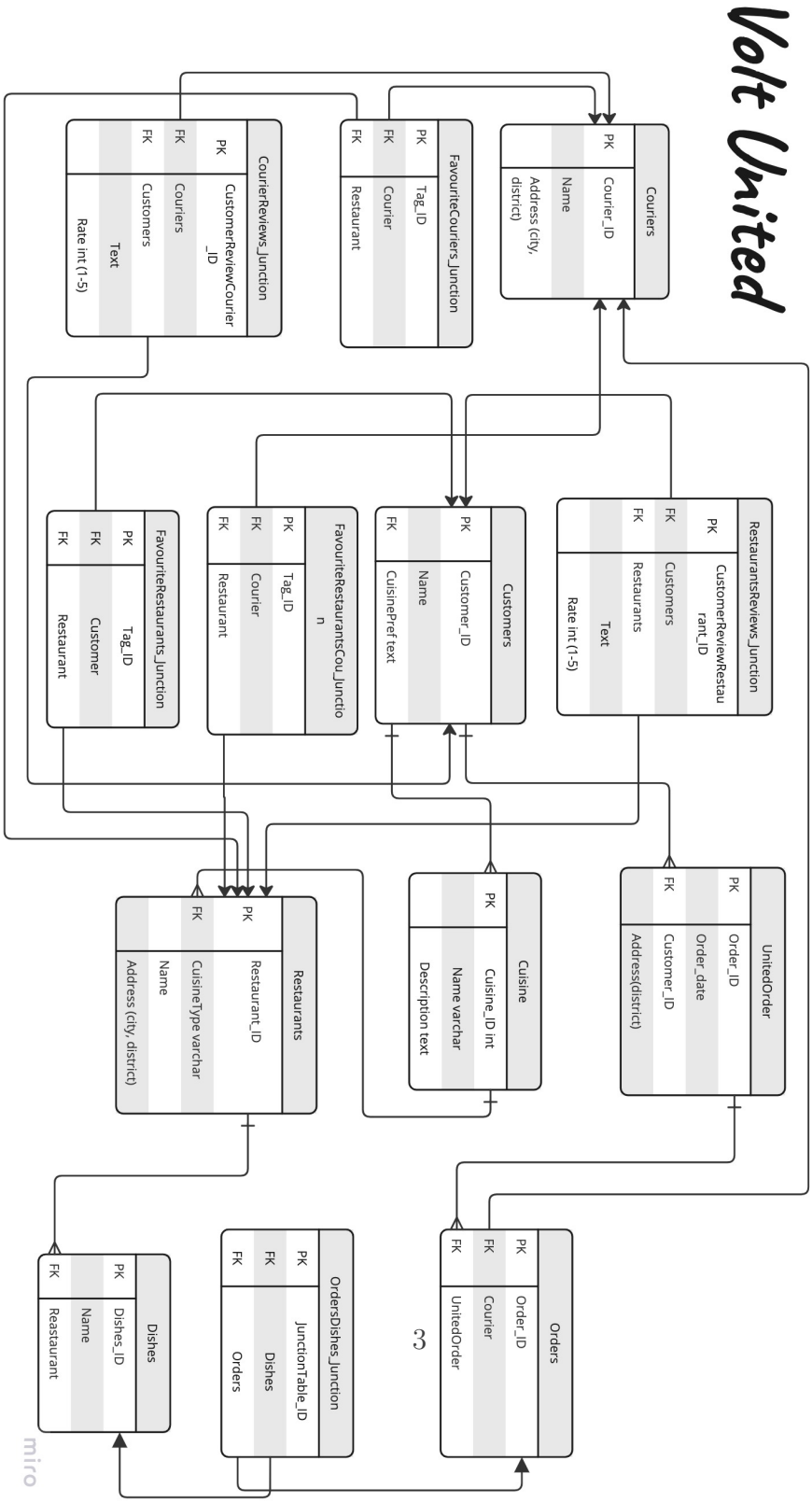
Assignment 1 — Volt United and Relational Calculus

Schmidt, Victor Alexander, rqc908
Ibsen, Helga Rykov, mcv462
Blixencrone-Møller, Laust Christian, sbh546

12. maj 2023

1 Volt United

a) E/R model for Volt United



b)

We've got three stakeholders — hence three entities: Restaurants, Customers and Couriers.

We've opted for binary relationships between them because they are three independent entities and we cannot use ISA to represent the relationship between them: i.e. none of the three inherit each other. Hence, no ISA-hierarchies have been used.

The three entities are related to each other using both uniqueness and referential integrity constraints. The example of the former:

```
1
2  CREATE TABLE Dishes (
3  DishesId SERIAL PRIMARY KEY,
4  Name VARCHAR(30),
5  RestaurantId INT REFERENCES Restaurants(Id)
6  );
7
```

By creating FOREIGN KEY (RestaurantId) that references the PRIMARY KEY of the table Restaurants, we enforce **the uniqueness relationship** between Dishes and Restaurants. We state that EACH dish has AT MOST one restaurant.

The referential integrity constraint ensures that data is not lost or corrupted when related records are updated or deleted. For example, we have a table of Customers and a table of UnitedOrders (see the diagram in 1.a)). The order table must have a foreign key that references the customer table's primary key. Referential integrity ensures that each united order is associated with a valid customer and that no orders exist for customers that do not exist in the customer table: each unitedOrder has EXACTLY one customer.

```
1
2  CREATE TABLE UnitedOrderJunction (
3  UnitedOrderId SERIAL PRIMARY KEY,
4  OrderDate DATE NOT NULL,
5  CustomerId INT REFERENCES Customers(Id) ON DELETE CASCADE
6  );
7
```

"ON DELETE CASCADE"option will delete all related rows in the UnitedOrderJunction table when a referenced row in the Customers table is deleted — hence, the referential integrity constrained will be enforced.

The issue of the base "location"of the three stakeholders is implemented through the attribute `Address`. The proximity is matched by the District attribute of the Address. The customer's address is included into the table of UnitedOrders. If districts of Restaurants, Couriers and UnitedOrders match, the order will be delivered to the customer, else — it cannot be delivered and is a take-out order. In our diagram, one united order can also be delivered by several couriers, assuming their Districts match those of Restaurants'.

As far as **weak entities** are concerned, the model involves multiple instances of those. Consider the Dishes table above. As already mentioned, it has FOREIGN KEY `RestautantId` that references the PRIMARY KEY of the table Restaurants. That means that dishes belong to restaurants and depend on them. And on adding "ON DELETE CASCADE"to the FOREIGN KEY of the table Dishes, any time we delete a restaurant from the table of Restaurants, the corresponding dishes in the Dishes table will also be deleted.

A note should be made on the **many-to-many** relation. The diagram involves six those "diamond"tables, which in our diagram are:

1. **OrdersDishes_Junction table**: establishes many-to-many relation between Dishes and Orders. An order may include many dishes and a dish may be part of many orders.

2. **FavouriteRestaurants_Junction table**: establishes many-to-many relation between Customers and Restaurants. A customer may tag many restaurants and a restaurant may be tagged by many customers.

3. **FavouriteRestaurantsCou_Junction table**: establishes many-to-many relation between Couriers and Restaurants. A courier may tag many restaurants and a restaurant may be tagged by many couriers.

4. **FavouriteCouriers_Junction table**: establishes many-to-many relation between Restaurants and Couriers. A restaurant may tag many couriers and a courier may be tagged by many restaurants.

5. **RestaurantsReviews_Junction table**: establishes many-to-many relation between Customers and Restaurants. One customer may write a review of many restaurants and one restaurant may be reviewed by many different customers.

6. **CourierReviews_Junction table**: establishes many-to-many relation between Customers and Couriers. A customer may write a review of many couriers and a courier may be reviewed by many customers.

Relational model

SQL tables

Entities

```
1
2  CREATE TABLE Couriers (
3  Id SERIAL,
4  Name VARCHAR(30),
5  Adress VARCHAR(50),
6  PRIMARY KEY (Id)
7  );
8
9  CREATE TABLE Cuisine (
10 Id SERIAL,
11 Name VARCHAR(30),
12 Description VARCHAR(50),
13 PRIMARY KEY (Id)
14 );
15
16 CREATE TABLE Customers (
17 Id SERIAL,
18 Cuisine_Id INT NOT NULL,
19 Name VARCHAR(30),
20 PRIMARY KEY (Id),
21 FOREIGN KEY (Cuisine_Id) REFERENCES Cuisine
22 );
23
24 CREATE TABLE Restaurants (
25 Id SERIAL,
26 Cuisine_Id INT NOT NULL,
27 Name VARCHAR(30),
28 Adress VARCHAR(50),
29 PRIMARY KEY (Id),
30 FOREIGN KEY (Cuisine_Id) REFERENCES Cuisine
31 );
32
33 CREATE TABLE Dishes (
34 Id SERIAL,
35 Restaurants_Id INT NOT NULL,
36 Name VARCHAR(30),
37 PRIMARY KEY (Id),
38 FOREIGN KEY (Restaurants_Id) REFERENCES Restaurants
39 );
40
41 CREATE TABLE Orders (
42 Id SERIAL,
```

```
43  Courier_Id INT NOT NULL,
44  UnitedOrder_Id INT NOT NULL,
45  PRIMARY KEY (Id),
46  FOREIGN KEY (Courier_Id) REFERENCES Courier,
47  FOREIGN KEY (UnitedOrder_Id) REFERENCES UnitedOrder ON DELETE
    CASCADE
48  );
49
50  CREATE TABLE UnitedOrder (
51  Id SERIAL,
52  Order_date DATE,
53  Customer_Id INT NOT NULL,
54  Adress VARCHAR(50),
55  PRIMARY KEY (Id),
56  FOREIGN KEY (Customer_Id) REFERENCES Customers ON DELETE
   CASCADE
57  );
58
```

Weak entities

```
1
2  CREATE TABLE FavouriteCouriers_Junction (
3  Id SERIAL,
4  Courier_Id INT NOT NULL,
5  Restaurant_Id INT NOT NULL,
6  PRIMARY KEY (Id),
7  FOREIGN KEY (Courier_Id) REFERENCES Courier ON DELETE CASCADE
   ,
8  FOREIGN KEY (Restaurant_Id) REFERENCES Restaurants ON DELETE
   CASCADE
9  );
10
11 CREATE TABLE CourierReviews_Junction (
12 Id SERIAL,
13 Courier_Id INT NOT NULL,
14 Customer_Id INT NOT NULL,
15 Description TEXT,
16 Rating INT,
17 PRIMARY KEY (Id),
18 FOREIGN KEY (Courier_Id) REFERENCES Courier ON DELETE CASCADE
   ,
19 FOREIGN KEY (Customer_Id) REFERENCES Customers ON DELETE
   CASCADE
20 );
21
22 CREATE TABLE RestaurantReviews_Junction (
23 Id SERIAL,
```

```
24  Restaurant_Id INT NOT NULL,
25  Customer_Id INT NOT NULL,
26  Description TEXT,
27  Rating INT,
28  PRIMARY KEY (Id),
29  FOREIGN KEY (Restaurant_Id) REFERENCES Restaurants ON DELETE
    CASCADE,
30  FOREIGN KEY (Customer_Id) REFERENCES Customers ON DELETE
    CASCADE
31  );
32
33  CREATE TABLE FavouriteRestaurantCourier_Junction (
34  Id SERIAL,
35  Courier_Id INT NOT NULL,
36  Restaurant_Id INT NOT NULL,
37  PRIMARY KEY (Id),
38  FOREIGN KEY (Courier_Id) REFERENCES Courier ON DELETE CASCADE
    ,
39  FOREIGN KEY (Restaurant_Id) REFERENCES Restaurants ON DELETE
    CASCADE
40  );
41
42  CREATE TABLE FavouriteRestaurantCustomer_Junction (
43  Id SERIAL,
44  Customer_Id INT NOT NULL,
45  Restaurant_Id INT NOT NULL,
46  PRIMARY KEY (Id),
47  FOREIGN KEY (Customer_Id) REFERENCES Customers ON DELETE
    CASCADE,
48  FOREIGN KEY (Restaurant_Id) REFERENCES Restaurants ON DELETE
    CASCADE
49  );
50
51  CREATE TABLE OrderDishes_Junction (
52  Id SERIAL,
53  Dish_Id INT NOT NULL,
54  Order_Id INT NOT NULL,
55  PRIMARY KEY (Id),
56  FOREIGN KEY (Dish_Id) REFERENCES Dishes ON DELETE CASCADE,
57  FOREIGN KEY (Order_Id) REFERENCES Orders ON DELETE CASCADE
58  );
59
```

Schema

```
Couriers(id:int, name:string, adress:string)
Customers(id:int, name:string, cuisine:string)
Cuisine(id:int, name:string, description:string)
Restaurants(id:int, name:string, adress:string, cuisineid:int)
Dishes(id:int, name:string, restaurantid:int)
Orders(id:int, courierid:int, unitedorderid:int)
UnitedOrder(id:int, orderdate:string, customerid:int, adress:string)
FavouriteCouriers_Junction(id:int, courierid:int, restaurantid:int)
CourierReviews_Junction(id:int, courierid:int, customerid:int,
    description:string, rating:int)
RestaurantReviews_Junction(id:int, customerid:int, restaurantid:int,
    description:string, rating:int)
FavouriteRestaurantCourier_Junction(id:int, courierid:int, restaurantid:int)
FavouriteRestaurantCustomer_Junction(id:int, customerid:int, restaurantid:int)
OrdersDishes_Junction(id:int, dish:int, order:int)
```

```
UnitedOrder
(1, 2023-01-05, 1, Noerrebro)
(2, 2023-02-10, 2, oesterbro)
(3, 2023-03-15, 3, Nordvest)
(4, 2023-04-20, 4, Indreby)
(5, 2023-05-25, 5, Amager)
(6, 2023-06-30, 6, Vesterbro)
(7, 2023-07-05, 7, Sydhavn)
(8, 2023-08-10, 8, Valby)
(9, 2023-09-15, 9, Vanloese)
(10, 2023-10-20, 10, Valby)
```

```
Couriers
 (1, John, Noerrebro)
 (2, Sarah, oesterbro)
 (3, David, Nordvest)
 (4, Emma, Indreby)
 (5, James, Amager)
 (6, Olivia, Vesterbro)
 (7, Benjamin, Sydhavn)
 (8, Ava, Valby)
 (9, Liam, Vanloese)
 (10, Mia, Valby)
```

```
 (11, Sophia, Noerrebro)
 (12, Lucas, oesterbro)
 (13, Isabella, Nordvest)
 (14, Noah, Indreby)
 (15, Amelia, Amager)
 (16, Ethan, Vesterbro)
 (17, Charlotte, Sydhavn)
 (18, Harper, Valby)
 (19, Elijah, Vanloese)
 (20, Henry, Valby)
 (21, Amelia, Noerrebro)
 (22, Alexander, oesterbro)
 (23, Ava, Nordvest)
 (24, Liam, Indreby)
 (25, Isabella, Amager)
 (26, Emma, Vesterbro)
 (27, Noah, Sydhavn)
 (28, Sophia, Valby)
 (29, Benjamin, Vanloese)
 (30, Mia, Valby)
 (31, William, Noerrebro)
 (32, Olivia, oesterbro)
 (33, James, Nordvest)
 (34, Charlotte, Indreby)
 (35, Ava, Amager)
 (36, Lucas, Vesterbro)
 (37, Amelia, Sydhavn)
 (38, Benjamin, Valby)
 (39, Emma, Vanloese)
 (40, Olivia, Valby)


Restaurants
(1, McDonalds, Noerrebro, 1)
 (2, Pizza_Hut, oesterbro, 2)
 (3, Thai_Garden, Nordvest, 3)
 (4, The_Italian_Place, Indreby, 4)
 (5, Sushi_Express, Amager, 5)
 (6, Burger_King, Vesterbro, 1)
 (7, Seafood_Paradise, Sydhavn, 2)
 (8, Indian_Spice, Valby, 3)
```

```
(9, Mexican_Delight, Vanloese, 4)
(10, Steakhouse_Grill, Valby, 5)
(11, McDonalds, Noerrebro, 1)
(12, Pizza_Hut, oesterbro, 2)
(13, Thai_Garden, Nordvest, 3)
(14, The_Italian_Place, Indreby, 4)
(15, Sushi_Express, Amager, 5)
(16, Burger_King, Vesterbro, 1)
(17, Seafood_Paradise, Sydhavn, 2)
(18, Indian_Spice, Valby, 3)
(19, Mexican_Delight, Vanloese, 4)
(20, Steakhouse_Grill, Valby, 5)
(21, McDonalds, Noerrebro, 1)
(22, Pizza_Hut, oesterbro, 2)
(23, Thai_Garden, Nordvest, 3)
(24, The_Italian_Place, Indreby, 4)
(25, Sushi_Express, Amager, 5)
(26, Burger_King, Vesterbro, 1)
(27, Seafood_Paradise, Sydhavn, 2)
(28, Indian_Spice, Valby, 3)
(29, Mexican_Delight, Vanloese, 4)
(30, Steakhouse_Grill, Valby, 5)
(31, McDonalds, Noerrebro, 1)
(32, Pizza_Hut, oesterbro, 2)
(33, Thai_Garden, Nordvest, 3)
(34, The_Italian_Place, Indreby, 4)
(35, Sushi_Express, Amager, 5)
(36, Burger_King, Vesterbro, 1)
(37, Seafood_Paradise, Sydhavn, 2)
(38, Indian_Spice, Valby, 3)
(39, Mexican_Delight, Vanloese, 4)
(40, Olivia, Valby, 5)


FavouriteRestaurantCustomer_Junction
(1, 1, 1)
(2, 2, 2)
(3, 3, 3)
(4, 4, 4)
(5, 5, 5)
(6, 6, 6)
```

```
 (7, 7, 7)
 (8, 8, 8)
 (9, 9, 9)
 (10, 10, 10)
 (11, 1, 11)
 (12, 2, 12)
 (13, 3, 13)
 (14, 4, 14)
 (15, 5, 15)
 (16, 6, 16)
 (17, 7, 17)
 (18, 8, 18)
 (19, 9, 19)
 (20, 10, 20)
 (21, 1, 21)
 (22, 2, 22)
 (23, 3, 23)
 (24, 4, 24)
 (25, 5, 25)
 (26, 6, 26)
 (27, 7, 27)
 (28, 8, 28)
 (29, 9, 29)
 (30, 10, 30)
 (31, 1, 31)
 (32, 2, 32)
 (33, 3, 33)
 (34, 4, 34)
 (35, 5, 35)
 (36, 6, 36)
 (37, 7, 37)
 (38, 8, 38)
 (39, 9, 39)
 (40, 10, 40)


RestaurantReviews_Junction
 (1, 1, 1, glad, 4)
 (2, 2, 2, glad, 5)
 (3, 3, 3, glad, 4)
 (4, 4, 4, glad, 3)
```

```
 (5, 5, 5, glad, 5)
 (6, 6, 6, glad, 4)
 (7, 7, 7, glad, 5)
 (8, 8, 8, sad, 2)
 (9, 9, 9, glad, 4)
 (10, 10, 10, glad, 5)
 (11, 1, 11, glad, 4)
 (12, 2, 12, glad, 5)
 (13, 3, 13, sad, 1)
 (14, 4, 14, glad, 4)
 (15, 5, 15, sad, 2)
 (16, 6, 16, glad, 4)
 (17, 7, 17, glad, 5)
 (18, 8, 18, sad, 1)
 (19, 9, 19, glad, 4)
 (20, 10, 20, glad, 5)
 (21, 1, 21, glad, 4)
 (22, 2, 22, glad, 5)
 (23, 3, 23, glad, 4)
 (24, 4, 24, sad, 2)
 (25, 5, 25, glad, 5)
 (26, 6, 26, glad, 4)
 (27, 7, 27, glad, 5)
 (28, 8, 28, sad, 1)
 (29, 9, 29, glad, 4)
 (30, 10, 30, glad, 5)
 (31, 1, 31, glad, 4)
 (32, 2, 32, glad, 5)
 (33, 3, 33, sad, 2)
 (34, 4, 34, glad, 4)
 (35, 5, 35, glad, 5)
 (36, 6, 36, glad, 4)
 (37, 7, 37, glad, 5)
 (38, 8, 38, sad, 1)
 (39, 9, 39, glad, 4)


FavouriteCouriers_Junction
 (1, 1, 1)
 (2, 2, 2)
 (3, 3, 3)
```

```
 (4, 4, 4)
 (5, 5, 5)
 (6, 6, 6)
 (7, 7, 7)
 (8, 8, 8)
 (9, 9, 9)
 (10, 10, 10)
 (11, 11, 11)
 (12, 12, 12)
 (13, 13, 13)
 (14, 14, 14)
 (15, 15, 15)
 (16, 16, 16)
 (17, 17, 17)
 (18, 18, 18)
 (19, 19, 19)
 (20, 20, 20)
 (21, 21, 21)
 (22, 22, 22)
 (23, 23, 23)
 (24, 24, 24)
 (25, 25, 25)
 (26, 26, 26)
 (27, 27, 27)
 (28, 28, 28)
 (29, 29, 29)
 (30, 30, 30)
 (31, 31, 31)
 (32, 32, 32)
 (33, 33, 33)
 (34, 34, 34)
 (35, 35, 35)
 (36, 36, 36)
 (37, 37, 37)
 (38, 38, 38)
 (39, 39, 39)
 (40, 40, 40)


CourierReviews_Junction
 (1, 1, 1, glad, 4)
```

```
(2, 2, 2, glad, 5)
(3, 3, 3, glad, 4)
(4, 4, 4, sad, 2)
(5, 5, 5, glad, 5)
(6, 6, 6, glad, 4)
(7, 7, 7, glad, 5)
(8, 8, 8, sad, 1)
(9, 9, 9, glad, 4)
(10, 10, 10, glad, 5)
(11, 11, 1, glad, 4)
(12, 12, 2, glad, 5)
(13, 13, 3, glad, 4)
(14, 14, 4, sad, 2)
(15, 15, 5, glad, 5)
(16, 16, 6, glad, 4)
(17, 17, 7, glad, 5)
(18, 18, 8, sad, 1)
(19, 19, 9, glad, 4)
(20, 20, 10, glad, 5)
(21, 21, 1, glad, 4)
(22, 22, 2, glad, 5)
(23, 23, 3, glad, 4)
(24, 24, 4, sad, 2)
(25, 25, 5, glad, 5)
(26, 26, 6, glad, 4)
(27, 27, 7, glad, 5)
(28, 28, 8, sad, 1)
(29, 29, 9, glad, 4)
(30, 30, 10, glad, 5)
(31, 31, 1, glad, 4)
(32, 32, 2, glad, 5)
(33, 33, 3, glad, 4)
(34, 34, 4, sad, 2)
(35, 35, 5, glad, 5)
(36, 36, 6, glad, 4)
(37, 37, 7, glad, 5)
(38, 38, 8, sad, 1)
(39, 39, 9, glad, 4)
```

```
Customers
```

```
(1, Jesper, Asian)
(2, Ole, Chinese)
(3, Flemming, American)
(4, Victor, Italian)
(5, Laust, French)
(6, Helga, Nordic)
(7, Thomas, Asian)
(8, Magnus, Chinese)
(9, Alti, American)
(10, Kristian, Italian)
```

e)

(i)

```
Input:
(EXISTS orderdate, cuisine, cuisineid, customerid, adress, rid, cid, n, nn.
UnitedOrder (id, orderdate, customerid, adress)
AND
Customers (customerid, name, cuisine)
AND
Restaurants (rid, n, adress, cuisineid)
AND
Couriers (cid, nn, adress))

Output:
(1,"Jesper")
(2,"Ole")
(3,"Flemming")
(4,"Victor")
(5,"Laust")
(6,"Helga")
(7,"Thomas")
(8,"Magnus")
(9,"Alti")
(10,"Kristian")
```

(ii)

```
Input:
EXISTS rat.(EXISTS rid, adress, cuisineid, favid, cid, revid, desc.
Restaurants (rid, n, adress, cuisineid)
AND
FavouriteRestaurantCustomer_Junction (favid, cid, rid)
AND
RestaurantReviews_Junction (revid, cid, rid, desc, rat)
AND
rat < 3)

Output:
("Indian_Spice")
("Sushi_Express")
("Thai_Garden")
("The_Italian_Place")
```

(iii)

```
Input:
EXISTS rat.(EXISTS rid, adress, favid, cid, revid, desc.
Couriers (cid, n, adress)
AND
FavouriteCouriers_Junction (favid, cid, rid)
AND
CourierReviews_Junction (revid, cid, rid, desc, rat)
AND
rat < 3)

Output:
("Ava")
("Emma")
```

(iv)

```
Input:
EXISTS crat, rrat.
(EXISTS customerid, costumercuisine, unitedid, orderdate, unitedadress,
restaurantreviewid, restaurantid, resrevdescription,
courierreviewid, courierid, courevdescription.
```

```
Customers (customerid, customername, costumercuisine)
AND
UnitedOrder (unitedid, orderdate, customerid, unitedadress)
AND
RestaurantReviews_Junction (restaurantreviewid, customerid,
restaurantid, resrevdescription, rrat)
AND
CourierReviews_Junction (courierreviewid, courierid, customerid,
courevdescription, crat)
AND
crat < 3
AND
rrat < 3
)

Output:
("Magnus")
("Victor")
```

2 Relational Calculus

a)

Query: EXISTS s, ram, hd, price. PC(model, s, ram, hd, price) AND ram=512

Result: (1002) (1003) (1005) (1013)

b)

Query:
EXISTS model, type, s, ram, hd, pr, screen. Product(maker, model, type)
AND
Laptop(model, s, ram, hd, screen, pr) AND screen=15

Result(4): ("A") ("E") ("F") ("G")

c)

Query:
EXISTS t. Product("E",model,t) AND
EXISTS s, r, hd. PC(model,s,r,hd,price) OR
EXISTS sp, ram, hd, sc. Laptop(model, sp, ram, hd, sc, price) OR
EXISTS c, t . Printer(model, c, t, price)

Result(9): (1011,959) (1012,649) (1013,529) (2001,3673) (2002,949) (2003,549) (3001,99) (3002,239) (3003,899)

d)

Query:
EXISTS t. Product(maker,model,t) AND
EXISTS s, r, hd. PC(model,s,r,hd,price) OR
EXISTS sp, ram, hd, sc. Laptop(model, sp, ram, hd, sc, price) OR
EXISTS c, t . Printer(model, c, t, price)

Result(30): ("A",1001,2114) ("A",1002,995) ("A",1003,478) ("A",2004,1150) ("A",2005,2500) ("A",2006,1700) ("B",1004,649) ("B",1005,630) ("B",1006,1049) ("B",2007,1429) ("C",1007,510) ("D",1008,770) ("D",1009,650) ("D",1010,770) ("D",3004,120) ("D",3005,120) ("E",1011,959) ("E",1012,649) ("E",1013,529) ("E",2001,3673) ("E",2002,949) ("E",2003,549) ("E",3001,99) ("E",3002,239)

("E",3003,899) ("F",2008,900) ("F",2009,680) ("G",2010,2300) ("H",3006,100) ("H",3007,200)

e)

Query:
EXISTS maker,t. Product(maker,model,t) AND
EXISTS c, t . Printer(model, c, t, price) AND
NOT maker ="E"AND
c = 1 AND
t = "laser"

Result: (3007,200)

f)

Query:
(EXISTS model, type1, speed, ram, hd, price1. Product(maker,model,type1) AND
PC (model, speed, ram, hd, price1)) AND NOT
(EXISTS modell, type2, color, type3, price. Product(maker,modell,type2) AND
Printer (modell, color, type3, price))

Result: ("A") ("B") ("C")

g)

Query :
EXISTS model, speed, ram, hd, price, model2, speed2, ram2, hd2, screen2, price2.
Laptop(model, speed, ram, hd, screen, price) AND
Laptop(model2, speed2, ram2, hd2, screen2, price2) AND
screen = screen2 AND NOT
model = model2
Result: (13) (15) (17)

h)

EXISTS hd2, hd3, hd4.
((EXISTS model1, type1, model2, speed2, ram2, screen2, price2.

Product(maker, model1, type1) AND
Laptop(model1, speed2, ram2, hd2, screen2, price2))AND
(EXISTS model1, type2, model3, speed3, ram3, screen3, price3.
Product(maker, model3, type2) AND
Laptop(model3, speed3, ram3, hd3, screen3, price3))AND
(EXISTS model1, type3, model4, speed4, ram4, screen4, price4.
Product(maker, model4, type3) AND
Laptop(model4, speed4, ram4, hd4, screen4, price4))AND
NOT hd2 = hd3 AND NOT hd3 = hd4 AND NOT hd2=hd4)
Result: ("A") ("E")

i)

EXISTS model1, model3, model4.
((EXISTS type1, color2, type2, price2.
Product(maker,model1,type1)AND
Printer(model1,color2,type2,price2))AND
(EXISTS type3, color4, type4, price4.
Product(maker,model3,type3)AND
Printer(model3,color4,type4,price4))AND
(EXISTS type5, color6, type6, price6.
Product(maker,model4,type5)AND
Printer(model4,color6,type6,price6))AND
NOT model1 = model3 AND NOT model1=model4 AND NOT model3=model4)
Result: ("E")

j)

EXISTS type2, type4.
((EXISTS model1, type1, color2, price2.
Product(maker,model1,type1)AND
Printer(model1,color2,type2,price2))AND
(EXISTS model3,type3, color4, price4.
Product(maker,model3,type3)AND
Printer(model3,color4,type4,price4))AND
NOT type2 = type4)
Result: ("D") ("E") ("H")