

GROUP 3

Project Deliverable_5

CS425-02: Database Organization

HEALTHCARE MANAGEMENT SYSTEM

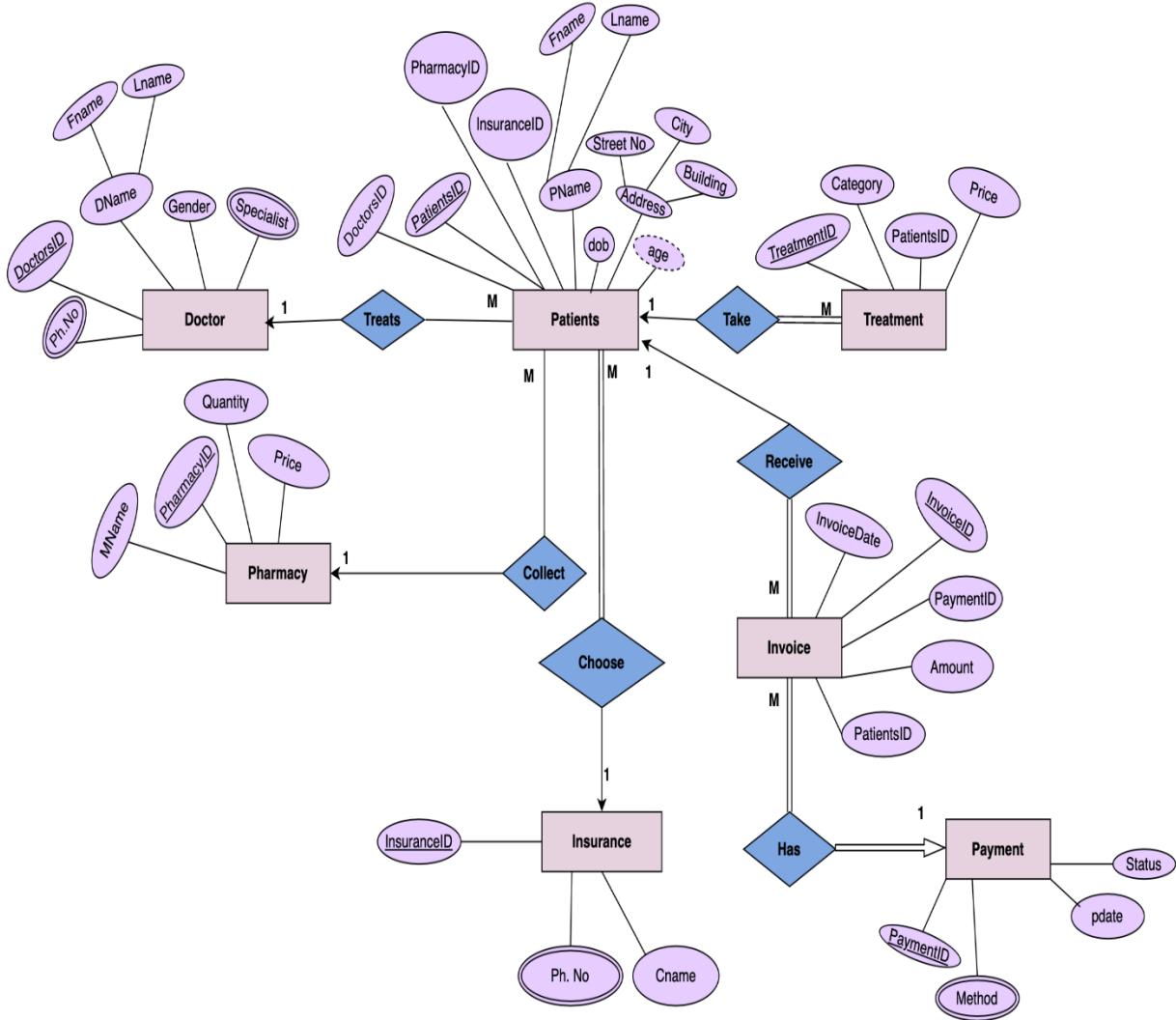
Group Members Details

Member name	CWID	Hawk Id
Akshitha Bedre Shivakumar	A20544641	abedreshivakumar@hawk.iit.edu
Jen Kai Wang	A20576222	jwang300@hawk.iit.edu
Ramya Venkatesh	A20546964	rvenkatesh1@hawk.iit.edu
Waseem Pasha Mohammad	A20551352	wmohammad@hawk.iit.edu

INTRODUCTION:

The effective administration of data is essential to the healthcare sector. A strong relational database that can store, organize, and retrieve vital patient data, medical records, and administrative data is the foundation of a strong Healthcare Management System (HMS). Consider the patient, physician, and billing information as discrete parts inside the library, each containing specific details such as names, dates, diagnoses, and expenses. These characteristics store certain bits of information, and this is where relationships come into play. Consider foreign keys as links between physicians and patients, diagnoses and patients, bills, and doctors. You can quickly browse the data on this linked web to learn more about a particular patient's medical. But creating this information center needs careful preparation. The initial step is to identify the key players, which are the patients, doctors, and treatment. The next step is to specify the information you wish to record for each, such as a patient's allergies or a doctor's specializations. After that, it's time to create those vital connections that guarantee consistent data and easy retrieval. Robust security measures are vital to safeguard it, adhering to strict regulations. Through comprehension of these fundamental ideas and customization to your requirements, you may construct a relational database that enables your HMS to efficiently handle medical data.

ENTITY RELATIONSHIP DIAGRAM:



ABBREVIATIONS OF THE ABOVE ENTITY RELATIONSHIP DIAGRAM:

DOCTOR:

DoctorsID: Doctor's ID
Dname: Doctor Name
Fname: First name
Lname: Last name
Gender: Doctor Gender
Specialist: Doctor Specialization
Ph.no: Doctor Phone Number

PATIENT:

PatientsID: Patient's ID
DoctorsID: Doctor's ID
PharmacyID: Pharmacy ID
InsuranceID: Insurance ID
Pname: Patient's Name
Fname: Full name
Lname: Last name
DOB: Patient's Date of Birth
Age: Patient's age
Address: Patient's Address
Street No: Patient's Street Number
City: Patient's City
Building: Patient's Building Number

TREATMENT:

TreatmentID: Treatment ID
PatientsID: Patient's ID
Category: Treatment Category
Price: Price of the Treatment

PHARMACY:

PharmacyID: Pharmacy ID

Mname: Medicine Name

Quantity: Medicine Quantity

Price: Medicine Price

INSURANCE:

InsuranceID: INSURANCE ID

CName: Insurance Company Name

Ph. No: Insurance Phone Number

INVOICE:

InvoiceID: Invoice ID

PaymentID: Payment ID

PatientsID: Patients ID

InvoiceDate: Date of the invoice generated.

Amount: Patient's amount for the Treatment

PAYMENT:

PaymentID: Payment ID

Status: Payment Status

Method: Payment Method

Pdate: Payment Date

SCHEMA IN ER DIAGRAM:

- Doctor (**DoctorsID**, Dname [fname, lname], Ph.No, Gender, Specialist)
- Patient (**PatientsID**, DoctorsID, PharmacyID, InsuranceID, Pname [fname, lname], dob, Age, Address [Street NO, City, Building])
- Treatment (**TreatmentID**, Price, PatientsID, Category)
- Pharmacy (**PharmacyID**, Price, Quantity, MName)
- Insurance (**InsuranceID**, Ph. No, Cname)
- Invoice (**InvoiceID**, PaymentID, PatientsID, InvoiceDate, Amount)
- Payment (**PaymentID**, Status, Method, pdate)

BUSINESS RULES:

- Doctor-Patient Relationship:
 - One Doctor can treat many Patients.
 - One Patient must have one primary care Doctor.
- Patient-pharmacy Relationship:
 - Many Patients can collect medicines from a Pharmacy.
 - One Pharmacy can serve many Patients.
- Patient-Invoice Relationship:
 - One Patient can receive many Invoices for services rendered.
 - One Invoice is associated with one Patient.

- Patient-Treatment Relationship:
 - One Patient can take many Treatments.
 - Multiple Treatments can be given to a Patient.
- Invoice-Payment Relationship:
 - One Payment has multiple Invoices.
 - Invoices can be paid by one Payment.
- Patient-Insurance Relationship:
 - Many Patients can choose one primary Insurance Provider.
 - One Insurance Provider can cover many Patients.

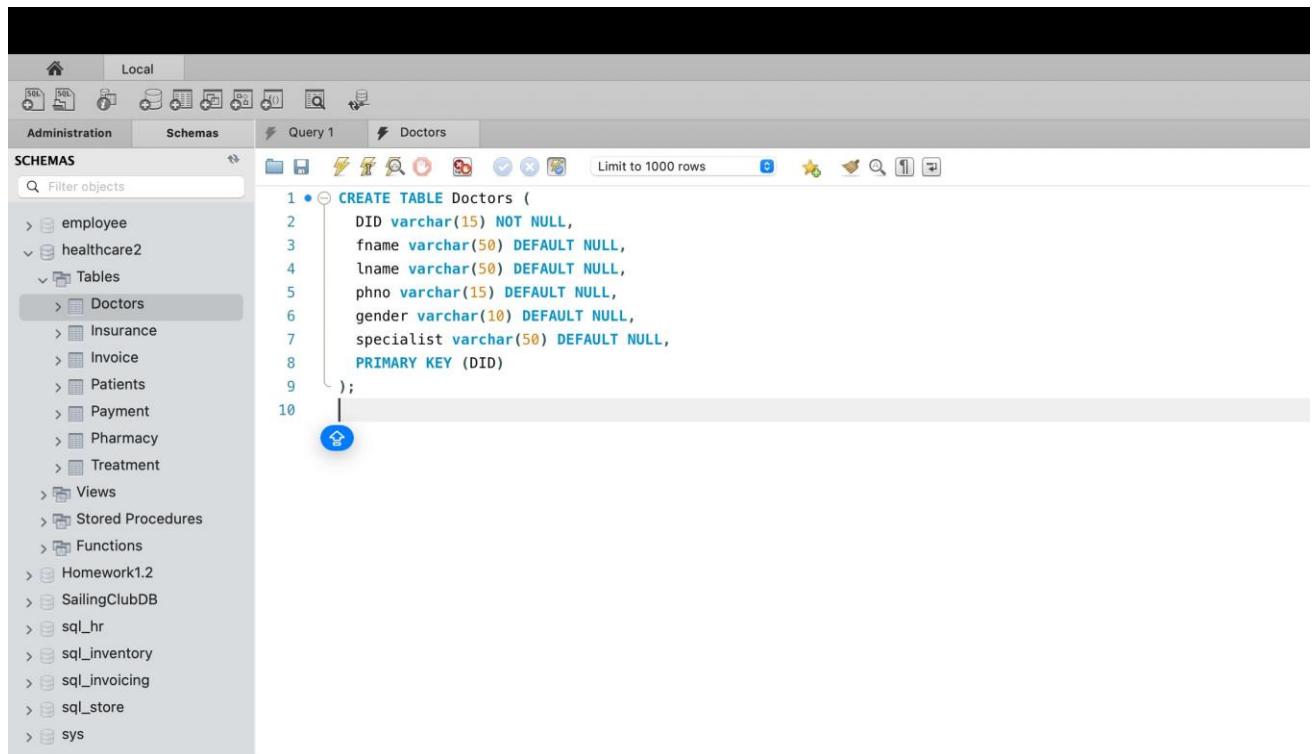
Initial Requirements:

Please make sure to fulfill the below requirements mentioned in the requirements.txt file before running the healthcare.py file.

```
(user = 'root',
host =
'localhost', port
= 3306,
password = 'pass*****')
```

TABLES CREATION:

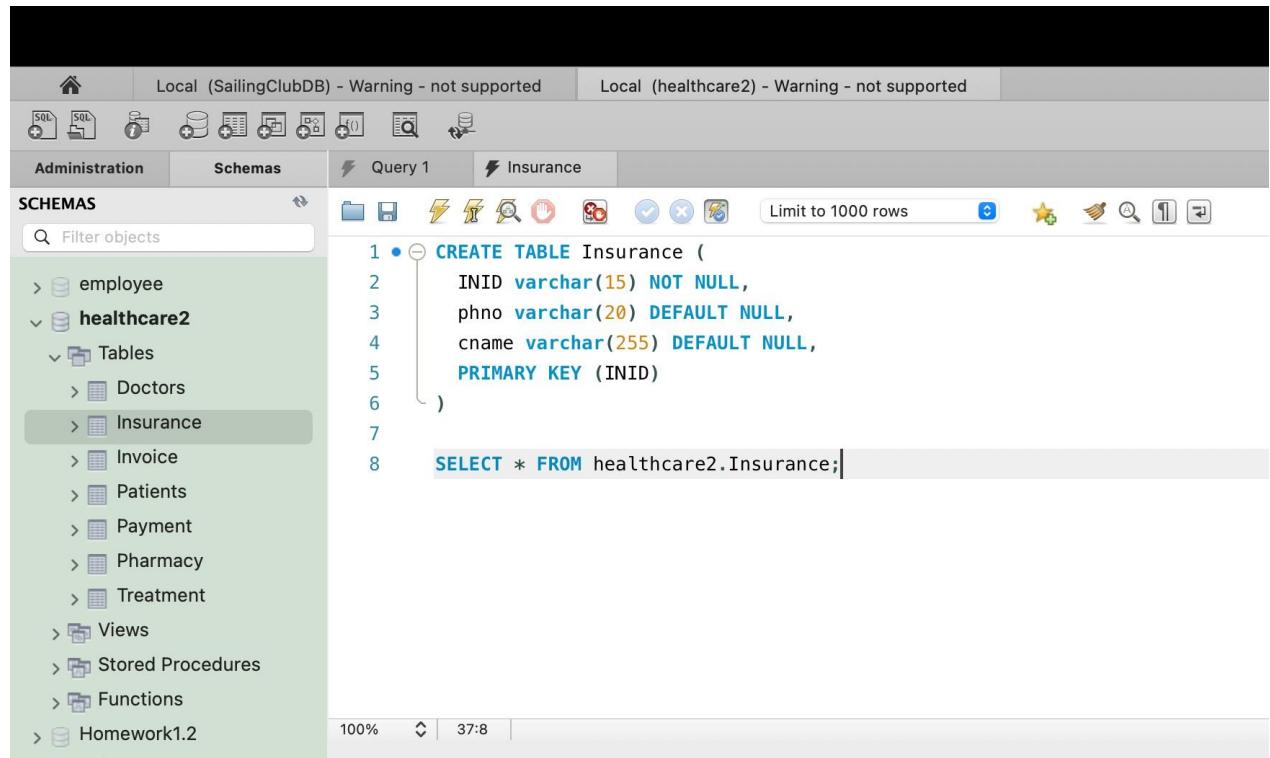
• Create Doctor Table



The screenshot shows a database management interface with a toolbar at the top and a sidebar on the left labeled "SCHEMAS". The "Tables" section is expanded, showing various tables like "Doctors", "Insurance", "Invoice", etc. The "Doctors" table is selected and highlighted with a blue border. On the right, a query editor window displays the SQL code for creating the "Doctors" table:

```
1 • ⓘ CREATE TABLE Doctors (
2     DID varchar(15) NOT NULL,
3     fname varchar(50) DEFAULT NULL,
4     lname varchar(50) DEFAULT NULL,
5     phno varchar(15) DEFAULT NULL,
6     gender varchar(10) DEFAULT NULL,
7     specialist varchar(50) DEFAULT NULL,
8     PRIMARY KEY (DID)
9 );
10
```

• Create Insurance table

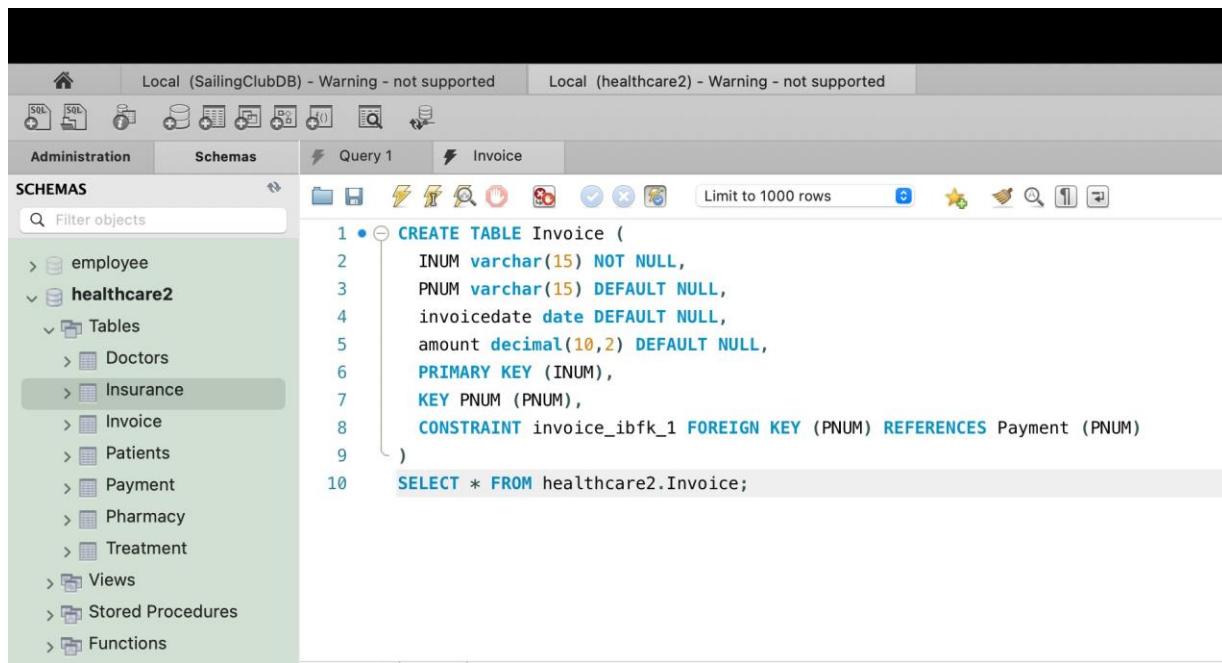


The screenshot shows the MySQL Workbench interface. The left sidebar displays the database schema with the 'healthcare2' schema selected. Inside 'healthcare2', the 'Tables' folder contains several tables: Doctors, Insurance (which is currently selected), Invoice, Patients, Payment, Pharmacy, Treatment, Views, Stored Procedures, Functions, and Homework1.2. The main pane shows a SQL query editor with the following code:

```
1 • ⊖ CREATE TABLE Insurance (
2     INID varchar(15) NOT NULL,
3     phno varchar(20) DEFAULT NULL,
4     cname varchar(255) DEFAULT NULL,
5     PRIMARY KEY (INID)
6 )
7
8     SELECT * FROM healthcare2.Insurance;
```

The code consists of two parts: a CREATE TABLE statement defining the Insurance table with columns INID, phno, and cname, and a SELECT statement to retrieve all rows from the Insurance table.

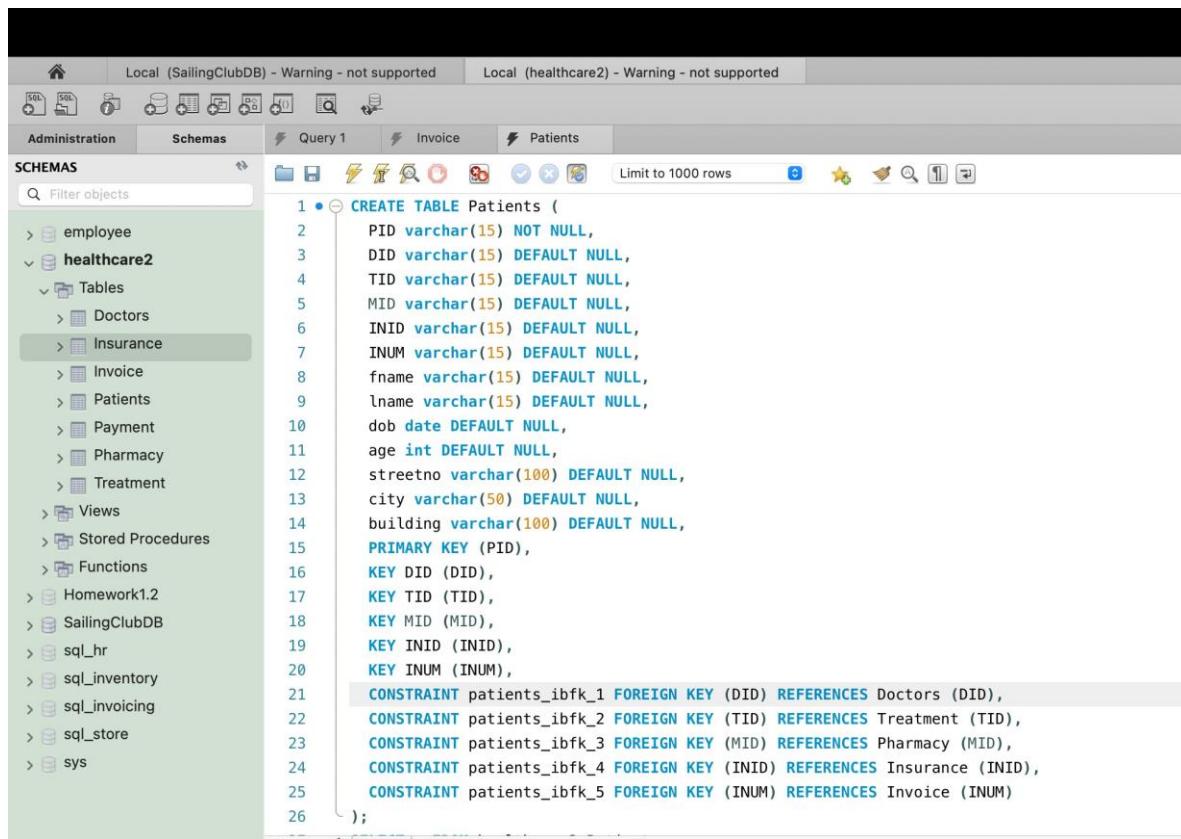
• Create Invoice table



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree, with 'healthcare2' expanded and 'Invoice' selected. The main pane contains the SQL query editor with the following code:

```
1 • CREATE TABLE Invoice (
2     INUM varchar(15) NOT NULL,
3     PNUM varchar(15) DEFAULT NULL,
4     invoicedate date DEFAULT NULL,
5     amount decimal(10,2) DEFAULT NULL,
6     PRIMARY KEY (INUM),
7     KEY PNUM (PNUM),
8     CONSTRAINT invoice_ibfk_1 FOREIGN KEY (PNUM) REFERENCES Payment (PNUM)
9 )
10 SELECT * FROM healthcare2.Invoice;
```

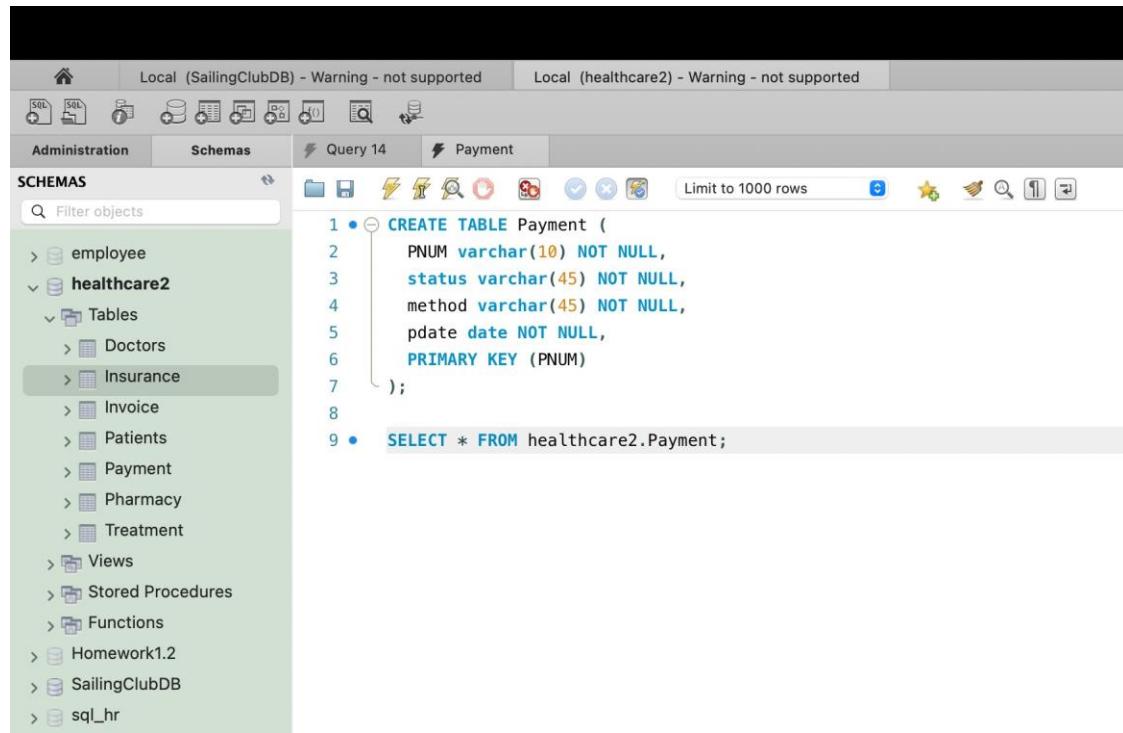
•Create Patients table



The screenshot shows the MySQL Workbench interface with the 'Schemas' tab selected. In the left pane, the 'healthcare2' schema is expanded, and the 'Insurance' table is selected. The main pane displays the SQL code for creating the 'Patients' table:

```
1 •  CREATE TABLE Patients (
2     PID varchar(15) NOT NULL,
3     DID varchar(15) DEFAULT NULL,
4     TID varchar(15) DEFAULT NULL,
5     MID varchar(15) DEFAULT NULL,
6     INID varchar(15) DEFAULT NULL,
7     INUM varchar(15) DEFAULT NULL,
8     fname varchar(15) DEFAULT NULL,
9     lname varchar(15) DEFAULT NULL,
10    dob date DEFAULT NULL,
11    age int DEFAULT NULL,
12    streetno varchar(100) DEFAULT NULL,
13    city varchar(50) DEFAULT NULL,
14    building varchar(100) DEFAULT NULL,
15    PRIMARY KEY (PID),
16    KEY DID (DID),
17    KEY TID (TID),
18    KEY MID (MID),
19    KEY INID (INID),
20    KEY INUM (INUM),
21    CONSTRAINT patients_ibfk_1 FOREIGN KEY (DID) REFERENCES Doctors (DID),
22    CONSTRAINT patients_ibfk_2 FOREIGN KEY (TID) REFERENCES Treatment (TID),
23    CONSTRAINT patients_ibfk_3 FOREIGN KEY (MID) REFERENCES Pharmacy (MID),
24    CONSTRAINT patients_ibfk_4 FOREIGN KEY (INID) REFERENCES Insurance (INID),
25    CONSTRAINT patients_ibfk_5 FOREIGN KEY (INUM) REFERENCES Invoice (INUM)
26 );
```

•Create Payment table



The screenshot shows the MySQL Workbench interface. The left sidebar displays the database schema with the 'Payment' table selected under the 'Tables' section of the 'healthcare2' schema. The main pane shows the SQL query editor with the following code:

```
1 • CREATE TABLE Payment (
2     PNUM varchar(10) NOT NULL,
3     status varchar(45) NOT NULL,
4     method varchar(45) NOT NULL,
5     pdate date NOT NULL,
6     PRIMARY KEY (PNUM)
7 );
8
9 •   SELECT * FROM healthcare2.Payment;
```

• Create Pharmacy table

The screenshot shows a SQL management interface with the following details:

- Top Bar:** Local (SailingClubDB) - Warning - not supported | Local (healthcare2) - Warning - not supported
- Toolbar:** Home, SQL, Scripts, Database, Schemas, Tables, Views, Procedures, Functions, Indexes, Constraints, Triggers, Events, Logins, Security, Help.
- Navigation:** Administration, Schemas, Query 14, Payment, Pharmacy.
- Object Explorer (Schema Tree):**
 - SCHEMAS:
 - employee
 - healthcare2
 - Tables
 - Doctors
 - Insurance
 - Invoice
 - Patients
 - Payment
 - Pharmacy
 - Treatment
 - Views
 - Stored Procedures
 - Functions
 - Homework1.2
 - SailingClubDB
 - sql_hr
- Query Editor:**
 - CREATE TABLE Pharmacy (
 - MID varchar(15) NOT NULL,
 - price decimal(10,2) DEFAULT NULL,
 - quantity int DEFAULT NULL,
 - mname varchar(255) DEFAULT NULL,
 - PRIMARY KEY (MID)
 -);
 - SELECT * FROM healthcare2.Pharmacy;
- Toolbar (Query Editor):** Filter objects, Limit to 1000 rows, Save, Undo, Redo, Find, Replace, Copy, Paste, Cut, Delete, Refresh, Help.

- Create Treatment table

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree, with 'healthcare2' selected. Under 'Tables', 'Treatment' is highlighted. The main pane shows two SQL statements:

```
1 • CREATE TABLE Treatment (
2     TID varchar(15) NOT NULL,
3     price decimal(10,2) DEFAULT NULL,
4     category varchar(255) DEFAULT NULL,
5     PRIMARY KEY (TID)
6 );
7 • SELECT * FROM healthcare2.Treatment;
```

DATA INSERTION FOR THE ABOVE TABLES:

- DATA INSERTION INTO DOCTORS TABLE

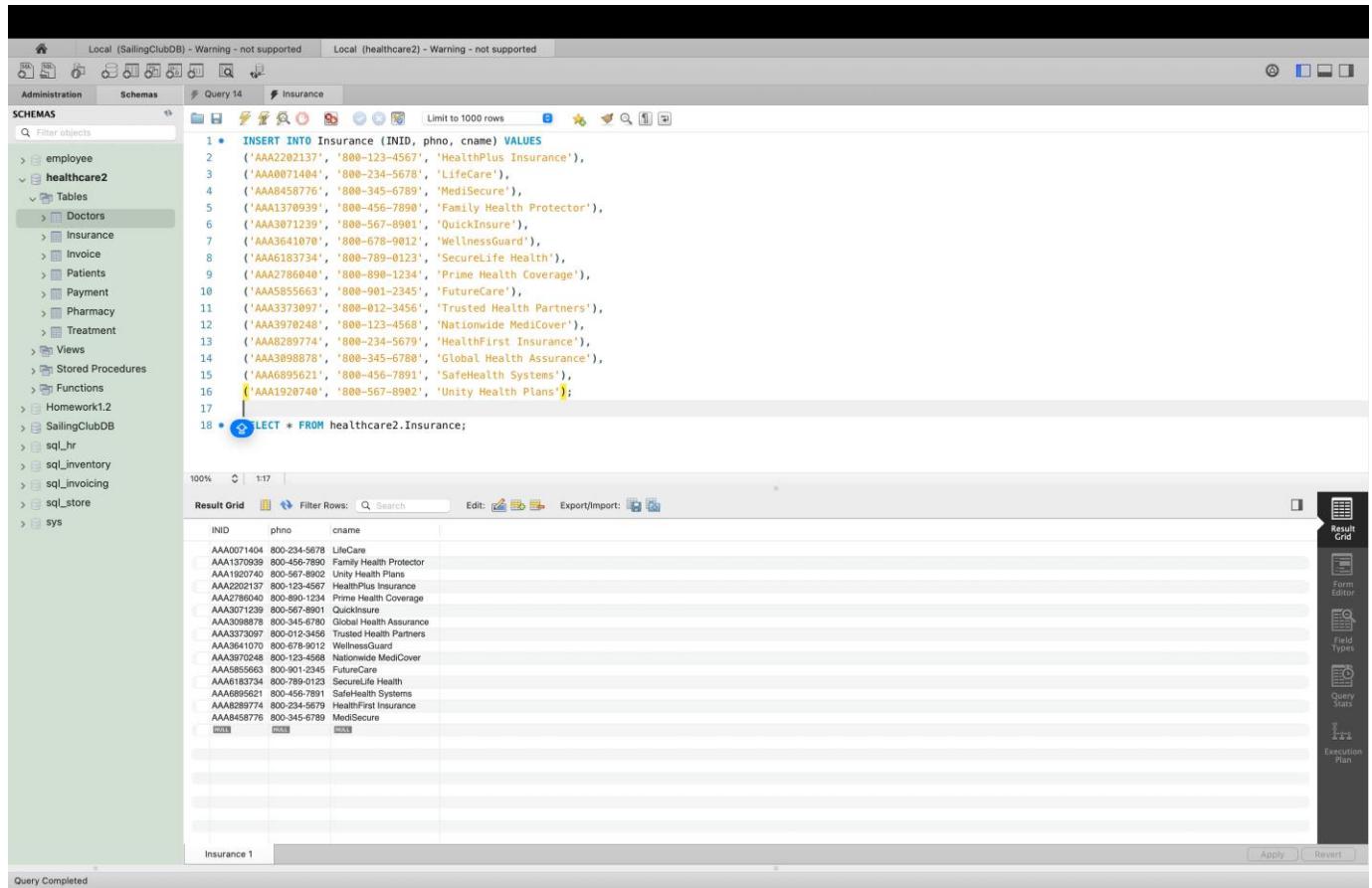
The screenshot shows the Oracle SQL Developer interface with a query editor window. The query is as follows:

```
1 • INSERT INTO Doctors (DID, fname, lname, phno, gender, specialist) VALUES
2 ('DR1001', 'John', 'Doe', '1234567890', 'Male', 'Cardiologist'),
3 ('DR1002', 'Jane', 'Smith', '2345678901', 'Female', 'Neurologist'),
4 ('DR1003', 'Michael', 'Brown', '3456789012', 'Male', 'Orthopedic'),
5 ('DR1004', 'Emily', 'Jones', '4567890123', 'Female', 'Dermatologist'),
6 ('DR1005', 'William', 'Garcia', '5678901234', 'Male', 'Pediatrician'),
7 ('DR1006', 'Olivia', 'Martinez', '6789012345', 'Female', 'Gynecologist'),
8 ('DR1007', 'David', 'Rodriguez', '7890123456', 'Male', 'Psychiatrist'),
9 ('DR1008', 'Sophia', 'Wilson', '8901234567', 'Female', 'Ophthalmologist'),
10 ('DR1009', 'James', 'Anderson', '9012345678', 'Male', 'Endocrinologist'),
11 ('DR1010', 'Isabella', 'Thomas', '0123456789', 'Female', 'Anesthesiologist'),
12 ('DR1011', 'Ethan', 'Taylor', '1234567890', 'Male', 'Radiologist'),
13 ('DR1012', 'Ava', 'Moore', '2345610987', 'Female', 'Oncologist'),
14 ('DR1013', 'Matthew', 'Jackson', '3456721098', 'Male', 'Cardiologist'),
15 ('DR1014', 'Sofia', 'Lee', '4567832109', 'Female', 'Neurologist'),
16 ('DR1015', 'Daniel', 'Perez', '5678943210', 'Male', 'Orthopedic');
17
18 • SELECT * FROM healthcare2.Doctors;
```

The result grid shows the inserted data:

DID	fname	lname	phno	gender	specialist
DR1001	John	Doe	1234567890	Male	Cardiologist
DR1002	Jane	Smith	2345678901	Female	Neurologist
DR1003	Michael	Brown	3456789012	Male	Orthopedic
DR1004	Emily	Jones	4567890123	Female	Dermatologist
DR1005	William	Garcia	5678901234	Male	Pediatrician
DR1006	Olivia	Martinez	6789012345	Female	Gynecologist
DR1007	David	Rodriguez	7890123456	Male	Psychiatrist
DR1008	Sophia	Wilson	8901234567	Female	Ophthalmologist
DR1009	James	Anderson	9012345678	Male	Endocrinologist
DR1010	Isabella	Thomas	0123456789	Female	Anesthesiologist
DR1011	Ethan	Taylor	1234567890	Male	Radiologist
DR1012	Ava	Moore	2345610987	Female	Oncologist
DR1013	Matthew	Jackson	3456721098	Male	Cardiologist
DR1014	Sofia	Lee	4567832109	Female	Neurologist
DR1015	Daniel	Perez	5678943210	Male	Orthopedic

• DATA INSERTION INTO INSURANCE TABLE



The screenshot shows the Oracle SQL Developer interface with a query editor window. The code entered is:

```

1 • INSERT INTO Insurance (INID, phno, cname) VALUES
2 ('AAA2202137', '800-123-4567', 'HealthPlus Insurance'),
3 ('AAA0071404', '800-234-5678', 'LifeCare'),
4 ('AAA0458776', '800-345-6789', 'MediSecure'),
5 ('AAA1370939', '800-456-7890', 'Family Health Protector'),
6 ('AAA30871239', '800-567-8901', 'QuickInsure'),
7 ('AAA3641070', '800-678-9012', 'WellnessGuard'),
8 ('AAA6183734', '800-789-0123', 'SecureLife Health'),
9 ('AAA2786040', '800-890-1234', 'Prime Health Coverage'),
10 ('AAA5855663', '800-901-2345', 'FutureCare'),
11 ('AAA3373097', '800-012-3456', 'Trusted Health Partners'),
12 ('AAA3970248', '800-123-4568', 'Nationwide MediCover'),
13 ('AAA8289774', '800-234-5679', 'HealthFirst Insurance'),
14 ('AAA3098878', '800-345-6780', 'Global Health Assurance'),
15 ('AAA6895621', '800-456-7891', 'SafeHealth Systems'),
16 ('AAA1920748', '800-567-8902', 'Unity Health Plans');
17
18 • SELECT * FROM healthcare2.Insurance;

```

The Result Grid shows the inserted data:

INID	phno	cname
AAA0071404	800-234-5678	LifeCare
AAA1370939	800-456-7890	Family Health Protector
AAA1920740	800-567-8902	Unity Health Plans
AAA2202137	800-123-4567	HealthPlus Insurance
AAA2786040	800-890-1234	Prime Health Coverage
AAA30871239	800-567-8901	QuickInsure
AAA3641070	800-678-9012	WellnessGuard
AAA3970248	800-123-4568	Nationwide MediCover
AAA5855663	800-901-2345	FutureCare
AAA6183734	800-789-0123	SecureLife Health
AAA6895621	800-456-7891	SafeHealth Systems
AAA8289774	800-234-5679	HealthFirst Insurance
AAA9458776	800-345-6789	MediSecure

•DATA INSERTION INTO INVOICE TABLE

```

Local (SailingClubDB) - Warning - not supported Local (healthcare2) - Warning - not supported
Administration Schemas Query 14 Insurance Invoice
SCHEMAS
  Filter objects
  employee
  healthcare2
    Tables
      Doctors
      Insurance
      Invoice
      Patients
      Payment
      Pharmacy
      Treatment
    Views
    Stored Procedures
    Functions
  Homework1.2
  SailingClubDB
  sql_hr
  sql_inventory
  sql_invoicing
  sql_store
  sys
  Other objects
  Limit to 1000 rows
  1 • INSERT INTO Invoice (INUM, PNUM, invoicedate, amount) VALUES
  2   ('IN1234', 'PPP97532', '2024-01-10', 170),
  3   ('IN5678', 'PPP62487', '2024-02-15', 335),
  4   ('IN9101', 'PPP21549', '2024-03-20', 305),
  5   ('IN1213', 'PPP38741', '2024-04-25', 575),
  6   ('IN1415', 'PPP76492', '2024-05-30', 115),
  7   ('IN1617', 'PPP14926', '2024-06-04', 225),
  8   ('IN1819', 'PPP53879', '2024-07-09', 455),
  9   ('IN2021', 'PPP69214', '2024-08-14', 395),
  10  ('IN2223', 'PPP83157', '2024-09-19', 515),
  11  ('IN2425', 'PPP46328', '2024-10-24', 680),
  12  ('IN2627', 'PPP78311', '2024-11-28', 645),
  13  ('IN2829', 'PPP53674', '2024-12-03', 815),
  14  ('IN3031', 'PPP31748', '2025-01-07', 875),
  15  ('IN3233', 'PPP68412', '2025-02-11', 935),
  16  ('IN3435', 'PPP45981', '2025-03-18', 995);
  17
  18 • SELECT * FROM healthcare2.Invoice;
  
```

Result Grid | Filter Rows: Q Search | Edit: | Export/Import: |

INUM	PNUM	invoicedate	amount
IN1213	PPP38741	2024-04-25	575.00
IN1415	PPP76492	2024-05-30	115.00
IN1617	PPP14926	2024-06-04	225.00
IN1819	PPP53879	2024-07-09	455.00
IN2021	PPP69214	2024-08-14	395.00
IN2223	PPP83157	2024-09-19	515.00
IN2425	PPP46328	2024-10-24	680.00
IN2627	PPP78311	2024-11-28	645.00
IN2829	PPP53674	2024-12-03	815.00
IN3031	PPP31748	2025-01-07	875.00
IN3233	PPP68412	2025-02-11	935.00
IN3435	PPP45981	2025-03-18	995.00

Invoice 1 | Apply | Revert |

Query Completed

• DATA INSERTION INTO PATIENTS TABLE

Schemas Administration Schemas Query 14 Insurance Invoice Patients

Limit to 1000 rows

1 • INSERT INTO Patients (PID, DID, TID, MID, INID, INUM, fname, lname, dob, age, streetno, city, building) VALUES

```

2 ('PT2016', 'DR1001', 'TR3001', 'MD03692891', 'AAA2202137', 'IN1234', 'Alice', 'Wong', '1990-01-01', 34, '123', 'New York', 'Liberty Towers'),
3 ('PT2017', 'DR1002', 'TR3002', 'MD09869879', 'AAA0071404', 'IN5678', 'Bob', 'Smith', '1985-02-12', 39, '456', 'Los Angeles', 'Sunset Plaza'),
4 ('PT2018', 'DR1003', 'TR3003', 'MD07368580', 'AAA8458776', 'IN9101', 'Charlie', 'Johnson', '1978-03-23', 46, '789', 'Chicago', 'Lake House'),
5 ('PT2019', 'DR1004', 'TR3004', 'MD047609337', 'AAA1370939', 'IN1213', 'Daisy', 'Lee', '1992-04-14', 32, '1011', 'Houston', 'Pine Apartments'),
6 ('PT2020', 'DR1005', 'TR3005', 'MD041093215', 'AAA3071239', 'IN1415', 'Ethan', 'Brown', '1980-05-25', 44, '1213', 'Phoenix', 'Desert Springs'),
7 ('PT2021', 'DR1006', 'TR3006', 'MD083218141', 'AAA43641070', 'IN1617', 'Fiona', 'Davis', '1994-06-06', 30, '1415', 'Philadelphia', 'River View'),
8 ('PT2022', 'DR1007', 'TR3007', 'MD093711778', 'AAA6183734', 'IN1819', 'George', 'Miller', '1982-07-17', 42, '1617', 'San Antonio', 'Oak Villas'),
9 ('PT2023', 'DR1008', 'TR3008', 'MD018868736', 'AAA2786040', 'IN2021', 'Hannah', 'Wilson', '1975-08-28', 49, '1819', 'San Diego', 'Marina Side'),
10 ('PT2024', 'DR1009', 'TR3009', 'MD093487613', 'AAA5055663', 'IN2223', 'Ivan', 'Moore', '1988-09-09', 36, '2021', 'Dallas', 'Highland Loft'),
11 ('PT2025', 'DR1010', 'TR3010', 'MD027465264', 'AAA3373897', 'IN2425', 'Julia', 'Taylor', '1991-10-10', 33, '2223', 'San Jose', 'Central Station'),
12 ('PT2026', 'DR1011', 'TR3011', 'MD16189598', 'AAA3970248', 'IN2627', 'Kevin', 'Anderson', '1983-11-11', 41, '2425', 'Austin', 'Greenwood Residences'),
13 ('PT2027', 'DR1012', 'TR3012', 'MD012194733', 'AAA8289774', 'IN2829', 'Lily', 'Thomas', '1979-12-12', 45, '2627', 'Jacksonville', 'Beachfront Homes'),
14 ('PT2028', 'DR1013', 'TR3013', 'MD045485169', 'AAA3098878', 'IN3031', 'Mason', 'Jackson', '1986-01-13', 38, '2829', 'San Francisco', 'Golden Gates'),
15 ('PT2029', 'DR1014', 'TR3014', 'MD027787881', 'AAA6895621', 'IN3233', 'Nora', 'Martinez', '1993-02-14', 31, '3031', 'Indianapolis', 'Maple Street'),
16 ('PT2030', 'DR1015', 'TR3015', 'MD022587921', 'AAA1920740', 'IN3435', 'Oscar', 'Hernandez', '1984-03-15', 40, '3233', 'Columbus', 'River Park');
17
18 • SELECT * FROM healthcare2.Patients;
```

Result Grid Filter Rows: Search Edit: Export/Import:

PID	DID	TID	MID	INID	INUM	fname	lname	dob	age	streetno	city	building
PT2016	DR1001	TR3001	MD03692891	AAA2202137	IN1234	Alice	Wong	1990-01-01	34	123	New York	Liberty Towers
PT2017	DR1002	TR3002	MD09869879	AAA0071404	IN5678	Bob	Smith	1985-02-12	39	456	Los Angeles	Sunset Plaza
PT2018	DR1003	TR3003	MD07368580	AAA8458776	IN9101	Charlie	Johnson	1978-03-23	46	789	Chicago	Lake House
PT2019	DR1004	TR3004	MD047609337	AAA1370939	IN1213	Daisy	Lee	1992-04-14	32	1011	Houston	Pine Apartments
PT2020	DR1005	TR3005	MD041093215	AAA3071239	IN1415	Ethan	Brown	1980-05-25	44	1213	Phoenix	Desert Springs
PT2021	DR1006	TR3006	MD083218141	AAA43641070	IN1617	Fiona	Davis	1994-06-06	30	1415	Philadelphia	River View
PT2022	DR1007	TR3007	MD093711778	AAA6183734	IN1819	George	Miller	1982-07-17	42	1617	San Antonio	Oak Villas
PT2023	DR1008	TR3008	MD018868736	AAA2786040	IN2021	Hannah	Wilson	1975-08-28	49	1819	San Diego	Marina Side
PT2024	DR1009	TR3009	MD093487613	AAA5055663	IN2223	Ivan	Moore	1988-09-09	36	2021	Dallas	Highland Loft
PT2025	DR1010	TR3010	MD027465264	AAA3373897	IN2425	Julia	Taylor	1991-10-10	33	2223	San Jose	Central Station
PT2026	DR1011	TR3011	MD16189598	AAA3970248	IN2627	Kevin	Anderson	1983-11-11	41	2425	Austin	Greenwood Residences
PT2027	DR1012	TR3012	MD012194733	AAA8289774	IN2829	Lily	Thomas	1979-12-12	45	2627	Jacksonville	Beachfront Homes
PT2028	DR1013	TR3013	MD045485169	AAA3098878	IN3031	Mason	Jackson	1986-01-13	38	2829	San Francisco	Golden Gates
PT2029	DR1014	TR3014	MD027787881	AAA6895621	IN3233	Nora	Martinez	1993-02-14	31	3031	Indianapolis	Maple Street
PT2030	DR1015	TR3015	MD022587921	AAA1920740	IN3435	Oscar	Hernandez	1984-03-15	40	3233	Columbus	River Park

Patients 1

Query Completed

•DATA INSERTION INTO PAYMENTS TABLE

The screenshot shows a database interface with a query editor and a results grid. The query editor contains the following SQL code:

```

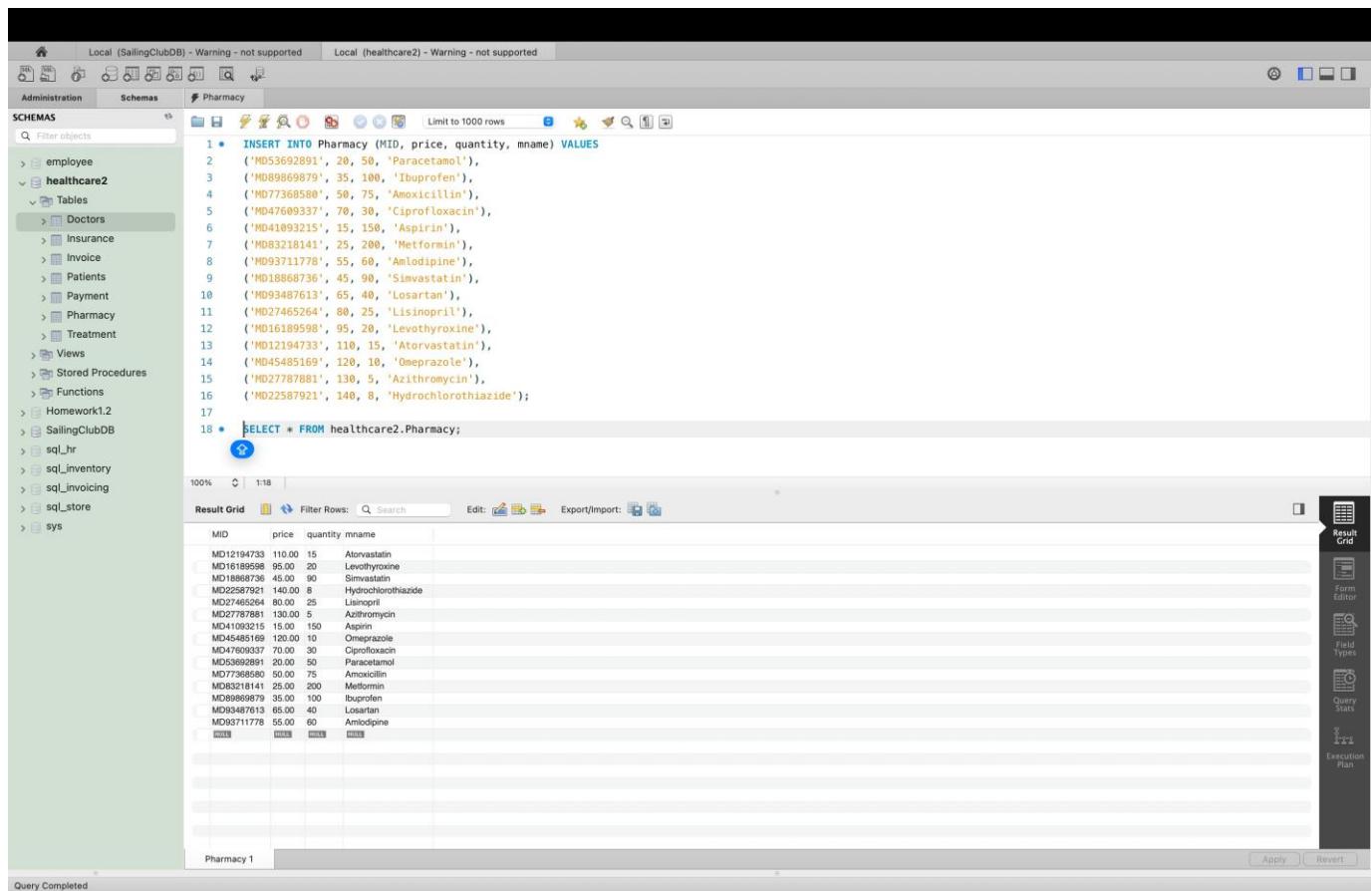
1 • INSERT INTO Payment (PNUM, status, method, pdate) VALUES
2 ('PPP97532', 'Completed', 'Credit Card', '2024-01-15'),
3 ('PPP92487', 'Pending', 'Bank Transfer', '2024-02-28'),
4 ('PPP21549', 'Completed', 'Cash', '2024-03-25'),
5 ('PPP38741', 'Completed', 'Credit Card', '2024-04-30'),
6 ('PPP76492', 'Pending', 'Cash', '2024-05-05'),
7 ('PPP14926', 'Completed', 'Bank Transfer', '2024-06-10'),
8 ('PPP53879', 'Completed', 'Credit Card', '2024-07-15'),
9 ('PPP69214', 'Completed', 'Cash', '2024-08-20'),
10 ('PPP83157', 'Pending', 'Bank Transfer', '2024-09-25'),
11 ('PPP46328', 'Completed', 'Credit Card', '2024-10-30'),
12 ('PPP57831', 'Completed', 'Cash', '2024-11-04'),
13 ('PPP92674', 'Completed', 'Bank Transfer', '2024-12-09'),
14 ('PPP31748', 'Pending', 'Credit Card', '2025-01-13'),
15 ('PPP68412', 'Completed', 'Cash', '2025-02-17'),
16 ('PPP45981', 'Completed', 'Bank Transfer', '2025-03-24');
17
18 • SELECT * FROM healthcare2.Payment;
  
```

The results grid displays the inserted data:

PNUM	status	method	pdate
PPP14926	Completed	Bank Transfer	2024-06-10
PPP92487	Completed	Credit Card	2024-02-28
PPP21549	Completed	Credit Card	2024-03-25
PPP38741	Completed	Credit Card	2024-04-30
PPP76492	Pending	Cash	2024-05-05
PPP14926	Completed	Bank Transfer	2024-06-10
PPP53879	Completed	Credit Card	2024-07-15
PPP69214	Completed	Cash	2024-08-20
PPP83157	Pending	Bank Transfer	2024-09-25
PPP46328	Completed	Credit Card	2024-10-30
PPP57831	Completed	Cash	2024-11-04
PPP92674	Completed	Bank Transfer	2024-12-09
PPP31748	Pending	Credit Card	2025-01-13
PPP68412	Completed	Cash	2025-02-17
PPP45981	Completed	Bank Transfer	2025-03-24

The interface includes various toolbars, a schema browser on the left, and a toolbar on the right with options like Result Grid, Form Editor, Field Types, Query Log, and Execution Plan.

• DATA INSERTION INTO PHARMACY TABLE



The screenshot shows the Oracle SQL Developer interface with a query editor window. The code entered is:

```

1 • INSERT INTO Pharmacy (MID, price, quantity, mname) VALUES
2 ('MD53692891', 20, 50, 'Paracetamol'),
3 ('MD89869879', 35, 100, 'Ibuprofen'),
4 ('MD77368580', 50, 75, 'Amoxicillin'),
5 ('MD47609337', 70, 30, 'Ciprofloxacin'),
6 ('MD41893215', 15, 150, 'Aspirin'),
7 ('MD83218141', 25, 200, 'Metformin'),
8 ('MD93711778', 55, 60, 'Amlodipine'),
9 ('MD18868736', 45, 90, 'Simvastatin'),
10 ('MD93487613', 65, 40, 'Losartan'),
11 ('MD27465264', 80, 25, 'Lisinopril'),
12 ('MD16189598', 95, 20, 'Levothyroxine'),
13 ('MD12194733', 110, 15, 'Atorvastatin'),
14 ('MD45485169', 120, 10, 'Omeprazole'),
15 ('MD27787881', 130, 5, 'Azithromycin'),
16 ('MD22587921', 140, 8, 'Hydrochlorothiazide');
17
18 • SELECT * FROM healthcare2.Pharmacy;

```

The Result Grid shows the inserted data:

MID	price	quantity	mname
MD12194733	110.00	15	Atorvastatin
MD16189598	95.00	20	Levothyroxine
MD89869879	45.00	90	Simvastatin
MD22587921	140.00	8	Hydrochlorothiazide
MD27465264	80.00	25	Lisinopril
MD27787881	130.00	5	Azithromycin
MD41893215	15.00	150	Aspirin
MD45485169	120.00	10	Omeprazole
MD47609337	70.00	30	Ciprofloxacin
MD53692891	20.00	50	Paracetamol
MD77368580	50.00	75	Amoxicillin
MD83218141	25.00	200	Metformin
MD89869879	35.00	100	Ibuprofen
MD93487613	65.00	40	Losartan
MD93711778	55.00	60	Amlodipine

•DATA INSERTION INTO TREATMENT TABLE

```

Local (SailingClubDB) - Warning - not supported Local (healthcare2) - Warning - not supported
Administration Schemas Treatment
SCHEMAS Filter objects
employee healthcare2
Tables Doctors Insurance Invoice Patients Payment Pharmacy Treatment Views Stored Procedures Functions Homework1.2 SailingClubDB sql_hr sql_inventory sql_invoicing sql_store sys
Limit to 1000 rows
1 • INSERT INTO Treatment (TID, price, category) VALUES
2 ('TR3001', 150, 'General Checkup'),
3 ('TR3002', 300, 'Medical Care'),
4 ('TR3003', 250, 'Vision Test'),
5 ('TR3004', 500, 'Surgery Consultation'),
6 ('TR3005', 100, 'Vaccination'),
7 ('TR3006', 200, 'Physical Therapy'),
8 ('TR3007', 400, 'Cardiology Consultation'),
9 ('TR3008', 350, 'Orthopedics'),
10 ('TR3009', 450, 'Dermatology'),
11 ('TR3010', 600, 'Major Surgery'),
12 ('TR3011', 550, 'Oncology Treatment'),
13 ('TR3012', 700, 'Transplant Surgery'),
14 ('TR3013', 750, 'Neurology Consultation'),
15 ('TR3014', 800, 'Psychiatry'),
16 ('TR3015', 850, 'Pediatrics');
17
18 • SELECT * FROM healthcare2.Treatment;

```

Result Grid Filter Rows: Search Edit: Export/Import:

TID	price	category
TR3001	150.00	General Checkup
TR3002	300.00	Medical Care
TR3003	250.00	Vision Test
TR3004	500.00	Surgery Consultation
TR3005	100.00	Vaccination
TR3006	200.00	Physical Therapy
TR3007	400.00	Cardiology Consultation
TR3008	350.00	Orthopedics
TR3009	450.00	Dermatology
TR3010	600.00	Major Surgery
TR3011	550.00	Oncology Treatment
TR3012	700.00	Transplant Surgery
TR3013	750.00	Neurology Consultation
TR3014	800.00	Psychiatry
TR3015	850.00	Pediatrics

Treatment 1

);

INDEX CREATION

CREATE INDEX idx_doctor_specialist ON Doctors (fname, specialist);

The screenshot shows the MySQL Workbench interface with two queries run against the 'Doctors' table.

Query 1 (Top):

```
1 • CREATE INDEX idx_doctor_specialist ON Doctors (fname, specialist);
2 • SHOW INDEX FROM DOCTORS
```

Query 2 (Bottom):

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
Doctors	0	PRIMARY	1	DID	A	15	NULL	NULL	YES	BTREE		YES	NULL	
Doctors	1	idx_doctor_specialist_1	1	fname	A	15	NULL	NULL	YES	BTREE		YES	NULL	
Doctors	1	idx_doctor_specialist_2	2	specialist	A	15	NULL	NULL	YES	BTREE		YES	NULL	

Message bar at the bottom: Query Completed

CREATE INDEX idx_pharmacy_medicine_name ON Pharmacy (mname);

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** Local (SailingClubDB) - Warning - not supported | Local (healthcare2) - Warning - not supported
- Query Editor:** Query 25 | Pharmacy
- SQL History:** 1 CREATE INDEX idx_pharmacy_medicine_name ON Pharmacy (mname);
2 • SHOW INDEX FROM PHARMACY
- Tables:** Shows the structure of the Pharmacy table.
- Result Grid:** Displays the results of the SHOW INDEX query.
- Table Headers:** Table, Non_unique, Key_name, Seq_in_Index, Column_name, Collation, Cardinality, Sub_part, Packed, Null, Index_type, Comment, Visible, Expression.
- Results:** Two rows are shown:
 - Pharmacy 0 PRIMARY 1 mname A 15 BTREE YES
 - Pharmacy 1 idx_pharmacy_medicine_name 1 mname A 15 BTREE YES
- Right Panel:** Shows icons for Result Grid, Form Editor, Field Types, Query Stats, and Execution Plan.
- Status Bar:** Read Only
- Bottom Status:** Query Completed

VIEWS CREATION:

Create view view_patient AS

```
select PID, fname AS patient_fname, lname patient_lname, streetno, city, building  
from patients;
```

The screenshot shows the Oracle SQL Developer interface. On the left, the 'SCHEMAS' tree view is expanded to show the 'healthcare2' schema, which contains tables like Doctors, Insurance, Invoice, Patients, Payment, Pharmacy, Treatment, Views, and Stored Procedures. The 'Views' node is selected. In the center, a query editor window titled 'Query 25' displays the following SQL code:

```
1 Create view view_patient AS  
2 select PID, fname AS patient_fname, lname patient_lname, streetno, city, building  
3 from patients;  
4 * select * from view_patient
```

Below the code, the 'Result Grid' shows the results of the query. The columns are PID, patient_fname, patient_lname, streetno, city, and building. The data includes rows for Alice Wong, Bob Smith, Charlie Johnson, etc. On the right side of the interface, there are several tabs: Result Grid, Form Editor, Field Types, Error Stats, and Execution Plan. The 'Result Grid' tab is currently active.

PID	patient_fname	patient_lname	streetno	city	building
PT2016	Alice	Wong	123	New York	Liberty Towers
PT2017	Bob	Smith	456	Los Angeles	Sunset Plaza
PT2018	Charlie	Johnson	789	Chicago	Wrigley Apartments
PT2019	Daisy	Lee	1011	Seattle	Pine Apartments
PT2020	Ethan	Brown	1213	Phoenix	Desert Springs
PT2021	Fiona	Davis	1415	Philadelphia	River View
PT2022	George	Miller	1617	San Antonio	Clock Tower
PT2023	Hannah	Wilson	1819	San Diego	Gaslamp Side
PT2024	Ivan	Moore	2021	Dallas	Highland Loft
PT2025	Juli	Taylor	2223	San Jose	Central Station
PT2026	Kevin	Anderson	2425	Austin	Greenwood Re...
PT2027	Lily	Thomas	2627	Charlotte	Southgate Ho...
PT2028	Mason	Jackson	2829	San Frand...	Golden Gate
PT2029	Nora	Martinez	3031	Indianapolis	Maple Street
PT2030	Oscar	Hernandez	3233	Columbus	River Park

• CREATE VIEW

patient_insurance_info AS

```
SELECT
    p.PID,
    p.fname AS patient_fname,
    p.lname AS patient_lname,
    i.INID,
    i.phno AS insurance_phno,
    i.cname AS insurance_company
FROM Patients p
LEFT JOIN Insurance i ON p.INID = i.INID;
```

The screenshot shows a SQL query editor interface with the following details:

- Toolbar:** Includes icons for file operations (New, Open, Save, Print, etc.), search, and various database management functions.
- Menu Bar:** Shows "Local (SailingClubDB) - Warning - not supported" and "Local (healthcare2) - Warning - not supported".
- Schemas:** A tree view showing the database structure under "Schemas". The "Tables" node is expanded, showing "Doctors", "Insurance", "Invoice", "Patients", "Payment", "Pharmacy", "Treatment", and "Views".
- Query Editor:** Displays the SQL code for creating the view "patient_insurance_info". The code is as follows:

```
CREATE VIEW patient_insurance_info AS
SELECT
    p.PID,
    p.fname AS patient_fname,
    p.lname AS patient_lname,
    i.INID,
    i.phno AS insurance_phno,
    i.cname AS insurance_company
FROM Patients p
LEFT JOIN Insurance i ON p.INID = i.INID;
```

The code is highlighted in blue, indicating it is selected. Below the code, there is a note: "12 • Select * from patient_insurance_info".

Result Grid: A table showing the results of the query. The columns are:

PID	patient_fname	patient_lname	INID	insurance_phno	insurance_company
PT2016	Alice	Wong	AAA2202137	800-123-4567	HealthPlus Insurance
PT2017	Bob	Smith	AAA0071404	800-234-5678	LifeCare
PT2018	Charlie	Johnson	AAA1111111	800-345-6789	MedSecure
PT2019	Daisy	Lee	AAA1370539	800-456-7890	Family Health Protector
PT2020	Ethan	Brown	AAA3071239	800-567-8901	Quickicare
PT2021	Fiona	Davis	AAA3641070	800-678-9012	WellnessGuard
PT2022	George	Miller	AAA6183734	800-789-0123	SecureLife Health
PT2023	Hannah	Wilson	AAA2786064	800-890-1234	Prime Health Coverage
PT2024	Ivan	Moore	AAA5855694	800-901-2345	FutureCare
PT2025	Julia	Taylor	AAA3970248	800-123-4598	Trusted Health Partners
PT2026	Karen	Anderson	AAA3970248	800-123-4598	National Health MedCover
PT2027	Lily	Thomas	AAA4899774	800-234-5679	HealthFirst Insurance
PT2028	Mason	Jackson	AAA3098878	800-345-6780	Global Health Assurance
PT2029	Nora	Martinez	AAA68995621	800-456-8991	SafeHealth Systems
PT2030	Oscar	Hernandez	AAA1920740	800-567-8902	Unity Health Plans

The result grid has a header row and 30 data rows. The columns are labeled: PID, patient_fname, patient_lname, INID, insurance_phno, and insurance_company.

Right Panel: Contains tabs for "Result Grid", "Form Editor", "Field Types", "Query Stats", and "Execution Plan".

Status Bar: Shows "Query Completed".

TEMPORARY TABLES CREATION:

• CREATE TEMPORARY TABLE

```
temp_expensive_treatments_patients ASSELECT
    p.PID,
    p.fname AS patient_fname,
    p.lname AS patient_lname,
    t.TID,
    t.price AS treatment_price,
    t.category AS treatment_category
FROM Patients p
JOIN Treatment t ON p.TID =
t.TID WHERE t.price >= 500;
```

The screenshot shows a database management interface with the following details:

- Schemas:** healthcare2
- Tables:** A list of tables including Doctors, Insurance, Invoice, Patients, Payment, Pharmacy, Treatment.
- Code Editor:** Contains the SQL code for creating a temporary table and selecting from it.
- Result Grid:** Displays the results of the query, showing 10 rows of data.
- Toolbar:** Includes icons for file operations, search, and export.
- Right Panel:** Shows tabs for Result Grid, Form Editor, Field Types, Query Stats, and Execution Plan.

```
CREATE TEMPORARY TABLE temp_expensive_treatments_patients AS
SELECT
    p.PID,
    p.fname AS patient_fname,
    p.lname AS patient_lname,
    t.TID,
    t.price AS treatment_price,
    t.category AS treatment_category
FROM Patients p
JOIN Treatment t ON p.TID = t.TID
WHERE t.price >= 500;
SELECT * FROM temp_expensive_treatments_patients;
```

PID	patient_fname	patient_lname	TID	treatment_price	treatment_category
PT2019	Daisy	Lee	TR3004	500.00	Surgery Consultation
PT2025	Julia	Taylor	TR3010	600.00	Major Surgery
PT2026	Alexander	Anderson	TR3012	650.00	Oncology Treatment
PT2027	Lily	Thomas	TR3012	700.00	Transplant Surgery
PT2028	Mason	Jackson	TR3013	750.00	Neurology Consultation
PT2029	Nora	Martinez	TR3014	800.00	Psychiatry
PT2030	Oscar	Hernandez	TR3015	850.00	Pediatrics

TRIGGERS CREATION:

- Trigger to set amount as 0 if the value is less than 0.

```
DELIMITER //
CREATE TRIGGER Triggers_Invoice BEFORE
INSERT ON invoice
FOR EACH
ROWBEGIN
IF NEW.amount < 0
THENSET
NEW.amount = 0;
END
IF;
END;
//
DELIMITE
R ;
```

```
INSERT into invoice (INUM,PNUM,invoicedate,amount)
values('IN10001','PPP38741','2024-01-01','-100');
SELECT * FROM invoice;
```

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** healthcare2
- Tables:** Tables (under healthcare2)
- Query Grid:** Shows the trigger definition and the executed SQL statements.
- Result Grid:** Shows the results of the SELECT query, listing various invoices with their details.
- Action Output:** Shows the execution log with rows numbered 1 through 18, detailing each step of the query execution.
- Response:** Shows the number of rows returned for each step.
- Duration / Fetch Time:** Shows the execution time for each step.

Trigger Definition (Query Grid):

```
DELIMITER //
CREATE TRIGGER Triggers_Invoice
BEFORE INSERT ON invoice
FOR EACH ROW
BEGIN
IF NEW.amount < 0 THEN
SET NEW.amount = 0;
END IF;
//
```

Execution Log (Action Output):

Row	Time	Action	Response	Duration / Fetch Time
1	16:09:48	Delimited SQL - Inquire mainmenu2.mainmenu LIMIT 0, 1000	0 rows(s) returned	0.001 sec / 0.00091 sec
2	16:09:49	SELECT * FROM healthcare2.Treatment LIMIT 0, 1000	15 row(s) returned	0.006 sec / 0.00088 sec
3	16:09:50	SELECT * FROM healthcare2.Doctors LIMIT 0, 1000	15 row(s) returned	0.006 sec / 0.00088 sec
4	16:09:51	SELECT * FROM healthcare2.Insurance LIMIT 0, 1000	15 row(s) returned	0.006 sec / 0.00088 sec
5	17:00:11	SELECT * FROM healthcare2.Insurance LIMIT 0, 1000	15 row(s) returned	0.006 sec / 0.00088 sec
6	17:00:12	SELECT * FROM healthcare2.Insurance LIMIT 0, 1000	15 row(s) returned	0.006 sec / 0.00088 sec
7	17:07:19	SELECT * FROM healthcare2.Insurance LIMIT 0, 1000	15 row(s) returned	0.006 sec / 0.0012 sec

- Trigger to calculate the age from date of birth.

DELIMITER //

```
CREATE TRIGGER CalculatePatientAge BEFORE
INSERT ON Patients
FOR EACH
ROWBEGIN
    SET NEW.age = TIMESTAMPDIFF(YEAR, NEW.dob,
CURDATE());END;
//
DELIMITE
R;
```

```
INSERT INTO Patients (PID, DID, TID, MID, INID, Inum, fname, lname, DOB,
streetno, city, building) VALUES
('PT201678', 'DR1001', 'TR3001', 'MD53692891', 'AAA2202137',
'IN1234', 'ANAYNA', 'GUPTA', '1997-01-01', '123', 'New York', 'Liberty
Towers');
```

SELECT * FROM Patients;

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** healthcare2
- Tables:** Patients
- Code Editor:**

```
DELIMITER //
CREATE TRIGGER CalculatePatientAge
BEFORE INSERT ON Patients
FOR EACH ROW
BEGIN
    SET NEW.age = TIMESTAMPDIFF(YEAR, NEW.dob, CURDATE());
END;
//
DELIMITE
R;
```
- Result Grid:** Shows the data inserted into the Patients table:

PID	DID	TID	MID	INID	Inum	fname	lname	DOB	age	streetno	city	building
PT2016	DR1001	TR3001	MD53692891	AAA2202137	IN1234	Alice	Wong	1997-01-01	34	123	New York	Liberty Towers
PT2017	DR1001	TR3001	MD53692891	AAA2202137	IN1234	ANAYNA	GUPTA	1997-01-01	27	123	New York	Liberty Towers
PT217	DR1002	TR3002	MD86868679	AAA2071404	IN5678	Bob	Smith	1965-02-12	39	456	Los Angeles	Sunset Plaza
PT218	DR1002	TR3002	MD86868680	AAA2071405	IN9778	Charlie	Johnson	1978-03-23	46	789	Chicago	Lake House
PT219	DR1004	TR3004	MD41993215	AAA3071239	IN1013	Dave	Anderson	1973-04-15	40	1011	Portland	Portland Apartments
PT220	DR1005	TR3005	MD41993215	AAA3071239	IN1415	Ethan	Brown	1960-05-25	44	1213	Phoenix	Desert Springs
PT221	DR1006	TR3006	MD82181141	AAA3610705	IN1117	Flora	Davis	1994-06-06	30	1413	Philadelphia	River View
PT222	DR1007	TR3007	MD82181142	AAA3610705	IN1118	George	Ford	1971-07-07	28	1613	Seattle	Space Needle View
PT223	DR1008	TR3008	MD16866736	AAA2796040	IN0521	Hannah	Wilson	1975-08-28	49	1819	San Diego	Marina Side
PT224	DR1009	TR3009	MD93487613	AAA5855640	IN2223	Ivan	Moon	1988-09-09	36	2021	Dallas	Highland Loft
PT225	DR1010	TR3010	MD27465264	AAA3373097	IN2425	Julie	Taylor	1991-10-10	33	2223	San Jose	Central Station
PT226	DR1011	TR3011	MD27465265	AAA3373098	IN2426	Karen	Williams	1992-11-11	32	2426	Oakland	Greenwood Re.
PT227	DR1012	TR3012	MD1194733	AAA8289774	IN2529	Lily	Thomas	1979-12-12	45	2627	Jacksonville	Beachfront Ho.
PT228	DR1013	TR3013	MD45485169	AAA3098878	IN0331	Mason	Jackson	1986-01-13	38	2827	San Francisco	Golden Gates
PT229	DR1014	TR3014	MD27787881	AAA8895621	IN2333	Nora	Marinez	1993-03-14	31	3031	Indianapolis	Maple Street
PT230	DR1015	TR3015	MD25879291	AAA1980740	IN435	Oscar	Hernandez	1988-03-15	40	3235	Columbus	River Park
- Action Output:**

Time	Action	Response	Duration / Fetch Time
07:37:55	SELECT * FROM healthcare2.Patients LIMIT 0, 1000	15 row(s) returned	0.048 sec / 0.00076...
07:38:55	INSERT INTO Patients (PID, DID, TID, MID, INID, Inum, fname, lname, DOB, streetno, city, building) VALUES ('PT201678', 'DR1001', 'TR3001', 'MD53692891', 'AAA2202137', 'IN1234', 'ANAYNA', 'GUPTA', '1997-01-01', '123', 'New York', 'Liberty Towers')	1 row(s) affected	0.093 sec
07:39:13	SELECT * FROM Patients LIMIT 0, 1000	16 row(s) returned	0.040 sec / 0.00070...

- **Trigger to calculate total amount and update invoice amount to total amount of medicines and treatment.**

```

DELIMITER //
CREATE TRIGGER
calculate_invoice_amount_and_update_dateAFTER INSERT
ON Patients
FOR EACH
ROWBEGIN
DECLARE totalPharmacyAmount DECIMAL(10,2);
DECLARE totalTreatmentAmount DECIMAL(10,2);

-- Calculate total pharmacy amount (price * quantity)
SELECT price * quantity INTO totalPharmacyAmount
FROM Pharmacy
WHERE MID = NEW.MID;

-- Get treatment price
SELECT price INTO
totalTreatmentAmountFROM Treatment
WHERE TID = NEW.TID;

-- Update invoice amount and invoice date
UPDATE Invoice
SET amount = totalPharmacyAmount + totalTreatmentAmount,
invoicedate = CURDATE()
WHERE INUM =
NEW.INUM;END;
//
DELIMITE
R ;
INSERT INTO Patients (PID, DID, TID, MID, INID, Inum, fname, lname, DOB,
streetno, city, building) VALUE

```

('PT888', 'DR1001', 'TR3001', 'MD53692891', 'AAA2202137', 'IN1234', 'kavya', 'v',
'1999-01-01', '123', 'New York', 'Liberty

Towers');SELECT * FROM Invoice where

Inum= "IN1234";

The screenshot shows the MySQL Workbench interface. The top bar displays two databases: Local (SailingClubDB) - Warning - not supported and Local (healthcare2) - Warning - not supported. The left sidebar shows the schema structure under Schemas, including the healthcare2 schema which contains tables like Doctors, Insurance, Invoice, Patients, Payment, Pharmacy, Treatment, Views, Stored Procedures, Functions, Homework1.2, SailingClubDB, sql_hr, sql_inventory, sql_invoicing, sql_store, and sys. The main area is a query editor with the following SQL code:

```
DELIMITER //
CREATE TRIGGER calculate_invoice_amount_and_update_date
AFTER INSERT ON Patients
FOR EACH ROW
BEGIN
    DECLARE totalPharmacyAmount DECIMAL(10,2);
    DECLARE totalTreatmentAmount DECIMAL(10,2);

    -- Calculate total pharmacy amount (price * quantity)
    SELECT price * quantity INTO totalPharmacyAmount
    FROM Pharmacy
    WHERE MID = NEW.MID;

    -- Get treatment price
    SELECT price INTO totalTreatmentAmount
    FROM Treatment
    WHERE TID = NEW.TID;

    -- Update invoice amount and invoice date
    UPDATE Invoice
    SET amount = totalPharmacyAmount + totalTreatmentAmount,
        invoicedate = CURDATE()
    WHERE INUM = NEW.INUM;
END;
//
DELIMITER ;

INSERT INTO Patients (PID, DID, TID, MID, INID, Inum, fname, lname, DOB, streetno, city, building) VALUES
('PT888', 'DR1001', 'TR3001', 'MD53692891', 'AAA2202137', 'IN1234', 'kavya', 'v', '1999-01-01', '123', 'New York', 'Liberty Towers');

SELECT * FROM Invoice where Inum= "IN1234";
```

The results grid below the editor shows one row of data:

INUM	PNUM	invoicedate	amount
IN1234	PPPP97532	2024-03-01	1150.00

STORED PROCEDURES CREATION:

- Stored Procedure to update insurance details

DELIMITER //

CREATE PROCEDURE

```
UpdateInsuranceCompany(IN p_inid
VARCHAR(15),
IN p_phno
VARCHAR(20), IN
p_cname VARCHAR(255)
)
BEGIN
UPDATE Insurance
SET phno = p_phno, cname =
p_cname WHERE INID = p_inid;
END;
//  
DELIMITE
R ;
```

```
call UpdateInsuranceCompany ("AAA0071404","123456789","Aetna");
SELECT * FROM INSURANCE WHERE INID ="AAA0071404"
```

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** Local (SellingClubDB) - Warning - not supported, Local (healthcare2) - Warning - not supported.
- Query Grid:** Shows the SQL code for creating the stored procedure and executing it with parameters.
- Result Grid:** Shows the output of the SELECT query, displaying a single row from the INSURANCE table.
- Toolbar:** Includes icons for Save, Undo, Redo, Copy, Paste, Find, Filter, and Export/Import.
- Bottom Panel:** Includes buttons for Result Grid, Paste Filter, Find Rows, Query Stats, and Execution Plan.

```
DELIMITER //
CREATE PROCEDURE UpdateInsuranceCompany(
    IN p_inid VARCHAR(15),
    IN p_phno VARCHAR(20),
    IN p_cname VARCHAR(255)
)
BEGIN
    UPDATE Insurance
    SET phno = p_phno, cname = p_cname
    WHERE INID = p_inid;
END;
//  
call UpdateInsuranceCompany ("AAA0071404","123456789","Aetna");
SELECT * FROM INSURANCE WHERE INID ="AAA0071404";
```

INID	phno	cname
AAA0071404	123456789	Aetna

- **Stored Procedure to add new doctor into Doctors table.**

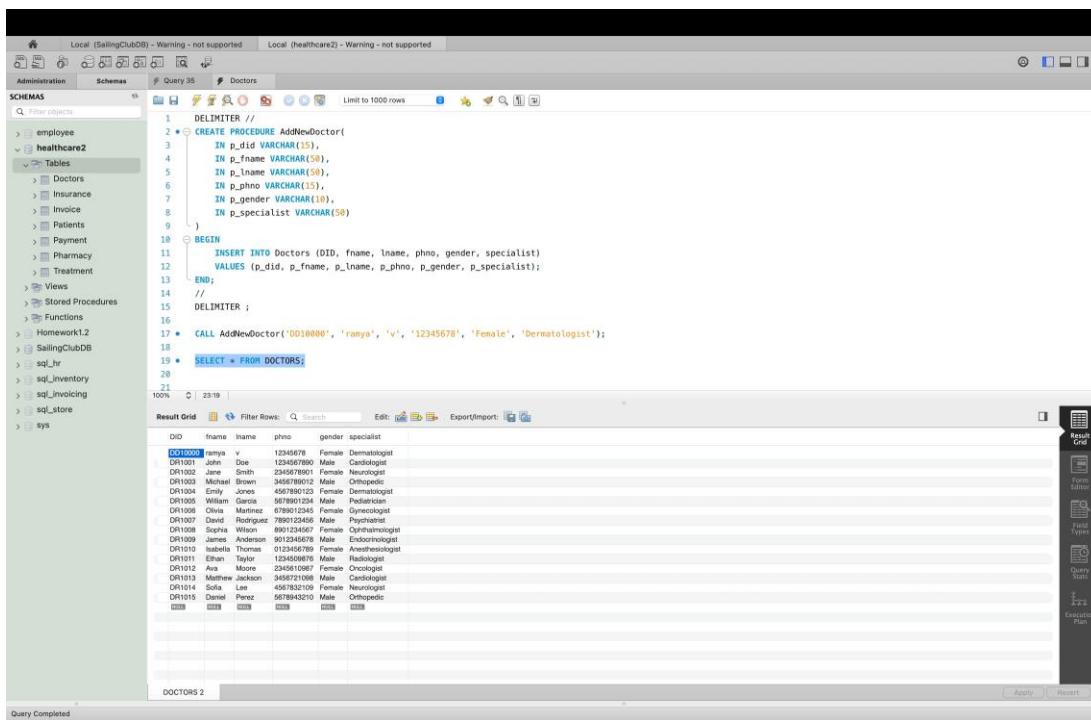
DELIMITER //

CREATE PROCEDURE

```
AddNewDoctor(IN p_did
VARCHAR(15),
IN p_fname
VARCHAR(50),IN
p_lname VARCHAR(50),
IN p_phno
VARCHAR(15), IN
p_gender VARCHAR(10),
IN p_specialist VARCHAR(50)
)
BEGIN
N
INSERT INTO Doctors (DID, fname, lname, phno, gender, specialist)
VALUES (p_did, p_fname, p_lname, p_phno, p_gender, p_specialist);
END;
// DELIMITER ;
```

CALL AddNewDoctor('DD10000', 'ramya', 'v', '12345678', 'Female',
'Dermatologist');

SELECT * FROM DOCTORS;



The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** healthcare2
- Tables:** Doctors
- Code Area:**

```
DELIMITER //
CREATE PROCEDURE AddNewDoctor(
    IN p_did VARCHAR(15),
    IN p_fname VARCHAR(50),
    IN p_lname VARCHAR(50),
    IN p_phno VARCHAR(15),
    IN p_gender VARCHAR(10),
    IN p_specialist VARCHAR(50)
)
BEGIN
    INSERT INTO Doctors (DID, fname, lname, phno, gender, specialist)
    VALUES (p_did, p_fname, p_lname, p_phno, p_gender, p_specialist);
END;
// DELIMITER ;
```
- Execution Area:**

```
17 • CALL AddNewDoctor('DD10000', 'ramya', 'v', '12345678', 'Female', 'Dermatologist');
18
19 • SELECT * FROM DOCTORS;
```
- Result Grid:**

DID	fname	lname	phno	gender	specialist
DD10000	Ramya	V	12345678	Female	Dermatologist
DR10001	David	Brown	5678901234	Male	Orthopedic
DR10002	Jane	Smith	2345678901	Female	Neurologist
DR10003	Michael	Brown	3456789012	Male	Orthopedic
DR10004	Emily	Jones	4567890123	Female	Dermatologist
DR10005	Robert	Anderson	5678901234	Male	Orthopedic
DR10006	Olivia	Martinez	6789012345	Female	Gynecologist
DR10007	David	Rodriguez	7890123456	Male	Psychiatrist
DR10008	Amanda	Anderson	8901234567	Female	Endocrinologist
DR10009	James	Anderson	9012345678	Male	Endocrinologist
DR1010	Isabella	Thomas	0123456789	Female	Anesthesiologist
DR1011	Matthew	Jackson	1234567890	Male	Orthopedic
DR1012	Ava	Moore	2345678901	Female	Oncologist
DR1013	Matthew	Jackson	3456789012	Male	Cardiologist
DR1014	Isabella	Thomas	4567890123	Female	Neurologist
DR1015	Daniel	Perez	5678901234	Male	Orthopedic
- Status Bar:** DOCTORS 2, Apply, Revert
- Bottom Status:** Query Completed

FUNCTIONS CREATION:

- Function to get count of doctors from same specialization

DELIMITER //

CREATE FUNCTION

GetDoctorCountBySpecialization(p_specialization VARCHAR(50))

RETURNS INT

DETERMINIST

IC BEGIN

DECLARE total_count INT;

SELECT COUNT(*) INTO total_count FROM Doctors WHERE specialist = p_specialization;

RETURN

total_count;END;

//DELIMITER ;

```
SELECT * from doctors where GetDoctorCountBySpecialization(Specialist) > 1;
SELECT GetDoctorCountBySpecialization('Neurologist') AS
total_doctors_in_specialization;
```

The screenshot shows the MySQL Workbench interface with a query editor and a result grid.

Query Editor:

```
DELIMITER //
CREATE FUNCTION GetDoctorCountBySpecialization(p_specialization VARCHAR(50)) RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE total_count INT;
    SELECT COUNT(*) INTO total_count FROM Doctors WHERE specialist = p_specialization;
    RETURN total_count;
END;
DELIMITER ;
SELECT * from doctors where GetDoctorCountBySpecialization(Specialist) > 1;
SELECT GetDoctorCountBySpecialization('Neurologist') AS total_doctors_in_specialization;
```

Result Grid:

DID	fname	lname	phno	gender	specialist
DR10000	Ramya	V	12345678	Female	Dermatologist
DR10001	John	Doe	1234567890	Male	Cardiologist
DR10002	Sarah	Smith	12345678901	Female	Orthopedic
DR1003	Michael	Brown	3456789012	Male	Orthopedic
DR1004	Emily	Jones	4567890123	Female	Dermatologist
DR1005	Matthew	Jackson	3456721098	Male	Cardiologist
DR1006	Sofia	Wilson	567890123456	Female	Orthopedic
DR1015	Daniel	Perez	5678943210	Male	Orthopedic

- Function to get total count of patients.

```

DELIMITER //
CREATE FUNCTION TotalPatients() RETURNS INT
Deterministic
BEGIN
    DECLARE total INT;
    SELECT COUNT(*) INTO
    total FROM Patients;RETURN
    total;
END;
//DELIMITER ;

```

```
SELECT TotalPatients() AS TotalNumberOfPatients;
```

The screenshot shows the MySQL Workbench interface with the following details:

- Left Panel (Schemas):** Shows the database structure with the **healthcare2** schema selected. Under **Tables**, there are **Doctors**, **Insurance**, **Invoice**, **Patients**, **Payment**, **Pharmacy**, and **Treatment**.
- Central Panel (Query Editor):** Displays the SQL code for creating the **TotalPatients()** function. The code is as follows:


```

1 DELIMITER //
2
3 • CREATE FUNCTION TotalPatients() RETURNS INT
4 deterministic
5 BEGIN
6     DECLARE total INT;
7     SELECT COUNT(*) INTO total FROM Patients;
8     RETURN total;
9 END;
10 //
11 DELIMITER ;
12
13 • SELECT TotalPatients() AS TotalNumberOfPatients;
14 
```
- Bottom Panel (Result Grid):** Shows the result of the query execution. The table has one row with the value **17**.
- Right Panel (Toolbars):** Includes buttons for Result Grid, Form Editor, Field Types, Query Stats, and Execution Plan.

We employed the Python Programming language to execute fundamental Create, Read, Update, and Delete (CRUD) operations. These operations entail adding, retrieving, modifying, and removing database records pertinent for the Database Organization project for "HEALTHCARE MANAGEMENT SYSTEM". We crafted a program named "Healthcare.py" to establish the Database and for the CRUD operations we are using the CRUD.py file. Within this database, we constructed various tables including Patients, Doctors, Insurance, Invoice, payment, pharmacy, treatment, we populated all tables with relevant data for the Healthcare Management System. Additionally, it provides functionality for computing aggregate functions, performing set operations, and implementing OLAP (Online Analytical Processing) operations.

1. CRUD OPERATIONS:

We have developed a robust program named "CRUD.py," functioning as a command line tool tailored for our "Healthcare" database. This tool seamlessly performs essential Create, Read, Update, and Delete (CRUD) operations. With meticulous attention to detail, the program is crafted to uphold code readability and maintainability. Throughout its structure, insightful comments are strategically integrated, enhancing comprehension of both its functionality and implementation details.

2. PROGRAM MENU:

a) Table Menu

The script presents a menu to the user, displaying options to perform CRUD operations on different tables in the database such as Patients, Doctors, Insurance, etc. Users can select a table and then choose an operation (Create, Read, Update, Delete) to perform on that table. Based on the user's choice, the script prompts for necessary input such as record data, record ID, column name, and new values for updating records.

```
[(base) akshithabedre@dhcp234 desktop % python CRUD.py

Healthcare Database Selected !!

Select a table to perform CRUD operations:
1. Patients
2. Doctors
3. Insurance
4. Invoice
5. Payment
6. Pharmacy
7. Treatment
8. Exit
Enter the number of the table: ]
```

b) Operations Menu

The operations menu within the Python script serves as a pivotal component, facilitating user interaction with the healthcare database through a command-line interface. Upon execution, users are prompted to select a specific table from a comprehensive list including Patients, Doctors, Insurance, and more, indicating the domain of their desired database operation. Following table selection, users are presented with a set of operation choices, enabling them to Create, Read, Update, or Delete records within the chosen table. For instance, users can seamlessly add new records to the database, retrieve existing records for review, modify specific record attributes, or remove outdated entries as needed. The menu meticulously guides users through each operation, prompting for essential inputs such as record data, IDs, or column names, and ensuring data integrity through validation checks.

```
Healthcare Database Selected !!  
  
Select a table to perform CRUD operations:  
1. Patients  
2. Doctors  
3. Insurance  
4. Invoice  
5. Payment  
6. Pharmacy  
7. Treatment  
8. Exit  
Enter the number of the table: 1  
  
Selected table: Patients  
  
Options:  
1. Create a record for the selected table.  
2. Read all records for the selected table.  
3. Update a record for the selected table.  
4. Delete a record for the selected table.  
5. Return to Main Menu.  
Enter your choice: 
```

Test Case 1: Create Operation

```
Select a table to perform CRUD operations:  
1. Patients  
2. Doctors  
3. Insurance  
4. Invoice  
5. Payment  
6. Pharmacy  
7. Treatment  
8. Exit  
Enter the number of the table: 2  
  
Selected table: Doctors  
  
Options:  
1. Create a record for the selected table.  
2. Read all records for the selected table.  
3. Update a record for the selected table.  
4. Delete a record for the selected table.  
5. Return to Main Menu.  
Enter your choice: 1  
Doctors  
Enter data in this order:  
DoctorsID, fname, lname, phno, gender, specialist  
Enter values (comma-separated):  
DR1016,Maheera,Khan,8265262891,Female,Cardiologist  
Record created successfully in the Doctors table.
```

```
Selected table: Doctors  
  
Options:  
1. Create a record for the selected table.  
2. Read all records for the selected table.  
3. Update a record for the selected table.  
4. Delete a record for the selected table.  
5. Return to Main Menu.  
Enter your choice: 2  
  
-----  
DoctorsID | fname | lname | phno | gender | specialist  
-----  
DR1001 | John | Doe | 1234567890 | Male | Cardiologist  
DR1002 | Jane | Smith | 2345678901 | Female | Neurologist  
DR1003 | Michael | Brown | 3456789012 | Male | Orthopedic  
DR1004 | Emily | Jones | 4567890123 | Female | Dermatologist  
DR1005 | William | Garcia | 5678901234 | Male | Pediatrician  
DR1006 | Olivia | Martinez | 6789012345 | Female | Gynecologist  
DR1007 | David | Rodriguez | 7890123456 | Male | Psychiatrist  
DR1008 | Sophia | Wilson | 8901234567 | Female | Ophthalmologist  
DR1009 | James | Anderson | 9012345678 | Male | Endocrinologist  
DR1010 | Isabella | Thomas | 0123456789 | Female | Anesthesiologist  
DR1011 | Ethan | Taylor | 1234509876 | Male | Radiologist  
DR1012 | Ava | Moore | 2345610987 | Female | Oncologist  
DR1013 | Matthew | Jackson | 3456721098 | Male | Cardiologist  
DR1014 | Sofia | Lee | 4567832109 | Female | Neurologist  
DR1015 | Daniel | Perez | 5678943210 | Male | Orthopedic  
DR1016 | Maheera | Khan | 8265262891 | Female | Cardiologist  
-----
```

● Test Case 2: Read Operation

```
Select a table to perform CRUD operations:  
1. Patients  
2. Doctors  
3. Insurance  
4. Invoice  
5. Payment  
6. Pharmacy  
7. Treatment  
8. Exit  
Enter the number of the table: 1  
  
Selected table: Patients  
  
Options:  
1. Create a record for the selected table.  
2. Read all records for the selected table.  
3. Update a record for the selected table.  
4. Delete a record for the selected table.  
5. Return to Main Menu.  
Enter your choice: 2  
  
-----  
PatientsID | DoctorsID | PharmacyID | InsuranceID | fname | lname | dob | age | streetno | city | building  
-----  
PT2016 | DR1001 | MD53692891 | AAA2202137 | Alice | Wong | 1990-01-01 | 34 | 123 | New York | Liberty Towers  
PT2017 | DR1002 | MD89869879 | AAA0071404 | Bob | Smith | 1985-02-12 | 39 | 456 | Los Angeles | Sunset Plaza  
PT2018 | DR1003 | MD77368580 | AAA8458776 | Charlie | Johnson | 1978-03-23 | 46 | 789 | Chicago | Lake House  
PT2019 | DR1004 | MD47609337 | AAA1370939 | Daisy | Lee | 1992-04-14 | 32 | 1011 | Houston | Pine Apartments  
PT2020 | DR1005 | MD41093215 | AAA3071239 | Ethan | Brown | 1980-05-25 | 44 | 1213 | Phoenix | Desert Springs  
PT2021 | DR1006 | MD83218141 | AAA3641070 | Fiona | Davis | 1994-06-06 | 30 | 1415 | Philadelphia | River View  
PT2022 | DR1007 | MD93711778 | AAA6183734 | George | Miller | 1982-07-17 | 42 | 1617 | San Antonio | Oak Villas  
PT2023 | DR1008 | MD18868736 | AAA2786040 | Hannah | Wilson | 1975-08-28 | 49 | 1819 | San Diego | Marina Side  
PT2024 | DR1009 | MD93487613 | AAA5855663 | Ivan | Moore | 1988-09-09 | 36 | 2021 | Dallas | Highland Loft  
PT2025 | DR1010 | MD27465264 | AAA3373097 | Julia | Taylor | 1991-10-10 | 33 | 2223 | San Jose | Central Station  
PT2026 | DR1011 | MD16189598 | AAA3970248 | Kevin | Anderson | 1983-11-11 | 41 | 2425 | Austin | Greenwood Residences  
PT2027 | DR1012 | MD12194733 | AAA8289774 | Lily | Thomas | 1979-12-12 | 45 | 2627 | Jacksonville | Beachfront Homes  
PT2028 | DR1013 | MD45485169 | AAA3098878 | Mason | Jackson | 1986-01-13 | 38 | 2829 | San Francisco | Golden Gates  
PT2029 | DR1014 | MD27787881 | AAA6895621 | Nora | Martinez | 1993-02-14 | 31 | 3031 | Indianapolis | Maple Street  
PT2030 | DR1015 | MD22587921 | AAA1920740 | Oscar | Hernandez | 1984-03-15 | 40 | 3233 | Columbus | River Park  
-----
```

• Test Case 3: Update Operation

```
Select a table to perform CRUD operations:
1. Patients
2. Doctors
3. Insurance
4. Invoice
5. Payment
6. Pharmacy
7. Treatment
8. Exit
Enter the number of the table: 3

Selected table: Insurance

Options:
1. Create a record for the selected table.
2. Read all records for the selected table.
3. Update a record for the selected table.
4. Delete a record for the selected table.
5. Return to Main Menu.
Enter your choice: 3
Enter the ID of the record you want to update in Insurance: AAA0071404
Insurance Table = ['InsuranceID', 'phno', 'cname']
Enter the name of the column you want to update from the above list: cname
Enter the new value for the cname column: Aetna
Aetna cname AAA0071404 Insurance
Record with ID AAA0071404 is successfully updated in the Insurance table.

Select a table to perform CRUD operations:
1. Patients
2. Doctors
3. Insurance
4. Invoice
5. Payment
6. Pharmacy
7. Treatment
8. Exit
Enter the number of the table: 3

Selected table: Insurance

Options:
1. Create a record for the selected table.
2. Read all records for the selected table.
3. Update a record for the selected table.
4. Delete a record for the selected table.
5. Return to Main Menu.
Enter your choice: 2

-----
InsuranceID | phno | cname
-----
AAA0071404 | 800-234-5678 | Aetna
AAA1370939 | 800-456-7890 | Family Health Protector
AAA1920740 | 800-567-8902 | Unity Health Plans
AAA2202137 | 800-123-4567 | HealthPlus Insurance
AAA2786040 | 800-890-1234 | Prime Health Coverage
AAA3071239 | 800-567-8901 | QuickInsure
AAA3098878 | 800-345-6780 | Global Health Assurance
AAA3373097 | 800-012-3456 | Trusted Health Partners
AAA3641070 | 800-678-9012 | WellnessGuard
AAA3970248 | 800-123-4568 | Nationwide MediCover
AAA5855663 | 800-901-2345 | FutureCare
AAA6183734 | 800-789-0123 | SecureLife Health
AAA6895621 | 800-456-7891 | SafeHealth Systems
AAA8289774 | 800-234-5679 | HealthFirst Insurance
AAA8458776 | 800-345-6789 | MediSecure
-----
```

- **Test Case 4: Delete Operation While executing this Testcase:**

```
Select a table to perform CRUD operations:  
1. Patients  
2. Doctors  
3. Insurance  
4. Invoice  
5. Payment  
6. Pharmacy  
7. Treatment  
8. Exit  
Enter the number of the table: 2  
  
Selected table: Doctors  
  
Options:  
1. Create a record for the selected table.  
2. Read all records for the selected table.  
3. Update a record for the selected table.  
4. Delete a record for the selected table.  
5. Return to Main Menu.  
Enter your choice: 4  
Enter the ID of the record you want to delete from Doctors: DR1016  
Record with ID DR1016 deleted from the Doctors table.  
  
Select a table to perform CRUD operations:  
1. Patients  
2. Doctors  
3. Insurance  
4. Invoice  
5. Payment  
6. Pharmacy  
7. Treatment  
8. Exit  
Enter the number of the table: 2  
  
Selected table: Doctors  
  
Options:  
1. Create a record for the selected table.  
2. Read all records for the selected table.  
3. Update a record for the selected table.  
4. Delete a record for the selected table.  
5. Return to Main Menu.  
Enter your choice: 2  
  
-----  
DoctorsID | fname | lname | phno | gender | specialist  
-----  
DR1001 | John | Doe | 1234567890 | Male | Cardiologist  
DR1002 | Jane | Smith | 2345678901 | Female | Neurologist  
DR1003 | Michael | Brown | 3456789012 | Male | Orthopedic  
DR1004 | Emily | Jones | 4567890123 | Female | Dermatologist  
DR1005 | William | Garcia | 5678901234 | Male | Pediatrician  
DR1006 | Olivia | Martinez | 6789012345 | Female | Gynecologist  
DR1007 | David | Rodriguez | 7890123456 | Male | Psychiatrist  
DR1008 | Sophia | Wilson | 8901234567 | Female | Ophthalmologist  
DR1009 | James | Anderson | 9012345678 | Male | Endocrinologist  
DR1010 | Isabella | Thomas | 0123456789 | Female | Anesthesiologist  
DR1011 | Ethan | Taylor | 1234509876 | Male | Radiologist  
DR1012 | Ava | Moore | 2345610987 | Female | Oncologist  
DR1013 | Matthew | Jackson | 3456721098 | Male | Cardiologist  
DR1014 | Sofia | Lee | 4567832109 | Female | Neurologist  
DR1015 | Daniel | Perez | 5678943210 | Male | Orthopedic
```

- **Test Case 5: Quit Option**

```
[(base) akshithabedre@dhcp234 desktop % python CRUD.py

Healthcare Database Selected !!

Select a table to perform CRUD operations:
1. Patients
2. Doctors
3. Insurance
4. Invoice
5. Payment
6. Pharmacy
7. Treatment
8. Exit
Enter the number of the table: 8
Exited from Healthcare Database
(base) akshithabedre@dhcp234 desktop % ]
```

3. Error Handling

We've developed code to handle errors and display concise error messages during the execution of test cases in the program it signifies that the development team has implemented specific programming instructions to effectively manage errors encountered during testing. This code not only detects errors but also generates clear and succinct error messages to provide valuable feedback to users or developers. By prioritizing error handling during the testing phase, the software ensures robustness and reliability, facilitating efficient debugging and enhancing the overall quality of the program. Here we are showing different error handling test cases and how our program lets the user to handle edge cases with clear instructions.

A) Invalid Table number:

When a user inputs a table number that is invalid, consisting of a combination of integers and alphabets, the system will generate a specific error message. This message will provide clear guidance on how to address the issue and proceed correctly. If the user's input does not adhere to our specified data entry format, an error is triggered.

```
Select a table to perform CRUD operations:
```

- 1. Patients
- 2. Doctors
- 3. Insurance
- 4. Invoice
- 5. Payment
- 6. Pharmacy
- 7. Treatment
- 8. Exit

```
Enter the number of the table: 22
```

```
Invalid table choice
```

```
Enter Again from 1 to 7
```

```
Select a table to perform CRUD operations:
```

- 1. Patients
- 2. Doctors
- 3. Insurance
- 4. Invoice
- 5. Payment
- 6. Pharmacy
- 7. Treatment
- 8. Exit

```
Enter the number of the table: █
```

B) Invalid Operation Number:

If a user selects an operation number that is invalid, meaning it doesn't correspond to any recognized operation within the program, the system ensures clarity by offering specific instructions. These instructions are designed to guide the user on how to handle this edge case scenario effectively. Essentially, the program acknowledges the possibility of encountering invalid operation numbers and proactively provides guidance to users, helping them understand what went wrong and how to proceed correctly. This proactive approach enhances user experience and reduces confusion in navigating the system.

```
[  
  Healthcare Database Selected !!  
  
  Select a table to perform CRUD operations:  
  1. Patients  
  2. Doctors  
  3. Insurance  
  4. Invoice  
  5. Payment  
  6. Pharmacy  
  7. Treatment  
  8. Exit  
  Enter the number of the table: 2  
  
  Selected table: Doctors  
  
  Options:  
  1. Create a record for the selected table.  
  2. Read all records for the selected table.  
  3. Update a record for the selected table.  
  4. Delete a record for the selected table.  
  5. Return to Main Menu.  
  Enter your choice: 21  
  Invalid Operation Choice.  
  Please enter choice from 1 to 4  
  
  Options:  
  1. Create a record for the selected table.  
  2. Read all records for the selected table.  
  3. Update a record for the selected table.  
  4. Delete a record for the selected table.  
  5. Return to Main Menu.  
  Enter your choice: █
```

C)Error in data for Create Operation:

In the event of an error during the creation operation, particularly when incorrect or incomplete data is provided, the program responds by recognizing the discrepancy and prompts the user to re-enter the necessary data. For instance, when attempting to create a record in the Hospital Table, if the user inputs partial or inaccurate information, the program detects this inconsistency and intervenes by requesting the user to input the data again accurately. This proactive behavior ensures data integrity and completeness within the system, ultimately enhancing the reliability and effectiveness of the program's operations.

```
Select a table to perform CRUD operations:
1. Patients
2. Doctors
3. Insurance
4. Invoice
5. Payment
6. Pharmacy
7. Treatment
8. Exit
Enter the number of the table: 6

Selected table: Pharmacy

Options:
1. Create a record for the selected table.
2. Read all records for the selected table.
3. Update a record for the selected table.
4. Delete a record for the selected table.
5. Return to Main Menu.
Enter your choice: 1
Pharmacy
Enter data in this order:
PharmacyID, price, quantity, mname
Enter values (comma-separated):
MD93816384,117F.5,18,Aspirin
1366 (HY000): Incorrect decimal value: '117F.5' for column 'price' at row 1

Select a table to perform CRUD operations:
1. Patients
2. Doctors
3. Insurance
4. Invoice
5. Payment
6. Pharmacy
7. Treatment
8. Exit
Enter the number of the table: █
```

D) Error in Update Operation:

In the event of an error during the update operation, specifically when an invalid ID is provided, the program responds by detecting the discrepancy and presenting it as an error message to the user. For instance, when attempting to update the Patients Table and the user inputs an invalid ID, the program recognizes this inconsistency and prompts the user to correct the input by displaying an error message. This proactive approach ensures data accuracy and integrity within the system, guiding users to input valid information and enhancing the overall reliability of the program's update functionality.

```
Select a table to perform CRUD operations:  
1. Patients  
2. Doctors  
3. Insurance  
4. Invoice  
5. Payment  
6. Pharmacy  
7. Treatment  
8. Exit  
Enter the number of the table: 1  
  
Selected table: Patients  
  
Options:  
1. Create a record for the selected table.  
2. Read all records for the selected table.  
3. Update a record for the selected table.  
4. Delete a record for the selected table.  
5. Return to Main Menu.  
Enter your choice: 3  
Enter the ID of the record you want to update in Patients: 12  
Record with ID 12 does not exist in the Patients table. Please enter a valid ID from Patients.  
Enter the ID of the record you want to update in Patients: 
```

```
Select a table to perform CRUD operations:
```

- 1. Patients
- 2. Doctors
- 3. Insurance
- 4. Invoice
- 5. Payment
- 6. Pharmacy
- 7. Treatment
- 8. Exit

```
Enter the number of the table: 3
```

```
Selected table: Insurance
```

```
Options:
```

- 1. Create a record for the selected table.
- 2. Read all records for the selected table.
- 3. Update a record for the selected table.
- 4. Delete a record for the selected table.
- 5. Return to Main Menu.

```
Enter your choice: 3
```

```
Enter the ID of the record you want to update in Insurance: AAA007404
```

```
Record with ID AAA007404 does not exist in the Insurance table. Please enter a valid ID from Insurance.
```

```
Enter the ID of the record you want to update in Insurance: AAA0071404
```

```
Insurance Table = ['InsuranceID', 'phno', 'cname']
```

```
Enter the name of the column you want to update from the above list: phno
```

```
Enter the new value for the phno column: 8127193214
```

```
8127193214 phno AAA0071404 Insurance
```

```
Record with ID AAA0071404 is successfully updated in the Insurance table.
```

```
Select a table to perform CRUD operations:
```

- 1. Patients
- 2. Doctors
- 3. Insurance
- 4. Invoice
- 5. Payment
- 6. Pharmacy
- 7. Treatment
- 8. Exit

```
Enter the number of the table: 
```

E) Error in Delete Operation

When attempting to delete a record from the "Hospital" table with an invalid ID, such as a random ID that does not correspond to any existing record, the system correctly responded with an error message indicating that the record was not found in the table. This error message is a safeguard against accidental deletion of data, ensuring that only valid and existing records can be removed, thus maintaining data integrity within the database, and providing clear feedback to users or developers about the issue at hand.

```
Select a table to perform CRUD operations:
1. Patients
2. Doctors
3. Insurance
4. Invoice
5. Payment
6. Pharmacy
7. Treatment
8. Exit
Enter the number of the table: 2

Selected table: Doctors

Options:
1. Create a record for the selected table.
2. Read all records for the selected table.
3. Update a record for the selected table.
4. Delete a record for the selected table.
5. Return to Main Menu.
Enter your choice: 4
Enter the ID of the record you want to delete from Doctors: DR1011
Record with ID DR1011 deleted from the Doctors table.

Select a table to perform CRUD operations:
1. Patients
2. Doctors
3. Insurance
4. Invoice
5. Payment
6. Pharmacy
7. Treatment
8. Exit
Enter the number of the table: 2

Selected table: Doctors

Options:
1. Create a record for the selected table.
2. Read all records for the selected table.
3. Update a record for the selected table.
4. Delete a record for the selected table.
5. Return to Main Menu.
Enter your choice: 2

-----
DoctorsID | fname | lname | phno | gender | specialist
-----
DR1001 | John | Doe | 1234567890 | Male | Cardiologist
DR1002 | Jane | Smith | 2345678901 | Female | Neurologist
DR1003 | Michael | Brown | 3456789012 | Male | Orthopedic
DR1004 | Emily | Jones | 4567890123 | Female | Dermatologist
DR1005 | William | Garcia | 5678901234 | Male | Pediatrician
DR1006 | Olivia | Martinez | 6789012345 | Female | Gynecologist
DR1007 | David | Rodriguez | 7890123456 | Male | Psychiatrist
DR1008 | Sophia | Wilson | 8901234567 | Female | Ophthalmologist
DR1009 | James | Anderson | 9012345678 | Male | Endocrinologist
DR1010 | Isabella | Thomas | 0123456789 | Female | Anesthesiologist
DR1012 | Ava | Moore | 2345610987 | Female | Oncologist
DR1013 | Matthew | Jackson | 3456721098 | Male | Cardiologist
DR1014 | Sofia | Lee | 4567832109 | Female | Neurologist
DR1015 | Daniel | Perez | 5678943210 | Male | Orthopedic
```

Demonstration (Loom Video)

We are providing high-quality Loom video recording that clearly explains the CRUD operations, highlights essential features, and showcases the system's functionality through various scenarios, all in a well-structured and engaging manner. This approach allows viewers to grasp the system's usage, observe its testing, and understand how it operates in different situations. Not only does this educate users on effective system utilization, but it also serves as a compelling display of the system's capabilities. Additionally, We are using functionality for computing aggregate functions, performing set operations, and implementing OLAP (Online Analytical Processing) operations.

Function Definitions:

CRUD Operations: Functions like `create_record`, `read_records`, `update_record`, and `delete_record` are defined to perform CRUD operations on the tables.

Aggregate Functions: Functions like `get_average_value`, `get_min_value`, `get_max_value`, and `get_unique_values` are defined to compute aggregate statistics on columns.

Set Operations: Functions like `Union_record` and `Intersect_record` are defined to perform set union and intersection operations between two tables.

OLAP Operations: Functions like `ntile_table`, `cumulative_distribution`, `rollup_data`, and `cube_simulation` are defined to perform OLAP operations such as NTILE, cumulative distribution, rollup, and cube.

Average(invoice):

```
waseempashamohammad@Waseems-MacBook-Pro Group assignment % python3 TD6.py
Healthcare database selected

Select a table to perform the operations:
1. Patients
2. Doctors
3. Insurance
4. Invoice
5. Payment
6. Pharmacy
7. Treatment
8. Exit
Enter the number of the table: 4
Selected table: Invoice

Options:
1. Create a record for the selected table
2. Read all records for selected table
3. Update a record for selected table
4. Delete a record for selected table
5. Average value of a column
6. Count appearances in a column
7. Minimum value of a column
8. Maximum value of a column
9. Union Operation
10. Intersect Operation
11. Ntile
12. Cumulative Distribution
13. Cube
14. Rollup
15. Return to Main Menu
Enter your choice: 5
['InvoiceID', 'PaymentID', 'PatientsID', 'invoicedate', 'amount']
Enter the name of the column to compute the average value: amount
The average value in amount of Invoice is: 535.666667
```

Count(doctors):

```
Select a table to perform the operations:
1. Patients
2. Doctors
3. Insurance
4. Invoice
5. Payment
6. Pharmacy
7. Treatment
8. Exit
Enter the number of the table: 2
Selected table: Doctors

Options:
1. Create a record for the selected table
2. Read all records for selected table
3. Update a record for selected table
4. Delete a record for selected table
5. Average value of a column
6. Count appearances in a column
7. Minimum value of a column
8. Maximum value of a column
9. Union Operation
10. Intersect Operation
11. Ntile
12. Cumulative Distribution
13. Cube
14. Rollup
15. Return to Main Menu
Enter your choice: 6
['DoctorsID', 'fname', 'lname', 'phno', 'gender', 'specialist']
Enter the name of the column to count records: gender
Enter the value to count in the gender column: male
Count of records with the value 'male' in gender: 8
```

Minimum(invoice):

```
Select a table to perform the operations:
```

- 1. Patients
- 2. Doctors
- 3. Insurance
- 4. Invoice
- 5. Payment
- 6. Pharmacy
- 7. Treatment
- 8. Exit

```
Enter the number of the table: 4
```

```
Selected table: Invoice
```

```
Options:
```

- 1. Create a record for the selected table
- 2. Read all records for selected table
- 3. Update a record for selected table
- 4. Delete a record for selected table
- 5. Average value of a column
- 6. Count appearances in a column
- 7. Minimun value of a column
- 8. Maximum value of a column
- 9. Union Operation
- 10. Intersect Operation
- 11. NTile
- 12. Cumulative Distribution
- 13. Cube
- 14. Rollup
- 15. Return to Main Menu

```
Enter your choice: 7
```

```
['InvoiceID', 'PaymentID', 'PatientsID', 'invoicedate', 'amount']
```

```
Enter the name of the column to compute the minimum value: amount
```

```
The minimum value in amount of Invoice is: 115.00
```

Maximum(invoice):

```
Select a table to perform the operations:  
1. Patients  
2. Doctors  
3. Insurance  
4. Invoice  
5. Payment  
6. Pharmacy  
7. Treatment  
8. Exit  
Enter the number of the table: 4  
Selected table: Invoice  
  
Options:  
1. Create a record for the selected table  
2. Read all records for selected table  
3. Update a record for selected table  
4. Delete a record for selected table  
5. Average value of a column  
6. Count appearances in a column  
7. Minimum value of a column  
8. Maximum value of a column  
9. Union Operation  
10. Intersect Operation  
11. NTile  
12. Cumulative Distribution  
13. Cube  
14. Rollup  
15. Return to Main Menu  
Enter your choice: 8  
['InvoiceID', 'PaymentID', 'PatientsID', 'invoicedate', 'amount']  
Enter the name of the column to compute the maximum value: amount  
The maximum value in amount of Invoice is: 995.00
```

UNION OPERATORS (pharmacy and payment):

```
Select a table to perform the operations:
```

- 1. Patients
- 2. Doctors
- 3. Insurance
- 4. Invoice
- 5. Payment
- 6. Pharmacy
- 7. Treatment
- 8. Exit

```
Enter the number of the table: 5
```

```
Selected table: Payment
```

```
Options:
```

- 1. Create a record for the selected table
- 2. Read all records for selected table
- 3. Update a record for selected table
- 4. Delete a record for selected table
- 5. Average value of a column
- 6. Count appearances in a column
- 7. Minimum value of a column
- 8. Maximum value of a column
- 9. Union Operation
- 10. Intersect Operation
- 11. NTile
- 12. Cumulative Distribution
- 13. Cube
- 14. Rollup
- 15. Return to Main Menu

```
Enter your choice: 9
```

- 1. Patients
- 2. Doctors
- 3. Insurance
- 4. Invoice
- 5. Payment
- 6. Pharmacy
- 7. Treatment

```
Enter the second table to execute Union operation: 6
```

```
('PPP14926', 'Completed', 'Bank Transfer', '2024-06-10')
('PPP21549', 'Completed', 'Cash', '2024-03-25')
('PPP31748', 'Pending', 'Credit Card', '2025-01-13')
('PPP38741', 'Completed', 'Credit Card', '2024-04-30')
('PPP45981', 'Completed', 'Bank Transfer', '2025-03-24')
('PPP46328', 'Completed', 'Credit Card', '2024-10-30')
('PPP53879', 'Completed', 'Credit Card', '2024-07-15')
('PPP57831', 'Completed', 'Cash', '2024-11-04')
('PPP62487', 'Pending', 'Bank Transfer', '2024-02-20')
('PPP68412', 'Completed', 'Cash', '2025-02-17')
('PPP69214', 'Completed', 'Cash', '2024-08-20')
('PPP76492', 'Pending', 'Cash', '2024-05-05')
('PPP83157', 'Pending', 'Bank Transfer', '2024-09-25')
('PPP92674', 'Completed', 'Bank Transfer', '2024-12-09')
('PPP97532', 'Completed', 'Credit Card', '2024-01-15')
('MD12194733', '110.00', '15', 'Atorvastatin')
('MD16189598', '95.00', '20', 'Levthyroxine')
('MD18868736', '45.00', '90', 'Simvastatin')
('MD22587921', '140.00', '8', 'Hydrochlorothiazide')
('MD27465264', '80.00', '25', 'Lisinopril')
('MD27787881', '130.00', '5', 'Azithromycin')
('MD41093215', '15.00', '150', 'Aspirin')
('MD45485169', '120.00', '10', 'Omeprazole')
('MD47609337', '70.00', '30', 'Ciprofloxacin')
('MD53692891', '20.00', '50', 'Paracetamol')
('MD77368580', '50.00', '75', 'Amoxicillin')
('MD83218141', '25.00', '200', 'Metformin')
('MD89869879', '35.00', '100', 'Ibuprofen')
('MD93487613', '65.00', '40', 'Losartan')
('MD93711778', '55.00', '60', 'Amlodipine')
```

Intersect (patients and doctor):

```
Select a table to perform the operations:  
1. Patients  
2. Doctors  
3. Insurance  
4. Invoice  
5. Payment  
6. Pharmacy  
7. Treatment  
8. Exit  
Enter the number of the table: 1  
Selected table: Patients  
  
Options:  
1. Create a record for the selected table  
2. Read all records for selected table  
3. Update a record for selected table  
4. Delete a record for selected table  
5. Average value of a column  
6. Count appearances in a column  
7. Minimum value of a column  
8. Maximum value of a column  
9. Union Operation  
10. Intersect Operation  
11. Ntile  
12. Cumulative Distribution  
13. Cube  
14. Rollup  
15. Return to Main Menu  
Enter your choice: 10  
1. Patients  
2. Doctors  
3. Insurance  
4. Invoice  
5. Payment  
6. Pharmacy  
7. Treatment  
Enter the second table to execute Intersect operation: 2  
  
Common column is: DoctorsID  
Intersection of two tables:  
( 'DR1001', 'John', 'Doe', '1234567890', 'Male', 'Cardiologist' )  
( 'DR1002', 'Jane', 'Smith', '2345678901', 'Female', 'Neurologist' )  
( 'DR1003', 'Michael', 'Brown', '3456789012', 'Male', 'Orthopedic' )  
( 'DR1004', 'Emily', 'Jones', '4567890123', 'Female', 'Dermatologist' )  
( 'DR1005', 'William', 'Garcia', '5678901234', 'Male', 'Pediatrician' )  
( 'DR1006', 'Olivia', 'Martinez', '6789012345', 'Female', 'Gynecologist' )  
( 'DR1007', 'David', 'Rodriguez', '7890123456', 'Male', 'Psychiatrist' )  
( 'DR1008', 'Sophia', 'Wilson', '8901234567', 'Female', 'Ophthalmologist' )  
( 'DR1009', 'James', 'Anderson', '9012345678', 'Male', 'Endocrinologist' )  
( 'DR1010', 'Isabella', 'Thomas', '0123456789', 'Female', 'Anesthesiologist' )  
( 'DR1011', 'Ethan', 'Taylor', '1234509876', 'Male', 'Radiologist' )  
( 'DR1012', 'Ava', 'Moore', '2345610987', 'Female', 'Oncologist' )  
( 'DR1013', 'Matthew', 'Jackson', '3456721098', 'Male', 'Cardiologist' )  
( 'DR1014', 'Sofia', 'Lee', '4567832109', 'Female', 'Neurologist' )  
( 'DR1015', 'Daniel', 'Perez', '5678943210', 'Male', 'Orthopedic' )
```

NTILE OPERATION:

```
Select a table to perform the operations:
```

- 1. Patients
- 2. Doctors
- 3. Insurance
- 4. Invoice
- 5. Payment
- 6. Pharmacy
- 7. Treatment
- 8. Exit

```
Enter the number of the table: 4
```

```
Selected table: Invoice
```

```
Options:
```

- 1. Create a record for the selected table
- 2. Read all records for selected table
- 3. Update a record for selected table
- 4. Delete a record for selected table
- 5. Average value of a column
- 6. Count appearances in a column
- 7. Minimum value of a column
- 8. Maximum value of a column
- 9. Union Operation
- 10. Intersect Operation
- 11. NTile
- 12. Cumulative Distribution
- 13. Cube
- 14. Rollup
- 15. Return to Main Menu

```
Enter your choice: 11
```

```
NTILE Operation Calculation
```

```
['InvoiceID', 'PaymentID', 'PatientsID', 'invoicedate', 'amount']
```

```
Enter the column name to partition: amount
```

```
Enter the number of buckets to divide into: 5
```

```
Enter the column name to partition: amount
```

```
Enter the number of buckets to divide into: 5
```

```
Data in Invoice divided into 5 buckets based on the 'amount' column:
```

```
Value: 115.00, Bucket: 1
```

```
Value: 170.00, Bucket: 1
```

```
Value: 225.00, Bucket: 1
```

```
Value: 305.00, Bucket: 2
```

```
Value: 335.00, Bucket: 2
```

```
Value: 395.00, Bucket: 2
```

```
Value: 455.00, Bucket: 3
```

```
Value: 515.00, Bucket: 3
```

```
Value: 575.00, Bucket: 3
```

```
Value: 645.00, Bucket: 4
```

```
Value: 680.00, Bucket: 4
```

```
Value: 815.00, Bucket: 4
```

```
Value: 875.00, Bucket: 5
```

```
Value: 935.00, Bucket: 5
```

```
Value: 995.00, Bucket: 5
```

Cumulative Distribution (Invoice):

```
Select a table to perform the operations:  
1. Patients  
2. Doctors  
3. Insurance  
4. Invoice  
5. Payment  
6. Pharmacy  
7. Treatment  
8. Exit  
Enter the number of the table: 4  
Selected table: Invoice  
  
Options:  
1. Create a record for the selected table  
2. Read all records for selected table  
3. Update a record for selected table  
4. Delete a record for selected table  
5. Average value of a column  
6. Count appearances in a column  
7. Minimum value of a column  
8. Maximum value of a column  
9. Union Operation  
10. Intersect Operation  
11. NTile  
12. Cumulative Distribution  
13. Cube  
14. Rollup  
15. Return to Main Menu  
Enter your choice: 12  
  
Cumulative Distribution Calculation  
['InvoiceID', 'PaymentID', 'PatientsID', 'invoicedate', 'amount']  
Enter the column name for which to calculate the cumulative distribution: amount  
Cumulative distribution for amount in Invoice:  
Value: 115.00, Cumulative Distribution: 0.0143  
Value: 170.00, Cumulative Distribution: 0.0355  
Value: 225.00, Cumulative Distribution: 0.0635  
Value: 305.00, Cumulative Distribution: 0.1014  
Value: 335.00, Cumulative Distribution: 0.1431  
Value: 395.00, Cumulative Distribution: 0.1923  
Value: 455.00, Cumulative Distribution: 0.2489  
Value: 515.00, Cumulative Distribution: 0.3130  
Value: 575.00, Cumulative Distribution: 0.3846  
Value: 645.00, Cumulative Distribution: 0.4648  
Value: 680.00, Cumulative Distribution: 0.5495  
Value: 815.00, Cumulative Distribution: 0.6509  
Value: 875.00, Cumulative Distribution: 0.7598  
Value: 935.00, Cumulative Distribution: 0.8762  
Value: 995.00, Cumulative Distribution: 1.0000
```

CUBE (invoice):

```
Select a table to perform the operations:  
1. Patients  
2. Doctors  
3. Insurance  
4. Invoice  
5. Payment  
6. Pharmacy  
7. Treatment  
8. Exit  
Enter the number of the table: 4  
Selected table: Invoice  
  
Options:  
1. Create a record for the selected table  
2. Read all records for selected table  
3. Update a record for selected table  
4. Delete a record for selected table  
5. Average value of a column  
6. Count appearances in a column  
7. Minimum value of a column  
8. Maximum value of a column  
9. Union Operation  
10. Intersect Operation  
11. Ntile  
12. Cumulative Distribution  
13. Cube  
14. Rollup  
15. Return to Main Menu  
Enter your choice: 13
```

```
['InvoiceID', 'PaymentID', 'PatientID', 'invoicedate', 'amount']  
Enter the first dimension for CUBE: amount  
Enter the second dimension for CUBE: invoicedate  
CUBE results for Invoice based on amount and invoicedate:  
(Decimal('575.00'), datetime.date(2024, 4, 25), 1, Decimal('575.00'))  
(Decimal('170.00'), datetime.date(2024, 1, 10), 1, Decimal('170.00'))  
(Decimal('115.00'), datetime.date(2024, 5, 30), 1, Decimal('115.00'))  
(Decimal('225.00'), datetime.date(2024, 6, 4), 1, Decimal('225.00'))  
(Decimal('455.00'), datetime.date(2024, 7, 9), 1, Decimal('455.00'))  
(Decimal('395.00'), datetime.date(2024, 8, 14), 1, Decimal('395.00'))  
(Decimal('515.00'), datetime.date(2024, 9, 19), 1, Decimal('515.00'))  
(Decimal('680.00'), datetime.date(2024, 10, 24), 1, Decimal('680.00'))  
(Decimal('645.00'), datetime.date(2024, 11, 28), 1, Decimal('645.00'))  
(Decimal('815.00'), datetime.date(2024, 12, 3), 1, Decimal('815.00'))  
(Decimal('875.00'), datetime.date(2025, 1, 7), 1, Decimal('875.00'))  
(Decimal('935.00'), datetime.date(2025, 2, 11), 1, Decimal('935.00'))  
(Decimal('995.00'), datetime.date(2025, 3, 18), 1, Decimal('995.00'))  
(Decimal('335.00'), datetime.date(2024, 2, 15), 1, Decimal('335.00'))  
(Decimal('305.00'), datetime.date(2024, 3, 20), 1, Decimal('305.00'))  
(Decimal('575.00'), None, 1, Decimal('575.00'))  
(Decimal('170.00'), None, 1, Decimal('170.00'))  
(Decimal('115.00'), None, 1, Decimal('115.00'))  
(Decimal('225.00'), None, 1, Decimal('225.00'))  
(Decimal('455.00'), None, 1, Decimal('455.00'))  
(Decimal('395.00'), None, 1, Decimal('395.00'))  
(Decimal('515.00'), None, 1, Decimal('515.00'))  
(Decimal('680.00'), None, 1, Decimal('680.00'))  
(Decimal('645.00'), None, 1, Decimal('645.00'))  
(Decimal('815.00'), None, 1, Decimal('815.00'))  
(Decimal('875.00'), None, 1, Decimal('875.00'))  
(Decimal('935.00'), None, 1, Decimal('935.00'))  
(Decimal('995.00'), None, 1, Decimal('995.00'))  
(Decimal('335.00'), None, 1, Decimal('335.00'))  
(Decimal('305.00'), None, 1, Decimal('305.00'))  
(None, datetime.date(2024, 4, 25), 1, Decimal('575.00'))  
(None, datetime.date(2024, 1, 10), 1, Decimal('170.00'))  
(None, datetime.date(2024, 5, 30), 1, Decimal('115.00'))  
(None, datetime.date(2024, 6, 4), 1, Decimal('225.00'))  
(None, datetime.date(2024, 7, 9), 1, Decimal('455.00'))  
(None, datetime.date(2024, 8, 14), 1, Decimal('395.00'))  
(None, datetime.date(2024, 9, 19), 1, Decimal('515.00'))  
(None, datetime.date(2024, 10, 24), 1, Decimal('680.00'))  
(None, datetime.date(2024, 11, 28), 1, Decimal('645.00'))  
(None, datetime.date(2024, 12, 3), 1, Decimal('815.00'))  
(None, datetime.date(2025, 1, 7), 1, Decimal('875.00'))  
(None, datetime.date(2025, 2, 11), 1, Decimal('935.00'))  
(None, datetime.date(2025, 3, 18), 1, Decimal('995.00'))  
(None, datetime.date(2024, 2, 15), 1, Decimal('335.00'))  
(None, datetime.date(2024, 3, 20), 1, Decimal('305.00'))  
(None, None, 15, Decimal('8035.00'))
```

14)ROLLUP (invoice)

```
Select a table to perform the operations:  
1. Patients  
2. Doctors  
3. Insurance  
4. Invoice  
5. Payment  
6. Pharmacy  
7. Treatment  
8. Exit  
Enter the number of the table: 4  
Selected table: Invoice  
  
Options:  
1. Create a record for the selected table  
2. Read all records for selected table  
3. Update a record for selected table  
4. Delete a record for selected table  
5. Average value of a column  
6. Count appearances in a column  
7. Minimum value of a column  
8. Maximum value of a column  
9. Union Operation  
10. Intersect Operation  
11. Ntile  
12. Cumulative Distribution  
13. Cube  
14. Rollup  
15. Return to Main Menu  
Enter your choice: 14  
  
Performing ROLLUP Operation  
['InvoiceID', 'PaymentID', 'PatientsID', 'invoicedate', 'amount']  
Enter the column names to group by (comma-separated): amount  
ROLLUP results for Invoice on amount:  
(Decimal('115.00'), 1, Decimal('115.00'))  
(Decimal('170.00'), 1, Decimal('170.00'))  
(Decimal('225.00'), 1, Decimal('225.00'))  
(Decimal('305.00'), 1, Decimal('305.00'))  
(Decimal('335.00'), 1, Decimal('335.00'))  
(Decimal('395.00'), 1, Decimal('395.00'))  
(Decimal('455.00'), 1, Decimal('455.00'))  
(Decimal('515.00'), 1, Decimal('515.00'))  
(Decimal('575.00'), 1, Decimal('575.00'))  
(Decimal('645.00'), 1, Decimal('645.00'))  
(Decimal('680.00'), 1, Decimal('680.00'))  
(Decimal('815.00'), 1, Decimal('815.00'))  
(Decimal('875.00'), 1, Decimal('875.00'))  
(Decimal('935.00'), 1, Decimal('935.00'))  
(Decimal('995.00'), 1, Decimal('995.00'))  
(None, 15, Decimal('8035.00'))
```