Describe the result of a query:
```
DESCRIBE SELECT * FROM tbl1
```

Get help for a Db2 command:
```
? command
```

Get help for a SQL code (SQLXXXX) or SQLstate (YYYYY):
```
? SQLxxxx
? YYYY
```

**DDL**

Create a schema:
```
CREATE SCHEMA sch1
```

Create a table specifying primary key:
```
CREATE TABLE tbl1 (col1 CHAR(1) NOT NULL PRIMARY KEY)
CREATE TABLE tbl2 (col1 INT NOT NULL, col2 DATE NOT NULL, PRIMARY KEY (col1, col2))
```

Create a table specifying tablespaces:
```
CREATE TABLE tbl3 (col1 INT NOT NULL, col2 CHAR(1)) IN ts1 INDEX IN ts2
```

Create a table specifying schema:
```
CREATE TABLE sch1.tbl14 (col1 INT)
```

Create a table with auto incremental column:
```
CREATE TABLE tbl5 (col1 INT NOT NULL GENERATED AS IDENTITY)
```

Create a table like another one:
```
CREATE TABLE tbl6 LIKE tbl1 IN ts1 INDEX IN ts2
```

Comment on table and column:
```
COMMENT ON TABLE tbl1 IS 'Comment in table'
COMMENT ON COLUMN tbl1.col1 IS 'Description of the field'
```

Declare a temporary table (session schema):
```
DECLARE GLOBAL TEMPORARY TABLE tmp1 (col1 INT, col2 DATE) ON COMMIT PRESERVE ROWS
```

Create a global temporary tablespace:
```
CREATE GLOBAL TEMPORARY TABLE tmp2 (col1 INT)
```

Create an index:
```
CREATE INDEX idx1 ON tbl2 (col2)
```

Create a unique index:
```
CREATE UNIQUE INDEX idx2 ON tbl5 (col1)
```

Drop an index:
```
DROP INDEX idx1
```

Add a column (requires Reorg table):
```
ALTER TABLE tbl1 ADD COLUMN col3 timestamp
```

Change the granularity of the locks:
```
ALTER TABLE tbl1 LOCKSIZE TABLE
```

Drop a table:
```
DROP TABLE tbl1
```

Rename a table:
```
RENAME TABLE tbl2 TO table2
```

Truncate a table:
```
TRUNCATE TABLE tbl1 IMMEDIATE
```

Create a sequence:
```
CREATE SEQUENCE seq AS INTEGER
```

Restart sequence:
```
ALTER SEQUENCE seq RESTART WITH 15
```

Execute a file in the console (db2clp).
- Semi-colon separated sentences:
  - db2 -t
- At sign separated sentences (when there is SQL PL code):
  - db2 -t\@?

Define a terminator character:
```
--\#SET TERMINATOR @
```

List all databases (aliases):
```
LIST DB DIRECTORY
```

Connect to a database (alias):
```
CONNECT TO mydb
```

Disconnect from a database:
```
CONNECT RESET TERMINATE
```

Get values from the environment (registry values).
- Current timestamp:
  - VALUES CURRENT TIMESTAMP
- Connected user:
  - VALUES CURRENT USER
- Current database:
  - VALUES CURRENT SERVER

List all tables:
```
LIST TABLES
LIST TABLES FOR SCHEMA myuser
LIST TABLES FOR ALL
```

Change current schema:
```
SET CURRENT SCHEMA otherschema
```

Change the isolation level (RR, RS, CS, UR):
```
SET ISOLATION RR
```

List all tablespaces with their status:
```
LIST TABLESPACES
```

Describe the structure of the table:
```
DESCRIBE TABLE tbl11
```

Change the isolation level (RR, RS, CS, UR):
```
SET ISOLATION RR
```

Change nullability:
```
ALTER TABLE tbl11 ALTER COLUMN col13 SET NOT NULL
```

Drop nullability:
```
ALTER TABLE tbl11 ALTER COLUMN col13 DROP NOT NULL
```

Rename a column:
```
ALTER TABLE tbl11 RENAME COLUMN col3 TO new3
```

Drop column:
```
ALTER TABLE tbl11 DROP COLUMN new3
```

Create a primary key constraint:
```
ALTER TABLE tbl15 ADD CONSTRAINT pkt5 PRIMARY KEY (col1)
```

Drop primary key:
```
ALTER TABLE tbl15 DROP PRIMARY KEY
```

Add identity:
```
ALTER TABLE tbl2 ALTER col1 SET GENERATED ALWAYS AS IDENTITY
```

Restart identity:
```
ALTER TABLE tbl2 ALTER col1 RESTART WITH 1
```

Drop identity:
```
ALTER TABLE tbl2 ALTER col1 DROP IDENTITY
```

Create a foreign key:
```
ALTER TABLE tbl15 ADD CONSTRAINT fkt5 FOREIGN KEY (col1) REFERENCES tbl11 (col1)
```

Create a check constraint:
```
ALTER TABLE tbl11 ADD CONSTRAINT chk CHECK (col1 in ('a', 'b', 'c'))
```

Enforce a constraint:
```
ALTER TABLE tbl11 ALTER CHECK chk ENFORCED
```

Not enforce a constraint:
```
ALTER TABLE tbl15 ALTER FOREIGN KEY fkt5 NOT ENFORCED
```

Change the granularity of the locks:
```
ALTER TABLE tbl1 LOCKSIZE TABLE
```

Drop a table:
```
DROP TABLE tbl1
```

Rename a table:
```
RENAME TABLE tbl2 TO table2
```

Truncate a table:
```
TRUNCATE TABLE tbl1 IMMEDIATE
```

Create a sequence:
```
CREATE SEQUENCE seq AS INTEGER
```

Restart sequence:
```
ALTER SEQUENCE seq RESTART WITH 15
```

Get help for a Db2 command:
```
? command
```

Get help for a SQL code (SQLXXXX) or SQLstate (YYYYY):
```
? SQLxxxx
? YYYY
```
Create a stored procedure:
CREATE OR REPLACE PROCEDURE prc1 (IN val INT, OUT ret DATE) SPECIFIC mypr BEGIN
SET ret = (SELECT col2 FROM tbl2 WHERE col1 = val);
END @

Create a trigger:
CREATE TRIGGER cp_val AFTER INSERT ON tbl1 REFERENCING NEW AS n FOR EACH ROW
INSERT INTO tbl2 VALUES (n.col1, n.col2)

Create a view:
CREATE VIEW vw1 AS SELECT col2 FROM tbl1

DCL
Grant on a table:
GRANT SELECT, INSERT ON TABLE tbl1 TO user
Grant execution on a stored procedure:
GRANT EXECUTE ON PROCEDURE prc1(INT, DATE) TO USER jdoe
GRANT EXECUTE ON SPECIFIC PROCEDURE mypr TO GROUP admins
Revoke on a table:
REVOKE DELETE ON TABLE mytable FROM recur

DML
Insert values on a table:
INSERT INTO tbl3 VALUES (2, 'b')
INSERT INTO tbl3 VALUES (3, 'c'), (4, 'd'), (5, 'e')
Insert certain columns:
INSERT INTO tbl1 (col1) VALUES (6)
Insert values from a select:
INSERT INTO tbl6 SELECT col1 FROM tbl1
Insert in temporary table:
INSERT INTO session.tmp1 VALUES (1)
Update fields:
UPDATE tbl3 SET col1 = 5, mycol2 = 'e' -- all table
UPDATE tbl3 SET col2 = 'd' WHERE col1 = 7
Merge (upsert):
MERGE INTO tbl3 AS t USING (SELECT col1 FROM tbl1) s ON (t.col1 = s.col1) WHEN MATCHED THEN UPDATE SET col2 = 'X' WHEN NOT MATCHED THEN INSERT VALUES (10, 'X')
Delete rows:
DELETE FROM tbl1 --all table
DELETE FROM tbl1 WHERE col1 > 5

Export:
EXPORT TO myfile OF DEL SELECT * FROM tbl1

Import:
IMPORT FROM myfile OF DEL INSERT INTO mytable1

Cursor:
DECLARE cur1 CURSOR FOR SELECT * FROM tbl1

Load:
LOAD FROM myfile OF DEL INSERT INTO tbl1
LOAD FROM cur1 OF CURSOR INSERT INTO tbl1

Query the status of the load in a table:
LOAD QUERY TABLE tbl1

Set integrity:
SET INTEGRITY FOR tbl1 IMMEDIATE CHECKED

Ingest:
INGEST FROM FILE myfile FORMAT DELIMITED INSERT INTO tbl1

Get the next value from a sequence:
VALUES NEXT VALUE FOR seq
VALUES NEXT VALUE FOR seq

TCL
Commit changes:
COMMIT

Create a savepoint:
SAVEPOINT sp1 ON ROLLBACK RETAIN CURSORS

Undo changes until savepoint:
ROLLBACK TO SAVEPOINT sp1

Undo changes:
ROLLBACK

Queries
Put a lock at table level:
LOCK TABLE tbl1 IN EXCLUSIVE MODE

Execute a query without regard of commit rows:
SELECT * FROM tbl1 WITH UR --RR,RS,CS

Execute a query with only 5 rows:
SELECT * FROM tbl1 FETCH FIRST 5 ROWS ONLY

Perform a query to a dummy table (dual):
SELECT 'Any string' FROM SYSIBM.SYSDUMMY1

Perform a query calling a function:
SELECT HEX(col2) FROM tbl5

Call a function:
VALUES HEX('AnyText')

Perform a cast:
VALUES CAST('123' AS INTEGER)

Concatenate:
VALUES 'AnyText' || 5
VALUES 'AnyText' concat 5

Escape a single quote in a text field:
VALUES 'Sinead O''Connor'

Query the database catalog:
SELECT * FROM SYSCAT.TABLES
SELECT * FROM SYSCAT.TABAUTH
SELECT * FROM SYSCAT.ROUTINES

SQL PL
Create a compound statement – Anonymous block:
BEGIN DECLARE val SMALLINT; SET val = 1;
WHILE (val <= 5) DO INSERT INTO tbl5 VALUES (val, val); SET val = val + 1; END WHILE; END @

Call a stored procedure with an IN and an OUTPUT parameter:
CALL prc1(5, ?)

Perform a reorg via ADMIN_CMD (Sometimes required after “alter table”):
CALL SYSPROC.ADMIN_CMD ('REORG TABLE tbl1')