THE UNIVERSITY OF TEXAS AT DALLAS

DATA BASE DESIGN PROJECT

CS 6360

TOPIC: CHILD DAY CARE SERVICES

Team Members:

Haneesha Gurugubelli - hxg170830

Namrata Rathore - nxr170630

Sushma Sunkollu Nagaraj - sxs173433

TABLE OF CONTENTS:

DESIGN REQUIREMENTS:

INTRODUCTION:

Child day care is the caring for and supervision of a child or children, usually ranging from age six weeks to age thirteen. Child care is the action or skill of looking after children by a day-care center, nannies, babysitter, teachers or other providers. Child care providers can be our children's first teachers, and therefore play an integral role in our systems of early childhood education. Quality child care has huge impact on the future successes of children. (Source : Wikipedia)

The caregivers play a vital role in the overall development of the child and they are provided extensive training in first aid and they are CPR certified. In addition, background checks, drug testing at all centers, and reference verification are normally a requirement.

Crux of Design:

In order to ensure child safety only parents are allowed to pick and drop off the children. Each child will also have an emergency contact. The centers not only look after the children but also involve children in various learning activities and games as well.

The Child Day care center has two departments they are the Mentoring department and managing department. Each department has a manager.

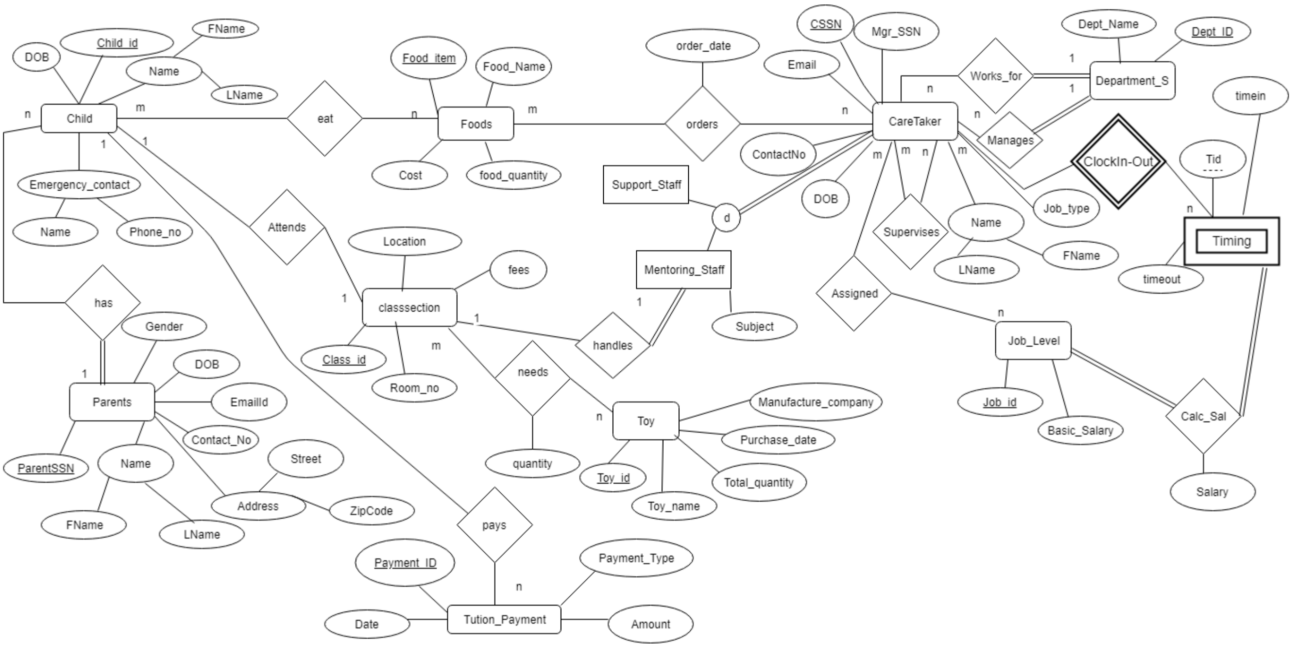
The mentoring department has two classes of employees one is the mentoring or the training staff and the other being support staff. The support staff take care of smooth running of the child care. The responsibilities usually includes procuring all the necessary supplies, food, clothes, toys and other supplies. They also keep track of the food, toys for the children, and every other needs of the children. They also deal with new enrolling new children, fee payment and inventory management.

The mentoring staff are the ones who play a vital role in teaching valuable lessons to children. They are completely responsible for taking care of the children. They involve the kids in various activities and train them on various lessons.

Each employee is required to sign in their time in and out time. Based on which their salary is calculated on hourly basis.

The parents are required to pay the fee for each day their child spends in the center. The fee is section dependent. Based on which fee payment is made on a daily basis or monthly basis.

ER DIAGRAM:



Assumptions made:

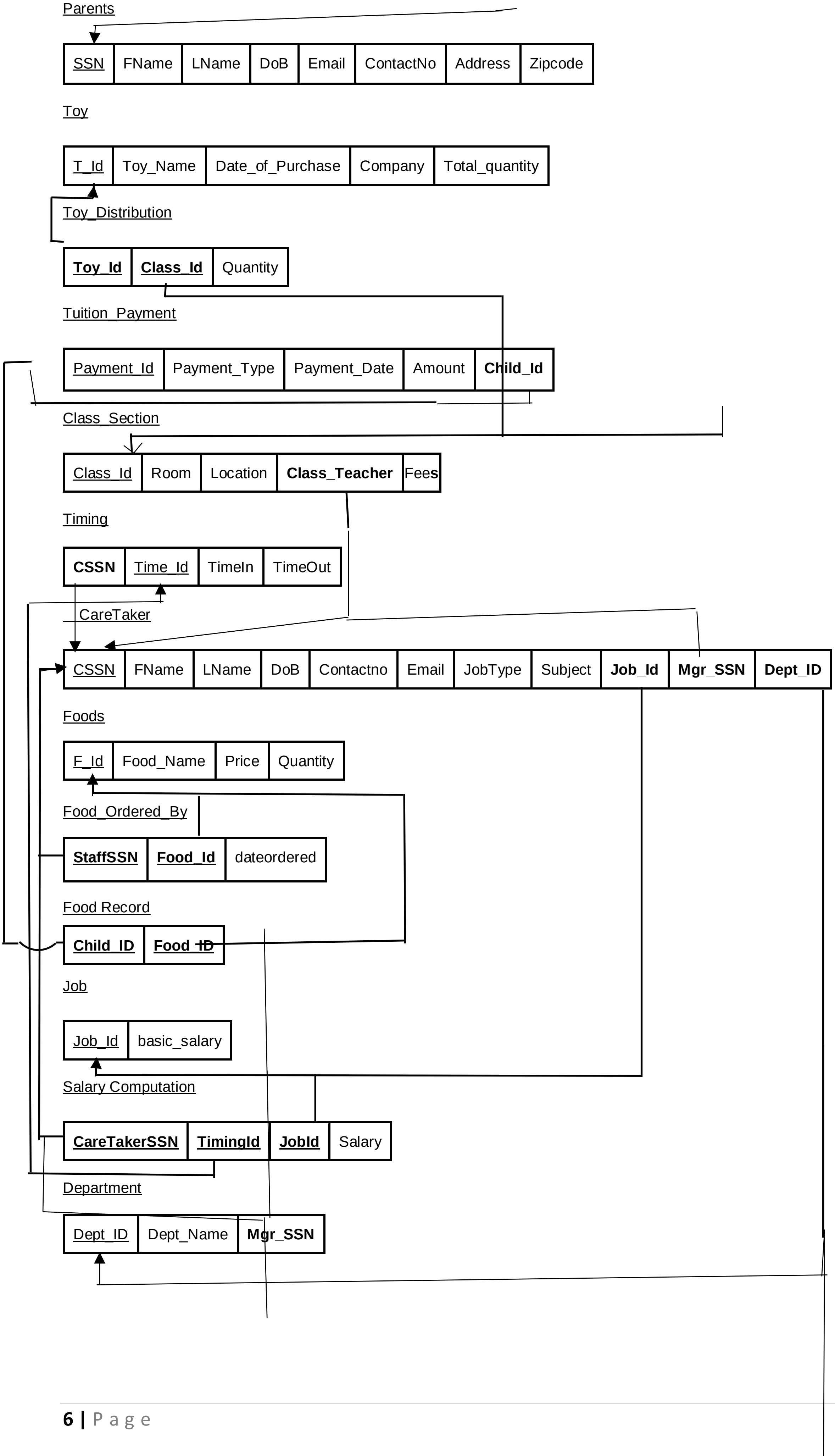
1. Only a parent can enroll the child at the day care center. The parent can enroll one or more children to the day care center.(one to many)
2. There is only one manager for each department.
3. Each class has unique identification number along with the room number and the location information.
4. Each child can only attend one class.
5. Child entity has unique id along with the Last name, first name, date of birth and emergency contact name and mobile number.
6. The child care fees have to be paid for every child. The fee is a recurring payment. It is paid on a monthly basis.
7. There are two departments teaching and support staff. Each caretaker belongs to either of the department.
8. Every child has a certain list of food they can consume. Food can be ordered by any of the staff members and they can order multiple food items.
9. The toys entity has information like the unique toy id, name, date of purchase, quantity of the toys available and the manufacture company.
10. One of the caretaker plays the role of both the manager and employee.
11. Each class is handled by a teaching staff. Both teaching staff and support staff belong to care taker entity. Each staff member can only be part of one department. Hence, they are disjoint entities.
12. Each class has been assigned different types of toys and the quantity based on the children enrolled.

13. Job entity, caretaker entity and timing are in ternary relationship. Timing details are recorded for each caretaker.
14. The salary is determined by the number of hours which are calculated from the time in and time out details and the job type. Each job type has a fixed salary per hour.

Relational Schema

Child

Child_Id	FName	LName	DoB	EName	Ephoneno	**PSSN**	**Class**



Parents

SSN	FName	LName	DoB	Email	ContactNo	Address	Zipcode

Toy

T_Id	Toy_Name	Date_of_Purchase	Company	Total_quantity

Toy_Distribution

Toy_Id	**Class_Id**	Quantity

Tuition_Payment

Payment_Id	Payment_Type	Payment_Date	Amount	**Child_Id**

Class_Section

Class_Id	Room	Location	**Class_Teacher**	Fees

Timing

CSSN	Time_Id	TimeIn	TimeOut

CareTaker

CSSN	FName	LName	DoB	Contactno	Email	JobType	Subject	**Job_Id**	**Mgr_SSN**	**Dept_ID**

Foods

F_Id	Food_Name	Price	Quantity

Food_Ordered_By

StaffSSN	**Food_Id**	dateordered

Food Record

Child_ID	**Food_ID**

Job

Job_Id	basic_salary

Salary Computation

CareTakerSSN	**TimingId**	**JobId**	Salary

Department

Dept_ID	Dept_Name	**Mgr_SSN**

CHILD:

CHILD_ID	FNAME	LNAME	DOB	ENAME	EPHONENO	PSSN	CLASS

Foreign Keys:

- FOREIGN KEY (PSSN) REFERENCES PARENTS(SSN)
- FOREIGN KEY (CLASS) REFERENCES CLASS_SECTION(CLASS_ID)

PARENTS:

SSN	FNAME	LNAME	DOB	EMAIL	CONTACT NO	ADDRESS	ZIPCODE

TOY:

T_ID	TOY_NAME	DATE_OF_PURCHASE	COMPANY	TOTAL_QUANTITY

TOY_DISTRIBUTION:

TOY_ID	CLASS_ID	QUANTITY

Foreign Keys:

- FOREIGN KEY (TOY_ID) REFERENCES TOY(T_ID)
- FOREIGN KEY (CLASS_ID) REFERENCES CLASS_SECTION(CLASS_ID)

TUITION_PAYMENT:

PAYMENT_ID	PAYMENT_TYPE	PAYMENT_DATE	AMOUNT	CHILD_ID

Foreign Keys:

- FOREIGN KEY (CHILD_ID) REFERENCES CHILD(CHILD_ID)

CLASS_SECTION:

CLASS_ID	ROOM	LOCATION	CLASS_TEACHER	FEES

Foreign Keys:

- FOREIGN KEY (CLASS_TEACHER) REFERENCES CARETAKER(CSSN)

TIMING:

TIME_ID	CSSN	TIMEIN	TIMEOUT

Foreign Keys:

- FOREIGN KEY (CSSN) REFERENCES CARETAKER(CSSN)

CARE_TAKER:

CSSN	FNAME	LNAME	DOB	EMAIL	CONTACTNO	JOBTYPE	SUBJECT	JOB_ID	MGR_SSN	DEPT_ID

Foreign Keys:

- FOREIGN KEY (JOB_ID) REFERENCES JOB(JOB_ID)
- FOREIGN KEY (DEPT_ID) REFERENCES DEPARTMENT(DEPT_ID)
- Constraint: SUPERVISOR_KEY: FOREIGN KEY (MGR_SSN) REFERENCES CARETAKER(CSSN)

FOODS:

F_ID	FOOD_NAME	PRICE	QUANTITY

FOOD_ORDERED_BY:

STAFFSSN	FOOD_ID	DATEORDERED

Foreign Keys:

- FOREIGN KEY (STAFFSSN) REFERENCES CARETAKER(CSSN)
- FOREIGN KEY (FOOD_ID) REFERENCES FOODS(F_ID)

FOOD_RECORD:

CHILD_ID	FOOD_ID

Foreign Keys:

- FOREIGN KEY (CHILD_ID) REFERENCES CHILD(CHILD_ID)
- FOREIGN KEY (FOOD_ID) REFERENCES FOODS(F_ID)

JOB:

JOB_ID	BASIC_SALARY

SALARY_COMPUTATION:

CARETAKERSSN	TIMINGID	JOBID	SALARY

Foreign Keys:

- FOREIGN KEY (CARETAKERSSN) REFERENCES CARETAKER(CSSN)
- FOREIGN KEY (TIMINGID) REFERENCES TIMING(TIME_ID)
- FOREIGN KEY (JOBID) REFERENCES JOB(JOB_ID)

DEPARTMENT:

DEPT_ID	DEPT_NAME	MGR_SSN

Foreign Keys:

- Constraint: MGR_KEY: FOREIGN KEY (MGR_SSN) REFERENCES CARETAKER(CSSN)

NORMALIZATION OF TABLES:

Violates 1st normal Form:

- FOOD_RECORD {CHILD_ID, [FOOD_ID]} → Child can eat multiple food Items

- FOOD_ORDERED_BY {STAFFSSN, [FOOD_ID], DATEORDERED} → Staff can have multiple food Items at same time.

Violates 2nd normal Form:

- CHILD {CHILD_ID, CLASS_ID, FNAME, LNAME, DOB, ENAME, EPHONENO, PSSN, CLASS, CLASS_TEACHER, LOCATION, ROOM, PFNAME, PLNAME, DOB, EMAIL, CONTACTNO, ADDRESS, ZIPCODE} → (CLASS_TEACHER, LOCATION, ROOM depends on Partial Primary key CLASS_ID, and FNAME, LNAME, DOB, ENAME, EPHONENO, PSSN, CLASS depends on partial Primary key CHILD_ID)

Violates 3rd normal Form:

- CHILD {CHILD_ID, FNAME, LNAME, DOB, ENAME, EPHONENO, PSSN, CLASS, PFNAME, PLNAME, DOB, EMAIL, CONTACTNO, ADDRESS, ZIPCODE} Here Primary Key is CHILD_ID but {PFNAME, PLNAME, DOB, EMAIL, CONTACTNO, ADDRESS, ZIPCODE} depends on non-primary key PSSN.

Hence Violates 3rd Normalization form.

Relational Schema After Normalization:

CSSN→ {FNAME, LNAME, DOB, EMAIL, CONTACTNO, JOBTYPE, SUBJECT, JOB_ID, MGR_SSN, DEPT_ID}

CHILD_ID → {FNAME, LNAME, DOB, ENAME, EPHONENO, PSSN, CLASS}

CLASS_ID → {CLASS_TEACHER, LOCATION, ROOM}

DEPT_ID → {DEPT_NAME, MGR_SSN}

F_ID → {FOOD_NAME, PRICE, QUANTITY}

FOOD_ORDERED_BY {STAFFSSN, FOOD_ID, DATEORDERED}

CHILD_ID → {FOOD_ID}

JOB_ID → {BASIC_SALARY}

SSN → {FNAME, LNAME, DOB, EMAIL, CONTACTNO, ADDRESS, ZIPCODE}

{CARETAKERSSN, TIMINGID, JOBID} → {SALARY}

TIME_ID → {CSSN, TIMEIN, TIMEOUT}

T_ID → {TOY_NAME, DATE_OF_PURCHASE, COMPANY, TOTAL_QUANTITY}

{TOY_ID, CLASS_ID} → {QUANTITY}

PAYMENT_ID → {PAYMENT_TYPE, PAYMENT_DATE, AMOUNT, CHILD_ID}

TABLES:

```
CREATE TABLE PARENTS (
SSN CHAR (9) NOT NULL, FNAME VARCHAR (50) NOT NULL,
LNAME VARCHAR (50), DOB DATE,
EMAIL VARCHAR (10),
CONTACTNO CHAR (10),
ADDRESS VARCHAR (30), ZIPCODE CHAR (5),
PRIMARY KEY (SSN)
);
```

```
CREATE TABLE TOY (
T_ID INT NOT NULL,
TOY_NAME VARCHAR (20),
DATE_OF_PURCHASE DATE,
COMPANY VARCHAR (10),
TOTAL_QUANTITY INT,
PRIMARY KEY (T_ID)
);
```

```
CREATE TABLE FOODS (
F_ID INT NOT NULL,
FOOD_NAME VARCHAR (20),
PRICE INT,
QUANTITY INT,
PRIMARY KEY(F_ID)
```

```
);
CREATE TABLE JOB (
JOB_ID INT NOT NULL,
BASIC_SALARY INT,
PRIMARY KEY(JOB_ID)
);

CREATE TABLE DEPARTMENT (
DEPT_ID INT NOT NULL AUTO_INCREMENT,
DEPT_NAME VARCHAR (20),
MGR_SSN CHAR (9),
PRIMARY KEY (DEPT_ID)
);

CREATE TABLE CARETAKER (
CSSN CHAR (9) NOT NULL,
FNAME VARCHAR (50) NOT NULL,
LNAME VARCHAR (50),
DOB DATE,
EMAIL VARCHAR (10),
CONTACTNO CHAR (10),
JOBTYPE CHAR (1),
SUBJECT VARCHAR (20),
JOB_ID INT,
MGR_SSN CHAR (9),
DEPT_ID INT,
PRIMARY KEY(CSSN),
FOREIGN KEY (JOB_ID) REFERENCES JOB (JOB_ID),

FOREIGN KEY (DEPT_ID) REFERENCES DEPARTMENT (DEPT_ID)

);


ALTER TABLE CARETAKER ADD CONSTRAINT SUPERVISOR_KEY
FOREIGN KEY (MGR_SSN) REFERENCES CARETAKER (CSSN);

ALTER TABLE DEPARTMENT ADD CONSTRAINT MGR_KEY FOREIGN KEY
(MGR_SSN) REFERENCES CARETAKER (CSSN);

CREATE TABLE CLASSSECTION (

CLASS_ID INT NOT NULL,

ROOM VARCHAR (10),

LOCATION VARCHAR (20),

CLASS_TEACHER CHAR (9),

PRIMARY KEY (CLASS_ID),

FOREIGN KEY (CLASS_TEACHER) REFERENCES CARETAKER (CSSN)

);


CREATE TABLE CHILD (

CHILD_ID INT NOT NULL,

FNAME VARCHAR (50) NOT NULL,

LNAME VARCHAR (50),

DOB DATE, ENAME VARCHAR (10), EPHONENO CHAR (10),

PSSN CHAR (9),

CLASS INT,

PRIMARY KEY (CHILD_ID),

FOREIGN KEY (PSSN) REFERENCES PARENTS(SSN),

FOREIGN KEY (CLASS) REFERENCES CLASSSECTION(CLASS_ID)

);
```

```
CREATE TABLE
TOY_DISTRIBUTION (

TOY_ID INT NOT NULL,

CLASS_ID INT NOT NULL,

QUANTITY INT,

FOREIGN KEY (TOY_ID) REFERENCES TOY (T_ID),

FOREIGN KEY (CLASS_ID) REFERENCES CLASSSECTION (CLASS_ID)

);

CREATE TABLE TUITION_PAYMENT (

PAYMENT_ID INT NOT NULL,

PAYMENT_TYPE VARCHAR (20) NOT NULL,

PAYMENT_DATE DATE, AMOUNT INT,

CHILD_ID INT,

PRIMARY KEY (PAYMENT_ID),

FOREIGN KEY (CHILD_ID) REFERENCES CHILD (CHILD_ID)

);

CREATE TABLE TIMING (

TIME_ID INT NOT NULL AUTO_INCREMENT,

CSSN CHAR (9) NOT NULL,

TIMEIN TIMESTAMP,

TIMEOUT TIMESTAMP,

PRIMARY KEY(TIME_ID),

FOREIGN KEY (CSSN) REFERENCES CARETAKER (CSSN));
```

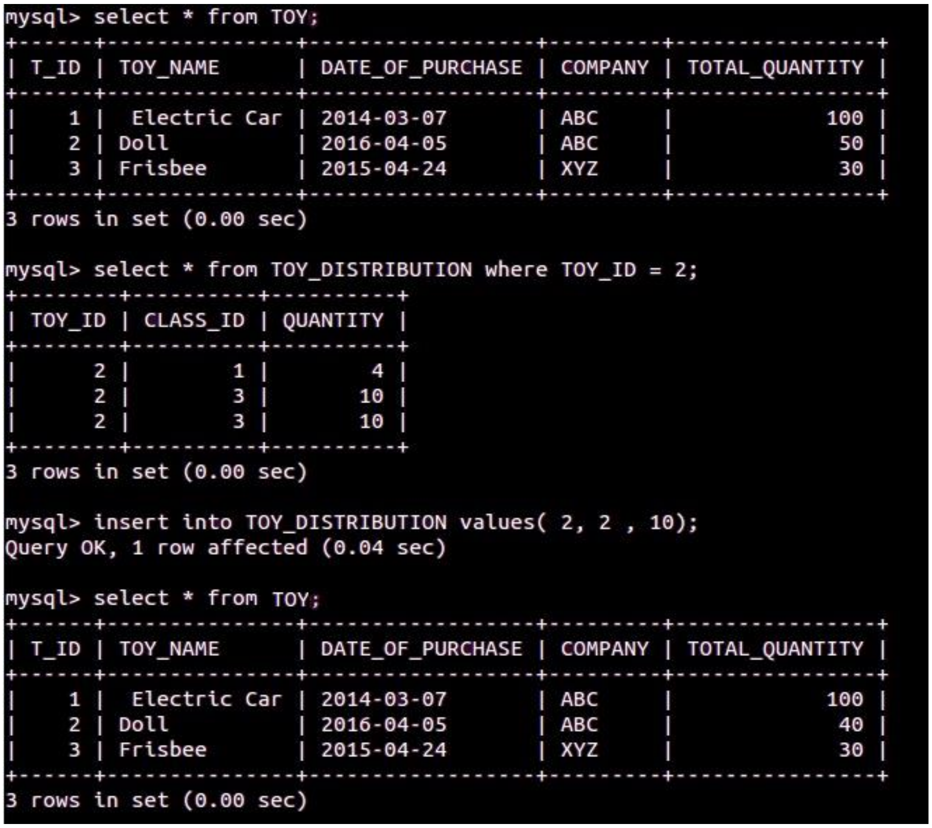
```
CREATE TABLE FOODORDEREDBY (
STAFFSSN CHAR (9) NOT NULL,
FOOD_ID INT NOT NULL,
DATEORDERED DATE,
FOREIGN KEY (STAFFSSN) REFERENCES CARETAKER (CSSN), FOREIGN KEY
(FOOD_ID) REFERENCES FOOD(F_ID)
);

CREATE TABLE FOODRECORD (
CHILD_ID INT NOT NULL,
FOOD_ID INT NOT NULL,
PRIMARY KEY (CHILD_ID, FOOD_ID),
FOREIGN KEY (CHILD_ID) REFERENCES CHILD(CHILD_ID), FOREIGN KEY
(FOOD_ID) REFERENCES FOODS(F_ID)
);

CREATE TABLE SALARYCOMPUTATION (
CARETAKERSSN CHAR (9) NOT NULL, TIMINGID INT NOT NULL, JOBID INT NOT
NULL, SALARY INT,
PRIMARY KEY (CARETAKERSSN, TIMINGID, JOBID),
FOREIGN KEY (CARETAKERSSN) REFERENCES CARETAKER (CSSN), FOREIGN
KEY (TIMINGID) REFERENCES TIMING (TIME_ID), FOREIGN KEY (JOBID)
REFERENCES JOB(JOB_ID)
);
```

TRIGGERS and PROCEDURES:

TRIGGERS:

1) When the Toys are distributed to Class, Their Total Quantity must be reduced from Toy Table.

```
DELIMITER$$

CREATE OR REPLACE TRIGGER `TOYS_UPDATE`

AFTER INSERT ON `TOY_DISTRIBUTION`

FOR EACH ROW

BEGIN

UPDATE `TOY` SET TOTAL_QUANTITY = TOTAL_QUANTITY - NEW.QUANTITY
WHERE T_ID=NEW.TOY_ID;

END$$

DELIMITER;
```

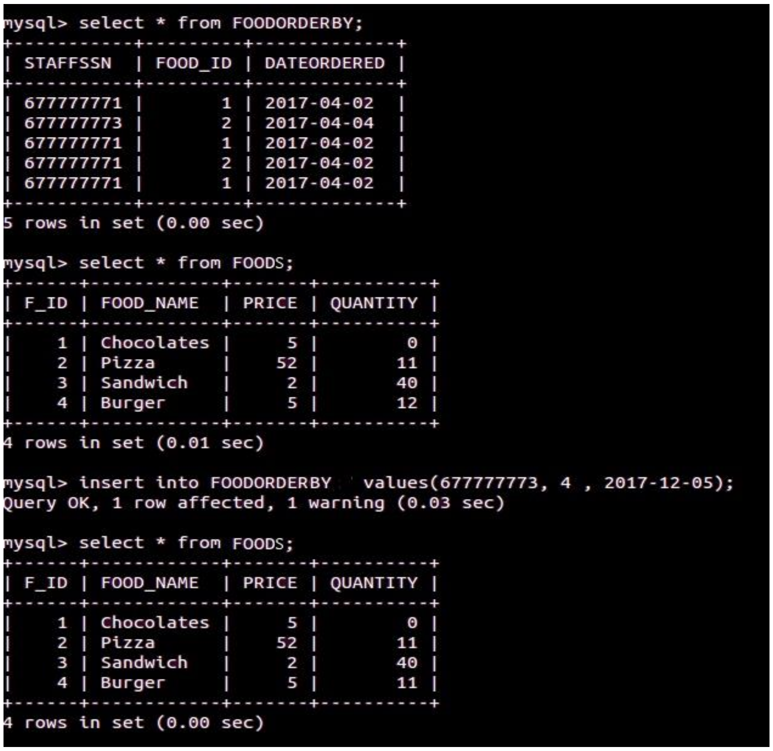


```
mysql> select * from TOY;
+------+--------------+------------------+---------+----------------+
| T_ID | TOY_NAME     | DATE_OF_PURCHASE | COMPANY | TOTAL_QUANTITY |
+------+--------------+------------------+---------+----------------+
|    1 |  Electric Car | 2014-03-07       | ABC     |            100 |
|    2 | Doll         | 2016-04-05       | ABC     |             50 |
|    3 | Frisbee      | 2015-04-24       | XYZ     |             30 |
+------+--------------+------------------+---------+----------------+
3 rows in set (0.00 sec)

mysql> select * from TOY_DISTRIBUTION where TOY_ID = 2;
+--------+----------+----------+
| TOY_ID | CLASS_ID | QUANTITY |
+--------+----------+----------+
|      2 |        1 |        4 |
|      2 |        3 |       10 |
|      2 |        3 |       10 |
+--------+----------+----------+
3 rows in set (0.00 sec)

mysql> insert into TOY_DISTRIBUTION values( 2, 2 , 10);
Query OK, 1 row affected (0.04 sec)

mysql> select * from TOY;
+------+--------------+------------------+---------+----------------+
| T_ID | TOY_NAME     | DATE_OF_PURCHASE | COMPANY | TOTAL_QUANTITY |
+------+--------------+------------------+---------+----------------+
|    1 |  Electric Car | 2014-03-07       | ABC     |            100 |
|    2 | Doll         | 2016-04-05       | ABC     |             40 |
|    3 | Frisbee      | 2015-04-24       | XYZ     |             30 |
+------+--------------+------------------+---------+----------------+
3 rows in set (0.00 sec)
```

2. When Food Is Ordered Its Quantity must be delete from Quantity of Food Table

```
DELIMITER $$

CREATE TRIGGER `UPDATE_FOOD_QUANTITY`

BEFORE INSERT ON `FOODORDEREDBY`

FOR EACH ROW

BEGIN

SELECT QUANTITY INTO @QUANTITY FROM FOOD WHERE F_ID = NEW.FOOD_ID;

IF @QUANTITY > 0 THEN

UPDATE FOOD SET QUANTITY = @QUANTITY - 1 WHERE F_ID = NEW.FOOD_ID;

ELSEIF @QUANTITY = 0 THEN

UPDATE `Error: Cannot Order FOOD, Quantity is ZERO` SET x=1;

END IF;

END$$

DELIMITER;
```

RESULTS:



```
mysql> select * from FOODORDERBY;
+-----------+---------+------------+
| STAFFSSN  | FOOD_ID | DATEORDERED |
+-----------+---------+------------+
| 677777771 |       1 | 2017-04-02  |
| 677777773 |       2 | 2017-04-04  |
| 677777771 |       1 | 2017-04-02  |
| 677777771 |       2 | 2017-04-02  |
| 677777771 |       1 | 2017-04-02  |
+-----------+---------+------------+
5 rows in set (0.00 sec)

mysql> select * from FOODS;
+------+------------+-------+----------+
| F_ID | FOOD_NAME  | PRICE | QUANTITY |
+------+------------+-------+----------+
|    1 | Chocolates |     5 |        0 |
|    2 | Pizza      |    52 |       11 |
|    3 | Sandwich   |     2 |       40 |
|    4 | Burger     |     5 |       12 |
+------+------------+-------+----------+
4 rows in set (0.01 sec)

mysql> insert into FOODORDERBY   values(677777773, 4 , 2017-12-05);
Query OK, 1 row affected, 1 warning (0.03 sec)

mysql> select * from FOODS;
+------+------------+-------+----------+
| F_ID | FOOD_NAME  | PRICE | QUANTITY |
+------+------------+-------+----------+
|    1 | Chocolates |     5 |        0 |
|    2 | Pizza      |    52 |       11 |
|    3 | Sandwich   |     2 |       40 |
|    4 | Burger     |     5 |       11 |
+------+------------+-------+----------+
4 rows in set (0.00 sec)
```

3. When TIMING IN and OUT is added in TIMING table for Particular CARETAKER, Depending on Number of Hours Worked his/her salary is Computed.

CREATE TABLE SALARYCOMPUTATION (

CARETAKERSSN CHAR (9) NOT NULL,

TIMINGID INT NOT NULL,

JOBID INT NOT NULL,

SALARY INT,

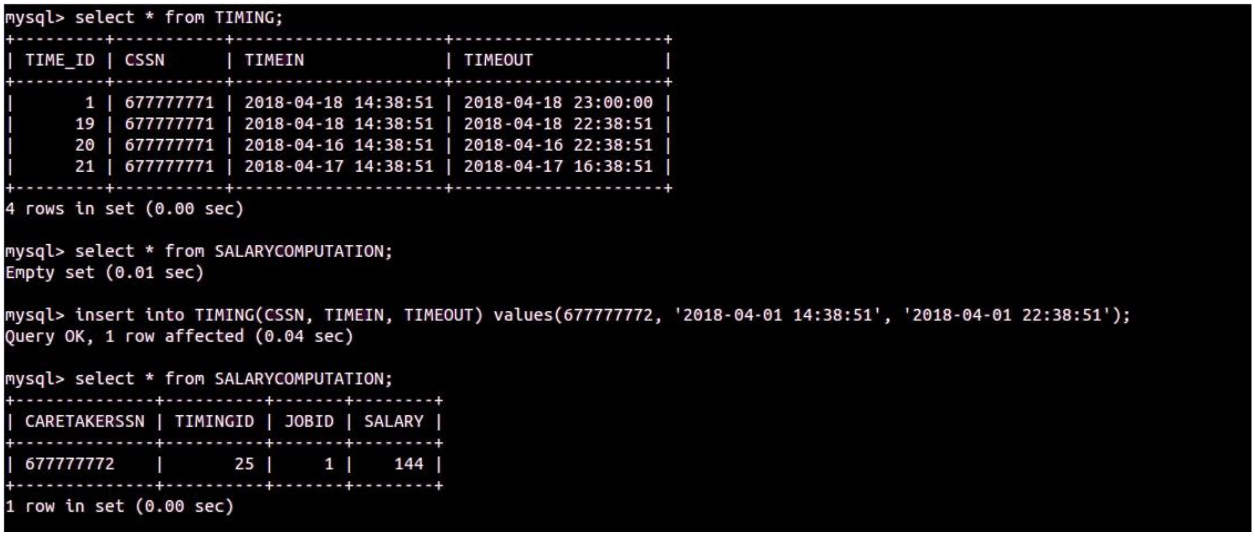
PRIMARY KEY (CARETAKERSSN, TIMINGID, JOBID),

FOREIGN KEY (CARETAKERSSN) REFERENCES CARETAKER(CSSN),

FOREIGN KEY (TIMINGID) REFERENCES TIMING(TIME_ID),

```
FOREIGN KEY (JOBID) REFERENCES JOB(JOB_ID));

DELIMITER $$

CREATE TRIGGER `SALARY_COMPUTE`

AFTER INSERT ON `TIMING`

FOR EACH ROW

BEGIN

SELECT C. CSSN, C.JOB_ID, J.BASIC_SALARY INTO @CSSN, @JID, @BASICSALARY FROM `CARETAKER` C JOIN JOB J ON J.JOB_ID = C.JOB_ID WHERE C.CSSN = NEW.CSSN;

SET @SALARYCOMPUTED = (NEW.TIMEOUT - NEW.TIMEIN) *@BASICSALARY/10000;

SET @TOTALTIME = (NEW.TIMEOUT - NEW.TIMEIN);

INSERT INTO `SALARYCOMPUTATION` VALUES (@CSSN, NEW.TIME_ID, @JID, @SALARYCOMPUTED);

END$$

DELIMITER;
```

RESULTS:



```
mysql> select * from TIMING;
+---------+-----------+---------------------+---------------------+
| TIME_ID | CSSN      | TIMEIN              | TIMEOUT             |
+---------+-----------+---------------------+---------------------+
|       1 | 677777771 | 2018-04-18 14:38:51 | 2018-04-18 23:00:00 |
|      19 | 677777771 | 2018-04-18 14:38:51 | 2018-04-18 22:38:51 |
|      20 | 677777771 | 2018-04-16 14:38:51 | 2018-04-16 22:38:51 |
|      21 | 677777771 | 2018-04-17 14:38:51 | 2018-04-17 16:38:51 |
+---------+-----------+---------------------+---------------------+
4 rows in set (0.00 sec)

mysql> select * from SALARYCOMPUTATION;
Empty set (0.01 sec)

mysql> insert into TIMING(CSSN, TIMEIN, TIMEOUT) values(677777772, '2018-04-01 14:38:51', '2018-04-01 22:38:51');
Query OK, 1 row affected (0.04 sec)

mysql> select * from SALARYCOMPUTATION;
+--------------+----------+-------+--------+
| CARETAKERSSN | TIMINGID | JOBID | SALARY |
+--------------+----------+-------+--------+
| 677777772    |       25 |     1 |    144 |
+--------------+----------+-------+--------+
1 row in set (0.00 sec)
```

PROCEDURES:

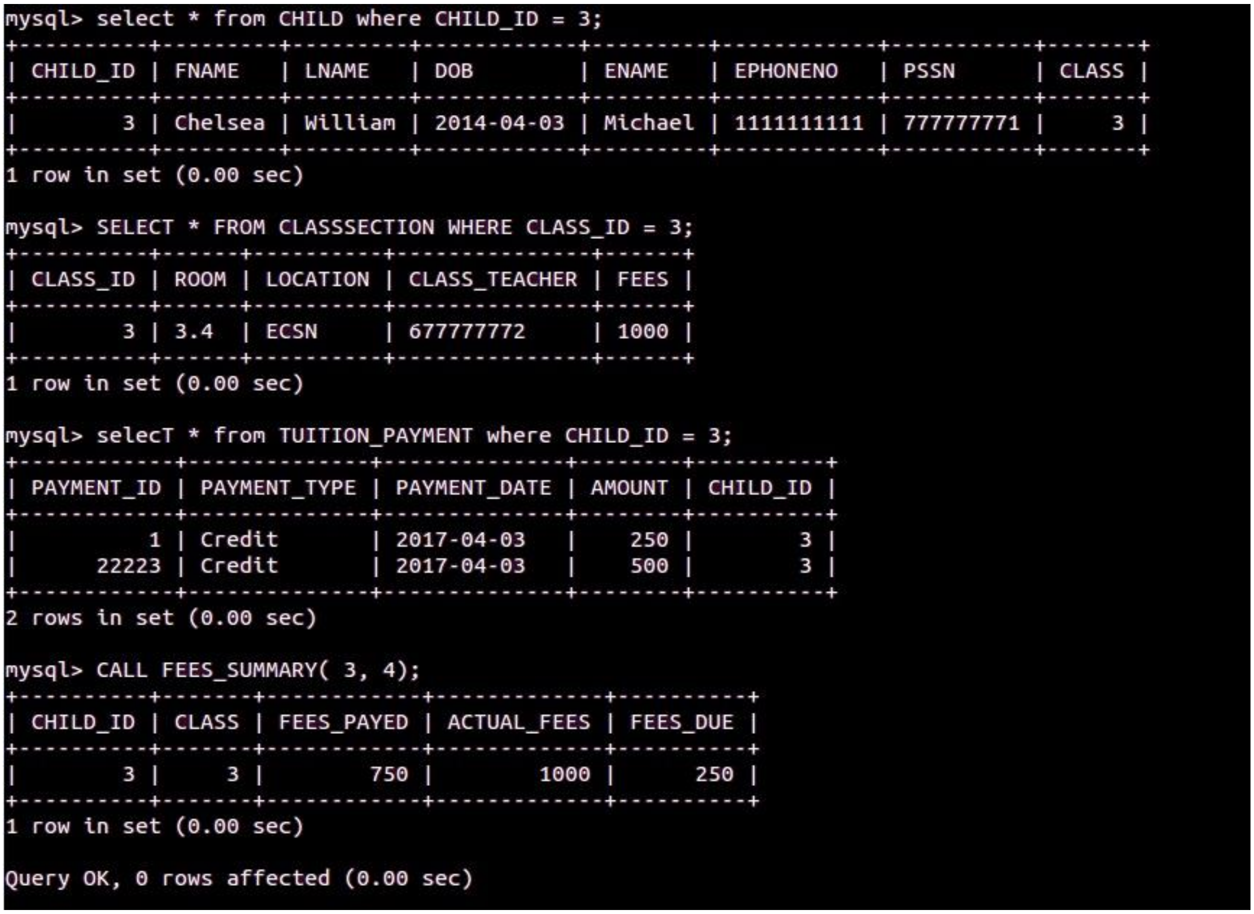
1. Fees Summary of Particular Child for Particular Month Is Computed in Following Procedure:

INPUT → (CHILD_ID, MONTH)

OUTPUT → (CHILD_ID, CLASS, FEES_PAYED, ACTUAL_FEES, FEES_DUE)

```
DELIMITER //

CREATE PROCEDURE FEES_SUMMARY (IN CHILD_ID INT, IN MONTH INT)

BEGIN

DECLARE CLASS INT;

DECLARE FEES_PAYED INT;

DECLARE ACTUAL_FEES INT;

DECLARE FEES_DUE INT;

SELECT  C.CLASS Class, SUM(T.AMOUNT) Fees_Payed , CS.FEES Actual_Fees, ( CS.FEES
- SUM(T.AMOUNT)) Fees_Due INTO CLASS, FEES_PAYED, ACTUAL_FEES, FEES_DUE
from TUITION_PAYMENT T, CLASSSECTION CS, CHILD C where T.CHILD_ID =
C.CHILD_ID and CS.CLASS_ID = C.CLASS and C.CHILD_ID = CHILD_ID and
MONTH(T.PAYMENT_DATE) = MONTH GROUP BY MONTH(PAYMENT_DATE),
T.CHILD_ID;

SELECT CHILD_ID, CLASS, FEES_PAYED, ACTUAL_FEES, FEES_DUE;

END//

DELIMITER;

CALL FEES_SUMMARY (3,4);
```

RESULTS:



```
mysql> select * from CHILD where CHILD_ID = 3;
+----------+---------+---------+------------+---------+------------+-----------+-------+
| CHILD_ID | FNAME   | LNAME   | DOB        | ENAME   | EPHONENO   | PSSN      | CLASS |
+----------+---------+---------+------------+---------+------------+-----------+-------+
|        3 | Chelsea | William | 2014-04-03 | Michael | 1111111111 | 777777771 |     3 |
+----------+---------+---------+------------+---------+------------+-----------+-------+
1 row in set (0.00 sec)

mysql> SELECT * FROM CLASSSECTION WHERE CLASS_ID = 3;
+----------+------+----------+---------------+------+
| CLASS_ID | ROOM | LOCATION | CLASS_TEACHER | FEES |
+----------+------+----------+---------------+------+
|        3 | 3.4  | ECSN     | 677777772     | 1000 |
+----------+------+----------+---------------+------+
1 row in set (0.00 sec)

mysql> selecT * from TUITION_PAYMENT where CHILD_ID = 3;
+------------+--------------+--------------+--------+----------+
| PAYMENT_ID | PAYMENT_TYPE | PAYMENT_DATE | AMOUNT | CHILD_ID |
+------------+--------------+--------------+--------+----------+
|          1 | Credit       | 2017-04-03   |    250 |        3 |
|      22223 | Credit       | 2017-04-03   |    500 |        3 |
+------------+--------------+--------------+--------+----------+
2 rows in set (0.00 sec)

mysql> CALL FEES_SUMMARY( 3, 4);
+----------+-------+------------+-------------+----------+
| CHILD_ID | CLASS | FEES_PAYED | ACTUAL_FEES | FEES_DUE |
+----------+-------+------------+-------------+----------+
|        3 |     3 |        750 |        1000 |      250 |
+----------+-------+------------+-------------+----------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

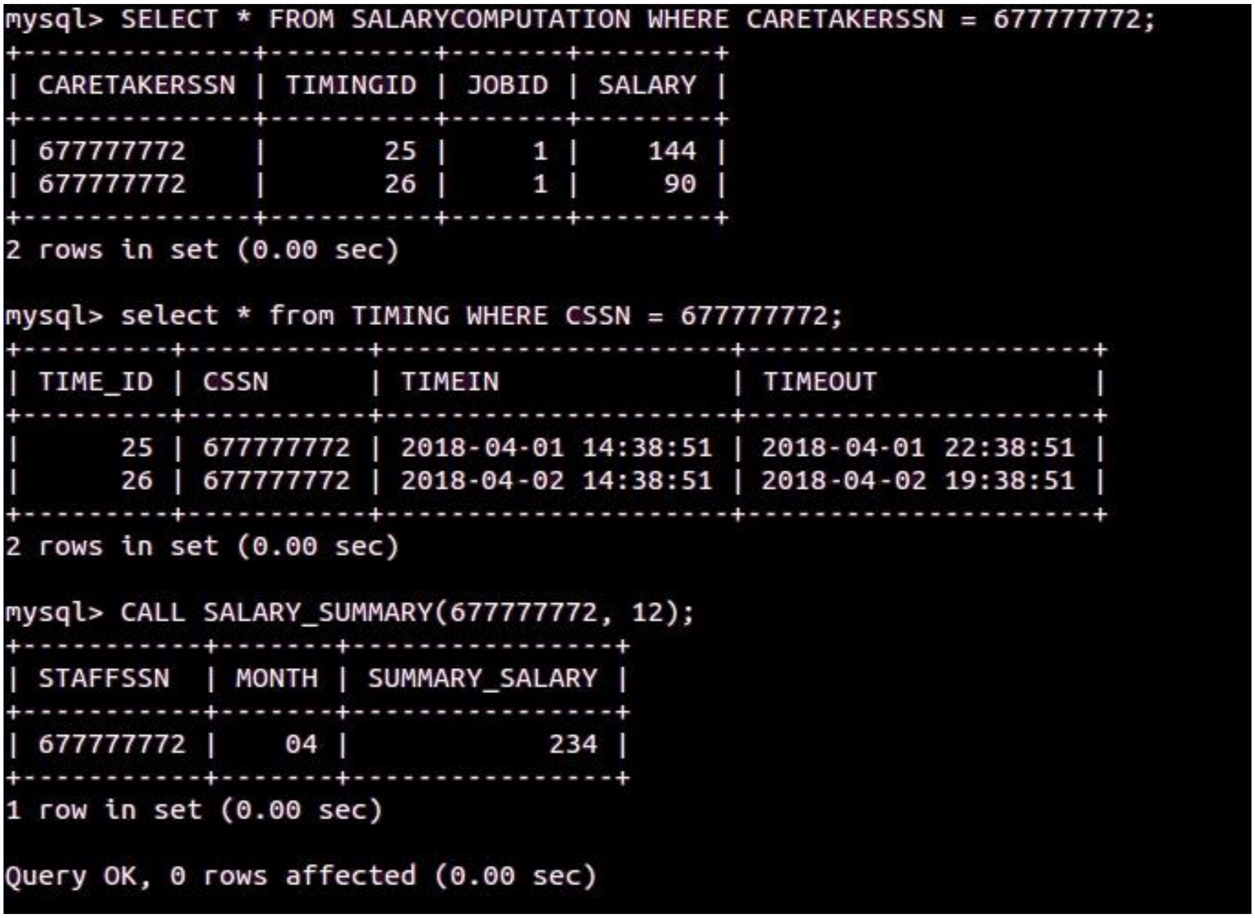
2. Compute Salary of Caretaker by Summing up all the salaries in salary Computation for Specific month.

INPUT → (STAFFSSN, MONTH)

OUTPUT → (STAFFSSN, MONTH, SUMMARY_SALARY)

```
DELIMITER//

CREATE PROCEDURE SALARY_SUMMARY (IN STAFFSSN INT, IN MONTH INT)

BEGIN

DECLARE SUMMARY_SALARY INT;

SELECT SUM (SALARY) INTO SUMMARY_SALARY from SALARYCOMPUTATION
where CARETAKERSSN = STAFFSSN and TIMINGID IN (select TIME_ID from TIMING
WHERE MONTH (TIMEIN) = MONTH);

SELECT STAFFSSN, MONTH, SUMMARY_SALARY;

END//

DELIMITER;

CALL SALARY_SUMMARY (677777772, 4)
```

RESULTS:



```
mysql> SELECT * FROM SALARYCOMPUTATION WHERE CARETAKERSSN = 677777772;
+--------------+----------+-------+--------+
| CARETAKERSSN | TIMINGID | JOBID | SALARY |
+--------------+----------+-------+--------+
| 677777772    |       25 |     1 |    144 |
| 677777772    |       26 |     1 |     90 |
+--------------+----------+-------+--------+
2 rows in set (0.00 sec)

mysql> select * from TIMING WHERE CSSN = 677777772;
+---------+-----------+---------------------+---------------------+
| TIME_ID | CSSN      | TIMEIN              | TIMEOUT             |
+---------+-----------+---------------------+---------------------+
|      25 | 677777772 | 2018-04-01 14:38:51 | 2018-04-01 22:38:51 |
|      26 | 677777772 | 2018-04-02 14:38:51 | 2018-04-02 19:38:51 |
+---------+-----------+---------------------+---------------------+
2 rows in set (0.00 sec)

mysql> CALL SALARY_SUMMARY(677777772, 12);
+-----------+-------+----------------+
| STAFFSSN  | MONTH | SUMMARY_SALARY |
+-----------+-------+----------------+
| 677777772 |    04 |            234 |
+-----------+-------+----------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```