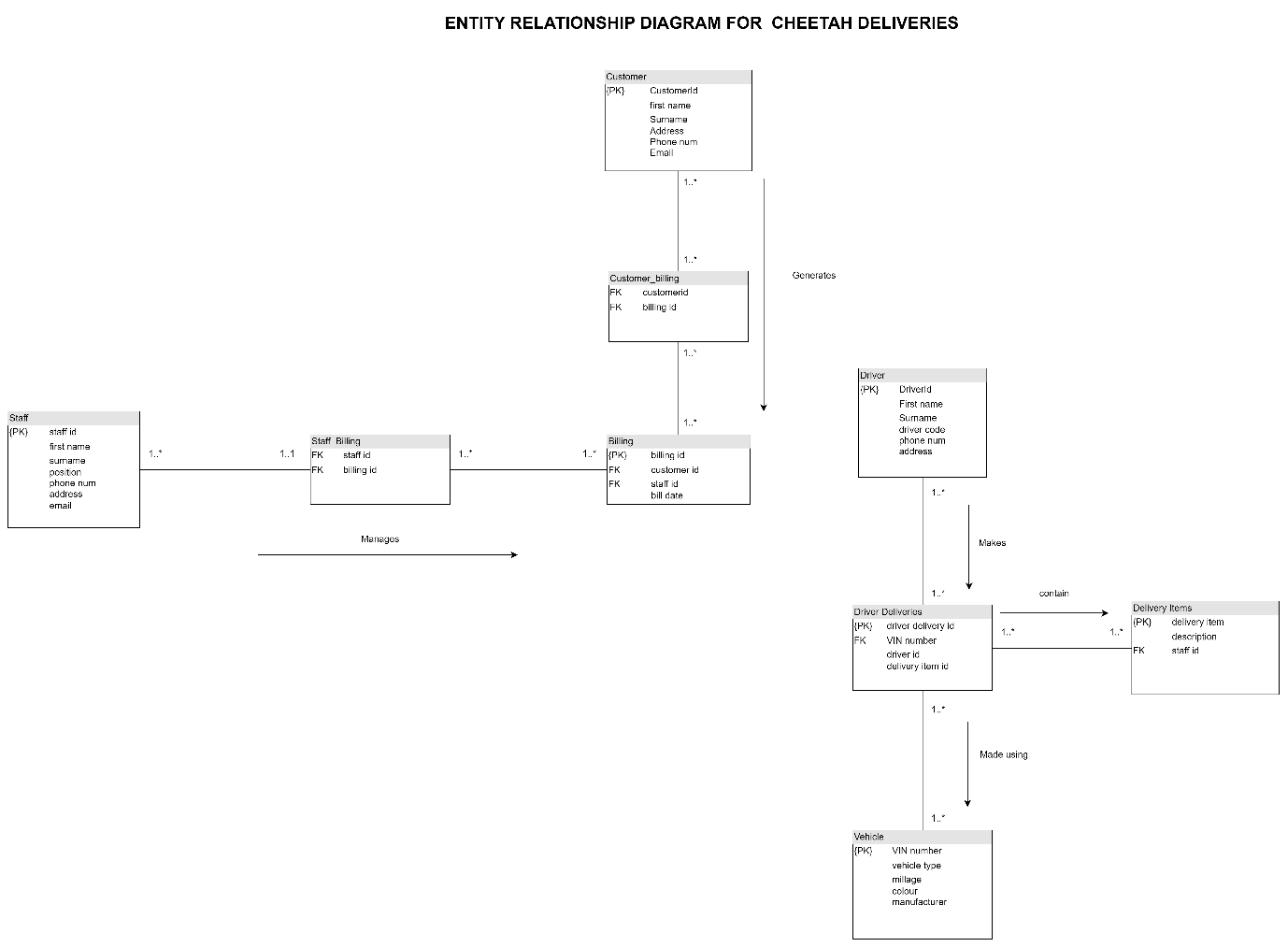
|  |
| --- |
| Stock market trading screen |
| ADDB7311 PRACTICAL ASSIGNMENT  SUBMISSION |
| |  |  |  | | --- | --- | --- | | Onello Travis Tarjanne |  | ADDB7311 | |

ADDB7311 PRACTICAL ASSIGNMENT

Question 1



Question 2

Code:

-- customers table

CREATE TABLE customers (

Customer\_id NUMBER,

first\_name VARCHAR2(50),

Surname VARCHAR2(50),

address VARCHAR2(50),

Phone\_num NUMBER,

email VARCHAR2(50)

);

SELECT \* FROM customers

-- driver table

CREATE TABLE driver (

Driver\_id NUMBER,

first\_name VARCHAR2(50),

Surname VARCHAR2(50),

Driver\_code VARCHAR2(50),

Phone\_num NUMBER,

address VARCHAR2(50)

);

SELECT \* FROM driver;

-- billing table

CREATE TABLE billing (

Bill\_id NUMBER,

customer\_id NUMBER,

staff\_id NUMBER,

bill\_date Date

);

SELECT \* FROM billing;

-- delivery items table

CREATE TABLE delivery\_items (

delivery\_item NUMBER,

description VARCHAR2(50),

staff\_id NUMBER

);

SELECT \* FROM delivery\_items;

-- deliverdeliveries

CREATE TABLE driverdeliveries (

driver\_delivery\_id NUMBER,

vin\_number VARCHAR2(50),

driver\_code NUMBER,

delivery NUMBER

);

SELECT \* FROM driverdeliveries;

-- staff table

CREATE TABLE staff (

staff\_id NUMBER,

first\_name VARCHAR2(50),

Surname VARCHAR2(50),

position VARCHAR2(50),

Phone\_num NUMBER,

address VARCHAR2(50),

email VARCHAR2(50)

);

SELECT \* FROM staff;

-- vehicle table

CREATE TABLE vehicle (

vin\_number VARCHAR2(50),

Surname VARCHAR2(50),

milage NUMBER,

address VARCHAR2(50),

email VARCHAR2(50)

);

SELECT \* FROM vehicles;

Import process:

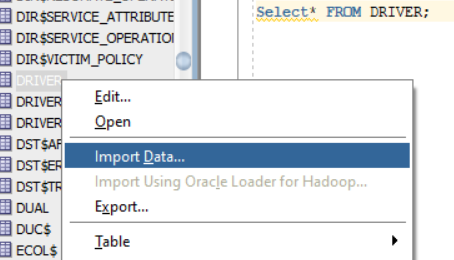


Figure 1 Importing option

A screenshot of a computer

Description automatically generated

Figure 2 Data import

A screenshot of a computer

Description automatically generated

Figure 3 Import method and selection

Tables:

A screenshot of a computer

Description automatically generated

Figure 4 Driver Table

A screenshot of a computer

Description automatically generated

Figure 5 Vehicle Table

A screenshot of a computer

Description automatically generated

Figure 6 Billing table

A screenshot of a computer

Description automatically generated

Figure 7 Customers Table

A screenshot of a computer

Description automatically generated

Figure 8 Staff table

A screenshot of a computer

Description automatically generated

Figure 9 driver deliveries table

A screenshot of a computer

Description automatically generated

Figure 10 Delivery items table

Question 3

3.1)Code:

BEGIN

-- Create user John

CREATE USER John IDENTIFIED BY Johnch2024;

-- Grant privilege to Jonh to select from any table in the database

GRANT SELECT ANY TABLE TO John;

-- Create user Hannah

CREATE USER Hannah IDENTIFIED BY Hannahch2024;

-- Grant privilege to Hannah to insert into any table in the database

GRANT INSERT ANY TABLE TO Hannah;

END;

/

3.2) The separation eliminates any confusion that may occur between the two users, they may have different roles, and the privileges are based on those roles.

Question 4

4.1)Code:

SELECT

d.Driver\_code AS "CODE",

d.first\_name || ' ' || d.Surname AS "Driver",

v.vin\_number AS "VIN NUMBER",

v.milage AS "MILAGE"

FROM

driver d

JOIN

driverdeliveries dd ON d.Driver\_code = dd.driver\_code

JOIN

vehicles v ON dd.vin\_number = v.vin\_number

WHERE

v.milage < 80000;

4.2) A flat file table stores data in a single file or table, with no relationships between the entities, while a relational database has multiple tables representing different entities with primary keys and foreign keys. The relational database model would be ideal for cheetah deliveries and this model is highly scalable and enforces data security with the use of primary and foreign keys.

Question 5

5.1) Code:

DECLARE

v\_staff\_id staff.staff\_id%TYPE;

v\_first\_name staff.first\_name%TYPE;

v\_surname staff.surname%TYPE;

v\_billing\_count NUMBER;

v\_max\_bills NUMBER := 0;

v\_top\_staff\_id staff.staff\_id%TYPE;

v\_top\_first\_name staff.first\_name%TYPE;

v\_top\_surname staff.surname%TYPE;

-- Cursor to select staff details

CURSOR c\_staff IS

SELECT s.staff\_id, s.first\_name, s.surname

FROM staff s;

BEGIN

FOR r\_staff IN c\_staff LOOP

v\_staff\_id := r\_staff.staff\_id;

v\_first\_name := r\_staff.first\_name;

v\_surname := r\_staff.surname;

-- Calculate the number of deliveries processed by the staff member

SELECT COUNT(\*)

INTO v\_billing\_count

FROM billing b

WHERE b.staff\_id = v\_staff\_id;

IF v\_billing\_count > v\_max\_bills THEN

v\_max\_bills := v\_billing\_count;

v\_top\_staff\_id := v\_staff\_id;

v\_top\_first\_name := v\_first\_name;

v\_top\_surname := v\_surname;

END IF;

END LOOP;

-- Output the report

DBMS\_OUTPUT.PUT\_LINE('----------------------------');

DBMS\_OUTPUT.PUT\_LINE('Staff ID: ' || v\_top\_staff\_id);

DBMS\_OUTPUT.PUT\_LINE('First name: ' || v\_top\_first\_name);

DBMS\_OUTPUT.PUT\_LINE('Surname: ' || v\_top\_surname);

DBMS\_OUTPUT.PUT\_LINE('Deliveries processed: ' || v\_max\_bills);

DBMS\_OUTPUT.PUT\_LINE('----------------------------');

END;

/

5.2) The 3 components of a SQL/PL block are the declaration block, the execution block and the exception block. In the declaration block all the variables are declared to hold the staff details displayed in the report. The execution block is where all the methods are executed and the logic of the project is, the count of the employees would be done there. Lastly is the exception block where all the error handling would be done.

5.3.1) A SQL view captures the query logic and simplifies the process of retrieving the necessary information, the simplicity eliminates the need for understanding the underlying SQL logic. SQL views can also be reused and be used to restrict access to specific data.

5.3.2)

-- creating the view

CREATE VIEW staff\_delivery\_report AS

-- selects the details for the

SELECT

s.staff\_id,

s.first\_name,

s.surname,

-- counts the items processed

COUNT(d.delivery\_item) AS delivery\_processed

FROM

staff s

LEFT JOIN

delivery\_items d ON s.staff\_id = d.staff\_id

-- groups all the details for the report

GROUP BY

s.staff\_id, s.first\_name, s.surname;

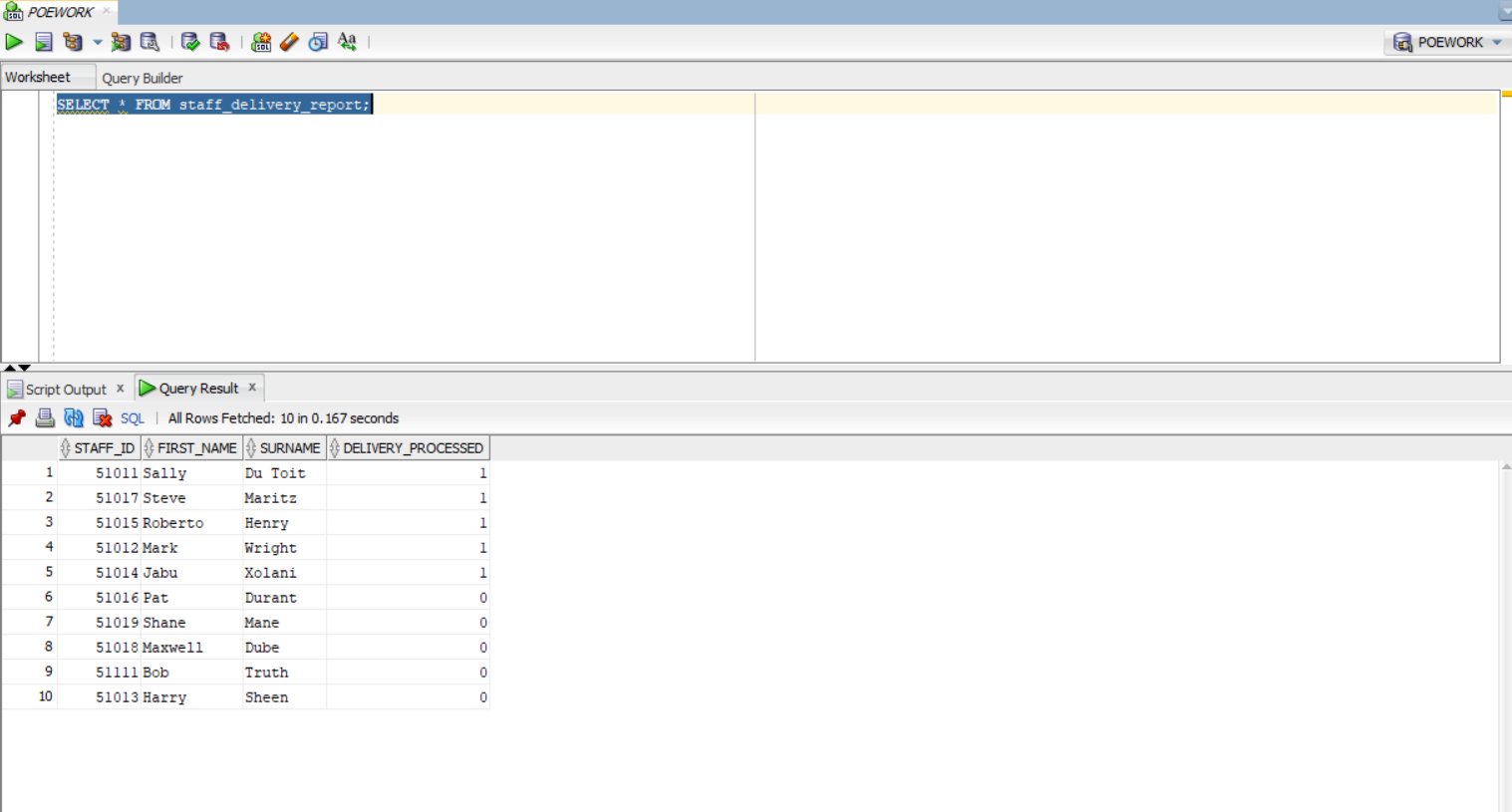


Figure 11 The view of all processed orders

Question 6

6.1) a) Implicit cursors are automatically generated by Oracle while an SQL statement is executed and used to process statements like INSERT, UPDATE, DELETE, in this scenario we would use implicit cursors to count the rows based on what is required.

b) Explicit cursors need to be defined explicitly by the user by providing a name and need to be created on a select statement. We would make use of them to fetch multiple rows as implicit can only fetch one at a time.

2.

DECLARE

total\_rows NUMBER;

BEGIN

-- Implicit cursor

UPDATE driver

SET address = '26Carling Ave'

WHERE first\_name = 'Wayne' AND surname = 'Smith';

IF SQL%NOTFOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No drivers updated');

ELSE

total\_rows := SQL%ROWCOUNT;

DBMS\_OUTPUT.PUT\_LINE(total\_rows || ' driver(s) updated');

END IF;

END;

/

DECLARE

c\_customer\_id customers.customer\_id%TYPE;

c\_name customers.name%TYPE;

c\_addr customers.address%TYPE;

CURSOR c\_customers IS -- Cursor declaration for explicit cursor

SELECT customer\_id, name, address FROM customers;

BEGIN

OPEN c\_customers;

LOOP

FETCH c\_customers INTO c\_customer\_id, c\_name, c\_addr;

EXIT WHEN c\_customers%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(c\_customer\_id || ' ' || c\_name || ' ' || c\_addr);

END LOOP;

CLOSE c\_customers;

END;

/

6.2)A sequence would be used for the customer ID in the database, a customer sequence could be used.

Code:

-- Creating sequence

CREATE SEQUENCE customer\_seq

START WITH 1

INCREMENT BY 1

NOCACHE

NOCYCLE;

-- Insert statements

INSERT INTO customers (customer\_id, first\_name, surname, address, phone\_num, email)

VALUES (customer\_seq.NEXTVAL, 'Autumn', 'James', '56 Hathaway St', 037373837, 'aj@inkedlink.com'),

(customer\_seq.NEXTVAL, 'James', 'Taylor', 'Duke St', 0873457765, 'johndoe@example.com');

A screenshot of a computer

Description automatically generated

Figure 12 Output for sequence after inserting 3 values

# REFERENCES

PL/SQL cursor - javatpoint (2024) www.javatpoint.com. Available at: https://www.javatpoint.com/pl-sql-cursor (Accessed: 04 September 2024).

GeeksforGeeks (2022) Difference between implicit and explicit cursors, GeeksforGeeks. Available at: https://www.geeksforgeeks.org/difference-between-implicit-and-explicit-cursors/ (Accessed: 04 September 2024).

PL/SQL cursor - javatpoint (2023) www.javatpoint.com. Available at: https://www.javatpoint.com/pl-sql-cursor (Accessed: 04 September 2024).

Gaetjen, S., Knox, D. and Maroulis, W. (2015) Oracle Database 12C security, O’Reilly Online Learning. Available at: https://www.oreilly.com/library/view/oracledatabase-12c/9780071824286/ (Accessed: 27 August 2024).

IIE(2024) ADVANCED DATABASES MODULE MANUAL Available at: <https://advtechonline.sharepoint.com/sites/TertiaryStudents/IIE%20Student%20Materials/Forms/Default%20View.aspx?id=%2Fsites%2FTertiaryStudents%2FIIE%20Student%20Materials%2FNew%20Student%20Materials%20CAT%2FADDB7311%2F2024%2FADDB7311MM%2Epdf&viewid=db15e059%2D4f93%2D487f%2Dabda%2De538b821c7b8&parent=%2Fsites%2FTertiaryStudents%2FIIE%20Student%20Materials%2FNew%20Student%20Materials%20CAT%2FADDB7311%2F2024>