Lab Assignment 2 & 3

Team Number: **50**
Team Members: Soumik Roy (B20AI042)
Kulkarni Tanmay Shreevallabh (B20CS029)
Yash Bhargava (B20AI050)

Q.1

```
Command to Compile: g++ q1.cpp -o q1.exe -lmysql
```

Q.2

```
Command to Compile: g++ q2.cpp -o q2.exe -lmysql
```

Q.3

```
Command to Compile:
      g++ q3_create_table.cpp -o q3_create_table.exe -lmysql
      g++ q3_a.cpp -o q3_a.exe -lmysql
      g++ q3_b.cpp -o q3_b.exe -lmysql
      g++ q3_c.cpp -o q3_c.exe -lmysql
```

In this question, you first need to run the file *q3_create_table.exe* as it creates the tables which are required in the other subparts.

The table so created has the following columns :

←T→	salesman_id	name	address_city	coverage_city	commission

A)
After entering some dummy data into the table and running all queries given in Q3 part A, the table looks like this:

	salesman_id	name	address_city	coverage_city	commission	date_of_employment	date_of_release
Edit Copy Delete	1	raj	mumbai	delhi	750	NULL	NULL
Edit Copy Delete	2	tom	bhopal	pune	450	NULL	NULL
Edit Copy Delete	3	manya	delhi	kolkata	700	NULL	NULL
Edit Copy Delete	4	abc	mumbai	chennai	600	NULL	NULL
Edit Copy Delete	5	bhim	surat	bhopal	500	NULL	NULL
Edit Copy Delete	6	sara	delhi	mumbai	600	NULL	NULL
Edit Copy Delete	7	raj	chennai	bhopal	450	NULL	NULL
Edit Copy Delete	8	manu	chennai	kolkata	850	NULL	NULL
Edit Copy Delete	9	garry	mumbai	mumbai	450	NULL	NULL
Edit Copy Delete	10	binny	delhi	delhi	500	NULL	NULL
Edit Copy Delete	11	geeta	bhopal	pune	450	NULL	NULL
Edit Copy Delete	12	blue	kolkata	kolkata	500	NULL	NULL
Edit Copy Delete	13	abc	pune	chennai	500	NULL	NULL
Edit Copy Delete	14	xyz	pune	bhopal	500	NULL	NULL
Edit Copy Delete	15	feeli	delhi	mumbai	700	NULL	NULL
Edit Copy Delete	16	raj	mumbai	mumbai	450	NULL	NULL
Edit Copy Delete	17	raj	mumbai	kolkata	900	NULL	NULL
Edit Copy Delete	18	raj	pune	pune	450	NULL	NULL
Edit Copy Delete	19	raj	delhi	chennai	850	NULL	NULL

Console

Here date_of_employment and release columns have been added after data entry, so they currently contain null values. So the employee entries made before adding the columns have null values, but now whenever any entry for an employee is made, date_of_emplomeny and release can be specified.

B)
The functional dependencies formed in the table after Q3-A-(k) are:-

salesman_id → {‘name’, ‘address_city’, ‘coverage_city’, ‘commission’}
(‘address_city’, ‘coverage_city’) → {‘commission’}

As we can see that there is a transitive dependency present in the table, thus it cannot be in 3NF.

Also, as there are no partial dependencies present in the table as the primary key is non-composite, thus it can be in 2NF.

Therefore, the normal form of the table after subpart A(k) is 2NF.

We can improve upon it by changing it to 3NF by removing the transitive dependency.

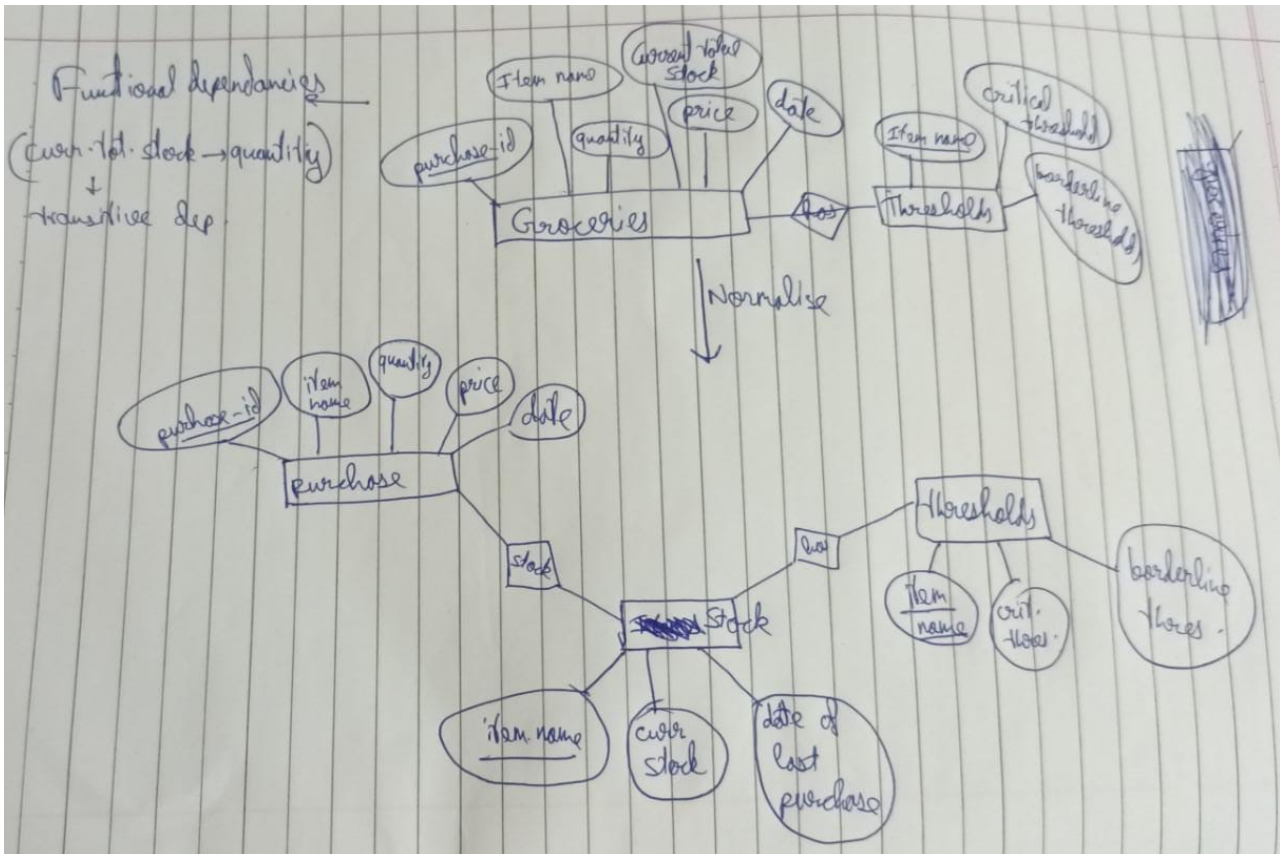
To do this, we have split our original table into 2 tables. First, we have a table called **commissions** consisting of *'salesman_id'*, *'address_city'*, *'coverage_city'*, and *'commission'* in which the primary key is ('salesman_id', 'address_city', 'comission_city'). Secondly, the original table **salesmen** now only contain *'salesman_id'*, *'name'*, *'date_of_employment'* and *'date_of_release'*, with the primary key 'salesman_id'.

New F.Ds are:-
Salesmen : salesman_id → {'name', 'date_of_employment', 'date_of_release'}
Commissions : ('salesman_id', 'address_city', 'coverage_city') → {'commission'}

Q.4

```
Command to Compile:
      g++ q4_a.cpp -o q4_a.exe -lmysql
      g++ q4_b.cpp -o q4_b.exe -lmysql
      g++ q4_c.cpp -o q4_c.exe -lmysql
```

ER Diagram:-



B)
Currently, the table has the following dependencies:-

‘purchase_id’ → {‘item_name’, ‘price’, ‘date_of_purchase’, ‘quantity’}
‘quantity’ → ‘current_total_stock’

As we can see that there is a transitive dependency present in the table, thus it cannot be in 3NF.

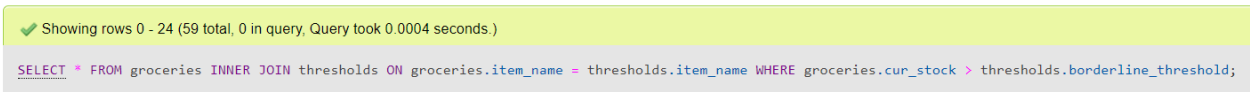
Also, as there are no partial dependencies present in the table as the primary key is non-composite, thus it can be in 2NF.

Thus, the table is not in BCNF.
Now, to normalize the table into BCNF, we need to decompose it into 2 tables namely, ‘groceries’ and ‘stock’. **groceries** consist of *‘purchase_id’*, *‘item_name’*, *‘quantity’*, *‘date_of_purchase’* and *‘price’*, with the primary key as ‘purchase_id’. **stock** consists of *‘item_name’*, *‘last_purchase_date’* and *‘current_total_stock’*, with the primary key as ‘item_name’.

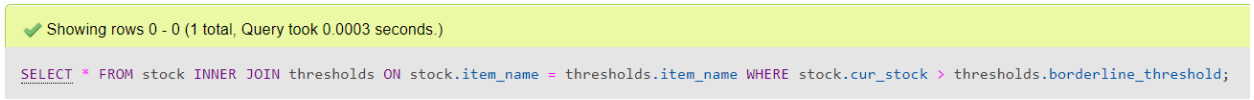
New F.Ds are:-
groceries : purchase_id → {‘item_name’, ‘date_of_purchase’, ‘quantity’, ‘price’}
stock : item_name → {‘last_purchase_date’, ‘current_total_stock’}

Since now all the functional dependencies have a primary key on the LHS, the table is in BCNF.

C)
Un-Normalized Tables -> Query of Q4-C-(v)



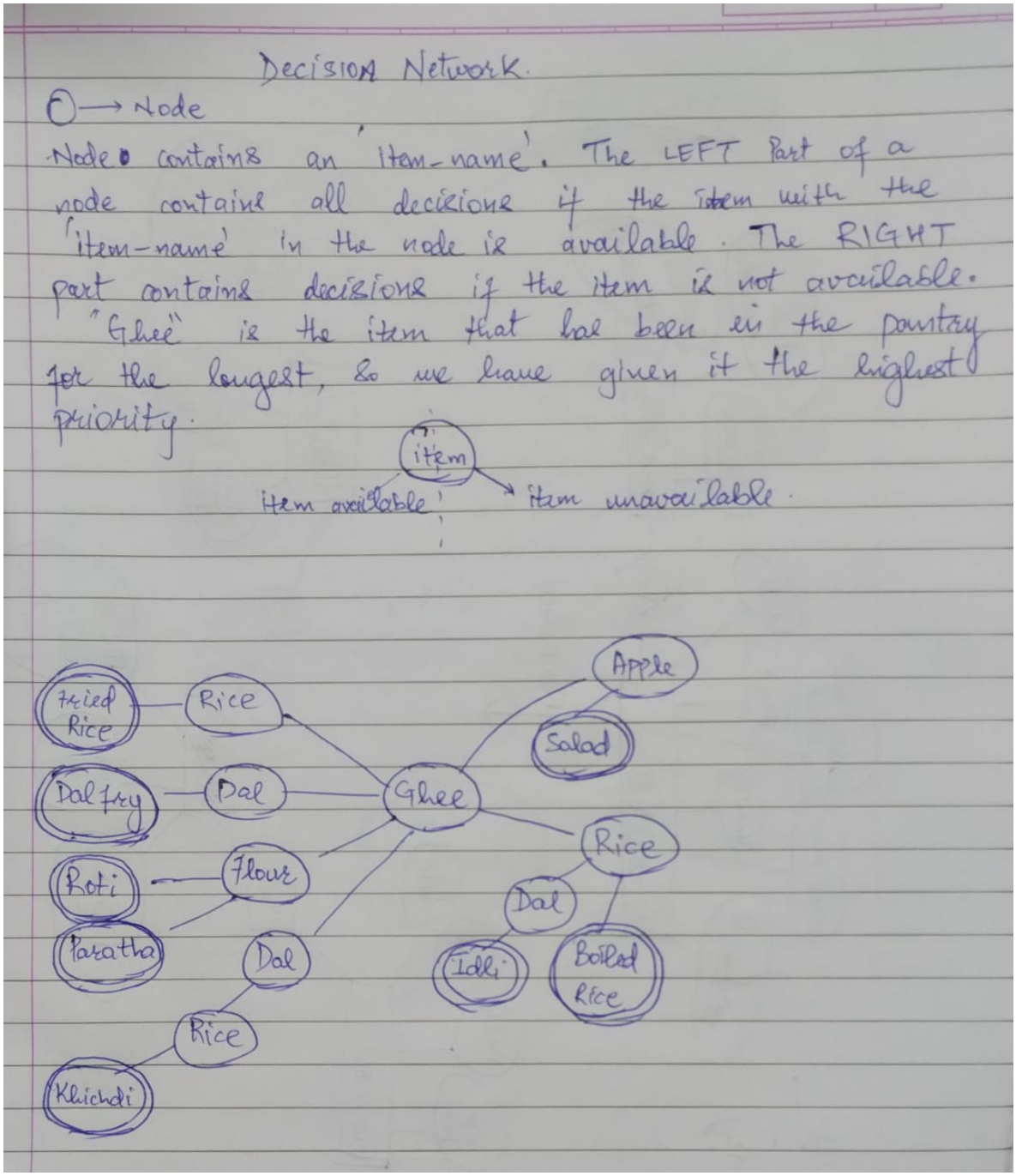
Normalized Tables -> Query of Q4-C-(v)



D)
As we can see that the execution time for the case on unnormalised data is longer than that in the normalised table. This is the case when the table has entries that are less than 100. Now when we deal with bigger databases, when the number of entries will be in millions, this difference will amplify, and will definitely be considerable in real-time

usage. Besides, Normalised data makes the schema of the database lucid and structured.

E)



Groceries table (un-normalized table)

	purchase_id	item_name	quantity_bought	cur_stock	date	price
Edit Copy Delete	1	flour	5000	5000	2022-07-01	200
Edit Copy Delete	2	oil	2000	2000	2022-07-01	200
Edit Copy Delete	3	ghee	200	200	2022-07-01	200
Edit Copy Delete	4	dal	5000	5000	2022-07-01	400
Edit Copy Delete	5	rice	5000	5000	2022-07-01	200
Edit Copy Delete	6	milk	2000	2000	2022-07-01	100
Edit Copy Delete	7	potato	1000	1000	2022-07-01	40
Edit Copy Delete	8	tomato	1000	1000	2022-07-01	60
Edit Copy Delete	9	onion	1000	1000	2022-07-01	30
Edit Copy Delete	10	watermelon	500	500	2022-07-01	50
Edit Copy Delete	11	apple	500	500	2022-07-01	80
Edit Copy Delete	12	chicken	1000	1000	2022-07-01	250
Edit Copy Delete	13	milk	1000	1000	2022-07-04	50
Edit Copy Delete	14	milk	1000	1000	2022-07-07	50
Edit Copy Delete	15	fish	1000	1000	2022-07-09	300
Edit Copy Delete	16	milk	1000	1000	2022-07-12	50
Edit Copy Delete	17	mutton	1000	1000	2022-07-15	500

Tables formed after normalization to BCNF:-

Purchase table

	purchase_id	item_name	quantity	date	price
Edit Copy Delete	1	flour	5000	2022-07-01	200
Edit Copy Delete	2	oil	2000	2022-07-01	200
Edit Copy Delete	3	ghee	200	2022-07-01	200
Edit Copy Delete	4	dal	5000	2022-07-01	400
Edit Copy Delete	5	rice	5000	2022-07-01	200
Edit Copy Delete	6	milk	2000	2022-07-01	100
Edit Copy Delete	7	potato	1000	2022-07-01	40
Edit Copy Delete	8	tomato	1000	2022-07-01	60
Edit Copy Delete	9	onion	1000	2022-07-01	30
Edit Copy Delete	10	watermelon	500	2022-07-01	50
Edit Copy Delete	11	apple	500	2022-07-01	80
Edit Copy Delete	12	chicken	1000	2022-07-01	250
Edit Copy Delete	13	milk	1000	2022-07-04	50
Edit Copy Delete	14	milk	1000	2022-07-07	50
Edit Copy Delete	15	fish	1000	2022-07-09	300
Edit Copy Delete	16	milk	1000	2022-07-12	50
Edit Copy Delete	17	mutton	1000	2022-07-15	500

Stock table

	item_name	cur_stock	last_purchase_date
Edit Copy Delete	apple	30	2022-09-01
Edit Copy Delete	chicken	0	2022-09-01
Edit Copy Delete	dal	100	2022-09-01
Edit Copy Delete	flour	500	2022-09-01
Edit Copy Delete	ghee	50	2022-09-01
Edit Copy Delete	milk	0	2022-09-01
Edit Copy Delete	mutton	0	2022-09-15
Edit Copy Delete	nutella	0	2022-09-09
Edit Copy Delete	oil	0	2022-09-01
Edit Copy Delete	onion	0	2022-09-01
Edit Copy Delete	paneer	0	2022-09-09
Edit Copy Delete	potato	0	2022-09-01
Edit Copy Delete	rice	250	2022-09-01

Thresholds table (For Q4 c (iii, iv and v))

Since different items will have different thresholds for when we need to/may/need not buy them.

	item_name	critical_threshold	borderline_threshold
Edit Copy Delete	apple	-1	200
Edit Copy Delete	chicken	-1	200
Edit Copy Delete	dal	250	500
Edit Copy Delete	fish	-1	0
Edit Copy Delete	flour	250	500
Edit Copy Delete	ghee	0	100
Edit Copy Delete	milk	0	200
Edit Copy Delete	mutton	-1	0
Edit Copy Delete	nutella	-1	-1
Edit Copy Delete	oil	250	500
Edit Copy Delete	onion	100	250
Edit Copy Delete	paneer	-1	0

Contributions

Soumik Roy:

ER Diagram, Code for Problem 1, Problem 2 and Parts of Problem 3.

Kulkarni Tanmay Shreevallabh:

Decision Network, Schema for problem 3, Code for problem 3, Problem 4

Yash Bhargava:

Code for Problem 4, Report writing, Decision Network.