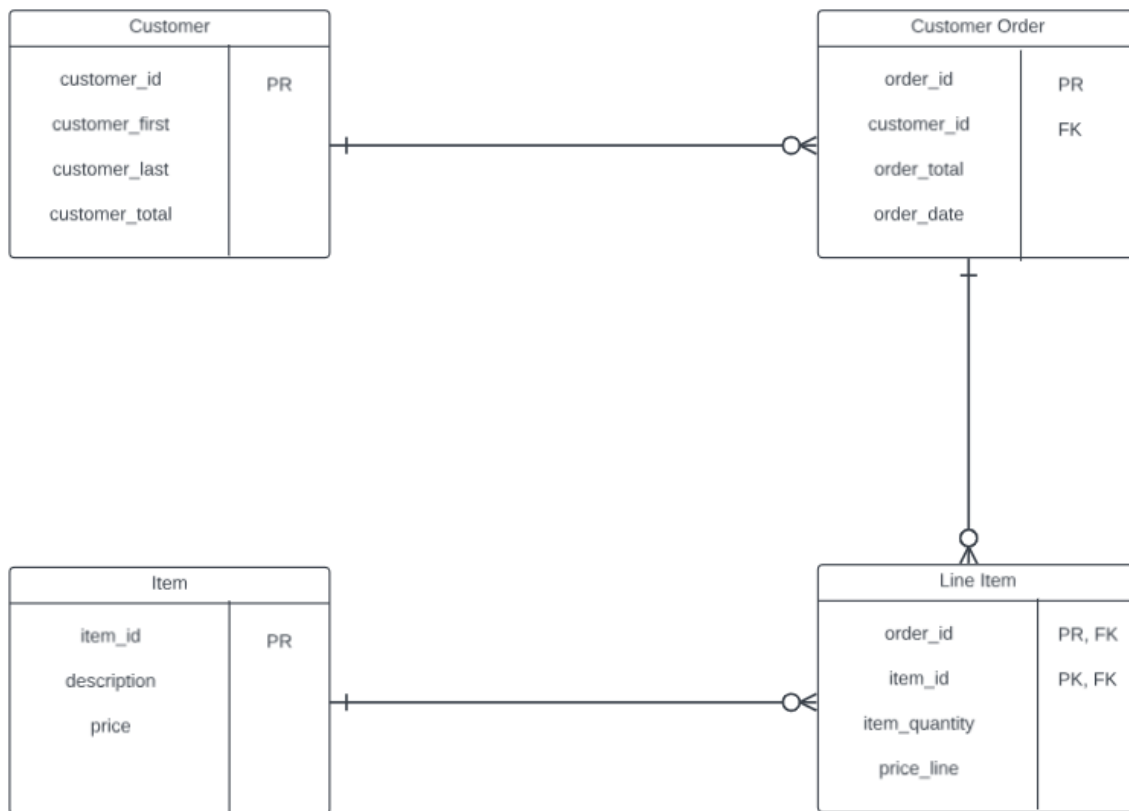


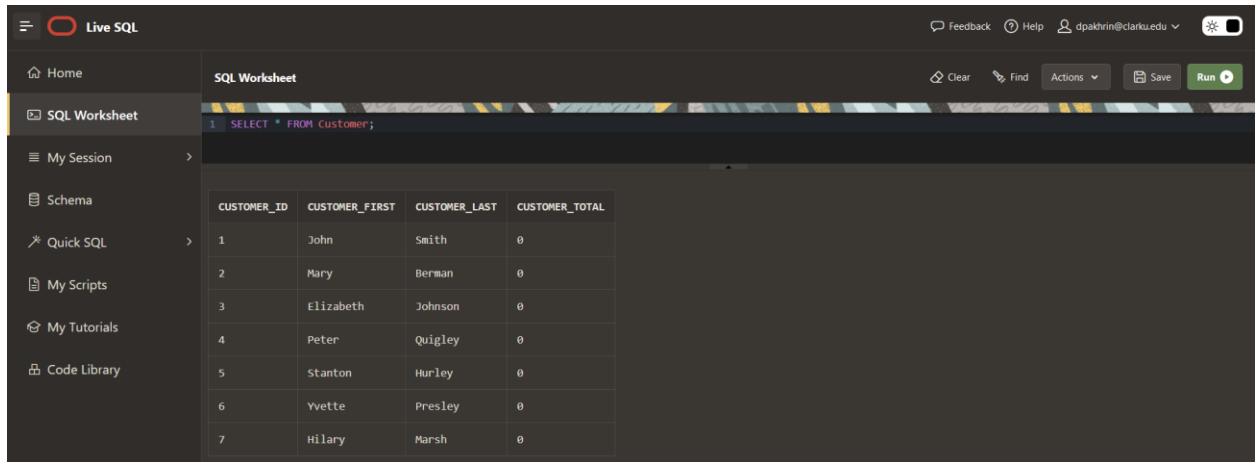
Lab 4 – Submission Sheet

Section One

1. ERD



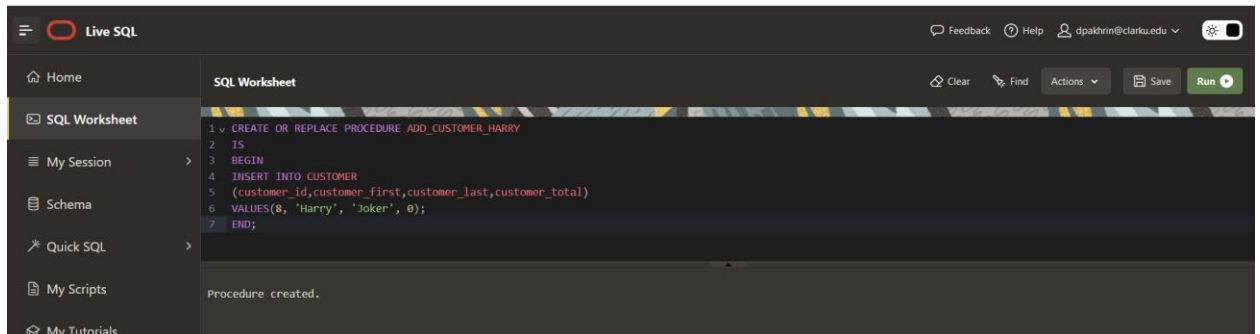
2. SELECT (Screen Shots)



The screenshot shows the Live SQL interface. The SQL Worksheet contains the query: `SELECT * FROM customer;`. The results are displayed in a table with the following data:

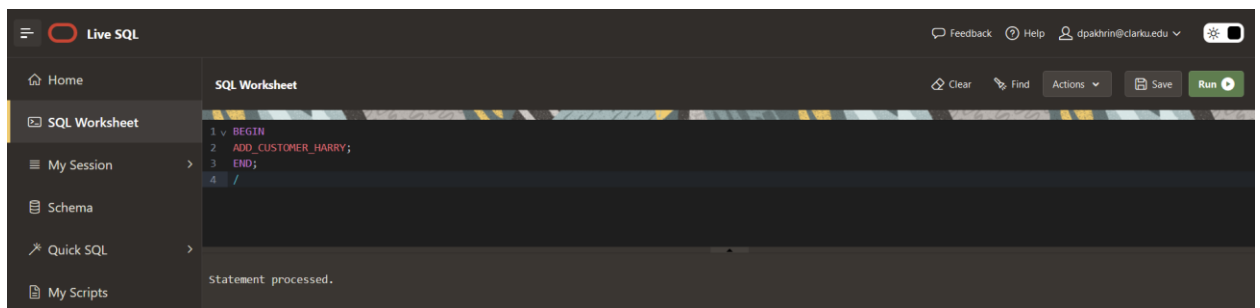
CUSTOMER_ID	CUSTOMER_FIRST	CUSTOMER_LAST	CUSTOMER_TOTAL
1	John	Smith	0
2	Mary	Berman	0
3	Elizabeth	Johnson	0
4	Peter	Quigley	0
5	Stanton	Hurley	0
6	Yvette	Presley	0
7	Hilary	Marsh	0

3. CREATE PROCEDURE (Screen Shot)



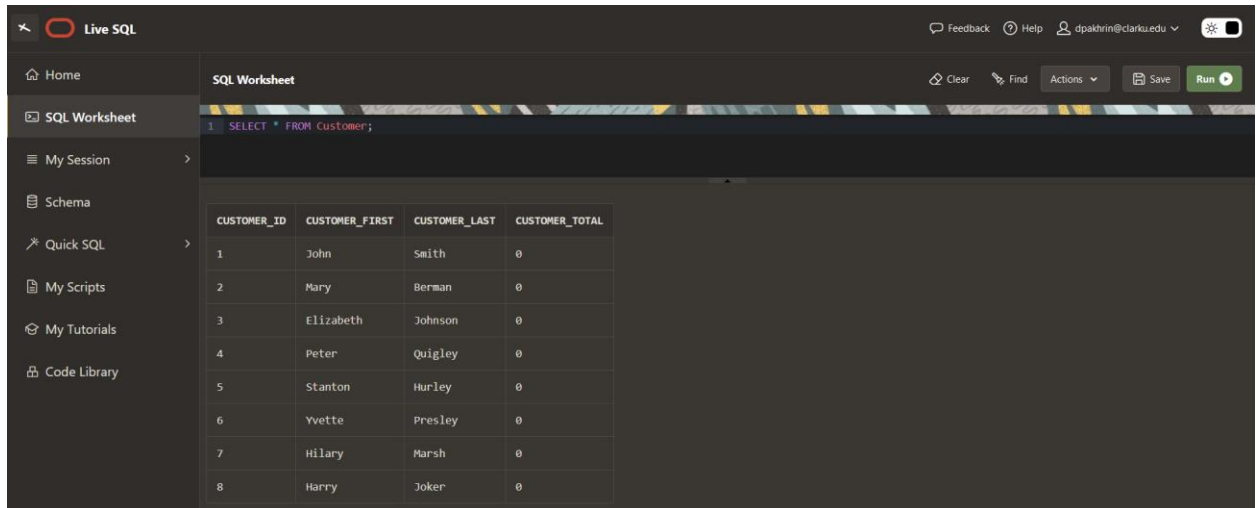
The screenshot shows the Live SQL interface. The SQL Worksheet contains the query: `CREATE OR REPLACE PROCEDURE ADD_CUSTOMER_HARRY IS BEGIN INSERT INTO CUSTOMER (customer_id, customer_first, customer_last, customer_total) VALUES (8, 'Harry', 'Joker', 0); END;`. The results display shows: `Procedure created.`

4. EXECUTE PROCEDURE (Screen Shot)



The screenshot shows the Live SQL interface. The SQL Worksheet contains the query: `BEGIN ADD_CUSTOMER_HARRY; END;`. The results display shows: `Statement processed.`

5. SELECT (Screen Shot)



The screenshot shows the Live SQL interface with the following components:

- Top Bar:** "Live SQL" logo, Feedback, Help, and user profile (dpakhrin@clarku.edu).
- Left Sidebar:** Home, SQL Worksheet (selected), My Session, Schema, Quick SQL, My Scripts, My Tutorials, and Code Library.
- SQL Worksheet:** Contains the query `1 SELECT * FROM Customer;`.
- Results:** A table with 4 columns: `CUSTOMER_ID`, `CUSTOMER_FIRST`, `CUSTOMER_LAST`, and `CUSTOMER_TOTAL`. It contains 8 rows of data.

CUSTOMER_ID	CUSTOMER_FIRST	CUSTOMER_LAST	CUSTOMER_TOTAL
1	John	Smith	0
2	Mary	Berman	0
3	Elizabeth	Johnson	0
4	Peter	Quigley	0
5	Stanton	Hurley	0
6	Vvette	Presley	0
7	Hilary	Marsh	0
8	Harry	Joker	0

6. EXECUTE PROCEDURE (Screen Shot)



The screenshot shows the Live SQL interface with the following components:

- Top Bar:** "Live SQL" logo, Feedback, Help, and user profile (dpakhrin@clarku.edu).
- Left Sidebar:** Home, SQL Worksheet (selected), My Session, Schema, Quick SQL, My Scripts, My Tutorials, and Code Library.
- SQL Worksheet:** Contains the query `1 BEGIN; 2 ADD_CUSTOMER_HARRY; 3 END; 4 /`.
- Results:** An error message: `ORA-00001: unique constraint (SQL_SYZWKSTCEHOUUEUWYCBRP.SYS_C00151368866) violated ORA-06512: at "SQL_SYZWKSTCEHOUUEUWYCBRP.ADD_CUSTOMER_HARRY", line 4 ORA-06512: at line 2 ORA-06512: at "SYS.DBMS_SQL", line 1721`. A link to Oracle error details is provided: <https://docs.oracle.com/error-help/db/ora-00001>.

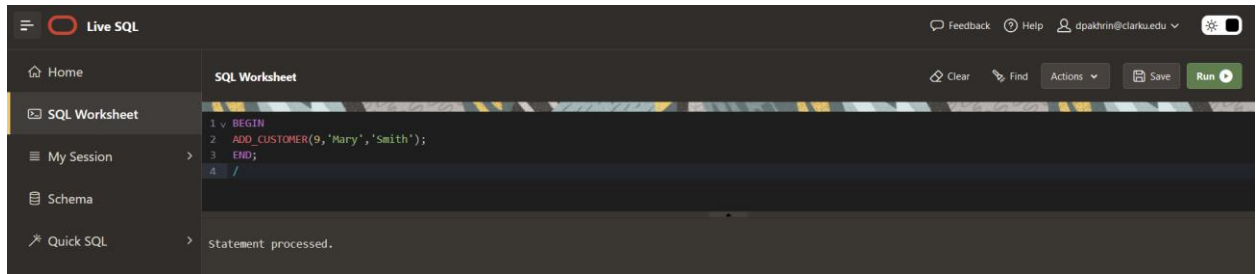
7. CREATE PROCEDURE (Screen Shot)



The screenshot shows the Live SQL interface with the following components:

- Top Bar:** "Live SQL" logo, Feedback, Help, and user profile (dpakhrin@clarku.edu).
- Left Sidebar:** Home, SQL Worksheet (selected), My Session, Schema, Quick SQL, My Scripts, My Tutorials, and Code Library.
- SQL Worksheet:** Contains the query `1 CREATE OR REPLACE PROCEDURE ADD_CUSTOMER(2 cust_id_arg IN DECIMAL, 3 first_name_arg IN VARCHAR, 4 last_name_arg IN VARCHAR 5) 6 IS 7 BEGIN 8 INSERT INTO CUSTOMER 9 (customer_id,customer_first,customer_last,customer_total) 10 VALUES(cust_id_arg,first_name_arg,last_name_arg,0); 11 END;`.
- Results:** The message "Procedure created."

8. EXECUTE PROCEDURE (Screen Shot)

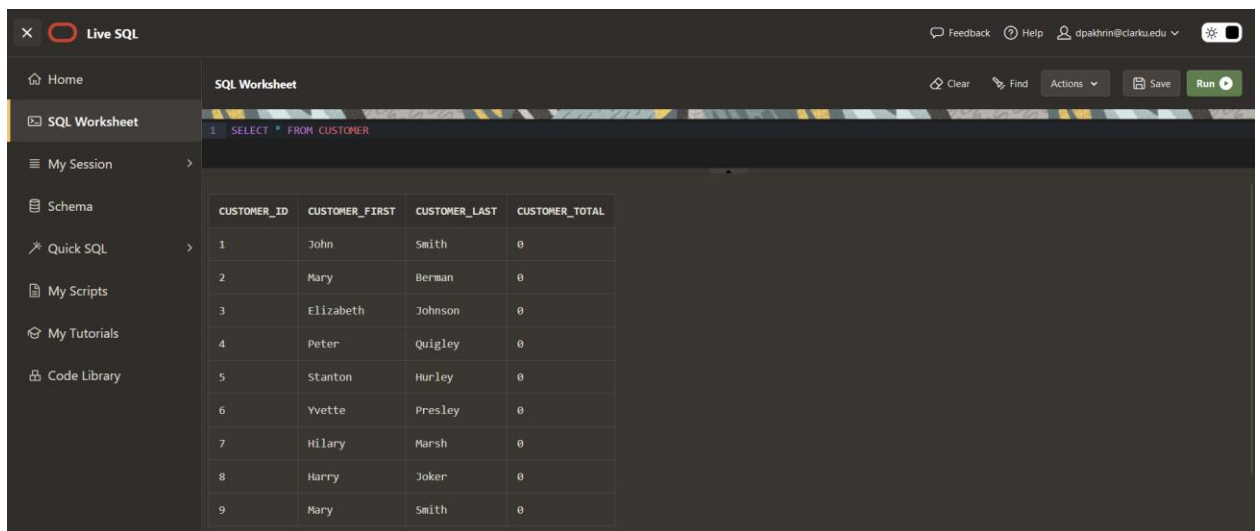


Live SQL interface showing an SQL Worksheet. The worksheet contains the following SQL code:

```
1 BEGIN
2 ADD_CUSTOMER(9,'Mary','Smith');
3 END;
4 /
```

The status bar at the bottom indicates "Statement processed."

9. SELECT (Screen Shot)



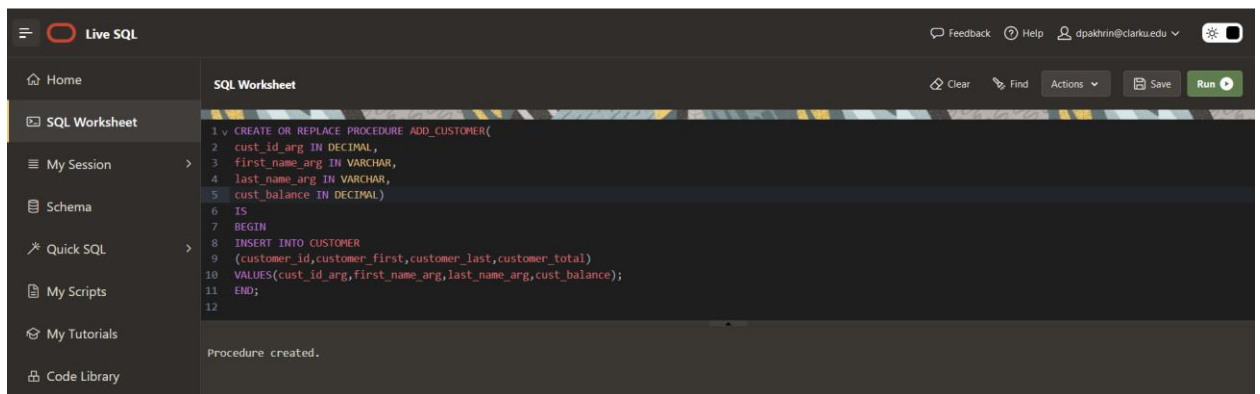
Live SQL interface showing an SQL Worksheet. The worksheet contains the following SQL code:

```
1 SELECT * FROM CUSTOMER
```

The results are displayed in a table with the following columns: CUSTOMER_ID, CUSTOMER_FIRST, CUSTOMER_LAST, and CUSTOMER_TOTAL.

CUSTOMER_ID	CUSTOMER_FIRST	CUSTOMER_LAST	CUSTOMER_TOTAL
1	John	Smith	0
2	Mary	Berman	0
3	Elizabeth	Johnson	0
4	Peter	Quigley	0
5	Stanton	Hurley	0
6	Yvette	Presley	0
7	Hilary	Marsh	0
8	Harry	Joker	0
9	Mary	Smith	0

10. CREATE PROCEDURE (Screen Shot)

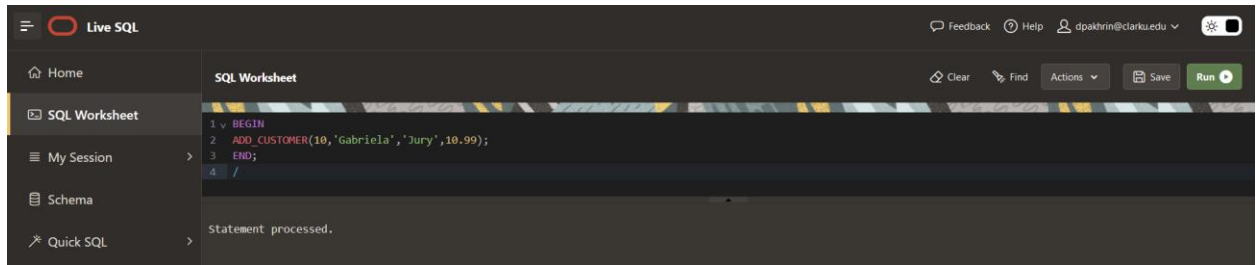


Live SQL interface showing an SQL Worksheet. The worksheet contains the following SQL code:

```
1 CREATE OR REPLACE PROCEDURE ADD_CUSTOMER(
2 cust_id_arg IN DECIMAL,
3 first_name_arg IN VARCHAR,
4 last_name_arg IN VARCHAR,
5 cust_balance IN DECIMAL)
6 IS
7 BEGIN
8 INSERT INTO CUSTOMER
9 (customer_id,customer_first,customer_last,customer_total)
10 VALUES(cust_id_arg,first_name_arg,last_name_arg,cust_balance);
11 END;
12
```

The status bar at the bottom indicates "Procedure created."

11. EXECUTE PROCEDURE (Screen Shot)



Live SQL

SQL Worksheet

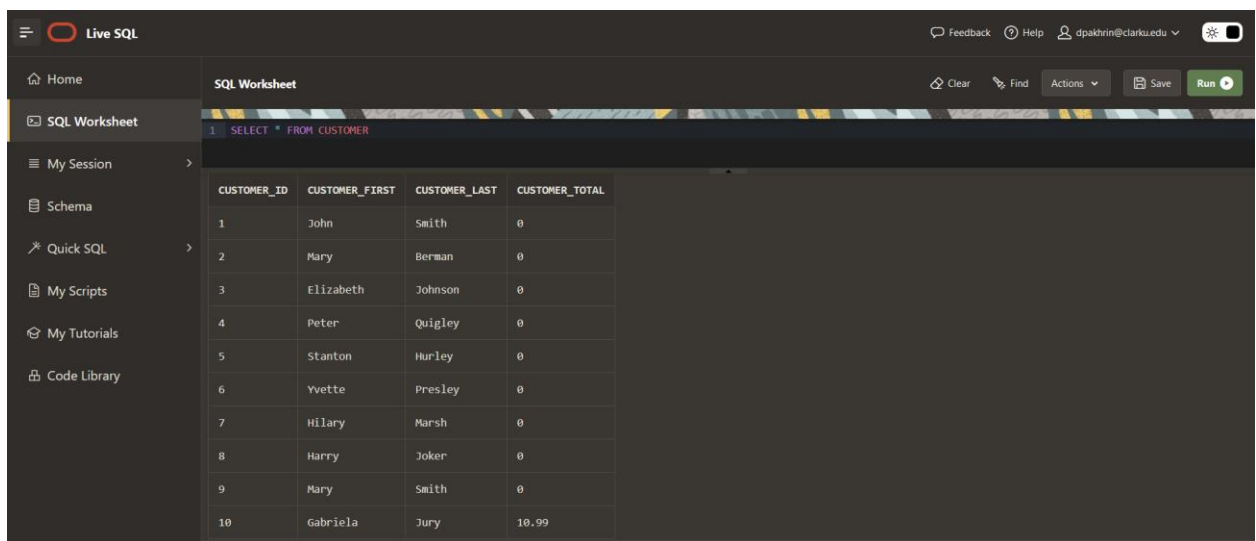
```

1 BEGIN
2 ADD_CUSTOMER(10, 'Gabriela', 'Jury', 10.99);
3 END;
4 /

```

Statement processed.

12. SELECT (Screen Shot)



Live SQL

SQL Worksheet

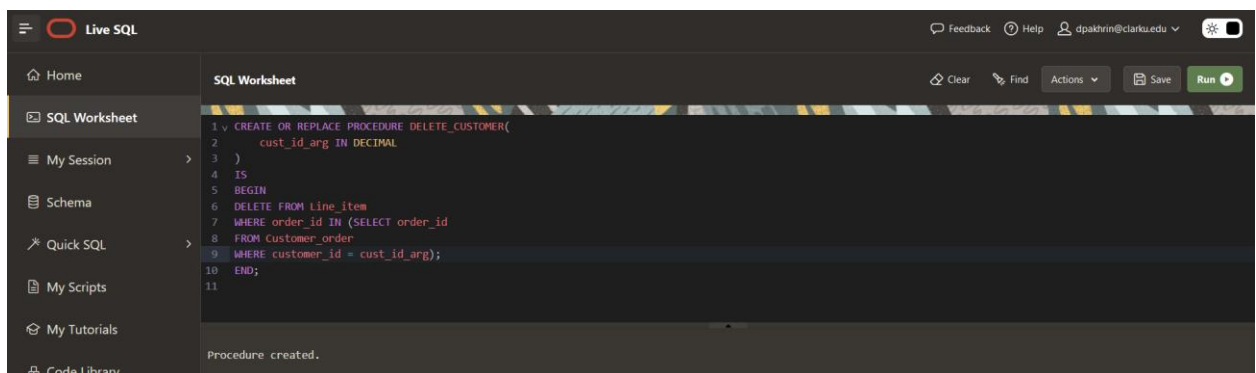
```

1 SELECT * FROM CUSTOMER

```

CUSTOMER_ID	CUSTOMER_FIRST	CUSTOMER_LAST	CUSTOMER_TOTAL
1	John	Smith	0
2	Mary	Berman	0
3	Elizabeth	Johnson	0
4	Peter	Quigley	0
5	Stanton	Hurley	0
6	Yvette	Presley	0
7	Hilary	Marsh	0
8	Harry	Joker	0
9	Mary	Smith	0
10	Gabriela	Jury	10.99

13. CREATE & EXECUTE PROCEDURE, VERIFY RESULTS (Screen Shots)



Live SQL

SQL Worksheet

```

1 CREATE OR REPLACE PROCEDURE DELETE_CUSTOMER(
2   cust_id_arg IN DECIMAL
3 )
4 IS
5 BEGIN
6   DELETE FROM Line_Item
7   WHERE order_id IN (SELECT order_id
8   FROM Customer_order
9   WHERE customer_id = cust_id_arg);
10 END;
11

```

Procedure created.

Live SQL

SQL Worksheet

```
1 BEGIN
2 DELETE_CUSTOMER(5);
3 END;
4 /
```

Statement processed.

Live SQL

SQL Worksheet

```
1 SELECT * FROM Line_item
```

ORDER_ID	ITEM_ID	ITEM_QUANTITY	LINE_PRICE
1	1	10	100
1	5	2	10
1	2	36	396
2	1	95	950
2	3	10	50
3	4	3	15
4	4	3	15
5	6	132	1584
6	1	10	100
8	5	2	10

Section Two

14. CREATE TRIGGER (Screen Shot)

Live SQL

SQL Worksheet

```
1 CREATE OR REPLACE TRIGGER no_neg_cust_bal_trg
2 BEFORE UPDATE OR INSERT ON customer
3 FOR EACH ROW
4 BEGIN
5 IF :NEW.customer_total < 0 THEN
6 RAISE_APPLICATION_ERROR(-20001,'customer balance cannot be
7 negative.');
```

```
8 END IF;
9 END;
```

Trigger created.

15. INSERT (Screen Shot)



The screenshot shows the Live SQL interface with the following SQL code:

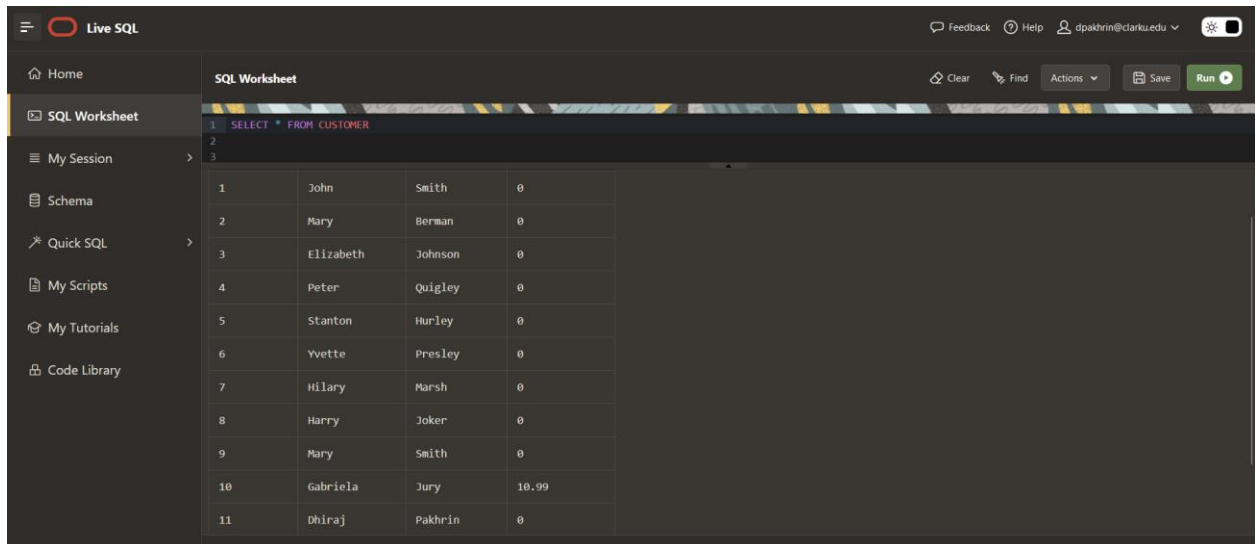
```
1 INSERT INTO CUSTOMER
2 VALUES(11,'Dhiraj','Pakhrin',0);
3
4 INSERT INTO CUSTOMER
5 VALUES(12,'Poonam','kulung',-20.25);
6
7
```

The output shows "1 row(s) inserted." followed by an error message:

```
ORA-20001: Customer balance cannot be
negative. ORA-06512: at "SQL_TKIDYTELHPTMAQPAJOWIZRY.NO_NEW_CUST_BAL_TRG", line 3
ORA-06512: at "SYS.DBMS_SQL", line 1721
```

More Details: <https://docs.oracle.com/error-help/db/ora-20001>

16. SELECT (Screen Shot)




The screenshot shows the Live SQL interface with the following SQL code:

```
1 SELECT * FROM CUSTOMER
2
3
```

The output is a table with 11 rows:

1	John	Smith	0
2	Mary	Berman	0
3	Elizabeth	Johnson	0
4	Peter	Quigley	0
5	Stanton	Hurley	0
6	Yvette	Presley	0
7	Hilary	Marsh	0
8	Harry	Joker	0
9	Mary	Smith	0
10	Gabriela	Jury	10.99
11	Dhiraj	Pakhrin	0

17. CREATE TRIGGER (Screen Shot)



The screenshot shows the Live SQL interface with the following SQL code:

```
1 CREATE OR REPLACE TRIGGER block_last_name
2 BEFORE UPDATE OR INSERT ON customer
3 FOR EACH ROW
4 BEGIN
5 IF :NEW.customer_last = 'Glass' THEN
6 RAISE_APPLICATION_ERROR(-20001,'Customers with last name "Glass" are not allowed.');
```

The output shows "Trigger created."

18. INSERT (Screen Shot)



The screenshot shows the Live SQL interface. The SQL Worksheet contains the following code:

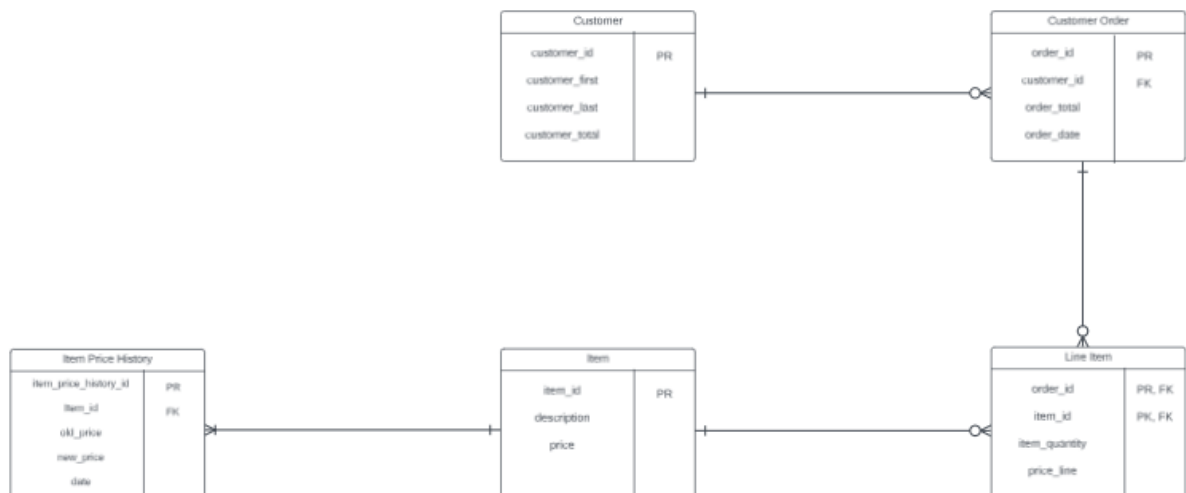
```
1 INSERT INTO CUSTOMER
2 VALUES(12, 'Poonam', 'Glass', 0);
```

The output area shows an error message:

```
ORA-20001: Customers with last name "Glass" are not allowed. ORA-00512: at "SQL_TXQYTELHFYTNQBAJOMJZBRY.BLOCK_LAST_NAME", line 1
ORA-00512: at "SYS.DBMS_SQL", line 1721
```

More Details: <https://docs.oracle.com/error-help/db/ora-20001>

19. UPDATED ERD



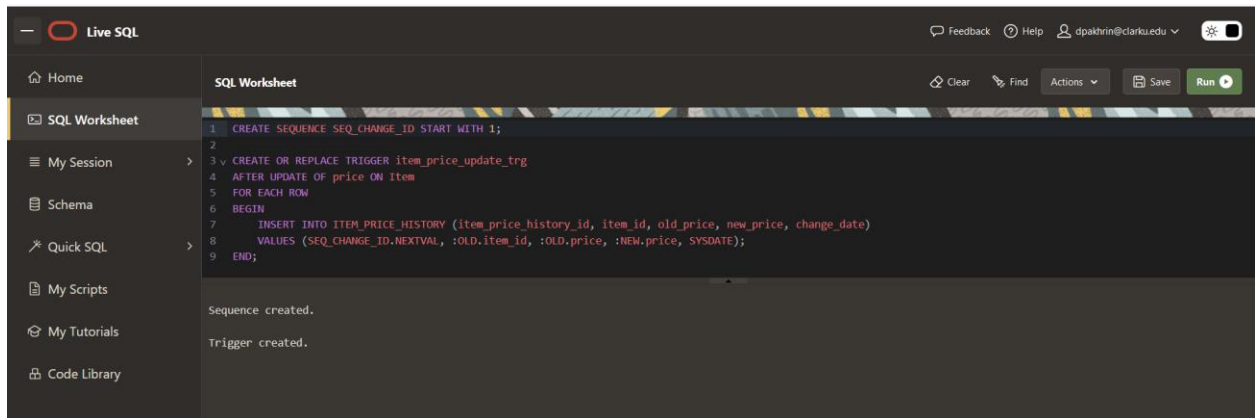
20. CREATE TRIGGER (Screen Shot and Logic Explanation)



The screenshot shows the Live SQL interface. The SQL Worksheet contains the following code:

```
1 CREATE TABLE ITEM_PRICE_HISTORY (
2   item_price_history_id DECIMAL(10) PRIMARY KEY,
3   item_id DECIMAL(10),
4   old_price DECIMAL(10, 2),
5   new_price DECIMAL(10, 2),
6   change_date DATE,
7   FOREIGN KEY (item_id) REFERENCES Item(item_id)
8 );
```

The output area shows the message: "Table created."



```

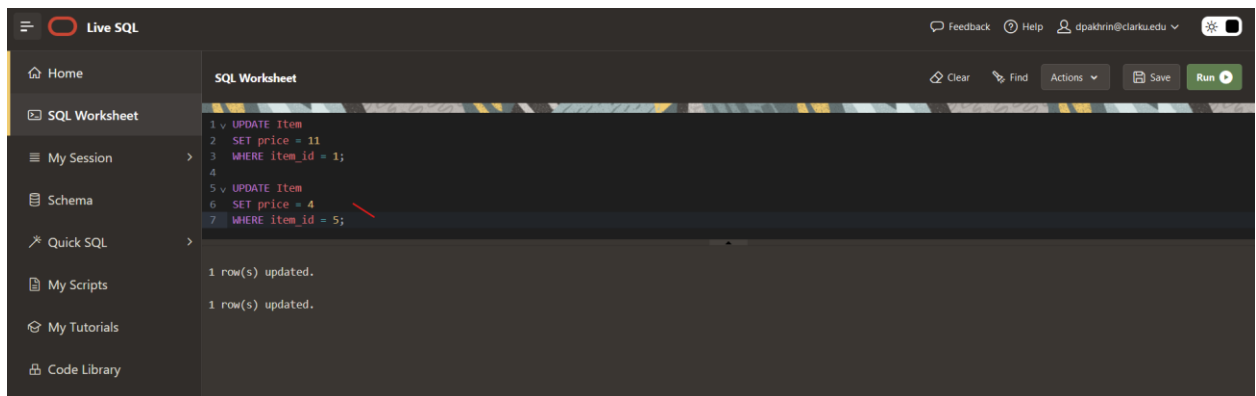
1 CREATE SEQUENCE SEQ_CHANGE_ID START WITH 1;
2
3 CREATE OR REPLACE TRIGGER item_price_update_trg
4 AFTER UPDATE OF price ON item
5 FOR EACH ROW
6 BEGIN
7     INSERT INTO ITEM_PRICE_HISTORY (item_price_history_id, item_id, old_price, new_price, change_date)
8     VALUES (SEQ_CHANGE_ID.NEXTVAL, :OLD.item_id, :OLD.price, :NEW.price, SYSDATE);
9 END;

```

Sequence created.
Trigger created.

Firstly, we create a new table named item_price_history to store value of price being changed in the item table. After that, we define the trigger named item_price_update trg which is called after the price of the items in the item table is updated. It inserts the record to the item_price_history table whenever it item_price_update_trg is triggered.

21. Price Change and Item_price_history

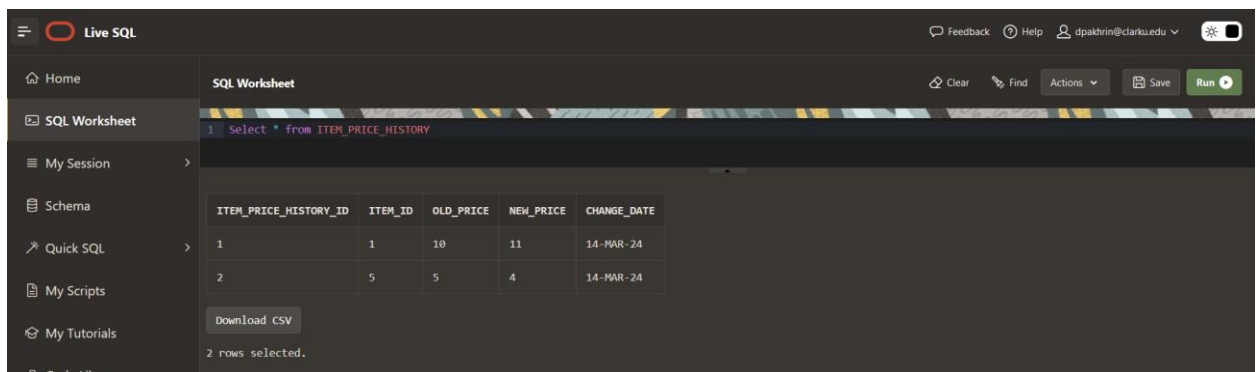


```

1 UPDATE Item
2 SET price = 11
3 WHERE item_id = 1;
4
5 UPDATE Item
6 SET price = 4
7 WHERE item_id = 5;

```

1 row(s) updated.
1 row(s) updated.



```

1 Select * from ITEM_PRICE_HISTORY

```

ITEM_PRICE_HISTORY_ID	ITEM_ID	OLD_PRICE	NEW_PRICE	CHANGE_DATE
1	1	10	11	14-MAR-24
2	5	5	4	14-MAR-24

Download CSV
2 rows selected.

Here, the price of the plate and spoon is updated to the new value 11 and 4 respectively. Since it is updated, it calls the trigger `item_price_update_trg` which is defined in problem 20. Inside the block of this trigger, insert query is defined which records the `item_price_history_id`, `item_id`, `old_price`, `new_price`, and `change_date` of the items which is being displayed in using the select query.