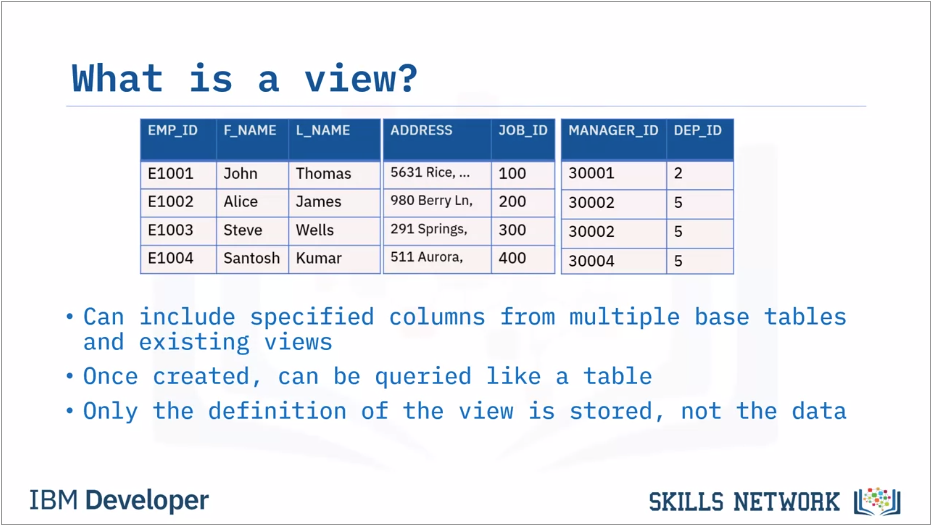
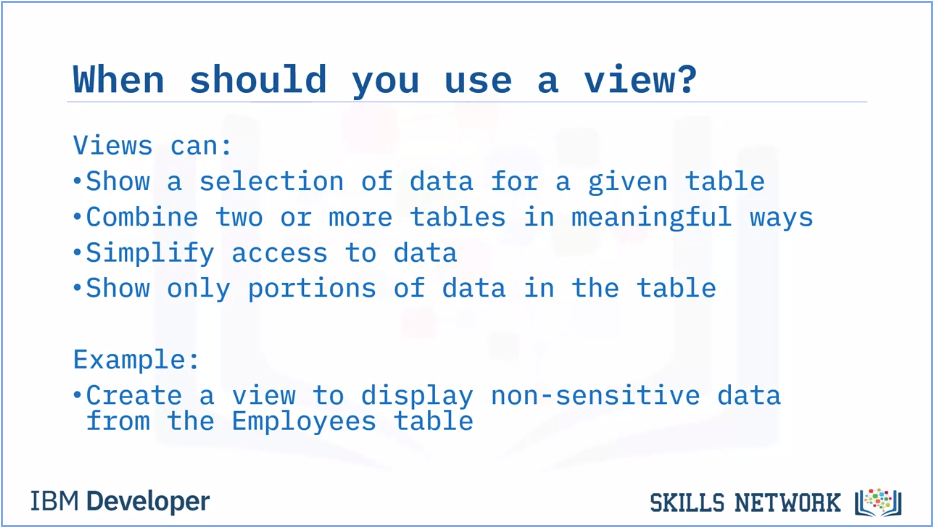
Views, Stored Procedures, and Transactions

**Views**

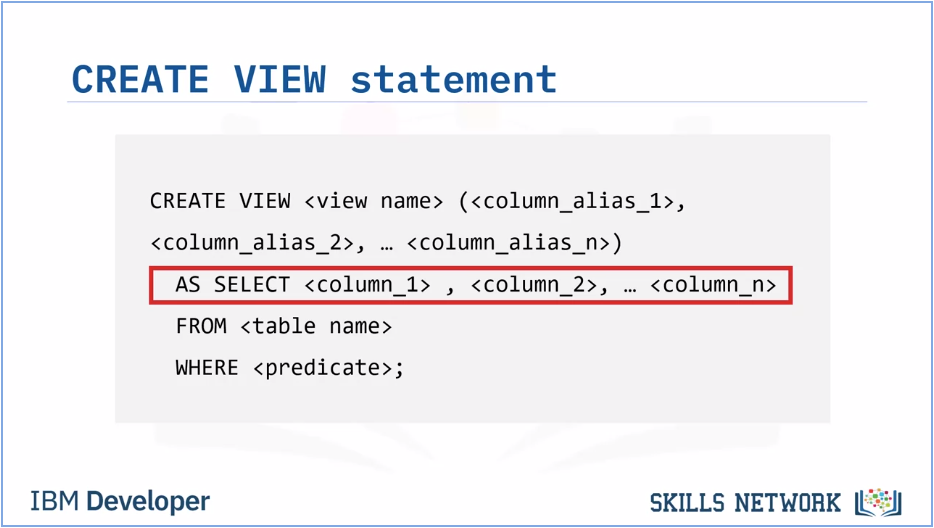
Views are an alternate way of accessing data in tables. They can include specified columns from multiple base tables and existing views. Once created, views can be queried like a table, and the data in the base table can be modified through the view. Views are dynamic; only the definition of the view is stored, not the data. We can use the CREATE VIEW statement to create a view based on one or more tables or existing views.



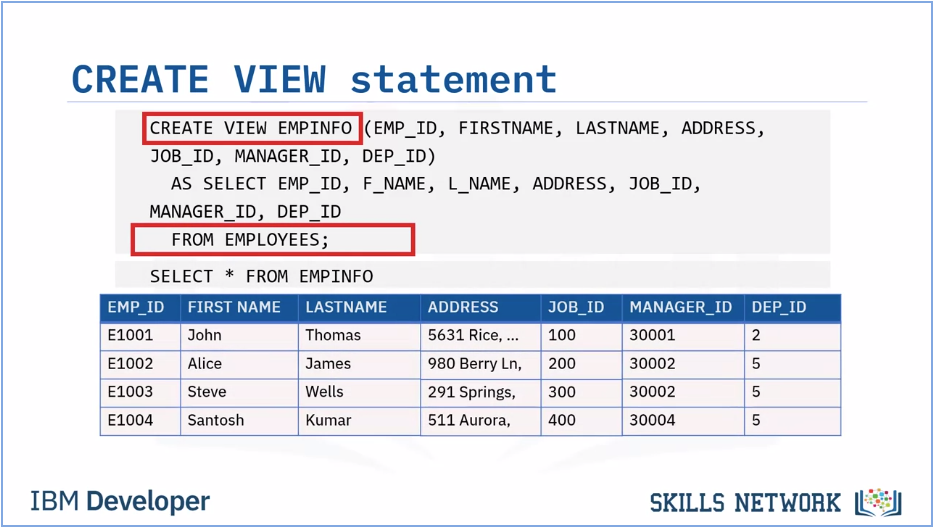
* A view is an alternative way of representing data that exists in one or more tables or views. A view can include all or some of the columns from one or more base tables or existing views. Creating a view creates a named specification of a results table, which can be queried in the same way as a table. We can also change the data in the base table by running insert, update, and delete queries against the view. When we define a view, the definition of the view is stored. The data that the view represents is stored in the base tables, not by the view itself.



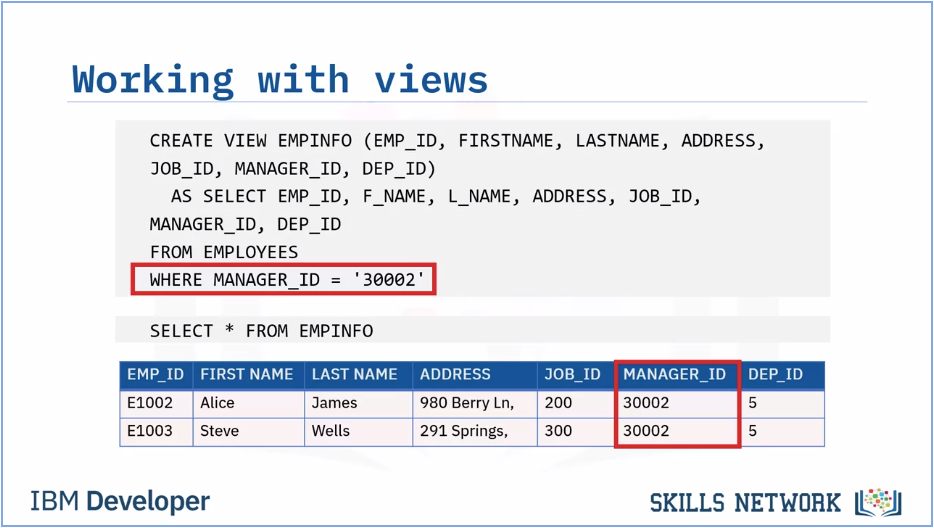
* We can use a view to: Show a selection of data for a given table, so we can omit sensitive data like tax information, birth dates, or salaries.
* Combine two or more tables in meaningful ways.
* Simplify access to data by granting access to a view without granting access to the underlying tables.
* Show only the portions of data relevant to the process that uses the view. For example, we can create a view that displays only non-sensitive data from the Employees table; Employee ID, name, address, job ID, manager ID, and department ID. The view does not show sensitive data like salary or birthdate.



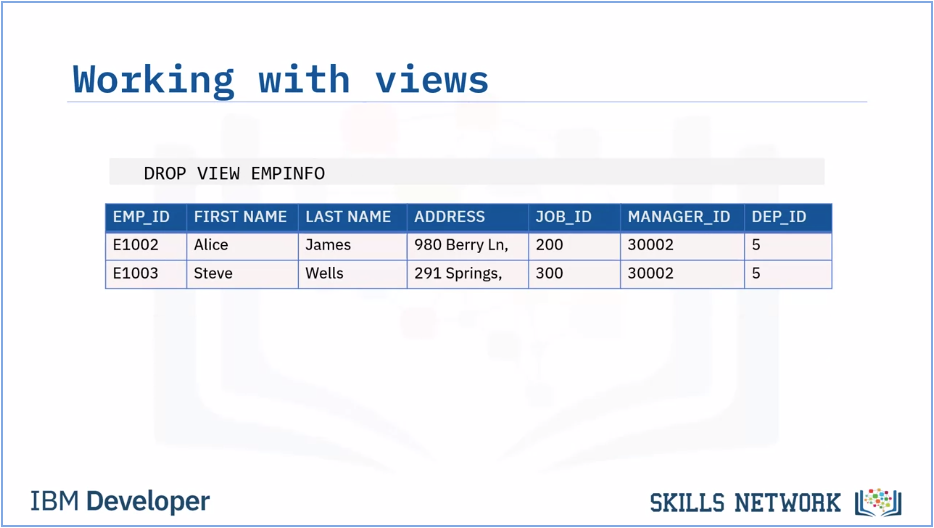
* We use the CREATE VIEW statement to create a view based on one or more tables or views. To define a view, use the CREATE VIEW statement and assign a name (up to 128 characters in length) to the view. List the columns that we want to include. We can use an alias to name the columns if we wish. Use the AS SELECT clause to specify the columns in the view, and the FROM clause to specify the base table name. We can also add an optional WHERE clause to refine the rows in the view.



* This CREATE VIEW statement, creates a view called EMPINFO based on the Employees table. The SELECT statement returns the data in the view, as shown in the table below. Views are dynamic; they consist of the data that would be returned from the SELECT statement used to create them. When we use a view in another SQL statement, it behaves as though we have used a SELECT statement that returns the content of the view.



* The SELECT statement that we use to create the view can name other views and tables, and it can use the WHERE, GROUP BY, and HAVING clauses.
* It cannot use the ORDER BY clause or name a host variable.
* In this example the EMPINFO view is created with only the rows where the MANAGER\_ID is 30002.
* We can use a SELECT statement to show the information from the view, and verify that only rows where the MANAGER\_ID is 30002 are included.



* If we need to remove a view completely, <click> we can use DROP VIEW.

If we want to replace a View:

CREATE OR REPLACE VIEW view\_name AS

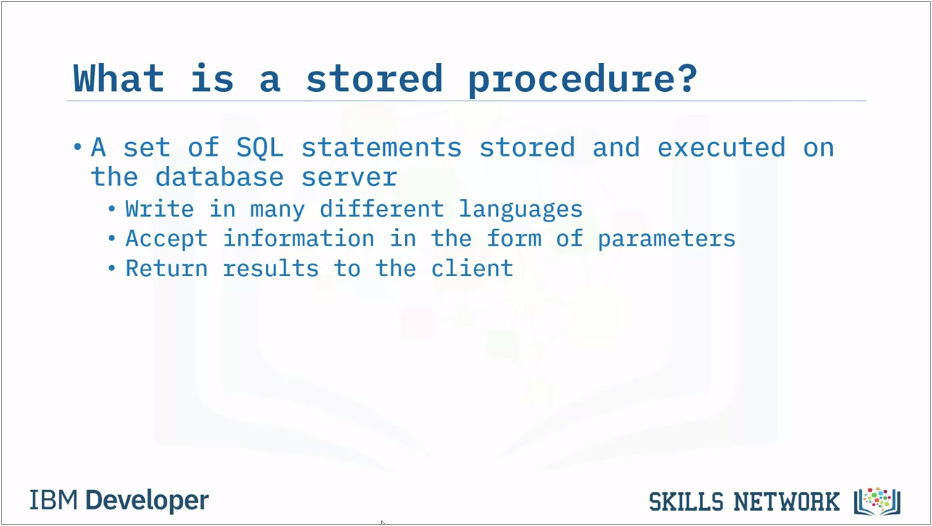
SELECT column1, column2, ...

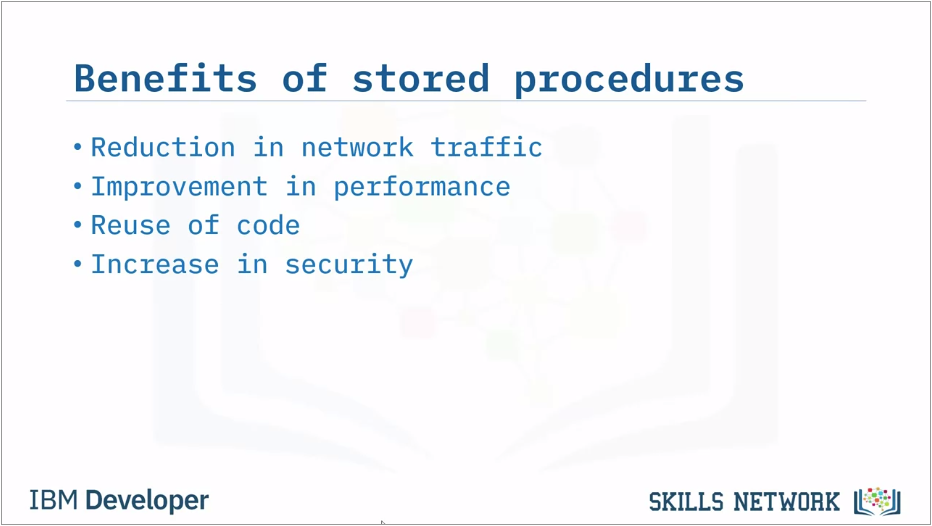
FROM table\_name

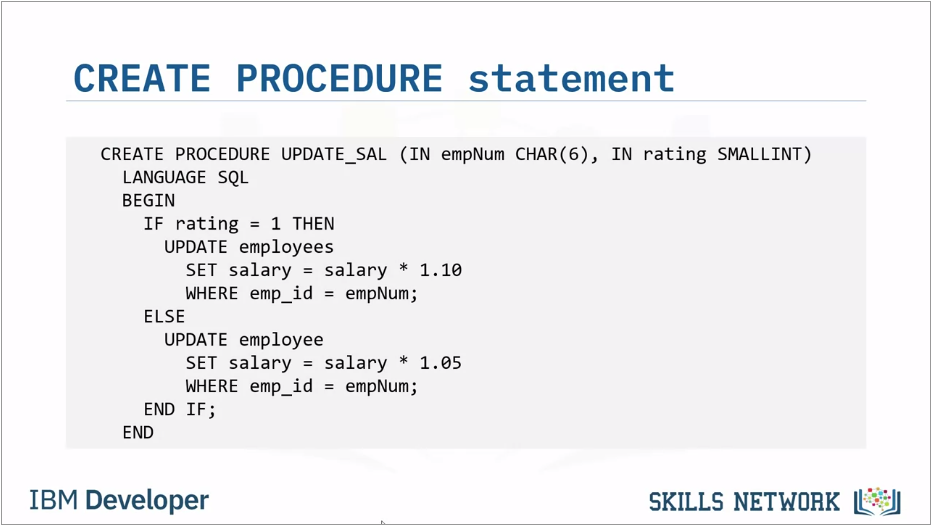
WHERE condition;

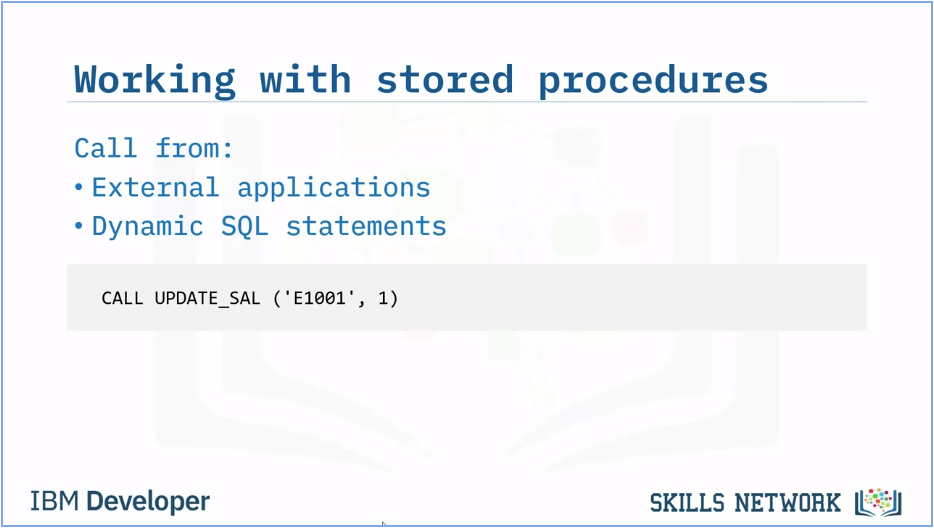
**Stored Procedures**

* Stored procedures are a set of SQL statements that execute on the server
* Stored procedures offer many benefits over sending SQL statements to the server
* We can use stored procedures in dynamic SQL statements and external applications



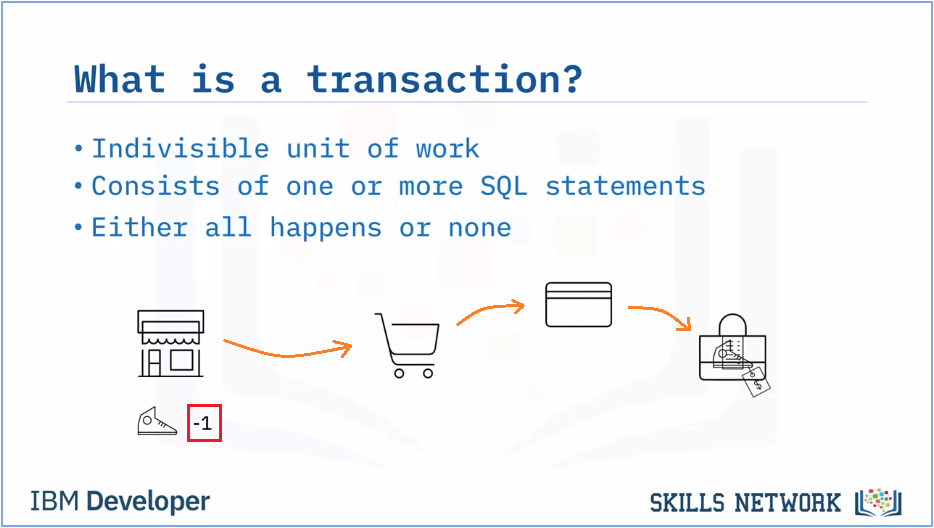




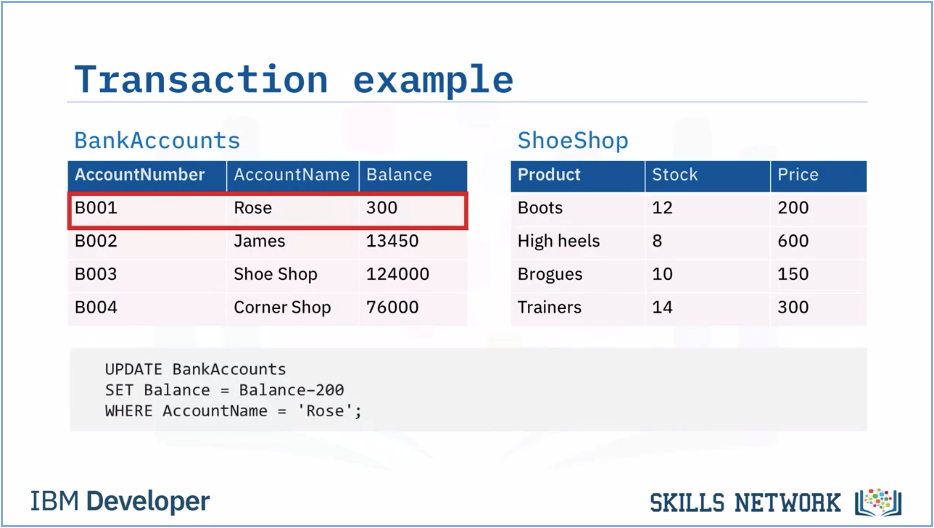


**ACID Transactions**

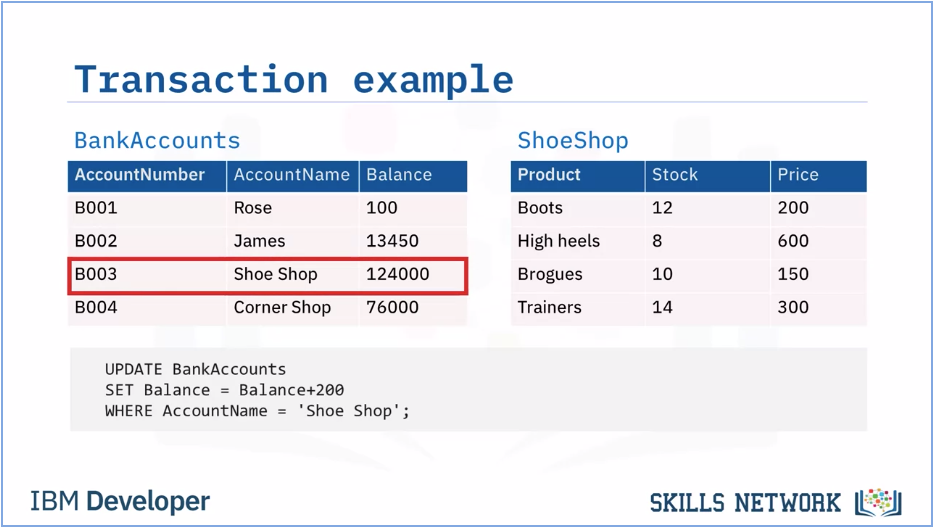
* A transaction represents a complete unit of work, which can be one or more SQL statements.
* An ACID transaction is one where all the SQL statements must complete successfully or none at all. This ensures the database is always in a consistent state.
* ACID stands for Atomic, Consistent, Isolated, Durable.
* SQL commands BEGIN, COMMIT, and ROLLBACK are used to manage ACID transactions. SQL commands can be called from languages like C, R and Python.



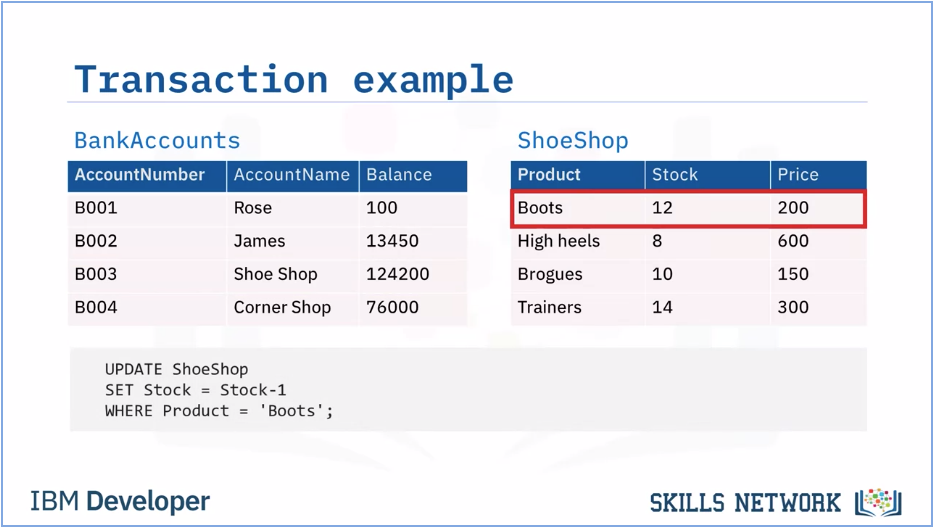
* For example, if we make a purchase using our bank card, many things must happen: The product must be added to our cart Our payment must be processed - Our account must be debited the correct amount and the store's account credited The inventory for that product must be reduced by the number purchased.



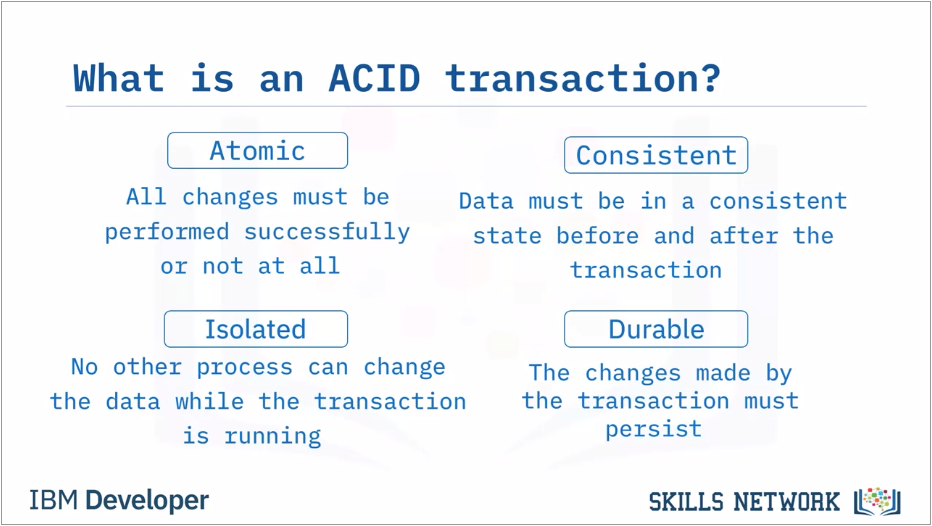
* If Rose buys boots for $200, then we can use an UPDATE statement to decrease her account balance.



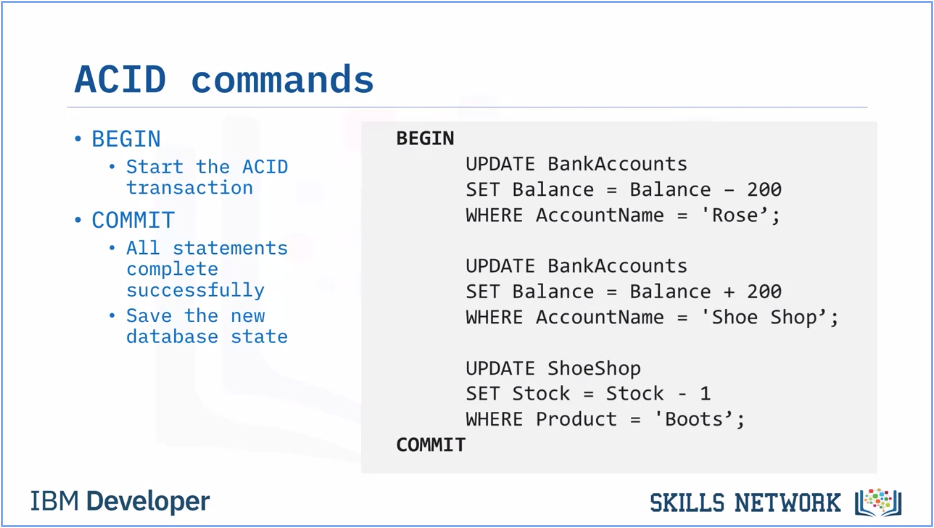
* And another UDATE statement to add $200 to the Shoe Shop balance.



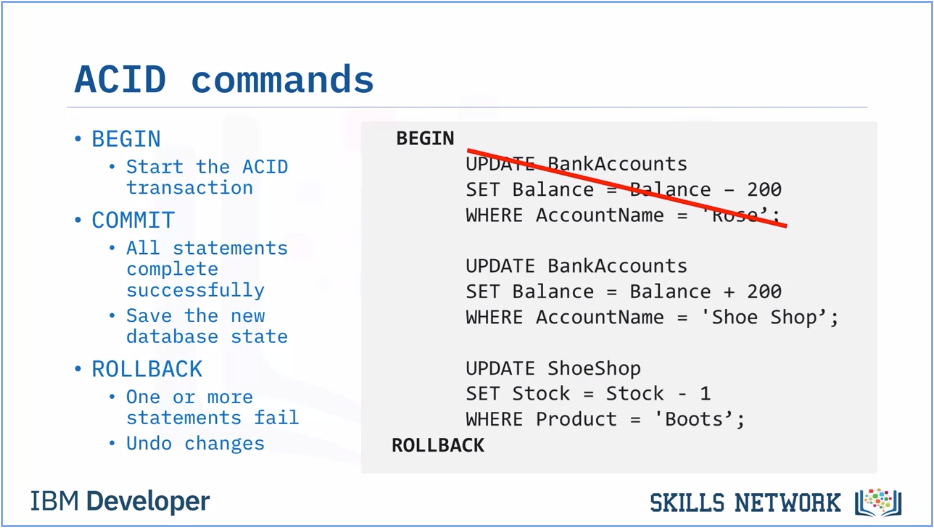
* And a final update statement to decrease the stock level of boots at the Shoe Shop by 1. If any of these UPDATE statements fail, the whole transaction should fail, to keep the data in a consistent state.



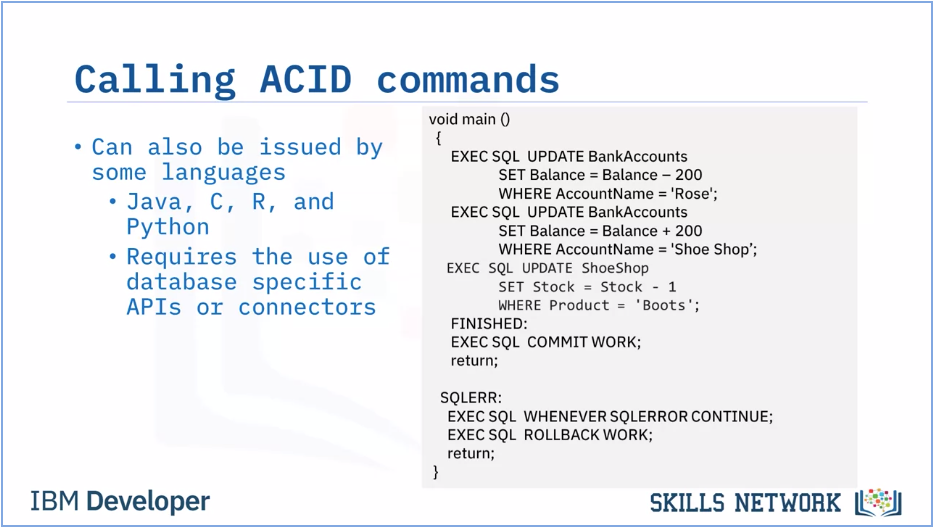
* The types of transaction in the examples are called ACID transactions.



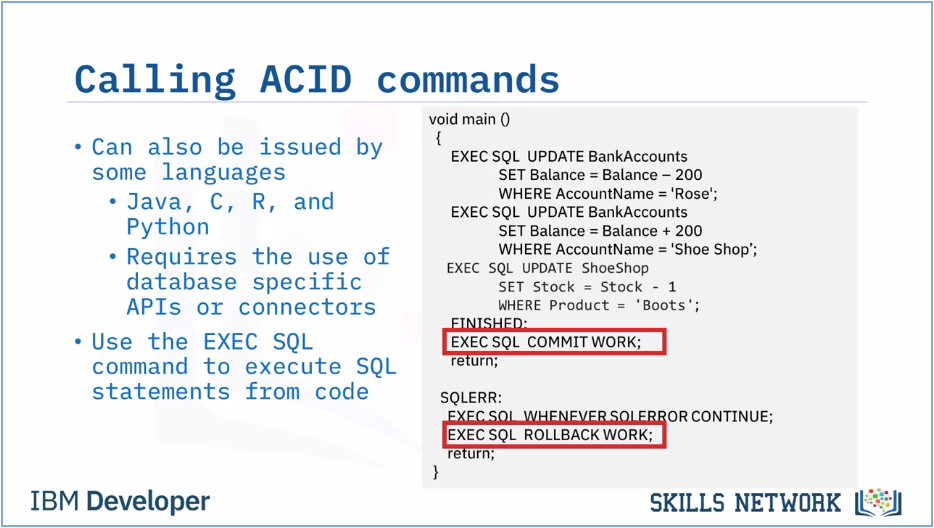
* To start an ACID transaction, use the command BEGIN. In db2 on Cloud, this command is implicit. Any commands we issue after that are part of the transaction, until we issue either COMMIT, or ROLLBACK.
* If all the commands complete successfully, issue a commit command to save everything in the database to a consistent, stable state.



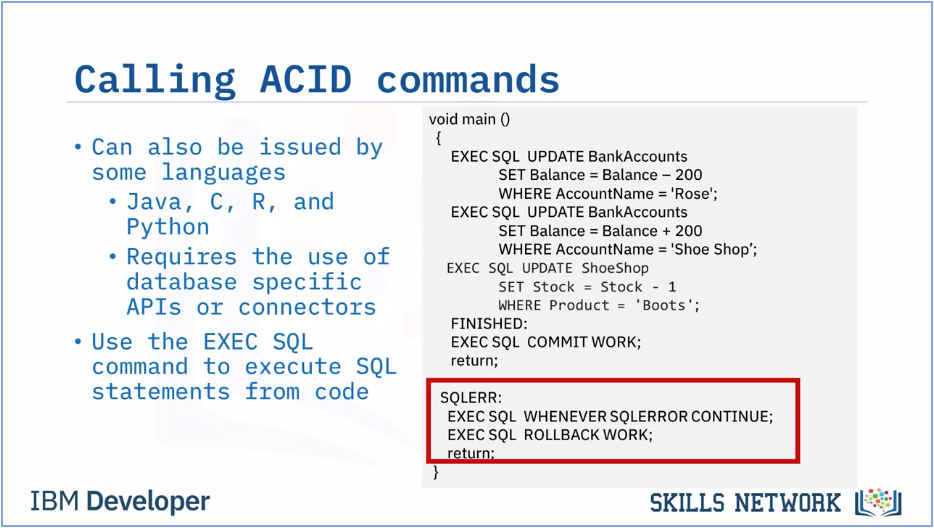
* If any of the commands fail; perhaps Rose’s account doesn’t have enough money to make the payment, we can issue a rollback command to undo all the changes and leave the database in its previously consistent stable state.



* SQL statements can be called from languages like Java, C, R, and Python. This requires the use of database-specific access APIs such as Java Database Connectivity (JDBC) for Java or a specific database connector like ibm\_db for Python.



Most languages use the EXEC SQL commands to initiate a SQL command, including COMMIT and ROLLBACK, as we can see in this example. Remember that BEGIN is implicit, we do not need to call it out explicitly.



Incorporating SQL commands into our application code gives we the opportunity to create error-checking routines that in turn control whether the transaction is committed or rolled back.

**Summary & Highlights**

* Views are a dynamic mechanism for presenting data from one or more tables.A transaction represents a complete unit of work, which can be one or more SQL statements.
* An ACID transaction is one where all the SQL statements must complete successfully, or none at all.
* A stored procedure is a set of SQL statements that are stored and executed on the database server, allowing we to send one statement as an alternative to sending multiple statements.
* We can write stored procedures in many different languages like SQL PL, PL/SQL, Java, and C.