**LAB 4: DQL (DATA QUERY LANGUAGE) WITH COMPLEX QUERIES**

**Objective:**

To understand and implement advanced Data Query Language (DQL) concepts in SQL, with a focus on handling complex queries involving conditional logic, aggregate functions, and group operations.

**Theory:**

**Data Query Language (DQL)** is a subset of SQL primarily used for retrieving and analyzing data from relational databases. Advanced DQL techniques enhance its utility by enabling users to:

* **Handle multiple conditions:** Utilize CASE, WHERE, and HAVING clauses to implement conditional logic in queries.
* **Perform aggregate analysis:** Leverage functions such as SUM, COUNT, AVG, MIN, and MAX for in-depth data analysis.
* **Group and filter data:** Use GROUP BY to organize data based on shared attributes and the HAVING clause to apply conditions to grouped data.
* **Integrate subqueries:** Employ nested queries to address hierarchical or dependent data retrieval scenarios.

**Key Features of Complex Queries:**

* **Conditional Logic**: Dynamically modify or analyze data using CASE statements and conditional clauses.
* **Aggregate Analysis**: Perform statistical operations, such as calculating totals, averages, or extreme values, to derive meaningful insights from datasets
* **Grouping and Filtering**: Organize data into logical groups using GROUP BY and refine results by applying conditions on aggregated data through the HAVING clause.
* **Subqueries**: Use queries within queries to handle multi-layered or dependent data operations efficiently.

**SQL QUERY:**

--CREATE DATABASE LAB4;

USE LAB4;

--DQL With complex queries

DROP TABLE issue;

DROP TABLE book;

DROP TABLE student;

DROP TABLE teacher;

CREATE TABLE book(

bid INT PRIMARY KEY,

bname VARCHAR(50),

publication VARCHAR(50),

author VARCHAR(50),

price DECIMAL(8,2)

);

SELECT \*FROM book;



-Created table book along with its various attributes.

CREATE TABLE student(

sid INT PRIMARY KEY,

sfname VARCHAR(50),

slname VARCHAR(50),

sbranch VARCHAR(50),

address VARCHAR(50)

);



-Created table student along with its various attributes.

CREATE TABLE teacher(

tid INT PRIMARY KEY,

tfname VARCHAR(50),

tlname VARCHAR(50),

tbranch VARCHAR(50),

tsalary BIGINT,

hid INT

);



-Created table teacher along with its various attributes.

CREATE TABLE issue(

iid INT PRIMARY KEY,

bid INT FOREIGN KEY REFERENCES book(bid),

sid INT FOREIGN KEY REFERENCES student(sid) ON DELETE CASCADE ON UPDATE SET NULL,

tid INT FOREIGN KEY REFERENCES teacher(tid) ON DELETE CASCADE ON UPDATE SET NULL,

dateOfIssue DATE,

);



-Created table issue along with its various attributes.

INSERT INTO book VALUES (101, 'Atomic Habits','Hamro Publication', 'Navathe', 500.00);

INSERT INTO book VALUES (102, 'The power of your subconscious mind', 'Modern Pub','Galvin', 600.00);

INSERT INTO book VALUES (103, 'Deep work','AdvancePub', 'Weiss', 450.00);

INSERT INTO book VALUES (104, 'The Let Them Theory', 'Apub','Tanenbaum', 700.00);

INSERT INTO book VALUES (105, 'The Anxious Generation','Bpub', 'Anthony', 800.00);

INSERT INTO book VALUES (106, 'The Daily Stoic ', 'Cpub','Bob', 900.00);

INSERT INTO book VALUES (107, 'Nexus ', 'Dpub','Casey', 200.00);

INSERT INTO book VALUES (108, 'Elon Musk', 'Epub','Donald', 300.00);

INSERT INTO book VALUES (109, 'The psychology of Money', 'Fpub','Emerald', 400.00);

INSERT INTO book VALUES (110, 'Revenge of the Tipping Point ','Gpub', 'Franklin', 500.00);

INSERT INTO book VALUES (111, 'Outlive','Gpub', 'Franklin', 500.00);

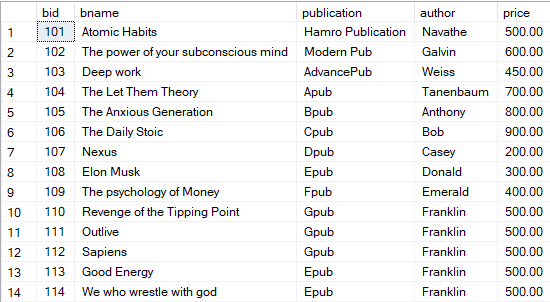
INSERT INTO book VALUES (112, 'Sapiens','Gpub', 'Franklin', 500.00);

INSERT INTO book VALUES (113, 'Good Energy','Epub', 'Franklin', 500.00);

INSERT INTO book VALUES (114, 'We who wrestle with god','Epub', 'Franklin', 500.00);

Output:

-inserted various data into the table book.



INSERT INTO student VALUES (1, 'Deepak','Thapa', 'BCT', 'Kageshowri Manohara'),

(2, 'Bipana','Ranabhat', 'BCE','Kathmandu'),

(3, 'Dikshya', 'Shrestha','BCT', 'Kathmandu'),

(4,'Abhiyan','Paudel','BEX','Swayambhu'),

(5,'Aashutosh','Jha','BEI','Kritipur'),

(6,'Abhinav','Sharma','BCT','Tokha'),

(7,'Adrin','Pradhan','BEI','Pokhara'),

(8,'Bishranta','Paudel','BEX','Chitwan'),

(9,'Aashutosh','Paudel','BCT','Jamal'),

(10,'Isha','Karki','BAG','Tokha');



INSERT INTO teacher VALUES (1, 'Dhawa','Adhikari', 'BCT', 100000,1),

(2, 'Sujan','Ranabhat', 'BCE',200000,1),

(3, 'Ritu', 'Shrestha','BCT',300000, 2),

(4,'Sharad','Paudel','BEX',400000,4),

(5,'Nabin','Jha','BEI',500000,4),

(6,'Anju','Sharma','BCT',600000,4),

(7,'Aman','Pradhan','BEI',700000,4),

(8,'Rubi','Paudel','BEX',800000,4),

(9,'Nikesh','Paudel','BCT',900000,10),

(10,'Ravi','Karki','BAG',950000,10);



INSERT INTO issue VALUES (1001,101,NULL,2,'2024-06-01'),

(1002,102,1,NULL,'2024-06-02'),

(1003,103,7,NULL,'2024-06-03'),

(1004,104,NULL,9,'2024-06-04'),

(1005,105,5,NULL,'2024-06-05'),

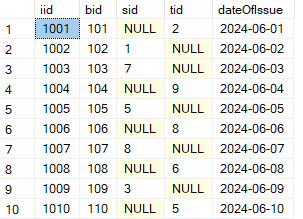
(1006,106,NULL,8,'2024-06-06'),

(1007,107,8,NULL,'2024-06-07'),

(1008,108,NULL,6,'2024-06-08'),

(1009,109,3,NULL,'2024-06-09'),

(1010,110,NULL,5,'2024-06-10');



**QUESTIONS:**

1. Update the salary of teacher by 10% if teacher earns in between 110000 to 140000, by 5% if teacher earns more than 140000 else by 20%.

**Ans:** UPDATE teacher SET tsalary = CASE WHEN tsalary>110000 AND tsalary<140000

THEN 1.1\*tsalary

WHEN tsalary>140000 THEN 1.05\*tsalary

ELSE 1.2\*tsalary

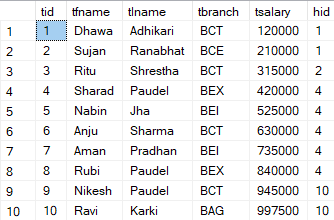
END

SELECT \*FROM teacher;

(πtid, tsalary×1.1(σtsalary>110000∧tsalary<140000(teacher)))∪(πtid, tsalary×1.05(σtsalary>140000(teacher)))∪

(πtid, tsalary×1.2(σtsalary≤110000∨tsalary≥140000(teacher)))

**Output:** Updated teachers table (tsalary) after above query implementation.



* Here, the salary of Dhawa is updated by 20% since his salary was less than 110000 whereas all others salary was updated by 5% since everyone’s salary was more than 140000.

1. What is the maximum salary of teacher, total salary of all teachers, no of teachers getting salary, minimum salary of teacher and average salary of teachers? Display them as their respective names.

**Ans:** SELECT MAX(tsalary) AS maximun\_salary FROM teacher;

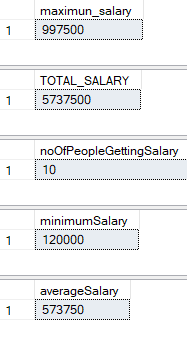
SELECT SUM(tsalary) AS TOTAL\_SALARY FROM teacher;

SELECT COUNT(tsalary)AS noOfPeopleGettingSalary FROM teacher;

SELECT MIN(tsalary) AS minimumSalary FROM teacher;

SELECT AVG(tsalary) AS averageSalary FROM teacher;

**Output:**



G MAX(tsalary) (teacher)

G MIN(tsalary) (teacher)

G AVG(tsalary) (teacher)

G SUM(tsalary) (teacher)

G COUNT(tsalary) (teacher)

-here, we have displayed total, max, count, min and average salary of teachers from the teachers table and to display it properly we have used: AS columnName to display the result along with its column name if we don’t use it then, we get output like this:

SELECT MAX(tsalary) FROM teacher;



1. Display the details of all the teachers with maximum salary.

**Ans:** SELECT \*FROM teacher WHERE tsalary=(SELECT MAX(tsalary) FROM teacher);

**Output:**



σ(salary = G(MAX(tsalary))(teacher))(teacher)

1. Display the details of teacher with 2nd maximum salary.

**Ans:** SELECT MAX (tsalary) AS SecondMaxSalary FROM teacher WHERE tsalary<(SELECT MAX(tsalary) FROM teacher);

**Output:**



**All details:**

SELECT \* FROM teacher WHERE tsalary=(SELECT MAX(tsalary) FROM teacher WHERE tsalary<(SELECT MAX(tsalary) FROM teacher));



c1= G MAX(tsalary) (teacher)

e1=σ(tsalary <c1)(teacher)

c2=G MAX(tsalary) (e1)

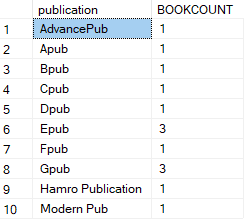


1. Display the count of the book according to the book publisher/ publication.

**Ans:** SELECT publication,COUNT(bid) AS BOOKCOUNT FROM book GROUP BY publication;

γpublication,COUNT(bid)→BOOKCOUNT(book)

**Output:**



1. Display the count of the book according to the book publisher/publication having count greater than 2.

**Ans:** SELECT publication,COUNT(bid) AS BOOKCOUNT FROM book GROUP BY publication HAVING COUNT(bid)>2;

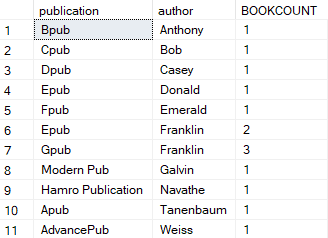
Ypublication,count(bid)>2→BOOKCOUNT(book)



1. Display the count of the book according to the book publisher/publication and author.

**Ans:** SELECT publication,author,COUNT(bid) AS BOOKCOUNT FROM book GROUP BY publication,author ;

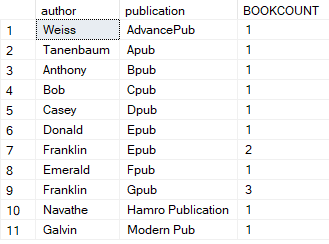
γpublication,author,count(bid)→BOOKCOUNT(book)



1. Display the count of the book according to the book author and publisher/publication(reverse).

**Ans:** SELECT author,publication,COUNT(bid) AS BOOKCOUNT FROM book GROUP BY author,publication ;

Yauthor,publication,count(bid)→BOOKCOUNT (book)



**Assignment Questions:**

1. Update the price of book by 10% if the price of the book is in between 300 to 600, by 5% if the price of the book is more than 600 else increase the price of the book by 20%.

**Ans:** UPDATE book SET price= CASE WHEN price>300 AND price<600

THEN 1.1\*price

WHEN price>600

THEN 1.05\*price

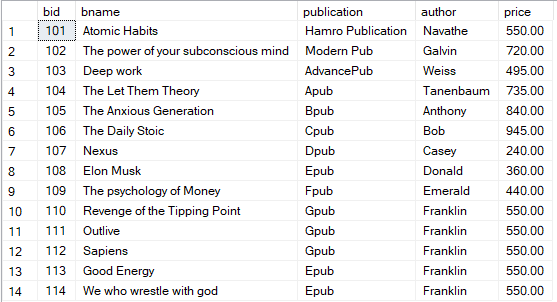
ELSE 1.2\*price

END

SELECT \*FROM book;

(πbid, price×1.1(σprice>300∧price<600(book)))∪(πbid, price×1.05(σprice>600(book)))∪

(πbid, price×1.2(σprice≤300∨price≥600(book)))



1. What is the maximum price of the book, total price of all books, no of books, minimum price of the book and average price of books.

**Ans:** SELECT MAX(price) AS maximumPriceOfBook FROM book;

SELECT SUM(price) AS totalPriceOfAllBooks FROM book;

SELECT COUNT(price) AS noOfBooks FROM book;

SELECT MIN(price) AS minimumPriceOfBook FROM book;

SELECT AVG(price) AS averagePriceOfBooks FROM book;

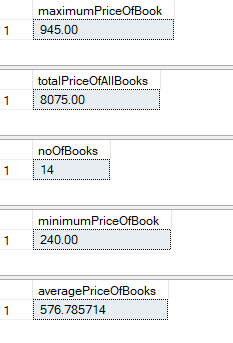
G MAX(price) →maximunPriceOfBook (book)

G SUM(price) →totalPriceOfBooks (book)

G COUNT(price) →noOfBooks (book)

G MIN(price) →minimumPriceofBook (book)

G AVG(price) →averagePriceOfBooks (book)



1. Display the details of books with maximum price.

**Ans:** SELECT \*FROM book WHERE price=(SELECT MAX(price) FROM book);

σ(price= G(MAX(price))(book))(book)



1. Display the details of book with 2nd maximum price.

**Ans:** SELECT \* FROM book WHERE price=(SELECT MAX(price) FROM book WHERE price<(SELECT MAX(price) FROM book));

c1= G MAX(price) (book)

e1=σ(price <c1)(book)

c2=G MAX(price) (e1)

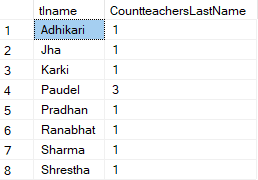
πe1.price(σ(price =c2)(e1))



1. Display the count of teacher according to the teachers last name.

**Ans:** SELECT tlname,COUNT(tid) AS CountteachersLastName FROM teacher GROUP BY tlname;

γtlname,COUNT(tid)→CountteachersLastName(teacher)



1. Display the count of teacher according to the teachers branch if count is greater than 3.

**Ans:** SELECT tbranch,COUNT(tid) AS teachersBranch FROM teacher GROUP BY tbranch HAVING COUNT(tid)>3;

σteachersBranch>3(γtbranch, count(tid)->teachersBranch(teacher))



1. Display the count of teacher according to the teachers last name if the count is greater than 2.

**Ans:** SELECT tlname,COUNT(tid) AS teachersLastName FROM teacher GROUP BY tlname HAVING COUNT(tid)>2;

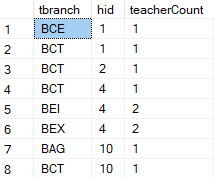
γtlname,COUNT(tid)→CountteachersLastName>2(teacher)



1. Display the count of the teacher according to the branch and head id.

**Ans:** SELECT tbranch,hid,COUNT(tid) AS teacherCount FROM teacher GROUP BY tbranch,hid;

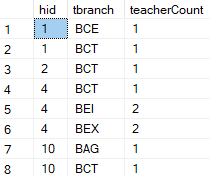
Ytbranch,hid,count(tid)→teacherCount (teacher)



1. Display the count of the teacher according to the head id and branch.

**Ans:** SELECT hid,tbranch,COUNT(tid) AS teacherCount FROM teacher GROUP BY tbranch,hid;

Yhid,tbranch,count(tid)→teacherCount (teacher)



**Discussion:**

This lab focused on advanced SQL operations, exploring complex queries with conditional logic and data aggregation. Key tasks included dynamically adjusting teacher salaries based on specific conditions, calculating statistical measures such as maximum, minimum, and average salaries, and using subqueries to identify details like the second-highest salary. We also applied aggregate functions with GROUP BY and used the HAVING clause to filter grouped data, analyzing information about books categorized by publishers and authors.  
  
Through these practical exercises, we developed the ability to write intricate SQL queries tailored to address real-world database scenarios. The lab highlighted the importance of integrating various SQL features to manage and analyze data accurately and efficiently.

**Conclusion:**

This lab enhanced our understanding of advanced SQL techniques through hands-on tasks. By working on conditional updates, statistical analysis, and group-based data exploration, we gained valuable experience in creating efficient and practical SQL queries for real-world applications.