Prepared by DataScape

OURSPACE

Kaung Nyo Lwin
Truong Vuong
Aymen Zubair Qureshi
Cassandra Chang

DataScape

OURSPACE - NOSQL

OurSpace is an online platform that connects people looking for shared spaces with owners who have spaces to rent, such as co-working spaces, meeting rooms or private offices.

Space owners can list their space with details like location, size , price , facilities or Rental rates. customers can search, book, and review spaces in just a few clicks. The platform handles payments securely and earns a small 3% service fee on each booking.

IMPORTANCE

Reasons why we think this is important (and interesting!):

- Space owners can rent out empty offices and earn extra income.
- Many people work from home or freelance and need short-term office spaces.
- The platform recommends spaces based on what customers like and need.
- Gives smart recommendations, shows simple reports, and improves based on customer feedback to make the platform easy to use.

DATA MODEL

USER:
- ID
- Name
- Email
- Phone
- Address
- Prefered Price Range
- Number of bookings
- Total Spent
- CreatedAt
- UpdatedAt

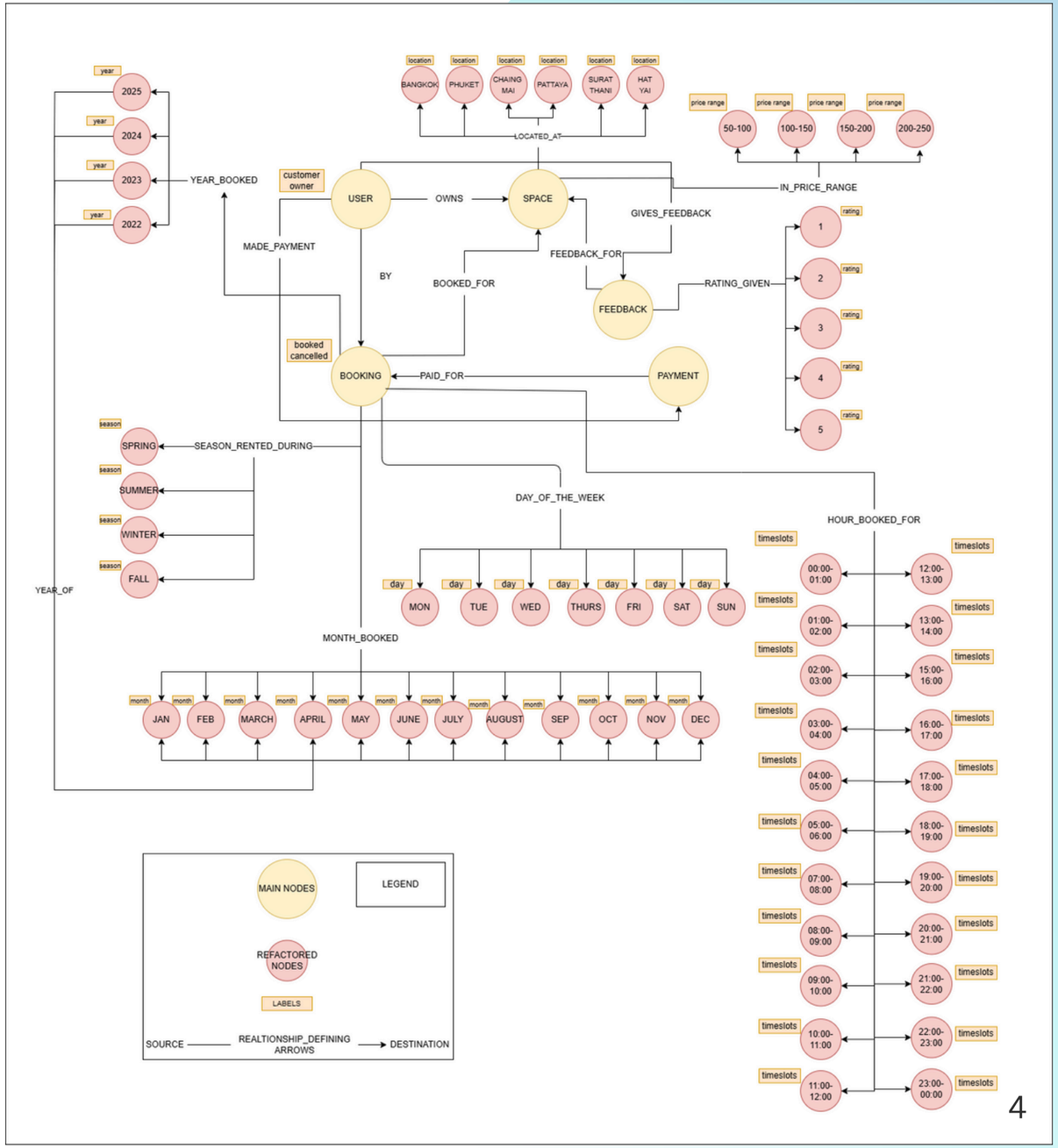
FEEDBACK:
- ID
- Reviews
- CreatedAt

Payment:
- Amount
- TransacId
- Bank
- Status
- CreatedAt
- UpdatedAt

SPACE:
- SpaceID
- SpaceName
- Address
- Size
- numRooms
- hourlyRate
- HalfdayRate
- FulldayRate
- Status
- Remark
- Rating
- CreatedAt
- UpdatedAt

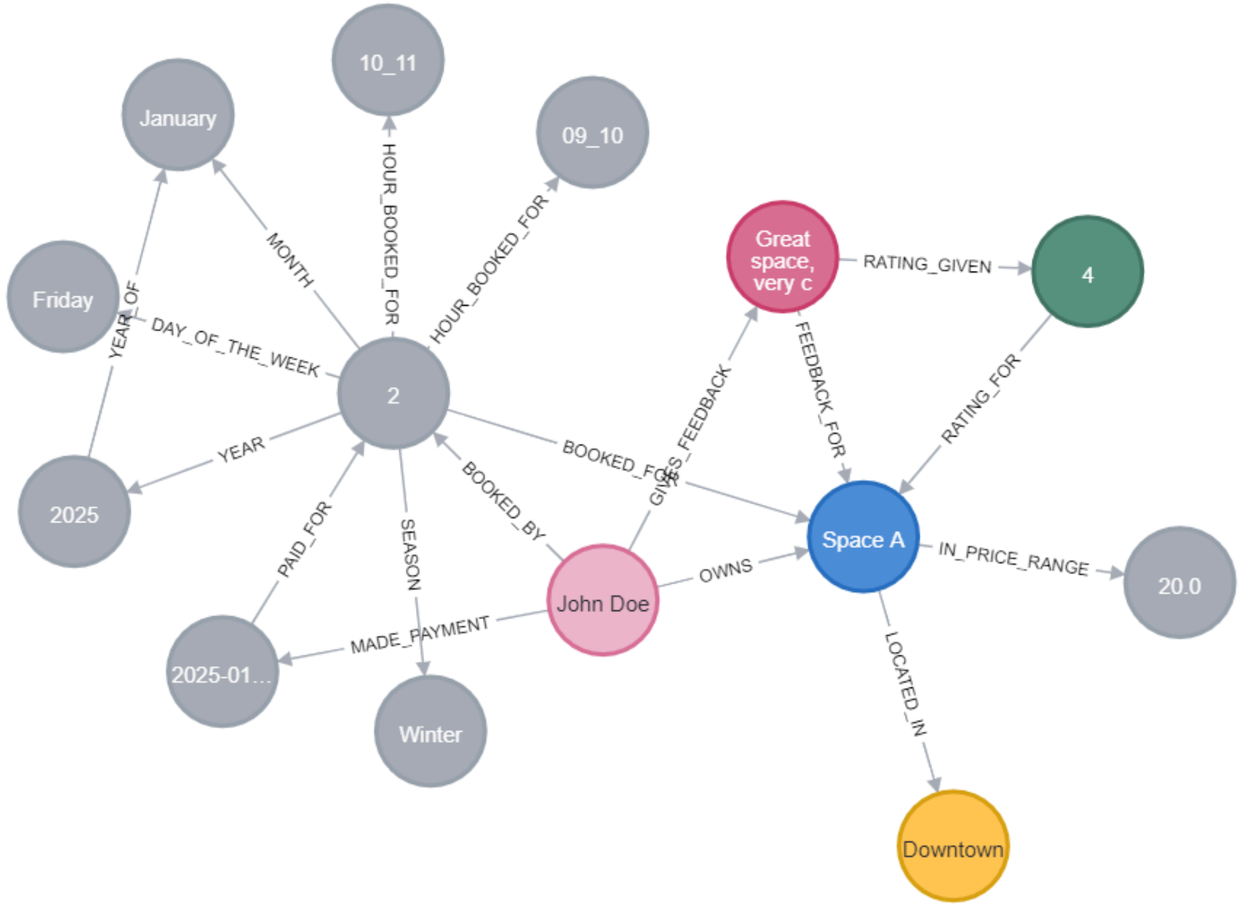
BOOKING:
- Booking Id
- Booking Date
- Space Price
- System Fee
- Total
- CreatedAt
- UpdatedAt

REFACTORED LABELS:
- ***Timeslot***
- ***Day***
- ***Month***
- ***Year***
- ***Season***
- ***Rating***
- ***Price_Range***
- ***Location***



All **refactored labels** except “rating” and “location” have the properties “**Number of Bookings**” and “**Total Spent Amount**” updated whenever a booking is made.

DATA MODEL IMPLEMNTATION

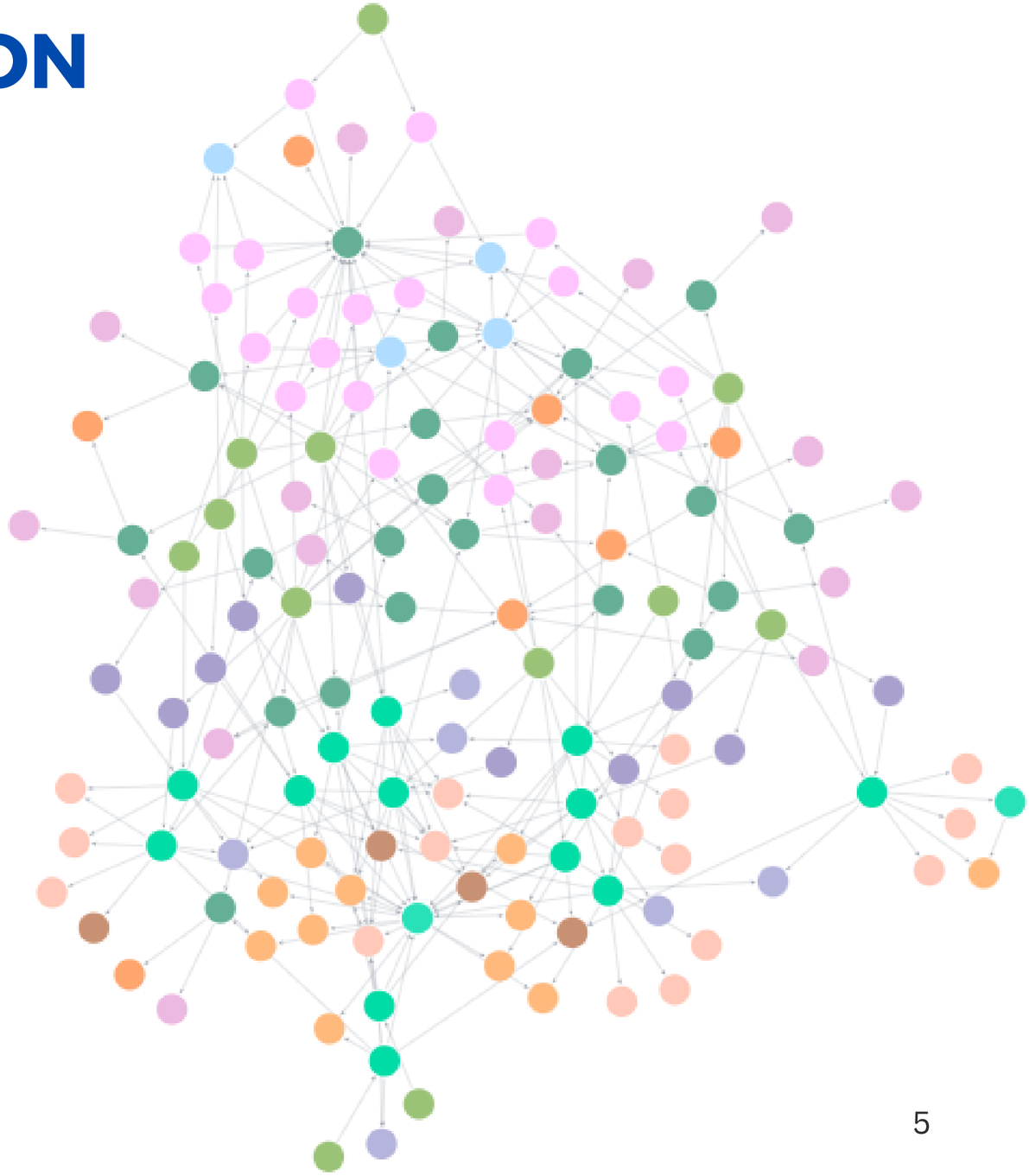


Nodes (151)

* Booking Cancelled Customer Day Feedback Location Month Owner Paid Payment PriceRange Rating Season Space Timeslot User Year

Relationships (278)

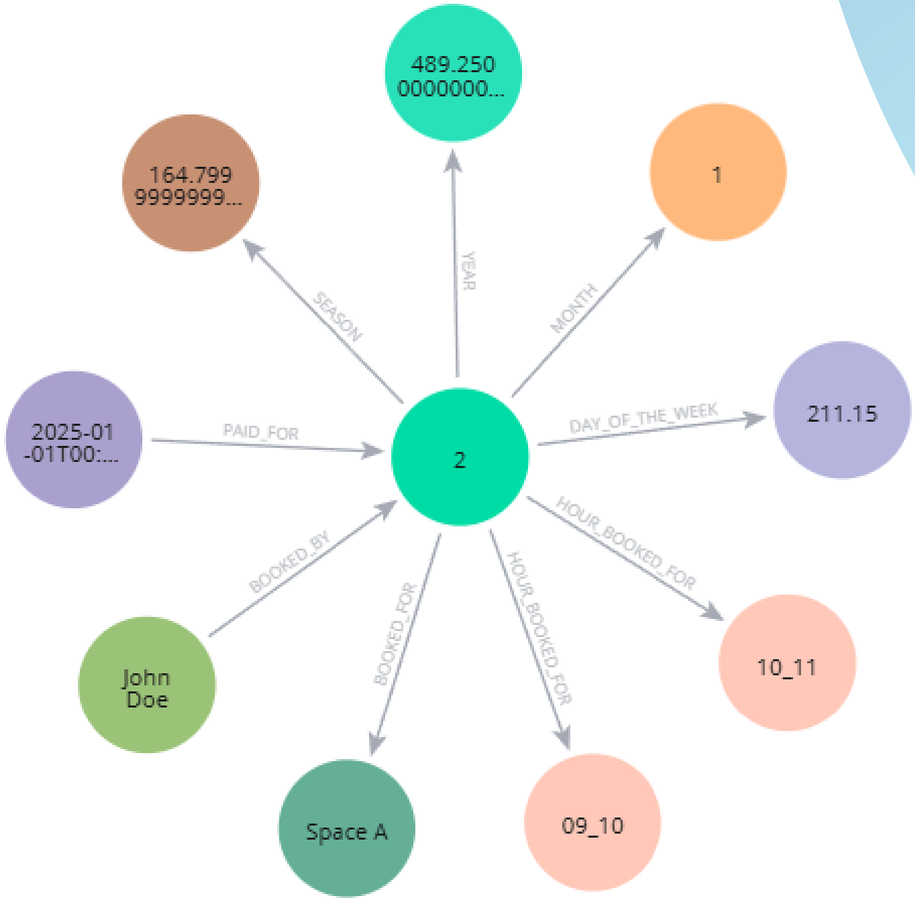
* BOOKED_BY BOOKED_FOR DAY_OF_THE_WEEK FEEDBACK_FOR GIVES_FEEDBACK HOUR_BOOKED_FOR IN_PRICE_RANGE LOCATED_IN MADE_PAYMENT MONTH OWNS PAID_FOR RATING_FOR RATING_GIVEN SEASON YEAR YEAR_OF



BOOKING

Kaung Nyo Lwin

```
MERGE (b1:Booking {id: 1,
                   bookingDate: '2025-01-10',
                   remark: 'First booking',
                   createdAt: '2025-01-01T00:00:00Z',
                   updatedAt: '2025-01-01T00:00:00Z'
                   })
// 2. Link Booking node to User
WITH b1
MATCH (u1:User {id: 1})
MERGE (u1)-[:BOOKED_BY]->(b1)
// 3. Link Booking node to Space
WITH b1, u1
MATCH (s1:Space {id: 1})
MERGE (b1)-[:BOOKED_FOR]->(s1)
// 4. Create Timeslot nodes and link to Booking
MERGE (t1:Timeslot {slot: '09_10'})
MERGE (b1)-[:HOUR_BOOKED_FOR]->(t1)
MERGE (t2:Timeslot {slot: '10_11'})
MERGE (b1)-[:HOUR_BOOKED_FOR]->(t2)
// 5. Calculate and update total duration and price and set the user as customer
WITH b1, u1, s1
// Count relationships using pattern comprehension
SET b1.totalDuration = SIZE([(b1)-[:HOUR_BOOKED_FOR]->() | 1])
// Fix CASE logic (use totalDuration)
SET b1.spacePrice = CASE
  WHEN b1.totalDuration < 12 THEN s1.hourlyRate * b1.totalDuration
  WHEN b1.totalDuration = 12 THEN s1.halfdayRate
  WHEN b1.totalDuration = 24 THEN s1.fulldayRate
  ELSE s1.halfdayRate + (s1.hourlyRate * (b1.totalDuration - 12))
END
SET b1.sysFee = b1.spacePrice * 0.03
SET b1.totalPrice = b1.spacePrice + b1.sysFee
SET u1.numOfBookings = u1.numOfBookings + 1
SET u1.totalSpent = u1.totalSpent + b1.totalPrice
SET u1:Customer
Create Day node
(d1:Day {day:
ASE date(b1.bookingDate).dayOfWeek
  WHEN 1 THEN 'Monday'
  WHEN 2 THEN 'Tuesday'
  WHEN 3 THEN 'Wednesday'
  WHEN 4 THEN 'Thursday'
  WHEN 5 THEN 'Friday'
  WHEN 6 THEN 'Saturday'
  ELSE 'Sunday'
ND})
EATE SET
1.numOfBookings = 0,
1.totalSpent = 0.0
Link Booking node to Day
(b1)-[:DAY_OF_THE_WEEK]->(d1)
```




```
// 14. Update Day, Month, Year, and Season nodes and price range
SET d1.numOfBookings = d1.numOfBookings + 1
SET d1.totalSpent = d1.totalSpent + b1.totalPrice
SET m1.numOfBookings = m1.numOfBookings + 1
SET m1.totalSpent = m1.totalSpent + b1.totalPrice
SET y1.numOfBookings = y1.numOfBookings + 1
SET y1.totalSpent = y1.totalSpent + b1.totalPrice
SET se1.numOfBookings = se1.numOfBookings + 1
SET se1.totalSpent = se1.totalSpent + b1.totalPrice
WITH b1, u1, s1
MATCH (s1)-[:IN_PRICE_RANGE]->(pr1)
SET pr1.numOfBookings = pr1.numOfBookings + 1
```

```
// 8. Create Month node
MERGE (m1:Month {month:
    CASE date(b1.bookingDate).month
        WHEN 1 THEN 'January'
        WHEN 2 THEN 'February'
        WHEN 3 THEN 'March'
        WHEN 4 THEN 'April'
        WHEN 5 THEN 'May'
        WHEN 6 THEN 'June'
        WHEN 7 THEN 'July'
        WHEN 8 THEN 'August'
        WHEN 9 THEN 'September'
        WHEN 10 THEN 'October'
        WHEN 11 THEN 'November'
        ELSE 'December'
    END,
    year: date(b1.bookingDate).year,
    numMonth: date(b1.bookingDate).month})
ON CREATE SET
    m1.numOfBookings = 0,
    m1.totalSpent = 0.0
// 9. Link Booking node to Month
MERGE (b1)-[:MONTH]->(m1)
// 10. Create Year node
MERGE (y1:Year {year: date(b1.bookingDate).year})
ON CREATE SET
    y1.numOfBookings = 0,
    y1.totalSpent = 0.0
// 11. Link Booking node to Year
MERGE (b1)-[:YEAR]->(y1)
// 12. Create Season node
MERGE (y1)-[:YEAR_OF]->(m1)
MERGE (se1:Season {season:
  CASE
    WHEN date(b1.bookingDate).month IN [1,2,3] THEN 'Winter'
    WHEN date(b1.bookingDate).month IN [4,5,6] THEN 'Spring'
    WHEN date(b1.bookingDate).month IN [7,8,9] THEN 'Summer'
    ELSE 'Fall'
  END})
ON CREATE SET
    se1.numOfBookings = 0,
    se1.totalSpent = 0.0
// 13. Link Booking node to Season
MERGE (b1)-[:SEASON]->(se1)
```

OTHER TRANSACTIONS

Kaung Nyo Lwin

User Registration:

```
MERGE (u1:User {id: 1,
                name: 'John Doe',
                email: 'johndoe@example.com',
                phone: '1234567890',
                address: '123 Main St',
                preferedRange: 10.0,
                numOfBookings: 0,
                totalSpent: 0.0,
                createdAt: '2023-01-01T00:00:00Z',
                updatedAt: '2023-01-01T00:00:00Z'
                }
      )
```

Paying the charges:

```
MATCH (b1:Booking {id: 1})
MERGE (p1:Payment {id: 1,
                   amount: b1.totalPrice,
                   transacId: 'TXN12345',
                   bank: 'Bank A',
                   status: 'completed',
                   createdAt: '2025-01-01T00:00:00Z',
                   updatedAt: '2025-01-01T00:00:00Z'
                   }
      )
MERGE (p1)-[:PAID_FOR]->(b1)
WITH p1, b1
MATCH (u1:User {id: 1})
MERGE (u1)-[:MADE_PAYMENT]->(p1)
SET b1:Paid
```

IMPORTANT DATA QUERIES AND REPORT

Most spenders Report: This report retrieves the most spending customers based on their rental activities of the specified period. The output will contain the customer name, spent amount, average spending per day and how many days the users have been on the platform.

```
MATCH (c:Customer)
RETURN
    c.name AS `Customer Name`,
    c.numOfBookings AS `Number of Bookings`,
    c.totalSpent AS `Total Spent`,
    c.totalSpent / ((datetime().epochSeconds - datetime(c.createdAt).epochSeconds)
    / 86400.0) AS `Spent per day`,
    (datetime().epochSeconds - datetime(c.createdAt).epochSeconds)
    / 86400.0 AS `Days on the platform`
ORDER BY c.totalSpent DESC
```

Popular Day of the Week report: This generates a report to see the most busy days of the weeks of the specified period. The output will contain the day of the week, the total number of bookings, price, and fee.

```
MATCH (d:Day)
RETURN d.day AS `Day of the Week`,
       d.numOfBookings AS `Number of Bookings`,
       d.totalSpent AS `Total Revenue`,
       d.totalSpent * 0.03 AS `System Fee`,
       d.totalSpent * 0.97 AS `Space Price`
ORDER BY d.numOfBookings DESC
```

Rental Report by price range: This generates a report that shows the number of bookings, spending amount, and total fee by defined price ranges

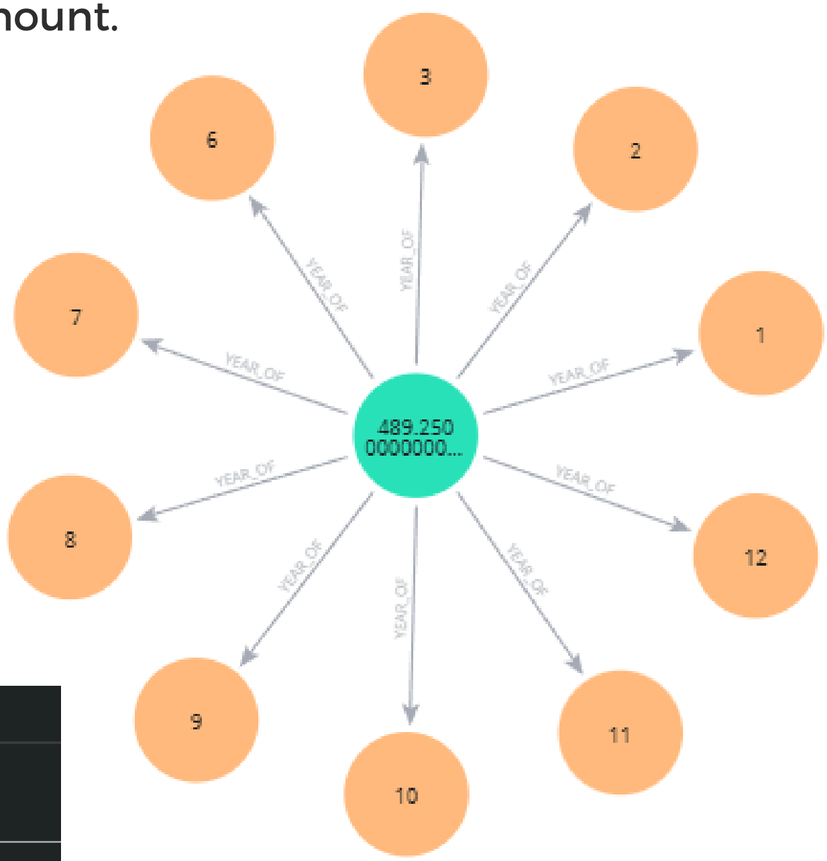
```
MATCH (pr:PriceRange)
RETURN pr.minPrice AS `Min Price`,
       pr.maxPrice AS `Max Price`,
       pr.numOfBookings AS `Number of Bookings`,
       pr.totalSpent AS `Total Revenue`,
       pr.totalSpent * 0.03 AS `System Fee`,
       pr.totalSpent * 0.97 AS `Space Price`
ORDER BY pr.minPrice ASC
```

Kaung Nyo Lwin

MONTHLY REVENUE REPORT

This generates a revenue report based on monthly detailed rental transactions of the specified period. The output will contain the month, the year, the price, the fee, and the total amount.

```
MATCH (y:Year {year: 2025}) - [yf:YEAR_OF] -> (m:Month)
// RETURN y, yf, m
RETURN m.month AS `Month`,
       m.year AS `Year`,
       m.numOfBookings AS `Number of Bookings`,
       m.totalSpent AS `Total Revenue`,
       m.totalSpent * 0.03 AS `System Fee`,
       m.totalSpent * 0.97 AS `Space Price`
ORDER BY m.numMonth ASC
```



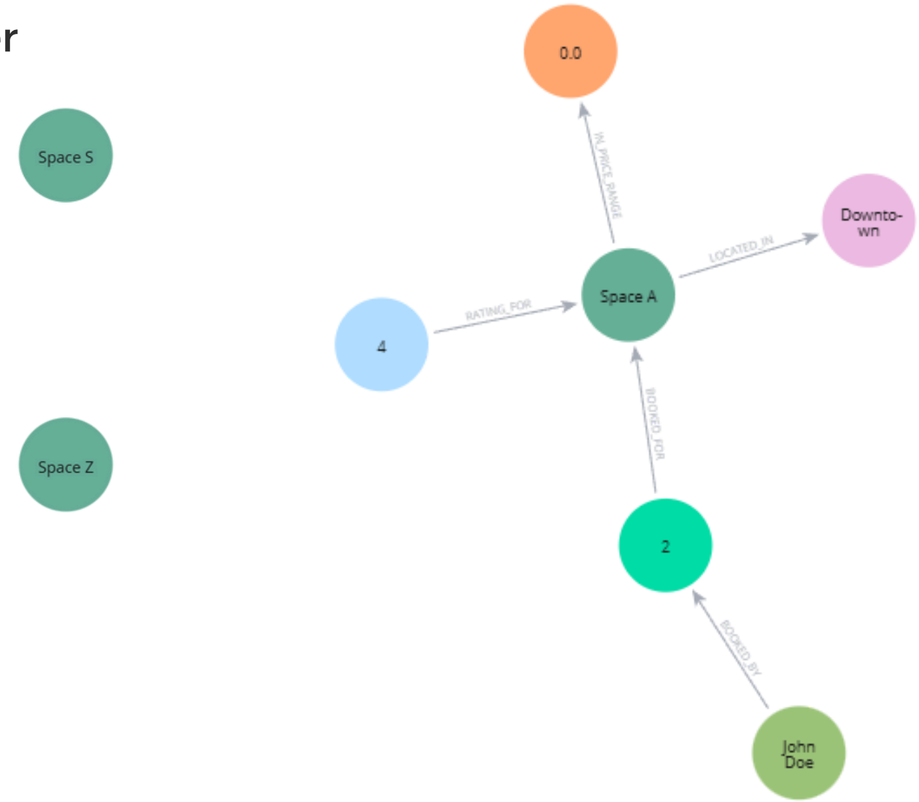
	Month	Year	Number of Bookin	Total Revenue	System Fee	Space Price
1	"January"	2025	2	56.650000000000006	1.6995	54.950500000000005
2	"February"	2025	1	30.9	0.9269999999999999	29.973
3	"March"	2025	1	77.25	2.3175	74.9325

Kaung Nyo Lwin

SPACE RECOMMEDATION QUERY

This query retrieves a list of available spaces based on the user's information such as preferred price range and history of booking by taking the input (location) of user

```
// Find spaces in the user has ever booked
MATCH (u:User {id: 1}) - [by:BOOKED_BY] -> (b:Booking) - [bf:BOOKED_FOR] -> (s:Space)
WITH u, b, s, by, bf
// Filter spaces in the location
MATCH (s)-[li:LOCATED_IN] -> (l:Location {name: 'Downtown'})
WITH u, b, s, by, bf, l, li
// Filter spaces in the price range
MATCH (s)-[ipr:IN_PRICE_RANGE] -> (pr:PriceRange)
WHERE u.preferedRange >= pr.minPrice AND u.preferedRange <= pr.maxPrice
// Return the results
// RETURN u, b, s, l, pr, by, bf, li, ipr,r, rf
RETURN s.name AS `Space`,
       l.name AS `Location`,
       Count(b) AS `Number of Bookings`,
       s.hourlyRate AS `Hourly Rate`,
       s.halfdayRate AS `Half Day Rate`,
       s.fulldayRate AS `Full Day Rate`,
       s.rating AS `Average Rating`
ORDER BY Count(b) DESC, s.rating DESC
// Get other spaces not booked by the user
UNION
MATCH (u:User {id: 1})
WITH u
MATCH (s:Space)
WHERE NOT (u:User {id: 1}) - [:BOOKED_BY] -> (:Booking) - [:BOOKED_FOR] -> (s:Space)
AND (s)-[:LOCATED_IN]->(:Location {name: 'Downtown'})
RETURN s.name AS `Space`,
       'Downtown' AS `Location`,
       0 AS `Number of Bookings`,
       s.hourlyRate AS `Hourly Rate`,
       s.halfdayRate AS `Half Day Rate`,
       s.fulldayRate AS `Full Day Rate`,
       s.rating AS `Average Rating`
ORDER BY s.rating DESC
```



	Space	Location	Number of Bookin	Hourly Rate	Half Day Rate	Full Day Rate	Average Rating
1	"Space A"	"Downtown"	1	30.0	120.0	200.0	2
2	"Space Z"	"Downtown"	0	20.0	100.0	180.0	0
3	"Space S"	"Downtown"	0	29.0	135.0	230.0	0

TRANSACTIONS

Truong Vuong

4

Space Status Update:

Space owners can update the availability status of their listed spaces (e.g., open, closed, available, unavailable or temporarily unavailable for maintenance).

```
MATCH (s:Space {name: "Space Z"})
SET s.status = "cloesed"
RETURN s.name, s.status
```

5

Space Removal:

The platform administrators can remove a space listing temporarily or permanently due to violations, safety concerns, or legal issues.

```
MATCH (s:Space {name: "Space Z"})
DETACH DELETE s
```

6

Change Number of Facilities:

Space owners can change the number of specific facilities available in their listed spaces (e.g., increase or decrease the number of chairs, tables, etc.).

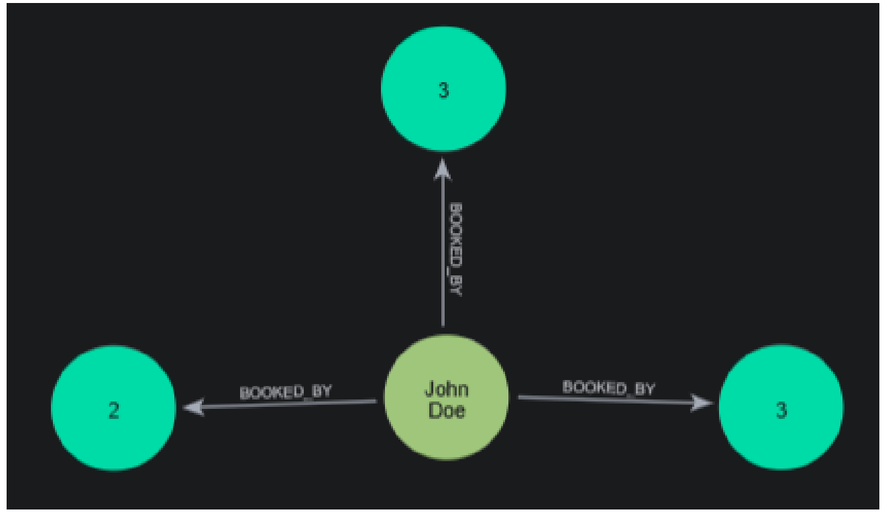
```
MATCH (s:Space {name: "Space Z"})
SET s.facilities = '{"AC": 2, "kitchen": 1}'
RETURN s.name, s.facilities
```

Truong Vuong

IMPORTANT DATA QUERIES AND REPORT

Customer Retention Report: Calculates the total bookings, total spend, last booking date, and retention status for each customer. Retention status can be based on the frequency of bookings (e.g., the number of TotalBokings is greater than or equal to 3 bookings => "Loyal").

```cypher
MATCH (u:User:Customer)-[:BOOKED_BY]->(b:Booking)
WITH
  u.name AS CustomerName,
  COUNT(b) AS TotalBookings,
  SUM(b.totalPrice) AS TotalSpend,
  MAX(b.bookingDate) AS LastBookingDate
RETURN
  CustomerName,
  TotalBookings,
  TotalSpend,
  LastBookingDate,
  CASE
    WHEN TotalBookings >= 3 THEN 'Loyal'
    ELSE 'Casual'
  END AS RetentionStatus
ORDER BY TotalBookings DESC
```



	CustomerName	TotalBookings	TotalSpend	LastBookingDate	RetentionStatus
1	"John Doe"	3	208.06	"2025-11-01"	"Loyal"
2	"Jane Smith"	2	117.41999999999999	"2025-02-15"	"Casual"
3	"Alice Johnson"	1	77.25	"2025-03-25"	"Casual"
4	"Bob Brown"	1	15.45	"2025-01-20"	"Casual"

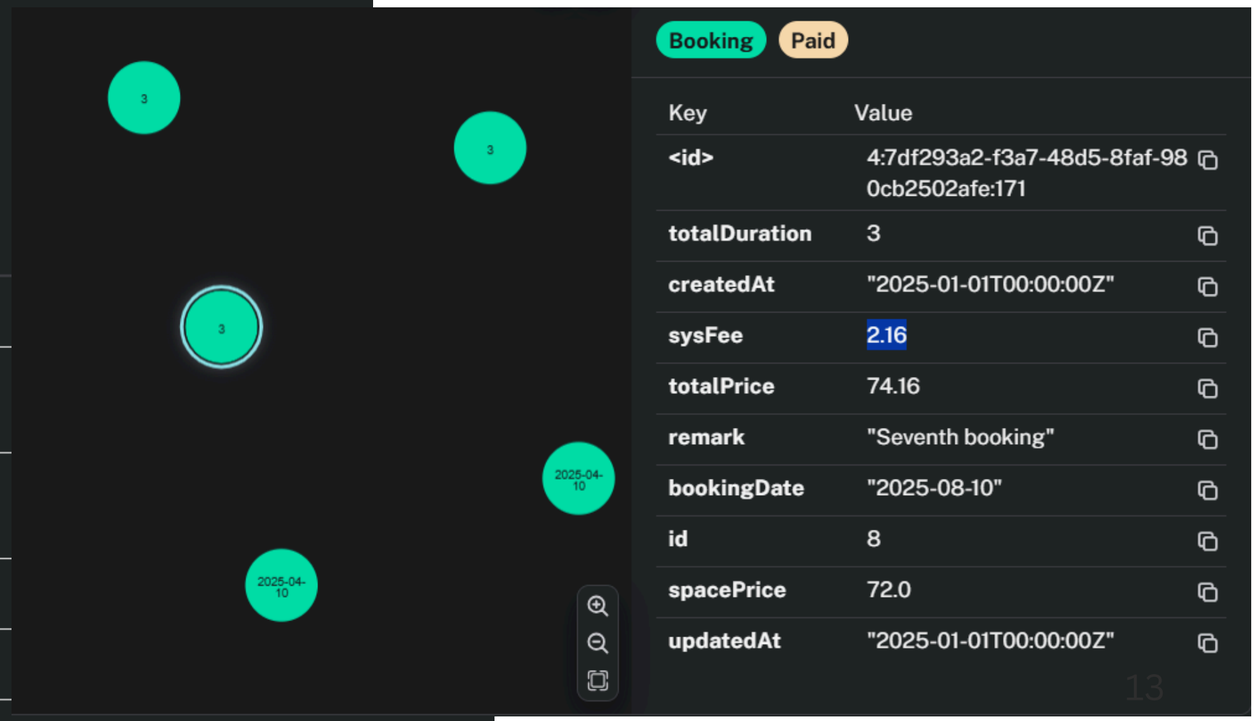
Truong Vuong

IMPORTANT DATA QUERIES AND REPORT

Total System Fee: calculates the total system fee by month, providing insights into monthly revenue trends specifically for System. This helps us monitor our revenue streams from booking transactions.

```
MATCH (b:Booking)
WITH
  date(b.bookingDate).year AS Year,
  date(b.bookingDate).month AS MonthNum,
  apoc.text.format('%04d-%02d', [date(b.bookingDate).year, date(b.bookingDate).month])
AS Month,
  SUM(b.sysFee) AS TotalSystemFee,
  COUNT(b) AS TotalBookings
RETURN
  Month,
  TotalSystemFee,
  TotalBookings
ORDER BY Month ASC
```

	Month	TotalSystemFee	TotalBookings
1	"2024-12"	2.52	1
2	"2025-01"	2.3099999999999996	3
3	"2025-02"	0.8999999999999999	1
4	"2025-03"	2.25	1
5	"2025-04"	0	7
6	"2025-06"	1.98	1



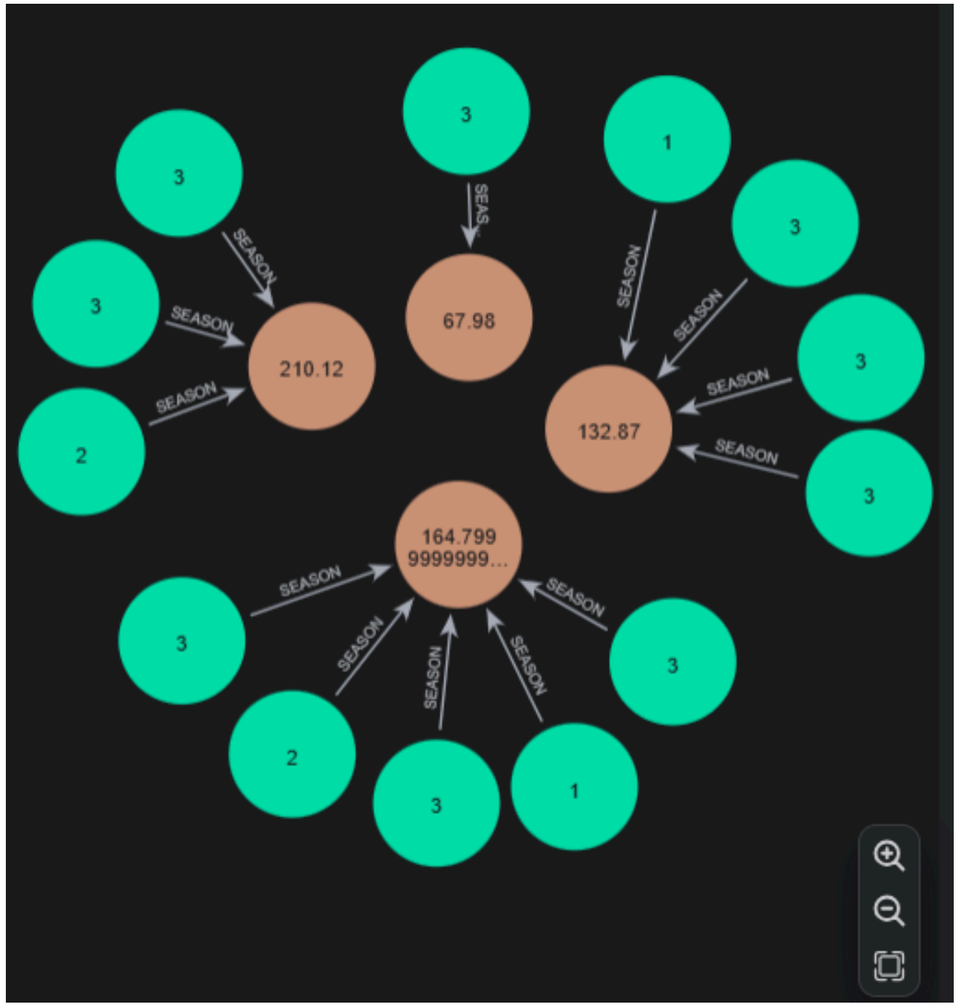
Truong Vuong

IMPORTANT DATA QUERIES AND REPORT

Booking Trends by Season Report: Analyzes seasonal demand and calculates the TotalOwnerRevenue, which represents the total revenue for space owners, excluding the sysFee. This helps owners adjust pricing and promotions accordingly.

```
MATCH (b:Booking)-[:SEASON]->(s:Season)
WITH
  s.season AS Season,
  COUNT(b) AS TotalBookings,
  SUM(b.totalPrice - b.sysFee) AS TotalOwnerRevenue
RETURN
  Season,
  TotalBookings,
  TotalOwnerRevenue
ORDER BY TotalOwnerRevenue
```

	Season	TotalBookings	TotalOwnerRevenue
1	"Spring"	1	66.0
2	"Winter"	5	182.0
3	"Summer"	3	204.0
4	"Fall"	4	254.0



Truong Vuong

IMPORTANT DATA QUERIES AND REPORT

Customer Feedback Sentiment Report: This query joins feedback data with the corresponding space information, then groups reviews by space. Reviews are categorized into positive (rating ≥ 4), neutral (≥ 3 and < 4), or negative (< 3), and the query calculates the number and percentage of each sentiment type, along with the average rating. The report provides insights into customer experience and highlights areas needing improvement.

```
MATCH (f:Feedback)-[:FEEDBACK_FOR]->(s:Space)
OPTIONAL MATCH (f)-[:RATING_GIVEN]->(r:Rating)
OPTIONAL MATCH (s)-[:LOCATED_IN]->(l:Location)
WITH
  s.name AS SpaceName,
  COALESCE(l.name, 'Unknown') AS Location,
  COUNT(f) AS TotalReviews,
  COUNT(CASE WHEN TOFLOAT(r.value) >= 4 THEN 1 END) AS PositiveReviews,
  COUNT(CASE WHEN TOFLOAT(r.value) >= 3 AND TOFLOAT(r.value) < 4 THEN 1 END) AS NeutralReviews,
  COUNT(CASE WHEN TOFLOAT(r.value) < 3 THEN 1 END) AS NegativeReviews,
  ROUND(AVG(TOFLOAT(r.value)), 2) AS AverageRating
RETURN
  SpaceName,
  Location,
  TotalReviews,
  CASE WHEN TotalReviews = 0 THEN 0 ELSE ROUND(PositiveReviews * 100.0 / TotalReviews, 2) END AS PositiveSentimentPercentage,
  CASE WHEN TotalReviews = 0 THEN 0 ELSE ROUND(NeutralReviews * 100.0 / TotalReviews, 2) END AS NeutralSentimentPercentage,
  CASE WHEN TotalReviews = 0 THEN 0 ELSE ROUND(NegativeReviews * 100.0 / TotalReviews, 2) END AS NegativeSentimentPercentage,
  AverageRating
ORDER BY SpaceName ASC
```

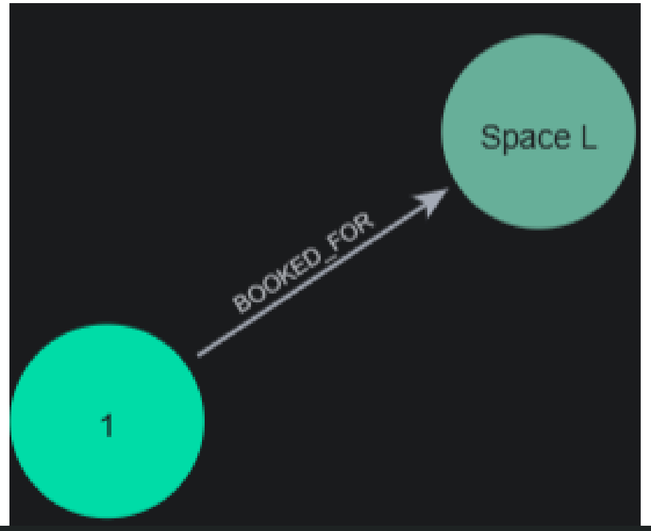
	SpaceName	Location	TotalReviews	PositiveSentiment	NeutralSentiment	NegativeSentimen	AverageRating
1	"Space A"	"Downtown"	2	50.0	50.0	0.0	3.5
2	"Space C"	"Midtown"	3	33.33	0.0	66.67	2.67
3	"Space D"	"Suburb"	14	71.43	14.29	14.29	3.86
4	"Space E"	"City Center"	21	61.9	19.05	19.05	3.62

Truong Vuong

IMPORTANT DATA QUERIES AND REPORT

Space Demand and Performance Report: The query gathers booking data for each space, calculating total revenue, total bookings, and average hourly rate. Based on booking volume, each space is classified as having "High", "Moderate", or "Low Demand." This report supports performance analysis, helps optimize space utilization, and informs pricing or promotional adjustments.

```
MATCH (b:Booking)-[:BOOKED_FOR]->(s:Space)
OPTIONAL MATCH (s)-[:LOCATED_IN]->(l:Location)
WITH
  s.name AS SpaceName,
  COALESCE(l.name, 'Unknown') AS Location,
  SUM(b.totalPrice) AS TotalRevenue,
  COUNT(b) AS TotalBookings,
  ROUND(AVG(s.hourlyRate), 2) AS AverageBookingRate
RETURN
  SpaceName,
  Location,
  TotalRevenue,
  TotalBookings,
  AverageBookingRate,
  CASE
    WHEN TotalBookings >= 5 THEN 'High Demand'
    WHEN TotalBookings >= 2 THEN 'Moderate Demand'
    ELSE 'Low Demand'
  END AS PerformanceStatus
ORDER BY TotalBookings DESC
```



	SpaceName	Location	TotalRevenue	TotalBookings	AverageBookingRa	PerformanceStatu
1	"Space D"	"Suburb"	425.45	5	5.0	"High Demand"
2	"Space B"	"Uptown"	273.9	3	10.0	"Moderate Demand"
3	"Space I"	"West Side"	55.62	1	27.0	"Low Demand"
4	"Space J"	"South Side"	46.35	1	15.0	"Low Demand"
5	"Space K"	"North Side"	92.7	1	30.0	"Low Demand"
6	"Space L"	"Central Park"	36.05	1	35.0	"Low Demand"

Cassandra Chang

TRANSACTIONS

7

Feedback Submission:

Information containing the feedback from the customer will be stored in the feedback collection as a document containing elements such as rating and review.

OPERATION: INSERTING A FEEDBACK FOR A SPACE

8

Rental Rate Adjustments:

Space owners can modify the rental rates (hourly, half-day, or full-day) for their listed spaces. This will be reflected onto the space collection.

OPERATION: UPDATING THE RENTAL RATES FOR A SPACE

9

User Profile Updates:

Users (owner or customer) can update their personal details (e.g., name, contact info, preferences). This update is only possible if the individual is a user of the platform.

OPERATION: UPDATING USER PROFILE DETAILS

TRANSACTIONS

FEEDBACK SUBMISSION

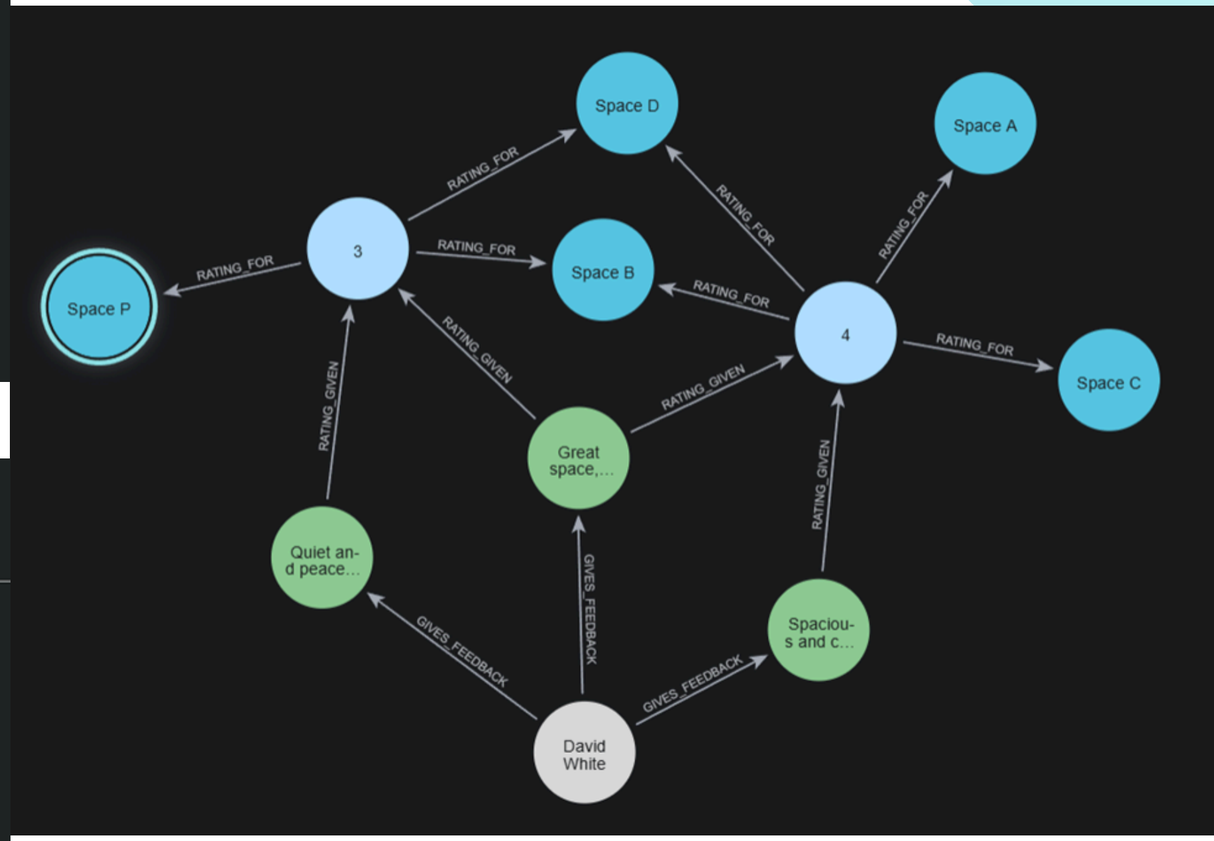
Query:

```
MERGE (f1:Feedback {reviews: 'Great space, very clean and well-maintained.'})
WITH f1
MATCH (u1:User {id: 6})
MERGE (u1)-[:GIVES_FEEDBACK]->(f1)
WITH f1,u1
MATCH (s1:Space {id: 16})
MERGE (f1)-[:FEEDBACK_FOR]->(s1)
MERGE (r1:Rating {value: 3})
MERGE (f1)-[:RATING_GIVEN]->(r1)
MERGE (r1)-[:RATING_FOR]->(s1)
SET s1.rating = (s1.rating + r1.value) / 2
```

Tabular Result:

a	b	c	d
(:User:Owner:Customer { createdAt: "2023-01-01T00:00:00Z", address: "987 Cedar St", preferedRange: 30.0, totalSpent: 67.98, phone: "7897897890", name: "David White", id: 6, numOfBookings: 1, email: "davidw@example.com", updatedAt: "2023-01-01T00:00:00Z" })	[:GIVES_FEEDBACK]	(:Feedback { reviews: "Great space, very clean and well-maintained."})	[:RATING_GIVEN]

Graph Result:



TRANSACTIONS

RENTAL RATE ADJUSTMENT

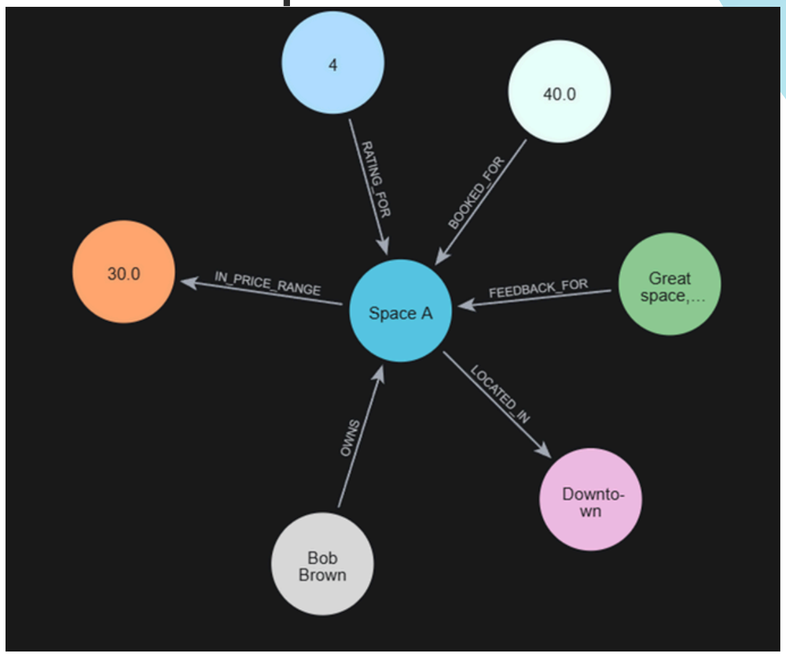
Query:

```
MATCH (s:Space {id: 1})
SET
  s.hourlyRate = 30.0,
  s.halfdayRate = 120.0,
  s.fulldayRate = 200.0,
  s.updatedAt = datetime()

// Remove old price range relation
WITH s
OPTIONAL MATCH (s)-[r:IN_PRICE_RANGE]->()
DELETE r

// Calculate new range and connect to PriceRange
WITH s, s.hourlyRate - (s.hourlyRate % 5) AS minPrice
WITH s, minPrice, minPrice + 5 AS maxPrice
MERGE (pr:PriceRange {minPrice: minPrice, maxPrice: maxPrice})
  ON CREATE SET pr.numOfBookings = 0, pr.totalSpent = 0.0
MERGE (s)-[:IN_PRICE_RANGE]->(pr)
```

Graph Result:



Tabular Result:

```
(:User:Owner:Customer {
  createdAt: "2023-01-01T00:0
0:00Z",
  address: "321 Birch St",
  preferedRange: 15.0,
  totalSpent: 15.45,
  phone: "3213213214",
  name: "Bob Brown",
  id: 4,
  numOfBookings: 1,
  email: "bobb@example.com",
  updatedAt: "2023-01-01T00:0
0:00Z"
})
```

```
[:OWNS]
```

```
(:Space {
  numRooms: 3,
  address: "123 Space St",
  rating: 2,
  remark: "Great space for eve
nts",
  createdAt: "2025-01-01T00:0
0:00Z",
  halfdayRate: 120.0,
  size: 100.0,
  rentFrom: "08:00:00",
  name: "Space A",
  id: 1,
  hourlyRate: 30.0,
  updatedAt: 2025-04-15T15:56:
52.042000000Z,
  status: "open",
  rentTo: "17:00:00",
  fulldayRate: 200.0
```

```
(:Location { name: "Downtow
n"})
```

TRANSACTIONS

USER PROFILE UPDATE

Query:

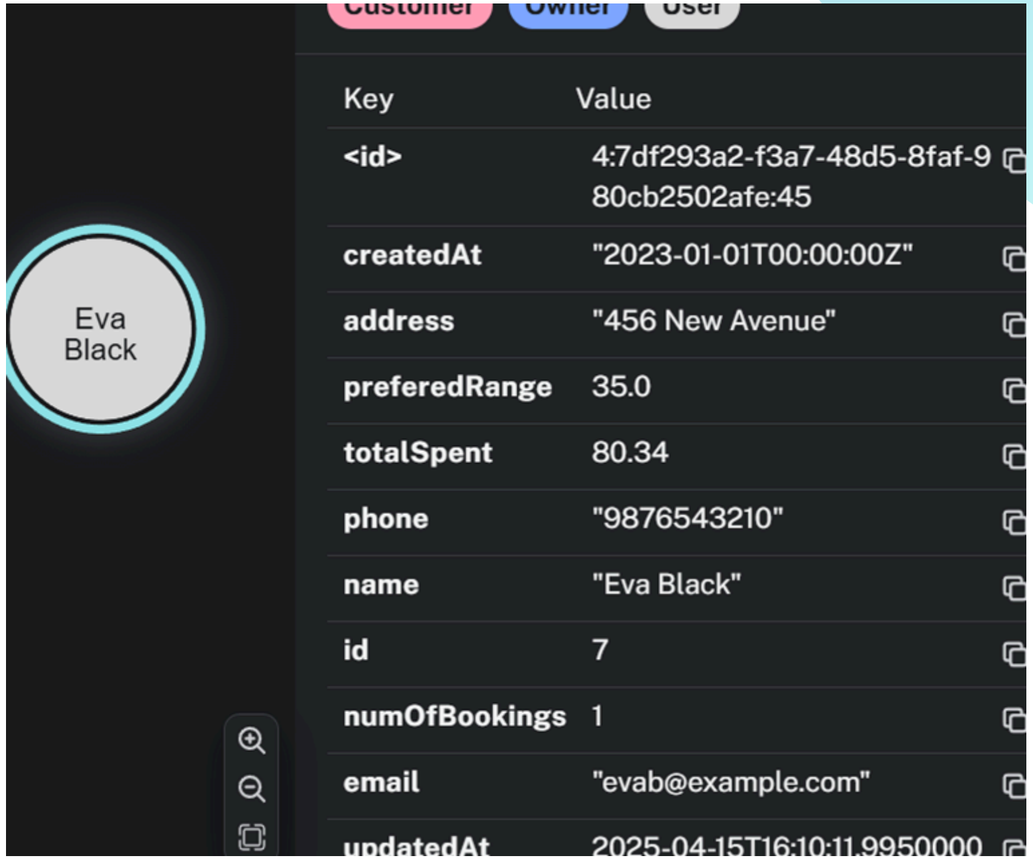
```
MATCH (u:User {id: 7})
SET
  u.address = '456 New Avenue',
  u.phone = '9876543210',
  u.updatedAt = datetime()
```

Tabular Result:

u

```
(:User:Owner:Customer {
  createdAt: "2023-01-01T00:00:00Z",
  address: "456 New Avenue",
  preferedRange: 35.0,
  totalSpent: 80.34,
  phone: "9876543210",
  name: "Eva Black",
  id: 7,
  numOfBookings: 1,
  email: "evab@example.com",
  updatedAt: 2025-04-15T16:10:11.995000000Z
})
```

Graph Result:



IMPORTANT DATA QUERIES AND REPORT

Longest Duration Booking Ever:

Report tells us who the higher spender is for each space. This was the system can gain insights for incentives or promotions to attract the spender again.

Customer Booking History:

This report fetches all bookings made by a specific customer, including space details, booking dates, facilities used, and total charges.

Space Owner Earnings Report:

This report shows earnings for a specific space owner, grouped by their spaces, including space name, total bookings, earnings, and platform fees.

Space Utilization Report:

This report helps the user (owner) understand which spaces have not been booked in the a month. This helps the owner get insights about the underperforming units and make changes to it.

Most Frequently Cancelled Spaces

Identify listings with high cancellation counts and update if the owners should be given a warning before flagging them.

IMPORTANT DATA QUERIES AND REPORT

Longest Duration Booking Ever:

Report tells us who the higher spender is for each space. This was the system can gain insights for incentives or promotions to attract the spender again.

```
// Step 1: Get the longest booking duration per space
MATCH (b:Booking)-[:BOOKED_FOR]->(s:Space)
WITH s, MAX(b.totalDuration) AS maxDuration

// Step 2: Find the bookings that match that max duration for each space
MATCH (b:Booking)-[:BOOKED_FOR]->(s)
WHERE b.totalDuration = maxDuration

// Step 3: Get the user who made the booking
MATCH (u:User)-[:BOOKED_BY]->(b)

RETURN
  s.name AS space,
  b.totalDuration AS hours,
  b.bookingDate AS date,
  elementId(b) AS bookingId,
  u.name AS bookedBy,
  b.totalPrice AS amountSpent
ORDER BY hours DESC
```

Customer Booking History:

This report fetches all bookings made by a specific customer, including space details, booking dates, facilities used, and total charges.

```
MATCH (u:User {name: "John Doe"})-[b:BOOKED_BY]->(c:Booking)-[:BOOKED_FOR]->(s:Space)
WITH u, b, c, s, labels(c) AS labelList,
     [label IN labels(c) WHERE label <> "Booking"][0] AS bookingStatus
RETURN
  u.name AS customerName,
  elementId(b) AS bookingId,
  s.name AS spaceName,
  c.totalPrice AS totalPrice,
  c.totalDuration AS stayDurationInHours,
  c.bookingDate AS bookingDate,
  c.updatedAt AS updatedAt,
  bookingStatus
```

IMPORTANT DATA QUERIES AND REPORT

Most Frequently Cancelled Spaces

Identify listings with high cancellation counts and update if the owners should be given a warning before flagging them.

```
MATCH (b:Booking:Cancelled)-[:BOOKED_FOR]->(s:Space)
WITH s.name AS spaceName, COUNT(*) AS cancelledCount
RETURN spaceName,
       cancelledCount,
       CASE
           WHEN cancelledCount > 5 THEN "Yes"
           ELSE "No"
       END AS sendFlagWarning
ORDER BY cancelledCount DESC
```

Cassandra Chang

IMPORTANT DATA QUERIES AND REPORT

Space Owner Earnings Report: This report shows earnings for a specific space owner, grouped by their spaces, including space name, total bookings, earnings, and platform fees.

```
MATCH (owner:User {name: "Bob Brown"})-[:OWNS]->(s:Space)
MATCH (b:Booking)-[:BOOKED_FOR]->(s)
WHERE b.spacePrice IS NOT NULL
WITH s.name AS spaceName,
     COUNT(b) AS totalBookings,
     SUM(b.spacePrice) AS totalEarnings,
    ROUND(AVG(b.totalDuration), 2) AS avgDurationHours
RETURN
  spaceName,
  totalBookings,
  totalEarnings,
  avgDurationHours
ORDER BY totalEarnings DESC
```

	spaceName	totalBookings	totalEarnings	avgDurationHours
1	"Space G"	1	78.0	3.0
2	"Space J"	1	45.0	3.0
3	"Space A"	1	40.0	2.0
4	"Space M"	1	22.0	1.0
5	"Space D"	1	15.0	3.0

IMPORTANT DATA QUERIES AND REPORT

Space Utilization Report: This report helps the user (owner) understand which spaces have not been booked in the a month. This helps the owner get insights about the underperforming units and make changes to it.

```
WITH date() AS today, date() - duration({months: 1}) AS oneMonthAgo

MATCH (owner:User {name: "Chris Green"})-[:OWNS]->(s:Space)
OPTIONAL MATCH (s)<-[:BOOKED_FOR]-(b:Booking)
WHERE b.bookingDate IS NOT NULL

WITH s, MAX(b.bookingDate) AS lastBookingDate, oneMonthAgo
WHERE lastBookingDate IS NULL OR lastBookingDate < oneMonthAgo

OPTIONAL MATCH (s)-[:LOCATED_IN]->(l:Location)

RETURN
  s.name AS spaceName,
  l.name AS location,
  s.status AS status,
  s.hourlyRate as hourlyRate,
  s.halfdayRate as halfDayRate,
  s.fulldayRate as fullDayRate,
  lastBookingDate
```

	spaceName	location	status	hourlyRate	halfDayRate	fullDayRate	lastBookingDate
1	"Space N"	"Tech Hub"	"open"	24.0	120.0	215.0	null
2	"Space Q"	"Cultural District"	"open"	26.0	135.0	230.0	null

TRANSACTIONS

Aymen Zubair Qureshi

Feedback Removal

Allows space owners to review customer feedback and request the removal of inappropriate or misleading reviews.

Feedback:

```
(:Space {
  numRooms: 5,
  listed_at: 2024-11-25,
  address: "456 Space St",
  rating: 2,
  remark: "Spacious and modern",
  createdAt: "2025-01-01T00:00:00Z",
  halfdayRate: 100.0,
  size: 200.0,
  date_listed: 2025-04-10T00:00:00Z,
  rentFrom: "08:00:00",
  name: "Space B",
  id: 2,
  hourlyRate: 10.0,
  updatedAt: "2025-01-01T00:00:00Z",
  status: "open",
  rentTo: "17:00:00",
  fulldayRate: 150.0
})
```

Query:

```
1 MATCH (s:Space {id: 2})
2 DETACH DELETE s
```

Results:

```
neo4j$ MATCH (s:Space {id: 2}) RETURN s
```

No changes, no records

TRANSACTIONS

Aymen Zubair Qureshi

New space listing:

The space owner can create a new listing by providing important details of the new space.

```
MATCH (owner:Owner {name: "Chris Green"})
CREATE (space:Space {
  id: 11,
  name: "Space Y",
  address: "101 Innovation Blvd",
  rating: 0,
  remark: "Creative open space",
  createdAt: datetime("2025-04-16T00:00:00Z"),
  updatedAt: datetime("2025-04-16T00:00:00Z"),
  status: "open",
  hourlyRate: 30.0,
  fulldayRate: 220.0,
  halfdayRate: 120.0,
  rentFrom: "09:00:00",
  rentTo: "17:00:00",
  size: 180.0,
  numRooms: 3
})
MERGE (owner)-[:OWNS]->(space)
```

Editing the bookings:

The customer can modify/cancel the booking dates.

```
MATCH (b:Booking {id: 3})
SET b.bookingDate = "2025-04-01"
RETURN b
```

IMPORTANT DATA QUERIES AND REPORT

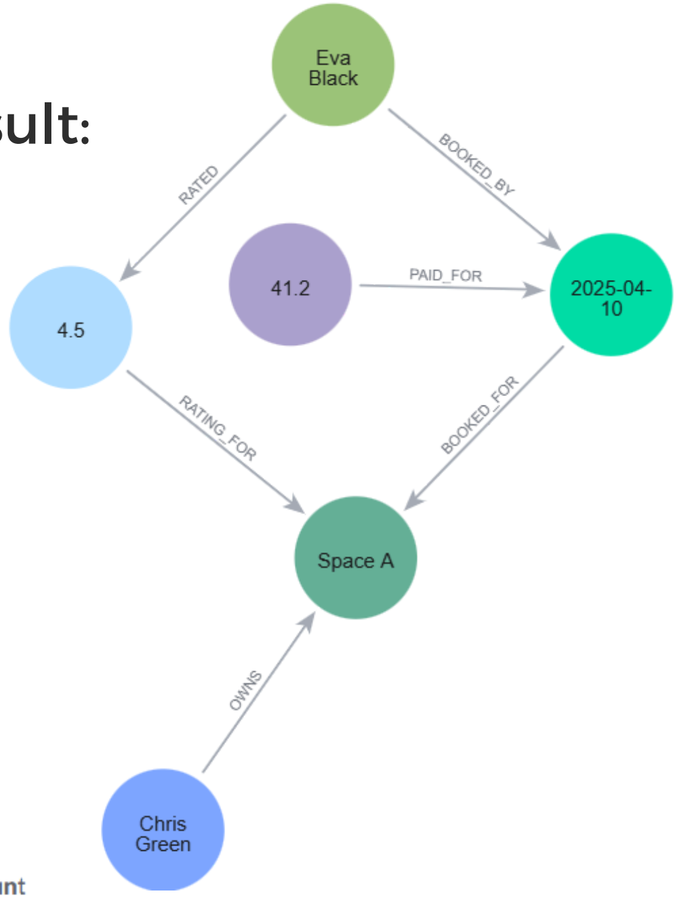
Retrieve a specific Bookings Details: retrieves specific details of bookings, including the customer's name, the space they booked, the booking date, and the payment, providing a clear overview of the booking information.

Query:

```
// Create Relationships
CREATE (owner)-[:OWNS]->(space)
CREATE (user)-[:BOOKED_BY]->(booking)
CREATE (booking)-[:BOOKED_FOR]->(space)
CREATE (user)-[:RATED]->(rating)
CREATE (rating)-[:RATING_FOR]->(space)
CREATE (payment)-[:PAID_FOR]->(booking)
WITH owner, space, booking, user, rating, payment
// Match relationships for visualization
MATCH (owner)-[r1:OWNS]->(space)
MATCH (user)-[r2:BOOKED_BY]->(booking)
MATCH (booking)-[r3:BOOKED_FOR]->(space)
MATCH (user)-[r4:RATED]->(rating)
MATCH (rating)-[r5:RATING_FOR]->(space)
MATCH (payment)-[r6:PAID_FOR]->(booking)
```

```
// Return nodes and their relationships
RETURN owner, space, booking, user, rating, payment,
       r1 AS owned_by_relation,
       r2 AS booked_by_relation,
       r3 AS booked_for_relation,
       r4 AS rated_by_relation,
       r5 AS rating_for_relation,
       r6 AS paid_for_relation
```

Graph Result:



Tabular Result:

Owner	Space	Location	BookingDate	User	Rating	PaymentAmount
"Chris Green"	"Space A"	"Downtown"	"2025-04-10"	"Eva Black"	4.5	41.2

IMPORTANT DATA QUERIES AND REPORT

Owner Review Feedback: This query shows the feedback given by users for spaces owned by a specific owner. It helps the owner view ratings and comments to understand user satisfaction and make improvements if needed.

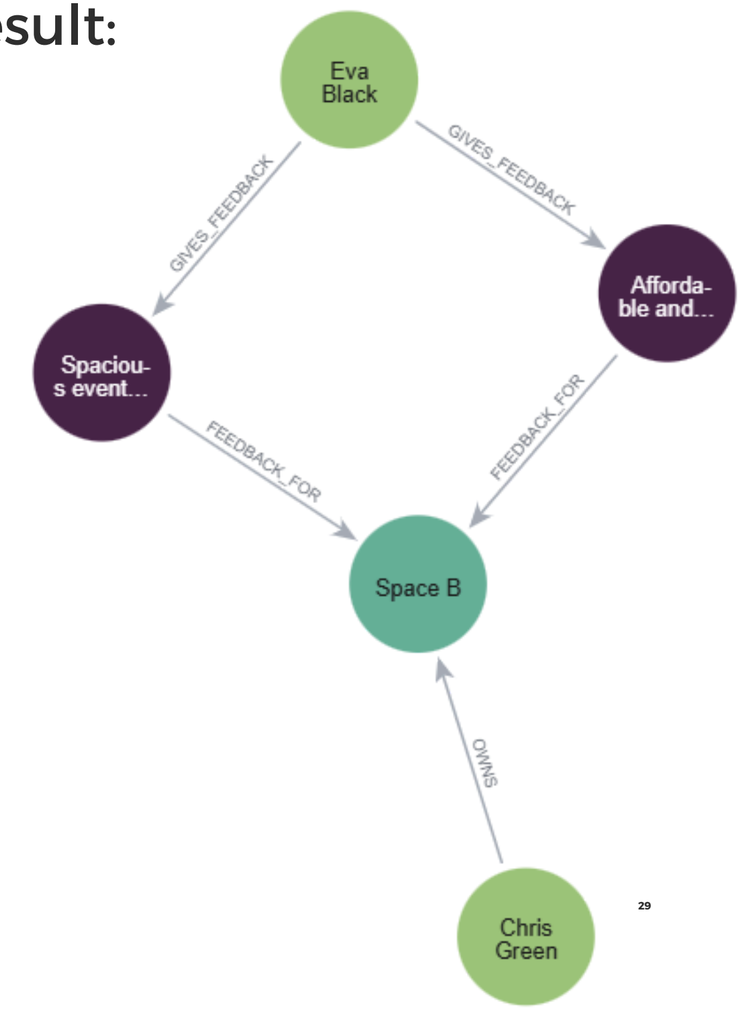
Query:

```
MATCH (owner: Owner {name: "Chris Green"})-[r1:OWNS]->(space: Space {name: "Space B"})
MATCH (user: User {name: "Eva Black"})-[r2:GIVES_FEEDBACK]->(feedback:Feedback) MERGE
(feedback)-[r3: FEEDBACK_FOR] ->(space)
MERGE (user)-[r4: GIVES_FEEDBACK]->(feedback)
RETURN owner, space, user, feedback, r1, r2, r3, r4
```

Tabular Result:

owner	space	user	feedback	r1
(:User:Owner { createdAt: "2023-01-01T00:00:00Z", address: "654 Pine St", preferedRange: 25.0, totalSpent: 0.0, phone: "4564564567", name: "Chris Green", id: 5, numOfBookings: 0, email: "chrisg@example.com", updatedAt: "2023-01-01T00:00:00Z" })	(:Space { numRooms: 5, listed_at: 2024-11-25, address: "456 Space St", rating: 2, remark: "Spacious and modern", createdAt: "2025-01-01T00:00:00Z", halfdayRate: 100.0, size: 200.0, date_listed: 2025-04-10T00:00:00Z, rentFrom: "08:00:00", name: "Space B", id: 2, hourlyRate: 10.0, updatedAt: "2025-01-01T00:0	(:User:Owner:Customer { createdAt: "2023-01-01T00:00:00Z", address: "456 New Avenue", preferedRange: 35.0, totalSpent: 80.34, phone: "9876543210", name: "Eva Black", id: 7, numOfBookings: 1, email: "evab@example.com", updatedAt: 2025-04-15T16:10:11.995000000Z })	(:Feedback { reviews: "Affordable and practical, great value."})	[:OWNS]

Graph Result:



IMPORTANT DATA QUERIES AND REPORT

Most Popular Spaces by Booking Frequency: This query identifies the most popular spaces by counting how often each space has been booked and ranks them accordingly.

Query:

```
// Get the most booked spaces by frequency
MATCH (space:Space)<-[r:BOOKED_FOR]-(booking:Booking)
RETURN space.name AS space_name, COUNT(r) AS booking_frequency
ORDER BY booking_frequency DESC
LIMIT 10

```

Tabular Result:

	space_name	booking_frequenc
1	"Space A"	8
2	"Space K"	1
3	"Space L"	1
4	"Space M"	1
5	"Space J"	1
6	"Space B"	1
7	"Space C"	1
8	"Space D"	1
9	"Space E"	1

IMPORTANT DATA QUERIES AND REPORT

Recently listed space: This identifies the most recently listed spaces by filtering spaces based on the owner's name and the listing date, helping to track new space listings.

Query:

```
MATCH (owner:Owner)-[:OWNS]->(space:Space)
WHERE space.createdAt IS NOT NULL
RETURN space.name AS Name,
       space.createdAt AS Created_At,
       owner.name AS Owner_Name
ORDER BY space.createdAt DESC
LIMIT 10
```

Tabular Result:

	Name	Created_At	Owner_Name
1	"Space Z"	"2025-01-01T00:00:00Z"	"John Doe"
2	"Space A"	"2025-01-01T00:00:00Z"	"Bob Brown"
3	"Space B"	"2025-01-01T00:00:00Z"	"Chris Green"
4	"Space C"	"2025-01-01T00:00:00Z"	"David White"
5	"Space D"	"2025-01-01T00:00:00Z"	"Bob Brown"
6	"Space E"	"2025-01-01T00:00:00Z"	"Chris Green"
7	"Space F"	"2025-01-01T00:	"David White"

IMPORTANT DATA QUERIES AND REPORT

Popular Booking Time Slots: This query finds the most popular booking time slots by counting how often each time slot was booked.

Query:

```
1 MATCH (b:Booking)-[:BOOKED_FOR]->(s:Space)
2 WHERE b.time_slot IS NOT NULL
3 WITH b.time_slot AS time_slot, count(*) AS total_bookings
4 RETURN time_slot, total_bookings
5 ORDER BY total_bookings DESC
6 LIMIT 5
```

Tabular Result:

Table RAW

time_slot	total_bookings
"12:00 PM - 1:0 0 PM"	22

COMPARISON AND CONCLUSION

Model	Benefit	Drawback	Use Cases
Relational	Good for transactions (booking, payment)	Difficult to store semi-structured information such as space and facilities	Booking and Payment
Document	Fast Retrieval using a single document	Data consistency (many update operations to enforce consistency)	Space and Facilities
Graph	Instance relationship (good for complex relationships)	Intensive Write Operations (for data summarization)	Area, Address and Location

Choice - ***Relational Database Management System (RDBMS)***


THANK YOU