AY2023/24 SEMESTER 2

Fifth Laboratory Session: Final Demonstration

SC2207 Introduction To Databases

Lab Supervisor: Prof. Long Cheng

Lab Teaching Assistant: Feng Shibo

Lab Group: SCSE

Group: Group 4

Date of Submission: 2 April 2024

Name	**Matriculation Number**
Aaron Jerome Lim Li Yang	U2221616A
Chan Fun Soon Nicholas	U2040164D
Keith Ang Kee Chun	U2220349G
Koh Yihao Kendrick	U2222663K
Oh ShuYi	U2220108H
Soh Shing Hui	U2040741F
Tan Jun Kiat	U2240182K

Table of contents

SQL DDL commands for table creation

```
CREATE DATABASE SCSEG4;
GO
USE SCSEG4;
GO

-- order is not numerical due to foreign key dependencies

CREATE TABLE USER_ACCOUNT ( --R1
   UID INT PRIMARY KEY,
   Gender CHAR(1),
   DOB DATE, -- DOB of the format YYYY-MM-DD
   Name VARCHAR(30),
   PhoneNumber INT,
   CHECK(Gender IN('M', 'F', 'O')), -- Male / Female / Others
   CHECK(PhoneNumber>=80000000 AND PhoneNumber<=99999999),
);

CREATE TABLE RELATED ( --R2
   Person1_UID INT NOT NULL FOREIGN KEY REFERENCES USER_ACCOUNT(UID),
   Person2_UID INT NOT NULL FOREIGN KEY REFERENCES USER_ACCOUNT(UID),
   Type VARCHAR(20) NOT NULL, -- custom relationship type allowed
   PRIMARY KEY(Person1_UID, Person2_UID),
);

CREATE TABLE MALL_MGMT_COMPANY ( --R6 (must create this before mall)
   COMPANY_CID INT PRIMARY KEY,
   Address VARCHAR(50) NOT NULL UNIQUE,
);

CREATE TABLE MALL ( --R5 (must create mall first before shop for foreign key)
   MID INT PRIMARY KEY,
   Address VARCHAR(50) NOT NULL UNIQUE,
   NumShops INT NOT NULL,
   COMPANY_CID INT FOREIGN KEY REFERENCES MALL_MGMT_COMPANY(COMPANY_CID), -- dont need
to belong to a company
   CHECK(NumShops>=0),
);

CREATE TABLE SHOP_NOUN ( --R3
   SID INT PRIMARY KEY,
   Type VARCHAR(20) NOT NULL,
   MID INT NOT NULL FOREIGN KEY REFERENCES MALL(MID), -- must be in a mall
   CHECK(Type IN('Restaurant', 'Cafe', 'Retail')),
);

CREATE TABLE SHOP_VERB ( --R4
   SID INT NOT NULL FOREIGN KEY REFERENCES SHOP_NOUN(SID),
   UID INT NOT NULL FOREIGN KEY REFERENCES USER_ACCOUNT(UID),
   Amount_spent DECIMAL(10,2) NOT NULL,
   Date_time_in SMALLDATETIME NOT NULL, -- format: YYYY-MM-DD HH:MI:SS (rounded to
nearest minute due to prev assumptions)
   Date_time_out SMALLDATETIME NOT NULL,
   PRIMARY KEY(SID, UID, Date_time_in),
   UNIQUE(SID, UID, Date_time_out),
   CHECK(Amount_spent>=0),
   CHECK(Date_time_in <= Date_time_out),
);

CREATE TABLE COMPLAINT ( --R7
   COMPLAINT_CID INT PRIMARY KEY,
   Text VARCHAR(500) NOT NULL,
   Status VARCHAR(20) NOT NULL,
   Filled_date_time SMALLDATETIME NOT NULL, -- format: YYYY-MM-DD HH:MI:SS (rounded to
nearest minute due to prev assumptions)
   UID INT NOT NULL FOREIGN KEY REFERENCES USER_ACCOUNT(UID),
   CHECK(Status IN('pending', 'being handled', 'addressed')),
);

CREATE TABLE COMPLAINTS_ON_SHOP ( --R8
   COMPLAINT_CID INT PRIMARY KEY FOREIGN KEY REFERENCES COMPLAINT(COMPLAINT_CID),
   SID INT NOT NULL FOREIGN KEY REFERENCES SHOP_NOUN(SID),
);

CREATE TABLE RESTAURANT_CHAIN ( --R11 (create chain before restaurant)
   RID INT PRIMARY KEY,
   Address VARCHAR(50) NOT NULL UNIQUE,
);

CREATE TABLE RESTAURANT_OUTLET ( --R10 (must create this before restaurant complaints)
   OID INT PRIMARY KEY,
   RID INT FOREIGN KEY REFERENCES RESTAURANT_CHAIN(RID), -- outlet dont need to belong
to a chain
```

```
   MID INT NOT NULL FOREIGN KEY REFERENCES MALL(MID),
);

CREATE TABLE COMPLAINTS_ON_RESTAURANT ( --R9
   COMPLAINT_CID INT PRIMARY KEY FOREIGN KEY REFERENCES COMPLAINT(COMPLAINT_CID),
   OID INT NOT NULL FOREIGN KEY REFERENCES RESTAURANT_OUTLET(OID),
);

CREATE TABLE DINE ( --R12
   UID INT NOT NULL FOREIGN KEY REFERENCES USER_ACCOUNT(UID),
   OID INT NOT NULL FOREIGN KEY REFERENCES RESTAURANT_OUTLET(OID),
   Amount_spent DECIMAL(10,2) NOT NULL,
   Date_time_in SMALLDATETIME NOT NULL, -- format: YYYY-MM-DD HH:MI:SS (rounded to
nearest minute due to prev assumptions)
   Date_time_out SMALLDATETIME NOT NULL,
   PRIMARY KEY(UID, OID, Date_time_in),
   UNIQUE(UID, OID, Date_time_out),
   CHECK(Amount_spent>=0),
   CHECK(Date_time_in <= Date_time_out),
);

CREATE TABLE VOUCHER ( --R18 (voucher before package)
   VID INT PRIMARY KEY,
   Status VARCHAR(20) NOT NULL,
   Expiry_date DATE NOT NULL, -- YYYY-MM-DD
   Date_Issued DATE NOT NULL, -- YYYY-MM-DD
   Description VARCHAR(500) NOT NULL,
   CHECK(Status IN('ALLOCATED', 'REDEEMED', 'EXPIRED')),
   CHECK(Date_Issued <= Expiry_date),
);

CREATE TABLE DAY_PACKAGE ( --R13
   DID INT PRIMARY KEY,
   Description VARCHAR(500) NOT NULL,
   UID INT NOT NULL FOREIGN KEY REFERENCES USER_ACCOUNT(UID), -- must have user
   VID INT NOT NULL UNIQUE FOREIGN KEY REFERENCES VOUCHER(VID), -- must have unique
voucher tagged to package
);

CREATE TABLE MALL_HAS_PACKAGE ( --R14
   MID INT NOT NULL FOREIGN KEY REFERENCES MALL(MID),
   DID INT NOT NULL FOREIGN KEY REFERENCES DAY_PACKAGE(DID),
```

```sql
   PRIMARY KEY(MID, DID),
);

CREATE TABLE PACKAGE_HAS_RESTAURANT ( --R15
   OID INT NOT NULL FOREIGN KEY REFERENCES RESTAURANT_OUTLET(OID),
   DID INT NOT NULL FOREIGN KEY REFERENCES DAY_PACKAGE(DID),
   PRIMARY KEY(OID, DID),
);

CREATE TABLE RECOMMENDATION ( --R16.1
   NID INT PRIMARY KEY,
   Valid_period DATE NOT NULL, -- valid period is represented as just the end date,
assume start date is date issued
   Date_Issued DATE NOT NULL, -- YYYY-MM-DD
   DID INT FOREIGN KEY REFERENCES DAY_PACKAGE(DID), -- dont need to have day package
   OID INT FOREIGN KEY REFERENCES RESTAURANT_OUTLET(OID), -- dont need to have
restaurant
   CHECK(Date_Issued <= Valid_period),

);

CREATE TABLE USER_USE_RECO( --R17
   UID INT NOT NULL FOREIGN KEY REFERENCES USER_ACCOUNT(UID),
   NID INT NOT NULL FOREIGN KEY REFERENCES RECOMMENDATION(NID),
   PRIMARY KEY(UID, NID),
);

CREATE TABLE PURCHASE_VOUCHER ( --R19
   VID INT PRIMARY KEY FOREIGN KEY REFERENCES VOUCHER(VID),
   Purchase_Discount INT NOT NULL,
   Date_time SMALLDATETIME, -- YYYY-MM-DD HH:MI:SS (if null means not used yet)
   UID INT NOT NULL FOREIGN KEY REFERENCES USER_ACCOUNT(UID), -- always tagged to a
user
   CHECK(Purchase_Discount<100 AND Purchase_Discount>0),
);

CREATE TABLE DINE_VOUCHER ( --R20
   VID INT PRIMARY KEY FOREIGN KEY REFERENCES VOUCHER(VID),
   Cash_Discount INT NOT NULL,
   Date_time SMALLDATETIME, -- YYYY-MM-DD HH:MI:SS (if null means not used yet)
   UID INT NOT NULL FOREIGN KEY REFERENCES USER_ACCOUNT(UID), -- always tagged to a
user
```

```
   CHECK(Cash_Discount>0),
);

CREATE TABLE GROUP_VOUCHER ( --R21
   VID INT PRIMARY KEY FOREIGN KEY REFERENCES VOUCHER(VID),
   Group_size INT NOT NULL,
   Group_discount INT NOT NULL,
   Date_time SMALLDATETIME, -- YYYY-MM-DD HH:MI:SS (if null means not used yet)
   UID INT NOT NULL FOREIGN KEY REFERENCES USER_ACCOUNT(UID), -- always tagged to a
user
   CHECK(Group_discount>0 AND Group_discount<100 AND Group_size>1),
);

CREATE TABLE PACKAGE_VOUCHER ( --R22
   VID INT PRIMARY KEY FOREIGN KEY REFERENCES VOUCHER(VID),
   Package_Discount INT NOT NULL,
   CHECK(Package_Discount>0 AND Package_Discount<100),
);
GO
```

Table population

```
INSERT INTO RELATED
VALUES
(0, 1,  'FAMILY'),
(0, 2,  'FAMILY'),
(1, 6,  'CLUB'),
(2, 3,  'FRIENDS'),
(2, 4,  'FRIENDS'),
(2, 7,  'CLUB'),
(3, 4,  'FRIENDS'),
(3, 6,  'CLUB'),
(4, 9,  'FAMILY'),
(5, 6,  'CLUB');

INSERT INTO MALL_MGMT_COMPANY
VALUES
(1, '2 Orchard Turn'),
(2, '27 Tiong Bahru Road'),
(4, '4 Tampines Central 5'),
(0, '638 Jurong West Street 61'),
(3, '762 Jurong West Street 75');

INSERT INTO MALL
VALUES
(0, '9 Eng Kong Terrace',   250,    0),
(1, '133 New Bridge Rd #03-08', 170,    1),
(2, '27 Lowland Road Lowland Garden',   180,    2),
(3, '41 Jln Tiga #01-05',   10, 3),
(4, '2 KALLANG AVENUE, #02-152A',   250,    4);

INSERT INTO SHOP_NOUN
VALUES
(0, 'Cafe', 0),
(1, 'Restaurant',   0),
(2, 'Cafe', 2),
(3, 'Cafe', 3),
(4, 'Restaurant',   4),
(5, 'Cafe', 0),
(6, 'Cafe', 0),
(7, 'Restaurant',   1),
(8, 'Restaurant',   1),
(9, 'Restaurant',   2),
(10,    'Restaurant',   2),
(11,    'Cafe', 3),
(12,    'Cafe', 3),
(13,    'Cafe', 4),
(14,    'Cafe', 4);

INSERT INTO COMPLAINT
VALUES
(0, 'the shop too expensive', 'pending', '2024-03-19 12:12:00', 1),
(1, 'the shop ugly', 'being handled', '2024-03-19 13:13:00', 2),
(2, 'food too expensive', 'addressed', '2012-04-20 05:33:00', 2),
(3, 'rude stuff', 'pending', '2020-05-21 08:22:00', 3),
(4, 'interior dirty', 'addressed', '2022-05-22 15:33:00', 4);

INSERT INTO COMPLAINTS_ON_SHOP
VALUES
(0, 2),
(1, 3),
(2, 1),
(3, 2),
(4, 0);

INSERT INTO RESTAURANT_CHAIN
VALUES
(3, '149 Geylang Road #02-05, singapore'),
(1, '161 Neil Road'),
(2, '17A Circular Rd'),
(0, '180 Cecil Street 14-01/04'),
(4, 'Block 28 Kallang Place #02-01 to 09');

INSERT INTO RESTAURANT_OUTLET
VALUES
(0  ,0, 0),
(1  ,0, 2),
(2  ,1, 1),
(3  ,1, 2),
(4, 2,  3),
(5, 3,  4),
(6, 2,  4),
(7, 1,  0),
(8, 2,  3);
```

```
INSERT INTO COMPLAINTS_ON_RESTAURANT
VALUES
(0, 1),
(1,  2),
(2,  2),
(3,  1),
(4,  3);

INSERT INTO SHOP_VERB
VALUES
(0, 0, 54.25, '05/12/2023:1132', '05/12/2023:1317'),
(1, 1, 60.80, '10/12/2023:1141', '10/12/2023:1256'),
(2, 2, 67.10, '15/12/2023:1158', '15/12/2023:1317'),
(3, 3, 75.50, '20/12/2023:1536', '20/12/2023:1721'),
(4, 4, 150.75, '25/12/2023:1923', '25/12/2023:1939'),
(0, 5, 50.45, '05/12/2023:1852', '05/12/2023:1947'),
(1, 6, 72.30, '10/12/2023:1959', '10/12/2023:2025'),
(2, 7, 95.05, '15/12/2023:1814', '15/12/2023:1848'),
(3, 8, 300.20, '20/12/2023:1719', '20/12/2023:1756'),
(4, 9, 100.50, '25/12/2023:1845', '25/12/2023:2000'),
(0, 6, 33.75, '01/11/2023:1520', '01/11/2023:1635'),
(1, 3, 40.55, '02/10/2023:1732', '02/10/2023:1847'),
(2, 4, 31.90, '03/09/2023:1707', '03/09/2023:1857'),
(3, 5, 97.25, '04/08/2023:1158', '04/08/2023:1333'),
(4, 6, 105.70, '05/07/2023:1146', '05/07/2023:1305'),
(0, 7, 200.20, '06/06/2023:1158', '06/06/2023:1324'),
(1, 8, 75.15, '07/05/2023:1142', '07/05/2023:1257'),
(2, 9, 70.60, '08/04/2023:1115', '08/04/2023:1245'),
(3, 3, 60.90, '09/03/2023:1132', '09/03/2023:1247'),
(4, 5, 150.30, '10/02/2023:1149', '10/02/2023:1318'),
(1, 6, 9.25, '11/12/2023:1111', '11/12/2023:1246'),
(2, 7, 150.55, '12/12/2023:1138', '12/12/2023:1304'),
(3, 8, 90.70, '13/12/2023:1127', '13/12/2023:1242'),
(4, 9, 150.95, '14/12/2023:1147', '14/12/2023:1312'),
(0, 3, 38.80, '15/12/2023:1139', '15/12/2023:1314'),
(1, 5, 75.35, '16/12/2023:1120', '16/12/2023:1247'),
(2, 6, 107.65, '17/12/2023:1158', '17/12/2023:1325'),
(3, 8, 100.15, '18/12/2023:1156', '18/12/2023:1321'),
(4, 5, 30.20, '19/12/2023:1152', '19/12/2023:1307'),
(0, 3, 150.40, '20/12/2023:1130', '20/12/2023:1314'),
(0, 3, 120.75, '05/12/2023:1417', '05/12/2023:1528'),
(1, 5, 180.60, '06/12/2023:1429', '06/12/2023:1614'),
(2, 6, 112.90, '07/12/2023:1434', '07/12/2023:1632'),
(3, 8, 150.40, '08/12/2023:1457', '08/12/2023:1632'),
(4, 3, 114.25, '09/12/2023:1423', '09/12/2023:1603'),
(0, 5, 180.50, '10/12/2023:1446', '10/12/2023:1621'),
(1, 6, 290.20, '11/12/2023:1418', '11/12/2023:1631'),
(2, 8, 32.75, '12/12/2023:1437', '12/12/2023:1616'),
(3, 3, 150.60, '13/12/2023:1442', '13/12/2023:1622'),
(4, 5, 22.30, '14/12/2023:1452', '14/12/2023:1635'),
(0, 6, 166.90, '15/12/2023:1441', '15/12/2023:1623'),
(1, 8, 98.45, '16/12/2023:1423', '16/12/2023:1633'),
(2, 3, 180.25, '17/12/2023:1436', '17/12/2023:1627');
GO

INSERT INTO DINE
VALUES
(0, 0,  50  , '2024-01-12 17:31:00', '2024-01-12 18:15:00'),(
0,  0,  85  , '2024-01-17 19:47:00', '2024-01-17 20:31:00'),(
0,  1,  80  , '2024-01-14 18:30:00', '2024-01-14 19:15:00'),(
0,  2,  0   , '2024-01-13 20:03:00', '2024-01-13 20:50:00'),(
0,  2,  30  , '2024-01-16 17:13:00', '2024-01-16 17:54:00'),(
0,  3,  0   , '2024-01-11 19:05:00', '2024-01-11 19:52:00'),(
0,  3,  100 , '2024-01-18 11:23:00', '2024-01-18 11:57:00'),(
0,  4,  65  , '2024-01-10 18:01:00', '2024-01-10 18:46:00'),(
0,  5,  60  , '2024-01-15 12:14:00', '2024-01-15 12:55:00'),(
1,  0,  35  , '2024-01-12 17:35:00', '2024-01-12 18:16:00'),(
1,  0,  0   , '2024-01-17 19:47:00', '2024-01-17 20:32:00'),(
1,  1,  45  , '2024-01-14 18:32:00', '2024-01-14 19:14:00'),(
1,  1,  120.25  , '2023-12-22 18:17:00', '2023-12-22 19:45:00'),(
1,  2,  90  , '2024-01-13 20:00:00', '2024-01-13 20:52:00'),(
1,  2,  40  , '2024-01-16 17:10:00', '2024-01-16 17:56:00'),(
1,  3,  75.5    , '2024-01-11 19:02:00', '2024-01-11 19:51:00'),(
1,  3,  112.9   , '2023-12-14 18:28:00', '2023-12-14 20:05:00'),(
1,  3,  25  , '2024-01-18 11:21:00', '2024-01-18 11:55:00'),(
1,  4,  0   , '2024-01-10 18:05:00', '2024-01-10 18:46:00'),(
1,  4,  0   , '2024-01-15 12:15:00', '2024-01-15 12:53:00'),(
2,  0,  54.75   , '2023-12-01 12:18:00', '2023-12-01 14:04:00'),(
2,  2,  10  , '2023-12-01 19:18:00', '2023-12-01 20:04:00'),(
2,  2,  22.65   , '2023-12-08 18:45:00', '2023-12-08 20:20:00'),(
2,  2,  140.5   , '2023-12-17 17:30:00', '2023-12-17 18:50:00'),(
2,  2,  112.3   , '2023-12-22 17:30:00', '2023-12-22 18:50:00'),(
2,  2,  170.9   , '2023-12-23 12:34:00', '2023-12-23 14:08:00'),(
2,  2,  112.3   , '2023-12-27 17:30:00', '2023-12-27 18:50:00'),(
2,  2,  47.65   , '2023-12-28 18:45:00', '2023-12-28 20:20:00'),(
2,  3,  120.5   , '2023-12-03 12:05:00', '2023-12-03 13:30:00'),(
2,  3,  95.2    , '2023-12-13 12:05:00', '2023-12-13 13:30:00'),(
2,  4,  20.65   , '2023-12-09 18:15:00', '2023-12-09 19:30:00'),(
2,  4,  114.25  , '2023-12-15 11:51:00', '2023-12-15 13:27:00'),(
3,  0,  38.8    , '2023-12-16 11:39:00', '2023-12-16 13:14:00'),(
3,  0,  150.4   , '2023-12-21 17:30:00', '2023-12-21 19:06:00'),(
3,  1,  50.75   , '2023-12-01 12:25:00', '2023-12-01 13:50:00'),(
3,  1,  90.2    , '2023-12-02 17:42:00', '2023-12-02 19:15:00'),(
3,  1,  70.2    , '2023-12-16 17:40:00', '2023-12-16 18:55:00'),(
3,  1,  95.8    , '2023-12-21 17:45:00', '2024-12-21 20:31:00'),(
3,  1,  95.8    , '2023-12-26 17:45:00', '2023-12-26 19:00:00'),(
3,  2,  105.8   , '2023-12-07 17:20:00', '2023-12-07 18:50:00'),(
3,  2,  82.3    , '2023-12-12 17:30:00', '2023-12-12 18:50:00'),(
3,  4,  30.2    , '2023-12-20 11:52:00', '2023-12-20 13:07:00'),(
4,  0,  40.3    , '2023-12-05 12:35:00', '2023-12-05 13:50:00'),(
4,  0,  55.7    , '2024-02-11 19:12:00', '2024-02-11 20:07:00'),(
4,  0,  51.7    , '2024-02-23 19:22:00', '2024-02-23 20:37:00'),(
4,  1,  67.5    , '2023-12-11 17:45:00', '2023-12-11 19:00:00'),(
4,  1,  10  , '2024-02-12 17:38:00', '2024-02-12 18:22:00'),(
4,  1,  33  , '2024-02-14 19:39:00', '2024-02-14 20:42:00'),(
4,  1,  17  , '2024-02-24 17:31:00', '2024-02-24 18:42:00'),(
4,  1,  18  , '2024-02-29 18:33:00', '2024-02-29 19:52:00'),(
4,  2,  130.5   , '2023-12-03 18:13:00', '2023-12-03 19:49:00'),(
4,  2,  0   , '2024-02-13 20:03:00', '2024-02-13 21:02:00'),(
4,  2,  0   , '2024-02-28 20:13:00', '2024-02-28 21:22:00'),(
4,  3,  97.9    , '2023-12-09 11:32:00', '2023-12-09 12:56:00'),(
4,  3,  180.2   , '2023-12-18 12:05:00', '2023-12-18 13:30:00'),(
4,  3,  70.2    , '2023-12-23 12:00:00', '2023-12-23 13:30:00'),(
4,  3,  130.75  , '2023-12-24 11:58:00', '2023-12-24 13:33:00'),(
4,  3,  70.2    , '2023-12-28 12:00:00', '2023-12-28 13:30:00'),(
4,  3,  102.9   , '2023-12-29 11:32:00', '2023-12-29 12:56:00'),(
4,  4,  0   , '2024-02-10 18:08:00', '2024-02-10 18:56:00'),(
4,  4,  0   , '2024-02-15 12:43:00', '2024-02-15 13:39:00'),(
4,  4,  33  , '2024-02-22 13:05:00', '2024-02-22 13:57:00'),(
4,  4,  28  , '2024-02-28 12:49:00', '2024-02-28 13:59:00'),(
5,  1,  75.35   , '2023-12-17 11:20:00', '2023-12-17 12:47:00'),(
5,  4,  76.3    , '2023-12-05 19:47:00', '2023-12-05 21:15:00'),(
6,  0,  97.9    , '2023-12-10 12:00:00', '2023-12-10 13:30:00'),(
6,  0,  42.45   , '2023-12-15 12:25:00', '2023-12-15 13:45:00'),(
6,  1,  62.45   , '2023-12-06 18:02:00', '2023-12-06 19:27:00'),(
6,  2,  80.2    , '2023-12-02 17:58:00', '2023-12-02 19:15:00'),(
6,  2,  107.65  , '2023-12-18 17:58:00', '2023-12-18 19:25:00'),(
6,  3,  120.75  , '2023-12-04 11:58:00', '2023-12-04 13:32:00'),(
6,  4,  170.5   , '2023-12-10 17:28:00', '2023-12-10 19:03:00'),(
6,  4,  130.65  , '2023-12-19 18:15:00', '2023-12-19 19:30:00'),(
6,  4,  130.65  , '2023-12-24 18:15:00', '2023-12-24 19:30:00'),(
6,  4,  90.5    , '2023-12-25 19:47:00', '2023-12-25 21:13:00'),(
6,  4,  130.65  , '2023-12-29 18:15:00', '2023-12-29 19:30:00'),(
6,  4,  180.5   , '2023-12-30 17:28:00', '2023-12-30 19:03:00'),(
7,  0,  42.75   , '2023-12-11 11:11:00', '2023-12-11 12:36:00'),(
8,  0,  62.45   , '2023-12-06 12:37:00', '2023-12-06 14:02:00'),(
8,  0,  170.9   , '2023-12-20 12:00:00', '2023-12-20 13:30:00'),(
8,  0,  170.9   , '2023-12-25 12:00:00', '2023-12-25 13:30:00'),(
8,  0,  77.25   , '2023-12-26 12:37:00', '2023-12-26 14:02:00'),(
8,  0,  170.9   , '2023-12-30 12:00:00', '2023-12-30 13:30:00'),(
8,  1,  85.2    , '2023-12-12 12:49:00', '2023-12-12 14:17:00'),(
8,  3,  130.1   , '2023-12-08 12:30:00', '2023-12-08 14:00:00'),(
8,  3,  100.15  , '2023-12-19 12:56:00', '2023-12-19 14:21:00'),(
8,  4,  90.75   , '2023-12-04 17:32:00', '2023-12-04 18:49:00'),(
8,  4,  110.75  , '2023-12-14 18:00:00', '2023-12-14 19:15:00'),(
9,  0,  0   , '2024-02-11 19:15:00', '2024-02-11 20:02:00'),(
9,  0,  0   , '2024-02-23 19:25:00', '2024-02-23 20:32:00'),(
9,  1,  150.8   , '2023-12-07 18:19:00', '2023-12-07 19:48:00'),(
9,  1,  25  , '2024-02-12 17:39:00', '2024-02-12 18:24:00'),(
9,  1,  42  , '2024-02-14 19:35:00', '2024-02-14 20:45:00'),(
9,  1,  32  , '2024-02-24 17:33:00', '2024-02-24 18:44:00'),(
9,  1,  130.8   , '2023-12-27 17:19:00', '2023-12-27 18:49:00'),(
9,  1,  22  , '2024-02-29 18:35:00', '2024-02-29 19:51:00'),(
9,  2,  40.2    , '2024-02-13 20:00:00', '2024-02-13 21:05:00'),(
9,  2,  40.65   , '2023-12-13 17:17:00', '2023-12-13 18:42:00'),(
9,  2,  43.4    , '2024-02-28 20:10:00', '2024-02-28 21:15:00'),(
9,  4,  45  , '2024-02-10 18:04:00', '2024-02-10 18:56:00'),(
9,  4,  32.6    , '2024-02-15 12:44:00', '2024-02-15 13:33:00'),(
9,  4,  25  , '2024-02-22 13:07:00', '2024-02-22 13:56:00'),(
9,  4,  32.3    , '2024-02-28 12:46:00', '2024-02-28 13:53:00'),(
10, 2,  33.5    , '2024-12-31 19:12:00', '2024-12-31 20:00:00');

SET IDENTITY_INSERT DAY_PACKAGE ON;

INSERT INTO DAY_PACKAGE
   (DID, Description, UID, VID)
VALUES
   (0, 'Package 1', 0, 0);
```

```
SET IDENTITY_INSERT DAY_PACKAGE OFF;

INSERT INTO DAY_PACKAGE (Description, UID, VID)
VALUES
('Package 1', 1, 1),
('Package 1', 1, 2),
('Package 1', 2, 3),
('Package 2', 3, 4),
('Package 2', 4, 5),
('Package 2', 5, 6),
('Package 3', 6, 7),
('Package 3', 7, 8),
('Package 3', 8, 9),
('Package 1', 1, 10),
('Package 1', 1, 11),
('Package 1', 1, 12),
('Package 1', 2, 13),
('Package 2', 3, 14),
('Package 2', 4, 15),
('Package 2', 5, 16),
('Package 3', 6, 17),
('Package 3', 7, 18),
('Package 3', 8, 19),
('Package 1', 1, 20);

INSERT INTO PACKAGE_VOUCHER (VID, Package_Discount)
VALUES
(0, 41),
(1, 66),
(2, 27),
(3, 32),
(4, 60),
(5, 56),
(6, 26),
(7, 30),
(8, 38),
(9, 58),
(10, 22),
(11, 41),
(12, 12),
(13, 22),
(14, 23),
(15, 50),
(16, 23),
(17, 45),
(18, 24),
(19, 5),
(20, 12);

INSERT INTO DINE_VOUCHER (VID, Cash_Discount, Date_time, UID)
VALUES
(21, 5, '2024-02-11 20:02:00', 0),
(30, 5, '2024-01-12 17:31:00', 1),
(22, 5, '2024-01-12 17:35:00', 1),
(23, 5, '2023-12-01 19:18:00', 2),
(24, 5, '2023-12-21 17:30:00', 3),
(25, 5, '2023-12-18 12:05:00', 4),
(26, 5, '2023-12-17 11:20:00', 5),
(27, 5, '2023-12-10 17:28:00', 6),
(28, 5, '2023-12-11 11:11:00', 7),
(29, 5, '2023-12-08 14:00:00', 8),
(30, 5, '2024-02-11 20:02:00', 9),
(31, 5, '2024-01-13 20:52:00', 1),
(32, 5, '2023-12-03 13:30:00', 2),
(33, 5, '2023-12-02 19:15:00', 3),
(34, 5, '2024-02-11 20:07:00', 4),
(35, 5, '2023-12-05 21:15:00', 5),
(36, 5, '2023-12-24 19:30:00', 6),
(37, 5, '2023-12-11 12:36:00', 7),
(38, 5, '2023-12-19 14:21:00', 8),
(39, 5, '2023-12-27 18:49:00', 9),
(40, 5, '2024-01-18 11:57:00', 0),
(41, 5, '2023-12-22 19:45:00', 1),
(42, 5, '2023-12-08 20:20:00', 2),
(43, 5, '2023-12-21 19:06:00', 3),
(44, 5, '2024-02-12 18:22:00', 4),
(45, 5, '2023-12-05 21:15:00', 5),
(46, 5, '2023-12-25 21:13:00', 6),
(47, 5, '2023-12-11 12:36:00', 7),
(48, 5, '2023-12-19 14:21:00', 8),
(49, 5, '2024-02-12 18:24:00', 9),
(50, 5, '2024-01-18 11:55:00', 1);
```

```sql
INSERT INTO GROUP_VOUCHER (VID, Group_size, Group_discount, Date_time, UID)
VALUES
(51, 5, 15, '2023-12-11 20:02:00', 2),
(52, 3, 10, '2023-12-12 14:30:00', 8),
(53, 7, 20, '2023-12-15 08:45:00', 4),
(54, 2, 5, '2023-12-20 18:10:00', 6),
(55, 6, 18, '2023-12-22 12:20:00', 3),
(56, 4, 12, '2023-12-25 09:55:00', 1),
(57, 2, 4, '2023-12-28 16:40:00', 5),
(58, 8, 25, '2024-01-02 21:15:00', 9),
(59, 5, 15, '2024-01-06 10:30:00', 7),
(60, 3, 10, '2024-01-10 14:00:00', 0),
(61, 7, 20, '2024-01-15 08:00:00', 10),
(62, 2, 5, '2024-01-20 18:30:00', 2),
(63, 6, 18, '2024-01-24 12:15:00', 8),
(64, 4, 12, '2024-01-28 09:45:00', 4),
(65, 2, 4, '2024-02-02 16:50:00', 6),
(66, 8, 25, '2024-02-06 21:20:00', 3),
(67, 5, 15, '2024-02-10 10:35:00', 1),
(68, 3, 10, '2024-02-14 14:10:00', 5),
(69, 7, 20, '2024-02-19 08:05:00', 9),
(70, 2, 5, '2024-02-23 18:40:00', 7);

INSERT INTO PURCHASE_VOUCHER (VID, Purchase_Discount, Date_time, UID)
VALUES
(71, 5, '2023-12-11 20:02:00', 1),
(72, 10, '2023-12-12 12:30:00', 3),
(73, 15, '2023-12-15 08:45:00', 5),
(74, 20, '2023-12-20 18:10:00', 7),
(75, 25, '2023-12-22 12:20:00', 9),
(76, 30, '2023-12-25 09:55:00', 2),
(77, 35, '2023-12-28 16:40:00', 4),
(78, 40, '2024-01-02 21:15:00', 6),
(79, 45, '2024-01-06 10:30:00', 8),
(80, 50, '2024-01-10 14:00:00', 10),
(81, 55, '2024-01-15 08:00:00', 0),
(82, 60, '2024-01-20 18:30:00', 1),
(83, 65, '2024-01-24 12:15:00', 3),
(84, 70, '2024-01-28 09:45:00', 5),
(85, 12, '2024-02-02 16:50:00', 7),
(86, 18, '2024-02-06 21:20:00', 9),
(87, 22, '2024-02-10 10:35:00', 2),
(88, 28, '2024-02-14 14:10:00', 4),
(89, 32, '2024-02-19 08:05:00', 6),
(90, 38, '2024-02-23 18:40:00', 8),
(91, 42, '2024-02-27 12:25:00', 10),
(92, 48, '2024-03-02 16:55:00', 0),
(93, 52, '2024-03-06 21:30:00', 1),
(94, 58, '2024-03-10 10:40:00', 3),
(95, 62, '2024-03-14 14:20:00', 5),
(96, 68, '2024-03-18 08:10:00', 7),
(97, 72, '2024-03-22 18:45:00', 9),
(98, 78, '2024-03-26 12:30:00', 2),
(99, 80, '2024-03-30 16:59:00', 4);

INSERT INTO MALL_HAS_PACKAGE (MID, DID)
VALUES
(0, 0),
(1, 0),
(2, 0),
(0, 1),
(1, 1),
(2, 1),
(0, 2),
(1, 2),
(2, 2),
(0, 3),
(1, 3),
(2, 3),
(0, 10),
(1, 10),
(2, 10),
(0, 11),
(1, 11),
(2, 11),
(0, 12),
(1, 12),
(2, 12),
(0, 13),
(1, 13),
(2, 13),
(1, 4),
(2, 4),
(3, 4),
(1, 5),
(2, 5),
(3, 5),
(1, 6),
(2, 6),
(3, 6),
(1, 14),
(2, 14),
(3, 14),
(1, 15),
(2, 15),
(3, 15),
(1, 16),
(2, 16),
(3, 16),
(2, 7),
(3, 7),
(4, 7),
(2, 8),
(3, 8),
(4, 8),
(2, 9),
(3, 9),
(4, 9),
(2, 17),
(3, 17),
(4, 17),
(2, 18),
(3, 18),
(4, 18),
(2, 19),
(3, 19),
(4, 19),
(0, 20),
(1, 20),
(2, 20);

INSERT INTO PACKAGE_HAS_RESTAURANT (OID, DID)
VALUES
(0, 0),
(0, 1),
(0, 2),
(0, 3),
(0, 10),
(0, 11),
(0, 12),
(0, 13),
(0, 20),
(1, 0),
(1, 1),
(1, 2),
(1, 3),
(1, 10),
(1, 11),
(1, 12),
(1, 13),
(1, 20),
(2, 0),
(2, 1),
(2, 2),
(2, 3),
(2, 10),
(2, 11),
(2, 12),
(2, 13),
(2, 20),
(3, 4),
(3, 5),
(3, 6),
(3, 14),
(3, 15),
(3, 16),
(4, 4),
(4, 5),
(4, 6),
(4, 14),
(4, 15),
(4, 16),
(5, 7),
(5, 8),
(5, 9),
(5, 17),
(5, 18),
(5, 19),
(6, 7),
(6, 8),
(6, 9),
(6, 17),
(6, 18),
(6, 19),
(7, 7),
(7, 8),
(7, 9),
(7, 17),
(7, 18),
(7, 19),
(8, 7),
(8, 8),
(8, 9),
(8, 17),
(8, 18),
(8, 19);

INSERT INTO RECOMMENDATION (Valid_period, Date_Issued, DID, OID) VALUES
('2022-12-01', '2022-06-01', NULL, 5),
('2022-11-01', '2022-05-01', NULL, 2),
('2023-07-01', '2023-01-01', NULL, 6),
('2023-09-01', '2023-03-01', NULL, 2),
('2024-02-01', '2023-08-01', NULL, 7),
('2023-10-01', '2023-04-01', NULL, 5),
('2022-11-01', '2022-05-01', NULL, 4),
('2022-09-01', '2022-03-01', NULL, 3),
('2023-05-01', '2022-11-01', NULL, 8),
('2024-08-01', '2024-02-01', NULL, 0),
('2022-11-01', '2022-05-01', NULL, 4),
('2023-01-01', '2022-07-01', NULL, 2),
('2023-05-01', '2022-11-01', NULL, 1),
('2024-08-01', '2024-02-01', NULL, 2),
('2024-04-01', '2023-10-01', NULL, 7),
('2023-06-01', '2022-12-01', NULL, 6),
('2023-08-01', '2023-02-01', NULL, 4),
('2022-08-01', '2022-02-01', NULL, 1),
('2024-01-01', '2023-07-01', NULL, 3),
('2023-09-01', '2023-03-01', NULL, 8),
('2023-05-01', '2022-11-01', NULL, 2),
('2022-12-01', '2022-06-01', NULL, 4),
('2023-04-01', '2022-10-01', NULL, 5),
('2023-07-01', '2023-01-01', NULL, 7),
('2023-03-01', '2022-09-01', NULL, 3),
('2022-11-01', '2022-05-01', NULL, 6),
('2023-02-01', '2022-08-01', NULL, 7),
('2022-10-01', '2022-04-01', NULL, 1),
('2023-03-01', '2022-09-01', NULL, 8),
('2023-08-01', '2023-02-01', NULL, 2),
('2023-06-01', '2022-12-01', NULL, 0),
('2023-02-01', '2022-08-01', NULL, 1),
('2023-07-01', '2023-01-01', NULL, 3),
('2023-04-01', '2022-10-01', NULL, 4),
('2023-10-01', '2023-04-01', NULL, 6),
('2022-11-01', '2022-05-01', NULL, 0),
('2023-05-01', '2022-11-01', NULL, 5),
('2022-12-01', '2022-06-01', NULL, 1),
('2024-02-01', '2023-08-01', NULL, 3),
('2023-01-01', '2022-07-01', NULL, 7),
('2023-03-01', '2022-09-01', NULL, 8),
('2023-09-01', '2023-03-01', NULL, 1),
('2023-08-01', '2023-02-01', NULL, 5),
('2023-06-01', '2022-12-01', NULL, 3),
('2022-09-01', '2022-03-01', NULL, 0),
('2023-07-01', '2023-01-01', NULL, 8),
('2024-04-01', '2023-10-01', NULL, 2),
('2023-05-01', '2022-11-01', NULL, 6),
('2023-08-01', '2023-02-01', NULL, 7),
('2024-05-01', '2023-11-01', NULL, 4),
('2022-11-01', '2022-05-01', NULL, 3),
('2023-07-01', '2023-01-01', NULL, 6),
('2023-09-01', '2023-03-01', NULL, 2),
('2024-02-01', '2023-08-01', NULL, 0),
('2023-10-01', '2023-04-01', NULL, 5),
('2022-11-01', '2022-05-01', NULL, 1),
('2022-09-01', '2022-03-01', NULL, 8),
('2023-05-01', '2022-11-01', NULL, 2),
('2024-08-01', '2024-02-01', NULL, 7),
('2022-11-01', '2022-05-01', NULL, 5),
('2023-01-01', '2022-07-01', NULL, 4),
('2023-05-01', '2022-11-01', NULL, 0),
('2024-08-01', '2024-02-01', NULL, 6),
('2024-04-01', '2023-10-01', NULL, 8),
('2023-06-01', '2022-12-01', NULL, 1),
('2023-08-01', '2023-02-01', NULL, 3),
('2022-08-01', '2022-02-01', NULL, 6),
('2024-01-01', '2023-07-01', NULL, 4),
('2023-09-01', '2023-03-01', NULL, 2),
('2023-05-01', '2022-11-01', NULL, 0),
('2022-12-01', '2022-06-01', NULL, 3),
('2023-04-01', '2022-10-01', NULL, 7),
('2023-07-01', '2023-01-01', NULL, 1),
('2023-03-01', '2022-09-01', NULL, 8),
('2022-11-01', '2022-05-01', NULL, 5),
('2023-02-01', '2022-08-01', NULL, 1),
('2022-10-01', '2022-04-01', NULL, 4),
('2023-03-01', '2022-09-01', NULL, 0),
('2023-08-01', '2023-02-01', NULL, 6),
('2023-06-01', '2022-12-01', NULL, 2),
('2023-02-01', '2022-08-01', NULL, 4),
('2023-07-01', '2023-01-01', NULL, 8),
('2023-04-01', '2022-10-01', NULL, 7),
('2023-10-01', '2023-04-01', NULL, 3),
('2022-11-01', '2022-05-01', NULL, 5),
('2023-05-01', '2022-11-01', NULL, 1),
('2022-12-01', '2022-06-01', NULL, 2),
('2024-02-01', '2023-08-01', NULL, 6),
('2023-01-01', '2022-07-01', NULL, 3),
('2023-03-01', '2022-09-01', NULL, 7),
('2023-09-01', '2023-03-01', NULL, 0),
('2023-08-01', '2023-02-01', NULL, 5),
('2023-06-01', '2022-12-01', NULL, 1),
('2022-09-01', '2022-03-01', NULL, 4),
('2023-07-01', '2023-01-01', NULL, 8),
('2024-04-01', '2023-10-01', NULL, 2),
('2023-05-01', '2022-11-01', NULL, 6),
('2022-09-06', '2022-03-06', 0, NULL),
('2023-08-17', '2023-02-17', 1, NULL),
('2023-04-18', '2022-10-18', 2, NULL),
('2023-07-20', '2023-01-20', 3, NULL),
('2024-07-17', '2024-01-17', 4, NULL),
('2023-03-20', '2022-09-20', 5, NULL),
('2022-08-16', '2022-02-16', 6, NULL),
('2023-06-21', '2022-12-21', 7, NULL),
('2023-10-23', '2023-04-23', 8, NULL),
('2022-09-14', '2022-03-14', 9, NULL),
('2023-01-07', '2022-07-07', 10, NULL),
('2023-04-17', '2022-10-17', 11, NULL),
('2023-04-26', '2022-10-26', 12, NULL),
('2023-02-21', '2022-08-21', 13, NULL),
('2023-08-14', '2023-02-14', 14, NULL),
('2023-10-06', '2023-04-06', 15, NULL),
('2023-01-16', '2022-07-16', 16, NULL),
('2023-09-07', '2023-03-07', 17, NULL),
('2024-01-24', '2023-07-24', 18, NULL),
('2024-01-18', '2023-07-18', 19, NULL),
('2022-10-07', '2022-04-07', 20, NULL);

INSERT INTO USER_USE_RECO(UID,NID) VALUES
(26, 47),
(15, 68),
(8, 21),
(44, 39),
(3, 5),
(48, 89),
(17, 12),
(30, 72),
(10, 60),
(20, 35),
(39, 18),
(13, 55),
(22, 87),
(42, 49),
(36, 31),
(6, 76),
(28, 2),
(19, 29),
(49, 9),
(1, 80),
(33, 63),
(0, 41),
(14, 85),
(7, 17),
(43, 69),
(45, 48),
(21, 26),
(27, 96),
(23, 74),
(4, 50),
(35, 23),
(24, 56),
(32, 91),
(29, 16),
(34, 43),
(9, 1),
(46, 78),
(16, 33),
(11, 19),
(5, 67),
(18, 42),
(47, 22),
(37, 90),
(25, 13),
(40, 65),
(38, 36),
(41, 75),
(31, 6),
(12, 82),
(50, 44),
(0, 20),
(3, 62),
(6, 3),
(8, 79),
(21, 52),
(29, 30),
(33, 95),
(9, 27),
(1, 81),
(14, 11),
(44, 71),
(16, 46),
(10, 10),
(36, 86),
(2, 15),
(28, 70),
(13, 40),
(20, 32),
(47, 57),
(17, 93),
(26, 24),
(25, 73),
(18, 51),
(42, 37),
(23, 7),
(37, 83),
(24, 45),
(30, 25),
(45, 64),
(19, 38),
(38, 94),
(48, 28),
(4, 53),
(22, 14),
(15, 84),
(32, 58),
(5, 4),
(39, 77),
(34, 34),
(46, 66),
(7, 9),
(27, 61),
(50, 0),
(33, 88),
(10, 77),
(16, 47),
(27, 94),
(40, 85),
(45, 25),
(19, 26),
(17, 40),
(22, 68),
(49, 51),
(36, 93),
(6, 19),
(44, 67),
(11, 84),
(34, 3),
(8, 16),
(39, 62),
(42, 29),
(48, 79),
(12, 36),
(14, 65),
(31, 44),
(2, 8),
(21, 30),
(20, 18),
(46, 64),
(23, 54),
(41, 24),
(50, 37),
(26, 75),
(32, 5),
(35, 28),
(4, 7),
(30, 2),
(15, 50),
(29, 9),
(24, 92),
(38, 15),
(47, 11),
(25, 10),
(18, 43),
(7, 87),
(3, 33),
(28, 12),
(13, 31),
(1, 49),
(5, 1),
(37, 56),
(43, 78),
(9, 89),
(0, 70),
(20, 55),
(48, 14),
(29, 96),
(14, 71),
(39, 86),
(23, 72),
(35, 59),
(17, 95),
(36, 61),
(11, 73),
(3, 4),
(19, 46),
(15, 66),
(27, 53),
(2, 34),
(49, 74),
(37, 63),
(1, 90),
(16, 32),
(44, 45),
(31, 42),
(45, 57),
(18, 35),
(24, 60),
(28, 97),
(10, 80),
(41, 22),
(22, 41),
(47, 81),
(30, 58),
(5, 13),
(33, 39),
(8, 83),
(25, 98),
(40, 69),
(12, 6),
(21, 23),
(50, 91),
(9, 38);


INSERT INTO VOUCHER
VALUES
(0,'REDEEMED', '2024/08/13 13:42:00','2023/12/09 13:42:00','Description 1'),
(1,'ALLOCATED', '2024/06/01 13:42:00','2023/12/21 13:42:00','Description 2'),
(2,'REDEEMED', '2024/08/16 13:42:00','2023/12/09 13:42:00','Description 3'),
(3,'REDEEMED', '2024/06/19 13:42:00','2024/01/10 13:42:00','Description 4'),
(4,'ALLOCATED', '2024/09/09 13:42:00','2023/09/26 13:42:00','Description 5'),
(5,'ALLOCATED', '2025/01/13 13:42:00','2023/12/09 13:42:00','Description 6'),
(6,'REDEEMED', '2024/12/16 13:42:00','2024/03/29 13:42:00','Description 7'),
(7,'EXPIRED', '2024/11/16 13:42:00','2023/11/22 13:42:00','Description 8'),
(8,'REDEEMED', '2024/05/10 13:42:00','2023/08/24 13:42:00','Description 9'),
(9,'REDEEMED', '2024/04/27 13:42:00','2024/02/06 13:42:00', 'Description 10'),
(10,'REDEEMED', '2024/08/12 13:42:00','2023/04/23 13:42:00', 'Description 11'),
(11,'REDEEMED', '2024/10/17 13:42:00','2023/08/20 13:42:00', 'Description 12'),
(12,'ALLOCATED', '2025/02/21 13:42:00','2023/08/19 13:42:00', 'Description 13'),
(13,'REDEEMED', '2024/12/23 13:42:00','2023/11/26 13:42:00', 'Description 14'),
(14,'EXPIRED', '2024/05/02 13:42:00','2023/11/22 13:42:00', 'Description 15'),
(15,'ALLOCATED', '2024/05/17 13:42:00','2024/02/07 13:42:00', 'Description 16'),
(16,'EXPIRED', '2024/05/31 13:42:00','2023/10/11 13:42:00', 'Description 17'),
(17,'EXPIRED', '2024/11/01 13:42:00','2023/08/26 13:42:00', 'Description 18'),
(18,'EXPIRED', '2025/01/17 13:42:00','2023/10/23 13:42:00', 'Description 19'),
(19,'EXPIRED', '2024/07/07 13:42:00','2023/09/15 13:42:00', 'Description 20'),
(20,'ALLOCATED', '2024/09/18 13:42:00','2023/07/28 13:42:00', 'Description 21'),
(21,'ALLOCATED', '2025/03/25 13:42:00','2023/05/12 13:42:00', 'Description 22'),
(22,'EXPIRED', '2024/10/30 13:42:00','2023/09/10 13:42:00', 'Description 23'),
(23,'EXPIRED', '2024/05/04 13:42:00','2023/09/29 13:42:00', 'Description 24'),
(24,'ALLOCATED', '2024/11/12 13:42:00','2023/11/29 13:42:00', 'Description 25'),
(25,'ALLOCATED', '2024/07/09 13:42:00','2023/07/20 13:42:00', 'Description 26'),
(26,'ALLOCATED', '2024/08/08 13:42:00','2023/06/11 13:42:00', 'Description 27'),
(27,'REDEEMED', '2024/12/12 13:42:00','2023/06/25 13:42:00', 'Description 28'),
(28,'REDEEMED', '2024/04/04 13:42:00','2023/05/11 13:42:00', 'Description 29'),
(29,'REDEEMED', '2024/11/03 13:42:00','2023/12/24 13:42:00', 'Description 30'),
(30,'EXPIRED', '2024/12/10 13:42:00','2023/09/15 13:42:00', 'Description 31'),
(31,'REDEEMED', '2025/03/24 13:42:00','2023/08/04 13:42:00', 'Description 32'),
(32,'EXPIRED', '2025/01/07 13:42:00','2023/11/08 13:42:00', 'Description 33'),
(33,'EXPIRED', '2024/10/23 13:42:00','2023/12/25 13:42:00', 'Description 34'),
(34,'REDEEMED', '2024/04/14 13:42:00','2023/09/12 13:42:00', 'Description 35'),
(35,'EXPIRED', '2025/03/11 13:42:00','2023/11/28 13:42:00', 'Description 36'),
(36,'REDEEMED', '2024/05/11 13:42:00','2023/09/26 13:42:00', 'Description 37'),
(37,'REDEEMED', '2024/09/25 13:42:00','2023/05/10 13:42:00', 'Description 38'),
(38,'ALLOCATED', '2024/06/01 13:42:00','2023/04/17 13:42:00', 'Description 39'),
(39,'REDEEMED', '2025/03/17 13:42:00','2023/07/16 13:42:00', 'Description 40'),
(40,'ALLOCATED', '2024/11/16 13:42:00','2023/11/04 13:42:00', 'Description 41'),
(41,'REDEEMED', '2024/11/26 13:42:00','2023/07/23 13:42:00', 'Description 42'),
(42,'REDEEMED', '2024/05/21 13:42:00','2023/04/20 13:42:00', 'Description 43'),
(43,'ALLOCATED', '2024/07/04 13:42:00','2023/11/05 13:42:00', 'Description 44'),
(44,'ALLOCATED', '2024/11/07 13:42:00','2023/11/04 13:42:00', 'Description 45'),
(45,'REDEEMED', '2024/11/16 13:42:00','2023/04/14 13:42:00', 'Description 46'),
(46,'EXPIRED', '2024/11/22 13:42:00','2023/09/14 13:42:00', 'Description 47'),
(47,'ALLOCATED', '2024/08/20 13:42:00','2023/05/28 13:42:00', 'Description 48'),
(48,'EXPIRED', '2024/09/17 13:42:00','2023/11/24 13:42:00', 'Description 49'),
(49,'EXPIRED', '2024/04/28 13:42:00','2024/02/21 13:42:00', 'Description 50'),
(50,'EXPIRED', '2024/05/05 13:42:00','2023/04/28 13:42:00', 'Description 51'),
(51,'ALLOCATED', '2024/04/12 13:42:00','2023/04/06 13:42:00', 'Description 52'),
(52,'EXPIRED', '2024/09/06 13:42:00','2023/11/23 13:42:00', 'Description 53'),
(53,'ALLOCATED', '2025/02/20 13:42:00','2023/07/08 13:42:00', 'Description 54'),
(54,'EXPIRED', '2024/10/03 13:42:00','2023/11/01 13:42:00', 'Description 55'),
(55,'REDEEMED', '2024/11/28 13:42:00','2023/06/07 13:42:00', 'Description 56'),
(56,'REDEEMED', '2024/06/24 13:42:00','2023/12/23 13:42:00', 'Description 57'),
(57,'ALLOCATED', '2025/01/08 13:42:00','2023/07/12 13:42:00', 'Description 58'),
(58,'REDEEMED', '2024/06/04 13:42:00','2023/07/23 13:42:00', 'Description 59'),
(59,'REDEEMED', '2024/09/16 13:42:00','2023/11/08 13:42:00', 'Description 60'),
(60,'EXPIRED', '2024/05/14 13:42:00','2023/04/20 13:42:00', 'Description 61'),
(61,'ALLOCATED', '2024/05/31 13:42:00','2023/12/10 13:42:00', 'Description 62'),
(62,'EXPIRED', '2024/08/11 13:42:00','2024/01/31 13:42:00', 'Description 63'),
(63,'EXPIRED', '2025/01/08 13:42:00','2024/03/29 13:42:00', 'Description 64'),
(64,'EXPIRED', '2024/04/27 13:42:00','2023/06/01 13:42:00', 'Description 65'),
(65,'EXPIRED', '2024/07/16 13:42:00','2023/07/21 13:42:00', 'Description 66'),
(66,'EXPIRED', '2024/05/13 13:42:00','2023/11/12 13:42:00', 'Description 67'),
(67,'EXPIRED', '2025/03/05 13:42:00','2024/02/23 13:42:00', 'Description 68'),
(68,'EXPIRED', '2025/01/10 13:42:00','2023/10/23 13:42:00', 'Description 69'),
(69,'ALLOCATED', '2025/02/24 13:42:00','2024/03/22 13:42:00', 'Description 70'),
(70,'REDEEMED', '2024/08/05 13:42:00','2023/08/11 13:42:00', 'Description 71'),
(71,'EXPIRED', '2025/03/13 13:42:00','2023/12/23 13:42:00', 'Description 72'),
(72,'EXPIRED', '2024/11/16 13:42:00','2023/11/05 13:42:00', 'Description 73'),
(73,'EXPIRED', '2024/10/06 13:42:00','2023/06/01 13:42:00', 'Description 74'),
(74,'EXPIRED', '2024/11/10 13:42:00','2023/09/05 13:42:00', 'Description 75'),
(75,'EXPIRED', '2025/01/07 13:42:00','2023/11/21 13:42:00', 'Description 76'),
(76,'EXPIRED', '2024/07/29 13:42:00','2023/10/31 13:42:00', 'Description 77'),
(77,'REDEEMED', '2024/07/24 13:42:00','2024/02/06 13:42:00', 'Description 78'),
(78,'REDEEMED', '2024/11/18 13:42:00','2023/12/02 13:42:00', 'Description 79'),
(79,'EXPIRED', '2024/12/14 13:42:00','2023/10/22 13:42:00', 'Description 80'),
(80,'REDEEMED', '2025/03/24 13:42:00','2023/12/06 13:42:00', 'Description 81'),
(81,'ALLOCATED', '2024/10/14 13:42:00','2024/01/16 13:42:00', 'Description 82'),
(82,'EXPIRED', '2024/08/14 13:42:00','2023/12/09 13:42:00', 'Description 83'),
(83,'ALLOCATED', '2025/02/11 13:42:00','2023/12/18 13:42:00', 'Description 84'),
(84,'ALLOCATED', '2024/09/11 13:42:00','2023/04/26 13:42:00', 'Description 85'),
(85,'EXPIRED', '2024/11/10 13:42:00','2023/07/21 13:42:00', 'Description 86'),
(86,'REDEEMED', '2025/01/31 13:42:00','2023/08/17 13:42:00', 'Description 87'),
(87,'ALLOCATED', '2024/11/19 13:42:00','2023/12/10 13:42:00', 'Description 88'),
(88,'EXPIRED', '2024/09/18 13:42:00','2023/12/23 13:42:00', 'Description 89'),
(89,'ALLOCATED', '2024/08/29 13:42:00','2023/10/30 13:42:00', 'Description 90'),
(90,'ALLOCATED', '2025/02/08 13:42:00','2023/04/17 13:42:00', 'Description 91'),
(91,'ALLOCATED', '2024/09/06 13:42:00','2023/04/28 13:42:00', 'Description 92'),
(92,'ALLOCATED', '2024/07/04 13:42:00','2023/09/19 13:42:00', 'Description 93'),
(93,'REDEEMED', '2024/11/18 13:42:00','2023/05/26 13:42:00', 'Description 94'),
(94,'ALLOCATED', '2024/09/26 13:42:00','2023/10/20 13:42:00', 'Description 95'),
(95,'ALLOCATED', '2024/07/21 13:42:00','2023/09/05 13:42:00', 'Description 96'),
(96,'ALLOCATED', '2025/02/11 13:42:00','2023/10/14 13:42:00', 'Description 97'),
(97,'REDEEMED', '2024/05/21 13:42:00','2023/10/22 13:42:00', 'Description 98'),
(98,'ALLOCATED', '2024/12/23 13:42:00','2024/01/23 13:42:00', 'Description 99'),
(99,'ALLOCATED', '2025/01/19 13:42:00','2023/06/16 13:42:00','Description 100');


INSERT INTO USER_ACCOUNT
VALUES
(0,'M','2001-01-27','Kendrick Koh',91234567),
(1,'F','2003-10-16','Oh ShuYi',92345678),
(2,'M','2004-01-01','Aaron Lim',93456789),
(3,'M','2001-04-23','Keith Ang',94567890),
(4,'M','2000-03-23','Derrick Brown',95678901),
(5,'F','1985-12-05','Jane Smith',96789012),
(6,'M','1995-08-10','Michael Johnson',97890123),
(7,'F','1982-04-15','Emily Brown',98901234),
(8,'M','1978-11-20','David Wilson',99012345),
(9,'F','1992-09-25','Sarah Martinez',90123456),
(10,'M','1995-12-03','John Doe',91234567),
(11,'F','1992-05-21','Emily Johnson',89765432),
(12,'M','1998-09-14','Michael Brown',87654321),
(13,'F','1990-02-08','Sophia Lee',92123456),
(14,'M','1993-11-19','William Wilson',83456789),
(15,'F','1997-07-25','Olivia Martinez',89012345),
(16,'M','1989-04-02','James Taylor',94321098),
(17,'F','1991-10-17','Ava Anderson',98765432),
(18,'M','1996-06-30','Alexander Thomas',91234567),
(19,'F','1994-03-10','Isabella Jackson',87654321),
(20,'M','1988-08-26','Ethan White',92123456),
(21,'F','1999-01-04','Sophia Harris',83456789),
(22,'M','1992-09-15','Michael Martin',89012345),
(23,'F','1995-05-29','Mia Thompson',94321098),
(24,'M','1990-11-11','Noah Garcia',98765432),
(25,'F','1987-07-22','Emma Rodriguez',91234567),
(26,'M','1993-04-03','Liam Martinez',87654321),
(27,'F','1998-10-16','Charlotte Wilson',92123456),
(28,'M','1991-06-01','Daniel Lee',83456789),
(29,'F','1989-03-11','Amelia Johnson',89012345),
(30,'M','1997-08-18','Benjamin Brown',94321098),
(31,'F','1994-01-28','Chloe Taylor',98765432),
(32,'M','1990-09-07','Jacob Anderson',91234567),
(33,'F','1988-05-20','Lily Thomas',87654321),
(34,'M','1995-11-25','Elijah Jackson',92123456),
(35,'F','1992-07-04','Avery White',83456789),
(36,'M','1999-04-16','William Harris',89012345),
(37,'F','1993-11-01','Grace Martin',94321098),
(38,'M','1987-07-10','Michael Garcia',98765432),
(39,'F','1996-03-21','Sofia Rodriguez',91234567),
(40,'M','1991-08-29','James Martinez',87654321),
(41,'F','1990-02-08','Emily Wilson',92123456),
(42,'M','1988-09-17','Alexander Lee',83456789),
(43,'F','1995-05-30','Isabella Thompson',89012345),
(44,'M','1992-11-11','Michael Garcia',94321098),
(45,'F','1989-07-22','Emma Harris',98765432),
(46,'M','1994-04-03','Liam Anderson',91234567),
(47,'F','1999-10-16','Charlotte Thomas',87654321),
(48,'M','1992-06-01','Daniel Jackson',92123456),
(49,'F','1988-03-11','Amelia White',83456789),
(50,'M','1997-08-18','Benjamin Harris',89012345);
```

SQL statements to solve queries

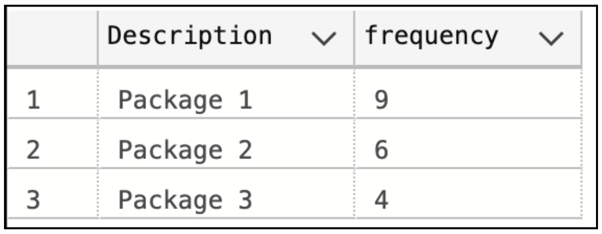
1. Find the most popular day packages where all participants are related to one another as either family members or members of the same club.

Assumptions:

1. Finding the most popular day package based on the assignment of day package to users.
2. More assignment = more popular because of the backend ML algorithm, means if the algorithm decides to assign a package → the package is more popular.
3. Abstraction layer (ML).

```
SELECT DP.Description, COUNT(*) AS frequency
FROM DAY_PACKAGE AS DP
JOIN (
  (SELECT DISTINCT Person1_UID AS Person_UID
  FROM RELATED
  WHERE [Type] = 'FAMILY' OR [Type] = 'CLUB')
  UNION
  (SELECT DISTINCT Person2_UID AS Person_UID
  FROM RELATED
  WHERE [Type] = 'FAMILY' OR [Type] = 'CLUB')
) AS FC ON DP.UID = FC.Person_UID
GROUP BY DP.Description
ORDER BY frequency DESC
```

Output:



	Description	frequency
1	Package 1	9
2	Package 2	6
3	Package 3	4

Figure 1: Final output for Appendix B Q1

Explanation:
Although the Day Package IDs (DID) is unique, there can be multiple users assigned to the same type of day package (package 1, package 2 etc.), we analyse the frequency of each package by the description of the package.

From the RELATION table, we count the number of unique users (UIDs) that are related either by “FAMILY” or “CLUB” by using the UNION of both Person1 & Person2 UIDs (UNION is used to remove duplicates). These unique UIDs are used to calculate the frequency of the usage of each day package.

1. SELECT DISTINCT Person1_UID with either “FAMILY” or “CLUB” relation
2. SELECT DISTINCT Person2_UID with either “FAMILY” or “CLUB” relation
3. UNION 1 & 2
4. JOIN on the 2 UIDs to output the unique UIDs

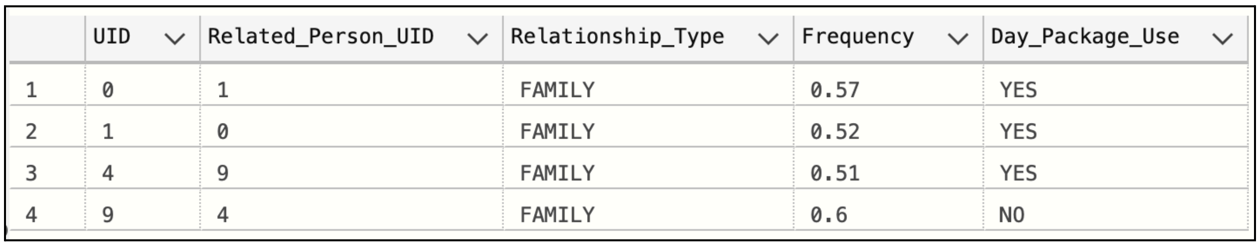
Lastly, we group the output by the description of the day package, and order by decreasing order of the frequency to view the most popular day packages.

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.

Assumptions: If a day package is assigned to a user, it means that the user will use the day package.

```
WITH RelatedDining AS (
   SELECT d1.*, d2.UID AS Related_Person_UID,r.Type AS Relationship_Type
   FROM DINE AS d1
   JOIN DINE AS d2 ON d1.OID = d2.OID AND LEFT(d1.Date_time_in, 13) =
LEFT(d2.Date_time_in, 13)
   JOIN RELATED AS r ON (d1.UID = r.Person1_UID AND d2.UID = r.Person2_UID) OR (d1.UID
= r.Person2_UID AND d2.UID = r.Person1_UID)
   WHERE d1.UID != d2.UID
   AND r.Type = 'FAMILY'
),

RelatedShopping AS (
   SELECT s1.*, s2.UID AS Related_Person_UID,r.Type AS Relationship_Type
   FROM SHOP_VERB AS s1
   JOIN SHOP_VERB AS s2 ON s1.SID = s2.SID AND LEFT(s1.Date_time_in, 13) =
LEFT(s2.Date_time_in, 13)
   JOIN RELATED AS r ON (s1.UID = r.Person1_UID AND s2.UID = r.Person2_UID) OR (s1.UID
= r.Person2_UID AND s2.UID = r.Person1_UID)
   WHERE s1.UID != s2.UID
   AND r.Type = 'FAMILY'
),

TotalVisits AS (
   SELECT UID,COUNT(*) AS TotalVisitsCount
   FROM (
       SELECT UID, Date_time_in FROM DINE
       UNION ALL
       SELECT UID, Date_time_in FROM SHOP_VERB
   ) AS SHOP_DINE
   GROUP BY UID
),

UIDFrequentVisits AS (
SELECT rd_counts.UID, RelatedDiningCount + RelatedShoppingCount AS
RelatedCount,TotalVisitsCount,
rd_counts.Related_Person_UID, rd_counts.Relationship_Type,
ROUND((CAST(RelatedDiningCount + RelatedShoppingCount AS FLOAT)/TotalVisitsCount),2)
AS Frequency
FROM (
   SELECT rd.UID, rd.Related_Person_UID, rd.Relationship_Type, COUNT(rd.UID) AS
RelatedDiningCount
   FROM RelatedDining AS rd
   GROUP BY rd.UID,rd.Related_Person_UID, rd.Relationship_Type
) AS rd_counts
JOIN (
   SELECT rs.UID, rs.Related_Person_UID, rs.Relationship_Type, COUNT(rs.UID) AS
RelatedShoppingCount
   FROM RelatedShopping AS rs
   GROUP BY rs.UID, rs.Related_Person_UID, rs.Relationship_Type
) AS rs_counts ON rd_counts.UID = rs_counts.UID
JOIN TotalVisits AS tv ON rd_counts.UID = tv.UID
)

SELECT ufv.UID, ufv.Related_Person_UID, ufv.Relationship_Type, ufv.Frequency,
   CASE
       WHEN COUNT(CASE WHEN ufv.UID = dp.UID AND dp.DID IS NOT NULL THEN 1 END) > 0
THEN 'YES'
       ELSE 'NO'
   END AS Day_Package_Use
   FROM UIDFrequentVisits AS ufv,
   DAY_PACKAGE AS dp
   GROUP BY ufv.UID, ufv.Related_Person_UID, ufv.Relationship_Type, ufv.Frequency
   HAVING ufv.Frequency >= 0.5
   ORDER BY ufv.UID ASC
```

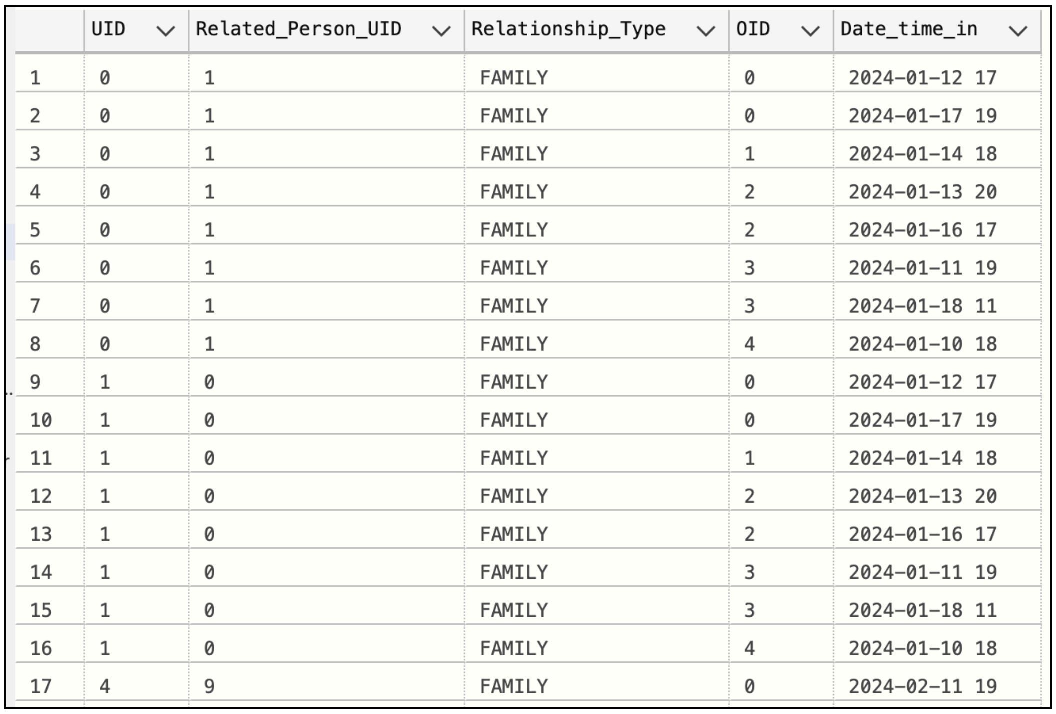
Output:



	UID	Related_Person_UID	Relationship_Type	Frequency	Day_Package_Use
1	0	1	FAMILY	0.57	YES
2	1	0	FAMILY	0.52	YES
3	4	9	FAMILY	0.51	YES
4	9	4	FAMILY	0.6	NO

Figure 2: Final output for Appendix B Q2

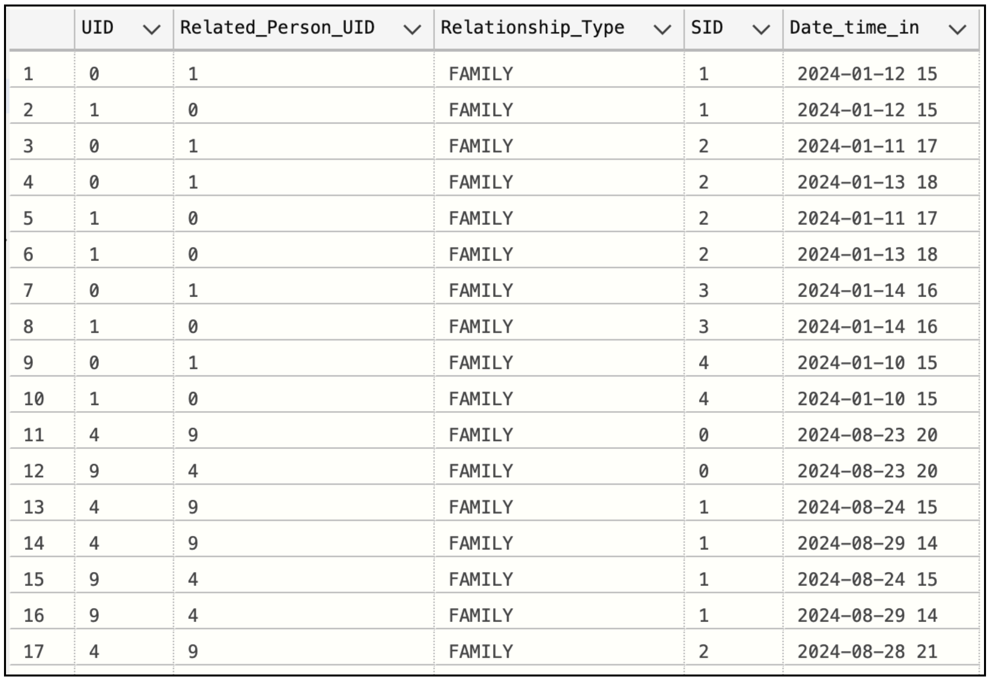
Explanation:
To find the dining events carried out by family members, a temporary table named "RelatedDining" is created which joins the "DINE" table with itself and the "RELATED" table, and selects all dining events carried out with users (Related_Person_UID) who are family members (Relationship_Type). This is done by checking if two distinct users visit the same restaurant outlet (OID) on the same day within the same hour. The first 13 characters from the left in the "Date_time_in" variable are extracted to obtain the date and hour of visit. It also checks if their relationship type is "FAMILY" from the "RELATED" table. We obtain the following output in **Figure 3**.



	UID	Related_Person_UID	Relationship_Type	OID	Date_time_in
1	0	1	FAMILY	0	2024-01-12 17
2	0	1	FAMILY	0	2024-01-17 19
3	0	1	FAMILY	1	2024-01-14 18
4	0	1	FAMILY	2	2024-01-13 20
5	0	1	FAMILY	2	2024-01-16 17
6	0	1	FAMILY	3	2024-01-11 19
7	0	1	FAMILY	3	2024-01-18 11
8	0	1	FAMILY	4	2024-01-10 18
9	1	0	FAMILY	0	2024-01-12 17
10	1	0	FAMILY	0	2024-01-17 19
11	1	0	FAMILY	1	2024-01-14 18
12	1	0	FAMILY	2	2024-01-13 20
13	1	0	FAMILY	2	2024-01-16 17
14	1	0	FAMILY	3	2024-01-11 19
15	1	0	FAMILY	3	2024-01-18 11
16	1	0	FAMILY	4	2024-01-10 18
17	4	9	FAMILY	0	2024-02-11 19

Figure 3: RelatedDining CTE

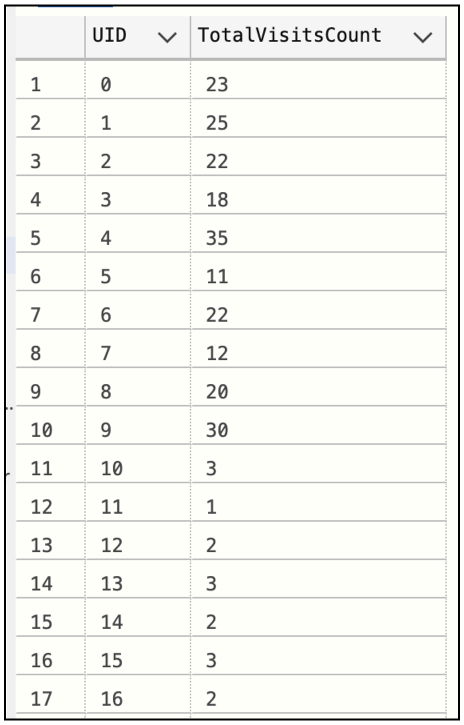
To find the dining events carried out by family members, a temporary table named "RelatedShopping" is created which joins the "SHOP_VERB" table with itself and the "RELATED" table, and selects all dining events carried out with users (Related_Person_UID) who are family members (Relationship_Type). This is done by checking if two distinct users visit the same shop (SID) on the same day within the same hour. The first 13 characters from the left in the "Date_time_in" variable are extracted to obtain the date and hour of visit. It also checks if their relationship type is "FAMILY" from the "RELATED" table. We obtain the following output in **Figure 4**.



	UID	Related_Person_UID	Relationship_Type	SID	Date_time_in
1	0	1	FAMILY	1	2024-01-12 15
2	1	0	FAMILY	1	2024-01-12 15
3	0	1	FAMILY	2	2024-01-11 17
4	0	1	FAMILY	2	2024-01-13 18
5	1	0	FAMILY	2	2024-01-11 17
6	1	0	FAMILY	2	2024-01-13 18
7	0	1	FAMILY	3	2024-01-14 16
8	1	0	FAMILY	3	2024-01-14 16
9	0	1	FAMILY	4	2024-01-10 15
10	1	0	FAMILY	4	2024-01-10 15
11	4	9	FAMILY	0	2024-08-23 20
12	9	4	FAMILY	0	2024-08-23 20
13	4	9	FAMILY	1	2024-08-24 15
14	4	9	FAMILY	1	2024-08-29 14
15	9	4	FAMILY	1	2024-08-24 15
16	9	4	FAMILY	1	2024-08-29 14
17	4	9	FAMILY	2	2024-08-28 21

Figure 4: RelatedShopping CTE

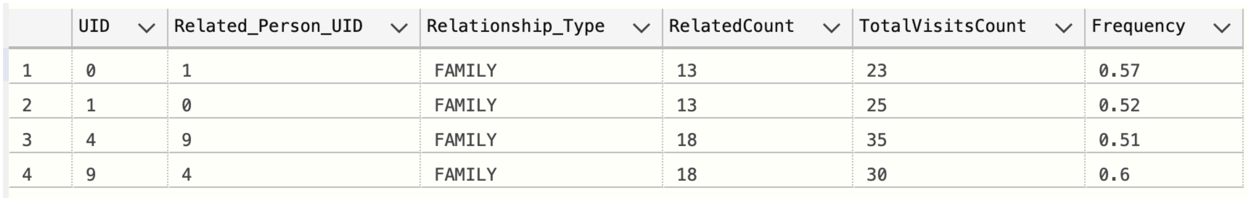
To calculate the total number of visits each user makes to restaurant outlets and shops, another temporary table named “TotalVisits” is created which joins the “DINE” and “SHOP_VERB” tables, and selects each user (UID) and their total count to shops and restaurant outlets (TotalVisitsCount). This is done using a subquery to combine the “UID” and “Date_time_in” from all dining and shopping events from “DINE” and “SHOP_VERB” tables into the “SHOP_DINE” table. The events are then grouped by users (UID) and counted. We obtain the following output in **Figure 5**.



	UID	TotalVisitsCount
1	0	23
2	1	25
3	2	22
4	3	18
5	4	35
6	5	11
7	6	22
8	7	12
9	8	20
10	9	30
11	10	3
12	11	1
13	12	2
14	13	3
15	14	2
16	15	3
17	16	2

Figure 5: TotalVisits CTE

Another temporary table named "UIDFrequentVisits" is created which joins the "RelatedDining", "RelatedShopping", and "TotalVisits" tables, and selects distinct pairs of users (UID and Related_Person_UID) who are family members (Relationship_Type), along with their corresponding number of visits alone (TotalVisitsCount) and with the family member (RelatedCount). Using "TotalVisitsCount" and "RelatedCount", the percentage of visits that each user makes with the related user is also calculated to give "Frequency". It uses a subquery to group the related dining and shopping events carried out by distinct pairs of users (UID and Related_Person_UID) and counts the number of events carried out by each pair (RelatedDiningCount and RelatedShoppingCount). We obtain the following output in **Figure 6**.



	UID	Related_Person_UID	Relationship_Type	RelatedCount	TotalVisitsCount	Frequency
1	0	1	FAMILY	13	23	0.57
2	1	0	FAMILY	13	25	0.52
3	4	9	FAMILY	18	35	0.51
4	9	4	FAMILY	18	30	0.6

Figure 6: UIDFrequentVisits CTE

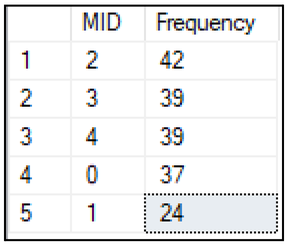
The main query then retrieves the final result by joining the "UIDFrequentVisits", "Day_package", and "Dine_Voucher" tables, and selects the distinct pairs of users (UID and Related_Person_UID) who are family members (Relationship_Type) and made visits with each other at least 50% of the time (Frequency), with indication of usage of day package (Day_Package_Use). It checks if a user has used a day package (Day_Package_Use) by counting the number of times that a day package (DID) is assigned to the user. If the count is more than 0, ‘YES’ will be output to indicate day package usage, otherwise ‘NO’ will be returned. The query also ensures that only pairs with a “Frequency” of at least 50% will be returned. The output is then sorted by ascending order of “UID”. The final output will be in **Figure 2**.

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

Assumptions: In any recommendation, the recommended restaurant (OID) is located in the recommended mall (MID).

```
SELECT *
FROM RESTAURANT_OUTLET RO
JOIN RECOMMENDATION REC ON RO.OID = REC.OID;

SELECT R.OID, R.RID, R.MID, R.NID, U.UID
FROM (
   SELECT RO.OID, RO.RID, RO.MID, REC.NID, REC.DID
   FROM RESTAURANT_OUTLET RO
   JOIN RECOMMENDATION REC ON RO.OID = REC.OID
) AS R
JOIN USER_USE_RECO U ON R.NID = U.NID;

SELECT R.MID, COUNT(*) AS Frequency
FROM (
   SELECT RO.OID, RO.RID, RO.MID, REC.NID
   FROM RESTAURANT_OUTLET RO
   JOIN RECOMMENDATION REC ON RO.OID = REC.OID
) AS R
JOIN USER_USE_RECO U ON R.NID = U.NID
GROUP BY R.MID
ORDER BY Frequency DESC;
```

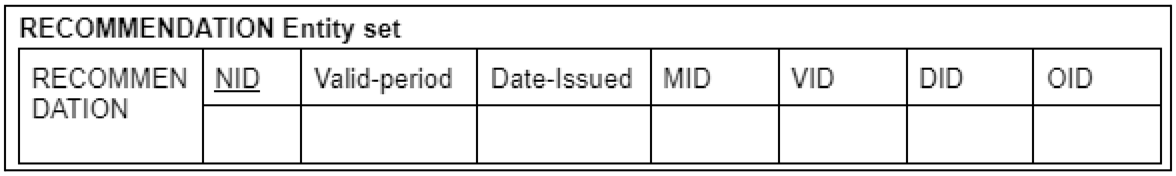
Output:



	MID	Frequency
1	2	42
2	3	39
3	4	39
4	0	37
5	1	24

Figure 7: Final output for Appendix B Q3

Explanation:

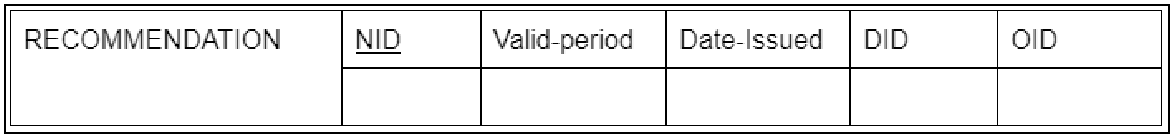


RECOMMENDATION Entity set

RECOMMENDATION	<u>NID</u>	Valid-period	Date-Issued	MID	VID	DID	OID

Figure 8: Original Schema for Recommendation

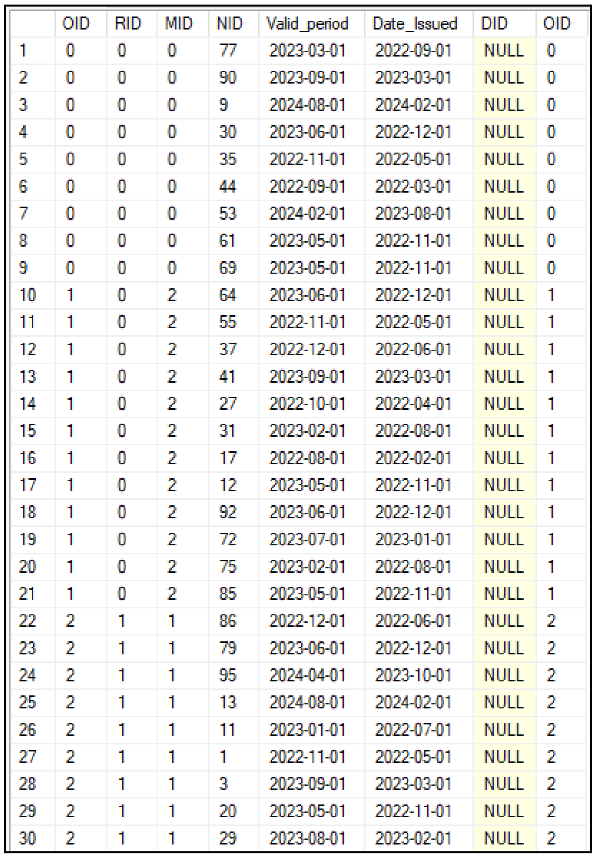
The original schema for RECOMMENDATION as shown in the **Figure 8** was decomposed to fulfil the criteria of 3NF. The resultant table after decomposition is RECOMMENDATION seen in **Figure 9**.



RECOMMENDATION	<u>NID</u>	Valid-period	Date-Issued	DID	OID

Figure 9: Decomposed Schema for Recommendation

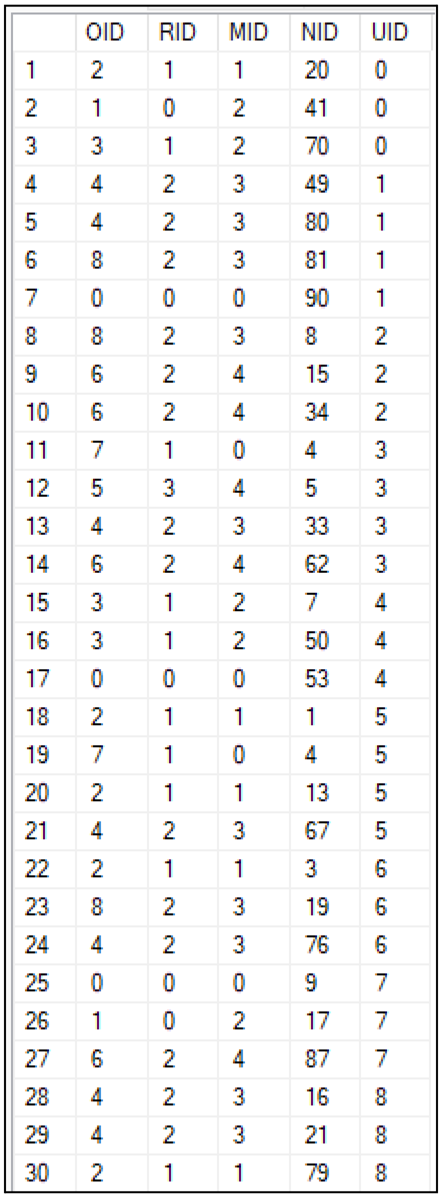
As MID (Mall ID) is no longer in the schema, we joined the tables RESTAURANT_OUTLET and RECOMMENDATION based on the common attribute OID (Outlet ID). This results in the following output in **Figure 10**.



	OID	RID	MID	NID	Valid_period	Date_Issued	DID	OID
1	0	0	0	77	2023-03-01	2022-09-01	NULL	0
2	0	0	0	90	2023-09-01	2023-03-01	NULL	0
3	0	0	0	9	2024-08-01	2024-02-01	NULL	0
4	0	0	0	30	2023-06-01	2022-12-01	NULL	0
5	0	0	0	35	2022-11-01	2022-05-01	NULL	0
6	0	0	0	44	2022-09-01	2022-03-01	NULL	0
7	0	0	0	53	2024-02-01	2023-08-01	NULL	0
8	0	0	0	61	2023-05-01	2022-11-01	NULL	0
9	0	0	0	69	2023-05-01	2022-11-01	NULL	0
10	1	0	2	64	2023-06-01	2022-12-01	NULL	1
11	1	0	2	55	2022-11-01	2022-05-01	NULL	1
12	1	0	2	37	2022-12-01	2022-06-01	NULL	1
13	1	0	2	41	2023-09-01	2023-03-01	NULL	1
14	1	0	2	27	2022-10-01	2022-04-01	NULL	1
15	1	0	2	31	2023-02-01	2022-08-01	NULL	1
16	1	0	2	17	2022-08-01	2022-02-01	NULL	1
17	1	0	2	12	2023-05-01	2022-11-01	NULL	1
18	1	0	2	92	2023-06-01	2022-12-01	NULL	1
19	1	0	2	72	2023-07-01	2023-01-01	NULL	1
20	1	0	2	75	2023-02-01	2022-08-01	NULL	1
21	1	0	2	85	2023-05-01	2022-11-01	NULL	1
22	2	1	1	86	2022-12-01	2022-06-01	NULL	2
23	2	1	1	79	2023-06-01	2022-12-01	NULL	2
24	2	1	1	95	2024-04-01	2023-10-01	NULL	2
25	2	1	1	13	2024-08-01	2024-02-01	NULL	2
26	2	1	1	11	2023-01-01	2022-07-01	NULL	2
27	2	1	1	1	2022-11-01	2022-05-01	NULL	2
28	2	1	1	3	2023-09-01	2023-03-01	NULL	2
29	2	1	1	20	2023-05-01	2022-11-01	NULL	2
30	2	1	1	29	2023-08-01	2023-02-01	NULL	2

Figure 10: RECOMMENDATION combined with RESTAURANT_OUTLET (first thirty rows)

After joining RESTAURANT_OUTLET with RECOMMENDATION, we then joined the resulting table with USER_USE_RECO based on the common attribute NID (Recommendation ID). We do this as a user can be given multiple recommendations, and a recommendation can be given to multiple users. This results in the following output in **Figure 11**.



	OID	RID	MID	NID	UID
1	2	1	1	20	0
2	1	0	2	41	0
3	3	1	2	70	0
4	4	2	3	49	1
5	4	2	3	80	1
6	8	2	3	81	1
7	0	0	0	90	1
8	8	2	3	8	2
9	6	2	4	15	2
10	6	2	4	34	2
11	7	1	0	4	3
12	5	3	4	5	3
13	4	2	3	33	3
14	6	2	4	62	3
15	3	1	2	7	4
16	3	1	2	50	4
17	0	0	0	53	4
18	2	1	1	1	5
19	7	1	0	4	5
20	2	1	1	13	5
21	4	2	3	67	5
22	2	1	1	3	6
23	8	2	3	19	6
24	4	2	3	76	6
25	0	0	0	9	7
26	1	0	2	17	7
27	6	2	4	87	7
28	4	2	3	16	8
29	4	2	3	21	8
30	2	1	1	79	8

Figure 11: Table in Figure 10 combined with USER_USE_RECO (first 30 rows)

Lastly, we computed the frequency of each MID (Mall ID) by grouping the results by MID and counting the occurrences, giving the resultant table as shown in **Figure 7**. Ordering the results by frequency in descending order provides a clear view of the popular recommendations from the app regarding malls.

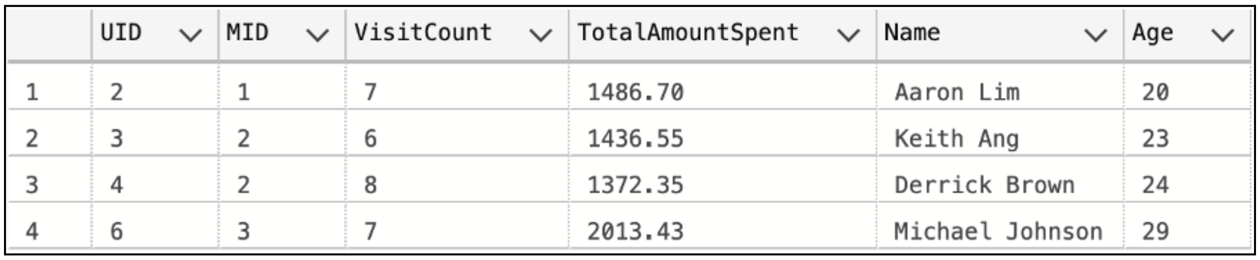
4. Compulsive shoppers are those who have visited a certain mall more than 5 times within a certain period of time. Find the youngest compulsive shoppers and the amount they spent in total during December 2023.

Assumptions.

1. For a user to be considered a compulsive shopper, they must visit any mall more than 5 times *within* December 2023.
2. Shop visits and Dine visits by a User on the same day in the same Mall count as the same Mall visits.
3. Finding the youngest may only return one result, hence we simply arrange by ascending age for the youngest to be the first entry and find the top youngest compulsive shoppers.

```
WITH VISITS AS ( --- 1
SELECT S.UID, COUNT(*) AS VISITCOUNT, S2.MID, SUM(S.Amount_spent) AS Amount_spent_Dec,
LEFT(S.Date_time_in,11) AS VisitDate
FROM SHOP_VERB AS S, SHOP_NOUN AS S2
WHERE S.SID = S2.SID AND S.Date_time_in >= '2023-12-01' AND S.Date_time_in <=
'2023-12-31'
GROUP BY S.UID, S2.MID, LEFT(S.Date_time_in,11)

UNION ALL
SELECT D.UID, COUNT(*) AS VISITCOUNT, D2.MID, SUM(D.Amount_spent) AS Amount_spent_Dec,
LEFT(D.Date_time_in,11) AS VisitDate
FROM DINE AS D, RESTAURANT_OUTLET AS D2
WHERE D.OID = D2.OID AND D.Date_time_in >= '2023-12-01' AND D.Date_time_in <=
'2023-12-31'
GROUP BY D.UID, D2.MID, LEFT(D.Date_time_in,11)
),

UNIQUEVISITS AS ( --- 2
   SELECT UID, MID, VisitDate
   FROM (
       SELECT UID, MID, VisitDate,
              ROW_NUMBER() OVER (PARTITION BY UID, MID, VisitDate ORDER BY VISITCOUNT
DESC) AS row_num
       FROM VISITS
   ) AS ranked_visits
   WHERE row_num = 1),

AMOUNTSPENT AS ( --- 3
   SELECT UID, SUM(Amount_spent_Dec) AS TotalAmountSpent
   FROM VISITS
   GROUP BY UID
)


SELECT uv.UID,  --- 4
      uv.MID,
      COUNT(*) AS VisitCount,
      ams.TotalAmountSpent,
      ua.Name, DATEDIFF(YEAR, ua.DOB, GETDATE()) AS Age
FROM UNIQUEVISITS AS uv
JOIN AMOUNTSPENT AS ams ON uv.UID = ams.UID
JOIN USER_ACCOUNT AS ua ON uv.UID = ua.UID
GROUP BY uv.UID, uv.MID, ams.TotalAmountSpent, ua.Name, ua.DOB
HAVING COUNT(*) > 5
ORDER BY Age ASC;
```

Output:

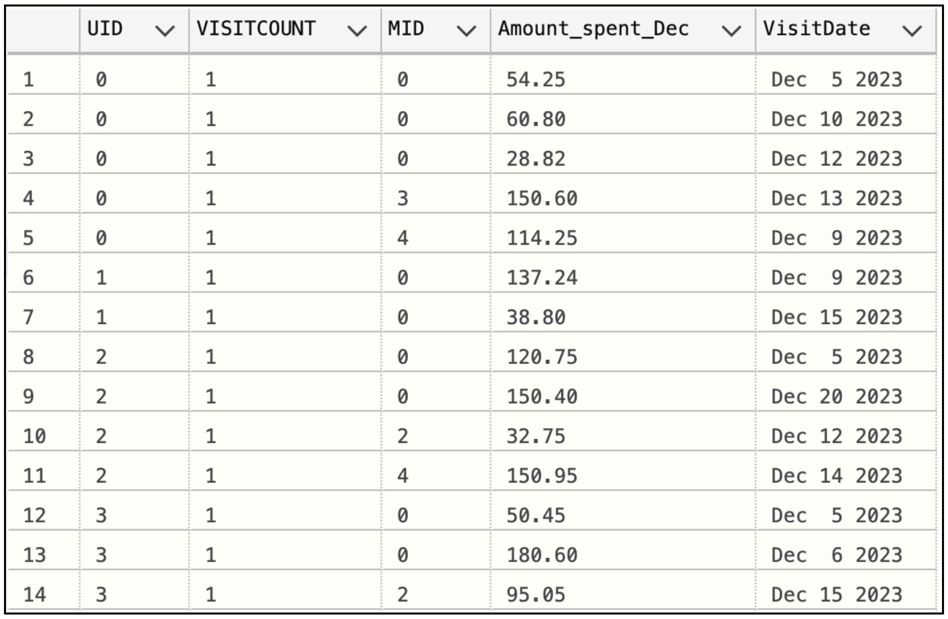


	UID	MID	VisitCount	TotalAmountSpent	Name	Age
1	2	1	7	1486.70	Aaron Lim	20
2	3	2	6	1436.55	Keith Ang	23
3	4	2	8	1372.35	Derrick Brown	24
4	6	3	7	2013.43	Michael Johnson	29

Figure 12: Final Output for Appendix B Q4

Explanation:

For --- 1, we first merge the two tables of SHOP_VERB and DINE using UNION ALL to keep all records, filtering only for the month of December 2023. Then using the Shop IDs (SID) for SHOP_VERB and Outlet IDs (OID) for DINE, we map and select the corresponding Mall IDs (MID) where these shops and restaurants were located. We also calculated the Amount Spent and grouped it by UID, MID, and VisitDate without time. We obtain the following output in **Figure 13**.

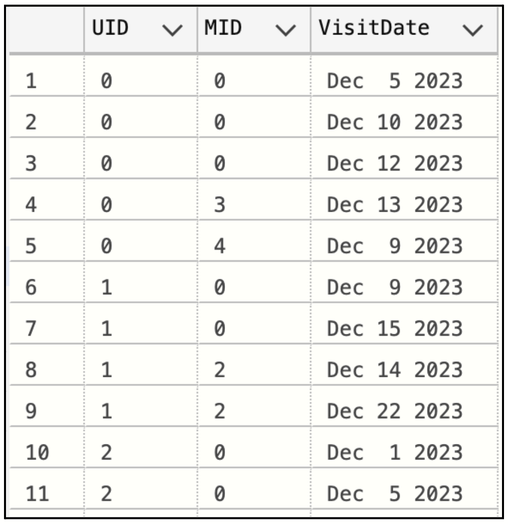


	UID	VISITCOUNT	MID	Amount_spent_Dec	VisitDate
1	0	1	0	54.25	Dec 5 2023
2	0	1	0	60.80	Dec 10 2023
3	0	1	0	28.82	Dec 12 2023
4	0	1	3	150.60	Dec 13 2023
5	0	1	4	114.25	Dec 9 2023
6	1	1	0	137.24	Dec 9 2023
7	1	1	0	38.80	Dec 15 2023
8	2	1	0	120.75	Dec 5 2023
9	2	1	0	150.40	Dec 20 2023
10	2	1	2	32.75	Dec 12 2023
11	2	1	4	150.95	Dec 14 2023
12	3	1	0	50.45	Dec 5 2023
13	3	1	0	180.60	Dec 6 2023
14	3	1	2	95.05	Dec 15 2023

Figure 13: VISITS CTE

However, the above CTE records a User going into a Shop in Mall 1 and then Dining in Mall 1 as two separate visits when it should be counted as one visit.

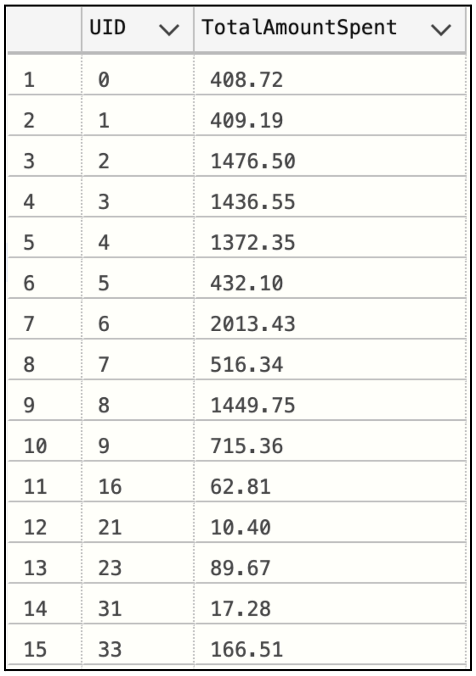
To address this in --- 2, we took the VISITS CTE and applied PARTITION to obtain unique combinations of UID, MID, and VisitDate, indicating a unique visit to a Mall in a day, before assigning each instance of a unique combination a ROW NUMBER(). All first instances of unique UID, MID, VisitDate will be given a row number of 1, increasing to 2,3, etc. for duplicates. Then we simply filtered out the instances where row numbers were not 1 to obtain unique mall visits. The output for this table is given in **Figure 14**.



	UID	MID	VisitDate
1	0	0	Dec 5 2023
2	0	0	Dec 10 2023
3	0	0	Dec 12 2023
4	0	3	Dec 13 2023
5	0	4	Dec 9 2023
6	1	0	Dec 9 2023
7	1	0	Dec 15 2023
8	1	2	Dec 14 2023
9	1	2	Dec 22 2023
10	2	0	Dec 1 2023
11	2	0	Dec 5 2023

Figure 14: UNIQUEVISTS CTE

For --- 3, we created a CTE AMOUNTSPENT to calculate the TotalAmountSpent in December using the VISITS CTE. Output is shown in **Figure 15**.



	UID	TotalAmountSpent
1	0	408.72
2	1	409.19
3	2	1476.50
4	3	1436.55
5	4	1372.35
6	5	432.10
7	6	2013.43
8	7	516.34
9	8	1449.75
10	9	715.36
11	16	62.81
12	21	10.40
13	23	89.67
14	31	17.28
15	33	166.51

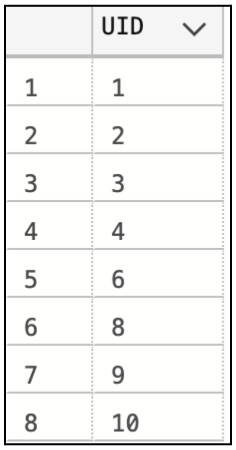
Figure 15: AMOUNTSPENT CTE

Finally, in --- 4, we applied the COUNT operator to the UNIQUEVISITS CTE to obtain the number of visits to a particular mall as VisitCount. We added TotalAmountSpent from AMOUNTSPENT CTE, joining based on matching UID. We also calculated age of these shoppers by joining with USER_ACCOUNT table on the condition of matching UIDs and using DATEDIFF. Lastly, we implemented a constraint where VisitCount had to be more than 5 to only show shoppers with compulsive tendencies. Output as shown in **Figure 12**.

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

```
-- Creation of tables to:
-- 1. Keep track of the number of shops a mall has (RestaurantCount)
-- 2. Keep track of users that visits any restaurants in a mall (RestaurantVisited)
WITH RestaurantCount AS (
 SELECT MID, COUNT(OID) AS Shops
 FROM RESTAURANT_OUTLET
 GROUP BY MID
),
RestaurantVisited AS(
 SELECT R.MID, D.UID, COUNT(DISTINCT D.OID) AS Visits
 FROM DINE D
 LEFT JOIN RESTAURANT_OUTLET R ON D.OID = R.OID
 GROUP BY R.MID, D.UID
)

--Finds users that visits no restaurant in a mall
(SELECT
 distinct_users.UID
FROM
 (SELECT DISTINCT MID FROM RESTAURANT_OUTLET) malls
CROSS JOIN
 (SELECT DISTINCT UID FROM DINE) distinct_users
LEFT JOIN RestaurantVisited ON
 RestaurantVisited.UID = distinct_users.UID
 AND RestaurantVisited.MID = malls.MID
WHERE
 RestaurantVisited.Visits IS NULL)

INTERSECT

--Finds users that visits all restaurants in a mall
(SELECT DISTINCT RV.UID
FROM RestaurantVisited RV
INNER JOIN RestaurantCount RC ON RV.MID = RC.MID
WHERE RV.Visits = RC.Shops);
```

Output:



	UID
1	1
2	2
3	3
4	4
5	6
6	8
7	9
8	10

Figure 16: Final output for Appendix B Q5

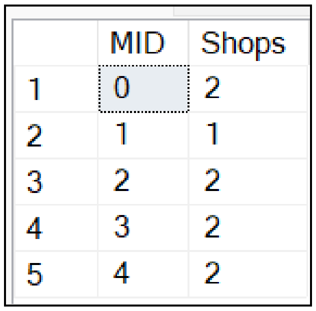
Explanation:

In solving this query, we first break down the query into 2 parts:

1. Find all users that visited all restaurants in some mall
2. Find all users that did not visit any restaurants in some other mall.

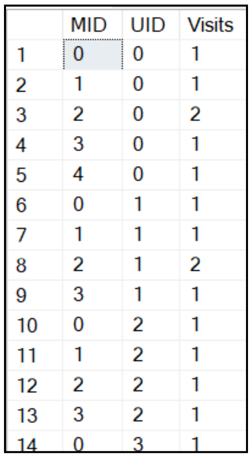
Then, we take the intersection of these 2 in order to get the query.

First, we created RestaurantCount in **Figure 17** to keep track of the number of restaurants in each mall and RestaurantVisited in **Figure 18** to keep track of the number of restaurants that each user visits in a mall.



	MID	Shops
1	0	2
2	1	1
3	2	2
4	3	2
5	4	2

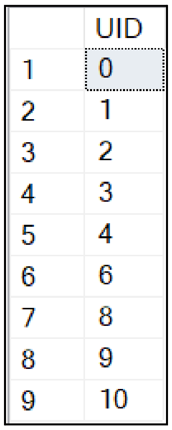
Figure 17: Table of RestaurantCount



	MID	UID	Visits
1	0	0	1
2	1	0	1
3	2	0	2
4	3	0	1
5	4	0	1
6	0	1	1
7	1	1	1
8	2	1	2
9	3	1	1
10	0	2	1
11	1	2	1
12	2	2	1
13	3	2	1
14	0	3	1

Figure 18: Table of RestaurantVisited

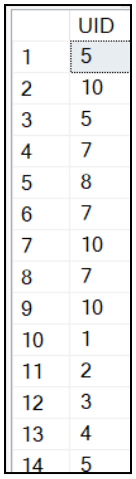
In order to find the users that visited all the restaurants in a mall, we do an inner join with RestaurantCount on RestaurantVisited based on the common attribute MID. We then compare the visits count with the shops counts and if they match means the user has indeed visited all restaurants in the mall. We will use this as a subquery, R1 in **Figure 19**, for intersection later.



	UID
1	0
2	1
3	2
4	3
5	4
6	6
7	8
8	9
9	10

Figure 19: Table that shows all the Users that have visited all the restaurants in a mall, R1

Next, we find the users that have not visited any restaurants in some other malls. To do so, we first do a cross joint with the MIDs from RESTAURANT_OUTLET as malls with the UIDs from DINE as distinct_users in order to get all the MIDs to associate with the UIDs to get the NULL values out for comparison later. Then, we do a LEFT JOIN into RestaurantVisited based on RestaurantVisited.UID = distinct_users.UID AND RestaurantVisited.MID = malls.MID in order to compare all the NULL values. By using the NULL values, we can find out those that did not visit all the restaurants in some other malls to give subquery R2 in **Figure 20**.



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

Figure 20: Table that shows all the Users who have not visited any of the restaurants in some other malls, R2

Lastly, we do an intersection of these 2 subqueries in order to find those that visited all restaurants in some malls (R1) but none of the restaurants in some other malls (R2).

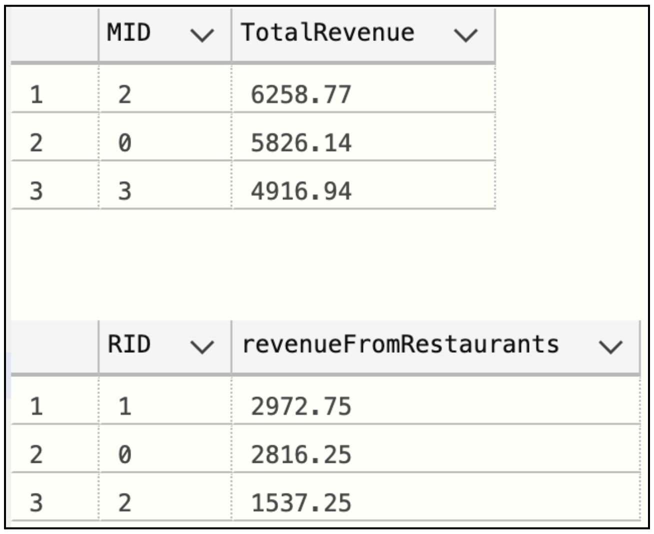
6. What are the top 3 highest earning malls and restaurants?

Assumptions:

1. Mall revenue consists of the sum of amount spent in every Shop and Restaurant_Outlet per Mall
2. Restaurant revenue consists of the sum of amount spent per Restaurant_Chain

```
--------------Querying Top 3 highest earning Malls---------------------
-- Producing table with Malls and their associated malls and summing the Amount_spent
WITH S AS (SELECT M.MID, SUM(SV.Amount_spent) AS revenueFromShops
FROM MALL M, SHOP_VERB SV, SHOP_NOUN SN
WHERE M.MID = SN.MID AND SN.SID = SV.SID
GROUP BY M.MID),

-- Producing table with Malls and their associated restaurants and summing the
Amount_spent
R AS (SELECT M.MID, SUM(D.Amount_spent) AS revenueFromRestaurants
FROM MALL M, DINE D, RESTAURANT_OUTLET RO
WHERE M.MID = RO.MID AND D.OID = RO.OID
GROUP BY M.MID)

-- Main code for Malls, adding revenue from shops and restaurants and displays top 3
Malls that produce the highest revenue, sorting them in DESC order
SELECT TOP 3 S.MID, (S.revenueFromShops+R.revenueFromRestaurants) AS TotalRevenue
FROM S, R
WHERE S.MID = R.MID
ORDER BY TotalRevenue DESC

--------------Querying Top 3 highest earning restaurants---------------------
-- Main code for Restaurant outlets, displays top 3 restaurant chains that produce the
highest revenue and sorting them in DESC order
SELECT TOP 3 RC.RID, SUM(D.Amount_spent) AS revenueFromRestaurants
FROM DINE D, RESTAURANT_OUTLET RO, RESTAURANT_CHAIN RC
WHERE RO.OID = D.OID AND RC.RID = RO.RID
GROUP BY RC.RID
ORDER BY revenueFromRestaurants DESC
```

Output:



	MID	TotalRevenue
1	2	6258.77
2	0	5826.14
3	3	4916.94

	RID	revenueFromRestaurants
1	1	2972.75
2	0	2816.25
3	2	1537.25

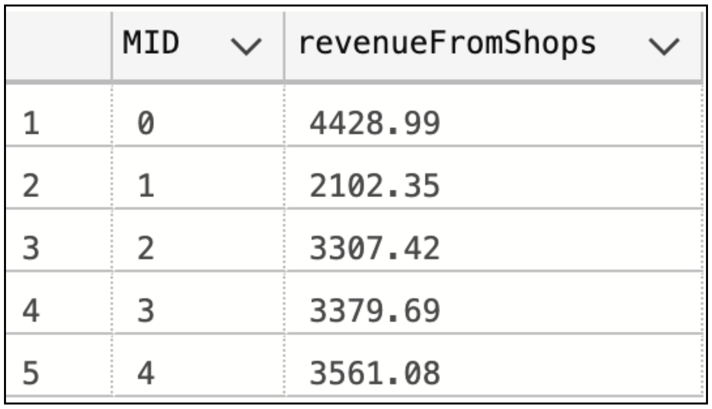
Figure 21: Final output for Appendix B Q6

Explanation:
To find the top 3 highest earning Malls and Restaurants, we focus on finding the MALLS and RESTAURANT_OUTLETs that produce the highest revenue by summing up all the "amount spent" at all shops in a MALL and all RESTAURANT_OUTLET in RESTAURANT_CHAIN.

The amount spent attribute is found in the SHOP_VERB and DINE schema respectively. The output shows the top 3 MALL and RESTAURANT_CHAIN based on MID and RID respectively.

Selecting Top 3 MALLs with the highest revenue from its shops (**Figure 22**):

- Finding SHOPS related to a given MALL and summing the Amount_spent (S):
- Join related entries in the MALL, DINE and RESTAURANT_OUTLET tables in the WHERE clause by matching their MID and SID
- GROUP them by MALL.MID
- SUM each MALL's individual shop revenue as revenueFromShops

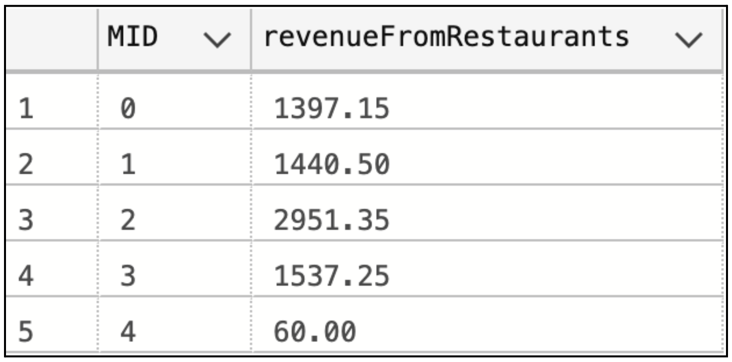


	MID	revenueFromShops
1	0	4428.99
2	1	2102.35
3	2	3307.42
4	3	3379.69
5	4	3561.08

Figure 22: Table for total shop revenue per mall

Finding RESTAURANT_OUTLETs related to a given MALL and summing the Amount_spent (R) (**Figure 23**):

- Join related entries in the MALL, SHOP_VERB and SHOP_NOUN tables in the WHERE clause by matching their MID and OID
- GROUP them by MALL.MID
- SUM each MALL's individual restaurant revenue as revenueFromRestaurants

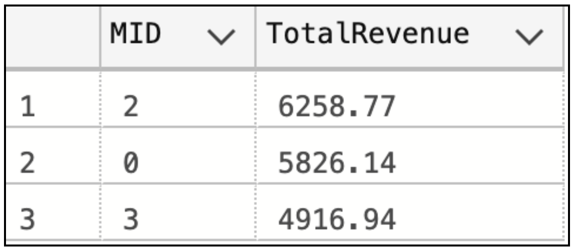


	MID	revenueFromRestaurants
1	0	1397.15
2	1	1440.50
3	2	2951.35
4	3	1537.25
5	4	60.00

Figure 23: Table for total restaurant revenue per mall

Summing revenueFromShops and revenueFromRestaurants and finding the top 3 highest earning MALLS (**Figure 24**):

- Join S and R in the WHERE clause by matching their MID
- Sum revenueFromShops and revenueFromRestaurants
- Order the output in DESC order

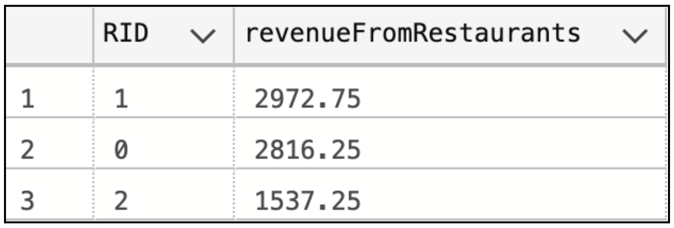


	MID	TotalRevenue
1	2	6258.77
2	0	5826.14
3	3	4916.94

Figure 24: Table for total revenue per mall

Selecting Top 3 RESTAURANT_OUTLETs with the highest revenue from its restaurants (**Figure 25**):

- Join related entries in the DINE, RESTAURANT_OUTLET and RESTAURANT_CHAIN tables in the WHERE clause by matching their OID and RID
- GROUP them by RESTAURANT_OUTLET.RID
- SUM each RESTAURANT_CHAIN's restaurant revenue as revenueFromRestaurants
- Order them in DESC order



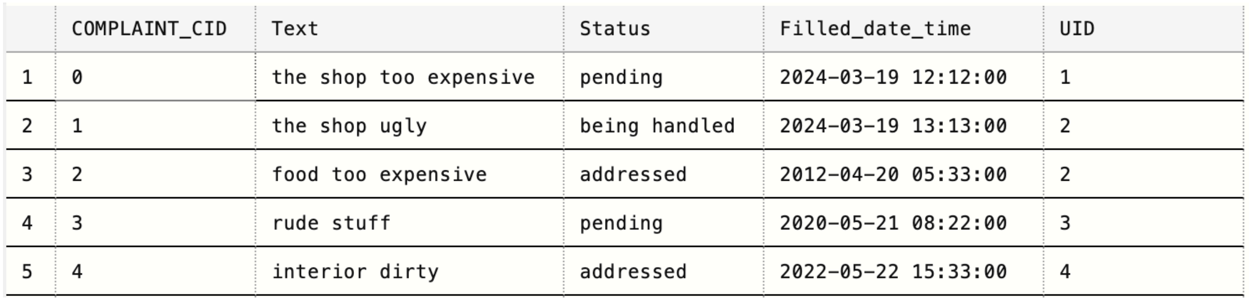
	RID	revenueFromRestaurants
1	1	2972.75
2	0	2816.25
3	2	1537.25

Figure 25: Table for total restaurant revenue per restaurant chain

Table Records

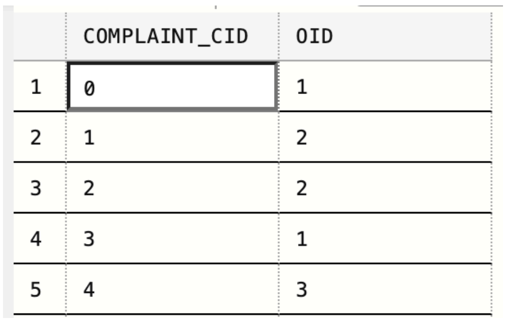
Complete table records on google sheets.

COMPLAINT



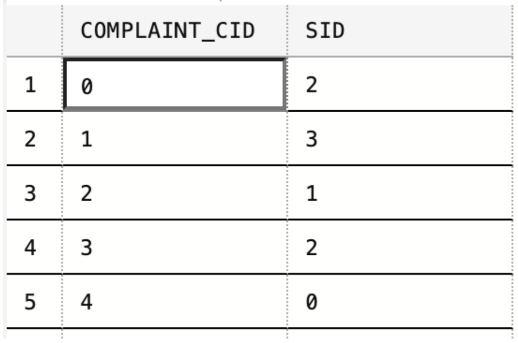
	COMPLAINT_CID	Text	Status	Filled_date_time	UID
1	0	the shop too expensive	pending	2024-03-19 12:12:00	1
2	1	the shop ugly	being handled	2024-03-19 13:13:00	2
3	2	food too expensive	addressed	2012-04-20 05:33:00	2
4	3	rude stuff	pending	2020-05-21 08:22:00	3
5	4	interior dirty	addressed	2022-05-22 15:33:00	4

COMPLAINTS_ON_RESTAURANT



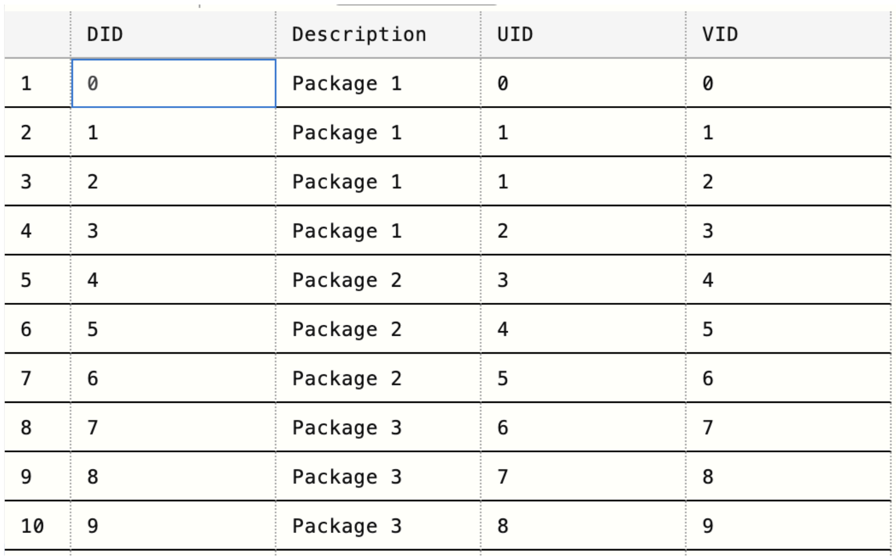
	COMPLAINT_CID	OID
1	0	1
2	1	2
3	2	2
4	3	1
5	4	3

COMPLAINTS_ON_SHOP



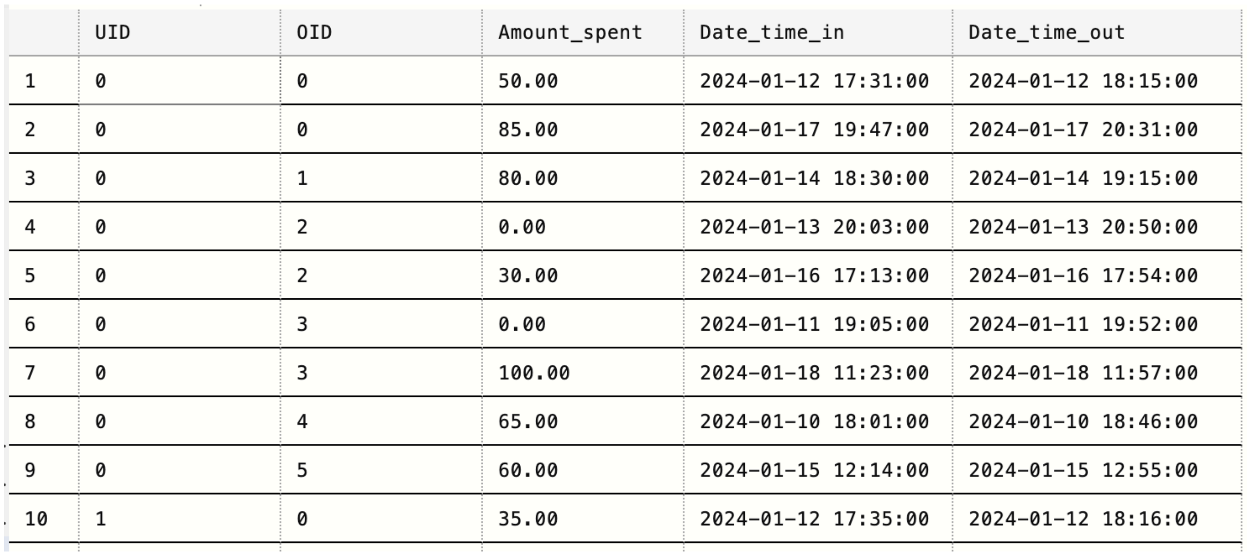
	COMPLAINT_CID	SID
1	0	2
2	1	3
3	2	1
4	3	2
5	4	0

DAY_PACKAGE (10 out of 21 rows)



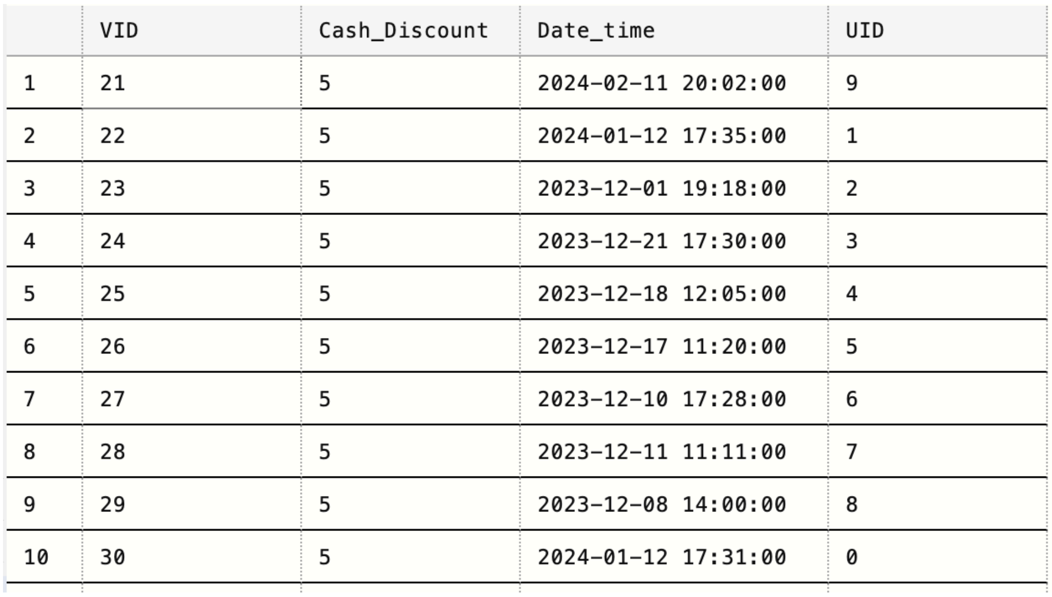
	DID	Description	UID	VID
1	0	Package 1	0	0
2	1	Package 1	1	1
3	2	Package 1	1	2
4	3	Package 1	2	3
5	4	Package 2	3	4
6	5	Package 2	4	5
7	6	Package 2	5	6
8	7	Package 3	6	7
9	8	Package 3	7	8
10	9	Package 3	8	9

DINE (10 out of 104 rows)



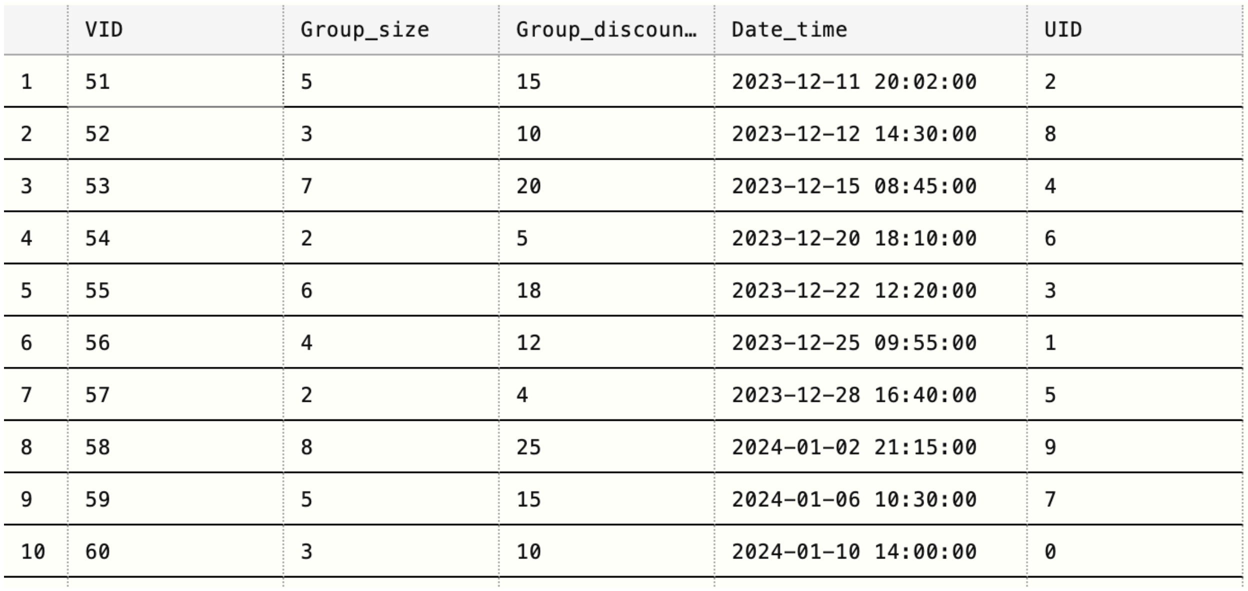
	UID	OID	Amount_spent	Date_time_in	Date_time_out
1	0	0	50.00	2024-01-12 17:31:00	2024-01-12 18:15:00
2	0	0	85.00	2024-01-17 19:47:00	2024-01-17 20:31:00
3	0	1	80.00	2024-01-14 18:30:00	2024-01-14 19:15:00
4	0	2	0.00	2024-01-13 20:03:00	2024-01-13 20:50:00
5	0	2	30.00	2024-01-16 17:13:00	2024-01-16 17:54:00
6	0	3	0.00	2024-01-11 19:05:00	2024-01-11 19:52:00
7	0	3	100.00	2024-01-18 11:23:00	2024-01-18 11:57:00
8	0	4	65.00	2024-01-10 18:01:00	2024-01-10 18:46:00
9	0	5	60.00	2024-01-15 12:14:00	2024-01-15 12:55:00
10	1	0	35.00	2024-01-12 17:35:00	2024-01-12 18:16:00

DINE_VOUCHER (10 out of 30 rows)



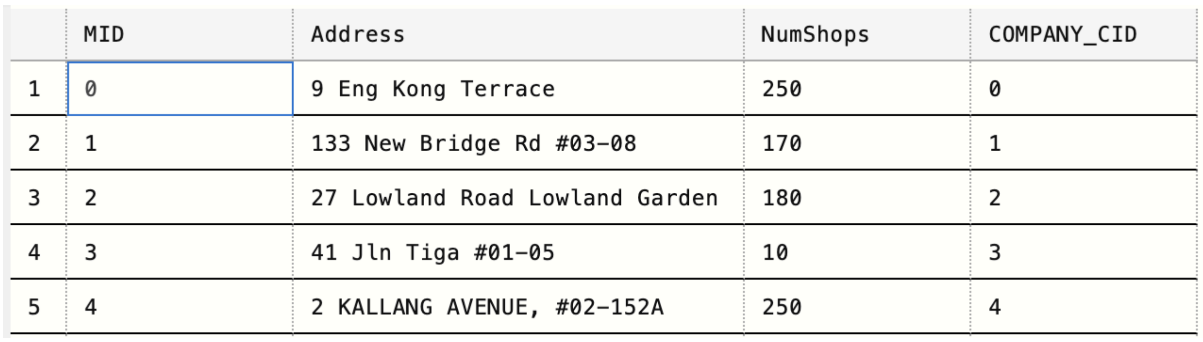
	VID	Cash_Discount	Date_time	UID
1	21	5	2024-02-11 20:02:00	9
2	22	5	2024-01-12 17:35:00	1
3	23	5	2023-12-01 19:18:00	2
4	24	5	2023-12-21 17:30:00	3
5	25	5	2023-12-18 12:05:00	4
6	26	5	2023-12-17 11:20:00	5
7	27	5	2023-12-10 17:28:00	6
8	28	5	2023-12-11 11:11:00	7
9	29	5	2023-12-08 14:00:00	8
10	30	5	2024-01-12 17:31:00	0

GROUP_VOUCHER (10 out of 20 rows)



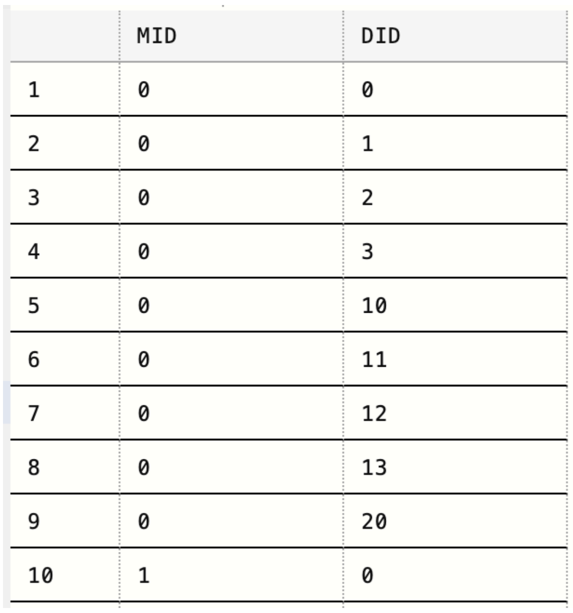
	VID	Group_size	Group_discoun…	Date_time	UID
1	51	5	15	2023-12-11 20:02:00	2
2	52	3	10	2023-12-12 14:30:00	8
3	53	7	20	2023-12-15 08:45:00	4
4	54	2	5	2023-12-20 18:10:00	6
5	55	6	18	2023-12-22 12:20:00	3
6	56	4	12	2023-12-25 09:55:00	1
7	57	2	4	2023-12-28 16:40:00	5
8	58	8	25	2024-01-02 21:15:00	9
9	59	5	15	2024-01-06 10:30:00	7
10	60	3	10	2024-01-10 14:00:00	0

MALL



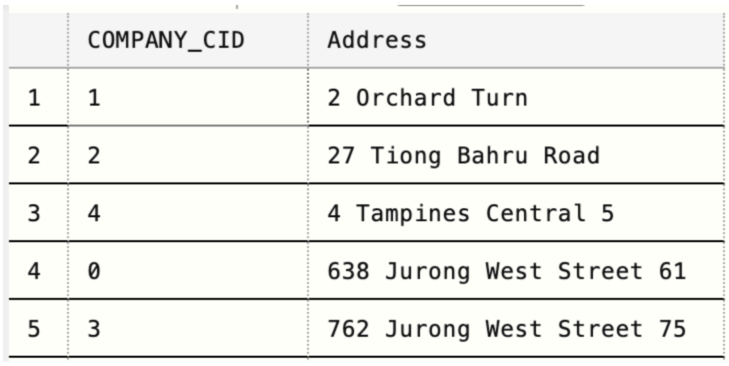
	MID	Address	NumShops	COMPANY_CID
1	0	9 Eng Kong Terrace	250	0
2	1	133 New Bridge Rd #03-08	170	1
3	2	27 Lowland Road Lowland Garden	180	2
4	3	41 Jln Tiga #01-05	10	3
5	4	2 KALLANG AVENUE, #02-152A	250	4

MALL_HAS_PACKAGE (10 out of 63 rows)



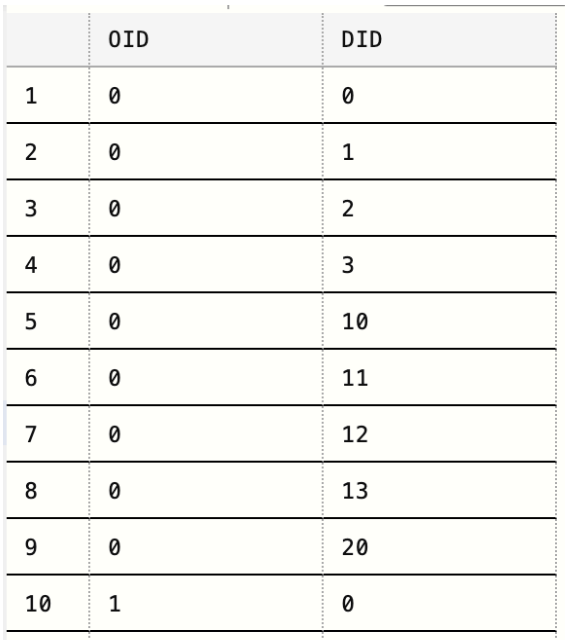
	MID	DID
1	0	0
2	0	1
3	0	2
4	0	3
5	0	10
6	0	11
7	0	12
8	0	13
9	0	20
10	1	0

MALL_MGMT_COMPANY



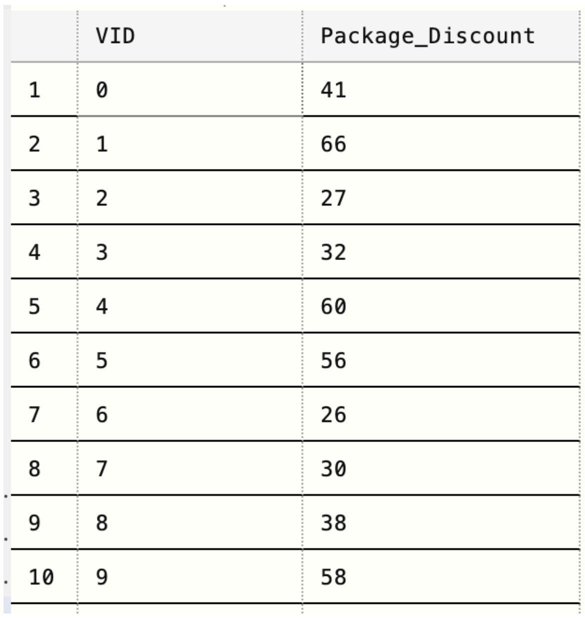
	COMPANY_CID	Address
1	1	2 Orchard Turn
2	2	27 Tiong Bahru Road
3	4	4 Tampines Central 5
4	0	638 Jurong West Street 61
5	3	762 Jurong West Street 75

PACKAGE_HAS_RESTAURANT (10 out of 63 rows)



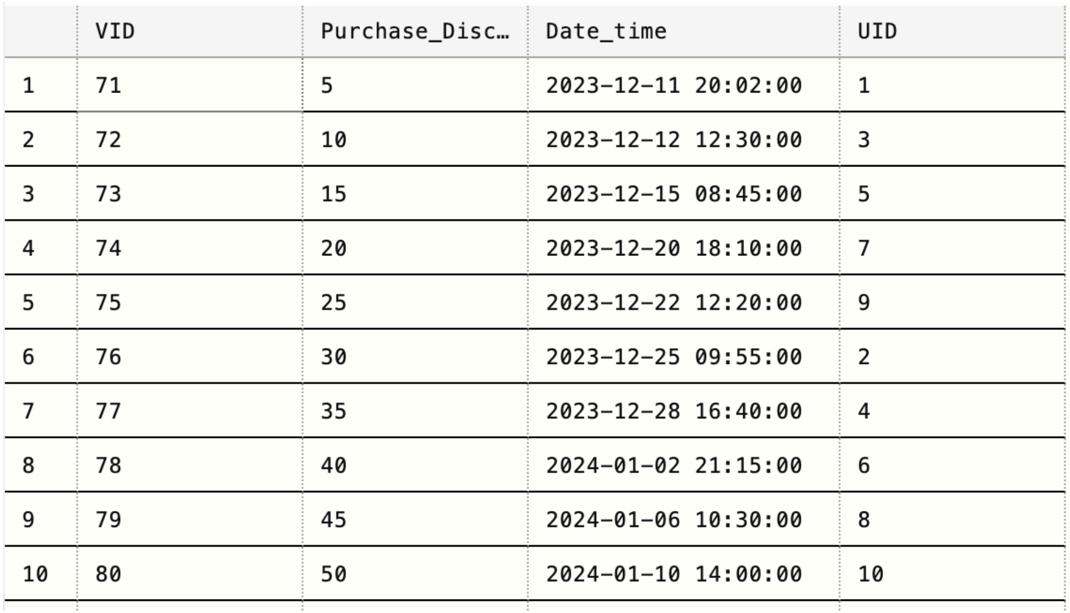
	OID	DID
1	0	0
2	0	1
3	0	2
4	0	3
5	0	10
6	0	11
7	0	12
8	0	13
9	0	20
10	1	0

PACKAGE_VOUCHER (10 out of 20 rows)



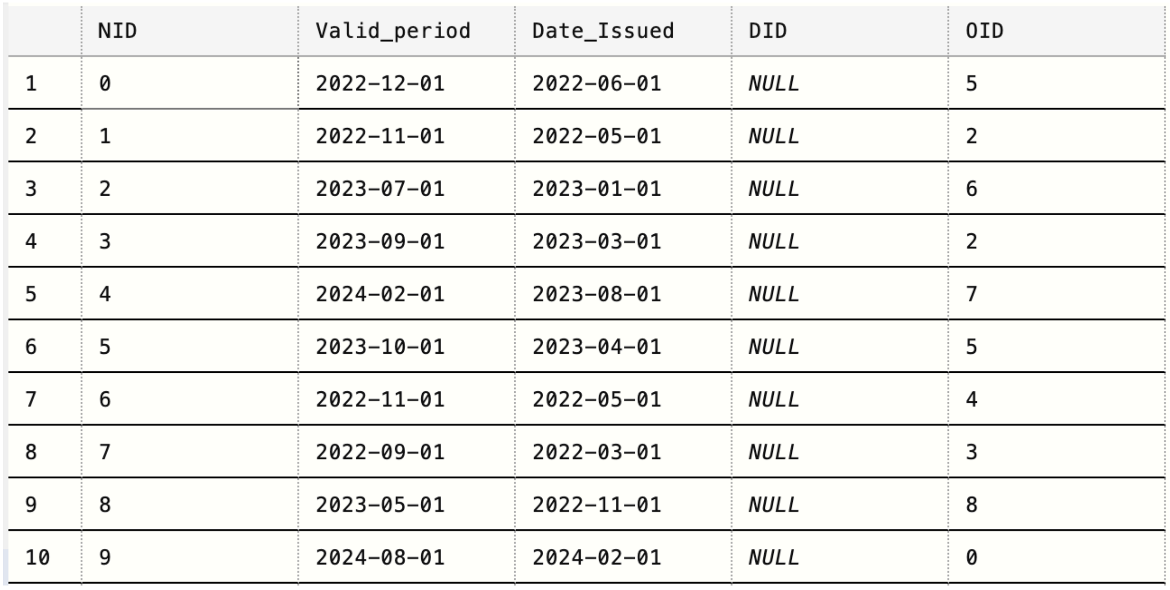
	VID	Package_Discount
1	0	41
2	1	66
3	2	27
4	3	32
5	4	60
6	5	56
7	6	26
8	7	30
9	8	38
10	9	58

PURCHASE_VOUCHER (10 out of 29 rows)



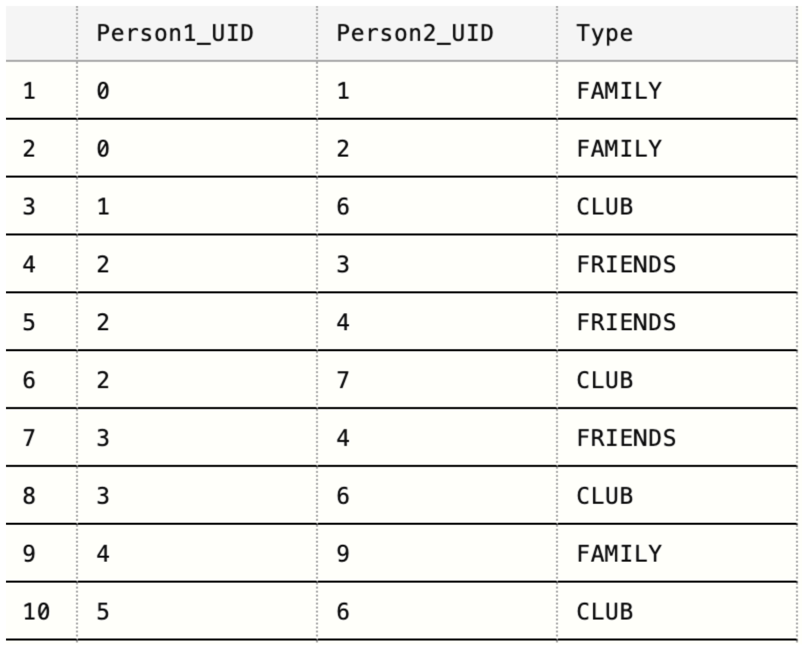
	VID	Purchase_Disc…	Date_time	UID
1	71	5	2023-12-11 20:02:00	1
2	72	10	2023-12-12 12:30:00	3
3	73	15	2023-12-15 08:45:00	5
4	74	20	2023-12-20 18:10:00	7
5	75	25	2023-12-22 12:20:00	9
6	76	30	2023-12-25 09:55:00	2
7	77	35	2023-12-28 16:40:00	4
8	78	40	2024-01-02 21:15:00	6
9	79	45	2024-01-06 10:30:00	8
10	80	50	2024-01-10 14:00:00	10

RECOMMENDATION (10 out of 118 rows)



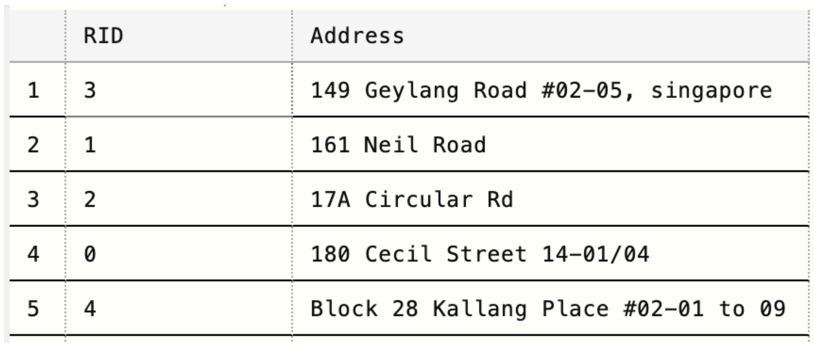
	NID	Valid_period	Date_Issued	DID	OID
1	0	2022-12-01	2022-06-01	*NULL*	5
2	1	2022-11-01	2022-05-01	*NULL*	2
3	2	2023-07-01	2023-01-01	*NULL*	6
4	3	2023-09-01	2023-03-01	*NULL*	2
5	4	2024-02-01	2023-08-01	*NULL*	7
6	5	2023-10-01	2023-04-01	*NULL*	5
7	6	2022-11-01	2022-05-01	*NULL*	4
8	7	2022-09-01	2022-03-01	*NULL*	3
9	8	2023-05-01	2022-11-01	*NULL*	8
10	9	2024-08-01	2024-02-01	*NULL*	0

RELATED



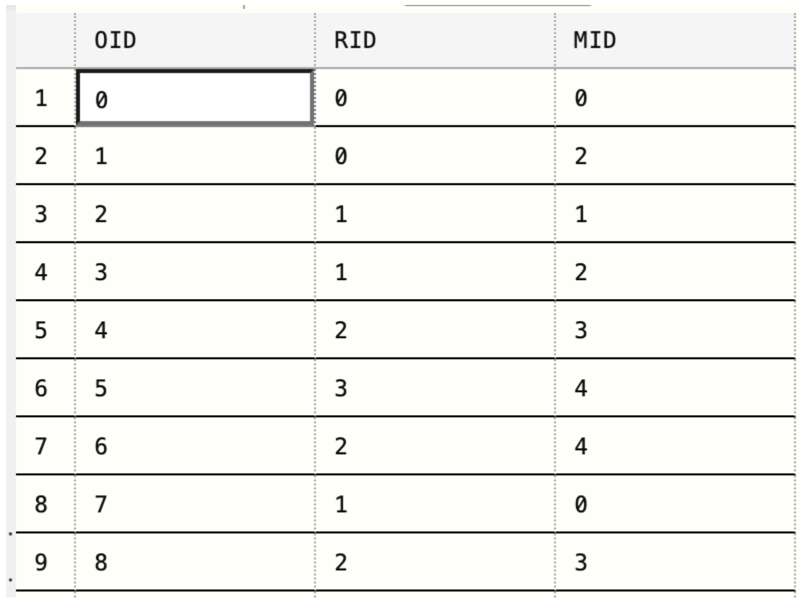
	Person1_UID	Person2_UID	Type
1	0	1	FAMILY
2	0	2	FAMILY
3	1	6	CLUB
4	2	3	FRIENDS
5	2	4	FRIENDS
6	2	7	CLUB
7	3	4	FRIENDS
8	3	6	CLUB
9	4	9	FAMILY
10	5	6	CLUB

RESTAURANT_CHAIN



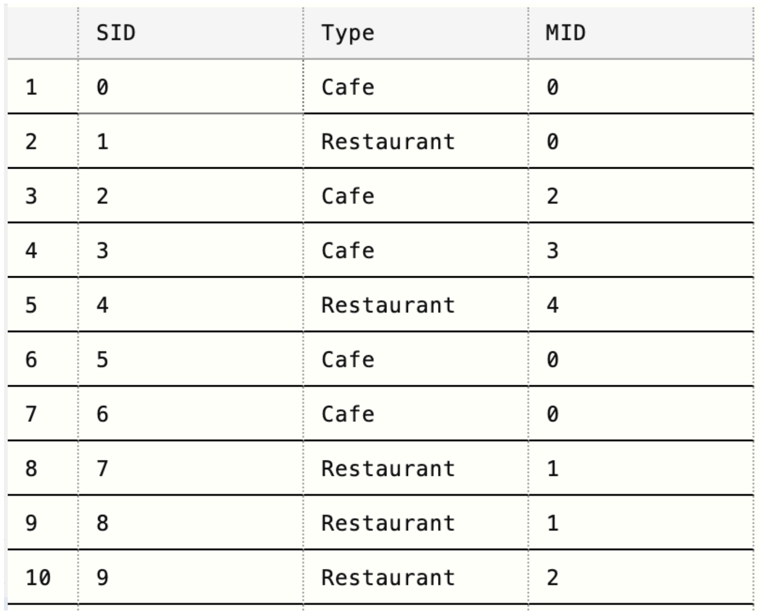
	RID	Address
1	3	149 Geylang Road #02-05, singapore
2	1	161 Neil Road
3	2	17A Circular Rd
4	0	180 Cecil Street 14-01/04
5	4	Block 28 Kallang Place #02-01 to 09

RESTAURANT_OUTLET



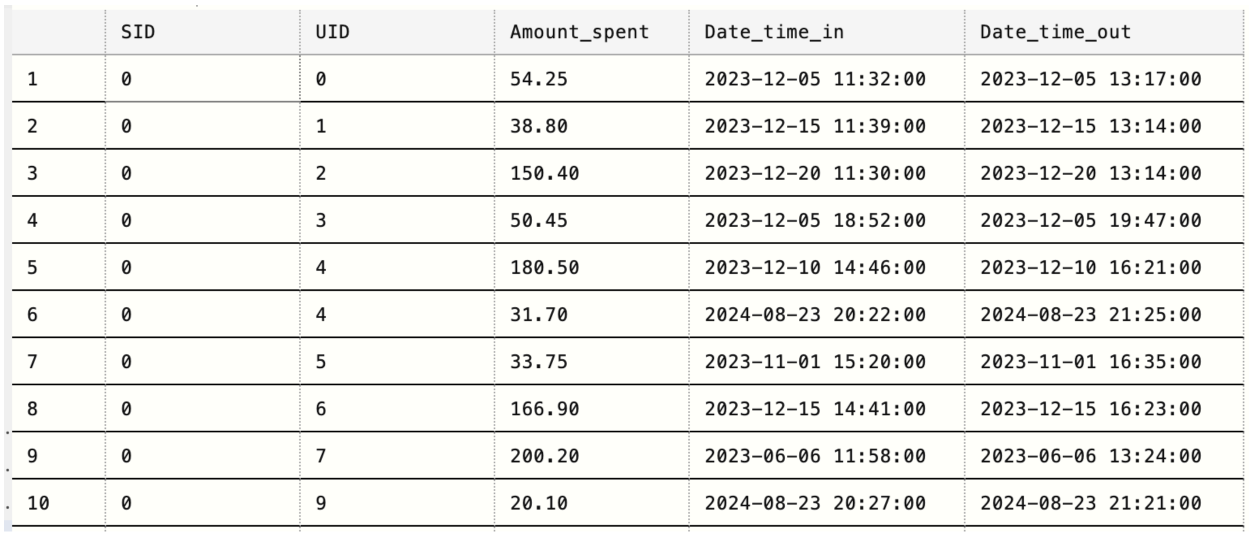
	OID	RID	MID
1	0	0	0
2	1	0	2
3	2	1	1
4	3	1	2
5	4	2	3
6	5	3	4
7	6	2	4
8	7	1	0
9	8	2	3

SHOP_NOUN (10 out of 15 rows)



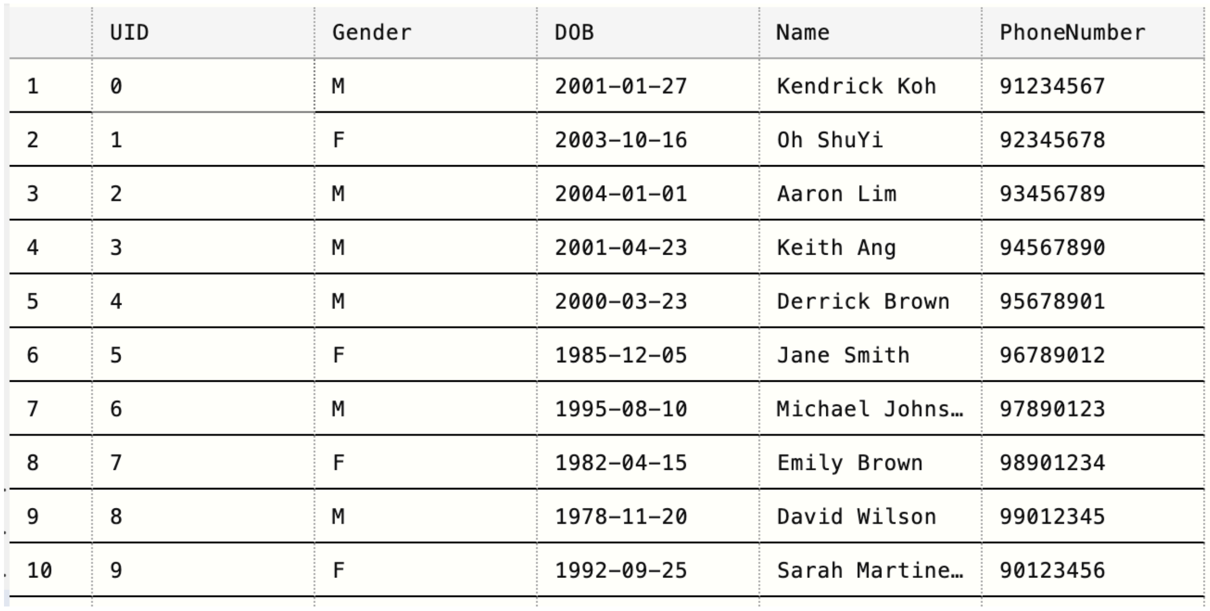
	SID	Type	MID
1	0	Cafe	0
2	1	Restaurant	0
3	2	Cafe	2
4	3	Cafe	3
5	4	Restaurant	4
6	5	Cafe	0
7	6	Cafe	0
8	7	Restaurant	1
9	8	Restaurant	1
10	9	Restaurant	2

SHOP_VERB (10 out of 195 rows)



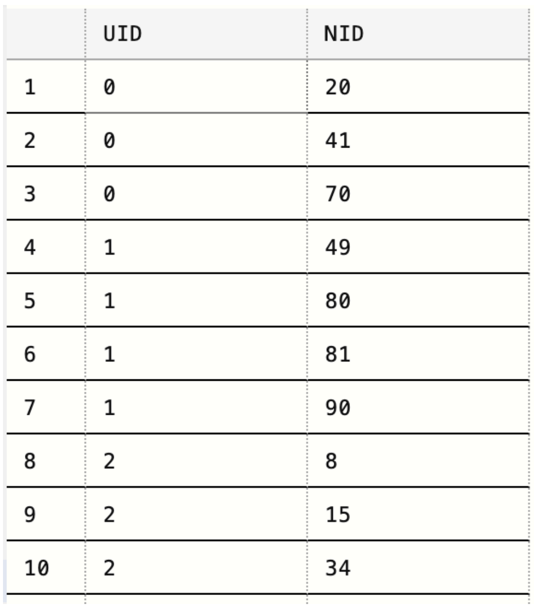
	SID	UID	Amount_spent	Date_time_in	Date_time_out
1	0	0	54.25	2023-12-05 11:32:00	2023-12-05 13:17:00
2	0	1	38.80	2023-12-15 11:39:00	2023-12-15 13:14:00
3	0	2	150.40	2023-12-20 11:30:00	2023-12-20 13:14:00
4	0	3	50.45	2023-12-05 18:52:00	2023-12-05 19:47:00
5	0	4	180.50	2023-12-10 14:46:00	2023-12-10 16:21:00
6	0	4	31.70	2024-08-23 20:22:00	2024-08-23 21:25:00
7	0	5	33.75	2023-11-01 15:20:00	2023-11-01 16:35:00
8	0	6	166.90	2023-12-15 14:41:00	2023-12-15 16:23:00
9	0	7	200.20	2023-06-06 11:58:00	2023-06-06 13:24:00
10	0	9	20.10	2024-08-23 20:27:00	2024-08-23 21:21:00

USER_ACCOUNT (10 out of 51 rows)



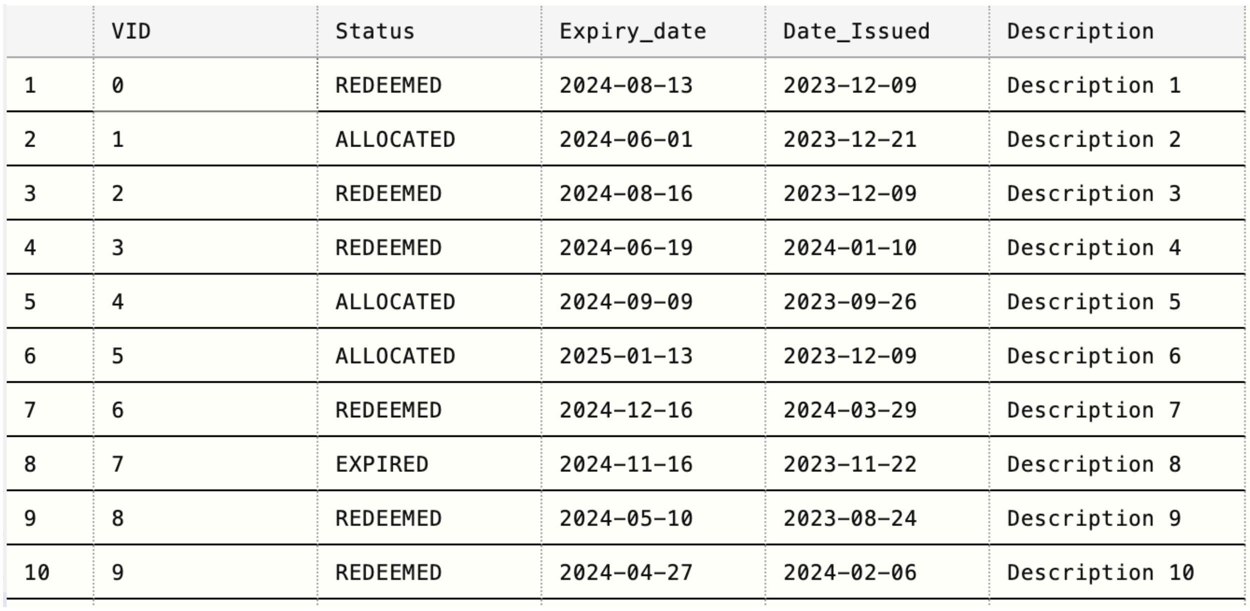
	UID	Gender	DOB	Name	PhoneNumber
1	0	M	2001-01-27	Kendrick Koh	91234567
2	1	F	2003-10-16	Oh ShuYi	92345678
3	2	M	2004-01-01	Aaron Lim	93456789
4	3	M	2001-04-23	Keith Ang	94567890
5	4	M	2000-03-23	Derrick Brown	95678901
6	5	F	1985-12-05	Jane Smith	96789012
7	6	M	1995-08-10	Michael Johns…	97890123
8	7	F	1982-04-15	Emily Brown	98901234
9	8	M	1978-11-20	David Wilson	99012345
10	9	F	1992-09-25	Sarah Martine…	90123456

USER_USE_RECO (10 out of 183 rows)



	UID	NID
1	0	20
2	0	41
3	0	70
4	1	49
5	1	80
6	1	81
7	1	90
8	2	8
9	2	15
10	2	34

VOUCHER (10 out of 100 rows)

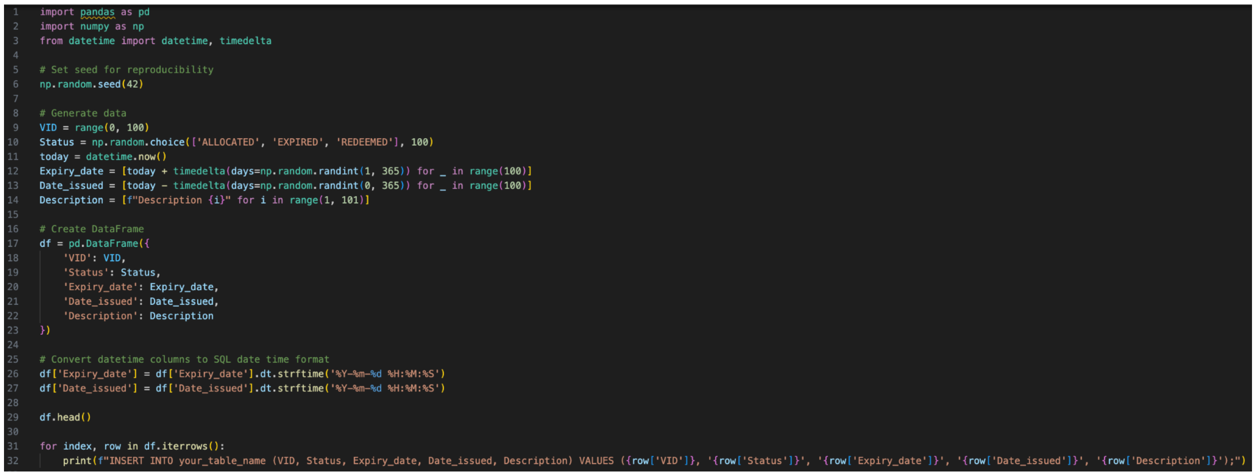


	VID	Status	Expiry_date	Date_Issued	Description
1	0	REDEEMED	2024-08-13	2023-12-09	Description 1
2	1	ALLOCATED	2024-06-01	2023-12-21	Description 2
3	2	REDEEMED	2024-08-16	2023-12-09	Description 3
4	3	REDEEMED	2024-06-19	2024-01-10	Description 4
5	4	ALLOCATED	2024-09-09	2023-09-26	Description 5
6	5	ALLOCATED	2025-01-13	2023-12-09	Description 6
7	6	REDEEMED	2024-12-16	2024-03-29	Description 7
8	7	EXPIRED	2024-11-16	2023-11-22	Description 8
9	8	REDEEMED	2024-05-10	2023-08-24	Description 9
10	9	REDEEMED	2024-04-27	2024-02-06	Description 10

Additional Efforts

- Used CHECK to simulate enums, to prevent invalid entries for gender and status.
- Used CHECK to prevent invalid values for PhoneNumber (80000000 to 99999999), AmountSpent (>=0), NumShops (>=0), Discounts (>0 & <100), Group_size (>1).
- Used CHECK to ensure dates are valid input (Date_time_in <= Date_time_out, Date_Issued <= Expiry_date/Valid_period).
- Auto increment counters (IDENTITY(1,1)) for some IDs (this cannot be implemented for other IDs due to other assumptions/constraints).
- Used python to generate fake records to populate tables.

E.g. Voucher Table Population



```python
import pandas as pd
import numpy as np
from datetime import datetime, timedelta

# Set seed for reproducibility
np.random.seed(42)

# Generate data
VID = range(0, 100)
Status = np.random.choice(['ALLOCATED', 'EXPIRED', 'REDEEMED'], 100)
today = datetime.now()
Expiry_date = [today + timedelta(days=np.random.randint(1, 365)) for _ in range(100)]
Date_issued = [today - timedelta(days=np.random.randint(0, 365)) for _ in range(100)]
Description = [f"Description {i}" for i in range(1, 101)]

# Create DataFrame
df = pd.DataFrame({
    'VID': VID,
    'Status': Status,
    'Expiry_date': Expiry_date,
    'Date_issued': Date_issued,
    'Description': Description
})

# Convert datetime columns to SQL date time format
df['Expiry_date'] = df['Expiry_date'].dt.strftime('%Y-%m-%d %H:%M:%S')
df['Date_issued'] = df['Date_issued'].dt.strftime('%Y-%m-%d %H:%M:%S')

df.head()

for index, row in df.iterrows():
    print(f"INSERT INTO your_table_name (VID, Status, Expiry_date, Date_issued, Description) VALUES ({row['VID']}, '{row['Status']}', '{row['Expiry_date']}', '{row['Date_issued']}', '{row['Description']}');")
```

- Exported tables to google sheets for neater table records submission.
- Provided step by step explanation with intermediate tables to illustrate thought process to solve queries for more complicated queries.

APPENDIX C: INDIVIDUAL CONTRIBUTION FORM

Full Name	Individual Contribution to Lab 1 Submission	Percentage of Contribution	Signature
Chan Fun Soon Nicholas	create recommendation & voucher entity, proposed new framework for insights entity	14.28%	[illegible]
Soh Shing Hui	create insights & added vouchers subclasses, proposed new relation framework	14.28%	[illegible]
Aaron Jerome Lim Li Yang	created user entity and relations, checked for RI, added user attributes and cleaned diagram	14.28%	[illegible]
Keith Ang Kee Chun	create shops entity and chatgpt entity, identified wrong arrows	14.28%	[illegible]
Koh Yihao Kendrick	create mall entity & identified weak shop entity, refined mall chain entity	14.28%	[illegible]
Oh ShuYi	create complaints & day package entity, helped with recommendation entity, cleaned diagram.	14.28%	[illegible]
Tan Jun Kiat	create restaurant chains refined user entity & refine visit entity to a relation	14.28%	[illegible]

Full Name	Individual Contribution to Lab 3 Submission	Percentage of Contribution	Signature
Chan Fun Soon Nicholas	R1 to R3, assumptions and explanations	14.28%	[illegible]
Soh Shing Hui	R4 to R6, assumptions and explanations	14.28%	[illegible]
Aaron Jerome Lim Li Yang	R7 to R9, assumptions and explanations	14.28%	[illegible]
Keith Ang Kee Chun	R10 to R12, 3NF decomposition	14.28%	[illegible]
Koh Yihao Kendrick	R13 to R15, 3NF decomposition	14.28%	[illegible]
Oh ShuYi	R16 to R18, 3NF decomposition	14.28%	[illegible]
Tan Jun Kiat	R19 to R22, checked final relational schema	14.28%	[illegible]

Full Name	Individual Contribution to Lab 5 Submission	Percentage of Contribution	Signature
Chan Fun Soon Nicholas	Query 4 solution and relevant table population	14.28%	[illegible]
Soh Shing Hui	Query 2 solution and relevant table population	14.28%	[illegible]
Aaron Jerome Lim Li Yang	Query 3 solution and relevant table population	14.28%	[illegible]
Keith Ang Kee Chun	Query 5 solution and relevant table population	14.28%	[illegible]
Koh Yihao Kendrick	Query 6 solution and relevant table population	14.28%	[illegible]
Oh ShuYi	Query 1 solution and relevant table population	14.28%	[illegible]
Tan Jun Kiat	Database and all tables creation with constraints	14.28%	[illegible]

APPENDIX D: USE OF AI TOOL(S) IN LAB WORK

Each team member should indicate either A or B:

A. I affirm that my contribution(s) to the lab work is my own, produced without help from any AI tool(s).
B. I affirm that my contribution(s) to the lab work has been produced with the use of AI tool(s).

Team member (full name)	Signature	Date	A or B
Chan Fun Soon Nicholas	[illegible]	2/4/24	A
Soh Shing Hui	[illegible]	2/4/24	A
Aaron Jerome Lim Li Yang	[illegible]	2/4/24	A
Keith Ang Kee Chun	[illegible]	2/4/24	A
Koh Yihao Kendrick	[illegible]	2/4/24	A
Oh ShuYi	[illegible]	2/4/24	A
Tan Jun Kiat	[illegible]	2/4/24	A

By signing this form, you declare that the above affirmation made is true and that you have read and understood NTU's policy on the use of AI tools.

If any team member answered B, the team member(s) must indicate and replicate the table below for every instance AI tool(s) is used:

Name of AI tool	< For example, ChatGPT >
Input prompt	< Insert the question that you asked ChatGPT >
Date generated	
Output generated	< Insert the response verbatim from ChatGPT >
Output screenshots	
Impact on submission	< Briefly explain which part of your submitted work was ChatGPT's response applied >