

Assignment No. 7

Aim: Implement nested sub queries. Perform a test for set membership (in, not in), set comparison (<some, >=some, <all etc.) and set cardinality (unique, not unique).

Objective:

- To learn different types of Joins.
- To implement different sub queries.

Theory :

MySQL JOINS are used with SELECT statement. It is used to retrieve data from multiple tables. It is performed whenever you need to fetch records from two or more tables.

There are three types of MySQL joins:

- MySQL INNER JOIN (or sometimes called simple join)
- MySQL LEFT OUTER JOIN (or sometimes called LEFT JOIN)
- MySQL RIGHT OUTER JOIN (or sometimes called RIGHT JOIN)

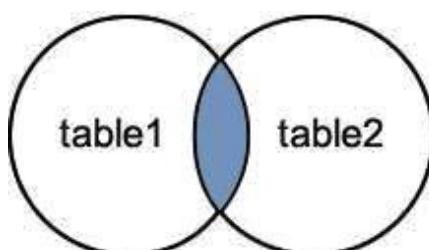
MySQL Inner JOIN (Simple Join)

The MySQL INNER JOIN is used to return all rows from multiple tables where the join condition is satisfied. It is the most common type of join.

Syntax:

```
SELECT columns  
FROM table1  
  
INNER JOIN table2  
ON table1.column = table2.column;
```

Image representation:



Let's take an example:

Consider two tables "officers" and "students", having the following data.

Execute the following query:

```
SELECT officers.officer_name, officers.address, students.course_name  
FROM officers  
INNER JOIN students  
ON officers.officer_id = students.student_id;
```

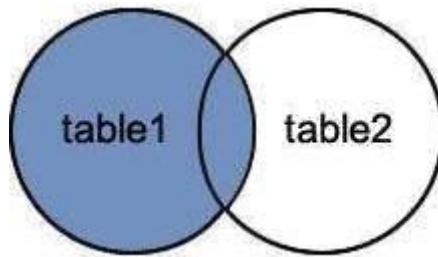
MySQL Left Outer Join

The LEFT OUTER JOIN returns all rows from the left hand table specified in the ON condition and only those rows from the other table where the join condition is fulfilled.

Syntax:

```
SELECT columns  
FROM table1  
LEFT [OUTER] JOIN table2  
ON table1.column = table2.column;
```

Image representation:



Let's take an example:

Consider two tables "officers" and "students", having the following data.

Execute the following query:

```
SELECT officers.officer_name, officers.address, students.course_name  
FROM officers  
LEFT JOIN students  
ON officers.officer_id = students.student_id;
```

MySQL Right Outer Join

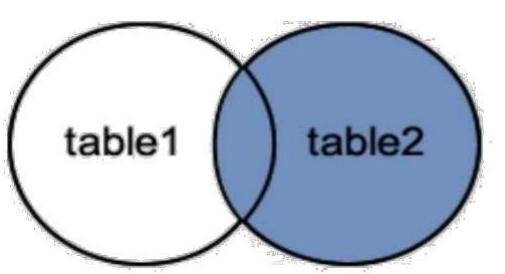
The MySQL Right Outer Join returns all rows from the RIGHT-hand table specified in the ON condition and only those rows from the other table where he join condition is fulfilled.

Syntax:

```
SELECT columns
```

```
FROM table1  
RIGHT [OUTER] JOIN table2  
ON table1.column = table2.column;
```

Image representation:



Let's take an example:

Consider two tables "officers" and "students", having the following data.

Execute the following query:

```
SELECT      officers.officer_name, officers.address,  
            students.course_name,  
            students.student_name  
        FROM officers  
    RIGHT JOIN students  
        ON officers.officer_id = students.student_id;
```

SPECIAL OPERATOR:

MySQL IN Condition

The MySQL IN condition is used to reduce the use of multiple OR conditions in a SELECT, INSERT, UPDATE and DELETE statement.

Syntax:

```
expression IN (value1, value2,.....value_n);
```

Parameters:

expression: It specifies a value to test.

value1, value2, or value_n: These are the values to test against expression. If any of these values matches expression, then the IN condition will evaluate to true. This is a quick method to test if any one of the values matches expression.

Execute the following query:

```
SELECT *  
  
FROM officers  
WHERE officer_name IN ('Ajeet', 'Vimal', 'Deepika');
```

MySQL NOT Condition

The MySQL NOT condition is opposite of MySQL IN condition. It is used to negate a condition in a SELECT, INSERT, UPDATE or DELETE statement.

Syntax:

NOT condition

Parameter:

condition: It specifies the conditions that you want to negate.

MySQL NOT Operator with IN condition

Consider a table "officers", having the following data.

Execute the following query:

```
SELECT *  
  
FROM officers  
WHERE officer_name NOT IN ('Ajeet','Vimal','Deepika');
```

MySQL IS NULL Condition

MySQL IS NULL condition is used to check if there is a NULL value in the expression. It is used with SELECT, INSERT, UPDATE and DELETE statement.

Syntax:

expression IS NULL

Parameter:

expression: It specifies a value to test if it is NULL

Execute the following query:

```
SELECT *  
FROM officers  
WHERE officer_name IS NULL;
```

MySQL IS NOT NULL Condition

MySQL IS NOT NULL condition is used to check the NOT NULL value in the expression. It is used with SELECT, INSERT, UPDATE and DELETE statements.

Syntax:

expression IS NOT NULL

Parameter:

expression: It specifies a value to test if it is not NULL value.

Execute the following query:

```
SELECT *
FROM officers
WHERE officer_name IS NOT NULL;
```

SET OPERATORS:

The Set operator combines the result of 2 queries into a single result. The following are the operators:

- Union
- Union all
- Intersect
- Minus

LAB PRACTICE ASSIGNMENT:

Consider the following table structure for this assignment:

- Location(Location_Id integer, Reginal_Group varchar(20))
- Department (Department_Id, Name, Location_Id)
- Job(Job_Id Integer, Function Varchar(30))
- Employee(Employee_Id, Lastname ,Firstname,Middlename, Job_Id, Manager_Id, Hiredate, Salary, Department_Id)
- Loan(Employee_Id, Firstname , Loan_Amount)

LOCATION TABLE

LOCATION_ID	REGINAL_GROUP
122	New York
123	Dallas
124	Chicago
167	Boostan

DEPARTMENT TABLE

DEPARTMENT_ID	NAME	LOCATION_ID
10	Accounting	122

20	Research	124
30	Sale	123
40	Operation	164

JOB TABLE

JOB_ID	FUNCTION
667	Cleark
668	Staff
669	Analyst
670	Saleperson
671	Manager
672	President

EMPLOYEE TABLE

EMPL_OYEE_ID	LAST_NAM_E	FIRS_TNA_ME	MIDD_LEN_A_ME	JO_B_I_D	MANA_GER_I_D	HIR_EDA_TE	SAL_AR_Y	DEPARTMENT_ID
7369	Smith	Jon	Q	667	7902	17-DEC-84	800	10
7499	Allen	Kevin	J	670	7698	20-FEB-85	1600	20
7505	Doyle	Jean	K	671	7839	04-APR-85	2850	20
7506	Dennis	Lynn	S	671	7839	15-MAY-85	2750	30
7507	Baker	Leslie	D	671	7839	10-JUN-85	2200	40
7521	wark	cynthia	D	670	7698	22-FEB-85	1250	10

Perform the following queries on the above table:

- 1) Perform all types of JOIN operations on Employee and Loan tables.

- 2) Perform all types of set operations on Employee and Loan tables.
- 3) Find out no.of employees working in “Sales” department
- 4) Find out the employees who are not working in department 10 or 30.
- 5) List out employee id, last name in descending order based on the salary column.
- 6) How many employees who are working in different departments wise in the organization
- 7) List out the department id having at least four employees
- 8) Display the employee who got the maximum salary.
- 9) Update the employees’ salaries, who are working as Clerk on the basis of 10%.
- 10) Delete the employees who are working in accounting department.
- 11) Find out whose department has not employees.
- 12) List out the department wise maximum salary, minimum salary, average salary of the employees
- 13) How many employees who are joined in 1985.
- 14)** Display the employees who are working in “New York”
- 15) List our employees with their department names

Conclusion:

We have implemented join, set operations, set cardinalities and nested sub queries.

Output:

1. CREATE TABLE:

a) LOCATION:

```
CREATE TABLE "Location" ( "Location_ID"  
    INTEGER NOT NULL, "Reginal_Group"  
    VARCHAR(20), PRIMARY  
    KEY("Location_ID")  
);
```

DB Browser for SQLite - D:\Akash Files\A Sem\Sem 5\Practical Assignments\DBMS\Assignment 7\Assignment7.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragmas Execute SQL

SQL 1

```
1 SELECT * FROM Location;
```

Location_ID	Reginal_Group
1 122	New York
2 123	Dallas
3 124	Chicago
4 164	Booston

Result: 4 rows returned in 113ms
At line 1:
SELECT * FROM Location;

Edit Database Cell

Type of data currently in cell: NULL
0 byte(s)

SQL Log

Show SQL submitted by Application

```
292 SELECT COUNT() FROM (SELECT "_rowid_","^" FROM ^  
293 SELECT "_rowid_","^" FROM "main"."Employee" ORDER  
294 INSERT INTO "main"."Employee"("Employee_ID","La  
295 INSERT INTO "main"."Employee"  
296 ("Employee_ID","LastName","FirstName","Middle  
297 VALUES (7521,'wark','cynthia','D',670,7698  
298 SELECT COUNT() FROM (SELECT "_rowid_","^" FROM ^  
299 SELECT "_rowid_","^" FROM "main"."Employee" ORDER  
300 SELECT "_rowid_","^" FROM "main"."Employee" ORDER  
301 SELECT COUNT() FROM (SELECT "_rowid_","^" FROM ^  
302 SELECT COUNT() FROM (SELECT * FROM Location);  
303 SELECT * FROM Location LIMIT 0, 49999;  
304 PRAGMA database_list;
```

SQL Log Plot DB Schema Remote

b) JOB:

```
CREATE TABLE "Job" (
    "Job_ID" INTEGER NOT NULL,
    "Function" TEXT NOT NULL,
    PRIMARY KEY("Job_ID")
);
```

DB Browser for SQLite - D:\Akash Files\A Sem\Sem 5\Practical Assignments\DBMS\Assignment 7\Assignment7.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragmas Execute SQL

SQL 1

```
1 SELECT * FROM Job;
```

Job_ID	Function
1 667	
2 668	Staff
3 669	Analyst
4 670	Salesperson
5 671	Manager
6 672	President

Result: 6 rows returned in 24ms
At line 1:
SELECT * FROM Job;

Edit Database Cell

Type of data currently in cell: NULL
0 byte(s)

SQL Log

Show SQL submitted by Application

```
300 SELECT "_rowid_","^" FROM "main"."Employee" ORDER  
301 SELECT COUNT() FROM (SELECT "_rowid_","^" FROM ^  
302 SELECT COUNT() FROM (SELECT * FROM Location);  
303 SELECT * FROM Location LIMIT 0, 49999;  
304 PRAGMA database_list;  
305 SELECT type,name,sql,tbl_name FROM "main".sqlite_master;  
306 SELECT "_rowid_","^" FROM "main"."Employee" ORDER  
307 SELECT COUNT() FROM (SELECT "_rowid_","^" FROM ^  
308 SELECT "_rowid_","^" FROM "main"."Employee" ORDER  
309 SELECT COUNT() FROM (SELECT "_rowid_","^" FROM ^  
310 SELECT COUNT() FROM (SELECT * FROM Job);  
311 SELECT * FROM Job LIMIT 0, 49999;  
312 PRAGMA database_list;
```

SQL Log Plot DB Schema Remote

c) DEPARTMENT:

```
CREATE TABLE "Department" (
    "Department_ID" INTEGER NOT NULL,
    "Name" TEXT NOT NULL,
    "Location_ID" INTEGER,
    PRIMARY KEY("Department_ID"),
    FOREIGN KEY("Location_ID") REFERENCES Location);
```

The screenshot shows the DB Browser for SQLite interface. At the top, the title bar reads "DB Browser for SQLite - Database: File1.sq3 (New) / Practical Assignments/SEM4 Assignment 7/Assignment7.sq3". Below the title bar is a menu bar with File, Edit, View, Tools, Help. Under the Tools menu, there are options like New Database, Open Database, Undo Changes, Redo Changes, Open Project, Save Project, Attach Database, and Close Database. The main window has tabs for Database Structure, Browse Data, Edit Progress, and Results SQL. In the Results SQL tab, the following SQL command is entered:

```
2. SELECT * FROM Department;
```

Below the SQL input field, the results are displayed in a table:

Department_ID	Name	Location_ID
1. 10	Accounting	122
2. 20	Research	124
3. 30	Sales	123
4. 40	Operational	164

At the bottom of the results pane, the message "Result: 4 rows affected in 0ms" and "SELECT * FROM Department;" is shown.

To the right of the results pane is the "Edit Database Cell" panel, which is currently empty. Below it is the "SQL Log" panel, which contains the following log entries:

```
104 TRIGER department_1491
105 SELECT copy_main.sql,101_main FROM "main".sql1
106 SELECT "rowid","* FROM "main"."Department"
107 SELECT COUNT(*) FROM (SELECT "rowid","*" FROM
108 "main"."Department")
109 SELECT "rowid","* FROM "main"."Department"
110 SELECT COUNT(*) FROM (SELECT "rowid","*" FROM
111 "main"."Department")
112 SELECT COUNT(*) FROM (SELECT "rowid","*" FROM
113 "main"."Department")
114 SELECT COUNT(*) FROM (SELECT "rowid","*" FROM
115 "main"."Department")
116 SELECT COUNT(*) FROM (SELECT "rowid","*" FROM
117 "main"."Department")
118 TRIGER department_1491
119 TRIGER department_1491
120 SELECT copy_main.sql,101_main FROM "main".sql1
121 SELECT COUNT(*) FROM (SELECT "rowid","*" FROM
122 "main"."Department")
123 SELECT COUNT(*) FROM (SELECT "rowid","*" FROM
124 "main"."Department")
125 SELECT COUNT(*) FROM (SELECT "rowid","*" FROM
126 "main"."Department")
127 TRIGER department_1491
128 TRIGER department_1491
```

d) EMPLOYEE:

```
CREATE TABLE "Employee" (
    "Employee_ID" INTEGER NOT NULL,
    "LastName" TEXT,
    "FirstName" TEXT NOT NULL,
    "MiddleName" TEXT,
    "Job_ID" INTEGER NOT NULL,
    "Manager_ID" INTEGER NOT NULL,
    "HireDate" TEXT NOT NULL, "Salary"
    INTEGER NOT NULL, "Department_ID"
    INTEGER NOT NULL, PRIMARY
    KEY("Employee_ID", "Manager_ID"),
    FOREIGN KEY("Department_ID") REFERENCES "Department"("Department_ID"),
    FOREIGN KEY("Job_ID") REFERENCES "Job"("Job_ID")
);
```

DB Browser for SQLite - D:\Akash Files\A Sem\Sem 5\Practical Assignments\DBMS\Assignment 7\Assignment7.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragmas Execute SQL

SQL 1

```
1 SELECT * FROM Employee;
```

Employee_ID	Lastname	FirstName	MiddleName	Job_ID	Manager_ID	HireDate	Salary	Department_ID
1 7369	Smith	Jon	Q	667	7902	17-DEC-84	800	10
2 7499	Alan	Kevin	J	670	7698	20-FEB-85	1600	20
3 7505	Doyle	Jean	K	671	7839	04-APR-85	2850	20
4 7506	Dennis	Lynn	S	671	7839	15-MAY-85	2750	30
5 7507	Baker	Leslie	D	671	7839	10-JUN-85	2200	40
6 7521	wark	cynthia	D	670	7698	22-FEB-85	1250	10

Result: 6 rows returned in 21ms
At line 1:
SELECT * FROM Employee;

Edit Database Cell

Type of data currently in cell: NULL
0 byte(s)

SQL Log

Show SQL submitted by Application

```
310 SELECT COUNT(*) FROM (SELECT * FROM Job);
311 SELECT * FROM Job LIMIT 0, 49999;
312 PRAGMA database_list;
313 SELECT type,name,sql,tbl_name FROM "main".sqlite_master;
314 SELECT COUNT(*) FROM (SELECT * FROM Department);
315 SELECT * FROM Department LIMIT 0, 49999;
316 PRAGMA database_list;
317 SELECT type,name,sql,tbl_name FROM "main".sqlite_master;
318 SELECT COUNT(*) FROM (SELECT "_rowid_" AS id, * FROM "main"."Employee");
319 SELECT "_rowid_" AS id, * FROM "main"."Employee" ORDER BY id;
320 SELECT COUNT(*) FROM (SELECT * FROM Employee);
321 SELECT * FROM Employee LIMIT 0, 49999;
322 PRAGMA database_list;
```

SQL Log Plot DB Schema Remote

e) LOAN:

CREATE TABLE "Loan" (
 "Employee_ID" INTEGER NOT NULL,
 "FirstName" TEXT NOT NULL,
 "Loan_Amount" INTEGER NOTNULL);

DB Browser for SQLite - D:\Akash Files\A Sem\Sem 5\Practical Assignments\DBMS\Assignment 7\Assignment7.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragmas Execute SQL

SQL 1

```
1 SELECT * FROM Loan;
```

Employee_ID	FirstName	Loan_Amount
1 7369	Jon	18000
2 7505	Jean	13000
3 7521	cynthia	14500

Result: 3 rows returned in 6ms
At line 1:
SELECT * FROM Loan;

Edit Database Cell

Type of data currently in cell: NULL
0 byte(s)

SQL Log

Show SQL submitted by Application

```
124 UPDATE "main"."Loan" SET "Loan_Amount"=? WHERE "Employee_ID"=?;
125 INSERT INTO "main"."Loan" ("Employee_ID", "FirstName", "Loan_Amount") VALUES (?, ?, ?);
126 UPDATE "main"."Loan" SET "Employee_ID"=? WHERE "Employee_ID"=?
127 UPDATE "main"."Loan" SET "FirstName"=? WHERE "Employee_ID"=?
128 UPDATE "main"."Loan" SET "Loan_Amount"=? WHERE "Employee_ID"=?
129 INSERT INTO "main"."Loan" ("Employee_ID", "FirstName", "Loan_Amount") VALUES (?, ?, ?);
130 UPDATE "main"."Loan" SET "Employee_ID"=? WHERE "Employee_ID"=?
131 UPDATE "main"."Loan" SET "FirstName"=? WHERE "Employee_ID"=?
132 UPDATE "main"."Loan" SET "Loan_Amount"=? WHERE "Employee_ID"=?
133 SELECT COUNT(*) FROM (SELECT * FROM Loan);
134 SELECT * FROM Loan LIMIT 0, 49999;
```

SQL Log Plot DB Schema Remote

2. JOIN:

```
SELECT Employee.FirstName, Loan.FirstName  
FROM Employee  
LEFT JOIN Loan ON Employee.Employee_ID = Loan.Employee_ID;
```

The screenshot shows the DB Browser for SQLite interface. The top menu bar includes File, Edit, View, Tools, Help, New Database, Open Database, Write Changes, Revert Changes, Open Project, Save Project, Attach Database, and Close Database. Below the menu is a toolbar with icons for Database Structure, Browse Data, Edit Pragmas, and Execute SQL. The main window has tabs for Database Structure, Browse Data, Edit Pragmas, and Execute SQL. The Execute SQL tab contains the following SQL code:

```
1 SELECT Employee.FirstName, Loan.FirstName  
2 FROM Employee  
3 LEFT JOIN Loan ON Employee.Employee_ID = Loan.Employee_ID;
```

Below the code is a table with two columns: FirstName and FirstName. The data is as follows:

	FirstName	FirstName
1	Jon	Jon
2	Kevin	NULL
3	Jean	Jean
4	Lynn	NULL
5	Leslie	NULL
6	cynthia	cynthia

Below the table, the status bar displays "Result: 6 rows returned in 13ms". The bottom right corner of the window shows "UTF-8".

3. SET OPERATIONS:

a) UNION:

```
SELECT * FROM Department  
UNION  
SELECT * FROM Loan;
```

DB Browser for SQLite - D:\Akash Files\A Sem\Sem 5\Practical Assignments\DBMS\Assignment 7\Assignment7.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragmas Execute SQL

SQL 1

```

1 SELECT * FROM Department
2 UNION
3 SELECT * FROM Loan;

```

Department_ID	Name	Location_ID
1 10	Accounting	122
2 20	Research	124
3 30	Sale	123
4 40	Operation	164
5 7369	Jon	18000
6 7505	Jean	13000
7 7521	cynthia	14500

Result: 7 rows returned in 14ms
At line 1:
SELECT * FROM Department
UNION
SELECT * FROM Loan;

Edit Database Cell

Type of data currently in cell: NULL
0 byte(s)

SQL Log

Show SQL submitted by Application

```

15 SELECT COUNT(*) FROM (SELECT "_rowid_","* FROM "main"."Department" ORDER BY "_rowid_")
16 SELECT COUNT(*) FROM (SELECT "_rowid_","* FROM "main"."Job" ORDER BY "_rowid_")
17 SELECT COUNT(*) FROM (SELECT "_rowid_","* FROM "main"."Loan" ORDER BY "_rowid_")
18 SELECT COUNT(*) FROM (SELECT "_rowid_","* FROM "main"."Location" ORDER BY "_rowid_")
19 SELECT COUNT(*) FROM (SELECT "_rowid_","* FROM "main"."Region" ORDER BY "_rowid_")
20 SELECT COUNT(*) FROM (SELECT "_rowid_","* FROM "main"."Employee" ORDER BY "_rowid_")
21 SELECT COUNT(*) FROM (SELECT "_rowid_","* FROM "main"."Customer" ORDER BY "_rowid_")
22 UNION
23 -SELECT * FROM Loan;
24 SELECT * FROM Department
25 UNION
26 SELECT * FROM Location LIMIT 0, 49999;
27 PRAGMA database_list;

```

SQL Log Plot DB Schema Remote

b) UNION ALL:

```

SELECT * FROM Job
UNION ALL
SELECT * FROM Location;

```

DB Browser for SQLite - D:\Akash Files\A Sem\Sem 5\Practical Assignments\DBMS\Assignment 7\Assignment7.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragmas Execute SQL

SQL 1

```

1 SELECT * FROM Job
2 UNION ALL
3 SELECT * FROM Location;

```

Job_ID	Function
1 667	Clerk
2 668	Staff
3 669	Analyst
4 670	Saleperson
5 671	Manager
6 672	President
7 122	New York
8 123	Dallas
9 124	Chicago
10 164	Boston

Result: 10 rows returned in 16ms
At line 1:
SELECT * FROM Job
UNION ALL
SELECT * FROM Location;

Edit Database Cell

Type of data currently in cell: NULL
0 byte(s)

SQL Log

Show SQL submitted by Application

```

42 SELECT COUNT(*) FROM (SELECT "_rowid_","* FROM "main"."Job" ORDER BY "_rowid_")
43 SELECT COUNT(*) FROM (SELECT "_rowid_","* FROM "main"."Location" ORDER BY "_rowid_")
44 SELECT COUNT(*) FROM (SELECT "_rowid_","* FROM "main"."Region" ORDER BY "_rowid_")
45 SELECT COUNT(*) FROM (SELECT "_rowid_","* FROM "main"."Employee" ORDER BY "_rowid_")
46 SELECT COUNT(*) FROM (SELECT "_rowid_","* FROM "main"."Customer" ORDER BY "_rowid_")
47 UNION ALL
48 -SELECT COUNT(*) FROM (SELECT * FROM Job
49 UNION ALL
50 -SELECT * FROM Location);
51 SELECT * FROM Job
52 UNION ALL
53 -SELECT * FROM Location LIMIT 0, 49999;
54

```

SQL Log Plot DB Schema Remote

c) INTERSECT:

```
SELECT Employee.FirstName FROM Employee  
INTERSECT  
SELECT Loan.FirstName FROM Loan;
```

The screenshot shows the DB Browser for SQLite interface. The SQL tab contains the following query:

```
1 SELECT Employee.FirstName FROM Employee  
2 INTERSECT  
3 SELECT Loan.FirstName FROM Loan;
```

The results pane shows a table with a single column 'FirstName' containing three rows: Jean, Jon, and cynthia. The SQL Log pane shows the full query and its execution plan.

4. No. of employees working in sales department:

```
SELECT count(Department_ID) as No_Of_Employees FROM Employee  
WHERE Employee.Department_ID = (SELECT Department_ID FROM Department  
WHERE Name='Sale');
```

The screenshot shows the DB Browser for SQLite interface. The SQL tab contains the following query:

```
1 SELECT count(Department_ID) as No_Of_Employees FROM Employee  
2 WHERE Employee.Department_ID = (SELECT Department_ID FROM Department  
3 WHERE Name='Sale');
```

The results pane shows a table with a single column 'No_Of_Employees' containing the value 1. The SQL Log pane shows the full query and its execution plan.

5. Find out the employees who are not working in department 10 or 30.

```
SELECT FirstName FROM Employee  
WHERE Department_ID NOT IN (10, 30);
```

The screenshot shows the SQL Server Management Studio interface. The top menu bar includes File, Edit, View, Tools, Help, New Database, Open Database, Write Changes, Revert Changes, Open Project, Save Project, Attach Database, and Close Database. Below the menu is a toolbar with icons for New Database, Open Database, Write Changes, Revert Changes, Open Project, Save Project, Attach Database, and Close Database. The main window has tabs for Database Structure, Browse Data, Edit Progress, and Execute SQL. The Execute SQL tab contains the following code:

```
1 SELECT FirstName FROM Employee  
2 WHERE Department_ID NOT IN (10, 30)
```

Below the code, the results pane shows the output:

FirstName
Kenn
Jean
Ulrica

The status bar at the bottom indicates "Result: 3 rows returned in 14ms" and "At line 1".

To the right of the main window, there is an "Edit Database Cell" window and a "SQL Log" window. The "Edit Database Cell" window has tabs for Model, Test, Insert, Update, Delete, and Export. The "SQL Log" window shows the history of queries submitted by Application:

```
487 WHERE Department_ID NOT IN (10, 30)  
488 SELECT FirstName FROM Employee  
489 WHERE Department_ID NOT IN (10, 30) LIMIT 0, 4  
490 SELECT COUNT(*) FROM (SELECT FirstName FROM Employee  
491 WHERE Department_ID NOT IN (10, 30)) AS T1  
492 SELECT FirstName FROM Employee  
493 WHERE Department_ID NOT IN (10, 30) LIMIT 0, 4  
494 SELECT COUNT(*) FROM (SELECT FirstName FROM Employee  
495 WHERE Department_ID NOT IN (10, 30)) AS T1  
496 SELECT FirstName FROM Employee  
497 WHERE Department_ID NOT IN (10, 30) LIMIT 0, 4
```

6. List out employee id, last name in descending order based on the salary column.

```
SELECT Employee_ID, LastName FROM  
Employee ORDER BY Salary DESC;
```

DB Browser for SQLite - D:\Akash Files\A Sem\Sem 5\Practical Assignments\DBMS\Assignment 7\Assignment7.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragmas Execute SQL

SQL 1

```
1 SELECT Employee_ID, LastName FROM Employee
2 ORDER BY Salary DESC;
```

Employee_ID	Lastname
1 7505	Doyle
2 7506	Dennis
3 7507	Baker
4 7499	Alan
5 7521	wark
6 7369	Smith

Result: 6 rows returned in 22ms
At line 1:
SELECT Employee_ID, LastName FROM Employee
ORDER BY Salary DESC;

Edit Database Cell Mode: Text Import Export Set as NULL

Type of data currently in cell: NULL
0 byte(s) Apply

SQL Log Show SQL submitted by Application Clear

```
455 WHERE Department_ID NOT IN (10, 30) LIMIT 0, 45
456 SELECT "_rowid_","* FROM "main"."Employee" ORDER
457 SELECT COUNT(*) FROM (SELECT "_rowid_","* FROM "
458 SELECT COUNT(*) FROM (SELECT Employee_ID, LastN
459 ORDER BY Salary DESC);
460 SELECT Employee_ID, LastName FROM Employee
461 ORDER BY Salary DESC LIMIT 0, 49999;
462 SELECT "_rowid_","* FROM "main"."Employee" ORDER
463 SELECT COUNT(*) FROM (SELECT "_rowid_","* FROM "
464 SELECT "_rowid_","* FROM "main"."Employee" ORDER
465 SELECT COUNT(*) FROM (SELECT "_rowid_","* FROM "
466
```

SQL Log Plot DB Schema Remote UTF-8

7. How many employees who are working in different departments wise in the organization

```
SELECT Department_ID, count(*) AS
No_of_employees FROM Employee
GROUP by Department_ID;
```

DB Browser for SQLite - D:\Akash Files\A Sem\Sem 5\Practical Assignments\DBMS\Assignment 7\Assignment7.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragmas Execute SQL

SQL 1

```
1 SELECT Department_ID, count(*) AS,
2 No_of_employees FROM Employee
3 GROUP by Department_ID;
```

Department_ID	No_of_employees
1 10	2
2 20	2
3 30	1
4 40	1

Result: 4 rows returned in 10ms
At line 1:
SELECT Department_ID, count(*) AS No_of_employees FROM Employee GROUP by Department_ID;

Edit Database Cell Mode: Text Import Export Set as NULL

Type of data currently in cell: NULL
0 byte(s) Apply

SQL Log Show SQL submitted by Application Clear

```
609 SELECT count(Department_ID) as Accounting_Emplo
610 FROM Employee WHERE Employee.Department_ID = (
611 SELECT Department_ID FROM Department WHERE Name
612 SELECT COUNT(*) FROM (SELECT Department_ID, cou
613 SELECT Department_ID, count(*) FROM Employee G
614 SELECT COUNT(*) FROM (SELECT "_rowid_","* FROM "
615 SELECT "_rowid_","* FROM "main"."Employee" ORDER
616 SELECT "_rowid_","* FROM "main"."Employee" ORDER
617 SELECT COUNT(*) FROM (SELECT "_rowid_","* FROM "
618 SELECT COUNT(*) FROM (SELECT Department_ID, cou
619 SELECT Department_ID, count(*) AS No_of_employe
620
```

SQL Log Plot DB Schema Remote

8. List out the department id having at least four employees

```
SELECT Department_ID, count(*) AS  
No_of_employees FROM Employee  
GROUP by Department_ID HAVING No_of_employees >= 4;
```

The screenshot shows the DB Browser for SQLite interface. The SQL tab contains the following query:

```
1 SELECT Department_ID, count(*) AS  
2 No_of_employees FROM Employee  
3 GROUP by Department_ID HAVING No_of_employees >= 4;
```

The results pane shows the output:

```
Result: 0 rows returned in 6ms  
At line 1:  
SELECT Department_ID, count(*) AS  
No_of_employees FROM Employee  
GROUP by Department_ID HAVING No_of_employees >= 4;
```

The SQL Log pane shows the history of submitted SQL statements, including the current query.

9. Display the employee who got the maximum salary.

```
SELECT * FROM Employee  
WHERE Salary = (SELECT max(Salary) FROM Employee);
```

The screenshot shows the DB Browser for SQLite interface. The SQL tab contains the following query:

```
1 SELECT * FROM Employee  
2 WHERE Salary = (SELECT max(Salary) FROM Employee);
```

The results pane shows the output:

Employee_ID	Lastname	Firstname	MiddleName	Job_ID	Manager_ID	HireDate	Salary	Department_ID
1 7505	Doyle	Jean	K	671	7839	04-APR-85	2850	20

The results pane also displays the following text:

```
Result: 1 rows returned in 9ms  
At line 1:  
SELECT * FROM Employee  
WHERE Salary = (SELECT max(Salary) FROM Employee);
```

The SQL Log pane shows the history of submitted SQL statements, including the current query.

10. Update the employees' salaries, who are working as Clerk on the basis of 10%.

```
UPDATE Employee  
SET Salary = Salary + (Salary*0.1)  
WHERE Job_ID = (  
SELECT Job_ID FROM Job  
WHERE Function = 'Clerk');
```

The screenshot shows the DB Browser for SQLite interface with the following details:

- Title Bar:** DB Browser for SQLite - D:\Makki\Reverie\Java\Practical Assignments\Assignment 7\Assignment7.db
- Menu Bar:** File, Edit, View, Tools, Help
- Toolbar:** New Database, Open Database, Write Changes, Recent Changes, Open Project, Save Project, Attach Database, Close Database
- Database Structures:** Database, Browse Data, Edit Programs, Execute SQL
- SQL Editor:** Contains the following SQL code:

```
1 UPDATE Employee  
2 SET Salary = Salary + (Salary*0.1)  
3 WHERE Job_ID = (  
4 SELECT Job_ID FROM Job  
5 WHERE Function = 'Clerk');
```
- Output Window:** Shows the result of the query execution:

```
Result: query executed successfully. Took 0ms, 1 rows affected  
At line 1:  
UPDATE Employee  
SET Salary = Salary + (Salary*0.1)  
WHERE Job_ID = (  
SELECT Job_ID FROM Job  
WHERE Function = 'Clerk');
```
- Log Window:** Shows the log entries corresponding to the query execution:

```
658 SELECT COUNT(*) FROM (SELECT * FROM "Employee") T1  
659 SELECT COUNT(*) FROM (SELECT * FROM "Employee") T1  
660 SELECT * FROM "Employee" ORDER BY  
661 SELECT COUNT(*) FROM (SELECT * FROM "Employee") T1  
662 SELECT * FROM "Employee" ORDER BY  
663 SELECT COUNT(*) FROM (SELECT * FROM "Employee") T1  
664 SELECT * FROM "Employee" ORDER BY  
665 SELECT COUNT(*) FROM (SELECT * FROM "Employee") T1  
666 SELECT COUNT(*) FROM (SELECT * FROM "Employee") T1  
667 SELECT COUNT(*) FROM (SELECT * FROM "Employee") T1  
668 SELECT * FROM "Employee" ORDER BY  
669 SELECT COUNT(*) FROM (SELECT * FROM "Employee") T1  
670 SELECT COUNT(*) FROM (SELECT * FROM "Employee") T1
```
- Bottom Navigation:** SQL Log, Ref, DB Schemas, Remote

DB Browser for SQLite - D:\Akash Files\A Sem\Sem 5\Practical Assignments\DBMS\Assignment 7\Assignment7.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragmas Execute SQL

SQL 1

```
1   SELECT * FROM Employee;
```

Employee_ID	Lastname	Firstname	MiddleName	Job_ID	Manager_ID	HireDate	Salary	Department_ID
1 7369	Smith	Jon	Q	667	7902	17-DEC-84	880	10
2 7499	Alan	Kevin	J	670	7698	20-FEB-85	1600	20
3 7505	Doyle	Jean	K	671	7839	04-APR-85	2850	20
4 7506	Dennis	Lynn	S	671	7839	15-MAY-85	2750	30
5 7507	Baker	Leslie	D	671	7839	10-JUN-85	2200	40
6 7521	wark	cynthia	D	670	7698	22-FEB-85	1250	10

Result: 6 rows returned in 1620ms
At line 1:
SELECT * FROM Employee;

Edit Database Cell

Type of data currently in cell: Text / Numeric
3 char(s) Apply

SQL Log

Show SQL submitted by Application Clear

```
661  SELECT "_rowid_",' FROM "main"."Job" ORDER BY " ^  
662  SELECT COUNT(*) FROM (SELECT "_rowid_",' FROM "  
663  SELECT "_rowid_",' FROM "main"."Job" ORDER BY "  
664  SELECT "_rowid_",' FROM "main"."Job" ORDER BY "  
665  SELECT COUNT(*) FROM (SELECT "_rowid_",' FROM "  
666  SELECT COUNT(*) FROM (SELECT "_rowid_",' FROM "  
667  SELECT "_rowid_",' FROM "main"."Job" ORDER BY "  
668  SELECT "_rowid_",' FROM "main"."Employee" ORDER  
669  SELECT COUNT(*) FROM (SELECT "_rowid_",' FROM "  
670  SELECT COUNT(*) FROM (SELECT * FROM Employee);  
671  SELECT * FROM Employee LIMIT 0, 49999;  
672
```

SQL Log Plot DB Schema Remote UTF-8

11. Delete the employees who are working in accounting department.

```
DELETE FROM Employee
WHERE Department_ID = (
SELECT Department_ID FROM Department
WHERE Name = 'Accounting');
```

DB Browser for SQLite - D:\Akash Files\A Sem\Sem 5\Practical Assignments\DBMS\Assignment 7\Assignment7.db

Session 1:

```

1  DELETE FROM Employee
2  WHERE Department_ID = (
3  SELECT Department_ID FROM Department
4  WHERE Name = 'Accounting');

```

Result: query executed successfully. Took 33ms, 2 rows affected

At line 1:
DELETE FROM Employee
WHERE Department_ID = (
SELECT Department_ID FROM Department
WHERE Name = 'Accounting');

Session 2:

```

1  SELECT * FROM Employee;

```

Employee_ID	Lastname	Firstname	MiddleName	Job_ID	Manager_ID	HireDate	Salary	Department_ID
1 7499	Alan	Kevin	J	670	7698	20-FEB-85	1600	20
2 7505	Doyle	Jean	K	671	7839	04-APR-85	2850	20
3 7506	Dennis	Lynn	S	671	7839	15-MAY-85	2750	30
4 7507	Baker	Leslie	D	671	7839	10-JUN-85	2200	40

Result: 4 rows returned in 12ms

At line 1:
SELECT * FROM Employee;

12. Find out whose department has not employees.

SELECT * FROM Employee

WHERE NOT EXISTS(

SELECT Department_ID FROM Department

WHERE Department.Department_ID = Employee.Department_ID);

DB Browser for SQLite - D:\Akash Files\A Sem\Sem 5\Practical Assignments\DBMS\Assignment 7\Assignment7.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragmas Execute SQL

SQL 1

```

1  SELECT * FROM Employee
2  WHERE NOT EXISTS(
3  SELECT Department_ID FROM Department
4  WHERE Department.Department_ID = Employee.Department_ID);

```

Result: 0 rows returned in 8ms
At line 1:
SELECT * FROM Employee
WHERE NOT EXISTS(
SELECT Department_ID FROM Department
WHERE Department.Department_ID = Employee.Department_ID);

Edit Database Cell Mode: Text Import Export Set as NULL

NULL

Type of data currently in cell: NULL
0 byte(s) Apply

SQL Log Show SQL submitted by Application

```

155 PRAGMA database_list;
156 SELECT type,name,sql,tbl_name FROM "main".sqlite_master
157 PRAGMA database_list;
158 SELECT type,name,sql,tbl_name FROM "main".sqlite_master
159 SELECT COUNT(*) FROM (SELECT * FROM Employee
160 WHERE NOT EXISTS(
161 SELECT Department_ID FROM Department
162 WHERE Department.Department_ID = Employee.Department_ID)
163 SELECT * FROM Employee
164 WHERE NOT EXISTS(
165 SELECT Department_ID FROM Department
166 WHERE Department.Department_ID = Employee.Department_ID);
167 PRAGMA database_list;

```

SQL Log Plot DB Schema Remote

UTF-8

13. List out the department wise maximum salary, minimum salary, average salary of the employees

```

SELECT Department_ID, count(*) as No_of_Employees,
       max(Salary), min(Salary), avg(Salary)
  FROM Employee GROUP BY Department_ID;

```

DB Browser for SQLite - D:\Akash Files\A Sem\Sem 5\Practical Assignments\DBMS\Assignment 7\Assignment7.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragmas Execute SQL

SQL 1

```

1  SELECT Department_ID, count(*) as No_of_Employees,
2  max(Salary), min(Salary), avg(Salary)
3  FROM Employee GROUP BY Department_ID;

```

Department_ID	No_of_Employees	max(Salary)	min(Salary)	avg(Salary)
1	20	2850	1600	2225.0
2	30	2750	2750	2750.0
3	40	2200	2200	2200.0

Result: 3 rows returned in 20ms
At line 1:
SELECT Department_ID, count(*) as No_of_Employees,
 max(Salary), min(Salary), avg(Salary)
 FROM Employee GROUP BY Department_ID;

Edit Database Cell Mode: Text Import Export Set as NULL

NULL

Type of data currently in cell: NULL
0 byte(s) Apply

SQL Log Show SQL submitted by Application

```

171 FROM Employee GROUP BY Department_ID;
172 SELECT Department_ID, count(*),
173 max(Salary), min(Salary), avg(Salary)
174 FROM Employee GROUP BY Department_ID LIMIT 0, 3
175 PRAGMA database_list;
176 SELECT type,name,sql,tbl_name FROM "main".sqlite_master
177 SELECT COUNT(*) FROM (SELECT Department_ID, co
178 max(Salary), min(Salary), avg(Salary)
179 FROM Employee GROUP BY Department_ID);
180 SELECT Department_ID, count(*) as No_of_Employees,
181 max(Salary), min(Salary), avg(Salary)
182 FROM Employee GROUP BY Department_ID LIMIT 0, 3
183 PRAGMA database_list;

```

SQL Log Plot DB Schema Remote

14. How many employees who are joined in 1985.

```
SELECT count(*) AS Emp_Joined_in_1985 FROM Employee  
WHERE HireDate like '%85';
```

The screenshot shows the DB Browser for SQLite interface. The main window displays the following SQL query:

```
1 SELECT count(*) AS Emp_Joined_in_1985 FROM Employee  
2 WHERE HireDate like '%85';
```

The results pane shows a single row with the value 4 under the column 'Emp_Joined_in_1985'. Below the results, the output is: "Result: 1 rows returned in 23ms". The SQL Log pane shows the history of the session, including the executed query.

15. Display the employees who are working in “New York”

```
SELECT * FROM Employee  
WHERE Department_ID = (  
SELECT Department_ID FROM Department  
WHERE Location_ID = (  
SELECT Location_ID FROM Location  
WHERE Reginal_Group = 'New York'));
```

DB Browser for SQLite - D:\Akash Files\A Sem\Sem 5\Practical Assignments\DBMS\Assignment 7\Assignment7.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragmas Execute SQL

SQL 1

```

1 SELECT * FROM Employee
2 WHERE Department_ID = (
3   SELECT Department_ID FROM Department
4   WHERE Location_ID = (
5     SELECT Location_ID FROM Location
6     WHERE Reginal_Group = 'New York'));

```

Result: 0 rows returned in 9ms
At line 1:
SELECT * FROM Employee
WHERE Department_ID = (
SELECT Department_ID FROM Department
WHERE Location_ID = (
SELECT Location_ID FROM Location
WHERE Reginal_Group = 'New York'));

Edit Database Cell

Type of data currently in cell: NULL
0 byte(s)

SQL Log

Show SQL submitted by Application

```

272 SELECT "_rowid_"," FROM "main"."Employee" ORDER BY "_rowid_"
273 SELECT COUNT(*) FROM (SELECT "_rowid_"," FROM "main"."Job")
274 SELECT "_rowid_"," FROM "main"."Job" ORDER BY "_rowid_"
275 SELECT "_rowid_"," FROM "main"."Job" ORDER BY "_rowid_"
276 SELECT COUNT(*) FROM (SELECT "_rowid_"," FROM "main"."Job")
277 SELECT COUNT(*) FROM (SELECT "_rowid_"," FROM "main"."Employee")
278 SELECT "_rowid_"," FROM "main"."Employee" ORDER BY "_rowid_"
279 SELECT COUNT(*) FROM (SELECT "_rowid_"," FROM "main"."Department")
280 SELECT "_rowid_"," FROM "main"."Department" ORDER BY "_rowid_"
281 SELECT COUNT(*) FROM (SELECT "_rowid_"," FROM "main"."Location")
282 SELECT "_rowid_"," FROM "main"."Location" ORDER BY "_rowid_"

```

SQL Log Plot DB Schema Remote

UTF-8

16. List our employees with their department names

```

SELECT * FROM Employee, Department
WHERE Employee.Department_ID =
      Department.Department_ID GROUP by Name;

```

DB Browser for SQLite - D:\Akash Files\A Sem\Sem 5\Practical Assignments\DBMS\Assignment 7\Assignment7.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragmas Execute SQL

SQL 1

```

1 SELECT * FROM Employee, Department
2 WHERE Employee.Department_ID = Department.Department_ID
3 GROUP by Name;

```

Employee_ID	Lastname	Firstname	MiddleName	Job_ID	Manager_ID	HireDate	Salary	Department_ID	Department_ID	Name
1 7507	Baker	Leslie	D	671	7839	10-JUN-85	2200	40	40	Operation
2 7499	Alan	Kevin	J	670	7698	20-FEB-85	1600	20	20	Research
3 7506	Dennis	Lynn	S.	671	7839	15-MAY-85	2750	30	30	Sale

Result: 3 rows returned in 25ms.
At line 1:
SELECT * FROM Employee, Department
WHERE Employee.Department_ID = Department.Department_ID
GROUP by Name;

Edit Database Cell

Type of data currently in cell: Text / Numeric
1 char(s)

SQL Log

Show SQL submitted by Application

```

308 SELECT * FROM Employee, Department
309 WHERE Employee.Department_ID = Department.Department_ID
310 GROUP by Name LIMIT 0, 49989;
311 PRAGMA database_list;
312 SELECT type, name, sql, tbl_name FROM "main".sql
313 SELECT "_rowid_"," FROM "main"."Department"
314 SELECT COUNT(*) FROM (SELECT "_rowid_"," FROM "main"."Department")
315 SELECT COUNT(*) FROM (SELECT "_rowid_"," FROM "main"."Employee")
316 SELECT "_rowid_"," FROM "main"."Employee" ORDER BY "_rowid_"
317 SELECT COUNT(*) FROM (SELECT "_rowid_"," FROM "main"."Location")
318 SELECT "_rowid_"," FROM "main"."Location" ORDER BY "_rowid_"
319

```

SQL Log Plot DB Schema Remote

