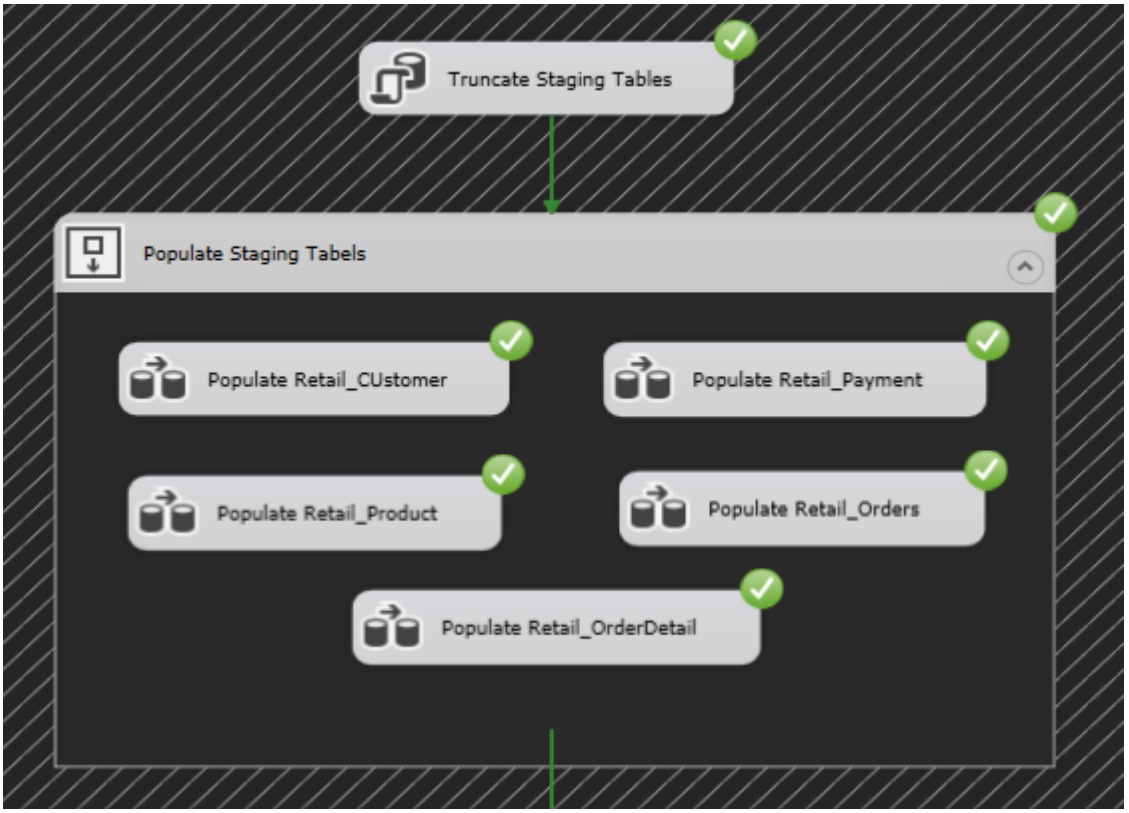
```
--DimProduct
IF NOT EXISTS (
   SELECT 1
   FROM sys.key_constraints
   WHERE [parent_object_id] = OBJECT_ID('DimProduct')
   AND [type] = 'PK'
)
BEGIN
        ALTER TABLE [dbo].[DimProduct]
        ADD CONSTRAINT pk_DimProduct PRIMARY KEY ([Productkey]);

END;

--DimProductcategory
IF NOT EXISTS (
   SELECT 1
   FROM sys.key_constraints
   WHERE [parent_object_id] = OBJECT_ID('DimProductcategory')
   AND [type] = 'PK'
)
BEGIN
        ALTER TABLE [dbo].[DimProductcategory]
        ADD CONSTRAINT pk_DimProductcategory PRIMARY KEY ([Productcategorykey]);

END;


--DimDate
IF NOT EXISTS (
   SELECT 1
   FROM sys.key_constraints
   WHERE [parent_object_id] = OBJECT_ID('DimDate')
   AND [type] = 'PK'
)
BEGIN
        ALTER TABLE [dbo].[DimDate]
        ADD CONSTRAINT pk_DimDate PRIMARY KEY ([DateKey]);

END;
```

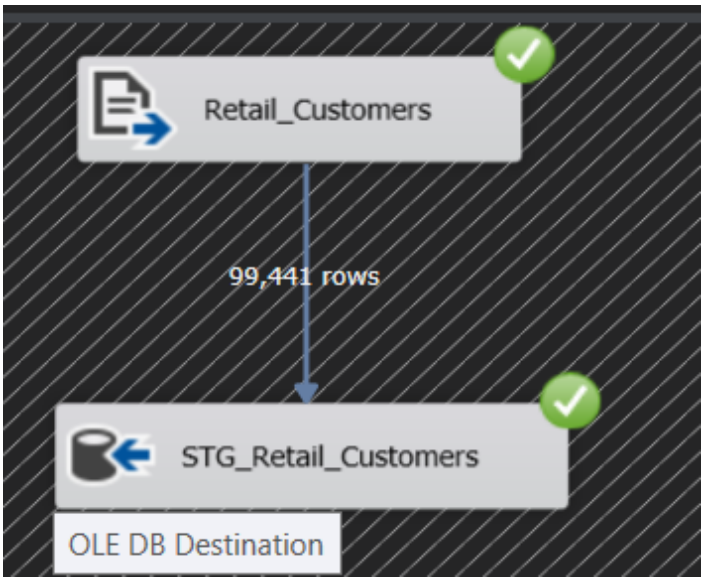
Burak Kaya Report-2

```
--DimDate
IF NOT EXISTS (
   SELECT 1
   FROM sys.key_constraints
   WHERE [parent_object_id] = OBJECT_ID('DimDate')
   AND [type] = 'PK'
)
BEGIN
        ALTER TABLE [dbo].[DimLocation]
        ADD CONSTRAINT pk_DimDate PRIMARY KEY ([DateKey]);

END;

--FK Constraint
ALTER TABLE [dbo].[FactOrder]
ADD CONSTRAINT fk_DimOrder
FOREIGN KEY ([Orderkey]) REFERENCES [dbo].[DimOrder]([Orderkey]);
ALTER TABLE [dbo].[FactOrder]
ADD CONSTRAINT fk_DimCustomer
FOREIGN KEY ([Customerkey]) REFERENCES [dbo].[DimCustomer]([Customerkey]);
ALTER TABLE [dbo].[FactOrder]
ADD CONSTRAINT fk_DimProduct
FOREIGN KEY ([ProductKey]) REFERENCES [dbo].[DimProduct]([ProductKey]);
ALTER TABLE [dbo].[FactOrder]
ADD CONSTRAINT fk_DimProductcategory
FOREIGN KEY ([ProductCategorykey]) REFERENCES [dbo].[DimProductcategory]([ProductCategorykey]);
ALTER TABLE [dbo].[FactOrder]
ADD CONSTRAINT fk_DimPaymentType
FOREIGN KEY ([PaymentTypeKey]) REFERENCES [dbo].[DimPaymentType] ([PaymentTypeKey]);
ALTER TABLE [dbo].[FactOrder]
ADD CONSTRAINT fk_DimDate_PurchaseDateKey
FOREIGN KEY ([PurchaseDateKey]) REFERENCES [dbo].[DimDate]([DateKey]);
ALTER TABLE [dbo].[FactOrder]
ADD CONSTRAINT fk_DimDate_DeliveredDateKey
FOREIGN KEY ([DeliveredDateKey]) REFERENCES [dbo].[DimDate]([DateKey]);
```

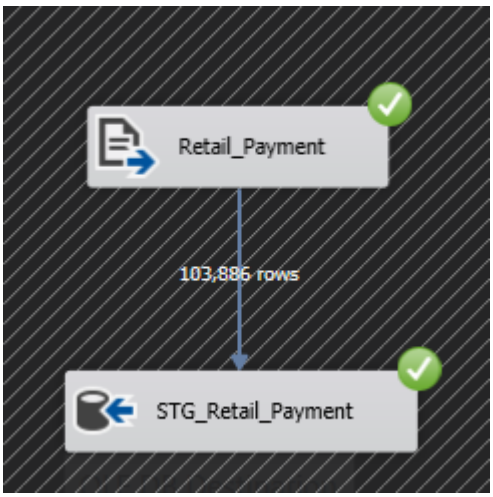
2.1.1 ETL PROCESSES

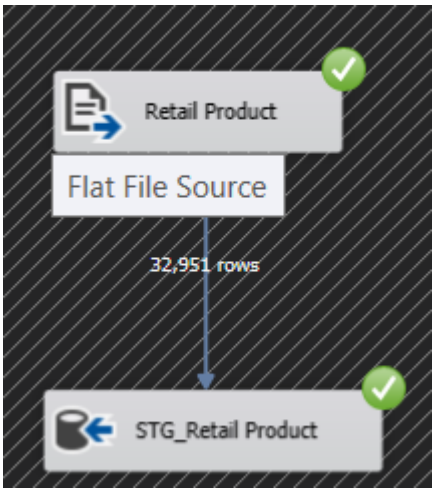
The model shows a clear process for filling a data warehouse. First, it empties the staging tables to remove old data. Then, it extracts data from different sources and loads it into these staging tables. For example, the process loads 99,441 customer rows, 32,951 product rows, 99,441 order rows, 112,650 order detail rows, and 103,886 payment rows into their respective staging tables. After loading, it transforms this data to fit the warehouse's structure and moves it into dimension tables like Customer, Product, Order, Payment, and ProductCategory. Finally, it combines data from these dimension tables to fill the main fact table, ensuring everything is correct and ready for analysis. This approach ensures the data warehouse is filled with accurate and well-organized data, ready for reporting and analysis.

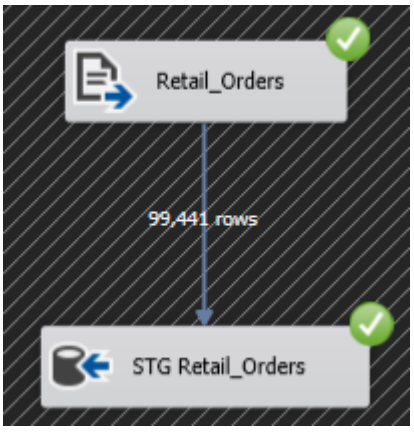


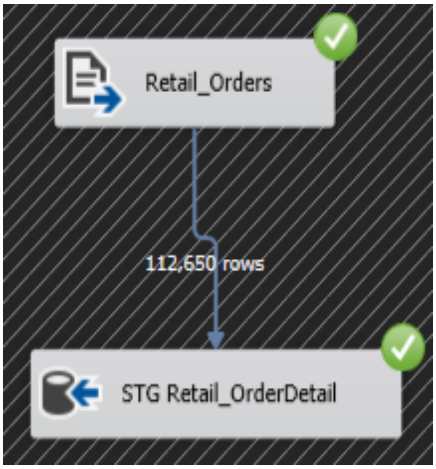


Burak Kaya Report-2



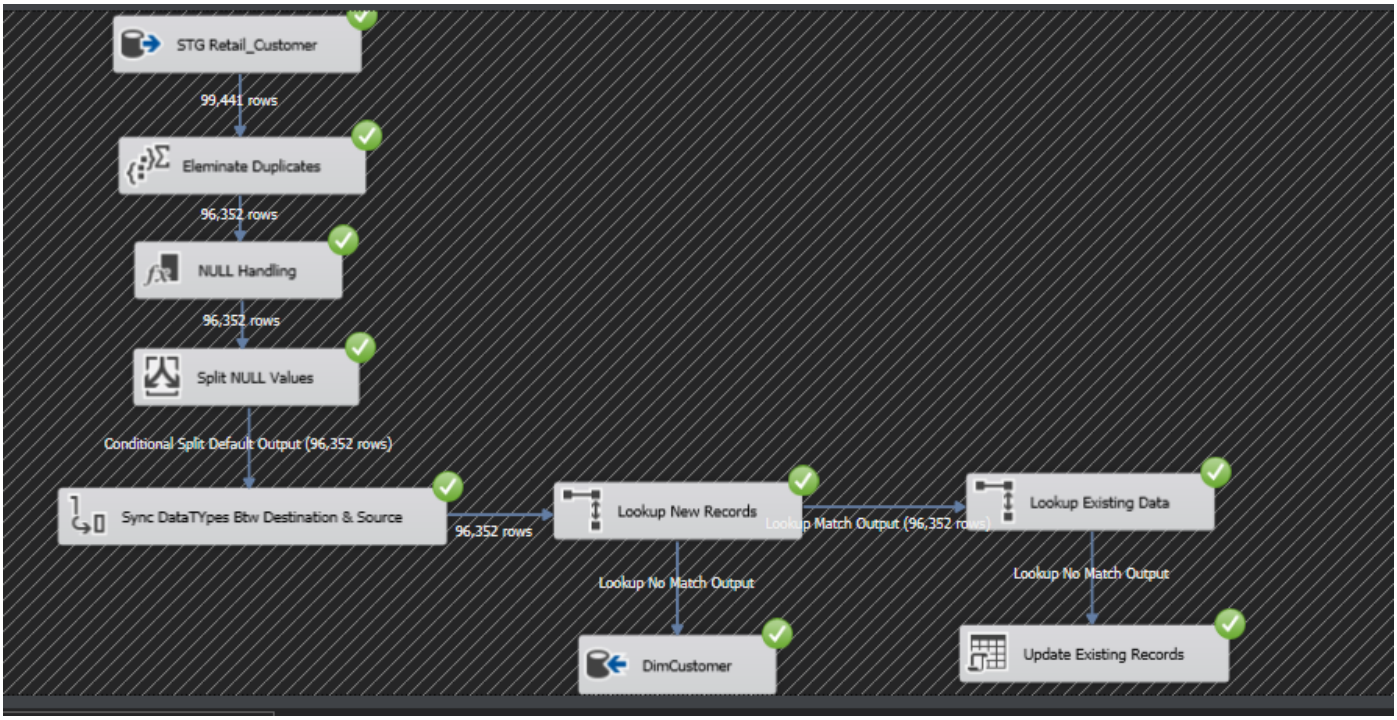






2.1.1.1 DIMENSION [X]

DimCustomer



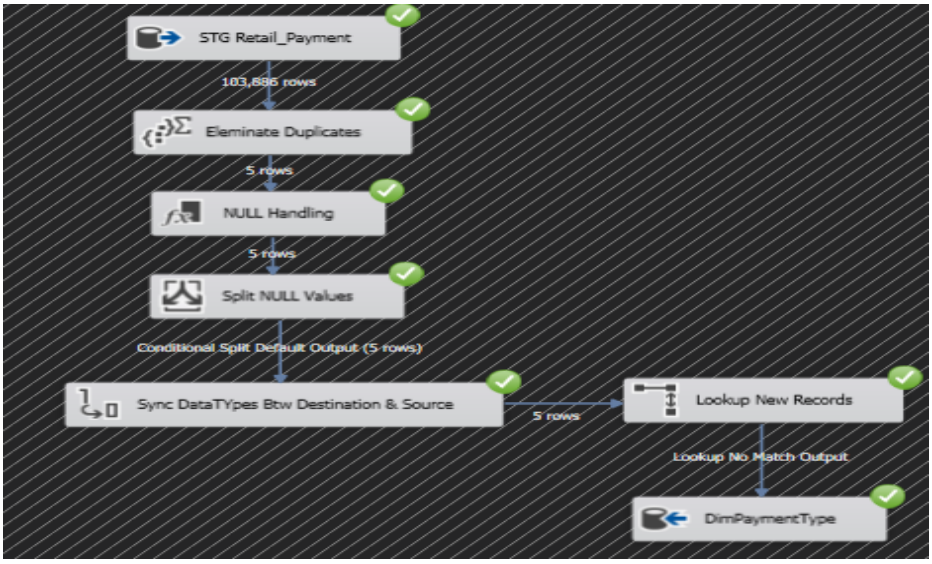
Burak Kaya Report-2

It shows the process of cleaning and updating customer data before adding it to the data warehouse. It starts with 99,441 rows of customer data in the STG_Retail_Customer table. The process removes duplicates, reducing the number to 96,352 rows. Next, it handles and splits any NULL values. After these steps, the data is checked to ensure it matches the correct data types between the source and destination. The cleaned data is then split into two paths: new records and existing records. New records are added to the DimCustomer table, while existing records are updated to keep everything current. This ensures the customer data is accurate and up-to-date in the data warehouse.

DimProduct

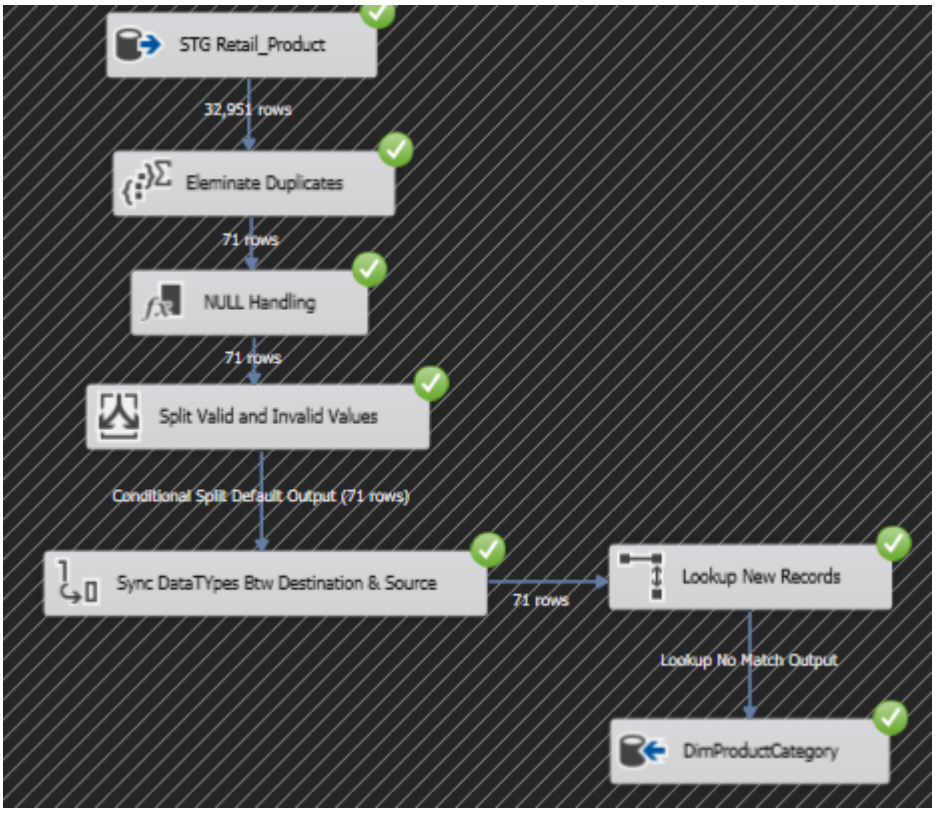
It shows the process of preparing product data for the data warehouse. It begins with 32,951 rows of product data in the STG_Retail_Product table. The process first removes duplicates, maintaining the same number of rows. Next, it handles NULL values and splits the data into valid and invalid entries. After these steps, the data is synced to ensure that the data types match between the source and destination. The cleaned data is then split into two paths: new records and existing records. New records are added to the DimProduct table, while existing records are updated to keep everything current. This ensures that the product data is accurate and up-to-date in the data warehouse.

DimPayment



It shows the process of preparing order data for the data warehouse. It starts with 99,441 rows of order data in the STG_Retail_Order table. The process first removes duplicates, keeping the same number of rows. Then, it handles NULL values and splits the data into valid and invalid entries. After these steps, the data is synced to ensure that the data types match between the source and destination. The cleaned data is then split into two paths: new records and existing records. New records are added to the DimOrder table, while existing records are updated to keep everything current. This ensures that the order data is accurate and up-to-date in the data warehouse.

DimProductCategory



It shows the process of preparing product category data for the data warehouse. It begins with 32,951 rows of product data in the STG_Retail_Product table. After removing duplicates, the rows are reduced to 71. The process then handles NULL values and splits the data into valid and invalid entries, maintaining 71 rows. Next, it ensures that the data types

DimDate

```
DECLARE @startDate DATE = '2015-01-01';
DECLARE @endDate DATE = GETDATE();
DECLARE @currentDate DATE = @startDate;

WHILE @currentDate <= @endDate
BEGIN
   INSERT INTO dimDate (DateKey, date, year, month, day, quarter, week, day_of_week, is_weekend)
   VALUES (
      CONVERT(INT, FORMAT(@currentDate, 'yyyyMMdd')),
      @currentDate,
      YEAR(@currentDate),
      MONTH(@currentDate),
```

```
    DAY(@currentDate),
    DATEPART(QUARTER, @currentDate),
    DATEPART(WEEK, @currentDate),
    DATEPART(WEEKDAY, @currentDate),
    CASE WHEN DATEPART(WEEKDAY, @currentDate) IN (1, 7) THEN 1 ELSE 0 END
  );
  SET @currentDate = DATEADD(DAY, 1, @currentDate);
END
```

It process of populating a dimDate table with date-related data for a range of dates. The process starts by setting the start date to '2015-01-01' and the end date to the current date. It initializes a variable @currentDate to the start date. A WHILE loop runs as long as @currentDate is less than or equal to the end date. Within the loop, the script inserts a new row into the dimDate ta ble. Each row includes various date components such as DateKey, date, year, month, day, quarter, week, day_of_week, and a flag is_weekend to indicate if the date falls on a weekend. The DateKey is formatted as an integer in 'yyyyMMdd' format. After inserting the row, the script increments @currentDate by one day and continues the loop until all dates in the range are processed. This ensures that the dimDate table is fully populated with a comprehensive set of date-related information for analysis.

2.1.1.2 FACT [X]

It is a stored procedure that prepares and inserts order data into the FactOrder table in a data warehouse. It starts by crea ting temporary tables to store intermediate results, such as customer orders, customer payments, and customer staging. The procedure joins various staging tables to gather all necessary data, including customer information, order details, payment types, and product categories. It then identifies new records that need to be inserted into the FactOrder table. The insertion is done in ba tches of 100,000 rows to handle large volumes of data efficiently. Each record includes keys for customer, order, product, payment typ e, purchase date, delivery date, and amounts for invoice, price, and logistics. This ensures the FactOrder table is populated with accurate and comprehensive order data for analysis.

```
CREATE OR ALTER PROCEDURE dbo.SpInsertFactOrders
AS
BEGIN
  IF OBJECT_ID('tempdb..#CustomerOrder', 'U') IS NOT NULL
    DROP TABLE #CustomerOrder;

  SELECT DISTINCT O.*
  INTO #CustomerOrder
  FROM [stg].[Customer] C
  JOIN [stg].[Orders] O ON C.Customer_id = O.customer_id;

  IF OBJECT_ID('tempdb..#CustomerPayment', 'U') IS NOT NULL
```

Burak Kaya Report-2

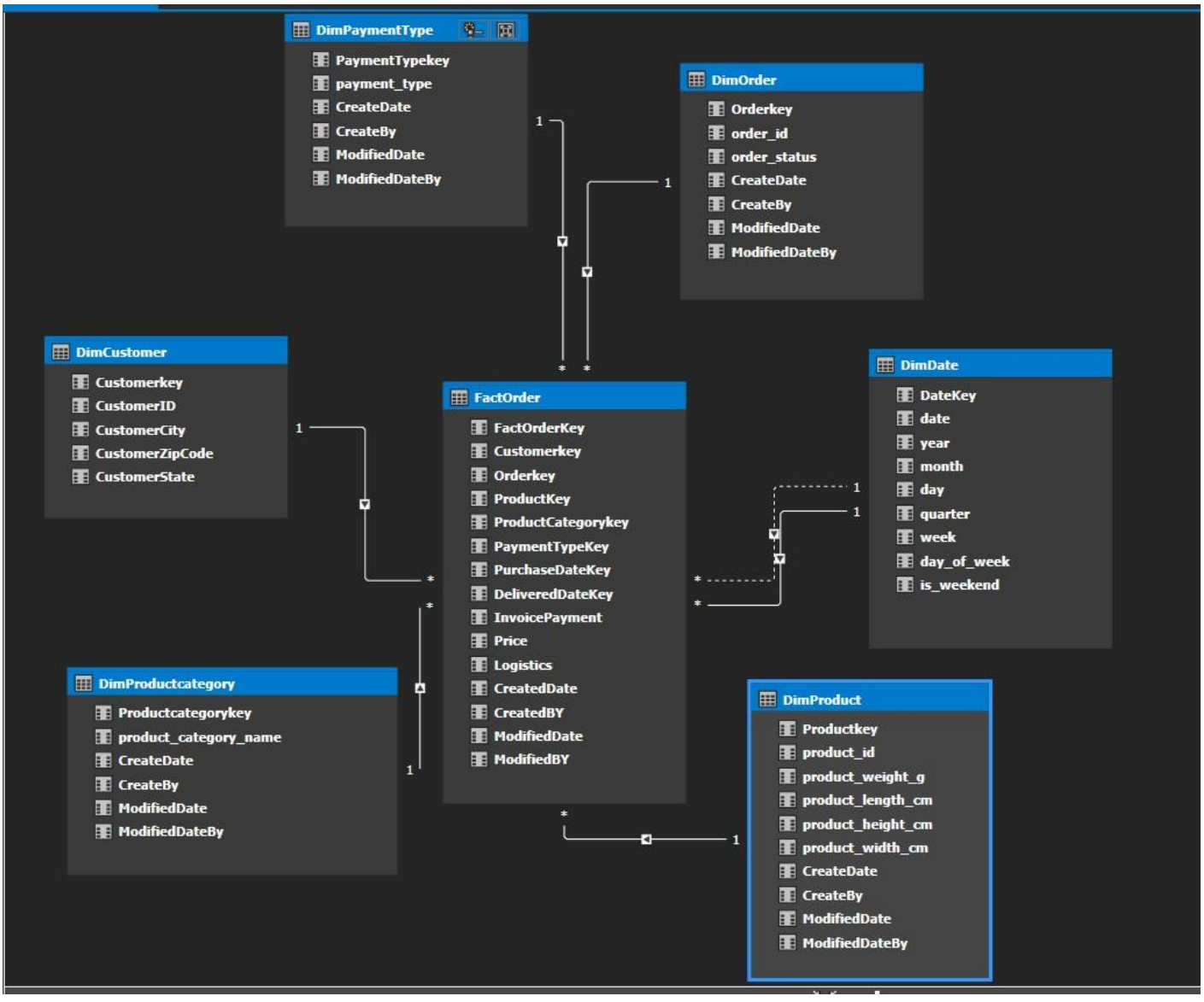
```
    DROP TABLE #CustomerPayment;


  SELECT DISTINCT C.*, P.payment_type, P.payment_value
  INTO #CustomerPayment
  FROM #CustomerOrder C
  JOIN [stg].[payments] P ON C.Order_id = P.Order_id;


  IF OBJECT_ID('tempdb..#CustomerStaging', 'U') IS NOT NULL
    DROP TABLE #CustomerStaging;


  SELECT DISTINCT
      P.*,
      PR.[product_category_name],
      OD.Product_id,
      OD.price,
      OD.shipping_charges
  INTO #CustomerStaging
  FROM #CustomerPayment P
  JOIN [stg].[orderdetail] OD ON OD.Order_id = P.Order_id
  JOIN [stg].[product] PR ON OD.product_id = PR.product_id;


  -- Preparing data to insert in FactOrder
  IF OBJECT_ID('tempdb..#InsertCustomers', 'U') IS NOT NULL
    DROP TABLE #InsertCustomers;


  SELECT DISTINCT
      NEWID() UNID,
      DC.Customerkey,
      O.Orderkey,
      DP.ProductKey,
      PC.[ProductCategorykey],
```

```
        PT.[PaymentTypeKey],
        PD.Datekey AS PurchaseDateKey,
        DD.Datekey AS DeliveredDateKey,
        C.payment_value AS InvoicePayment,
        C.Price,
        c.shipping_charges AS Logistics,
        GETUTCDATE() AS CreatedDate,
        ORIGINAL_LOGIN() AS CreatedBY,
        GETUTCDATE() AS ModifiedDate,
        ORIGINAL_LOGIN() AS ModifiedBY
    INTO #InsertCustomers
    FROM #CustomerStaging C
    JOIN dbo.DimCustomer DC ON DC.customerid = C.Customer_id
    JOIN dbo.Dimproduct DP ON DP.[product_id] = C.[product_id]
    JOIN [dbo].[DimPaymentType] PT ON PT.[payment_type] = C.[payment_type]
    JOIN [dbo].[Dimdate] PD ON CONVERT(DATE, PD.[date]) = CONVERT(DATE, C.order_purchase_timestamp)
    JOIN [dbo].[Dimdate] DD ON CONVERT(DATE, DD.[date]) = CONVERT(DATE, C.order_delivered_timestamp)
    JOIN [dbo].[Dimorder] O ON O.order_id = C.order_id
    JOIN [dbo].[DimProductcategory] PC ON PC.[product_category_name] = C.[product_category_name];

    -- Getting all those records that are new to insert
    IF OBJECT_ID('tempdb..#InsertFactOrder', 'U') IS NOT NULL
        DROP TABLE #InsertFactOrder;

    SELECT DISTINCT C.*
    INTO #InsertFactOrder
    FROM #InsertCustomers C
    LEFT JOIN dbo.factorder F ON C.[Customerkey] = ISNULL(F.[Customerkey], '')
                    AND C.[Orderkey] = ISNULL(F.[Orderkey], '')
                    AND C.[ProductKey] = ISNULL(F.[ProductKey], '')
                    AND C.[ProductCategorykey] = ISNULL(F.[ProductCategorykey], '')
```

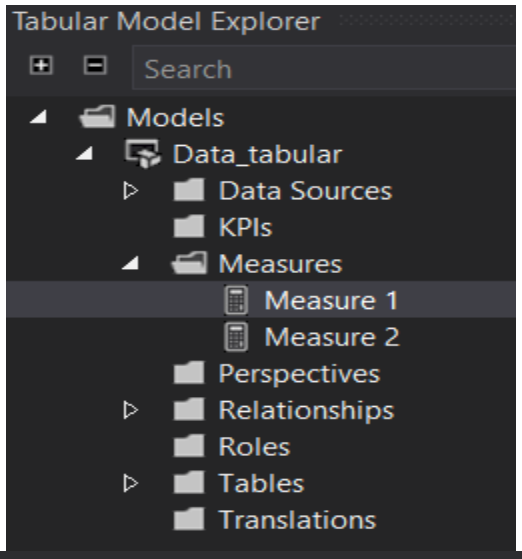
```
                AND C.[PaymentTypeKey] = ISNULL(F.[PaymentTypeKey], '')
                AND C.[PurchaseDateKey] = ISNULL(F.[PurchaseDateKey], '')
                AND C.[DeliveredDateKey] = ISNULL(F.[DeliveredDateKey], '')
  WHERE F.Customerkey IS NULL;

  -- INSERT in batch of 100000 FactOrder
  DECLARE @i INT = 0;
  DECLARE @batchSize INT = 100000;
  DECLARE @totalCount INT;

  SELECT @totalCount = COUNT(1) FROM #InsertFactOrder;

  WHILE (@i < @totalCount)
  BEGIN
    PRINT 'Batch Starts';

    INSERT INTO [dbo].[FactOrder] (
      [Customerkey], [Orderkey], [ProductKey],
      [ProductCategorykey], [PaymentTypeKey],
      [PurchaseDateKey], [DeliveredDateKey], [InvoicePayment],
      [Price], [Logistics]
    )
    SELECT
      ISNULL([Customerkey], 0) AS [Customerkey],
      ISNULL([Orderkey], 0) AS [Orderkey],
      ISNULL([ProductKey], 0) AS [ProductKey],
      ISNULL([ProductCategorykey], 0) AS [ProductCategorykey],
      ISNULL([PaymentTypeKey], 0) AS [PaymentTypeKey],
      ISNULL([PurchaseDateKey], 9999999) AS [PurchaseDateKey],
      ISNULL([DeliveredDateKey], 9999999) AS [DeliveredDateKey],
      ISNULL([InvoicePayment], 0) AS [InvoicePayment],
```

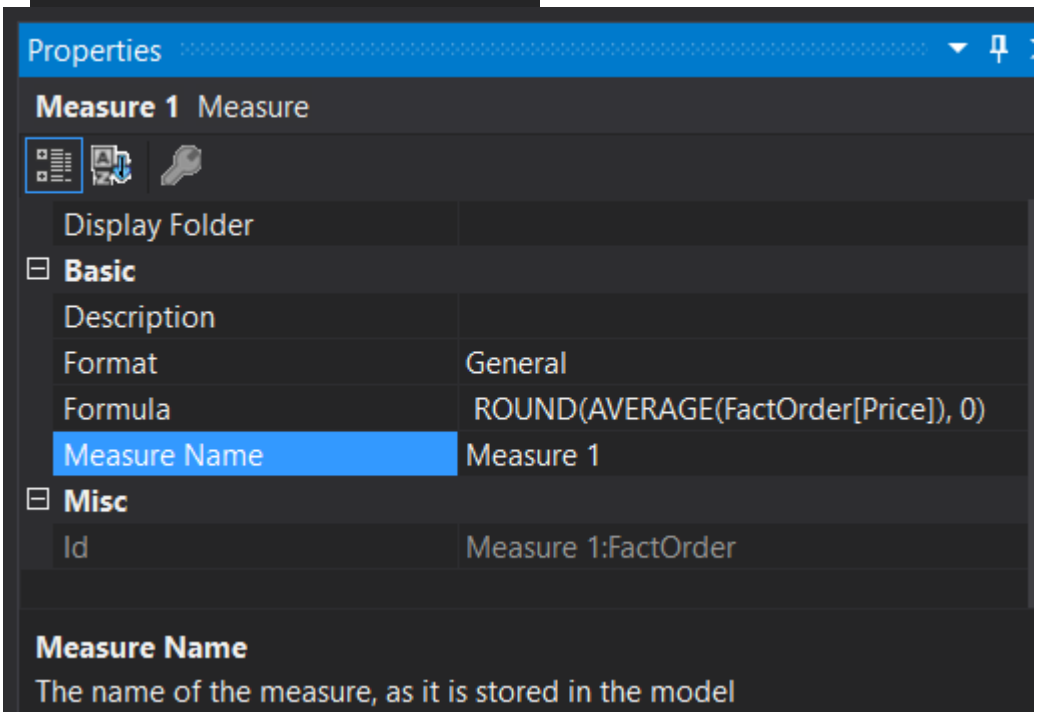
```
        ISNULL([Price], 0) AS [Price],
        ISNULL([Logistics], 0) AS [Logistics]
    FROM #InsertFactOrder
    ORDER BY UNID
    OFFSET @i ROWS
    FETCH NEXT @batchSize ROWS ONLY;

    SET @i = @i + @batchSize;
  END
END;
```

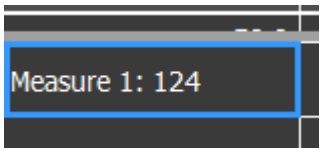
2.2.1 FINAL CUBE STRUCTURE



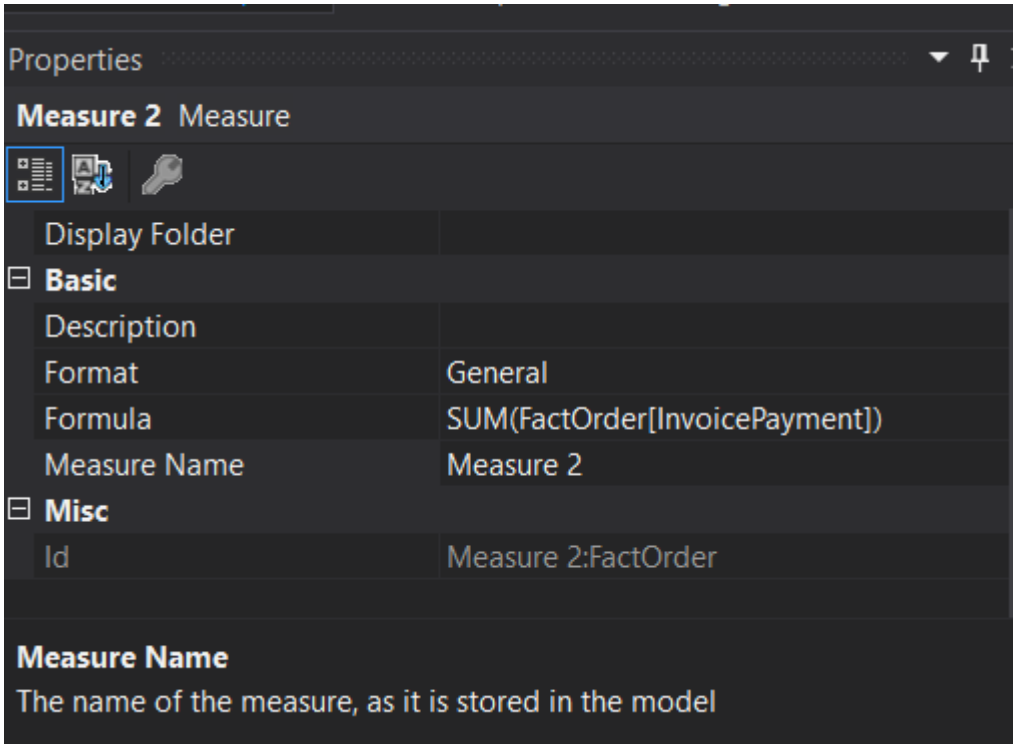
2.2.2 MEASURES

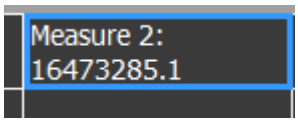






Burak Kaya Report-2





Measure 1

The measure Measure 1 calculates the average price from the FactOrder table. The DAX formula used is ROUND(AVERAGE(FactOrder[Price]), 0), which computes the average value of the Price column and then rounds it to the nearest whole number. The result shown for Measure 1 is 124, indicating that the average price of all orders, rounded to the nearest whole number, is 124.

Measure 2

The measure Measure 2 calculates the total sum of invoice payments from the FactOrder table. The DAX formula used is SUM(FactOrder[InvoicePayment]), which sums up all values in the InvoicePayment column. The result shown for Measure 2 is 16473285.1, indicating that the total sum of all invoice payments in the FactOrder table is 16,473,285.1. This value represents the cumulative amount of money from all invoices in the dataset.

2.2.3 DIMENSIONS

DimCustomer

[Customerkey] f_x

	Custom...	CustomerID	CustomerCity	CustomerZipCode	CustomerState	Add Column
1	1	7831e0249ea...	sao paulo	5267	SP	
2	5	c7f4afeac875...	sao paulo	5422	SP	
3	6	7a43cf5e239...	sao paulo	4726	SP	
4	13	80444e31ba7...	sao paulo	5348	SP	
5	15	33ebf671045...	sao paulo	4773	SP	

DimDate

[DateKey]

	Dat...	date	year	month	day	quarter	week	day_of_week	is_weekend	Add Column
1	20150101	1/1/2...	2015	1	1	1	1	5	0	
2	20150108	1/8/2...	2015	1	8	1	2	5	0	
3	20150115	1/15/...	2015	1	15	1	3	5	0	
4	20150122	1/22/...	2015	1	22	1	4	5	0	
5	20150129	1/29/...	2015	1	29	1	5	5	0	

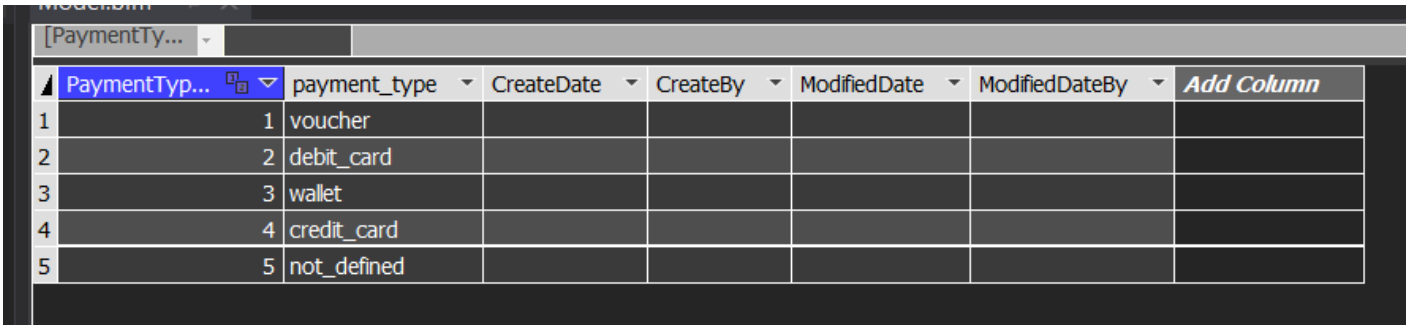
DimOrder

Model.bim*

[Orderkey]

	Ord...	order_id	order_status	CreateDate	CreateBy	ModifiedDate	ModifiedDateBy	Add Column
1	1	9ac146c5...	delivered					
2	2	f4dd7fe0a...	delivered					
3	3	47d62434...	delivered					
4	4	beebcdb8...	delivered					
5	5	e7d970bf...	delivered					
6	6	d1ce1812...	delivered					
7	7	7aec0be6...	delivered					

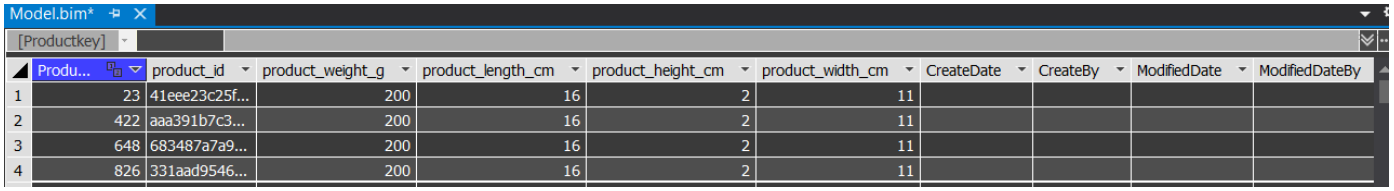
DimPaymentType



[PaymentTy...

	PaymentTyp...	payment_type	CreateDate	CreateBy	ModifiedDate	ModifiedDateBy	Add Column
1	1	voucher					
2	2	debit_card					
3	3	wallet					
4	4	credit_card					
5	5	not_defined					

DimProduct

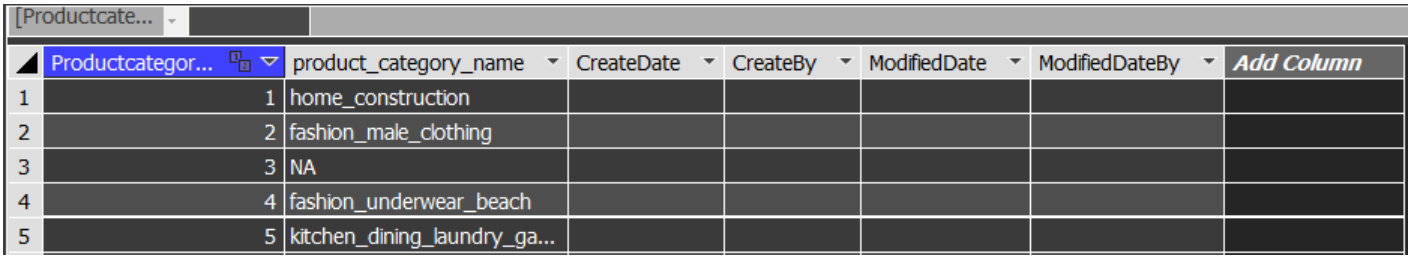


Model.bim*

[Productkey]

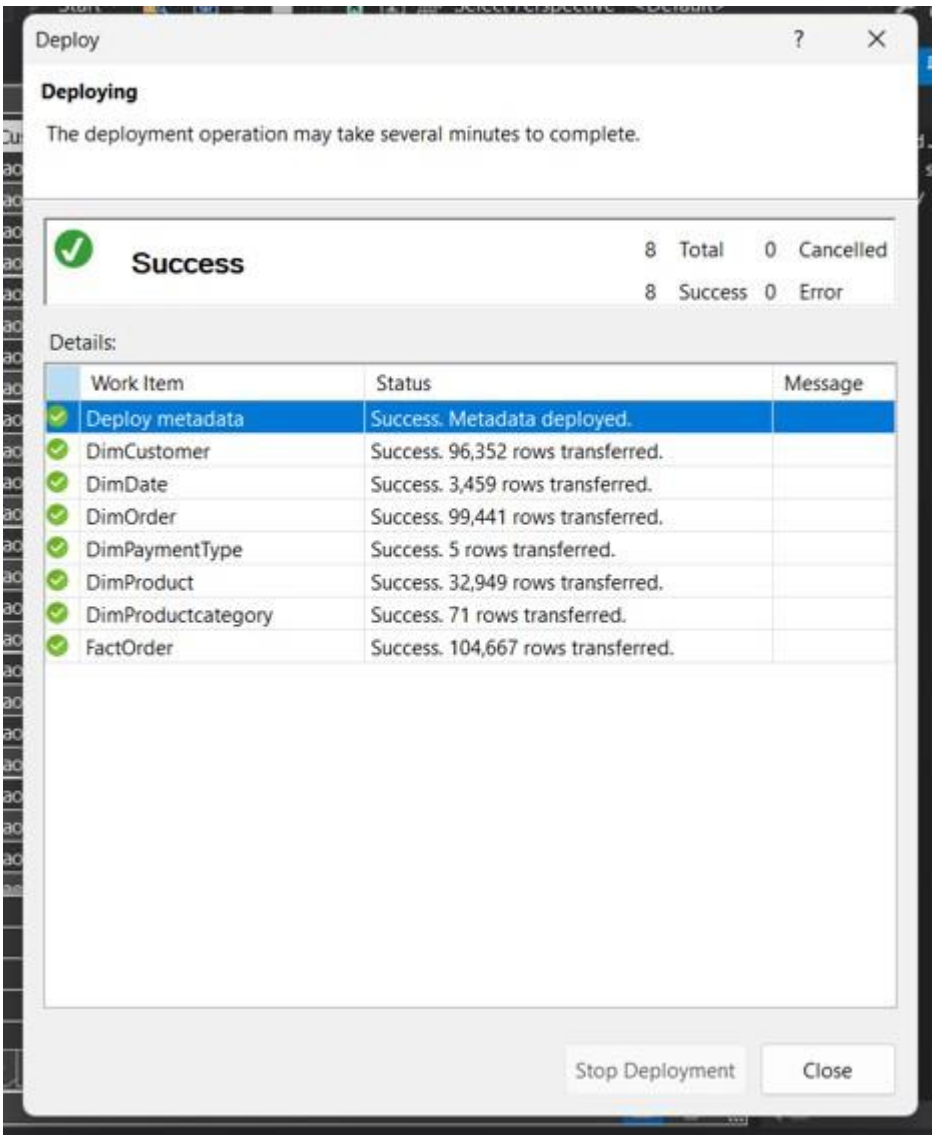
	Produ...	product_id	product_weight_g	product_length_cm	product_height_cm	product_width_cm	CreateDate	CreateBy	ModifiedDate	ModifiedDateBy
1	23	41eee23c25f...	200	16	2	11				
2	422	aaa391b7c3...	200	16	2	11				
3	648	683487a7a9...	200	16	2	11				
4	826	331aad9546...	200	16	2	11				

DimProductCategory



[Productcate...

	Productcategor...	product_category_name	CreateDate	CreateBy	ModifiedDate	ModifiedDateBy	Add Column
1	1	home_construction					
2	2	fashion_male_clothing					
3	3	NA					
4	4	fashion_underwear_beach					
5	5	kitchen_dining_laundry_ga...					



GENERAL CONCLUSIONS:

This phase focused on designing and implementing the core components of the data warehouse, including dimensional modeling, the ETL process, and a tabular model in SSAS. I created a star schema to efficiently handle OLAP queries, covering critical e-commerce data like customer orders, products, payments, and shipping.

Using SQL Server Integration Services (SSIS), I developed an ETL process to manage data extraction, transformation, and loading, incorporating data quality checks and incremental loading to maintain data accuracy and enable real-time updates. The SSAS tabular model organized data for quick aggregation and advanced analysis, supporting multidimensional insights across various business areas.

This work has established a flexible data warehouse infrastructure that enables scalable, real-time analytics, supporting data-driven decisions in the competitive e-commerce environment.