

**DBMS SKILL-6
PRELAB**

1. How to use Auto Increment in SQL?

DBMS Skill-6190031187

1. Auto-increment allows a unique number to be generated automatically when a new record is inserted into a table.

Often this is the primary key field that we would like to be created automatically every time a new record is inserted.

Example:-

```
create table persons ( personid int NOT NULL  
    AUTO-INCREMENT, lastname varchar(255),  
    firstname varchar(255), age int,  
    primary key (personid) );
```

My SQL uses AUTO-INCREMENT keyword to perform an Auto-increment feature.

By default, the starting value for AUTO-INCREMENT is 1, and it will increment by 1 for each new record.

2. What is embedded and dynamic SQL?

2. Embedded / static SQL

It is those SQL statements that are fixed and can't be changed at runtime in an application. These statements are compiled at the compile-time only.

The benefit of using this statement is that you know the path of execution of statements because you have the SQL statements with you, so you can optimize your SQL query and can execute the query in the best and fastest possible way. The way of accessing data is predefined and these static SQL statements are generally used on those databases that are uniformly distributed.

Dynamic SQL

These are those SQL statements that are created or executed at run-time. The users can execute their own query in some application. These statements are compiled at run-time. These kind of SQL statements are used when there is a non-uniformity in the data stored in the database. It is more flexible as



compared to static SQL.

3. What is meant by ALIAS in SQL?

3. Aliases are the temporary name given to table or column for the purpose of a particular SQL query. It is used when name of column or table is used other than their original names, but the modified name is only temporary.

→ Aliases are created to make table or columns names more readable

→ Aliases are useful when table or column names are big or not very readable.

→ These are preferred when there are more than one table involved in a query.

6 Which operator has the highest precedence among the following – AND, NOT, OR?

6. Highest precedence among

AND, NOT, OR is

" NOT "

4 While executing certain commands Mr. Jack is confused to decide whether View is a logical storage or physical storage. State him an appropriate solution with a valid reason?

view is a logical storage:-
4. Views are a special version of tables in SQL. They provide a virtual table environment for various complex operations. You can select data from multiple tables, or you can select specific data based on certain criteria in views. It does not hold actual data; it holds only the definition of the view in the data dictionary.

It does not use physical memory only the query is stored in the data dictionary, it is computed dynamically whenever the user performs any query on it. Changes made at any point in view are reflected in the actual base table.

The view has primarily two purposes:-

- 1) simplify complex SQL queries
- 2) provides restriction to users from accessing sensitive data

5 How to build authentication to a database?

5. (1) Your database should now be listed on the left with your other database schemas
- (2) click Home icon in the top left corner to return to workbench central screen. click on MySQL server instance under the server Administrator section of MySQL workbench to create a new database user and assign privileges to your new database
- (3) click on users and privileges. Then click on Add account. Enter login name for the new user, type localhost and new password as shown. click apply to create the new user account.
- (4) To assign privileges for this user to access a specific database, click on schema privilege tab. click the user account from list of users on the left. click on Add entry button
- (5) select the selected schema radio button, and your database schema from the list
- (6) select appropriate privileges to allow the user access the selected database. click save changes to complete your new user setup
- (7) you can now test your new user logic by using MySQL workbench with your newly created user account.



7 What is DEFAULT?

7. The MySQL DEFAULT keyword is a database constraint or rule that is applied when inserting new records into a table. If the value is not provided to any column while inserting a new row in the table, the default value will be used instead.



Scanned with
CamScanner

INLAB

Implement SQL Queries on Case Study 8 (SAINT GOBAIN)

QUOTATION:

Cust_ID	Cust_Name	Cust_Phone	Glass_Type	Glass_thick	Glass_Measure	Glass_color	Address	Exp_Amt	Advance_Paid
1	Raju	8767895698	Clear glass	4MM	140CM	Black	Hyd	10000	2000
2	Hari	9999887766	Mirror	5MM	150CM	Blue	Delhi	11000	2500
3	Arun	7567896546	Clear glass	6MM	120CM	Black	Mumbai	9000	1000
4	Kiran	6754567890	Mirror	3MM	200CM	Blue	Hyd	20000	5000
5	Chand	7164567897	Mirror	4MM	120CM	Blue	Hyd	15000	6000

BILL:

Bill_ID	Cust_Name	Cust_Phone	Address	Glass_Feature	Mode_Pay
100	Raju	8767895698	Hyd	Good	Cash
101	Hari	9999887766	Delhi	Good	Credit
102	Arun	7567896546	Mumbai	Good	Cash
103	Kiran	6754567890	Hyd	Good	Cash
104	Chand	7164567897	Hyd	Good	Cash

1) Create tables with the required constraints for the given case study

QUOTATION

```
create table quotation(cust_id int primary key,cust_name varchar(30),cust_phone
bigint,glass_type varchar(30),glass_thick varchar(10),glass_measure varchar(10),glass_color
varchar(10),address varchar(50),exp_amt int,advance_paid int);
```

BILL

```
create table bill(bill_id int primary key,cust_name varchar(30),cust_phone bigint,
address varchar(50),glass_feature varchar(20),mode_pay varchar(20));
```

2) Insert 10 records into the created tables

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following code:

```

14 insert into quotation values(7,'Hohith',9988556644,'Mirror','SMH','150CM','Blue','Delhi',11000,2500);
15 insert into quotation values(8,'Pavan',9856423183,'Clear glass','GMH','120CM','Black','Mumbai',9000,1800);
16 insert into quotation values(9,'Navneet',7894561233,'Mirror','SMH','200CM','Blue','Hyd',20000,5000);
17 insert into quotation values(10,'Rithik',9874563910,'Mirror','4WH','120CM','Blue','Hyd',10000,6000);
18
19 select * from quotation;

```

The Result Grid shows the following data:

cust_id	cust_name	cust_phone	glass_type	glass_thick	glass_measure	glass_color	address	exp_amt	advance_paid
7	Raju	8767895698	Clear glass	4MM	140CM	Black	Hyd	10000	2000
8	Hari	9999887766	Mirror	5MM	150CM	Blue	Delhi	11000	2500
9	Arun	7567896546	Clear glass	6MM	120CM	Black	Mumbai	9000	1800
10	Kiran	6754567890	Mirror	3MM	200CM	Blue	Hyd	20000	5000
11	Chand	6754567890	Mirror	4MM	120CM	Blue	Hyd	10000	6000
12	Sudheer	1122334455	Clear glass	4MM	140CM	Black	Hyd	10000	2000

The Output window shows the execution of the query, with a message indicating that 10 rows were returned.

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following code:

```

27 insert into bill values(100,'Hohith',9988556644,'Delhi','Good','Credit');
28 insert into bill values(101,'Pavan',9856423183,'Mumbai','Good','Cash');
29 insert into bill values(102,'Navneet',7894561233,'Hyd','Good','Cash');
30 insert into bill values(103,'Rithik',9874563910,'Hyd','Good','Cash');
31
32 select * from bill;

```

The Result Grid shows the following data:

bill_id	cust_name	cust_phone	address	glass_feature	mode_pay
100	Raju	8767895698	Hyd	Good	Cash
101	Hari	9999887766	Delhi	Good	Credit
102	Arun	7567896546	Mumbai	Good	Cash
103	Kiran	6754567890	Hyd	Good	Cash
104	Chand	6754567890	Hyd	Good	Cash
105	Sudheer	1122334455	Hyd	Good	Cash

The Output window shows the execution of the query, with a message indicating that 10 rows were returned.

3) Write a SQL query to find out Customer ID and Customer Name in ascending order.

select * from quotation order by cust_name,cust_id asc;

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following queries:

```

31
32 * select * from bill;
33
34 -- 3
35 * select * from quotation order by cust_name,cust_id asc;
36 -- 4

```

The Result Grid displays the following data:

cust_id	cust_name	cust_phone	glass_type	glass_thick	glass_measure	glass_color	address	exp_amt	advance_paid
3	Arun	7567896546	Clear glass	6MM	120CM	Black	Mumbai	9000	1000
5	Chand	6754567890	Mirror	4MM	120CM	Blue	Hyd	10000	6000
2	Hari	9999887766	Mirror	5MM	150CM	Blue	Delhi	11000	2500
4	Kiran	6754567890	Mirror	3MM	200CM	Blue	Hyd	20000	5000
7	Mohith	9988556644	Mirror	5MM	150CM	Blue	Delhi	11000	2500
9	Navneet	7894561233	Mirror	3MM	200CM	Blue	Hyd	20000	5000

The Output pane shows the following messages:

```

5 13:07:22 select content.sum(population) from world group by content having sum(population)>=1... 1 row(s) returned 0.047 sec / 0.000 sec
6 13:07:39 select content.sum(population) from world group by content having sum(population)>=1... 0 row(s) returned 0.000 sec / 0.000 sec
7 14:01:01 select * from quotation LIMIT 0, 1000 10 row(s) returned 0.078 sec / 0.000 sec
8 14:01:14 select * from bill LIMIT 0, 1000 10 row(s) returned 0.000 sec / 0.000 sec
9 14:01:33 select * from quotation order by cust_name,cust_id asc LIMIT 0, 1000 10 row(s) returned 0.000 sec / 0.000 sec

```

4) Find unique Customer Name in the Quotation table.

select distinct cust_name from quotation;

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following queries:

```

32 * select * from bill;
33
34 -- 3
35 * select * from quotation order by cust_name,cust_id asc;
36 -- 4
37 * select distinct cust_name from quotation;

```

The Result Grid displays the following data:

cust_name
Raju
Hari
Arun
Kiran
Chand
Sudheer

The Output pane shows the following messages:

```

6 13:07:39 select content.sum(population) from world group by content having sum(population)>=1... 0 row(s) returned 0.000 sec / 0.000 sec
7 14:01:01 select * from quotation LIMIT 0, 1000 10 row(s) returned 0.078 sec / 0.000 sec
8 14:01:14 select * from bill LIMIT 0, 1000 10 row(s) returned 0.000 sec / 0.000 sec
9 14:01:33 select * from quotation order by cust_name,cust_id asc LIMIT 0, 1000 10 row(s) returned 0.000 sec / 0.000 sec
10 14:02:00 select distinct cust_name from quotation LIMIT 0, 1000 10 row(s) returned 0.000 sec / 0.000 sec

```

5) SQL query to find the Glass Thickness where number of Customer of highest thickness.

6) Write a SQL query to find out the amount has to paid by the customer in Quotation Table.

select cust_name,exp_amt-advance_paid as amt_to_be_paid from quotation;

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following queries:

```

35 select * from quotation order by cust_name,cust_id asc;
36 -- 4
37 select distinct cust_name from quotation;
38 -- 5
39 -- 6
40 select cust_name,exp_amt-advance_paid as amt_to_be_paid from quotation;

```

The Result Grid shows the output of the last query:

cust_name	amt_to_be_paid
Raju	8000
Hari	8500
Arun	8000
Kiran	15000
Chand	4000
Sudheer	8000

The Output tab shows the execution log:

#	Time	Action	Message	Duration / Fetch
7	14:01:01	select * from quotation LIMIT 0, 1000	10 row(s) returned	0.078 sec / 0.000 sec
8	14:01:14	select * from bill LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
9	14:01:33	select * from quotation order by cust_name,cust_id asc LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
10	14:02:00	select distinct cust_name from quotation LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
11	14:02:19	select cust_name,exp_amt-advance_paid as amt_to_be_paid from quotation LIMIT 0, 1...	10 row(s) returned	0.000 sec / 0.000 sec

7) Write a SQL query to list the name of those whose name starts with 'A' in Quotation Table.
 select * from quotation where cust_name like 'A%';

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following queries:

```

40 select cust_name,exp_amt-advance_paid as amt_to_be_paid from quotation;
41 -- 7
42 select * from quotation where cust_name like 'A%';
43 -- 8
44 select * from bill where cust_phone like '%910';
45 -- 9

```

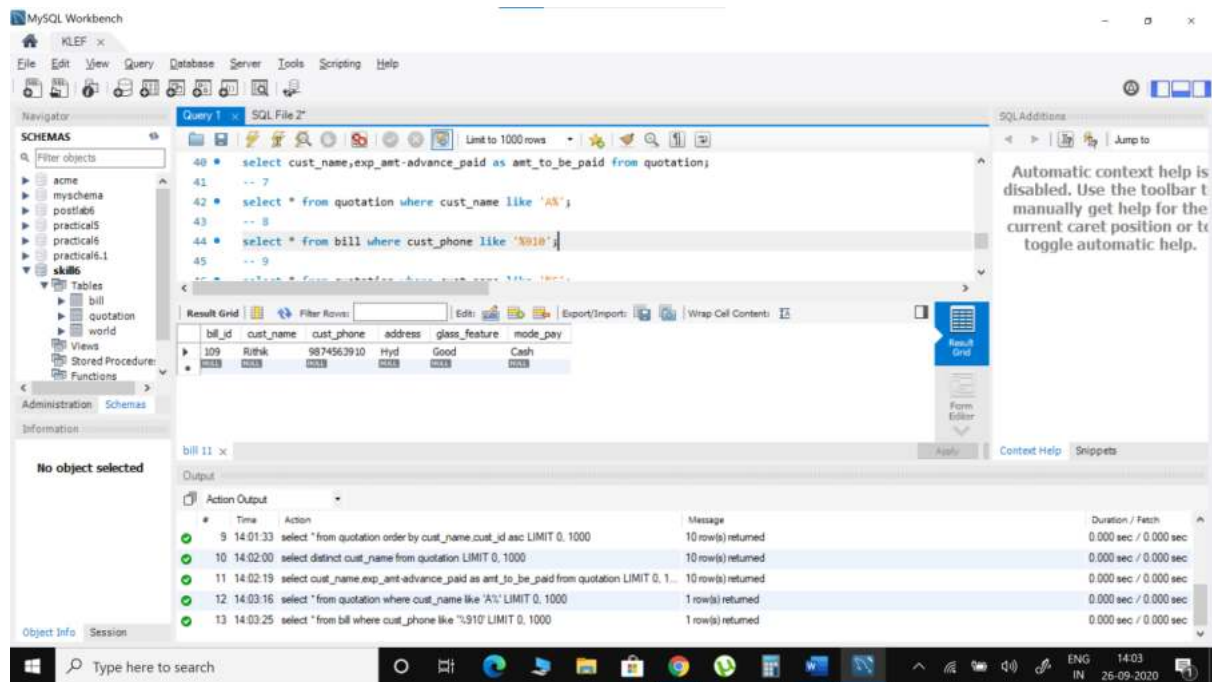
The Result Grid shows the output of the last query:

cust_id	cust_name	cust_phone	glass_type	glass_thick	glass_measure	glass_color	address	exp_amt	advance_paid
3	Arun	7567896546	Clear glass	6MM	120CM	Black	Mumbai	9000	1000

The Output tab shows the execution log:

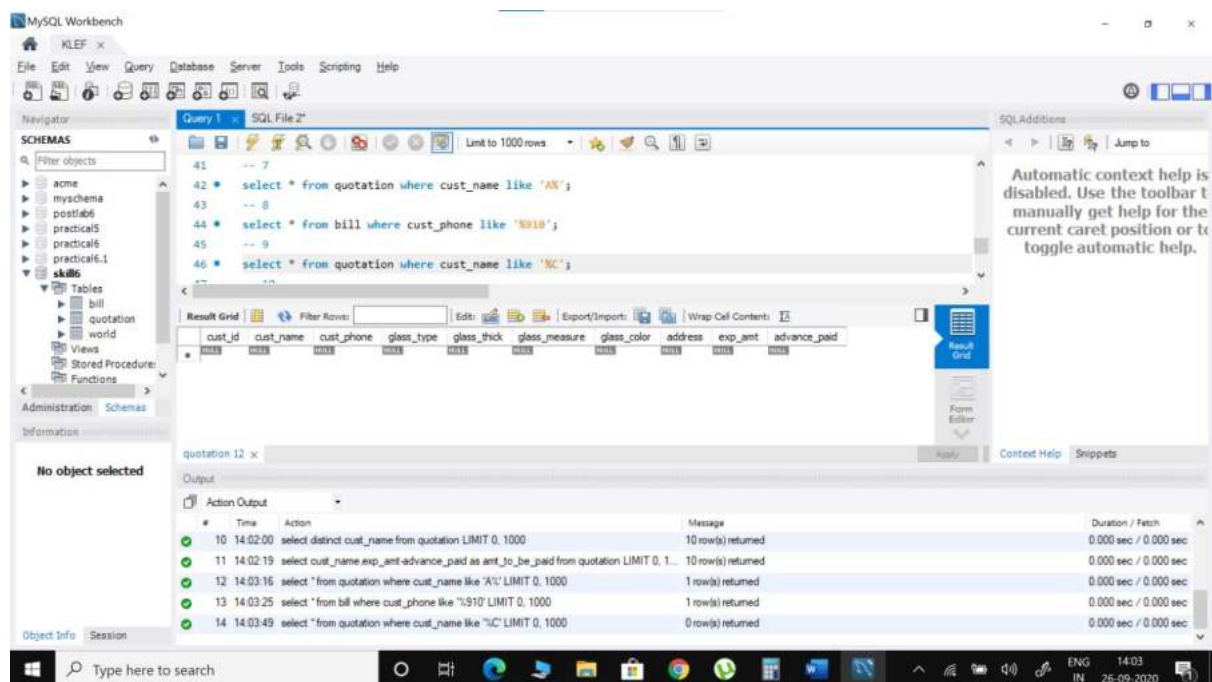
#	Time	Action	Message	Duration / Fetch
8	14:01:14	select * from bill LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
9	14:01:33	select * from quotation order by cust_name,cust_id asc LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
10	14:02:00	select distinct cust_name from quotation LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
11	14:02:19	select cust_name,exp_amt-advance_paid as amt_to_be_paid from quotation LIMIT 0, 1...	10 row(s) returned	0.000 sec / 0.000 sec
12	14:03:16	select * from quotation where cust_name like 'A%' LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

8) Write a SQL query to list the Phone no. that ends with "910" in the Bill table.
 select * from bill where cust_phone like '%910';



9) Write an SQL query to print details of the Customer whose Name ends with 'C'

select * from quotation where cust_name like '%C';



10) Write an SQL query to print details of the customers whose advance paid lies between 1000 and 2000.

select * from quotation where advance_paid between 1000 and 2000;

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following queries:

```

43 -- 8
44 select * from bill where cust_phone like "9310";
45 -- 9
46 select * from quotation where cust_name like "C";
47 -- 10
48 select * from quotation where advance_paid between 1000 and 2000;

```

The Result Grid shows the following data:

cust_id	cust_name	cust_phone	glass_type	glass_thick	glass_measure	glass_color	address	exp_amt	advance_paid
1	Raju	8767895698	Clear glass	4MM	140CM	Black	Hyd	10000	2000
3	Arun	756789546	Clear glass	6MM	120CM	Black	Mumbai	9000	1000
6	Sudheer	1122334455	Clear glass	4MM	140CM	Black	Hyd	10000	2000
8	Pavan	9856423183	Clear glass	6MM	120CM	Black	Mumbai	9000	1000

The Output pane shows the execution results of the queries:

#	Time	Action	Message	Duration / Fetch
11	14:02:19	select cust_name,exp_amt,advance_paid as amt_to_be_paid from quotation LIMIT 0, 1	10 row(s) returned	0.000 sec / 0.000 sec
12	14:03:16	select * from quotation where cust_name like "A" LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
13	14:03:25	select * from bill where cust_phone like "9310" LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
14	14:03:49	select * from quotation where cust_name like "C" LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
15	14:04:30	select * from quotation where advance_paid between 1000 and 2000 LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec

11) Write an SQL query to fetch the list of customers with the same Mode of payment.

select cust_name,mode_pay from bill where mode_pay in

(select distinct mode_pay from bill group by mode_pay having count(mode_pay)>1);

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following query:

```

46 select * from quotation where cust_name like "C";
47 -- 10
48 select * from quotation where advance_paid between 1000 and 2000;
49 -- 11
50 select cust_name,mode_pay from bill where mode_pay in
51 (select distinct mode_pay from bill group by mode_pay having count(mode_pay)>1);

```

The Result Grid shows the following data:

cust_name	mode_pay
Raju	Cash
Hari	Credit
Arun	Cash
Kiran	Cash
Chand	Cash
Sudheer	Cash

The Output pane shows the execution results of the queries:

#	Time	Action	Message	Duration / Fetch
12	14:03:16	select * from quotation where cust_name like "A" LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
13	14:03:25	select * from bill where cust_phone like "9310" LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
14	14:03:49	select * from quotation where cust_name like "C" LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
15	14:04:30	select * from quotation where advance_paid between 1000 and 2000 LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
16	14:04:54	select cust_name,mode_pay from bill where mode_pay in (select distinct mode_pay from bill group by mode_pay having count(mode_pay)>1)	10 row(s) returned	0.000 sec / 0.000 sec

12) Write an SQL query to fetch the Unique records in Quotation and Bill Table

select distinct * from bill;

select distinct * from quotation;

The image displays two screenshots of the MySQL Workbench interface, showing SQL queries and their results.

Top Screenshot:

- Query 1:**

```

select * from quotation where advance_paid between 1000 and 2000;
-- 11
select cust_name, mode_pay from bill where mode_pay in
(select distinct mode_pay from bill group by mode_pay having count(mode_pay) > 1);
-- 12
select distinct * from bill;

```
- Result Grid:**

bill_id	cust_name	cust_phone	address	glass_feature	mode_pay
100	Raju	8767895698	Hyd	Good	Cash
101	Hari	9999887766	Delhi	Good	Credit
102	Arun	7567896546	Mumbai	Good	Cash
103	Kiran	6754567890	Hyd	Good	Cash
104	Chand	6754567890	Hyd	Good	Cash
105	Sudheer	1122334455	Hyd	Good	Cash
- Output:**

#	Time	Action	Message	Duration / Fetch
13	14:03:25	select * from bill where cust_phone like "1510" LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
14	14:03:49	select * from quotation where cust_name like "LC" LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
15	14:04:30	select * from quotation where advance_paid between 1000 and 2000 LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
16	14:04:54	select cust_name, mode_pay from bill where mode_pay in (select distinct mode_pay fro...	10 row(s) returned	0.000 sec / 0.000 sec
17	14:05:11	select distinct * from bill LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec

Bottom Screenshot:

- Query 1:**

```

-- 11
select cust_name, mode_pay from bill where mode_pay in
(select distinct mode_pay from bill group by mode_pay having count(mode_pay) > 1);
-- 12
select distinct * from bill;
select distinct * from quotation;

```
- Result Grid:**

cust_id	cust_name	cust_phone	glass_type	glass_thick	glass_measure	glass_color	address	exp_amt	advance_paid
1	Raju	8767895698	Clear glass	4MM	140CM	Black	Hyd	10000	2000
2	Hari	9999887766	Mirror	5MM	150CM	Blue	Delhi	11000	2500
3	Arun	7567896546	Clear glass	6MM	120CM	Black	Mumbai	9000	1000
4	Kiran	6754567890	Mirror	3MM	200CM	Blue	Hyd	20000	5000
5	Chand	6754567890	Mirror	4MM	120CM	Blue	Hyd	10000	6000
6	Sudheer	1122334455	Clear glass	4MM	140CM	Black	Hyd	10000	2000
- Output:**

#	Time	Action	Message	Duration / Fetch
14	14:03:49	select * from quotation where cust_name like "LC" LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
15	14:04:30	select * from quotation where advance_paid between 1000 and 2000 LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
16	14:04:54	select cust_name, mode_pay from bill where mode_pay in (select distinct mode_pay fro...	10 row(s) returned	0.000 sec / 0.000 sec
17	14:05:11	select distinct * from bill LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
18	14:05:25	select distinct * from quotation LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec

POSTLAB

name	continent	area	population	gdp
Afghanistan	Asia	652230	25500100	20343000000
Albania	Europe	28748	2831741	12960000000
Algeria	Africa	2381741	37100000	188681000000
Andorra	Europe	468	78115	3712000000
Angola	Africa	1246700	20609294	100990000000

create table world (name varchar(50),contient varchar(50),area int,population bigint,gdp bigint);

insert into world values('Afghanistan','Asia',652230,2500100,20343000000);

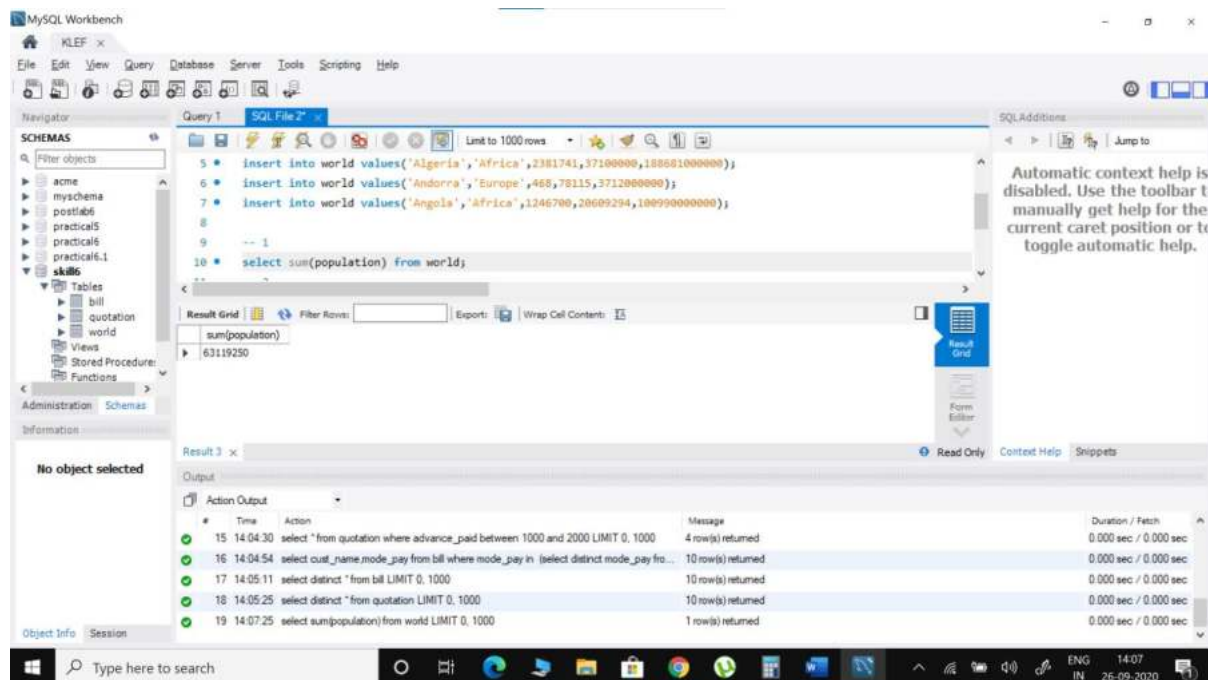
insert into world values('Albania','Europe',28748,2831741,12960000000);

insert into world values('Algeria','Africa',2381741,37100000,188681000000);

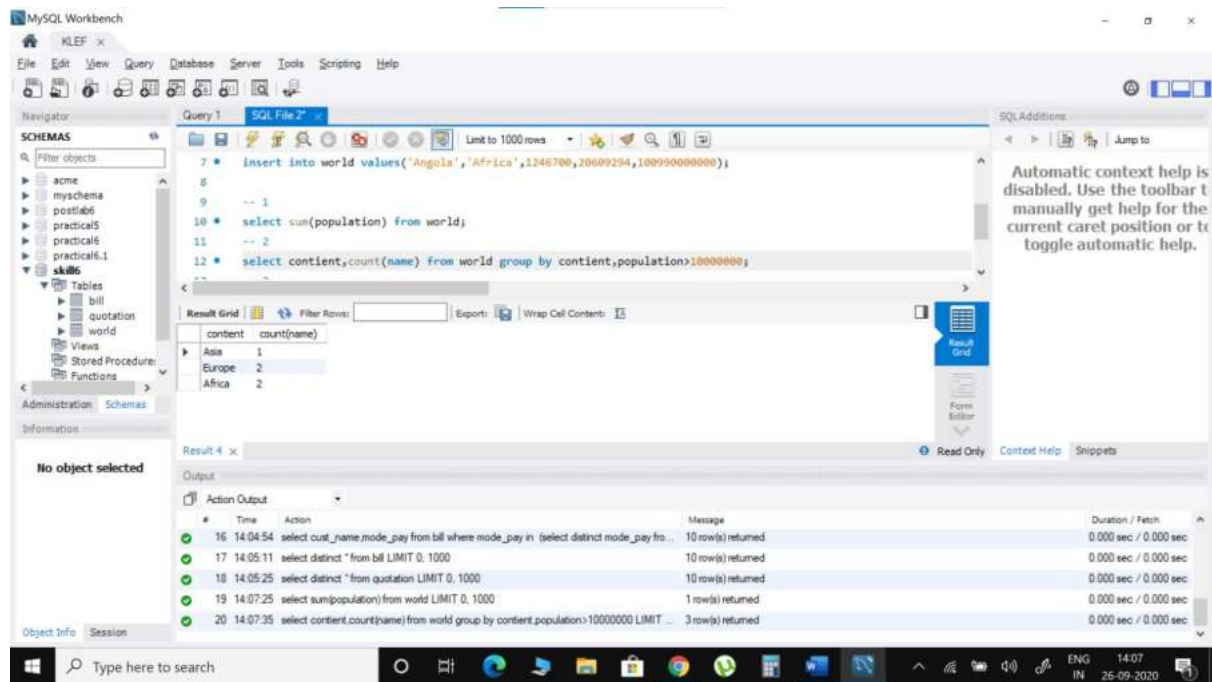
insert into world values('Andorra','Europe',468,78115,3712000000);

insert into world values('Angola','Africa',1246700,20609294,100990000000);

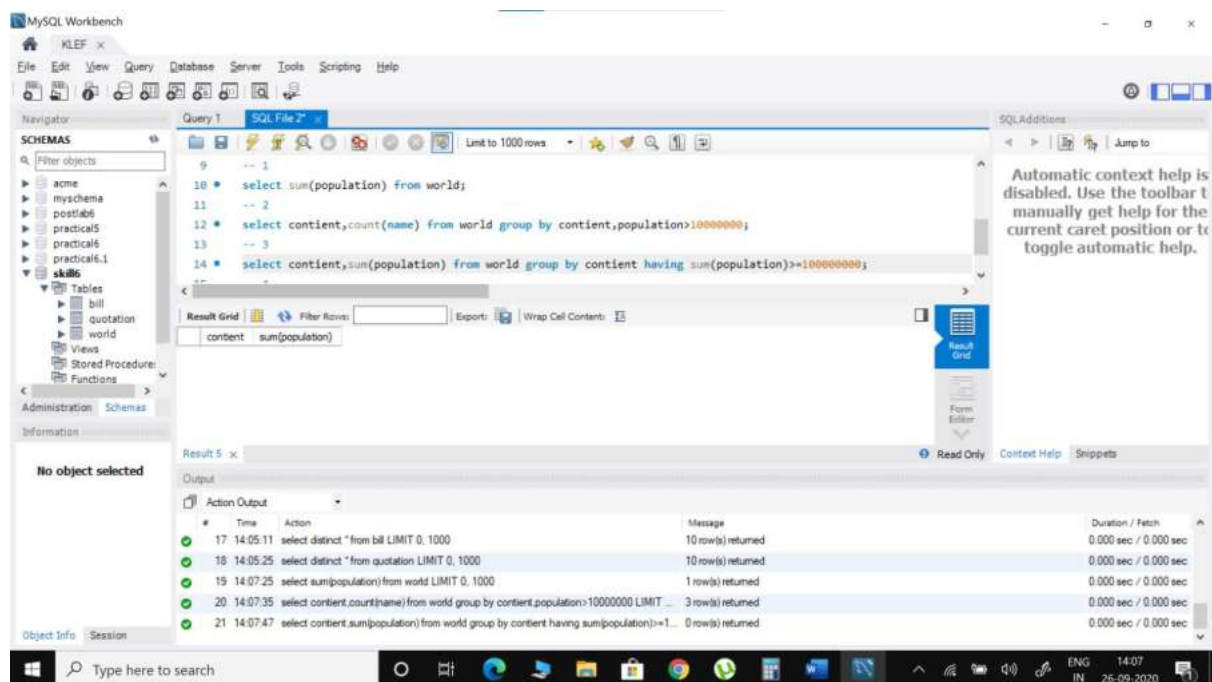
- 1) Show the total population of the world. World (name, continent, area, population, gdp)
select sum(population) from world;



- 2) For each continent show the continent and number of countries with populations of at least 10 million.
select contient,count(name) from world group by
contient,population>10000000;



3) List the continents that have a total population of at least 100 million.
 select contient,sum(population) from world group by contient having
 sum(population)>=100000000;



4) Display the list of continents in the world
 select distinct contient from world;

The screenshot displays the MySQL Workbench interface. The left sidebar shows the 'SCHEMAS' tree with 'world' selected. The main editor shows a SQL query with four statements. The 'Result Grid' shows the results of the first statement, which is a table with one column 'continent' and three rows: 'Asia', 'Europe', and 'Africa'. The 'Output' pane at the bottom shows the execution log with timestamps and messages for each statement.

SQL Query:

```
11 -- 2
12 * select continent, count(name) from world group by continent, population > 10000000;
13 -- 3
14 * select continent, sum(population) from world group by continent having sum(population) >= 100000000;
15 -- 4
16 * select distinct continent from world;
```

Result Grid:

continent
Asia
Europe
Africa

Output Log:

#	Time	Action	Message	Duration / Fetch
18	14:05:25	select distinct " from quotation LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
19	14:07:25	select sum(population) from world LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
20	14:07:35	select continent, count(name) from world group by continent, population > 10000000 LIMIT ...	3 row(s) returned	0.000 sec / 0.000 sec
21	14:07:47	select continent, sum(population) from world group by continent having sum(population) >= 1...	0 row(s) returned	0.000 sec / 0.000 sec
22	14:08:01	select distinct continent from world LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec