```
/*
Noah Zhou
CNIT 272 Fall 2023
Lab Time: Friday 7:30am - 9:20am
*/

--
**************************************************************************
********
--Question 1
--To add a new worker as well as new lunch order records to the database.

--Step 1
/*
First, confirm that there is no worker ID 600 in the WORKER table. Run a
SQL statement and search for worker ID 600.
(no rows selected)
*/

SELECT worker_id
FROM worker
WHERE worker_id = '600';

--Step 2
/*
Add a worker to the WORKER table with the following details:
*/

INSERT INTO worker (worker_id, first_name, last_name, city, dept_code,
hire_date, credit_limit, phone_number, manager_id)
VALUES ('600', 'Katie', 'Smith', 'Chicago', 'Com', '30-JAN-23', '22',
'7592', '565');

--Step 3
/*
Verify the new worker by selecting all columns for worker_id 600 in the
WORKER table.
(1 row selected)
*/

SELECT *
FROM worker
WHERE worker_id = '600';

--Step 4
/*
Confirm that there are no lunch IDs associated with worker ID 600 in the
LUNCH table. Run a SQL statement and search for worker ID 600.
(no row selected)
*/

SELECT worker_id
FROM lunch
WHERE worker_id = '600';
```

```
--Step 5
/*
Add 2 lunches to the LUNCH table with the following details:
*/

INSERT INTO lunch (lunch_id, lunch_date, worker_id)
VALUES ( '53', '02-FEB-23', '600');

INSERT INTO lunch (lunch_id, lunch_date, worker_id)
VALUES ('54', '04-FEB-23', '600');

--Step 6
/*
Create a new lunch order of your own for the new worker..
*/

INSERT INTO lunch (lunch_id, lunch_date, worker_id)
VALUES ('55', '06-FEB-23', '600');

--Step 7
/*
Verify the 3 new lunches by selecting all columns for worker ID 600 in
the
LUNCH table (3 rows selected)
*/

SELECT *
FROM lunch
WHERE worker_id = '600';

--Step 8
/*
NOW, Issue a COMMIT
*/

COMMIT;

/*
Results:
[Step 1]
no rows selected

[Step 2]
1 row inserted.

[Step 3]
WOR FIRST_NAME LAST_NAME           CITY                           DEP
HIRE_DATE CREDIT_LIMIT PHON MAN
--- ---------- -------------------- ------------------------------ --- --
------- ------------ ---- ---
600 Katie      Smith                Chicago                        Com
30-JAN-23           22 7592 565
```

```
1 row selected.

[Step 4]
no rows selected

[Step 5]
1 row inserted.

1 row inserted.

[Step 6]
1 row inserted.

[Step 7]
LUNCH_ID LUNCH_DAT WOR
---------- --------- ---
      53 02-FEB-23 600
      54 04-FEB-23 600
      55 06-FEB-23 600

[Step 8]
Commit complete.
*/

--
**********************************************************************
********
--Question 2
--To update a workerâ€™s details in the database.

--Step 1
/*
List the Worker ID, First Name, Last Name, and city for the worker ID
600.
(WORKER table)
(1 row selected)
*/

SELECT worker_id, first_name, last_name, city
FROM WORKER
WHERE worker_id = '600';

--Step 2
/*
The worker 600 has moved and changed their last name. The new worker's
last name is Kelly, and the city is now Oak Brook. Update the database by
changing the name and city for worker ID 600.
(1 row updated)
*/

UPDATE worker
SET last_name = 'Kelly', city = 'Oak Brook'
WHERE worker_id = '600';
```

```
--Step 3
/*
Issue a COMMIT.
*/

COMMIT;

--Step 4
/*
To verify the update, Worker ID, First Name, Last Name, and city for the
worker ID 600. (1 row selected)
*/

SELECT worker_id, first_name, last_name, city
FROM WORKER
WHERE worker_id = '600';

/*
Results:
[Step 1]
WOR FIRST_NAME LAST_NAME            CITY
--- ---------- -------------------- ------------------------------
600 Katie      Smith                Chicago

[Step 2]
1 row updated.

[Step 3]
Commit complete.

[Step 4]
WOR FIRST_NAME LAST_NAME            CITY
--- ---------- -------------------- ------------------------------
600 Katie      Kelly                Oak Brook
*/

--
**********************************************************************
********

--Question 3
/*
To update the price of food items in the database. The new owners of
supplier_ID Ard have decided to increase the price of each food item that
they offer.
*/

--Step 1
/*
Run a query that lists the supplier ID, description, and price for each
food
item on the lunch menu from supplier Ard. (FOOD table).
(3 rows selected)
*/
```

```
SELECT supplier_id, description, price
FROM food
WHERE supplier_id = 'Ard'

--Step 2
/*
It is determined that the price of each food item must be increased on
the
menu. Increase the price by 57% for all of the food items selected above.
(3 rows updated)
(DON'T Issue a COMMIT here.)
*/

UPDATE food
SET price = price * (1.57)
WHERE supplier_id = 'Ard';

--Step 3
/*
To verify, rerun the query from step 1, and verify that the price values
are
now increased by 57%.
(3 rows selected)
*/

SELECT supplier_id, description, price
FROM food
WHERE supplier_id = 'Ard'

/*
Results:
[Step 1]
SUP DESCRIPTION               PRICE
--- -------------------- ----------
Ard PB Cookie                   1.25
Ard Veggie Pizza                6.25
Ard Chicken Avocado Wrap        5.25

[Step 2]
3 rows updated.

[Step 3]
SUP DESCRIPTION               PRICE
--- -------------------- ----------
Ard PB Cookie                   1.96
Ard Veggie Pizza                9.81
Ard Chicken Avocado Wrap        8.24
*/

--
**************************************************************************
********
--Question 4
```

```
--Deletion of food records that havenâ€™t been purchased as lunch items.

--Step 1
/*
List the supplier ID and the product code for any food items that have
never
been purchased as lunch items.
ï,· In the SELECT clause, select the supplier id and the product code
ï,· In the FROM clause, use a left join from FOOD to LUNCH_ITEM (donâ€™t
forget the ON clause with the composite PK relationships)
ï,· In the WHERE clause filter that the lunch item supplier id is NULL
ï,· 4 rows selected
*/

SELECT f.supplier_id, f.product_code
FROM food f LEFT JOIN lunch_item li
ON f.supplier_id = li.supplier_id AND f.product_code = li.product_code
WHERE li.supplier_id IS NULL;

--Step 2
/*
Remove these food items from the database.
ï,· Type the delete from FOOD as the first line
ï,· The next line is the where clause. Concatenate the supplier id and
product
code into one field because we will be comparing this one field to a
nested query.
ï,· Be sure to use the IN condition to compare to the nested query.
ï,· Place the query from step 1 into parentheses as a nested query.
ï,· NOTE: In order to compare and match the records selected by the
LEFT JOIN in step 1, we have to first concatenate supplier_id and
product_code into one field in the SELECT clause.
ï,· 4 records removed
*/

DELETE FROM food
WHERE supplier_id ||''|| product_code IN (SELECT f.supplier_id ||''||
f.product_code
    FROM food f LEFT JOIN lunch_item li
    ON f.supplier_id = li.supplier_id AND f.product_code =
li.product_code
    WHERE li.supplier_id IS NULL);

--Step 3
/*
Verify that they have been removed by rerunning step 1.
(no rows selected)
*/

SELECT f.supplier_id, f.product_code
FROM food f LEFT JOIN lunch_item li
ON f.supplier_id = li.supplier_id AND f.product_code = li.product_code
WHERE li.supplier_id IS NULL;
```

```
/*
Results:
[Step 1]
SUP PR
--- --
Ard Sw
Jmd Vt
Ard Ds
Gls Vr

[Step 2]
4 rows deleted.

[Step 3]
no rows selected
*/

--
**************************************************************************
********
--Question 5
/*
Deletion of supplier records that havenâ€™t supplied food items. This
time with
a NOT IN nested query.
*/

--Step 1
/*
List the Supplier ID, Supplier name of the food suppliers with no food
items
(Use NOT IN with a nested query). (6 records selected)
*/

SELECT supplier_id, supplier_name
FROM food_supplier
WHERE supplier_id NOT IN (SELECT supplier_id FROM food);

--Step 2
/*
Remove these suppliers from the database. (6 records deleted)
*/

DELETE FROM food_supplier
WHERE supplier_id NOT IN (SELECT supplier_id FROM food);

--Step 3
/*
Verify that they have been removed by rerunning step 1.
(No rows selected)
*/

SELECT supplier_id, supplier_name
FROM food_supplier
```

```
WHERE supplier_id NOT IN (SELECT supplier_id FROM food);

--Step 4
/*
Issue a ROLLBACK statement
*/

ROLLBACK;

--Step 5
/*
Run Step 1 again to see that the rollback was done correctly
(6 records selected)
*/

SELECT supplier_id, supplier_name
FROM food_supplier
WHERE supplier_id NOT IN (SELECT supplier_id FROM food);


/*
Results:
[Step 1]
SUP SUPPLIER_NAME
--- ------------------------------
Fas Framer and Samson
Fdv Fresh Daily Vegetables
Gio Giovana and Sons
Har Harold Bakery
Met Under the Metra
Rby Rosemont Bakery

6 rows selected.

[Step 2]
6 rows deleted.

[Step 3]
no rows selected

[Step 4]
Rollback complete.

[Step 6]
SUP SUPPLIER_NAME
--- ------------------------------
Fas Framer and Samson
Fdv Fresh Daily Vegetables
Gio Giovana and Sons
Har Harold Bakery
Met Under the Metra
Rby Rosemont Bakery

6 rows selected.
```

```
*/

--
**************************************************************************
********
--Question 6
--DDL + DML

--Step 1
/*
Create a new table called TRAVEL (Drop the table first if there is
already
one existing table). This table explains the columns and constraints
found
in the table you are creating:
*/

CREATE TABLE TRAVEL
(
Worker_ID char(3),
Dept_Code varchar2(4),
Travel_limit number(5,2),
Authorization char(2),
CONSTRAINT WorkerID_PK PRIMARY KEY(Worker_ID)
);

--Step 2
/*
Alter the TRAVEL table by adding a foreign key constraint (use constraint
name: travel_FK). The TRAVEL table references the WORKER table. The
Worker_ID is the foreign key in the TRAVEL table and the primary key in
the WORKER table.
(Table TRAVEL altered.)
*/

ALTER TABLE TRAVEL
ADD CONSTRAINT TRAVEL_FK FOREIGN KEY (Worker_ID) REFERENCES WORKER
(Worker_ID);

--Step 3
/*
Using data from the WORKER table for only the employees that work for
Tracy Reynolds (manager_ID 562), insert rows into TRAVEL with one
difference: Set the credit limit at 50% higher than what is stored in the
WORKER table. The increased credit_limit will be stored as travel_limit.
Hint: Review how to use INSERT INTO with a select statement, and also
use the calculation for the credit limit: (Credit_Limit +
Credit_limit*.50).
>> Show the new TRAVEL data by selecting all columns in the table.
(2 or 3 rows inserted, 2 or 3 rows selected.)
*/

INSERT INTO TRAVEL (Worker_ID, Dept_Code, Travel_limit)
```

```
SELECT worker_id, dept_code, (credit_limit + (credit_limit * 0.5)) AS
Travel_limit FROM worker
WHERE manager_id = '562';

--Step 4
/*
Update the authorization code of these employees: Update all of the
records in the TRAVEL table with a value of A5 for the Authorization
field.
(2 or 3 rows updated)
*/

UPDATE TRAVEL
SET Authorization = 'A5';

--Step 5
/*
Verify the new table. Run a query that joins the WORKER table and the
TRAVEL table. List the worker id, the department code, credit limit,
travel limit, and authorization.
(2 or 3 rows selected)
*/

SELECT *
FROM travel;

SELECT w.worker_id, w.dept_code, credit_limit, travel_limit,
authorization
FROM worker w INNER JOIN travel t
ON w.worker_id = t.worker_id;

--Step 6
/*
Issue a COMMIT
*/

COMMIT;

/*
Results:
[Step 1]
Table TRAVEL created.

[Step 2]
Table TRAVEL altered.

[Step 3]
2 rows inserted.

[Step 4]
2 rows updated.

[Step 5-A]
WOR DEPT TRAVEL_LIMIT AU
```

```
--- ---- ------------ --
574                48 A5
583              37.5 A5

[Step 5-B]
WOR DEP CREDIT_LIMIT TRAVEL_LIMIT AU
--- --- ------------ ------------ --
574                32           48 A5
583                25         37.5 A5

[Step 6]
Commit complete.
*/
--
********************************************************************************
********
```