Path Wala

14

Grouping Records, Joins in SQL

In This Chapter

14.1 Introduction
14.2 Types of SQL Functions
14.3 Grouping Result – GROUP BY
14.4 Joins

14.1 INTRODUCTION

You have worked with basic SQL commands in your previous class and used database MySQL. This chapter will talk about some SQL commands in details. After this chapter, you shall be able to display groups of records, summary of groups of records and display selected groups of records.

14.2 TYPES OF SQL FUNCTIONS

Now that you have learnt different types of functions, let us talk about their broad categories. SQL supports many and many functions. All these functions can be generally categorized into following *two* types :

- Single Row (or Scalar) functions.
- Multiple Row (or Group or Aggregate) functions.

(*i*) ***Single Row functions*** work with a single row at a time. A single row function returns a result for every row of a queried table.

(*ii*) ***Multiple Row or Group functions*** work with data of multiple rows at a time and return aggregated value.

Examples of multiple row functions are the group functions that you have learnt in previous section *i.e.*, sum( ), count( ), max( ), min( ), Avg( ) etc.

Path Wala

The difference between these two types of functions is in the number of rows they act upon. A **single row function** works with the data of a single row at a time and returns a single result for each row queried upon ; a **multiple row function** works with a group of rows and returns a single result for that group.

14.3 GROUPING RESULT – GROUP BY

The GROUP BY clause combines all those records that have identical values in a particular field or a group of fields. This grouping results into one summary record per group if group-functions are used with it. In other words, the GROUP BY clause is used in SELECT statements to divide the table into groups. Grouping can be done by a column name, or with aggregate functions in which case the aggregate produces a value for each group.

For example, to calculate the *number of employees in each grade*, you use the command

```
SELECT job, COUNT(*)
FROM empl
GROUP BY job ;
```

GROUP BY applies the aggregate functions independently to a series of groups that are defined by having a field value in common.

```
+------------+----------+
| job        | COUNT(*) |
+------------+----------+
| ANALYST    |    2     |
| CLERK      |    4     |
| MANAGER    |    3     |
| PRESIDENT  |    1     |
| SALESMAN   |    4     |
+------------+----------+
```

Now consider the following query, which is also grouping records based on *deptno*.

```
mysql> SELECT deptno, COUNT(*), SUM(sal)
       FROM empl
       GROUP BY deptno ;
```

```
+----------+----------+-----------+
| deptno   | COUNT(*) | SUM(sal)  |
+----------+----------+-----------+
|   10     |    3     | 8750.00   |
|   20     |    5     | 10885.00  |
|   30     |    6     | 9400.00   |
+----------+----------+-----------+
```

As you can make out that the above query is displaying count of records and sum of salaries in each group and the groups are formed on the basis of *deptno*. Thus, from the above output, you can make out that in department number 10, there are 3 employees (records) and total of all salaries is 8750.00 ; in department number 20, there are 5 employees and total of salaries is 10885.00 ; and so on.

14.3.1 Nested Groups – Grouping on Multiple Columns

With GROUP BY clause, you can create groups within groups. Such type of grouping is called ***Nested grouping***. This can be done by specifying in GROUP BY expression, where the *first field* determines the *highest group level*, the *second field* determines the *second group level*, and so on. The *last field* determines the *lowest level of grouping*.

In order to fully understand this concept, consider **Table 12.1 empl, Chapter 12**.

Path Wala

See there are multiple records having same value for field **Deptno**, we can group records on the basis of field *Deptno*. For instance, if you want to count the number of employees in each group, you need to issue a query statement as given below :

```
SELECT COUNT(empno) FROM empl
GROUP BY Deptno ;
```

And the result produced by this query is

```
+-------------+
|count(empno) |
+-------------+
|     3       |
|     5       |
|     6       |
+-------------+
```

But can you make out, these are employee-counts for which departments ? To get this information, you may modify the SELECT list as :

```
SELECT Deptno, COUNT(empno)
FROM empl
GROUP BY Deptno ;
```

Now the result will be :

```
+---------+-------------+
| deptno  | count(empno) |
+---------+-------------+
| 10      |     3       |
| 20      |     5       |
| 30      |     6       |
+---------+-------------+
```

See, now it is more clear. But one thing that you should keep in mind is that while grouping, you should include only those values in the *select list* that either have the same value for a group or contain a group (aggregate) function *i.e.*, a **group-expression**. Like in the above query, the first expression **Deptno** field has one (same) value for a group and the other expression **COUNT(empno)** contains a group function.

MySQL as such would not create any error even if you include a **non-group expression** in the select-list. A **non-group field** (or expression) is the field that has different values in the rows belonging to the group.

In this case, it will return **the value from first record** of the group for that non-group field *e.g.*, if you issue command like :

```
SELECT deptno, count(empno), mgr
FROM empl
GROUP BY deptno ;
```

This is non-group field as it has multiple values for a group.

The output returned will be :

```
+---------+-------------+------+
| deptno  | count(empno) | mgr  |
+---------+-------------+------+
| 10      |     3       | NULL |
| 20      |     5       | 8902 |
| 30      |     6       | 8698 |
+---------+-------------+------+
```

See, first record of group with deptno 10 has value NULL in mgr field ; first record of group with deptno 20 has 8902 in mgr field ; and first record of group with deptno 30 has value 8698 in mgr field. Hence this output.

To create a group within a group *i.e.*, nested group, you need to specify multiple fields in the GROUP BY expression. If you have a look at the records of *Empl* table, you can make out that there exists a *group of jobs* within the *department group* as there are same values for **Job** field, in one department group's records.

NOTE

In the select list of a group, only those fields or expressions state be included that either return single value for a group or are constants. Otherwise you may not get authentic results.

Path Wala

To group records *job wise* within *Deptno* wise, you need to issue a query statement like :

```
SELECT Deptno, Job, COUNT(empno)
FROM empl
GROUP BY Deptno, Job ;
```

And the result produced is :

deptno	job	count(empno)
10	CLERK	1
10	MANAGER	1
10	PRESIDENT	1
20	ANALYST	2
20	CLERK	2
20	MANAGER	1
30	CLERK	1
30	MANAGER	1
30	SALESMAN	4

14.3.2 Placing Conditions on Groups – HAVING Clause

The HAVING clause places conditions on groups in contrast to WHERE clause that places conditions on individual rows. While WHERE conditions cannot include aggregate functions, HAVING conditions can do so.

For example, to calculate the average gross and total gross for employees belonging to 'E4' grade, the command would be :

```
SELECT AVG(gross), SUM(gross)
FROM employee
GROUP BY grade
HAVING grade = 'E4' ;
```

This condition would be applicable on group and not on individual rows

To display the jobs where the number of employees is less than 3, you use the command :

```
SELECT JOB, COUNT(*)
FROM empl
GROUP BY job
HAVING count(*) < 3 ;
```

This will produce the following output :

JOB	COUNT(*)
ANALYST	2
PRESIDENT	1

2 rows in set (0.11 sec)

Path Wala

The HAVING clause can contain either a simple boolean expression (*i.e.*, an expression or condition that results into *true* or *false*) or use aggregate function in the having condition.

You can include more than one condition in HAVING clause, of course, by using logical operators. Consider this :

```
SELECT Deptno, AVG(Comm), AVG(Sal)
FROM empl
GROUP BY Deptno
HAVING AVG(Comm) > 750 AND
AVG(Sal) > 2000 ;
```

Also, you can use an aggregate function in the HAVING clause even if it is not in the SELECT list. Consider the following query to understand this :

```
SELECT Deptno, AVG(Sal)
FROM empl
GROUP BY Deptno
HAVING COUNT(*) <= 3 ;
```

You can also use IN or BETWEEN operators with HAVING clause. Following two queries illustrate this :

```
SELECT Deptno, Job, AVG(Sal)
FROM empl
GROUP BY Deptno, Job
HAVING JOB IN ('CLERK', 'SALESMAN') ;
SELECT Deptno, Job, SUM(sal)
FROM empl
GROUP BY Deptno, Job
HAVING SUM(sal) BETWEEN 3000 AND 7000 ;
```

14.3.3 Non-Group Expressions with GROUP BY

As mentioned before, if you include a non-group expression in the select-list of a query with GROUP BY, MySQL will not produce any error. Rather it will pick value of the specified non-group field from the first row of the group. But we do not recommend this practice because it will produce ambiguous results. For instance, consider the following query and its output.

```
mysql>  SELECT ename, sum(sal)
        FROM empl
        GROUP BY deptno ;
```

```
+--------+----------+
| ename  | sum(sal) |
+--------+----------+
| AMIR   |  8750.00 |
| SMITH  | 10885.00 |
| ANYA   |  9400.00 |
+--------+----------+
```

See, isn't it conveying that AMIR's salary-sum is 8750.00, SMITH'S 10885.00 and ANYA'S 9400.00 ?

Thus, we recommend not to use non-group expressions in GROUP BY query unless otherwise necessary.

Path Wala

14.4 JOINS

A **join** is a query that combines rows from two or more tables[1]. In a *join-query*, more than one table are listed in FROM clause. The function of combining data from multiple tables is called *joining*.

SQL can produce data from several related tables by performing either a physical or virtual join of the tables.

The WHERE clause is most often used to perform the JOIN function where two or more tables have common columns.

Consider the following example :

```
SELECT patient_no, description, normal_charge, charge
FROM billed, item
WHERE billed.item_code = item.item_code ;
```

tables being joined

join condition

The above query is joining two tables *billed* and *item* (*see their structures on the left*) by equating their *item_codes*. The select_list consists of fields coming from both the tables. When two or more tables have a column with the *same name*, the name can be *qualified* by using the table name combined with the period (.) in referring to the column, *e.g., billed.patient_no*. Note that columns with unique names *do not* have to be qualified by the table name.

Table Structures

A. Billed
- Bill_No
- Patient_No
- Item_Code
- Charge

B. Item
- Item_Code
- Description
- Normal_Charge

14.4.1 Cartesian Product

Consider the following query :

```
SELECT *
FROM EMP, DEPT ;
```

This query will give you *the Cartesian product i.e.*, all possible concatenations are formed of all rows of both the tables EMP and DEPT. That is, when no particular rows (using WHERE clause) and columns (through SELECT list) are selected. Such an operation is also known as **Unrestricted Join**. It returns **n1** × **n2** rows where **n1** is number of rows in first table and **n2** is number of rows in second table.

CARTESIAN PRODUCT

In unrestricted join or *Cartesian product* of two tables, all possible concatenations are formed of all rows of both the tables.

14.4.2 Table Aliases

A **Table Alias** is a temporary label given along with table name in FROM clause.

To cut down on the amount of typing required in your queries you can use aliases for table names in the SELECT and WHERE clauses. Consider the following join query. Consider the example 12.1.

Table Structures

C. Treats
- Phy_Id
- Patient_No
- Procedure_No
- Date_Treated
- Treat_Result

D. Physicians
- Phy_Id
- Phy_Phone
- Phy_Name

EXAMPLE 14.1. *List the patients who had either Dr. N. Pandya, Dr. A. Sapra, Dr. Suleman Rashid, Dr. Keith John as physician. [Dr. N. Pandya has physician id as 8883, Dr. Sapra as 8887, Dr. Rashid as 8886 and Dr. John as 8882]. (see table structures on the left)*

1. A join can combine views also. A view is a stored query that appears like a table and can be used like a table.

Path Wala

SOLUTION.

```
SELECT DISTINCT patient_no, phy_id
FROM treats
WHERE phy_id IN (8887, 8886, 8883, 8882) ;
```

The above example 12.1 where the query listed the *PATIENT_NO* and *PHY_ID*, but not the PAT_NAME of *PHY_NAME* fields. The following query joins the *PATIENT*, *PHYSICIAN*, and *TREATS* tables to produce the desired information. The query also demonstrates the use of an ALIAS for a table name – here we have three aliases (*PA*, *PH*, and *TR*), one for each table.

EXAMPLE 14.2. *List which patient was treated by which physician. Also list their ids along with their names. (see table structures on the left)*

SOLUTION.

```
SELECT DISTINCT PA.patient_no, pat_name, PH.phy_id, phy_name
FROM patient PA, physician PH, treats TR
WHERE    PA.patient_no = TR.patient_no AND
PH.phy_id = TR.phy_id AND
PH.phy_id IN (8887, 8886, 8883, 8882) ;
```

14.4.3 Equi-Join and Natural Join

The Join, in which columns are compared for equality, is called **Equi-Join.**

A *non-equi-join* is a query that specifies some relationship other than equality between the columns.

The Join in which only one of the identical columns (coming from joined tables) exists, is called **Natural Join.**

The Equi-Join and Natural Join are equivalent except that duplicate columns are eliminated in the Natural Join that would otherwise appear in the Equi-Join.

EQUI-JOIN

The Join, in which columns are compared for equality, is called *Equi-Join*.

NATURAL JOIN

The Join in which only one of the identical columns (coming from joined tables) exists, is called *Natural Join*.

EXAMPLE 14.3. *Display patient_no, the date when he/she is discharged and the charge paid by him/her.*

SOLUTION.

```
SELECT billed.patient_no, date_discharged, charge
FROM billed, patient
WHERE patient.patient_no = billed.patient_no ;
```

The above query shows the *PATIENT_NO* and *DATE_DISCHARGED* from the *PATIENT* table and the associated *CHARGE* from the *BILLED* table.

14.4.4 Additional Search Conditions in Joins

Once you have joined tables, you can filter information from joined table by incorporating additional search conditions.

Refer to *example 14.4* given below.

Path Wala

EXAMPLE 14.4. *Display details like department number, department name, employee number, employee name, job, and salary. And order the rows by employee number with department number.*

SOLUTION.

```
SQL > SELECT EMP.DEPTNO, DNAME, EMPNO, ENAME, JOB, SAL
      FROM EMP, DEPT
      WHERE EMP.DEPTNO = DEPT.DEPTNO          - - join on DEPTNO
      ORDER BY EMP.DEPTNO, EMPNO ;
```

If you want to extract such information only for SALES department, you can do so by giving a query similar to the one given in example 14.5.

EXAMPLE 14.5. *Refer to example 14,4. Do this only for SALES department.*

SOLUTION.

```
SQL > SELECT E.DEPTNO, DNAME, EMPNO, ENAME, JOB, SAL
      FROM EMP E, DEPT D
      WHERE E.DEPTNO = D.DEPTNO
      AND DNAME = 'SALES'
      ORDER BY E.DEPTNO, EMPNO ;
```

And now the output produced will be :

DEPTNO	DNAME	EMPNO	ENAME	JOB	SAL
.....	.....	.....	.....	.....	.....
30	SALES	7499	ALLEN	SALESMAN	1600
30	SALES	7521	WARD	SALESMAN	1250
30	SALES	7654	MARTIN	SALESMAN	1250
30	SALES	7698	BLAKE	MANAGER	3750
30	SALES	7844	TURNER	SALESMAN	1500
30	SALES	7900	JAMES	CLERK	950

6 rows selected.

Consider one more example.

EXAMPLE 14.6. *Display details like department number, department name, employee number, employee name, job, and salary. And order the rows by employee number with department number. These details should be only for employees earning atleast Rs. 1500 and of SALES department.*

SOLUTION.

```
SQL > SELECT E.DEPTNO, DNAME, EMPNO, ENAME, JOB, SAL
      FROM EMP E, DEPT D
      WHERE E.DEPTNO = D.DEPTNO
      AND DNAME = 'SALES'
      AND SAL >= 1500
      ORDER BY E.DEPTNO, EMPNO ;
```

Path Wala

And this time the output is :

DEPTNO	DNAME	EMPNO	ENAME	JOB	SAL
.....	.....	.....	.....	.....	.....
30	SALES	7499	ALLEN	SALESMAN	1600
30	SALES	7698	BLAKE	MANAGER	3750
30	SALES	7844	TURNER	SALESMAN	1500

So, finally you know that a join query combines columns of one table to that of another to create a single table. A join query matches up a column with one table to a column in another column. A join query does not alter either table, but temporarily combine data from each table to be viewed as a single table.

LET US REVISE

- *The GROUP BY clause combines all those records that have identical value in a particular field or a group of fields.*
- *GROUP BY clause is used to divide the result in groups.*
- *A group within another group is called* ***Nested Group****.*
- *Nested grouping can be done by providing multiple fields in the GROUP BY expression.*
- *All fields containing a NULL value are considered to have a value and are grouped to have a value and are grouped with the fields containing non-NULL values.*
- *The SELECT list of a group can include expressions returning single value per group or constants.*
- *The HAVING clause is used to specify filtering condition for groups.*
- *The difference between WHERE and HAVING clause is that WHERE conditions are applicable on individual rows whereas HAVING conditions are applicable on groups as formed by GROUP BY clause.*
- *The Join, in which columns are compared for equality, is called Equi-Join.*

Objective Type Questions — OTQs

Multiple Choice Questions

1. A _____ is a query that retrieves rows from more than one table or view:
   (*a*) Start (*b*) End (*c*) Join (*d*) All of these
2. The HAVING clause does which of the following ?
   (*a*) Acts EXACTLY like a WHERE clause.
   (*b*) Acts like a WHERE clause but is used for columns rather than groups.
   (*c*) Acts like a WHERE clause but is used for groups rather than rows.
   (*d*) Acts like a WHERE clause but is used for rows rather than columns.
3. Aggregate functions can be used in the select list or the_____clause of a select statement. They cannot be used in a _____ clause.
   (*a*) Where, having (*b*) Having, where
   (*c*) Group by, having (*c*) Group by, where

Path Wala

4. SQL applies conditions on the groups through _____ clause after groups have been formed,
   (a) Group by (b) With
   (c) Where (d) Having
5. Which clause is used with "aggregate functions" ?
   (a) GROUP BY (b) SELECT
   (c) WHERE (d) Both (a) and (c)
6. What is the meaning of "HAVING" clause in SELECT query ?
   (a) To filter out the summary groups
   (b) To filter out the column groups
   (c) To filter out the row and column values
   (d) None of the mentioned
7. Where and Having clauses can be used interchangeably in SELECT queries ?
   (a) True (b) False
   (c) Only in views (c) With order by
8. The operation whose result contains all pairs of tuples from the two relations, regardless of whether their attribute values match.
   (a) Join (b) Cartesian product
   (c) Intersection (d) Set difference
9. Which SQL function is used to count the number of rows in a SQL query ?
   (a) COUNT( ) (b) NUMBER( )
   (c) SUM( ) (d) COUNT(*)
10. Which of the following is not an aggregate function ?
   (a) Avg (b) Sum
   (c) With (d) Min
11. All aggregate functions except _____ ignore null values in their input collection.
   (a) Count(attribute) (b) Count(*)
   (c) Avg (d) Sum
12. Which of the following is a SQL aggregate function ?
   (a) LEFT (b) AVG
   (c) JOIN (d) LEN
13. Which of the following group functions ignore NULL values ?
   (a) MAX (b) COUNT
   (c) SUM (d) All of the above
14. The functions which work with individual rows' data are called _____ function.
   (a) Single row (b) Multiple rows
   (c) Aggregate (d) None of these
15. Function count( ) is a/an _____ function.
   (a) Single row (b) Multiple rows
   (c) Aggregate (d) None of these

Path Wala

16. The sum( ), if used in a condition, is used with _____ clause.
    (a) Group by
    (b) With
    (c) Where
    (d) Having
17. Which clause cannot be used with "aggregate functions" ?
    (a) GROUP BY
    (b) SELECT
    (c) WHERE
    (d) Both (a) and (c)
18. What is the meaning of "WHERE" clause in a GROUP BE query ?
    (a) To filter out the summary groups
    (b) To filter out the column groups
    (c) To filter out the row and column values before creating groups
    (d) None of the mentioned
19. The following SQL is which type of join :

```
SELECT CUSTOMER.CUST_ID, ORDER.CUST_ID, NAME, ORDER_ID
FROM CUSTOMER,ORDER
WHERE CUSTOMER.CUST_ID=ORDER.CUST_ID?
```

    (a) Equi-join
    (b) Natural join
    (c) Outer join
    (d) Cartesian product
20. The following SQL is which type of join :

```
SELECT CUSTOMER.CUST_ID, ORDER.CUST_ID, NAME, ORDER_ID
FROM CUSTOMER,ORDER
```

    (a) Equi-join
    (b) Natural join
    (c) Outer join
    (d) Cartesian product
21. Which product is returned in a join query have no join condition :
    (a) Equijoins
    (b) Cartesian product
    (c) Both (a) and (b)
    (d) None of the mentioned
22. Which is a join condition contains an equality operator :
    (a) Equijoins
    (b) Cartesian product
    (c) Both (a) and (b)
    (d) None of the mentioned

Fill in the Blanks

1. To specify condition with a GROUP BY clause , _____ clause is used
2. Only _____ functions are used with GROUP BY.
3. A _____ is a means for combining fields from two tables by using values common to each.
4. The equi join uses _____ operator in the join condition.
5. Natural join joins two tables on the basis of a _____ field.
6. The SQL built-in function _____ totals values in numeric columns.
7. The SQL built-in function _____ computes the average of values in numeric columns.
8. The SQL built-in function _____ obtains the largest value in a numeric column.
9. The SQL built-in function _____ obtains the smallest value in a numeric column.

Path Wala

10. The SQL built-in function _____ computes the number of rows in a table.
11. The SELECT clause _____ is used to collect those rows that have the same value in a specified column.
12. To compare an aggregate value in a condition, _____ clause is used.
13. To create a summary of records based on the common value in a field in different rows of the table _____ clause is used.
14. To get data from two or more tables having some common fields, _____ query is created.
15. In equi join, the join condition joins the two table using _____ operator.

True/False Questions

1. The HAVING clause acts like a WHERE clause, but it identifies groups that meet a criterion, rather than rows.
2. Data manipulation language (DML) commands are used to define a database, including creating, altering, and dropping tables and establishing constraints.
3. The SQL keyword GROUP BY instructs the DBMS to group together those rows that have the same value in a column.
4. Equi join can use any operator for joining two tables.
5. Missing join condition in join query produces catersian product.
6. COUNT(field_name) tallies only those rows that contain a value; it ignores all null values.
7. SUM, AVG, MIN, and MAX can only be used with numeric columns.
8. The SQL keyword GROUP BY instructs the DBMS to group together those rows that have the same value in a column.
9. The HAVING and WHERE clauses are interchangeable.
10. The HAVING clauses can take any valid SQL function in its condition.
11. HAVING clause can only be used if the SELECT query has GROUP BY clause.
12. With GROUP BY, the select-list of the SELECT statement can only take the group-field and/or aggregate function.
13. A join query without the join condition produces a cartesian product.
14. You can specify any type of condition, using any comparison operator in an equi-join.
15. Join can only be created from two tables.

NOTE : Answers for OTQs are given at the end of the book.

Solved Problems

1. *The SQL SELECT provides clauses for sorting data and for summarizing results. Write the names of clauses for these.*

Solution. The ORDER BY clause of SQL SELECT statement allows to sort the data of result set. The GROUP BY clause of SQL SELECT statement allows to create summarized results of grouped data from table.

 Path Wala

2. *You want to group the result set based on some column's value. Also, you want that the grouped result should appear in a sorted order. In which order will you write the two clauses (for sorting and for grouping). Give example to support your answer.*

Solution. When we use GROUP BY clause (for grouping of data) and ORDER BY clause (for sorting data) together, the ORDER BY clause always follows other clauses. That is, the GROUP BY clause will come before ORDER BY clause.

For example,

```
SELECT userid, SUM(score) AS total_score
FROM user_score
GROUP BY userid
ORDER BY userid ASC;
```

3. *What are different types of SQL functions ?*

Solution. (*i*) Single Row (or Scalar) functions.

(*ii*) Multiple Row (or Group or Aggregate) functions.

(*i*) ***Single Row Functions*** work with a single row at a time. A single row function returns a result for every row of a queried table.

(*ii*) ***Multiple Row or Group Functions*** work with data of multiple rows at a time and return aggregated value.

4. *What is the significance of GROUP BY clause in a SQL query ?*

Solution. The GROUP BY clause combines all those records that have identical values in a particular field or a group of fields. This grouping results into one summary record per group if group-functions are used with it.

5. *What is the difference between a WHERE clause and a HAVING clause of SQL SELECT statement ?*

Solution. The difference between WHERE and HAVING clause is that WHERE conditions are applicable on individual rows whereas HAVING conditions are applicable on groups as formed by GROUP BY clause.

6. *Write a query to display the Sum, Average, Highest and Lowest salary of the employees.*

Solution.

```
mysql> SELECT SUM (sal), AVG (sal), MAX (sal), MIN (sal)
       FROM empl ;
```

7. *Write a query to display the Sum, Average, Highest and Lowest salary of the employees grouped by department number.*

Solution.

```
mysql> SELECT SUM (sal), AVG (sal), MAX (sal), MIN (sal)
       FROM empl GROUP BY deptno ;
```

8. *Write a query to display the Sum, Average, Highest and Lowest salary of the employees grouped by department number and sub-grouped by job.*

Solution.

```
mysql> SELECT SUM (sal), AVG (sal), MAX (sal), MIN (sal)
       FROM empl
       GROUP BY deptno, job ;
```

Path Wala

9. *Write a query to display the difference of highest and lowest salary of each department having maximum salary > 4000.*

Solution.

```
mysql> SELECT MAX (sal) – MIN (sal) "Difference" FROM empl
       GROUP BY deptno
       HAVING MAX (sal) > 4000 ;
```

10. *Gopi Krishna is using a table Employee. It has the following columns :* [CBSE D 2014]

Code, Name, Salary, Deptcode

He wants to display maximum salary departmentwise. He wrote the following command :

SELECT Deptcode, Max(Salary) FROM Employee ;

But he did not get the desired result.

Rewrite the above query with necessary changes to help him get the desired output.

Solution.

```
SELECT Deptcode, Max(Salary)
FROM Employee
GROUP BY Deptcode ;
```

11. *Shanya Khanna is using a table Employee. It has the following columns :* [CBSE OD 2014]

Admno, Name, Agg, Stream

[column Agg contains Aggregate marks]

She wants to display highest Agg obtained in each Stream.

She wrote the following statement :

SELECT Stream, MAX(Agg) FROM Employee ;

But she did not get the desired result. Rewrite the above query with necessary changes to help her get the desired output.

Solution.

```
SELECT Stream, MAX(Agg)
FROM Employee
GROUP BY Stream ;
```

12. *Write a query that counts the number of salespeople registering orders for each day. (If a salesperson has more than one order on a given day, he or she should be counted only once.).*

Solution.

```
SELECT ord_date, count (DISTINCT salesman_code) FROM orders
GROUP BY ord_date ;
```

13. *Write a query on the customers table that will find the highest rating in each city. Put the output in this form : For the city (city), the highest rating is : (rating).*

Solution.

```
SELECT 'For the city', city, 'the highest rating is :', MAX (rating)
FROM customers
GROUP BY city ;
```

Path Wala

14. *What is a SQL Join ?*

Solution. A SQL join is an query used within the SQL (Structured Query Language) to combine data from two tables on the basis of a common field

15. *What is natural join ?*

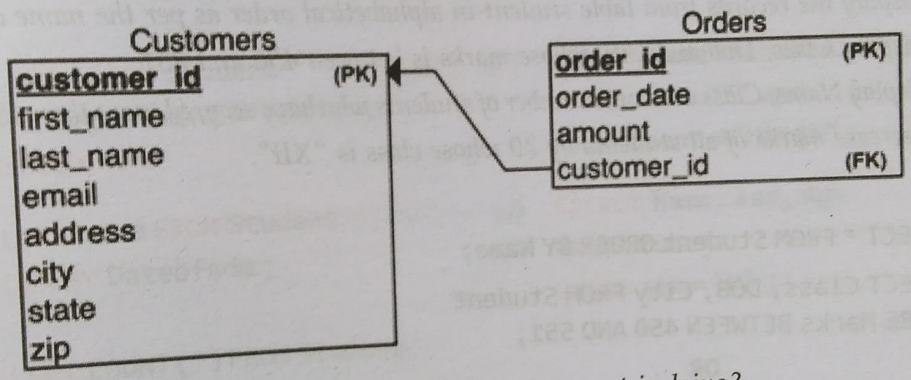
Solution. A NATURAL JOIN is a JOIN operation that creates an implicit join clause for you based on the common columns in the two tables being joined. Common columns are columns that have the same name in both tables.

16. *What is equi join ?*

Solution. An Equi join is a sql join where we use the equal sign as the comparison operator *i.e.*, "=" between two tables while specifying the join condition, *e.g.*,

```
SELECT * FROM EMP E,DEPT D WHERE E.DEPTNO = D.DEPTNO ;
```

17. *Two tables Customers and Orders shown below have common key as customer_id :*



Consider the following statement and state what this statement is doing?

```
SELECT order_date, order_amount
FROM customers, orders
WHERE customers.customer_id = orders.customer_id
AND customer_id = 3 ;
```

Solution. The given query is joining the two tables on the common field customer_id and selecting order_date and order_amount from the orders placed by customer_id = 3.

18. *Write a output for SQL queries (i) to (iii), which are based on the table : **STUDENT** given below :*

Table : **STUDENT**

RollNo	Name	Class	DOB	Gender	City	Marks
1	Nanda	X	06-06-1995	M	Agra	551
2	Saurabh	XII	07-05-1993	M	Mumbai	462
3	Sanal	XI	06-05-1994	F	Delhi	400
4	Trisla	XII	08-08-1995	F	Mumbai	450
5	Stort	XII	08-10-1995	M	Delhi	369
6	Marisla	XI	12-12-1994	F	Dubai	250
7	Neha	X	08-12-1995	F	Moscow	377
8	Nishant	X	12-06-1995	M	Moscow	489

Path Wala

(i) SELECT COUNT(*), City FROM STUDENT GROUP BY CITY HAVING COUNT(*) > 1 ;

(ii) SELECT MAX(DOB), MIN(DOB) FROM STUDENT;

(iii) SELECT NAME, GENDER FROM STUDENT WHERE CITY = "Delhi" ; [CBSE Sample Paper 2019-20]

Solution.

(i)

COUNT(*)	City
2	Mumbai
2	Delhi
2	Moscow

(ii)

MAX(DOB)	MIN(DOB)
08-12-1995	07-05-1993

(iii)

NAME	GENDER
Sanal	F
Stort	M

19. *Write SQL queries for (i) to (iv), which are based on the table : STUDENT given in the SP 18 on previous page :*

(i) *To display the records from table student in alphabetical order as per the name of the student.*

(ii) *To display Class, Dob and City whose marks is between 450 and 551.*

(iii) *To display Name, Class and total number of students who have secured more than 450 marks, class wise.*

(iv) *To increase marks of all students by 20 whose class is "XII".* [CBSE Sample Paper 2019-20]

Solution.

(i) SELECT * FROM Student ORDER BY Name;

(ii) SELECT Class, DOB, City FROM Student WHERE Marks BETWEEN 450 AND 551;

OR

SELECT Class, DOB, City FROM Student WHERE Marks >= 450 AND Marks <= 551 ;

(iii) SELECT Name,Class, COUNT(*) FROM Student GROUP BY Class HAVING Marks > 450;

(iv) UPDATE Student SET Marks = Marks + 20 WHERE Class = "XII";

20. *Given the following student relation :* (CBSE Outside Delhi 1999)

relation Student

No.	Name	Age	Department	Dateofadm	Fee	Sex
1.	Pankaj	24	Computer	10/01/97	120	M
2.	Shalini	21	History	24/03/98	200	F
3.	Sanjay	22	Hindi	12/12/96	300	M
4.	Sudha	25	History	01/07/99	400	F
5.	Rakesh	22	Hindi	05/09/97	250	M
6.	Shakeel	30	History	27/06/98	300	M
7.	Surya	34	Computer	25/02/97	210	M
8.	Shikha	23	Hindi	31/07/97	200	F

Path Wala

Write SQL commands for (a) to (f) and write output for (g).

(a) To show all information about the students of History department

(b) *To list the names of female students who are in Hindi department*

(c) *To list names of all students with their date of admission in ascending order.*

(d) *To display student's Name, Fee, Age for male Students only.*

(e) *To count the number of student with Age < 23.*

(f) *To inset a new row in the STUDENT table with the following data :*

9, "Zaheer", 36, "Computer", {12/03/97}, 230, "M"

(g) *Give the output of following SQL statements :*

(i) Select COUNT (distinct department) from STUDENT ;

(ii) Select MAX (Age) from STUDENT where Sex = "F" ;

(iii) Select AVG (Fee) from STUDENT where Dateofadm < {01/01/98} ;

(iv) Select SUM (Fee) from STUDENT where Dateofadm < {01/01/98} ;

Solution.

(a)
```
SELECT * FROM Student
WHERE Department = "History" ;
```

(b)
```
SELECT Name FROM Student
WHERE sex = "F" and
Department = "Hindi" ;
```

(c)
```
SELECT name FROM Student
ORDER BY Dateofadm ;
```

(d)
```
SELECT Name, Fee, Age
FROM Student
WHERE sex = "M" ;
```

(e)
```
SELECT COUNT (*) FROM Student
WHERE Age < 23 ;
```

(f)
```
INSERT INTO Student
VALUES (9, "Zaheer", "Computer", "12/03/97", 230, "M") ;
```

(g) (i) 3 (ii) 25 (iii) 216 (iv) 1080

21. *Given the following tables for a database LIBRARY :* (CBSE Delhi 2004)

Table : BOOKS

Book_Id	Book_Name	Author_Name	Publishers	Price	Type	Qty.
C0001	Fast Cook	Lata Kapoor	EPB	355	Cookery	5
F0001	The Tears	William Hopkins	First Publ.	650	Fiction	20
T0001	My First C++	Brian & Brooke	EPB	350	Text	10
T0002	C++ Brainworks	A.W. Rossaine	TDH	350	Text	15
F0002	Thunderbolts	Anna Roberts	First Publ.	750	Fiction	50

Table : ISSUED

Book_Id	Quantity_Issued
T0001	4
C0001	5
F0001	2

Path Wala

Write SQL queries for (a) to (f) :

(a) To show Book name, Author name and Price of books of First Publ. publishers.

(b) *To list the names from books of Text type.*

(c) *To display the names and price from books in ascending order of their price.*

(d) *To increase the price of all books of EPB Publishers by 50.*

(e) *To display the Book_Id, Book_Name and Quantity_Issued for all books which have been issued. (The query will require contents from both the tables.)*

(f) *To insert a new row in the table Issued having the following data : "F0003", 1*

(g) *Give the output of the following queries based on the above tables :*

(i) SELECT COUNT(*) FROM Books ;

(ii) SELECT MAX(Price) FROM Books WHERE Quantity >=15 ;

(iii) SELECT Book_Name, Author_Name FROM Books WHERE Publishers = "EPB" ;

(iv) SELECT COUNT (DISTINCT Publishers) FROM Books WHERE Price >= 400 ;

Solution.

(a)
```
SELECT Book_Name, Author_Name, Price
FROM Books
WHERE Publishers = "First Publ." ;
```

(b)
```
SELECT Book_Name
FROM Books
WHERE Type = "Text" ;
```

(c)
```
SELECT Book_Name, Price
FROM Books
ORDER BY Price ;
```

(d)
```
UPDATE Books
SET Price = Price + 50
WHERE Publishers = "EPB" ;
```

(e)
```
SELECT Books.Book_Id, Book_Name, Quantity_Issued
FROM Books, Issued
WHERE Books.Book_Id = Issued.Book_Id ;
```

(f)
```
INSERT INTO Issued
VALUES("F0003", 1) ;
```

(g) (i) 5 (ii) 750 (iii) Fast Cook Lata Kapoor
My First C++ Brian & Brooke (iv) 1

22. *Consider the following DEPT and WORKER tables. Write SQL queries for (i) to and (iv) find outputs for SQL queries (v) to (viii) :* (CBSE Delhi 2015)

Table : DEPT

DCODE	DEPARTMENT	CITY
D01	MEDIA	DELHI
D02	MARKETING	DELHI
D03	INFRASTRUCTURE	MUMBAI
D05	FINANCE	KOLKATA
D04	HUMAN RESOURCE	MUMBAI

Path Wala

Table: WORKER

WNO	NAME	DOJ	DOB	GENDER	DCODE
1001	George K	2013-09-02	1991-09-01	MALE	D01
1002	Ryma Sen	2012-12-11	1990-12-15	FEMALE	D03
1003	Mohitesh	2013-02-03	1987-09-04	MALE	D05
1007	Anil Jha	2014-01-17	1984-10-19	MALE	D04
1004	Manila Sahai	2012-12-09	1986-11-14	FEMALE	D01
1005	R Sahay	2013-11-18	1987-03-31	MALE	D02
1006	Jaya Priya	2014-06-09	1985-06-23	FEMALE	D05

Note. *DOJ refers to date of joining and DOB refers to date of birth of workers.*

(*i*) To display Wno, Name, Gender from the table WORKER in descending order of Wno.

(*ii*) *To display the Name of all the FEMALE workers from the table WORKER.*

(*iii*) *To display the Wno and Name of those workers from the table WORKER who are born between '1987-01-01' and '1991-12-01'.*

(*iv*) *To count and display MALE workers who have joined after '1986-01-01'.*

(*v*)
```
SELECT COUNT(*), DCODE FROM WORKER
GROUP BY DCODE HAVING COUNT(*) > 1 ;
```

(*vi*)
```
SELECT DISTINCT DEPARTMENT FROM DEPT ;
```

(*vii*)
```
SELECT NAME, DEPARTMENT, CITY FROM WORKER W, DEPT D WHERE
W.DCODE = D.DCODE AND WNO < 1003 ;
```

(*viii*)
```
SELECT MAX(DOJ), MIN(DOB) FROM WORKER ;
```

Solution.

(*i*)
```
SELECT WNO, NAME, Gender,
FROM WORKER
ORDER BY WNO Desc ;
```

(*ii*)
```
SELECT NAME
FROM WORKER
WHERE GENDER = 'FEMALE' ;
```

(*iii*)
```
SELECT WNO, NAME
FROM WORKER
WHERE    DOB BETWEEN '1987-01-01' AND '1991-12-01' ;
```

(*iv*)
```
SELECT count(*)
FROM WORKER
WHERE GENDER = 'MALE' AND DOJ > '1986-01-01' ;
```

(*v*)

COUNT(*)	DCODE
2	D01
2	D05

(*vi*)

DISTINCT DEPARTMENT
MEDIA
MARKETING
INFRASTRUCTURE
FINANCE
HUMAN RESOURCE

Path Wala

(vii)

NAME	DEPARTMENT	CITY
George K	MEDIA	DELHI
Ryma Sen	INFRASTRUCTURE	MUMBAI

(viii)

Max(DOJ)	Min(DOB)
2014-06-09	1984-10-19

GLOSSARY

Single Row function	A function that works on a value in single row.
Multiple Row function	A function that works on values of a group of rows.
Group function	Function that works on a group of rows. A multiple row function.
Aggregate function	Function that works on aggregate of rows. A multiple row function.

Assignment

Type A : Short Answer Questions/Conceptual Questions

1. What is the difference between HAVING and WHERE clause ?
2. What is the use of GROUP BY clause ?
3. What are aggregate functions? What is their use? Give some examples.
4. What type of functions can you use with GROUP BY and HAVING clauses ?
5. What is sql join ? How is it useful ?
6. What are most common types of SQL joins ?

Type B : Application Based Questions

Following questions are based on these tables :

Table ***BOOK_INFORMATION***

Column Name
BOOK_ID
BOOK_TITLE
PRICE

Table ***SALES***

Column Name
STORE_ID
SALES_DATE
SALES_AMOUNT

Table ***EXAM_RESULTS***

STU_ID	FNAME	LNAME	EXAM_ID	EXAM_SCORE
10	LAURA	LYNCH	1	90
10	LAURA	LYNCH	2	85
11	GRACE	BROWN	1	78
11	GRACE	BROWN	2	72
12	JAY	JACKSON	1	95
12	JAY	JACKSON	2	92
13	WILLIAM	BISHOP	1	70
13	WILLIAM	BISHOP	2	100
14	CHARLES	PRADA	2	85

Path Wala

1. Which SQL statement allows you to find the highest price from the table BOOK_INFORMATION?
   (a) SELECT BOOK_ID, BOOK_TITLE, MAX(PRICE) FROM BOOK_INFORMATION;
   (b) SELECT MAX(PRICE) FROM BOOK_INFORMATION;
   (c) SELECT MAXIMUM(PRICE) FROM BOOK_INFORMATION;
   (d) SELECT PRICE FROM BOOK_INFORMATION ORDER BY PRICE DESC;

2. Which SQL statement lets you find the sales amount for each store?
   (a) SELECT STORE_ID, SUM(SALES_AMOUNT) FROM SALES;
   (b) SELECT STORE_ID, SUM(SALES_AMOUNT) FROM SALES ORDER BY STORE_ID;
   (c) SELECT STORE_ID, SUM(SALES_AMOUNT) FROM SALES GROUP BY STORE_ID;
   (d) SELECT STORE_ID, SUM(SALES_AMOUNT) FROM SALES HAVING UNIQUE STORE_ID;

3. Which SQL statement lets you list all stores whose total sales amount is over 5000 ?
   (a) SELECT STORE_ID, SUM(SALES_AMOUNT) FROM SALES GROUP BY STORE_ID HAVING SUM(SALES_AMOUNT) > 5000;
   (b) SELECT STORE_ID, SUM(SALES_AMOUNT) FROM SALES GROUP BY STORE_ID HAVING SALES_AMOUNT > 5000;
   (c) SELECT STORE_ID, SUM(SALES_AMOUNT) FROM SALES WHERE SUM(SALES_AMOUNT) > 5000 GROUP BY STORE_ID;
   (d) SELECT STORE_ID, SUM(SALES_AMOUNT) FROM SALES WHERE SALES_AMOUNT > 5000 GROUP BY STORE_ID;

4. Which SQL statement lets you find the total number of stores in the SALES table?
   (a) SELECT COUNT(STORE_ID) FROM SALES;
   (b) SELECT COUNT(DISTINCT STORE_ID) FROM SALES;
   (c) SELECT DISTINCT STORE_ID FROM SALES;
   (d) SELECT COUNT(STORE_ID) FROM SALES GROUP BY STORE_ID;

5. Which SQL statement allows you to find the total sales amount for Store ID 25 and the total sales amount for Store ID 45?
   (a) SELECT STORE_ID, SUM(SALES_AMOUNT) FROM SALES WHERE STORE_ID IN (25,45) GROUP BY STORE_ID;
   (b) SELECT STORE_ID, SUM(SALES_AMOUNT) FROM SALES GROUP BY STORE_ID HAVING STORE_ID IN (25,45);
   (c) SELECT STORE_ID, SUM(SALES_AMOUNT) FROM SALES WHERE STORE_ID IN (25,45);
   (d) SELECT STORE_ID, SUM(SALES_AMOUNT) FROM SALES WHERE STORE_ID = 25 AND STORE_ID = 45 GROUP BY STORE_ID;

6. What SQL statement do we use to find the average exam score for EXAM_ID = 1?
   (a) SELECT AVG(EXAM_SCORE) FROM EXAM_RESULTS;
   (b) SELECT AVG(EXAM_SCORE) FROM EXAM_RESULTS GROUP BY EXAM_ID WHERE EXAM_ID = 1;
   (c) SELECT AVG(EXAM_SCORE) FROM EXAM_RESULTS GROUP BY EXAM_ID HAVING EXAM_ID = 1;
   (d) SELECT COUNT(EXAM_SCORE) FROM EXAM_RESULTS WHERE EXAM_ID = 1;

Path Wala

7. Which SQL statement do we use to find out how many students took each exam?
   (a) SELECT COUNT(DISTINCT STU_ID) FROM EXAM_RESULTS GROUP BY EXAM_ID;
   (b) SELECT EXAM_ID, MAX(STU_ID) FROM EXAM_RESULTS GROUP BY EXAM_ID;
   (c) SELECT EXAM_ID, COUNT(DISTINCT STU_ID) FROM EXAM_RESULTS GROUP BY EXAM_ID;
   (d) SELECT EXAM_ID, MIN(STU_ID) FROM EXAM_RESULTS GROUP BY EXAM_ID;

8. What SQL statement do we use to print out the record of all students whose last name starts with 'L'?
   (a) SELECT * FROM EXAM_RESULTS WHERE LNAME LIKE 'L%';
   (b) SELECT * FROM EXAM_RESULTS WHERE LNAME LIKE 'L';
   (c) SELECT * FROM EXAM_RESULTS WHERE LNAME = 'L';
   (d) SELECT * FROM EXAM_RESULTS WHERE LNAME <> 'L';

9. What is the result of the following SQL statement?

```
SELECT MAX(EXAM_SCORE) FROM EXAM_RESULTS GROUP BY EXAM_ID HAVING EXAM_ID = 1;
```

   (a) 90 (b) 85 (c) 100 (d) 95

10. Given the following table :

Table : CLUB

COACH-ID	COACHNAME	AGE	SPORTS	DATOFAPP	PAY	SEX
1.	KUKREJA	35	KARATE	27/03/1996	1000	M
2.	RAVINA	34	KARATE	20/01/1998	1200	F
3.	KARAN	34	SQUASH	19/02/1998	2000	M
4.	TARUN	33	BASKETBALL	01/01/1998	1500	M
5.	ZUBIN	36	SWIMMING	12/01/1998	750	M
6.	KETAKI	36	SWIMMING	24/02/1998	800	F
7.	ANKITA	39	SQUASH	20/02/1998	2200	F
8.	ZAREEN	37	KARATE	22/02/1998	1100	F
9.	KUSH	41	SWIMMING	13/01/1998	900	M
10.	SHAILYA	37	BASKETBALL	19/02/1998	1700	M

Give the output of following SQL statements :

(i) SELECT COUNT (DISTINCT SPORTS) FROM Club ;
(ii) SELECT MIN(Age) FROM CLUB WHERE Sex = 'F' ;
(iii) SELECT AVG(Pay) FROM CLUB WHERE Sports = 'KARATE' ;
(iv) SELECT SUM(Pay) FROM CLUB WHERE Datofapp > '31/01/98' ;

Path Wala

11. Given the following table :

Table : STUDENT

No.	Name	Stipend	Stream	AvgMark	Grade	Class
1	Karan	400.00	Medical	78.5	B	12B
2	Divakar	450.00	Commerce	89.2	A	11C
3	Divya	300.00	Commerce	68.6	C	12C
4	Arun	350.00	Humanities	73.1	B	12C
5	Sabina	500.00	Nonmedical	90.6	A	11A
6	John	400.00	Medical	75.4	B	12B
7	Robert	250.00	Humanities	64.4	C	11A
8	Rubina	450.00	Nonmedical	88.5	A	12A
9	Vikas	500.00	Nonmedical	92.0	A	12A
10	Mohan	300.00	Commerce	67.5	C	12C

Give the output of following SQL statements :

(*i*) SELECT MIN(AvgMark) FROM STUDENT WHERE AvgMark < 75 ;

(*ii*) SELECT SUM(Stipend) FROM Student WHERE Grade = 'B' ;

(*iii*) SELECT AVG(Stipend) FROM Student WHERE Class = '12A' ;

(*iv*) SELECT COUNT(DISTINCT) FROM Student ;

12. *In a Database, there are two tables given below :*

Table : EMPLOYEE

EMPLOYEEID	NAME	SALES	JOBID
E1	SUMIT SINHA	1100000	102
E2	VIJAY SINGH TOMAR	1300000	101
E3	AJAY RAJPAL	1400000	103
E4	MOHIT RAMNANI	1250000	102
E5	SHAILJA SINGH	1450000	103

Table : JOB

JOBID	JOBTITLE	SALARY
101	President	200000
102	Vice President	125000
103	Administration Assistant	80000
104	Accounting Manager	70000
105	Accountant	65000
106	Sales Manager	80000

Write SQL Queries for the following :

(*i*) *To display employee ids, names of employees, job ids with corresponding job titles.*

(*ii*) *To display names of employees, sales and corresponding job titles who have achieved sales more than 1300000.*

(*iii*) *To display names and corresponding job titles of those employees who have 'SINGH' (anywhere) in their names.*

(*iv*) *Identify foreign key in the table EMPLOYEE.*

(*v*) Write SQL command to change the JOBID to 104 of the EMPLOYEE with ID as E4 in the table 'EMPLOYEE'. [CBSE D 15]

Path Wala

13. Consider the following tables Employee and Salary. Write SQL commands for the statements (*i*) to (*iv*) and give outputs for SQL queries (*v*) to (*vii*)

Table : Employee

Eid	Name	Depid	Qualification	Sec
1	Deepali Gupta	101	MCA	F
2	Rajat Tyagi	101	BCA	M
3	Hari Mohan	102	B.A.	M
4	Harry	102	M.A.	M
5	Sumit Mittal	103	B.Tech.	M
6	Jyoti	101	M.Tech.	F

Table : Salary

Eid	Basic	D.A.	HRA	Bonus
1	6000	2000	2300	200
2	2000	300	300	30
3	1000	300	300	40
4	1500	390	490	30
5	8000	900	900	80
6	10000	300	490	89

(*i*) To display the frequency of employees department wise.
(*ii*) To list the names of those employees only whose name starts with 'H'
(*iii*) To add a new column in salary table. The column name is Total_Sal.
(*iv*) To store the corresponding values in the Total_Sal column.
(*v*) Select max(Basic) from Salary where Bonus > 40 ;
(*vi*) Select count(*) from Employee group by Sex ;
(*vii*) Select Distinct Depid from Employee ;

14. With reference to following relations PERSONAL and JOB answer the questions that follow :

Create following tables such that *Empno* and *Sno* are not null and unique, *date of birth* is after '12-Jan-1960', *name* is never blank, *Area* and *Native place* is valid, *hobby, dept* is not empty, salary is between 4000 and 10000.

Table : Personal

Empno	Name	Dobirth	Native-place	Hobby
123	Amit	23-Jan-1965	Delhi	Music
127	Manoj	12-dec-1976	Mumbai	Writing
124	Abhai	11-aug-1975	Allahabad	Music
125	Vinod	04-apr-1977	Delhi	Sports
128	Abhay	10-mar-1974	Mumbai	Gardening
129	Ramesh	28-oct-1981	Pune	Sports

Path Wala

Table : Job

Sno	Area	App_date	Salary	Retd_date	Dept
123	Agra	25-jan-2006	5000	25-jan-2026	Marketing
127	Mathura	22-dec-2006	6000	22-dec-2026	Finance
124	Agra	19-aug-2007	5500	19-aug-2027	Marketing
125	Delhi	14-apr-2004	8500	14-apr-2018	Sales
128	Pune	13-mar-2008	7500	13-mar-2028	Sales

(*a*) Show empno, name and salary of those who have Sports as hobby.

(*b*) Show name of the eldest employee.

(*c*) Show number of employee area wise.

(*d*) Show youngest employees from each Native place.

(*e*) Show Sno, Name, Hobby and Salary in descending order of Salary.

(*f*) Show the hobbies of those whose name pronounces as 'Abhay'.

(*g*) Show the appointment date and native place of those whose name starts with 'A' or ends in 'd'.

(*h*) Show the salary expense with suitable column heading of those who shall retire after 20-jan-2006.

(*i*) Show additional burden on the company in case salary of employees having hobby as sports, is increased by 10%.

(*j*) Show the hobby of which there are 2 or more employees.

(*k*) Show how many employee shall retire today if maximum length of service is 20 years.

(*l*) Show those employee name and date of birth who have served more than 17 years as on date.

(*m*) Show names of those who earn more than all of the employees of Sales dept.

(*n*) Increase salary of the employees by 5 % of their present salary with hobby as Music or they have completed atleast 3 years of service.

(*o*) Write the output of :

(*i*) Select distinct hobby from personal ;

(*ii*) Select avg(salary) from personal, job where Personal.Empno = Job.Sno and Area in ('Agra','Delhi') ;

(*iii*) Select count(distinct Native_place) from personal.

(*iv*) Select name, max(salary) from Personal, Job where Personal.Empno = Job.Sno ;

(*p*) Add a new tuple in the table Personal essentially with hobby as Music.

(*q*) Insert a new column email in Job table

(*r*) Create a table with values of columns empno, name, and hobby.

(*s*) Create a view of Personal and Job details of those who have served less than 15 years.

(*t*) Erase the records of employee from Job table whose hobby is not Sports.

(*u*) Remove the table Personal.

Path Wala

15. With reference to the table below, answer the questions that follow :

Table : Employees

Empid	Firstname	Lastname	Address	City
010	Ravi	Kumar	Raj nagar	GZB
105	Harry	Waltor	Gandhi nagar	GZB
152	Sam	Tones	33 Elm St.	Paris
215	Sarah	Ackerman	440 U.S. 110	Upton
244	Manila	Sengupta	24 Friends street	New Delhi
300	Robert	Samuel	9 Fifth Cross	Washington
335	Ritu	Tondon	Shastri Nagar	GZB
400	Rachel	Lee	121 Harrison St.	New York
441	Peter	Thompson	11 Red Road	Paris

Table : EmpSalary

Empid	Salary	Benefits	Designation
010	75000	15000	Manager
105	65000	15000	Manager
152	80000	25000	Director
215	75000	12500	Manager
244	50000	12000	Clerk
300	45000	10000	Clerk
335	40000	10000	Clerk
400	32000	7500	Salesman
441	28000	7500	Salesman

Write the SQL commands for the following using above tables :

(*i*) To show firstname, lastname, address and city of all employees living in Pairs

(*ii*) To display the content of Employees table in descending order of Firstname.

(*iii*) To display the firstname, lastname and total salary of all managers from the tables Employes and EmpSalary, where total salary is calculated as Salary + Benefits.

(*iv*) To display the maximum salary among managers and clerks from the table EmpSalary.

Give the Output of following SQL commands :

(*i*) Select firstname, Salary from Employees, Empsalary where Designation = 'Salesman' and Employees.Empid = Empsalary.Empid ;

(*ii*) Select count(distinct designation) from EmpSalary ;

(*iii*) Select designation, sum(salary) from EmpSalary group by designation having count(*) >2 ;

(*iv*) Select sum(Benefits) from EmpSalary where Designation = 'Clerk' ;

Path Wala

16. Show the average salary for all departments with more than 3 people for a job.
17. Display only the jobs with maximum salary greater than or equal to 3000.
18. Find out number of employees having "Manager" as Job.
19. List the count of employees grouped by deptno. (table EMPL)
20. List the sum of employees' salaries grouped by department. (table EMPL)
21. List the maximum salary of employee grouped by their department number.

Consider the tables Customers, Parts and Orders given in solved problems. Answer Questions 22 to 24 based on these :

22. List the total of customers' orders grouped by customer (id).
23. List the sum of the totals of orders grouped by customer and state.
24. List the customers (name) and the total amount of all their orders.

Consider tables EMPL, Dept, SalaryGrade. The Empl table has already been listed in earlier chapters. Schemas of tables SalaryGrade and Dept are being shown below :

```
SALARYGRADE (Lowsal, Highsal, Grade)
Dept (Deptno, DeptName, Location)
```

Answer Questions 25 and 26 on the basis of these tables :

25. List the department names and the number of their employees.
26. List the employee names and the name of their departments.