BOSTON UNIVERSITY

CS779
Advanced Database Management

TERM PROJECT REPORT

Hospital Administration Data WareHouse

- Vaidehi Shah

Prof. Jack Polnar

BU MET, Spring 2023

INTRODUCTION

This project aims to develop a tailored data warehousing solution for Hospital Administration, enabling data-driven decision-making, streamlined processes, and enhanced efficiency. Implementing a tailored data warehousing solution empowers the Hospital Administration with a centralized platform for efficient data access, analysis, and interpretation. It enhances patient care, financial operations, and resource allocation.

The Hospital Administration has several key areas requiring comprehensive reporting and analysis:

1. **Patient Data**: By leveraging the data warehousing solution, the Hospital Administration gains access to a comprehensive patient database that enables a deeper understanding of patient demographics and medical history.
2. **Financial Operations Optimization**: This analysis aids in identifying potential areas for improvement, streamlining financial operations, and optimizing revenue generation.
3. **Resource Allocation & Planning**: By analyzing various operational metrics, such as patient flow and hospital provider performance, the administration can make informed decisions regarding staffing, equipment, and facility utilization, leading to optimized resource allocation and planning.

DESIGNING HOSPITAL ADMINISTRATION SCHEMA

The Hospital Administration Schema is designed to efficiently manage and organize data within a hospital facility. It consists of several tables that capture information related to patients, clinical notes and reports, hospital providers and billing. The schema ensures data integrity and establishes relationships between entities through foreign key constraints. Indexes are created to optimize data retrieval performance.

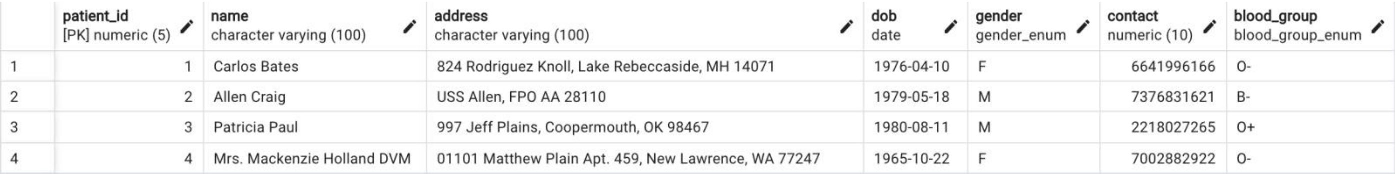
The core tables in the schema include:

TABLE NAME	DESCRIPTION	ATTRIBUTES
patient_data	Stores patient information	patient ID, name, address, date of birth, gender, contact number, and blood group
medical_history	Captures the medical history of patients	surgeries, allergies, and medical conditions
diagnosis	Contains details of patient diagnoses	diagnosis ID, patient ID, doctor ID, nurse ID, visit date, symptoms, case details, and clinical notes
treatment	Tracks information about patient treatments	treatment ID, diagnosis ID, begin date, and end date.
medicines	Stores information about prescribed medications	medicine ID, medicine name, and dosage instructions.
outcomes	Records treatment outcomes	outcome ID, treatment ID, and recovery status
vital_signs	Captures vital signs measurements for patients	vital signs ID, diagnosis ID, blood pressure, oxygen level, temperature, heart rate, weight, and height
lab_results	Stores lab test results	result ID, diagnosis ID, test names, and corresponding results
insurance_claims	Contains information about insurance claims	insurance ID, patient ID, insurance provider name, insurance type, and expiry date
billing_info	Tracks billing details for treatments	treatment ID, bill number, bill date, amount, payment type, payment status, and payment date
doctors	Stores information about doctors	doctor ID, name, contact, speciality
nurses	Stores information about nurses	nurse ID, name, contact
tests	Store information about specific medical tests	test ID, test_name

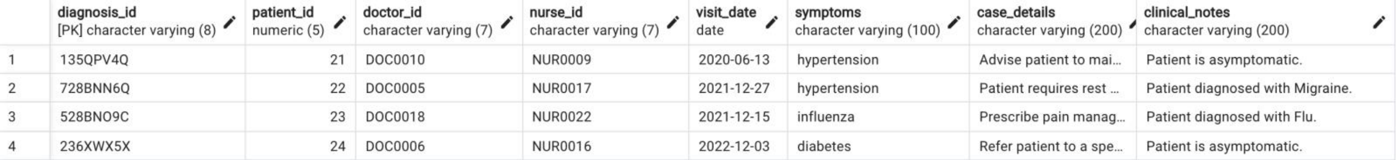
Foreign key constraints establish relationships between entities, ensuring referential integrity. Indexes are created on key columns to optimize data retrieval speed.

There are two bridge tables namely **diagnosis_bridge_tests** and **treatments_bridge_meds** in order to establish one-to-many relations.

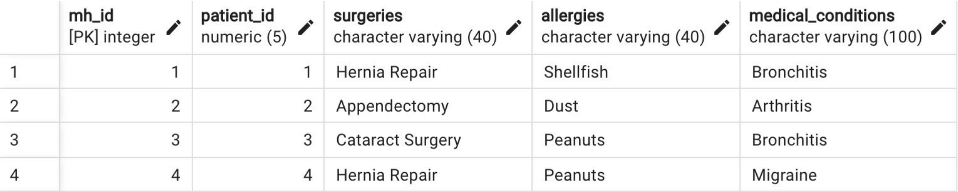
Overall, the Hospital Administration Schema provides a comprehensive and organized structure for managing hospital administration data, facilitating efficient information retrieval and decision-making within the hospital facility.



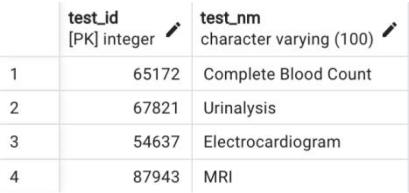
	patient_id [PK] numeric (5)	name character varying (100)	address character varying (100)	dob date	gender gender_enum	contact numeric (10)	blood_group blood_group_enum
1	1	Carlos Bates	824 Rodriguez Knoll, Lake Rebeccaside, MH 14071	1976-04-10	F	6641996166	O-
2	2	Allen Craig	USS Allen, FPO AA 28110	1979-05-18	M	7376831621	B-
3	3	Patricia Paul	997 Jeff Plains, Coopermouth, OK 98467	1980-08-11	M	2218027265	O+
4	4	Mrs. Mackenzie Holland DVM	01101 Matthew Plain Apt. 459, New Lawrence, WA 77247	1965-10-22	F	7002882922	O-



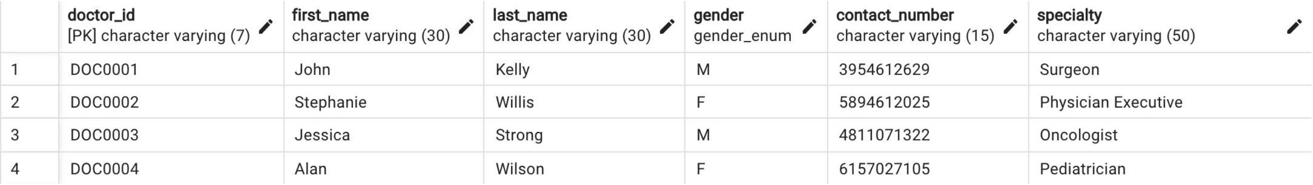
	diagnosis_id [PK] character varying (8)	patient_id numeric (5)	doctor_id character varying (7)	nurse_id character varying (7)	visit_date date	symptoms character varying (100)	case_details character varying (200)	clinical_notes character varying (200)
1	135QPV4Q	21	DOC0010	NUR0009	2020-06-13	hypertension	Advise patient to mai...	Patient is asymptomatic.
2	728BNN6Q	22	DOC0005	NUR0017	2021-12-27	hypertension	Patient requires rest ...	Patient diagnosed with Migraine.
3	528BNO9C	23	DOC0018	NUR0022	2021-12-15	influenza	Prescribe pain manag...	Patient diagnosed with Flu.
4	236XWX5X	24	DOC0006	NUR0016	2022-12-03	diabetes	Refer patient to a spe...	Patient is asymptomatic.



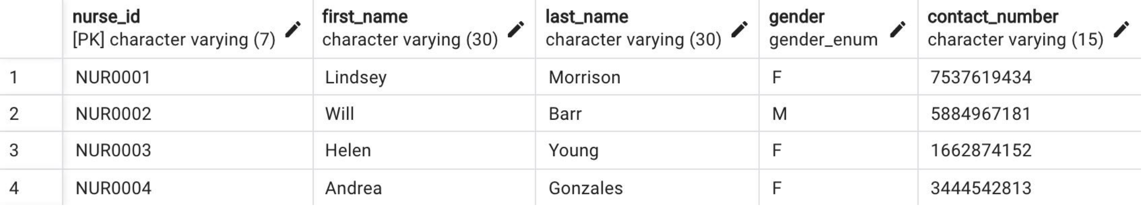
	mh_id [PK] integer	patient_id numeric (5)	surgeries character varying (40)	allergies character varying (40)	medical_conditions character varying (100)
1	1	1	Hernia Repair	Shellfish	Bronchitis
2	2	2	Appendectomy	Dust	Arthritis
3	3	3	Cataract Surgery	Peanuts	Bronchitis
4	4	4	Hernia Repair	Peanuts	Migraine



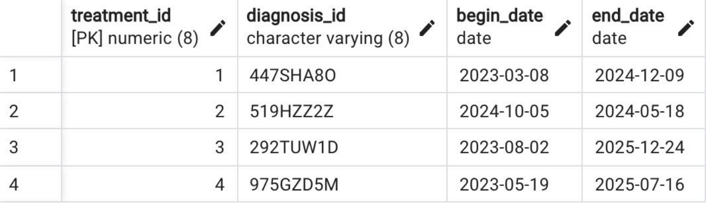
	test_id [PK] integer	test_nm character varying (100)
1	65172	Complete Blood Count
2	67821	Urinalysis
3	54637	Electrocardiogram
4	87943	MRI



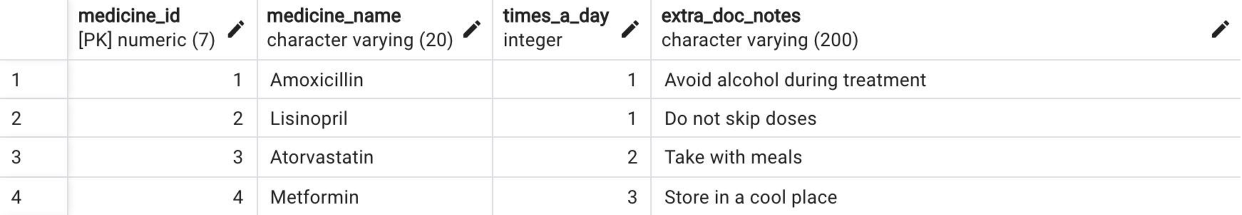
	doctor_id [PK] character varying (7)	first_name character varying (30)	last_name character varying (30)	gender gender_enum	contact_number character varying (15)	specialty character varying (50)
1	DOC0001	John	Kelly	M	3954612629	Surgeon
2	DOC0002	Stephanie	Willis	F	5894612025	Physician Executive
3	DOC0003	Jessica	Strong	M	4811071322	Oncologist
4	DOC0004	Alan	Wilson	F	6157027105	Pediatrician



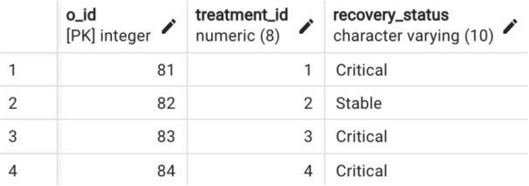
	nurse_id [PK] character varying (7)	first_name character varying (30)	last_name character varying (30)	gender gender_enum	contact_number character varying (15)
1	NUR0001	Lindsey	Morrison	F	7537619434
2	NUR0002	Will	Barr	M	5884967181
3	NUR0003	Helen	Young	F	1662874152
4	NUR0004	Andrea	Gonzales	F	3444542813



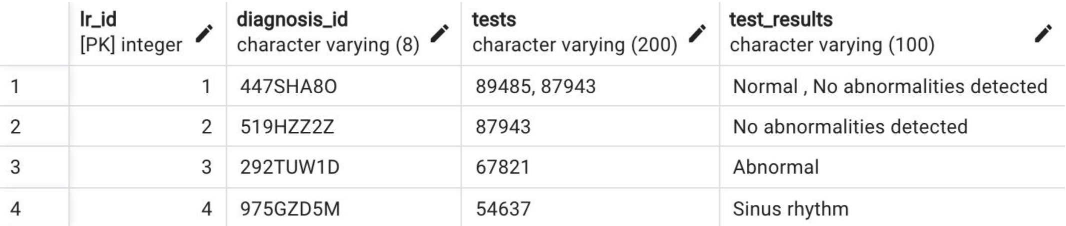
	treatment_id [PK] numeric (8)	diagnosis_id character varying (8)	begin_date date	end_date date
1	1	447SHA8O	2023-03-08	2024-12-09
2	2	519HZZ2Z	2024-10-05	2024-05-18
3	3	292TUW1D	2023-08-02	2025-12-24
4	4	975GZD5M	2023-05-19	2025-07-16



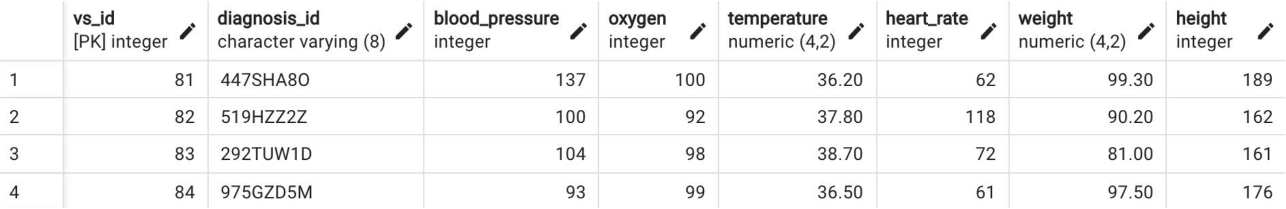
	medicine_id [PK] numeric (7)	medicine_name character varying (20)	times_a_day integer	extra_doc_notes character varying (200)
1	1	Amoxicillin	1	Avoid alcohol during treatment
2	2	Lisinopril	1	Do not skip doses
3	3	Atorvastatin	2	Take with meals
4	4	Metformin	3	Store in a cool place



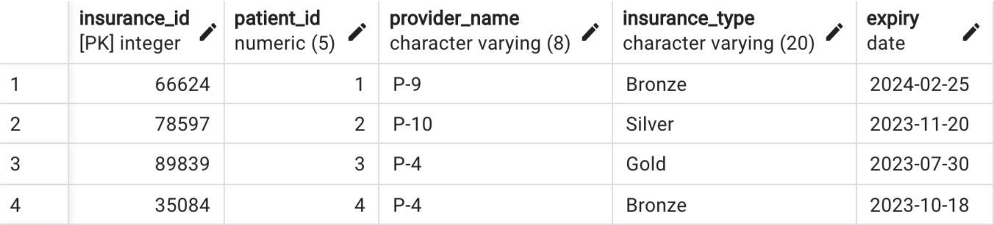
	o_id [PK] integer	treatment_id numeric (8)	recovery_status character varying (10)
1	81	1	Critical
2	82	2	Stable
3	83	3	Critical
4	84	4	Critical



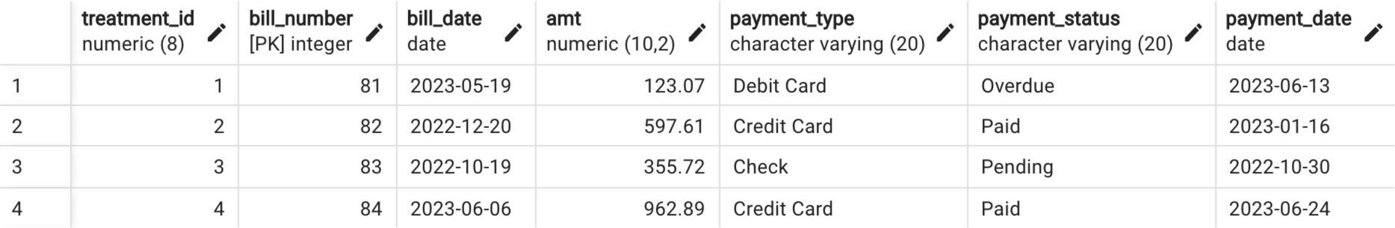
	lr_id [PK] integer	diagnosis_id character varying (8)	tests character varying (200)	test_results character varying (100)
1	1	447SHA8O	89485, 87943	Normal , No abnormalities detected
2	2	519HZZ2Z	87943	No abnormalities detected
3	3	292TUW1D	67821	Abnormal
4	4	975GZD5M	54637	Sinus rhythm



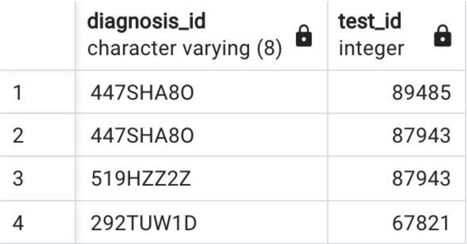
	vs_id [PK] integer	diagnosis_id character varying (8)	blood_pressure integer	oxygen integer	temperature numeric (4,2)	heart_rate integer	weight numeric (4,2)	height integer
1	81	447SHA8O	137	100	36.20	62	99.30	189
2	82	519HZZ2Z	100	92	37.80	118	90.20	162
3	83	292TUW1D	104	98	38.70	72	81.00	161
4	84	975GZD5M	93	99	36.50	61	97.50	176



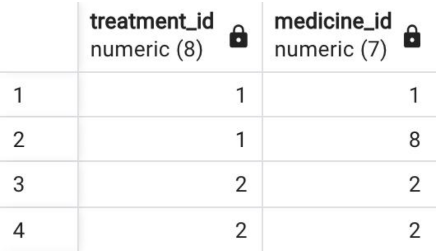
	insurance_id [PK] integer	patient_id numeric (5)	provider_name character varying (8)	insurance_type character varying (20)	expiry date
1	66624	1	P-9	Bronze	2024-02-25
2	78597	2	P-10	Silver	2023-11-20
3	89839	3	P-4	Gold	2023-07-30
4	35084	4	P-4	Bronze	2023-10-18



	treatment_id numeric (8)	bill_number [PK] integer	bill_date date	amt numeric (10,2)	payment_type character varying (20)	payment_status character varying (20)	payment_date date
1	1	81	2023-05-19	123.07	Debit Card	Overdue	2023-06-13
2	2	82	2022-12-20	597.61	Credit Card	Paid	2023-01-16
3	3	83	2022-10-19	355.72	Check	Pending	2022-10-30
4	4	84	2023-06-06	962.89	Credit Card	Paid	2023-06-24

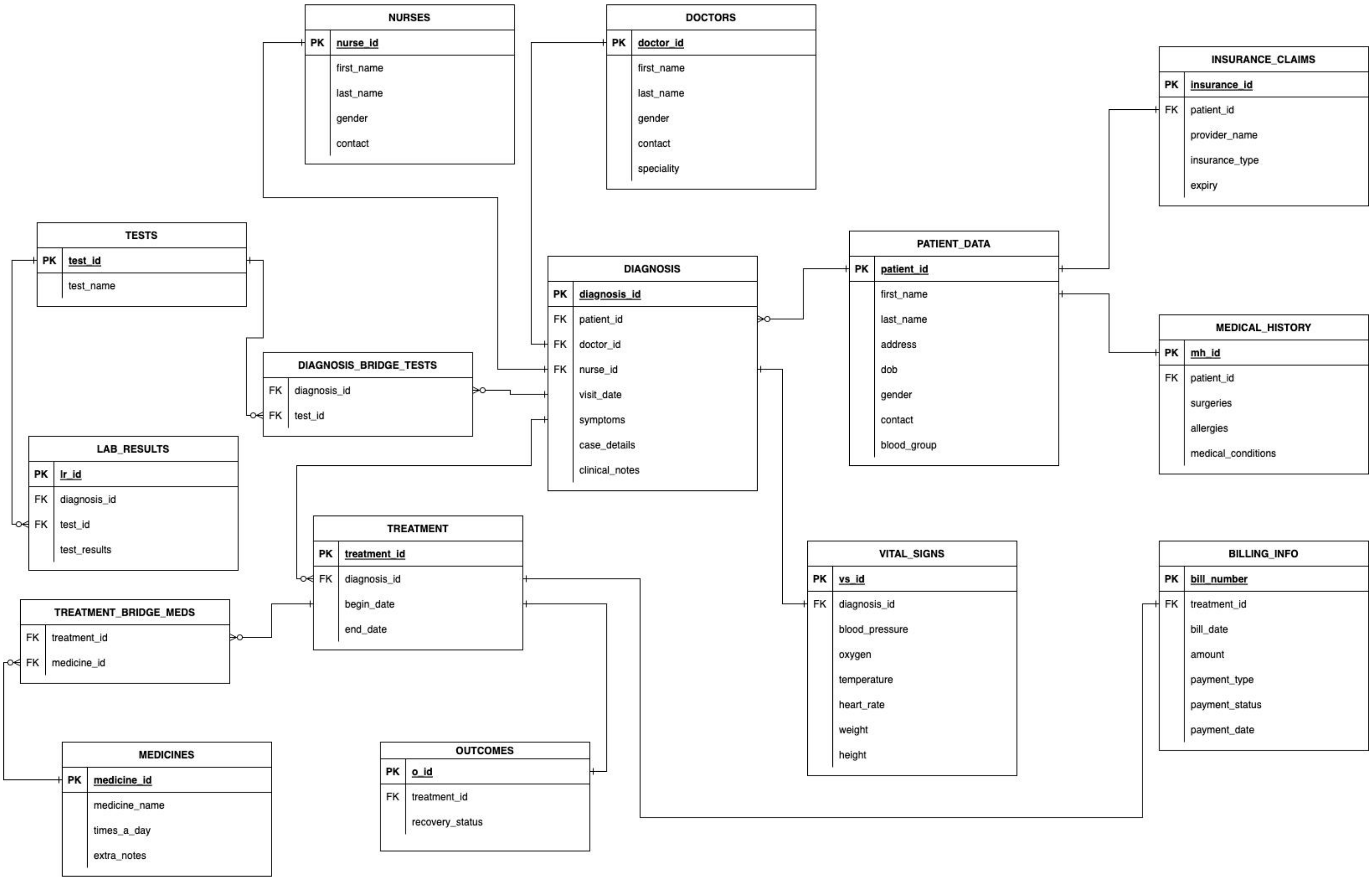


	diagnosis_id character varying (8)	test_id integer
1	447SHA8O	89485
2	447SHA8O	87943
3	519HZZ2Z	87943
4	292TUW1D	67821



	treatment_id numeric (8)	medicine_id numeric (7)
1	1	1
2	1	8
3	2	2
4	2	2

ENTITY-RELATION DIAGRAM



DATA GENERATION & DATABASE CREATION

Here is a comprehensive overview of the implemented Python code which utilizes loops and conditional statements to ensure the desired volume and characteristics of the data to be generated for the database. The code leverages various libraries, including

1. **Faker:** Used for generating fake data, such as names, addresses, and other relevant information.
2. **csv:** Employed for handling CSV files, facilitating efficient data manipulation and storage.
3. **random:** Utilized to generate random numbers, enabling diverse data distribution.
4. **pandas:** Employed for creating DataFrames, a convenient data structure, and for exporting data to Excel.

This section outlines the approach employed to generate data, organize it into multiple tables in CSV format, and subsequently import the tables into a database using create table and insert value queries.

I have worked on PostgreSQL on pgAdmin Tool for this project. To import the generated data into a database, I used the create table and insert value queries in SQL. The create table query facilitates the creation of appropriate tables within the database including the constraints, and the insert value query allows for the population of these tables with the generated data. By executing these queries, the data is seamlessly imported into the database, ready for further analysis and utilization.

DATA MODELING

1. Identify Attributes and Entities: The main entities and attributes that I will be focusing on in this section of datawarehousing are as follows:

ENTITIES	ATTRIBUTES
patient_data	patient_id, name, contact, address, gender, dob, blood_group
diagnosis	diagnosis_id, visit_date
medical_history	allergies, surgeries, medical_conditions
vital_signs	blood_pressure, oxygen, temperature, heart_rate, height, weight, bmi
lab_results	test_results
treatment	treatment_id
outcomes	recovery_status
doctors	doctor_id, name, contact, speciality
nurses	nurse_id, name, contact
insurance_claims	insurance_id, provider_name, insurance_type, expiry
billing_info	bill_number, bill_date, amt, payment_type, payment_status, payment_date, total_amt

2. Dimension & Fact Tables: Dimension tables and fact tables are important aspect of a data warehouse.

a) Dimension Tables: Dimension tables contain descriptive attributes that provide context and details about the data in a data warehouse. The tables in my data warehouse that are classified as dimension tables are: patient_data, medical_history, tests, doctors, nurses, diagnosis, outcomes.

b) Fact Tables: Fact tables contain the quantitative and measurable data that are the focus of analysis in a data warehouse. The tables in my data warehouse that are classified as fact tables are: treatment, vital_signs, lab_results, insurance_claims, billing_info.

3. Snowflake Dimensional Model: Based on my data modeling for the data warehouse, Snowflake Schema is the appropriate choice. In a snowflake schema, the dimension tables are structured hierarchically, with each level of the hierarchy represented by a separate table. This normalization process reduces data redundancy by removing repeating groups of attributes. The normalized tables are connected through foreign key relationships, creating a "snowflake" shape when viewed visually. It involves certain important dimension tables (diagnosis, patient_data, treatment) which are in turn connected to several other tables (medical_history, vital_signs, outcomes, doctors, nurses, etc.) to extract information.

Pros of Snowflake Schema:

a) Improved Data Integrity: Normalization reduces data redundancy, leading to improved data consistency and integrity.

b) Space Efficiency: By eliminating duplicate data, the snowflake schema can save storage space.

c) Flexibility: The hierarchical structure of the dimension tables allows for more complex relationships and accommodates additional attributes.

d) Scalability: Snowflake schemas are well-suited for large-scale data warehouses that require efficient storage and support for complex queries.

Cons of Snowflake Schema:

e) Increased Complexity: The normalized structure of the snowflake schema requires more joins to retrieve data, which can lead to increased query complexity and potential performance overhead.

f) Query Performance: Due to the additional joins involved, queries on a snowflake schema may take longer to execute compared to a star schema, especially for complex queries spanning multiple tables.

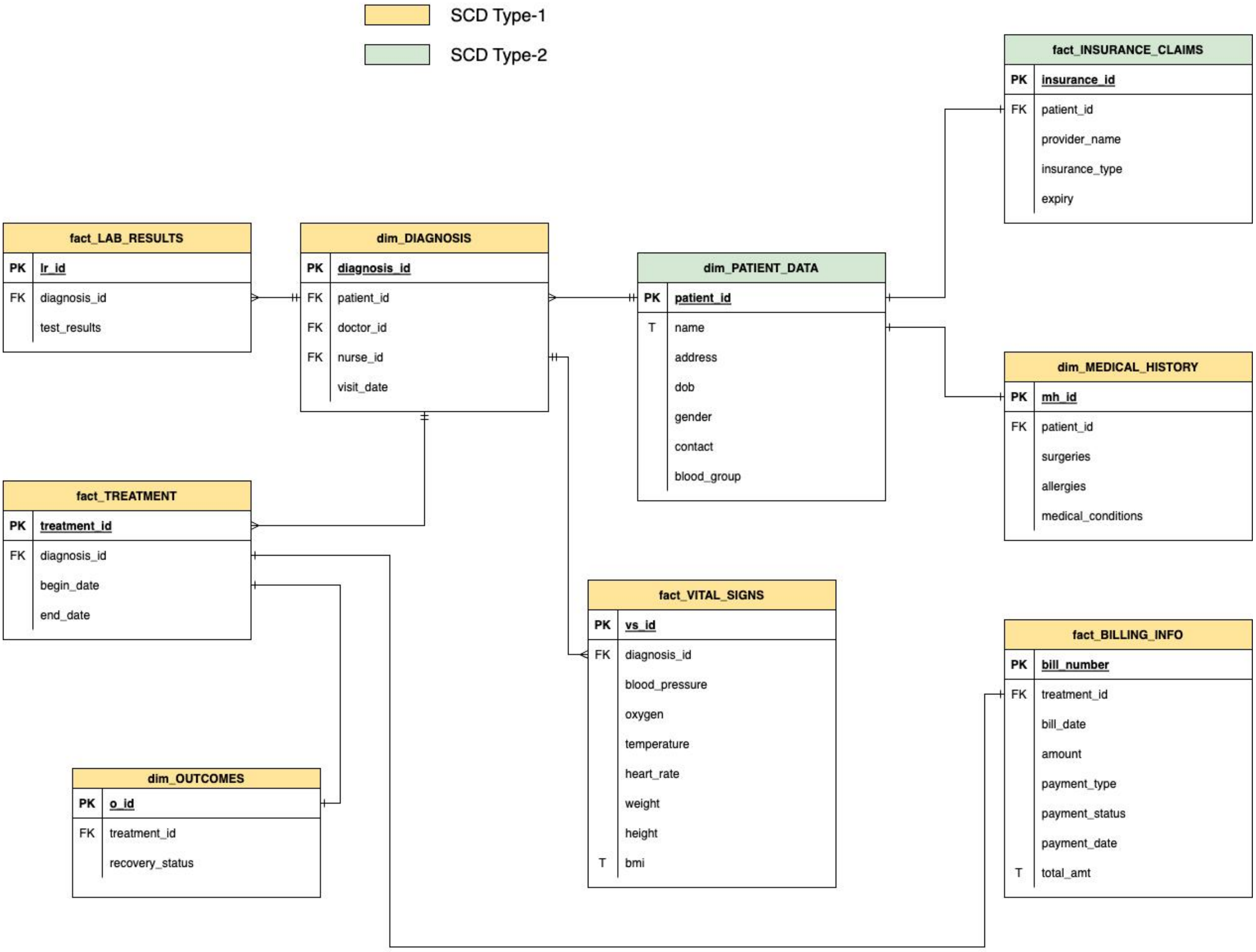
g) Maintenance Overhead: The increased number of tables and joins in a snowflake schema can make it more challenging to maintain and update the database.

4. SCDs (Slowly Changing Dimensions):

Type-1: In a Type 1 SCD, when a change occurs in a dimension attribute, the existing record is simply updated with the new value, overwriting the old value. This means the historical values are lost, and the dimension only reflects the latest state of the attribute.

Type-2: In a Type 2 SCD, when a change occurs, a new record is added to the dimension table to represent the updated attribute value. This way, historical versions of the data are preserved, and the dimension table maintains a history of changes over time.

Type-3: In a Type 3 SCD, only a limited set of historical attribute values are stored in the dimension table. Usually, two columns are added to represent the current value and a previous value.



ELT (Extract Load Transform)

In ELT, data is first extracted from various sources, then loaded into a target data storage system (such as a data warehouse or data lake), and finally transformed within the target system itself using its native processing capabilities, such as SQL queries, stored procedures, or distributed computing frameworks. The key difference between ELT and ETL lies in the location where data transformation occurs.

Advantages of ELT:

1. Scalability: ELT leverages the power and scalability of the target system, allowing it to process large volumes of data efficiently.
2. Flexibility: With ELT, data can be transformed and analyzed in various ways based on the specific needs of different users or applications. Users have access to raw data and can perform transformations as per their requirements.
3. Cost-effectiveness: ELT reduces the need for dedicated transformation servers or systems, which can lower infrastructure and maintenance costs.
4. Real-time insights: Since data is loaded first and transformations occur within the target system, ELT can enable near real-time analytics and reporting.

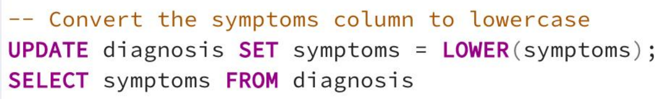
Disadvantages of ELT:

1. Data quality and governance: With ELT, data quality and governance practices must be implemented within the target system itself, as data transformation happens there. This requires careful planning and management to ensure consistent data quality and compliance.

2. Performance impact: Depending on the target system and the complexity of transformations, ELT processes may put a higher load on the system, impacting overall performance. System optimization and tuning might be necessary to mitigate any performance issues.

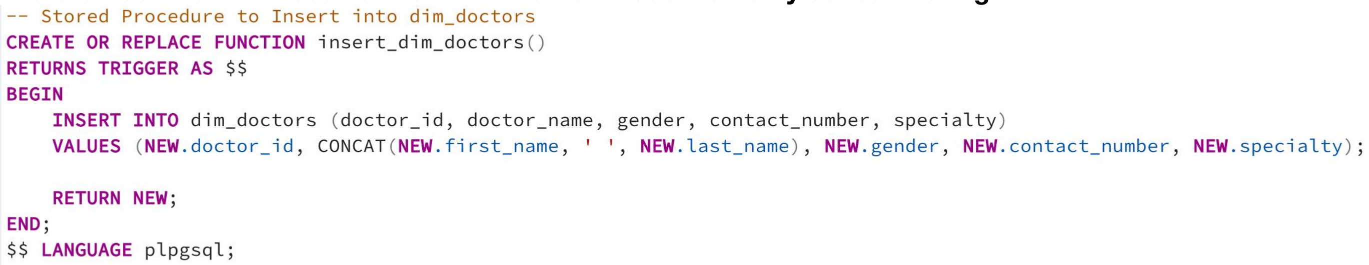
Transformations Used in Data Warehouse:

1. convert data to lowercase

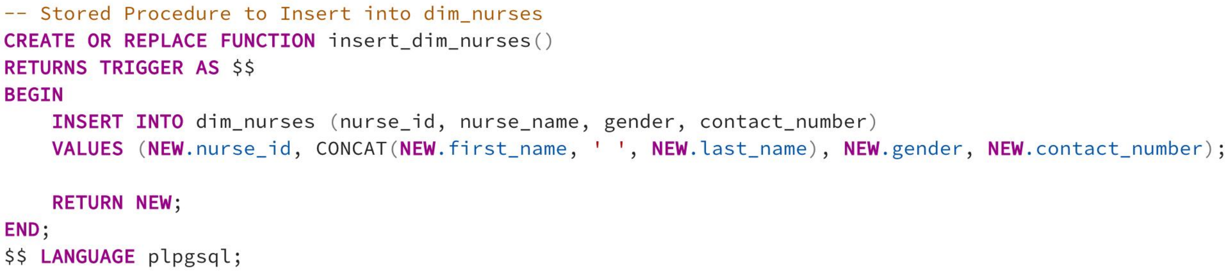


```sql
-- Convert the symptoms column to lowercase
UPDATE diagnosis SET symptoms = LOWER(symptoms);
SELECT symptoms FROM diagnosis
```

2. store doctor and nurses' full name in the warehouse table by concatenating:

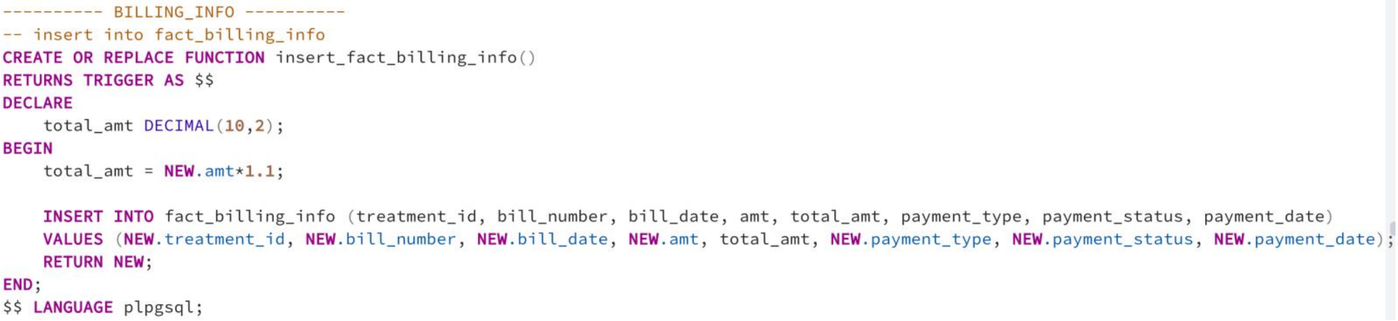


```sql
-- Stored Procedure to Insert into dim_doctors
CREATE OR REPLACE FUNCTION insert_dim_doctors()
RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO dim_doctors (doctor_id, doctor_name, gender, contact_number, specialty)
    VALUES (NEW.doctor_id, CONCAT(NEW.first_name, ' ', NEW.last_name), NEW.gender, NEW.contact_number, NEW.specialty);

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```



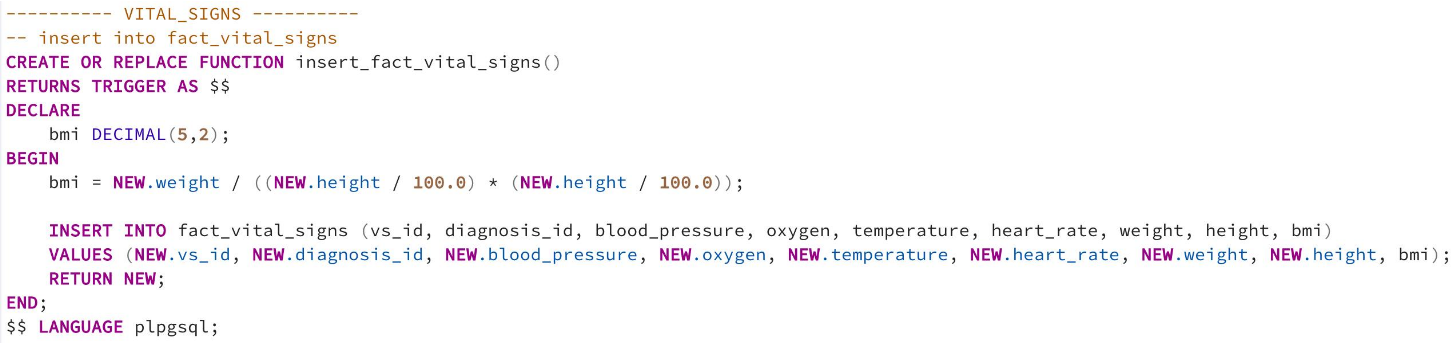
```sql
-- Stored Procedure to Insert into dim_nurses
CREATE OR REPLACE FUNCTION insert_dim_nurses()
RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO dim_nurses (nurse_id, nurse_name, gender, contact_number)
    VALUES (NEW.nurse_id, CONCAT(NEW.first_name, ' ', NEW.last_name), NEW.gender, NEW.contact_number);

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

3. calculate the total_amt (with 10% tax) on the patient's bill:



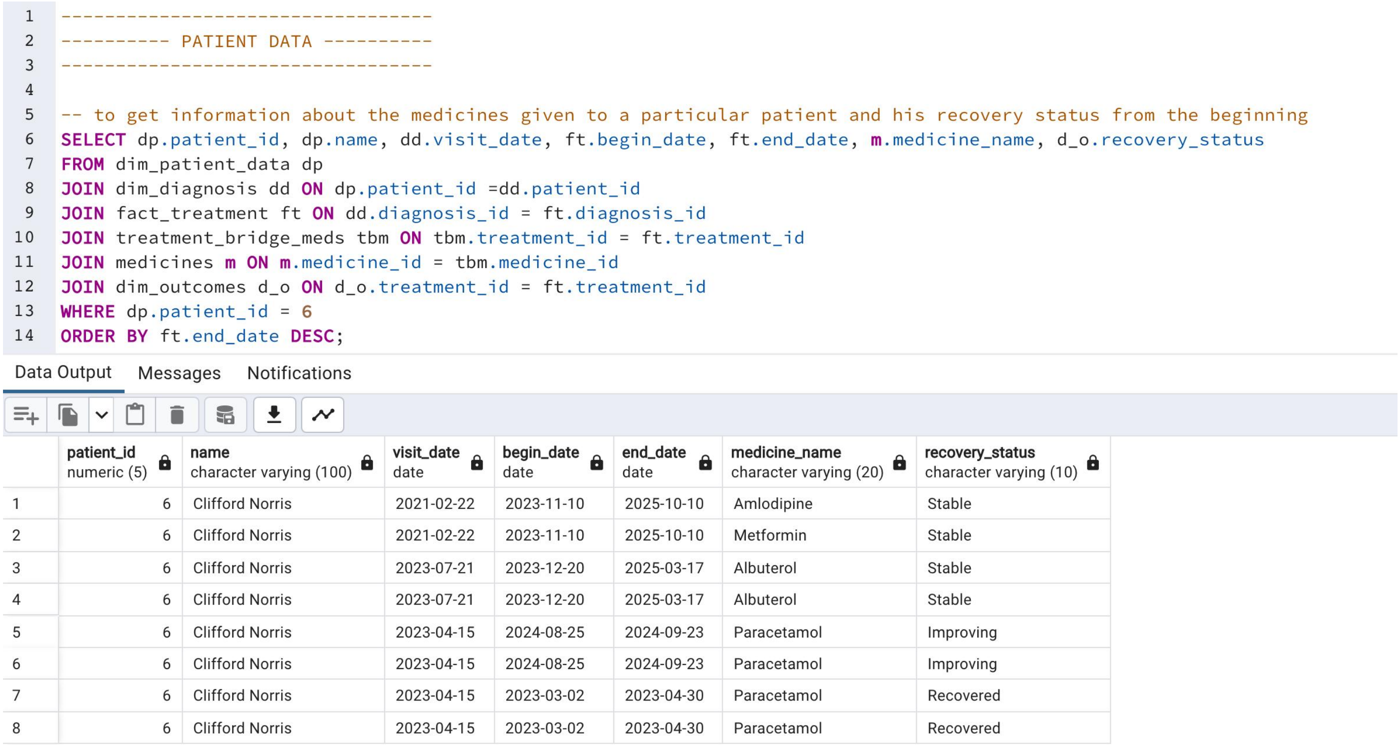
```sql
---------- BILLING_INFO ----------
-- insert into fact_billing_info
CREATE OR REPLACE FUNCTION insert_fact_billing_info()
RETURNS TRIGGER AS $$
DECLARE
    total_amt DECIMAL(10,2);
BEGIN
    total_amt = NEW.amt*1.1;

    INSERT INTO fact_billing_info (treatment_id, bill_number, bill_date, amt, total_amt, payment_type, payment_status, payment_date)
    VALUES (NEW.treatment_id, NEW.bill_number, NEW.bill_date, NEW.amt, total_amt, NEW.payment_type, NEW.payment_status, NEW.payment_date);
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

4. calculating the bmi of a patient based on the vital signs (weight and height) noted at the time of treatment:



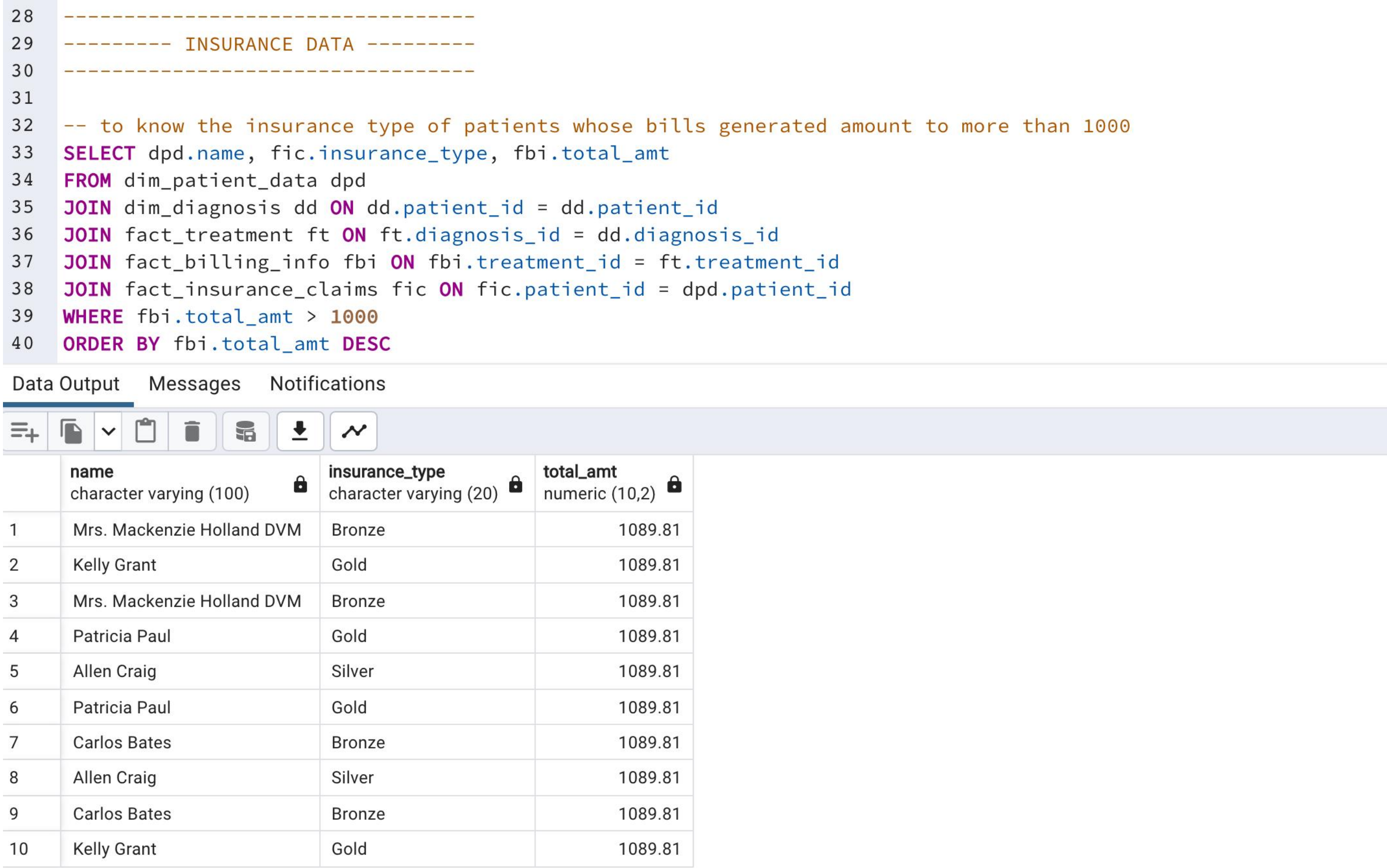
```sql
---------- VITAL_SIGNS ----------
-- insert into fact_vital_signs
CREATE OR REPLACE FUNCTION insert_fact_vital_signs()
RETURNS TRIGGER AS $$
DECLARE
    bmi DECIMAL(5,2);
BEGIN
    bmi = NEW.weight / ((NEW.height / 100.0) * (NEW.height / 100.0));

    INSERT INTO fact_vital_signs (vs_id, diagnosis_id, blood_pressure, oxygen, temperature, heart_rate, weight, height, bmi)
    VALUES (NEW.vs_id, NEW.diagnosis_id, NEW.blood_pressure, NEW.oxygen, NEW.temperature, NEW.heart_rate, NEW.weight, NEW.height, bmi);
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

QUERIES



```sql
-----------------------------------
---------- PATIENT DATA ----------
-----------------------------------

-- to get information about the medicines given to a particular patient and his recovery status from the beginning
SELECT dp.patient_id, dp.name, dd.visit_date, ft.begin_date, ft.end_date, m.medicine_name, d_o.recovery_status
FROM dim_patient_data dp
JOIN dim_diagnosis dd ON dp.patient_id =dd.patient_id
JOIN fact_treatment ft ON dd.diagnosis_id = ft.diagnosis_id
JOIN treatment_bridge_meds tbm ON tbm.treatment_id = ft.treatment_id
JOIN medicines m ON m.medicine_id = tbm.medicine_id
JOIN dim_outcomes d_o ON d_o.treatment_id = ft.treatment_id
WHERE dp.patient_id = 6
ORDER BY ft.end_date DESC;
```

	patient_id numeric (5)	name character varying (100)	visit_date date	begin_date date	end_date date	medicine_name character varying (20)	recovery_status character varying (10)
1	6	Clifford Norris	2021-02-22	2023-11-10	2025-10-10	Amlodipine	Stable
2	6	Clifford Norris	2021-02-22	2023-11-10	2025-10-10	Metformin	Stable
3	6	Clifford Norris	2023-07-21	2023-12-20	2025-03-17	Albuterol	Stable
4	6	Clifford Norris	2023-07-21	2023-12-20	2025-03-17	Albuterol	Stable
5	6	Clifford Norris	2023-04-15	2024-08-25	2024-09-23	Paracetamol	Improving
6	6	Clifford Norris	2023-04-15	2024-08-25	2024-09-23	Paracetamol	Improving
7	6	Clifford Norris	2023-04-15	2023-03-02	2023-04-30	Paracetamol	Recovered
8	6	Clifford Norris	2023-04-15	2023-03-02	2023-04-30	Paracetamol	Recovered



```sql
----------------------------------
--------- INSURANCE DATA ---------
----------------------------------

-- to know the insurance type of patients whose bills generated amount to more than 1000
SELECT dpd.name, fic.insurance_type, fbi.total_amt
FROM dim_patient_data dpd
JOIN dim_diagnosis dd ON dd.patient_id = dd.patient_id
JOIN fact_treatment ft ON ft.diagnosis_id = dd.diagnosis_id
JOIN fact_billing_info fbi ON fbi.treatment_id = ft.treatment_id
JOIN fact_insurance_claims fic ON fic.patient_id = dpd.patient_id
WHERE fbi.total_amt > 1000
ORDER BY fbi.total_amt DESC
```

	name character varying (100)	insurance_type character varying (20)	total_amt numeric (10,2)
1	Mrs. Mackenzie Holland DVM	Bronze	1089.81
2	Kelly Grant	Gold	1089.81
3	Mrs. Mackenzie Holland DVM	Bronze	1089.81
4	Patricia Paul	Gold	1089.81
5	Allen Craig	Silver	1089.81
6	Patricia Paul	Gold	1089.81
7	Carlos Bates	Bronze	1089.81
8	Allen Craig	Silver	1089.81
9	Carlos Bates	Bronze	1089.81
10	Kelly Grant	Gold	1089.81

CONCLUSION

In conclusion, the implementation of a tailored data warehousing solution for Hospital Administration is essential for achieving data-driven decision-making, streamlined processes, and enhanced efficiency.

The solution provides a comprehensive patient database, enabling valuable insights into patient demographics and medical history. It also facilitates financial efficiency by identifying improvement areas and streamlining processes for financial stability. Moreover, by analyzing operational metrics such as patient flow and hospital provider performance, the administration can make informed decisions on staffing, equipment utilization, and facility planning.

Overall, this tailored data warehousing solution empowers the Hospital Administration to leverage data strategically, resulting in improved patient care, optimized financial operations, and enhanced resource allocation and planning. Embracing this solution drives positive changes, optimizes processes, and ensures the delivery of high-quality hospital services while achieving operational excellence.