3CS2005 DataBases Assignment #3

Tauha Imran | 22i-1239

i221239@nu.edu.pk

Saffi Muhammad Hashir | 22i- 22i-1293

i221293@nu.edu.pk

Introduction

In the realm of database management, assignments serve as practical laboratories for students to apply theoretical concepts in real-world scenarios. This case study delves into the intricacies of a database assignment centred around the Pakistan Super League (PSL) dataset. The assignment challenges students to design and implement a comprehensive database schema tailored to the requirements of the PSL, encompassing player statistics, match details, team dynamics, and tournament outcomes. Through the utilisation of SQL queries incorporating concepts such as correlated nested queries, UNION, GROUP BY, LIKE comparisons, and the HAVING clause, students are tasked with extracting meaningful insights from the dataset, thereby honing their skills in database querying and analysis..

Part#1 – Design

1. MySQL Design SQL Schema for the given dataset. Identify the Entities/Tables and corresponding columns, constraints, and primary keys. Also, identify the relationships between different Entities and map them through foreign keys correctly.

Here are the entities…

- **Audit_trail:**
  - **audit_id:** Integer, primary key, auto-increments.
  - **cashier_id:** Integer, foreign key referencing cashier_id in the Cashier table.
  - **user_id:** Integer, foreign key referencing customer_id in the Customer table.
  - **timespan:** Date.
  - **action:** String (VARCHAR(255)).
- **Customer:**
  - **customer_id:** Integer, primary key, auto-increments.
  - **name:** String (VARCHAR(255)), not null.
  - **address:** String (VARCHAR(255)), not null.
  - **phone:** Big integer (BIGINT).
  - **email:** String (VARCHAR(255)).
- **Account:**
  - **account_id:** Integer, primary key, auto-increments.

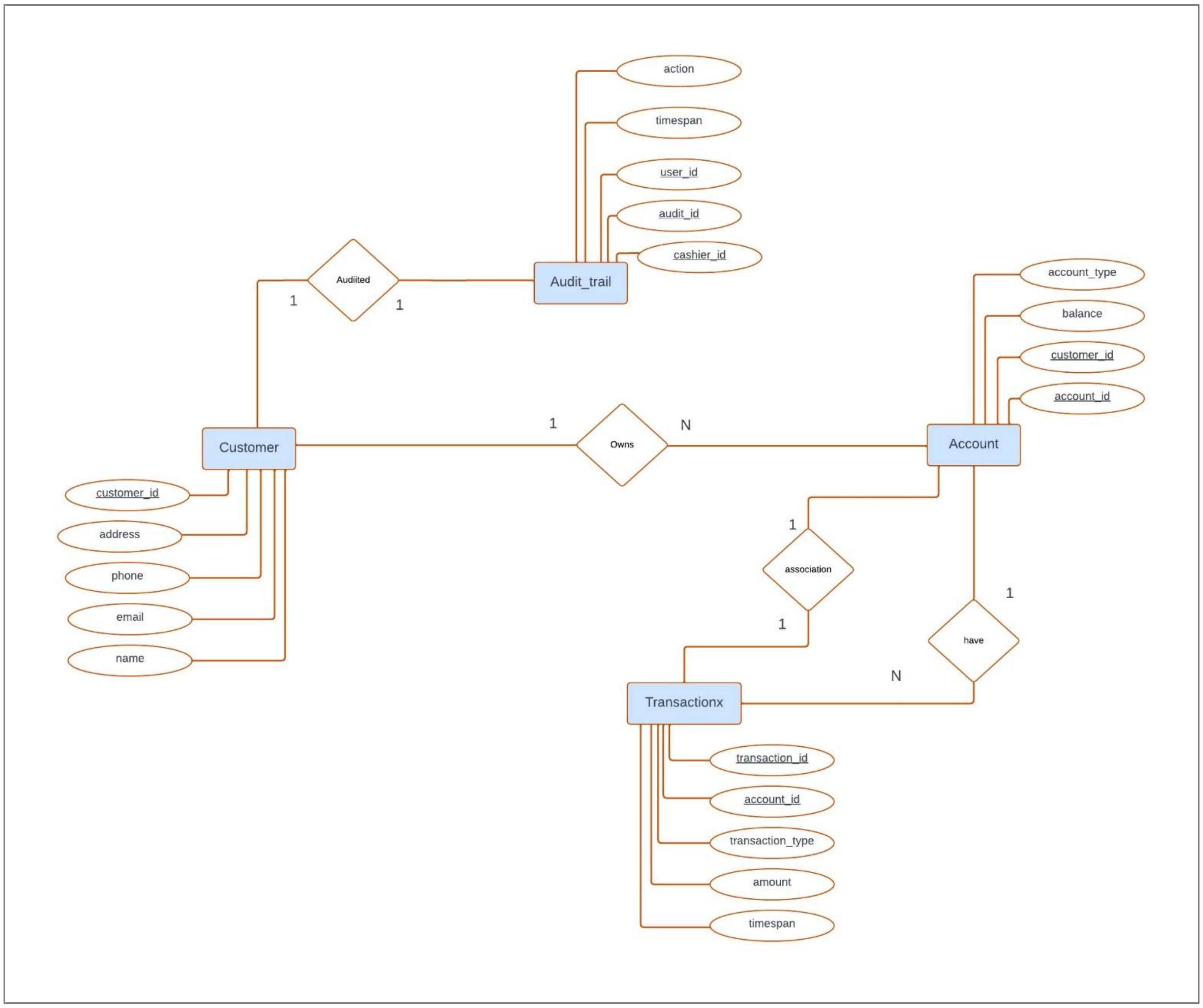
    - **customer_id:** Integer, foreign key referencing customer_id in the Customer table.
    - **balance:** Integer, not null.
    - **account_type:** String (VARCHAR(255)), not null.

- **Transaction:**
    - **transaction_id:** Integer, primary key, auto-increments.
    - **account_id:** Integer, foreign key referencing account_id in the Account table.
    - **transaction_type:** String (VARCHAR(255)), not null.
    - **amount:** Integer, not null.
    - **timespan:** Date

Here are the relationships…

1. **One customer can have multiple accounts (One-to-Many):**
   - Customer (1) ---- (1 to Many) ---> Account
2. **Each account can have multiple transactions (One-to-Many):**
   - Account (1) --- (1 to Many) ---> Transaction
3. **Each transaction is associated with one account (Many-to-One):**
   - Transaction (Many) --- (1 to 1) ---> Account
4. **Each operation recorded in the audit trail is associated with one account (Many-to-One):**
   - Audit_Trail (Many) --- (1 to 1) ---> Account

Here is the ERD (Entity Relationship diagram) …


action
timespan
user_id
audit_id
cashier_id
Audiited
1
1
Audit_trail
account_type
balance
customer_id
account_id
Customer
1
Owns
N
Account
customer_id
address
phone
email
name
1
association
1
1
have
N
Transactionx
transaction_id
account_id
transaction_type
amount
timespan

2. MySQL Using SQL queries perform following operations

a. Create Database and Tables. Define Constraints, Primary Keys and Foreign Keys

b. Identify the relationships between different Entities and map them through

foreign keys correctly.
The provided MySQL code defines foreign key relationships between several tables to ensure data integrity and consistency. Here's a breakdown of the foreign key mappings:

- **Audit_trail table:**
    1. **cashier_id:**
       This foreign key references the cashier_id primary key in the Cashier table. This ensures that any cashier performing an action recorded in the Audit_trail table actually exists in the Cashier table.
    2. **user_id:**
       This foreign key references the customer_id (assuming it's the primary key) in the Customer table. This links actions in the Audit_trail to specific customers involved.
- **Account table:**
    1. **customer_id:**
       This foreign key references the customer_id (assuming it's the primary key) in the Customer table.

This establishes a connection between customer information and their associated accounts.

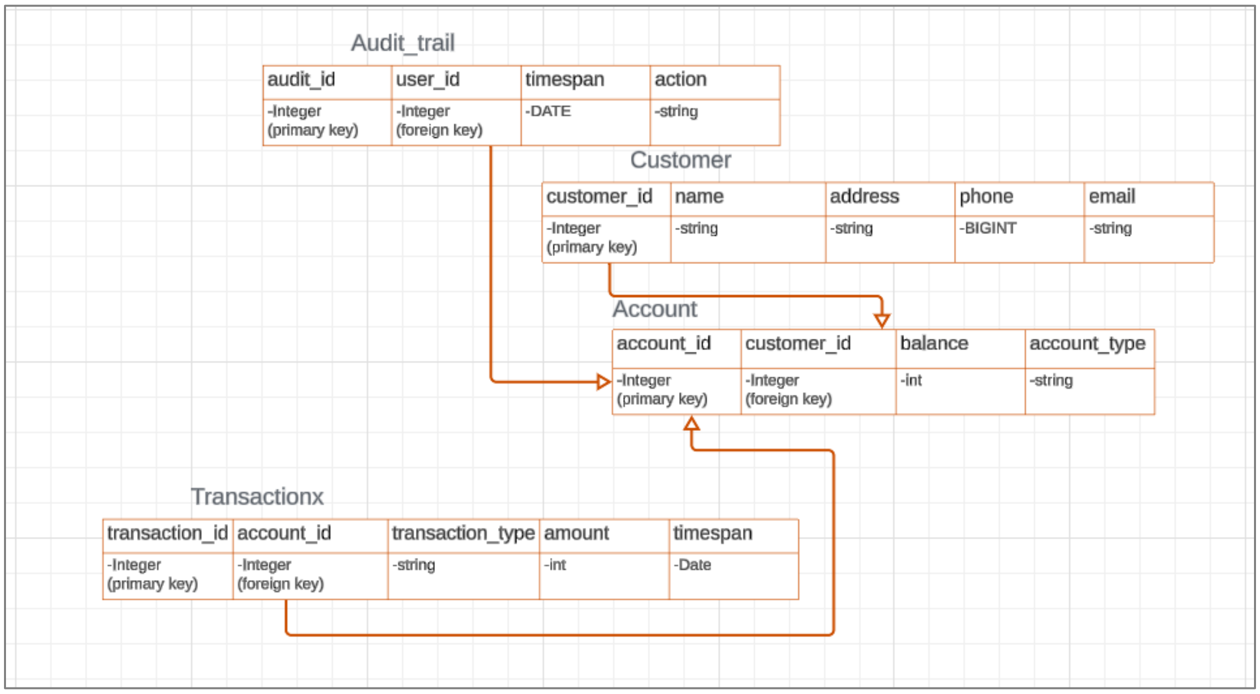
- **Transaction table:**

  1. **account_id:**

     This foreign key references the account_id (assuming it's the primary key) in the Account table. This ensures that all transactions recorded are linked to a valid account in the system.

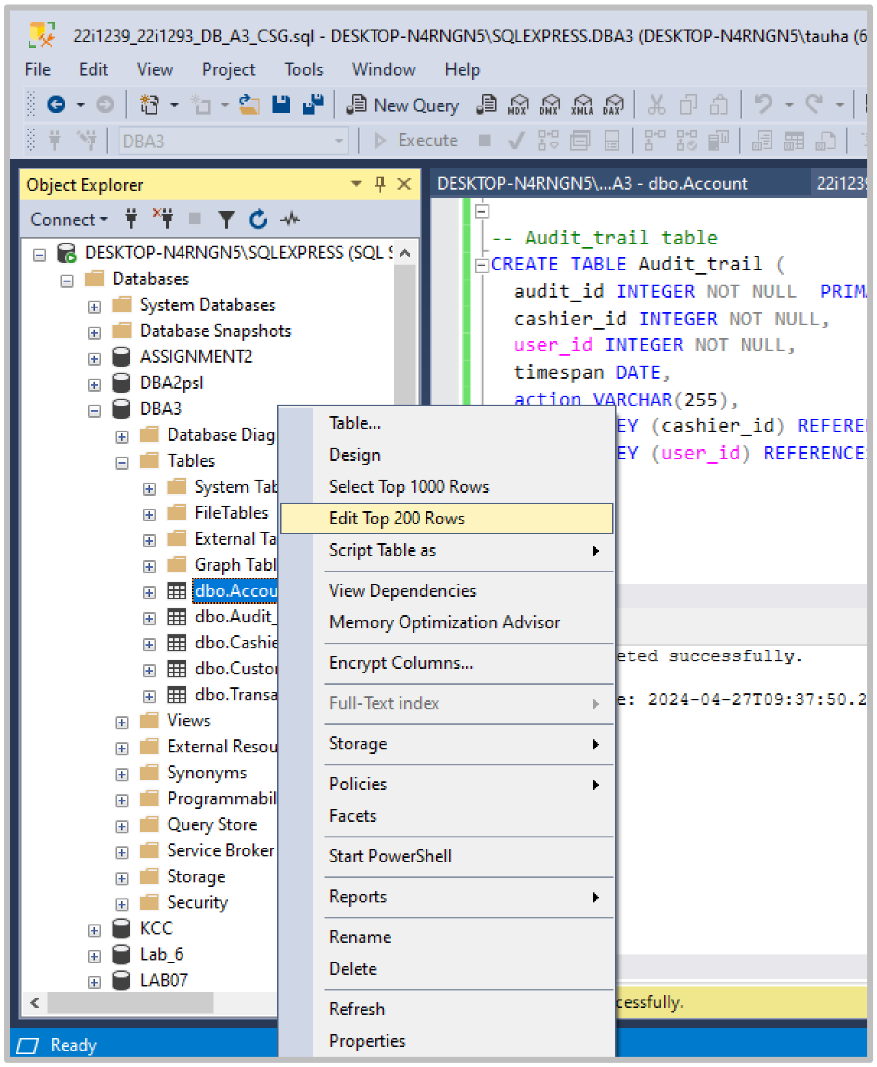
These foreign key constraints prevent invalid data entries. For example, you cannot record an action in the Audit_trail for a non-existent cashier or link a transaction to a non-existent account

Here is the RDM (Relational Data Model) ...



Part#2 – Insertions

Download the provided dataset and load it into the created tables. You can use any utility for this purpose. Using individual insert queries for each row is not allowed (and possible) for the given dataset. Learn Data Loading from Excel sheet into SQL server by yourself. It is very easy to learn and use. You have to understand the type of data in a column while designing the schema. You can assume empty cells as NULL.



```
-- Audit_trail table
CREATE TABLE Audit_trail (
    audit_id INTEGER NOT NULL  PRIM
    cashier_id INTEGER NOT NULL,
    user_id INTEGER NOT NULL,
    timespan DATE,
    action VARCHAR(255),
                                EY (cashier_id) REFERE
                                EY (user_id) REFERENCE
```

```
eted successfully.

e: 2024-04-27T09:37:50.2
```

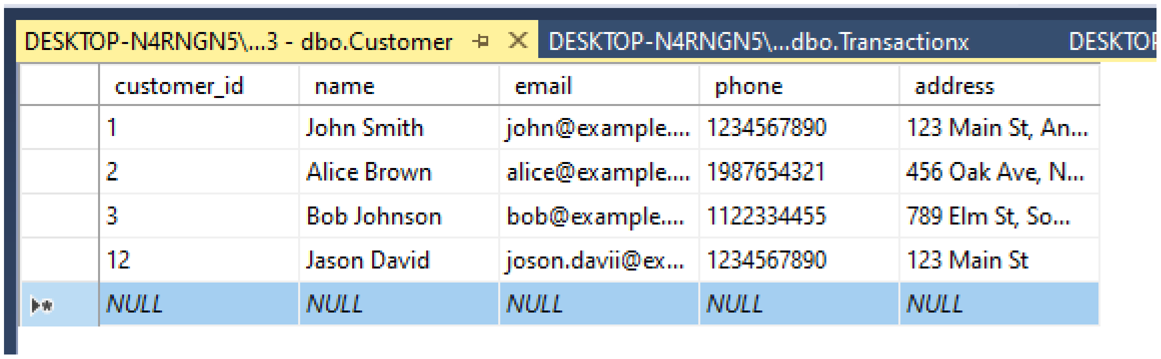
cessfully.

Creating tables

```sql
-- DB - Assignment#3
-- Tauha Imran 22i1239 - cs-g
-- Saffi Muhammad Hashir 22i1293 - cs-g


create database DBA3
use DBA3

-- Customer table
drop table Customer
CREATE TABLE Customer (
  customer_id INTEGER NOT NULL  PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  email VARCHAR(255),
  phone BIGINT,
  address VARCHAR(255) NOT NULL
);

-- Account table
--drop table Account
CREATE TABLE Account (
  account_id INTEGER NOT NULL  PRIMARY KEY,
  customer_id INTEGER NOT NULL,
  account_type VARCHAR(255) ,
  balance INTEGER ,
  FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
);

-- Transaction table
--drop table Transactionx
CREATE TABLE Transactionx (
  transaction_id INTEGER NOT NULL  PRIMARY KEY IDENTITY(1,1),
  account_id INTEGER NOT NULL,
  transaction_type VARCHAR(255) NOT NULL,
  amount INTEGER NOT NULL,
  timespan DATE,
  FOREIGN KEY (account_id) REFERENCES Account(account_id)
);

-- Audit_trail table
drop table Audit_trail
CREATE TABLE Audit_trail (
  audit_id INT not null PRIMARY KEY IDENTITY(1,1),
  user_id INTEGER NOT NULL,
  action VARCHAR(255),
  timespan DATE,
  details VARCHAR(255),
);
```

Imported data.

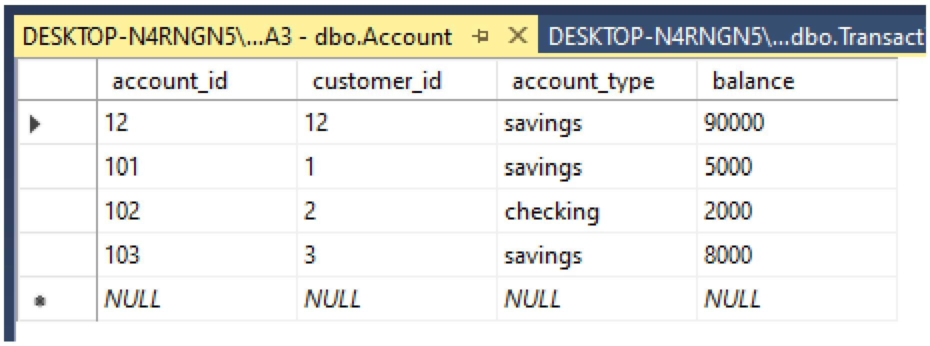
Customer table



DESKTOP-N4RNGN5\...3 - dbo.Customer DESKTOP-N4RNGN5\...dbo.Transactionx DESKTOP

	customer_id	name	email	phone	address
	1	John Smith	john@example....	1234567890	123 Main St, An...
	2	Alice Brown	alice@example....	1987654321	456 Oak Ave, N...
	3	Bob Johnson	bob@example....	1122334455	789 Elm St, So...
	12	Jason David	joson.davii@ex...	1234567890	123 Main St
▶*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

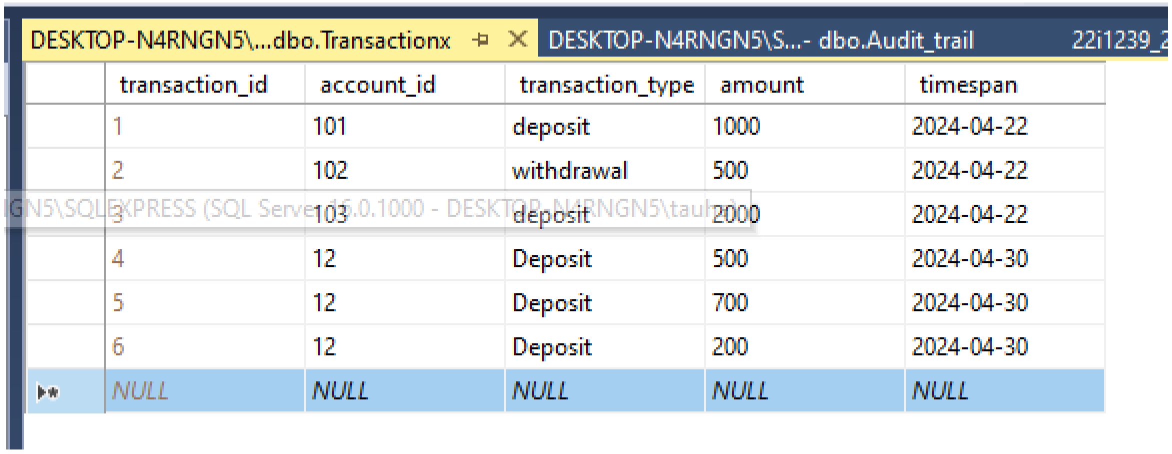
Account table



DESKTOP-N4RNGN5\...A3 - dbo.Account DESKTOP-N4RNGN5\...dbo.Transact

	account_id	customer_id	account_type	balance
▶	12	12	savings	90000
	101	1	savings	5000
	102	2	checking	2000
	103	3	savings	8000
*	*NULL*	*NULL*	*NULL*	*NULL*

Transctionx table

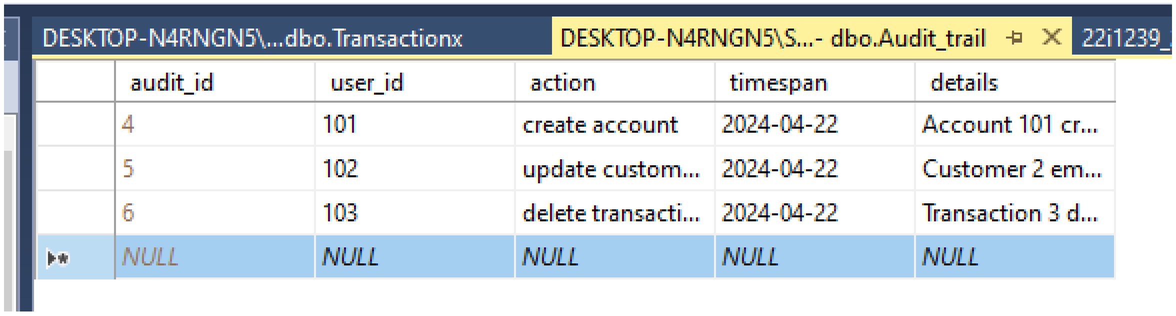


DESKTOP-N4RNGN5\...dbo.Transactionx | DESKTOP-N4RNGN5\S...- dbo.Audit_trail | 22i1239_

	transaction_id	account_id	transaction_type	amount	timespan
	1	101	deposit	1000	2024-04-22
	2	102	withdrawal	500	2024-04-22
	3	103	deposit	2000	2024-04-22
	4	12	Deposit	500	2024-04-30
	5	12	Deposit	700	2024-04-30
	6	12	Deposit	200	2024-04-30
▶*	NULL	NULL	NULL	NULL	NULL

GN5\SQLEXPRESS (SQL Server 16.0.1000 - DESKTOP-N4RNGN5\tauhid)

Audit_trail table



DESKTOP-N4RNGN5\...dbo.Transactionx | DESKTOP-N4RNGN5\S...- dbo.Audit_trail | 22i1239_

	audit_id	user_id	action	timespan	details
	4	101	create account	2024-04-22	Account 101 cr...
	5	102	update custom...	2024-04-22	Customer 2 em...
	6	103	delete transacti...	2024-04-22	Transaction 3 d...
▶*	NULL	NULL	NULL	NULL	NULL

Part#3 – Queries

Creating Triggers

Q1

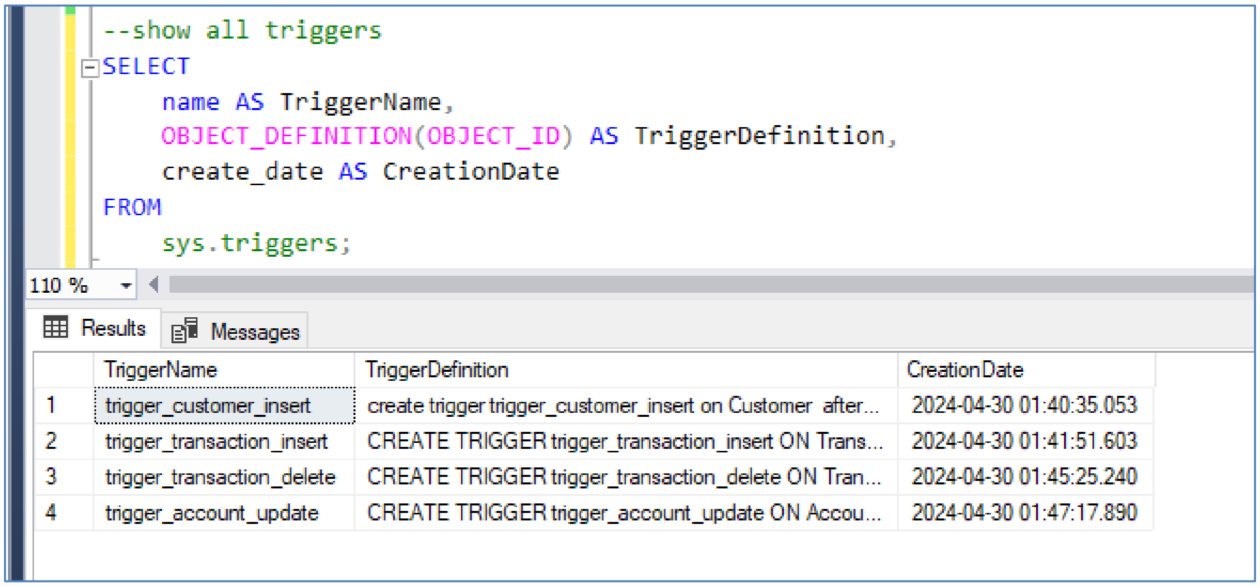
```
--part3

-->> Creating Triggers
select * from Audit_trail
/*1. Create Insert trigger when a new customer is added, save an entry in Audit_Trail table
during Insert Operation. A new customer is added when a new record is inserted into
Customer Table.*/
--drop trigger trigger_customer_insert
create trigger trigger_customer_insert on Customer
after insert
as
begin
       insert into Audit_trail ( user_id , action , timespan,details)
       values ((select customer_id from inserted) , 'Customer-inserted' , GETDATE() , 'a
new customer was inserted' );
end;
```

Q2

```
/*2. Create Insert and Delete triggers on Transaction Table to save an entry in Audit_Trail
table during Insert and Delete Operation on Transactions Table.*/
--drop trigger trigger_transaction_insert
CREATE TRIGGER trigger_transaction_insert ON Transactionx
AFTER INSERT
as
BEGIN
  INSERT INTO Audit_trail ( user_id,action, timespan, details)
  VALUES ((SELECT account_id FROM inserted),'transaction-added', GETDATE(), 'a new
transaction was added');
END;

--drop trigger trigger_transaction_delete
CREATE TRIGGER trigger_transaction_delete ON Transactionx
AFTER delete
as
BEGIN
  INSERT INTO Audit_trail ( user_id,action, timespan, details)
  VALUES ((SELECT account_id FROM inserted),'transaction-deleted', GETDATE(), 'an old
transaction was removed');
END;
```

Q3

```
/*3. Create update Trigger on Account Table to save an entry in Audit_Trail table during an
Update Operation*/
drop trigger trigger_account_update
CREATE TRIGGER trigger_account_update ON Account
AFTER update
as
BEGIN
  INSERT INTO Audit_trail ( user_id,action, timespan, details)
  VALUES ((SELECT account_id FROM inserted),'account_updated', GETDATE(), 'details of an
account were changed');
END;

--show all triggers
SELECT
    name AS TriggerName,
    OBJECT_DEFINITION(OBJECT_ID) AS TriggerDefinition,
    create_date AS CreationDate
FROM
    sys.triggers;
```



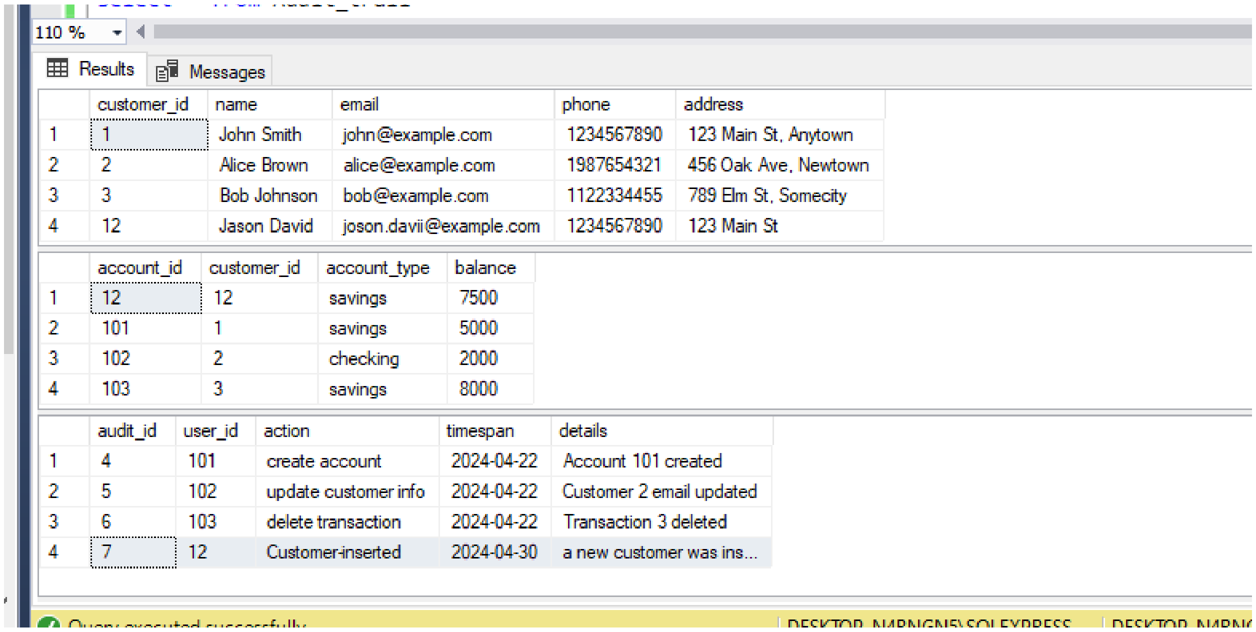
```
--show all triggers
SELECT
    name AS TriggerName,
    OBJECT_DEFINITION(OBJECT_ID) AS TriggerDefinition,
    create_date AS CreationDate
FROM
    sys.triggers;
```

	TriggerName	TriggerDefinition	CreationDate
1	trigger_customer_insert	create trigger trigger_customer_insert on Customer after...	2024-04-30 01:40:35.053
2	trigger_transaction_insert	CREATE TRIGGER trigger_transaction_insert ON Trans...	2024-04-30 01:41:51.603
3	trigger_transaction_delete	CREATE TRIGGER trigger_transaction_delete ON Tran...	2024-04-30 01:45:25.240
4	trigger_account_update	CREATE TRIGGER trigger_account_update ON Accou...	2024-04-30 01:47:17.890

Operations

Q4

```
/*4. Add a new customer and create his account (Creating account means that you need an
entry for the customer in Audit Table). Check the audit table, include snapshot of audit
table in report.*/
-- Add Customer
insert into Customer (customer_id,name, address, phone, email)
values (12,'Jason David', '123 Main St', 1234567890,'joson.davii@example.com' );

insert into Account( account_id , customer_id , balance , account_type)
values ( 012 , 12 , 7500 , 'savings');

select * from Customer
select * from Account
select * from Audit_trail
```



110 %

Results | Messages

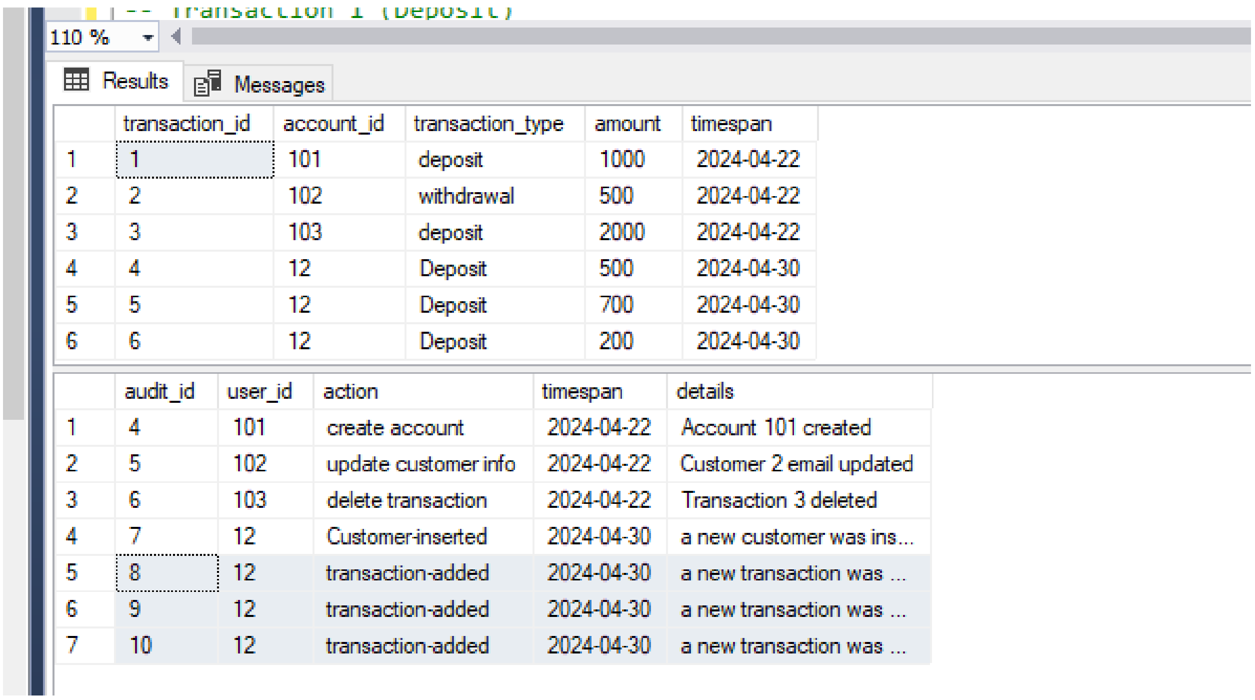
	customer_id	name	email	phone	address
1	1	John Smith	john@example.com	1234567890	123 Main St, Anytown
2	2	Alice Brown	alice@example.com	1987654321	456 Oak Ave, Newtown
3	3	Bob Johnson	bob@example.com	1122334455	789 Elm St, Somecity
4	12	Jason David	joson.davii@example.com	1234567890	123 Main St

	account_id	customer_id	account_type	balance
1	12	12	savings	7500
2	101	1	savings	5000
3	102	2	checking	2000
4	103	3	savings	8000

	audit_id	user_id	action	timespan	details
1	4	101	create account	2024-04-22	Account 101 created
2	5	102	update customer info	2024-04-22	Customer 2 email updated
3	6	103	delete transaction	2024-04-22	Transaction 3 deleted
4	7	12	Customer-inserted	2024-04-30	a new customer was ins...

Q5

```sql
/*5. Perform 4 Financial Transaction by the customer and Check the audit table, include
snapshot of audit table in report.*/
-- Assuming you have retrieved the customer's account ID (replace with actual value)
select * from Account
select * from Transactionx
-- Transaction 1 (Deposit)
INSERT INTO Transactionx ( account_id ,transaction_type, amount,timespan)
VALUES (12, 'Deposit', 500 , GETDATE());

-- Transaction 2 (withrawal)
INSERT INTO Transactionx (account_id, transaction_type, amount,timespan)
VALUES (12, 'Deposit', 700, GETDATE());

-- Transaction 1 (Deposit)
INSERT INTO Transactionx ( account_id ,transaction_type, amount,timespan)
VALUES (12, 'Deposit', 200 , GETDATE());

-- Check Audit Trail
SELECT * from Transactionx
SELECT * FROM Audit_trail;
```



110 %

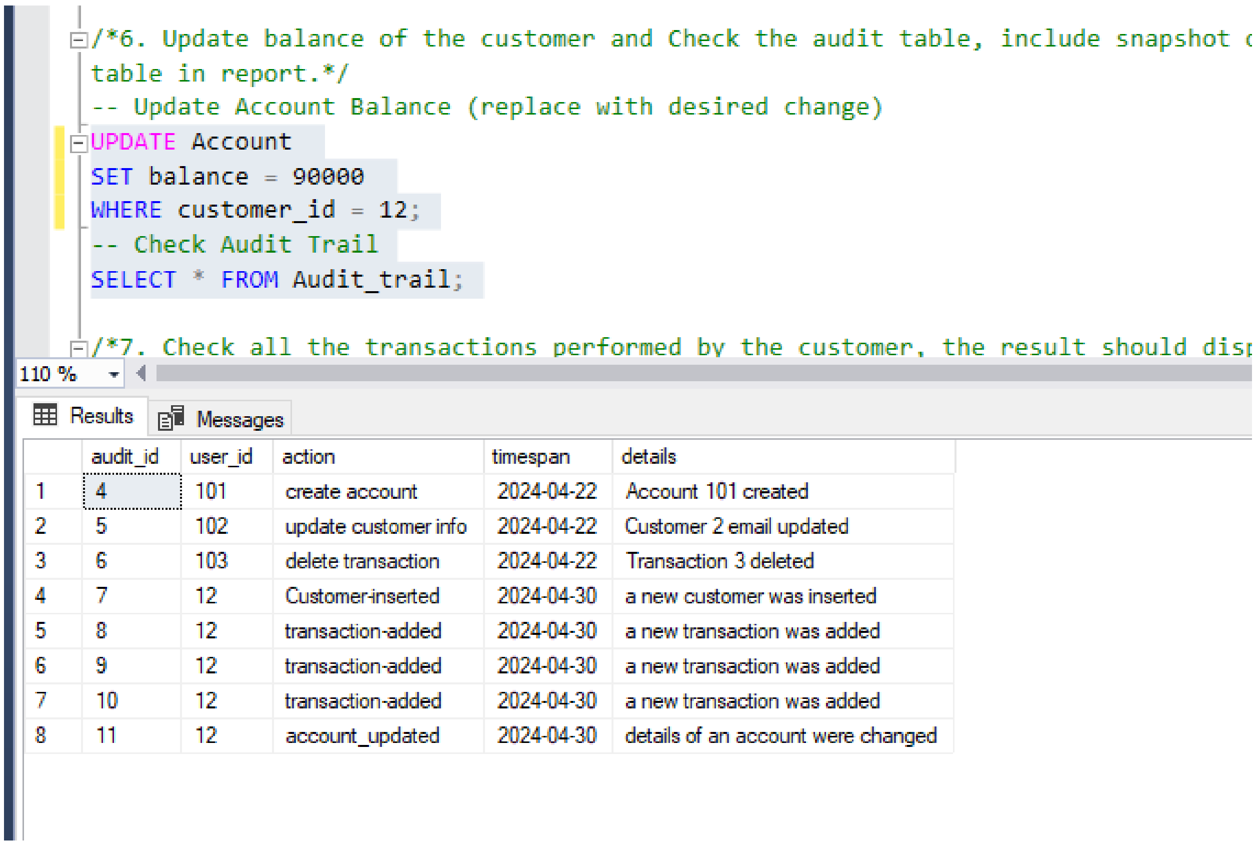
Results | Messages

	transaction_id	account_id	transaction_type	amount	timespan
1	1	101	deposit	1000	2024-04-22
2	2	102	withdrawal	500	2024-04-22
3	3	103	deposit	2000	2024-04-22
4	4	12	Deposit	500	2024-04-30
5	5	12	Deposit	700	2024-04-30
6	6	12	Deposit	200	2024-04-30

	audit_id	user_id	action	timespan	details
1	4	101	create account	2024-04-22	Account 101 created
2	5	102	update customer info	2024-04-22	Customer 2 email updated
3	6	103	delete transaction	2024-04-22	Transaction 3 deleted
4	7	12	Customer-inserted	2024-04-30	a new customer was ins...
5	8	12	transaction-added	2024-04-30	a new transaction was ...
6	9	12	transaction-added	2024-04-30	a new transaction was ...
7	10	12	transaction-added	2024-04-30	a new transaction was ...

Q6

```
/*6. Update balance of the customer and Check the audit table, include snapshot of audit
table in report.*/
-- Update Account Balance (replace with desired change)
UPDATE Account
SET balance = 90000
WHERE customer_id = 12;
-- Check Audit Trail
SELECT * FROM Audit_trail;
```



```
/*6. Update balance of the customer and Check the audit table, include snapshot c
table in report.*/
-- Update Account Balance (replace with desired change)
UPDATE Account
SET balance = 90000
WHERE customer_id = 12;
-- Check Audit Trail
SELECT * FROM Audit_trail;

/*7. Check all the transactions performed by the customer, the result should disp
```

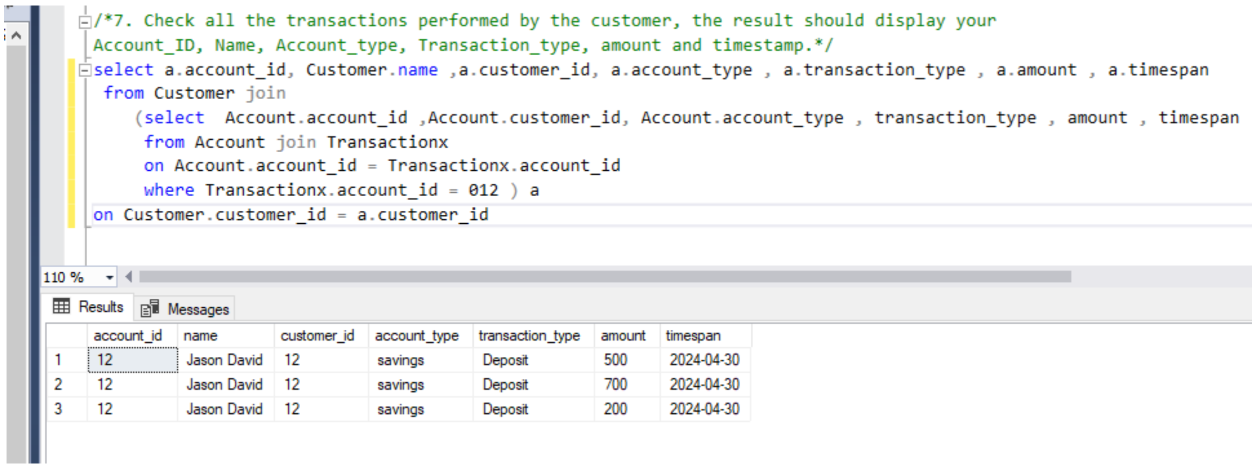
110 %

Results | Messages

	audit_id	user_id	action	timespan	details
1	4	101	create account	2024-04-22	Account 101 created
2	5	102	update customer info	2024-04-22	Customer 2 email updated
3	6	103	delete transaction	2024-04-22	Transaction 3 deleted
4	7	12	Customer-inserted	2024-04-30	a new customer was inserted
5	8	12	transaction-added	2024-04-30	a new transaction was added
6	9	12	transaction-added	2024-04-30	a new transaction was added
7	10	12	transaction-added	2024-04-30	a new transaction was added
8	11	12	account_updated	2024-04-30	details of an account were changed

Q7

```
/*7. Check all the transactions performed by the customer, the result should display your
Account_ID, Name, Account_type, Transaction_type, amount and timestamp.*/
select a.account_id, Customer.name ,a.customer_id, a.account_type , a.transaction_type ,
a.amount , a.timespan
 from Customer join
       (select  Account.account_id ,Account.customer_id, Account.account_type ,
transaction_type , amount , timespan
        from Account join Transactionx
        on Account.account_id = Transactionx.account_id
        where Transactionx.account_id = 012 ) a
on Customer.customer_id = a.customer_id
```



```
/*7. Check all the transactions performed by the customer, the result should display your
Account_ID, Name, Account_type, Transaction_type, amount and timestamp.*/
select a.account_id, Customer.name ,a.customer_id, a.account_type , a.transaction_type , a.amount , a.timespan
 from Customer join
    (select  Account.account_id ,Account.customer_id, Account.account_type , transaction_type , amount , timespan
     from Account join Transactionx
     on Account.account_id = Transactionx.account_id
     where Transactionx.account_id = 012 ) a
on Customer.customer_id = a.customer_id
```

	account_id	name	customer_id	account_type	transaction_type	amount	timespan
1	12	Jason David	12	savings	Deposit	500	2024-04-30
2	12	Jason David	12	savings	Deposit	700	2024-04-30
3	12	Jason David	12	savings	Deposit	200	2024-04-30

CONCLUSION

In conclusion, database assignments such as the one centred around the PSL dataset provide invaluable opportunities for students to sharpen their database management skills in a practical setting. By grappling with real-world data and applying SQL queries to extract actionable insights, students gain hands-on experience in designing, querying, and optimising database structures—an essential skill set for aspiring database administrators and analysts. As the assignment unfolds, students delve into the complexities of data manipulation, learning to navigate relational databases and derive meaningful conclusions from complex datasets. Through these endeavours, students not only deepen their understanding of database concepts but also develop the problem-solving abilities necessary for success in the dynamic field of data management.