Cover Page

ST10439309

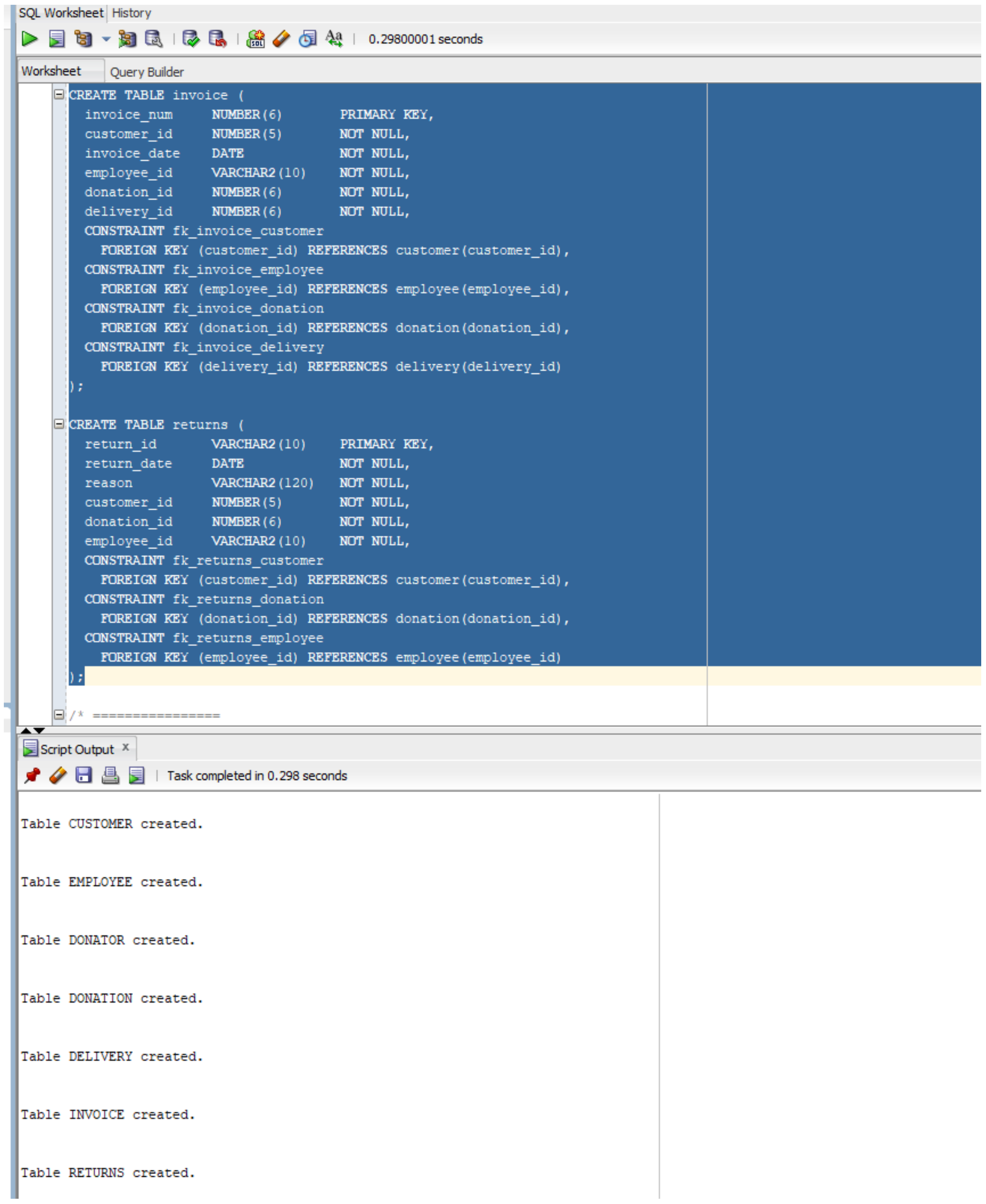
INSY7213

Information Systems 2C

Table of Contents

Q1

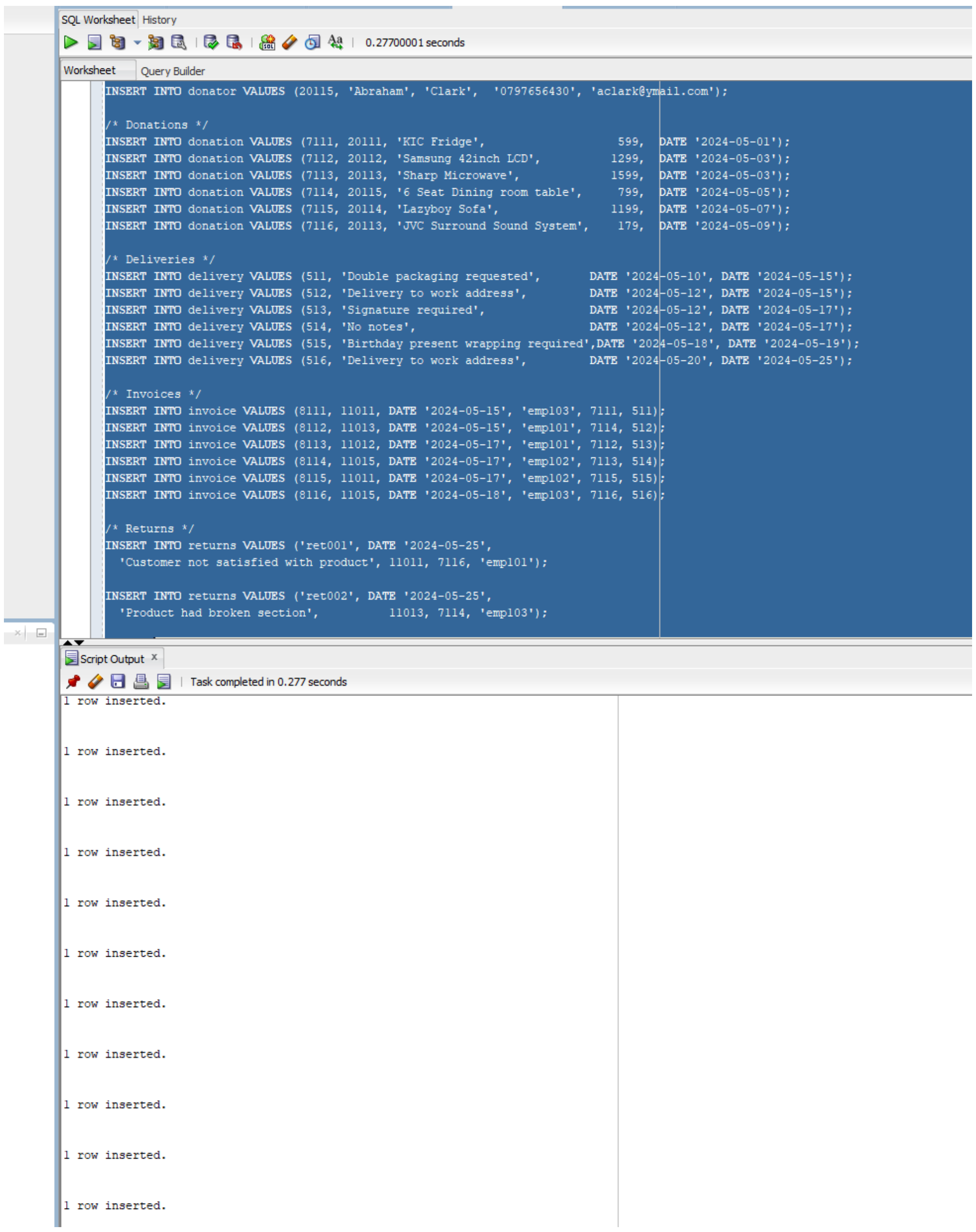
Table creations:



```sql
CREATE TABLE invoice (
  invoice_num     NUMBER(6)       PRIMARY KEY,
  customer_id     NUMBER(5)       NOT NULL,
  invoice_date    DATE            NOT NULL,
  employee_id     VARCHAR2(10)    NOT NULL,
  donation_id     NUMBER(6)       NOT NULL,
  delivery_id     NUMBER(6)       NOT NULL,
  CONSTRAINT fk_invoice_customer
    FOREIGN KEY (customer_id) REFERENCES customer(customer_id),
  CONSTRAINT fk_invoice_employee
    FOREIGN KEY (employee_id) REFERENCES employee(employee_id),
  CONSTRAINT fk_invoice_donation
    FOREIGN KEY (donation_id) REFERENCES donation(donation_id),
  CONSTRAINT fk_invoice_delivery
    FOREIGN KEY (delivery_id) REFERENCES delivery(delivery_id)
);

CREATE TABLE returns (
  return_id       VARCHAR2(10)    PRIMARY KEY,
  return_date     DATE            NOT NULL,
  reason          VARCHAR2(120)   NOT NULL,
  customer_id     NUMBER(5)       NOT NULL,
  donation_id     NUMBER(6)       NOT NULL,
  employee_id     VARCHAR2(10)    NOT NULL,
  CONSTRAINT fk_returns_customer
    FOREIGN KEY (customer_id) REFERENCES customer(customer_id),
  CONSTRAINT fk_returns_donation
    FOREIGN KEY (donation_id) REFERENCES donation(donation_id),
  CONSTRAINT fk_returns_employee
    FOREIGN KEY (employee_id) REFERENCES employee(employee_id)
);

/* ================
```

```
Table CUSTOMER created.


Table EMPLOYEE created.


Table DONATOR created.


Table DONATION created.


Table DELIVERY created.


Table INVOICE created.


Table RETURNS created.
```

Table insertions:

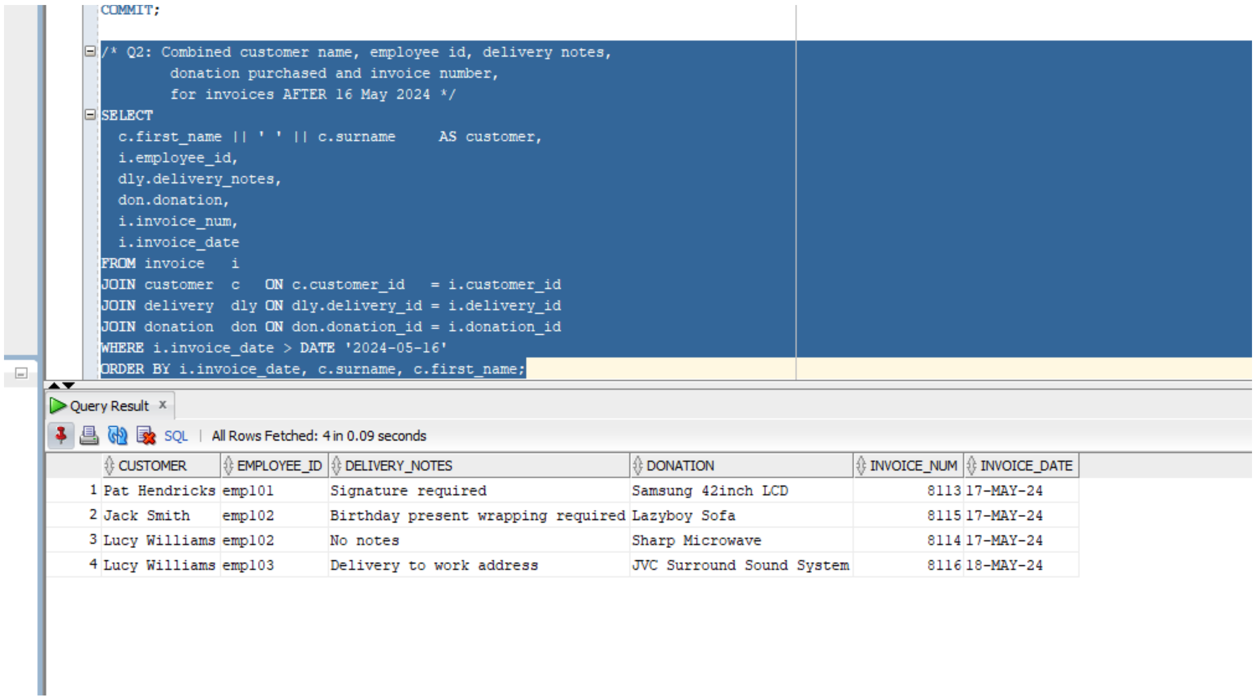

0.27700001 seconds

```sql
INSERT INTO donator VALUES (20115, 'Abraham', 'Clark',  '0797656430', 'aclark@ymail.com');

/* Donations */
INSERT INTO donation VALUES (7111, 20111, 'KIC Fridge',                  599,  DATE '2024-05-01');
INSERT INTO donation VALUES (7112, 20112, 'Samsung 42inch LCD',         1299,  DATE '2024-05-03');
INSERT INTO donation VALUES (7113, 20113, 'Sharp Microwave',            1599,  DATE '2024-05-03');
INSERT INTO donation VALUES (7114, 20115, '6 Seat Dining room table',    799,  DATE '2024-05-05');
INSERT INTO donation VALUES (7115, 20114, 'Lazyboy Sofa',               1199,  DATE '2024-05-07');
INSERT INTO donation VALUES (7116, 20113, 'JVC Surround Sound System',   179,  DATE '2024-05-09');

/* Deliveries */
INSERT INTO delivery VALUES (511, 'Double packaging requested',       DATE '2024-05-10', DATE '2024-05-15');
INSERT INTO delivery VALUES (512, 'Delivery to work address',         DATE '2024-05-12', DATE '2024-05-15');
INSERT INTO delivery VALUES (513, 'Signature required',               DATE '2024-05-12', DATE '2024-05-17');
INSERT INTO delivery VALUES (514, 'No notes',                         DATE '2024-05-12', DATE '2024-05-17');
INSERT INTO delivery VALUES (515, 'Birthday present wrapping required',DATE '2024-05-18', DATE '2024-05-19');
INSERT INTO delivery VALUES (516, 'Delivery to work address',         DATE '2024-05-20', DATE '2024-05-25');

/* Invoices */
INSERT INTO invoice VALUES (8111, 11011, DATE '2024-05-15', 'emp103', 7111, 511);
INSERT INTO invoice VALUES (8112, 11013, DATE '2024-05-15', 'emp101', 7114, 512);
INSERT INTO invoice VALUES (8113, 11012, DATE '2024-05-17', 'emp101', 7112, 513);
INSERT INTO invoice VALUES (8114, 11015, DATE '2024-05-17', 'emp102', 7113, 514);
INSERT INTO invoice VALUES (8115, 11011, DATE '2024-05-17', 'emp102', 7115, 515);
INSERT INTO invoice VALUES (8116, 11015, DATE '2024-05-18', 'emp103', 7116, 516);

/* Returns */
INSERT INTO returns VALUES ('ret001', DATE '2024-05-25',
  'Customer not satisfied with product', 11011, 7116, 'emp101');

INSERT INTO returns VALUES ('ret002', DATE '2024-05-25',
  'Product had broken section',          11013, 7114, 'emp103');
```

Task completed in 0.277 seconds

```
1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.
```

Q2



```
COMMIT;

/* Q2: Combined customer name, employee id, delivery notes,
        donation purchased and invoice number,
        for invoices AFTER 16 May 2024 */
SELECT
  c.first_name || ' ' || c.surname     AS customer,
  i.employee_id,
  dly.delivery_notes,
  don.donation,
  i.invoice_num,
  i.invoice_date
FROM invoice   i
JOIN customer  c   ON c.customer_id   = i.customer_id
JOIN delivery  dly ON dly.delivery_id = i.delivery_id
JOIN donation  don ON don.donation_id = i.donation_id
WHERE i.invoice_date > DATE '2024-05-16'
ORDER BY i.invoice_date, c.surname, c.first_name;
```

Query Result ×

SQL | All Rows Fetched: 4 in 0.09 seconds

	CUSTOMER	EMPLOYEE_ID	DELIVERY_NOTES	DONATION	INVOICE_NUM	INVOICE_DATE
1	Pat Hendricks	emp101	Signature required	Samsung 42inch LCD	8113	17-MAY-24
2	Jack Smith	emp102	Birthday present wrapping required	Lazyboy Sofa	8115	17-MAY-24
3	Lucy Williams	emp102	No notes	Sharp Microwave	8114	17-MAY-24
4	Lucy Williams	emp103	Delivery to work address	JVC Surround Sound System	8116	18-MAY-24

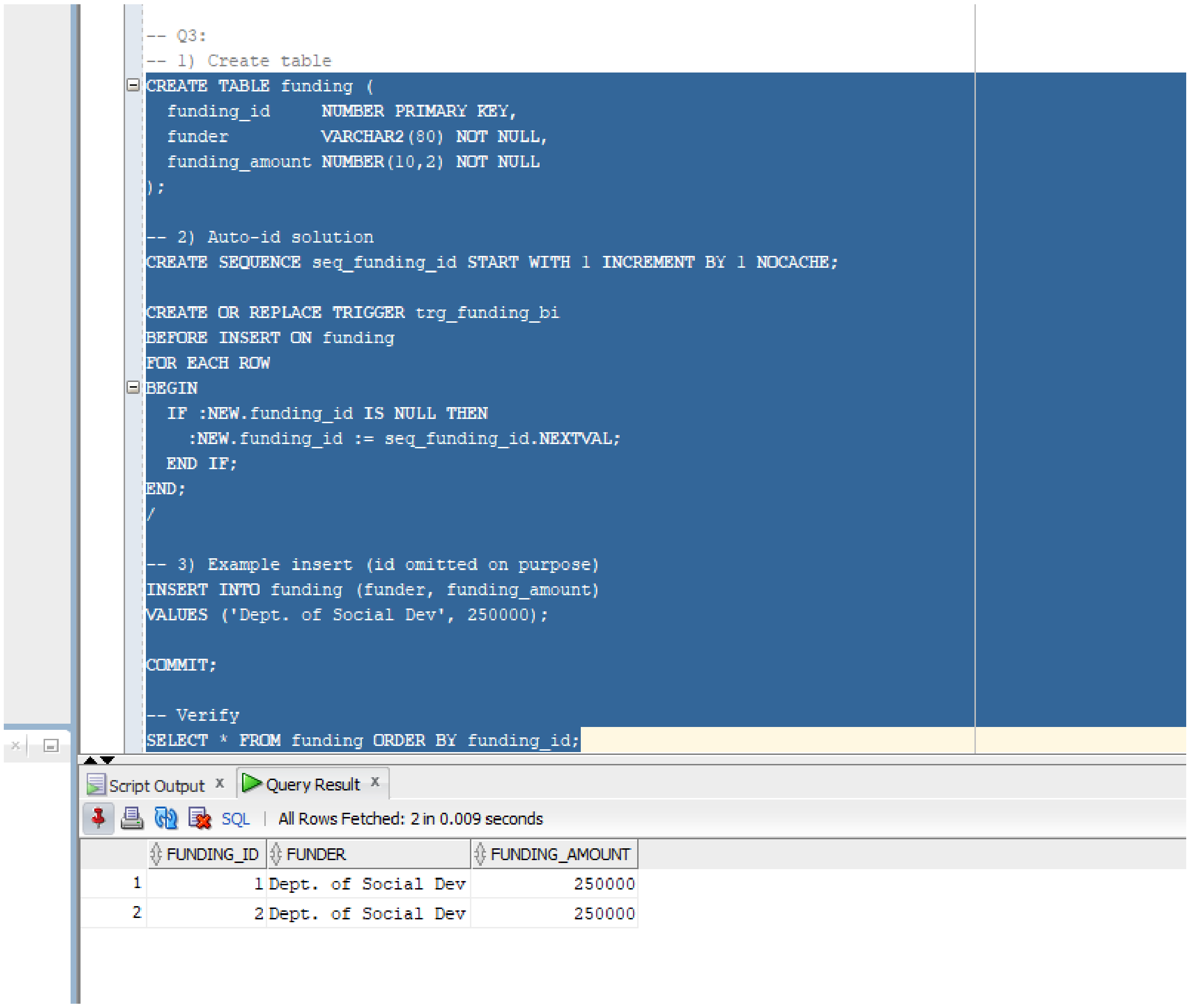
Q3

Code:

```
CREATE TABLE funding (
 funding_id    NUMBER PRIMARY KEY,
 funder       VARCHAR2(80) NOT NULL,
 funding_amount NUMBER(10,2) NOT NULL
);

CREATE SEQUENCE seq_funding_id START WITH 1 INCREMENT BY 1 NOCACHE;

CREATE OR REPLACE TRIGGER trg_funding_bi

BEFORE INSERT ON funding

FOR EACH ROW BEGIN

IF :      NEW.funding_id IS NULL THEN :

          NEW.funding_id := seq_funding_id.NEXTVAL;

          END IF;

END;
INSERT INTO funding (funder, funding_amount)
VALUES ('Dept. of Social Dev', 250000);
COMMIT;
SELECT * FROM funding ORDER BY funding_id;
```

Justification: The SEQUENCE + BEFORE INSERT trigger pattern ensures every new row gets a unique funding_id even if the application doesn’t pass one, and it’s compatible with older Oracle versions.

Output:



```
-- Q3:
-- 1) Create table
CREATE TABLE funding (
  funding_id     NUMBER PRIMARY KEY,
  funder         VARCHAR2(80) NOT NULL,
  funding_amount NUMBER(10,2) NOT NULL
);

-- 2) Auto-id solution
CREATE SEQUENCE seq_funding_id START WITH 1 INCREMENT BY 1 NOCACHE;

CREATE OR REPLACE TRIGGER trg_funding_bi
BEFORE INSERT ON funding
FOR EACH ROW
BEGIN
  IF :NEW.funding_id IS NULL THEN
    :NEW.funding_id := seq_funding_id.NEXTVAL;
  END IF;
END;
/

-- 3) Example insert (id omitted on purpose)
INSERT INTO funding (funder, funding_amount)
VALUES ('Dept. of Social Dev', 250000);

COMMIT;

-- Verify
SELECT * FROM funding ORDER BY funding_id;
```

All Rows Fetched: 2 in 0.009 seconds

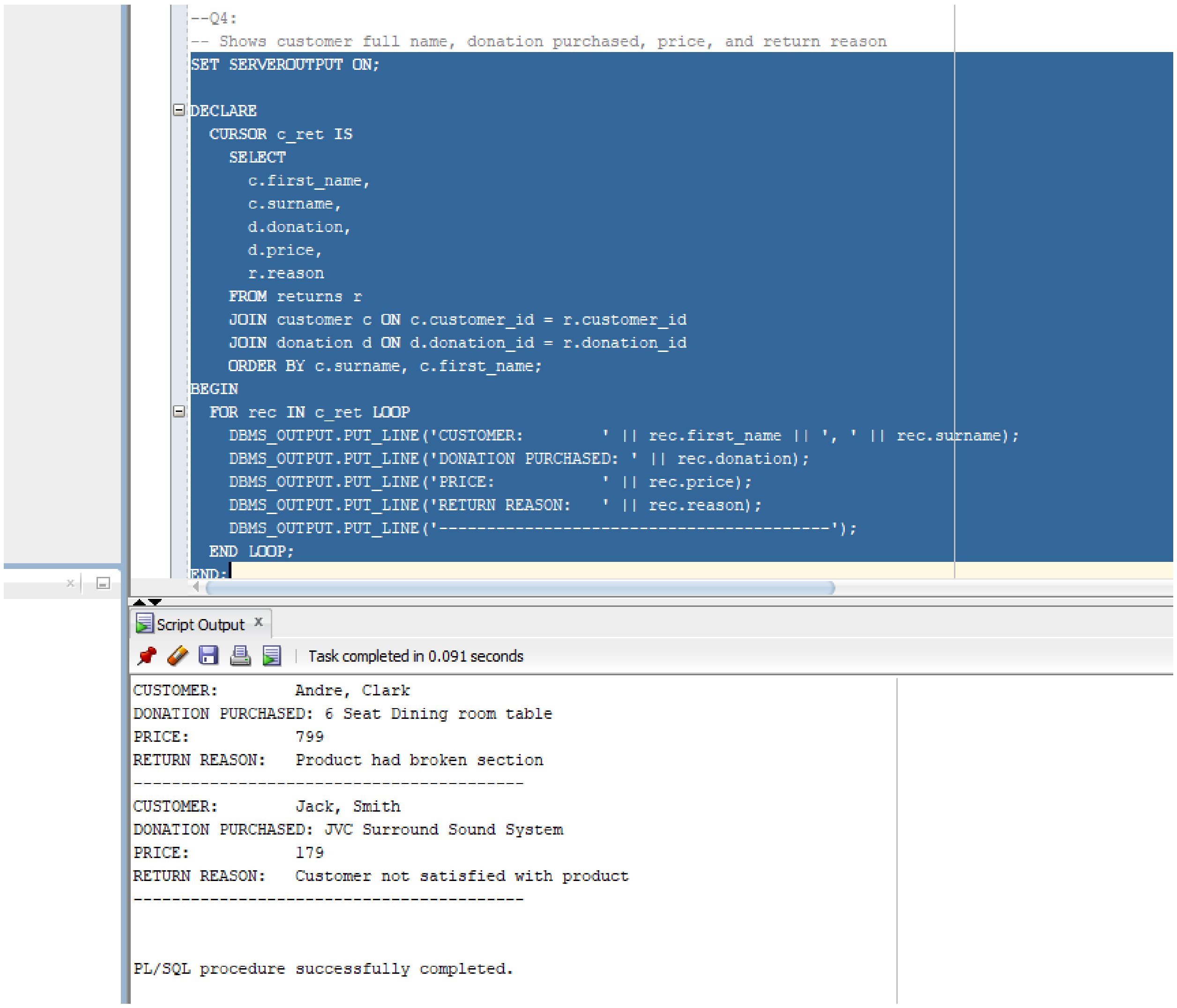
	FUNDING_ID	FUNDER	FUNDING_AMOUNT
1	1	Dept. of Social Dev	250000
2	2	Dept. of Social Dev	250000

Q4

Code:

```
SET SERVEROUTPUT ON;

DECLARE

CURSOR c_ret IS

SELECT

c.first_name,

c.surname,

d.donation,

d.price,

r.reason

FROM returns r

JOIN customer c ON c.customer_id = r.customer_id

JOIN donation d ON d.donation_id = r.donation_id

ORDER BY c.surname, c.first_name;

BEGIN

FOR rec IN c_ret LOOP
        DBMS_OUTPUT.PUT_LINE('CUSTOMER: ' || rec.first_name || ', ' || rec.surname);
        DBMS_OUTPUT.PUT_LINE('DONATION PURCHASED: ' || rec.donation);
        DBMS_OUTPUT.PUT_LINE('PRICE: ' || rec.price);
        DBMS_OUTPUT.PUT_LINE('RETURN REASON: ' || rec.reason);
        DBMS_OUTPUT.PUT_LINE('-----------------------------------------');

END LOOP;

END;
```

Output:



```
--Q4:
-- Shows customer full name, donation purchased, price, and return reason
SET SERVEROUTPUT ON;

DECLARE
  CURSOR c_ret IS
    SELECT
      c.first_name,
      c.surname,
      d.donation,
      d.price,
      r.reason
    FROM returns r
    JOIN customer c ON c.customer_id = r.customer_id
    JOIN donation d ON d.donation_id = r.donation_id
    ORDER BY c.surname, c.first_name;
BEGIN
  FOR rec IN c_ret LOOP
    DBMS_OUTPUT.PUT_LINE('CUSTOMER:         ' || rec.first_name || ', ' || rec.surname);
    DBMS_OUTPUT.PUT_LINE('DONATION PURCHASED: ' || rec.donation);
    DBMS_OUTPUT.PUT_LINE('PRICE:            ' || rec.price);
    DBMS_OUTPUT.PUT_LINE('RETURN REASON:    ' || rec.reason);
    DBMS_OUTPUT.PUT_LINE('-----------------------------------------');
  END LOOP;
END;
```

```
CUSTOMER:         Andre, Clark
DONATION PURCHASED: 6 Seat Dining room table
PRICE:            799
RETURN REASON:    Product had broken section
-----------------------------------------
CUSTOMER:         Jack, Smith
DONATION PURCHASED: JVC Surround Sound System
PRICE:            179
RETURN REASON:    Customer not satisfied with product
-----------------------------------------


PL/SQL procedure successfully completed.
```

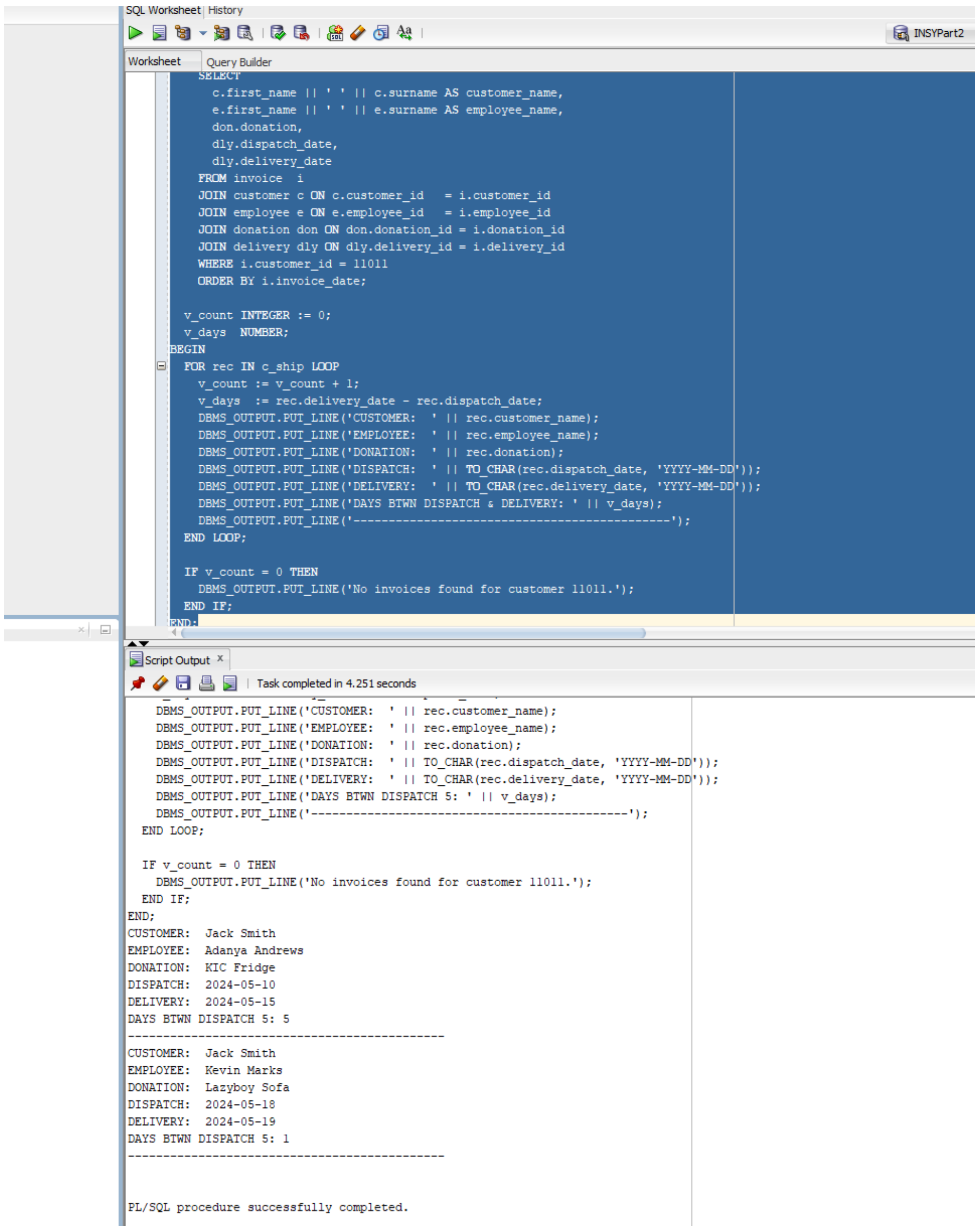
Q5

Code:

```
SET SERVEROUTPUT ON;

DECLARE

CURSOR c_ship IS

SELECT

c.first_name || ' ' || c.surname AS customer_name,

e.first_name || ' ' || e.surname AS employee_name,

don.donation,

dly.dispatch_date,

dly.delivery_date

FROM invoice i

JOIN customer c ON c.customer_id = i.customer_id

JOIN employee e ON e.employee_id = i.employee_id

JOIN donation don ON don.donation_id = i.donation_id

JOIN delivery dly ON dly.delivery_id = i.delivery_id

WHERE i.customer_id = 11011

ORDER BY i.invoice_date;

v_count INTEGER := 0;

v_days NUMBER;

BEGIN

FOR rec IN c_ship LOOP
```

```
v_count := v_count + 1;

v_days := rec.delivery_date - rec.dispatch_date;

DBMS_OUTPUT.PUT_LINE('CUSTOMER: ' || rec.customer_name);
DBMS_OUTPUT.PUT_LINE('EMPLOYEE: ' || rec.employee_name);
DBMS_OUTPUT.PUT_LINE('DONATION: ' || rec.donation);
DBMS_OUTPUT.PUT_LINE('DISPATCH: ' || TO_CHAR(rec.dispatch_date, 'YYYY-MM-DD'));
DBMS_OUTPUT.PUT_LINE('DELIVERY: ' || TO_CHAR(rec.delivery_date, 'YYYY-MM-DD'));
DBMS_OUTPUT.PUT_LINE('DAYS BTWN DISPATCH & DELIVERY: ' || v_days);
DBMS_OUTPUT.PUT_LINE('---------------------------------------------');

END LOOP;

IF v_count = 0 THEN

DBMS_OUTPUT.PUT_LINE('No invoices found for customer 11011.');

END IF;

END;
```

Output:



SQL Worksheet | History

INSYPart2

Worksheet | Query Builder

```
    SELECT
      c.first_name || ' ' || c.surname AS customer_name,
      e.first_name || ' ' || e.surname AS employee_name,
      don.donation,
      dly.dispatch_date,
      dly.delivery_date
    FROM invoice  i
    JOIN customer c ON c.customer_id   = i.customer_id
    JOIN employee e ON e.employee_id   = i.employee_id
    JOIN donation don ON don.donation_id = i.donation_id
    JOIN delivery dly ON dly.delivery_id = i.delivery_id
    WHERE i.customer_id = 11011
    ORDER BY i.invoice_date;

  v_count INTEGER := 0;
  v_days  NUMBER;
BEGIN
  FOR rec IN c_ship LOOP
    v_count := v_count + 1;
    v_days  := rec.delivery_date - rec.dispatch_date;
    DBMS_OUTPUT.PUT_LINE('CUSTOMER:  ' || rec.customer_name);
    DBMS_OUTPUT.PUT_LINE('EMPLOYEE:  ' || rec.employee_name);
    DBMS_OUTPUT.PUT_LINE('DONATION:  ' || rec.donation);
    DBMS_OUTPUT.PUT_LINE('DISPATCH:  ' || TO_CHAR(rec.dispatch_date, 'YYYY-MM-DD'));
    DBMS_OUTPUT.PUT_LINE('DELIVERY:  ' || TO_CHAR(rec.delivery_date, 'YYYY-MM-DD'));
    DBMS_OUTPUT.PUT_LINE('DAYS BTWN DISPATCH & DELIVERY: ' || v_days);
    DBMS_OUTPUT.PUT_LINE('---------------------------------------------');
  END LOOP;

  IF v_count = 0 THEN
    DBMS_OUTPUT.PUT_LINE('No invoices found for customer 11011.');
  END IF;
END;
```

Script Output ×

Task completed in 4.251 seconds

```
    DBMS_OUTPUT.PUT_LINE('CUSTOMER:  ' || rec.customer_name);
    DBMS_OUTPUT.PUT_LINE('EMPLOYEE:  ' || rec.employee_name);
    DBMS_OUTPUT.PUT_LINE('DONATION:  ' || rec.donation);
    DBMS_OUTPUT.PUT_LINE('DISPATCH:  ' || TO_CHAR(rec.dispatch_date, 'YYYY-MM-DD'));
    DBMS_OUTPUT.PUT_LINE('DELIVERY:  ' || TO_CHAR(rec.delivery_date, 'YYYY-MM-DD'));
    DBMS_OUTPUT.PUT_LINE('DAYS BTWN DISPATCH 5: ' || v_days);
    DBMS_OUTPUT.PUT_LINE('---------------------------------------------');
  END LOOP;

  IF v_count = 0 THEN
    DBMS_OUTPUT.PUT_LINE('No invoices found for customer 11011.');
  END IF;
END;
CUSTOMER:  Jack Smith
EMPLOYEE:  Adanya Andrews
DONATION:  KIC Fridge
DISPATCH:  2024-05-10
DELIVERY:  2024-05-15
DAYS BTWN DISPATCH 5: 5
---------------------------------------------
CUSTOMER:  Jack Smith
EMPLOYEE:  Kevin Marks
DONATION:  Lazyboy Sofa
DISPATCH:  2024-05-18
DELIVERY:  2024-05-19
DAYS BTWN DISPATCH 5: 1
---------------------------------------------


PL/SQL procedure successfully completed.
```

Q6

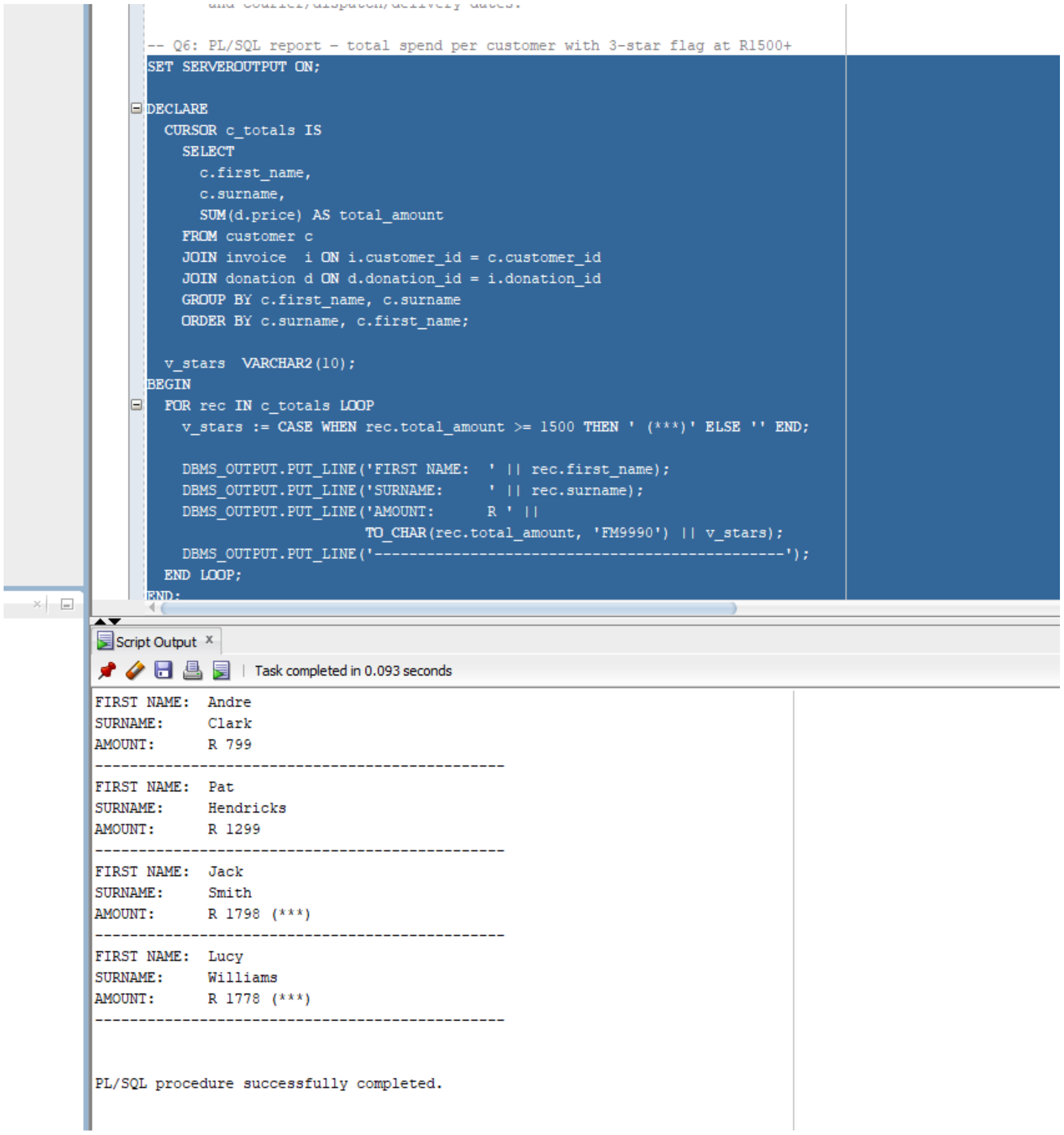
Code:

```
SET SERVEROUTPUT ON;

DECLARE

CURSOR c_totals IS

SELECT

c.first_name,

c.surname,

SUM(d.price) AS total_amount

FROM customer c

JOIN invoice i ON i.customer_id = c.customer_id

JOIN donation d ON d.donation_id = i.donation_id

GROUP BY c.first_name, c.surname

ORDER BY c.surname, c.first_name;

v_stars VARCHAR2(10);

BEGIN

FOR rec IN c_totals LOOP

v_stars := CASE WHEN rec.total_amount >= 1500 THEN ' (***)' ELSE '' END;
DBMS_OUTPUT.PUT_LINE('FIRST NAME:  ' || rec.first_name);
DBMS_OUTPUT.PUT_LINE('SURNAME:     ' || rec.surname);
DBMS_OUTPUT.PUT_LINE('AMOUNT:      R ' ||
                     TO_CHAR(rec.total_amount, 'FM9990') || v_stars);
DBMS_OUTPUT.PUT_LINE('-----------------------------------------------
');
```

END LOOP;

END;

Output:



```
-- Q6: PL/SQL report - total spend per customer with 3-star flag at R1500+
SET SERVEROUTPUT ON;

DECLARE
  CURSOR c_totals IS
    SELECT
      c.first_name,
      c.surname,
      SUM(d.price) AS total_amount
    FROM customer c
    JOIN invoice  i ON i.customer_id = c.customer_id
    JOIN donation d ON d.donation_id = i.donation_id
    GROUP BY c.first_name, c.surname
    ORDER BY c.surname, c.first_name;

  v_stars  VARCHAR2(10);
BEGIN
  FOR rec IN c_totals LOOP
    v_stars := CASE WHEN rec.total_amount >= 1500 THEN ' (***)' ELSE '' END;

    DBMS_OUTPUT.PUT_LINE('FIRST NAME:  ' || rec.first_name);
    DBMS_OUTPUT.PUT_LINE('SURNAME:     ' || rec.surname);
    DBMS_OUTPUT.PUT_LINE('AMOUNT:      R ' ||
                         TO_CHAR(rec.total_amount, 'FM9990') || v_stars);
    DBMS_OUTPUT.PUT_LINE('-----------------------------------------------');
  END LOOP;
END;
```

Script Output

Task completed in 0.093 seconds

```
FIRST NAME:  Andre
SURNAME:     Clark
AMOUNT:      R 799
-----------------------------------------------
FIRST NAME:  Pat
SURNAME:     Hendricks
AMOUNT:      R 1299
-----------------------------------------------
FIRST NAME:  Jack
SURNAME:     Smith
AMOUNT:      R 1798 (***)
-----------------------------------------------
FIRST NAME:  Lucy
SURNAME:     Williams
AMOUNT:      R 1778 (***)
-----------------------------------------------


PL/SQL procedure successfully completed.
```

Q7

Initial:

```
SET SERVEROUTPUT ON;

DECLARE

/* ===== %TYPE examples ===== */

v_cust_id customer.customer_id%TYPE := 11011; -- ties var type to column

v_total_spent donation.price%TYPE; -- matches NUMBER(8,2)

/* ===== %ROWTYPE example ===== */

r_invoice invoice%ROWTYPE; -- full row structure of INVOICE

/* ===== Exception for FK violations (ORA-02291) ===== */

e_fk_violation EXCEPTION;

PRAGMA EXCEPTION_INIT(e_fk_violation, -2291);

BEGIN
```

7.1

Code:

```
SELECT NVL(SUM(d.price), 0)

INTO v_total_spent

FROM invoice i

JOIN donation d ON d.donation_id = i.donation_id

WHERE i.customer_id = v_cust_id;

DBMS_OUTPUT.PUT_LINE('Total spent by customer ' || v_cust_id || ' = ' || v_total_spent);
```

7.2

Code:

```
SELECT * INTO r_invoice

FROM invoice

WHERE invoice_num = 8111; -- existing invoice from Q1 data

DBMS_OUTPUT.PUT_LINE('Invoice ' || r_invoice.invoice_num || ' | customer ' ||
r_invoice.customer_id || ' | employee ' || r_invoice.employee_id || ' | delivery ' ||
r_invoice.delivery_id);
```

7.3

Code:

```
/* ---- Exception handling #1: NO_DATA_FOUND ---- */

BEGIN

DECLARE v_missing_first customer.first_name%TYPE;

BEGIN

SELECT first_name INTO v_missing_first

FROM customer

WHERE customer_id = 99999; -- nonexistent on purpose
DBMS_OUTPUT.PUT_LINE('(Unexpected) Found: ' || v_missing_first);

EXCEPTION

WHEN NO_DATA_FOUND THEN

DBMS_OUTPUT.PUT_LINE('Handled NO_DATA_FOUND: customer 99999 does not exist.');
END;
```

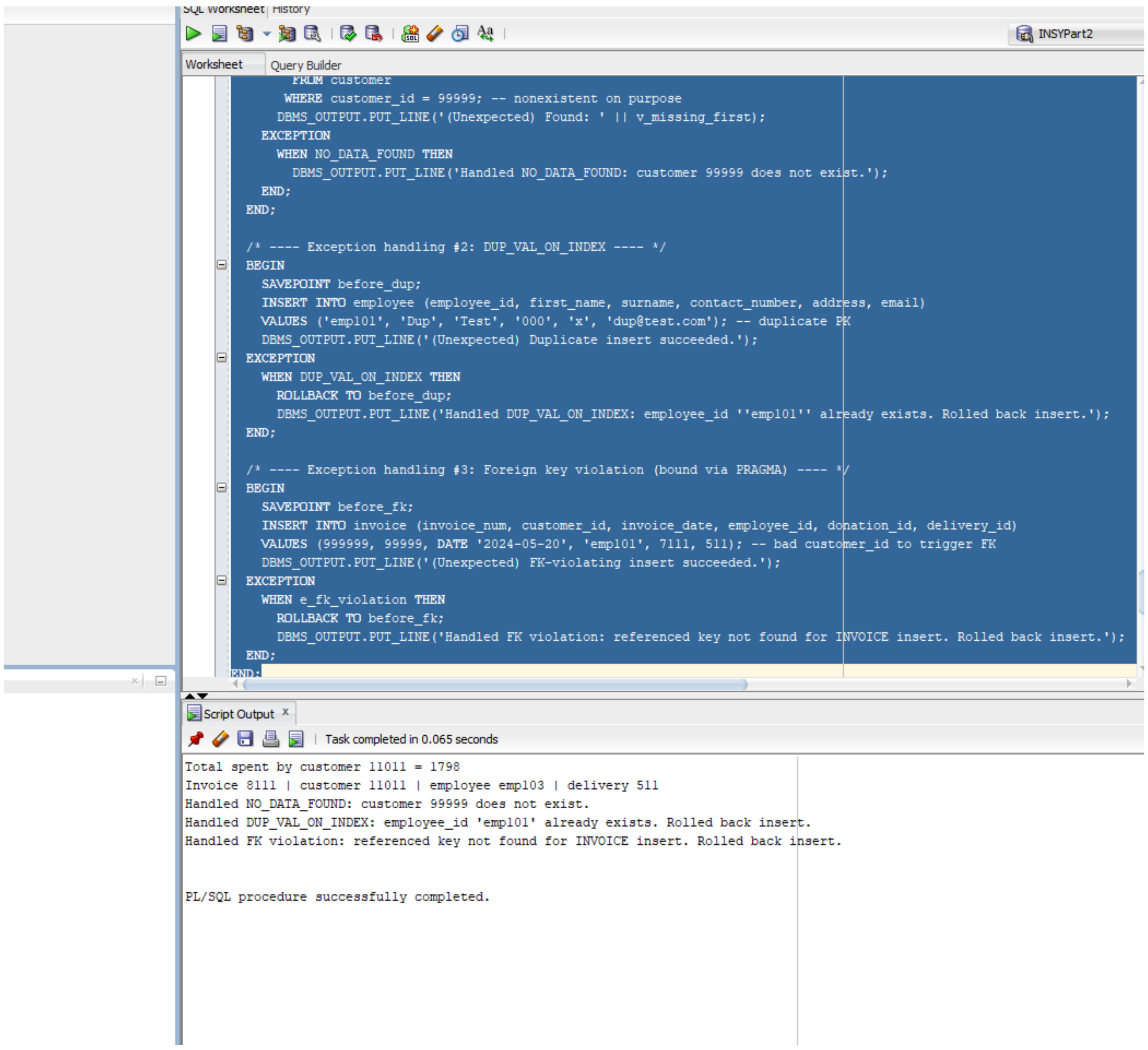
```
END;

/* ---- Exception handling #2: DUP_VAL_ON_INDEX ---- */

BEGIN

SAVEPOINT before_dup;

INSERT INTO employee (employee_id, first_name, surname, contact_number, address, email)

VALUES ('emp101', 'Dup', 'Test', '000', 'x', 'dup@test.com'); -- duplicate PK
DBMS_OUTPUT.PUT_LINE('(Unexpected) Duplicate insert succeeded.');

EXCEPTION

WHEN DUP_VAL_ON_INDEX THEN

ROLLBACK TO before_dup;

DBMS_OUTPUT.PUT_LINE('Handled DUP_VAL_ON_INDEX: employee_id ''emp101'' already exists. Rolled back insert.');

END;

/* ---- Exception handling #3: Foreign key violation (bound via PRAGMA) ---- */

BEGIN

SAVEPOINT before_fk;

INSERT INTO invoice (invoice_num, customer_id, invoice_date, employee_id, donation_id, delivery_id)

VALUES (999999, 99999, DATE '2024-05-20', 'emp101', 7111, 511); -- bad customer_id to trigger FK

DBMS_OUTPUT.PUT_LINE('(Unexpected) FK-violating insert succeeded.');

EXCEPTION

WHEN e_fk_violation THEN
```

ROLLBACK TO before_fk; DBMS_OUTPUT.PUT_LINE('Handled FK violation: referenced key not found for INVOICE insert. Rolled back insert.');

END;

END;

Output:



```
      FROM customer
     WHERE customer_id = 99999; -- nonexistent on purpose
    DBMS_OUTPUT.PUT_LINE('(Unexpected) Found: ' || v_missing_first);
  EXCEPTION
    WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('Handled NO_DATA_FOUND: customer 99999 does not exist.');
  END;
END;

/* ---- Exception handling #2: DUP_VAL_ON_INDEX ---- */
BEGIN
  SAVEPOINT before_dup;
  INSERT INTO employee (employee_id, first_name, surname, contact_number, address, email)
  VALUES ('empl01', 'Dup', 'Test', '000', 'x', 'dup@test.com'); -- duplicate PK
  DBMS_OUTPUT.PUT_LINE('(Unexpected) Duplicate insert succeeded.');
EXCEPTION
  WHEN DUP_VAL_ON_INDEX THEN
    ROLLBACK TO before_dup;
    DBMS_OUTPUT.PUT_LINE('Handled DUP_VAL_ON_INDEX: employee_id ''empl01'' already exists. Rolled back insert.');
END;

/* ---- Exception handling #3: Foreign key violation (bound via PRAGMA) ---- */
BEGIN
  SAVEPOINT before_fk;
  INSERT INTO invoice (invoice_num, customer_id, invoice_date, employee_id, donation_id, delivery_id)
  VALUES (999999, 99999, DATE '2024-05-20', 'empl01', 7111, 511); -- bad customer_id to trigger FK
  DBMS_OUTPUT.PUT_LINE('(Unexpected) FK-violating insert succeeded.');
EXCEPTION
  WHEN e_fk_violation THEN
    ROLLBACK TO before_fk;
    DBMS_OUTPUT.PUT_LINE('Handled FK violation: referenced key not found for INVOICE insert. Rolled back insert.');
END;
END;
```

```
Total spent by customer 11011 = 1798
Invoice 8111 | customer 11011 | employee emp103 | delivery 511
Handled NO_DATA_FOUND: customer 99999 does not exist.
Handled DUP_VAL_ON_INDEX: employee_id 'empl01' already exists. Rolled back insert.
Handled FK violation: referenced key not found for INVOICE insert. Rolled back insert.


PL/SQL procedure successfully completed.
```

Q8

Tiers:

- Platinum: >= 2000
- Gold: >= 1500
- Silver: >= 1000
- Bronze: < 1000

Code:

```
CREATE OR REPLACE VIEW vw_customer_rating AS

SELECT

c.customer_id,

c.first_name || ' ' || c.surname AS customer,

ROUND(SUM(d.price), 2) AS total_spent,

CASE

WHEN SUM(d.price) >= 2000 THEN 'Platinum'

WHEN SUM(d.price) >= 1500 THEN 'Gold'

WHEN SUM(d.price) >= 1000 THEN 'Silver'

ELSE 'Bronze'

END AS rating,

CASE

WHEN COUNT(r.return_id) > 0 THEN 'At-risk (has returns)'

ELSE 'OK'

END AS return_flag

FROM customer c

JOIN invoice i ON i.customer_id = c.customer_id
```

JOIN donation d ON d.donation_id = i.donation_id

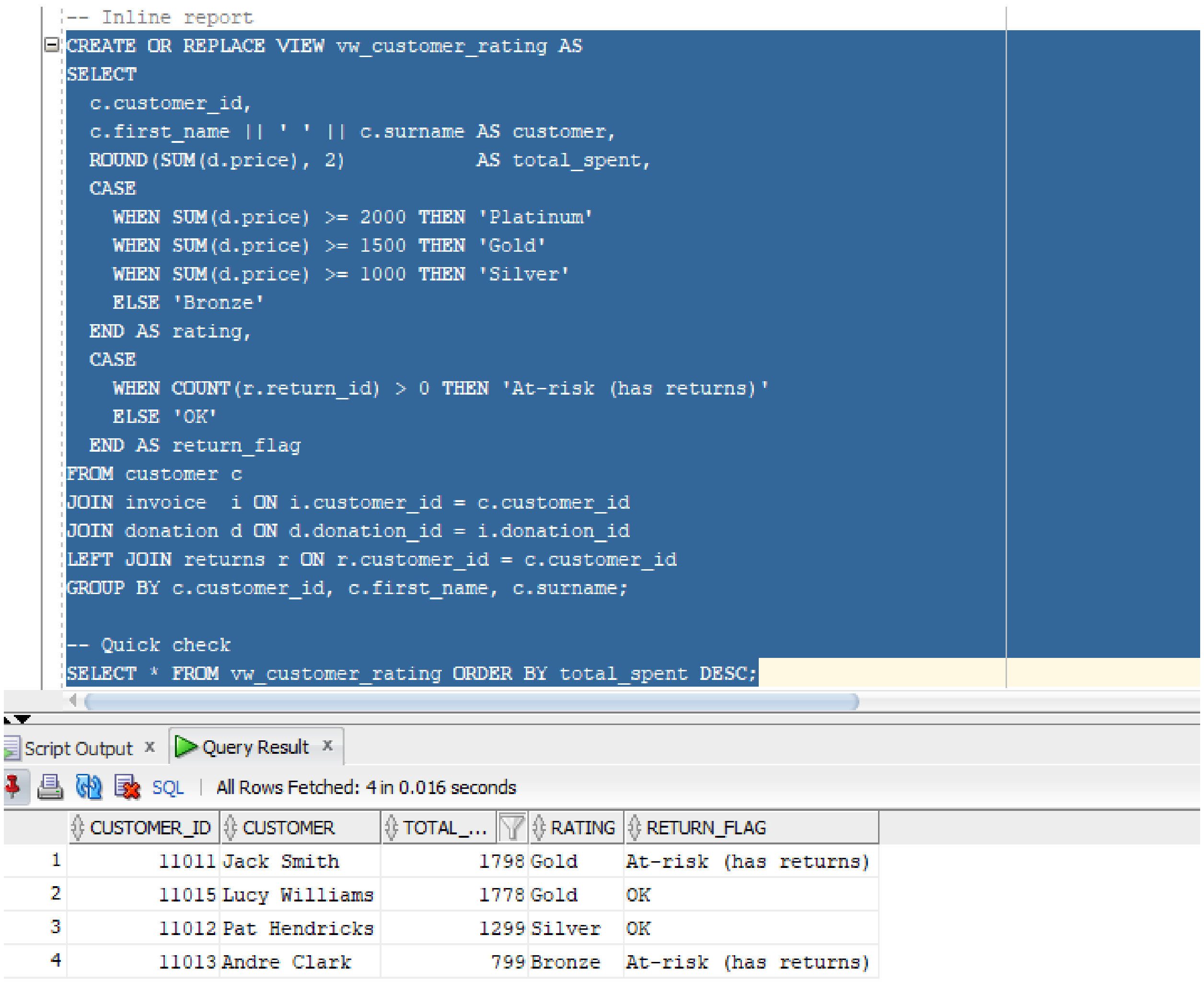
LEFT JOIN returns r ON r.customer_id = c.customer_id

GROUP BY c.customer_id, c.first_name, c.surname;

-- Quick check

SELECT * FROM vw_customer_rating ORDER BY total_spent DESC;

Output:



```
-- Inline report
CREATE OR REPLACE VIEW vw_customer_rating AS
SELECT
  c.customer_id,
  c.first_name || ' ' || c.surname AS customer,
  ROUND(SUM(d.price), 2)           AS total_spent,
  CASE
    WHEN SUM(d.price) >= 2000 THEN 'Platinum'
    WHEN SUM(d.price) >= 1500 THEN 'Gold'
    WHEN SUM(d.price) >= 1000 THEN 'Silver'
    ELSE 'Bronze'
  END AS rating,
  CASE
    WHEN COUNT(r.return_id) > 0 THEN 'At-risk (has returns)'
    ELSE 'OK'
  END AS return_flag
FROM customer c
JOIN invoice  i ON i.customer_id = c.customer_id
JOIN donation d ON d.donation_id = i.donation_id
LEFT JOIN returns r ON r.customer_id = c.customer_id
GROUP BY c.customer_id, c.first_name, c.surname;

-- Quick check
SELECT * FROM vw_customer_rating ORDER BY total_spent DESC;
```

All Rows Fetched: 4 in 0.016 seconds

	CUSTOMER_ID	CUSTOMER	TOTAL_...	RATING	RETURN_FLAG
1	11011	Jack Smith	1798	Gold	At-risk (has returns)
2	11015	Lucy Williams	1778	Gold	OK
3	11012	Pat Hendricks	1299	Silver	OK
4	11013	Andre Clark	799	Bronze	At-risk (has returns)