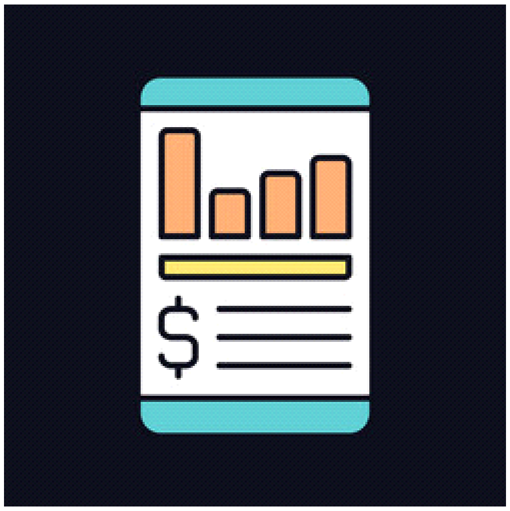
FINANCE TRACKING SYSTEM



ABSTRACT

The Finance Tracking System is a versatile tool that can benefit individuals, small businesses and organizations seeking to improve their financial management practices. By harnessing the capabilities of SQL, this project offers a scalable and adaptable solution for tracking, analyzing and optimizing financial resources.

It is a comprehensive SQL-based project designed to streamline and enhance financial management processes for individuals and organizations. In today's dynamic financial landscape, efficient tracking and management of financial data are crucial for making informed decisions, maintaining fiscal responsibility and achieving long-term financial goals.

AIM OF PROJECT-

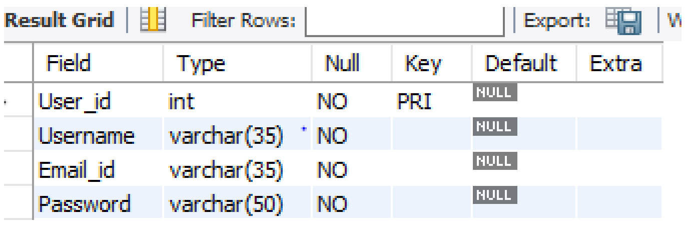
This project aims to create a robust and user-friendly finance system by leveraging the power of Structured Query Language (SQL). The System's primary objectives include data collection, storage, retrieval and analysis of financial information in a secure and efficient manner.

INTRODUCTION -

Finance is at the heart of nearly every aspect of modern life, whether it be personal budgeting, corporate financial planning, or nonprofit fund management. The Finance Tracking System SQL recognizes the need for precision, transparency, and accessibility in managing financial data. The Finance Tracking System represents a pivotal solution designed to address the intricate demands of financial tracking and management in a world driven by data.

Structure of Table

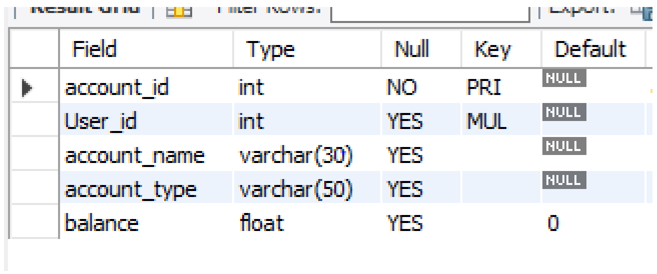
Users-



Result Grid | Filter Rows: | Export:

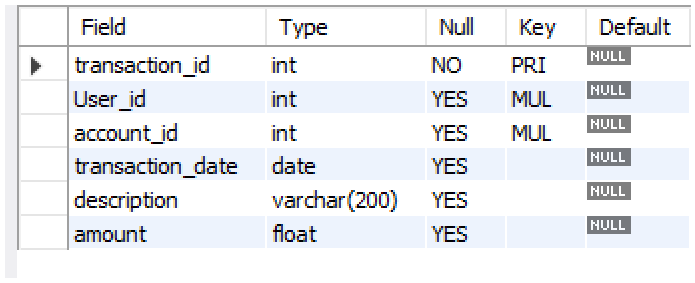
Field	Type	Null	Key	Default	Extra
User_id	int	NO	PRI	NULL	
Username	varchar(35)	NO		NULL	
Email_id	varchar(35)	NO		NULL	
Password	varchar(50)	NO		NULL	

Accounts-



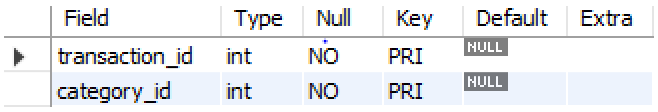
Field	Type	Null	Key	Default
account_id	int	NO	PRI	NULL
User_id	int	YES	MUL	NULL
account_name	varchar(30)	YES		NULL
account_type	varchar(50)	YES		NULL
balance	float	YES		0

Transactions-



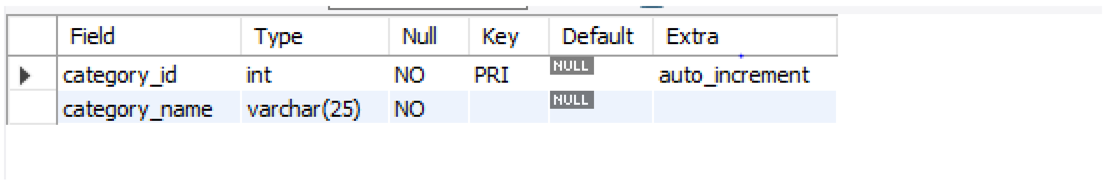
Field	Type	Null	Key	Default
transaction_id	int	NO	PRI	NULL
User_id	int	YES	MUL	NULL
account_id	int	YES	MUL	NULL
transaction_date	date	YES		NULL
description	varchar(200)	YES		NULL
amount	float	YES		NULL

Transaction_Categories -



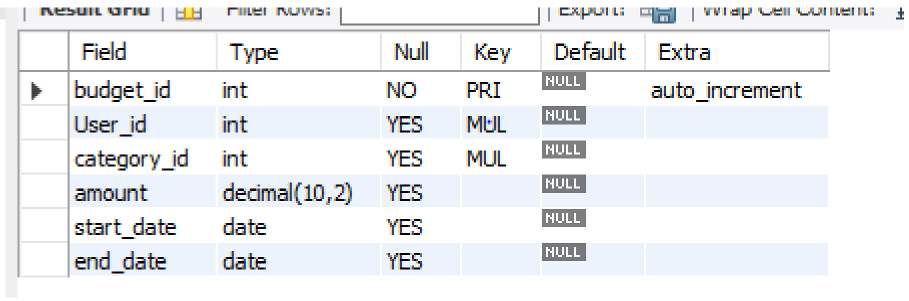
Field	Type	Null	Key	Default	Extra
transaction_id	int	NO	PRI	NULL	
category_id	int	NO	PRI	NULL	

Categories-



Field	Type	Null	Key	Default	Extra
category_id	int	NO	PRI	NULL	auto_increment
category_name	varchar(25)	NO		NULL	

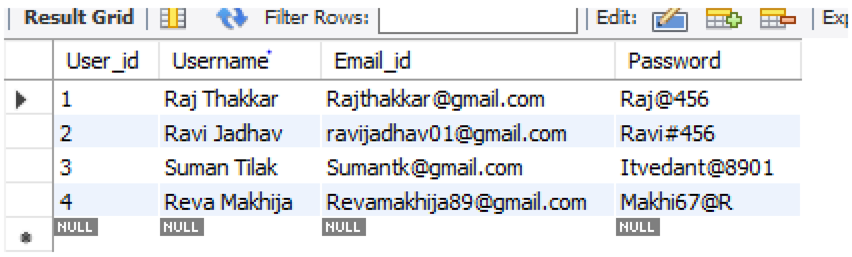
Budgets-



Field	Type	Null	Key	Default	Extra
budget_id	int	NO	PRI	NULL	auto_increment
User_id	int	YES	MUL	NULL	
category_id	int	YES	MUL	NULL	
amount	decimal(10,2)	YES		NULL	
start_date	date	YES		NULL	
end_date	date	YES		NULL	

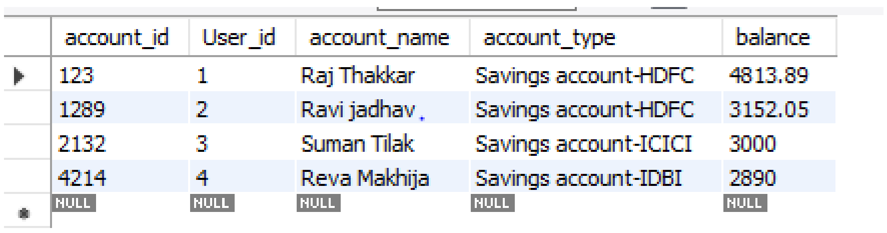
Contents of Table

Users-



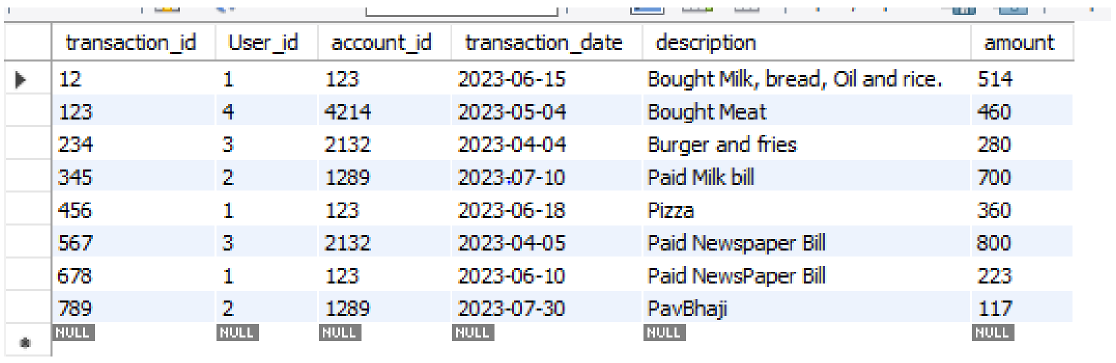
User_id	Username	Email_id	Password
1	Raj Thakkar	Rajthakkar@gmail.com	Raj@456
2	Ravi Jadhav	ravijadhav01@gmail.com	Ravi#456
3	Suman Tilak	Sumantk@gmail.com	Itvedant@8901
4	Reva Makhija	Revamakhija89@gmail.com	Makhi67@R
NULL	NULL	NULL	NULL

Accounts-



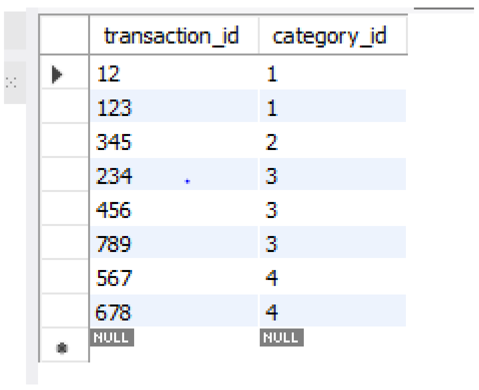
account_id	User_id	account_name	account_type	balance
123	1	Raj Thakkar	Savings account-HDFC	4813.89
1289	2	Ravi jadhav	Savings account-HDFC	3152.05
2132	3	Suman Tilak	Savings account-ICICI	3000
4214	4	Reva Makhija	Savings account-IDBI	2890
NULL	NULL	NULL	NULL	NULL

Transactions-



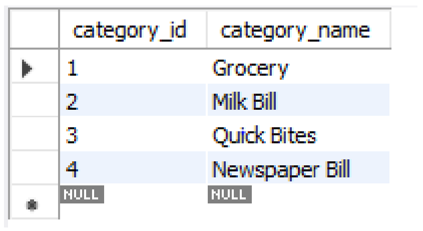
transaction_id	User_id	account_id	transaction_date	description	amount
12	1	123	2023-06-15	Bought Milk, bread, Oil and rice.	514
123	4	4214	2023-05-04	Bought Meat	460
234	3	2132	2023-04-04	Burger and fries	280
345	2	1289	2023-07-10	Paid Milk bill	700
456	1	123	2023-06-18	Pizza	360
567	3	2132	2023-04-05	Paid Newspaper Bill	800
678	1	123	2023-06-10	Paid NewsPaper Bill	223
789	2	1289	2023-07-30	PavBhaji	117
NULL	NULL	NULL	NULL	NULL	NULL

Transaction_Categories -



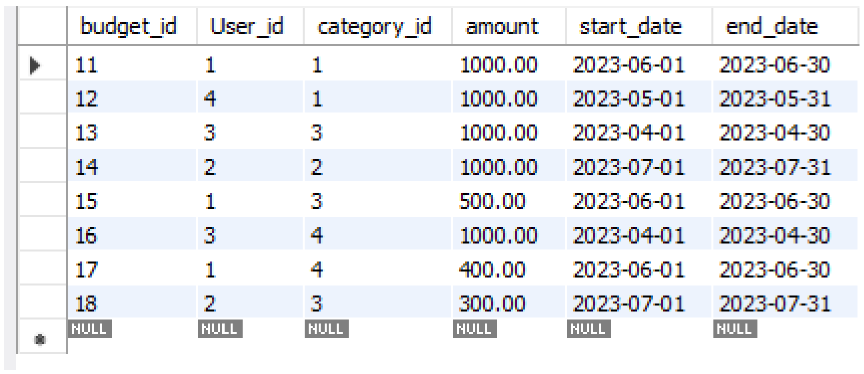
transaction_id	category_id
12	1
123	1
345	2
234	3
456	3
789	3
567	4
678	4
NULL	NULL

Categories-



category_id	category_name
1	Grocery
2	Milk Bill
3	Quick Bites
4	Newspaper Bill
NULL	NULL

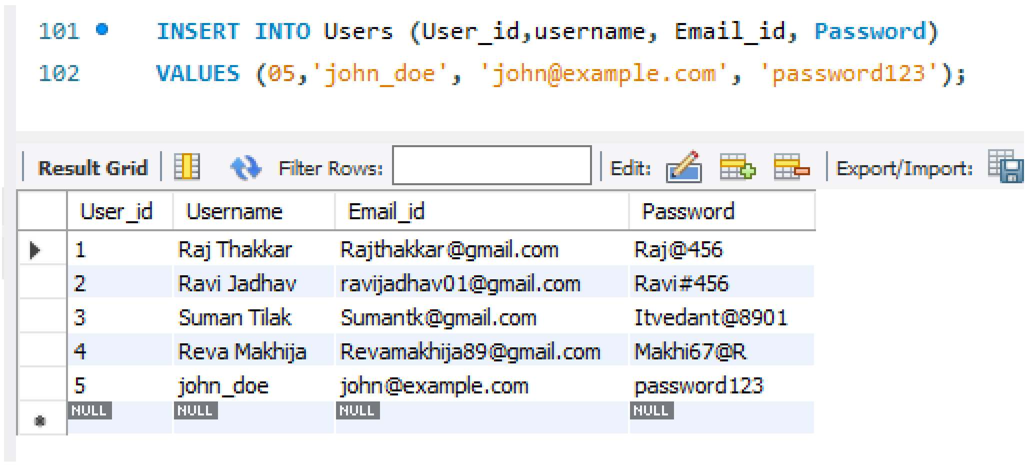
Budgets-



budget_id	User_id	category_id	amount	start_date	end_date
11	1	1	1000.00	2023-06-01	2023-06-30
12	4	1	1000.00	2023-05-01	2023-05-31
13	3	3	1000.00	2023-04-01	2023-04-30
14	2	2	1000.00	2023-07-01	2023-07-31
15	1	3	500.00	2023-06-01	2023-06-30
16	3	4	1000.00	2023-04-01	2023-04-30
17	1	4	400.00	2023-06-01	2023-06-30
18	2	3	300.00	2023-07-01	2023-07-31
NULL	NULL	NULL	NULL	NULL	NULL

SQL QUERIES :

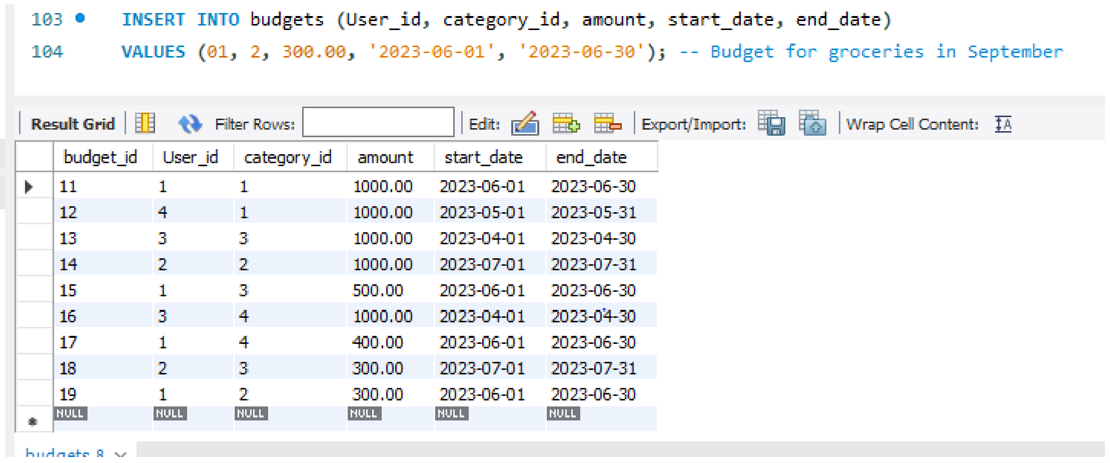
1)Insert a New User.



```sql
INSERT INTO Users (User_id,username, Email_id, Password)
VALUES (05,'john_doe', 'john@example.com', 'password123');
```

User_id	Username	Email_id	Password
1	Raj Thakkar	Rajthakkar@gmail.com	Raj@456
2	Ravi Jadhav	ravijadhav01@gmail.com	Ravi#456
3	Suman Tilak	Sumantk@gmail.com	Itvedant@8901
4	Reva Makhija	Revamakhija89@gmail.com	Makhi67@R
5	john_doe	john@example.com	password123
NULL	NULL	NULL	NULL

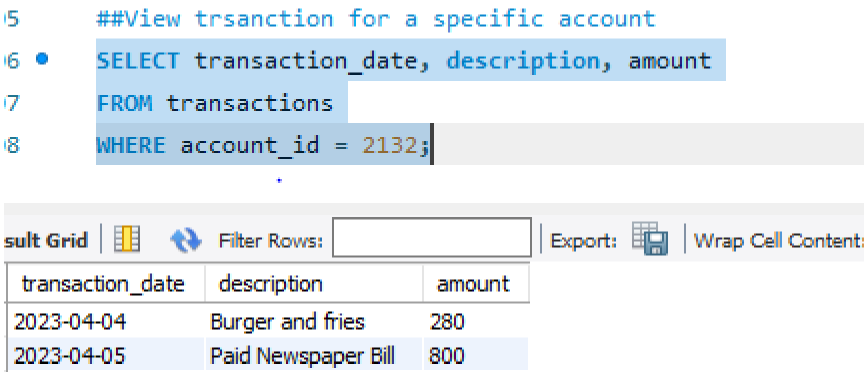
2)Create a new budget



```sql
INSERT INTO budgets (User_id, category_id, amount, start_date, end_date)
VALUES (01, 2, 300.00, '2023-06-01', '2023-06-30'); -- Budget for groceries in September
```

budget_id	User_id	category_id	amount	start_date	end_date
11	1	1	1000.00	2023-06-01	2023-06-30
12	4	1	1000.00	2023-05-01	2023-05-31
13	3	3	1000.00	2023-04-01	2023-04-30
14	2	2	1000.00	2023-07-01	2023-07-31
15	1	3	500.00	2023-06-01	2023-06-30
16	3	4	1000.00	2023-04-01	2023-04-30
17	1	4	400.00	2023-06-01	2023-06-30
18	2	3	300.00	2023-07-01	2023-07-31
19	1	2	300.00	2023-06-01	2023-06-30
NULL	NULL	NULL	NULL	NULL	NULL

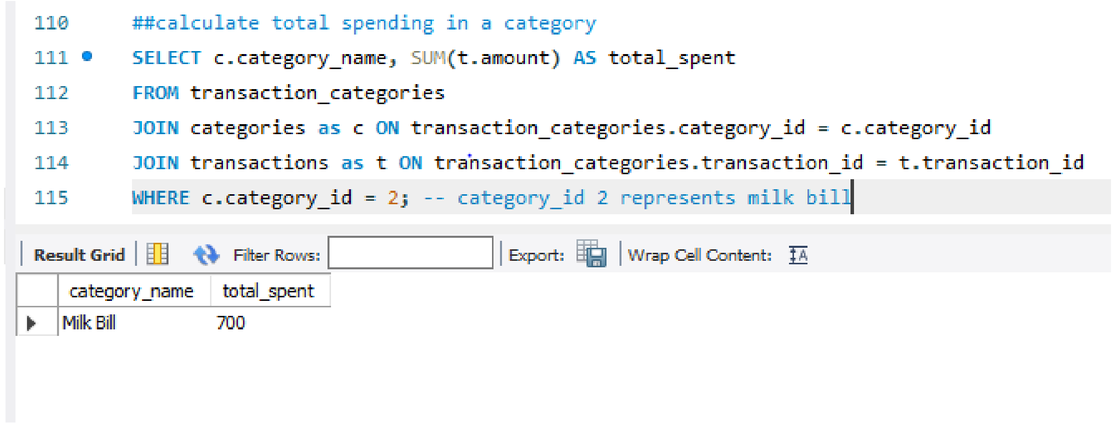
3)View Transaction for a specific Account



```
##View trsanction for a specific account
SELECT transaction_date, description, amount
FROM transactions
WHERE account_id = 2132;
```

transaction_date	description	amount
2023-04-04	Burger and fries	280
2023-04-05	Paid Newspaper Bill	800

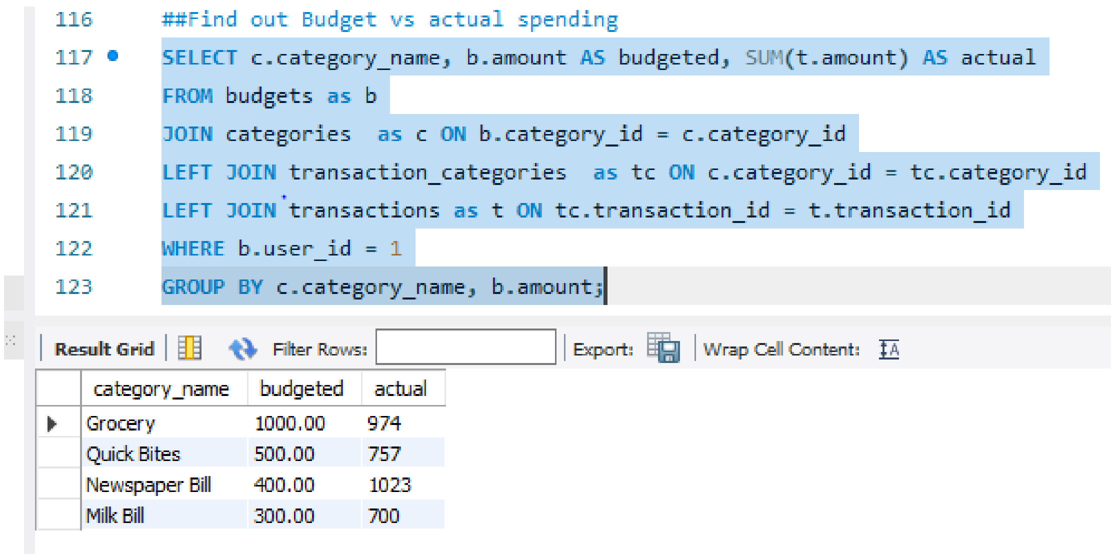
4)Calculate total spending in a category(JOIN)



```
##calculate total spending in a category
SELECT c.category_name, SUM(t.amount) AS total_spent
FROM transaction_categories
JOIN categories as c ON transaction_categories.category_id = c.category_id
JOIN transactions as t ON transaction_categories.transaction_id = t.transaction_id
WHERE c.category_id = 2; -- category_id 2 represents milk bill
```

category_name	total_spent
Milk Bill	700

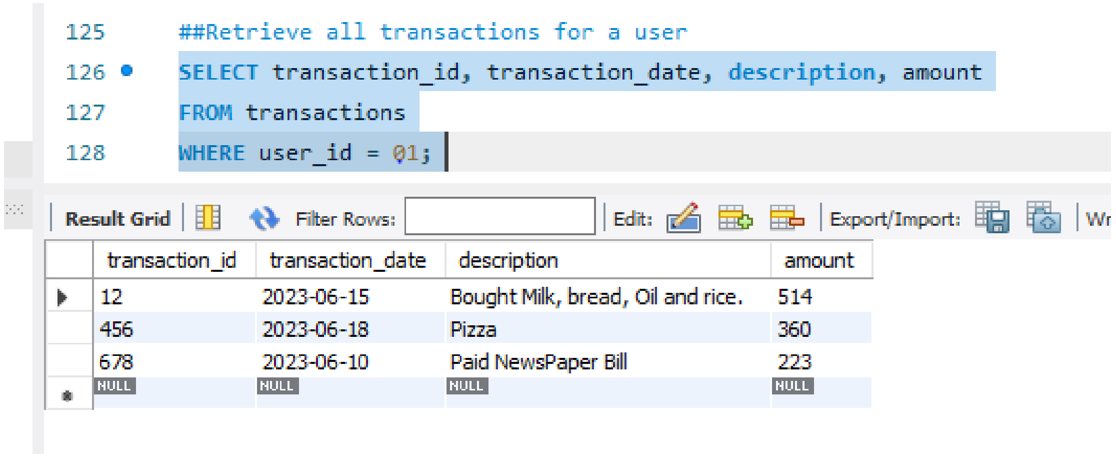
5)Find out budget vs actual spending(LEFT JOIN)



```
##Find out Budget vs actual spending
SELECT c.category_name, b.amount AS budgeted, SUM(t.amount) AS actual
FROM budgets as b
JOIN categories  as c ON b.category_id = c.category_id
LEFT JOIN transaction_categories  as tc ON c.category_id = tc.category_id
LEFT JOIN transactions as t ON tc.transaction_id = t.transaction_id
WHERE b.user_id = 1
GROUP BY c.category_name, b.amount;
```

category_name	budgeted	actual
Grocery	1000.00	974
Quick Bites	500.00	757
Newspaper Bill	400.00	1023
Milk Bill	300.00	700

6)Retrieve all transactions for a User

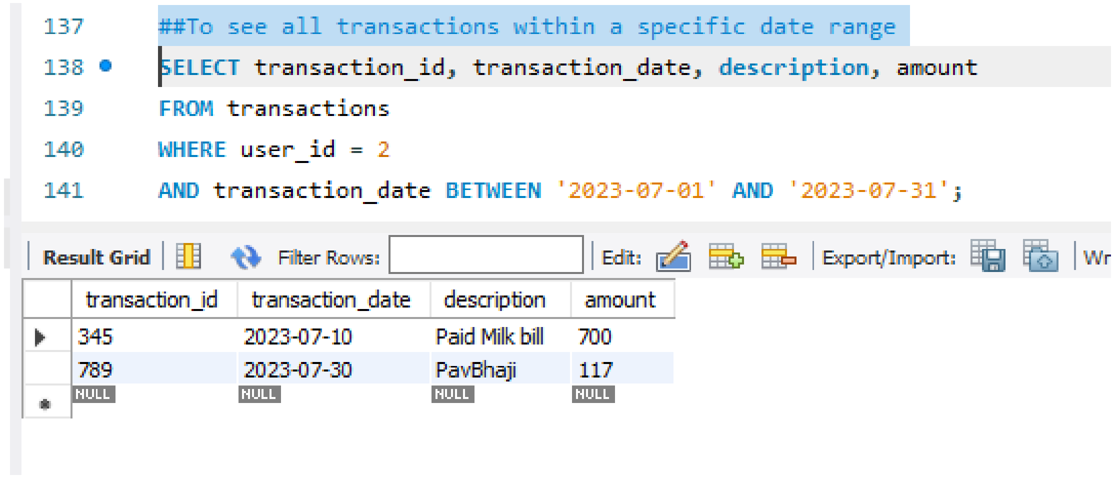


```
##Retrieve all transactions for a user
SELECT transaction_id, transaction_date, description, amount
FROM transactions
WHERE user_id = 01;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wr

transaction_id	transaction_date	description	amount
12	2023-06-15	Bought Milk, bread, Oil and rice.	514
456	2023-06-18	Pizza	360
678	2023-06-10	Paid NewsPaper Bill	223
NULL	NULL	NULL	NULL

7)Find out all transactions for a user withing specific date range.

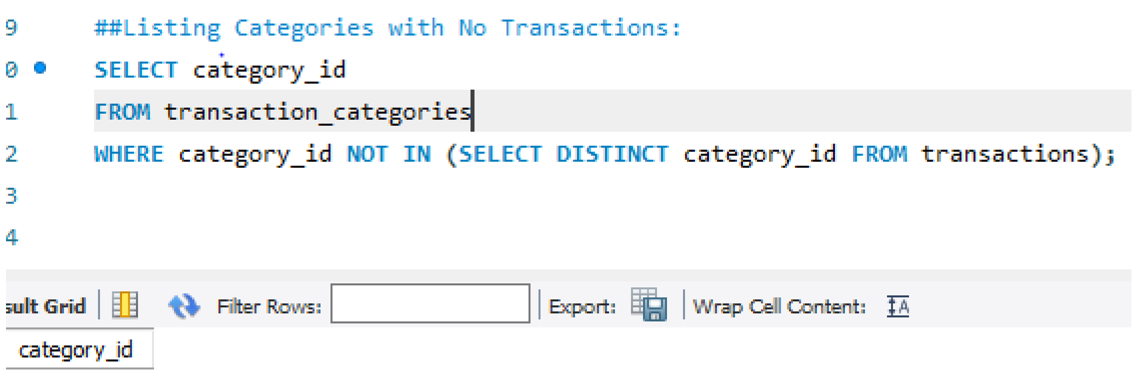


```
##To see all transactions within a specific date range
SELECT transaction_id, transaction_date, description, amount
FROM transactions
WHERE user_id = 2
AND transaction_date BETWEEN '2023-07-01' AND '2023-07-31';
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wr

transaction_id	transaction_date	description	amount
345	2023-07-10	Paid Milk bill	700
789	2023-07-30	PavBhaji	117
NULL	NULL	NULL	NULL

8)List categories with no transactions.(SUBQUERY)

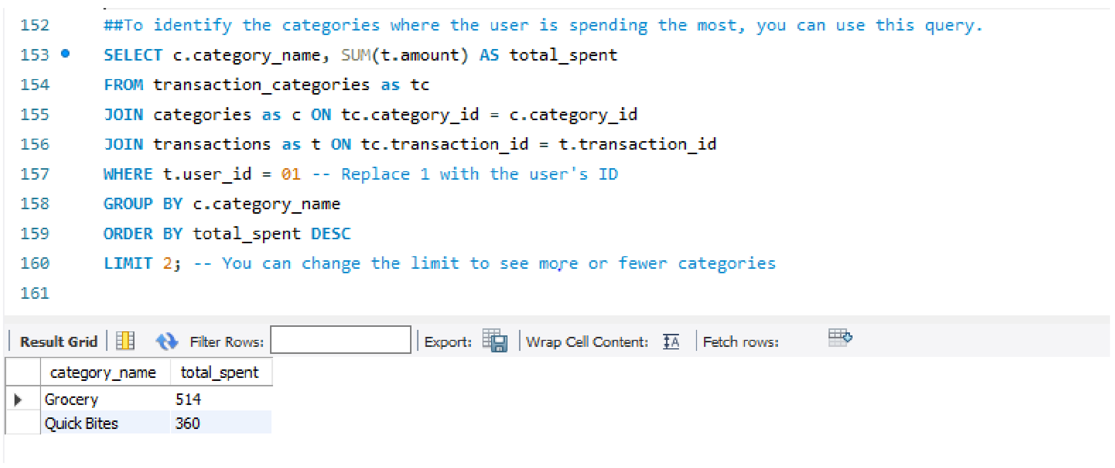


```
##Listing Categories with No Transactions:
SELECT category_id
FROM transaction_categories
WHERE category_id NOT IN (SELECT DISTINCT category_id FROM transactions);
```

sult Grid | Filter Rows: | Export: | Wrap Cell Content:

category_id

9)Find out the categories where the user is spending most .

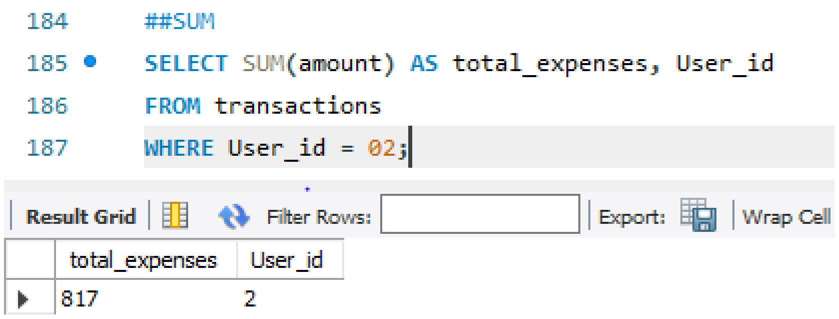


```
##To identify the categories where the user is spending the most, you can use this query.
SELECT c.category_name, SUM(t.amount) AS total_spent
FROM transaction_categories as tc
JOIN categories as c ON tc.category_id = c.category_id
JOIN transactions as t ON tc.transaction_id = t.transaction_id
WHERE t.user_id = 01 -- Replace 1 with the user's ID
GROUP BY c.category_name
ORDER BY total_spent DESC
LIMIT 2; -- You can change the limit to see more or fewer categories
```

category_name	total_spent
Grocery	514
Quick Bites	360

10)AGGREGATE FUNCTIONS :

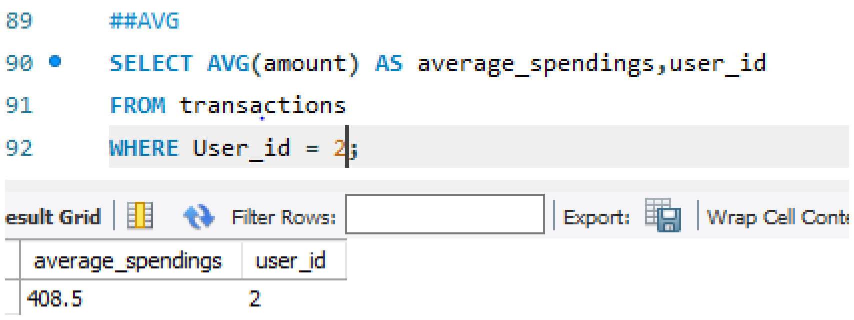
1)SUM :



```
##SUM
SELECT SUM(amount) AS total_expenses, User_id
FROM transactions
WHERE User_id = 02;
```

total_expenses	User_id
817	2

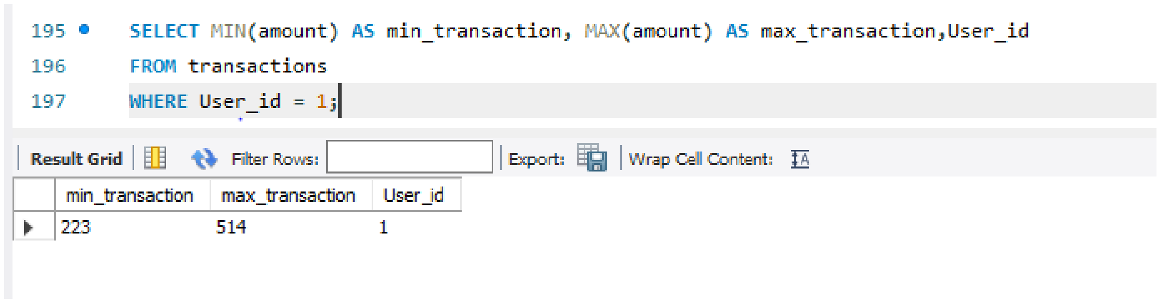
2)AVG:



```
##AVG
SELECT AVG(amount) AS average_spendings,user_id
FROM transactions
WHERE User_id = 2;
```

average_spendings	user_id
408.5	2

3)MIN AND MAX:

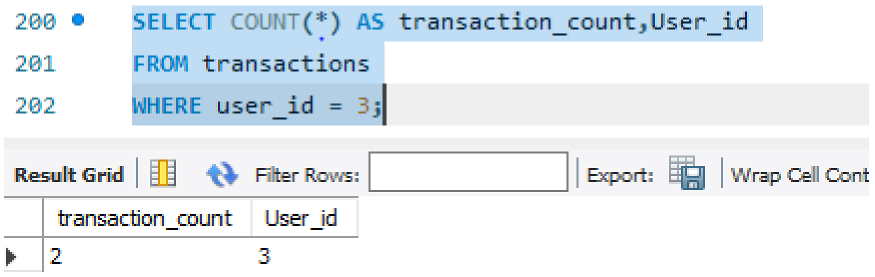


```
SELECT MIN(amount) AS min_transaction, MAX(amount) AS max_transaction,User_id
FROM transactions
WHERE User_id = 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

min_transaction	max_transaction	User_id
223	514	1

4)COUNT:



```
SELECT COUNT(*) AS transaction_count,User_id
FROM transactions
WHERE user_id = 3;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Cont

transaction_count	User_id
2	3

THANK YOU