```sql
use online_retail;

CREATE TABLE online_retail_2009_2010 (
    Invoice VARCHAR(20),
    StockCode VARCHAR(20),
    Description TEXT,
    Quantity INT,
    InvoiceDate DATETIME,
    Price FLOAT,
    CustomerID INT,
    Country VARCHAR(50)
);

CREATE TABLE online_retail_2010_2011 (
    Invoice VARCHAR(20),
    StockCode VARCHAR(20),
    Description TEXT,
    Quantity INT,
    InvoiceDate DATETIME,
    Price FLOAT,
    CustomerID INT,
    Country VARCHAR(50)
);

LOAD DATA INFILE ' Add your file path here '

INTO TABLE online_retail_2009_2010
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(Invoice, StockCode, Description, Quantity, InvoiceDate, Price,
CustomerID, Country);

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server
8.0/Uploads/cleaned_online_retail2.csv'
INTO TABLE online_retail_2010_2011
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(Invoice, StockCode, Description, Quantity, InvoiceDate, Price,
CustomerID, Country);


SELECT * FROM online_retail_2009_2010 LIMIT 10;

SELECT * FROM online_retail_2010_2011 LIMIT 10;

CREATE TABLE combined_online_retail AS
SELECT * FROM online_retail_2009_2010
UNION ALL
SELECT * FROM online_retail_2010_2011;

/*Start of RFM Analysis-*/

SET SQL_SAFE_UPDATES = 0;
```

```sql
/*deleting records with customer ID=0 and Quantity and Price <=0*/
delete from combined_online_retail
where CustomerID = 0 or CustomerId is null
or Quantity <=0
or Price <=0;

/*query to see how many records are there*/
SELECT count(*) as total_rows
from combined_online_retail;

/*Setting up the Reference date for the analysis - usually we take up
the recent date*/
set @reference_date = (select max(InvoiceDate) from
combined_online_retail);

/*Creating RFM Metrics Table*/
CREATE TABLE rfm_metrics AS
SELECT
    CustomerID,
    DATEDIFF(@reference_date, MAX(InvoiceDate)) AS Recency,
    COUNT(DISTINCT Invoice) AS Frequency,
    SUM(Price * Quantity) AS Monetary
FROM combined_online_retail
GROUP BY CustomerID;

/*Filling up the RFM Score for the Customers*/
Alter table rfm_metrics
add column R_score int,
add column F_score int,
add column M_score int;

/*Recency Score*/
CREATE TABLE recency_sorted (
    id INT AUTO_INCREMENT PRIMARY KEY,
    CustomerID INT,
    Recency INT
) ENGINE=InnoDB
SELECT
    CustomerID,
    Recency
FROM rfm_metrics
ORDER BY Recency ASC;

SET @total_count = (SELECT COUNT(*) FROM recency_sorted);

ALTER TABLE recency_sorted ADD COLUMN percentile_rank FLOAT;

UPDATE recency_sorted
SET percentile_rank = id / @total_count;

UPDATE rfm_metrics AS r
JOIN recency_sorted AS s ON r.CustomerID = s.CustomerID
SET r.R_Score = CASE
    WHEN s.percentile_rank <= 0.2 THEN 5
    WHEN s.percentile_rank <= 0.4 THEN 4
    WHEN s.percentile_rank <= 0.6 THEN 3
    WHEN s.percentile_rank <= 0.8 THEN 2
    ELSE 1
```

```sql
END;

SELECT CustomerID, Recency, R_Score
FROM rfm_metrics
ORDER BY R_Score DESC
LIMIT 10;

/*Frequency Score*/
CREATE TABLE frequency_sorted (
    id INT AUTO_INCREMENT PRIMARY KEY,
    CustomerID INT,
    Frequency INT
) ENGINE=InnoDB
SELECT
    CustomerID,
    Frequency
FROM rfm_metrics
ORDER BY Frequency DESC; -- High frequency gets high scores

SET @total_count = (SELECT COUNT(*) FROM frequency_sorted);

ALTER TABLE frequency_sorted ADD COLUMN percentile_rank FLOAT;

UPDATE frequency_sorted
SET percentile_rank = id / @total_count;

UPDATE rfm_metrics AS r
JOIN frequency_sorted AS s ON r.CustomerID = s.CustomerID
SET r.F_Score = CASE
    WHEN s.percentile_rank <= 0.2 THEN 1
    WHEN s.percentile_rank <= 0.4 THEN 2
    WHEN s.percentile_rank <= 0.6 THEN 3
    WHEN s.percentile_rank <= 0.8 THEN 4
    ELSE 5
END;

SELECT CustomerID, Frequency, F_Score
FROM rfm_metrics
ORDER BY F_Score DESC
LIMIT 10;


/*Monetary Score*/
CREATE TABLE monetary_sorted (
    id INT AUTO_INCREMENT PRIMARY KEY,
    CustomerID INT,
    Monetary FLOAT
) ENGINE=InnoDB
SELECT
    CustomerID,
    Monetary
FROM rfm_metrics
ORDER BY Monetary DESC; -- High monetary value gets high scores

SET @total_count = (SELECT COUNT(*) FROM monetary_sorted);

ALTER TABLE monetary_sorted ADD COLUMN percentile_rank FLOAT;
```

```sql
UPDATE monetary_sorted
SET percentile_rank = id / @total_count;

UPDATE rfm_metrics AS r
JOIN monetary_sorted AS s ON r.CustomerID = s.CustomerID
SET r.M_Score = CASE
    WHEN s.percentile_rank <= 0.2 THEN 1
    WHEN s.percentile_rank <= 0.4 THEN 2
    WHEN s.percentile_rank <= 0.6 THEN 3
    WHEN s.percentile_rank <= 0.8 THEN 4
    ELSE 5
END;

SELECT CustomerID, Monetary, M_Score
FROM rfm_metrics
ORDER BY M_Score DESC
LIMIT 10;

select count(*) from rfm_metrics;

SELECT * FROM rfm_metrics LIMIT 10;

SELECT CustomerID, Recency, R_Score, Frequency, F_Score, Monetary,
M_Score
FROM rfm_metrics
ORDER BY R_Score DESC, F_Score DESC, M_Score DESC
LIMIT 20;

create table rfm_table as
select *
from rfm_metrics;

select * from rfm_table limit 20;
```