MySQL Queries

Basic Keywords

Sl.No	Keyword	Description
01.	SELECT	Used to state which columns to query. Use * for all
02.	FROM	Declares which table/view etc to select from
03.	WHERE	Introduces a condition
04.	=	Used for comparing a value to a specified input
05.	LIKE	Special operator used with the WHERE clause to search for a specific pattern in a column
06.	GROUP BY	Arranges identical data into groups
07.	HAVING	Specifies that only rows where aggregate values meet the specified conditions should be returned. Used because the WHERE keyword cannot be used with aggregate functions
08.	INNER JOIN	Returns all rows where key record of one table is equal to key records of another

Sl.No	Keyword	Description
09.	LEFT JOIN	Returns all rows from the 'left' (1st) table with the matching rows in the right (2nd)
10.	RIGHT JOIN	Returns all rows from the 'right' (2nd) table with the matching rows in the left (1st)
11.	FULL OUTER JOIN	Returns rows that match either in the left or right table

Reporting Aggregate functions

In database management, an aggregate function is a function where the values of multiples rows are grouped to form a single value.

Sl.No	Function	Description
01.	COUNT	Return the number of rows in a certain table/view
02.	SUM	Accumulate the values
03.	AVG	Returns the average for a group of values
04.	MIN	Returns the smallest value of the group
05.	MAX	Returns the largest value of the group

Querying data from a table

Sl.No	Query	Description
01.	SELECT c1 FROM t	Select data in column c1 from a table named t
02.	SELECT * FROM t	Select all rows and columns from a table named t
03.	SELECT c1 FROM t WHERE c1 = 'test'	Select data in column c1 from a table named t where the value in c1 = 'test'
05.	SELECT c1 FROM t ORDER BY c1 ASC (DESC)	Select data in column c1 from a table name t and order by c1, either in ascending (default) or descending order

Sl.No	Query	Description
07.	SELECT c1 FROM t ORDER BY c1LIMIT n OFFSET offset	Select data in column c1 from a table named t and skip offset of rows and return the next n rows
08.	SELECT c1, aggregate(c2) FROM t GROUP BY c1	Select data in column c1 from a table named t and group rows using an aggregate function
09.	SELECT c1, aggregate(c2) FROM t GROUP BY c1 HAVING condition	Select data in column c1 from a table named t and group rows using an aggregate function and filter these groups using 'HAVING' clause

Querying data from multiple tables

Sl.No	Query	Description
01.	SELECT c1, c2 FROM t1 INNER JOIN t2 on condition	Select columns c1 and c2 from a table named t1 and perform an inner join between t1 and t2
02.	SELECT c1, c2 FROM t1 LEFT JOIN t2 on condition	Select columns c1 and c2 from a table named t1 and perform a left join between t1 and t2
03.	SELECT c1, c2 FROM t1 RIGHT JOIN t2 on condition	Select columns c1 and c2 from a table named t1 and perform a right join between t1 and t2
04.	SELECT c1, c2 FROM t1 FULL OUTER JOIN t2 on condition	Select columns c1 and c2 from a table named t1 and perform a full outer join between t1 and t2
05.	SELECT c1, c2 FROM t1 CROSS JOIN t2	Select columns c1 and c2 from a table named t1 and produce a Cartesian product of rows in tables
06.	SELECT c1, c2 FROM t1, t2	Select columns c1 and c2 from a table named t1 and produce a Cartesian product of rows in tables

Sl.No	Query	Description
07.	SELECT c1, c2 FROM t1 A INNER JOIN t2 B on condition	Select columns c1 and c2 from a table named t1 and joint it to itself using an INNER JOIN clause

Using SQL Operators

Sl.No	Query	Description
01.	SELECT c1 FROM t1 UNION [ALL] SELECT c1 FROM t2	Select column c1 from a table named t1 and column c1 from a table named t2 and combine the rows from these two queries
02.	SELECT c1 FROM t1 INTERSECT SELECT c1 FROM t2	Select column c1 from a table named t1 and column c1 from a table named t2 and return the intersection of two queries
03.	SELECT c1 FROM t1 MINUS SELECT c1 FROM t2	Select column c1 from a table named t1 and column c1 from a table named t2 and subtract the 2nd result set from the 1st
04.	SELECT c1 FROM t WHERE c1 [NOT] LIKE pattern	Select column c1 from a table named t and query the rows using pattern matching %
05.	SELECT c1 FROM t WHERE c1 [NOT] in test_list	Select column c1 from a table name t and return the rows that are (or are not) in test_list
06.	SELECT c1 FROM t WHERE c1 BETWEEN min AND max	Select column c1 from a table named t and return the rows where c1 is between min and max
07.	SELECT c1 FROM t WHERE c1 IS [NOT] NULL	Select column c1 from a table named t and check if the values are NULL or not

Data modification

Sl.No	Query	Description
01.	INSERT INTO t(column_list) VALUES(value_list)	Insert one row into a table named t

Sl.No	Query	Description
02.	INSERT INTO t(column_list) VALUES (value_list), (value_list), ...	Insert multiple rows into a table named t
03.	INSERT INTO t1(column_list) SELECT column_list FROM t2	Insert rows from t2 into a table named t1
04.	UPDATE tSET c1 = new_value	Update a new value in table t in the column c1 for all rows
05.	UPDATE tSET c1 = new_value, c2 = new_value WHERE condition	Update values in column c1 and c2 in table t that match the condition
06.	DELETE FROM t	Delete all the rows from a table named t
07.	DELETE FROM tWHERE condition	Delete all rows from that a table named t that match a certain condition

Views

A view is a virtual table that is a result of a query. They can be extremely useful and are often used as a security mechanism, letting users access the data through the view, rather than letting them access the underlying base table:

Sl.No	Query	Description
01.	CREATE VIEW view1 AS SELECT c1, c2 FROM t1 WHERE condition	Create a view, comprising of columns c1 and c2 from a table named t1 where a certain condition has been met.

Indexes

An index is used to speed up the performance of queries by reducing the number of database pages that have to be visited:

Sl.No	Query	Description
01.	CREATE INDEX index_nameON t(c1, c2)	Create an index on columns c1 and c2 of the table t
02.	CREATE UNIQUE INDEX index_name ON t(c3, c4)	Create a unique index on columns c3 and c4 of the table t

Sl.No	Query	Description
03.	DROP INDEX index_name	Drop an index

Stored Procedure

A stored procedure is a set of SQL statements with an assigned name that can then be easily reused and share by multiple programs:

Sl.No	Query	Description
01.	CREATE PROCEDURE procedure_name @variable AS datatype = value AS -- Comments SELECT * FROM tGO	Create a procedure called procedure_name, create a local variable and then select from table t

Triggers

A trigger is a special type of stored procedure that automatically executes when a user tries to modify data through a DML event (data manipulation language). A DML event is an INSERT, UPDATE or DELETE statement on a table or view:

```
CREATE OR MODIFY TRIGGER trigger_name
WHEN EVENT

ON table_name TRIGGER_TYPE

EXECUTE stored_procedure
```

WHEN:

- BEFORE – invoke before the event occurs
- AFTER – invoke after the event occurs

EVENT:

- INSERT – invoke for insert
- UPDATE – invoke for update
- DELETE – invoke for delete

TRIGGER_TYPE:

- FOR EACH ROW

- FOR EACH STATEMENT

```
!-- Delete a specific trigger
DROP TRIGGER trigger_name
```