```sql
Create Database Walmart_Sales_Database;

USE Walmart_Sales_Database;

CREATE TABLE Store (
    Store INT,
    Date DATE,
    CPI FLOAT,
    Weekly_Sales FLOAT
);

CREATE TABLE Holiday (
    Holiday INT,
    Fuel_Price FLOAT,
    Unemployment FLOAT,
    Temperature FLOAT
);

SELECT * FROM Store;
SELECT * FROM Holiday;

DROP TABLE Holiday;

# Calculate the average fuel price from the Holiday table?
SELECT AVG(Fuel_Price) AS AvgFuelPrice
FROM Holiday;

# Find the total sum of weekly sales from the Store table.
SELECT SUM(Weekly_Sales) AS TotalWeeklySales
FROM Store;

# Extract the year from the "Date" column in the Store table.
SELECT YEAR(Date) AS Year
FROM Store;

# Calculate the number of days between two given dates in the Store table.
SELECT DATEDIFF(end_date, start_date) AS NumberOfDays
FROM (
    SELECT MIN(Date) AS start_date, MAX(Date) AS end_date
    FROM Store
) AS date_range;

# What is the maximum value of CPI in the Store table?
SELECT MAX(CPI) AS MaxCPI
FROM Store;

# Calculate the total number of holidays in the Holiday table.
SELECT COUNT(*) AS TotalHolidays
FROM Holiday;

# Rank stores based on their weekly sales.
SELECT Store, Weekly_Sales,
```

```
       RANK() OVER (ORDER BY Weekly_Sales DESC) AS SalesRank
FROM Store;
# Calculate the cumulative sum of weekly sales over time for each store.
SELECT Store, Date, Weekly_Sales,
       SUM(Weekly_Sales) OVER (PARTITION BY Store ORDER BY Date) AS
CumulativeSales
FROM Store;

# Classify each store's weekly sales into categories like "low," "medium,"
or "high." SELECT Store,
       CASE
           WHEN Weekly_Sales < 5000 THEN 'Low'
           WHEN Weekly_Sales >= 5000 AND Weekly_Sales < 10000 THEN
'Medium'
           ELSE 'High'
       END AS SalesCategory
FROM Store;

# Retrieve the weekly sales and fuel prices for each store on a given date
SELECT s.Store, s.Date, s.Weekly_Sales, h.Fuel_Price
FROM Store s
JOIN Holiday h ON s.Date = h.Holiday;

# Combine information from the Store and Holiday tables to analyze weekly
sales during holidays.
SELECT s.Store, s.Date, s.Weekly_Sales, h.Holiday, h.Fuel_Price,
h.Unemployment, h.Temperature
FROM Store s
JOIN Holiday h ON s.Date = h.Holiday;

# Find stores with weekly sales higher than the average weekly sales
across all stores.
SELECT Store, AVG(Weekly_Sales) AS AvgWeeklySales
FROM Store
GROUP BY Store;

SELECT Store, Weekly_Sales
FROM Store
WHERE Weekly_Sales > (SELECT AVG(Weekly_Sales) FROM Store);

# Identify holidays with fuel prices higher than the average fuel price.
SELECT Holiday, Fuel_Price
FROM Holiday
WHERE Fuel_Price > (SELECT AVG(Fuel_Price) FROM Holiday);

# Create a CTE to calculate the total weekly sales for each store.
WITH TotalSalesCTE AS (
    SELECT Store, SUM(Weekly_Sales) AS TotalWeeklySales
    FROM Store
    GROUP BY Store
)
```

```
SELECT *
FROM TotalSalesCTE;
# Use a CTE to find the average unemployment rate during holidays.
WITH HolidayUnemploymentCTE AS (
    SELECT AVG(Unemployment) AS AvgUnemploymentRate
    FROM Holiday
)
SELECT *
FROM HolidayUnemploymentCTE;

# Create a view to display the weekly sales, date, and CPI for each store.
CREATE VIEW WeeklySalesCPIView AS SELECT
s.Store, s.Date, s.Weekly_Sales, s.CPI
FROM Store s
JOIN Holiday h ON s.Date = h.Holiday;

SELECT *
FROM WeeklySalesCPIView;

# Generate a view that combines store and holiday information for easy
analysis.
CREATE VIEW StoreHolidayAnalysis AS
SELECT s.*, h.Holiday, h.Fuel_Price, h.Unemployment, h.Temperature
FROM Store s
JOIN Holiday h ON s.Date = h.Holiday;

SELECT *
FROM StoreHolidayAnalysis;

# Create a stored procedure to calculate the percentage change in CPI over
time DELIMITER //
CREATE PROCEDURE CalculateCPIPercentageChange(
    IN startDate DATE,
    IN endDate DATE
)
BEGIN
    DECLARE startCPI FLOAT;
    DECLARE endCPI FLOAT;
    DECLARE percentageChange FLOAT;

    -- Get CPI for the start date
    SELECT CPI INTO startCPI
    FROM Stores
    WHERE Date = startDate;

    -- Get CPI for the end date
    SELECT CPI INTO endCPI
    FROM Stores
    WHERE Date = endDate;

    -- Calculate percentage change
    SET percentageChange = ((endCPI - startCPI) / startCPI) * 100;
```

```
    -- Output the result
    SELECT CONCAT('Percentage change in CPI from ', startDate, ' to ',
endDate, ': ', percentageChange, '%') AS Result;
END//
DELIMITER ;

# Develop a stored procedure to classify stores into regions based on
their weekly sales.
DELIMITER //

CREATE PROCEDURE ClassifyStoresIntoRegions()
BEGIN
    DECLARE avgWeeklySales FLOAT;

    -- Calculate the average weekly sales for each store
    CREATE TEMPORARY TABLE AvgWeeklySalesPerStore AS
    SELECT Store, AVG(Weekly_Sales) AS AvgWeeklySales
    FROM Store
    GROUP BY Store;

    -- Classify stores into regions based on average weekly sales
    UPDATE AvgWeeklySalesPerStore
    SET Region = CASE
                    WHEN AvgWeeklySales >= 10000 THEN 'High Sales Region'
                    WHEN AvgWeeklySales >= 5000 AND AvgWeeklySales < 10000
THEN 'Medium Sales Region'
                    ELSE 'Low Sales Region'
                END;

    -- Display the classification results
    SELECT * FROM AvgWeeklySalesPerStore;
END //

DELIMITER ;

# Set up a trigger to log changes in weekly sales for auditing purposes.
DELIMITER //

CREATE TRIGGER AuditWeeklySalesUpdate
AFTER UPDATE ON Store
FOR EACH ROW
BEGIN
    IF OLD.Weekly_Sales != NEW.Weekly_Sales THEN
        INSERT INTO WeeklySalesAudit (Store, OldWeeklySales,
NewWeeklySales, ChangeDateTime)
        VALUES (OLD.Store, OLD.Weekly_Sales, NEW.Weekly_Sales, NOW());
END IF; END //

DELIMITER ;

# Apply an index on Date column in the Store table
CREATE INDEX idx_date ON Store (Date);
```