Experiment No:5

Aim: Study the use of order by, group by, commit, rollback.

Theory:

Order by:

The SQL **ORDER BY** clause is used to sort the data in ascending or descending order, based on one or more columns. Some database sorts query results in ascending order by default.

Syntax:

The basic syntax of ORDER BY clause is as follows:

```
SELECT column-list
FROM table_name
[WHERE condition]
[ORDER BY column1, column2, .. columnN] [ASC | DESC];
```

You can use more than one column in the ORDER BY clause. Make sure whatever column you are using to sort, that column should be in column-list.

Group by:

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

The GROUP BY clause follows the WHERE clause in a SELECT statement and precedes the ORDER BY clause.

Syntax:

The basic syntax of GROUP BY clause is given below. The GROUP BY clause must follow the conditions in the WHERE clause and must precede the ORDER BY clause if one is used.

```
SELECT column1, column2
FROM table_name
WHERE [ conditions ]
GROUP BY column1, column2
ORDER BY column1, column2
```

Transactions:

A transaction is a unit of work that is performed against a database. Transactions are units or sequences of work accomplished in a logical order, whether in a manual fashion by a user or automatically by some sort of a database program.

A transaction is the propagation of one or more changes to the database. For example, if you are creating a record or updating a record or deleting a record from the table, then you are performing transaction on the table. It is important to control transactions to ensure data integrity and to handle database errors.

Practically, you will club many SQL queries into a group and you will execute all of them together as a part of a transaction.

Properties of Transactions:

Transactions have the following four standard properties, usually referred to by the acronym ACID:

- **Atomicity:** ensures that all operations within the work unit are completed successfully; otherwise, the transaction is aborted at the point of failure, and previous operations are rolled back to their former state.
- **Consistency:** ensures that the database properly changes states upon a successfully committed transaction.
- **Isolation:** enables transactions to operate independently of and transparent to each other.
- **Durability:** ensures that the result or effect of a committed transaction persists in case of a system failure.

Transaction Control:

There are following commands used to control transactions:

- **COMMIT:** to save the changes.
- **ROLLBACK:** to rollback the changes.
- **SAVEPOINT:** creates points within groups of transactions in which to ROLLBACK
- **SET TRANSACTION:** Places a name on a transaction.

Transactional control commands are only used with the DML commands INSERT, UPDATE and DELETE only. They can not be used while creating tables or dropping them because these operations are automatically commited in the database.

The COMMIT Command:

The COMMIT command is the transactional command used to save changes invoked by a transaction to the database.

The COMMIT command saves all transactions to the database since the last COMMIT or ROLLBACK command.

The syntax for COMMIT command is as follows:

```
COMMIT;
```

Example:

Consider the CUSTOMERS table having the following records:

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

Following is the example which would delete records from the table having age = 25 and then COMMIT the changes in the database.

```
SQL> DELETE FROM CUSTOMERS
     WHERE AGE = 25;
SQL> COMMIT;
```

As a result, two rows from the table would be deleted and SELECT statement would produce the following result:

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

The ROLLBACK Command:

The ROLLBACK command is the transactional command used to undo transactions that have not already been saved to the database.

The ROLLBACK command can only be used to undo transactions since the last COMMIT or ROLLBACK command was issued.

The syntax for ROLLBACK command is as follows:

```
ROLLBACK;
```

Example:

Consider the CUSTOMERS table having the following records:

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

Following is the example, which would delete records from the table having age = 25 and then ROLLBACK the changes in the database.

```
SQL> DELETE FROM CUSTOMERS
     WHERE AGE = 25;
SQL> ROLLBACK;
```

As a result, delete operation would not impact the table and SELECT statement would produce the following result:

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

The SAVEPOINT Command:

A SAVEPOINT is a point in a transaction when you can roll the transaction back to a certain point without rolling back the entire transaction.

The syntax for SAVEPOINT command is as follows:

```
SAVEPOINT SAVEPOINT_NAME;
```

This command serves only in the creation of a SAVEPOINT among transactional statements. The ROLLBACK command is used to undo a group of transactions.

The syntax for rolling back to a SAVEPOINT is as follows:

```
ROLLBACK TO SAVEPOINT_NAME;
```

Following is an example where you plan to delete the three different records from the CUSTOMERS table. You want to create a SAVEPOINT before each delete, so that you can ROLLBACK to any SAVEPOINT at any time to return the appropriate data to its original state:

Example:

Consider the CUSTOMERS table having the following records:

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

Now, here is the series of operations:

```
SQL> SAVEPOINT SP1;
Savepoint created.
SQL> DELETE FROM CUSTOMERS WHERE ID=1;
1 row deleted.
SQL> SAVEPOINT SP2;
Savepoint created.
SQL> DELETE FROM CUSTOMERS WHERE ID=2;
1 row deleted.
SQL> SAVEPOINT SP3;
Savepoint created.
SQL> DELETE FROM CUSTOMERS WHERE ID=3;
1 row deleted.
```

Now that the three deletions have taken place, say you have changed your mind and decided to ROLLBACK to the SAVEPOINT that you identified as SP2. Because SP2 was created after the first deletion, the last two deletions are undone:

```
SQL> ROLLBACK TO SP2;
Rollback complete.
```

Notice that only the first deletion took place since you rolled back to SP2:

```
SQL> SELECT * FROM CUSTOMERS;
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
6 rows selected.
```

The RELEASE SAVEPOINT Command:

The RELEASE SAVEPOINT command is used to remove a SAVEPOINT that you have created.

The syntax for RELEASE SAVEPOINT is as follows:

```
RELEASE SAVEPOINT SAVEPOINT_NAME;
```

Once a SAVEPOINT has been released, you can no longer use the ROLLBACK command to undo transactions performed since the SAVEPOINT.

Code:

```
mysql> use exp1a
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----------------+
| Tables_in_exp1a |
+-----------------+
| CGPA            |
| compare         |
| customers       |
| employee        |
| exp2            |
| instructor      |
| orders          |
| student         |
| sv              |
| t               |
+-----------------+
10 rows in set (0.00 sec)

mysql> select * from instructor;
+-------+------------+-----------+--------+
| id    | name       | dept_name | salary |
+-------+------------+-----------+--------+
| 10101 | Srinivasan | Comp.Sci  |  65000 |
| 12121 | Wu         | Finance   |  90000 |
| 15151 | Mozart     | Music     |  40000 |
| 22222 | Einstein   | Physics   |  95000 |
| 32343 | El Said    | History   |  60000 |
| 33456 | Gold       | Physics   |  87000 |
| 45565 | Katz       | Comp.Sci  |  75000 |
+-------+------------+-----------+--------+
7 rows in set (0.01 sec)

mysql> insert into instructor values
(58583,"Califieri","History",62000),(76543,"Singh","Finance",80000),
(76766,"Crick","Biology",72000),(83821,"Brandt","Comp.Sci",92000),(9
8345,"Kim","Elec.Eng.",80000);
Query OK, 5 rows affected (0.00 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> select * from instructor;
+-------+------------+-----------+--------+
| id    | name       | dept_name | salary |
+-------+------------+-----------+--------+
| 10101 | Srinivasan | Comp.Sci  |  65000 |
| 12121 | Wu         | Finance   |  90000 |
| 15151 | Mozart     | Music     |  40000 |
| 22222 | Einstein   | Physics   |  95000 |
| 32343 | El Said    | History   |  60000 |
| 33456 | Gold       | Physics   |  87000 |
| 45565 | Katz       | Comp.Sci  |  75000 |
| 58583 | Califieri  | History   |  62000 |
| 76543 | Singh      | Finance   |  80000 |
| 76766 | Crick      | Biology   |  72000 |
| 83821 | Brandt     | Comp.Sci  |  92000 |
| 98345 | Kim        | Elec.Eng. |  80000 |
+-------+------------+-----------+--------+
12 rows in set (0.00 sec)

mysql> show tables;
+-----------------+
| Tables_in_exp1a |
+-----------------+
| CGPA            |
| compare         |
| customers       |
| employee        |
| exp2            |
| instructor      |
| orders          |
| student         |
| sv              |
| t               |
+-----------------+
10 rows in set (0.01 sec)

mysql> create table takes(ID int,course_ID varchar(20),sec_id
int,semester varchar(20),year int);
Query OK, 0 rows affected (0.01 sec)

mysql> insert into takes values (10101,"CS-
101",1,"Fall",2009),(10101,"CS-315",1,"Fall",2010),(10101,"CS-
347",1,"Fall",2009),(12121,"FN-201",1,"Spring",2010),(15151,"MU-
199",1,"Spring",2010),(22222,"PHY-101",1,"Fall",2009),(32343,"HIS-
351",1,"Spring",2010),(45565,"CS-101",1,"Spring",2010),(45565,"CS-
319",1,"Spring",2010),(76766,"BIO-101",1,"Summer",2009),(76766,"BIO-
301",1,"Summer",2010),(83821,"CS-190",1,"Spring",2009),(83821,"CS-
190",2,"Spring",2009),(83821,"CS-319",2,"Spring",2010),(98345,"EE-
181",1,"Spring",2009);
Query OK, 15 rows affected (0.00 sec)
Records: 15  Duplicates: 0  Warnings: 0

mysql> select * from takes;
+-------+-----------+--------+----------+------+
| ID    | course_ID | sec_id | semester | year |
+-------+-----------+--------+----------+------+
| 10101 | CS-101    |      1 | Fall     | 2009 |
| 10101 | CS-315    |      1 | Fall     | 2010 |
| 10101 | CS-347    |      1 | Fall     | 2009 |
| 12121 | FN-201    |      1 | Spring   | 2010 |
| 15151 | MU-199    |      1 | Spring   | 2010 |
| 22222 | PHY-101   |      1 | Fall     | 2009 |
| 32343 | HIS-351   |      1 | Spring   | 2010 |
| 45565 | CS-101    |      1 | Spring   | 2010 |
| 45565 | CS-319    |      1 | Spring   | 2010 |
| 76766 | BIO-101   |      1 | Summer   | 2009 |
| 76766 | BIO-301   |      1 | Summer   | 2010 |
| 83821 | CS-190    |      1 | Spring   | 2009 |
| 83821 | CS-190    |      2 | Spring   | 2009 |
| 83821 | CS-319    |      2 | Spring   | 2010 |
| 98345 | EE-181    |      1 | Spring   | 2009 |
+-------+-----------+--------+----------+------+
15 rows in set (0.01 sec)

mysql> select dept_name,avg(salary) as avg_salary
    -> from instructor
    -> group by dept_name;
+-----------+------------+
| dept_name | avg_salary |
+-----------+------------+
| Biology   | 72000.0000 |
| Comp.Sci  | 77333.3333 |
| Elec.Eng. | 80000.0000 |
| Finance   | 85000.0000 |
| History   | 61000.0000 |
| Music     | 40000.0000 |
| Physics   | 91000.0000 |
+-----------+------------+
7 rows in set (0.00 sec)

mysql> select dept_name,avg(salary) as avg_salary from instructor
group by dept_name having avg(salary)>61000;
+-----------+------------+
| dept_name | avg_salary |
+-----------+------------+
| Biology   | 72000.0000 |
| Comp.Sci  | 77333.3333 |
| Elec.Eng. | 80000.0000 |
| Finance   | 85000.0000 |
| Physics   | 91000.0000 |
+-----------+------------+
5 rows in set (0.00 sec)

mysql> commit;
Query OK, 0 rows affected (0.00 sec)

mysql> rollback;
Query OK, 0 rows affected (0.00 sec)

mysql> commit;
Query OK, 0 rows affected (0.00 sec)


151070031@ltsp85:~$ mysql -h 172.18.61.59 -u 151070031
ERROR 1045 (28000): Access denied for user
'151070031'@'172.18.39.85' (using password: NO)
151070031@ltsp85:~$ mysql -h 172.18.61.59 -u 151070031 -p151070031
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1079
```

```
Server version: 5.5.44-0ubuntu0.14.04.1 (Ubuntu)

Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights
reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
statement.

mysql> use exp1a
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from instructor;
+-------+------------+-----------+--------+
| id    | name       | dept_name | salary |
+-------+------------+-----------+--------+
| 10101 | Srinivasan | Comp.Sci  |  65000 |
| 12121 | Wu         | Finance   |  90000 |
| 15151 | Mozart     | Music     |  40000 |
| 22222 | Einstein   | Physics   |  95000 |
| 32343 | El Said    | History   |  60000 |
| 33456 | Gold       | Physics   |  87000 |
| 45565 | Katz       | Comp.Sci  |  75000 |
| 58583 | Califieri  | History   |  62000 |
| 76543 | Singh      | Finance   |  80000 |
| 76766 | Crick      | Biology   |  72000 |
| 83821 | Brandt     | Comp.Sci  |  92000 |
| 98345 | Kim        | Elec.Eng. |  80000 |
+-------+------------+-----------+--------+
12 rows in set (0.01 sec)

mysql> select * from takes;
+-------+-----------+--------+----------+------+
| ID    | course_ID | sec_id | semester | year |
+-------+-----------+--------+----------+------+
| 10101 | CS-101    |      1 | Fall     | 2009 |
| 10101 | CS-315    |      1 | Fall     | 2010 |
| 10101 | CS-347    |      1 | Fall     | 2009 |
| 12121 | FN-201    |      1 | Spring   | 2010 |
| 15151 | MU-199    |      1 | Spring   | 2010 |
| 22222 | PHY-101   |      1 | Fall     | 2009 |
| 32343 | HIS-351   |      1 | Spring   | 2010 |
| 45565 | CS-101    |      1 | Spring   | 2010 |
| 45565 | CS-319    |      1 | Spring   | 2010 |
| 76766 | BIO-101   |      1 | Summer   | 2009 |
| 76766 | BIO-301   |      1 | Summer   | 2010 |
| 83821 | CS-190    |      1 | Spring   | 2009 |
| 83821 | CS-190    |      2 | Spring   | 2009 |
| 83821 | CS-319    |      2 | Spring   | 2010 |
| 98345 | EE-181    |      1 | Spring   | 2009 |
+-------+-----------+--------+----------+------+
15 rows in set (0.01 sec)

mysql> select * from takes order by ID asc,course_ID desc;
+-------+-----------+--------+----------+------+
| ID    | course_ID | sec_id | semester | year |
+-------+-----------+--------+----------+------+
| 10101 | CS-347    |      1 | Fall     | 2009 |
| 10101 | CS-315    |      1 | Fall     | 2010 |
| 10101 | CS-101    |      1 | Fall     | 2009 |
| 12121 | FN-201    |      1 | Spring   | 2010 |
| 15151 | MU-199    |      1 | Spring   | 2010 |
| 22222 | PHY-101   |      1 | Fall     | 2009 |
| 32343 | HIS-351   |      1 | Spring   | 2010 |
| 45565 | CS-319    |      1 | Spring   | 2010 |
| 45565 | CS-101    |      1 | Spring   | 2010 |
| 76766 | BIO-301   |      1 | Summer   | 2010 |
| 76766 | BIO-101   |      1 | Summer   | 2009 |
| 83821 | CS-319    |      2 | Spring   | 2010 |
| 83821 | CS-190    |      1 | Spring   | 2009 |
| 83821 | CS-190    |      2 | Spring   | 2009 |
| 98345 | EE-181    |      1 | Spring   | 2009 |
+-------+-----------+--------+----------+------+
15 rows in set (0.01 sec)

mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> delete from takes where ID = 98345;
Query OK, 1 row affected (0.00 sec)

mysql> rollback;
Query OK, 0 rows affected (0.01 sec)

mysql> select * from takes order by ID asc,course_ID desc;
+-------+-----------+--------+----------+------+
| ID    | course_ID | sec_id | semester | year |
+-------+-----------+--------+----------+------+
| 10101 | CS-347    |      1 | Fall     | 2009 |
| 10101 | CS-315    |      1 | Fall     | 2010 |
| 10101 | CS-101    |      1 | Fall     | 2009 |
| 12121 | FN-201    |      1 | Spring   | 2010 |
| 15151 | MU-199    |      1 | Spring   | 2010 |
| 22222 | PHY-101   |      1 | Fall     | 2009 |
| 32343 | HIS-351   |      1 | Spring   | 2010 |
| 45565 | CS-319    |      1 | Spring   | 2010 |
| 45565 | CS-101    |      1 | Spring   | 2010 |
| 76766 | BIO-301   |      1 | Summer   | 2010 |
| 76766 | BIO-101   |      1 | Summer   | 2009 |
| 83821 | CS-319    |      2 | Spring   | 2010 |
| 83821 | CS-190    |      1 | Spring   | 2009 |
| 83821 | CS-190    |      2 | Spring   | 2009 |
| 98345 | EE-181    |      1 | Spring   | 2009 |
+-------+-----------+--------+----------+------+
15 rows in set (0.00 sec)
```

Conclusion:

Thus,

1. The use of group by clause for aggregate functions and order by clause for (natural) ordering were understood and implemented.
2. The concept of transactions, commit, rollback, etc. were studied, understood & implemented.