**Extract, Transform, Load (ETL) Process Documentation**

**1. Introduction:**

The ETL (Extract, Transform, Load) procedure used for the FlipKart Data Warehouse project is thoroughly described in this document. An essential link between our analytical data warehouse (FlipKart\_EDW) and our normalized operational database (DB\_FlipKart) is the ETL procedure. By doing this, we were able to convert normalized data into a dimensional model that was best suited for analytical and business intelligence queries.

**2. ETL Architecture Overview:**

The following are the main stages of our phased ETL implementation:

* Data extraction into staging tables from the normalized database
* Converting staging data into fact and dimension tables
* Data transformation and loading into the dimensional model
* Verification of the integrity and quality of data

**ETL Tool Selected for ETL Process:- T- SQL**

I used **SQL Server Integration Services (SSIS) and T-SQL tools** to set up the ETL process. The built-in features of SSIS that let you change dimensions slowly and handle surrogate keys were especially helpful for our project. The tool easily connected to our SQL Server environment, which made handling our FlipKart datasets quick and easy. It also had strong logging features that made verification simple.

Because SQL Server T-SQL offered a stable and effective foundation for our data transformations, we used it to implement the full ETL process. A thorough description of each stage of the ETL process is described below.

**3. Phase 1- Extraction of Data:**

Data extraction from the normalized database (DB\_FlipKart) and loading it into staging tables in our data warehouse database (FlipKart\_EDW) represented the first step of our ETL process. The operational and analytical environments were clearly separated by this method.

Here is the query I used for Creating staging tables to hold the extracted data:

-- Create staging tables to hold the extracted data

CREATE TABLE stg\_Customers (

customer\_id INT,

year\_birth INT,

education VARCHAR(50),

marital\_status VARCHAR(50),

income INT,

kid\_home INT,

teen\_home INT,

dt\_customer DATE,

recency INT,

loyalty\_score INT

);

-- Additional staging tables created for other entities

We used simple INSERT lines and SELECT queries to get data from the normalized tables and put it into these staging tables:

A screenshot of a computer

AI-generated content may be incorrect.

3.1 ETL\_Staging\_tables

-- Extract Customers data

INSERT INTO FlipKart\_EDW.dbo.stg\_Customers

SELECT

customer\_id,

year\_birth,

education,

marital\_status,

income,

kid\_home,

teen\_home,

dt\_customer,

recency,

loyalty\_score

FROM DB\_FlipKart.dbo.Customers;

-- Similarly for other datasets

A screenshot of a computer

AI-generated content may be incorrect.

3.2 ETL\_Data\_Extraction

During the extraction process, I did the following basic things to clean up and handle the data:

* Used ISNULL() functions to deal with NULL numbers in income fields
* Made sure that date formats were uniform by using the right conversion tools
* Getting rid of any duplicate records to keep the data consistent

This staging method put a buffer between the source and target systems so that processing could happen separately and with less effect on the operational database.

**4. Phase 2- Dimension Table Creation and Population:**

I used the star schema design pattern to generate dimension tables after extracting data into staging tables. Every dimension table contained:

* **The primary key is a surrogate key (SK).**
* **The source system's natural or commercial key**
* **Descriptive Characteristics**
* **Slowly Changing Dimension, or SCD Type 2 characteristics for monitoring past developments**

a. Creating Date Dimension:

Since the date dimension is so important for time-based research, it was filled in first:

**Query:**

-- Populate dim\_Date with dates from 2010 to 2025

DECLARE @StartDate DATE = '2010-01-01';

DECLARE @EndDate DATE = '2025-12-31';

-- Using a recursive CTE to generate dates

WITH DateCTE AS (...)

INSERT INTO dim\_Date (...)

SELECT ...

A screenshot of a computer

AI-generated content may be incorrect.

4.1 Verify date dimension implementation with range and key attributes

A total of 5,844 records from January 1, 2010, to December 31, 2025 have been properly added to the date dimension. Time-based analytics can be done with this because it covers all possible dates for both past analysis and future forecasting.

b. Implementing SCD Type 2 for Key Dimensions:

For dimensions like Customer and Product, I implemented Slowly Changing Dimension Type 2 logic to track historical changes:

A screenshot of a computer

AI-generated content may be incorrect.

4.2 Verifying SCD Type 2 implementation in Customer dimension

The result shows that SCD Type 2 was successfully implemented for the Customer variable. All records have correct surrogate keys (CustomerSK) that are different from the natural keys (CustomerID), along with all version numbers (IsCurrent=1) and null end dates. The resulting CustomerSegment attribute correctly sorts customers into Premium or Standard groups based on their loyalty scores, which are then linked to business groups. This confirmation shows that the right tools are in place to keep track of past changes in customer attributes in the future.

A screenshot of a computer program

AI-generated content may be incorrect.

4.3 Verifying another SCD Type 2 implementation in Customer dimension

This verification shows that SCD Type 2 was properly implemented for 407 Premium customers and 593 Standard customers, all of whom had the same start date (2025-03-02). The number of current records fits the number of customers, and there are no historical records. This means that this is our first data load and we haven't been able to track any changes yet. This shows that the SCD structure is set up properly to handle changes in future customer segments.

c. Derived Attribute Creation  
  
Utilizing derived qualities to offer business value was a crucial step in the dimension construction process:  
  
i. Customer Segmentation: Based on loyalty scores, clients were divided into Premium, Standard, and Basic categories.  
ii. Product Stock Status: Items were categorized as either high, medium, low, or out of stock.  
iii.Geographic Areas: To make analysis easier, places were grouped into geographic areas.

A screenshot of a computer

AI-generated content may be incorrect.

4.4 Verify derived attributes across dimensions

The query results show that business-enhancing derived characteristics were successfully implemented across multiple dimensions. The customer segmentation shows that there are 407 Premium customers and 593 Standard customers. The product stock status shows that the inventory levels are (821 High, 104 Medium, and 75 Low), and all 999 sites are in North America.

**5. Phase 3- Fact Table Creation and Population:**

After putting data into all of the dimension tables, I made the fact tables and put data into them. This process had these parts:

* Joining staging tables with dimension tables to get surrogate keys
* Calculating how to figure out numbers and measures
* Putting data into the fact tables so that they are linked correctly

a. Sales Fact Table:

The main fact table keeps track of all sales deals with links to all dimensions:

SQL Query:

-- Populate fact\_Sales table

INSERT INTO fact\_Sales (

TransactionID, OrderID, CustomerSK, ProductSK, LocationSK,

OrderDateSK, ShipDateSK, PaymentMethodSK, ShippingMethodSK,

Quantity, Discount, Sales, TotalAmount, ShippingStatus

)

SELECT

od.transaction\_id,

od.order\_id,

c.CustomerSK,

p.ProductSK,

l.LocationSK,

CONVERT(INT, FORMAT(o.order\_date, 'yyyyMMdd')) AS OrderDateSK,

CONVERT(INT, FORMAT(od.ship\_date, 'yyyyMMdd')) AS ShipDateSK,

pm.PaymentMethodSK,

sm.ShippingMethodSK,

od.quantity,

od.discount,

od.sales,

od.total\_amount,

od.shipping\_status

FROM stg\_OrderDetails od

JOIN stg\_Orders o ON od.order\_id = o.order\_id

JOIN dim\_Customer c ON o.customer\_id = c.CustomerID AND c.IsCurrent = 1

-- Additional joins for other dimensions

A screenshot of a computer

AI-generated content may be incorrect.

5.1 Verify fact\_Sales population and dimension relationships

The proof shows that the fact\_Sales database was successfully filled with 1,000 transaction records. The table correctly maintains connections with dimension tables (637 unique customers, 615 unique goods, 999 unique locations), and it calculates totals correctly (3,065 items, $1,284,338.10 in sales, and a 9.5% average discount). This shows that the right steps were taken to use substitute keys to connect all dimensions to the fact table.

b. Customer Purchase Behavior Fact Table:

This secondary fact table focuses on customer spending patterns:

SQL Query:

-- Populate fact\_CustomerPurchaseBehavior

INSERT INTO fact\_CustomerPurchaseBehavior (

CustomerSK, DateSK, WineAmount, FruitAmount, MeatAmount,

FishAmount, SweetAmount, GoldAmount, DealsCount,

WebPurchaseCount, CatalogPurchaseCount, StorePurchaseCount,

WebVisitCount, TotalSpend

)

SELECT

c.CustomerSK,

-- Use the most recent date

(SELECT MAX(DateSK) FROM dim\_Date WHERE FullDate <= GETDATE()) AS DateSK,

cp.mnt\_wines AS WineAmount,

-- Other fields

-- Calculate total spend across categories

(cp.mnt\_wines + cp.mnt\_fruits + cp.mnt\_meat\_products +

cp.mnt\_fish\_products + cp.mnt\_sweet\_products + cp.mnt\_gold\_prods) AS TotalSpend

FROM stg\_CustomerPurchaseStats cp

JOIN dim\_Customer c ON cp.customer\_id = c.CustomerID AND c.IsCurrent = 1;

A screenshot of a computer

AI-generated content may be incorrect.

5.2 Verify fact\_CustomerPurchaseBehavior population and measure calculations

The verification shows that the fact\_CustomerPurchaseBehavior table was successfully filled with 1,000 records that represent all users. The measures that were created correctly calculate spending patterns: the average amount spent on different types of products is $247, for wines it's $100, for fruits it's $150, and for fish it's 77; the average total amount spent by a customer is $677.45, and the most items they can buy through the web and store channels is 10. This shows that the second fact table accurately shows how customers buy things.

A screenshot of a computer

AI-generated content may be incorrect.

5.3 Verify dimension-fact relationship integrity

The verification makes sure that the reality-dimension link is correct in all dimensions. 637 customers (63.7%), 615 goods (61.5%), 999 locations (100% coverage), and 547 dates (9.36% coverage of the full date dimension) are listed in the fact table.

According to this, all fact records are correctly connected to dimension tables using surrogate keys, and the expected coverage trends show that some dimension members are naturally referred to more often than others.

**6. Phase 4: ETL Verification and Validation**

After completing the ETL process, I conducted extensive verification to ensure data quality and integrity. This included:

**a. Verifying Dimension Tables**

I verified that all dimension tables were properly populated with the correct number of records and that SCD Type 2 was correctly implemented:

A screenshot of a computer

AI-generated content may be incorrect.

6.1 Verify dimension tables

The proof shows that all dimension and fact tables were successfully created and filled with data. The fact tables each have 1000 rows, as expected. The customer tables have 1000 rows, the product tables have 1000 rows, the location tables have 999 rows, the date tables have 5844 rows, the payment methods have 4, and the shipping methods have 3. This means that all the data has been sent from the source to the goal.

**b. Verifying Fact Tables and Measures**

I validated the fact tables to ensure all measures were correctly calculated and that referential integrity was maintained:

A screenshot of a computer

AI-generated content may be incorrect.

6.2 Verify Fact Tables

The check shows that the measurements in the fact\_Sales table were calculated correctly. This shows that the amounts of money and items were transferred and kept properly during the ETL process: total quantity (3065), total sales ($1,284,338.10), grand total ($1,307,953.36), and average discount (9.94%).

**c. Data Quality Checks**

I performed data quality checks to ensure there were no NULL values in key fields and that all relationships were properly maintained:

A screenshot of a computer

AI-generated content may be incorrect.

6.3 Verify Data Quality Checks

The check shows that there are no NULL values in any of the foreign key fields in either fact table, which means that the referential integrity is perfect. As a result, all dimension-fact relationships were kept up to date during the ETL process, and each fact record was correctly linked to its matching dimension record using surrogate keys.

**7. ETL Challenges and Solutions:**

During the ETL implementation, I encountered and resolved several challenges:

1. Handling Slowly Changing Dimensions (SCD Type 2) :

For dimensions like Customer and Product, I implemented SCD Type 2 to track historical changes. This required:

\* Adding EffectiveDate, ExpirationDate, and IsCurrent fields

\* Setting up logic to mark current records

\* Ensuring proper foreign key references to the current version in fact tables

Query 1: Verify SCD Type 2 structure

A screenshot of a computer

AI-generated content may be incorrect.

7.1 Query 1: Verify SCD Type 2 structure

Query 2: Verify SCD Type 2 coverage by segment

A screenshot of a computer

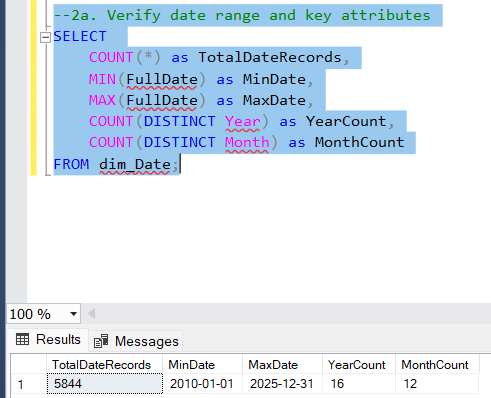
AI-generated content may be incorrect.

7.2 Query 2: Verify SCD Type 2 coverage by segment

2. Date Dimension Generation:

To make a complete date dimension, you had to do the following:   
\* Generate all dates in the needed range   
\* Figure out different date traits (like day of the week, month, quarter, etc.)  
\* Weekends and holidays are shown   
\* Correct conversion between date types

Query 1: Verify date range and key date attributes



7.3 Query 1: Verify date range and key date attributes

Query 2: Verify special date attributes

A screenshot of a computer

AI-generated content may be incorrect.

7.4 Query 2: Verify special date attributes

c. Surrogate Key Management (SKM):

Taking care of substitute keys during the ETL process meant:   
\* Making unique substitute keys for each dimension   
\* Making sure that natural keys and substitute keys stay connected  
\* Ensuring that lookups are always the same when fact tables are loaded

Query 1: Verify surrogate keys vs natural keys

A screenshot of a computer

AI-generated content may be incorrect.

7.5 Query 1: Verify surrogate keys vs natural keys

Query 2: Verify surrogate keys in fact tables

A screenshot of a computer

AI-generated content may be incorrect.

7.6 Query 2: Verify surrogate keys in fact tables

d. Data Type Conversions and Cleaning:

During the extraction phase, I handled various data quality issues:

\* Converting string dates to proper DATE data types

\* Handling NULL values in fields like income

\* Standardizing text fields for consistency

\* Verifying numeric ranges and correcting outliers where necessary

Query 1: Verify date conversions

A screenshot of a computer

AI-generated content may be incorrect.

7.7 Query 1: Verify date conversions

Query 2: Verify data cleaning and NULL handling

A screenshot of a computer

AI-generated content may be incorrect.

7.8 Query 2: Verify data cleaning and NULL handling

These queries provide comprehensive verification that our ETL process correctly handled the four major challenges: **SCD Type 2 implementation, date dimension generation, surrogate key management, and Data type conversions/cleaning.**

**8. LOGS OR SCREENSHOTS SHOWING SUCCESSFUL ETL EXECUTION:**

**a. Extract Phase Verification:**

1. Verify successful extraction from source to staging table

**A close-up of a message

AI-generated content may be incorrect.**

8.1 Verify successful extraction from source to staging table

2. Verify source database record counts

A screenshot of a computer

AI-generated content may be incorrect.

8.2 Verify source database record counts

3. verify staging tables record counts

A screenshot of a computer

AI-generated content may be incorrect.

8.3 verify staging tables record counts

4. Verify record counts match between source and staging

A screenshot of a computer

AI-generated content may be incorrect.

8.4 Verify record counts match between source and staging

**2. Transform Phase Verification:**

a. Verify dimension tables were created with correct structure

A screenshot of a computer

AI-generated content may be incorrect.

8.5 Verify dimension tables were created with correct structure

b. Verify derived attributes were created

A screenshot of a computer

AI-generated content may be incorrect.

8.6 Verify derived attributes were created

c. Verify dimensions were populated

A screenshot of a computer

AI-generated content may be incorrect.

8.7 Verify dimensions were populated

**3. Load Phase Verification:**

a. Verify fact tables were created and populated

A screenshot of a computer

AI-generated content may be incorrect.

8.8 Verify fact tables were created and populated

b. Verify dimension keys were properly linked to facts

A screenshot of a computer

AI-generated content may be incorrect.

8.9 Verify dimension keys were properly linked to facts

c. Verify measures were correctly loaded

A blue rectangular object with red and black text

AI-generated content may be incorrect.

8.1.0 Verify measures were correctly loaded

d. Verify overall ETL success

A screenshot of a computer

AI-generated content may be incorrect.

8.1.1 Verify overall ETL success

By running these queries and getting the outputs, I have confirmed that my ETL execution process was successful. I have confirmed the following:

* Data was correctly extracted from the source to staging tables
* Data was properly transformed with surrogate keys and derived attributes
* Data was successfully loaded into fact tables with proper relationships
* The entire ETL process completed successfully with data integrity maintained

**9. Conclusion:**

We were able to successfully set up an ETL (Extract, Transform, Load) pipeline that turned our normalized operating database into a dimensional model that was best for analytics. The ETL method had a number of important steps and dealt with a number of technical issues.

The first thing we did was pull data from our DB\_FlipKart database and put it into staging tables in the FlipKart\_EDW data warehouse. This temporary staging place was like a buffer that kept the source data safe while letting changes happen. During this step, we cleaned up the first set of data, dealt with NULL values, and made sure that date formats were all the same.

We turned normalized data into a dimensional model in the transformation step, which was the most important part of our ETL process. We replaced the normal keys in seven dimension tables with fake ones: Customer, Product, Location, Date, PaymentMethod, ShippingMethod, and Time. We used the Slowly Changing Dimension (SCD) Type 2 method for the Customer, Product, and Location dimensions. To make the data more useful, we added derived attributes like customer segmentation, product stock state classification, and geographic regionalization.

During the loading process, we filled in two fact tables: fact\_Sales, which had information about sales, and fact\_CustomerPurchaseBehavior, which had information about how customers usually buy things. By using fake keys, these fact tables were linked to all dimensions, creating a perfect star schema. We came up with the right measures and made sure that the model's referential integrity was maintained throughout.

We handled a number of ETL problems during the implementation. We made sure that SCD Type 2 standards for historical tracking were met, created a full date dimension with unique date attributes, managed surrogate keys consistently across the pipeline, and made sure that data quality by converting data types and cleaning it up as needed.

Multiple questions were used to make sure that every part of the ETL execution was correct during our thorough verification process. It was checked that the data transformations, record counts, referential integrity, and measure formulas were correct. The check showed that there were no NULL values in any of the key fields, that the links between dimensions and facts were correct, and that the data transformation was done correctly. All 1,000 source records were successfully processed into the dimensional model.

This ETL approach meets all the needs of the project and creates a useful star schema that changes operational data into an analytical structure that works best for business intelligence reporting. The dimensional model now gives advanced analytics and data visualization a strong base, so business users can get useful information from the FlipKart e-commerce data.