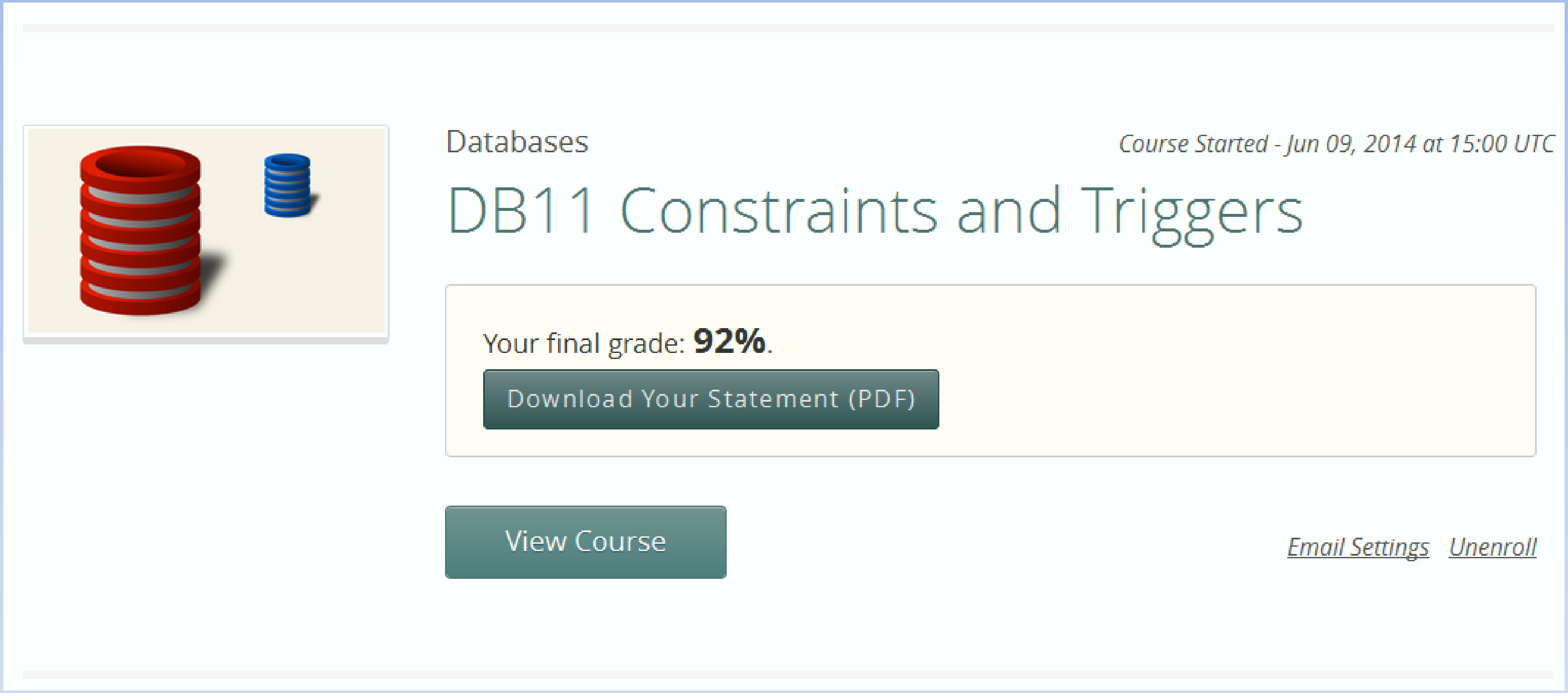
Database Systems: The Complete Book(3rd)
by Hector Garcia-Molina, Jeffrey D. Ullman & Jennifer Widom

Chapter 7: Constraints

Stanford Online: DB11



Databases

Course Started - Jun 09, 2014 at 15:00 UTC

DB11 Constraints and Triggers

Your final grade: **92%**.

Integrity Constraints

Impose restrictions on allowable data, beyond those imposed by structure and types

Referential integrity

= Integrity of references

= No “dangling pointers”

Simple Example Database

`Person`

SSN	Name	Birthdate
123		
235		
358		
459		

`BankAccount`

PersonID	AccountID	Amount
123	901	
123	902	
235	801	

Referential integrity from *BankAccount.PersonID* to *Person.SSN*

Each value in column ***PersonID*** of table ***BankAccount*** must appear in column ***SSN*** of table ***Person***

Referential integrity from *BankAccount.PersonID* to *Person.SSN*

Each value in column *PersonID* of table *BankAccount* must appear in column *SSN* of table *Person*

- *PersonID* is called the "foreign key"
- *SSN* is usually required to be the *primary key* for table *Person* or at least *unique*
- Multi-attribute foreign keys are allowed

Person

SSN	Name	Birthdate
123		
235		
358		
459		

BankAccount

PersonID	AccountID	Amount
123	901	
123	902	
235	801	

Referential Integrity Enforcement (*BankAccount.PersonID* to *Person.SSN*)

Potentially violating modifications:

- Insert into BankAccount
- Delete from Person
- Update BankAccount.PersonID
- Update Person.SSN

`Person`

SSN	Name	Birthdate
123		
235		
358		
459		

`BankAccount`

PersonID	AccountID	Amount
123	901	
123	902	
235	801	

Foreign Keys in SQL

```
References <table> (<attribute>)
```

Person

PersonID	Name	Birthdate
123		
235		
358		
459		

BankAccount

PersonID	AccountID	Amount
123	901	
123	902	
235	801	

Foreign Keys in SQL

```
References <table> (<attribute>)
```

```
create table BankAccount(
PersonID char(3) REFERENCES Person(SSN),
AccountID int, Amount int);
```

Foreign Keys in SQL

```
FOREIGN KEY (<attributes>)
REFERENCES <table> (<attributes>)
```

```
create table BankAccount(PersonID char(3),
  AccountID int, Amount int,
FOREIGN KEY (PersonID)
REFERENCES Person(SSN)
);
```

Naming Constraints in SQL

```
Constraint <name>
FOREIGN KEY (<attributes)
REFERENCES <table> (<attributes)
```

```
create table BankAccount(PersonID char(3),
  AccountID int, Amount int,
CONSTRAINT fk_PersonID PersonID
FOREIGN KEY (PersonID)
REFERENCES Person(SSN)
);
```

Maintaining Referential Integrity

- On Deletion/On Update
  - Restrict
    - Stop the action w/ Error
  - Set null
    - When reference value for Foreign Key Changes, set value null.
  - Cascade
    - When reference value for Foreign Key Changes, set value to new value.

Referential Integrity Enforcement (*BankAccount.PersonID* to *Person.SSN*)

Potentially violating modifications:

- Insert into BankAccount
- Delete from Person
- Update BankAccount.PersonID
- Update Person.SSN

Person

SSN	Name	Birthdate
123		
235		
358		
459		

BankAccount

PersonID	AccountID	Amount
123	901	
123	902	
235	801	

Referential Integrity Enforcement (*BankAccount.PersonID* to *Person.SSN*)

Special actions:

- Insert into BankAccount.PersonID
- Update BankAccount.PersonID

  `Restrict` (default), `Set Null`, `Cascade`

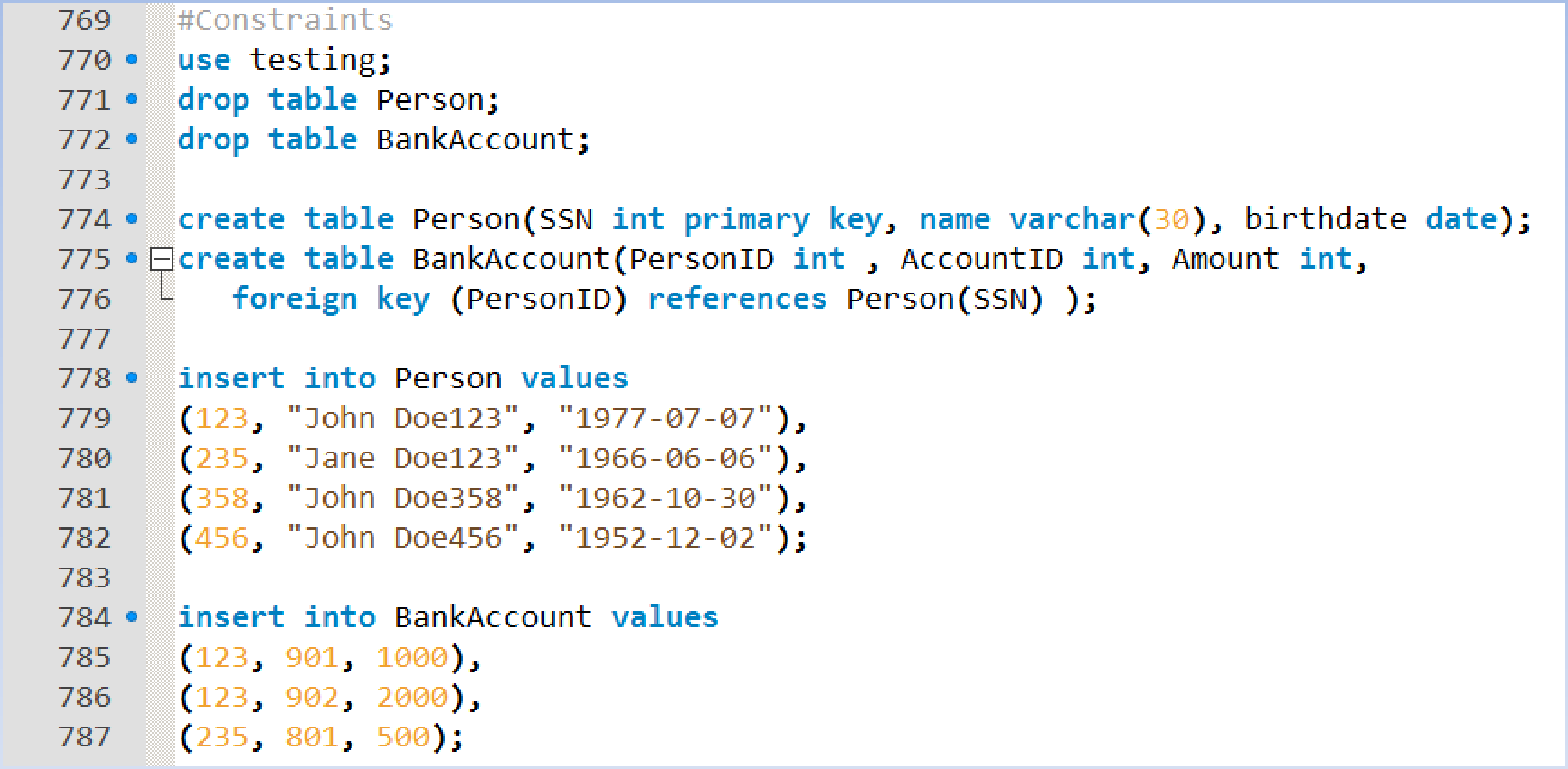
Person

SSN	Name	Birthdate
123		
235		
358		
459		

BankAccount

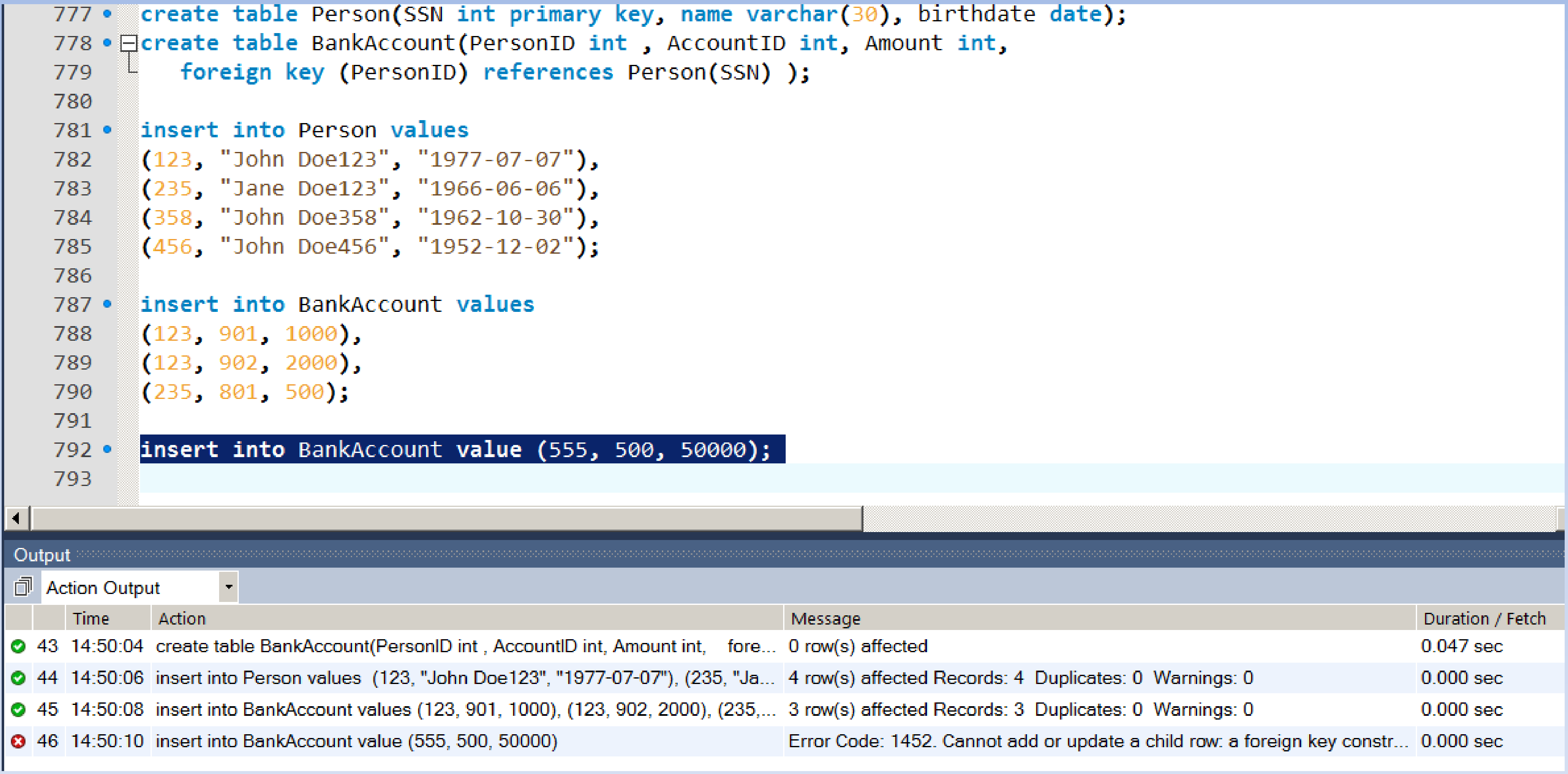
PersonID	AccountID	Amount
123	901	
123	902	
235	801	

Insert into BankAccount.PersonID



```
#Constraints
use testing;
drop table Person;
drop table BankAccount;

create table Person(SSN int primary key, name varchar(30), birthdate date);
create table BankAccount(PersonID int , AccountID int, Amount int,
   foreign key (PersonID) references Person(SSN) );

insert into Person values
(123, "John Doe123", "1977-07-07"),
(235, "Jane Doe123", "1966-06-06"),
(358, "John Doe358", "1962-10-30"),
(456, "John Doe456", "1952-12-02");

insert into BankAccount values
(123, 901, 1000),
(123, 902, 2000),
(235, 801, 500);
```

Insert into BankAccount.PersonID RESTRICTED



```
create table Person(SSN int primary key, name varchar(30), birthdate date);
create table BankAccount(PersonID int , AccountID int, Amount int,
   foreign key (PersonID) references Person(SSN) );

insert into Person values
(123, "John Doe123", "1977-07-07"),
(235, "Jane Doe123", "1966-06-06"),
(358, "John Doe358", "1962-10-30"),
(456, "John Doe456", "1952-12-02");

insert into BankAccount values
(123, 901, 1000),
(123, 902, 2000),
(235, 801, 500);

insert into BankAccount value (555, 500, 50000);
```

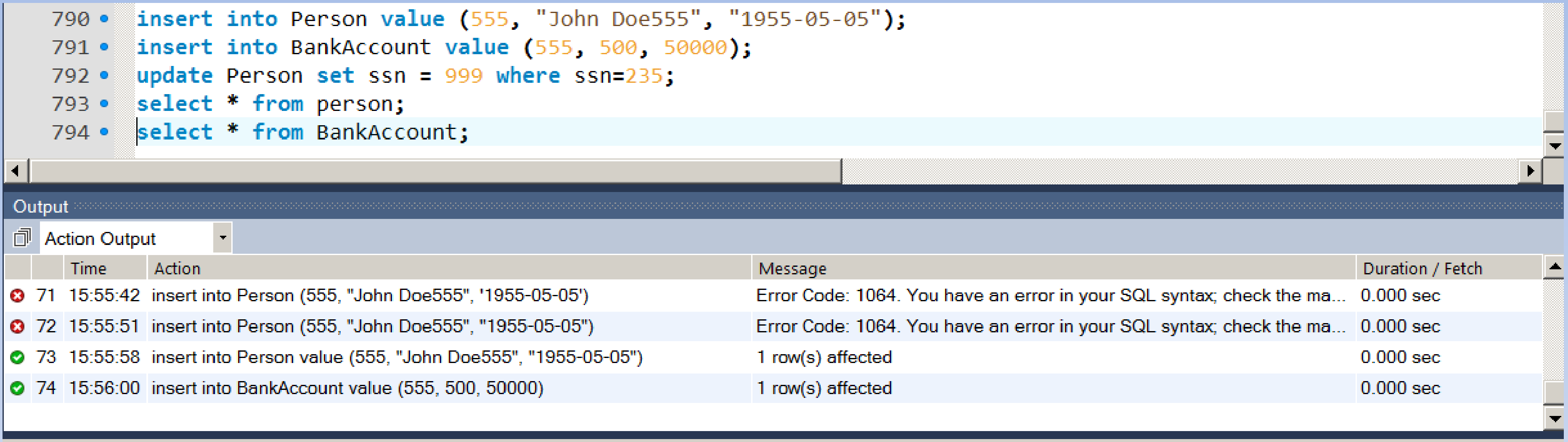
Output

Action Output

		Time	Action	Message	Duration / Fetch
✓	43	14:50:04	create table BankAccount(PersonID int , AccountID int, Amount int, fore...	0 row(s) affected	0.047 sec
✓	44	14:50:06	insert into Person values (123, "John Doe123", "1977-07-07"), (235, "Ja...	4 row(s) affected Records: 4 Duplicates: 0 Warnings: 0	0.000 sec
✓	45	14:50:08	insert into BankAccount values (123, 901, 1000), (123, 902, 2000), (235,...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.000 sec
✗	46	14:50:10	insert into BankAccount value (555, 500, 50000)	Error Code: 1452. Cannot add or update a child row: a foreign key constr...	0.000 sec

Insert into BankAccount.PersonID

- First insert person into Person Table

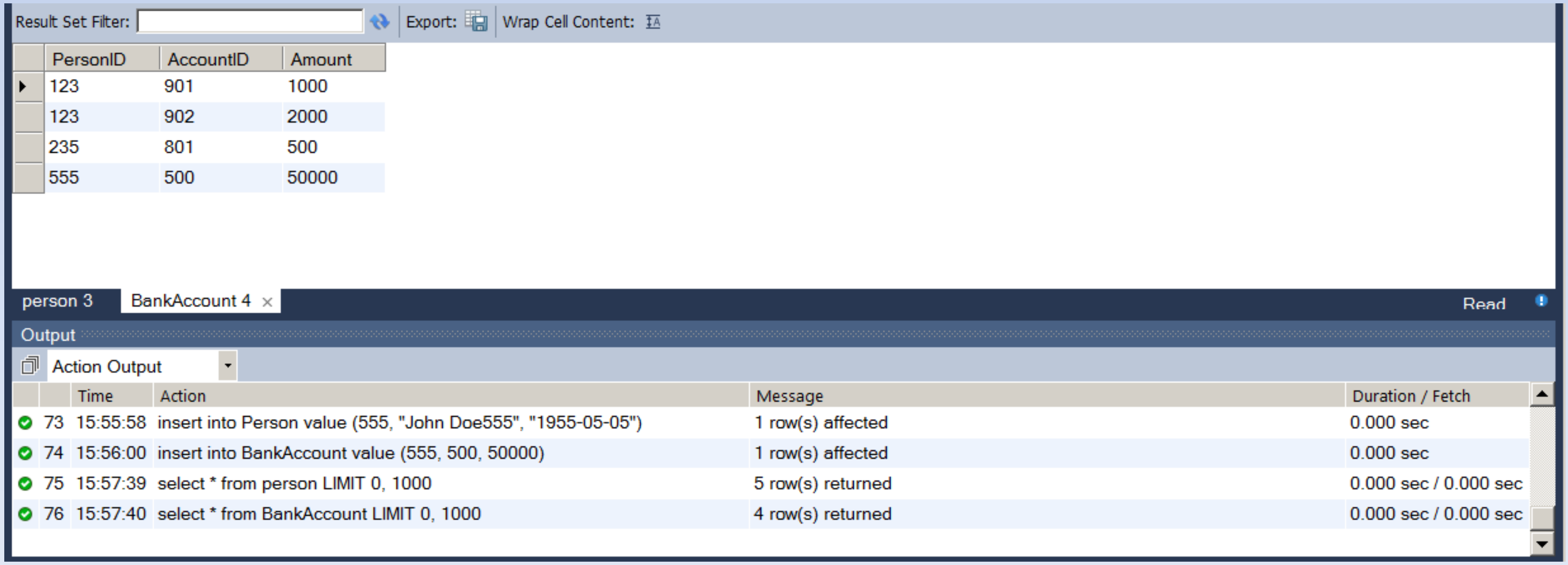


```
insert into Person value (555, "John Doe555", "1955-05-05");
insert into BankAccount value (555, 500, 50000);
update Person set ssn = 999 where ssn=235;
select * from person;
select * from BankAccount;
```

Output

Action Output

		Time	Action	Message	Duration / Fetch
✗	71	15:55:42	insert into Person (555, "John Doe555", '1955-05-05')	Error Code: 1064. You have an error in your SQL syntax; check the ma...	0.000 sec
✗	72	15:55:51	insert into Person (555, "John Doe555", "1955-05-05")	Error Code: 1064. You have an error in your SQL syntax; check the ma...	0.000 sec
✓	73	15:55:58	insert into Person value (555, "John Doe555", "1955-05-05")	1 row(s) affected	0.000 sec
✓	74	15:56:00	insert into BankAccount value (555, 500, 50000)	1 row(s) affected	0.000 sec



Result Set Filter: Export: Wrap Cell Content:

PersonID	AccountID	Amount
123	901	1000
123	902	2000
235	801	500
555	500	50000

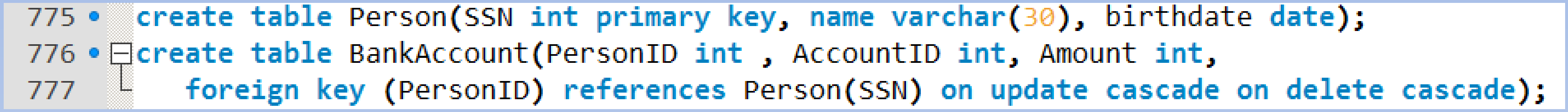
person 3 BankAccount 4 × Read

Output

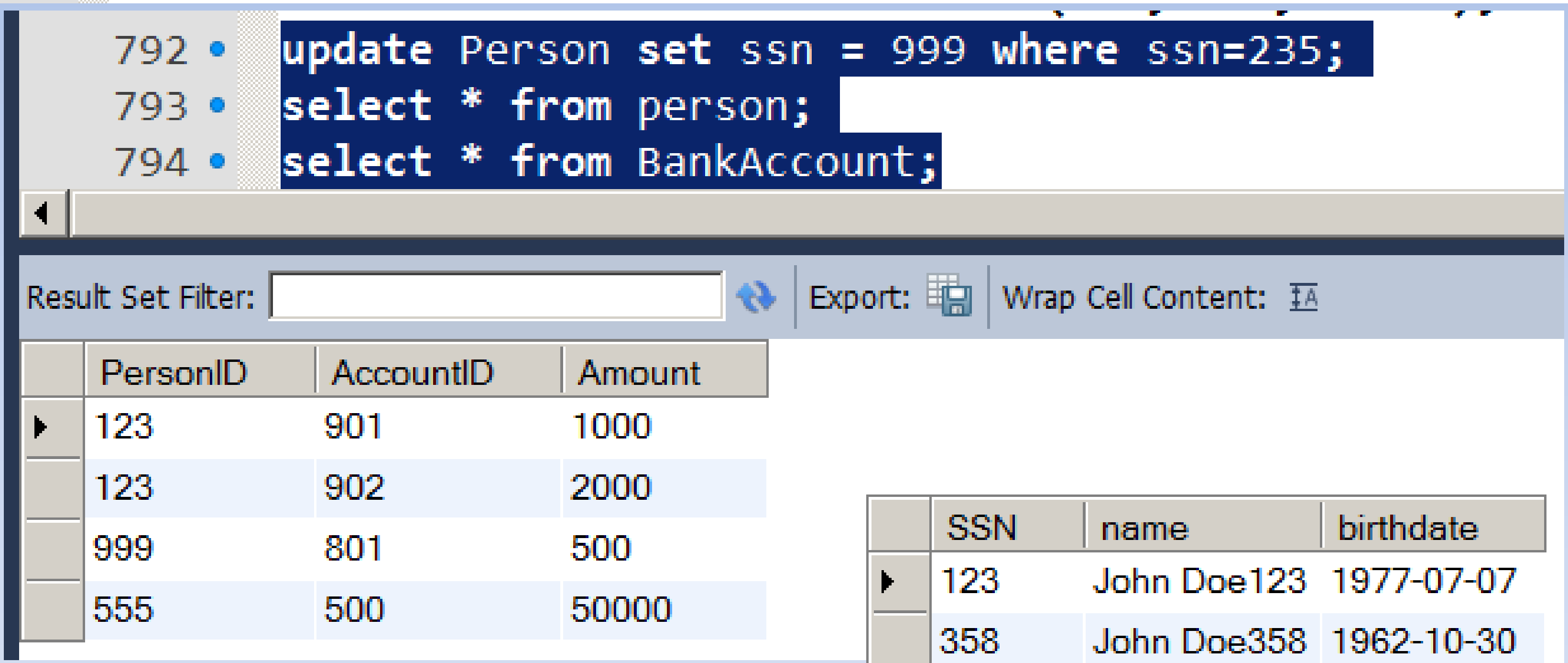
Action Output

		Time	Action	Message	Duration / Fetch
✓	73	15:55:58	insert into Person value (555, "John Doe555", "1955-05-05")	1 row(s) affected	0.000 sec
✓	74	15:56:00	insert into BankAccount value (555, 500, 50000)	1 row(s) affected	0.000 sec
✓	75	15:57:39	select * from person LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
✓	76	15:57:40	select * from BankAccount LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec

Update BankAccount.PersonID CASCADE

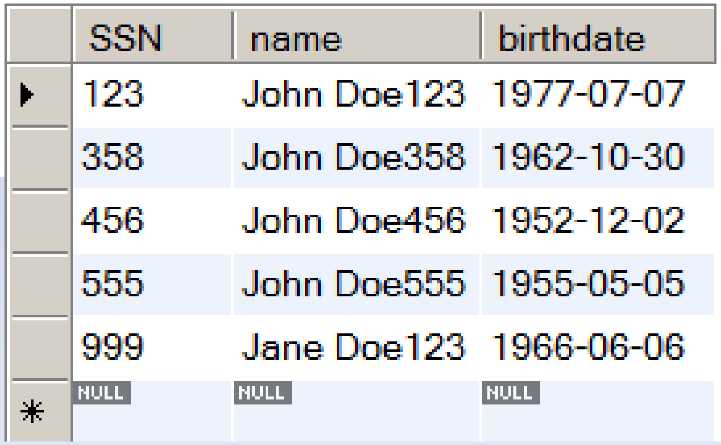


```sql
create table Person(SSN int primary key, name varchar(30), birthdate date);
create table BankAccount(PersonID int , AccountID int, Amount int,
   foreign key (PersonID) references Person(SSN) on update cascade on delete cascade);
```



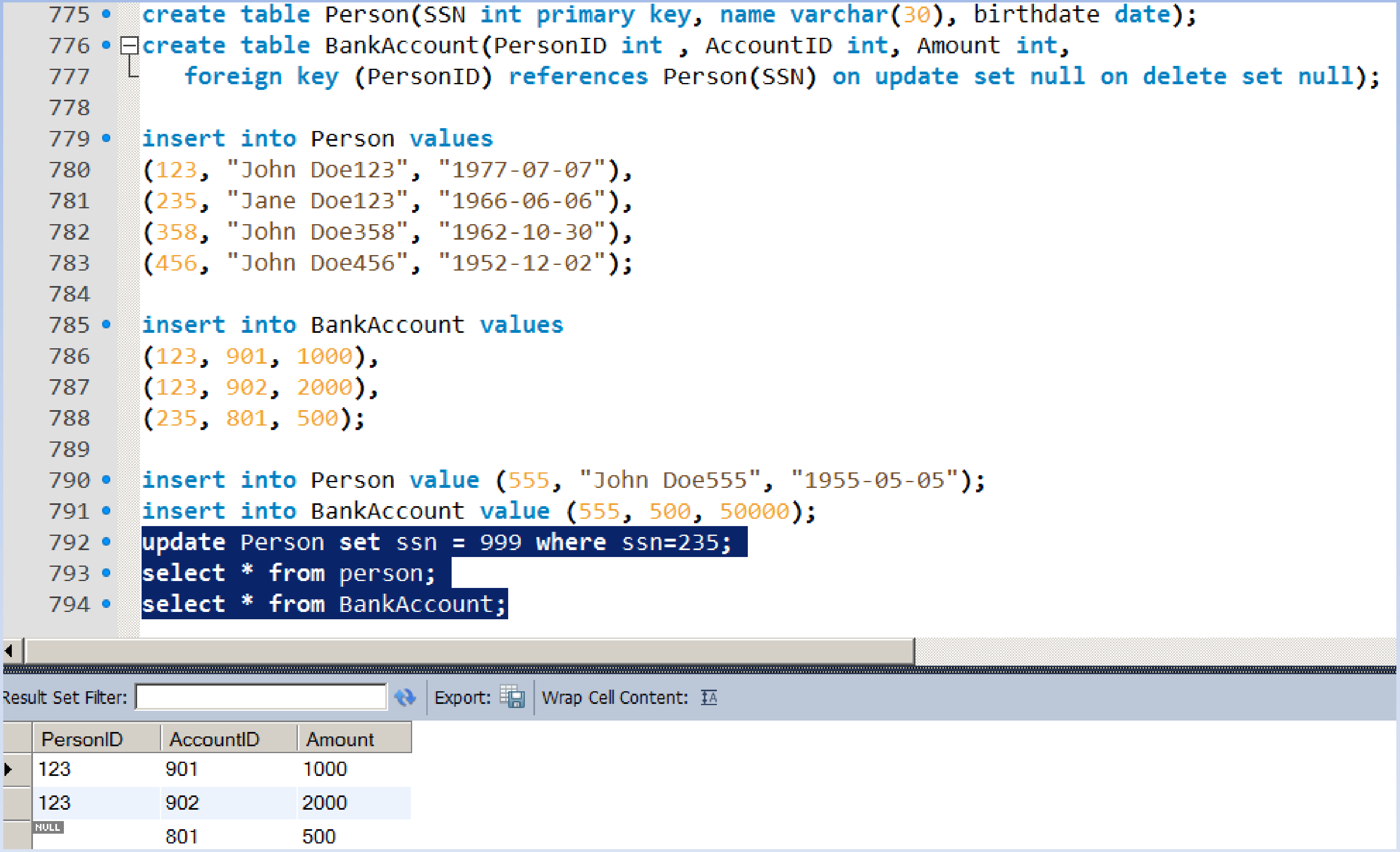
```sql
update Person set ssn = 999 where ssn=235;
select * from person;
select * from BankAccount;
```

	PersonID	AccountID	Amount
▶	123	901	1000
	123	902	2000
	999	801	500
	555	500	50000



	SSN	name	birthdate
▶	123	John Doe123	1977-07-07
	358	John Doe358	1962-10-30
	456	John Doe456	1952-12-02
	555	John Doe555	1955-05-05
	999	Jane Doe123	1966-06-06
*	NULL	NULL	NULL

Update BankAccount.PersonID Set Null



```
create table Person(SSN int primary key, name varchar(30), birthdate date);
create table BankAccount(PersonID int , AccountID int, Amount int,
    foreign key (PersonID) references Person(SSN) on update set null on delete set null);

insert into Person values
(123, "John Doe123", "1977-07-07"),
(235, "Jane Doe123", "1966-06-06"),
(358, "John Doe358", "1962-10-30"),
(456, "John Doe456", "1952-12-02");

insert into BankAccount values
(123, 901, 1000),
(123, 902, 2000),
(235, 801, 500);

insert into Person value (555, "John Doe555", "1955-05-05");
insert into BankAccount value (555, 500, 50000);
update Person set ssn = 999 where ssn=235;
select * from person;
select * from BankAccount;
```

Result Set Filter: Export: Wrap Cell Content:

PersonID	AccountID	Amount
123	901	1000
123	902	2000
NULL	801	500

Referential Integrity Enforcement (*BankAccount.PersonID* to *Person.SSN*)

Special actions:

- Delete from Person.SSN

  `Restrict` (default), `Set Null`, `Cascade`

- Update Person.SSN

  `Restrict` (default), `Set Null`, `Cascade`

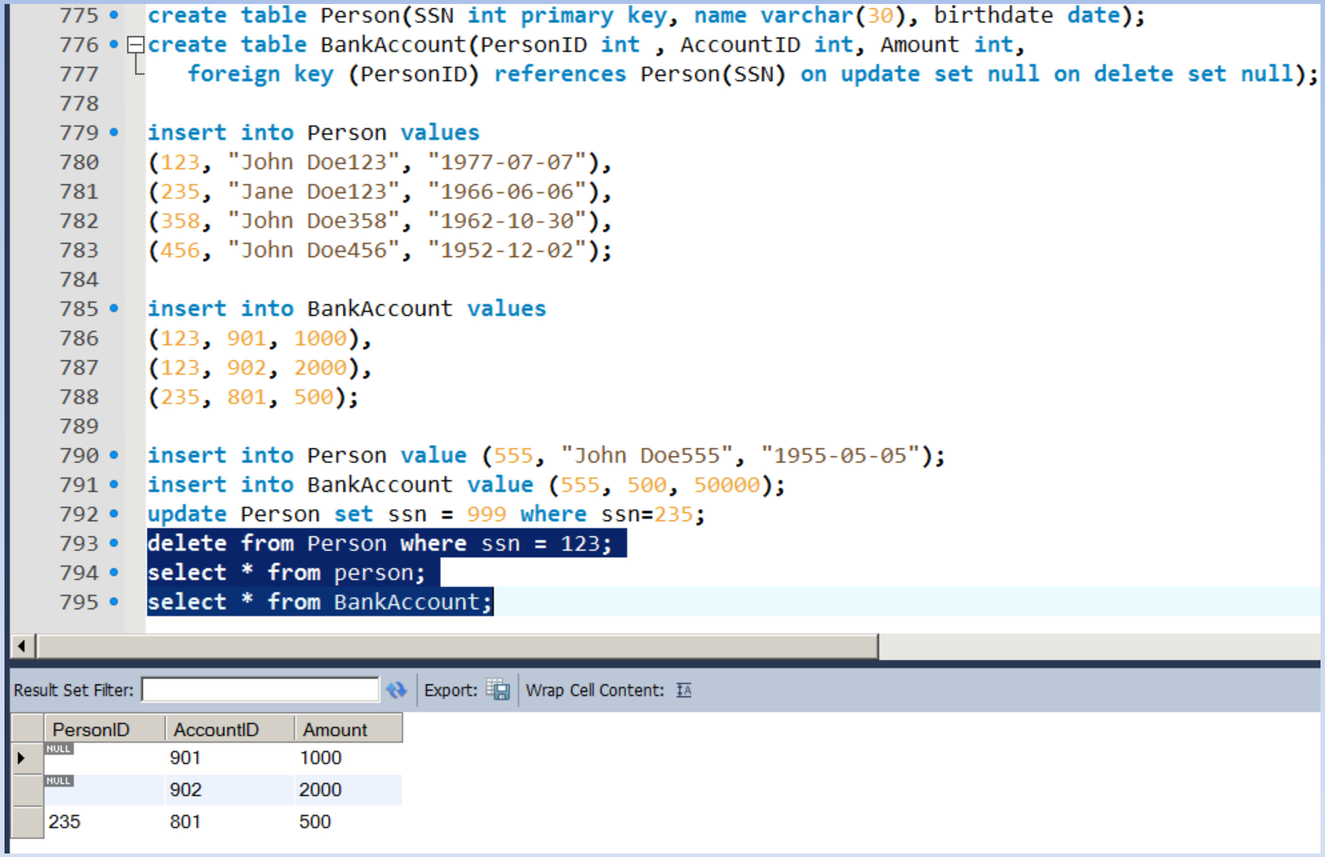
`Person`

SSN	Name	Birthdate
123		
235		
358		
459		

`BankAccount`

PersonID	AccountID	Amount
123	901	
123	902	
235	801	

Delete from Person where SSN=123;
Set Null

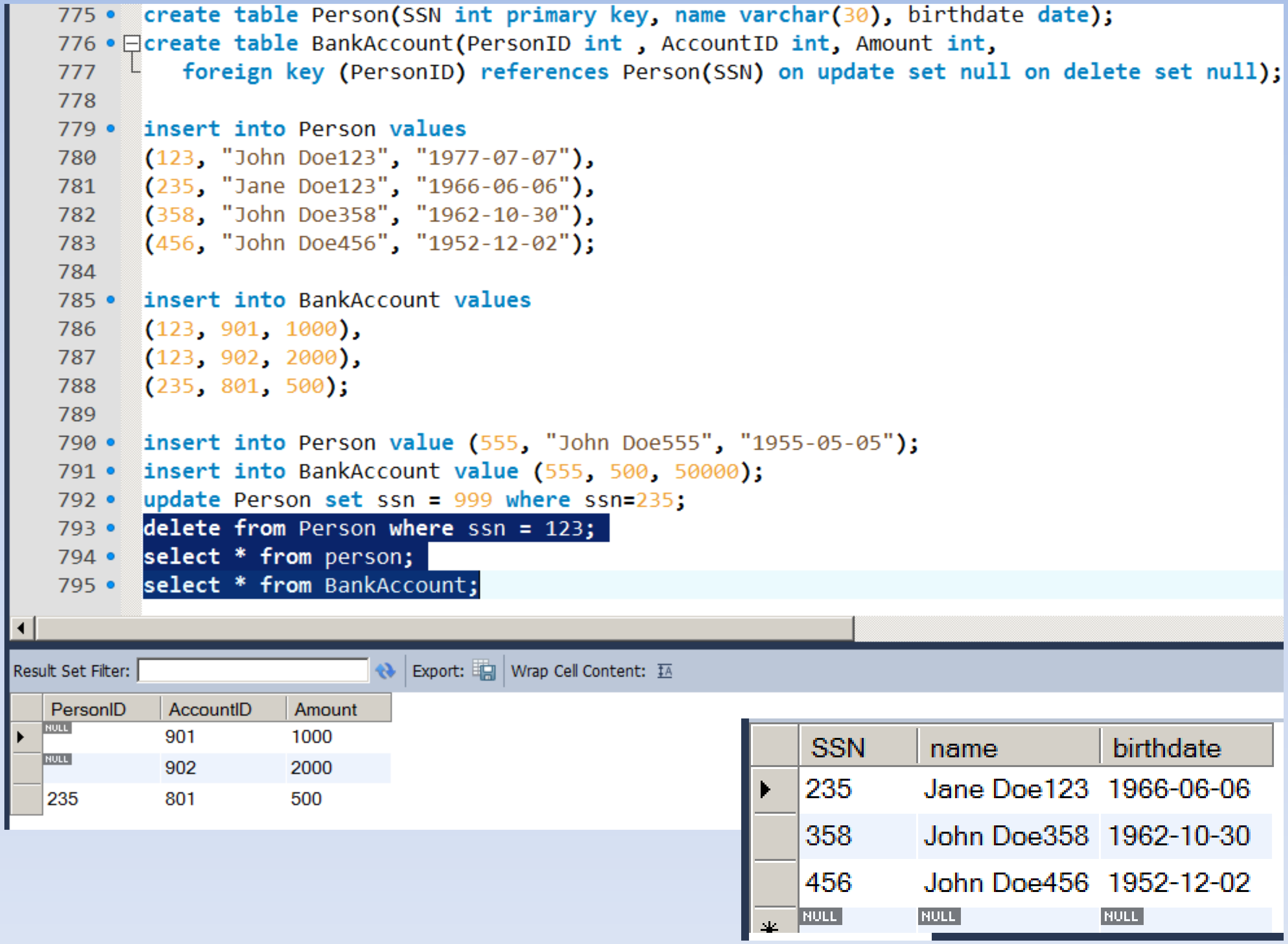


```
create table Person(SSN int primary key, name varchar(30), birthdate date);
create table BankAccount(PersonID int , AccountID int, Amount int,
   foreign key (PersonID) references Person(SSN) on update set null on delete set null);

insert into Person values
(123, "John Doe123", "1977-07-07"),
(235, "Jane Doe123", "1966-06-06"),
(358, "John Doe358", "1962-10-30"),
(456, "John Doe456", "1952-12-02");

insert into BankAccount values
(123, 901, 1000),
(123, 902, 2000),
(235, 801, 500);

insert into Person value (555, "John Doe555", "1955-05-05");
insert into BankAccount value (555, 500, 50000);
update Person set ssn = 999 where ssn=235;
delete from Person where ssn = 123;
select * from person;
select * from BankAccount;
```

Result Set Filter: Export: Wrap Cell Content:

PersonID	AccountID	Amount
NULL	901	1000
NULL	902	2000
235	801	500

Delete from Person where SSN=123;
Set Null



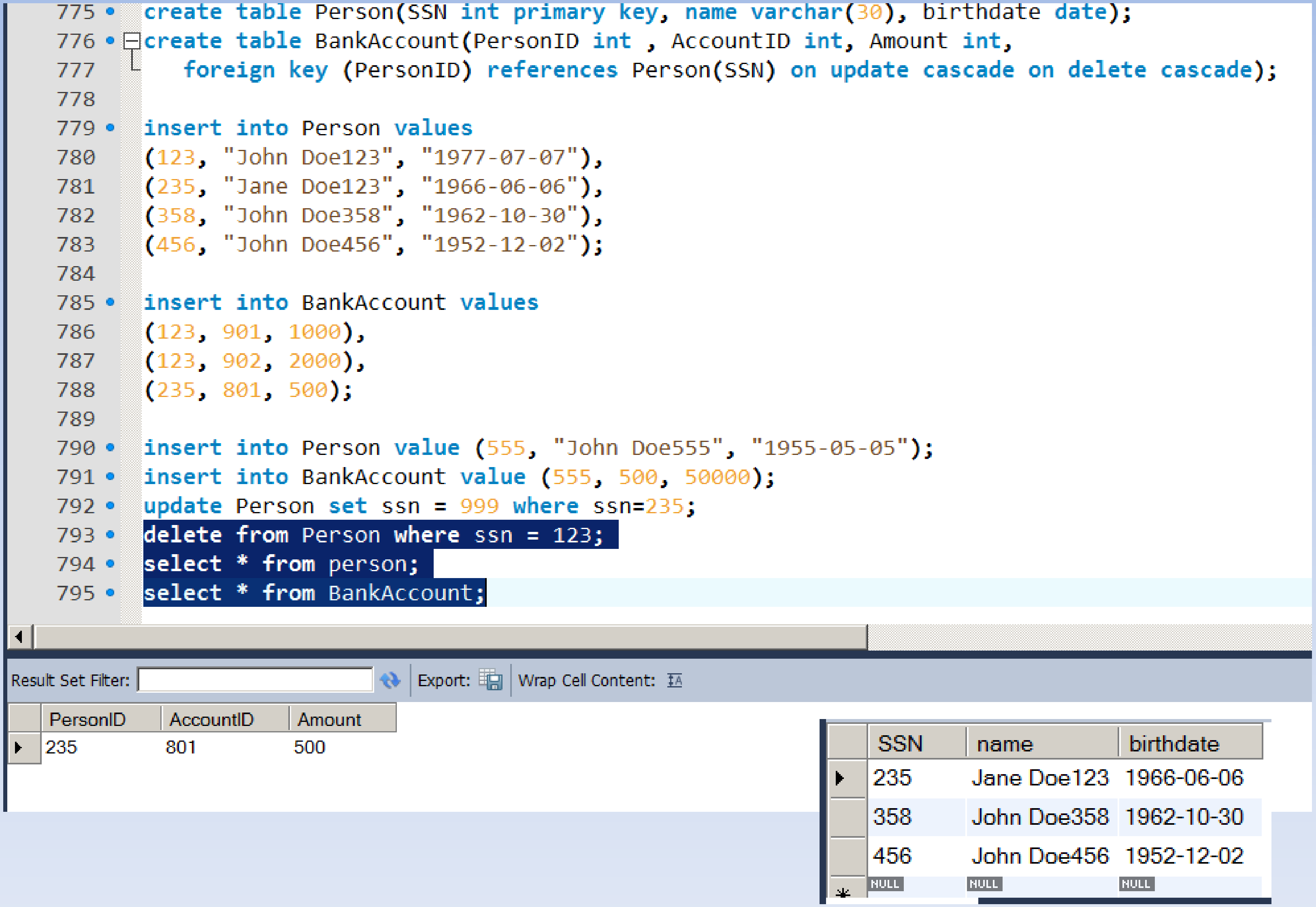
```
create table Person(SSN int primary key, name varchar(30), birthdate date);
create table BankAccount(PersonID int , AccountID int, Amount int,
   foreign key (PersonID) references Person(SSN) on update set null on delete set null);

insert into Person values
(123, "John Doe123", "1977-07-07"),
(235, "Jane Doe123", "1966-06-06"),
(358, "John Doe358", "1962-10-30"),
(456, "John Doe456", "1952-12-02");

insert into BankAccount values
(123, 901, 1000),
(123, 902, 2000),
(235, 801, 500);

insert into Person value (555, "John Doe555", "1955-05-05");
insert into BankAccount value (555, 500, 50000);
update Person set ssn = 999 where ssn=235;
delete from Person where ssn = 123;
select * from person;
select * from BankAccount;
```

Result Set Filter: Export: Wrap Cell Content:

PersonID	AccountID	Amount
NULL	901	1000
NULL	902	2000
235	801	500

SSN	name	birthdate
235	Jane Doe123	1966-06-06
358	John Doe358	1962-10-30
456	John Doe456	1952-12-02
NULL	NULL	NULL

Delete from Person where SSN=123; Cascade



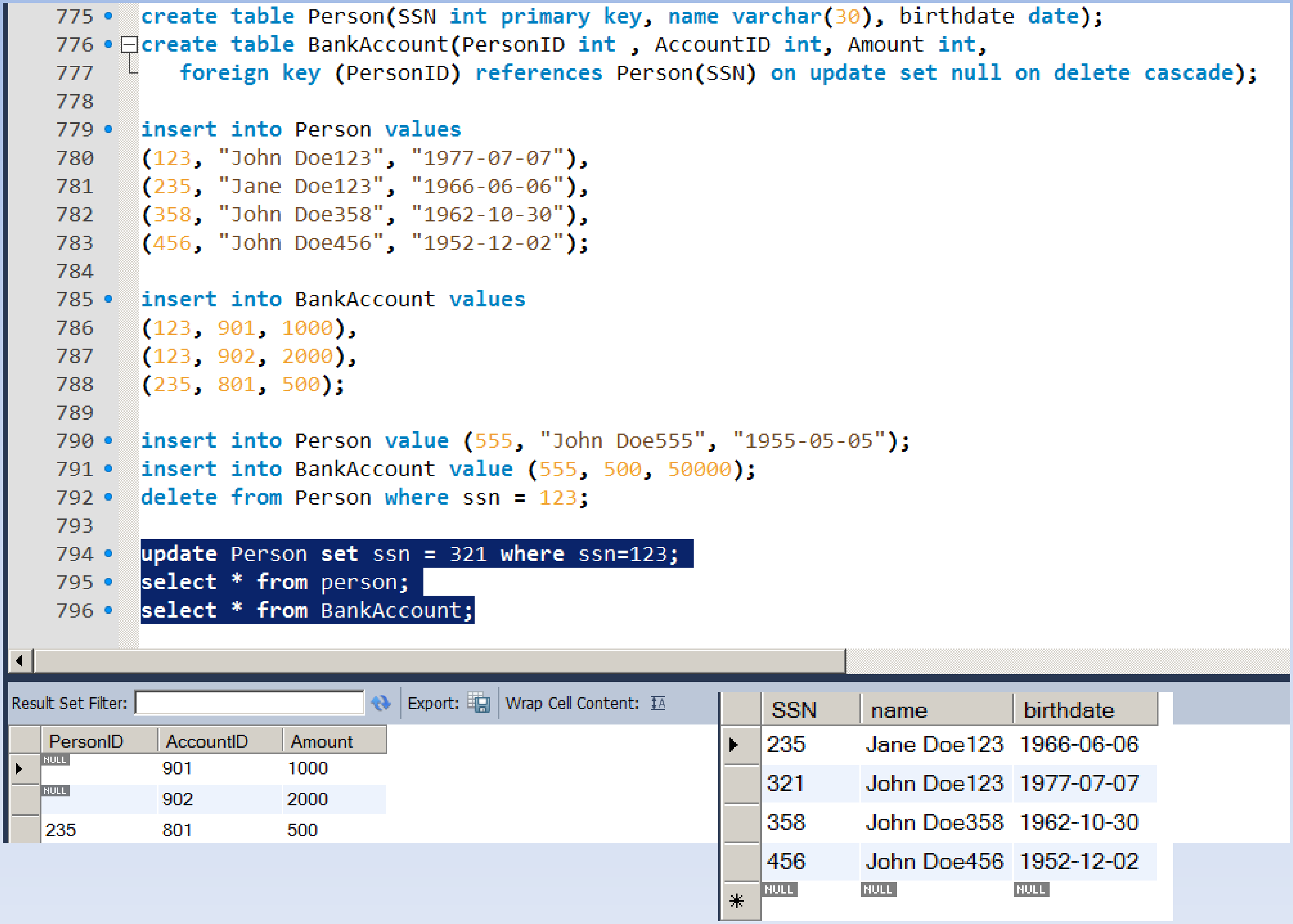
```
create table Person(SSN int primary key, name varchar(30), birthdate date);
create table BankAccount(PersonID int , AccountID int, Amount int,
   foreign key (PersonID) references Person(SSN) on update cascade on delete cascade);

insert into Person values
(123, "John Doe123", "1977-07-07"),
(235, "Jane Doe123", "1966-06-06"),
(358, "John Doe358", "1962-10-30"),
(456, "John Doe456", "1952-12-02");

insert into BankAccount values
(123, 901, 1000),
(123, 902, 2000),
(235, 801, 500);

insert into Person value (555, "John Doe555", "1955-05-05");
insert into BankAccount value (555, 500, 50000);
update Person set ssn = 999 where ssn=235;
delete from Person where ssn = 123;
select * from person;
select * from BankAccount;
```

Result Set Filter: Export: Wrap Cell Content:

PersonID	AccountID	Amount
235	801	500

SSN	name	birthdate
235	Jane Doe123	1966-06-06
358	John Doe358	1962-10-30
456	John Doe456	1952-12-02
NULL	NULL	NULL

Update Person set SSN=321 where SSN=123; Set Null



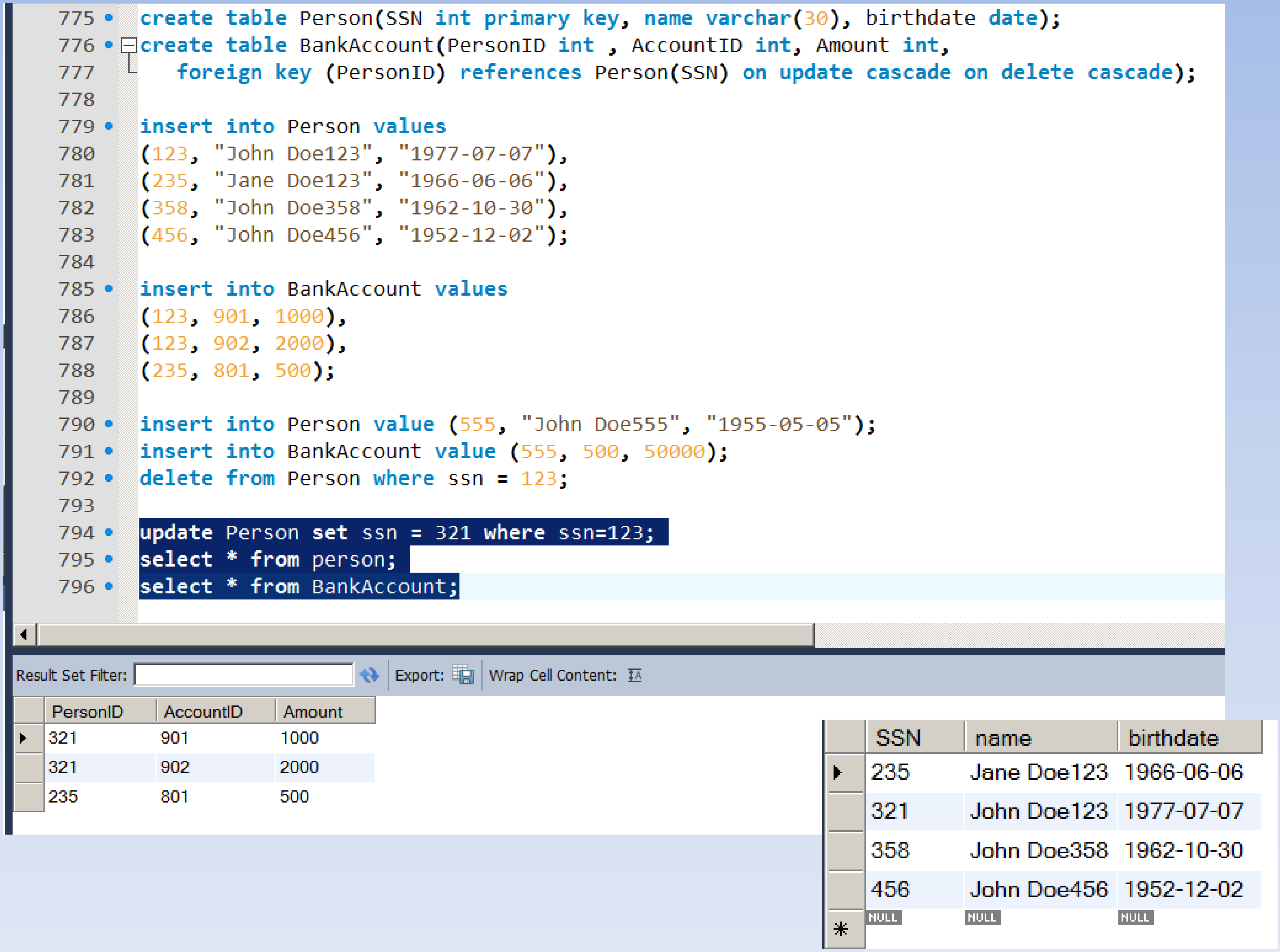
```
create table Person(SSN int primary key, name varchar(30), birthdate date);
create table BankAccount(PersonID int , AccountID int, Amount int,
   foreign key (PersonID) references Person(SSN) on update set null on delete cascade);

insert into Person values
(123, "John Doe123", "1977-07-07"),
(235, "Jane Doe123", "1966-06-06"),
(358, "John Doe358", "1962-10-30"),
(456, "John Doe456", "1952-12-02");

insert into BankAccount values
(123, 901, 1000),
(123, 902, 2000),
(235, 801, 500);

insert into Person value (555, "John Doe555", "1955-05-05");
insert into BankAccount value (555, 500, 50000);
delete from Person where ssn = 123;

update Person set ssn = 321 where ssn=123;
select * from person;
select * from BankAccount;
```

Result Set Filter: Export: Wrap Cell Content:

PersonID	AccountID	Amount
NULL	901	1000
NULL	902	2000
235	801	500

SSN	name	birthdate
235	Jane Doe123	1966-06-06
321	John Doe123	1977-07-07
358	John Doe358	1962-10-30
456	John Doe456	1952-12-02
NULL	NULL	NULL

Update Person set SSN=321 where SSN=123; Cascade

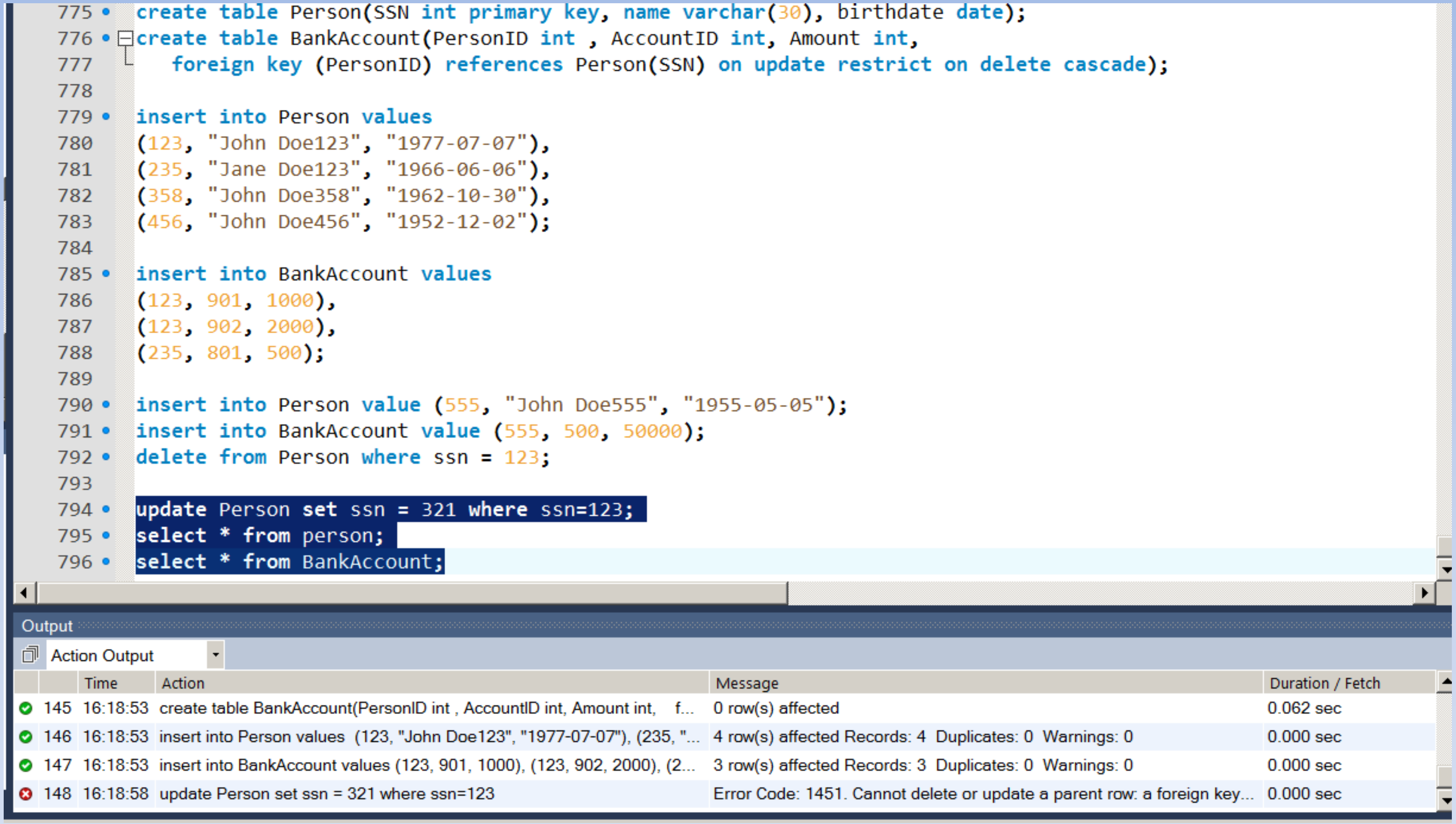


```
create table Person(SSN int primary key, name varchar(30), birthdate date);
create table BankAccount(PersonID int , AccountID int, Amount int,
    foreign key (PersonID) references Person(SSN) on update cascade on delete cascade);

insert into Person values
(123, "John Doe123", "1977-07-07"),
(235, "Jane Doe123", "1966-06-06"),
(358, "John Doe358", "1962-10-30"),
(456, "John Doe456", "1952-12-02");

insert into BankAccount values
(123, 901, 1000),
(123, 902, 2000),
(235, 801, 500);

insert into Person value (555, "John Doe555", "1955-05-05");
insert into BankAccount value (555, 500, 50000);
delete from Person where ssn = 123;

update Person set ssn = 321 where ssn=123;
select * from person;
select * from BankAccount;
```

PersonID	AccountID	Amount
321	901	1000
321	902	2000
235	801	500

SSN	name	birthdate
235	Jane Doe123	1966-06-06
321	John Doe123	1977-07-07
358	John Doe358	1962-10-30
456	John Doe456	1952-12-02
NULL	NULL	NULL

Update Person set SSN=321 where SSN=123; w/ Restrict



```sql
create table Person(SSN int primary key, name varchar(30), birthdate date);
create table BankAccount(PersonID int , AccountID int, Amount int,
   foreign key (PersonID) references Person(SSN) on update restrict on delete cascade);

insert into Person values
(123, "John Doe123", "1977-07-07"),
(235, "Jane Doe123", "1966-06-06"),
(358, "John Doe358", "1962-10-30"),
(456, "John Doe456", "1952-12-02");

insert into BankAccount values
(123, 901, 1000),
(123, 902, 2000),
(235, 801, 500);

insert into Person value (555, "John Doe555", "1955-05-05");
insert into BankAccount value (555, 500, 50000);
delete from Person where ssn = 123;

update Person set ssn = 321 where ssn=123;
select * from person;
select * from BankAccount;
```

Output

Action Output

	Time	Action	Message	Duration / Fetch
145	16:18:53	create table BankAccount(PersonID int , AccountID int, Amount int, f...	0 row(s) affected	0.062 sec
146	16:18:53	insert into Person values (123, "John Doe123", "1977-07-07"), (235, "...	4 row(s) affected Records: 4 Duplicates: 0 Warnings: 0	0.000 sec
147	16:18:53	insert into BankAccount values (123, 901, 1000), (123, 902, 2000), (2...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.000 sec
148	16:18:58	update Person set ssn = 321 where ssn=123	Error Code: 1451. Cannot delete or update a parent row: a foreign key...	0.000 sec

Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (`testing`.`bankaccount`, CONSTRAINT `bankaccount_ibfk_1` FOREIGN KEY (`PersonID`) REFERENCES `person` (`SSN`) ON DELETE CASCADE)

Maintaining Referential Integrity in SQL

```
FOREIGN KEY (<attributes)
REFERENCES <table> (<attributes)
[On Delete|On Update] [Set Null|Cascade]
```

```
create table BankAccount(PersonID char(3),
  AccountID int, Amount int,
FOREIGN KEY (PersonID)
REFERENCES Person(PersonID)
ON DELETE Set Null
ON INSERT Cascade
);
```

Modifying Constraints in SQL

```
ALTER TABLE <name>
[Add | Drop] Constraint <name>
```

```
ALTER TABLE BankAccount
DROP CONSTRAINT fk_PersonID
```

DEFERRED CONSTRAINT CHECKING:

Referential integrity from *R.A* to *S.B*

Each value in column *A* of table *R* must appear in column *B* of table *S*

- *Constraint checking delayed until end of transaction*
- *REMEMBER: Proof for constraints required trans end check*

Person

PersonID	Name	Birthdate
123		
235		
358		
459		

BankAccount

PersonID	AccountID	Amount
123	901	
123	902	
235	801	

Deferred Constraint Checking in SQL (Not currently supported with InnoDB)

```
FOREIGN KEY (<attributes)
REFERENCES <table> (<attributes)
DEFERRABLE INITIALLY [Deferred | Immediate]
```

```
create table BankAccount(PersonID char(3),
  AccountID int, Amount int,
FOREIGN KEY (PersonID)
REFERENCES Person(PersonID)
DEFERABLE INITIALLY DEFERRED
);
```

Check Constraints in SQL (Not currently supported with InnoDB)

```
CHECK (<condition>)
```

```
create table BankAccount(PersonID char(3),
  AccountID int,
  Amount int CHECK (Amount < 500000)
);
```

Check Constraints
in SQL (Not currently supported with InnoDB)

```
CHECK (<condition>)
```

```
create table BankAccount(PersonID char(3),
  AccountID int, Amount int,
CHECK (Amount < 500000)
);
```

Check Constraints
in SQL (Not currently supported with InnoDB)

```
CONSTRAINT <name>
CHECK (<condition>)
```

```
create table BankAccount(PersonID char(3),
  AccountID int, Amount int,
CONSTRAINT ck_amount
CHECK (Amount < 500000)
);
```

Exercise 7.1.3

- Suggest suitable keys and foreign keys for the relations of the PC database:
  - Product(maker, model, type)
  - PC(model, speed, ram, hd, price)
  - Laptop(model, speed, ram, hd, screen, price)
  - Printer(model, color, type, price)
- Modify your SQL schema to include the declarations of these keys.

Exercise 7.1.3
Example Keys

- Suggest suitable keys and foreign keys for the relations of the PC database:
  - – Product(maker, <u>model</u>, type)
  - – PC(*model*, speed, ram, hd, price)
  - – Laptop(*model*, speed, ram, hd, screen, price)
  - – Printer(*model*, color, type, price)
- Key is underlined
- Foreign Keys Italic

Exercise 7.1.3
Alter Command

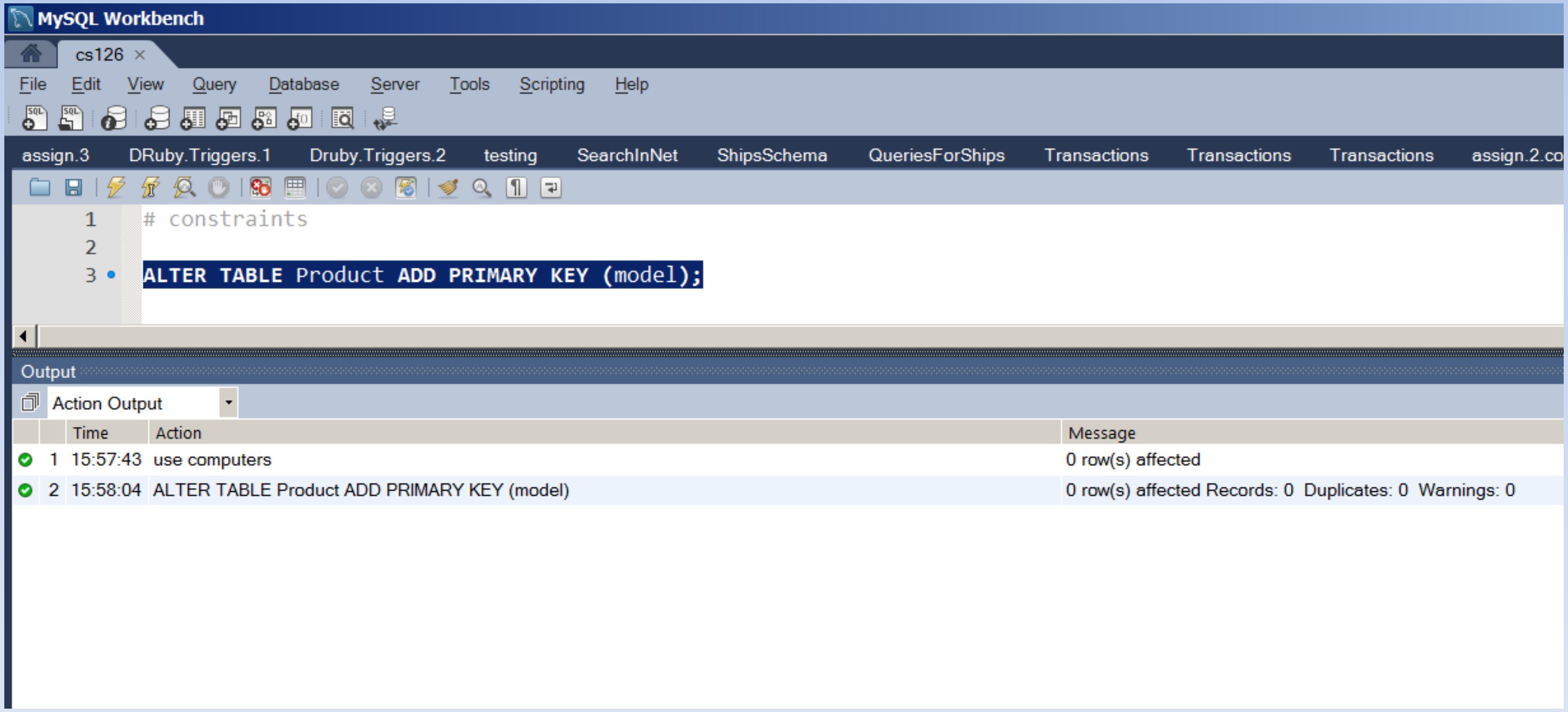
- Add the primary key to table Product

```
ALTER TABLE Product ADD PRIMARY KEY (model);
```

Exercise 7.1.3
Alter Command

- Add the primary key to table Product

ALTER TABLE Product ADD PRIMARY KEY (model);


MySQL Workbench
cs126
File Edit View Query Database Server Tools Scripting Help
assign.3 DRuby.Triggers.1 Druby.Triggers.2 testing SearchInNet ShipsSchema QueriesForShips Transactions Transactions Transactions assign.2.co
Output
Action Output

```
# constraints

ALTER TABLE Product ADD PRIMARY KEY (model);
```

		Time	Action	Message
✓	1	15:57:43	use computers	0 row(s) affected
✓	2	15:58:04	ALTER TABLE Product ADD PRIMARY KEY (model)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0

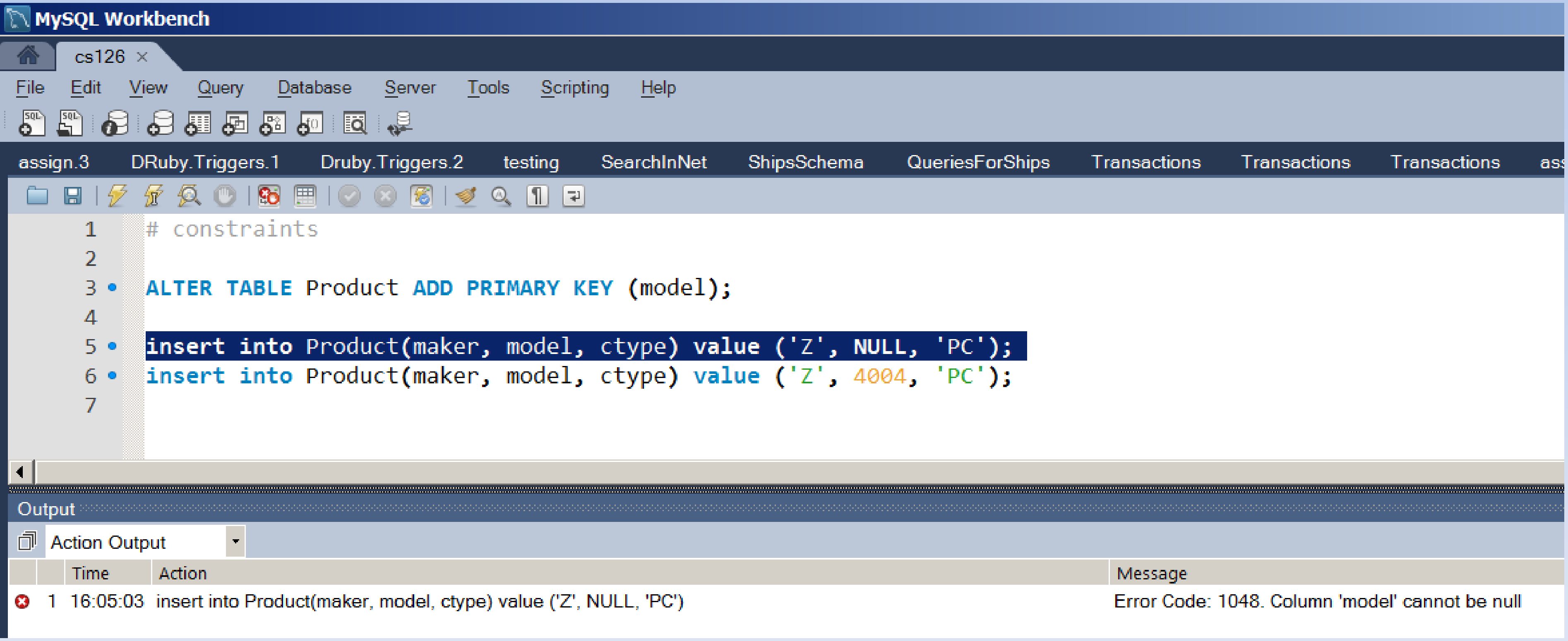
Exercise 7.1.3
Inserting Nulls

- With Model as Primary Key for Product
  - No Nulls
  - No Duplicates

```
insert into Product(maker, model, ctype) value
('Z', NULL, 'PC');
```

Exercise 7.1.3
Inserting Nulls

- With Model as Primary Key for Product
  - No Nulls

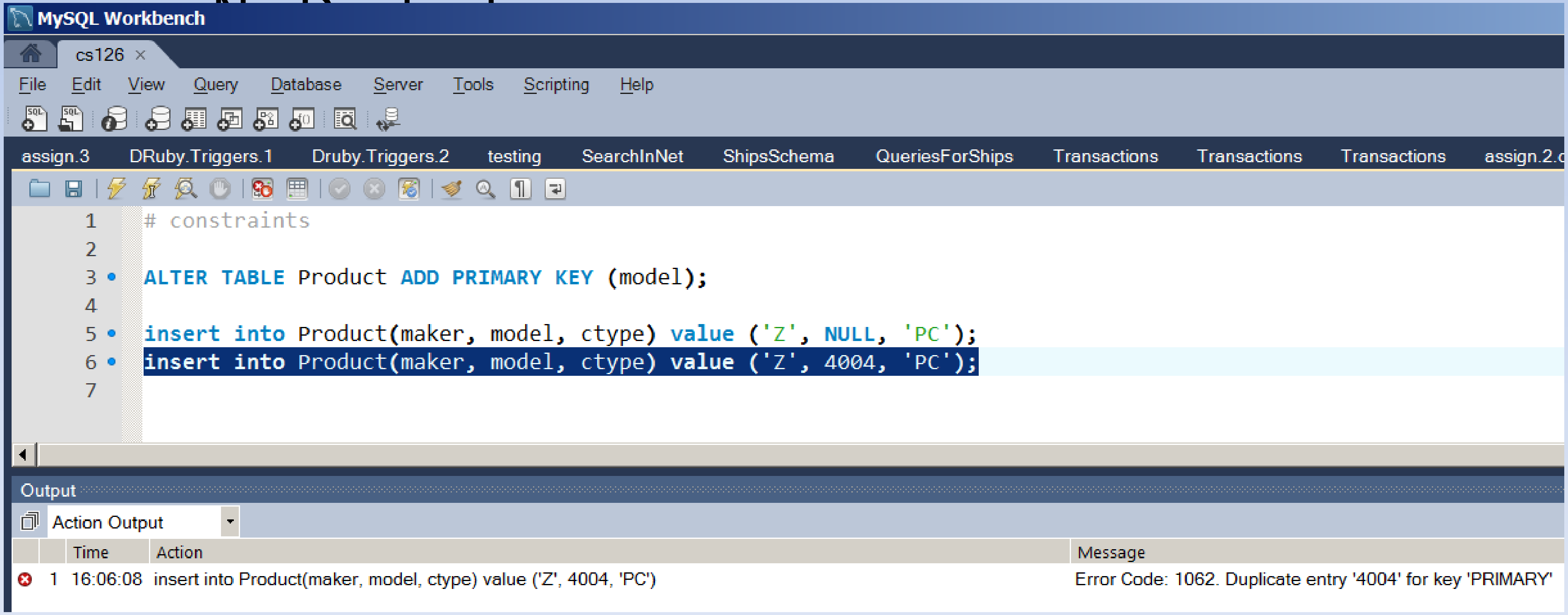

MySQL Workbench
cs126
File Edit View Query Database Server Tools Scripting Help
assign.3 DRuby.Triggers.1 Druby.Triggers.2 testing SearchInNet ShipsSchema QueriesForShips Transactions Transactions Transactions
Output
Action Output

```
# constraints

ALTER TABLE Product ADD PRIMARY KEY (model);

insert into Product(maker, model, ctype) value ('Z', NULL, 'PC');
insert into Product(maker, model, ctype) value ('Z', 4004, 'PC');
```

```
   Time      Action                                                       Message
1  16:05:03  insert into Product(maker, model, ctype) value ('Z', NULL, 'PC')  Error Code: 1048. Column 'model' cannot be null
```

Exercise 7.1.3
Inserting Duplicate

- With Model as Primary Key for Product
  - No Nulls
  - No Duplicat



MySQL Workbench
cs126
File Edit View Query Database Server Tools Scripting Help
assign.3 DRuby.Triggers.1 Druby.Triggers.2 testing SearchInNet ShipsSchema QueriesForShips Transactions Transactions Transactions assign.2.c
Output
Action Output

```
# constraints

ALTER TABLE Product ADD PRIMARY KEY (model);

insert into Product(maker, model, ctype) value ('Z', NULL, 'PC');
insert into Product(maker, model, ctype) value ('Z', 4004, 'PC');
```

```
   Time      Action                                                  Message
1  16:06:08  insert into Product(maker, model, ctype) value ('Z', 4004, 'PC')   Error Code: 1062. Duplicate entry '4004' for key 'PRIMARY'
```

Exercise 7.1.3

  - Product(maker, model, type)
  - PC(model, speed, ram, hd, price)
  - Laptop(model, speed, ram, hd, screen, price)
  - Printer(model, color, type, price)
- **Modify your SQL schema to include the declarations of these keys.**

Exercise 7.1.3
Alter Commands w/ Foreign Keys

- ALTER TABLE Product ADD PRIMARY KEY (model);
- ALTER TABLE PC ADD FOREIGN KEY (model)

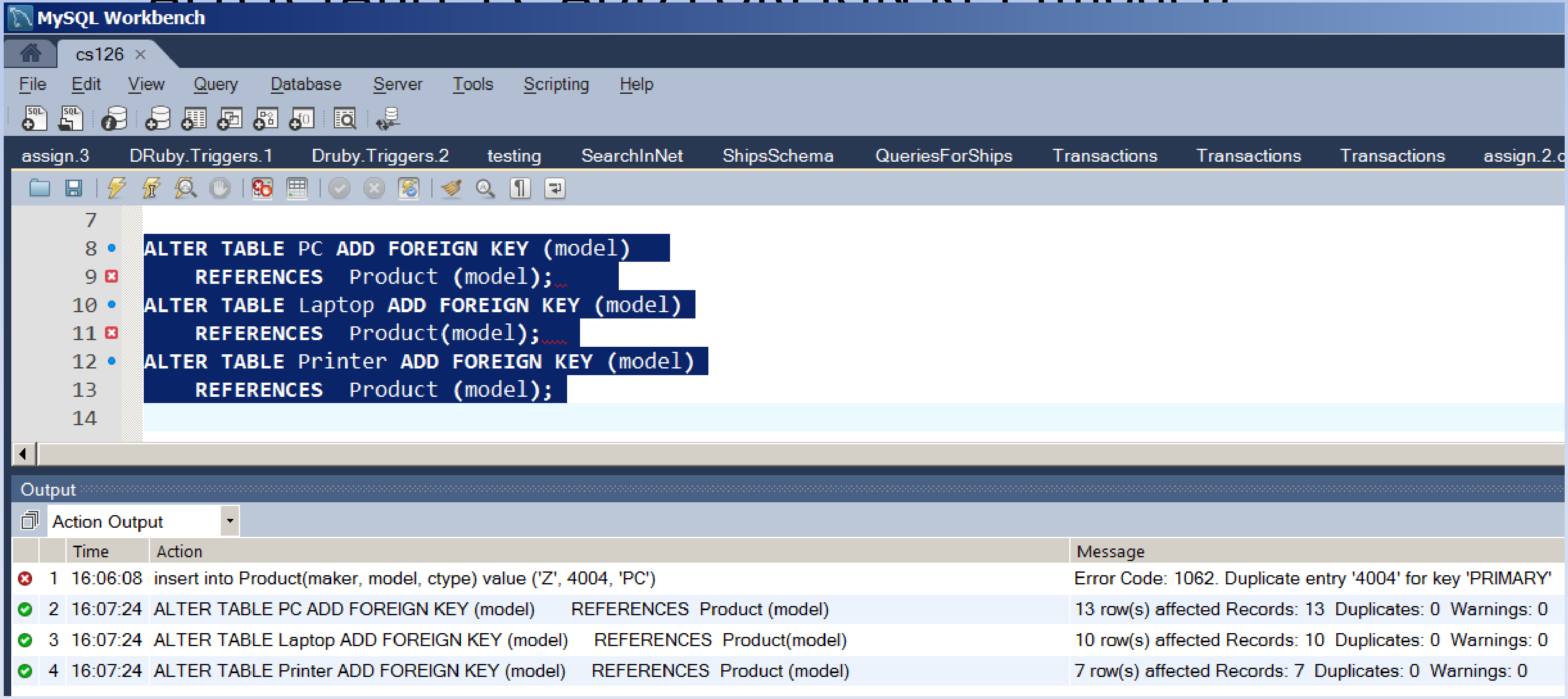
  REFERENCES Product (model);
- ALTER TABLE Laptop ADD FOREIGN KEY (model)

  REFERENCES Product(model);
- ALTER TABLE Printer ADD FOREIGN KEY (model)

  REFERENCES Product (model);

Exercise 7.1.3
Alter Commands w/ Foreign Keys

- ALTER TABLE Product ADD PRIMARY KEY (model);
- ALTER TABLE PC ADD FOREIGN KEY (model)


MySQL Workbench
cs126
File Edit View Query Database Server Tools Scripting Help
assign.3 DRuby.Triggers.1 Druby.Triggers.2 testing SearchInNet ShipsSchema QueriesForShips Transactions Transactions Transactions assign.2.c
Output
Action Output

```
ALTER TABLE PC ADD FOREIGN KEY (model)
    REFERENCES  Product (model);
ALTER TABLE Laptop ADD FOREIGN KEY (model)
    REFERENCES  Product(model);
ALTER TABLE Printer ADD FOREIGN KEY (model)
    REFERENCES  Product (model);
```

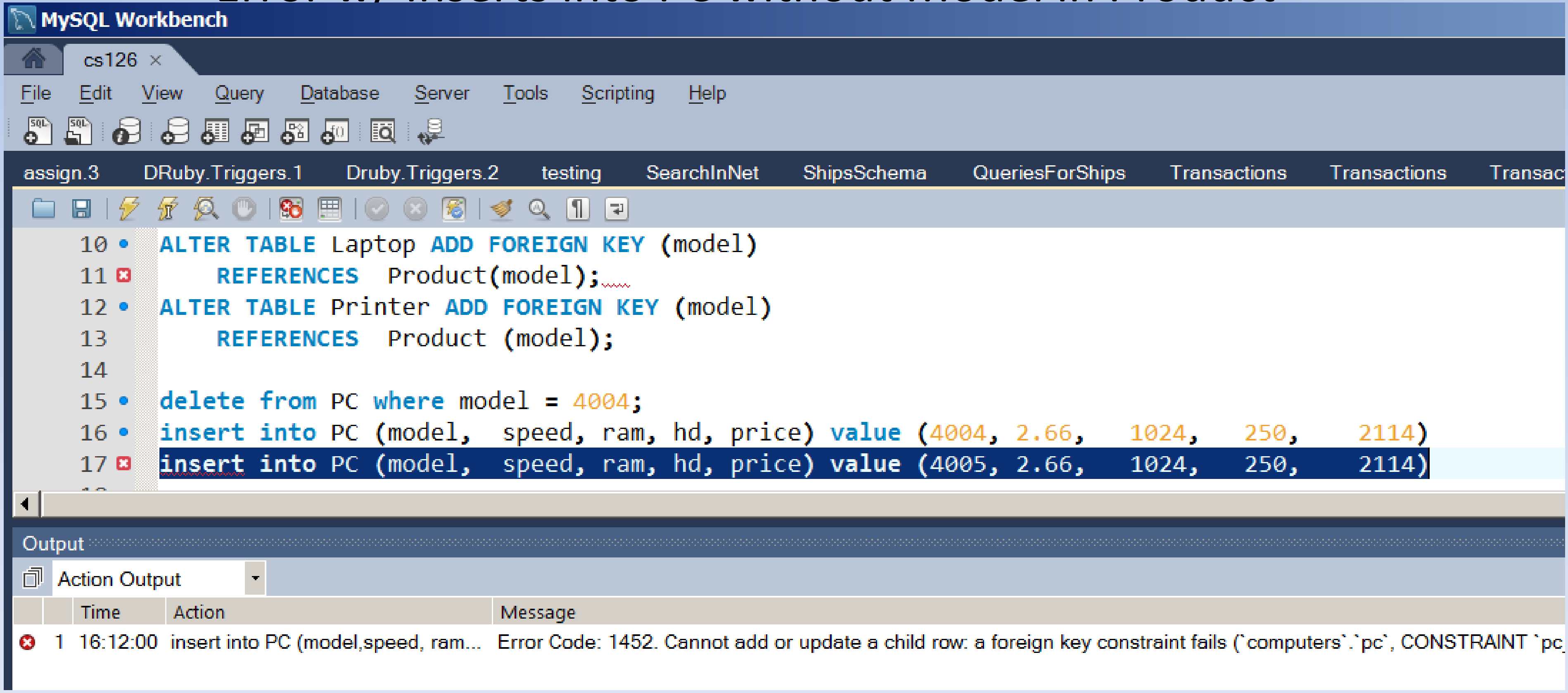
```
   Time      Action                                                                      Message
1  16:06:08  insert into Product(maker, model, ctype) value ('Z', 4004, 'PC')           Error Code: 1062. Duplicate entry '4004' for key 'PRIMARY'
2  16:07:24  ALTER TABLE PC ADD FOREIGN KEY (model)      REFERENCES  Product (model)     13 row(s) affected Records: 13  Duplicates: 0  Warnings: 0
3  16:07:24  ALTER TABLE Laptop ADD FOREIGN KEY (model)      REFERENCES  Product(model)  10 row(s) affected Records: 10  Duplicates: 0  Warnings: 0
4  16:07:24  ALTER TABLE Printer ADD FOREIGN KEY (model)      REFERENCES  Product (model) 7 row(s) affected Records: 7  Duplicates: 0  Warnings: 0
```

Exercise 7.1.3
Insert Commands w/ Foreign Keys

- Now w/ Foreign Keys
  - Error w/ Inserts into PC without Model in Product

Exercise 7.1.3
Insert Commands w/ Foreign Keys

- Now w/ Foreign Keys
  - Error w/ Inserts into PC without Model in Product

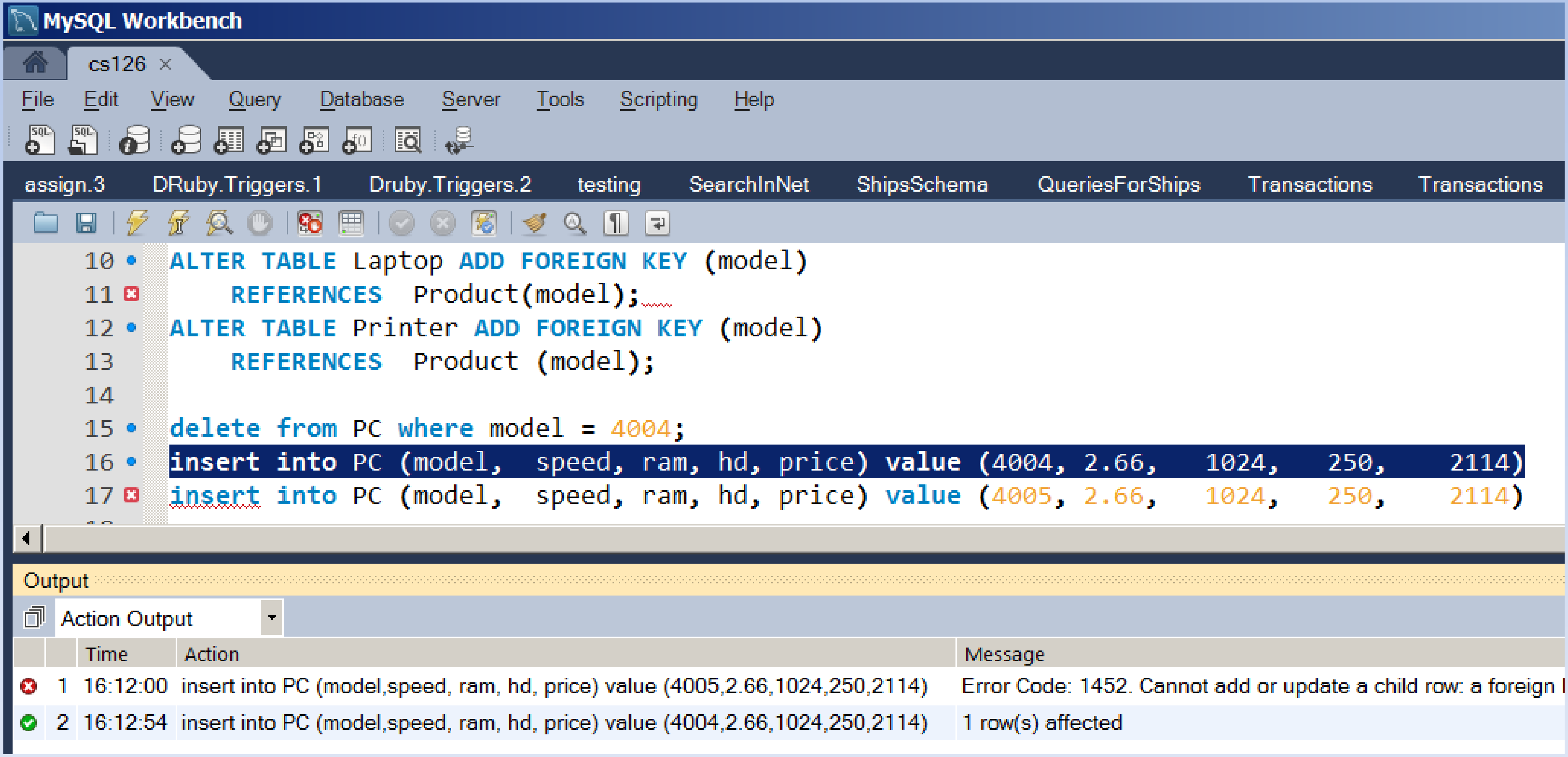

MySQL Workbench
cs126
File Edit View Query Database Server Tools Scripting Help
assign.3 DRuby.Triggers.1 Druby.Triggers.2 testing SearchInNet ShipsSchema QueriesForShips Transactions Transactions Transac
Output
Action Output

```
ALTER TABLE Laptop ADD FOREIGN KEY (model)
    REFERENCES  Product(model);
ALTER TABLE Printer ADD FOREIGN KEY (model)
    REFERENCES  Product (model);

delete from PC where model = 4004;
insert into PC (model,  speed, ram, hd, price) value (4004, 2.66,   1024,   250,    2114)
insert into PC (model,  speed, ram, hd, price) value (4005, 2.66,   1024,   250,    2114)
```

	Time	Action	Message
1	16:12:00	insert into PC (model,speed, ram...	Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`computers`.`pc`, CONSTRAINT `pc

Exercise 7.1.3
Insert Commands w/ Foreign Keys

- Now w/ Foreign Keys


MySQL Workbench
cs126
File Edit View Query Database Server Tools Scripting Help
assign.3 DRuby.Triggers.1 Druby.Triggers.2 testing SearchInNet ShipsSchema QueriesForShips Transactions Transactions
Output
Action Output

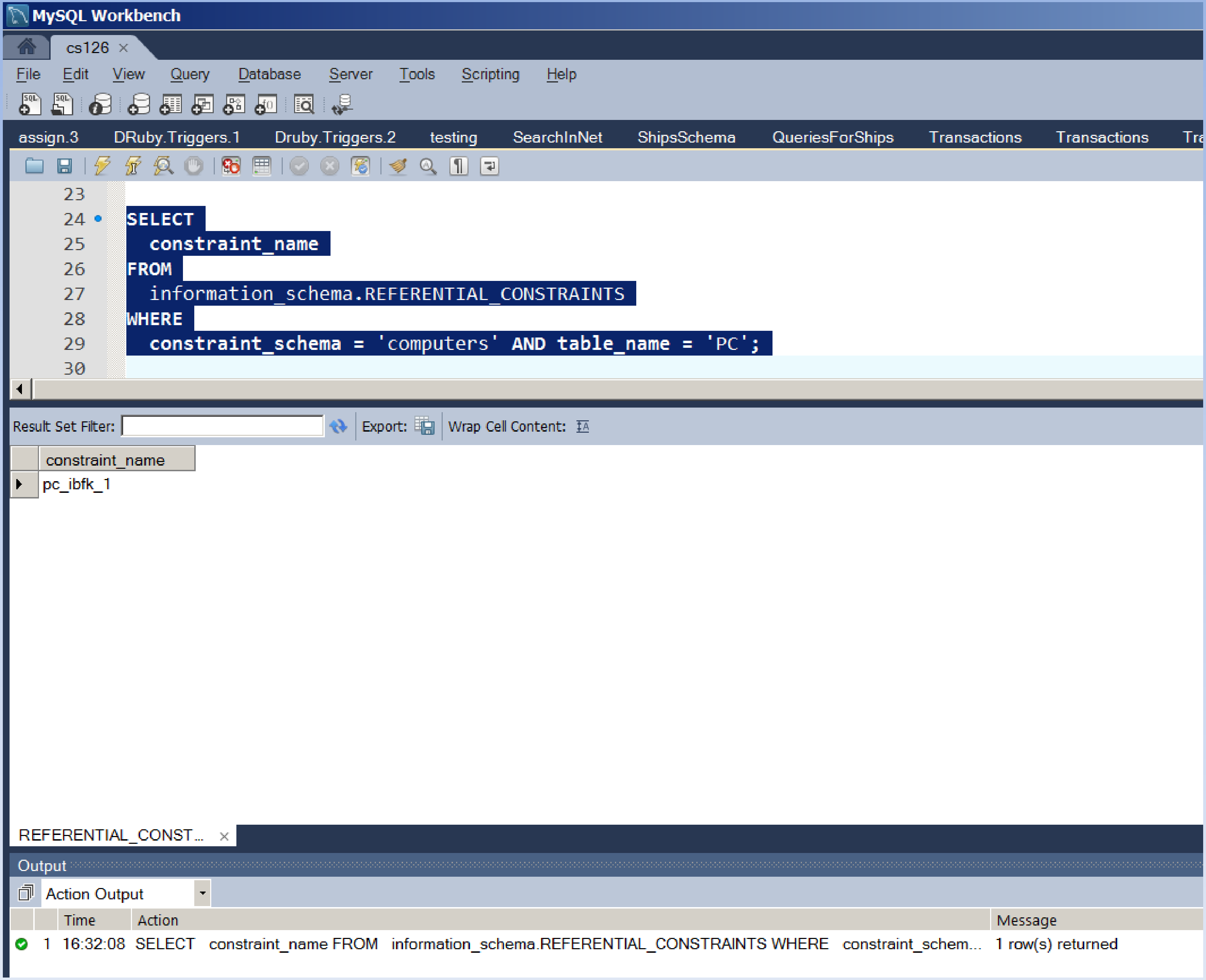
```
ALTER TABLE Laptop ADD FOREIGN KEY (model)
    REFERENCES  Product(model);
ALTER TABLE Printer ADD FOREIGN KEY (model)
    REFERENCES  Product (model);

delete from PC where model = 4004;
insert into PC (model,  speed, ram, hd, price) value (4004, 2.66,   1024,   250,    2114)
insert into PC (model,  speed, ram, hd, price) value (4005, 2.66,   1024,   250,    2114)
```

	Time	Action	Message
1	16:12:00	insert into PC (model,speed, ram, hd, price) value (4005,2.66,1024,250,2114)	Error Code: 1452. Cannot add or update a child row: a foreign
2	16:12:54	insert into PC (model,speed, ram, hd, price) value (4004,2.66,1024,250,2114)	1 row(s) affected

Exercise 7.1.3
Dropping Foreign Keys

- Dropping foreign keys requires finding their name

```
SELECT  constraint_name
FROM  information_schema.REFERENTIAL_CONSTRAINTS
WHERE  constraint_schema = 'computers'
  AND table_name = 'PC';
```


MySQL Workbench
cs126
File Edit View Query Database Server Tools Scripting Help
assign.3 DRuby.Triggers.1 Druby.Triggers.2 testing SearchInNet ShipsSchema QueriesForShips Transactions Transactions

```
SELECT
  constraint_name
FROM
  information_schema.REFERENTIAL_CONSTRAINTS
WHERE
  constraint_schema = 'computers' AND table_name = 'PC';
```

Result Set Filter: Export: Wrap Cell Content:

constraint_name
pc_ibfk_1

REFERENTIAL_CONST...
Output
Action Output

	Time	Action	Message
1	16:32:08	SELECT constraint_name FROM information_schema.REFERENTIAL_CONSTRAINTS WHERE constraint_schem...	1 row(s) returned

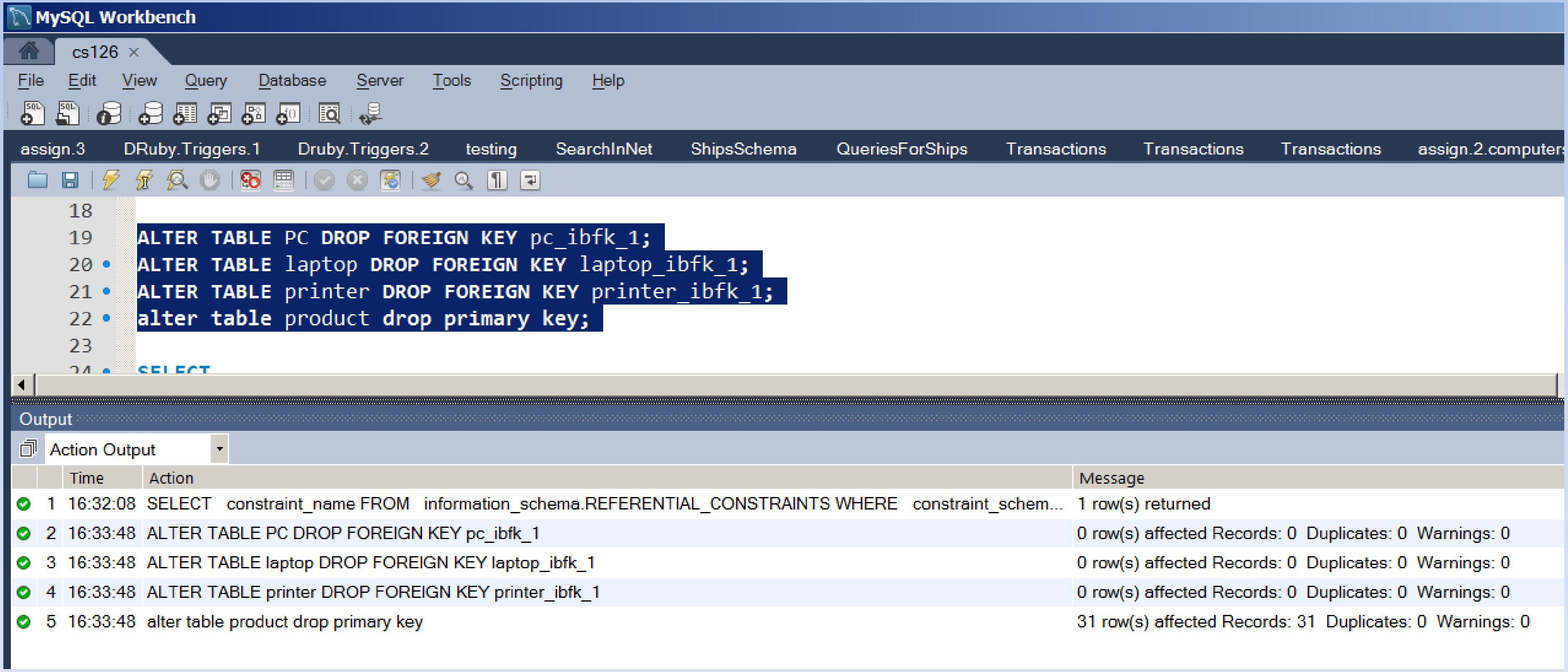
Exercise 7.1.3
Dropping Foreign Keys

- Dropping named foreign key:

```
ALTER TABLE PC DROP FOREIGN KEY pc_ibfk_1;
ALTER TABLE laptop DROP FOREIGN KEY laptop_ibfk_1;
ALTER TABLE printer DROP FOREIGN KEY printer_ibfk_1;

ALTER TABLE product DROP PRIMARY KEY;
```

Exercise 7.1.3
Dropping Foreign Keys

- Dropping named foreign key:


MySQL Workbench
cs126
File Edit View Query Database Server Tools Scripting Help
assign.3 DRuby.Triggers.1 Druby.Triggers.2 testing SearchInNet ShipsSchema QueriesForShips Transactions Transactions Transactions assign.2.computers
Output
Action Output

```
ALTER TABLE PC DROP FOREIGN KEY pc_ibfk_1;
ALTER TABLE laptop DROP FOREIGN KEY laptop_ibfk_1;
ALTER TABLE printer DROP FOREIGN KEY printer_ibfk_1;
alter table product drop primary key;

SELECT
```

	Time	Action	Message
1	16:32:08	SELECT constraint_name FROM information_schema.REFERENTIAL_CONSTRAINTS WHERE constraint_schem...	1 row(s) returned
2	16:33:48	ALTER TABLE PC DROP FOREIGN KEY pc_ibfk_1	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
3	16:33:48	ALTER TABLE laptop DROP FOREIGN KEY laptop_ibfk_1	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
4	16:33:48	ALTER TABLE printer DROP FOREIGN KEY printer_ibfk_1	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
5	16:33:48	alter table product drop primary key	31 row(s) affected Records: 31 Duplicates: 0 Warnings: 0

Exercise 7.1.4

- Suggest suitable keys and foreign keys for the relations of the battleships database:

Classes(class, stype, country, numGuns,
  bore, displacement);
Battles(bname, bdate);
Outcomes(ship, battle, result);
Ships(sname, class, launched);

Exercise 7.1.4

- Suggest suitable keys and foreign keys for the relations of the battleships database:

Classes(<u>class</u>, stype, country, numGuns,
bore, displacement);

Battles(<u>bname</u>, bdate);

Outcomes(ship, battle, result);

Ships(<u>sname</u>, class, launched);

Exercise 7.1.4

- Suggest suitable keys and foreign keys for the relations of the battleships database:

```
ALTER TABLE Classes
		ADD PRIMARY KEY (class);

ALTER TABLE Ships
		ADD PRIMARY KEY (sname);

ALTER TABLE Ships
		ADD FOREIGN KEY (class)  REFERENCES Classes (class);

ALTER TABLE Battles
		ADD PRIMARY KEY (bname);

ALTER TABLE Outcomes
		ADD FOREIGN KEY (ship) REFERENCES Ships (sname);

ALTER TABLE Outcomes
		ADD FOREIGN KEY (battle) REFERENCES Battles (sname);

ALTER TABLE Outcomes
		ADD PRIMARY KEY (ship, battle);
```

Exercise 7.1.5

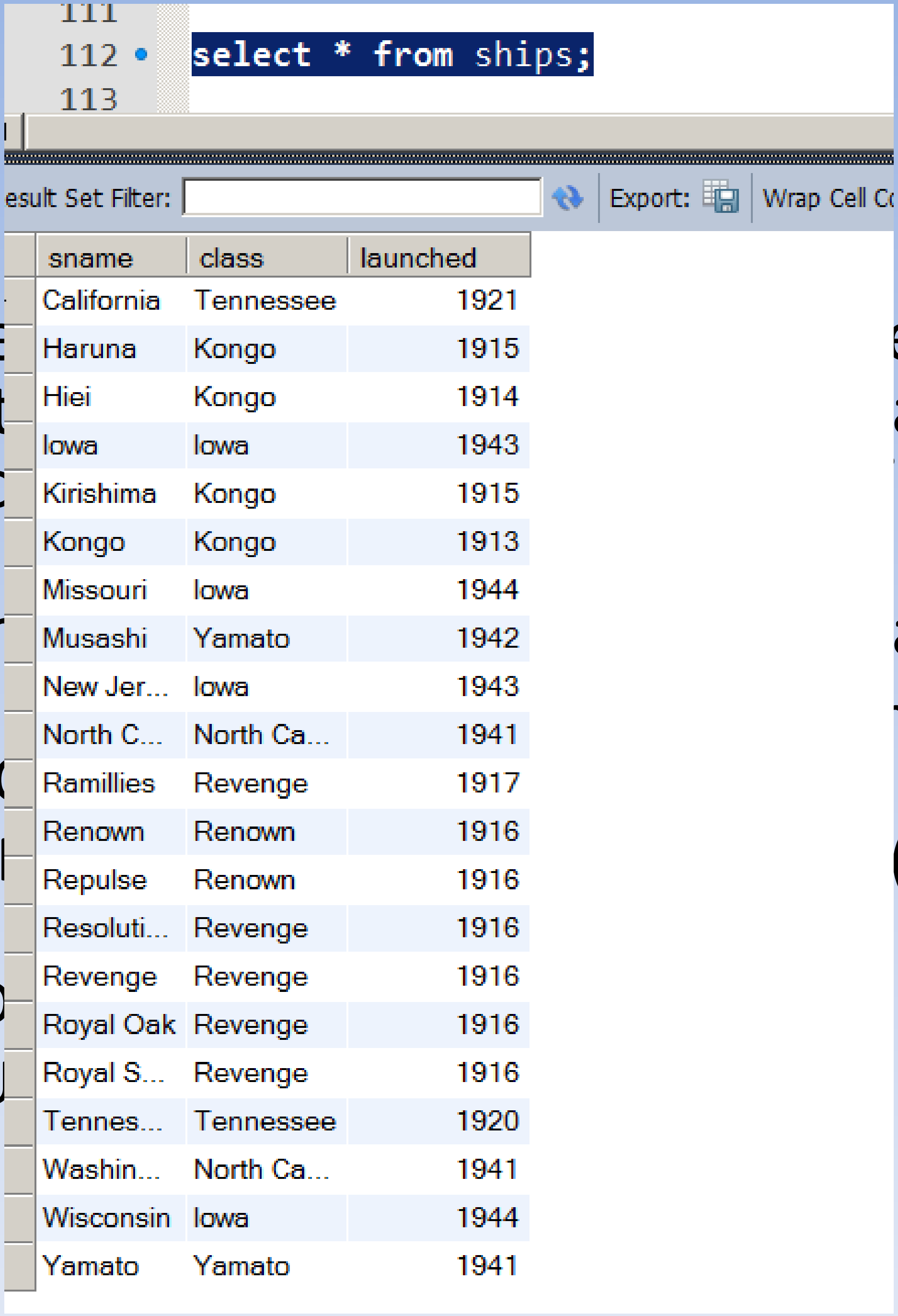
- Write the following referential integrity constraints for the battleships database as in Exercise 7.1.4: Use your assumptions about keys from that exercise, and handle all violations by setting the referencing attribute value to NULL
  - (a) Every Class mentioned in Ships must be mentioned in Classes.

Exercise 7.1.5

- Write the following referential integrity constraints for the battleships database as in Exercise 7.1.4: Use your assumptions about keys from that exercise, and handle all violations by setting the referencing attribute value to NULL
  - (a) Every Class mentioned in Ships must be mentioned in Classes.

```
ALTER TABLE Ships ADD FOREIGN KEY (class)
REFERENCES Classes (class)
ON DELETE SET NULL
ON UPDATE SET NULL;
```

- Write … egrity const… abase as in Exerc… ons about keys from … violations by settir… alue to NULL
  - (a) … ust be mentioned in (
  ALTE… (class) REFE…
  ON D…
  ON U…



```
select * from ships;
```

Result Set Filter: Export: Wrap Cell Co

sname	class	launched
California	Tennessee	1921
Haruna	Kongo	1915
Hiei	Kongo	1914
Iowa	Iowa	1943
Kirishima	Kongo	1915
Kongo	Kongo	1913
Missouri	Iowa	1944
Musashi	Yamato	1942
New Jer...	Iowa	1943
North C...	North Ca...	1941
Ramillies	Revenge	1917
Renown	Renown	1916
Repulse	Renown	1916
Resoluti...	Revenge	1916
Revenge	Revenge	1916
Royal Oak	Revenge	1916
Royal S...	Revenge	1916
Tennes...	Tennessee	1920
Washin...	North Ca...	1941
Wisconsin	Iowa	1944
Yamato	Yamato	1941

Exercise 7.1.5

```
ALTER TABLE Ships ADD FOREIGN KEY (class)
REFERENCES Classes (class)
ON DELETE SET NULL
ON UPDATE SET NULL;
```

Now Execute Update Query:

```
delete from classes where class = 'Tennessee';
```

AL

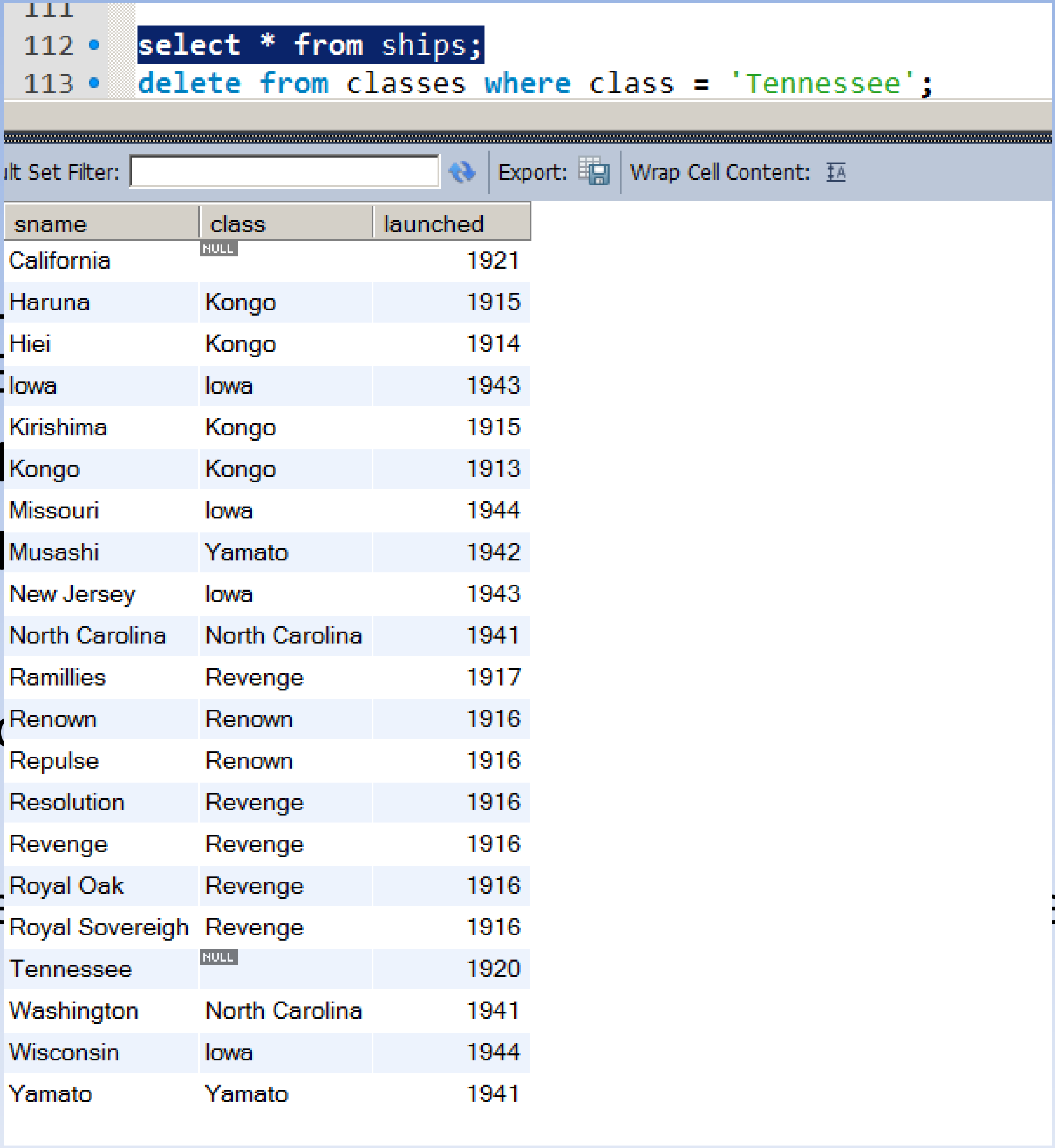
RE

Ol

Ol

N

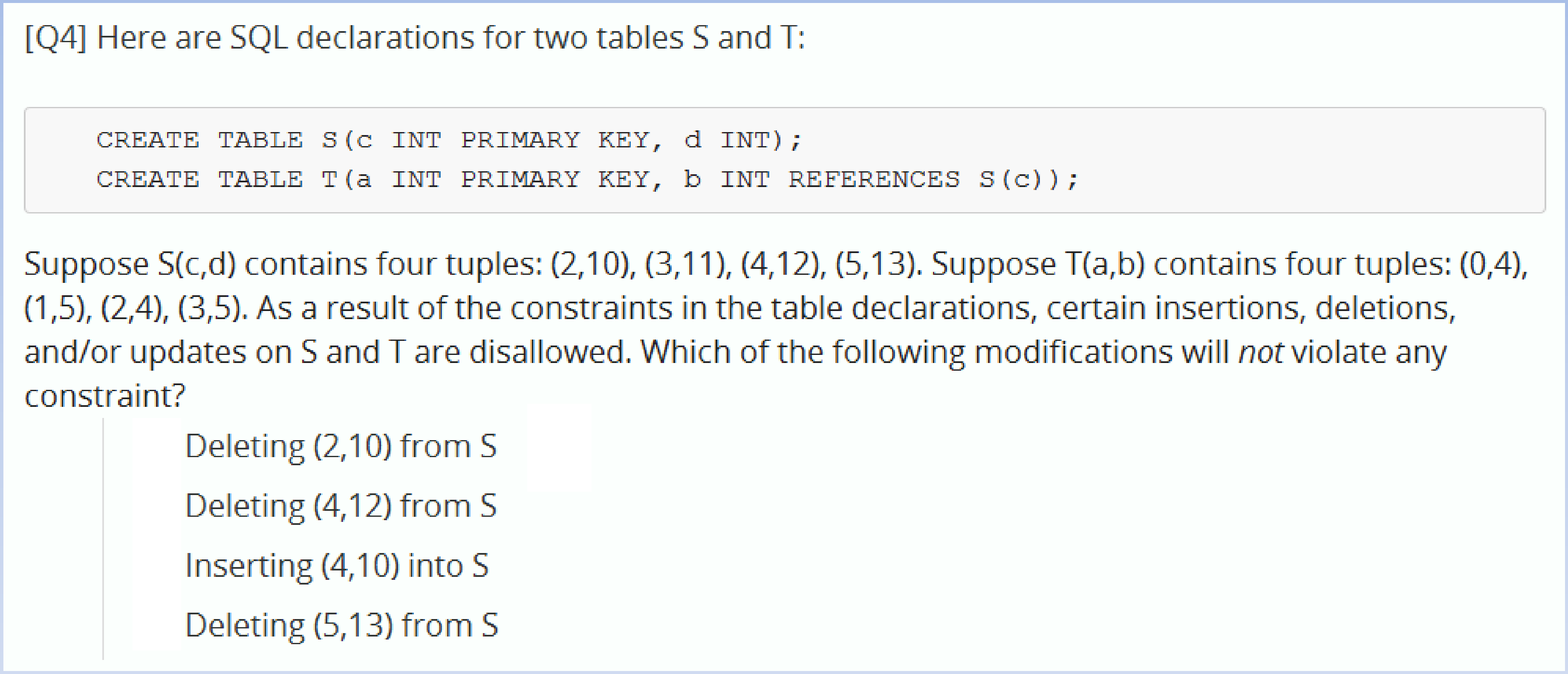
de e';



```sql
select * from ships;
delete from classes where class = 'Tennessee';
```

sname	class	launched
California	NULL	1921
Haruna	Kongo	1915
Hiei	Kongo	1914
Iowa	Iowa	1943
Kirishima	Kongo	1915
Kongo	Kongo	1913
Missouri	Iowa	1944
Musashi	Yamato	1942
New Jersey	Iowa	1943
North Carolina	North Carolina	1941
Ramillies	Revenge	1917
Renown	Renown	1916
Repulse	Renown	1916
Resolution	Revenge	1916
Revenge	Revenge	1916
Royal Oak	Revenge	1916
Royal Sovereigh	Revenge	1916
Tennessee	NULL	1920
Washington	North Carolina	1941
Wisconsin	Iowa	1944
Yamato	Yamato	1941

Stanford Online



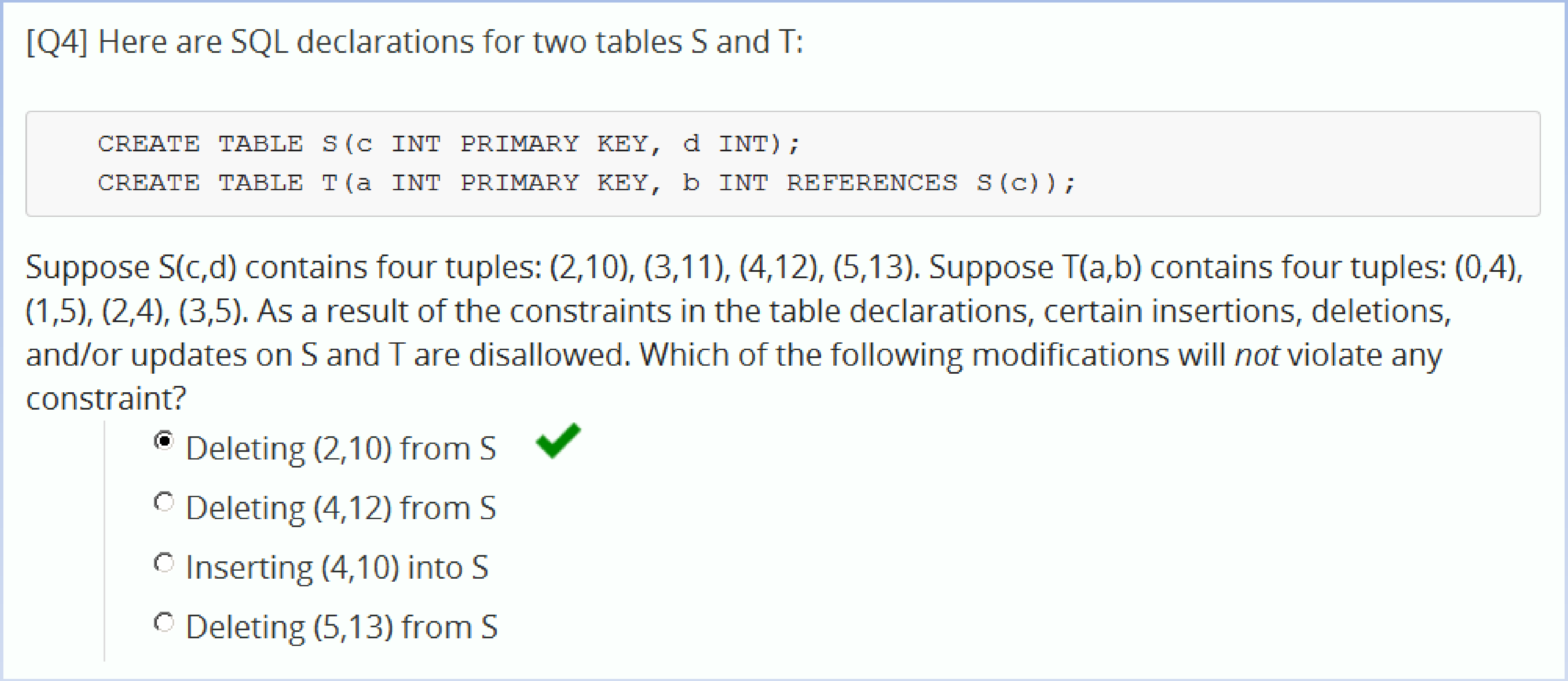
[Q4] Here are SQL declarations for two tables S and T:

```
    CREATE TABLE S(c INT PRIMARY KEY, d INT);
    CREATE TABLE T(a INT PRIMARY KEY, b INT REFERENCES S(c));
```

Suppose S(c,d) contains four tuples: (2,10), (3,11), (4,12), (5,13). Suppose T(a,b) contains four tuples: (0,4), (1,5), (2,4), (3,5). As a result of the constraints in the table declarations, certain insertions, deletions, and/or updates on S and T are disallowed. Which of the following modifications will *not* violate any constraint?

> Deleting (2,10) from S
>
> Deleting (4,12) from S
>
> Inserting (4,10) into S
>
> Deleting (5,13) from S

Stanford Online



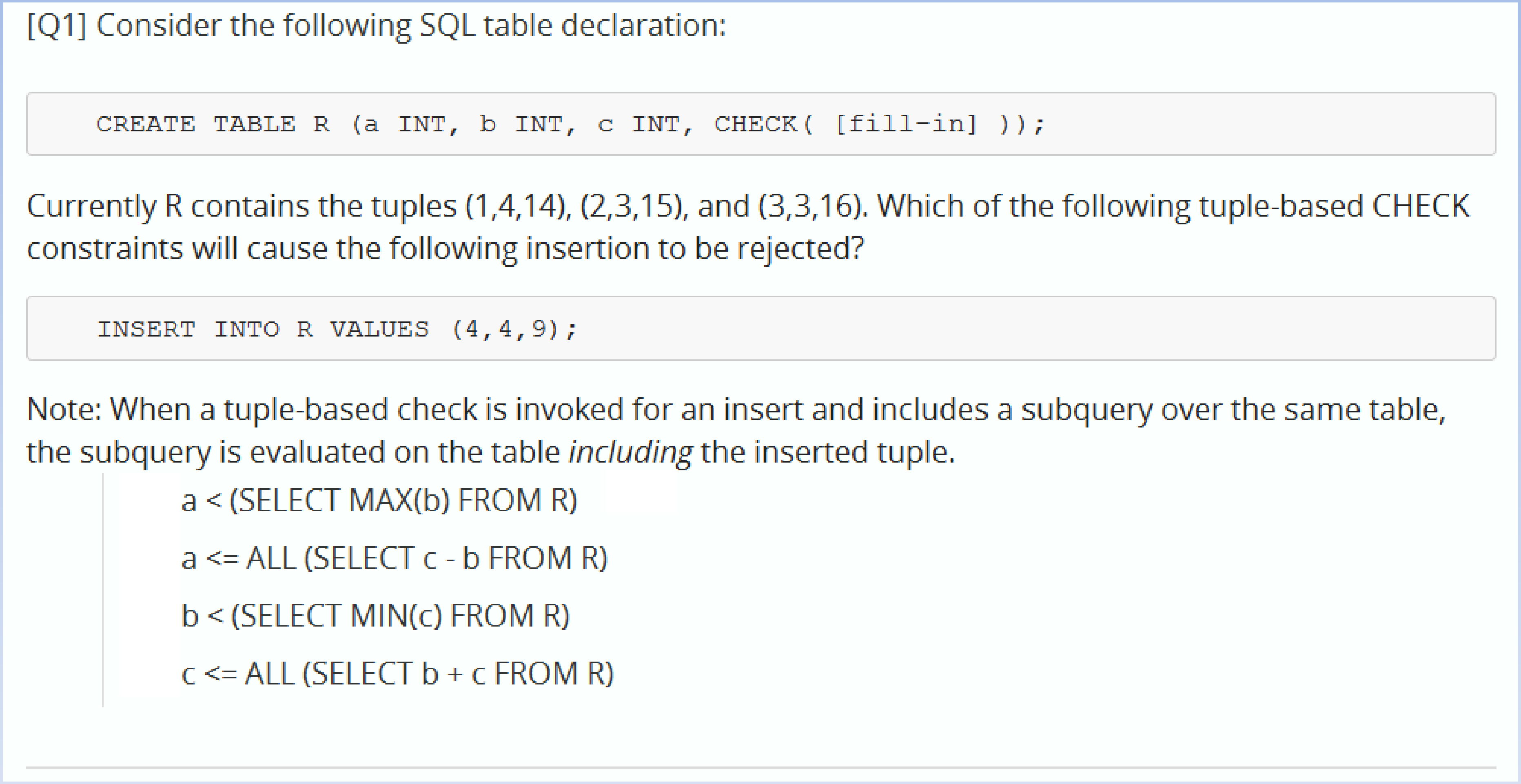
[Q4] Here are SQL declarations for two tables S and T:

```
CREATE TABLE S(c INT PRIMARY KEY, d INT);
CREATE TABLE T(a INT PRIMARY KEY, b INT REFERENCES S(c));
```

Suppose S(c,d) contains four tuples: (2,10), (3,11), (4,12), (5,13). Suppose T(a,b) contains four tuples: (0,4), (1,5), (2,4), (3,5). As a result of the constraints in the table declarations, certain insertions, deletions, and/or updates on S and T are disallowed. Which of the following modifications will *not* violate any constraint?

- [✓ Deleting (2,10) from S] ✓
- Deleting (4,12) from S
- Inserting (4,10) into S
- Deleting (5,13) from S

Stanford Online



[Q1] Consider the following SQL table declaration:

```
CREATE TABLE R (a INT, b INT, c INT, CHECK( [fill-in] ));
```

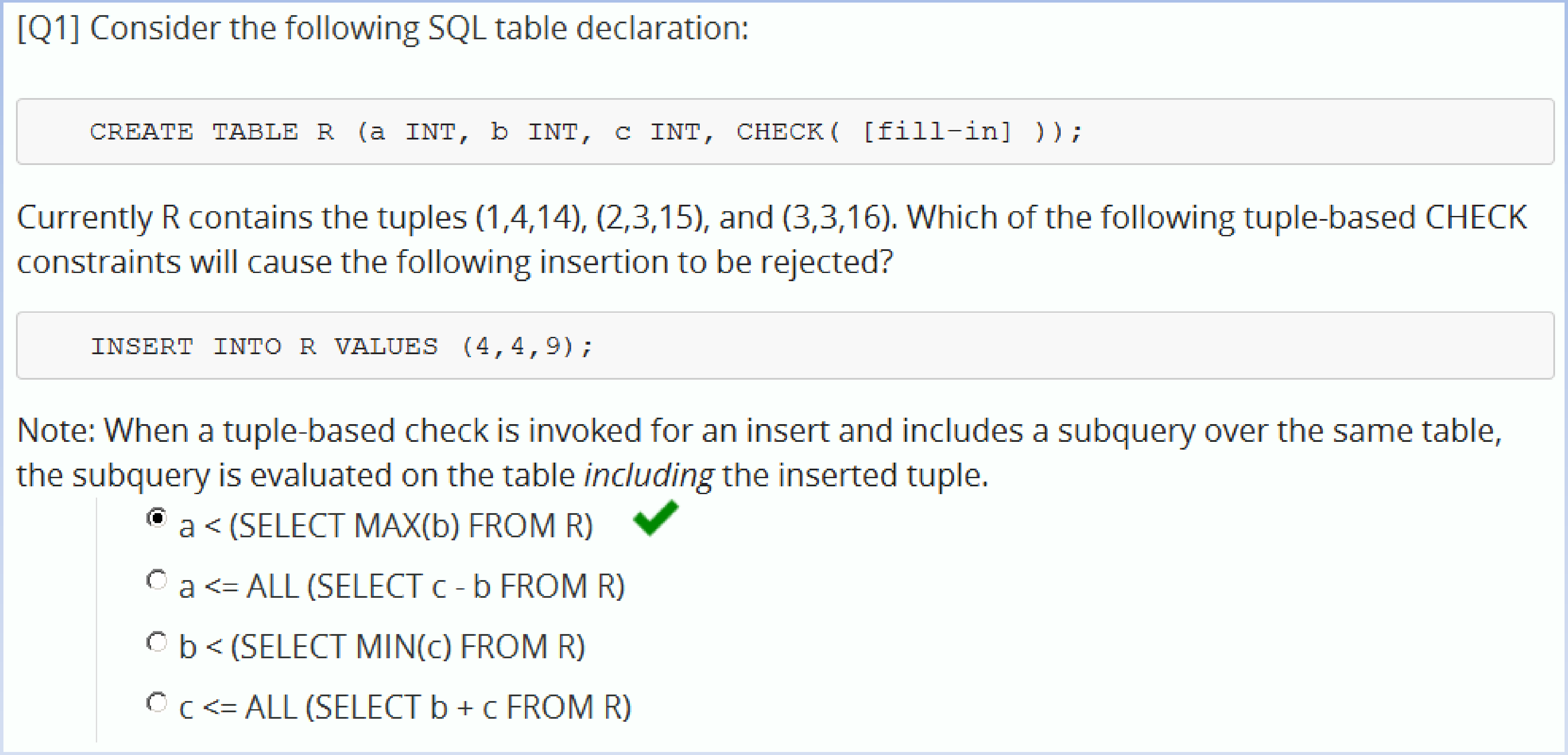
Currently R contains the tuples (1,4,14), (2,3,15), and (3,3,16). Which of the following tuple-based CHECK constraints will cause the following insertion to be rejected?

```
INSERT INTO R VALUES (4,4,9);
```

Note: When a tuple-based check is invoked for an insert and includes a subquery over the same table, the subquery is evaluated on the table *including* the inserted tuple.

> a < (SELECT MAX(b) FROM R)
>
> a <= ALL (SELECT c - b FROM R)
>
> b < (SELECT MIN(c) FROM R)
>
> c <= ALL (SELECT b + c FROM R)

Stanford Online



[Q1] Consider the following SQL table declaration:

```
    CREATE TABLE R (a INT, b INT, c INT, CHECK( [fill-in] ));
```

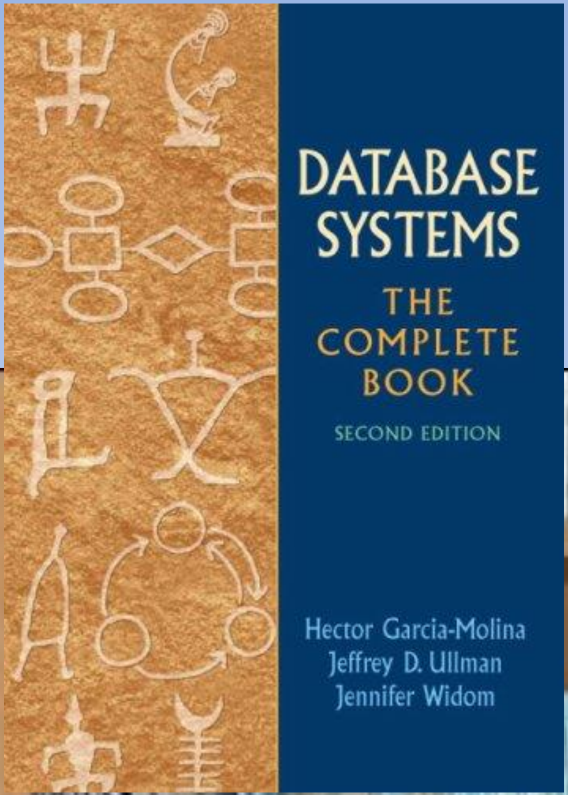
Currently R contains the tuples (1,4,14), (2,3,15), and (3,3,16). Which of the following tuple-based CHECK constraints will cause the following insertion to be rejected?

```
    INSERT INTO R VALUES (4,4,9);
```

Note: When a tuple-based check is invoked for an insert and includes a subquery over the same table, the subquery is evaluated on the table *including* the inserted tuple.

- [✓ a < (SELECT MAX(b) FROM R)] ✓
- a <= ALL (SELECT c - b FROM R)
- b < (SELECT MIN(c) FROM R)
- c <= ALL (SELECT b + c FROM R)

Chapter 8: Views & **Indexes**



Views

- A *view* is a relation defined in terms of stored tables (called *base tables*) and other views.
- Two kinds:

1. *Virtual* = not stored in the database; just a query for constructing the relation.
2. *Materialized* = actually constructed and stored.

Declaring Views

- Declare by:

  CREATE [MATERIALIZED] VIEW <name> AS <query>;
- Default is virtual.
- MySQL does not allow Materialized Views

Example: View Definition

SunkShips

`(ShipName, Class, Type, BattleName, Date)`

is a view “containing” ships that sunk in battle, along with their class, battle name, and date sunk.

- First: create a sunk ships query:

Classes(class, stype, country, numGuns, bore, displacement);

Battles(bname, bdate);

Outcomes(ship, battle, result);

Ships(sname, class, launched);

Ships Schema

- Classes(class, stype, country, numGuns, bore, displacement)
- Ships(sname, class, launched)
- Battles(bname, bdate)
- Outcomes(Ship, Battle, result)

Example: View Definition

SunkShips
 (ShipName, Class, Type, BattleName, Date)
Classes(class, stype, country, numGuns, bore, displacement);
Battles(bname, bdate);
Outcomes(ship, battle, result);
Ships(sname, class, launched);

- Attributes/Tables:

- Class, Stype:	**Classes**
- ShipName, Class:	**Ships**
- Sunk Test:	**Outcomes**
- BattleName, Date:	**Battles**

Example: View Definition

SunkShips
 `(ShipName, Class, Type, BattleName, Date)`
is a view “containing” ships that sunk in battle, along with their class, battle name, and date sunk.

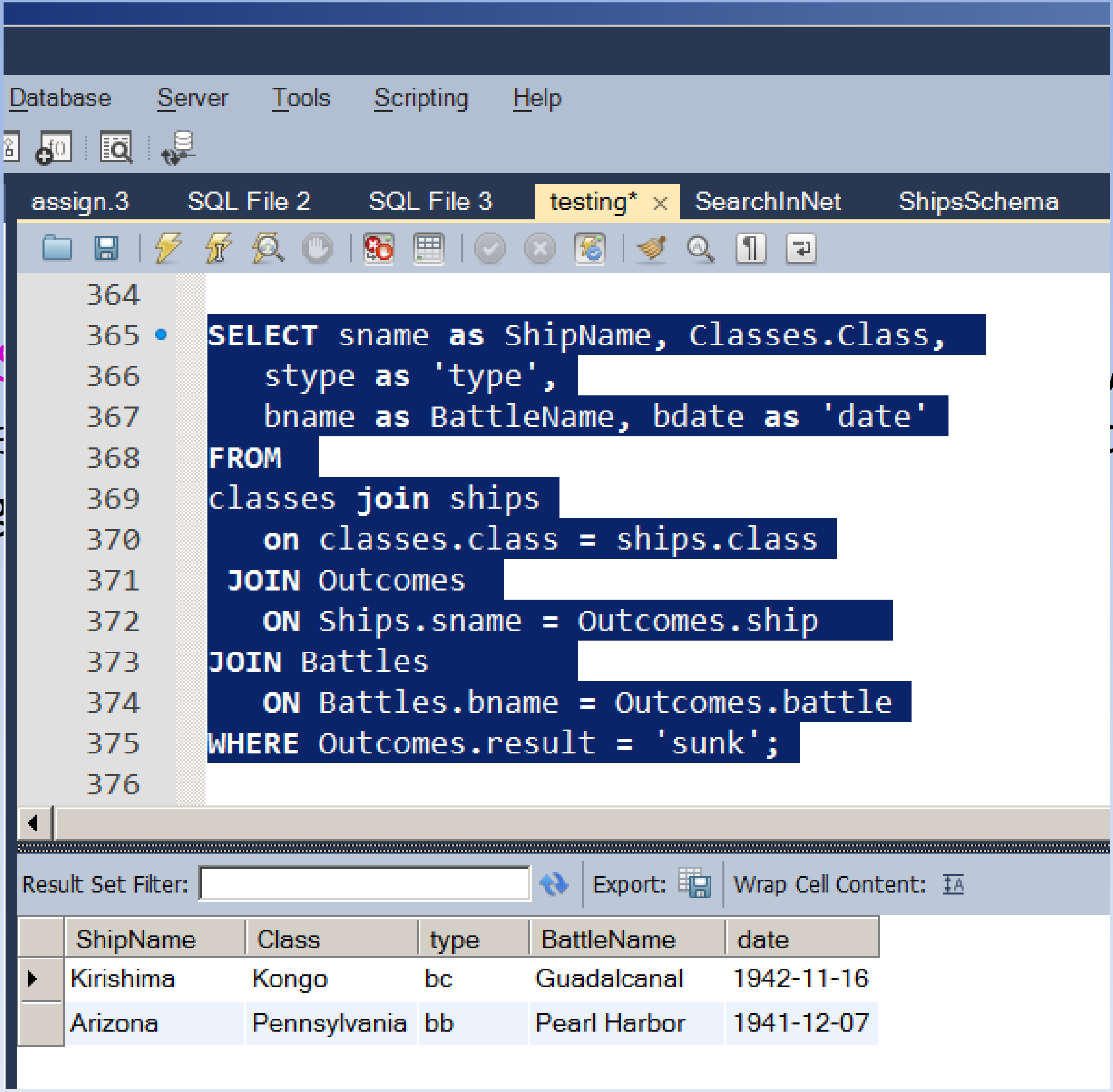
- First: create a sunk ships query:

```
SELECT sname as ShipName,
    Classes.Class, stype as 'type',
    bname as BattleName, bdate as 'date'
FROM
classes JOIN ships
   ON classes.class = ships.class
JOIN Outcomes
   ON Ships.sname = Outcomes.ship
JOIN Battles
   ON Battles.bname = Outcomes.battle
WHERE Outcomes.result = 'sunk';
```

Sunk

Date

along



```
SELECT sname as ShipName, Classes.Class,
   stype as 'type',
   bname as BattleName, bdate as 'date'
FROM
classes join ships
   on classes.class = ships.class
 JOIN Outcomes
   ON Ships.sname = Outcomes.ship
JOIN Battles
   ON Battles.bname = Outcomes.battle
WHERE Outcomes.result = 'sunk';
```

ShipName	Class	type	BattleName	date
Kirishima	Kongo	bc	Guadalcanal	1942-11-16
Arizona	Pennsylvania	bb	Pearl Harbor	1941-12-07

,

le,

Example: View Definition

SunkShips
 (ShipName, Class, Type, BattleName, Date)
is a view “containing” ships that sunk in battle, along with their class, battle name, and date sunk:

```
CREATE VIEW SunkShips AS
SELECT sname as ShipName,
    Classes.Class, stype as 'type',
    bname as BattleName, bdate as 'date'
FROM
classes JOIN ships
   ON classes.class = ships.class
JOIN Outcomes
   ON Ships.sname = Outcomes.ship
JOIN Battles
   ON Battles.bname = Outcomes.battle
WHERE Outcomes.result = 'sunk';
```

Example: Accessing a View

- Query a view as if it were a base table.
  - Also: a limited ability to modify views if it makes sense as a modification of one underlying base table.
- Example query:

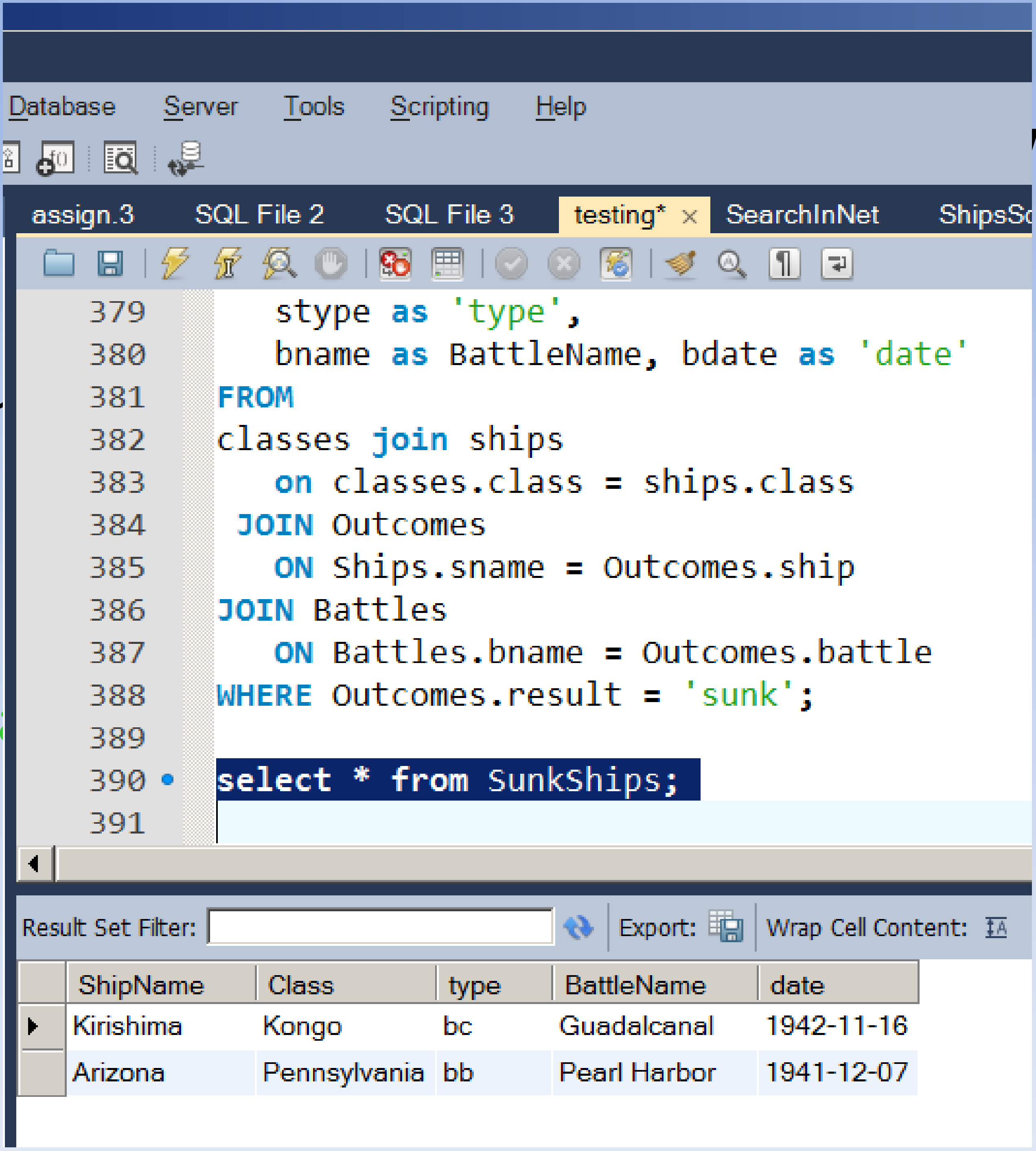
```
SELECT * FROM SunkShips;
```

• Qu

–

• Ex

SEL



```
    stype as 'type',
    bname as BattleName, bdate as 'date'
FROM
classes join ships
    on classes.class = ships.class
 JOIN Outcomes
    ON Ships.sname = Outcomes.ship
JOIN Battles
    ON Battles.bname = Outcomes.battle
WHERE Outcomes.result = 'sunk';

select * from SunkShips;
```

Result Set Filter: Export: Wrap Cell Content:

ShipName	Class	type	BattleName	date
Kirishima	Kongo	bc	Guadalcanal	1942-11-16
Arizona	Pennsylvania	bb	Pearl Harbor	1941-12-07

Example: Accessing a View

- Another query:
  - Find just the name of every battleship sunk.

Original Schema:

Classes(class, stype, country, numGuns, bore, displacement);

Battles(bname, bdate);

Outcomes(ship, battle, result);

Ships(sname, class, launched);

Example: Accessing a View

- Another query:
  - Find just the name of every battleship sunk.

Original Schema + View:

SunkShips

`(ShipName, Class, Type, BattleName, Date)`

Classes(class, stype, country, numGuns, bore, displacement);

Battles(bname, bdate);

Outcomes(ship, battle, result);

Ships(sname, class, launched);

Example: Accessing a View

- Another query:
  - Find just the name of every battleship sunk.

SELECT ShipName from SunkShips;

Original Schema + View:

SunkShips
 (`ShipName, Class, Type, BattleName, Date`)

Classes(class, stype, country, numGuns, bore, displacement);

Battles(bname, bdate);

Outcomes(ship, battle, result);

Ships(sname, class, launched);

Example: Accessing a View

- Another query:
  - Find sunk battleships

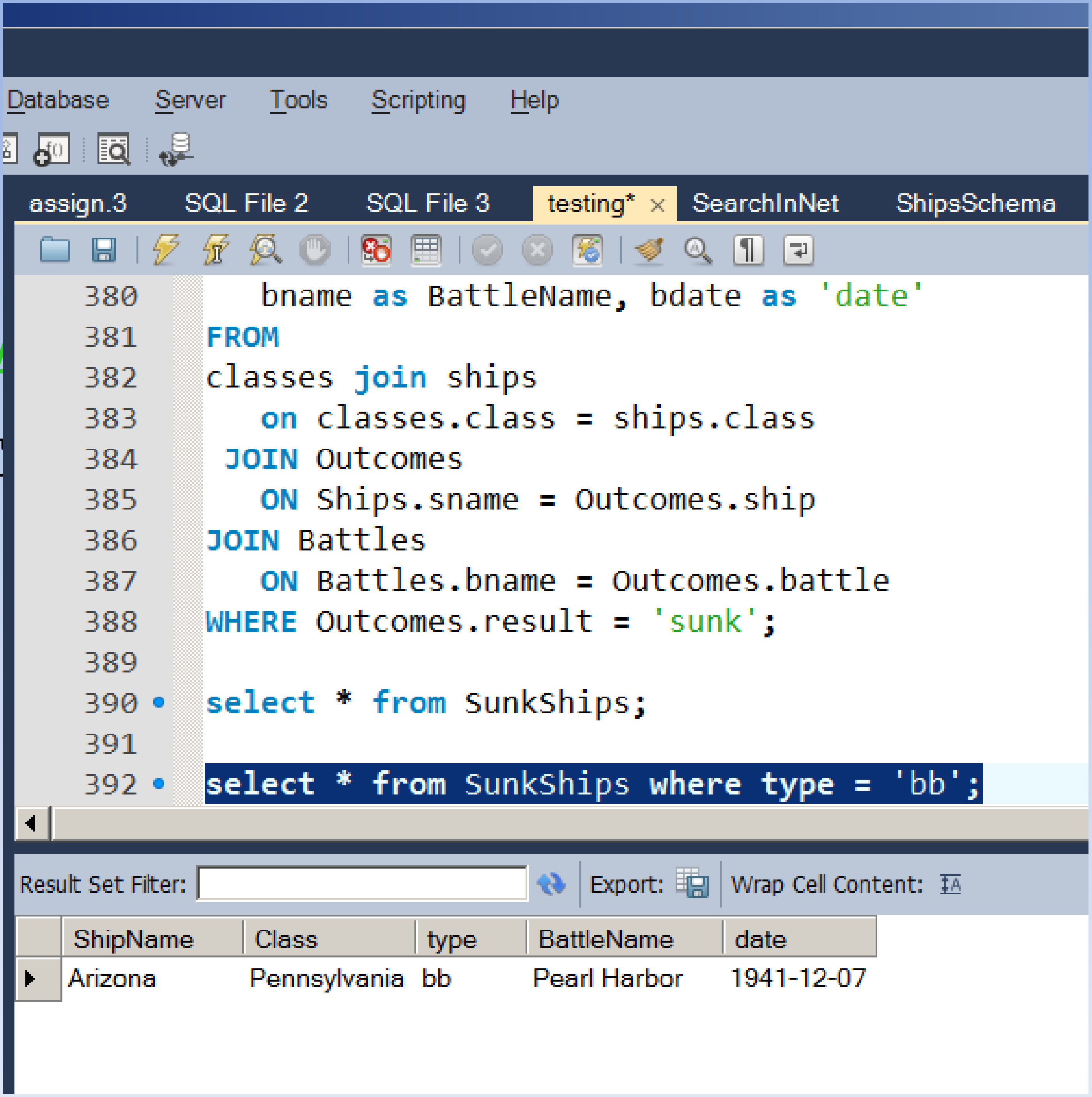
Example: Accessing a View

- Another query:
  - Find sunk battleships

```
select * from SunkShips where
  type = 'bb';
```

- A

SE s



```
   bname as BattleName, bdate as 'date'
FROM
classes join ships
   on classes.class = ships.class
 JOIN Outcomes
   ON Ships.sname = Outcomes.ship
JOIN Battles
   ON Battles.bname = Outcomes.battle
WHERE Outcomes.result = 'sunk';

select * from SunkShips;

select * from SunkShips where type = 'bb';
```

ShipName	Class	type	BattleName	date
Arizona	Pennsylvania	bb	Pearl Harbor	1941-12-07

Example: Accessing a View

- Another query:
  - Find just the name of every battleship sunk.

Original Schema + View:

SunkShips

 (`ShipName, Class, Type, BattleName, Date`)

Classes(class, stype, country, numGuns, bore, displacement);

Battles(bname, bdate);

Outcomes(ship, battle, result);

Ships(sname, class, launched);

Example: Accessing a View

- Another query:
  - Find just the name of every battleship sunk.

Select ShipName from SunkShips where type = 'bb';

Original Schema + View:

SunkShips
  (`ShipName, Class, Type, BattleName, Date`)

Classes(class, stype, country, numGuns, bore, displacement);

Battles(bname, bdate);

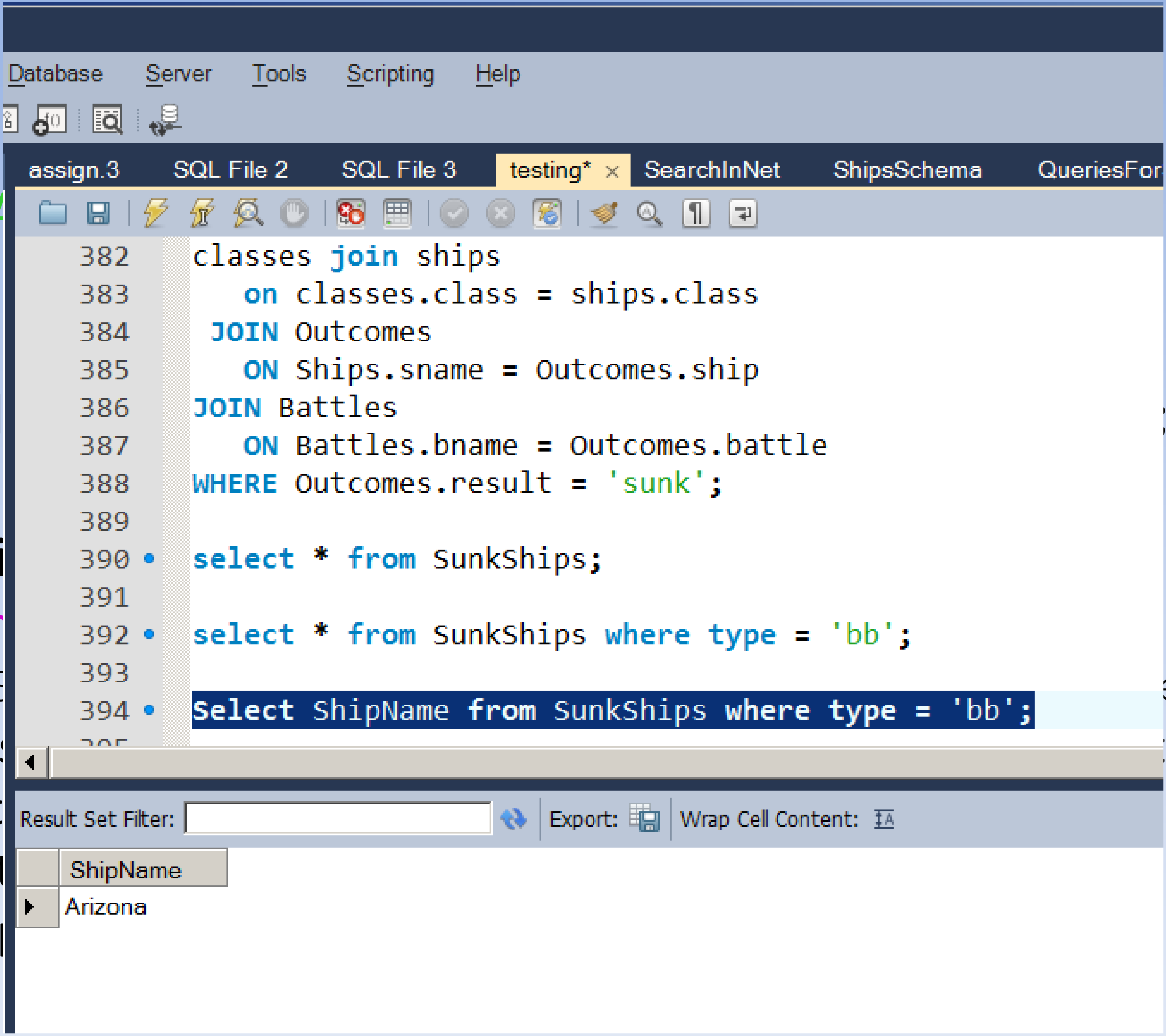
Outcomes(ship, battle, result);

Ships(sname, class, launched);

- A

Sel ;

Ori
Sur
(S e)
Clas);
Bat
Out
Shi



```
classes join ships
   on classes.class = ships.class
 JOIN Outcomes
   ON Ships.sname = Outcomes.ship
JOIN Battles
   ON Battles.bname = Outcomes.battle
WHERE Outcomes.result = 'sunk';

select * from SunkShips;

select * from SunkShips where type = 'bb';

Select ShipName from SunkShips where type = 'bb';
```

ShipName
Arizona

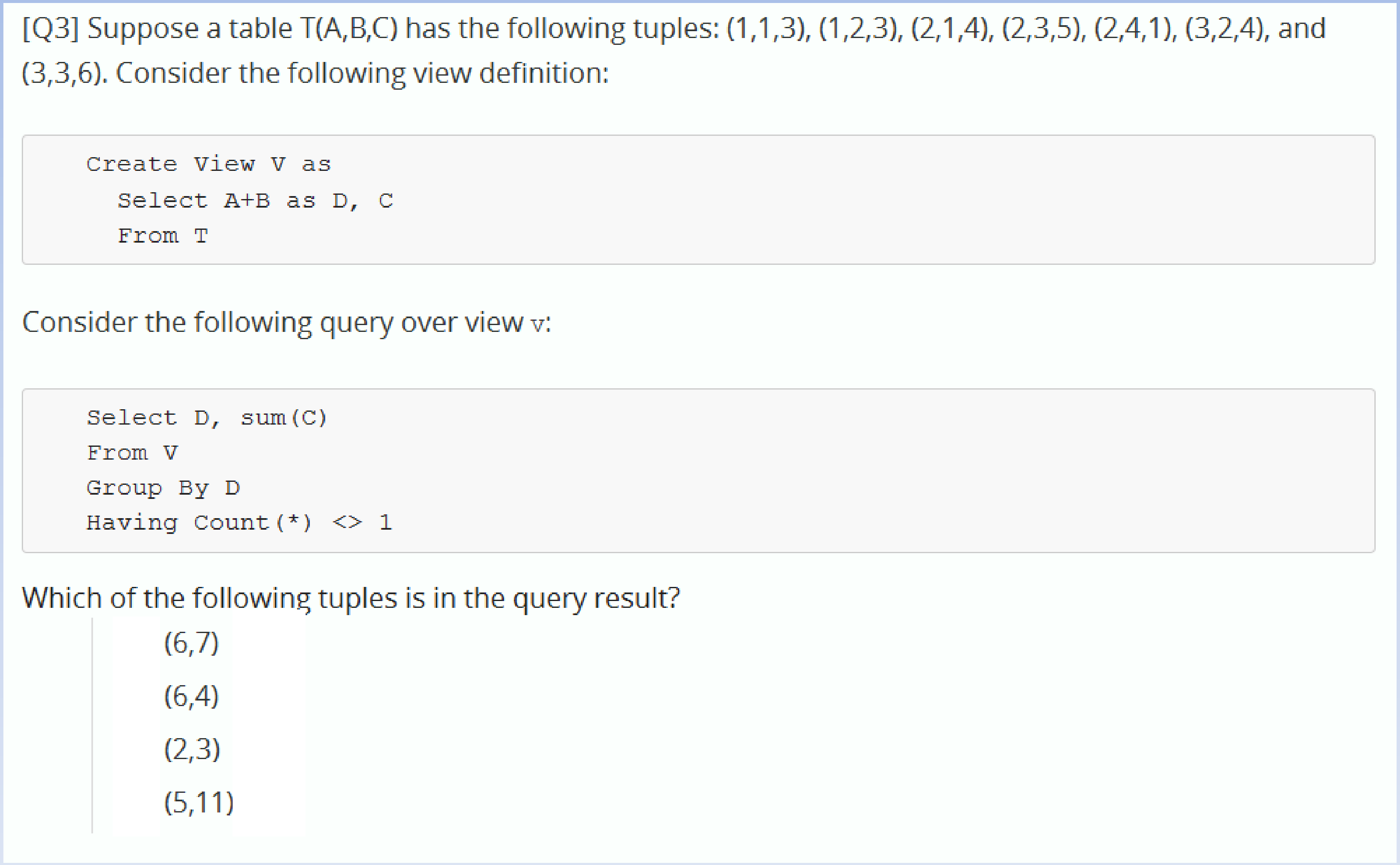
Materialized Views

- Problem: each time a base table changes, the materialized view may change.
  - Cannot afford to recompute the view with each change.
- Solution: Periodic reconstruction of the materialized view, which is otherwise “out of date.”

Example: A Data Warehouse

- Wal-Mart stores every sale at every store in a database.
- Overnight, the sales for the day are used to update a *data warehouse* = materialized views of the sales.
- The warehouse is used by analysts to predict trends and move goods to where they are selling best.

Question



[Q3] Suppose a table T(A,B,C) has the following tuples: (1,1,3), (1,2,3), (2,1,4), (2,3,5), (2,4,1), (3,2,4), and (3,3,6). Consider the following view definition:

```
Create View V as
  Select A+B as D, C
  From T
```

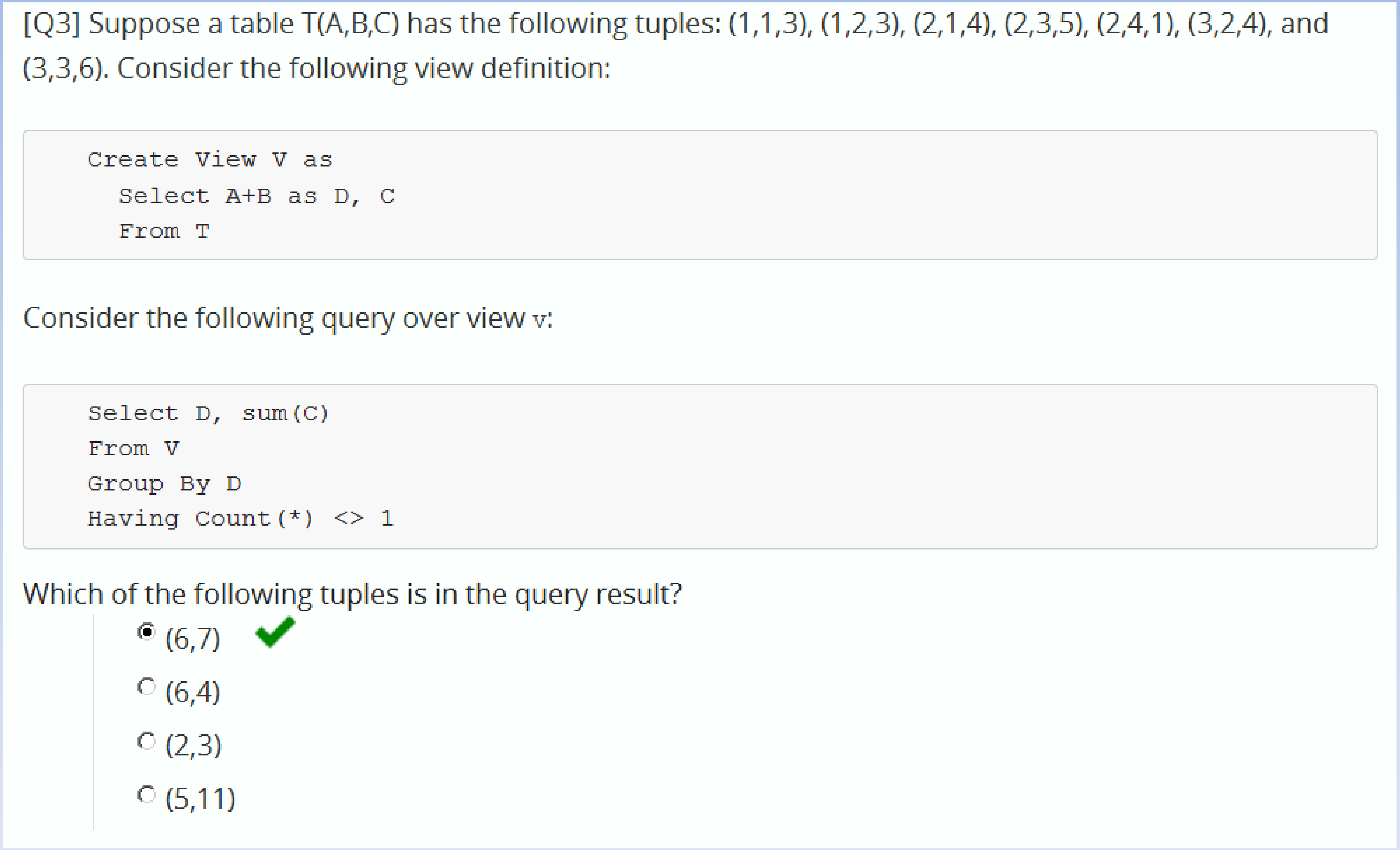
Consider the following query over view `V`:

```
Select D, sum(C)
From V
Group By D
Having Count(*) <> 1
```

Which of the following tuples is in the query result?

- (6,7)
- (6,4)
- (2,3)
- (5,11)

Question



[Q3] Suppose a table T(A,B,C) has the following tuples: (1,1,3), (1,2,3), (2,1,4), (2,3,5), (2,4,1), (3,2,4), and (3,3,6). Consider the following view definition:

```
Create View V as
  Select A+B as D, C
  From T
```

Consider the following query over view V:

```
Select D, sum(C)
From V
Group By D
Having Count(*) <> 1
```

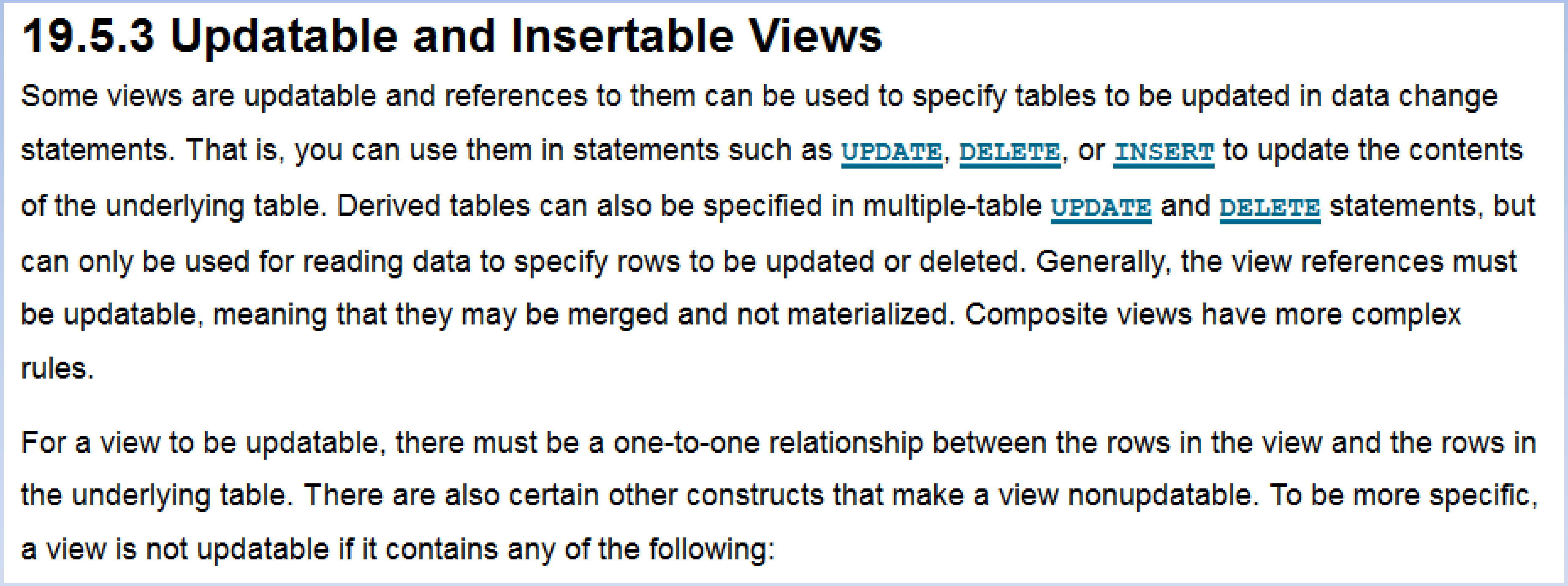
Which of the following tuples is in the query result?

- [✓ (6,7)] ✓
- (6,4)
- (2,3)
- (5,11)

Modifying Views

- Once view *V* is defined, can we modify *V* like any table **?**
  - Doesn't make sense: *V* is not stored
  - Has to make sense: views are some users' entire "view" of the database
- Solution: Modifications to *V* rewritten to modify base tables

MySQL

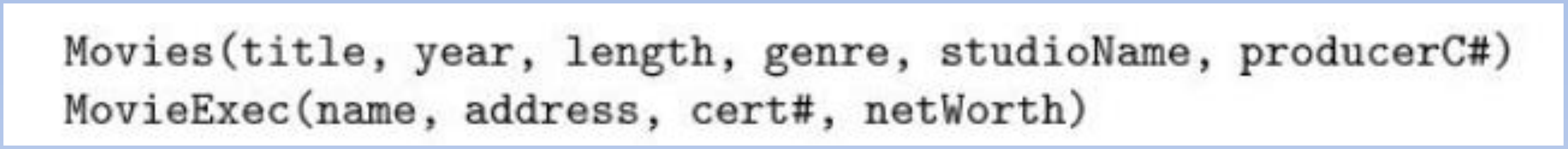


19.5.3 Updatable and Insertable Views

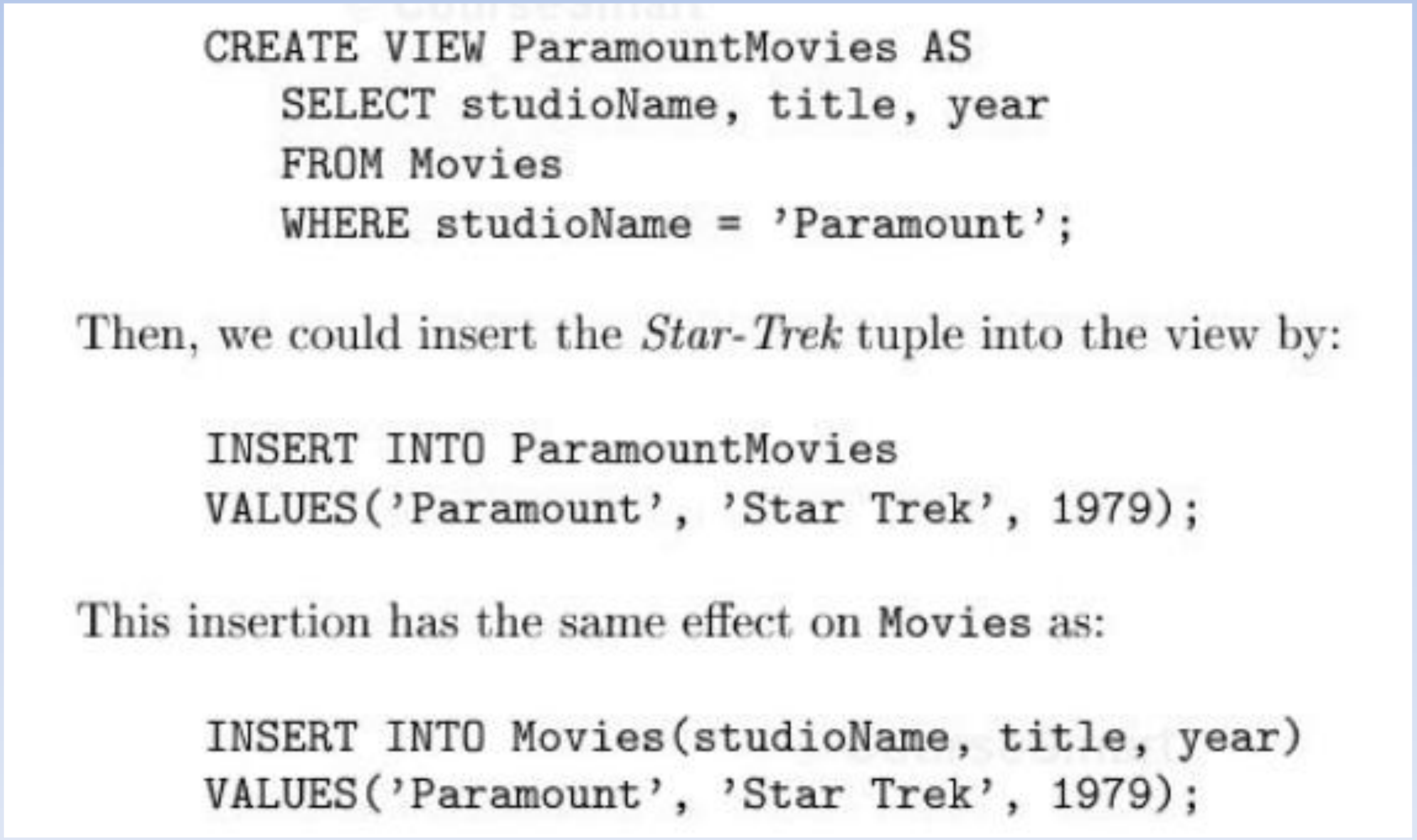
Some views are updatable and references to them can be used to specify tables to be updated in data change statements. That is, you can use them in statements such as `UPDATE`, `DELETE`, or `INSERT` to update the contents of the underlying table. Derived tables can also be specified in multiple-table `UPDATE` and `DELETE` statements, but can only be used for reading data to specify rows to be updated or deleted. Generally, the view references must be updatable, meaning that they may be merged and not materialized. Composite views have more complex rules.

For a view to be updatable, there must be a one-to-one relationship between the rows in the view and the rows in the underlying table. There are also certain other constructs that make a view nonupdatable. To be more specific, a view is not updatable if it contains any of the following:

Modifiable Views



```
Movies(title, year, length, genre, studioName, producerC#)
MovieExec(name, address, cert#, netWorth)
```



```
CREATE VIEW ParamountMovies AS
   SELECT studioName, title, year
   FROM Movies
   WHERE studioName = 'Paramount';
```

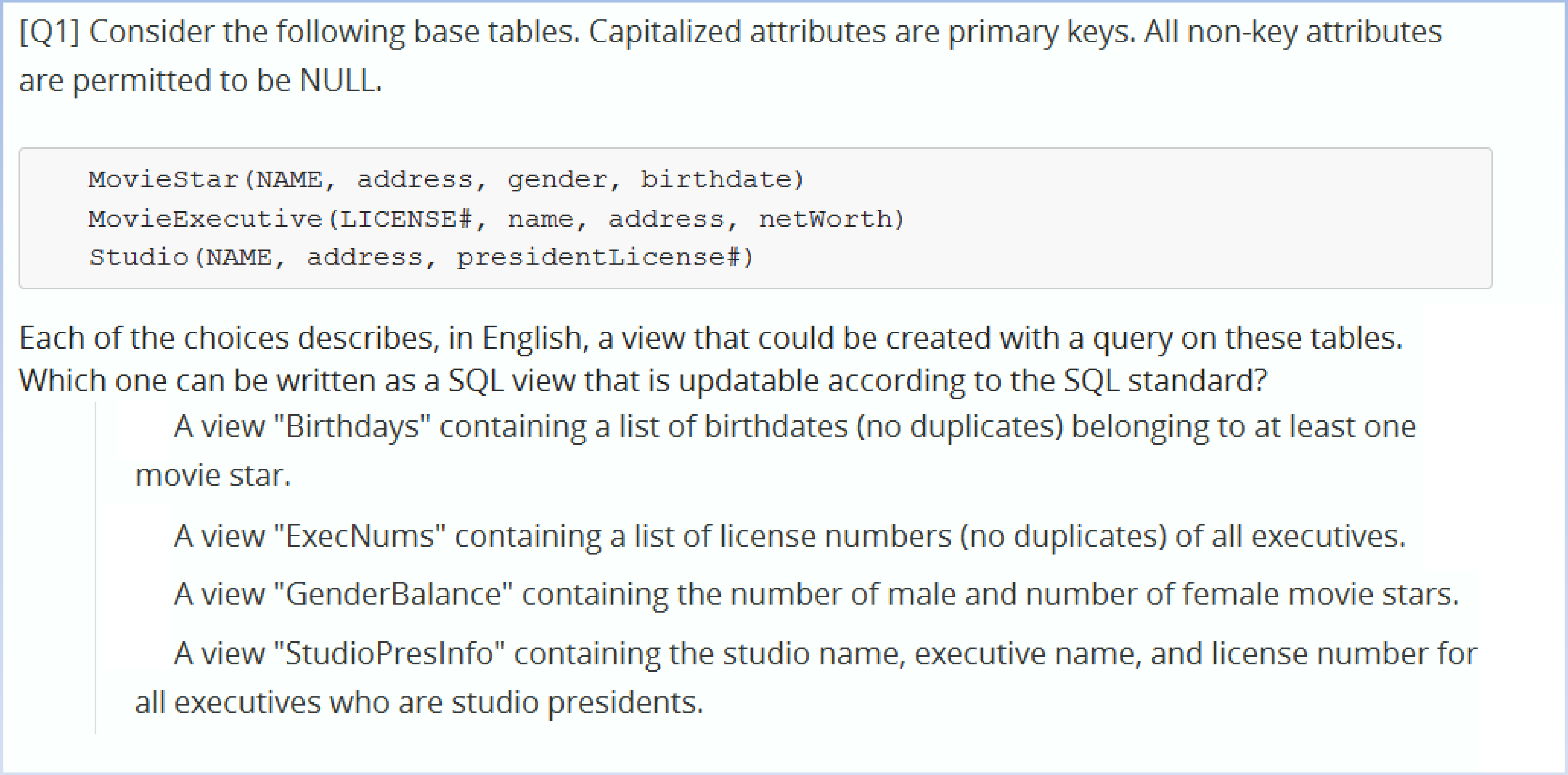
Then, we could insert the *Star-Trek* tuple into the view by:

```
INSERT INTO ParamountMovies
VALUES('Paramount', 'Star Trek', 1979);
```

This insertion has the same effect on Movies as:

```
INSERT INTO Movies(studioName, title, year)
VALUES('Paramount', 'Star Trek', 1979);
```

Question



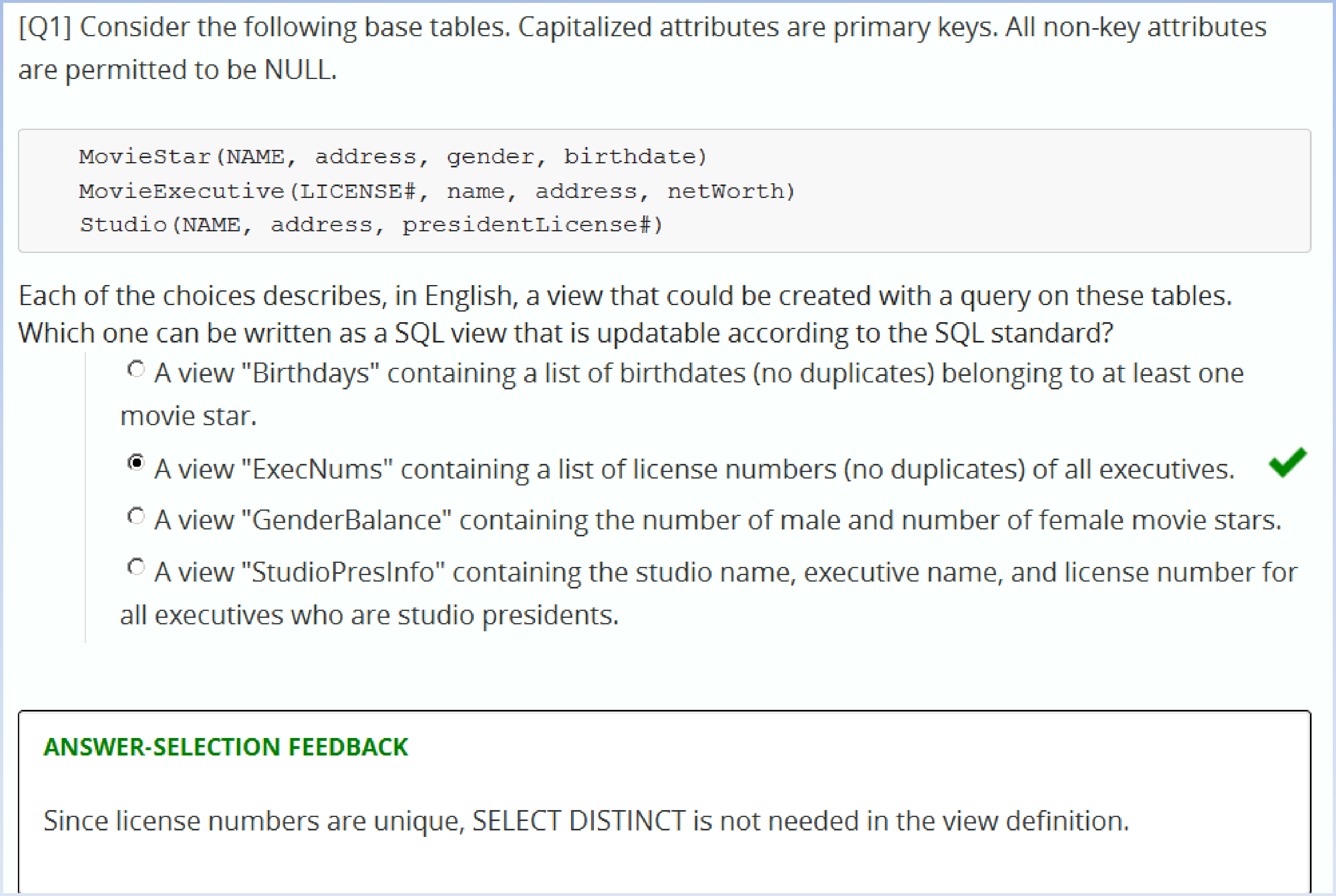
[Q1] Consider the following base tables. Capitalized attributes are primary keys. All non-key attributes are permitted to be NULL.

```
    MovieStar(NAME, address, gender, birthdate)
    MovieExecutive(LICENSE#, name, address, netWorth)
    Studio(NAME, address, presidentLicense#)
```

Each of the choices describes, in English, a view that could be created with a query on these tables. Which one can be written as a SQL view that is updatable according to the SQL standard?

> A view "Birthdays" containing a list of birthdates (no duplicates) belonging to at least one movie star.
>
> A view "ExecNums" containing a list of license numbers (no duplicates) of all executives.
>
> A view "GenderBalance" containing the number of male and number of female movie stars.
>
> A view "StudioPresInfo" containing the studio name, executive name, and license number for all executives who are studio presidents.

Question



[Q1] Consider the following base tables. Capitalized attributes are primary keys. All non-key attributes are permitted to be NULL.

```
    MovieStar(NAME, address, gender, birthdate)
    MovieExecutive(LICENSE#, name, address, netWorth)
    Studio(NAME, address, presidentLicense#)
```

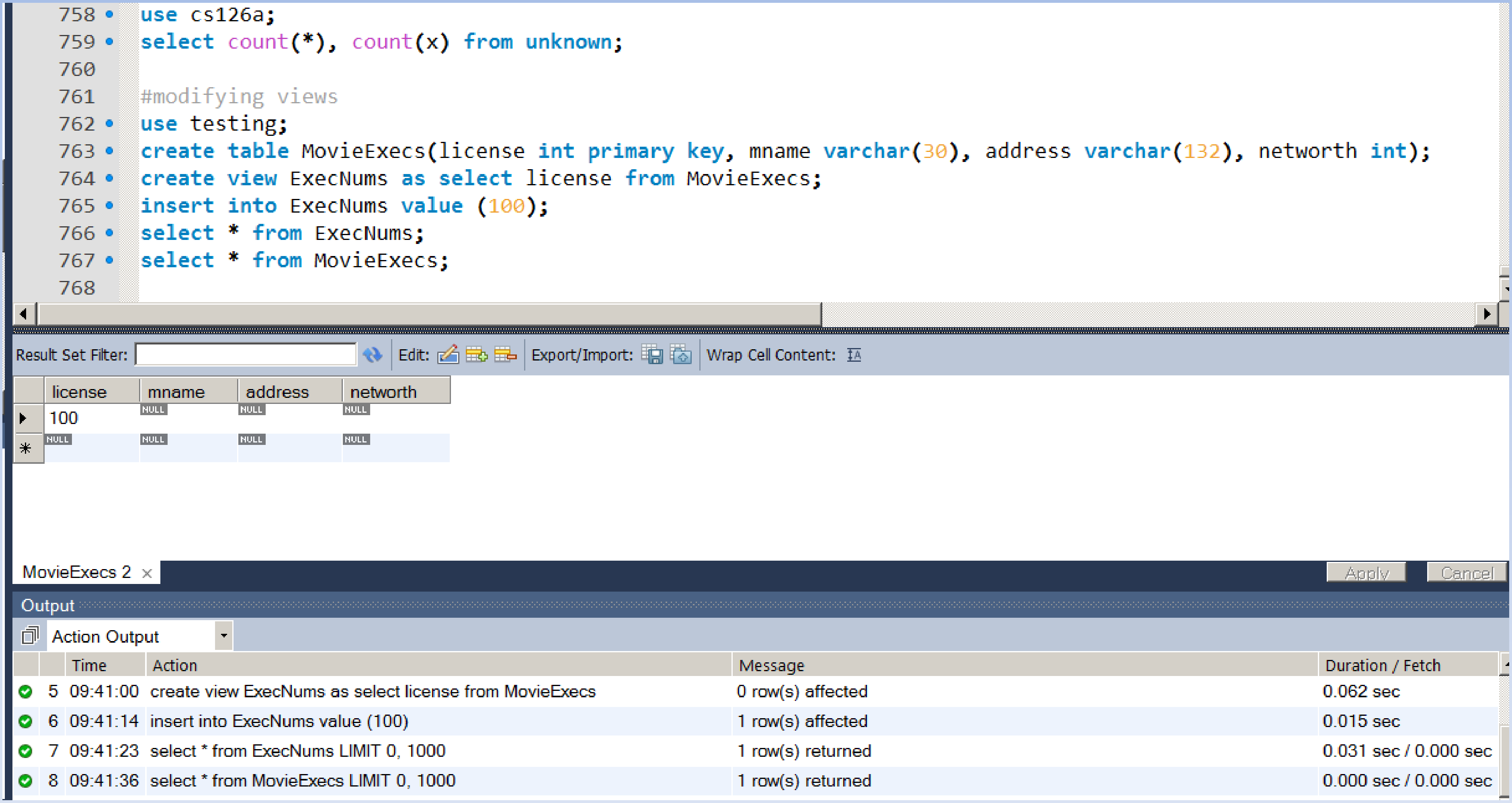
Each of the choices describes, in English, a view that could be created with a query on these tables. Which one can be written as a SQL view that is updatable according to the SQL standard?

- A view "Birthdays" containing a list of birthdates (no duplicates) belonging to at least one movie star.
- [✓ A view "ExecNums" containing a list of license numbers (no duplicates) of all executives.]
- A view "GenderBalance" containing the number of male and number of female movie stars.
- A view "StudioPresInfo" containing the studio name, executive name, and license number for all executives who are studio presidents.

ANSWER-SELECTION FEEDBACK

Since license numbers are unique, SELECT DISTINCT is not needed in the view definition.

MySQL



```
use cs126a;
select count(*), count(x) from unknown;

#modifying views
use testing;
create table MovieExecs(license int primary key, mname varchar(30), address varchar(132), networth int);
create view ExecNums as select license from MovieExecs;
insert into ExecNums value (100);
select * from ExecNums;
select * from MovieExecs;
```

Result Set Filter: Edit: Export/Import: Wrap Cell Content:

license	mname	address	networth
100	NULL	NULL	NULL
NULL	NULL	NULL	NULL

MovieExecs 2 × Apply Cancel

Output

Action Output

	Time	Action	Message	Duration / Fetch
5	09:41:00	create view ExecNums as select license from MovieExecs	0 row(s) affected	0.062 sec
6	09:41:14	insert into ExecNums value (100)	1 row(s) affected	0.015 sec
7	09:41:23	select * from ExecNums LIMIT 0, 1000	1 row(s) returned	0.031 sec / 0.000 sec
8	09:41:36	select * from MovieExecs LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec