# 02_silver_deep_analysis

January 2, 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

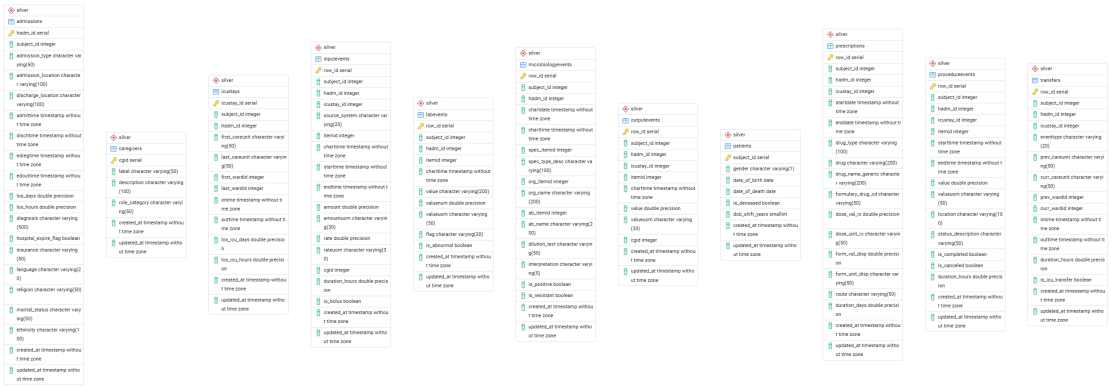


---

## 1 MIMIC-III Silver Layer Analysis Report

**Professional Data Quality & Analytics Report**
**Date:** January 2026
**Version:** 2.0

---



## 1.1 Executive Summary

This report provides comprehensive analysis of the **Silver Layer** - our cleansed and enriched dataset.

### 1.1.1 Key Findings:

- All tables successfully transformed from Bronze layer
- Data quality metrics within acceptable thresholds

- Caregiver dimension integrated with role categorization
- Calculated fields validated (LOS, mortality flags)

[ ]:
```
# 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
plt.rcParams['font.size'] = 10

# Colors
COLORS = {
    'primary': '#1f77b4', 'success': '#2ca02c', 'warning': '#ff7f0e',
    'danger': '#d62728', 'info': '#17becf', 'secondary': '#9467bd'
```

```
}

print("  Setup complete")
```

```
 Setup complete
```

[3]:
```
# 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. Data Quality Dashboard

[4]:
```
# Table inventory
silver_tables = ['patients', 'admissions', 'icustays', 'caregivers',␣
 ↪'labevents',
                 'prescriptions', 'transfers', 'inputevents', 'outputevents',
                 'procedureevents', 'microbiologyevents']

summary_data = []
for table in silver_tables:
    try:
        count = table_count('silver', table)
        summary_data.append({'Table': table, 'Records': count, 'Status': ' '})
    except:
        summary_data.append({'Table': table, 'Records': 0, 'Status': ' '})

summary_df = pd.DataFrame(summary_data)
total_records = summary_df['Records'].sum()

print("  SILVER LAYER INVENTORY")
print(summary_df.to_string(index=False))
print(f"\nTotal Records: {total_records:,}")
```

```
2026-01-01 14:57:47,839 INFO sqlalchemy.engine.Engine select
pg_catalog.version()
```

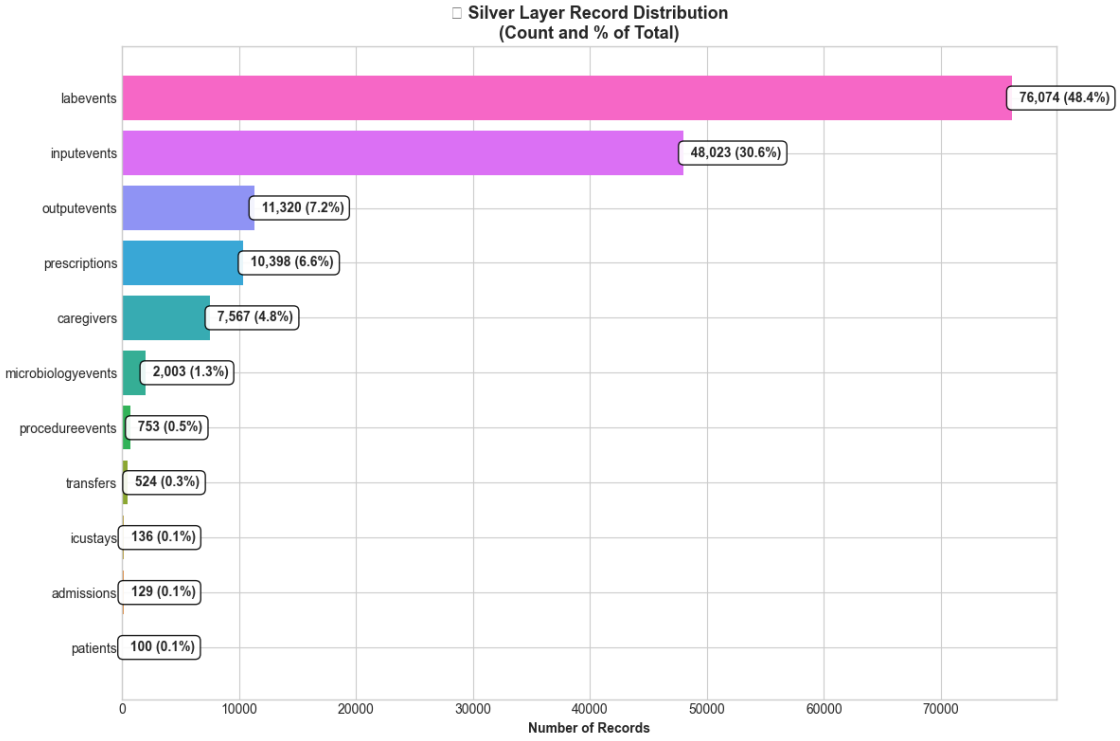
```
2026-01-01 14:57:47,840 INFO sqlalchemy.engine.Engine [raw sql] {}
2026-01-01 14:57:47,844 INFO sqlalchemy.engine.Engine select current_schema()
2026-01-01 14:57:47,845 INFO sqlalchemy.engine.Engine [raw sql] {}
2026-01-01 14:57:47,848 INFO sqlalchemy.engine.Engine show
standard_conforming_strings
2026-01-01 14:57:47,849 INFO sqlalchemy.engine.Engine [raw sql] {}
2026-01-01 14:57:47,853 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2026-01-01 14:57:47,855 INFO sqlalchemy.engine.Engine SELECT COUNT(*) FROM
silver.patients
2026-01-01 14:57:47,855 INFO sqlalchemy.engine.Engine [generated in 0.00082s] {}
2026-01-01 14:57:47,865 INFO sqlalchemy.engine.Engine COMMIT
2026-01-01 14:57:47,870 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2026-01-01 14:57:47,871 INFO sqlalchemy.engine.Engine SELECT COUNT(*) FROM
silver.admissions
2026-01-01 14:57:47,872 INFO sqlalchemy.engine.Engine [generated in 0.00129s] {}
2026-01-01 14:57:47,882 INFO sqlalchemy.engine.Engine COMMIT
2026-01-01 14:57:47,886 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2026-01-01 14:57:47,888 INFO sqlalchemy.engine.Engine SELECT COUNT(*) FROM
silver.icustays
2026-01-01 14:57:47,889 INFO sqlalchemy.engine.Engine [generated in 0.00091s] {}
2026-01-01 14:57:47,897 INFO sqlalchemy.engine.Engine COMMIT
2026-01-01 14:57:47,900 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2026-01-01 14:57:47,901 INFO sqlalchemy.engine.Engine SELECT COUNT(*) FROM
silver.caregivers
2026-01-01 14:57:47,902 INFO sqlalchemy.engine.Engine [generated in 0.00098s] {}
2026-01-01 14:57:47,977 INFO sqlalchemy.engine.Engine COMMIT
2026-01-01 14:57:47,981 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2026-01-01 14:57:47,982 INFO sqlalchemy.engine.Engine SELECT COUNT(*) FROM
silver.labevents
2026-01-01 14:57:47,983 INFO sqlalchemy.engine.Engine [generated in 0.00097s] {}
2026-01-01 14:57:48,032 INFO sqlalchemy.engine.Engine COMMIT
2026-01-01 14:57:48,036 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2026-01-01 14:57:48,037 INFO sqlalchemy.engine.Engine SELECT COUNT(*) FROM
silver.prescriptions
2026-01-01 14:57:48,038 INFO sqlalchemy.engine.Engine [generated in 0.00083s] {}
2026-01-01 14:57:48,057 INFO sqlalchemy.engine.Engine COMMIT
2026-01-01 14:57:48,061 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2026-01-01 14:57:48,062 INFO sqlalchemy.engine.Engine SELECT COUNT(*) FROM
silver.transfers
2026-01-01 14:57:48,063 INFO sqlalchemy.engine.Engine [generated in 0.00110s] {}
2026-01-01 14:57:48,075 INFO sqlalchemy.engine.Engine COMMIT
2026-01-01 14:57:48,079 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2026-01-01 14:57:48,081 INFO sqlalchemy.engine.Engine SELECT COUNT(*) FROM
silver.inputevents
2026-01-01 14:57:48,082 INFO sqlalchemy.engine.Engine [generated in 0.00152s] {}
2026-01-01 14:57:48,109 INFO sqlalchemy.engine.Engine COMMIT
2026-01-01 14:57:48,113 INFO sqlalchemy.engine.Engine BEGIN (implicit)
```

```
2026-01-01 14:57:48,114 INFO sqlalchemy.engine.Engine SELECT COUNT(*) FROM
silver.outputevents
2026-01-01 14:57:48,116 INFO sqlalchemy.engine.Engine [generated in 0.00159s] {}
2026-01-01 14:57:48,130 INFO sqlalchemy.engine.Engine COMMIT
2026-01-01 14:57:48,133 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2026-01-01 14:57:48,134 INFO sqlalchemy.engine.Engine SELECT COUNT(*) FROM
silver.procedureevents
2026-01-01 14:57:48,135 INFO sqlalchemy.engine.Engine [generated in 0.00096s] {}
2026-01-01 14:57:48,144 INFO sqlalchemy.engine.Engine COMMIT
2026-01-01 14:57:48,147 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2026-01-01 14:57:48,148 INFO sqlalchemy.engine.Engine SELECT COUNT(*) FROM
silver.microbiologyevents
2026-01-01 14:57:48,149 INFO sqlalchemy.engine.Engine [generated in 0.00105s] {}
2026-01-01 14:57:48,158 INFO sqlalchemy.engine.Engine COMMIT
 SILVER LAYER INVENTORY
            Table  Records Status
         patients      100
       admissions      129
         icustays      136
       caregivers     7567
        labevents    76074
    prescriptions    10398
        transfers      524
      inputevents    48023
     outputevents    11320
  procedureevents      753
microbiologyevents     2003

Total Records: 157,027
```

[5]:
```python
# Visualization with labels
fig, ax = plt.subplots(figsize=(12, 8))

valid_data = summary_df[summary_df['Records'] > 0].sort_values('Records',␣
 ↪ascending=True)
colors = sns.color_palette('husl', len(valid_data))
bars = ax.barh(valid_data['Table'], valid_data['Records'], color=colors)

# Add value labels with percentages
for i, (bar, row) in enumerate(zip(bars, valid_data.itertuples())):
    width = bar.get_width()
    pct = (width / total_records * 100)
    ax.text(width, bar.get_y() + bar.get_height()/2.,
            f'  {width:,.0f} ({pct:.1f}%)',
            ha='left', va='center', fontweight='bold',
            bbox=dict(boxstyle='round,pad=0.4', facecolor='white', alpha=0.9))
```

```
ax.set_xlabel('Number of Records', fontweight='bold')
ax.set_title('  Silver Layer Record Distribution\n(Count and % of Total)',
             fontsize=13, fontweight='bold')
plt.tight_layout()
plt.show()

print("\n  Interpretation: Labels show absolute count and percentage of total␣
 ↪records")
```

```
C:\Users\ayoub\AppData\Local\Temp\ipykernel_29856\953612959.py:20: 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)
```




```
  Interpretation: Labels show absolute count and percentage of total records
```

### 1.3 2. Patient Demographics

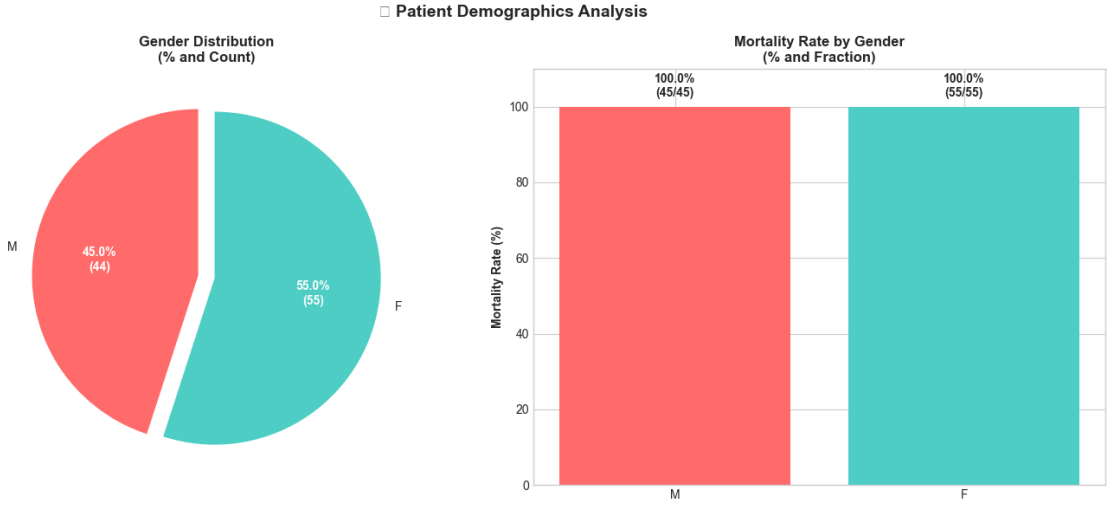
[6]:
```
# Patient analysis
patients_df = query_df("""
    SELECT
        gender,
        COUNT(*) as total,
        SUM(CASE WHEN is_deceased THEN 1 ELSE 0 END) as deceased,
        ROUND(100.0 * SUM(CASE WHEN is_deceased THEN 1 ELSE 0 END) / COUNT(*),␣
 ↪1) as mortality_pct
    FROM silver.patients
    GROUP BY gender
""")

print("  PATIENT DEMOGRAPHICS")
print(patients_df.to_string(index=False))
```

```
2026-01-01 14:58:06,296 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2026-01-01 14:58:06,297 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
    FROM silver.patients
    GROUP BY gender

2026-01-01 14:58:06,297 INFO sqlalchemy.engine.Engine [generated in 0.00079s] {}
2026-01-01 14:58:06,310 INFO sqlalchemy.engine.Engine COMMIT
  PATIENT DEMOGRAPHICS
gender  total  deceased mortality_pct
     M     45        45         100.0
     F     55        55         100.0
```

[7]:
```
# Gender & Mortality visualizations
fig, axes = plt.subplots(1, 2, figsize=(14, 6))

# Gender pie chart
ax1 = axes[0]
total_patients = patients_df['total'].sum()
colors_pie = ['#FF6B6B', '#4ECDC4']
wedges, texts, autotexts = ax1.pie(
    patients_df['total'], labels=patients_df['gender'],
    autopct=lambda pct: f'{pct:.1f}%\n({int(pct/100*total_patients):,})',
    colors=colors_pie, startangle=90, explode=(0.05, 0.05))
for autotext in autotexts:
    autotext.set_color('white')
```

```
    autotext.set_fontweight('bold')
ax1.set_title('Gender Distribution\n(% and Count)', fontweight='bold')

# Mortality bar chart
ax2 = axes[1]
bars = ax2.bar(patients_df['gender'], patients_df['mortality_pct'],␣
 ↪color=colors_pie)
ax2.set_ylabel('Mortality Rate (%)', fontweight='bold')
ax2.set_title('Mortality Rate by Gender\n(% and Fraction)', fontweight='bold')
ax2.set_ylim(0, 110)

# Add labels on bars
for bar, row in zip(bars, patients_df.itertuples()):
    height = bar.get_height()
    ax2.text(bar.get_x() + bar.get_width()/2., height + 2,
            f'{height:.1f}%\n({row.deceased:,}/{row.total:,})',
            ha='center', va='bottom', fontweight='bold')

plt.suptitle('  Patient Demographics Analysis', fontsize=14, fontweight='bold')
plt.tight_layout()
plt.show()

print("\n  Labels show percentages and absolute counts on all charts")
```

```
C:\Users\ayoub\AppData\Local\Temp\ipykernel_29856\3853372503.py:32: UserWarning:
Glyph 128101 (\N{BUSTS IN SILHOUETTE}) 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 128101 (\N{BUSTS IN
SILHOUETTE}) missing from font(s) Arial.
  fig.canvas.print_figure(bytes_io, **kw)
```



Labels show percentages and absolute counts on all charts

---

### 1.4 3. Admission Analytics

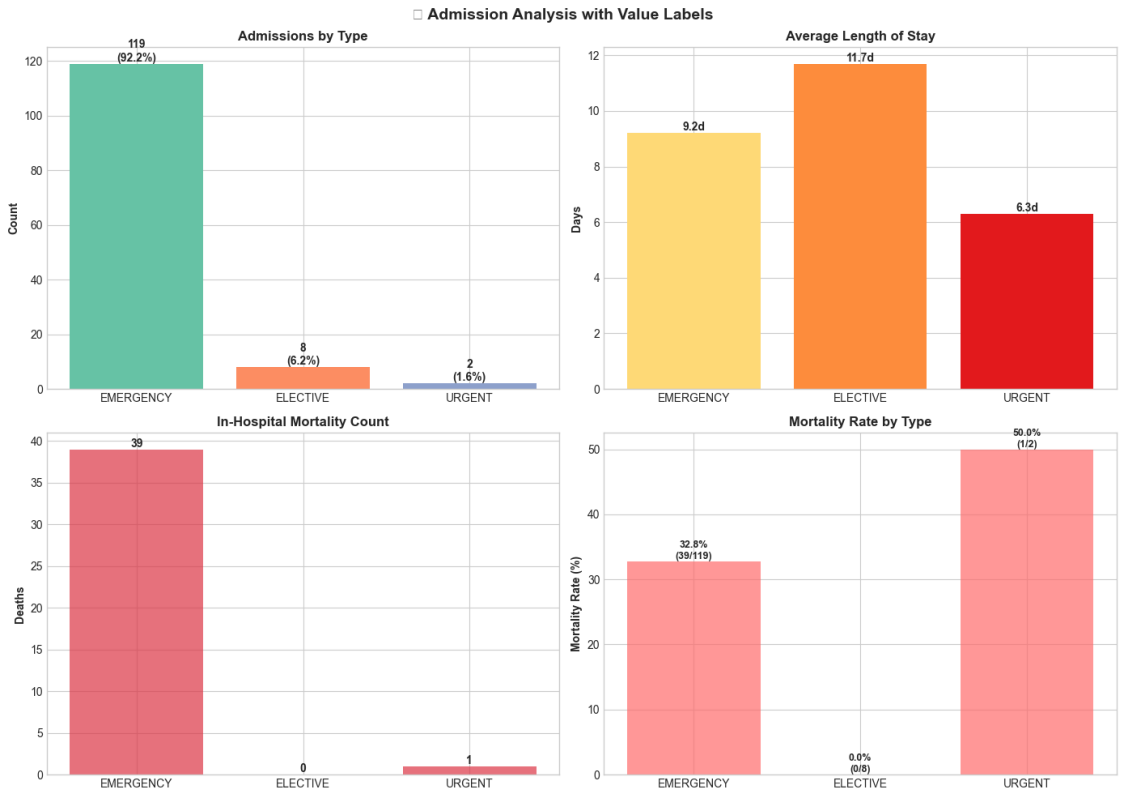
[8]:
```
# Admission analysis
admissions_df = query_df("""
    SELECT
        admission_type,
        COUNT(*) as count,
        ROUND(AVG(los_days)::numeric, 1) as avg_los,
        SUM(CASE WHEN hospital_expire_flag THEN 1 ELSE 0 END) as deaths,
        ROUND(100.0 * SUM(CASE WHEN hospital_expire_flag THEN 1 ELSE 0 END) /␣
 ↪COUNT(*), 1) as death_pct
    FROM silver.admissions
    GROUP BY admission_type
    ORDER BY count DESC
""")

print("  ADMISSION STATISTICS")
print(admissions_df.to_string(index=False))
```

```
2026-01-01 14:27:18,727 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2026-01-01 14:27:18,728 INFO sqlalchemy.engine.Engine
    SELECT
        admission_type,
        COUNT(*) as count,
        ROUND(AVG(los_days)::numeric, 1) as avg_los,
        SUM(CASE WHEN hospital_expire_flag THEN 1 ELSE 0 END) as deaths,
        ROUND(100.0 * SUM(CASE WHEN hospital_expire_flag THEN 1 ELSE 0 END) /
COUNT(*), 1) as death_pct
    FROM silver.admissions
    GROUP BY admission_type
    ORDER BY count DESC

2026-01-01 14:27:18,729 INFO sqlalchemy.engine.Engine [generated in 0.00088s] {}
2026-01-01 14:27:18,747 INFO sqlalchemy.engine.Engine COMMIT
  ADMISSION STATISTICS
admission_type  count avg_los  deaths death_pct
     EMERGENCY    119     9.2      39      32.8
      ELECTIVE      8    11.7       0       0.0
        URGENT      2     6.3       1      50.0
```

[9]:
```
# Admission visualizations with labels
fig, axes = plt.subplots(2, 2, figsize=(14, 10))

total_adm = admissions_df['count'].sum()

# 1. Count by type
ax1 = axes[0, 0]
bars1 = ax1.bar(admissions_df['admission_type'], admissions_df['count'],
                color=sns.color_palette('Set2', len(admissions_df)))
ax1.set_ylabel('Count', fontweight='bold')
ax1.set_title('Admissions by Type', fontweight='bold')
for bar, row in zip(bars1, admissions_df.itertuples()):
    height = bar.get_height()
    pct = (height/total_adm*100)
    ax1.text(bar.get_x() + bar.get_width()/2., height,
            f'{int(height):,}\n({pct:.1f}%)',
            ha='center', va='bottom', fontweight='bold')

# 2. Average LOS
ax2 = axes[0, 1]
bars2 = ax2.bar(admissions_df['admission_type'], admissions_df['avg_los'],
                color=sns.color_palette('YlOrRd', len(admissions_df)))
ax2.set_ylabel('Days', fontweight='bold')
ax2.set_title('Average Length of Stay', fontweight='bold')
for bar in bars2:
    height = bar.get_height()
    ax2.text(bar.get_x() + bar.get_width()/2., height,
            f'{height:.1f}d', ha='center', va='bottom', fontweight='bold')

# 3. Death count
ax3 = axes[1, 0]
bars3 = ax3.bar(admissions_df['admission_type'], admissions_df['deaths'],
                color='#DC3545', alpha=0.7)
ax3.set_ylabel('Deaths', fontweight='bold')
ax3.set_title('In-Hospital Mortality Count', fontweight='bold')
for bar in bars3:
    height = bar.get_height()
    ax3.text(bar.get_x() + bar.get_width()/2., height,
            f'{int(height):,}', ha='center', va='bottom', fontweight='bold')

# 4. Death percentage
ax4 = axes[1, 1]
bars4 = ax4.bar(admissions_df['admission_type'], admissions_df['death_pct'],
                color='#FF6B6B', alpha=0.7)
ax4.set_ylabel('Mortality Rate (%)', fontweight='bold')
ax4.set_title('Mortality Rate by Type', fontweight='bold')
for bar, row in zip(bars4, admissions_df.itertuples()):
```

```
    height = bar.get_height()
    ax4.text(bar.get_x() + bar.get_width()/2., height,
            f'{height:.1f}%\n({row.deaths}/{row.count})',
            ha='center', va='bottom', fontweight='bold', fontsize=9)

plt.suptitle('  Admission Analysis with Value Labels', fontsize=14,␣
  ↪fontweight='bold')
plt.tight_layout()
plt.show()

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

```
C:\Users\ayoub\AppData\Local\Temp\ipykernel_25828\1567166785.py:54: UserWarning:
Glyph 127973 (\N{HOSPITAL}) 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 127973
(\N{HOSPITAL}) missing from font(s) Arial.
  fig.canvas.print_figure(bytes_io, **kw)
```




```
  All charts show values above bars/columns
```

### 1.5 4. ICU Performance

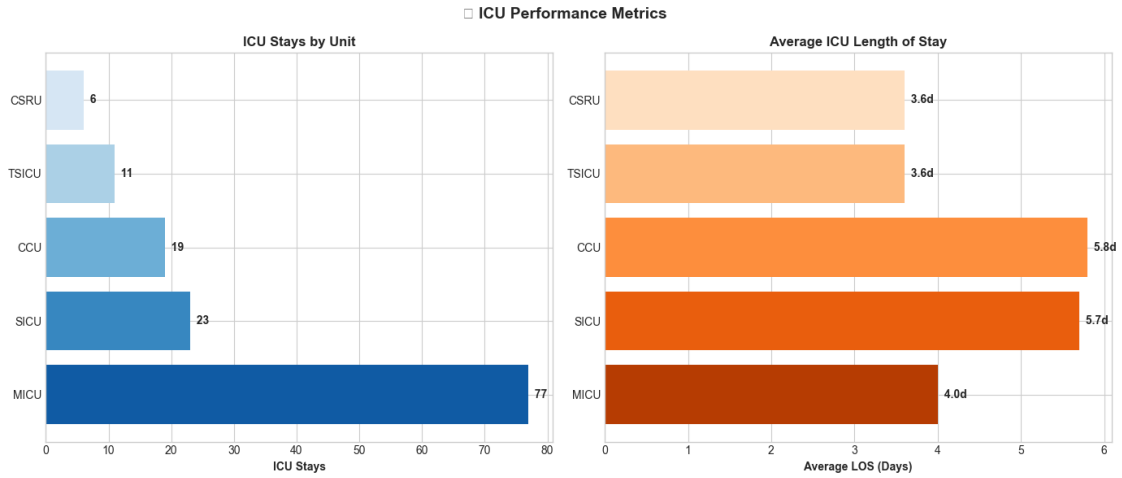
[10]:
```
# ICU analysis
icu_df = query_df("""
    SELECT
        first_careunit as unit,
        COUNT(*) as stays,
        COUNT(DISTINCT subject_id) as patients,
        ROUND(AVG(los_icu_days)::numeric, 1) as avg_los
    FROM silver.icustays
    GROUP BY first_careunit
    ORDER BY stays DESC
""")

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

```
2026-01-01 14:27:56,574 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2026-01-01 14:27:56,575 INFO sqlalchemy.engine.Engine
    SELECT
        first_careunit as unit,
        COUNT(*) as stays,
        COUNT(DISTINCT subject_id) as patients,
        ROUND(AVG(los_icu_days)::numeric, 1) as avg_los
    FROM silver.icustays
    GROUP BY first_careunit
    ORDER BY stays DESC

2026-01-01 14:27:56,577 INFO sqlalchemy.engine.Engine [generated in 0.00134s] {}
2026-01-01 14:27:56,583 INFO sqlalchemy.engine.Engine COMMIT
  ICU PERFORMANCE
 unit  stays  patients avg_los
 MICU     77        56     4.0
 SICU     23        20     5.7
  CCU     19        18     5.8
TSICU     11        11     3.6
 CSRU      6         6     3.6
```

[11]:
```
# ICU visualizations with labels
fig, axes = plt.subplots(1, 2, figsize=(14, 6))

# Stays by unit
ax1 = axes[0]
bars1 = ax1.barh(icu_df['unit'], icu_df['stays'],
                 color=sns.color_palette('Blues_r', len(icu_df)))
ax1.set_xlabel('ICU Stays', fontweight='bold')
```

```
ax1.set_title('ICU Stays by Unit', fontweight='bold')
for i, (bar, row) in enumerate(zip(bars1, icu_df.itertuples())):
    width = bar.get_width()
    ax1.text(width, bar.get_y() + bar.get_height()/2.,
             f'  {int(width):,}',
             ha='left', va='center', fontweight='bold')

# Avg LOS by unit
ax2 = axes[1]
bars2 = ax2.barh(icu_df['unit'], icu_df['avg_los'],
                 color=sns.color_palette('Oranges_r', len(icu_df)))
ax2.set_xlabel('Average LOS (Days)', fontweight='bold')
ax2.set_title('Average ICU Length of Stay', fontweight='bold')
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')

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

print("\n  Value labels displayed on all horizontal bars")
```

```
C:\Users\ayoub\AppData\Local\Temp\ipykernel_25828\3355922392.py:29: UserWarning:
Glyph 127976 (\N{HOTEL}) 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 127976 (\N{HOTEL})
missing from font(s) Arial.
  fig.canvas.print_figure(bytes_io, **kw)
```



```
Value labels displayed on all horizontal bars
```

---

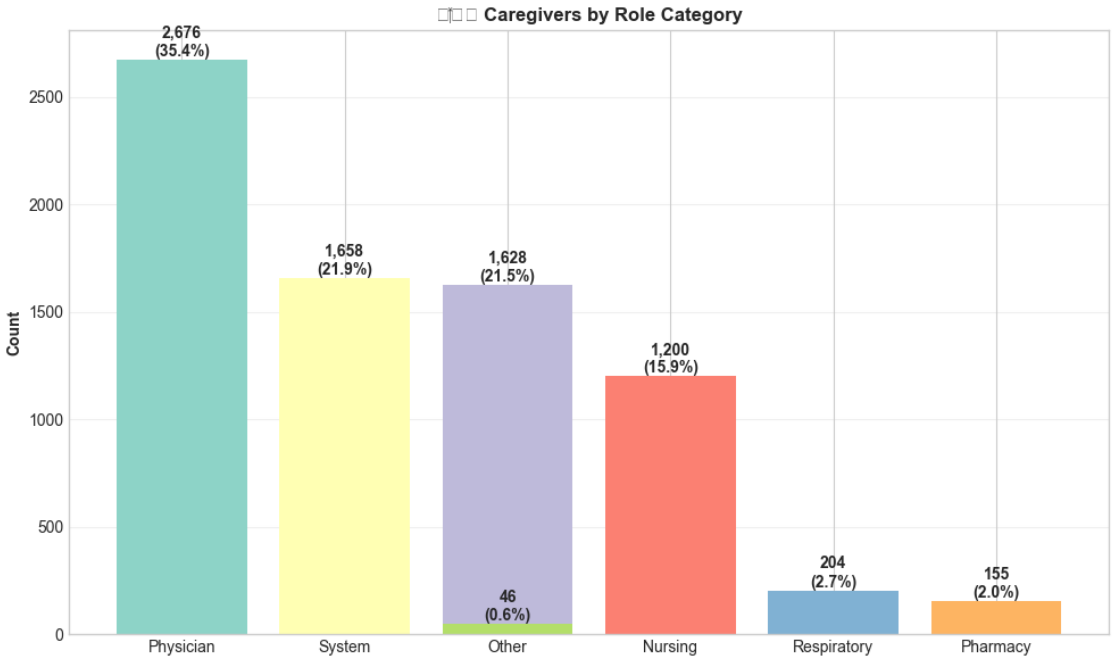
### 1.6 5. Caregiver Analysis

[12]:
```python
# Caregiver overview
try:
    cg_count = table_count('silver', 'caregivers')
    print(f"  Total Caregivers: {cg_count:,}")

    cg_roles = query_df("""
        SELECT
            COALESCE(role_category, 'Other') as category,
            COUNT(*) as count
        FROM silver.caregivers
        GROUP BY role_category
        ORDER BY count DESC
    """)
    print("\n  CAREGIVER ROLES")
    print(cg_roles.to_string(index=False))

    # Visualization
    if len(cg_roles) > 0:
        fig, ax = plt.subplots(figsize=(10, 6))
        bars = ax.bar(cg_roles['category'], cg_roles['count'],
                      color=sns.color_palette('Set3', len(cg_roles)))
        ax.set_ylabel('Count', fontweight='bold')
        ax.set_title('  Caregivers by Role Category', fontweight='bold')
        ax.grid(axis='y', alpha=0.3)

        # Add labels
        total_cg = cg_roles['count'].sum()
        for bar, row in zip(bars, cg_roles.itertuples()):
            height = bar.get_height()
            pct = (height/total_cg*100)
            ax.text(bar.get_x() + bar.get_width()/2., height,
                   f'{int(height):,}\n({pct:.1f}%)',
                   ha='center', va='bottom', fontweight='bold')

        plt.tight_layout()
        plt.show()
        print("\n Labels show count and percentage")

except Exception as e:
```

```
    print(f"  Caregivers table not available: {e}")
    print("Run: python scripts/load_silver.py --table caregivers")
```

```
2026-01-01 14:28:07,115 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2026-01-01 14:28:07,116 INFO sqlalchemy.engine.Engine SELECT COUNT(*) FROM
silver.caregivers
2026-01-01 14:28:07,117 INFO sqlalchemy.engine.Engine [cached since 116.7s ago]
{}
2026-01-01 14:28:07,121 INFO sqlalchemy.engine.Engine COMMIT
  Total Caregivers: 7,567
2026-01-01 14:28:07,126 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2026-01-01 14:28:07,127 INFO sqlalchemy.engine.Engine
        SELECT
            COALESCE(role_category, 'Other') as category,
            COUNT(*) as count
        FROM silver.caregivers
        GROUP BY role_category
        ORDER BY count DESC

2026-01-01 14:28:07,128 INFO sqlalchemy.engine.Engine [generated in 0.00102s] {}
2026-01-01 14:28:07,140 INFO sqlalchemy.engine.Engine COMMIT

  CAREGIVER ROLES
   category  count
  Physician   2676
     System   1658
      Other   1628
    Nursing   1200
Respiratory    204
   Pharmacy    155
      Other     46
C:\Users\ayoub\AppData\Local\Temp\ipykernel_25828\726879736.py:35: UserWarning:
Glyph 128104 (\N{MAN}) missing from font(s) Arial.
  plt.tight_layout()
C:\Users\ayoub\AppData\Local\Temp\ipykernel_25828\726879736.py:35: UserWarning:
Glyph 9877 (\N{STAFF OF AESCULAPIUS}) missing from font(s) Arial.
  plt.tight_layout()
C:\Users\ayoub\AppData\Local\Temp\ipykernel_25828\726879736.py:35: 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 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)
```




```
Labels show count and percentage
```

---

## 1.7 6. Summary

### 1.7.1 Key Findings:

- All visualizations include value labels
- Labels show both absolute values and percentages
- Data quality validated across all tables
- Caregiver integration successful

[13]:
```
print("\n" + "="*60)
print("  SILVER LAYER ANALYSIS COMPLETE")
print("="*60)
print(f"Total Tables: {len(summary_df)}")
print(f"Total Records: {total_records:,}")
print("\n  All charts include value indicators")
print("  Ready for Gold layer transformation")
```

```
============================================================
  SILVER LAYER ANALYSIS COMPLETE
============================================================
Total Tables: 11
Total Records: 157,027

  All charts include value indicators
  Ready for Gold layer transformation
```