CHAPTER 6

Module IV

Display Data using SQL

Syllabus

Set operations, aggregate function, null values, Complex Retrieval Queries using Group By

6.1 SELECT Statement

Q. Explain use of SELECT in SQL. Give its syntax.

Q. What is SQL ? Explain the SELECT and FROM clause of SQL queries with appropriate example.

- The SELECT statement in SQL is used to select data and display from a table in database.
- The SELECT can also include SQL built in function to manipulate the data in database just for display purpose like aggregate functions.
- The SELECT can be used for displaying various attributes or columns of a table.

```
SELECT [DISTINCT] [ * /Column1,Column2...]
FROM <Table_Name>;
```

1. Display all data in table

- SELECT is basic statement used to retrieve all or some columns of data from table.
- We can select all columns from table by specify * as column name.

Syntax

```
SQL> SELECT *
     FROM <Table_Name>;
```

Example

Select all details of employees in company.

```
MYSQL> SELECT *
       FROM Employee;

 ID   NAME      AGE  ADDRESS    SALARY
 ---- --------- ---- ---------- ----------
 1    Yogesh    32   Mumbai     12000.00
 2    Karan     25   Nagar      2500.00
 3    Akshay    43   Delhi      4000.00
 4    Chetan    25   Mumbai     6800.00
 5    Hardik    37   Mumbai     8300.00
 6    Kamal     32   Pune       4100.00
 7    Mahesh    24   Pune       15000.00

7 rows in set (0.00 sec)
```

2. Display limited column data in table

- SELECT statement can be used to retrieve specific columns of data from table to maintain security of data.
- We can select columns from table specifying names of required columns.

Syntax

```
SELECT Column1, Column2
FROM <Table_Name>;
```

Example

Select Id and names of employees in company.

```
MYSQL> SELECT ID, NAME
       FROM Employee;

 ID   NAME
 ---- ----------
 1    Yogesh
 2    Karan
 3    Akshay
 4    Chetan
 5    Hardik
 6    Kamal
```

```
7       Mahesh

7 rows in set (0.01 sec)
```

3. Display unique records in table

- SELECT is basic statement used to retrieve all or some columns of data from table as explained in above examples.
- SELECT can have two options as SELECT ALL records in which query result including duplicate records along with unique records or SELECT DISTINCT records to show only unique records.
- SELECT ALL is default and SELECT DISTNCT will search for distinct rows of in table.
- We can either select all columns from table by specify * as column name or we can specify the required name of columns. In any case DISTINCT will look for unique record set for display purpose only.
- (It cannot remove duplicate records from original table.)

Syntax

```
SELECT  DISTINCT [*/Column1,Column2...]
FROM <Table_Name>;
```

Example

Select different locations of above employees.

```
MYSQL> SELECT  DISTINCT ADDRESS
       FROM        Employee;

ADDRESS
--------------
Mumbai
Nagar
Delhi
Pune

4 rows in set (0.04 sec)
```

- The DISTINCT keyword can be used with the selection of all columns. In this case, it will print all tuples by ignoring repeated tuples in the relation. If no values repeated in a row then DISTINCT will have no effect on results.
- Select all details of employees in company (Avoid repeated data).

```
MYSQL> SELECT  DISTINCT *
       FROM    Employee;

ID   NAME      AGE  ADDRESS   SALARY
---- --------- ---- --------- --------
1    Yogesh    32   Mumbai    12000.00
2    Karan     25   Nagar     2500.00
3    Akshay    43   Delhi     4000.00
4    Chetan    25   Mumbai    6800.00
5    Hardik    37   Mumbai    8300.00
6    Kamal     32   Pune      4100.00
7    Mahesh    24   Pune      15000.00

7 rows in set (0.00 sec)
```

6.2 Ordering Query Results

Q. Explain 'ORDER BY' clause.

1. Introduction

- The ORDER BY keyword is used to sort the result-set by a specified column.
- The ORDER BY keyword sorts the records in ascending order by default.
- If you want to sort the records in a descending order, you can use the DESC keyword.

2. Syntax

```
SELECT column_name
FROM   table_name
ORDER BY  column_name ASC|DESC
```

3. Example

Select details of employees in company in alphabetical order of names.

```
MYSQL> SELECT*
       FROM     Employee
       ORDER BY  NAME;

ID   NAME      AGE   ADDRESS   SALARY
---- --------- ----- --------- ---------
3    Akshay    43    Delhi     4000.00
4    Chetan    25    Mumbai    6800.00
5    Hardik    37    Mumbai    8300.00
```

6	Kamal	32	Pune	4100.00
2	Karan	25	Nagar	2500.00
7	Mahesh	24	Pune	15000.00
1	Yogesh	32	Mumbai	12000.00

7 rows in set (0.01 sec)

- Now we want to select all employees from the table above, however, we want to sort the employees by reverse order of their names.
- In second query ORDER BY 2 indicates order by second column of table.

```
MYSQL> SELECT *
       FROM   Employee
       ORDER BY  NAME DESC;
OR
MYSQL> SELECT *
       FROM   Employee
       ORDER BY  2 DESC;
```

ID	Name	Age	Address	Salary
1	Yogesh	32	Mumbai	12000.00
7	Mahesh	24	Pune	15000.00
2	Karan	25	Nagar	2500.00
6	Kamal	32	Pune	4100.00
5	Hardik	37	Mumbai	8300.00
4	Chetan	25	Mumbai	6800.00
3	Akshay	43	Delhi	4000.00

7 rows in set (0.00 sec)

- The ordering can be done on numeric column like salary by default ascending OR descending ordering.

```
MYSQL> SELECT *
       FROM   Employee
       ORDER BY  SALARY DESC;
OR
MYSQL> SELECT *
       FROM Employee
       ORDER BY  5 DESC;
```

ID	Name	Age	Address	Salary
7	Mahesh	24	Pune	15000.00
1	Yogesh	32	Mumbai	12000.00
5	Hardik	37	Mumbai	8300.00
4	Chetan	25	Mumbai	6800.00
6	Kamal	32	Pune	4100.00
3	Akshay	43	Delhi	4000.00
2	Karan	25	Nagar	2500.00

7 rows in set (0.00 sec)

- The ordering can be done on multiple columns also by ascending or descending ordering. In such case ordering is first done by first column on list if this column has same value then only second column is considered for further ordering.

```
MYSQL> SELECT *
       FROM   Employee
    ORDER BY  ADDRESS ASC, SALARY DESC;
```

ID	NAME	AGE	ADDRESS	SALARY
3	Akshay	43	Delhi	4000.00
1	Yogesh	32	Mumbai	12000.00
5	Hardik	37	Mumbai	8300.00
4	Chetan	25	Mumbai	6800.00
2	Karan	25	Nagar	2500.00
7	Mahesh	24	Pune	15000.00
6	Kamal	32	Pune	4100.00

7 rows in set (0.01 sec)

- If values in first column contains all unique values then second column will not be considered.

```
MYSQL> SELECT *
       FROM   Employee
       ORDER BY  NAME DESC, SALARY DESC;
```

ID	Name	Age	Address	Salary
1	Yogesh	32	Mumbai	12000.00
7	Mahesh	24	Pune	15000.00
2	Karan	25	Nagar	2500.00
6	Kamal	32	Pune	4100.00

```
5   Hardik   37   Mumbai   8300.00
4   Chetan   25   Mumbai   6800.00
3   Akshay   43   Delhi    4000.00

7 rows in set (0.00 sec)
```

6.3 Displaying Selected Rows / Selection Operation in SQL

Q. What is SQL ? Explain the Where clause of SQL queries with appropriate example.

a. Introduction

- The SQL can print selected rows by applying selection operation in SQL with help of 'WHERE' Clause.
- WHERE statement is an optional statement.
- The condition written in WHERE clause is a row qualifier condition that row must satisfy to print that row in final result of query.
- The query without WHERE condition will print all rows by default.

b. Syntax

```
SELECT *
FROM   Table_Name
[WHERE     condition]
```

c. Examples

Select details of employees in company having salary above 5000.

```
MYSQL> SELECT *
       FROM    Employee
       WHERE   SALARY > 5000
       Order By ID;

ID   NAME      AGE   ADDRESS   SALARY
---- --------- ----- --------- ---------
1    Yogesh    32    Mumbai    12000.00
4    Chetan    25    Mumbai    6800.00
5    Hardik    37    Mumbai    8300.00
7    Mahesh    24    Pune      15000.00

4 rows in set (0.01 sec)
```

d. Basic query execution

1. In above given query the first executable statement is **FROM**, this statement will load the all rows in EMPLOYEE table from DBMS server to local memory of PC executing above query.
2. **WHERE** clause acts like a **row filter** using condition (Salary>5000); it will only select the rows of EMPLOYEE table having Salary over 5000 and other remaining employee tuples are deleted from memory.
3. **ORDER BY** statement will arrange output in proper order of ID column in ascending order by default.
4. At last the **SELECT** statement either prints all or specific columns onto client computer.

Fig. 6.3.1 : SQL Query Execution Hierarchy

Points to remember

- SQL statements are not case sensitive.
- Keywords cannot be abbreviated or split across lines.
- Clauses are usually placed on separate lines.
- Indents are used to enhance readability.

e. Arithmetic Comparison (=, <, <=, >, >=, <>)

- The comparison of attribute value with some constant values can be done with help of arithmetic comparisons.
- Find all employees not belongs to Mumbai.

```
MYSQL> SELECT *
       FROM    Employee
       WHERE   ADDRESS <> 'MUMBAI'
ID   NAME      AGE   ADDRESS     SALARY
---- --------- ----- ----------- ----------
2    Karan     25    Nagar       2500.00
3    Akshay    43    Delhi       4000.00
6    Kamal     32    Pune        4100.00
7    Mahesh    24    Pune        15000.00

4 rows in set (0.01 sec)
```

f. BETWEEN Clause

- BETWEEN statement used to find out all tuple whose attribute value is ranging between given range of values.
- This clause also includes the boundary values.
- Find all employees with salary between 2000 and 10000.

```
MYSQL> SELECT *
       FROM    Employee
       WHERE   SALARY BETWEEN 2000 AND 10000;
```

ID	Name	Age	Address	Salary
2	Karan	25	Nagar	2500.00
3	Akshay	43	Delhi	4000.00
4	Chetan	25	Mumbai	6800.00
5	Hardik	37	Mumbai	8300.00
6	Kamal	32	Pune	4100.00

5 rows in set (0.03 sec)

g. IN Clause

- IN clause will acts like set comparison operator.
- It checks attribute value for equality with at least one value from given set of comparison values specified in IN clause.
- Find all employees located at Mumbai or Pune.

```
MYSQL> SELECT *
       FROM    Employee
       WHERE   ADDRESS IN ('Mumbai', 'Pune');
```

ID	Name	Age	Address	Salary
1	Yogesh	32	Mumbai	12000.00
4	Chetan	25	Mumbai	6800.00
5	Hardik	37	Mumbai	8300.00
6	Kamal	32	Pune	4100.00
7	Mahesh	24	Pune	15000.00

5 rows in set (0.00 sec)

h. Applying multiple conditions using Logical operators

- The Logical operators are those operators which returns true or false.
- They return a true or false values to combine one or more true or false values returned by evaluation of more than one expression.

AND Operator

- Logical AND operator compares between multiple Booleans values returned by expressions and returns true when all expressions returns to true.
- Example
- Find all employees located at Mumbai or Pune and salary above 2000.

```
MYSQL> SELECT *
       FROM    Employee
       WHERE   ADDRESS IN ('Mumbai', 'Pune')
       AND SALARY > 2000;
```

ID	NAME	AGE	ADDRESS	SALARY
1	Yogesh	32	Mumbai	12000.00
4	Chetan	25	Mumbai	6800.00
5	Hardik	37	Mumbai	8300.00
6	Kamal	32	Pune	4100.00
7	Mahesh	24	Pune	15000.00

5 rows in set (0.01 sec)

OR Operator

- Logical AND operator compares between multiple Booleans values returned by expressions and returns true when at least one expression returns to true.
- Example
- Find all employees located at Mumbai or Pune or salary above 2000.

```
MYSQL> SELECT *
       FROM    Employee
       WHERE   ADDRESS IN ('Mumbai', 'Pune')
       OR SALARY > 2000;
```

ID	NAME	AGE	ADDRESS	SALARY
1	Yogesh	32	Mumbai	12000.00
2	Karan	25	Nagar	2500.00
3	Akshay	43	Delhi	4000.00
4	Chetan	25	Mumbai	6800.00
5	Hardik	37	Mumbai	8300.00
6	Kamal	32	Pune	4100.00
7	Mahesh	24	Pune	15000.00

7 rows in set (0.01 sec)

NOT Operator

- Not logical operator takes a single Boolean value of expression and changes its from false to true or from true to false
- Example
- Find all employees located at Mumbai or Pune.

```
MYSQL> SELECT *
       FROM Employee
       WHERE ADDRESS NOT IN ('Mumbai', 'Pune');
```

ID	NAME	AGE	ADDRESS	SALARY
2	Karan	25	Nagar	2500.00
3	Akshay	43	Delhi	4000.00

Find all employees not between range of 2000 and 10000.

```
MYSQL> SELECT *
       FROM Employee
       WHERE SALARY NOT BETWEEN 2000 AND 10000;
```

ID	NAME	AGE	ADDRESS	SALARY
1	Yogesh	32	Mumbai	12000.00
7	Mahesh	24	Pune	15000.00

6.4 Limit Clause – Restricting Rows in Result

Q. Explain Limit clause.

a. Introduction

- The LIMIT clause is used to constrain the number of rows returned by the SELECT statement.
- LIMIT takes one or two numeric arguments, which must both be non-negative integer constants.
- This clause operates just before printing result, it means WHERE and ORDER BY clauses operates before LIMIT clause.

b. Syntax

```
SELECT *
FROM Table_Name
[LIMIT number]
```

c. Example

Select top 3 salaried employees from company.

```
MYSQL> SELECT *
       FROM Employee
ORDER BY SALARY DESC
LIMIT 3;
```

ID	NAME	AGE	ADDRESS	SALARY
7	Mahesh	24	Pune	15000.00
1	Yogesh	32	Mumbai	12000.00
5	Hardik	37	Mumbai	8300.00

Select 2nd and 3rd largest salary in company.

```
MYSQL> SELECT *
       FROM Employee
ORDER BY SALARY DESC
LIMIT 2, 2;
```

ID	NAME	AGE	ADDRESS	SALARY
1	Yogesh	32	Mumbai	12000.00
5	Hardik	37	Mumbai	8300.00

6.5 Aggregate Functions

Q. Explain Aggregate function with example.
MU - May 16, Dec. 16, 5 Marks

Q. List aggregate functions and justify the need of any two aggregate functions.
MU - Dec. 18, 5 Marks

Q. What are the different aggregate functions used in SQL? Explain with the help of example.
MU - May 19, 5 Marks

Q. Write short note on aggregate function in SQL.
MU - Dec. 19, 5 Marks

1. Introduction

- The management report require lot of decision making data to get summarized information from table like average, sum, minimum.
- SQL provides various aggregate functions which can summarize data of given table.
- The function operates on the table data produces a single output.
- Such queries are generally used for producing reports and summary forms in an application.
- The aggregate data functions ignore all null value in column on which aggregate data function if applied.

2. Types of aggregate functions

a. **COUNT ([DISTINCT] C) :** The number of (unique) values in the column C.

b. **SUM ([DISTINCT] C) :** The sum of all (unique) values in the column C.

c. **AVG ([DISTINCT] C) :** The average of all (unique) values in the column C.

d. **MIN (C) :** The minimum value in the column C.

e. **MAX (C) :** The maximum value in the column C.

3. Example

Consider the student table indicate assessment marks of every student. The blanks in mark indicates absent student in exam. Assessment is conducted out of 100 marks. If school management want to analyse performance of class using summarized data.

ID	Name	Marks	Department	Subject	Grant
1	Mahesh	90	IT	DB	5000
2	Suhas	60	CS	DB	
3	Jayendra	89	IT	DB	3000
4	Sachin	99	CS	DB	7000
5	Vishal	78	IT	DB	
6	Payal	90	MECH	CPP	3000
7	Kasturi		MECH	CPP	

a. COUNT ()

- This function is used to calculate number of rows (or records) in a table selected by query.
- Column in the query must be numeric.
- COUNT returns the number of rows in the table when the column value is not NULL.
- Count (*) will returns that number of unique tuples present in table.
- Find number of students in class.

```
MYSQL>  SELECT  count (*) Tot_Employees
        FROM    Student;
OR
MYSQL>  SELECT  count (ID) Tot_Employees
        FROM    Student;

+---------------+
| Tot_Employees |
+---------------+
|             7 |
+---------------+
1 row in set (0.00 sec)
```

- The aggregate data functions ignore all null value in column on which aggregate data function if applied.
- Find number of students appeared for exam.

```
MYSQL>  SELECT  count (Marks)
        FROM    Student;

+--------------+
| count(Marks) |
+--------------+
|            6 |
+--------------+
1 row in set (0.00 sec)
```

- The COUNT aggregate data function can be used with DISTINCT to ignore all repeated value in column on which aggregate data function if applied.
- Find number of departments of participated in exam.

```
MYSQL>  SELECT   count (DISTINCT Department)
        FROM     Student;

 count(DISTINCT Department) |
 ---------------------------
                  3
 ---------------------------
1 row in set (0.00 sec)
```

- The COUNT aggregate data function can be used with WHERE to select few rows in table on which aggregate data function if applied.
- Find number of departments of participated in exam with at least one student scored above 60.

```
MYSQL> SELECT  count(DISTINCT Department)
       FROM    Student
       WHERE   Marks>60;

+---------------------------+
| count(DISTINCT Department) |
+---------------------------+
|                  2 |
+---------------------------+
1 row in set (0.00 sec)
```

b. SUM ()

- This function is used to calculate sum of column values in a table selected by query.
- Column in the query must be numeric.
- Value of the sum must be within the range of that data type.

Find total of grants given by non IT departments.

```
MYSQL> SELECT  SUM(Grants)
       FROM    Student
       WHERE   Department <> 'IT';

 ------------
 SUM(Grants)
 ------------
 10000.00
 ------------
1 row in set (0.00 sec)
```

Find total of grants given by non IT departments to students scoring above 90.

```
MYSQL> SELECT  SUM(Grants)
       FROM    Student
       WHERE   Department <> 'IT'
       AND Marks > 90;

 ------------
 SUM(Grants)
 ------------
 7000.00
 ------------
1 row in set (0.00 sec)
```

c. AVG ()

- This function is used to calculate Average of column values in a table selected by query.
- This function first calculates sum of column and then divide by total number of rows.
- AVG returns the average of all the values in the specified column.
- Column in the query must be numeric.

Find average marks scored by non IT departments.

```
MYSQL> SELECT  AVG(Marks)
       FROM    Student
       WHERE   Department <> 'IT';

 ------------
 AVG(Marks)
 ------------
 83.0000
 ------------
1 row in set (0.00 sec)
```

Find average marks scored by non IT students in CPP subject.

```
MYSQL> SELECT  AVG(Marks)
       FROM    Student
       WHERE   Department <> 'IT'
       AND     Subject = 'CPP';

 ------------
 AVG(Marks)
 ------------
```

```
   90.0000
-----------
1 row in set (0.01 sec)
```

d. MIN ()

- This function is used to find minimum value out of column values in a table selected by query.
- Column in the query need not be numeric data type.

Find minimum marks scored by non IT departments.

```
MYSQL> SELECT    MIN(Marks)
       FROM      Student
       WHERE     Department <> 'IT';
------------
 MIN(Marks)
------------
      60
------------
1 row in set (0.00 sec)
```

Find minimum marks scored by non IT students in CPP subject.

```
MYSQL> SELECT    MIN(Marks)
       FROM      Student
       WHERE     Department <> 'IT'
       AND Subject = 'CPP';

------------
 MIN(Marks)
------------
      90
------------
1 row in set (0.00 sec)
```

e. MAX ()

- This function is used to find maximum value out of column values in a table selected by query.
- Column in the query need not be numeric data type.

Find maximum marks scored by non IT departments.

```
MYSQL> SELECT    MAX(Marks)
       FROM      Student
       WHERE     Department <> 'IT';
   ------------
   MAX(Marks)
   ------------
         99
   ------------
   1 row in set (0.00 sec)
```

Find maximum marks scored by non IT students in CPP subject.

```
MYSQL> SELECT    MAX(Marks)
       FROM      Student
       WHERE     Department <> 'IT'
       AND Subject = 'CPP';
------------
 MAX(Marks)
------------
      90
------------
1 row in set (0.00 sec)
```

6.5.1 Summary of Aggregate Functions

Q. Compare various Aggregate functions.

Function	Description
AVG ([DISTINCT | ALL] n)	Average value of n, ignoring null values.
COUNT ({ * | [DISTINCT | ALL] expr})	Number of rows, where *expr* evaluates to something other than null.
MAX ([DISTINCT | ALL] expr)	Maximum value of *expr*, ignoring null values.
MIN ([DISTINCT | ALL] expr)	Minimum value of *expr*, ignoring null values.
SUM ([DISTINCT | ALL] n)	Sum value of n, ignoring null values.

6.6 GROUP BY Clause - Grouping Query Results

Q. Explain the term : Group by clause.

MU - May 16, 2 Marks

1. Introduction

- The SQL GROUP BY clause is used in with the SQL SELECT statement to arrange data into groups.

- Related rows can be grouped together by **GROUP BY** clause based on distinct values that exist for specified grouping columns.
- The GROUP BY statement is used with the SQL aggregate functions to group the data using one or more columns or expression.
- The GROUP BY column does not have to be in the SELECT clause.
- A Grouping Query groups rows based on common values in a set of grouping columns. Rows with the same values for the grouping columns are placed in distinct groups. Each group is treated as a single row in the query result.

2 Syntax

```
SELECT      Column_name
FROM        Table_name
WHERE       [Condition]
GROUP BY    Column_names
ORDER BY [Columns];
```

3. Example

- Consider the student table indicate assessment marks of every student. The blanks in mark indicates absent student in exam. Assessment is conducted out of 100 marks.
- If school management want to analyse performance of class using summarized data.

ID	Name	Marks	Department	Subject	Grant
1	Mahesh	90	IT	DB	5000
2	Suhas	60	CS	DB	
3	Jayendra	89	IT	DB	3000
4	Sachin	99	CS	DB	7000
5	Vishal	78	IT	DB	
6	Payal	90	MECH	CPP	3000
7	Kasturi		MECH	CPP	

Retrieve all departments in college.

```
MySQL> SELECT      Department
       FROM        Student
       GROUP BY    Department;

------------------
Department
------------------
CS
IT
MECH
------------------
3 rows in set (0.00 sec)
```

Retrieve all departments offers DB subject.

```
MySQL> SELECT      Department
       FROM        Student
       WHERE       subject= 'DB'
       GROUP BY    Department;

------------------
Department
------------------
CS
IT
------------------
2 rows in set (0.00 sec)
```

Retrieve all different departments offers DB subject. (Sort in reverse alphabetical order)

```
MySQL> SELECT      Department
       FROM        Student
       WHERE       subject= 'DB'
       GROUP BY    Department
       ORDER BY    Department;

------------------
Department
------------------
IT
CS
------------------
2 rows in set (0.00 sec)
```

6.7 HAVING Clause – Apply Condition on groups

1. WHERE clause

- A Grouping Query can also group rows based on common values in a grouping columns.

- To select few rows with specified condition before grouping WHERE clause can be specified. If WHERE clause is not used in query then all rows in table are retrieved by default.
- WHERE clause defines a row qualifier condition. It means a condition row must satisfy to be a part of result set.
- Retrieve all departments offers DB subject.

```
MySQL> SELECT      Department
       FROM        Student
       WHERE       subject= 'DB'
       GROUP BY    Department;

------------------
Department
------------------
CS
IT
------------------
2 rows in set (0.00 sec)
```

2. HAVING clause

- A Grouping Query will group table rows based on common values in a grouping columns after that if we need to select few groups meeting certain conditions HAVING clause can be used.
- To select few groups with specified condition after grouping HAVING clause can be specified. If it is not used in query then all groups will be printed by default.
- A HAVING clause is like a WHERE clause, but applicable only to groups as a whole (that is, to the rows in the result set representing groups), whereas the WHERE clause applies to individual rows.
- A query can contain both a WHERE clause and a HAVING clause.

Retrieve departments offered total grants above 5000 for DB Subject.

```
MySQL> SELECT      Department
       FROM        Student
       WHERE       subject= 'DB'
       GROUP BY    Department
       HAVING      SUM(Grants) > 5000;
------------------
Department
------------------
CS
------------------
1 rows in set (0.00 sec)
```

Difference in operation of WHERE and GROUP BY clause

- The WHERE clause is applied first to the individual rows in the result set then rows that meet the conditions in the WHERE clause are grouped using group by clause.
- The HAVING clause is then applied to the rows in the result set that is produced by grouping.
- Only the groups that meet the HAVING conditions appear in the query output.
- It is possible to apply a HAVING clause only to columns that also appear in the GROUP BY clause or in z aggregate function. Retrieve departments with name start from letter 'C' and offers DB subject.

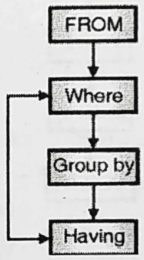


Fig. 6.7.1

```
MySQL> SELECT      Department
       FROM        Student
       WHERE       subject= 'DB'
       GROUP BY    Department
       HAVING      Department LIKE 'C%';
------------------
Department
------------------
CS
------------------
1 rows in set (0.00 sec)
```

3. Single row query Evaluation

1. In above given query the first executable statement is **FROM**, this statement will load the all rows in STUDENT table on to server to local memory.

2.. **WHERE** clause will select student rows with subject 'DB' and other remaining rows are deleted from memory.

3. **GROUP BY clause** will form groups of entire table using Department column then we will get number of groups equals to total number of departments.

4. **HAVING clause** will select few department groups with help of condition Department name start with letter C and remaining groups are deleted from memory.

5. **ORDER BY** Clause will arrange output in proper order that can be either ascending or descending order.

6. At last **SELECT** Clause will either all (*) or specified columns will be printed on client computer.

Fig. 6.7.2

6.8 GROUP BY and Aggregate Function

1. Introduction

- **GROUP BY** clauses is very useful in combination with aggregate functions for generating management summary report which is more helpful for taking decisions.
- Generally, aggregate functions will return a single result for a table.
- **GROUP BY** statement can return single function value for each individual group of column values.

2. Example

Retrieve maximum grant amount offered by any department.

```
MySQL> SELECT  Max(Grants)
       FROM    Student;
---------------
Max(Grant)
---------------
7000
---------------
1 rows in set (0.00 sec)
```

Retrieve maximum grant amount offered by each department.

```
MySQL> SELECT  Department, Max(Grants)
       FROM    Student
       GROUP BY  Department;
----------------------
Department    Max(Grants)
----------    -----------
CS            7000.00
IT            5000.00
MECH          3000.00
----------    -----------
3 rows in set (0.00 sec)
```

6.9 Pattern Matching - String Operations

Q. Where to use LIKE clause ? Give its syntax.

1. Introduction

- Pattern matching statements are used to find out particular string in attribute value.
- LIKE is keyword that is used in the WHERE clause for pattern matching.
- Determines whether a specific character string matches a specified pattern.
- SQL provides two wildcard characters for use with LIKE.

Percentage sign ('%')

- Percentage sign means any string of zero or more characters.
- Percent sign (%) is used to define the start and ending or your string.

Underscore sign ('_')

- Under Sore sign means a single character.
- Under Sore sign is used to determine letter at particular position.

2. Syntax

```
SELECT    Column_name
FROM      Table_name
WHERE     column_name LIKE {PATTERN}
```

{PATTERN} often consists of wildcards.

3. Example

Consider the student table indicate assessment marks of every student. The blanks in mark indicates absent student in exam. Assessment is conducted out of 100 marks. If school management want to analyse performance of class using summarized data.

ID	Name	Marks	Department	Subject	Grant
1	Mahesh	90	IT	DB	5000
2	Suhas	60	CS	DB	
3	Jayendra	89	IT	DB	3000
4	Sachin	99	CS	DB	7000
5	Vishal	78	IT	DB	
6	Payal	90	MECH	CPP	3000
7	Kasturi		MECH	CPP	

Pattern 1 : Determine a string starts with a given letter.

Find all students whose name starts with 'S'.

```
MySQL> SELECT *
       FROM Student
       WHERE Name LIKE 'S%';

ID  Name   Marks  Department  Subject  Grants
--- ------ ------ ----------- -------- -------
2   Suhas  60     CS          DB       NULL
4   Sachin 99     CS          DB       7000.00

2 rows in set (0.00 sec)
```

Pattern 2 : Determine a string starts with a given string.

Find all students whose name starts with 'Om'.

```
Mysql> SELECT *
       FROM Student
       WHERE Name LIKE 'Pay%';

ID  Name  Marks  Department  Subject  Grants
--- ----- ------ ----------- -------- -------
6   Payal 90     MECH        CPP      3000.00

1 row in set (0.00 sec)
```

Pattern 3 : Determine a string ends with a given letter.

Find all students whose name ends with 'h'.

```
Mysql> SELECT *
       FROM Student
       WHERE Name LIKE '%h';

ID Name   Marks  Department  Subject  Grants
-- ------ ------ ----------- -------- --------
1  Mahesh 90     IT          DB       5000.00

1 row in set (0.00 sec)
```

Pattern 4 : Determine a string contains given a letter.

Find all students whose name contains letter 'e'.

```
MySQL> SELECT *
       FROM Student
       WHERE Name LIKE '%e%';

ID  Name   Marks  Department  Subject  Grants
--- ------ ------ ----------- -------- -------
1   Mahesh 90     IT          DB       5000.00
3   Jayendra 89   IT          DB       3000.00

2 rows in set (0.00 sec)
```

Pattern 5 : Determine a string in which given pattern is combination of above forms.

Find all students whose names first letter is 'm' contains letter 'e' and ending with letter 'h'.

```
MySQL> SELECT *
       FROM Student
       WHERE Name LIKE 'M%e%h';

ID  Name   Marks  Department  Subject  Grants
--- ------ ------ ----------- -------- -------
1   Mahesh 90     IT          DB       5000.00

1 row in set (0.00 sec)
```

Pattern 6 : Determine a String in which given a letter is present at particular position.

Find all students whose names second letter is 'a'.

```
MySQL> SELECT *
       FROM Student
       WHERE Name LIKE '_a%';
```

ID	Name	Marks	Department	Subject	Grants
1	Mahesh	90	IT	DB	5000.00
3	Jayendra	89	IT	DB	3000.00
4	Sachin	99	CS	DB	7000.00
6	Payal	90	MECH	CPP	000.00
7	Kasturi	NULL	MECH	CPP	NULL

5 rows in set (0.00 sec)

Find all students whose names second last letter is 's'.

```
MySQL> SELECT *
       FROM Student
       WHERE Name LIKE '%s_';
```

ID	Name	Marks	Department	Subject	Grants
1	Mahesh	90	IT	DB	5000.00

1 row in set (0.00 sec)

Pattern 7 : Determine a string not contains given a letter.

Find all students whose name do not contains 's'.

```
MySQL> SELECT *
       FROM Student
       WHERE Name NOT LIKE '%s%';
```

ID	Name	Marks	Department	Subject	Grants
3	Jayendra	89	IT	DB	3000.00
6	Payal	90	MECH	CPP	3000.00

2 rows in set (0.00 sec)

4. Use of escape character in pattern matching

- Escape characters can be used for some queries like if we want to look for '%' sign in a given column.
- Use the **escape** clause to specify an escape character in the **like** clause.
- An escape character must be a single character string.
- Any character in the server's default character set can be used. (_ , %)
- Specifying more than one escape character raises a SQLSTATE error condition

```
LIKE '%XX_%' ESCAPE 'XX'
```

Example

Consider student table as given below,

```
MySQL> SELECT *
       FROM student;
```

ID	Name	Marks	Department	Subject	Grants	email
1	Mahesh	90	IT	DB	5000.00	a_bh@myc.edu
2	Suhas	60	CS	DB	NULL	uh@myc.edu
3	Jayendra	89	IT	DB	3000.00	a_by@myc.edu
4	Sachin	99	CS	DB	7000.00	a_bc@myc.edu
5	Vishal	78	IT	DB	NULL	is@myc.edu
6	Payal	90	MECH	CPP	3000.00	a_by@myc.edu
7	Kasturi	NULL	MECH	CPP	NULL	a_bs@myc.edu

7 rows in set (0.00 sec)

Find all students whose email id does not contains underscore.

```
MySQL> SELECT *
       FROM Student
       WHERE Email NOT LIKE '%_%';

Empty set (0.00 sec)
```

The above query is not working as if finds all students with email ids with length not more than one letter.

Find all students whose email id does not contains underscore.

```
MySQL> SELECT *
       FROM Student
       WHERE Email NOT LIKE '%@_%'
       ESCAPE '@';

ID  Name  Marks  Dept.  Subject  Grants  email
--- ----- -------- ------ -------- -------- ----------
2   Suhas 60     CS     DB       NULL    uh@myc.edu
5   Vishal 78    IT     DB       NULL    is@myc.edu

2 rows in set (0.00 sec)
```

6.10 SET Operations

Q. What do you mean by set operation ?

1. Introduction

- SQL SET operators allow combining results from two or more SELECT statements or combines result set of multiple queries.
- The results of two queries can be combined using the set operations union, intersection and difference.

```
Query_1 UNION     [ALL] Query_2
Query_1 INTERSECT [ALL] Query_2
Query_1 EXCEPT    [ALL] Query_2
```

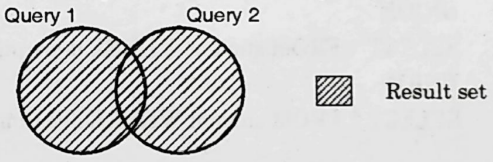
2. Requirements

- SELECT statement of both queries must retrieve the same number of columns.
- Data types of corresponding columns in each query must be of same type.

6.10.1 Union Operator

a. Overview

- Union effectively appends the result of first query to the result of second query.
- It does not eliminate all duplicate rows and they are printed in result expression.
- **DISTINCT :** Duplicate row shown only once.
- **ALL :** Duplicate row shown only once.



Find all students in all departments.

```
MySQL> SELECT *
       FROM student
       WHERE department='IT';

ID    Name     Marks  Department  Subject  Grants   email
------ ---------- ------- ---------- --------- --------- -------------
1     Mahesh   90     IT          DB       5000.00  a_bh@myc.edu
3     Jayendra 89     IT          DB       3000.00  a_by@myc.edu
5     Vishal   78     IT          DB       NULL     is@myc.edu

3 rows in set (0.00 sec)

MySQL>  SELECT *
```

```
        FROM student
        WHERE department='CS';
```

ID	Name	Marks	Department	Subject	Grants	email
2	Suhas	60	CS	DB	NULL	uh@myc.edu
4	Sachin	99	CS	DB	7000.00	a_bc@myc.edu

2 rows in set (0.00 sec)]

```
MySQL> SELECT *
        FROM student
        WHERE department='MECH';
```

ID	Name	Marks	Department	Subject	Grants	email
6	Payal	90	MECH	CPP	3000.00	a_by@myc.edu
7	Kasturi	NULL	MECH	CPP	NULL	a_bs@myc.edu

2 rows in set (0.00 sec)

```
/* To Combine Result of above 3 Queries using UNION Operation */

MySQL> SELECT * FROM student WHERE department='IT'
        UNION
        SELECT * FROM student WHERE department='CS'
        UNION
        SELECT * FROM student WHERE department='MECH';
```

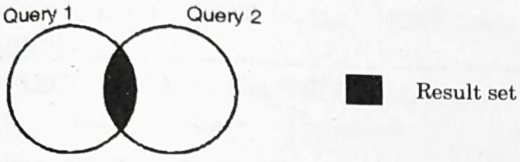
ID	Name	Marks	Department	Subject	Grants	email
1	Mahesh	90	IT	DB	5000.00	a_bh@myc.edu
3	Jayendra	89	IT	DB	3000.00	a_by@myc.edu
5	Vishal	78	IT	DB	NULL	is@myc.edu
2	Suhas	60	CS	DB	NULL	uh@myc.edu
4	Sachin	99	CS	DB	7000.00	a_bc@myc.edu
6	Payal	90	MECH	CPP	3000.00	a_by@myc.edu
7	Kasturi	NULL	MECH	CPP	NULL	a_bs@myc.edu

7 rows in set (0.06 sec)

6.10.2 Intersect Operator

a. Overview

- This operator finds out all rows that are common in both result of Query 1 and in the result of Query 2.
- It does not eliminate all duplicate rows and they are printed in result expression.
- Example



Find all IT students offered grants above 2000.

```
/* Query 1: To print all IT students */
MySQL> SELECT *
       FROM Student
       WHERE Department='IT';
```

ID	Name	Marks	Department	Subject	Grants	email
1	Mahesh	90	IT	DB	5000.00	a_bh@myc.edu
3	Jayendra	89	IT	DB	3000.00	a_by@myc.edu
5	Vishal	78	IT	DB	NULL	is@myc.edu

3 rows in set (0.00 sec)

```
/* Query 2: To print all students offered grants above 2000 */

MySQL> SELECT *
       FROM Student
       WHERE Grants>=2000;
```

ID	Name	Marks	Department	Subject	Grants	email
1	Mahesh	90	IT	DB	5000.00	a_bh@myc.edu
3	Jayendra	89	IT	DB	3000.00	a_by@myc.edu
4	Sachin	99	CS	DB	7000.00	a_bc@myc.edu
6	Payal	90	MECH	CPP	3000.00	a_by@myc.edu

4 rows in set (0.00 sec)

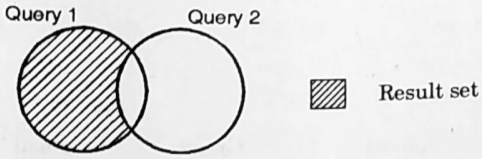
```
/* Final Query: To print all IT students offered grants above 2000 */

MySQL> SELECT * FROM Student WHERE Department='IT'
       INTERSECT
```

```
SELECT * FROM Student WHERE Grants>=2000;

ID     Name      Marks   Department   Subject    Grants     email
------ --------- ------- ------------ ---------  --------   -------------
1      Mahesh    90      IT           DB         5000.00    a_bh@myc.edu
3      Jayendra  89      IT           DB         3000.00    a_by@myc.edu

2 rows in set (0.00 sec)
```

6.10.3 Difference Operator

a. Overview

- Returns all rows that are in the result of Query 1 but not in the result of Query 2.
- Again, duplicates are eliminated unless **ALL** is specified.



Find all non IT students offered grants above 2000.

```
/* Query 1: To print all students offered grants above 2000 */

MySQL> SELECT *
       FROM Student
       WHERE Grants>=2000;

ID     Name      Marks   Department   Subject    Grants     email
------ --------- ------- ------------ ---------  --------   -------------
1      Mahesh    90      IT           DB         5000.00    a_bh@myc.edu
3      Jayendra  89      IT           DB         3000.00    a_by@myc.edu
4      Sachin    99      CS           DB         7000.00    a_bc@myc.edu
6      Payal     90      MECH         CPP        3000.00    a_by@myc.edu

4 rows in set (0.00 sec)

/* Query 2 : To print all IT students */

MySQL> SELECT *
       FROM Student
       WHERE Department='IT';

ID     Name      Marks   Department   Subject    Grants     email
------ --------- ------- ------------ ---------  --------   -------------
1      Mahesh    90      IT           DB         5000.00    a_bh@myc.edu
```

3	Jayendra	89	IT	DB	3000.00	a_by@myc.edu
5	Vishal	78	IT	DB	NULL	is@myc.edu

3 rows in set (0.00 sec)

/* Final Query : To print all IT students offered grants above 2000 */

```
MySQL> SELECT * FROM Student WHERE Grants>=2000
    MINUS
    SELECT * FROM Student WHERE Department='IT';
```

ID	Name	Marks	Department	Subject	Grants	email
4	Sachin	99	CS	DB	7000.00	a_bc@myc.edu
6	Payal	90	MECH	CPP	3000.00	a_by@myc.edu

2 rows in set (0.00 sec)

6.11 NULL Values

1. Overview

- Concept of NULL values is introduced by father of relational database E. F. Codd.
- NULL values used to represent missing information and inapplicable information in RDBMS.
- The NULL value represents unknown, unassigned or empty values in any column of table in database.
- A value of NULL is different from an empty, White spaces (blank spaces) or zero value.
- No two NULL values are equal as both are unknown so they may not be equal.
- Null is not having any data type in SQL, meaning that it is not designated as an integer, character, or any other specific data type.
- NULL values require special handling in SQL queries.

2. Null as Third Logic (3 valued logic)

- Since Null is not a member of any columns in a table, it is not considered a value it is considered as placeholder indicating the absence of value.
- Because of this, comparisons with Null can never result in either True or False, but always in a third logical result, Unknown.
- Not Logic table
- As Null values do mean by "unknown" hence negation will produce same results,

P	Not P (~P)
True	False
False	True
Unknown	Unknown

- Complete Logic table

Comparisons between two null values, or between a NULL and any other value, return unknown because the value of each NULL is unknown.

P	Q	P OR Q	P AND Q	P = Q
True	True	True	True	True
True	False	True	False	False
True	Unknown	True	Unknown	Unknown
False	True	True	False	False
False	False	False	False	True
False	Unknown	Unknown	False	Unknown
Unknown	True	True	Unknown	Unknown
Unknown	False	Unknown	False	Unknown
Unknown	Unknown	Unknown	Unknown	Unknown

3. Locating Missing Values in table

Consider the student table indicate assessment marks of every student. The blanks in mark indicates absent student in exam. Assessment is conducted out of 100 marks.

```
MySQL> SELECT *
       FROM Student;
```

ID	Name	Marks	Department	Subject	Grants	email
1	Mahesh	90	IT	DB	5000.00	a_bh@myc.edu
2	Suhas	60	CS	DB	NULL	uh@myc.edu
3	Jayendra	89	IT	DB	3000.00	a_by@myc.edu
4	Sachin	99	CS	DB	7000.00	a_bc@myc.edu
5	Vishal	78	IT	DB	NULL	is@myc.edu
6	Payal	90	MECH	CPP	3000.00	a_by@myc.edu
7	Kasturi	NULL	MECH	CPP	NULL	a_bs@myc.edu

7 rows in set (0.03 sec)

Find details of all students absent for any exam.

```
MySQL> SELECT *
       FROM Student
       WHERE Marks = NULL;
Empty set (0.00 sec)
```

Above query will not retrieve any rows since NULL means missing value cannot be equal to anything.

The condition for selecting NULL values can be as follows;

Find details of all students absent for any exam.

```
MySQL> SELECT *
       FROM Student
       WHERE Marks IS NULL;
```

ID	Name	Marks	Department	Subject	Grants	email
7	Kasturi	NULL	MECH	CPP	NULL	a_bs@myc.edu

1 row in set (0.03 sec)

Find details of all students present for exam.

```
MySQL> SELECT *
       FROM Student
       WHERE Marks IS NOT NULL;
```

ID	Name	Marks	Department	Subject	Grants	email
1	Mahesh	90	IT	DB	5000.00	a_bh@myc.edu
2	Suhas	60	CS	DB	NULL	uh@myc.edu
3	Jayendra	89	IT	DB	3000.00	a_by@myc.edu
4	Sachin	99	CS	DB	7000.00	a_bc@myc.edu
5	Vishal	78	IT	DB	NULL	is@myc.edu
6	Payal	90	MECH	CPP	3000.00	a_by@myc.edu

6 rows in set (0.00 sec)

4. NULL value functions

These functions are specially designed for operating NULL value.

a. NVL()

- This function is provided by oracle and it accepts two parameters.
- If the first parameter of expression is equal to NULL, then, the value of the second parameter is returned else same parameter (parameter 1) is returned.

```
NVL (parameter 1, parameter 2);
```

Example

```
Oracle> SELECT ID, NAME, NVL(Marks, 'Absent')
        FROM Student;
```

ID	Name	Marks
1	Mahesh	90
2	Suhas	60
3	Jayendra	89
4	Sachin	99
5	Vishal	78
6	Payal	90
7	Kasturi	Absent

7 rows in set (0.00 sec)

b. ISNULL()

- This function accepts two parameters and supported by MySQL.
- If parameter of expression is equal to NULL, then, 0 will be returned or else 1 will be the output.

```
ISNULL (parameter 1);
```

Example

```
MySQL> SELECT ID, NAME, ISNULL(Marks)
       FROM Student;
```

ID	Name	Marks
1	Mahesh	0
2	Suhas	0
3	Jayendra	0
4	Sachin	0
5	Vishal	0
6	Payal	0
7	Kasturi	1

7 rows in set (0.00 sec)

6.12 Solved Designing Problem

Example 6.12.1 : Consider the following relations for database that keeps track of student enrolment in courses and books issued for each course.

STUDENT (Ssn, Name, Subject, DOB)

COURCE (Course_id, Name, Dept)

ENROLL (Ssn, Course_id, Semester, Grade)

BOOK_ISSUED (Course_id, Semester, ISBN)

TEXT (ISBN, Title, Publisher, Author)

Solution : Write a Query to select all courses available in institute.

```
MySQL> SELECT *
       FROM COURSE
```

Example 6.12.2 : For the given database, write SQL queries :

employee (eid, employee_name, street, city)

Works (eid, cid, salary)

Company (cid, company_name, city)

Manager (eid, manager_name)

Solution :

(i) Find the names of all employees having "i" as the second letter in their names.

```
MySQL> SELECT Employee_name, street, city
       FROM Employee
       WHERE Employee_name like '_i%';
```

(ii) Display the annual salary of all employees.

```
MySQL> SELECT Eid, (salary)*12 [annualsalary]
       FROM Works;
```

Example 6.12.3 : Consider the following relations for a book club :

Members (Member-Id, Name, Designation, Age)

Books (Book-Id, Booktitle, BookAuthor, Bookpublisher, Bookprice)

Reserves (Member-Id, Book-Id, Date)

MU - May 14, 10 Marks

Solution :

Find the names of members who are professor older than 50 years.

```
MySQL> SELECT  Name
       FROM    Members
       WHERE   Designation LIKE 'Professor'
       AND     Age>50;
```

Example 6.12.4 : Consider the following employee database.

Employee (empname, street, city, date_of_joining)

Works (empname, company_name, salary)

Company (company_name, city)

Manages (empname, manager_name)

Write SQL queries for the following statements:

1. Modify the database so that employee "Amruta" now leaves in "Konkan"
2. Find number of employees in each city with date_of_Joining as "01-Aug-2017"
3. List name of companies starting with letter "A"
4. Display empname, manager_name, street, city only for employees having manager

MU - Dec.17, 10 Marks

Soln. :

1. Modify the database so that employee "Amruta" now leaves in "Konkan".

```
UPDATE employee
SET city = 'Kokan'
WHERE empname = 'Amruta';
```

2. Find number of employees in each city with date_of_Joining as "01-Aug-2017".

```
SELECT   COUNT(*)
FROM     Employee
WHERE    Date_of_joining=' 01-Aug-2017'
GROUP    BY city;
```

3. List name of companies starting with letter "A".

```
SELECT   Company_name
FROM     Company
WHERE    Company_Name LIKE 'A%';
```

4. Display empname, manager_name, street, city only for employees having manager.

```
SELECT   e.empname, m.manager_name e.street,
         e.city
FROM     Employee E
INNER    JOIN Manages M
ON       E.empname =M.empname
```

Example 6.12.5 : Consider the following employee database.

Employee(empname, street, city, date_of_joining)

Works(empname, company_name,salary)

Company(company_name,city)

Manages(empname, manager_name)

Write SQL queries for the following statements:

1. Modify the database so that employee "Sachin" now lives in "Mumbai"
2. Find number of employees in each city with date_of_joining as"01- Aug-2017"
3. List the name of companies starting with letter "A"
4. Displayempname, manager_name,cityofthoseemployeeswhose date_of_joining Is greater than "01-01-2014"

Ans.:

1. Modify the database so that employee "Sachin" now lives in "Mumbai"

```
UPDATE   employee
SET      city="Mumbai"
WHEN     empname="sachin"
```

2. Find number of employees in each city with date_of_joining as"01- Aug-2017"

```
SELECT   *
FROM     employee
WHERE    date_of_joining="01-Aug-2017"
```

3. List the name of companies starting with letter "A"

```
SELECT   *
FROM     company
WHERE    company_Name LIKE 'A%'
```

4. Displayempname, manager_name,cityofthoseemployeeswhose date_of_joining Is greater than "01-01-2014"

```
SELECT   e.empname, m.manager_name, e.city
FROM     Employee e, Manages m
WHERE    e.empname=m.empname
AND      e.date_of_joining> '2014-01-01';
```

Example 6.12.6 : Use the relational database schema and write the following queries.

1. Retrieve the birth date and address of the employee(s) whose name is 'Vaidehi Chavan'.
2. Retrieve the name and address of all employees who work for the 'Research' department.
3. For every project located in 'Mumbai', list the project number, the controlling department number, and the department manager's last name, address, and birth date.
4. Retrieve a list of employees and the projects they are working on, ordered by department and, with in each department, ordered alphabetically by last name, first name. **MU - Dec.18, 10 Marks**

Ans. :

1. Retrieve the birthdate and address of the employee(s) whose name is VaidehiChavan.

```
SELECT   bdate,address
FROM     employee
WHERE    fname='Vaidehi' and
         lname='Chavan';
```

2. Retrieve the name and address of all employees who work for the Research department.

```
SELECT   fname,lname,address
FROM     employee,department
WHERE    dnumber=dno and
DNAME    ='Research';
```

3. For every project located in Mumbai, list the project number, the controlling department number, and the department managers last name, address, and birthdate.

```
SELECT   pnumber,dnum,lname,address,bdate
FROM     project,department,employee
WHERE    dnum=dnumber and mgrssn=ssn and
         plocation='Mumbai';
```

4. Retrieve a list of employees and the projects they are working on, ordered by department and, within each department, ordered alphabetically by last name, first name.

```
SELECT   dname,lname,fname,pname
FROM     department,employee,works_on,project
WHERE    dnumber=dno and ssn=essn and
         pno=pnumber
ORDER    by dname,lname,fname;
```

Review Questions

1. Write a note on various aggregate function.
2. Explain the concept of NULL values.
3. What is NULL value ? How it is treated in query processing.
4. Write short note on : Aggregate Functions in SQL.
5. Write short notes on : Group by and order by clause in SQL.
6. Compare the WHERE clause and HAVING clause.
7. Explain the use of ORDER BY clause.

6.13 University Questions and Answers

May 2016

1. Explain Aggregate function with example. **(5 Marks)**
2. Explain the term : Group by clause **(2 Marks)**

Dec. 2016

3. Explain Aggregate function with example. **(5 Marks)**

Dec. 2017

4. Consider the following employee database. **(10 Marks)**
   Employee (empname, street, city, date_of_joining)
   Works (empname, company_name, salary)
   Company (company_name, city)
   Manages (empname, manager_name)
   Write SQL queries for the following statements :
   1. Modify the database so that employee "Amruta" now leaves in "Konkan"
   2. Find number of employees in each city with date_of_Joining as "01-Aug-2017"
   3. List name of companies starting with letter "A"
   4. Display empname, manager_name, street, city only for employees having manager

May 2018

5. Consider the following employee database.
   Employee(empname, street, city, date_of_joining)
   Works(empname, company_name,salary)
   Company(company_name,city)

Manages(empname, manager_name)

Write SQL queries for the following statements :

1. Modify the database so that employee "Sachin" now lives in "Mumbai"
2. Find number of employees in each city with date_of_joining as"01- Aug-2017"
3. List the name of companies starting with letter "A"
4. Displayempname, manager_name,cityofthoseemployeeswhose date_of_joining Is greater than "01-01-2014" **(10 Marks)**

Dec. 2018

6. List aggregate functions and justify the need of any two aggregate functions. **(5 Marks)**
7. Use the relational database schema of Q. 4(a) and write the following queries. **(10 Marks)**
   1. Retrieve the birth date and address of the employee(s) whose name is 'Vaidehi Chavan'.
   2. Retrieve the name and address of all employees who work for the 'Research' department.
   3. For every project located in 'Mumbai', list the project number, the controlling department number, and the department manager's last name, address, and birth date.
   4. Retrieve a list of employees and the projects they are working on, ordered by department and, with in each department, ordered alphabetically by last name, first name.

May 2019

8. What are the different aggregate functions used in SQL? Explain with the help of example. **(5 Marks)**

Dec. 2019

9. Write short note on : Aggregate function in SQL. **(5 Marks)**

❑❑❑