A project report on

ETL ANALYSIS OF ANDHRA PRADESH'S HEALTHLANDSCAPE

Submitted in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

By

SPARSH RATHOUR (20BCI0264)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

APRIL,2024

DECLARATION

I hereby declare that the thesis entitled "ETL ANALYSIS OF ANDHRA PRADESH HEALTH'S LANDSCAPE" submitted by me, for the award of the degree of Bachelor Of Technology is a record of bonafide work carried out by me under the supervision of Lucky Mishra

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Vellore **Sparsh Rathour**

Date:23-04-2024 Signature of the Candidate

CHUBB

Chubb Business Services India Private Limited
(formerly known as Chubb Business Services India LLP)
Fifteenth floor, Unit No. 3, Parcel – 4, Octave Block,
Knowledge City, Plot No 2, Phase 1, Survey No.83/1,
Raidurg Village, Serilingampally Mandal,
Hyderabad – 500081

Date: Apr-24-2024

TO WHOMEVER IT MAY CONCERN

This is to certify that SPARSH RATHOUR is currently undergoing his internship at Chubb India, Hyderabad, from December 26, 2023, and tentatively until June 30, 2024.

During the internship, he has demonstrated diligence and enthusiasm in learning and delivering projects on time. SPARSH RATHOUR has successfully completed and delivered a capstone project as part of his internship.

We wish SPARSH RATHOUR the very best for his career and future endeavours.

For Chubb Business Services India Private Limited

[illegible]

Lucky Misra
Lead Campus Program

ABSTRACT

This project focuses on leveraging Informatica as an ETL (Extract, Transform, Load) tool to analyze a CSV file containing data about various hospitals in Andhra Pradesh. The dataset includes attributes such as age, sex, caste, surgery details, preauthorization and claim amounts, mortality status, and more. The main objective is to extract meaningful insights from the data to improve healthcare services and decision-making.The ETL process involves extracting the data from the CSV file, transforming it through cleaning, filtering, and formatting operations, and loading it into a target system. Data quality is ensured by handling missing or incorrect information and standardizing data formats.After loading the transformed data, statistical analysis and visualization techniques are applied to derive insights. The focus is on identifying trends in surgeries, analyzing preauthorization and claim amounts, examining mortality rates, and understanding other relevant factors. The obtained insights aim to inform policymakers, hospital administrators, and other stakeholders in making data-driven decisions for better healthcare services.The project also encompasses optimizing the ETL performance by employing parallel processing, database optimization, and utilizing Informatica's performance tuning capabilities.Ultimately, the project aims to generate a comprehensive report with key findings, trends, and patterns, contributing to the enhancement of healthcare services in Andhra Pradesh. The insights derived from this analysis can facilitate evidence-based decision-making, resource allocation optimization, and overall improvements in the healthcare landscape.

ACKNOWLEDGEMENT

It is my pleasure to express with deep sense of gratitude to Adarsh Singh, Senior Data Engineer in Chubb India and Lucky Mishra, Lead program Manger in Chubb India, for their constant guidance, continual encouragement, and understanding; more than all, he taught me patience in my endeavor. My association with him/her is not confined to academics only, but it is a great opportunity on my part to work with an intellectual and expert in the field of data.

I would like to express my gratitude to Vishwanathan G, Shankar Vishwanathan, Selvam, Dr.Ramesh Babu k, School of Computer Science and Engineering,for providing with an environment to work in and for his inspiration during the tenure of the course.

In a jubilant mood, I express ingeniously my whole-hearted thanks to Satiyaraj R(HOD infosec)all teaching staff and members working as limbs of our university for their not-self-centred enthusiasm coupled with timely encouragements showered on me with zeal, which prompted the acquirement of the requisite knowledge to finalize my course study successfully. I would like to thank my parents for their support.

It is indeed a pleasure to thank my friends who persuaded and encouraged me to take up and complete this task. Last but not least, I express my gratitude and appreciation to all those who have helped me directly or indirectly toward the successful completion of this project.

Place: Vellore **SPARSH RATHOUR**

Date: Name of the student

CONTENTS

::

::

::

LIST OF FIGURES

LIST OF ACRONYMS

1.	ETL	EXTRACT TRANSFORM LOAD
2.	STG	STAGING
3.	DIM	DIMENSION
4.	HOSP	HOSPITAL NAME
5.	SQL	STRUCTURED QUERY LANGUAGE

CHAPTER 1

1.1 INTRODUCTION

In today's data-driven world, the ability to extract meaningful insights from vast amounts of data is crucial. With the advancements in technology, tools like Informatica have emerged as powerful tools for performing ETL (Extract, Transform, Load) processes, enabling organizations to leverage their data effectively.

Our project focuses on utilizing Informatica to perform ETL on a CSV file that contains data about different hospitals in Andhra Pradesh. This dataset comprises various attributes such as serial number (sno), age, sex, caste, category code and name, surgery code and name, village, mandal (sub-district) name, district name, preauthorization date and amount, claim date and amount, hospital name and type, hospital district, surgery date, discharge date, and mortality status.

The goal of this project is to extract, transform, and load the data into a target system, such as a data warehouse or database, and derive valuable insights that can enhance decision-making and improve healthcare services in Andhra Pradesh. By analyzing this dataset, we can uncover trends and patterns related to surgeries, preauthorization and claim amounts, mortality rates, and other relevant factors.

Through the ETL process in Informatica, we will perform data cleansing, data transformation, and data enrichment tasks. This involves identifying and addressing missing or incorrect data, removing duplicates, standardizing data formats, and applying necessary calculations or aggregations. By ensuring the quality and consistency of the data, we can subsequently generate accurate insights and reports.

Once the data has been loaded, we will employ various analysis techniques, including statistical analysis and visualization, to gain a comprehensive understanding of the data. By examining the relationships between different attributes, we can uncover correlations and identify potential areas for improvement in the healthcare system. Furthermore, the generated insights can aid policymakers, hospital administrators, and other stakeholders in making informed decisions to enhance patient care and optimize resource allocation.

1.2 Scope

The scope of your project involves several key areas, including ETL process management, data modeling, data warehouse architecture, ETL tool development, and ETL testing. Here's a detailed breakdown of the scope based on the provided sources:

1. **ETL Process Management**: This involves outlining the ETL process, setting the boundaries of data processing, providing system architecture for each element and the whole data pipeline, documenting the system requirements, managing its development, and taking part in the actual

development/implementation of ETL tools. It also includes conducting testing of the tools and data pipelines .

2. **Data Modeling**: This involves defining data collection methods, data models, types, and outlining the transformation process. It's crucial for understanding the structure of the data and how it will be transformed and loaded into the target system .

3. **Data Warehouse Architecture**: The project will require designing and developing the data warehouse architecture, which includes the modeling, development, and maintenance of data storages. This is essential for ensuring that the data is stored in a way that supports efficient querying and analysis .

4. **ETL Tool Development:** This involves using ETL tools to automate the extraction, transformation, and loading of data from the source to the target system. The development of these tools is a critical part of the project, as it enables the automation of data processing tasks .

5. **ETL Testing:** Comprehensive testing of the ETL process, including data model testing, data warehouse architecture testing, representation tools check, data flow validation, uploading/downloading/querying speed testing, and system performance tests, is essential. This ensures that the data is correctly transformed and loaded into the target system and that the system performs optimally .

6. **Collaboration with Other Roles:** The project will likely involve collaboration with various roles, including data analysts, database/warehouse developers, DBAs, data scientists, and business intelligence developers. Each of these roles plays a crucial part in the overall ETL process, from defining data collection methods to developing and maintaining data storages and developing BI interfaces .

7. **Continuous Improvement and Maintenance:** The project will also involve ongoing maintenance and improvement of the ETL process, data models, and data warehouse architecture. This includes monitoring business requirements, conducting user interviews, and validating system designs and publication.

1.3 About the Dataset

It includes a wide range of information that spans various aspects of patient care, including demographic details, surgical procedures, and outcomes. Here's a breakdown of the dataset's components and their significance:

- **sno:** This is likely a serial number or unique identifier for each record, ensuring that each entry can be uniquely identified.
- **age:** The age of the patient, which is a crucial demographic factor that can influence healthcare outcomes and the type of care required.
- **sex:** The gender of the patient, which can impact healthcare outcomes and the selection of treatments.
- **caste_name:** This indicates the social or economic caste of the patient, which can influence access to healthcare services and outcomes.

- **category_code, category_name:** These fields likely categorize the patient's condition or the type of care they require, which is essential for planning and delivering appropriate healthcare services.
- **surgery_code, surgery:** These fields detail the surgical procedures performed on the patient, including the code for the procedure and its name. This is crucial for tracking the types of surgeries performed and their outcomes.
- **village, mandal_name, district_name:** These fields provide geographical information about the patient, which can be useful for planning healthcare services and understanding healthcare access and utilization patterns.
- **preauth_amt, claim_date, claim_amount:** These fields are related to the financial aspects of healthcare, including pre-authorization amounts, the date of the claim, and the total claim amount. This information is crucial for understanding healthcare costs and reimbursement patterns.
- **hosp_name, hosp_type, hosp_district:** These fields provide details about the hospital where the patient received care, including the name, type, and district of the hospital. This information is important for understanding the distribution of healthcare services and the quality of care provided.
- **surgery_date, discharge_date, mortality_date:** These fields detail the dates of key events in the patient's care, including the date of surgery, discharge from the hospital, and any mortality date. This information is crucial for tracking the timeline of care and outcomes.

1.4 Overview:

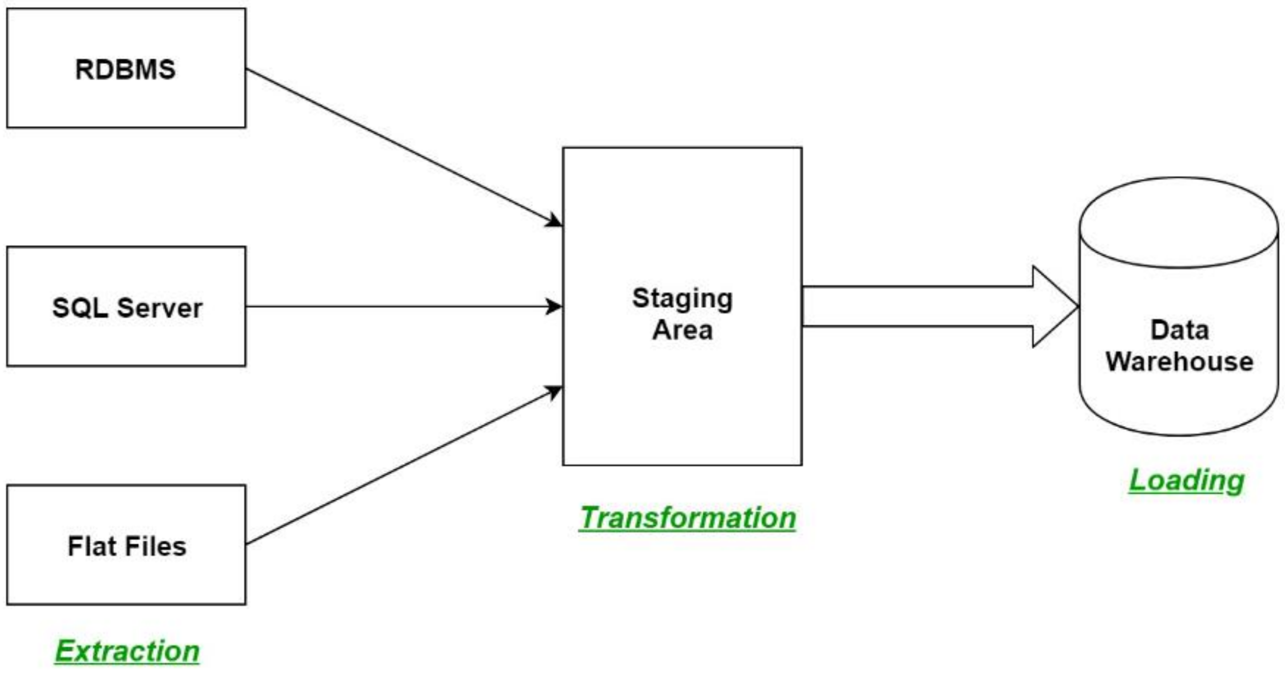


Fig 1

The project aims to transform customer experiences in the healthcare industry by synthesizing disparate data across the organization, thereby accelerating clinical and business processes while improving experiences for members, patients, and providers .

Informatica's approach to ETL (Extract, Transform, Load) in healthcare is centered around creating data pipelines that can ingest, enrich, transform, prepare, scale, and share data across a multi-cloud environment. This includes integrating with cloud services like Amazon Web Services, Microsoft Azure, Google Cloud, Snowflake, and Databricks, among others. The goal is to operationalize end-to-end data pipelines and modernize legacy applications for AI, ensuring that data is clean, trustworthy, and accessible for decision-making .

The project will utilize Informatica's industry-leading data integration tools and solutions, offering a codeless, AI-powered cloud-native data integration platform. This platform is designed to handle data at any volume, velocity, and latency, making it suitable for both data integration and data science initiatives. The project will also involve signing up for the free Informatica Cloud Data Integration trial to experience the broad out-of-the-box connectivity, prebuilt advanced transformations, and orchestrations that can help accelerate data pipelines .

In summary, the project aims to reinvent the future of healthcare data management by leveraging Informatica's ETL capabilities to integrate, transform, and load data across a multi-cloud environment. This approach is expected to accelerate clinical and business processes, improve patient and provider experiences, and ensure that all data practitioners have access to reliable data for decision-making

CHAPTER 2

2.1 Requirements

- Informatica ETL Tool
- SSMS(SQL Server Management System)
- Python
- VS CODE
- Spyder

2.2 About Informatica

Informatica is a leading provider of enterprise cloud data management solutions, focusing on helping businesses transform and manage their data to drive growth and innovation. The company offers a wide range of products and services designed to address the complexities of data integration, master data management (MDM), and data quality.

Core Capabilities and Products:

- Informatica PowerCenter: This is a comprehensive data integration tool that enables businesses to ingest, transform, and load data from various sources into a data warehouse or data lake. It supports a wide range of data formats and provides advanced transformation capabilities, including handling complex XML, industry formats (SWIFT, HL7, EDI X12), unstructured documents, PDFs, and Microsoft Office documents. PowerCenter also offers add-on packages for enhanced functionality, such as real-time connectivity, web services, Change Data Capture (CDC) support, and centralized management through a data integration hub

- Informatica MDM Hub: As part of Informatica's Master Data Management (MDM) solutions, the MDM Hub serves as the enterprise MDM platform. It provides core capabilities for managing and governing master data across the organization, ensuring data consistency, accuracy, and quality. This is crucial for businesses to maintain a single, trusted view of their data, which is essential for decision-making and operational efficiency

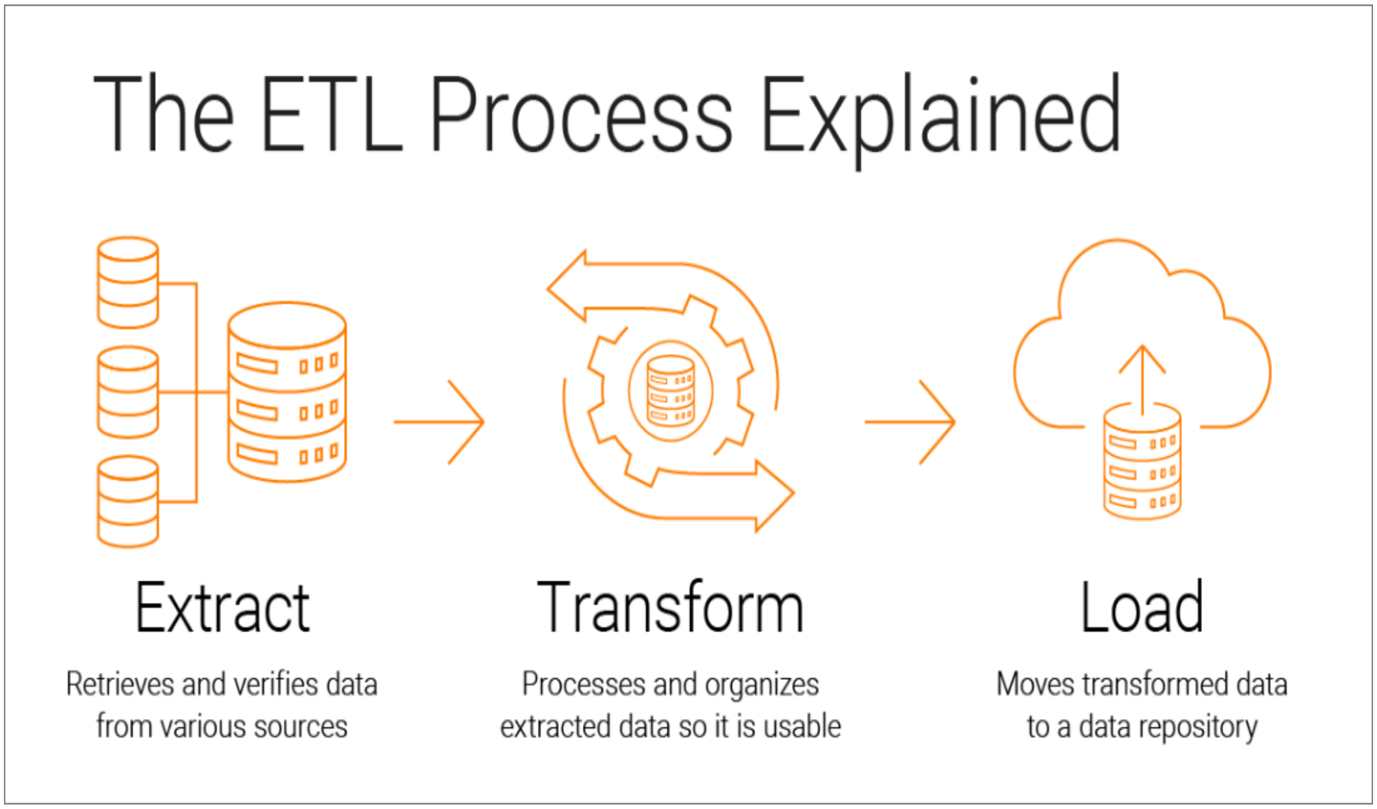

The ETL Process Explained
Extract
Retrieves and verifies data from various sources
Transform
Processes and organizes extracted data so it is usable
Load
Moves transformed data to a data repository

Fig 2 (ETL Process)

2.3 Data Warehouse Design Diagram

Creating a data warehouse diagram for the project involving Informatica's PowerCenter and its architecture requires understanding the components and their interactions within the system.

1. **Informatica Administrator:** This is a web application used for administering the domain and PowerCenter. It serves as the central point for managing the overall architecture and configuration of the data warehouse project.

2. **Informatica Domain:** The domain is the primary unit for management and administration of services in PowerCenter. It consists of one or more nodes, a service manager, and application services. The domain is crucial for organizing and controlling the resources and services used in the data warehouse project.

3. **Nodes:** Nodes are logical representations of machines within a domain. They can be configured to run application services like the integration service or repository service. The master gateway node hosts the domain and manages requests from other nodes.

4. **Service Manager:** The service manager supports the domain and application services. It runs on each node in the domain, starting and managing the application services.

5. **Application Services:** These are groups of services that represent the Informatica server-based functionality. Key application services include the PowerCenter repository service, integration service, and metadata manager service.

6. **PowerCenter Repository:** The repository stores metadata in a relational database. It contains the instructions for extracting, transforming, and loading data. The repository service accepts requests to create and modify metadata in the repository.

7. **PowerCenter Integration Service:** This service is responsible for extracting data from the source, transforming the data according to the instructions coded in the workflow, and loading the data into the targets. It plays a critical role in the ETL process within the data warehouse.

8. **Metadata Manager Service:** This service runs the metadata manager web application, allowing users to analyze metadata from various metadata repositories.

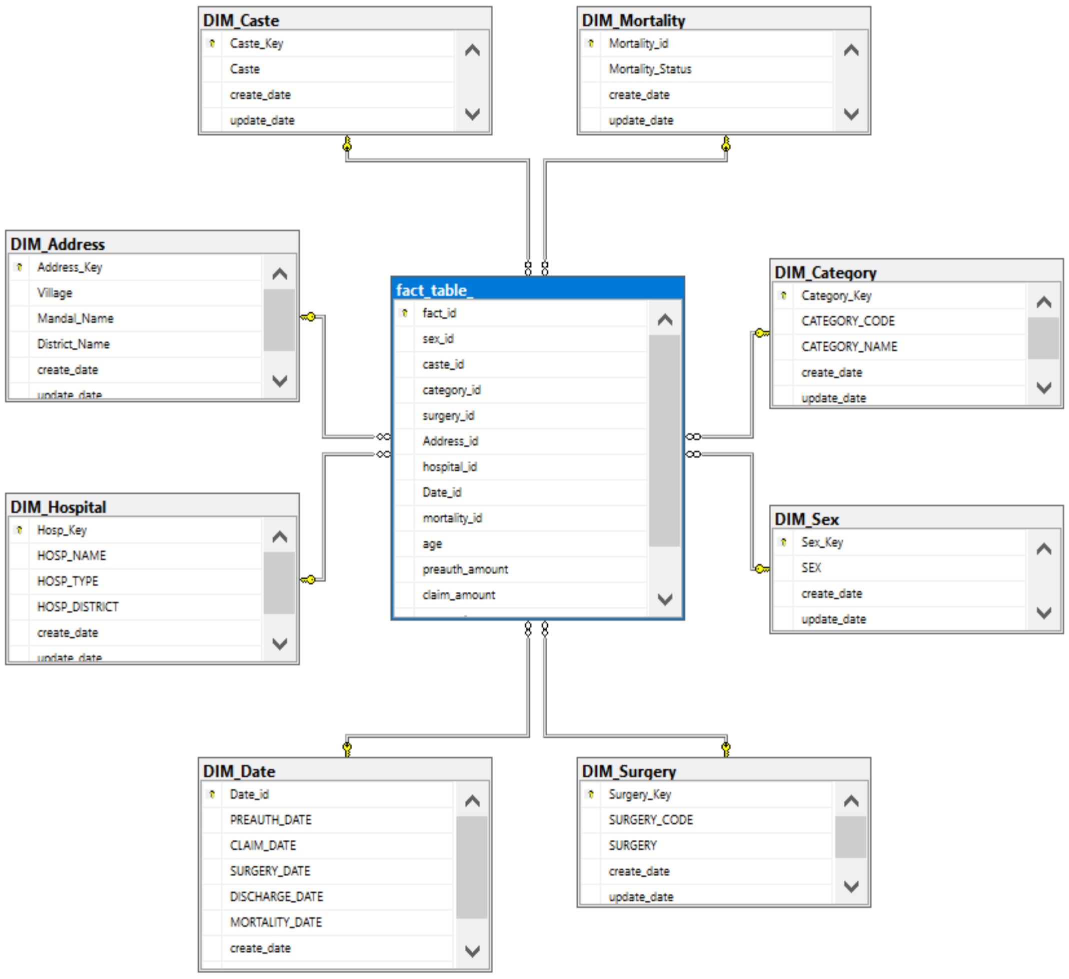


Fig 3 (Design Diagram)

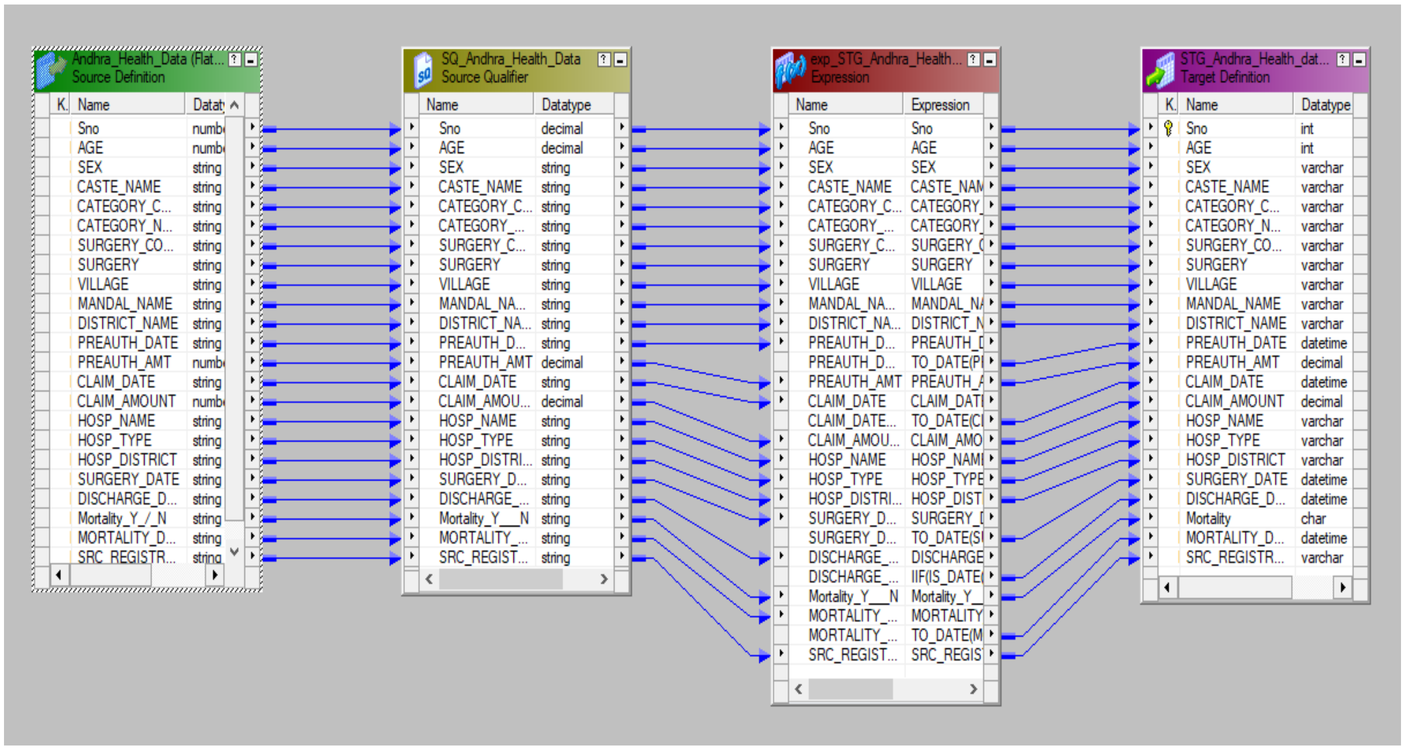
2.4 Stages of the project

Informatica ETL Stages for Andhra Health Data

To perform ETL tasks on your CSV file using Informatica, you'll follow a structured approach that involves reading data from the database into a CSV file, breaking down the data into dimension tables, and finally attaching them to a final fact table. Here's a detailed breakdown of the process:

Step 1: Data Extraction from Database to CSV

- Source Qualifier: Use the Source Qualifier transformation to define the database as your source. Specify the database connection details, the query to extract the data, and the output format as CSV.
- Target Qualifier: Define the CSV file as your target using the Target Qualifier transformation. Specify the file path and ensure the output format is set to CSV.
- Mapping: Create a mapping that connects the Source Qualifier to the Target Qualifier. This mapping will extract data from the database and write it to the CSV file.



Andhra_Health_Data (Flat...
Source Definition

K.	Name	Dataty
	Sno	numb
	AGE	numb
	SEX	string
	CASTE_NAME	string
	CATEGORY_C...	string
	CATEGORY_N...	string
	SURGERY_CO...	string
	SURGERY	string
	VILLAGE	string
	MANDAL_NAME	string
	DISTRICT_NAME	string
	PREAUTH_DATE	string
	PREAUTH_AMT	numb
	CLAIM_DATE	string
	CLAIM_AMOUNT	numb
	HOSP_NAME	string
	HOSP_TYPE	string
	HOSP_DISTRICT	string
	SURGERY_DATE	string
	DISCHARGE_D...	string
	Mortality_Y_/_N	string
	MORTALITY_D...	string
	SRC_REGISTR...	string

SQ_Andhra_Health_Data
Source Qualifier

Name	Datatype
Sno	decimal
AGE	decimal
SEX	string
CASTE_NAME	string
CATEGORY_C...	string
CATEGORY_...	string
SURGERY_C...	string
SURGERY	string
VILLAGE	string
MANDAL_NA...	string
DISTRICT_NA...	string
PREAUTH_D...	string
PREAUTH_AMT	decimal
CLAIM_DATE	string
CLAIM_AMOU...	decimal
HOSP_NAME	string
HOSP_TYPE	string
HOSP_DISTRI...	string
SURGERY_D...	string
DISCHARGE_...	string
Mortality_Y___N	string
MORTALITY_...	string
SRC_REGIST...	string

exp_STG_Andhra_Health...
Expression

Name	Expression
Sno	Sno
AGE	AGE
SEX	SEX
CASTE_NAME	CASTE_NAM
CATEGORY_C...	CATEGORY_
CATEGORY_...	CATEGORY_
SURGERY_C...	SURGERY_(
SURGERY	SURGERY
VILLAGE	VILLAGE
MANDAL_NA...	MANDAL_N/
DISTRICT_NA...	DISTRICT_N
PREAUTH_D...	PREAUTH_[
PREAUTH_D...	TO_DATE(PI
PREAUTH_AMT	PREAUTH_/
CLAIM_DATE	CLAIM_DATI
CLAIM_DATE...	TO_DATE(CI
CLAIM_AMOU...	CLAIM_AMO
HOSP_NAME	HOSP_NAMI
HOSP_TYPE	HOSP_TYPE
HOSP_DISTRI...	HOSP_DIST
SURGERY_D...	SURGERY_[
SURGERY_D...	TO_DATE(SI
DISCHARGE_...	DISCHARGE
DISCHARGE_...	IIF(IS_DATE(
Mortality_Y___N	Mortality_Y_
MORTALITY_...	MORTALITY
MORTALITY_...	TO_DATE(M
SRC_REGIST...	SRC_REGIS

STG_Andhra_Health_dat...
Target Definition

K.	Name	Datatype
	Sno	int
	AGE	int
	SEX	varchar
	CASTE_NAME	varchar
	CATEGORY_C...	varchar
	CATEGORY_N...	varchar
	SURGERY_CO...	varchar
	SURGERY	varchar
	VILLAGE	varchar
	MANDAL_NAME	varchar
	DISTRICT_NAME	varchar
	PREAUTH_DATE	datetime
	PREAUTH_AMT	decimal
	CLAIM_DATE	datetime
	CLAIM_AMOUNT	decimal
	HOSP_NAME	varchar
	HOSP_TYPE	varchar
	HOSP_DISTRICT	varchar
	SURGERY_DATE	datetime
	DISCHARGE_D...	datetime
	Mortality	char
	MORTALITY_D...	datetime
	SRC_REGISTR...	varchar

Fig 4 (Design Diagram)

Step 2: Breaking Down Data into Dimension Tables

- Source Qualifier: Use the Source Qualifier transformation again to define the CSV file as your source.
- Expression Transformation: Use the Expression transformation to clean or transform the data as needed. For example, you might want to convert dates to a standard format or split fields into multiple columns.
- Filter Transformation: Use the Filter transformation to exclude records that do not meet certain criteria.
- Sorter Transformation: Use the Sorter transformation to sort the data if necessary.
- Target Qualifier: For each dimension table (DIM_caste, DIM_Mortality, DIM_address, DIM_Category, DIM_SEX, DIM_Mortality, DIM_hospital, DIM_Date, DIM_SURGERY), use the Target Qualifier transformation to define the target table in your database. Specify the table name, connection details, and any other necessary settings.
- Mapping: Create a mapping for each dimension table that connects the Source Qualifier to the Target Qualifier. This mapping will load data into each dimension table.

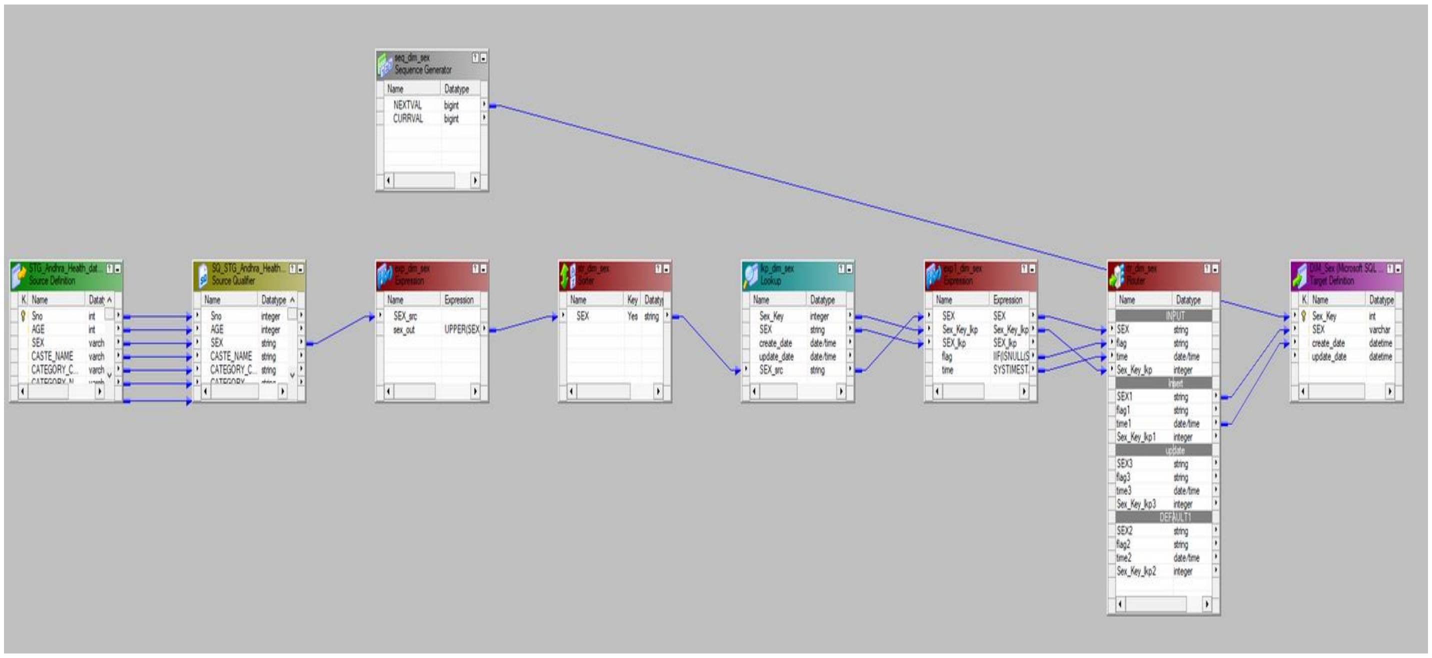


Fig 5(DIM_SEX)

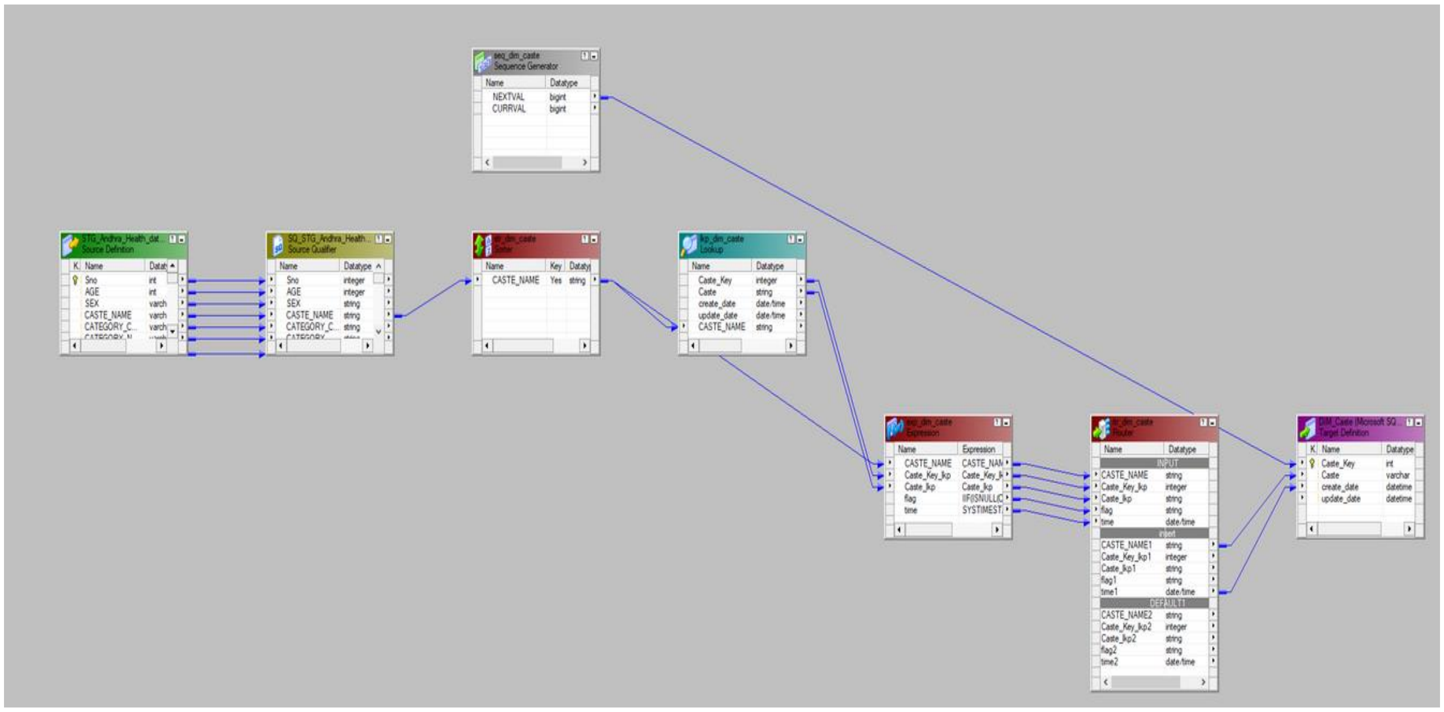


Fig 6(DIM_CASTE)

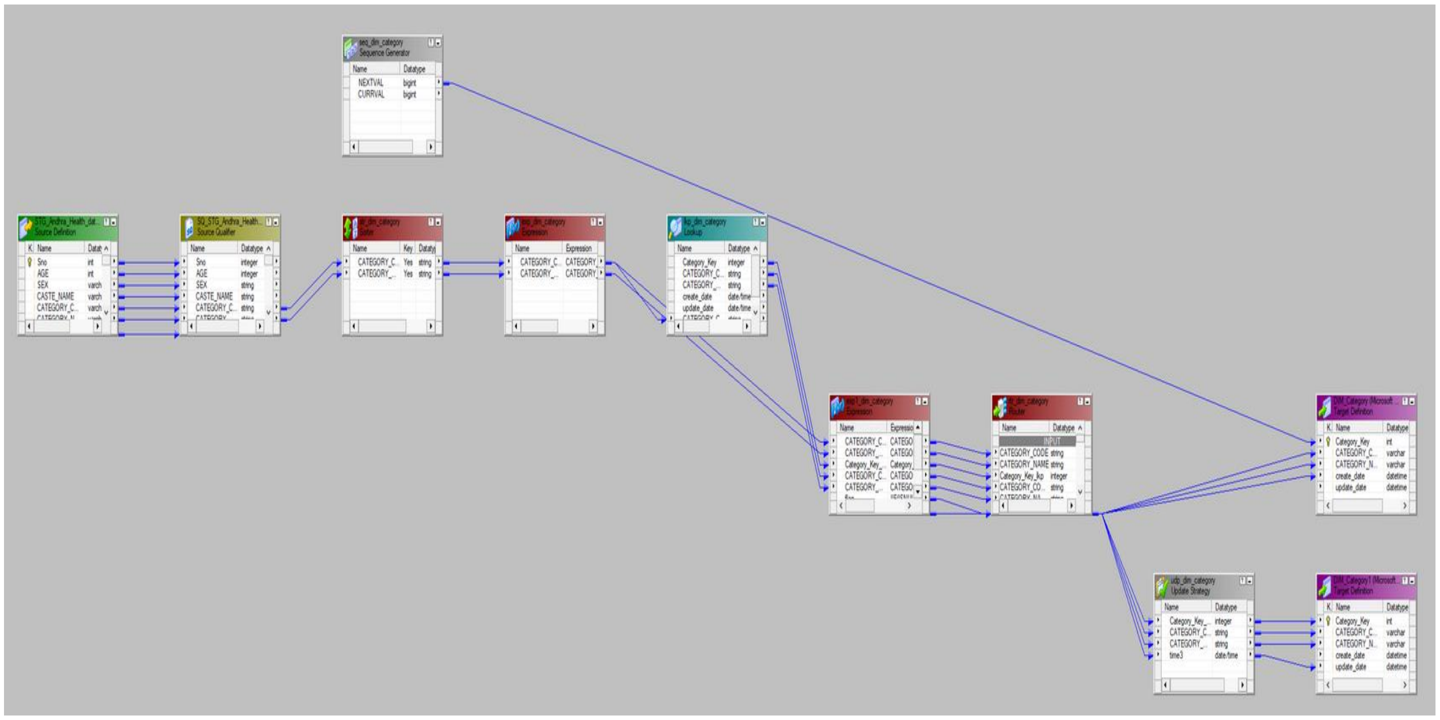


Fig 7 (DIM_Category)

.

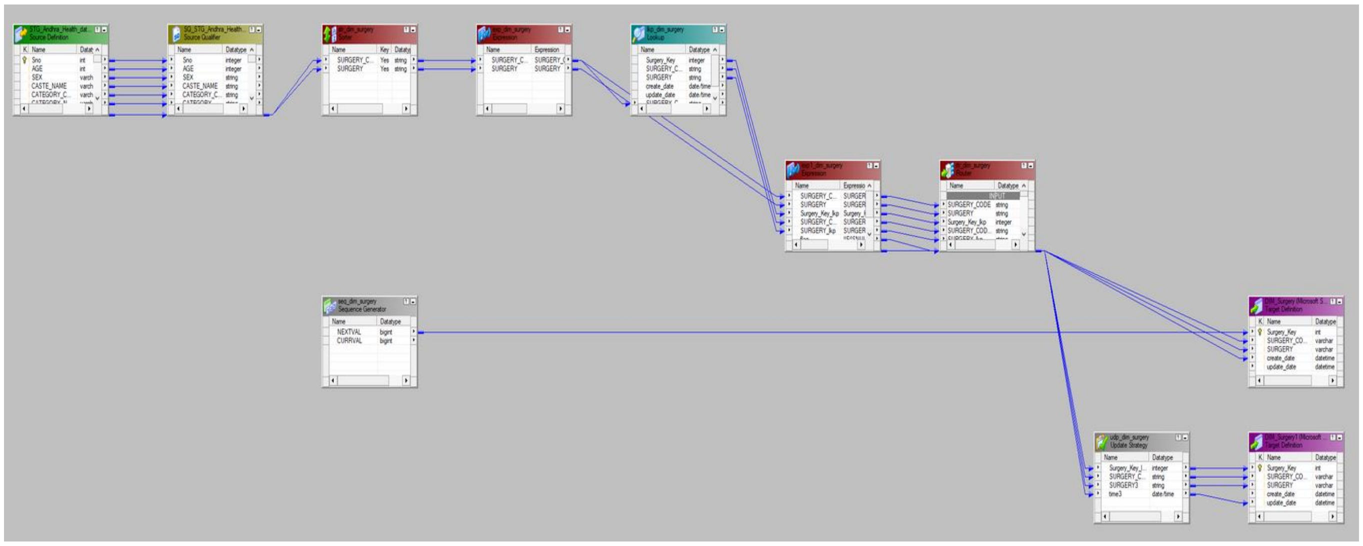


Fig 8 (DIM_SURGERY)

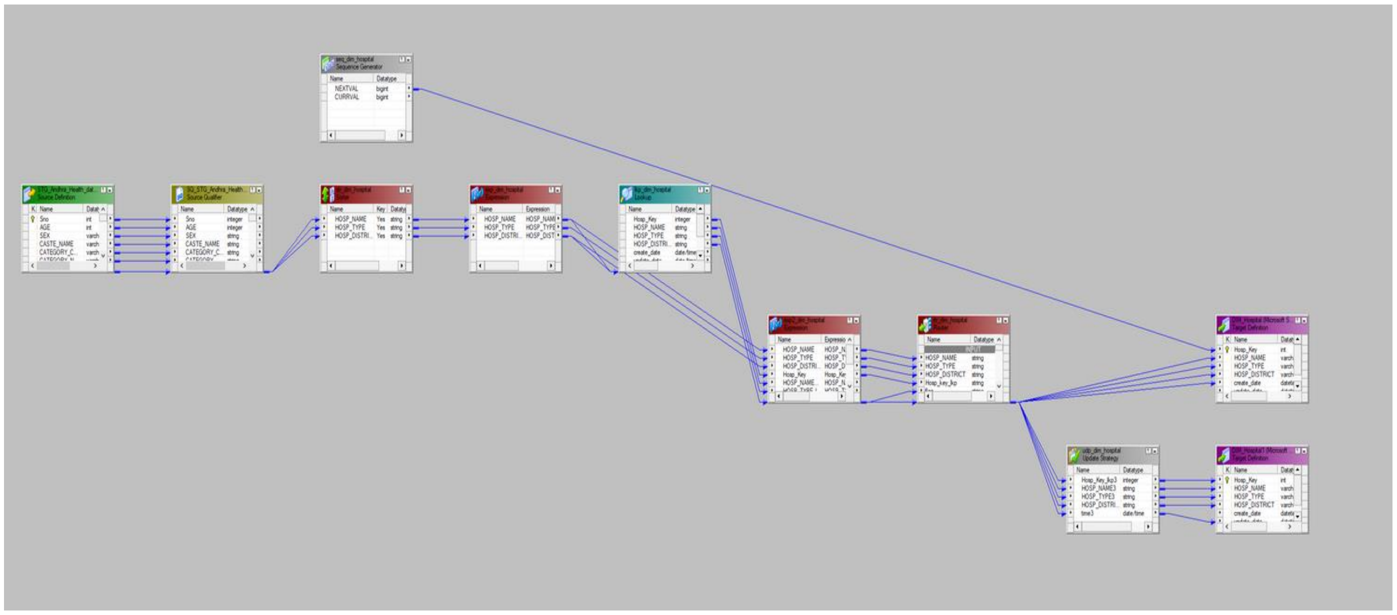


Fig 9(DIM_HOSPITAL)

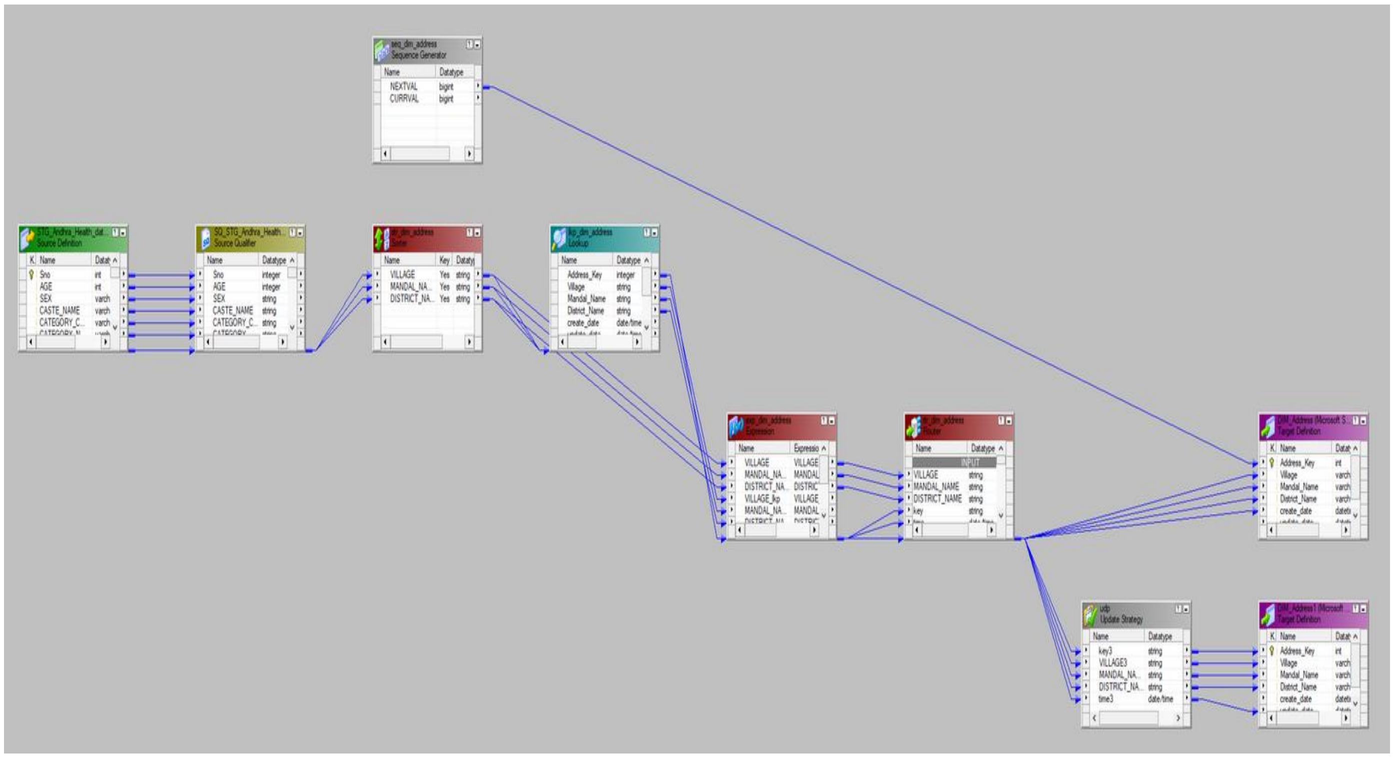


Fig 10 (DIM_ADDRESS)

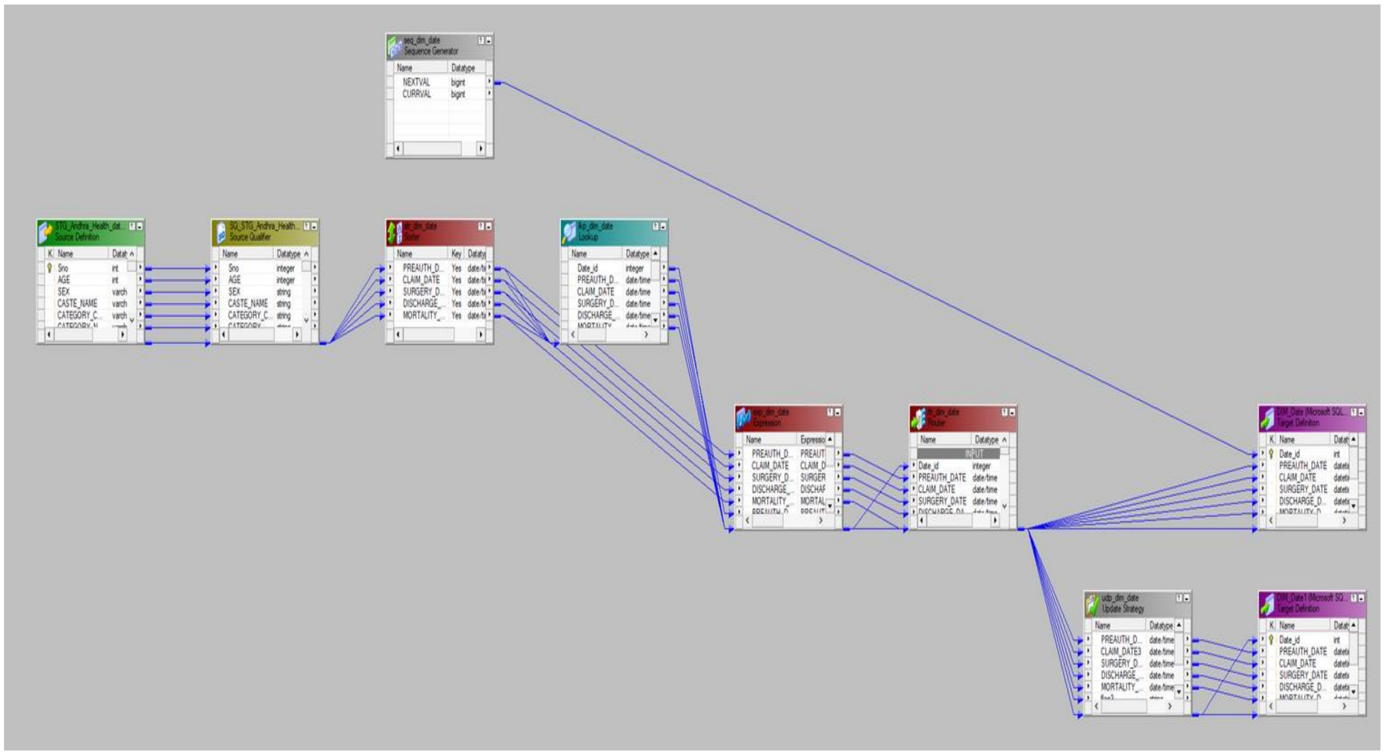


Fig 8 (DIM_DATES)

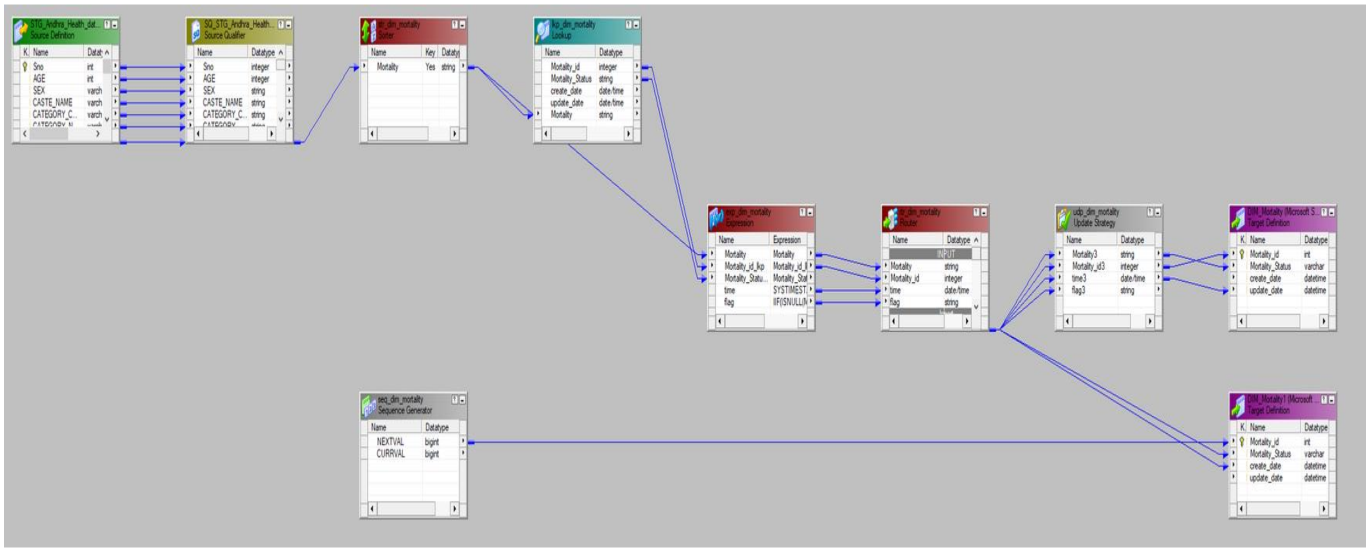


Fig 9 (DIM_MORTALITY)

Informatica provides tools for creating and populating these dimension tables by extracting relevant data from the staging tables. This separation between facts (quantitative measures) and dimensions (descriptive attributes) enhances the flexibility of the data warehouse, allowing users to easily "slice and dice" the data based on specific dimensions during analysis.

Connected and Unconnected Lookups are two types of Lookup transformations in Informatica that serve different purposes.

Connected Lookup:

- Has input and output ports that you connect to other transformations in a mapping.
- Receives input values directly from the pipeline or from the result of a :LKP expression in another transformation.
- Uses a dynamic or static cache.
- Cache includes the lookup source columns in the lookup condition and the lookup source columns that are output ports.
- Returns multiple columns from the same row or inserts into the dynamic lookup cache.
- Returns one column from each row to a return port.
- If there is no match for the lookup condition, the Integration Service returns the default value for all output ports.
- If you configure dynamic cacing, the Integration Service inserts rows into the cache or leaves it unchanged.
- If there is a match for the lookup condition, the Integration Service returns the result of the lookup condition for all lookup/output ports.

- If you configure dynamic caching, the Integration Service either updates the row the in the cache or leaves the row unchanged.
- Passes multiple output values to another transformation.
- Link lookup/output ports to another transformation.

Unconnected Lookup:

- Is a Lookup transformation that is not connected to other transformations in a mapping.
- Receives input values directly from the mapping pipeline or from the result of a :LKP expression in another transformation.
- Cache includes all lookup columns used in the mapping, including columns in the lookup condition and columns linked as output fields to other transformations.
- Can use static or dynamic cache.
- Cache includes all lookup/output fields in the lookup condition and the lookup/return field.
- Cannot use dynamic cache.
- Returns multiple values from the same row.
- Returns the specified field for each row.
- If there is no match for a lookup condition, Data Integration returns the default value for all output fields.
- If there is a match, Data Integration returns the results of the lookup condition for all lookup/output fields.
- Passes multiple output values to another transformation.
- Links lookup/output fields to another transformation.

The purpose of Connected Lookup is to perform a lookup operation on a specific input value and return the corresponding output value to be used in other transformations in the mapping. It is used when you need to retrieve data from a reference table or a lookup table based on the input value and pass it to another transformation for further processing. The purpose of Unconnected Lookup is to perform a lookup operation on a specific input value and return the corresponding output value to be used in the same transformation. It is used when you need to retrieve data from a reference table or a lookup table based on the input value and use it within the same transformation for further processing.

Step 3: Attaching Data to the Final Fact Table

- Source Qualifier: Use the Source Qualifier transformation to define the CSV file as your source.
- Expression Transformation: Use the Expression transformation to clean or transform the data as needed.
- Filter Transformation: Use the Filter transformation to exclude records that do not meet certain criteria.
- Sorter Transformation: Use the Sorter transformation to sort the data if necessary.
- Target Qualifier: Use the Target Qualifier transformation to define the fact table as your target. Specify the table name, connection details, and any other necessary settings.
- Mapping: Create a mapping that connects the Source Qualifier to the Target Qualifier. This mapping will load data into the fact table.

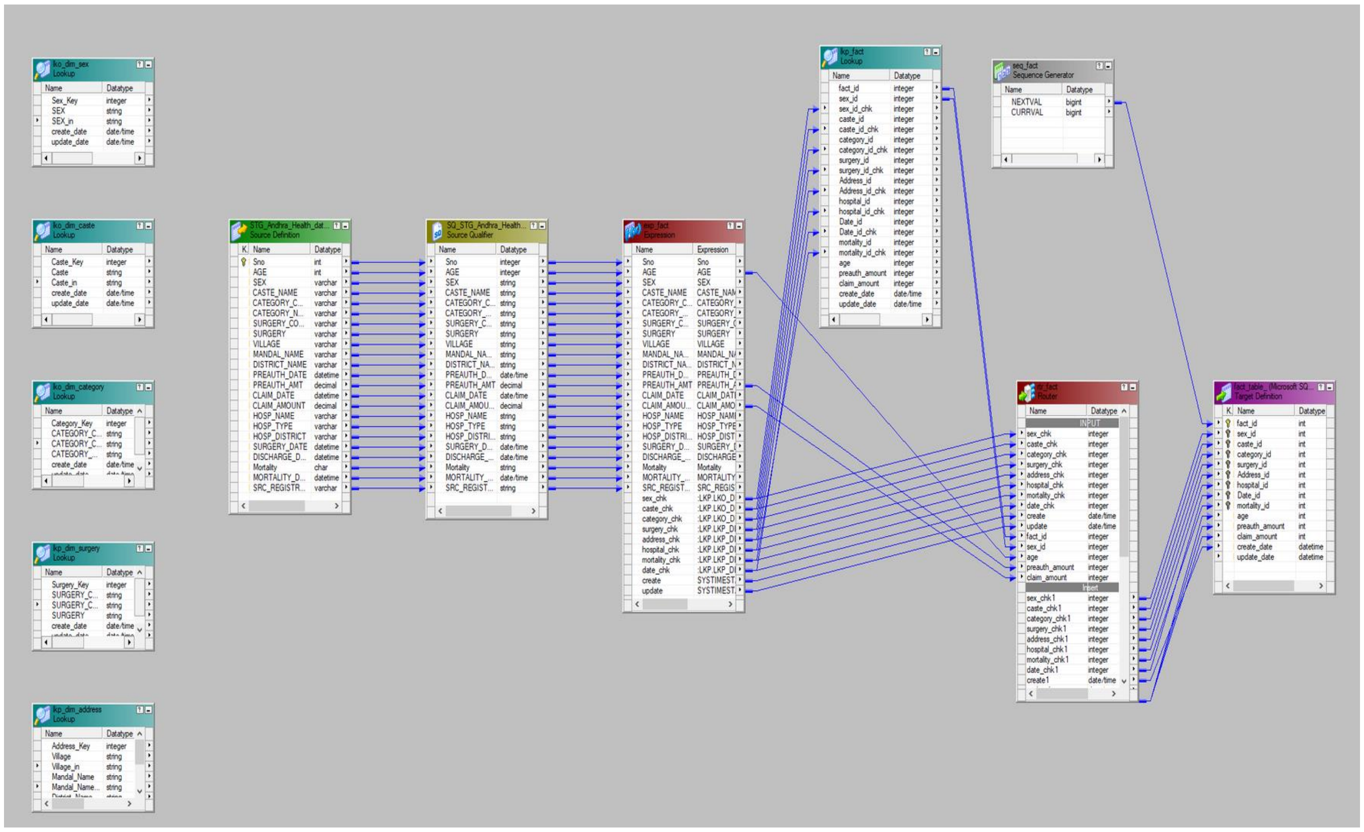
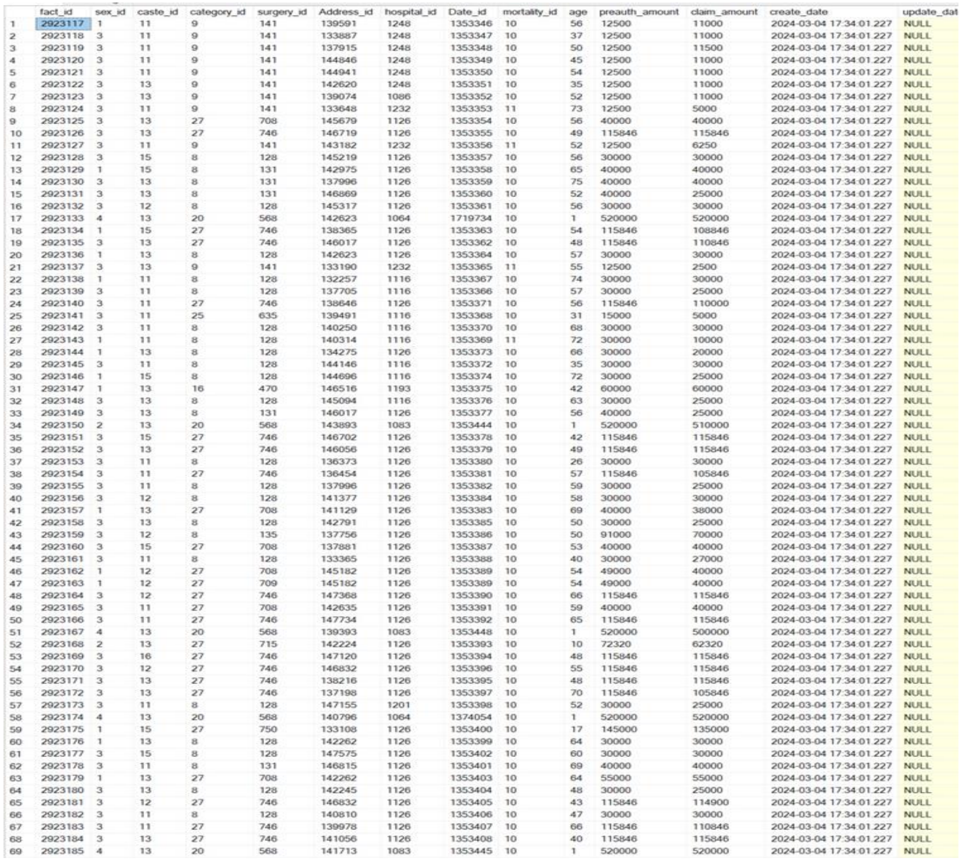

lko_dim_sex Lookup
lko_dim_caste Lookup
lko_dim_category Lookup
lkp_dim_surgery Lookup
lkp_dim_address Lookup
STG_Andhra_Health_dat... Source Definition
SQ_STG_Andhra_Health... Source Qualifier
exp_fact Expression
lkp_fact Lookup
seq_fact Sequence Generator
rtr_fact Router
fact_table_(Microsoft SQ... Target Definition

Fig 10(Fact_table)

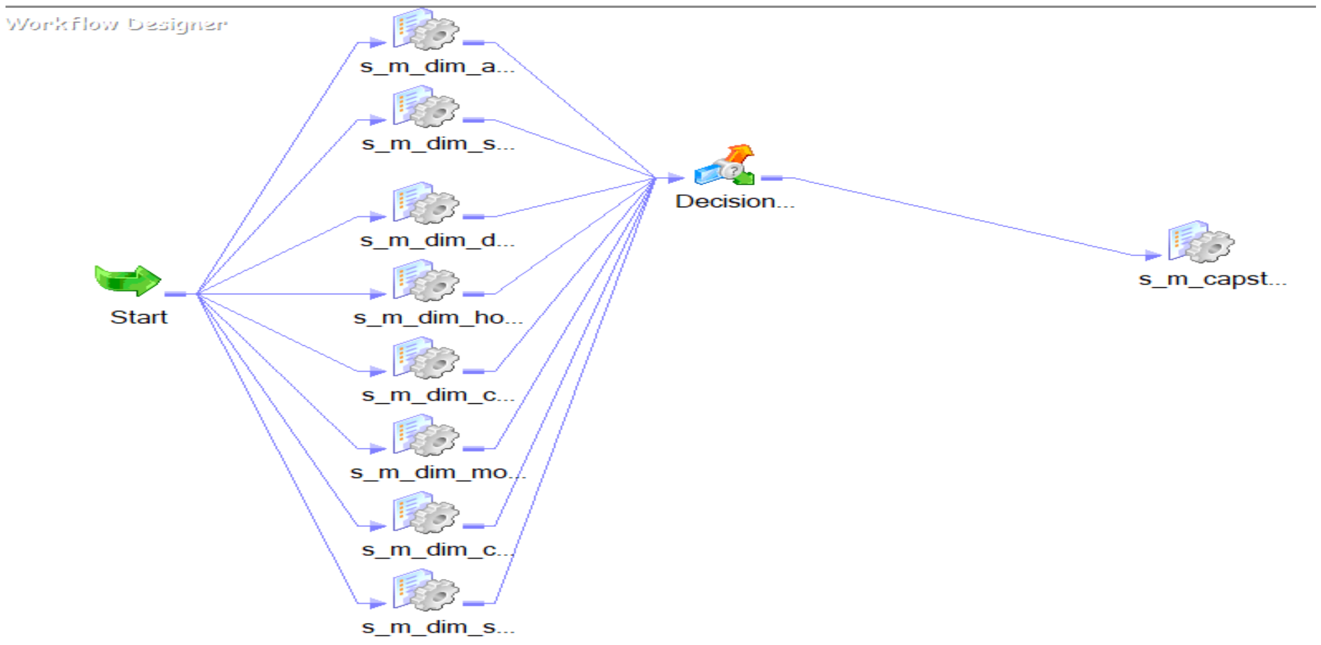


	fact_id	sex_id	caste_id	category_id	surgery_id	Address_id	hospital_id	Date_id	mortality_id	age	preauth_amount	claim_amount	create_date	update_dat
1	2923117	1	11	9	141	139591	1248	1353346	10	56	12500	11000	2024-03-04 17:34:01.227	NULL
2	2923118	3	11	9	141	133887	1248	1353347	10	37	12500	11000	2024-03-04 17:34:01.227	NULL
3	2923119	3	11	9	141	137915	1248	1353348	10	50	12500	11500	2024-03-04 17:34:01.227	NULL
4	2923120	3	11	9	141	144846	1248	1353349	10	45	12500	11000	2024-03-04 17:34:01.227	NULL
5	2923121	3	11	9	141	144941	1248	1353350	10	54	12500	11000	2024-03-04 17:34:01.227	NULL
6	2923122	3	13	9	141	142620	1248	1353351	10	35	12500	11000	2024-03-04 17:34:01.227	NULL
7	2923123	3	13	9	141	139074	1086	1353352	10	52	12500	11000	2024-03-04 17:34:01.227	NULL
8	2923124	3	11	9	141	133648	1232	1353353	11	73	12500	5000	2024-03-04 17:34:01.227	NULL
9	2923125	3	13	27	708	145679	1126	1353354	10	56	40000	40000	2024-03-04 17:34:01.227	NULL
10	2923126	3	13	27	746	146719	1126	1353355	10	49	115846	115846	2024-03-04 17:34:01.227	NULL
11	2923127	3	11	9	141	143182	1232	1353356	11	52	12500	6250	2024-03-04 17:34:01.227	NULL
12	2923128	3	15	8	128	145219	1126	1353357	10	56	30000	30000	2024-03-04 17:34:01.227	NULL
13	2923129	1	15	8	131	142975	1126	1353358	10	65	40000	40000	2024-03-04 17:34:01.227	NULL
14	2923130	3	13	8	131	137996	1126	1353359	10	75	40000	40000	2024-03-04 17:34:01.227	NULL
15	2923131	3	13	8	131	146869	1126	1353360	10	52	40000	25000	2024-03-04 17:34:01.227	NULL
16	2923132	3	12	8	128	145317	1126	1353361	10	56	30000	30000	2024-03-04 17:34:01.227	NULL
17	2923133	4	13	20	568	142623	1064	1719734	10	1	520000	520000	2024-03-04 17:34:01.227	NULL
18	2923134	1	15	27	746	138365	1126	1353363	10	54	115846	108846	2024-03-04 17:34:01.227	NULL
19	2923135	3	13	27	746	146017	1126	1353362	10	48	115846	110846	2024-03-04 17:34:01.227	NULL
20	2923136	1	13	8	128	142623	1126	1353364	10	57	30000	30000	2024-03-04 17:34:01.227	NULL
21	2923137	3	13	9	141	133190	1232	1353365	11	55	12500	2500	2024-03-04 17:34:01.227	NULL
22	2923138	1	11	8	128	132257	1116	1353367	10	74	30000	30000	2024-03-04 17:34:01.227	NULL
23	2923139	3	11	8	128	137705	1116	1353366	10	57	30000	25000	2024-03-04 17:34:01.227	NULL
24	2923140	3	11	27	746	138646	1126	1353371	10	56	115846	110000	2024-03-04 17:34:01.227	NULL
25	2923141	3	11	25	635	139491	1116	1353368	10	31	15000	5000	2024-03-04 17:34:01.227	NULL
26	2923142	3	11	8	128	140250	1116	1353370	10	68	30000	30000	2024-03-04 17:34:01.227	NULL
27	2923143	1	11	8	128	140314	1116	1353369	11	72	30000	10000	2024-03-04 17:34:01.227	NULL
28	2923144	1	13	8	128	134275	1126	1353373	10	66	30000	20000	2024-03-04 17:34:01.227	NULL
29	2923145	3	11	8	128	144146	1116	1353372	10	35	30000	30000	2024-03-04 17:34:01.227	NULL
30	2923146	1	15	8	128	144696	1116	1353374	10	72	30000	25000	2024-03-04 17:34:01.227	NULL
31	2923147	1	13	16	470	146516	1193	1353375	10	42	60000	60000	2024-03-04 17:34:01.227	NULL
32	2923148	3	13	8	128	145094	1116	1353376	10	63	30000	25000	2024-03-04 17:34:01.227	NULL
33	2923149	3	13	8	131	146017	1126	1353377	10	56	40000	25000	2024-03-04 17:34:01.227	NULL
34	2923150	2	13	20	568	143893	1083	1353444	10	1	520000	510000	2024-03-04 17:34:01.227	NULL
35	2923151	3	15	27	746	146702	1126	1353378	10	42	115846	115846	2024-03-04 17:34:01.227	NULL
36	2923152	3	13	27	746	146056	1126	1353379	10	49	115846	115846	2024-03-04 17:34:01.227	NULL
37	2923153	3	11	8	128	136373	1126	1353380	10	26	30000	30000	2024-03-04 17:34:01.227	NULL
38	2923154	3	11	27	746	136454	1126	1353381	10	57	115846	105846	2024-03-04 17:34:01.227	NULL
39	2923155	3	11	8	128	137996	1126	1353382	10	59	30000	25000	2024-03-04 17:34:01.227	NULL
40	2923156	3	12	8	128	141377	1126	1353384	10	58	30000	30000	2024-03-04 17:34:01.227	NULL
41	2923157	1	13	27	708	141129	1126	1353383	10	69	40000	38000	2024-03-04 17:34:01.227	NULL
42	2923158	3	13	8	128	142791	1126	1353385	10	50	30000	25000	2024-03-04 17:34:01.227	NULL
43	2923159	3	12	8	135	137756	1126	1353386	10	50	91000	70000	2024-03-04 17:34:01.227	NULL
44	2923160	3	15	27	708	137881	1126	1353387	10	53	40000	40000	2024-03-04 17:34:01.227	NULL
45	2923161	3	11	8	128	133365	1126	1353388	10	40	30000	27000	2024-03-04 17:34:01.227	NULL
46	2923162	1	12	27	708	145182	1126	1353389	10	54	49000	40000	2024-03-04 17:34:01.227	NULL
47	2923163	1	12	27	709	145182	1126	1353389	10	54	49000	40000	2024-03-04 17:34:01.227	NULL
48	2923164	3	12	27	746	147368	1126	1353390	10	66	115846	115846	2024-03-04 17:34:01.227	NULL
49	2923165	3	11	27	708	142635	1126	1353391	10	59	40000	40000	2024-03-04 17:34:01.227	NULL
50	2923166	3	11	27	746	147734	1126	1353392	10	65	115846	115846	2024-03-04 17:34:01.227	NULL
51	2923167	4	13	20	568	139393	1083	1353448	10	1	520000	500000	2024-03-04 17:34:01.227	NULL
52	2923168	2	13	27	715	142224	1126	1353393	10	10	72320	62320	2024-03-04 17:34:01.227	NULL
53	2923169	3	16	27	746	147120	1126	1353394	10	48	115846	115846	2024-03-04 17:34:01.227	NULL
54	2923170	3	12	27	746	146832	1126	1353396	10	55	115846	115846	2024-03-04 17:34:01.227	NULL
55	2923171	3	13	27	746	138216	1126	1353395	10	48	115846	115846	2024-03-04 17:34:01.227	NULL
56	2923172	3	13	27	746	137198	1126	1353397	10	70	115846	105846	2024-03-04 17:34:01.227	NULL
57	2923173	3	11	8	128	147155	1201	1353398	10	52	30000	25000	2024-03-04 17:34:01.227	NULL
58	2923174	4	13	20	568	140796	1064	1374054	10	1	520000	520000	2024-03-04 17:34:01.227	NULL
59	2923175	1	15	27	750	133108	1126	1353400	10	17	145000	135000	2024-03-04 17:34:01.227	NULL
60	2923176	1	13	8	128	142262	1126	1353399	10	64	30000	30000	2024-03-04 17:34:01.227	NULL
61	2923177	3	15	8	128	147575	1126	1353402	10	60	30000	30000	2024-03-04 17:34:01.227	NULL
62	2923178	3	11	8	131	146815	1126	1353401	10	69	40000	40000	2024-03-04 17:34:01.227	NULL
63	2923179	1	13	27	708	142262	1126	1353403	10	64	55000	55000	2024-03-04 17:34:01.227	NULL
64	2923180	3	13	8	128	142245	1126	1353404	10	48	30000	25000	2024-03-04 17:34:01.227	NULL
65	2923181	3	12	27	746	146832	1126	1353405	10	43	115846	114900	2024-03-04 17:34:01.227	NULL
66	2923182	3	11	8	128	140810	1126	1353406	10	47	30000	30000	2024-03-04 17:34:01.227	NULL
67	2923183	3	11	27	746	139978	1126	1353407	10	66	115846	110846	2024-03-04 17:34:01.227	NULL
68	2923184	3	13	27	746	141056	1126	1353408	10	40	115846	115846	2024-03-04 17:34:01.227	NULL
69	2923185	4	13	20	568	141713	1083	1353445	10	1	520000	520000	2024-03-04 17:34:01.227	NULL

Fig 11(Fact_Data)

Following the data warehousing stage, Python is employed as a versatile tool for extracting knowledge from the Andhra Health dataset. Python scripts are utilized to perform exploratory data analysis, which involves cleaning and transforming the data, identifying trends and patterns, and uncovering hidden correlations. Through data visualization, the project highlights growth rates, correlations between GDP and other economic indicators, and segmented data based on geographical regions or timeframes. By leveraging Python's analytical capabilities, the project uncovers valuable economic insights that can inform decision-making processes across various sectors.
The combination of Informatica ETL and Python analysis in this project offers a comprehensive approach to data analysis, enabling the extraction of valuable economic insights from the Andhra Health dataset. By consolidating data from various sources into a centralized data warehouse, the project ensures data integrity and facilitates efficient data retrieval. Through Python's analytical capabilities, the project uncovers hidden patterns, trends, and correlations within the Andhra Health data, providing actionable insights that can inform decision-making processes. This project demonstrates the potential of data warehousing and Python analysis in providing valuable economic intelligence, highlighting the importance of data-driven decision-making in today's economy.

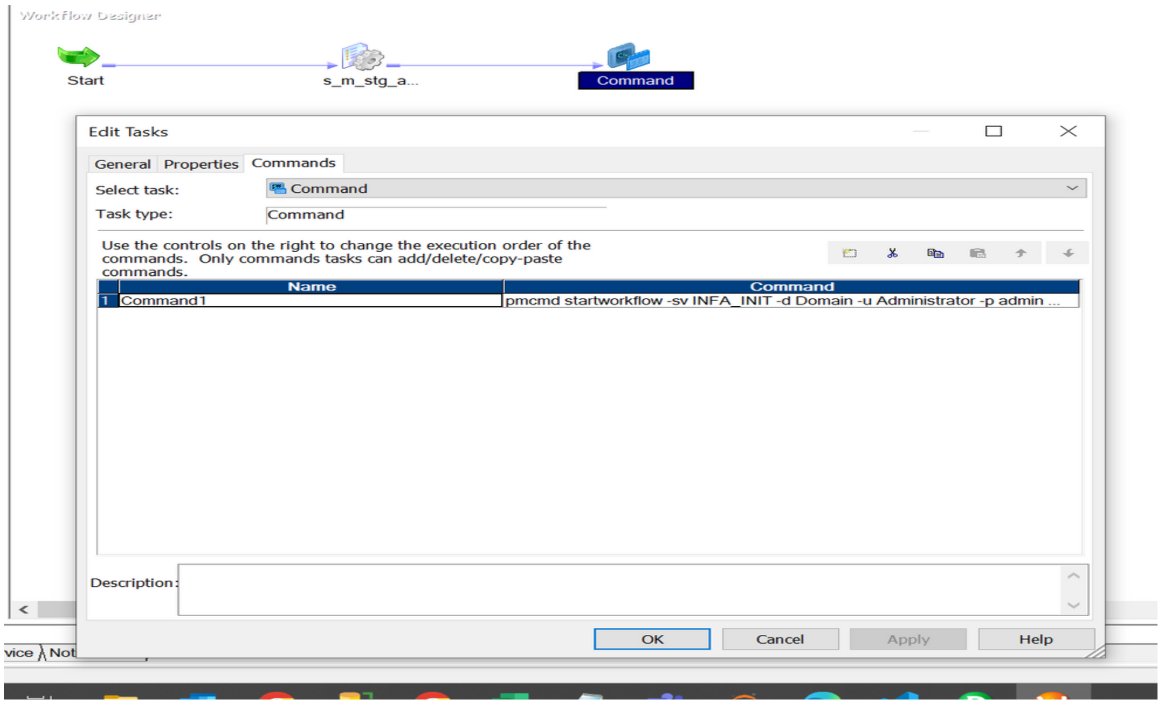
Wolkflow Diagram



Choice, Command, and Occurrence An essential part of Informatica PowerCenter workflows for controlling data processing flow are wait jobs.
• Decision tasks give processes conditional logic by enabling users to specify conditions depending on the

state of earlier tasks or variables.
• This makes it possible for the workflow to diverge and carry out various activities in response to the condition's result.
• Control tasks offer an approach to oversee the workflow's flow, incorporating the capacity to perform activities in parallel, loop over tasks, and regulate the sequence in which they are completed. Among other tasks, the Link, Sequence, and Parallel tasks are examples of control tasks.

PMCMD



Select task: Command

Task type: Command

Use the controls on the right to change the execution order of the commands. Only commands tasks can add/delete/copy-paste commands.

	Name	Command
1	Command1	pmcmd startworkflow -sv INFA_INIT -d Domain -u Administrator -p admin ...

Description:

The pmcmd command in Informatica is a powerful tool used for managing and executing tasks within the Informatica PowerCenter environment. It can be used in two primary modes: command line mode and

Command Line Mode: In this mode, you invoke and exit pmcmd each time you issue a command. This mode is particularly useful for writing scripts to schedule workflows with the command line syntax. Each command you write in command line mode must include connection information to the Integration Service. This mode is suitable for automating tasks and executing commands without the need for an active connection to the Integration Service 2.

Interactive Mode: In interactive mode, you establish and maintain an active connection to the Integration Service. This allows you to issue a series of commands without needing to re-establish the connection for each command. This mode is beneficial for interactive sessions where you want to execute multiple commands in sequence without the overhead of establishing a connection for each command 2.

Running Commands: When running commands in command line mode, you can specify various options and parameters to control the execution of your tasks. This includes specifying the workflow to run, the target environment, and other parameters relevant to your task 2.

Scripting pmcmd Commands: You can script pmcmd commands to automate the execution of tasks. This is particularly useful for scheduling workflows and managing tasks in a consistent and repeatable manner. Scripting pmcmd commands allows you to leverage the power of automation to manage your Informatica PowerCenter environment more efficiently 2.

Entering Command Options: When using pmcmd, you can enter various command options to customize the behavior of your commands. These options can be used to specify the target environment, the workflow to run, and other parameters relevant to your task. Understanding and utilizing these options can help you achieve more precise control over your tasks 2.

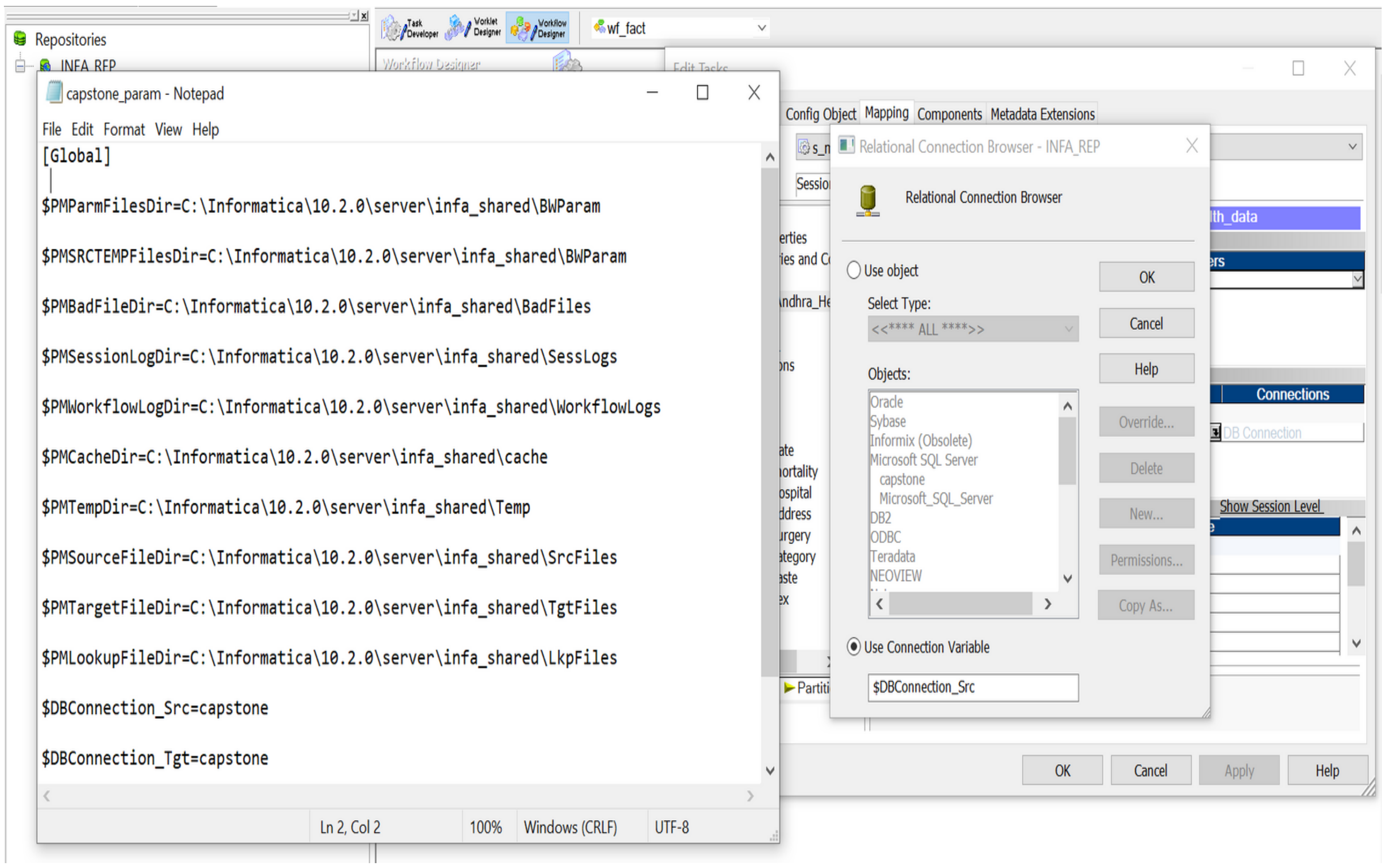
Troubleshooting: If you encounter issues when using pmcmd, such as the command not being found, ensure that the Informatica PowerCenter installation path is correctly set in your system's $PATH variable. This is a common issue when trying to execute pmcmd commands in a UNIX environment, and it can be resolved by updating the $PATH variable to include the directory where Informatica PowerCenter is installed 4.

In summary, pmcmd is a versatile command-line tool in Informatica PowerCenter that allows for the management and execution of tasks in both command line and interactive modes. It supports scripting for automation and offers various options for customizing task execution. Understanding how to use pmcmd effectively can significantly enhance your ability to manage and automate tasks within the Informatica PowerCenter environment.

Parameterization

Overriding Parameters

1. Create a Taskflow: Start by creating a taskflow in Informatica Cloud.
2. Add a Data Task Step: Include a Data Task step in your taskflow canvas.
3. Add the Mapping Task with Parameters: Within the Data Task step, add the mapping task that contains the parameters you wish to override.
4. Override Connection and Data Object Parameters:
    - For connection parameters, select the connection parameter from the list, click Edit, choose Content as the Field type, and select the desired connection from the list of available connections.
    - For data object parameters, select the data object parameter, click Edit, choose Content as the Field type, and enter the new value for the data object.
5. Repeat for All Parameters: Continue this process for all connection and data object parameters you wish to override from the taskflow.


```
[Global]

$PMParmFilesDir=C:\Informatica\10.2.0\server\infa_shared\BWParam

$PMSRCTEMPFilesDir=C:\Informatica\10.2.0\server\infa_shared\BWParam

$PMBadFileDir=C:\Informatica\10.2.0\server\infa_shared\BadFiles

$PMSessionLogDir=C:\Informatica\10.2.0\server\infa_shared\SessLogs

$PMWorkflowLogDir=C:\Informatica\10.2.0\server\infa_shared\WorkflowLogs

$PMCacheDir=C:\Informatica\10.2.0\server\infa_shared\cache

$PMTempDir=C:\Informatica\10.2.0\server\infa_shared\Temp

$PMSourceFileDir=C:\Informatica\10.2.0\server\infa_shared\SrcFiles

$PMTargetFileDir=C:\Informatica\10.2.0\server\infa_shared\TgtFiles

$PMLookupFileDir=C:\Informatica\10.2.0\server\infa_shared\LkpFiles

$DBConnection_Src=capstone

$DBConnection_Tgt=capstone
```

Relational Connection Browser - INFA_REP

Use object [✓ Use Connection Variable]

Select Type: <<**** ALL ****>>

Objects: Oracle, Sybase, Informix (Obsolete), Microsoft SQL Server, capstone, Microsoft_SQL_Server, DB2, ODBC, Teradata, NEOVIEW

Use Connection Variable: $DBConnection_Src

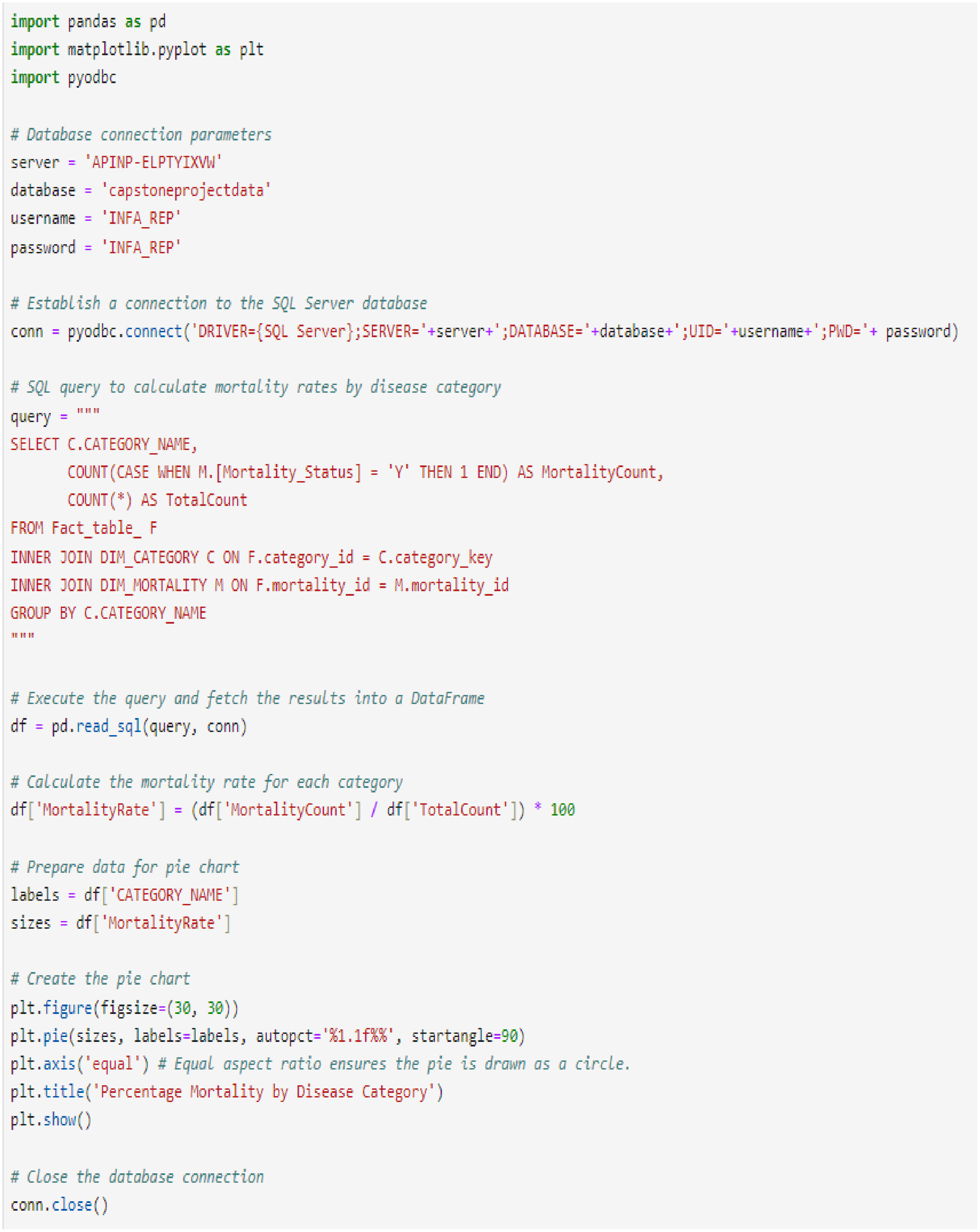
Data Analysis with SQL

Following the ETL stages, this project delves into data analysis using SQL queries. The fact table, constructed in Stage 4, serves as the central repository for quantitative data. SQL, a structured query language, allows you to extract valuable insights from this data. Here's a breakdown of the analyses you've outlined:

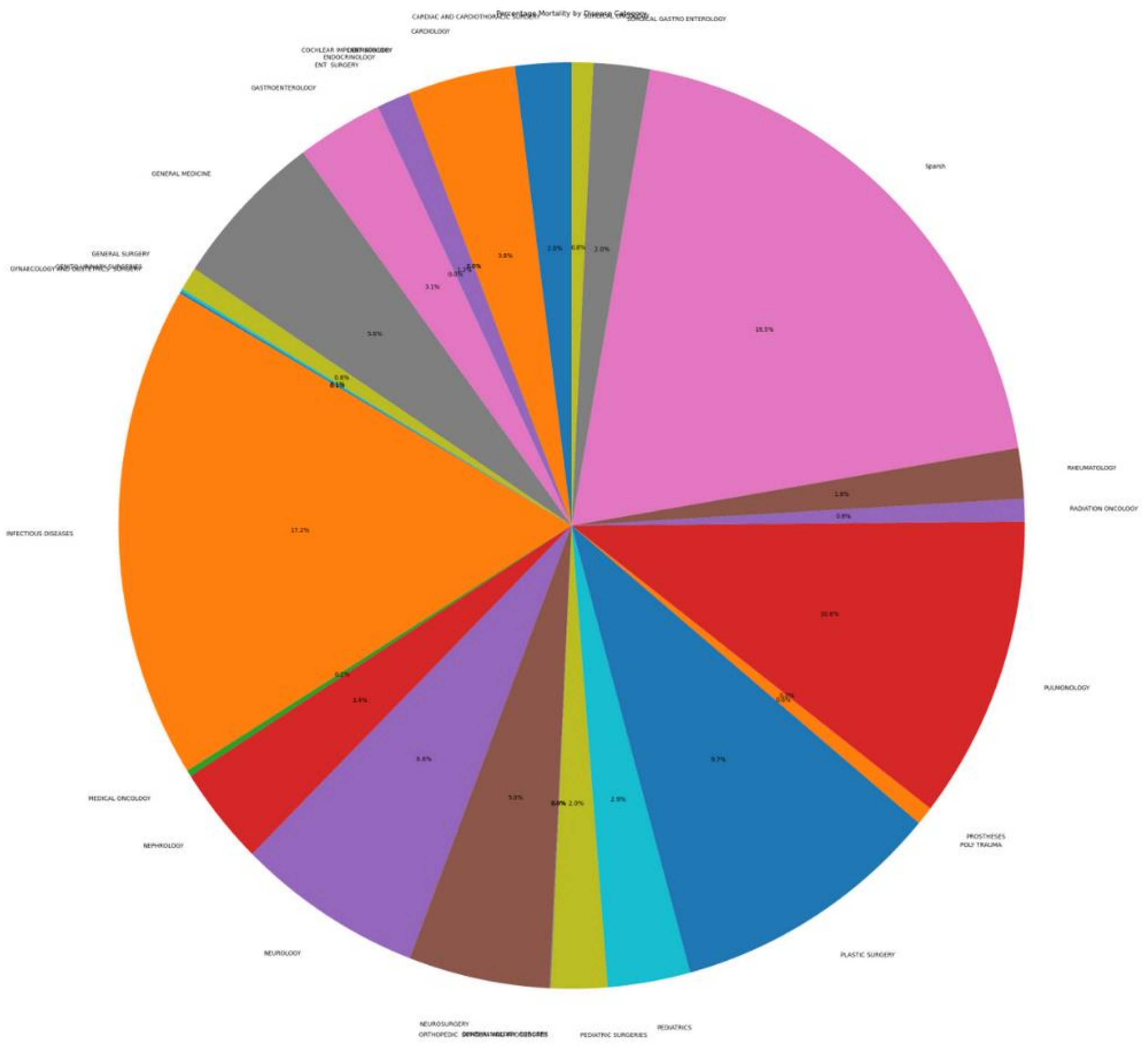
1. Mortality Percentage For different category of Diseases

```
SELECT C.CATEGORY_NAME,
    COUNT(CASE WHEN M.[Mortality_Status] = 'Y' THEN 1 END) AS MortalityCount,
    COUNT(*) AS TotalCount
FROM Fact_table_ F
INNER JOIN DIM_CATEGORY C ON F.category_id = C.category_key
INNER JOIN DIM_MORTALITY M ON F.mortality_id = M.mortality_id
GROUP BY C.CATEGORY_NAME
```

Python code



```python
import pandas as pd
import matplotlib.pyplot as plt
import pyodbc

# Database connection parameters
server = 'APINP-ELPTYIXVW'
database = 'capstoneprojectdata'
username = 'INFA_REP'
password = 'INFA_REP'

# Establish a connection to the SQL Server database
conn = pyodbc.connect('DRIVER={SQL Server};SERVER='+server+';DATABASE='+database+';UID='+username+';PWD='+ password)

# SQL query to calculate mortality rates by disease category
query = """
SELECT C.CATEGORY_NAME,
       COUNT(CASE WHEN M.[Mortality_Status] = 'Y' THEN 1 END) AS MortalityCount,
       COUNT(*) AS TotalCount
FROM Fact_table_ F
INNER JOIN DIM_CATEGORY C ON F.category_id = C.category_key
INNER JOIN DIM_MORTALITY M ON F.mortality_id = M.mortality_id
GROUP BY C.CATEGORY_NAME
"""

# Execute the query and fetch the results into a DataFrame
df = pd.read_sql(query, conn)

# Calculate the mortality rate for each category
df['MortalityRate'] = (df['MortalityCount'] / df['TotalCount']) * 100

# Prepare data for pie chart
labels = df['CATEGORY_NAME']
sizes = df['MortalityRate']

# Create the pie chart
plt.figure(figsize=(30, 30))
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)
plt.axis('equal') # Equal aspect ratio ensures the pie is drawn as a circle.
plt.title('Percentage Mortality by Disease Category')
plt.show()

# Close the database connection
conn.close()
```



Selection of Columns: It starts by selecting the CATEGORY_NAME from the DIM_CATEGORY table. This is the category name that will be used to group the results.

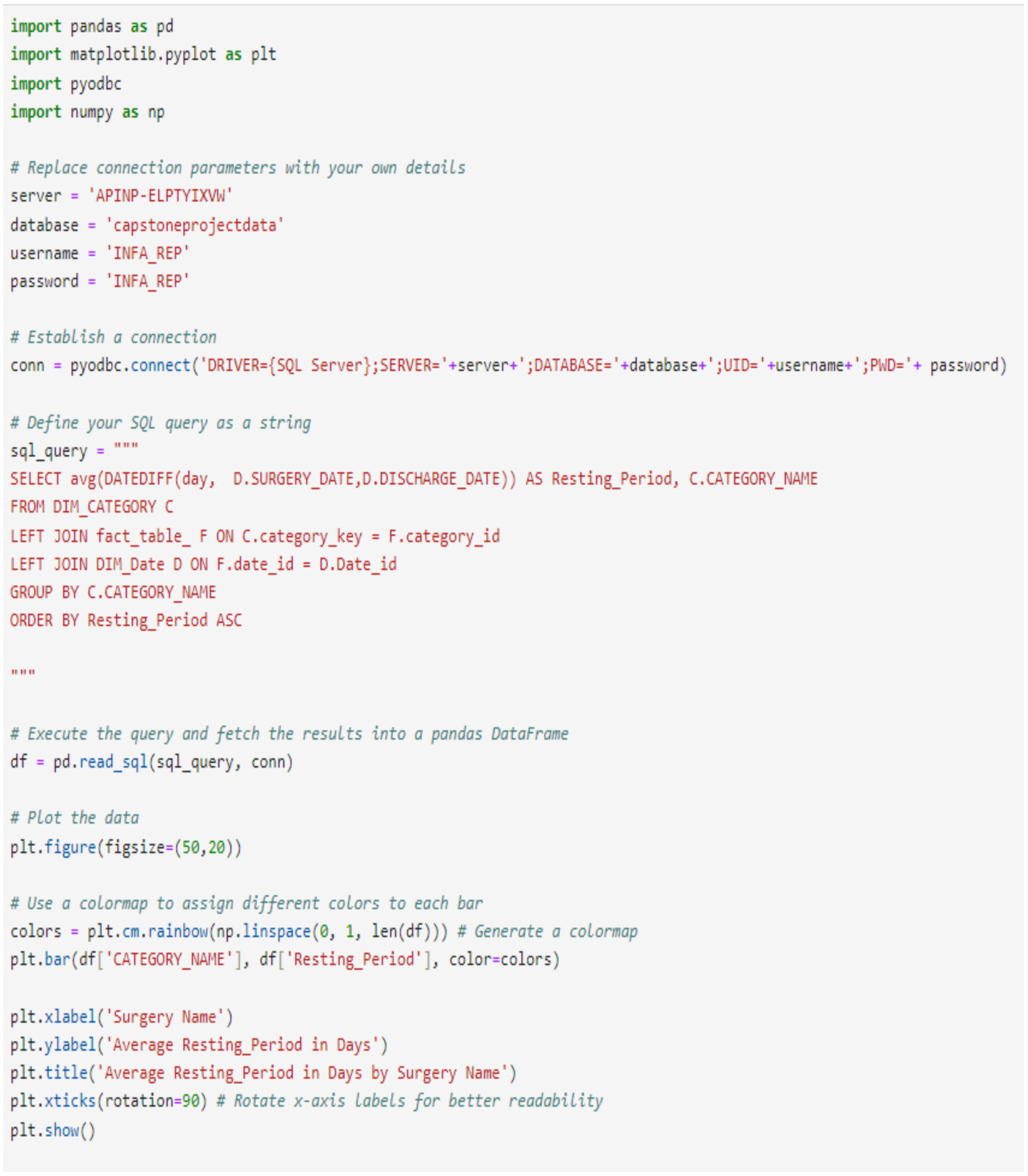
Conditional Counting: It then counts the number of records where the mortality status is 'Y' (indicating a mortality case) for each category. This is achieved using a CASE statement within the COUNT function, which only counts records where the condition is met.

Total Count: Additionally, it counts the total number of records for each category, regardless of mortality status. This is done using a simple COUNT(*) function.

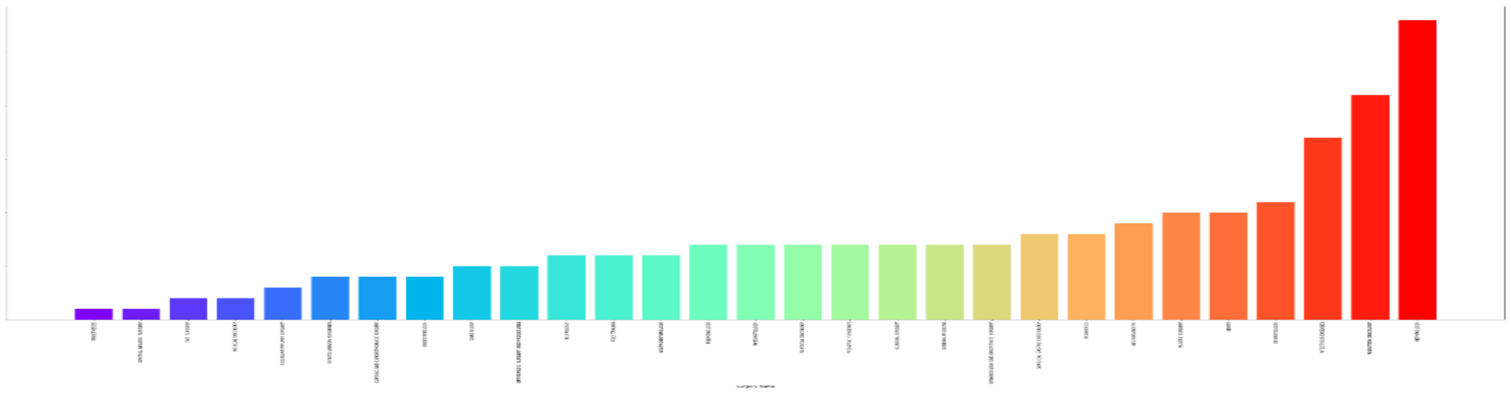
Joining Tables: The query joins the fact table with the dimension tables based on specific keys. It uses an INNER JOIN to connect the fact table with the DIM_CATEGORY table using the category_id from the fact table and the category_key from the dimension table. Similarly, it connects the fact table with the DIM_MORTALITY table using the mortality_id from both tables.

Grouping Results: Finally, it groups the results by the CATEGORY_NAME from the DIM_CATEGORY table. This means that the query will return one row for each category, with columns for the category name, the count of mortality cases, and the total count of records for that category.

2.Average Recovery Rates for different Diseases



```
import pandas as pd
import matplotlib.pyplot as plt
import pyodbc
import numpy as np

# Replace connection parameters with your own details
server = 'APINP-ELPTYIXVW'
database = 'capstoneprojectdata'
username = 'INFA_REP'
password = 'INFA_REP'

# Establish a connection
conn = pyodbc.connect('DRIVER={SQL Server};SERVER='+server+';DATABASE='+database+';UID='+username+';PWD='+ password)

# Define your SQL query as a string
sql_query = """
SELECT avg(DATEDIFF(day,  D.SURGERY_DATE,D.DISCHARGE_DATE)) AS Resting_Period, C.CATEGORY_NAME
FROM DIM_CATEGORY C
LEFT JOIN fact_table_ F ON C.category_key = F.category_id
LEFT JOIN DIM_Date D ON F.date_id = D.Date_id
GROUP BY C.CATEGORY_NAME
ORDER BY Resting_Period ASC

"""

# Execute the query and fetch the results into a pandas DataFrame
df = pd.read_sql(sql_query, conn)

# Plot the data
plt.figure(figsize=(50,20))

# Use a colormap to assign different colors to each bar
colors = plt.cm.rainbow(np.linspace(0, 1, len(df))) # Generate a colormap
plt.bar(df['CATEGORY_NAME'], df['Resting_Period'], color=colors)

plt.xlabel('Surgery Name')
plt.ylabel('Average Resting_Period in Days')
plt.title('Average Resting_Period in Days by Surgery Name')
plt.xticks(rotation=90) # Rotate x-axis labels for better readability
plt.show()
```



```
SELECT avg(DATEDIFF(day,  D.SURGERY_DATE,D.DISCHARGE_DATE)) AS Resting_Period,
C.CATEGORY_NAME
FROM DIM_CATEGORY C
LEFT JOIN fact_table_ F ON C.category_key = F.category_id
LEFT JOIN DIM_Date D ON F.date_id = D.Date_id
GROUP BY C.CATEGORY_NAME
ORDER BY Resting_Period ASC
```

1. **Selection of Columns**: It starts by selecting the Category_name from the DIM_CATEGORY table. This is the category name that will be used to group the results.

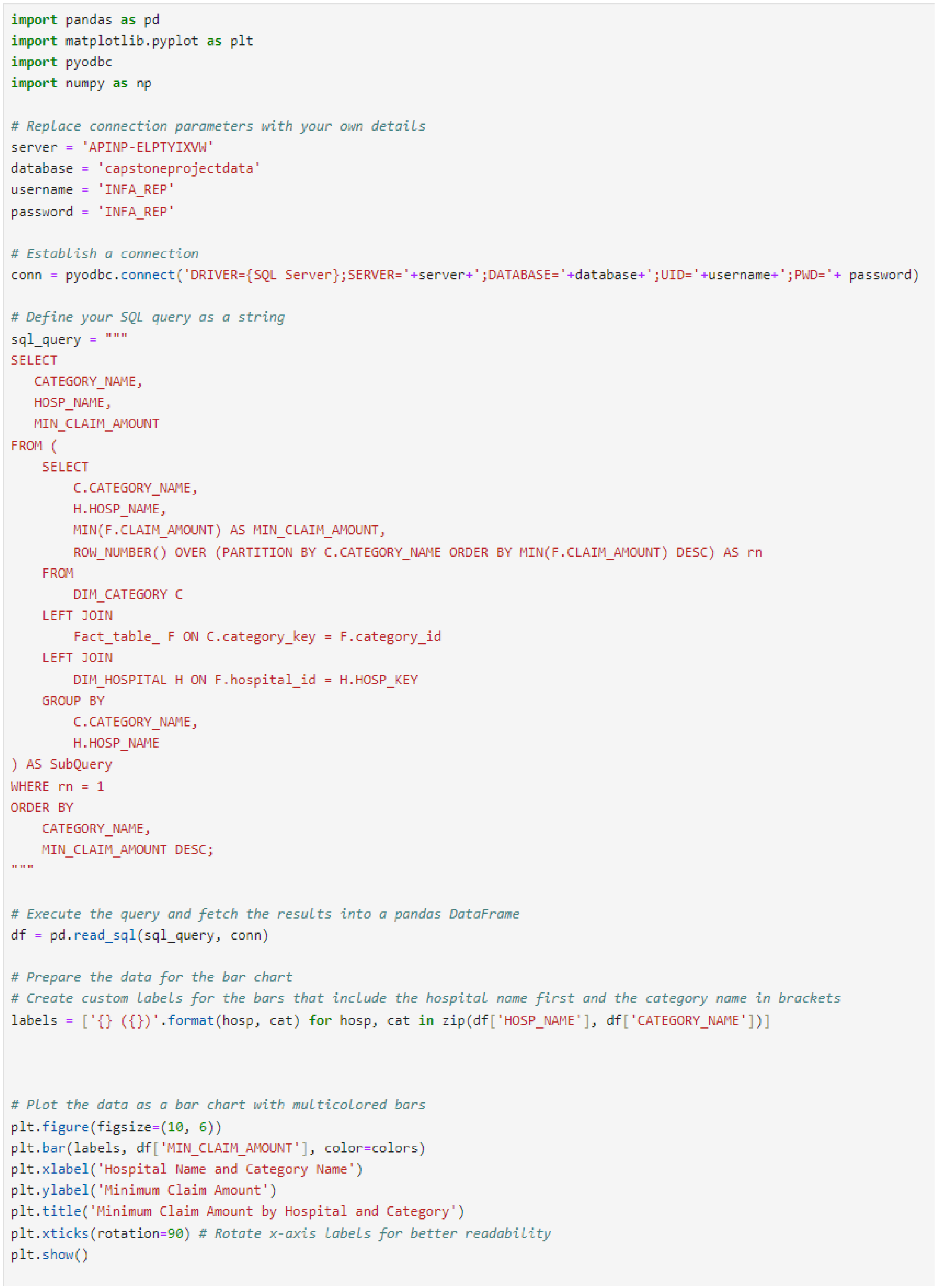
2. **Conditional Counting**: It then counts the number of records where the mortality status is 'Y' (indicating a mortality case) for each category. This is achieved using a CASE statement within the COUNT function, which only counts records where the condition is met.

3. **Total Count**: Additionally, it counts the total number of records for each category, regardless of mortality status. This is done using a simple COUNT(*) function.

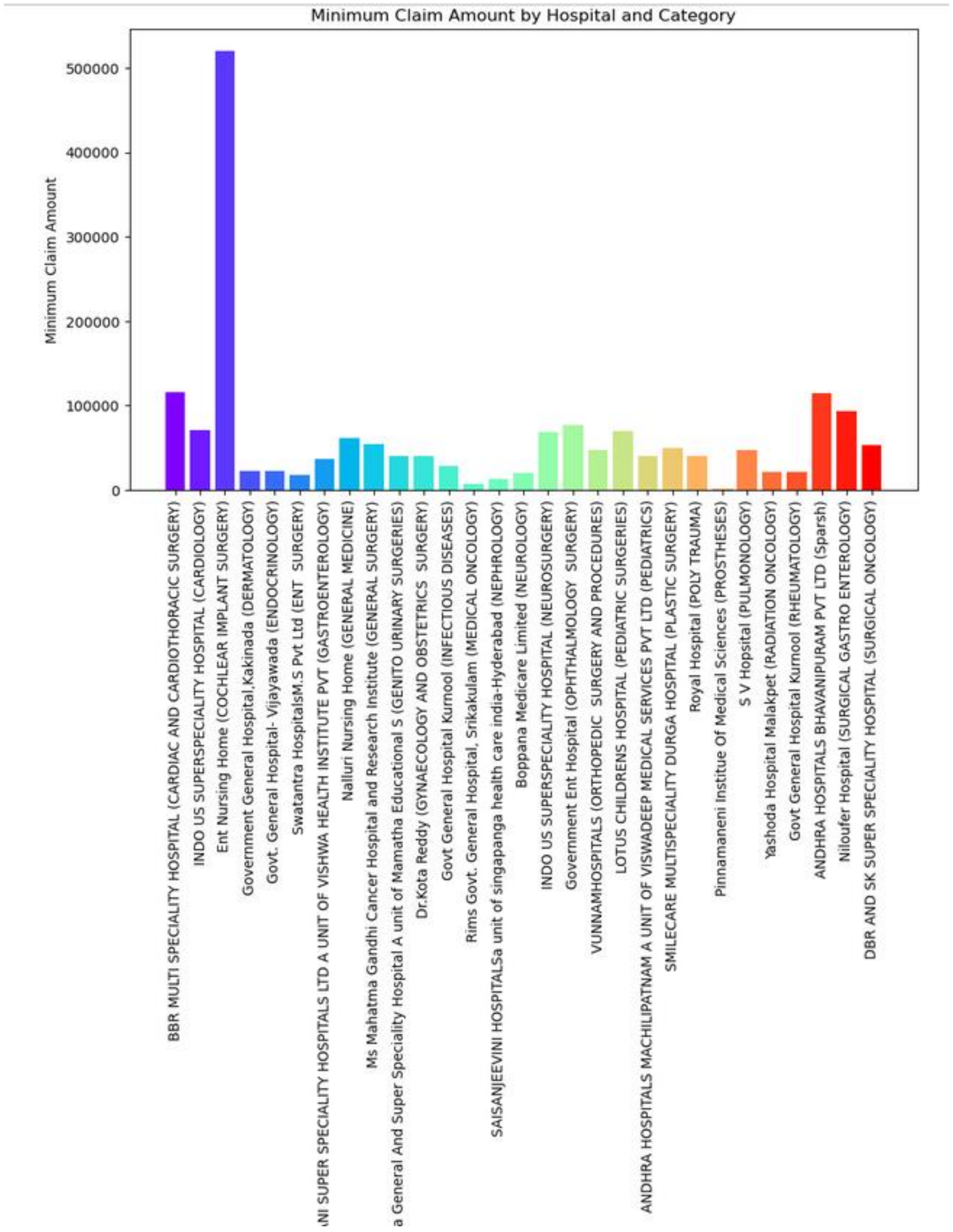
4. **Joining Tables**: The query joins the fact table with the dimension tables based on specific keys. It uses an INNER_JOIN to connect the fact table with the DIM_CATEGORY table using the category_id from the fact table and the category_key from the dimension table. Similarly, it connects the fact table with the DIM_MORTALITY table using the mortality_id from both tables.

5. **Grouping Results**: Finally, it groups the results by the CATEGORY_NAME from the DIM_CATEGORY table. This means that the query will return one row for each category, with columns for the category name, the count of mortality cases, and the total count of records for that category.

3. **Cheapest Hospitals for each Category of Disease**



```
import pandas as pd
import matplotlib.pyplot as plt
import pyodbc
import numpy as np

# Replace connection parameters with your own details
server = 'APINP-ELPTYIXVW'
database = 'capstoneprojectdata'
username = 'INFA_REP'
password = 'INFA_REP'

# Establish a connection
conn = pyodbc.connect('DRIVER={SQL Server};SERVER='+server+';DATABASE='+database+';UID='+username+';PWD='+ password)

# Define your SQL query as a string
sql_query = """
SELECT
   CATEGORY_NAME,
   HOSP_NAME,
   MIN_CLAIM_AMOUNT
FROM (
    SELECT
        C.CATEGORY_NAME,
        H.HOSP_NAME,
        MIN(F.CLAIM_AMOUNT) AS MIN_CLAIM_AMOUNT,
        ROW_NUMBER() OVER (PARTITION BY C.CATEGORY_NAME ORDER BY MIN(F.CLAIM_AMOUNT) DESC) AS rn
    FROM
        DIM_CATEGORY C
    LEFT JOIN
        Fact_table_ F ON C.category_key = F.category_id
    LEFT JOIN
        DIM_HOSPITAL H ON F.hospital_id = H.HOSP_KEY
    GROUP BY
        C.CATEGORY_NAME,
        H.HOSP_NAME
) AS SubQuery
WHERE rn = 1
ORDER BY
    CATEGORY_NAME,
    MIN_CLAIM_AMOUNT DESC;
"""

# Execute the query and fetch the results into a pandas DataFrame
df = pd.read_sql(sql_query, conn)

# Prepare the data for the bar chart
# Create custom labels for the bars that include the hospital name first and the category name in brackets
labels = ['{} ({})'.format(hosp, cat) for hosp, cat in zip(df['HOSP_NAME'], df['CATEGORY_NAME'])]



# Plot the data as a bar chart with multicolored bars
plt.figure(figsize=(10, 6))
plt.bar(labels, df['MIN_CLAIM_AMOUNT'], color=colors)
plt.xlabel('Hospital Name and Category Name')
plt.ylabel('Minimum Claim Amount')
plt.title('Minimum Claim Amount by Hospital and Category')
plt.xticks(rotation=90) # Rotate x-axis labels for better readability
plt.show()
```


Minimum Claim Amount by Hospital and Category
Minimum Claim Amount
0
100000
200000
300000
400000
500000
BBR MULTI SPECIALITY HOSPITAL (CARDIAC AND CARDIOTHORACIC SURGERY)
INDO US SUPERSPECIALITY HOSPITAL (CARDIOLOGY)
Ent Nursing Home (COCHLEAR IMPLANT SURGERY)
Government General Hospital,Kakinada (DERMATOLOGY)
Govt. General Hospital- Vijayawada (ENDOCRINOLOGY)
Swatantra HospitalsM.S Pvt Ltd (ENT SURGERY)
NI SUPER SPECIALITY HOSPITALS LTD A UNIT OF VISHWA HEALTH INSTITUTE PVT (GASTROENTEROLOGY)
Nalluri Nursing Home (GENERAL MEDICINE)
Ms Mahatma Gandhi Cancer Hospital and Research Institute (GENERAL SURGERY)
a General And Super Speciality Hospital A unit of Mamatha Educational S (GENITO URINARY SURGERIES)
Dr.Kota Reddy (GYNAECOLOGY AND OBSTETRICS SURGERY)
Govt General Hospital Kurnool (INFECTIOUS DISEASES)
Rims Govt. General Hospital, Srikakulam (MEDICAL ONCOLOGY)
SAISANJEEVINI HOSPITALSa unit of singapanga health care india-Hyderabad (NEPHROLOGY)
Boppana Medicare Limited (NEUROLOGY)
INDO US SUPERSPECIALITY HOSPITAL (NEUROSURGERY)
Government Ent Hospital (OPHTHALMOLOGY SURGERY)
VUNNAMHOSPITALS (ORTHOPEDIC SURGERY AND PROCEDURES)
LOTUS CHILDRENS HOSPITAL (PEDIATRIC SURGERIES)
ANDHRA HOSPITALS MACHILIPATNAM A UNIT OF VISWADEEP MEDICAL SERVICES PVT LTD (PEDIATRICS)
SMILECARE MULTISPECIALITY DURGA HOSPITAL (PLASTIC SURGERY)
Royal Hospital (POLY TRAUMA)
Pinnamaneni Institue Of Medical Sciences (PROSTHESES)
S V Hopsital (PULMONOLOGY)
Yashoda Hospital Malakpet (RADIATION ONCOLOGY)
Govt General Hospital Kurnool (RHEUMATOLOGY)
ANDHRA HOSPITALS BHAVANIPURAM PVT LTD (Sparsh)
Niloufer Hospital (SURGICAL GASTRO ENTEROLOGY)
DBR AND SK SUPER SPECIALITY HOSPITAL (SURGICAL ONCOLOGY)

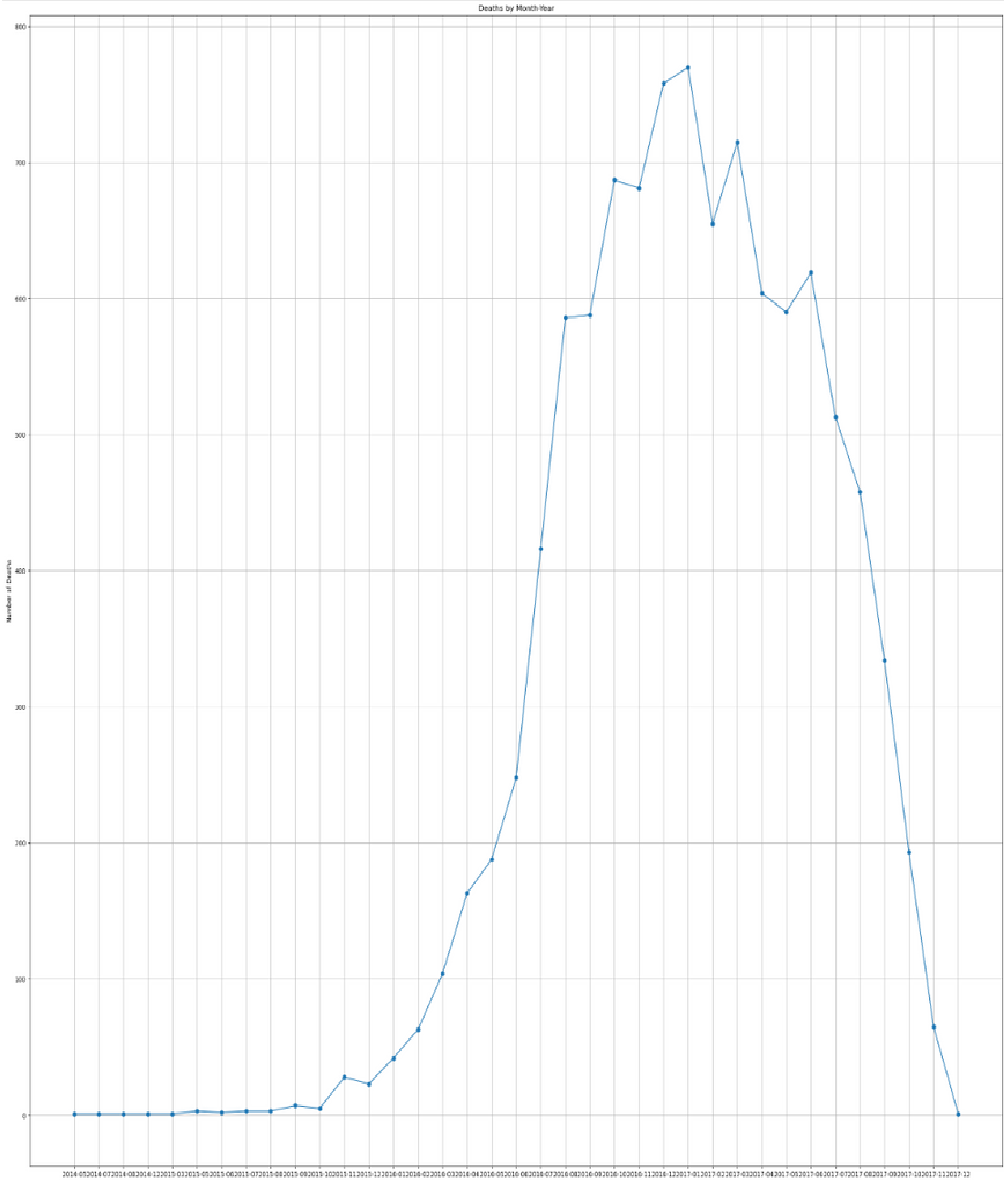
```
SELECT
    CATEGORY_NAME,
    HOSP_NAME,
    MIN_CLAIM_AMOUNT
FROM (
    SELECT
        CATEGORY_NAME,
        HOSP_NAME,
        MIN(CLAIM_AMOUNT) AS MIN_CLAIM_AMOUNT,
        ROW_NUMBER() OVER (PARTITION BY CATEGORY_NAME ORDER BY
MIN(CLAIM_AMOUNT) DESC) AS rn
    FROM
        STG_Andhra_Health_data
    GROUP BY
        CATEGORY_NAME,
        HOSP_NAME
) AS SubQuery
WHERE rn = 1
ORDER BY
    CATEGORY_NAME,
    MIN_CLAIM_AMOUNT DESC;
```

1. **Subquery**: The query begins with a subquery that selects CATEGORY_NAME, HOSP_NAME, and the minimum CLAIM_AMOUNT for each combination of CATEGORY_NAME and HOSP_NAME. It also assigns a row number to each row within each partition of CATEGORY_NAME, ordered by the minimum claim amount in descending order. This is achieved using the ROW_NUMBER function with PARTITION BY CATEGORY NAME and ORDER BY MINIMUM CLAIM AMOUNT. The PARTITION_BY clause divides the result set into partitions based on CATEGORY_NAME, and the ORDER_BY clause within the OVER clause specifies the order of rows within each partition. The ROW_NUMBER() function then assigns a unique row number to each row within its partition, starting from 1 for the first row in each partition.

2. **Filtering**: The outer query filters the results of the subquery to include only the rows where rn is 1. This effectively selects the hospital with the lowest claim amount for each category, as the row with the lowest claim amount within each category will have a row number of 1.

3. **Ordering**: Finally, the outer query orders the results by CATEGORY_NAME and MIN_CLAIM_AMOUNT in descending order. This ensures that the output is sorted first by category name and then by the minimum claim amount within each category, from highest to lowest.

4. **No of Deaths Year-Month wise**



```python
import pandas as pd
import matplotlib.pyplot as plt
import pyodbc
import numpy as np

# Database connection parameters
server = 'APINP-ELPTYIXVW'
database = 'capstoneprojectdata'
username = 'INFA_REP'
password = 'INFA_REP'

# Establish a connection to the database
conn = pyodbc.connect('DRIVER={SQL Server};SERVER='+server+';DATABASE='+database+';UID='+username+';PWD='+ password)

# SQL query to fetch mortality data
sql_query = """
SELECT
    [Mortality_Date],
    COUNT(*) AS DeathCount FROM
DIM_DATE D
LEFT JOIN
    Fact_table_ F ON D.DATE_ID = F.DATE_id
    LEFT JOIN
    DIM_Mortality M ON M.Mortality_id = F.mortality_id
WHERE
    [Mortality_Status] = 'Y'
GROUP BY
    [Mortality_Date]
ORDER BY
    [Mortality_Date]
"""

# Execute the query and fetch the results into a pandas DataFrame
df = pd.read_sql(sql_query, conn)

# Convert MORTALITY_DATE to datetime and extract month and year
df['Mortality_Date'] = pd.to_datetime(df['Mortality_Date'])
df['MonthYear'] = df['Mortality_Date'].dt.to_period('M')

# Convert Period objects to strings for plotting
df['MonthYear'] = df['MonthYear'].astype(str)

# Aggregate data by MonthYear
aggregated_data = df.groupby('MonthYear')['DeathCount'].sum().reset_index()

# Plot the aggregated data as a time series graph with a single line connecting the points
plt.figure(figsize=(30, 30))
plt.plot(aggregated_data['MonthYear'], aggregated_data['DeathCount'], marker='o', linestyle='-')

plt.xlabel('Month-Year')
plt.ylabel('Number of Deaths')
plt.title('Deaths by Month-Year')
plt.grid(True)
plt.show()

# Close the database connection
conn.close()
```

```
SELECT
   [Mortality_Date],
   COUNT(*) AS DeathCount FROM
DIM_DATE D
LEFT JOIN
   Fact_table_ F ON D.DATE_ID = F.DATE_id
         LEFT JOIN
   DIM_Mortality M ON M.Mortality_id = F.mortality_id
WHERE
   [Mortality_Status] = 'Y'
GROUP BY
   [Mortality_Date]
ORDER BY
   [Mortality_Date]
```

1. **SELECT Clause**: The query starts by selecting two columns:
   - Mortality_date: This is the date of mortality, which will be used to group the results.
   - COUNT(*) Death_Count: This counts the total number of records for each date, effectively counting the number of deaths.

2. **FROM Clause**: The query specifies the main table () from which to retrieve the data.

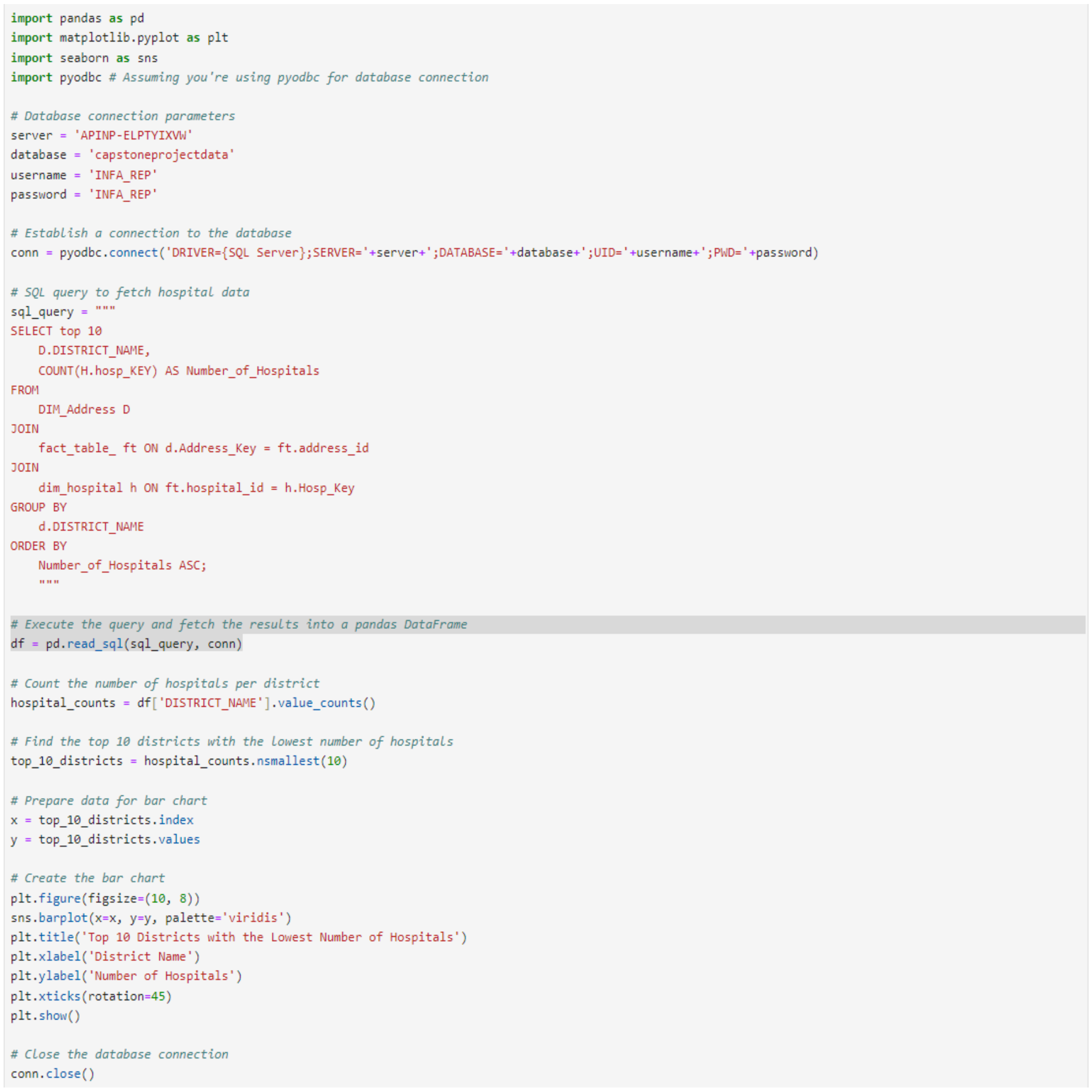
3. **LEFT JOIN Clauses**: The query uses two operations to connect the DIM_DATE table with the and tables. The first connects the DIM_DATE table with the using the from the table and the Date_id from the Fact_table_ . The second Left Join connects the Fact_table_ with the Dim_mortality table using the Mortality_id from the Fact_table_ and the Mortality_id from the DIM_Mortality table. The use of LEFT JOIN ensures that all records from the DIM_DATE table are included in the result set, even if there are no matching records in the other tables.

4. **WHERE Clause**: This clause filters the records to include only those where the Mortality_Status is 'Y', indicating a death.

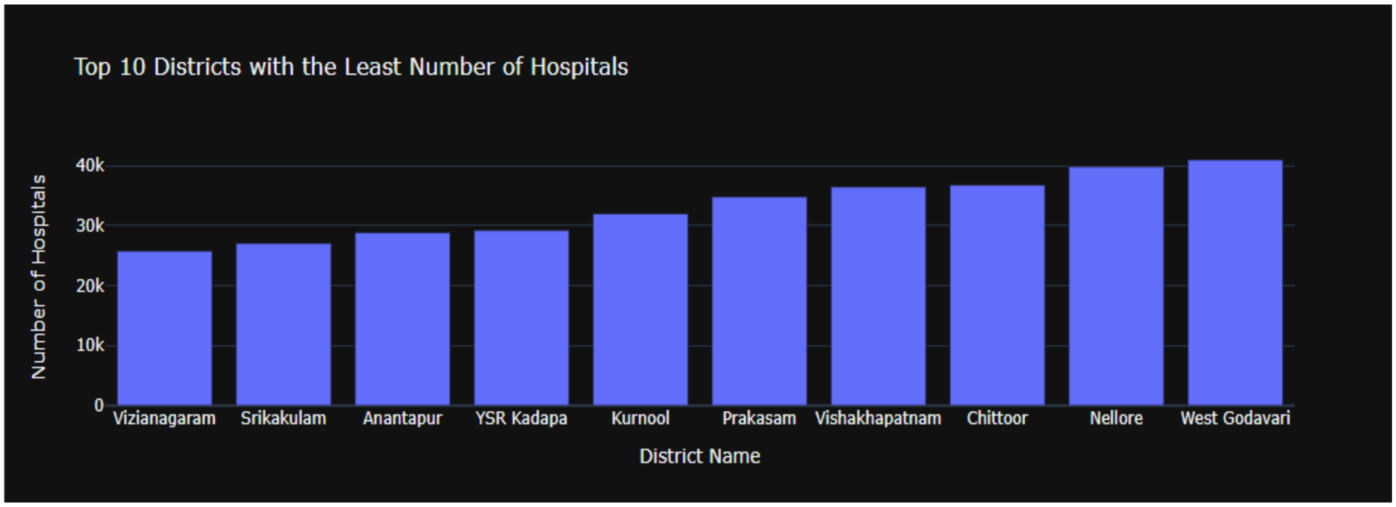
5. **GROUP BY Clause**: The query groups the results by the Mortality_Date, meaning it will return one row for each date, with the count of deaths for that date.

6. **ORDER BY Clause**: Finally, the query orders the results by Mortality_Date, ensuring that the output is sorted by date.

5. Top 10 Districts with least Number of Hospitals



```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import pyodbc # Assuming you're using pyodbc for database connection

# Database connection parameters
server = 'APINP-ELPTYIXVW'
database = 'capstoneprojectdata'
username = 'INFA_REP'
password = 'INFA_REP'

# Establish a connection to the database
conn = pyodbc.connect('DRIVER={SQL Server};SERVER='+server+';DATABASE='+database+';UID='+username+';PWD='+password)

# SQL query to fetch hospital data
sql_query = """
SELECT top 10
    D.DISTRICT_NAME,
    COUNT(H.hosp_KEY) AS Number_of_Hospitals
FROM
    DIM_Address D
JOIN
    fact_table_ ft ON d.Address_Key = ft.address_id
JOIN
    dim_hospital h ON ft.hospital_id = h.Hosp_Key
GROUP BY
    d.DISTRICT_NAME
ORDER BY
    Number_of_Hospitals ASC;
    """

# Execute the query and fetch the results into a pandas DataFrame
df = pd.read_sql(sql_query, conn)

# Count the number of hospitals per district
hospital_counts = df['DISTRICT_NAME'].value_counts()

# Find the top 10 districts with the lowest number of hospitals
top_10_districts = hospital_counts.nsmallest(10)

# Prepare data for bar chart
x = top_10_districts.index
y = top_10_districts.values

# Create the bar chart
plt.figure(figsize=(10, 8))
sns.barplot(x=x, y=y, palette='viridis')
plt.title('Top 10 Districts with the Lowest Number of Hospitals')
plt.xlabel('District Name')
plt.ylabel('Number of Hospitals')
plt.xticks(rotation=45)
plt.show()

# Close the database connection
conn.close()
```



```
sql_query = """
SELECT top 10
    D.DISTRICT_NAME,
    COUNT(H.hosp_KEY) AS Number_of_Hospitals
FROM
    DIM_Address D
JOIN
    fact_table_ ft ON d.Address_Key = ft.address_id
JOIN
    dim_hospital h ON ft.hospital_id = h.Hosp_Key
GROUP BY
    d.DISTRICT_NAME
ORDER BY
    Number_of_Hospitals ASC;
    """
```

Data Analysis with Python

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
from sklearn.preprocessing import StandardScaler
# Load your dataset
# Assuming the dataset is named 'data.csv' and it's in the same directory as
your script
data = pd.read_csv('data.csv')
# Preprocessing: Handling missing values, encoding categorical variables,
etc.
# This step depends on your specific dataset
# For simplicity, let's assume all necessary preprocessing has been done
# Feature selection: Select the features you want to use for prediction
# Here, we're using all columns except 'mortality_rate' as features
X = data.drop('mortality_rate', axis=1)
y = data['mortality_rate']
# Standardize the features to have zero mean and unit variance
scaler = StandardScaler()
X = scaler.fit_transform(X)
# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
# Initialize the Random Forest Classifier
clf = RandomForestClassifier(n_estimators=100, random_state=42)
# Train the model
clf.fit(X_train, y_train)
# Make predictions
y_pred = clf.predict(X_test)
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
# Print classification report and confusion matrix for a detailed evaluation
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

The Python code provided earlier uses several key components and algorithms to build a Random Forest Classifier for predicting mortality rates. Let's delve into the details of each part of the code, including the algorithms and methods used:

1. Data Preprocessing

- Pandas: The pandas library is used to load and manipulate the dataset. It provides a DataFrame structure that is ideal for handling tabular data, such as CSV files.

2. Feature Selection

- Drop: The drop method is used to remove the target variable (mortality_date) from the dataset, leaving only the features to be used for prediction.

3. Standardization

- StandardScaler: The StandardScale from sklearnpreproccessing is used to standardize the features by removing the mean and scaling to unit variance. This step is crucial for many machine learning algorithms, including Random Forest, as it ensures that all features contribute equally to the model, regardless of their original scale.

4. Model Training and Evaluation

- **RandomForestClassifier**: The core of the code is the Random Forest Classifier, an ensemble learning method that constructs a multitude of decision trees at training time and outputs the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. It is known for its robustness, accuracy, and ability to handle a large number of features and prevent overfitting

- **Train-Test Split**: The train_split function from sklearn.modelselection is used to split the dataset into a training set and a test set. This is a common practice in machine learning to evaluate the model's performance on unseen data.

- **fit**: The fit method is used to train the Random Forest Classifier on the training data.

- **predict**: The predict method is used to make predictions on the test data.

- **Evaluation Metrics**:
    - **Accuracy**: Measures the proportion of correct predictions among the total number of cases examined.
    - **Classification Report**: Provides a detailed breakdown of the model's performance, including precision, recall, f1-score, and support for each class.
    - **Confusion Matrix**: A table that describes the performance of a classification model, showing the true positives, true negatives, false positives, and false negatives.

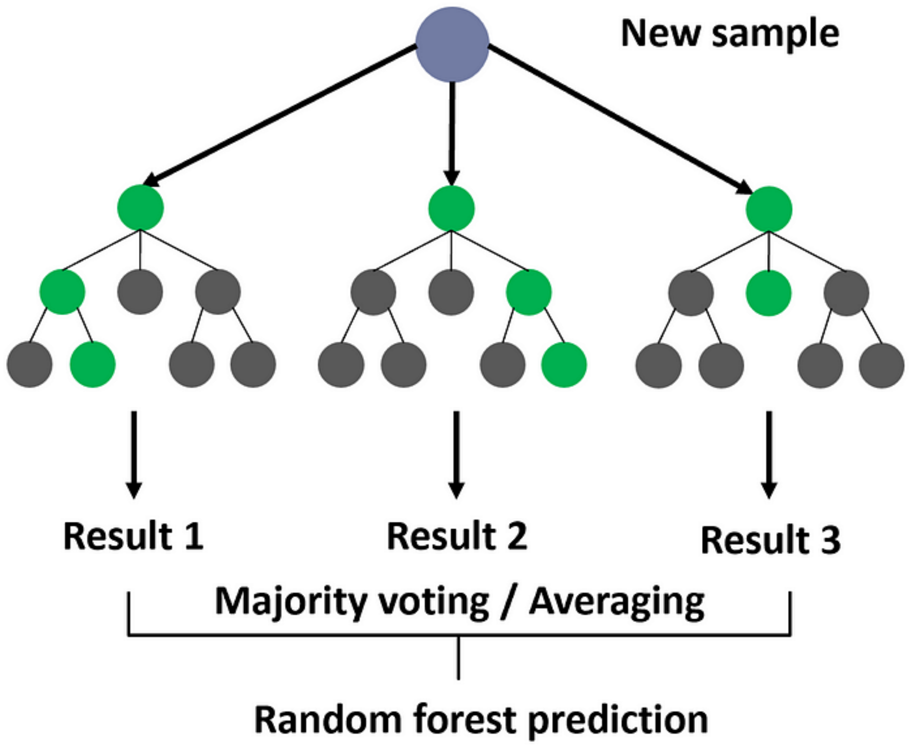
Key Algorithms and Methods

1. How Random Forest Works

- **Ensemble of Decision Trees**: Random Forest constructs multiple decision trees during the training phase. Each tree is built from a random subset of the data, and at each node, a random subset of features is considered for splitting. This process introduces randomness, which helps in reducing overfitting and improving the model's generalization ability .

- **Bagging Method**: The algorithm uses the bagging method, where each tree is trained on a different subset of the data. The final prediction is made by averaging the predictions of all the trees. This method helps in reducing variance and improving the stability of the model .

- **Feature Importance**: Random Forest can also provide insights into the importance of different features in making predictions. This is because each tree in the forest may use different features for splitting, and the overall importance of a feature can be determined by how often it is used across all trees .



Advantages of Random Forest

- **Reduced Overfitting**: By averaging the predictions of multiple trees, Random Forest reduces the risk of overfitting to the training data. This makes it a more reliable model for making predictions on unseen data .

- **High Accuracy**: Random Forest is known for its high accuracy in both classification and regression tasks. It can handle large datasets efficiently and provides a high level of accuracy in predicting outcomes .

- **Handling Different Feature Types**: Unlike some other algorithms, Random Forest can handle a variety of feature types, including binary, categorical, and numerical data. This flexibility makes it a versatile tool for many different types of prediction problems .

- **Ease of Use**: Random Forest is relatively easy to implement and understand. It is a good choice for anyone looking to develop a model quickly, as its simplicity makes it difficult to build a "bad" model. It also provides a good indicator of the importance of features in the model.

Limitations of Random Forest

- **Computationally Intensive**: While Random Forest is generally fast, it can be computationally intensive, especially with a large number of trees or a very large dataset. This might make it less suitable for real-time applications or very large datasets .

- **Less Interpretability**: Compared to single decision trees, Random Forest models are less interpretable. While it's possible to understand the importance of features, the model's decision-making process is more complex and harder to visualize

2. Standardization is a preprocessing step in machine learning that transforms the features of a dataset to have a mean of 0 and a standard deviation of 1. This process is crucial for many machine learning algorithms, as it ensures that all features contribute equally to the model, regardless of their original scale. Here's a detailed explanation of standardization, including its mathematical formulation and practical applications:

Mathematical Formulation

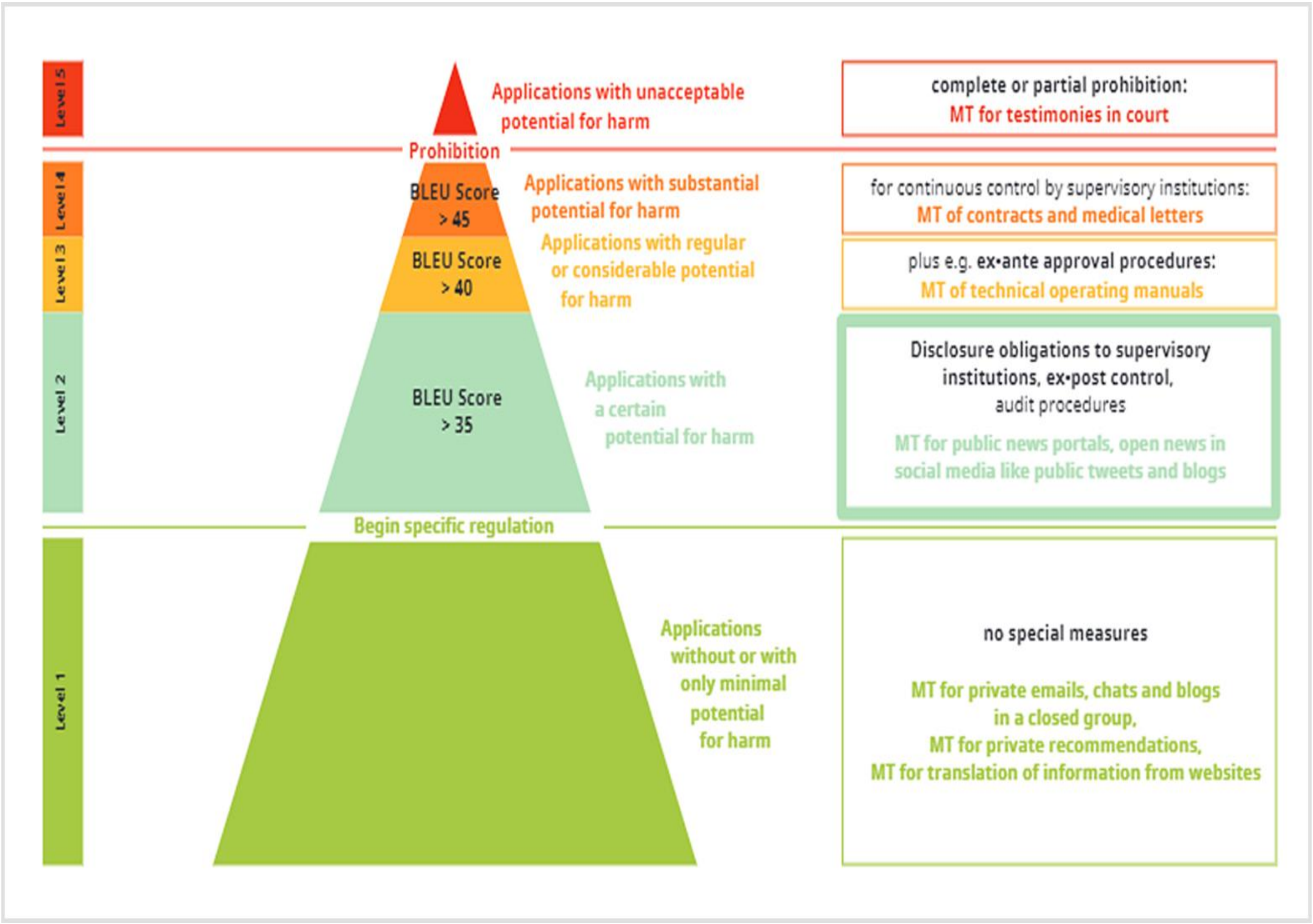
Standardization is performed using the formula:

[z = \frac{x - \mu}{\sigma}]

where:

- (x) is the original feature value,
- (\mu) is the mean of the feature values,
- (\sigma) is the standard deviation of the feature values.

If, the mean is subtracted from the feature values, and if , the feature values are divided by the standard deviation. This allows for flexibility in the standardization process .


Level 5
Applications with unacceptable potential for harm
complete or partial prohibition:
MT for testimonies in court
Prohibition
Level 4
BLEU Score > 45
Applications with substantial potential for harm
for continuous control by supervisory institutions:
MT of contracts and medical letters
Level 3
BLEU Score > 40
Applications with regular or considerable potential for harm
plus e.g. ex-ante approval procedures:
MT of technical operating manuals
Level 2
BLEU Score > 35
Applications with a certain potential for harm
Disclosure obligations to supervisory institutions, ex-post control, audit procedures
MT for public news portals, open news in social media like public tweets and blogs
Begin specific regulation
Level 1
Applications without or with only minimal potential for harm
no special measures
MT for private emails, chats and blogs in a closed group,
MT for private recommendations,
MT for translation of information from websites

Practical Applications

- Centering and Scaling: Standardization centers the data around 0 and scales it to have a standard deviation of 1. This is done independently for each feature, ensuring that all features are on the same scale. This is particularly important for algorithms that assume features are normally distributed, such as Support Vector Machines (SVM) with the RBF kernel or linear models with L1 and L2 regularizers .

- Handling Outliers: StandardScaler is sensitive to outliers. Features with outliers may scale differently from others, potentially affecting the model's performance. However, it's still a valuable preprocessing step for many algorithms .

- Sparse Data: StandardScaler can be applied to sparse matrices by setting. This avoids breaking the sparsity structure of the data, which is important for preserving memory efficiency

Example Usage

Here's an example of how to use StandardScaler in Python with scikit-learn:

```
from sklearn.preprocessing import StandardScaler

# Create an instance of the scaler
scaler = StandardScaler()

# Fit on training data
scaler.fit(X_train)

# Transform both training and test data
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In this example, StandaardScalar is first imported from. An instance of StandardScalar is created and fitted to the training data (X_Train). This step calculates the mean and standard deviation for each feature in the training set. The training and test sets are then transformed using the transform method, applying the scaling transformation based on the calculated mean and standard deviation .

Importance of Standardization

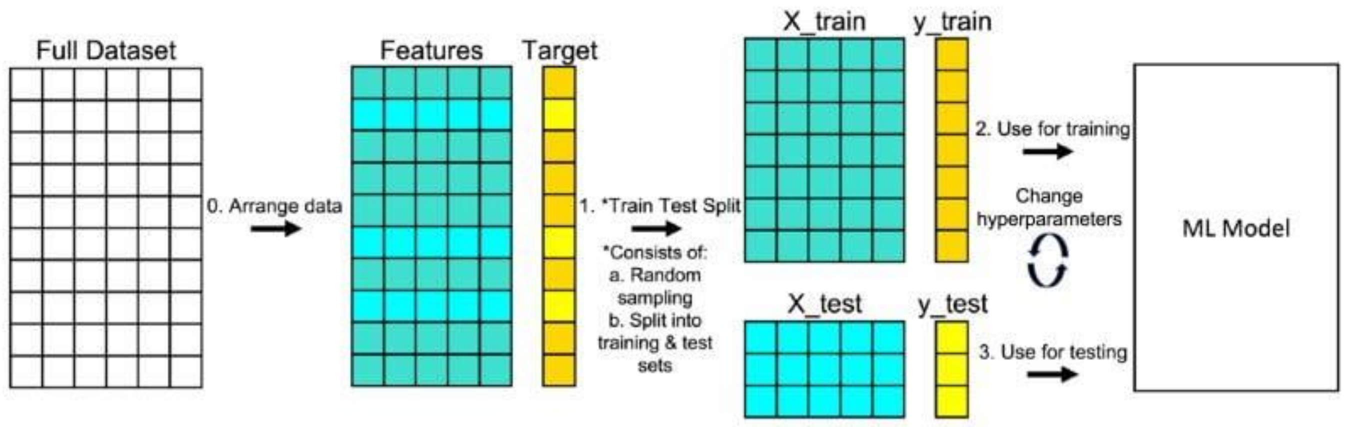
Standardization is essential for many machine learning algorithms because it ensures that all features are on the same scale, preventing features with larger scales from dominating the model. This is particularly important for algorithms that use distance measures, such as k-nearest neighbors (KNN) or support vector machines (SVM), where the scale of the features can significantly affect the model's performance .

In summary, standardization is a critical preprocessing step that helps ensure the fairness and effectiveness of machine learning models by scaling features to have a mean of 0 and a standard deviation of 1.

3. **Train-Test Split**: The train-test split is a fundamental technique in machine learning for evaluating the performance of a model on unseen data. It involves dividing a dataset into two subsets: a training set and a test set. The training set is used to train the model, while the test set is used to evaluate the model's performance. This method is crucial for assessing how well a model can generalize to new, unseen data, which is essential for real-world applications.

Importance of Data Splitting

- Model Evaluation and Validation: The train-test split allows for unbiased evaluation and validation of a model. It ensures that the model's performance is assessed on data it has not seen during training, providing a more accurate indication of its predictive capabilities .
- Preventing Overfitting and Underfitting: By separating the data into training and test sets, the risk of overfitting (where the model performs well on the training data but poorly on new data) and underfitting (where the model fails to capture the underlying patterns in the data) is reduced. This is because the model is trained on a larger portion of the data and tested on a separate portion, which helps in identifying its ability to generalize .



Procedure

1. Dataset Division: The dataset is divided into two subsets. The first subset, known as the training set, is used to fit the model. The second subset, referred to as the test set, is used to evaluate the model's performance. This division is typically done randomly to ensure that both subsets are representative of the original dataset .
2. Model Training: The model is trained on the training dataset. This involves adjusting the model's parameters to minimize the difference between the model's predictions and the actual values in the training set.
3. Model Evaluation: The model's performance is evaluated on the test dataset. This involves comparing the model's predictions on the test set with the actual values. The evaluation metrics, such as accuracy, precision, recall, or F1 score, are used to quantify the model's performance .

Appropriate Use Cases

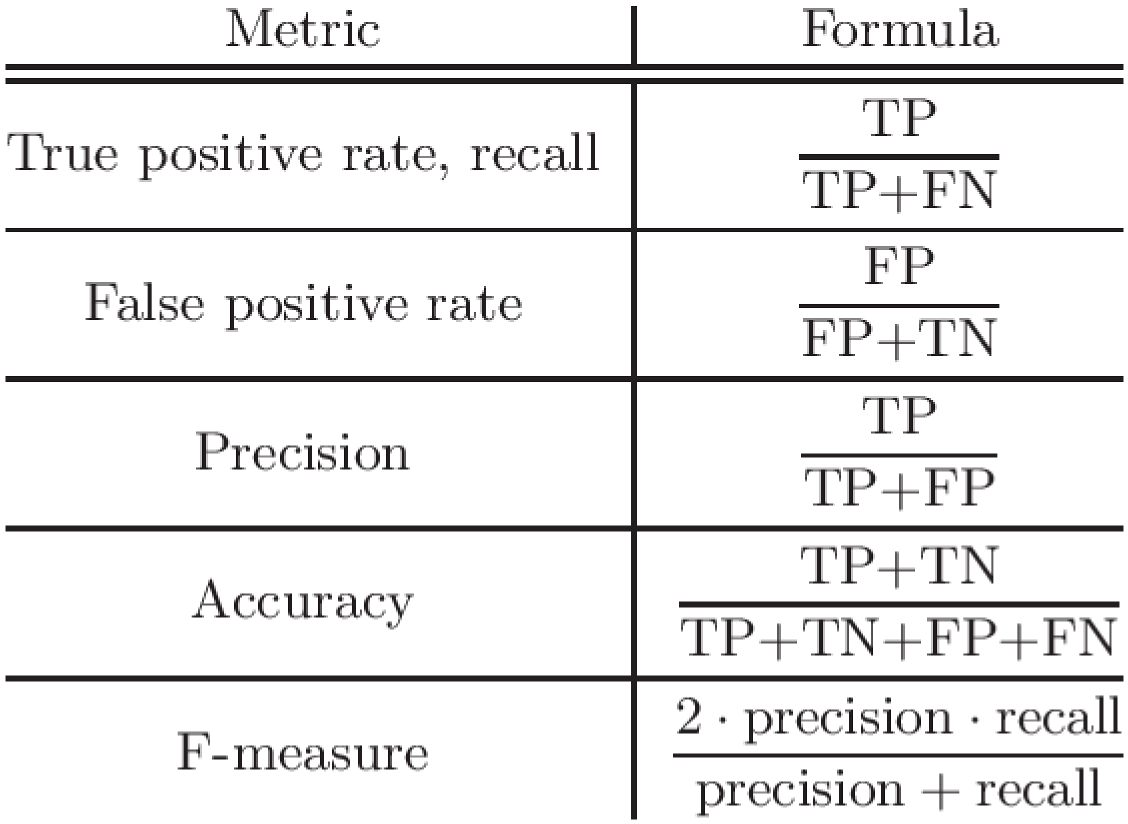
- **Large Datasets**: The train-test split is most appropriate when there is a sufficiently large dataset available. This ensures that both the training and test datasets are representative of the problem domain and contain enough data for the model to learn effectively and for the evaluation to be meaningful .
- **Computational Efficiency**: For models that are computationally expensive to train, the train-test split provides a quick way to estimate model performance. It is also suitable for projects that require a quick estimate of model performance .

Limitations

- **Small Datasets**: The train-test split may not be suitable for small datasets. In such cases, the model may not learn effectively from the limited data available in the training set, and the evaluation may not be reliable. In these scenarios, other evaluation techniques, such as k-fold cross-validation, might be more appropriate .
- **Data Leakage**: There is a risk of data leakage when the test set is used to inform the model's training process. This can lead to overly optimistic performance estimates. To mitigate this, it's important to ensure that the test set is truly unseen by the model during training .

In summary, the train-test split is a critical technique in machine learning for evaluating model performance on unseen data. It helps in assessing how well a model can generalize to new data, ensuring that the model's performance is not overly optimistic or pessimistic. However, it's important to use this technique judiciously, considering the size of the dataset and the computational resources available.

4. **Evaluation Metrics**: The key metrics include accuracy, precision, recall, and the confusion matrix. Each of these metrics offers a different perspective on the model's performance, making them valuable for understanding the model's effectiveness in various scenarios.



Metric	Formula
True positive rate, recall	$\frac{\text{TP}}{\text{TP+FN}}$
False positive rate	$\frac{\text{FP}}{\text{FP+TN}}$
Precision	$\frac{\text{TP}}{\text{TP+FP}}$
Accuracy	$\frac{\text{TP+TN}}{\text{TP+TN+FP+FN}}$
F-measure	$\frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$

Accuracy

- Definition: Accuracy is the ratio of the number of correct predictions to the total number of predictions made. It is a straightforward metric that measures the overall correctness of the model's predictions.
- Use Cases: Accuracy is particularly useful when dealing with balanced datasets where the model's performance across all classes is equally important. However, it may not be the best metric for imbalanced datasets where some classes are significantly underrepresented .

Precision

- Definition: Precision is the ratio of true positives (TP) to the sum of true positives and false positives (FP). It measures the proportion of positive identifications that were actually correct.

- Use Cases: Precision is crucial in situations where false positives are particularly undesirable, such as in medical diagnoses or spam detection. It helps in understanding how well the model can correctly identify positive instances without incorrectly flagging negative instances as positive [3].

Recall

- Definition: Recall, also known as sensitivity or true positive rate, is the ratio of true positives to the sum of true positives and false negatives (FN). It measures the proportion of actual positives that were identified correctly.
- Use Cases: Recall is important in scenarios where it is more critical to capture all positive instances, such as in fraud detection or identifying rare diseases. It helps in understanding how well the model can identify positive instances without missing any [3].

Confusion Matrix

- Definition: A confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known. It provides a more detailed view of the model's performance by showing the number of true positives, true negatives, false positives, and false negatives.
- Use Cases: The confusion matrix is particularly useful for understanding the model's performance across different classes, especially in multi-class classification problems. It allows for the calculation of various metrics, including accuracy, precision, and recall, and can be used to identify which classes the model is performing well on and which it is struggling with [4].

F1 Score

- Definition: The F1 score is the harmonic mean of precision and recall. It provides a single metric that balances the trade-off between precision and recall, making it a useful measure when both false positives and false negatives are important.
- Use Cases: The F1 score is particularly useful in situations where both precision and recall are important, such as in information retrieval, named entity recognition, and many other applications. It helps in understanding the overall effectiveness of the model in identifying positive instances without missing any

Chapter 3

3.1 Appendices

1. Linear Regression

Linear regression is used for predicting a continuous outcome variable (Y) based on one or more predictor variables (X).

```
from sklearn.linear_model import LinearRegression
from sklearn.datasets import make_regression

X, y = make_regression(n_samples=100, n_features=1, noise=0.1)
model = LinearRegression()
model.fit(X, y)
```

2. Logistic Regression

Logistic regression is used for binary classification problems.

```
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import make_classification

X, y = make_classification(n_samples=100, n_features=20, n_informative=2, n_redundant=10,
random_state=42)
model = LogisticRegression()
model.fit(X, y)
```

3. Decision Tree

Decision trees are used for both classification and regression tasks.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import load_iris

iris = load_iris()
model = DecisionTreeClassifier()
model.fit(iris.data, iris.target)
```

4. Random Forest

Random Forest is an ensemble method that uses multiple decision trees.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import load_iris

iris = load_iris()
model = RandomForestClassifier()
model.fit(iris.data, iris.target)
```

5. Support Vector Machine (SVM)

SVM is used for classification and regression analysis.

```
from sklearn.svm import SVC
from sklearn.datasets import load_iris

iris = load_iris()
model = SVC()
model.fit(iris.data, iris.target)
```

6. K-Nearest Neighbors (KNN)

KNN is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure.

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import load_iris

iris = load_iris()
model = KNeighborsClassifier()
model.fit(iris.data, iris.target)
```

7. Naive Bayes

Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong independence assumptions between the features.

```
from sklearn.naive_bayes import GaussianNB
from sklearn.datasets import load_iris

iris = load_iris()
model = GaussianNB()
model.fit(iris.data, iris.target)
```

8. Gradient Boosting

Gradient Boosting is an ensemble method that uses decision trees.

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.datasets import load_iris

iris = load_iris()
model = GradientBoostingClassifier()
model.fit(iris.data, iris.target)
```

9. AdaBoost

AdaBoost is another ensemble method that uses decision trees.

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.datasets import load_iris

iris = load_iris()
model = AdaBoostClassifier()
model.fit(iris.data, iris.target)
```

10. Neural Networks

Neural networks are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns.

```
from sklearn.neural_network import MLPClassifier
from sklearn.datasets import load_iris

iris = load_iris()
model = MLPClassifier()
model.fit(iris.data, iris.target)
```

Explanation

1. Linear Regression
Purpose: Predicts a continuous outcome variable based on one or more predictor variables.
Code Explanation: Generates a synthetic regression dataset using make_regression, then fits a linear regression model to the data using LinearRegression().fit().

2. Logistic Regression
Purpose: Used for binary classification problems.
Code Explanation: Creates a synthetic classification dataset with make_classification, then fits a logistic regression model to the data using LogisticRegression().fit().

3. Decision Tree
Purpose: Can be used for both classification and regression tasks.
Code Explanation: Loads the Iris dataset using load_iris(), then fits a decision tree classifier to the data using DecisionTreeClassifier().fit().

4. Random Forest
Purpose: An ensemble method that uses multiple decision trees.
Code Explanation: Similar to the decision tree example, it loads the Iris dataset and fits a random forest classifier to the data using RandomForestClassifier().fit().

5. Support Vector Machine (SVM)
Purpose: Used for classification and regression analysis.
Code Explanation: Uses the Iris dataset and fits an SVM classifier to the data using SVC().fit().

6. K-Nearest Neighbors (KNN)
Purpose: Classifies new cases based on a similarity measure.

Code Explanation: Uses the Iris dataset and fits a KNN classifier to the data using KNeighborsClassifier().fit().

7. Naive Bayes
Purpose: A family of simple probabilistic classifiers based on applying Bayes' theorem with strong independence assumptions between the features.
Code Explanation: Uses the Iris dataset and fits a Gaussian Naive Bayes classifier to the data using GaussianNB().fit().

8. Gradient Boosting
Purpose: An ensemble method that uses decision trees.
Code Explanation: Uses the Iris dataset and fits a gradient boosting classifier to the data using GradientBoostingClassifier().fit().

9. AdaBoost
Purpose: Another ensemble method that uses decision trees.
Code Explanation: Similar to gradient boosting, it uses the Iris dataset and fits an AdaBoost classifier to the data using AdaBoostClassifier().fit().

10. Neural Networks
Purpose: A set of algorithms, modeled loosely after the human brain, designed to recognize patterns.
Code Explanation: Uses the Iris dataset and fits a multi-layer perceptron (MLP) classifier to the data using MLPClassifier().fit().

Each of these algorithms serves a different purpose in machine learning, from predicting continuous outcomes to classifying data into categories. The scikit-learn library provides a unified interface for these algorithms, making it easier to implement and compare different models

ETL (Extract, Transform, Load) and ELT (Extract, Load, Transform) are two fundamental data integration processes that have been widely adopted in the data management and analytics landscape. Each method has its unique characteristics, advantages, and use cases, making them suitable for different scenarios depending on the specific requirements of data processing and analysis.

1.ETL (Extract, Transform, Load)

- Extract: Data is extracted from various source systems, which could include databases, files, APIs, or other data repositories.
- Transform: The extracted data is then transformed into a format that is suitable for analysis. This step often involves cleaning the data (removing duplicates, handling missing values), aggregating data from multiple sources, and applying business rules to ensure data quality and consistency.
- Load: Finally, the transformed data is loaded into a target system, typically a data warehouse or a database, where it can be analyzed and used for reporting and decision-making.

ETL is a traditional approach that has been around for a long time and is well-suited for scenarios where data needs to be cleaned and prepared before analysis. It is particularly useful when dealing with complex transformations that require significant processing power and when the target system is optimized for analytical queries .

2.ELT (Extract, Load, Transform)

- Extract: Similar to ETL, data is extracted from source systems.
- Load: The extracted data is loaded directly into the target system without any transformation. This approach leverages the scalable storage and computing resources of modern cloud platforms to retain all extracted data.
- Transform: Transformation is performed after the data is loaded into the target system. This can be done using the target system's capabilities, such as SQL queries or BI tools, to transform the data as needed for analysis.

ELT is a more modern approach that is beneficial for handling large volumes of data and when the transformation logic is complex. By loading data first and then transforming it, ELT allows for more efficient use of the target system's resources and can be more cost-effective, especially when using cloud-based storage and computing .

3.Reverse ETL

Reverse ETL is a process that extracts data from an analytical system or data warehouse and loads it back into operational systems or other downstream applications. This method serves as a bridge between analytical insights and operational actions, enabling organizations to leverage the valuable insights gained through analytics by pushing that information back into their operational systems in real-time or near-real-time. This allows for more immediate decision-making and automation based on the analyzed data [2].

Reverse ETL is particularly useful for closing the loop between analytics and operations, ensuring that insights are not left unused but immediately put into action. It facilitates real-time data synchronization between analytical systems and operational systems, helping businesses stay agile and responsive in today's fast-paced market landscape .

Key Differences

- Data Flow Direction: Traditional ETL moves data from source systems to target systems for analysis, while Reverse ETL moves analyzed or processed data from target systems back to source systems for action.
- Data Transformation and Extraction: ETL involves complex transformations of raw source data before loading it into a target system, while Reverse ETL focuses on extracting already transformed or processed data from a central repository and delivering it back to operational systems.
- Use Cases: ETL is suitable for scenarios requiring extensive data cleaning and preparation before analysis, while ELT is beneficial for handling large volumes of data and when the transformation logic is complex. Reverse ETL is ideal for leveraging analytical insights in real-time operational scenarios .

Understanding these differences is crucial for organizations looking to implement or optimize their data integration processes, as each method offers unique advantages depending on the specific requirements of data processing and analysis.

Data analysis is a comprehensive process that involves inspecting, cleansing, transforming, and modeling data to discover useful information, draw conclusions, and support decision-making. It is a multifaceted process that empowers organizations to make informed decisions, predict trends, and improve operational efficiency. Data analysis can be categorized into four main types: descriptive, diagnostic, predictive, and prescriptive analysis, each serving a unique purpose and providing different insights .

1.Descriptive Analysis

Descriptive analysis is the starting point for any analytic reflection. It aims to answer the question of what happened by ordering, manipulating, and interpreting raw data from various sources to turn it into valuable insights for an organization. This type of analysis is essential for presenting insights in a meaningful way, organizing data, and preparing it for further investigations. However, it does not predict future outcomes or answer why something happened .

2.Diagnostic Analysis

Diagnostic data analytics empowers analysts and executives by helping them gain a firm contextual understanding of why something happened. It is used to determine the cause of a particular event or outcome, providing insights into the reasons behind the data. This analysis is crucial for pinpointing the exact ways of tackling an issue or challenge, as it combines descriptive and diagnostic analysis to understand both how and why something occurred .

3.Predictive Analysis

Predictive analysis uses past data to predict future outcomes. It involves statistical modeling and machine learning techniques to forecast future trends, behaviors, or events based on historical data. This type of analysis is particularly useful in fields like finance, marketing, and healthcare, where forecasting future trends can help in planning and decision-making .

4.Prescriptive Analysis

Prescriptive analysis suggests actions based on predictions. It goes beyond predictive analysis by not only forecasting future outcomes but also recommending actions that can be taken to achieve desired outcomes. This type of analysis is used in strategic planning and decision-making processes, where it helps in determining the best course of action to achieve specific goals .

5.Additional Data Analysis Techniques

Beyond these four main types, there are numerous techniques used in data analysis, each with its unique purpose and application. Some of the most commonly used techniques include:

- Exploratory Analysis: Used to explore data relationships and find connections, generate hypotheses, and solutions for specific problems. It is particularly useful in data mining .
- Regression Analysis: A statistical method used to understand the relationship between dependent and independent variables. It helps in predicting the value of a dependent variable based on the value of an independent variable .
- Monte Carlo Simulation: A computational technique used to model the probability of different outcomes in a process that cannot easily be predicted due to the intervention of random variables .
- Factor Analysis: A statistical method used to reduce the number of variables under study by obtaining a set of fewer variables that are uncorrelated and that successively maximize the variance explained by the original set of variables .

- Cohort Analysis: A method used to study the behavior of groups of individuals over time. It is often used in marketing to understand customer behavior and retention .
- Cluster Analysis: A technique used to group similar data points together based on their characteristics. It is often used in market segmentation and customer segmentation .
- Time Series Analysis: A statistical technique that deals with time series data, or trend analysis. It is used to analyze data points sequentially connected in time .
- Sentiment Analysis: A method used to determine the sentiment or emotion behind a piece of text. It is often used in social media monitoring and brand management to gauge public opinion .

Each of these techniques and methods has its unique application and is chosen based on the specific requirements of the data analysis task at hand. Understanding these different types and techniques is crucial for effectively interpreting data and making informed decisions

Data ingestion is a critical process in the data analytics pipeline, involving the collection, cleansing, transforming, and integration of data from various sources into a single system for analysis. It is the first step in analytics-related data pipelines, where data is collected, loaded, and transformed for insights. The process of data ingestion can be categorized into several methods, each with its unique characteristics, benefits, and use cases.

1.Real-Time Ingestion

Real-time ingestion involves streaming data into a data warehouse or analytics system in real-time. This method is particularly useful for applications that require immediate insights, such as fraud detection, real-time analytics, and monitoring systems. Real-time ingestion leverages cloud-based systems that can ingest data quickly, store it in the cloud, and then release it to users almost immediately. This approach ensures that data is collected without delay, allowing analysts to work with fresh data .

2.Batch Ingestion

Batch ingestion involves collecting large amounts of raw data from various sources and processing it later. This method is used when there is a need to order a large amount of information before processing it all at once. Batch ingestion is suitable for scenarios where the volume of data is not too high, or when the data does not require immediate analysis. It is often used for historical data analysis, where the focus is on processing and analyzing data over a longer period .

3.Hybrid Ingestion

Hybrid ingestion combines both real-time and batch ingestion methods. This approach is beneficial for organizations that need to handle both immediate and historical data. Hybrid ingestion allows for the flexibility to process data as it arrives (real-time) and also to batch process data for long-term analysis. This method ensures that data is available for immediate analysis while also providing the option for comprehensive analysis over time .

4.Manual vs. Automated Ingestion

There are two main approaches to data ingestion: manual and automated. Manual data ingestion involves manually coding a data pipeline, which can be time-consuming and prone to errors. Automated data ingestion, on the other hand, uses data integration platforms to streamline the process. Automated ingestion solutions, such as Fivetran, facilitate automated data ingestion using pre-built connectors, which can connect to sources in minutes and help construct a reliable data pipeline without any code. This approach ensures that data is collected without delay and that analysts have fresh data to work with .

Benefits of Data Ingestion

- Timely Data: Real-time ingestion ensures that data is available for immediate analysis, enabling organizations to make quicker decisions.
- Scalability: Automated ingestion solutions can handle large volumes of data efficiently, making them suitable for growing businesses.
- Flexibility: Hybrid ingestion provides the flexibility to process data as it arrives and also to batch process data for long-term analysis.
- Reduced Data Silos: Data ingestion helps in breaking down data silos by collecting data from various sources into a centralized storage system like a cloud data warehouse or a data lake .

Challenges of Data Ingestion

- Data Volume: Handling large volumes of data can be challenging, especially when dealing with real-time data streams.
- Data Quality: Ensuring the accuracy and consistency of data as it is ingested is crucial, but maintaining data quality can be challenging.
- Complexity: The complexity of data ingestion processes can increase with the number of data sources and the volume of data being ingested .

Understanding these different data ingestion techniques and their benefits and challenges is crucial for organizations looking to leverage data effectively for analytics and decision-making.

3.2 Future Work

The future of your project, which involves performing ETL (Extract, Transform, Load) operations on a database containing data about different hospitals, diseases, and surgeries, is promising and multifaceted. ETL processes are crucial in the healthcare sector for several reasons, and the future of such projects is likely to be shaped by advancements in data management, analytics, and artificial intelligence (AI).

Data Centralization and Standardization

- Data Centralization: ETL processes facilitate the centralization of data from various sources into a single, unified location. This centralization is essential for healthcare organizations as it improves efficiency by making data easier to access and use. It also supports the integration of legacy data with new information, enhancing the overall data quality and usability .
- Standardization: ETL ensures that data is converted into a common format, which is crucial for analytics and AI operations. These technologies require large collections of standardized data to function effectively. By standardizing data through ETL, healthcare organizations can leverage AI for diagnosis tools, research algorithms, decision-making, and even surgery planning .

Enhanced Analytics and Decision-Making

- Improved Analytics: With data centralized and standardized, healthcare organizations can perform more sophisticated analytics. This includes identifying patterns, trends, and insights that can inform clinical practices, patient care, and research. ETL processes ensure that the data used for analytics is accurate, reliable, and retains its contextual information, such as historical data and metadata .
- Informed Decision-Making: The insights gained from analytics can lead to more informed decision-making processes. This can range from optimizing patient care to improving operational efficiency in hospitals. ETL's role in ensuring data accuracy and integrity supports these decision-making processes by providing reliable data upon which to base decisions .

Compliance and Regulatory Adherence

- Compliance and Regulatory Adherence: Healthcare data must meet strict standards for accuracy and compliance with regulations. ETL processes ensure that data integrity is maintained throughout the data lifecycle, supporting compliance and enabling sound business decisions. This is particularly important in the healthcare sector, where adherence to regulations is critical .

Future Trends and Opportunities

- Artificial Intelligence and Machine Learning: The future of healthcare data management and analytics is likely to be shaped by advancements in AI and machine learning. ETL processes will play a crucial role in preparing data for these technologies, enabling healthcare organizations to leverage AI for predictive analytics, personalized medicine, and more.
- Interoperability and Interconnectivity: As healthcare systems become more interconnected, the need for ETL processes to integrate data from various sources will continue to grow. This includes data from electronic health records (EHRs), health information exchanges (HIEs), and other healthcare IT systems. ETL will be essential for ensuring that these disparate data sources can be effectively integrated and analyzed

3.2 CONCLUSION

The conclusion for your project, which involves performing ETL operations on a database containing data about different hospitals, diseases, and surgeries, highlights the transformative impact of ETL on healthcare data management and its future potential.

Transformative Impact

- Streamlined Data Management: The ETL process has centralized and standardized patient data, reducing the need for manual data entry and integration. This has not only saved time but also improved data accuracy, enhancing patient care and operational efficiencies .
- Enhanced Research and Decision Making: By providing access to a rich dataset, ETL has fostered advancements in healthcare research and data-driven decision-making. This has led to improved

operational efficiencies and the ability to conduct meaningful medical studies, positioning the organization at the forefront of healthcare innovation.

- Improved Data Security and Compliance: The secure data repository implemented through ETL has ensured data privacy and compliance with regulatory requirements, safeguarding patient information and supporting the organization's adherence to industry standards .
- Enhanced Interoperability: ETL has enhanced interoperability between various healthcare systems and databases, facilitating data exchange and collaboration among healthcare stakeholders. This has been crucial for integrating data from diverse sources and adhering to industry standards, enabling seamless access to comprehensive patient records .

Future Potential

- Dynamic ETL (D-ETL): The future of ETL in healthcare is likely to be shaped by the implementation of Dynamic ETL (D-ETL), which supports a flexible and transparent process to transform and load health data into a target data model. This approach lowers technical barriers, promoting the advancement of comparative effectiveness research using secondary electronic health data .
- Scalability and Customization: D-ETL automates part of the ETL process through scalable, reusable, and customizable code, while retaining manual aspects that require knowledge of complex coding syntax. This provides the flexibility required for the ETL of heterogeneous data and transparency of transformation logic, essential for implementing ETL conventions across clinical research sharing networks

3.3 REFERENCES

1.https://www.indiastat.com/andhra-pradesh-state/data/health

2.https://www.kaggle.com/models?query=&task=&lang=&id=&datatype=&framework=&license=&size=&fine-tunable=&minUsabilityRating=&publisher=

3.https://books.google.com/books?hl=en&lr=&id=OuQwgShUdGAC&oi=fnd&pg=PR7&dq=health+data&ots=6s2JgRk_5G&sig=0DoOL7dLCAPawIeHWUvK8gJt-nw

4.https://link.springer.com/article/10.1007/s00392-016-1025-6

5.https://content.iospress.com/articles/informatica/inf1097

6.https://heinonline.org/hol-cgi-bin/get_pdf.cgi?handle=hein.journals/dpencrim27&section=21