

**Lab 5: Final demonstration**

SC2207 - Introduction to Databases

**SCSY Team 5**

| **Name** | **Matriculation Number** |
| --- | --- |
| Gauthaman Ramya Varshini | U2123023K |
| Pearlina Tan Qinlin | U2221690F |
| Chin Ao-Wen | U2240595B |
| Tan Jie Ning, Jolynn | U2220577B |
| Sum Yuan Sen | U2221346D |
| Vilan Chan | U2020092L |

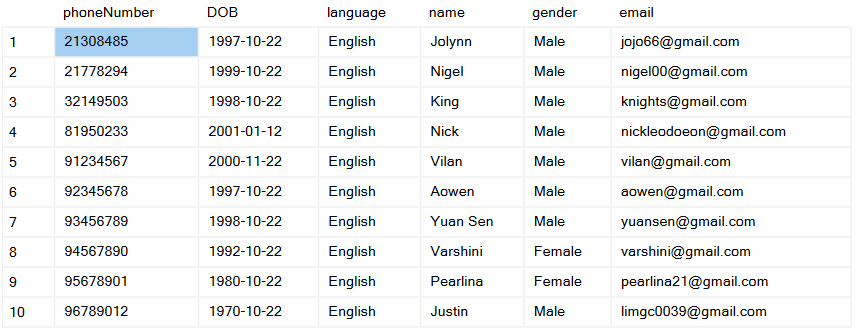
### 

| **SQL DDL Commands for Table Creation** **& Table Outputs** |
| --- |

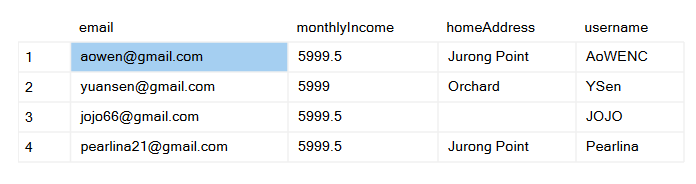
**Relation 1: User**(phoneNumber, DOB, language, name, gender, email, monthlyIncome, homeAddress, username)

**Table 1: Users**

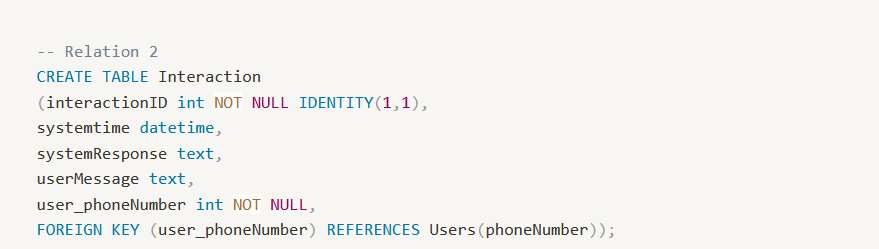
****

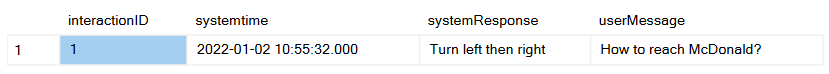
****

**Table 2: UsersInfo**

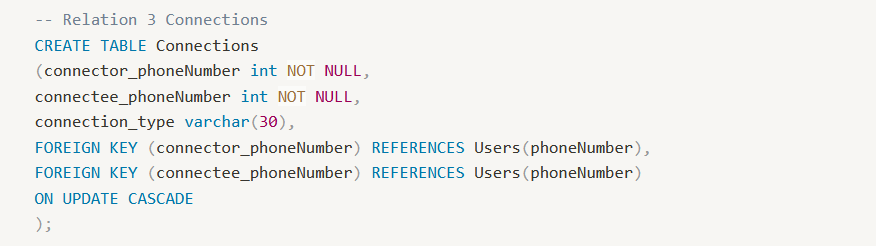
****

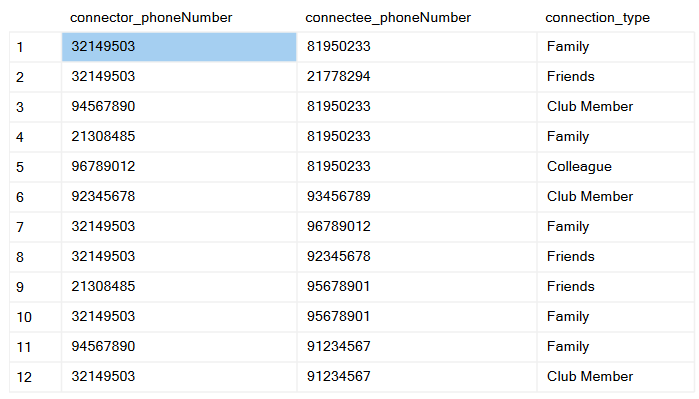
**Relation 2: Interaction** (interactionID, time, systemResponse, userMessage, user\_phoneNumber)



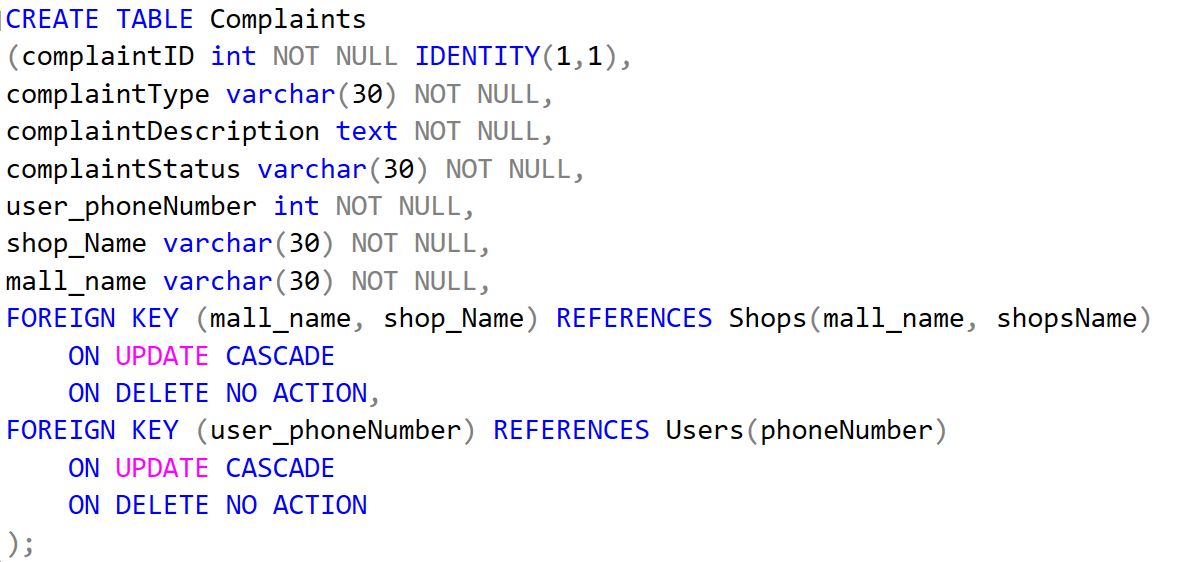


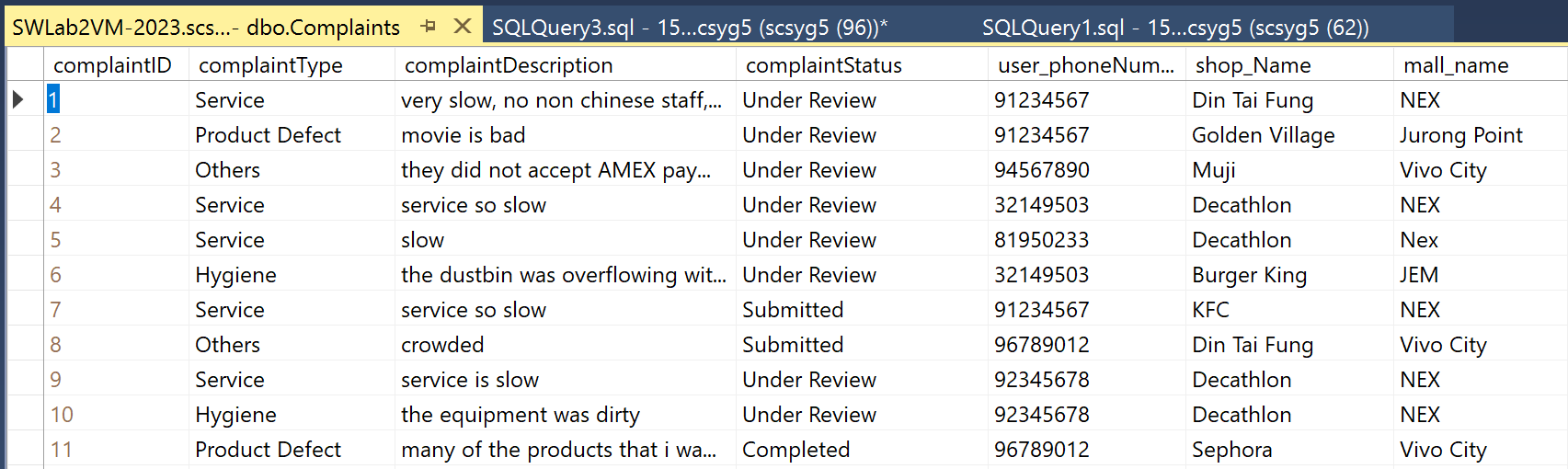
**Relation 3: Connections**(connector\_phoneNumber, connectee\_phoneNumber, type)



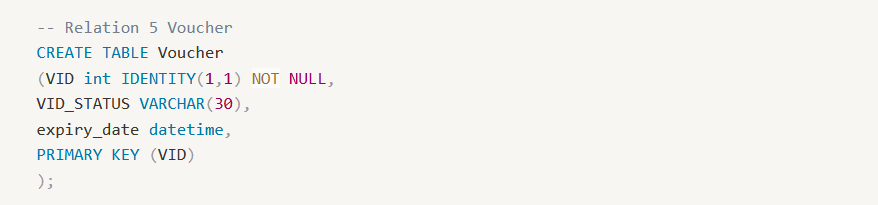
****

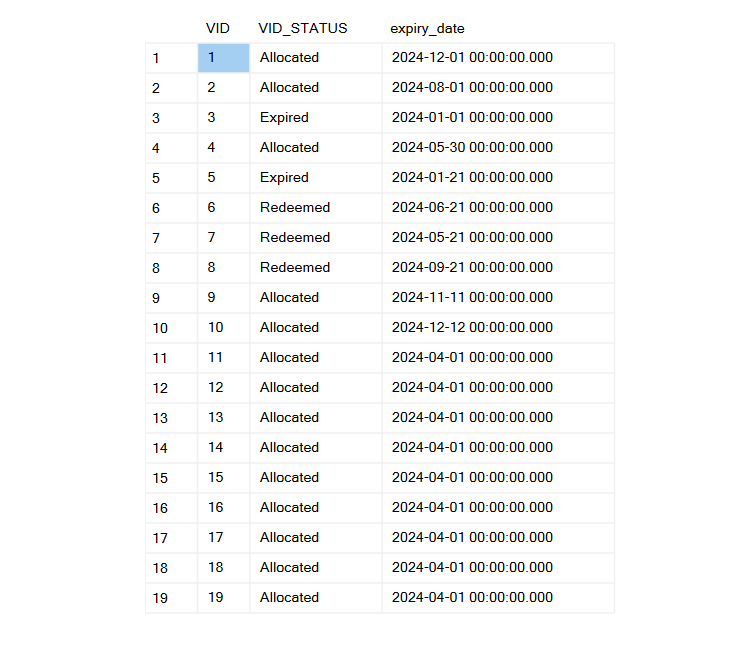
**Relation 4**: **Complaints** (complaintID, complaintType, complaintDescription, complaintStatus, user\_phoneNumber, shop\_Name, mall\_name)





**Relation 5:** **Voucher** (VID, status, expiryDate)

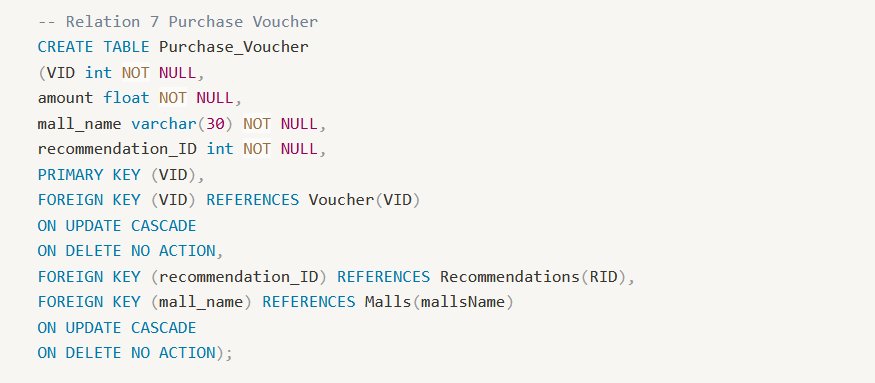


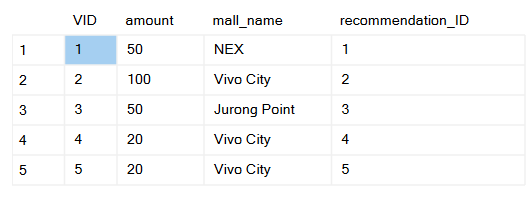
****

**Relation 6:** **Discount Voucher** (voucher\_VID, amount, mall\_name)

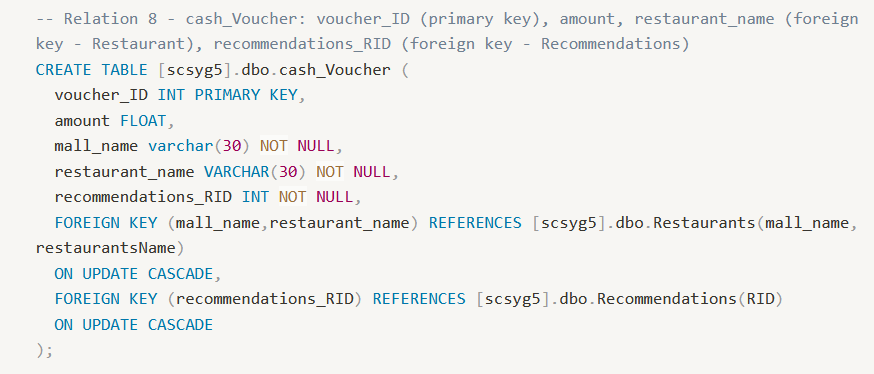
|  |  |
| --- | --- |

**Relation 7:** **Purchase Voucher**(voucher\_VID, amount, mall\_name, recommendations\_RID)



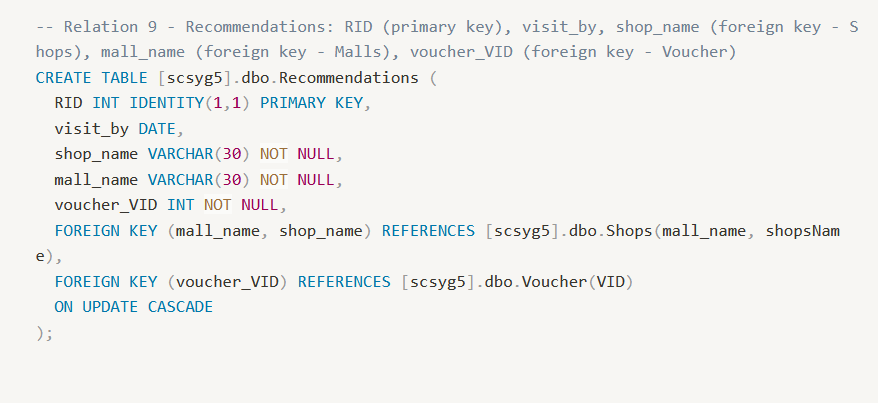
****

**Relation 8: Cash Voucher**(voucher\_ID, amount, restaurant\_name, recommendations\_RID)

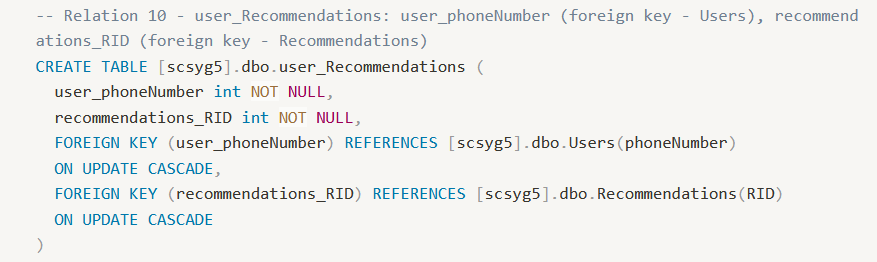


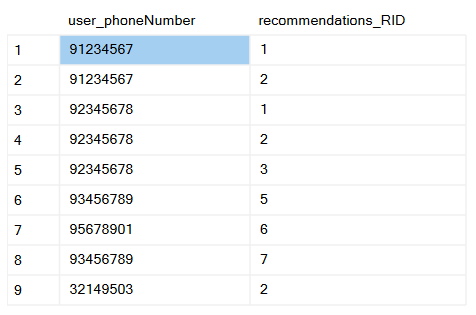
****

**Relation 9: Recommendations** (RID, visitBy, shop\_name, mall\_name, voucher\_VID)

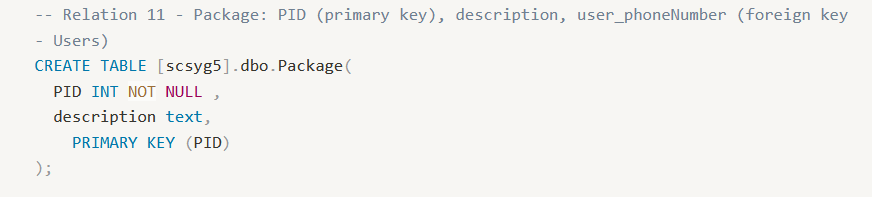


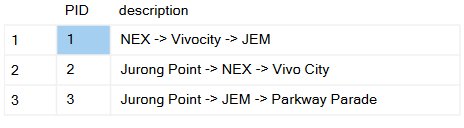
****

**Relation 10**: **user\_Recommendations** (user\_phoneNumber, recommendations\_RID)

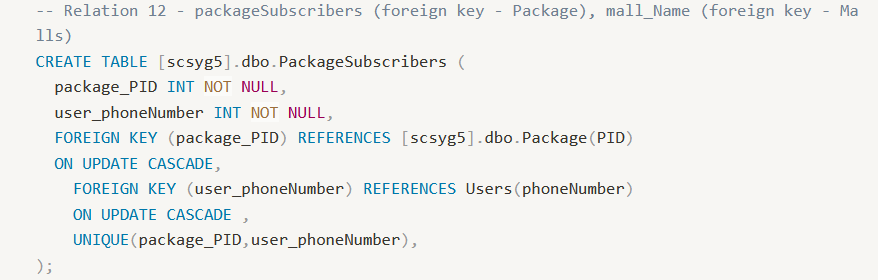


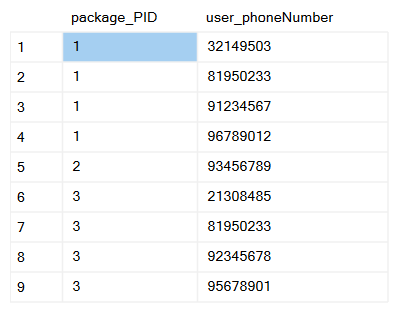
**Relation 11: Package** (PID, description, user\_phoneNumber)



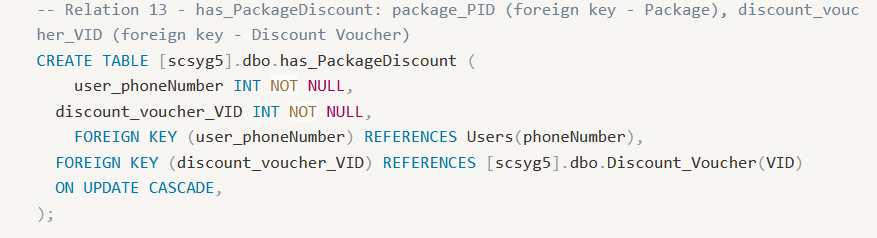
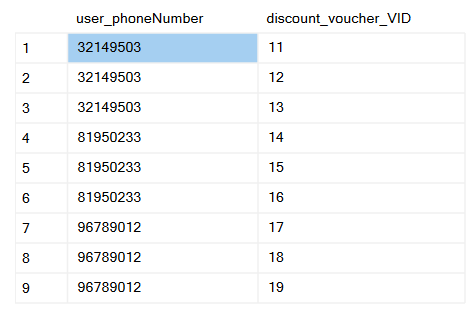
****

**Relation 12:** **has\_PackageMall** (package\_PID, mall\_Name)





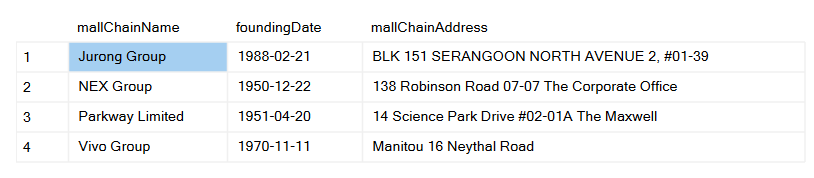
**Relation 13:** **has\_PackageDiscount** (package\_PID, discount\_voucher\_VID)

****

**Relation 14:** **Mall Chain**(name, foundingDate, contactInfo, address)

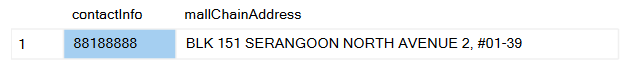
**Table 1: Mall Chain**



****

**Table 2: Mall Chain Info**

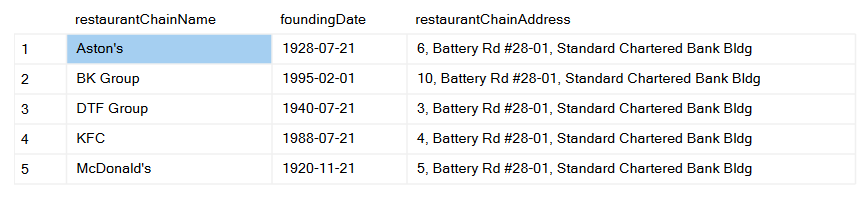
****

****

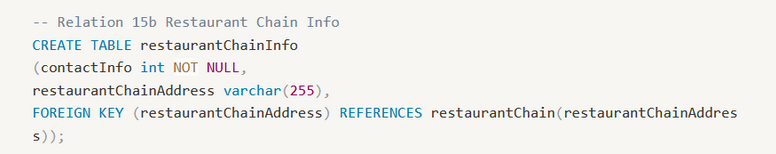
**Relation 15: Restaurant Chain**(name, contactInfo, foundingDate, address)

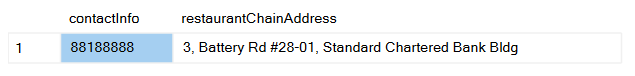
**Table 1: Restaurant Chain**



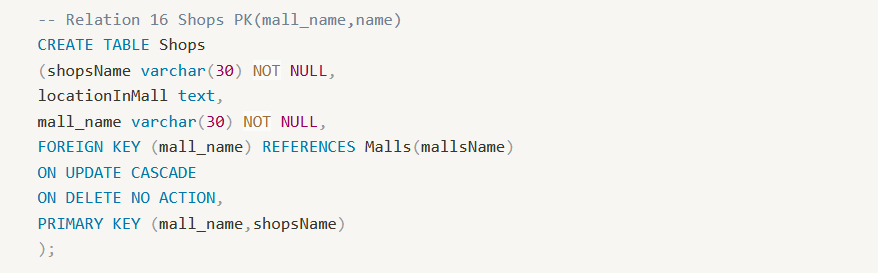
****

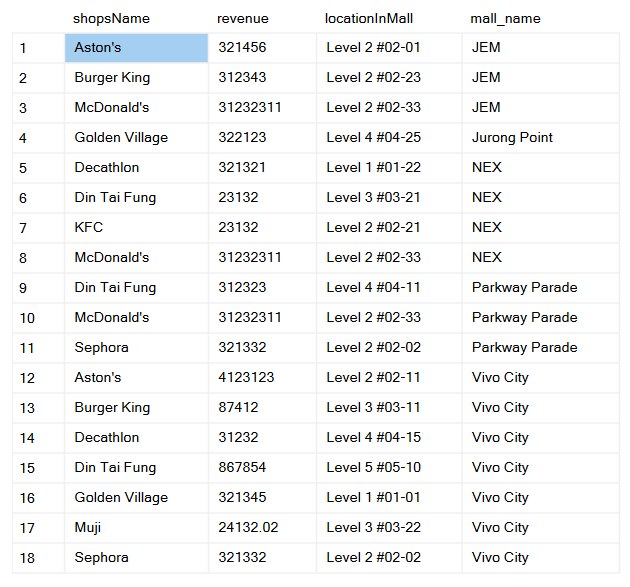
**Table 2: Restaurant Chain Info**

****

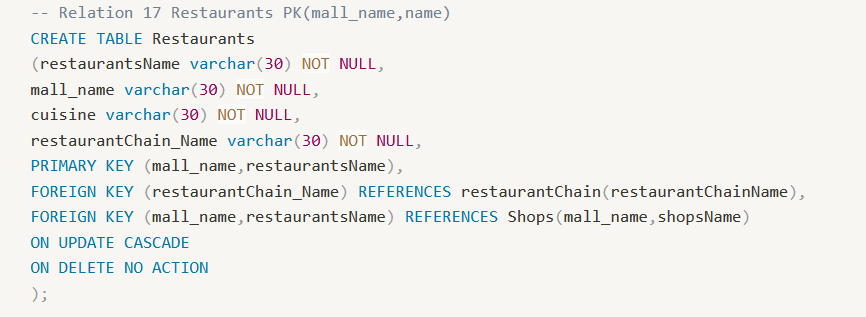
****

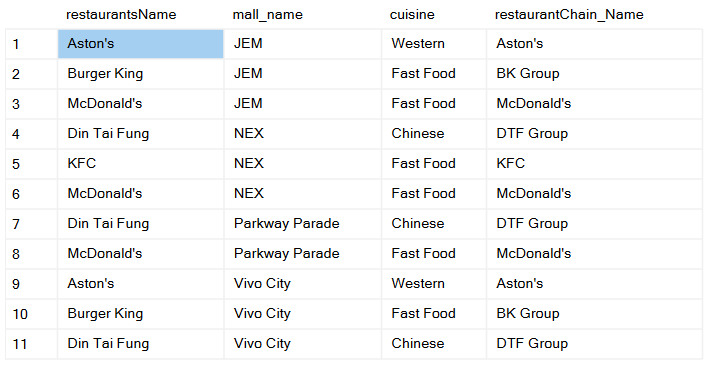
**Relation 16: Shops** (shopsName, locationInMall, revenue mall\_name)



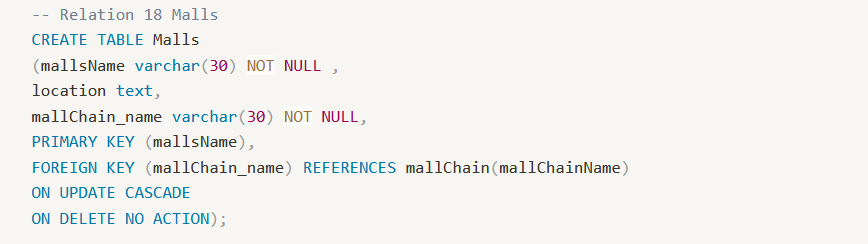
****

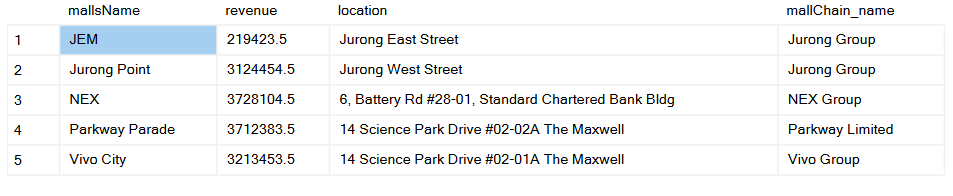
**Relation 17: Restaurant** (restaurantsName, cuisine, restaurantChain\_name, mall\_name)

****

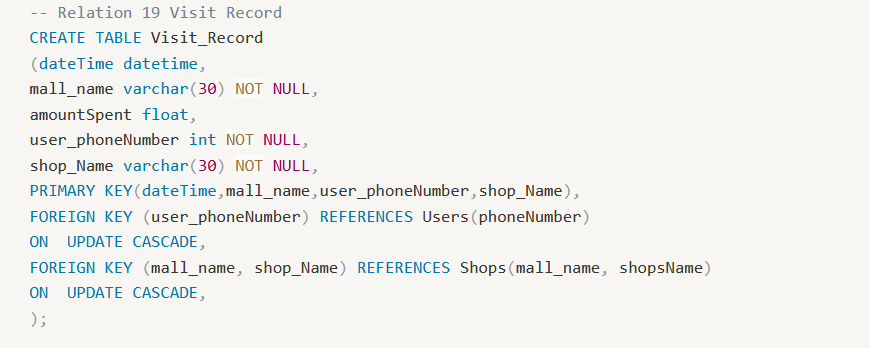
****

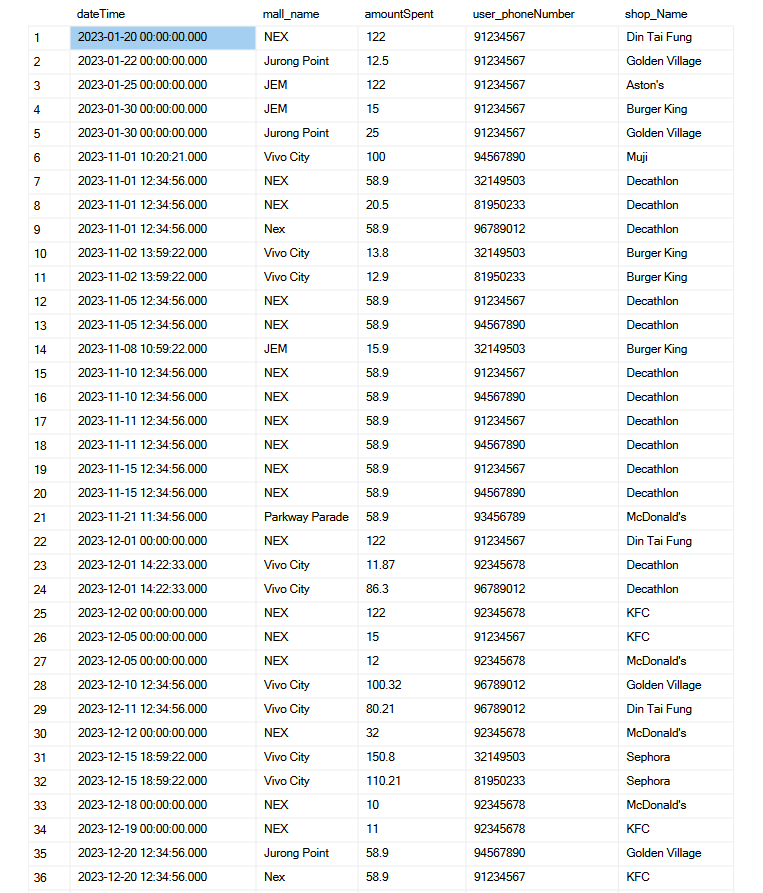
**Relation 18: Malls** (mallsName, location, revenue mallChain\_name)



****

**Relation 19: Visit Record** (dateTime, amountSpent, mall\_name, user\_phoneNumber, shop\_name)

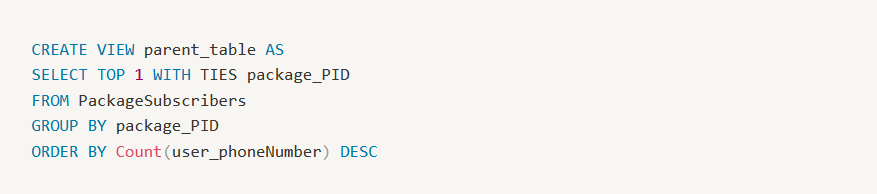




### **SQL Statements to Solve the Queries**

**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.**

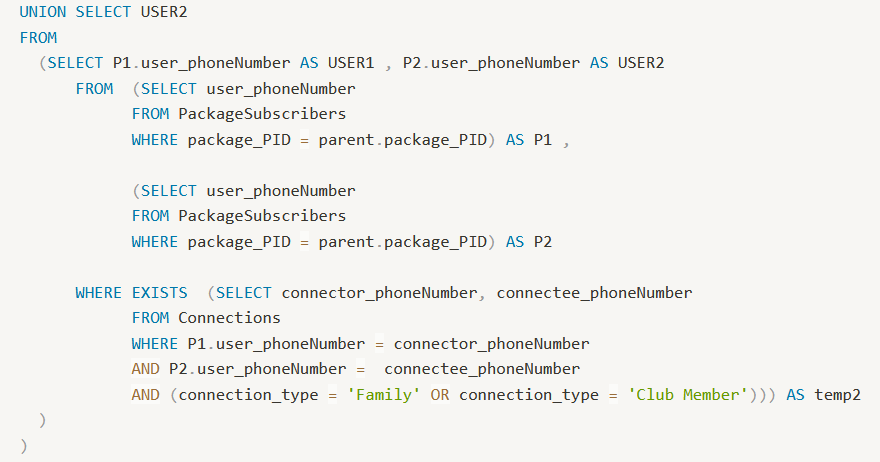
**Creation of Views**

****

The parent\_table finds and returns the package\_PID with the highest subscriber count. If there is more than 1 maximum PID, the SELECT TOP WITH TIES will return them all.

**SQL Query**

****

****

The query involves the usage of a correlated query. For easier visualization, the query can be seen as follows:

SELECT \*

FROM parent\_table

WHERE NOT EXISTS (

**A** EXCLUDE **B**

);

In **A**, the query finds all package subscribers subscribing to the max PID from parent\_table. On the other hand, **B** will then get all package subscribers and search through the Connection table. The result of **B** will return subscribers from **A** who are actually related to one another as a family or club member. Thus, if A EXCLUDE B returns any row, it means that NOT all users in A are related to one another as a family or club member.

**Query Output:**

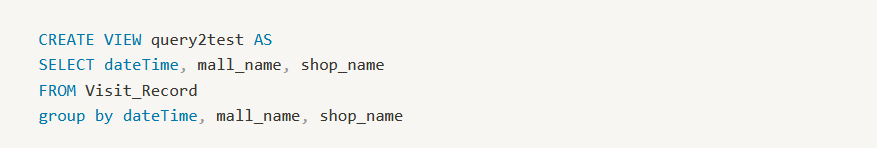
****

**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.**

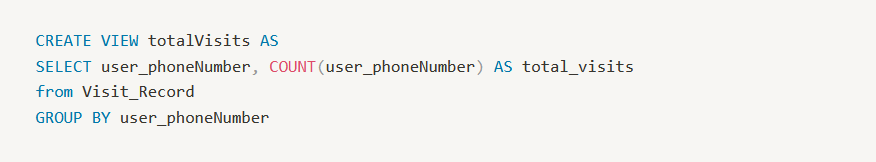
We will be comparing the amount of times a user went out and visited shops with their family members and the amount of times they went out by themselves, including times they went out with the family members. The output of the query will be family visits >50% of total visits.

**Creating Views:**

The query2test table finds and returns all the dateTime, mall\_name and shop\_name in the visit record table group by dateTime, mall\_name and shop\_name.

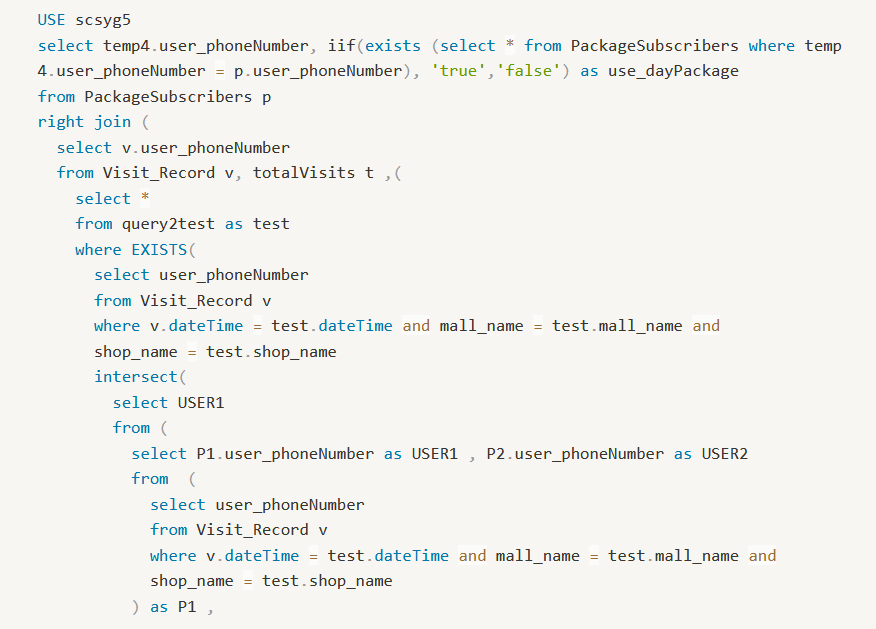


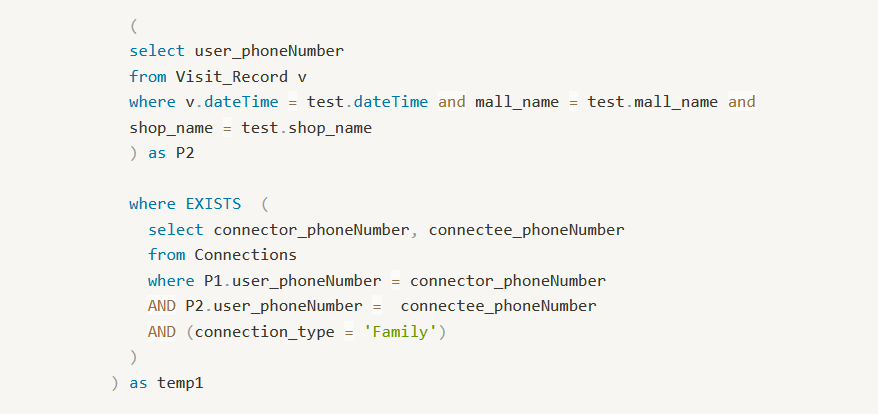
The totalVisits table finds and returns all the phone numbers and the amount of times they visited the shops.

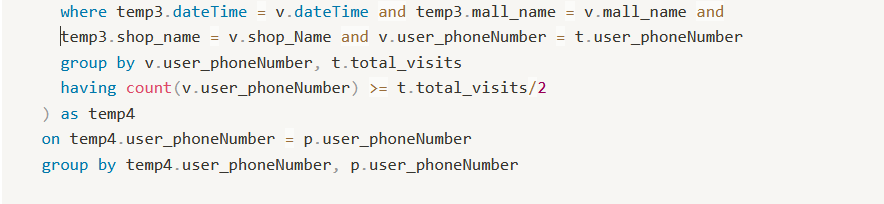


|  |  |
| --- | --- |

**SQL Query:**







**SQL Query Explanation:**

The query uses multiple subqueries. The first part that will be explained is the simplified to:

Select \* from query2test as test

Where exists (

**A** includes **B**

) as temp3

In **A**, the query finds all users that visited a shop in a mall at a certain time in the query2test, **B** will then get all the users and search through the Connection table. The result of **B** will return users from **A** who are actually related to one another as a family.

Next part will be for temp3:

SELECT v.phoneNumber

FROM visit\_record v, total\_visit, (subquery: temp3)

AS temp4

This part of the query is to compare the visit records mall\_name, dateTime and shop\_name with the one in the subquery, which gives us all the records of phone numbers that visited shops with families. Here we can do a Count the record in this query with the one in total\_visits to check if the phone number is more than 50%.

Lastly we join with the Package\_Subscriber table to check if the user is using day packages.

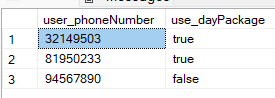
temp4.user\_phoneNumber, (iif for day\_package)

FROM PackageSubcriber p

Right join (subquery temp4)

Here we use the function iif to check if there are any day package records returned by using the phone number from subquery temp4. If no records return, the value will be false which means they didn't use any day package and return true when they used day packages.

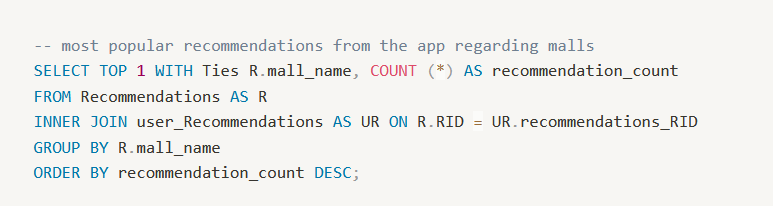
**Query Output:**



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

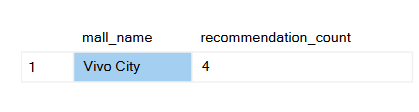
**SQL Query**

We need to count the number of times each distinct mall was recommended. The COUNT(\*) function is used to count the number of rows returned by the query for each distinct mall name. We group the results by the mall\_name column.



**Query Output**

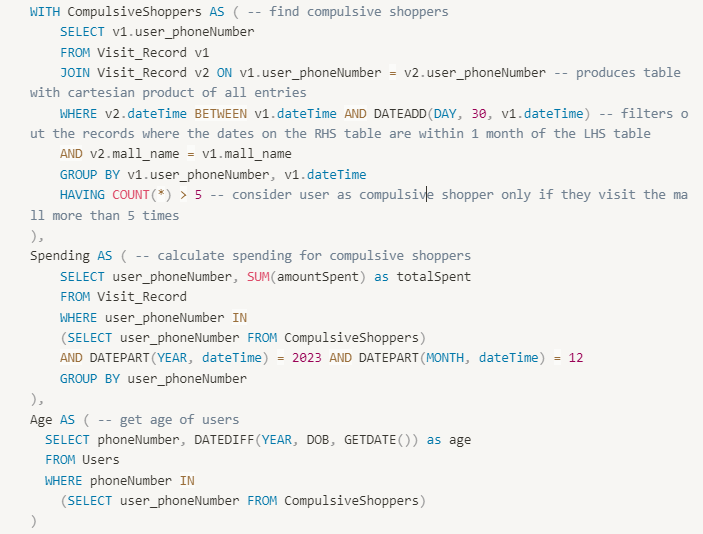
This means that the count of recommendations is calculated separately for each unique mall name. The TOP 1 WITH TIES clause ensures that if multiple malls have the same highest recommendation count, they are all included in the result set. Hence, by looking at the table record, we can conclude that VivoCity is the most popular recommendation from the app regarding malls.



**4. Find the youngest compulsive shoppers (visiting a mall more than 5 times within a certain period of time) and the amount they spent in total during December 2023.**

Firstly, we define the period of time to be 30 days (approximately 1 month) - if a user visits a certain mall more than 5 times in 30 days, he is considered to be a compulsive shopper. We create 3 temporary views:

1. CompulsiveShoppers: identifies compulsive shoppers using a self join, which creates a table of cartesian products consisting of combinations between each date, for each phone number. This is filtered for records where the date in the RHS table is within 30 days after the date in the LHS table. This results in a table of phone numbers which have at least 6 records in the following 30 days of any of its dates.
2. Spending: calculates the total amount spent per user, filtered for users who are found to be compulsive shoppers in the previous temp view, as well as spending days within December 2023.
3. Age: calculates the age of the users who were identified to be compulsive shoppers.



Finally, the intended output columns, phone number, name, total spent (in Dec 2023), and age, are selected, by joining the Spending and Age temporary views on the Users table, filtering the ages to match the youngest age in the final table.

****

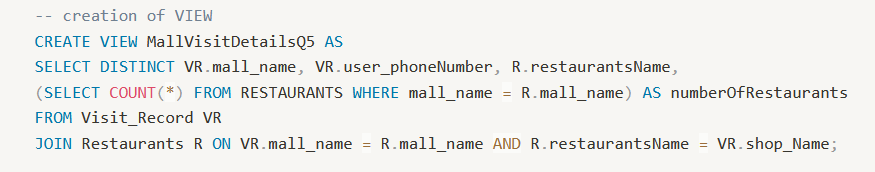
**Query Output**



The output from the query shows that there is one 24 year old user Vilan who has spent $195.90.

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

First, we create views to show the unique (distinct) restaurants and malls each user (user\_phoneNumber) visited at least once, and to make sure that there are no duplicates.





**SQL Query & Output**

Then, we check to see if the total number of restaurants visited (A) for a particular mall matches the total number of restaurants in that mall (B).

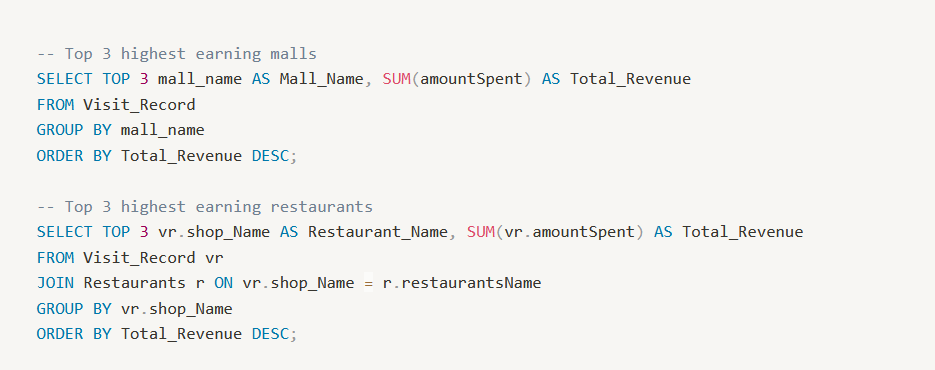
* If A > B, that means restaurants from other malls were visited.
* If B > A, that means not all restaurants in that mall were visited.

|  |  |
| --- | --- |

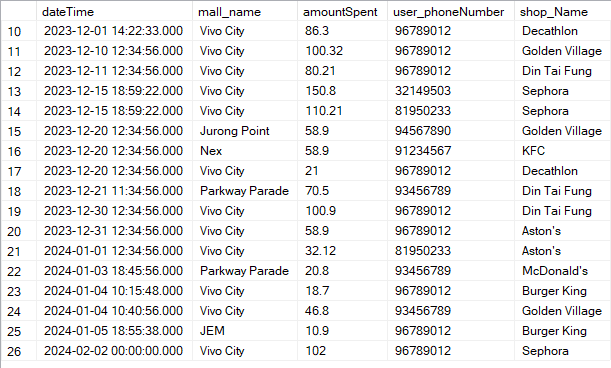
**6. What are the top 3 highest earning malls and restaurants?**

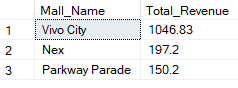
In our Visit\_Record table, we have the column amountSpent which gives us the amount different users spend at different malls and shops. Thus, to output the top 3 highest earning malls is pretty straightforward - we just calculate the sum of amountSpent for each mall, and output the results in descending order.

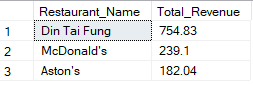
However, for the top 3 highest earning restaurants, we need to make sure that the query only considers shops that are restaurants. To do this, we join the Visit\_Record and Restaurants tables. Then, we do the same thing as before to output the top 3 highest earning restaurants.



**Query Output**







### 

### **Extra Query**

1. **Find restaurants that are more popular for young adults (18 to 35 years old) than for older adults (50 years or older).**

First we use a temporary view ‘AgeGroup’ to classify each restaurant visit as being visited by either someone aged ‘18-35’ or ‘>50’. Then, we use another temporary view ‘Counts’ to count the number of times each age group visits each restaurant. Lastly, we filter out the records where there were more people aged ‘18-35’ as compared to those aged ‘>50’.

****

**Query Output**

****

### 

### **Additional Effort**

1. **Analyze the sentiment of user complaints submitted through the app to identify areas for improvement.**

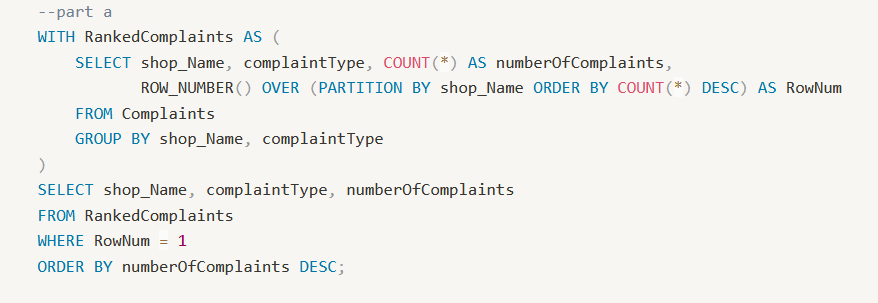
From the Complaints table, we first separate the user feedback to be grouped by:

a) Highest number of complaintType complaints for each shop

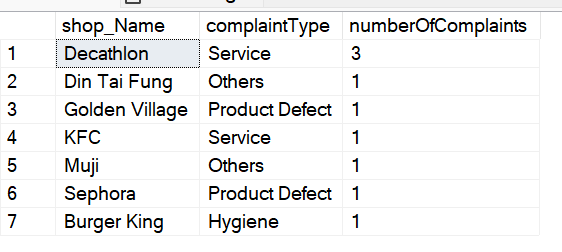
b) Complaints grouped by types and their mall.

Highest number of complaintType complaints for each shop

From the Complaints table, sum the number of complaints and rank each shop by the number of complaints using the ROW\_NUMBER().From the temporary view RankedComplaints, select shop\_Name, complaintType and number of complaints.



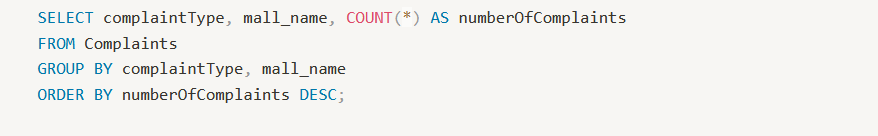
**Query output**

****

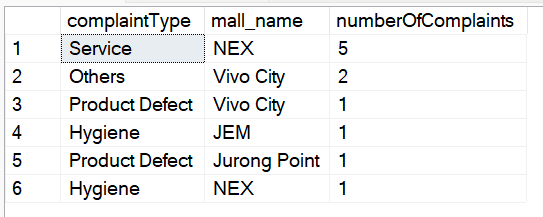
From this we are able to conclude the complaintTypes that each shop has to improve on. For example, the complaintType with the highest number of complaints for Decathlon is Service, and for Burger King is Hygiene Thus, Decathlon should work on improving its services, and Burger King should work on Hygiene.

**Complaints grouped by types their mall**

From the Complaints table, sum the number of complaints for each complaintType for each mall. Then, filter and group by the complaintType and mall name.



**Query Output**



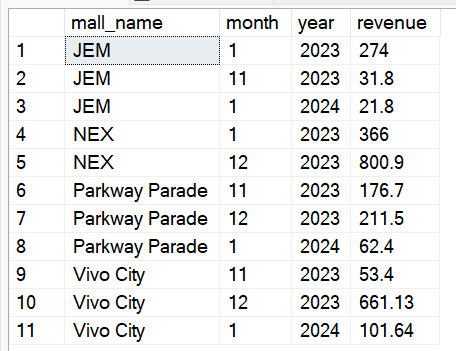
From this we can conclude the recurring common problem in each mall. For example, the biggest problem for the mall NEX is Service since Service complaintType has the highest numberOfComplaints within NEX and amongst all the malls. Thus, NEX should work on Service.

1. **Highest revenue for each mall per month**

From the Visit\_Record table, we first sum the amount spent at each restaurant for each month in each mall. Then, we ranks these revenue values within each mall-month combination using the ROW\_NUMBER(). Then we select the rows where the rank is equal to 1, indicating the highest revenue for each mall-month combination, and order the results by mall name, year, and month.

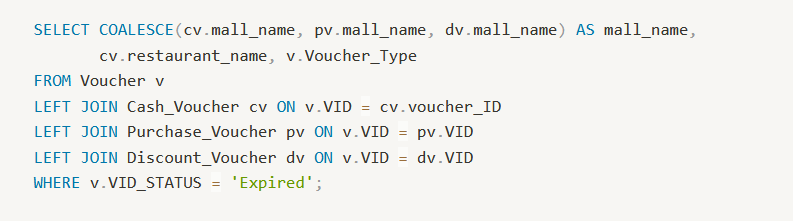


**Query output**

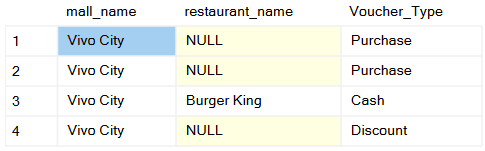
****

1. **What is the most unpopular voucher?**

In order to show the most unpopular vouchers, we output the vouchers which include the purchase voucher, cash voucher and discount vouchers that have the VID\_STATUS ‘Expired’. We came to the conclusion that there is not one unpopular voucher type, but that all the unpopular vouchers seem to have the same pattern which is that they come from the same mall - Vivo City.

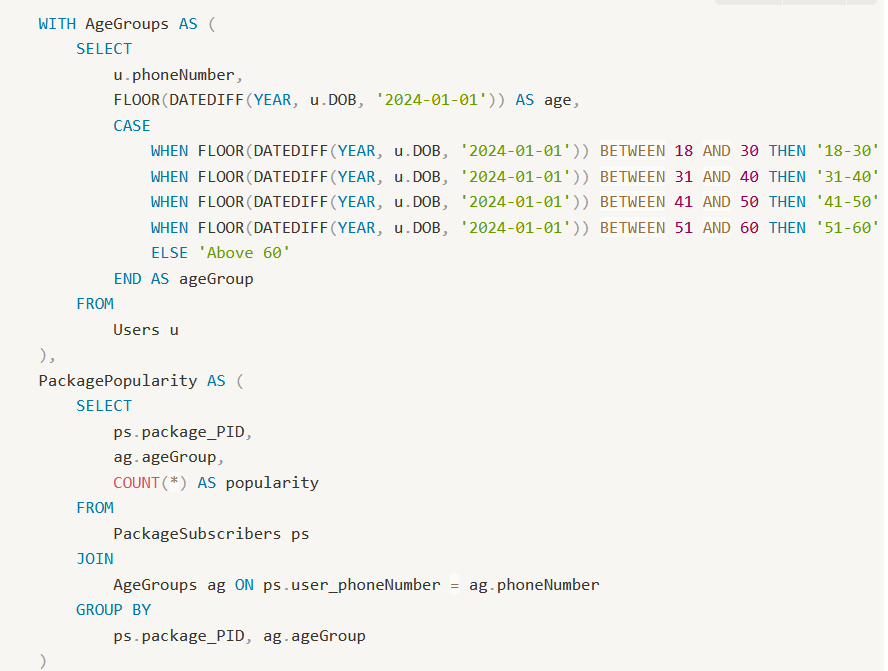


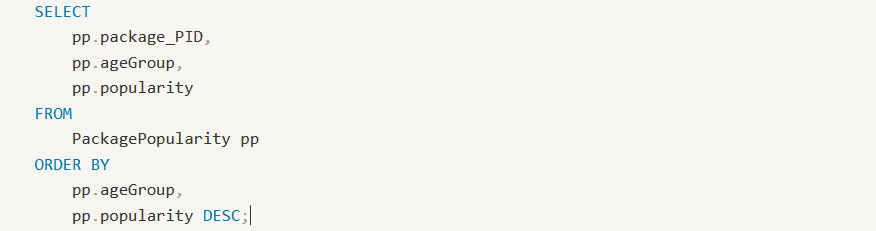
**Query Output**



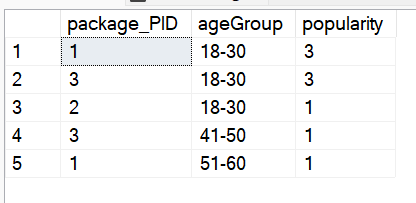
**4. Determine the popularity of each package within different age groups.**

We first categorize users into age groups based on their dates of birth, then Count the number of users subscribing to each package within each age group. We join the User and PackageSubscribers tables, grouping the data by package ID and age group, and counting the occurrences. The table presented the package ID, age group, and the corresponding popularity count. From the table, we can conclude that ageGroup of 18-30 years old are more likely to subscribe for the packages.





**Query Output:**



**Appendix C: Individual Contribution Form**

| **Full Name** | **Individual Contribution to Lab 5 Submission** | **Percentage of Contribution** | **Signature** |
| --- | --- | --- | --- |
| Gauthaman Ramya Varshini | Contributed to providing for 1 of the required queries | 16.67% |  |
| Pearlina Tan Qinlin | Contributed to one of the required queries and doing up the additional effort | 16.67% |  |
| Chin Ao-Wen | Creation of table and providing the query for 1 of the required queries. | 16.67% |  |
| Tan Jie Ning, Jolynn | Contributed to the creation of table, providing for 1 of the required queries, and doing up the additional effort. | 16.67% |  |
| Sum Yuan Sen | Contributed to providing the query for 1 of the required queries | 16.67% |  |
| Vilan Chan | Contributed creation of table and providing for 1 of the required queries | 16.67% |  |

**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 |
| --- | --- | --- | --- |
| Gauthaman Ramya Varshini |  | 3/3/24 | A |
| Pearlina Tan Qinlin |  | 3/3/24 | A |
| Chin Ao-Wen |  | 3/3/24 | A |
| Tan Jie Ning, Jolynn |  | 3/3/24 | A |
| Sum Yuan Sen |  | 3/3/24 | A |
| Vilan Chan |  | 3/3/24 | 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 > |