COS Audit – End-to-End Technical Documentation (Power Query + Power BI)

Author: Dipankar Nikesh Sugat
Purpose: Payroll audit to validate *Change of Shift (COS)* penalties — whether COS **should** have occurred vs whether it was **actually paid**.

Part A — Excel Setup (One-time, Never Breaks)

A1. Fixed Folder Structure

Create a permanent folder:

```
C:\COS_Audit
```

Place the raw payroll file here and rename it to:

```
Payroll_Raw.xlsx
```

Why: Power BI refresh depends on a stable file path.

A2. Convert Raw Data to an Excel Table (MANDATORY)

1. Open `Payroll_Raw.xlsx`
2. Select any cell in the raw data
3. Press **Ctrl + T** → tick **My table has headers** → OK
4. Go to **Table Design → Table Name**
5. Rename table to:

   ```
   Raw_Payroll
   ```

A3. Column Rules (Never Change These)

These names and types must remain unchanged:

Column Name	Type
EmployeeNumber	Text
partitionDate	Date
ShiftStartDateTime	DateTime
ShiftEndDateTime	DateTime
SumofHours	Decimal
Type	Text (ORD / LEAVE / COS)
payCode	Text
costCenter	Text
Classification	Text

Outcome: Replace file → Refresh → Done.

Part B — Power BI Project Setup

B1. Create PBIX

1. Open Power BI Desktop
2. Save as:

```
C:\COS_Audit\COS_Audit.pbix
```

B2. Load Excel Data

1. Home → Get Data → Excel
2. Select `Payroll_Raw.xlsx`
3. Tick `Raw_Payroll`
4. Click **Transform Data**

You are now in **Power Query Editor**.

Part C — Base Query (No Business Logic)

Query 1: `Raw_Payroll`

Purpose: clean + typing only.

Steps: - Rename query → `Raw_Payroll` - Set data types exactly as defined above

❌No filtering
❌No grouping
❌No COS logic

This query is NEVER touched again.

Part D — Split Data (Critical Design Rule)

Query 2: `Shift_Base` (ORD + LEAVE only)

1. Right-click `Raw_Payroll` → **Reference**
2. Rename → `Shift_Base`
3. Filter `Type` → keep **ORD, LEAVE**

Why: COS should never influence expected-COS logic.

Query 3: `COS_Paid` (Facts Only)

1. Right-click `Raw_Payroll` → **Reference**
2. Rename → `COS_Paid`
3. Filter `Type` → keep **COS**
4. Keep only:
5. EmployeeNumber
6. partitionDate
7. (optional descriptors)
8. Remove duplicates on *(EmployeeNumber + partitionDate)*

Meaning: One row = COS was paid.

Part E — Daily Consolidation

Query 4: `Daily_Shifts`

Goal: One row per *employee per day*.

Grouping

Home → Group By (Advanced): - Group by: EmployeeNumber, partitionDate - Aggregations: - ShiftStartDateTime → Min - ShiftEndDateTime → Max - SumofHours → Sum - Add **All Rows** → name `AllRows`

Create Winning Type by Hours

FinalType (M code)

```
let
    t = [AllRows],
    byType = Table.Group(
        t,
        {"Type"},
        {{"TypeHours", each List.Sum([SumofHours]), type number}}
    ),
    sorted = Table.Sort(byType, {{"TypeHours", Order.Descending}}),
    topType = sorted{0}[Type]
in
    topType
```

Meaning: Type with highest hours wins the day.

WinnerRow (Bring Back Descriptors)

```
let
    t = [AllRows],
    w = [FinalType],
    onlyWinner = Table.SelectRows(t, each [Type] = w),
    sorted = Table.Sort(onlyWinner, {{"SumofHours", Order.Descending}})
in
    sorted{0}
```

Expand `WinnerRow` → payCode, costCenter, Classification.

Cleanup: - Remove `AllRows`, `WinnerRow` - Rename `FinalType` → `Type`

Part F — COS Expected Logic

Query 5: `COS_Expected`

Reference `Daily_Shifts`.

Mandatory Sort Order

1. EmployeeNumber ↑
2. partitionDate ↑

3. ShiftStartDateTime ↑

OrdStartOnly

```
if [Type] = "ORD" then Time.From([ShiftStartDateTime]) else null
```

Fill Down.

Purpose: LEAVE inherits last ORD start.

Previous ORD Start (Index Pattern)

Add Index: - `Index` → start 0 - `Index_Prev` → start 1

Self-merge on: - Index = Index_Prev

Expand: - OrdStartOnly → rename `Prev_OrdStart`

Employee Partition Fix (CRITICAL)

Create `Prev_EmployeeNumber` using the same index merge.

```
if [EmployeeNumber] = [Prev_EmployeeNumber]
then [Prev_OrdStart]
else null
```

GapHours

```
if [Type] <> "ORD" or [Prev_OrdStart] = null then null
else Duration.TotalHours(
    Time.From([ShiftStartDateTime]) - [Prev_OrdStart]
)
```

Expected_COS

```
if [GapHours] <> null and Number.Abs([GapHours]) >= 4
then "Yes"
else "No"
```

Part G — Compare With COS Paid

Merge `COS_Expected` with `COS_Paid` on: - EmployeeNumber - partitionDate

Left Outer Join.

Expand ONE column (EmployeeNumber).

COS_Actually_Paid

```
if [COS_Paid.EmployeeNumber] <> null then "Yes" else "No"
```

Final Audit Status

```
if [Expected_COS] = "Yes" and [COS_Actually_Paid] = "No" then "❌ Missing COS"
else if [Expected_COS] = "No" and [COS_Actually_Paid] = "Yes" then "⚠️ Wrongly
Paid"
else "✅ Correct"
```

Final Mental Model (Remember This)

- **Index** → previous shift
- **GapHours** → should COS exist
- **Merge** → was COS paid
- **Compare** → audit result

This model is payroll-auditor safe and production-ready.