Table of Contents

Deliverable 1: Business Rules & ER Diagram

Deliverable 2: Normalization, Data Dictionary, and DDL Statements

Deliverable 3: Data, Queries & Updates

Deliverable 4: Complex Queries

Deliverable 5: Views, Triggers & Stored Procedures

Deliverable 6: Final Report

Deliverable 1

Part 1

Our organization is a hospital for a medium sized town. The users would be doctors, nurses, patients, data management, and insurance. Doctors and nurses would use the database to view and update patient records. Patients would use the database to view their own records. Data management would be the overseers of the database, ensuring it works properly as well as verifying records. Insurance would use the database to view information about their registered patients.

Part 2

Entities:

Employees

Positions

Patients

Appointments

Insurance Agencies

Departments

Invoice

Prescriptions

Operations

Rooms

https://www.w3resource.com/sql-exercises/hospital-database-exercise/index.php

https://www.geeksforgeeks.org/dbms/how-to-design-a-database-for-healthcare-management-system

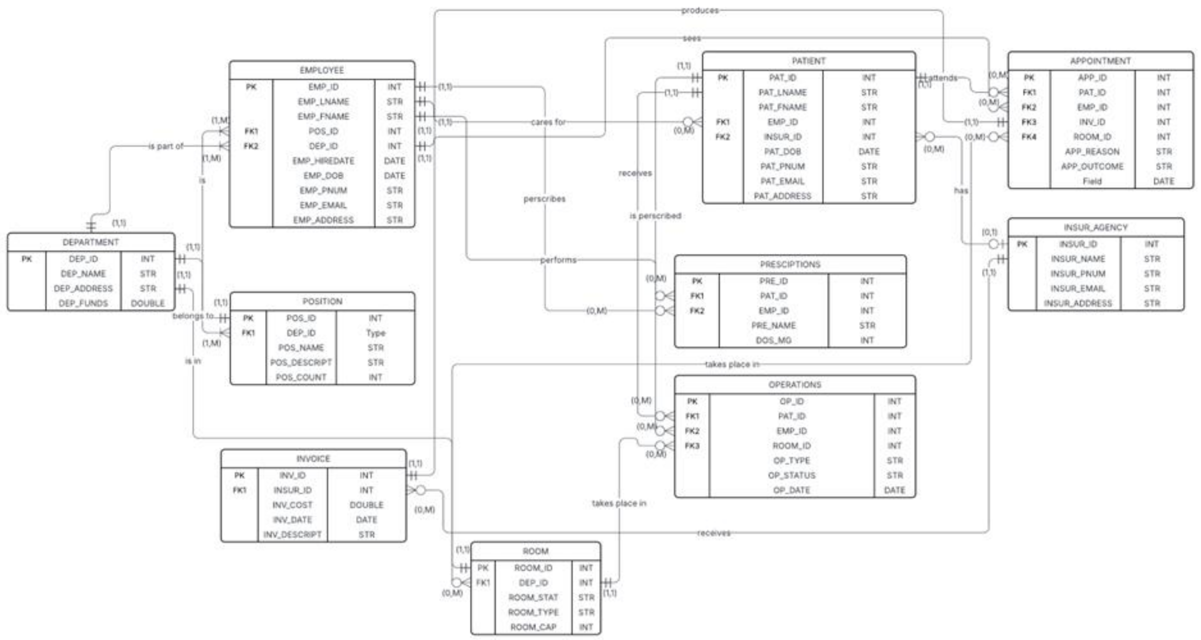
Business Rules:

1. An employee has one position.
2. A position can have multiple employees.
3. A patient can have one employee.

4. An employee can have many patients.
5. An appointment can have one employee.
6. An employee can have many appointments.
7. An appointment can have one patient.
8. A patient can have multiple appointments.
9. A patient can only be admitted to one room at a time
10. Each department must have at least one room, and each room belongs to exactly one department
11. An insurance agency can have multiple patients.
12. A patient can have one insurance agency.
13. A department can have many employees.
14. An employee only has one department.
15. An invoice has one employee.
16. An employee can have multiple invoices.
17. An invoice has one patient.
18. A patient can have many invoices.
19. An invoice can have one insurance agency.
20. An insurance agency can have many invoices.
21. An invoice can have one appointment.
22. An appointment can have one invoice.

Part 3

For better visibility: Lucid Chart



EMPLOYEE		
PK	EMP_ID	INT
	EMP_LNAME	STR
	EMP_FNAME	STR
FK1	POS_ID	INT
FK2	DEP_ID	INT
	EMP_HIREDATE	DATE
	EMP_DOB	DATE
	EMP_PNUM	STR
	EMP_EMAIL	STR
	EMP_ADDRESS	STR

DEPARTMENT		
PK	DEP_ID	INT
	DEP_NAME	STR
	DEP_ADDRESS	STR
	DEP_FUNDS	DOUBLE

POSITION		
PK	POS_ID	INT
FK1	DEP_ID	Type
	POS_NAME	STR
	POS_DESCRIPT	STR
	POS_COUNT	INT

PATIENT		
PK	PAT_ID	INT
	PAT_LNAME	STR
	PAT_FNAME	STR
FK1	EMP_ID	INT
FK2	INSUR_ID	INT
	PAT_DOB	DATE
	PAT_PNUM	STR
	PAT_EMAIL	STR
	PAT_ADDRESS	STR

APPOINTMENT		
PK	APP_ID	INT
FK1	PAT_ID	INT
FK2	EMP_ID	INT
FK3	INV_ID	INT
FK4	ROOM_ID	INT
	APP_REASON	STR
	APP_OUTCOME	STR
	Field	DATE

INSUR_AGENCY		
PK	INSUR_ID	INT
	INSUR_NAME	STR
	INSUR_PNUM	STR
	INSUR_EMAIL	STR
	INSUR_ADDRESS	STR

PRESCIPTIONS		
PK	PRE_ID	INT
FK1	PAT_ID	INT
FK2	EMP_ID	INT
	PRE_NAME	STR
	DOS_MG	INT

OPERATIONS		
PK	OP_ID	INT
FK1	PAT_ID	INT
FK2	EMP_ID	INT
FK3	ROOM_ID	INT
	OP_TYPE	STR
	OP_STATUS	STR
	OP_DATE	DATE

INVOICE		
PK	INV_ID	INT
FK1	INSUR_ID	INT
	INV_COST	DOUBLE
	INV_DATE	DATE
	INV_DESCRIPT	STR

ROOM		
PK	ROOM_ID	INT
FK1	DEP_ID	INT
	ROOM_STAT	STR
	ROOM_TYPE	STR
	ROOM_CAP	INT

Deliverable 2

Q1.

1. EMPLOYEE

Primary Key: EMP_ID
Attributes:
EMP_LNAME, EMP_FNAME, POS_ID, DEP_ID, EMP_HIREDATE, EMP_DOB, EMP_PNUM, EMP_EMAIL, EMP_ADDRESS

- No partial dependencies EMP_ID is the only PK.
- No transitive dependency

EMPLOYEE is in 3NF

2. POSITION

Primary Key: POS_ID
Attributes: DEP_ID, POS_NAME, POS_DESCRIPT, POS_COUNT

- No partial dependencies (single PK).
- No transitive dependency

POSITION is in 3NF

3. DEPARTMENT

Primary Key: DEP_ID
Attributes: DEP_NAME, DEP_ADDRESS, DEP_FUNDS

- No partial or transitive dependencies.

4. PATIENT

Primary Key: PAT_ID
Attributes:
PAT_LNAME, PAT_FNAME, EMP_ID, INSUR_ID, PAT_DOB, PAT_PNUM, PAT_EMAIL, PAT_ADDRESS

- No partial dependencies (single PK).
- No transitive dependency

PATIENT is in 3NF

INSUR_AGENCY

Primary Key: INSUR_ID
Attributes: INSUR_NAME, INSUR_PNUM, INSUR_EMAIL, INSUR_ADDRESS

- No partial or transitive dependencies.

INSUR_AGENCY is in 3NF

PRESCRIPTIONS

Primary Key: PRE_ID
Attributes: PAT_ID, EMP_ID, PRE_NAME, DOS_MG

- No partial dependencies.
- No transitive dependencies

PRESCRIPTIONS is in 3NF

OPERATIONS

Primary Key: OP_ID
Attributes: PAT_ID, EMP_ID, ROOM_ID, OP_TYPE, OP_STATUS, OP_DATE

- No partial dependencies.
- No transitive dependencies.

ROOM

Primary Key: ROOM_ID
Attributes: DEP_ID, ROOM_STAT, ROOM_TYPE, ROOM_CAP

- No partial dependencies.
- No transitive dependencies.

ROOM is in 3NF

APPOINTMENT

Primary Key: APP_ID
Attributes: PAT_ID, EMP_ID, INV_ID, ROOM_ID, APP_REASON, APP_OUTCOME, DATE

- No partial dependencies.
- No transitive dependencies.

APPOINTMENT is in 3NF

INVOICE

Primary Key: INV_ID
Attributes: INSUR_ID, EMP_ID, PAT_ID, INV_COST, INV_DATE

- No partial dependencies.

- No transitive dependency

Dependency Diagram

https://lucid.app/lucidchart/5fc9590b-28e4-489c-b211-a543050abd89/edit?viewport_loc=-69%2C-515%2C2537%2C1213%2C0_0&invitationId=inv_146ba88b-7221-49ac-9a5f-aaed7683121a

Data Dictionary

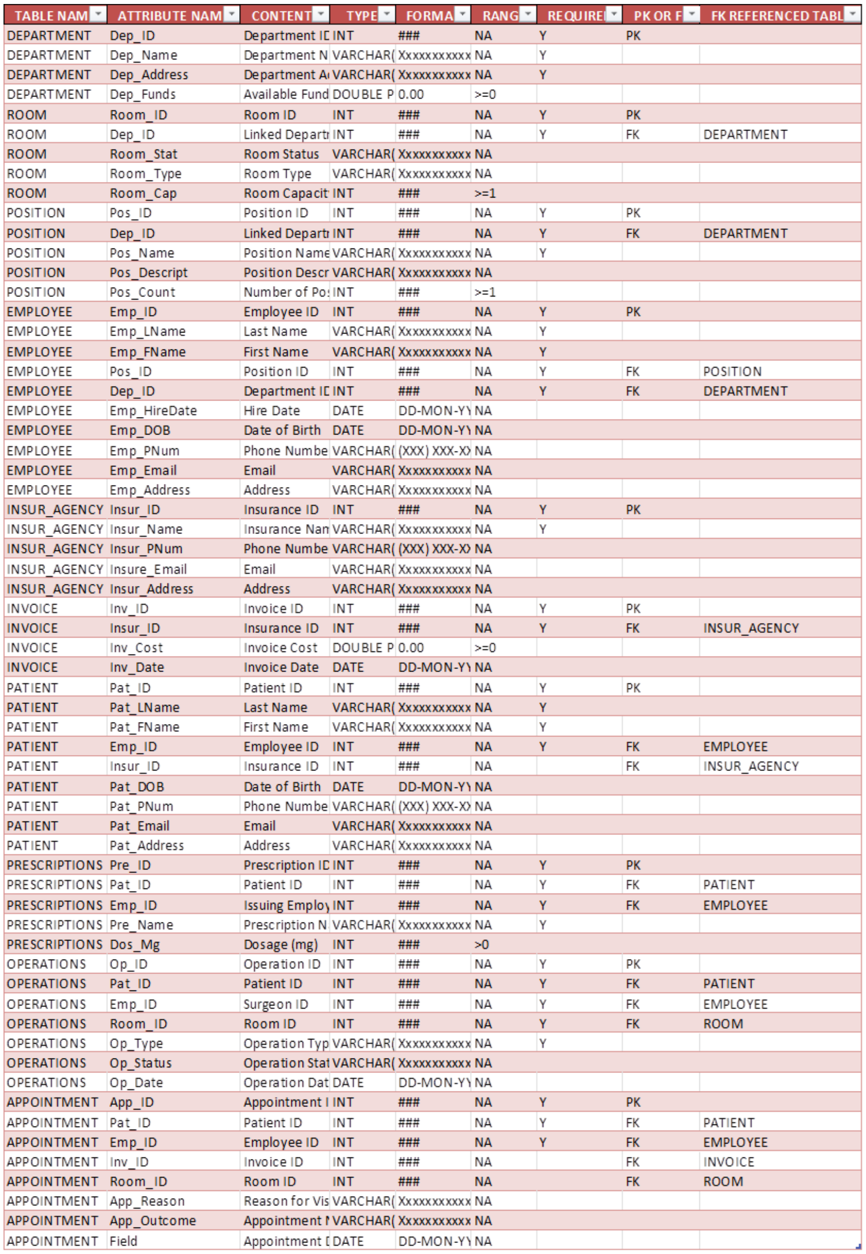
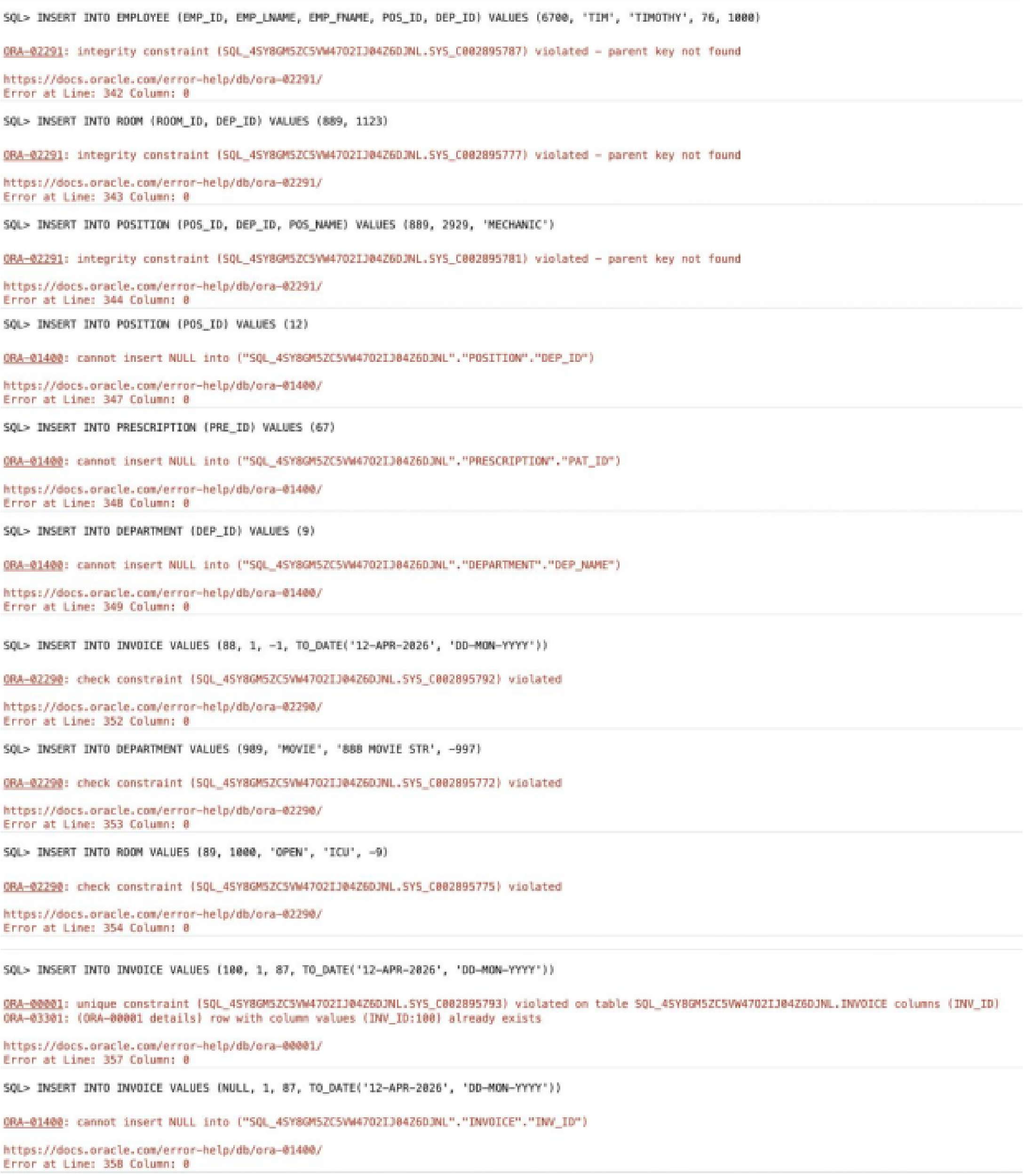


TABLE NAM	ATTRIBUTE NAM	CONTENT	TYPE	FORMA	RANG	REQUIRE	PK OR F	FK REFERENCED TABL
DEPARTMENT	Dep_ID	Department I	INT	###	NA	Y	PK	
DEPARTMENT	Dep_Name	Department N	VARCHAR(	Xxxxxxxxxxx	NA	Y		
DEPARTMENT	Dep_Address	Department A	VARCHAR(	Xxxxxxxxxxx	NA	Y		
DEPARTMENT	Dep_Funds	Available Fund	DOUBLE P	0.00	>=0			
ROOM	Room_ID	Room ID	INT	###	NA	Y	PK	
ROOM	Dep_ID	Linked Depart	INT	###	NA	Y	FK	DEPARTMENT
ROOM	Room_Stat	Room Status	VARCHAR(	Xxxxxxxxxxx	NA			
ROOM	Room_Type	Room Type	VARCHAR(	Xxxxxxxxxxx	NA			
ROOM	Room_Cap	Room Capacit	INT	###	>=1			
POSITION	Pos_ID	Position ID	INT	###	NA	Y	PK	
POSITION	Dep_ID	Linked Depart	INT	###	NA	Y	FK	DEPARTMENT
POSITION	Pos_Name	Position Name	VARCHAR(	Xxxxxxxxxxx	NA	Y		
POSITION	Pos_Descript	Position Descr	VARCHAR(	Xxxxxxxxxxx	NA			
POSITION	Pos_Count	Number of Po	INT	###	>=1			
EMPLOYEE	Emp_ID	Employee ID	INT	###	NA	Y	PK	
EMPLOYEE	Emp_LName	Last Name	VARCHAR(	Xxxxxxxxxxx	NA	Y		
EMPLOYEE	Emp_FName	First Name	VARCHAR(	Xxxxxxxxxxx	NA	Y		
EMPLOYEE	Pos_ID	Position ID	INT	###	NA	Y	FK	POSITION
EMPLOYEE	Dep_ID	Department I	INT	###	NA	Y	FK	DEPARTMENT
EMPLOYEE	Emp_HireDate	Hire Date	DATE	DD-MON-Y	NA			
EMPLOYEE	Emp_DOB	Date of Birth	DATE	DD-MON-Y	NA			
EMPLOYEE	Emp_PNum	Phone Numbe	VARCHAR(	(XXX) XXX-X	NA			
EMPLOYEE	Emp_Email	Email	VARCHAR(	Xxxxxxxxxxx	NA			
EMPLOYEE	Emp_Address	Address	VARCHAR(	Xxxxxxxxxxx	NA			
INSUR_AGENCY	Insur_ID	Insurance ID	INT	###	NA	Y	PK	
INSUR_AGENCY	Insur_Name	Insurance Nan	VARCHAR(	Xxxxxxxxxxx	NA	Y		
INSUR_AGENCY	Insur_PNum	Phone Numbe	VARCHAR(	(XXX) XXX-X	NA			
INSUR_AGENCY	Insure_Email	Email	VARCHAR(	Xxxxxxxxxxx	NA			
INSUR_AGENCY	Insur_Address	Address	VARCHAR(	Xxxxxxxxxxx	NA			
INVOICE	Inv_ID	Invoice ID	INT	###	NA	Y	PK	
INVOICE	Insur_ID	Insurance ID	INT	###	NA	Y	FK	INSUR_AGENCY
INVOICE	Inv_Cost	Invoice Cost	DOUBLE P	0.00	>=0			
INVOICE	Inv_Date	Invoice Date	DATE	DD-MON-Y	NA			
PATIENT	Pat_ID	Patient ID	INT	###	NA	Y	PK	
PATIENT	Pat_LName	Last Name	VARCHAR(	Xxxxxxxxxxx	NA	Y		
PATIENT	Pat_FName	First Name	VARCHAR(	Xxxxxxxxxxx	NA	Y		
PATIENT	Emp_ID	Employee ID	INT	###	NA	Y	FK	EMPLOYEE
PATIENT	Insur_ID	Insurance ID	INT	###	NA		FK	INSUR_AGENCY
PATIENT	Pat_DOB	Date of Birth	DATE	DD-MON-Y	NA			
PATIENT	Pat_PNum	Phone Numbe	VARCHAR(	(XXX) XXX-X	NA			
PATIENT	Pat_Email	Email	VARCHAR(	Xxxxxxxxxxx	NA			
PATIENT	Pat_Address	Address	VARCHAR(	Xxxxxxxxxxx	NA			
PRESCRIPTIONS	Pre_ID	Prescription I	INT	###	NA	Y	PK	
PRESCRIPTIONS	Pat_ID	Patient ID	INT	###	NA	Y	FK	PATIENT
PRESCRIPTIONS	Emp_ID	Issuing Employ	INT	###	NA	Y	FK	EMPLOYEE
PRESCRIPTIONS	Pre_Name	Prescription N	VARCHAR(	Xxxxxxxxxxx	NA	Y		
PRESCRIPTIONS	Dos_Mg	Dosage (mg)	INT	###	>0			
OPERATIONS	Op_ID	Operation ID	INT	###	NA	Y	PK	
OPERATIONS	Pat_ID	Patient ID	INT	###	NA	Y	FK	PATIENT
OPERATIONS	Emp_ID	Surgeon ID	INT	###	NA	Y	FK	EMPLOYEE
OPERATIONS	Room_ID	Room ID	INT	###	NA	Y	FK	ROOM
OPERATIONS	Op_Type	Operation Typ	VARCHAR(	Xxxxxxxxxxx	NA	Y		
OPERATIONS	Op_Status	Operation Stat	VARCHAR(	Xxxxxxxxxxx	NA			
OPERATIONS	Op_Date	Operation Dat	DATE	DD-MON-Y	NA			
APPOINTMENT	App_ID	Appointment I	INT	###	NA	Y	PK	
APPOINTMENT	Pat_ID	Patient ID	INT	###	NA	Y	FK	PATIENT
APPOINTMENT	Emp_ID	Employee ID	INT	###	NA	Y	FK	EMPLOYEE
APPOINTMENT	Inv_ID	Invoice ID	INT	###	NA		FK	INVOICE
APPOINTMENT	Room_ID	Room ID	INT	###	NA		FK	ROOM
APPOINTMENT	App_Reason	Reason for Vis	VARCHAR(	Xxxxxxxxxxx	NA			
APPOINTMENT	App_Outcome	Appointment	VARCHAR(	Xxxxxxxxxxx	NA			
APPOINTMENT	Field	Appointment	DATE	DD-MON-Y	NA			

Deliverable 3

Natasha Linares

Jacob Kitchens



```
SQL> INSERT INTO EMPLOYEE (EMP_ID, EMP_LNAME, EMP_FNAME, POS_ID, DEP_ID) VALUES (6700, 'TIM', 'TIMOTHY', 76, 1000)

ORA-02291: integrity constraint (SQL_4SY8GM5ZC5VW47O2IJ04Z6DJNL.SYS_C002895787) violated - parent key not found

https://docs.oracle.com/error-help/db/ora-02291/
Error at Line: 342 Column: 0

SQL> INSERT INTO ROOM (ROOM_ID, DEP_ID) VALUES (889, 1123)

ORA-02291: integrity constraint (SQL_4SY8GM5ZC5VW47O2IJ04Z6DJNL.SYS_C002895777) violated - parent key not found

https://docs.oracle.com/error-help/db/ora-02291/
Error at Line: 343 Column: 0

SQL> INSERT INTO POSITION (POS_ID, DEP_ID, POS_NAME) VALUES (889, 2929, 'MECHANIC')

ORA-02291: integrity constraint (SQL_4SY8GM5ZC5VW47O2IJ04Z6DJNL.SYS_C002895781) violated - parent key not found

https://docs.oracle.com/error-help/db/ora-02291/
Error at Line: 344 Column: 0

SQL> INSERT INTO POSITION (POS_ID) VALUES (12)

ORA-01400: cannot insert NULL into ("SQL_4SY8GM5ZC5VW47O2IJ04Z6DJNL"."POSITION"."DEP_ID")

https://docs.oracle.com/error-help/db/ora-01400/
Error at Line: 347 Column: 0

SQL> INSERT INTO PRESCRIPTION (PRE_ID) VALUES (67)

ORA-01400: cannot insert NULL into ("SQL_4SY8GM5ZC5VW47O2IJ04Z6DJNL"."PRESCRIPTION"."PAT_ID")

https://docs.oracle.com/error-help/db/ora-01400/
Error at Line: 348 Column: 0

SQL> INSERT INTO DEPARTMENT (DEP_ID) VALUES (9)

ORA-01400: cannot insert NULL into ("SQL_4SY8GM5ZC5VW47O2IJ04Z6DJNL"."DEPARTMENT"."DEP_NAME")

https://docs.oracle.com/error-help/db/ora-01400/
Error at Line: 349 Column: 0

SQL> INSERT INTO INVOICE VALUES (88, 1, -1, TO_DATE('12-APR-2026', 'DD-MON-YYYY'))

ORA-02290: check constraint (SQL_4SY8GM5ZC5VW47O2IJ04Z6DJNL.SYS_C002895792) violated

https://docs.oracle.com/error-help/db/ora-02290/
Error at Line: 352 Column: 0

SQL> INSERT INTO DEPARTMENT VALUES (909, 'MOVIE', '888 MOVIE STR', -997)

ORA-02290: check constraint (SQL_4SY8GM5ZC5VW47O2IJ04Z6DJNL.SYS_C002895772) violated

https://docs.oracle.com/error-help/db/ora-02290/
Error at Line: 353 Column: 0

SQL> INSERT INTO ROOM VALUES (89, 1000, 'OPEN', 'ICU', -9)

ORA-02290: check constraint (SQL_4SY8GM5ZC5VW47O2IJ04Z6DJNL.SYS_C002895775) violated

https://docs.oracle.com/error-help/db/ora-02290/
Error at Line: 354 Column: 0

SQL> INSERT INTO INVOICE VALUES (100, 1, 87, TO_DATE('12-APR-2026', 'DD-MON-YYYY'))

ORA-00001: unique constraint (SQL_4SY8GM5ZC5VW47O2IJ04Z6DJNL.SYS_C002895793) violated on table SQL_4SY8GM5ZC5VW47O2IJ04Z6DJNL.INVOICE columns (INV_ID)
ORA-03301: (ORA-00001 details) row with column values (INV_ID:100) already exists

https://docs.oracle.com/error-help/db/ora-00001/
Error at Line: 357 Column: 0

SQL> INSERT INTO INVOICE VALUES (NULL, 1, 87, TO_DATE('12-APR-2026', 'DD-MON-YYYY'))

ORA-01400: cannot insert NULL into ("SQL_4SY8GM5ZC5VW47O2IJ04Z6DJNL"."INVOICE"."INV_ID")

https://docs.oracle.com/error-help/db/ora-01400/
Error at Line: 358 Column: 0
```

Deliverable 4

Natasha Linares

Jacob Kitchens

Deliverable 4

Queries:

1. **Display funds from departments, total income of employees**: SELECT SUM(D.DEP_FUNDS) FUNDS,
   SUM(I.INV_COST) INCOME
   FROM DEPARTMENT D
   FULL JOIN EMPLOYEE E ON E.DEP_ID = D.DEP_ID
   FULL JOIN APPOINTMENT A ON A.EMP_ID = E.EMP_ID
   FULL JOIN INVOICE I ON I.INV_ID = A.INV_ID;

2. **Count number of patients for an employee:** SELECT E.EMP_ID, COUNT(P.PAT_ID) EMP_PATIENTS
   FROM EMPLOYEE E
   JOIN PATIENT P ON P.EMP_ID = E.EMP_ID
   GROUP BY E.EMP_ID;

3. **Display patient's id, first and last name, that have an assigned doctor.** SELECT PAT_ID, PAT_LNAME, PAT_FNAME FROM PATIENT join employee on PATIENT.EMP_ID=employee.EMP_ID where employee.DEP_ID in(SELECT DEP_ID FROM POSITION WHERE POS_NAME = 'DOCTOR');

4. **Find number of employees per department:** SELECT d.DEP_ID, d.DEP_NAME, COUNT(e.EMP_ID) AS EMP_COUNT FROM DEPARTMENT d JOIN EMPLOYEE e ON d.DEP_ID = e.DEP_ID GROUP BY d.DEP_ID, d.DEP_NAME;

5. **Display departments with more than 2 employees hired after 1980:** SELECT d.DEP_ID, d.DEP_NAME, COUNT(e.EMP_ID) AS RECENT_HIRES FROM DEPARTMENT d JOIN EMPLOYEE e ON d.DEP_ID = e.DEP_ID WHERE e.EMP_HIREDATE > TO_DATE('01-JAN-1980', 'DD-MON-YYYY') GROUP BY d.DEP_ID, d.DEP_NAME HAVING COUNT(e.EMP_ID) > 2;

6. **Average invoice of insurance company:** SELECT i.INSUR_NAME, AVG(inv.INV_COST) AS AVG_INVOICE FROM INSUR_AGENCY i JOIN INVOICE inv ON i.INSUR_ID = inv.INSUR_ID GROUP BY i.INSUR_NAME;

7. **<u>List patients whos invoice is above average:</u>** SELECT p.PAT_ID, p.PAT_LNAME, p.PAT_FNAME, inv.INV_COST FROM PATIENT p JOIN INVOICE inv ON p.INSUR_ID = inv.INSUR_ID WHERE inv.INV_COST > (SELECT AVG(INV_COST) FROM INVOICE);

8. **<u>Employees who have prescribed more than 3 prescriptions:</u>** SELECT e.EMP_ID, e.EMP_FNAME, e.EMP_LNAME, COUNT(pr.PRE_ID) AS PRESCRIPTIONS FROM EMPLOYEE e JOIN PRESCRIPTION pr ON e.EMP_ID = pr.EMP_ID GROUP BY e.EMP_ID, e.EMP_FNAME, e.EMP_LNAME HAVING COUNT(pr.PRE_ID) > 3;

9. **<u>Employees who have operations on patients with invoices above $500:</u>**SELECT DISTINCT e.EMP_ID, e.EMP_FNAME, e.EMP_LNAME FROM EMPLOYEE e JOIN OPERATION o ON e.EMP_ID = o.EMP_ID JOIN PATIENT p ON o.PAT_ID = p.PAT_ID WHERE EXISTS (SELECT 1 FROM INVOICE inv WHERE inv.INSUR_ID = p.INSUR_ID AND inv.INV_COST > 500);

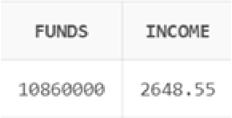
10. **<u>Number of appointments per employee and department:</u>** SELECT e.EMP_ID, e.EMP_FNAME, e.EMP_LNAME, d.DEP_NAME, COUNT(a.APP_ID) AS NUM_APPOINTMENTS FROM EMPLOYEE e JOIN DEPARTMENT d ON e.DEP_ID = d.DEP_ID JOIN APPOINTMENT a ON e.EMP_ID = a.EMP_ID GROUP BY e.EMP_ID, e.EMP_FNAME, e.EMP_LNAME, d.DEP_NAME;

11. **<u>Employees with more than 2 operations on patients with invoices over $100:</u>**SELECT e.EMP_ID, e.EMP_FNAME, e.EMP_LNAME, COUNT(o.OP_ID) AS NUM_OPERATIONS FROM EMPLOYEE e JOIN OPERATION o ON e.EMP_ID = o.EMP_ID JOIN PATIENT p ON o.PAT_ID = p.PAT_ID JOIN INVOICE inv ON p.INSUR_ID = inv.INSUR_ID WHERE inv.INV_COST > 100 GROUP BY e.EMP_ID, e.EMP_FNAME, e.EMP_LNAME HAVING COUNT(o.OP_ID) > 2;

12. **<u>Count how many appointments each employee has, but only for employees who have more than 2 patients:</u>** SELECT e.EMP_ID, e.EMP_FNAME, e.EMP_LNAME, COUNT(o.OP_ID) AS NUM_OPERATIONS FROM EMPLOYEE e JOIN OPERATION o ON e.EMP_ID = o.EMP_ID JOIN PATIENT p ON o.PAT_ID = p.PAT_ID JOIN INVOICE inv ON p.INSUR_ID = inv.INSUR_ID WHERE inv.INV_COST > 100 GROUP BY e.EMP_ID, e.EMP_FNAME, e.EMP_LNAME HAVING COUNT(o.OP_ID) > 2;

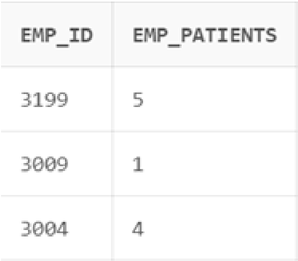
Outputs:

1.



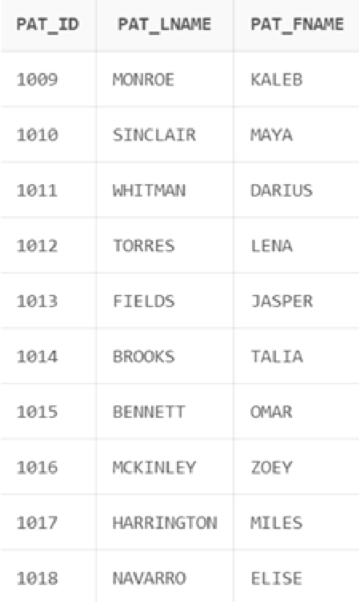
FUNDS	INCOME
10860000	2648.55

2.



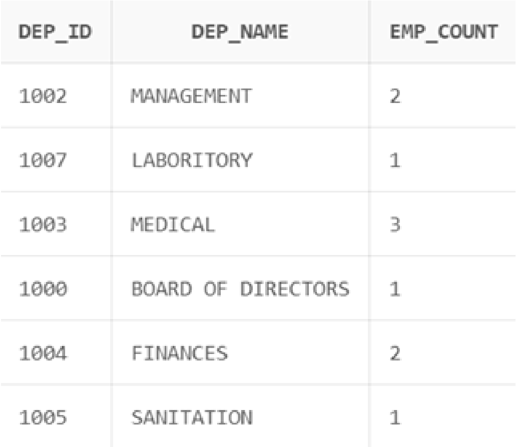
EMP_ID	EMP_PATIENTS
3199	5
3009	1
3004	4

3.



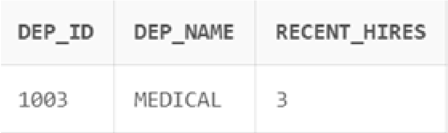
PAT_ID	PAT_LNAME	PAT_FNAME
1009	MONROE	KALEB
1010	SINCLAIR	MAYA
1011	WHITMAN	DARIUS
1012	TORRES	LENA
1013	FIELDS	JASPER
1014	BROOKS	TALIA
1015	BENNETT	OMAR
1016	MCKINLEY	ZOEY
1017	HARRINGTON	MILES
1018	NAVARRO	ELISE

4.



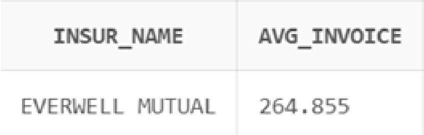
DEP_ID	DEP_NAME	EMP_COUNT
1002	MANAGEMENT	2
1007	LABORITORY	1
1003	MEDICAL	3
1000	BOARD OF DIRECTORS	1
1004	FINANCES	2
1005	SANITATION	1

5.



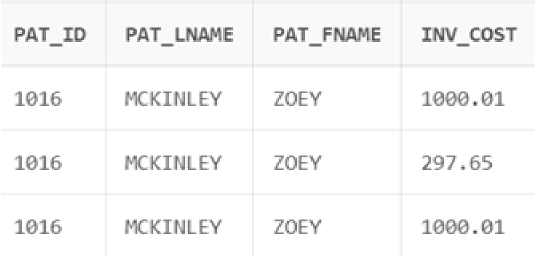
DEP_ID	DEP_NAME	RECENT_HIRES
1003	MEDICAL	3

6.



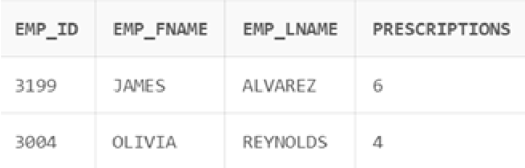
INSUR_NAME	AVG_INVOICE
EVERWELL MUTUAL	264.855

7.



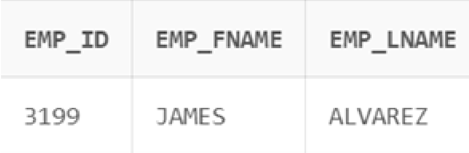
PAT_ID	PAT_LNAME	PAT_FNAME	INV_COST
1016	MCKINLEY	ZOEY	1000.01
1016	MCKINLEY	ZOEY	297.65
1016	MCKINLEY	ZOEY	1000.01

8.



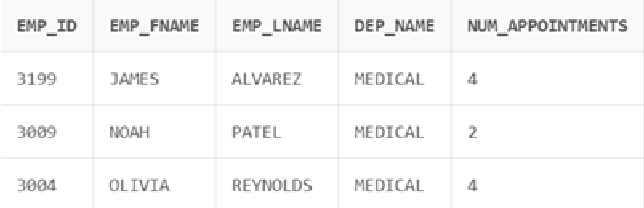
EMP_ID	EMP_FNAME	EMP_LNAME	PRESCRIPTIONS
3199	JAMES	ALVAREZ	6
3004	OLIVIA	REYNOLDS	4

9.



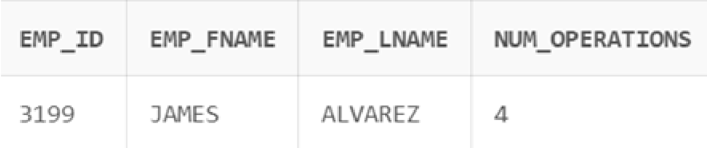
EMP_ID	EMP_FNAME	EMP_LNAME
3199	JAMES	ALVAREZ

10.



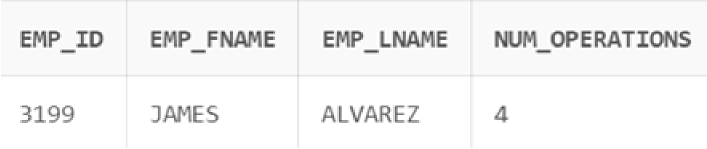
EMP_ID	EMP_FNAME	EMP_LNAME	DEP_NAME	NUM_APPOINTMENTS
3199	JAMES	ALVAREZ	MEDICAL	4
3009	NOAH	PATEL	MEDICAL	2
3004	OLIVIA	REYNOLDS	MEDICAL	4

11.



EMP_ID	EMP_FNAME	EMP_LNAME	NUM_OPERATIONS
3199	JAMES	ALVAREZ	4

12.



EMP_ID	EMP_FNAME	EMP_LNAME	NUM_OPERATIONS
3199	JAMES	ALVAREZ	4

Deliverable 5

Natasha Linares

Jacob Kitchens

Outputs

Stored Procedures and Functions:

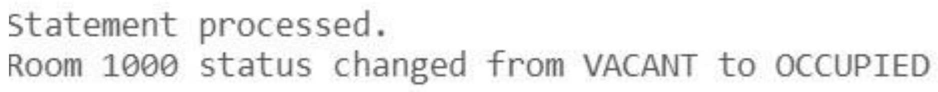
1. Procedure: Update Room Status
   Ex.

   BEGIN

     update_room_status(1001, 'OCCUPIED');

   END;

   /



```
Statement processed.
Room 1000 status changed from VACANT to OCCUPIED
```

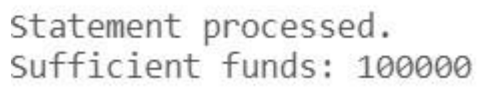
2. Procedure: Check Employee Department Funds
   Ex.

   BEGIN

     check_funds(1000);

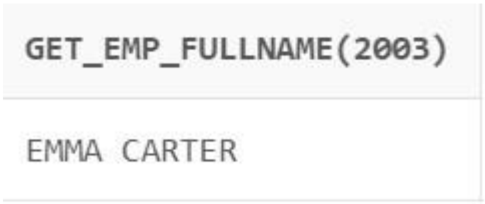
   END;

   /



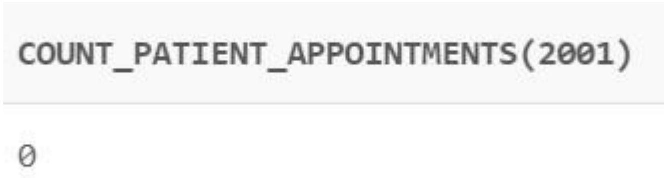
```
Statement processed.
Sufficient funds: 100000
```

3. Function: Get Employee Full Name
   SELECT get_emp_fullname(2003) FROM dual;



GET_EMP_FULLNAME(2003)
EMMA CARTER

4. Function: Count Appointments for Patient
   SELECT count_patient_appointments(2001) FROM dual;

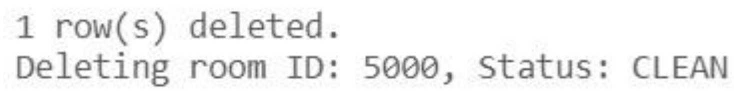


COUNT_PATIENT_APPOINTMENTS(2001)
0

Triggers:

1. Delete Room

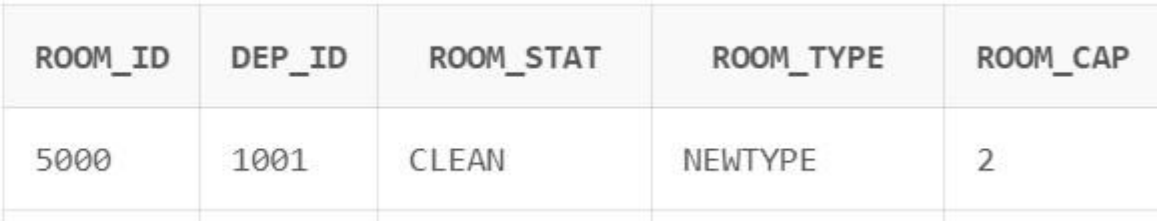
DELETE FROM ROOM WHERE ROOM_ID = 5000;



```
1 row(s) deleted.
Deleting room ID: 5000, Status: CLEAN
```

2. Insert new room

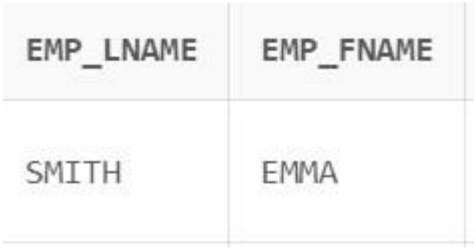
INSERT INTO ROOM (ROOM_ID, DEP_ID, ROOM_TYPE, ROOM_CAP) VALUES (5000, 1001, 'NEWTYPE', 2);



ROOM_ID	DEP_ID	ROOM_STAT	ROOM_TYPE	ROOM_CAP
5000	1001	CLEAN	NEWTYPE	2

3. Update employee last name

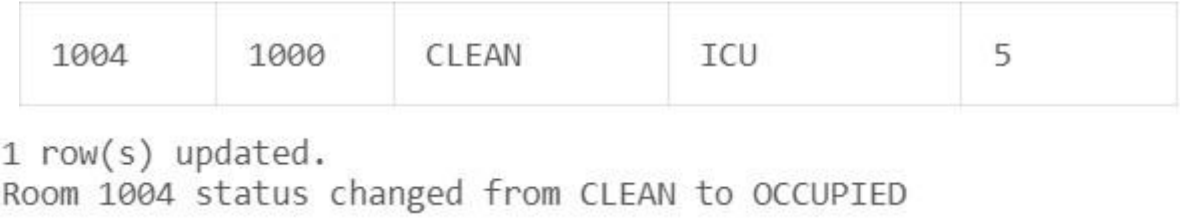
UPDATE EMPLOYEE SET EMP_LNAME = 'SMITH' WHERE EMP_ID = 1001;



EMP_LNAME	EMP_FNAME
SMITH	EMMA

4. Change room status:

SET SERVEROUTPUT ON; UPDATE ROOM SET ROOM_STAT = 'OCCUPIED' WHERE ROOM_ID=1004;



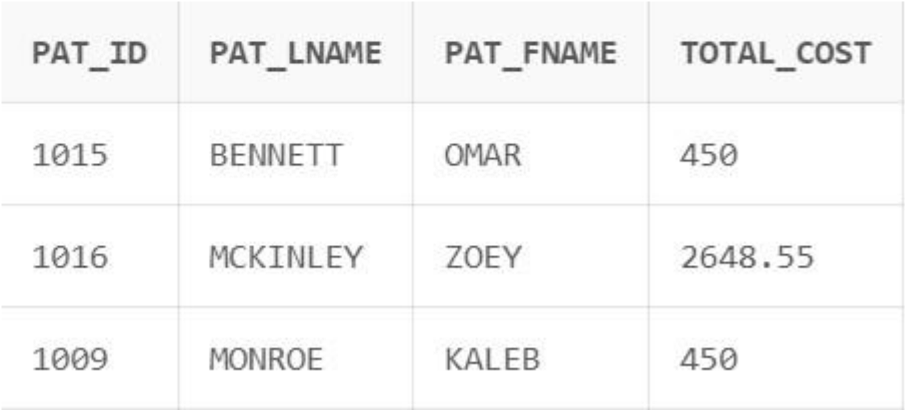
1004	1000	CLEAN	ICU	5

```
1 row(s) updated.
Room 1004 status changed from CLEAN to OCCUPIED
```

Views:

1. **vw_patient_invoice_summary** – Shows total invoice cost per patient.

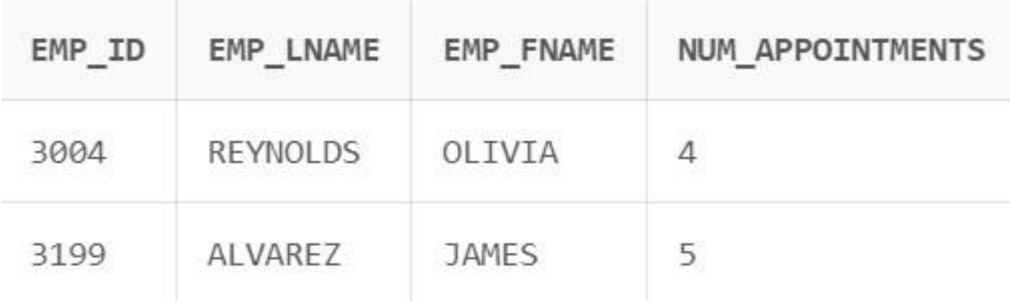
SELECT * FROM vw_patient_invoice_summary;



PAT_ID	PAT_LNAME	PAT_FNAME	TOTAL_COST
1015	BENNETT	OMAR	450
1016	MCKINLEY	ZOEY	2648.55
1009	MONROE	KALEB	450

2. **vw_busy_doctors** – Lists doctors with more than 3 appointments.

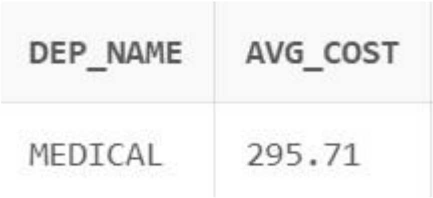
SELECT * FROM vw_busy_doctors;



EMP_ID	EMP_LNAME	EMP_FNAME	NUM_APPOINTMENTS
3004	REYNOLDS	OLIVIA	4
3199	ALVAREZ	JAMES	5

3. **vw_avg_invoice_by_department** – Shows average invoice cost by department.

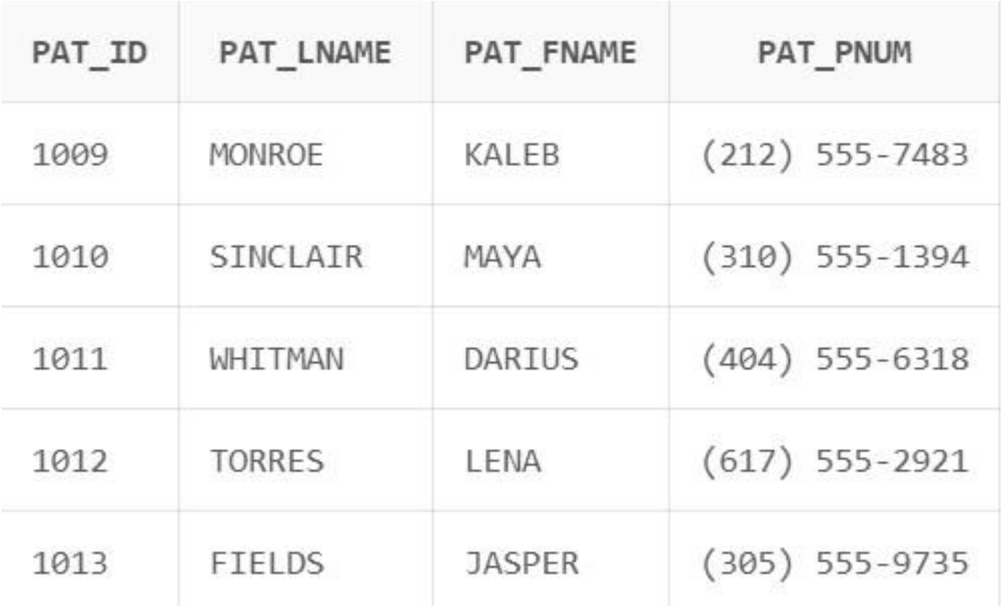
SELECT * FROM vw_avg_invoice_by_department;



DEP_NAME	AVG_COST
MEDICAL	295.71

4. **vw_patient_contact** – A simple, updatable view for patient contact info.

SELECT * FROM vw_patient_contact;



PAT_ID	PAT_LNAME	PAT_FNAME	PAT_PNUM
1009	MONROE	KALEB	(212) 555-7483
1010	SINCLAIR	MAYA	(310) 555-1394
1011	WHITMAN	DARIUS	(404) 555-6318
1012	TORRES	LENA	(617) 555-2921
1013	FIELDS	JASPER	(305) 555-9735

Deliverable 6

Jacob Kitchens

Natasha Linares

Overview

Our database is a hospital management database. The setting is that the database is for a medium sized hospital in a fictional town. The database is intended to be used by hospital staff, hospital management, patients, and insurance agencies and it contains basically all the data the hospital holds. For simplicity purposes, the database created for this project is dimmed down to contain ten rows per table, except for the rooms table that has forty rows.

Update Analysis

1)
```sql
UPDATE POSITION SET POS_DESCRIPT = 'HOSPITAL STAFF' WHERE DEP_ID IN (SELECT DEP_ID FROM ROOM WHERE ROOM_ID >= 2000 AND ROOM_ID < 3000);
```
This update updates all the descriptions of the positions in the department with room IDs in the 2000s. The 2000s are the arbitrary range I decided that represents the hospital building itself; so essentially, this puts the description of all positions located in the hospital building to “HOSPITAL STAFF”.

2)
```sql
UPDATE APPOINTMENT SET APP_REASON = 'TORN LEG LIGAMENT' WHERE PAT_ID = 1014;
```
This updates all of patient 1014’s appointments to be for a torn leg ligament.

3)
```sql
UPDATE ROOM SET ROOM_CAP = 5 WHERE ROOM_ID != 1000;
```
This update sets all rooms other than room 1000 to have a capacity of 5. This specification is in response to a previous update that updates room 1000 to have a capacity of 15.

4)
```sql
UPDATE INVOICE SET INV_COST = 15 WHERE INV_DATE >= TO_DATE('01-JUNE-2020', 'DD-MON-YYYY');
```
This sets the cost of all invoices made after June 1st, 2020 to be $15.

Constraint Violations

1)
```sql
INSERT INTO EMPLOYEE (EMP_ID, EMP_LNAME, EMP_FNAME, POS_ID, DEP_ID) VALUES (6700, 'TIM', 'TIMOTHY', 76, 1000);
```
This returns a referential integrity violation because it refers to the position ID 76, which does not exist in the position table.

2)
```sql
INSERT INTO ROOM (ROOM_ID, DEP_ID) VALUES (889, 1123);
```

This returns a referential integrity violation because it refers to the department ID 1123, which does not exist in the department table.

3)
```
INSERT INTO POSITION (POS_ID, DEP_ID, POS_NAME) VALUES (889, 2929, 'MECHANIC');
```
This returns a referential integrity violation because it refers to the department ID 2929, which does not exist in the department table.

4)
```
INSERT INTO POSITION (POS_ID) VALUES (12);
```
This returns a not null violation because the department ID field in the position table cannot be null and this insert doesn’t include a department ID.

5)
```
INSERT INTO PRESCRIPTION (PRE_ID) VALUES (67);
```
This returns a not null violation because the patient ID field in the prescription table cannot be null and this insert doesn’t include a patient ID.

6)
```
INSERT INTO DEPARTMENT (DEP_ID) VALUES (9);
```
This returns a not null violation because the department name field in the department table cannot be null and this insert doesn’t include a department name

7)
```
INSERT INTO INVOICE VALUES (88, 1, -1, TO_DATE('12-APR-2026', 'DD-MON-YYYY'));
```
This returns a check restraint violation because the invoice cost value cannot be negative, and this is trying to insert a negative 1.

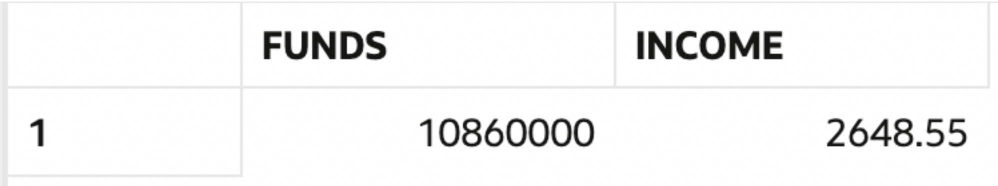
8)
```
INSERT INTO DEPARTMENT VALUES (989, 'MOVIE', '888 MOVIE STR', -997);
```
This returns a check restraint violation because the department funds attribute cannot be negative, but this tries to insert –997.

9)
```
INSERT INTO ROOM VALUES (89, 1000, 'OPEN', 'ICU', -9);
```
This returns a check restraint violation because the room capacity attribute cannot be negative, but this inserts a –9.

10)
```
INSERT INTO INVOICE VALUES (100, 1, 87, TO_DATE('12-APR-2026', 'DD-MON-YYYY'));
INSERT INTO INVOICE VALUES (NULL, 1, 87, TO_DATE('12-APR-2026', 'DD-MON-YYYY'));
```
Both of these return primary key violations. The first one returns a primary key violation because the primary key 100 already exists in the invoice table, thereby violating the unique requirement of a primary key. The second one violates the not null requirement of the primary key.

Query Results

1.
```
SELECT SUM(D.DEP_FUNDS) FUNDS, SUM(I.INV_COST) INCOME FROM DEPARTMENT D FULL JOIN EMPLOYEE
E ON E.DEP_ID = D.DEP_ID FULL JOIN APPOINTMENT A ON A.EMP_ID = E.EMP_ID FULL JOIN INVOICE I ON
I.INV_ID = A.INV_ID;
```
Prints the total funds of the hospital and the income of the hospital.

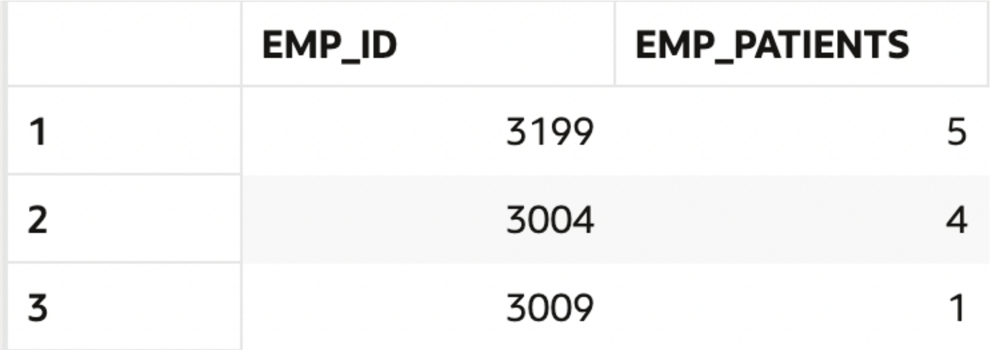


	FUNDS	INCOME
1	10860000	2648.55

2.

```
SELECT E.EMP_ID, COUNT(P.PAT_ID) EMP_PATIENTS FROM EMPLOYEE E JOIN PATIENT P ON P.EMP_ID =
E.EMP_ID GROUP BY E.EMP_ID;
```

Prints the amount of patients each doctor is appointed.

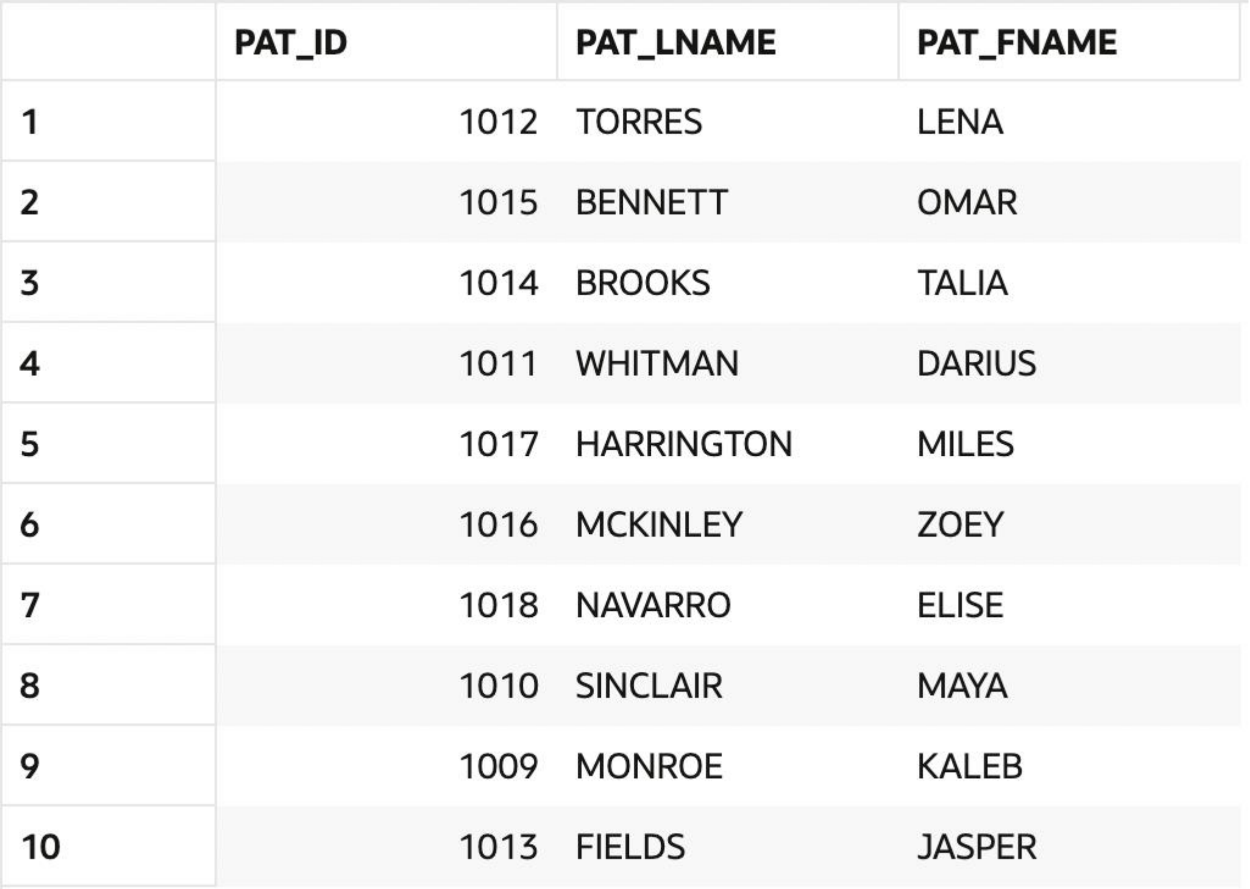


	EMP_ID	EMP_PATIENTS
1	3199	5
2	3004	4
3	3009	1

3.

```
SELECT PAT_ID, PAT_LNAME, PAT_FNAME FROM PATIENT join employee on
PATIENT.EMP_ID=employee.EMP_ID where employee.DEP_ID in(SELECT DEP_ID FROM POSITION WHERE
POS_NAME = 'DOCTOR');
```

Prints the ID and name of each patient with a doctor.

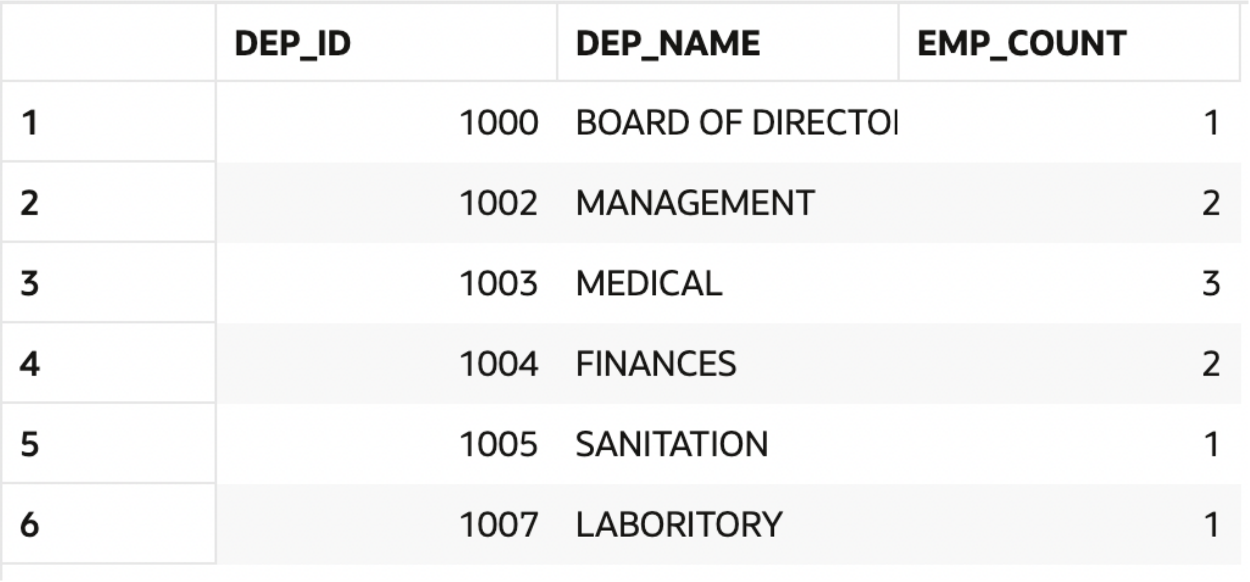


	PAT_ID	PAT_LNAME	PAT_FNAME
1	1012	TORRES	LENA
2	1015	BENNETT	OMAR
3	1014	BROOKS	TALIA
4	1011	WHITMAN	DARIUS
5	1017	HARRINGTON	MILES
6	1016	MCKINLEY	ZOEY
7	1018	NAVARRO	ELISE
8	1010	SINCLAIR	MAYA
9	1009	MONROE	KALEB
10	1013	FIELDS	JASPER

4.

```
SELECT d.DEP_ID, d.DEP_NAME, COUNT(e.EMP_ID) AS EMP_COUNT FROM DEPARTMENT d JOIN EMPLOYEE e
ON d.DEP_ID = e.DEP_ID GROUP BY d.DEP_ID, d.DEP_NAME;
```

Displays the number of employees in each department

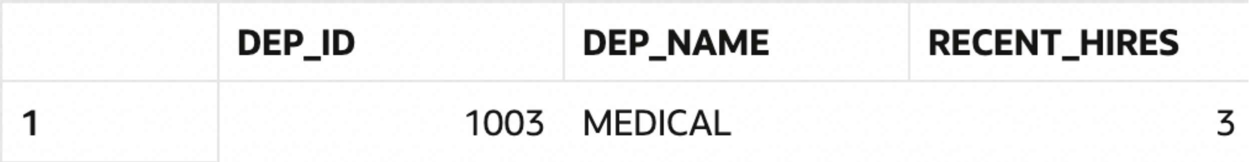


	DEP_ID	DEP_NAME	EMP_COUNT
1	1000	BOARD OF DIRECTOI	1
2	1002	MANAGEMENT	2
3	1003	MEDICAL	3
4	1004	FINANCES	2
5	1005	SANITATION	1
6	1007	LABORITORY	1

5.

```
SELECT d.DEP_ID, d.DEP_NAME, COUNT(e.EMP_ID) AS RECENT_HIRES FROM DEPARTMENT d JOIN EMPLOYEE
e ON d.DEP_ID = e.DEP_ID WHERE e.EMP_HIREDATE > TO_DATE('01-JAN-1980', 'DD-MON-YYYY') GROUP BY
d.DEP_ID, d.DEP_NAME HAVING COUNT(e.EMP_ID) > 2;
```

Display departments with more than 2 employees hired after 1980.

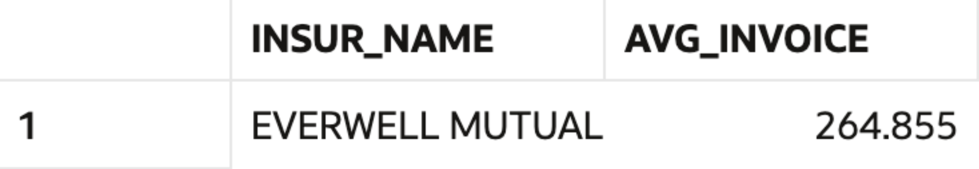


	DEP_ID	DEP_NAME	RECENT_HIRES
1	1003	MEDICAL	3

6.

```
SELECT i.INSUR_NAME, AVG(inv.INV_COST) AS AVG_INVOICE FROM INSUR_AGENCY i JOIN INVOICE inv ON
i.INSUR_ID = inv.INSUR_ID GROUP BY i.INSUR_NAME;
```

Displays the average invoice of each insurance company.

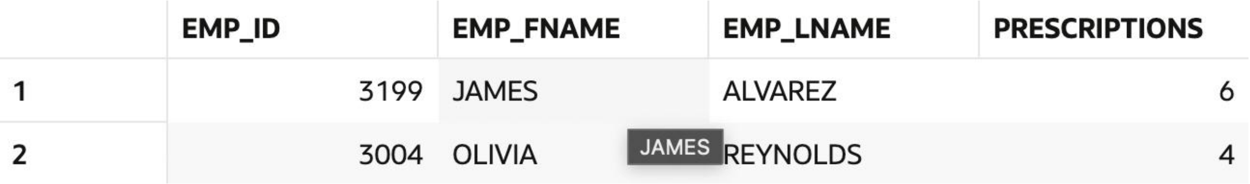


	INSUR_NAME	AVG_INVOICE
1	EVERWELL MUTUAL	264.855

7.

```
SELECT e.EMP_ID, e.EMP_FNAME, e.EMP_LNAME, COUNT(pr.PRE_ID) AS PRESCRIPTIONS FROM EMPLOYEE e
JOIN PRESCRIPTION pr ON e.EMP_ID = pr.EMP_ID GROUP BY e.EMP_ID, e.EMP_FNAME, e.EMP_LNAME HAVING
COUNT(pr.PRE_ID) > 3;
```

Shows the employees who have prescribed more than one medication.

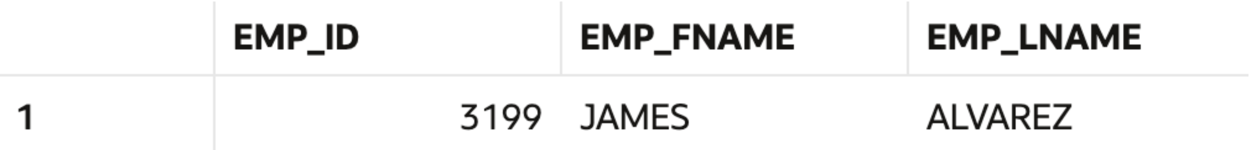


	EMP_ID	EMP_FNAME	EMP_LNAME	PRESCRIPTIONS
1	3199	JAMES	ALVAREZ	6
2	3004	OLIVIA	REYNOLDS	4

8.

```
SELECT DISTINCT e.EMP_ID, e.EMP_FNAME, e.EMP_LNAME FROM EMPLOYEE e JOIN OPERATION o ON
e.EMP_ID = o.EMP_ID JOIN PATIENT p ON o.PAT_ID = p.PAT_ID WHERE EXISTS ( SELECT 1 FROM INVOICE inv
WHERE inv.INSUR_ID = p.INSUR_ID AND inv.INV_COST > 500 );
```

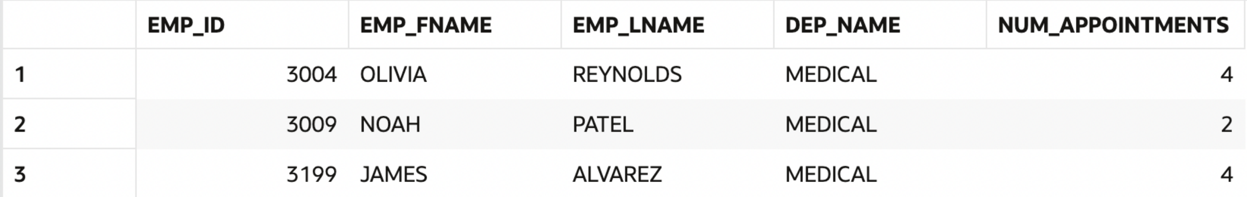
Shows employees with operations with invoices above $500.



	EMP_ID	EMP_FNAME	EMP_LNAME
1	3199	JAMES	ALVAREZ

9. SELECT e.EMP_ID, e.EMP_FNAME, e.EMP_LNAME, d.DEP_NAME, COUNT(a.APP_ID) AS NUM_APPOINTMENTS FROM EMPLOYEE e JOIN DEPARTMENT d ON e.DEP_ID = d.DEP_ID JOIN APPOINTMENT a ON e.EMP_ID = a.EMP_ID GROUP BY e.EMP_ID, e.EMP_FNAME, e.EMP_LNAME, d.DEP_NAME;

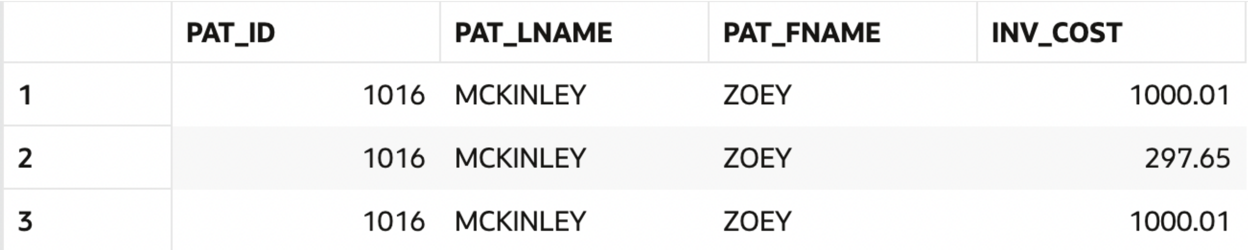
   Shows the number of appointments per employee, grouped by department.



	EMP_ID	EMP_FNAME	EMP_LNAME	DEP_NAME	NUM_APPOINTMENTS
1	3004	OLIVIA	REYNOLDS	MEDICAL	4
2	3009	NOAH	PATEL	MEDICAL	2
3	3199	JAMES	ALVAREZ	MEDICAL	4

10. SELECT p.PAT_ID, p.PAT_LNAME, p.PAT_FNAME, inv.INV_COST FROM PATIENT p JOIN INVOICE inv ON p.INSUR_ID = inv.INSUR_ID WHERE inv.INV_COST > (SELECT AVG(INV_COST) FROM INVOICE);

    Shows the patient info and invoice costs for all invoices with a cost above the average invoice cost.



	PAT_ID	PAT_LNAME	PAT_FNAME	INV_COST
1	1016	MCKINLEY	ZOEY	1000.01
2	1016	MCKINLEY	ZOEY	297.65
3	1016	MCKINLEY	ZOEY	1000.01