

Database Programming with SQL 16-1: Working with Sequences Practice Activities

# Objectives

* List at least three useful characteristics of a sequence
* Write and execute a SQL statement that creates a sequence
* Query the data dictionary using USER\_SEQUENCES to confirm a sequence definition
* Apply the rules for using NEXTVAL to generate sequential numbers for use in a table
* List the advantages of caching sequence values
* Name three reasons why gaps can occur in a sequence

# Vocabulary

Identify the vocabulary word for each definition below.

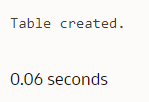
|  |  |
| --- | --- |
| **CREATE SEQUENCE***(CREATE SEQUENCE some\_name)* | Command that automatically generates sequential numbers |
| **Sequences** | Generates a numeric value |
| **NEXTVAL** | Returns the next available sequence value |
| **INCREMENT BY***(INCREMENT BY n)* | Specifies the interval between sequence numbers |
| **NOMAXVALUE** | Specifies a maximum value of 10^27 for an ascending sequence and -1 for a descending sequence (default) |
| **CURRVAL** | returns the current sequence value |
| **MINVALUE***(MINVALUE n)* | specifies the minimum sequence value |
| **CYCLE/NOCYCLE** | specifies whether the sequence continues to generate values after reaching its maximum or minimum values |
| **NOMINVALUE** | specifies a minimum value of 1 for an ascending sequence and – (10^26) for a descending sequence (default) |
| **MAXVALUE/NOMAXVALUE***(MAXVALUE n | NOMAXVALUE)* | specifies a maximum or default value the sequence can generate |
| **START WITH***(START WITH n)* | specifies the first sequence number to be generated |
| **CACHE/NOCACHE***(CACHE n | NOCACHE)* | specifies how many values the Server pre-allocates and keeps in memory |

# Try It / Solve It

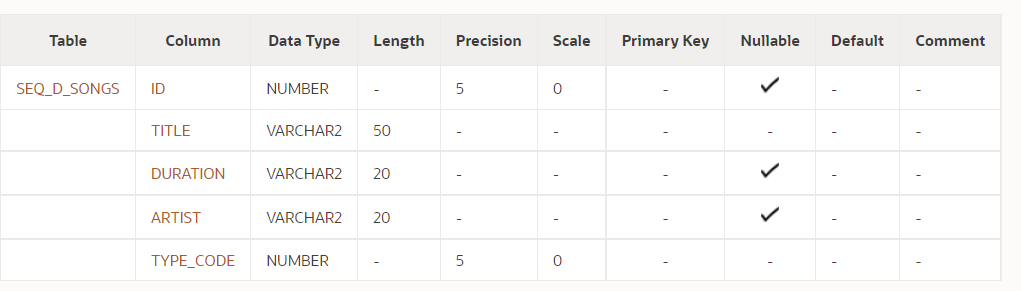
1. Using CREATE TABLE AS subquery syntax, create a seq\_d\_songs table of all the columns in the DJs on Demand database table d\_songs. Use the SELECT \* in the subquery to make sure that you have copied all of the columns.

**CREATE TABLE seq\_d\_songs**

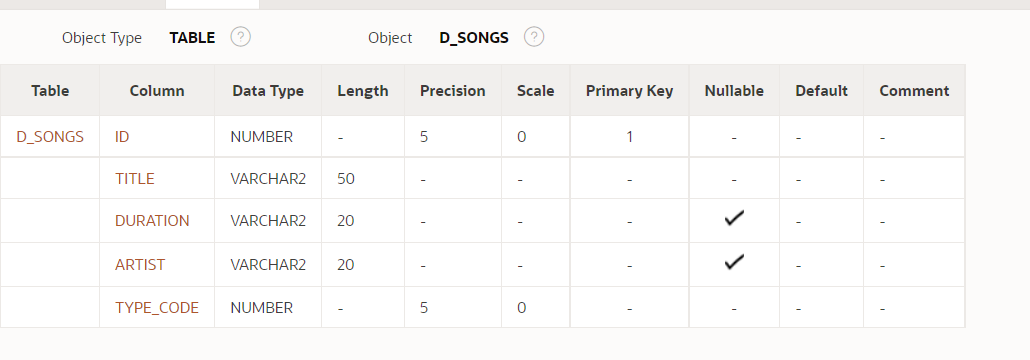
**AS ( SELECT \* FROM d\_songs);**



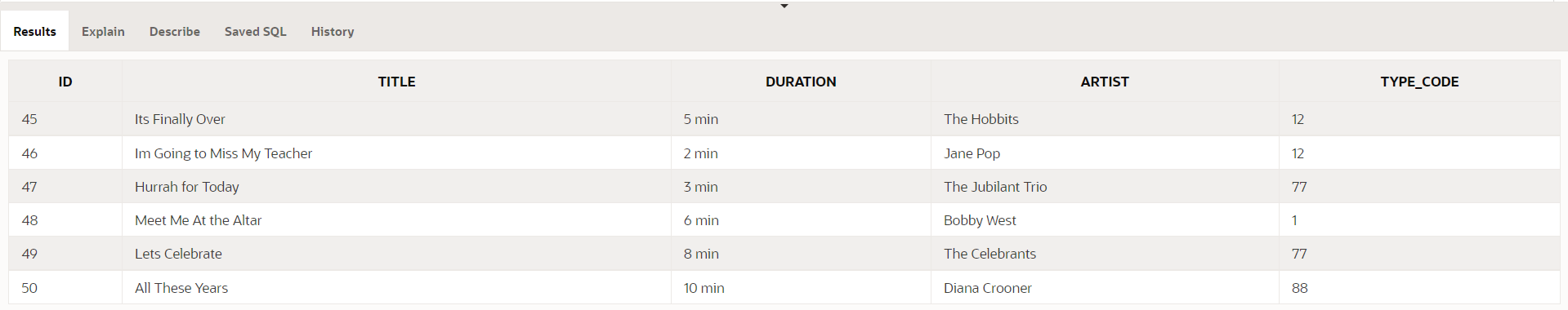
**DESCRIBE seq\_d\_songs;**



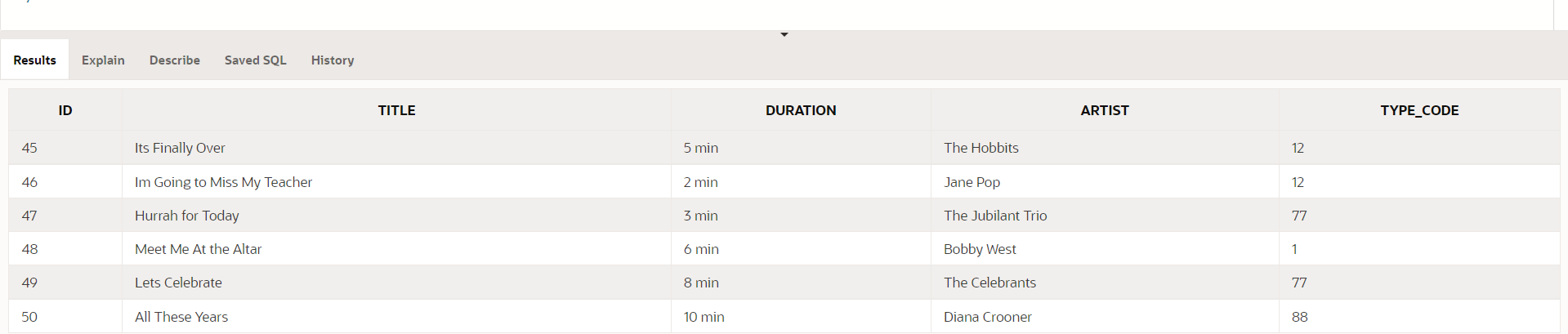
**DESCRIBE d\_songs;**



**SELECT \* FROM d\_songs;**



**SELECT \* FROM seq\_d\_songs;**



1. Because you are using copies of the original tables, the only constraints that were carried over were the NOT NULL constraints. Create a sequence to be used with the primary-key column of the seq\_d\_songs table. To avoid assigning primary-key numbers to these tables that already exist, the sequence should start at 100 and have a maximum value of 1000. Have your sequence increment by 2 and have NOCACHE and NOCYCLE. Name the sequence seq\_d\_songs\_seq.

**CREATE SEQUENCE seq\_d\_songs\_seq**

**INCREMENT BY 2**

**START WITH 100**

**MAXVALUE 1000**

**NOCYCLE**

**NOCACHE;**

*When I alter a sequence, a new increased MAXVALUE can be entered without changing the existing number order.*

1. Query the USER\_SEQUENCES data dictionary to verify the seq\_d\_songs\_seq SEQUENCE settings.

**SELECT \* FROM user\_sequences WHERE sequence\_name = UPPER('seq\_d\_songs\_seq');**

1. Insert two rows into the seq\_d\_songs table. Be sure to use the sequence that you created for the ID column. Add the two songs shown in the graphic.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | TITLE | DURATION | ARTIST | TYPE\_CODE |
|  | Island Fever | 5 min | Hawaiian Islanders | 12 |
|  | Castle of Dreams | 4 min | The Wanderers | 77 |

**INSERT INTO seq\_d\_songs (id,title,duration,artist,type\_code)**

**VALUES(seq\_d\_songs\_seq.NEXTVAL,'Surfing Summer',NULL,NULL,12);**

**INSERT INTO seq\_d\_songs (id,title,duration,artist,type\_code)**

**VALUES(seq\_d\_songs\_seq.NEXTVAL,'Victory Victory','5 min',NULL,12);**

SELECT \* FROM seq\_d\_songs ORDER BY id DESC;

SELECT \* FROM user\_sequences WHERE sequence\_name = UPPER('seq\_d\_songs\_seq');

1. Write out the syntax for seq\_d\_songs\_seq to view the current value for the sequence. Use the DUAL table. (Oracle Application Developer will not run this query.)

**SELECT seq\_d\_songs\_seq.CURRVAL FROM DUAL;**

1. What are three benefits of using SEQUENCEs?

**May be used to generate identity column values - the unique numbers. They also avoid concurrency issues.**

         **Save time and efforts of coding.**

         **Same sequence may be used in multiple tables since they are generated independent of the table using it, but I will prefer to use one sequence for one purpose.**

         **There is cache option available in sequences.**

         **These are sharable objects and multiple users can access it.**

1. What are the advantages of caching sequence values?

**Using cache option gives a slight performance advantage as the numbers are pre-allocated and stored in-memory.**

1. Name three reasons why gaps may occur in a sequence?

         **Rolling back a statement containing a sequence.**

         **System crash if system caches values in-memory**

         **Same sequence being used by multiple tables.**

Extension Exercise

1. Create a table called “students”. You can decide which columns belong in that table and what datatypes these columns require. (The students may create a table with different columns; however, the important piece that must be there is the student\_id column with a numeric datatype. This column length must allow the sequence to fit, e.g. a column length of 4 with a sequence that starts with 1 and goes to 10000000 will not work after student #9999 is entered.)

**CREATE TABLE students(**

**student\_id NUMBER(6) CONSTRAINT sdt\_sdt\_id\_pk PRIMARY KEY,**

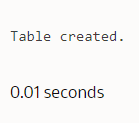
**fname VARCHAR2(12),**

**lname VARCHAR(20),**

**sex CHAR(1),**

**major VARCHAR2(24)**

**);**



1. Create a sequence called student\_id\_seq so that you can assign unique student\_id numbers for all students that you add to your table.

**CREATE SEQUENCE student\_id\_seq**

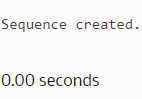
**INCREMENT BY 1**

**START WITH 1**

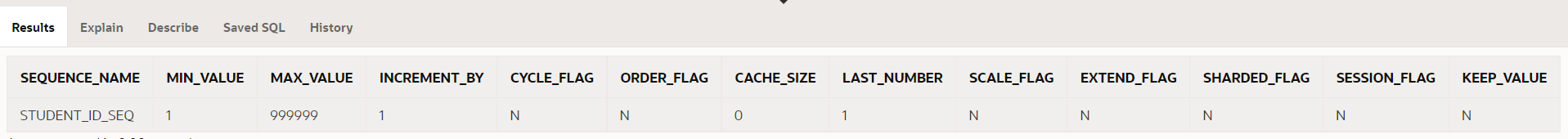
**MAXVALUE 999999**

**NOCYCLE**

**NOCACHE;**



SELECT \* FROM user\_sequences WHERE sequence\_name = UPPER('student\_id\_seq');



1. Now write the code to add students to your STUDENTS table, using your sequence “database object.”

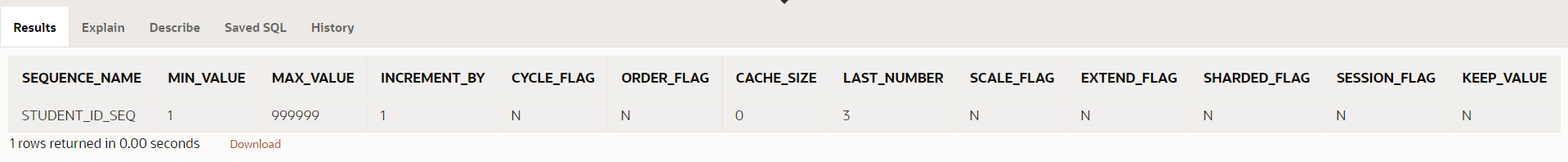
**INSERT INTO students(student\_id ,fname,lname ,sex ,major )**

**VALUES(student\_id\_seq.NEXTVAL,'Hemant','Kumar','M','Web');**

**INSERT INTO students(student\_id ,fname,lname ,sex ,major )**

**VALUES(student\_id\_seq.NEXTVAL,'Hemant2','Kumar2','M','CyberSec');**

SELECT \* FROM user\_sequences WHERE sequence\_name = UPPER('student\_id\_seq');



SELECT \* FROM students;

