DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

LAB MANUAL

Academic Year: 2018-19 ODD SEMESTER

Program : UG - CSE

Semester : 5

Course Code : 15IT302J

Course Title : DATABASE MANAGEMENT SYSTEM

Prepared By

Abhash Shrestha, RA1611003011297 CSE

Naresh Bohara, RA1611003011324 CSE

Submitted To,

Mr. R.SUBASH,

Assistant Professor (O.G),

Department of Computer Science and Engineering

FACULTY OF ENGINEERING AND TECHNOLOGY SRM Institute of Science and Technology

SRM Nagar, Kattankulathur- 603203 Kancheepuram District

LIST OF EXPERIMENTS& SCHEDULE

Course Code: 15IT302J

Course Title: Database Management System

Exp. No.	Title	Week No.
1	Creating database table	1
2	Working with Data Manipulation commands	2
3	Basic SELECT statements	3
4	Advanced SELECT statements	4
5	Integrity and Constraints	5
6	Joining Tables	6
7	SQL functions	7
8	Sub queries	8
9	Views	9
10	Basics of PL/SQL	10,11
11	Design and Develop applications	12,13

HARDWARE AND SOFTWARE REQUIREMENTS

HARDWARE REQUIREMENTS:

INTEL PENTIUM 915 GV

80GB HDD

512MB DDR

SOFTWARE REQUIREMENTS:

ORACLE 8i, 9i.

MY SQL,

DB2

INTERNAL ASSESSMENT MARK SPLIT UP

Observation: 30 Marks + Mini Project with the Report: 10 Marks : 40 Marks

Model Exam : 10 Marks

Record : 5 Marks

MCQ / Quiz / Viva Voice : 5Marks

TOTAL MARKS : 60 Marks

EXTERNAL ASSESSMENT MARK : 40 Marks

EX.NO:1 CREATING DATABASE TABLE

AIM:

To create a DDL to perform creation of table, alter, modify and drop column.

QUERY:

1. create table emp(empno number(10) , ename varchar2(10),des varchar2(10),salary number(10,2));

DISCRIPTION:

The Create Table Command: - it defines each column of the table uniquely. Each column has minimum of three attributes, a name , data type and size.

OUTPUT: table created , desc emp

NAME	NULL ?	TYPE
Empno		number(4)
Ename		varchar 2(10)
Des		varchar 2(10)
Salary		number(10,2)

QUERY:

2. create table emp1 As select * from emp;

DISCRIPTION:

We copied all the data of emp to the newly created table emp1.

OUTPUT:

NAME	NULL ?	TYPE
Empno		number(10)
Ename		varchar 2(10)
Des		varchar 2(10)
Salary		number(10,2)

QUERY:

3. create table emp2 As select empno,ename from emp;

DISCRIPTION:

we only copied empno & ename from emp to emp2.

OUTPUT: Table created ,desc emp2

NAME	**NULL ?**	**TYPE**
Empno		number(10)
Ename		varchar 2(10)

QUERY:

4. create table emp3 As select * from emp where 1>2;

DISCRIPTION:

We only copy the basic structure of the table and not original.

OUTPUT: Table created ,desc emp2

NAME	**NULL ?**	**TYPE**
Empno		number(10)
Ename		varchar 2(10)
Des		varchar 2(10)
Salary		number(10,2)

QUERY:

5. alter table emp modify empno number(20);

DISCRIPTION:

We here change the size of attribute empno.

OUTPUT: Table created ,desc emp2

NAME	NULL ?	TYPE
Empno		number(20)
Ename		varchar 2(10)
Des		varchar 2(10)
Salary		number(10)

QUERY:

6. alter table emp modify (empno number(20),ename varchar2(20);

DISCRIPTION:

Here we change the size of both emp no and ename.

OUTPUT: Table created ,desc emp.

NAME	NULL ?	TYPE
Empno		number(20)
Ename		varchar 2(20)
Des		varchar 2(10)
Salary		number(10,2)

QUERY:

7. alter table emp add qualification varchar2(20);

DISCRIPTION:

We added a new column of type varchar2(20).

OUTPUT: Table created ,desc emp

NAME	NULL ?	TYPE
Empno		number(20)
Ename		varchar 2(20)
Des		varchar 2(10)

Salary number(10,2)

Qualification varchar2(20)

QUERY:

8. alter table emp add (DOJ date ,DOB date);

DISCRIPTION:

We should added two new column to emp named DOJ and DOB of the type data.

OUTPUT: Table created ,desc emp

NAME NULL ? TYPE

Empno number(20)

Ename varchar 2(20)

Des varchar 2(10)

Salary number(10,2)

Qualification varchar2(20)

DOJ date

DOB date

QUERY:

9. alter table emp drop column DOJ;

DISCRIPTION:

we drop (delete) the column DOJ from the table emp.

OUTPUT: Table created , desc emp

NAME NULL ? TYPE

Empno number(20)

Ename varchar 2(20)

Des varchar 2(10)

Salary number(10,2)

Qualification varchar2(20)

DOB date

QUERY:

10. alter table emp drop (qualification,DOB);

DISCRIPTION:

We deleted attributes qualification and DOB.

OUTPUT: Table created ,desc emp

NAME	**NULL ?**	**TYPE**
Empno		number(20)
Ename		varchar 2(20)
Des		varchar 2(10)
Salary		number(10,2)

QUERY:

11. insert into emp values (1001,'soyash','GM',10000);

DISCRIPTION:

we need inserted data in the table.

OUTPUT: 1 row inserted.

QUERY:

12. select * from emp ;

DISCRIPTION:

To show the data in the table we use this command.

OUTPUT:

NAME	**NULL ?**	**TYPE**
Empno	**ename**	**des** **salary**
1001	yesh	GM 100000

QUERY:

13. alter table emp rename to emp14;

DISCRIPTION:

We changed the name of table emp as emp14.

OUTPUT: table altered.

QUERY:

14. drop table emp14;

DISCRIPTION:

To delete the whole table including.

OUTPUT: table droped.

QUERY:

15. truncate table emp14.

DISCRIPTION:

To delete only data inside the table.

OUTPUT: table truncate.

RESULT:

Thus the DDL commands have been executed successfully.

EX.NO:2 WORKING WITH DATA MANIPULATION COMMANDS

AIM :

To study the various DML commands and implement them on the database.

QUERY:

1. insert into dept values (1001,'arjun','99525',’AP’,10000);
2. insert into emp values(&empno,'&ename','&mobile’,&job',&sal);
3. insert into emp (empno,ename,phno) values(1002,sam,99667.’ASP’,12000);
4. insert into emp (empno,ename,phno) values(1003,Ram,99867,’ASP’,12000);

DISCRIPTION:

Insert command is used to put the values in the values in the table formed before we can take “&” symbol runtime input,while we can also change order of insertion.

OUTPUT: select * from emp;

EMPNO	ENAME	MOBILE	JOB	SALARY
1001	ARJUN	99525	AP	10000
1002	SAM	99667	ASP	12000
1003	RAM	99867	ASP	12000

QUERY:

5. update emp set sal=15000 where job='ASP';

DISCRIPTION:

Update the value in the table.

OUTPUT: select * from emp;

EMPNO	ENAME	MOBILE	JOB	SALARY
1001	ARJUN	99525	AP	10000
1002	SAM	99667	ASP	15000
1003	RAM	99867	ASP	15000

QUERY:

6. select ename, job from emp;

DISCRIPTION:

Select the attributs from the table.

OUTPUT: select ename,job from emp;

ENAME	JOB
ARJUN	AP
SAM	ASP
RAM	ASP

RESULT:

Thus the DML commands have been executed successfully.

EX.NO:3 BASIC SELECT STATEMENTS

AIM:

To study the various Basic Select statement on the database.

QUERY:

1. select * from emp order by sal;

DISCRIPTION:

Row sorting ascending order.

OUTPUT:

EMPNO	ENAME	MOBILE	JOB	SALARY
1001	ARJUN	99525	AP	10000
1002	SAM	99667	ASP	15000
1003	RAM	99867	ASP	15000

QUERY:

2. delete from emp where job='AP';

DISCRIPTION:

To delete a particular row.

OUTPUT: 1 row deleted, select * from emp;

EMPNO	ENAME	MOBILE	JOB	SALARY
1002	SAM	99667	ASP	15000
1003	RAM	99867	ASP	15000

QUERY:

3. select * from emp where empno='1002';

DISCRIPTION:

Column-row filter contained.

OUTPUT:

EMPNO	ENAME	MOBILE	JOB	SALARY
1002	SAM	99667	ASP	15000

QUERY:

4. select distinct empno from emp;

DISCRIPTION:

Display the distinct employee number.

OUTPUT:

EMPNO
1002
1003

QUERY:

5. select ename from emp where sal>14000;

DISCRIPTION:

Display the employee name who salaray is greater than 14000.

OUTPUT:

EMPNO	SALARY
1002	15000
1003	15000

QUERY:

6. SELECT * FROM Store_Information WHERE Txn_Date BETWEEN 'Jan-06-1999' AND 'Jan-10-1999';

DISCRIPTION:

Display the store information between the given date.

OUTPUT:

Store_Name	Sales	Txn_Date
San Diego	250	Jan-07-1999
San Francisco	300	Jan-08-1999
Boston	700	Jan-08-1999

QUERY:

7. SELECT * FROM Store_Information WHERE Store_Name IN ('Los Angeles', 'San Diego');

DISCRIPTION:

Display the store information of certain city.

OUTPUT:

Store_Name	Sales	Txn_Date
Los Angeles	1500	Jan-05-1999
San Diego	250	Jan-07-1999

QUERY:

8. SELECT * FROM Store_Information WHERE Store_Name LIKE '%AN%';

DISCRIPTION:

We want to find all stores whose name contains 'AN'.

OUTPUT:

Store_Name	Sales	Txn_Date
LOS ANGELES	1500	Jan-05-1999
SAN DIEGO	250	Jan-07-1999
SAN FRANCISCO	300	Jan-08-1999

QUERY:

9. SELECT Store_Name, Sales, Txn_Date FROM Store_Information ORDER BY Sales DESC;

DISCRIPTION:

To list the contents of Table Store_Information by Sales in descending order.

OUTPUT:

Store_Name	Sales	Txn_Date
Los Angeles	1500	Jan-05-1999
Boston	700	Jan-08-1999
San Francisco	300	Jan-08-1999
San Diego	250	Jan-07-1999

QUERY:

10. SELECT Store_Name, SUM(Sales)
FROM Store_Information GROUP BY Store_Name HAVING SUM(Sales) > 1500;

DISCRIPTION:

To see only the stores with sales over $1,500.

OUTPUT:

Store_Name	SUM(Sales)
Los Angeles	1800

RESULT:

Thus the Basic select commands have been executed successfully.

EX.NO:4 ADVANCED SELECT STATEMENTS

AIM:

To study the various Advanced Select statement on the database.

- **LIMIT:**

QUERY:

```
1. select * from courier limit 3;
```

DISCRIPTION:

LIMIT clause is used to specify the number of records to return.

OUTPUT:

```
name      | pincode | amount | day        | weight | delivery_type
sravanthi |  390008 |    250 | 1998-09-02 |     25 | express
yash      |  390007 |    520 | 1997-10-09 |     52 | express
medha     |  603203 |    460 | 1997-08-31 |     46 | express
```

- **CASE:**

QUERY:

```
2. select name, pincode,
          case
          when weight<100 then "weight is less than 100"
          else "weight >= 100"
          end
          from courier;
```

DISCRIPTION:

The CASE function lets you evaluate conditions and return a value.

OUTPUT:

```
  name         | pincode | case

| sravanthi   |  390008 | weight is less than 100
|
| yash        |  390007 | weight is less than 100
|
| medha       |  603203 | weight is less than 100
|
| shubhranshu |  203203 | weight is less than 100
|
| latha       |  400205 | weight is less than 100


                                                    |
| reddy       |  502504 | weight is less than 100
|
| padmavathi  |  502504 | weight >= 100
```

- **UNION:**

QUERY:

```
3. select pincode from courier union select pincode from courier1;
```

DISCRIPTION:

The UNION operator is used to combine the result-set of two or more SELECT statements.
Each SELECT statement within UNION must have the same number of columns
The columns must also have similar data types.
The columns in each SELECT statement must also be in the same order.

OUTPUT:

```
 PINCODE
----------
 110044
 110062
 110076
 390006
 390007
 390008
 516432
 516434
 516436
 602401
 603203
```

- **INTERSECT :**

QUERY:

```
4. select pincode from courier intersect select pincode from
   courier1;
```

DISCRIPTION:

The SQL INTERSECT clause/operator is used to combine two SELECT statements, but returns rows only from the first SELECT statement that are identical to a row in the second SELECT statement. This means INTERSECT returns only common rows returned by the two SELECT statements.

OUTPUT:

```
 PINCODE
---------
  603203
```

- **MINUS :**

QUERY:

```
5. select pincode from courier minus select pincode from courier1;
```

DISCRIPTION:

The Minus Operator in SQL is used with two SELECT statements. The MINUS operator is used to subtract the result set obtained by first SELECT query from the result set obtained by second SELECT query.

OUTPUT:

```
   PINCODE
----------
    110044
    110062
    110076
    390006
    390007
    390008
```

- **EXISTS:**

QUERY:

```
6. select place_name from courier where exists(select name from
   courier1 where pincode=courier.pin
   code and bill<1000);
```

DISCRIPTION:

The EXISTS operator is used to test for the existence of any record in a subquery. The EXISTS operator returns true if the subquery returns one or more records.

OUTPUT:

```
PLACE_NAME
-----------
 Potheri
```

RESULT:

Thus the advanced Select Statements have been executed successfully.

EX.NO:5 INTEGRITY AND CONSTRAINTS

AIM:

To implement various constraint in sql plus.

- **Primary Key:**

QUERY:

1. Create table emp(empno number(10) primary key , ename varchar2(10));
2. Create table emp2(empno number(10) Constraint_emp primary key,ename varchar2(10));
3. Create table emp3(empno number(10) ,ename varchar2(10), Constraint_emp primary key(emp no));
4. , Constraint emp_FK Foreign key(dept) reference dept(dept no));

DISCRIPTION:

It does not allow repletion and take unique and not null entites.

OUTPUT: desc emp 1

1) NAME	**NULL ?**	**TYPE**
Empno	not null	number(10)
Ename		varchar 2(10)

OUTPUT: desc dept

2) NAME	**NULL ?**	**TYPE**
Dept no	not null	number(10)
Dname		varchar 2(10)

OUTPUT: desc emp5

3) NAME	**NULL ?**	**TYPE**
Empno		number(10)
Dm		number(10)

- ## Foreign Key:

QUERY: {creating main table}

- Create table dept(dptno number(10) primary key , dname varchar2(10));

{constriants}:

1. Create table emp5(empno number(10) ,dept number(10) reference dept(dept no));
2. Create table emp6(empno number(10), dept number(10) Constraint emp_FK reference dept(dept no));
3. Create table emp7(empno number(10), dept number(10) , Constraint emp_FK Foreign key(dept) reference dept(dept no));
4. Alter table emp8 ADD Constraint emp_FK Foreign key(dept) reference dept(dept no));

DISCRIPTION:

Use to get reference of data from the main table defined by the user.

OUTPUT: desc dept

1) NAME NULL ? TYPE

Dept no	not null	number(10)
Name		varchar 2(10)
Location		varchar 2(10)

OUTPUT: desc emp5

2) NAME NULL ? TYPE

emo no	number(10)
Dept no	number(10)
Design	varchar 2(10)

- ## Check:

QUERY:

1. Create table emp9(empno number(10) ,sal number(10) check (sal>500 and sal <1000));
2. Create table emp10(empno number(10) ,sal number(10) Constraint emp_un check (sal>500 and sal <1000));
3. Create table emp11(empno number(10) ,sal number(10), Constraint emp_un , check (sal>500 and sal <1000));
4. Alter table emp12 ADD Constraint emp_um2 check (sal>500 and sal <1000));

DISCRIPTION:

It is used when we have to check the given condition and the data shown in the output should abide by the condition.

OUTPUT:

NAME	**NULL ?**	**TYPE**
Empno		number(10)
Sal		number(10)

- ## UNIQUE:

QUERY:

1. Create table emp13(empno number(10) unique ,sal number(10));
2. Create table emp14(empno number(10) constraint emp2_emp_um unique ,sal number(10));
3. Create table emp16(empno number(10) ,sal number(10), constraint emp9_um unique (emp no));
4. Alter table emp15 ADD Constraint emp_um unique(emp no));

DISCRIPTION:

It accept unique entities irrespective od null or not null values.

OUTPUT:

NAME	NULL ?	TYPE
Empno		number(10)
Sal		number(10)

- **NOT NULL:**

QUERY:

1. Create table emp17(empno number(10) ,dm number(10) not null);
2. Create table emp18(empno number(10) ,dm number(10) constraint em_1 not null);
3. Create table emp19(empno number(10) ,dm number(10) constraint em_3 not null(dm));

DISCRIPTION:

It can accept not null values.

OUTPUT:

NAME	NULL ?	TYPE
Empno		number(10)
Dm	not null	number(10)

- **NULL:**

QUERY:

1. Create table emp20(empno number(10) ,dm number(10) null);
2. Create table emp21(empno number(10) ,dm number(10) constraint em null);

DISCRIPTION:

It can accept not null values.

OUTPUT:

NAME	NULL ?	TYPE
Empno		number(10)
Dm		number(10)

- **DEFAULT:**

QUERY:

1. Create table emp22(empno number(10) ,dm number(10) default (10));

DISCRIPTION:

In thid type of constraints we set a default value for the query.

OUTPUT:

NAME	NULL ?	TYPE
Empno		number(10)
Dm		number(10)

RESULT:

Thus the Integrity and Constraints have been executed successfully

EX.NO:6 JOINING TABLES

AIM :

To study the various Join operations on the database.

QUERY:

1. `SELECT * FROM COURIER1;`

DISCRIPTION:

It shows table 1 content.

OUTPUT:

```
NAME                 ADDRESS            PINCODE    CONTACT     AMOUNT DAY
-------------------- --------------- ---------- ---------- ---------- -----
----
SARANYA              KANCHEEPURAM        603203 7845129663        250 21-
MAY-18

JHON                 CHENNAI             603202 9887456321        250 21-
MAY-18

KALAIVANI            KONCHI              502203 8798456321        250 21-
MAY-18

VIJYAKUMAR           BANGALORE           402201 9888456321        300

RAJAKUMARAN          TRICHY              542103 7845123690        250 21-
MAY-18

RAM                  LUCKNOW             226010 9415400435        250 21-
MAY-18

JASMEET              CHANDIGARH          335001 9002145630        320
```

QUERY:

2. `SELECT * FROM COURIER2;`

DISCRIPTION:

It shows the table 2 content.

OUTPUT:

```
TO_NAME    TO_ADDRESS TO_CONTACT TO_PINCODE     AMOUNT NAME
---------- ---------- ---------- ---------- ---------- ----------
MEDHA      LKO        8687333303     226010        300 SARANYA
DEVANSH    UK         7478015222     226611        300 JHON
SHRUTI     TN         9415400435     603203        300 KALAIVANI
HARSH      KA         9876543210     401020        300 KALAM
ADITI      BANGALORE  9878654510     226550        300 PALAK
SUNIDHI    MUMBAI     8787662100     403203        300 JASMEET
SANYOGITA  DELHI      9045022310     502330        300 RAM
```

- **INNER JOIN:**

QUERY:

3. `SELECT ADDRESS,PINCODE FROM COURIER1 INNER JOIN COURIER2 ON COURIER1.NAME=COURIER2.NAME;`

DISCRIPTION:

The INNER JOIN keyword selects records that have matching values in both tables.

OUTPUT:

```
ADDRESS            PINCODE
--------------- ----------
KANCHEEPURAM        603203
CHENNAI             603202
KONCHI              502203
LUCKNOW             226010
CHANDIGARH          335001
```

- **LEFT JOIN:**

QUERY:

4. SELECT TO_ADDRESS, TO_CONTACT FROM COURIER2 LEFT JOIN COURIER1 ON COURIER2.NAME=COURIER1.NAME;

DISCRIPTION:

The LEFT JOIN keyword returns all records from the left table (table1), and the matched records from the right table (table2). The result is NULL from the right side, if there is no match.

OUTPUT:

```
TO_ADDRESS TO_CONTACT

---------- ----------

LKO        8687333303

UK         7478015222

TN         9415400435

DELHI      9045022310

MUMBAI     8787662100

BANGALORE  9878654510

KA         9876543210
```

- **RIGHT JOIN:**

QUERY:

5. SELECT ADDRESS,CONTACT FROM COURIER1 RIGHT JOIN COURIER2 ON COURIER1.NAME= COURIER2.NAME;

DISCRIPTION:

The RIGHT JOIN keyword returns all records from the right table (table2), and the matched records from the left table (table1). The result is NULL from the left side, when there is no match.

OUTPUT:

ADDRESS	CONTACT
KANCHEEPURAM	7845129663
CHENNAI	9887456321
KONCHI	8798456321
LUCKNOW	9415400435
CHANDIGARH	9002145630

- **FULL OUTER JOIN:**

QUERY:

6. SELECT ADDRESS,PINCODE FROM COURIER1 FULL OUTER JOIN COURIER2 ON COURIER1.NAME=COURIER2.NAME;

DISCRIPTION:

The FULL OUTER JOIN keyword return all records when there is a match in either left (table1) or right (table2) table records.

OUTPUT:

ADDRESS	PINCODE
KANCHEEPURAM	603203
CHENNAI	603202
KONCHI	502203
CHANDIGARH	335001
LUCKNOW	226010
TRICHY	542103
BANGALORE	402201

RESULT:

Thus the joining tables have been executed successfully.

EX.NO:7 SQL FUNCTIONS

QUERY:

1. select upper('welcome') from dual;

DISCRIPTION:

Convert it capital letter.

OUTPUT:

WELCOME

QUERY:

2. select lower('HAI') from dual;

DISCRIPTION:

Convert it lower case.

OUTPUT:

Hai

QUERY:

3. select initcap('hello world') from dual;

DISCRIPTION:

Convert the starting letter into the upper case.

OUTPUT:

Hello World

QUERY:

4. select ltrim(' hai') from dual;

DISCRIPTION:

It trim the left side of the character.

OUTPUT:

hai

QUERY:

5. select rtrim('hai ')from dual;

DISCRIPTION:

It trim the right side of the character.

OUTPUT:

hai

QUERY:

6. select concat('SRM',' university')from dual;

DISCRIPTION:

Join the two different string.

OUTPUT:

SRM university

QUERY:

7. select length('SRM')from dual;

DISCRIPTION:

Display the length of the given string.

OUTPUT:

3

QUERY:

8. select replace('SRM university', 'SRM','Anna')from dual;

DISCRIPTION:

Replace the string from another string.

OUTPUT:

Anna university

QUERY:

9. select substr('SRM university', 7,6)from dual;

DISCRIPTION:

Remove the subtraction that remove at certain position size.

OUTPUT:

Iversi

QUERY:

10. select rpad('hai',3,'*')from dual;

DISCRIPTION:

Display the right side of the portion of the string.

OUTPUT:

hai***

QUERY:

11. select lpad('hai',3,'*')from dual;

DISCRIPTION:

Display the left side of the portion of the string.

OUTPUT:

***hai

QUERY:

12. select replace('Dany','y','ie')from dual;

DISCRIPTION:

Replace the one character to another character.

OUTPUT:

Danie

QUERY:

13. select translate('cold','ld','ol')from dual;

DISCRIPTION:

Translate one character to another character.

OUTPUT:

cool

- **DATE & TIME FUNCTION:**

QUERY:

14. select sysdate from dual;

DISCRIPTION:

Display system date.

OUTPUT:

07-APR-18

QUERY:

15. select round(sysdate)from dual;

DISCRIPTION:

Display round system date.

OUTPUT:

07-APR-18

QUERY:

16. select add_months(sysdate,3)from dual;

DISCRIPTION:

Add the month to the system date.

OUTPUT:

31-AUG-18

QUERY:

17. select sysdate+20 from dual;

DISCRIPTION:

Add +20 days to the present day and display.

OUTPUT:

27-APR-18

QUERY:

18. select next_day(sysdate,'tuesday')from dual;

DISCRIPTION:

Display next Tuesday date.

OUTPUT:

13-APR-18

- **NUMERIC FUNCTION:**

QUERY:

19. select ceil(23.20)from dual;

DISCRIPTION:

Display the round of value.

OUTPUT:

24

QUERY:

20. select floor(23.20)from dual;

DISCRIPTION:

Display the number excluding the point .

OUTPUT:

23

QUERY:

21. select trunc(15.56743)from dual;

DISCRIPTION:

Display the number before the point.

OUTPUT:

15

QUERY:

22. select sign(-345)from dual;

DISCRIPTION:

Display the sign of the given number .

OUTPUT:

-1

QUERY:

23. select abs(-70)from dual;

DISCRIPTION:

Display the opposite sign value.

OUTPUT:

70

- **MATH FUNCTION:**

QUERY:

24. select power(10,12) from dual;

DISCRIPTION:

Display the power of the given input.

OUTPUT:

1.000E+12

QUERY:

25. select mod(11,5) from dual;

DISCRIPTION:

Display remainder of the given input.

OUTPUT:

1

QUERY:

26. select exp(10) from dual;

DISCRIPTION:

Display the exponent value of given input.

OUTPUT:

22026.466

QUERY:

27. select sqrt(225) from dual;

DISCRIPTION:

Display the square root value of the given input.

OUTPUT:

15

- **AGGREGATE FUNCTION:**

QUERY:

28. Create table abc (xyz number(10));
29. Insert into abc value(&xyz);

DISCRIPTION: To main table.

OUTPUT: select *from abc;

xyz

1

2

3

4

5

QUERY:

30. Select min(xyz) from abc where abc>0;

DISCRIPTION:

To find minimum value.

OUTPUT:

Min(xyz)

1

QUERY:

31. Select max(xyz) from abc ;

DISCRIPTION:

To find maximum value.

OUTPUT:

Max(xyz)

5

QUERY:

32. Select count(xyz) from abc ;

DISCRIPTION:

To find the count of the given table.

OUTPUT:

Count(xyz)

5

QUERY:

33. Select avg(xyz) from abc ;

DISCRIPTION:

To find the average of the given table.

OUTPUT:

AVG(xyz)

3

QUERY:

34. Select sum(xyz) from abc ;

DISCRIPTION:

To find the sum of table.

OUTPUT:

SUM(xyz)

15

RESULT:

Thus the SQL Functions have been executed successfully.

EX.NO:9 VIEWS

AIM:

To study various VIEW operations on database.

1.CREATE VIEW:

QUERY:

```
35. CREATE VIEW PACKAGE AS SELECT NAME,AMOUNT,DAY FROM COURIER1 ;
```

DISCRIPTION:

Database views are created using the CREATE VIEW statement.

OUTPUT: `SELECT * FROM PACKAGE;`

```
NAME                      AMOUNT DAY

-------------------- ---------- ---------

SARANYA                      250 21-MAY-18

JHON                         250 21-MAY-18

KALAIVANI                    250 21-MAY-18

VIJYAKUMAR                   300

RAJAKUMARAN                  250 21-MAY-18

RAM                          250 21-MAY-18

JASMEET                      320
```

2. INSERT INTO VIEW:

QUERY:

```
36. INSERT INTO PACKAGE VALUES('MEDHA',225,'20-SEP-2018');
37. INSERT INTO PACKAGE VALUES('&NAME',&AMOUNT,'&DAY');
```

DISCRIPTION:

We can insert into table using views. This updates both the original table and the view we created.

OUTPUT1:

```
1 row created.
```

OUTPUT2:

```
Enter value for name: DEVANSH

Enter value for amount: 320

Enter value for day: 14-SEP-2018

old   1: INSERT INTO PACKAGE VALUES('&NAME',&AMOUNT,'&DAY')

new   1: INSERT INTO PACKAGE VALUES('DEVANSH',320,'14-SEP-2018')

1 row created.
```

3. UPDATE VIEW:

QUERY:

```
38. UPDATE PACKAGE SET DAY='26-SEP-18' WHERE NAME='JASMEET';
39. UPDATE PACKAGE SET DAY='23-SEP-18' WHERE AMOUNT=300;
```

DISCRIPTION:

We can update the table using this. Both the table and the view will be updated.

OUTPUT1:

```
1 row created.
```

OUTPUT2:

```
1 row created.
```

4. DELETE FROM VIEW:

QUERY:

```
40. DELETE FROM PACKAGE WHERE DAY='26-SEP-18';
41. DELETE FROM PACKAGE;
```

DISCRIPTION:

You can delete a record from view and the same will also be reflected in the original table.

OUTPUT1:

```
1 row deleted.
```

OUTPUT2:

```
8  rows deleted.
```

5. DROP VIEW:

QUERY:

42. DROP VIEW PACKAGE;

DISCRIPTION:

This will drop the full view but wont affect the original table.

OUTPUT:

View dropped.

RESULT:

The view operations were successfully implemented.

EX.NO:10 BASICS OF PL/SQL

AIM :

To study the various basic PL/SQL view operations on the database.

1.PL/SQL CODING FOR ADDITION OF TWO NUMBERS:

QUERY:

```
set serveroutput on;
declare
a number;
b number;
c number;
begin
a:=&a;
b:=&b;
c:=a+b;
dbms_output.put_line('sum of'||a||'and'||b||'is'||c);
end;
/
```

DISCRIPTION:

```
It will calculate addition of Two numbers
```

INPUT:

```
Enter value for a: 23
old 6: a:=&a;
new 6: a:=23;
Enter value for b: 12
old 7: b:=&b;
new 7: b:=12;
```

OUTPUT:

```
sum of23and12is35
```

2.PL/ SQL GENERAL SYNTAX FOR IF CONDITION:

QUERY:

```
set serveroutput on;
DECLARE
b number;
c number;
BEGIN
B:=10;
C:=20;
if(C>B) THEN
dbms_output.put_line('C is maximum');
 end if;
 end;
   /
```

DISCRIPTION:

It will show the basic syntax of if condition. It will calculate the maximum of 2 numbers.

OUTPUT:

C is maximum

3.PL/ SQL GENERAL SYNTAX FOR IF AND ELSECONDITION:

QUERY:

```
 declare
n number;
begin
dbms_output. put_line('enter a number');
n:=&number;
if n<5 then
dbms_output.put_line('entered number is less than 5');
else
dbms_output.put_line('entered number is greater than 5');
end if;
end;
/
```

DISCRIPTION:

It will show the basic syntax of ifand else condition. It will calculate the maximum of 2 numbers.

OUTPUT: n=3

The entered number is greater than 5

4.PL/ SQL GENERAL SYNTAX FOR NESTED IF:

QUERY:

```
declare
a number;
b number;
c number;
d number;
begin
a:=&a;
b:=&b;
c:=&b;
if(a>b)and(a>c) then
dbms_output.put_line('A is maximum');
elsif(b>a)and(b>c)then
dbms_output.put_line('B is maximum');
else
dbms_output.put_line('C is maximum');
end if;
end;
/
```

DISCRIPTION:

It shows the basic syntax of nested if condition.It will calculate the maximum of 2 numbers.It will calculate the maximum of 3 numbers.

INPUT:

```
A=6
B=7
C=5
```

OUTPUT:

```
B is maximum
```

5.PL/ SQL GENERAL SYNTAX FOR LOOPING STATEMENT:

QUERY:

```
declare
n number;
sum1 number default 0;
endvalue number;
begin
endvalue:=&endvalue;
n:=1;
for n in 1..endvalue
Loop
if mod(n,2)=1
Then
sum1:=sum1+n;
end if;
end loop;
dbms_output.put_line('sum ='||sum1);
end;
/
```

DISCRIPTION:

It shows the basic syntax of looping statement.

INPUT:

N=5

OUTPUT:

Sum=10

6. TRIGGER:

TYPE 1- TRIGGER AFTER UPDATE:

QUERY:

```
 CREATE OR REPLACE TRIGGER VIJAY AFTER UPDATE OR INSERT
OR DELETE ON EMP FOR EACH ROW BEGIN IF UPDATING THEN
DBMS_OUTPUT.PUT_LINE('TABLE IS UPDATED'); ELSIF INSERTING
THEN  DBMS_OUTPUT.PUT_LINE('TABLE IS INSERTED'); ELSIF
DELETING THEN

DBMS_OUTPUT.PUT_LINE('TABLE IS DELETED');
 END IF;
 END;
 /
```

DISCRIPTION:

It shows the basic syntax of trigger statement.

OUTPUT:

```
Trigger created. SQL> update emp set income =900 where
empname='kumar';
TABLE IS UPDATED 1 row updated.
SQL> insert into emp values ( 4,'Chandru',700,250,80); TABLE
IS INSERTED 1 row created.
SQL> DELETE FROM EMP WHERE EMPID = 4; TABLE IS DELETED
1 row deleted.
```

TYPE 2 - TRIGGER BEFORE UPDATE:

QUERY:

```
 CREATE OR REPLACE TRIGGER VASANTH
BEFORE UPDATE OR INSERT OR DELETE ON EMPLOYEE
 FOR EACH ROW
 BEGIN
IF UPDATING THEN
 DBMS_OUTPUT.PUT_LINE('TABLE IS UPDATED');
ELSIF INSERTING THEN
 DBMS_OUTPUT.PUT_LINE('TABLE IS INSERTED');
 ELSIF DELETING THEN
DBMS_OUTPUT.PUT_LINE('TABLE IS DELETED');
END IF;
 END;
/
```

OUTPUT:

```
Trigger created. SQL> INSERT INTO EMP VALUES
(4,'SANKAR',700,98,564);
TABLE IS INSERTED
1 row created.
 SQL> UPDATE EMP SET EMPID = 5 WHERE EMPNAME = 'SANKAR';
TABLE IS UPDATED
1 row updated.
SQL> DELETE EMP WHERE EMPNAME='SANKAR'; TABLE IS DELETED
 1  row deleted
```

3.Create a Trigger to check the age valid or not Using Message Alert:

QUERY:

```
CREATE TRIGGER TRIGNEW
AFTER INSERT OR UPDATE OF AGE ON TRIG
FOR EACH ROW
 BEGIN IF(:NEW.AGE < 0 ) THEN
DBMS_OUTPUT.PUT_LINE('INVALID AGE');
 ELSE
DBMS_OUTPUT.PUT_LINE('VALID AGE');
END IF;
END;
/
SQL> CREATE TRIGGER DATACHECK
AFTER INSERT OR UPDATE OF AGE ON DATA
 FOR EACH ROW
 BEGIN
IF(:NEW.AGE < 0) THEN
RAISE_APPLICATION_ERROR(-20000,'NO NEGATIVE AGE ALLOWED');
END IF;
END;
/
```

OUTPUT:

```
Trigger created. SQL> insert into trig values('abc',15);
Valid age 1 row created.
SQL> insert into trig values('xyz',-12);
 Invalid age 1 row created.
NAME  AGE
--------
abc   15
xyz  -12
SQL> create table data(name char(10),age number(3));
Table created.
```

PROCEDURE USING POSITIONAL PARAMETERS:

QUERY:

```
SQL> SET SERVEROUTPUT ON;
SQL> CREATE OR REPLACE PROCEDURE PROC
1 AS
2 BEGIN
3 DBMS_OUTPUT.PUT_LINE('Hello from procedure...');
4 END;
5 /
```

DISCRIPTION:

It will give the position using positional parameters.

OUTPUT:

```
: Procedure created.
SQL> EXECUTE PROC1 Hello from procedure...
```

PROCEDURE USING NOTATIONAL PARAMETERS:

QUERY:

```
CREATE OR REPLACE PROCEDURE PROC2
2 (N1 IN NUMBER,N2 IN NUMBER,TOT OUT NUMBER) IS
3 BEGIN
4 TOT := N1 + N2;
5 END;
6 /
```

DISCRIPTION:

It will procedure using notational parameters.

INPUT:

```
Output: Procedure created.
SQL> VARIABLE T NUMBER
SQL> EXEC PROC2(33,66,:T)
 PL/SQL procedure successfully completed.
SQL> PRINT T
T
```

OUTPUT:

99

EX.NO:8 SUB QUERIES

AIM:

To study the various SQL sub queries operations on the database

QUERY:

1. select deptid , deptname from emp dept where empid=(select empid from empsalary where empid >103);

DISCRIPTION:

Select department and department name of the company which is assigned to the employee whose employee id is greater than 103.

OUTPUT:

Name Dept id Emp id

Dev 226010 105

Sakti 236611 108

Rohit 603203 109

QUERY:

1. select salary from emp salary where project id =(select project id from project where project id = p-2);

DISCRIPTION:

Select salary of the employee who is currently working on the project p-2.

OUTPUT:

Name salary project- id

Dev 22600 p-2

Sakti 23611 p-2

Rohit 60103 p-2

QUERY:

1. select empname from employee where deptname = (select deptname from empdept where deptname = 'HR');

DISCRIPTION:

Select name of the employee who is department head is HR.

OUTPUT:

Name	salary	project- id	Dept Head
Dev	22600	p-2	HR
Sakti	22611	p-2	HR

QUERY:

1. select empname from employee where deptname = (select deptname from empdept where deptname = 'SALES');

DISCRIPTION:

Select name of the employee who is department head in soles.

OUTPUT:

Name	salary	project- id	Dept Head
Dev	22600	p-2	sales
Sakti	22611	p-2	sales

QUERY:

1. select empname from employee where deptname = (select deptname from empdept where deptname = 'SALES' or empdept.deptname = 'HR');

DISCRIPTION:

Select name of the employee who is department head is sales or department head is HR.

OUTPUT:

Name	salary	project- id	Dept Head
Dev	22600	p-2	sales
Sakti	22611	p-2	sales
Medha	22600	p-2	HR

QUERY:

2. Select duration from project where project ID not in (select project ID from empproject);

DISCRIPTION:

Duration of the project which didn't assigned to any candidate.

OUTPUT:

Name	salary	project- id	Dept Head
RAM	22600	p-2	Sales

QUERY:

1. select empname from employee where deptname = (select deptid from empdept where deptname = 'HR') and empid in (select empid from empsalary where is permanent = 'Yes');

DISCRIPTION:

Select the name of the candidate who is working HR deparment and they are working as permanent.

OUTPUT:

Name	salary	project- id	Dept Head	Ispermanenet
RAM	22600	p-2	Sales	yes

RESULT:

Thus the SQL sub query has been executed successfully.

EX.NO:11 Design and Develop applications

Hostel Management System

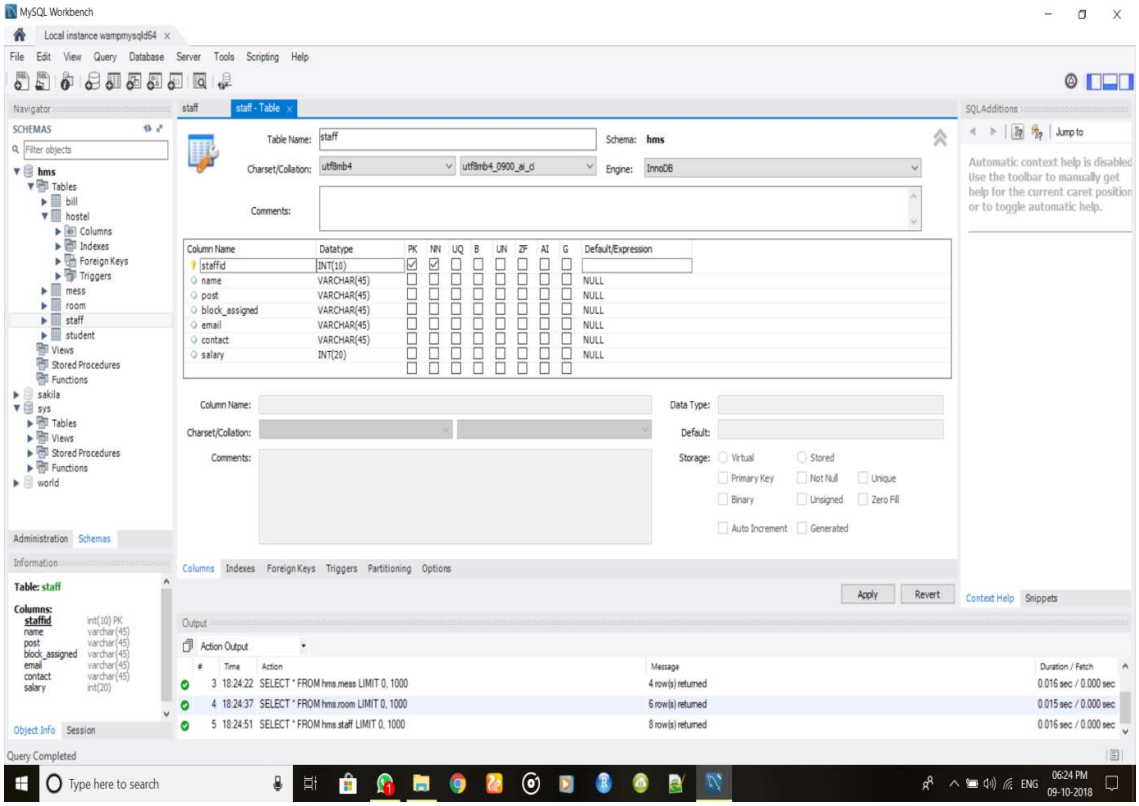
SOURCE CODE:

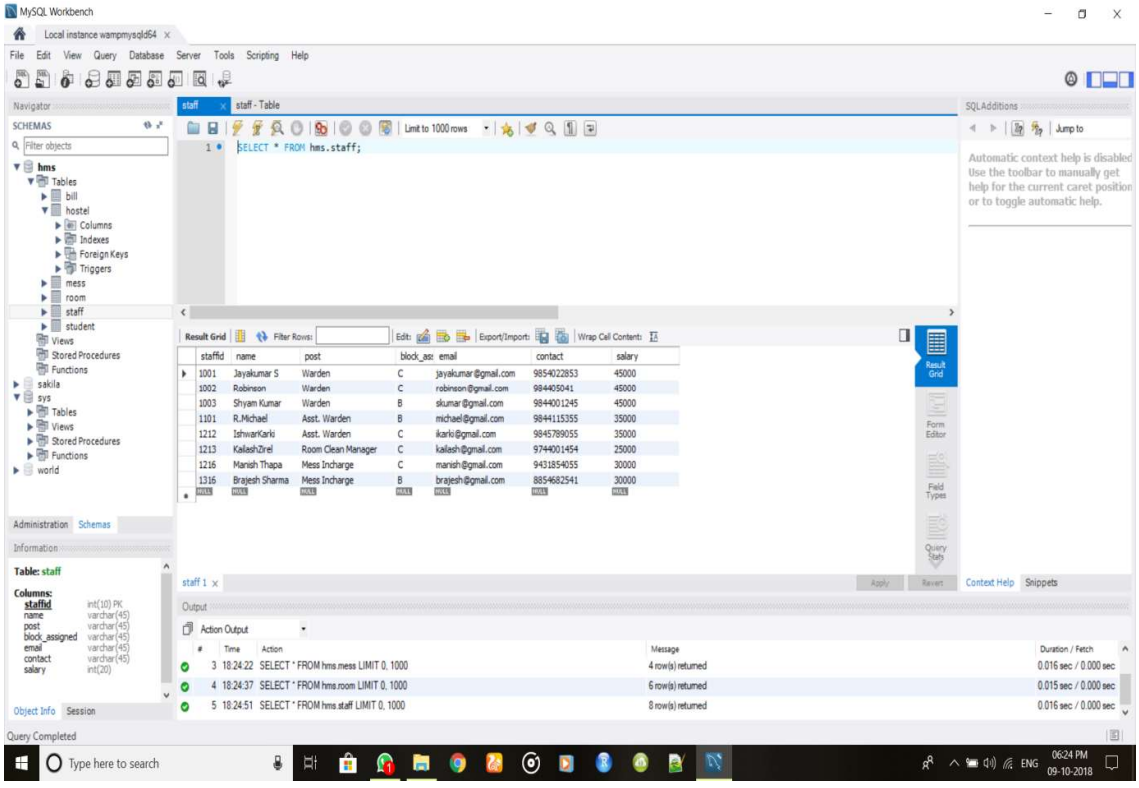
#Creating a table for staff where staff id is the primary key

```
CREATE TABLE `hms`.`staff` (
  `staffid` INT(20) NOT NULL,
  `name` VARCHAR(45) NULL,
  `post` VARCHAR(45) NULL,
  `block_assigned` VARCHAR(45) NULL,
  `email` VARCHAR(45) NULL,
  `contact` INT(20) NULL,
  `salary` INT(20) NULL,
  PRIMARY KEY (`staffid`));
```

#Inserting values into staff

```
 INSERT INTO `hms`.`staff` (`staffid`, `name`, `post`, `block_assigned`, `email`, `contact`,
`salary`) VALUES ('1001', 'Jayakumar S', 'Warden', 'C', 'jayakumar@gmail.com', '9854022853',
'45000');
INSERT INTO `hms`.`staff` (`staffid`, `name`, `post`, `block_assigned`, `email`, `contact`,
`salary`) VALUES ('1002', 'Robinson', 'Warden', 'C', 'robinson@gmail.com', '984405041',
'45000');
INSERT INTO `hms`.`staff` (`staffid`, `name`, `post`, `block_assigned`, `email`, `contact`,
`salary`) VALUES ('1003', 'Shyam Kumar', 'Warden', 'B', 'skumar@gmail.com', '9844001245',
'45000');
INSERT INTO `hms`.`staff` (`staffid`, `name`, `post`, `block_assigned`, `email`, `contact`,
`salary`) VALUES ('1101', 'R.Michael', 'Asst. Warden', 'B', 'michael@gmail.com', '9844115355',
'35000');
INSERT INTO `hms`.`staff` (`staffid`, `name`, `post`, `block_assigned`, `email`, `contact`,
`salary`) VALUES ('1212', 'IshwarKarki', 'Asst. Warden', 'C', 'ikarki@gmail.com', '9845789055',
'35000');
INSERT INTO `hms`.`staff` (`staffid`, `name`, `post`, `block_assigned`, `email`, `contact`,
`salary`) VALUES ('1213', 'KailashZirel', 'Room Clean Manager', 'C', 'kailash@gmail.com',
'9744001454', '25000');
INSERT INTO `hms`.`staff` (`staffid`, `name`, `post`, `block_assigned`, `email`, `contact`,
`salary`) VALUES ('1216', 'Manish Thapa', 'Mess Incharge', 'C', 'manish@gmail.com',
'9431854055', '30000');
INSERT INTO `hms`.`staff` (`staffid`, `name`, `post`, `block_assigned`, `email`, `contact`,
`salary`) VALUES ('1316', 'Brajesh Sharma', 'Mess Incharge', 'B', 'brajesh@gmail.com',
'8854682541', '30000');
```


MySQL Workbench
Local instance wampmysqld64
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
hms
Tables
bill
hostel
Columns
Indexes
Foreign Keys
Triggers
mess
room
staff
student
Views
Stored Procedures
Functions
sakila
sys
Tables
Views
Stored Procedures
Functions
world
Administration Schemas
Information
Table: staff
Columns:
staffid int(10) PK
name varchar(45)
post varchar(45)
block_assigned varchar(45)
email varchar(45)
contact varchar(45)
salary int(20)
Object Info Session
staff
staff - Table
Table Name: staff
Schema: hms
Charset/Collation: utf8mb4 utf8mb4_0900_ai_ci
Engine: InnoDB
Comments:
Column Name Datatype PK NN UQ B UN ZF AI G Default/Expression
staffid INT(10)
name VARCHAR(45) NULL
post VARCHAR(45) NULL
block_assigned VARCHAR(45) NULL
email VARCHAR(45) NULL
contact VARCHAR(45) NULL
salary INT(20) NULL
Column Name:
Charset/Collation:
Comments:
Data Type:
Default:
Storage: Virtual Stored
Primary Key Not Null Unique
Binary Unsigned Zero Fill
Auto Increment Generated
Columns Indexes Foreign Keys Triggers Partitioning Options
Apply Revert
SQLAdditions
Jump to
Automatic context help is disabled
Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
3 18:24:22 SELECT * FROM hms.mess LIMIT 0, 1000 4 row(s) returned 0.016 sec / 0.000 sec
4 18:24:37 SELECT * FROM hms.room LIMIT 0, 1000 6 row(s) returned 0.015 sec / 0.000 sec
5 18:24:51 SELECT * FROM hms.staff LIMIT 0, 1000 8 row(s) returned 0.016 sec / 0.000 sec
Query Completed
Type here to search
ENG
06:24 PM
09-10-2018


MySQL Workbench
Local instance wampmysqld64
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
hms
Tables
bill
hostel
Columns
Indexes
Foreign Keys
Triggers
mess
room
staff
student
Views
Stored Procedures
Functions
sakila
sys
Tables
Views
Stored Procedures
Functions
world
Administration Schemas
Information
Table: staff
Columns:
staffid int(10) PK
name varchar(45)
post varchar(45)
block_assigned varchar(45)
email varchar(45)
contact varchar(45)
salary int(20)
Object Info Session
staff
staff - Table
Limit to 1000 rows
1 SELECT * FROM hms.staff;
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
staffid name post block_ass email contact salary
1001 Jayakumar S Warden C jayakumar@gmail.com 9854022853 45000
1002 Robinson Warden C robinson@gmail.com 984405041 45000
1003 Shyam Kumar Warden B skumar@gmail.com 9844001245 45000
1101 R.Michael Asst. Warden B michael@gmail.com 9844115355 35000
1212 IshwarKarki Asst. Warden C ikarki@gmail.com 9845789055 35000
1213 KailashZirel Room Clean Manager C kailash@gmail.com 9744001454 25000
1216 Manish Thapa Mess Incharge C manish@gmail.com 9431854055 30000
1316 Brajesh Sharma Mess Incharge B brajesh@gmail.com 8854682541 30000
Result Grid
Form Editor
Field Types
Query Stats
staff 1
Apply Revert
SQLAdditions
Jump to
Automatic context help is disabled
Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
3 18:24:22 SELECT * FROM hms.mess LIMIT 0, 1000 4 row(s) returned 0.016 sec / 0.000 sec
4 18:24:37 SELECT * FROM hms.room LIMIT 0, 1000 6 row(s) returned 0.015 sec / 0.000 sec
5 18:24:51 SELECT * FROM hms.staff LIMIT 0, 1000 8 row(s) returned 0.016 sec / 0.000 sec
Query Completed
Type here to search
ENG
06:24 PM
09-10-2018

#Creating a table for student where registration number of a student is the primary key

```
CREATE TABLE `hms`.`student` (
`reg_no` INT(20) NOT NULL,
`name` VARCHAR(45) NULL,
`dob` DATE NULL,
`program` VARCHAR(45) NULL,
`year` INT(10) NULL,
`nationality` VARCHAR(45) NULL,
`address` VARCHAR(45) NULL,
`contact` INT(20) NULL,
`hostel` VARCHAR(45) NULL,
`room_no` INT(10) NULL,
`room_type` VARCHAR(45) NULL,
PRIMARY KEY (`reg_no`));
```

#Inserting values into Student

```
INSERT INTO `hms`.`student` (`reg_no`, `name`, `dob`, `program`, `year`, `nationality`, `address`, `contact`, `hostel`, `room_no`, `room_type`) VALUES ('1001', 'hardik', '23-11-1998', 'CSE', '2016', 'indian', 'madhya pradesh', '9876543210', 'Green Pearl Hostel', '102', 'Twin');

INSERT INTO `hms`.`student` (`reg_no`, `name`, `dob`, `program`, `year`, `nationality`, `address`, `contact`, `hostel`, `room_no`, `room_type`) VALUES ('1002', 'apoorva', '12-12-1997', 'IT', '2016', 'indian', 'jaipur', '8974561230', 'NRI Premium Hostel', '501', 'Twin');

INSERT INTO `hms`.`student` (`reg_no`, `name`, `dob`, `program`, `year`, `nationality`, `address`, `contact`, `hostel`, `room_no`, `room_type`) VALUES ('1003', 'naresh', '24-11-1998', 'IT', '2016', 'indian', 'haryana', '7894561230', 'Green Pearl Hostel', '102', 'Twin');

INSERT INTO `hms`.`student` (`reg_no`, `name`, `dob`, `program`, `year`, `nationality`, `address`, `contact`, `hostel`, `room_no`, `room_type`) VALUES ('1004', 'monika', '02-03-1997', 'SW', '2017', 'indian', 'up', '8459671230', 'Green Pearl Hostel', '103', 'Tripple');
```

INSERT INTO `hms`.`student` (`reg_no`, `name`, `dob`, `program`, `year`, `nationality`, `address`, `contact`, `hostel`, `room_no`, `room_type`) VALUES ('1005', 'abhash', '03-11-1997', 'SW', '2016', 'indian', 'patna', '8745963210', 'NRI Premium Hostel', '501', 'Twin');

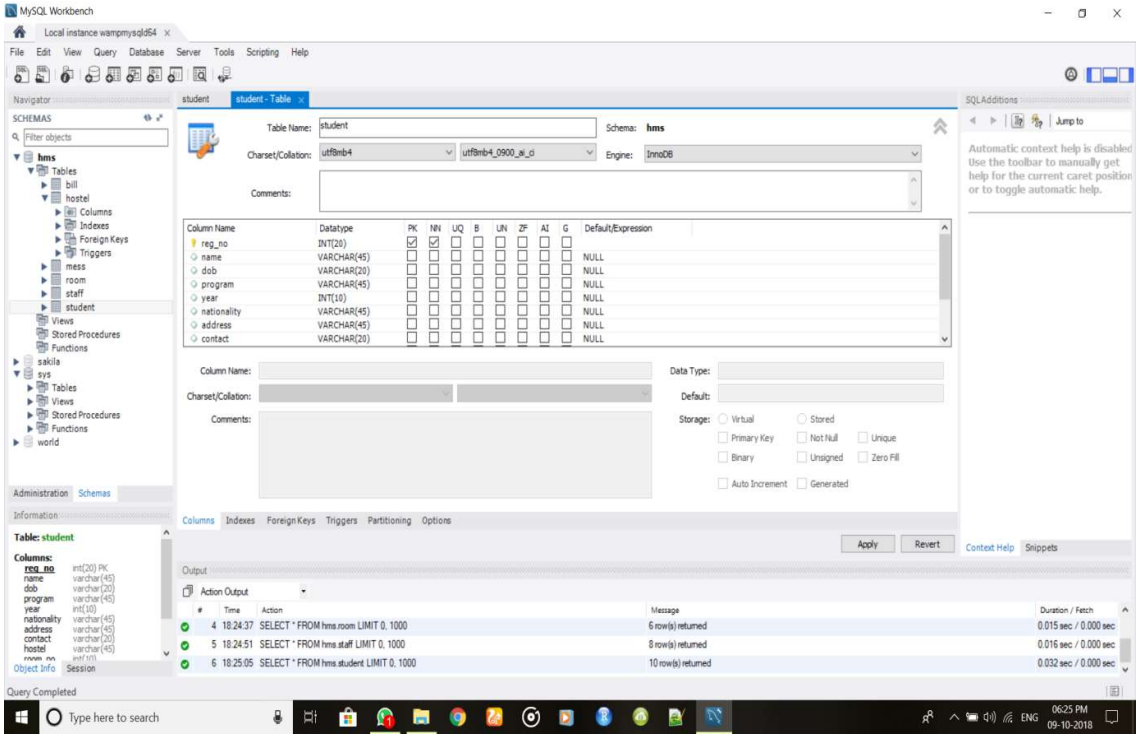
INSERT INTO `hms`.`student` (`reg_no`, `name`, `dob`, `program`, `year`, `nationality`, `address`, `contact`, `hostel`, `room_no`, `room_type`) VALUES ('1006', 'roshni', '01-12-1998', 'CSE', '2017', 'nepal', 'kolkata', '7410258963', 'Green Pearl Hostel', '204', 'Twin');

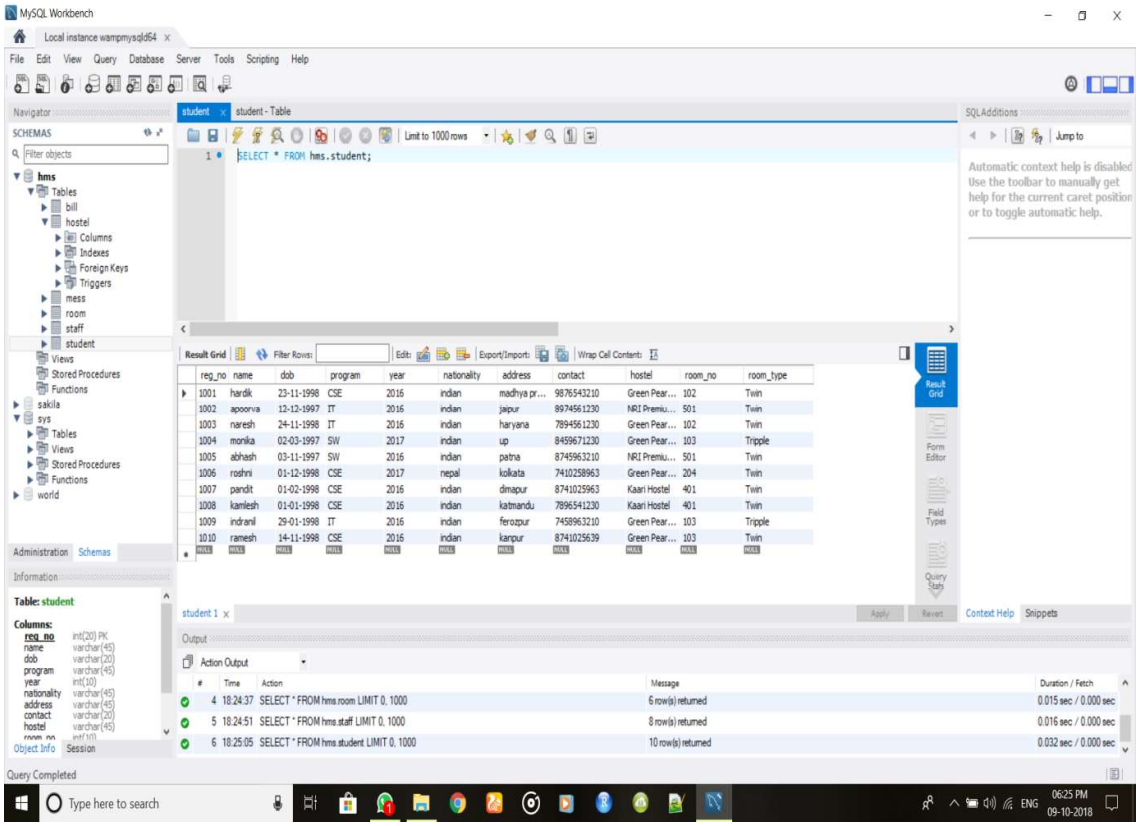
INSERT INTO `hms`.`student` (`reg_no`, `name`, `dob`, `program`, `year`, `nationality`, `address`, `contact`, `hostel`, `room_no`, `room_type`) VALUES ('1007', 'pandit', '01-02-1998', 'CSE', '2016', 'indian', 'dimapur', '8741025963', 'Kaari Hostel', '401', 'Twin');

INSERT INTO `hms`.`student` (`reg_no`, `name`, `dob`, `program`, `year`, `nationality`, `address`, `contact`, `hostel`, `room_no`, `room_type`) VALUES ('1008', 'kamlesh', '01-01-1998', 'CSE', '2016', 'indian', 'katmandu', '7896541230', 'Kaari Hostel', '401', 'Twin');

INSERT INTO `hms`.`student` (`reg_no`, `name`, `dob`, `program`, `year`, `nationality`, `address`, `contact`, `hostel`, `room_no`, `room_type`) VALUES ('1009', 'indranil', '29-01-1998', 'IT', '2016', 'indian', 'ferozpur', '7458963210', 'Green Pearl Hostel', '103', 'Tripple');

INSERT INTO `hms`.`student` (`reg_no`, `name`, `dob`, `program`, `year`, `nationality`, `address`, `contact`, `hostel`, `room_no`, `room_type`) VALUES ('1010', 'ramesh', '14-11-1998', 'CSE', '2016', 'indian', 'kanpur', '8741025639', 'Green Pearl Hostel', '103', 'Twin');


MySQL Workbench
Local instance wampmysqld64
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
hms
Tables
bill
hostel
Columns
Indexes
Foreign Keys
Triggers
mess
room
staff
student
Views
Stored Procedures
Functions
sakila
sys
Tables
Views
Stored Procedures
Functions
world
Administration Schemas
Information
Table: student
Columns:
reg_no int(20) PK
name varchar(45)
dob varchar(20)
program varchar(45)
year int(10)
nationality varchar(45)
address varchar(45)
contact varchar(20)
hostel varchar(45)
Object Info Session
student student - Table
Table Name: student
Schema: hms
Charset/Collation: utf8mb4 utf8mb4_0900_ai_ci
Engine: InnoDB
Comments:
Column Name Datatype PK NN UQ B UN ZF AI G Default/Expression
reg_no INT(20) NULL
name VARCHAR(45) NULL
dob VARCHAR(20) NULL
program VARCHAR(45) NULL
year INT(10) NULL
nationality VARCHAR(45) NULL
address VARCHAR(45) NULL
contact VARCHAR(20) NULL
Column Name:
Charset/Collation:
Comments:
Data Type:
Default:
Storage: Virtual Stored
Primary Key Not Null Unique
Binary Unsigned Zero Fill
Auto Increment Generated
Columns Indexes Foreign Keys Triggers Partitioning Options
Apply Revert
SQLAdditions
Jump to
Automatic context help is disabled
Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
4 18:24:37 SELECT * FROM hms.room LIMIT 0, 1000 6 row(s) returned 0.015 sec / 0.000 sec
5 18:24:51 SELECT * FROM hms.staff LIMIT 0, 1000 8 row(s) returned 0.016 sec / 0.000 sec
6 18:25:05 SELECT * FROM hms.student LIMIT 0, 1000 10 row(s) returned 0.032 sec / 0.000 sec
Query Completed
Type here to search
ENG
06:25 PM
09-10-2018

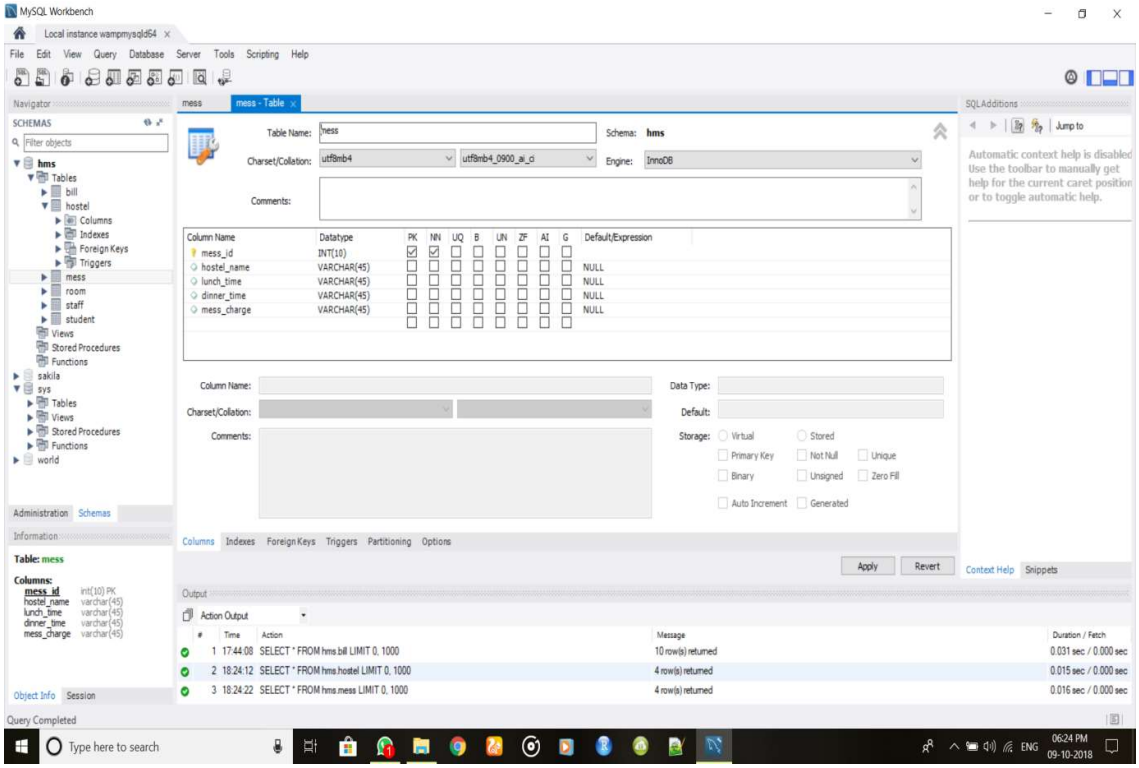

MySQL Workbench
Local instance wampmysqld64
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
hms
Tables
bill
hostel
Columns
Indexes
Foreign Keys
Triggers
mess
room
staff
student
Views
Stored Procedures
Functions
sakila
sys
Tables
Views
Stored Procedures
Functions
world
Administration Schemas
Information
Table: student
Columns:
reg_no int(20) PK
name varchar(45)
dob varchar(20)
program varchar(45)
year int(10)
nationality varchar(45)
address varchar(45)
contact varchar(20)
hostel varchar(45)
Object Info Session
student student - Table
Limit to 1000 rows
1 SELECT * FROM hms.student;
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
reg_no name dob program year nationality address contact hostel room_no room_type
1001 hardik 23-11-1998 CSE 2016 indian madhya pr... 9876543210 Green Pear... 102 Twin
1002 apoorva 12-12-1997 IT 2016 indian jaipur 8974561230 NRI Premiu... 501 Twin
1003 naresh 24-11-1998 IT 2016 indian haryana 7894561230 Green Pear... 102 Twin
1004 monika 02-03-1997 SW 2017 indian up 8459671230 Green Pear... 103 Tripple
1005 abhash 03-11-1997 SW 2016 indian patna 8745963210 NRI Premiu... 501 Twin
1006 roshni 01-12-1998 CSE 2017 nepal kolkata 7410258963 Green Pear... 204 Twin
1007 pandit 01-02-1998 CSE 2016 indian dmapur 8741025963 Kaari Hostel 401 Twin
1008 kamlesh 01-01-1998 CSE 2016 indian katmandu 7896541230 Kaari Hostel 401 Twin
1009 indranil 29-01-1998 IT 2016 indian ferozpur 7458963210 Green Pear... 103 Tripple
1010 ramesh 14-11-1998 CSE 2016 indian kanpur 8741025639 Green Pear... 103 Twin
Result Grid
Form Editor
Field Types
Query Stats
student 1
Apply Revert
SQLAdditions
Jump to
Automatic context help is disabled
Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
4 18:24:37 SELECT * FROM hms.room LIMIT 0, 1000 6 row(s) returned 0.015 sec / 0.000 sec
5 18:24:51 SELECT * FROM hms.staff LIMIT 0, 1000 8 row(s) returned 0.016 sec / 0.000 sec
6 18:25:05 SELECT * FROM hms.student LIMIT 0, 1000 10 row(s) returned 0.032 sec / 0.000 sec
Query Completed
Type here to search
ENG
06:25 PM
09-10-2018

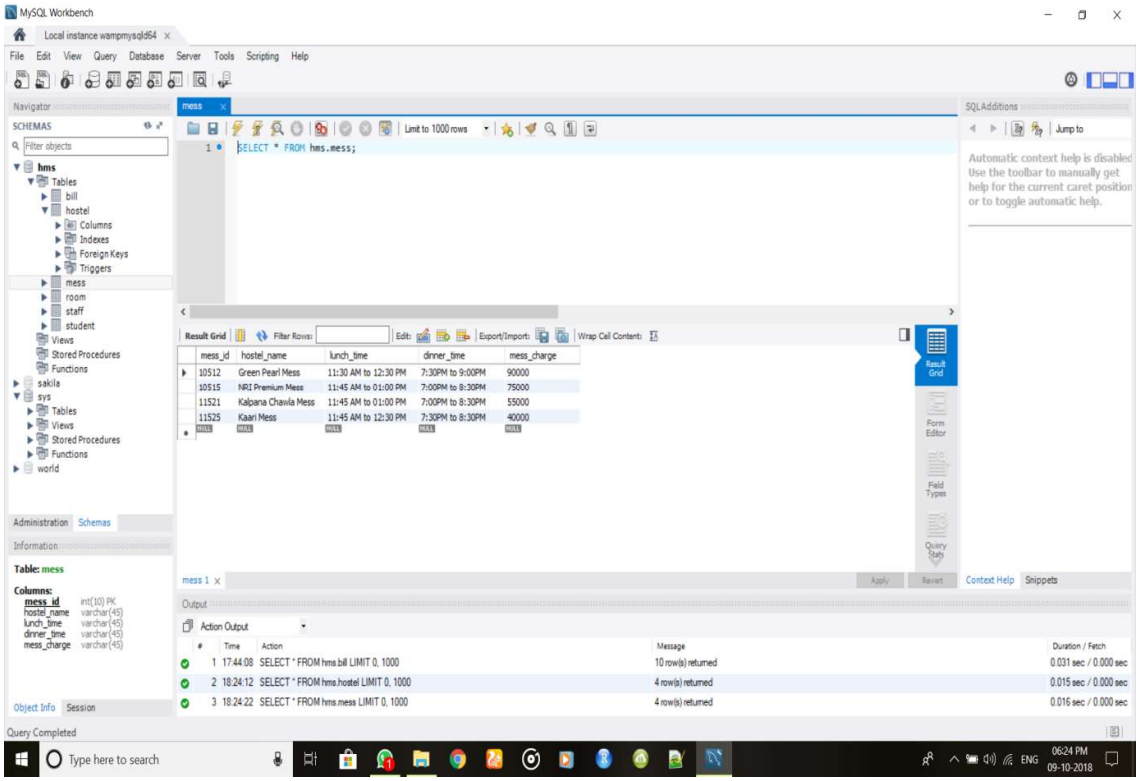
#Creating a table for mess where mess id is the primary key

```
CREATE TABLE `hms`.`mess` (
`mess_id` INT(10) NOT NULL,
`mess_name` VARCHAR(45) NULL,
`lunch_time` VARCHAR(45) NULL,
`dinner_time` VARCHAR(45) NULL,
`mess_charge` VARCHAR(45) NULL,
PRIMARY KEY (`mess_id`));
```

#Inserting values into mess

```
INSERT INTO `hms`.`mess` (`mess_id`, `mess_name`, `lunch_time`, `dinner_time`, `mess_charge`) VALUES ('10512', 'Green Pearl Mess', '11:30 AM to 12:30 PM', '7:30PM to 9:00PM', '90000');

INSERT INTO `hms`.`mess` (`mess_id`, `mess_name`, `lunch_time`, `dinner_time`, `mess_charge`) VALUES ('10515', 'NRI Premium Mess', '11:45 AM to 01:00 PM', '7:00PM to 8:30PM', '75000');

INSERT INTO `hms`.`mess` (`mess_id`, `mess_name`, `lunch_time`, `dinner_time`, `mess_charge`) VALUES ('11521', 'Kalpana Chawla Mess', '11:45 AM to 01:00 PM', '7:00PM to 8:30PM', '55000');

INSERT INTO `hms`.`mess` (`mess_id`, `mess_name`, `lunch_time`, `dinner_time`, `mess_charge`) VALUES ('11525', 'Kaari Mess', '11:45 AM to 12:30 PM', '7:30PM to 8:30PM', '40000');
```


MySQL Workbench
Local instance wampmysqld64
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
hms
Tables
bill
hostel
Columns
Indexes
Foreign Keys
Triggers
mess
room
staff
student
Views
Stored Procedures
Functions
sakila
sys
Tables
Views
Stored Procedures
Functions
world
Administration Schemas
Information
Table: mess
Columns:
mess_id int(10) PK
hostel_name varchar(45)
lunch_time varchar(45)
dinner_time varchar(45)
mess_charge varchar(45)
Object Info Session
mess
mess - Table
Table Name: mess
Schema: hms
Charset/Collation: utf8mb4 utf8mb4_0900_ai_ci
Engine: InnoDB
Comments:
Column Name Datatype PK NN UQ B UN ZF AI G Default/Expression
mess_id INT(10) NULL
hostel_name VARCHAR(45) NULL
lunch_time VARCHAR(45) NULL
dinner_time VARCHAR(45) NULL
mess_charge VARCHAR(45) NULL
Column Name:
Data Type:
Charset/Collation:
Default:
Comments:
Storage: Virtual Stored
Primary Key Not Null Unique
Binary Unsigned Zero Fill
Auto Increment Generated
Columns Indexes Foreign Keys Triggers Partitioning Options
Apply Revert
SQLAdditions
Jump to
Automatic context help is disabled
Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
1 17:44:08 SELECT * FROM hms.bill LIMIT 0, 1000 10 row(s) returned 0.031 sec / 0.000 sec
2 18:24:12 SELECT * FROM hms.hostel LIMIT 0, 1000 4 row(s) returned 0.015 sec / 0.000 sec
3 18:24:22 SELECT * FROM hms.mess LIMIT 0, 1000 4 row(s) returned 0.016 sec / 0.000 sec
Query Completed
Type here to search
ENG
06:24 PM
09-10-2018

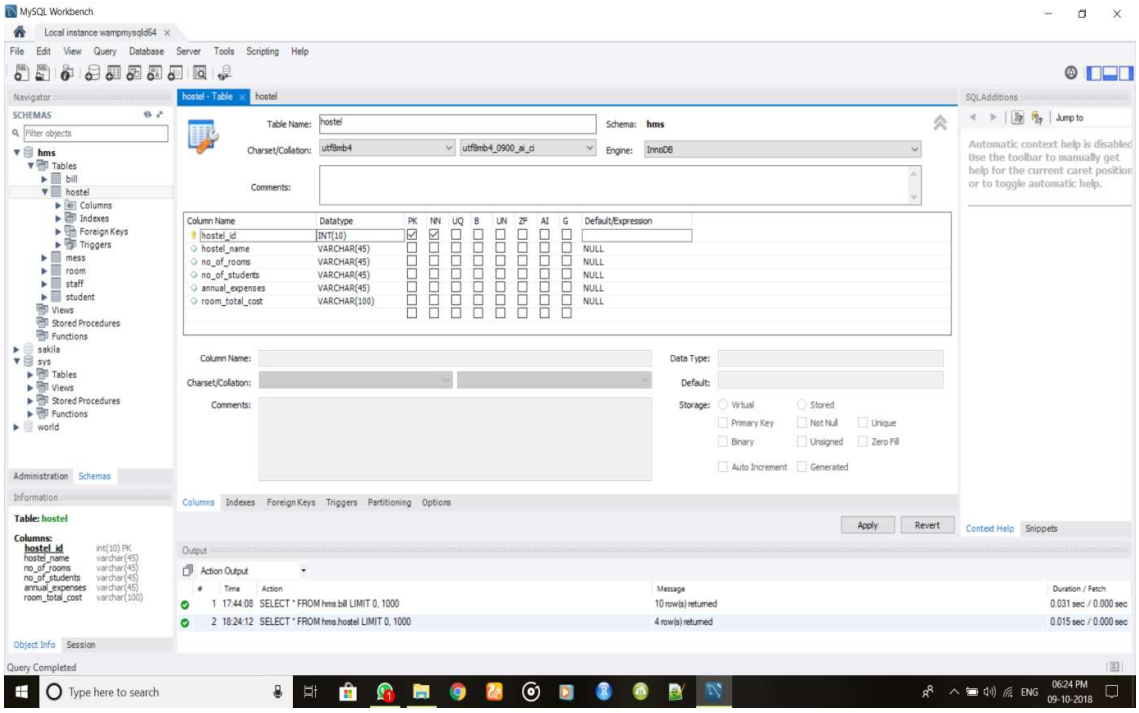

MySQL Workbench
Local instance wampmysqld64
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
hms
Tables
bill
hostel
Columns
Indexes
Foreign Keys
Triggers
mess
room
staff
student
Views
Stored Procedures
Functions
sakila
sys
Tables
Views
Stored Procedures
Functions
world
Administration Schemas
Information
Table: mess
Columns:
mess_id int(10) PK
hostel_name varchar(45)
lunch_time varchar(45)
dinner_time varchar(45)
mess_charge varchar(45)
Object Info Session
mess
Limit to 1000 rows
1 SELECT * FROM hms.mess;
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
mess_id hostel_name lunch_time dinner_time mess_charge
10512 Green Pearl Mess 11:30 AM to 12:30 PM 7:30PM to 9:00PM 90000
10515 NRI Premium Mess 11:45 AM to 01:00 PM 7:00PM to 8:30PM 75000
11521 Kalpana Chawla Mess 11:45 AM to 01:00 PM 7:00PM to 8:30PM 55000
11525 Kaari Mess 11:45 AM to 12:30 PM 7:30PM to 8:30PM 40000
Result Grid
Form Editor
Field Types
Query Stats
mess 1
Apply Revert
SQLAdditions
Jump to
Automatic context help is disabled
Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
1 17:44:08 SELECT * FROM hms.bill LIMIT 0, 1000 10 row(s) returned 0.031 sec / 0.000 sec
2 18:24:12 SELECT * FROM hms.hostel LIMIT 0, 1000 4 row(s) returned 0.015 sec / 0.000 sec
3 18:24:22 SELECT * FROM hms.mess LIMIT 0, 1000 4 row(s) returned 0.016 sec / 0.000 sec
Query Completed
Type here to search
ENG
06:24 PM
09-10-2018

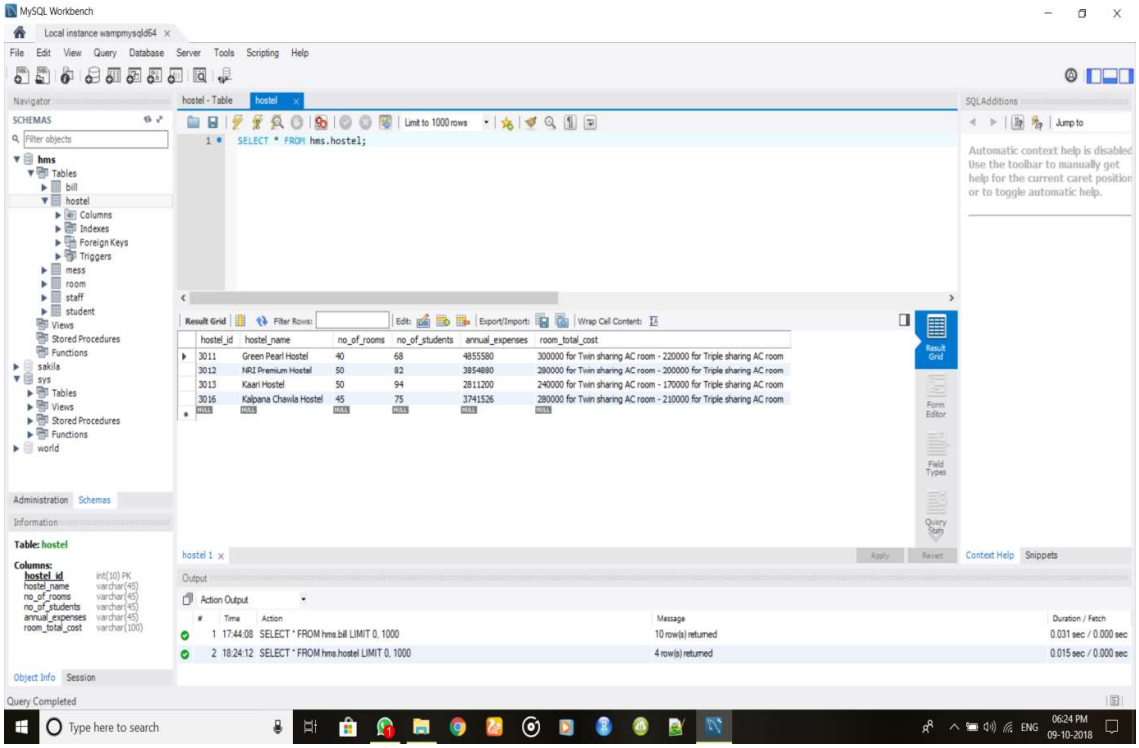
#Creating a table for hostel where hostel id is the primary key

```
CREATE TABLE `hms`.`hostel` (
`hostel_id` INT(10) NOT NULL,
`hostel_name` VARCHAR(45) NULL,
`no_of_rooms` VARCHAR(45) NULL,
`no_of_students` VARCHAR(45) NULL,
`annual_expenses` VARCHAR(45) NULL,
`room_total_cost` VARCHAR(45) NULL,
PRIMARY KEY (`hostel_id`));
```

#Inserting values into hostel

```
INSERT INTO `hms`.`hostel` (`hostel_id`, `hostel_name`, `no_of_rooms`, `no_of_students`, `annual_expenses`, `room_total_cost`) VALUES ('3011', 'Green Pearl Hostel', '40', '68', '4855580', '300000 for Twin sharing AC room - 220000 for Triple sharing AC room');

INSERT INTO `hms`.`hostel` (`hostel_id`, `hostel_name`, `no_of_rooms`, `no_of_students`, `annual_expenses`, `room_total_cost`) VALUES ('3012', 'NRI Premium Hostel', '50', '82', '3854880', '280000 for Twin sharing AC room - 200000 for Triple sharing AC room');

INSERT INTO `hms`.`hostel` (`hostel_id`, `hostel_name`, `no_of_rooms`, `no_of_students`, `annual_expenses`, `room_total_cost`) VALUES ('3013', 'Kaari Hostel', '50', '94', '2811200', '240000 for Twin sharing AC room - 170000 for Triple sharing AC room');

INSERT INTO `hms`.`hostel` (`hostel_id`, `hostel_name`, `no_of_rooms`, `no_of_students`, `annual_expenses`, `room_total_cost`) VALUES ('3016', 'Kalpana Chawla Hostel', '45', '75', '3741526', '280000 for Twin sharing AC room - 210000 for Triple sharing AC room');
```


MySQL Workbench
Local instance wampmysqld64
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
hms
Tables
bill
hostel
Columns
Indexes
Foreign Keys
Triggers
mess
room
staff
student
Views
Stored Procedures
Functions
sakila
sys
world
Administration Schemas
Information
Table: hostel
Columns:
hostel_id int(10) PK
hostel_name varchar(45)
no_of_rooms varchar(45)
no_of_students varchar(45)
annual_expenses varchar(45)
room_total_cost varchar(100)
Object Info Session
hostel - Table
hostel
Table Name: hostel
Schema: hms
Charset/Collation: utf8mb4 utf8mb4_0900_ai_ci
Engine: InnoDB
Comments:
Column Name Datatype PK NN UQ B UN ZF AI G Default/Expression
hostel_id INT(10)
hostel_name VARCHAR(45) NULL
no_of_rooms VARCHAR(45) NULL
no_of_students VARCHAR(45) NULL
annual_expenses VARCHAR(45) NULL
room_total_cost VARCHAR(100) NULL
Column Name:
Charset/Collation:
Comments:
Data Type:
Default:
Storage: Virtual Stored
Primary Key Not Null Unique
Binary Unsigned Zero Fill
Auto Increment Generated
Columns Indexes Foreign Keys Triggers Partitioning Options
Apply Revert
SQLAdditions
Jump to
Automatic context help is disabled
Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
1 17:44:08 SELECT * FROM hms.bill LIMIT 0, 1000 10 row(s) returned 0.031 sec / 0.000 sec
2 18:24:12 SELECT * FROM hms.hostel LIMIT 0, 1000 4 row(s) returned 0.015 sec / 0.000 sec
Query Completed
Type here to search
ENG
06:24 PM
09-10-2018


MySQL Workbench
Local instance wampmysqld64
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
hms
Tables
bill
hostel
Columns
Indexes
Foreign Keys
Triggers
mess
room
staff
student
Views
Stored Procedures
Functions
sakila
sys
world
Administration Schemas
Information
Table: hostel
Columns:
hostel_id int(10) PK
hostel_name varchar(45)
no_of_rooms varchar(45)
no_of_students varchar(45)
annual_expenses varchar(45)
room_total_cost varchar(100)
Object Info Session
hostel - Table
hostel
Limit to 1000 rows
1 SELECT * FROM hms.hostel;
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
hostel_id hostel_name no_of_rooms no_of_students annual_expenses room_total_cost
3011 Green Pearl Hostel 40 68 4855580 300000 for Twin sharing AC room - 220000 for Triple sharing AC room
3012 NRI Premium Hostel 50 82 3854880 280000 for Twin sharing AC room - 200000 for Triple sharing AC room
3013 Kaari Hostel 50 94 2811200 240000 for Twin sharing AC room - 170000 for Triple sharing AC room
3016 Kalpana Chawla Hostel 45 75 3741526 280000 for Twin sharing AC room - 210000 for Triple sharing AC room
Result Grid
Form Editor
Field Types
Query Stats
hostel 1
Apply Revert
SQLAdditions
Jump to
Automatic context help is disabled
Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
1 17:44:08 SELECT * FROM hms.bill LIMIT 0, 1000 10 row(s) returned 0.031 sec / 0.000 sec
2 18:24:12 SELECT * FROM hms.hostel LIMIT 0, 1000 4 row(s) returned 0.015 sec / 0.000 sec
Query Completed
Type here to search
ENG
06:24 PM
09-10-2018

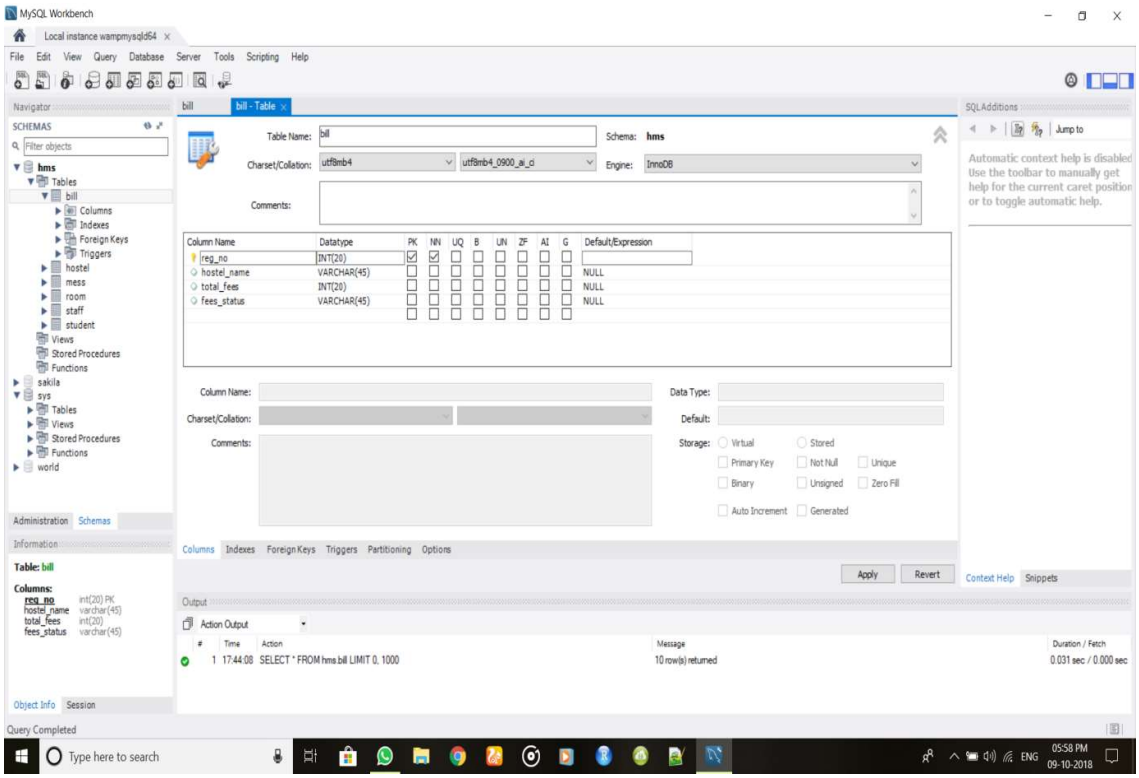
#Creating a table for bill where registration number of a student is a primary key

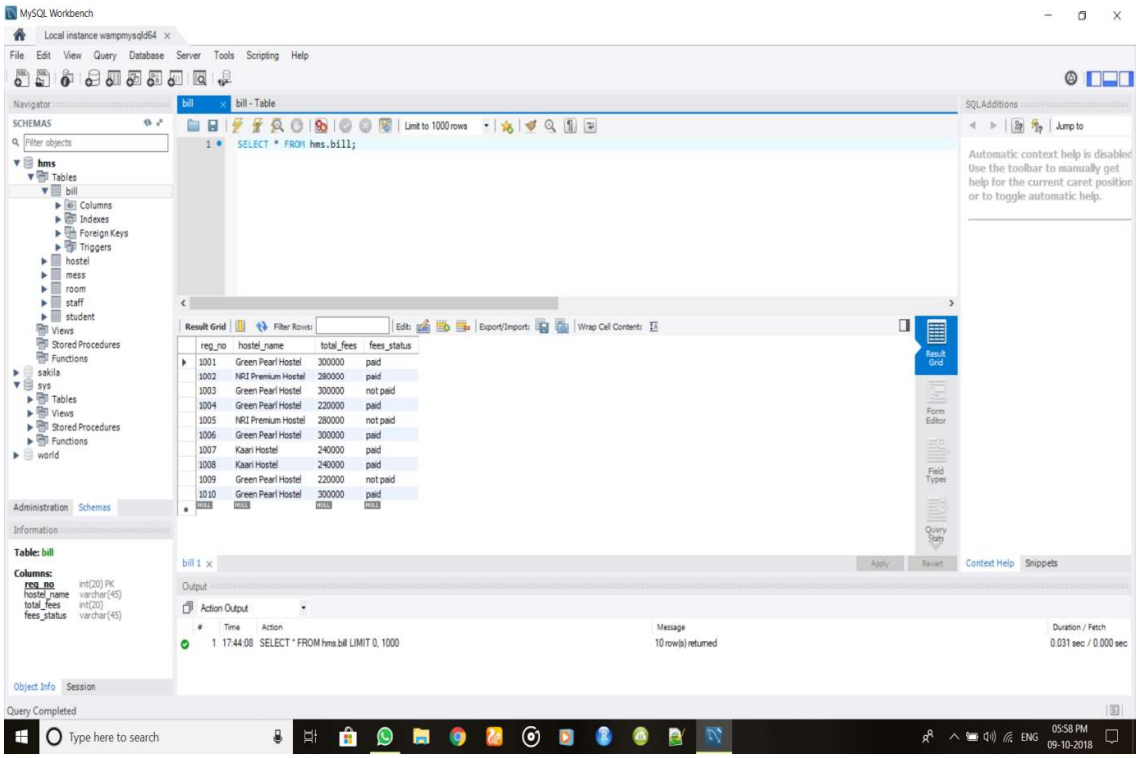
```
CREATE TABLE `hms`.`bill` (
`reg_no` INT(20) NOT NULL,
`hostel_name` VARCHAR(45) NULL,
`total_fees` INT(20) NULL,
`fees_status` VARCHAR(45) NULL,
PRIMARY KEY (`reg_no`));
```

#Inserting values into bill

```
INSERT INTO `hms`.`bill` (`reg_no`, `hostel_name`, `total_fees`, `fees_status`) VALUES ('1002', 'NRI Premium Hostel', '280000', 'paid');

INSERT INTO `hms`.`bill` (`reg_no`, `hostel_name`, `total_fees`, `fees_status`) VALUES ('1003', 'Green Pearl Hostel', '300000', 'not paid');

INSERT INTO `hms`.`bill` (`reg_no`, `hostel_name`, `total_fees`, `fees_status`) VALUES ('1004', 'Green Pearl Hostel', '220000', 'paid');

INSERT INTO `hms`.`bill` (`reg_no`, `hostel_name`, `total_fees`, `fees_status`) VALUES ('1005', 'NRI Premium Hostel', '280000', 'not paid');

INSERT INTO `hms`.`bill` (`reg_no`, `hostel_name`, `total_fees`, `fees_status`) VALUES ('1006', 'Green Pearl Hostel', '300000', 'paid');

INSERT INTO `hms`.`bill` (`reg_no`, `hostel_name`, `total_fees`, `fees_status`) VALUES ('1008', 'Kaari Hostel', '240000', 'paid');

INSERT INTO `hms`.`bill` (`reg_no`, `hostel_name`, `total_fees`, `fees_status`) VALUES ('1009', 'Green Pearl Hostel', '220000', 'not paid');

INSERT INTO `hms`.`bill` (`reg_no`, `hostel_name`, `total_fees`, `fees_status`) VALUES ('1010', 'Green Pearl Hostel', '300000', 'paid');

INSERT INTO `hms`.`bill` (`reg_no`, `hostel_name`, `total_fees`, `fees_status`) VALUES ('1001', 'Green Pearl Hostel', '300000', 'paid');
```

INSERT INTO `hms`.`bill` (`reg_no`, `hostel_name`, `total_fees`, `fees_status`) VALUES ('1007', 'Kaari Hostel', '240000', 'paid');



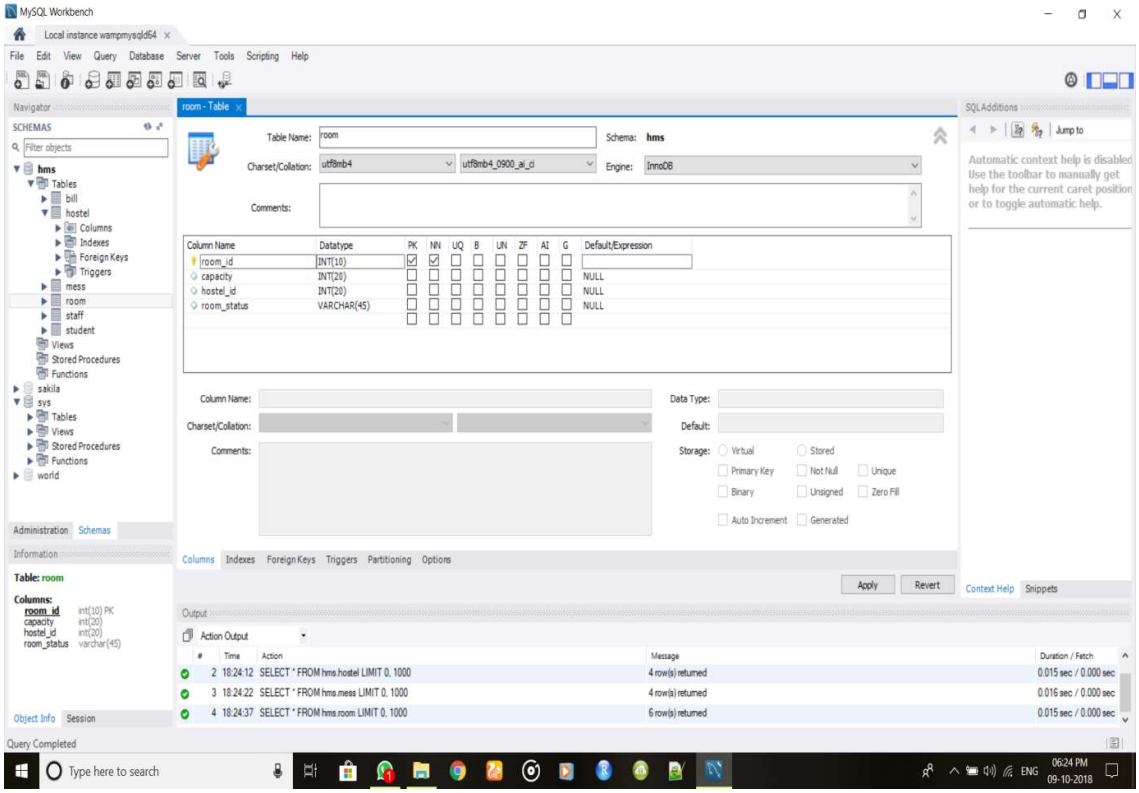


#Creating a table for room where room id is a primary key

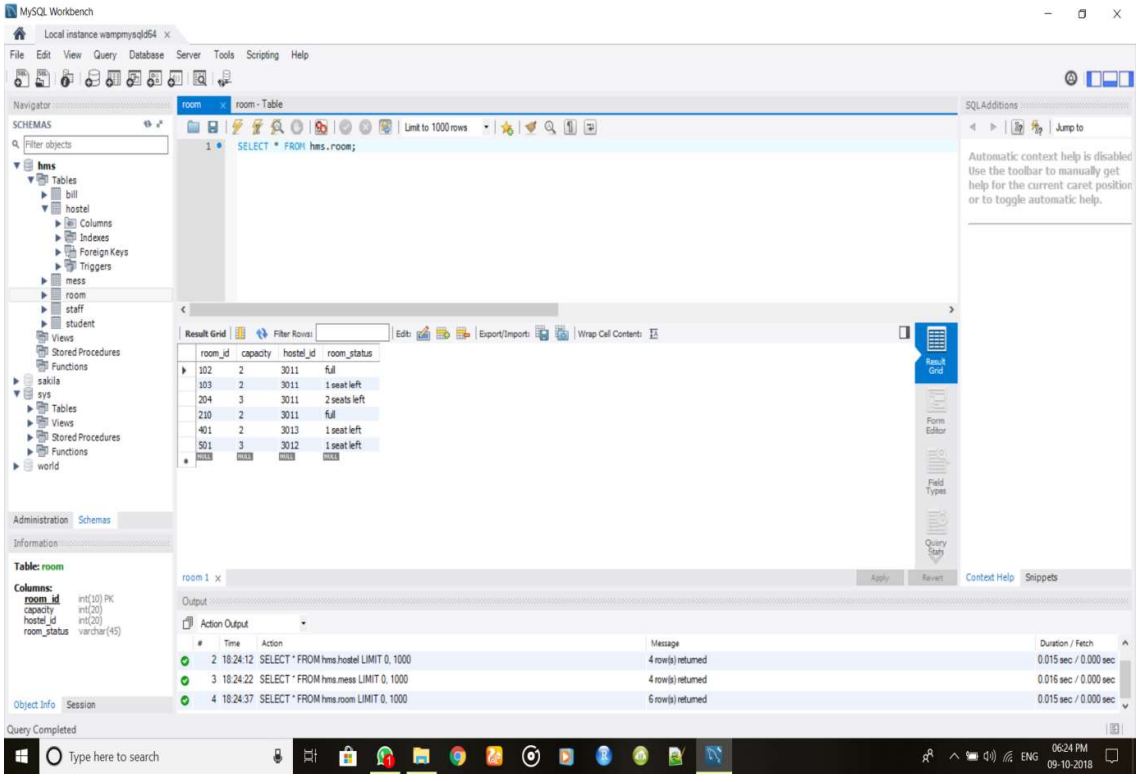
```
CREATE TABLE `hms`.`room` (
`room_id` INT(10) NOT NULL,
`capacity` INT(20) NULL,
`hostel_id` INT(20) NULL,
`room_status` VARCHAR(45) NULL,
PRIMARY KEY (`room_id`));
```

#Inserting values into room

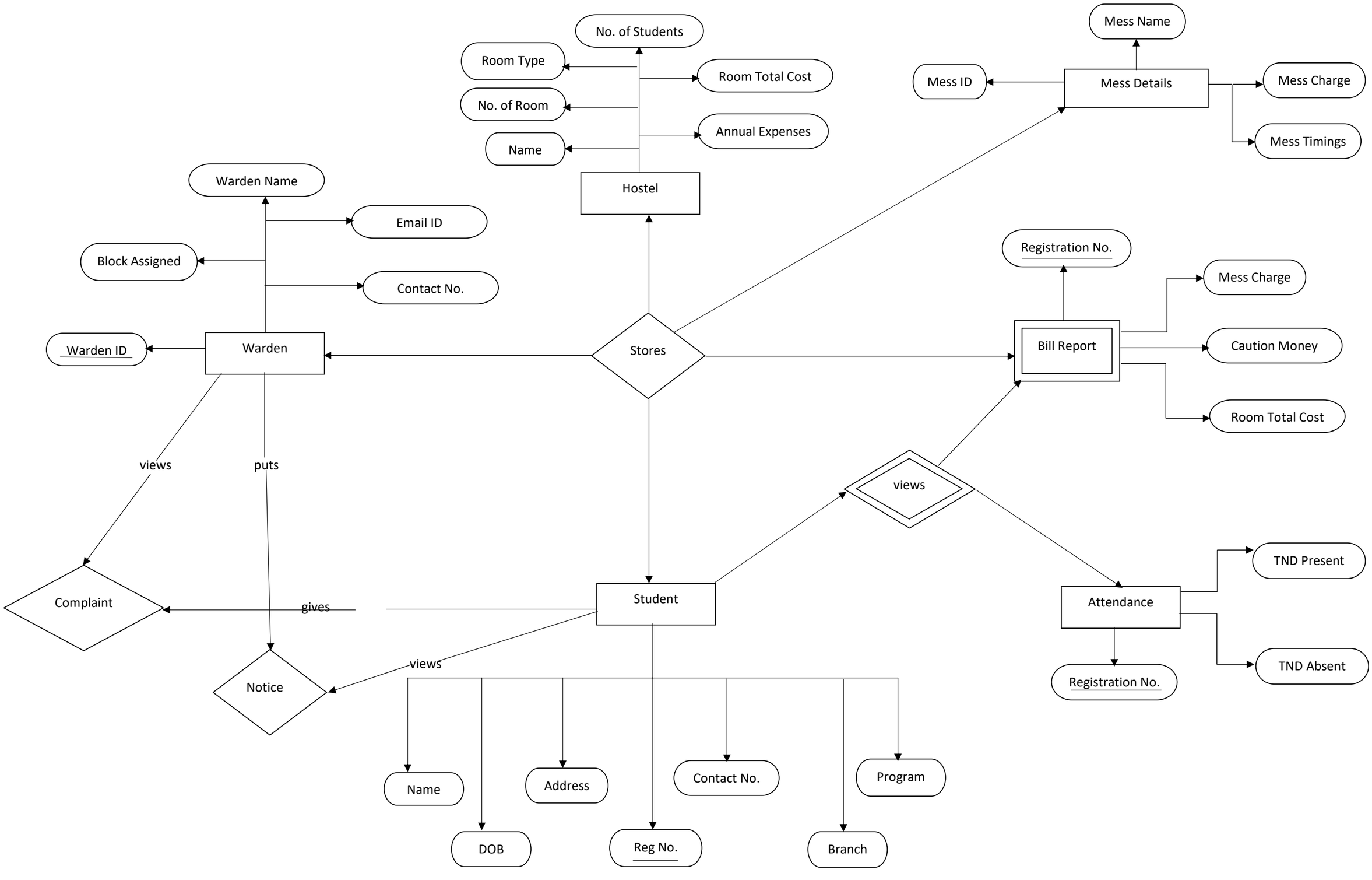
```
INSERT INTO `hms`.`room` (`reg_no`, `hostel_name`, `total_fees`, `fees_status`) VALUES ('102', '2', '3011', 'full')

INSERT INTO `hms`.`room` (`room_id`, `capacity`, `hostel_id`, `room_status`) VALUES ('103', '2', '3011', '1 seat left')

INSERT INTO `hms`.`room` (`room_id`, `capacity`, `hostel_id`, `room_status`) VALUES ('204', '3', '3011', '2 seats left')

INSERT INTO `hms`.`room` (`room_id`, `capacity`, `hostel_id`, `room_status`) VALUES ('501', '3', '3012', '1 seat left')

INSERT INTO `hms`.`room` (`room_id`, `capacity`, `hostel_id`, `room_status`) VALUES ('401', '2', '3013', '1 seat left')

INSERT INTO `hms`.`room` (`room_id`, `capacity`, `hostel_id`, `room_status`) VALUES ('210', '2', '3011', 'full')
```


MySQL Workbench
Local instance wampmysqld64
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
hms
Tables
bill
hostel
Columns
Indexes
Foreign Keys
Triggers
mess
room
staff
student
Views
Stored Procedures
Functions
sakila
sys
Tables
Views
Stored Procedures
Functions
world
Administration Schemas
Information
Table: room
Columns:
room_id int(10) PK
capacity int(20)
hostel_id int(20)
room_status varchar(45)
Object Info Session
room - Table
Table Name: room
Schema: hms
Charset/Collation: utf8mb4 utf8mb4_0900_ai_ci
Engine: InnoDB
Comments:
Column Name Datatype PK NN UQ B UN ZF AI G Default/Expression
room_id INT(10)
capacity INT(20) NULL
hostel_id INT(20) NULL
room_status VARCHAR(45) NULL
Column Name:
Charset/Collation:
Comments:
Data Type:
Default:
Storage: Virtual Stored
Primary Key Not Null Unique
Binary Unsigned Zero Fill
Auto Increment Generated
Columns Indexes Foreign Keys Triggers Partitioning Options
Apply Revert
SQLAdditions
Jump to
Automatic context help is disabled
Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
2 18:24:12 SELECT * FROM hms.hostel LIMIT 0, 1000 4 row(s) returned 0.015 sec / 0.000 sec
3 18:24:22 SELECT * FROM hms.mess LIMIT 0, 1000 4 row(s) returned 0.016 sec / 0.000 sec
4 18:24:37 SELECT * FROM hms.room LIMIT 0, 1000 6 row(s) returned 0.015 sec / 0.000 sec
Query Completed
Type here to search
ENG
06:24 PM
09-10-2018

,


MySQL Workbench
Local instance wampmysqld64
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
hms
Tables
bill
hostel
Columns
Indexes
Foreign Keys
Triggers
mess
room
staff
student
Views
Stored Procedures
Functions
sakila
sys
Tables
Views
Stored Procedures
Functions
world
Administration Schemas
Information
Table: room
Columns:
room_id int(10) PK
capacity int(20)
hostel_id int(20)
room_status varchar(45)
Object Info Session
room room - Table
Limit to 1000 rows
1 SELECT * FROM hms.room;
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
room_id capacity hostel_id room_status
102 2 3011 full
103 2 3011 1 seat left
204 3 3011 2 seats left
210 2 3011 full
401 2 3013 1 seat left
501 3 3012 1 seat left
Result Grid
Form Editor
Field Types
Query Stats
room 1
Apply Revert
SQLAdditions
Jump to
Automatic context help is disabled
Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
2 18:24:12 SELECT * FROM hms.hostel LIMIT 0, 1000 4 row(s) returned 0.015 sec / 0.000 sec
3 18:24:22 SELECT * FROM hms.mess LIMIT 0, 1000 4 row(s) returned 0.016 sec / 0.000 sec
4 18:24:37 SELECT * FROM hms.room LIMIT 0, 1000 6 row(s) returned 0.015 sec / 0.000 sec
Query Completed
Type here to search
ENG
06:24 PM
09-10-2018

HOSTEL MANAGEMENT SYSTEM



BASIC QUERIES

Query:

```
mysql>  select * from bill where fees_status='Paid';
```

Output:

```
+--------+--------------------+------------+-------------+
| reg_no | hostel_name        | total_fees | fees_status |
+--------+--------------------+------------+-------------+
|   1001 | Green Pearl Hostel |     300000 | paid        |
|   1002 | NRI Premium Hostel |     280000 | paid        |
|   1004 | Green Pearl Hostel |     220000 | paid        |
|   1006 | Green Pearl Hostel |     300000 | paid        |
|   1007 | Kaari Hostel       |     240000 | paid        |
|   1008 | Kaari Hostel       |     240000 | paid        |
|   1010 | Green Pearl Hostel |     300000 | paid        |
+--------+--------------------+------------+-------------+
7 rows in set (0.00 sec)
```

Query:

```
mysql> select sum(mess_charge)from mess;
```

Output:

```
+------------------+
| sum(mess_charge) |
+------------------+
|           260000 |
+------------------+
1 row in set (0.00 sec)
```

Query:

```
mysql> select count(name) from student;
```

Output:

```
+-------------+
| count(name) |
+-------------+
|          10 |
+-------------+
1 row in set (0.00 sec)
```

Query:

```
mysql> select count(name) from staff;
```

Output:

```
+-------------+
| count(name) |
+-------------+
|           8 |
+-------------+
1 row in set (0.00 sec)
```

Query:

```
mysql> select * from student where program='CSE';
```

Output:

```
+--------+---------+------------+---------+------+-------------+----------------+--
----------+--------------------+---------+-----------+
| reg_no | name    | dob        | program | year | nationality | address        |
contact    | hostel             | room_no | room_type |
+--------+---------+------------+---------+------+-------------+----------------+--
----------+--------------------+---------+-----------+
|   1001 | hardik  | 23-11-1998 | CSE     | 2016 | indian      | madhya pradesh |
9876543210 | Green Pearl Hostel |     102 | Twin      |
|   1006 | roshni  | 01-12-1998 | CSE     | 2017 | nepal       | kolkata        |
7410258963 | Green Pearl Hostel |     204 | Twin      |
|   1007 | pandit  | 01-02-1998 | CSE     | 2016 | indian      | dimapur        |
8741025963 | Kaari Hostel       |     401 | Twin      |
|   1008 | kamlesh | 01-01-1998 | CSE     | 2016 | indian      | katmandu       |
7896541230 | Kaari Hostel       |     401 | Twin      |
|   1010 | ramesh  | 14-11-1998 | CSE     | 2016 | indian      | kanpur         |
8741025639 | Green Pearl Hostel |     103 | Twin      |
+--------+---------+------------+---------+------+-------------+----------------+--
----------+--------------------+---------+-----------+
5 rows in set (0.00 sec)
```

Query:

```
mysql> select name,reg_no,program from student where nationality='nepal';
```

Output:

```
+--------+--------+---------+
| name   | reg_no | program |
+--------+--------+---------+
| roshni |   1006 | CSE     |
+--------+--------+---------+
1 row in set (0.00 sec)
```

BASIC QUERIES

Query:

```
mysql> select staffid,name,post from staff ORDER BY staffid;
```

Output:

```
+---------+---------------+--------------------+
| staffid | name          | post               |
+---------+---------------+--------------------+
|    1001 | Jayakumar S   | Warden             |
|    1002 | Robinson      | Warden             |
|    1003 | Shyam Kumar   | Warden             |
|    1101 | R.Michael     | Asst. Warden       |
|    1212 | IshwarKarki   | Asst. Warden       |
|    1213 | KailashZirel  | Room Clean Manager |
|    1216 | Manish Thapa  | Mess Incharge      |
|    1316 | Brajesh Sharma | Mess Incharge     |
+---------+---------------+--------------------+
8 rows in set (0.00 sec)
```

Query:

```
mysql>  select staffid,name,salary from staff where salary between 25000 and 40000;
```

Output:

```
+---------+---------------+--------+
| staffid | name          | salary |
+---------+---------------+--------+
|    1101 | R.Michael     |  35000 |
|    1212 | IshwarKarki   |  35000 |
|    1213 | KailashZirel  |  25000 |
|    1216 | Manish Thapa  |  30000 |
|    1316 | Brajesh Sharma |  30000 |
+---------+---------------+--------+
5 rows in set (0.00 sec)
```

Query:

```
mysql> select hostel_id,hostel_name,no_of_students from hostel ORDER BY
no_of_students;
```

Output:

```
+-----------+----------------------+----------------+
| hostel_id | hostel_name          | no_of_students |
+-----------+----------------------+----------------+
```

```
|      3011 | Green Pearl Hostel    | 68             |
|      3016 | Kalpana Chawla Hostel | 75             |
|      3012 | NRI Premium Hostel    | 82             |
|      3013 | Kaari Hostel          | 94             |
+-----------+-----------------------+----------------+
4 rows in set (0.00 sec)
```

Query:

```
mysql> select * from room where room_status='1 seat left';
```

Output:

```
+---------+----------+-----------+-------------+
| room_id | capacity | hostel_id | room_status |
+---------+----------+-----------+-------------+
|     103 |        2 |      3011 | 1 seat left |
|     401 |        2 |      3013 | 1 seat left |
|     501 |        3 |      3012 | 1 seat left |
+---------+----------+-----------+-------------+
3 rows in set (0.00 sec)
```

Query:

```
mysql> select avg(total_fees) from  bill;
```

Output:

```
+-----------------+
| avg(total_fees) |
+-----------------+
|     268000.0000 |
+-----------------+
1 row in set (0.00 sec)
```

Query:

```
mysql> select max(mess_charge) from mess;
```

Output:

```
+------------------+
| max(mess_charge) |
+------------------+
| 90000            |
+------------------+
1 row in set (0.00 sec)
```

BASIC QUERIES

Query:

```
mysql> select avg(salary),max(salary),min(salary),sum(salary) from staff;
```

Output:

```
+-------------+-------------+-------------+-------------+
| avg(salary) | max(salary) | min(salary) | sum(salary) |
+-------------+-------------+-------------+-------------+
|  36250.0000 |       45000 |       25000 |      290000 |
+-------------+-------------+-------------+-------------+
1 row in set (0.00 sec)
```

Query:

```
mysql> select s.name,s.reg_no,b.total_fees,b.fees_status from student s join bill b
on(s.reg_no=b.reg_no);
```

Output:

```
+----------+--------+------------+-------------+
| name     | reg_no | total_fees | fees_status |
+----------+--------+------------+-------------+
| hardik   |   1001 |     300000 | paid        |
| apoorva  |   1002 |     280000 | paid        |
| naresh   |   1003 |     300000 | not paid    |
| monika   |   1004 |     220000 | paid        |
| abhash   |   1005 |     280000 | not paid    |
| roshni   |   1006 |     300000 | paid        |
| pandit   |   1007 |     240000 | paid        |
| kamlesh  |   1008 |     240000 | paid        |
| indranil |   1009 |     220000 | not paid    |
| ramesh   |   1010 |     300000 | paid        |
+----------+--------+------------+-------------+
10 rows in set (0.00 sec)
```

Query:

```
mysql> select s.name,s.reg_no,s.program,b.total_fees from student s join bill b
on(s.reg_no=b.reg_no) and s.reg_no=1001;
```

Output:

```
+--------+--------+---------+------------+
| name   | reg_no | program | total_fees |
+--------+--------+---------+------------+
| hardik |   1001 | CSE     |     300000 |
+--------+--------+---------+------------+
1 row in set (0.00 sec)
```

Query:

```
mysql> select s.name,s.reg_no,s.program,b.total_fees from student s join bill b
on(s.reg_no=b.reg_no) where s.reg_no=1008;
```

Output:

```
+---------+--------+---------+------------+
| name    | reg_no | program | total_fees |
+---------+--------+---------+------------+
```

```
| kamlesh |   1008 | CSE     |     240000 |
+---------+--------+---------+------------+
1 row in set (0.00 sec)
```

Query:

```
mysql> select s.name,s.reg_no,b.total_fees,b.fees_status from student s left outer
join bill b on(s.reg_no=b.reg_no);
```

Output:

```
+----------+--------+------------+-------------+
| name     | reg_no | total_fees | fees_status |
+----------+--------+------------+-------------+
| hardik   |   1001 |     300000 | paid        |
| apoorva  |   1002 |     280000 | paid        |
| naresh   |   1003 |     300000 | not paid    |
| monika   |   1004 |     220000 | paid        |
| abhash   |   1005 |     280000 | not paid    |
| roshni   |   1006 |     300000 | paid        |
| pandit   |   1007 |     240000 | paid        |
| kamlesh  |   1008 |     240000 | paid        |
| indranil |   1009 |     220000 | not paid    |
| ramesh   |   1010 |     300000 | paid        |
+----------+--------+------------+-------------+
10 rows in set (0.00 sec)
```

Query:

```
mysql> select s.name,s.reg_no,s.address,b.total_fees from student s right outer
join bill b on(s.reg_no=b.reg_no);
```

Output:

```
+----------+--------+----------------+------------+
| name     | reg_no | address        | total_fees |
+----------+--------+----------------+------------+
| hardik   |   1001 | madhya pradesh |     300000 |
| apoorva  |   1002 | jaipur         |     280000 |
| naresh   |   1003 | haryana        |     300000 |
| monika   |   1004 | up             |     220000 |
| abhash   |   1005 | patna          |     280000 |
| roshni   |   1006 | kolkata        |     300000 |
| pandit   |   1007 | dimapur        |     240000 |
| kamlesh  |   1008 | katmandu       |     240000 |
| indranil |   1009 | ferozpur       |     220000 |
| ramesh   |   1010 | kanpur         |     300000 |
+----------+--------+----------------+------------+
```

10 rows in set (0.00 sec)

Query:

```
mysql> select h.hostel_id,m.hostel_name from hostel h cross join mess m;
```

Output:

```
+-----------+---------------------+
| hostel_id | hostel_name         |
+-----------+---------------------+
|      3011 | Green Pearl Mess    |
|      3011 | NRI Premium Mess    |
|      3011 | Kalpana Chawla Mess |
|      3011 | Kaari Mess          |
|      3012 | Green Pearl Mess    |
|      3012 | NRI Premium Mess    |
|      3012 | Kalpana Chawla Mess |
|      3012 | Kaari Mess          |
|      3013 | Green Pearl Mess    |
|      3013 | NRI Premium Mess    |
|      3013 | Kalpana Chawla Mess |
|      3013 | Kaari Mess          |
|      3016 | Green Pearl Mess    |
|      3016 | NRI Premium Mess    |
|      3016 | Kalpana Chawla Mess |
|      3016 | Kaari Mess          |
+-----------+---------------------+
16 rows in set (0.00 sec)
```

SUBQUERIES

Query:

```
mysql> select staffid,name,contact,salary from staff where salary >(select salary
from staff where staffid='1212');
```

Output:

```
+---------+-------------+------------+--------+
| staffid | name        | contact    | salary |
+---------+-------------+------------+--------+
|    1001 | Jayakumar S | 9854022853 |  45000 |
|    1002 | Robinson    | 984405041  |  45000 |
|    1003 | Shyam Kumar | 9844001245 |  45000 |
+---------+-------------+------------+--------+
3 rows in set (0.00 sec)
```

Query:

```
mysql> select staffid,name,contact,salary from staff where salary <(select salary
from staff where staffid='1002');
```

Output:

```
+---------+----------------+------------+--------+
| staffid | name           | contact    | salary |
+---------+----------------+------------+--------+
|    1101 | R.Michael      | 9844115355 |  35000 |
|    1212 | IshwarKarki    | 9845789055 |  35000 |
|    1213 | KailashZirel   | 9744001454 |  25000 |
|    1216 | Manish Thapa   | 9431854055 |  30000 |
|    1316 | Brajesh Sharma | 8854682541 |  30000 |
+---------+----------------+------------+--------+
5 rows in set (0.00 sec)
```

Query:

```
mysql> select staffid,post,block_assigned from staff where salary <(select salary
from staff where staffid='1002');
```

Output:

```
+---------+--------------------+----------------+
| staffid | post               | block_assigned |
+---------+--------------------+----------------+
|    1101 | Asst. Warden       | B              |
|    1212 | Asst. Warden       | C              |
|    1213 | Room Clean Manager | C              |
```

```
|    1216 | Mess Incharge      | C              |
|    1316 | Mess Incharge      | B              |
+---------+--------------------+----------------+
5 rows in set (0.00 sec)
```

Query:

```
mysql> select hostel_id,hostel_name,no_of_students from hostel where no_of_students
>(select no_of_students from hostel where hostel_id='3011');
```

Output:

```
+-----------+----------------------+----------------+
| hostel_id | hostel_name          | no_of_students |
+-----------+----------------------+----------------+
|      3012 | NRI Premium Hostel   | 82             |
|      3013 | Kaari Hostel         | 94             |
|      3016 | Kalpana Chawla Hostel | 75            |
+-----------+----------------------+----------------+
3 rows in set (0.00 sec)
```

Query:

```
mysql> select hostel_id,hostel_name,no_of_students,annual_expenses from hostel
where no_of_students >(select no_of_students from hostel where hostel_id='3011');
```

Output:

```
+-----------+----------------------+----------------+-----------------+
| hostel_id | hostel_name          | no_of_students | annual_expenses |
+-----------+----------------------+----------------+-----------------+
|      3012 | NRI Premium Hostel   | 82             | 3854880         |
|      3013 | Kaari Hostel         | 94             | 2811200         |
|      3016 | Kalpana Chawla Hostel | 75            | 3741526         |
+-----------+----------------------+----------------+-----------------+
3 rows in set (0.00 sec)
```

Query:

```
mysql> select hostel_id,hostel_name,no_of_students,annual_expenses from hostel
where no_of_students >(select no_of_students from hostel where hostel_id='3011');
```

Output:

```
+-----------+----------------------+----------------+-----------------+
| hostel_id | hostel_name          | no_of_students | annual_expenses |
+-----------+----------------------+----------------+-----------------+
|      3012 | NRI Premium Hostel   | 82             | 3854880         |
```

```
|      3013 | Kaari Hostel          | 94             | 2811200         |
|      3016 | Kalpana Chawla Hostel | 75             | 3741526         |
+-----------+-----------------------+----------------+-----------------+
3 rows in set (0.00 sec)
```

Query:

```
mysql> select hostel_id,hostel_name,no_of_students,no_of_rooms,annual_expenses from
hostel where no_of_students <=(select no_of_students from hostel where
hostel_id='3013');
```

Output:

```
+-----------+-----------------------+----------------+-------------+--------------
--+
| hostel_id | hostel_name           | no_of_students | no_of_rooms |
annual_expenses |
+-----------+-----------------------+----------------+-------------+--------------
--+
|      3011 | Green Pearl Hostel    | 68             | 40          | 4855580
 |
|      3012 | NRI Premium Hostel    | 82             | 50          | 3854880
 |
|      3013 | Kaari Hostel          | 94             | 50          | 2811200
 |
|      3016 | Kalpana Chawla Hostel | 75             | 45          | 3741526
 |
+-----------+-----------------------+----------------+-------------+--------------
--+
4 rows in set (0.00 sec)
```

Query:

```
mysql> select mess_id,hostel_name,lunch_time from mess where mess_charge <(select
mess_charge from mess where mess_id='10512');
```

Output:

```
+---------+--------------------+----------------------+
| mess_id | hostel_name        | lunch_time           |
+---------+--------------------+----------------------+
|   10515 | NRI Premium Mess   | 11:45 AM to 01:00 PM |
|   11521 | Kalpana Chawla Mess | 11:45 AM to 01:00 PM |
|   11525 | Kaari Mess         | 11:45 AM to 12:30 PM |
+---------+--------------------+----------------------+
3 rows in set (0.00 sec)
```

Query:

```
mysql> select reg_no,hostel_name, total_fees,fees_status from bill where total_fees
<= (select total_fees from bill where reg_no ='1001');
```

Output:

```
+--------+--------------------+------------+-------------+
| reg_no | hostel_name        | total_fees | fees_status |
+--------+--------------------+------------+-------------+
|   1001 | Green Pearl Hostel |     300000 | paid        |
|   1002 | NRI Premium Hostel |     280000 | paid        |
|   1003 | Green Pearl Hostel |     300000 | not paid    |
|   1004 | Green Pearl Hostel |     220000 | paid        |
|   1005 | NRI Premium Hostel |     280000 | not paid    |
|   1006 | Green Pearl Hostel |     300000 | paid        |
|   1007 | Kaari Hostel       |     240000 | paid        |
|   1008 | Kaari Hostel       |     240000 | paid        |
|   1009 | Green Pearl Hostel |     220000 | not paid    |
|   1010 | Green Pearl Hostel |     300000 | paid        |
+--------+--------------------+------------+-------------+
10 rows in set (0.00 sec)
```