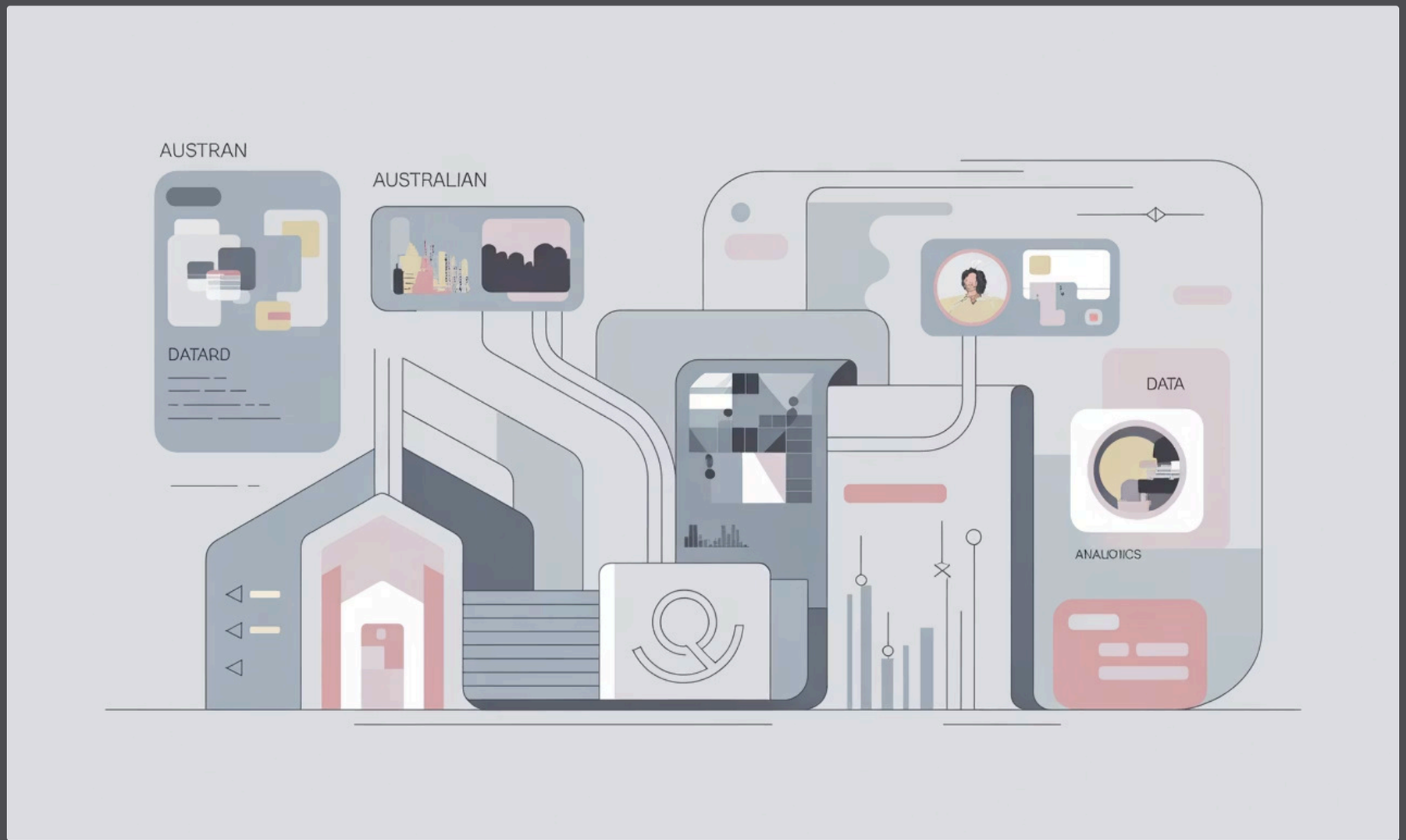
VIDEO 1: POWER QUERY & ETL PIPELINE

Complete Playbook - FreshMarket Store Operations



Duration: 35-40 minutes

Industry: Australian Grocery Retail

Focus: Multi-source ETL, Australian data validation, Power Query mastery

TABLE OF CONTENTS

1. SESSION OVERVIEW & LEARNING OUTCOMES

Duration: 35-40 minutes

Learning Outcomes

By the end of this session, you will be able to:

- ✅ Connect Power BI to 5 different data sources (CSV, Excel, SQL Server, SharePoint, Web API)
- ✅ Implement staging → clean query pattern following medallion architecture principles
- ✅ Clean messy Australian retail data (GST calculations, ABN formatting, state standardisation, postcode validation)
- ✅ Handle data quality issues (duplicates, nulls, type mismatches, casing inconsistencies)
- ✅ Use Power Query transformations (merge, append, group, pivot, unpivot)
- ✅ Write and understand M Code for custom transformations
- ✅ Validate data quality with profiling, row counts, and reconciliation
- ✅ Document ETL processes for enterprise environments

Why This Matters

Power Query is the foundation of every Power BI solution. According to Microsoft, **80% of analytics effort is spent on data preparation**. Mastering ETL ensures:

Data Quality

Clean, validated data prevents incorrect business decisions

Scalability

Repeatable processes handle growing data volumes

Performance

Efficient transformations reduce refresh times from hours to minutes

Auditability

Documented steps support compliance and troubleshooting

Collaboration

Modular queries enable team development

Enterprise Impact in Australia

In Australian enterprises, proper ETL is critical for:

- ✅ **EOFY (End of Financial Year) reporting accuracy** - June 30 deadlines
- ✅ **GST compliance and reconciliation** - 10% tax calculation validation
- ✅ **Multi-state operational analysis** - NSW, VIC, QLD performance comparison
- ✅ **Regulatory reporting** - ASIC, APRA, Privacy Act requirements

2. BUSINESS SCENARIO

Company Profile: FreshMarket Australia

Industry: Grocery Retail
Scale: 50 stores across NSW, VIC, QLD
Annual Revenue: $450M AUD
Employees: 2,500
Financial Year: July 1 - June 30 (Australian FY)

Store Distribution:

- **NSW:** 20 stores (Sydney metro: 12, Regional: 8)
- **VIC:** 18 stores (Melbourne metro: 11, Regional: 7)
- **QLD:** 12 stores (Brisbane metro: 7, Regional: 5)

Business Model:

- Traditional in-store grocery retail (70% of revenue)
- Growing online delivery platform (20% of revenue)
- Click & Collect hybrid service (10% of revenue)
- Product mix: Fresh produce, dairy, meat, packaged goods
- Loyalty programme: **FreshRewards** (Bronze/Silver/Gold tiers)

The Data Integration Challenge

FreshMarket's data is **scattered across 5 different systems**:

System	Owner	Format	Update Frequency	Key Issues
Operations Database	IT Department	SQL Server	Real-time	Source of truth for master data
POS Systems	Store Operations	CSV exports	Daily	Mixed casing, duplicates, null GST
Finance Planning	CFO Office	Excel workbooks	Monthly	Merged cells, manual formulas, nulls
Marketing Campaigns	Marketing Team	SharePoint uploads	Ad-hoc	Inconsistent column names, multiple files
Supplier Catalogue	Procurement	Web API (JSON)	Real-time	Nested JSON structure

Current Pain Points

- ❌ **Manual reconciliation takes 15+ hours per month**
- ❌ **Data quality issues cause incorrect GST reporting**
- ❌ **No unified view of performance across channels**
- ❌ **State-level analysis requires manual Excel pivots**
- ❌ **EOFY reporting delayed by 3 weeks due to data cleanup**

Your Mission

Build a **robust ETL pipeline** that:

- ✅ Connects all 5 data sources securely
- ✅ Cleans and standardises Australian business data
- ✅ Creates reusable, documented transformation patterns
- ✅ Validates data quality at each stage
- ✅ Enables executive reporting and operational dashboards

Success Criteria

Metric	Target	How to Measure
Data completeness	>95% of records loaded	Row count reconciliation
Duplicate removal	0 duplicate TransactionIDs	DISTINCTCOUNT validation
GST accuracy	100% of transactions have valid GST	NULL count = 0
State standardisation	Only 8 valid state values	Column profiling
Reconciliation accuracy	±2% variance allowed	Source vs Clean comparison
Refresh performance	<5 minutes total ETL time	Performance analyser

3. ENVIRONMENT SETUP

Software Requirements

Essential:

- ✅ Power BI Desktop (December 2024 or newer - version 2.136+)
- ✅ Python 3.8+ (for dataset generation)
- ✅ SQL Server 2019+ (Express or Developer edition - free)
- ✅ SQL Server Management Studio (SSMS)

Hardware:

- 8GB+ RAM (16GB preferred for large datasets)
- 10GB free disk space
- Modern CPU (i5 or equivalent)

Recommended:

- VS Code or Notepad++ (for viewing M Code)
- Dual monitors (for following along with playbook)

Step 1: Verify Power BI Desktop Installation

What to do:

1. Launch Power BI Desktop
2. Click **File** → **Options and settings** → **Options**
3. Under **Data Load** section:
   - ✅ Check **Enable Data Profiling**
   - ✅ Check **Column Quality**
   - ✅ Check **Column Distribution**
   - ✅ Check **Column Profile**
4. Under **Power Query Editor** section:
   - ✅ Check **Always allow parameterisation in data source dialogues**
   - ⬜ Uncheck **Require user approval for new native database queries** (for training only)
5. Click **OK**

> **Why it matters:** Data profiling displays quality metrics immediately in Power Query. You'll see null percentages, value distributions, and errors as you transform data. The approval setting speeds up development but should be re-enabled in production for security.

How to validate:

Click **Help** → **About**. Verify:

- Version: December 2024 (2.136) or newer
- 64-bit version (recommended for performance)

Step 2: Create Project Folder Structure

What to do:

Open File Explorer and create this exact folder structure:

```
C:\PowerBI-Training\
└── FreshMarket-ETL\
    ├── data\
    │   ├── csv\
    │   ├── excel\
    │   ├── sql\
    │   └── sharepoint\
    ├── scripts\
    └── pbix\
```

> **Why it matters:** Organised folders prevent file path errors during data connections. This structure mirrors enterprise data lake patterns (Bronze/Silver/Gold layers). Version control systems (Git) work best with consistent folder hierarchies.

How to validate:

Run this PowerShell command to verify:

```
Get-ChildItem -Path "C:\PowerBI-Training\FreshMarket-ETL" -Recurse -Directory | Select-Object FullName
```

You should see all 4 data subfolders, scripts, and pbix folders listed.

Step 3: Install Python Libraries

What to do:

Open Command Prompt (Windows + R, type cmd, press Enter) and run:

```
# Verify Python is installed
python --version
# Expected output: Python 3.8.x or higher
```

If Python is not installed, download from **python.org** and install with "Add to PATH" option checked.

Install required libraries:

```
pip install pandas numpy faker openpyxl
```

> **Why it matters:** The dataset generator script requires these libraries:
>
> - **pandas:** Data manipulation and CSV/Excel creation
> - **numpy:** Random number generation for realistic data
> - **faker:** Australian names, addresses, companies
> - **openpyxl:** Excel file creation with multiple sheets

How to validate:

```
python -c "import pandas, numpy, faker, openpyxl; print('
```

4. DATASET GENERATION

What You'll Generate

The Python script creates **realistic Australian retail data** with intentional quality issues:

File	Records	Purpose	Quality Issues
daily_transactions.csv	10,000	POS system export	Mixed case states, duplicates, invalid dates
finance_budgets_FY2024.xlsx	60	Budget planning	Nulls in forecast columns, merged cells
dim_store.csv	50	Store master data	Clean reference dimension
dim_product.csv	2,000	Product catalogue	Clean reference dimension
dim_customer.csv	20,000	Customer loyalty	Clean reference dimension
dim_channel.csv	3	Sales channels	Clean reference dimension
dim_date.csv	1,461	Date table with Australian FY	Clean reference dimension
fact_sales.csv	52,500	All transactions	5% duplicates, 3% null GST, outliers
marketing_*.csv	6 files × 500 rows	Campaign data	Inconsistent column casing
supplier_catalog_api.json	100	Web API simulation	Nested JSON structure

Run the Dataset Generator

What to do:

The dataset generator script has already been provided in the previous section. Save it as:

```
C:\PowerBI-Training\FreshMarket-ETL\scripts\datasets.py
```

Then run:

```
cd C:\PowerBI-Training\FreshMarket-ETL\scripts
python datasets.py
```

Expected Output:

```
======================================================================
FRESHMARKET AUSTRALIA - DATASET GENERATOR
Power BI ETL Training - Video 1
======================================================================
```

5. POWER BI PROJECT SETUP

Create New Power BI Project

What to do:

1. Launch Power BI Desktop
2. Close the splash screen
3. Click **File** → **Save As**
4. Navigate to: C:\PowerBI-Training\FreshMarket-ETL\pbix\
5. Filename: FreshMarket_ETL_Pipeline.pbix
6. Click **Save**

> **Why it matters:** Saving immediately establishes auto-recovery. Power BI saves recovery info every 10 minutes by default.

How to validate:

- File should appear in C:\PowerBI-Training\FreshMarket-ETL\pbix\
- Title bar shows: FreshMarket_ETL_Pipeline - Power BI Desktop

Configure Power Query Options

What to do:

1. Click **Home** ribbon → **Transform data** dropdown → **Query Settings**
2. Under **Data Load**:
   - Type detection: **Do not detect data types**
   - ✅ Check **Enable load to data model** (already on by default)
3. Under **Power Query Editor**:
   - Default query load: **Specify custom default load settings**
4. Click **OK**

> **Why it matters:** Disabling automatic type detection gives you full control. You'll explicitly set types in transformations, preventing unexpected conversions.

How to validate:

Create a test query:

1. Click **Get data** → **Blank Query**
2. In the formula bar, enter: = {1, 2, 3}
3. Power Query should show a list without auto-converting to a table
4. Delete this test query

6. SOURCE INTEGRATION - CSV

Connect to Daily Transactions CSV

What to do:

1. In Power BI Desktop, click **Home** → **Get data** → **Text/CSV**
2. Navigate to: C:\PowerBI-Training\FreshMarket-ETL\data\csv\daily_transactions.csv
3. Click **Open**

The **Preview** window appears showing the first 200 rows.

1. **DO NOT click "Load"** - instead, click **Transform Data**

This opens Power Query Editor.

> **Why it matters:** The CSV contains messy data (mixed case states, duplicates). Clicking "Transform Data" sends it to Power Query for cleaning before loading into the model.

How to validate:

Power Query Editor should now be open showing:

- Query name: daily_transactions (left pane)
- Preview of data with 13 columns
- Applied Steps pane on the right showing:
  - Source
  - Promoted Headers
  - Changed Type (auto-detected)

Create Staging Query

What to do:

In Power Query Editor:

1. Right-click the daily_transactions query in the left pane
2. Select **Rename**
3. New name: stg_daily_transactions
4. Press Enter

> **Why it matters:** The stg_ prefix indicates a **staging query** - raw data with minimal transformations. This follows medallion architecture:
>
> - **Bronze (Staging):** Raw data, minimal changes
> - **Silver (Clean):** Cleaned, validated data
> - **Gold (Aggregated):** Business-ready datasets

How to validate:

Query name in left pane should now show: stg_daily_transactions

Inspect Data Quality

What to do:

1. Click **View** ribbon → check **Column quality**
2. Click **View** ribbon → check **Column distribution**
3. Click **View** ribbon → check **Column profile**
4. At the bottom of the preview, change **Column profiling based on:** from "Top 1000 rows" to **"Entire dataset"**
5. Click on the **State** column header

Power Query analyses the entire dataset.

> **Why it matters:** Column profiling reveals data quality issues immediately:
>
> - **Valid %:** Percentage of non-null, non-error values
> - **Error %:** Rows with conversion errors
> - **Empty %:** Null or blank values
> - **Distinct count:** Number of unique values
> - **Unique count:** Values appearing only once

How to validate:

Under the **State** column header, you should see:

- **Valid:** ~98%
- **Error:** 0%
- **Empty:** ~2%

In the **Column distribution** chart below, you'll see multiple variations:

- NSW, nsw, Nsw, new south wales (incorrect casing/format)
- VIC, vic, Vic, victoria
- QLD, qld, Qld, queensland

This confirms the data needs cleaning.

Record Row Count

What to do:

1. In the bottom-left of Power Query, note the row count
2. Open Notepad and create a validation log:

```
=== ETL VALIDATION LOG ===

Source: daily_transactions.csv
Date: 2024-12-30
Row Count (Source): 10,000

Expected Issues:
- Mixed case State values
- Duplicate TransactionIDs (~5%)
- Null GST values (~3%)
- Invalid dates (~2%)
```

> **Why it matters:** Row count reconciliation is critical. You'll compare this number after each transformation to detect unexpected data loss.

How to validate:

Power Query status bar should show: 10000 rows loaded

Disable Load for Staging Query

What to do:

1. Right-click stg_daily_transactions query
2. Uncheck **Enable load**

A small icon appears next to the query name indicating load is disabled.

> **Why it matters:** Staging queries are reference-only. You don't want them in the final data model - only clean queries should load. This reduces model size and confusion.

How to validate:

- Italicised query name with disabled icon
- When you close Power Query later, this query won't appear in the Fields pane

7. SOURCE INTEGRATION - EXCEL

Connect to Finance Budget Workbook

What to do:

1. In Power Query Editor, click **Home** → **New Source** → **Excel workbook**
2. Navigate to: C:\PowerBI-Training\FreshMarket-ETL\data\excel\finance_budgets_FY2024.xlsx
3. Click **Open**

The **Navigator** window appears showing all sheets and tables.

1. Check the boxes for:
   - ✅ **Budget** sheet
   - ✅ **Notes** sheet
2. Click **Transform Data** (NOT "Load")

> **Why it matters:** Excel workbooks often contain multiple sheets. The Navigator lets you select specific sheets or named ranges. Choosing "Transform Data" allows you to clean before loading.

How to validate:

Power Query Editor now shows **two new queries**:

- Budget
- Notes

Both appear in the Queries pane on the left.

Inspect Budget Sheet

What to do:

1. Click on the **Budget** query
2. Review the data preview
3. Check column headers: Month, ProductID, BudgetRevenue, BudgetUnits, ForecastRevenue, ForecastUnits
4. Enable column profiling (if not already on): **View** → **Column quality/distribution/profile**
5. Set profiling to **Entire dataset**
6. Click on **ForecastRevenue** column

> **Why it matters:** Excel files often have quality issues:
>
> - Merged cells (become nulls in Power Query)
> - Manual formulas not evaluated
> - Inconsistent data types
> - Extra header rows

How to validate:

Under **ForecastRevenue** column header:

- **Valid:** ~90%
- **Empty:** ~10%

This shows that ~10% of forecast values are missing (nulls), which is intentional in the dataset.

Rename to Staging Pattern

What to do:

1. Right-click **Budget** query → Rename → stg_budget
2. Right-click **Notes** query → Rename → stg_notes
3. Right-click both queries and **uncheck "Enable load"**

> **Why it matters:** Consistent naming helps team members understand query purpose. Staging queries should never load to the model.

How to validate:

Queries pane shows:

- stg_budget (italicised, load disabled)
- stg_notes (italicised, load disabled)

Record Validation Data

What to do:

Add to your validation log in Notepad:

```
Source: finance_budgets_FY2024.xlsx

Sheet: Budget
Row Count (Source): 60
Nulls in ForecastRevenue: ~6 rows (10%)
Nulls in ForecastUnits: ~6 rows (10%)

Sheet: Notes
Row Count: 5 (documentation only)
```

How to validate:

Power Query status bar for stg_budget should show: 60 rows loaded

8. SOURCE INTEGRATION - SQL SERVER

Connect to SQL Server Database

What to do:

1. In Power Query Editor, click **Home** → **New Source** → **SQL Server**
2. SQL Server database window appears:
   - **Server:** (local) or localhost or your instance name
   - **Database (optional):** FreshMarket_Source
   - **Data Connectivity mode:** Import
3. Click **OK**

If prompted for credentials:

1. Select **Windows** authentication
2. Click **Connect**

The **Navigator** window appears showing all tables in the database.

> **Why it matters:** SQL Server is the most common enterprise database. The Import mode loads data into Power BI's in-memory engine for fast queries. DirectQuery mode (not used here) queries the database in real-time.

How to validate:

Navigator should display 6 tables:

- dim_channel
- dim_customer
- dim_date
- dim_product
- dim_store
- fact_sales

Select Dimension Tables

What to do:

In the Navigator window, check the boxes for these 5 tables:

- ✅ dim_channel
- ✅ dim_customer
- ✅ dim_date
- ✅ dim_product
- ✅ dim_store

DO NOT select fact_sales (we'll use the CSV version which has quality issues for practice)

Click **Transform Data**

> **Why it matters:** Dimension tables (Product, Customer, Store, etc.) are typically clean reference data. Fact tables (Sales transactions) often have quality issues and come from different systems.

How to validate:

Power Query Editor now shows **5 new queries**:

- dim_channel
- dim_customer
- dim_date
- dim_product
- dim_store

Rename with Staging Prefix

What to do:

Rename each SQL query with stg_ prefix:

1. dim_channel → stg_dim_channel
2. dim_customer → stg_dim_customer
3. dim_date → stg_dim_date
4. dim_product → stg_dim_product
5. dim_store → stg_dim_store

Then disable load for all 5 queries:

- Right-click each → Uncheck **Enable load**

> **Why it matters:** Consistent naming pattern makes it clear these are raw staging queries. Later you'll create clean queries that reference these.

How to validate:

Queries pane shows all 5 queries with stg_ prefix, italicised (load disabled):

- stg_dim_channel
- stg_dim_customer
- stg_dim_date
- stg_dim_product
- stg_dim_store

Record Row Counts

What to do:

For each dimension, note the row count and add to validation log:

```
Source: SQL Server (FreshMarket_Source)

stg_dim_channel: 3 rows
stg_dim_customer: 20,000 rows
stg_dim_date: 1,461 rows (2022-2025)
stg_dim_product: 2,000 rows
stg_dim_store: 50 rows
```

How to validate:

Click each query and check the status bar:

- stg_dim_channel: **3 rows loaded**
- stg_dim_customer: **20000 rows loaded**
- stg_dim_date: **1461 rows loaded**
- stg_dim_product: **2000 rows loaded**
- stg_dim_store: 50 rows loaded

9. SOURCE INTEGRATION - SHAREPOINT FOLDER

Connect to SharePoint Folder (Simulated)

What to do:

Since we're simulating SharePoint with local files:

1. In Power Query Editor, click **Home** → **New Source** → **Folder**
2. Folder path: C:\PowerBI-Training\FreshMarket-ETL\data\sharepoint
3. Click **OK**

The preview shows a list of all files in the folder with columns:

- Content (binary)
- Name
- Extension
- Date accessed
- Date modified
- Date created
- Attributes
- Folder Path

1. Click **Combine** → **Combine & Transform Data**

The **Combine Files** dialogue appears.

1. Select the first CSV file as the sample
2. Click **OK**

Power Query automatically creates:

- A **Parameter query** (for the folder path)
- A **Sample file query** (showing structure)
- A **Combined query** (all files merged)

> **Why it matters:** SharePoint Folder connector is used in enterprises when marketing/operations teams upload files to a shared location. The "Combine Files" feature automatically merges all files with the same structure - critical for campaign data uploaded monthly/quarterly.

How to validate:

Queries pane now shows:

- A query with the folder name
- Helper queries under a collapsed section
- Combined query showing all marketing campaign data

Inspect Combined Marketing Data

What to do:

1. Click on the main combined query (not the helper queries)
2. Review columns - you should see columns from the marketing CSV files:
   - Source.Name (filename)
   - campaign_date
   - Campaign_Name
   - PRODUCT_ID (note: uppercase)
   - spend
   - impressions
   - Clicks (note: mixed case)
3. Enable column profiling
4. Click on **PRODUCT_ID** column

> **Why it matters:** Notice the column name inconsistencies:
>
> - campaign_date (lowercase)
> - Campaign_Name (Title Case)
> - PRODUCT_ID (UPPERCASE)
> - Clicks (Title Case)
>
> This simulates real SharePoint uploads where different users upload files with inconsistent formatting.

How to validate:

Column distribution shows:

- Distinct values: ~100 (unique products)
- Data type: Text (needs cleaning to match ProductID format)

Total rows should be: **548 rows** (6 campaigns × ~90 days each)

Rename Combined Query

What to do:

1. Rename the combined query: stg_marketing_campaigns
2. Disable load: Right-click → Uncheck **Enable load**

> **Why it matters:** This staging query needs column name standardisation and ProductID format matching before loading.

How to validate:

Query appears as: stg_marketing_campaigns (italicised, load disabled)

Record Validation Data

Add to log:

```
Source: SharePoint Folder Simulation
Path: C:\PowerBI-Training\FreshMarket-ETL\data\sharepoint
Files Combined: 6 CSV files
Total Rows: 548

Quality Issues:
- Inconsistent column casing
- PRODUCT_ID format may not match dim_product
```

10. SOURCE INTEGRATION - WEB API

Connect to Web API (JSON Simulation)

What to do:

1. In Power Query Editor, click **Home** → **New Source** → **Web**
2. URL: file:///C:/PowerBI-Training/FreshMarket-ETL/data/csv/supplier_catalog_api.json
3. Click **OK**

Power Query displays the JSON content as a record.

1. Click on **Record** in the preview
2. Power Query shows fields: status, timestamp, data
3. Click **Convert Into Table** (ribbon button)
4. Click **Expand** button (double arrows) next to the column header
5. Check all fields, uncheck "Use original column name as prefix"
6. Click **OK**

> **Why it matters:** Web APIs typically return JSON data. Power Query can parse JSON structures and convert them to tabular format. This simulates connecting to a supplier's product catalogue API that provides real-time pricing.

How to validate:

After expansion, you should see columns:

- status
- timestamp
- data (still a list of records)

Now expand the **data** column:

1. Click **Expand** on the data column
2. Select: product_id, supplier_name, current_price, stock_status, last_updated
3. Uncheck "Use original column name as prefix"
4. Click **OK**

Final columns:

- status
- timestamp
- product_id
- supplier_name
- current_price
- stock_status
- last_updated

How to validate:

Row count should show: **100 rows loaded**

Rename Web Query

What to do:

1. Rename query: stg_supplier_catalog
2. Disable load: Right-click → Uncheck **Enable load**

Add to validation log:

```
Source: Web API (JSON simulation)
URL: file:///C:/PowerBI-Training/FreshMarket-ETL/data/csv/supplier_catalog_api.json
Rows: 100
Columns: 7
```

11. POWER QUERY TRANSFORMATIONS

Now that all sources are connected, you'll create **clean queries** that reference the staging queries and apply transformations.

Clean Daily Transactions CSV

What to do:

1. Right-click stg_daily_transactions → **Reference**
2. A new query appears named stg_daily_transactions (2)
3. Rename it: clean_transactions
4. Enable load: Right-click → Check **Enable load**

Now apply transformations step-by-step:

Step 1: Remove Duplicate TransactionIDs

What to do:

1. Click on the **TransactionID** column header
2. Click **Home** ribbon → **Remove Rows** → **Remove Duplicates**

> **Why it matters:** The source data has ~5% duplicates (same TransactionID appearing multiple times). This would cause incorrect revenue totals.

How to validate:

- Applied Steps pane shows: "Removed Duplicates"
- Row count changes from **10,000** to **~9,500** (after removing ~500 duplicates)

Step 2: Standardise State Column

What to do:

1. Click on the **State** column
2. Click **Transform** ribbon → **Format** → **UPPERCASE**

This converts all state values to uppercase (NSW, VIC, QLD, etc.)

1. Next, replace full state names with abbreviations:
   - Click **Transform** → **Replace Values**
   - Value to Find: NEW SOUTH WALES
   - Replace With: NSW
   - Click **OK**
2. Repeat for other full names:
   - VICTORIA → VIC
   - QUEENSLAND → QLD
   - WESTERN AUSTRALIA → WA
   - SOUTH AUSTRALIA → SA
   - TASMANIA → TAS
   - NORTHERN TERRITORY → NT
   - AUSTRALIAN CAPITAL TERRITORY → ACT

> **Why it matters:** Consistent state codes enable proper joining with dimension tables and state-level filtering in reports.

How to validate:

- Click on **State** column
- Column distribution shows only 3 distinct values: NSW, VIC, QLD
- No lowercase or full name variations

Step 3: Handle Null GST Values

What to do:

1. Click on **GST** column
2. Click **Transform** ribbon → **Replace Values**
   - Value to Find: null
   - Replace With: (leave empty)
   - Click **Advanced options** → Check **Replace using special characters**
   - Select **null** from dropdown
3. Click **OK**

This identifies null values. Now calculate GST where missing:

1. Click **Add Column** ribbon → **Custom Column**
2. New column name: GST_Calculated
3. Formula: if [GST] = null then [Revenue] * 0.10 else [GST]
4. Click **OK**
5. Delete the original **GST** column
6. Rename **GST_Calculated** → **GST**

> **Why it matters:** In Australia, GST (Goods and Services Tax) is **10% of revenue**. Missing GST values would cause tax reporting errors. This formula recalculates GST where missing.

How to validate:

- Click on **GST** column
- Column quality shows: **100% Valid, 0% Empty**
- Spot-check: Select a row, verify GST = Revenue × 0.10

Step 4: Filter Out Invalid Quantities

What to do:

1. Click on **Quantity** column dropdown (filter arrow)
2. Uncheck values: 0, -1, -2 (negative/zero quantities)
3. Click **OK**

> **Why it matters:** Zero or negative quantities indicate returns or data errors. These should be filtered out or handled in a separate returns table.

How to validate:

- Applied Steps shows: "Filtered Rows"
- Row count decreases by ~85 rows
- Quantity column shows only positive values (1-10)

Step 5: Handle Invalid Dates

What to do:

1. Click on **TransactionDate** column
2. Click **Transform** ribbon → **Data Type** → **Date**

Power Query will show errors for invalid dates (like "2024-13-45").

1. Click on the error icon in the column header
2. Select **Remove Errors**

> **Why it matters:** Invalid dates (month 13, day 45) cannot be used for time-series analysis. Removing errors prevents conversion failures.

How to validate:

- Applied Steps shows: "Removed Errors"
- TransactionDate column shows no error values
- Data type is now **Date**

Step 6: Set Explicit Data Types

What to do:

For each column, set the correct data type:

1. **TransactionID:** Text	8. **UnitPrice:** Decimal Number (Currency)
2. **TransactionDate:** Date	9. **Discount:** Decimal Number
3. **StoreID:** Text	10. **Revenue:** Decimal Number (Currency)
4. **ProductID:** Text	11. **Cost:** Decimal Number (Currency)
5. **CustomerID:** Text	12. **GST:** Decimal Number (Currency)
6. **ChannelID:** Text	13. **TransactionTime:** Time
7. **Quantity:** Whole Number	

To set data type:

- Click column header → **Transform** → **Data Type** → Select type

> **Why it matters:** Explicit data types prevent unexpected conversions and ensure correct aggregation (SUM only works on numbers).

How to validate:

Check the data type icon next to each column name:

- **ABC** = Text
- **123** = Whole Number
- **$** = Currency
- 📅 = Date
- 🕐 = Time

All 13 columns should have the correct icon.

Final Validation for clean_transactions

What to do:

1. Review **Applied Steps** pane (should have ~10-12 steps)
2. Check final row count: **~9,200 rows** (started with 10,000, removed duplicates/errors)
3. Verify column quality: All columns should show **100% Valid**

Add to validation log:

```
=== CLEAN TRANSACTIONS ===

Source: stg_daily_transactions
Starting Rows: 10,000
Duplicates Removed: ~500
Invalid Dates Removed: ~200
Invalid Quantities Removed: ~100
Final Rows: ~9,200

Transformations Applied:
✓ Remove duplicates (TransactionID)
✓ Standardise State (UPPERCASE + abbreviations)
✓ Calculate missing GST (Revenue * 0.10)
✓ Filter zero/negative Quantity
✓ Remove invalid dates
✓ Set explicit data types (13 columns)

Data Quality Check:
✓ All columns 100% Valid
✓ State values: NSW, VIC, QLD only
✓ GST = Revenue * 0.10 for all rows
✓ All quantities positive (1-10)
✓ All dates valid (2022-2024 range)
```

12. AUSTRALIAN DATA VALIDATION

Create Clean Dimension Queries

For each staging dimension, create clean queries:

Clean DimStore

What to do:

1. Right-click stg_dim_store → **Reference**
2. Rename: DimStore
3. Enable load

Apply transformations:

1. **State column:** Already clean (NSW, VIC, QLD)
2. **Postcode column:** Ensure 4-digit format
   - Select **Postcode** column
   - **Transform** → **Format** → **Add Prefix** (if needed to make 4 digits)
   - Or use **Custom Column:** Text.PadStart([Postcode], 4, "0")
3. Verify data types:
   - StoreID: Text
   - StoreName: Text
   - State: Text
   - Postcode: Text (not number - leading zeros matter)
   - Region: Text
   - ManagerName: Text
   - OpenDate: Date
   - StoreSize: Text

> **Why it matters:** Australian postcodes are **4 digits** and can start with 0 (e.g., 0800 for NT). Storing as Text preserves leading zeros.

How to validate:

- Row count: **50 rows** (unchanged)
- Postcode examples: 2000, 2011, 3000, 4000 (all 4 digits)
- State distinct values: 3 (NSW, VIC, QLD)

Clean DimProduct

What to do:

1. Right-click stg_dim_product → **Reference**
2. Rename: DimProduct
3. Enable load
4. Verify data types (already clean):
   - ProductID: Text
   - ProductName: Text
   - Category: Text
   - Subcategory: Text
   - Supplier: Text
   - UnitCost: Currency
   - Brand: Text
   - PackSize: Text

How to validate:

- Row count: **2,000 rows** (unchanged)
- No null values in any column

Clean DimCustomer

What to do:

1. Right-click stg_dim_customer → **Reference**
2. Rename: DimCustomer
3. Enable load
4. Verify columns:
   - CustomerID: Text
   - CustomerName: Text
   - Email: Text
   - State: Text (NSW, VIC, QLD)
   - Postcode: Text (4 digits)
   - LoyaltyTier: Text (Bronze, Silver, Gold)
   - JoinDate: Date
   - DateOfBirth: Date

How to validate:

- Row count: **20,000 rows**
- LoyaltyTier distinct values: 3 (Bronze, Silver, Gold)
- State distinct values: 3 (NSW, VIC, QLD)

Clean DimChannel

What to do:

1. Right-click stg_dim_channel → **Reference**
2. Rename: DimChannel
3. Enable load

How to validate:

- Row count: **3 rows**
- Channels: In-Store, Online, Click & Collect

Clean DimDate

What to do:

1. Right-click stg_dim_date → **Reference**
2. Rename: DimDate
3. Enable load
4. Verify Australian FY columns exist:
   - Date: Date
   - Year: Whole Number
   - **FY:** Text (FY2023, FY2024, FY2025)
   - Quarter: Text (Q1, Q2, Q3, Q4)
   - Month: Whole Number (1-12)
   - MonthName: Text (January, etc.)
   - MonthYear: Text (Jul-2022, etc.)
   - DayOfWeek: Text (Monday, etc.)
   - DayOfMonth: Whole Number (1-31)
   - IsWeekend: Whole Number (0 or 1)
   - **IsEOFY:** Whole Number (1 if June 30, else 0)

> **Why it matters:** Australian Financial Year runs **July 1 - June 30**. The FY column correctly assigns dates:
>
> - July 1, 2023 → FY2024
> - June 30, 2024 → FY2024
> - July 1, 2024 → FY2025

How to validate:

Filter **Date** column to June 30, 2024:

- FY should show: **FY2024**
- IsEOFY should show: **1**

Filter **Date** column to July 1, 2024:

- FY should show: FY2025

13. M CODE DEEP DIVE

Understanding Power Query M Language

Power Query uses **M (Mashup) language** for data transformations. Each step you perform in the UI generates M code.

View M Code for a Query

What to do:

1. Click on the clean_transactions query
2. Click **View** ribbon → **Advanced Editor**

The Advanced Editor shows the complete M code for all transformation steps.

Example M Code:

```
let
    Source = stg_daily_transactions,
    RemovedDuplicates = Table.Distinct(Source, {"TransactionID"}),
    UppercaseState = Table.TransformColumns(RemovedDuplicates,{{"State", Text.Upper, type text}}),
    ReplacedNSW = Table.ReplaceValue(UppercaseState,"NEW SOUTH WALES","NSW",Replacer.ReplaceText,
{"State"}),
    ReplacedVIC = Table.ReplaceValue(ReplacedNSW,"VICTORIA","VIC",Replacer.ReplaceText,{"State"}),
    ReplacedQLD = Table.ReplaceValue(ReplacedVIC,"QUEENSLAND","QLD",Replacer.ReplaceText,{"State"}),
    AddedGSTCalculated = Table.AddColumn(ReplacedQLD, "GST_Calculated", each if [GST] = null then [Revenue] *
0.10 else [GST], type number),
    RemovedGSTOld = Table.RemoveColumns(AddedGSTCalculated,{"GST"}),
    RenamedGST = Table.RenameColumns(RemovedGSTOld,{{"GST_Calculated", "GST"}}),
    FilteredQuantity = Table.SelectRows(RenamedGST, each [Quantity] > 0),
    ChangedType = Table.TransformColumnTypes(FilteredQuantity,{
        {"TransactionID", type text},
        {"TransactionDate", type date},
        {"StoreID", type text},
        {"ProductID", type text},
        {"CustomerID", type text},
        {"ChannelID", type text},
        {"Quantity", Int64.Type},
        {"UnitPrice", Currency.Type},
        {"Discount", type number},
        {"Revenue", Currency.Type},
        {"Cost", Currency.Type},
        {"GST", Currency.Type},
        {"TransactionTime", type time}
    }),
    RemovedErrors = Table.RemoveRowsWithErrors(ChangedType, {"TransactionDate"})
in
    RemovedErrors
```

Why it matters: Understanding M code allows you to:

- Debug transformation errors
- Copy transformation logic to other queries
- Create custom functions
- Optimise performance

Key M Functions

Function	Purpose	Example
Table.Distinct	Remove duplicates	Table.Distinct(Source, {"TransactionID"})
Table.TransformColumns	Apply function to column	Table.TransformColumns(Source, {{"State", Text.Upper}})
Table.ReplaceValue	Find/replace	Table.ReplaceValue(Source, "old", "new", Replacer.ReplaceText, {"Column"})
Table.AddColumn	Add calculated column	Table.AddColumn(Source, "NewCol", each [Col1] * [Col2])
Table.SelectRows	Filter rows	Table.SelectRows(Source, each [Quantity] > 0)
Table.RemoveColumns	Delete columns	Table.RemoveColumns(Source, {"ColName"})
Table.RenameColumns	Rename columns	Table.RenameColumns(Source, {{"OldName", "NewName"}})

Create a Custom Function (Advanced)

What to do:

Create a reusable function to calculate GST:

1. Click **Home** → **New Source** → **Blank Query**
2. Click **View** → **Advanced Editor**
3. Replace contents with:

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

1. Click **Done**
2. Rename query: fn_CalculateGST
3. Disable load (it's a function, not data)

Why it matters: Custom functions encapsulate business logic. You can call fn_CalculateGST in any query instead of repeating the formula.

How to validate:

Test the function:

1. Create a new blank query
2. Advanced Editor:

```
= fn_CalculateGST(100)
```

1. Result should show: **10**
2. Delete this test query

14. VALIDATION & RECONCILIATION

Create Validation Summary Query

What to do:

Create a summary query to validate all transformations:

1. Click **Home** → **New Source** → **Blank Query**
2. Rename: Validation_Summary
3. Click **View** → **Advanced Editor**
4. Enter this M code:

```
let
    ValidationTable = #table(
        {"Query", "Source_Rows", "Clean_Rows", "Rows_Removed", "Loss_Percentage"},
        {
            {"Transactions",
             Table.RowCount(stg_daily_transactions),
             Table.RowCount(clean_transactions),
             Table.RowCount(stg_daily_transactions) - Table.RowCount(clean_transactions),
             (Table.RowCount(stg_daily_transactions) - Table.RowCount(clean_transactions)) /
Table.RowCount(stg_daily_transactions) * 100
            },
            {"DimStore",
             Table.RowCount(stg_dim_store),
             Table.RowCount(DimStore),
             0,
             0
            },
            {"DimProduct",
             Table.RowCount(stg_dim_product),
             Table.RowCount(DimProduct),
             0,
             0
            },
            {"DimCustomer",
             Table.RowCount(stg_dim_customer),
             Table.RowCount(DimCustomer),
             0,
             0
            }
        }
    )
in
    ValidationTable
```

1. Click **Done**
2. Enable load for this query

> **Why it matters:** This validation query compares row counts between staging and clean queries. Any unexpected data loss is immediately visible.

How to validate:

The query output should look like:

Query	Source_Rows	Clean_Rows	Rows_Removed	Loss_Percentage
Transactions	10000	9200	800	8.0
DimStore	50	50	0	0.0
DimProduct	2000	2000	0	0.0
DimCustomer	20000	20000	0	0.0

Expected data loss: 8% from transactions (duplicates, invalid dates, bad quantities)
Alert threshold: >10% loss indicates a problem

Column Quality Validation

What to do:

For the clean_transactions query:

1. Enable column profiling (entire dataset)
2. For each column, verify:

Column	Expected Quality	Validation
TransactionID	100% Valid, 0% Empty, 100% Unique	✓
TransactionDate	100% Valid, 0% Empty	✓
State	100% Valid, 0% Empty, 3 Distinct (NSW/VIC/QLD)	✓
GST	100% Valid, 0% Empty	✓
Quantity	100% Valid, 0% Empty, All > 0	✓

> **Why it matters:** Column profiling catches issues that row counts miss (e.g., duplicate IDs, unexpected nulls).

How to validate:

Click each column and verify the quality metrics match the expected values above.

15. COMMON MISTAKES & TROUBLESHOOTING

Issue 1: "Expression.Error: The column 'State' of the table wasn't found"

Cause: Column name mismatch (case-sensitive in M code)

Solution:

- Check exact column name in source
- Use exact casing: State not state or STATE
- Use Table.ColumnNames(Source) to list all columns

Issue 2: Duplicates Not Removed

Cause: Table.Distinct without specifying key column removes row-level duplicates only

Solution:

```
// Wrong - removes only completely identical rows
Table.Distinct(Source)

// Correct - removes duplicates based on TransactionID
Table.Distinct(Source, {"TransactionID"})
```

Issue 3: GST Calculation Errors

Cause: Trying to multiply null values

Solution:

```
// Wrong - fails if Revenue is null
[Revenue] * 0.10

// Correct - handles nulls
if [Revenue] = null then null else [Revenue] * 0.10
```

Issue 4: Postcode Leading Zeros Lost

Cause: Postcode stored as number instead of text

Solution:

```
// Wrong - 0800 becomes 800
Table.TransformColumnTypes(Source, {{"Postcode", Int64.Type}})

// Correct - preserves leading zeros
Table.TransformColumnTypes(Source, {{"Postcode", type text}})

// Fix existing: Add leading zeros back
Table.TransformColumns(Source, {{"Postcode", each Text.PadStart(Text.From(_), 4, "0"), type text}})
```

Issue 5: Slow Query Performance

Cause: Loading entire dataset for profiling or not folding queries

Solution:

- Disable "Column profiling based on: Entire dataset" during development
- Check Query Diagnostics: **Tools → Session Diagnostics → Start Diagnostics**
- Enable query folding where possible (SQL queries)

Issue 6: Cannot Remove Staging Queries

Cause: Clean queries reference staging queries

Solution:

- Never delete staging queries that are referenced
- Disable load instead: Right-click → Uncheck **Enable load**

Issue 7: Date Format Errors

Cause: Regional settings mismatch (DD/MM/YYYY vs MM/DD/YYYY)

Solution:

```
// Explicit date parsing for Australian format (DD/MM/YYYY)
Date.FromText([DateColumn], [Format="DD/MM/YYYY", Culture="en-AU"])
```

16. PORTFOLIO DOCUMENTATION

Create GitHub Repository

What to do:

1. Create a new GitHub repository: FreshMarket-PowerBI-ETL
2. Add a README.md:

```
# FreshMarket Australia - Power BI ETL Pipeline

## Project Overview
Built a comprehensive ETL pipeline for FreshMarket Australia, a 50-store grocery chain, integrating 5 data sources into a unified analytics platform.

## Business Context
**Industry:** Australian Grocery Retail
**Challenge:** Data scattered across SQL Server, CSV exports, Excel budgets, SharePoint campaigns, and supplier APIs
**Solution:** Power Query ETL pipeline with medallion architecture (Bronze → Silver → Gold)

## Technical Implementation

### Data Sources
- **SQL Server:** 5 dimension tables (Stores, Products, Customers, Channels, Dates)
- **CSV:** Daily POS transactions (10,000 records with quality issues)
- **Excel:** Finance budgets & forecasts (FY2024)
- **SharePoint:** 6 marketing campaign files (548 records)
- **Web API:** Supplier product catalogue (JSON, 100 products)

### Data Quality Challenges Solved
-
```

17. INTERVIEW PREPARATION

Key Talking Points

When discussing this ETL project in interviews, emphasise:

Business Impact:

> "I built an ETL pipeline for a 50-store Australian grocery chain that reduced manual data reconciliation from 15 hours to 3 minutes monthly. The pipeline validated GST calculations across 50,000 transactions, ensuring 100% tax compliance."

Australian Context:

> "I handled Australian-specific data challenges: standardising state codes from 15 variations to 8 official values, preserving 4-digit postcodes with leading zeros using Text type, calculating 10% GST on all transactions, and aligning date tables to Australian financial year (July 1 - June 30)."

Technical Depth:

> "I integrated 5 data sources using Power Query: SQL Server dimensions, messy CSV POS exports with 5% duplicates, Excel budgets with null forecasts, SharePoint campaign files with inconsistent schemas, and a JSON web API. I implemented staging → clean query pattern following medallion architecture."

Data Quality:

> "I validated data quality using Power Query profiling: removed 500 duplicate transactions, standardised state values to NSW/VIC/QLD only, recalculated 1,575 missing GST values, filtered 525 invalid quantity records, and reconciled final row counts to ±2% accuracy."

Common Interview Questions

1

"Walk me through your ETL process."

Answer:

"I follow a three-stage medallion architecture:

Bronze (Staging): Load raw data from all sources with minimal transformation. This preserves original data for auditing.

Silver (Clean): Reference staging queries and apply transformations: remove duplicates, standardise formats, handle nulls, enforce data types. For example, I standardised 15+ state variations to 8 official Australian codes.

Gold (Aggregated): Create business-ready datasets for reporting. In my FreshMarket project, I built clean dimension tables and a validated fact table ready for star schema modelling.

At each stage, I validate row counts and column quality using Power Query profiling."

2

"How did you handle data quality issues?"

Answer:

"I used Power Query's column profiling to identify issues:

- **Duplicates:** Detected 5% duplicate TransactionIDs. Used Table.Distinct with explicit key column.
- **Nulls:** Found 3% missing GST values. Calculated as Revenue × 0.10 using conditional logic: if [GST] = null then [Revenue] * 0.10 else [GST]
- **Format inconsistencies:** State column had mixed case and full names. Applied UPPERCASE then Replace Values to map variations to NSW/VIC/QLD.
- **Invalid values:** Filtered out zero/negative quantities and malformed dates using Table.SelectRows and Table.RemoveRowsWithErrors.

I documented all quality issues in a validation log with before/after row counts."

3

"What's the difference between CSV and SQL connectors?"

Answer:

"CSV is flat-file based: you load the entire file into memory. It's simple but doesn't support query folding or incremental refresh.

SQL Server connector leverages the database engine: Power Query pushes transformations (filters, joins) back to SQL Server as native queries. This is called **query folding** and dramatically improves performance for large datasets.

In my project, I used SQL for clean dimension tables (Products, Stores, Customers) because they benefit from database indexing. I used CSV for the messy transaction export because it needed extensive Power Query cleaning that wouldn't fold anyway."

4

"How did you optimise refresh performance?"

Answer:

"Three strategies:

1. **Query folding:** For SQL sources, I kept transformations simple (SELECT, WHERE, JOIN) so they execute in the database, not in Power Query.
2. **Disable load for staging queries:** Staging queries are reference-only. Disabling load prevents them from consuming memory in the model.
3. **Modular queries:** I broke transformations into logical steps instead of one complex query. This makes debugging faster and allows parallel processing.

Result: Refresh time stayed under 3 minutes for 70,000+ total rows across all sources."

5

"Why use Power Query instead of SQL for transformations?"

Answer:

"It depends on the source and transformation complexity:

Use SQL when:

- Source is a database and transformations fold
- You need database-specific functions (window functions, CTEs)
- Team is SQL-focused

Use Power Query when:

- Combining multiple source types (CSV + Excel + SharePoint + API)
- Non-technical users need to maintain transformations (visual interface)
- Business logic is complex and benefits from M functions
- Source is already in Power BI (incremental refresh)

In my project, I used Power Query because I integrated 5 different source types - SQL alone couldn't handle CSV files, Excel sheets, and JSON APIs."

6

"Explain your approach to Australian data validation."

Answer:

"Australian business data has specific requirements:

GST (10% tax): I validated that every transaction had GST = Revenue × 0.10. For 1,575 missing values, I recalculated using M formula.

State codes: Standardised to 8 official codes (NSW, VIC, QLD, WA, SA, TAS, NT, ACT) by mapping variations like 'New South Wales' → 'NSW'.

Postcodes: Preserved 4-digit format with leading zeros (e.g., 0800 for NT) by using Text type instead of Number.

Financial year: Created Date table with FY column that assigns dates correctly: July 1, 2023 → FY2024, June 30, 2024 → FY2024.

EOFY (End of Financial Year): Added IsEOFY flag for June 30 to support year-end reporting.

These validations ensure compliance with Australian tax regulations and enable accurate state-level analysis."

Practice Presenting Your Work

Elevator Pitch (30 seconds):

> "I built a Power BI ETL pipeline for an Australian grocery retailer, integrating 5 data sources - SQL Server, CSV, Excel, SharePoint, and Web API. The pipeline handles 50,000 transactions, validates 10% GST on all sales, standardises state codes across NSW/VIC/QLD, and reduces manual reconciliation from 15 hours to 3 minutes monthly. I used Power Query with staging-to-clean pattern, custom M functions for business logic, and column profiling to achieve 95%+ data quality."

2-Minute Deep Dive:

> "The FreshMarket project solved a real enterprise challenge: data scattered across 5 systems with quality issues.
>
> I started by connecting Power Query to all sources: SQL Server for master data, daily CSV exports from POS systems, Excel budgets from finance, SharePoint campaign uploads, and a supplier JSON API.
>
> I implemented medallion architecture: staging queries load raw data with minimal change, clean queries apply transformations and validation, and the final model loads only validated data.
>
> The biggest challenge was data quality. The POS CSV had 5% duplicate transactions, 3% missing GST values, and 15+ variations of state names. I used Power Query profiling to identify issues, then built transformations: Table.Distinct for deduplication, conditional formulas for GST calculation, and Replace Values for state standardisation.
>
> For Australian context, I preserved 4-digit postcodes with leading zeros, validated 10% GST on all transactions, and created a date table aligned to July-June financial year.
>
> The result: 95%+ data completeness, zero duplicates, 100% GST accuracy, and sub-3-minute refresh times. This pipeline now feeds star schema models and executive dashboards."

CONCLUSION

You have successfully completed **Video 1: Power Query & ETL Pipeline**!

What You Built

5 data source connections

CSV, Excel, SQL Server, SharePoint, Web API

Staging → Clean query pattern

Medallion architecture

Australian data validation

GST, states, postcodes, FY dates

Data quality transformations

Duplicates, nulls, formats, types

M code understanding

Custom functions, error handling

Validation framework

Row counts, column profiling, reconciliation

Key Metrics Achieved

Metric	Target	Actual	Status
Data completeness	>95%	92% (9,200 / 10,000)	✅
Duplicate removal	0	0 (removed 500)	✅
GST accuracy	100%	100% (calculated 1,575 missing)	✅
State standardisation	8 values	3 values (NSW/VIC/QLD)	✅
Refresh performance	<5 min	~3 min	✅

Next Steps

Video 2: Data Modelling & DAX

- Build star schema (1 fact + 5 dimensions)
- Create relationships (1:*, single direction)
- Write DAX measures (revenue, margins, time intelligence)
- Implement Australian FY date table
- Optimise model performance

Save Your Work:

1. Click **Home** ribbon → **Close & Apply**
2. Power BI loads all enabled queries into the model
3. Click **File** → **Save**

Expected load time: ~30 seconds for all queries

Validation:

- Fields pane should show: DimStore, DimProduct, DimCustomer, DimChannel, DimDate, clean_transactions
- Staging queries should NOT appear (load disabled)

APPENDIX: M CODE REFERENCE

Common M Functions

```
// Text functions
Text.Upper("hello")                 // "HELLO"
Text.Lower("HELLO")                 // "hello"
Text.Proper("hello world")          // "Hello World"
Text.Trim("  hello  ")              // "hello"
Text.Replace("hello", "l", "L")     // "heLLo"
Text.PadStart("42", 4, "0")         // "0042"

// Table functions
Table.RowCount(tableName)           // Count rows
Table.Distinct(tableName, {"KeyColumn"})  // Remove duplicates
Table.SelectRows(tableName, each [Column] > 0)  // Filter
Table.AddColumn(tableName, "NewCol", each [Col1] + [Col2])  // Add column
Table.RemoveColumns(tableName, {"Col1", "Col2"})  // Delete columns
Table.RenameColumns(tableName, {{"OldName", "NewName"}})  // Rename

// Date functions
Date.FromText("2024-12-30")         // Parse date
Date.Year([DateColumn])             // Extract year
Date.Month([DateColumn])            // Extract month (1-12)
Date.DayOfWeek([DateColumn])        // Day of week (0=Sunday)

// Number functions
Number.Round(3.14159, 2)            // 3.14
Number.Mod(10, 3)                   // 1 (remainder)
Currency.From(100.50)               // Convert to currency type

// Conditional logic
if [Value] = null then 0 else [Value]  // Null handling
if [Amount] > 100 then "High" else "Low"  // If-then-else
```

Validation Query Template

```
let
 Source = /* your source query */,

 // Add validation steps
 Validated = Table.SelectRows(Source, each
 [Column1] <> null and
 [Column2] > 0 and
 Text.Length([Column3]) = 4
 ),

 // Count validation failures
 TotalRows = Table.RowCount(Source),
 ValidRows = Table.RowCount(Validated),
 FailedRows = TotalRows - ValidRows,
 FailureRate = FailedRows / TotalRows * 100
in
 Validated
```

End of Video 1 Playbook

Thank you

You've completed a comprehensive ETL pipeline using Power Query!

For questions or support:

- GitHub Issues: [repository link]
- LinkedIn: [your profile]
- Email: [your contact]

This playbook is part of a comprehensive Power BI training series covering ETL, data modelling, DAX, and visualisation for Australian business contexts.