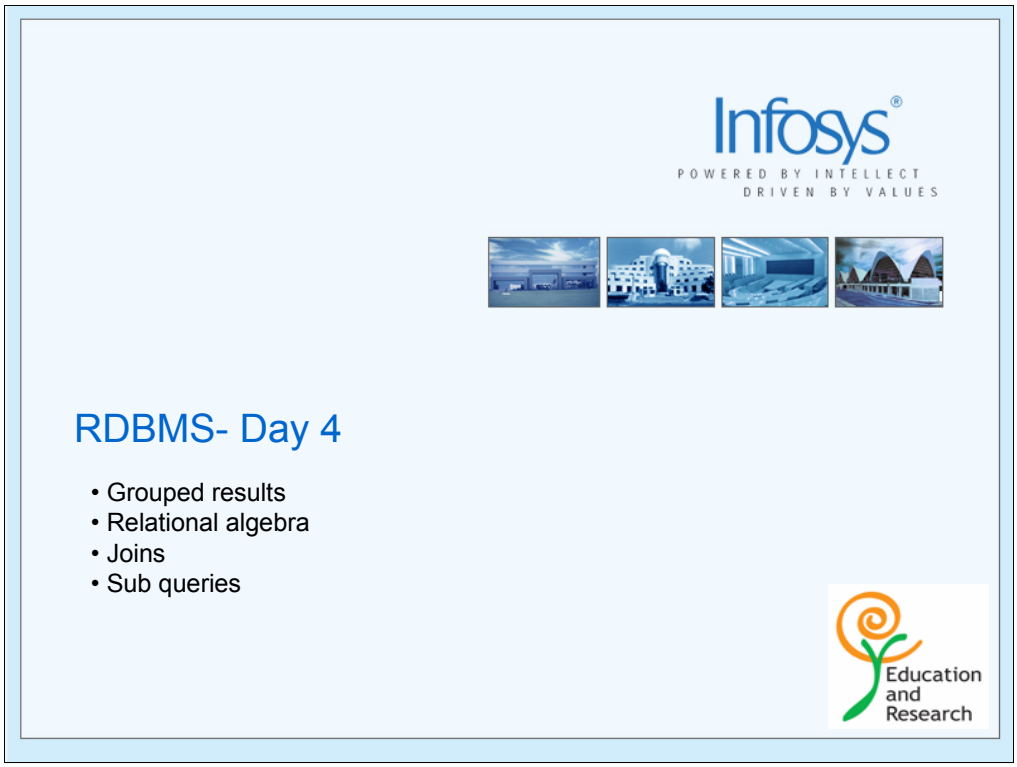


In today's session we will discuss about the concept of sub queries.

Infosys®
POWERED BY INTELLECT
DRIVEN BY VALUES

Grouped results

Education and Research

SQL - Using GROUP BY

- Related rows can be grouped together by **GROUP BY** clause by specifying a column as a grouping column.
- **GROUP BY** is associated with an aggregate function
- *To retrieve the total loan-amount of all loans taken by each Customer.*

```
SELECT Cust_ID, SUM(Amount_in_Dollars)
        FROM Customer_Loan
                GROUP BY Cust_ID;
```

In the output table all the rows with an identical value in the grouping column will be grouped together.

SQL – Group By

```
SELECT Cust_ID, SUM(Amount_in_Dollars) FROM Customer_Loan GROUP BY Cust_ID;
```

GROUP BY Cust_ID

Cust_ID	Loan_No	Amount_in_Dollars
101	1011	8755.00
103	2010	2555.00
104	2056	3050.00
103	2015	2000.00

Customer_Loan records from Customer_Loan table

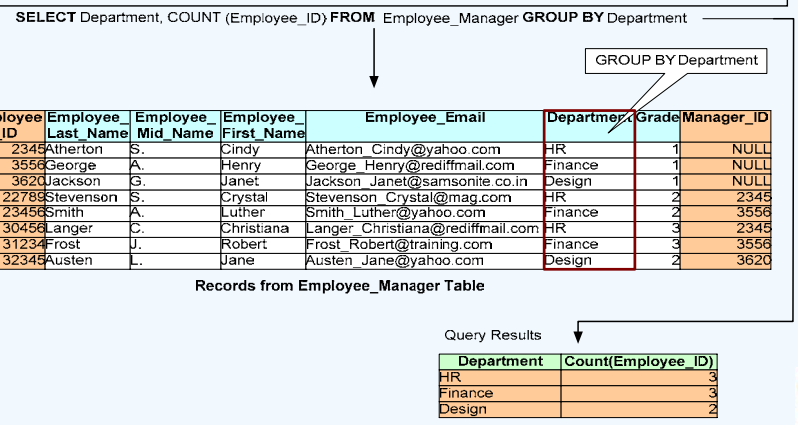
Query Results

Cust_ID	Sum(Amount_in_Dollars)
101	8755.00
103	4555.00
104	3050.00

SQL – Group BY

- *To retrieve Number of Employees in each Department*

```
SELECT  Department, COUNT (Employee_ID)
        FROM  Employee_Manager
              GROUP BY Department
```



Employee_ID	Employee_Last_Name	Employee_Mid_Name	Employee_First_Name	Employee_Email	Department	Grade	Manager_ID
2345	Atherton	S.	Cindy	Atherton_Cindy@yahoo.com	HR	1	NULL
3556	George	A.	Henry	George_Henry@rediffmail.com	Finance	1	NULL
3620	Jackson	G.	Janet	Jackson_Janet@samsonite.co.in	Design	1	NULL
22789	Stevenson	S.	Crystal	Stevenson_Crystal@mag.com	HR	2	2345
23456	Smith	A.	Luther	Smith_Luther@yahoo.com	Finance	2	3556
30456	Langer	C.	Christiana	Langer_Christiana@rediffmail.com	HR	3	2345
31234	Frost	J.	Robert	Frost_Robert@training.com	Finance	3	3556
32345	Austen	L.	Jane	Austen_Jane@yahoo.com	Design	2	3620

Records from Employee_Manager Table

Query Results

Department	Count(Employee_ID)
HR	3
Finance	3
Design	2

Retrieval using GROUP BY

Example:

Invalid SQL statement

```
SELECT Department, Manager_ID, COUNT(Employee_ID)
        FROM Employee_Manager
                GROUP BY Manager_ID;
```

Valid SQL Statement

```
SELECT Department, Manager_ID, COUNT(Employee_ID)
        FROM Employee_Manager
                GROUP BY Manager_ID, Department;
```

SQL – Group By

SELECT Department, Manager_ID, COUNT (Employee_ID) **FROM** Employee_Manager
GROUP BY Manager_ID, Department

Group By Manager_ID,Department

Employee_ID	Employee_Last_Name	Employee_Mid_Name	Employee_First_Name	Employee_Email	Department	Grade	Manager_ID
2345	Atherton	S.	Cindy	Atherton_Cindy@yahoo.com	HR	1	NULL
3556	George	A.	Henry	George_Henry@rediffmail.com	Finance	1	NULL
3620	Jackson	G.	Janet	Jackson_Janet@samsonite.co.in	Design	1	NULL
22789	Stevenson	S.	Crystal	Stevenson_Crystal@mag.com	HR	2	2345
23456	Smith	A.	Luther	Smith_Luther@yahoo.com	Finance	2	3556
30456	Langer	C.	Christiana	Langer_Christiana@rediffmail.com	HR	3	2345
31234	Frost	J.	Robert	Frost_Robert@training.com	Finance	3	3556
32345	Austen	L.	Jane	Austen_Jane@yahoo.com	Design	2	3620

Records from Employee_Manager Table

Query Results

Department	Manager_ID	Count(Employee_ID)
HR	2345	2
Finance	3556	2
Design	3620	1
HR	NULL	1
Finance	NULL	1
Design	NULL	1

Retrieval using HAVING

- Used to specify condition on group

List all customers who are having loans greater than 4000

```
Select Cust_ID,SUM(Amount_in_Dollars)
    From Customer_Loan
        Group By Cust_ID  Having SUM(Amount_in_Dollars) > 4000.00;
```

```
SELECT Cust_ID, SUM(Amount_in_Dollars) FROM Customer_Loan GROUP BY Cust_ID
HAVING SUM(Amount_in_Dollars) > 4000.00;
```

GROUP BY Cust_ID

Cust_ID	Loan_No	Amount_in_Dollars
101	1011	8755.00
103	2010	2555.00
104	2056	3050.00
103	2015	2000.00

Customer_Loan records from Customer_Loan table

Query Results

Cust_ID	Sum(Amount_in_Dollars)
101	8755.00
103	4555.00

Can you identify any error…?

```
Select Cust_ID,SUM(Amount_in_Dollars)
    From Customer_Loan
        Group By Cust_ID  Having LOAN_NO > 4000.00;
```

Ans: The Having condition has to be based on some column that appears in the select list

Infosys®
POWERED BY INTELLECT
DRIVEN BY VALUES

Relational algebra operations

Education and Research

Infosys®

POWERED BY INTELLECT
DRIVEN BY VALUES

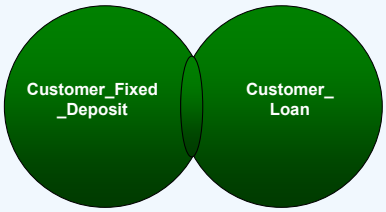
SET operations

Education and Research

Retrieval using UNION

List all the customer who has either Fixed Deposit or Loan or Both

```
SELECT Cust_ID
   FROM Customer_Fixed_Deposit
        UNION
SELECT Cust_ID
   FROM Customer_Loan;
```



The UNION operation

- Combines the rows from two sets of query results.
- By default, the UNION operation **eliminates duplicate rows** as part of its processing.

The results of two independent SELECT statements can be worked with using the SET operation – UNION. By default, UNION returns only distinct values. Union is like an “OR” operation. If the tuple occurs in relation 1 or relation 2, it is selected. Set theoretic notation indicates union as indicated in the slide

Union (Contd…)

Cust_ID	Cust_Last_Name	Cust_Mid_Name	Cust_First_Name	Cust_Email	Fixed_Deposit_No	Amount_in_Dollars	Rate_of_Interest_in_Percent
101	Smith	A.	Mike	Smith_Mike@yahoo.com	2011	8055.00	6.5
103	Langer	G.	Justin	Langer_Justin@yahoo.com	2015	2060.00	6.5
104	Quails	D.	Jack	Quails_Jack@yahoo.com	3010	3050.00	6.5

Customer_Fixed_Deposit records from Customer_Fixed_Deposit table

Cust_ID	Loan_No	Amount_in_Dollars
101	1011	8755.00
103	2010	2555.00
104	2056	3050.00
103	2015	2000.00

Customer_Loan records from Customer_Loan table

Cust_ID
101
103
104
103

Cust_ID
101
103
104

UNION

Query Results

101
103
104

Union All

```
SELECT Cust_ID  FROM Customer_Fixed_Deposit
    UNION ALL
SELECT Cust_ID  FROM Customer_Loan;
```

Cust_ID	Cust_Last_Name	Cust_Mid_Name	Cust_First_Name	Cust_Email	Fixed_Deposit_No	Amount_in_Dollars	Rate_of_Interest_in_Percent
101	Smith	A.	Mike	Smith_Mike@yahoo.com	2011	8055.00	6.5
103	Langer	G.	Justin	Langer_Justin@yahoo.com	2015	2060.00	6.5
104	Quails	D.	Jack	Quails_Jack@yahoo.com	3010	3050.00	6.5

Customer_Fixed_Deposit records from Customer_Fixed_Deposit table

Cust_ID	Loan_No	Amount_in_Dollars
101	1011	8755.00
103	2010	2555.00
104	2056	3050.00
103	2015	2000.00

Customer_Loan records from Customer_Loan table

Cust_ID
101
103
104
103

Cust_ID
101
103
104

UNION ALL

Query Results

Cust_ID
101
103
104
103
101
103
104

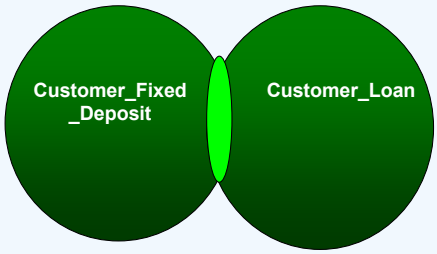
Union - Restrictions

- The SELECT statements must contain the **same number of columns**
- Data type
  - Each column in the first table must be the same as the **data type** of the corresponding column in the second table.
  - Data width and column name can differ
- Neither of the two tables can be sorted with the **ORDER BY** clause.
  - **Combined query results can be sorted**

Retrieval using INTERSECT

List all the customer who have both Fixed Deposit and Loan.

```
SELECT Cust_ID
  FROM Customer_Fixed_Deposit
       INTERSECT
SELECT Cust_ID
  FROM Customer_Loan;
```

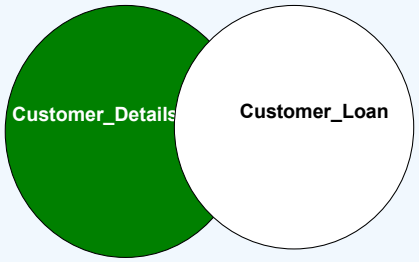


An intersection is an AND operation. It retrieves those tuples which are present in both relation

Minus

- Get All the Customer who have not taken loan

```
Select Cust_ID
        from Customer_details
              MINUS
Select Cust_Id
        from Customer_loan;
```



This is the difference operation. It retrieves tuples which are present in relation 1 but not in relation 2.

Other RA operations

- Restriction
- Projection
- Join

Restriction

- Restricts the rows that can be chosen from a relation using a **WHERE** clause
- Takes a **horizontal subset of values** from the original relation
  - Example: select * from employee where salary > 10000;

This will retrieve only those rows of the table which satisfy the condition in the where clause

Projection

- Projection is projecting a set of attributes of a relation so that rows of values corresponding to those columns will figure in the output
- This takes a **vertical subset** of the relation
- Example: select empid, name, salary from employee;

Infosys®

POWERED BY INTELLECT
DRIVEN BY VALUES

Joins

Education and Research

JOINS

- Cartesian Product
- Inner join
- Equi join
- Outer join
  - Left-outer join
  - Right-outer join
- Self join

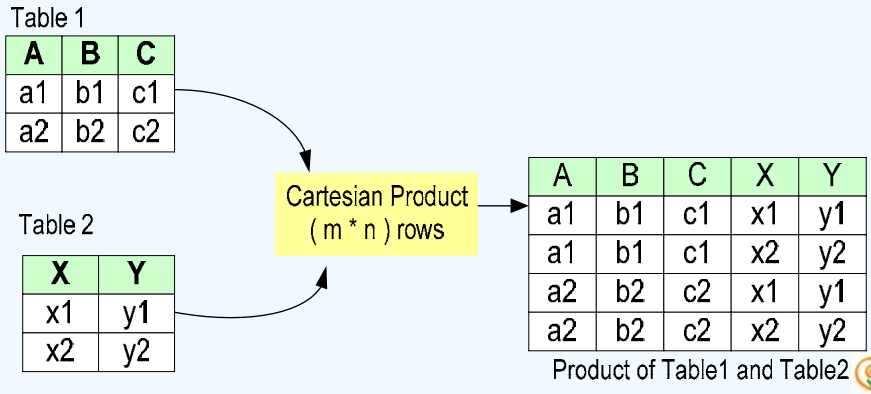
In relational databases, data is spread over multiple tables. Sometimes we may want data from two or more tables. A join is an operation which combines results from two or more tables.

Cartesian Product Or Cross Join

- Returns **All** rows from first table, Each row from the first table is combined with all rows from the second table

  Example

  Select * from Table1,Table2;



Inner Joins

- Common type of join
- An inner join between two (or more) tables is the Cartesian product that **satisfies the join condition in the WHERE clause**

Retrieval from Multiple tables-Equi join

Get all combinations of emp and cust information such that the emp and cust are co-located.

```
SELECT Table1.Emp_ID, Table1.City, Table2.Cust_ID, Table2.City
        FROM Table1, Table2
                WHERE Table1.City = Table2.City;
```

Table1

Emp_ID	CITY
A1	New YorK
A2	NULL
A3	Chicago
A4	Chicago
A5	Paris

Table2

Cust_ID	CITY
B1	New York
B2	New York
B3	NULL
B4	Chicago
B5	Moscow

INNER JOIN

Output Table

Table1.Emp_ID	Table1.City	Table2.Cust_ID	Table2.City
A1	New York	B1	New York
A1	New York	B2	New York
A3	Chicago	B4	Chicago
A4	Chicago	B4	Chicago

Here the where clause is based on the equality condition “=“. Hence it is called equi join

Retrieval from Multiple tables- Equi join

Display the First and Last Name of Customer who have taken Loan

```
Select a.Cust_Id,b.Cust_First_Name,b.Cust_Last_Name
        from Customer_loan a, customer_details b
                where a.cust_id = b.cust_id;
```

If the where clause is based on a non quality condition (<). ?Hence, it is called non-equi join

Outer join

- Retrieve all rows that match the **WHERE** clause and also those that have a **NULL** value in the column used for join.

The inner join takes into account only those non NULL rows from the tables involved. If you want the result to include even those rows having a NULL for a particular row in the selected column, then go for an outer join. The syntax for representing this is slightly different in each RDBMS product. What follows in the next slide is the oracle style.

Left/Right-Outer join

- Left outer joins include all records from the first (left) of two tables,
  A = B **(+)**

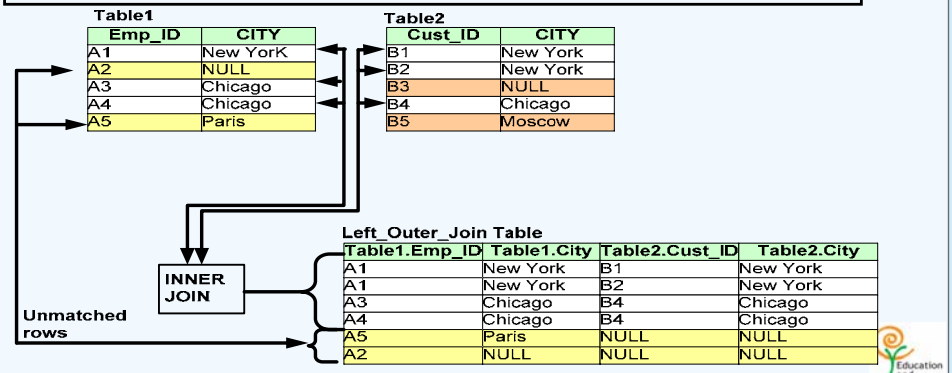
- Right outer joins include all records from the second (right) of two tables,

  A **(+)** = B

Example of left-join

List all cities of Table1 if there is match in cities in Table2 & also unmatched Cities from Table1

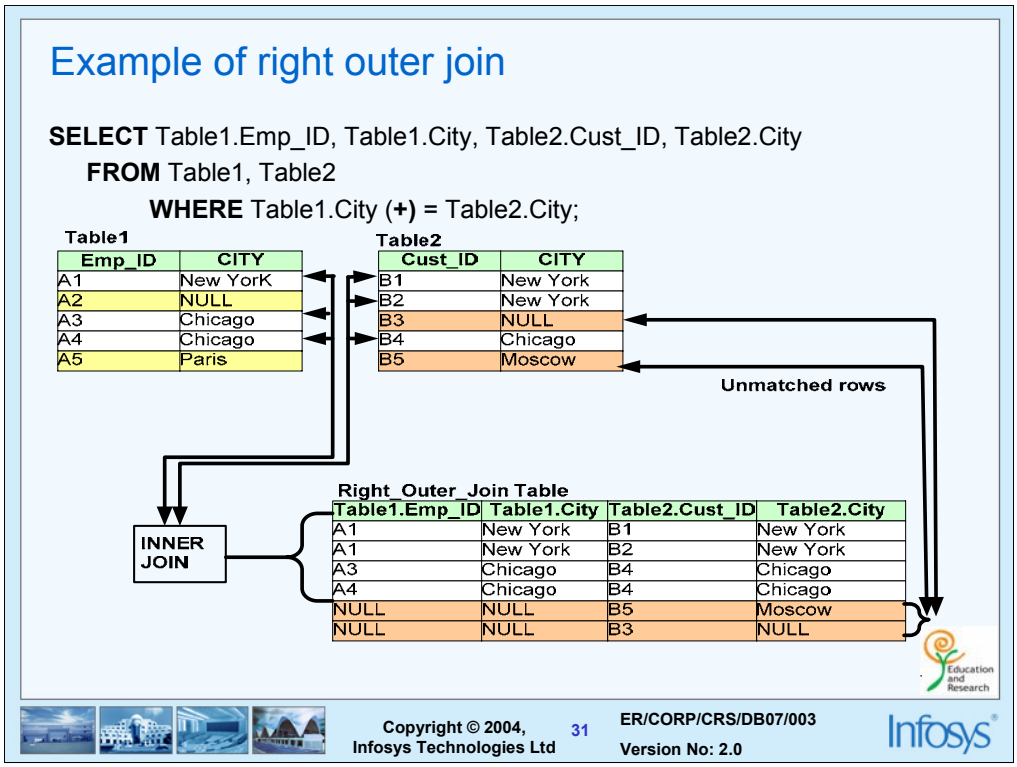
```
SELECT Table1.Emp_ID, Table1.City, Table2.Cust_ID, Table2.City
        FROM Table1, Table2
                WHERE Table1.City  = Table2.City (+);
```



Example of Left Outer Join

- List **all** customer details and loan details if they have availed loans.

```
Select Customer_details.Cust_id,Cust_Last_name,Loan_no,Amount_in_dollars
    from Customer_details,Customer_loan
        where Customer_details.Cust_id = Customer_loan.Cust_id (+);
```



The (+) symbol is next to the column which needs to be expanded to include null values also. In the example above, there may be some customers who have not made any orders, so if we select their names from the customers table (the second table based on int position in the query), the corresponding order detail would be null. Even then such values have to be selected . That's what is indicated. A typical output would look like:

ORDER_NUM CUST_NAME

--------- ----------

5 radha

first corp

jcp inc.

Self join-Joining a table with itself

To list all the Employees along with their Managers

```
Select
        Emp.Employee_ID                 as      "Employee ID",
        Emp.Employee_Last_Name          as      "Employee Last Name",
        Emp.Employee_first_Name         as      "Employee First Name",
        Emp.Manager_Id                  as      "Manager ID",
        Manager.Employee_Last_Name      as      "Manager Last Name",
        Manager.Employee_first_Name     as      "Manager first Name"

From    employee_Manager  Emp ,  employee_Manager  Manager

Where Emp.Manager_ID  =  Manager.Employee_ID;
```

When you wish to join a table with itself based on some criteria, use the concept of synonyms. Treat the table as two different tables by giving synonyms

Self Join (Contd…)

Employee_ID	Employee_Last_Name	Employee_Mid_Name	Employee_First_Name	Employee_Email	Department	Grade	Manager_ID
2345	Atherton	S.	Cindy	Atherton_Cindy@yahoo.com	HR	1	NULL
3556	George	A.	Henry	George_Henry@rediffmail.com	Finance	1	NULL
3620	Jackson	G.	Janet	Jackson_Janet@samsonite.co.in	Design	1	NULL
22789	Stevenson	S.	Crystal	Stevenson_Crystal@mag.com	HR	2	2345
23456	Smith	A.	Luther	Smith_Luther@yahoo.com	Finance	2	3556
30456	Langer	C.	Christiana	Langer_Christiana@rediffmail.com	HR	3	2345
31234	Frost	J.	Robert	Frost_Robert@training.com	Finance	3	3556
32345	Austen	L.	Jane	Austen_Jane@yahoo.com	Design	2	3620

```
SELECT Emp.Employee_ID as "Employee ID", Emp.Employee_Last_Name as "Employee Last Name",
Emp.Employee_First_Name as "Employee First Name",Emp.Manager_ID as "Manager ID",
Manager.Employee_Last_Name as "Manager Last Name" , Manager.Employee_First_Name as "Manager First Name"

FROM Employee_Manager  Emp, Employee_Manager  Manager

WHERE Emp.Manager_ID = Manager.Employee_ID ;
```

Self Join

Query Results

Employee ID	Employee Last Name	Employee First Name	Manager ID	Manager Last Name	Manager First Name
22789	Stevenson	Crystal	2345	Atherton	Cindy
23456	Smith	Luther	3556	George	Henry
30456	Langer	Christiana	2345	Atherton	Cindy
31234	Frost	Robert	3556	George	Henry
32345	Austen	Jane	3620	Jackson	Janet

Infosys®
POWERED BY INTELLECT
DRIVEN BY VALUES

Independent subqueries

Education and Research

Independent sub-queries

- Inner query is independent of outer query.
- Inner query is executed first and the results are stored.
- Outer query then runs on the stored results.

These are queries where there are two parts to the query. We need to collect one type of information based on which other set of information has to be retrieved from the table.

For e.g :

Select all sales reps who have a higher quota than sales rep 101.

We need to analyze this query and understand how to break it into sub problems

1. *First we need to find out what is the quota of sales rep 101*
2. *Based on this info, we need to select sales reps who have a higher quota than this value*

3. *So, the inner query will find the quota of sales rep 101 and the outer query will extract sales reps exceeding this quota value. The solution would look like:*

```
SELECT Rep
FROM SalesReps
WHERE Quota >
    SELECT Quota
    FROM SalesReps
    WHERE Empl_Num = 101;
```

Retrieval using SUB QUERIES

To list the Cust_ID and Loan_No for all Customers who have taken a loan of amount greater than the loan amount of Customer (Cust_ID = 104).

```
Select cust_ID, Loan_no
From Customer_Loan
Where amount_in_dollars >
                    (Select amount_in_dollars
                          From Customer_Loan
                               Where Cust_ID = 104);
```

Sub Query (Contd…)

```
SELECT Cust_ID, Loan_No
    FROM Customer_Loan
        WHERE Amount_in_Dollars >
            (SELECT Amount_in_Dollars
                FROM Customer_Loan
                    WHERE Cust_ID = 104);
```

Sub-Query

```
SELECT Amount_in_Dollars
    FROM Customer_Loan
        WHERE Cust_ID = 104;
```

data

Cust_ID	Loan_No	Amount_in_Dollars
101	1011	8755.00
103	2010	2555.00
104	2056	3050.00
103	2015	2000.00

Customer_Loan records from Customer_Loan table

3050.00

Cust_ID	Loan_No	Amount_in_Dollars
101	1011	8755.00
103	2010	2555.00
104	2056	3050.00
103	2015	2000.00

Customer_Loan records from Customer_Loan table

Query Result

101	**1011**

Retrieval using SUB QUERIES

List customer names of all customers who have taken a loan > $3000.00.

```
SELECT Cust_Last_Name, Cust_Mid_Name, Cust_First_Name
    FROM Customer_Details
        WHERE Cust_ID
                IN
                ( SELECT Cust_ID
                        FROM Customer_Loan
                                WHERE Amount_in_Dollars > 3000.00);
```

Retrieval using SUB QUERIES

List customer names of all customers who have the same Account_type as Customer 'Jones Simon' .

```
SELECT Cust_Last_Name, Cust_Mid_Name, Cust_First_Name
    FROM Customer_Details
            WHERE Account_Type
                        =
                ( SELECT Account_Type
                        FROM Customer_Details
                                WHERE Cust_Last_Name = 'Jones'
                                AND Cust_First_Name = 'Simon');
```

Retrieval using SUB QUERIES

List customer names of all customers who do not have a Fixed Deposit.

```
SELECT Cust_Last_Name, Cust_Mid_Name, Cust_First_Name
        FROM Customer_Details
                WHERE Cust_ID
                        NOT IN
                                ( SELECT Cust_ID
                                        FROM Customer_Fixed_Deposit);
```

Retrieval using SUB QUERIES

***List customer names of all customers who have either a Fixed Deposit or a loan but not both at any of Bank Branches*. The list includes customers who have no fixed deposit and loan at any of the bank branches.**

```
SELECT Cust_Last_Name, Cust_Mid_Name, Cust_First_Name
FROM Customer_Details
WHERE Cust_ID
        NOT IN
         ( SELECT Cust_ID
          FROM Customer_Loan
          WHERE Cust_ID
                  IN
                      (SELECT Cust_ID
                       FROM Customer_Fixed_Deposit ));
```

Summary

- The result of a query can be grouped based on a grouping column
- While checking for conditions after grouping by a column , Having is used instead of where
- Grouped queries help look at data category wise
- When the query consists of more than one component, it is implemented in the form of a nested query depending on the nature of the query
- Sub queries help split a problem involving different levels of data
- Relational algebra operations like union, intersect, difference, restriction, projection and join help us get different combinations of data from more than one table

Thank You!