DATABASE MANGEMENT SYSTEMS
2015-2016 FALL SEMESTER
LABORATORY MANUAL

Experiment 5

Aggregate Functions

The SQL aggregate functions, as their title suggests are used to retrieve minimum and maximum values from a column, to sum values in a column, to get the average of a column values, or to simply count a number of records according to a search condition.

MIN	returns the smallest value in a given column
MAX	returns the largest value in a given column
SUM	returns the sum of the numeric values in a given column
AVG	returns the average value of a given column
COUNT	returns the total number of values in a given column
COUNT(*)	returns the number of rows in a table

```
SELECT COUNT(*) FROM Cars WHERE Color = 'Gray';
SELECT MIN(Year) FROM Cars;
```

Order By Clause

The ORDER BY keyword is used to sort the result-set by one or more columns.

The ORDER BY keyword sorts the records in ascending order by default. To sort the records in a descending order, you can use the DESC keyword.

```
SELECT * FROM Customers ORDER BY Country;
```

In & Between

The IN operator allows you to specify multiple values in a WHERE clause.

```
SELECT * FROM Customers WHERE City IN ('Paris','London');
```

The BETWEEN operator selects values within a range. The values can be numbers, text, or dates.

```
SELECT * FROM Products WHERE Price BETWEEN 10 AND 20;
```

Union

The purpose of the SQL UNION query is to combine the results of two queries together. In this respect, UNION is somewhat similar to JOIN in that they are both used to related information from multiple tables. One restriction of UNION is that all corresponding columns

need to be of the same data type. Also, when using UNION, only distinct values are selected (similar to SELECT DISTINCT).

```
SELECT Txn_Date FROM Store_Information
UNION
SELECT Txn_Date FROM Internet_Sales;
```

Intersect

Similar to the UNION command, INTERSECT also operates on two SQL statements. The difference is that, while UNION essentially acts as an OR operator (value is selected if it appears in either the first or the second statement), the INTERSECT command acts as an AND operator (value is selected only if it appears in both statements).

```
SELECT Txn_Date FROM Store_Information
INTERSECT
SELECT Txn_Date FROM Internet_Sales;
```

Except

The EXCEPT command operates on two SQL statements. It takes all the results from the first SQL statement, and then subtract out the ones that are present in the second SQL statement to get the final answer. If the second SQL statement includes results not present in the first SQL statement, such results are ignored.

```
SELECT Txn_Date FROM Store_Information
EXCEPT
SELECT Txn_Date FROM Internet_Sales;
```

Group By

The GROUP BY clause will gather all of the rows together that contain data in the specified column(s) and will allow aggregate functions to be performed on the one or more columns. This can best be explained by an example:

```
SELECT max(salary), dept
FROM employee
GROUP BY dept;
```

Having

The HAVING clause allows you to specify conditions on the rows for each group - in other words, which rows should be selected will be based on the conditions you specify. The HAVING clause should follow the GROUP BY clause if you are going to use it.

```
SELECT dept, avg(salary)
FROM employee
GROUP BY dept
HAVING avg(salary) > 20000;
```

Subquery

It is possible to embed a SQL statement within another. When this is done on the WHERE or the HAVING statements, we have a subquery construct. The syntax is as follows:

```
SELECT "column_name1"
FROM "table_name1"
WHERE "column_name2" [Comparison Operator]
(SELECT "column_name3"
FROM "table_name2"
WHERE "condition");
```

[Comparison Operator] could be equality operators such as =, >, <, >=, <=. It can also be a text operator such as "LIKE". The portion in red is considered as the "inner query", while the portion in green is considered as the "outer query".

Let's use the following two tables to illustrate the concept of subquery.

Table *Store_Information*

Store_Name	Sales	Txn_Date
Los Angeles	1500	Jan-05-1999
San Diego	250	Jan-07-1999
Los Angeles	300	Jan-08-1999
Boston	700	Jan-08-1999

Table *Geography*

Region_Name	Store_Name
East	Boston
East	New York
West	Los Angeles
West	San Diego

We want to use a subquery to find the sales of all stores in the West region.

```
SELECT SUM (Sales) FROM Store_Information
WHERE Store_Name IN
(SELECT Store_Name FROM Geography
WHERE Region_Name = 'West');
```

In this example, instead of joining the two tables directly and then adding up only the sales amount for stores in the West region, we first use the subquery to find out which stores are in the West region, and then we sum up the sales amount for these stores.

In the above example, the inner query is first executed, and the result is then fed into the outer query. This type of subquery is called a simple subquery. If the inner query is dependent on the outer query, we will have a correlated subquery. An example of a correlated subquery is shown below:

```
SELECT SUM (a1.Sales) FROM Store_Information a1
WHERE a1.Store_Name IN
(SELECT Store_Name FROM Geography a2
WHERE a2.Store_Name = a1.Store_Name);
```

Notice the WHERE clause in the inner query, where the condition involves a table from the outer query.

Exists

EXISTS simply tests whether the inner query returns any row. If it does, then the outer query proceeds. If not, the outer query does not execute, and the entire SQL statement returns nothing.

```
SELECT SUM(Sales) FROM Store_Information
WHERE EXISTS
(SELECT * FROM Geography
WHERE Region_Name = 'West');
```

At first, this may appear confusing, because the subquery includes the [region_name = 'West'] condition, yet the query summed up stores for all regions. Upon closer inspection, we find that since the subquery returns more than 0 row, the EXISTS condition is true, and the condition placed inside the inner query does not influence how the outer query is run.

Exercises :

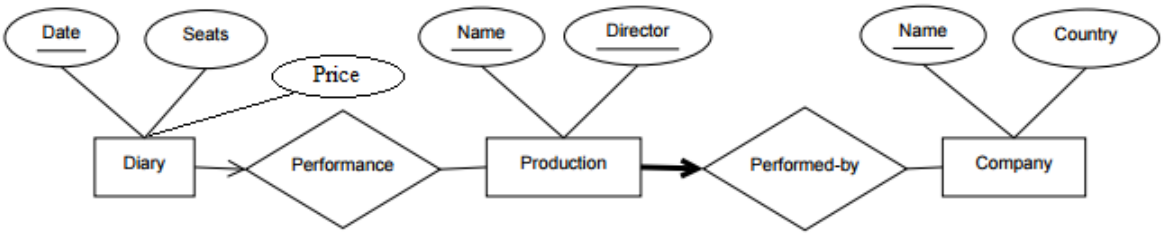

Fig. 1. The ER diagram for a theatre manager database

Theatre manager database ER diagram is given in Fig. 1. This database keeps information of forthcoming shows by acting companies. Answer following questions using SQL according to this database.

1. Create database named as “db_TheatreManager” .
2. Create tables according to ER diagram in Fig. 1.
3. Insert at least 5 records in each table.
4. Find the max price of production name performed by company named as ‘A’.
5. Find the number of productions performed by company named as ‘B’.
6. Select dates of production performed by company ‘A’ and ‘B’.
7. Sort the seat number of production ascending.
8. Display the date and name of production between 23.11.2015 and 29.11.2015.
9. Select direction of production performed by company ‘A’ or ‘B’.
10. Display name of production performed by company ‘B’ but not ‘C’.
11. Find the name of all production performed in ‘Eskişehir’. Use ‘IN’ and ‘EXISTS’ syntax for this question and explain the difference of them.
12. Display average price and name of production that has average price greater than 10 YTL.