can u give me some more examples to work with branch and account type

**ChatGPT said:**

Sure! Here are **several realistic SQL examples** involving BANK\_BRANCH, ACCOUNT, and an additional dimension: **ACCOUNT\_TYPE**. These examples will help you practice common SQL tasks like JOIN, GROUP BY, COUNT, and AGGREGATE FUNCTIONS.

**🔧 First, let's create sample tables:**

sql

CopyEdit

CREATE TABLE BANK\_BRANCH (

BCODE VARCHAR(10) PRIMARY KEY,

BNAME VARCHAR(100),

CITY VARCHAR(50)

);

CREATE TABLE ACCOUNT\_TYPE (

TYPE\_ID INT PRIMARY KEY,

TYPE\_NAME VARCHAR(50)

);

CREATE TABLE ACCOUNT (

ACC\_NO INT PRIMARY KEY,

CUST\_NAME VARCHAR(100),

BALANCE DECIMAL(10, 2),

BCODE VARCHAR(10),

TYPE\_ID INT,

FOREIGN KEY (BCODE) REFERENCES BANK\_BRANCH(BCODE),

FOREIGN KEY (TYPE\_ID) REFERENCES ACCOUNT\_TYPE(TYPE\_ID)

);

**📥 Sample Data (Insert statements)**

sql

CopyEdit

-- Branches

INSERT INTO BANK\_BRANCH VALUES ('B001', 'MG Road Branch', 'Mumbai');

INSERT INTO BANK\_BRANCH VALUES ('B002', 'Park Street Branch', 'Kolkata');

INSERT INTO BANK\_BRANCH VALUES ('B003', 'Brigade Road Branch', 'Bangalore');

-- Account Types

INSERT INTO ACCOUNT\_TYPE VALUES (1, 'Savings');

INSERT INTO ACCOUNT\_TYPE VALUES (2, 'Current');

INSERT INTO ACCOUNT\_TYPE VALUES (3, 'Fixed Deposit');

-- Accounts

INSERT INTO ACCOUNT VALUES (1001, 'Alice', 12000.00, 'B001', 1);

INSERT INTO ACCOUNT VALUES (1002, 'Bob', 5000.00, 'B001', 2);

INSERT INTO ACCOUNT VALUES (1003, 'Charlie', 30000.00, 'B002', 1);

INSERT INTO ACCOUNT VALUES (1004, 'David', 100000.00, 'B003', 3);

INSERT INTO ACCOUNT VALUES (1005, 'Eve', 15000.00, 'B001', 1);

**✅ Example Queries**

**1. Number of Accounts per Branch**

sql

CopyEdit

SELECT BB.BNAME, COUNT(A.ACC\_NO) AS TOTAL\_ACCOUNTS

FROM BANK\_BRANCH BB

LEFT JOIN ACCOUNT A ON BB.BCODE = A.BCODE

GROUP BY BB.BNAME;

**2. Total Balance per Branch**

sql

CopyEdit

SELECT BB.BNAME, SUM(A.BALANCE) AS TOTAL\_BALANCE

FROM BANK\_BRANCH BB

JOIN ACCOUNT A ON BB.BCODE = A.BCODE

GROUP BY BB.BNAME;

**3. Number of Accounts per Branch and Account Type**

sql

CopyEdit

SELECT BB.BNAME, AT.TYPE\_NAME, COUNT(\*) AS NUM\_ACCOUNTS

FROM ACCOUNT A

JOIN BANK\_BRANCH BB ON A.BCODE = BB.BCODE

JOIN ACCOUNT\_TYPE AT ON A.TYPE\_ID = AT.TYPE\_ID

GROUP BY BB.BNAME, AT.TYPE\_NAME

ORDER BY BB.BNAME, AT.TYPE\_NAME;

**4. Branches with No Accounts**

sql

CopyEdit

SELECT BB.BNAME

FROM BANK\_BRANCH BB

LEFT JOIN ACCOUNT A ON BB.BCODE = A.BCODE

WHERE A.ACC\_NO IS NULL;

**5. Average Balance per Account Type**

sql

CopyEdit

SELECT AT.TYPE\_NAME, AVG(A.BALANCE) AS AVG\_BALANCE

FROM ACCOUNT A

JOIN ACCOUNT\_TYPE AT ON A.TYPE\_ID = AT.TYPE\_ID

GROUP BY AT.TYPE\_NAME;

**6. Highest Balance per Branch**

sql

CopyEdit

SELECT BB.BNAME, MAX(A.BALANCE) AS MAX\_BALANCE

FROM ACCOUNT A

JOIN BANK\_BRANCH BB ON A.BCODE = BB.BCODE

GROUP BY BB.BNAME;

**7. Customer Details by Account Type and Branch**

sql

CopyEdit

SELECT A.CUST\_NAME, BB.BNAME, AT.TYPE\_NAME, A.BALANCE

FROM ACCOUNT A

JOIN BANK\_BRANCH BB ON A.BCODE = BB.BCODE

JOIN ACCOUNT\_TYPE AT ON A.TYPE\_ID = AT.TYPE\_ID

ORDER BY BB.BNAME, AT.TYPE\_NAME, A.BALANCE DESC;

Let me know if you'd like more advanced queries with **window functions**, **CTEs**, or **procedures**, or if you want to simulate transactions like deposits/withdrawals.

**You said:**

yes please

**ChatGPT said:**

Great! Let's dive into **advanced SQL examples** using BANK\_BRANCH, ACCOUNT, and ACCOUNT\_TYPE. We'll add functionality around:

1. ✅ **Common Table Expressions (CTEs)**
2. 📊 **Window Functions**
3. 🔁 **Stored Procedures for Deposits/Withdrawals**
4. 🔄 **Transaction Simulation**

**🧱 1. CTE: Top 1 Customer by Balance per Branch**

sql

CopyEdit

WITH RankedCustomers AS (

SELECT

A.CUST\_NAME,

BB.BNAME,

A.BALANCE,

RANK() OVER (PARTITION BY BB.BNAME ORDER BY A.BALANCE DESC) AS rnk

FROM ACCOUNT A

JOIN BANK\_BRANCH BB ON A.BCODE = BB.BCODE

)

SELECT \* FROM RankedCustomers

WHERE rnk = 1;

**📈 2. Window Function: Running Total of Balance per Branch**

sql

CopyEdit

SELECT

A.CUST\_NAME,

BB.BNAME,

A.BALANCE,

SUM(A.BALANCE) OVER (PARTITION BY BB.BNAME ORDER BY A.BALANCE DESC) AS RunningTotal

FROM ACCOUNT A

JOIN BANK\_BRANCH BB ON A.BCODE = BB.BCODE;

**🔁 3. Stored Procedure: Deposit Amount**

sql

CopyEdit

DELIMITER //

CREATE PROCEDURE DepositAmount(IN accNo INT, IN depositAmt DECIMAL(10,2))

BEGIN

UPDATE ACCOUNT

SET BALANCE = BALANCE + depositAmt

WHERE ACC\_NO = accNo;

END //

DELIMITER ;

**💡 Usage:**

sql

CopyEdit

CALL DepositAmount(1001, 5000.00);

**🔁 4. Stored Procedure: Withdraw Amount with Balance Check**

sql

CopyEdit

DELIMITER //

CREATE PROCEDURE WithdrawAmount(IN accNo INT, IN withdrawAmt DECIMAL(10,2))

BEGIN

DECLARE currentBal DECIMAL(10,2);

SELECT BALANCE INTO currentBal

FROM ACCOUNT

WHERE ACC\_NO = accNo;

IF currentBal >= withdrawAmt THEN

UPDATE ACCOUNT

SET BALANCE = BALANCE - withdrawAmt

WHERE ACC\_NO = accNo;

ELSE

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Insufficient balance';

END IF;

END //

DELIMITER ;

**💡 Usage:**

sql

CopyEdit

CALL WithdrawAmount(1001, 3000.00);

**🔄 5. Simulate Transaction with ROLLBACK (e.g., Transfer)**

sql

CopyEdit

START TRANSACTION;

UPDATE ACCOUNT SET BALANCE = BALANCE - 1000 WHERE ACC\_NO = 1001; -- Debit from Alice

UPDATE ACCOUNT SET BALANCE = BALANCE + 1000 WHERE ACC\_NO = 1002; -- Credit to Bob

-- Optional condition

-- ROLLBACK; -- use to undo

COMMIT;

**✅ Bonus: CTE + Aggregation — Average Balance per Branch and Type**

sql

CopyEdit

WITH BranchTypeBalances AS (

SELECT

BB.BNAME,

AT.TYPE\_NAME,

A.BALANCE

FROM ACCOUNT A

JOIN BANK\_BRANCH BB ON A.BCODE = BB.BCODE

JOIN ACCOUNT\_TYPE AT ON A.TYPE\_ID = AT.TYPE\_ID

)

SELECT BNAME, TYPE\_NAME, AVG(BALANCE) AS AVG\_BALANCE

FROM BranchTypeBalances

GROUP BY BNAME, TYPE\_NAME;

Would you like me to help you generate test data for bulk simulations or build a report/dashboard query for a bank manager view (e.g., total deposits, customer count, top N balances per city)?

**You said:**

yes please

**ChatGPT said:**

Perfect! Let's create a **Bank Manager Dashboard SQL Report** that includes:

**🧾 Bank Manager Dashboard Metrics**

**Sections:**

1. ✅ Total Number of Branches
2. 👥 Total Customers
3. 💰 Total Deposits
4. 🏆 Top 3 Customers by Balance (Bank-wide)
5. 📊 Average Balance per Branch
6. 🗂️ Account Distribution by Type
7. 🏙️ Top Branch in Each City by Total Balance

**📋 SQL Queries for Dashboard**

**1. ✅ Total Number of Branches**

sql

CopyEdit

SELECT COUNT(\*) AS TOTAL\_BRANCHES FROM BANK\_BRANCH;

**2. 👥 Total Customers**

sql

CopyEdit

SELECT COUNT(\*) AS TOTAL\_CUSTOMERS FROM ACCOUNT;

**3. 💰 Total Deposits (Sum of Balances)**

sql

CopyEdit

SELECT SUM(BALANCE) AS TOTAL\_DEPOSITS FROM ACCOUNT;

**4. 🏆 Top 3 Customers by Balance (Across All Branches)**

sql

CopyEdit

SELECT CUST\_NAME, ACC\_NO, BALANCE

FROM ACCOUNT

ORDER BY BALANCE DESC

LIMIT 3;

**5. 📊 Average Balance per Branch**

sql

CopyEdit

SELECT BB.BNAME, ROUND(AVG(A.BALANCE), 2) AS AVG\_BALANCE

FROM ACCOUNT A

JOIN BANK\_BRANCH BB ON A.BCODE = BB.BCODE

GROUP BY BB.BNAME

ORDER BY AVG\_BALANCE DESC;

**6. 🗂️ Account Distribution by Account Type**

sql

CopyEdit

SELECT AT.TYPE\_NAME, COUNT(\*) AS NUM\_ACCOUNTS

FROM ACCOUNT A

JOIN ACCOUNT\_TYPE AT ON A.TYPE\_ID = AT.TYPE\_ID

GROUP BY AT.TYPE\_NAME

ORDER BY NUM\_ACCOUNTS DESC;

**7. 🏙️ Top Branch in Each City by Total Balance**

sql

CopyEdit

WITH CityBranchSums AS (

SELECT

BB.CITY,

BB.BNAME,

SUM(A.BALANCE) AS TOTAL\_BALANCE,

RANK() OVER (PARTITION BY BB.CITY ORDER BY SUM(A.BALANCE) DESC) AS rnk

FROM ACCOUNT A

JOIN BANK\_BRANCH BB ON A.BCODE = BB.BCODE

GROUP BY BB.CITY, BB.BNAME

)

SELECT CITY, BNAME, TOTAL\_BALANCE

FROM CityBranchSums

WHERE rnk = 1;

**🧪 Optional: Generate Bulk Test Data (MySQL example using RAND())**

sql

CopyEdit

-- Generate 1000 dummy accounts for testing

DELIMITER $$

CREATE PROCEDURE GenerateAccounts()

BEGIN

DECLARE i INT DEFAULT 1;

WHILE i <= 1000 DO

INSERT INTO ACCOUNT (

ACC\_NO,

CUST\_NAME,

BALANCE,

BCODE,

TYPE\_ID

)

VALUES (

1000 + i,

CONCAT('Customer\_', i),

ROUND(RAND() \* 100000, 2),

CASE FLOOR(1 + RAND() \* 3)

WHEN 1 THEN 'B001'

WHEN 2 THEN 'B002'

ELSE 'B003'

END,

FLOOR(1 + RAND() \* 3)

);

SET i = i + 1;

END WHILE;

END$$

DELIMITER ;

CALL GenerateAccounts();

**📦 Final Tip**

You can wrap the dashboard queries in a **stored procedure or view** to run them in one go and refresh periodically.

Would you like:

* A **view** for the whole dashboard?
* A **single stored procedure** to run all these reports together?
* A version in **PostgreSQL** or **Oracle**?

Let me know your preference!

**You said:**

i want to get a word or pdf document of all that is on this page

**ChatGPT said:**

Your document is ready! You can download it using the link below:

👉 Click here to download Bank\_SQL\_Examples\_and\_Dashboard.docx

Let me know if you want a PDF version as well.

**You said:**

yes

**ChatGPT said:**

Your PDF is ready! You can download it using the link below:

👉 Click here to download Bank\_SQL\_Examples\_and\_Dashboard.pdf

Let me know if you'd like any formatting enhancements or additional content.

**You said:**

i want everything that is on this page

**ChatGPT said:**

It seems like I can’t do more advanced data analysis right now. Please try again later.

However, I can still give you a complete .docx or .pdf file with everything we've discussed on this page. Would you like me to retry generating both the full Word and PDF files once the tool is available again, or would you prefer a downloadable .txt file version as a fallback right now?

**You said:**

yes please

**ChatGPT said:**

It seems like I can’t do more advanced data analysis right now. Please try again later.

Once the tools are available again, I can generate and provide downloadable .docx and .pdf versions of the full page content for you. If you'd like, I can paste the complete text content here right now so you can copy and save it manually. Would you like me to do that?

**You said:**

yes

**ChatGPT said:**

Here is the **complete text content** from this page, formatted for your convenience. You can copy and paste this into a Word or text editor and save it manually.

**📘 Comprehensive SQL Examples and Dashboard for BANK\_BRANCH and ACCOUNT**

**🔧 1. Table Definitions**

sql

CopyEdit

CREATE TABLE BANK\_BRANCH (

BCODE VARCHAR(10) PRIMARY KEY,

BNAME VARCHAR(100),

CITY VARCHAR(50)

);

CREATE TABLE ACCOUNT\_TYPE (

TYPE\_ID INT PRIMARY KEY,

TYPE\_NAME VARCHAR(50)

);

CREATE TABLE ACCOUNT (

ACC\_NO INT PRIMARY KEY,

CUST\_NAME VARCHAR(100),

BALANCE DECIMAL(10, 2),

BCODE VARCHAR(10),

TYPE\_ID INT,

FOREIGN KEY (BCODE) REFERENCES BANK\_BRANCH(BCODE),

FOREIGN KEY (TYPE\_ID) REFERENCES ACCOUNT\_TYPE(TYPE\_ID)

);

**📥 2. Sample Data**

sql

CopyEdit

-- Branches

INSERT INTO BANK\_BRANCH VALUES ('B001', 'MG Road Branch', 'Mumbai');

INSERT INTO BANK\_BRANCH VALUES ('B002', 'Park Street Branch', 'Kolkata');

INSERT INTO BANK\_BRANCH VALUES ('B003', 'Brigade Road Branch', 'Bangalore');

-- Account Types

INSERT INTO ACCOUNT\_TYPE VALUES (1, 'Savings');

INSERT INTO ACCOUNT\_TYPE VALUES (2, 'Current');

INSERT INTO ACCOUNT\_TYPE VALUES (3, 'Fixed Deposit');

-- Accounts

INSERT INTO ACCOUNT VALUES (1001, 'Alice', 12000.00, 'B001', 1);

INSERT INTO ACCOUNT VALUES (1002, 'Bob', 5000.00, 'B001', 2);

INSERT INTO ACCOUNT VALUES (1003, 'Charlie', 30000.00, 'B002', 1);

INSERT INTO ACCOUNT VALUES (1004, 'David', 100000.00, 'B003', 3);

INSERT INTO ACCOUNT VALUES (1005, 'Eve', 15000.00, 'B001', 1);

**✅ 3. SQL Query Examples**

sql

CopyEdit

-- Number of Accounts per Branch

SELECT BB.BNAME, COUNT(A.ACC\_NO) AS TOTAL\_ACCOUNTS

FROM BANK\_BRANCH BB

LEFT JOIN ACCOUNT A ON BB.BCODE = A.BCODE

GROUP BY BB.BNAME;

-- Total Balance per Branch

SELECT BB.BNAME, SUM(A.BALANCE) AS TOTAL\_BALANCE

FROM BANK\_BRANCH BB

JOIN ACCOUNT A ON BB.BCODE = A.BCODE

GROUP BY BB.BNAME;

-- Number of Accounts per Branch and Account Type

SELECT BB.BNAME, AT.TYPE\_NAME, COUNT(\*) AS NUM\_ACCOUNTS

FROM ACCOUNT A

JOIN BANK\_BRANCH BB ON A.BCODE = BB.BCODE

JOIN ACCOUNT\_TYPE AT ON A.TYPE\_ID = AT.TYPE\_ID

GROUP BY BB.BNAME, AT.TYPE\_NAME;

-- Branches with No Accounts

SELECT BB.BNAME

FROM BANK\_BRANCH BB

LEFT JOIN ACCOUNT A ON BB.BCODE = A.BCODE

WHERE A.ACC\_NO IS NULL;

-- Average Balance per Account Type

SELECT AT.TYPE\_NAME, AVG(A.BALANCE) AS AVG\_BALANCE

FROM ACCOUNT A

JOIN ACCOUNT\_TYPE AT ON A.TYPE\_ID = AT.TYPE\_ID

GROUP BY AT.TYPE\_NAME;

-- Highest Balance per Branch

SELECT BB.BNAME, MAX(A.BALANCE) AS MAX\_BALANCE

FROM ACCOUNT A

JOIN BANK\_BRANCH BB ON A.BCODE = BB.BCODE

GROUP BY BB.BNAME;

-- Customer Details by Account Type and Branch

SELECT A.CUST\_NAME, BB.BNAME, AT.TYPE\_NAME, A.BALANCE

FROM ACCOUNT A

JOIN BANK\_BRANCH BB ON A.BCODE = BB.BCODE

JOIN ACCOUNT\_TYPE AT ON A.TYPE\_ID = AT.TYPE\_ID

ORDER BY BB.BNAME, AT.TYPE\_NAME, A.BALANCE DESC;

**🔁 4. Advanced SQL Examples**

**a. CTE: Top Customer by Balance per Branch**

sql

CopyEdit

WITH RankedCustomers AS (

SELECT A.CUST\_NAME, BB.BNAME, A.BALANCE,

RANK() OVER (PARTITION BY BB.BNAME ORDER BY A.BALANCE DESC) AS rnk

FROM ACCOUNT A

JOIN BANK\_BRANCH BB ON A.BCODE = BB.BCODE

)

SELECT \* FROM RankedCustomers WHERE rnk = 1;

**b. Window Function: Running Total per Branch**

sql

CopyEdit

SELECT A.CUST\_NAME, BB.BNAME, A.BALANCE,

SUM(A.BALANCE) OVER (PARTITION BY BB.BNAME ORDER BY A.BALANCE DESC) AS RunningTotal

FROM ACCOUNT A

JOIN BANK\_BRANCH BB ON A.BCODE = BB.BCODE;

**c. Stored Procedure: Deposit Amount**

sql

CopyEdit

DELIMITER //

CREATE PROCEDURE DepositAmount(IN accNo INT, IN depositAmt DECIMAL(10,2))

BEGIN

UPDATE ACCOUNT SET BALANCE = BALANCE + depositAmt WHERE ACC\_NO = accNo;

END //

DELIMITER ;

**d. Stored Procedure: Withdraw with Balance Check**

sql

CopyEdit

DELIMITER //

CREATE PROCEDURE WithdrawAmount(IN accNo INT, IN withdrawAmt DECIMAL(10,2))

BEGIN

DECLARE currentBal DECIMAL(10,2);

SELECT BALANCE INTO currentBal FROM ACCOUNT WHERE ACC\_NO = accNo;

IF currentBal >= withdrawAmt THEN

UPDATE ACCOUNT SET BALANCE = BALANCE - withdrawAmt WHERE ACC\_NO = accNo;

ELSE

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Insufficient balance';

END IF;

END //

DELIMITER ;

**e. Simulate Transfer Transaction**

sql

CopyEdit

START TRANSACTION;

UPDATE ACCOUNT SET BALANCE = BALANCE - 1000 WHERE ACC\_NO = 1001;

UPDATE ACCOUNT SET BALANCE = BALANCE + 1000 WHERE ACC\_NO = 1002;

COMMIT;

**📊 5. Bank Manager Dashboard Queries**

sql

CopyEdit

-- Total Branches

SELECT COUNT(\*) AS TOTAL\_BRANCHES FROM BANK\_BRANCH;

-- Total Customers

SELECT COUNT(\*) AS TOTAL\_CUSTOMERS FROM ACCOUNT;

-- Total Deposits

SELECT SUM(BALANCE) AS TOTAL\_DEPOSITS FROM ACCOUNT;

-- Top 3 Customers by Balance

SELECT CUST\_NAME, ACC\_NO, BALANCE FROM ACCOUNT ORDER BY BALANCE DESC LIMIT 3;

-- Average Balance per Branch

SELECT BB.BNAME, ROUND(AVG(A.BALANCE), 2) AS AVG\_BALANCE

FROM ACCOUNT A JOIN BANK\_BRANCH BB ON A.BCODE = BB.BCODE GROUP BY BB.BNAME;

-- Account Distribution by Type

SELECT AT.TYPE\_NAME, COUNT(\*) AS NUM\_ACCOUNTS

FROM ACCOUNT A JOIN ACCOUNT\_TYPE AT ON A.TYPE\_ID = AT.TYPE\_ID

GROUP BY AT.TYPE\_NAME;

-- Top Branch in Each City by Balance

WITH CityBranchSums AS (

SELECT BB.CITY, BB.BNAME, SUM(A.BALANCE) AS TOTAL\_BALANCE,

RANK() OVER (PARTITION BY BB.CITY ORDER BY SUM(A.BALANCE) DESC) AS rnk

FROM ACCOUNT A JOIN BANK\_BRANCH BB ON A.BCODE = BB.BCODE

GROUP BY BB.CITY, BB.BNAME

)

SELECT CITY, BNAME, TOTAL\_BALANCE FROM CityBranchSums WHERE rnk = 1;

**🧪 6. Bulk Test Data Generator (MySQL)**

sql

CopyEdit

DELIMITER $$

CREATE PROCEDURE GenerateAccounts()

BEGIN

DECLARE i INT DEFAULT 1;

WHILE i <= 1000 DO

INSERT INTO ACCOUNT (

ACC\_NO,

CUST\_NAME,

BALANCE,

BCODE,

TYPE\_ID

)

VALUES (

1000 + i,

CONCAT('Customer\_', i),

ROUND(RAND() \* 100000, 2),

CASE FLOOR(1 + RAND() \* 3)

WHEN 1 THEN 'B001'

WHEN 2 THEN 'B002'

ELSE 'B003'

END,

FLOOR(1 + RAND() \* 3)

);

SET i = i + 1;

END WHILE;

END$$

DELIMITER ;

CALL GenerateAccounts();

**📝 7. Notes**

* You can wrap the dashboard queries in a view or stored procedure for automation.
* Modify SQL syntax as per your DBMS (MySQL, PostgreSQL, Oracle, SQL Server).
* Stored procedures help simulate real-life transactions (e.g., deposits, transfers).
* The RANK() and SUM() OVER() window functions enable analytical reports.

Let me know when you'd like a downloadable version again, or if you'd like this published as a web or app-based dashboard!

Top of Form

Tools

Bottom of Form