DBMS

LAB 2 & 3

Members : Rahul Barodia (B20CS047)
Vageesh (B20AI049)
Chakshu Anup Dhannawat (B20AI006)

Question 1)

SELECT n1,n2,n3
Where n1, n2, and n3 can be replaced by numbers
Example SELECT 2,4,5
Output It will be print of 3 numbers from 3 columns

Question 2)

SELECT “enter your arithmetic expression here”
Example- SELECT 60+1-5*2

Question 3)

```
#include <iostream>
#include <windows.h>
#include <mysql.h>

using namespace std;

MYSQL *conn;

int main(){
    conn = mysql_init(NULL);
```

```
    if (conn == NULL) {
        cout<<"Error: "<<mysql_error(conn)<<endl;
        exit(1);
    }

    // mysql_real_connect(Connection Instance, Username, Password,
Database, Port, Unix Socket, Client Flag)
    if (mysql_real_connect(conn, "localhost", "dbms_demo_vageesh_2",
"DBMSDemo_vageesh_2", "dbms_demo_vageesh_2", 3306, NULL, 0)){
        cout<<"Connected Successfully!"<<endl;
        char tableName[256] = "sales";
        char query[256],query1[256];
        snprintf(query, 256, "CREATE TABLE `%s` (`salesman_id` int NOT
NULL PRIMARY KEY,`name` varchar(255),`adrress_city` varchar(255),
`coverage_city` varchar(255),`commission` int);", tableName);

        int createTableStatus = mysql_query(conn, query);
        if (createTableStatus != 0) {
            cout<<"Error while creating table: "<<mysql_error(conn)<<endl;
        }

        snprintf(query1,256,"INSERT INTO `sales_table` (`salesman_id`,
`name`, `address_city`, `coverage_city`, `commission`) VALUES ('1',
'Chakshu', 'Mumbai', 'Jodhpur', '1000'), ('2', 'Vageesh', 'Haryana',
'Delhi', '2000'), ('3', 'Rahul', 'Amravati', 'Nagpur', '1500'), ('4',
'Rushil', 'Thane', 'Jodhpur', '1000'), ('5', 'Kaustubh', 'Goa', 'Delhi',
'1000'), ('6', 'Aman', 'Lucknow', 'Mumbai', '500');");
        int insertTableStatus = mysql_query(conn, query1);
        if (insertTableStatus != 0) {
            cout<<"Error while creating table: "<<mysql_error(conn)<<endl;
        }

    } else {
        cout<<"Error while connecting!"<<endl;
    }

    return 0;
```

salesman_id	name	address_city	coverage_city	commission
1	Chakshu	Mumbai	Jodhpur	1000
2	Vageesh	Haryana	Delhi	2000
3	Rahul	Amravati	Nagpur	1500
4	Rushil	Thane	Jodhpur	1000
5	Kaustubh	Goa	Delhi	1000
6	Aman	Lucknow	Mumbai	500

A)

part(a)

```
SELECT  *
FROM sales_table
—------------------------------------------------
```

part(b)

```
SELECT *
FROM sales_table
WHERE name=’abc’;
—------------------------------------------------------
```

part(c)

```
SELECT address_city,commission FROM sales_table WHERE address_city='Chennai' or
address_city='Mumbai';
—------------------------------------------------------
```

part(d)

```
SELECT name,salesman_id FROM sales_table WHERE sales_table.address_city =
sales_table.coverage_city;
—------------------------------------------------------
```

part(e)

```
SELECT name,salesman_id
FROM sales_table
WHERE sales_table..address_city != sales_table.coverage_city;
—------------------------------------------------------
```

part(f)

```
SELECT MAX(commission)
FROM sales_table
```

—----------

part(g)

```
SELECT coverage_city FROM sales_table WHERE commission = ( SELECT
MAX(commission) FROM sales_table);
```

—----------

part(h)

```
SELECT coverage_city , AVG(commission) AS avgcom FROM sales_table GROUP by
coverage_city;
```

—----------

part(i)

Group_by city, check count of distinct commission rates, if count is 1, return coverage_city name.

```
SELECT COUNT( DISTINCT commission), coverage_city FROM sales_table GROUP BY
coverage_city HAVING COUNT( DISTINCT commission) =1;
```

—----------

part(j)

Group_by city, check count of distinct commission rates, if count is 1, return coverage_city name.

```
SELECT COUNT( DISTINCT coverage_city),commission FROM sales_table GROUP BY
commission HAVING COUNT( DISTINCT coverage_city)=1;
```

—----------

part(k)

—----------

part(l)

```
#include <iostream>
#include <windows.h>
#include <mysql.h>
```

```
using namespace std;

MYSQL *conn;

int main(){
    conn = mysql_init(NULL);
    if (conn == NULL) {
        cout<<"Error: "<<mysql_error(conn)<<endl;
        exit(1);
    }

    // mysql_real_connect(Connection Instance, Username, Password,
Database, Port, Unix Socket, Client Flag)
    if (mysql_real_connect(conn, "localhost", "dbms_demo_vageesh_2",
"DBMSDemo_vageesh_2", "dbms_demo_vageesh_2", 3306, NULL, 0)){
        cout<<"Connected Successfully!"<<endl;
        char tableName[256] = "sales";
        char query[256],query1[256];
        snprintf(query, 256, "CREATE TABLE `%s` (`salesman_id` int NOT
NULL PRIMARY KEY,`name` varchar(255),`adrress_city` varchar(255),
`coverage_city` varchar(255),`commission` int,`date of employ` Date,`date
of release` Date);", tableName);

        int createTableStatus = mysql_query(conn, query);
        if (createTableStatus != 0) {
            cout<<"Error while creating table: "<<mysql_error(conn)<<endl;
        }

        snprintf(query1,256,"INSERT INTO `sales_table` (`salesman_id`,
`name`, `address_city`, `coverage_city`, `commission`,`date of
employ`,`date of release`) VALUES ('1', 'Chakshu', 'Mumbai', 'Jodhpur',
'1000','2021-05-12','2022-02-22'), ('2', 'Vageesh', 'Haryana', 'Delhi',
'2000','2022-06-19',NULL), ('3', 'Rahul', 'Amravati', 'Nagpur',
'1500','2020-07-15',NULL), ('4', 'Rushil', 'Thane', 'Jodhpur',
'1000','2020-08-18',NULL), ('5', 'Kaustubh', 'Goa', 'Delhi',
'1000','2019-02-15','2022-05-15'), ('6', 'Aman', 'Lucknow', 'Mumbai',
'500','2018-01-23','2021-06-16');");
        int insertTableStatus = mysql_query(conn, query1);
        if (insertTableStatus != 0) {
            cout<<"Error while creating table: "<<mysql_error(conn)<<endl;
```

```
        }

    } else {
        cout<<"Error while connecting!"<<endl;
    }

    return 0;
```

—--

B)

Q3) B] Our table has columns.

Salesman_id, name, address_city, coverage_city, commission

Functional dependencies:

Salesman_id → name

Salesman_id → address_city, coverage_city, commission

Thus, candidate key = { Salesman_id }

Primary key = Salesman_id.

Since all non-prime attributes are completely dependent on primary key, and no transitive dependency exists, we can say that table is in BCNF

- We can convert it to 4NF and further normal forms, but that is not necessary, as the table already has 0% redundancy. Thus, normalizing the table further would not be necessary.

→ After adding columns, 'date_of_employment' and 'date_of_release' only functional dependencies which will be added are,

Salesman_id → date_of_employment, date_of_release.

This table is also in BCNF, thus normalizing it is not required.

C)

a) SELECT * FROM sales_table_2 WHERE date_of_release IS null;

b) SELECT * FROM sales_table_2 WHERE date_of_release IS NOT null

c) SELECT * FROM sales_table_2 WHERE date_of_release IS null
ORDER BY date_of_employement LIMIT 1

d) SELECT COUNT(*) FROM salesman_table_2 WHERE date_of_employment >= '2022-01-01'

e)

f)

Question 4)

```
//Command to Compile: g++ demo.cpp -o demo.exe -lmysql

#include <iostream>
#include <windows.h>
#include <mysql.h>
// #include <float.h>

using namespace std;

MYSQL *conn;
// MYSQL *conn;
//  MYSQL_RES res; /* holds the result set */
//     MYSQL_ROW row;


int main()
{
```

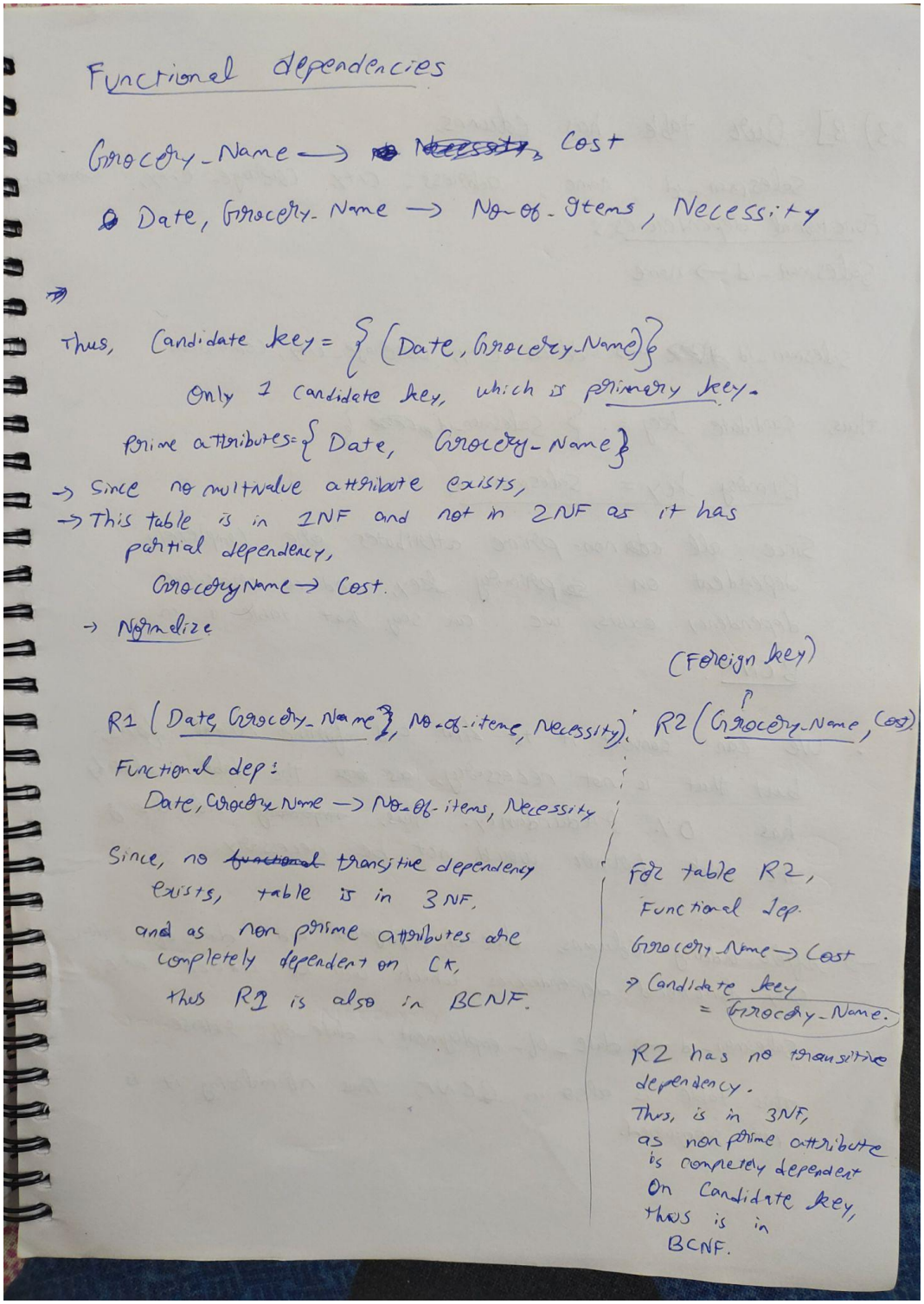
```
    conn = mysql_init(NULL);

    if (conn == NULL) {
        cout<<"Error: "<<mysql_error(conn)<<endl;
        exit(1);
    }

    // mysql_real_connect(Connection Instance, Username, Password,
    // Database, Port, Unix Socket, Client Flag)
    if (mysql_real_connect(conn, "localhost", "chakshu123",
"chakshu12345","lab2_q4", 3306, NULL, 0)) {

        cout<<"Connected Successfully!"<<endl;
        // char query[256];
        // snprintf(query,256,"DROP TABLE Report");
        //char tableName[256] = "lab2_q4";
        char query[256],query1[100000],query2[100000],query3[100000];

        //snprintf(query, 256, "CREATE TABLE `lab2_q4`.`lab2_q4` (`Date`
DATE NOT NULL , `Grocery_Name` VARCHAR(255) NOT NULL , `No_of_Items` INT
NOT NULL DEFAULT '1' , `Necessity` VARCHAR(255) NOT NULL DEFAULT 'Need' ,
`Cost` INT NOT NULL , PRIMARY KEY (`Date`, `Grocery_Name`));",tableName);

        //int createTableStatus = mysql_query(conn, query);


        //if (createTableStatus != 0) {
          //  cout<<"Error while creating table:
"<<mysql_error(conn)<<endl;
        //}
        snprintf(query1,100000,"INSERT INTO
lab2_q4(`Date`,`Grocery_Name`,`No_of_Items`,`Necessity`,`cost`) VALUES
('2022-07-02','Apple',10,'Need',300),('2022-07-02','Orange',0,'do
not',0),('2022-07-02','Cauliflower',3,'may',90),('2022-07-12','Tomato',4,'
Need',100),('2022-08-03','Apple',8,'eed',250),(2022-08-03,'Tomato',4,'need
',100),(2022-08-03,'Spinach',3,'may',90),(2022-08-03,'Egplant',2,'do
not',80),(2022-08-15,'Banana',6,'need',30),(2022-08-25,'Tomato',6,'need',1
20),(2022-09-04,'Apple',8,'need',300),(2022-09-04,'Tomato',4,'need',120),(
2022-09-04,'Spinach',3,'may',120),(2022-09-04,'Orange',2,'do
not',40),(2022-09-04,'Cauliflower',3,'need',70),(2022-09-11,'Onion',5,'nee
```

```
d',100),(2022-09-16,'Grapes',3,'may',120),(2022-09-17,'Apple',4,'do
not',150)");
        int insertTableStatus = mysql_query(conn, query1);
        //insertTableStatus = mysql_query(conn, query2);
        //insertTableStatus = mysql_query(conn, query3);
        if (insertTableStatus != 0) {
            cout<<"Error while inserting in
table:"<<mysql_error(conn)<<endl;

        }

    }
    else {
        cout<<"Error while connecting!"<<endl;
    }


    return 0;
}
```

a)

Date	Grocery_Name	No_of_Items	Necessity	Cost
2022-07-02	Apple	10	Need	300
2022-07-02	Cauliflower	3	may	90
2022-07-02	Orange	0	do not	0
2022-07-12	Tomato	4	Need	100
2022-08-03	Apple	8	Need	250
2022-08-03	Egplant	2	do not	80
2022-08-03	Spinach	3	may	90
2022-08-03	Tomato	4	Need	100
2022-08-15	Banana	6	Need	30
2022-08-25	Tomato	6	Need	120
2022-09-04	Apple	8	Need	300
2022-09-04	Cauliflower	3	Need	70
2022-09-04	Orange	2	do not	40
2022-09-04	Spinach	3	may	120
2022-09-04	Tomato	4	Need	120
2022-09-11	Onion	5	Need	100
2022-09-16	Grapes	3	may	120
2022-09-17	Apple	4	do not	150

b)



Functional dependencies

Grocery_Name → ~~Necessity~~, Cost

Date, Grocery_Name → No_of_Items, Necessity

Thus, Candidate key = {(Date, Grocery_Name)}

Only 1 candidate key, which is primary key.

Prime attributes = {Date, Grocery_Name}

→ Since no multivalue attribute exists,
→ This table is in 1NF and not in 2NF as it has partial dependency,
GroceryName → Cost.
→ Normalize

R1 (Date, Grocery_Name, No_of_items, Necessity)

Functional dep:
Date, Grocery Name → No_Of_items, Necessity

Since, no ~~functional~~ transitive dependency exists, table is in 3NF,
and as non prime attributes are completely dependent on CK,
thus R1 is also in BCNF.

(Foreign key)

R2 (Grocery_Name, Cost)

For table R2,
Functional dep.
Grocery_Name → Cost
→ Candidate key = Grocery_Name.

R2 has no transitive dependency.
Thus, is in 3NF,
as non prime attribute is completely dependent On Candidate key, thus is in BCNF.

After Normalizing in BCNF form we observe that 2 table are formed and they are :

Date	Grocery_Name	No_of_Items	Necessity	Cost
2022-07-02	Apple	10	Need	300
2022-07-02	Cauliflower	3	may	90
2022-07-02	Orange	0	do not	0
2022-07-12	Tomato	4	Need	100
2022-08-03	Apple	8	Need	250
2022-08-03	Egplant	2	do not	80
2022-08-03	Spinach	3	may	90
2022-08-03	Tomato	4	Need	100
2022-08-15	Banana	6	Need	30
2022-08-25	Tomato	6	Need	120
2022-09-04	Apple	8	Need	300
2022-09-04	Cauliflower	3	Need	70
2022-09-04	Orange	2	do not	40
2022-09-04	Spinach	3	may	120
2022-09-04	Tomato	4	Need	120
2022-09-11	Onion	5	Need	100
2022-09-16	Grapes	3	may	120
2022-09-17	Apple	4	do not	150

Date	Grocery_Name	Cost
2022-07-02	Apple	30
2022-07-02	Cauliflower	30
2022-07-02	Orange	20
2022-07-12	Tomato	25
2022-08-03	Apple	30
2022-08-03	Egplant	40
2022-08-03	Spinach	30
2022-08-03	Tomato	25
2022-08-15	Banana	5
2022-08-25	Tomato	25
2022-09-04	Apple	30
2022-09-04	Cauliflower	30
2022-09-04	Orange	20
2022-09-04	Spinach	30
2022-09-04	Tomato	25
2022-09-11	Onion	50
2022-09-16	Grapes	40
2022-09-17	Apple	30

c)

1) For the original table

```
select * from lab2_og order by Date desc limit 5;
```

Time taken is 0.0002 seconds.

For the Normalized table

```
select * from lab2_q4 NATURAL JOIN t2 ORDER by Date desc limit 5;
```

Time taken 0.0006 seconds

2) For the original table

SELECT *FROM lab2_og WHERE month(Date)=8;

Time taken is 0.0004 seconds

For the normalized table

SELECT *FROM lab2_q4 NATURAL JOIN t2 WHERE month(Date)=8;
Time taken is 0.0005 seconds

3) For the original table
   SELECT *FROM lab2_og WHERE month(Date)=9 and Necessity='Need'
   Time taken is 0.0003 seconds

   For the normalized table
   SELECT *FROM lab2_q4 NATURAL Join t2 WHERE month(Date)=9 and Necessity='Need'
   Time taken is 0.0005 seconds

4) For the original table
   SELECT *FROM lab2_og WHERE month(Date)=9 and Necessity='do not'
   Time taken is 0.0002 seconds
   For the normalized table
   SELECT *FROM lab2_q4 NATURAL JOIN t2 WHERE month(Date)=9 and Necessity='do not'
   Time taken is 0.0002 seconds

5) For the original table
   SELECT *FROM lab2_og WHERE month(Date)=9 and Necessity='may'
   Time taken is 0.0003 seconds

   For the Normalized table
   SELECT *FROM lab2_q4 NATURAL JOIN t2 WHERE month(Date)=9 and Necessity='may'

   Time taken is 0.0004 seconds

6) For the original table

d) By observing the time taken in the cases of normalized and original form we observe that time required in normalized form either increases or remains the same as compared to time taken in original form. It is happening because in BCNF form we need to search simultaneously in two tables.

e)