Power Query & ETL Pipeline

# Building Production-Ready Data Integration for Australian Retail

SESSION GOALS

# What You'll Learn Today

## Learning Outcomes:

- Connect 5 different data sources to Power BI
- Clean messy Australian business data (GST, states, postcodes)
- Implement staging → clean query pattern
- Validate data quality with profiling and reconciliation
- Write M code for custom transformations

## Agenda:

1. Business scenario: FreshMarket Australia (5 min)
2. Connect data sources: CSV, Excel, SQL, SharePoint, Web (10 min)
3. Power Query transformations (15 min)
4. Australian data validation (5 min)
5. M code deep dive (3 min)
6. Validation & portfolio tips (2 min)

The Reality

# Why ETL is 80% of Analytics Work

> "80% of analytics effort is spent on data preparation" — Microsoft Research

## What Power Query Solves:

### Before Power Query

- Manual Excel copy-paste
- 15 hours/month reconciliation
- Data quality errors
- Undocumented processes
- Scattered across 5 systems

### With Power Query

- Automated refresh
- 3 minutes refresh time
- Validated transformations
- Transparent applied steps
- Single unified pipeline

## Enterprise Benefits:

### Scalability

Handle millions of rows

### Auditability

Every step documented in M code

### Reusability

Copy queries across projects

### Performance

Query folding pushes work to databases

### Collaboration

Visual interface + code = team friendly

Business Scenario

# The Data Integration Challenge

# When Your Data Lives in 5 Different Places

## Company Profile:

- **Industry:** Australian Grocery Retail
- **Scale:** 50 stores (NSW: 20, VIC: 18, QLD: 12)
- **Revenue:** $450M AUD annually
- **Challenge:** Data scattered across 5 systems

# FreshMarket's Five Data Sources

| System | Owner | Format | Update Frequency | Key Issues |
|---|---|---|---|---|
| **Operations DB** | IT | SQL Server | Real-time | Master data (clean) |
| **POS Exports** | Stores | CSV | Daily | Duplicates, mixed casing |
| **Finance Plans** | CFO | Excel | Monthly | Nulls, merged cells |
| **Campaigns** | Marketing | SharePoint | Ad-hoc | Inconsistent columns |
| **Supplier Catalog** | Procurement | Web API (JSON) | Real-time | Nested structure |

## Current Pain Points:

- ❌ 15+ hours monthly reconciliation
- ❌ GST reporting errors due to missing values
- ❌ No unified performance view
- ❌ EOFY reporting delayed 3 weeks

**Your Mission:** Build a single ETL pipeline that connects all 5 sources and delivers clean, validated data.

# Power Query Connector Decision Matrix

## Choosing the Right Connector for Each Source

| Connector | Best For | Pros | Cons | When to Use |
|---|---|---|---|---|
| **Text/CSV** | Flat files, exports | Simple, portable | No query folding | POS exports, data dumps |
| **Excel Workbook** | Finance, planning | Handles multiple sheets | Merged cells issues | Budgets, forecasts |
| **SQL Server** | Enterprise DBs | Query folding, fast | Requires database | Master data, warehouses |
| **SharePoint Folder** | Shared uploads | Combines multiple files | Authentication setup | Marketing campaigns |
| **Web** | APIs, JSON | Real-time data | Complex structure | Supplier catalogs, pricing |

---

## Key Concepts:

### Query Folding:

- Power Query pushes transformations to the database
- Filters, joins, aggregations run in SQL Server (fast)
- Custom M code breaks folding (runs in Power BI memory)
- ✅ Use SQL connector + simple transforms = folding
- ❌ Use CSV + complex transforms = no folding

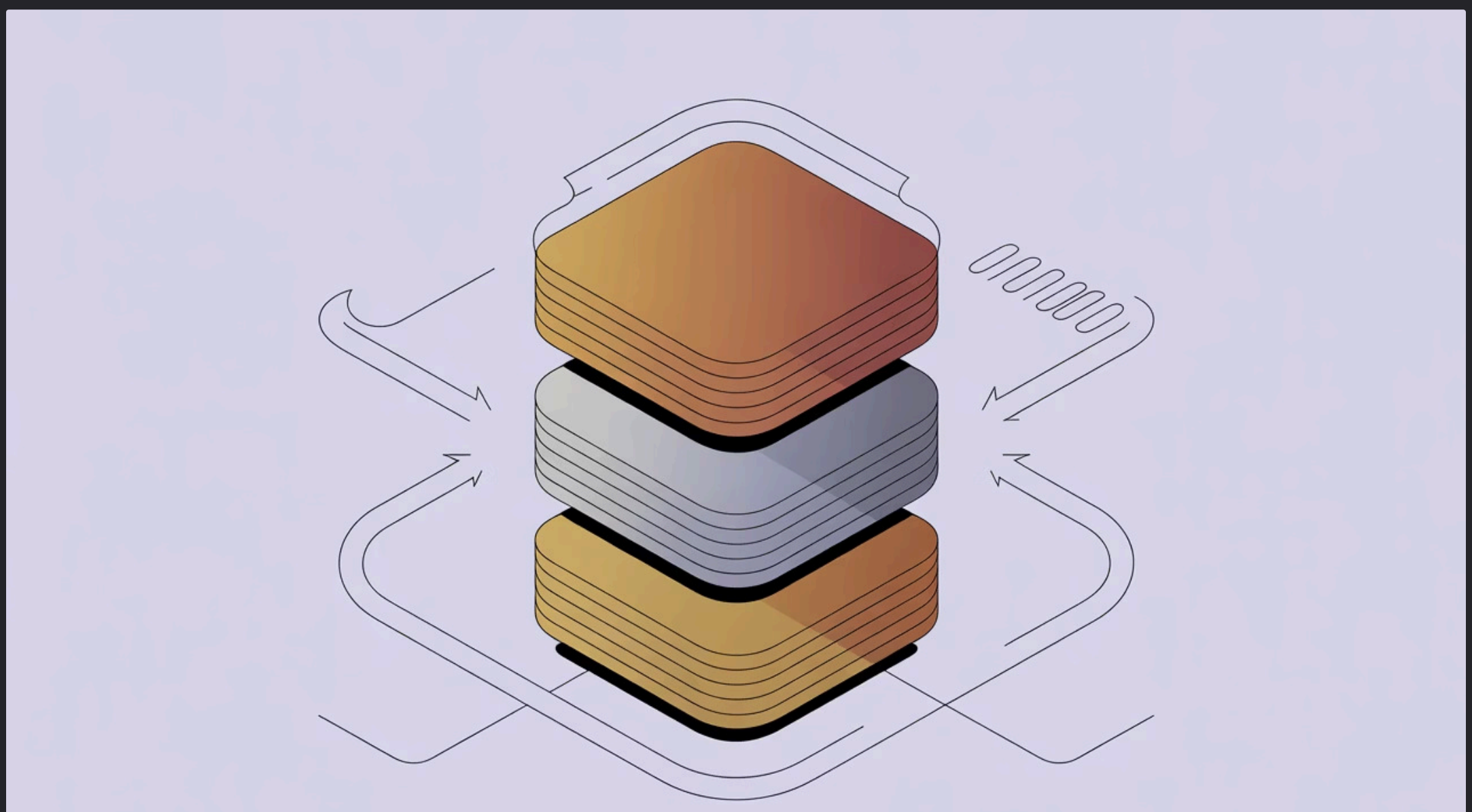
### Authentication:

- Windows Auth (SQL Server, SharePoint on-prem)
- Organizational Account (SharePoint Online, cloud services)
- API Key (Web APIs)
- Anonymous (public APIs, local files)

Architecture Pattern

# Staging → Clean Pattern Explained

# Medallion Architecture: Bronze → Silver → Gold



## The Three-Layer Pattern:

### 🥉 BRONZE (Staging)

**Raw data, minimal transformation**

**Purpose:** Preserve source data exactly as received

**Load:** DISABLED (reference only)

**Example:** stg_daily_transactions

### 🥈 SILVER (Clean)

**Validated, standardised, deduplicated**

**Purpose:** Business-ready data for analysis

**Load:** ENABLED (loads to model)

**Example:** clean_transactions

### 🥇 GOLD (Aggregated)

**Pre-aggregated, optimised for reporting**

**Purpose:** Fast dashboard queries

**Load:** ENABLED (loads to model)

**Example:** FactSales (star schema)

## Why Three Layers?

| Layer | What | Why | How to Validate |
|---|---|---|---|
| **Bronze** | Load raw data | Auditability, reprocessing | Row count = source |
| **Silver** | Clean + validate | Data quality, standardisation | Profile columns (100% valid) |
| **Gold** | Aggregate + model | Performance, business logic | Reconcile totals vs Silver |

**Naming Convention:**

- Bronze: stg_tablename (stg = staging)
- Silver: clean_tablename or DimTableName
- Gold: FactTableName, AggTableName

Data Quality

# Australian Data Validation Rules

## Handling Australian Business Data Correctly

GST, States, Postcodes, Financial Year

### Critical Australian Data Rules:

1

#### GST (Goods & Services Tax)

- **Rate:** 10% of revenue (always)
- **Formula:** GST = Revenue × 0.10
- **Validation:** if [GST] = null then [Revenue] * 0.10 else [GST]
- **Common Error:** Missing GST in 3% of POS transactions

2

#### State Codes

- **Valid Values:** NSW, VIC, QLD, WA, SA, TAS, NT, ACT (8 only)
- **Common Variations:** "New South Wales", "nsw", "Nsw", "Victoria", "vic"
- **Fix:** Text.Upper + Replace Values to standardise
- **Validation:** Column profile shows exactly 8 distinct values

3

#### Postcodes

- **Format:** 4 digits (e.g., 2000, 0800)
- **Critical:** Leading zeros (NT: 0800, ACT: 0200)
- **Data Type:** TEXT (not number)
- **Fix:** Text.PadStart([Postcode], 4, "0")
- **Common Error:** Number type drops leading zero (0800 becomes 800)

4

#### Australian Financial Year (EOFY)

- **Period:** July 1 - June 30
- **FY Assignment:** If month ≥ 7, FY = year + 1, else FY = year
- **Example:** July 1, 2023 → FY2024; June 30, 2024 → FY2024
- **EOFY Flag:** IsEOFY = 1 if date = June 30

# Power Query Transformation Workflow

## Step-by-Step Data Cleaning Process

The Standard Cleaning Workflow:

01

### Remove Duplicates ⚙️

- Identify key column (TransactionID, CustomerID, etc.)
- Table.Distinct(Source, {"TransactionID"})
- Validate: Row count decreases, keys are unique

02

### Standardise Formats 🔤

- Uppercase/lowercase consistently
- Replace variations with standard values
- Text.Upper, Table.ReplaceValue
- Validate: Column distribution shows standard values only

03

### Handle Nulls ⚠️

- Calculate missing values using formulas
- Replace nulls with business defaults
- if [Column] = null then [Default] else [Column]
- Validate: Column quality shows 0% empty

04

### Filter Invalid Data 🚫

- Remove zero/negative values
- Remove malformed dates/formats
- Table.SelectRows(Source, each [Quantity] > 0)
- Validate: All values within expected range

05

### Set Explicit Data Types 📋

- Text, Whole Number, Decimal, Currency, Date, Time
- Never rely on auto-detection
- Table.TransformColumnTypes
- Validate: Correct icon next to each column

06

### Reconcile Row Counts ✅

- Compare source rows vs. clean rows
- Document expected data loss (duplicates, errors)
- Alert if loss > 10%
- Table.RowCount(Source) vs. Table.RowCount(Clean)

Technical Deep Dive

# M Code Fundamentals

## Understanding the M Language

Power Query's Secret Weapon

> "M is the formula language behind Power Query. Every visual transformation generates M code."

### Key Concepts:

#### 1. Let-In Expression

```
let
    Source = Csv.Document(...),
    Step2 = Table.TransformColumns(Source, ...),
    Step3 = Table.SelectRows(Step2, ...)
in
    Step3  // Last step is the result
```

#### 2. Common M Functions

| Function | Purpose |
|---|---|
| Table.Distinct | Remove duplicates |
| Table.SelectRows | Filter rows |
| Table.AddColumn | Add calculated column |
| Text.Upper | Uppercase text |
| Date.Year | Extract year |

#### 3. Conditional Logic

```
// If-then-else
if [GST] = null then
    [Revenue] * 0.10
else
    [GST]

// And/or logic
each [Quantity] > 0 and [Revenue] > 0
```

#### 4. Custom Functions

```
(Revenue as number) as number =>
let
 GST = Revenue * 0.10
in
 GST
```

# Validation Checklist

## How to Validate Your ETL Pipeline

*Never Trust, Always Verify*

### Six-Point Validation Framework:

1

#### Row Count Reconciliation 📊

- Source rows vs. clean rows
- Document expected loss (duplicates, errors)
- Alert if >10% variance

```
Source: 10,000 rows
Clean: 9,200 rows
Loss: 800 rows (8%)
Breakdown: 500 duplicates + 300 invalid
```

2

#### Column Profiling 🔍

- Enable: View → Column quality/distribution/profile
- Set to: "Entire dataset" (not top 1000)
- Check each column: Valid%, Error%, Empty%
- Goal: 100% valid for all business columns

3

#### Data Type Verification ✓

- Every column has explicit type
- Text, Whole Number, Currency, Date, Time
- No "ABC123" (auto-detected)
- Validate: Correct icon next to column name

4

#### Business Rules 📋

- GST = Revenue × 0.10 (within ±$0.01)
- State in [NSW, VIC, QLD, WA, SA, TAS, NT, ACT]
- Postcode = 4 digits (text type)
- Quantity > 0, Revenue > 0
- Dates between 2022-2025

5

#### Distinct Value Counts 🎯

- State: Exactly 8 values
- LoyaltyTier: Exactly 3 values (Bronze, Silver, Gold)
- Channel: Exactly 3 values (In-Store, Online, Click & Collect)
- Use column distribution chart

6

#### Referential Integrity 🔗

- All StoreIDs exist in DimStore
- All ProductIDs exist in DimProduct
- All CustomerIDs exist in DimCustomer
- Use merge query to validate (Left Anti Join)

### Create a Validation Query:

```
let
 ValidationTable = #table(
 {"Query", "Source_Rows", "Clean_Rows", "Loss_%"},
 {
 {"Transactions", 10000, 9200, 8.0},
 {"Stores", 50, 50, 0.0},
 {"Products", 2000, 2000, 0.0}
 }
 )
in
 ValidationTable
```

# Common Mistakes & Troubleshooting

## What Goes Wrong and How to Fix It

### Top 7 Power Query Pitfalls:

1 "Column 'State' not found" Error ❌

**Cause:** Case-sensitive column names in M code

**Fix:** Use exact casing from source: [State] not [state]

**Prevention:** Copy column name from query, don't type

2 Duplicates Not Removed 🔄

**Cause:** Table.Distinct(Source) removes row-level duplicates only

**Fix:** Specify key column: Table.Distinct(Source, {"TransactionID"})

**Prevention:** Always use explicit key

3 Postcode Leading Zeros Lost 0️⃣

**Cause:** Stored as number (0800 → 800)

**Fix:** Change type to Text, use Text.PadStart([Postcode], 4, "0")

**Prevention:** Australian postcodes always TEXT type

4 GST Calculation Fails on Nulls 💰

**Cause:** [Revenue] * 0.10 errors when Revenue is null

**Fix:** if [Revenue] = null then null else [Revenue] * 0.10

**Prevention:** Always handle nulls in calculations

5 Slow Query Performance 🐌

**Cause:** Not using query folding; profiling entire dataset

**Fix:** Keep SQL queries simple; disable profiling after dev

**Prevention:** Check Query Diagnostics for folding indicators

6 Cannot Delete Staging Query 🚫

**Cause:** Clean queries reference it

**Fix:** Don't delete—disable load instead

**Prevention:** Staging queries always have load disabled

7 Date Format Confusion 📅

**Cause:** DD/MM/YYYY (AU) vs MM/DD/YYYY (US)

**Fix:** Date.FromText([Date], [Culture="en-AU"])

**Prevention:** Explicit date parsing with Australian culture