Sql Queries for analysis:

Sales Performance Analysis:

1. Total Sales by Product Category:
Question: Calculate the total sales revenue for each product category across all channels.

```
SELECT
    i.i_category AS category_name,
    SUM(ss.ss_sales_price * ss.ss_quantity) AS total_sales
FROM
    store_sales ss
JOIN
    item i ON ss.ss_item_sk = i.i_item_sk
GROUP BY
    i.i_category
ORDER BY
    total_sales DESC;
```

Result Grid | Filter Rows:

category_name	total_sales
Music	534844157.08
Shoes	526014719.25
Electronics	514394120.76
Sports	511116280.28
Women	510858644.91

2. Sales Trend Over Time:
Question: Analyze monthly sales trends for the past two years.

```
SELECT
     DATE_FORMAT(d_date, '%Y-%m') AS sale_month, SUM(ss_sales_price) AS
total_sales
FROM
    store_sales
JOIN
```

```
    date_dim ON store_sales.ss_sold_date_sk = date_dim.d_date_sk
GROUP BY
    sale_month;
```

sale_month	total_sales
2003-01	223002.64
2002-12	3481855.68
2002-11	3271835.60
2002-10	2295555.42
2002-09	2207359.07

3. Top 10 Best-Selling Products:

Question: Identify the top 10 best-selling products by total revenue.

```
SELECT
    ss_item_sk AS product_id,
    SUM(ss_sales_price) AS total_revenue
FROM
    store_sales
GROUP BY
    ss_item_sk
ORDER BY
    total_revenue DESC
LIMIT 10;
```

product_id	total_revenue
15229	15217.40
8299	15179.36
6013	14821.16
14905	14560.45
349	14427.50

4. Sales by Region:

Question: Calculate the total sales revenue by region for each sales channel.

```
SELECT
    a.ca_state AS state,
```

```
    SUM(ss.ss_sales_price * ss.ss_quantity) AS total_sales
FROM
    store_sales ss
JOIN
    customer_address a ON ss.ss_addr_sk = a.ca_address_sk
GROUP BY
    a.ca_state
ORDER BY
    total_sales DESC;
```

state	total_sales
TX	402719861.05
GA	244598917.31
VA	221793677.34
KY	191963424.74
KS	175181232.83

5. Year-over-Year Sales Growth:
Question: Compare the year-over-year sales growth for the current and previous year.

```
WITH YearlySales AS (
    SELECT
        d.d_year AS year,
        SUM(ss.ss_sales_price * ss.ss_quantity) AS total_sales
    FROM
        store_sales ss
    JOIN
        date_dim d ON ss.ss_sold_date_sk = d.d_date_sk
    GROUP BY
        d.d_year
),
SalesGrowth AS (
    SELECT
        current.year AS current_year,
        current.total_sales AS current_year_sales,
        previous.total_sales AS previous_year_sales,
```

```
        (current.total_sales - previous.total_sales) / previous.total_sales
* 100 AS growth_percentage
    FROM
        YearlySales current
    LEFT JOIN
        YearlySales previous ON current.year = previous.year + 1
    WHERE
        current.year = YEAR(CURDATE())
SELECT
    current_year,
    previous_year_sales,
    current_year_sales,
    growth_percentage
FROM
    SalesGrowth;
```

6. Sales Contribution by Channel:

Question: Determine the contribution of each sales channel (store, catalog, online) to the overall sales.

```
WITH SalesByChannel AS (
    SELECT
        'Store' AS channel,
        SUM(ss.ss_sales_price * ss.ss_quantity) AS total_sales
    FROM
        store_sales ss
    UNION ALL
    SELECT
        'Catalog' AS channel,
        SUM(cs.cs_sales_price * cs.cs_quantity) AS total_sales
    FROM
        catalog_sales cs
    UNION ALL
    SELECT
        'Online' AS channel,
        SUM(ws.ws_sales_price * ws.ws_quantity) AS total_sales
    FROM
        web_sales ws
),
TotalSales AS (
    SELECT
```

```
        SUM(total_sales) AS overall_sales
    FROM
        SalesByChannel
)
SELECT
    channel,
    total_sales,
    (total_sales / (SELECT overall_sales FROM TotalSales)) * 100 AS
contribution_percentage
FROM
    SalesByChannel;
```

channel	total_sales	contribution_percentage
Store	5141904166.52	48.393914
Catalog	3648626287.69	34.339673
Online	1834574423.87	17.266412

7. Sales Performance of New Products:

Question: Analyze the sales performance of products introduced in the last 6 months.

```
WITH RecentProducts AS (
    SELECT i_item_sk AS product_id
    FROM item
    WHERE i_rec_start_date >= CURDATE() - INTERVAL 6 MONTH
)

SELECT
    rp.product_id,
    COALESCE(SUM(cs.cs_sales_price * cs.cs_quantity), 0) AS total_sales
FROM
    RecentProducts rp
LEFT JOIN
    catalog_sales cs ON rp.product_id = cs.cs_item_sk
GROUP BY
    rp.product_id
```

```
ORDER BY
    total_sales DESC;
```

8. Average Order Value:

Question: Calculate the average order value for each sales channel.

```
SELECT
    'Store' AS sales_channel,
    SUM(ss_sales_price) / COUNT(DISTINCT ss_ticket_number) AS
average_order_value
FROM
    store_sales
UNION ALL
SELECT
    'Web' AS sales_channel,
    SUM(ws_sales_price) / COUNT(DISTINCT ws_order_number) AS
average_order_value
FROM
    web_sales
UNION ALL
SELECT
    'Catalog' AS sales_channel,
    SUM(cs_sales_price) / COUNT(DISTINCT cs_order_number) AS
average_order_value
FROM
    catalog_sales;
```

	sales_channel	average_order_value
▶	Store	434.299732
	Web	606.221152
	Catalog	452.646089

9. Seasonal Sales Analysis:

Question: Identify seasonal sales patterns by comparing sales during different quarters of the year.

```
SELECT
    CASE
        WHEN MONTH(sales_date) IN (6, 7, 8) THEN 'Summer'
        WHEN MONTH(sales_date) IN (12, 1, 2) THEN 'Winter'
```

```
        ELSE 'Other'
    END AS season,
    YEAR(sales_date) AS sales_year,
    SUM(sales_amount) AS total_sales
FROM
    (
        SELECT ss_sold_date_sk AS sales_date, ss_sales_price AS
sales_amount
        FROM store_sales
        UNION ALL
        SELECT ws_sold_date_sk AS sales_date, ws_sales_price AS
sales_amount
        FROM web_sales
        UNION ALL
        SELECT cs_sold_date_sk AS sales_date, cs_sales_price AS
sales_amount
        FROM catalog_sales
    ) AS sales
WHERE
    MONTH(sales_date) IN (6, 7, 8, 12, 1, 2)
GROUP BY
    season, sales_year
ORDER BY
    sales_year, season;
```

season	sales_year	total_sales
Summer	245	1101702.81
Winter	245	2121003.48

10. Product Category Sales Distribution:

Question: Determine the sales distribution across different product categories

```
SELECT
    i.i_category AS product_category,
    SUM(sales.sales_amount) AS total_sales
FROM
    item AS i
JOIN
```

```
    (
        SELECT ss_item_sk AS item_id, ss_sales_price AS sales_amount,
ss_sold_date_sk AS sales_date
        FROM store_sales
        UNION ALL
        SELECT ws_item_sk AS item_id, ws_sales_price AS sales_amount,
ws_sold_date_sk AS sales_date
        FROM web_sales
        UNION ALL
        SELECT cs_item_sk AS item_id, cs_sales_price AS sales_amount,
cs_sold_date_sk AS sales_date
        FROM catalog_sales
    ) AS sales
ON
    i.i_item_id = sales.item_id
WHERE
    sales.sales_date >= CURRENT_DATE() - INTERVAL 1 YEAR
GROUP BY
i.i_category
ORDER BY
    total_sales DESC
LIMIT 3;
```

Inventory Management Analysis

11. Inventory Turnover Ratio:

Question: Calculate the inventory turnover ratio for each product category.

```
SELECT
    ss_item_sk,
    (SUM(ss_quantity * ss_wholesale_cost)) / SUM(ss_quantity) AS
turnover_ratio
FROM
    store_sales
GROUP BY
    ss_item_sk;
```

ss_item_sk	turnover_ratio
1	53.962811
2	46.864247
3	51.994559
4	51.707984
5	46.045080

12. Stockout Rate by Product:

Question: Identify the products with the highest stockout rates in the past month.

```
SELECT
    i.i_item_sk AS Product_ID,
    COUNT(CASE WHEN inv.inv_quantity_on_hand = 0 THEN 1 END) AS
Stockout_Days,
    COUNT(DISTINCT inv.inv_date_sk) AS Total_Days,
    (COUNT(CASE WHEN inv.inv_quantity_on_hand = 0 THEN 1 END) * 1.0 /
COUNT(DISTINCT inv.inv_date_sk)) AS Stockout_Rate
FROM
    inventory inv
JOIN
    item i ON inv.inv_item_sk = i.i_item_sk
GROUP BY
    i.i_item_sk
ORDER BY
    Stockout_Rate DESC;
```

Product_ID	Stockout_Days	Total_Days	Stockout_Rate
5052	3	52	0.05769
13896	3	52	0.05769
9852	3	52	0.05769
16757	5	104	0.04808
1462	5	105	0.04762

13. Days of Inventory on Hand:

Question: Calculate the average days of inventory on hand for each product category.

```
    SELECT
    ss_item_sk,
    (SUM(ss_quantity) / NULLIF(SUM(ss_sales_price) / 30, 0)) AS
days_on_hand
FROM
    store_sales
GROUP BY
    ss_item_sk;
```

	ss_item_sk	days_on_hand
▶	1	37.7740
	2	39.1204
	3	41.3681
	4	39.5532
	5	39.2641

14. Top 10 Overstocked Products:

Question: List the top 10 products with the highest overstock levels.

```
 SELECT
    ss_item_sk AS Product_ID,
    SUM(ss_quantity) AS Total_Sold
FROM
    store_sales
GROUP BY
    ss_item_sk
ORDER BY
    Total_Sold DESC
LIMIT 10;
```

	Product_ID	Total_Sold
▶	9325	19072
	4279	18501
	7507	18475
	5953	18451
	16753	18446

15. Replenishment Frequency:

Question: Determine the replenishment frequency for high-demand products.

```
WITH HighDemand AS (
```

```
    SELECT
        ss_item_sk AS item_id,
        SUM(ss_quantity) AS total_sales
    FROM
        store_sales
    GROUP BY
        ss_item_sk
    ORDER BY
        total_sales DESC
    LIMIT 10
)
SELECT
    h.item_id,
    COUNT(*) AS replenishment_count
FROM
    HighDemand h
JOIN
    inventory inv ON h.item_id = inv.inv_item_sk
GROUP BY
    h.item_id
ORDER BY
    replenishment_count DESC;
```

item_id	replenishment_count
9325	1305
4279	1305
7507	1305
5953	1305
16753	1305

16. Inventory Aging Analysis:

Question: Analyze the aging of inventory to identify slow-moving products.

```
SELECT
    ss.ss_item_sk AS Product_ID,
    DATEDIFF(CURDATE(), MAX(d.d_date)) AS Days_Since_Last_Sale
FROM
    store_sales ss
JOIN
```

```
    date_dim d ON ss.ss_sold_date_sk = d.d_date_sk
GROUP BY
    ss.ss_item_sk
ORDER BY
    Days_Since_Last_Sale DESC
LIMIT 1000;
```

Product_ID	Days_Since_Last_Sale
2830	9049
3682	9045
3892	9044
856	9043
12562	9042

17. Warehouse Inventory Levels:

Question: Monitor the current inventory levels across all warehouses.

```
 SELECT
    ss_store_sk AS Store_ID,
    SUM(ss_quantity) AS Inventory_Level
FROM
    store_sales
GROUP BY
    ss_store_sk
LIMIT 100;
```

Store_ID	Inventory_Level
NULL	3291433
1	22551072
2	22634281
4	22564788
7	22620491

Customer Behavior Analysis

18. Customer Segmentation by Demographics:

Question: Segment customers based on age, income, and region.

```
WITH Customer_Age AS (
    SELECT
        c_customer_sk,
        YEAR(CURDATE()) - c_birth_year - (CASE WHEN (MONTH(CURDATE()) <
c_birth_month) OR (MONTH(CURDATE()) = c_birth_month AND DAY(CURDATE()) <
c_birth_day) THEN 1 ELSE 0 END) AS age,
        c_birth_country AS region
    FROM
        customer
)
SELECT
    CASE
        WHEN age < 25 THEN 'Under 25'
        WHEN age BETWEEN 25 AND 34 THEN '25-34'
        WHEN age BETWEEN 35 AND 44 THEN '35-44'
        WHEN age BETWEEN 45 AND 54 THEN '45-54'
        WHEN age BETWEEN 55 AND 64 THEN '55-64'
        ELSE '65 and Over'
    END AS Age_Group,
    'Income Data Missing' AS Income_Group,
    region,
    COUNT(*) AS Customer_Count
FROM
    Customer_Age
GROUP BY
    Age_Group, Income_Group, region
ORDER BY
    Age_Group, Income_Group, region
LIMIT 1000;
```

Age_Group	Income_Group	region	Customer_Count
25-34	Income Data Missing	NULL	80
25-34	Income Data Missing	AFGHANISTAN	18
25-34	Income Data Missing	ALAND ISLANDS	17
25-34	Income Data Missing	ALBANIA	28
25-34	Income Data Missing	ALGERIA	27

19. Customer Lifetime Value (CLTV):

Question: Calculate the customer lifetime value based on past purchase behavior.

```
 SELECT
    c.c_customer_id AS Customer_ID,
    SUM(ss.ss_sales_price * ss.ss_quantity) AS Total_Spend
FROM
    customer AS c
JOIN
    store_sales AS ss ON c.c_customer_sk = ss.ss_customer_sk
GROUP BY
    c.c_customer_id
LIMIT 0, 1000;
```

Customer_ID	Total_Spend
AAAAAAAABAAAAAAA	25824.66
AAAAAAAACAAAAAAA	70640.18
AAAAAAAADAAAAAAA	14111.51
AAAAAAAAEAAAAAAA	16653.74
AAAAAAAAFAAAAAAA	130184.88

20. Repeat Purchase Rate:

Question: Determine the repeat purchase rate for each customer segment

```
SELECT
    c.c_current_hdemo_sk AS Segment,
    COUNT(DISTINCT CASE WHEN ss.ss_ticket_number IS NOT NULL THEN
c.c_customer_id END) AS Repeat_Customers,
    COUNT(DISTINCT c.c_customer_id) AS Total_Customers,
    CASE
        WHEN COUNT(DISTINCT c.c_customer_id) = 0 THEN 0
        ELSE (COUNT(DISTINCT CASE WHEN ss.ss_ticket_number IS NOT NULL THEN
c.c_customer_id END) / COUNT(DISTINCT c.c_customer_id)) * 100
    END AS Repeat_Purchase_Rate
FROM
    customer c
LEFT JOIN
    store_sales ss ON c.c_customer_sk = ss.ss_customer_sk
GROUP BY
    c.c_current_hdemo_sk;
```

Segment	Repeat_Customers	Total_Customers	Repeat_Purchase_Rate
NULL	3128	3431	91.1688
1	13	16	81.2500
2	13	13	100.0000
3	20	21	95.2381
4	13	13	100.0000

21. Average Purchase Frequency:

Question: Calculate the average purchase frequency per customer.

```
SELECT
    AVG(purchase_count) AS Avg_Purchases_Per_Customer
FROM (
    SELECT
        c.c_customer_id,
        COUNT(ss.ss_ticket_number) AS purchase_count
    FROM
        customer c
    LEFT JOIN
        store_sales ss ON c.c_customer_sk = ss.ss_customer_sk
    GROUP BY
        c.c_customer_id
) AS Customer_Purchases;
```

Avg_Purchases_Per_Customer
27.5065

22. Customer Churn Analysis:

Question: Identify customers who have not made a purchase in the last year.

```
SELECT DISTINCT
    ss.ss_customer_sk AS Customer_ID
FROM
    store_sales ss
JOIN
    date_dim d ON ss.ss_sold_date_sk = d.d_date_sk
WHERE
    d.d_date >= '1902-01-01'
```

```
LIMIT 15;
```

Customer_ID
53877
43909
79890
99200
59649

23. Top 10 Most Valuable Customers:

Question: List the top 10 customers by total spend.

```
 SELECT
    c.c_customer_id,
    SUM(ss.ss_sales_price * ss.ss_quantity) AS Total_Spend
FROM
    customer c
JOIN
    store_sales ss ON c.c_customer_sk = ss.ss_customer_sk
GROUP BY
    c.c_customer_id
ORDER BY
    Total_Spend DESC
LIMIT 10
```

c_customer_id	Total_Spend
AAAAAAAABGPBBAAA	259827.38
AAAAAAAAJKONAAAA	259699.24
AAAAAAAANAMEAAAA	252380.65
AAAAAAAAKHEOAAAA	250773.25
AAAAAAAANBFBBAAA	249034.69

24. Customer Acquisition by Channel:

Question: Analyze how customers are acquired through different sales channels.

```sql
SELECT
    st.s_store_name AS Store_Name,
    COUNT(c.c_customer_id) AS Customer_Count
FROM
    customer c
JOIN
    store_sales ss ON c.c_customer_sk = ss.ss_customer_sk
JOIN
    store st ON ss.ss_store_sk = st.s_store_sk
GROUP BY
    st.s_store_name
ORDER BY
    Customer_Count DESC
LIMIT 10;
```

Store_Name	Customer_Count
bar	448600
eing	448519
ese	447529
ation	447501
able	447330

25. Customer Satisfaction Analysis:

Question: Correlate customer satisfaction scores with purchase behavior (requires hypothetical satisfaction data).

```sql
SELECT
    c.c_customer_id AS Customer_ID,
    ROUND(RAND() * 5, 2) AS Hypothetical_Satisfaction_Score,
    SUM(ss.ss_sales_price * ss.ss_quantity) AS Total_Spend
FROM
    customer c
JOIN
    store_sales ss ON c.c_customer_sk = ss.ss_customer_sk
GROUP BY
    c.c_customer_id
ORDER BY
    Hypothetical_Satisfaction_Score DESC
LIMIT 10;
```

Customer_ID	Hypothetical_Satisfaction_Score	Total_Spend
AAAAAAAAFDIAAAAA	5	64127.49
AAAAAAAAINDDAAAA	5	46054.50
AAAAAAAABCPCAAAA	5	18104.13
AAAAAAAAEHBBAAAA	5	38041.88
AAAAAAAAPBPBAAAA	5	12913.07

Promotional Effectiveness Analysis

26. Promotion Uplift Analysis:

Question: Measure the increase in sales during promotional periods compared to non-promotional periods.

```
SELECT
    CASE
        WHEN ss.ss_promo_sk IS NOT NULL THEN 'Promotional Period'
        ELSE 'Non-Promotional Period'
    END AS Period,
    SUM(ss.ss_sales_price * ss.ss_quantity) AS Total_Sales
FROM
    store_sales ss
LEFT JOIN
    promotion p ON ss.ss_promo_sk = p.p_promo_sk
GROUP BY
    Period
ORDER BY
    Total_Sales DESC;
```

Period	Total_Sales
Promotional Period	5079463924.47
Non-Promotional Period	62440242.05

27. ROI of Promotional Campaigns:

Question: Calculate the return on investment (ROI) for each promotional campaign.

```
SELECT
    p.p_promo_name AS Promotion_Name,
    SUM(ss.ss_sales_price * ss.ss_quantity) AS Total_Revenue,
```

```
    SUM(p.p_cost) AS Total_Cost,
    (SUM(ss.ss_sales_price * ss.ss_quantity) - SUM(p.p_cost)) /
SUM(p.p_cost) AS ROI
FROM
    store_sales ss
JOIN
    promotion p ON ss.ss_promo_sk = p.p_promo_sk
GROUP BY
    p.p_promo_name
ORDER BY
    ROI DESC;
```

Promotion_Name	Total_Revenue	Total_Cost	ROI
NULL	119369930.68	9289000.00	11.850676
bar	493889564.35	266943000.00	0.850169
eing	490195515.71	265135000.00	0.848853
ese	490376541.11	265275000.00	0.848559
ought	475516725.89	257298000.00	0.848117

28. Customer Response Rate to Promotions:

Question: Determine the response rate of customers to different promotions.

```
SELECT
    p.p_promo_name AS Promotion_Name,
    COUNT(DISTINCT ss.ss_customer_sk) AS Customer_Count,
    COUNT(ss.ss_promo_sk) AS Total_Promo_Uses,
    (COUNT(DISTINCT ss.ss_customer_sk) / COUNT(ss.ss_promo_sk)) * 100 AS
Response_Rate
FROM
    store_sales ss
JOIN
    promotion p ON ss.ss_promo_sk = p.p_promo_sk
GROUP BY
    p.p_promo_name
ORDER BY
    Response_Rate DESC;
```

Promotion_Name	Customer_Count	Total_Promo_Uses	Response_Rate
NULL	42574	64400	66.1087
ought	79187	257298	30.7764
eing	79864	265135	30.1220
cally	79843	265532	30.0691
ese	79721	265275	30.0522

Result 8

29. Effectiveness of Discounts vs. Coupons:

Question: Compare the effectiveness of discount-based promotions versus coupon-based promotions.

```
SELECT
    'Catalog' AS Sale_Type,
    COALESCE(SUM(cs_ext_discount_amt), 0) AS Total_Discount,
    COALESCE(SUM(cs_coupon_amt), 0) AS Total_Coupon
FROM
    catalog_sales
WHERE
    cs_sold_date_sk >= CURDATE() - INTERVAL 90 DAY

UNION ALL

SELECT
    'Web' AS Sale_Type,
    COALESCE(SUM(ws_ext_discount_amt), 0) AS Total_Discount,
    COALESCE(SUM(ws_coupon_amt), 0) AS Total_Coupon
FROM
    web_sales
WHERE
    ws_sold_date_sk >= CURDATE() - INTERVAL 90 DAY

UNION ALL

SELECT
    'Store' AS Sale_Type,
    COALESCE(SUM(ss_ext_discount_amt), 0) AS Total_Discount,
    COALESCE(SUM(ss_coupon_amt), 0) AS Total_Coupon
FROM
    store_sales
WHERE
```

```
    ss_sold_date_sk >= CURDATE() - INTERVAL 90 DAY;
```

	Sale_Type	Total_Discount	Total_Coupon
▶	Catalog	0.00	0.00
	Web	0.00	0.00
	Store	0.00	0.00

30. Promotion in Recents:
Question: Identify how many new customers were acquired during promotions in the last 3 months.

```
-- Catalog Sales
SELECT
    'Catalog' AS Sales_Channel,
    COUNT(DISTINCT cs.cs_bill_customer_sk) AS New_Customers
FROM
    catalog_sales cs
JOIN
    promotion p ON cs.cs_promo_sk = p.p_promo_sk
WHERE
    p.p_start_date_sk >= (SELECT MAX(p_start_date_sk) - 90 FROM promotion)

UNION ALL

SELECT
    'Web' AS Sales_Channel,
    COUNT(DISTINCT ws.ws_bill_customer_sk) AS New_Customers
FROM
    web_sales ws
JOIN
    promotion p ON ws.ws_promo_sk = p.p_promo_sk
WHERE
    p.p_start_date_sk >= (SELECT MAX(p_start_date_sk) - 90 FROM promotion)

UNION ALL

SELECT
    'Store' AS Sales_Channel,
    COUNT(DISTINCT ss.ss_customer_sk) AS New_Customers
FROM
```

```
    store_sales ss
JOIN
    promotion p ON ss.ss_promo_sk = p.p_promo_sk
WHERE
    p.p_start_date_sk >= (SELECT MAX(p_start_date_sk) - 90 FROM promotion);
```

Sales_Channel	New_Customers
Catalog	59435
Web	33447
Store	79361

31. Customer Response Rate:

Question: Determine the customer response rate to specific promotions in the last quarter.

```
  SELECT
    (COUNT(DISTINCT cs.c_customer_id) / (SELECT COUNT(DISTINCT
c.c_customer_id) FROM customer c)) * 100 AS Response_Rate
FROM
    customer c
JOIN
    store_sales ss ON c.c_customer_sk = ss.ss_customer_sk
JOIN
    promotion p ON ss.ss_promo_sk = p.p_promo_sk
WHERE
    p.p_start_date_sk BETWEEN DATE_SUB(CURDATE(), INTERVAL 3 MONTH) AND
CURDATE()
```

Total_Customers
100000

32. Promotion-Driven New Customer Acquisition:

Question: Identify how many new customers were acquired during promotional periods.

```
SELECT
    COUNT(DISTINCT c.c_customer_id) AS New_Customers_Acquired
FROM
    customer c
```

```
JOIN
    store_sales ss ON c.c_customer_sk = ss.ss_customer_sk
JOIN
    promotion p ON ss.ss_promo_sk = p.p_promo_sk
WHERE
    c.c_first_sales_date_sk BETWEEN p.p_start_date_sk AND p.p_end_date_sk
    AND p.p_start_date_sk >= CURDATE() - INTERVAL 3 MONTH
```

	New_Customers_Acquired
▶	0

Channel Performance Analysis

33. Sales Contribution by Channel:

Question: Calculate the contribution of each sales channel to the total revenue.

```
 WITH ChannelRevenue AS (
    SELECT
        'Catalog Sales' AS channel,
        SUM(cs.cs_net_paid) AS total_revenue
    FROM
        catalog_sales cs
    UNION ALL
    SELECT
        'Web Sales' AS channel,
SUM(ws.ws_net_paid) AS total_revenue
    FROM
        web_sales ws
    UNION ALL
    SELECT
        'Store Sales' AS channel,
        SUM(ss.ss_net_paid) AS total_revenue
    FROM
        store_sales ss
),
TotalRevenue AS (
    SELECT
        SUM(total_revenue) AS overall_revenue
    FROM
        ChannelRevenue
),
ChannelContribution AS (
```

```
    SELECT
        cr.channel,
        cr.total_revenue,
        (cr.total_revenue / tr.overall_revenue) * 100 AS
contribution_percentage
    FROM
        ChannelRevenue cr
    CROSS JOIN
        TotalRevenue tr
)
SELECT
    channel,
    total_revenue,
    contribution_percentage
FROM
    ChannelContribution
ORDER BY
    contribution_percentage DESC;
```

channel	total_revenue	contribution_percentage
Store Sales	4741589953.76	48.966960
Catalog Sales	3291204790.03	33.988661
Web Sales	1650448767.17	17.044379

34. Customer Satisfaction by Channel:

Question: Analyze customer satisfaction scores across different sales channels (requires hypothetical satisfaction data).

```
WITH customer_satisfaction AS (
    SELECT 1 AS cs_order_number, 'Catalog' AS cs_channel, 4.5 AS
cs_satisfaction
    UNION ALL
    SELECT 2, 'Web', 4.0
    UNION ALL
    SELECT 3, 'Store', 3.8
    UNION ALL
    SELECT 4, 'Catalog', 4.7
    UNION ALL
    SELECT 5, 'Web', 4.2
```

```
    UNION ALL
    SELECT 6, 'Store', 3.9
    UNION ALL
    SELECT 7, 'Catalog', 4.6
    UNION ALL
    SELECT 8, 'Web', 4.1
    UNION ALL
    SELECT 9, 'Store', 4.0
)
SELECT
    cs_channel AS channel,
    AVG(cs_satisfaction) AS average_satisfaction,
    COUNT(cs_order_number) AS number_of_responses
FROM
    customer_satisfaction
GROUP BY
    cs_channel
ORDER BY
    average_satisfaction DESC;
```

channel	average_satisfaction	number_of_responses
Catalog	4.60000	3
Web	4.10000	3
Store	3.90000	3

35. Conversion Rate for Online Sales:

Question: Calculate the conversion rate for web visitors who complete a purchase.

```
WITH total_web_visitors AS (
    SELECT COUNT(DISTINCT ws_bill_customer_sk) AS total_visitors
    FROM web_sales
),
completed_purchases AS (
    SELECT COUNT(DISTINCT ws_order_number) AS completed_orders
    FROM web_sales
)

SELECT
    (cp.completed_orders / twv.total_visitors) * 100 AS
```

```
conversion_rate_percentage
FROM
    completed_purchases cp,
    total_web_visitors twv;
```

	conversion_rate_percentage
▶	132.8580

36. In-Store vs. Online Sales Growth:

Question: Analyze which products perform best in-store vs. online.

```
SELECT
    COALESCE(c.product_id, o.product_id) AS product_id,
    COALESCE(c.total_instore_sales, 0) AS total_instore_sales,
    COALESCE(o.total_online_sales, 0) AS total_online_sales
FROM
    (SELECT
        cs_item_sk AS product_id,
        SUM(cs_net_paid) AS total_instore_sales
     FROM
        catalog_sales
     GROUP BY cs_item_sk) c
LEFT JOIN
    (SELECT
        ws_item_sk AS product_id,
        SUM(ws_net_paid) AS total_online_sales
     FROM
        web_sales
     GROUP BY ws_item_sk) o
ON c.product_id = o.product_id

UNION

SELECT
    COALESCE(c.product_id, o.product_id) AS product_id,
    COALESCE(c.total_instore_sales, 0) AS total_instore_sales,
    COALESCE(o.total_online_sales, 0) AS total_online_sales
FROM
    (SELECT
```

```
        cs_item_sk AS product_id,
        SUM(cs_net_paid) AS total_instore_sales
     FROM
        catalog_sales
     GROUP BY cs_item_sk) c
RIGHT JOIN
    (SELECT
        ws_item_sk AS product_id,
        SUM(ws_net_paid) AS total_online_sales
     FROM
        web_sales
     GROUP BY ws_item_sk) o
ON c.product_id = o.product_id;
```

Result Grid | Filter Rows: | Expo

product_id	total_instore_sales	total_online_sales
1	428942.79	164719.39
2	181213.30	88413.65
3	130612.79	99457.15
4	146213.69	62337.10
5	134249.85	55319.58
6	55685.22	60067.93
7	304139.72	165151.07
8	166736.46	84123.38
9	117865.73	62099.20
10	202179.87	33569.61

Result 42

37. Product Performance by Channel:

Question: Analyze which products perform best in each sales channel.

```
SELECT
    COALESCE(c.product_id, o.product_id) AS product_id,
    COALESCE(c.total_instore_sales, 0) AS total_instore_sales,
    COALESCE(o.total_online_sales, 0) AS total_online_sales
FROM
    (SELECT
```

```sql
        cs_item_sk AS product_id,
        SUM(cs_net_paid) AS total_instore_sales
     FROM
        catalog_sales
     GROUP BY cs_item_sk) c
LEFT JOIN
    (SELECT
        ws_item_sk AS product_id,
        SUM(ws_net_paid) AS total_online_sales
     FROM
        web_sales
     GROUP BY ws_item_sk) o
ON c.product_id = o.product_id

UNION

SELECT
    COALESCE(c.product_id, o.product_id) AS product_id,
    COALESCE(c.total_instore_sales, 0) AS total_instore_sales,
    COALESCE(o.total_online_sales, 0) AS total_online_sales
FROM
    (SELECT
        cs_item_sk AS product_id,
        SUM(cs_net_paid) AS total_instore_sales
     FROM
        catalog_sales
     GROUP BY cs_item_sk) c
RIGHT JOIN
    (SELECT
        ws_item_sk AS product_id,
        SUM(ws_net_paid) AS total_online_sales
     FROM
        web_sales
     GROUP BY ws_item_sk) o
ON c.product_id = o.product_id
ORDER BY
    total_instore_sales DESC,
    total_online_sales DESC;
```

product_id	total_instore_sales	total_online_sales
1057	537858.18	229403.29
15739	537800.79	146433.65
2131	527254.30	121462.63
1429	523132.26	106252.93
17575	519510.44	192110.63

38. Channel Profitability Analysis:

Question: Calculate the profitability of each sales channel by comparing revenue to associated costs.

```
SELECT
    'in-store' AS channel,
    SUM(cs_net_paid) AS total_revenue,
    SUM(cs_ext_wholesale_cost) AS total_cost,
    SUM(cs_net_paid) - SUM(cs_ext_wholesale_cost) AS total_profit
FROM
    catalog_sales

UNION ALL

SELECT
    'online' AS channel,
    SUM(ws_net_paid) AS total_revenue,
    SUM(ws_ext_wholesale_cost) AS total_cost,
    SUM(ws_net_paid) - SUM(ws_ext_wholesale_cost) AS total_profit
FROM
    web_sales;
```

channel	total_revenue	total_cost	total_profit
in-store	3291204790.03	3657224304.67	-366019514.64
online	1650448767.17	1837751907.65	-187303140.48

Supply Chain and Logistics Analysis

39. Warehouse Turnover Rate:

Question: Calculate the inventory turnover rate for each warehouse.

```
WITH cogs_catalog AS (
    SELECT cs_warehouse_sk AS warehouse_sk,
           SUM(cs_ext_wholesale_cost) AS total_cogs_catalog
    FROM catalog_sales
    GROUP BY cs_warehouse_sk
),
cogs_web AS (
    SELECT ws_warehouse_sk AS warehouse_sk,
           SUM(ws_ext_wholesale_cost) AS total_cogs_web
    FROM web_sales
    GROUP BY ws_warehouse_sk
),
combined_cogs AS (
    SELECT cogs_catalog.warehouse_sk,
           COALESCE(total_cogs_catalog, 0) AS total_cogs_catalog,
           COALESCE(total_cogs_web, 0) AS total_cogs_web
    FROM cogs_catalog
    LEFT JOIN cogs_web
    ON cogs_catalog.warehouse_sk = cogs_web.warehouse_sk

    UNION

    SELECT cogs_web.warehouse_sk,
           COALESCE(total_cogs_catalog, 0) AS total_cogs_catalog,
           COALESCE(total_cogs_web, 0) AS total_cogs_web
    FROM cogs_web
    LEFT JOIN cogs_catalog
    ON cogs_web.warehouse_sk = cogs_catalog.warehouse_sk
)
SELECT warehouse_sk,
       (total_cogs_catalog + total_cogs_web) AS total_cogs,
       (total_cogs_catalog + total_cogs_web) / 100000 AS
inventory_turnover_rate -- Assuming avg inventory = 100000
FROM combined_cogs;
```

warehouse_sk	total_cogs	inventory_turnover_rate
NULL	9016874.67	90.168747
1	1098154342.35	10981.543424
2	1099716387.50	10997.163875
3	1094871335.07	10948.713351
4	1097639178.68	10976.391787
5	1095385414.41	10953.854144
NULL	192679.64	1.926796

40. Average Shipping Time:

Question: Determine the average shipping time for orders across different regions.

```
 WITH shipping_times AS (
    SELECT
        s.s_market_id AS region_id,
        (dd_ship.d_date - dd_sold.d_date) AS shipping_time
    FROM
        web_sales ws
    JOIN
        date_dim dd_sold ON ws.ws_sold_date_sk = dd_sold.d_date_sk
    JOIN
        date_dim dd_ship ON ws.ws_ship_date_sk = dd_ship.d_date_sk
    JOIN
        store s ON ws.ws_ship_addr_sk = s.s_store_sk
)
SELECT
    region_id,
    AVG(shipping_time) AS avg_shipping_time
FROM
    shipping_times
GROUP BY
    region_id;
```

	region_id	avg_shipping_time
▶	2	5415.6250
	8	3766.7347
	10	4623.1538
	9	183.6000
	6	531.0417

41. Delivery Success Rate:

Question: Analyze the delivery success rate and identify regions with high failure rates.

```
WITH delivery_status AS (
    SELECT
        s.s_market_id AS region_id,
        'catalog' AS source,
        COUNT(*) AS total_deliveries,
        SUM(CASE WHEN cs.cs_ship_date_sk IS NOT NULL THEN 1 ELSE 0 END) AS
successful_deliveries,
        SUM(CASE WHEN cs.cs_ship_date_sk IS NULL THEN 1 ELSE 0 END) AS
failed_deliveries
    FROM
        catalog_sales cs
    JOIN
        store s ON cs.cs_warehouse_sk = s.s_store_sk
    GROUP BY
        s.s_market_id
    UNION ALL
    SELECT
        s.s_market_id AS region_id,
        'web' AS source,
        COUNT(*) AS total_deliveries,
        SUM(CASE WHEN ws.ws_ship_date_sk IS NOT NULL THEN 1 ELSE 0 END) AS
successful_deliveries,
        SUM(CASE WHEN ws.ws_ship_date_sk IS NULL THEN 1 ELSE 0 END) AS
failed_deliveries
    FROM
        web_sales ws
    JOIN
        store s ON ws.ws_warehouse_sk = s.s_store_sk
    GROUP BY
        s.s_market_id
)
```

```sql
SELECT
    region_id,
    source,
    total_deliveries,
    successful_deliveries,
    failed_deliveries,
    ROUND((successful_deliveries * 100.0) / total_deliveries, 2) AS
success_rate,
    ROUND((failed_deliveries * 100.0) / total_deliveries, 2) AS
failure_rate
FROM
    delivery_status
ORDER BY
    failure_rate DESC;
```

region_id	source	total_deliveries	successful_deliveries	failed_deliveries	success_rate	failure_rate
2	catalog	287570	286847	723	99.75	0.25
8	catalog	574102	572650	1452	99.75	0.25
4	catalog	286506	285809	697	99.76	0.24
7	catalog	286224	285551	673	99.76	0.24
8	web	287603	287569	34	99.99	0.01

42. Warehouse Stock Levels:

Question: Monitor the stock levels of key products in each warehouse.

```sql
WITH combined_sales AS (
    SELECT
        cs.cs_item_sk AS item_id,
        SUM(cs.cs_quantity) AS total_sales_qty
    FROM
        catalog_sales cs
    GROUP BY
        cs.cs_item_sk

    UNION ALL

    SELECT
        ws.ws_item_sk AS item_id,
        SUM(ws.ws_quantity) AS total_sales_qty
    FROM
        web_sales ws
    LEFT JOIN
        catalog_sales cs ON ws.ws_item_sk = cs.cs_item_sk
    WHERE
```

```
        cs.cs_item_sk IS NULL
    GROUP BY
        ws.ws_item_sk
)
SELECT
    item_id,
    100 - coalesce(SUM(total_sales_qty), 0) AS current_stock_level
FROM
    combined_sales
GROUP BY
    item_id
ORDER BY
    item_id;
```

item_id	current_stock_level
1	-8416
2	-4577
3	-3400
4	-2888
5	-3187

43. Shipping Mode Efficiency:

Question: Compare the efficiency of different shipping modes in terms of cost and delivery time.

```
SELECT
    ws_ship_mode_sk AS shipping_mode,
    AVG(ws_net_paid) AS avg_cost,
    AVG(ws_ship_date_sk - ws_sold_date_sk) AS avg_delivery_time
FROM
    web_sales
GROUP BY
    ws_ship_mode_sk
ORDER BY
    avg_delivery_time ASC;
```

shipping_mode	avg_cost	avg_delivery_time
NULL	2238.832048	53.7561
9	2267.240892	60.2359
2	2282.178815	60.3649
19	2273.845577	60.3753
16	2284.012681	60.4073

44. Supply Chain Bottleneck Analysis:

Question: Identify bottlenecks in the supply chain by analyzing delays in order fulfillment.

```
WITH order_delays AS (
    SELECT
        ws_order_number,
        ws_ship_date_sk - ws_sold_date_sk AS delivery_delay,
        s.s_store_name AS warehouse_name
    FROM
        web_sales ws
    JOIN
        store s ON ws.ws_warehouse_sk = s.s_store_sk
    WHERE
        ws_ship_date_sk IS NOT NULL
)
SELECT
    warehouse_name,
    AVG(delivery_delay) AS avg_delay,
    MAX(delivery_delay) AS max_delay
FROM
    order_delays
GROUP BY
    warehouse_name
ORDER BY
    avg_delay DESC;
```

warehouse_name	avg_delay	max_delay
ought	60.6283	120
ese	60.5927	120
able	60.5382	120
anti	60.5297	120

45. Order Fulfillment Rate:

Question: Calculate the order fulfillment rate to ensure timely delivery of products.

```
WITH OrderStats AS (
    SELECT
        ca.ca_state AS state,
        COUNT(DISTINCT orders.order_number) AS total_orders,
        COUNT(DISTINCT CASE WHEN orders.ship_date IS NOT NULL THEN
orders.order_number END) AS fulfilled_orders
    FROM
        (SELECT cs_order_number AS order_number, cs_bill_addr_sk,
cs_ship_date_sk AS ship_date FROM catalog_sales
         UNION ALL
         SELECT ws_order_number AS order_number, ws_bill_addr_sk,
ws_ship_date_sk AS ship_date FROM web_sales) orders
    JOIN
        customer_address ca ON orders.cs_bill_addr_sk = ca.ca_address_sk
    GROUP BY
        ca.ca_state
)

SELECT
    state,
    total_orders,
    fulfilled_orders,
    CASE WHEN total_orders = 0 THEN 0 ELSE (fulfilled_orders /
total_orders) * 100 END AS fulfillment_rate
FROM
    OrderStats
ORDER BY
    fulfillment_rate DESC;
```

state	total_orders	fulfilled_orders	fulfillment_rate
NULL	6711	6711	100.0000
AK	1566	1566	100.0000
AL	4364	4364	100.0000
AR	5078	5078	100.0000
AZ	930	930	100.0000