PRESTIGE CARS NORMALIZATION

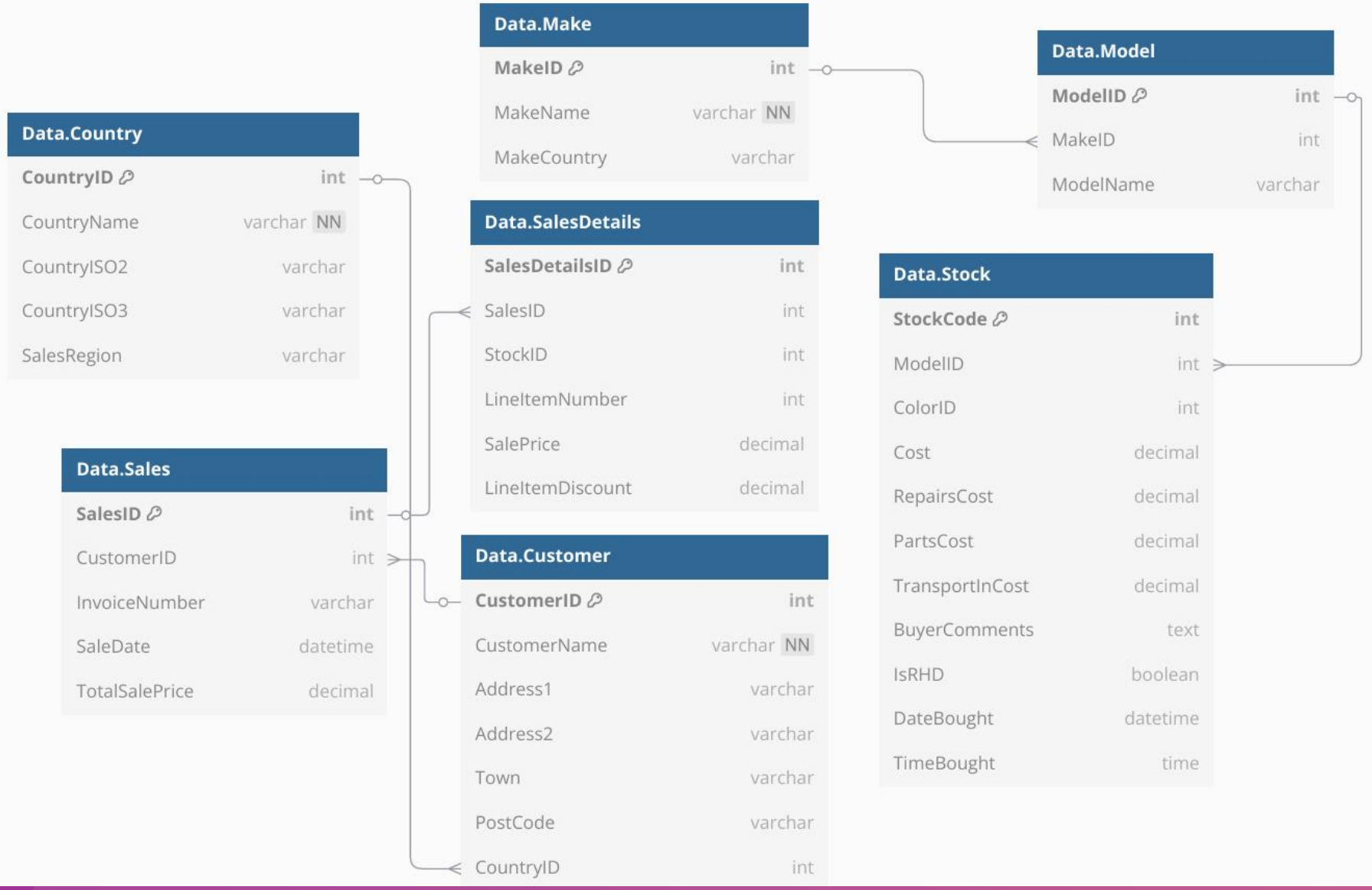
PRESENTED BY GROUP#3

OVERVIEW OF THE PROJECT

1. The Prestige Cars database normalization project involves restructuring the schema, introducing new tables, and ensuring compliance with best practices for database design. The objectives include schema redesign, data cleansing, and indexing for optimized performance.

Introduction

This document outlines the process of normalizing the Prestige Cars database into a relational database in 3NF. The project aims to enhance data integrity, promote reuse through user-defined datatypes, and implement robust constraints for data validation. The deliverables include a database backup file and an annotated presentation.


Data.Make
MakeID int
MakeName varchar NN
MakeCountry varchar
Data.Model
ModelID int
MakeID int
ModelName varchar
Data.Country
CountryID int
CountryName varchar NN
CountryISO2 varchar
CountryISO3 varchar
SalesRegion varchar
Data.SalesDetails
SalesDetailsID int
SalesID int
StockID int
LineItemNumber int
SalePrice decimal
LineItemDiscount decimal
Data.Stock
StockCode int
ModelID int
ColorID int
Cost decimal
RepairsCost decimal
PartsCost decimal
TransportInCost decimal
BuyerComments text
IsRHD boolean
DateBought datetime
TimeBought time
Data.Sales
SalesID int
CustomerID int
InvoiceNumber varchar
SaleDate datetime
TotalSalePrice decimal
Data.Customer
CustomerID int
CustomerName varchar NN
Address1 varchar
Address2 varchar
Town varchar
PostCode varchar
CountryID int

📘 Summary Table Of Cardinality

From Table	To Table	Cardinality	Notes
Country	Customer	1:N	Each country has many customers
Country	Make	1:N	Each country can be origin of many makes
Color	Stock	1:N	Each color used in many vehicles
Make	Model	1:N	A make has many models
Model	Stock	1:N	A model maps to many stock vehicles
Customer	Sales	1:N	A customer can place many sales orders
Sales	SalesDetails	1:N	A sale has many line items
Stock	SalesDetails	1:1 or 1:N	Ideally 1:1 (one car sold once)
Color	PivotTable	1:1	One color per Pivot row

Step 2. recreate tables use the script to make constrains and checks. Use the udts

```
-- Country Table
CREATE TABLE Data.Country (
    CountryID UDT_CountryID PRIMARY KEY,
    CountryName UDT_CountryName NOT NULL DEFAULT N'Unknown',
    CountryISO2 UDT_CountryISO2 DEFAULT N'--',
    CountryISO3 UDT_CountryISO3 DEFAULT N'---',
    SalesRegion UDT_Region DEFAULT N'Unknown'
);
GO

-- Make Table
CREATE TABLE Data.Make (
    MakeID UDT_MakeID PRIMARY KEY,
    MakeName UDT_MakeName NOT NULL DEFAULT N'Unknown Make',
    MakeCountry UDT_MakeCountry DEFAULT '---'
);
GO


-- Model Table
CREATE TABLE Data.Model (
    ModelID UDT_ModelID PRIMARY KEY,
    MakeID UDT_MakeID FOREIGN KEY REFERENCES Data.Make(MakeID),
    ModelName UDT_ModelName DEFAULT N'Unknown Model',
);
GO
```

- ASHLY

```sql
-- Model Table
CREATE TABLE Data.Model (
    ModelID UDT_ModelID PRIMARY KEY,
    MakeID UDT_MakeID FOREIGN KEY REFERENCES Data.Make(MakeID),
    ModelName UDT_ModelName DEFAULT N'Unknown Model',
);
GO


-- Customer Table
CREATE TABLE Data.Customer (
    CustomerID UDT_CustomerID PRIMARY KEY,
    CustomerName UDT_CustomerName NOT NULL DEFAULT N'Anonymous',
    Address1 UDT_Address DEFAULT N'Address Not Provided',
    Address2 UDT_Address DEFAULT N'Address Not Provided',
    Town UDT_Address DEFAULT N'Town not found',
    PostCode UDT_Address DEFAULT N'00000',
    CountryID UDT_CountryID FOREIGN KEY REFERENCES Data.Country(CountryID),
    IsReseller UDT_IsReseller DEFAULT 0,
    IsCreditRisk UDT_IsCreditRisk DEFAULT 0
);
GO


-- Sales Table
CREATE TABLE Data.Sales (
    SalesID UDT_SalesID PRIMARY KEY,
    CustomerID UDT_CustomerID FOREIGN KEY REFERENCES Data.Customer(CustomerID),
    InvoiceNumber UDT_InvoiceNumber DEFAULT '00000000',
    SaleDate UDT_SaleDate DEFAULT GETDATE(),
    TotalSalePrice UDT_TotalSalePrice DEFAULT 0
);
GO
```

```sql
-- Stock Table (foreign keys aligned)
CREATE TABLE Data.Stock (
    StockCode UDT_StockCode PRIMARY KEY,
    ModelID UDT_ModelID FOREIGN KEY REFERENCES Data.Model(ModelID),
    ColorID UDT_ColorID,  -- FK removed if Color table doesn't exist
    Cost UDT_Cost DEFAULT 0,
    RepairsCost UDT_RepairsCost DEFAULT 0,
    PartsCost UDT_PartsCost DEFAULT 0,
    TransportInCost UDT_TransportCost DEFAULT 0,
    BuyerComments UDT_BuyerComments DEFAULT N'',
    IsRHD UDT_IsReseller DEFAULT 0,
    DateBought UDT_DateBought DEFAULT GETDATE(),
    TimeBought UDT_TimeBought DEFAULT CAST(GETDATE() AS TIME(7))
);
GO


-- SalesDetails Table (foreign keys aligned)
CREATE TABLE Data.SalesDetails (
    SalesDetailsID INT PRIMARY KEY,
    SalesID UDT_SalesID FOREIGN KEY REFERENCES Data.Sales(SalesID),
    StockID UDT_StockCode,  -- FK removed if Color table doesn't exist
    LineItemNumber UDT_LineItemNumber DEFAULT 1,
    SalePrice UDT_SalePrice DEFAULT 0,
    LineItemDiscount UDT_LineItemDiscount DEFAULT 0
);
GO
```

- ASHLY

Notes as a "driver"
What does it mean to clean up the tables?

Notes

Data.country
Sales region grouping should probably just use acronyms
https://en.wikipedia.org/wiki/List_of_country_groupings

Data.customer
Need to look into the postcodes
Country is kinda redundant here since it's already in its own table. Not sure what changes to make. Someone help
Address1 data format is inconsistent. There's commas in some and the lot number is at the end for some

Data.make
tentative

Data.Model
Yearlastproduced is completely null and so is yearfirstproduced
Modelname is repeated
Is make and model necessary?

Data.PivotTable
Replace the nulls with 0.00

Data.Sales
Nothing to change here

Data.SalesDetails
Replace the nulls in LineItemDiscount with 0.00

Data.SalesRegion
POPULATE THIS. this is empty

Data.Stock
Are the Colors columns inconsistent? What is Blue and Night Blue?
Remove nulls in modelid, cost, repaircost, partscost, transportincost, color, buyercomments, datebought, timebought
All the rows in TimeBought are the same. Change this

- MAITRI

Data Normalization Steps:

1. Create Schemas and User-Defined Types
2. DefineTables (3NF)
3. Insert and Clean Data

4. Create Utility Procesdures (Truncate and Drop FKs)

5. Create Views and Functions

6. Validation and Verificaton

- MEHTAB

Data Normalization Steps:

1. Create Schemas and User-Defined Types
   i) Create new 3NF database and switch context
   ii) Create schemas to organiza tables
   iii) Define all UDTs if they did not exist already for all tables

- MEHTAB

```sql
-- Create new 3NF database and switch context to it
IF DB_ID('PrestigeCars_3NF') IS NULL
    CREATE DATABASE PrestigeCars_3NF;
GO
USE PrestigeCars_3NF;
GO

-- Create necessary schemas for organizing tables
CREATE SCHEMA [Data];
GO
CREATE SCHEMA [Reference];
GO
CREATE SCHEMA [SourceData];
GO
CREATE SCHEMA [DataTransfer];
GO
CREATE SCHEMA [Output];
GO
CREATE SCHEMA [Project2.5];
GO
CREATE SCHEMA [Process];
GO
```

->

```sql
/* CreateUDTs.sql - Author: Mehtab Mahir */
-- Define all User-Defined Types (UDTs) if they do not already exist
-- COUNTRY-related UDTs
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_CountryID')
    CREATE TYPE dbo.UDT_CountryID FROM INT;
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_CountryName')
    CREATE TYPE dbo.UDT_CountryName FROM NVARCHAR(150);
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_CountryISO2')
    CREATE TYPE dbo.UDT_CountryISO2 FROM NCHAR(10);
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_CountryISO3')
    CREATE TYPE dbo.UDT_CountryISO3 FROM NCHAR(10);
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_Region')
    CREATE TYPE dbo.UDT_Region FROM NVARCHAR(20);

-- MAKE/MODEL-related UDTs
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_MakeID')
    CREATE TYPE dbo.UDT_MakeID FROM INT;
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_MakeName')
    CREATE TYPE dbo.UDT_MakeName FROM NVARCHAR(100);
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_MakeCountry')
    CREATE TYPE dbo.UDT_MakeCountry FROM CHAR(3);
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_ModelID')
    CREATE TYPE dbo.UDT_ModelID FROM INT;
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_ModelName')
    CREATE TYPE dbo.UDT_ModelName FROM NVARCHAR(150);
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_ModelVariant')
    CREATE TYPE dbo.UDT_ModelVariant FROM NVARCHAR(150);
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_Year')
    CREATE TYPE dbo.UDT_Year FROM CHAR(4);
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_IsRHD')
    CREATE TYPE dbo.UDT_IsRHD FROM BIT;
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_Color')
    CREATE TYPE dbo.UDT_Color FROM NVARCHAR(50);
```

```sql
-- CUSTOMER-related UDTs
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_CustomerID')
    CREATE TYPE dbo.UDT_CustomerID FROM NVARCHAR(5);
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_CustomerName')
    CREATE TYPE dbo.UDT_CustomerName FROM NVARCHAR(150);
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_Address')
    CREATE TYPE dbo.UDT_Address FROM NVARCHAR(50);
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_CountryRef')
    CREATE TYPE dbo.UDT_CountryRef FROM INT;
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_IsReseller')
    CREATE TYPE dbo.UDT_IsReseller FROM BIT;
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_IsCreditRisk')
    CREATE TYPE dbo.UDT_IsCreditRisk FROM BIT;

-- STOCK-related UDTs
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_StockCode')
    CREATE TYPE dbo.UDT_StockCode FROM NVARCHAR(50);
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_ModelRef')
    CREATE TYPE dbo.UDT_ModelRef FROM SMALLINT;
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_ColorID')
    CREATE TYPE dbo.UDT_ColorID FROM INT;
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_Cost')
    CREATE TYPE dbo.UDT_Cost FROM MONEY;
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_RepairsCost')
    CREATE TYPE dbo.UDT_RepairsCost FROM MONEY;
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_PartsCost')
    CREATE TYPE dbo.UDT_PartsCost FROM MONEY;
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_TransportCost')
    CREATE TYPE dbo.UDT_TransportCost FROM MONEY;
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_BuyerComments')
    CREATE TYPE dbo.UDT_BuyerComments FROM NVARCHAR(4000);
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_DateBought')
    CREATE TYPE dbo.UDT_DateBought FROM DATE;
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_TimeBought')
    CREATE TYPE dbo.UDT_TimeBought FROM TIME(7);
```

```sql
-- SALES-related UDTs
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_SalesID')
    CREATE TYPE dbo.UDT_SalesID FROM INT;
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_InvoiceNumber')
    CREATE TYPE dbo.UDT_InvoiceNumber FROM CHAR(8);
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_SaleDate')
    CREATE TYPE dbo.UDT_SaleDate FROM DATETIME;
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_TotalSalePrice')
    CREATE TYPE dbo.UDT_TotalSalePrice FROM NUMERIC(18,2);

-- SALES DETAILS-related UDTs
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_SalesDetailsID')
    CREATE TYPE dbo.UDT_SalesDetailsID FROM INT;
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_LineItemNumber')
    CREATE TYPE dbo.UDT_LineItemNumber FROM TINYINT;
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_SalePrice')
    CREATE TYPE dbo.UDT_SalePrice FROM NUMERIC(18,2);
IF NOT EXISTS (SELECT 1 FROM sys.types WHERE name = 'UDT_LineItemDiscount')
    CREATE TYPE dbo.UDT_LineItemDiscount FROM NUMERIC(18,2);
GO
```

Data Normalization Steps:
2. DefineTables (3NF)
    i) Reference schema tables
    ii) Data schema tables
    iii) DataTransfer schema tables
    iv) SourceData schema tables
    v) Output schema tables

The steps taken for all the schema tables were to
Drop the table if it exists then create new tables with the user defined types.

- MEHTAB

```sql
-- Reference.Country: list of countries (normalized from Data.Custom
DROP TABLE IF EXISTS Reference.Country;
CREATE TABLE Reference.Country (
    CountryID     dbo.UDT_CountryID     IDENTITY(1,1) PRIMARY KEY,
    CountryName   dbo.UDT_CountryName   NOT NULL,
    CountryISO2   dbo.UDT_CountryISO2   NOT NULL,
    CountryISO3   dbo.UDT_CountryISO3   NULL,
    SalesRegion   dbo.UDT_Region        NULL,
    CountryFlag       VARBINARY(MAX)    NULL,
    FlagFileName      NVARCHAR(50)      NULL,
    FlagFileType      NCHAR(3)          NULL
);
GO

-- Reference.Color: list of distinct vehicle colors
DROP TABLE IF EXISTS Reference.Color;
CREATE TABLE Reference.Color (
    ColorID    dbo.UDT_ColorID    IDENTITY(1,1) PRIMARY KEY,
    Color      dbo.UDT_Color      NOT NULL
);
GO

-- Reference.Budget: budget values by period and category
DROP TABLE IF EXISTS Reference.Budget;
CREATE TABLE Reference.Budget (
    BudgetKey       INT             IDENTITY(1,1) PRIMARY KEY,
    BudgetValue     MONEY           NULL,
    Year            INT             NULL,
    Month           TINYINT         NULL,
    BudgetDetail    NVARCHAR(50)    NULL,
    BudgetElement   NVARCHAR(50)    NULL
);
GO
```

```sql
-- Reference.Forex: foreign exchange rates (currency conversion rates)
DROP TABLE IF EXISTS Reference.Forex;
CREATE TABLE Reference.Forex (
    ExchangeDate   DATE          NULL,
    ISOCurrency    CHAR(3)       NULL,
    ExchangeRate   MONEY         NULL
);
GO

-- Reference.MarketingCategories: marketing categories for makes
DROP TABLE IF EXISTS Reference.MarketingCategories;
CREATE TABLE Reference.MarketingCategories (
    MakeName        NVARCHAR(100) NULL,
    MarketingType   NVARCHAR(200) NULL
);
GO

-- Reference.MarketingInformation: customer marketing info (external data)
DROP TABLE IF EXISTS Reference.MarketingInformation;
CREATE TABLE Reference.MarketingInformation (
    CUST            NVARCHAR(150) NULL,    -- Customer name or code
    Country         NCHAR(10)     NULL,    -- Country code
    SpendCapacity   VARCHAR(25)   NULL
);
GO

-- Reference.SalesBudgets: yearly sales budget by Color (normalized from original)
DROP TABLE IF EXISTS Reference.SalesBudgets;
CREATE TABLE Reference.SalesBudgets (
    SalesBudgetID   dbo.UDT_SalesID    IDENTITY(1,1) PRIMARY KEY,
    ColorID         dbo.UDT_ColorID    NOT NULL,
    BudgetYear      dbo.UDT_Year       NOT NULL,
    BudgetAmount    dbo.UDT_SalePrice  NOT NULL,
    CONSTRAINT FK_SalesBudgets_Color FOREIGN KEY (ColorID)
        REFERENCES Reference.Color(ColorID)
);
GO
```

REFERENCE SCHEMA TABLES

```sql
-- Reference.SalesCategory: sales category thresholds
DROP TABLE IF EXISTS Reference.SalesCategory;
CREATE TABLE Reference.SalesCategory (
    LowerThreshold       INT           NULL,
    UpperThreshold       INT           NULL,
    CategoryDescription  NVARCHAR(50)  NULL
);
GO

-- Reference.Staff: staff members with hierarchy info
DROP TABLE IF EXISTS Reference.Staff;
CREATE TABLE Reference.Staff (
    StaffID     INT           IDENTITY(1,1) PRIMARY KEY,
    StaffName   NVARCHAR(50)  NULL,
    ManagerID   INT           NULL,
    Department  NVARCHAR(50)  NULL
);
GO
```

```sql
-- Reference.StaffHierarchy: hierarchy representation of staff (using hierarchyid)
DROP TABLE IF EXISTS Reference.StaffHierarchy;
CREATE TABLE Reference.StaffHierarchy (
    HierarchyReference  HIERARCHYID   NULL,
    StaffID             INT           IDENTITY(1,1) PRIMARY KEY,
    StaffName           NVARCHAR(50)  NULL,
    ManagerID           INT           NULL,
    Department          NVARCHAR(50)  NULL
);
GO

-- Reference.YearlySales: combined sales data (2015-2018) - as per original structure
DROP TABLE IF EXISTS Reference.YearlySales;
CREATE TABLE Reference.YearlySales (
    MakeName       NVARCHAR(100) NULL,
    ModelName      NVARCHAR(150) NULL,
    CustomerName   NVARCHAR(150) NULL,
    CountryName    NVARCHAR(150) NULL,
    Cost           MONEY         NULL,
    RepairsCost    MONEY         NULL,
    PartsCost      MONEY         NULL,
    TransportInCost MONEY        NULL,
    SalePrice      NUMERIC(18,2) NULL,
    SaleDate       DATETIME      NULL
);
GO
```

RREFERENCE SCHEMA TABLES

```sql
-- Data.Make: car manufacturer (Make) with country reference
DROP TABLE IF EXISTS Data.Make;
CREATE TABLE Data.Make (
    MakeID       dbo.UDT_MakeID      IDENTITY(1,1) PRIMARY KEY,
    MakeName     dbo.UDT_MakeName    NOT NULL,
    -- Normalize country of origin into reference table
    CountryID    dbo.UDT_CountryRef  NULL,
    CONSTRAINT FK_Make_Country FOREIGN KEY (CountryID) REFERENCES Reference.Country(CountryID)
);
GO
-- The following Data.Model table definition is based on `data.model.sql`, contributed by Ashly.*

-- Data.Model: car model, linked to its Make
DROP TABLE IF EXISTS Data.Model;
CREATE TABLE Data.Model (
    ModelID           dbo.UDT_ModelID     IDENTITY(1,1) PRIMARY KEY,
    MakeID            dbo.UDT_MakeID      NOT NULL,
    ModelName         dbo.UDT_ModelName   NOT NULL,
    ModelVariant      dbo.UDT_ModelVariant NULL,
    YearFirstProduced dbo.UDT_Year        NULL,
    YearLastProduced  dbo.UDT_Year        NULL,
    CONSTRAINT FK_Model_Make FOREIGN KEY (MakeID) REFERENCES Data.Make(MakeID)
);
GO

-- Data.Customer: customer information
DROP TABLE IF EXISTS Data.Customer;
CREATE TABLE Data.Customer (
    CustomerID    dbo.UDT_CustomerID    NOT NULL PRIMARY KEY,
    CustomerName  dbo.UDT_CustomerName  NOT NULL  DEFAULT(''),   -- Default empty string for no name
    Address1      dbo.UDT_Address       NOT NULL  DEFAULT(''),
    Address2      dbo.UDT_Address       NULL      DEFAULT(''),
    Town          dbo.UDT_Address       NOT NULL  DEFAULT('Unknown'),
    PostCode      dbo.UDT_CountryISO2   NOT NULL  DEFAULT(''),
    CountryID     dbo.UDT_CountryRef    NOT NULL,  -- normalized country reference
    IsReseller    dbo.UDT_IsReseller    NOT NULL  DEFAULT(0),
    IsCreditRisk  dbo.UDT_IsCreditRisk  NOT NULL  DEFAULT(0),
    CONSTRAINT FK_Customer_Country FOREIGN KEY (CountryID) REFERENCES Reference.Country(CountryID)
);
GO
```

RDATA SCHEMA TABLES

```sql
-- Data.Stock: inventory of car stock (each vehicle), with reference to Model and Color
DROP TABLE IF EXISTS Data.Stock;
CREATE TABLE Data.Stock (
    StockCode       dbo.UDT_StockCode    NOT NULL PRIMARY KEY,  -- using StockCode as unique identifier for stoc
    ModelID         dbo.UDT_ModelRef     NOT NULL,
    Cost            dbo.UDT_Cost         NULL      DEFAULT(0),
    RepairsCost     dbo.UDT_RepairsCost  NULL      DEFAULT(0),
    PartsCost       dbo.UDT_PartsCost    NULL      DEFAULT(0),
    TransportInCost dbo.UDT_TransportCost NULL     DEFAULT(0),
    IsRHD           dbo.UDT_IsRHD        NOT NULL  DEFAULT(0),
    ColorID         dbo.UDT_ColorID      NULL,
    BuyerComments   dbo.UDT_BuyerComments NULL,
    DateBought      dbo.UDT_DateBought   NULL,
    TimeBought      dbo.UDT_TimeBought   NULL,
    CONSTRAINT FK_Stock_Model FOREIGN KEY (ModelID) REFERENCES Data.Model(ModelID),
    CONSTRAINT FK_Stock_Color FOREIGN KEY (ColorID) REFERENCES Reference.Color(ColorID)
);
GO

-- Data.Sales: sales transactions (one per invoice/sale)
DROP TABLE IF EXISTS Data.Sales;
CREATE TABLE Data.Sales (
    SalesID       dbo.UDT_SalesID       NOT NULL PRIMARY KEY,
    CustomerID    dbo.UDT_CustomerID    NOT NULL,
    InvoiceNumber dbo.UDT_InvoiceNumber NULL,
    TotalSalePrice dbo.UDT_TotalSalePrice NULL,
    SaleDate      dbo.UDT_SaleDate      NULL,
    CONSTRAINT FK_Sales_Customer FOREIGN KEY (CustomerID) REFERENCES Data.Customer(CustomerID)
    -- (Dropped redundant [ID] identity from original; SalesID serves as primary key)
);
GO
-- Data.SalesDetails: line items for each sale (each vehicle sold)
DROP TABLE IF EXISTS Data.SalesDetails;
CREATE TABLE Data.SalesDetails (
    SalesDetailsID    dbo.UDT_SalesDetailsID  IDENTITY(1,1) PRIMARY KEY,
    SalesID           dbo.UDT_SalesID         NOT NULL,
    LineItemNumber    dbo.UDT_LineItemNumber  NOT NULL,
    StockID           dbo.UDT_StockCode       NOT NULL,  -- references Stock.StockCode
    SalePrice         dbo.UDT_SalePrice       NULL      DEFAULT(0),
    LineItemDiscount  dbo.UDT_LineItemDiscount NULL     DEFAULT(0),
    CONSTRAINT FK_SalesDetails_Sales FOREIGN KEY (SalesID) REFERENCES Data.Sales(SalesID),
    CONSTRAINT FK_SalesDetails_Stock FOREIGN KEY (StockID) REFERENCES Data.Stock(StockCode)
    -- Ensure no duplicate line items per sale
    --, CONSTRAINT UQ_SalesDetails UNIQUE(SalesID, LineItemNumber)
);
GO

-- Data.PivotTable: pre-calculated sales totals per Color by year (2015-2018)
DROP TABLE IF EXISTS Data.PivotTable;
CREATE TABLE Data.PivotTable (
    ColorID   dbo.UDT_ColorID   NOT NULL,
    [2015]    dbo.UDT_SalePrice NULL,
    [2016]    dbo.UDT_SalePrice NULL,
    [2017]    dbo.UDT_SalePrice NULL,
    [2018]    dbo.UDT_SalePrice NULL,
    CONSTRAINT PK_PivotTable PRIMARY KEY CLUSTERED (ColorID),
    CONSTRAINT FK_PivotTable_Color FOREIGN KEY (ColorID) REFERENCES Reference.Color(ColorID)
);
GO
```

->

```sql
-- DataTransfer.Sales2015: staging data for sales in 2015 (raw import)
DROP TABLE IF EXISTS DataTransfer.Sales2015;
CREATE TABLE DataTransfer.Sales2015 (
    MakeName        NVARCHAR(100) NULL,
    ModelName       NVARCHAR(150) NULL,
    CustomerName    NVARCHAR(150) NULL,
    CountryName     NVARCHAR(150) NULL,
    Cost            MONEY         NULL,
    RepairsCost     MONEY         NULL,
    PartsCost       MONEY         NULL,
    TransportInCost MONEY         NULL,
    SalePrice       NUMERIC(18,2) NULL,
    SaleDate        DATETIME      NULL
);
GO

-- Repeat for 2016, 2017, 2018
DROP TABLE IF EXISTS DataTransfer.Sales2016;
CREATE TABLE DataTransfer.Sales2016 (
    MakeName        NVARCHAR(100) NULL,
    ModelName       NVARCHAR(150) NULL,
    CustomerName    NVARCHAR(150) NULL,
    CountryName     NVARCHAR(150) NULL,
    Cost            MONEY         NULL,
    RepairsCost     MONEY         NULL,
    PartsCost       MONEY         NULL,
    TransportInCost MONEY         NULL,
    SalePrice       NUMERIC(18,2) NULL,
    SaleDate        DATETIME      NULL
);
GO
```

```sql
DROP TABLE IF EXISTS DataTransfer.Sales2017;
CREATE TABLE DataTransfer.Sales2017 (
    MakeName        NVARCHAR(100) NULL,
    ModelName       NVARCHAR(150) NULL,
    CustomerName    NVARCHAR(150) NULL,
    CountryName     NVARCHAR(150) NULL,
    Cost            MONEY         NULL,
    RepairsCost     MONEY         NULL,
    PartsCost       MONEY         NULL,
    TransportInCost MONEY         NULL,
    SalePrice       NUMERIC(18,2) NULL,
    SaleDate        DATETIME      NULL
);
GO

DROP TABLE IF EXISTS DataTransfer.Sales2018;
CREATE TABLE DataTransfer.Sales2018 (
    MakeName        NVARCHAR(100) NULL,
    ModelName       NVARCHAR(150) NULL,
    CustomerName    NVARCHAR(150) NULL,
    CountryName     NVARCHAR(150) NULL,
    Cost            MONEY         NULL,
    RepairsCost     MONEY         NULL,
    PartsCost       MONEY         NULL,
    TransportInCost MONEY         NULL,
    SalePrice       NUMERIC(18,2) NULL,
    SaleDate        DATETIME      NULL
);
GO
```

DATATRANSFER SCHEMA TABLES

```
-- SourceData.SalesInPounds: raw sales cost data in GBP
DROP TABLE IF EXISTS SourceData.SalesInPounds;
CREATE TABLE SourceData.SalesInPounds (
    MakeName     NVARCHAR(100) NULL,
    ModelName    NVARCHAR(150) NULL,
    VehicleCost  VARCHAR(51)   NULL   -- costs in text (e.g., "£12345")
);
GO

-- SourceData.SalesText: raw sales data in text form
DROP TABLE IF EXISTS SourceData.SalesText;
CREATE TABLE SourceData.SalesText (
    CountryName  NVARCHAR(150) NULL,
    MakeName     NVARCHAR(100) NULL,
    Cost         VARCHAR(20)   NULL,
    SalePrice    VARCHAR(20)   NULL
);
GO

-- SourceData.SalesInPounds_Cleaned: cleaned & normalized version of SalesInPounds
DROP TABLE IF EXISTS SourceData.SalesInPounds_Cleaned;
CREATE TABLE SourceData.SalesInPounds_Cleaned (
    SalesInPoundsID  dbo.UDT_SalesID    IDENTITY(1,1) PRIMARY KEY,
    CustomerID       dbo.UDT_CustomerID NULL,
    StockID          dbo.UDT_StockCode  NULL,
    SaleDate         dbo.UDT_SaleDate   NULL,
    SalePriceGBP     dbo.UDT_SalePrice  NULL,
    ConvertedUSD     dbo.UDT_SalePrice  NULL  -- optional: store converted USD value
);
GO
```

OURCEDATA SCHEMA TABLES

```sql
-- Output.StockPrices: output table (e.g., for reporting stock costs by model)
DROP TABLE IF EXISTS Output.StockPrices;
CREATE TABLE Output.StockPrices (
    MakeName   NVARCHAR(100) NULL,
    ModelName  NVARCHAR(150) NULL,
    Cost       MONEY         NULL
);
GO
```

OUTPUT SCHEMA TABLES

Data Normalization Steps:
3. Insert and Clean Data

The steps here was to transfer all data from PrestigeCars into the new normalized Databse all while cleaning the data.

- MEHTAB

->

```sql
-- Insert Cleaning Logic for Reference.Country
-- Populate Reference.Country from original Data.Country
INSERT INTO Reference.Country (CountryName, CountryISO2, CountryISO3, SalesRegion, CountryFl
SELECT
    RTRIM(LTRIM(C.CountryName)),          -- trim country name
    UPPER(RTRIM(LTRIM(C.CountryISO2))),   -- trim and upper-case ISO2 code
    UPPER(RTRIM(LTRIM(C.CountryISO3))),   -- trim and upper-case ISO3 code
    RTRIM(LTRIM(C.SalesRegion)),          -- trim region name
    C.CountryFlag,
    C.FlagFileName,
    C.FlagFileType
FROM PrestigeCars.Data.Country AS C;
```

```sql
-- Insert Cleaning Logic for Reference.Color
-- Populate Reference.Color lookup from distinct Stock colors in original data
INSERT INTO Reference.Color (Color)
SELECT DISTINCT UPPER(RTRIM(LTRIM(S.Color)))  -- use upper-case color name for consistency
FROM PrestigeCars.Data.Stock AS S
WHERE S.Color IS NOT NULL AND LTRIM(RTRIM(S.Color)) <> '';
```

```sql
-- Insert Cleaning Logic for Data.Make
INSERT INTO Data.Make (MakeName, CountryID)
SELECT
    RTRIM(LTRIM(M.MakeName)),              -- trim Make name
    CR.CountryID                           -- lookup CountryID by country code
FROM PrestigeCars.Data.Make AS M
LEFT JOIN Reference.Country AS CR
    ON UPPER(RTRIM(LTRIM(M.MakeCountry))) = CR.CountryISO3;
```

```sql
-- Insert Cleaning Logic for Data.Model
INSERT INTO Data.Model (MakeID, ModelName, ModelVariant, YearFirstProduced, YearLastProduced)
SELECT
    M2.MakeID,
    RTRIM(LTRIM(Old.ModelName)),
    NULLIF(RTRIM(LTRIM(Old.ModelVariant)), ''),  -- trim variant, blank to NULL
    NULLIF(RTRIM(LTRIM(Old.YearFirstProduced)), ''),  -- blank years to NULL
    NULLIF(RTRIM(LTRIM(Old.YearLastProduced)), '')
FROM PrestigeCars.Data.Model AS Old
JOIN Data.Make AS M2 ON M2.MakeID = Old.MakeID;
```

```sql
-- Insert Cleaning Logic for Data.Customer
INSERT INTO Data.Customer (CustomerID, CustomerName, Address1, Address2, Town, PostCode, CountryID, IsRes
SELECT
    UPPER(RTRIM(LTRIM(C.CustomerID))),                -- CustomerID to upper-case
    RTRIM(LTRIM(C.CustomerName)),                     -- trim CustomerName
    RTRIM(LTRIM(C.Address1)),                         -- trim Address1
    NULLIF(RTRIM(LTRIM(C.Address2)), ''),             -- trim Address2, blank to NULL
    ISNULL(NULLIF(RTRIM(LTRIM(C.Town)), ''), 'Unknown'),-- trim Town, blank to 'Unknown'
    CASE
        WHEN C.PostCode IS NULL OR LTRIM(RTRIM(C.PostCode)) = '' THEN 'UNKNOWN'
        ELSE UPPER(REPLACE(C.PostCode, ' ', ''))
    END,    -- remove spaces, upper-case; use 'UNKNOWN' if blank/null
    CR.CountryID,                                     -- lookup CountryID by country code
    ISNULL(C.IsReseller, 0),                          -- default IsReseller to 0 if NULL
    ISNULL(C.IsCreditRisk, 0)                         -- default IsCreditRisk to 0 if NULL
FROM PrestigeCars.Data.Customer AS C
LEFT JOIN Reference.Country AS CR
    ON UPPER(RTRIM(LTRIM(C.Country))) = CR.CountryISO2;
```

-V

```sql
-- Insert Cleaning Logic for Data.Stock
INSERT INTO Data.Stock (StockCode, ModelID, Cost, RepairsCost, PartsCost, TransportInCost, Isf
SELECT
    UPPER(RTRIM(LTRIM(S.StockCode))),    -- normalize StockCode to upper-case
    MD.ModelID,                          -- new ModelID (foreign key mapping)
    ISNULL(S.Cost, 0),                   -- replace NULL costs with 0
    ISNULL(S.RepairsCost, 0),
    ISNULL(S.PartsCost, 0),
    ISNULL(S.TransportInCost, 0),
    ISNULL(S.IsRHD, 0),
    CO.ColorID,                          -- lookup ColorID from color name
    NULLIF(RTRIM(LTRIM(S.BuyerComments)), ''),  -- trim comments, empty to NULL
    S.DateBought,
    S.TimeBought
FROM PrestigeCars.Data.Stock AS S
JOIN Data.Model AS MD ON MD.ModelID = S.ModelID
LEFT JOIN Reference.Color AS CO
    ON UPPER(RTRIM(LTRIM(S.Color))) = CO.Color;
```

```sql
-- Insert Cleaning Logic for Data.Sales
INSERT INTO Data.Sales (SalesID, CustomerID, InvoiceNumber, TotalSalePrice, SaleDate)
SELECT
    S.SalesID,
    S.CustomerID,                -- CustomerIDs already cleaned in Data.Customer
    UPPER(S.InvoiceNumber),      -- ensure InvoiceNumber is upper-case (if alphanumeric)
    ISNULL(S.TotalSalePrice, 0.00),
    S.SaleDate
FROM PrestigeCars.Data.Sales AS S
WHERE S.CustomerID IN (SELECT CustomerID FROM Data.Customer);  -- only include sales with vali
```

```sql
-- Insert Cleaning Logic for Data.SalesDetails
INSERT INTO Data.SalesDetails (SalesID, LineItemNumber, StockID, SalePrice, LineItemDiscount)
SELECT
    SD.SalesID,
    SD.LineItemNumber,
    UPPER(RTRIM(LTRIM(SD.StockID))),  -- ensure StockID matches upper-case StockCode in new Stock
    ISNULL(SD.SalePrice, 0.00),
    ISNULL(SD.LineItemDiscount, 0.00)
FROM PrestigeCars.Data.SalesDetails AS SD
JOIN Data.Sales AS S ON SD.SalesID = S.SalesID
JOIN Data.Stock AS ST ON UPPER(RTRIM(LTRIM(SD.StockID))) = ST.StockCode;  -- include only details w
```

```sql
-- Insert Cleaning Logic for Data.PivotTable
INSERT INTO Data.PivotTable (ColorID, [2015], [2016], [2017], [2018])
SELECT
    CO.ColorID,
    P.[2015], P.[2016], P.[2017], P.[2018]
FROM PrestigeCars.Data.PivotTable AS P
JOIN Reference.Color AS CO
    ON UPPER(RTRIM(LTRIM(P.Color))) = CO.Color;
```

```sql
-- Insert Cleaning Logic for DataTransfer.Sales2015
INSERT INTO DataTransfer.Sales2015 (MakeName, ModelName, CustomerName, CountryName, Cost, RepairsCo
SELECT
    RTRIM(LTRIM(T.MakeName)),
    RTRIM(LTRIM(T.ModelName)),
    RTRIM(LTRIM(T.CustomerName)),
    RTRIM(LTRIM(T.CountryName)),
    T.Cost,
    T.RepairsCost,
    T.PartsCost,
    T.TransportInCost,
    T.SalePrice,
    T.SaleDate
FROM PrestigeCars.DataTransfer.Sales2015 AS T;
```

```sql
-- Insert Cleaning Logic for DataTransfer.Sales2016
INSERT INTO DataTransfer.Sales2016 (MakeName, ModelName, CustomerName, CountryName, Cost, RepairsCost, PartsCost, TransportInCo
SELECT
    RTRIM(LTRIM(T.MakeName)),
    RTRIM(LTRIM(T.ModelName)),
    RTRIM(LTRIM(T.CustomerName)),
    RTRIM(LTRIM(T.CountryName)),
    T.Cost,
    T.RepairsCost,
    T.PartsCost,
    T.TransportInCost,
    T.SalePrice,
    T.SaleDate
FROM PrestigeCars.DataTransfer.Sales2016 AS T;
```

```sql
-- Insert Cleaning Logic for DataTransfer.Sales2017
INSERT INTO DataTransfer.Sales2017 (MakeName, ModelName, CustomerName, CountryName, Cost, RepairsCost, PartsCost, TransportInCo
SELECT
    RTRIM(LTRIM(T.MakeName)),
    RTRIM(LTRIM(T.ModelName)),
    RTRIM(LTRIM(T.CustomerName)),
    RTRIM(LTRIM(T.CountryName)),
    T.Cost,
    T.RepairsCost,
    T.PartsCost,
    T.TransportInCost,
    T.SalePrice,
    T.SaleDate
FROM PrestigeCars.DataTransfer.Sales2017 AS T;
```

```sql
-- Insert Cleaning Logic for DataTransfer.Sales2018
INSERT INTO DataTransfer.Sales2018 (MakeName, ModelName, CustomerName, CountryName, Cost, RepairsCost, PartsCost, TransportInCo
SELECT
    RTRIM(LTRIM(T.MakeName)),
    RTRIM(LTRIM(T.ModelName)),
    RTRIM(LTRIM(T.CustomerName)),
    RTRIM(LTRIM(T.CountryName)),
    T.Cost,
    T.RepairsCost,
    T.PartsCost,
    T.TransportInCost,
    T.SalePrice,
    T.SaleDate
FROM PrestigeCars.DataTransfer.Sales2018 AS T;
```

```sql
-- Insert Cleaning Logic for Reference.Budget
INSERT INTO Reference.Budget (BudgetValue, Year, Month, BudgetDetail, BudgetElement)
SELECT
    B.BudgetValue,
    B.Year,
    B.Month,
    RTRIM(LTRIM(B.BudgetDetail)),
    RTRIM(LTRIM(B.BudgetElement))
FROM PrestigeCars.Reference.Budget AS B;


-- Insert Cleaning Logic for Reference.Forex
INSERT INTO Reference.Forex (ExchangeDate, ISOCurrency, ExchangeRate)
SELECT F.ExchangeDate, UPPER(F.ISOCurrency), F.ExchangeRate
FROM PrestigeCars.Reference.Forex AS F;


-- Insert Cleaning Logic for Reference.MarketingCategories
INSERT INTO Reference.MarketingCategories (MakeName, MarketingType)
SELECT RTRIM(LTRIM(MC.MakeName)), RTRIM(LTRIM(MC.MarketingType))
FROM PrestigeCars.Reference.MarketingCategories AS MC;


-- Insert Cleaning Logic for Reference.MarketingInformation
INSERT INTO Reference.MarketingInformation (CUST, Country, SpendCapacity)
SELECT RTRIM(LTRIM(MI.CUST)), UPPER(RTRIM(LTRIM(MI.Country))), RTRIM(LTRIM(MI.SpendCapacity))
FROM PrestigeCars.Reference.MarketingInformation AS MI;
```

```sql
-- Insert Cleaning Logic for Reference.SalesCategory
INSERT INTO Reference.SalesCategory (LowerThreshold, UpperThreshold, CategoryDescription)
SELECT SC.LowerThreshold, SC.UpperThreshold, RTRIM(LTRIM(SC.CategoryDescription))
FROM PrestigeCars.Reference.SalesCategory AS SC;


-- Insert Cleaning Logic for Reference.Staff
INSERT INTO Reference.Staff (StaffName, ManagerID, Department)
SELECT RTRIM(LTRIM(SF.StaffName)), SF.ManagerID, RTRIM(LTRIM(SF.Department))
FROM PrestigeCars.Reference.Staff AS SF;


-- Insert Cleaning Logic for Reference.StaffHierarchy
INSERT INTO Reference.StaffHierarchy (HierarchyReference, StaffName, ManagerID, Department)
SELECT SH.HierarchyReference, RTRIM(LTRIM(SH.StaffName)), SH.ManagerID, RTRIM(LTRIM(SH.Department))
FROM PrestigeCars.Reference.StaffHierarchy AS SH;


-- Insert Cleaning Logic for Reference.YearlySales
INSERT INTO Reference.YearlySales (MakeName, ModelName, CustomerName, CountryName, Cost, RepairsCost, PartsCost, TransportInCos
SELECT
    RTRIM(LTRIM(Y.MakeName)),
    RTRIM(LTRIM(Y.ModelName)),
    RTRIM(LTRIM(Y.CustomerName)),
    RTRIM(LTRIM(Y.CountryName)),
    Y.Cost,
    Y.RepairsCost,
    Y.PartsCost,
    Y.TransportInCost,
    Y.SalePrice,
    Y.SaleDate
FROM PrestigeCars.Reference.YearlySales AS Y;
```

```sql
-- Insert Cleaning Logic for SourceData.SalesInPounds
INSERT INTO SourceData.SalesInPounds (MakeName, ModelName, VehicleCost)
SELECT RTRIM(LTRIM(SIP.MakeName)), RTRIM(LTRIM(SIP.ModelName)), RTRIM(LTRIM(SIP.VehicleCost))
FROM PrestigeCars.SourceData.SalesInPounds AS SIP;


-- Insert Cleaning Logic for SourceData.SalesText
INSERT INTO SourceData.SalesText (CountryName, MakeName, Cost, SalePrice)
SELECT RTRIM(LTRIM(ST.CountryName)), RTRIM(LTRIM(ST.MakeName)), RTRIM(LTRIM(ST.Cost)), RTRIM(LTRIM(ST.SalePrice))
FROM PrestigeCars.SourceData.SalesText AS ST;


-- (Optional) After conversion (not shown), SourceData.SalesInPounds_Cleaned could be populated
-- For now, we leave SourceData.SalesInPounds_Cleaned empty or handle via separate ETL.


-- Insert Cleaning Logic for Output.StockPrices
-- For this output, combine stock cost data with model/make names for reporting
INSERT INTO Output.StockPrices (MakeName, ModelName, Cost)
SELECT
    MK.MakeName,
    MD.ModelName,
    ST.Cost
FROM Data.Stock AS ST
JOIN Data.Model AS MD ON ST.ModelID = MD.ModelID
JOIN Data.Make  AS MK ON MD.MakeID = MK.MakeID;
```

```sql
-- ============================================
-- Author: Nayem Sarker
-- Purpose: Drop all foreign keys in the 3NF database
-- ============================================

-- Drop the Drop-FKs Procedure if it exists
DROP PROCEDURE IF EXISTS [Project2.5].[DropForeignKeysFromStarSchemaData];
GO

-- Recreate Drop-FKs Procedure
CREATE PROCEDURE [Project2.5].[DropForeignKeysFromStarSchemaData]
    @UserAuthorizationKey INT
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @ForeignKeyName VARCHAR(255);
    DECLARE @TableName NVARCHAR(255);
    DECLARE @SQL NVARCHAR(MAX);

    DECLARE ForeignKeyCursor CURSOR FOR
        SELECT fk.name AS ForeignKeyName,
               QUOTENAME(OBJECT_SCHEMA_NAME(fk.parent_object_id)) + '.' + QUOTENAME(OBJECT_NAME(fk.parent_object_id)) AS TableName
        FROM sys.foreign_keys AS fk
        INNER JOIN sys.tables AS t
            ON fk.parent_object_id = t.object_id
        WHERE OBJECT_SCHEMA_NAME(fk.parent_object_id) IN ('Data', 'Reference', 'DataTransfer', 'SourceData', 'Output');  -- match same schema filter

    OPEN ForeignKeyCursor;
    FETCH NEXT FROM ForeignKeyCursor INTO @ForeignKeyName, @TableName;

    WHILE @@FETCH_STATUS = 0
    BEGIN
        SET @SQL = 'ALTER TABLE ' + @TableName + ' DROP CONSTRAINT ' + QUOTENAME(@ForeignKeyName) + ';';
        EXEC(@SQL);

        FETCH NEXT FROM ForeignKeyCursor INTO @ForeignKeyName, @TableName;
    END

    CLOSE ForeignKeyCursor;
    DEALLOCATE ForeignKeyCursor;
```

```sql
    EXEC [Process].[usp_TrackWorkFlow]
         @WorkFlowStepDescription = 'Drop Foreign Keys.',
         @UserAuthorizationKey = @UserAuthorizationKey,
         @WorkFlowStepTableRowCount = -1;
END;
GO
```

```sql
USE PrestigeCars_3NF
GO

SET ANSI_NULLS ON;
GO
SET QUOTED_IDENTIFIER ON;
GO

-- ============================================
-- Author: Nayem Sarker
-- Purpose: Truncate all tables in key schemas after dropping FKs
-- ============================================

-- Drop the Truncate Procedure if it exists
DROP PROCEDURE IF EXISTS [Project2.5].[TruncateStarSchemaData];
GO

-- Recreate Truncate Procedure
CREATE PROCEDURE [Project2.5].[TruncateStarSchemaData]
    @UserAuthorizationKey INT
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE TableCursor CURSOR FOR
        SELECT DISTINCT '[' + TABLE_SCHEMA + '].[' + TABLE_NAME + ']' AS FullyQualifiedTableName
        FROM INFORMATION_SCHEMA.TABLES
        WHERE TABLE_TYPE = 'BASE TABLE'
          AND TABLE_SCHEMA IN ('Data', 'Reference', 'DataTransfer', 'SourceData', 'Output'); -- updated schema list

    OPEN TableCursor;

    DECLARE @TableName NVARCHAR(255);
    DECLARE @SQL NVARCHAR(MAX);

    FETCH NEXT FROM TableCursor INTO @TableName;

    WHILE @@FETCH_STATUS = 0
    BEGIN
        SET @SQL = 'TRUNCATE TABLE ' + @TableName;
        EXEC(@SQL);

        FETCH NEXT FROM TableCursor INTO @TableName;
    END

    CLOSE TableCursor;
    DEALLOCATE TableCursor;

    EXEC [Process].[usp_TrackWorkFlow]
         @WorkFlowStepDescription = 'Truncate Star Schema Data.',
         @UserAuthorizationKey = @UserAuthorizationKey,
         @WorkFlowStepTableRowCount = -1;
END;
GO
```

```sql
USE PrestigeCars_3NF
GO

/* Author: Ashly*/

DROP PROCEDURE IF EXISTS [Process].[usp_TrackWorkFlow];
GO

CREATE PROCEDURE [Process].[usp_TrackWorkFlow]
    @WorkFlowStepDescription NVARCHAR(100),
    @UserAuthorizationKey INT,
    @WorkFlowStepTableRowCount INT
AS
BEGIN
    -- Stub: In a real system, log or track workflow steps here
    PRINT 'Workflow: ' + @WorkFlowStepDescription;
END;
GO
```

Views

Combining Sales Data Across Years

```
CREATE VIEW vw_AllSales AS
SELECT * FROM DataTransfer.Sales2015
UNION ALL
SELECT * FROM DataTransfer.Sales2016
UNION ALL
SELECT * FROM DataTransfer.Sales2017
UNION ALL
SELECT * FROM DataTransfer.Sales2018;
```

This query creates a view called vw_AllSales that simply just combines all the sales data from the tables of DataTransfer schema into one view(virtual table).

Show car models, costs, and their marketing types

```
CREATE VIEW vw_StockWithMarketing AS
SELECT
    s.MakeName,
    s.ModelName,
    s.Cost,
    m.MarketingType
FROM Output.StockPrices s
LEFT JOIN Reference.MarketingCategories m
    ON s.MakeName = m.MakeName;
```

Combines car stock data(make, model, cost) with their corresponding marketing type. It uses a left join to include all stock items.

William Wang

```
CREATE VIEW vw_MonthlyBudgetSummary AS
SELECT
    Year,
    Month,
    SUM(BudgetValue) AS TotalBudget
FROM Reference.Budget
GROUP BY Year, Month;
```

Calculates the total budge for each year and moth then provide a breakdown of totals overtime.

William Wang

View for Reference.Forex table

```
CREATE VIEW vw_Forex AS
SELECT
        ExchangeDate,
        ISOCurrency,
        ExchangeRate
FROM Reference.Forex;
GO
```

Shows exchange rates(date, currency, code, and rate) from the Forex table of Reference schema. Just a simple display.

View for Marketing Categories table

```
CREATE VIEW vw_MarketingCategories
AS
SELECT
    MakeName,
    MarketingType
FROM Reference.MarketingCategories;
GO
```

Extracts the MakeName(carbrand) and MarketingType from the Reference.MarketingCategories table.

William Wang

View for Marketing Information table

```
CREATE VIEW vw_MarketingInformation
AS
SELECT
    CUST,
    Country,
    SpendCapacity
FROM Reference.MarketingInformation;
GO
```

Displays CUST(customerid), Country, and SpendCapacity from the MarketingInformation table from Reference schema.

William Wang

View for Sales Budgets table

```
CREATE VIEW vw_SalesBudgets
AS
SELECT
    BudgetArea,
    BudgetAmount,
    BudgetYear,
    DateUpdated,
    Comments,
    BudgetMonth
FROM Reference.SalesBudgets;
GO
```

Displays budget details. Area, amount, year, month, date, and comments from the SalesBudgets table of the Reference schema.

William Wang

View for Sales Category table

```
CREATE VIEW vw_SalesCategory
AS
SELECT
    LowerThreshold,
    UpperThreshold,
    CategoryDescription
FROM Reference.SalesCategory;
GO
```

Retrieves the LowerThreshold, UpperThreshold, and CategoryDescription from the SalesCategory table from the Reference schema.

William Wang

View for Staff table

```
CREATE VIEW vw_Staff
AS
SELECT
    StaffID,
    StaffName,
    ManagerID,
    Department
FROM Reference.Staff;
GO
```

Displays StaffID, StaffName, ManagerID, and Department from the Reference.Staff table.

William Wang

View for Staff Hierarchy table

```
CREATE VIEW vw_StaffHierarchy
AS
SELECT
    HierarchyReference,
    StaffID,
    StaffName,
    ManagerID,
    Department
FROM Reference.StaffHierarchy;
GO
```

Retrieves HierarchyReference, StaffID, StaffName, ManagerID, and Department from the Reference.StaffHierarchy table.

William Wang

View for Yearly Sales table

```
CREATE VIEW vw_YearlySales
AS
SELECT
    MakeName,
    ModelName,
    CustomerName,
    CountryName,
    Cost,
    RepairsCost,
    PartsCost,
    TransportInCost,
    SalePrice,
    SaleDate
FROM Reference.YearlySales;
GO
```

Extracts key sales transaction details—including vehicle make/model, customer, country, costs, sale price, and date—from the Reference.YearlySales table.

William Wang

View for Sales in Pounds table

```
CREATE VIEW vw_SalesInPounds
AS
SELECT
    MakeName,
    ModelName,
    VehicleCost
FROM SourceData.SalesInPounds;
GO
```

Displays three columns—MakeName, ModelName, and VehicleCost—from the SourceData.SalesInPounds table.

View for Sales Text table

```
CREATE VIEW vw_SalesText
AS
SELECT
    CountryName,
    MakeName,
    Cost,
    SalePrice
FROM SourceData.SalesText;
GO
```

Displays CountryName, MakeName, Cost, and SalePrice from the SourceData.SalesText table.

William Wang

INLINE TABLE VALUE FUNCTIONS (IVTFS)

WILLIAM WANG

Get all sales from 2015 to 2018

```
CREATE FUNCTION dbo.fn_GetAllSales()
RETURNS TABLE
AS
RETURN
(
    SELECT *, 2015 AS SalesYear FROM DataTransfer.Sales2015
    UNION ALL
    SELECT *, 2016 AS SalesYear FROM DataTransfer.Sales2016
    UNION ALL
    SELECT *, 2017 AS SalesYear FROM DataTransfer.Sales2017
    UNION ALL
    SELECT *, 2018 AS SalesYear FROM DataTransfer.Sales2018
);
```

Combines sales data from four yearly tables (2015–2018) into a single result set.

IVTFs are like parameterized views.

Function to filter StockPrices by Make

```
CREATE FUNCTION fn_StockPricesByMake (@MakeName NVARCHAR(100))
RETURNS TABLE
AS
RETURN (
    SELECT
        MakeName,
        ModelName,
        Cost
    FROM Output.StockPrices
    WHERE MakeName = @MakeName
);
GO
```

Returns a filtered list of vehicle stock prices (make, model, cost) based on a provided brand name (e.g., 'Toyota'). It acts like a parameterized view, allowing dynamic filtering when called in queries.

Function to get Budget entries by Year/Month

```
CREATE FUNCTION fn_BudgetByYearMonth (
    @Year INT,
    @Month TINYINT
)
RETURNS TABLE
AS
RETURN (
    SELECT
        BudgetKey,
        BudgetValue,
        Year,
        Month,
        BudgetDetail,
        BudgetElement
    FROM Reference.Budget
    WHERE Year = @Year
      AND Month = @Month
);
GO
```

Extracts budget data for a specific year and month combination from the Reference.Budget table. The function returns key budget details including value, description, and elements, filtered dynamically by the input parameters.

William Wang

Function to filter by currency code

```
CREATE FUNCTION fn_ForexByCurrency (@ISOCurrency CHAR(3))
RETURNS TABLE
AS
RETURN (
    SELECT
        ExchangeDate,
        ISOCurrency,
        ExchangeRate
    FROM Reference.Forex
    WHERE ISOCurrency = @ISOCurrency
);
GO
```

Retrieves exchange rate history for a specific 3-letter currency code (like 'USD' or 'EUR'). The function returns the date, currency code, and exchange rate filtered by the input parameter, acting like a targeted currency query tool.

William Wang

Filter by make name

```
CREATE FUNCTION fn_MarketingCategoriesByMake (@MakeName NVARCHAR(100))
RETURNS TABLE
AS
RETURN (
    SELECT
        MakeName,
        MarketingType
    FROM Reference.MarketingCategories
    WHERE MakeName = @MakeName
);
GO
```

Returns marketing classifications (MakeName, MarketingType) filtered by a specific vehicle brand parameter. Acts as a parameterized lookup for marketing categories.

Filter by country code

```
CREATE FUNCTION fn_MarketingInfoByCountry (@Country NCHAR(10))
RETURNS TABLE
AS
RETURN (
    SELECT
        CUST,
        Country,
        SpendCapacity
    FROM Reference.MarketingInformation
    WHERE Country = @Country
);
GO
```

Returns customer marketing data (CUST, Country, SpendCapacity) filtered by a specified country parameter.

William Wang

Filter SalesBudgets by Year/Month combination

```
CREATE FUNCTION fn_SalesBudgetsByYearMonth (
    @BudgetYear INT,
    @BudgetMonth TINYINT
)
RETURNS TABLE
AS
RETURN (
    SELECT
        BudgetArea,
        BudgetAmount,
        BudgetYear,
        DateUpdated,
        Comments,
```

Returns sales budget details filtered by year and month parameters.

Find SalesCategory by value threshold

```
CREATE FUNCTION fn_SalesCategoryByValue (@Value INT)
RETURNS TABLE
AS
RETURN (
    SELECT
        LowerThreshold,
        UpperThreshold,
        CategoryDescription
    FROM Reference.SalesCategory
    WHERE @Value BETWEEN LowerThreshold AND UpperThreshold
);
GO
```

Returns the sales category thresholds and description that match the specified value parameter. Identifies which predefined value range (e.g., "High", "Medium", "Low") the input falls into.

William Wang

Filter Staff by Department

```
CREATE FUNCTION fn_StaffByDepartment (@Department NVARCHAR(50))
RETURNS TABLE
AS
RETURN (
   SELECT
      StaffID,
      StaffName,
      ManagerID,
      Department
   FROM Reference.Staff
   WHERE Department = @Department
);
GO
```

Filters staff records by department parameter, returning employee IDs, names, manager IDs, and departments.

William Wang

Filter StaffHierarchy by Manager

```
CREATE FUNCTION fn_StaffHierarchyByManager (@ManagerID INT)
RETURNS TABLE
AS
RETURN (
    SELECT
        HierarchyReference,
        StaffID,
        StaffName,
        ManagerID,
        Department
    FROM Reference.StaffHierarchy
    WHERE ManagerID = @ManagerID
);
GO
```

Returns staff hierarchy records filtered by a specific manager ID parameter. Shows all employees who directly report to the given manager, including their department and hierarchy details.

William Wang

Filter YearlySales by date range

```
CREATE FUNCTION fn_YearlySalesByDateRange (
    @StartDate DATETIME,
    @EndDate DATETIME
)
RETURNS TABLE
AS
RETURN (
    SELECT
        MakeName,
        ModelName,
        CustomerName,
        CountryName,
        Cost,
        RepairsCost,
        PartsCost,
        TransportInCost,
        SalePrice,
        SaleDate
    FROM Reference.YearlySales
    WHERE SaleDate BETWEEN @StartDate AND @EndDate
);
GO
```

Filters yearly sales data between specified start and end date parameters. Returns comprehensive sales transaction details including vehicle, customer, costs, and pricing within the date range.

Filter SalesInPounds by Make/Model

```
CREATE FUNCTION fn_SalesInPoundsByMakeModel (
    @MakeName NVARCHAR(100),
    @ModelName NVARCHAR(150)
)
RETURNS TABLE
AS
RETURN (
    SELECT
        MakeName,
        ModelName,
        VehicleCost
    FROM SourceData.SalesInPounds
    WHERE MakeName = @MakeName
      AND ModelName = @ModelName
);
GO
```

Returns vehicle cost data in GBP filtered by specific make and model parameters. Provides targeted pricing lookup for individual vehicle configurations.

Filter SalesText by Country and Make

```
CREATE FUNCTION fn_SalesTextByCountryMake (
    @CountryName NVARCHAR(150),
    @MakeName NVARCHAR(100)
)
RETURNS TABLE
AS
RETURN (
    SELECT
        CountryName,
        MakeName,
        Cost,
        SalePrice
    FROM SourceData.SalesText
    WHERE CountryName = @CountryName
      AND MakeName = @MakeName
);
GO
```

Filters sales metrics (cost and sale price) by country and vehicle make parameters. Delivers market-specific pricing analysis for targeted brands.

Check In

- Have we normalized all tables into 3NF, created a comprehensive Physical Data Model, and noted the cardinality of relationships?
- Did we create appropriate UDTs and document them?
- Have we made constraints and ensured data integrity?
- Have we made views and ITVFs?
- Did we clean data from the original database and index our primary keys?
- Did we document everyone's work?

- Sabrina

THANK YOU

Normalization is important because it ensures a database is structured efficiently by minimizing data redundancy and enforcing data integrity. It organizes data into logical, related tables, which makes it easier to update, maintain, and understand. This leads to more accurate data, reduces the risk of inconsistencies, and supports scalability as the system grows. Overall, normalization is a foundational skill for designing clean, reliable, and high-performance databases.