

Technical Report: Final Project DS 5110: Introduction to Data Management and Processing

Team Members: Mia Khan and Onn Ye Young

Khoury College of Computer Sciences

Data Science Program

khan.mia@northeastern.edu and young.on@northeastern.edu

December 5, 2024

Contents

| | | |
|----------|--|----------|
| 1 | Introduction | 3 |
| 2 | Literature Review | 3 |
| 3 | Methodology | 4 |
| 3.1 | Data Collection | 4 |
| 3.2 | Data Preprocessing | 4 |
| 3.3 | Analysis Techniques | 4 |
| 4 | Results and Analysis | 5 |
| 4.1 | Data Visualization | 5 |
| 4.1.1 | Age Distribution | 5 |
| 4.1.2 | Income Distribution | 6 |
| 4.1.3 | Religion Distribution | 7 |
| 4.1.4 | Marriage Status Distribution | 8 |
| 4.1.5 | Procedure Types Distribution | 9 |
| 4.1.6 | Procedure Cost Distribution | 9 |
| 4.1.7 | Procedure Cost Distribution (Less than 20,000 PKR) | 10 |
| 4.1.8 | Procedure Cost Distribution (More than 20,000 PKR) | 11 |
| 4.1.9 | Average Procedure Cost by Type | 11 |
| 4.1.10 | Average Age per Procedure | 12 |
| 4.1.11 | Income Distribution by Marriage Status | 13 |
| 4.1.12 | Income by Occupation | 13 |
| 4.2 | SQL Queries | 14 |
| 4.2.1 | Informational Reports | 14 |
| 4.2.1.1 | Patient Demographic Report | 14 |
| 4.2.1.2 | Procedure History Report | 14 |
| 4.2.1.3 | Financial Aid Report | 15 |
| 4.2.1.4 | Zakat Eligibility Report | 15 |

| | | |
|----------|--|-----------|
| 4.2.1.5 | Financial Overview Report | 16 |
| 4.2.1.6 | Patient Financial Summary Report | 16 |
| 4.2.1.7 | Patient Contact Information Report | 17 |
| 4.2.2 | Analytical Reports | 17 |
| 4.2.2.1 | Dependency and Financial Burden Report | 17 |
| 4.2.2.2 | Monthly Aid Distribution Report | 18 |
| 4.2.2.3 | Occupation-Based Financial Analysis Report | 18 |
| 4.2.2.4 | Procedure Cost Analysis Report | 18 |
| 4.2.2.5 | Patient Aid Dependence Ratio Report | 19 |
| 4.3 | Dashboard | 19 |
| 4.3.1 | Dashboard Structure | 19 |
| 4.3.2 | Interactive Features | 20 |
| 4.3.3 | Key Visualizations | 21 |
| 4.3.3.1 | Procedure Analysis | 21 |
| 4.3.3.2 | Demographic Insights | 21 |
| 4.3.3.3 | Financial Analysis | 21 |
| 4.3.4 | Technical Implementation | 21 |
| 4.3.5 | Benefits | 22 |
| 5 | Discussion | 22 |
| 6 | Conclusion | 23 |
| 7 | References | 24 |
| A | Appendix A: Code | 24 |
| A.1 | Data Cleaning and Preprocessing | 24 |
| A.2 | Additional Code for Aid Type and Amount | 32 |
| A.3 | Data Visualization | 33 |
| A.4 | Exporting Cleaned Database | 36 |
| A.5 | SQL Queries and Reports | 39 |
| B | Appendix B: Additional Figures | 45 |
| B.1 | Correlation Matrix Heatmap | 45 |
| B.2 | Procedure Types by Occupation | 46 |
| B.3 | Procedure Types by Occupation pc | 46 |

1 Introduction

The goal of this project is to create a database for a charity medical foundation, called the Dow Patient Care Association (DPCA), which is a student-led NGO that aims to financially assist the non-affording patients of Dow Hospital in Pakistan. This charity foundation helps eligible patients pay for their medical procedures through a reimbursement process, taking into account factors such as occupation, income, and alternative forms of financial support.

The creation of this database would allow members of the foundation to easily perform data analysis and run queries on the data, creating static and dynamic visualizations and utilizing a dashboard to filter the data accordingly. This would provide the foundation valuable information on procedure requests and trends, as well as analysis of patient demographics and financials.

The scope of this project includes a populated database in SQL that has been subsequently cleaned, standardized, transformed, and validated in Python, as well as queries on the data, static and interactive data visualizations, and a dashboard displaying the visualizations.

2 Literature Review

The Global Burden of Diseases, Injuries, and Risk Factors Study (GBD) was carried out in 2019 for Pakistan, comparing health indicators since 1990 to provide insights to strengthen the health care system, reduce inequalities, and improve patient outcomes. In order to obtain these health indicators, the GBD 2019 utilized data inputs to estimate factors including the socio-demographic index, healthy life expectancy, and risk factors [1]. This was done at the national and subnational levels, for Pakistan as a whole and its four provinces and three territories [1].

From these estimates, it was found that “Pakistan is showing signs of undergoing an epidemiological transition as the burden shifts to NCDs” [1]. NCDs are noncommunicable diseases, which largely consist of “cardiovascular diseases, diabetes, cancers, and chronic respiratory diseases” [2]. Worldwide, NCDs are about 74% of all deaths globally, with 85% of premature NCD deaths occurring in low- and middle-income countries [2]. To reduce the risk of NCDs, efforts focused on informing young populations about lifestyle choices and medical interventions, as well as investments in preventative medicine and health systems are key [1].

As part of these efforts in bolstering preventative medicine and health systems, the Pakistan Government launched the Sehat Sahulat Programme in 2016, covering the cost of most medical expenses at over 150 hospitals in Pakistan for those below the poverty line - 80 million people as of 2019 [3]. Programs such as this one, in addition to foundations such as the DPCA, are imperative in reducing present and future healthcare burdens in Pakistan by increasing healthcare accessibility across the country. Data analysis on patient demographics, resource allocation, and healthcare costs allow for evaluation of program success and provide pathways for improvements in healthcare systems and program outreach.

3 Methodology

3.1 Data Collection

The data in the dataset is collected from patients who go to the DPCA. Employees of the DPCA will ask patients various questions, filling out a Google Form that gets inputted into a Google Sheet that contains all the data.

3.2 Data Preprocessing

After the database was created and populated in SQL, each table was imported into Python for cleaning and preprocessing. To clean the data, the duplicate headers were first removed and the proper data types were set. Then, the dataframes were combined into a dataframe with all of the data from the database.

With all of the data in a single dataframe, duplicate rows and columns were checked for and dropped. Following this step, some of the data was then mapped, changing text into numbers for easier processing.

The next step was then to find the number of null values in each column and replace them with the appropriate values, such as "N.A", "Unknown", or 0, depending on the category.

Once there were no longer any null values, functions to clean different columns were created, replacing non-numeric values and strings with NaN, and handling range values by taking the midpoint of the range. Once the column was cleaned, the values were set to the proper data type.

With the data cleaned, standardized, and transformed, it was then validated, ensuring that the values contained within the database made logical sense. This includes validating that there are no values less than 0 for categories such as procedure cost, income, or age.

3.3 Analysis Techniques

The analysis of the data was conducted using various visualization methods to uncover trends and patterns. These techniques included:

- **Histograms:** Used to understand the distribution of numerical variables such as procedure costs and patient ages.
- **Bar Plots:** Applied to compare categorical variables like procedure types and their average costs.
- **Pie Charts:** Utilized to show proportional distributions, such as the breakdown of patients by religion or marriage status.
- **Box Plots:** Used for examining the spread and outliers in income distributions across categories like occupation and marriage status.

The visualizations were created using Python libraries such as Matplotlib and Seaborn, ensuring clarity and effective communication of results.

4 Results and Analysis

4.1 Data Visualization

4.1.1 Age Distribution

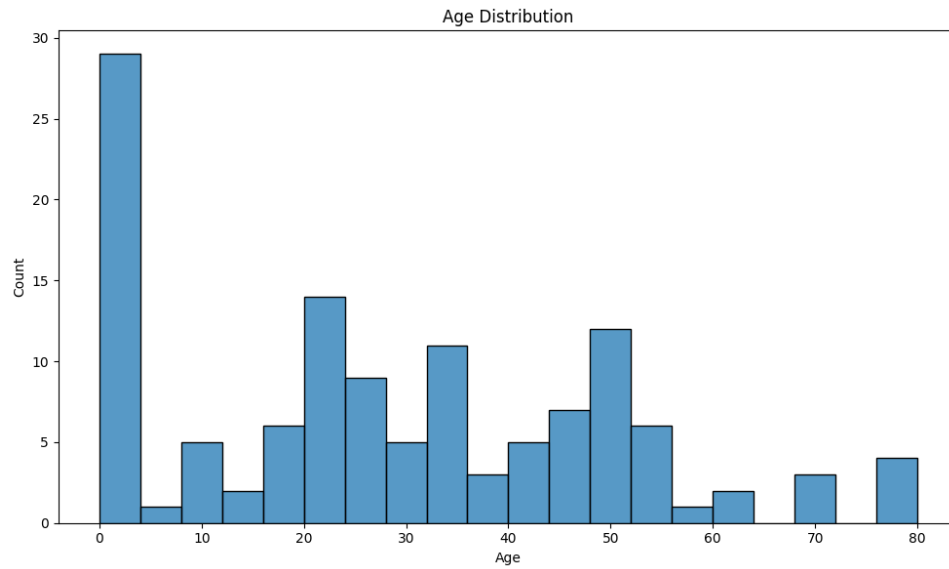


Figure 1: Age Distribution

Figure 1 highlights the age distribution of patients. The population is relatively young, with the largest group being 0–10 years old. This suggests a need for pediatric coverage. The working-age population (20–50 years) might benefit from partial reimbursement schemes, while the smaller elderly population (60+) may require specialized care.

4.1.2 Income Distribution

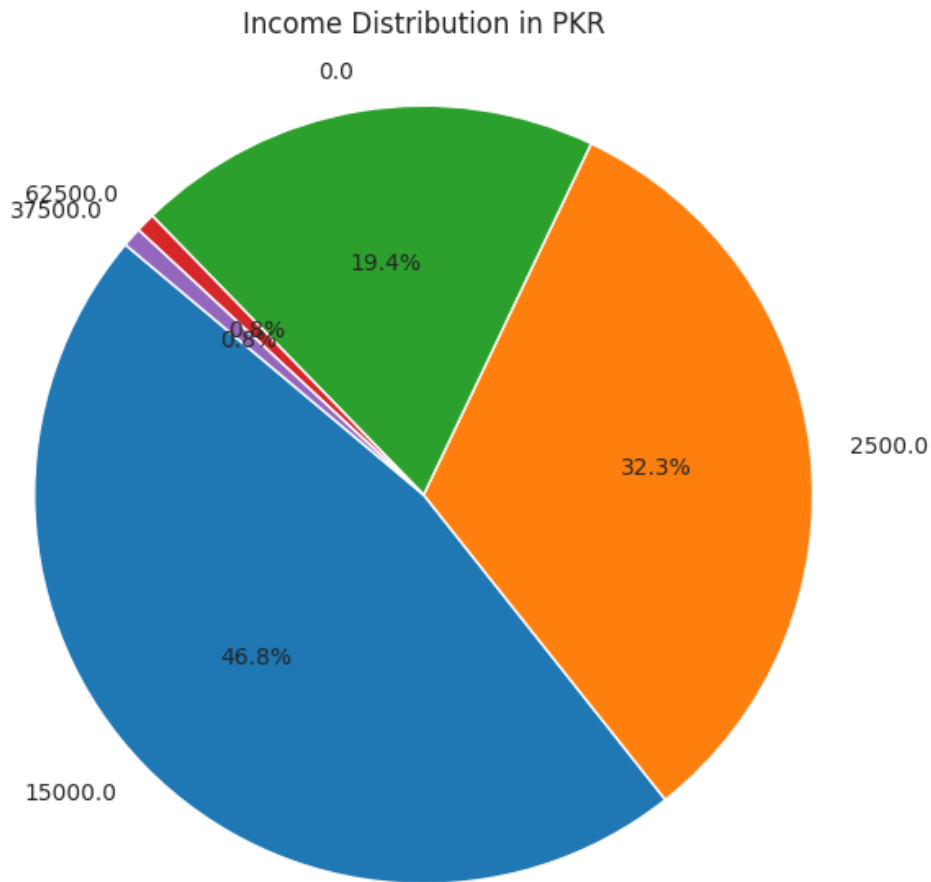


Figure 2: Income Distribution in PKR/month

The pie chart in Figure 2 illustrates the distribution of monthly income across patients. 98.5% of the population earns 15,000 PKR or less. As of November 2024, 15,000 PKR equates to 54 USD. Included in this 98.5% is 20% of the total population that earns no income, requiring substantial support, and 32% that earn only 2,500 PKR, likely needing full reimbursement. The 46.8% in the 15,000 PKR bracket could qualify for partial reimbursement, while less than 2% in highest brackets of 37,500 to 62,500 PKR might need minimal support. A large percentage of the population falls into lower-income brackets, highlighting economic disparities.

4.1.3 Religion Distribution

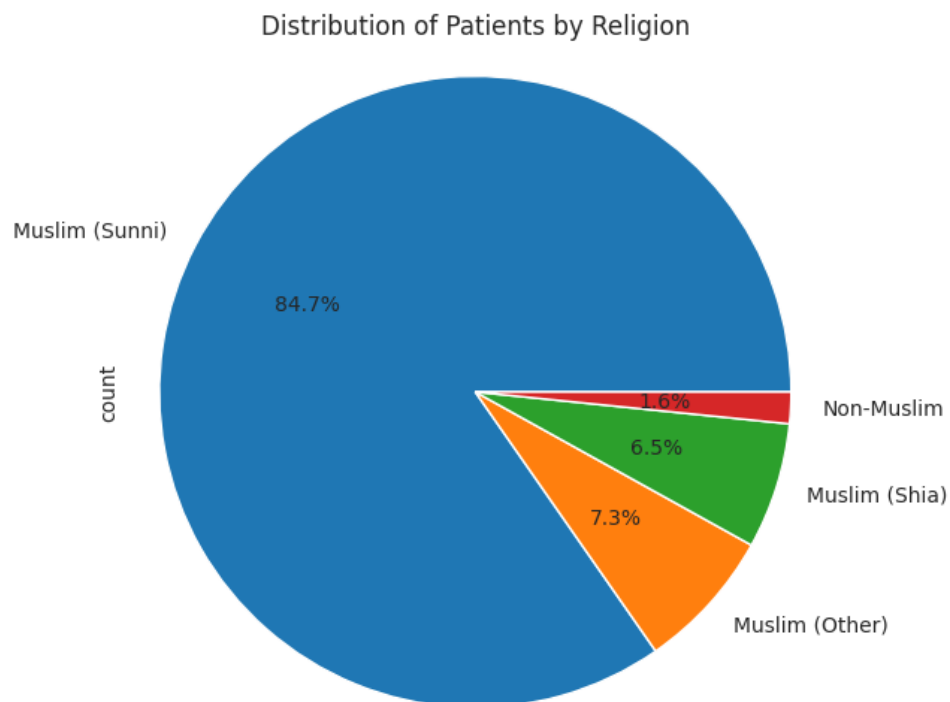


Figure 3: Distribution of Patients by Religion

Figure 3 shows the proportion of patients belonging to different religions. The population served is predominantly Sunni (84.7%), with the next largest groups being Muslim (Other) and Muslim (Shia), followed by Non-Muslim. This understanding could guide culturally sensitive outreach programs and identify potential gaps in service delivery to other communities.

4.1.4 Marriage Status Distribution

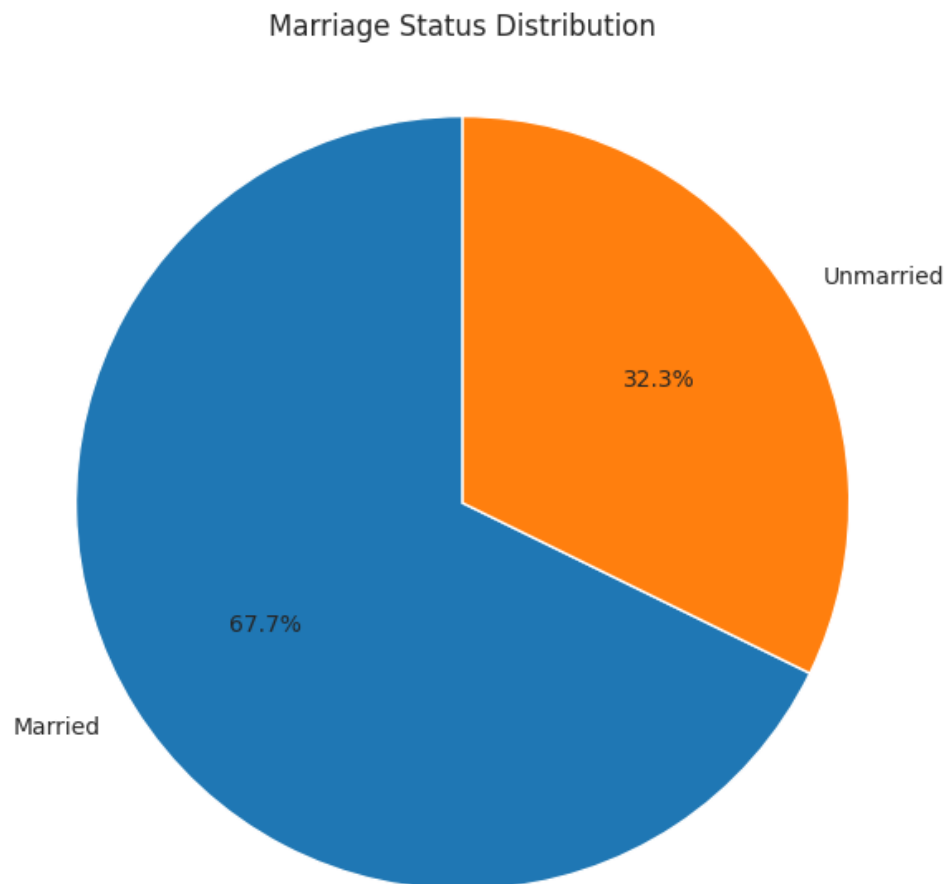


Figure 4: Marriage Status Distribution

In Figure 4, the marriage status distribution of patients is presented. A majority of patients (67.7%) are married, indicating the potential for family-based coverage plans. However, unmarried patients (32.3%) might need more individualized support options.

4.1.5 Procedure Types Distribution

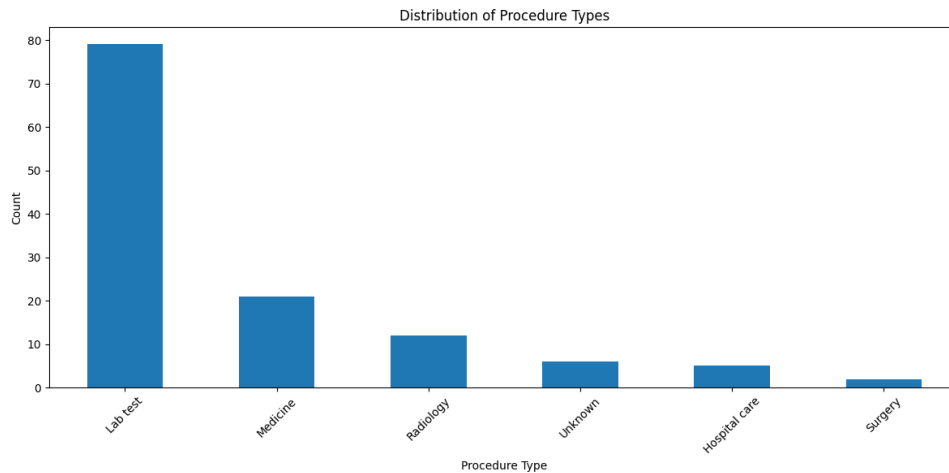


Figure 5: Distribution of Procedure Types

Figure 5 shows the different procedure types. Lab tests are the most common procedures (78 cases), reflecting high demand for diagnostic services. Medicine-related procedures are the second most common, while surgeries are the least utilized. This trend suggests prioritization of diagnostic service reimbursements and potential bulk negotiation with labs.

4.1.6 Procedure Cost Distribution

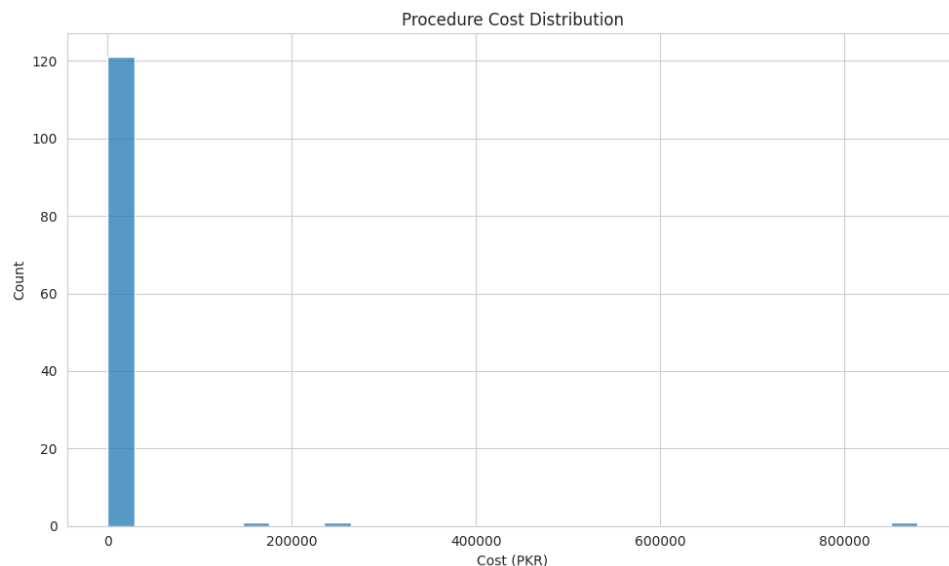


Figure 6: Distribution of Procedure Costs

The histogram in Figure 6 shows the distribution of procedure costs. Most procedures fall within the low-cost range, highlighting the foundation's focus on basic medical needs. Fewer procedures fall into high-cost ranges, so suggesting tiered reimbursement thresholds could be implemented to allocate funds effectively. Because there is a wide range of

procedure costs displayed, Figure 6 does not provide detailed insights on low and high cost procedure distributions. Thus, Figure 7 and Figure 8 were created, focusing on the lower and higher ends of procedure costs, respectively.

4.1.7 Procedure Cost Distribution (Less than 20,000 PKR)

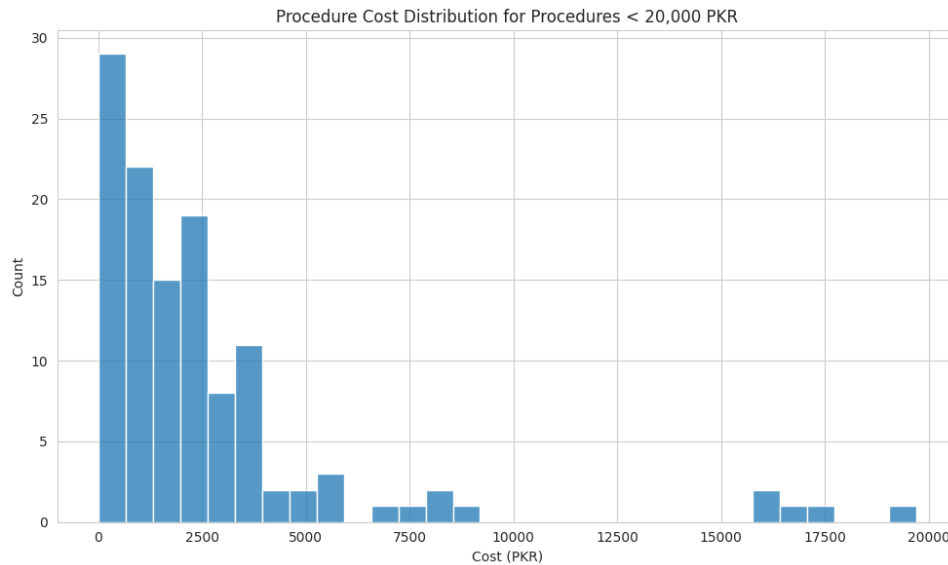


Figure 7: Distribution of Procedure Costs

Figure 7 shows the distribution of procedure costs that are less than 20,000 PKR. With 98.5% of the patients requesting aid making 15,000 PKR or less a month, according to Figure 2, the cost of a single procedure could cause significant financial burden to a patient and their household. Furthermore, even with many procedures costing 2,500 PKR or less, 2,500 PKR or less is the monthly income of over 50% of patients requesting aid from DPCA (Figure 2).

Within the lower cost procedures, there seems to be two groups: one group with costs below 10,000 PKR, and another with costs between 15,000 and 20,000 PKR. These groups likely consist of different procedure types. The average cost per procedure type will be discussed further in Figure 9.

4.1.8 Procedure Cost Distribution (More than 20,000 PKR)

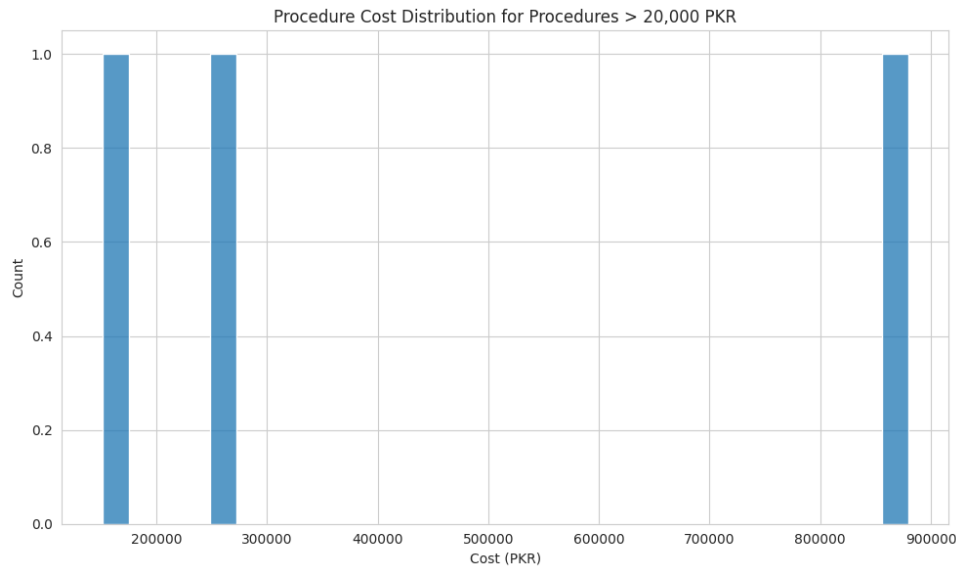


Figure 8: Distribution of Procedure Costs

Figure 8 shows the distribution of procedure costs that are more than 20,000 PKR. There are only 3 procedure aid requests that fall within this range, but they are far above the monthly incomes of patients, with the least expensive of these three procedures ($\approx 175,000$ PKR) nearly double the highest monthly income recorded (67,500 PKR).

4.1.9 Average Procedure Cost by Type

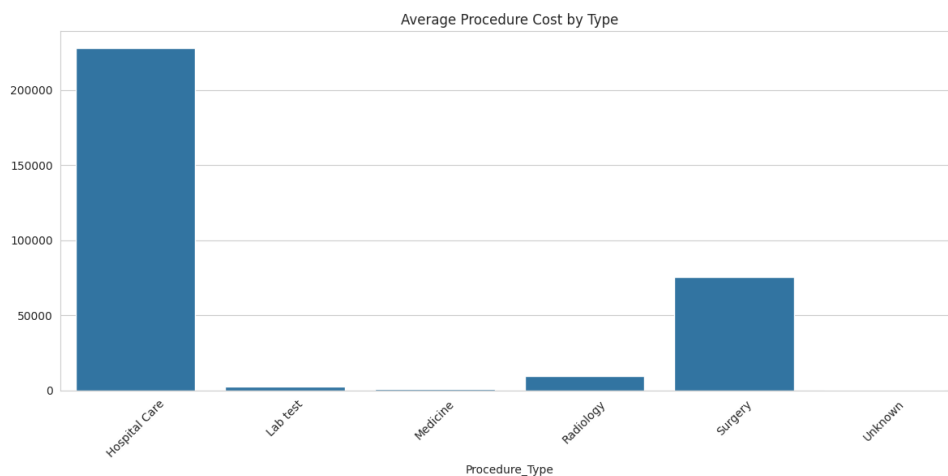


Figure 9: Average Procedure Cost by Type

In Figure 9, the average cost per procedure type is shown. Radiology procedures are the most expensive ($\approx 7,500$ PKR), followed by lab tests ($\approx 2,500$ PKR). Medicine and surgery are more affordable, potentially requiring less financial aid. This helps identify which procedures are costlier on average and could guide resource allocation or cost management strategies.

4.1.10 Average Age per Procedure

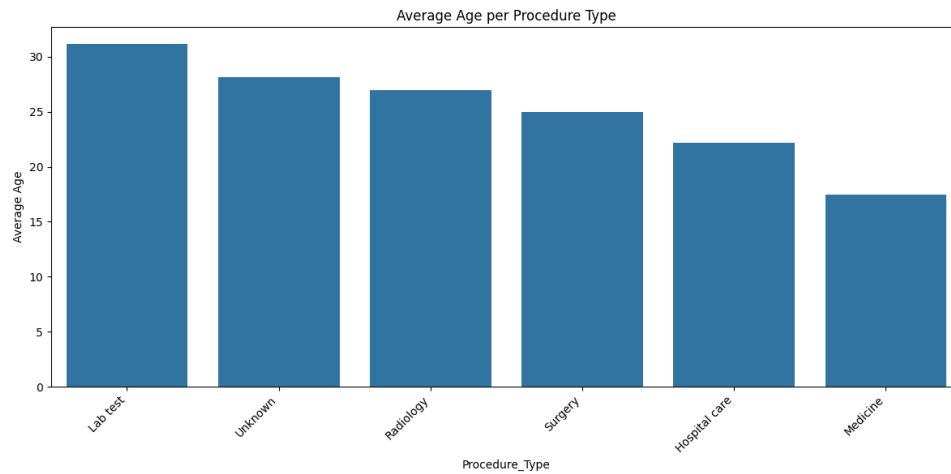


Figure 10: Average Age per Procedure Type

Figure 10 displays the average age of patients for each procedure type. Older patients (average age around 30) tend to undergo lab tests, reflecting a need for diagnostic services in this age group. Medicine-related procedures are most common among younger patients (average age around 17), indicating their relevance for pediatric and adolescent care. Radiology and other procedures fall within a moderate age range (20 to 31), suggesting a balanced demand across age groups. As a result of this data, age-specific outreach and reimbursement plans can be tailored for different procedure types, as younger patients may benefit from medicine-focused coverage and older patients may require enhanced support for diagnostic and advanced procedures.

4.1.11 Income Distribution by Marriage Status

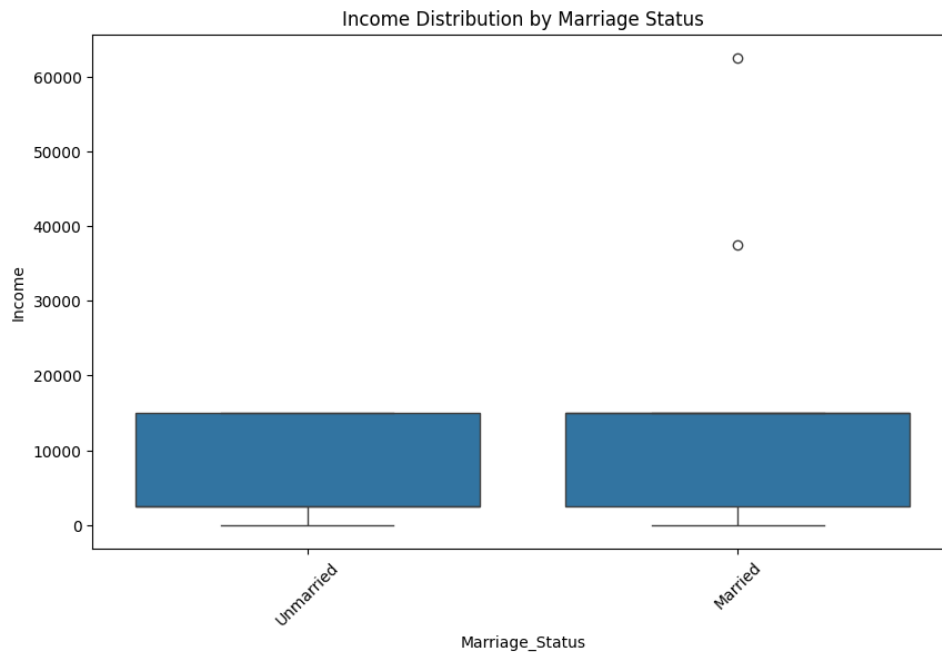


Figure 11: Income Distribution by Marriage Status

The box plot in Figure 11 compares income distribution based on marital status. Both married and unmarried groups display similar median income levels, suggesting that marital status alone may not strongly influence income. The married group tends to have slightly more outliers than the unmarried group, possibly due to dual-income households or shared financial resources.

4.1.12 Income by Occupation

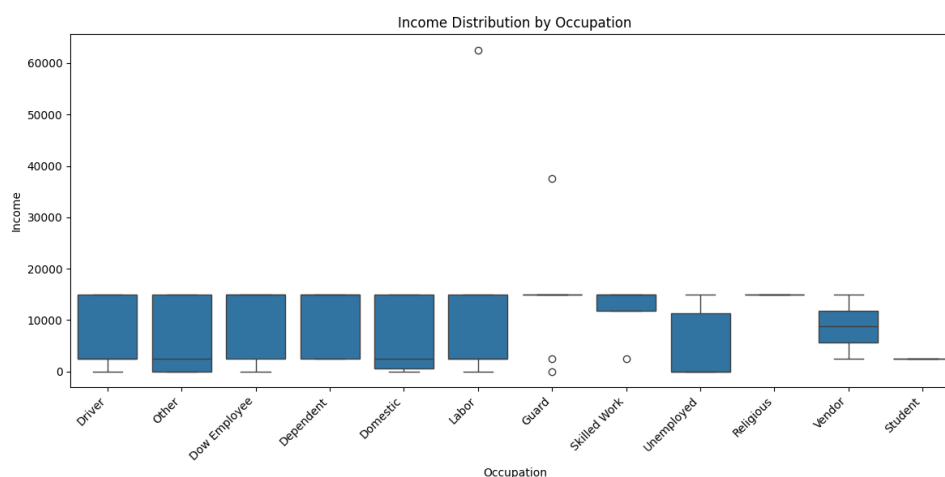


Figure 12: Income Distribution by Occupation

The box plot in Figure 12 examines incomes across various occupations. The box plot shows considerable variation in income across different occupations. Occupations

like Skilled Workers have a noticeably higher median income compared to other groups, which could suggest a more stable or specialized labor market for these roles. Guards, Laborers, and Domestic Workers display consistently lower median incomes, highlighting the potential need for additional financial support. Significant income inequality is observed across occupations, especially in categories like Guards and Religious occupations, where some individuals earn much higher than others (outliers). This could point to the presence of a few high earners within these occupations, but the majority earn low wages.

4.2 SQL Queries

SQL queries were performed on the data to gain a better understanding of the data and draw meaningful conclusions. A total of 12 reports were generated, grouped into informational reports, which focused on pulling specific data, and analytical reports, which performed some sort of computation for analysis. Please see Appendix A.5 for the SQL query code for each of the following reports.

4.2.1 Informational Reports

4.2.1.1 Patient Demographic Report

The Patient Demographic Report lists patient details, such as name, age, religion, marital status, and contact information. This allows employees to quickly view patient demographics.

The first 5 results of this query are as follows, with results ordered by Patient ID.

| Patient_ID | Patient_Name | Age | Religion | Marriage_Status | Contact_No | Address |
|------------|-----------------|------|----------------|-----------------|----------------|---------------------------------------|
| 1 | Zaur Hussain | 25.0 | Muslim (Shia) | Unmarried | +92 3241407991 | Aib Ghout |
| 2 | Sumera | 25.0 | Muslim (Sunni) | Married | +92 3032900156 | A-63 Muhalla Yusuf Khan Goth Malir |
| 3 | Muhammad Shafiq | 46.0 | Muslim (Sunni) | Married | 3022029763 | Kaj Muhammad Brohi R-57 Karachi Gharb |
| 4 | Fehmeeda | 0.0 | Muslim (Sunni) | Married | 03083599048 | Safoora, karachi |
| 5 | Fehmeeda | 0.0 | Muslim (Sunni) | Married | 03083599048 | Safoora, karachi |

Table 1: Patient Demographics Report

4.2.1.2 Procedure History Report

The Procedure History Report displays details of medical procedures performed, including procedure types, dates, costs, and payment information. This report helps track patient procedures and associated financial transactions.

| Patient_ ID | Patient_ Name | Procedure_ Type | Procedure_ Date | Procedure_ Cost | Amount_ Paid_ DPCA | Amount_ Paid_ Patient | MR_ number |
|----------------|------------------|--------------------|--------------------|--------------------|--------------------------|-----------------------------|---------------|
| 1 | Zaur Hussain | Lab test | 9/11/2024 12:44:36 | 4,740.0 | 4,740.0 | 0.0 | Unknown |
| 2 | Sumera | Lab test | 9/11/2024 13:09:09 | 3,510.0 | 3,510.0 | 0.0 | Unknown |
| 3 | Muhammad Shafiq | Lab test | 9/11/2024 13:22:51 | 350.0 | 350.0 | 0.0 | Unknown |
| 4 | Fehmeeda | Lab test | 9/12/2024 13:48:51 | 3,730.0 | 3,730.0 | 0.0 | Unknown |
| 5 | Fehmeeda | Medicine | 9/12/2024 14:31:46 | 1,371.0 | 1,371.0 | 0.0 | Unknown |

Table 2: Procedure History Report

4.2.1.3 Financial Aid Report

The Financial Aid Report shows details of financial assistance provided to patients, including aid types, amounts, and their relationship to patient income. This report helps track the distribution of financial support across different procedures.

| Patient_ ID | Patient_ Name | Aid_ Date | Aid_ Type | Aid_ Amount | Monthly_ Income | Aid_Percentage_ of_ Income | Procedure_ Type |
|----------------|------------------|---------------------|--------------|----------------|--------------------|----------------------------------|--------------------|
| 1 | Zaur Hussain | 2024-09-11 12:44:36 | Zakat Funds | 4,740.0 | 2,500.0 | 189.6 | Lab test |
| 2 | Sumera | 2024-09-11 13:09:09 | Zakat Funds | 3,510.0 | 15,000.0 | 23.4 | Lab test |
| 3 | Muhammad Shafiq | 2024-09-11 13:22:51 | Zakat Funds | 350.0 | 15,000.0 | 2.33 | Lab test |
| 4 | Fehmeeda | 2024-09-12 13:48:51 | Zakat Funds | 3,730.0 | 15,000.0 | 24.87 | Lab test |
| 5 | Fehmeeda | 2024-09-12 14:31:46 | Zakat Funds | 1,371.0 | 2,500.0 | 54.84 | Medicine |

Table 3: Financial Aid Report

4.2.1.4 Zakat Eligibility Report

The Zakat Eligibility Report provides detailed information about patients' eligibility for Zakat funds based on religious criteria, income status, and asset ownership. This report helps ensure proper distribution of Zakat funds according to Islamic guidelines.

| Patient_ID | Patient_Name | Religion | Syed_NonSyed | Is_Eligible | Eligibility_Criteria | Monthly_Income | Properties_Owned | Aid_Type |
|------------|-----------------|----------------|--------------|-------------|---|----------------|------------------|-------------|
| 1 | Zaur Hussain | Muslim (Shia) | Non-Syed | 1 | Muslim, Non-syed, Does not own an equivalence of 52.5 tola silver | 2,500.0 | 0 | Zakat Funds |
| 2 | Sumera | Muslim (Sunni) | Non-Syed | 1 | Muslim, Non-syed, Does not own an equivalence of 52.5 tola silver | 15,000.0 | 0 | Zakat Funds |
| 3 | Muhammad Shafiq | Muslim (Sunni) | Non-Syed | 1 | Muslim, Non-syed, Does not own an equivalence of 52.5 tola silver | 15,000.0 | 0 | Zakat Funds |
| 4 | Fehmeeda | Muslim (Sunni) | Non-Syed | 1 | Muslim, Non-syed, Does not own an equivalence of 52.5 tola silver | 15,000.0 | 1 | Zakat Funds |
| 5 | Fehmeeda | Muslim (Sunni) | Non-Syed | 1 | Muslim, Non-syed, Does not own an equivalence of 52.5 tola silver | 2,500.0 | 1 | Zakat Funds |

Table 4: Zakat Eligibility Report

4.2.1.5 Financial Overview Report

The Financial Overview Report provides a financial profile for each patient, displaying attributes like occupation, income, number of dependent members, number of properties owned, and financial support methods. This allows DPCA employees to quickly view each patient's financial information for tasks such as aid amount determination.

The first 5 results of this query are as follows, with results ordered by Patient ID.

| Patient_ID | Occupation | Income | Dependent_Members | Properties_Owned | Financial_Support_Method |
|------------|--------------|---------|-------------------|------------------|--------------------------|
| 1 | Driver | 2500.0 | 8.0 | 0 | None |
| 2 | Other | 15000.0 | 6.0 | 0 | None |
| 3 | Dow Employee | 15000.0 | 7.0 | 0 | None |
| 4 | Dow Employee | 15000.0 | 10.0 | 1 | Employed |
| 5 | Dow Employee | 2500.0 | 10.0 | 1 | Unknown |

Table 5: Financial Overview Report

4.2.1.6 Patient Financial Summary Report

The Patient Financial Summary Report offers a consolidated view of each patient's complete financial data, including personal financial information, procedure costs, and financial aid information. This provides a clear view of each patient's financial data, potentially assisting in aid tracking and aid allocation analysis.

| Patient_ID | Monthly_Income | Dependent_Members | Financial_Support | Properties_Owned | Procedure_Cost | Amount_Paid_DPCA | Aid_Date | Aid_Type |
|------------|----------------|-------------------|-------------------|------------------|----------------|------------------|---------------------|-------------|
| 1 | 2,500.0 | 8.0 | None | 0 | 4,740.0 | 4,740.0 | 2024-09-11 12:44:36 | Zakat Funds |
| 2 | 15,000.0 | 6.0 | None | 0 | 3,510.0 | 3,510.0 | 2024-09-11 13:09:09 | Zakat Funds |
| 3 | 15,000.0 | 7.0 | None | 0 | 350.0 | 350.0 | 2024-09-11 13:22:51 | Zakat Funds |
| 4 | 15,000.0 | 10.0 | Employed | 1 | 3,730.0 | 3,730.0 | 2024-09-12 13:48:51 | Zakat Funds |
| 5 | 2,500.0 | 10.0 | Unknown | 1 | 1,371.0 | 1,371.0 | 2024-09-12 14:31:46 | Zakat Funds |

Table 6: Patient Financial Summary Report

4.2.1.7 Patient Contact Information Report

The Patient Contact Information Report provides contact details of patients for follow-up or communication purposes, allowing DPCA employees to reach out to patients quickly. This includes contact numbers and address information.

The first 5 results of this query are as follows, with results ordered by Patient ID.

| Patient_ID | Patient_Name | Contact_No | Address |
|------------|-----------------|----------------|---------------------------------------|
| 1 | Zaur Hussain | +92 3241407991 | Aib Ghout |
| 2 | Sumera | +92 3032900156 | A-63 Muhalla Yusuf Khan Goth Malir |
| 3 | Muhammad Shafiq | 3022029763 | Kaj Muhammad Brohi R-57 Karachi Gharb |
| 4 | Fehmeeda | 03083599048 | Safoora, karachi |
| 5 | Fehmeeda | 03083599048 | Safoora, karachi |

Table 7: Patient Contact Information Report

4.2.2 Analytical Reports

4.2.2.1 Dependency and Financial Burden Report

The Dependency and Financial Burden Report provides an overview of patients' financial situations, including their income, number of dependents, and total medical costs. This report helps assess the financial burden of medical care on patients and their families.

| Patient_ID | Patient_Name | Monthly_Income | Dependent_Members | Income_Per_Dependent | Financial_Support_Method | Properties_Owned | Total_Procedures | Total_Medical_Costs |
|------------|---------------|----------------|-------------------|----------------------|--------------------------|------------------|------------------|---------------------|
| 10 | Sajid Mehmood | 0.0 | 0.0 | 0.0 | Welfare | 1 | 1 | 8,740.0 |
| 12 | Uzair faisal | 0.0 | 11.0 | 0.0 | Family | 1 | 1 | 1,300.0 |
| 28 | Anila | 0.0 | 9.0 | 0.0 | Family | 0 | 1 | 4,390.0 |
| 40 | Mrs Nazeera | 0.0 | 0.0 | 0.0 | Unknown | 0 | 1 | 8,400.0 |
| 44 | Roopa | 0.0 | 7.0 | 0.0 | Employed | 0 | 1 | 1,570.0 |

Table 8: Dependency and Financial Burden Report

4.2.2.2 Monthly Aid Distribution Report

The Monthly Aid Distribution Report provides an overview of all financial aid distributed by month, categorized by aid type. There are two aid types, Zakat and general. These aid types come out of separate funds.

The results of this query are shown below.

| month | Aid_Type | total_amount | average_aid_per_patient |
|---------|---------------|--------------|-------------------------|
| 2024-09 | General Funds | 11417.0 | 1902.83 |
| 2024-09 | Zakat Funds | 277397.0 | 9565.41 |
| 2024-10 | General Funds | 81580.0 | 2204.86 |
| 2024-10 | Zakat Funds | 94939.0 | 1825.75 |

Table 9: Monthly Aid Distribution Report

From the results of this query, it is shown that September utilized significantly more Zakat funds, with each patient getting an average of 7500 PKR more than those utilizing general funds. However, in October, while more total Zakat funds are being used, patients utilizing general funds are getting more aid each on average, indicating fewer patients with higher cost procedures.

4.2.2.3 Occupation-Based Financial Analysis Report

The Occupation-Based Financial Analysis Report summarizes financial metrics across different occupational categories, including income levels, dependency ratios, and aid distribution. This report helps analyze financial assistance patterns across different occupational groups.

The first 5 results of this query are as follows, with results ordered by the total aid given, in descending order.

| Occupation | Average_Income | Average_Dependents | Average_Aid_Amount | Total_Procedures | Average_Procedure_Cost | Total_Aid_Given |
|------------|----------------|--------------------|--------------------|------------------|------------------------|-----------------|
| Other | 5,781.25 | 3.81 | 13,851.44 | 16 | 57,253.56 | 221,623.0 |
| Domestic | 7,500.0 | 6.5 | 4,027.05 | 22 | 4,184.3 | 88,595.0 |
| Labor | 8,382.35 | 5.62 | 1,420.44 | 34 | 9,781.21 | 48,295.0 |
| Guard | 14,687.5 | 5.63 | 2,103.75 | 16 | 1,948.75 | 33,660.0 |
| Driver | 7,000.0 | 8.8 | 4,556.0 | 5 | 6,096.0 | 22,780.0 |

Table 10: Occupation-Based Financial Analysis Report

The results of this query show that those with a miscellaneous occupation have the most amount of total aid, with the highest average procedure cost. This is followed by domestic workers, laborers, guards, and drivers. It is interesting to note that while domestic workers and guards received more total aid than laborers and drivers, respectively, laborers and drivers had higher average procedure costs.

4.2.2.4 Procedure Cost Analysis Report

The Procedure Cost Analysis Report summarizes the costs associated with different types of procedures, identifying which treatments are affordable and which are high-

cost. This query finds the average procedure cost of each procedure type, sorting average procedure costs from lowest to highest.

| Procedure_Type | Average_Procedure_Cost |
|----------------|------------------------|
| Unknown | 304.00 |
| Medicine | 771.66 |
| Lab test | 2,463.31 |
| Radiology | 9,291.66 |
| Surgery | 75,518.00 |
| Hospital Care | 227,809.20 |

Table 11: Procedure Cost Analysis Report

This query shows that of the known procedures, medicine and lab tests are relatively cheap, with radiology within a middle price range, and surgery and hospital care as the most expensive. This aligns with expectations of procedure costs, as medicine and lab tests are typically a part of annual diagnostics and health exams, while radiology, surgery, and hospital care are procedures that are required for more serious or complex incidents.

4.2.2.5 Patient Aid Dependence Ratio Report

The Patient Aid Dependence Ratio Report analyzes the ratio of aid received compared to the patient's income. This ratio is calculated by dividing the amount of aid given to the patient by the patient's monthly income. The higher the ratio, the greater the impact paying for the procedure without aid would have on a patient's monthly financial burden.

This query finds the aid dependence ratio for each patient, ordering results by Patient ID, but ordering results based on income, aid given, or the aid dependence ratio could also be insightful. The first 5 results of the query are shown below.

| Patient_ID | Amount_Paid_DPCA | Income | aid_to_income_ratio |
|------------|------------------|---------|---------------------|
| 1 | 4740.0 | 2500.0 | 1.896 |
| 2 | 3510.0 | 15000.0 | 0.234 |
| 3 | 350.0 | 15000.0 | 0.0233 |
| 4 | 3730.0 | 15000.0 | 0.2486 |
| 5 | 1371.0 | 2500.0 | 0.5484 |

Table 12: Patient Aid Dependence Ratio Report

4.3 Dashboard

The DPCA Dashboard is an interactive data visualization tool designed to analyze and present patient and procedure data. Here's a breakdown of its key components and functionality:

4.3.1 Dashboard Structure

The dashboard consists of multiple tabs for different analytical purposes:

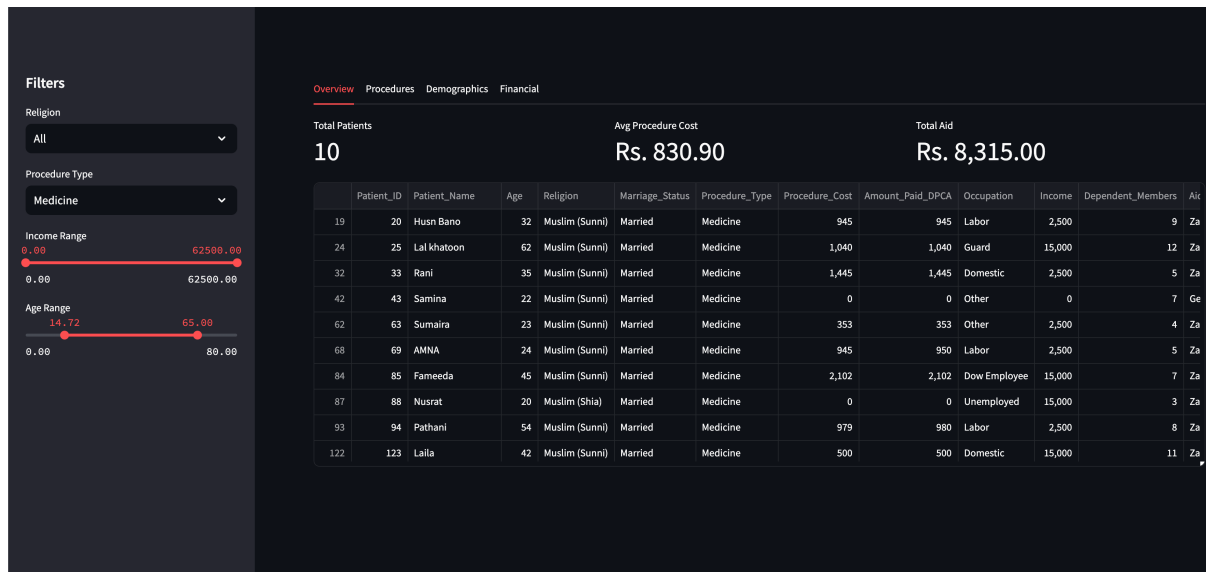


Figure 13: Overview dashboard displaying key metrics and detailed patient records

- **Overview Tab:** Provides key metrics including total patients, average procedure costs, and total aid disbursed
- **Procedures Tab:** Visualizes procedure distribution and costs
- **Demographics Tab:** Shows patient distribution by religion and age
- **Financial Tab:** Displays financial metrics including income distribution and aid allocation

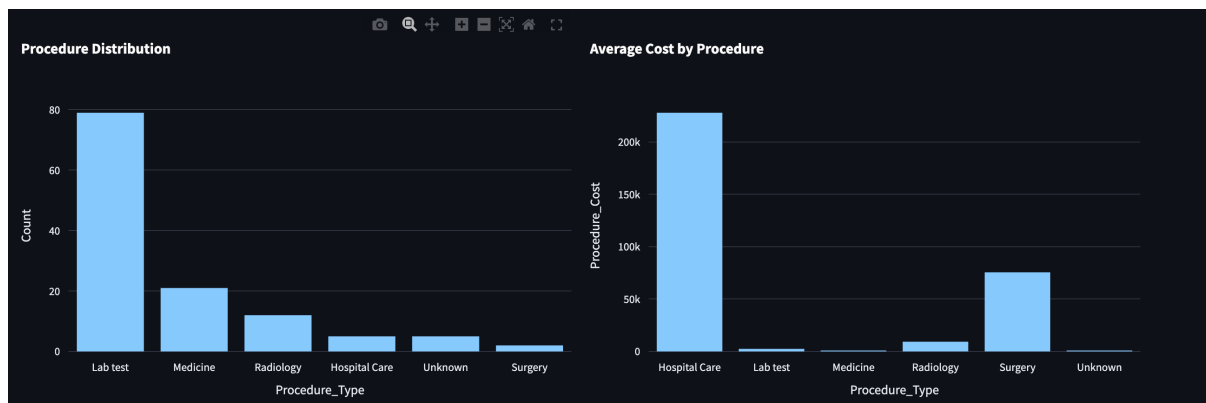


Figure 14: Procedures dashboard showing procedure distribution and average costs

4.3.2 Interactive Features

The dashboard includes several interactive filtering options through its sidebar:

- **Sidebar Filters:**
 - Religion selector: Allows filtering data by patient religion

- Procedure Type selector: Enables viewing data for specific medical procedures
- Income Range slider: Filters patients based on income levels
- Age Range slider: Allows demographic filtering by age groups

4.3.3 Key Visualizations

4.3.3.1 Procedure Analysis

- Bar chart showing procedure distribution helps identify most common treatments
- Average cost visualization per procedure type enables cost pattern analysis

4.3.3.2 Demographic Insights

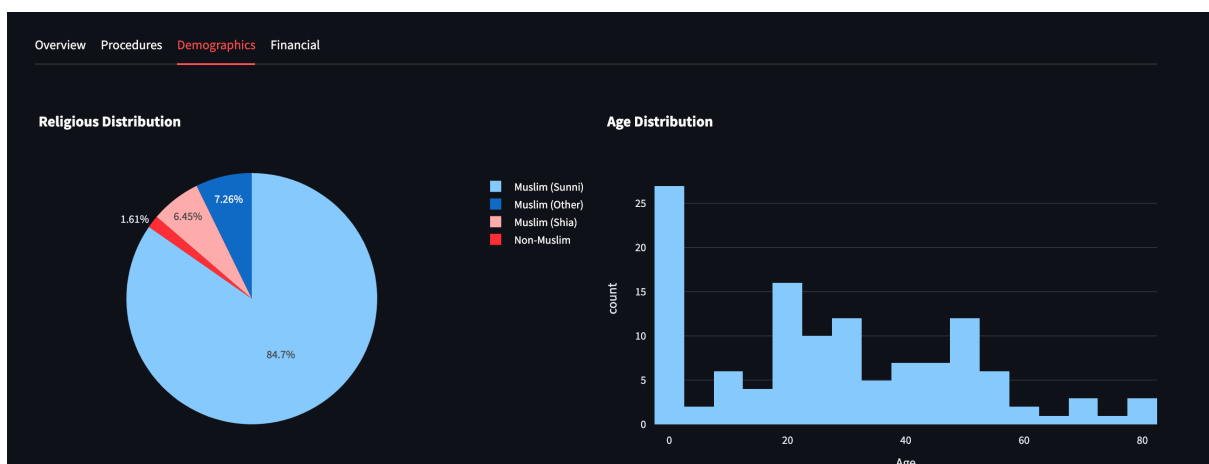


Figure 15: Demographics dashboard showing religious and age distribution of patients

- Pie chart of religious distribution shows patient demographic makeup
- Age distribution histogram reveals key age groups seeking assistance

4.3.3.3 Financial Analysis

- Income distribution by occupation helps understand socioeconomic patterns
- Aid amount vs. income scatter plot shows aid allocation patterns
- Financial metrics help assess program impact

4.3.4 Technical Implementation

The dashboard is built using modern web technologies:

- Built using Streamlit for web interface
- Uses Plotly for interactive visualizations
- Connects to SQLite database for data management
- Real-time filtering and data updates

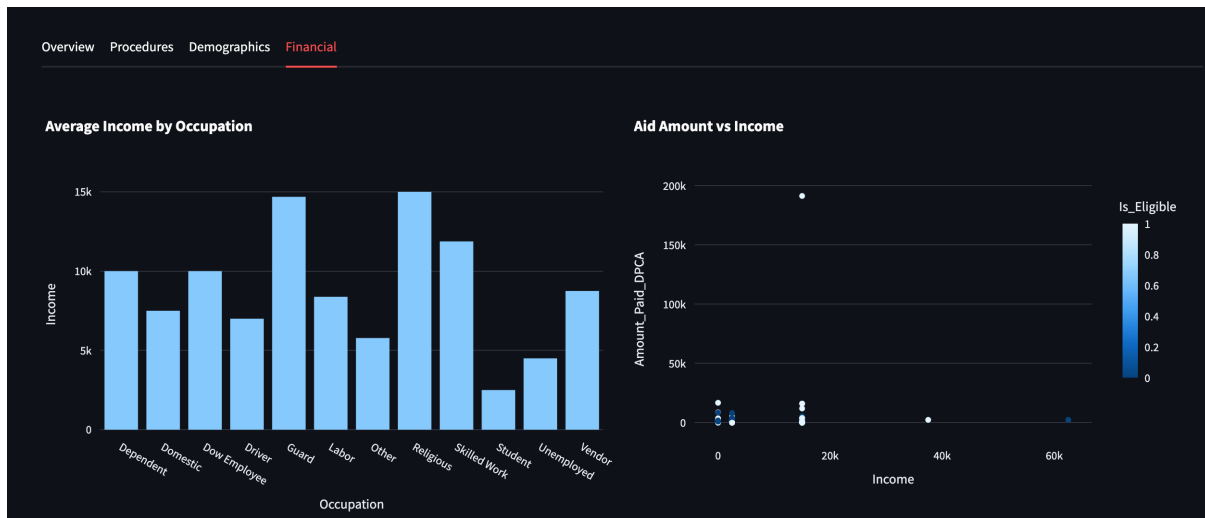


Figure 16: Financial metrics dashboard showing average income by occupation and aid amount distribution

4.3.5 Benefits

The dashboard provides numerous advantages for program management:

- Enables data-driven decision making
- Provides quick insights into patient demographics
- Helps track financial aid distribution
- Allows for pattern identification in medical procedures
- Facilitates resource allocation planning

This comprehensive dashboard (Figures 13–16) serves as a powerful tool for DPCA administrators to monitor program effectiveness, track aid distribution, and understand patient demographics, ultimately helping improve healthcare service delivery.

5 Discussion

In analyzing general patient demographic information, such as age, religion, and marriage status, a better image of who the DPCA serves is created. The most abundant patient age falls below 10 years old. However, there is a wide spread of data, with the DPCA providing funds to those in all stages of life (Figure 1). When looking at income, more than 98% of the patients make less than 15,000 PKR a month, indicating a dire need for financial support and highlighting the importance of programs and foundations that provide it (Figure 2). This is supported by the results of the Patient Aid Dependence Report (Table 12), which analyzes the ratio of aid received compared to a patient's income. An evaluation of patient religions show that 84.7% are Sunni, which could guide the creation of outreach programs to expand awareness of healthcare related financial aid programs to other communities (Figure 3). About a third of patients are unmarried, while the rest of the population is married (Figure 4). Providing various support options

depending on patients marital status could be beneficial, offering a family-based coverage plan in addition to more individualized support.

To better evaluate the success of resource allocation to patients, distributions of procedure costs and procedure types were created. The most common procedure type is lab tests (Figure 5), which are important for preventative screening, potentially lowering the healthcare burden of NCDs, as discussed in the literature review. Evaluation of procedure costs show that financial support for patients is necessary. Most procedure costs are below 2500 PKR (Figure 7), but with 52% of patients making 2500 PKR or less a month (Figure 2), the cost of healthcare would create a serious financial burden. Looking at the average procedure cost by type, lab tests cost 2,463.31 PKR on average (Table 11), further displaying the financial burden that would be imposed on patients if they paid for procedures themselves. Lab tests also have the highest average age of patients, reflecting a need for preventative and diagnostic services (Figure 10). Thus, it is shown that the DPCA is providing much needed financial support for lower income patients, allowing them to take preventative measures against disease.

Understanding how income varies with factors such as marital status and occupation is key for identifying groups that might need greater financial support. In an evaluation of the effect of marital status on income, both married and unmarried groups show similar median incomes, but there are a few high-earning outliers in married group, potentially indicating dual-income households or shared financial resources (Figure 11). Evaluating how income varies with occupation, there are considerable variations in median income (Figure 12). Domestic workers and students display lower median incomes, providing a potential area of outreach for the foundation. Additional income inequality is seen among groups like labor, guard work, and skilled work, with a wide variation of incomes within the occupations themselves.

While the analysis of the data in this database has already generated valuable insights on the success of the foundation and areas for potential improvement, this project was limited by the quantity of data, with only the information from 124 patients. As the DPCA continues serving patients, more data will be added to the dataset and additional analyses can be performed, allowing for continuous monitoring of success in reducing the financial burden of low-income patients. There was also a significant need for data cleaning due to unstandardized manual data entry. This could easily be resolved by reformatting the Google Form to only allow preset answers or training DPCA employees on data entry.

6 Conclusion

The creation of the database and subsequent data visualizations has provided important insights on current resource allocation of the foundation and has highlighted potential areas of outreach for certain groups. With lab tests being the most common procedure type, utilizing funds to support preventative care and diagnostic services is a key initiative to prevent the future healthcare burden of NCDs. Further, highlighting the disparities in income by occupation could lead to outreach programs for low-income groups to reduce the financial burden of healthcare they may face. Additional outreach programs could also be created for those of different communities, as an evaluation of patient religions show that more than 80% of patients utilizing the foundation's services are Sunni.

Additional future work for this project could also include a re-evaluation of the amount of aid allocated to each patient depending on factors such as income, monthly budgets, and patient eligibility, potentially allowing the DPCA to serve a larger quantity of patients. An initial iteration of this has been created and can be seen in Appendix A.2. It would also be possible to easily adapt the framework created for database creation, data cleaning and preprocessing, and data visualization creation for similar healthcare charity foundations, broadening the impact of this project by increasing access to healthcare for as many patients as possible around the globe.

7 References

References

- [1] Hafeez, Assad et al., *The state of health in Pakistan and its provinces and territories, 1990–2019: a systematic analysis for the Global Burden of Disease Study 2019*, The Lancet Global Health, Volume 11, Issue 2, e229 - e243, 2023.
- [2] World Health Organization, *Communicable and Noncommunicable Diseases and Mental Health*, Available: <https://www.who.int/our-work/communicable-and-noncommunicable-diseases-and-mental-health>, Accessed: 16-Nov-2024.
- [3] Dawn, The state of public health in Pakistan, Dawn, Available: <https://www.dawn.com/news/1461789>, Accessed: 25-Nov-2024.

A Appendix A: Code

A.1 Data Cleaning and Preprocessing

```

1 # import libraries
2 import numpy as np
3 import pandas as pd
4 import sqlite3
5 import re
6 import matplotlib.pyplot as plt
7 import seaborn as sns
8 from google.colab import files
9 from datetime import datetime
10 import os
11
12 # import database
13 def ReadDB(table_name):
14     conn = sqlite3.connect('/content/DPCA.db')
15
16     query = f'SELECT * FROM {table_name}'
17     df = pd.read_sql(query, conn)
18
19     conn.close()
20
21     return df

```



```

22
23 df_patient = ReadDB('Patient')
24 df_procedure = ReadDB('Procedure')
25 df_patient_financials = ReadDB('Patient_Financials')
26 df_financial_aid = ReadDB('Financial_Aid')
27 df_zakat_eligibility = ReadDB('Zakat_Eligibility')
28
29 # combine dataframes
30 df_all = pd.concat([df_patient, df_procedure, df_patient_financials,
31                    df_financial_aid, df_zakat_eligibility], axis=1)
32
33 # check for row duplicates
34 df_all.duplicated().sum()
35
36 # check for column duplicates
37 transposed_df = df_all.transpose()           # transpose dataframe
38 transposed_df.duplicated().sum()             # find column
39                                           duplicates
40
41 # drop column duplicates (no row duplicates)
42 transposed_df.drop_duplicates(inplace = True)
43 transposed_df.duplicated().sum()
44
45 # transpose dataframe to return to original dataframe with no
46 # duplicates
47 df_all = transposed_df.transpose()
48
49 # Remove duplicate headers and set proper data types
50 df_all = df_all.iloc[1:] # Remove duplicate header row
51
52 # Clean Patient Data
53 df_all.loc[:, 'Age'] = pd.to_numeric(df_all['Age'], errors='coerce')
54 df_all.loc[:, 'Patient_ID'] = pd.to_numeric(df_all['Patient_ID'],
55                                             errors='coerce')
56
57 # Clean Procedure Data
58 df_all.loc[:, 'Procedure_ID'] = pd.to_numeric(df_all['Procedure_ID'],
59                                             errors='coerce')
60 df_all.loc[:, 'Patient_ID'] = pd.to_numeric(df_all['Patient_ID'],
61                                             errors='coerce')
62 df_all.loc[:, 'Procedure_Cost'] = pd.to_numeric(df_all['Procedure_Cost'],
63                                             errors='coerce')
64 df_all.loc[:, 'Amount_Paid_DPCA'] = pd.to_numeric(df_all['
65     Amount_Paid_DPCA'], errors='coerce')
66
67 # Clean Patient Financials Data
68 df_all.loc[:, 'Financial_Info_ID'] = pd.to_numeric(df_all['
69     Financial_Info_ID'], errors='coerce')
70 df_all.loc[:, 'Patient_ID'] = pd.to_numeric(df_all['Patient_ID'],
71                                             errors='coerce')
72 df_all.loc[:, 'Dependent_Members'] = pd.to_numeric(df_all['
73     Dependent_Members'], errors='coerce')
74
75 # Clean Financial Aid Data
76 df_all.loc[:, 'Financial_Aid_ID'] = pd.to_numeric(df_all['
77     Financial_Aid_ID'], errors='coerce')
78 df_all.loc[:, 'Patient_ID'] = pd.to_numeric(df_all['Patient_ID'],
79                                             errors='coerce')

```

```

67 df_all.loc[:, 'Aid_Date'] = pd.to_datetime(df_all['Aid_Date'], errors='
    coerce')
68
69 # map the data - change text to numbers
70 df_all['Religion'] = df_all['Religion'].map({'Muslim (Shia)':0, 'Muslim
    (Sunni)':1, 'Muslim (Other)':2, 'Non-Muslim':3})
71 df_all['Syed_NonSyed'] = df_all['Syed_NonSyed'].map({'Syed':0, 'Non-
    Syed':1})
72 df_all['Marriage_Status'] = df_all['Marriage_Status'].map({'Married':0,
    'Unmarried':1})
73 df_all['Income'] = df_all['Income'].map({'No Income': 0, 'No income':
    0, '1-10000/month': 5000, '10001-50000/month': 30000, '50001-100000/
    month': 75000, '100001-150000/month': 125000}).fillna(df_all['Income
    ']) # mapped to middle of income range
74
75 # return # of each column's none values
76 df_all.replace('', pd.NA, inplace=True) # Replace empty strings with
    NaN (if needed)
77 df_all.isnull().sum()
78
79 #Transforming data/ filling or dropping null values
80 df_all['Procedure_Cost'] = df_all['Procedure_Cost'].fillna('N.A')
81 df_all['Aid_Type'] = df_all['Aid_Type'].fillna('N.A')
82 df_all['Eligibility_Criteria'] = df_all['Eligibility_Criteria'].fillna(
    'N.A')
83
84 df_all['Religion'] = df_all['Religion'].fillna('Unknown')
85 df_all['Syed_NonSyed'] = df_all['Syed_NonSyed'].fillna('Unknown')
86 df_all['Marriage_Status'] = df_all['Marriage_Status'].fillna('Unknown')
87 df_all['Contact_No'] = df_all['Contact_No'].fillna('Unknown')
88 df_all['Address'] = df_all['Address'].fillna('Unknown')
89 df_all['Occupation'] = df_all['Occupation'].fillna('Unknown')
90 df_all['Procedure_Date'] = df_all['Procedure_Date'].fillna('Unknown')
91
92 df_all['Dependent_Members'] = df_all['Dependent_Members'].fillna(0)
93 df_all['Properties_Owned'] = df_all['Properties_Owned'].fillna(0)
94 df_all['Amount_Paid_DPCA'] = df_all['Amount_Paid_DPCA'].fillna(0)
95 df_all['Financial_Support_Method'] = df_all['Financial_Support_Method']
    .fillna('Unknown')
96 df_all['Is_Eligible'] = df_all['Is_Eligible'].fillna('No')
97
98 df_all['Patient_ID'] = df_all['Patient_ID'].fillna('Unknown') # or use
    a default ID if needed
99 df_all['Patient_Name'] = df_all['Patient_Name'].fillna('Unknown')
100 df_all['Age'] = df_all['Age'].fillna(0) # assuming age 0 for missing
    data
101 df_all['CNIC_No'] = df_all['CNIC_No'].fillna('Unknown')
102 df_all['Procedure_Type'] = df_all['Procedure_Type'].fillna('Unknown')
103 df_all['MR_number'] = df_all['MR_number'].fillna('Unknown')
104 df_all['Aid_Date'] = df_all['Aid_Date'].fillna('Unknown')
105
106 df_all.isnull().sum()
107
108 # Check non-numeric entries
109 def check_non_numeric(column):
110     return df_all[column][pd.to_numeric(df_all[column], errors='coerce'
    ).isna()]
111

```

```

112 print("Non-numeric entries in Amount_Paid_DPCA:", check_non_numeric('
    Amount_Paid_DPCA'))
113 print("Non-numeric entries in Procedure_Cost:", check_non_numeric('
    Procedure_Cost'))
114 print("Non-numeric entries in Income:", check_non_numeric('Income'))
115
116 # Function to clean the 'Income' and 'Procedure_Cost' columns
117 def clean_numeric_column(column):
118     # Replace non-numeric values with NaN to handle them separately
119     df_all[column] = pd.to_numeric(df_all[column], errors='coerce')
120
121     # Replace specific strings with NaN (or you could replace with 0 if
    you prefer)
122     df_all[column] = df_all[column].replace(['No income', 'N.A'], np.
    nan)
123
124     # Handle range values by taking the midpoint of the range
125     def convert_range(value):
126         if isinstance(value, str) and '-' in value:
127             parts = value.split('-')
128             try:
129                 # Convert each part of the range to an integer and take
    the average
130                 return (int(parts[0]) + int(parts[1].split('/')[0])) /
    2
131             except ValueError:
132                 return np.nan
133             return value
134
135     # Apply range conversion function
136     df_all[column] = df_all[column].apply(convert_range)
137
138     # Convert remaining non-numeric entries to NaN
139     df_all[column] = pd.to_numeric(df_all[column], errors='coerce')
140
141     # Fill NaN with 0 or another default value if needed
142     df_all[column] = df_all[column].fillna(0)
143
144 # Clean 'Income' and 'Procedure_Cost' columns
145 clean_numeric_column('Income')
146 clean_numeric_column('Procedure_Cost')
147
148 # Verify that non-numeric values are handled
149 print("Non-numeric entries in Amount_Paid_DPCA:", check_non_numeric('
    Amount_Paid_DPCA'))
150 print("Non-numeric entries in Procedure_Cost:", check_non_numeric('
    Procedure_Cost'))
151 print("Non-numeric entries in Income:", check_non_numeric('Income'))
152
153 # Function to clean Financial column
154 def clean_financial_column(value):
155     if pd.isna(value): # If the value is NaN, keep it as NaN
156         return value
157
158     # Remove any non-numeric characters except for decimal points
159     value = re.sub(r'[^\\d.]', '', str(value))
160
161     # Handle cases where value is empty after removing non-numeric

```

```

characters
162     if value == '':
163         return pd.NA
164
165     # Convert the cleaned value to a float
166     try:
167         return float(value)
168     except ValueError:
169         return pd.NA # Return NaN if conversion fails
170
171 df_all['Amount_Paid_DPCA'] = df_all['Amount_Paid_DPCA'].apply(
172     clean_financial_column)
173 df_all['Procedure_Cost'] = df_all['Procedure_Cost'].apply(
174     clean_financial_column)
175 df_all['Income'] = df_all['Income'].apply(clean_financial_column)
176
177 # Function to clean Income column
178 def clean_income(value):
179     if pd.isna(value):
180         return value
181
182     # Handle "No income" case
183     if "no income" in str(value).lower():
184         return 0
185
186     # Handle ranges (e.g., "1-10000/month" -> take the midpoint)
187     range_match = re.match(r'(\d+)\s+[\s/]+\s+(\d+)', str(value))
188     if range_match:
189         lower = int(range_match.group(1))
190         upper = int(range_match.group(2))
191         return (lower + upper) / 2 # Return the midpoint of the range
192
193     # Clean and convert as a regular financial value
194     return clean_financial_column(value)
195
196 # Apply to the Income column
197 df_all['Income'] = df_all['Income'].apply(clean_income)
198
199 # Check data types
200 print(df_all[['Amount_Paid_DPCA', 'Procedure_Cost', 'Income']].dtypes)
201
202 # Display any remaining non-numeric entries
203 print("Remaining non-numeric in Amount_Paid_DPCA:", df_all['
204     Amount_Paid_DPCA'][pd.to_numeric(df_all['Amount_Paid_DPCA'], errors=
205     'coerce').isna()])
206 print("Remaining non-numeric in Procedure_Cost:", df_all['
207     Procedure_Cost'][pd.to_numeric(df_all['Procedure_Cost'], errors=
208     'coerce').isna()])
209 print("Remaining non-numeric in Income:", df_all['Income'][pd.
210     to_numeric(df_all['Income'], errors='coerce').isna()])
211
212 # Convert columns to numeric, coercing any remaining non-numeric values
213 # to NaN
214 df_all['Amount_Paid_DPCA'] = pd.to_numeric(df_all['Amount_Paid_DPCA'],
215     errors='coerce')
216 df_all['Procedure_Cost'] = pd.to_numeric(df_all['Procedure_Cost'],
217     errors='coerce')
218 df_all['Income'] = pd.to_numeric(df_all['Income'], errors='coerce')

```

```
209
210 # Check data types again
211 print(df_all[['Amount_Paid_DPCA', 'Procedure_Cost', 'Income']].dtypes)
212
213 # Validate financial columns
214 invalid_amount_paid = df_all[df_all['Amount_Paid_DPCA'] < 0]
215 invalid_procedure_cost = df_all[df_all['Procedure_Cost'] < 0]
216 invalid_income = df_all[df_all['Income'] < 0]
217
218 print("Invalid Amount_Paid_DPCA:", invalid_amount_paid)
219 print("Invalid Procedure_Cost:", invalid_procedure_cost)
220 print("Invalid Income:", invalid_income)
221
222 # Further clean the 'Age' column to ensure all values are numeric
223
224 def clean_age(age):
225     if isinstance(age, str):
226         # Extract numeric part
227         numbers = re.findall(r'\d+\.?\d*', age)
228         if numbers:
229             return float(numbers[0])
230     try:
231         return float(age)
232     except ValueError:
233         return np.nan # Return NaN for any non-convertible values
234
235 # Apply the cleaning function
236 df_all['Age'] = df_all['Age'].apply(clean_age)
237
238 # Drop any rows with NaN values in 'Age' or 'Procedure_Cost' to ensure
    clean calculations
239 df_all.dropna(subset=['Age', 'Procedure_Cost'], inplace=True)
240
241 # Clean Occupation column
242 def clean_occupation(value):
243     # Convert the value to a string for consistent handling
244     value_str = str(value).lower()
245
246     # Handle "Dow" case
247     if "dow" in value_str:
248         return "Dow Employee"
249
250     # Handle "Driver" case
251     if "rickshaw" in value_str or "driver" in value_str or "rikshaw" in
    value_str:
252         return "Driver"
253
254     # Handle "Guard" case
255     if "guard" in value_str or "gaurd" in value_str or "security" in
    value_str:
256         return "Guard"
257
258     # Handle "Domestic" case
259     if "house" in value_str or "maid" in value_str or "cook" in
    value_str or "clean" in value_str:
260         return "Domestic"
261
262     # Handle "Labor" case
```

```

263     if any(term in value_str for term in ["labour", "labor", "mazdoor",
264         "sweep", "janitor", "cement", "block maker", "trash", "vendor"]):
265         return "Labor"
266
267     # Handle "Dependent" case
268     if any(term in value_str for term in ["father", "husband", "brother",
269         "son", "wife"]):
270         return "Dependent"
271
272     # Handle "Vendor" case
273     if any(term in value_str for term in ["sell", "vendor", "stall"]):
274         return "Vendor"
275
276     # Handle "Student" case
277     if "student" in value_str:
278         return "Student"
279
280     # Handle "Unemployed" case
281     if any(term in value_str for term in ["unemployed", "not", "none",
282         "jobless", "does nothing", "-", "N/A"]):
283         return "Unemployed"
284
285     # Handle "Religious" case
286     if "imam" in value_str:
287         return "Religious"
288
289     # Handle "Skilled Work" case
290     if any(term in value_str for term in ["sewing", "tailor", "packager",
291         "chai hotel", "operator", "machine"]):
292         return "Skilled Work"
293
294     # Default case
295     return "Other"
296
297 # Apply to the Income column
298 df_all['Occupation'] = df_all['Occupation'].apply(clean_occupation)
299
300 # Clean Procedure Type column
301 def clean_procedure(value):
302     # Convert the value to a string for consistent handling
303     value_str = str(value).lower()
304
305     # Handle "Lab test" case
306     if "lab test" in value_str:
307         return "Lab test"
308
309     # Handle "Surgery" case
310     if "surgery" in value_str:
311         return "Surgery"
312
313     # Handle "Medicine" case
314     if "med" in value_str:
315         return "Medicine"
316
317     # Handle "Hospital care" case
318     if "hospital" in value_str:
319         return "Hospital Care"

```

```

317     # Handle "Radiology" case
318     if "radiology" in value_str:
319         return "Radiology"
320
321     # Default case: return the original value
322     return value
323
324 # Apply to the Income column
325 df_all['Procedure_Type'] = df_all['Procedure_Type'].apply(
326     clean_procedure)
327
328 # Clean Financial Support Method column
329 def clean_financial_support(value):
330     # Convert the value to a string for consistent handling
331     value_str = str(value).lower()
332
333     # Handle "Family" case
334     if any(term in value_str for term in ["husband", "wife", "father",
335     "mother", "brother", "sister", "daughter", "son", "relative", "
336     friend"]):
337         return "Family"
338
339     # Handle "Self Employed" case
340     if "self" in value_str:
341         return "Self Employed"
342
343     # Handle "Welfare" case
344     if "welfare" in value_str:
345         return "Welfare"
346
347     # Handle "Employed" case
348     if any(term in value_str for term in ["job", "work", "sell", "sold"
349     , "work", "employed", "sweeper", "labour", "dipca"]):
350         return "Employed"
351
352     # Handle "Unknown" case
353     if any(term in value_str for term in ["unknown"]):
354         return "Unknown"
355
356     # Handle "None" case
357     if any(term in value_str for term in ["-", "no", "nil", "N/A", "n/a"
358     , "none", "non", ".."]):
359         return "None"
360
361     # Default case: return the original value
362     return value
363
364 # Apply to the Income column
365 df_all['Financial_Support_Method'] = df_all['Financial_Support_Method']
366     ].apply(clean_financial_support)
367
368 # Clean Properties Owned column
369 def clean_properties_owned(value):
370     # Convert the value to a string for consistent handling
371     value_str = str(value).lower()
372
373     # Handle "2" case
374     if any(term in value_str for term in ["and"]):

```

```

369         return "2"
370
371     # Handle "0" case
372     if any(term in value_str for term in ["0", "rent", "no", "-", "has
gold earrings", "kacha"]):
373         return "0"
374
375     # Handle "1" case
376     if any(term in value_str for term in ["1", "home", "house", "jhopri
"]):
377         return "1"
378
379     # Default case: return the original value
380     return value
381
382 # Apply to the Income column
383 df_all['Properties_Owned'] = df_all['Properties_Owned'].apply(
    clean_properties_owned)
384
385 # Print out cleaned and processed database for checking
386 df_all.head(50)

```

Listing 1: Code for Data Cleaning and Preprocessing

A.2 Additional Code for Aid Type and Amount

```

1
2 print("\
3 Top 3 most common occupation-procedure combinations:")
4 # Get the top 3 occupation-procedure combinations
5 flat_crosstab = job_procedure_crosstab.unstack()
6 top_combinations = flat_crosstab.sort_values(ascending=False)[:3]
7 for (occupation, procedure), count in top_combinations.items():
8     print(f"{occupation} - {procedure}: {count} cases")
9
10 # Create aid_type column based on eligibility criteria
11 def determine_aid_type(criteria):
12     if isinstance(criteria, str):
13         criteria = criteria.lower()
14         # Check for all required conditions for Zakat
15         muslim_condition = 'muslim' in criteria
16         non_syed_condition = 'non-syed' in criteria or 'non syed' in
criteria
17         wealth_condition = '52.5 tola' in criteria or 'does not own' in
criteria
18
19         if muslim_condition and non_syed_condition and wealth_condition
:
20             return 'Zakat Funds'
21         return 'General Funds'
22
23 # Apply the function to create aid_type column
24 df_all['Aid_Type'] = df_all['Eligibility_Criteria'].apply(
    determine_aid_type)
25
26 # Create is_eligible column based on aid_type

```



```

27 df_all['Is_Eligible'] = df_all['Aid_Type'].apply(lambda x: 'Yes' if x
    == 'Zakat Funds' else 'No')
28
29 # Display sample results
30 print("Sample of Eligibility Criteria, Aid Type, and Eligibility:")
31 print(df_all[['Eligibility_Criteria', 'Aid_Type', 'Is_Eligible']].head
    (10))
32
33 # Display distribution of aid types
34 print("Distribution of Aid Types:")
35 print(df_all['Aid_Type'].value_counts())
36
37 # Display eligibility counts
38 print("Distribution of Eligibility:")
39 print(df_all['Is_Eligible'].value_counts())
40
41 # Calculation of Aid Amounts based on eligibility
42
43 # monthly_budget = 168000 # Rupees
44
45 # # Calculate total aid by type
46 # aid_summary = df_all.groupby('Aid_Type').agg({
47 #     'Aid_Amount': ['sum', 'count', 'mean']
48 # }).round(2)
49
50 # aid_summary.columns = ['Total_Amount', 'Number_of_Patients', '
    Average_Amount']
51 # aid_summary = aid_summary.reset_index()
52
53 # print("Aid Allocation Summary:")
54 # print(aid_summary)
55
56 # # Calculate percentages
57 # total_aid = aid_summary['Total_Amount'].sum()
58 # aid_summary['Percentage_of_Total_Aid'] = (aid_summary['Total_Amount']
    / total_aid * 100).round(2)
59
60 # print("\
61 # Detailed Aid Distribution:")
62 # print(df_all.head())

```

Listing 2: Additional Code for Filling in Aid Type and Amount based on Eligibility

A.3 Data Visualization

```

1
2 # Undo mapped data for charts
3 df_all['Religion'] = df_all['Religion'].map({0: 'Muslim (Shia)', 1: '
    Muslim (Sunni)'})
4 df_all['Syed_NonSyed'] = df_all['Syed_NonSyed'].map({0: 'Syed', 1: 'Non-
    Syed'})
5 df_all['Marriage_Status'] = df_all['Marriage_Status'].map({0: 'Married',
    1: 'Unmarried'})
6
7 # Visualizations
8 sns.set_style('whitegrid')
9

```

```
10 # Histogram for Procedure Cost
11 plt.figure(figsize=(10, 6))
12 sns.histplot(data=df_all, x='Procedure_Cost', bins=30)
13 plt.title('Procedure Cost Distribution')
14 plt.xlabel('Cost (PKR)')
15 plt.ylabel('Count')
16 plt.tight_layout()
17 plt.show()
18
19 # Histogram for Procedure Cost < 20K PKR
20 less20 = df_all[df_all['Procedure_Cost'] <= 20000]
21
22 plt.figure(figsize=(10, 6))
23 sns.histplot(data=less20, x='Procedure_Cost', bins=30)
24 plt.title('Procedure Cost Distribution for Procedures < 20,000 PKR')
25 plt.xlabel('Cost (PKR)')
26 plt.ylabel('Count')
27 plt.tight_layout()
28 plt.show()
29
30 # Histogram for Procedure Cost > 10K PKR
31 more20 = df_all[df_all['Procedure_Cost'] > 20000]
32
33 plt.figure(figsize=(10, 6))
34 sns.histplot(data=more20, x='Procedure_Cost', bins=30)
35 plt.title('Procedure Cost Distribution for Procedures > 20,000 PKR')
36 plt.xlabel('Cost (PKR)')
37 plt.ylabel('Count')
38 plt.tight_layout()
39 plt.show()
40
41 # Procedure Types Distribution
42 plt.figure(figsize=(12, 6))
43 df_all['Procedure_Type'].value_counts().plot(kind='bar')
44 plt.title('Distribution of Procedure Types')
45 plt.xlabel('Procedure Type')
46 plt.ylabel('Count')
47 plt.xticks(rotation=45)
48 plt.tight_layout()
49 plt.show()
50
51 # Average Procedure Cost by Type
52 plt.figure(figsize=(12, 6))
53 avg_cost = df_all.groupby('Procedure_Type')['Procedure_Cost'].mean()
54 sns.barplot(x=avg_cost.index, y=avg_cost.values)
55 plt.title('Average Procedure Cost by Type')
56 plt.xticks(rotation=45)
57 plt.tight_layout()
58 #plt.savefig('avg_procedure_cost.png')
59 plt.show()
60
61 # Average Age per Procedure
62 plt.figure(figsize=(12, 6))
63 avg_age = df_all.groupby('Procedure_Type')['Age'].mean().sort_values(
    ascending=False)
64 sns.barplot(x=avg_age.index, y=avg_age.values)
65 plt.title('Average Age per Procedure Type')
66 plt.xticks(rotation=45, ha='right')
```

```
67 plt.ylabel('Average Age')
68 plt.tight_layout()
69 #plt.savefig('avg_age_per_procedure.png')
70 plt.show()
71
72 # Religion Distribution
73 plt.figure(figsize=(6, 6))
74 df_all['Religion'].value_counts().plot(kind='pie', autopct='%1.1f%%')
75 plt.title('Distribution of Patients by Religion')
76 plt.axis('equal')
77 plt.show()
78
79
80 # Income Distribution by Marriage Status
81 plt.figure(figsize=(10, 6))
82 sns.boxplot(data=df_all, x='Marriage_Status', y='Income')
83 plt.title('Income Distribution by Marriage Status')
84 plt.xticks(rotation=45)
85 plt.show()
86
87 # Pie chart for income distribution
88 plt.figure(figsize=(6, 6))
89 income_counts = df_all['Income'].value_counts()
90 plt.pie(income_counts, labels=income_counts.index, autopct='%1.1f%%',
91         startangle=140)
92 plt.title('Income Distribution in PKR')
93 plt.axis('equal')
94 plt.tight_layout()
95 plt.show()
96
97 # Pie chart for marriage status
98 plt.figure(figsize=(6, 6))
99 marriage_counts = df_all['Marriage_Status'].value_counts()
100 plt.pie(marriage_counts, labels=marriage_counts.index, autopct='%1.1f%%',
101         startangle=90)
102 plt.title('Marriage Status Distribution')
103 plt.axis('equal')
104 plt.tight_layout()
105 plt.show()
106
107 # Histogram for age distribution
108 plt.figure(figsize=(10, 6))
109 sns.histplot(data=df_all, x='Age', bins=20)
110 plt.title('Age Distribution')
111 plt.xlabel('Age')
112 plt.ylabel('Count')
113 plt.tight_layout()
114 plt.show()
115
116 # Box plots for income by occupation
117 plt.figure(figsize=(12, 6))
118 sns.boxplot(x='Occupation', y='Income', data=df_all)
119 plt.xticks(rotation=45, ha='right')
120 plt.title('Income Distribution by Occupation')
121 plt.tight_layout()
122 plt.show()
123
124 numeric_df = df_all.select_dtypes(include=[np.number])
```

```

123
124 # Calculate the correlation matrix for numeric columns
125 correlation_matrix = numeric_df.corr()
126
127 # Create a heatmap for the correlation matrix
128 plt.figure(figsize=(12, 8))
129 sns.heatmap(correlation_matrix, annot=True, cmap='crest', fmt='.2f')
130 plt.title('Correlation Matrix Heatmap')
131 plt.tight_layout()
132 plt.show()
133
134 print("Correlation matrix heatmap created")
135
136 # Analyze job impact on procedure type
137 job_procedure_crosstab = pd.crosstab(df_all['Occupation'], df_all['
    Procedure_Type'])
138
139 # Create a heatmap of the job-procedure relationship
140 plt.figure(figsize=(12, 8))
141 sns.heatmap(job_procedure_crosstab, annot=True, fmt='d', cmap='YlOrRd')
142 plt.title('Procedure Types by Occupation')
143 plt.xlabel('Procedure Type')
144 plt.ylabel('Occupation')
145 plt.xticks(rotation=45, ha='right')
146 plt.tight_layout()
147 plt.show()
148
149 # Calculate percentage distribution of procedures within each
    occupation
150 procedure_pct = job_procedure_crosstab.div(job_procedure_crosstab.sum(
    axis=1), axis=0) * 100
151
152 # Create a heatmap of the percentage distribution
153 plt.figure(figsize=(12, 8))
154 sns.heatmap(procedure_pct, annot=True, fmt='.1f', cmap='YlOrRd')
155 plt.title('Procedure Types Distribution by Occupation (%)')
156 plt.xlabel('Procedure Type')
157 plt.ylabel('Occupation')
158 plt.xticks(rotation=45, ha='right')
159 plt.tight_layout()
160 plt.show()

```

Listing 3: Code for Data Visualization

A.4 Exporting Cleaned Database

```

1
2 # Split large dataframe into individual dataframes for tables
3 clean_df_patient = df_all[['Patient_ID', 'Patient_Name', 'Age', 'Religion'
    , 'Syed_NonSyed', 'Marriage_Status', 'CNIC_No', 'Contact_No', 'Address']]
4 clean_df_procedure = df_all[['Procedure_ID', 'Procedure_Type', '
    Procedure_Date', 'MR_number', 'Amount_Paid_DPCA', 'Procedure_Cost', '
    Patient_ID']]
5 clean_df_patient_financials = df_all[['Financial_Info_ID', 'Occupation',
    'Income', 'Dependent_Members', 'Financial_Support_Method', '
    Properties_Owned', 'Patient_ID']]

```

```

6 clean_df_financial_aid = df_all[['Financial_Aid_ID','Aid_Date','
    Aid_Type','Aid_Amount','Financial_Info_ID','Zakat_Eligibility_ID','
    Patient_ID']]
7 clean_df_zakat_eligibility = df_all[['Zakat_Eligibility_ID','
    Is_Eligible','Eligibility_Criteria','Patient_ID','Financial_Aid_ID'
    ]]
8
9 # Connect to SQLite
10 conn = sqlite3.connect("Cleaned_DPCA.db")
11
12 # Export dataframes to SQLite tables
13 clean_df_patient.to_sql(name="Patient", con = conn, if_exists="replace"
    , index=False)
14 clean_df_procedure.to_sql(name="Procedure", con = conn, if_exists="
    replace", index=False)
15 clean_df_patient_financials.to_sql(name="Patient_Financials", con =
    conn, if_exists="replace", index=False)
16 clean_df_financial_aid.to_sql(name="Financial_Aid", con = conn,
    if_exists="replace", index=False)
17 clean_df_zakat_eligibility.to_sql(name="Zakat_Eligibility", con = conn,
    if_exists="replace", index=False)
18
19 # Define relationships via primary and foreign keys
20 conn.execute("PRAGMA foreign_keys = ON;")
21
22 # Patient table
23 conn.execute("""
24 CREATE TABLE IF NOT EXISTS Patient (
25     Patient_ID INT PRIMARY KEY,
26     Patient_Name VARCHAR(255) NOT NULL,
27     Age INT, Religion VARCHAR(255),
28     Syed_NonSyed VARCHAR(255),
29     Marriage_Status VARCHAR(255),
30     CNIC_No VARCHAR(255),
31     Contact_No VARCHAR(255),
32     Address VARCHAR(255)
33 );
34 """)
35
36 clean_df_patient.to_sql(name="Patient", con=conn, if_exists="replace",
    index=False)      # insert data into table
37
38 # Procedure table
39 conn.execute("""
40 CREATE TABLE IF NOT EXISTS Procedure (
41     Procedure_ID INT PRIMARY KEY,
42     Procedure_Type VARCHAR(255) NOT NULL,
43     Procedure_Date DATE,
44     MR_number VARCHAR(255),
45     Amount_Paid_DPCA INT,
46     Procedure_Cost INT,
47     Patient_ID INT,
48     FOREIGN KEY (Patient_ID) REFERENCES Patient(Patient_ID)
49 );
50 """)
51
52 clean_df_procedure.to_sql(name="Procedure", con=conn, if_exists="
    replace", index=False)      # insert data into table

```

```

53
54 # Patient Financials table
55 conn.execute("""
56 CREATE TABLE IF NOT EXISTS Patient_Financials (
57     Financial_Info_ID INT PRIMARY KEY,
58     Occupation VARCHAR(255),
59     Income INT,
60     Dependent_Members INT,
61     Financial_Support_Method VARCHAR(255),
62     Properties_Owned INT,
63     Patient_ID INT,
64     FOREIGN KEY (Patient_ID) REFERENCES Patient(Patient_ID)
65 );
66 """)
67
68 clean_df_patient_financials.to_sql(name="Patient_Financials", con=conn,
69     if_exists="replace", index=False)          # insert data into table
69
70 # Financial Aid table
71 conn.execute("""
72 CREATE TABLE IF NOT EXISTS Financial_Aid (
73     Financial_Aid_ID INT PRIMARY KEY,
74     Aid_Date DATE,
75     Aid_Type VARCHAR(255),
76     Aid_Amount INT,
77     Financial_Info_ID INT,
78     Zakat_Eligibility_ID INT,
79     Patient_ID INT,
80     FOREIGN KEY (Financial_Info_ID) REFERENCES Financial_Info(
81     Financial_Info_ID),
82     FOREIGN KEY (Zakat_Eligibility_ID) REFERENCES Zakat_Eligibility(
83     Zakat_Eligibility_ID),
84     FOREIGN KEY (Patient_ID) REFERENCES Patient(Patient_ID)
85 );
86 """)
87
88 clean_df_financial_aid.to_sql(name="Financial_Aid", con=conn, if_exists
89     ="replace", index=False)          # insert data into table
89
90 # Zakat Eligibility table
91 conn.execute("""
92 CREATE TABLE IF NOT EXISTS Zakat_Eligibility (
93     Zakat_Eligibility_ID INT PRIMARY KEY,
94     Is_Eligible BOOLEAN,
95     Eligibility_Criteria VARCHAR(255),
96     Patient_ID INT, Financial_Aid_ID INT,
97     FOREIGN KEY (Patient_ID) REFERENCES Patient(Patient_ID),
98     FOREIGN KEY (Financial_Aid_ID) REFERENCES Financial_Aid(
99     Financial_Aid_ID)
100 );
101 """)
102
103 clean_df_zakat_eligibility.to_sql(name="Zakat_Eligibility", con=conn,
104     if_exists="replace", index=False)          # insert data into table
104
105 # Export database out of colab
106 db_file = '/content/Cleaned_DPCA.db'

```

```

105 #files.download(db_file) # download the file (commented out to prevent
    unintended downloads)
106
107 conn.close()

```

Listing 4: Exporting Cleaned Database

A.5 SQL Queries and Reports

```

1
2 # Create connection to new database
3 conn = sqlite3.connect("Cleaned_DPCA.db")
4
5 if not os.path.exists('reports'):
6     os.makedirs('reports')
7
8 # Define SQL queries
9 queries = {
10     # Procedure History Report
11     'procedure_history': """
12     SELECT
13         p.Patient_ID,
14         COALESCE(p.Patient_Name, 'Unknown') as Patient_Name,
15         COALESCE(proc.Procedure_Type, 'Unknown') as Procedure_Type,
16         COALESCE(proc.Procedure_Date, 'No Date') as Procedure_Date,
17         COALESCE(proc.Procedure_Cost, 0) as Procedure_Cost,
18         COALESCE(proc.Amount_Paid_DPCA, 0) as Amount_Paid_DPCA,
19         COALESCE(proc.Procedure_Cost, 0) - COALESCE(proc.
Amount_Paid_DPCA, 0) as Patient_Contribution,
20         COALESCE(proc.MR_number, 'Not Provided') as MR_number
21     FROM
22         Patient p
23     JOIN
24         Procedure proc ON p.Patient_ID = proc.Patient_ID
25     WHERE
26         proc.Procedure_Cost IS NOT NULL
27         AND proc.Procedure_Type IS NOT NULL
28         AND proc.Procedure_Type != 'Unknown'
29     ORDER BY
30         p.Patient_ID, proc.Procedure_Date
31     """,
32
33     # Financial Aid Report
34     'financial_aid': """
35     SELECT
36         p.Patient_ID,
37         COALESCE(p.Patient_Name, 'Unknown') as Patient_Name,
38         fa.Aid_Date,
39         COALESCE(fa.Aid_Type, 'Not Specified') as Aid_Type,
40         COALESCE(proc.Amount_Paid_DPCA, 0) as Aid_Amount,
41         COALESCE(pf.Income, 0) as Monthly_Income,
42         CASE
43             WHEN COALESCE(pf.Income, 0) > 0
44             THEN ROUND((COALESCE(proc.Amount_Paid_DPCA, 0) * 100.0 / pf
.Income), 2)
45             ELSE 0
46         END as Aid_Percentage_of_Income,

```

```

47         COALESCE(proc.Procedure_Type, 'Not Specified') as
Procedure_Type
48     FROM
49         Patient p
50     JOIN
51         Financial_Aid fa ON p.Patient_ID = fa.Patient_ID
52     LEFT JOIN
53         Patient_Financials pf ON p.Patient_ID = pf.Patient_ID
54     LEFT JOIN
55         Procedure proc ON p.Patient_ID = proc.Patient_ID
56     WHERE
57         fa.Aid_Date IS NOT NULL
58         AND proc.Amount_Paid_DPCA IS NOT NULL
59     ORDER BY
60         p.Patient_ID, fa.Aid_Date
61     """,
62
63     # Zakat Eligibility Report
64     'zakat_eligibility': ""
65     SELECT
66         p.Patient_ID,
67         COALESCE(p.Patient_Name, 'Unknown') as Patient_Name,
68         COALESCE(p.Religion, 'Not Specified') as Religion,
69         COALESCE(p.Syed_NonSyed, 'Not Specified') as Syed_NonSyed,
70         ze.Is_Eligible,
71         COALESCE(ze.Eligibility_Criteria, 'Not Specified') as
Eligibility_Criteria,
72         COALESCE(pf.Income, 0) as Monthly_Income,
73         COALESCE(pf.Properties_Owned, 0) as Properties_Owned,
74         fa.Aid_Type
75     FROM
76         Patient p
77     JOIN
78         Zakat_Eligibility ze ON p.Patient_ID = ze.Patient_ID
79     LEFT JOIN
80         Patient_Financials pf ON p.Patient_ID = pf.Patient_ID
81     LEFT JOIN
82         Financial_Aid fa ON p.Patient_ID = fa.Patient_ID
83     WHERE
84         (fa.Aid_Type = 'Zakat Funds' OR ze.Is_Eligible = 1
85         OR ze.Is_Eligible = '1')
86         AND ze.Eligibility_Criteria IS NOT NULL
87         AND ze.Eligibility_Criteria != 'N.A'
88     ORDER BY
89         p.Patient_ID;
90     """,
91
92     # Dependency and Financial Burden Report
93     'dependency_financial': ""
94     SELECT
95         p.Patient_ID,
96         COALESCE(p.Patient_Name, 'Unknown') as Patient_Name,
97         COALESCE(pf.Income, 0) as Monthly_Income,
98         COALESCE(pf.Dependent_Members, 0) as Dependent_Members,
99         CASE
100             WHEN COALESCE(pf.Dependent_Members, 0) > 0
101             THEN ROUND(CAST(COALESCE(pf.Income, 0) AS FLOAT) / pf.
Dependent_Members, 2)

```



```

102         ELSE COALESCE(pf.Income, 0)
103     END as Income_Per_Dependent,
104     COALESCE(pf.Financial_Support_Method, 'Not Specified') as
Financial_Support_Method,
105     COALESCE(pf.Properties_Owned, 0) as Properties_Owned,
106     COUNT(proc.Procedure_ID) as Total_Procedures,
107     COALESCE(SUM(proc.Procedure_Cost), 0) as Total_Medical_Costs
108 FROM
109     Patient p
110 JOIN
111     Patient_Financials pf ON p.Patient_ID = pf.Patient_ID
112 LEFT JOIN
113     Procedure proc ON p.Patient_ID = proc.Patient_ID
114 GROUP BY
115     p.Patient_ID, p.Patient_Name, pf.Income, pf.Dependent_Members,
116     pf.Financial_Support_Method, pf.Properties_Owned
117 HAVING
118     Total_Medical_Costs > 0
119 ORDER BY
120     Income_Per_Dependent
121     "",
122
123 # Occupation-Based Financial Analysis Report
124 'occupation_analysis': ""
125 SELECT
126     COALESCE(pf.Occupation, 'Not Specified') as Occupation,
127     COUNT(DISTINCT p.Patient_ID) as Total_Patients,
128     ROUND(AVG(COALESCE(pf.Income, 0)), 2) as Average_Income,
129     ROUND(AVG(COALESCE(pf.Dependent_Members, 0)), 2) as
Average_Dependents,
130     ROUND(AVG(COALESCE(proc.Amount_Paid_DPCA, 0)), 2) as
Average_Aid_Amount,
131     COUNT(proc.Procedure_ID) as Total_Procedures,
132     ROUND(AVG(COALESCE(proc.Procedure_Cost, 0)), 2) as
Average_Procedure_Cost,
133     ROUND(SUM(COALESCE(proc.Amount_Paid_DPCA, 0)), 2) as
Total_Aid_Given
134 FROM
135     Patient_Financials pf
136 JOIN
137     Patient p ON pf.Patient_ID = p.Patient_ID
138 LEFT JOIN
139     Financial_Aid fa ON p.Patient_ID = fa.Patient_ID
140 LEFT JOIN
141     Procedure proc ON p.Patient_ID = proc.Patient_ID
142 WHERE
143     pf.Occupation IS NOT NULL
144     AND pf.Occupation != ''
145 GROUP BY
146     pf.Occupation
147 HAVING
148     COUNT(DISTINCT p.Patient_ID) > 0
149 ORDER BY
150     Total_Aid_Given DESC
151     "",
152
153 # Patient Demographic Report
154 'patient_demographics': ""

```

```

155 SELECT
156     p.Patient_ID,
157     p.Patient_Name,
158     p.Age,
159     p.Religion,
160     p.Marriage_Status,
161     p.Contact_No,
162     p.Address
163 FROM
164     Patient p
165 ORDER BY
166     p.Patient_ID
167 "",
168
169 # Financial Overview Report
170 'financial_overview': ""
171 SELECT
172     p.Patient_ID,
173     pf.Occupation,
174     pf.Income,
175     pf.Dependent_Members,
176     pf.Properties_Owned,
177     pf.Financial_Support_Method
178 FROM
179     Patient_Financials pf
180 JOIN
181     Patient p ON pf.Patient_ID = p.Patient_ID
182 ORDER BY
183     p.Patient_ID
184 "",
185
186 # Patient Financial Summary
187 'patient_financial_summary': ""
188 SELECT
189     p.Patient_ID,
190     pf.Income,
191     pf.Dependent_Members,
192     pf.Financial_Support_Method,
193     pf.Properties_Owned,
194     proc.Procedure_Cost,
195     proc.Amount_Paid_DPCA,
196     fa.Aid_Date,
197     fa.Aid_Type
198 FROM
199     Patient p
200 JOIN
201     Patient_Financials pf ON p.Patient_ID = pf.Patient_ID
202 LEFT JOIN
203     Financial_Aid fa ON p.Patient_ID = fa.Patient_ID
204 LEFT JOIN
205     Procedure proc ON p.Patient_ID = proc.Patient_ID
206 ORDER BY
207     p.Patient_ID
208 "",
209
210 # Monthly Aid Distribution Report
211 'monthly_aid_distribution': ""
212 SELECT

```

```

213         strftime('%Y-%m', fa.Aid_Date) AS month,
214         fa.Aid_Type,
215         SUM(proc.Amount_Paid_DPCA) AS total_amount,
216         SUM(proc.Amount_Paid_DPCA) / COUNT(DISTINCT fa.Financial_Aid_ID
) AS average_aid_per_patient
217     FROM
218         Financial_Aid fa
219     JOIN
220         Procedure proc ON fa.Patient_ID = proc.Patient_ID
221     GROUP BY
222         month, fa.Aid_Type
223     ORDER BY
224         month, fa.Aid_Type
225     """,
226
227     # Patient Contact Information Report
228     'patient_contact_information': ""
229     SELECT
230         p.Patient_ID,
231         p.Patient_Name,
232         p.Contact_No,
233         p.Address
234     FROM
235         Patient p
236     ORDER BY
237         p.Patient_ID
238     """,
239
240     # Patient Aid Dependence Ratio Report
241     'patient_aid_dependence_ratio': ""
242     SELECT
243         p.Patient_ID,
244         proc.Amount_Paid_DPCA,
245         pf.Income,
246         proc.Amount_Paid_DPCA / NULLIF(pf.Income, 0) AS
aid_to_income_ratio
247     FROM
248         Patient p
249     JOIN
250         Patient_Financials pf ON p.Patient_ID = pf.Patient_ID
251     LEFT JOIN
252         Procedure proc ON p.Patient_ID = proc.Patient_ID
253     ORDER BY
254         p.Patient_ID
255     """,
256
257     # Procedure Cost Analysis Report
258     'procedure_cost_analysis': ""
259     SELECT
260         proc.Procedure_Type,
261         AVG(proc.Procedure_Cost) as average_procedure_cost
262     FROM
263         Procedure proc
264     GROUP BY
265         proc.Procedure_Type
266     ORDER BY
267         average_procedure_cost
268     """)

```

```

269 }
270
271
272 # Function to generate reports
273 def generate_reports():
274
275     conn = sqlite3.connect("Cleaned_DPCA.db")
276     timestamp = datetime.now().strftime('%Y%m%d_%H%M%S')
277
278     for report_name, query in queries.items():
279         try:
280             # Execute query
281             df = pd.read_sql_query(query, conn)
282
283             # Save to CSV
284             filename = f'reports/DPCA_{report_name}_report_{timestamp}.
csv'
285
286             df.to_csv(filename, index=False)
287
288             # Print summary
289             print(f"\nReport generated: {report_name}")
290             print(f"Total records: {len(df)}")
291             print(f"Saved to: {filename}")
292             print("-" * 50)
293
294             if len(df) > 0:
295                 # Display preview
296                 print("\nFirst few records:")
297                 print(df.head())
298
299                 # Generate summary statistics for numerical columns
300                 numeric_cols = df.select_dtypes(include=[np.number]).
columns
301
302                 if len(numeric_cols) > 0:
303                     print("\nSummary statistics:")
304                     print(df[numeric_cols].describe())
305                 else:
306                     print(f"Warning: No records found for {report_name}
report!")
307
308             except Exception as e:
309                 print(f"Error generating {report_name} report: {str(e)}")
310                 print(f"Query used: {query}") # Print the query for
debugging
311
312             conn.close()
313
314 # Execute report generation
315 generate_reports()
316
317 print("\nAll reports have been generated and saved in the 'reports'
directory.")

```

Listing 5: SQL Queries and Reports

B Appendix B: Additional Figures

B.1 Correlation Matrix Heatmap

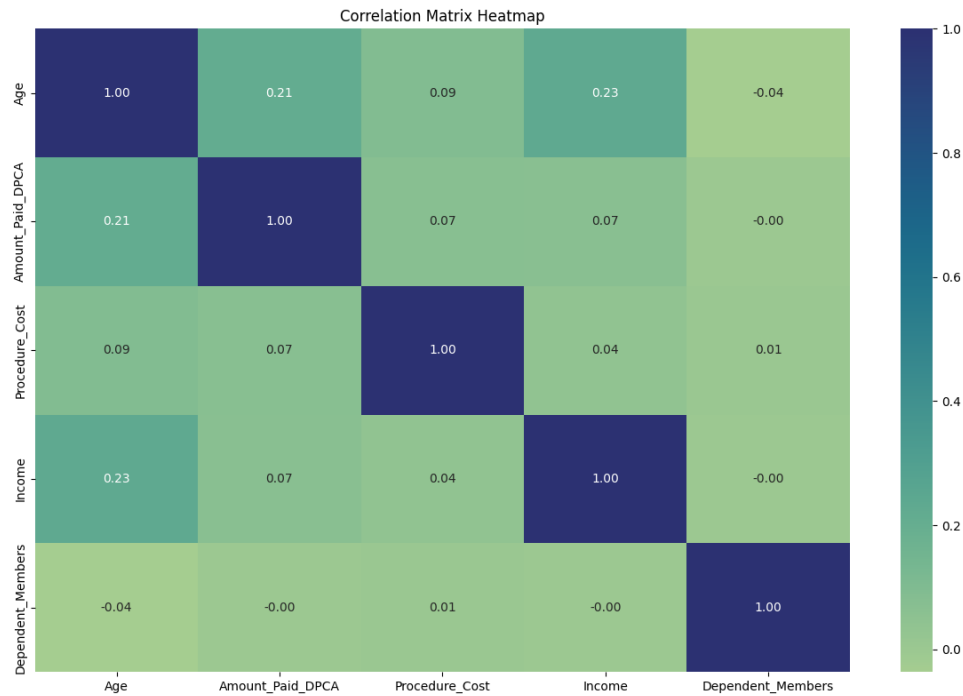


Figure 17: Correlation Matrix Heatmap

B.2 Procedure Types by Occupation

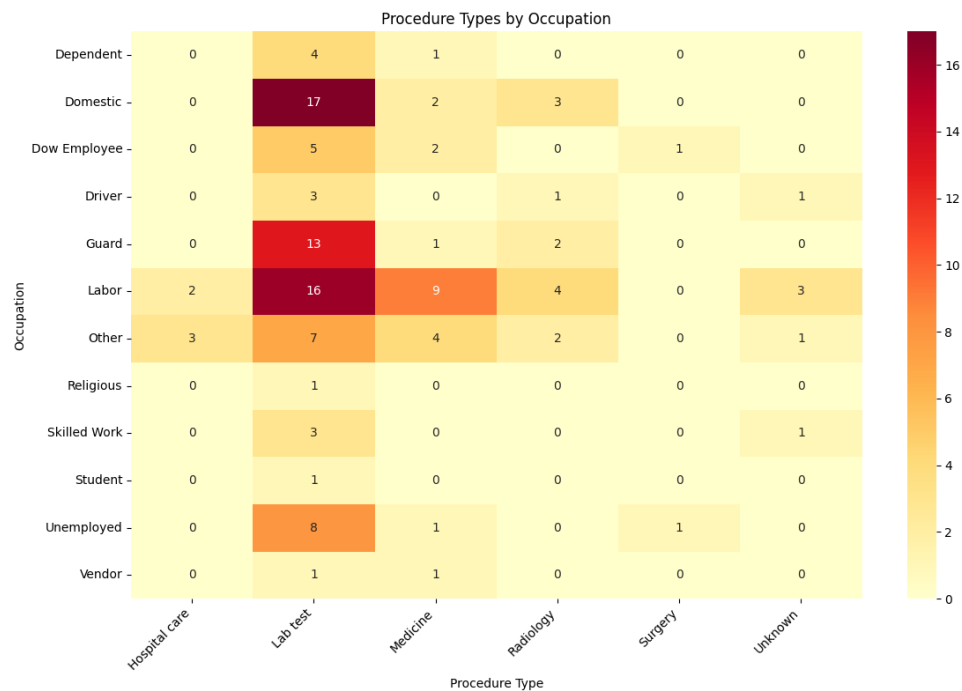


Figure 18: Procedure Types by Occupation

B.3 Procedure Types by Occupation pc

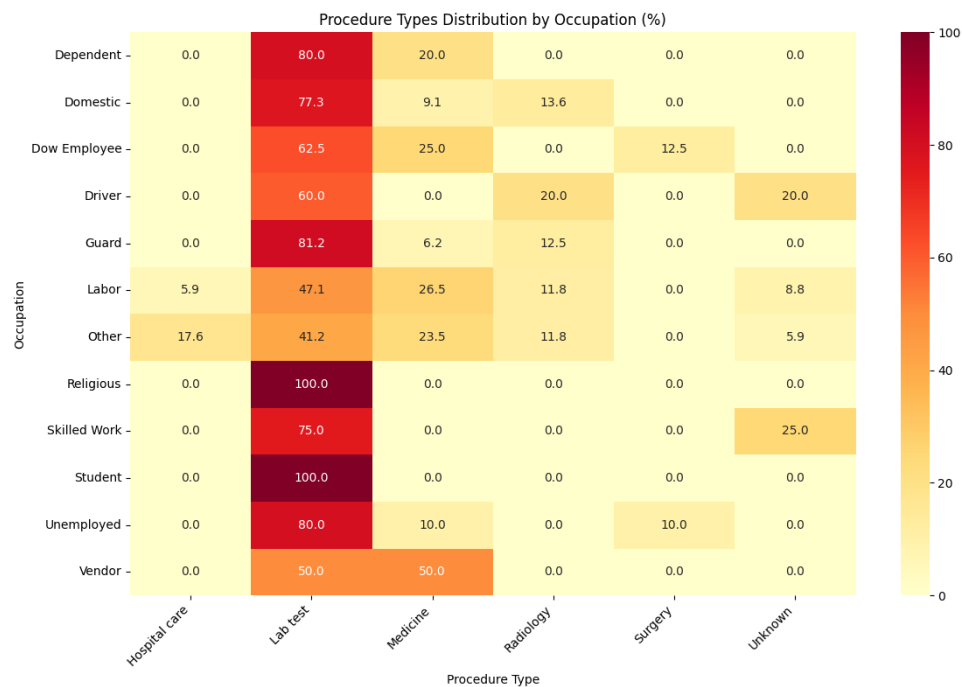


Figure 19: Procedure Types by Occupation pc