

Lab 05- DDL

Objective:

The purpose of this lab is to introduce you to the DDL set of statements in SQL. By writing SQL to create tables, constraints, and views, you will have the tools needed to implement database designs that you will create later in the course. By finishing this lab, the student will be able to:

- create, modify, and drop tables based on design specifications provided,
- inserting new data into tables, update data in tables, and delete data from tables while considering referential integrity,
- enforce constraints on tables to ensure data integrity and consistency,
- create a table using the structure and data from an existing table,
- import data into a table from other tables.

Submission:

Your submission will be a single WORD file with the solutions provided.

Your submission needs to include a comment header block and be commented to include the question and the solutions. Make sure every SQL statement terminates with a semicolon.

Tasks:

Add

```
SET AUTOCOMMIT ON;
```

under the comment header and execute it

Consider the following table specifications:

Part A (DDL) :

1. Create all the following tables and their given constraints:

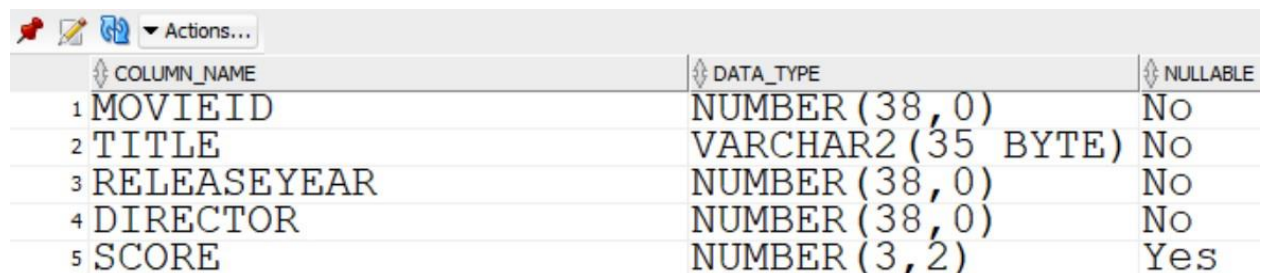
MOVIES (movieid:int, title:varchar(35), releaseYear:int, director:int, score:decimal(3,2))

Column Name	Column DataType	PK	Not Null	Unique	FK	Default Value	Validation
movieid	Int	✓					
title	varchar(35)		✓				
releaseYear	Int		✓				
director	Int		✓				
score	decimal(3,2)						< 10 and > 3

```

CREATE TABLE MOVIES (
movieid INT PRIMARY KEY,
title VARCHAR(35) NOT NULL,
releaseYear INT NOT NULL,
director INT NOT NULL,
score DECIMAL(3,2),
CONSTRAINT score CHECK (score BETWEEN 3 AND 10)
);

```



	COLUMN_NAME	DATA_TYPE	NULLABLE
1	MOVIEID	NUMBER (38, 0)	No
2	TITLE	VARCHAR2 (35 BYTE)	No
3	RELEASEYEAR	NUMBER (38, 0)	No
4	DIRECTOR	NUMBER (38, 0)	No
5	SCORE	NUMBER (3, 2)	Yes

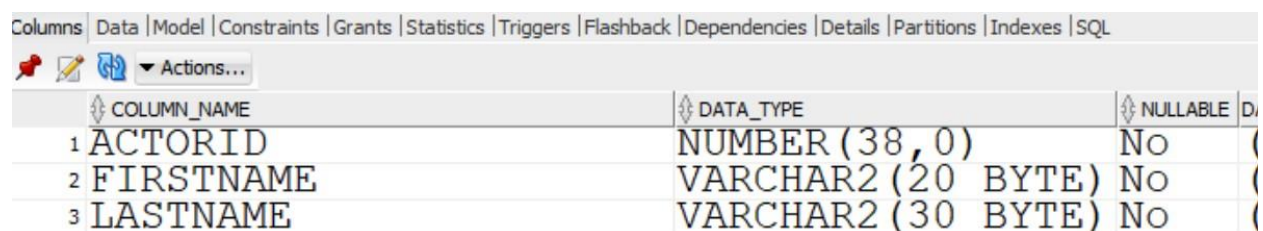
ACTORS (actorid:int, firstname:varchar(20), lastname:varchar(30))

Column Name	Column DataType	PK	Not Null	Unique	FK	Default Value	Validation
actorid	Int	✓					
firstName	varchar(20)		✓				
lastName	Varchar(30)		✓				

```

CREATE TABLE ACTORS (
actorid INT PRIMARY KEY,
firstName VARCHAR(20) NOT NULL,
lastName VARCHAR(30) NOT NULL
);

```



	COLUMN_NAME	DATA_TYPE	NULLABLE
1	ACTORID	NUMBER (38, 0)	No
2	FIRSTNAME	VARCHAR2 (20 BYTE)	No
3	LASTNAME	VARCHAR2 (30 BYTE)	No

CASTINGS (movieid:int, actorid:int)

Column Name	Column DataType	PK	Not Null	Unique	FK	Default Value	Validation
movieid	Int	✓			✓ (movies)		
actorid	int	✓			✓ (actors)		

CREATE TABLE CASTINGS (

movieid INT,

actorid INT,

CONSTRAINT keys_primaryfields PRIMARY

KEY(movieid,actorid),

ADD CONSTRAINT movies_fk FOREIGN KEY (movieid) REFERENCES MOVIES(movieid),

**ADD CONSTRAINT actor_fk FOREIGN KEY (actorid) REFERENCES ACTORS(actorid)
);**

COLUMN_NAME	DATA_TYPE
1 MOVIEID	NUMBER (38, 0)
2 ACTORID	NUMBER (38, 0)

DIRECTORS (directorid:int, firstname:varchar(20), lastname:varchar(30))

Column Name	Column DataType	PK	Not Null	Unique	FK	Default Value	Validation
directorid	Int	✓					
firstname	varchar(20)		✓				
lastname	varchar(30)		✓				




CREATE TABLE DIRECTORS (

directorid INT PRIMARY KEY,

firstName VARCHAR(20) NOT NULL,

lastName VARCHAR(30) NOT NULL

);

Columns	Data	Model	Constraints	Grants	Statistics	Triggers	Flashback	Dependencies	Details	Partitions	Indexes	SQL
  	Actions...											
	COLUMN_NAME					DATA_TYPE					NULLABLE	
1	DIRECTORID					NUMBER (38, 0)					No	
2	FIRSTNAME					VARCHAR2 (20 BYTE)					No	
3	LASTNAME					VARCHAR2 (30 BYTE)					No	

2. Modify the **movies** table to create a foreign key constraint that refers to table **directors**.

ALTER TABLE MOVIES

**ADD CONSTRAINT movies_director_fk FOREIGN KEY (director) REFERENCES
DIRECTORS(directorid);**

Table MOVIES altered.

3. Modify the *movies* table to create a new constraint so the uniqueness of the movie title is guaranteed.

ALTER TABLE MOVIES

ADD CONSTRAINT u_title UNIQUE (title);

4. Write insert statements to add the following data to table *directors* and *movies*.

Director

directorid	First name	Last name
1010	Rob	Minkoff
1020	Bill	Condon
1050	Josh	Cooley
2010	Brad	Bird
3020	Lake	Bell

INSERT ALL

```
INTO DIRECTORS VALUES(1010, 'Rob', 'Minkoff') INTO
DIRECTORS VALUES(1020, 'Bill', 'Condon') INTO
DIRECTORS VALUES(1050, 'Josh', 'Cooley') INTO
DIRECTORS VALUES(2010, 'Brad', 'Bird') INTO
DIRECTORS VALUES(3020, 'Lake', 'Bell') SELECT * FROM
DUAL;

COMMIT
```

Movies

Id	title	year	director	score
100	The Lion King	2019	3020	3.50
200	Beauty and the Beast	2017	1050	4.20
300	Toy Story 4	2019	1020	4.50
400	Mission Impossible	2018	2010	5.00
500	The Secret Life of Pets	2016	1010	3.90

```
INSERT INTO MOVIES (movieid, title, releaseyear, director, score) VALUES
(100, 'The Lion King', 2019, 3020, 3.50);
INSERT INTO MOVIES (movieid, title, releaseyear, director, score) VALUES
(200, 'Beauty and the Beast', 2017, 1050, 4.20);
INSERT INTO MOVIES (movieid, title, releaseyear, director, score) VALUES
(300, 'Toy Story 4', 2019, 1020, 4.50);
INSERT INTO MOVIES (movieid, title, releaseyear, director, score) VALUES
(400, 'Mission Impossible', 2018, 3020, 5.00);
INSERT INTO MOVIES (movieid, title, releaseyear, director, score) VALUES
(500, 'The Secret Life of Pets', 2016, 1010, 3.90);
```

```
commit;
```

5. Write SQL statements to remove all above tables.
Is the order of tables important when removing? Why?

```
DROP TABLE CASTINGS; DROP
TABLE MOVIES; DROP TABLE
ACTORS; DROP TABLE
DIRECTORS;
Table CASTINGS dropped.
Table MOVIES dropped. Table
ACTORS dropped.
```

Table DIRECTORS dropped.

Child tables must be dropped first before dropping parent table. You can't drop a parent table if you have a child table with a foreign key constraint in place, unless you specify the CASCADE CONSTRAINTS clause: DROP TABLE P CASCADE CONSTRAINTS; This command drops the FK constraint too. Deleting a table will necessarily drop all constraints related to this table.

DROP TABLE MOVIES CASCADE CONSTRAINTS; DROP
TABLE ACTORS CASCADE CONSTRAINTS;

DROP TABLE CASTINGS CASCADE CONSTRAINTS;

DROP TABLE DIRECTORS CASCADE CONSTRAINTS;

Table CASTINGS dropped.

Table MOVIES dropped. Table

ACTORS dropped. Table

DIRECTORS dropped.