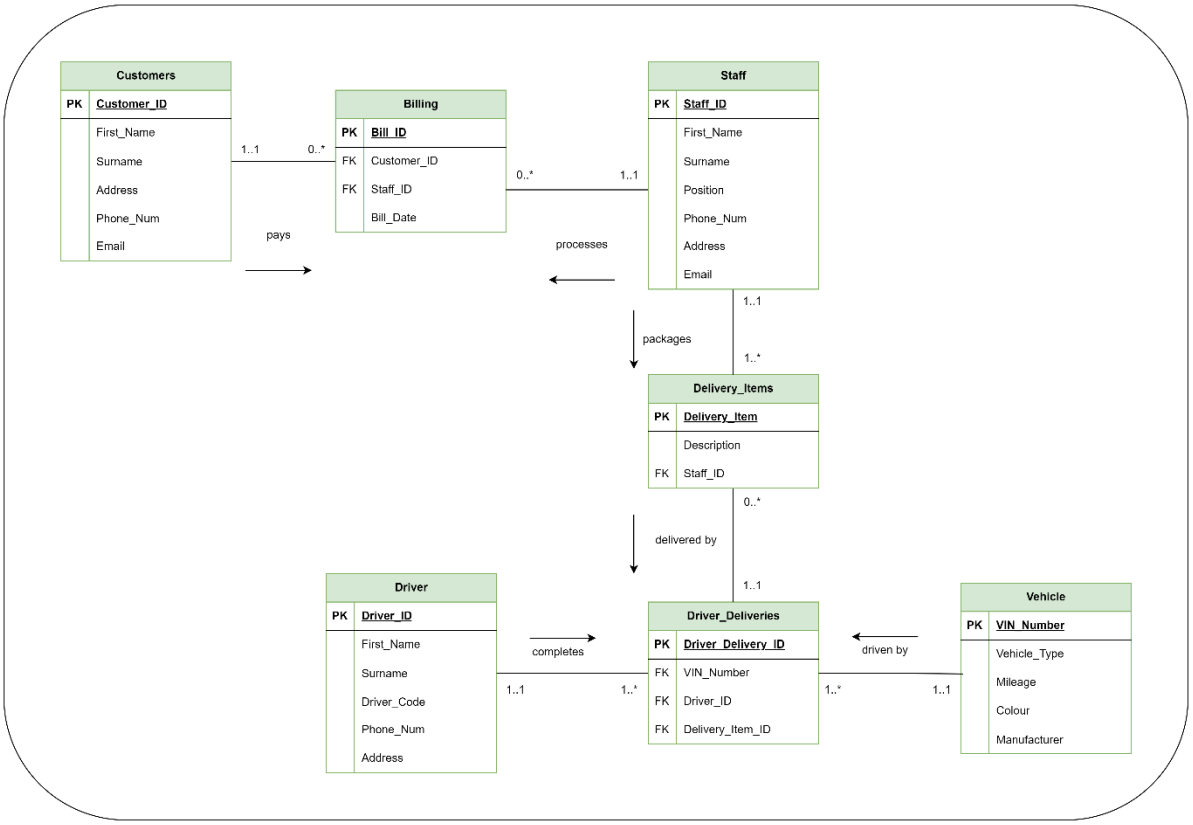
5/9/2024

ADDB 7311

Assignment 1

Sajana Bidesi
ST10249843

QUESTION 1


Customers
PK Customer_ID
First_Name
Surname
Address
Phone_Num
Email
1..1
0..*
pays
Billing
PK Bill_ID
FK Customer_ID
FK Staff_ID
Bill_Date
0..*
1..1
processes
Staff
PK Staff_ID
First_Name
Surname
Position
Phone_Num
Address
Email
1..1
packages
1..*
Delivery_Items
PK Delivery_Item
Description
FK Staff_ID
0..*
delivered by
1..1
Driver
PK Driver_ID
First_Name
Surname
Driver_Code
Phone_Num
Address
completes
1..1
1..*
Driver_Deliveries
PK Driver_Delivery_ID
FK VIN_Number
FK Driver_ID
FK Delivery_Item_ID
1..*
1..1
driven by
Vehicle
PK VIN_Number
Vehicle_Type
Mileage
Colour
Manufacturer

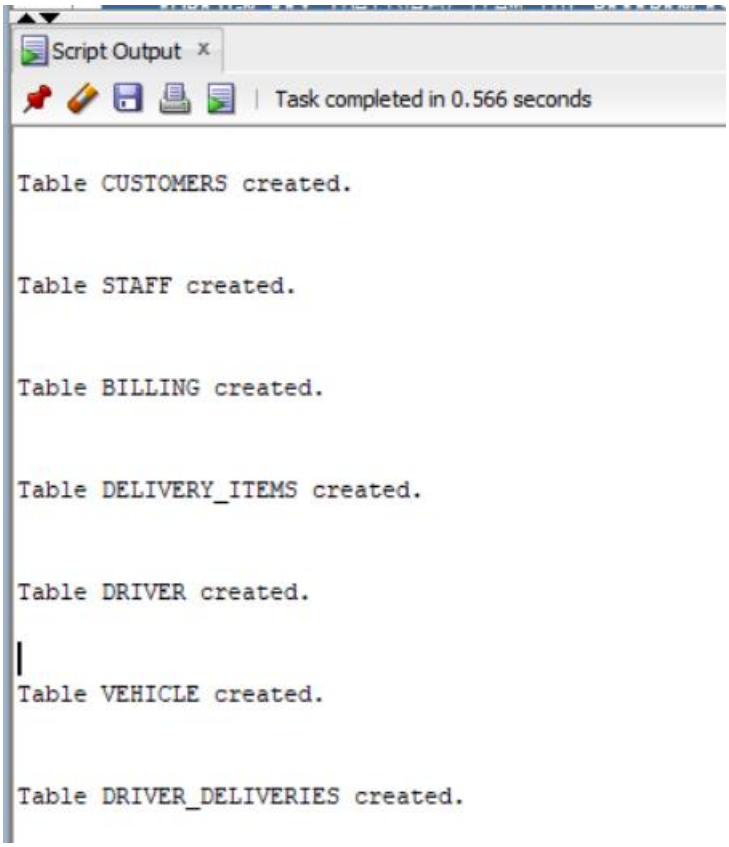
QUESTION 2

Creating tables

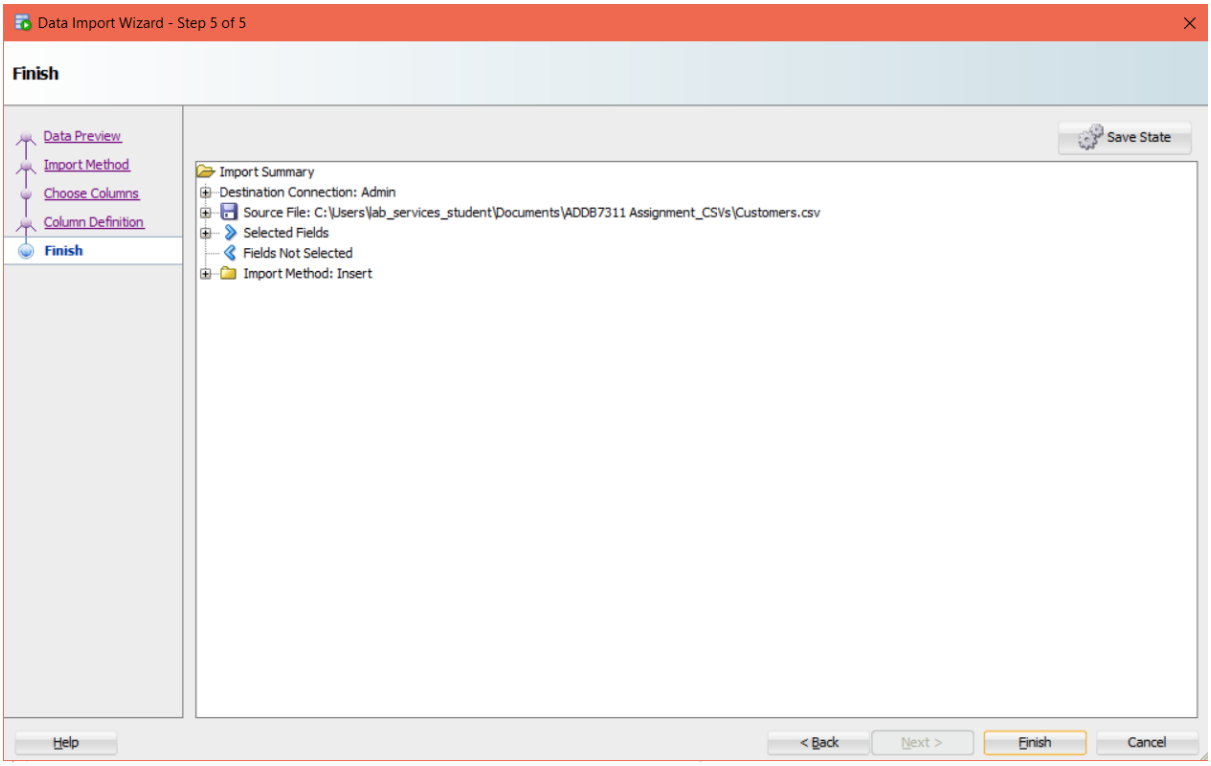
```
-- Create the Customers table
CREATE TABLE Customers (
   Customer_ID NUMBER PRIMARY KEY NOT NULL,
   First_Name VARCHAR(50) NOT NULL,
   Surname VARCHAR(50) NOT NULL,
   Address VARCHAR(100),
   Phone_Num VARCHAR(15),
   Email VARCHAR(100)
);

-- Create the Staff table
CREATE TABLE Staff (
   Staff_ID NUMBER PRIMARY KEY NOT NULL,
   First_Name VARCHAR(50) NOT NULL,
   Surname VARCHAR(50) NOT NULL,
   Position VARCHAR(50),
   Phone_Num VARCHAR(15),
   Address VARCHAR(100),
   Email VARCHAR(100)
);

-- Create the Billing table
CREATE TABLE Billing (
   Bill_ID NUMBER PRIMARY KEY NOT NULL,
   Customer_ID NUMBER(5),
   Staff_ID NUMBER(5),
   Bill_Date DATE,
   FOREIGN KEY (Customer_ID) REFERENCES Customers(Customer_ID),
```

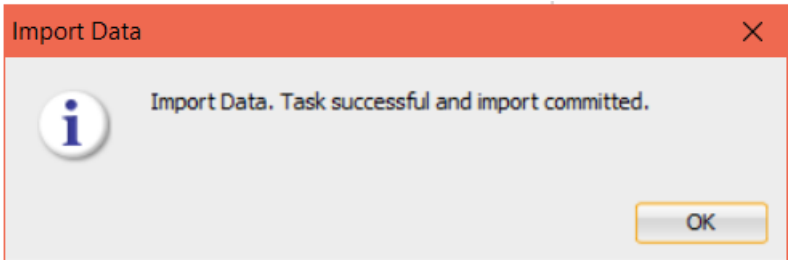
```
  FOREIGN KEY (Staff_ID) REFERENCES Staff(Staff_ID)
);

--  Create the Delivery_Items table
CREATE TABLE Delivery_Items (
  Delivery_Item_ID NUMBER PRIMARY KEY NOT NULL,
  Description VARCHAR(100),
  Staff_ID NUMBER(5),
  FOREIGN KEY (Staff_ID) REFERENCES Staff(Staff_ID)
);

-- Create the Driver table
CREATE TABLE Driver (
  Driver_ID NUMBER PRIMARY KEY NOT NULL,
  First_Name VARCHAR(50) NOT NULL,
  Surname VARCHAR(50) NOT NULL,
  Driver_Code VARCHAR(10),
  Phone_Num VARCHAR(15),
  Address VARCHAR(100)
);

-- Create the Vehicle table
CREATE TABLE Vehicle (
  VIN_Number VARCHAR(20) PRIMARY KEY NOT NULL,
  Vehicle_Type VARCHAR(50),
  Mileage NUMBER,
  Colour VARCHAR(20),
  Manufacturer VARCHAR2(50)
);
```

```
-- Create the Driver_Deliveries table
CREATE TABLE Driver_Deliveries (
   Driver_Delivery_ID NUMBER(5) PRIMARY KEY NOT NULL,
   VIN_Number VARCHAR(20),
   Driver_ID NUMBER(5),
   Delivery_Item_ID NUMBER(5),
   FOREIGN KEY (VIN_Number) REFERENCES Vehicle(VIN_Number),
   FOREIGN KEY (Driver_ID) REFERENCES Driver(Driver_ID),
   FOREIGN KEY (Delivery_Item_ID) REFERENCES Delivery_Items(Delivery_Item_ID)
);
```



Importing data





Customers Table

Query Result

SQL | All Rows Fetched: 15 in 0.01 seconds

	CUSTOMER_ID	FIRST_NAME	SURNAME	ADDRESS	PHONE_NUM	EMAIL
1	11011	Bob	Smith	18 Water rd	0877277521	bobs@isat.com
2	11012	Sam	Hendricks	22 Water rd	0863257857	shen@mcom.co.za
3	11013	Larry	Clark	101 Summer lane	0834567891	larc@mcom.co.za
4	11014	Jeff	Ionen	55 Mountain way	0612547895	jj@isat.co.za
5	11015	Andre	Kerk	5 Main rd	0827238521	akerk@mcal.co.za
6	11016	Wayne	Smith	13 Water rd	0877277522	ws@isat.com
7	11017	John	Hendricks	29 Water rd	0863257851	jhen@mcom.co.za
8	11018	Sally	Clark	111 Summer lane	0834567892	sallyc@mcom.co.za
9	11019	Bridget	Bitterhour	125 Mountain way	0612547896	bb@isat.co.za
10	11111	Nicole	Kerk	175 Main rd	0827238529	nk@mcal.co.za
11	11112	Catherine	Smith	19 Water rd	0877277523	cath@isat.com
12	11113	Mel	Hendricks	5 Water rd	0863257852	melh@mcom.co.za
13	11114	Lucy	Du Plessis	221 Summer lane	0834567892	Idup@mcom.co.za
14	11116	Josh	Maverick	155 Mountain way	0612547897	joshm@isat.co.za
15	11117	Stuart	Jones	35 Main rd	0827238521	sjones@mcal.co.za

Billing Table

Query Result

SQL | All Rows Fetched: 21 in 0.011 seconds

	BILL_ID	CUSTOMER_ID	STAFF_ID	BILL_DATE
1	800	11011	51011	06-SEP-22
2	801	11012	51013	07-SEP-22
3	802	11014	51015	10-NOV-22
4	803	11015	51012	09-DEC-22
5	804	11013	51014	09-DEC-22
6	805	11111	51011	06-SEP-22
7	806	11012	51013	07-SEP-22
8	807	11014	51015	10-NOV-22
9	808	11015	51012	09-DEC-22
10	809	11113	51018	09-DEC-22
11	810	11011	51011	06-SEP-22
12	811	11012	51013	07-SEP-22
13	812	11014	51016	10-NOV-22
14	813	11117	51012	09-DEC-22
15	814	11013	51014	09-DEC-22
16	815	11012	51111	06-SEP-22
17	816	11012	51019	07-SEP-22
18	817	11014	51015	10-NOV-22
19	818	11112	51012	09-DEC-22
20	819	11013	51014	09-DEC-22
21	820	11116	51019	09-DEC-22

Delivery_Items Table

Query Result

SQL | All Rows Fetched: 5 in 0.006 seconds

	DELIVERY_ITEM_ID	DESCRIPTION	STAFF_ID
1	71011	House relocation	51011
2	71012	Delivery of specialized consignments	51017
3	71013	Delivery of specialized consignments	51015
4	71014	Office relocation	51012
5	71015	Delivery of specialized consignments	51014

Driver Table

Query Result

SQL | All Rows Fetched: 5 in 0.008 seconds

	DRIVER_ID	FIRST_NAME	SURNAME	DRIVER_CODE	PHONE_NUM	ADDRESS
1	81011	Buthelezi	Marshall	C1	0725698547	18 Leopard creek
2	81012	Tina	Mtati	C	0636984178	12 Cape rd
3	81013	Jono	Mvondo	EC1	0712546895	10 Fifth Ave
4	81014	Sarah	Buthelezi	EC	0786954286	15 First Ave
5	81015	Toby	Davidson	B	0612548795	110 Summer lane

Driver_Deliveries Table

Script Output | Query Result

SQL | All Rows Fetched: 4 in 0.004 seconds

	DRIVER_DELIVERY_ID	VIN_NUMBER	DRIVER_ID	DELIVERY_ITEM_ID
1	91011	1ZA55858541	81011	71011
2	91012	1ZA35858543	81012	71013
3	91013	1ZA17851545	81011	71015
4	91015	1ZA15851545	81014	71012

Staff Table

Script Output | Query Result

SQL | All Rows Fetched: 10 in 0.007 seconds

	STAFF_ID	FIRST_NAME	SURNAME	POSITION	PHONE_NUM	ADDRESS	EMAIL
1	51011	Sally	Du Toit	Logistics	0825698547	18 Main rd	sdut@isat.com
2	51012	Mark	Wright	CRM	0836984178	12 Cape Way	mwright@isat.com
3	51013	Harry	Sheen	Logistics	0725648965	15 Water Street	hsheen@isat.com
4	51014	Jabu	Xolani	Logistics	0823116598	18 White Lane	jxo@isat.com
5	51015	Roberto	Henry	Packaging	0783521451	55 Cape Street	rhenry@isat.com
6	51016	Pat	Durant	Logistics	0825698542	1 Main rd	pd@isat.com
7	51017	Steve	Maritz	CRM	0836984173	2 Cape Way	sm@isat.com
8	51018	Maxwell	Dube	Logistics	0725648964	5 Water Street	max@isat.com
9	51019	Shane	Mane	Logistics	0823116595	8 White Lane	smane@isat.com
10	51111	Bob	Truth	Packaging	0783521456	35 Cape Street	btruth@isat.com

Vehicle Table

Script Output | Query Result

SQL | All Rows Fetched: 20 in 0.006 seconds

	VIN_NUMBER	VEHICLE_TYPE	MILEAGE	COLOUR	MANUFACTURER
1	1ZA55858541	Cutaway van chassis	115352	RED	MAN
2	1ZA51858542	Flatbed truck	315856	BLUE	ISUZU
3	1ZA35858543	Medium Standard Truck	789587	SILVER	MAN
4	1ZA15851545	Flatbed truck	555050	WHITE	TATA
5	1ZA35868540	Cutaway van chassis	79058	WHITE	ISUZU
6	1ZA65858541	Cutaway van chassis	215352	RED	MAN
7	1ZA61858542	Flatbed truck	215856	BLUE	ISUZU
8	1ZA65858543	Medium Standard Truck	889587	SILVER	MERC
9	1ZA65851545	Flatbed truck	155050	WHITE	MAN
10	1ZA65868540	Cutaway van chassis	19058	WHITE	ISUZU
11	1ZA75858541	Cutaway van chassis	315352	RED	MAN
12	1ZA71858542	Flatbed truck	115856	BLUE	ISUZU
13	1ZA75858543	Medium Standard Truck	989587	SILVER	MAN
14	1ZA17851545	Flatbed truck	755050	WHITE	TATA
15	1ZA75868540	Cutaway van chassis	29058	WHITE	ISUZU
16	1ZA85858541	Cutaway van chassis	515352	RED	MERC
17	1ZA81858542	Flatbed truck	715856	BLUE	ISUZU
18	1ZA85858543	Medium Standard Truck	789587	SILVER	MAN
19	1ZA85851545	Flatbed truck	955050	WHITE	TATA
20	1ZA85868540	Cutaway van chassis	39058	WHITE	MERC

Question 3.1

Create user c##John Identified by Johnch2024;

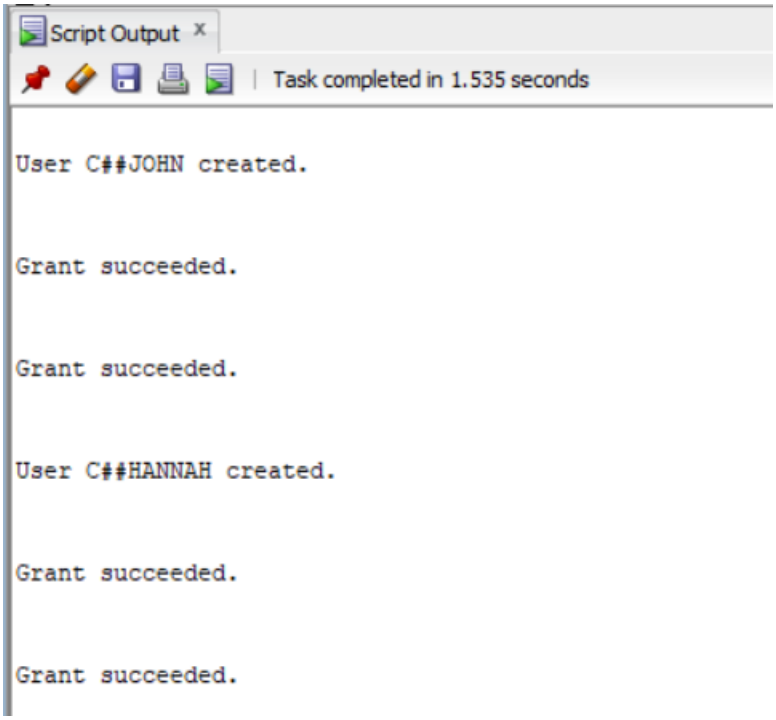
grant create session to c##John;

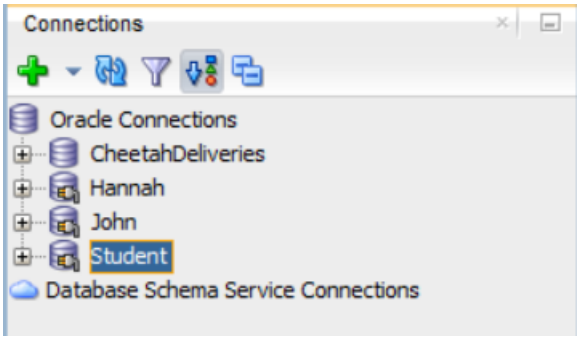
grant SELECT ANY TABLE to c##John;

Create user c##Hannah Identified by Hannahch2024;

grant create session to c##Hannah;

grant INSERT ANY TABLE to c##Hannah;





Question 3.2

It is very necessary in an Oracle database environment to strengthen security, integrity, and efficiency. Separation of duties (SoD) is one way through which an organization can prevent unauthorized acts of fraud by distributing responsibilities. Examples include the separation of duties between database designing and data entry or management of users, where not one person can have full control over everything in the database. (Awati, 2022)

Furthermore, SoD maintains integrity in data while ensuring operational efficiency through the allocation of specifically intended tasks to specifically permitted users, hence avoiding errors, which infuses database management with stability. Adherence to the principles of the SoD justifies regulatory compliance like that with the Sarbanes-Oxley Act. (Awati, 2022), thus making it possible to reduce litigations while fostering relations by ensuring that data handling is safe and accountable.

Question 4.1

```
DECLARE
v_driverName varchar(50);
v_driverCode varchar(50);
v_vinNum varchar(100);
v_mileage NUMBER;
begin
for output in(SELECT d.FIRST_NAME || ',' || d.SURNAME AS FullName, d.Driver_Code,v.VIN_NUMBER, v.Mileage
from driver_deliveries dD
JOIN
  Driver d ON dD.Driver_ID = d.Driver_ID
JOIN
  Vehicle v ON dD.VIN_NUMBER = v.VIN_NUMBER
WHERE
  v.Mileage < 80000)
LOOP
v_driverName := output.FullName;
v_driverCode := output.Driver_Code;
v_vinNum := output.VIN_NUMBER;
v_mileage := output.Mileage;

DBMS_OUTPUT.PUT_LINE('DRIVER: ' || v_driverName);
DBMS_OUTPUT.PUT_LINE('CODE: ' || v_driverCode);
DBMS_OUTPUT.PUT_LINE('VIN NUMBER: ' || v_vinNum);
DBMS_OUTPUT.PUT_LINE('MILEAGE: ' || v_mileage);
DBMS_OUTPUT.PUT_LINE('----------------------------------------');
end loop;
end;
```

```
PL/SQL procedure successfully completed.

DRIVER: Jono,Mvuyisi
CODE: EC1
VIN NUMBER: 1ZA35868540
MILEAGE: 79058
----------------------------------------


PL/SQL procedure successfully completed.
```

Question 4.2

These two models approach database management from different angles: the flat file model and the relational model. The flat file model, in use currently by Cheetah Deliveries, stores data in a single, unstructured file. This leads to large redundancies and maintenance issues regarding data integrity. Relationships between elements of data are not explicitly enforced within the system, and performance issues with large volumes of data are a sure concern. Besides, security features extend only as far as the permissions on the plain files.

On the other hand, the relational model stores data in structured tables, with defined relationships to prevent redundancy and make sure the data integrity is observed. It can support any level of query and reporting due to SQL; hence, management and analysis of data become easy. This therefore means that a company like Cheetah Deliveries will have better compatibility to handle large volumes of data, high-level queries, and tight security, which improves the efficiency of operation and handling of data.

Changing from a flat file system to a relational database management system will provide Cheetah Deliveries with a more efficient, scalable, and secure solution. This transition will enhance their ability to manage data, improve operational efficiency, and support better decision-making, ultimately addressing the key challenges they face in their dynamic and competitive courier service industry.

Question 5.1

```
DECLARE
   v_staff_id Staff.STAFF_ID%TYPE;
   v_first_name Staff.FIRST_NAME%TYPE;
   v_surname Staff.SURNAME%TYPE;
   v_deliveries_processed NUMBER;
BEGIN

   SELECT s.STAFF_ID, s.FIRST_NAME, s.SURNAME, COUNT(dd.DRIVER_DELIVERY_ID)
   INTO v_staff_id, v_first_name, v_surname, v_deliveries_processed
   FROM Staff s
   JOIN Delivery_Items di ON s.STAFF_ID = di.STAFF_ID
   JOIN Driver_Deliveries dd ON di.DELIVERY_ITEMS = dd.DELIVERY_ITEM_ID
   GROUP BY s.STAFF_ID, s.FIRST_NAME, s.SURNAME
   ORDER BY COUNT(dd.DRIVER_DELIVERY_ID) DESC
   FETCH FIRST ROW ONLY;

   DBMS_OUTPUT.PUT_LINE('------------------------------');
   DBMS_OUTPUT.PUT_LINE('STAFF ID: ' || v_staff_id);
   DBMS_OUTPUT.PUT_LINE('FIRST NAME: ' || v_first_name);
   DBMS_OUTPUT.PUT_LINE('SURNAME: ' || v_surname);
   DBMS_OUTPUT.PUT_LINE('DELIVERIES PROCESSED: ' || v_deliveries_processed);
   DBMS_OUTPUT.PUT_LINE('------------------------------');
END;
/
```

```
------------------------------
STAFF ID: 51014
FIRST NAME: Jabu
SURNAME: Xolani
DELIVERIES PROCESSED: 2
------------------------------
```

Question 5.2

In PL/SQL programming, a block or a subprogram is divided into three sections: the declarative section, the executable section, and the exception-handling section. Each of these parts plays a certain role in ensuring that code runs effectively, efficiently, and robustly.

Declarative Section: In my code, the declarative section starts with the DECLARE keyword, and within this declarative section, the declaration of variables like v_staff_id, v_first_name, v_surname, and v_deliveries_processed has been made. These variables are declared with types derived from the columns in the Staff table or as NUMBER for counting purposes.

Declarative part: In a subprogram, unlike declaring variables, the declarative section does not begin with DECLARE (Great Learning, n.d.) but plays a similar role of declaring types, constants, and variables local to the subprogram that exist for the duration of the execution of the subprogram.

Executable Section: This is where the core logic of the block is executed. For example, the code in the BEGIN section draws on a SELECT statement containing information about staff; their IDs, first names, surnames, and numbers of deliveries processed. The information is then fed into the variables declared earlier using the INTO clause which are then fed to DBMS_OUTPUT.PUT_LINE statements to print out the contents. Similarly, the executable section of the procedures contains statements that execute the intended action. Great Learning, n.d.

Exception-handling section: My code does not include an exception-handling section. If there is a runtime error, like no data found by the SELECT statement, that type of error is not managed within the block. For a complete procedure, an EXCEPTION section would catch and manage those types of errors.

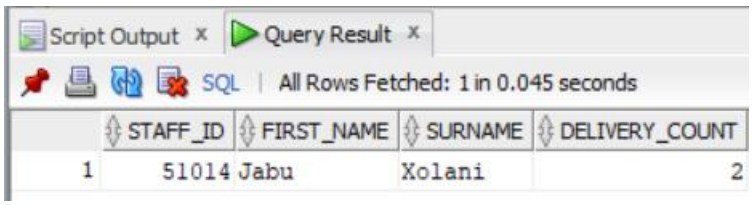
Question 5.3.1

In SQL, a view is a sort of virtual table based on the result set of a query. The view itself does not store the data; it retrieves the data from one or more underlying tables. In querying a view, the result would always be in real time from the database, deriving its basis from the tables and conditions specified in the query. For instance, the following view, `Staff_Delivery_Count`, is based on aggregated data from the `Staff`, `Delivery_Items`, and `Driver_Deliveries` tables to present the count of delivery for each staff member. As such, users can request data or reports in simple commands with greatly reduced chances of errors and efficient data management. (Gupta, 2019)

Question 5.3.2

```
-- Create or replace a view named 'Staff_Delivery_Count'

CREATE OR REPLACE VIEW Staff_Delivery_Count AS

SELECT

  s.STAFF_ID,          -- Selects the Staff ID from the Staff table

  s.FIRST_NAME,        -- Selects the First Name of the staff from the Staff table

  s.SURNAME,           -- Selects the Surname of the staff from the Staff table

  COUNT(dd.DRIVER_DELIVERY_ID) AS DELIVERY_COUNT  -- Counts the number of deliveries associated with the staff

FROM

  Staff s  -- The Staff table is aliased as 's'

JOIN

  Delivery_Items di ON s.STAFF_ID = di.STAFF_ID  -- Joins the Staff table with the Delivery_Items table on Staff ID

JOIN

  Driver_Deliveries dd ON di.DELIVERY_ITEM = dd.DELIVERY_ITEM_ID -- Joins the Delivery_Items table with the Driver_Deliveries table on Delivery Item ID

GROUP BY

  s.STAFF_ID, s.FIRST_NAME, s.SURNAME;  -- Groups the result by Staff ID, First Name, and Surname


-- Declare a PL/SQL block

DECLARE

  v_staff_id Staff.STAFF_ID%TYPE;        -- Variable to store the Staff ID

  v_first_name Staff.FIRST_NAME%TYPE;    -- Variable to store the First Name of the staff

  v_surname Staff.SURNAME%TYPE;          -- Variable to store the Surname of the staff

  v_deliveries_processed NUMBER;         -- Variable to store the number of deliveries processed by the staff

BEGIN

  -- Select the Staff ID, First Name, Surname, and Delivery Count from the view 'Staff_Delivery_Count'

  -- Order the results by Delivery Count in descending order and fetch only the first row (staff with the highest delivery count)
```

```
    SELECT STAFF_ID, FIRST_NAME, SURNAME, DELIVERY_COUNT
    INTO v_staff_id, v_first_name, v_surname, v_deliveries_processed
    FROM Staff_Delivery_Count
    ORDER BY DELIVERY_COUNT DESC
    FETCH FIRST ROW ONLY;

    -- Output the details of the staff member with the highest delivery count
    DBMS_OUTPUT.PUT_LINE('');  -- Prints a blank line
    DBMS_OUTPUT.PUT_LINE('STAFF ID: ' || v_staff_id);  -- Prints the Staff ID
    DBMS_OUTPUT.PUT_LINE('FIRST NAME: ' || v_first_name);  -- Prints the First Name
    DBMS_OUTPUT.PUT_LINE('SURNAME: ' || v_surname);  -- Prints the Surname
    DBMS_OUTPUT.PUT_LINE('DELIVERIES PROCESSED: ' || v_deliveries_processed);  -- Prints the number of deliveries processed
    DBMS_OUTPUT.PUT_LINE('');  -- Prints a blank line
END;
```



Question 6.1.1

Question A: Implicit Cursors

Implicit cursors are automatically created for the user whenever a SELECT, INSERT, UPDATE, or DELETE statement is executed. Implicit cursors are extremely competent for handling simple queries and updates because the user does not need to declare and manage these cursors explicitly. Implicit cursors have some useful attributes: `SQL%FOUND` informs you whether the last performed operation returned any rows, `SQL%NOTFOUND` informs you whether no rows were found, and `SQL%ROWCOUNT` tells how many rows an operation affected. This makes it possible to perform quick checks regarding the status of the last SQL operation performed without explicitly taking care of cursor actions. (geeksforgeeks, 2020)

Question 6.1.1

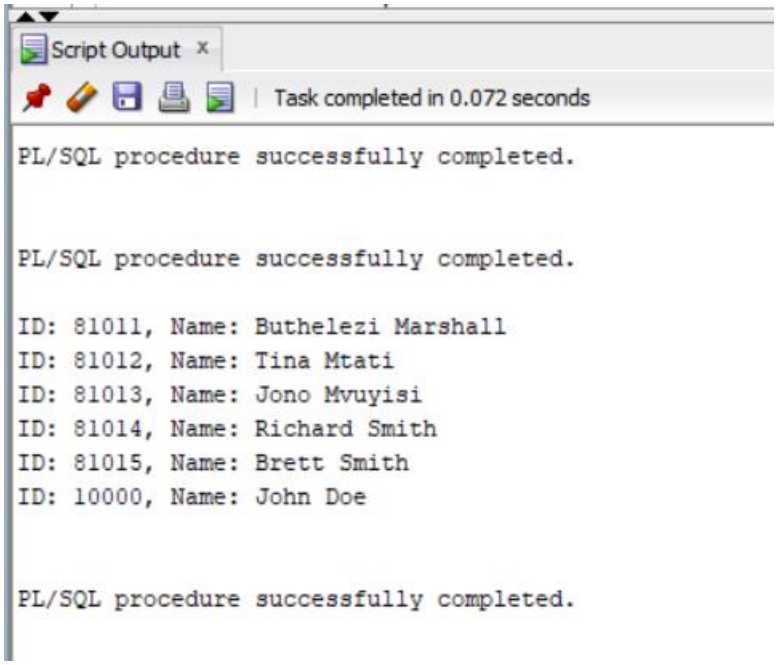
Question B: Explicit Cursors

Explicit cursors are declared and maintained by the user. Explicit cursors provide more control when fetching multiple rows that are returned by a query. This, in essence, means that an explicit cursor must be declared, opened, fetched from, and closed by the user; hence, explicit cursors are capable of handling complex data processing scenarios. The key explicit cursor attributes are: `cur_name%FOUND' tells whether a row was returned by the last fetch; `cur_name%NOTFOUND' signals that no row was fetched, and `cur_name%ROWCOUNT' reports the number of rows that have been fetched so far. This level of control is highly useful when dealing with large volumes of data or when some kind of specific row-by-row processing becomes necessary. (geeksforgeeks, 2020)

Question 6.1.2

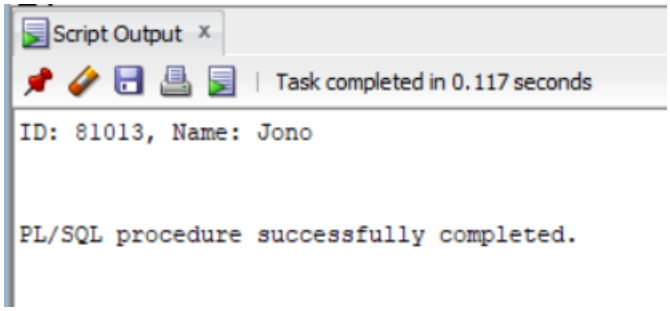
Explicit:

```
-- Declare the explicit cursor
DECLARE
  CURSOR driver_cursor IS
    SELECT DRIVER_ID, FIRST_NAME, SURNAME, DRIVER_CODE, PHONE_NUM, ADDRESS
    FROM Driver;

  -- Define a record to hold each row
  driver_record driver_cursor%ROWTYPE;
BEGIN
  -- Open the cursor
  OPEN driver_cursor;

  -- Fetch rows from the cursor one by one
  LOOP
    FETCH driver_cursor INTO driver_record;
    EXIT WHEN driver_cursor%NOTFOUND;

    -- Process each row (for demonstration, we'll just output the data)
    DBMS_OUTPUT.PUT_LINE('ID: ' || driver_record.DRIVER_ID || ', Name: ' || driver_record.FIRST_NAME || ' ' || driver_record.SURNAME);
  END LOOP;

  -- Close the cursor
  CLOSE driver_cursor;
END;
/
```


Script Output
Task completed in 0.072 seconds
PL/SQL procedure successfully completed.
PL/SQL procedure successfully completed.
ID: 81011, Name: Buthelezi Marshall
ID: 81012, Name: Tina Mtati
ID: 81013, Name: Jono Mvuyisi
ID: 81014, Name: Richard Smith
ID: 81015, Name: Brett Smith
ID: 10000, Name: John Doe
PL/SQL procedure successfully completed.

Implicit:

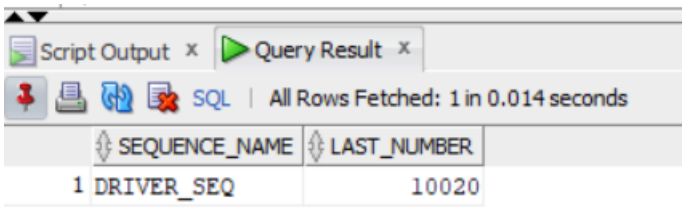
```
-- Example using an implicit cursor for a SELECT statement
DECLARE
  v_driver_id Driver.DRIVER_ID%TYPE;
  v_first_name Driver.FIRST_NAME%TYPE;
BEGIN
  -- Fetch data into variables using a SELECT INTO statement
  SELECT DRIVER_ID, FIRST_NAME
  INTO v_driver_id, v_first_name
  FROM Driver
  WHERE DRIVER_CODE = 'EC1';

  -- Output the data
  DBMS_OUTPUT.PUT_LINE('ID: ' || v_driver_id || ', Name: ' || v_first_name);
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('No data found for the given DRIVER_CODE.');
END;
/
```



Question 6.2

```
BEGIN
  EXECUTE IMMEDIATE 'CREATE SEQUENCE driver_seq
    START WITH 10000
    INCREMENT BY 1
    MAXVALUE 99999
    CYCLE
    CACHE 20';
EXCEPTION
  WHEN OTHERS THEN
    IF SQLCODE = -955 THEN
      -- Sequence already exists
      NULL;
    ELSE
      RAISE;
    END IF;
END;
/


BEGIN
  INSERT INTO Driver (DRIVER_ID, FIRST_NAME, SURNAME, DRIVER_CODE, PHONE_NUM, ADDRESS)
  VALUES (driver_seq.NEXTVAL, 'John', 'Doe', 'EC2', '0831234567', '20 Park Avenue');
    COMMIT;
EXCEPTION
  WHEN OTHERS THEN
    ROLLBACK;
    RAISE;
END;
/
SELECT sequence_name, last_number
```

```
FROM user_sequences
WHERE sequence_name = 'DRIVER_SEQ';
/
```



References

Awati, R. (2022). *What is Segregation of Duties (SoD)?* [online] WhatIs.com. Available at: https://www.techtarget.com/whatis/definition/segregation-of-duties-SoD.

geeksforgeeks (2020). *Difference between Implicit and Explicit Cursors*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/difference-between-implicit-and-explicit-cursors/.

Great Learning (n.d.). *Parts of PL/SQL Subprogram*. [online] Great Learning. Available at: https://www.mygreatlearning.com/plsql/tutorials/parts-of-pl-sql-subprogram.

Gupta, R. (2019). *SQL View - A complete introduction and walk-through*. [online] SQL Shack - articles about database auditing, server performance, data recovery, and more. Available at: https://www.sqlshack.com/sql-view-a-complete-introduction-and-walk-through/.