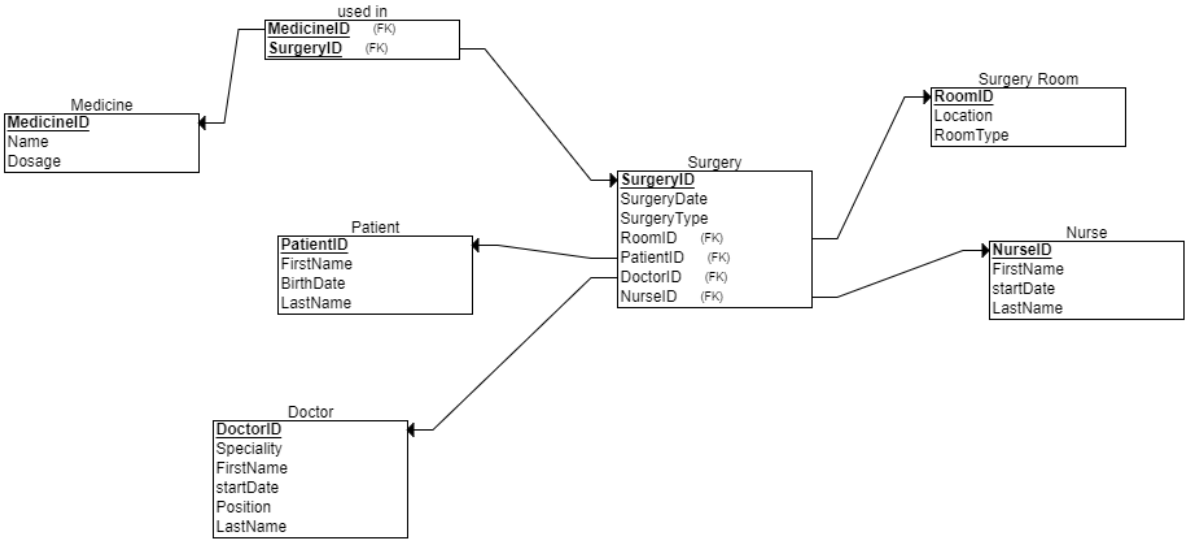
Report stage 4

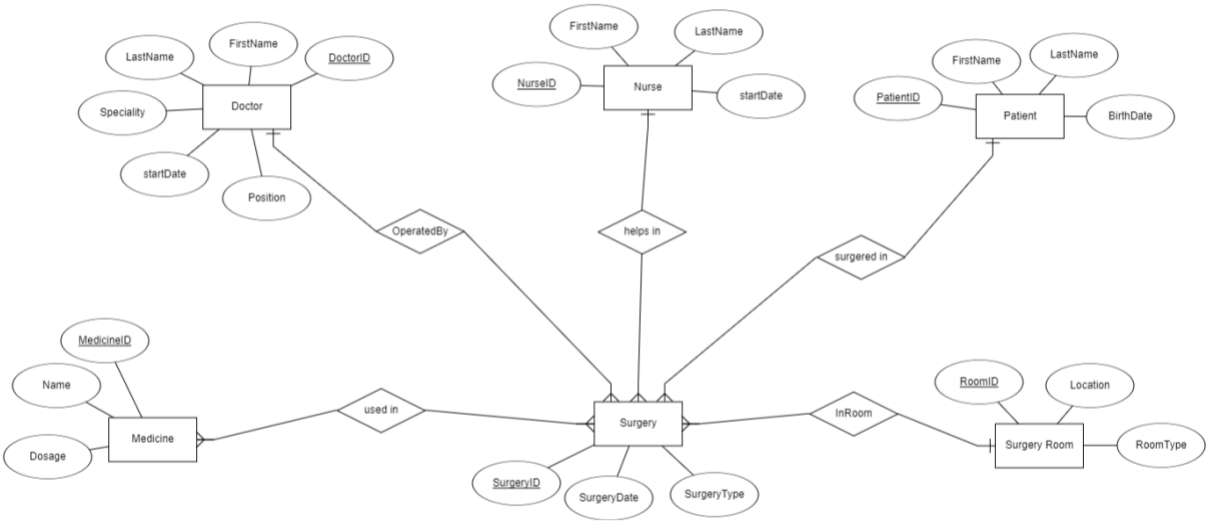
Yuval Yefet 213938905

Jeremie Tordjman 1828264

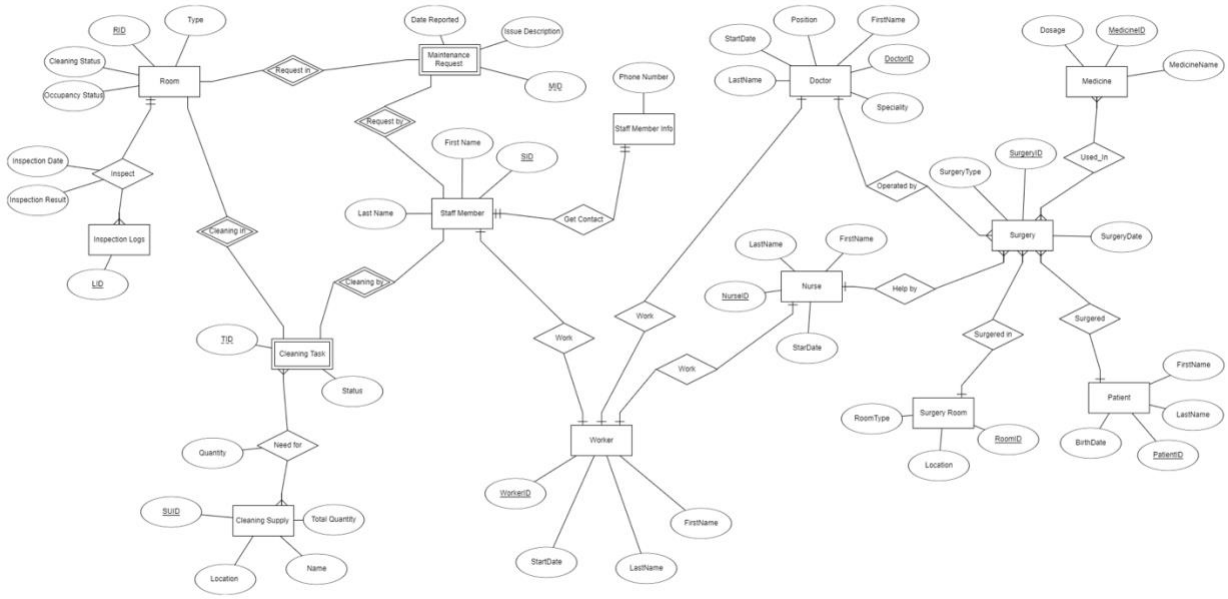
DSD created via backup received



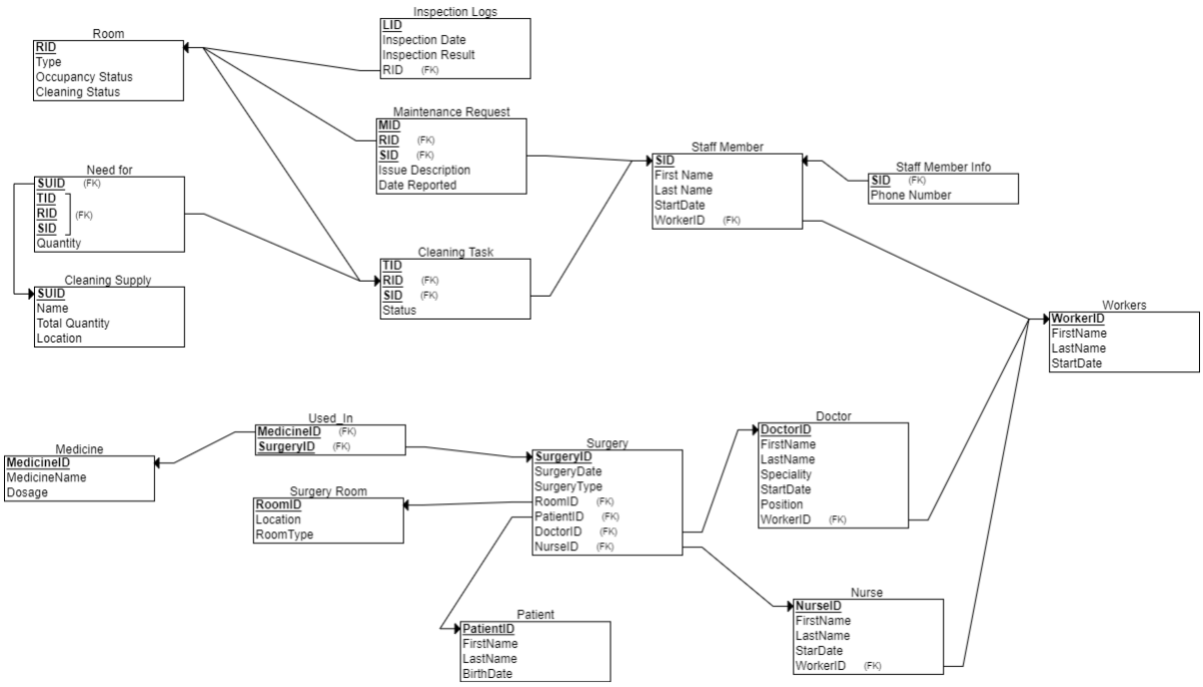
ERD created via reverse engineering



ERD Unify



DSD Unify



Integration decisions

In the DSD we received from the other team, both Doctor and Nurse had common fields. In addition, our staff member table did not have a start date field (Doctor and Nurse had), and there are also common fields with Doctor and Nurse. Thus, we have decided to create a new table named Worker which will represent all the workers in the hospital (Doctor, Nurse and Staff members), in order to have the correct structure.

First, we add the start date field to staff_member (Update_staff_member.sql):

SQL Output Statistics

```
ALTER TABLE Staff_Member
ADD StartDate DATE;

DECLARE
  v_sid Staff_Member.SID%TYPE;
  v_start_date DATE;
BEGIN
  FOR r IN (SELECT SID FROM Staff_Member) LOOP
    -- create a random date between january 2020 and january 2023
    v_start_date := TO_DATE('2020-01-01', 'YYYY-MM-DD') + ROUND(DBMS_RANDOM.VALUE(0, 1095)); -- 1095 for 3 years
    v_sid := r.SID;

    -- Update StartDate
    UPDATE Staff_Member
    SET StartDate = v_start_date
    WHERE SID = v_sid;
  END LOOP;

  COMMIT;
END;
/
```

Secondly, we created the Worker table (See integrate.sql):

Surgery_Details_View_secondQuery.sql | select * from surgery_deta ... | Integrate.sql | Update_Staff_Member.sql

SQL | Output | Statistics

```sql
CREATE TABLE Worker
(
  WorkerID INT GENERATED BY DEFAULT AS IDENTITY,
  FirstName VARCHAR2(30) NOT NULL,
  LastName VARCHAR2(30) NOT NULL,
  StartDate DATE NOT NULL,
  PRIMARY KEY (WorkerID)
);

ALTER TABLE Staff_Member
ADD WorkerID INT;

BEGIN
  FOR rec IN (SELECT FirstName, LastName, StartDate FROM Staff_Member) LOOP
    INSERT INTO Worker (FirstName, LastName, StartDate)
    VALUES (rec.FirstName, rec.LastName, rec.StartDate);
  END LOOP;
END;
/

BEGIN
  FOR rec IN (SELECT SID, FirstName, LastName, StartDate FROM Staff_Member) LOOP
    UPDATE Staff_Member
    SET WorkerID = (SELECT WorkerID FROM Worker WHERE Worker.FirstName = rec.FirstName AND Worker.LastName = rec.Last
    WHERE SID = rec.SID;
  END LOOP;
END;
/

ALTER TABLE Staff_Member
ADD CONSTRAINT fk_worker_staff_member
FOREIGN KEY (WorkerID) REFERENCES Worker(WorkerID);

ALTER TABLE Doctor
ADD WorkerID INT;

BEGIN
```

Integrate.sql

Creating the Worker Table:

- This command creates a Worker table with columns for an auto-generated WorkerID, FirstName, LastName, and StartDate. The WorkerID is the primary key, meaning it uniquely identifies each worker.

Adding WorkerID to Staff_Member Table:

- This command adds a new column called WorkerID to the Staff_Member table. This column will be used to reference the Worker table.

Populating Worker Table from Staff_Member:

- This PL/SQL block selects the first name, last name, and start date from the Staff_Member table and inserts this data into the Worker table for each record. This ensures that each staff member has a corresponding entry in the Worker table.

Updating WorkerID in Staff_Member:

- This PL/SQL block updates the WorkerID in the Staff_Member table. It finds the WorkerID in the Worker table that matches the first name, last name, and start date of each staff member and sets this WorkerID in the Staff_Member table.

Adding Foreign Key Constraint to Staff_Member:

- This command adds a foreign key constraint to the Staff_Member table. The WorkerID in Staff_Member is linked to the WorkerID in the Worker table, ensuring that every WorkerID in Staff_Member corresponds to a valid entry in the Worker table.

Adding WorkerID to Doctor Table:

- This command adds a new column called WorkerID to the Doctor table. This column will be used to reference the Worker table.

Populating Worker Table from Doctor:

- This PL/SQL block selects the first name, last name, and start date from the Doctor table and inserts this data into the Worker table for each record. This ensures that each doctor has a corresponding entry in the Worker table.

Updating WorkerID in Doctor:

- This PL/SQL block updates the WorkerID in the Doctor table. It finds the WorkerID in the Worker table that matches the first name, last name, and start date of each doctor and sets this WorkerID in the Doctor table.

Adding Foreign Key Constraint to Doctor:

- This command adds a foreign key constraint to the Doctor table. The WorkerID in Doctor is linked to the WorkerID in the Worker table, ensuring that every WorkerID in Doctor corresponds to a valid entry in the Worker table.

Adding WorkerID to Nurse Table:

- This command adds a new column called WorkerID to the Nurse table. This column will be used to reference the Worker table.

Populating Worker Table from Nurse:

- This PL/SQL block selects the first name, last name, and start date from the Nurse table and inserts this data into the Worker table for each record. This ensures that each nurse has a corresponding entry in the Worker table.

Updating WorkerID in Nurse:

- This PL/SQL block updates the WorkerID in the Nurse table. It finds the WorkerID in the Worker table that matches the first name, last name, and start date of each nurse and sets this WorkerID in the Nurse table.

Adding Foreign Key Constraint to Nurse:

- This command adds a foreign key constraint to the Nurse table. The WorkerID in Nurse is linked to the WorkerID in the Worker table, ensuring that every WorkerID in Nurse corresponds to a valid entry in the Worker table.

This first view, named StaffSupplyUsage, provides a summary of staff supply usage. It combines data from multiple tables to display staff ID, full name, phone number, supply ID, supply name, total quantity, location, and total used quantity of supplies. It uses LEFT JOIN to ensure all staff members are included, even if they haven't used supplies, and groups the results by relevant fields.

```sql
-- Create a view to track staff supply usage
CREATE VIEW StaffSupplyUsage AS
SELECT
    sm.SID,
    sm.FirstName || ' ' || sm.LastName AS Staff_Name,
    smi.Phone_Number,
    cs.SUID,
    cs.Name AS Supply_Name,
    cs.Total_Quantity,
    cs.Location,
    NVL(SUM(nf.Quantity), 0) AS Total_Used
FROM Staff_Member sm
LEFT JOIN Staff_Member_Info smi ON sm.SID = smi.SID
LEFT JOIN Cleaning_Task ct ON sm.SID = ct.SID
LEFT JOIN Need_for nf ON ct.TID = nf.TID AND ct.RID = nf.RID AND ct.SID = nf.SID
LEFT JOIN Cleaning_Supply cs ON nf.SUID = cs.SUID
GROUP BY sm.SID, sm.FirstName, sm.LastName, smi.Phone_Number, cs.SUID, cs.Name, cs.Total_Quantity, cs.Location;
```

SQL Output Statistics

```sql
select * from staffsupplyusage
```

	SID	STAFF_NAME	PHONE_NUMBER	SUID	SUPPLY_NAME	TOTAL_QUANTITY	LOCATION	TOTAL_USED
157	264	Orlando Fraser	972-530742465	123	Floor Polish	215	Storage A	18
173	365	Jared Lennox	972-599189936	255	Floor Polish	89	Storage A	18
268	223	King Strong	972-560438452	17	Floor Polish	130	Storage C	17
78	111	Sandra Def	972-541643633	360	Glass Cleaner	281	Storage A	19
241	188	Marley Duke	972-527194563	296	Grout Cleaner	383	Storage B	19
382	19	Amy Orton	972-507830103	253	Grout Cleaner	260	Storage A	19
34	127	Aimee Adler	972-565158736	386	Hand Sanitizer	327	Storage B	19
369	190	Patricia Knight	972-545404089	117	Hand Sanitizer	231	Storage A	18
83	302	Gailard Logue	972-592419900	397	Hand Sanitizer	37	Storage A	17
237	370	Leelee Nicholson	972-535786354	251	Hand Sanitizer	30	Storage C	16
76	330	Dwight Gano	972-592649343	320	Laundry Detergent	149	Storage B	16
109	167	Clarence Lee	972-533878622	174	Lemon Juice	420	Storage C	16
243	378	Luis Dunaway	972-535634204	301	Lint Roller	119	Storage B	20
284	66	Brendan Kinski	972-530470142	187	Microfiber Cloths	304	Storage A	19
60	56	Rebeka Crouse	972-513333084	127	Microfiber Cloths	409	Storage C	18
126	355	Oded Quinones	972-578745209	361	Microfiber Cloths	27	Storage B	18
215	348	Chrissie Allison	972-525940819	106	Microfiber Cloths	224	Storage A	17
33	72	Debby Newman	972-541542963	375	Microfiber Cloths	309	Storage A	19
17	152	Dom Seagal	972-575536654	115	Mop	85	Storage B	16
4	376	Anne Wong	972-535764495	33	Mop	213	Storage C	20
286	225	Don Wopat	972-552144092	205	Mop	174	Storage C	19
276	387	Chazz Neill	972-573597504	101	Oven Cleaner	43	Storage B	18
321	67	Nora Frampton	972-578634600	103	Oven Cleaner	463	Storage A	18
3	37	Jared Serbedzija	972-596483436	28	Oven Cleaner	172	Storage C	20

This query summarizes the total supplies used by each staff member. It aggregates the total amount of supplies used (Total_Supplies_Used) and provides a detailed breakdown of each supply used by the staff member (Supply_Usage_Breakdown). The results are filtered to include only records where supplies have been used (Total_Used > 0), grouped by staff ID, name, and phone number, and ordered by the total supplies used in descending order.

StaffSupplyUsage_View.sql | StaffSupplyUsage_view_firstQuery.sql | StaffSupplyUsage_view_secondQuery.sql | Surgery_Details_View.sc

SQL | Output | Statistics

```sql
-- Query to summarize staff supply usage
SELECT
    Staff_Name,
    Phone_Number,
    SUM(Total_Used) AS Total_Supplies_Used,
    LISTAGG(Supply_Name || ': ' || Total_Used, ', ') WITHIN GROUP (ORDER BY Supply_Name) AS Supply_Usage_Breakdown
FROM StaffSupplyUsage
WHERE Total_Used > 0
GROUP BY SID, Staff_Name, Phone_Number
ORDER BY Total_Supplies_Used DESC;
```

	STAFF_NAME	PHONE_NUMBER	TOTAL_SUPPLIES_USED	SUPPLY_USAGE_BREAKDOWN
1	Fionnula Rucker	972-525425014	39	Broom: 19, White Vinegar: 20
2	Judd Soul	972-539384438	39	Essential Oils: 20, Floor Polish: 19
3	Goran Winger	972-574246193	37	Bleach: 17, Broom: 20
4	David Jones	972-52212345	36	Stainless Steel Cleaner: 16, Trash Bags: 20
5	Grace Rodriguez	972-58745678	36	Fabric Softener: 17, Toilet Bowl Cleaner: 19
6	Leelee Leguizamo	972-580885779	36	Detergent: 16, Window Squeegee: 20
7	Dom Seagal	972-575536654	36	Mop: 16, Rubber Gloves: 20
8	Brothers Duncan	972-570332552	32	Trash Bags: 16, Window Squeegee: 16
9	Elias Jenkins	972-529319526	20	White Vinegar: 20
10	Hal Cassidy	972-505174675	20	Baking Soda: 20
11	Jared Serbedzija	972-596483436	20	Oven Cleaner: 20
12	Anne Wong	972-535764495	20	Mop: 20
13	Maceo Johansson	972-590118515	20	Rubber Gloves: 20
14	Luis Dunaway	972-535634204	20	Lint Roller: 20
15	Davis Grier	972-569798575	20	Essential Oils: 20
16	Mary McCoy	972-570524835	20	Toilet Bowl Cleaner: 20
17	Sean Caldwell	972-590322212	20	Stainless Steel Cleaner: 20

This query evaluates the restock status of supplies based on their current stock and usage. It calculates the total quantity used and the number of staff using each supply and categorizes restock status as 'Urgent', 'Low', or 'Sufficient' based on stock levels. The results are filtered to include only supplies in use (SUID IS NOT NULL), grouped by supply name, total quantity, and location, and ordered by restock urgency and total usage in descending order.

StaffSupplyUsage_View.sql | StaffSupplyUsage_view_firstQuery.sql | StaffSupplyUsage_view_secondQuery.sql | Surgery_Details_View.sc

SQL | Output | Statistics

```sql
-- Query to determine supply restock status
SELECT
    Supply_Name,
    Total_Quantity AS Current_Stock,
    Location,
    SUM(Total_Used) AS Total_Used,
    COUNT(DISTINCT SID) AS Staff_Using,
    CASE
        WHEN Total_Quantity <= SUM(Total_Used) * 0.2 THEN 'Urgent'
        WHEN Total_Quantity <= SUM(Total_Used) * 0.5 THEN 'Low'
        ELSE 'Sufficient'
    END AS Restock_Status
FROM StaffSupplyUsage
WHERE SUID IS NOT NULL
GROUP BY Supply_Name, Total_Quantity, Location
HAVING SUM(Total_Used) > 0
ORDER BY
    CASE Restock_Status
        WHEN 'Urgent' THEN 1
        WHEN 'Low' THEN 2
        ELSE 3
    END,
    Total_Used DESC;
```

	SUPPLY_NAME	CURRENT_STOCK	LOCATION	TOTAL_USED	STAFF_USING	RESTOCK_STATUS
1	Rubber Gloves	110	Storage C	20	1	Sufficient
2	Stainless Steel Cleaner	31	Storage C	20	1	Sufficient
3	Carpet Cleaner	177	Storage A	20	1	Sufficient
4	Oven Cleaner	172	Storage C	20	1	Sufficient
5	Broom	313	Storage B	20	1	Sufficient
6	Mop	213	Storage C	20	1	Sufficient
7	Lint Roller	119	Storage B	20	1	Sufficient
8	Bleach	402	Storage A	20	1	Sufficient
9	Essential Oils	101	Storage B	20	1	Sufficient

This second view, named Surgery_Details_View, consolidates surgery details along with patient and doctor information. It selects surgery ID, date, type, room ID, patient ID, patient first and last name, doctor ID, and doctor first and last name. The data is gathered by joining the Surgery table with the Patient and Doctor tables based on their respective IDs.

```sql
-- Create a view to display surgery details including patient and doctor information
CREATE VIEW Surgery_Details_View AS
SELECT
  Surgery.SurgeryID,
  Surgery.SurgeryDate,
  Surgery.SurgeryType,
  Surgery.RoomID,
  Surgery.PatientID,
  Patient.FirstName AS PatientFirstName,
  Patient.LastName AS PatientLastName,
  Doctor.DoctorID,
  Doctor.FirstName AS DoctorFirstName,
  Doctor.LastName AS DoctorLastName
FROM
  Surgery
JOIN
  Patient ON Surgery.PatientID = Patient.PatientID
JOIN
  Doctor ON Surgery.DoctorID = Doctor.DoctorID;
```

Surgery_Details_View.sql | Surgery_Details_View_fisrtQuery.sql | Surgery_Details_View_secondQuery.sql | select * from surgery_deta ...

SQL | Output | Statistics

```sql
select * from surgery_details_view
```

	SURGERYID	SURGERYDATE	SURGERYTYPE	ROOMID	PATIENTID	PATIENTFIRSTNAME	PATIENTLASTNAME	DOCTORID	DOCTORFIRSTNAME	DOCTORLASTNAME
1	1	4/20/2021	Short_surgery	138	22	Tyler	Golden	312	Jocelin	Nare
2	2	3/24/2018	Intermediate_surgerie	56	225	Vielka	Horn	158	Lorena	Nibloe
3	3	12/5/2012	Long_surgery	75	1	Channing	Craig	191	Sibbie	Lamball
4	4	9/23/2020	Short_surgery	128	128	Sydney	Good	242	Maryl	Ketcher
5	5	4/13/2018	Intermediate_surgerie	162	92	Murphy	Douglas	309	Lilian	McWilliam
6	6	8/6/2012	Intermediate_surgerie	207	174	Cherokee	Coleman	51	Roanna	Yirrell
7	7	6/14/2014	Short_surgery	378	108	Kristen	Rivas	375	Fannie	Penrith
8	8	1/30/2020	Long_surgery	273	195	Griffin	Emerson	134	Adler	Robiou
9	9	7/6/2022	Short_surgery	353	158	Signe	Browning	53	Andrey	Gilbank
10	10	12/30/2021	Short_surgery	318	153	Joan	Buckner	375	Fannie	Penrith
11	11	8/10/2015	Long_surgery	198	83	Keane	Bradshaw	261	Doris	Paske
12	12	3/4/2016	Intermediate_surgerie	355	385	Jesse	Wallace	353	Darryl	Belsey
13	13	8/15/2020	Intermediate_surgerie	227	87	Aurelia	Miranda	64	Derk	Brewitt
14	14	7/14/2015	Intermediate_surgerie	242	172	Lane	Clements	105	Deny	Doick
15	15	4/29/2021	Long_surgery	397	343	Samson	Fitzpatrick	33	Troy	Hellwich
16	16	12/31/2013	Long_surgery	245	358	Kathleen	Ball	337	Antonio	Queyeiro
17	17	1/18/2011	Intermediate_surgerie	8	360	Cassidy	Cobb	373	Keen	Grassot
18	18	8/22/2021	Short_surgery	127	242	Octavia	Mccullough	36	Abbi	Crooks
19	19	9/10/2017	Long_surgery	346	169	Shad	Rowland	224	Jaquelyn	Geindre
20	20	5/15/2021	Long_surgery	293	297	Alfreda	Dorsey	362	Bartram	Ayris
21	21	11/26/2010	Long_surgery	192	341	MacKensie	Battle	238	Barri	Tallow
22	22	2/14/2023	Long_surgery	167	322	Quinn	Everett	145	Alexina	Lyon
23	23	12/27/2021	Intermediate_surgerie	282	292	Wade	Morrow	250	Lucas	Benam
24	24	3/5/2019	Intermediate_surgerie	341	374	John	Matthews	71	Jehu	Meeland
25	25	9/28/2017	Short_surgery	62	71	Jada	Moody	331	Albie	Hazelden
26	26	6/18/2023	Intermediate_surgerie	275	306	Timothy	Morse	291	Lynea	Champley
27	27	10/12/2018	Long_surgery	188	105	Vaughan	Tucker	53	Andrey	Gilbank

This query retrieves details of surgeries that occurred within the last four years. It selects surgery ID, date, type, room ID, patient's first and last names, and doctor's first and last names from the Surgery_Details_View view. The results are filtered to include only those surgeries where the date is within the past 1460 days (approximately four years) and are ordered by the surgery date in ascending order.

StaffSupplyUsage_view_secondQuery.sql | Surgery_Details_View.sql | Surgery_Details_View_fisrtQuery.sql | Surgery_Details_View_secor

SQL | Output | Statistics

```sql
-- Query to retrieve surgery details within the last 4 years
SELECT
  SurgeryID,
  SurgeryDate,
  SurgeryType,
  RoomID,
  PatientFirstName,
  PatientLastName,
  DoctorFirstName,
  DoctorLastName
FROM
  Surgery_Details_View
WHERE
  SurgeryDate BETWEEN SYSDATE - 1460   AND SYSDATE  -- 1460 days == 4 years
order by surgeryDate
```

	SURGERYID	SURGERYDATE	SURGERYTYPE	ROOMID	PATIENTFIRSTNAME	PATIENTLASTNAME	DOCTORFIRSTNAME	DOCTORLASTNAME
1	985	7/17/2020	Intermediate_surgerie	140	Babara	Medwell	Mar	Petrosian
2	3929	7/17/2020	Intermediate_surgerie	300	Dasie	Van der Krui	Hillary	Brougham
3	1326	7/19/2020	Short_surgery	216	Elisha	Torrans	Bunnie	Lydster
4	1669	7/19/2020	Intermediate_surgerie	167	Thea	Berthot	Kalie	Storre
5	4033	7/19/2020	Intermediate_surgerie	129	Odie	Cuffe	Shane	Ramalhete
6	3550	7/24/2020	Intermediate_surgerie	66	Granger	Boseley	Burtie	Dysert
7	312	7/26/2020	Short_surgery	142	Jonas	Morales	Geno	Cogger
8	5380	7/31/2020	Long_surgery	134	Lodovico	Scherme	Shane	Ramalhete
9	224	8/1/2020	Short_surgery	316	Vielka	Horn	Walliw	Genner
10	195	8/3/2020	Intermediate_surgerie	286	Griffin	Emerson	Miguela	Lemonby
11	3572	8/4/2020	Long_surgery	113	Tamar	Keppin	Bartram	Ayris
12	2087	8/6/2020	Short_surgery	395	Winonah	Hackinge	Valdemar	Leyden
13	3872	8/10/2020	Long_surgery	250	Dollie	Grimsell	Alecia	Pockett

This query retrieves data from the Surgery_Details_View to count the total surgeries performed by each doctor. It selects the DoctorID, DoctorFirstName, and DoctorLastName columns, along with counting the occurrences of SurgeryID for each doctor. The GROUP BY clause is used to group the results by DoctorID, DoctorFirstName, and DoctorLastName, ensuring each doctor's total surgeries are aggregated correctly in the output.

Surgery_Details_View.sql | Surgery_Details_View_fisrtQuery.sql | Surgery_Details_View_secondQuery.sql

SQL | Output | Statistics

```sql
-- Query to count total surgeries performed by each doctor
SELECT
  DoctorID,
  DoctorFirstName,
  DoctorLastName,
  COUNT(SurgeryID) AS Total_Surgeries
FROM
  Surgery_Details_View
GROUP BY
  DoctorID,
  DoctorFirstName,
  DoctorLastName;
```

	DOCTORID	DOCTORFIRSTNAME	DOCTORLASTNAME	TOTAL_SURGERIES
1	1	Sol	Wythill	20
2	2	Miguela	Lemonby	14
3	3	Opal	Zum Felde	11
4	4	Florence	Labern	11
5	5	Hulda	Idale	13
6	6	Risa	Wasiela	11
7	7	Ray	Bream	16
8	8	Colby	Blunsden	17
9	9	Anjanette	Garbar	15
10	10	Virginie	Halegarth	15
11	11	Thain	Jaher	12
12	12	Francisca	Allcott	12
13	13	Weston	Toop	11
14	14	Leigh	Vakhrushev	8
15	15	Emmie	Gibbetts	12
16	16	Johan	Colebourn	12