

SQL ACID transaction & stored procedure

Server query optimization and transaction control

About project

In an ACID transaction, if any of these actions fail (for example, if your payment doesn't go through), then the entire transaction fails and none of the changes are made. This ensures that the data in the database remains consistent.

Benefits: ACID transaction is a way to ensure that a group of actions in a database either all happen successfully or none of them happen at all.

Project case study: Let's say you're buying a pair of boots online. When you make the purchase, several things need to happen: the boots need to be added to your cart, your payment needs to be processed, your account needs to be debited the correct amount, the store's account needs to be credited, and the inventory of boots needs to be reduced by one.

- Created a stored procedure routine named **TRANSACTION_ROSE** which will include TCL commands like COMMIT and ROLLBACK, developed the routine based on the given scenario to execute a transaction. **Scenario:** Let's buy Rose a pair of Boots from ShoeShop. So we have to update the Rose balance as well as the ShoeShop balance in the BankAccounts table. Then we also have to update Boots stock in the ShoeShop table. After Boots, let's also attempt to buy Rose a pair of Trainers.
- Created a stored procedure **TRANSACTION_JAMES** to execute a transaction based on the following **scenario:** First buy James 4 pairs of Trainers from ShoeShop. Update his balance as well as the balance of ShoeShop. Also, update the stock of Trainers at ShoeShop. Then attempt to buy James a pair of Brogues from ShoeShop. If any of the UPDATE statements fail, the whole transaction fails the transaction rolls back and commit the transaction only if the whole transaction is successful.

Skills utilised



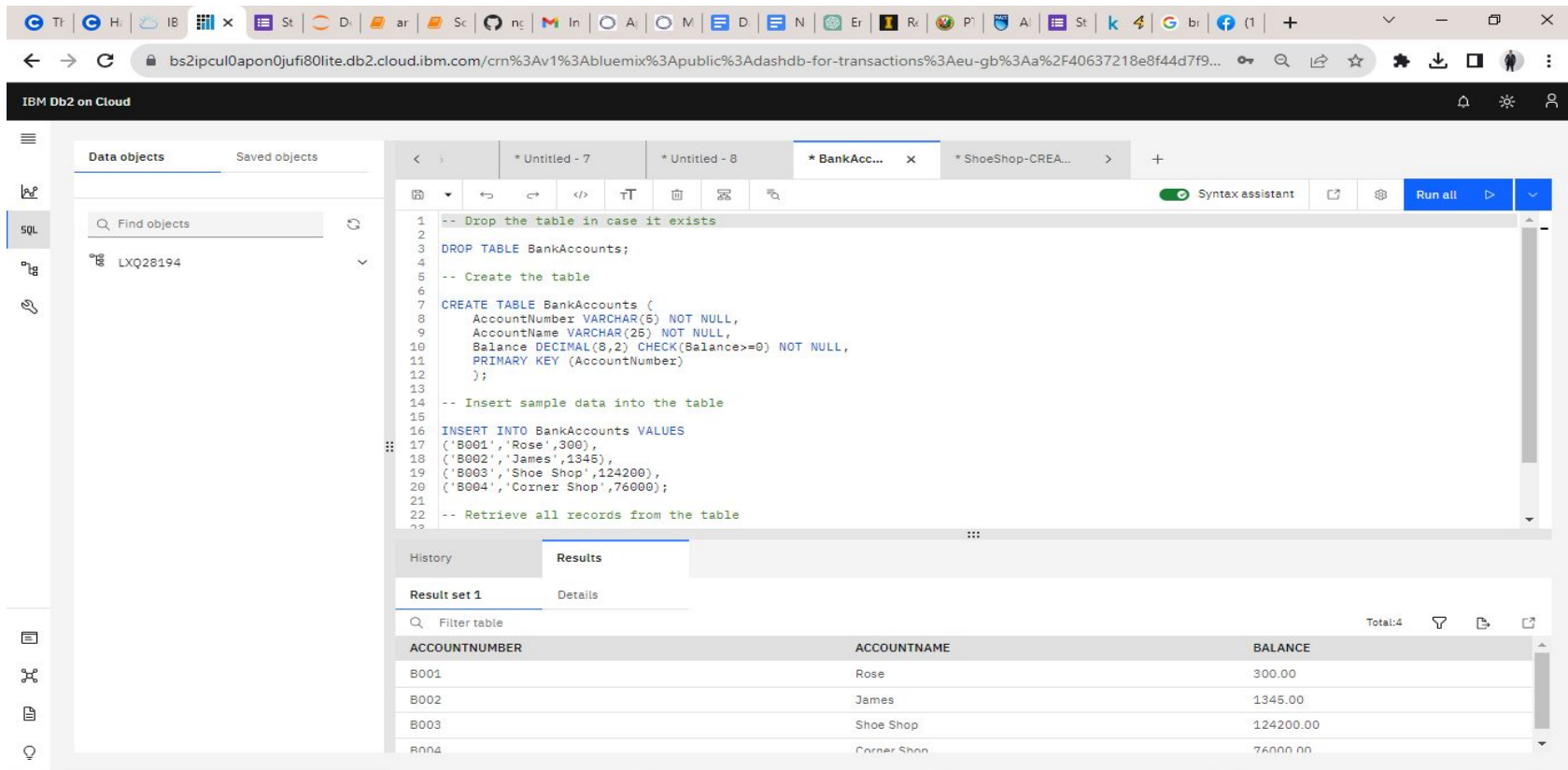
Software used: IBM Db2 cloud database

Project screenshots and explanations

Next eleven slides

Create database one

This was what the database looks like in IBM Db2 database instance. The table shown contains list of customers account details(including Rose's) with their account balance and account balance.



The screenshot displays the IBM Db2 on Cloud web interface. The browser address bar shows the URL: `bs2ipcul0apon0jufi80lite.db2.cloud.ibm.com/crn%3Av1%3Abluemix%3Apublic%3Adashdb-for-transactions%3Aeu-gb%3Aa%2F40637218e8f44d7f9...`. The interface includes a left sidebar with navigation icons and a main workspace. The workspace has a top bar with tabs for 'Data objects' and 'Saved objects'. Below this is a search bar labeled 'Find objects' with the text 'LXQ28194'. The main area is divided into two sections: a top section for SQL execution and a bottom section for results. The SQL section shows a script with the following content:

```
-- Drop the table in case it exists
DROP TABLE BankAccounts;

-- Create the table
CREATE TABLE BankAccounts (
  AccountNumber VARCHAR(5) NOT NULL,
  AccountName VARCHAR(25) NOT NULL,
  Balance DECIMAL(8,2) CHECK(Balance>=0) NOT NULL,
  PRIMARY KEY (AccountNumber)
);

-- Insert sample data into the table
INSERT INTO BankAccounts VALUES
('B001','Rose',300),
('B002','James',1345),
('B003','Shoe Shop',124200),
('B004','Corner Shop',76000);

-- Retrieve all records from the table
```

The bottom section, titled 'Results', shows 'Result set 1' with a table of 4 rows and 3 columns. The table has the following data:

ACCOUNTNUMBER	ACCOUNTNAME	BALANCE
B001	Rose	300.00
B002	James	1345.00
B003	Shoe Shop	124200.00
B004	Corner Shop	76000.00

Create database two

This was what the database looks like in IBM Db2 database instance. The table shown at the bottom of the image contains inventory details of Shoeshop business owner with product name, stock(quantity) and price for each.

The screenshot displays the IBM Db2 on Cloud web interface. The browser address bar shows the URL: `bs2ipcul0apon0jufi80lite.db2.cloud.ibm.com/crn%3Av1%3Abluemix%3Apublic%3Adashdb-for-transactions%3Aeu-gb%3Aa%2F40637218e8f44d7f9...`. The interface includes a sidebar with navigation icons and a main workspace. The workspace has tabs for 'Data objects' and 'Saved objects'. The 'Data objects' tab is active, showing a search bar and a list of objects, including 'LXQ28194'. The main workspace contains a SQL editor with the following code:

```
-- Drop the table in case it exists
DROP TABLE ShoeShop;

-- Create the table
CREATE TABLE ShoeShop (
  Product VARCHAR(25) NOT NULL,
  Stock INTEGER NOT NULL,
  Price DECIMAL(8,2) CHECK(Price>0) NOT NULL,
  PRIMARY KEY (Product)
);

-- Insert sample data into the table
INSERT INTO ShoeShop VALUES
('Boots',11,200),
('High heels',8,600),
('Brogues',10,150),
('Trainers',14,300);
```

Below the SQL editor, the 'Results' tab is selected, showing 'Result set 1' with a 'Details' view. The results are displayed in a table with the following data:

PRODUCT	STOCK	PRICE
Boots	11	200.00
High heels	8	600.00
Brogues	10	150.00
Trainers	14	300.00

The table also includes a 'Total:4' indicator and icons for filtering, sorting, and exporting data.

TRANSACTION_ROSE STORED PROCEDURE
TRANSACTION

Create Stored Procedure and save in the server

This is screenshot of the stored procedure(TRANSACTION_ROSE) and the SQL successfully created, ran and stored in the server

The screenshot displays the IBM Db2 on Cloud web interface. The browser address bar shows the URL: `bs2ipcul0apon0jufi80lite.db2.cloud.ibm.com/crn%3Av1%3Abluemix%3Apublic%3Adashdb-for-transactions%3Aeu-gb%3Aa%2F40637218e8f4...`. The interface includes a sidebar with navigation options like Data objects, Saved objects, and SQL. The main editor area shows a SQL script for creating a stored procedure named TRANSACTION_ROSE. The script includes comments explaining the purpose of each line, such as setting the terminator, declaring variables, and handling exceptions. The procedure performs updates on the BankAccounts and ShoeShop tables. The bottom panel shows the execution results, indicating a successful run with a runtime of 0.087 seconds.

```
--SET TERMINATOR @
CREATE PROCEDURE TRANSACTION_ROSE
LANGUAGE SQL
MODIFIES SQL DATA
BEGIN
    DECLARE SQLCODE INTEGER DEFAULT 0;
    DECLARE retcode INTEGER DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
    SET retcode = SQLCODE;

    UPDATE BankAccounts
    SET Balance = Balance-200
    WHERE AccountName = 'Rose';

    UPDATE BankAccounts
    SET Balance = Balance+200
    WHERE AccountName = 'Shoe Shop';

    UPDATE ShoeShop
    SET Stock = Stock-1
    WHERE Product = 'Boots';

    UPDATE BankAccounts
    SET Balance = Balance-300
    WHERE AccountName = 'Rose';

    IF retcode < 0 THEN
        ROLLBACK WORK;
    
```

Run time: 0.087 s

Run by: lxq28194

Database: ...

Full query body

```
CREATE PROCEDURE TRANSACTION_ROSE
LANGUAGE SQL
MODIFIES SQL DATA
BEGIN
    -- Name of this stored procedure routine
    -- Language used in this routine
    -- This routine will only write/modify data in the table

```


Confirm the stored procedure is saved on the server

This screenshot shows on the right hand side that the **TRANSACTION_ROSE** Procedure now exists in the database server and ready to run when called

The screenshot displays the IBM Db2 on Cloud web interface. The browser address bar shows the URL: `bs2ipcul0apon0jufi80lite.db2.cloud.ibm.com/crn%3Av1%3Abluemix%3Apublic%3Adashdb-for-transactions%3Aeu-gb%3Aa%2F40637218e8f4...`. The interface has a dark header with the text "IBM Db2 on Cloud" and a navigation bar with tabs: "Load Data", "Load History", "Tables", "Views", "Indexes", "Aliases", "MQTs", "Sequences", and "Application objects". The "Application objects" tab is selected, and within it, the "Stored procedures" sub-tab is active. A search bar labeled "Find schemas or procedures" is present. The main content area is split into two panels. The left panel, titled "Schemas", shows a table with one entry: "LXQ28194" with a "Definer type" of "User" and "3" procedures. The right panel, titled "Procedures", shows a table with three entries: "RETRIEVE_ALL", "TRANSACTION_ROSE", and "UPDATE_SALEPRICE", all under the "LXQ28194" schema. A "New procedure" button is visible in the top right of the Procedures panel.

IBM Db2 on Cloud

Load Data Load History Tables Views Indexes Aliases MQTs Sequences Application objects

Stored procedures User-defined Types User-defined Functions

Find schemas or procedures Refresh

Name	Definer type	Procedures
LXQ28194	User	3

Name	Schema	Properties
RETRIEVE_ALL	LXQ28194	...
TRANSACTION_ROSE	LXQ28194	...
UPDATE_SALEPRICE	LXQ28194	...

Call stored Procedure

I called the stored procedure **TRANSACTION_ROSE** and it executed successfully, I will need to confirm whether there is a commit action or rollback action.

The screenshot displays the IBM Db2 on Cloud console interface. On the left, a sidebar shows the 'Data objects' tab with a search bar and a list of objects, including 'LXQ28194'. The main area is divided into two sections: a top section for editing SQL scripts and a bottom section for viewing execution history.

The top section shows a SQL script in a text editor with the following content:

```
1 CALL TRANSACTION_ROSE; -- Caller query
2 SELECT * FROM BankAccounts;
3 SELECT * FROM ShoeShop;
```

The bottom section, titled 'History', displays a table of execution results. The table has four columns: 'Script', 'Date', 'Status', and 'Runtime'. The data is as follows:

Script	Date	Status	Runtime
Untitled - 5	Aug 20, 2023 8:19:42 PM	✓ 3	0.208 s
CALL TRANSACTION_ROSE		✓	0.191 s
-- Caller query SELECT * FROM BankAccounts		✓	0.010 s
SELECT * FROM ShoeShop		✓	0.007 s
Untitled - 5	Aug 20, 2023 8:07:58 PM	✓ 1	0.087 s
CREATE PROCEDURE TRANSACTION_ROSE -- Name of this stored procedure routine LANGUAGE SQL -- L		✓	0.087 s
ShoeShop-CREATE.sql	Aug 20, 2023 7:48:14 PM	✓ 3 ✗ 1	0.867 s
-- Drop the table in case it exists DROP TABLE ShoeShop		✗	0.020 s

Confirm ROSE Account balance

After the stored procedure call, nothing changed on the account balance of ROSE. **HERE IS WHY:** The first three UPDATES should run successfully. Both the balance of Rose and ShoeShop should have been updated in the BankAccounts table. The current balance of Rose should stand at $300 - 200$ (price of a pair of Boots) = 100. The current balance of ShoeShop should stand at $124200 + 200 = 124400$. The stock of Boots should also be updated in the ShoeShop table after the successful purchase for Rose, $11 - 1 = 10$. The last UPDATE statement tries to buy Rose a pair of Trainers, **but her balance becomes insufficient** (Current balance of Rose: $100 < \text{Price of Trainers: } 300$) after buying a pair of Boots. So, the last UPDATE statement fails. **Since the whole transaction fails if any of the SQL statements fail, the transaction won't be committed and the database data records remains unchanged leaving ROSE money untampered with.**

The screenshot shows the IBM Db2 on Cloud web interface. The browser address bar displays the URL: `bs2ipcul0apon0jufi80lite.db2.cloud.ibm.com/crn%3Av1%3Abluemix%3Apublic%3Adashdb-for-transactions%3Aeu-gb%3Aa%2F40637218e8f4...`. The interface has a dark header bar with the text "IBM Db2 on Cloud". Below the header, there's a sidebar with "Data objects" and "Saved objects" tabs. The "Data objects" tab is active, showing a search bar with "Find objects" and a list of objects including "LXQ28194". The main area displays a SQL editor with the following code:

```
1 CALL TRANSACTION_ROSE; -- Caller query
2 SELECT * FROM BankAccounts;
3 SELECT * FROM ShoeShop;
```

Below the editor, there's a "Run all" button. The results section is visible, showing "Result set 1" with a table of account balances. The table has three columns: ACCOUNTNUMBER, ACCOUNTNAME, and BALANCE. The data is as follows:

ACCOUNTNUMBER	ACCOUNTNAME	BALANCE
B001	Rose	300.00
B002	James	1345.00
B003	Shoe Shop	124200.00

Confirm ShoeShop product inventory details for Boots

After the **TRANSACTION_ROSE** call, nothing changed on the quantity of **Boots**. **HERE IS WHY**: since one of the **ACID** update transactions failed, all the transaction rolled back and nothing happened to the quantity of **Boots** available in the inventory database which remained at 11. **So the store owner has nothing to worry about!**

The screenshot shows the IBM Db2 on Cloud console interface. The top navigation bar includes the IBM logo and various icons. The main content area is divided into a left sidebar and a main workspace. The sidebar contains a 'Data objects' tab and a search bar. The main workspace displays a SQL query in a text editor, which has been executed. The results are shown in a table with columns 'PRODUCT', 'STOCK', and 'PRICE'. The table contains four rows of data: Boots, High heels, Brogues, and Trainers. The 'Boots' row shows a stock quantity of 11 and a price of 200.00. The 'High heels' row shows a stock quantity of 8 and a price of 600.00. The 'Brogues' row shows a stock quantity of 10 and a price of 150.00. The 'Trainers' row shows a stock quantity of 14 and a price of 300.00. The total number of rows is 4.

IBM Db2 on Cloud

SQL

Find objects

LXQ28194

```
1 CALL TRANSACTION_ROSE; -- Caller query
2 SELECT * FROM BankAccounts;
3 SELECT * FROM ShoeShop;
```

History Results

Result set 1 Details

Filter table

PRODUCT	STOCK	PRICE
Boots	11	200.00
High heels	8	600.00
Brogues	10	150.00
Trainers	14	300.00

Total: 4

TRANSACTION_JAMES STORED PROCEDURE
TRANSACTION

Create Stored Procedure TRANSACTION_JAMES and save in the server

This is screenshot of the stored procedure(TRANSACTION_JAMES) and the SQL successfully created, ran and stored in the server

The screenshot displays the IBM Db2 on Cloud web interface. The browser address bar shows the URL: `bs2ipcul0apon0jufi80lite.db2.cloud.ibm.com/crn%3Av1%3Abluemix%3Apublic%3Adashdb-for-transactions%3Aeu-gb%3Aa%2F40637218e8f44d7f9a386...`. The interface includes a sidebar with navigation icons for Data objects, SQL, and other tools. The main editor area shows a SQL script for creating a stored procedure named TRANSACTION_JAMES. The script includes comments explaining the purpose of each line, such as setting the terminator, language, and modifying data in the table. The procedure body contains SQL statements to update the BankAccounts table and the ShoeShop table. The Results tab at the bottom shows the execution details, including the run time (0.065 s), the user (lxq28194), the database (crn:v1:bluemix:public:dashdb-for-tr...), and the affected rows (0). The Full query body tab shows the complete SQL script that was executed.

```
--SET TERMINATOR @
CREATE PROCEDURE TRANSACTION_JAMES
LANGUAGE SQL
MODIFIES SQL DATA
BEGIN
    DECLARE SQLCODE INTEGER DEFAULT 0;
    DECLARE retcode INTEGER DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
    SET retcode = SQLCODE;

    UPDATE BankAccounts
    SET Balance = Balance-1200
    WHERE AccountName = 'James';

    UPDATE BankAccounts
    SET Balance = Balance+1200
    WHERE AccountName = 'Shoe Shop';

    UPDATE ShoeShop
```

Results

Run time	Full query body
0.065 s	CREATE PROCEDURE TRANSACTION_JAMES
Run by lxq28194	LANGUAGE SQL
Database crn:v1:bluemix:public:dashdb-for-tr...	MODIFIES SQL DATA
Affected rows 0	in the table
	BEGIN
	DECLARE SQLCODE INTEGER DEFAULT 0;
	ed 0
	DECLARE retcode INTEGER DEFAULT 0;
	assigned 0
	DECLARE CONTINUE HANDLER FOR SQLEXCEPTION

Call stored Procedure

I called the stored procedure **TRANSACTION_JAMES** and it executed successfully, I will need to confirm whether there is a commit action or rollback action.

The screenshot shows the IBM Db2 on Cloud web interface. The browser address bar displays the URL: `bs2ipcul0apon0jufi80lite.db2.cloud.ibm.com/crn%3Av1%3Abluemix%3Apublic%3Adashdb-for-transactions%3Aeu-gb%3Aa%2F40637218e8f44d7f9a386...`. The interface includes a sidebar with navigation icons and a main workspace. The workspace has a top bar with tabs for different SQL scripts, including `* BankAccounts-...`, `* ShoeShop-CREA...`, `*Untitled - 5`, `*Untitled - 7`, and `*Untitled ...`. The `*Untitled ...` tab is active, showing a SQL script with three lines: `1 CALL TRANSACTION_JAMES; -- Caller query`, `2 SELECT * FROM BankAccounts;`, and `3 SELECT * FROM ShoeShop;`. Below the script editor, there is a `History` tab selected, displaying a table of execution history. The table has columns for `Script`, `Date`, `Status`, and `Runtime`. The history shows multiple successful executions of the script, with the most recent one on August 20, 2023, at 9:15:25 PM, with a status of 3 and a runtime of 0.142 s.

Script	Date	Status	Runtime
Untitled - 2	Aug 20, 2023 9:15:25 PM	3	0.142 s
CALL TRANSACTION_JAMES		✓	0.131 s
-- Caller query SELECT * FROM BankAccounts		✓	0.009 s
SELECT * FROM ShoeShop		✓	0.002 s
Untitled - 7	Aug 20, 2023 9:11:50 PM	1	0.065 s
CREATE PROCEDURE TRANSACTION_JAMES -- Name of this stored procedure routine LANGUAGE SQL --		✓	0.065 s
Untitled - 5	Aug 20, 2023 8:19:42 PM	3	0.208 s
CALL TRANSACTION_ROSE		✓	0.191 s
-- Caller query SELECT * FROM BankAccounts		✓	0.010 s
SELECT * FROM ShoeShop		✓	0.007 s

Confirm JAMES Account balance

After the stored procedure call, nothing changed on the account balance of ROSE as shown below. **HERE IS WHY:** The first three UPDATES should run successfully. Both the balance of james and ShoeShop should have been updated in the BankAccounts table. The current balance of James should stand at $1345 - 1200$ (price of 4 Trainers, 300×4) = 1200. The current balance of ShoeShop should stand at $124200 + 1200$ = 125400. The stock of Trainers should also be updated in the ShoeShop table after the successful purchase for James, $14 - 4$ = 10. The last UPDATE statement tries to buy James a Brogues, **but his balance becomes insufficient** (Current balance of james: $145 < \text{Price of Brogues: } 150$) after buying a 4 Trainers. So, the last UPDATE statement fails. **Since the whole transaction fails if any of the SQL statements fail, the transaction won't be committed and the database data records remains unchanged leaving JAMES money untampered**

The screenshot shows the IBM Db2 on Cloud interface. The top navigation bar includes various icons and a search bar. The main area displays a SQL query in a text editor, which has been executed. The results are shown in a table format below the query editor.

SQL Query:

```
1 CALL TRANSACTION_JAMES; -- Caller query
2 SELECT * FROM BankAccounts;
3 SELECT * FROM ShoeShop;
```

Results:

ACCOUNTNUMBER	ACCOUNTNAME	BALANCE
B001	Rose	300.00
B002	James	1345.00
B003	Shoe Shop	124200.00
B004	Corner Shop	76000.00

Confirm ShoeShop product inventory details for Trainers and Brogues

After the **TRANSACTION_JAMES** call, nothing changed on the quantity of **Trainer and Brogues**. **HERE IS WHY**: since one of the **ACID** update transactions failed, all the transaction rolled back and nothing happened to the quantity of Trainers/Brogues available in the inventory database which remained at 14 and 10 respectively. **So the store owner has nothing to worry about!**

The screenshot shows the IBM Db2 on Cloud web interface. The browser address bar displays the URL: `bs2ipcul0apon0jufi80lite.db2.cloud.ibm.com/crn%3Av1%3Abluemix%3Apublic%3Adashdb-for-transactions%3Aeu-gb%3Aa%2F40637218e8f44d7f9a386...`. The interface has a dark header bar with the text "IBM Db2 on Cloud".

On the left sidebar, the "SQL" tab is selected. The main area shows a SQL editor with the following query:

```
1 CALL TRANSACTION_JAMES; -- Caller query
2 SELECT * FROM BankAccounts;
3 SELECT * FROM ShoeShop;
```

Below the editor, the "Results" tab is active, displaying "Result set 1". The results are shown in a table with 3 columns: PRODUCT, STOCK, and PRICE. The table contains 4 rows of data. The total number of rows is 4.

PRODUCT	STOCK	PRICE
Boots	11	200.00
High heels	8	600.00
Brogues	10	150.00
Trainers	14	300.00

Conclusion

ACID transactions play a crucial role in maintaining the reliability, integrity, and consistency of data, which directly benefits businesses and their customers by providing a solid foundation for accurate and trustworthy operations.



ngwuogbonnaprince@gmail.com