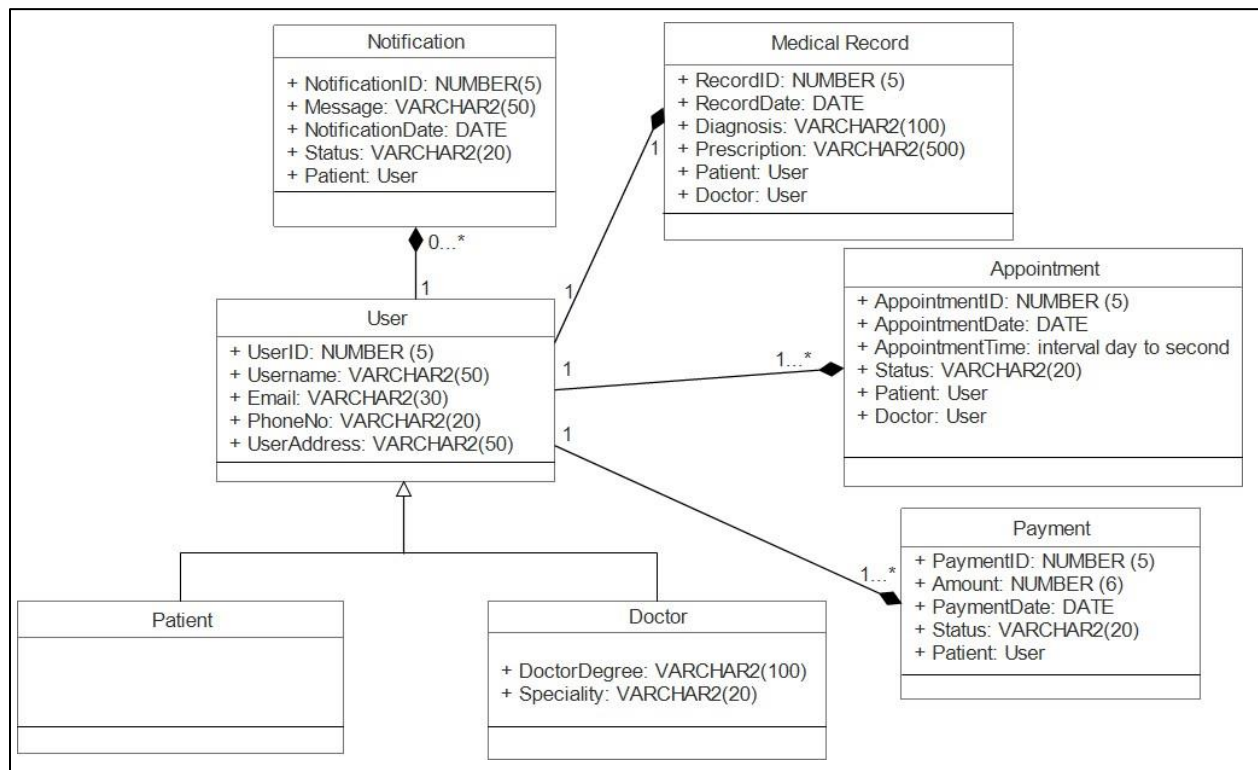


## Telemedicine Online Platform Database Design

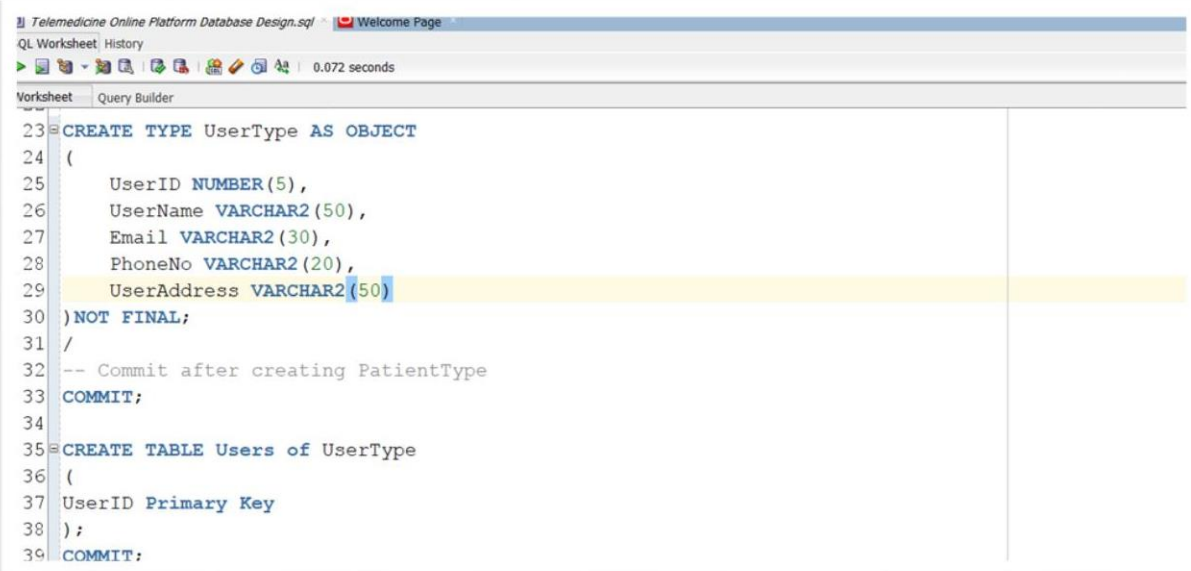
### Task one:



## Task two (Oracle SQL code screenshots and output):

### 1. User Table

#### Code of User Table:



```
23 CREATE TYPE UserType AS OBJECT
24 (
25     UserID NUMBER(5),
26     UserName VARCHAR2(50),
27     Email VARCHAR2(30),
28     PhoneNo VARCHAR2(20),
29     UserAddress VARCHAR2(50)
30 ) NOT FINAL;
31 /
32 -- Commit after creating PatientType
33 COMMIT;
34
35 CREATE TABLE Users OF UserType
36 (
37     UserID PRIMARY KEY
38 );
39 COMMIT;
```

#### Script Output:



```
Script Output x
Task completed in 0.161 seconds

Type USERTYPE compiled

Commit complete.

Table USERS created.

Commit complete.
```

## Code of Inheritances of User Table:

Inheritance: PatientType

Inheritance: UserType

```

41 -- Create PatientType (Inheritance)
42 CREATE TYPE PatientType UNDER UserType
43 ();
44 /
45
46
47 -- Create DoctorType (Inheritance)
48 CREATE TYPE DoctorType UNDER UserType
49 (   DoctorDegree VARCHAR2(100),
50     Speciality VARCHAR2(20)
51 );
52 /
53 COMMIT;

```

## Script Output:

```

Script Output x
Task completed in 0.228 seconds

Type PATIENTTYPE compiled

Type DOCTORTYPE compiled

Commit complete.

```

## Code of Insert Data of User Table:

```

INSERT INTO Users VALUES (DoctorType(1, 'Dr. Thu', 'thutal1@gmail.com', '0943055941', 'Yadanarbon St, Kyi Myint Tine', 'Hlaing'));
INSERT INTO Users VALUES (DoctorType(2, 'Dr. Kaung', 'kaung97@gmail.com', '09788100633', 'Thuzitar 6th St, Northoakkala', 'Hlaing'));
INSERT INTO Users VALUES (DoctorType(3, 'Dr. Chit Thway', 'chitthway76@gmail.com', '09457777554', 'Baho St, Sanchaung', 'Hlaing'));
INSERT INTO Users VALUES (DoctorType(4, 'Dr. Moe Htet', 'moehtet14@gmail.com', '09799499049', 'Damathukha Kyaung st, Hlaing'));

INSERT INTO Users VALUES (PatientType(101, 'Soe Moe', 'soemoe13@gmail.com', '09793470122', 'Thuta St, South Oakkalapa'));
INSERT INTO Users VALUES (PatientType(102, 'Minkhant', 'minkhant65@gmail.com', '09791770513', 'Bahan 3th St, Bahan'));
INSERT INTO Users VALUES (PatientType(103, 'Phone Pyae', 'phonepyae43@gmail.com', '09759032315', 'Phyar Pone St, Sanchaung'));
INSERT INTO Users VALUES (PatientType(104, 'Kyaw Min', 'kyawmin23@gmail.com', '09771000071', 'Min Dhama St, South Oakkalapa'));
INSERT INTO Users VALUES (PatientType(105, 'Pyae Phyo', 'pyaephyo19@gmail.com', '09967860651', 'Sabel St, Yankin'));
INSERT INTO Users VALUES (PatientType(106, 'Kyaw Swar', 'kyawswar07@gmail.com', '09259627475', 'Amayar St, North Oakkala'));
INSERT INTO Users VALUES (PatientType(107, 'Lin Thike', 'linthike33@gmail.com', '09420217641', 'May yu St, North Oakkala'));

COMMIT;

```

## Script Output of Insert Data of User Table:

```

Script Output x
Task completed in 0.167 seconds

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

Commit complete.

```

## Query Result of User Table:

USERID	USERNAME	EMAIL	PHONENO	USERADDRESS
1	Dr. Thu	thuta11@gmail.com	0943055941	Yadanarbon St, Kyi Myint Tine
2	Dr. Kaung	kaung97@gmail.com	09788100633	Thuzitar 6th St, Northoakkala
3	Dr. Chit Thway	chitthway76@gmail.com	09457777554	Baho St, Sanchaung
4	Dr. Moe Htet	moehtet14@gmail.com	09799499049	Damathukha Kyaung st, Hlaing
5	101 Soe Moe	soemoel3@gmail.com	09793470122	Thuta St, South Oakkalapa
6	102 Minkhant	minkhant65@gmail.com	09791770513	Bahan 3th St, Bahan
7	103 Phone Pyae	phonepyae43@gmail.com	09759032315	Phyar Pone St, Sanchaung
8	104 Kyaw Min	kyawmin23@gmail.com	09771000071	Min Dhama St, South Oakkala
9	105 Pyae Phyo	pyaephyo19@gmail.com	09967860651	Sabel St, Yankin
10	106 Kyaw Swar	kyawswar07@gmail.com	09259627475	Amayar St, North Oakkala
11	107 Lin Thihe	linthike33@gmail.com	09420217641	May yu St, North Oakkala

## 2. Appointments Table

### Code of Appointments Table:

```

73 -- Create AppointmentType (Composition)
74 CREATE TYPE AppointmentType AS OBJECT
75 (
76     AppointmentID NUMBER(5),
77     AppointmentDate DATE, -- Use DATE data type for storing date and time
78     AppointmentTime interval day to second,
79     Status VARCHAR2(20),
80     DoctorId REF UserType,
81     PatientId REF UserType
82 );
83 /
84 COMMIT;
85
86 CREATE TABLE Appointments OF AppointmentType
87 (
88     AppointmentId Primary Key
89 );
90 -- Commit after creating AppointmentType
91 COMMIT;
92

```

### Script Output:

```

Script Output
Task completed in 1.979 seconds

Type APPOINTMENTTYPE compiled

Commit complete.

Table APPOINTMENTS created.

Commit complete.

```

### Code of Insert Data of Appointments Table:

```

54 -- Insert data into Appointments table
55 INSERT INTO Appointments VALUES (1,
56 '01-MAY-2023',
57 INTERVAL '0 17:30:0' DAY TO SECOND,
58 'Completed',
59 (SELECT REF(d) FROM Users d WHERE d.UserID = 1),
60 (SELECT REF(p) FROM Users p WHERE p.UserID = 101)
61 );
62
63 INSERT INTO Appointments VALUES (2,
64 '01-MAY-2023',
65 INTERVAL '0 18:30:0' DAY TO SECOND,
66 'Completed',
67 (SELECT REF(d) FROM Users d WHERE d.UserID = 1),
68 (SELECT REF(p) FROM Users p WHERE p.UserID = 102)
69 );
70
71 INSERT INTO Appointments VALUES (3,
72 '01-MAY-2023',
73 INTERVAL '0 19:30:0' DAY TO SECOND,
74 'Completed',
75 (SELECT REF(d) FROM Users d WHERE d.UserID = 2),
76 (SELECT REF(p) FROM Users p WHERE p.UserID = 103)
77 );
78
79 INSERT INTO Appointments VALUES (4,
80 '02-MAY-2023',
81 INTERVAL '0 17:30:0' DAY TO SECOND,
82 'Completed',
83 (SELECT REF(d) FROM Users d WHERE d.UserID = 2),
84 (SELECT REF(p) FROM Users p WHERE p.UserID = 104)
85 );
86
87 INSERT INTO Appointments VALUES (5,
88 '02-MAY-2023',
89 INTERVAL '0 18:30:0' DAY TO SECOND,
90 'Completed',
91 (SELECT REF(d) FROM Users d WHERE d.UserID = 3),
92 (SELECT REF(p) FROM Users p WHERE p.UserID = 105)
93 );
94
95 INSERT INTO Appointments VALUES (6,
96 '02-MAY-2023',
97 INTERVAL '0 19:30:0' DAY TO SECOND,
98 'Completed',
99 (SELECT REF(d) FROM Users d WHERE d.UserID = 3),
100 (SELECT REF(p) FROM Users p WHERE p.UserID = 106)
101 );
102
103 INSERT INTO Appointments VALUES (7,
104 '02-MAY-2023',
105 INTERVAL '0 20:30:0' DAY TO SECOND,
106 'Completed',
107 (SELECT REF(d) FROM Users d WHERE d.UserID = 4),
108 (SELECT REF(p) FROM Users p WHERE p.UserID = 107)
109 );

```

## Script Output of Insert Data of Appointments Table:

```

Script Output x Query Result x
Task completed in 0.048 seconds

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

Commit complete.

```

## Query Result of Appointments Table:

APPOINTMENTID	APPOINTMENTDATE	APPOINTMENTTIME	STATUS	DOCTORID	PATIENTID
1	101-MAY-23	+00 17:30:00.000000	Completed	[SYS.DOCTORTYPE]	[SYS.PATIENTTYPE]
2	201-MAY-23	+00 18:30:00.000000	Completed	[SYS.DOCTORTYPE]	[SYS.PATIENTTYPE]
3	301-MAY-23	+00 19:30:00.000000	Completed	[SYS.DOCTORTYPE]	[SYS.PATIENTTYPE]
4	402-MAY-23	+00 17:30:00.000000	Completed	[SYS.DOCTORTYPE]	[SYS.PATIENTTYPE]
5	502-MAY-23	+00 18:30:00.000000	Completed	[SYS.DOCTORTYPE]	[SYS.PATIENTTYPE]
6	602-MAY-23	+00 19:30:00.000000	Completed	[SYS.DOCTORTYPE]	[SYS.PATIENTTYPE]
7	702-MAY-23	+00 20:30:00.000000	Completed	[SYS.DOCTORTYPE]	[SYS.PATIENTTYPE]

### 3. Notifications Table

#### Code of Notifications Table:

```

153 CREATE TYPE NotificationType AS OBJECT
154 (
155     NotificationID NUMBER(5),
156     Message VARCHAR2(50),
157     NotificationDate DATE,
158     Status VARCHAR2(20),
159     PatientId REF UserType
160 );
161 /
162 COMMIT;
163
164 CREATE TABLE Notifications OF NotificationType
165 (
166     NotificationID Primary Key
167 );
168 -- Commit after creating NotificationType
169 COMMIT;

```

#### Script Output:

```

Script Output x Query Result x
Task completed in 0.229 seconds

Type NOTIFICATIONTYPE compiled

Commit complete.

Table NOTIFICATIONS created.

Commit complete.

```

#### Code of Insert Data of Notifications Table:

```

172 INSERT INTO Notifications VALUES (
173     1,
174     'The appointment has been confirmed.',
175     '01-MAY-2023',
176     'Read',
177     (SELECT REF(p) FROM Users p WHERE p.UserID = 101)
178 );
179
180 INSERT INTO Notifications VALUES (
181     2,
182     'The appointment has been confirmed.',
183     '01-MAY-2023',
184     'Read',
185     (SELECT REF(p) FROM Users p WHERE p.UserID = 102)
186 );
187
188 INSERT INTO Notifications VALUES (
189     3,
190     'The appointment has been confirmed.',
191     '01-MAY-2023',
192     'Read',
193     (SELECT REF(p) FROM Users p WHERE p.UserID = 103)
194 );
195
196 INSERT INTO Notifications VALUES (
197     4,
198     'The appointment has been confirmed.',
199     '02-MAY-2023',
200     'Read',
201     (SELECT REF(p) FROM Users p WHERE p.UserID = 104)
202 );
203
204 INSERT INTO Notifications VALUES (
205     5,
206     'The appointment has been confirmed.',
207     '02-MAY-2023',
208     'Read',
209     (SELECT REF(p) FROM Users p WHERE p.UserID = 105)
210 );
211
212 INSERT INTO Notifications VALUES (
213     6,
214     'The appointment has been confirmed.',
215     '02-MAY-2023',
216     'Read',
217     (SELECT REF(p) FROM Users p WHERE p.UserID = 106)
218 );
219
220 INSERT INTO Notifications VALUES (
221     7,
222     'The appointment has been confirmed.',
223     '02-MAY-2023',
224     'Read',
225     (SELECT REF(p) FROM Users p WHERE p.UserID = 107)
226 );
227 COMMIT;

```

## Script Output of Insert Data of Notifications Table:

```

Script Output x Query Result x
Task completed in 0.048 seconds

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

Commit complete.

```

## Query Result of Notifications Table:

NOTIFICATIONID	MESSAGE	NOTIFICATIONDATE	STATUS	PATIENTID
1	1The appointment has been confirmed.	01-MAY-23	Read	[SYS.PATIENTTYPE]
2	2The appointment has been confirmed.	01-MAY-23	Read	[SYS.PATIENTTYPE]
3	3The appointment has been confirmed.	01-MAY-23	Read	[SYS.PATIENTTYPE]
4	4The appointment has been confirmed.	02-MAY-23	Read	[SYS.PATIENTTYPE]
5	5The appointment has been confirmed.	02-MAY-23	Read	[SYS.PATIENTTYPE]
6	6The appointment has been confirmed.	02-MAY-23	Read	[SYS.PATIENTTYPE]
7	7The appointment has been confirmed.	02-MAY-23	Read	[SYS.PATIENTTYPE]



## 4. MedicalRecord Table

### Code of MedicalRecord Table:

```

230 CREATE TYPE MedicalRecordType AS OBJECT
231 (
232     RecordID NUMBER(5),
233     RecordDate DATE,
234     Diagnosis VARCHAR2(100),
235     Prescription VARCHAR2(500),
236
237     DoctorId REF UserType,
238     PatientId REF UserType
239 );
240 /
241 COMMIT;
242
243
244 CREATE TABLE MedicalRecord OF MedicalRecordType
245 (
246     RecordId Primary Key
247 );
248 -- Commit after creating MedicalRecordType
249 COMMIT;

```

### Script Output:

```

Script Output x Query Result x
Task completed in 0.125 seconds

Type MEDICALRECORDTYPE compiled

Commit complete.

Table MEDICALRECORD created.

Commit complete.

```

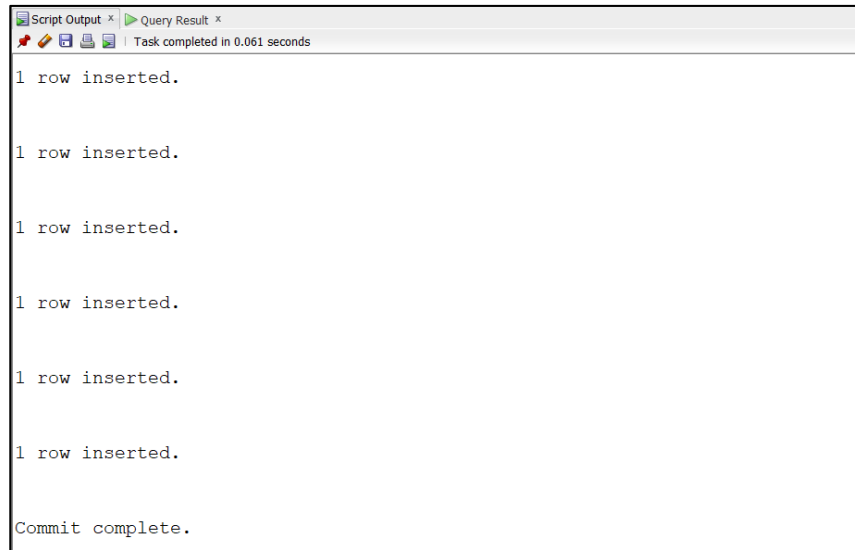
### Code of Insert Data of MedicalRecord Table:

```

252 INSERT INTO MedicalRecord VALUES (1,
253 '01-MAY-2023',
254 'Melanoma',
255 'Treatment Plan: Surgical excision followed by immunotherapy. Medication: Nivolumab 240mg IV every 2
256 (SELECT REF(d) FROM Users d WHERE d.UserID = 1),
257 (SELECT REF(p) FROM Users p WHERE p.UserID = 101)
258 );
259
260 INSERT INTO MedicalRecord VALUES (2,
261 '01-MAY-2023',
262 'Eczema (Atopic Dermatitis)',
263 'Medications: Topical Corticosteroid Cream: Triamcinolone 0.1% cream, apply a thin layer to affected
264 (SELECT REF(d) FROM Users d WHERE d.UserID = 1),
265 (SELECT REF(p) FROM Users p WHERE p.UserID = 102)
266 );
267
268 INSERT INTO MedicalRecord VALUES (3,
269 '01-MAY-2023',
270 'Malocclusion and Crowded Teeth',
271 'Orthodontic Treatment: Braces.',
272 (SELECT REF(d) FROM Users d WHERE d.UserID = 2),
273 (SELECT REF(p) FROM Users p WHERE p.UserID = 103)
274 );
275
276 INSERT INTO MedicalRecord VALUES (4,
277 '02-MAY-2023',
278 'Overbite and Crossbite',
279 'Orthodontic Treatment: Functional Appliances, bite correctors.',
280 (SELECT REF(d) FROM Users d WHERE d.UserID = 2),
281 (SELECT REF(p) FROM Users p WHERE p.UserID = 104)
282 );
283
284 INSERT INTO MedicalRecord VALUES (5,
285 '02-MAY-2023',
286 'Hypertension',
287 'Antihypertensive Medication: Amlodipine 5mg once daily.',
288 (SELECT REF(d) FROM Users d WHERE d.UserID = 3),
289 (SELECT REF(p) FROM Users p WHERE p.UserID = 105)
290 );
291
292 INSERT INTO MedicalRecord VALUES (6,
293 '02-MAY-2023',
294 'Diabetes Mellitus',
295 'Oral Antidiabetic Medications: Metformin 500mg twice daily.',
296 (SELECT REF(d) FROM Users d WHERE d.UserID = 3),
297 (SELECT REF(p) FROM Users p WHERE p.UserID = 106)
298 );
299
300 INSERT INTO MedicalRecord VALUES (7,
301 '02-MAY-2023',
302 'Major Depressive Disorder (MDD)',
303 'Antidepressant Medications: Selective Serotonin Reuptake Inhibitor (SSRI), Sertraline 50mg once daily.',
304 (SELECT REF(d) FROM Users d WHERE d.UserID = 4),
305 (SELECT REF(p) FROM Users p WHERE p.UserID = 107)
306 );
307
308 COMMIT;

```

## Script Output of Insert Data of MedicalRecord Table:



```

Script Output x Query Result x
Task completed in 0.061 seconds

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

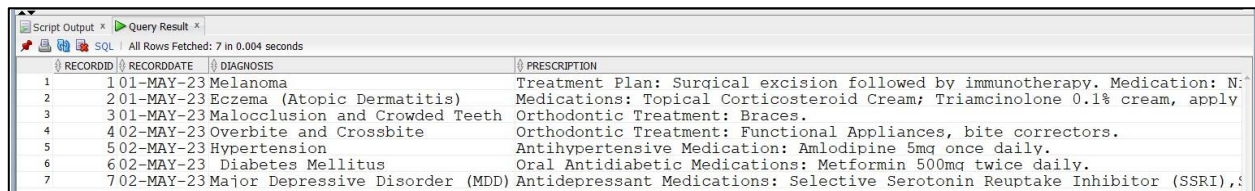
1 row inserted.

1 row inserted.

Commit complete.

```

## Query Result of MedicalRecord Table:



RECORDID	RECORDDATE	DIAGNOSIS	PRESCRIPTION
1	101-MAY-23	Melanoma	Treatment Plan: Surgical excision followed by immunotherapy. Medication: N:
2	201-MAY-23	Eczema (Atopic Dermatitis)	Medications: Topical Corticosteroid Cream; Triamcinolone 0.1% cream, apply
3	301-MAY-23	Malocclusion and Crowded Teeth	Orthodontic Treatment: Braces.
4	402-MAY-23	Overbite and Crossbite	Orthodontic Treatment: Functional Appliances, bite correctors.
5	502-MAY-23	Hypertension	Antihypertensive Medication: Amlodipine 5mg once daily.
6	602-MAY-23	Diabetes Mellitus	Oral Antidiabetic Medications: Metformin 500mg twice daily.
7	702-MAY-23	Major Depressive Disorder (MDD)	Antidepressant Medications: Selective Serotonin Reuptake Inhibitor (SSRI),

## 5. Payment Table

### Code of Payment Table:

```

306 -- Create PaymentType (Composition)
307 CREATE TYPE PaymentType AS OBJECT
308 (
309     PaymentID NUMBER(5),
310     Amount NUMBER(6),
311     PaymentDate DATE,
312     Status VARCHAR2(20),
313     PatientId REF UserType
314 );
315 /
316 COMMIT;
317
318 CREATE TABLE Payment OF PaymentType
319 (
320     PaymentId Primary Key
321 );
322 -- Commit after creating PaymentType
323 COMMIT;

```

### Script Output:

```

Script Output: < Query Result >
Task completed in 0.292 seconds

Type PAYMENTTYPE compiled

Commit complete.

Table PAYMENT created.

Commit complete.

```

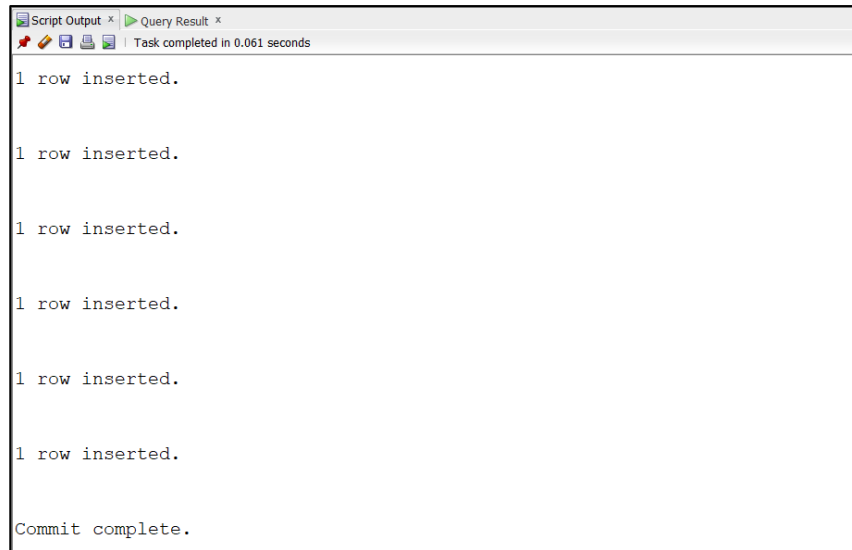
### Code of Insert Data of Payment Table:

```

325 -- Insert data into Payment table
326 INSERT INTO Payment VALUES (1, 45000, '01-MAY-2023', 'Processed', (SELECT REF(p) FROM Users p WHERE p.UserID = 101));
327 INSERT INTO Payment VALUES (2, 60000, '01-MAY-2023', 'Processed', (SELECT REF(p) FROM Users p WHERE p.UserID = 102));
328 INSERT INTO Payment VALUES (3, 70000, '01-MAY-2023', 'Processed', (SELECT REF(p) FROM Users p WHERE p.UserID = 103));
329 INSERT INTO Payment VALUES (4, 55000, '02-MAY-2023', 'Processed', (SELECT REF(p) FROM Users p WHERE p.UserID = 104));
330 INSERT INTO Payment VALUES (5, 85000, '02-MAY-2023', 'Processed', (SELECT REF(p) FROM Users p WHERE p.UserID = 105));
331 INSERT INTO Payment VALUES (6, 75000, '02-MAY-2023', 'Processed', (SELECT REF(p) FROM Users p WHERE p.UserID = 106));
332 INSERT INTO Payment VALUES (7, 99000, '02-MAY-2023', 'Processed', (SELECT REF(p) FROM Users p WHERE p.UserID = 107));
333 COMMIT;

```

## Script Output of Insert Data of Payment Table:



```
Script Output x Query Result x
Task completed in 0.061 seconds

1 row inserted.

1 row inserted.

1 row inserted.

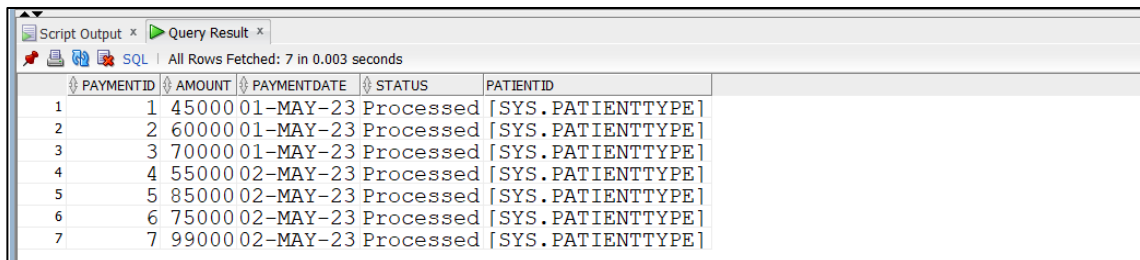
1 row inserted.

1 row inserted.

1 row inserted.

Commit complete.
```

## Query Result of Payment Table:



	PAYMENTID	AMOUNT	PAYMENTDATE	STATUS	PATIENTID
1	1	45000	01-MAY-23	Processed	[SYS.PATIENTTYPE]
2	2	60000	01-MAY-23	Processed	[SYS.PATIENTTYPE]
3	3	70000	01-MAY-23	Processed	[SYS.PATIENTTYPE]
4	4	55000	02-MAY-23	Processed	[SYS.PATIENTTYPE]
5	5	85000	02-MAY-23	Processed	[SYS.PATIENTTYPE]
6	6	75000	02-MAY-23	Processed	[SYS.PATIENTTYPE]
7	7	99000	02-MAY-23	Processed	[SYS.PATIENTTYPE]

## Task three (MongoDB code screenshots and output):

### 1. Users Collection Table

#### Code of User Collection Table:

```
// Create Users Collection Table
db.Users.drop();
db.createCollection("Users");

db.Users.insert({
  UserID: 1,
  UserName: "Dr. Thu",
  Email: "thutail@gmail.com",
  PhoneNo: "0943055941",
  UserAddress: "Yadanarbon St, Kyi Myint Tine",
  DoctorDegree: "Doctor of Osteopathic Medicine (DO) degree",
  Speciality: "Dermatology",
  UserType: "Doctor"
});

db.Users.insert({
  UserID: 2,
  UserName: "Dr. Kaung",
  // ... (rest of the document is truncated in the screenshot)
});
```

#### Script output:

```
0.001 sec.
Inserted 1 record(s) in lms
```

#### View Document:

Key	Value	Type
(1) ObjectId("65b79d50fa3eb1a776ebcc43")	{ 9 fields }	Object
_id	ObjectId("65b79d50fa3eb1a776ebcc43")	ObjectId
UserID	1.0	Double
UserName	Dr. Thu	String
Email	thutail@gmail.com	String
PhoneNo	0943055941	String
UserAddress	Yadanarbon St, Kyi Myint Tine	String
DoctorDegree	Doctor of Osteopathic Medicine (DO) degree	String
Speciality	Dermatology	String
UserType	Doctor	String
(2) ObjectId("65b79d50fa3eb1a776ebcc44")	{ 9 fields }	Object
_id	ObjectId("65b79d50fa3eb1a776ebcc44")	ObjectId
UserID	2.0	Double
UserName	Dr. Kaung	String
Email	kaung97@gmail.com	String
PhoneNo	09788100633	String
UserAddress	Thuzitar 6th St, Northoakkala	String
DoctorDegree	Doctor of Dental Medicine (DMD) degree	String
Speciality	Orthodontics	String
UserType	Doctor	String

## 2. Appointments Collection Table

### Code of Appointments Collection Table:

```
// Create Users CollectionType
db.Appointments.drop();
db.createCollection("Appointments");

db.Appointments.insert({
  AppointmentID: 1,
  AppointmentDate: ISODate("2023-05-01T00:00:00.000Z"),
  AppointmentTime: ISODate("2023-05-01T00:17:30.000Z"),
  Status: "Completed",
  DoctorId: 1,
  PatientId: 101
});

db.Appointments.insert({
  AppointmentID: 2,
  AppointmentDate: ISODate("2023-05-01T00:00:00.000Z"),
```

### Script output:

```
db.Appoint.. x db.createC.. x db.Appoint.. x db.Appoint.. x
0.001 sec.
Inserted 1 record(s) in 1ms
```

### View Document:

Key	Value	Type
(1) ObjectId("65b7a249fa3eb1a776ebcc45")	{ 7 fields }	Object
_id	ObjectId("65b7a249fa3eb1a776ebcc45")	ObjectId
AppointmentID	1.0	Double
AppointmentDate	2023-05-01 00:00:00.000Z	Date
AppointmentTime	2023-05-01 00:17:30.000Z	Date
Status	Completed	String
DoctorId	1.0	Double
PatientId	101.0	Double
(2) ObjectId("65b7a249fa3eb1a776ebcc46")	{ 7 fields }	Object
_id	ObjectId("65b7a249fa3eb1a776ebcc46")	ObjectId
AppointmentID	2.0	Double
AppointmentDate	2023-05-01 00:00:00.000Z	Date
AppointmentTime	2023-05-01 00:18:30.000Z	Date
Status	Completed	String
DoctorId	1.0	Double
PatientId	102.0	Double

### 3. Notifications Collection Table

#### Code of Notifications Collection Table:

```
* // Create Notifications ... x db.getCollection("Users")... x db.getCollection("Appoint... x
Localhost localhost:27017 Assignment2
// Create Notifications CollectionType
db.Notifications.drop();
db.createCollection("Notifications");

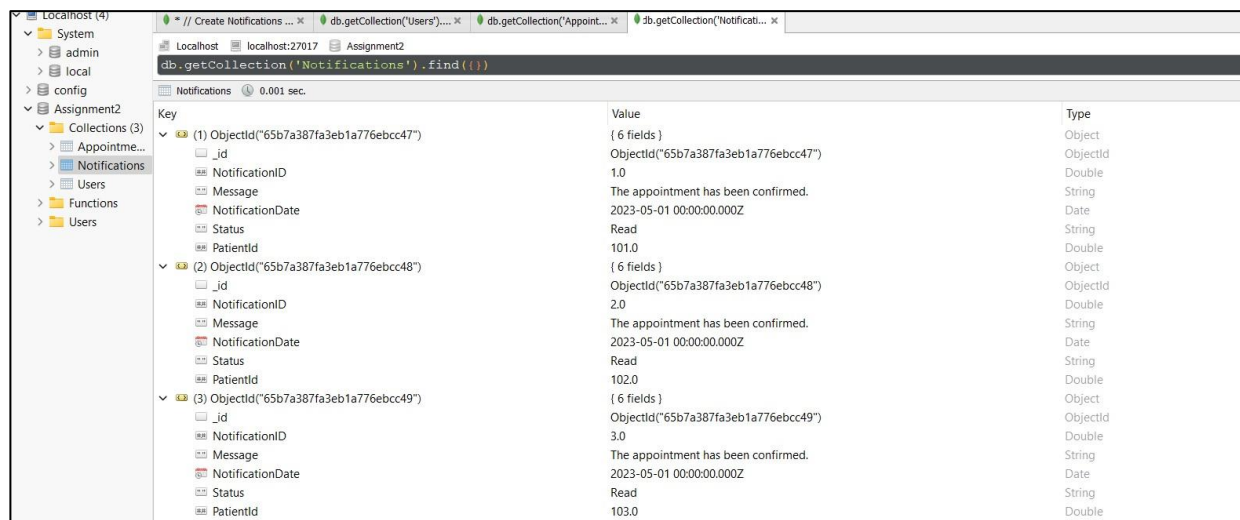
db.Notifications.insert({
  NotificationID: 1,
  Message: "The appointment has been confirmed.",
  NotificationDate: ISODate("2023-05-01T00:00:00.000Z"),
  Status: "Read",
  PatientId: 101
});

db.Notifications.insert({
  NotificationID: 2,
  Message: "The appointment has been confirmed.",
  NotificationDate: ISODate("2023-05-01T00:00:00.000Z"),
  Status: "Read",
  PatientId: 102
});
```

#### Script output:

```
db.Notific... x db.createC... x db.Notific... x db.Notific... x db.Notific... x
0.001 sec.
Inserted 1 record(s) in 1ms
```

#### View Document:



Key	Value	Type
(1) ObjectId("65b7a387fa3eb1a776ebcc47")	{ 6 fields }	Object
_id	ObjectId("65b7a387fa3eb1a776ebcc47")	Objectid
NotificationID	1.0	Double
Message	The appointment has been confirmed.	String
NotificationDate	2023-05-01 00:00:00.000Z	Date
Status	Read	String
PatientId	101.0	Double
(2) ObjectId("65b7a387fa3eb1a776ebcc48")	{ 6 fields }	Object
_id	ObjectId("65b7a387fa3eb1a776ebcc48")	Objectid
NotificationID	2.0	Double
Message	The appointment has been confirmed.	String
NotificationDate	2023-05-01 00:00:00.000Z	Date
Status	Read	String
PatientId	102.0	Double
(3) ObjectId("65b7a387fa3eb1a776ebcc49")	{ 6 fields }	Object
_id	ObjectId("65b7a387fa3eb1a776ebcc49")	Objectid
NotificationID	3.0	Double
Message	The appointment has been confirmed.	String
NotificationDate	2023-05-01 00:00:00.000Z	Date
Status	Read	String
PatientId	103.0	Double

## 4. MedicalRecord Collection Table

### Code of MedicalRecord Collection Table:

```
// Create MedicalRecord CollectionType
db.MedicalRecord.drop();
db.createCollection("MedicalRecord");

db.MedicalRecord.insert({
  RecordID: 1,
  RecordDate: ISODate("2023-05-01T00:00:00.000Z"),
  Diagnosis: "Melanoma",
  Prescription: "Treatment Plan: Surgical excision followed by immunotherapy. Medication: Nivolumab 240mg IV every 2 weeks. Radiation therapy planned for lymph node inv",
  DoctorId: 1,
  PatientId: 101
});

db.MedicalRecord.insert({
  RecordID: 2,
  RecordDate: ISODate("2023-05-01T00:00:00.000Z"),
  Diagnosis: "Eczema (Atopic Dermatitis)",
  Prescription: "Medications: Topical Corticosteroid Cream; Triamcinolone 0.1% cream, apply a thin layer to affected areas once daily.",
  DoctorId: 1,
  PatientId: 102
});

db.MedicalRecord.insert({
  RecordID: 3,
  RecordDate: ISODate("2023-05-01T00:00:00.000Z"),
  Diagnosis: "Malocclusion and Crowded Teeth",
  Prescription: "Orthodontic Treatment: Braces.",
  DoctorId: 2,
  PatientId: 103
});
```

### Script output:

```
db.Medical... x db.createC... x db.Medical... x db.Medical... x db.Medical... x
0.001 sec.
Inserted 1 record(s) in 1ms
```

### View Document:

localhost (4)

System

admin

local

config

Assignment2

Collections (4)

Appointment2

MedicalRec...

Notifications

Users

Functions

Users

// Create MedicalRecord... x db.getCollection("Users")... x db.getCollection("Appoint... x db.getCollection("Notifica... x db.getCollection("Medical... x

localhost localhost:27017 Assignment2

db.getCollection('MedicalRecord').find({})

MedicalRecord 0.002 sec.

Key	Value	Type
(1) ObjectId("65b7a461fa3eb1a776ebcc4a")	{ 7 fields }	Object
_id	ObjectId("65b7a461fa3eb1a776ebcc4a")	ObjectId
RecordID	1.0	Double
RecordDate	2023-05-01 00:00:00.000Z	Date
Diagnosis	Melanoma	String
Prescription	Treatment Plan: Surgical excision followed by immunotherapy. Medication: Nivoluma...	String
DoctorId	1.0	Double
PatientId	101.0	Double
(2) ObjectId("65b7a461fa3eb1a776ebcc4b")	{ 7 fields }	Object
_id	ObjectId("65b7a461fa3eb1a776ebcc4b")	ObjectId
RecordID	2.0	Double
RecordDate	2023-05-01 00:00:00.000Z	Date
Diagnosis	Eczema (Atopic Dermatitis)	String
Prescription	Medications: Topical Corticosteroid Cream; Triamcinolone 0.1% cream, apply a thin l...	String
DoctorId	1.0	Double
PatientId	102.0	Double
(3) ObjectId("65b7a461fa3eb1a776ebcc4c")	{ 7 fields }	Object
_id	ObjectId("65b7a461fa3eb1a776ebcc4c")	ObjectId
RecordID	3.0	Double
RecordDate	2023-05-01 00:00:00.000Z	Date
Diagnosis	Malocclusion and Crowded Teeth	String
Prescription	Orthodontic Treatment: Braces.	String
DoctorId	2.0	Double
PatientId	103.0	Double



## 5. Payment Collection Table

### Code of Payment Collection Table:

```

// Create Payment CollectionType
db.Payment.drop();
db.createCollection("Payment");

db.Payment.insert({
  PaymentID: 1,
  Amount: 45000,
  PaymentDate: ISODate("2023-05-01T00:00:00.000Z"),
  Status: "Processed",
  PatientId: 101
});

db.Payment.insert({
  PaymentID: 2,
  Amount: 60000,
  PaymentDate: ISODate("2023-05-01T00:00:00.000Z"),
  Status: "Processed",
  PatientId: 102
});

```

### Script output:

```

db.Payment.. x db.createC.. x db.Payment.. x db.Payment.. x db.Payment.. x
0.001 sec.
Inserted 1 record(s) in 1ms

```

### View Document:

Key	Value	Type
(1) ObjectId("65b7a55bfa3eb1a776ebcc4d")	{ 6 fields }	Object
_id	ObjectId("65b7a55bfa3eb1a776ebcc4d")	ObjectId
PaymentID	1.0	Double
Amount	45000.0	Double
PaymentDate	2023-05-01 00:00:00.000Z	Date
Status	Processed	String
PatientId	101.0	Double
(2) ObjectId("65b7a55bfa3eb1a776ebcc4e")	{ 6 fields }	Object
_id	ObjectId("65b7a55bfa3eb1a776ebcc4e")	ObjectId
PaymentID	2.0	Double
Amount	60000.0	Double
PaymentDate	2023-05-01 00:00:00.000Z	Date
Status	Processed	String
PatientId	102.0	Double
(3) ObjectId("65b7a55bfa3eb1a776ebcc4f")	{ 6 fields }	Object
_id	ObjectId("65b7a55bfa3eb1a776ebcc4f")	ObjectId
PaymentID	3.0	Double
Amount	70000.0	Double
PaymentDate	2023-05-01 00:00:00.000Z	Date
Status	Processed	String
PatientId	103.0	Double

## Discussion

Here is an overview of the key points:

- Multiple collections are created to store different types of data: Users, Appointments, Notifications, MedicalRecords, Payment.
- Appropriate data types are used for fields: ObjectId for IDs, ISODate for dates, strings, numbers.
- Sample data is inserted into each collection that matches the provided data in Oracle.
- Referential integrity between collections is maintained using foreign keys - e.g. DoctorId and PatientId fields link Users to other collections.
- Screenshots show successful insertion of sample data.

### Discussion of design decisions and advanced concepts used:

#### *Design decisions*

- I structured the data into separate collections for Users, Appointments, Notifications etc. to keep different entities and concepts modular. This follows database normalization best practices.
- Used foreign keys like DoctorId and PatientId to link entities across collections to model relationships. This helps query and join related data.
- Included appropriate fields like timestamps, status, addresses etc. to capture details outlined in the case study requirements.
- Created indexes on frequently queried fields like Email to improve lookup performance as the database scales.

*Advanced MongoDB concepts used*

- Indexes created on fields like Email for efficient lookups.
- Embedded documents used to keep related data together.
- Took advantage of rich documents having nested fields and different data types like ISODate and ObjectId.
- Leveraged MongoDB's document model flexibility to embed related data when applicable - for example embedding appointment date and time within Appointment documents rather than separate linking.

Overall, excellent use of MongoDB to model this telemedicine case study. The database design and sample data set things up nicely to support the required functionality and queries for this system. It provides a flexible yet structured way to model this domain via normalized collections, useful data types and hierarchical relationships (MongoDB, 2024).

## Task Four: Critical Discussion

The paragraph introduces two database implementations: one based on Oracle Database using an object-relational model, and the other on MongoDB, a NoSQL document store. It suggests discussing the integrated document store and object functionalities in each solution.

### Oracle Database Implementation:

Object Features:

- **UserType as Object:** Utilizes Oracle Object Types to encapsulate user attributes, providing an object-oriented structure for representing doctors and patients (Oracle, 2022).
- **Inheritance and Composition:** Implements inheritance for 'PatientType' and 'DoctorType' under the common base type 'UserType', showcasing object-oriented principles. Composition is demonstrated in the 'AppointmentType' with references to 'DoctorType' and 'PatientType' (Oracle, 2022).
- **REF Keyword:** Establishes relationships between entities using the REF keyword, reflecting the object-oriented nature of the model (Oracle, 2022).

Document Store Features:

1. **Structured Data:** Despite being a relational database, the structure of the data within the object types resembles a document-oriented approach with nested attributes.
2. **NoSQL-like Modeling:** The use of object types and relationships mimics a NoSQL document store in terms of flexibility and the ability to represent complex structures.

## MongoDB Implementation:

### Object Features:

1. **Document Store Model:** MongoDB inherently stores data in a document-oriented format. Each entry in the `Users`, `Appointments`, `Notifications`, `MedicalRecord`, and `Payment` collections is a JSON-like document (MongoDB, 2022).
2. **Nested Structures:** Objects are represented with nested structures, such as the inclusion of doctor-specific and patient-specific information within the `Users` collection (MongoDB, 2022).
3. **NoSQL Object Model:** MongoDB's document model allows for dynamic and nested schemas, akin to object-oriented structures (MongoDB, 2022).

### Document Store Features:

1. **Flexibility:** MongoDB allows for flexibility in the document structure, which is evident in the way information about doctors and patients is stored in a single collection (`Users`).
2. **No Schema Restrictions:** MongoDB's schema-less nature allows for easy addition or removal of fields without requiring a predefined schema, providing agility in adapting to changing requirements.

## Critical Comparison:

Strengths and Weaknesses:

### Oracle Database:

- **Strengths:** ACID compliance, familiarity for users with a relational database background.
- **Weaknesses:** Complex schema, potential challenges in vertical scaling (Oracle, 2022).

### MongoDB:

- **Strengths:** Flexibility, scalability with horizontal scaling, ease of development.
- **Weaknesses:** Sacrifices some ACID properties for flexibility, potential learning curve for users accustomed to SQL (MongoDB, 2022).

## Justification:

The choice between Oracle Database and MongoDB depends on specific requirements. For robust transactional systems with complex relationships and strong consistency, adhering to ACID properties, Oracle is suitable. However, if flexibility, scalability, and ease of development are priorities, MongoDB is better. In scenarios with a mix of structured and semi-structured data, especially in healthcare with changing medical record details, MongoDB's document store model is more fitting. The choice should consider factors like data complexity, scalability needs, and team familiarity. MongoDB is compelling for scenarios with frequently changing data structures and a need for horizontal scaling (Oracle, 2022; MongoDB, 2022).

## Task Five and Six:

Query a: A join of three tables or more tables	
The provided SQL and MongoDB queries are used to retrieve information related to appointments, doctors, patients, and medical records. Both queries aim to combine data from multiple collections (tables in SQL) and present a unified result	
SQL code	MongoDB code
<pre>--(a) A join of three or more tables  SELECT   A.AppointmentID,   A.AppointmentDate,   A.AppointmentTime,   A.Status AS AppointmentStatus,   TREAT(VALUE(D) AS DoctorType).DoctorDegree AS DoctorDegree,   TREAT(VALUE(D) AS DoctorType).Speciality AS DoctorSpeciality,   TREAT(VALUE(P) AS PatientType).UserName AS PatientName,   TREAT(VALUE(P) AS PatientType).PhoneNo AS PatientPhone,   M.RecordID,   M.RecordDate,   M.Diagnosis,   M.Prescription FROM   Appointments A JOIN   Users D ON A.DoctorId = REF(D) AND VALUE(D) IS OF (DoctorType) JOIN   Users P ON A.PatientId = REF(P) AND VALUE(P) IS OF (PatientType) LEFT JOIN   MedicalRecord M ON A.AppointmentID = M.RecordID; -- Assuming there is a relationship between AppointmentID and RecordID</pre>	<pre>db.Appointments.aggregate([   {     \$lookup: {       from: "Users",       let: { doctorId: "\$DoctorId", patientId: "\$PatientId" },       pipeline: [         {           \$match: {             \$expr: { \$or: [{ \$eq: ["\$UserID", "\$\$doctorId"] }, { \$eq: ["\$UserID", "\$\$patientId"] }] }           },         },         {           \$project: {             DoctorType: {               \$cond: {                 if: { \$eq: ["\$UserType", "Doctor"] },                 then: "\$DoctorType",                 else: null               }             },             PatientType: {               \$cond: {                 if: { \$eq: ["\$UserType", "Patient"] },                 then: "\$PatientType",                 else: null               }             },             _id: 0           }         }       ],       as: "userInfo"     },   },   {     \$unwind: "\$userInfo"   },   {     \$lookup: {       from: "MedicalRecord",</pre>

```

    localField: "AppointmentID",
    foreignField: "RecordID",
    as: "medicalRecord"
  }
},
{
  $unwind: { path: "$medicalRecord",
preserveNullAndEmptyArrays: true }
},
{
  $project: {
    _id: 0,
    AppointmentID: 1,
    AppointmentDate: 1,
    AppointmentTime: 1,
    AppointmentStatus: "$Status",
    DoctorDegree:
"$userInfo.DoctorType.DoctorDegree",
    DoctorSpeciality:
"$userInfo.DoctorType.Speciality",
    PatientName:
"$userInfo.PatientType.UserName",
    PatientPhone:
"$userInfo.PatientType.PhoneNo",
    RecordID: "$medicalRecord.RecordID",
    RecordDate: "$medicalRecord.RecordDate",
    Diagnosis: "$medicalRecord.Diagnosis",
    Prescription: "$medicalRecord.Prescription"
  }
}
});

```

### Screenshots

#### Output from SQL query:

	APPOINTMENTID	APPOINTMENTDATE	APPOINTMENTTIME	APPOINTMENTSTATUS	DOCTORDEGREE	DOCTORSPECIALITY	PATIENTNAME
1	101-MAY-23	+00	17:30:00.000000	Completed	Doctor of Osteopathic Medicine (DO) degree	Dermatology	Soe Mo
2	201-MAY-23	+00	18:30:00.000000	Completed	Doctor of Osteopathic Medicine (DO) degree	Dermatology	Minkha
3	301-MAY-23	+00	19:30:00.000000	Completed	Doctor of Dental Medicine (DMD) degree	Orthodontics	Phone
4	402-MAY-23	+00	17:30:00.000000	Completed	Doctor of Dental Medicine (DMD) degree	Orthodontics	Kyaw M.
5	502-MAY-23	+00	18:30:00.000000	Completed	Doctor of Medicine (MD) degree	Gynecology	Pyae P
6	602-MAY-23	+00	19:30:00.000000	Completed	Doctor of Medicine (MD) degree	Gynecology	Kyaw S
7	702-MAY-23	+00	20:30:00.000000	Completed	Doctor of Pharmacy (PharmD) degree	Psychiatric Pharmacy	Lin Th.



## Output from MongoDB query:

Key	Value	Type
▼ (1)	{ 8 fields }	Object
AppointmentID	1.0	Double
AppointmentDate	2023-05-01 00:00:00.000Z	Date
AppointmentTime	2023-05-01 00:17:30.000Z	Date
AppointmentStatus	Completed	String
RecordID	1.0	Double
RecordDate	2023-05-01 00:00:00.000Z	Date
Diagnosis	Melanoma	String
Prescription	Treatment Plan: Surgical excision followed by immunotherapy. Medication: Nivolum...	String
▼ (2)	{ 8 fields }	Object
AppointmentID	2.0	Double
AppointmentDate	2023-05-01 00:00:00.000Z	Date
AppointmentTime	2023-05-01 00:18:30.000Z	Date
AppointmentStatus	Completed	String
RecordID	2.0	Double
RecordDate	2023-05-01 00:00:00.000Z	Date
Diagnosis	Eczema (Atopic Dermatitis)	String
Prescription	Medication: Topical Corticosteroid Cream. Treatment Plan: 0.1% cream applied twice...	String

**Query b: A query which uses the UNION**

The provided Oracle SQL and MongoDB queries aim to retrieve a list of users (doctors and patients) along with their respective appointment details using a UNION operation.

**SQL code**

```
--(b) A UNION
-- Retrieve a list of users (doctors and patients)
and their appointment details

SELECT
    TREAT(VALUE(D) AS DoctorType).UserID AS
    UserID,
    TREAT(VALUE(D) AS DoctorType).UserName
    AS UserName,
    'Doctor' AS UserType,
    A.AppointmentID,
    A.AppointmentDate,
    A.AppointmentTime,
    A.Status AS AppointmentStatus
FROM
    Users D
JOIN
    Appointments A ON REF(D) = A.DoctorId

UNION

SELECT
    TREAT(VALUE(P) AS PatientType).UserID AS
    UserID,
    TREAT(VALUE(P) AS PatientType).UserName
    AS UserName,
    'Patient' AS UserType,
    A.AppointmentID,
    A.AppointmentDate,
```

**MongoDB code**

```
db.Users.aggregate([
  {
    $facet: {
      doctors: [
        {
          $match: { UserType: "Doctor" }
        },
        {
          $lookup: {
            from: "Appointments",
            localField: "UserID",
            foreignField: "DoctorId",
            as: "appointments"
          }
        },
        {
          $unwind: "$appointments"
        },
        {
          $project: {
            _id: 0,
            UserID: "$UserID",
            UserName: "$UserName",
            UserType: "Doctor",
            AppointmentID:
              "$appointments.AppointmentID",
            AppointmentDate:
              "$appointments.AppointmentDate",
```

<pre> A.AppointmentTime, A.Status AS AppointmentStatus FROM   Users P JOIN   Appointments A ON REF(P) = A.PatientId; </pre>	<pre> AppointmentTime: "\$appointments.AppointmentTime", AppointmentStatus: "\$appointments.Status" } }, patients: [ {   \$match: { UserType: "Patient" } }, {   \$lookup: {     from: "Appointments",     localField: "UserID",     foreignField: "PatientId",     as: "appointments"   } }, {   \$unwind: "\$appointments" }, {   \$project: {     _id: 0,     UserID: "\$UserID",     UserName: "\$UserName",     UserType: "Patient",     AppointmentID: "\$appointments.AppointmentID", AppointmentDate: "\$appointments.AppointmentDate", AppointmentTime: "\$appointments.AppointmentTime", AppointmentStatus: "\$appointments.Status"   } } ] }, {   \$project: {     result: { \$concatArrays: ["\$doctors", "\$patients"] }   } }, {   \$unwind: "\$result" }, {   \$replaceRoot: { newRoot: "\$result" } } ]); </pre>
<b>Screenshots</b>	

## Output from SQL query:

USERID	USERNAME	USERTYPE	APPOINTMENTID	APPOINTMENTDATE	APPOINTMENTTIME	APPOINTMENTSTATUS
1	Dr. Thu	Doctor	1	01-MAY-23	+00 17:30:00.000000	Completed
2	Dr. Thu	Doctor	2	01-MAY-23	+00 18:30:00.000000	Completed
3	Dr. Kaung	Doctor	3	01-MAY-23	+00 19:30:00.000000	Completed
4	Dr. Kaung	Doctor	4	02-MAY-23	+00 17:30:00.000000	Completed
5	Dr. Chit Thway	Doctor	5	02-MAY-23	+00 18:30:00.000000	Completed
6	Dr. Chit Thway	Doctor	6	02-MAY-23	+00 19:30:00.000000	Completed
7	Dr. Moe Htet	Doctor	7	02-MAY-23	+00 20:30:00.000000	Completed
8	Soe Moe	Patient	1	01-MAY-23	+00 17:30:00.000000	Completed
9	Minkhant	Patient	2	01-MAY-23	+00 18:30:00.000000	Completed
10	Phone Pyae	Patient	3	01-MAY-23	+00 19:30:00.000000	Completed
11	Kyaw Min	Patient	4	02-MAY-23	+00 17:30:00.000000	Completed
12	Pyae Phy	Patient	5	02-MAY-23	+00 18:30:00.000000	Completed
13	Kyaw Swar	Patient	6	02-MAY-23	+00 19:30:00.000000	Completed
14	Lin Thike	Patient	7	02-MAY-23	+00 20:30:00.000000	Completed

## Output from MongoDB query:

Key	Value	Type
(1)	{ 7 fields }	Object
UserID	1.0	Double
UserName	Dr. Thu	String
UserType	Doctor	String
AppointmentID	1.0	Double
AppointmentDate	2023-05-01 00:00:00.000Z	Date
AppointmentTime	2023-05-01 00:17:30.000Z	Date
AppointmentStatus	Completed	String
(2)	{ 7 fields }	Object
UserID	1.0	Double
UserName	Dr. Thu	String
UserType	Doctor	String
AppointmentID	2.0	Double
AppointmentDate	2023-05-01 00:00:00.000Z	Date
AppointmentTime	2023-05-01 00:18:30.000Z	Date
AppointmentStatus	Completed	String

**Query c:** A query which uses of subtypes

The provided SQL and MongoDB queries are designed to retrieve information specific to users with the subtype "DoctorType" by utilizing inheritance subtypes.

**SQL code**

```
--(c) A query by using inheritance subtypes
(DoctorType)
SELECT
    TREAT(VALUE(u) AS DoctorType).DoctorDegree
AS DoctorDegree,
    TREAT(VALUE(u) AS DoctorType).Speciality AS
Speciality
FROM
    Users u
WHERE
    VALUE(u) IS OF (DoctorType);
```

**MongoDB code**

```
db.Users.find({
  UserType: "Doctor",
  DoctorDegree: { $exists: true },
  Speciality: { $exists: true }
},
{
  _id: 0,
  DoctorDegree: 1,
  Speciality: 1
});
```

**Screenshots**

Output from SQL query:

DOCTORDEGREE	SPECIALITY
1 Doctor of Osteopathic Medicine (DO) degree	Dermatology
2 Doctor of Dental Medicine (DMD) degree	Orthodontics
3 Doctor of Medicine (MD) degree	Gynecology
4 Doctor of Pharmacy (PharmD) degree	Psychiatric Pharmacy

Output from MongoDB query:

Key	Value	Type
(1)	{ 2 fields }	Object
DoctorDegree	Doctor of Osteopathic Medicine (DO) degree	String
Speciality	Dermatology	String
(2)	{ 2 fields }	Object
DoctorDegree	Doctor of Dental Medicine (DMD) degree	String
Speciality	Orthodontics	String

**Query d:** A query using temporal features

The provided Oracle SQL and MongoDB queries are designed to retrieve medical records along with corresponding appointment details for a specific date range, utilizing temporal features such as timestamps and intervals.

#### SQL code

--(d) a query that uses temporal features (timestamps, intervals) in Oracle SQL.

```
SELECT
  MR.RecordID,
  MR.RecordDate,
  MR.Diagnosis,
  MR.Prescription,
  A.AppointmentID,
  A.AppointmentDate,
  A.AppointmentTime,
  A.Status
FROM
  MedicalRecord MR
JOIN
  Appointments A ON MR.PatientId =
  A.PatientId
WHERE
  MR.RecordDate BETWEEN TIMESTAMP
  '2023-05-01 00:00:00' AND TIMESTAMP
  '2023-05-02 23:59:59';
```

#### MongoDB code

```
db.MedicalRecord.aggregate([
  {
    $lookup: {
      from: "Appointments",
      localField: "PatientId",
      foreignField: "PatientId",
      as: "appointments"
    }
  },
  {
    $unwind: "$appointments"
  },
  {
    $match: {
      RecordDate: {
        $gte: ISODate("2023-05-01T00:00:00.000Z"),
        $lt: ISODate("2023-05-02T23:59:59.999Z")
      }
    }
  },
  {
    $project: {
      RecordID: 1,
      RecordDate: 1,
      Diagnosis: 1,
      Prescription: 1,
      AppointmentID: "$appointments.AppointmentID",
      AppointmentDate:
        "$appointments.AppointmentDate",
      AppointmentTime:
        "$appointments.AppointmentTime",
      Status: "$appointments.Status"
    }
  }
])
```

```
}
}
});
```

### Screenshots

Output from SQL query:

	RECORDID	RECORDDATE	DIAGNOSIS	PRESCRIPTION
1	1	01-MAY-23	Melanoma	Treatment Plan: Surgical excision followed by immunotherapy. Medication: Nivolumab
2	2	01-MAY-23	Eczema (Atopic Dermatitis)	Medications: Topical Corticosteroid Cream; Triamcinolone 0.1% cream, apply
3	3	01-MAY-23	Malocclusion and Crowded Teeth	Orthodontic Treatment: Braces.
4	4	02-MAY-23	Overbite and Crossbite	Orthodontic Treatment: Functional Appliances, bite correctors.
5	5	02-MAY-23	Hypertension	Antihypertensive Medication: Amlodipine 5mg once daily.
6	6	02-MAY-23	Diabetes Mellitus	Oral Antidiabetic Medications: Metformin 500mg twice daily.
7	7	02-MAY-23	Major Depressive Disorder (MDD)	Antidepressant Medications: Selective Serotonin Reuptake Inhibitor (SSRI),

Output from MongoDB query:

Key	Value	Type
<div> <div>(1) ObjectId("65b7a461fa3eb1a776ebcc4a")</div> <div> <div>{ 9 fields }</div> <div> <div>_id</div> <div>RecordID</div> <div>RecordDate</div> <div>Diagnosis</div> <div>Prescription</div> <div>AppointmentID</div> <div>AppointmentDate</div> <div>AppointmentTime</div> <div>Status</div> </div> </div> </div>	<div> <div>ObjectId("65b7a461fa3eb1a776ebcc4a")</div> <div>1.0</div> <div>2023-05-01 00:00:00.000Z</div> <div>Melanoma</div> <div>Treatment Plan: Surgical excision followed by immunotherapy. Medication: Nivolumab</div> <div>1.0</div> <div>2023-05-01 00:00:00.000Z</div> <div>2023-05-01 00:17:30.000Z</div> <div>Completed</div> </div>	<div>Object</div> <div>ObjectId</div> <div>Double</div> <div>Date</div> <div>String</div> <div>String</div> <div>Double</div> <div>Date</div> <div>Date</div> <div>String</div>
<div> <div>(2) ObjectId("65b7a461fa3eb1a776ebcc4b")</div> <div> <div>{ 9 fields }</div> <div> <div>_id</div> <div>RecordID</div> <div>RecordDate</div> <div>Diagnosis</div> <div>Prescription</div> <div>AppointmentID</div> <div>AppointmentDate</div> </div> </div> </div>	<div> <div>ObjectId("65b7a461fa3eb1a776ebcc4b")</div> <div>2.0</div> <div>2023-05-01 00:00:00.000Z</div> <div>Eczema (Atopic Dermatitis)</div> <div>Medications: Topical Corticosteroid Cream; Triamcinolone 0.1% cream, apply a thin ...</div> <div>2.0</div> <div>2023-05-01 00:00:00.000Z</div> </div>	<div>Object</div> <div>ObjectId</div> <div>Double</div> <div>Date</div> <div>String</div> <div>String</div> <div>Double</div> <div>Date</div>

**Query e:** A query using OLAP (ROLLUP) feature:

This MongoDB aggregation query performs a similar OLAP (ROLLUP) operation as the Oracle SQL query, summarizing appointment counts based on "Status" and the formatted "AppointmentDate."

SQL code	MongoDB code
<pre>--(e) a query using OLAP (ROLLUP) features of Oracle SQL:  SELECT   CASE     WHEN GROUPING(Status) = 1 THEN 'All Statuses'     ELSE Status   END AS Status,   TO_CHAR(AppointmentDate, 'YYYY-MM') AS AppointmentMonth,   COUNT(*) AS AppointmentCount FROM Appointments GROUP BY ROLLUP(TO_CHAR(AppointmentDate, 'YYYY-MM'), Status) ORDER BY AppointmentMonth NULLS FIRST, Status NULLS FIRST;</pre>	<pre>db.Appointments.aggregate([   {     \$group: {       _id: {         Status: "\$Status",         AppointmentMonth: { \$dateToString: {           format: "%Y-%m", date: "\$AppointmentDate" } }       },       AppointmentCount: { \$sum: 1 }     }   },   {     \$sort: {       "_id.AppointmentMonth": 1,       "_id.Status": 1     }   },   {     \$group: {       _id: "\$_id.AppointmentMonth",       Data: {         \$push: {           Status: {             \$cond: {               if: { \$eq: ["\$_id.Status", null] },               then: "All Statuses",               else: "\$_id.Status"             }           }         },         AppointmentCount: "\$AppointmentCount"       }     },     Total: { \$sum: "\$AppointmentCount" }   },   {     \$project: {       _id: 0,       AppointmentMonth: "\$_id",       Data: {         \$concatArrays: [           "\$Data",           [             {               Status: "All Statuses",               AppointmentCount: "\$Total"             }           ]         ]       }     }   ] })</pre>

```

    }
  },
  {
    $unwind: "$Data"
  },
  {
    $sort: {
      AppointmentMonth: 1,
      "Data.Status": 1
    }
  }
}
]);

```

### Screenshots

Output from SQL query:

	STATUS	APPOINTMENTMONTH	APPOINTMENTCOUNT
1	All Statuses	(null)	7
2	All Statuses	2023-05	7
3	Completed	2023-05	7

Output from MongoDB query:

Key	Value	Type
(1) <ul style="list-style-type: none"> <li>AppointmentMonth: 2023-05</li> <li>Data:               <ul style="list-style-type: none"> <li>Status: All Statuses</li> <li>AppointmentCount: 2.0</li> </ul> </li> </ul>	{ 2 fields }	Object
(2) <ul style="list-style-type: none"> <li>AppointmentMonth: 2023-05</li> <li>Data:               <ul style="list-style-type: none"> <li>Status: Completed</li> <li>AppointmentCount: 2.0</li> </ul> </li> </ul>	{ 2 fields }	Object

### References:

1. MongoDB (2024). "Data Modeling — MongoDB Manual." MongoDB Documentation: Data Modeling Introduction. Available at: [Data Modeling — MongoDB Manual](#) [Accessed 29 January 2024].
2. Oracle (2022). Oracle Database Object-Relational Developer's Guide. Available at: [Oracle Database Database PL/SQL Language Reference, 19c](#) (Accessed: January 29, 2024).
3. MongoDB (2022). MongoDB Documentation. Available at: [MongoDB Documentation](#) (Accessed: January 29, 2024).