Experiment - 1

Aim :- To draw an ER diagram.

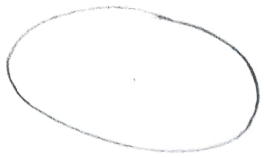
Theory :- DBMS or Database Management system is a software for storing and retrieving user's data while considering appropriate security measures It consists of a group of programs which manipulate the database.

SQL or Structured Query Language is a computer language for storing, manipulating and retrieving data stored in a relational database. It is the standard language for RDBMS or relational database management system.

ER - Diagram or Entity Relationship Diagram displays the relationships of entity set stored in a database. We can say that ER diagram can help you to explain the logical structure of databases.

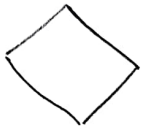
Entity

Weak entity

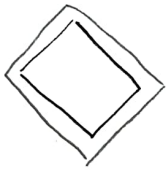


Attribute

Multivalued attribute



Relationship



Weak relationship

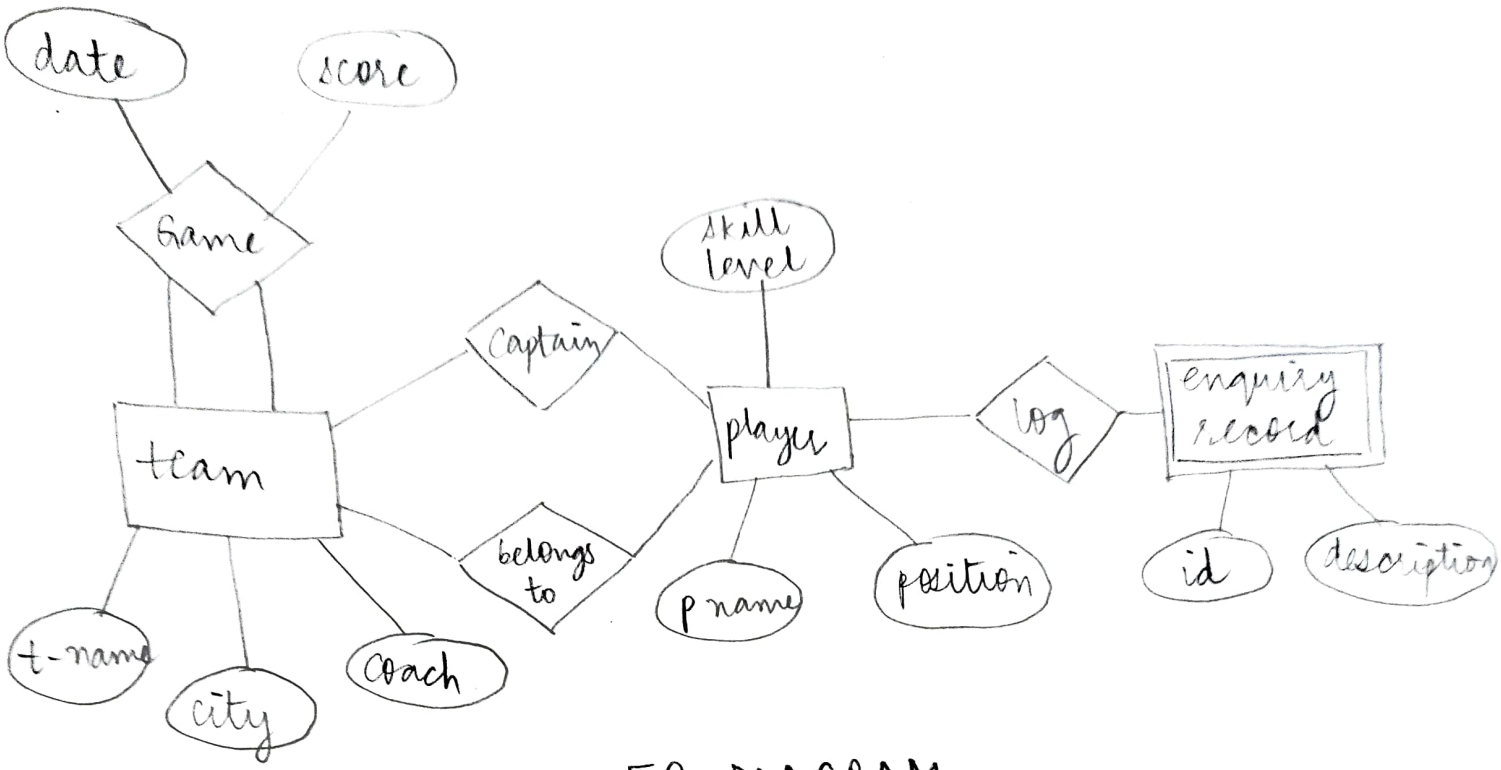
ER Diagram

Question 1 National Hockey League

1) NHL has many teams and each team has name, city, coach, captain and set of players.
2) Each player belongs to only one team and has skill level, position and set of records
3) Team captain is also a player
4) Game is played between two teams (host and guest) and have date and score.

Soln

National Hockey League



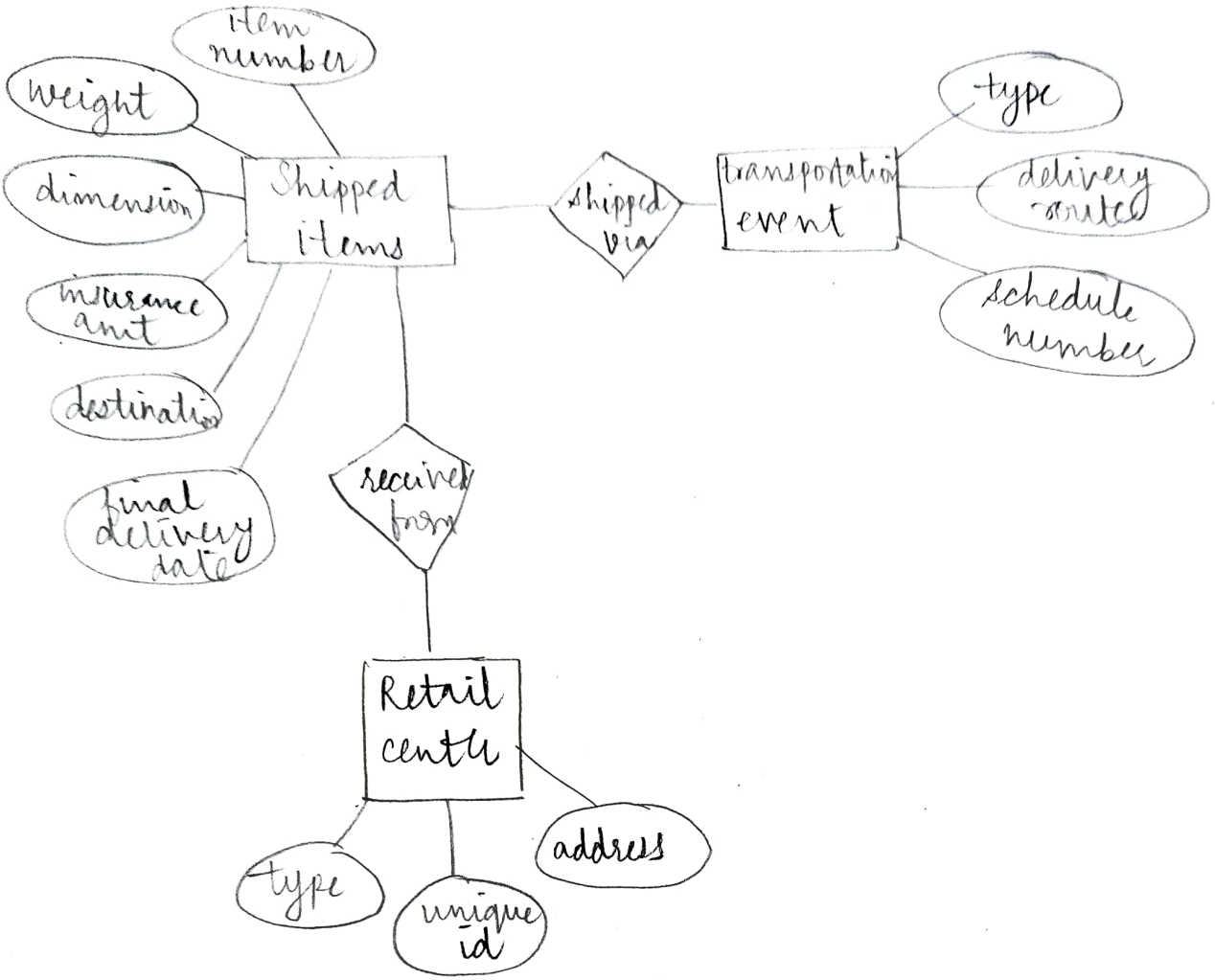
ER DIAGRAM

Ques 2 - ER Diagram (UPS Database)

Following is the information:

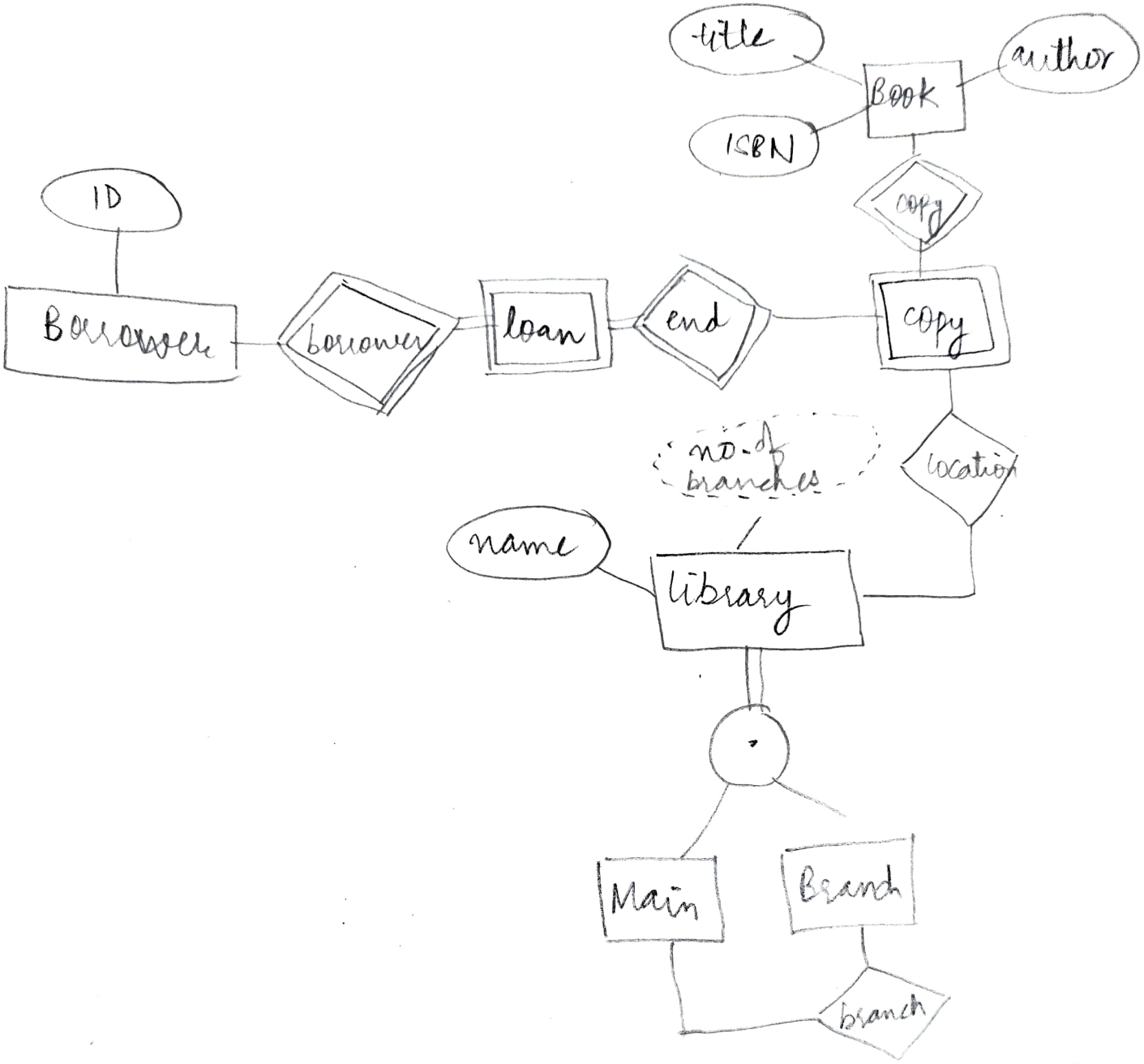
1) UPS has many teams and each team has name, city, coach, captain and set of players.
2) Each player belongs to only one team and shipped items that can be characterised by item no., weight, dimensions, insurance amount.
3) Shipped items are transported using transportation that have schedule number

ER Diagram 2 :- UPS Database



Ques3 - Library Database

1) Each book has unique ISBN number, a title and one or more authors.
2) Each box can have several copies of identified by copy number and has price assigned
3) Every library has unique name and can be either main or branch.
4) Main library has zero or more branch libraries and branch library is exactly one more library.



RESULT - ER Diagram for three database requirements are designed and studied successfully..

EXPERIMENT- 2

AIM - To study DLL commands in SQL

THEORY -

Structured Query Language (SQL) as well know is the database language by the use of which we can perform certain operations on the existing database and also we can perform certain operations on the existing database and also we can use this language to create a database and also we can use this language to create a database. SQL uses certain commands like create, drop, insert etc to carry out the required tasks.

These SQL commands are mainly categorized into four categories as:

1. DLL Data Definition Language
2. DML Data Manipulation Language
3. DQL Data Query Language
4. DCL Data Control Language

DDL (Data Definition Language)

DDL or Data Definition Language actually consists of the SQL commands that can be used to define the

database schema

egs of DDL commands :

CREATE - is used to create the database or objects

DROP - is used to delete objects from the database

ALTER - is used to alter the structure of the database

TRUNCATE - is used to remove all records from a table, including all spaces allocated for the records are removed.

COMMENT - is used to add comments to the data dictionary

RENAME - is used to rename an object existing in the database

TableName: Student

Name	Branch	Semester	ROLL_NO.	Grade
Heena	ECE	6	102	A
Kanika	CSE	7	111	A
Tanya	ECE	7	208	A
Anshika	IT	8	411	A
Kriti	ECE	5	201	A

Query to create table

Syntax :-

```
create table tablename ( column1 datatype ,
                         column2 datatype ,
                         column3 datatype ,
                         .... ) ;
```

Command :-

```
create table Student ( Name varchar (100) ,
                       Branch varchar (20) ,
                       Semester number ,
                       Roll_No number ,
                       Grade varchar (5) ) ;
```

Write SQL query to drop the given table

Syntax : Drop table table_name ;

Command : Drop table Student ;

Write SQL query to add column "Address" in the table given above.

Syntax : Alter table table_name ADD column_name datatype ;

Command : Alter table Student ADD Address varchar(100);

Write SQL query to delete the data in the table student

Syntax : Truncate table table_name

Command : Truncate table student;

Change the name of table "student" to "university - students"

Syntax : Alter table table_name RENAME TO new_table_name;

Command : Alter table student RENAME TO University - students;

RESULT -

The DDL commands in SQL have been studied and implemented.

```sql
create table test_table(id integer PRIMARY KEY, name char(100));
DESCRIBE test_table;
```

```
id      int      NO      PRI      NULL
name    char(100)        YES              NULL
```

```sql
insert into test_table(id, name) values(1, "Harshika"),(2, "Hardik");
select * from test_table;
```

```
1       Harshika
2       Hardik
```

```sql
alter table test_table add ph_no INT;
DESCRIBE test table;
```

```
id      int     NO      PRI     NULL
name    char(100)       YES             NULL
ph_no   int     YES             NULL
```

```sql
alter table test_table modify name char(20);
DESCRIBE test table;
```

```
id      int     NO      PRI     NULL
name    char(20)        YES             NULL
ph_no   int     YES             NULL
```

```sql
alter table test_table CHANGE COLUMN name stu_name CHAR(20);
DESCRIBE test table;
```

```
id      int     NO      PRI     NULL
stu_name        char(20)        YES             NULL
ph no   int     YES             NULL
```

```sql
alter table test_table drop column ph_no;
DESCRIBE test_table;
```

```
id      int     NO      PRI     NULL
stu_name        char(20)        YES             NULL
```

```sql
truncate table test_table;
select * from test_table;

drop table test_table
```

EXPERIMENT - 3

AIM - To study and implement DML commands and constraints in SQL.

THEORY - DML (Data Manipulation Language)

The SQL commands that deals with the manipulation of data present in the database belong to DML or this includes most of the SQL statements

eg of DML :-

INSERT - is used to insert data into a table

UPDATE - is used to update existing data within a table

DELETE - is used to delete records from a database table.

Table Name : Voter

F_Name	L_Name	City	DOB	Age	Voter id
Riya	Saxena	Delhi	15-12-1999	26	211
Siya	Rao	Mumbai	18-12-1999	21	803
Tiya	Gupta	Pune	31-10-1999	21	200
Jiya	Sharma	Noida	05-09-2000	20	121

⇒ SQL Query to insert the data of one more voter

Syntax :- Insert into table_name (Column1, Column2, ...) values (value 1, value 2, value 3, ...);

Command: Insert into voter (F_Name, L_Name, city, DOB, Age, Voter_id)
values ("Sita", "Sharma", "Noida", "05/08/1998", 22, 123);

⇒ Write SQL query to update L_Name = "Aggarwal" where Voter_id = 311

Syntax : Update table_name SET column1 = value1, column2 = value 2, ...
WHERE condition;

Command : Update Voter SET L_Name = "Aggarwal" where Voter_id = 311);

⇒ Write SQL query to delete the details of Voter "Ritu".

Syntax : Delete from table_name WHERE condition;

Command : Delete from Voter WHERE Voter_id=428;

Display the details of Voter who belong to Delhi

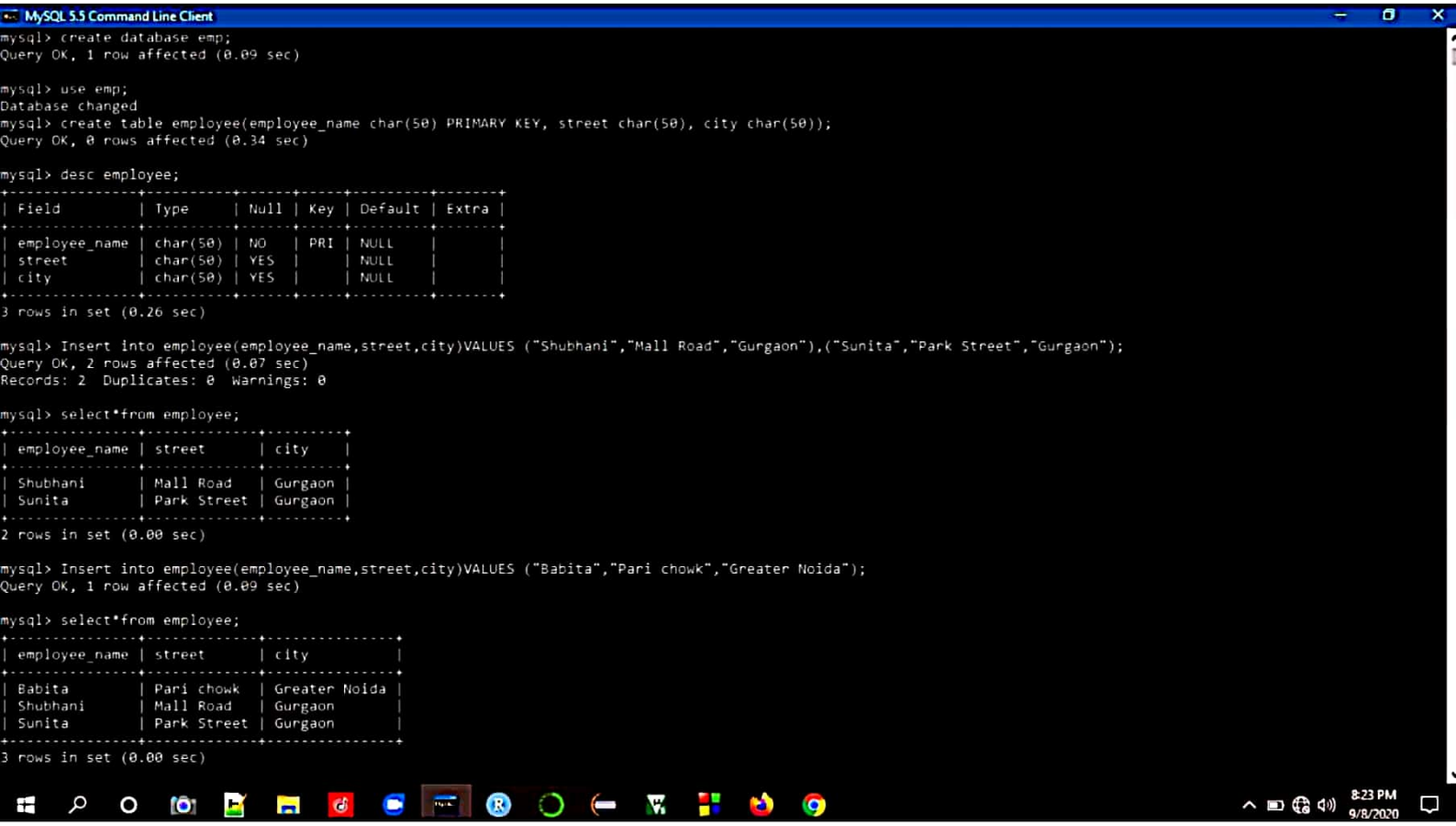
Syntax : Select column1 , column2 , ...
FROM table_name WHERE condition;

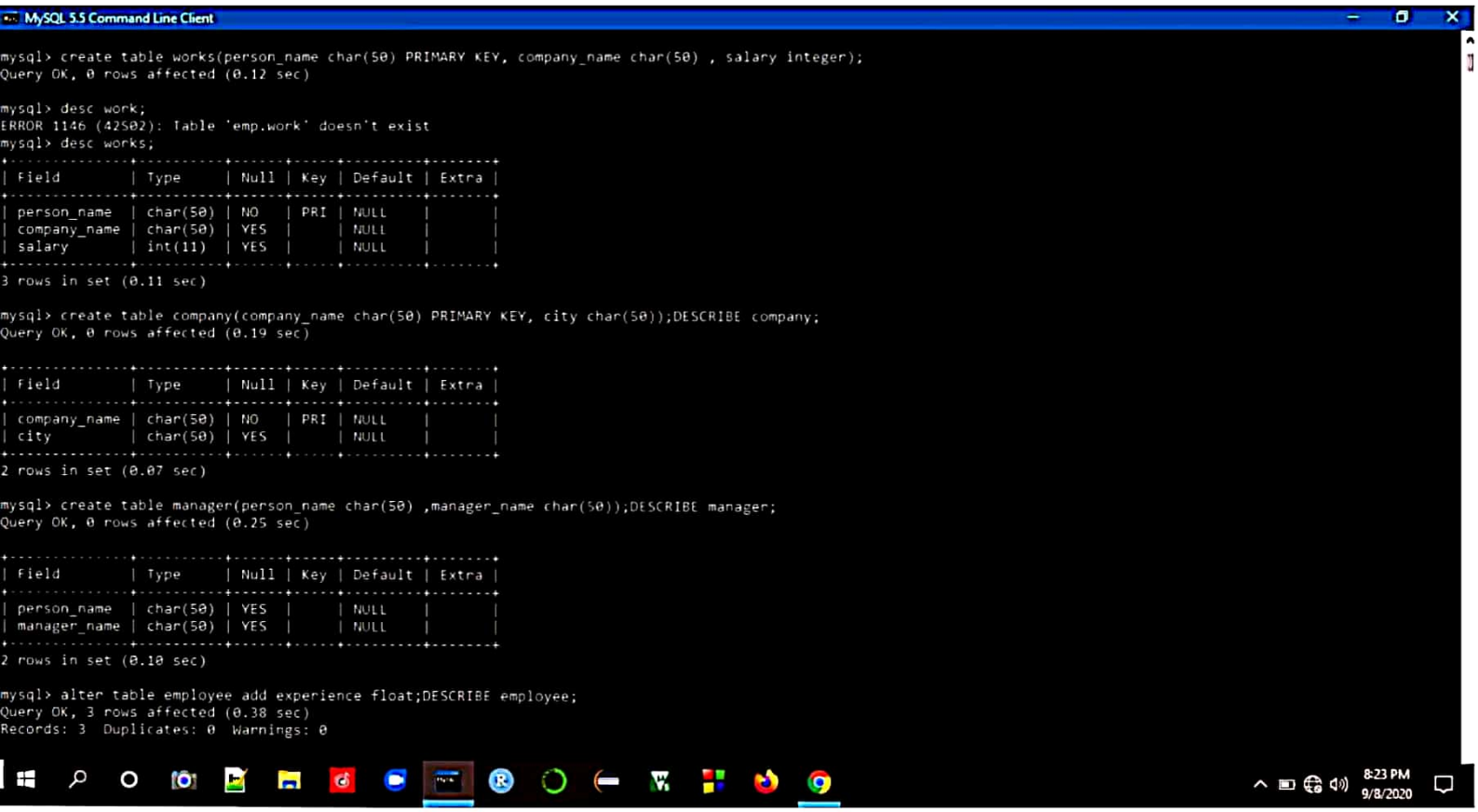
Command : Select * from Voter where city = "Delhi";

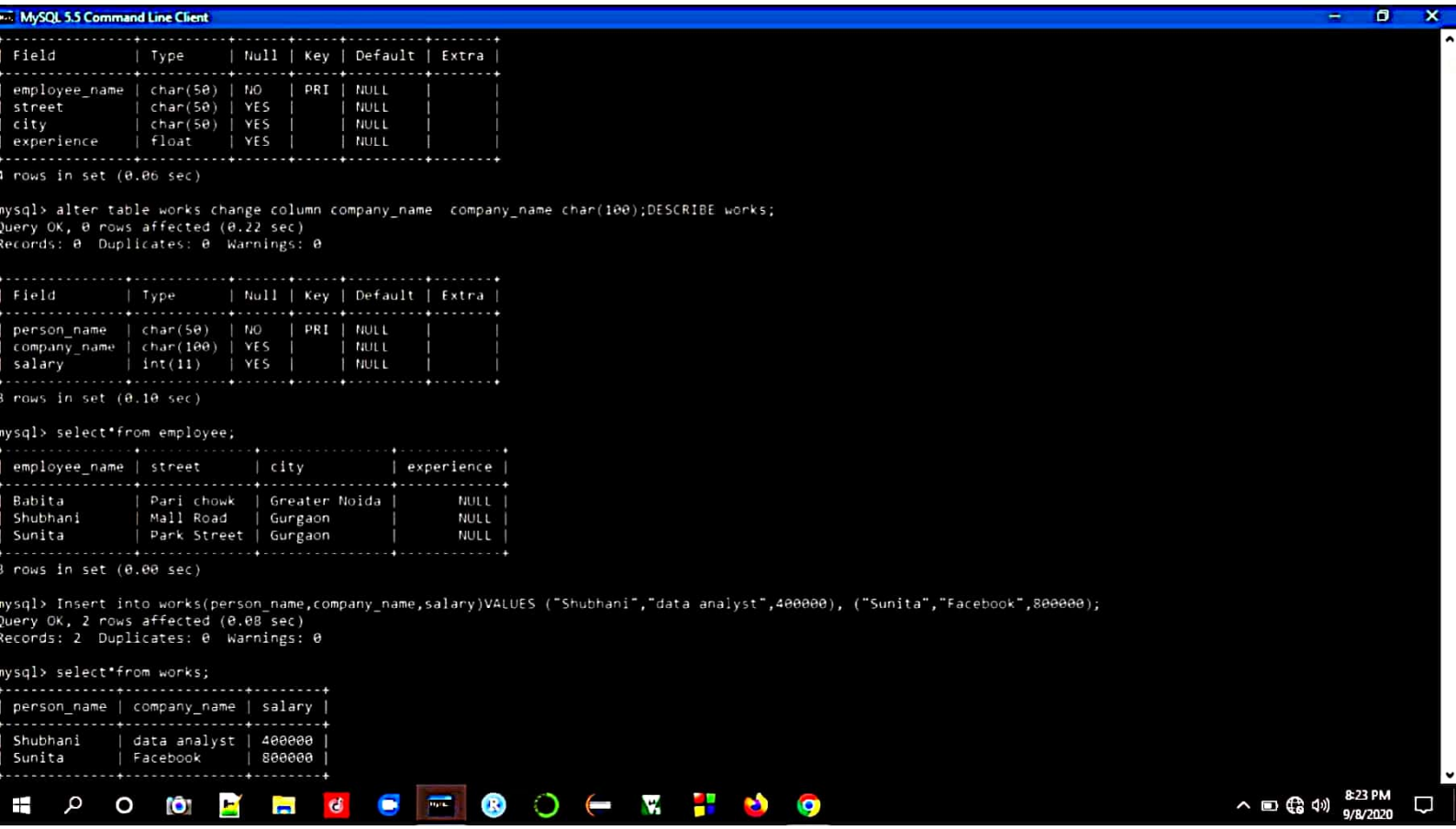
```
create table Voter ( F_Name varchar(20) NOTNULL;
                     L_Name varchar(20) NOT NULL;
                     City varchar(50) DEFAULT "Delhi";
                     DOB date,
                     Age number, CHECK (Age >= 18),
                     Voter_id number PRIMARY KEY);
```

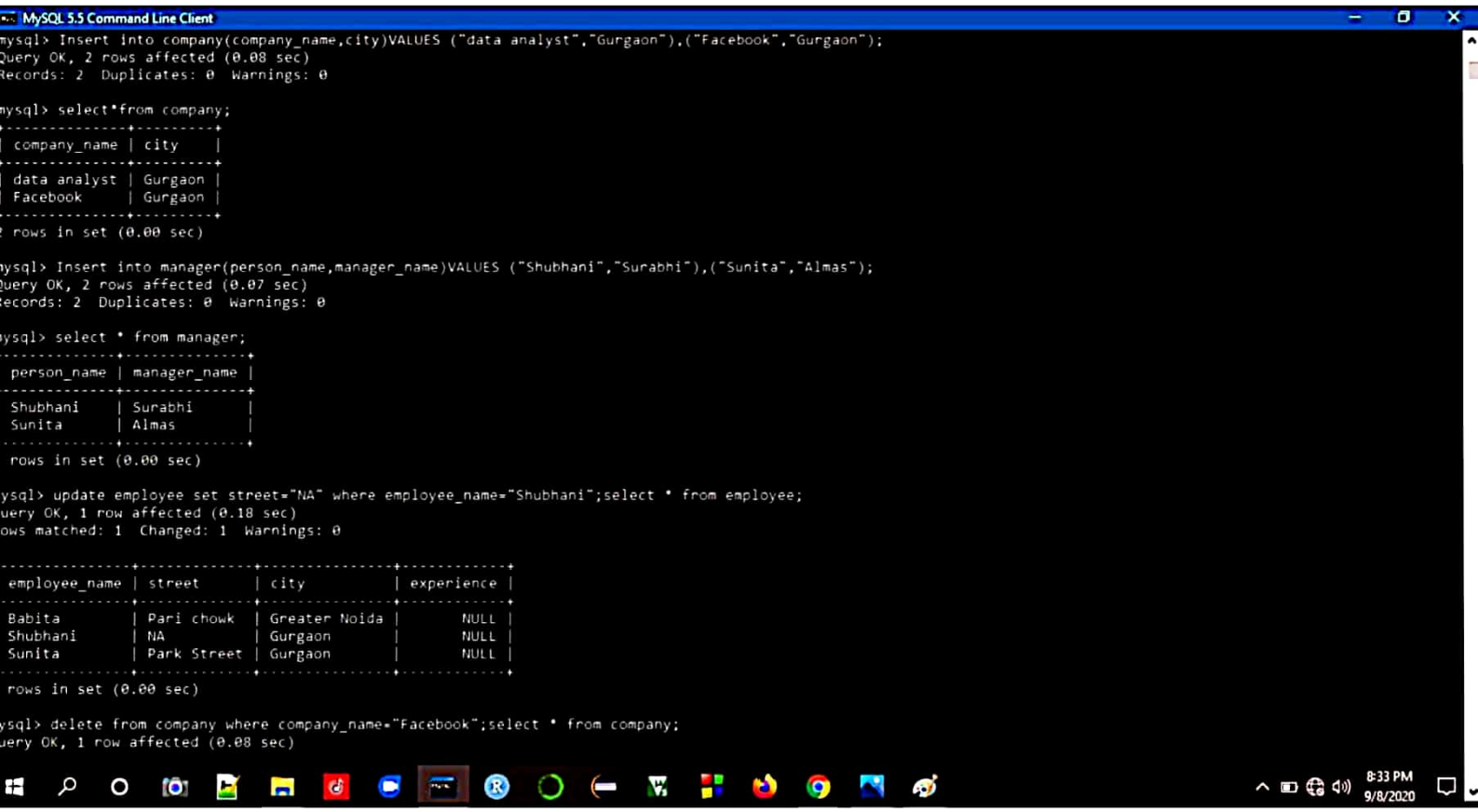
RESULT -

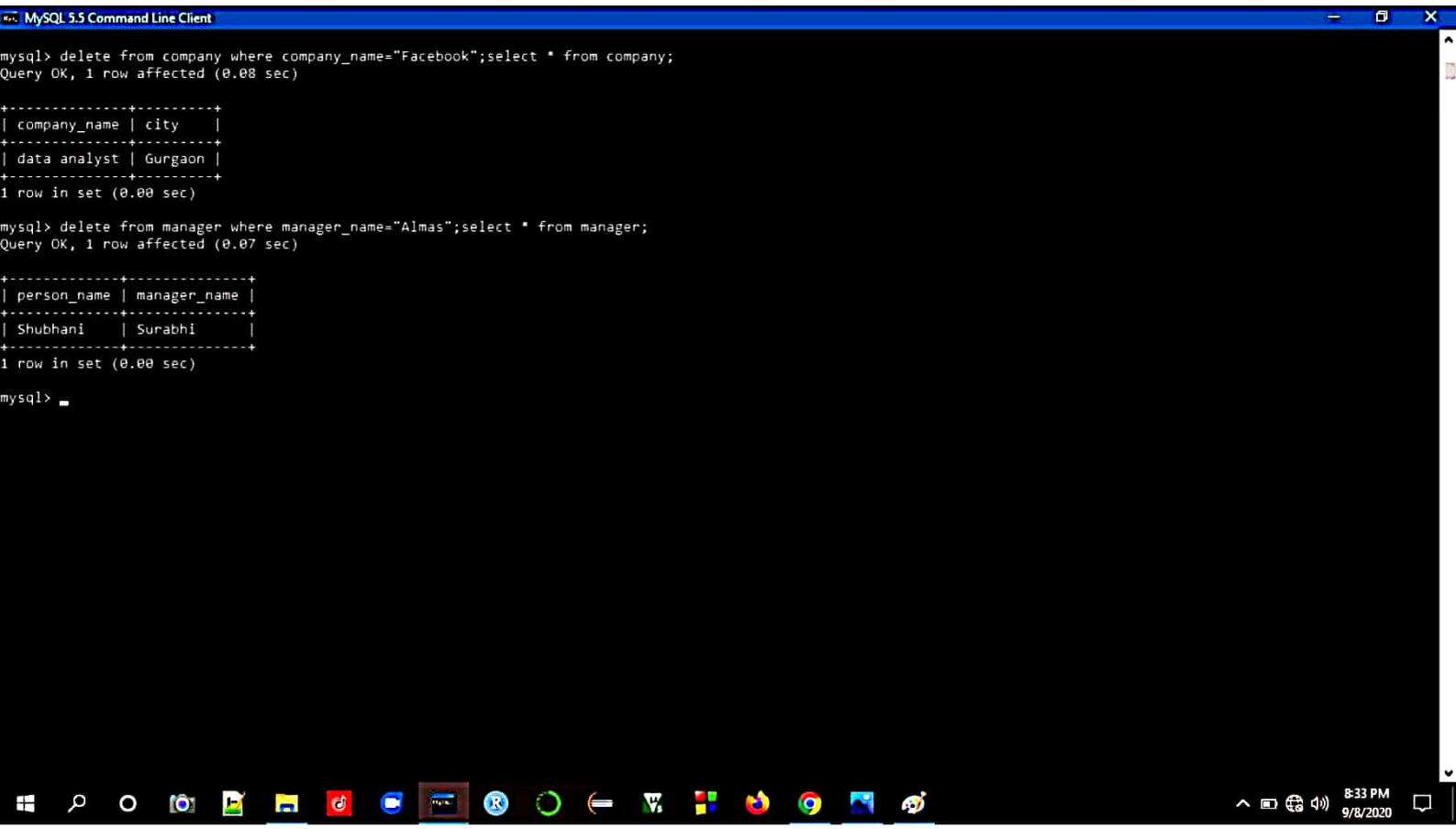
The DML commands have been studied and implemented successfully.











EXPERIMENT-4

AIM -

To implement the set operation and clauses like where order by, group by, having operations.

SOFTWARE USED -

MySQL 5.0

THEORY -

SQL support few set operation which can be performed on the table data. These are used to get meaningful results from data stored in the table under different special conditions.

4 SET operations :-

1) UNION
2) UNION ALL
3) INTERSECT
4) MINUS

UNION - used to combine the result of two or more 'select' statement.

INTERSECT

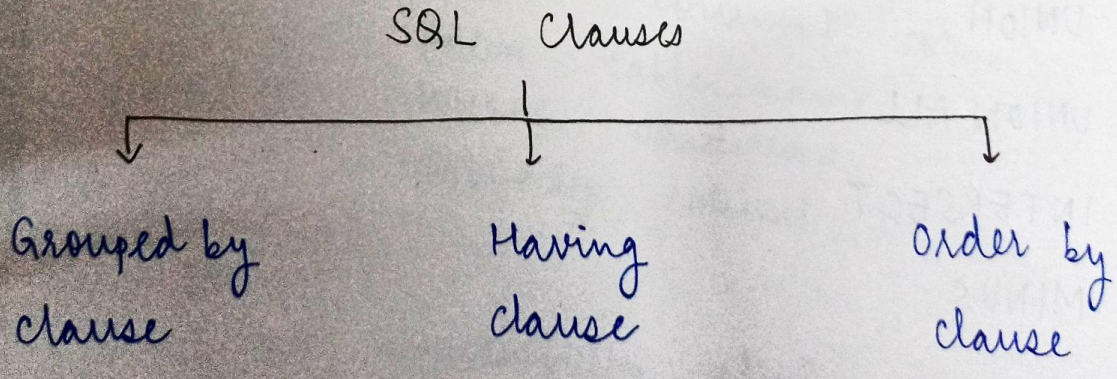
Intersect operation is used to combine two SELECT statement but it only return the records which are common form of both select statement.

SQL Query -

```
SELECT * FROM first
INTERSECT
SELECT * FROM second
```

SQL clauses -

Clauses are condition to fetch the data from the table using commands.



1) GROUP BY –

Syntax –

```
SELECT  column
FROM    <table_name>
WHERE   <conditions>
GROUP BY  column
ORDER BY  column;
```

2) HAVING –

Syntax –

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

3) ORDER BY

```
SELECT  column1, colum2
FROM    table_name
WHERE   conditions
ORDER By  colum1, colum2... ASC/DESC
```

RESULT –

Hence we successfully studies and implemented the set operation & clause using MySQL.

Execute | Share | main.sql | STDIN

```sql
BEGIN TRANSACTION;

/* Create a table called NAMES */
CREATE TABLE CUSTOMER(ID integer PRIMARY KEY, NAME char(50),AGE INT, ADDRESS
    varchar(50),  SALARY int,DeptID integer);

CREATE TABLE DEPARTMENT(DeptID integer, DeptNAME varchar(100));

/* Create few records in this table */
INSERT INTO CUSTOMER VALUES(1,'Ramesh', 32,'Ahmedabad',2000,1001);
INSERT INTO CUSTOMER VALUES(2,'Khilan', 25,'Delhi',1500,1002);
INSERT INTO CUSTOMER VALUES(3,'Kaushik', 23,'Kota',2000,1003);
INSERT INTO CUSTOMER VALUES(4,'Chaitali', 25,'Mumbai',6500,1001);
INSERT INTO CUSTOMER VALUES(5,'Hardik', 27,'Bhopal',8500,1001);
INSERT INTO CUSTOMER VALUES(6,'Komal', 22,'MP',4500,1002);
INSERT INTO CUSTOMER VALUES(7,'Muffy', 24,'Indore',10000,1003);

INSERT INTO DEPARTMENT VALUES(1001,'IT');
INSERT INTO DEPARTMENT VALUES(1002,'HR');
INSERT INTO DEPARTMENT VALUES(1003,'FINANCE');


COMMIT;

/* Display all the records from the table */

SELECT * FROM CUSTOMER;

SELECT NAME, SUM(SALARY) FROM CUSTOMER
   GROUP BY NAME;

```

Result

```
$sqlite3 database.sdb < main.sql
1|Ramesh|32|Ahmedabad|2000|1001
2|Khilan|25|Delhi|1500|1002
3|Kaushik|23|Kota|2000|1003
4|Chaitali|25|Mumbai|6500|1001
5|Hardik|27|Bhopal|8500|1001
6|Komal|22|MP|4500|1002
7|Muffy|24|Indore|10000|1003
Chaitali|6500
Hardik|8500
Kaushik|2000
Khilan|1500
Komal|4500
Muffy|10000
Ramesh|2000
```

Execute | Share | main.sql | STDIN

```sql

SELECT COUNT(ID), ADDRESS
FROM CUSTOMER
GROUP BY ADDRESS;

SELECT NAME, AVG(SALARY)
FROM CUSTOMER
GROUP BY NAME
HAVING AVG(SALARY)>2000;


SELECT DeptNAME,AVG(SALARY)
FROM CUSTOMER JOIN DEPARTMENT USING(deptID) GROUP BY DeptNAME;

SELECT DeptID FROM CUSTOMER
UNION
SELECT DeptID FROM DEPARTMENT
ORDER BY DeptID;
```

Result

```
$sqlite3 database.sdb < main.sql
1|Ahmedabad
1|Bhopal
1|Delhi
1|Indore
1|Kota
1|MP
1|Mumbai
Chaitali|6500.0
Hardik|8500.0
Komal|4500.0
Muffy|10000.0
FINANCE|6000.0
HR|3000.0
IT|5666.66666666667
1001
1002
1003
```

EXPERIMENT-5

AIM - Implement various inbuilt functions and views.

SOFTWARE USED -

Oracle Editor

THEORY -

Function are very powerful features of SQL used to manipulate data items.

Functions are similar to operators in that they manipulate data items and return as a result.

Types of function

Single Row function :-

used in Select where and ORDER by clauses.

func_name [(args 1, arg 2, ...)]

Multi Row function :-

Return one result for set of row

STRING / CHARACTER FUNCTION

1) Lower -

return char with all letter in lower case

syntax - lower (char)

2) INITCAP -

return a string with the first letter of each word in upper case.

syntax - initcap (char)

3) UPPER -

returns char, with all letters in upper case.

syntax - upper (char)

4) SUBSTR -

returns a portion of char beginning at char m and going up to character n. First position of char is 1

syntax - substr (<string>, <start-position> [<length>]

Numerical Function

1) ABS - return the absolute value of 'n'

syntax - ABC (-13)

2) Power -

return m raised to the n^{th} power n must be an integer else an error is returned

syntax - power (m, n)

3) Round -

returns n, rounded to m places to the right of decimal point.

4) Least -

returns least value in the list of expression

RESULT - Various inbuilt SQL functions have been successfully implemented.

EXPERIMENT 5

AIM: Implement various inbuilt functions and views.

Statement 1

```
create table customer(c_id integer primary key, c_name char(30), city char(20), address char(100), postal_code integer, country char(20))
```

Table created.

Statement 2

```
insert into customer values(1,'TOM','texas','baker_street',221,'usa')
```

1 row(s) inserted.

Statement 3

```
insert into customer values(2,'JERRY','california','oven_street',222,'usa')
```

1 row(s) inserted.

Statement 4

```
insert into customer values(3,'BOB','hyderabad','bake_street',212,'india')
```

1 row(s) inserted.

Statement 5

```
insert into customer values(4,'OGGY','delhi','paschim_vihar',272,'india')
```

1 row(s) inserted.

Statement 6

```
select * from customer
```

C_ID	C_NAME	CITY	ADDRESS	POSTAL_CODE	COUNTRY
1	TOM	texas	baker_street	221	usa
2	JERRY	california	oven_street	222	usa
3	BOB	hyderabad	bake_street	212	india
4	OGGY	delhi	paschim_vihar	272	india

Download CSV

4 rows selected.

Statement 7

```
select lower(c_name) "lower" from customer
```

lower
tom
jerry
bob
oggy

Download CSV

4 rows selected.

Statement 8

```
select initcap(c_name) "Title case" from customer
```

Title case
Tom
Jerry
Bob
Oggy

Download CSV

4 rows selected.

Statement 9

```
select upper(c_name) "capitalized" from customer
```

capitalized
TOM
JERRY
BOB
OGGY

Download CSV
4 rows selected.

Statement 10

```
select substr(address,3,5) "Substring" from customer
```

Substring
ker_s
en_st
ke_st
schim

Download CSV
4 rows selected.

Statement 11

```
select ascii(address) "ascii" from customer
```

ascii
98
111
98
112

Download CSV
4 rows selected.

Statement 12

```
select length(c_name) "length" from customer
```

length
30
30
30
30

Download CSV
4 rows selected.

Statement 13

```
select ltrim(address,'b') "ltrim" from customer
```

ltrim
aker_street
oven_street
ake_street
paschim_vihar

Download CSV
4 rows selected.

Statement 14

```
select lpad(c_name,35,'*') "lpad" from customer
```

lpad
*****TOM
*****JERRY
*****BOB
*****OGGY

Download CSV
4 rows selected.

Statement 15

```
select rpad(c_name,35,'*') "rpad" from customer
```

rpad
TOM *****
JERRY *****
BOB *****
OGGY *****

Download CSV

4 rows selected.

Statement 16

```
select vsize(c_name) "size" from customer
```

size
30
30
30
30

Download CSV

4 rows selected.

Statement 17

```
select instr(address,'a'),instr(address,'a',1,2) from customer
```

INSTR(ADDRESS,'A')	INSTR(ADDRESS,'A',1,2)
2	0
0	0
2	0
2	12

Download CSV

4 rows selected.

Statement 18

```
select power(postal_code,2) "power" from customer
```

power
48841
49284
44944
73984

Download CSV

4 rows selected.

Statement 19

```
select floor(sqrt(postal_code)) from customer
```

FLOOR(SQRT(POSTAL_CODE))
14
14
14
16

Download CSV

4 rows selected.

Statement 20

```
select ABS(postal_code) from customer
```

ABS(POSTAL_CODE)
221
222
212
272

Download CSV

4 rows selected.

Statement 21

```
select round(postal_code,1) from customer
```

ROUND(POSTAL_CODE,1)
221
222
212
272

Download CSV

4 rows selected.

Statement 22

```
select sqrt(postal_code) from customer
```

SQRT(POSTAL_CODE)
14.866068747318505522612008213931396651 44
14.899664425751339719331816046123951140 23
14.560219778561036542194604983054065587 55
16.492422502470642199285639423896338100 58

Download CSV

4 rows selected.

Statement 23

```
select exp(postal_code) from customer
```

EXP(POSTAL_CODE)
952972790236720253863556349863048922624800000000000000000000000000000000000000000000000000000000000000000000000000000
259044861871639010318301712871307125371000000000000000000000000000000000000000000000000000000000000000000000000000000
117606185343050012273356472412781022112000000000000000000000000000000000000000000000000000000000000000000000000000000
1.343071327497961308585925029761342178070000000000000000000000000000000000000000000000000000000000000000000E+118

Download CSV

4 rows selected.

Statement 24

```
select mod(postal_code,3) from customer
```

MOD(POSTAL_CODE,3)
2
0
2
2

Download CSV

4 rows selected.

Statement 25

```
select trunc(postal_code,1) from customer
```

TRUNC(POSTAL_CODE,1)
221
222
212
272

Download CSV

4 rows selected.

Statement 26

```
select ceil(postal_code) from customer
```

CEIL(POSTAL_CODE)
221
222
212
272

Download CSV

4 rows selected.

Statement 27

```
select max(postal_code) from customer
```

MAX(POSTAL_CODE)
272

Download CSV

Statement 28

```
select min(postal_code) from customer
```

MIN(POSTAL_CODE)
212

Download CSV

Statement 29

```
select greatest(postal_code) from customer
```

GREATEST(POSTAL_CODE)
221
222
212
272

Download CSV

4 rows selected.

Statement 30

```
select least(postal_code) from customer
```

LEAST(POSTAL_CODE)
221
222
212
272

Download CSV

4 rows selected.

EXPERIMENT-6

AIM -

Implement the nested queries and various operations on joins.

THEORY -

A MySQL subquery is a query nested within another query such as SELECT, INSERT, UPDATE or DELETE.

MySQL subquery is called an inner query while the query that contains the subquery is called an another query.

A subquery can be used anywhere that expression is used and most be closed in parantheses.

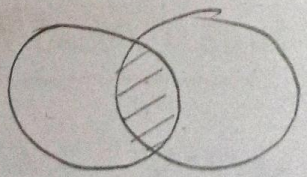
eg -

```
SELECT
    Last Name, First Name
FROM
    employees
```

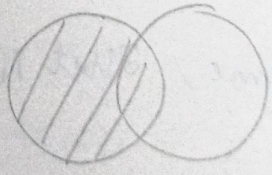
```
WHERE
    OFFICE code , IN( SELECT
                        Office code

    FROM
                office

    WHERE
                country = USA) ;
```

The another query selects the last name and first name of employees work in the offices.

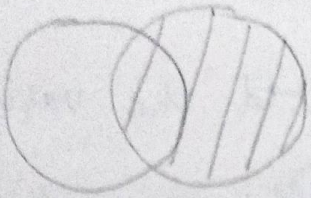
```
                Outer Query
SELECT lastName, firstName

    FROM employee                    Subquery / Inner query
WHERE office In (  SELECT Office code
                   FROM offices
                   WHERE country = "USA")
```



INNER JOIN



LEFT JOIN



RIGHT JOIN

FULL JOIN

SQL join -

Types of join :-

→ INNER JOIN - return records that have matching values in both tables.

→ LEFT JOIN -

return all records from the Left table and matched records from the right table.

→ RIGHT JOIN -

returns all records from the right table and matched records from the left table.

→ FULL (Outer) JOIN -

Returns all records when there is a matching in either left & right table.

RESULT –

We have successfully implemented the various joins & studied nested queries.

MySQL 5.5 Command Line Client

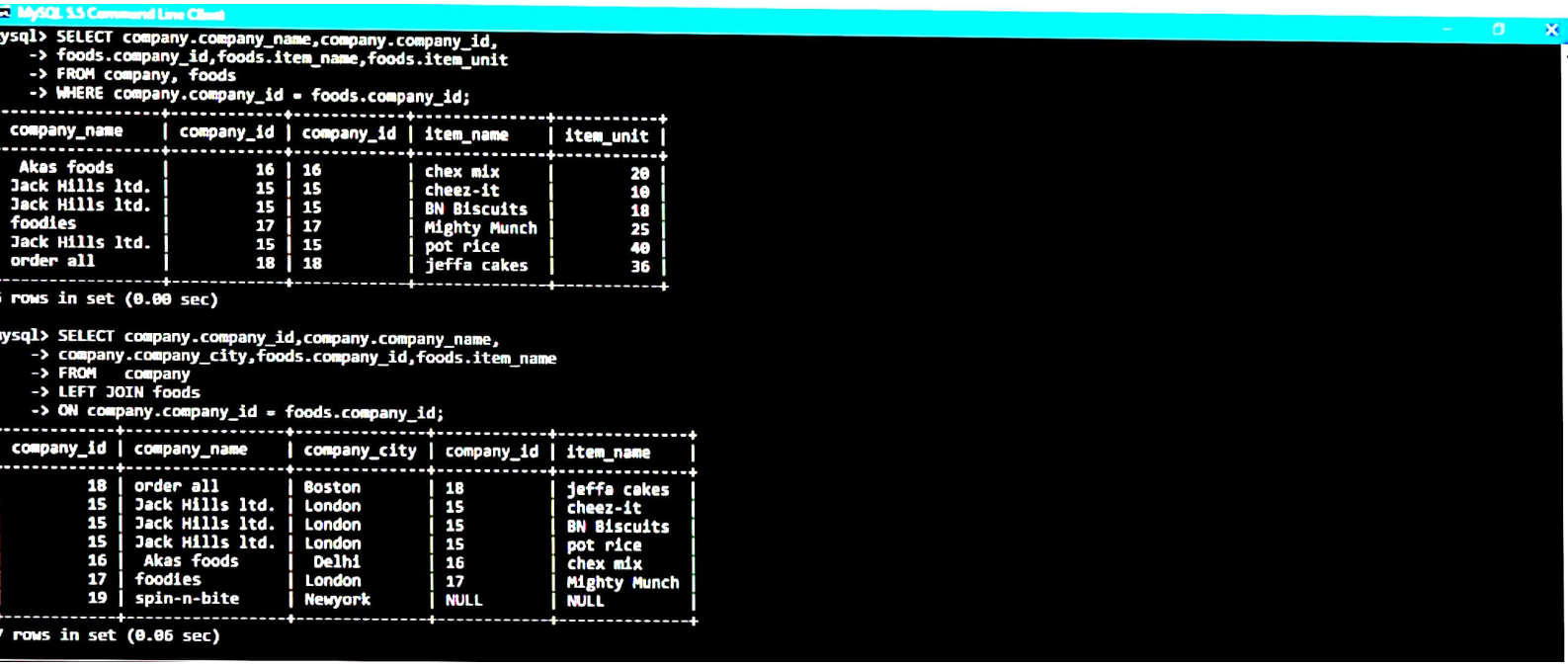
```
mysql> create database food;
ERROR 1007 (HY000): Can't create database 'food'; database exists
mysql>
mysql> use food;
Database changed
mysql> create table foods;
ERROR 1113 (42000): A table must have at least 1 column
mysql> create table foods(Item_id int,Item_name varchar(20),Item_unit int,company_id varchar(30));
Query OK, 0 rows affected (0.13 sec)

mysql>
mysql> show tables;
+----------------+
| Tables_in_food |
+----------------+
| foods          |
+----------------+
1 row in set (0.00 sec)

mysql>
mysql> insert into foods(Item_id,Item_name,Item_unit,company_id)values(1,"chex mix",20,16);
Query OK, 1 row affected (0.04 sec)

mysql>
mysql> insert into foods(Item_id,Item_name,Item_unit,company_id)values(2,"cheez-it",10,15);
Query OK, 1 row affected (0.03 sec)

mysql>
mysql> insert into foods(Item_id,Item_name,Item_unit,company_id)values(3,"BN Biscuits",18,15);
Query OK, 1 row affected (0.03 sec)

mysql>
mysql> insert into foods(Item_id,Item_name,Item_unit,company_id)values(4,"Mighty Munch",25,17);
Query OK, 1 row affected (0.03 sec)

mysql>
mysql> insert into foods(Item_id,Item_name,Item_unit,company_id)values(5,"pot rice",40,15);
Query OK, 1 row affected (0.03 sec)

mysql>
mysql> insert into foods(Item_id,Item_name,Item_unit,company_id)values(6,"jeffa cakes",36,18);
Query OK, 1 row affected (0.04 sec)
```

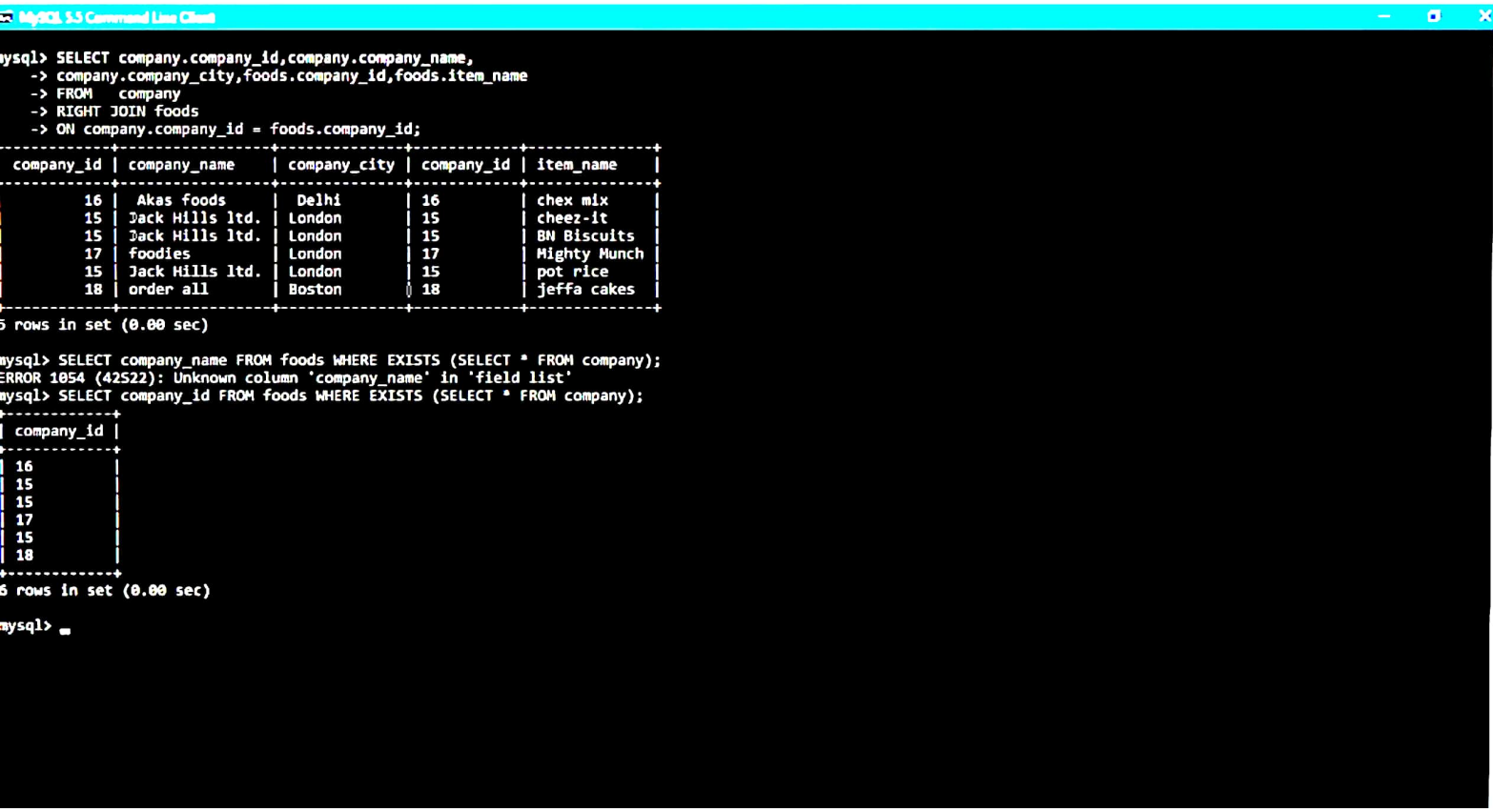

MySQL 5.5 Command Line Client
mysql> select * from foods;
+---------+--------------+-----------+------------+
| Item_id | Item_name | Item_unit | company_id |
+---------+--------------+-----------+------------+
1	chex mix	20	16
2	cheez-it	10	15
3	BN Biscuits	18	15
4	Mighty Munch	25	17
5	pot rice	40	15
6	jeffa cakes	36	18
+---------+--------------+-----------+------------+
6 rows in set (0.00 sec)
mysql> create table company(company_id int,company_name varchar(25),company_city varchar(25));
Query OK, 0 rows affected (0.15 sec)
mysql>
mysql> insert into company(company_id,company_name,company_city)values(18,"order all","Boston");
Query OK, 1 row affected (0.04 sec)
mysql>
mysql> insert into company(company_id,company_name,company_city)values(15,"Jack Hills ltd.","London");
Query OK, 1 row affected (0.03 sec)
mysql>
mysql> insert into company(company_id,company_name,company_city)values(16," Akas foods "," Delhi");
Query OK, 1 row affected (0.03 sec)
mysql>
mysql> insert into company(company_id,company_name,company_city)values(17,"foodies","London");
Query OK, 1 row affected (0.06 sec)
mysql>
mysql> insert into company(company_id,company_name,company_city)values(19,"spin-n-bite","Newyork");
Query OK, 1 row affected (0.03 sec)

```
mysql> select * from company;
+------------+----------------+--------------+
| company_id | company_name   | company_city |
+------------+----------------+--------------+
|         18 | order all      | Boston       |
|         15 | Jack Hills ltd. | London       |
|         16 |  Akas foods    |  Delhi       |
|         17 | foodies        | London       |
|         19 | spin-n-bite    | Newyork      |
+------------+----------------+--------------+
5 rows in set (0.00 sec)

mysql> SELECT foods.item_name,foods.item_unit,
    -> company.company_name,company.company_city
    -> FROM foods
    -> INNER JOIN company
    -> ON foods.company_id =company.company_id;
+-------------+-----------+----------------+--------------+
| item_name   | item_unit | company_name   | company_city |
+-------------+-----------+----------------+--------------+
| chex mix    |        20 |  Akas foods    |  Delhi       |
| cheez-it    |        10 | Jack Hills ltd. | London       |
| BN Biscuits |        18 | Jack Hills ltd. | London       |
| Mighty Munch |        25 | foodies        | London       |
| pot rice    |        40 | Jack Hills ltd. | London       |
| jeffa cakes |        36 | order all      | Boston       |
+-------------+-----------+----------------+--------------+
6 rows in set (0.11 sec)

mysql> SELECT *
    -> FROM foods
    -> JOIN company
    -> ON foods.company_id =company.company_id;
+---------+-------------+-----------+------------+------------+----------------+--------------+
| Item_id | Item_name   | Item_unit | company_id | company_id | company_name   | company_city |
+---------+-------------+-----------+------------+------------+----------------+--------------+
|       1 | chex mix    |        20 | 16         |         16 |  Akas foods    |  Delhi       |
|       2 | cheez-it    |        10 | 15         |         15 | Jack Hills ltd. | London       |
|       3 | BN Biscuits |        18 | 15         |         15 | Jack Hills ltd. | London       |
|       4 | Mighty Munch |        25 | 17         |         17 | foodies        | London       |
|       5 | pot rice    |        40 | 15         |         15 | Jack Hills ltd. | London       |
|       6 | jeffa cakes |        36 | 18         |         18 | order all      | Boston       |
+---------+-------------+-----------+------------+------------+----------------+--------------+
6 rows in set (0.00 sec)
```

MySQL 5.5 Command Line Client

```
mysql> SELECT foods.item_name,foods.item_unit,
    -> company.company_name,company.company_city
    -> FROM foods
    -> CROSS JOIN company;
+-------------+-----------+-----------------+--------------+
| item_name   | item_unit | company_name    | company_city |
+-------------+-----------+-----------------+--------------+
| chex mix    |        20 | order all       | Boston       |
| chex mix    |        20 | Jack Hills ltd. | London       |
| chex mix    |        20 |  Akas foods     |  Delhi       |
| chex mix    |        20 | foodies         | London       |
| chex mix    |        20 | spin-n-bite     | Newyork      |
| cheez-it    |        10 | order all       | Boston       |
| cheez-it    |        10 | Jack Hills ltd. | London       |
| cheez-it    |        10 |  Akas foods     |  Delhi       |
| cheez-it    |        10 | foodies         | London       |
| cheez-it    |        10 | spin-n-bite     | Newyork      |
| BN Biscuits |        18 | order all       | Boston       |
| BN Biscuits |        18 | Jack Hills ltd. | London       |
| BN Biscuits |        18 |  Akas foods     |  Delhi       |
| BN Biscuits |        18 | foodies         | London       |
| BN Biscuits |        18 | spin-n-bite     | Newyork      |
| Mighty Munch|        25 | order all       | Boston       |
| Mighty Munch|        25 | Jack Hills ltd. | London       |
| Mighty Munch|        25 |  Akas foods     |  Delhi       |
| Mighty Munch|        25 | foodies         | London       |
| Mighty Munch|        25 | spin-n-bite     | Newyork      |
| pot rice    |        40 | order all       | Boston       |
| pot rice    |        40 | Jack Hills ltd. | London       |
| pot rice    |        40 |  Akas foods     |  Delhi       |
| pot rice    |        40 | foodies         | London       |
| pot rice    |        40 | spin-n-bite     | Newyork      |
| jeffa cakes |        36 | order all       | Boston       |
| jeffa cakes |        36 | Jack Hills ltd. | London       |
| jeffa cakes |        36 |  Akas foods     |  Delhi       |
| jeffa cakes |        36 | foodies         | London       |
| jeffa cakes |        36 | spin-n-bite     | Newyork      |
+-------------+-----------+-----------------+--------------+
30 rows in set (0.00 sec)
```


MySQL 5.5 Command Line Client
ysql> SELECT company.company_name,company.company_id,
-> foods.company_id,foods.item_name,foods.item_unit
-> FROM company, foods
-> WHERE company.company_id = foods.company_id;
company_name | company_id | company_id | item_name | item_unit
Akas foods | 16 | 16 | chex mix | 20
Jack Hills ltd. | 15 | 15 | cheez-it | 10
Jack Hills ltd. | 15 | 15 | BN Biscuits | 18
foodies | 17 | 17 | Mighty Munch | 25
Jack Hills ltd. | 15 | 15 | pot rice | 40
order all | 18 | 18 | jeffa cakes | 36
rows in set (0.00 sec)
ysql> SELECT company.company_id,company.company_name,
-> company.company_city,foods.company_id,foods.item_name
-> FROM company
-> LEFT JOIN foods
-> ON company.company_id = foods.company_id;
company_id | company_name | company_city | company_id | item_name
18 | order all | Boston | 18 | jeffa cakes
15 | Jack Hills ltd. | London | 15 | cheez-it
15 | Jack Hills ltd. | London | 15 | BN Biscuits
15 | Jack Hills ltd. | London | 15 | pot rice
16 | Akas foods | Delhi | 16 | chex mix
17 | foodies | London | 17 | Mighty Munch
19 | spin-n-bite | Newyork | NULL | NULL
rows in set (0.06 sec)


MySQL 5.5 Command Line Client
mysql> SELECT company.company_id,company.company_name,
-> company.company_city,foods.company_id,foods.item_name
-> FROM company
-> RIGHT JOIN foods
-> ON company.company_id = foods.company_id;
company_id | company_name | company_city | company_id | item_name
16 | Akas foods | Delhi | 16 | chex mix
15 | Jack Hills ltd. | London | 15 | cheez-it
15 | Jack Hills ltd. | London | 15 | BN Biscuits
17 | foodies | London | 17 | Mighty Munch
15 | Jack Hills ltd. | London | 15 | pot rice
18 | order all | Boston | 18 | jeffa cakes
6 rows in set (0.00 sec)
mysql> SELECT company_name FROM foods WHERE EXISTS (SELECT * FROM company);
ERROR 1054 (42S22): Unknown column 'company_name' in 'field list'
mysql> SELECT company_id FROM foods WHERE EXISTS (SELECT * FROM company);
company_id
16
15
15
17
15
18
6 rows in set (0.00 sec)
mysql>

EXPERIMENT - 7

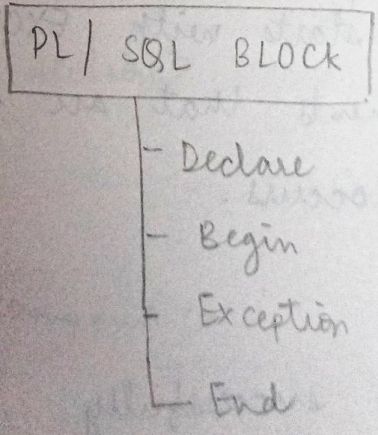
AIM -

Introduction to PL/SQL

SOFTWARE - PL/SQL is a block structured language that enables developers to combine the power of SQL with procedural statements.

Features -

1) It doesn't provide with condition checking, looping and branching.
2) Offers intensive error checking.
3) Mainly used to create an application.



⇒ A Block has the structures :-

DECLARE

declaration statements ;

BEGIN

executable statements ;

EXCEPTIONS

exception handling statements

→ Declare section starts with DECLARE keyword in which variables, constant, records as cursors can be declared which stores data temporarily.

→ Execution section starts with BEGIN and ends with END keyword.

→ Exception section starts with EXCEPTION keyword It contains statements that are executed when run-time error occurs.

RESULT -

PL/ SQL is studied successfully.

Code : -

```
-- DECLARING VARIABLES
DECLARE
xacct_no number(5);

-- here, minimum balance is set to 1000;
xmin_bal number(5):=1000;
xbalance number(5);

BEGIN

-- taking input from user
xacct_no:=&xacct_no;

-- selecting balance of that user INTO "xbalance";
select balance into xbalance
from acct_master
where acct_no=xacct_no;

-- if condition true, updating balance
-- with balance = balance - 100
IF(xbalance < xmin_bal) THEN --condition check
update acct_master
set balance=balance-100
where acct_no=xacct_no;

-- remaining
amount

xbalance:=xbalance-100;
dbms_output.put_line('Rs 100 is deducted
```

```
                    and current balance is '||xbalance);

-- if condition is false
ELSE
dbms_output.put_line('Current balance is '||xbalance);

--ENDING IF
END IF;

-- ENDING OF BEGIN
END;

/       -- FOR DISPLAYING OUTPUT IN SCREEN
```

Output : -

```
old 6: xacct_no:=&xacct_no;
new 6: xacct_no:=2;
Rs 100 is deducted and current balance is 0

PL/SQL procedure successfully completed.

SQL> /
Enter value for xacct_no: 3
old   6: xacct_no:=&xacct_no;
new   6: xacct_no:=3;
Current balance is 1100

PL/SQL procedure successfully completed.
```

EXPERIMENT - 8

AIM -

Write a program in PL/SQL and link the code with a table.

SOFTWARE USED -

Oracle

THEORY -

PL/SQL programming language was developed by oracle corporation in the late 1980s as procedural execution for SQL.

PL/SQL Basic Syntax

```
DECLARE
        < declaration section >

BEGIN
        < executable command >

EXECUTION
```

< exception handling >

END;

Examples -

```
DECLARE
message varchar 2(20) := "Hello World"
BEGIN
dbms_output.put_line (message)
END;
```

Output - Hello world

PL/SQL Literals

A literal is an explicit numeric, character, string or boolean value not represented by an identifier

It supports :-

- Numeric literals
- Character literals
- String literals
- Boolean literals
- Data & time literals.

RESULT -

Program is written in PL/SQL successfully.

INNOVATION - 1

AIM - Introduction to Cursor

SOFTWARE USED -

Oracle Editor

THEORY -

The 'Cursor' is the PL/SQL construct that allows the user to name the work area and access the stored information.

A cursor is a pointer to the context area PL/SQL controls the context area through a cursor

Cursor Action -

→ Declare
→ Open
→ Fetch

Syntax

CURSOR cursor_name is select_statement ;

Types of Cursor

Implicit Cursor -

It is created 'automatically' for the user by Oracle when a query is executed and is simpler to code.

Explicit Cursor-

A cursor can also be opened for processing data through a PL/SQL block on demand such as user defined cursor is explicit cursor

RESULT -

Cursors are studied successfully.

INNOVATION-2

AIM -

Implement implicit and explicit cursors

SOFTWARE - Oracle Editor

THEORY -

Attributes -

1) %. FOUND

Returns TRUE if an INSERT, UPDATE or DELETE statement affected one or more rows or a SELECT INTO statement.

2) %. NOT FOUND

returns TRUE, if an INSERT, UPDATE or DELETE statement affected no rows.

3) %. IS OPEN

Always return FALSE for implicit cursor

4) %. ROW COUNT

Returns the no. of rows affected by INSERT, UPDATE, DELETE statement.

Syntax -

```
DECLARE variables ;
records ;
create a cursor ;
BEGIN
open cursor ;
FETCH cursor ;
process the records ;
CLOSE cursor ;
END ;
```

RESULT -

Implicit and Explicit cursors are implemented & studied successfully

Code for Cursor--

```
DECLARE
  c_id customers.id%type;
  c_name customer.name%type;
  c_addr customers.address%type;
  CURSOR c_customers is
    SELECT id, name, address FROM customers;
BEGIN
  OPEN c_customers;
  LOOP
  FETCH c_customers into c_id, c_name, c_addr;
    EXIT WHEN c_customers%notfound;
    dbms_output.put_line(c_id || ' ' || c_name || ' ' || c_addr);
  END LOOP;
  CLOSE c_customers;
END;
/
```

result –

```
1 Ramesh Ahmedabad
2 Khilan Delhi
3 kaushik Kota
4 Chaitali Mumbai
5 Hardik Bhopal
6 Komal MP

PL/SQL procedure successfully
```