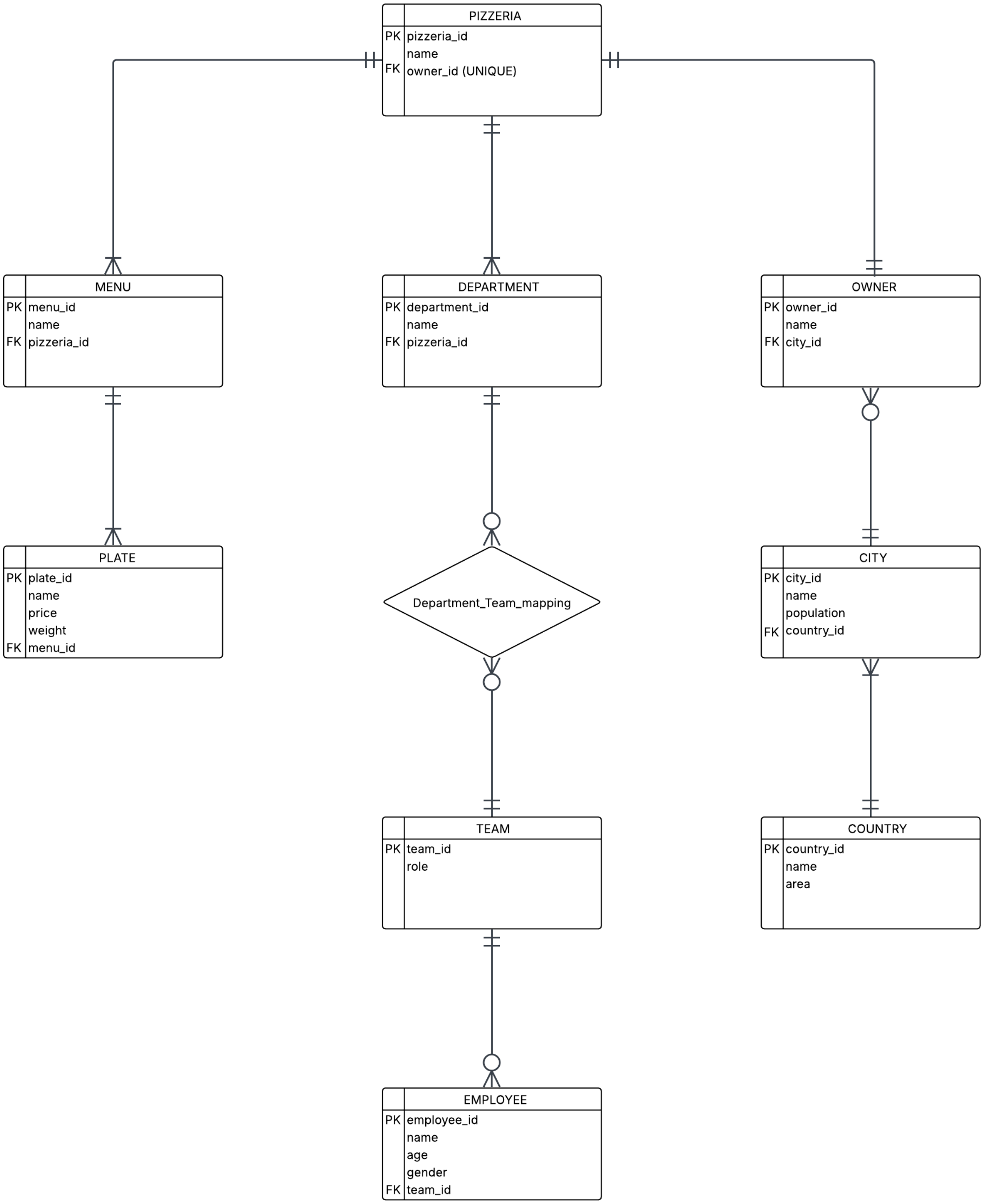
1. Introduction
    a. Brief presentation of the designed model and its rules

The project's data model represents a pizzeria chain with the following key characteristics: each pizzeria has one owner (whose birthplace is recorded through city and country entities), maintains one or more unique menus with distinct plates, and operates through departments that are staffed by teams of employees organized by role.

b. Conceptual diagram



c. Relational schemas

PIZZERIA (pizzeria_id: INT, name: VARCHAR(100), owner_id: INT)
- PK: pizzeria_id
- FK: owner_id REFERENCES OWNER (owner_id)
- UNIQUE: owner_id

MENU (menu_id: INT, name: VARCHAR(100), pizzeria_id: INT)
- PK: menu_id
- FK: pizzeria_id REFERENCES PIZZERIA (pizzeria_id)

PLATE (plate_id: INT, name: VARCHAR(100), price: NUMBER(10,2), weight: INT, menu_id: INT)
- PK: plate_id
- FK: menu_id REFERENCES MENU (menu_id)

DEPARTMENT (department_id: INT, name: VARCHAR(100), pizzeria_id: INT)
- PK: department_id
- FK: pizzeria_id REFERENCES PIZZERIA (pizzeria_id)

DEPARTMENT_TEAM_MAPPING (department_id: INT, team_id: INT)
- PK: (department_id, team_id)
- FK: department_id REFERENCES DEPARTMENT (department_id)
- FK: team_id REFERENCES TEAM (team_id)

TEAM (team_id: INT, role: VARCHAR(100))
- PK: team_id

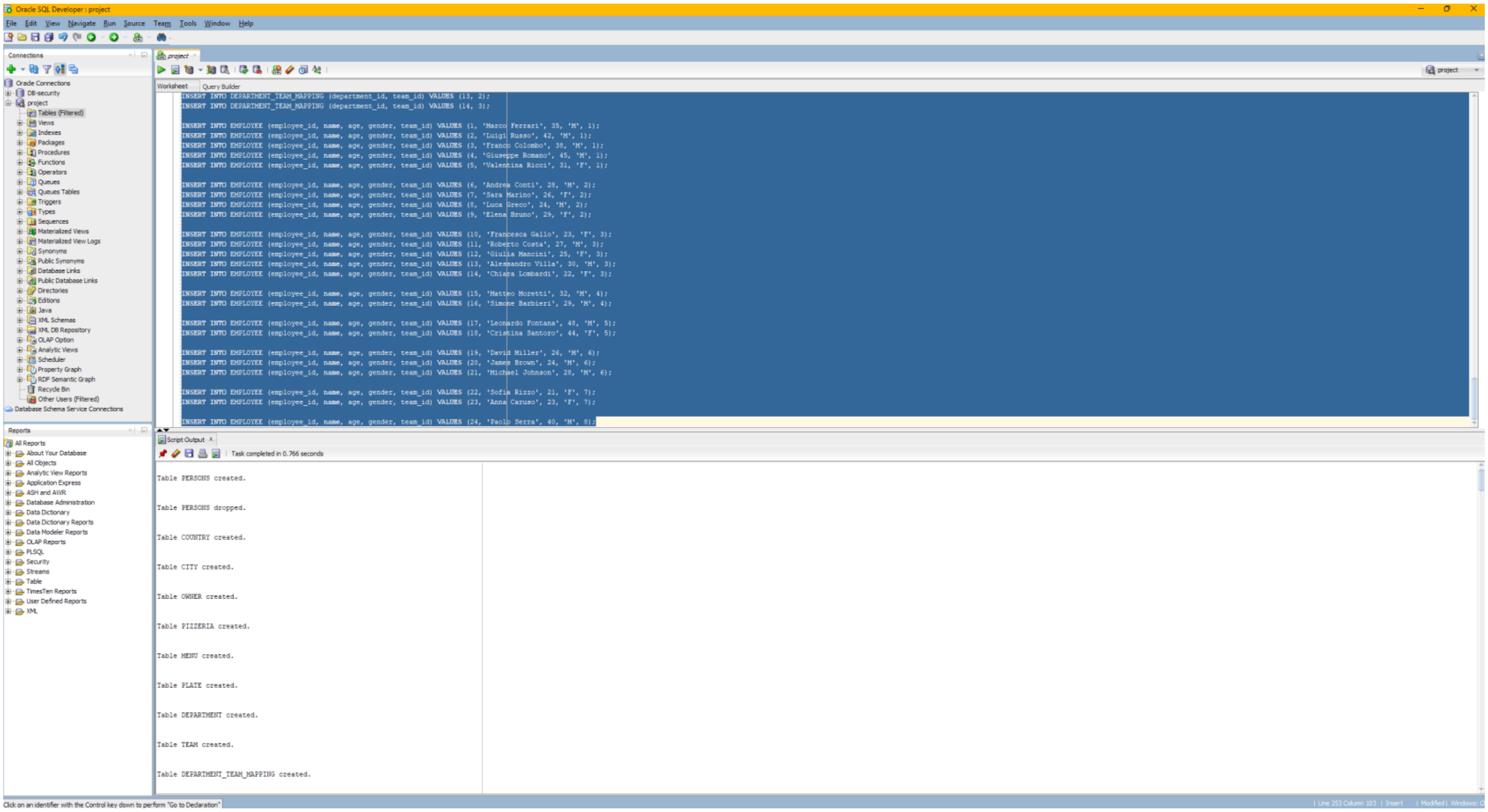
EMPLOYEE (employee_id: INT, name: VARCHAR(100), age: INT, gender: VARCHAR(1), team_id: INT)
- PK: employee_id
- FK: team_id REFERENCES TEAM (team_id)

OWNER (owner_id: INT, name: VARCHAR(100), city_id: INT)
- PK: owner_id
- FK: city_id REFERENCES CITY (city_id)

CITY (city_id: INT, name: VARCHAR(100), population: INT, country_id: INT)
- PK: city_id
- FK: country_id REFERENCES COUNTRY (country_id)

COUNTRY (country_id: INT, name: VARCHAR(100), area: INT)
- PK: country_id

d. Table creation (separate script)
Onutu_Radu-Constantin_510-create_insert.txt

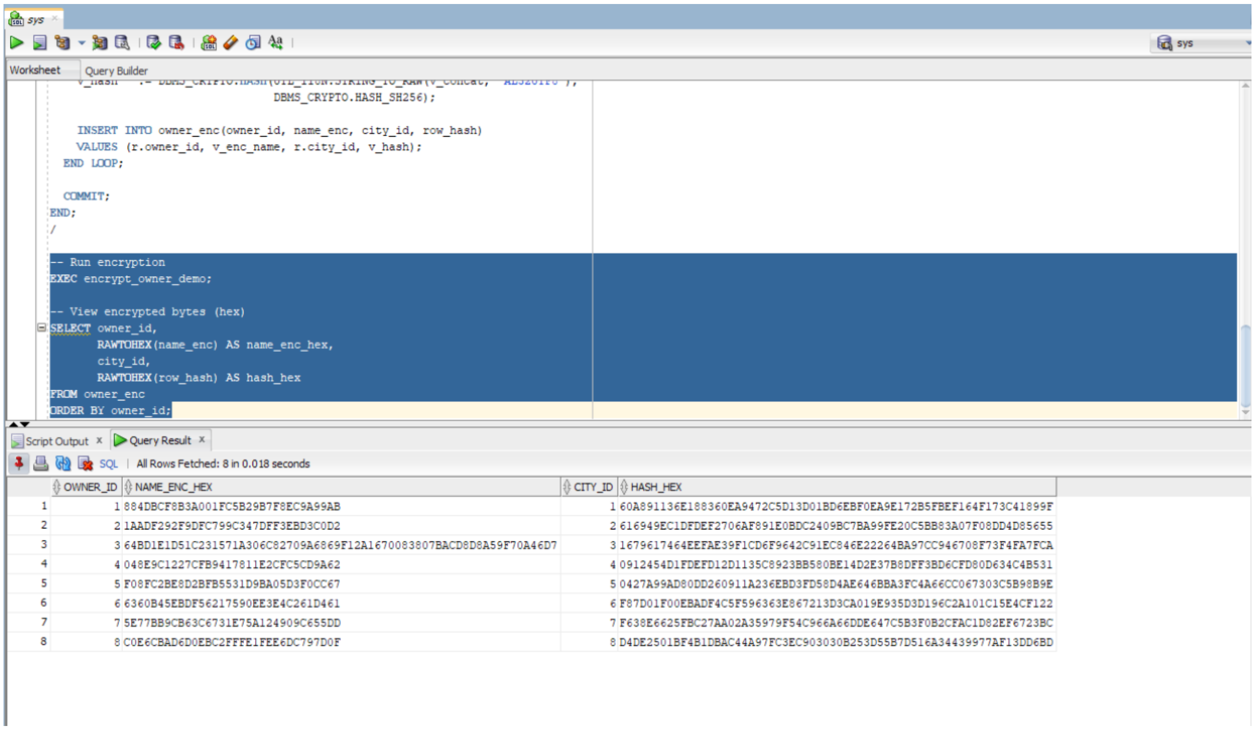

Oracle SQL Developer : project
File Edit View Navigate Run Source Team Tools Window Help
Connections
project
Worksheet
Query Builder
Reports
Script Output
Task completed in 0.766 seconds

```
INSERT INTO DEPARTMENT_TEAM_MAPPING (department_id, team_id) VALUES (13, 2);
INSERT INTO DEPARTMENT_TEAM_MAPPING (department_id, team_id) VALUES (14, 3);

INSERT INTO EMPLOYEE (employee_id, name, age, gender, team_id) VALUES (1, 'Marco Ferrari', 35, 'M', 1);
INSERT INTO EMPLOYEE (employee_id, name, age, gender, team_id) VALUES (2, 'Luigi Russo', 42, 'M', 1);
INSERT INTO EMPLOYEE (employee_id, name, age, gender, team_id) VALUES (3, 'Franco Colombo', 38, 'M', 1);
INSERT INTO EMPLOYEE (employee_id, name, age, gender, team_id) VALUES (4, 'Giuseppe Romano', 45, 'M', 1);
INSERT INTO EMPLOYEE (employee_id, name, age, gender, team_id) VALUES (5, 'Valentina Ricci', 31, 'F', 1);

INSERT INTO EMPLOYEE (employee_id, name, age, gender, team_id) VALUES (6, 'Andrea Conti', 28, 'M', 2);
INSERT INTO EMPLOYEE (employee_id, name, age, gender, team_id) VALUES (7, 'Sara Marino', 26, 'F', 2);
INSERT INTO EMPLOYEE (employee_id, name, age, gender, team_id) VALUES (8, 'Luca Greco', 24, 'M', 2);
INSERT INTO EMPLOYEE (employee_id, name, age, gender, team_id) VALUES (9, 'Elena Bruno', 29, 'F', 2);

INSERT INTO EMPLOYEE (employee_id, name, age, gender, team_id) VALUES (10, 'Francesca Gallo', 23, 'F', 3);
INSERT INTO EMPLOYEE (employee_id, name, age, gender, team_id) VALUES (11, 'Roberto Costa', 27, 'M', 3);
INSERT INTO EMPLOYEE (employee_id, name, age, gender, team_id) VALUES (12, 'Giulia Mancini', 25, 'F', 3);
INSERT INTO EMPLOYEE (employee_id, name, age, gender, team_id) VALUES (13, 'Alessandro Villa', 30, 'M', 3);
INSERT INTO EMPLOYEE (employee_id, name, age, gender, team_id) VALUES (14, 'Chiara Lombardi', 22, 'F', 3);

INSERT INTO EMPLOYEE (employee_id, name, age, gender, team_id) VALUES (15, 'Matteo Moretti', 32, 'M', 4);
INSERT INTO EMPLOYEE (employee_id, name, age, gender, team_id) VALUES (16, 'Simone Barbieri', 29, 'M', 4);

INSERT INTO EMPLOYEE (employee_id, name, age, gender, team_id) VALUES (17, 'Leonardo Fontana', 48, 'M', 5);
INSERT INTO EMPLOYEE (employee_id, name, age, gender, team_id) VALUES (18, 'Cristina Santoro', 44, 'F', 5);

INSERT INTO EMPLOYEE (employee_id, name, age, gender, team_id) VALUES (19, 'David Miller', 26, 'M', 6);
INSERT INTO EMPLOYEE (employee_id, name, age, gender, team_id) VALUES (20, 'James Brown', 24, 'M', 6);
INSERT INTO EMPLOYEE (employee_id, name, age, gender, team_id) VALUES (21, 'Michael Johnson', 28, 'M', 6);

INSERT INTO EMPLOYEE (employee_id, name, age, gender, team_id) VALUES (22, 'Sofia Rizzo', 21, 'F', 7);
INSERT INTO EMPLOYEE (employee_id, name, age, gender, team_id) VALUES (23, 'Anna Caruso', 23, 'F', 7);

INSERT INTO EMPLOYEE (employee_id, name, age, gender, team_id) VALUES (24, 'Paolo Serra', 40, 'M', 8);
```

```
Table PERSONS created.


Table PERSONS dropped.


Table COUNTRY created.


Table CITY created.


Table OWNER created.


Table PIZZERIA created.


Table MENU created.


Table PLATE created.


Table DEPARTMENT created.


Table TEAM created.


Table DEPARTMENT_TEAM_MAPPING created.
```

e. Presentation of the security rules to be applied to the model
Data Encryption, Database activity Auditing, Management of Database Users and Computational Resources, Privileges and Roles, Database Applications and Data Security, Data Masking

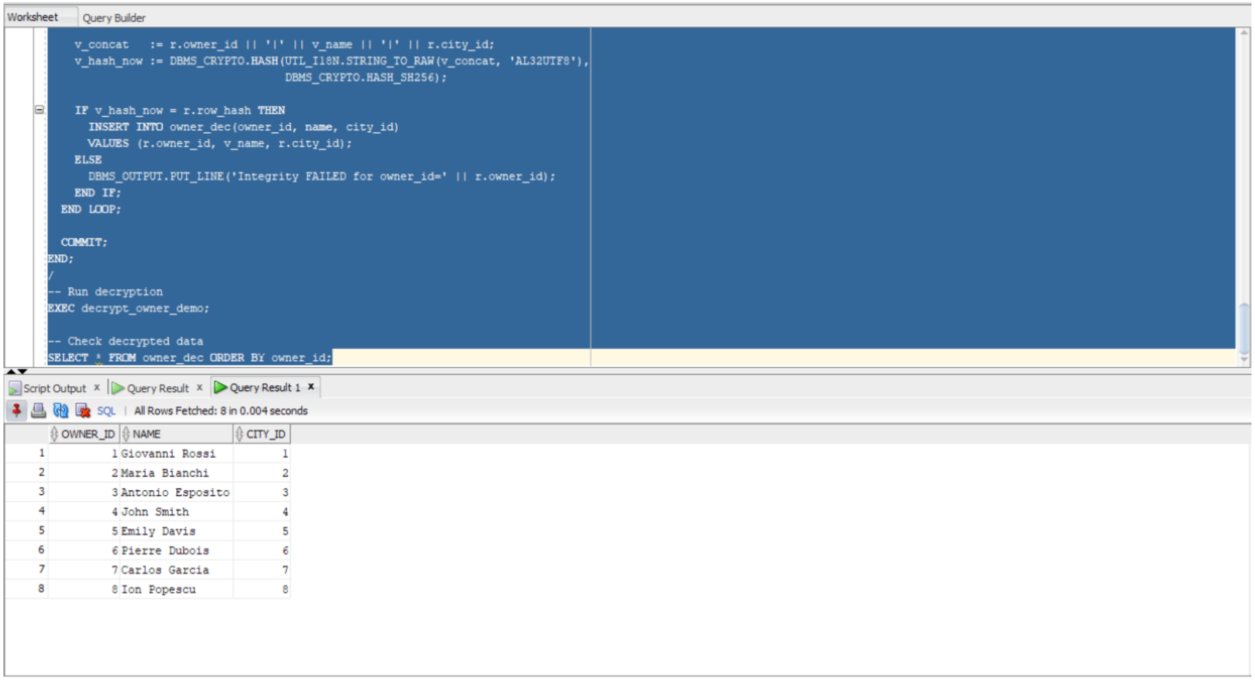
2. Data Encryption
   Onutu_Radu-Constantin_510-encryption.txt



```
    v_hash    := DBMS_CRYPTO.HASH(UTL_I18N.STRING_TO_RAW(v_concat, 'AL32UTF8'),
                                  DBMS_CRYPTO.HASH_SH256);

    INSERT INTO owner_enc(owner_id, name_enc, city_id, row_hash)
    VALUES (r.owner_id, v_enc_name, r.city_id, v_hash);
  END LOOP;

  COMMIT;
END;
/

-- Run encryption
EXEC encrypt_owner_demo;

-- View encrypted bytes (hex)
SELECT owner_id,
       RAWTOHEX(name_enc) AS name_enc_hex,
       city_id,
       RAWTOHEX(row_hash) AS hash_hex
FROM owner_enc
ORDER BY owner_id;
```

All Rows Fetched: 8 in 0.018 seconds

	OWNER_ID	NAME_ENC_HEX	CITY_ID	HASH_HEX
1	1	884DBCF8B3A001FC5B29B7F8EC9A99AB	1	60A891136E188360EA9472C5D13D01BD6EBF0EA9E172B5FBEF164F173C41899F
2	2	1AADF292F9DFC799C347DFF3EBD3C0D2	2	616949EC1DFDEF2706AF891E0BDC2409BC7BA99FE20C5BB83A07F08DD4D85655
3	3	64BD1E1D51C231571A306C82709A6869F12A1670083807BACD8D8A59F70A46D7	3	1679617464EEFAE39F1CD6F9642C91EC846E22264BA97CC946708F73F4FA7FCA
4	4	048E9C1227CFB9417811E2CFC5CD9A62	4	0912454D1FDEFD12D1135C8923BB580BE14D2E37B8DFF3BD6CFD80D634C4B531
5	5	F08FC2BE8D2BFB5531D9BA05D3F0CC67	5	0427A99AD80DD260911A236EBD3FD58D4AE646BBA3FC4A66CC067303C5B98B9E
6	6	6360B45EBDF56217590EE3E4C261D461	6	F87D01F00EBADF4C5F596363E867213D3CA019E935D3D196C2A101C15E4CF122
7	7	5E77BB9CB63C6731E75A124909C655DD	7	F638E6625FBC27AA02A35979F54C966A66DDE647C5B3F0B2CFAC1D82EF6723BC
8	8	C0E6CBAD6D0EBC2FFFE1FEE6DC797D0F	8	D4DE2501BF4B1DBAC44A97FC3EC903030B253D55B7D516A34439977AF13DD6BD



```
    v_concat   := r.owner_id || '|' || v_name || '|' || r.city_id;
    v_hash_now := DBMS_CRYPTO.HASH(UTL_I18N.STRING_TO_RAW(v_concat, 'AL32UTF8'),
                                   DBMS_CRYPTO.HASH_SH256);

    IF v_hash_now = r.row_hash THEN
      INSERT INTO owner_dec(owner_id, name, city_id)
      VALUES (r.owner_id, v_name, r.city_id);
    ELSE
      DBMS_OUTPUT.PUT_LINE('Integrity FAILED for owner_id=' || r.owner_id);
    END IF;
  END LOOP;

  COMMIT;
END;
/
-- Run decryption
EXEC decrypt_owner_demo;

-- Check decrypted data
SELECT * FROM owner_dec ORDER BY owner_id;
```

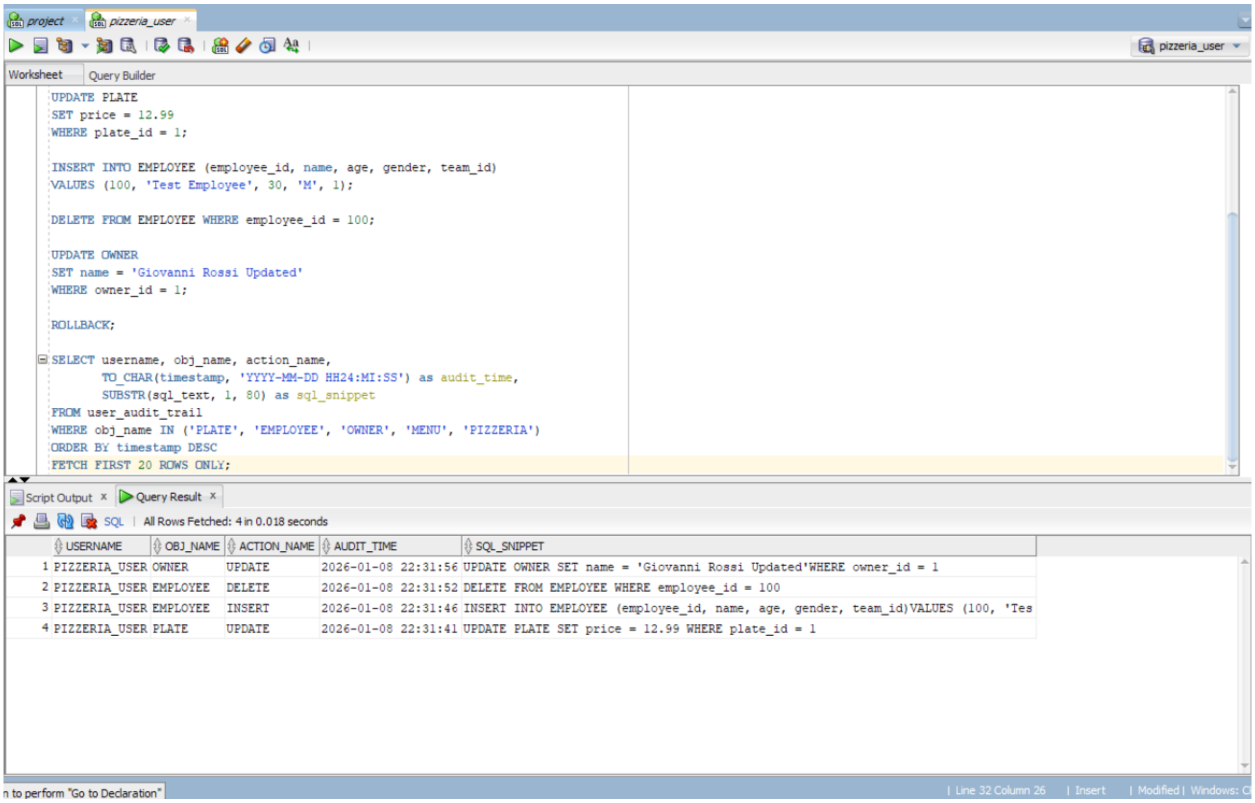
All Rows Fetched: 8 in 0.004 seconds

	OWNER_ID	NAME	CITY_ID
1	1	Giovanni Rossi	1
2	2	Maria Bianchi	2
3	3	Antonio Esposito	3
4	4	John Smith	4
5	5	Emily Davis	5
6	6	Pierre Dubois	6
7	7	Carlos Garcia	7
8	8	Ion Popescu	8

3. Database activity Auditing

   Onutu_Radu-Constantin_510-audit.txt

   a. Standard Auditing

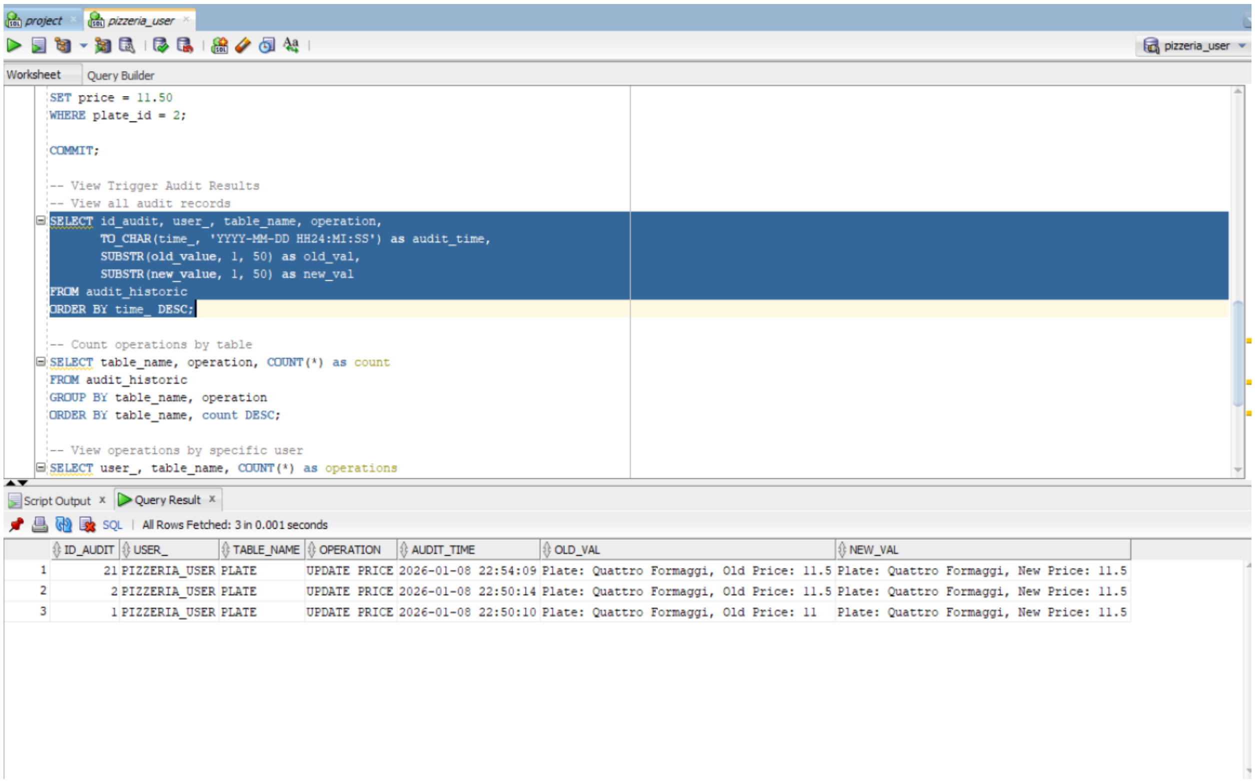


```sql
UPDATE PLATE
SET price = 12.99
WHERE plate_id = 1;

INSERT INTO EMPLOYEE (employee_id, name, age, gender, team_id)
VALUES (100, 'Test Employee', 30, 'M', 1);

DELETE FROM EMPLOYEE WHERE employee_id = 100;

UPDATE OWNER
SET name = 'Giovanni Rossi Updated'
WHERE owner_id = 1;

ROLLBACK;

SELECT username, obj_name, action_name,
       TO_CHAR(timestamp, 'YYYY-MM-DD HH24:MI:SS') as audit_time,
       SUBSTR(sql_text, 1, 80) as sql_snippet
FROM user_audit_trail
WHERE obj_name IN ('PLATE', 'EMPLOYEE', 'OWNER', 'MENU', 'PIZZERIA')
ORDER BY timestamp DESC
FETCH FIRST 20 ROWS ONLY;
```

All Rows Fetched: 4 in 0.018 seconds

	USERNAME	OBJ_NAME	ACTION_NAME	AUDIT_TIME	SQL_SNIPPET
1	PIZZERIA_USER	OWNER	UPDATE	2026-01-08 22:31:56	UPDATE OWNER SET name = 'Giovanni Rossi Updated'WHERE owner_id = 1
2	PIZZERIA_USER	EMPLOYEE	DELETE	2026-01-08 22:31:52	DELETE FROM EMPLOYEE WHERE employee_id = 100
3	PIZZERIA_USER	EMPLOYEE	INSERT	2026-01-08 22:31:46	INSERT INTO EMPLOYEE (employee_id, name, age, gender, team_id)VALUES (100, 'Tes
4	PIZZERIA_USER	PLATE	UPDATE	2026-01-08 22:31:41	UPDATE PLATE SET price = 12.99 WHERE plate_id = 1

   b. Audit Triggers

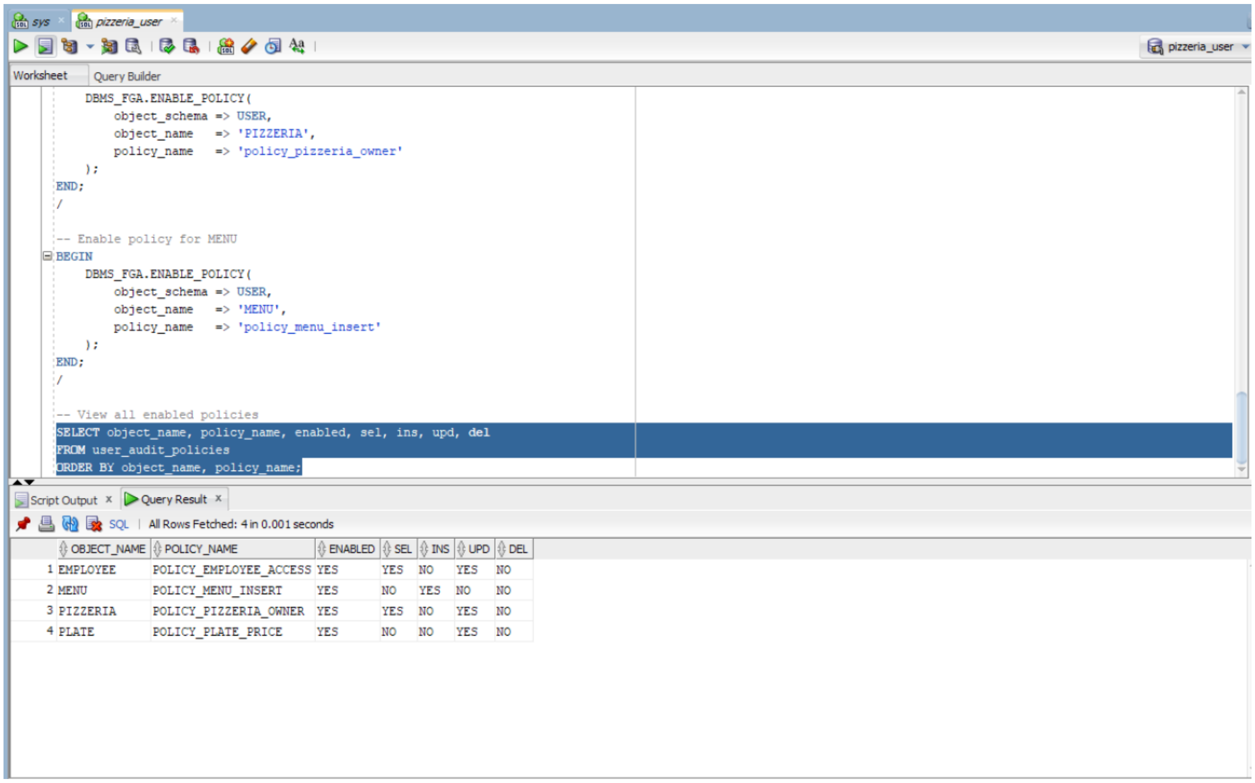


```sql
SET price = 11.50
WHERE plate_id = 2;

COMMIT;

-- View Trigger Audit Results
-- View all audit records
SELECT id_audit, user_, table_name, operation,
       TO_CHAR(time_, 'YYYY-MM-DD HH24:MI:SS') as audit_time,
       SUBSTR(old_value, 1, 50) as old_val,
       SUBSTR(new_value, 1, 50) as new_val
FROM audit_historic
ORDER BY time_ DESC;

-- Count operations by table
SELECT table_name, operation, COUNT(*) as count
FROM audit_historic
GROUP BY table_name, operation
ORDER BY table_name, count DESC;

-- View operations by specific user
SELECT user_, table_name, COUNT(*) as operations
```

All Rows Fetched: 3 in 0.001 seconds

	ID_AUDIT	USER_	TABLE_NAME	OPERATION	AUDIT_TIME	OLD_VAL	NEW_VAL
1	21	PIZZERIA_USER	PLATE	UPDATE PRICE	2026-01-08 22:54:09	Plate: Quattro Formaggi, Old Price: 11.5	Plate: Quattro Formaggi, New Price: 11.5
2	2	PIZZERIA_USER	PLATE	UPDATE PRICE	2026-01-08 22:50:14	Plate: Quattro Formaggi, Old Price: 11.5	Plate: Quattro Formaggi, New Price: 11.5
3	1	PIZZERIA_USER	PLATE	UPDATE PRICE	2026-01-08 22:50:10	Plate: Quattro Formaggi, Old Price: 11	Plate: Quattro Formaggi, New Price: 11.5

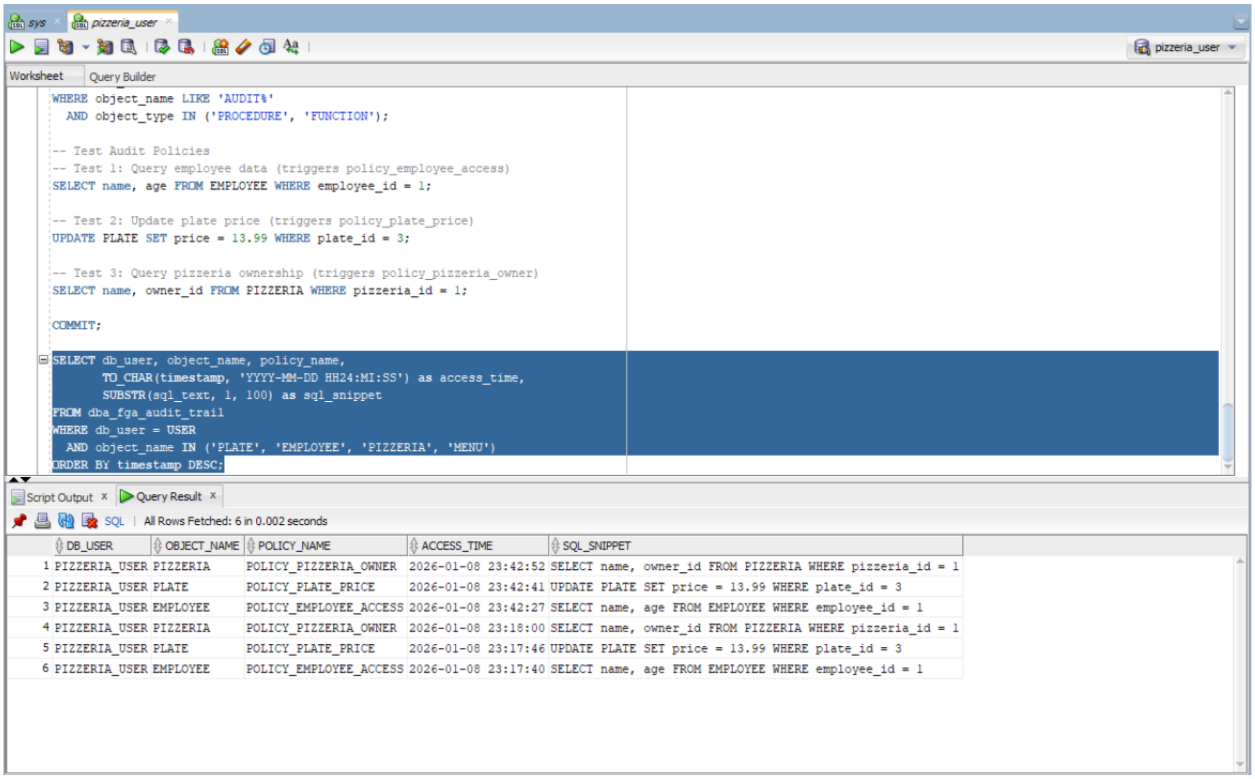
c. Audit Policies



```
    DBMS_FGA.ENABLE_POLICY(
        object_schema => USER,
        object_name   => 'PIZZERIA',
        policy_name   => 'policy_pizzeria_owner'
    );
END;
/

-- Enable policy for MENU
BEGIN
    DBMS_FGA.ENABLE_POLICY(
        object_schema => USER,
        object_name   => 'MENU',
        policy_name   => 'policy_menu_insert'
    );
END;
/

-- View all enabled policies
SELECT object_name, policy_name, enabled, sel, ins, upd, del
FROM user_audit_policies
ORDER BY object_name, policy_name;
```

All Rows Fetched: 4 in 0.001 seconds

	OBJECT_NAME	POLICY_NAME	ENABLED	SEL	INS	UPD	DEL
1	EMPLOYEE	POLICY_EMPLOYEE_ACCESS	YES	YES	NO	YES	NO
2	MENU	POLICY_MENU_INSERT	YES	NO	YES	NO	NO
3	PIZZERIA	POLICY_PIZZERIA_OWNER	YES	YES	NO	YES	NO
4	PLATE	POLICY_PLATE_PRICE	YES	NO	NO	YES	NO



```
WHERE object_name LIKE 'AUDIT%'
  AND object_type IN ('PROCEDURE', 'FUNCTION');

-- Test Audit Policies
-- Test 1: Query employee data (triggers policy_employee_access)
SELECT name, age FROM EMPLOYEE WHERE employee_id = 1;

-- Test 2: Update plate price (triggers policy_plate_price)
UPDATE PLATE SET price = 13.99 WHERE plate_id = 3;

-- Test 3: Query pizzeria ownership (triggers policy_pizzeria_owner)
SELECT name, owner_id FROM PIZZERIA WHERE pizzeria_id = 1;

COMMIT;

SELECT db_user, object_name, policy_name,
       TO_CHAR(timestamp, 'YYYY-MM-DD HH24:MI:SS') as access_time,
       SUBSTR(sql_text, 1, 100) as sql_snippet
FROM dba_fga_audit_trail
WHERE db_user = USER
  AND object_name IN ('PLATE', 'EMPLOYEE', 'PIZZERIA', 'MENU')
ORDER BY timestamp DESC;
```

All Rows Fetched: 6 in 0.002 seconds

	DB_USER	OBJECT_NAME	POLICY_NAME	ACCESS_TIME	SQL_SNIPPET
1	PIZZERIA_USER	PIZZERIA	POLICY_PIZZERIA_OWNER	2026-01-08 23:42:52	SELECT name, owner_id FROM PIZZERIA WHERE pizzeria_id = 1
2	PIZZERIA_USER	PLATE	POLICY_PLATE_PRICE	2026-01-08 23:42:41	UPDATE PLATE SET price = 13.99 WHERE plate_id = 3
3	PIZZERIA_USER	EMPLOYEE	POLICY_EMPLOYEE_ACCESS	2026-01-08 23:42:27	SELECT name, age FROM EMPLOYEE WHERE employee_id = 1
4	PIZZERIA_USER	PIZZERIA	POLICY_PIZZERIA_OWNER	2026-01-08 23:18:00	SELECT name, owner_id FROM PIZZERIA WHERE pizzeria_id = 1
5	PIZZERIA_USER	PLATE	POLICY_PLATE_PRICE	2026-01-08 23:17:46	UPDATE PLATE SET price = 13.99 WHERE plate_id = 3
6	PIZZERIA_USER	EMPLOYEE	POLICY_EMPLOYEE_ACCESS	2026-01-08 23:17:40	SELECT name, age FROM EMPLOYEE WHERE employee_id = 1

4. Management of Database Users and Computational Resources
   Onutu_Radu-Constantin_510-identity_resource_mgmt.txt
   a. Designing the identity management configuration in the database (process-user, entity-process, entity-user matrices)

      Users of the Pizzeria Chain Database:

      - Chain Administrator (1 user) - Manages entire pizzeria chain
      - Pizzeria Managers (8 users) - One manager per pizzeria location
      - Kitchen Staff (10 users) - Chefs and cooks across locations
      - Service Staff (8 users) - Waiters and hosts
      - Inventory Clerks (3 users) - Manage supplies and ingredients
      - Customers (represents general public access)

      Application processes:

      - P1: Manage pizzeria locations (create, update, delete pizzerias)
      - P2: Manage menu items (add, update, delete plates)
      - P3: View all menus across chain
      - P4: Manage employees (hire, update, terminate)
      - P5: Assign teams to departments
      - P6: Update plate prices
      - P7: View sales reports
      - P8: Manage inventory
      - P9: Customer menu browsing
      - P10: Place orders
      - P11: Process payments
      - P12: View pizzeria performance metrics

      Process-user matrix:

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12
Admin	X	X	X	X	X	X	X	X				
Manager		X	X	X	X	X		X	X			X
Kitchen		X	X					X				
Service		X	X						X	X	X	
Inventory							X	X				
Customer						X		X	X	X		

Entity-process matrix:

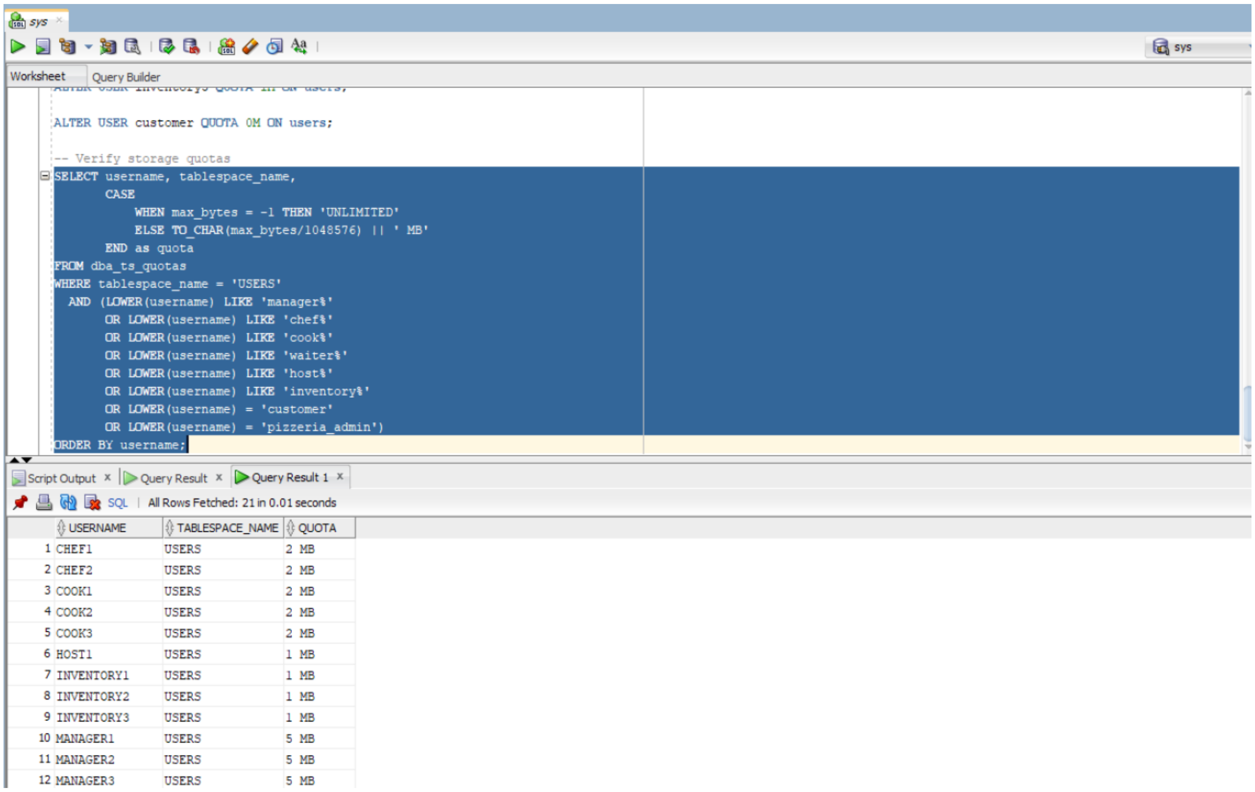
	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12
PIZZERIA	I,U,D	S	S	S	S	S	S					S
OWNER	S	S	S	S	S	S	S					S
MENU		I,U,D	S			S			S	S		
PLATE		I,U,D	S		I,U	S			S	S		
EMPLOYEE			S	I,U,D	S							
TEAM					S,U							
DEPARTMENT					I,U,S							
DEPARTMENT_TEAM_MAPPING					I,U,D							
CITY	S	S	S	S	S	S	S					S
COUNTRY	S	S	S	S	S	S	S					S

Legend: I=Insert, U=Update, D=Delete, S=Select

Entity-user matrix:

	Admin	Manager	Kitchen	Service	Invetory	Customer
PIZZERIA	I,U,D,S	S	S	S	S	S
OWNER	I,U,D,S	S	S	S	S	
MENU	I,U,D,S	I,U,D,S	I,U,S	S		S
PLATE	I,U,D,S	I,U,D,S	I,U,S	S		S
EMPLOYEE	I,U,D,S	I,U,D,S	S	I,U,S	I,U,S	
TEAM	I,U,D,S	I,U,S	S	S	I,U,S	
DEPARTMENT	I,U,D,S	I,U,D,S	S	S	S	
DEPARTMENT_TEAM_MAPPING	I,U,D,S	I,U,S	S	S	I,U,S	
CITY	S	S	S	S	S	S
COUNTRY	S	S	S	S	S	S

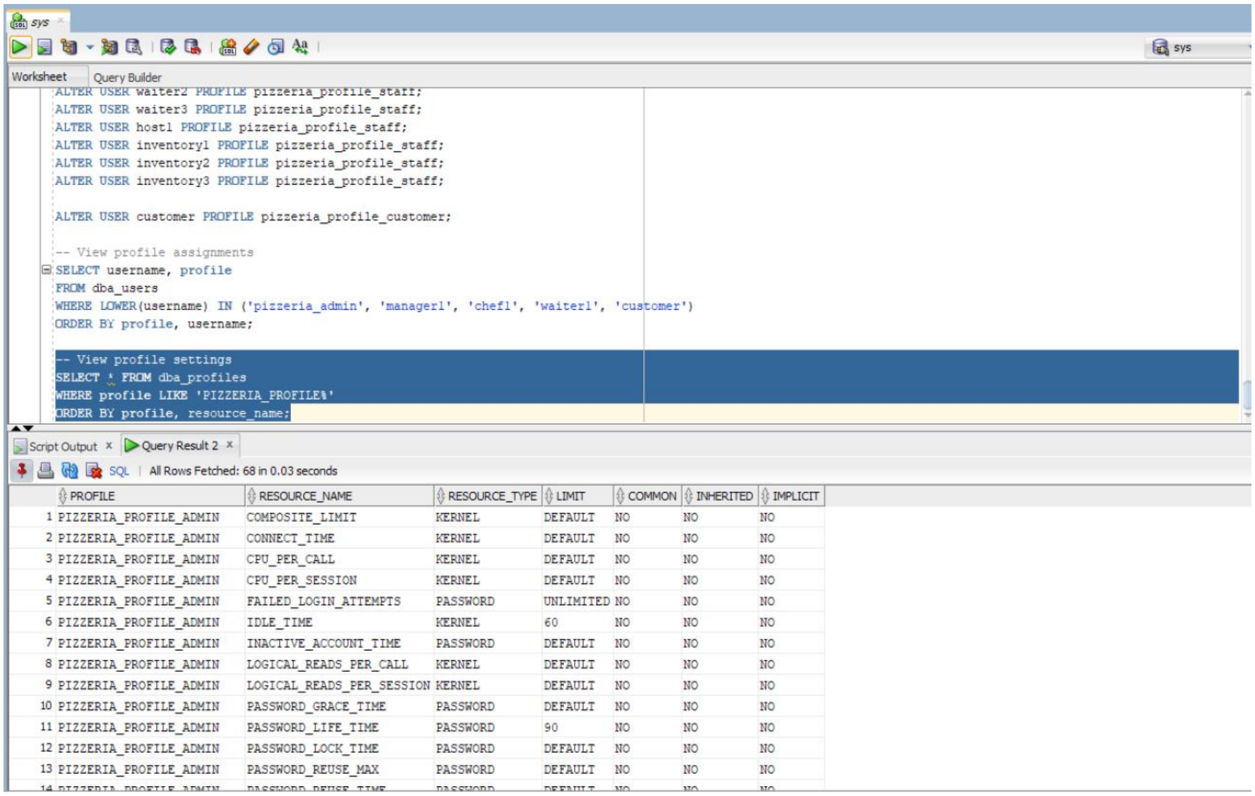
b. Implementing the identity management configuration in the database



```sql
ALTER USER inventory3 QUOTA 1M ON users;

ALTER USER customer QUOTA 0M ON users;

-- Verify storage quotas
SELECT username, tablespace_name,
       CASE
           WHEN max_bytes = -1 THEN 'UNLIMITED'
           ELSE TO_CHAR(max_bytes/1048576) || ' MB'
       END as quota
FROM dba_ts_quotas
WHERE tablespace_name = 'USERS'
  AND (LOWER(username) LIKE 'manager%'
       OR LOWER(username) LIKE 'chef%'
       OR LOWER(username) LIKE 'cook%'
       OR LOWER(username) LIKE 'waiter%'
       OR LOWER(username) LIKE 'host%'
       OR LOWER(username) LIKE 'inventory%'
       OR LOWER(username) = 'customer'
       OR LOWER(username) = 'pizzeria_admin')
ORDER BY username;
```

All Rows Fetched: 21 in 0.01 seconds

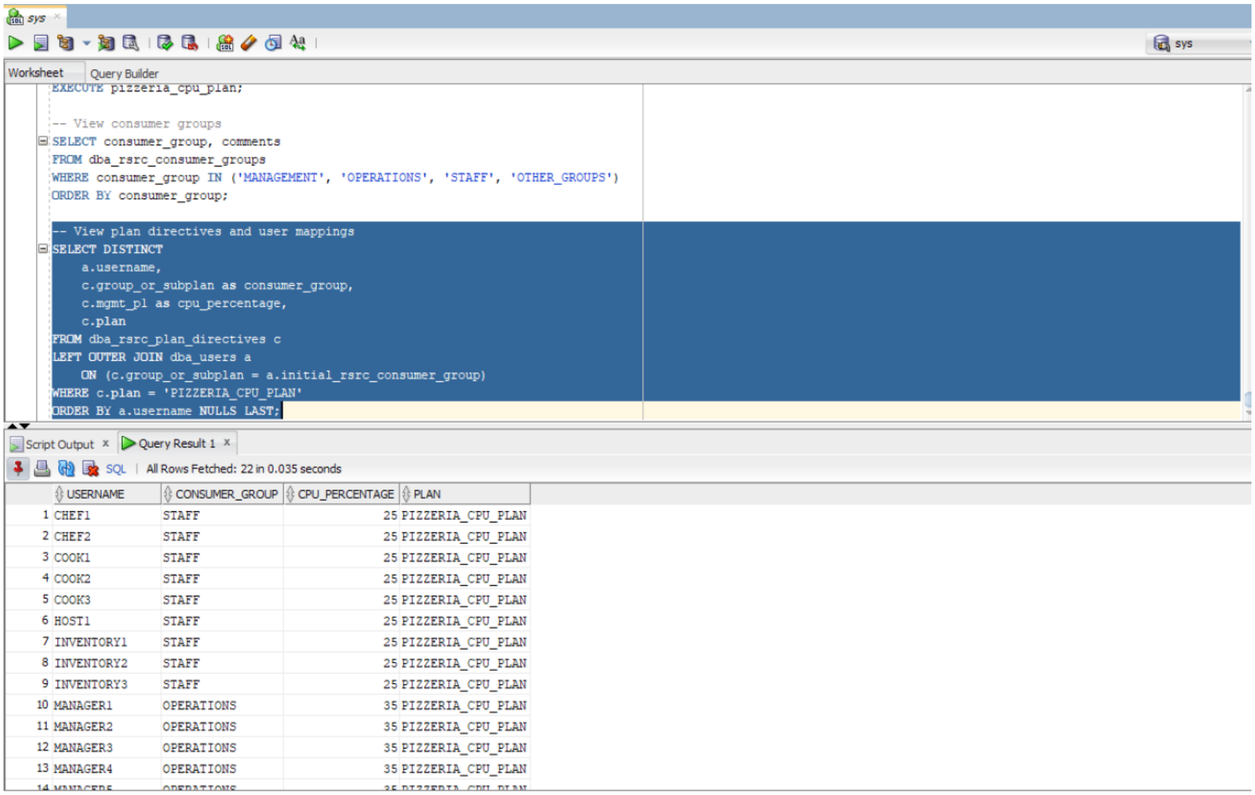
	USERNAME	TABLESPACE_NAME	QUOTA
1	CHEF1	USERS	2 MB
2	CHEF2	USERS	2 MB
3	COOK1	USERS	2 MB
4	COOK2	USERS	2 MB
5	COOK3	USERS	2 MB
6	HOST1	USERS	1 MB
7	INVENTORY1	USERS	1 MB
8	INVENTORY2	USERS	1 MB
9	INVENTORY3	USERS	1 MB
10	MANAGER1	USERS	5 MB
11	MANAGER2	USERS	5 MB
12	MANAGER3	USERS	5 MB



```sql
ALTER USER waiter2 PROFILE pizzeria_profile_staff;
ALTER USER waiter3 PROFILE pizzeria_profile_staff;
ALTER USER host1 PROFILE pizzeria_profile_staff;
ALTER USER inventory1 PROFILE pizzeria_profile_staff;
ALTER USER inventory2 PROFILE pizzeria_profile_staff;
ALTER USER inventory3 PROFILE pizzeria_profile_staff;

ALTER USER customer PROFILE pizzeria_profile_customer;

-- View profile assignments
SELECT username, profile
FROM dba_users
WHERE LOWER(username) IN ('pizzeria_admin', 'manager1', 'chef1', 'waiter1', 'customer')
ORDER BY profile, username;

-- View profile settings
SELECT * FROM dba_profiles
WHERE profile LIKE 'PIZZERIA_PROFILE%'
ORDER BY profile, resource_name;
```

All Rows Fetched: 68 in 0.03 seconds

	PROFILE	RESOURCE_NAME	RESOURCE_TYPE	LIMIT	COMMON	INHERITED	IMPLICIT
1	PIZZERIA_PROFILE_ADMIN	COMPOSITE_LIMIT	KERNEL	DEFAULT	NO	NO	NO
2	PIZZERIA_PROFILE_ADMIN	CONNECT_TIME	KERNEL	DEFAULT	NO	NO	NO
3	PIZZERIA_PROFILE_ADMIN	CPU_PER_CALL	KERNEL	DEFAULT	NO	NO	NO
4	PIZZERIA_PROFILE_ADMIN	CPU_PER_SESSION	KERNEL	DEFAULT	NO	NO	NO
5	PIZZERIA_PROFILE_ADMIN	FAILED_LOGIN_ATTEMPTS	PASSWORD	UNLIMITED	NO	NO	NO
6	PIZZERIA_PROFILE_ADMIN	IDLE_TIME	KERNEL	60	NO	NO	NO
7	PIZZERIA_PROFILE_ADMIN	INACTIVE_ACCOUNT_TIME	PASSWORD	DEFAULT	NO	NO	NO
8	PIZZERIA_PROFILE_ADMIN	LOGICAL_READS_PER_CALL	KERNEL	DEFAULT	NO	NO	NO
9	PIZZERIA_PROFILE_ADMIN	LOGICAL_READS_PER_SESSION	KERNEL	DEFAULT	NO	NO	NO
10	PIZZERIA_PROFILE_ADMIN	PASSWORD_GRACE_TIME	PASSWORD	DEFAULT	NO	NO	NO
11	PIZZERIA_PROFILE_ADMIN	PASSWORD_LIFE_TIME	PASSWORD	90	NO	NO	NO
12	PIZZERIA_PROFILE_ADMIN	PASSWORD_LOCK_TIME	PASSWORD	DEFAULT	NO	NO	NO
13	PIZZERIA_PROFILE_ADMIN	PASSWORD_REUSE_MAX	PASSWORD	DEFAULT	NO	NO	NO
14	PIZZERIA_PROFILE_ADMIN	PASSWORD_REUSE_TIME	PASSWORD	DEFAULT	NO	NO	NO



```sql
EXECUTE pizzeria_cpu_plan;

-- View consumer groups
SELECT consumer_group, comments
FROM dba_rsrc_consumer_groups
WHERE consumer_group IN ('MANAGEMENT', 'OPERATIONS', 'STAFF', 'OTHER_GROUPS')
ORDER BY consumer_group;

-- View plan directives and user mappings
SELECT DISTINCT
    a.username,
    c.group_or_subplan as consumer_group,
    c.mgmt_p1 as cpu_percentage,
    c.plan
FROM dba_rsrc_plan_directives c
LEFT OUTER JOIN dba_users a
    ON (c.group_or_subplan = a.initial_rsrc_consumer_group)
WHERE c.plan = 'PIZZERIA_CPU_PLAN'
ORDER BY a.username NULLS LAST;
```

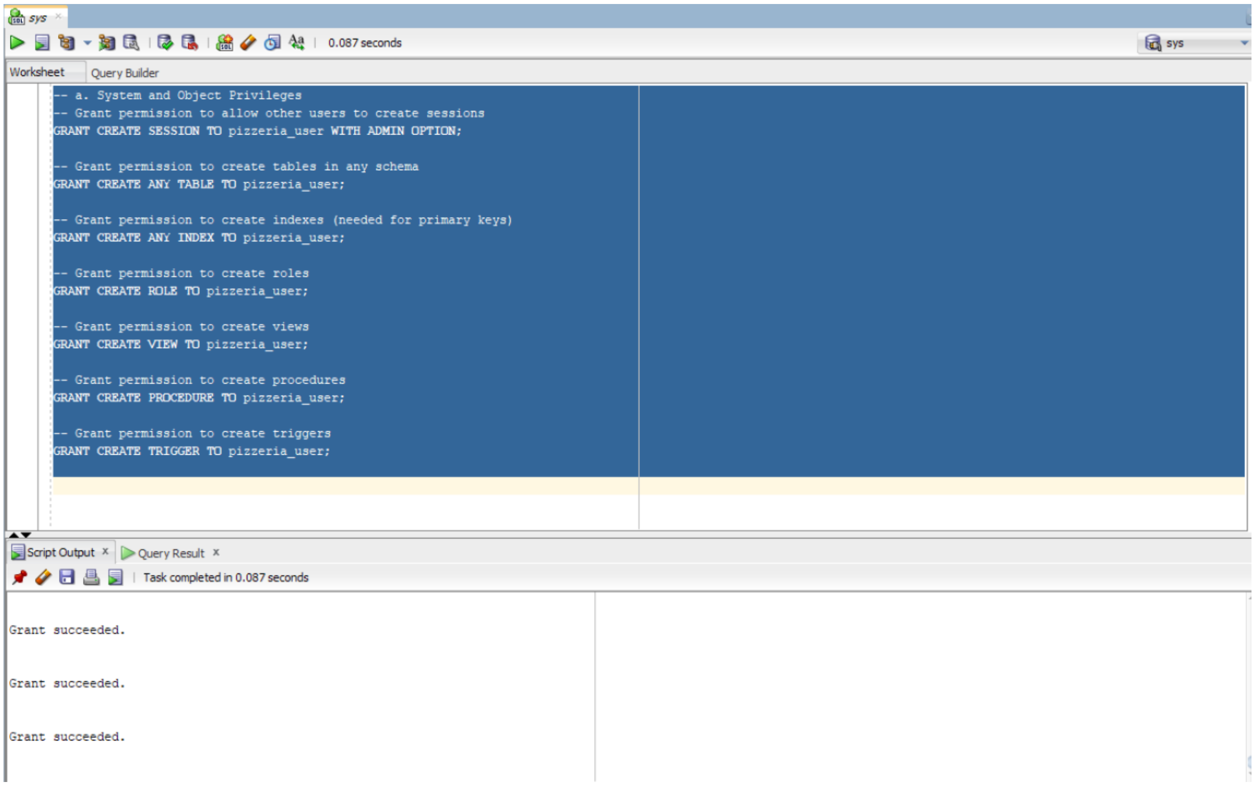
All Rows Fetched: 22 in 0.035 seconds

	USERNAME	CONSUMER_GROUP	CPU_PERCENTAGE	PLAN
1	CHEF1	STAFF	25	PIZZERIA_CPU_PLAN
2	CHEF2	STAFF	25	PIZZERIA_CPU_PLAN
3	COOK1	STAFF	25	PIZZERIA_CPU_PLAN
4	COOK2	STAFF	25	PIZZERIA_CPU_PLAN
5	COOK3	STAFF	25	PIZZERIA_CPU_PLAN
6	HOST1	STAFF	25	PIZZERIA_CPU_PLAN
7	INVENTORY1	STAFF	25	PIZZERIA_CPU_PLAN
8	INVENTORY2	STAFF	25	PIZZERIA_CPU_PLAN
9	INVENTORY3	STAFF	25	PIZZERIA_CPU_PLAN
10	MANAGER1	OPERATIONS	35	PIZZERIA_CPU_PLAN
11	MANAGER2	OPERATIONS	35	PIZZERIA_CPU_PLAN
12	MANAGER3	OPERATIONS	35	PIZZERIA_CPU_PLAN
13	MANAGER4	OPERATIONS	35	PIZZERIA_CPU_PLAN
14	MANAGER5	OPERATIONS	35	PIZZERIA_CPU_PLAN

5. Privileges and Roles

   Onutu_Radu-Constantin_510-privs_roles.txt

   a. System and Object Privileges

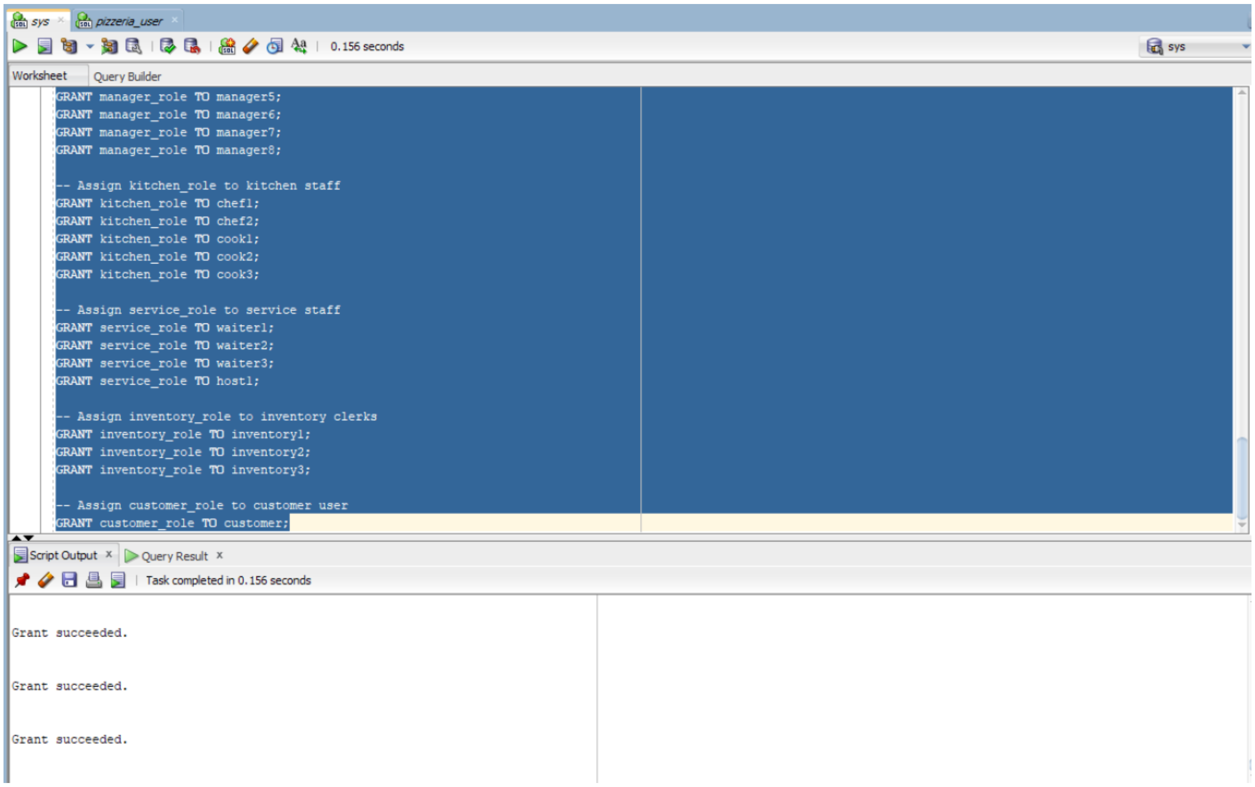


```sql
-- a. System and Object Privileges
-- Grant permission to allow other users to create sessions
GRANT CREATE SESSION TO pizzeria_user WITH ADMIN OPTION;

-- Grant permission to create tables in any schema
GRANT CREATE ANY TABLE TO pizzeria_user;

-- Grant permission to create indexes (needed for primary keys)
GRANT CREATE ANY INDEX TO pizzeria_user;

-- Grant permission to create roles
GRANT CREATE ROLE TO pizzeria_user;

-- Grant permission to create views
GRANT CREATE VIEW TO pizzeria_user;

-- Grant permission to create procedures
GRANT CREATE PROCEDURE TO pizzeria_user;

-- Grant permission to create triggers
GRANT CREATE TRIGGER TO pizzeria_user;
```

Task completed in 0.087 seconds

```
Grant succeeded.


Grant succeeded.


Grant succeeded.
```

b. Privileges hierarchies

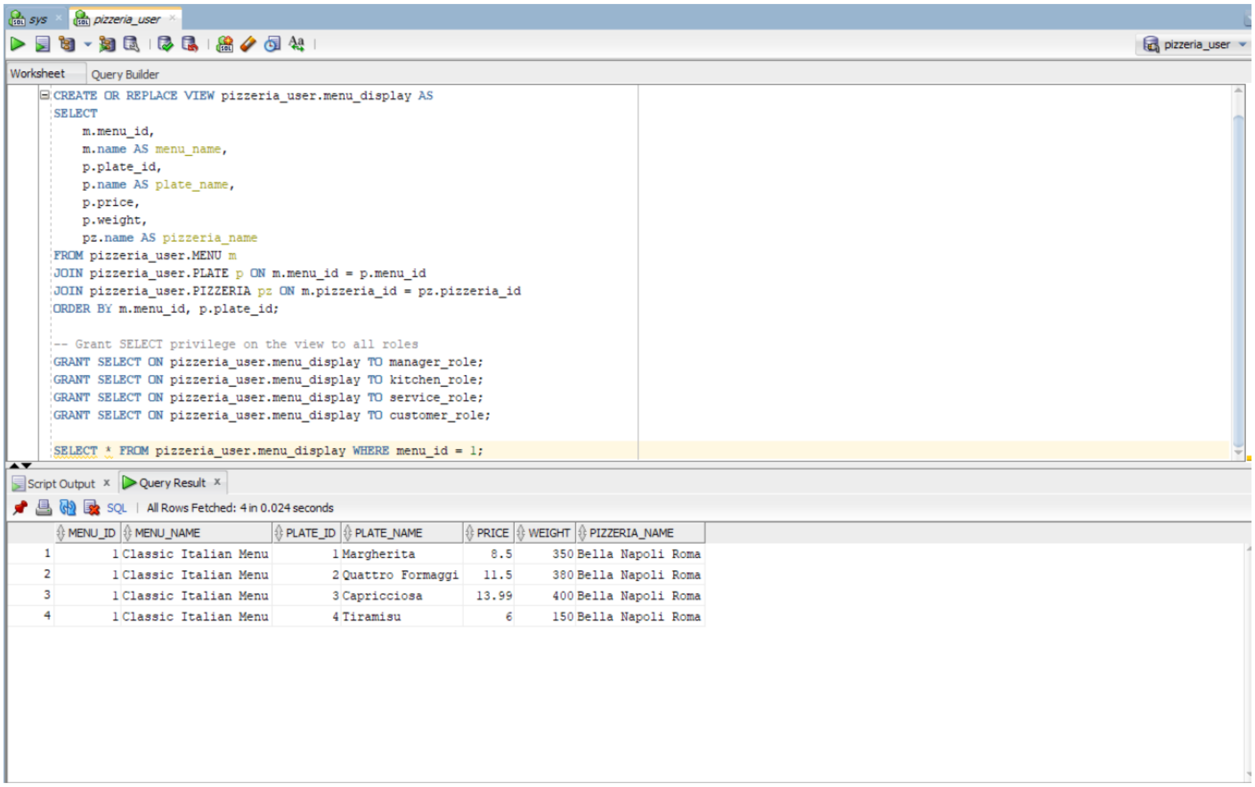


```
GRANT manager_role TO manager5;
GRANT manager_role TO manager6;
GRANT manager_role TO manager7;
GRANT manager_role TO manager8;

-- Assign kitchen_role to kitchen staff
GRANT kitchen_role TO chef1;
GRANT kitchen_role TO chef2;
GRANT kitchen_role TO cook1;
GRANT kitchen_role TO cook2;
GRANT kitchen_role TO cook3;

-- Assign service_role to service staff
GRANT service_role TO waiter1;
GRANT service_role TO waiter2;
GRANT service_role TO waiter3;
GRANT service_role TO host1;

-- Assign inventory_role to inventory clerks
GRANT inventory_role TO inventory1;
GRANT inventory_role TO inventory2;
GRANT inventory_role TO inventory3;

-- Assign customer_role to customer user
GRANT customer_role TO customer;
```

Task completed in 0.156 seconds

```
Grant succeeded.


Grant succeeded.


Grant succeeded.
```

c. Privileges on depdendent objects



```
CREATE OR REPLACE VIEW pizzeria_user.menu_display AS
SELECT
    m.menu_id,
    m.name AS menu_name,
    p.plate_id,
    p.name AS plate_name,
    p.price,
    p.weight,
    pz.name AS pizzeria_name
FROM pizzeria_user.MENU m
JOIN pizzeria_user.PLATE p ON m.menu_id = p.menu_id
JOIN pizzeria_user.PIZZERIA pz ON m.pizzeria_id = pz.pizzeria_id
ORDER BY m.menu_id, p.plate_id;

-- Grant SELECT privilege on the view to all roles
GRANT SELECT ON pizzeria_user.menu_display TO manager_role;
GRANT SELECT ON pizzeria_user.menu_display TO kitchen_role;
GRANT SELECT ON pizzeria_user.menu_display TO service_role;
GRANT SELECT ON pizzeria_user.menu_display TO customer_role;

SELECT * FROM pizzeria_user.menu_display WHERE menu_id = 1;
```

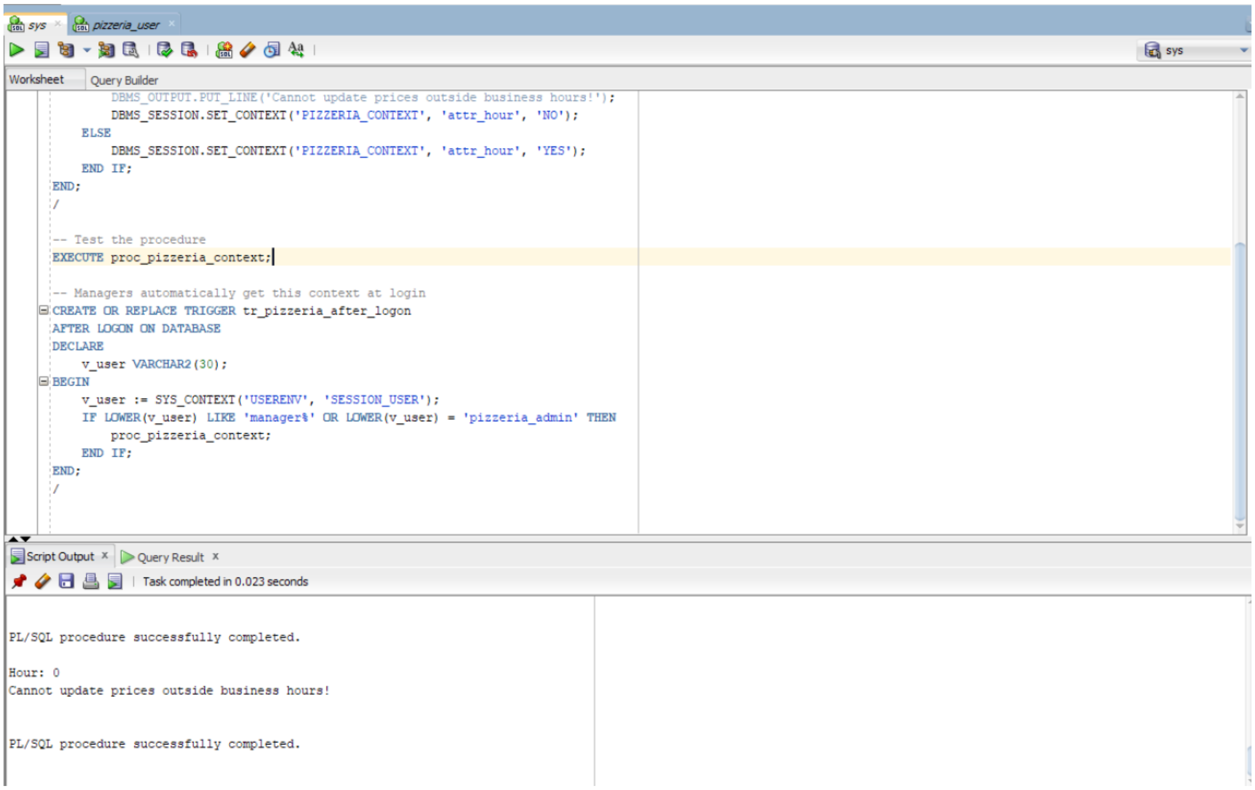
All Rows Fetched: 4 in 0.024 seconds

	MENU_ID	MENU_NAME	PLATE_ID	PLATE_NAME	PRICE	WEIGHT	PIZZERIA_NAME
1	1	Classic Italian Menu	1	Margherita	8.5	350	Bella Napoli Roma
2	1	Classic Italian Menu	2	Quattro Formaggi	11.5	380	Bella Napoli Roma
3	1	Classic Italian Menu	3	Capricciosa	13.99	400	Bella Napoli Roma
4	1	Classic Italian Menu	4	Tiramisu	6	150	Bella Napoli Roma

6. Database Applications and Data Security

Onutu_Radu-Constantin_510-application_security.txt

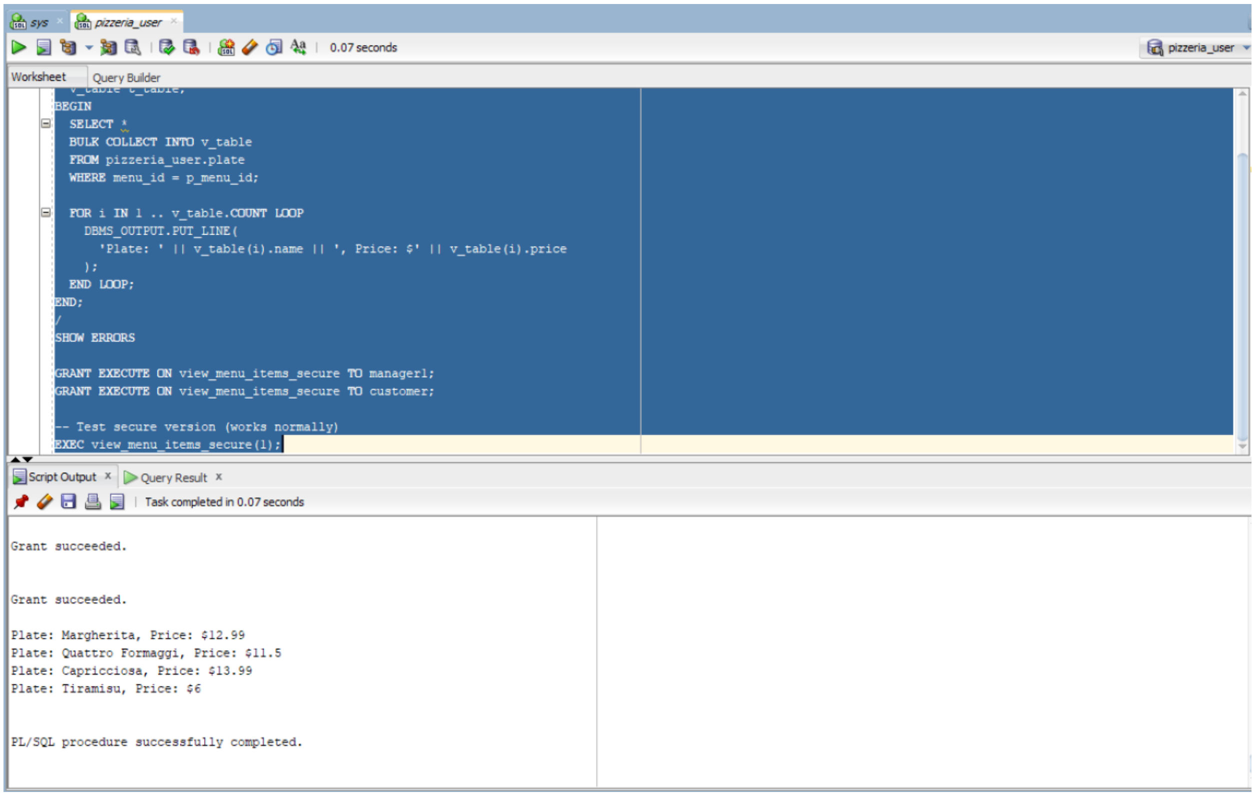
a. Application Context



```
            DBMS_OUTPUT.PUT_LINE('Cannot update prices outside business hours!');
            DBMS_SESSION.SET_CONTEXT('PIZZERIA_CONTEXT', 'attr_hour', 'NO');
        ELSE
            DBMS_SESSION.SET_CONTEXT('PIZZERIA_CONTEXT', 'attr_hour', 'YES');
        END IF;
END;
/

-- Test the procedure
EXECUTE proc_pizzeria_context;

-- Managers automatically get this context at login
CREATE OR REPLACE TRIGGER tr_pizzeria_after_logon
AFTER LOGON ON DATABASE
DECLARE
    v_user VARCHAR2(30);
BEGIN
    v_user := SYS_CONTEXT('USERENV', 'SESSION_USER');
    IF LOWER(v_user) LIKE 'manager%' OR LOWER(v_user) = 'pizzeria_admin' THEN
        proc_pizzeria_context;
    END IF;
END;
/
```

```
PL/SQL procedure successfully completed.

Hour: 0
Cannot update prices outside business hours!


PL/SQL procedure successfully completed.
```

b. SQL Injection

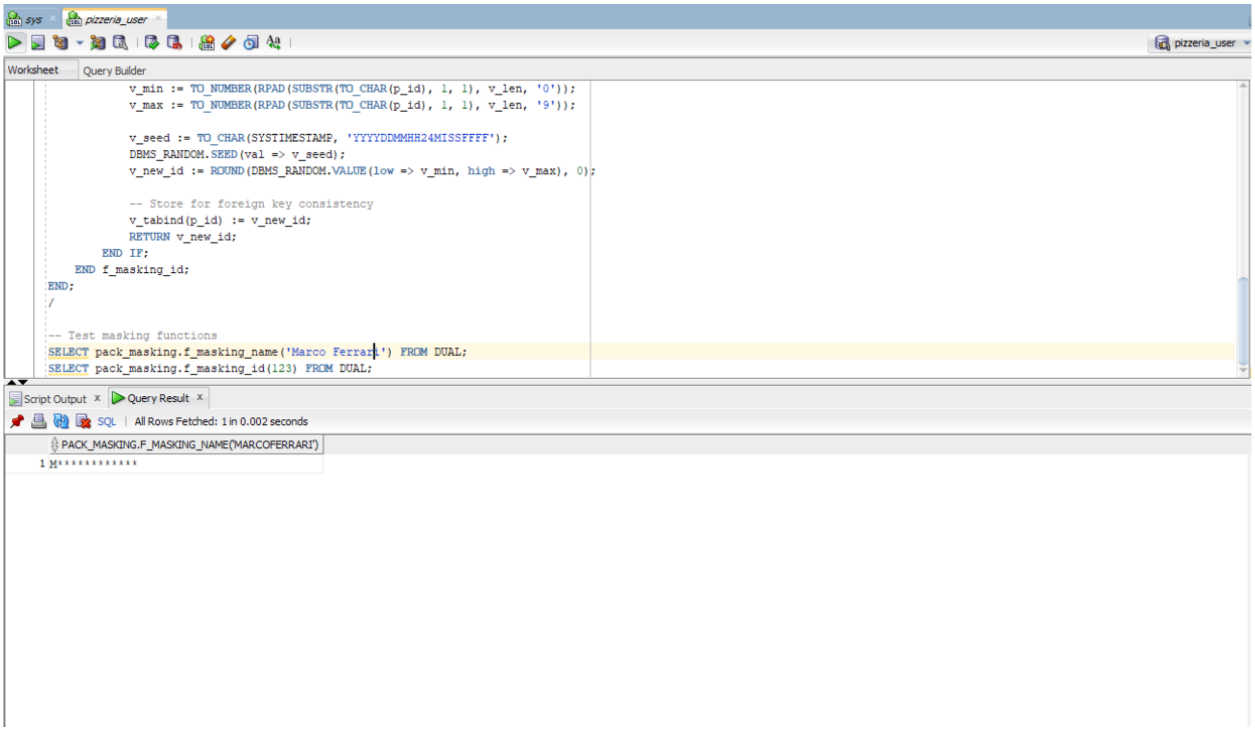


```
  v_table t_table;
BEGIN
  SELECT *
  BULK COLLECT INTO v_table
  FROM pizzeria_user.plate
  WHERE menu_id = p_menu_id;

  FOR i IN 1 .. v_table.COUNT LOOP
    DBMS_OUTPUT.PUT_LINE(
      'Plate: ' || v_table(i).name || ', Price: $' || v_table(i).price
    );
  END LOOP;
END;
/
SHOW ERRORS

GRANT EXECUTE ON view_menu_items_secure TO manager1;
GRANT EXECUTE ON view_menu_items_secure TO customer;

-- Test secure version (works normally)
EXEC view_menu_items_secure(1);
```

```
Grant succeeded.


Grant succeeded.

Plate: Margherita, Price: $12.99
Plate: Quattro Formaggi, Price: $11.5
Plate: Capricciosa, Price: $13.99
Plate: Tiramisu, Price: $6


PL/SQL procedure successfully completed.
```

7. Data Masking

Onutu_Radu-Constantin_510-data_masking.txt



```
            v_min := TO_NUMBER(RPAD(SUBSTR(TO_CHAR(p_id), 1, 1), v_len, '0'));
            v_max := TO_NUMBER(RPAD(SUBSTR(TO_CHAR(p_id), 1, 1), v_len, '9'));

            v_seed := TO_CHAR(SYSTIMESTAMP, 'YYYYDDMMHH24MISSFFFF');
            DBMS_RANDOM.SEED(val => v_seed);
            v_new_id := ROUND(DBMS_RANDOM.VALUE(low => v_min, high => v_max), 0);

            -- Store for foreign key consistency
            v_tabind(p_id) := v_new_id;
            RETURN v_new_id;
        END IF;
    END f_masking_id;
END;
/

-- Test masking functions
SELECT pack_masking.f_masking_name('Marco Ferrari') FROM DUAL;
SELECT pack_masking.f_masking_id(123) FROM DUAL;
```

All Rows Fetched: 1 in 0.002 seconds

```
  PACK_MASKING.F_MASKING_NAME('MARCOFERRARI')
1 M************
```