```sql
select * from [Sample Interchange Dataset]


-- Null check
SELECT
    CASE
        WHEN EXISTS (
            SELECT 1
            FROM [Sample Interchange Dataset]
            WHERE
                InterChangeID IS NULL OR
                ReportingDate IS NULL OR
                CardNumber IS NULL OR
                AcquirerNetworkGroup IS NULL OR
                Merchant IS NULL OR
                UsageType IS NULL OR
                ProductType IS NULL OR
                SettlementAmount IS NULL OR
                InterchangeRevenue IS NULL OR
                TransactionCount IS NULL
        )
        THEN 'NULLS Found'
        ELSE 'NO NULLS Found'
    END AS NullCheck;


--Null check column wise
    SELECT
    SUM(CASE WHEN InterChangeID IS NULL THEN 1 ELSE 0 END) AS Null_InterChangeID,
    SUM(CASE WHEN ReportingDate IS NULL THEN 1 ELSE 0 END) AS Null_ReportingDate,
    SUM(CASE WHEN CardNumber IS NULL THEN 1 ELSE 0 END) AS Null_CardNumber,
    SUM(CASE WHEN AcquirerNetworkGroup IS NULL THEN 1 ELSE 0 END) AS
      Null_AcquirerNetworkGroup,
    SUM(CASE WHEN Merchant IS NULL THEN 1 ELSE 0 END) AS Null_Merchant,
    SUM(CASE WHEN UsageType IS NULL THEN 1 ELSE 0 END) AS Null_UsageType,
    SUM(CASE WHEN ProductType IS NULL THEN 1 ELSE 0 END) AS Null_ProductType,
    SUM(CASE WHEN SettlementAmount IS NULL THEN 1 ELSE 0 END) AS
      Null_SettlementAmount,
    SUM(CASE WHEN InterchangeRevenue IS NULL THEN 1 ELSE 0 END) AS
      Null_InterchangeRevenue,
    SUM(CASE WHEN TransactionCount IS NULL THEN 1 ELSE 0 END) AS Null_TransactionCount
FROM [Sample Interchange Dataset];

--DATA QUALITY CHECKS
-- Check for duplicate InterChangeID
SELECT InterChangeID, COUNT(*) AS Cnt
FROM [Sample Interchange Dataset]
GROUP BY InterChangeID
HAVING COUNT(*) > 1;

-- Check invalid dates
SELECT *
```

```sql
FROM [Sample Interchange Dataset]
WHERE ReportingDate IS NULL;

--Check for negative amounts (should not happen)
SELECT *
FROM [Sample Interchange Dataset]
WHERE SettlementAmount < 0 OR InterchangeRevenue < 0;

--Check missing Merchant names
SELECT *
FROM [Sample Interchange Dataset]
WHERE Merchant IS NULL OR LTRIM(RTRIM(Merchant)) = '';

--Check for zero or unrealistic transaction counts
SELECT *
FROM [Sample Interchange Dataset]
WHERE TransactionCount <= 0;

--check all data types
SELECT COLUMN_NAME, DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'Sample Interchange Dataset';


--check date format issues
SELECT *
FROM [Sample Interchange Dataset]
WHERE TRY_CONVERT(date, ReportingDate) IS NULL;

--Check duplicates using the entire row (all columns)
SELECT *
FROM [Sample Interchange Dataset]
GROUP BY
    InterChangeID,
    ReportingDate,
    CardNumber,
    AcquirerNetworkGroup,
    Merchant,
    UsageType,
    ProductType,
    SettlementAmount,
    InterchangeRevenue,
    TransactionCount
HAVING COUNT(*) > 1;


-- Check duplicates based on key fields (InterChangeID + ReportingDate)
SELECT
    InterChangeID,
    ReportingDate,
    COUNT(*) AS DuplicateCount
FROM [Sample Interchange Dataset]
```

```sql
GROUP BY InterChangeID, ReportingDate
HAVING COUNT(*) > 1;


-- Identify duplicate rows with ROW_NUMBER()
WITH Dups AS
(
    SELECT *,
           ROW_NUMBER() OVER (
                PARTITION BY
                    InterChangeID,
                    ReportingDate,
                    CardNumber,
                    AcquirerNetworkGroup,
                    Merchant,
                    UsageType,
                    ProductType,
                    SettlementAmount,
                    InterchangeRevenue,
                    TransactionCount
                ORDER BY InterChangeID
           ) AS rn
    FROM [Sample Interchange Dataset]
)
SELECT *
FROM Dups
WHERE rn > 1;


-- Remove white spaces from each column
UPDATE [Sample Interchange Dataset]
SET
    InterChangeID = LTRIM(RTRIM(InterChangeID)),
    CardNumber = LTRIM(RTRIM(CardNumber)),
    AcquirerNetworkGroup = LTRIM(RTRIM(AcquirerNetworkGroup)),
    Merchant = LTRIM(RTRIM(Merchant)),
    UsageType = LTRIM(RTRIM(UsageType)),
    ProductType = LTRIM(RTRIM(ProductType));


-- Creating Schema (Star), Dimension and Fact Tables

CREATE TABLE Dim_Date (
    DateKey INT PRIMARY KEY,
    FullDate DATE,
    Year INT,
    Month INT,
    MonthName VARCHAR(20),
    Quarter INT
);
```

```sql
CREATE TABLE Dim_Merchant (
    MerchantKey INT IDENTITY(1,1) PRIMARY KEY,
    MerchantName VARCHAR(255)
);


CREATE TABLE Dim_Card (
    CardKey INT IDENTITY(1,1) PRIMARY KEY,
    CardNumber VARCHAR(50)
);


CREATE TABLE Dim_Acquirer (
    AcquirerKey INT IDENTITY(1,1) PRIMARY KEY,
    AcquirerNetworkGroup VARCHAR(100)
);


CREATE TABLE Dim_Category (
    CategoryKey INT IDENTITY(1,1) PRIMARY KEY,
    UsageType VARCHAR(100),
    ProductType VARCHAR(100),
    CombinedCategory VARCHAR(200)
);


INSERT INTO Dim_Date (DateKey, FullDate, Year, Month, MonthName, Quarter)
SELECT DISTINCT
    CONVERT(VARCHAR(8), ReportingDate, 112) AS DateKey,
    ReportingDate,
    YEAR(ReportingDate),
    MONTH(ReportingDate),
    DATENAME(MONTH, ReportingDate),
    DATEPART(QUARTER, ReportingDate)
FROM [Sample Interchange Dataset];


INSERT INTO Dim_Merchant (MerchantName)
SELECT DISTINCT Merchant
FROM [Sample Interchange Dataset];


INSERT INTO Dim_Card (CardNumber)
SELECT DISTINCT CardNumber
FROM [Sample Interchange Dataset];


INSERT INTO Dim_Acquirer (AcquirerNetworkGroup)
SELECT DISTINCT AcquirerNetworkGroup
FROM [Sample Interchange Dataset];
```

```sql
INSERT INTO Dim_Category (UsageType, ProductType, CombinedCategory)
SELECT DISTINCT
    UsageType,
    ProductType,
    UsageType + ' - ' + ProductType
FROM [Sample Interchange Dataset];


CREATE TABLE Fact_Interchange (
    InterChangeID INT PRIMARY KEY,
    DateKey INT,
    MerchantKey INT,
    CardKey INT,
    AcquirerKey INT,
    CategoryKey INT,
    SettlementAmount DECIMAL(18,2),
    InterchangeRevenue DECIMAL(18,2),
    TransactionCount INT,

    FOREIGN KEY (DateKey) REFERENCES Dim_Date(DateKey),
    FOREIGN KEY (MerchantKey) REFERENCES Dim_Merchant(MerchantKey),
    FOREIGN KEY (CardKey) REFERENCES Dim_Card(CardKey),
    FOREIGN KEY (AcquirerKey) REFERENCES Dim_Acquirer(AcquirerKey),
    FOREIGN KEY (CategoryKey) REFERENCES Dim_Category(CategoryKey)
);


INSERT INTO Fact_Interchange (
    InterChangeID, DateKey, MerchantKey, CardKey, AcquirerKey, CategoryKey,
    SettlementAmount, InterchangeRevenue, TransactionCount
)
SELECT
    s.InterChangeID,
    CONVERT(VARCHAR(8), s.ReportingDate, 112) AS DateKey,
    m.MerchantKey,
    c.CardKey,
    a.AcquirerKey,
    cat.CategoryKey,
    s.SettlementAmount,
    s.InterchangeRevenue,
    s.TransactionCount
FROM [Sample Interchange Dataset] s
JOIN Dim_Merchant m ON s.Merchant = m.MerchantName
JOIN Dim_Card c ON s.CardNumber = c.CardNumber
JOIN Dim_Acquirer a ON s.AcquirerNetworkGroup = a.AcquirerNetworkGroup
JOIN Dim_Category cat
     ON s.UsageType = cat.UsageType
    AND s.ProductType = cat.ProductType;


select * from Dim_Category
select * from Dim_Acquirer
```

```sql
select * from Dim_Card
Select * from Dim_Date
select * from Dim_Merchant
select * from Fact_Interchange



-- Business Requirements

select * from Dim_Category
select * from Dim_Acquirer
select * from Dim_Card
Select * from Dim_Date
select * from Dim_Merchant
select * from Fact_Interchange

-- Which top 10 merchants have the highest interchange volume?

SELECT
    M.MerchantName,
    SUM(F.SettlementAmount) AS InterchangeVolume
FROM dbo.Fact_Interchange F
INNER JOIN dbo.Dim_Merchant M
       ON F.MerchantKey = M.MerchantKey
GROUP BY
    M.MerchantName
ORDER BY
    InterchangeVolume DESC
OFFSET 0 ROWS FETCH NEXT 10 ROWS ONLY;


-- Can we see the interchange rate per merchant?
SELECT
    M.MerchantName,
    SUM(F.InterchangeRevenue) AS TotalInterchangeRevenue,
    SUM(F.SettlementAmount) AS TotalSettlementAmount,
    CASE
        WHEN SUM(F.SettlementAmount) = 0 THEN '0%'
        ELSE FORMAT(
            (SUM(F.InterchangeRevenue) * 1.0 / SUM(F.SettlementAmount)),
            'P2'
        )
    END AS InterchangeRate_Percentage
FROM dbo.Fact_Interchange F
INNER JOIN dbo.Dim_Merchant M
       ON F.MerchantKey = M.MerchantKey
GROUP BY
    M.MerchantName
ORDER BY
    (SUM(F.InterchangeRevenue) * 1.0 / SUM(F.SettlementAmount)) DESC;
```

```sql
-- What percent of the total does the top interchange volume merchant represent?

WITH MerchantVolumes AS
(
    SELECT
        M.MerchantName,
        SUM(F.SettlementAmount) AS TotalVolume
    FROM dbo.Fact_Interchange F
    INNER JOIN dbo.Dim_Merchant M
        ON F.MerchantKey = M.MerchantKey
    GROUP BY M.MerchantName
),
Total AS
(
    SELECT SUM(TotalVolume) AS GrandTotalVolume
    FROM MerchantVolumes
)
SELECT
    MV.MerchantName AS TopMerchant,
    MV.TotalVolume AS TopMerchantVolume,
    T.GrandTotalVolume,
    FORMAT( (MV.TotalVolume * 1.0 / T.GrandTotalVolume), 'P2')
        AS PercentageOfTotal
FROM MerchantVolumes MV
CROSS JOIN Total T
ORDER BY MV.TotalVolume DESC
OFFSET 0 ROWS FETCH NEXT 1 ROWS ONLY;


--Can you rank the merchants in order of greatest to least interchange volume?

SELECT
    dm.MerchantName,
    SUM(fi.SettlementAmount) AS TotalInterchangeVolume,
    RANK() OVER (ORDER BY SUM(fi.SettlementAmount) DESC) AS MerchantRank
FROM Fact_Interchange fi
JOIN Dim_Merchant dm
      ON fi.MerchantKey = dm.MerchantKey
GROUP BY dm.MerchantName
ORDER BY TotalInterchangeVolume DESC;


-- Can we see the current rank, interchange revenue, and percentage of total versus
  the previous month's rank, interchange revenue, and percentage of total?

WITH MonthlyData AS
(
    SELECT
        dm.MerchantKey,
        dm.MerchantName,
        d.Year,
        d.Month,
```

```
        SUM(fi.InterchangeRevenue) AS TotalRevenue
    FROM Fact_Interchange fi
    JOIN Dim_Merchant dm ON fi.MerchantKey = dm.MerchantKey
    JOIN Dim_Date d ON fi.DateKey = d.DateKey
    GROUP BY dm.MerchantKey, dm.MerchantName, d.Year, d.Month
),

RankedData AS
(
    SELECT
        *,
        RANK() OVER (
            PARTITION BY Year, Month
            ORDER BY TotalRevenue DESC
        ) AS RevenueRank,

        (TotalRevenue * 1.0 / SUM(TotalRevenue) OVER (PARTITION BY Year, Month)) * 100
            AS PercentOfTotal
    FROM MonthlyData
),

WithPrev AS
(
    SELECT
        cur.MerchantKey,
        cur.MerchantName,
        cur.Year,
        cur.Month,

        cur.RevenueRank AS CurrentRank,
        cur.TotalRevenue AS CurrentRevenue,
        cur.PercentOfTotal AS CurrentPercent,

        prev.RevenueRank AS PrevRank,
        prev.TotalRevenue AS PrevRevenue,
        prev.PercentOfTotal AS PrevPercent
    FROM RankedData cur
    LEFT JOIN RankedData prev
        ON prev.MerchantKey = cur.MerchantKey
        AND (
            (prev.Year = cur.Year AND prev.Month = cur.Month - 1)
            OR (prev.Year = cur.Year - 1 AND cur.Month = 1 AND prev.Month = 12)
        )
),

LatestMonth AS
(
    SELECT
        MAX(Year * 100 + Month) AS MaxYearMonth
    FROM MonthlyData
)
```

```sql
SELECT *
FROM WithPrev
WHERE (Year * 100 + Month) = (SELECT MaxYearMonth FROM LatestMonth)
ORDER BY CurrentRevenue DESC;


-- Create a derived table of merchants for current and previous month ranks by reporting date. Share
--your SQL query used to build the derived table. SQL is a major job skill for this position, be sure to
--emphasize your talent.
;WITH Latest AS (
    SELECT
        MAX(d.Year * 100 + d.Month) AS MaxYM
    FROM Dim_Date d
),
LatestMonth AS (
    SELECT
        (MaxYM / 100) AS LatestYear,
        (MaxYM % 100) AS LatestMonth
    FROM Latest
),
PreviousMonth AS (
    SELECT
        CASE
            WHEN LatestMonth = 1 THEN LatestYear - 1
            ELSE LatestYear
        END AS PrevYear,
        CASE
            WHEN LatestMonth = 1 THEN 12
            ELSE LatestMonth - 1
        END AS PrevMonth
    FROM LatestMonth
),
CurrentRanks AS (
    SELECT
        dm.MerchantKey,
        dm.MerchantName,
        RANK() OVER (ORDER BY SUM(fi.InterchangeRevenue) DESC) AS CurrentRank
    FROM Fact_Interchange fi
    JOIN Dim_Merchant dm ON fi.MerchantKey = dm.MerchantKey
    JOIN Dim_Date d ON fi.DateKey = d.DateKey
    CROSS JOIN LatestMonth lm
    WHERE d.Year = lm.LatestYear
      AND d.Month = lm.LatestMonth
    GROUP BY dm.MerchantKey, dm.MerchantName
),
PreviousRanks AS (
    SELECT
        dm.MerchantKey,
        RANK() OVER (ORDER BY SUM(fi.InterchangeRevenue) DESC) AS PreviousRank
    FROM Fact_Interchange fi
```

```
    JOIN Dim_Merchant dm ON fi.MerchantKey = dm.MerchantKey
    JOIN Dim_Date d ON fi.DateKey = d.DateKey
    CROSS JOIN PreviousMonth pm
    WHERE d.Year = pm.PrevYear
      AND d.Month = pm.PrevMonth
    GROUP BY dm.MerchantKey
)

SELECT
    c.MerchantKey,
    c.MerchantName,
    c.CurrentRank,
    p.PreviousRank
FROM CurrentRanks c
LEFT JOIN PreviousRanks p
    ON c.MerchantKey = p.MerchantKey
ORDER BY c.CurrentRank;
```