CPSC 304 Project Cover Page

Milestone #: **2**

Date: **10/15/2024**

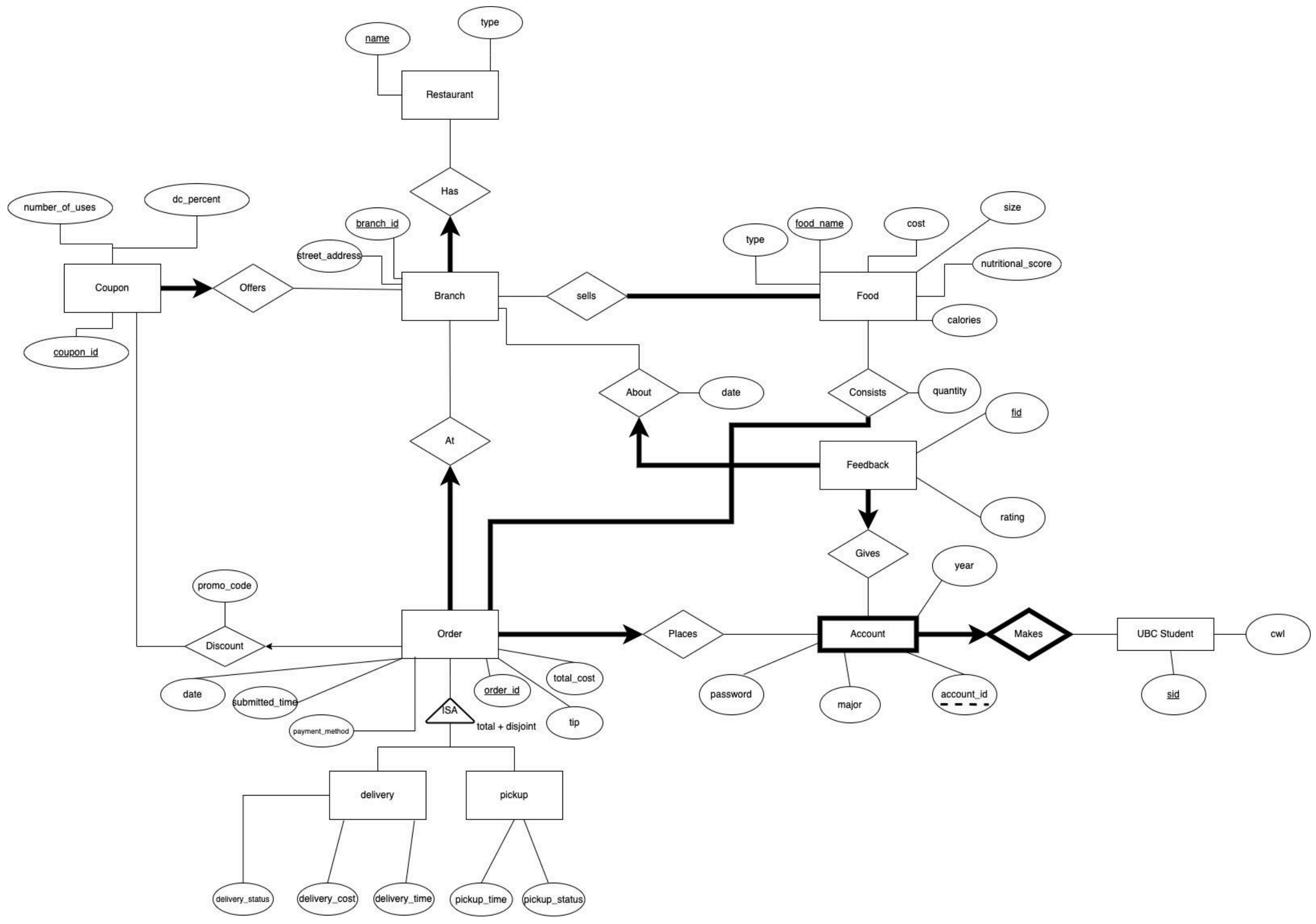
Group Number: **35**

Name	**Student Number**	**CS Alias (Userid)**	**Preferred E-mail Address**
Pearl (Jin Ju) Park	24181646	q1i7x	jinjpark@student.ubc.ca
Zayan Sheikh	23414329	m1g3t	zayans@student.ubc.ca
Daksh Mathur	45359395	w5i1d	dmathu01@student.ubc.ca

Summary of Project

Our application strives to make food at UBC-affiliated restaurants affordable and satisfying for UBC students, whilst also providing valuable business information and demographic data for restaurants. The data stored with our application will help UBC restaurants to figure out the food preference of UBC students based on their demographics and time of year. The students in turn will be able to buy tastier and more affordable food on campus based on their feedback as well as their food preferences, and the restaurants will be able to see this, while also being able to observe the types of students visiting them (what year, major, etc.), how much they're spending, etc.

ER Diagram



<u>**Changes made to ER:**</u>

- Attribute name changes:
    - **Account** – Changing id to account_id
    - Changing relationship name between **Feedback** and **Account** to “Gives” since already used relationship name makes for different relationship
    - Changing the “type” attribute to be specifically “rating” for **Feedback**
- Relationship changes:
    - Changing relationship from **Account** makes **UBC_Student** to a many to one relationship to allow a **UBC_Student** to make multiple **Accounts**
    - Making **Order** have full participation in relationship to **Coupon**
- Attribute additions:
    - Adding type and size attributes to **Food**
    - Adding quantity attribute to “consists” relationship
    - Adding 3 more attributes to **Delivery** and **Pickup** each
    - Adding submitted_time for **Order**
    - Adding quantity for **Consists**

<u>**Schema**</u>

- Restaurant(<u>name:</u> VARCHAR, type: VARCHAR)

    <u>**PK**</u>: name

    <u>**FK**</u>: no FKs

- Sells(<u>**food_name**</u>: CHAR(8)<u>**,**</u> <u>**branch_id:**</u>CHAR(5))

    <u>**PK:**</u> food_name, branch_id

    <u>**FK:**</u> food_name, branch_id

- Coupon(<u>coupon_id</u>: CHAR(8), **branch_id**: CHAR(5) NOT NULL, dc_percent: FLOAT, number_of_uses: INT)

    <u>**PK:**</u> coupon_id

    <u>**FK:**</u> branch_id

- Branch(<u>branch_id:</u> CHAR(5), street_address: VARCHAR, **restaurant_name**: VARCHAR NOT NULL)

    <u>**PK:**</u> branch_id

    <u>**FK:**</u> restaurant_name

CK: street_address

- Food(food_name: VARCHAR, cost: INT, calories: INT, nutritional_score: INT, type: VARCHAR, size: VARCHAR, **branch_id**: INT NOT NULL)

    PK: food_name

    FK: branch_id

- Consists(**order_id**: INT, **food_name**: VARCHAR NOT NULL, quantity: INTEGER)

    PK: (order_id, food_name)

    FK: order_id, food_name

- Feedback(fid: INTEGER, **account_id**: VARCHAR, **sid**: CHAR(8), date: DATE, **branch_id**: CHAR(5))

    PK: fid

    FK: account_id, sid, branch_id

- Account(account_id**:** VARCHAR, year: INT, major: INT, password: VARCHAR, **sid**: CHAR(8), cwl: VARCHAR)

    PK: (account_id, sid)

    FK: sid

- UBC_Student(sid: CHAR(8), cwl: VARCHAR)

    PK: sid

    FK: none

- Delivery(order_id: INT, total_cost: DECIMAL(10,2), date: DATE, payment_method: VARCHAR, promo_code: VARCHAR, **coupon_id**: CHAR(8)**, branch_id**: CHAR(5) NOT NULL**, account_id**: VARCHAR NOT NULL, **sid**: CHAR(8) NOT NULL, delivery_cost: DECIMAL(4,2), delivery_status VARCHAR, delivery_time FLOAT)

    PK: order_id

    FK: coupon_id, branch_id, account_id, sid

- Pickup(order_id: INT, total_cost: DECIMAL(10,2), date: DATE, payment_method: VARCHAR, promo_code: VARCHAR, **coupon_id**: CHAR(8)**, branch_id**: CHAR(5) NOT NULL**, account_id**: VARCHAR NOT NULL, **sid**: CHAR(8) NOT NULL, pickup_time: FLOAT, pickup_status: VARCHAR)
  - **PK:** order_id
  - **FK:** coupon_id, branch_id, account_id, sid

FDs

1. Restaurant

- **FDs**:
  - name → type

2. Branch

- **FDs**:
  - branch_id → street_address, restaurant_name
  - street_address → branch_id

3. Food

- **PK**: **food_name**
- **FDs**:
  - food_name → cost, calories, nutritional_score, size, type, branch_id
  - Calories, type, size → nutritional_score

4. Coupon

- **FDs**:
  - Coupon_id → number_of_uses, dc_percent, branch_id

5. Feedback

- **FDs**
  - **FD1**: fid → rating, date, food_name, account_id, sid, branch_id
  - **FD2**: account_id, sid, date, branch_id → rating

6. Account

- **PK**: **id**
- **FDs**:
  - id → password, major, sid
  - sid → cwl (The student's sid determines the CWL login)

7. Delivery

- **PK**: **order_id**
- **FDs**:
    - Delivery_cost, coupon_id, order_id→ total_cost
    - Order_id → total_cost, date, payment_method, coupon_id, branch_id, account_id, promo_code, sid, delivery_cost

8. Pickup

- **PK**: **order_id**
    - Order_id → total_cost, date, payment_method, coupon_id, branch_id, account_id, promo_code,sid

9. UBC Student

- **PK**: **sid**
- **FDs**:
    - sid → cwl (Each student is uniquely identified by their sid, which determines their CWL login)
    - **Non-trivial FD**: sid → major, year (The student ID may also determine their major and academic year)

10. Consists:

- **FDs**:
    - No non trivial FDs

11. Sells:

- **FDs**:
    - No non trivial FDs

Normalization

The relations Food and Feedback are not in BCNF, we use lossless-join BCNF decomposition algorithm for normalization below:

1. Food (food_name, cost, calories, nutritonal_score, branch_id, type, size)

Legend:

food_name = F
cost = C

calories = CA,
nutritional_score = N
branch_id = B
type = T
size = S

R(F, C, CA, N, B, T, S)

Current FDs:

F → C, CA, N, S, T, B
CA, T, S → N
S → C

Closures

F += C, CA, N, S, T, B
CA, T, S += CA, T, S, N, C
S += S, C

Key: F

*** CA, T, S → N and S → C are not superkeys; violates BCNF; decompose

BCNF Decomposition

- Decompose R on CA, T, S → N:
  R_1(CA, T, S, N), R_2(S, T, CA, B, C, F)

- S → C violates BCNF decompose R_2 on S → C
  R_3(S C), R_4(CA, T, S, B, F)

Final relations: R_1**(**CA, T, S, N**),** R_3**(S,** C), R_4(F, B**, CA, T, S**)

Tables after decomposition of Food:

Food_Information(calories: INT, type: VARCHAR, size: VARCHAR, nutritional_score: INT)

PK: type, calories, size

FK: none

Food_Cost(**size**: VARCHAR, cost: DECIMAL(4,2))

PK: size

FK: size

Food(<u>food_name:</u> VARCHAR, **type:** VARCHAR, **calories**: INT, **size**: VARCHAR, **branch_id**: INT NOT NULL)

<u>PK:</u> food_name

<u>FK:</u> type, calories, size, branch_id

2. Feedback (fid, rating, account_id, sid, date, branch_id)

<u>Legend:</u>

Fid = Feedback ID (fid)
R = Rating (rating)
AccID = Account ID (account_id)
Sid = Student ID (sid)
D = Date (date)
BID = Branch ID (branch_id)

<u>Current FDs:</u>

1. Fid → R, AccID, Sid, D, BID (Feedback ID determines everything else)
2. AccID, Sid, D, BID → R (The combination of Account, Student, Date, and Branch determines the Rating)

<u>Closures:</u>

- Fid+ = Fid, R, AccID, Sid, D, BID
    - The closure of Fid (Feedback ID) includes all attributes, so Fid is the primary key.
- AccID, Sid, D, BID+ = AccID, Sid, D, BID, R
    - This combination of attributes determines the Rating (R), but does not determine Fid, so this combination is not a superkey.

<u>BCNF Decomposition:</u>

According to BCNF, every functional dependency (FD) must have a superkey on the left-hand side. In the current structure:

- FD1: Fid → R, AccID, Sid, D, BID is fine, because Fid is the primary key (a superkey).
- FD2: AccID, Sid, D, BID → R violates BCNF, because AccID, Sid, D, BID is not a superkey.

This violation of BCNF means we need to decompose the table based on FD2.

To resolve the BCNF violation, we decompose the Feedback table into two relations.

Decompose Feedback(Fid, R, AccID, Sid, D, BID) based on FD2 (AccID, Sid, D, BID → R):

1. R1(AccID, Sid, D, BID, R): This relation stores the combination of Account ID, Student ID, Date, Branch ID, and the associated Rating.
   - Key: (AccID, Sid, D, BID)
2. R2(Fid, AccID, Sid, D, BID): This relation links the Feedback ID (Fid) with the combination of Account ID, Student ID, Date, and Branch ID.
   - Key: Fid

After decomposition, we need to check if both new relations are in BCNF.

R1(<u>AccID, Sid, D, BID</u>, R):

- FD: AccID, Sid, D, BID → R
- The composite key (AccID, Sid, D, BID) is a superkey and determines all attributes in R1, so this relation is in BCNF.

R2(<u>Fid</u>, AccID, Sid, D, BID):

- FD: Fid → AccID, Sid, D, BID
- The primary key Fid determines all other attributes in R2, so this relation is also in BCNF.

Final relations: R1(<u>AccID, Sid, D, BID</u>, R), R2(<u>Fid</u>, AccID, Sid, D, BID)

<u>Tables after decomposition for feedback:</u>

Feedback_Rating (<u>account_id</u>: VARCHAR, <u>sid</u>: CHAR(8), <u>date</u>: DATE, <u>branch_id</u>: CHAR(5), rating: INTEGER)

<u>PK:</u> (account_id, sid, date, branch_id)

<u>FK:</u> none

Feedback_Link (<u>fid</u>: INTEGER, **account_id**: VARCHAR, **sid**: CHAR(8), date: DATE, **branch_id**: CHAR(5))

<u>PK:</u> fid

<u>FK:</u> account_id, sid, branch_id

SQL DDL Command

```
CREATE TABLE Feedback_Rating (
    account_id VARCHAR NOT NULL,
    sid CHAR(8) NOT NULL,
    date DATE NOT NULL,
    branch_id CHAR(5) NOT NULL,
    rating INTEGER NOT NULL,
    PRIMARY KEY (account_id, sid, date, branch_id),
    FOREIGN KEY (account_id) REFERENCES Account (account_id) ON
DELETE CASCADE,
    FOREIGN KEY (sid) REFERENCES UBC_Student (sid) ON DELETE CASCADE,
    FOREIGN KEY (branch_id) REFERENCES Branch (branch_id) ON DELETE
CASCADE
);

CREATE TABLE Feedback_Link (
    fid INTEGER PRIMARY KEY,
    account_id VARCHAR NOT NULL,
    sid CHAR(8) NOT NULL,
    date DATE NOT NULL,
    branch_id CHAR(5) NOT NULL,
    FOREIGN KEY (account_id) REFERENCES Account (account_id) ON
DELETE CASCADE,
    FOREIGN KEY (sid) REFERENCES UBC_Student (sid) ON DELETE CASCADE,
    FOREIGN KEY (branch_id) REFERENCES Branch (branch_id) ON DELETE
CASCADE
);


CREATE TABLE Account (
    account_id VARCHAR,
    year INTEGER,
    major INTEGER,
    password VARCHAR,
    sid CHAR(8),
    cwl VARCHAR,
    PRIMARY KEY (account_id, sid),
    FOREIGN KEY (sid) REFERENCES UBC_Student(sid) ON DELETE CASCADE
```

```
);


CREATE TABLE UBC_Student(
    sid CHAR(8) PRIMARY KEY,
    cwl VARCHAR
);

CREATE TABLE Food (
     food_name VARCHAR PRIMARY KEY,
     type VARCHAR,
     calories INTEGER,
     size VARCHAR,
     branch_id CHAR(5) NOT NULL,
     FOREIGN KEY (type, calories, size) REFERENCES Food_Information
     (cost, calories) ON DELETE CASCADE,
     FOREIGN KEY (branch_id) REFERENCES Branch (branch_id) ON DELETE
     CASCADE
);

CREATE TABLE Food_Information (
     calories INTEGER,
     type VARCHAR,
     size VARCHAR,
     nutritional_score INTEGER,
     PRIMARY KEY (type, calories, size)
);


CREATE TABLE Food_Cost (
     size VARCHAR PRIMARY KEY,
     cost DECIMAL(4,2)
     FOREIGN KEY (size) REFERENCES Food_Information(size)
);

CREATE TABLE Restaurant (
     name VARCHAR PRIMARY KEY,
     type VARCHAR,
);

CREATE TABLE Branch (
     branch_id CHAR(5) PRIMARY KEY,
     street_address VARCHAR,
     restaurant_name VARCHAR NOT NULL,
     FOREIGN KEY (restaurant_name) REFERENCES Restaurant ON DELETE
CASCADE
);

CREATE TABLE Sells (
     food_name CHAR(8),
     branch_id CHAR(5),
     PRIMARY KEY (food_name, branch_id)
     FOREIGN KEY (food_name) REFERENCES Food ON DELETE CASCADE,
     FOREIGN KEY (branch_id) REFERENCES Branch ON DELETE CASCADE
);


CREATE TABLE Coupon (
     coupon_id CHAR(8) PRIMARY KEY,
     branch_id CHAR(5) NOT NULL,
     dc_percent FLOAT,
     number_of_uses INTEGER,
     FOREIGN KEY(branch_id) REFERENCES Branch ON DELETE CASCADE
);

CREATE TABLE Consists (
    order_id INTEGER,
    food_name VARCHAR NOT NULL,
    quantity INTEGER NOT NULL,
    PRIMARY KEY (order_id, food_name),
    FOREIGN KEY (order_id) REFERENCES Delivery (order_id) ON DELETE
CASCADE,
    FOREIGN KEY (food_name) REFERENCES Food (food_name) ON DELETE
CASCADE
);

CREATE TABLE Delivery(
     order_id INTEGER PRIMARY KEY,
     total_cost DECIMAL(10,2),
     date DATE
     payment_method VARCHAR,
     promo_code VARCHAR,
     coupon_id CHAR(8) DEFAULT NULL,
     branch_id CHAR(5) NOT NULL,
     account_id VARCHAR NOT NULL,
     sid CHAR(8) NOT NULL,
     delivery_cost DECIMAL(4,2),
     delivery_status VARCHAR,
     delivery_time FLOAT,
      FOREIGN KEY (coupon_id) REFERENCES Coupon
           ON DELETE SET DEFAULT,
      FOREIGN KEY (branch_id) REFERENCES Branch
           ON DELETE CASCADE,
      FOREIGN KEY (account_id) REFERENCES Account
           ON DELETE CASCADE,
      FOREIGN KEY (sid) REFERENCES UBC_Student
           ON DELETE CASCADE
);

CREATE TABLE Pickup
(
      order_id INTEGER PRIMARY KEY,
      total_cost DECIMAL(10,2),
      date DATE
      payment_method VARCHAR,
      promo_code VARCHAR,
      coupon_id CHAR(8) DEFAULT NULL,
      branch_id CHAR(5) NOT NULL,
      account_id VARCHAR NOT NULL,
      sid CHAR(8) NOT NULL,
      pickup_time FLOAT,
      pickup_status VARCHAR,
      FOREIGN KEY (coupon_id) REFERENCES Coupon
           ON DELETE SET DEFAULT,
      FOREIGN KEY (branch_id) REFERENCES Branch
           ON DELETE CASCADE,
      FOREIGN KEY (account_id) REFERENCES Account
           ON DELETE CASCADE,
      FOREIGN KEY (sid) REFERENCES UBC_Student
           ON DELETE CASCADE
)
```

INSERT statements

```
INSERT INTO Restaurant VALUES(‘Starbucks’, ‘cafe’)
INSERT INTO Restaurant VALUES(‘Triple_Os’, ‘restaurant’)
INSERT INTO Restaurant VALUES(‘Harvest’, ‘grocery_store’)
INSERT INTO Restaurant VALUES(‘Subway’, ‘restaurant’)
INSERT INTO Restaurant VALUES(‘Pacific_Poke’, ‘restaurant’)
INSERT INTO Branch VALUES(‘S0002’, ‘6138 Student Union Blvd’,
‘Starbucks’)
INSERT INTO Branch VALUES(‘T0001’, ‘2015 Main Mall’, ‘Triple_Os’)
INSERT INTO Branch VALUES(‘H0001’, ‘6445 University Blvd’, ‘Harvest’)
INSERT INTO Branch VALUES(‘S0001’, ‘6138 Student Union Blvd’,
‘Subway’)
INSERT INTO Branch VALUES(‘P0001’, ‘6138 Student Union Blvd’,
‘Pacific_Poke’)

INSERT INTO Food_Information VALUES(10, ‘beverage’, ‘small’, 1)
INSERT INTO Food_Information VALUES(930, ‘food’, ‘large’, 3)
INSERT INTO Food_Information VALUES(316, ‘food’, ‘regular’, 8)
INSERT INTO Food_Information VALUES(207, ‘food’, ‘foot_long’,  6)
INSERT INTO Food_Information VALUES(850, ‘food’, ‘regular’,  7)

INSERT INTO Food VALUES(‘americano’, ’food’, 10, ‘small’, ‘S0002’)
INSERT INTO Food VALUES(‘cheese_burger’, ’food’, 930, ‘large’,
‘T0001’)
INSERT INTO Food VALUES(‘ceasar_salad’, ’food’, 316,‘regular’,
‘H0001’)
INSERT INTO Food VALUES(‘roasted_chicken_sub’, ’food’, 207,
‘foot_long’, ‘S0001’)
INSERT INTO Food VALUES(‘main_bowl’, ’food’, 850,‘regular’, ‘P0001’)

INSERT INTO Food_Cost VALUES(‘small’, 5.45)
INSERT INTO Food_Cost VALUES(‘large’, 12.00)
INSERT INTO Food_Cost VALUES(‘regular’, 8.00)
INSERT INTO Food_Cost VALUES(‘foot_long’, 8.79)
INSERT INTO Food_Cost VALUES(‘regular’, 18.00)

INSERT INTO Feedback_Rating VALUES ('acc001', '21402983',
'2024-10-12', 'S0002', 5);
INSERT INTO Feedback_Rating VALUES ('acc002', '85392374',
'2024-10-12', 'T0001', 4);
INSERT INTO Feedback_Rating VALUES ('acc003', '24712948',
'2024-10-12', 'H0001', 3);
INSERT INTO Feedback_Rating VALUES ('acc004', '41238733',
'2024-10-12', 'S0001', 5);
INSERT INTO Feedback_Rating VALUES ('acc005', '28232237',
'2024-10-12', 'P0001', 4);

INSERT INTO Feedback_Link VALUES (1001, 'acc001', '21402983',
'2024-10-12', 'S0002');
INSERT INTO Feedback_Link VALUES (1002, 'acc002', '85392374',
'2024-10-12', 'T0001');
```

```
INSERT INTO Feedback_Link VALUES (1003, 'acc003', '24712948',
'2024-10-12', 'H0001');
INSERT INTO Feedback_Link VALUES (1004, 'acc004', '41238733',
'2024-10-12', 'S0001');
INSERT INTO Feedback_Link VALUES (1005, 'acc005', '28232237',
'2024-10-12', 'P0001');

INSERT INTO UBC_Student VALUES(‘21402983’, ‘ssj123’)
INSERT INTO UBC_Student VALUES(‘85392374’, ‘hi1928’)
INSERT INTO UBC_Student VALUES(‘24712948’, ‘jpark’)
INSERT INTO UBC_Student VALUES(‘41238733’, ‘johndoe’)
INSERT INTO UBC_Student VALUES(‘28232237’, ‘p133’)

INSERT INTO Account VALUES ('acc001', 2024, 1, 'pass123', '21402983',
'ssj123');
INSERT INTO Account VALUES ('acc002', 2024, 2, 'pass456', '85392374',
'hi1928');
INSERT INTO Account VALUES ('acc003', 2024, 3, 'pass789', '24712948',
'jpark');
INSERT INTO Account VALUES ('acc004', 2024, 4, 'pass321', '41238733',
'johndoe');
INSERT INTO Account VALUES ('acc005', 2024, 5, 'pass654', '28232237',
'p133');

INSERT INTO Sells VALUES(‘americano’, ‘S0002’)
INSERT INTO Sells VALUES(‘cheese_burger’, ‘T0001’)
INSERT INTO Sells VALUES(‘ceasar_salad’, ‘H0001’)
INSERT INTO Sells VALUES(‘roasted_chicken_sub’, ‘S0001’)
INSERT INTO Sells VALUES(‘main_bowl’, ‘P0001’)

INSERT INTO Consists VALUES (1, 'americano', 1);
INSERT INTO Consists VALUES (2, 'cheese_burger', 1);
INSERT INTO Consists VALUES (3, 'ceasar_salad', 1);
INSERT INTO Consists VALUES (4, 'roasted_chicken_sub', 1);
INSERT INTO Consists VALUES (5, 'main_bowl', 1);

INSERT INTO Coupon VALUES(‘2G2303D3’, ‘S0002’, 0.15, 4)
INSERT INTO Coupon VALUES(‘B152R99G’, ‘T0001’, 0.25, 2)
INSERT INTO Coupon VALUES(‘K0E5G001’, ‘H0001’, 0.02, 1)
INSERT INTO Coupon VALUES(‘B0F13D01’, ‘H0001’, 0.05, 0)
INSERT INTO Coupon VALUES(‘P0F33N20’, ‘P0001’, 0.10, 0)

INSERT INTO Delivery VALUES (1, 10.99, TO_DATE('17/12/2015',
'DD/MM/YYYY'), ‘Debit’,’FAKECOUPON’,‘2G2303D3’,
‘S0002’,'acc001','21402983',2.99, ‘Complete’, 1.2
);
INSERT INTO Delivery VALUES (2, 50.49, TO_DATE('13/10/2024',
'DD/MM/YYYY'), ‘Credit’,’LOL’,‘B152R99G’,
‘T0001’,'acc002','85392374',2.99, ‘Complete’, 0.2
);
INSERT INTO Delivery VALUES (3, 29.99, TO_DATE('17/12/2015',
'DD/MM/YYYY'), ‘Debit’, ’COUPON1’,’K0E5G001’,
‘H0001’,'acc003','24712948',2.99, ‘Placed’, 2
);
INSERT INTO Delivery VALUES (4, 100000.99, TO_DATE('15/10/2024',
'DD/MM/YYYY'), ‘Cash’, ’FAKECOUPON’,‘2G2303D3’,
‘S0002’,'acc004','41238733',4.99, ‘Delivering’, 1
);
INSERT INTO Delivery VALUES (5, 3.99, TO_DATE('17/12/2015',
'DD/MM/YYYY'), ‘Credit’,’WEWANTAGOODGRADE100’,‘B0F13D01’,
‘H0001’,'acc005','28232237',0.99, ‘Placed’, 777
);

INSERT INTO Pickup VALUES (1, 10.99, TO_DATE('17/12/2015',
'DD/MM/YYYY'), ‘Debit’,’’, NULL, ‘S0002’,'acc001','21402983',1,
‘Complete’
);
INSERT INTO Pickup VALUES (2, 50.49, TO_DATE('13/10/2024',
'DD/MM/YYYY'), ‘Credit’,’LOL’,‘B152R99G’,
‘T0001’,'acc002','85392374',0.9, ‘Complete’
);
INSERT INTO Pickup VALUES (3, 29.99, TO_DATE('17/12/2015',
'DD/MM/YYYY'), ‘Debit’,’Coupon2’,2G2303D3,
‘S0002’,'acc003','24712948',0.2, ‘Placed’
);
INSERT INTO Pickup VALUES (4, 100000.99, TO_DATE('15/10/2024',
'DD/MM/YYYY'), ‘Cash’,’FAKECOUPON’,‘2G2303D3’,
‘S0002’,'acc004','41238733',1, ‘Delivering’
);
INSERT INTO Pickup VALUES (5, 3.99, TO_DATE('17/12/2015', 'DD/MM/YYYY'),
‘Credit’,’WEWANTAGOODGRADE100’,‘B0F13D01’,
‘H0001’,'acc005','28232237',0.12, ‘Placed’
);
```