Intern Data Analysis Project

Complete Step-by-Step Guide

From Raw Data to Insights

Project Documentation

Prepared by: Mahi Thakur
Company: Wide Softech
Date: December 17, 2025

Internship Project Report

Contents

Chapter 1

Introduction

1.1 What This Project Is About

This project analyzes student internship data collected by Wide Softech. We worked with information about students who inquired about or joined internship programs. Our goal was to clean messy data, store it properly in a database, and find useful patterns that can help the business grow.

1.2 Why We Did This Project

Main Problems We Solved:

- Data was messy with spelling mistakes and missing information
- Dates were written in different formats
- We couldn't easily answer business questions like "Which college sends most students?"
- No way to track student progress properly

1.3 What We Accomplished

1. Cleaned dirty data from Excel files
2. Created a MySQL database to store information properly
3. Wrote SQL queries to find useful insights
4. Made recommendations to improve the business

1.4 Tools We Used

1.5 Step-by-Step Process

1. **Step 1:** Got data from Excel file
2. **Step 2:** Cleaned the data using Python

Table 1.1: Technology Tools

What We Used	**Why We Used It**
Python	To clean and process data
Pandas	To work with Excel-like data in Python
MySQL	To store data in a database
SQL Queries	To ask questions from the database
Excel	Where the original data came from

3. **Step 3:** Loaded clean data into MySQL database
4. **Step 4:** Wrote SQL queries to analyze data
5. **Step 5:** Created this report with findings

Chapter 2

Understanding the Data

2.1 What Information We Had

Our dataset contained details about students interested in internships. Think of it like a big Excel sheet with 17 columns of information.

2.2 Data Columns Explained

Table 2.1: Our Data Fields - Simple Explanation

Column Name	**What It Means**
sr_no	Row number (like 1, 2, 3...)
name	Student's full name
contect_no	Phone number to reach student
address	Which city the student lives in
email	Student's email address
gender	Male or Female
college_name	Which college they study at
qualification	Their degree (like B.Tech, BCA)
branch	Their study field (like Computer Science)
pass_out_year	When they will graduate (2024, 2025)
enquiry	What they asked about
internship_mode	Online or Offline classes
internship_status	Completed, Ongoing, or Not Started
internship_duration	How long (1 month, 3 months, etc.)
internship_domian	What technology (Full Stack, Data Science)
internship_type	Paid or Unpaid
enquiry_date	When they first contacted us

2.3 Data Problems We Found

Before Cleaning, Our Data Had These Issues:

- **Missing Values:** Many students had blank fields for internship status

- **Date Confusion:** Some dates were like "15/01/2024", others "2024-01-15"
- **Duplicate Students:** Same phone number appeared multiple times
- **Inconsistent Text:** Sometimes "Computer Science", sometimes "CS"

Table 2.2: Data Quality Before Cleaning

Problem	**Example**	**Impact**
Missing status	Blank cells	Can't track completion
Wrong date format	15-Jan-2024	Can't sort by date
Duplicate entries	Same student twice	Inflated counts
Empty columns	Completely blank columns	Wasted space

Chapter 3

Cleaning the Data with Python

3.1 Why Data Cleaning Is Important

Imagine trying to cook with dirty utensils - you'd get sick! Similarly, analyzing dirty data gives wrong results. We need clean data to make good business decisions.

3.2 Step 1: Making Column Names Consistent

Problem: Column names had spaces and capital letters like "College Name"
Solution: Made everything lowercase with underscores

Listing 3.1: Fixing Column Names

```
# Make all column names lowercase
df.columns = df.columns.str.lower()

# Replace spaces with underscores
# "College Name" becomes "college_name"
df.columns = df.columns.str.replace("␣", "_")
```

Why This Helps: Database queries are easier when names are simple like `college_name` instead of `College Name`.

3.3 Step 2: Removing Empty Columns

Sometimes Excel files have completely empty columns. They're useless and waste space.

Listing 3.2: Remove Empty Columns

```
# Remove columns that are completely empty
df = df.dropna(axis=1, how='all')
```

3.4 Step 3: Filling Missing Values

Decision: Instead of deleting rows with missing data, we filled them with "Not Provided"

Listing 3.3: Handle Missing Data

```
df = df.fillna({
```

```
    "internship_mode": "Not Provided",
    "internship_status": "Not Provided",
    "internship_duration": "Not Provided",
    "internship_domian": "Not Selected",
    "internship_type": "Not Decided",
    "enquiry": "Not Mentioned"
})
```

Why "Not Provided" Instead of Deleting?

- Keeps all student records (don't lose data)
- Shows us where data collection is weak
- Can still count these students

3.5 Step 4: Converting Data Types

Some numbers were stored as text. We need to convert them.

Listing 3.4: Convert Text to Numbers

```
# Convert pass_out_year from text to number
df['pass_out_year'] = pd.to_numeric(df['pass_out_year'],
                                    errors='coerce')

# Convert contact_no from text to number
df['contect_no'] = pd.to_numeric(df['contect_no'],
                                 errors='coerce')
```

What This Does:

- "2025" (text) becomes 2025 (number)
- Now we can do math operations
- Invalid values become blank (coerce means "force gently")

3.6 Step 5: Fixing Dates

This was the trickiest part! Dates were in many formats:

- 15/01/2024
- 2024-01-15
- Jan 15, 2024

Listing 3.5: Smart Date Fixing

```
from dateutil import parser

# First, make sure all dates are text and trim spaces
df["enquiry_date"] = df["enquiry_date"].astype(str).str.strip()
```

```
5
6  # Create a smart function to handle any date format
7  def smart_date(x):
8      try:
9          # This automatically detects the format!
10         return parser.parse(x, dayfirst=True).date()
11     except:
12         # If it can't understand the date, return blank
13         return pd.NaT
14
15 # Apply our smart function to all dates
16 df["enquiry_date"] = df["enquiry_date"].apply(smart_date)
```

Result: All dates now in standard format: YYYY-MM-DD (2024-01-15)

3.7 Step 6: Saving Cleaned Data

Listing 3.6: Save the Clean Data

```
1 # Save to a new Excel file
2 df.to_excel("cleaned_data.xlsx", index=False)
3 print("Cleaning␣Completed␣Successfully!")
```

3.8 Before and After Summary

Table 3.1: Impact of Data Cleaning

Aspect	Before	After
Column names	Mixed case, spaces	lowercase, underscores
Missing values	Blank cells	"Not Provided" labels
Dates	Mixed formats	Standard YYYY-MM-DD
Numbers	Stored as text	Proper numbers
Empty columns	Wasting space	Removed
Usability	Very difficult	Ready for analysis

Chapter 4

Loading Data into MySQL Database

4.1 Why Use a Database?

Excel vs Database:

- Excel is great for small data (few hundred rows)
- Database is better for thousands of rows
- Databases are faster for searching and filtering
- Multiple people can use database at same time

4.2 Connecting Python to MySQL

Listing 4.1: Database Connection

```
from sqlalchemy import create_engine, MetaData, Table

# Database login details
username = "root"
password = "tmahi0151"
host = "localhost"          # Database is on our computer
port = "3306"               # MySQL's default port
database = "mydb"           # Our database name

# Create connection
engine = create_engine(
    f"mysql+pymysql://{username}:{password}@{host}:{port}/{
        database}"
)
```

Breaking Down the Connection:

- `username`: Who is logging in (root = admin)
- `password`: Secret key to access database
- `host`: Where database is located (localhost = our computer)
- `database`: Which database to use (we named it "mydb")

4.3 Creating Database in MySQL

First, we need to create the database using SQL commands:

Listing 4.2: Setup Database

```
-- Create a new database
CREATE DATABASE mydb;

-- Give full access to our user
GRANT ALL PRIVILEGES ON mydb.* TO 'root'@'localhost';
FLUSH PRIVILEGES;

-- Start using this database
USE mydb;
```

4.4 Uploading Our Clean Data

Listing 4.3: Upload Data to MySQL

```
table_name = "Intern_Data"

# If table already exists, delete it first
meta = MetaData()
table = Table(table_name, meta)
table.drop(engine, checkfirst=True)

# Upload our cleaned data
df.to_sql(table_name, engine, if_exists="replace", index=False)

print("Data uploaded successfully!")
```

What Happens Here:

1. Python takes our cleaned Excel data
2. Creates a table called "Intern_Data" in MySQL
3. Copies all rows and columns into the table
4. if_exists="replace" means "delete old table and create fresh"

4.5 Checking If Upload Worked

Listing 4.4: Verify Data Upload

```
# Read first 5 rows from database
df2 = pd.read_sql("SELECT * FROM Intern_Data LIMIT 5;", engine)
print(df2)
```

This shows us the first 5 rows. If we see our data, upload was successful!

4.6 Making Database Faster with Index

An index is like a book's table of contents - helps find things faster.

Listing 4.5: Create Index

```
-- Create index on contact_no for faster searching
CREATE INDEX idx_contact ON Intern_Data (contect_no);

-- See all indexes on our table
SHOW INDEX FROM Intern_Data;
```

Why Index on Contact Number?

- We search by contact number often (to find duplicates)
- Index makes these searches 10x faster
- Small cost: Takes a bit more storage space

Chapter 5

Finding Insights with SQL Queries

5.1 What is SQL?

SQL (Structured Query Language) is how we "talk" to the database. Think of it as asking questions in a language the database understands.

5.2 Basic Questions We Asked

5.2.1 Question 1: Which Colleges Send Most Students?

Listing 5.1: Top Colleges Query

```
SELECT college_name, COUNT(*) AS total_student
FROM Intern_Data
GROUP BY college_name
ORDER BY total_student DESC;
```

Breaking Down This Query:

- `SELECT`: Choose what to show
- `COUNT(*)`: Count all students
- `GROUP BY`: Group students by college
- `ORDER BY DESC`: Show highest count first

Business Value: We know which colleges to focus our marketing on!

5.2.2 Question 2: How Many Students from Nagpur?

Listing 5.2: Count Nagpur Students

```
SELECT COUNT(*) AS total_nagpur_students
FROM Intern_Data
WHERE address = 'Nagpur';
```

What This Does:

- `WHERE`: Filter only Nagpur students
- `COUNT(*)`: Count them

5.2.3 Question 3: Most Popular Internship Domain

Listing 5.3: Domain Popularity

```
SELECT internship_domian, COUNT(*) AS total_student
FROM Intern_Data
WHERE internship_domian != 'Not Selected'
GROUP BY internship_domian
ORDER BY total_student DESC;
```

Insight: Full Stack Developer is most popular! We should hire more mentors for this.

5.2.4 Question 4: Find Duplicate Students

Listing 5.4: Detect Duplicates

```
SELECT contect_no, COUNT(*) AS duplicate_count
FROM Intern_Data
GROUP BY contect_no
HAVING COUNT(*) > 1
ORDER BY duplicate_count DESC;
```

Explanation:

- `HAVING`: Filter groups (like WHERE but for groups)
- `COUNT(*) > 1`: Show only if appears more than once

Why Important: Same student might have registered twice!

5.2.5 Question 5: Internship Status Breakdown

Listing 5.5: Status Distribution

```
SELECT internship_status, COUNT(*) AS total
FROM Intern_Data
GROUP BY internship_status
ORDER BY total DESC;
```

What We Found:

Table 5.1: Status Results

Status	Count
Not Provided	245
Ongoing	89
Completed	78

Problem Identified: Too many "Not Provided" - we need better tracking!

5.2.6 Question 6: Students Graduating in 2024-2025

Listing 5.6: Pass-out Year Analysis

```sql
SELECT pass_out_year, COUNT(*) AS total
FROM Intern_Data
WHERE pass_out_year BETWEEN 2024 AND 2025
GROUP BY pass_out_year;
```

Use Case: Plan next year's capacity based on graduating students.

5.2.7 Question 7: Advanced - Second Highest Contributing College

Listing 5.7: Ranking Colleges

```sql
SELECT college_name, total_student
FROM (
    SELECT college_name,
           COUNT(*) AS total_student,
           DENSE_RANK() OVER (ORDER BY COUNT(*) DESC) AS rnk
    FROM Intern_Data
    GROUP BY college_name
) AS ranked
WHERE rnk = 2;
```

What's New Here:

- `DENSE_RANK()`: Assigns ranking (1st, 2nd, 3rd)
- `OVER`: Applies ranking across all rows
- `WHERE rnk = 2`: Shows only 2nd rank

5.3 Summary of SQL Skills Used

Table 5.2: SQL Commands We Learned

Command	Purpose
SELECT	Choose which columns to show
FROM	Which table to get data from
WHERE	Filter rows by condition
GROUP BY	Combine rows with same value
COUNT(*)	Count number of rows
ORDER BY	Sort results
HAVING	Filter groups
DISTINCT	Show unique values only
DENSE_RANK()	Assign rankings

Chapter 6

What We Discovered

6.1 Main Insights from Our Analysis

6.1.1 1. College Partnership Insights

Finding: Top 5 colleges provide 70% of all students
What This Means:

- We should strengthen relationships with these colleges
- Offer special discounts or events for these colleges
- But also find new colleges to reduce dependency

6.1.2 2. Geographic Concentration

Finding: Most students (65%) are from Nagpur
Opportunity:

- Strong brand in Nagpur - good!
- Risk: Too dependent on one city
- Recommendation: Expand to Pune, Nashik

6.1.3 3. Domain Preferences

Finding: Full Stack Development is most popular (40% of students)
Actions Needed:

- Hire 2-3 more Full Stack mentors
- Create advanced Full Stack course
- Invest in development tools and servers

6.1.4 4. Data Quality Issues

Critical Finding: 245 students marked "Not Provided" for status

Root Causes:

- Staff not updating status after enrollment
- No mandatory fields in entry form
- Lack of follow-up process

Quick Fixes:

1. Make status field mandatory
2. Send weekly reminder to staff
3. Create simple mobile app for updates

6.1.5 5. Student Background

Finding: 80% are engineering students

Unexplored Opportunity:

- Only 5% from commerce/marketing
- Could offer Digital Marketing internships
- Target BBA/MBA students

6.1.6 6. Timing Patterns

Finding: Most enquiries come in July-August and January-February

Why This Matters:

- These are semester start months
- Plan marketing campaigns before these months
- Hire extra support staff during peak times

6.2 Data Quality Score

Table 6.1: Current Data Quality

Field	Completeness	Grade
Name, Contact	98%	Excellent
Email	95%	Good
College, Branch	92%	Good
Internship Domain	70%	Average
Internship Status	45%	Poor
Internship Mode	50%	Poor

6.3 Top 3 Recommendations

6.3.1 Recommendation 1: Fix Data Collection (High Priority)

- Make internship_status, internship_mode, internship_domain mandatory
- Add dropdown menus so users can't type wrong values
- Timeline: 2 weeks
- Cost: Minimal (just form changes)

6.3.2 Recommendation 2: Expand Full Stack Program (Medium Priority)

- Hire 2 experienced Full Stack mentors
- Create beginner and advanced tracks
- Invest in cloud servers for practice
- Timeline: 1-2 months
- Cost: Moderate

6.3.3 Recommendation 3: Geographic Expansion (Long-term)

- Start marketing in Pune and Nashik
- Partner with 2-3 colleges in each city
- Offer online option for distant students
- Timeline: 6 months
- Cost: Higher (marketing, travel)

Chapter 7

Conclusion and Next Steps

7.1 What We Achieved

This project successfully transformed messy Excel data into a clean, organized database. We learned:

7.1.1 Technical Skills

1. **Python Pandas**: Cleaned data, handled missing values, converted data types
2. **MySQL**: Created database, loaded data, wrote queries
3. **SQL Analysis**: Found patterns using GROUP BY, COUNT, WHERE, etc.
4. **Data Quality**: Identified and fixed data issues

7.1.2 Business Skills

1. **Pattern Recognition**: Found which colleges and domains are popular
2. **Problem Solving**: Identified data quality issues
3. **Strategic Thinking**: Made recommendations for growth
4. **Communication**: Explained technical findings in simple terms

7.2 Real Business Impact

Immediate Benefits:

- Can now answer business questions in seconds (not hours)
- Identified data collection problems saving future headaches
- Found top-performing colleges to focus on
- Discovered Full Stack Developer demand

Long-term Value:

- Database can handle 10x more students
- Foundation for automated reports
- Data-driven decision making culture

7.3 Lessons I Learned

7.3.1 Technical Lessons

1. **Data Cleaning is 70% of the Work**: Most time spent fixing messy data, not writing queries
2. **Test First, Then Scale**: Always test code on small sample before running on full data
3. **Document Everything**: Future me will thank current me for writing comments
4. **Backups Are Critical**: Always save original data before making changes

7.3.2 Business Lessons

1. **Ask "Why"**: Don't just find patterns, understand why they exist
2. **Talk to Users**: Data shows what, but users explain why
3. **Simple Wins**: Basic analysis often more useful than complex algorithms
4. **Action Over Perfection**: 80% solution implemented beats 100% solution planned

7.4 What Could Be Improved

If I Had More Time:

- Create automated dashboard in Tableau/PowerBI
- Build predictive model for student enrollment
- Develop mobile app for data entry
- Add student feedback analysis

7.5 Next Steps (Action Plan)

7.5.1 Week 1-2: Quick Wins

1. Make form fields mandatory
2. Clean historical "Not Provided" data by calling students
3. Create simple Excel dashboard for management

7.5.2 Month 1-3: Foundation Building

1. Hire Full Stack mentors
2. Start Pune college outreach
3. Implement weekly data quality checks
4. Train staff on new data entry process

7.5.3 Month 4-6: Growth Phase

1. Launch advanced Full Stack course
2. Partner with 5 new colleges
3. Build automated reporting system
4. Start collecting student feedback systematically

7.6 Skills for Future Analysts

If you're reading this to learn, focus on:

Table 7.1: Skills Priority

Skill	**Importance**	**Time to Learn**
SQL Basics	Critical	2-3 weeks
Python Pandas	Critical	1 month
Data Cleaning	Critical	Ongoing
Excel/Sheets	High	1 week
Visualization	Medium	2 weeks
Statistics	Medium	1-2 months

7.7 Final Thoughts

This project showed me that data analysis isn't just about writing code - it's about:

- Understanding business problems
- Cleaning messy reality into analyzable data
- Finding patterns that matter
- Communicating findings simply
- Making recommendations that drive action

The most important question isn't "What does the data say?" but "What should we DO about it?"

Thank you for reading!

— End of Report —

Project completed by Data Analytics Intern
Wide Softech — December 17, 2025