What is the result?

```
SELECT    SSN, 0
FROM      Employee E
WHERE     FName = 'John';
```

E

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

Review of Last Lecture

- Data definition
  - Create table
    - Domain, FK, PK and other IC
    - NOT NUL, UNIQUE, Check
  - Drop and alter
- Data modification
  - Delete, Insert, Update
  - 2 step process

In-class Exercise #1

- Find the names of employees who have dependent(s) and a cumulative > 100 total hours on projects.

```
SELECT      FName, LName
FROM        Employee
WHERE       SSN IN (
```

Subquery1

INTERSECT

Subquery2

);

In-class Exercise #1 – Subquery1

- Find the names of employees who have dependent(s) and cumulative > 100 total hours on projects.
  - Employees with dependent

```
/* Subquery1 */
SELECT      ESSN
FROM        Dependent
```

DEPENDENT

ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP
333445555	Alice	F	1986-04-05	DAUGHTER
333445555	Theodore	M	1983-10-25	SON
333445555	Joy	F	1958-05-03	SPOUSE
987654321	Abner	M	1942-02-28	SPOUSE
123456789	Michael	M	1988-01-04	SON
123456789	Alice	F	1988-12-30	DAUGHTER
123456789	Elizabeth	F	1967-05-05	SPOUSE

In-class Exercise #1 – Subquery2

- Find the names of employees who have dependent(s) and cumulative > 100 total hours on projects.
  - Employees with > 100 total hours on projects.

```
/* Subquery2 */
SELECT      ESSN
FROM        Works_ON
GROUP BY    ESSN
HAVING      SUM(Hours) > 100
```

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

Today's Plan

- Advanced SQL
  - View
  - Assertion
  - Trigger
- SQL Exercises

PART VI ADVANCED TOPICS

Views

- A **view** is a “virtual table,” a relation that is defined in terms of the contents of other tables and views.
- Declare by:

  CREATE VIEW <name> AS <query>;

- In contrast, a relation whose value is really stored in the database is called a **base table**.

SQL Views: An Example

- Specify a different WORKS_ON table

```
CREATE VIEW WORKS_ON_NEW AS
     SELECT     FNAME, LNAME, PNAME, HOURS
     FROM       EMPLOYEE, PROJECT, WORKS_ON
     WHERE      SSN=ESSN AND PNO=PNUMBER
     GROUP BY   PNAME;
```

Accessing a View

- We can specify SQL queries on a newly create table (view):

```
SELECT    FNAME, LNAME
FROM      WORKS_ON_NEW
WHERE     PNAME='Seena';
```

- When no longer needed, a view can be dropped:

```
DROP WORKS_ON_NEW;
```

- In SQLite, use DROP VIEW instead.

```
drop view WORKS_ON_NEW;
```

What Happens When a View Is Accessed?

- The DBMS replaces the view with its query definition.
  - Typical DBMS turns the query into relational algebra.
- The queries defining any views used by the query are also replaced by their algebraic equivalents, and “spliced into” the expression tree for the query.

```
SELECT    FNAME, LNAME
FROM      (SELECT    FNAME, LNAME, PNAME, HOURS
          FROM       EMPLOYEE, PROJECT, WORKS_ON
          WHERE      SSN=ESSN AND PNO=PNUMBER
                     GROUP BY   PNAME)

WHERE     PNAME=‘Seena’;
```

Why View?

- To hide data from users
- To hide complexity from users
- Logical data independence
  - If applications deal with views, we can change the underlying schema without affecting applications
  - Recall physical data independence: change the physical organization of data without affecting applications
- Real database applications use tons of views

Student(SID, FName, LName, GPA, Program, E-mail, BDate)

Scenario: Same Data Different Displays

- People have their own display preference
  - CEO wants to see employees with their department
  - CFO wants to see employees with their salary in descending order
  - HR manager wants to see employees with their number of dependents
- These can be easily accomplished by queries
- Problem: queries are not permanent
- Solution: CREATE VIEW

Scenario: Access Control of Sensitive Data

- We want employees be able to search each other (like the white page on UW) but want to hide sensitive information like birthday and salary
- Problem: Standard SQL can only control access at the table level
  - GRANT you or NOT GRANT you the right to access the employee table
  - It CANNOT control access to individual attribute within a table
- Solution: CREATE VIEW

It is a CRIME punishable by being FIRED!

- DO NOT USE VIEW AS INTERMIDEIATE STEPS WHEN WRITING A QUERY
  - Use VIEW as intermediate results when writing a query. Similar to R1, R2 … … Result in RA.
- YOU WILL GET ZERO IF YOU DO THAT IN EXAMS AND ASSINMENTS.
- Why?

Analogy: counting fingers is OK for a 4 years old. But for a 20+ with a UW degree … …

Assertion Motivation: Business Logic

- We want to inject some business rule into our database:
  - Say, employees with children should not work more than 100 hours on any project.
  - Or manager should have the highest salary in the department
  - Or students with average below 80 cannot take more than 5 courses.
- Mechanism in SQL: CREATE ASSERTION
  - Components include: a constraint name, followed by CHECK, followed by a condition

Using General Assertions

- Specify a query that finds all the violations of the condition; include inside a `NOT EXISTS` clause
- Query result must be empty
  - If the query result is not empty, the assertion has been violated. An error message should pop up
- Assertion (and later triggers) will be checked when a database is modified (insert, delete, update)

Assertions: Exercise

- The salary of an employee must not be greater than the salary of the manager of the department that the employee works for.

```
CREATE ASSERTION SALARY_CONSTRAINT
CHECK NOT EXISTS
(
        /* Query finds employee with salary greater than the manager */

);
```

Assertions: Answer

- The salary of an employee must not be greater than the salary of the manager of the department that the employee works for.

```
CREATE ASSERTION SALARY_CONSTRAINT
CHECK (NOT EXISTS
(
    /* Query finds employee with salary greater than the manager */
);
```

E

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

M

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5

SQL Triggers

- Objective: to monitor a database and take action when a condition occurs
- Triggers are expressed in a syntax similar to assertions and include the following:
  - event (e.g., an update operation)
  - condition
  - action (to be taken when the condition is satisfied)

SQL Triggers: Example (FYI)

Transactions(TID, accNo, amount, type, date, status)
Account(accNo, balance, owner)

- When a withdrawal is made, the balance in the account table should be updated accordingly

```
CREATE TRIGGER Update_balance
AFTER INSERT OF
  TID, amount, status ON Transactions

  FOR EACH ROW
       WHEN
      (new.status = "Success") AND (type = "Withdrawal")

       UPDATE        Account
       SET           balance = balance - new.amount
```

SQL Triggers: An Example (FYI)

- A trigger to compare an employee's salary to his/her supervisor during insert or update operations:

```
CREATE TRIGGER INFORM_SUPERVISOR
BEFORE INSERT OR UPDATE OF
  SALARY, SUPERSSN ON EMPLOYEE
  FOR EACH ROW
        WHEN
        (NEW.SALARY> (SELECT SALARY FROM EMPLOYEE
                      WHERE SSN=NEW.SUPERSSN))
        INFORM_SUPERVISOR (NEW.SUPERVISOR_SSN,NEW.SSN);
```

In-class Exercise #2

- Write a view called Expert that includes all employees (name and SSN) that work on all the projects.

```
CREATE VIEW Expert AS
      SELECT     FName, LName, SSN
      FROM       Employee, Works_ON
      WHERE      SSN = ESSN
      GROUP BY   SSN, FName, LName
      HAVING     COUNT(PNO) IN
                 (SELECT    COUNT(PNumber)
                 FROM       Project);
```

In-class Exercise #3

- Employee who works more than 1000 hours on projects should have salary more than $50,000

```
CREATE ASSERTION SALARY_CONSTRAINT
CHECK  NOT EXISTS
(
      SELECT       SSN
      FROM         Employee
      WHERE        Salary < 50000

      /* Find E with salary $50k or less, and 1000+ hours on projects*/

                   FROM         Works_ON
                   GROUP BY     ESSN
                   HAVING       SUM(Hours) > 1000
                   )
);
```

In-class Exercise #4

1. Find the names of employees who work on all the projects controlled by department number 5 (if you want an even tougher challenge, replace D5 with “Research”).

2. Employee who has dependent should work no more than 200 total hours on projects.

Summary of Today's Lecture

- Advanced SQL
  - View
  - Assertion
  - Trigger
- SQL exercise