```
/* REQUIREMENT 1 */
CREATE TABLE FactSales (
    CustomerKey NUMBER(10) NOT NULL,
    LocationKey NUMBER(10) NOT NULL,
    ProductKey NUMBER(10) NOT NULL,
    SalespersonKey NUMBER(10) NOT NULL,
    SupplierKey NUMBER(10) NOT NULL,  -- Add SupplierKey column
    DateKey NUMBER(8) NOT NULL,
    Quantity NUMBER(4) NOT NULL,
    UnitPrice NUMBER(18,2) NOT NULL,
    TaxRate NUMBER(18,3) NOT NULL,
    TotalBeforeTax NUMBER(18,2) NOT NULL,
    TotalAfterTax NUMBER(18,2) NOT NULL,

    -- Add foreign key constraints
    CONSTRAINT FK_FactSales_CustomerKey FOREIGN KEY (CustomerKey)
REFERENCES DimCustomers(CustomerKey),
    CONSTRAINT FK_FactSales_LocationKey FOREIGN KEY (LocationKey)
REFERENCES DimLocation(LocationKey),
    CONSTRAINT FK_FactSales_ProductKey FOREIGN KEY (ProductKey)
REFERENCES DimProducts(ProductKey),
    CONSTRAINT FK_FactSales_SalespersonKey FOREIGN KEY (SalespersonKey)
REFERENCES DimSalesPeople(SalespersonKey),
    CONSTRAINT FK_FactSales_SupplierKey FOREIGN KEY (SupplierKey)
REFERENCES DimSupplier(SupplierKey),  -- Add SupplierKey constraint
    CONSTRAINT FK_FactSales_DateKey FOREIGN KEY (DateKey) REFERENCES
DimDate(DateKey),

    -- Primary key constraint
    CONSTRAINT PK_FactSales PRIMARY KEY (CustomerKey, LocationKey,
ProductKey, SalespersonKey, DateKey)
);


drop table FactSales;

CREATE INDEX IX_FactSales_CustomerKey ON FactSales(CustomerKey);
CREATE INDEX IX_FactSales_CityKey ON FactSales(LocationKey);
CREATE INDEX IX_FactSales_ProductKey ON FactSales(ProductKey);
CREATE INDEX IX_FactSales_SalespersonKey ON FactSales(SalespersonKey);
CREATE INDEX IX_FactSales_DateKey ON FactSales(DateKey);

CREATE TABLE DimDate (
DateKey NUMBER(10) NOT NULL,
DateValue DATE NOT NULL,
CYear NUMBER(10) NOT NULL,
CQtr NUMBER(1) NOT NULL,
CMonth NUMBER(2) NOT NULL,
DayNo NUMBER(2) NOT NULL,
StartOfMonth DATE NOT NULL,
EndOfMonth DATE NOT NULL,
MonthName VARCHAR2(9) NOT NULL,
DayOfWeekName VARCHAR2(9) NOT NULL,
CONSTRAINT PK_DimDate PRIMARY KEY ( DateKey )
```

```
);

CREATE TABLE DimLocation(
LocationKey NUMBER(10),
CityName NVARCHAR2(50) NULL,
StateProvCode NVARCHAR2(5) NULL,
StateProvName NVARCHAR2(50) NULL,
CountryName NVARCHAR2(60) NULL,
CountryFormalName NVARCHAR2(60) NULL,
CONSTRAINT PK_DimLocation PRIMARY KEY ( LocationKey )
);


CREATE TABLE DimSalesPeople(
SalespersonKey NUMBER(10),
FullName NVARCHAR2(50) NULL,
PreferredName NVARCHAR2(50) NULL,
LogonName NVARCHAR2(50) NULL,
PhoneNumber NVARCHAR2(20) NULL,
FaxNumber NVARCHAR2(20) NULL,
EmailAddress NVARCHAR2(256) NULL,
CONSTRAINT PK_DimSalesPeople PRIMARY KEY (SalespersonKey )
);

CREATE TABLE DimProducts(
ProductKey NUMBER(10),
ProductName NVARCHAR2(100) NULL,
ProductColour NVARCHAR2(20) NULL,
ProductBrand NVARCHAR2(50) NULL,
ProductSize NVARCHAR2(20) NULL,
StartDate DATE NOT NULL,
EndDate DATE NULL,
CONSTRAINT PK_DimProducts PRIMARY KEY ( ProductKey )
);

CREATE TABLE DimCustomers(
CustomerKey NUMBER(10),
CustomerName NVARCHAR2(100) NULL,
CustomerCategoryName NVARCHAR2(50) NULL,
DeliveryCityName NVARCHAR2(50) NULL,
DeliveryStateProvCode NVARCHAR2(5) NULL,
DeliveryCountryName NVARCHAR2(50) NULL,
PostalCityName NVARCHAR2(50) NULL,
PostalStateProvCode NVARCHAR2(5) NULL,
PostalCountryName NVARCHAR2(50) NULL,
StartDate DATE NOT NULL,
EndDate DATE NULL,
CONSTRAINT PK_DimCustomers PRIMARY KEY ( CustomerKey )
);

CREATE TABLE DimSupplier (
    SupplierKey INT PRIMARY KEY,
    FullName VARCHAR(100),
    PhoneNumber VARCHAR(20),
    FaxNumber VARCHAR(20),
    WebsiteUrl VARCHAR(100),
    ValidFrom DATE,
    ValidTo DATE,
    CurrentFlag CHAR(1) DEFAULT 'Y', -- 'Y' if the row is current, 'N' if
not
    CONSTRAINT UK_Supplier_BusinessKey UNIQUE (FullName)
);

CREATE INDEX IX_Supplier_CurrentFlag ON DimSupplier(CurrentFlag);

CREATE TABLE DimCityPreferences (
    CityKey INT PRIMARY KEY,
    BrandPreference VARCHAR(100),
    ColourPreference VARCHAR(50),
    PricePreference VARCHAR(50)
);

CREATE TABLE DimSupplierPerformance (
    SupplierKey INT PRIMARY KEY,
    TotalSalesAmount DECIMAL(18, 2),
    TotalOrdersProcessed INT,
    TopSellingProducts VARCHAR(255),
    TopCustomerLocations VARCHAR(255)
);

/* REQUIREMENT 2 */

CREATE OR REPLACE PROCEDURE DimDate_Load (DateValue IN DATE)
IS
    -- Variables to store start and end dates
    StartDate DATE := TO_DATE('2012-01-01', 'YYYY-MM-DD');
    EndDate DATE := ADD_MONTHS(StartDate, 12 * 5); -- Add 5 years

    -- Current date variable
    CurrentDate DATE := StartDate;
BEGIN
    -- Loop to insert date values into DimDate table
    WHILE CurrentDate <= EndDate LOOP
        INSERT INTO DimDate (DateKey, DateValue, CYear, CQtr, CMonth,
DayNo, StartOfMonth, EndOfMonth, MonthName, DayOfWeekName)
        VALUES (
            EXTRACT(YEAR FROM CurrentDate) * 10000 + EXTRACT(MONTH FROM
CurrentDate) * 100 + EXTRACT(DAY FROM CurrentDate),
            CurrentDate,
            EXTRACT(YEAR FROM CurrentDate),
            TO_NUMBER(TO_CHAR(CurrentDate, 'Q')),
            EXTRACT(MONTH FROM CurrentDate),
            EXTRACT(DAY FROM CurrentDate),
            TRUNC(CurrentDate, 'MM'),
            LAST_DAY(CurrentDate),
            TO_CHAR(CurrentDate, 'MONTH'),
            TO_CHAR(CurrentDate, 'DY')
        );
        -- Move to the next date
        CurrentDate := CurrentDate + 1; -- Increment by one day
    END LOOP;
END;

/* REQUIREMENT 3 */

SELECT
    dc.CustomerName AS customer_name,
    dl.CityName AS city_name,
    dsp.FullName AS salesperson_name,
    dp.ProductName AS product_name,
    ds.FullName AS supplier_name,
    dd.DateValue AS order_date,
    fs.Quantity,
    fs.UnitPrice,
    fs.TotalBeforeTax,
    fs.TotalAfterTax,
    -- Sales performance metrics
    SUM(fs.TotalBeforeTax) OVER (PARTITION BY dp.ProductName) AS
total_sales_amount,
    AVG(fs.UnitPrice) OVER (PARTITION BY dp.ProductName) AS
average_unit_price,
    -- Segmentation analysis
    CASE
        WHEN dc.CustomerCategoryName = 'High Value' THEN 'High Value
Customer'
        WHEN dc.CustomerCategoryName = 'Mid Value' THEN 'Mid Value
Customer'
        ELSE 'Low Value Customer'
    END AS customer_segment,
    -- Trend analysis
    LAG(fs.TotalBeforeTax) OVER (ORDER BY dd.DateValue) AS
previous_month_sales,
    LEAD(fs.TotalBeforeTax) OVER (ORDER BY dd.DateValue) AS
next_month_sales
FROM
    FactSales fs
JOIN
    DimCustomers dc ON fs.CustomerKey = dc.CustomerKey
JOIN
    DimLocation dl ON fs.LocationKey = dl.LocationKey
JOIN
    DimSalesPeople dsp ON fs.SalespersonKey = dsp.SalespersonKey
JOIN
    DimProducts dp ON fs.ProductKey = dp.ProductKey
JOIN
    DimDate dd ON fs.DateKey = dd.DateKey
JOIN
    DimSupplier ds ON fs.SupplierKey = ds.SupplierKey;

/* REQUIREMENT 4 */
CREATE TABLE StageCustomers (
    CustomerID INT,
    CustomerName NVARCHAR2(100),
    CategoryName NVARCHAR2(50),
    City NVARCHAR2(100),
    State NVARCHAR2(100),
    Country NVARCHAR2(100)
);

CREATE TABLE StageSalespeople (
    SalespersonID NUMBER,
    SalespersonName NVARCHAR2(100)
);

CREATE TABLE StageOrders (
    OrderID NUMBER,
    OrderDate DATE,
    CustomerID NUMBER,
    SalespersonID NUMBER
);

CREATE TABLE StageSuppliers (
    SupplierID NUMBER,
    SupplierName NVARCHAR2(100),
    SupplierCategoryName NVARCHAR2(50)
);

CREATE OR REPLACE PROCEDURE Customers_Extract AS
    RowCt NUMBER(10);
    v_sql VARCHAR(255);
BEGIN
    -- Truncate staging table
    v_sql := 'TRUNCATE TABLE Customers_Stage';
    EXECUTE IMMEDIATE v_sql;

    -- Insert data into staging table
    INSERT INTO Customers_Stage
    WITH CityDetails AS (
        SELECT ci.CityID,
               ci.CityName,
               sp.StateProvinceCode,
               sp.StateProvinceName,
               co.CountryName,
               co.FormalName
        FROM Cities ci
        LEFT JOIN StateProvinces sp ON ci.StateProvinceID =
sp.StateProvinceID
        LEFT JOIN Countries co ON sp.CountryID = co.CountryID
    )
    SELECT cust.CustomerName,
           cat.CustomerCategoryName,
           dc.CityName AS DeliveryCityName,
           dc.StateProvinceCode AS DeliveryStateProvinceCode,
           dc.StateProvinceName AS DeliveryStateProvinceName,
           dc.CountryName AS DeliveryCountryName,
           dc.FormalName AS DeliveryFormalName,
           pc.CityName AS PostalCityName,
           pc.StateProvinceCode AS PostalStateProvinceCode,
           pc.StateProvinceName AS PostalStateProvinceName,
           pc.CountryName AS PostalCountryName,
           pc.FormalName AS PostalFormalName
    FROM Customers cust
    LEFT JOIN CustomerCategories cat ON cust.CustomerCategoryID = 
cat.CustomerCategoryID
    LEFT JOIN CityDetails dc ON cust.DeliveryCityID = dc.CityID
    LEFT JOIN CityDetails pc ON cust.PostalCityID = pc.CityID;

    -- Check if records were inserted
    RowCt := SQL%ROWCOUNT;
    IF SQL%NOTFOUND THEN
        DBMS_OUTPUT.PUT_LINE('No records found. Check with source 
system.');
    ELSIF SQL%FOUND THEN
        DBMS_OUTPUT.PUT_LINE(RowCt || ' Rows have been inserted!');
    END IF;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(SQLERRM);
        DBMS_OUTPUT.PUT_LINE(v_sql);
END;

CREATE OR REPLACE PROCEDURE Products_Extract AS
    RowCt NUMBER(10);
    v_sql VARCHAR(255);
BEGIN
    -- Truncate staging table
    v_sql := 'TRUNCATE TABLE Products_Stage';
    EXECUTE IMMEDIATE v_sql;

    -- Insert data into staging table
    INSERT INTO Products_Stage
    SELECT si.StockItemID,
           si.StockItemName AS ProductName,
           c.ColorName AS Color
    FROM Warehouse.StockItems si
    JOIN Warehouse.Colors c ON si.ColorID = c.ColorID;

    -- Check if records were inserted
    RowCt := SQL%ROWCOUNT;
    IF SQL%NOTFOUND THEN
        DBMS_OUTPUT.PUT_LINE('No records found. Check with source 
system.');
    ELSIF SQL%FOUND THEN
        DBMS_OUTPUT.PUT_LINE(RowCt || ' Rows have been inserted!');
    END IF;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(SQLERRM);
        DBMS_OUTPUT.PUT_LINE(v_sql);
END;
```

```
/

CREATE OR REPLACE PROCEDURE Salespeople_Extract AS
    RowCt NUMBER(10);
    v_sql VARCHAR(255);
BEGIN
    -- Truncate staging table
    v_sql := 'TRUNCATE TABLE Salespeople_Stage';
    EXECUTE IMMEDIATE v_sql;

    -- Insert data into staging table
    INSERT INTO Salespeople_Stage
    SELECT PersonID AS SalespersonID,
           FullName AS SalespersonName
    FROM Application.People
    WHERE IsSalesperson = 1;

    -- Check if records were inserted
    RowCt := SQL%ROWCOUNT;
    IF SQL%NOTFOUND THEN
        DBMS_OUTPUT.PUT_LINE('No records found. Check with source
system.');
    ELSIF SQL%FOUND THEN
        DBMS_OUTPUT.PUT_LINE(RowCt || ' Rows have been inserted!');
    END IF;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(SQLERRM);
        DBMS_OUTPUT.PUT_LINE(v_sql);
END;

drop procedure Orders_Extract;
CREATE OR REPLACE PROCEDURE Orders_Extract
AS
    RowCt NUMBER(10);
    v_sql VARCHAR(255) := 'TRUNCATE TABLE wwidmuser.Orders_Stage DROP
STORAGE';
BEGIN
    EXECUTE IMMEDIATE v_sql;

    INSERT INTO wwidmuser.Orders_Stage
    WITH CityDetails AS (
        SELECT ci.CityID,
               ci.CityName,
               sp.StateProvinceCode,
               sp.StateProvinceName,
               co.CountryName,
               co.FormalName
        FROM wwidbuser.Cities ci
        LEFT JOIN wwidbuser.StateProvinces sp
            ON ci.StateProvinceID = sp.StateProvinceID
        LEFT JOIN wwidbuser.Countries co
            ON sp.CountryID = co.CountryID
    )
    SELECT o.OrderDate
        ,ol.Quantity
        ,ol.UnitPrice
        ,ol.TaxRate
        ,c.CustomerName
        ,dc.cityname
        ,dc.stateprovincename
        ,dc.countryname
        ,stk.StockItemName
        ,p.LogonName
    FROM wwidbuser.Orders o
        LEFT JOIN wwidbuser.OrderLines ol
            ON o.OrderID = ol.OrderID
        LEFT JOIN wwidbuser.customers c
            ON o.CustomerID = c.CustomerID
        LEFT JOIN CityDetails dc
            ON c.DeliveryCityID = dc.CityID
        LEFT JOIN wwidbuser.stockitems stk
            ON ol.Stockitemid = stk.StockItemID
        LEFT JOIN wwidbuser.People p
            ON o.salespersonpersonid = p.personid AND IsSalesPerson = 1;

    RowCt := SQL%ROWCOUNT;
    IF sql%notfound THEN
       dbms_output.put_line('No records found. Check with source
system.');
    ELSIF sql%found THEN
       dbms_output.put_line(TO_CHAR(RowCt) ||' Rows have been
inserted!');
    END IF;

  EXCEPTION
    WHEN OTHERS THEN
       dbms_output.put_line(SQLERRM);
       dbms_output.put_line(v_sql);
END;


CREATE OR REPLACE PROCEDURE Suppliers_Extract AS
    RowCt NUMBER(10);
    v_sql VARCHAR(255);
BEGIN
    -- Truncate staging table
    v_sql := 'TRUNCATE TABLE Suppliers_Stage';
    EXECUTE IMMEDIATE v_sql;

    -- Insert data into staging table
    INSERT INTO Suppliers_Stage
    SELECT s.SupplierID,
           s.SupplierName,
           sc.SupplierCategoryName
    FROM Suppliers s
    JOIN SupplierCategories sc ON s.SupplierCategoryID =
sc.SupplierCategoryID;

    -- Check if records were inserted
    RowCt := SQL%ROWCOUNT;
    IF SQL%NOTFOUND THEN
        DBMS_OUTPUT.PUT_LINE('No records found. Check with source
system.');
    ELSIF SQL%FOUND THEN
        DBMS_OUTPUT.PUT_LINE(RowCt || ' Rows have been inserted!');
    END IF;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(SQLERRM);
        DBMS_OUTPUT.PUT_LINE(v_sql);
END;

BEGIN
    Customers_Extract;
END;

BEGIN
    Salespeople_Extract;
END;

BEGIN
    Products_Extract;
END;

BEGIN
    Orders_Extract(TO_DATE('2013-01-01', 'YYYY-MM-DD'));
END;

BEGIN
    Suppliers_Extract;
END;



/ REQUIREMENT 5 */


CREATE TABLE Customers_Preload (
   CustomerKey NUMBER(10) NOT NULL,
   CustomerName NVARCHAR2(100) NULL,
   CustomerCategoryName NVARCHAR2(50) NULL,
   DeliveryCityName NVARCHAR2(50) NULL,
   DeliveryStateProvCode NVARCHAR2(5) NULL,
   DeliveryCountryName NVARCHAR2(50) NULL,
   PostalCityName NVARCHAR2(50) NULL,
   PostalStateProvCode NVARCHAR2(5) NULL,
   PostalCountryName NVARCHAR2(50) NULL,
   StartDate DATE NOT NULL,
   EndDate DATE NULL,
   CONSTRAINT PK_Customers_Preload PRIMARY KEY ( CustomerKey )
);

CREATE SEQUENCE LocationKey START WITH 1;
CREATE SEQUENCE CustomerKey START WITH 1;

CREATE TABLE Location_Preload (
    LocationKey NUMBER(10) NOT NULL,
    CityName NVARCHAR2(50) NULL,
    StateProvCode NVARCHAR2(5) NULL,
    StateProvName NVARCHAR2(50) NULL,
    CountryName NVARCHAR2(60) NULL,
    CountryFormalName NVARCHAR2(60) NULL,
    CONSTRAINT PK_Location_Preload PRIMARY KEY (LocationKey)
);

CREATE OR REPLACE PROCEDURE Locations_Transform
AS
  RowCt NUMBER(10);
  v_sql VARCHAR(255) := 'TRUNCATE TABLE Locations_Preload DROP STORAGE';
BEGIN
    EXECUTE IMMEDIATE v_sql;
    INSERT INTO Location_Preload /* Column list excluded for brevity */
    SELECT LocationKey.NEXTVAL AS LocationKey,
           cu.DeliveryCityName,
           cu.DeliveryStateProvinceCode,
           cu.DeliveryStateProvinceName,
           cu.DeliveryCountryName,
           cu.DeliveryFormalName
    FROM Customers_Stage cu
    WHERE NOT EXISTS
      ( SELECT 1
              FROM DimLocation ci
              WHERE cu.DeliveryCityName = ci.CityName
                AND cu.DeliveryStateProvinceName = ci.StateProvName
                AND cu.DeliveryCountryName = ci.CountryName
        );

    INSERT INTO Location_Preload /* Column list excluded for brevity */
    SELECT ci.LocationKey,
           cu.DeliveryCityName,
           cu.DeliveryStateProvinceCode,
           cu.DeliveryStateProvinceName,
           cu.DeliveryCountryName,
           cu.DeliveryFormalName
    FROM Customers_Stage cu
    JOIN DimLocation ci
        ON cu.DeliveryCityName = ci.CityName
        AND cu.DeliveryStateProvinceName = ci.StateProvName
        AND cu.DeliveryCountryName = ci.CountryName;

    RowCt := SQL%ROWCOUNT;
    IF sql%notfound THEN
```

```
       dbms_output.put_line('No records found. Check with source
system.');
    ELSIF sql%found THEN
       dbms_output.put_line(TO_CHAR(RowCt) ||' Rows have been
inserted!');
    END IF;

  EXCEPTION
    WHEN OTHERS THEN
       dbms_output.put_line(SQLERRM);
       dbms_output.put_line(v_sql);
END;

SET SERVEROUT ON;
TRUNCATE TABLE Location_Preload;
EXECUTE Locations_Transform;
SELECT COUNT(*) FROM location_preload;


CREATE OR REPLACE PROCEDURE Customers_Transform
AS
  RowCt NUMBER(10);
  v_sql VARCHAR(255) := 'TRUNCATE TABLE Customers_Preload DROP STORAGE';
  StartDate DATE := SYSDATE;
  EndDate DATE := SYSDATE - 1;
BEGIN
    EXECUTE IMMEDIATE v_sql;
 --BEGIN TRANSACTION;
 -- Add updated records
    INSERT INTO Customers_Preload /* Column list excluded for brevity */
    SELECT CustomerKey.NEXTVAL AS CustomerKey,
           stg.CustomerName,
           stg.CustomerCategoryName,
           stg.DeliveryCityName,
           stg.DeliveryStateProvinceCode,
           stg.DeliveryCountryName,
           stg.PostalCityName,
           stg.PostalStateProvinceCode,
           stg.PostalCountryName,
           StartDate,
           NULL
    FROM Customers_Stage stg
    JOIN DimCustomers cu
        ON stg.CustomerName = cu.CustomerName AND cu.EndDate IS NULL
    WHERE stg.CustomerCategoryName <> cu.CustomerCategoryName
          OR stg.DeliveryCityName <> cu.DeliveryCityName
          OR stg.DeliveryStateProvinceCode <> cu.DeliveryStateProvCode
          OR stg.DeliveryCountryName <> cu.DeliveryCountryName
          OR stg.PostalCityName <> cu.PostalCityName
          OR stg.PostalStateProvinceCode <> cu.PostalStateProvCode
          OR stg.PostalCountryName <> cu.PostalCountryName;

    -- Add existing records, and expire as necessary
    INSERT INTO Customers_Preload /* Column list excluded for brevity */
    SELECT cu.CustomerKey,
           cu.CustomerName,
           cu.CustomerCategoryName,
           cu.DeliveryCityName,
           cu.DeliveryStateProvCode,
           cu.DeliveryCountryName,
           cu.PostalCityName,
           cu.PostalStateProvCode,
           cu.PostalCountryName,
           cu.StartDate,
           CASE
               WHEN pl.CustomerName IS NULL THEN NULL
               ELSE cu.EndDate
           END AS EndDate
    FROM DimCustomers cu
    LEFT JOIN Customers_Preload pl
        ON pl.CustomerName = cu.CustomerName
        AND cu.EndDate IS NULL;
 -- Create new records
    INSERT INTO Customers_Preload /* Column list excluded for brevity */
    SELECT CustomerKey.NEXTVAL AS CustomerKey,
           stg.CustomerName,
           stg.CustomerCategoryName,
           stg.DeliveryCityName,
           stg.DeliveryStateProvinceCode,
           stg.DeliveryCountryName,
           stg.PostalCityName,
           stg.PostalStateProvinceCode,
           stg.PostalCountryName,
           StartDate,
           NULL
    FROM Customers_Stage stg
    WHERE NOT EXISTS ( SELECT 1 FROM DimCustomers cu WHERE
stg.CustomerName = cu.CustomerName );
    -- Expire missing records
    INSERT INTO Customers_Preload /* Column list excluded for brevity */
    SELECT cu.CustomerKey,
           cu.CustomerName,
           cu.CustomerCategoryName,
           cu.DeliveryCityName,
           cu.DeliveryStateProvCode,
           cu.DeliveryCountryName,
           cu.PostalCityName,
           cu.PostalStateProvCode,
           cu.PostalCountryName,
           cu.StartDate,
           EndDate
    FROM DimCustomers cu
    WHERE NOT EXISTS ( SELECT 1 FROM Customers_Stage stg WHERE
stg.CustomerName = cu.CustomerName )
          AND cu.EndDate IS NULL;

    RowCt := SQL%ROWCOUNT;
    dbms_output.put_line(TO_CHAR(RowCt) ||' Rows have been inserted!');
```

```
--COMMIT TRANSACTION;
  EXCEPTION
    WHEN OTHERS THEN
       dbms_output.put_line(SQLERRM);
       dbms_output.put_line(v_sql);
END;

CREATE TABLE PreLoad_DimCustomers (
    CustomerID NUMBER, -- Business Key
    CustomerName VARCHAR2(100),
    -- Add other attributes as needed
    EffectiveStartDate DATE,
    EffectiveEndDate DATE,
    IsCurrent CHAR(1)
);

CREATE TABLE PreLoad_DimProducts (
    ProductID NUMBER, -- Business Key
    ProductName VARCHAR2(100),
    -- Add other attributes as needed
    EffectiveStartDate DATE,
    EffectiveEndDate DATE,
    IsCurrent CHAR(1)
);

CREATE TABLE PreLoad_DimSalespeople (
    SalespersonID NUMBER, -- Business Key
    SalespersonName VARCHAR2(100),
    -- Add other attributes as needed
    EffectiveStartDate DATE,
    EffectiveEndDate DATE,
    IsCurrent CHAR(1)
);

CREATE TABLE PreLoad_DimSuppliers (
    SupplierID NUMBER, -- Business Key
    SupplierName VARCHAR2(100),
    SupplierCategoryName VARCHAR2(50),
    -- Add other attributes as needed
    EffectiveStartDate DATE,
    EffectiveEndDate DATE,
    IsCurrent CHAR(1)
);

CREATE OR REPLACE PROCEDURE Transform_SCD1_Dimensions AS
BEGIN
    -- Update existing records
    UPDATE DimCustomers dc
    SET dc.CustomerName = (
        SELECT pldc.CustomerName
        FROM PreLoad_DimCustomers pldc
        WHERE pldc.CustomerID = dc.CustomerID
    )
    WHERE EXISTS (
        SELECT 1
        FROM PreLoad_DimCustomers pldc
        WHERE pldc.CustomerID = dc.CustomerID
    );

    -- Insert new records
    INSERT INTO DimCustomers (CustomerID, CustomerName,
EffectiveStartDate, EffectiveEndDate, IsCurrent)
    SELECT pldc.CustomerID, pldc.CustomerName, SYSDATE, NULL, 'Y'
    FROM PreLoad_DimCustomers pldc
    WHERE NOT EXISTS (
        SELECT 1
        FROM DimCustomers dc
        WHERE dc.CustomerID = pldc.CustomerID
    );

    -- Similar transformations for other SCD Type 1 dimensions
END;

CREATE OR REPLACE PROCEDURE Transform_SCD2_Dimensions AS
BEGIN
    -- Handle updates
    FOR pldsup IN (SELECT * FROM PreLoad_DimSuppliers)
    LOOP
        IF pldsup.SupplierID IS NOT NULL THEN
            -- Check if the supplier exists in the dimension
            IF EXISTS (
                SELECT 1
                FROM DimSuppliers ds
                WHERE ds.SupplierID = pldsup.SupplierID
            ) THEN
                -- Expire the existing record
                UPDATE DimSuppliers ds
                SET ds.EffectiveEndDate = SYSDATE,
                    ds.IsCurrent = 'N'
                WHERE ds.SupplierID = pldsup.SupplierID
                AND ds.IsCurrent = 'Y';

                -- Insert the updated record
                INSERT INTO DimSuppliers (SupplierID, SupplierName,
SupplierCategoryName, EffectiveStartDate, EffectiveEndDate, IsCurrent)
                VALUES (pldsup.SupplierID, pldsup.SupplierName,
pldsup.SupplierCategoryName, SYSDATE, NULL, 'Y');
            ELSE
                -- Insert new record if not exists
                INSERT INTO DimSuppliers (SupplierID, SupplierName,
SupplierCategoryName, EffectiveStartDate, EffectiveEndDate, IsCurrent)
                VALUES (pldsup.SupplierID, pldsup.SupplierName,
pldsup.SupplierCategoryName, SYSDATE, NULL, 'Y');
            END IF;
        END IF;
    END LOOP;
END;
```

```
/* REQUIREMENT 6 */

-- Create a savepoint for the transaction
SAVEPOINT START_TRANSACTION;

-- Create stored procedure to load dimensions and fact table
CREATE OR REPLACE PROCEDURE Load_Dimensions_And_Fact AS
BEGIN
    -- Start transaction
    BEGIN
        -- Load changed records into dimension tables
        -- Example for DimCustomers
        MERGE INTO DimCustomers dc
        USING PreLoad_DimCustomers pldc
        ON (dc.CustomerID = pldc.CustomerID)
        WHEN MATCHED THEN
            UPDATE SET
                dc.CustomerName = pldc.CustomerName,
                dc.CustomerCategoryName = pldc.CustomerCategoryName,
                dc.DeliveryCityName = pldc.DeliveryCityName,
                dc.DeliveryStateProvCode = pldc.DeliveryStateProvCode,
                dc.DeliveryCountryName = pldc.DeliveryCountryName,
                dc.PostalCityName = pldc.PostalCityName,
                dc.PostalStateProvCode = pldc.PostalStateProvCode,
                dc.PostalCountryName = pldc.PostalCountryName
        WHEN NOT MATCHED THEN
            INSERT (CustomerID, CustomerName, CustomerCategoryName,
                    DeliveryCityName, DeliveryStateProvCode,
DeliveryCountryName,
                    PostalCityName, PostalStateProvCode,
PostalCountryName)
            VALUES (pldc.CustomerID, pldc.CustomerName,
pldc.CustomerCategoryName,
                    pldc.DeliveryCityName, pldc.DeliveryStateProvCode,
pldc.DeliveryCountryName,
                    pldc.PostalCityName, pldc.PostalStateProvCode,
pldc.PostalCountryName);

        -- Similar merge operations for other dimension tables

        -- Load data into the fact table
        -- Example for FactSales
        INSERT INTO FactSales (CustomerID, ProductID, SalesAmount,
OrderDate)
        SELECT pdc.CustomerID, pp.ProductID, ol.TotalAmount, o.OrderDate
        FROM PreLoad_DimCustomers pdc
        JOIN PreLoad_Products pp ON pp.ProductName = ol.ProductName
        JOIN PreLoad_Orders o ON o.CustomerID = pdc.CustomerID
        JOIN PreLoad_OrderLines ol ON ol.OrderID = o.OrderID;

        -- Commit transaction if successful
        COMMIT;
    EXCEPTION
        -- Roll back to savepoint if an error occurs
        WHEN OTHERS THEN
            ROLLBACK TO START_TRANSACTION;
            -- Raise an exception or log the error as needed
            RAISE;
    END;
END;

/* REQUIREMENT 7 */

EXEC ExtractOrders @OrderDate = '2013-01-01';
EXEC ExtractOrders @OrderDate = '2013-01-02';
EXEC ExtractOrders @OrderDate = '2013-01-03';
EXEC ExtractOrders @OrderDate = '2013-01-04';

EXEC LoadCustomerDimension;
EXEC LoadProductDimension;
EXEC LoadSalespersonDimension;
EXEC LoadSupplierDimension;
EXEC LoadDateDimension;
-- Load fact table
EXEC LoadSalesFactTable;


SELECT
    dc.CustomerName AS Customer,
    dl.CityName AS City,
    dsp.FullName AS Salesperson,
    dp.ProductName AS Product,
    ds.FullName AS Supplier,
    dd.DateValue AS OrderDate,
    fs.TotalAfterTax AS TotalSales
FROM
    FactSales fs
JOIN
    DimCustomers dc ON fs.CustomerKey = dc.CustomerKey
JOIN
    DimLocation dl ON dl.LocationKey = dl.LocationKey
JOIN
    DimSalesPeople dsp ON fs.SalespersonKey = dsp.SalespersonKey
JOIN
    DimProducts dp ON fs.ProductKey = dp.ProductKey
JOIN
    DimSupplier ds ON fs.SupplierKey = ds.SupplierKey
JOIN
    DimDate dd ON fs.DateKey = dd.DateKey
WHERE
    dd.DateValue BETWEEN TO_DATE('2013-01-01', 'YYYY-MM-DD') AND
TO_DATE('2013-01-04', 'YYYY-MM-DD')
ORDER BY
    dd.DateValue;
```