



ADDB 7311












ADVANCED DATABASES

ASSIGNMENT 1

NIVAD RAMDASS

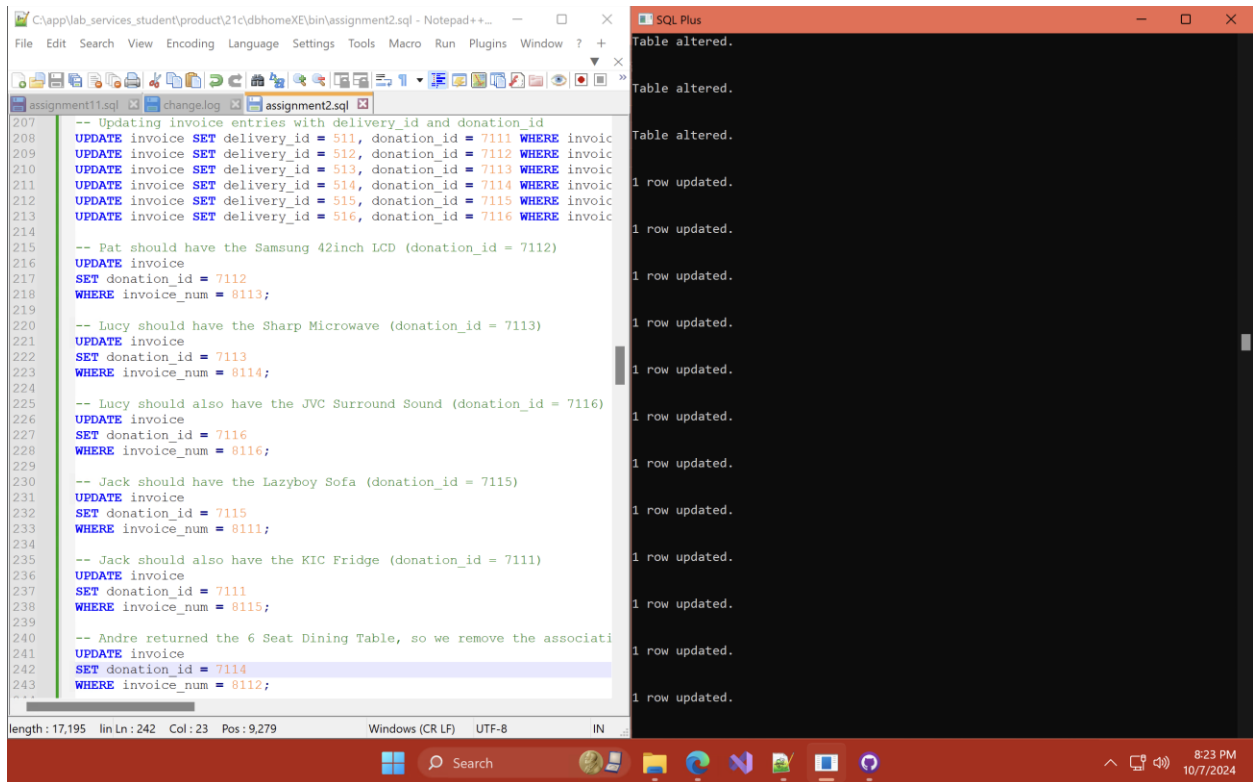
By submitting this assignment, I acknowledge that I have read and understood all the rules as per the terms in the registration contract, in particular the assignment and assessment rules in The IIE Assessment Strategy and Policy (IE009), the intellectual integrity and plagiarism rules in the Intellectual Integrity and Property Rights Policy (IE023), as well as any rules and regulations published in the student portal.

PLAGIARISM STATEMENTS

Statement	Signature	Statement	Signature
I have read the assessment rules provided in this declaration.		I have not shared this assessment with any other student.	
This assessment is my own work.		I have not presented the work of published sources as my own work.	
I have not copied any other student's work in this assessment.		I have correctly cited all my sources of information.	
I have not uploaded the assessment question to any website or App offering assessment assistance.		My referencing is technically correct, consistent, and congruent.	
I have not downloaded my assessment response from a website.		I have acted in an academically honest way in this assessment.	
I have not used any AI tool without reviewing, re-writing, and re-working this information, and referencing any AI tools in my work.			

Question 1: Dropping Tables

For this question, I wrote PL/SQL blocks using EXECUTE IMMEDIATE to drop existing tables if they existed. I used exception handling to ensure no errors were raised if the tables were already dropped.



```
-- Updating invoice entries with delivery_id and donation_id
207 UPDATE invoice SET delivery_id = 511, donation_id = 7111 WHERE invoice_id = 511;
208 UPDATE invoice SET delivery_id = 512, donation_id = 7112 WHERE invoice_id = 512;
209 UPDATE invoice SET delivery_id = 513, donation_id = 7113 WHERE invoice_id = 513;
210 UPDATE invoice SET delivery_id = 514, donation_id = 7114 WHERE invoice_id = 514;
211 UPDATE invoice SET delivery_id = 515, donation_id = 7115 WHERE invoice_id = 515;
212 UPDATE invoice SET delivery_id = 516, donation_id = 7116 WHERE invoice_id = 516;
213
214
215 -- Pat should have the Samsung 42inch LCD (donation_id = 7112)
216 UPDATE invoice
217 SET donation_id = 7112
218 WHERE invoice_num = 8113;
219
220 -- Lucy should have the Sharp Microwave (donation_id = 7113)
221 UPDATE invoice
222 SET donation_id = 7113
223 WHERE invoice_num = 8114;
224
225 -- Lucy should also have the JVC Surround Sound (donation_id = 7116)
226 UPDATE invoice
227 SET donation_id = 7116
228 WHERE invoice_num = 8116;
229
230 -- Jack should have the Lazyboy Sofa (donation_id = 7115)
231 UPDATE invoice
232 SET donation_id = 7115
233 WHERE invoice_num = 8111;
234
235 -- Jack should also have the KIC Fridge (donation_id = 7111)
236 UPDATE invoice
237 SET donation_id = 7111
238 WHERE invoice_num = 8115;
239
240 -- Andre returned the 6 Seat Dining Table, so we remove the association
241 UPDATE invoice
242 SET donation_id = 7114
243 WHERE invoice_num = 8112;
```

Table altered.
Table altered.
Table altered.
1 row updated.
1 row updated.
1 row updated.
1 row updated.
1 row updated.
1 row updated.
1 row updated.
1 row updated.
1 row updated.
1 row updated.
1 row updated.
1 row updated.
1 row updated.
1 row updated.
1 row updated.
1 row updated.
1 row updated.

Question 2: Creating and Populating Tables

I created several tables including customer, employee, donator, donation, delivery, and invoice, then inserted sample data into each. Additionally, I added donation_id and delivery_id columns to the invoice table to link it with the donation and delivery tables, ensuring proper relationships between these entities.

```

247 COLUMN invoice_num FORMAT 9999;
248
249 -- Set column widths for other fields as well if necessary
250 COLUMN customer_name FORMAT A15;
251 COLUMN employee_id FORMAT A10;
252 COLUMN delivery_notes FORMAT A20;
253 COLUMN donation_item FORMAT A15;
254 COLUMN invoice_date FORMAT A15;
255
256 -- Query to display all invoices after May 16th
257 SELECT
258     c.first_name || ' ' || c.surname AS customer_name,
259     i.employee_id,
260     d.delivery_notes,
261     don.item AS donation_item,
262     i.invoice_num,
263     i.invoice_date
264 FROM
265     invoice i
266 JOIN
267     customer c ON i.customer_id = c.customer_id
268 JOIN
269     delivery d ON i.delivery_id = d.delivery_id
270 JOIN
271     donation don ON i.donation_id = don.donation_id
272 WHERE
273     i.invoice_date > TO_DATE('16-MAY-2024', 'DD-MON-YYYY');
274
275 -- Question 3: Drop the funding table if it exists, avoiding errors w
276 BEGIN
277     EXECUTE IMMEDIATE 'DROP TABLE funding CASCADE CONSTRAINTS';
278 EXCEPTION
279     WHEN OTHERS THEN
280         IF SQLCODE != -942 THEN
281             RAISE;
282         END IF;
283 END;

```

CUSTOMER_NAME	EMPLOYEE_I	DELIVERY_NOTES	DONATION_ITEM	INVOICE_NUM
Jack Smith	emp102	Birthday present wra	KIC Fridge	8115
Pat Hendricks	emp101	Delivery to work add Samsung 42inch	LCD	8113
Lucy Williams	emp102	No notes	Sharp Microwave	8114
Lucy Williams	emp103	Delivery to work add JVC Surround So	und	8116

Question 3: Dropping and Creating the Funding Table

I dropped the funding table (if it existed) using EXECUTE IMMEDIATE, and then created a new funding table with an auto-incrementing funding_id. I inserted a row into the table to demonstrate its functionality.

Question 4: Displaying Returns

In this question, I used a PL/SQL block with a FOR loop to query customer returns. The query joined the customer, donation, and returns tables to display the customer's name, donation item, price, and reason for the return, using DBMS_OUTPUT to display the information.

```
C:\app\lab_services_student\product\21c\dbhomeXE\bin\assignment2.sql - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ? +
assignment1.sql changelog assignment2.sql
206 -- Creating the Funding table
207 CREATE TABLE funding (
208     funding_id NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY,
209     funder VARCHAR2(100) NOT NULL,
210     funding_amount NUMBER(10, 2)
211 );
212
213 -- Inserting data into funding table
214 INSERT INTO funding (funder, funding_amount) VALUES ('National Funding', 1000000);
215
216 -- Question 4: PL/SQL block to display returns with customer, donation item, price, return reason, and return date
217 BEGIN
218     FOR rec IN (
219         SELECT
220             c.first_name || ' ' || c.surname AS customer_name,
221             don.item AS donation_item,
222             don.price AS donation_price,
223             r.reason AS return_reason,
224             r.return_date
225         FROM
226             returns r
227         JOIN
228             invoice i ON r.invoice_num = i.invoice_num
229         JOIN
230             customer c ON i.customer_id = c.customer_id
231         JOIN
232             donation don ON i.donation_id = don.donation_id
233     )
234     LOOP
235         DBMS_OUTPUT.PUT_LINE('Customer: ' || rec.customer_name);
236         DBMS_OUTPUT.PUT_LINE('Donation Item: ' || rec.donation_item);
237         DBMS_OUTPUT.PUT_LINE('Price: ' || TO_CHAR(rec.donation_price));
238         DBMS_OUTPUT.PUT_LINE('Return Reason: ' || rec.return_reason);
239         DBMS_OUTPUT.PUT_LINE('Return Date: ' || TO_CHAR(rec.return_date));
240         DBMS_OUTPUT.PUT_LINE('-----');
241     END LOOP;
242 END;
```

```
SQL Plus
Lucy Williams emp102 No notes Sharp Microwave 8114
CUSTOMER_NAME EMPLOYEE_I DELIVERY_NOTES DONATION_ITEM INVOICE_NUM
INVOICE_DATE
-----
17-MAY-24
Lucy Williams emp103 Delivery to work add JVC Surround So 8116
ress und
18-MAY-24
PL/SQL procedure successfully completed.
Table created.
1 row created.
Customer: Jack Smith
Donation Item: Lazyboy Sofa
Price: 1199.00
Return Reason: Customer not satisfied with product
Return Date: 25-MAY-2024
-----
Customer: Pat Hendricks
Donation Item: Samsung 42inch LCD
Price: 1299.00
Return Reason: Product had broken section
Return Date: 25-MAY-2024
-----
PL/SQL procedure successfully completed.
CUSTOMER: Jack Smith
EMPLOYEE: Adanva Adebayo
DONATION: Lazyboy Sofa
DISPATCH DATE: 10-MAY-2024
DELIVERY DATE: 15-MAY-2024
DAYS TO DELIVERY: 5
-----
```

Question 5: Customer Deliveries

I wrote a PL/SQL block to display deliveries for a specific customer (Jack). The block retrieved the customer’s name, employee handling the delivery, the donation item, and the delivery dates, along with the number of days between dispatch and delivery. This information was output using DBMS_OUTPUT.

```

-- Question 5
SET SERVEROUTPUT ON;

-- PL/SQL block to display customer 11011's deliveries with employee,
BEGIN
  FOR rec IN (
    SELECT
      c.first_name || ' ' || c.surname AS customer_name,
      e.first_name || ' ' || e.surname AS employee_name,
      don.item AS donation_item,
      d.dispatch_date,
      i.delivery_date,
      (i.delivery_date - d.dispatch_date) AS days_to_delivery
    FROM
      invoice i
    JOIN
      customer c ON i.customer_id = c.customer_id
    JOIN
      employee e ON i.employee_id = e.employee_id
    JOIN
      donation don ON i.donation_id = don.donation_id
    JOIN
      delivery d ON i.delivery_id = d.delivery_id
    WHERE
      i.customer_id = 11011 -- Only display information for customer 11011
  )
  LOOP
    -- Output each record
    DBMS_OUTPUT.PUT_LINE('CUSTOMER: ' || rec.customer_name);
    DBMS_OUTPUT.PUT_LINE('EMPLOYEE: ' || rec.employee_name);
    DBMS_OUTPUT.PUT_LINE('DONATION: ' || rec.donation_item);
    DBMS_OUTPUT.PUT_LINE('DISPATCH DATE: ' || TO_CHAR(rec.dispatch_date, 'YYYY-MM-DD'));
    DBMS_OUTPUT.PUT_LINE('DELIVERY DATE: ' || TO_CHAR(rec.delivery_date, 'YYYY-MM-DD'));
    DBMS_OUTPUT.PUT_LINE('DAYS TO DELIVERY: ' || rec.days_to_delivery);
    DBMS_OUTPUT.PUT_LINE('-----');
  END LOOP;
END;

```

```

Return Reason: Customer not satisfied with product
Return Date: 25-MAY-2024
-----
Customer: Pat Hendricks
Donation Item: Samsung 42inch LCD
Price: 1299.00
Return Reason: Product had broken section
Return Date: 25-MAY-2024
-----
PL/SQL procedure successfully completed.

CUSTOMER: Jack Smith
EMPLOYEE: Adanva Adebayo
DONATION: Lazyboy Sofa
DISPATCH DATE: 10-MAY-2024
DELIVERY DATE: 15-MAY-2024
DAYS TO DELIVERY: 5
-----
CUSTOMER: Jack Smith
EMPLOYEE: Kevin Marks
DONATION: KIC Fridge
DISPATCH DATE: 18-MAY-2024
DELIVERY DATE: 19-MAY-2024
DAYS TO DELIVERY: 1
-----
PL/SQL procedure successfully completed.

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

```

Question 6: Total Amount Spent by Each Customer

This PL/SQL block aggregated the total amount spent by each customer based on their donations. I grouped by the customer's name and used SUM to calculate the total donation value, applying a rating system for customers based on the amount they spent. The results were displayed using DBMS_OUTPUT.

```

-- Question 6
SET SERVEROUTPUT ON;

-- PL/SQL block to display the combined customer name and total amount
BEGIN
  FOR rec IN (
    SELECT
      c.first_name,
      c.surname,
      SUM(don.price) AS total_amount
    FROM
      invoice i
    JOIN
      customer c ON i.customer_id = c.customer_id
    JOIN
      donation don ON i.donation_id = don.donation_id
    GROUP BY
      c.first_name, c.surname
  ) LOOP
    -- Output each record
    DBMS_OUTPUT.PUT_LINE('FIRST NAME: ' || rec.first_name);
    DBMS_OUTPUT.PUT_LINE('SURNAME: ' || rec.surname);

    -- Calculate and display the amount with rating
    IF rec.total_amount >= 1500 THEN
      DBMS_OUTPUT.PUT_LINE('AMOUNT: R ' || TO_CHAR(rec.total_amount));
    ELSE
      DBMS_OUTPUT.PUT_LINE('AMOUNT: R ' || TO_CHAR(rec.total_amount));
    END IF;

    DBMS_OUTPUT.PUT_LINE('-----');
  END LOOP;
END;
/

```

```

Customer: Pat Hendricks
Donation Item: Samsung 42inch LCD
Price: 1299.00
Return Reason: Product had broken section
Return Date: 25-MAY-2024

PL/SQL procedure successfully completed.

CUSTOMER: Jack Smith
EMPLOYEE: Adanva Adebayo
DONATION: Lazyboy Sofa
DISPATCH DATE: 18-MAY-2024
DELIVERY DATE: 15-MAY-2024
DAYS TO DELIVERY: 5

CUSTOMER: Jack Smith
EMPLOYEE: Kevin Marks
DONATION: KIC Fridge
DISPATCH DATE: 18-MAY-2024
DELIVERY DATE: 19-MAY-2024
DAYS TO DELIVERY: 1

PL/SQL procedure successfully completed.

FIRST NAME: Jack
SURNAME: Smith
AMOUNT: R 1798 (***)

FIRST NAME: Pat
SURNAME: Hendricks
AMOUNT: R 1299

FIRST NAME: Lucy
SURNAME: Williams
AMOUNT: R 1778 (***)

FIRST NAME: Andre
SURNAME: Clark
AMOUNT: R 799

PL/SQL procedure successfully completed.

```

Question 7.1: %TYPE Attribute Example

In this question, we used the %TYPE attribute to declare a variable of the same data type as the email column in the customer table. We assigned a specific customer ID (Jack's customer_id = 11011) and retrieved his email address using a SELECT INTO query, then displayed it using DBMS_OUTPUT.

Question 7.2: %ROWTYPE Attribute Example

Here, we used the %ROWTYPE attribute to declare a record that mirrors the structure of a row from the customer table. We retrieved all of Jack's details (name, address, email, etc.) and displayed them using DBMS_OUTPUT.

Question 7.3: User-Defined Exception Example

This question involved creating a custom PL/SQL exception. We queried the total donations for a customer and raised the custom no_donations_found exception if the customer had not made any donations. The exception was handled with a DBMS_OUTPUT message indicating no donations were made.

The screenshot shows a Notepad++ window with a PL/SQL script and an SQL Plus window displaying the execution results. The script includes a procedure to retrieve customer details and another to calculate total donations with a user-defined exception.

```

SET SERVEROUTPUT ON;

DECLARE
-- Declaring a record that holds the same structure as a row in t
customer_record customer%ROWTYPE;
BEGIN
-- Retrieving a full row for customer with ID 11011
SELECT * INTO customer_record
FROM customer
WHERE customer_id = 11011;

-- Output the customer details to the console
DBMS_OUTPUT.PUT_LINE('Customer Name: ' || customer_record.first_name);
DBMS_OUTPUT.PUT_LINE('Address: ' || customer_record.address);
DBMS_OUTPUT.PUT_LINE('Email: ' || customer_record.email);
DBMS_OUTPUT.PUT_LINE('Contact Number: ' || customer_record.contact_number);
END;
/
-- End of Q7.2

-- Q7.3: Example of User-Defined Exception
SET SERVEROUTPUT ON;

DECLARE
-- User-defined exception to handle the case where a customer has
no_donations_found EXCEPTION;
total_donations NUMBER;
customer_id customer.customer_id%TYPE := 11013; -- Define the customer ID
BEGIN
-- Calculate the total donations made by a specific customer
SELECT SUM(don.price) INTO total_donations
FROM invoice i
JOIN donation don ON i.donation_id = don.donation_id
WHERE i.customer_id = customer_id;

-- If no donations are found (NULL or 0 total), raise the custom

```

The SQL Plus window shows the following output:

```

PL/SQL procedure successfully completed.

Customer Name: Jack Smith
Address: 18 Water Rd
Email: smith@isat.com
Contact Number: 0877277521

PL/SQL procedure successfully completed.

Total Donations by Customer 11013: R 5674

PL/SQL procedure successfully completed.

FIRST_NAME
-----
SURNAME
-----
AMOUNT          CUS
-----
Andre
Clark
R 799          *

Jack
Smith
R 1798        ***

FIRST_NAME
-----
SURNAME
-----
AMOUNT          CUS
-----
Lucy
Williams
R 1778        ***

Pat
Hendricks

```

Question 8: Customer Ratings and Amounts

We used a query to calculate the total amount spent by each customer on donations and applied a rating system based on the total amount. The query included a CASE statement to assign star ratings and displayed the customer's name, total amount spent, and their rating.

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ? +

assignment11.sql change.log assignment2.sql

```
457
458
459 -- Question 8
460 SELECT
461     c.first_name,
462     c.surname,
463     'R ' || SUM(don.price) AS amount,
464     CASE
465         WHEN SUM(don.price) >= 1500 THEN '***' -- 3-star rating
466         WHEN SUM(don.price) BETWEEN 1000 AND 1499 THEN '**' -- 2-star rating
467         ELSE '*' -- 1-star rating
468     END AS customer_rating
469 FROM
470     invoice i
471 JOIN
472     customer c ON i.customer_id = c.customer_id
473 JOIN
474     donation don ON i.donation_id = don.donation_id
475 GROUP BY
476     c.first_name, c.surname
477 ORDER BY
478     c.first_name;
479
480 SELECT
481     c.first_name || ' ' || c.surname AS customer_name,
482     don.item AS donation_item,
483     i.invoice_date
484 FROM
485     customer c
486 JOIN
487     invoice i ON c.customer_id = i.customer_id
488 JOIN
489     donation don ON i.donation_id = don.donation_id
490 ORDER BY
491     c.first_name, c.surname;
492
```

length: 17,195 lin Ln: 271 Col: 28 Pos: 10,029 Windows (CR LF) UTF-8 IN

SQL Plus

Contact Number: 0877277521

PL/SQL procedure successfully completed.

Total Donations by Customer 11013: R 5674

PL/SQL procedure successfully completed.

FIRST_NAME	SURNAME	AMOUNT	CUS
Andre	Clark	R 799	*
Jack	Smith	R 1798	***
Lucy	Williams	R 1778	***
Pat	Hendricks	R 1299	**

8:24 PM

10/7/2024