family Heirloom Managment System - PostgreSQL.

* PostgreSQL

- It is an free open source db system that supports both relational (SQL) and non relational (JSON) queries
- It is back-end db for dynamic websites and web appl's.

* SQL schell is a command based appl^{n} that allows us to interact with the postgreSQL data base. There is another appl^{n} that comes built in with it, pgAdmin, which also offers to interact with the db but in a more user-friendly way.

* problem statement:- A large, historical family has been passing down valuable heirlooms from generation to generation. The family members want a digital database to keep track of these heirlooms, their history, current ownership, estimated value & even keep potential disputes over ownership.

* ER diagram Design

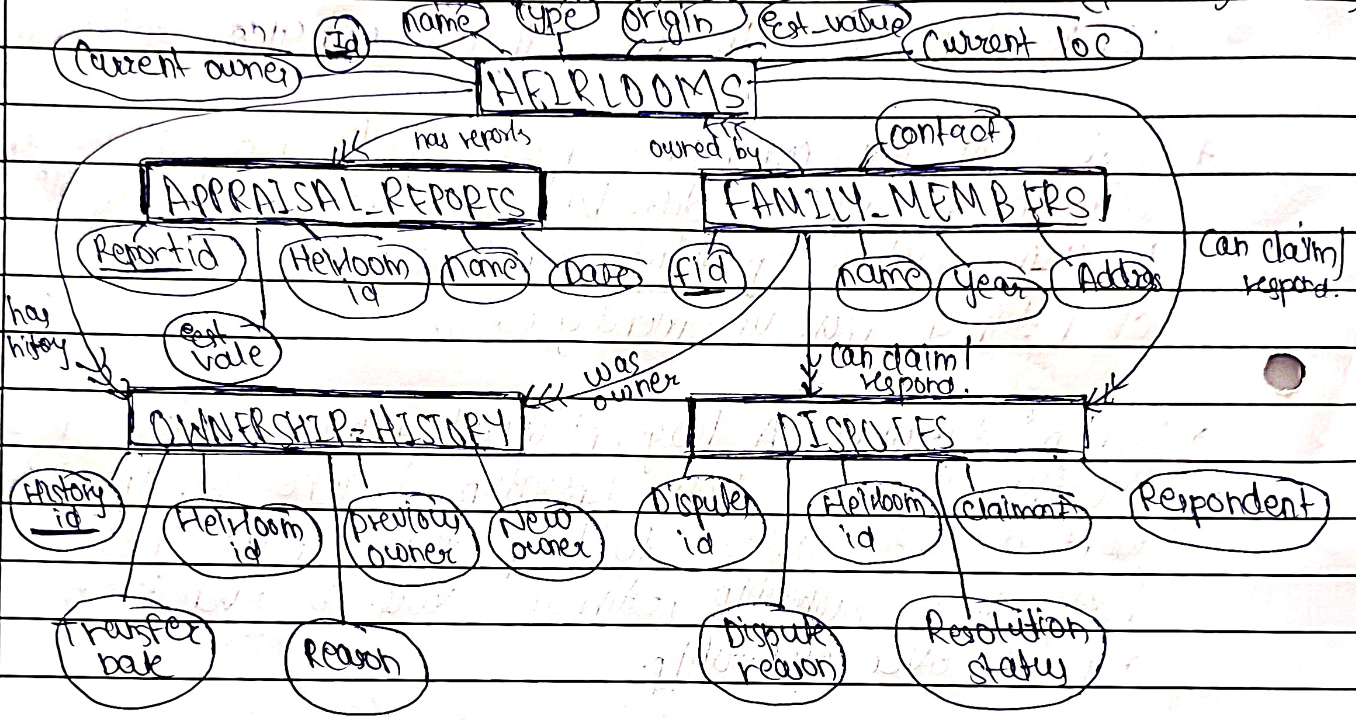
1] Heirlooms → Heirloom_id (primary key), name, type, origin, estimated value, current_location, current_owner (foreign key → Family member)

2] Family members:→ member_id (PK), name, birth year, relation, contact, address.

3] Ownership history → id (pk), heirloop_id (FK → heirlooms), previous_owner (FK → family members), new_owner (FK → f_m), transfer_date, reason

4] Appraisal reports:- report_id (pk), Heirloom_id (FK → Heirlooms), Appraiser_name, Appraisal_date, estimated_value, condition_report

5] Disputes (If family members fight over ownership) → Disput_id(pk) Heirloom_ID (FK → Heirlooms), Claimant (FK → f_m), Respondent (FK → f_M), Dispute_reason, Resolution_ status (pending, settled)



Rectangle ▭ : Entities in ER model.

Ellipse ⬭ : Attributes in ER model

Diamond ◇ : Relationships among entities

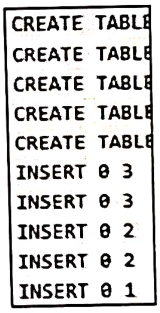
line —— : Attributes to entities relation.

←——↠ : one to many

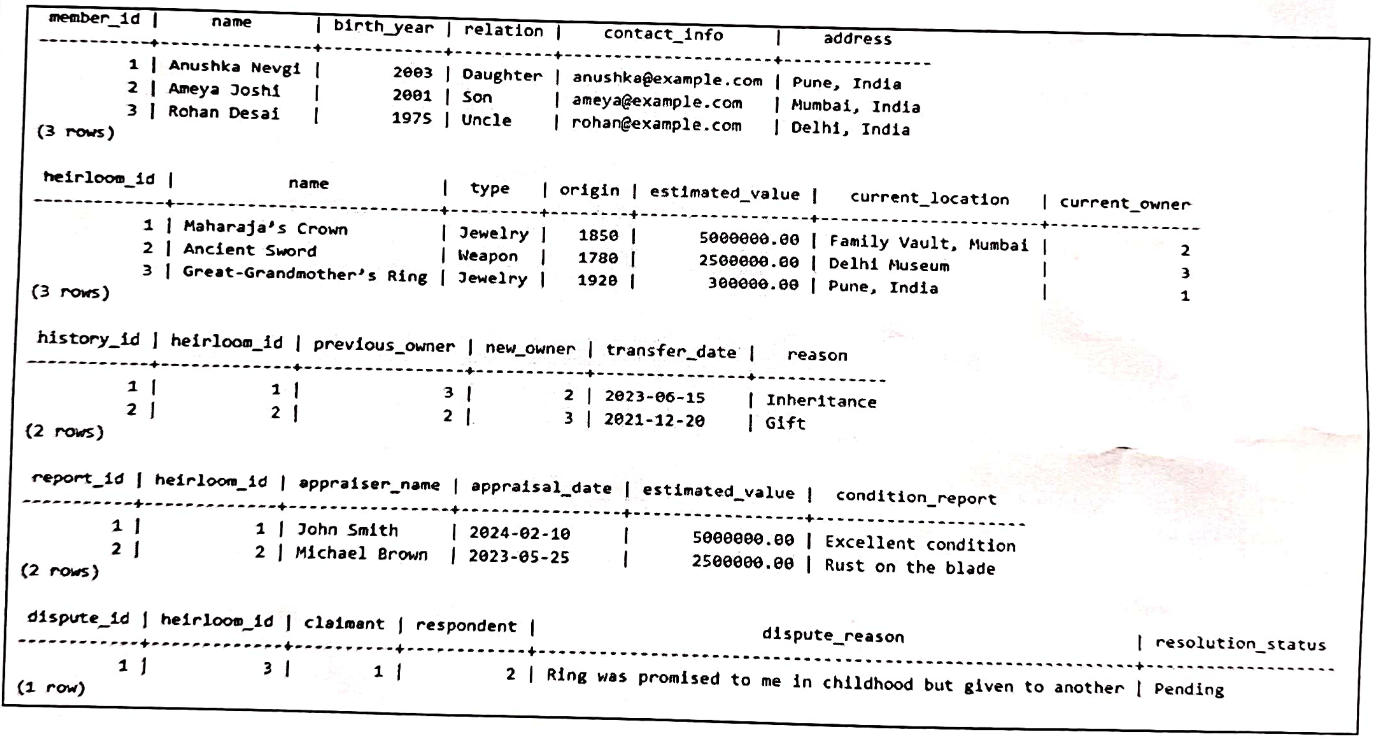
↞——→ : many to one

Data retrieval using PostgreSQL:

```
CREATE TABLE Family_Members (
   Member_ID SERIAL PRIMARY KEY,
   Name VARCHAR(255),
   Birth_Year INT,
   Relation VARCHAR(50),
   Contact_Info VARCHAR(100),
   Address TEXT
);

CREATE TABLE Heirlooms (
   Heirloom_ID SERIAL PRIMARY KEY,
   Name VARCHAR(255),
   Type VARCHAR(50),
   Origin INT,
   Estimated_Value DECIMAL(10,2),
   Current_Location VARCHAR(255),
   Current_Owner INT REFERENCES Family_Members(Member_ID)
);

CREATE TABLE Ownership_History (
   History_ID SERIAL PRIMARY KEY,
   Heirloom_ID INT REFERENCES Heirlooms(Heirloom_ID),
   Previous_Owner INT REFERENCES Family_Members(Member_ID),
   New_Owner INT REFERENCES Family_Members(Member_ID),
   Transfer_Date DATE,
   Reason VARCHAR(50)
);

CREATE TABLE Appraisal_Reports (
   Report_ID SERIAL PRIMARY KEY,
   Heirloom_ID INT REFERENCES Heirlooms(Heirloom_ID),
   Appraiser_Name VARCHAR(255),
   Appraisal_Date DATE,
   Estimated_Value DECIMAL(10,2),
   Condition_Report TEXT
);

CREATE TABLE Disputes (
   Dispute_ID SERIAL PRIMARY KEY,
   Heirloom_ID INT REFERENCES Heirlooms(Heirloom_ID),
   Claimant INT REFERENCES Family_Members(Member_ID),
   Respondent INT REFERENCES Family_Members(Member_ID),
   Dispute_Reason TEXT,
   Resolution_Status VARCHAR(20) CHECK (Resolution_Status IN ('Pending', 'Settled', 'Denied'))
);

INSERT INTO Family_Members (Name, Birth_Year, Relation, Contact_Info, Address) VALUES
('Anushka Nevgi', 2003, 'Daughter', 'anushka@example.com', 'Pune, India'),
('Ameya Joshi', 2001, 'Son', 'ameya@example.com', 'Mumbai, India'),
('Rohan Desai', 1975, 'Uncle', 'rohan@example.com', 'Delhi, India');
```



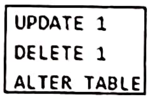
```
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
INSERT 0 3
INSERT 0 3
INSERT 0 2
INSERT 0 2
INSERT 0 1
INSERT INTO Heirlooms (Name, Type, Origin, Estimated_Value, Current_Location, Current_Owner)
VALUES
('Maharaja’s Crown', 'Jewelry', 1850, 5000000.00, 'Family Vault, Mumbai', 2),
('Ancient Sword', 'Weapon', 1780, 2500000.00, 'Delhi Museum', 3),
('Great-Grandmother’s Ring', 'Jewelry', 1920, 300000.00, 'Pune, India', 1);

INSERT INTO Ownership_History (Heirloom_ID, Previous_Owner, New_Owner, Transfer_Date, Reason)
VALUES
(1, 3, 2, '2023-06-15', 'Inheritance'),
(2, 2, 3, '2021-12-20', 'Gift');

INSERT INTO Appraisal_Reports (Heirloom_ID, Appraiser_Name, Appraisal_Date, Estimated_Value,
Condition_Report) VALUES
(1, 'John Smith', '2024-02-10', 5000000.00, 'Excellent condition'),
(2, 'Michael Brown', '2023-05-25', 2500000.00, 'Rust on the blade');

INSERT INTO Disputes (Heirloom_ID, Claimant, Respondent, Dispute_Reason, Resolution_Status) VALUES
(3, 1, 2, 'Ring was promised to me in childhood but given to another', 'Pending');

SELECT * FROM Family_Members;
SELECT * FROM Heirlooms;
SELECT * FROM Ownership_History;
SELECT * FROM Appraisal_Reports;
SELECT * FROM Disputes;
```



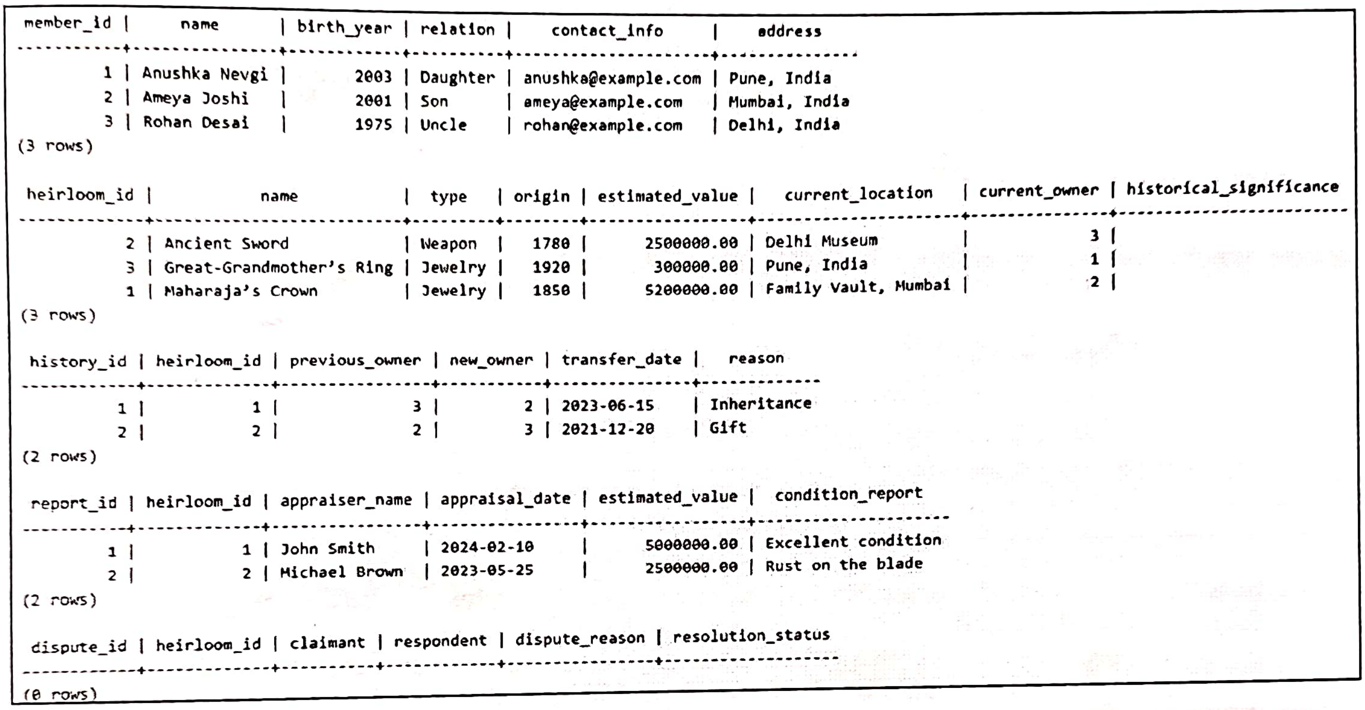
```
 member_id |     name      | birth_year | relation |     contact_info     |    address
-----------+---------------+------------+----------+----------------------+---------------
         1 | Anushka Nevgi |       2003 | Daughter | anushka@example.com  | Pune, India
         2 | Ameya Joshi   |       2001 | Son      | ameya@example.com    | Mumbai, India
         3 | Rohan Desai   |       1975 | Uncle    | rohan@example.com    | Delhi, India
(3 rows)

 heirloom_id |           name            |  type   | origin | estimated_value |   current_location   | current_owner
-------------+---------------------------+---------+--------+-----------------+----------------------+---------------
           1 | Maharaja’s Crown          | Jewelry |   1850 |      5000000.00 | Family Vault, Mumbai |             2
           2 | Ancient Sword             | Weapon  |   1780 |      2500000.00 | Delhi Museum         |             3
           3 | Great-Grandmother’s Ring  | Jewelry |   1920 |       300000.00 | Pune, India          |             1
(3 rows)

 history_id | heirloom_id | previous_owner | new_owner | transfer_date |   reason
------------+-------------+----------------+-----------+---------------+-------------
          1 |           1 |              3 |         2 | 2023-06-15    | Inheritance
          2 |           2 |              2 |         3 | 2021-12-20    | Gift
(2 rows)

 report_id | heirloom_id | appraiser_name | appraisal_date | estimated_value |  condition_report
-----------+-------------+----------------+----------------+-----------------+---------------------
         1 |           1 | John Smith     | 2024-02-10     |      5000000.00 | Excellent condition
         2 |           2 | Michael Brown  | 2023-05-25     |      2500000.00 | Rust on the blade
(2 rows)

 dispute_id | heirloom_id | claimant | respondent |                      dispute_reason                       | resolution_status
------------+-------------+----------+------------+-----------------------------------------------------------+-------------------
          1 |           3 |        1 |          2 | Ring was promised to me in childhood but given to another | Pending
(1 row)
```

Apply update, delete, alter and drop commands:

```
UPDATE Heirlooms
SET Estimated_Value = 5200000.00
WHERE Heirloom_ID = 1;
DELETE FROM Disputes WHERE Dispute_ID = 1;
```

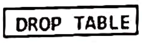


```
UPDATE 1
DELETE 1
ALTER TABLE
ALTER TABLE Heirlooms ADD COLUMN Historical_Significance TEXT;

SELECT * FROM Family_Members;
SELECT * FROM Heirlooms;
SELECT * FROM Ownership_History;
SELECT * FROM Appraisal_Reports;
SELECT * FROM Disputes;
```



```
 member_id |     name      | birth_year | relation |    contact_info     |    address
-----------+---------------+------------+----------+---------------------+---------------
         1 | Anushka Nevgi |       2003 | Daughter | anushka@example.com | Pune, India
         2 | Ameya Joshi   |       2001 | Son      | ameya@example.com   | Mumbai, India
         3 | Rohan Desai   |       1975 | Uncle    | rohan@example.com   | Delhi, India
(3 rows)

 heirloom_id |           name            |  type   | origin | estimated_value |    current_location    | current_owner | historical_significance
-------------+---------------------------+---------+--------+-----------------+------------------------+---------------+-------------------------
           2 | Ancient Sword             | Weapon  |   1780 |      2500000.00 | Delhi Museum           |             3 |
           3 | Great-Grandmother's Ring  | Jewelry |   1920 |       300000.00 | Pune, India            |             1 |
           1 | Maharaja's Crown          | Jewelry |   1850 |      5200000.00 | Family Vault, Mumbai   |             2 |
(3 rows)

 history_id | heirloom_id | previous_owner | new_owner | transfer_date |   reason
------------+-------------+----------------+-----------+---------------+-------------
          1 |           1 |              3 |         2 | 2023-06-15    | Inheritance
          2 |           2 |              2 |         3 | 2021-12-20    | Gift
(2 rows)

 report_id | heirloom_id | appraiser_name | appraisal_date | estimated_value |  condition_report
-----------+-------------+----------------+----------------+-----------------+--------------------
         1 |           1 | John Smith     | 2024-02-10     |      5000000.00 | Excellent condition
         2 |           2 | Michael Brown  | 2023-05-25     |      2500000.00 | Rust on the blade
(2 rows)

 dispute_id | heirloom_id | claimant | respondent | dispute_reason | resolution_status
------------+-------------+----------+------------+----------------+-------------------
(0 rows)
```

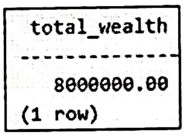
```
DROP TABLE Disputes;
```



```
DROP TABLE
```

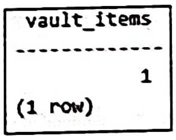
Apply aggregate queries:

```
SELECT SUM(Estimated_Value) AS Total_Wealth FROM Heirlooms;
```



```
 total_wealth
--------------
   8000000.00
(1 row)
```

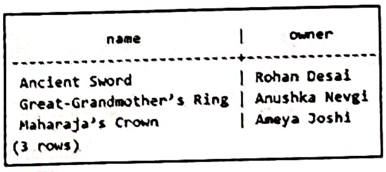
```
SELECT COUNT(*) AS Vault_Items FROM Heirlooms WHERE Current_Location = 'Family
Vault, Mumbai';
```



```
 vault_items
-------------
           1
(1 row)
```

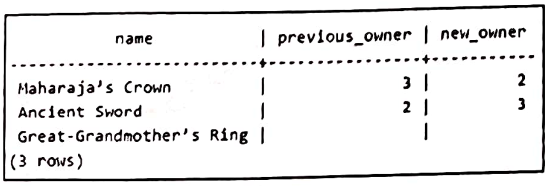
Apply joins:

```
SELECT H.Name, F.Name AS Owner
FROM Heirlooms H
INNER JOIN Family_Members F ON H.Current_Owner = F.Member_ID;
```



```
           name            |     owner
---------------------------+---------------
 Ancient Sword             | Rohan Desai
 Great-Grandmother's Ring  | Anushka Nevgi
 Maharaja's Crown          | Ameya Joshi
(3 rows)
```

```
SELECT H.Name, OH.Previous_Owner, OH.New_Owner
FROM Heirlooms H
LEFT JOIN Ownership_History OH ON H.Heirloom_ID = OH.Heirloom_ID;
```



```
           name            | previous_owner | new_owner
---------------------------+----------------+-----------
 Maharaja's Crown          |              3 |         2
 Ancient Sword             |              2 |         3
 Great-Grandmother's Ring  |                |
(3 rows)
```

JS Object notation → easy for humans to read and write, server & web app.

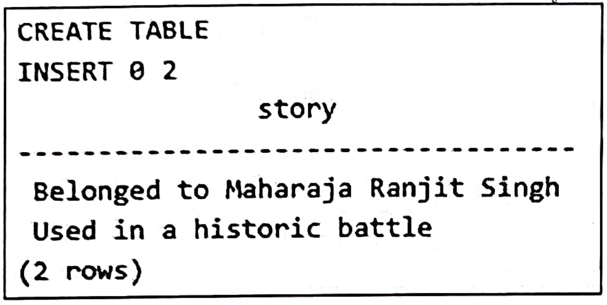
Create table and insert multiple JSON data into it:

```
CREATE TABLE Heirloom_Stories (
   Story_ID SERIAL PRIMARY KEY,
   Heirloom_ID INT REFERENCES Heirlooms(Heirloom_ID),
   Story_Data JSONB → Binary JSON → faster executing JSON data
);

INSERT INTO Heirloom_Stories (Heirloom_ID, Story_Data) VALUES
(1, '{"Legend": "Belonged to Maharaja Ranjit Singh", "Passed_Down": "Over 6 generations"}'),
(2, '{"Legend": "Used in a historic battle", "Condition": "Slightly damaged"}');
```

Extract JSON data from the table:

```
SELECT Story_Data->>'Legend' AS Story FROM Heirloom_Stories;
```

retrieves value of legend key & assigns to story



```
CREATE TABLE
INSERT 0 2
                story
-----------------------------------
 Belonged to Maharaja Ranjit Singh
 Used in a historic battle
(2 rows)
```

Differences Between PostgreSQL and MySQL based on this case study:

1. **Auto-Increment Column**
   - In **PostgreSQL**, we used SERIAL to create an auto-incrementing primary key.
   - In **MySQL**, we would use AUTO_INCREMENT instead.
2. **JSON Handling**
   - In **PostgreSQL**, we used JSONB for storing structured data efficiently.
   - In **MySQL**, only JSON is available, and it is not as optimized for indexing and querying.
3. **Foreign Keys and Constraints**
   - In **PostgreSQL**, we directly applied foreign key constraints when creating tables.
   - In **MySQL**, foreign key constraints are available but work best with the InnoDB engine.
4. **Joins and Query Execution**
   - We performed INNER JOIN, LEFT JOIN, and RIGHT JOIN, which work the same in both databases.
   - **PostgreSQL** supports FULL OUTER JOIN, which is **not available in MySQL** (it requires a UNION).
5. **Indexing JSON Fields**
   - In **PostgreSQL**, we can create a GIN index for efficient JSONB queries.
   - In **MySQL**, JSON indexing is very limited, requiring additional workarounds.
6. **Table Modifications (ALTER TABLE)**
   - **PostgreSQL** allows renaming columns and adding constraints flexibly.
   - **MySQL** is more restrictive in modifying existing constraints.

Conclusion

The main differences in this experiment were **JSON handling, indexing, and full outer joins**, where **PostgreSQL has more advanced features than MySQL**. However, basic SQL operations like CREATE TABLE, INSERT, SELECT, and regular joins work similarly in both databases.