

Cliira Descriptive Analytics: SQL Queries Documentation

August 26, 2025

Contents

1	Introduction	1
1.1	Note on Placeholders	2
2	SQL Queries	2
2.1	KPI/Section: Department List Fetch	2
2.2	KPI/Section: Category List Fetch	2
2.3	KPI/Section: Global Metrics (Total Revenue, Patients, SKUs, Transactions)	2
2.4	KPI/Section: Revenue Analysis - Department Level	2
2.5	KPI/Section: Revenue Analysis - SKU Level	3
2.6	KPI/Section: Margin Analysis - Department Level	3
2.7	KPI/Section: Margin Analysis - SKU Level	4
2.8	KPI/Section: Prescription Analytics - Department Level	5
2.9	KPI/Section: Prescription Analytics - SKU Level	5
2.10	KPI/Section: Margin per Patient Analysis - Department Level	6
2.11	KPI/Section: Margin per Patient Analysis - SKU Level	6
2.12	KPI/Section: Bounce Rate Analysis - Department Level	7
2.13	KPI/Section: Bounce Rate Analysis - SKU Level	8
2.14	KPI/Section: Inventory Turnover Analysis - Department Level	9
2.15	KPI/Section: Inventory Turnover Analysis - SKU Level	10
2.16	KPI/Section: Formulary Adherence Analysis - Department Level	10
2.17	KPI/Section: Formulary Adherence Analysis - Physician Level	11
2.18	KPI/Section: Outpatient to Prescription Conversion	11
2.19	KPI/Section: Expiry Items Analysis	12
2.20	KPI/Section: Consumables Analytics - Value Analysis	13
2.21	KPI/Section: Consumables Analytics - Quantity Analysis	13
2.22	KPI/Section: Consumables Analytics - Per Patient Usage	13
2.23	KPI/Section: Consumables Analytics - Turnover Analysis	14
2.24	KPI/Section: Enhanced Generic vs Brand Analysis - Department Level	15
2.25	KPI/Section: Enhanced Generic vs Brand Analysis - SKU Level	15
2.26	KPI/Section: Average Consumable Usage Patterns	15

1 Introduction

This document provides a comprehensive list of SQL queries used in the Cliira Descriptive Analytics Dashboard to fetch data for various Key Performance Indicators (KPIs) and analyses. Each entry includes the KPI or section name and the corresponding SQL query extracted from the dashboard code.

1.1 Note on Placeholders

Many Queries contain dynamic placeholders such as {dept_filter} and {category_filter}. These placeholders are replaced at runtime with actual SQL filter clauses (e.g., AND d.dept_name IN ('Dept1', 'Dept2')) based on user selections in the dashboard's sidebar filters for departments and categories. If no filters are applied, these placeholders resolve to empty strings.

2 SQL Queries

2.1 KPI/Section: Department List Fetch

```
1 SELECT DISTINCT dept_name FROM departments ORDER BY dept_name
```

2.2 KPI/Section: Category List Fetch

```
1 SELECT DISTINCT category FROM skus WHERE category IS NOT NULL ORDER BY
   category
```

2.3 KPI/Section: Global Metrics (Total Revenue, Patients, SKUs, Transactions)

```
1 WITH transactions_all AS (
2     SELECT * FROM transactions_2023
3     UNION ALL
4     SELECT * FROM transactions_2024
5 )
6 SELECT
7     SUM(t.total_cost) as total_revenue,
8     COUNT(DISTINCT t.patient_id) as total_patients,
9     COUNT(DISTINCT t.sku_id) as total_skus,
10    COUNT(DISTINCT t.transaction_id) as total_transactions
11 FROM transactions_all t
12 JOIN departments d ON t.dept_id = d.dept_id AND t.hospital_id =
   d.hospital_id
13 JOIN skus s ON t.sku_id = s.sku_id
14 WHERE t.total_cost IS NOT NULL AND t.quantity_consumed > 0
15     {dept_filter}
16     {category_filter}
```

2.4 KPI/Section: Revenue Analysis - Department Level

```
1 SELECT
2     d.dept_name,
3     d.dept_id,
4     COUNT(DISTINCT t.sku_id) as unique_skus,
5     SUM(t.quantity_consumed) as total_quantity,
6     SUM(t.total_cost) as total_revenue
7 FROM (
8     SELECT * FROM transactions_2023
9     UNION ALL
10    SELECT * FROM transactions_2024
11 ) t
12 JOIN departments d ON t.dept_id = d.dept_id
13 JOIN skus s ON t.sku_id = s.sku_id
14 WHERE t.total_cost IS NOT NULL
15     AND t.quantity_consumed > 0
```

```

16     {dept_filter}
17     {category_filter}
18 GROUP BY d.dept_name, d.dept_id
19 ORDER BY total_revenue DESC;

```

2.5 KPI/Section: Revenue Analysis - SKU Level

```

1  SELECT
2      s.skus_name,
3      s.skus_id,
4      s.category,
5      s.sub_category,
6      s.therapeutic_area,
7      s.brand_generic_flag,
8      SUM(t.quantity_consumed) AS sku_total_quantity,
9      AVG(t.unit_cost) AS sku_avg_unit_cost,
10     SUM(t.total_cost) AS sku_total_revenue,
11     COUNT(DISTINCT t.transaction_id) AS total_transactions,
12     COUNT(DISTINCT t.patient_id) AS unique_patients_served,
13     COUNT(DISTINCT t.physician_id) AS unique_physicians_prescribing,
14     ROUND((
15         SUM(t.total_cost) / NULLIF(COUNT(DISTINCT t.patient_id), 0)
16     )::numeric, 2) AS revenue_per_patient,
17     ROUND((
18         SUM(t.total_cost) / NULLIF(COUNT(DISTINCT t.transaction_id), 0)
19     )::numeric, 2) AS revenue_per_transaction
20 FROM (
21     SELECT * FROM transactions_2023
22     UNION ALL
23     SELECT * FROM transactions_2024
24 ) t
25 JOIN skus s ON t.skus_id = s.skus_id
26 WHERE t.total_cost IS NOT NULL
27     AND t.quantity_consumed > 0
28     {category_filter}
29 GROUP BY s.skus_name, s.skus_id,
30         s.category, s.sub_category, s.therapeutic_area,
31         s.brand_generic_flag
32 ORDER BY sku_total_revenue DESC;

```

2.6 KPI/Section: Margin Analysis - Department Level

```

1  WITH delivery_ranked AS (
2      SELECT
3          sku_id,
4          hospital_id,
5          delivery_date,
6          actual_unit_price,
7          ROW_NUMBER() OVER (PARTITION BY sku_id, hospital_id ORDER BY
8              delivery_date DESC) as rn
9      FROM deliveries
10 ),
11 transactions_all AS (
12     SELECT * FROM transactions_2023
13     UNION ALL
14     SELECT * FROM transactions_2024

```

```

15 SELECT
16     dep.dept_id,
17     dep.dept_name,
18     SUM(t.total_cost) as total_revenue,
19     SUM(t.quantity_consumed * d.actual_unit_price) as total_cost,
20     SUM(t.total_cost) - SUM(t.quantity_consumed * d.actual_unit_price) as
        margin_amount,
21     CASE
22         WHEN SUM(t.quantity_consumed * d.actual_unit_price) > 0
23         THEN ((SUM(t.total_cost) - SUM(t.quantity_consumed *
                d.actual_unit_price)) / SUM(t.quantity_consumed *
                d.actual_unit_price) * 100)
24         ELSE 0
25     END as margin_percentage
26 FROM transactions_all t
27 JOIN departments dep ON t.dept_id = dep.dept_id AND t.hospital_id =
        dep.hospital_id
28 JOIN skus s ON t.sku_id = s.sku_id
29 JOIN delivery_ranked d ON t.sku_id = d.sku_id
30     AND t.hospital_id = d.hospital_id
31     AND d.delivery_date <= t.transaction_date
32     AND d.rn = 1
33 WHERE 1=1 {dept_filter} {category_filter}
34 GROUP BY dep.dept_id, dep.dept_name
35 ORDER BY margin_amount DESC;

```

2.7 KPI/Section: Margin Analysis - SKU Level

```

1 WITH delivery_ranked AS (
2     SELECT
3         sku_id,
4         hospital_id,
5         delivery_date,
6         actual_unit_price,
7         ROW_NUMBER() OVER (PARTITION BY sku_id, hospital_id ORDER BY
                delivery_date DESC) as rn
8     FROM deliveries
9 ),
10 transactions_all AS (
11     SELECT * FROM transactions_2023
12     UNION ALL
13     SELECT * FROM transactions_2024
14 )
15 SELECT
16     s.sku_id,
17     s.sku_name,
18     s.category,
19     SUM(t.total_cost) as total_revenue,
20     SUM(t.quantity_consumed * d.actual_unit_price) as total_cost,
21     SUM(t.total_cost) - SUM(t.quantity_consumed * d.actual_unit_price) as
        margin_amount,
22     CASE
23         WHEN SUM(t.quantity_consumed * d.actual_unit_price) > 0
24         THEN ((SUM(t.total_cost) - SUM(t.quantity_consumed *
                d.actual_unit_price)) / SUM(t.quantity_consumed *
                d.actual_unit_price) * 100)
25         ELSE 0

```

```

26     END as margin_percentage
27 FROM transactions_all t
28 JOIN skus s ON t.sku_id = s.sku_id
29 JOIN delivery_ranked d ON t.sku_id = d.sku_id
30     AND t.hospital_id = d.hospital_id
31     AND d.delivery_date <= t.transaction_date
32     AND d.rn = 1
33 WHERE 1=1 {category_filter}
34 GROUP BY s.sku_id, s.sku_name, s.category
35 ORDER BY margin_amount DESC;

```

2.8 KPI/Section: Prescription Analytics - Department Level

```

1 WITH all_transactions AS (
2     SELECT DISTINCT t.transaction_id, t.dept_id, t.transaction_date,
3         t.sku_id
4     FROM transactions_2023 t
5     JOIN skus s ON t.sku_id = s.sku_id
6     WHERE t.transaction_type = 'Prescription' {category_filter}
7     UNION
8     SELECT DISTINCT t.transaction_id, t.dept_id, t.transaction_date,
9         t.sku_id
10    FROM transactions_2024 t
11    JOIN skus s ON t.sku_id = s.sku_id
12    WHERE t.transaction_type = 'Prescription' {category_filter}
13 )
14 SELECT
15     d.dept_name,
16     d.dept_type,
17     COUNT(DISTINCT t.transaction_id) as total_prescriptions
18 FROM all_transactions t
19 JOIN departments d ON t.dept_id = d.dept_id
20 WHERE 1=1 {dept_filter}
21 GROUP BY d.dept_name, d.dept_type
22 ORDER BY total_prescriptions DESC;

```

2.9 KPI/Section: Prescription Analytics - SKU Level

```

1 WITH all_transactions AS (
2     SELECT transaction_id, dept_id, sku_id, transaction_date
3     FROM transactions_2023
4     WHERE transaction_type = 'Prescription'
5     UNION ALL
6     SELECT transaction_id, dept_id, sku_id, transaction_date
7     FROM transactions_2024
8     WHERE transaction_type = 'Prescription'
9 )
10 SELECT
11     s.sku_name,
12     s.category,
13     s.sub_category,
14     s.therapeutic_area,
15     s.brand_generic_flag,
16     COUNT(DISTINCT t.transaction_id
17         System: id) as prescriptions_containing_this_sku,
18     COUNT(DISTINCT t.dept_id) as prescribed_by_departments
19

```

```

20 FROM all_transactions t
21 JOIN skus s ON t.sku_id = s.sku_id
22 WHERE 1=1 {category_filter}
23 GROUP BY s.sku_name, s.category, s.sub_category, s.therapeutic_area,
           s.brand_generic_flag
24 ORDER BY prescriptions_containing_this_sku DESC;

```

2.10 KPI/Section: Margin per Patient Analysis - Department Level

```

1  WITH delivery_ranked AS (
2      SELECT
3          sku_id,
4          hospital_id,
5          delivery_date,
6          actual_unit_price,
7          ROW_NUMBER() OVER (PARTITION BY sku_id, hospital_id ORDER BY
8              delivery_date DESC) as rn
9      FROM deliveries
10 ),
11 transactions_all AS (
12     SELECT * FROM transactions_2023
13     UNION ALL
14     SELECT * FROM transactions_2024
15 )
16 SELECT
17     dep.dept_id,
18     dep.dept_name,
19     COUNT(DISTINCT t.patient_id) as total_patients,
20     SUM(t.total_cost) - SUM(t.quantity_consumed * d.actual_unit_price) as
21     total_margin,
22     CASE
23         WHEN COUNT(DISTINCT t.patient_id) > 0
24         THEN (SUM(t.total_cost) - SUM(t.quantity_consumed *
25             d.actual_unit_price)) / COUNT(DISTINCT t.patient_id)
26         ELSE 0
27     END as margin_per_patient
28 FROM transactions_all t
29 JOIN departments dep ON t.dept_id = dep.dept_id AND t.hospital_id =
30     dep.hospital_id
31 JOIN skus s ON t.sku_id = s.sku_id
32 JOIN delivery_ranked d ON t.sku_id = d.sku_id
33     AND t.hospital_id = d.hospital_id
34     AND d.delivery_date <= t.transaction_date
35     AND d.rn = 1
36 WHERE t.patient_id IS NOT NULL {dept_filter} {category_filter}
37 GROUP BY dep.dept_id, dep.dept_name
38 ORDER BY margin_per_patient DESC;

```

2.11 KPI/Section: Margin per Patient Analysis - SKU Level

```

1  WITH delivery_ranked AS (
2      SELECT
3          sku_id,
4          hospital_id,
5          delivery_date,
6          actual_unit_price,

```

```

7         ROW_NUMBER() OVER (PARTITION BY sku_id, hospital_id ORDER BY
8             delivery_date DESC) as rn
9     FROM deliveries
10 ),
11 transactions_all AS (
12     SELECT * FROM transactions_2023
13     UNION ALL
14     SELECT * FROM transactions_2024
15 )
16 SELECT
17     s.sku_id,
18     s.sku_name,
19     s.category,
20     COUNT(DISTINCT t.patient_id) as total_patients,
21     SUM(t.total_cost) - SUM(t.quantity_consumed * d.actual_unit_price) as
22     total_margin,
23     CASE
24         WHEN COUNT(DISTINCT t.patient_id) > 0
25         THEN (SUM(t.total_cost) - SUM(t.quantity_consumed *
26             d.actual_unit_price)) / COUNT(DISTINCT t.patient_id)
27         ELSE 0
28     END as margin_per_patient
29 FROM transactions_all t
30 JOIN skus s ON t.sku_id = s.sku_id
31 JOIN delivery_ranked d ON t.sku_id = d.sku_id
32     AND t.hospital_id = d.hospital_id
33     AND d.delivery_date <= t.transaction_date
34     AND d.rn = 1
35 WHERE t.patient_id IS NOT NULL {category_filter}
36 GROUP BY s.sku_id, s.sku_name, s.category
37 ORDER BY margin_per_patient DESC;

```

2.12 KPI/Section: Bounce Rate Analysis - Department Level

```

1 WITH transactions_all AS (
2     SELECT * FROM transactions_2023
3     UNION ALL
4     SELECT * FROM transactions_2024
5 ),
6 dept_avg_cost AS (
7     SELECT
8         t.dept_id,
9         t.sku_id,
10        AVG(t.unit_cost) as avg_unit_cost
11 FROM transactions_all t
12 JOIN skus s ON t.sku_id = s.sku_id
13 WHERE 1=1 {category_filter}
14 GROUP BY t.dept_id, t.sku_id
15 )
16 SELECT
17     dep.dept_id,
18     dep.dept_name,
19     COUNT(*) as total_sku_records,
20     COUNT(CASE WHEN i.bounce_status = 'Bounced' THEN 1 END) as
21     bounced_records,
22     SUM(CASE WHEN i.bounce_status = 'Bounced' THEN
23         COALESCE(i.bounce_quantity, 0) ELSE 0 END) as

```

```

22         total_bounced_quantity,
        SUM(CASE WHEN i.bounce_status = 'Bounced' THEN
            COALESCE(i.bounce_quantity, 0) * COALESCE(dac.avg_unit_cost, 0)
            ELSE 0 END) as financial_impact,
23     ROUND(
24         (COUNT(CASE WHEN i.bounce_status = 'Bounced' THEN 1 END) * 100.0)
            / NULLIF(COUNT(*), 0),
25         2
26     ) as bounce_rate_percentage
27 FROM inventory i
28 JOIN departments dep ON i.hospital_id = dep.hospital_id
29 JOIN skus s ON i.sku_id = s.sku_id
30 LEFT JOIN dept_avg_cost dac ON dep.dept_id = dac.dept_id AND i.sku_id =
    dac.sku_id
31 WHERE i.bounce_status IS NOT NULL {dept_filter} {category_filter}
32 GROUP BY dep.dept_id, dep.dept_name
33 ORDER BY total_bounced_quantity DESC;

```

2.13 KPI/Section: Bounce Rate Analysis - SKU Level

```

1  WITH transactions_all AS (
2      SELECT * FROM transactions_2023
3      UNION ALL
4      SELECT * FROM transactions_2024
5  ),
6  sku_avg_cost AS (
7      SELECT
8          t.sku_id,
9          AVG(t.unit_cost) as avg_unit_cost
10     FROM transactions_all t
11     GROUP BY t.sku_id
12 )
13 SELECT
14     s.sku_id,
15     s.sku_name,
16     s.category,
17     COUNT(*) as total_records,
18     COUNT(CASE WHEN i.bounce_status = 'Bounced' THEN 1 END) as
        bounced_records,
19     SUM(CASE WHEN i.bounce_status = 'Bounced' THEN
        COALESCE(i.bounce_quantity, 0) ELSE 0 END) as
        total_bounced_quantity,
20     SUM(CASE WHEN i.bounce_status = 'Bounced' THEN
        COALESCE(i.bounce_quantity, 0) * COALESCE(sac.avg_unit_cost, 0)
        ELSE 0 END) as financial_impact,
21     ROUND(
22         (COUNT(CASE WHEN i.bounce_status = 'Bounced' THEN 1 END) * 100.0)
            / NULLIF(COUNT(*), 0),
23         2
24     ) as bounce_rate_percentage
25 FROM inventory i
26 JOIN skus s ON i.sku_id = s.sku_id
27 LEFT JOIN sku_avg_cost sac ON i.sku_id = sac.sku_id
28 WHERE i.bounce_status IS NOT NULL
29 GROUP BY s.sku_id, s.sku_name, s.category
30 HAVING COUNT(CASE WHEN i.bounce_status = 'Bounced' THEN 1 END) > 0
31 ORDER BY total_bounced_quantity DESC;

```


2.14 KPI/Section: Inventory Turnover Analysis - Department Level

```
1 WITH transactions_all AS (  
2     SELECT * FROM transactions_2023  
3     UNION ALL  
4     SELECT * FROM transactions_2024  
5 ),  
6 dept_inventory AS (  
7     SELECT  
8         t.dept_id,  
9         i.sku_id,  
10        i.hospital_id,  
11        i.opening_stock_h1_2023,  
12        (i.stock_in_h1_2023 + i.stock_in_h2_2023 + i.stock_in_h1_2024 +  
13         i.stock_in_h2_2024) as total_stock_in,  
14        (i.stock_out_h1_2023 + i.stock_out_h2_2023 + i.stock_out_h1_2024 +  
15         i.stock_out_h2_2024) as total_stock_out,  
16        i.current_stock  
17     FROM inventory i  
18     JOIN transactions_all t ON i.sku_id = t.sku_id AND i.hospital_id =  
19     t.hospital_id  
20     JOIN skus s ON i.sku_id = s.sku_id  
21     WHERE 1=1 {category_filter}  
22     GROUP BY t.dept_id, i.sku_id, i.hospital_id, i.opening_stock_h1_2023,  
23     i.stock_in_h1_2023, i.stock_in_h2_2023, i.stock_in_h1_2024,  
24     i.stock_in_h2_2024,  
25     i.stock_out_h1_2023, i.stock_out_h2_2023, i.stock_out_h1_2024,  
26     i.stock_out_h2_2024,  
27     i.current_stock  
28 )  
29 SELECT  
30     dep.dept_id,  
31     dep.dept_name,  
32     SUM(total_stock_out) as total_consumed_quantity,  
33     SUM((openingSTS  
34     System: opening_stock_h1_2023 + i.stock_in_h1_2023 + i.stock_in_h2_2023 +  
35     i.stock_in_h1_2024 + i.stock_in_h2_2024) as total_ordered_quantity,  
36     SUM(i.stock_out_h1_2023 + i.stock_out_h2_2023 + i.stock_out_h1_2024 +  
37     i.stock_out_h2_2024) as total_consumed_quantity,  
38     SUM(i.current_stock) as total_current_stock,  
39     SUM((i.opening_stock_h1_2023 + i.stock_in_h1_2023 + i.stock_in_h2_2023  
40     + i.stock_in_h1_2024 + i.stock_in_h2_2024) / 2.0) as  
41     average_inventory,  
42     CASE  
43         WHEN SUM((i.opening_stock_h1_2023 + i.stock_in_h1_2023 +  
44         i.stock_in_h2_2023 + i.stock_in_h1_2024 + i.stock_in_h2_2024) /  
45         2.0) > 0  
46         THEN SUM(i.stock_out_h1_2023 + i.stock_out_h2_2023 +  
47         i.stock_out_h1_2024 + i.stock_out_h2_2024) /  
48         SUM((i.opening_stock_h1_2023 + i.stock_in_h1_2023 +  
49         i.stock_in_h2_2023 + i.stock_in_h1_2024 +  
50         i.stock_in_h2_2024) / 2.0)  
51         ELSE 0  
52     END as inventory_turnover_ratio  
53 FROM dept_inventory di  
54 JOIN departments dep ON di.dept_id = dep.dept_id AND di.hospital_id =  
55     dep.hospital_id  
56 WHERE 1=1 {dept_filter}
```

```

43 GROUP BY dep.dept_id, dep.dept_name
44 ORDER BY inventory_turnover_ratio DESC;

```

2.15 KPI/Section: Inventory Turnover Analysis - SKU Level

```

1  SELECT
2      s.sku_id,
3      s.sku_name,
4      s.category,
5      SUM(i.opening_stock_h1_2023) as total_opening_stock,
6      SUM(i.stock_in_h1_2023 + i.stock_in_h2_2023 + i.stock_in_h1_2024 +
7          i.stock_in_h2_2024) as total_ordered_quantity,
8      SUM(i.stock_out_h1_2023 + i.stock_out_h2_2023 + i.stock_out_h1_2024 +
9          i.stock_out_h2_2024) as total_consumed_quantity,
10     SUM(i.current_stock) as total_current_stock,
11     SUM((i.opening_stock_h1_2023 + i.stock_in_h1_2023 + i.stock_in_h2_2023
12         + i.stock_in_h1_2024 + i.stock_in_h2_2024) / 2.0) as
13     average_inventory,
14     CASE
15         WHEN SUM((i.opening_stock_h1_2023 + i.stock_in_h1_2023 +
16             i.stock_in_h2_2023 + i.stock_in_h1_2024 + i.stock_in_h2_2024) /
17             2.0) > 0
18         THEN SUM(i.stock_out_h1_2023 + i.stock_out_h2_2023 +
19             i.stock_out_h1_2024 + i.stock_out_h2_2024) /
20             SUM((i.opening_stock_h1_2023 + i.stock_in_h1_2023 +
21                 i.stock_in_h2_2023 + i.stock_in_h1_2024 +
22                 i.stock_in_h2_2024) / 2.0)
23         ELSE 0
24     END as inventory_turn_finding
25 GROUP BY s.sku_id, s.sku_name, s.category
26 ORDER BY s.sku_id, s.sku_name, s.formulary_status
27 WHERE formulary_status IS NOT NULL
28 GROUP BY s.sku_id, s.sku_name, s.formulary_status

```

2.16 KPI/Section: Formulary Adherence Analysis - Department Level

```

1  WITH transactions_all AS (
2      SELECT * FROM transactions_2023
3      UNION ALL
4      SELECT * FROM transactions_2024
5  )
6  SELECT
7      dep.dept_id,
8      dep.dept_name,
9      COUNT(*) as total_transactions,
10     COUNT(CASE WHEN s.formulary_status = 'Formulary' THEN 1 END) as
11     formulary_transactions,
12     COUNT(CASE WHEN s.formulary_status = 'Non-Formulary' THEN 1 END) as
13     non_formulary_transactions,
14     CASE
15         WHEN COUNT(*) > 0
16         THEN (COUNT(CASE WHEN s.formulary_status = 'Formulary' THEN 1 END)
17             * 100.0) / COUNT(*)
18         ELSE 0
19     END as formulary_adherence_rate,
20     SUM(CASE WHEN s.formulary_status = 'Formulary' THEN t.total_cost ELSE
21         0 END) as formulary_revenue,

```

```

18     SUM(CASE WHEN s.formulary_status = 'Non-Formulary' THEN t.total_cost
19           ELSE 0 END) as non_formulary_revenue
19 FROM transactions_all t
20 JOIN departments dep ON t.dept_id = dep.dept_id AND t.hospital_id =
    dep.hospital_id
21 JOIN skus s ON t.sku_id = s.sku_id
22 WHERE s.formulary_status IS NOT NULL {dept_filter} {category_filter}
23 GROUP BY dep.dept_id, dep.dept_name
24 ORDER BY non_formulary_revenue DESC;

```

2.17 KPI/Section: Formulary Adherence Analysis - Physician Level

```

1 WITH transactions_all AS (
2     SELECT * FROM transactions_2023
3     UNION ALL
4     SELECT * FROM transactions_2024
5 )
6 SELECT
7     p.physician_id,
8     p.physician_name,
9     p.specialty,
10    dep.dept_name,
11    COUNT(*) as total_transactions,
12    COUNT(CASE WHEN s.formulary_status = 'Formulary' THEN 1 END) as
        formulary_transactions,
13    COUNT(CASE WHEN s.formulary_status = 'Non-Formulary' THEN 1 END) as
        non_formulary_transactions,
14    CASE
15        WHEN COUNT(*) > 0
16        THEN (COUNT(CASE WHEN s.formulary_status = 'Formulary' THEN 1 END)
            * 100.0) / COUNT(*)
17        ELSE 0
18    END as formulary_adherence_rate,
19    SUM(CASE WHEN s.formulary_status = 'Non-Formulary' THEN t.total_cost
20          ELSE 0 END) as non_formulary_revenue_impact
21 FROM transactions_all t
22 JOIN physicians p ON t.physician_id = p.physician_id AND t.hospital_id =
    p.hospital_id
23 JOIN departments dep ON p.primary_dept_id = dep.dept_id AND p.hospital_id
    = dep.hospital_id
24 JOIN skus s ON t.sku_id = s.sku_id
25 WHERE s.formulary_status IS NOT NULL {dept_filter} {category_filter}
26 GROUP BY p.physician_id, p.physician_name, p.specialty, dep.dept_name
27 ORDER BY non_formulary_revenue_impact DESC;

```

2.18 KPI/Section: Outpatient to Prescription Conversion

```

1 WITH transactions_all AS (
2     SELECT * FROM transactions_2023
3     UNION ALL
4     SELECT * FROM transactions_2024
5 ),
6 filtered_transactions AS (
7     SELECT t.*
8     FROM transactions_all t
9     JOIN departments d ON t.dept_id = d.dept_id AND t.hospital_id =
        d.hospital_id

```

```

10 JOIN skus s ON t.sku_id = s.sku_id
11 WHERE 1=1
12 AND d.dept_name IN ('Cardiology','Emergency Medicine','General
13 Surgery','ICU','Internal Medicine','Nephrology','Obstetrics &
14 Gynecology','Oncology','Orthopedics','Pediatrics')
15 AND s.category IN ('Consumables','Pharmacy')
16 ),
17 outpatient_analysis AS (
18 SELECT
19 COUNT(DISTINCT CASE WHEN EXISTS (
20 SELECT 1 FROM filtered_transactions ft WHERE ft.patient_id =
21 p.patient_id
22 ) THEN p.patient_id END) as total_outpatients,
23 COUNT(DISTINCT CASE WHEN EXISTS (
24 SELECT 1 FROM filtered_transactions ft
25 WHERE ft.patient_id = p.patient_id AND ft.transaction_type =
26 'Prescription'
27 ) THEN p.patient_id END) as outpatients_with_prescriptions
28 FROM patients p
29 WHERE p.patient_type = 'Outpatient'
30 )
31 SELECT
32 total_outpatients,
33 outpatients_with_prescriptions,
34 CASE
35 WHEN total_outpatients > 0
36 THEN (outpatients_with_prescriptions * 100.0) / total
37
38 System: id
39 FROM skus s ON t.sku_id = s.sku_id
40 WHERE t.transaction_type = 'Prescription'
41 GROUP BY s.formulary_status

```

2.19 KPI/Section: Expiry Items Analysis

```

1 SELECT
2 d.dept_name,
3 s.sku_name,
4 s.category,
5 i.current_stock,
6 i.expiry_date,
7 s.standard_cost,
8 (i.current_stock * s.standard_cost) as inventory_value,
9 EXTRACT(days FROM (i.expiry_date - CURRENT_DATE)) as days_to_expiry,
10 CASE
11 WHEN i.expiry_date <= CURRENT_DATE + INTERVAL '365 days' THEN
12 '0-12 months'
13 WHEN i.expiry_date <= CURRENT_DATE + INTERVAL '730 days' THEN '1-2
14 years'
15 WHEN i.expiry_date <= CURRENT_DATE + INTERVAL '1095 days' THEN
16 '2-3 years'
17 ELSE '3+ years'
18 END as expiry_timeframe
19 FROM inventory i
20 JOIN departments d ON i.hospital_id = d.hospital_id
21 JOIN skus s ON i.sku_id = s.sku_id
22 WHERE i.expiry_date IS NOT NULL

```

```

20     AND i.current_stock > 0
21     AND i.expiry_date <= CURRENT_DATE
22     {dept_filter.replace('dep.dept_name', 'd.dept_name') if dept_filter
23       else ''}
23     {category_filter}
24 ORDER BY i.expiry_date;

```

2.20 KPI/Section: Consumables Analytics - Value Analysis

```

1 WITH transactions_all AS (
2     SELECT * FROM transactions_2023
3     UNION ALL
4     SELECT * FROM transactions_2024
5 )
6 SELECT
7     dep.dept_id,
8     dep.dept_name,
9     SUM(t.total_cost) as total_consumables_value
10 FROM transactions_all t
11 JOIN departments dep ON t.dept_id = dep.dept_id AND t.hospital_id =
    dep.hospital_id
12 JOIN skus s ON t.sku_id = s.sku_id
13 WHERE s.category = 'Consumables' {dept_filter} {category_filter}
14 GROUP BY dep.dept_id, dep.dept_name
15 ORDER BY total_consumables_value DESC;

```

2.21 KPI/Section: Consumables Analytics - Quantity Analysis

```

1 WITH transactions_all AS (
2     SELECT * FROM transactions_2023
3     UNION ALL
4     SELECT * FROM transactions_2024
5 )
6 SELECT
7     s.sku_id,
8     s.sku_name,
9     s.sub_category,
10    s.unit_of_measure,
11    SUM(t.quantity_consumed) as total_consumables_quantity
12 FROM transactions_all t
13 JOIN skus s ON t.sku_id = s.sku_id
14 JOIN departments dep ON t.dept_id = dep.dept_id AND t.hospital_id =
    dep.hospital_id
15 WHERE s.category = 'Consumables' {dept_filter} {category_filter}
16 GROUP BY s.sku_id, s.sku_name, s.sub_category, s.unit_of_measure
17 ORDER BY total_consumables_quantity DESC;

```

2.22 KPI/Section: Consumables Analytics - Per Patient Usage

```

1 WITH transactions_all AS (
2     SELECT * FROM transactions_2023
3     UNION ALL
4     SELECT * FROM transactions_2024
5 )
6 SELECT
7     dep.dept_id,

```

```

8      dep.dept_name,
9      COUNT(DISTINCT t.patient_id) as total_patients,
10     SUM(t.quantity_consumed) as total_consumables_quantity,
11     SUM(t.total_cost) as total_consumables_value,
12     CASE
13         WHEN COUNT(DISTINCT t.patient_id) > 0
14         THEN SUM(t.quantity_consumed) / COUNT(DISTINCT t.patient_id)
15         ELSE 0
16     END as consumables_quantity_per_patient,
17     CASE
18         WHEN COUNT(DISTINCT t.patient_id) > 0
19         THEN SUM(t.total_cost) / COUNT(DISTINCT t.patient_id)
20         ELSE 0
21     END as consumables_value_per_patient
22 FROM transactions_all t
23 JOIN departments dep ON t.dept_id = dep
24
25 System: id
26 FROM skus s ON t.sku_id = s.sku_id
27 WHERE t.transaction_type = 'Prescription'

```

2.23 KPI/Section: Consumables Analytics - Turnover Analysis

```

1 WITH transactions_all AS (
2     SELECT * FROM transactions_2023
3     UNION ALL
4     SELECT * FROM transactions_2024
5 ),
6 dept_consumables_inventory AS (
7     SELECT
8         t.dept_id,
9         i.sku_id,
10        i.hospital_id,
11        i.opening_stock_h1_2023,
12        (i.stock_in_h1_2023 + i.stock_in_h2_2023 + i.stock_in_h1_2024 +
13         i.stock_in_h2_2024) as total_ordered_quantity,
14        SUM(i.stock_out_h1_2023 + i.stock_out_h2_2023 +
15         i.stock_out_h1_2024 + i.stock_out_h2_2024) as
16        total_consumed_quantity,
17        SUM(i.current_stock) as total_current_stock,
18        SUM((i.opening_stock_h1_2023 + i.stock_in_h1_2023 +
19         i.stock_in_h2_2023 + i.stock_in_h1_2024 + i.stock_in_h2_2024) /
20        2.0) as average_inventory,
21        CASE
22            WHEN SUM((i.opening_stock_h1_2023 + i.stock_in_h1_2023 +
23             i.stock_in_h2_2024 + i.stock_in_h2_2024) / 2.0) > 0
24            THEN SUM(i.stock_out_h1_2023 + i.stock_out_h2_2023 +
25             i.stock_out_h1_2024 + i.stock_out_h2_2024 +
26             i.stock_in_h2_2024) /
27             SUM((i.opening_stock_h1_2023 + i.stock_in_h1_2023 + i

```

```

21 System: id
22 FROM skus s ON t.sku_id = s.sku_id
23 JOIN inventory i ON s.sku_id = i.sku_id
24 WHERE s.category = 'Consumables' {dept_filter} {category_filter}
25 GROUP BY s.sku_id, s.sku_name, s.category
26 ORDER BY total_consumed_quantity DESC;

```

2.24 KPI/Section: Enhanced Generic vs Brand Analysis - Department Level

```
1 WITH transactions AS (  
2     SELECT * FROM transactions_2023  
3     UNION ALL  
4     SELECT * FROM transactions_2024  
5 )  
6 SELECT  
7     d.dept_name,  
8     s.brand_generic_flag,  
9     COUNT(DISTINCT s.sku_id) AS sku_count,  
10    SUM(t.quantity_consumed) AS total_volume,  
11    SUM(t.total_cost) AS total_revenue,  
12    AVG(t.unit_cost) AS avg_unit_cost,  
13    ROUND(  
14        (SUM(t.total_cost) / NULLIF(COUNT(DISTINCT s.sku_id), 0)  
15    )::numeric, 2  
16    ) AS revenue_percentage_in_dept  
17 FROM transactions t  
18 JOIN departments d ON t.dept_id = d.dept_id  
19 JOIN skus s ON t.sku_id = s.sku_id  
20 WHERE t.total_cost IS NOT NULL  
21        AND t.quantity_consumed > 0  
22        AND s.brand_generic_flag IS NOT NULL  
23        {dept_filter}  
24        {category_filter}  
25 GROUP BY d.dept_name, s.brand_generic_flag  
26 ORDER BY total_revenue DESC;
```

2.25 KPI/Section: Enhanced Generic vs Brand Analysis - SKU Level

```
1 WITH transactions AS (  
2     SELECT * FROM transactions_2023  
3     UNION ALL  
4     SELECT * FROM transactions_2024  
5 )  
6 SELECT  
7     s.brand_generic_flag,  
8     s.sku_name,  
9     s.sku_id,  
10    s.category,  
11    SUM(t.quantity_consumed) AS total_volume,  
12    SUM(t.total_cost) AS total_revenue,  
13    AVG(t.unit_cost) AS avg_unit_cost  
14 FROM transactions t  
15 JOIN skus s ON t.sku_id = s.sku_id  
16 WHERE t.total_cost IS NOT NULL  
17        AND t.quantity_consumed > 0  
18        AND s.brand_generic_flag IS NOT NULL  
19        {category_filter}  
20 GROUP BY s.brand_generic_flag, s.sku_name, s.sku_id, s.category  
21 ORDER BY total_revenue DESC;
```

2.26 KPI/Section: Average Consumable Usage Patterns

```
1 WITH transactions_all AS (  
2     SELECT * FROM transactions_2023
```

```

3      UNION ALL
4      SELECT * FROM transactions_2024
5  )
6  SELECT
7      dep.dept_id,
8      dep.dept_name,
9      COUNT(DISTINCT t.transaction_id) as total_transactions,
10     COUNT(DISTINCT t.patient_id) as total_patients,
11     COUNT(DISTINCT t.sku_id) as unique_consumables_used,
12     SUM(t.quantity_consumed) as total_quantity_used,
13     SUM(t.total_cost) as total_value_used,
14     CASE
15         WHEN COUNT(DISTINCT t.patient_id) > 0
16         THEN SUM(t.quantity_consumed) / COUNT(DISTINCT t.patient_id)
17         ELSE 0
18     END as avg_quantity_per_transaction,
19     CASE
20         WHEN COUNT(DISTINCT t.patient_id) > 0
21         THEN SUM(t.total_cost) / COUNT(DISTINCT t.patient_id)
22         ELSE 0
23     END as avg_value_per_patient,
24     CASE
25         WHEN COUNT(DISTINCT t.patient_id) > 0
26         THEN SUM(t.quantity_consumed) / COUNT(DISTINCT t.patient_id)
27         ELSE 0
28     END as avg_value_per_patient
29 FROM transactions_all t
30 JOIN departments dep ON t.dept_id = dep.dept_id AND t.hospital_id =
    dep.hospital_id
31 JOIN skus s ON t.sku_id = s.sku_id
32 WHERE s.category = 'Consumables' {dept_filter.replace('dep.dept_name',
    'd.dept_name') if dept_filter else ''} {category_filter}
33 GROUP BY dep.dept_id, dep.dept_name
34 ORDER BY total_quantity_used DESC;

```