Table of Contents

SQL statements to solve the queries in Appendix B and additional queries

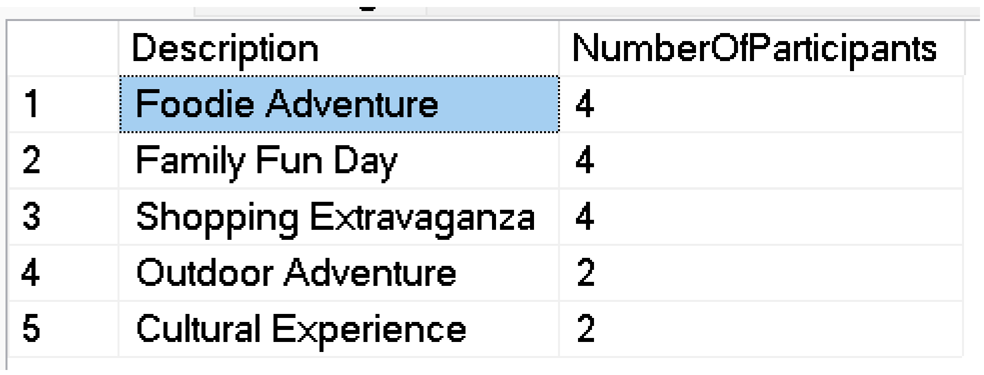
Query 1

Find the most popular day packages where all participants are related to one another as either family members or members of the same club.

Code

```sql
SELECT TOP 5
    dp.Description,
    COUNT(*) AS NumberOfParticipants
FROM
    DAY_PACKAGE dp
    INNER JOIN USER_ACCOUNT ua ON dp.UID = ua.UID
    INNER JOIN RELATED r ON ua.UID = r.Person1_UID OR ua.UID = r.Person2_UID
WHERE
    r.Type IN ('Family', 'Club')
GROUP BY
    dp.Description
ORDER BY
    NumberOfParticipants DESC;
```

Output



	Description	NumberOfParticipants
1	Foodie Adventure	4
2	Family Fun Day	4
3	Shopping Extravaganza	4
4	Outdoor Adventure	2
5	Cultural Experience	2

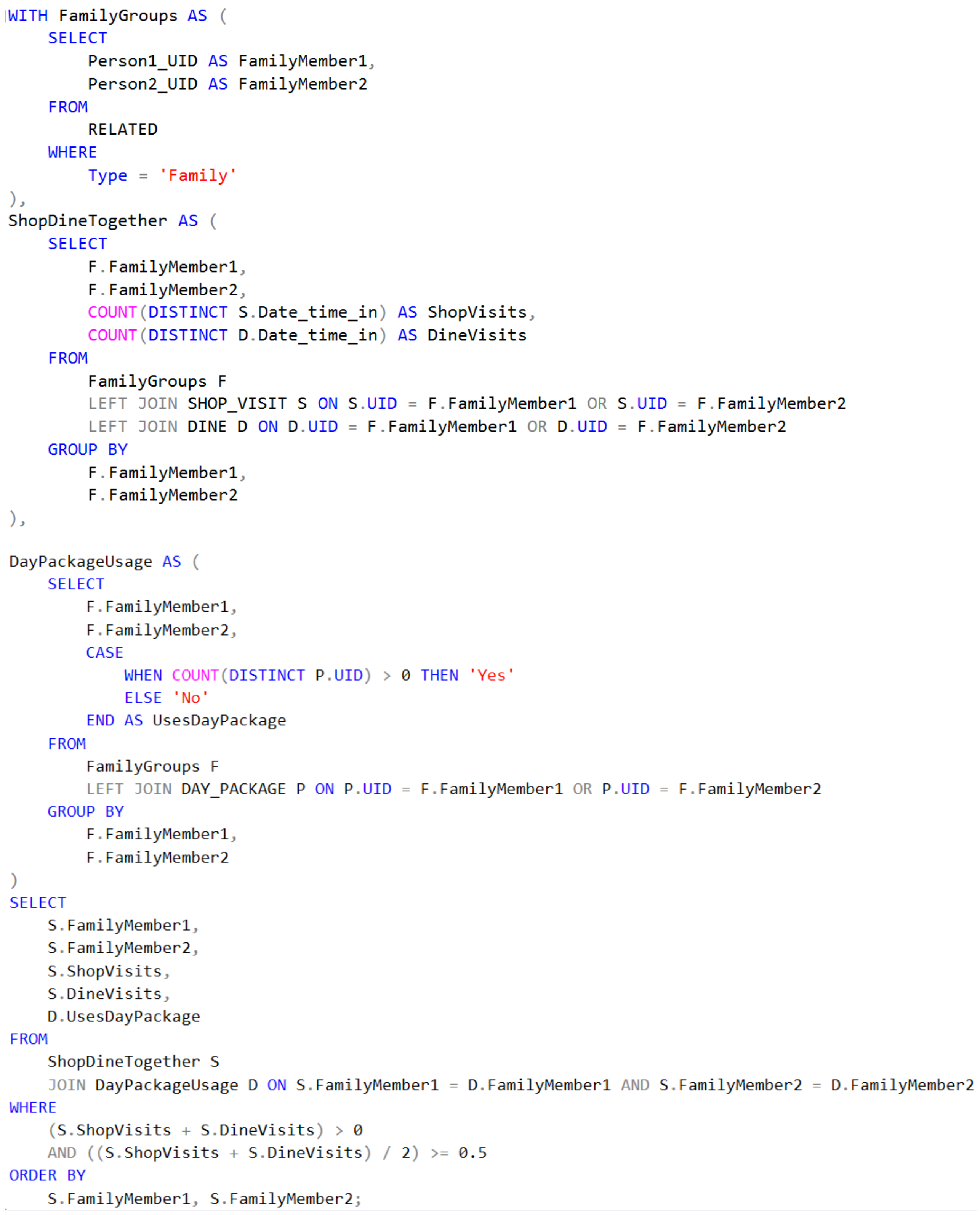
Explain on how the output is obtained

We first perform an inner join with USER_ACCOUNT and DAY_PACKAGE tables to link them together. Then we inner join USER_ACCOUNT and RELATED tables to find the relationships between users. We select the day packages where its users have the relationship type of 'Family' or 'Club', before counting the number of users tied to each day package. Finally we display the day packages in descending order, where the more participants, the more popular the day package.

Query 2

Find families who frequently shopped and dined together, with or without day packages. As part of your output, indicate whether these families use day packages or not. “frequently” means at least 50% of the time.

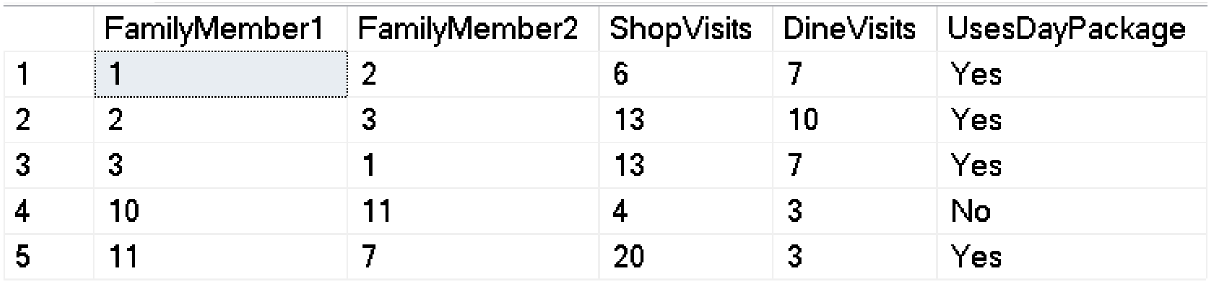
Code



```
WITH FamilyGroups AS (
    SELECT
        Person1_UID AS FamilyMember1,
        Person2_UID AS FamilyMember2
    FROM
        RELATED
    WHERE
        Type = 'Family'
),
ShopDineTogether AS (
    SELECT
        F.FamilyMember1,
        F.FamilyMember2,
        COUNT(DISTINCT S.Date_time_in) AS ShopVisits,
        COUNT(DISTINCT D.Date_time_in) AS DineVisits
    FROM
        FamilyGroups F
        LEFT JOIN SHOP_VISIT S ON S.UID = F.FamilyMember1 OR S.UID = F.FamilyMember2
        LEFT JOIN DINE D ON D.UID = F.FamilyMember1 OR D.UID = F.FamilyMember2
    GROUP BY
        F.FamilyMember1,
        F.FamilyMember2
),

DayPackageUsage AS (
    SELECT
        F.FamilyMember1,
        F.FamilyMember2,
        CASE
            WHEN COUNT(DISTINCT P.UID) > 0 THEN 'Yes'
            ELSE 'No'
        END AS UsesDayPackage
    FROM
        FamilyGroups F
        LEFT JOIN DAY_PACKAGE P ON P.UID = F.FamilyMember1 OR P.UID = F.FamilyMember2
    GROUP BY
        F.FamilyMember1,
        F.FamilyMember2
)
SELECT
    S.FamilyMember1,
    S.FamilyMember2,
    S.ShopVisits,
    S.DineVisits,
    D.UsesDayPackage
FROM
    ShopDineTogether S
    JOIN DayPackageUsage D ON S.FamilyMember1 = D.FamilyMember1 AND S.FamilyMember2 = D.FamilyMember2
WHERE
    (S.ShopVisits + S.DineVisits) > 0
    AND ((S.ShopVisits + S.DineVisits) / 2) >= 0.5
ORDER BY
    S.FamilyMember1, S.FamilyMember2;
```

Output



	FamilyMember1	FamilyMember2	ShopVisits	DineVisits	UsesDayPackage
1	1	2	6	7	Yes
2	2	3	13	10	Yes
3	3	1	13	7	Yes
4	10	11	4	3	No
5	11	7	20	3	Yes

Explain on how the output is obtained

We perform the query using 3 tables:

FamilyGroups: Lists all pairs of family members.

ShopDineTogether: Counts the number of times each family pair shopped and dined together.

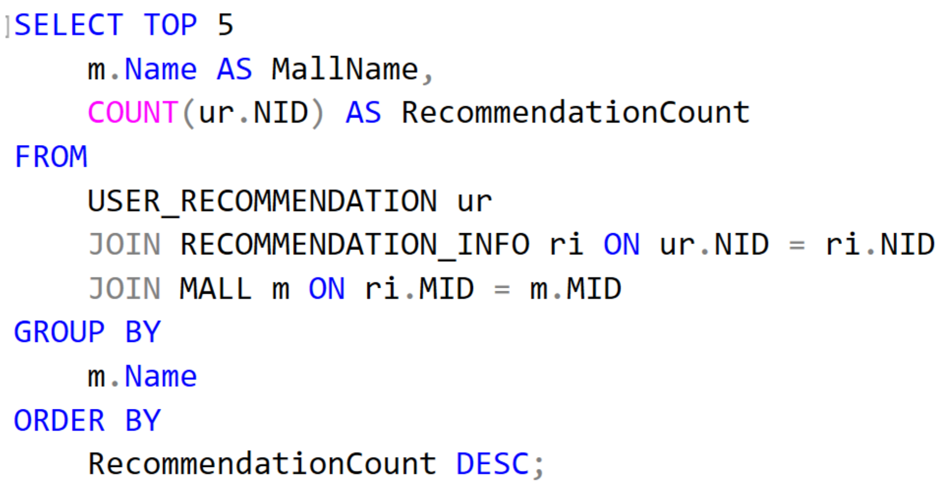
DayPackageUsage: Determines whether each family pair used day packages during their activities.

The final SELECT statement combines these results and filters for families that meet the criteria of shopping and dining together frequently (at least 50% of the time). It also indicates whether these families used day packages or not.

Query 3

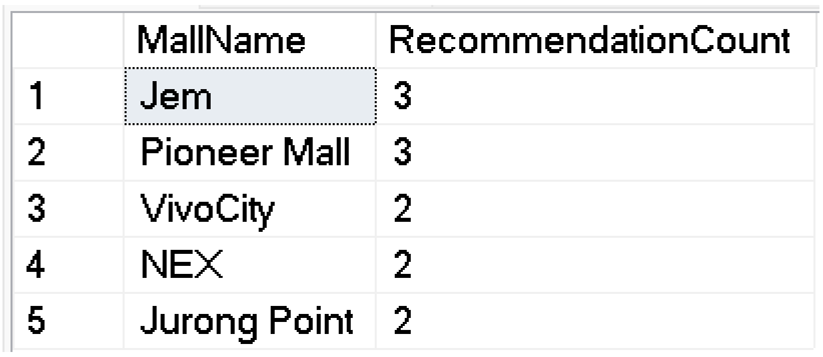
What are the most popular recommendations from the app regarding malls?

Code



```
SELECT TOP 5
    m.Name AS MallName,
    COUNT(ur.NID) AS RecommendationCount
FROM
    USER_RECOMMENDATION ur
    JOIN RECOMMENDATION_INFO ri ON ur.NID = ri.NID
    JOIN MALL m ON ri.MID = m.MID
GROUP BY
    m.Name
ORDER BY
    RecommendationCount DESC;
```

Output



	MallName	RecommendationCount
1	Jem	3
2	Pioneer Mall	3
3	VivoCity	2
4	NEX	2
5	Jurong Point	2

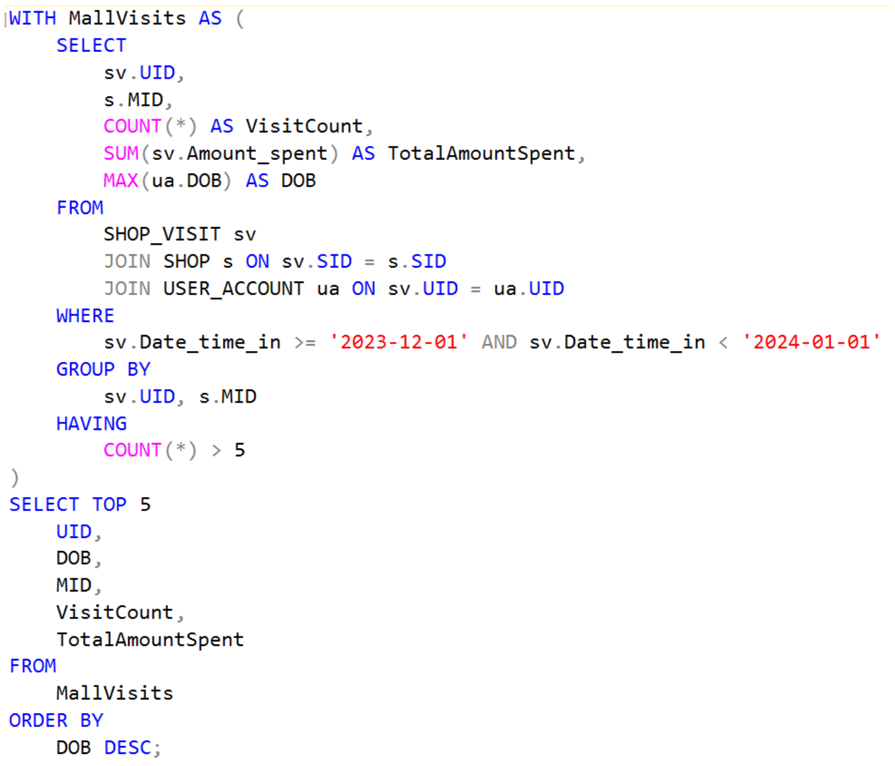
Explain on how the output is obtained

To find the most popular malls that are being recommended to users, we first join USER_RECOMMENDATION and RECOMMENDATION_INFO tables together to link the malls that are being recommended to the user. Next, we join the MALL table to retrieve the malls' names. Finally for each mall name, we count the number of users said mall is being recommended to, and display the top 5 results.

Query 4

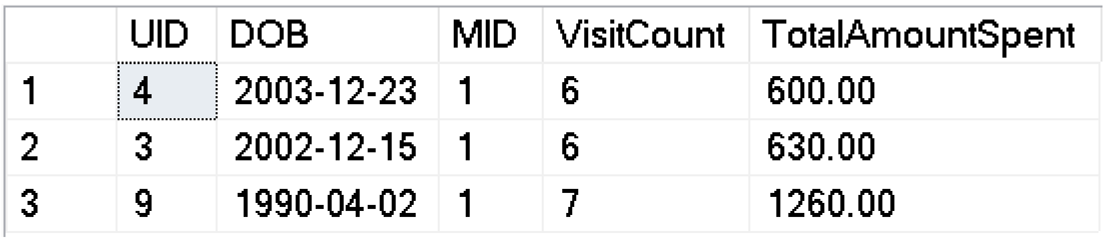
Compulsive shoppers are those who have visited a certain mall more than 5 times within a certain period of time. Find the youngest compulsive shoppers and the amount they spent in total during December 2023

Code



```
WITH MallVisits AS (
    SELECT
        sv.UID,
        s.MID,
        COUNT(*) AS VisitCount,
        SUM(sv.Amount_spent) AS TotalAmountSpent,
        MAX(ua.DOB) AS DOB
    FROM
        SHOP_VISIT sv
        JOIN SHOP s ON sv.SID = s.SID
        JOIN USER_ACCOUNT ua ON sv.UID = ua.UID
    WHERE
        sv.Date_time_in >= '2023-12-01' AND sv.Date_time_in < '2024-01-01'
    GROUP BY
        sv.UID, s.MID
    HAVING
        COUNT(*) > 5
)
SELECT TOP 5
    UID,
    DOB,
    MID,
    VisitCount,
    TotalAmountSpent
FROM
    MallVisits
ORDER BY
    DOB DESC;
```

Output



	UID	DOB	MID	VisitCount	TotalAmountSpent
1	4	2003-12-23	1	6	600.00
2	3	2002-12-15	1	6	630.00
3	9	1990-04-02	1	7	1260.00

Explain on how the output is obtained

We first perform a subquery. In subquery, we join the SHOP_VIST, SHOP and USER_ACCOUNT together to link which shops any said user visits in the month of December. Shops belong to a mall. For each user and mall combination, the subquery counts the number of visits and sums up the total amount spent during the specified period. This is done by grouping the records by the user ID and the mall ID (MID). Next we use the HAVING clause to filter out users who visited a single mall more than 5 times in December 2023. Finally, we SELECT the relevant columns to display from the subquery.

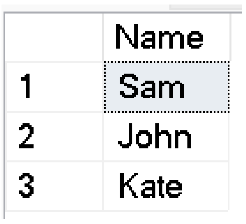
Query 5

Find users who have dined in all the restaurants in some malls, but have never dined in any restaurants in some other malls.

Code

```sql
WITH MallDineCounts AS (
    SELECT
        r.MID,
        COUNT(DISTINCT r.OID) AS TotalOutlets
    FROM
        RESTAURANT_OUTLET r
    GROUP BY
        r.MID
),
UserDineCounts AS (
    SELECT
        d.UID,
        r.MID,
        COUNT(DISTINCT d.OID) AS DinedOutlets
    FROM
        DINE d
        JOIN RESTAURANT_OUTLET r ON d.OID = r.OID
    GROUP BY
        d.UID, r.MID
),
UsersWithCompleteAndIncompleteDining AS (
    SELECT
        udc.UID
    FROM
        UserDineCounts udc
        JOIN MallDineCounts mdc ON udc.MID = mdc.MID
    GROUP BY
        udc.UID
    HAVING
        SUM(CASE WHEN udc.DinedOutlets = mdc.TotalOutlets THEN 1 ELSE 0 END) > 0
        AND SUM(CASE WHEN udc.DinedOutlets < mdc.TotalOutlets THEN 1 ELSE 0 END) > 0
)
SELECT
    ua.Name
FROM
    UsersWithCompleteAndIncompleteDining uc
    JOIN USER ACCOUNT ua ON uc.UID = ua.UID;
```

Output



	Name
1	Sam
2	John
3	Kate

Explain on how the output is obtained

MallDineCounts calculates the total number of restaurant outlets in each mall by counting the distinct restaurant outlet IDs from the RESTAURANT_OUTLET table and grouping the results by mall ID.

UserDineCounts We then count the number of distinct restaurant outlets each user has dined in, grouped by mall. It joins the DINE table (which records dining instances) with the

RESTAURANT_OUTLET table (which links outlets to malls) and groups the results by user ID and mall ID.

UsersWithCompleteAndIncompleteDining we then join the previous 2 tables and select users who dined in all the outlets of at least one mall (udc.DinedOutlets = mdc.TotalOutlets) and having dined in fewer than all the outlets of at least one other mall (udc.DinedOutlets < mdc.TotalOutlets)

Finally we join UsersWithCompleteAndIncompleteDining with USER_ACCOUNT table to SELECT the names of the users and return result.

Query 6

What are the top 3 highest earning malls and restaurants?

Code

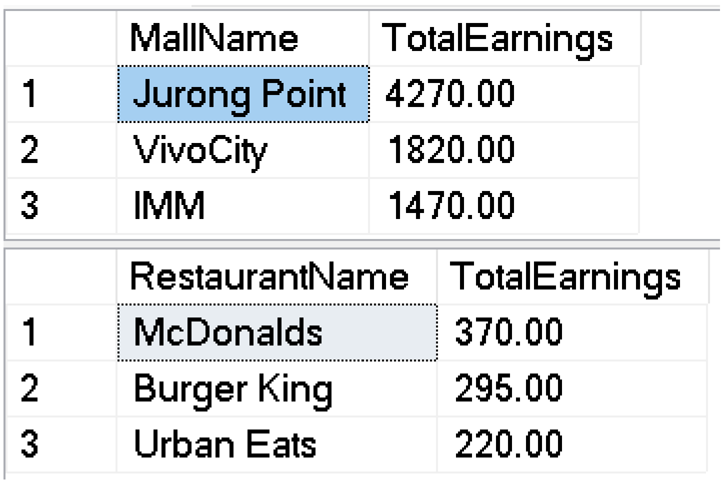
```sql
-- Top 3 highest earning malls
SELECT TOP 3
    m.Name AS MallName,
    SUM(sv.Amount_spent) AS TotalEarnings
FROM
    SHOP_VISIT sv
    JOIN SHOP s ON sv.SID = s.SID
    JOIN MALL m ON s.MID = m.MID
GROUP BY
    m.Name
ORDER BY
    TotalEarnings DESC;

-- Top 3 highest earning restaurants
SELECT TOP 3
    RestaurantName,
    TotalEarnings
FROM
    (SELECT
        Name AS RestaurantName,
        SUM(Amount_spent) AS TotalEarnings
    FROM
        DINE
        JOIN RESTAURANT_OUTLET ON DINE.OID = RESTAURANT_OUTLET.OID
    WHERE
        RID IS NULL  -- Restaurants without a chain
    GROUP BY
        Name

    UNION

    SELECT
        r.Name AS RestaurantName,
        SUM(d.Amount_spent) AS TotalEarnings
    FROM
        DINE d
        JOIN RESTAURANT_OUTLET ro ON d.OID = ro.OID
        JOIN RESTAURANT_CHAIN r ON ro.RID = r.RID
    GROUP BY
        r.Name) AS subquery
ORDER BY
    TotalEarnings DESC;
```

Output



	MallName	TotalEarnings
1	Jurong Point	4270.00
2	VivoCity	1820.00
3	IMM	1470.00

	RestaurantName	TotalEarnings
1	McDonalds	370.00
2	Burger King	295.00
3	Urban Eats	220.00

Explain on how the output is obtained

For Top 3 Highest Earning Malls:

The query joins the SHOP_VISIT, SHOP, and MALL tables. It calculates the total earnings for each mall by summing the Amount_spent in the SHOP_VISIT table for visits associated with each mall (m.MID). The results are grouped by the mall name (m.Name). The malls are then ordered by their total earnings in descending order (TotalEarnings DESC). The top 3 malls are selected using TOP 3.

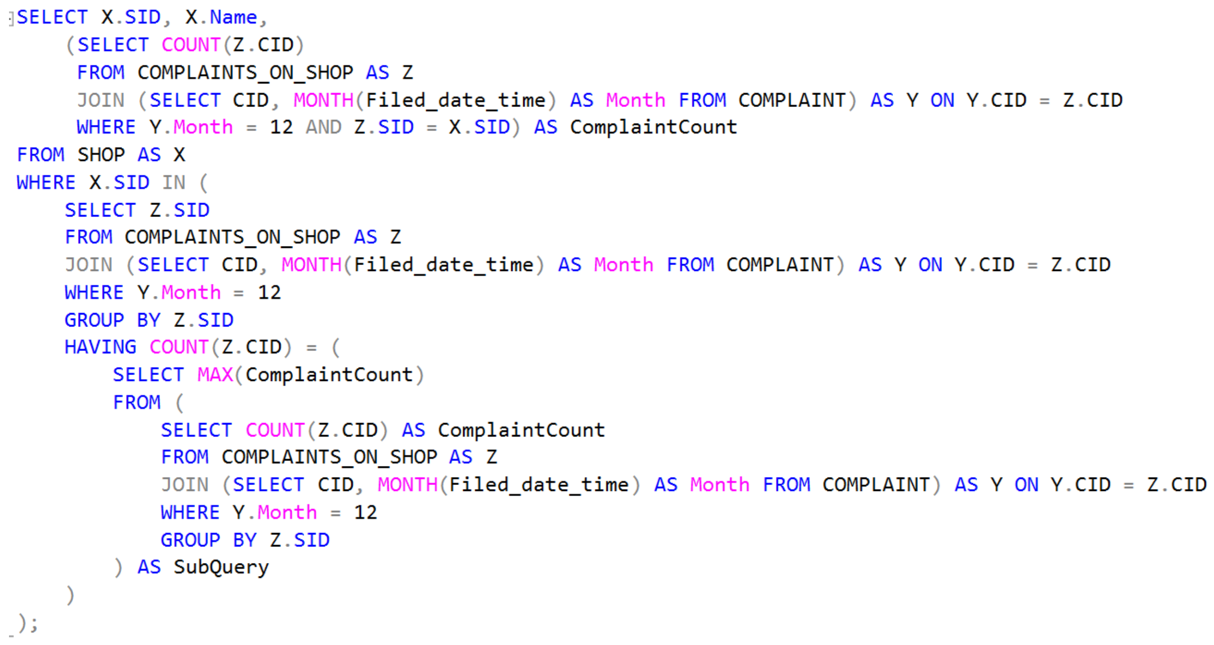
For Top 3 Highest Earning Restaurants:

The query joins the DINE, RESTAURANT_OUTLET, and RESTAURANT_CHAIN tables. It calculates the total earnings for each restaurant by summing the Amount_spent in the DINE table for dining instances associated with each restaurant (ro.Name). The results are grouped by the restaurant name (ro.Name). The restaurants are then ordered by their total earnings in descending order (TotalEarnings DESC). The top 3 restaurants are selected using TOP 3.

Query 7

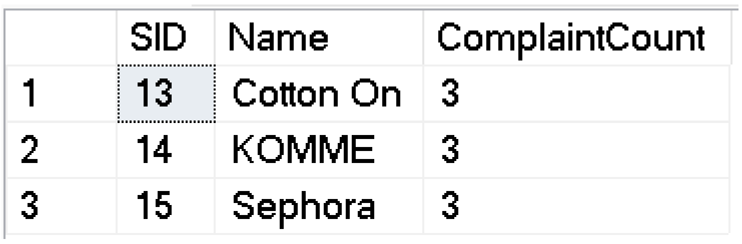
Find shops that received the most complaints in December 2023.

Code



```
SELECT X.SID, X.Name,
    (SELECT COUNT(Z.CID)
     FROM COMPLAINTS_ON_SHOP AS Z
     JOIN (SELECT CID, MONTH(Filed_date_time) AS Month FROM COMPLAINT) AS Y ON Y.CID = Z.CID
     WHERE Y.Month = 12 AND Z.SID = X.SID) AS ComplaintCount
FROM SHOP AS X
WHERE X.SID IN (
    SELECT Z.SID
    FROM COMPLAINTS_ON_SHOP AS Z
    JOIN (SELECT CID, MONTH(Filed_date_time) AS Month FROM COMPLAINT) AS Y ON Y.CID = Z.CID
    WHERE Y.Month = 12
    GROUP BY Z.SID
    HAVING COUNT(Z.CID) = (
        SELECT MAX(ComplaintCount)
        FROM (
            SELECT COUNT(Z.CID) AS ComplaintCount
            FROM COMPLAINTS_ON_SHOP AS Z
            JOIN (SELECT CID, MONTH(Filed_date_time) AS Month FROM COMPLAINT) AS Y ON Y.CID = Z.CID
            WHERE Y.Month = 12
            GROUP BY Z.SID
        ) AS SubQuery
    )
);
```

Output



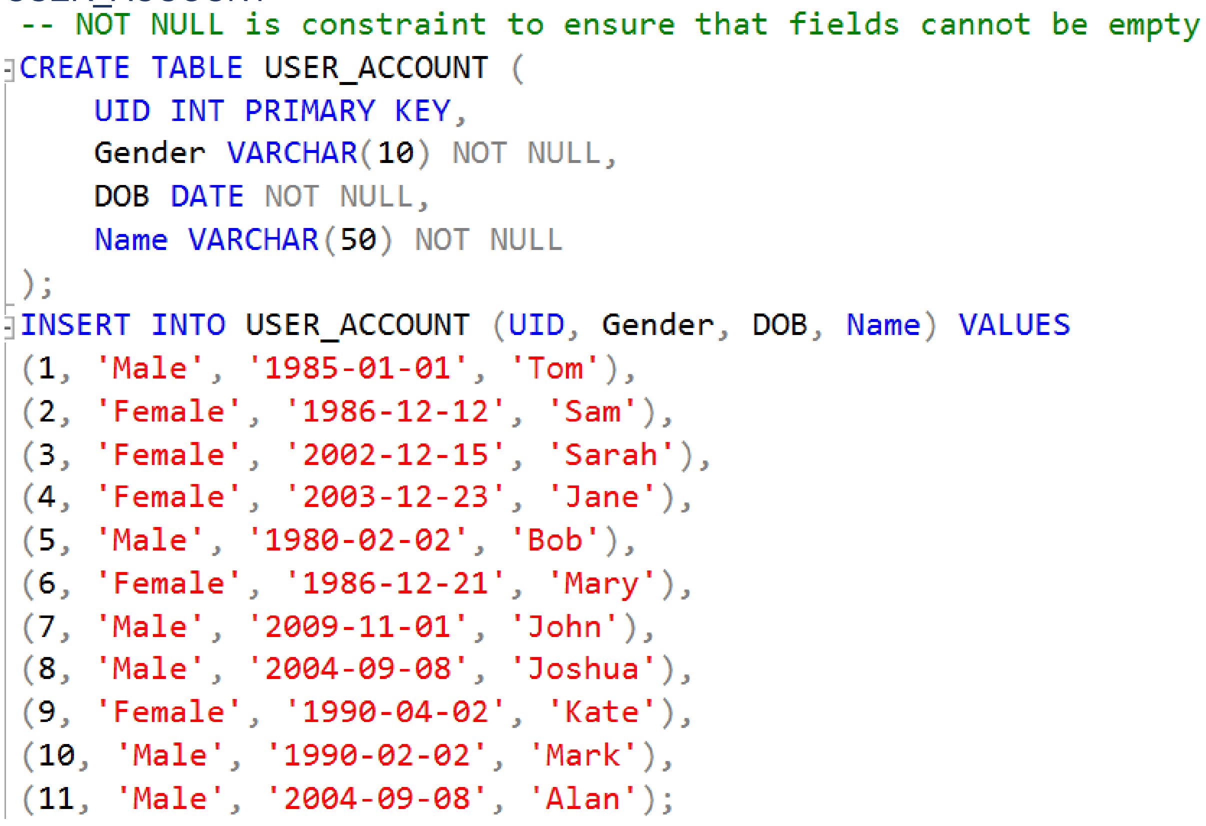
	SID	Name	ComplaintCount
1	13	Cotton On	3
2	14	KOMME	3
3	15	Sephora	3

Explain on how the output is obtained

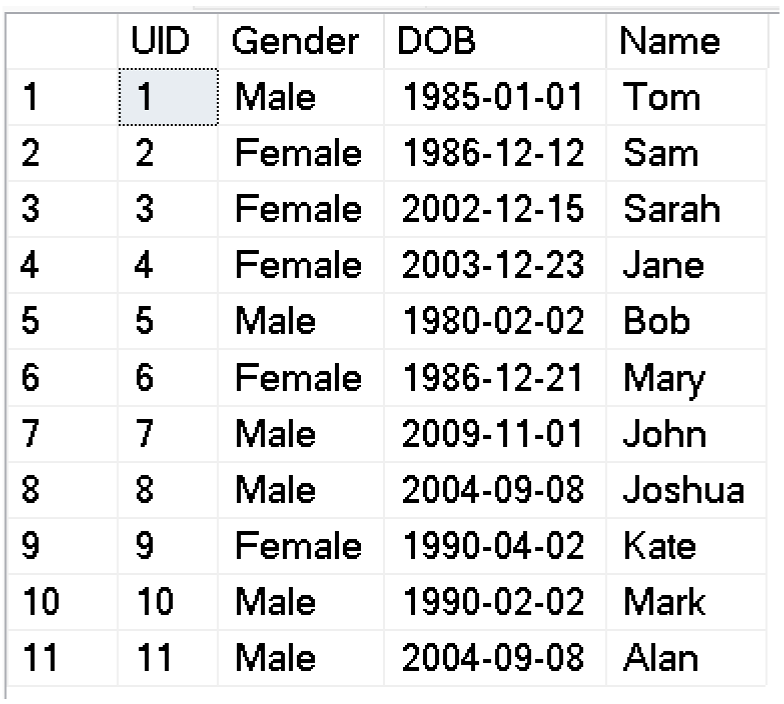
The query joins the SHOP, COMPLAINTS_ON_SHOP, and COMPLAINT tables to calculate the number of complaints for each shop by grouping them by SID. It then identifies the maximum complaint count and uses the HAVING clause to filter shops with this maximum count. In the data, shops with SID 13 (Cotton On), 14 (KOMME), and 15 (Sephora) have the highest number of complaints, which is 13, and the query output correctly reflects this.

SQL DDL commands for table creation & Printout of table records

USER_ACCOUNT

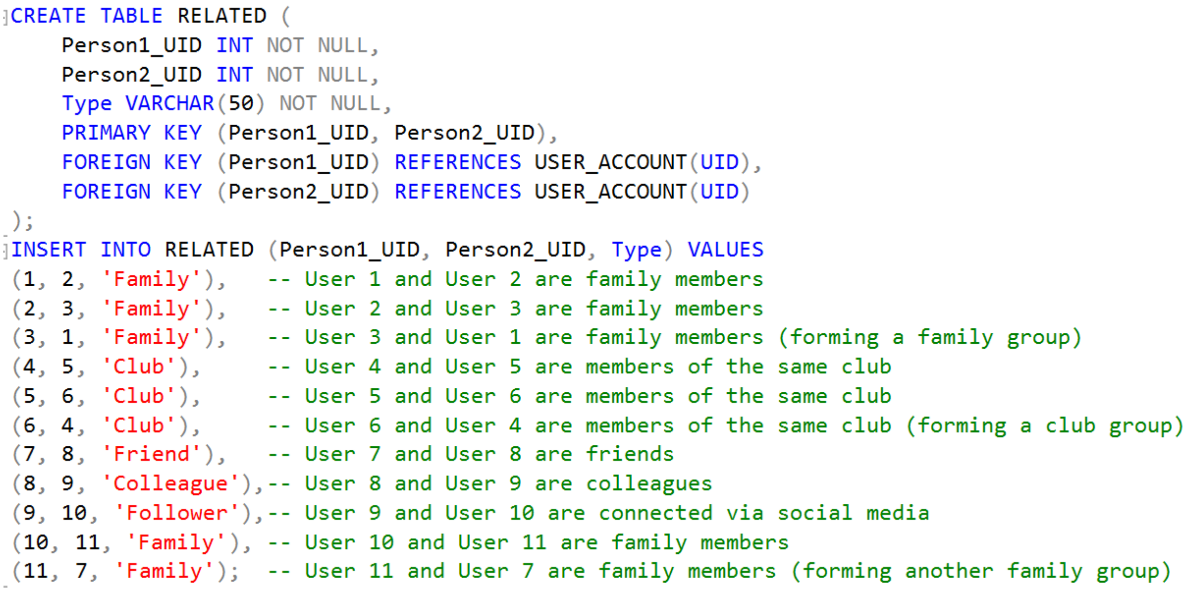


```
-- NOT NULL is constraint to ensure that fields cannot be empty
CREATE TABLE USER_ACCOUNT (
    UID INT PRIMARY KEY,
    Gender VARCHAR(10) NOT NULL,
    DOB DATE NOT NULL,
    Name VARCHAR(50) NOT NULL
);
INSERT INTO USER_ACCOUNT (UID, Gender, DOB, Name) VALUES
(1, 'Male', '1985-01-01', 'Tom'),
(2, 'Female', '1986-12-12', 'Sam'),
(3, 'Female', '2002-12-15', 'Sarah'),
(4, 'Female', '2003-12-23', 'Jane'),
(5, 'Male', '1980-02-02', 'Bob'),
(6, 'Female', '1986-12-21', 'Mary'),
(7, 'Male', '2009-11-01', 'John'),
(8, 'Male', '2004-09-08', 'Joshua'),
(9, 'Female', '1990-04-02', 'Kate'),
(10, 'Male', '1990-02-02', 'Mark'),
(11, 'Male', '2004-09-08', 'Alan');
```

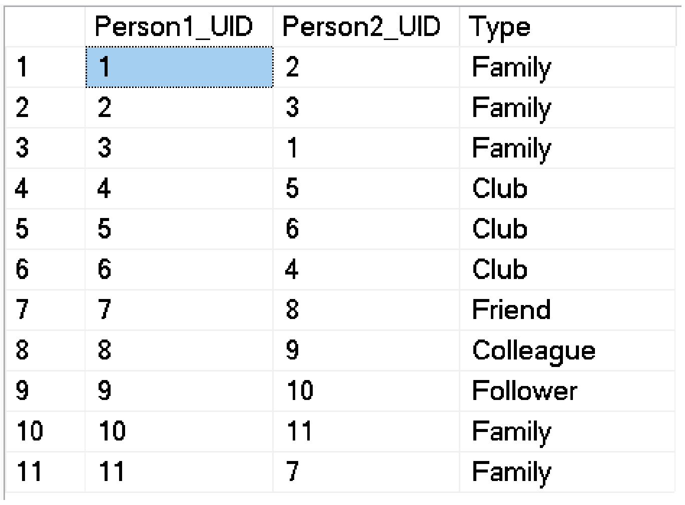


	UID	Gender	DOB	Name
1	1	Male	1985-01-01	Tom
2	2	Female	1986-12-12	Sam
3	3	Female	2002-12-15	Sarah
4	4	Female	2003-12-23	Jane
5	5	Male	1980-02-02	Bob
6	6	Female	1986-12-21	Mary
7	7	Male	2009-11-01	John
8	8	Male	2004-09-08	Joshua
9	9	Female	1990-04-02	Kate
10	10	Male	1990-02-02	Mark
11	11	Male	2004-09-08	Alan

RELATED

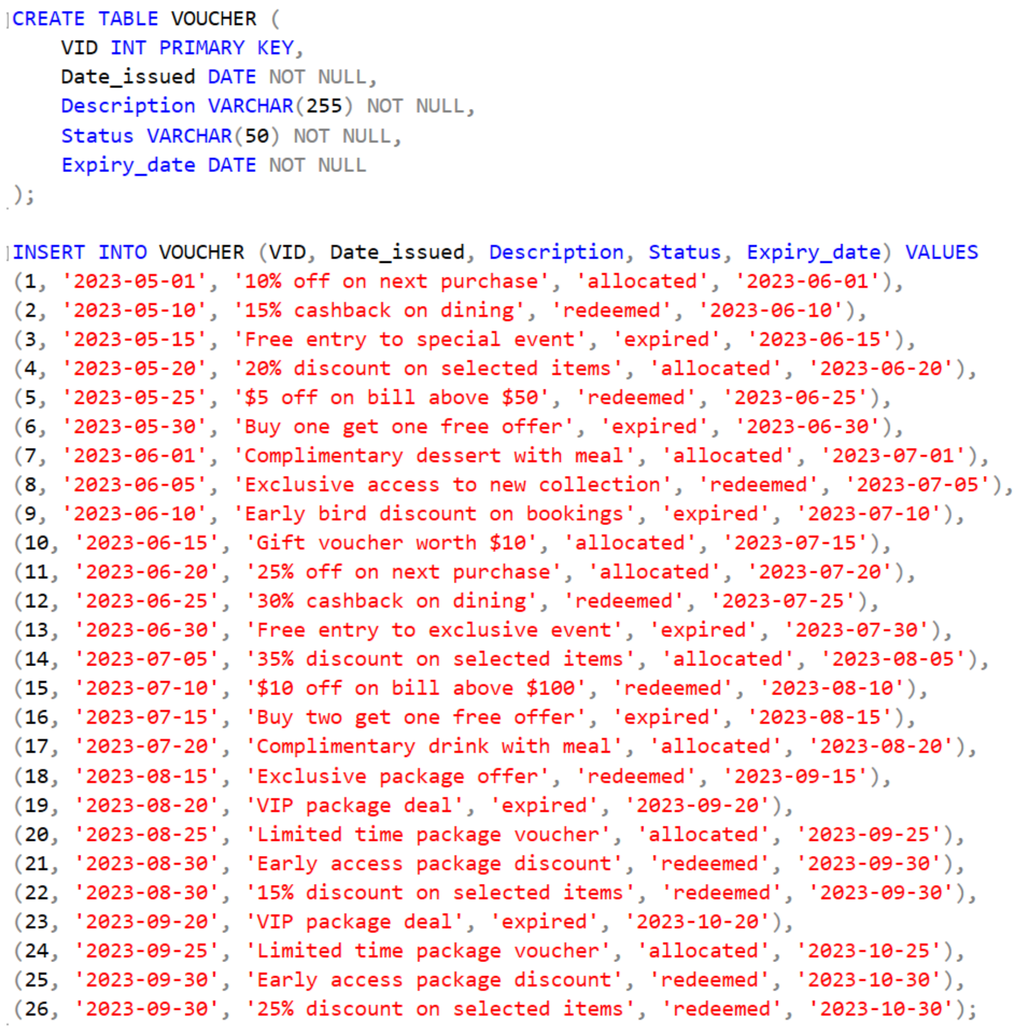


```
CREATE TABLE RELATED (
    Person1_UID INT NOT NULL,
    Person2_UID INT NOT NULL,
    Type VARCHAR(50) NOT NULL,
    PRIMARY KEY (Person1_UID, Person2_UID),
    FOREIGN KEY (Person1_UID) REFERENCES USER_ACCOUNT(UID),
    FOREIGN KEY (Person2_UID) REFERENCES USER_ACCOUNT(UID)
);
INSERT INTO RELATED (Person1_UID, Person2_UID, Type) VALUES
(1, 2, 'Family'),    -- User 1 and User 2 are family members
(2, 3, 'Family'),    -- User 2 and User 3 are family members
(3, 1, 'Family'),    -- User 3 and User 1 are family members (forming a family group)
(4, 5, 'Club'),      -- User 4 and User 5 are members of the same club
(5, 6, 'Club'),      -- User 5 and User 6 are members of the same club
(6, 4, 'Club'),      -- User 6 and User 4 are members of the same club (forming a club group)
(7, 8, 'Friend'),    -- User 7 and User 8 are friends
(8, 9, 'Colleague'),-- User 8 and User 9 are colleagues
(9, 10, 'Follower'),-- User 9 and User 10 are connected via social media
(10, 11, 'Family'), -- User 10 and User 11 are family members
(11, 7, 'Family');  -- User 11 and User 7 are family members (forming another family group)
```



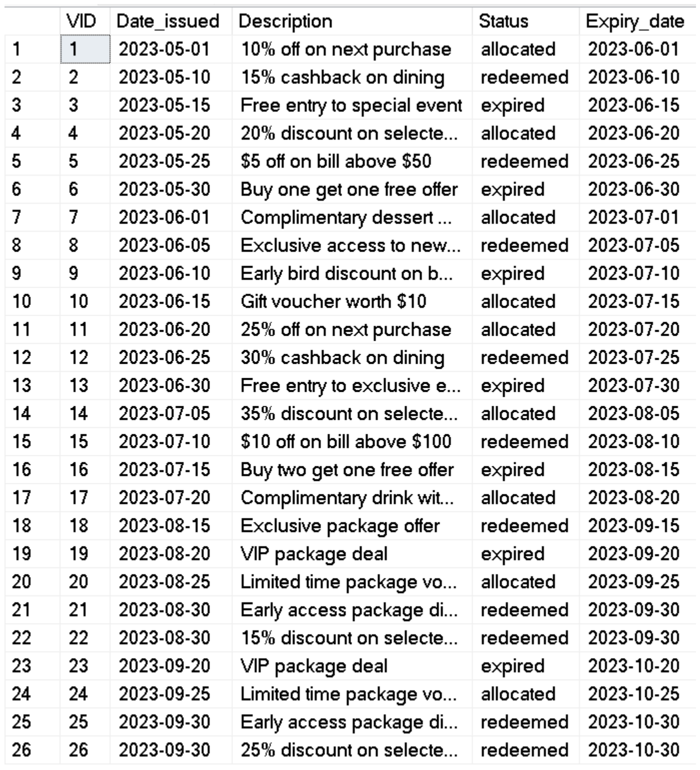
	Person1_UID	Person2_UID	Type
1	1	2	Family
2	2	3	Family
3	3	1	Family
4	4	5	Club
5	5	6	Club
6	6	4	Club
7	7	8	Friend
8	8	9	Colleague
9	9	10	Follower
10	10	11	Family
11	11	7	Family

VOUCHER



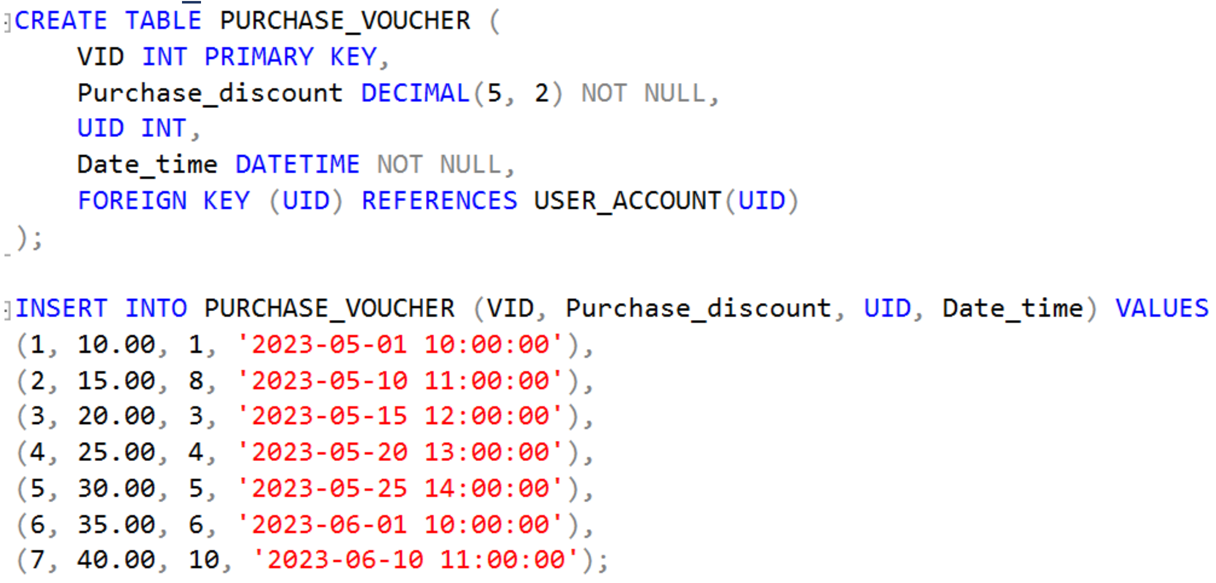
```sql
CREATE TABLE VOUCHER (
    VID INT PRIMARY KEY,
    Date_issued DATE NOT NULL,
    Description VARCHAR(255) NOT NULL,
    Status VARCHAR(50) NOT NULL,
    Expiry_date DATE NOT NULL
);

INSERT INTO VOUCHER (VID, Date_issued, Description, Status, Expiry_date) VALUES
(1, '2023-05-01', '10% off on next purchase', 'allocated', '2023-06-01'),
(2, '2023-05-10', '15% cashback on dining', 'redeemed', '2023-06-10'),
(3, '2023-05-15', 'Free entry to special event', 'expired', '2023-06-15'),
(4, '2023-05-20', '20% discount on selected items', 'allocated', '2023-06-20'),
(5, '2023-05-25', '$5 off on bill above $50', 'redeemed', '2023-06-25'),
(6, '2023-05-30', 'Buy one get one free offer', 'expired', '2023-06-30'),
(7, '2023-06-01', 'Complimentary dessert with meal', 'allocated', '2023-07-01'),
(8, '2023-06-05', 'Exclusive access to new collection', 'redeemed', '2023-07-05'),
(9, '2023-06-10', 'Early bird discount on bookings', 'expired', '2023-07-10'),
(10, '2023-06-15', 'Gift voucher worth $10', 'allocated', '2023-07-15'),
(11, '2023-06-20', '25% off on next purchase', 'allocated', '2023-07-20'),
(12, '2023-06-25', '30% cashback on dining', 'redeemed', '2023-07-25'),
(13, '2023-06-30', 'Free entry to exclusive event', 'expired', '2023-07-30'),
(14, '2023-07-05', '35% discount on selected items', 'allocated', '2023-08-05'),
(15, '2023-07-10', '$10 off on bill above $100', 'redeemed', '2023-08-10'),
(16, '2023-07-15', 'Buy two get one free offer', 'expired', '2023-08-15'),
(17, '2023-07-20', 'Complimentary drink with meal', 'allocated', '2023-08-20'),
(18, '2023-08-15', 'Exclusive package offer', 'redeemed', '2023-09-15'),
(19, '2023-08-20', 'VIP package deal', 'expired', '2023-09-20'),
(20, '2023-08-25', 'Limited time package voucher', 'allocated', '2023-09-25'),
(21, '2023-08-30', 'Early access package discount', 'redeemed', '2023-09-30'),
(22, '2023-08-30', '15% discount on selected items', 'redeemed', '2023-09-30'),
(23, '2023-09-20', 'VIP package deal', 'expired', '2023-10-20'),
(24, '2023-09-25', 'Limited time package voucher', 'allocated', '2023-10-25'),
(25, '2023-09-30', 'Early access package discount', 'redeemed', '2023-10-30'),
(26, '2023-09-30', '25% discount on selected items', 'redeemed', '2023-10-30');
```



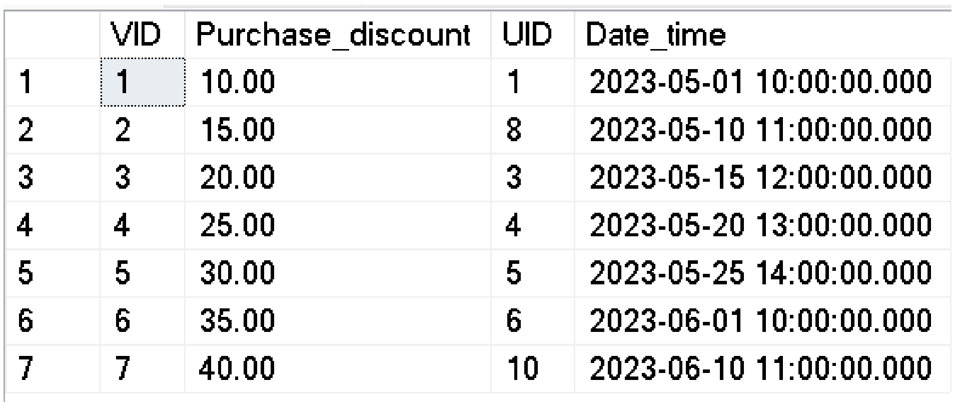
	VID	Date_issued	Description	Status	Expiry_date
1	1	2023-05-01	10% off on next purchase	allocated	2023-06-01
2	2	2023-05-10	15% cashback on dining	redeemed	2023-06-10
3	3	2023-05-15	Free entry to special event	expired	2023-06-15
4	4	2023-05-20	20% discount on selecte...	allocated	2023-06-20
5	5	2023-05-25	$5 off on bill above $50	redeemed	2023-06-25
6	6	2023-05-30	Buy one get one free offer	expired	2023-06-30
7	7	2023-06-01	Complimentary dessert ...	allocated	2023-07-01
8	8	2023-06-05	Exclusive access to new...	redeemed	2023-07-05
9	9	2023-06-10	Early bird discount on b...	expired	2023-07-10
10	10	2023-06-15	Gift voucher worth $10	allocated	2023-07-15
11	11	2023-06-20	25% off on next purchase	allocated	2023-07-20
12	12	2023-06-25	30% cashback on dining	redeemed	2023-07-25
13	13	2023-06-30	Free entry to exclusive e...	expired	2023-07-30
14	14	2023-07-05	35% discount on selecte...	allocated	2023-08-05
15	15	2023-07-10	$10 off on bill above $100	redeemed	2023-08-10
16	16	2023-07-15	Buy two get one free offer	expired	2023-08-15
17	17	2023-07-20	Complimentary drink wit...	allocated	2023-08-20
18	18	2023-08-15	Exclusive package offer	redeemed	2023-09-15
19	19	2023-08-20	VIP package deal	expired	2023-09-20
20	20	2023-08-25	Limited time package vo...	allocated	2023-09-25
21	21	2023-08-30	Early access package di...	redeemed	2023-09-30
22	22	2023-08-30	15% discount on selecte...	redeemed	2023-09-30
23	23	2023-09-20	VIP package deal	expired	2023-10-20
24	24	2023-09-25	Limited time package vo...	allocated	2023-10-25
25	25	2023-09-30	Early access package di...	redeemed	2023-10-30
26	26	2023-09-30	25% discount on selecte...	redeemed	2023-10-30

PURCHASE_VOUCHER



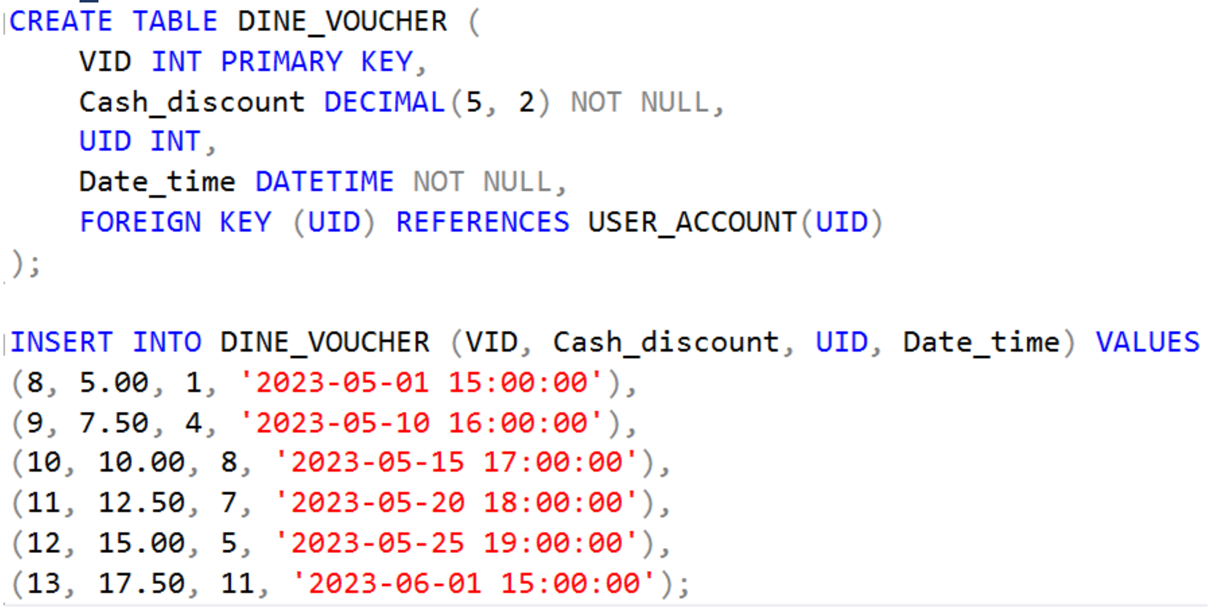
```
CREATE TABLE PURCHASE_VOUCHER (
    VID INT PRIMARY KEY,
    Purchase_discount DECIMAL(5, 2) NOT NULL,
    UID INT,
    Date_time DATETIME NOT NULL,
    FOREIGN KEY (UID) REFERENCES USER_ACCOUNT(UID)
);

INSERT INTO PURCHASE_VOUCHER (VID, Purchase_discount, UID, Date_time) VALUES
(1, 10.00, 1, '2023-05-01 10:00:00'),
(2, 15.00, 8, '2023-05-10 11:00:00'),
(3, 20.00, 3, '2023-05-15 12:00:00'),
(4, 25.00, 4, '2023-05-20 13:00:00'),
(5, 30.00, 5, '2023-05-25 14:00:00'),
(6, 35.00, 6, '2023-06-01 10:00:00'),
(7, 40.00, 10, '2023-06-10 11:00:00');
```



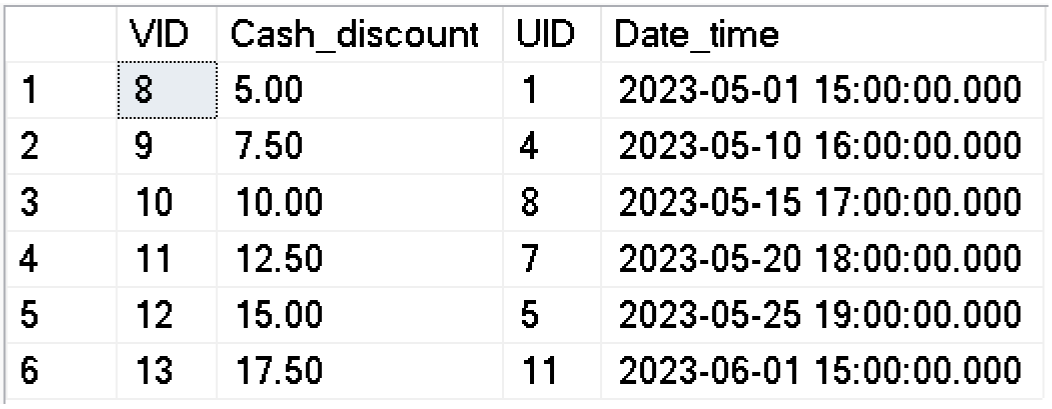
	VID	Purchase_discount	UID	Date_time
1	1	10.00	1	2023-05-01 10:00:00.000
2	2	15.00	8	2023-05-10 11:00:00.000
3	3	20.00	3	2023-05-15 12:00:00.000
4	4	25.00	4	2023-05-20 13:00:00.000
5	5	30.00	5	2023-05-25 14:00:00.000
6	6	35.00	6	2023-06-01 10:00:00.000
7	7	40.00	10	2023-06-10 11:00:00.000

DINE_VOUCHER



```
CREATE TABLE DINE_VOUCHER (
    VID INT PRIMARY KEY,
    Cash_discount DECIMAL(5, 2) NOT NULL,
    UID INT,
    Date_time DATETIME NOT NULL,
    FOREIGN KEY (UID) REFERENCES USER_ACCOUNT(UID)
);

INSERT INTO DINE_VOUCHER (VID, Cash_discount, UID, Date_time) VALUES
(8, 5.00, 1, '2023-05-01 15:00:00'),
(9, 7.50, 4, '2023-05-10 16:00:00'),
(10, 10.00, 8, '2023-05-15 17:00:00'),
(11, 12.50, 7, '2023-05-20 18:00:00'),
(12, 15.00, 5, '2023-05-25 19:00:00'),
(13, 17.50, 11, '2023-06-01 15:00:00');
```

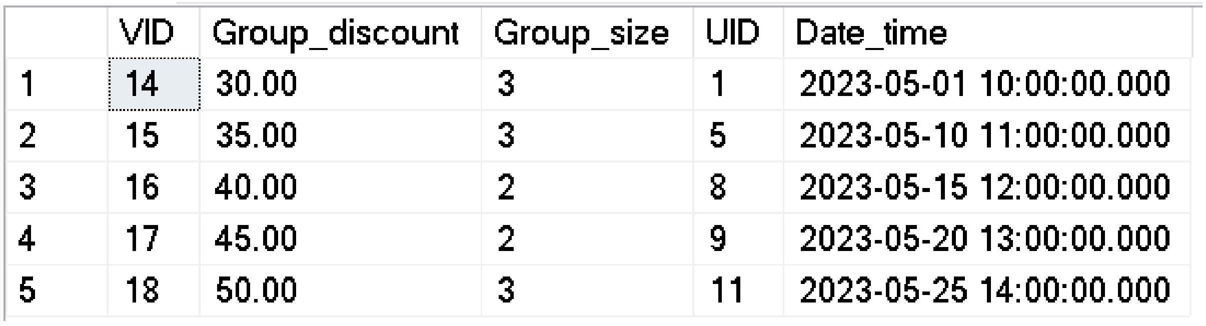


	VID	Cash_discount	UID	Date_time
1	8	5.00	1	2023-05-01 15:00:00.000
2	9	7.50	4	2023-05-10 16:00:00.000
3	10	10.00	8	2023-05-15 17:00:00.000
4	11	12.50	7	2023-05-20 18:00:00.000
5	12	15.00	5	2023-05-25 19:00:00.000
6	13	17.50	11	2023-06-01 15:00:00.000

GROUP_VOUCHER

```
CREATE TABLE GROUP_VOUCHER (
    VID INT PRIMARY KEY,
    Group_discount DECIMAL(5, 2) NOT NULL,
    Group_size INT NOT NULL,
    UID INT NOT NULL,
    Date_time DATETIME NOT NULL,
    FOREIGN KEY (UID) REFERENCES USER_ACCOUNT(UID)
);

INSERT INTO GROUP_VOUCHER (VID, Group_discount, Group_size, UID, Date_time) VALUES
(14, 30.00, 3, 1, '2023-05-01 10:00:00'),
(15, 35.00, 3, 5, '2023-05-10 11:00:00'),
(16, 40.00, 2, 8, '2023-05-15 12:00:00'),
(17, 45.00, 2, 9, '2023-05-20 13:00:00'),
(18, 50.00, 3, 11, '2023-05-25 14:00:00');
```

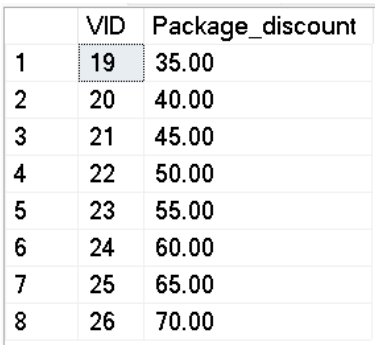


	VID	Group_discount	Group_size	UID	Date_time
1	14	30.00	3	1	2023-05-01 10:00:00.000
2	15	35.00	3	5	2023-05-10 11:00:00.000
3	16	40.00	2	8	2023-05-15 12:00:00.000
4	17	45.00	2	9	2023-05-20 13:00:00.000
5	18	50.00	3	11	2023-05-25 14:00:00.000

PACKAGE_VOUCHER

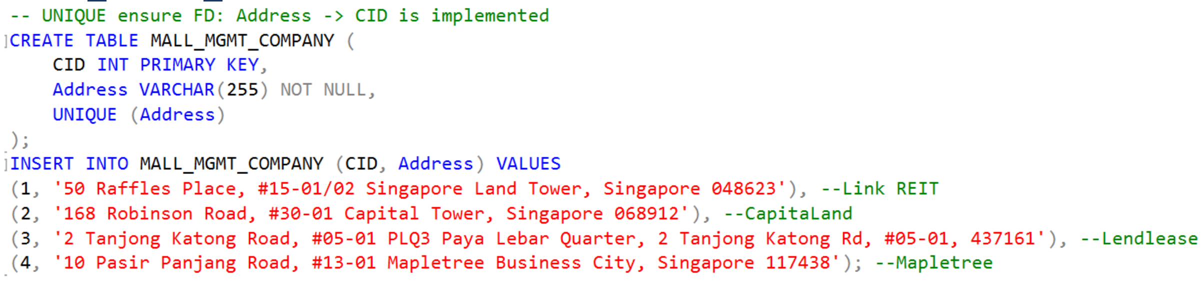
```
CREATE TABLE PACKAGE_VOUCHER (
    VID INT PRIMARY KEY,
    Package_discount DECIMAL(5, 2) NOT NULL
);

INSERT INTO PACKAGE_VOUCHER (VID, Package_discount) VALUES
(19, 35.00),
(20, 40.00),
(21, 45.00),
(22, 50.00),
(23, 55.00),
(24, 60.00),
(25, 65.00),
(26, 70.00);
```

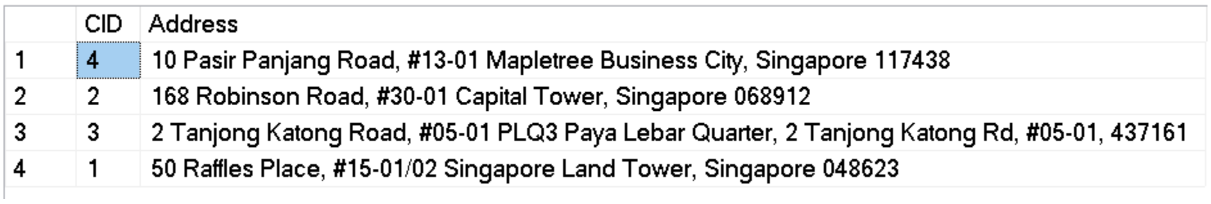


	VID	Package_discount
1	19	35.00
2	20	40.00
3	21	45.00
4	22	50.00
5	23	55.00
6	24	60.00
7	25	65.00
8	26	70.00

MALL_MGMT_COMPANY

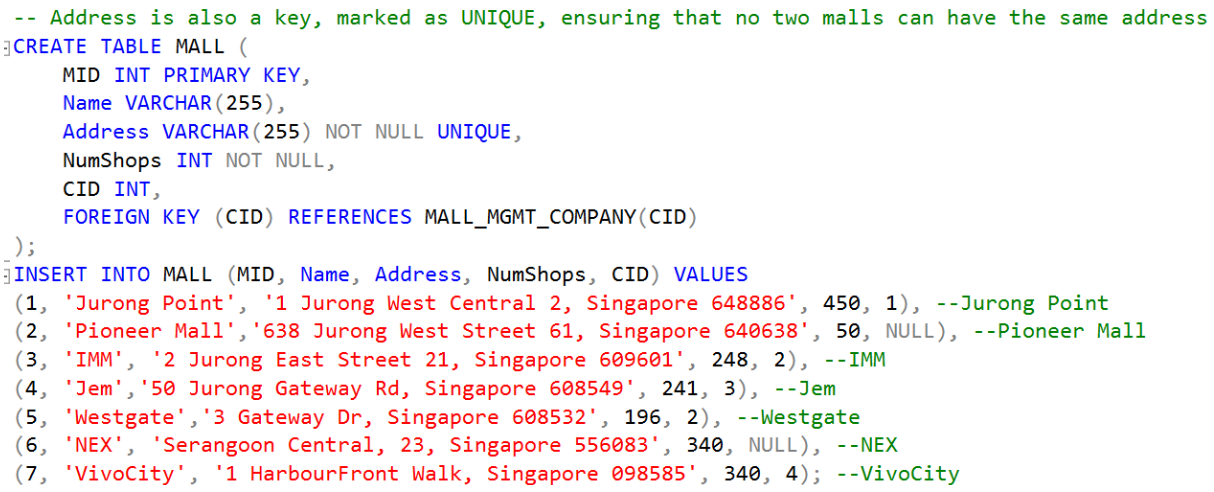


```sql
-- UNIQUE ensure FD: Address -> CID is implemented
CREATE TABLE MALL_MGMT_COMPANY (
    CID INT PRIMARY KEY,
    Address VARCHAR(255) NOT NULL,
    UNIQUE (Address)
);
INSERT INTO MALL_MGMT_COMPANY (CID, Address) VALUES
(1, '50 Raffles Place, #15-01/02 Singapore Land Tower, Singapore 048623'), --Link REIT
(2, '168 Robinson Road, #30-01 Capital Tower, Singapore 068912'), --CapitaLand
(3, '2 Tanjong Katong Road, #05-01 PLQ3 Paya Lebar Quarter, 2 Tanjong Katong Rd, #05-01, 437161'), --Lendlease
(4, '10 Pasir Panjang Road, #13-01 Mapletree Business City, Singapore 117438'); --Mapletree
```

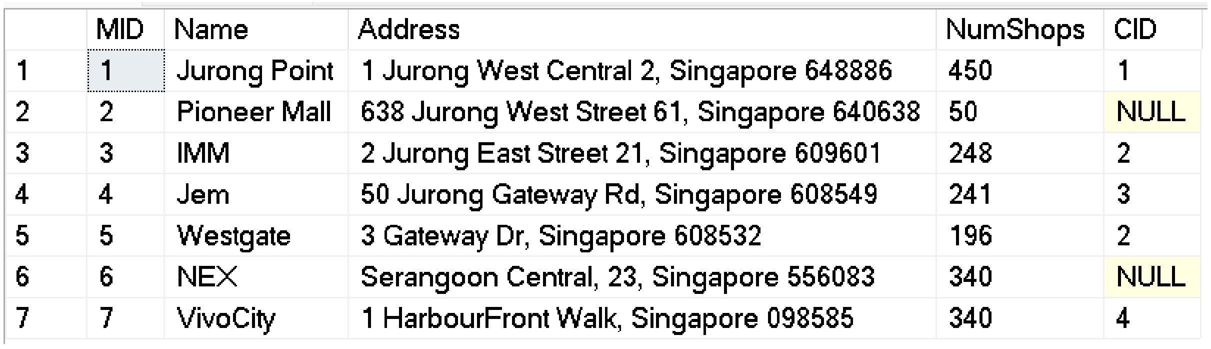


	CID	Address
1	4	10 Pasir Panjang Road, #13-01 Mapletree Business City, Singapore 117438
2	2	168 Robinson Road, #30-01 Capital Tower, Singapore 068912
3	3	2 Tanjong Katong Road, #05-01 PLQ3 Paya Lebar Quarter, 2 Tanjong Katong Rd, #05-01, 437161
4	1	50 Raffles Place, #15-01/02 Singapore Land Tower, Singapore 048623

MALL

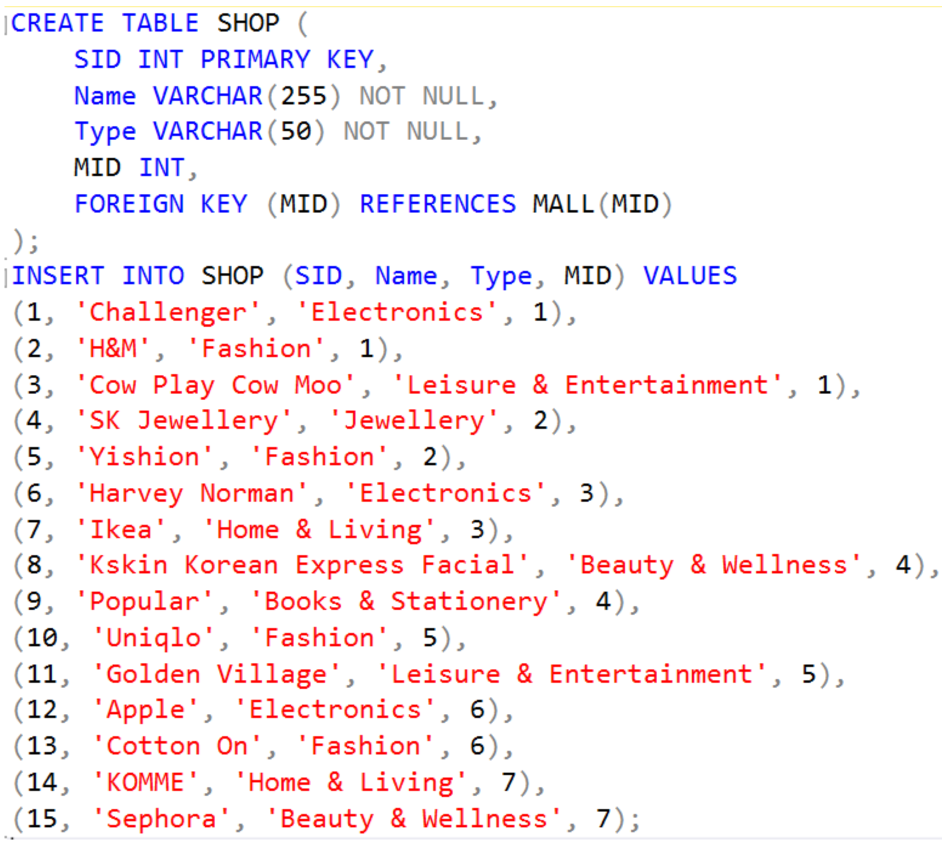


```sql
 -- Address is also a key, marked as UNIQUE, ensuring that no two malls can have the same address
CREATE TABLE MALL (
    MID INT PRIMARY KEY,
    Name VARCHAR(255),
    Address VARCHAR(255) NOT NULL UNIQUE,
    NumShops INT NOT NULL,
    CID INT,
    FOREIGN KEY (CID) REFERENCES MALL_MGMT_COMPANY(CID)
);
INSERT INTO MALL (MID, Name, Address, NumShops, CID) VALUES
(1, 'Jurong Point', '1 Jurong West Central 2, Singapore 648886', 450, 1), --Jurong Point
(2, 'Pioneer Mall','638 Jurong West Street 61, Singapore 640638', 50, NULL), --Pioneer Mall
(3, 'IMM', '2 Jurong East Street 21, Singapore 609601', 248, 2), --IMM
(4, 'Jem','50 Jurong Gateway Rd, Singapore 608549', 241, 3), --Jem
(5, 'Westgate','3 Gateway Dr, Singapore 608532', 196, 2), --Westgate
(6, 'NEX', 'Serangoon Central, 23, Singapore 556083', 340, NULL), --NEX
(7, 'VivoCity', '1 HarbourFront Walk, Singapore 098585', 340, 4); --VivoCity
```

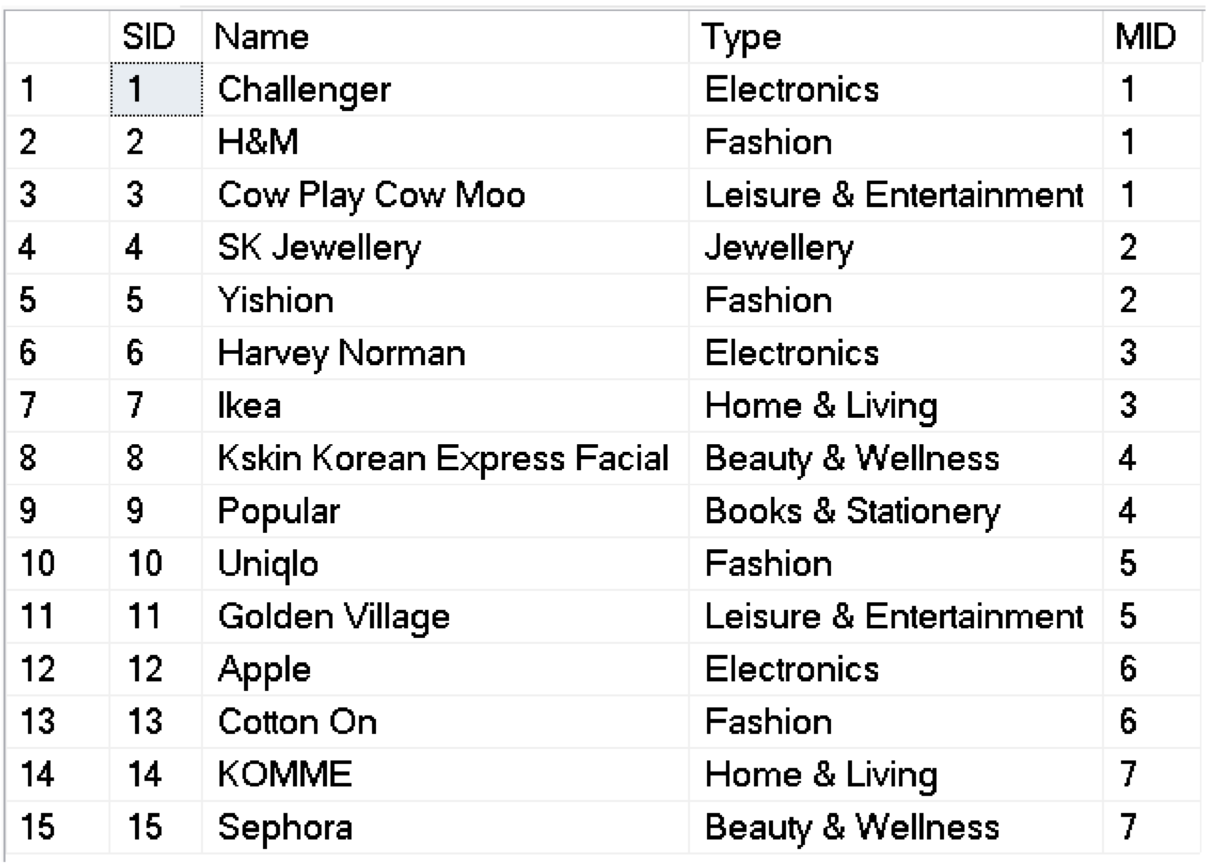


	MID	Name	Address	NumShops	CID
1	1	Jurong Point	1 Jurong West Central 2, Singapore 648886	450	1
2	2	Pioneer Mall	638 Jurong West Street 61, Singapore 640638	50	NULL
3	3	IMM	2 Jurong East Street 21, Singapore 609601	248	2
4	4	Jem	50 Jurong Gateway Rd, Singapore 608549	241	3
5	5	Westgate	3 Gateway Dr, Singapore 608532	196	2
6	6	NEX	Serangoon Central, 23, Singapore 556083	340	NULL
7	7	VivoCity	1 HarbourFront Walk, Singapore 098585	340	4

SHOP



```
CREATE TABLE SHOP (
    SID INT PRIMARY KEY,
    Name VARCHAR(255) NOT NULL,
    Type VARCHAR(50) NOT NULL,
    MID INT,
    FOREIGN KEY (MID) REFERENCES MALL(MID)
);
INSERT INTO SHOP (SID, Name, Type, MID) VALUES
(1, 'Challenger', 'Electronics', 1),
(2, 'H&M', 'Fashion', 1),
(3, 'Cow Play Cow Moo', 'Leisure & Entertainment', 1),
(4, 'SK Jewellery', 'Jewellery', 2),
(5, 'Yishion', 'Fashion', 2),
(6, 'Harvey Norman', 'Electronics', 3),
(7, 'Ikea', 'Home & Living', 3),
(8, 'Kskin Korean Express Facial', 'Beauty & Wellness', 4),
(9, 'Popular', 'Books & Stationery', 4),
(10, 'Uniqlo', 'Fashion', 5),
(11, 'Golden Village', 'Leisure & Entertainment', 5),
(12, 'Apple', 'Electronics', 6),
(13, 'Cotton On', 'Fashion', 6),
(14, 'KOMME', 'Home & Living', 7),
(15, 'Sephora', 'Beauty & Wellness', 7);
```

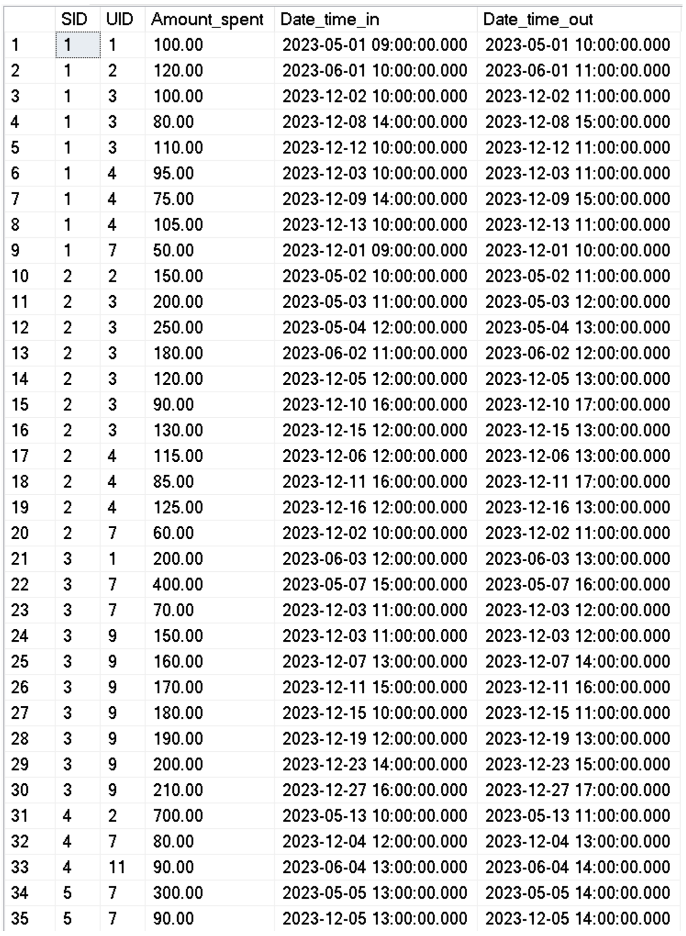
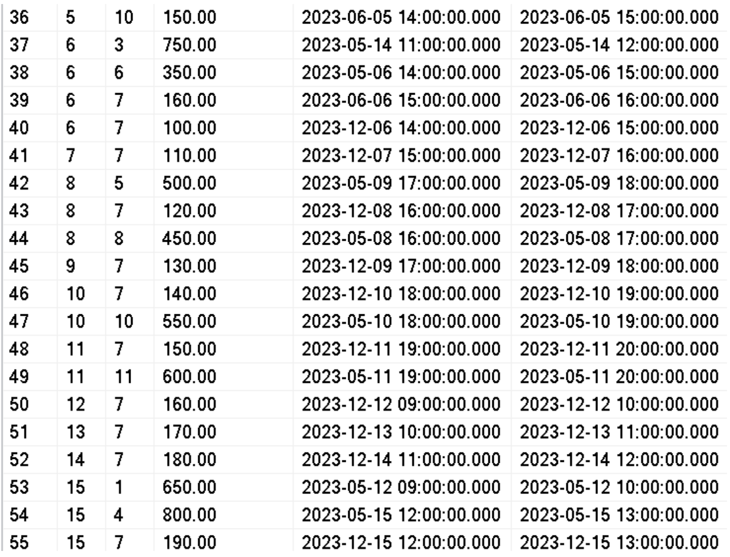


	SID	Name	Type	MID
1	1	Challenger	Electronics	1
2	2	H&M	Fashion	1
3	3	Cow Play Cow Moo	Leisure & Entertainment	1
4	4	SK Jewellery	Jewellery	2
5	5	Yishion	Fashion	2
6	6	Harvey Norman	Electronics	3
7	7	Ikea	Home & Living	3
8	8	Kskin Korean Express Facial	Beauty & Wellness	4
9	9	Popular	Books & Stationery	4
10	10	Uniqlo	Fashion	5
11	11	Golden Village	Leisure & Entertainment	5
12	12	Apple	Electronics	6
13	13	Cotton On	Fashion	6
14	14	KOMME	Home & Living	7
15	15	Sephora	Beauty & Wellness	7

SHOP_VISIT

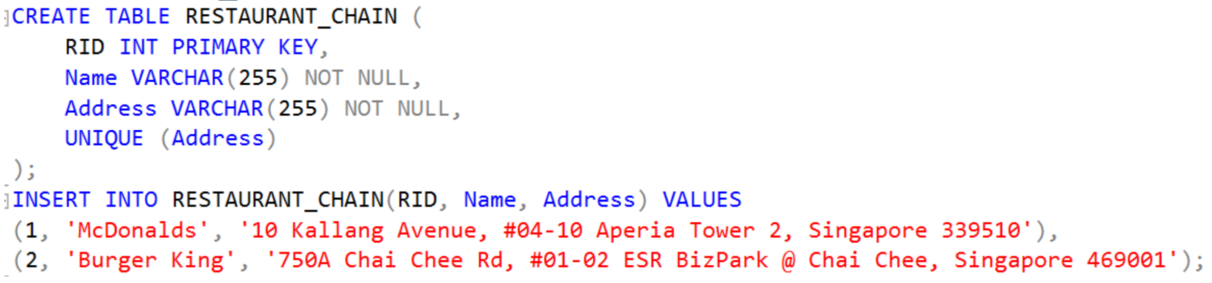
```
CREATE TABLE SHOP_VISIT (
    SID INT NOT NULL,
    UID INT NOT NULL,
    Amount_spent DECIMAL(10, 2),
    Date_time_in DATETIME NOT NULL,
    Date_time_out DATETIME NOT NULL,
    PRIMARY KEY (SID, UID, Date_time_in),
    FOREIGN KEY (SID) REFERENCES SHOP(SID),
    FOREIGN KEY (UID) REFERENCES USER_ACCOUNT(UID)
);
INSERT INTO SHOP_VISIT (SID, UID, Amount_spent, Date_time_in, Date_time_out) VALUES
(1, 1, 100.00, '2023-05-01 09:00:00', '2023-05-01 10:00:00'),
(2, 2, 150.00, '2023-05-02 10:00:00', '2023-05-02 11:00:00'),
(2, 3, 200.00, '2023-05-03 11:00:00', '2023-05-03 12:00:00'),
(2, 3, 250.00, '2023-05-04 12:00:00', '2023-05-04 13:00:00'),
(5, 7, 300.00, '2023-05-05 13:00:00', '2023-05-05 14:00:00'),
(6, 6, 350.00, '2023-05-06 14:00:00', '2023-05-06 15:00:00'),
(3, 7, 400.00, '2023-05-07 15:00:00', '2023-05-07 16:00:00'),
(8, 8, 450.00, '2023-05-08 16:00:00', '2023-05-08 17:00:00'),
(8, 5, 500.00, '2023-05-09 17:00:00', '2023-05-09 18:00:00'),
(10, 10, 550.00, '2023-05-10 18:00:00', '2023-05-10 19:00:00'),
(11, 11, 600.00, '2023-05-11 19:00:00', '2023-05-11 20:00:00'),
(15, 1, 650.00, '2023-05-12 09:00:00', '2023-05-12 10:00:00'),
(4, 2, 700.00, '2023-05-13 10:00:00', '2023-05-13 11:00:00'),
(6, 3, 750.00, '2023-05-14 11:00:00', '2023-05-14 12:00:00'),
(15, 4, 800.00, '2023-05-15 12:00:00', '2023-05-15 13:00:00'),
(1, 7, 50.00, '2023-12-01 09:00:00', '2023-12-01 10:00:00'),
(2, 7, 60.00, '2023-12-02 10:00:00', '2023-12-02 11:00:00'),
(3, 7, 70.00, '2023-12-03 11:00:00', '2023-12-03 12:00:00'),
(4, 7, 80.00, '2023-12-04 12:00:00', '2023-12-04 13:00:00'),
(5, 7, 90.00, '2023-12-05 13:00:00', '2023-12-05 14:00:00'),
(6, 7, 100.00, '2023-12-06 14:00:00', '2023-12-06 15:00:00'),
(7, 7, 110.00, '2023-12-07 15:00:00', '2023-12-07 16:00:00'),
(8, 7, 120.00, '2023-12-08 16:00:00', '2023-12-08 17:00:00'),
```

```
(9, 7, 130.00, '2023-12-09 17:00:00', '2023-12-09 18:00:00'),
(10, 7, 140.00, '2023-12-10 18:00:00', '2023-12-10 19:00:00'),
(11, 7, 150.00, '2023-12-11 19:00:00', '2023-12-11 20:00:00'),
(12, 7, 160.00, '2023-12-12 09:00:00', '2023-12-12 10:00:00'),
(13, 7, 170.00, '2023-12-13 10:00:00', '2023-12-13 11:00:00'),
(14, 7, 180.00, '2023-12-14 11:00:00', '2023-12-14 12:00:00'),
(15, 7, 190.00, '2023-12-15 12:00:00', '2023-12-15 13:00:00'),
(1, 2, 120.00, '2023-06-01 10:00:00', '2023-06-01 11:00:00'),
(2, 3, 180.00, '2023-06-02 11:00:00', '2023-06-02 12:00:00'),
(3, 1, 200.00, '2023-06-03 12:00:00', '2023-06-03 13:00:00'),
(4, 11, 90.00, '2023-06-04 13:00:00', '2023-06-04 14:00:00'),
(5, 10, 150.00, '2023-06-05 14:00:00', '2023-06-05 15:00:00'),
(6, 7, 160.00, '2023-06-06 15:00:00', '2023-06-06 16:00:00'),
(1, 3, 100.00, '2023-12-02 10:00:00', '2023-12-02 11:00:00'),
(2, 3, 120.00, '2023-12-05 12:00:00', '2023-12-05 13:00:00'),
(1, 3, 80.00,  '2023-12-08 14:00:00', '2023-12-08 15:00:00'),
(2, 3, 90.00,  '2023-12-10 16:00:00', '2023-12-10 17:00:00'),
(1, 3, 110.00, '2023-12-12 10:00:00', '2023-12-12 11:00:00'),
(2, 3, 130.00, '2023-12-15 12:00:00', '2023-12-15 13:00:00'),
(1, 4, 95.00,  '2023-12-03 10:00:00', '2023-12-03 11:00:00'),
(2, 4, 115.00, '2023-12-06 12:00:00', '2023-12-06 13:00:00'),
(1, 4, 75.00,  '2023-12-09 14:00:00', '2023-12-09 15:00:00'),
(2, 4, 85.00,  '2023-12-11 16:00:00', '2023-12-11 17:00:00'),
(1, 4, 105.00, '2023-12-13 10:00:00', '2023-12-13 11:00:00'),
(2, 4, 125.00, '2023-12-16 12:00:00', '2023-12-16 13:00:00'),
(3, 9, 150.00, '2023-12-03 11:00:00', '2023-12-03 12:00:00'),
(3, 9, 160.00, '2023-12-07 13:00:00', '2023-12-07 14:00:00'),
(3, 9, 170.00, '2023-12-11 15:00:00', '2023-12-11 16:00:00'),
(3, 9, 180.00, '2023-12-15 10:00:00', '2023-12-15 11:00:00'),
(3, 9, 190.00, '2023-12-19 12:00:00', '2023-12-19 13:00:00'),
(3, 9, 200.00, '2023-12-23 14:00:00', '2023-12-23 15:00:00'),
(3, 9, 210.00, '2023-12-27 16:00:00', '2023-12-27 17:00:00');
```

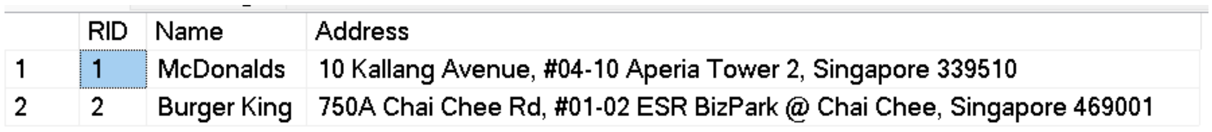



	SID	UID	Amount_spent	Date_time_in	Date_time_out
1	1	1	100.00	2023-05-01 09:00:00.000	2023-05-01 10:00:00.000
2	1	2	120.00	2023-06-01 10:00:00.000	2023-06-01 11:00:00.000
3	1	3	100.00	2023-12-02 10:00:00.000	2023-12-02 11:00:00.000
4	1	3	80.00	2023-12-08 14:00:00.000	2023-12-08 15:00:00.000
5	1	3	110.00	2023-12-12 10:00:00.000	2023-12-12 11:00:00.000
6	1	4	95.00	2023-12-03 10:00:00.000	2023-12-03 11:00:00.000
7	1	4	75.00	2023-12-09 14:00:00.000	2023-12-09 15:00:00.000
8	1	4	105.00	2023-12-13 10:00:00.000	2023-12-13 11:00:00.000
9	1	7	50.00	2023-12-01 09:00:00.000	2023-12-01 10:00:00.000
10	2	2	150.00	2023-05-02 10:00:00.000	2023-05-02 11:00:00.000
11	2	3	200.00	2023-05-03 11:00:00.000	2023-05-03 12:00:00.000
12	2	3	250.00	2023-05-04 12:00:00.000	2023-05-04 13:00:00.000
13	2	3	180.00	2023-06-02 11:00:00.000	2023-06-02 12:00:00.000
14	2	3	120.00	2023-12-05 12:00:00.000	2023-12-05 13:00:00.000
15	2	3	90.00	2023-12-10 16:00:00.000	2023-12-10 17:00:00.000
16	2	3	130.00	2023-12-15 12:00:00.000	2023-12-15 13:00:00.000
17	2	4	115.00	2023-12-06 12:00:00.000	2023-12-06 13:00:00.000
18	2	4	85.00	2023-12-11 16:00:00.000	2023-12-11 17:00:00.000
19	2	4	125.00	2023-12-16 12:00:00.000	2023-12-16 13:00:00.000
20	2	7	60.00	2023-12-02 10:00:00.000	2023-12-02 11:00:00.000
21	3	1	200.00	2023-06-03 12:00:00.000	2023-06-03 13:00:00.000
22	3	7	400.00	2023-05-07 15:00:00.000	2023-05-07 16:00:00.000
23	3	7	70.00	2023-12-03 11:00:00.000	2023-12-03 12:00:00.000
24	3	9	150.00	2023-12-03 11:00:00.000	2023-12-03 12:00:00.000
25	3	9	160.00	2023-12-07 13:00:00.000	2023-12-07 14:00:00.000
26	3	9	170.00	2023-12-11 15:00:00.000	2023-12-11 16:00:00.000
27	3	9	180.00	2023-12-15 10:00:00.000	2023-12-15 11:00:00.000
28	3	9	190.00	2023-12-19 12:00:00.000	2023-12-19 13:00:00.000
29	3	9	200.00	2023-12-23 14:00:00.000	2023-12-23 15:00:00.000
30	3	9	210.00	2023-12-27 16:00:00.000	2023-12-27 17:00:00.000
31	4	2	700.00	2023-05-13 10:00:00.000	2023-05-13 11:00:00.000
32	4	7	80.00	2023-12-04 12:00:00.000	2023-12-04 13:00:00.000
33	4	11	90.00	2023-06-04 13:00:00.000	2023-06-04 14:00:00.000
34	5	7	300.00	2023-05-05 13:00:00.000	2023-05-05 14:00:00.000
35	5	7	90.00	2023-12-05 13:00:00.000	2023-12-05 14:00:00.000
36	5	10	150.00	2023-06-05 14:00:00.000	2023-06-05 15:00:00.000
37	6	3	750.00	2023-05-14 11:00:00.000	2023-05-14 12:00:00.000
38	6	6	350.00	2023-05-06 14:00:00.000	2023-05-06 15:00:00.000
39	6	7	160.00	2023-06-06 15:00:00.000	2023-06-06 16:00:00.000
40	6	7	100.00	2023-12-06 14:00:00.000	2023-12-06 15:00:00.000
41	7	7	110.00	2023-12-07 15:00:00.000	2023-12-07 16:00:00.000
42	8	5	500.00	2023-05-09 17:00:00.000	2023-05-09 18:00:00.000
43	8	7	120.00	2023-12-08 16:00:00.000	2023-12-08 17:00:00.000
44	8	8	450.00	2023-05-08 16:00:00.000	2023-05-08 17:00:00.000
45	9	7	130.00	2023-12-09 17:00:00.000	2023-12-09 18:00:00.000
46	10	7	140.00	2023-12-10 18:00:00.000	2023-12-10 19:00:00.000
47	10	10	550.00	2023-05-10 18:00:00.000	2023-05-10 19:00:00.000
48	11	7	150.00	2023-12-11 19:00:00.000	2023-12-11 20:00:00.000
49	11	11	600.00	2023-05-11 19:00:00.000	2023-05-11 20:00:00.000
50	12	7	160.00	2023-12-12 09:00:00.000	2023-12-12 10:00:00.000
51	13	7	170.00	2023-12-13 10:00:00.000	2023-12-13 11:00:00.000
52	14	7	180.00	2023-12-14 11:00:00.000	2023-12-14 12:00:00.000
53	15	1	650.00	2023-05-12 09:00:00.000	2023-05-12 10:00:00.000
54	15	4	800.00	2023-05-15 12:00:00.000	2023-05-15 13:00:00.000
55	15	7	190.00	2023-12-15 12:00:00.000	2023-12-15 13:00:00.000

RESTAURANT_CHAIN

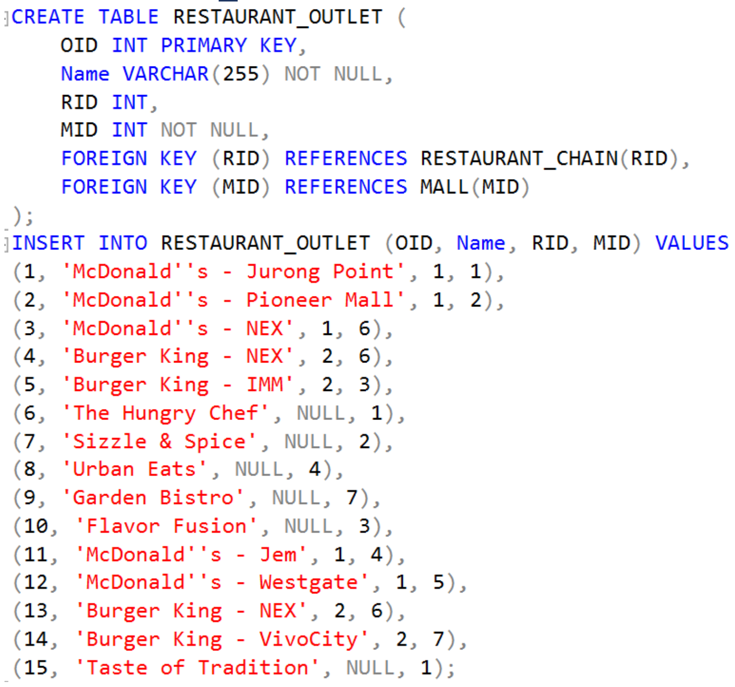


```
CREATE TABLE RESTAURANT_CHAIN (
    RID INT PRIMARY KEY,
    Name VARCHAR(255) NOT NULL,
    Address VARCHAR(255) NOT NULL,
    UNIQUE (Address)
);
INSERT INTO RESTAURANT_CHAIN(RID, Name, Address) VALUES
(1, 'McDonalds', '10 Kallang Avenue, #04-10 Aperia Tower 2, Singapore 339510'),
(2, 'Burger King', '750A Chai Chee Rd, #01-02 ESR BizPark @ Chai Chee, Singapore 469001');
```

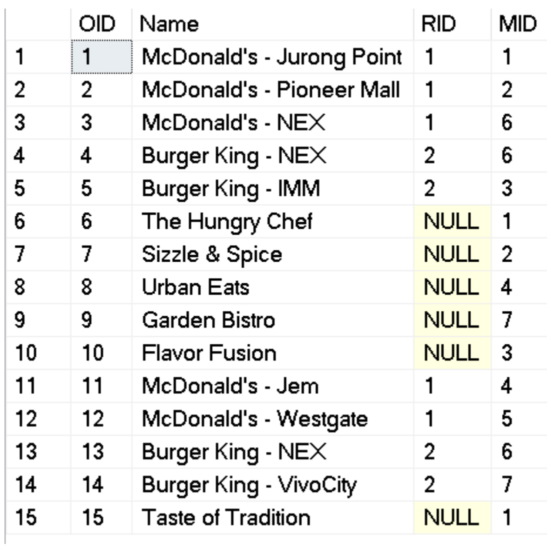


	RID	Name	Address
1	1	McDonalds	10 Kallang Avenue, #04-10 Aperia Tower 2, Singapore 339510
2	2	Burger King	750A Chai Chee Rd, #01-02 ESR BizPark @ Chai Chee, Singapore 469001

RESTAURANT_OUTLET

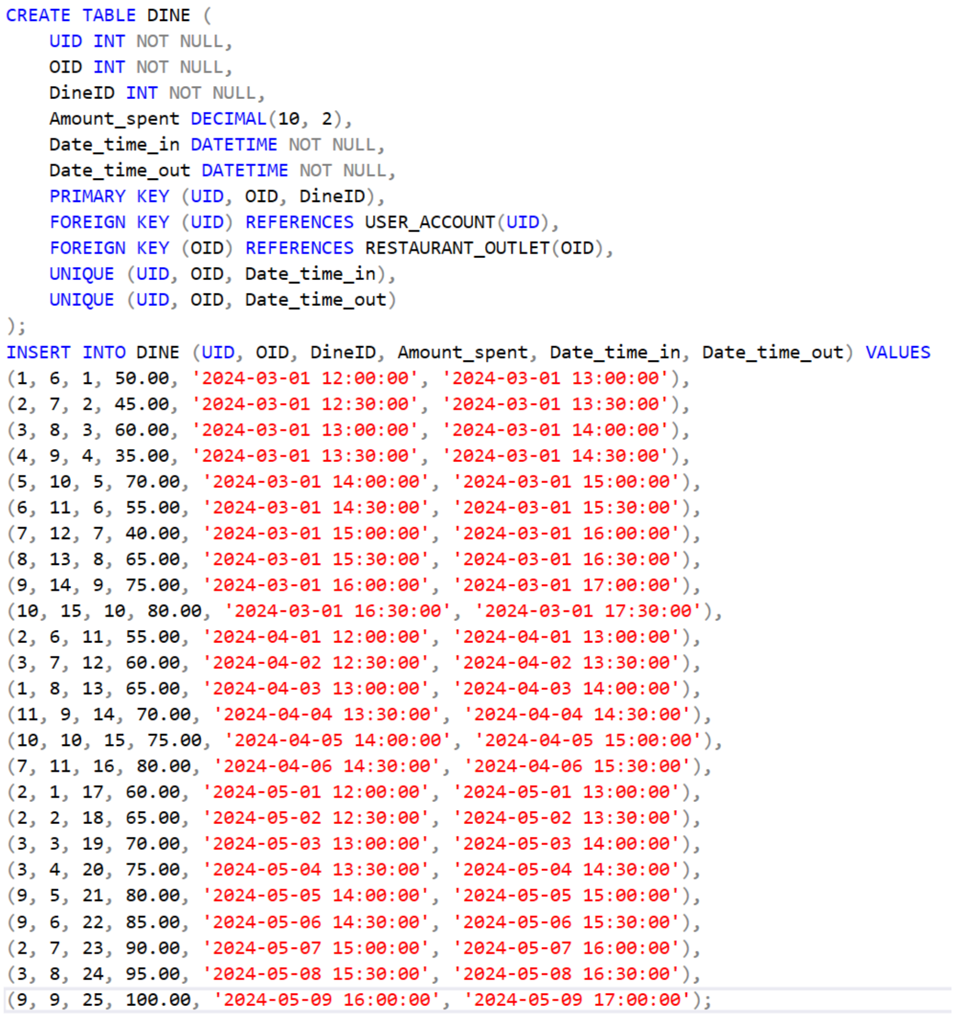


```
CREATE TABLE RESTAURANT_OUTLET (
    OID INT PRIMARY KEY,
    Name VARCHAR(255) NOT NULL,
    RID INT,
    MID INT NOT NULL,
    FOREIGN KEY (RID) REFERENCES RESTAURANT_CHAIN(RID),
    FOREIGN KEY (MID) REFERENCES MALL(MID)
);
INSERT INTO RESTAURANT_OUTLET (OID, Name, RID, MID) VALUES
(1, 'McDonald''s - Jurong Point', 1, 1),
(2, 'McDonald''s - Pioneer Mall', 1, 2),
(3, 'McDonald''s - NEX', 1, 6),
(4, 'Burger King - NEX', 2, 6),
(5, 'Burger King - IMM', 2, 3),
(6, 'The Hungry Chef', NULL, 1),
(7, 'Sizzle & Spice', NULL, 2),
(8, 'Urban Eats', NULL, 4),
(9, 'Garden Bistro', NULL, 7),
(10, 'Flavor Fusion', NULL, 3),
(11, 'McDonald''s - Jem', 1, 4),
(12, 'McDonald''s - Westgate', 1, 5),
(13, 'Burger King - NEX', 2, 6),
(14, 'Burger King - VivoCity', 2, 7),
(15, 'Taste of Tradition', NULL, 1);
```

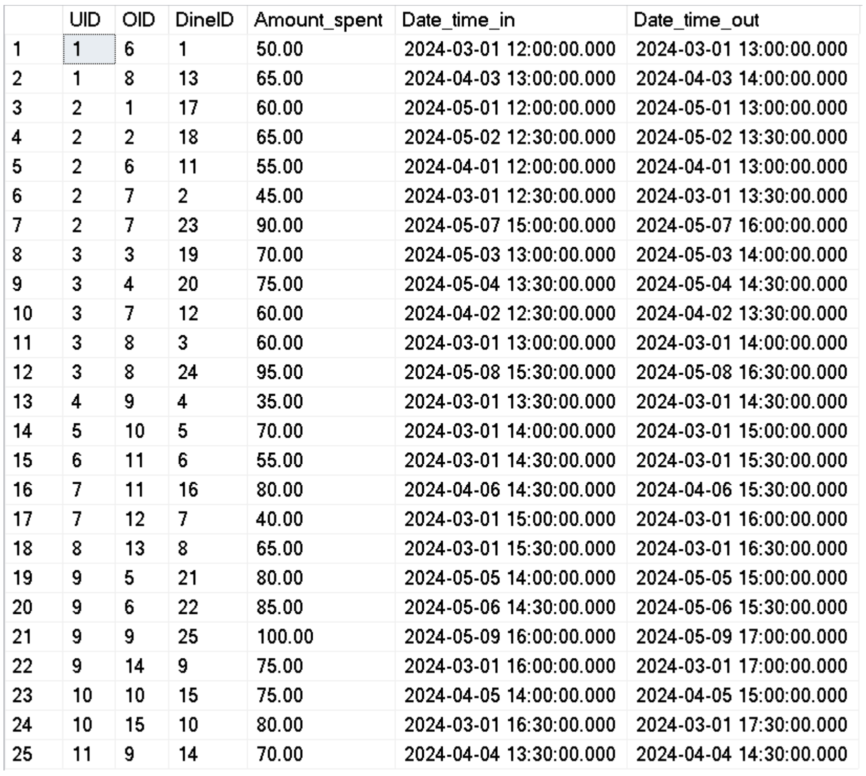


	OID	Name	RID	MID
1	1	McDonald's - Jurong Point	1	1
2	2	McDonald's - Pioneer Mall	1	2
3	3	McDonald's - NEX	1	6
4	4	Burger King - NEX	2	6
5	5	Burger King - IMM	2	3
6	6	The Hungry Chef	NULL	1
7	7	Sizzle & Spice	NULL	2
8	8	Urban Eats	NULL	4
9	9	Garden Bistro	NULL	7
10	10	Flavor Fusion	NULL	3
11	11	McDonald's - Jem	1	4
12	12	McDonald's - Westgate	1	5
13	13	Burger King - NEX	2	6
14	14	Burger King - VivoCity	2	7
15	15	Taste of Tradition	NULL	1

DINE

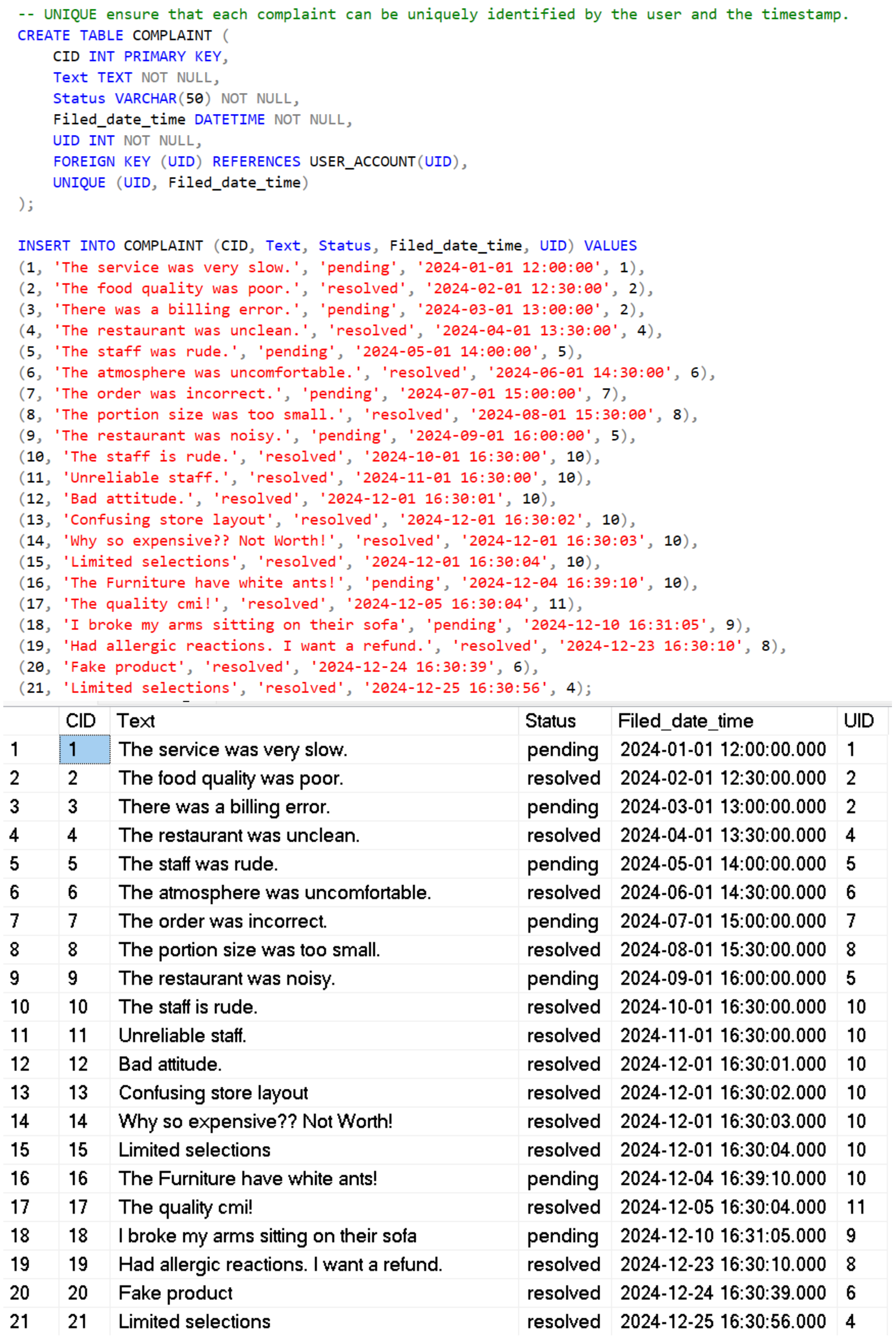


```
CREATE TABLE DINE (
    UID INT NOT NULL,
    OID INT NOT NULL,
    DineID INT NOT NULL,
    Amount_spent DECIMAL(10, 2),
    Date_time_in DATETIME NOT NULL,
    Date_time_out DATETIME NOT NULL,
    PRIMARY KEY (UID, OID, DineID),
    FOREIGN KEY (UID) REFERENCES USER_ACCOUNT(UID),
    FOREIGN KEY (OID) REFERENCES RESTAURANT_OUTLET(OID),
    UNIQUE (UID, OID, Date_time_in),
    UNIQUE (UID, OID, Date_time_out)
);
INSERT INTO DINE (UID, OID, DineID, Amount_spent, Date_time_in, Date_time_out) VALUES
(1, 6, 1, 50.00, '2024-03-01 12:00:00', '2024-03-01 13:00:00'),
(2, 7, 2, 45.00, '2024-03-01 12:30:00', '2024-03-01 13:30:00'),
(3, 8, 3, 60.00, '2024-03-01 13:00:00', '2024-03-01 14:00:00'),
(4, 9, 4, 35.00, '2024-03-01 13:30:00', '2024-03-01 14:30:00'),
(5, 10, 5, 70.00, '2024-03-01 14:00:00', '2024-03-01 15:00:00'),
(6, 11, 6, 55.00, '2024-03-01 14:30:00', '2024-03-01 15:30:00'),
(7, 12, 7, 40.00, '2024-03-01 15:00:00', '2024-03-01 16:00:00'),
(8, 13, 8, 65.00, '2024-03-01 15:30:00', '2024-03-01 16:30:00'),
(9, 14, 9, 75.00, '2024-03-01 16:00:00', '2024-03-01 17:00:00'),
(10, 15, 10, 80.00, '2024-03-01 16:30:00', '2024-03-01 17:30:00'),
(2, 6, 11, 55.00, '2024-04-01 12:00:00', '2024-04-01 13:00:00'),
(3, 7, 12, 60.00, '2024-04-02 12:30:00', '2024-04-02 13:30:00'),
(1, 8, 13, 65.00, '2024-04-03 13:00:00', '2024-04-03 14:00:00'),
(11, 9, 14, 70.00, '2024-04-04 13:30:00', '2024-04-04 14:30:00'),
(10, 10, 15, 75.00, '2024-04-05 14:00:00', '2024-04-05 15:00:00'),
(7, 11, 16, 80.00, '2024-04-06 14:30:00', '2024-04-06 15:30:00'),
(2, 1, 17, 60.00, '2024-05-01 12:00:00', '2024-05-01 13:00:00'),
(2, 2, 18, 65.00, '2024-05-02 12:30:00', '2024-05-02 13:30:00'),
(3, 3, 19, 70.00, '2024-05-03 13:00:00', '2024-05-03 14:00:00'),
(3, 4, 20, 75.00, '2024-05-04 13:30:00', '2024-05-04 14:30:00'),
(9, 5, 21, 80.00, '2024-05-05 14:00:00', '2024-05-05 15:00:00'),
(9, 6, 22, 85.00, '2024-05-06 14:30:00', '2024-05-06 15:30:00'),
(2, 7, 23, 90.00, '2024-05-07 15:00:00', '2024-05-07 16:00:00'),
(3, 8, 24, 95.00, '2024-05-08 15:30:00', '2024-05-08 16:30:00'),
(9, 9, 25, 100.00, '2024-05-09 16:00:00', '2024-05-09 17:00:00');
```



	UID	OID	DineID	Amount_spent	Date_time_in	Date_time_out
1	1	6	1	50.00	2024-03-01 12:00:00.000	2024-03-01 13:00:00.000
2	1	8	13	65.00	2024-04-03 13:00:00.000	2024-04-03 14:00:00.000
3	2	1	17	60.00	2024-05-01 12:00:00.000	2024-05-01 13:00:00.000
4	2	2	18	65.00	2024-05-02 12:30:00.000	2024-05-02 13:30:00.000
5	2	6	11	55.00	2024-04-01 12:00:00.000	2024-04-01 13:00:00.000
6	2	7	2	45.00	2024-03-01 12:30:00.000	2024-03-01 13:30:00.000
7	2	7	23	90.00	2024-05-07 15:00:00.000	2024-05-07 16:00:00.000
8	3	3	19	70.00	2024-05-03 13:00:00.000	2024-05-03 14:00:00.000
9	3	4	20	75.00	2024-05-04 13:30:00.000	2024-05-04 14:30:00.000
10	3	7	12	60.00	2024-04-02 12:30:00.000	2024-04-02 13:30:00.000
11	3	8	3	60.00	2024-03-01 13:00:00.000	2024-03-01 14:00:00.000
12	3	8	24	95.00	2024-05-08 15:30:00.000	2024-05-08 16:30:00.000
13	4	9	4	35.00	2024-03-01 13:30:00.000	2024-03-01 14:30:00.000
14	5	10	5	70.00	2024-03-01 14:00:00.000	2024-03-01 15:00:00.000
15	6	11	6	55.00	2024-03-01 14:30:00.000	2024-03-01 15:30:00.000
16	7	11	16	80.00	2024-04-06 14:30:00.000	2024-04-06 15:30:00.000
17	7	12	7	40.00	2024-03-01 15:00:00.000	2024-03-01 16:00:00.000
18	8	13	8	65.00	2024-03-01 15:30:00.000	2024-03-01 16:30:00.000
19	9	5	21	80.00	2024-05-05 14:00:00.000	2024-05-05 15:00:00.000
20	9	6	22	85.00	2024-05-06 14:30:00.000	2024-05-06 15:30:00.000
21	9	9	25	100.00	2024-05-09 16:00:00.000	2024-05-09 17:00:00.000
22	9	14	9	75.00	2024-03-01 16:00:00.000	2024-03-01 17:00:00.000
23	10	10	15	75.00	2024-04-05 14:00:00.000	2024-04-05 15:00:00.000
24	10	15	10	80.00	2024-03-01 16:30:00.000	2024-03-01 17:30:00.000
25	11	9	14	70.00	2024-04-04 13:30:00.000	2024-04-04 14:30:00.000

COMPLAINT



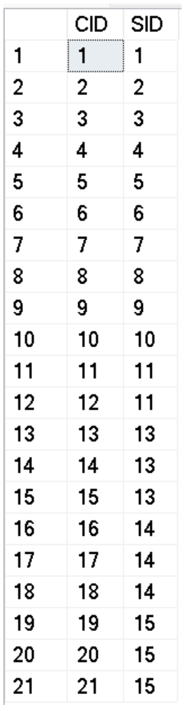
```sql
-- UNIQUE ensure that each complaint can be uniquely identified by the user and the timestamp.
CREATE TABLE COMPLAINT (
    CID INT PRIMARY KEY,
    Text TEXT NOT NULL,
    Status VARCHAR(50) NOT NULL,
    Filed_date_time DATETIME NOT NULL,
    UID INT NOT NULL,
    FOREIGN KEY (UID) REFERENCES USER_ACCOUNT(UID),
    UNIQUE (UID, Filed_date_time)
);

INSERT INTO COMPLAINT (CID, Text, Status, Filed_date_time, UID) VALUES
(1, 'The service was very slow.', 'pending', '2024-01-01 12:00:00', 1),
(2, 'The food quality was poor.', 'resolved', '2024-02-01 12:30:00', 2),
(3, 'There was a billing error.', 'pending', '2024-03-01 13:00:00', 2),
(4, 'The restaurant was unclean.', 'resolved', '2024-04-01 13:30:00', 4),
(5, 'The staff was rude.', 'pending', '2024-05-01 14:00:00', 5),
(6, 'The atmosphere was uncomfortable.', 'resolved', '2024-06-01 14:30:00', 6),
(7, 'The order was incorrect.', 'pending', '2024-07-01 15:00:00', 7),
(8, 'The portion size was too small.', 'resolved', '2024-08-01 15:30:00', 8),
(9, 'The restaurant was noisy.', 'pending', '2024-09-01 16:00:00', 5),
(10, 'The staff is rude.', 'resolved', '2024-10-01 16:30:00', 10),
(11, 'Unreliable staff.', 'resolved', '2024-11-01 16:30:00', 10),
(12, 'Bad attitude.', 'resolved', '2024-12-01 16:30:01', 10),
(13, 'Confusing store layout', 'resolved', '2024-12-01 16:30:02', 10),
(14, 'Why so expensive?? Not Worth!', 'resolved', '2024-12-01 16:30:03', 10),
(15, 'Limited selections', 'resolved', '2024-12-01 16:30:04', 10),
(16, 'The Furniture have white ants!', 'pending', '2024-12-04 16:39:10', 10),
(17, 'The quality cmi!', 'resolved', '2024-12-05 16:30:04', 11),
(18, 'I broke my arms sitting on their sofa', 'pending', '2024-12-10 16:31:05', 9),
(19, 'Had allergic reactions. I want a refund.', 'resolved', '2024-12-23 16:30:10', 8),
(20, 'Fake product', 'resolved', '2024-12-24 16:30:39', 6),
(21, 'Limited selections', 'resolved', '2024-12-25 16:30:56', 4);
```

	CID	Text	Status	Filed_date_time	UID
1	1	The service was very slow.	pending	2024-01-01 12:00:00.000	1
2	2	The food quality was poor.	resolved	2024-02-01 12:30:00.000	2
3	3	There was a billing error.	pending	2024-03-01 13:00:00.000	2
4	4	The restaurant was unclean.	resolved	2024-04-01 13:30:00.000	4
5	5	The staff was rude.	pending	2024-05-01 14:00:00.000	5
6	6	The atmosphere was uncomfortable.	resolved	2024-06-01 14:30:00.000	6
7	7	The order was incorrect.	pending	2024-07-01 15:00:00.000	7
8	8	The portion size was too small.	resolved	2024-08-01 15:30:00.000	8
9	9	The restaurant was noisy.	pending	2024-09-01 16:00:00.000	5
10	10	The staff is rude.	resolved	2024-10-01 16:30:00.000	10
11	11	Unreliable staff.	resolved	2024-11-01 16:30:00.000	10
12	12	Bad attitude.	resolved	2024-12-01 16:30:01.000	10
13	13	Confusing store layout	resolved	2024-12-01 16:30:02.000	10
14	14	Why so expensive?? Not Worth!	resolved	2024-12-01 16:30:03.000	10
15	15	Limited selections	resolved	2024-12-01 16:30:04.000	10
16	16	The Furniture have white ants!	pending	2024-12-04 16:39:10.000	10
17	17	The quality cmi!	resolved	2024-12-05 16:30:04.000	11
18	18	I broke my arms sitting on their sofa	pending	2024-12-10 16:31:05.000	9
19	19	Had allergic reactions. I want a refund.	resolved	2024-12-23 16:30:10.000	8
20	20	Fake product	resolved	2024-12-24 16:30:39.000	6
21	21	Limited selections	resolved	2024-12-25 16:30:56.000	4

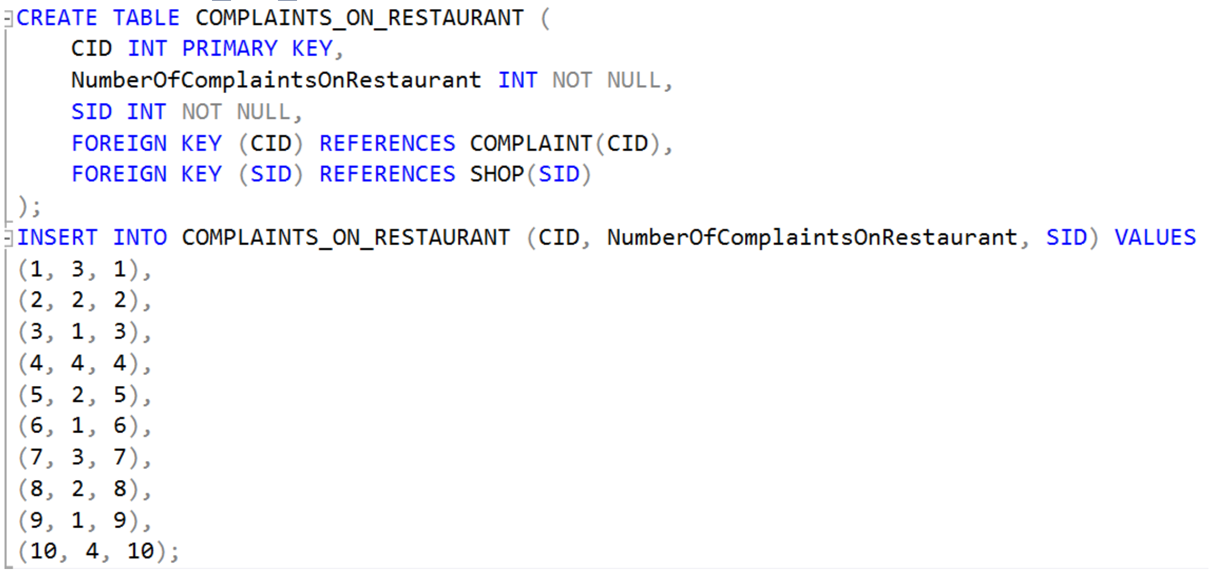
COMPLAINTS_ON_SHOP

```
CREATE TABLE COMPLAINTS_ON_SHOP (
    CID INT PRIMARY KEY,
    SID INT NOT NULL,
    FOREIGN KEY (CID) REFERENCES COMPLAINT(CID),
    FOREIGN KEY (SID) REFERENCES SHOP(SID)
);
INSERT INTO COMPLAINTS_ON_SHOP (CID, SID) VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5),
(6, 6),
(7, 7),
(8, 8),
(9, 9),
(10, 10),
(11, 11),
(12, 11),
(13, 13),
(14, 13),
(15, 13),
(16, 14),
(17, 14),
(18, 14),
(19, 15),
(20, 15),
(21, 15);
```

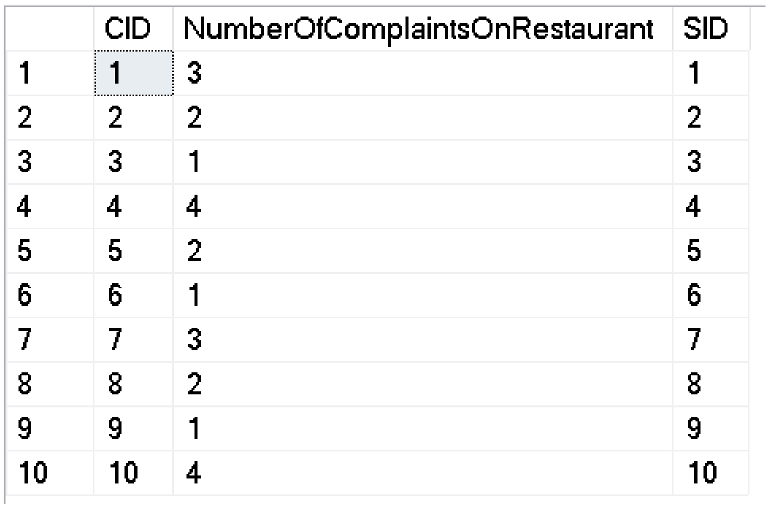


	CID	SID
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	9	9
10	10	10
11	11	11
12	12	11
13	13	13
14	14	13
15	15	13
16	16	14
17	17	14
18	18	14
19	19	15
20	20	15
21	21	15

COMPLAINTS_ON_RESTAURANT

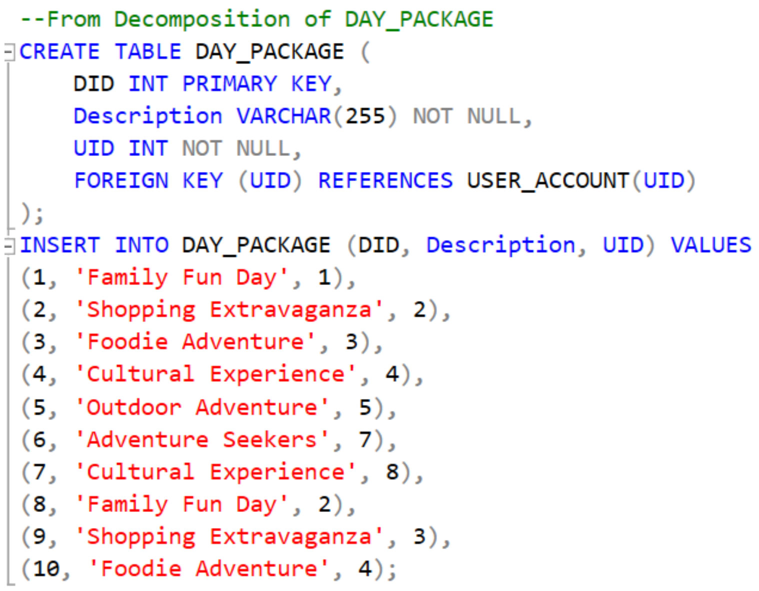


```
CREATE TABLE COMPLAINTS_ON_RESTAURANT (
    CID INT PRIMARY KEY,
    NumberOfComplaintsOnRestaurant INT NOT NULL,
    SID INT NOT NULL,
    FOREIGN KEY (CID) REFERENCES COMPLAINT(CID),
    FOREIGN KEY (SID) REFERENCES SHOP(SID)
);
INSERT INTO COMPLAINTS_ON_RESTAURANT (CID, NumberOfComplaintsOnRestaurant, SID) VALUES
(1, 3, 1),
(2, 2, 2),
(3, 1, 3),
(4, 4, 4),
(5, 2, 5),
(6, 1, 6),
(7, 3, 7),
(8, 2, 8),
(9, 1, 9),
(10, 4, 10);
```

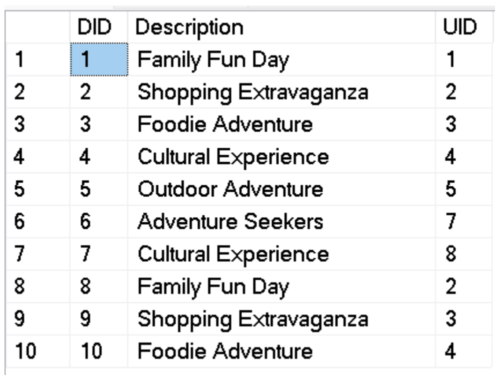


	CID	NumberOfComplaintsOnRestaurant	SID
1	1	3	1
2	2	2	2
3	3	1	3
4	4	4	4
5	5	2	5
6	6	1	6
7	7	3	7
8	8	2	8
9	9	1	9
10	10	4	10

DAY_PACKAGE

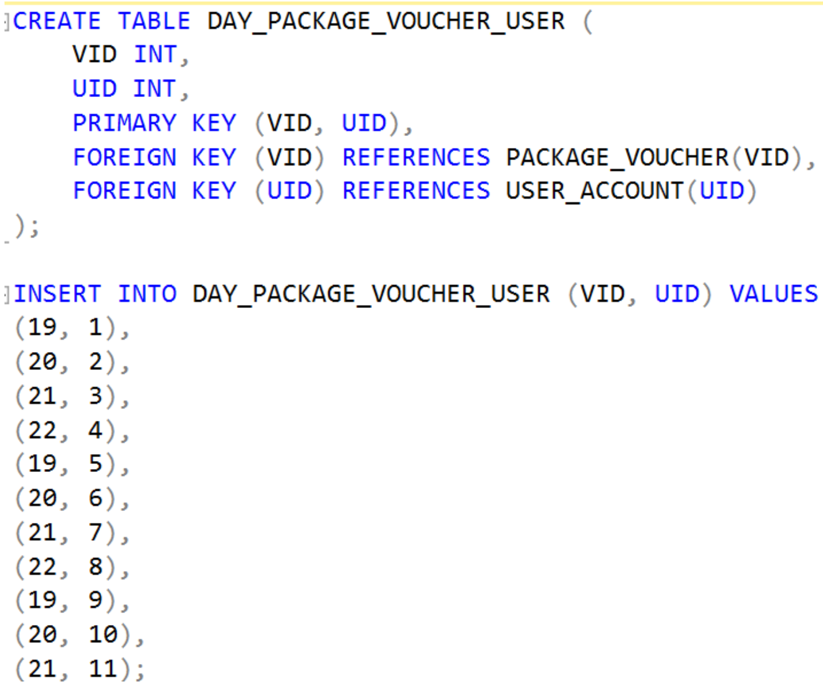


```sql
--From Decomposition of DAY_PACKAGE
CREATE TABLE DAY_PACKAGE (
    DID INT PRIMARY KEY,
    Description VARCHAR(255) NOT NULL,
    UID INT NOT NULL,
    FOREIGN KEY (UID) REFERENCES USER_ACCOUNT(UID)
);
INSERT INTO DAY_PACKAGE (DID, Description, UID) VALUES
(1, 'Family Fun Day', 1),
(2, 'Shopping Extravaganza', 2),
(3, 'Foodie Adventure', 3),
(4, 'Cultural Experience', 4),
(5, 'Outdoor Adventure', 5),
(6, 'Adventure Seekers', 7),
(7, 'Cultural Experience', 8),
(8, 'Family Fun Day', 2),
(9, 'Shopping Extravaganza', 3),
(10, 'Foodie Adventure', 4);
```



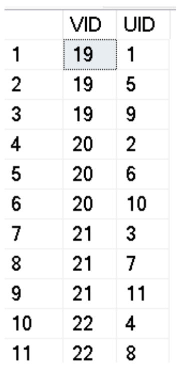
	DID	Description	UID
1	1	Family Fun Day	1
2	2	Shopping Extravaganza	2
3	3	Foodie Adventure	3
4	4	Cultural Experience	4
5	5	Outdoor Adventure	5
6	6	Adventure Seekers	7
7	7	Cultural Experience	8
8	8	Family Fun Day	2
9	9	Shopping Extravaganza	3
10	10	Foodie Adventure	4

DAY_PACKAGE_VOUCHER_USER



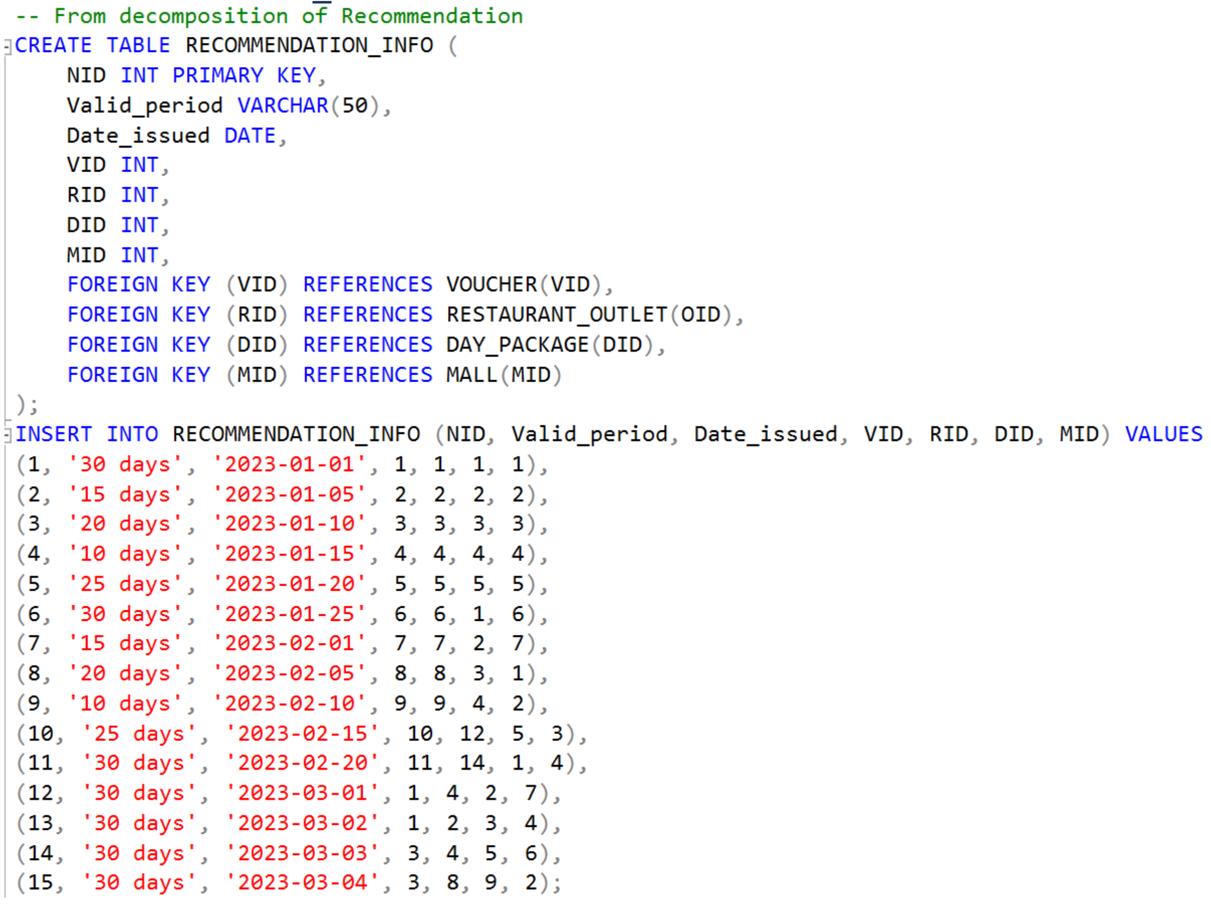
```
CREATE TABLE DAY_PACKAGE_VOUCHER_USER (
    VID INT,
    UID INT,
    PRIMARY KEY (VID, UID),
    FOREIGN KEY (VID) REFERENCES PACKAGE_VOUCHER(VID),
    FOREIGN KEY (UID) REFERENCES USER_ACCOUNT(UID)
);

INSERT INTO DAY_PACKAGE_VOUCHER_USER (VID, UID) VALUES
(19, 1),
(20, 2),
(21, 3),
(22, 4),
(19, 5),
(20, 6),
(21, 7),
(22, 8),
(19, 9),
(20, 10),
(21, 11);
```

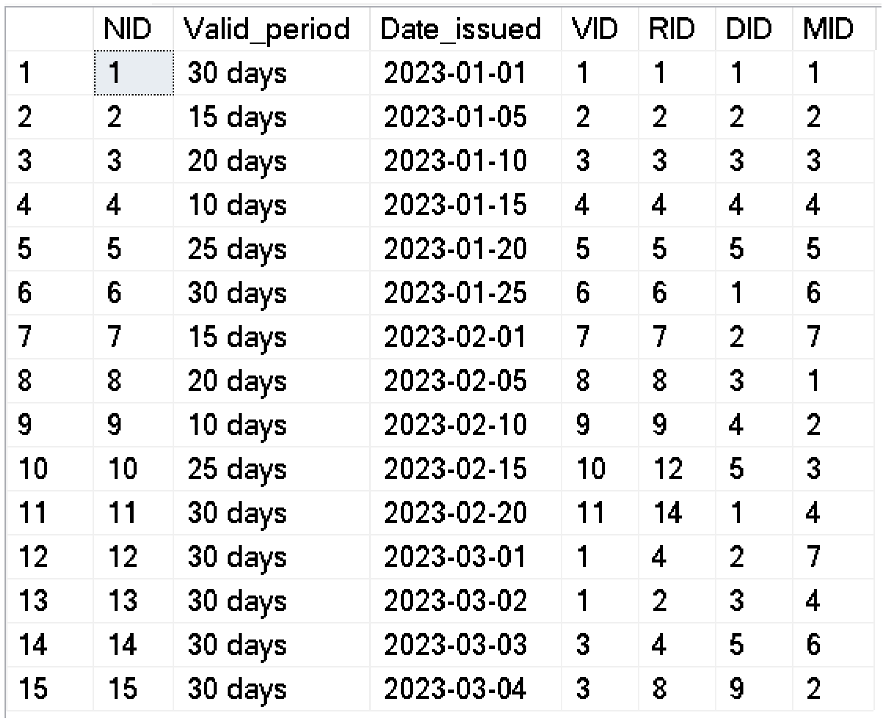


	VID	UID
1	19	1
2	19	5
3	19	9
4	20	2
5	20	6
6	20	10
7	21	3
8	21	7
9	21	11
10	22	4
11	22	8

RECOMMENDATION_INFO

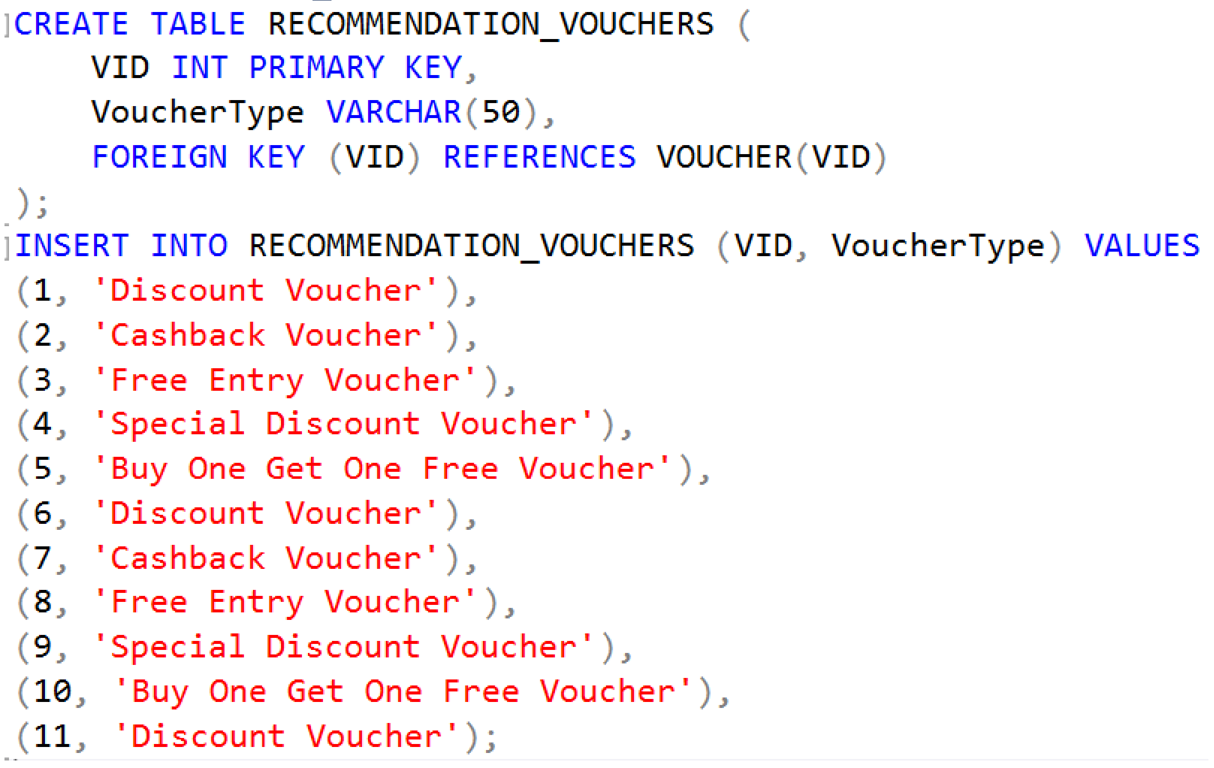


```sql
-- From decomposition of Recommendation
CREATE TABLE RECOMMENDATION_INFO (
    NID INT PRIMARY KEY,
    Valid_period VARCHAR(50),
    Date_issued DATE,
    VID INT,
    RID INT,
    DID INT,
    MID INT,
    FOREIGN KEY (VID) REFERENCES VOUCHER(VID),
    FOREIGN KEY (RID) REFERENCES RESTAURANT_OUTLET(OID),
    FOREIGN KEY (DID) REFERENCES DAY_PACKAGE(DID),
    FOREIGN KEY (MID) REFERENCES MALL(MID)
);
INSERT INTO RECOMMENDATION_INFO (NID, Valid_period, Date_issued, VID, RID, DID, MID) VALUES
(1, '30 days', '2023-01-01', 1, 1, 1, 1),
(2, '15 days', '2023-01-05', 2, 2, 2, 2),
(3, '20 days', '2023-01-10', 3, 3, 3, 3),
(4, '10 days', '2023-01-15', 4, 4, 4, 4),
(5, '25 days', '2023-01-20', 5, 5, 5, 5),
(6, '30 days', '2023-01-25', 6, 6, 1, 6),
(7, '15 days', '2023-02-01', 7, 7, 2, 7),
(8, '20 days', '2023-02-05', 8, 8, 3, 1),
(9, '10 days', '2023-02-10', 9, 9, 4, 2),
(10, '25 days', '2023-02-15', 10, 12, 5, 3),
(11, '30 days', '2023-02-20', 11, 14, 1, 4),
(12, '30 days', '2023-03-01', 1, 4, 2, 7),
(13, '30 days', '2023-03-02', 1, 2, 3, 4),
(14, '30 days', '2023-03-03', 3, 4, 5, 6),
(15, '30 days', '2023-03-04', 3, 8, 9, 2);
```

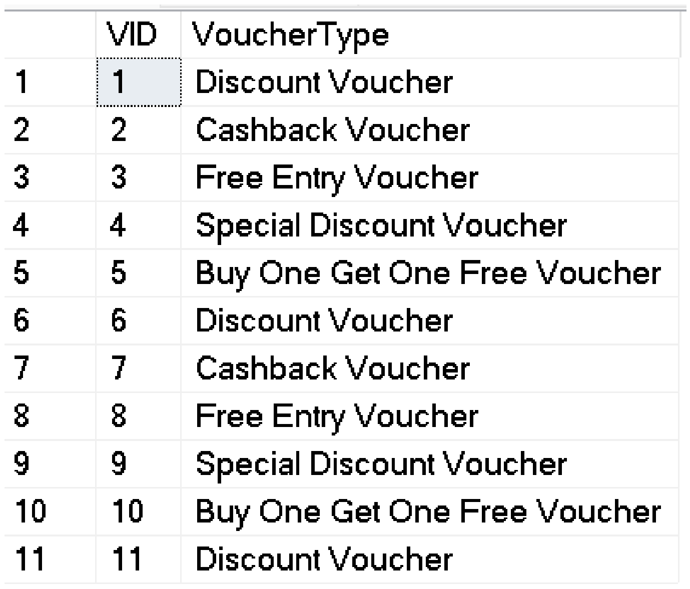


	NID	Valid_period	Date_issued	VID	RID	DID	MID
1	1	30 days	2023-01-01	1	1	1	1
2	2	15 days	2023-01-05	2	2	2	2
3	3	20 days	2023-01-10	3	3	3	3
4	4	10 days	2023-01-15	4	4	4	4
5	5	25 days	2023-01-20	5	5	5	5
6	6	30 days	2023-01-25	6	6	1	6
7	7	15 days	2023-02-01	7	7	2	7
8	8	20 days	2023-02-05	8	8	3	1
9	9	10 days	2023-02-10	9	9	4	2
10	10	25 days	2023-02-15	10	12	5	3
11	11	30 days	2023-02-20	11	14	1	4
12	12	30 days	2023-03-01	1	4	2	7
13	13	30 days	2023-03-02	1	2	3	4
14	14	30 days	2023-03-03	3	4	5	6
15	15	30 days	2023-03-04	3	8	9	2

RECOMMENDATION_VOUCHERS



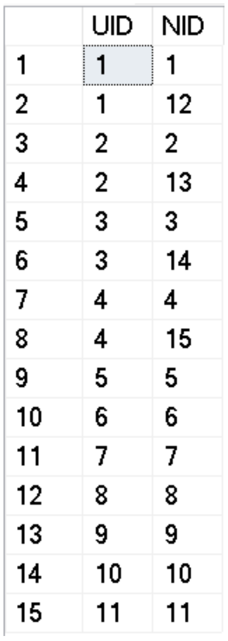
```sql
CREATE TABLE RECOMMENDATION_VOUCHERS (
    VID INT PRIMARY KEY,
    VoucherType VARCHAR(50),
    FOREIGN KEY (VID) REFERENCES VOUCHER(VID)
);
INSERT INTO RECOMMENDATION_VOUCHERS (VID, VoucherType) VALUES
(1, 'Discount Voucher'),
(2, 'Cashback Voucher'),
(3, 'Free Entry Voucher'),
(4, 'Special Discount Voucher'),
(5, 'Buy One Get One Free Voucher'),
(6, 'Discount Voucher'),
(7, 'Cashback Voucher'),
(8, 'Free Entry Voucher'),
(9, 'Special Discount Voucher'),
(10, 'Buy One Get One Free Voucher'),
(11, 'Discount Voucher');
```



	VID	VoucherType
1	1	Discount Voucher
2	2	Cashback Voucher
3	3	Free Entry Voucher
4	4	Special Discount Voucher
5	5	Buy One Get One Free Voucher
6	6	Discount Voucher
7	7	Cashback Voucher
8	8	Free Entry Voucher
9	9	Special Discount Voucher
10	10	Buy One Get One Free Voucher
11	11	Discount Voucher

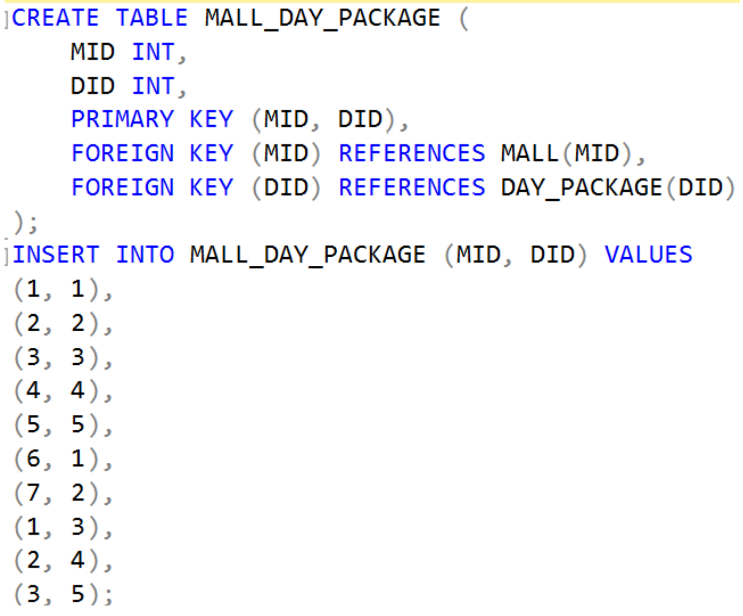
USER_RECOMMENDATION

```
CREATE TABLE USER_RECOMMENDATION (
    UID INT,
    NID INT,
    PRIMARY KEY (UID, NID),
    FOREIGN KEY (UID) REFERENCES USER_ACCOUNT(UID),
    FOREIGN KEY (NID) REFERENCES RECOMMENDATION_INFO(NID)
);
INSERT INTO USER_RECOMMENDATION (UID, NID) VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5),
(6, 6),
(7, 7),
(8, 8),
(9, 9),
(10, 10),
(11, 11),
(1, 12),
(2, 13),
(3, 14),
(4, 15);
```

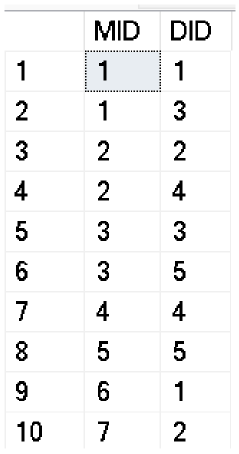


	UID	NID
1	1	1
2	1	12
3	2	2
4	2	13
5	3	3
6	3	14
7	4	4
8	4	15
9	5	5
10	6	6
11	7	7
12	8	8
13	9	9
14	10	10
15	11	11

MALL_DAY_PACKAGE

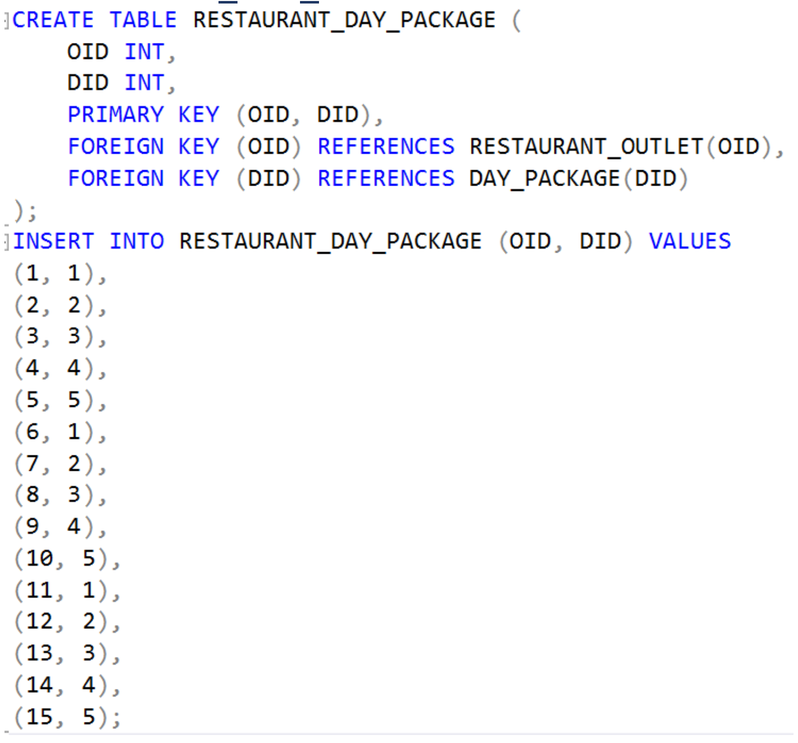


```sql
CREATE TABLE MALL_DAY_PACKAGE (
    MID INT,
    DID INT,
    PRIMARY KEY (MID, DID),
    FOREIGN KEY (MID) REFERENCES MALL(MID),
    FOREIGN KEY (DID) REFERENCES DAY_PACKAGE(DID)
);
INSERT INTO MALL_DAY_PACKAGE (MID, DID) VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5),
(6, 1),
(7, 2),
(1, 3),
(2, 4),
(3, 5);
```

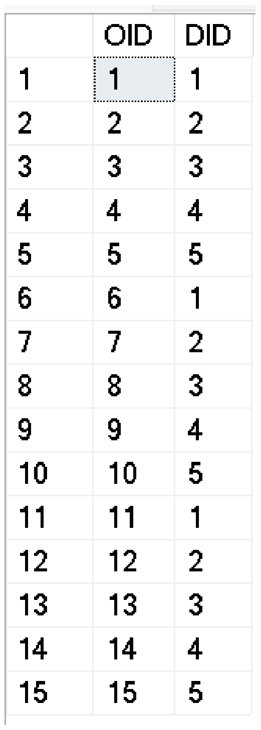


	MID	DID
1	1	1
2	1	3
3	2	2
4	2	4
5	3	3
6	3	5
7	4	4
8	5	5
9	6	1
10	7	2

RESTAURANT_DAY_PACKAGE



```
CREATE TABLE RESTAURANT_DAY_PACKAGE (
    OID INT,
    DID INT,
    PRIMARY KEY (OID, DID),
    FOREIGN KEY (OID) REFERENCES RESTAURANT_OUTLET(OID),
    FOREIGN KEY (DID) REFERENCES DAY_PACKAGE(DID)
);
INSERT INTO RESTAURANT_DAY_PACKAGE (OID, DID) VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5),
(6, 1),
(7, 2),
(8, 3),
(9, 4),
(10, 5),
(11, 1),
(12, 2),
(13, 3),
(14, 4),
(15, 5);
```



	OID	DID
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	1
7	7	2
8	8	3
9	9	4
10	10	5
11	11	1
12	12	2
13	13	3
14	14	4
15	15	5

Description of any additional effort made

- Addition of Names for Malls, Restaurants, Restaurant Chains, Shops into our schemas so that users know what is the business name rather than just ID numbers in the SQL query.
- Research and use of some real-life data to keep the lab realistic and of some use.
- Extensive tweakings and insertions of relations to fulfil the lab’s output requirements.
- Ensuring constraints on almost every variables to ensure the candidate keys integrity.