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 3, 2024

Contents

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 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

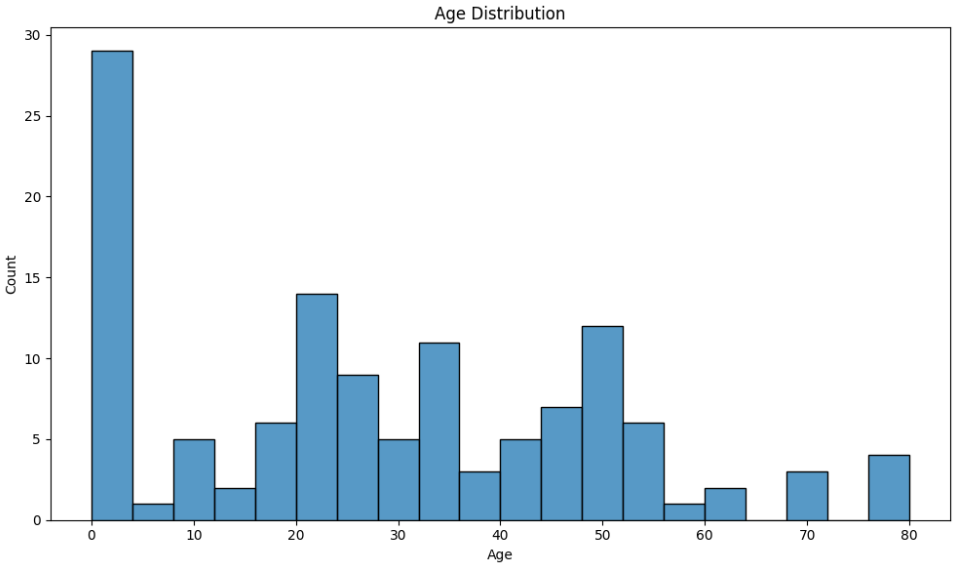

Figure 1: Age Distribution

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

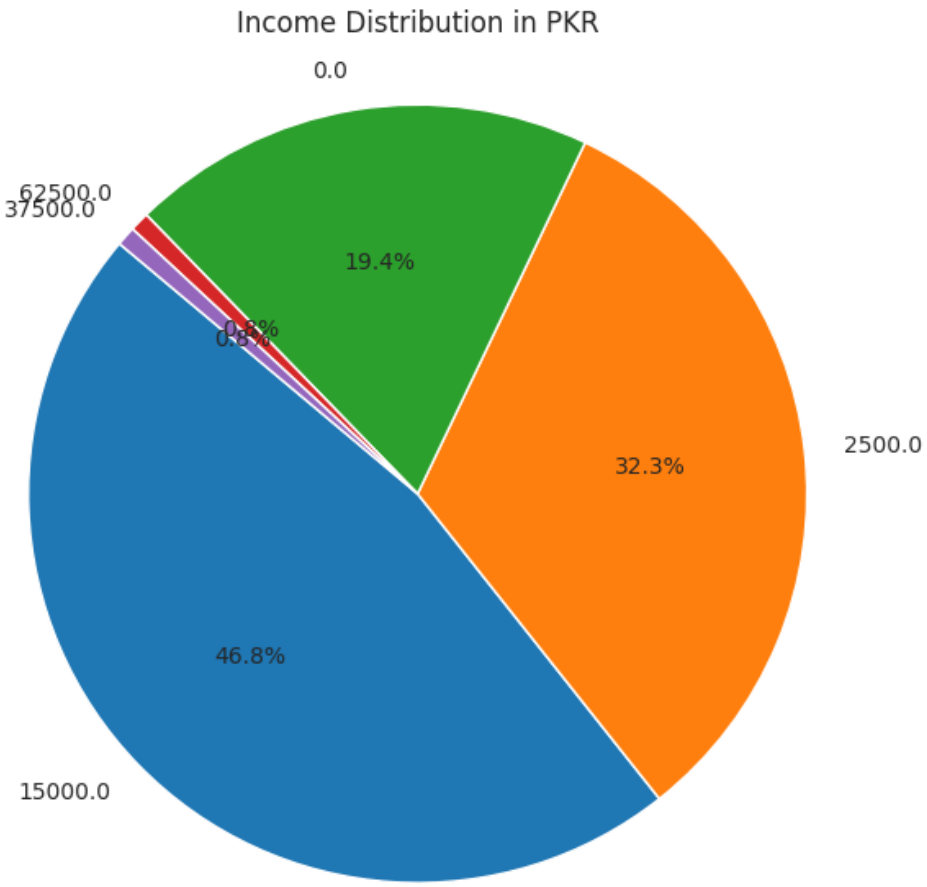

Figure 2: Income Distribution in PKR/month

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

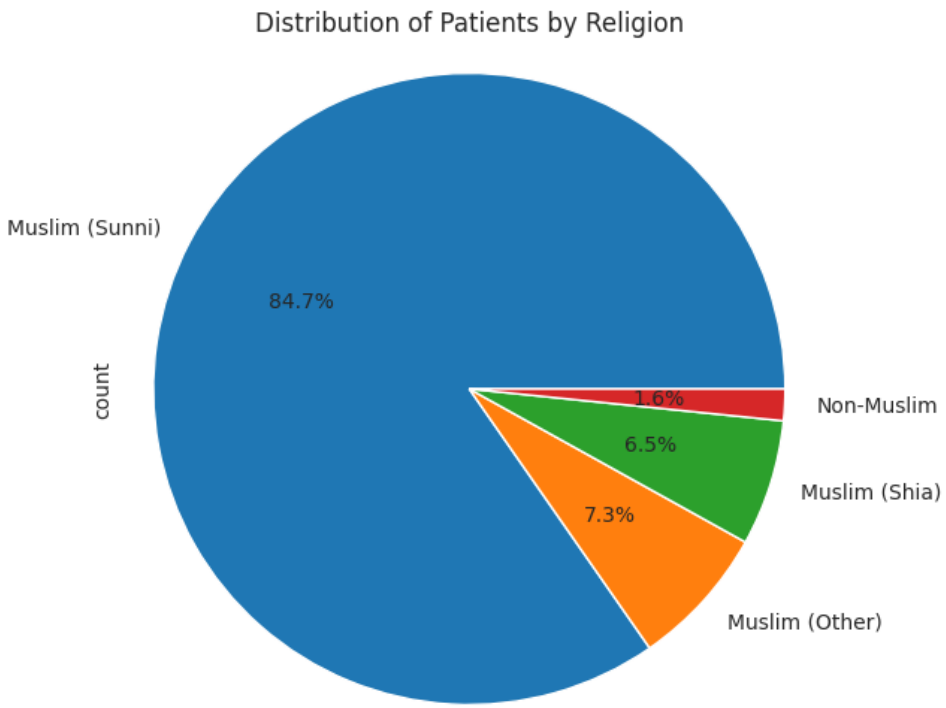

Figure 3: Distribution of Patients by Religion

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

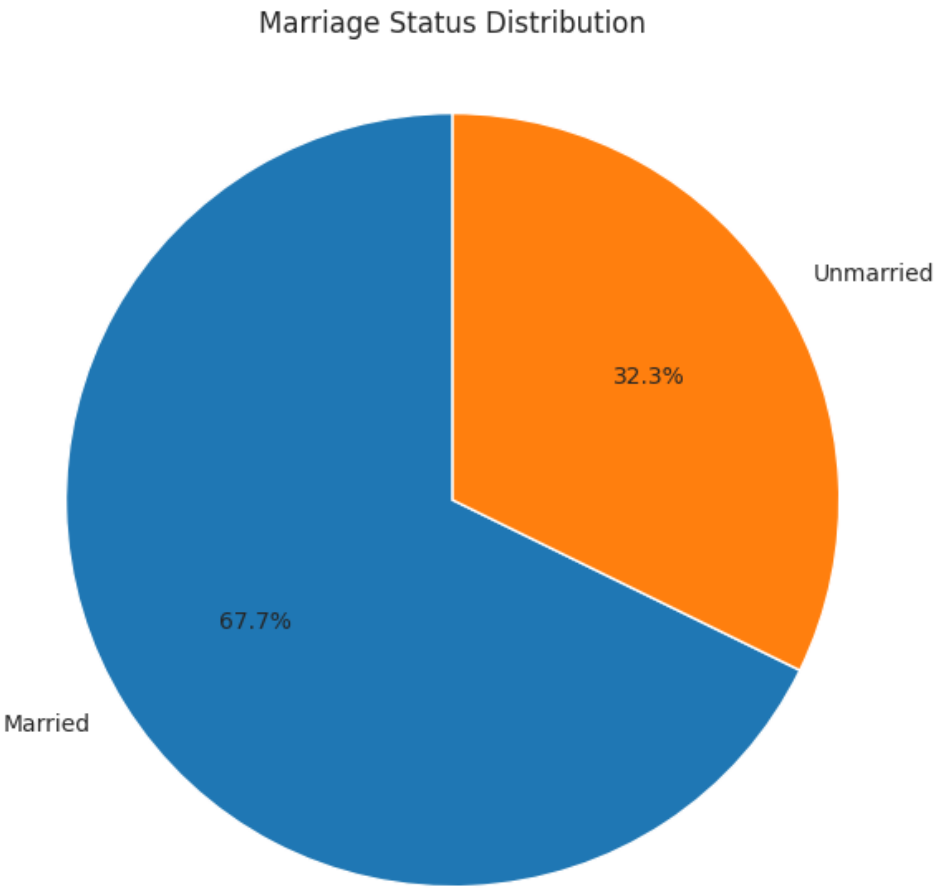

Figure 4: Marriage Status Distribution

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

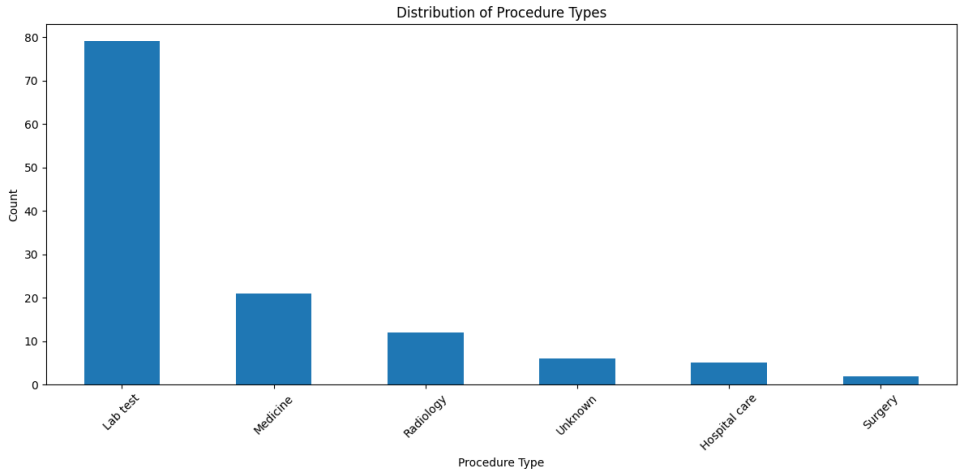

Figure 5: Distribution of Procedure Types

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

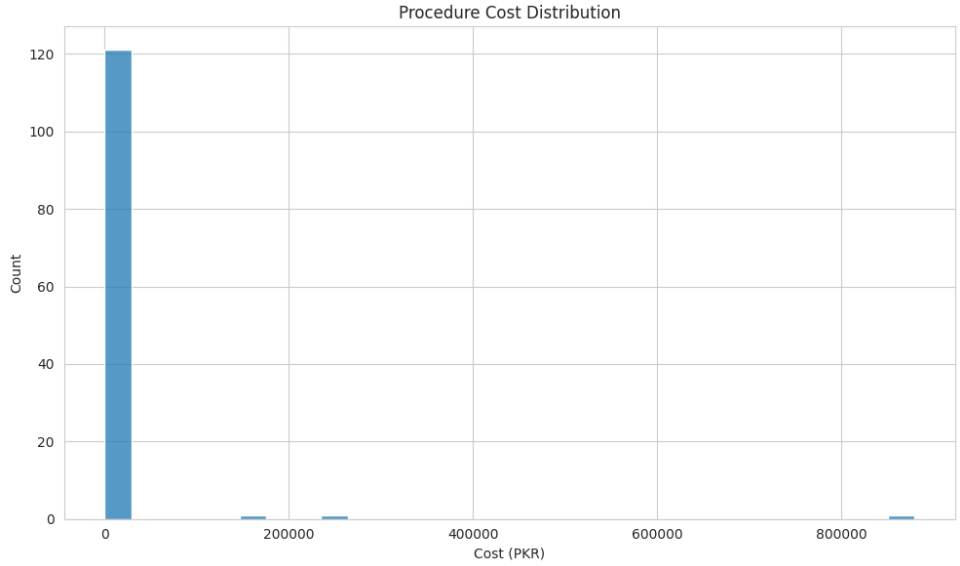

Figure 6: Distribution of Procedure Costs

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 of the few expensive

procedures in the dataset, Figure 6 does not provide detailed insights on low cost procedures 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

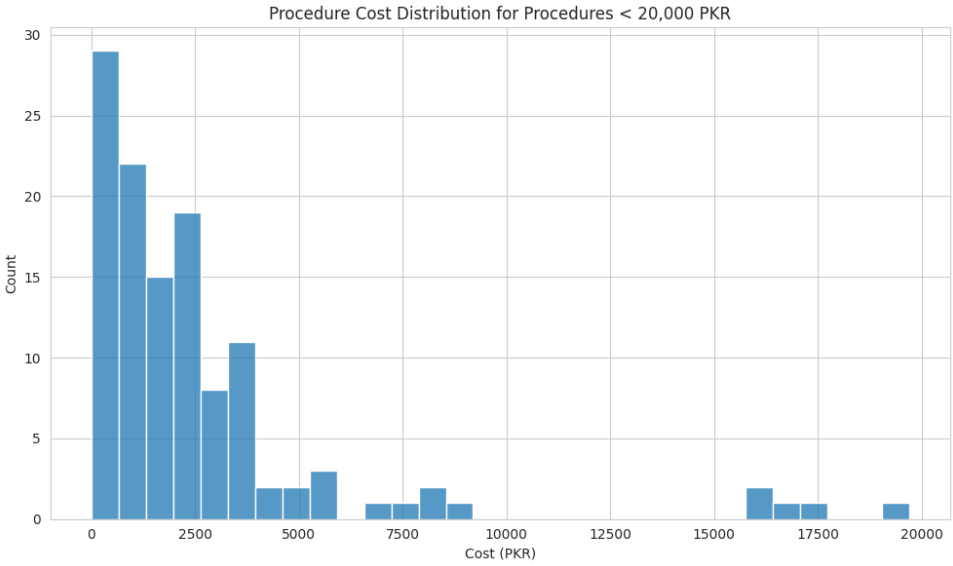

Figure 7: Distribution of Procedure Costs

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, which will be discussed further in Figure 9.

4.1.8 Procedure Cost Distribution More than 20,000 PKR

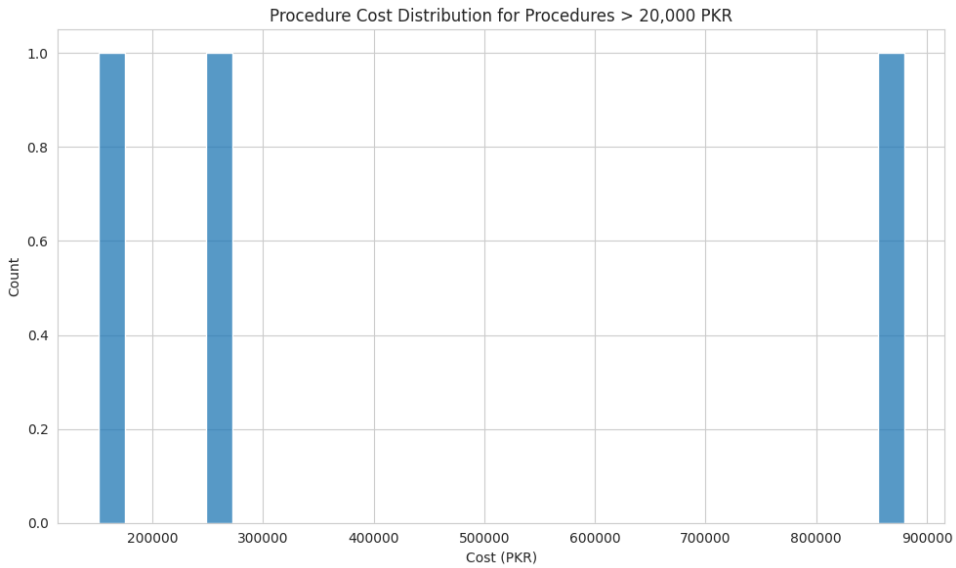

Figure 8: Distribution of Procedure Costs

Figure 8 shows the distribution of procedure costs that are more than 20,000 PKR. There are only 3 procedures 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

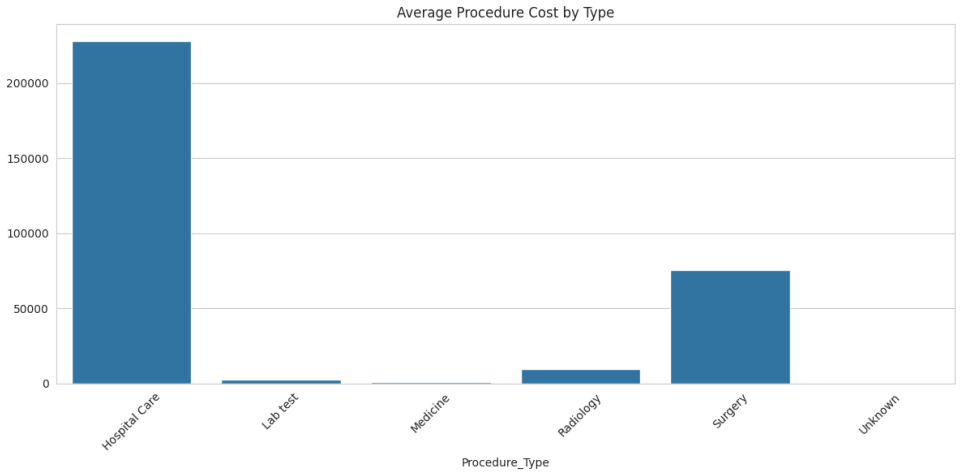

Figure 9: Average Procedure Cost by Type

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

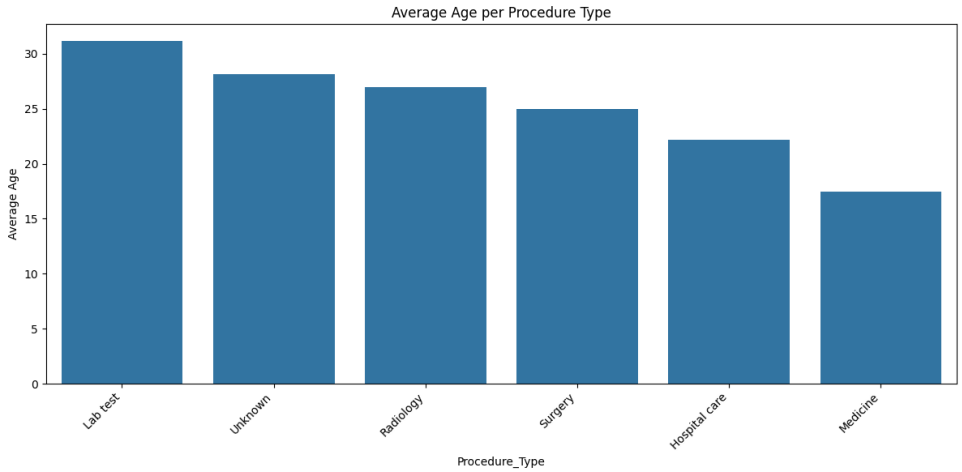

Figure 10: Average Age per Procedure Type

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

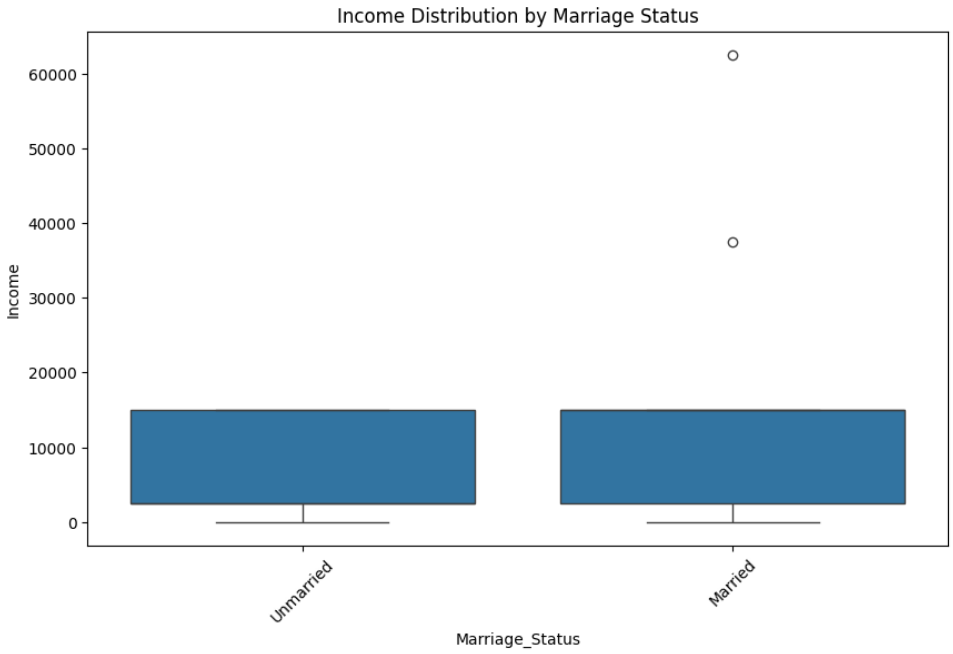

Figure 11: Income Distribution by Marriage Status

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

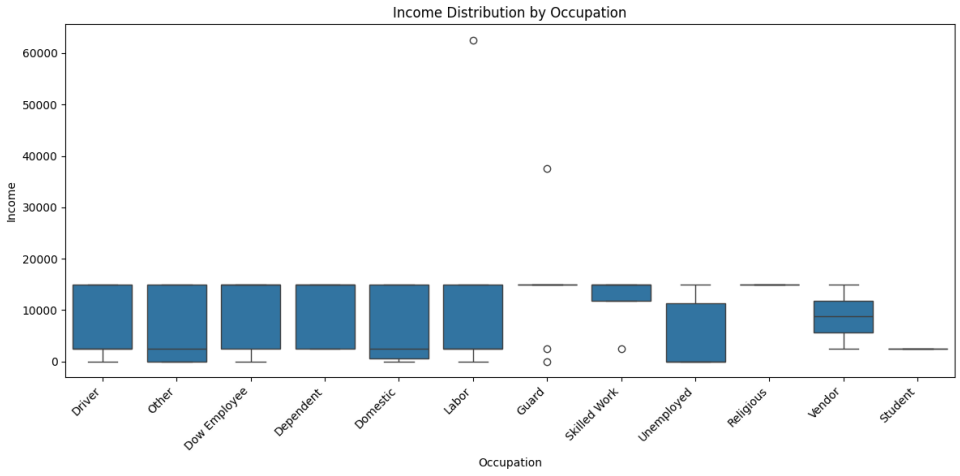

Figure 12: Income Distribution by Occupation

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:

- **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

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

- 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

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 dashboard serves as a comprehensive 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

```
# import libraries
import numpy as np
import pandas as pd
import sqlite3
import re
import matplotlib.pyplot as plt
import seaborn as sns
from google.colab import files
from datetime import datetime
import os

# import database
def ReadDB(table_name):
    conn = sqlite3.connect('/content/DPCA.db')

    query = f'SELECT * FROM {table_name}'
    df = pd.read_sql(query, conn)

    conn.close()

    return df

df_patient = ReadDB('Patient')
df_procedure = ReadDB('Procedure')
df_patient_financials = ReadDB('Patient_Financials')
df_financial_aid = ReadDB('Financial_Aid')
df_zakat_eligibility = ReadDB('Zakat_Eligibility')

# combine dataframes
df_all = pd.concat([df_patient, df_procedure, df_patient_financials,
    df_financial_aid, df_zakat_eligibility], axis=1)

# check for row duplicates
df_all.duplicated().sum()

# check for column duplicates
transposed_df = df_all.transpose()              # transpose dataframe
transposed_df.duplicated().sum()                # find column
    duplicates

# drop column duplicates (no row duplicates)
transposed_df.drop_duplicates(inplace = True)
transposed_df.duplicated().sum()
```

```
42
43 # transpose dataframe to return to original dataframe with no
   duplicates
44 df_all = transposed_df.transpose()
45
46 # Remove duplicate headers and set proper data types
47 df_all = df_all.iloc[1:]  # Remove duplicate header row
48
49 # Clean Patient Data
50 df_all.loc[:, 'Age'] = pd.to_numeric(df_all['Age'], errors='coerce')
51 df_all.loc[:, 'Patient_ID'] = pd.to_numeric(df_all['Patient_ID'],
   errors='coerce')
52
53 # Clean Procedure Data
54 df_all.loc[:, 'Procedure_ID'] = pd.to_numeric(df_all['Procedure_ID'],
   errors='coerce')
55 df_all.loc[:, 'Patient_ID'] = pd.to_numeric(df_all['Patient_ID'],
   errors='coerce')
56 df_all.loc[:, 'Procedure_Cost'] = pd.to_numeric(df_all['Procedure_Cost'
   ], errors='coerce')
57 df_all.loc[:, 'Amount_Paid_DPCA'] = pd.to_numeric(df_all['
   Amount_Paid_DPCA'], errors='coerce')
58
59 # Clean Patient Financials Data
60 df_all.loc[:, 'Financial_Info_ID'] = pd.to_numeric(df_all['
   Financial_Info_ID'], errors='coerce')
61 df_all.loc[:, 'Patient_ID'] = pd.to_numeric(df_all['Patient_ID'],
   errors='coerce')
62 df_all.loc[:, 'Dependent_Members'] = pd.to_numeric(df_all['
   Dependent_Members'], errors='coerce')
63
64 # Clean Financial Aid Data
65 df_all.loc[:, 'Financial_Aid_ID'] = pd.to_numeric(df_all['
   Financial_Aid_ID'], errors='coerce')
66 df_all.loc[:, 'Patient_ID'] = pd.to_numeric(df_all['Patient_ID'],
   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')
```

```
df_all['Aid_Type'] = df_all['Aid_Type'].fillna('N.A')
df_all['Eligibility_Criteria'] = df_all['Eligibility_Criteria'].fillna('N.A')

df_all['Religion'] = df_all['Religion'].fillna('Unknown')
df_all['Syed_NonSyed'] = df_all['Syed_NonSyed'].fillna('Unknown')
df_all['Marriage_Status'] = df_all['Marriage_Status'].fillna('Unknown')
df_all['Contact_No'] = df_all['Contact_No'].fillna('Unknown')
df_all['Address'] = df_all['Address'].fillna('Unknown')
df_all['Occupation'] = df_all['Occupation'].fillna('Unknown')
df_all['Procedure_Date'] = df_all['Procedure_Date'].fillna('Unknown')

df_all['Dependent_Members'] = df_all['Dependent_Members'].fillna(0)
df_all['Properties_Owned'] = df_all['Properties_Owned'].fillna(0)
df_all['Amount_Paid_DPCA'] = df_all['Amount_Paid_DPCA'].fillna(0)
df_all['Financial_Support_Method'] = df_all['Financial_Support_Method'].fillna('Unknown')
df_all['Is_Eligible'] = df_all['Is_Eligible'].fillna('No')

df_all['Patient_ID'] = df_all['Patient_ID'].fillna('Unknown')  # or use a default ID if needed
df_all['Patient_Name'] = df_all['Patient_Name'].fillna('Unknown')
df_all['Age'] = df_all['Age'].fillna(0)  # assuming age 0 for missing data
df_all['CNIC_No'] = df_all['CNIC_No'].fillna('Unknown')
df_all['Procedure_Type'] = df_all['Procedure_Type'].fillna('Unknown')
df_all['MR_number'] = df_all['MR_number'].fillna('Unknown')
df_all['Aid_Date'] = df_all['Aid_Date'].fillna('Unknown')

df_all.isnull().sum()

# Check non-numeric entries
def check_non_numeric(column):
    return df_all[column][pd.to_numeric(df_all[column], errors='coerce').isna()]

print("Non-numeric entries in Amount_Paid_DPCA:", check_non_numeric('Amount_Paid_DPCA'))
print("Non-numeric entries in Procedure_Cost:", check_non_numeric('Procedure_Cost'))
print("Non-numeric entries in Income:", check_non_numeric('Income'))

# Function to clean the 'Income' and 'Procedure_Cost' columns
def clean_numeric_column(column):
    # Replace non-numeric values with NaN to handle them separately
    df_all[column] = pd.to_numeric(df_all[column], errors='coerce')

    # Replace specific strings with NaN (or you could replace with 0 if you prefer)
    df_all[column] = df_all[column].replace(['No income', 'N.A'], np.nan)

    # Handle range values by taking the midpoint of the range
    def convert_range(value):
        if isinstance(value, str) and '-' in value:
            parts = value.split('-')
            try:
                # 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(
    clean_financial_column)
172 df_all['Procedure_Cost'] = df_all['Procedure_Cost'].apply(
    clean_financial_column)
173 df_all['Income'] = df_all['Income'].apply(clean_financial_column)
174
175 # Function to clean Income column
176 def clean_income(value):
177     if pd.isna(value):
178         return value
179
180     # Handle "No income" case
```

```
    if "no income" in str(value).lower():
        return 0

    # Handle ranges (e.g., "1-10000/month" -> take the midpoint)
    range_match = re.match(r'(\d+)[\D]+(\d+)', str(value))
    if range_match:
        lower = int(range_match.group(1))
        upper = int(range_match.group(2))
        return (lower + upper) / 2  # Return the midpoint of the range

    # Clean and convert as a regular financial value
    return clean_financial_column(value)

# Apply to the Income column
df_all['Income'] = df_all['Income'].apply(clean_income)

# Check data types
print(df_all[['Amount_Paid_DPCA', 'Procedure_Cost', 'Income']].dtypes)

# Display any remaining non-numeric entries
print("Remaining non-numeric in Amount_Paid_DPCA:", df_all['
    Amount_Paid_DPCA'][pd.to_numeric(df_all['Amount_Paid_DPCA'], errors=
    'coerce').isna()])
print("Remaining non-numeric in Procedure_Cost:", df_all['
    Procedure_Cost'][pd.to_numeric(df_all['Procedure_Cost'], errors='
    coerce').isna()])
print("Remaining non-numeric in Income:", df_all['Income'][pd.
    to_numeric(df_all['Income'], errors='coerce').isna()])

# Convert columns to numeric, coercing any remaining non-numeric values
    to NaN
df_all['Amount_Paid_DPCA'] = pd.to_numeric(df_all['Amount_Paid_DPCA'],
    errors='coerce')
df_all['Procedure_Cost'] = pd.to_numeric(df_all['Procedure_Cost'],
    errors='coerce')
df_all['Income'] = pd.to_numeric(df_all['Income'], errors='coerce')

# Check data types again
print(df_all[['Amount_Paid_DPCA', 'Procedure_Cost', 'Income']].dtypes)

# Validate financial columns
invalid_amount_paid = df_all[df_all['Amount_Paid_DPCA'] < 0]
invalid_procedure_cost = df_all[df_all['Procedure_Cost'] < 0]
invalid_income = df_all[df_all['Income'] < 0]

print("Invalid Amount_Paid_DPCA:", invalid_amount_paid)
print("Invalid Procedure_Cost:", invalid_procedure_cost)
print("Invalid Income:", invalid_income)

# Further clean the 'Age' column to ensure all values are numeric

def clean_age(age):
    if isinstance(age, str):
        # Extract numeric part
        numbers = re.findall(r'\d+\.?\d*', age)
        if numbers:
            return float(numbers[0])
    try:
```

```
        return float(age)
    except ValueError:
        return np.nan  # Return NaN for any non-convertible values

# Apply the cleaning function
df_all['Age'] = df_all['Age'].apply(clean_age)

# Drop any rows with NaN values in 'Age' or 'Procedure_Cost' to ensure clean calculations
df_all.dropna(subset=['Age', 'Procedure_Cost'], inplace=True)

# Clean Occupation column
def clean_occupation(value):
    # Convert the value to a string for consistent handling
    value_str = str(value).lower()

    # Handle "Dow" case
    if "dow" in value_str:
        return "Dow Employee"

    # Handle "Driver" case
    if "rickshaw" in value_str or "driver" in value_str or "rikshaw" in value_str:
        return "Driver"

    # Handle "Guard" case
    if "guard" in value_str or "gaurd" in value_str or "security" in value_str:
        return "Guard"

    # Handle "Domestic" case
    if "house" in value_str or "maid" in value_str or "cook" in value_str or "clean" in value_str:
        return "Domestic"

    # Handle "Labor" case
    if any(term in value_str for term in ["labour", "labor", "mazdoor", "sweep", "janitor", "cement", "block maker", "trash", "vendor"]):
        return "Labor"

    # Handle "Dependent" case
    if any(term in value_str for term in ["father", "husband", "brother", "son", "wife"]):
        return "Dependent"

    # Handle "Vendor" case
    if any(term in value_str for term in ["sell", "vendor", "stall"]):
        return "Vendor"

    # Handle "Student" case
    if "student" in value_str:
        return "Student"

    # Handle "Unemployed" case
    if any(term in value_str for term in ["unemployed", "not", "none", "jobless", "does nothing", "-", "N/A"]):
        return "Unemployed"

```

```
    # Handle "Religious" case
    if "imam" in value_str:
        return "Religious"

    # Handle "Skilled Work" case
    if any(term in value_str for term in ["sewing", "tailor", "packager", "chai hotel", "operator", "machine"]):
        return "Skilled Work"

    # Default case
    return "Other"

# Apply to the Income column
df_all['Occupation'] = df_all['Occupation'].apply(clean_occupation)

# Clean Procedure Type column
def clean_procedure(value):
    # Convert the value to a string for consistent handling
    value_str = str(value).lower()

    # Handle "Lab test" case
    if "lab test" in value_str:
        return "Lab test"

    # Handle "Surgery" case
    if "surgery" in value_str:
        return "Surgery"

    # Handle "Medicine" case
    if "med" in value_str:
        return "Medicine"

    # Handle "Hospital care" case
    if "hospital" in value_str:
        return "Hospital Care"

    # Handle "Radiology" case
    if "radiology" in value_str:
        return "Radiology"

    # Default case: return the original value
    return value

# Apply to the Income column
df_all['Procedure_Type'] = df_all['Procedure_Type'].apply(clean_procedure)

# Clean Financial Support Method column
def clean_financial_support(value):
    # Convert the value to a string for consistent handling
    value_str = str(value).lower()

    # Handle "Family" case
    if any(term in value_str for term in ["husband", "wife", "father", "mother", "brother", "sister", "daughter", "son", "relative", "friend"]):
        return "Family"

```

```
    # Handle "Self Employed" case
    if "self" in value_str:
        return "Self Employed"

    # Handle "Welfare" case
    if "welfare" in value_str:
        return "Welfare"

    # Handle "Employed" case
    if any(term in value_str for term in ["job", "work", "sell", "sold", "work", "employed", "sweeper", "labour", "dipca"]):
        return "Employed"

   # Handle "Unknown" case
    if any(term in value_str for term in ["unknown"]):
        return "Unknown"

    # Handle "None" case
    if any(term in value_str for term in ["-", "no", "nil", "N/A", "n/a", "none", "non", ".."]):
        return "None"

    # Default case: return the original value
    return value

# Apply to the Income column
df_all['Financial_Support_Method'] = df_all['Financial_Support_Method'].apply(clean_financial_support)

# Clean Properties Owned column
def clean_properties_owned(value):
    # Convert the value to a string for consistent handling
    value_str = str(value).lower()

    # Handle "2" case
    if any(term in value_str for term in ["and"]):
        return "2"

    # Handle "0" case
    if any(term in value_str for term in ["0", "rent", "no", "-", "has gold earrings", "kacha"]):
        return "0"

    # Handle "1" case
    if any(term in value_str for term in ["1", "home", "house", "jhopri"]):
        return "1"

    # Default case: return the original value
    return value

# Apply to the Income column
df_all['Properties_Owned'] = df_all['Properties_Owned'].apply(clean_properties_owned)

# Print out cleaned and processed database for checking
```

```
df_all.head(50)
```

Listing 1: Code for Data Cleaning and Preprocessing

A.2 Additional Code for Aid Type and Amount

```

print("\
Top 3 most common occupation-procedure combinations:")
# Get the top 3 occupation-procedure combinations
flat_crosstab = job_procedure_crosstab.unstack()
top_combinations = flat_crosstab.sort_values(ascending=False)[:3]
for (occupation, procedure), count in top_combinations.items():
    print(f"{occupation} - {procedure}: {count} cases")

# Create aid_type column based on eligibility criteria
def determine_aid_type(criteria):
    if isinstance(criteria, str):
        criteria = criteria.lower()
        # Check for all required conditions for Zakat
        muslim_condition = 'muslim' in criteria
        non_syed_condition = 'non-syed' in criteria or 'non syed' in criteria
        wealth_condition = '52.5 tola' in criteria or 'does not own' in criteria

        if muslim_condition and non_syed_condition and wealth_condition:
            return 'Zakat Funds'
    return 'General Funds'

# Apply the function to create aid_type column
df_all['Aid_Type'] = df_all['Eligibility_Criteria'].apply(determine_aid_type)

# Create is_eligible column based on aid_type
df_all['Is_Eligible'] = df_all['Aid_Type'].apply(lambda x: 'Yes' if x == 'Zakat Funds' else 'No')

# Display sample results
print("Sample of Eligibility Criteria, Aid Type, and Eligibility:")
print(df_all[['Eligibility_Criteria', 'Aid_Type', 'Is_Eligible']].head(10))

# Display distribution of aid types
print("Distribution of Aid Types:")
print(df_all['Aid_Type'].value_counts())

# Display eligibility counts
print("Distribution of Eligibility:")
print(df_all['Is_Eligible'].value_counts())

# Calculation of Aid Amounts based on eligibility

# monthly_budget = 168000  # Rupees

# # Calculate total aid by type
```

```
# aid_summary = df_all.groupby('Aid_Type').agg({
#     'Aid_Amount': ['sum', 'count', 'mean']
# }).round(2)

# aid_summary.columns = ['Total_Amount', 'Number_of_Patients', '
    Average_Amount']
# aid_summary = aid_summary.reset_index()

# print("Aid Allocation Summary:")
# print(aid_summary)

# # Calculate percentages
# total_aid = aid_summary['Total_Amount'].sum()
# aid_summary['Percentage_of_Total_Aid'] = (aid_summary['Total_Amount']
    / total_aid * 100).round(2)

# print("\
# Detailed Aid Distribution:")
# print(df_all.head())
```

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

A.3 Data Visualization

```

# Undo mapped data for charts
df_all['Religion'] = df_all['Religion'].map({0:'Muslim (Shia)', 1:'
    Muslim (Sunni)'})
df_all['Syed_NonSyed'] = df_all['Syed_NonSyed'].map({0:'Syed', 1:'Non-
    Syed'})
df_all['Marriage_Status'] = df_all['Marriage_Status'].map({0:'Married',
    1:'Unmarried'})

# Visualizations
sns.set_style('whitegrid')

# Histogram for Procedure Cost
plt.figure(figsize=(10, 6))
sns.histplot(data=df_all, x='Procedure_Cost', bins=30)
plt.title('Procedure Cost Distribution')
plt.xlabel('Cost (PKR)')
plt.ylabel('Count')
plt.tight_layout()
plt.show()

# Histogram for Procedure Cost < 20K PKR
less20 = df_all[df_all['Procedure_Cost'] <= 20000]

plt.figure(figsize=(10, 6))
sns.histplot(data=less20, x='Procedure_Cost', bins=30)
plt.title('Procedure Cost Distribution for Procedures < 20,000 PKR')
plt.xlabel('Cost (PKR)')
plt.ylabel('Count')
plt.tight_layout()
plt.show()

# Histogram for Procedure Cost > 10K PKR
```

```
more20 = df_all[df_all['Procedure_Cost'] > 20000]

plt.figure(figsize=(10, 6))
sns.histplot(data=more20, x='Procedure_Cost', bins=30)
plt.title('Procedure Cost Distribution for Procedures > 20,000 PKR')
plt.xlabel('Cost (PKR)')
plt.ylabel('Count')
plt.tight_layout()
plt.show()

# Procedure Types Distribution
plt.figure(figsize=(12, 6))
df_all['Procedure_Type'].value_counts().plot(kind='bar')
plt.title('Distribution of Procedure Types')
plt.xlabel('Procedure Type')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Average Procedure Cost by Type
plt.figure(figsize=(12, 6))
avg_cost = df_all.groupby('Procedure_Type')['Procedure_Cost'].mean()
sns.barplot(x=avg_cost.index, y=avg_cost.values)
plt.title('Average Procedure Cost by Type')
plt.xticks(rotation=45)
plt.tight_layout()
#plt.savefig('avg_procedure_cost.png')
plt.show()

# Average Age per Procedure
plt.figure(figsize=(12, 6))
avg_age = df_all.groupby('Procedure_Type')['Age'].mean().sort_values(
    ascending=False)
sns.barplot(x=avg_age.index, y=avg_age.values)
plt.title('Average Age per Procedure Type')
plt.xticks(rotation=45, ha='right')
plt.ylabel('Average Age')
plt.tight_layout()
#plt.savefig('avg_age_per_procedure.png')
plt.show()

# Religion Distribution
plt.figure(figsize=(6, 6))
df_all['Religion'].value_counts().plot(kind='pie', autopct='%1.1f%%')
plt.title('Distribution of Patients by Religion')
plt.axis('equal')
plt.show()


# Income Distribution by Marriage Status
plt.figure(figsize=(10, 6))
sns.boxplot(data=df_all, x='Marriage_Status', y='Income')
plt.title('Income Distribution by Marriage Status')
plt.xticks(rotation=45)
plt.show()

# Pie chart for income distribution
```

```
plt.figure(figsize=(6, 6))
income_counts = df_all['Income'].value_counts()
plt.pie(income_counts, labels=income_counts.index, autopct='%1.1f%%',
    startangle=140)
plt.title('Income Distribution in PKR')
plt.axis('equal')
plt.tight_layout()
plt.show()

# Pie chart for marriage status
plt.figure(figsize=(6, 6))
marriage_counts = df_all['Marriage_Status'].value_counts()
plt.pie(marriage_counts, labels=marriage_counts.index, autopct='%1.1f%%
    ', startangle=90)
plt.title('Marriage Status Distribution')
plt.axis('equal')
plt.tight_layout()
plt.show()

# Histogram for age distribution
plt.figure(figsize=(10, 6))
sns.histplot(data=df_all, x='Age', bins=20)
plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Count')
plt.tight_layout()
plt.show()

# Box plots for income by occupation
plt.figure(figsize=(12, 6))
sns.boxplot(x='Occupation', y='Income', data=df_all)
plt.xticks(rotation=45, ha='right')
plt.title('Income Distribution by Occupation')
plt.tight_layout()
plt.show()

numeric_df = df_all.select_dtypes(include=[np.number])

# Calculate the correlation matrix for numeric columns
correlation_matrix = numeric_df.corr()

# Create a heatmap for the correlation matrix
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='crest', fmt='.2f')
plt.title('Correlation Matrix Heatmap')
plt.tight_layout()
plt.show()

print("Correlation matrix heatmap created")

# Analyze job impact on procedure type
job_procedure_crosstab = pd.crosstab(df_all['Occupation'], df_all['
    Procedure_Type'])

# Create a heatmap of the job-procedure relationship
plt.figure(figsize=(12, 8))
sns.heatmap(job_procedure_crosstab, annot=True, fmt='d', cmap='YlOrRd')
plt.title('Procedure Types by Occupation')
```

```
plt.xlabel('Procedure Type')
plt.ylabel('Occupation')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()

# Calculate percentage distribution of procedures within each occupation
procedure_pct = job_procedure_crosstab.div(job_procedure_crosstab.sum(axis=1), axis=0) * 100

# Create a heatmap of the percentage distribution
plt.figure(figsize=(12, 8))
sns.heatmap(procedure_pct, annot=True, fmt='.1f', cmap='YlOrRd')
plt.title('Procedure Types Distribution by Occupation (%)')
plt.xlabel('Procedure Type')
plt.ylabel('Occupation')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

Listing 3: Code for Data Visualization

A.4 Exporting Cleaned Database

```

# Split large dataframe into individual dataframes for tables
clean_df_patient = df_all[['Patient_ID','Patient_Name','Age','Religion','Syed_NonSyed','Marriage_Status','CNIC_No','Contact_No','Address']]
clean_df_procedure = df_all[['Procedure_ID','Procedure_Type','Procedure_Date','MR_number','Amount_Paid_DPCA','Procedure_Cost','Patient_ID']]
clean_df_patient_financials = df_all[['Financial_Info_ID','Occupation','Income','Dependent_Members','Financial_Support_Method','Properties_Owned','Patient_ID']]
clean_df_financial_aid = df_all[['Financial_Aid_ID','Aid_Date','Aid_Type','Aid_Amount','Financial_Info_ID','Zakat_Eligibility_ID','Patient_ID']]
clean_df_zakat_eligibility = df_all[['Zakat_Eligibility_ID','Is_Eligible','Eligibility_Criteria','Patient_ID','Financial_Aid_ID']]

# Connect to SQLite
conn = sqlite3.connect("Cleaned_DPCA.db")

# Export dataframes to SQLite tables
clean_df_patient.to_sql(name="Patient", con = conn, if_exists="replace", index=False)
clean_df_procedure.to_sql(name="Procedure", con = conn, if_exists="replace", index=False)
clean_df_patient_financials.to_sql(name="Patient_Financials", con = conn, if_exists="replace", index=False)
clean_df_financial_aid.to_sql(name="Financial_Aid", con = conn, if_exists="replace", index=False)
clean_df_zakat_eligibility.to_sql(name="Zakat_Eligibility", con = conn, if_exists="replace", index=False)

```

```
# Define relationships via primary and foreign keys
conn.execute("PRAGMA foreign_keys = ON;")

# Patient table
conn.execute("""
CREATE TABLE IF NOT EXISTS Patient (
    Patient_ID INT PRIMARY KEY,
    Patient_Name VARCHAR(255) NOT NULL,
    Age INT, Religion VARCHAR(255),
    Syed_NonSyed VARCHAR(255),
    Marriage_Status VARCHAR(255),
    CNIC_No VARCHAR(255),
    Contact_No VARCHAR(255),
    Address VARCHAR(255)
);
""")

clean_df_patient.to_sql(name="Patient", con=conn, if_exists="replace",
    index=False)       # insert data into table

# Procedure table
conn.execute("""
CREATE TABLE IF NOT EXISTS Procedure (
    Procedure_ID INT PRIMARY KEY,
    Procedure_Type VARCHAR(255) NOT NULL,
    Procedure_Date DATE,
    MR_number VARCHAR(255),
    Amount_Paid_DPCA INT,
    Procedure_Cost INT,
    Patient_ID INT,
    FOREIGN KEY (Patient_ID) REFERENCES Patient(Patient_ID)
);
""")

clean_df_procedure.to_sql(name="Procedure", con=conn, if_exists="
    replace", index=False)       # insert data into table

# Patient Financials table
conn.execute("""
CREATE TABLE IF NOT EXISTS Patient_Financials (
    Financial_Info_ID INT PRIMARY KEY,
    Occupation VARCHAR(255),
    Income INT,
    Dependent_Members INT,
    Financial_Support_Method VARCHAR(255),
    Properties_Owned INT,
    Patient_ID INT,
    FOREIGN KEY (Patient_ID) REFERENCES Patient(Patient_ID)
);
""")

clean_df_patient_financials.to_sql(name="Patient_Financials", con=conn,
    if_exists="replace", index=False)       # insert data into table

# Financial Aid table
conn.execute("""
CREATE TABLE IF NOT EXISTS Financial_Aid (
    Financial_Aid_ID INT PRIMARY KEY,
```

```
    Aid_Date DATE,
    Aid_Type VARCHAR(255),
    Aid_Amount INT,
    Financial_Info_ID INT,
    Zakat_Eligibility_ID INT,
    Patient_ID INT,
    FOREIGN KEY (Financial_Info_ID) REFERENCES Financial_Info(Financial_Info_ID),
    FOREIGN KEY (Zakat_Eligibility_ID) REFERENCES Zakat_Eligibility(Zakat_Eligibility_ID),
    FOREIGN KEY (Patient_ID) REFERENCES Patient(Patient_ID)
);
""")

clean_df_financial_aid.to_sql(name="Financial_Aid", con=conn, if_exists="replace", index=False)       # insert data into table

# Zakat Eligibility table
conn.execute("""
CREATE TABLE IF NOT EXISTS Zakat_Eligibility (
    Zakat_Eligibility_ID INT PRIMARY KEY,
    Is_Eligible BOOLEAN,
    Eligibility_Criteria VARCHAR(255),
    Patient_ID INT, Financial_Aid_ID INT,
    FOREIGN KEY (Patient_ID) REFERENCES Patient(Patient_ID),
    FOREIGN KEY (Financial_Aid_ID) REFERENCES Financial_Aid(Financial_Aid_ID)
);
""")

clean_df_zakat_eligibility.to_sql(name="Zakat_Eligibility", con=conn, if_exists="replace", index=False)       # insert data into table

# Export database out of colab
db_file = '/content/Cleaned_DPCA.db'

#files.download(db_file) # download the file (commented out to prevent unintended downloads)

conn.close()
```

Listing 4: Exporting Cleaned Database

A.5 SQL Queries and Reports

```

# Create connection to new database
conn = sqlite3.connect("Cleaned_DPCA.db")

if not os.path.exists('reports'):
    os.makedirs('reports')

# Define SQL queries
queries = {
    # Procedure History Report
    'procedure_history': """
    SELECT
```

```
        p.Patient_ID,
        COALESCE(p.Patient_Name, 'Unknown') as Patient_Name,
        COALESCE(proc.Procedure_Type, 'Unknown') as Procedure_Type,
        COALESCE(proc.Procedure_Date, 'No Date') as Procedure_Date,
        COALESCE(proc.Procedure_Cost, 0) as Procedure_Cost,
        COALESCE(proc.Amount_Paid_DPCA, 0) as Amount_Paid_DPCA,
        COALESCE(proc.Procedure_Cost, 0) - COALESCE(proc.Amount_Paid_DPCA, 0) as Patient_Contribution,
        COALESCE(proc.MR_number, 'Not Provided') as MR_number
    FROM
        Patient p
    JOIN
        Procedure proc ON p.Patient_ID = proc.Patient_ID
    WHERE
        proc.Procedure_Cost IS NOT NULL
        AND proc.Procedure_Type IS NOT NULL
        AND proc.Procedure_Type != 'Unknown'
    ORDER BY
        p.Patient_ID, proc.Procedure_Date
    """,

    # Financial Aid Report
    'financial_aid': """
    SELECT
        p.Patient_ID,
        COALESCE(p.Patient_Name, 'Unknown') as Patient_Name,
        fa.Aid_Date,
        COALESCE(fa.Aid_Type, 'Not Specified') as Aid_Type,
        COALESCE(proc.Amount_Paid_DPCA, 0) as Aid_Amount,
        COALESCE(pf.Income, 0) as Monthly_Income,
        CASE
            WHEN COALESCE(pf.Income, 0) > 0
            THEN ROUND((COALESCE(proc.Amount_Paid_DPCA, 0) * 100.0 / pf.Income), 2)
            ELSE 0
        END as Aid_Percentage_of_Income,
        COALESCE(proc.Procedure_Type, 'Not Specified') as Procedure_Type
    FROM
        Patient p
    JOIN
        Financial_Aid fa ON p.Patient_ID = fa.Patient_ID
    LEFT JOIN
        Patient_Financials pf ON p.Patient_ID = pf.Patient_ID
    LEFT JOIN
        Procedure proc ON p.Patient_ID = proc.Patient_ID
    WHERE
        fa.Aid_Date IS NOT NULL
        AND proc.Amount_Paid_DPCA IS NOT NULL
    ORDER BY
        p.Patient_ID, fa.Aid_Date
    """,

    # Zakat Eligibility Report
    'zakat_eligibility': """
    SELECT
        p.Patient_ID,
        COALESCE(p.Patient_Name, 'Unknown') as Patient_Name,
```

```
        COALESCE(p.Religion, 'Not Specified') as Religion,
        COALESCE(p.Syed_NonSyed, 'Not Specified') as Syed_NonSyed,
        ze.Is_Eligible,
        COALESCE(ze.Eligibility_Criteria, 'Not Specified') as
    Eligibility_Criteria,
        COALESCE(pf.Income, 0) as Monthly_Income,
        COALESCE(pf.Properties_Owned, 0) as Properties_Owned,
        fa.Aid_Type
    FROM
        Patient p
    JOIN
        Zakat_Eligibility ze ON p.Patient_ID = ze.Patient_ID
    LEFT JOIN
        Patient_Financials pf ON p.Patient_ID = pf.Patient_ID
    LEFT JOIN
        Financial_Aid fa ON p.Patient_ID = fa.Patient_ID
    WHERE
        (fa.Aid_Type = 'Zakat Funds' OR ze.Is_Eligible = 1
         OR ze.Is_Eligible = '1')
        AND ze.Eligibility_Criteria IS NOT NULL
        AND ze.Eligibility_Criteria != 'N.A'
    ORDER BY
        p.Patient_ID;
    """,

    # Dependency and Financial Burden Report
    'dependency_financial': """
    SELECT
        p.Patient_ID,
        COALESCE(p.Patient_Name, 'Unknown') as Patient_Name,
        COALESCE(pf.Income, 0) as Monthly_Income,
        COALESCE(pf.Dependent_Members, 0) as Dependent_Members,
        CASE
            WHEN COALESCE(pf.Dependent_Members, 0) > 0
            THEN ROUND(CAST(COALESCE(pf.Income, 0) AS FLOAT) / pf.
    Dependent_Members, 2)
            ELSE COALESCE(pf.Income, 0)
        END as Income_Per_Dependent,
        COALESCE(pf.Financial_Support_Method, 'Not Specified') as
    Financial_Support_Method,
        COALESCE(pf.Properties_Owned, 0) as Properties_Owned,
        COUNT(proc.Procedure_ID) as Total_Procedures,
        COALESCE(SUM(proc.Procedure_Cost), 0) as Total_Medical_Costs
    FROM
        Patient p
    JOIN
        Patient_Financials pf ON p.Patient_ID = pf.Patient_ID
    LEFT JOIN
        Procedure proc ON p.Patient_ID = proc.Patient_ID
    GROUP BY
        p.Patient_ID, p.Patient_Name, pf.Income, pf.Dependent_Members,
        pf.Financial_Support_Method, pf.Properties_Owned
    HAVING
        Total_Medical_Costs > 0
    ORDER BY
        Income_Per_Dependent
    """,

```

```
    # Occupation-Based Financial Analysis Report
    'occupation_analysis': """
    SELECT
        COALESCE(pf.Occupation, 'Not Specified') as Occupation,
        COUNT(DISTINCT p.Patient_ID) as Total_Patients,
        ROUND(AVG(COALESCE(pf.Income, 0)), 2) as Average_Income,
        ROUND(AVG(COALESCE(pf.Dependent_Members, 0)), 2) as
    Average_Dependents,
        ROUND(AVG(COALESCE(proc.Amount_Paid_DPCA, 0)), 2) as
    Average_Aid_Amount,
        COUNT(proc.Procedure_ID) as Total_Procedures,
        ROUND(AVG(COALESCE(proc.Procedure_Cost, 0)), 2) as
    Average_Procedure_Cost,
        ROUND(SUM(COALESCE(proc.Amount_Paid_DPCA, 0)), 2) as
    Total_Aid_Given
    FROM
        Patient_Financials pf
    JOIN
        Patient p ON pf.Patient_ID = p.Patient_ID
    LEFT JOIN
        Financial_Aid fa ON p.Patient_ID = fa.Patient_ID
    LEFT JOIN
        Procedure proc ON p.Patient_ID = proc.Patient_ID
    WHERE
        pf.Occupation IS NOT NULL
        AND pf.Occupation != ''
    GROUP BY
        pf.Occupation
    HAVING
        COUNT(DISTINCT p.Patient_ID) > 0
    ORDER BY
        Total_Aid_Given DESC
    """,

    # Patient Demographic Report
    'patient_demographics': """
    SELECT
        p.Patient_ID,
        p.Patient_Name,
        p.Age,
        p.Religion,
        p.Marriage_Status,
        p.Contact_No,
        p.Address
    FROM
        Patient p
    ORDER BY
        p.Patient_ID
    """,

    # Financial Overview Report
    'financial_overview': """
    SELECT
        p.Patient_ID,
        pf.Occupation,
        pf.Income,
        pf.Dependent_Members,
        pf.Properties_Owned,
```

```
        pf.Financial_Support_Method
    FROM
        Patient_Financials pf
    JOIN
        Patient p ON pf.Patient_ID = p.Patient_ID
    ORDER BY
        p.Patient_ID
    """,

    # Patient Financial Summary
    'patient_financial_summary': """
    SELECT
        p.Patient_ID,
        pf.Income,
        pf.Dependent_Members,
        pf.Financial_Support_Method,
        pf.Properties_Owned,
        proc.Procedure_Cost,
        proc.Amount_Paid_DPCA,
        fa.Aid_Date,
        fa.Aid_Type
    FROM
        Patient p
    JOIN
        Patient_Financials pf ON p.Patient_ID = pf.Patient_ID
    LEFT JOIN
        Financial_Aid fa ON p.Patient_ID = fa.Patient_ID
    LEFT JOIN
        Procedure proc ON p.Patient_ID = proc.Patient_ID
    ORDER BY
        p.Patient_ID
    """,

    # Monthly Aid Distribution Report
    'monthly_aid_distribution': """
    SELECT
        strftime('%Y-%m', fa.Aid_Date) AS month,
        fa.Aid_Type,
        SUM(proc.Amount_Paid_DPCA) AS total_amount,
        SUM(proc.Amount_Paid_DPCA) / COUNT(DISTINCT fa.Financial_Aid_ID
    ) AS average_aid_per_patient
    FROM
        Financial_Aid fa
    JOIN
        Procedure proc ON fa.Patient_ID = proc.Patient_ID
    GROUP BY
        month, fa.Aid_Type
    ORDER BY
        month, fa.Aid_Type
    """,

    # Patient Contact Information Report
    'patient_contact_information': """
    SELECT
        p.Patient_ID,
        p.Patient_Name,
        p.Contact_No,
        p.Address
```

```
    FROM
        Patient p
    ORDER BY
        p.Patient_ID
    """,

    # Patient Aid Dependence Ratio Report
    'patient_aid_dependence_ratio': """
    SELECT
        p.Patient_ID,
        proc.Amount_Paid_DPCA,
        pf.Income,
        proc.Amount_Paid_DPCA / NULLIF(pf.Income, 0) AS
   aid_to_income_ratio
    FROM
        Patient p
    JOIN
        Patient_Financials pf ON p.Patient_ID = pf.Patient_ID
    LEFT JOIN
        Procedure proc ON p.Patient_ID = proc.Patient_ID
    ORDER BY
        p.Patient_ID
    """,

    # Procedure Cost Analysis Report
    'procedure_cost_analysis': """
    SELECT
        proc.Procedure_Type,
        AVG(proc.Procedure_Cost) as average_procedure_cost
    FROM
        Procedure proc
    GROUP BY
        proc.Procedure_Type
    ORDER BY
        average_procedure_cost
    """
}


# Function to generate reports
def generate_reports():

    conn = sqlite3.connect("Cleaned_DPCA.db")
    timestamp = datetime.now().strftime('%Y%m%d_%H%M%S')

    for report_name, query in queries.items():
        try:
            # Execute query
            df = pd.read_sql_query(query, conn)

            # Save to CSV
            filename = f'reports/DPCA_{report_name}_report_{timestamp}.
   csv'
            df.to_csv(filename, index=False)

            # Print summary
            print(f"\nReport generated: {report_name}")
            print(f"Total records: {len(df)}")
```

```
            print(f"Saved to: {filename}")
            print("-" * 50)

            if len(df) > 0:
                # Display preview
                print("\nFirst few records:")
                print(df.head())

                # Generate summary statistics for numerical columns
                numeric_cols = df.select_dtypes(include=[np.number]).
    columns
                if len(numeric_cols) > 0:
                    print("\nSummary statistics:")
                    print(df[numeric_cols].describe())
            else:
                print(f"Warning: No records found for {report_name}
    report!")

        except Exception as e:
            print(f"Error generating {report_name} report: {str(e)}")
            print(f"Query used: {query}")  # Print the query for
    debugging

    conn.close()

# Execute report generation
generate_reports()


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

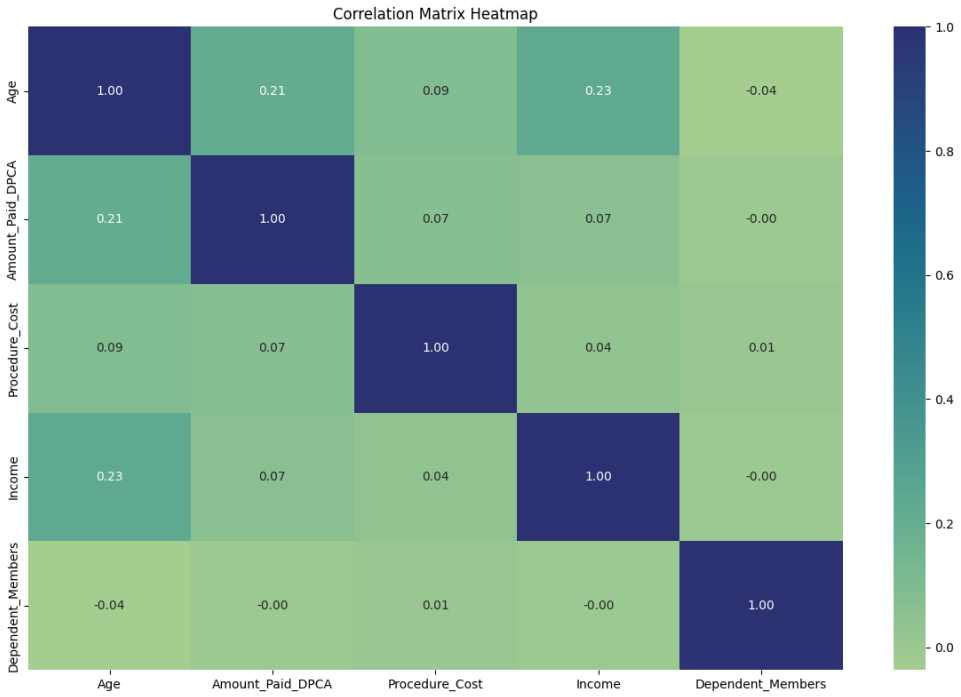

Figure 13: Correlation Matrix Heatmap

B.2 Procedure Types by Occupation

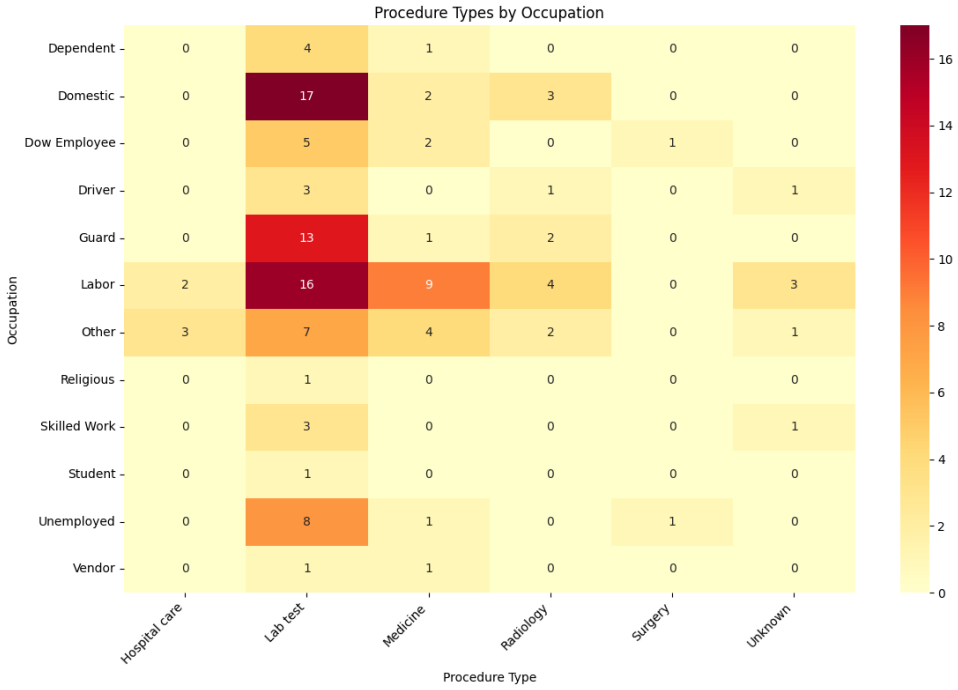

Figure 14: Procedure Types by Occupation

B.3 Procedure Types by Occupation pc

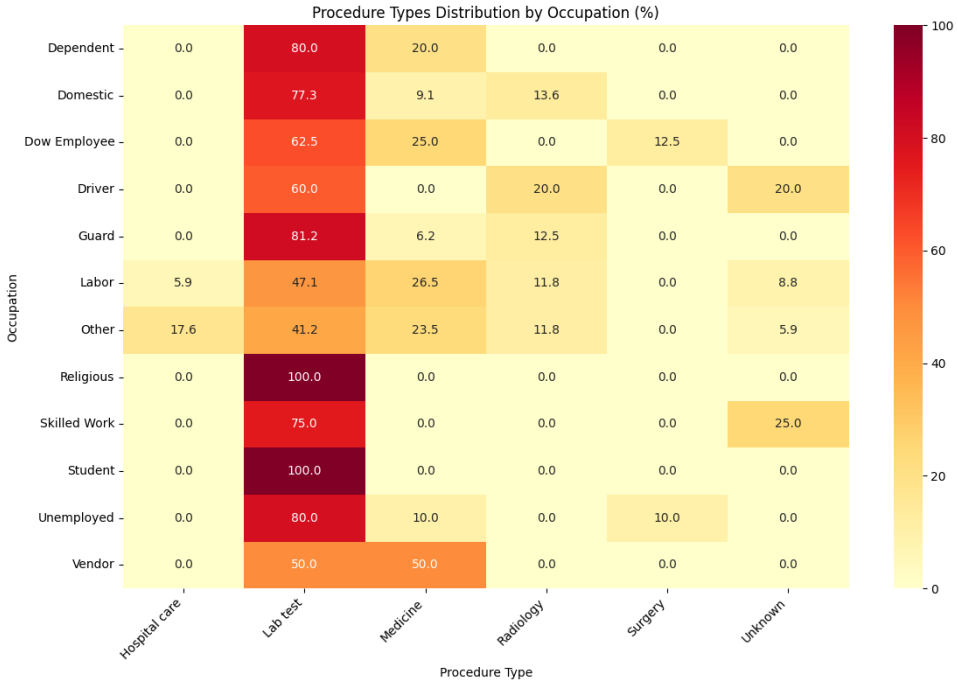

Figure 15: Procedure Types by Occupation pc