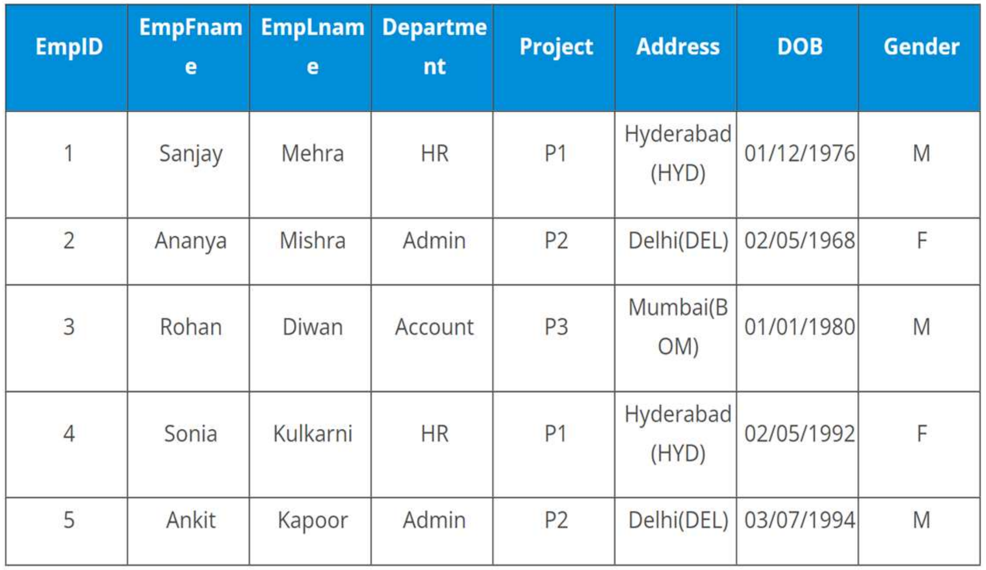
Tutorial 6: Data Repositories and Querying

Outline

- **SQL query**
- **NoSQL query**

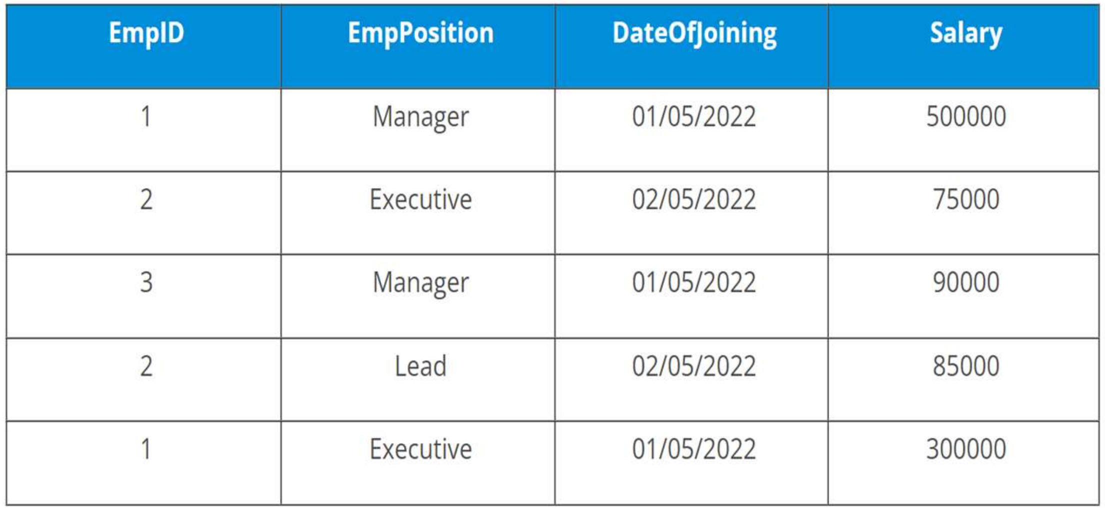
SQL Query

EmployeeInfo Table:



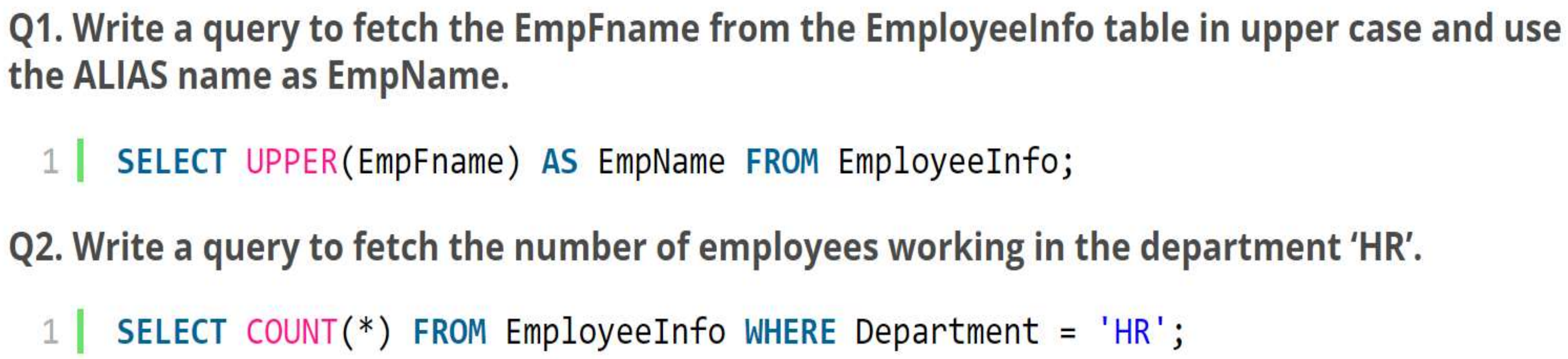
EmpID	EmpFname	EmpLname	Department	Project	Address	DOB	Gender
1	Sanjay	Mehra	HR	P1	Hyderabad (HYD)	01/12/1976	M
2	Ananya	Mishra	Admin	P2	Delhi(DEL)	02/05/1968	F
3	Rohan	Diwan	Account	P3	Mumbai(BOM)	01/01/1980	M
4	Sonia	Kulkarni	HR	P1	Hyderabad (HYD)	02/05/1992	F
5	Ankit	Kapoor	Admin	P2	Delhi(DEL)	03/07/1994	M

EmployeePosition Table:



EmpID	EmpPosition	DateOfJoining	Salary
1	Manager	01/05/2022	500000
2	Executive	02/05/2022	75000
3	Manager	01/05/2022	90000
2	Lead	02/05/2022	85000
1	Executive	01/05/2022	300000

SQL



Q1. Write a query to fetch the EmpFname from the EmployeeInfo table in upper case and use the ALIAS name as EmpName.

```sql
SELECT UPPER(EmpFname) AS EmpName FROM EmployeeInfo;
```

Q2. Write a query to fetch the number of employees working in the department ‘HR’.

```sql
SELECT COUNT(*) FROM EmployeeInfo WHERE Department = 'HR';
```

SQL

- **Q3. Write a query to get the current date.**
  - You can write a query as follows in SQL Server:
    - SELECT **GETDATE();**
  - You can write a query as follows in MySQL:
    - SELECT **SYSTDATE();**

SQL

Q4. Write a query to retrieve the first four characters of EmpLname from the EmployeeInfo table.

```
SELECT SUBSTRING(EmpLname, 1, 4) FROM EmployeeInfo;
```

Q5. Write a query to fetch only the place name(string before brackets) from the Address column of EmployeeInfo table.

Using the MID function in MySQL

```
SELECT MID(Address, 0, LOCATE('(',Address)) FROM EmployeeInfo;
```

Using SUBSTRING

```
SELECT SUBSTRING(Address, 1, CHARINDEX('(',Address)) FROM EmployeeInfo;
```

SQL

Q6. Write a query to create a new table which consists of data and structure copied from the other table.

Using the SELECT INTO command:

```sql
SELECT * INTO NewTable FROM EmployeeInfo WHERE 1 = 0;
```

Q7. Write q query to find all the employees whose salary is between 50000 to 100000.

```sql
SELECT * FROM EmployeePosition WHERE Salary BETWEEN '50000' AND '100000';
```

Q8. Write a query to find the names of employees that begin with 'S'

```sql
SELECT * FROM EmployeeInfo WHERE EmpFname LIKE 'S%';
```

SQL

Q9. Write a query to fetch top N records.

By using the TOP command in SQL Server:

```sql
SELECT TOP N * FROM EmployeePosition ORDER BY Salary DESC;
```

By using the LIMIT command in MySQL:

```sql
SELECT * FROM EmpPosition ORDER BY Salary DESC LIMIT N;
```

Q10. Write a query to retrieve the EmpFname and EmpLname in a single column as “FullName”. The first name and the last name must be separated with space.

```sql
SELECT CONCAT(EmpFname, ' ', EmpLname) AS 'FullName' FROM EmployeeInfo;
```

SQL

Q12. Write a query to fetch all the records from the EmployeeInfo table ordered by EmpLname in descending order and Department in the ascending order.

To order the records in ascending and descnding order, you have to use the ORDER BY statement in SQL.

```
SELECT * FROM EmployeeInfo ORDER BY EmpFname desc, Department asc;
```

Q13. Write a query to fetch details of employees whose EmpLname ends with an alphabet 'A' and contains five alphabets.

To fetch details mathcing a certain value, you have to use the LIKE operator in SQL.

```
SELECT * FROM EmployeeInfo WHERE EmpLname LIKE '____a';
```

Q14. Write a query to fetch details of all employees excluding the employees with first names, "Sanjay" and "Sonia" from the EmployeeInfo table.

```
SELECT * FROM EmployeeInfo WHERE EmpFname NOT IN ('Sanjay','Sonia');
```

SQL

Q15. Write a query to fetch details of employees with the address as "DELHI(DEL)".

```sql
SELECT * FROM EmployeeInfo WHERE Address LIKE 'DELHI(DEL)%';
```

Q16. Write a query to fetch all employees who also hold the managerial position.

```sql
SELECT E.EmpFname, E.EmpLname, P.EmpPosition
FROM EmployeeInfo E INNER JOIN EmployeePosition P ON
E.EmpID = P.EmpID AND P.EmpPosition IN ('Manager');
```

Q17. Write a query to fetch the department-wise count of employees sorted by department's count in ascending order.

```sql
SELECT Department, count(EmpID) AS EmpDeptCount
FROM EmployeeInfo GROUP BY Department
ORDER BY EmpDeptCount ASC;
```

SQL

Q18. Write a query to calculate the even and odd records from a table.

To retrieve the even records from a table, you have to use the MOD() function as follows:

```sql
SELECT EmpID FROM (SELECT rowno, EmpID from EmployeeInfo) WHERE MOD(rowno,2)=0;
```

Similarly, to retrieve the odd records from a table, you can write a query as follows:

```sql
SELECT EmpID FROM (SELECT rowno, EmpID from EmployeeInfo) WHERE MOD(rowno,2)=1;
```

Q19. Write a SQL query to retrieve employee details from EmployeeInfo table who have a date of joining in the EmployeePosition table.

```sql
SELECT * FROM EmployeeInfo E
WHERE EXISTS
(SELECT * FROM EmployeePosition P WHERE E.EmpId = P.EmpId);
```

SQL

Q20. Write a query to retrieve two minimum and maximum salaries from the EmployeePosition table.

To retrieve two minimum salaries, you can write a query as below:

```sql
SELECT DISTINCT Salary FROM EmployeePosition E1
 WHERE 2 >= (SELECTCOUNT(DISTINCT Salary)FROM EmployeePosition E2
  WHERE E1.Salary >= E2.Salary) ORDER BY E1.Salary DESC;
```

To retrieve two maximum salaries, you can write a query as below:

```sql
SELECT DISTINCT Salary FROM EmployeePosition E1
 WHERE 2 >= (SELECTCOUNT(DISTINCT Salary) FROM EmployeePosition E2
  WHERE E1.Salary <= E2.Salary) ORDER BY E1.Salary DESC;
```

SQL Practice

Q21. Write a query to find the Nth highest salary from the table without using TOP/limit keyword.

Q22. Write a query to retrieve duplicate records from a table.

Q23. Write a query to retrieve the list of employees working in the same department.

SQL Practice

Q24. Write a query to retrieve the last 3 records from the EmployeeInfo table.

Q25. Write a query to find the third-highest salary from the EmpPosition table.

SQL Practice

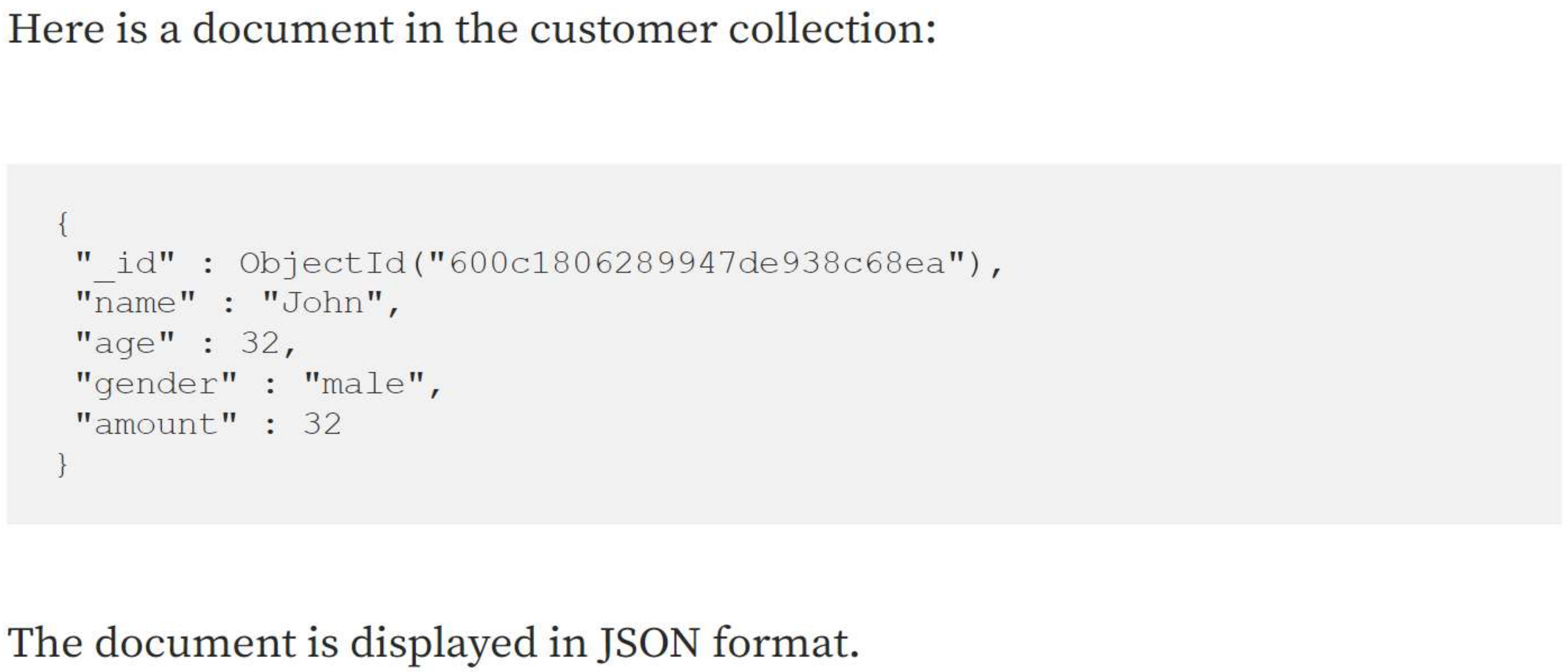
Q26. Write a query to display the first and the last record from the EmployeeInfo table.

To display the first record from the EmployeeInfo table, you can write a query as follows:

To display the last record from the EmployeeInfo table, you can write a query as follows:

NoSQL Query

- **Take MongoDB as example**



Here is a document in the customer collection:

```
{
 "_id" : ObjectId("600c1806289947de938c68ea"),
 "name" : "John",
 "age" : 32,
 "gender" : "male",
 "amount" : 32
}
```

The document is displayed in JSON format.

NoSQL

We want to see the document belongs to customer John so the name field needs to be specified in the find method.

```
> db.customer.find( {name: "John"} )

{ "_id" : ObjectId("600c1806289947de938c68ea"), "name" : "John", "age"
: 32, "gender" : "male", "amount" : 32 }
```

NoSQL

We can attach pretty method to make the document seem more appealing.

```
> db.customer.find( {name: "John"} ).pretty()


{
 "_id" : ObjectId("600c1806289947de938c68ea"),
 "name" : "John",
 "age" : 32,
 "gender" : "male",
 "amount" : 32
}
```

NoSQL

Query documents that belong to customers older than 40.

The condition is applied to age field using a logical operator. The “$gt” stands for “greater than” and is used as follows.

```
> db.customer.find( {age: {$gt:40}} ).pretty()

{
 "_id" : ObjectId("600c19d2289947de938c68ee"),
 "name" : "Jenny",
 "age" : 42,
 "gender" : "female",
 "amount" : 36
}
```

NoSQL

Query documents that belong to female customers who are younger than 25.

This example is like a combination of the previous two examples. Both conditions must be met so we use "and" logic to combine the conditions. It can be done by writing both conditions separated by comma.

```
> db.customer.find( {gender: "female", age: {$lt:25}} ).pretty()

{
 "_id" : ObjectId("600c19d2289947de938c68f0"),
 "name" : "Samantha",
 "age" : 21,
 "gender" : "female",
 "amount" : 41
}

{
 "_id" : ObjectId("600c19d2289947de938c68f1"),
 "name" : "Laura",
 "age" : 24,
 "gender" : "female",
 "amount" : 51
}
```

NoSQL

In this example, we will repeat the previous example in a different way. Multiple conditions can also be combined with “and” logic as below.

```
> db.customer.find( {$and :[ {gender: "female", age: {$lt:25}} ]}
).pretty()
```

The logic used for combining the conditions is indicated at the beginning. The remaining part is same as the previous example but we need to put the conditions in a list ([ ]).

NoSQL

Query customers who are either male or younger than 25.

This example requires a compound query with “or” logic. We just need to change “$and” to “$or”.

```
> db.customer.find( { $or: [ {gender: "male"}, {age: {$lt: 22}} ] })

{ "_id" : ObjectId("600c1806289947de938c68ea"), "name" : "John", "age"
: 32, "gender" : "male", "amount" : 32 }

{ "_id" : ObjectId("600c19d2289947de938c68ed"), "name" : "Martin",
"age" : 28, "gender" : "male", "amount" : 49 }

{ "_id" : ObjectId("600c19d2289947de938c68ef"), "name" : "Mike", "age"
: 29, "gender" : "male", "amount" : 22 }

{ "_id" : ObjectId("600c19d2289947de938c68f0"), "name" : "Samantha",
"age" : 21, "gender" : "female", "amount" : 41 }
```

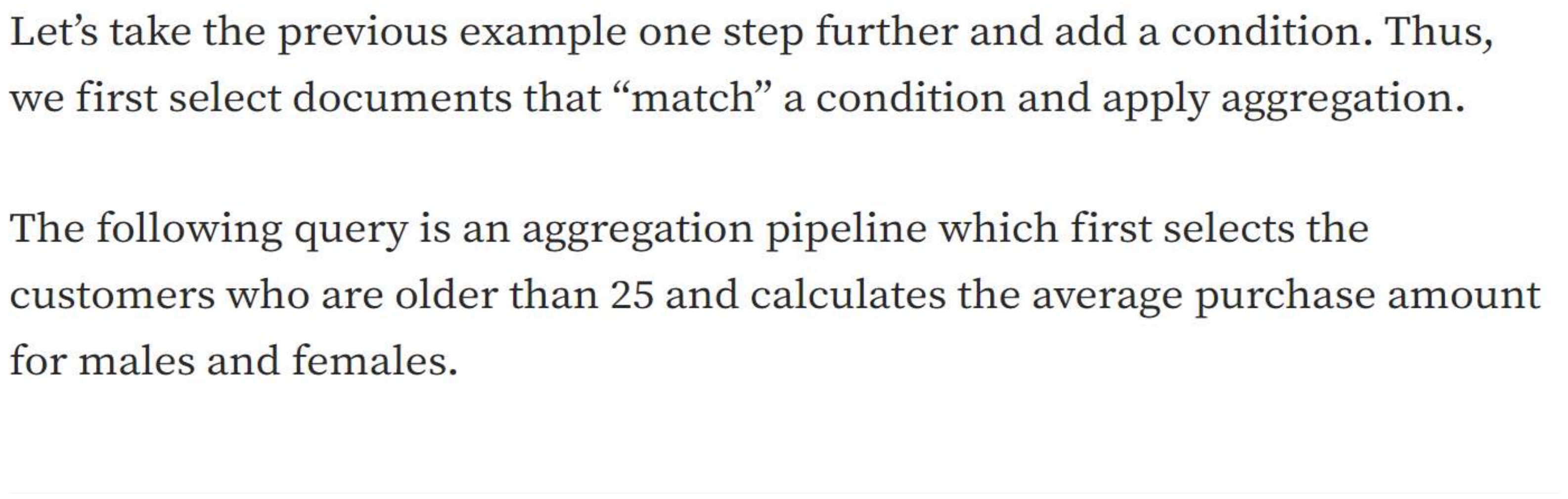
NoSQL

MongoDB allows for aggregating values while retrieving from the database. For instance, we can calculate the total purchase amount for males and females. The aggregate method is used instead of the find method.

```
> db.customer.aggregate([
... { $group: {_id: "$gender", total: {$sum: "$amount"} } }
... ])

{ "_id" : "female", "total" : 198 }
{ "_id" : "male", "total" : 103 }
```

NoSQL Practice



Let’s take the previous example one step further and add a condition. Thus, we first select documents that “match” a condition and apply aggregation.

The following query is an aggregation pipeline which first selects the customers who are older than 25 and calculates the average purchase amount for males and females.

NoSQL Practice

The query in the previous example contains only two groups so it is not necessary to sort the results. However, we might have queries that return several values. In such cases, sorting the results is a good practice.

We can sort the results of the previous query by the average amount in ascending order.