CIND-110
Data Organization for Data Analysts

Lab Manual Module 4
DML and DDL

Lead Instructor: Dr. Tamer ABDOU

Contents

Objectives

Practice DML and DDL statements.

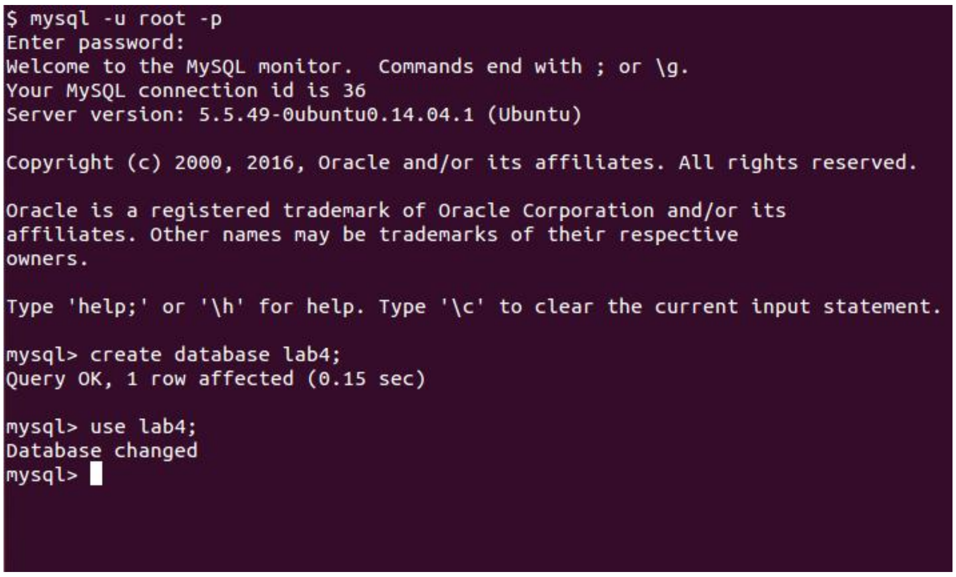
- Develop more Complex SQL Retrieval Queries.
- Schema Change Statements in SQL.
- Views (Virtual Tables) in SQL
- Specifying Constraints as Assertions and Actions as Triggers

1. CREATE and USE Database Statements

Login to mysql from your Linux (Ubuntu) terminal session, and create a database and use it.

Create an Use a Database

```
$ mysql -u root -p
mysql> create database lab4;
mysql> use lab4;
```



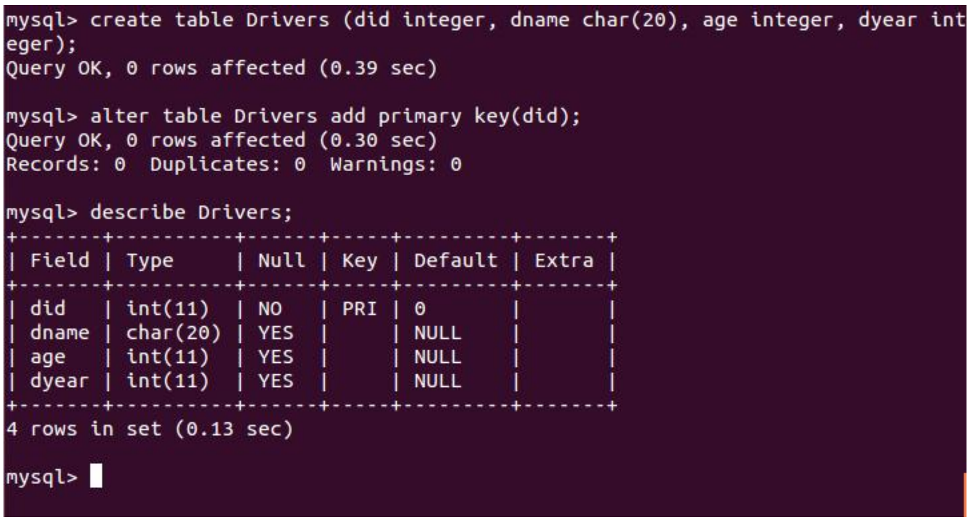
```
$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 36
Server version: 5.5.49-0ubuntu0.14.04.1 (Ubuntu)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database lab4;
Query OK, 1 row affected (0.15 sec)

mysql> use lab4;
Database changed
mysql> 
```

2. CREATE TABLE and ALTER TABLE commands

Create and alter table commands are data definition language (DDL) statements. CREATE TABLE defines a schema entity or relation in the database and ALTER TABLE subsequently modifies it. Though primary and foreign keys and constraints may be specified at the time a table is created, they may also be provided afterwards.
Create tables for a car rental scenario and practice schema change operations

Create and alter a table

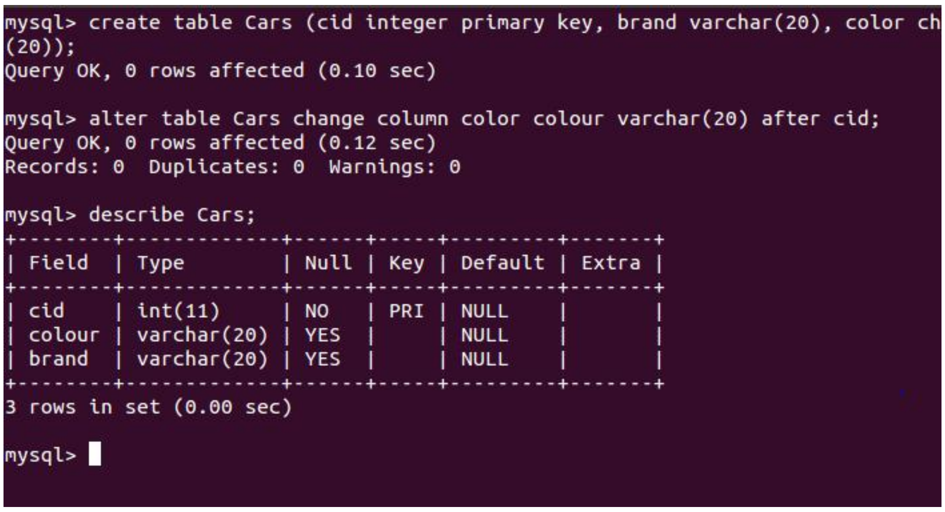
```
CREATE TABLE Drivers (did integer, dname char(20)
   , age integer, dyear integer);
ALTER TABLE Drivers ADD PRIMARY KEY(did);
DESCRIBE Drivers;
```



```
mysql> create table Drivers (did integer, dname char(20), age integer, dyear int
eger);
Query OK, 0 rows affected (0.39 sec)

mysql> alter table Drivers add primary key(did);
Query OK, 0 rows affected (0.30 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> describe Drivers;
+-------+----------+------+-----+---------+-------+
| Field | Type     | Null | Key | Default | Extra |
+-------+----------+------+-----+---------+-------+
| did   | int(11)  | NO   | PRI | 0       |       |
| dname | char(20) | YES  |     | NULL    |       |
| age   | int(11)  | YES  |     | NULL    |       |
| dyear | int(11)  | YES  |     | NULL    |       |
+-------+----------+------+-----+---------+-------+
4 rows in set (0.13 sec)

mysql> 
```

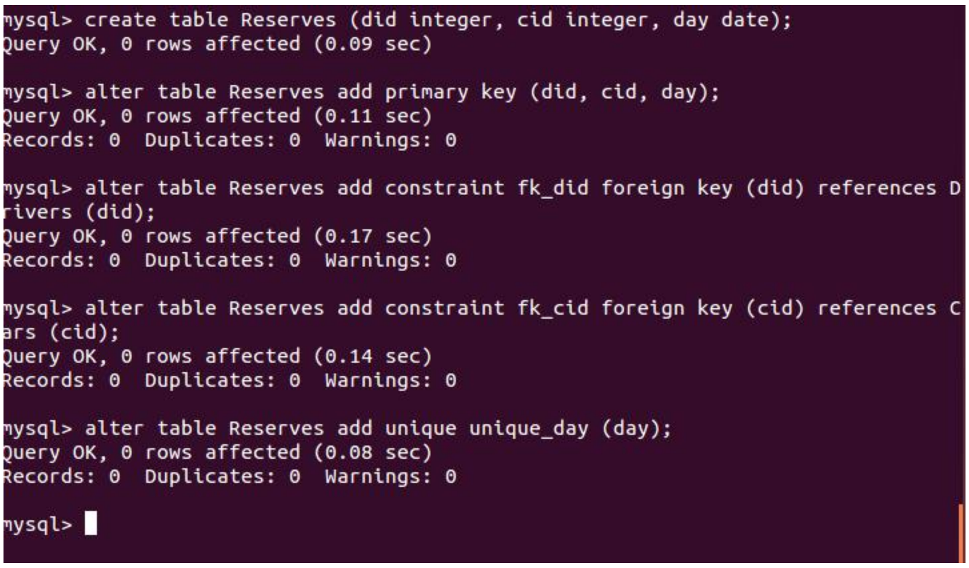
Create and alter Cars table

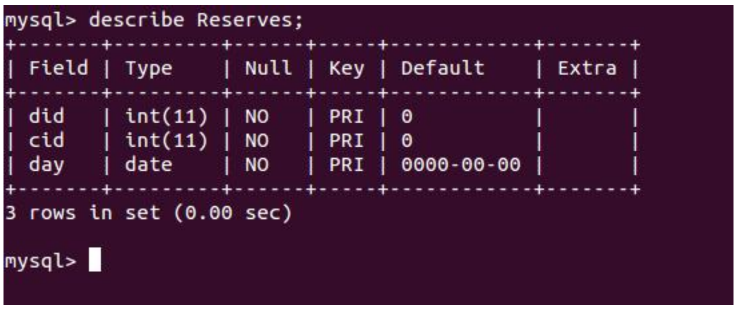
```
CREATE TABLE Cars (cid integer primary key, brand
    varchar(20), color char(20));
ALTER TABLE Cars CHANGE COLUMN color colour
   varchar(20) AFTER cid;
DESCRIBE Cars;
```


```
mysql> create table Cars (cid integer primary key, brand varchar(20), color ch
(20));
Query OK, 0 rows affected (0.10 sec)

mysql> alter table Cars change column color colour varchar(20) after cid;
Query OK, 0 rows affected (0.12 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> describe Cars;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| cid    | int(11)     | NO   | PRI | NULL    |       |
| colour | varchar(20) | YES  |     | NULL    |       |
| brand  | varchar(20) | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> 
```

Create and Alter Reserves table

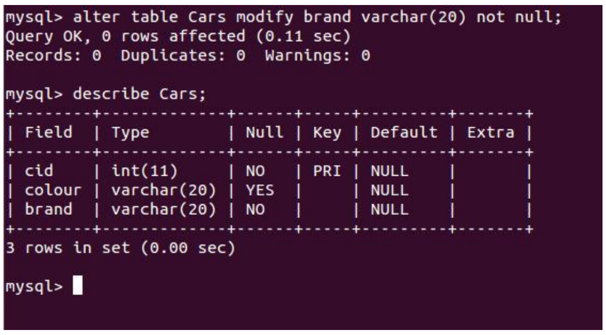
```sql
CREATE TABLE Reserves (did integer, cid integer,
   day date);
ALTER TABLE Reserves ADD PRIMARY KEY (did, cid,
   day);
ALTER TABLE Reserves ADD CONSTRAINT fk_did
   FOREIGN KEY (did) REFERENCES Drivers (did);
ALTER TABLE Reserves ADD CONSTRAINT fk_cid
   FOREIGN KEY (cid) REFERENCES Cars (cid);
ALTER TABLE Reserves ADD UNIQUE unique_day (day);
DESCRIBE Reserves;
```


```
mysql> create table Reserves (did integer, cid integer, day date);
Query OK, 0 rows affected (0.09 sec)

mysql> alter table Reserves add primary key (did, cid, day);
Query OK, 0 rows affected (0.11 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> alter table Reserves add constraint fk_did foreign key (did) references D
rivers (did);
Query OK, 0 rows affected (0.17 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> alter table Reserves add constraint fk_cid foreign key (cid) references C
ars (cid);
Query OK, 0 rows affected (0.14 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> alter table Reserves add unique unique_day (day);
Query OK, 0 rows affected (0.08 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> 
```


```
mysql> describe Reserves;
+-------+---------+------+-----+------------+-------+
| Field | Type    | Null | Key | Default    | Extra |
+-------+---------+------+-----+------------+-------+
| did   | int(11) | NO   | PRI | 0          |       |
| cid   | int(11) | NO   | PRI | 0          |       |
| day   | date    | NO   | PRI | 0000-00-00 |       |
+-------+---------+------+-----+------------+-------+
3 rows in set (0.00 sec)

mysql> 
```

Alter Cars table

```
ALTER TABLE Cars MODIFY brand varchar(20) NOT
   NULL;
DESCRIBE Cars;
```

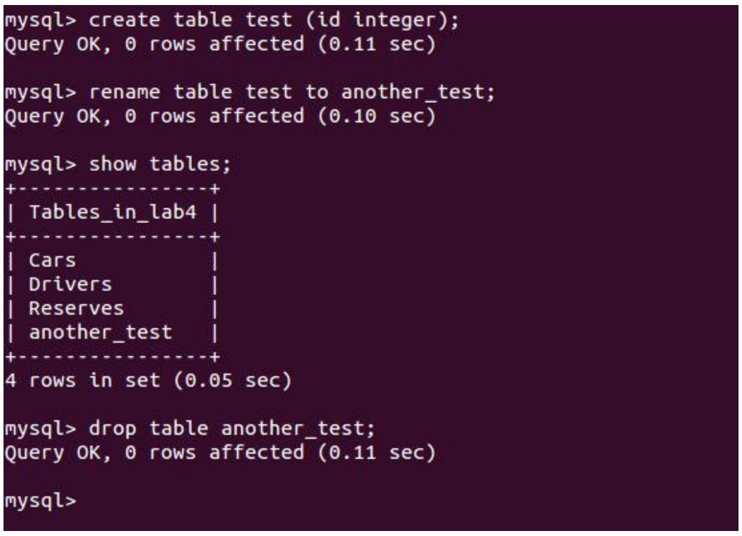

```
mysql> alter table Cars modify brand varchar(20) not null;
Query OK, 0 rows affected (0.11 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> describe Cars;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| cid    | int(11)     | NO   | PRI | NULL    |       |
| colour | varchar(20) | YES  |     | NULL    |       |
| brand  | varchar(20) | NO   |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> 
```

3. RENAME TABLE and DROP TABLE

Tables cab be renamed with the RENAME TABLE command. The DDL below creates and renames a table and then drops it from the database.

Rename and Drop a table

```sql
CREATE TABLE test (id integer);
RENAME TABLE test TO another_test;
SHOW TABLES;
DROP TABLE another_test;
```

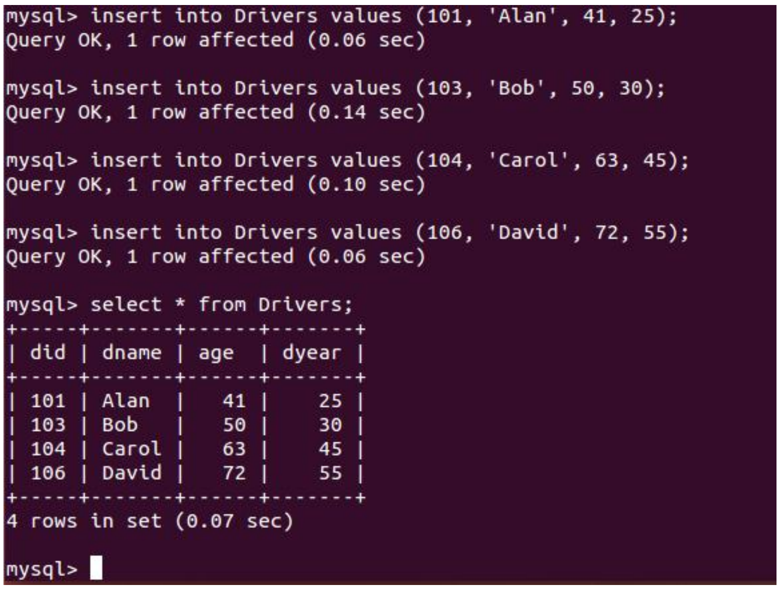


```
mysql> create table test (id integer);
Query OK, 0 rows affected (0.11 sec)

mysql> rename table test to another_test;
Query OK, 0 rows affected (0.10 sec)

mysql> show tables;
+----------------+
| Tables_in_lab4 |
+----------------+
| Cars           |
| Drivers        |
| Reserves       |
| another_test   |
+----------------+
4 rows in set (0.05 sec)

mysql> drop table another_test;
Query OK, 0 rows affected (0.11 sec)

mysql>
```

Populate tables for car rental scenario and practice select operations

Insert into Drivers table

```sql
INSERT INTO Drivers VALUES (101, 'Alan', 41, 25);
INSERT INTO Drivers VALUES (103, 'Bob', 50, 30);
INSERT INTO Drivers VALUES (104, 'Carol', 63, 45)
    ;
INSERT INTO Drivers VALUES (106, 'David', 72, 55)
    ;
SELECT * FROM Drivers;
```



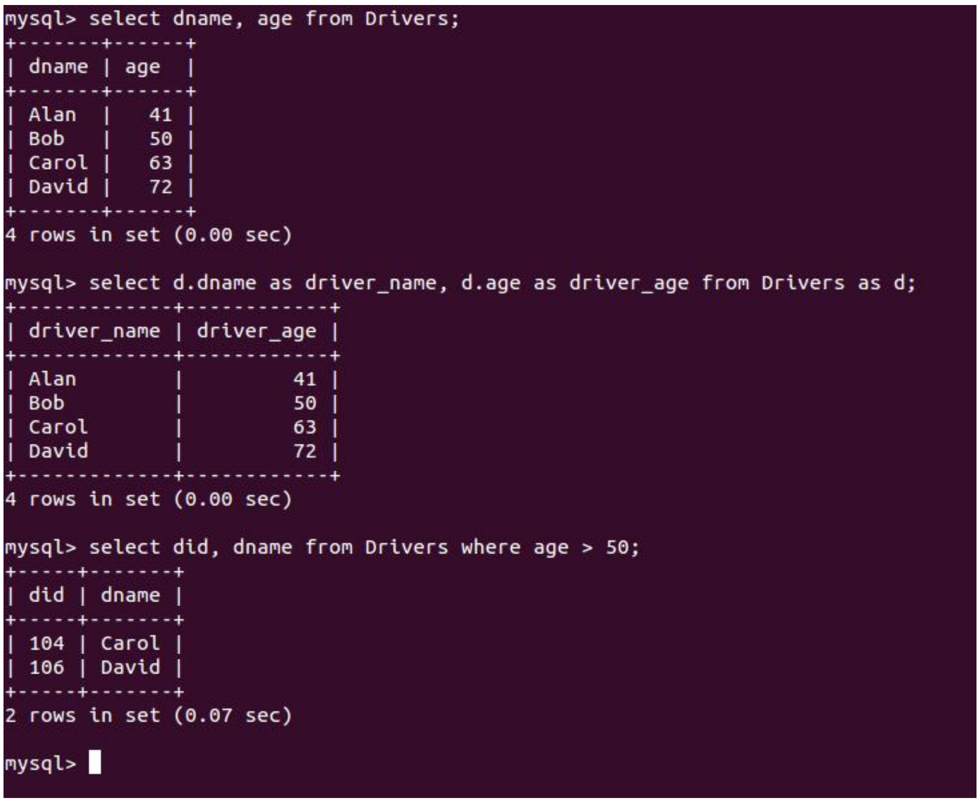
```
mysql> insert into Drivers values (101, 'Alan', 41, 25);
Query OK, 1 row affected (0.06 sec)

mysql> insert into Drivers values (103, 'Bob', 50, 30);
Query OK, 1 row affected (0.14 sec)

mysql> insert into Drivers values (104, 'Carol', 63, 45);
Query OK, 1 row affected (0.10 sec)

mysql> insert into Drivers values (106, 'David', 72, 55);
Query OK, 1 row affected (0.06 sec)

mysql> select * from Drivers;
+-----+-------+------+-------+
| did | dname | age  | dyear |
+-----+-------+------+-------+
| 101 | Alan  |   41 |    25 |
| 103 | Bob   |   50 |    30 |
| 104 | Carol |   63 |    45 |
| 106 | David |   72 |    55 |
+-----+-------+------+-------+
4 rows in set (0.07 sec)

mysql> 
```

4. SELECT statement aliases and WHERE clause

A SELECT statement with a * will retrieve all the attributes from a table. When only specific attributes of the relation are of interest, they may be directly enumerated. The AS keyword allows columns (attributes) and tables to be referred to by aliases. Tables are often aliased for readability in an SQL statement and are very useful when multiple tables are involved in a query. The WHERE clause filters which rows are returned on a SELECT statements by requiring that they meet a supplied condition.

Select statement and Where clause

```
SELECT dname , age FROM Drivers;
SELECT d.dname AS driver_name , d.age AS
   driver_age FROM Drivers AS d;
SELECT did , dname FROM Drivers WHERE age > 50;
```



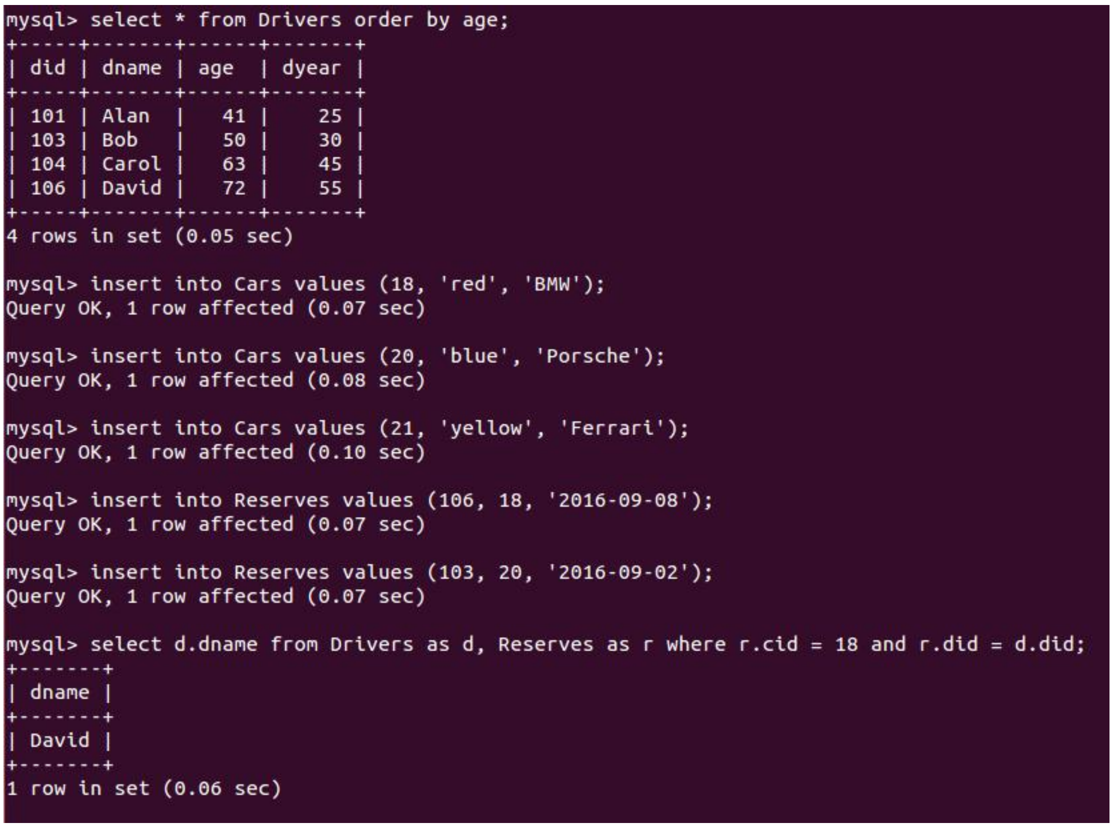
```
mysql> select dname, age from Drivers;
+-------+------+
| dname | age  |
+-------+------+
| Alan  |   41 |
| Bob   |   50 |
| Carol |   63 |
| David |   72 |
+-------+------+
4 rows in set (0.00 sec)

mysql> select d.dname as driver_name, d.age as driver_age from Drivers as d;
+-------------+------------+
| driver_name | driver_age |
+-------------+------------+
| Alan        |         41 |
| Bob         |         50 |
| Carol       |         63 |
| David       |         72 |
+-------------+------------+
4 rows in set (0.00 sec)

mysql> select did, dname from Drivers where age > 50;
+-----+-------+
| did | dname |
+-----+-------+
| 104 | Carol |
| 106 | David |
+-----+-------+
2 rows in set (0.07 sec)

mysql> 
```

5. SELECT statement ORDER BY

The ORDER BY clause instructs the DBMS to sort the rows returned by a query according to a supplied attribute or column.
Practice ORDER BY clause and insert below rows in Cars and Reserves.

Inser into Cars and Reserves table

```
SELECT * FROM Drivers ORDER BY age;
INSERT INTO Cars VALUES (18, 'red', 'BMW');
INSERT INTO Cars VALUES (20, 'blue', 'Porsche');
INSERT INTO Cars VALUES (21, 'yellow', 'Ferrari')
    ;
INSERT INTO Reserves VALUES (106, 18, '2016-09-08
    ');
INSERT INTO Reserves VALUES (103, 20, '2016-09-02
    ');
```



```
mysql> select * from Drivers order by age;
+-----+-------+------+-------+
| did | dname | age  | dyear |
+-----+-------+------+-------+
| 101 | Alan  |   41 |    25 |
| 103 | Bob   |   50 |    30 |
| 104 | Carol |   63 |    45 |
| 106 | David |   72 |    55 |
+-----+-------+------+-------+
4 rows in set (0.05 sec)

mysql> insert into Cars values (18, 'red', 'BMW');
Query OK, 1 row affected (0.07 sec)

mysql> insert into Cars values (20, 'blue', 'Porsche');
Query OK, 1 row affected (0.08 sec)

mysql> insert into Cars values (21, 'yellow', 'Ferrari');
Query OK, 1 row affected (0.10 sec)

mysql> insert into Reserves values (106, 18, '2016-09-08');
Query OK, 1 row affected (0.07 sec)

mysql> insert into Reserves values (103, 20, '2016-09-02');
Query OK, 1 row affected (0.07 sec)

mysql> select d.dname from Drivers as d, Reserves as r where r.cid = 18 and r.did = d.did;
+-------+
| dname |
+-------+
| David |
+-------+
1 row in set (0.06 sec)
```

6. SQL Join, Group By, and Sub-query Statements

INNER JOIN operation Often, we wish to join two or more relations having a comparable attribute before retrieving rows. The join conditions may be supplied either as part of an INNER JOIN clause or as part of the WHERE clause when multiple tables are supplied in the FROM table list.

Inner JOIN

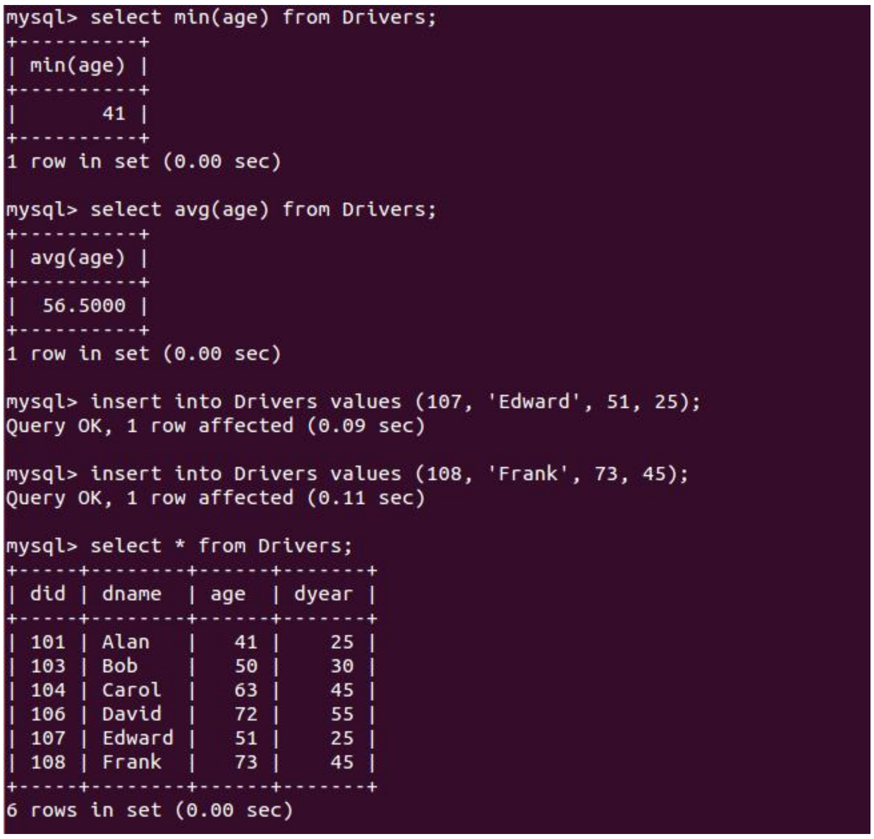
```sql
SELECT d.dname FROM Drivers AS d, Reserves AS r
   WHERE r.cid = 18 AND r.did = d.did;
```

7. SELECT statement aggregation functions

Various aggregation functions may be used in the SELECT-list to obtain a single value derived from among the set of attributes in a relation. For example, the minimum value can be obtained with the min() function and the average value can be obtained with the avg() function.

min and avg functions

```sql
SELECT min(age) FROM Drivers;
SELECT avg(age) FROM Drivers;
INSERT INTO Drivers VALUES (107, 'Edward', 51,
   25);
INSERT INTO Drivers VALUES (108, 'Frank', 73, 45)
   ;
SELECT * FROM Drivers;
```



```
mysql> select min(age) from Drivers;
+----------+
| min(age) |
+----------+
|       41 |
+----------+
1 row in set (0.00 sec)

mysql> select avg(age) from Drivers;
+----------+
| avg(age) |
+----------+
|  56.5000 |
+----------+
1 row in set (0.00 sec)

mysql> insert into Drivers values (107, 'Edward', 51, 25);
Query OK, 1 row affected (0.09 sec)

mysql> insert into Drivers values (108, 'Frank', 73, 45);
Query OK, 1 row affected (0.11 sec)

mysql> select * from Drivers;
+-----+--------+------+-------+
| did | dname  | age  | dyear |
+-----+--------+------+-------+
| 101 | Alan   |   41 |    25 |
| 103 | Bob    |   50 |    30 |
| 104 | Carol  |   63 |    45 |
| 106 | David  |   72 |    55 |
| 107 | Edward |   51 |    25 |
| 108 | Frank  |   73 |    45 |
+-----+--------+------+-------+
6 rows in set (0.00 sec)
```

8. SELECT statement GROUP BY and HAVING clauses

Often, we will group the rows of a relation by a supplied attribute and then apply an aggregation function to the rows where the attribute has the same value. The HAVING clause is optionally used with the GROUP BY clause to filter the results by a given condition.

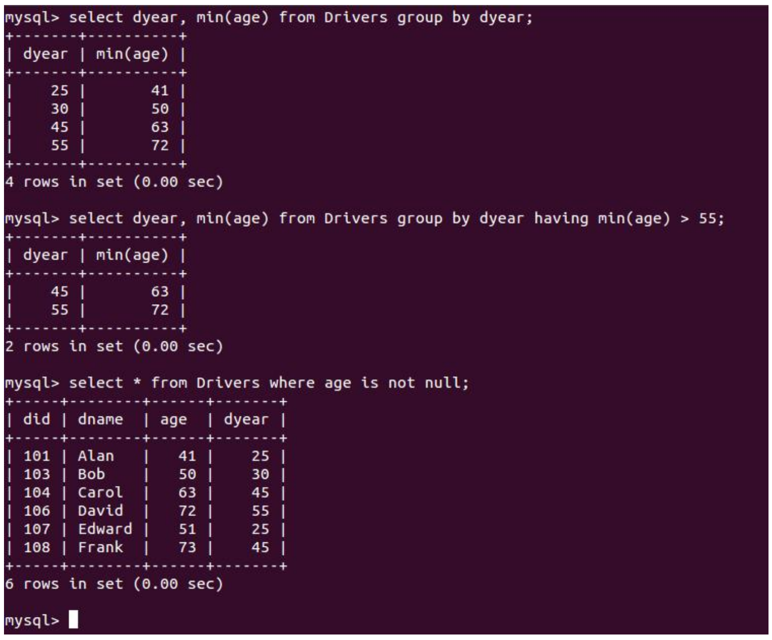
GROUP BY and HAVING clause

```
SELECT dyear, min(age) FROM Drivers GROUP BY
   dyear;
SELECT dyear, min(age) FROM Drivers GROUP BY
   dyear HAVING min(age) > 55;
```

SELECT statement NULL attribute check It is possible to limit the rows returned by a query by requiring that certain attributes are or are not NULL. The check for a NULL attribute is supplied in the WHERE clause of a SELECT statement.
Practice group by, having, and sub-query operations.

Checking NOT NULL attribute

```
SELECT * FROM Drivers WHERE age IS NOT NULL;
```



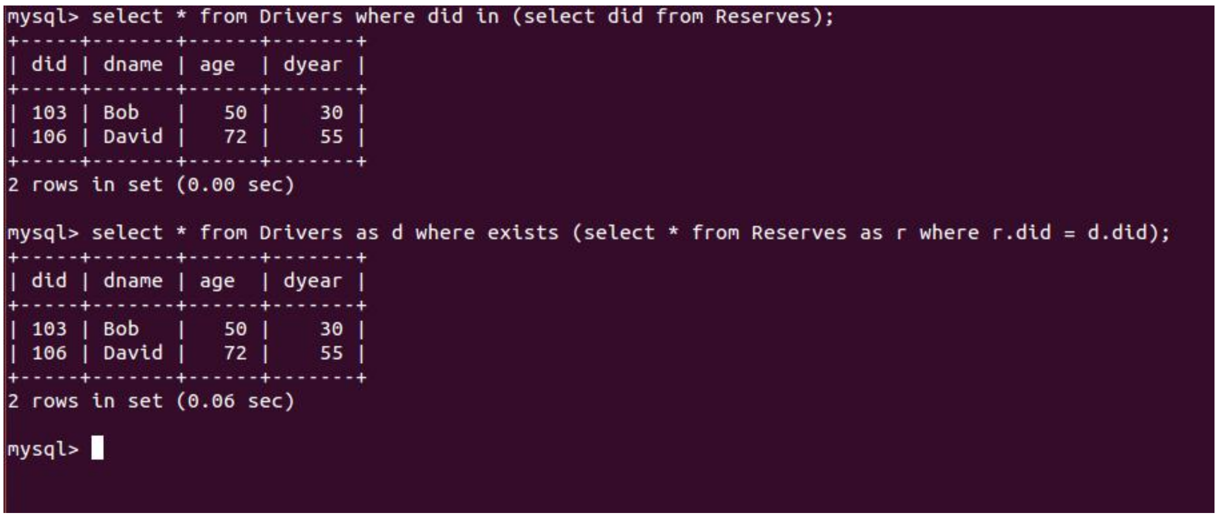
```
mysql> select dyear, min(age) from Drivers group by dyear;
+-------+----------+
| dyear | min(age) |
+-------+----------+
|    25 |       41 |
|    30 |       50 |
|    45 |       63 |
|    55 |       72 |
+-------+----------+
4 rows in set (0.00 sec)

mysql> select dyear, min(age) from Drivers group by dyear having min(age) > 55;
+-------+----------+
| dyear | min(age) |
+-------+----------+
|    45 |       63 |
|    55 |       72 |
+-------+----------+
2 rows in set (0.00 sec)

mysql> select * from Drivers where age is not null;
+-----+--------+------+-------+
| did | dname  | age  | dyear |
+-----+--------+------+-------+
| 101 | Alan   |   41 |    25 |
| 103 | Bob    |   50 |    30 |
| 104 | Carol  |   63 |    45 |
| 106 | David  |   72 |    55 |
| 107 | Edward |   51 |    25 |
| 108 | Frank  |   73 |    45 |
+-----+--------+------+-------+
6 rows in set (0.00 sec)

mysql> 
```

9. SELECT statement sub-queries

A sub-query is a select statement within a select statement. They may appear in the FROM clause or the WHERE clause. With the IN keyword the results of the sub-query can be compared with an outer attribute. The EXISTS condition ensures that rows are returned in the outer query only when rows are present in the sub-query. The example queries below accomplish the same result.

SUB-QUERIES

```
SELECT * FROM Drivers WHERE did IN (SELECT did
   FROM Reserves);
SELECT * FROM Drivers AS d WHERE EXISTS (SELECT *
    FROM Reserves AS r WHERE
r.did = d.did);
```



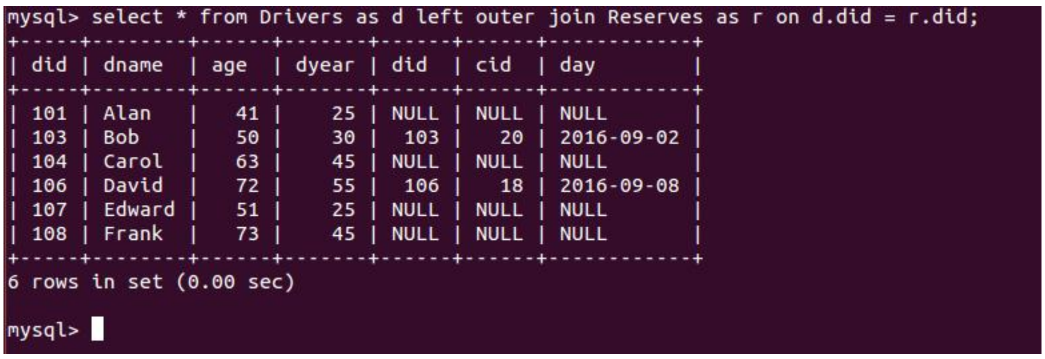
```
mysql> select * from Drivers where did in (select did from Reserves);
+-----+-------+------+-------+
| did | dname | age  | dyear |
+-----+-------+------+-------+
| 103 | Bob   |   50 |    30 |
| 106 | David |   72 |    55 |
+-----+-------+------+-------+
2 rows in set (0.00 sec)

mysql> select * from Drivers as d where exists (select * from Reserves as r where r.did = d.did);
+-----+-------+------+-------+
| did | dname | age  | dyear |
+-----+-------+------+-------+
| 103 | Bob   |   50 |    30 |
| 106 | David |   72 |    55 |
+-----+-------+------+-------+
2 rows in set (0.06 sec)

mysql> 
```

10. LEFT OUTER JOIN operation

With an INNER JOIN, only the rows from the joined relations that satisfy the join condition are returned. With a LEFT OUTER JOIN, all the rows the left table are returned, with the attributes of the rows from the right table not matching the condition filled in with NULL.
Practice left outer join statement

LEFT OUTER JOIN

```sql
SELECT * FROM Drivers AS d LEFT OUTER JOIN
    Reserves AS r ON d.did = r.did;
```



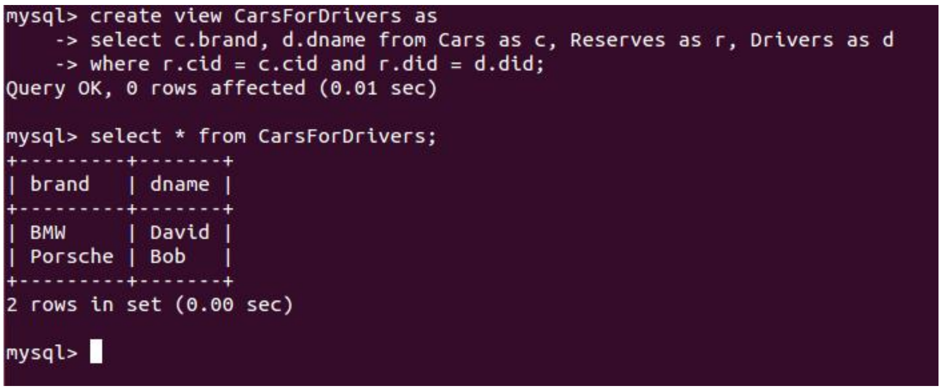
```
mysql> select * from Drivers as d left outer join Reserves as r on d.did = r.did;
+-----+--------+------+-------+------+------+------------+
| did | dname  | age  | dyear | did  | cid  | day        |
+-----+--------+------+-------+------+------+------------+
| 101 | Alan   |   41 |    25 | NULL | NULL | NULL       |
| 103 | Bob    |   50 |    30 |  103 |   20 | 2016-09-02 |
| 104 | Carol  |   63 |    45 | NULL | NULL | NULL       |
| 106 | David  |   72 |    55 |  106 |   18 | 2016-09-08 |
| 107 | Edward |   51 |    25 | NULL | NULL | NULL       |
| 108 | Frank  |   73 |    45 | NULL | NULL | NULL       |
+-----+--------+------+-------+------+------+------------+
6 rows in set (0.00 sec)

mysql> 
```

11. Views and Triggers

CREATE VIEW command When a SELECT statement is used frequently it can be codified with the CREATE VIEW statement. The rows of the SELECT statement can then be returned by selecting from the view. The behavior of views and triggers varies by DBMS. The data represented in the view may or may not exist as materialized rows in the DBMS and view updates are similarly implementation dependent.
Practice create view statement.

Create VIEW

```sql
CREATE VIEW CarsForDrivers AS SELECT c.brand, d.
   dname FROM Cars AS c, Reserves AS r, Drivers AS
    d WHERE r.cid = c.cid AND r.did = d.did;
SELECT * FROM CarsForDrivers;
```



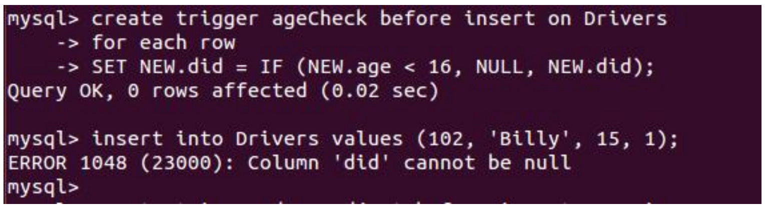
```
mysql> create view CarsForDrivers as
    -> select c.brand, d.dname from Cars as c, Reserves as r, Drivers as d
    -> where r.cid = c.cid and r.did = d.did;
Query OK, 0 rows affected (0.01 sec)

mysql> select * from CarsForDrivers;
+---------+-------+
| brand   | dname |
+---------+-------+
| BMW     | David |
| Porsche | Bob   |
+---------+-------+
2 rows in set (0.00 sec)

mysql> 
```

12. CREATE TRIGGER and DROP TRIGGER

CREATE TRIGGER command The CREATE TRIGGER statement specifies an action to take before or after the occurrence of an insert, update, or delete on a table in the database. In the first example below, if a new driver is under 16, the id of the new driver row is set to NULL so that the insert statement fails. In the second example, the number of years of driving is adjusted prior to inserting a new record so that it is no more than the driver's age minus 16.
Practice create trigger statements.

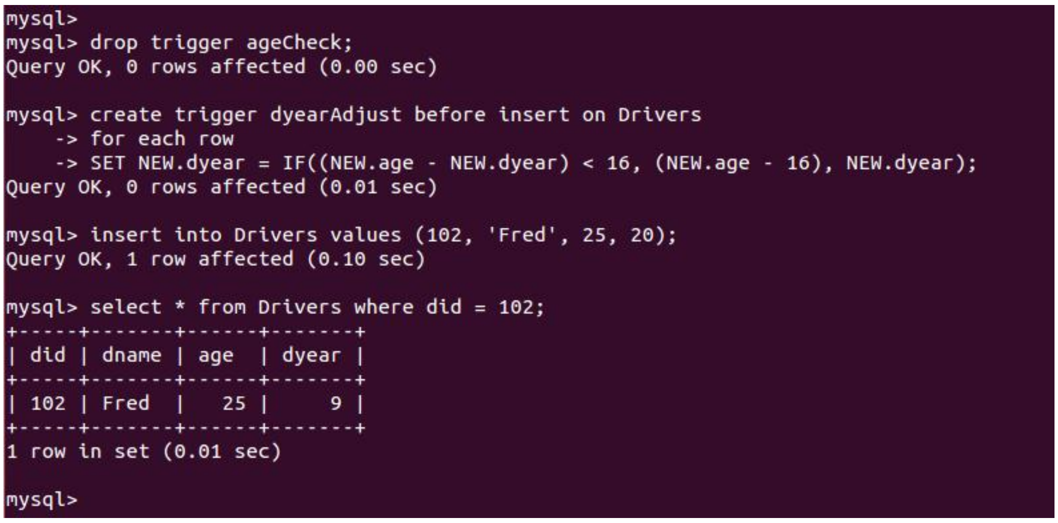
Create trigger

```
CREATE TRIGGER ageCheck BEFORE INSERT ON Drivers
   FOR EACH ROW SET NEW.did = IF (NEW.age < 16,
   NULL, NEW.did);
INSERT INTO Drivers VALUES (102, 'Billy', 15, 1);
```



```
mysql> create trigger ageCheck before insert on Drivers
    -> for each row
    -> SET NEW.did = IF (NEW.age < 16, NULL, NEW.did);
Query OK, 0 rows affected (0.02 sec)

mysql> insert into Drivers values (102, 'Billy', 15, 1);
ERROR 1048 (23000): Column 'did' cannot be null
mysql>
```

```
DROP TRIGGER ageCheck;
CREATE TRIGGER dyearAdjust BEFORE INSERT ON
   Drivers FOR EACH ROW SET NEW.dyear = IF((NEW.
   age - NEW.dyear) < 16, (NEW.age - 16), NEW.
   dyear);
INSERT INTO Drivers VALUES (102, 'Fred', 25, 20);
SELECT * FROM Drivers WHERE did = 102;
```



```
mysql>
mysql> drop trigger ageCheck;
Query OK, 0 rows affected (0.00 sec)

mysql> create trigger dyearAdjust before insert on Drivers
    -> for each row
    -> SET NEW.dyear = IF((NEW.age - NEW.dyear) < 16, (NEW.age - 16), NEW.dyear);
Query OK, 0 rows affected (0.01 sec)

mysql> insert into Drivers values (102, 'Fred', 25, 20);
Query OK, 1 row affected (0.10 sec)

mysql> select * from Drivers where did = 102;
+-----+-------+------+-------+
| did | dname | age  | dyear |
+-----+-------+------+-------+
| 102 | Fred  |   25 |     9 |
+-----+-------+------+-------+
1 row in set (0.01 sec)

mysql>
```