**UK Train Railway Data Cleaning and Preprocessing**

**Step 1: Create Dim\_Ticket**

**Objective:** Move purchase details to a separate table and replace purchase-related information with Purchase ID.

1. **Extract Unique Purchases**
   * Select the Purchase Type.
   * Remove Duplicates to keep unique purchase records.
   * Rename the query as **Dim\_Purchase**.
2. **Add an Index Column (Purchase ID)** => Start from 1
   * Rename the column to Purchase ID.
3. **Merge Purchase Data into Fact\_Transactions**
   * Match **Purchase Type** in **Fact\_Transaction** with **Dim\_Purchase**.
   * Expand the merged table to keep only Purchase ID.
4. **Remove the Original Purchase Columns**
   * Delete the Purchase Type column from Fact\_Transaction.

**Step : Create Dim\_Payment**

**Objective:** Move payment details to a separate table and replace payment method with Payment ID.

1. **Extract Unique Payment Methods**
   * Remove Duplicates to keep unique payment records.
   * Rename the query as **Dim\_Payment**.
2. **Add an Index Column (Payment ID)** => Start from 1
   * Rename the column to Payment ID.
3. **Merge Payment Data into Fact\_Transactions**
   * Match Payment Method in Fact\_Transaction with Dim\_Payment.
   * Expand the merged table to keep only Payment ID.
4. **Remove the Original Payment Columns**
   * Delete Payment Method from Fact\_Transaction.

**Step 4: Create Dim\_Railcard**

**Objective:** Move railcard discount details to a separate table and replace them with **Railcard ID**.

1. **Extract Unique Railcard Details**
   * Select **Railcard Type**.
   * Remove Duplicates to keep unique railcard records.
   * Rename the query as **Dim\_Railcard**.
2. **Add an Index Column (Railcard ID)** => Start from 1
   * Rename the column to Railcard ID.
3. **Add a Conditional Column for Railcard Holder**

Set the condition:

* + If Railcard Type = "None", then "Non-Holder"
  + Else, "Holder"
* Rename the new column as **Railcard Holder**.

1. **Merge Railcard Data into Fact\_Transactions**
   * Match **Railcard Type** in **Fact\_Transaction** with **Dim\_Railcard**.
   * Expand the merged table to keep only **Railcard ID**.
2. **Remove the Original Railcard Columns**
   * Delete **Railcard Type** from **Fact\_Transaction**.

**Step 5: Create Dim\_Ticket**

**Objective:** Move ticket details to a separate table and replace ticket type and class with **Ticket ID**.

1. **Extract Unique Ticket Details**
   * Select **Ticket Type** and **Ticket Class**.
   * Remove Duplicates to keep unique ticket records.
   * Rename the query as **Dim\_Ticket**.
2. **Add an Index Column (Ticket ID)** => Start from 1
   * Rename the column to **Ticket ID**.
3. **Merge Ticket Data into Fact\_Transactions**
   * Match Ticket Type & Class in Fact\_Transaction with Dim\_Ticket.
   * Expand the merged table to keep only **Ticket ID**.
4. **Remove the Original Ticket Columns**
   * Delete Ticket Type and Ticket Class from Fact\_Transaction.

**Step 6: Create Dim\_Departure**

**Objective:** Move departure station and time to a separate table and replace them with **Departure ID**.

1. **Extract Unique Departure Stations**
   * Select **Departure Station** .
   * Remove Duplicates to keep unique departure records.
   * Rename the query as **Dim\_Departure**.
2. **Add an Index Column (Departure ID)** => Start from 1
   * Rename the column to **Departure ID**.
3. **Merge Departure Data into Fact\_Transactions**
   * Match **Departure Station** in **Fact\_Transaction** with **Dim\_Departure**.
   * Expand the merged table to keep only **Departure ID**.
4. **Remove the Original Departure Columns**
   * Delete **Departure Station** from **Fact\_Transaction**.

**Step 7: Create Dim\_Arrival**

**Objective:** Move arrival station and time to a separate table and replace them with **Arrival ID**.

1. **Extract Unique Arrival Stations**
   * Select **Arrival Station**.
   * Remove Duplicates to keep unique arrival records.
   * Rename the query as **Dim\_Arrival**.
2. **Add an Index Column (Arrival ID)** => Start from 1
   * Rename the column to **Arrival ID**.
3. **Merge Arrival Data into Fact\_Transactions**
   * Match **Arrival Station** in **Fact\_Transaction** with **Dim\_Arrival**.
   * Expand the merged table to keep only **Arrival ID**.
4. **Remove the Original Arrival Columns**
   * Delete **Arrival Station** from **Fact\_Transaction**.

**Step 8: Create Dim\_Delay Table**

**Objective:** Move train delay details to a separate table and replace them with **Delay ID**.

1. **Extract Unique Delay Records**
   1. Select **Reason for Delay**.
   2. Remove Duplicates to keep unique delay records.
   3. Rename the query as **Dim\_Delay**.
2. **Replace Inconsistent Values**

Go to **Transform** → **Replace Values**.

Apply the following replacements:

* 1. "Signal failure" → "Signal Failure"
  2. "Staff Shortage" → "Staffing"
  3. "Weather Conditions" → "Weather"

1. **Verify the Changes**

Scroll through the column to ensure that the replacements have been correctly applied.

1. **Add an Index Column (Delay ID)** => Start from 1
   1. Rename the column to **Delay ID**.
2. **Merge Delay Data into Fact\_Transactions**
   1. Match **Reason for Delay** in **Fact\_Transaction** with **Dim\_Delay**.
   2. Expand the merged table to keep only **Delay ID**.
3. **Remove the Original Delay Columns**
   1. Delete **Reason for Delay** from **Fact\_Transaction**.

**Step 9: Create Dim\_Calendar**

**Objective:** Create a Date Dimension table to support time-based analysis, including trends over years, months, weeks, and days.

**Power Query Code for Dim\_Calendar:**

// Create Date Dimension

(StartDate as date, EndDate as date) =>

let

// Capture the date range from the parameters

StartDate = #date(Date.Year(StartDate), Date.Month(StartDate), Date.Day(StartDate)),

EndDate = #date(Date.Year(EndDate), Date.Month(EndDate), Date.Day(EndDate)),

// Get the number of dates required for the table

GetDateCount = Duration.Days(EndDate - StartDate),

// Generate a list of dates

GetDateList = List.Dates(StartDate, GetDateCount, #duration(1,0,0,0)),

// Convert the list into a table

DateListToTable = Table.FromList(GetDateList, Splitter.SplitByNothing(), {"Date"}, null, ExtraValues.Error),

// Add Year Column

YearNumber = Table.AddColumn(DateListToTable, "Year", each Date.Year([Date])),

// Add Quarter Column

QuarterNumber = Table.AddColumn(YearNumber, "Quarter", each "Q" & Number.ToText(Date.QuarterOfYear([Date]))),

// Add Week Number Column

WeekNumber = Table.AddColumn(QuarterNumber, "Week Number", each Date.WeekOfYear([Date])),

// Add Month Number Column

MonthNumber = Table.AddColumn(WeekNumber, "Month Number", each Date.Month([Date])),

// Add Month Name Column

MonthName = Table.AddColumn(MonthNumber, "Month", each Date.ToText([Date],"MMMM")),

// Add Day of Week Column

DayOfWeek = Table.AddColumn(MonthName, "Day of Week", each Date.ToText([Date],"dddd"))

in

DayOfWeek

1. **Fact\_Transaction**

**Objective:**

The **Fact\_Transaction** table captures transactional details related to train journeys, including ticket purchases, journey times, delays, and payments. It links to various dimension tables to ensure a well-structured and optimized data model. Ensure data integrity and replace NULL values in Actual Arrival Time for canceled journeys.

Ensure data integrity and replace NULL values in Actual Arrival Time for canceled journeys.

**1. Handling NULL Values in Actual Arrival Time**

* If the Status field is marked as "Canceled," replace NULL values in the Actual Arrival Time column with "00:00:00".

**2. Grouping Queries**

* Staging Area Group: Create a dedicated staging area group for all transformation-related queries to ensure a structured data processing pipeline.
* Fact Tables Group: Organize Fact\_Transication into a separate group for easy access and management.
* Dimension Tables Group: Classify and store all dimension tables in a separate group to maintain a well-structured data model.

**Conclusion**

This document outlines a structured approach to cleaning, transforming, and optimizing the dataset for efficient reporting. The final model ensures data consistency, eliminates redundancies, and enhances analytical capabilities, ultimately supporting more effective decision-making.