جامعة أم القرى

كلية الحاسبات

Data Management and Warehousing Project

Project Team:

names	Student number	Work percentage
Mohammed Khalid Altufayhi	44400XXXX	20%
Anas Mohammed Alsubhi	44400XXXX	20%
Faisal Hammad Alomari	44400XXXX	20%
Nawaf Abdulrahman Alageel	4441XXXX	20%
Albadar Ibrahim Almaymani	4440XXXX	20%

Introduction

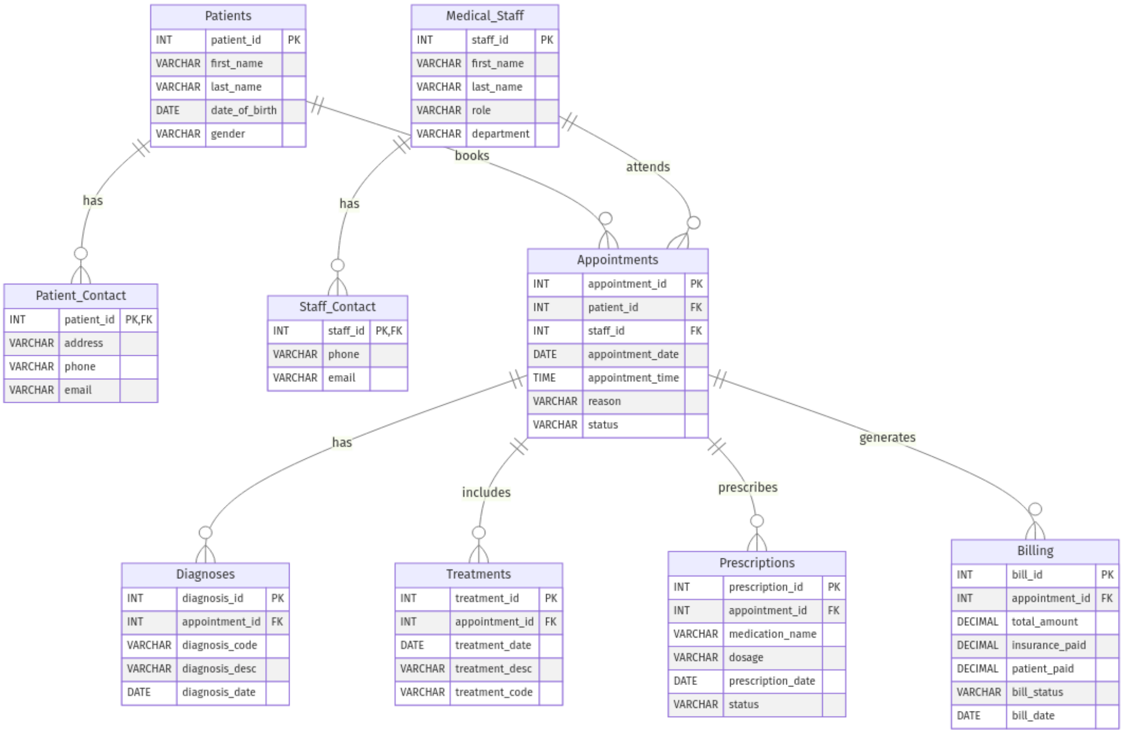
The healthcare domain often deals with massive datasets that contain sensitive and critical information, including patient records, diagnoses, treatments, prescriptions, and billing information. A well-designed data warehouse schema can help efficiently manage and analyze such data for better decision-making, operational insights, and compliance with healthcare standards. This report explores three different warehouse schemas designed for the given healthcare transactional database: the **Snowflake Schema**, the **Galaxy Schema**, and the **Star Schema**. Each schema is designed with a unique structure to meet various analytical needs, balancing performance, storage efficiency, and ease of querying.

Schema Designs

1. Snowflake Schema

The **Snowflake Schema** is a normalized version that reduces redundancy by dividing the tables into multiple dimensions. This schema separates contact information for both patients and medical staff into dedicated tables to achieve a higher level of normalization.

- **Design**:
    - The Patients table is divided into Patients and Patient_Contact, where Patient_Contact stores contact information.
    - Similarly, the Medical_Staff table is split into Medical_Staff and Staff_Contact.
    - Other tables such as Appointments, Diagnoses, Treatments, Prescriptions, and Billing remain as individual dimension tables linked to Patients and Medical_Staff.



Patients		
INT	patient_id	PK
VARCHAR	first_name	
VARCHAR	last_name	
DATE	date_of_birth	
VARCHAR	gender	

Medical_Staff		
INT	staff_id	PK
VARCHAR	first_name	
VARCHAR	last_name	
VARCHAR	role	
VARCHAR	department	

Patient_Contact		
INT	patient_id	PK,FK
VARCHAR	address	
VARCHAR	phone	
VARCHAR	email	

Staff_Contact		
INT	staff_id	PK,FK
VARCHAR	phone	
VARCHAR	email	

Appointments		
INT	appointment_id	PK
INT	patient_id	FK
INT	staff_id	FK
DATE	appointment_date	
TIME	appointment_time	
VARCHAR	reason	
VARCHAR	status	

Diagnoses		
INT	diagnosis_id	PK
INT	appointment_id	FK
VARCHAR	diagnosis_code	
VARCHAR	diagnosis_desc	
DATE	diagnosis_date	

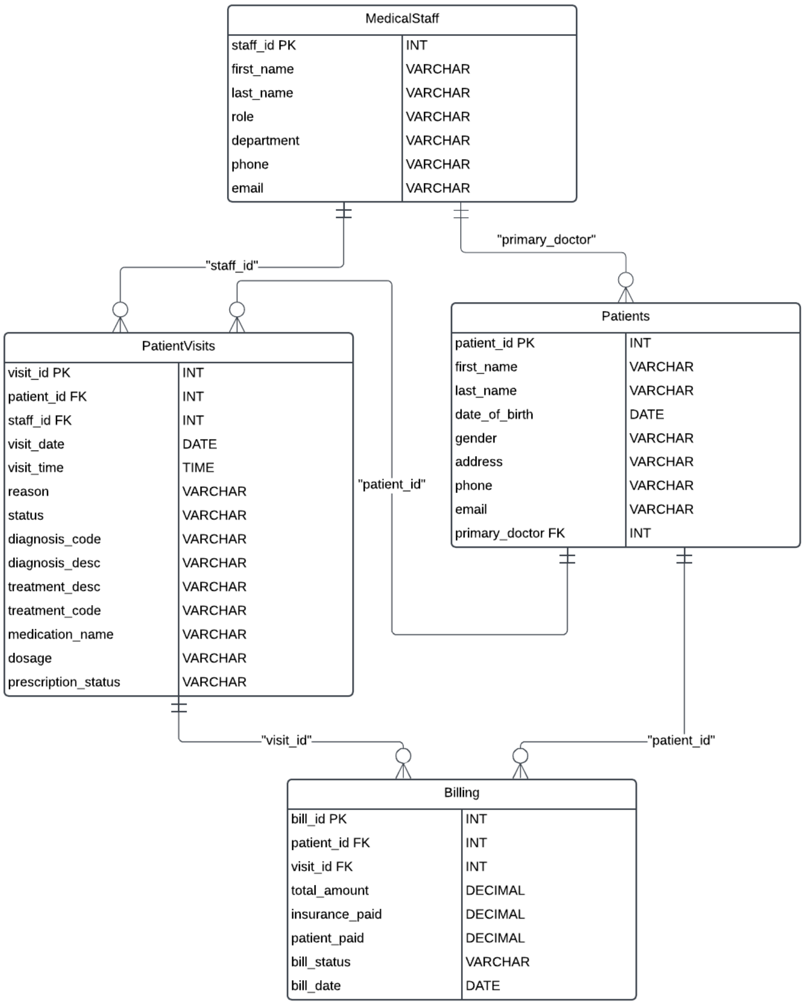
Treatments		
INT	treatment_id	PK
INT	appointment_id	FK
DATE	treatment_date	
VARCHAR	treatment_desc	
VARCHAR	treatment_code	

Prescriptions		
INT	prescription_id	PK
INT	appointment_id	FK
VARCHAR	medication_name	
VARCHAR	dosage	
DATE	prescription_date	
VARCHAR	status	

Billing		
INT	bill_id	PK
INT	appointment_id	FK
DECIMAL	total_amount	
DECIMAL	insurance_paid	
DECIMAL	patient_paid	
VARCHAR	bill_status	
DATE	bill_date	

2. Galaxy Schema

- The Galaxy Schema is a combination of the star and snowflake schemas, where key tables are slightly denormalized, reducing the number of tables without introducing significant redundancy.
- **Design**:
  - Patients: Contains patient info and contact details, with primary_doctor linking to MedicalStaff.
  - PatientVisits serves as a consolidated table for Appointments, Diagnoses, Treatments, and Prescriptions, linking each patient visit to the appropriate staff.
  - Billing tracks financial details for each visit, connected to PatientVisits and Patients.



MedicalStaff	
staff_id PK	INT
first_name	VARCHAR
last_name	VARCHAR
role	VARCHAR
department	VARCHAR
phone	VARCHAR
email	VARCHAR

PatientVisits	
visit_id PK	INT
patient_id FK	INT
staff_id FK	INT
visit_date	DATE
visit_time	TIME
reason	VARCHAR
status	VARCHAR
diagnosis_code	VARCHAR
diagnosis_desc	VARCHAR
treatment_desc	VARCHAR
treatment_code	VARCHAR
medication_name	VARCHAR
dosage	VARCHAR
prescription_status	VARCHAR

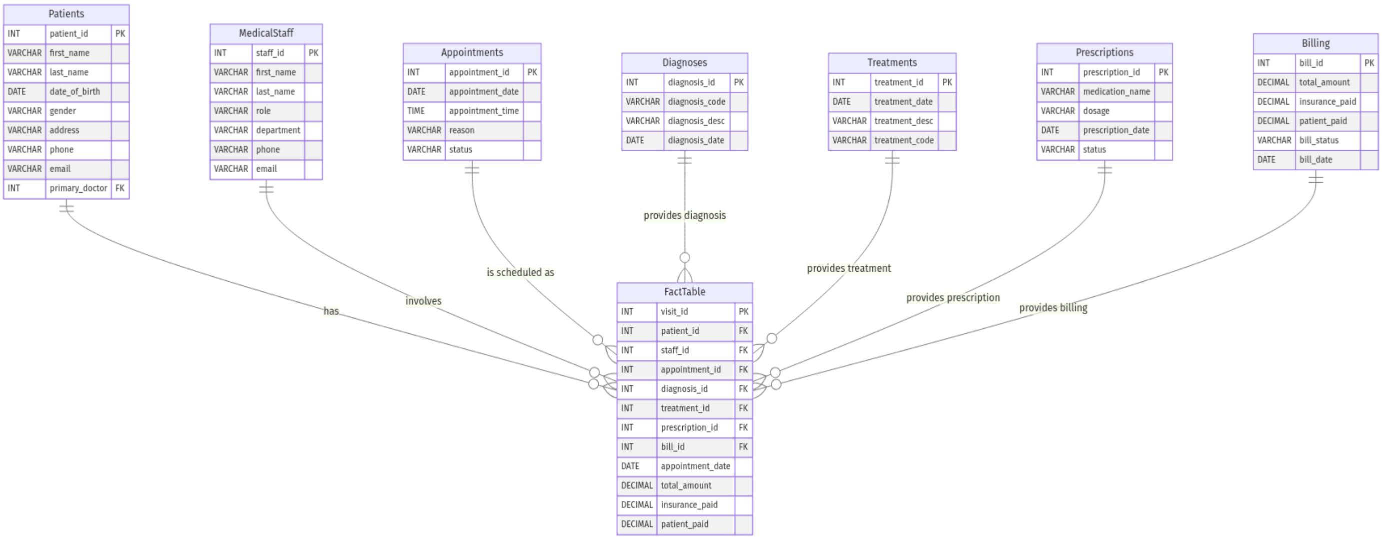
Patients	
patient_id PK	INT
first_name	VARCHAR
last_name	VARCHAR
date_of_birth	DATE
gender	VARCHAR
address	VARCHAR
phone	VARCHAR
email	VARCHAR
primary_doctor FK	INT

Billing	
bill_id PK	INT
patient_id FK	INT
visit_id FK	INT
total_amount	DECIMAL
insurance_paid	DECIMAL
patient_paid	DECIMAL
bill_status	VARCHAR
bill_date	DATE

3. Star Schema

The **Star Schema** is a denormalized structure with a central fact table, FactTable, that links to all dimension tables, creating a "star" shape. This schema is optimized for fast querying and is suited for large-scale analytical processing.

- **Design**:
  - A central FactTable holds references to dimension tables: PATIENTS, MEDICALSTAFF, APPOINTMENTS, DIAGNOSES, TREATMENTS, PRESCRIPTIONS, and BILLING.
  - This schema is designed to allow rapid aggregation of data across various dimensions with minimal joins, making it ideal for data warehousing and reporting in high-performance scenarios.



Patients

INT	patient_id	PK
VARCHAR	first_name	
VARCHAR	last_name	
DATE	date_of_birth	
VARCHAR	gender	
VARCHAR	address	
VARCHAR	phone	
VARCHAR	email	
INT	primary_doctor	FK

MedicalStaff

INT	staff_id	PK
VARCHAR	first_name	
VARCHAR	last_name	
VARCHAR	role	
VARCHAR	department	
VARCHAR	phone	
VARCHAR	email	

Appointments

INT	appointment_id	PK
DATE	appointment_date	
TIME	appointment_time	
VARCHAR	reason	
VARCHAR	status	

Diagnoses

INT	diagnosis_id	PK
VARCHAR	diagnosis_code	
VARCHAR	diagnosis_desc	
DATE	diagnosis_date	

Treatments

INT	treatment_id	PK
DATE	treatment_date	
VARCHAR	treatment_desc	
VARCHAR	treatment_code	

Prescriptions

INT	prescription_id	PK
VARCHAR	medication_name	
VARCHAR	dosage	
DATE	prescription_date	
VARCHAR	status	

Billing

INT	bill_id	PK
DECIMAL	total_amount	
DECIMAL	insurance_paid	
DECIMAL	patient_paid	
VARCHAR	bill_status	
DATE	bill_date	

FactTable

INT	visit_id	PK
INT	patient_id	FK
INT	staff_id	FK
INT	appointment_id	FK
INT	diagnosis_id	FK
INT	treatment_id	FK
INT	prescription_id	FK
INT	bill_id	FK
DATE	appointment_date	
DECIMAL	total_amount	
DECIMAL	insurance_paid	
DECIMAL	patient_paid	

Pros and Cons of Each Schema

Snowflake Schema

- **Pros**:
  - Reduces data redundancy, leading to potentially smaller storage requirements.
  - Normalized structure makes updates easier and minimizes duplication.
- **Cons**:
  - Complex queries due to multiple joins across tables, which may slow down performance.
  - Less suitable for large-scale data analysis as normalization requires more relational navigation.

Galaxy Schema

- **Pros**:
  - Flexibility: Allows for analysis of multiple processes or events with shared data.
  - Provides a balanced approach to data redundancy and storage efficiency.
- **Cons**:
  - Increased Redundancy,the consolidated **PatientVisits** table may have some redundancy, as each visit stores multiple details.
  - Combining tables may limit flexibility if future use cases require further breakdown or normalization of data.

Star Schema

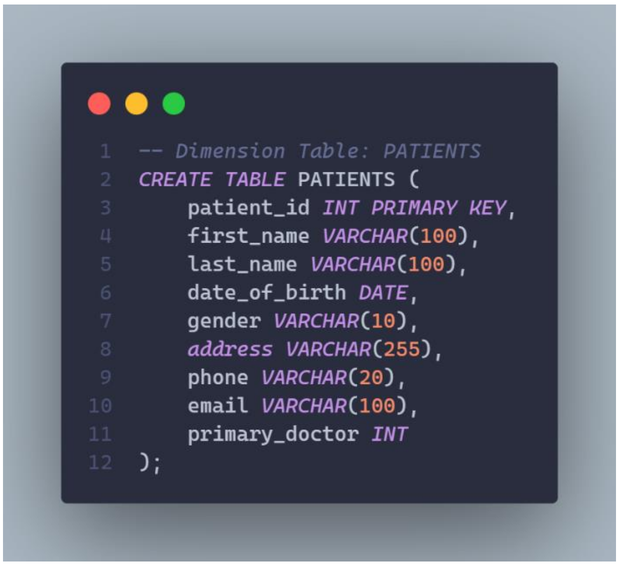
- **Pros**:
  - Fast query performance due to minimal joins with a central fact table.
  - Ideal for analytical and reporting purposes with high volumes of data.
- **Cons**:
  - Increased data redundancy, leading to larger storage requirements.
  - Updating data is more complex due to the denormalized structure.

Implementing Schema

We chose the **Star Schema** because it is the best fit for large-scale analytical queries in the healthcare domain. The Star Schema's denormalized structure (less splitting of tables) allows for faster query performance, which is essential for analyzing healthcare data quickly and efficiently. In this setup, all essential details are linked to a single **FactTable**, simplifying analysis across different tables.

```sql
CREATE DATABASE HealthCare;
USE HealthCare;
```

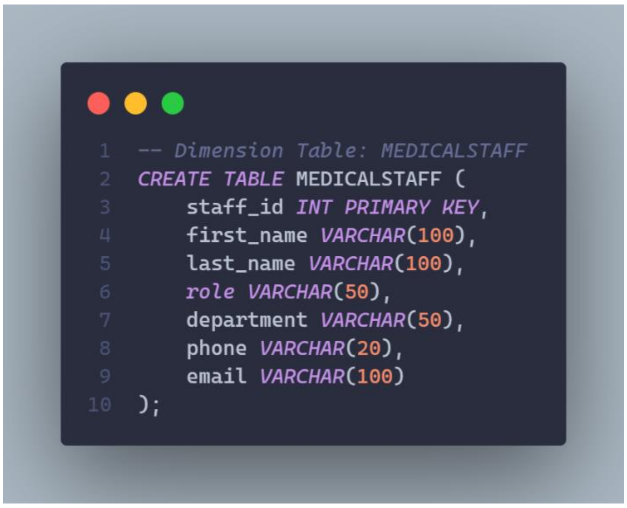
The database is created and selected as HealthCare to organize all the healthcare-related tables.



```sql
-- Dimension Table: PATIENTS
CREATE TABLE PATIENTS (
    patient_id INT PRIMARY KEY,
    first_name VARCHAR(100),
    last_name VARCHAR(100),
    date_of_birth DATE,
    gender VARCHAR(10),
    address VARCHAR(255),
    phone VARCHAR(20),
    email VARCHAR(100),
    primary_doctor INT
);
```

- **Description**: This table stores basic information about each patient, such as their name, birth date, contact details, and gender. It also includes a primary_doctor field, linking each patient to a primary care doctor.

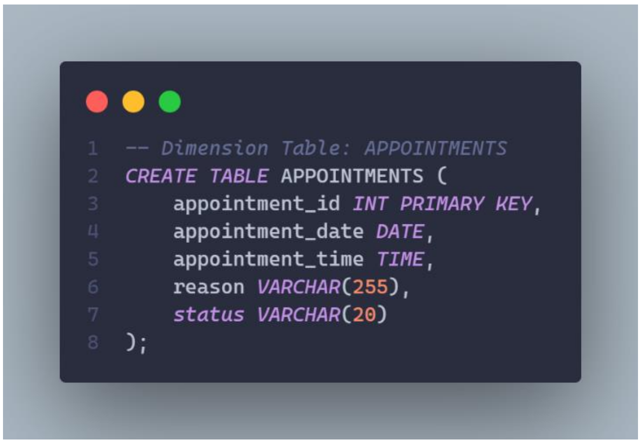
- **Purpose**: Serves as a dimension to provide patient-specific information when analyzing healthcare events.



```sql
-- Dimension Table: MEDICALSTAFF
CREATE TABLE MEDICALSTAFF (
    staff_id INT PRIMARY KEY,
    first_name VARCHAR(100),
    last_name VARCHAR(100),
    role VARCHAR(50),
    department VARCHAR(50),
    phone VARCHAR(20),
    email VARCHAR(100)
);
```

- **Description**: Contains data about the medical staff, including their role (doctor, nurse), department, and contact details.

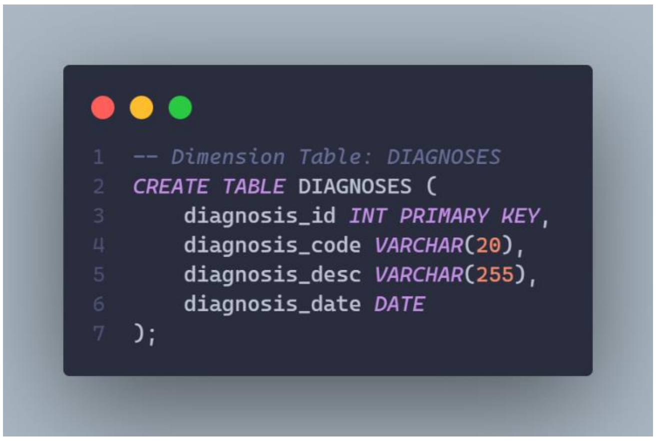
- **Purpose**: Acts as a dimension table for information about healthcare providers, helping to track which staff member interacted with which patient.



```sql
-- Dimension Table: APPOINTMENTS
CREATE TABLE APPOINTMENTS (
    appointment_id INT PRIMARY KEY,
    appointment_date DATE,
    appointment_time TIME,
    reason VARCHAR(255),
    status VARCHAR(20)
);
```

- **Description**: Holds information on appointments, including the date, time, reason, and status (completed, canceled).

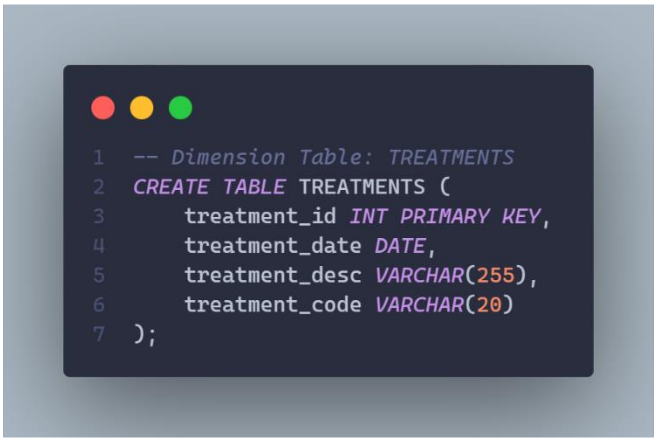
- **Purpose**: This dimension allows for analyzing patient visit patterns, including the timing and purpose of appointments.



```sql
-- Dimension Table: DIAGNOSES
CREATE TABLE DIAGNOSES (
    diagnosis_id INT PRIMARY KEY,
    diagnosis_code VARCHAR(20),
    diagnosis_desc VARCHAR(255),
    diagnosis_date DATE
);
```

- **Description**: Stores diagnosis information, such as a code (ICD code), description, and the date the diagnosis was made.

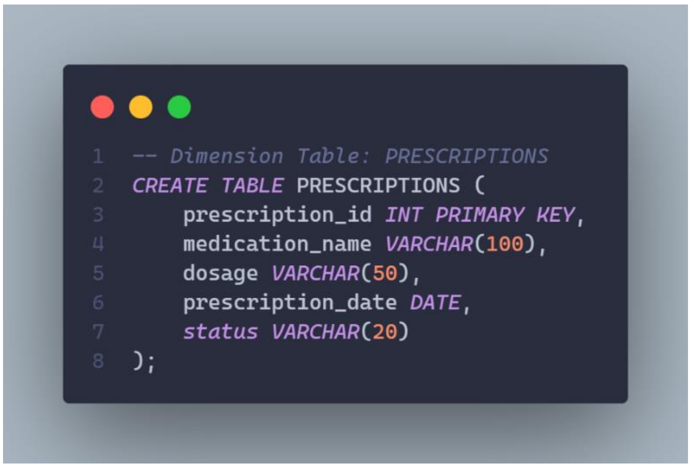
- **Purpose**: Helps in tracking the conditions diagnosed for each patient, which can be useful for both reporting and longitudinal studies.



```sql
-- Dimension Table: TREATMENTS
CREATE TABLE TREATMENTS (
    treatment_id INT PRIMARY KEY,
    treatment_date DATE,
    treatment_desc VARCHAR(255),
    treatment_code VARCHAR(20)
);
```

- **Description**: Contains information about treatments provided to patients, including a description, date, and a procedure code.

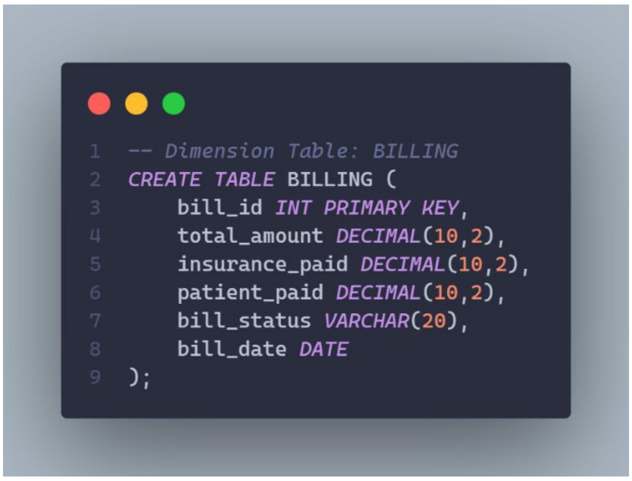
- **Purpose**: This table enables the analysis of treatments given, allowing insights into healthcare practices and patient outcomes.



```sql
-- Dimension Table: PRESCRIPTIONS
CREATE TABLE PRESCRIPTIONS (
    prescription_id INT PRIMARY KEY,
    medication_name VARCHAR(100),
    dosage VARCHAR(50),
    prescription_date DATE,
    status VARCHAR(20)
);
```

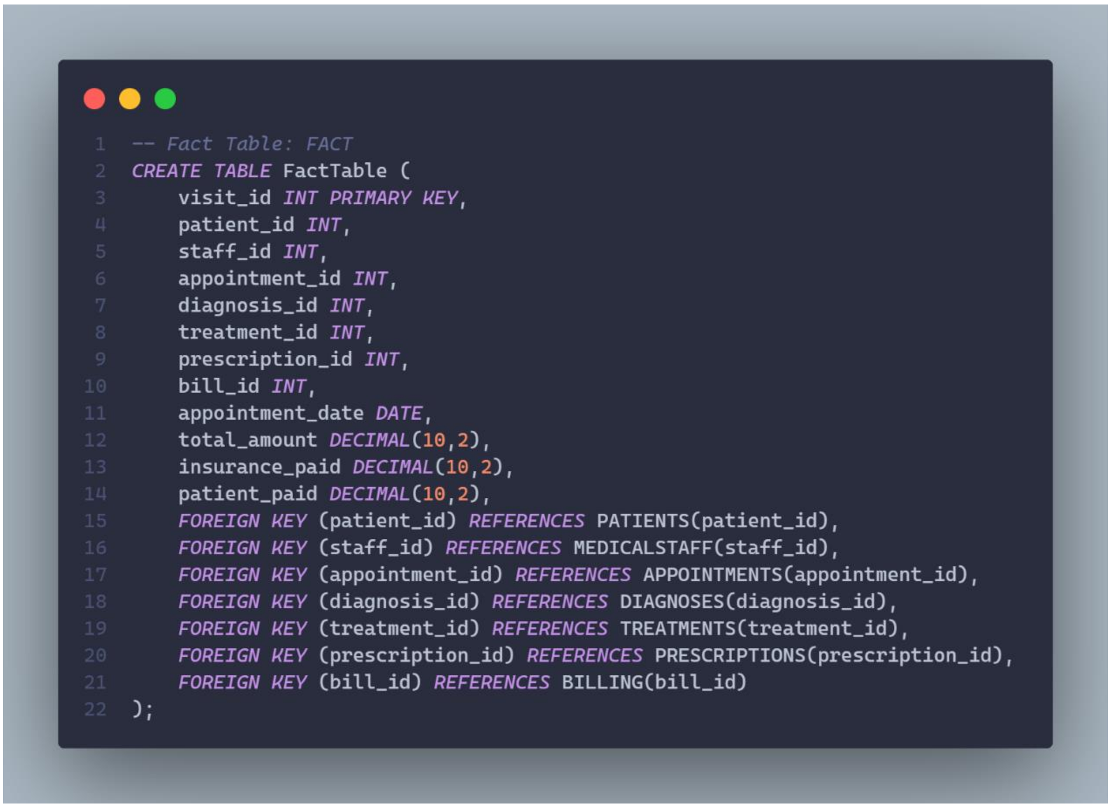
- **Description**: Tracks prescriptions issued to patients, with details about the medication, dosage, date, and current status (active, expired).

- **Purpose**: This table allows analysis of medication usage and patterns, useful for tracking prescribed medications over time.



```sql
-- Dimension Table: BILLING
CREATE TABLE BILLING (
    bill_id INT PRIMARY KEY,
    total_amount DECIMAL(10,2),
    insurance_paid DECIMAL(10,2),
    patient_paid DECIMAL(10,2),
    bill_status VARCHAR(20),
    bill_date DATE
);
```

- **Description**: Stores billing and payment details for healthcare services, including the total amount, amount covered by insurance, and any remaining balance.
- **Purpose**: Facilitates financial reporting and analysis of patient payments, insurance contributions, and outstanding balances.



```sql
-- Fact Table: FACT
CREATE TABLE FactTable (
    visit_id INT PRIMARY KEY,
    patient_id INT,
    staff_id INT,
    appointment_id INT,
    diagnosis_id INT,
    treatment_id INT,
    prescription_id INT,
    bill_id INT,
    appointment_date DATE,
    total_amount DECIMAL(10,2),
    insurance_paid DECIMAL(10,2),
    patient_paid DECIMAL(10,2),
    FOREIGN KEY (patient_id) REFERENCES PATIENTS(patient_id),
    FOREIGN KEY (staff_id) REFERENCES MEDICALSTAFF(staff_id),
    FOREIGN KEY (appointment_id) REFERENCES APPOINTMENTS(appointment_id),
    FOREIGN KEY (diagnosis_id) REFERENCES DIAGNOSES(diagnosis_id),
    FOREIGN KEY (treatment_id) REFERENCES TREATMENTS(treatment_id),
    FOREIGN KEY (prescription_id) REFERENCES PRESCRIPTIONS(prescription_id),
    FOREIGN KEY (bill_id) REFERENCES BILLING(bill_id)
);
```

- **Description**: The central FactTable links all dimension tables (Patients, Medical Staff, Appointments, Diagnoses, Treatments, Prescriptions, and Billing). It stores the primary key (visit_id) for each visit or interaction, linking to each relevant dimension.

- **Purpose**: This table is essential for fast data retrieval and analysis across multiple dimensions. By centralizing the links to all patient interactions (appointments, diagnoses, treatments, prescriptions, and billing), this table supports quick aggregation and reporting, making it ideal for analytical queries.

Conclusion

In this **Star Schema** design, each dimension table represents an entity in the healthcare domain, while the **FactTable** serves as the central table linking all patient interactions. This structure allows efficient querying across multiple dimensions, such as analyzing patient care, tracking medical staff interactions, and monitoring billing details. This schema is well-suited for analytical purposes, providing a balanced approach to data organization and query performance.