```sql
use sales_delivery;

-- Modifying the Customer table with new constraints

-- Ensure that Customer_Name is not NULL and has a maximum length of 100
characters
-- Also, enforce NOT NULL constraints for Province, Region,
Customer_Segment
-- Set Cust_id as the primary key with a maximum length of 20 characters
ALTER TABLE cust_dimen
MODIFY Customer_Name VARCHAR(100) NOT NULL,
MODIFY Province VARCHAR(50) NOT NULL,
MODIFY Region VARCHAR(50) NOT NULL,
MODIFY Customer_Segment VARCHAR(50) NOT NULL,
MODIFY Cust_id VARCHAR(20) PRIMARY KEY;

-- Modifying the Orders table with new constraints

-- Convert the order_date to a date format
UPDATE orders_dimen
SET order_date = STR_TO_DATE(order_date, '%d-%m-%Y');

-- Modify Order_ID to an integer, order_date to date, and Order_Priority
to VARCHAR(20)
-- Set Ord_id as the primary key with a maximum length of 50 characters
ALTER TABLE orders_dimen
MODIFY COLUMN Order_ID INT,
MODIFY order_date DATE,
MODIFY COLUMN Order_Priority VARCHAR(20),
MODIFY COLUMN Ord_id VARCHAR(50) PRIMARY KEY;

-- Modifying the Products table with new constraints

-- Set Product_Category, Product_Sub_Category, and Prod_id as NOT NULL
-- Set Prod_id as the primary key with a maximum length of 20 characters
ALTER TABLE prod_dimen
MODIFY COLUMN Product_Category VARCHAR(50) NOT NULL,
MODIFY COLUMN Product_Sub_Category VARCHAR(50) NOT NULL,
MODIFY COLUMN Prod_id VARCHAR(20) PRIMARY KEY;

-- Modifying the Shipping table with new constraints

-- Convert the ship_date to a date format
UPDATE shipping_dimen
SET ship_date = STR_TO_DATE(ship_date, '%d-%m-%Y');

-- Modify Order_ID to an integer, Ship_Mode to VARCHAR(50), and ship_date
to date
-- Set Ship_id as the primary key with a maximum length of 20 characters
ALTER TABLE shipping_dimen
MODIFY COLUMN Order_ID INT,
MODIFY COLUMN Ship_Mode VARCHAR(50),
MODIFY COLUMN ship_date DATE,
MODIFY COLUMN Ship_id VARCHAR(20) PRIMARY KEY;
```

```
-- Modifying the Market Fact table with new constraints

-- Set foreign key relationships for Ord_id, Prod_id, Ship_id, and Cust_id
-- Ensure appropriate data types and constraints for various columns
ALTER TABLE market_fact
MODIFY COLUMN Ord_id VARCHAR(50),
MODIFY COLUMN Prod_id VARCHAR(20),
MODIFY COLUMN Ship_id VARCHAR(20),
MODIFY COLUMN Cust_id VARCHAR(20),
MODIFY COLUMN Sales FLOAT,
MODIFY COLUMN Discount FLOAT,
MODIFY COLUMN Order_Quantity FLOAT,
MODIFY COLUMN Profit FLOAT,
MODIFY COLUMN Shipping_Cost FLOAT,
MODIFY COLUMN Product_Base_Margin FLOAT,
MODIFY COLUMN days_taken_for_delivery INT,
ADD FOREIGN KEY (ord_id) REFERENCES orders_dimen(ord_id),
ADD FOREIGN KEY (prod_id) REFERENCES prod_dimen(prod_id),
ADD FOREIGN KEY (ship_id) REFERENCES shipping_dimen(ship_id),
ADD FOREIGN KEY (cust_id) REFERENCES cust_dimen(cust_id);


# Question 1: Find the top 3 customers who have the maximum number of
orders

SELECT
    c.cust_id, -- Selecting customer ID from cust_dimen table
    c.customer_name, -- Selecting customer name from cust_dimen table
    COUNT(m.ord_id) AS orders_count -- Counting the number of orders for
each customer from market_fact table
FROM
    market_fact m -- Alias for market_fact table
JOIN
    cust_dimen c ON c.cust_id = m.cust_id -- Joining cust_dimen and
market_fact tables on cust_id
GROUP BY
    c.cust_id, c.customer_name -- Grouping the results by customer ID and
customer name
ORDER BY
    orders_count DESC -- Ordering the results in descending order based on
the number of orders
LIMIT
    3; -- Limiting the output to the top 3 results

/*
The above query retrieves the top 3 customers who have placed the maximum
number of orders.
It uses the market_fact and cust_dimen tables, joining them on the
customer ID,
and then groups the results by customer ID and name.
The result is ordered by the count of orders in descending order, and only
the top 3 rows are selected.
*/
```

```
#Question 2: Create a new column DaysTakenForDelivery that contains the
date difference between Order_Date and Ship_Date.

ALTER TABLE market_fact
ADD days_taken_for_delivery INT; -- Adding the 'DaysTakenForDelivery'
column to the 'market_fact' table

UPDATE market_fact AS m  -- Updating the column with datediff(ship_date,
order_date)
JOIN
   orders_dimen AS o
ON m.ord_id = o.ord_id -- Joining the 'market_fact' table with
'orders_dimen' on 'ord_id'
JOIN
   shipping_dimen s
ON m.ship_id = s.ship_id  -- Joining the result with 'shipping_dimen' on
'ship_id'
SET
  m.days_taken_for_delivery = DATEDIFF(s.ship_date, o.order_date);--
Setting the 'days_taken_for_delivery' column using DATEDIFF function

/*
The above set of SQL statements alters the market_fact table to add a new
column named 'DaysTakenForDelivery'.
It is then updated with the date difference between the 'Ship_Date' and
'Order_Date' for each order, using the DATEDIFF function.
*/


# Question 3: Find the customer whose order took the maximum time to get
delivered.

SELECT
    m.cust_id,          -- Selecting customer ID from market_fact table
    c.customer_name,    -- Selecting customer name from cust_dimen table
    m.ord_id,           -- Selecting order ID from market_fact table
    MAX(m.days_taken_for_delivery) AS delivery_time  -- Calculating the
maximum delivery time
FROM
    market_fact m
JOIN
    cust_dimen c USING (cust_id)  -- Performing an inner join based on
customer ID
GROUP BY
    1, 2, 3   -- Grouping the results by customer ID, customer name, and
order ID
ORDER BY
    delivery_time DESC  -- Sorting the results by delivery time in
descending order
LIMIT 1;   -- Limiting the result set to only the top record (with the
maximum delivery time)
```

```
/*
The above query identifies the customer whose order took the maximum time
for delivery.
It utilizes the 'market_fact' and 'cust_dimen' tables, grouping by
customer ID, customer name, and order ID.
The result is ordered in descending order based on the delivery time, and
only the top row is selected.
*/

# Question 4: Retrieve total sales made by each product from the data (use
Windows function)

SELECT DISTINCT
    prod_id, -- Selecting distinct product IDs from the market_fact table.

    -- Using the SUM() window function to calculate the total sales for
each product.
    -- The PARTITION BY clause ensures that the summation is done
separately for each product.
    SUM(sales) OVER (PARTITION BY prod_id) AS total_sales

FROM
    market_fact; -- Selecting data from the market_fact table.

/*
This query calculates the total sales made by each product using a window
function (SUM over PARTITION BY).
It considers the 'market_fact' table and retrieves distinct product IDs
along with the corresponding total sales for each product.
*/


# Question 5: Retrieve the total profit made from each product from the
data (use windows function)

SELECT DISTINCT
    prod_id, -- Selecting unique product IDs
    SUM(profit) OVER (PARTITION BY prod_id) AS total_profit -- Calculating
total profit for each product using a window function
FROM
    market_fact; -- Selecting data from the market_fact table

/*
Similar to Question 4, the above query computes the total profit made from
each product using a window function (SUM over PARTITION BY).
 It considers the 'market_fact' table and returns distinct product IDs
along with their total profits.
 */


# Question 6: Count the total number of unique customers in January and
how many of them came
#back every month over the entire year in 2011
```

```
SELECT
COUNT(DISTINCT m.cust_id) AS total_unique_customers_in_january --
Selecting the count of distinct customer IDs as
total_unique_customers_in_january
FROM
    market_fact m
JOIN
    orders_dimen o ON m.ord_id = o.ord_id -- Joining the market_fact table
with orders_dimen table using ord_id as the common key
WHERE
    YEAR(o.order_date) = 2011
    AND MONTH(o.order_date) = 1; -- Filtering the data to include only
orders made in January 2011

-- Counting customers who came back every month in 2011
-- Subquery to select distinct customer IDs who made purchases in every
month of 2011
SELECT COUNT(*) AS customers_came_back_every_month
FROM (
    SELECT m.cust_id      -- Selecting customer IDs from the market_fact
table

    FROM market_fact m
    JOIN orders_dimen o ON m.ord_id = o.ord_id      -- Joining market_fact
with orders_dimen on order ID
    WHERE YEAR(o.order_date) = 2011       -- Filtering records for the
year 2011
    GROUP BY m.cust_id                    -- Grouping by customer ID
    HAVING COUNT(DISTINCT YEAR(o.order_date)) * 12 = 12   -- Filtering
customers who made purchases in all 12 months of the year
) AS returning_customers;

/*
The first part of the query counts the total number of unique customers in
January 2011.
The second part counts the number of customers who placed orders every
month over the entire year in 2011.
The results provide insights into customer behavior during that specific
period.
*/
```