

SQL Cheat Sheet: Views, Stored Procedures and Transactions



Views

Topic	Syntax	Description	Example
Create View	CREATE VIEW view_name AS SELECT column1, column2, ... FROM table_name WHERE condition;	A CREATE VIEW is an alternative way of representing data that exists in one or more tables.	CREATE VIEW EMP_SALARY AS SELECT EMP_ID, F_NAME, L_NAME, B_DATE, SEX, SALARY FROM EMPLOYEES;
Update a View	CREATE OR REPLACE VIEW view_name AS SELECT column1, column2, ... FROM table_name WHERE condition;	The CREATE OR REPLACE VIEW command updates a view.	CREATE OR REPLACE VIEW EMP_SALARY AS SELECT EMP_ID, F_NAME, L_NAME, B_DATE, SEX, JOB_TITLE, MIN_SALARY, MAX_SALARY FROM EMPLOYEES, JOBS WHERE EMPLOYEES.JOB_ID = JOBS.JOB_IDENT;
Drop a View	DROP VIEW view_name;	Use the DROP VIEW statement to remove a view from the database.	DROP VIEW EMP_SALARY;

Stored Procedures in IBM Db2 using SQL

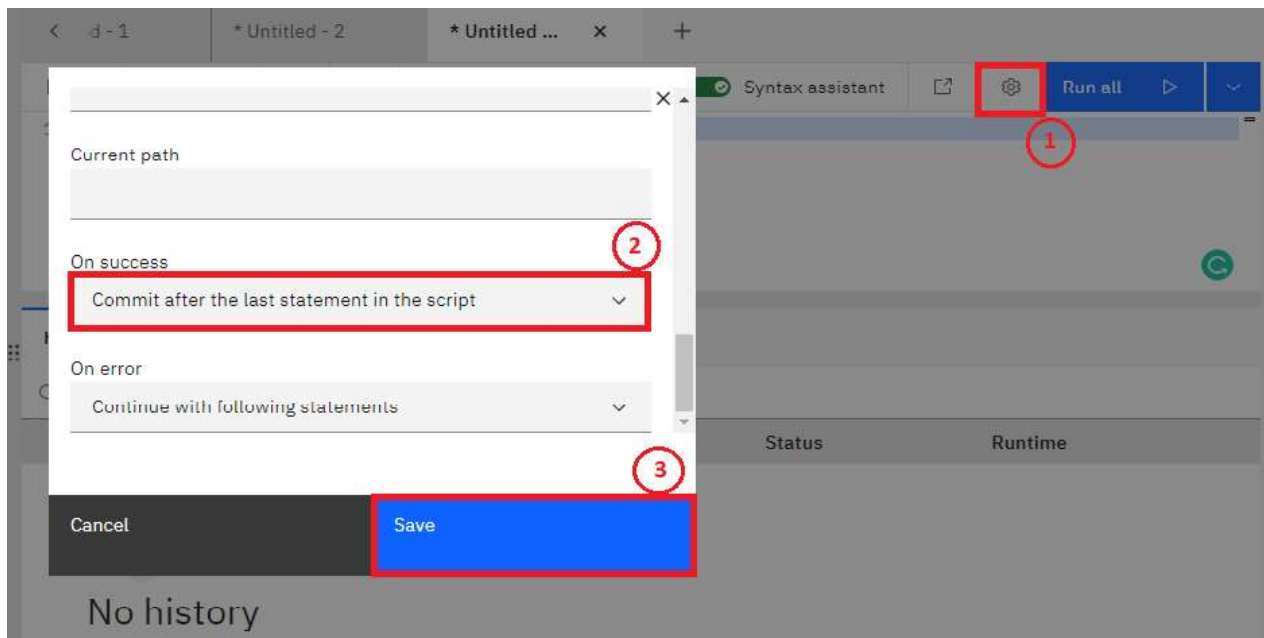
			--#SET TERMINATOR @ CREATE PROCEDURE RETRIEVE_ALL
	--#SET TERMINATOR @ CREATE PROCEDURE PROCEDURE_NAME		LANGUAGE SQL
	LANGUAGE	A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.	READS SQL DATA
Stored Procedures	BEGIN		DYNAMIC RESULT SETS 1
	END @	The default terminator for a stored procedure is semicolon(;). To set a different terminator we use SET TERMINATOR clause followed by the terminator such as '@'.	BEGIN
			DECLARE C1 CURSOR WITH RETURN FOR
			SELECT * FROM PETSAL;
			OPEN C1;
			END @

Stored Procedures in MySQL using phpMyAdmin

	DELIMITER //		DELIMITER //
	CREATE PROCEDURE PROCEDURE_NAME	A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.	CREATE PROCEDURE RETRIEVE_ALL()
Stored Procedures	BEGIN		BEGIN
	END //	The default terminator for a stored procedure is semicolon (;). To set a different terminator we use DELIMITER clause followed by the terminator such as \$\$ or //.	SELECT * FROM PETSAL;
	DELIMITER ;		END //
			DELIMITER ;

Transactions with Db2

		A COMMIT command is used to persist the changes in the database.	
Commit command	COMMIT;	CREATE TABLE employee(ID INT, Name VARCHAR(20), City VARCHAR(20), Salary INT, Age INT); INSERT INTO employee(ID, Name, City, Salary, Age) VALUES(1, 'Priyanka pal', 'Nasik', 36000, 21), (2, 'Riya chowdary', 'Bangalor', 82000, 29); SELECT *FROM employee; COMMIT;	
		The default terminator for a COMMIT command is semicolon (;).	
Rollback command	ROLLBACK;	As auto-commit is enabled by default, all transactions will be committed. We need to disable this option to see how rollback works. For db2, we have to disable auto-commit manually. Click the gear icon located on the right side of the SQL Assistant window. Next, select the "On Success" drop-down and choose "commit after the last statement in the script" Remember to save your changes!	
		The default terminator for a ROLLBACK command is semicolon (;).	



```
INSERT INTO employee VALUES (3, 'Swetha Tiwari', 'Kanpur', 38000, 38);

SELECT *FROM employee;
ROLLBACK;
SELECT *FROM employee;
```

Transactions with MySQL

Commit command	COMMIT;	<p>A COMMIT command is used to persist the changes in the database.</p> <p>The default terminator for a COMMIT command is semicolon (;).</p>	<pre>CREATE TABLE employee(ID INT, Name VARCHAR(20), City VARCHAR(20), Salary INT, Age INT); START TRANSACTION; INSERT INTO employee(ID, Name, City, Salary, Age) VALUES(1, 'Priyanka pal', 'Nasik', 36000, 21), (2, 'Riya chowdary', 'Bangalor', 82000, 29); SELECT *FROM employee; COMMIT;</pre> <p>As auto-commit is enabled by default, all transactions will be committed. We need to disable this option to see how rollback works. For MySQL use the command "SET autocommit = 0;"</p>
Rollback command	ROLLBACK;	<p>A ROLLBACK command is used to rollback the transactions which are not saved in the database.</p> <p>The default terminator for a ROLLBACK command is semicolon (;).</p>	<pre>INSERT INTO employee VALUES (3, 'Swetha Tiwari', 'Kanpur', 38000, 38); SELECT *FROM employee; ROLLBACK; SELECT *FROM employee;</pre>

Db2 Transactions using Stored Procedure

Commit command	<pre>--#SET TERMINATOR @ CREATE PROCEDURE PROCEDURE_NAME BEGIN COMMIT; END @</pre>	<p>A COMMIT command is used to persist the changes in the database.</p> <p>The default terminator for a COMMIT command is semicolon (;).</p>	<pre>--#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-300 WHERE AccountName = 'Rose'; IF retcode < 0 THEN ROLLBACK WORK; ELSE COMMIT WORK; END IF; END @ --#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;</pre>
Rollback command	<pre>--#SET TERMINATOR @ CREATE PROCEDURE PROCEDURE_NAME BEGIN ROLLBACK; COMMIT;</pre>	<p>A ROLLBACK command is used to rollback the transactions which are not saved in the database.</p> <p>The default terminator for a ROLLBACK command is semicolon (;).</p>	

```
END
@
```

MySQL Transactions using Stored Procedure

Commit
command

```
DELIMITER //

CREATE PROCEDURE PROCEDURE_NAME

BEGIN

COMMIT;

END //

DELIMITER ;
```

A COMMIT command is used to persist the changes in the database.

The default terminator for a COMMIT command is semicolon (;).

Rollback
command

```
DELIMITER //

CREATE PROCEDURE PROCEDURE_NAME

BEGIN

ROLLBACK;

COMMIT;

END //

DELIMITER ;
```

A ROLLBACK command is used to rollback the transactions which are not saved in the database.

The default terminator for a ROLLBACK command is semicolon (;).

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

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

IF retcode < 0 THEN
ROLLBACK WORK;

ELSE
COMMIT WORK;

END IF;

END
@

DELIMITER //

CREATE PROCEDURE TRANSACTION_ROSE()

BEGIN

DECLARE EXIT HANDLER FOR SQLEXCEPTION
BEGIN
ROLLBACK;
RESIGNAL;
END;

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

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

COMMIT;

END //

DELIMITER ;
DELIMITER //

CREATE PROCEDURE TRANSACTION_ROSE()

BEGIN

DECLARE EXIT HANDLER FOR SQLEXCEPTION
BEGIN
ROLLBACK;
RESIGNAL;
END;

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

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

COMMIT;

END //

DELIMITER ;
```

Author(s)

[D.M Naidu](#)

Changelog

Date	Version	Changed by	Change Description
2022-10-04	1.0	D.M.Naidu	Initial Version