# 03_gold_bi_analysis

January 1, 2026

### 0.1 About the Author

This report was authored by Ayoub Majjid, a fifth-year computer engineering student at EMSI with a background in Experimental Sciences. His academic journey has provided a strong foundation in mathematics, physics, and chemistry, and has shaped a growing expertise in technology, system design, and data engineering.

Ayoub currently serves as Tech Lead and Entrepreneur at Intellcap, where he leads three innovation projects focused on building impactful and scalable startup solutions. His work emphasizes transforming ideas into robust technical systems, with a particular interest in data platforms, system architecture, and end-to-end engineering workflows.

Email: ayoub@majjid.com Website: https://majjid.com

---

### 0.2 Copyright & Disclaimer

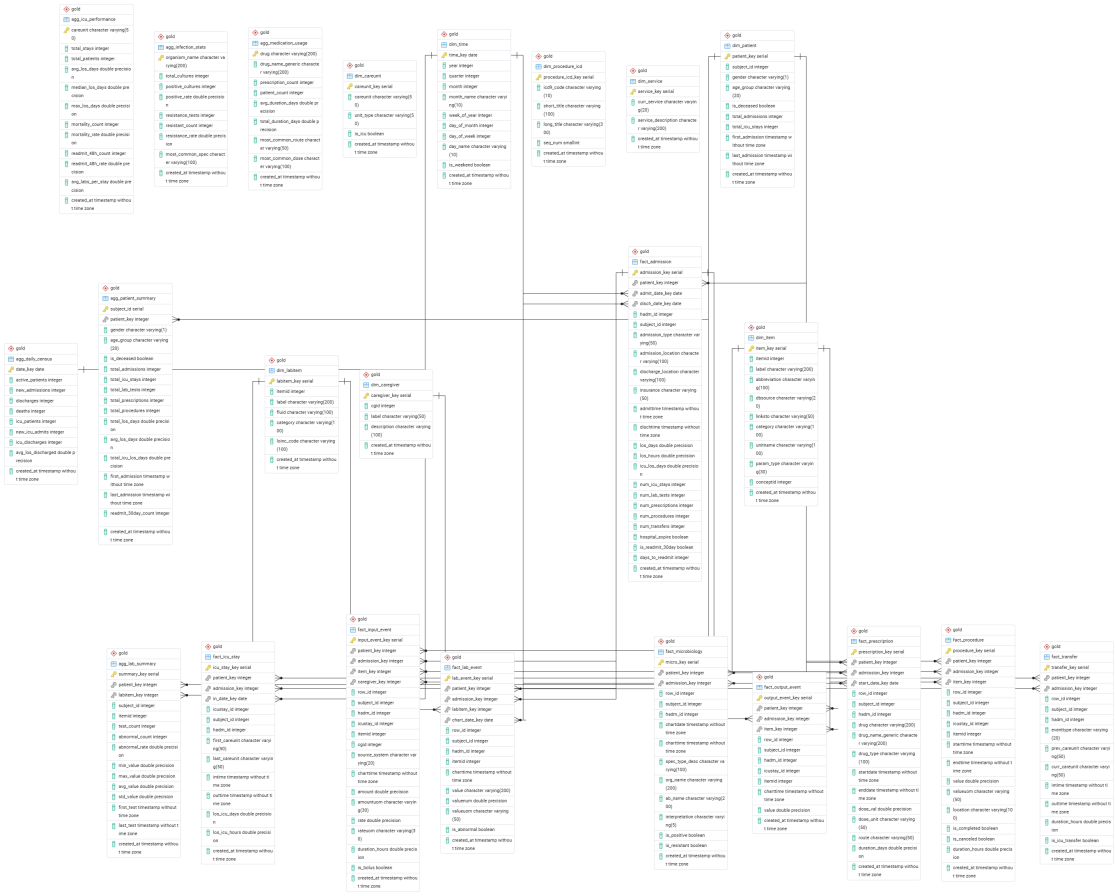


---

## 1 MIMIC-III Gold Layer Business Intelligence Report

**Professional BI & Decision Support Report**
**Date:** January 2026
**Version:** 2.0

---



## 1.1 Executive Summary

Comprehensive business intelligence analysis from the Gold layer data warehouse with actionable insights.

### 1.1.1 Report Highlights:

- KPI dashboard with key metrics
- Patient outcomes and correlations
- ICU performance benchmarks
- Caregiver workload analysis
- Medication and lab insights
- Actionable recommendations

[ ]:
```
# Setup
import sys
sys.path.insert(0, '../../..')

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from sqlalchemy import text
from app.shared import get_db

# Configuration
pd.set_option('display.max_columns', None)
pd.set_option('display.float_format', '{:.2f}'.format)
plt.style.use('seaborn-v0_8-whitegrid')
plt.rcParams['figure.dpi'] = 100

# Colors
COLORS = {
    'primary': '#2E86AB', 'secondary': '#A23B72', 'success': '#28A745',
    'warning': '#FFC107', 'danger': '#DC3545', 'info': '#17A2B8'
}

print(" Setup complete")
```

```
Setup complete
```

[ ]:
```
# Utility functions
def query_df(sql, limit=None):
    with get_db() as session:
        result = session.execute(text(sql))
        df = pd.DataFrame(result.fetchall(), columns=result.keys())
    return df.head(limit) if limit else df

def table_count(schema, table):
    with get_db() as session:
        return session.execute(text(f"SELECT COUNT(*) FROM {schema}.{table}")).
 ↪scalar()

print(" Functions ready")
```

```
Functions ready
```

---

## 1.2 1. Executive KPI Dashboard

[10]:
```
# Calculate KPIs
kpis = {
    'Patients': table_count('gold', 'dim_patient'),
    'Admissions': table_count('gold', 'fact_admission'),
    'ICU Stays': table_count('gold', 'fact_icu_stay'),
    'Lab Tests': table_count('gold', 'fact_lab_event'),
    'Prescriptions': table_count('gold', 'fact_prescription'),
    'Input Events': table_count('gold', 'fact_input_event'),
}
```

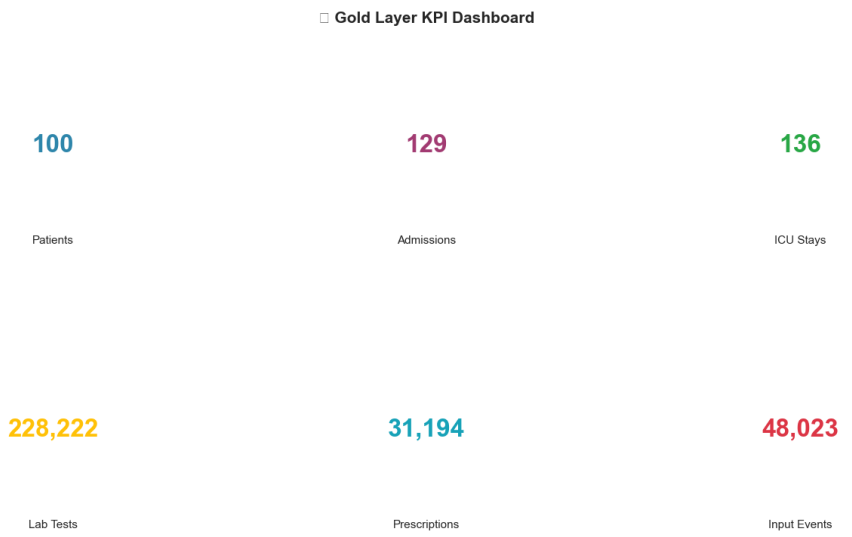
```python
print("  KEY PERFORMANCE INDICATORS")
print("="*50)
for metric, value in kpis.items():
    print(f" {metric:.<30} {value:>15,}")
print("="*50)
```

```
2026-01-01 14:28:45,475 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2026-01-01 14:28:45,476 INFO sqlalchemy.engine.Engine SELECT COUNT(*) FROM
gold.dim_patient
2026-01-01 14:28:45,477 INFO sqlalchemy.engine.Engine [cached since 783.7s ago]
{}
2026-01-01 14:28:45,480 INFO sqlalchemy.engine.Engine COMMIT
2026-01-01 14:28:45,484 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2026-01-01 14:28:45,485 INFO sqlalchemy.engine.Engine SELECT COUNT(*) FROM
gold.fact_admission
2026-01-01 14:28:45,486 INFO sqlalchemy.engine.Engine [cached since 783.7s ago]
{}
2026-01-01 14:28:45,490 INFO sqlalchemy.engine.Engine COMMIT
2026-01-01 14:28:45,495 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2026-01-01 14:28:45,496 INFO sqlalchemy.engine.Engine SELECT COUNT(*) FROM
gold.fact_icu_stay
2026-01-01 14:28:45,498 INFO sqlalchemy.engine.Engine [cached since 783.7s ago]
{}
2026-01-01 14:28:45,501 INFO sqlalchemy.engine.Engine COMMIT
2026-01-01 14:28:45,504 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2026-01-01 14:28:45,505 INFO sqlalchemy.engine.Engine SELECT COUNT(*) FROM
gold.fact_lab_event
2026-01-01 14:28:45,506 INFO sqlalchemy.engine.Engine [cached since 783.7s ago]
{}
2026-01-01 14:28:45,529 INFO sqlalchemy.engine.Engine COMMIT
2026-01-01 14:28:45,532 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2026-01-01 14:28:45,533 INFO sqlalchemy.engine.Engine SELECT COUNT(*) FROM
gold.fact_prescription
2026-01-01 14:28:45,534 INFO sqlalchemy.engine.Engine [cached since 783.6s ago]
{}
2026-01-01 14:28:45,540 INFO sqlalchemy.engine.Engine COMMIT
2026-01-01 14:28:45,544 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2026-01-01 14:28:45,545 INFO sqlalchemy.engine.Engine SELECT COUNT(*) FROM
gold.fact_input_event
2026-01-01 14:28:45,546 INFO sqlalchemy.engine.Engine [cached since 783.6s ago]
{}
2026-01-01 14:28:45,553 INFO sqlalchemy.engine.Engine COMMIT
  KEY PERFORMANCE INDICATORS
==================================================
 Patients…                100
 Admissions…              129
 ICU Stays…               136
 Lab Tests…           228,222
```

```
 Prescriptions…          31,194
 Input Events…          48,023
================================================
```

[11]:
```python
# KPI Dashboard Visualization
fig, axes = plt.subplots(2, 3, figsize=(15, 8))
axes = axes.flatten()

colors = [COLORS['primary'], COLORS['secondary'], COLORS['success'],
          COLORS['warning'], COLORS['info'], COLORS['danger']]

for i, (metric, value) in enumerate(kpis.items()):
    ax = axes[i]
    # Main number
    ax.text(0.5, 0.6, f"{value:,}", ha='center', va='center',
            fontsize=24, fontweight='bold', color=colors[i])
    # Label
    ax.text(0.5, 0.25, metric, ha='center', va='center', fontsize=11)
    ax.set_xlim(0, 1)
    ax.set_ylim(0, 1)
    ax.axis('off')
    ax.set_facecolor('#f8f9fa')

plt.suptitle('  Gold Layer KPI Dashboard', fontsize=15, fontweight='bold')
plt.tight_layout()
plt.show()

print("\n  Dashboard displays key metrics with values clearly labeled")
```

```
C:\Users\ayoub\AppData\Local\Temp\ipykernel_20120\677564994.py:21: UserWarning:
Glyph 128202 (\N{BAR CHART}) missing from font(s) Arial.
  plt.tight_layout()
e:\vs-code\github\Medical-Data-Mining\data-warehouse\project\venv\Lib\site-
packages\IPython\core\pylabtools.py:170: UserWarning: Glyph 128202 (\N{BAR
CHART}) missing from font(s) Arial.
  fig.canvas.print_figure(bytes_io, **kw)
```




Dashboard displays key metrics with values clearly labeled

---

### 1.3 2. Patient Analytics

[12]:
```
# Patient analysis by gender
patient_analysis = query_df("""
    SELECT
        gender,
        COUNT(*) as total,
        SUM(CASE WHEN is_deceased THEN 1 ELSE 0 END) as deceased,
        ROUND(100.0 * SUM(CASE WHEN is_deceased THEN 1 ELSE 0 END) / COUNT(*),␣
 ↪1) as mortality_pct,
        ROUND(AVG(total_admissions)::numeric, 1) as avg_admissions,
        ROUND(AVG(total_icu_stays)::numeric, 1) as avg_icu_stays
    FROM gold.dim_patient
    GROUP BY gender
""")

print("  PATIENT ANALYTICS BY GENDER")
print(patient_analysis.to_string(index=False))
```

```
2026-01-01 14:28:54,185 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2026-01-01 14:28:54,187 INFO sqlalchemy.engine.Engine
    SELECT
        gender,
        COUNT(*) as total,
```

```
        SUM(CASE WHEN is_deceased THEN 1 ELSE 0 END) as deceased,
        ROUND(100.0 * SUM(CASE WHEN is_deceased THEN 1 ELSE 0 END) / COUNT(*),
1) as mortality_pct,
        ROUND(AVG(total_admissions)::numeric, 1) as avg_admissions,
        ROUND(AVG(total_icu_stays)::numeric, 1) as avg_icu_stays
    FROM gold.dim_patient
    GROUP BY gender

2026-01-01 14:28:54,188 INFO sqlalchemy.engine.Engine [generated in 0.00121s] {}
2026-01-01 14:28:54,193 INFO sqlalchemy.engine.Engine COMMIT
 PATIENT ANALYTICS BY GENDER
gender  total  deceased mortality_pct avg_admissions avg_icu_stays
     M     45        45         100.0            1.6           1.6
     F     55        55         100.0            1.1           1.1
```

[ ]:
```
# Correlation: Gender vs Outcomes (WITH VALUE LABELS)
fig, axes = plt.subplots(1, 3, figsize=(15, 5))

# Mortality by gender
ax1 = axes[0]
bars1 = ax1.bar(patient_analysis['gender'], patient_analysis['mortality_pct'],
                color=[COLORS['danger'], COLORS['primary']])
ax1.set_ylabel('Mortality Rate (%)', fontweight='bold')
ax1.set_title('  Mortality Rate by Gender', fontweight='bold')
ax1.set_ylim(0, 110)
# Add labels
for bar, row in zip(bars1, patient_analysis.itertuples()):
    height = bar.get_height()
    ax1.text(bar.get_x() + bar.get_width()/2., height + 2,
            f'{height:.1f}%\n({row.deceased}/{row.total})',
            ha='center', va='bottom', fontweight='bold')

# Average admissions by gender
ax2 = axes[1]
bars2 = ax2.bar(patient_analysis['gender'], patient_analysis['avg_admissions'],
                color=[COLORS['warning'], COLORS['info']])
ax2.set_ylabel('Avg Admissions', fontweight='bold')
ax2.set_title('  Avg Admissions by Gender', fontweight='bold')
# Add labels
for bar in bars2:
    height = bar.get_height()
    ax2.text(bar.get_x() + bar.get_width()/2., height,
            f'{height:.1f}', ha='center', va='bottom', fontweight='bold')

# Average ICU stays by gender
ax3 = axes[2]
bars3 = ax3.bar(patient_analysis['gender'], patient_analysis['avg_icu_stays'],
```

```
                color=[COLORS['secondary'], COLORS['success']])
ax3.set_ylabel('Avg ICU Stays', fontweight='bold')
ax3.set_title('  Avg ICU Stays by Gender', fontweight='bold')
# Add labels
for bar in bars3:
    height = bar.get_height()
    ax3.text(bar.get_x() + bar.get_width()/2., height,
            f'{height:.1f}', ha='center', va='bottom', fontweight='bold')

plt.suptitle('  Patient Outcomes by Gender (with Labels)', fontsize=14,␣
 ↪fontweight='bold')
plt.tight_layout()
plt.show()

print("\n  All bars show values above columns")
```

---

### 1.4 3. ICU Performance Analysis

[13]:
```
# ICU performance metrics
icu_perf = query_df("""
    SELECT
        careunit,
        total_stays,
        total_patients,
        avg_los_days
    FROM gold.agg_icu_performance
    ORDER BY total_stays DESC
""")

print("  ICU PERFORMANCE METRICS")
print(icu_perf.to_string(index=False))
```

```
2026-01-01 14:28:58,691 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2026-01-01 14:28:58,693 INFO sqlalchemy.engine.Engine
    SELECT
        careunit,
        total_stays,
        total_patients,
        avg_los_days
    FROM gold.agg_icu_performance
    ORDER BY total_stays DESC

2026-01-01 14:28:58,694 INFO sqlalchemy.engine.Engine [generated in 0.00116s] {}
2026-01-01 14:28:58,714 INFO sqlalchemy.engine.Engine COMMIT
  ICU PERFORMANCE METRICS
careunit  total_stays  total_patients  avg_los_days
```

```
   MICU          77              56          3.96
   SICU          23              20          5.67
    CCU          19              18          5.75
  TSICU          11              11          3.59
   CSRU           6               6          3.63
```

[ ]:
```python
# ICU visualizations WITH LABELS
fig, axes = plt.subplots(1, 2, figsize=(14, 6))

# Stays by unit
ax1 = axes[0]
bars1 = ax1.barh(icu_perf['careunit'], icu_perf['total_stays'],
                 color=sns.color_palette('Blues_r', len(icu_perf)))
ax1.set_xlabel('Total ICU Stays', fontweight='bold')
ax1.set_title('  ICU Stays by Unit\n(with Counts)', fontweight='bold')
# Add labels
for bar in bars1:
    width = bar.get_width()
    ax1.text(width, bar.get_y() + bar.get_height()/2.,
            f'  {int(width):,}',
            ha='left', va='center', fontweight='bold',
            bbox=dict(boxstyle='round,pad=0.3', facecolor='white', alpha=0.9))

# Average LOS by unit
ax2 = axes[1]
bars2 = ax2.barh(icu_perf['careunit'], icu_perf['avg_los_days'],
                 color=sns.color_palette('Oranges_r', len(icu_perf)))
ax2.set_xlabel('Average LOS (Days)', fontweight='bold')
ax2.set_title('  Average LOS by ICU\n(with Days)', fontweight='bold')
# Add labels
for bar in bars2:
    width = bar.get_width()
    ax2.text(width, bar.get_y() + bar.get_height()/2.,
            f'  {width:.1f}d',
            ha='left', va='center', fontweight='bold',
            bbox=dict(boxstyle='round,pad=0.3', facecolor='white', alpha=0.9))

plt.suptitle('  ICU Performance Analysis', fontsize=14, fontweight='bold')
plt.tight_layout()
plt.show()

print("\n  Value labels on all horizontal bars")
print("\n  Decision Insight:")
if len(icu_perf) > 0:
    busiest = icu_perf.iloc[0]
    print(f"   • Busiest Unit: {busiest['careunit']} ({busiest['total_stays']:
 ↪,} stays)")
```

### 1.5 4. Caregiver Workload Analysis

[14]:
```
# Caregiver dimension
try:
    cg_count = table_count('gold', 'dim_caregiver')
    print(f"  Total Caregivers: {cg_count:,}")

    # Workload from input events
    cg_workload = query_df("""
        SELECT
            COALESCE(dc.label, 'Unknown') as role,
            COUNT(*) as events,
            COUNT(DISTINCT f.subject_id) as patients
        FROM gold.fact_input_event f
        LEFT JOIN gold.dim_caregiver dc ON f.caregiver_key = dc.caregiver_key
        GROUP BY dc.label
        ORDER BY events DESC
        LIMIT 10
    """)

    print("\n  CAREGIVER WORKLOAD (Top 10)")
    print(cg_workload.to_string(index=False))

except Exception as e:
    print(f"  Caregiver data not available: {e}")
    cg_workload = pd.DataFrame()
```

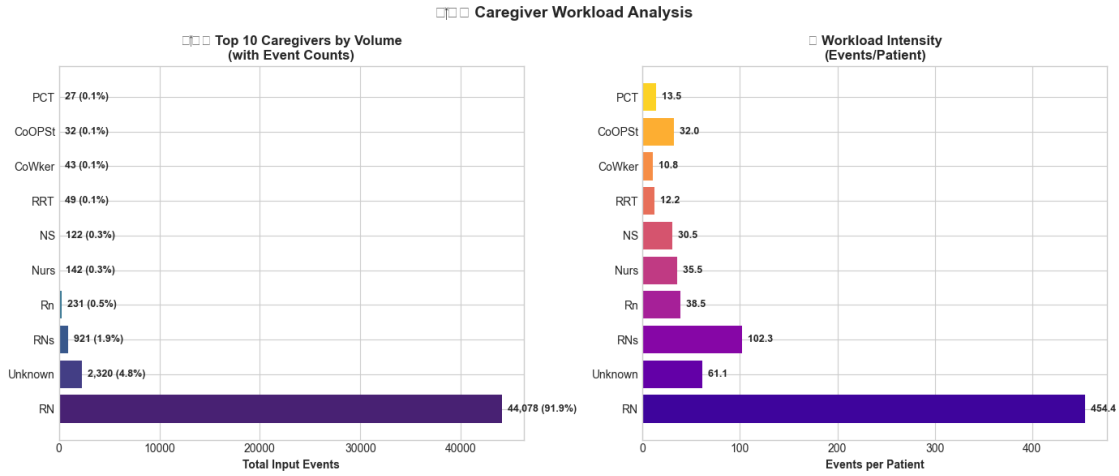
```
2026-01-01 14:29:03,456 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2026-01-01 14:29:03,458 INFO sqlalchemy.engine.Engine SELECT COUNT(*) FROM
gold.dim_caregiver
2026-01-01 14:29:03,458 INFO sqlalchemy.engine.Engine [generated in 0.00082s] {}
2026-01-01 14:29:03,472 INFO sqlalchemy.engine.Engine COMMIT
  Total Caregivers: 7,567
2026-01-01 14:29:03,476 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2026-01-01 14:29:03,477 INFO sqlalchemy.engine.Engine
        SELECT
            COALESCE(dc.label, 'Unknown') as role,
            COUNT(*) as events,
            COUNT(DISTINCT f.subject_id) as patients
        FROM gold.fact_input_event f
        LEFT JOIN gold.dim_caregiver dc ON f.caregiver_key = dc.caregiver_key
        GROUP BY dc.label
        ORDER BY events DESC
        LIMIT 10

2026-01-01 14:29:03,478 INFO sqlalchemy.engine.Engine [generated in 0.00114s] {}
```

```
2026-01-01 14:29:03,561 INFO sqlalchemy.engine.Engine COMMIT

  CAREGIVER WORKLOAD (Top 10)
   role  events  patients
     RN   44078        97
Unknown    2320        38
    RNs     921         9
     Rn     231         6
   Nurs     142         4
     NS     122         4
    RRT      49         4
 CoWker      43         4
 CoOPSt      32         1
    PCT      27         2
```

[16]:
```python
# Caregiver workload visualization WITH LABELS
if len(cg_workload) > 0:
    fig, axes = plt.subplots(1, 2, figsize=(14, 6))

    # Top 10 by events
    ax1 = axes[0]
    top10 = cg_workload.head(10)
    bars1 = ax1.barh(top10['role'], top10['events'],
                     color=sns.color_palette('viridis', len(top10)))
    ax1.set_xlabel('Total Input Events', fontweight='bold')
    ax1.set_title('  Top 10 Caregivers by Volume\n(with Event Counts)',␣
 ↪fontweight='bold')
    # Add labels
    total_events = top10['events'].sum()
    for bar, row in zip(bars1, top10.itertuples()):
        width = bar.get_width()
        pct = (width/total_events*100)
        ax1.text(width, bar.get_y() + bar.get_height()/2.,
                f'  {int(width):,} ({pct:.1f}%)',
                ha='left', va='center', fontweight='bold', fontsize=9)

    # Events per patient
    ax2 = axes[1]
    top10['events_per_patient'] = top10['events'] / top10['patients']
    bars2 = ax2.barh(top10['role'], top10['events_per_patient'],
                     color=sns.color_palette('plasma', len(top10)))
    ax2.set_xlabel('Events per Patient', fontweight='bold')
    ax2.set_title('  Workload Intensity\n(Events/Patient)', fontweight='bold')
    # Add labels
    for bar in bars2:
        width = bar.get_width()
        ax2.text(width, bar.get_y() + bar.get_height()/2.,
```

```
                f'  {width:.1f}',
                ha='left', va='center', fontweight='bold', fontsize=9)

    plt.suptitle('  Caregiver Workload Analysis', fontsize=14,␣
 ↪fontweight='bold')
    plt.tight_layout()
    plt.show()

    print("\n  All bars labeled with values and percentages")
else:
    print("  No caregiver workload data to visualize")
```

```
C:\Users\ayoub\AppData\Local\Temp\ipykernel_20120\765124215.py:23:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  top10['events_per_patient'] = top10['events'] / top10['patients']
C:\Users\ayoub\AppData\Local\Temp\ipykernel_20120\765124215.py:36: UserWarning:
Glyph 128104 (\N{MAN}) missing from font(s) Arial.
  plt.tight_layout()
C:\Users\ayoub\AppData\Local\Temp\ipykernel_20120\765124215.py:36: UserWarning:
Glyph 9877 (\N{STAFF OF AESCULAPIUS}) missing from font(s) Arial.
  plt.tight_layout()
C:\Users\ayoub\AppData\Local\Temp\ipykernel_20120\765124215.py:36: UserWarning:
Glyph 65039 (\N{VARIATION SELECTOR-16}) missing from font(s) Arial.
  plt.tight_layout()
C:\Users\ayoub\AppData\Local\Temp\ipykernel_20120\765124215.py:36: UserWarning:
Glyph 128202 (\N{BAR CHART}) missing from font(s) Arial.
  plt.tight_layout()
e:\vs-code\github\Medical-Data-Mining\data-warehouse\project\venv\Lib\site-
packages\IPython\core\pylabtools.py:170: UserWarning: Glyph 128104 (\N{MAN})
missing from font(s) Arial.
  fig.canvas.print_figure(bytes_io, **kw)
e:\vs-code\github\Medical-Data-Mining\data-warehouse\project\venv\Lib\site-
packages\IPython\core\pylabtools.py:170: UserWarning: Glyph 9877 (\N{STAFF OF
AESCULAPIUS}) missing from font(s) Arial.
  fig.canvas.print_figure(bytes_io, **kw)
e:\vs-code\github\Medical-Data-Mining\data-warehouse\project\venv\Lib\site-
packages\IPython\core\pylabtools.py:170: UserWarning: Glyph 65039 (\N{VARIATION
SELECTOR-16}) missing from font(s) Arial.
  fig.canvas.print_figure(bytes_io, **kw)
e:\vs-code\github\Medical-Data-Mining\data-warehouse\project\venv\Lib\site-
packages\IPython\core\pylabtools.py:170: UserWarning: Glyph 128202 (\N{BAR
CHART}) missing from font(s) Arial.
  fig.canvas.print_figure(bytes_io, **kw)
```




All bars labeled with values and percentages

---

## 1.6 5. Medication Analysis

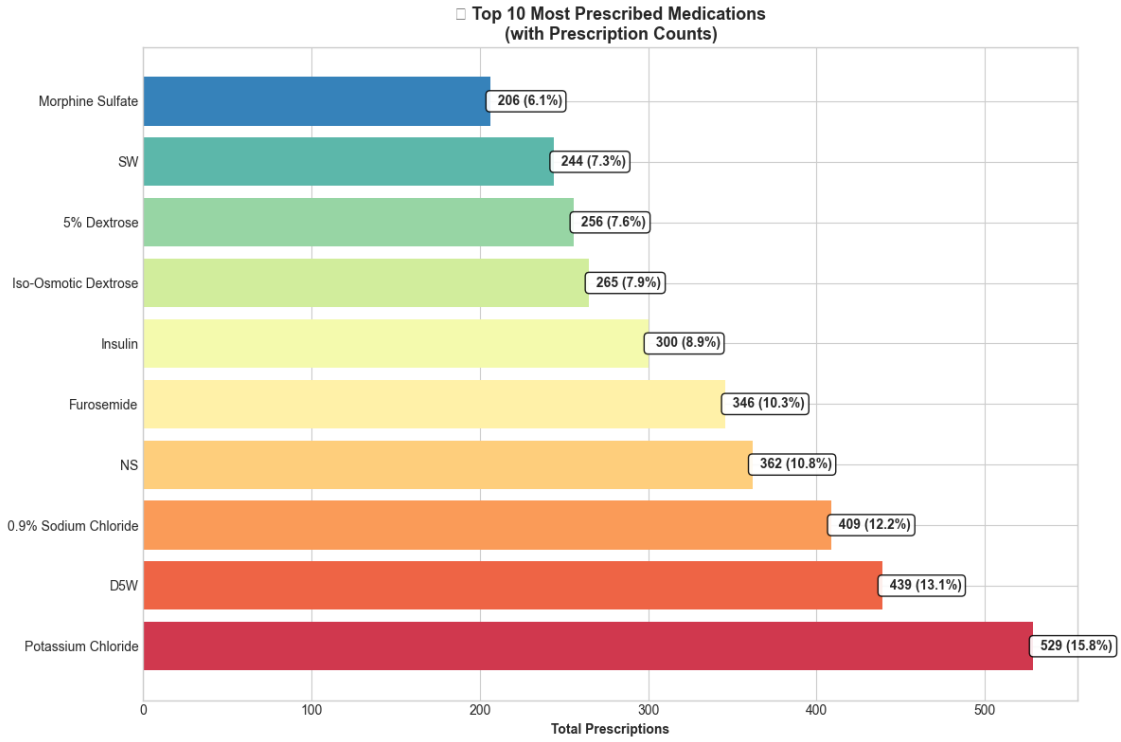
[17]:
```
# Top medications
meds = query_df("""
    SELECT
        drug,
        prescription_count,
        patient_count
    FROM gold.agg_medication_usage
    ORDER BY prescription_count DESC
    LIMIT 10
""")

print("  TOP 10 MEDICATIONS")
print(meds.to_string(index=False))
```

```
2026-01-01 14:30:02,306 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2026-01-01 14:30:02,307 INFO sqlalchemy.engine.Engine
    SELECT
        drug,
        prescription_count,
        patient_count
    FROM gold.agg_medication_usage
    ORDER BY prescription_count DESC
    LIMIT 10

2026-01-01 14:30:02,309 INFO sqlalchemy.engine.Engine [generated in 0.00193s] {}
```

```
2026-01-01 14:30:02,318 INFO sqlalchemy.engine.Engine COMMIT
 TOP 10 MEDICATIONS
                drug  prescription_count  patient_count
 Potassium Chloride                 529             66
                D5W                 439             68
0.9% Sodium Chloride                409             39
                 NS                 362             56
          Furosemide                346             52
             Insulin                300             67
Iso-Osmotic Dextrose                265             60
         5% Dextrose                256             32
                 SW                 244             51
    Morphine Sulfate                206             55
```

[18]:
```python
# Medication chart WITH LABELS
if len(meds) > 0:
    fig, ax = plt.subplots(figsize=(12, 8))
    bars = ax.barh(meds['drug'], meds['prescription_count'],
                   color=sns.color_palette('Spectral', len(meds)))
    ax.set_xlabel('Total Prescriptions', fontweight='bold')
    ax.set_title('  Top 10 Most Prescribed Medications\n(with Prescription␣
 ↪Counts)',
                 fontweight='bold', fontsize=13)

    # Add labels
    total_rx = meds['prescription_count'].sum()
    for bar, row in zip(bars, meds.itertuples()):
        width = bar.get_width()
        pct = (width/total_rx*100)
        ax.text(width, bar.get_y() + bar.get_height()/2.,
                f'  {int(width):,} ({pct:.1f}%)',
                ha='left', va='center', fontweight='bold',
                bbox=dict(boxstyle='round,pad=0.3', facecolor='white', alpha=0.
 ↪9))

    plt.tight_layout()
    plt.show()
    print("\n Labels show count and percentage of total prescriptions")
```

```
C:\Users\ayoub\AppData\Local\Temp\ipykernel_20120\34440865.py:20: UserWarning:
Glyph 128138 (\N{PILL}) missing from font(s) Arial.
  plt.tight_layout()
e:\vs-code\github\Medical-Data-Mining\data-warehouse\project\venv\Lib\site-
packages\IPython\core\pylabtools.py:170: UserWarning: Glyph 128138 (\N{PILL})
missing from font(s) Arial.
  fig.canvas.print_figure(bytes_io, **kw)
```




Labels show count and percentage of total prescriptions

---

## 1.7 6. Lab Test Analysis

[19]:
```
# Lab abnormality analysis
lab_analysis = query_df("""
    SELECT
        dl.label as test,
        COUNT(*) as total,
        SUM(CASE WHEN f.is_abnormal THEN 1 ELSE 0 END) as abnormal,
        ROUND(100.0 * SUM(CASE WHEN f.is_abnormal THEN 1 ELSE 0 END) /␣
 ↪COUNT(*), 1) as abnormal_pct
    FROM gold.fact_lab_event f
    JOIN gold.dim_labitem dl ON f.labitem_key = dl.labitem_key
    GROUP BY dl.label
    HAVING COUNT(*) > 100
    ORDER BY abnormal_pct DESC
    LIMIT 10
""")

print("  TOP 10 LAB TESTS BY ABNORMALITY RATE")
```

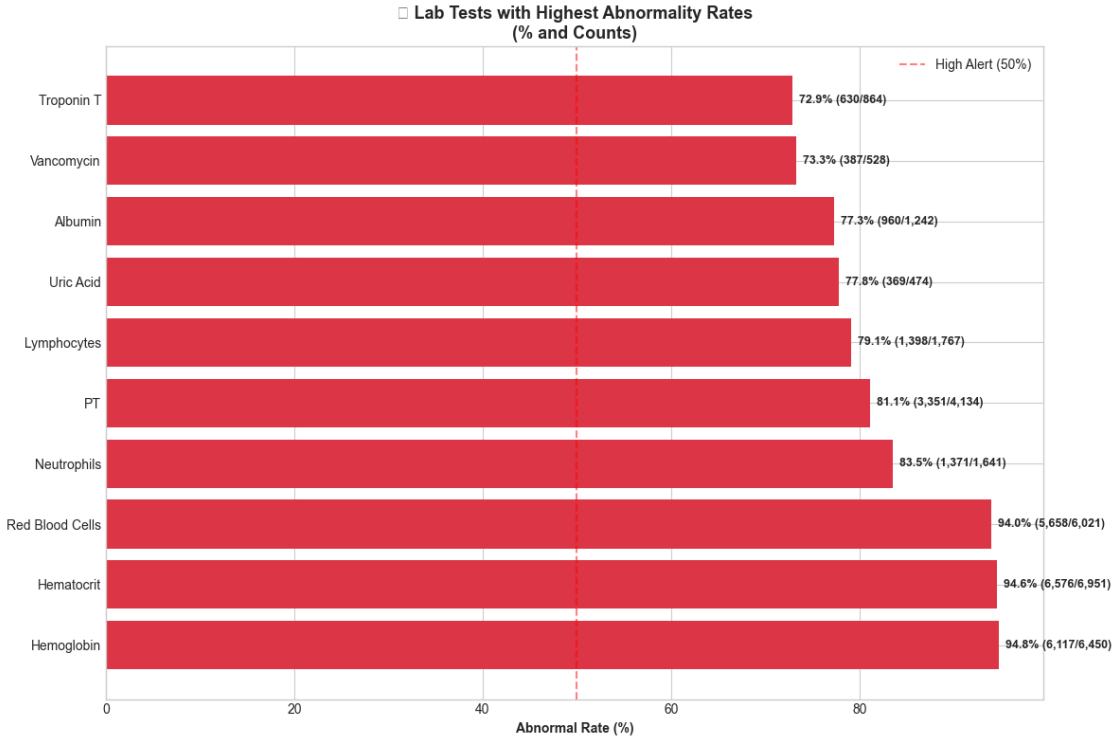
```
print(lab_analysis.to_string(index=False))
```

```
2026-01-01 14:30:19,950 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2026-01-01 14:30:19,952 INFO sqlalchemy.engine.Engine
    SELECT
        dl.label as test,
        COUNT(*) as total,
        SUM(CASE WHEN f.is_abnormal THEN 1 ELSE 0 END) as abnormal,
        ROUND(100.0 * SUM(CASE WHEN f.is_abnormal THEN 1 ELSE 0 END) / COUNT(*),
1) as abnormal_pct
    FROM gold.fact_lab_event f
    JOIN gold.dim_labitem dl ON f.labitem_key = dl.labitem_key
    GROUP BY dl.label
    HAVING COUNT(*) > 100
    ORDER BY abnormal_pct DESC
    LIMIT 10

2026-01-01 14:30:19,953 INFO sqlalchemy.engine.Engine [generated in 0.00107s] {}
2026-01-01 14:30:20,479 INFO sqlalchemy.engine.Engine COMMIT
 TOP 10 LAB TESTS BY ABNORMALITY RATE
          test  total  abnormal abnormal_pct
     Hemoglobin   6450      6117         94.8
     Hematocrit   6951      6576         94.6
Red Blood Cells   6021      5658         94.0
    Neutrophils   1641      1371         83.5
             PT   4134      3351         81.1
    Lymphocytes   1767      1398         79.1
      Uric Acid    474       369         77.8
        Albumin   1242       960         77.3
     Vancomycin    528       387         73.3
     Troponin T    864       630         72.9
```

[20]:
```
# Lab abnormality visualization WITH LABELS
if len(lab_analysis) > 0:
    fig, ax = plt.subplots(figsize=(12, 8))

    # Color code by severity
    colors = ['#DC3545' if x > 50 else '#FFC107' if x > 25 else '#28A745'
              for x in lab_analysis['abnormal_pct']]
    bars = ax.barh(lab_analysis['test'], lab_analysis['abnormal_pct'],␣
 ↪color=colors)
    ax.set_xlabel('Abnormal Rate (%)', fontweight='bold')
    ax.set_title('  Lab Tests with Highest Abnormality Rates\n(% and Counts)',
                 fontweight='bold', fontsize=13)
    ax.axvline(x=50, color='red', linestyle='--', alpha=0.5, label='High Alert␣
 ↪(50%)')
    ax.legend()
```

```
    # Add labels
    for bar, row in zip(bars, lab_analysis.itertuples()):
        width = bar.get_width()
        ax.text(width, bar.get_y() + bar.get_height()/2.,
                f'  {width:.1f}% ({row.abnormal:,}/{row.total:,})',
                ha='left', va='center', fontweight='bold', fontsize=9)

    plt.tight_layout()
    plt.show()
    print("\n Labels show percentage and fraction (abnormal/total)")
    print(" Red bars: >50% abnormal rate (high alert)")
    print(" Yellow bars: 25-50% abnormal rate")
    print(" Green bars: <25% abnormal rate")
```

```
C:\Users\ayoub\AppData\Local\Temp\ipykernel_20120\2611925556.py:22: UserWarning:
Glyph 129514 (\N{TEST TUBE}) missing from font(s) Arial.
  plt.tight_layout()
e:\vs-code\github\Medical-Data-Mining\data-warehouse\project\venv\Lib\site-
packages\IPython\core\pylabtools.py:170: UserWarning: Glyph 129514 (\N{TEST
TUBE}) missing from font(s) Arial.
  fig.canvas.print_figure(bytes_io, **kw)
```



```
Labels show percentage and fraction (abnormal/total)
Red bars: >50% abnormal rate (high alert)
Yellow bars: 25-50% abnormal rate
Green bars: <25% abnormal rate
```

---

### 1.8 7. Infection Analysis

[21]:
```
# Infection statistics
infections = query_df("""
    SELECT
        organism_name,
        total_cultures,
        resistance_tests,
        resistant_count,
        ROUND(100.0 * resistant_count / NULLIF(resistance_tests, 0), 1) as␣
 ↪resistance_pct
    FROM gold.agg_infection_stats
    WHERE organism_name IS NOT NULL
    ORDER BY total_cultures DESC
    LIMIT 10
""")

print("  TOP ORGANISMS BY CULTURE COUNT")
print(infections.to_string(index=False))
```

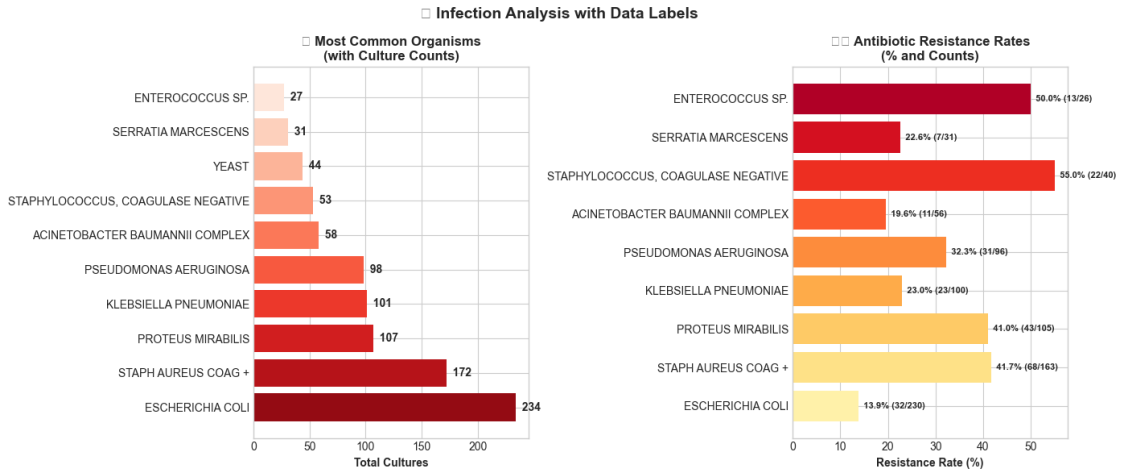
```
2026-01-01 14:31:05,707 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2026-01-01 14:31:05,708 INFO sqlalchemy.engine.Engine
    SELECT
        organism_name,
        total_cultures,
        resistance_tests,
        resistant_count,
        ROUND(100.0 * resistant_count / NULLIF(resistance_tests, 0), 1) as
resistance_pct
    FROM gold.agg_infection_stats
    WHERE organism_name IS NOT NULL
    ORDER BY total_cultures DESC
    LIMIT 10

2026-01-01 14:31:05,709 INFO sqlalchemy.engine.Engine [generated in 0.00130s] {}
2026-01-01 14:31:05,715 INFO sqlalchemy.engine.Engine COMMIT
  TOP ORGANISMS BY CULTURE COUNT
                    organism_name  total_cultures  resistance_tests
resistant_count resistance_pct
                 ESCHERICHIA COLI             234               230
32           13.9
```

```
                STAPH AUREUS COAG +             172               163
68           41.7
                    PROTEUS MIRABILIS             107               105
43           41.0
                KLEBSIELLA PNEUMONIAE             101               100
23           23.0
               PSEUDOMONAS AERUGINOSA              98                96
31           32.3
   ACINETOBACTER BAUMANNII COMPLEX              58                56
11           19.6
STAPHYLOCOCCUS, COAGULASE NEGATIVE              53                40
22           55.0
                              YEAST              44                 0
0            None
                   SERRATIA MARCESCENS              31                31
7            22.6
                     ENTEROCOCCUS SP.              27                26
13           50.0
```

[22]:
```
# Infection visualizations WITH LABELS
if len(infections) > 0:
    fig, axes = plt.subplots(1, 2, figsize=(14, 6))

    # Culture counts
    ax1 = axes[0]
    bars1 = ax1.barh(infections['organism_name'], infections['total_cultures'],
                     color=sns.color_palette('Reds_r', len(infections)))
    ax1.set_xlabel('Total Cultures', fontweight='bold')
    ax1.set_title('  Most Common Organisms\n(with Culture Counts)',␣
 ↪fontweight='bold')
    # Add labels
    for bar in bars1:
        width = bar.get_width()
        ax1.text(width, bar.get_y() + bar.get_height()/2.,
                f'  {int(width):,}',
                ha='left', va='center', fontweight='bold')

    # Resistance rates
    ax2 = axes[1]
    resistance_df = infections[infections['resistance_pct'].notna()]
    if len(resistance_df) > 0:
        bars2 = ax2.barh(resistance_df['organism_name'],␣
 ↪resistance_df['resistance_pct'],
                         color=sns.color_palette('YlOrRd', len(resistance_df)))
        ax2.set_xlabel('Resistance Rate (%)', fontweight='bold')
        ax2.set_title('  Antibiotic Resistance Rates\n(% and Counts)',␣
 ↪fontweight='bold')
```

```
        # Add labels
        for bar, row in zip(bars2, resistance_df.itertuples()):
            width = bar.get_width()
            ax2.text(width, bar.get_y() + bar.get_height()/2.,
                    f'  {width:.1f}% ({row.resistant_count}/{row.
↪resistance_tests})',
                    ha='left', va='center', fontweight='bold', fontsize=8)

    plt.suptitle('  Infection Analysis with Data Labels', fontsize=14,␣
↪fontweight='bold')
    plt.tight_layout()
    plt.show()
    print("\n  All values labeled on bars")
```

```
C:\Users\ayoub\AppData\Local\Temp\ipykernel_20120\2897347958.py:34: UserWarning:
Glyph 129440 (\N{MICROBE}) missing from font(s) Arial.
  plt.tight_layout()
C:\Users\ayoub\AppData\Local\Temp\ipykernel_20120\2897347958.py:34: UserWarning:
Glyph 9888 (\N{WARNING SIGN}) missing from font(s) Arial.
  plt.tight_layout()
C:\Users\ayoub\AppData\Local\Temp\ipykernel_20120\2897347958.py:34: UserWarning:
Glyph 65039 (\N{VARIATION SELECTOR-16}) missing from font(s) Arial.
  plt.tight_layout()
e:\vs-code\github\Medical-Data-Mining\data-warehouse\project\venv\Lib\site-
packages\IPython\core\pylabtools.py:170: UserWarning: Glyph 129440 (\N{MICROBE})
missing from font(s) Arial.
  fig.canvas.print_figure(bytes_io, **kw)
e:\vs-code\github\Medical-Data-Mining\data-warehouse\project\venv\Lib\site-
packages\IPython\core\pylabtools.py:170: UserWarning: Glyph 9888 (\N{WARNING
SIGN}) missing from font(s) Arial.
  fig.canvas.print_figure(bytes_io, **kw)
e:\vs-code\github\Medical-Data-Mining\data-warehouse\project\venv\Lib\site-
packages\IPython\core\pylabtools.py:170: UserWarning: Glyph 65039 (\N{VARIATION
SELECTOR-16}) missing from font(s) Arial.
  fig.canvas.print_figure(bytes_io, **kw)
```




```
All values labeled on bars
```

---

## 1.9 8. Actionable Recommendations

### 1.9.1 Data-Driven Decisions

[23]:
```
print("\n" + "="*70)
print("  ACTIONABLE BUSINESS RECOMMENDATIONS")
print("="*70)

recommendations = [
    ("Patient Care", "Identify high-utilization patients for care␣
 ↪coordination"),
    ("ICU Operations", "Optimize capacity for busiest units"),
    ("Staffing", "Balance caregiver workload based on event volumes"),
    ("Medication", "Monitor top prescriptions for adherence"),
    ("Lab Monitoring", "Use high abnormality tests as early warning␣
 ↪indicators"),
    ("Infection Control", "Prioritize antibiotic stewardship programs"),
]

for area, rec in recommendations:
    print(f"\n    {area}:")
    print(f"       {rec}")

print("\n" + "="*70)
print("  ANALYSIS COMPLETE - All charts include value labels")
print("="*70)
```

```
======================================================================
  ACTIONABLE BUSINESS RECOMMENDATIONS
======================================================================

    Patient Care:
       Identify high-utilization patients for care coordination

    ICU Operations:
       Optimize capacity for busiest units

    Staffing:
       Balance caregiver workload based on event volumes

    Medication:
       Monitor top prescriptions for adherence

    Lab Monitoring:
       Use high abnormality tests as early warning indicators

    Infection Control:
       Prioritize antibiotic stewardship programs

======================================================================
  ANALYSIS COMPLETE - All charts include value labels
======================================================================
```