Cliira Descriptive Analytics: SQL Queries Documentation

August 26, 2025

Contents

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

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

2.2 KPI/Section: Category List Fetch

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

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

```sql
WITH transactions_all AS (
    SELECT * FROM transactions_2023
    UNION ALL
    SELECT * FROM transactions_2024
)
SELECT
    SUM(t.total_cost) as total_revenue,
    COUNT(DISTINCT t.patient_id) as total_patients,
    COUNT(DISTINCT t.sku_id) as total_skus,
    COUNT(DISTINCT t.transaction_id) as total_transactions
FROM transactions_all t
JOIN departments d ON t.dept_id = d.dept_id AND t.hospital_id =
    d.hospital_id
JOIN skus s ON t.sku_id = s.sku_id
WHERE t.total_cost IS NOT NULL AND t.quantity_consumed > 0
    {dept_filter}
    {category_filter}
```

2.4 KPI/Section: Revenue Analysis - Department Level

```sql
SELECT
    d.dept_name,
    d.dept_id,
    COUNT(DISTINCT t.sku_id) as unique_skus,
    SUM(t.quantity_consumed) as total_quantity,
    SUM(t.total_cost) as total_revenue
FROM (
    SELECT * FROM transactions_2023
    UNION ALL
    SELECT * FROM transactions_2024
) t
JOIN departments d ON t.dept_id = d.dept_id
JOIN skus s ON t.sku_id = s.sku_id
WHERE t.total_cost IS NOT NULL
    AND t.quantity_consumed > 0
```

```
    {dept_filter}
    {category_filter}
GROUP BY d.dept_name, d.dept_id
ORDER BY total_revenue DESC;
```

2.5 KPI/Section: Revenue Analysis - SKU Level

```
SELECT
    s.sku_name,
    s.sku_id,
    s.category,
    s.sub_category,
    s.therapeutic_area,
    s.brand_generic_flag,
    SUM(t.quantity_consumed) AS sku_total_quantity,
    AVG(t.unit_cost) AS sku_avg_unit_cost,
    SUM(t.total_cost) AS sku_total_revenue,
    COUNT(DISTINCT t.transaction_id) AS total_transactions,
    COUNT(DISTINCT t.patient_id) AS unique_patients_served,
    COUNT(DISTINCT t.physician_id) AS unique_physicians_prescribing,
    ROUND((
        SUM(t.total_cost) / NULLIF(COUNT(DISTINCT t.patient_id), 0)
    )::numeric, 2) AS revenue_per_patient,
    ROUND((
        SUM(t.total_cost) / NULLIF(COUNT(DISTINCT t.transaction_id), 0)
    )::numeric, 2) AS revenue_per_transaction
FROM (
    SELECT * FROM transactions_2023
    UNION ALL
    SELECT * FROM transactions_2024
) t
JOIN skus s ON t.sku_id = s.sku_id
WHERE t.total_cost IS NOT NULL
    AND t.quantity_consumed > 0
    {category_filter}
GROUP BY s.sku_name, s.sku_id,
         s.category, s.sub_category, s.therapeutic_area,
            s.brand_generic_flag
ORDER BY sku_total_revenue DESC;
```

2.6 KPI/Section: Margin Analysis - Department Level

```
WITH delivery_ranked AS (
    SELECT
        sku_id,
        hospital_id,
        delivery_date,
        actual_unit_price,
        ROW_NUMBER() OVER (PARTITION BY sku_id, hospital_id ORDER BY
            delivery_date DESC) as rn
    FROM deliveries
),
transactions_all AS (
    SELECT * FROM transactions_2023
    UNION ALL
    SELECT * FROM transactions_2024
)
```

```sql
SELECT
    dep.dept_id,
    dep.dept_name,
    SUM(t.total_cost) as total_revenue,
    SUM(t.quantity_consumed * d.actual_unit_price) as total_cost,
    SUM(t.total_cost) - SUM(t.quantity_consumed * d.actual_unit_price) as
       margin_amount,
    CASE
        WHEN SUM(t.quantity_consumed * d.actual_unit_price) > 0
        THEN ((SUM(t.total_cost) - SUM(t.quantity_consumed *
           d.actual_unit_price)) / SUM(t.quantity_consumed *
           d.actual_unit_price) * 100)
        ELSE 0
    END as margin_percentage
FROM transactions_all t
JOIN departments dep ON t.dept_id = dep.dept_id AND t.hospital_id =
   dep.hospital_id
JOIN skus s ON t.sku_id = s.sku_id
JOIN delivery_ranked d ON t.sku_id = d.sku_id
    AND t.hospital_id = d.hospital_id
    AND d.delivery_date <= t.transaction_date
    AND d.rn = 1
WHERE 1=1 {dept_filter} {category_filter}
GROUP BY dep.dept_id, dep.dept_name
ORDER BY margin_amount DESC;
```

2.7 KPI/Section: Margin Analysis - SKU Level

```sql
WITH delivery_ranked AS (
    SELECT
        sku_id,
        hospital_id,
        delivery_date,
        actual_unit_price,
        ROW_NUMBER() OVER (PARTITION BY sku_id, hospital_id ORDER BY
           delivery_date DESC) as rn
    FROM deliveries
),
transactions_all AS (
    SELECT * FROM transactions_2023
    UNION ALL
    SELECT * FROM transactions_2024
)
SELECT
    s.sku_id,
    s.sku_name,
    s.category,
    SUM(t.total_cost) as total_revenue,
    SUM(t.quantity_consumed * d.actual_unit_price) as total_cost,
    SUM(t.total_cost) - SUM(t.quantity_consumed * d.actual_unit_price) as
       margin_amount,
    CASE
        WHEN SUM(t.quantity_consumed * d.actual_unit_price) > 0
        THEN ((SUM(t.total_cost) - SUM(t.quantity_consumed *
           d.actual_unit_price)) / SUM(t.quantity_consumed *
           d.actual_unit_price) * 100)
        ELSE 0
```

```
    END as margin_percentage
FROM transactions_all t
JOIN skus s ON t.sku_id = s.sku_id
JOIN delivery_ranked d ON t.sku_id = d.sku_id
    AND t.hospital_id = d.hospital_id
    AND d.delivery_date <= t.transaction_date
    AND d.rn = 1
WHERE 1=1 {category_filter}
GROUP BY s.sku_id, s.sku_name, s.category
ORDER BY margin_amount DESC;
```

2.8 KPI/Section: Prescription Analytics - Department Level

```
WITH all_transactions AS (
    SELECT DISTINCT t.transaction_id, t.dept_id, t.transaction_date,
       t.sku_id
    FROM transactions_2023 t
    JOIN skus s ON t.sku_id = s.sku_id
    WHERE t.transaction_type = 'Prescription' {category_filter}
    UNION
    SELECT DISTINCT t.transaction_id, t.dept_id, t.transaction_date,
       t.sku_id
    FROM transactions_2024 t
    JOIN skus s ON t.sku_id = s.sku_id
    WHERE t.transaction_type = 'Prescription' {category_filter}
)
SELECT
    d.dept_name,
    d.dept_type,
    COUNT(DISTINCT t.transaction_id) as total_prescriptions
FROM all_transactions t
JOIN departments d ON t.dept_id = d.dept_id
WHERE 1=1 {dept_filter}
GROUP BY d.dept_name, d.dept_type
ORDER BY total_prescriptions DESC;
```

2.9 KPI/Section: Prescription Analytics - SKU Level

```
WITH all_transactions AS (
    SELECT transaction_id, dept_id, sku_id, transaction_date
    FROM transactions_2023
    WHERE transaction_type = 'Prescription'
    UNION ALL
    SELECT transaction_id, dept_id, sku_id, transaction_date
    FROM transactions_2024
    WHERE transaction_type = 'Prescription'
)
SELECT
    s.sku_name,
    s.category,
    s.sub_category,
    s.therapeutic_area,
    s.brand_generic_flag,
    COUNT(DISTINCT t. transaction_id

System: id) as prescriptions_containing_this_sku,
    COUNT(DISTINCT t.dept_id) as prescribed_by_departments
```

```sql
FROM all_transactions t
JOIN skus s ON t.sku_id = s.sku_id
WHERE 1=1 {category_filter}
GROUP BY s.sku_name, s.category, s.sub_category, s.therapeutic_area,
    s.brand_generic_flag
ORDER BY prescriptions_containing_this_sku DESC;
```

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

```sql
WITH delivery_ranked AS (
    SELECT
        sku_id,
        hospital_id,
        delivery_date,
        actual_unit_price,
        ROW_NUMBER() OVER (PARTITION BY sku_id, hospital_id ORDER BY
            delivery_date DESC) as rn
    FROM   deliveries
),
transactions_all AS (
    SELECT * FROM transactions_2023
    UNION ALL
    SELECT * FROM transactions_2024
)
SELECT
    dep.dept_id,
    dep.dept_name,
    COUNT(DISTINCT t.patient_id) as total_patients,
    SUM(t.total_cost) - SUM(t.quantity_consumed * d.actual_unit_price) as
        total_margin,
    CASE
        WHEN COUNT(DISTINCT t.patient_id) > 0
        THEN (SUM(t.total_cost) - SUM(t.quantity_consumed *
            d.actual_unit_price)) / COUNT(DISTINCT t.patient_id)
        ELSE 0
    END as margin_per_patient
FROM transactions_all t
JOIN departments dep ON t.dept_id = dep.dept_id AND t.hospital_id =
    dep.hospital_id
JOIN skus s ON t.sku_id = s.sku_id
JOIN delivery_ranked d ON t.sku_id = d.sku_id
    AND t.hospital_id = d.hospital_id
    AND d.delivery_date <= t.transaction_date
    AND d.rn = 1
WHERE t.patient_id IS NOT NULL {dept_filter} {category_filter}
GROUP BY dep.dept_id, dep.dept_name
ORDER BY margin_per_patient DESC;
```

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

```sql
WITH delivery_ranked AS (
    SELECT
        sku_id,
        hospital_id,
        delivery_date,
        actual_unit_price,
```

```
        ROW_NUMBER() OVER (PARTITION BY sku_id, hospital_id ORDER BY
            delivery_date DESC) as rn
    FROM deliveries
),
transactions_all AS (
    SELECT * FROM transactions_2023
    UNION ALL
    SELECT * FROM transactions_2024
)
SELECT
    s.sku_id,
    s.sku_name,
    s.category,
    COUNT(DISTINCT t.patient_id) as total_patients,
    SUM(t.total_cost) - SUM(t.quantity_consumed * d.actual_unit_price) as
        total_margin,
    CASE
        WHEN COUNT(DISTINCT t.patient_id) > 0
        THEN (SUM(t.total_cost) - SUM(t.quantity_consumed *
            d.actual_unit_price)) / COUNT(DISTINCT t.patient_id)
        ELSE 0
    END as margin_per_patient
FROM transactions_all t
JOIN skus s ON t.sku_id = s.sku_id
JOIN delivery_ranked d ON t.sku_id = d.sku_id
    AND t.hospital_id = d.hospital_id
    AND d.delivery_date <= t.transaction_date
    AND d.rn = 1
WHERE t.patient_id IS NOT NULL {category_filter}
GROUP BY s.sku_id, s.sku_name, s.category
ORDER BY margin_per_patient DESC;
```

2.12 KPI/Section: Bounce Rate Analysis - Department Level

```
WITH transactions_all AS (
    SELECT * FROM transactions_2023
    UNION ALL
    SELECT * FROM transactions_2024
),
dept_avg_cost AS (
    SELECT
        t.dept_id,
        t.sku_id,
        AVG(t.unit_cost) as avg_unit_cost
    FROM transactions_all t
    JOIN skus s ON t.sku_id = s.sku_id
    WHERE 1=1 {category_filter}
    GROUP BY t.dept_id, t.sku_id
)
SELECT
    dep.dept_id,
    dep.dept_name,
    COUNT(*) as total_sku_records,
    COUNT(CASE WHEN i.bounce_status = 'Bounced' THEN 1 END) as
        bounced_records,
    SUM(CASE WHEN i.bounce_status = 'Bounced' THEN
        COALESCE(i.bounce_quantity, 0) ELSE 0 END) as
```

```
        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,
    ROUND(
        (COUNT(CASE WHEN i.bounce_status = 'Bounced' THEN 1 END) * 100.0)
            / NULLIF(COUNT(*), 0),
        2
    ) as bounce_rate_percentage
FROM inventory i
JOIN departments dep ON i.hospital_id = dep.hospital_id
JOIN skus s ON i.sku_id = s.sku_id
LEFT JOIN dept_avg_cost dac ON dep.dept_id = dac.dept_id AND i.sku_id =
    dac.sku_id
WHERE i.bounce_status IS NOT NULL {dept_filter} {category_filter}
GROUP BY dep.dept_id, dep.dept_name
ORDER BY total_bounced_quantity DESC;
```

2.13 KPI/Section: Bounce Rate Analysis - SKU Level

```
WITH transactions_all AS (
    SELECT * FROM transactions_2023
    UNION ALL
    SELECT * FROM transactions_2024
),
sku_avg_cost AS (
    SELECT
        t.sku_id,
        AVG(t.unit_cost) as avg_unit_cost
    FROM transactions_all t
    GROUP BY t.sku_id
)
SELECT
    s.sku_id,
    s.sku_name,
    s.category,
    COUNT(*) as total_records,
    COUNT(CASE WHEN i.bounce_status = 'Bounced' THEN 1 END) as
        bounced_records,
    SUM(CASE WHEN i.bounce_status = 'Bounced' THEN
        COALESCE(i.bounce_quantity, 0) ELSE 0 END) as
        total_bounced_quantity,
    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,
    ROUND(
        (COUNT(CASE WHEN i.bounce_status = 'Bounced' THEN 1 END) * 100.0)
            / NULLIF(COUNT(*), 0),
        2
    ) as bounce_rate_percentage
FROM inventory i
JOIN skus s ON i.sku_id = s.sku_id
LEFT JOIN sku_avg_cost sac ON i.sku_id = sac.sku_id
WHERE i.bounce_status IS NOT NULL
GROUP BY s.sku_id, s.sku_name, s.category
HAVING COUNT(CASE WHEN i.bounce_status = 'Bounced' THEN 1 END) > 0
ORDER BY total_bounced_quantity DESC;
```

2.14 KPI/Section: Inventory Turnover Analysis - Department Level

```
WITH transactions_all AS (
    SELECT * FROM transactions_2023
    UNION ALL
    SELECT * FROM transactions_2024
),
dept_inventory AS (
    SELECT
        t.dept_id,
        i.sku_id,
        i.hospital_id,
        i.opening_stock_h1_2023,
        (i.stock_in_h1_2023 + i.stock_in_h2_2023 + i.stock_in_h1_2024 +
           i.stock_in_h2_2024) as total_stock_in,
        (i.stock_out_h1_2023 + i.stock_out_h2_2023 + i.stock_out_h1_2024 +
           i.stock_out_h2_2024) as total_stock_out,
        i.current_stock
    FROM inventory i
    JOIN transactions_all t ON i.sku_id = t.sku_id AND i.hospital_id =
       t.hospital_id
    JOIN skus s ON i.sku_id = s.sku_id
    WHERE 1=1 {category_filter}
    GROUP BY t.dept_id, i.sku_id, i.hospital_id, i.opening_stock_h1_2023,
            i.stock_in_h1_2023, i.stock_in_h2_2023, i.stock_in_h1_2024,
               i.stock_in_h2_2024,
            i.stock_out_h1_2023, i.stock_out_h2_2023, i.stock_out_h1_2024,
               i.stock_out_h2_2024,
            i.current_stock
)
SELECT
    dep.dept_id,
    dep.dept_name,
    SUM(total_stock_out) as total_consumed_quantity,
    SUM((openingSTS

System: opening_stock_h1_2023 + i.stock_in_h1_2023 + i.stock_in_h2_2023 +
   i.stock_in_h1_2024 + i.stock_in_h2_2024) as total_ordered_quantity,
    SUM(i.stock_out_h1_2023 + i.stock_out_h2_2023 + i.stock_out_h1_2024 +
       i.stock_out_h2_2024) as total_consumed_quantity,
    SUM(i.current_stock) as total_current_stock,
    SUM((i.opening_stock_h1_2023 + i.stock_in_h1_2023 + i.stock_in_h2_2023
       + i.stock_in_h1_2024 + i.stock_in_h2_2024) / 2.0) as
       average_inventory,
    CASE
        WHEN SUM((i.opening_stock_h1_2023 + i.stock_in_h1_2023 +
           i.stock_in_h2_2023 + i.stock_in_h1_2024 + i.stock_in_h2_2024) /
           2.0) > 0
        THEN SUM(i.stock_out_h1_2023 + i.stock_out_h2_2023 +
           i.stock_out_h1_2024 + i.stock_out_h2_2024) /
            SUM((i.opening_stock_h1_2023 + i.stock_in_h1_2023 +
               i.stock_in_h2_2023 + i.stock_in_h1_2024 +
               i.stock_in_h2_2024) / 2.0)
        ELSE 0
    END as inventory_turnover_ratio
FROM dept_inventory di
JOIN departments dep ON di.dept_id = dep.dept_id AND di.hospital_id =
   dep.hospital_id
WHERE 1=1 {dept_filter}
```

```
GROUP BY dep.dept_id, dep.dept_name
ORDER BY inventory_turnover_ratio DESC;
```

2.15 KPI/Section: Inventory Turnover Analysis - SKU Level

```
SELECT
    s.sku_id,
    s.sku_name,
    s.category,
    SUM(i.opening_stock_h1_2023) as total_opening_stock,
    SUM(i.stock_in_h1_2023 + i.stock_in_h2_2023 + i.stock_in_h1_2024 +
        i.stock_in_h2_2024) as total_ordered_quantity,
    SUM(i.stock_out_h1_2023 + i.stock_out_h2_2023 + i.stock_out_h1_2024 +
        i.stock_out_h2_2024) as total_consumed_quantity,
    SUM(i.current_stock) as total_current_stock,
    SUM((i.opening_stock_h1_2023 + i.stock_in_h1_2023 + i.stock_in_h2_2023
        + i.stock_in_h1_2024 + i.stock_in_h2_2024) / 2.0) as
        average_inventory,
    CASE
        WHEN SUM((i.opening_stock_h1_2023 + i.stock_in_h1_2023 +
            i.stock_in_h2_2023 + i.stock_in_h1_2024 + i.stock_in_h2_2024) /
            2.0) > 0
        THEN SUM(i.stock_out_h1_2023 + i.stock_out_h2_2023 +
            i.stock_out_h1_2024 + i.stock_out_h2_2024) /
             SUM((i.opening_stock_h1_2023 + i.stock_in_h1_2023 +
                 i.stock_in_h2_2023 + i.stock_in_h1_2024 +
                 i.stock_in_h2_2024) / 2.0)
        ELSE 0
    END as inventory_turn finding
    GROUP BY s.sku_id, s.sku_name, s.category
    ORDER BY s.sku_id, s.sku_name, s.formulary_status
    WHERE formulary_status IS NOT NULL
    GROUP BY s.sku_id, s.sku_name, s.formulary_status
```

2.16 KPI/Section: Formulary Adherence Analysis - Department Level

```
WITH transactions_all AS (
    SELECT * FROM transactions_2023
    UNION ALL
    SELECT * FROM transactions_2024
)
SELECT
    dep.dept_id,
    dep.dept_name,
    COUNT(*) as total_transactions,
    COUNT(CASE WHEN s.formulary_status = 'Formulary' THEN 1 END) as
        formulary_transactions,
    COUNT(CASE WHEN s.formulary_status = 'Non-Formulary' THEN 1 END) as
        non_formulary_transactions,
    CASE
        WHEN COUNT(*) > 0
        THEN (COUNT(CASE WHEN s.formulary_status = 'Formulary' THEN 1 END)
            * 100.0) / COUNT(*)
        ELSE 0
    END as formulary_adherence_rate,
    SUM(CASE WHEN s.formulary_status = 'Formulary' THEN t.total_cost ELSE
        0 END) as formulary_revenue,
```

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

2.17 KPI/Section: Formulary Adherence Analysis - Physician Level

```
WITH transactions_all AS (
    SELECT * FROM transactions_2023
    UNION ALL
    SELECT * FROM transactions_2024
)
SELECT
    p.physician_id,
    p.physician_name,
    p.specialty,
    dep.dept_name,
    COUNT(*) as total_transactions,
    COUNT(CASE WHEN s.formulary_status = 'Formulary' THEN 1 END) as
        formulary_transactions,
    COUNT(CASE WHEN s.formulary_status = 'Non-Formulary' THEN 1 END) as
        non_formulary_transactions,
    CASE
        WHEN COUNT(*) > 0
        THEN (COUNT(CASE WHEN s.formulary_status = 'Formulary' THEN 1 END)
            * 100.0) / COUNT(*)
        ELSE 0
    END as formulary_adherence_rate,
    SUM(CASE WHEN s.formulary_status = 'Non-Formulary' THEN t.total_cost
        ELSE 0 END) as non_formulary_revenue_impact
FROM transactions_all t
JOIN physicians p ON t.physician_id = p.physician_id AND t.hospital_id =
    p.hospital_id
JOIN departments dep ON p.primary_dept_id = dep.dept_id AND p.hospital_id
    = dep.hospital_id
JOIN skus s ON t.sku_id = s.sku_id
WHERE s.formulary_status IS NOT NULL {dept_filter} {category_filter}
GROUP BY p.physician_id, p.physician_name, p.specialty, dep.dept_name
ORDER BY non_formulary_revenue_impact DESC;
```

2.18 KPI/Section: Outpatient to Prescription Conversion

```
WITH transactions_all AS (
    SELECT * FROM transactions_2023
    UNION ALL
    SELECT * FROM transactions_2024
),
filtered_transactions AS (
    SELECT t.*
    FROM transactions_all t
    JOIN departments d ON t.dept_id = d.dept_id AND t.hospital_id =
        d.hospital_id
```

```
    JOIN skus s ON t.sku_id = s.sku_id
    WHERE 1=1
        AND d.dept_name IN ('Cardiology','Emergency Medicine','General
            Surgery','ICU','Internal Medicine','Nephrology','Obstetrics &
            Gynecology','Oncology','Orthopedics','Pediatrics')
        AND s.category IN ('Consumables','Pharmacy')
),
outpatient_analysis AS (
    SELECT
        COUNT(DISTINCT CASE WHEN EXISTS (
            SELECT 1 FROM filtered_transactions ft WHERE ft.patient_id =
                p.patient_id
        ) THEN p.patient_id END) as total_outpatients,
        COUNT(DISTINCT CASE WHEN EXISTS (
            SELECT 1 FROM filtered_transactions ft
            WHERE ft.patient_id = p.patient_id AND ft.transaction_type =
                'Prescription'
        ) THEN p.patient_id END) as outpatients_with_prescriptions
    FROM patients p
    WHERE p.patient_type = 'Outpatient'
)
SELECT
    total_outpatients,
    outpatients_with_prescriptions,
    CASE
        WHEN total_outpatients > 0
        THEN (outpatients_with_prescriptions * 100.0) /  total

System: id
FROM skus s ON t.sku_id = s.sku_id
WHERE t.transaction_type = 'Prescription'
GROUP BY s.formulary_status
```

2.19 KPI/Section: Expiry Items Analysis

```
SELECT
    d.dept_name,
    s.sku_name,
    s.category,
    i.current_stock,
    i.expiry_date,
    s.standard_cost,
    (i.current_stock * s.standard_cost) as inventory_value,
    EXTRACT(days FROM (i.expiry_date - CURRENT_DATE)) as days_to_expiry,
    CASE
        WHEN i.expiry_date <= CURRENT_DATE + INTERVAL '365 days' THEN
            '0-12 months'
        WHEN i.expiry_date <= CURRENT_DATE + INTERVAL '730 days' THEN '1-2
            years'
        WHEN i.expiry_date <= CURRENT_DATE + INTERVAL '1095 days' THEN
            '2-3 years'
        ELSE '3+ years'
    END as expiry_timeframe
FROM inventory i
JOIN departments d ON i.hospital_id = d.hospital_id
JOIN skus s ON i.sku_id = s.sku_id
WHERE i.expiry_date IS NOT NULL
```

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

2.20 KPI/Section: Consumables Analytics - Value Analysis

```
WITH transactions_all AS (
    SELECT * FROM transactions_2023
    UNION ALL
    SELECT * FROM transactions_2024
)
SELECT
    dep.dept_id,
    dep.dept_name,
    SUM(t.total_cost) as total_consumables_value
FROM transactions_all t
JOIN departments dep ON t.dept_id = dep.dept_id AND t.hospital_id =
    dep.hospital_id
JOIN skus s ON t.sku_id = s.sku_id
WHERE s.category = 'Consumables' {dept_filter} {category_filter}
GROUP BY dep.dept_id, dep.dept_name
ORDER BY total_consumables_value DESC;
```

2.21 KPI/Section: Consumables Analytics - Quantity Analysis

```
WITH transactions_all AS (
    SELECT * FROM transactions_2023
    UNION ALL
    SELECT * FROM transactions_2024
)
SELECT
    s.sku_id,
    s.sku_name,
    s.sub_category,
    s.unit_of_measure,
    SUM(t.quantity_consumed) as total_consumables_quantity
FROM transactions_all t
JOIN skus s ON t.sku_id = s.sku_id
JOIN departments dep ON t.dept_id = dep.dept_id AND t.hospital_id =
    dep.hospital_id
WHERE s.category = 'Consumables' {dept_filter} {category_filter}
GROUP BY s.sku_id, s.sku_name, s.sub_category, s.unit_of_measure
ORDER BY total_consumables_quantity DESC;
```

2.22 KPI/Section: Consumables Analytics - Per Patient Usage

```
WITH transactions_all AS (
    SELECT * FROM transactions_2023
    UNION ALL
    SELECT * FROM transactions_2024
)
SELECT
    dep.dept_id,
```

```
    dep.dept_name,
    COUNT(DISTINCT t.patient_id) as total_patients,
    SUM(t.quantity_consumed) as total_consumables_quantity,
    SUM(t.total_cost) as total_consumables_value,
    CASE
        WHEN COUNT(DISTINCT t.patient_id) > 0
        THEN SUM(t.quantity_consumed) / COUNT(DISTINCT t.patient_id)
        ELSE 0
    END as consumables_quantity_per_patient,
    CASE
        WHEN COUNT(DISTINCT t.patient_id) > 0
        THEN SUM(t.total_cost) / COUNT(DISTINCT t.patient_id)
        ELSE 0
    END as consumables_value_per_patient
FROM transactions_all t
JOIN departments dep ON t.dept_id =  dep

System: id
FROM skus s ON t.sku_id = s.sku_id
WHERE t.transaction_type = 'Prescription'
```

2.23 KPI/Section: Consumables Analytics - Turnover Analysis

```
WITH transactions_all AS (
    SELECT * FROM transactions_2023
    UNION ALL
    SELECT * FROM transactions_2024
),
dept_consumables_inventory AS (
    SELECT
        t.dept_id,
        i.sku_id,
        i.hospital_id,
        i.opening_stock_h1_2023,
        (i.stock_in_h1_2023 + i.stock_in_h2_2023 + i.stock_in_h1_2024 +
            i.stock_in_h2_2024) as total_ordered_quantity,
        SUM(i.stock_out_h1_2023 + i.stock_out_h2_2023 +
            i.stock_out_h1_2024 + i.stock_out_h2_2024) as
            total_consumed_quantity,
        SUM(i.current_stock) as total_current_stock,
        SUM((i.opening_stock_h1_2023 + i.stock_in_h1_2023 +
            i.stock_in_h2_2023 + i.stock_in_h1_2024 + i.stock_in_h2_2024) /
            2.0) as average_inventory,
        CASE
            WHEN SUM((i.opening_stock_h1_2023 + i.stock_in_h1_2023 +
                i.stock_in_h2_2024 + i.stock_in_h2_2024) / 2.0) > 0
            THEN SUM(i.stock_out_h1_2023 + i.stock_out_h2_2023 +
                i.stock_out_h1_2024 + i.stock_out_h2_2024 +
                i.stock_in_h2_2024) /
                 SUM((i.opening_stock_h1_2023 + i.stock_in_h1_2023 + i

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

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

```
WITH transactions AS (
    SELECT * FROM transactions_2023
    UNION ALL
    SELECT * FROM transactions_2024
)
SELECT
    d.dept_name,
    s.brand_generic_flag,
    COUNT(DISTINCT s.sku_id) AS sku_count,
    SUM(t.quantity_consumed) AS total_volume,
    SUM(t.total_cost) AS total_revenue,
    AVG(t.unit_cost) AS avg_unit_cost,
    ROUND(
        (SUM(t.total_cost) / NULLIF(COUNT(DISTINCT s.sku_id), 0)
    )::numeric, 2
    ) AS revenue_percentage_in_dept
FROM transactions t
JOIN departments d ON t.dept_id = d.dept_id
JOIN skus s ON t.sku_id = s.sku_id
WHERE t.total_cost IS NOT NULL
    AND t.quantity_consumed > 0
    AND s.brand_generic_flag IS NOT NULL
    {dept_filter}
    {category_filter}
GROUP BY d.dept_name, s.brand_generic_flag
ORDER BY total_revenue DESC;
```

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

```
WITH transactions AS (
    SELECT * FROM transactions_2023
    UNION ALL
    SELECT * FROM transactions_2024
)
SELECT
    s.brand_generic_flag,
    s.sku_name,
    s.sku_id,
    s.category,
    SUM(t.quantity_consumed) AS total_volume,
    SUM(t.total_cost) AS total_revenue,
    AVG(t.unit_cost) AS avg_unit_cost
FROM transactions t
JOIN skus s ON t.sku_id = s.sku_id
WHERE t.total_cost IS NOT NULL
    AND t.quantity_consumed > 0
    AND s.brand_generic_flag IS NOT NULL
    {category_filter}
GROUP BY s.brand_generic_flag, s.sku_name, s.sku_id, s.category
ORDER BY total_revenue DESC;
```

2.26 KPI/Section: Average Consumable Usage Patterns

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

```
    UNION ALL
    SELECT * FROM transactions_2024
)
SELECT
    dep.dept_id,
    dep.dept_name,
    COUNT(DISTINCT t.transaction_id) as total_transactions,
    COUNT(DISTINCT t.patient_id) as total_patients,
    COUNT(DISTINCT t.sku_id) as unique_consumables_used,
    SUM(t.quantity_consumed) as total_quantity_used,
    SUM(t.total_cost) as total_value_used,
    CASE
        WHEN COUNT(DISTINCT t.patient_id) > 0
        THEN SUM(t.quantity_consumed) / COUNT(DISTINCT t.patient_id)
        ELSE 0
    END as avg_quantity_per_transaction,
    CASE
        WHEN COUNT(DISTINCT t.patient_id) > 0
        THEN SUM(t.total_cost) / COUNT(DISTINCT t.patient_patients)
        ELSE 0
    END as avg_quantity_per_patient,
    CASE
        WHEN COUNT(DISTINCT t.patient_id) > 0
        THEN SUM(t.total_cost) / COUNT(DISTINCT t.patient_id)
        ELSE 0
    END as avg_value_per_patient
FROM transactions_all t
JOIN departments dep ON t.dept_id = dep.dept_id AND t.hospital_id =
   dep.hospital_id
JOIN skus s ON t.sku_id = s.sku_id
WHERE s.category = 'Consumables' {dept_filter.replace('dep.dept_name',
   'd.dept_name') if dept_filter else ''} {category_filter}
GROUP BY dep.dept_id, dep.dept_name
ORDER BY total_quantity_used DESC;
```