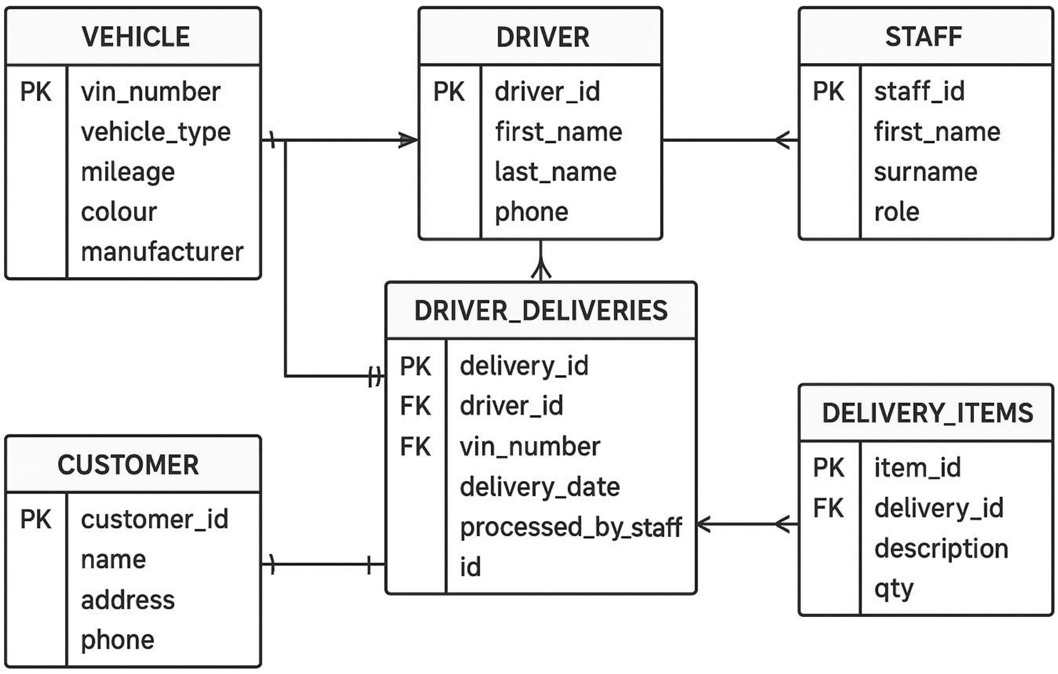
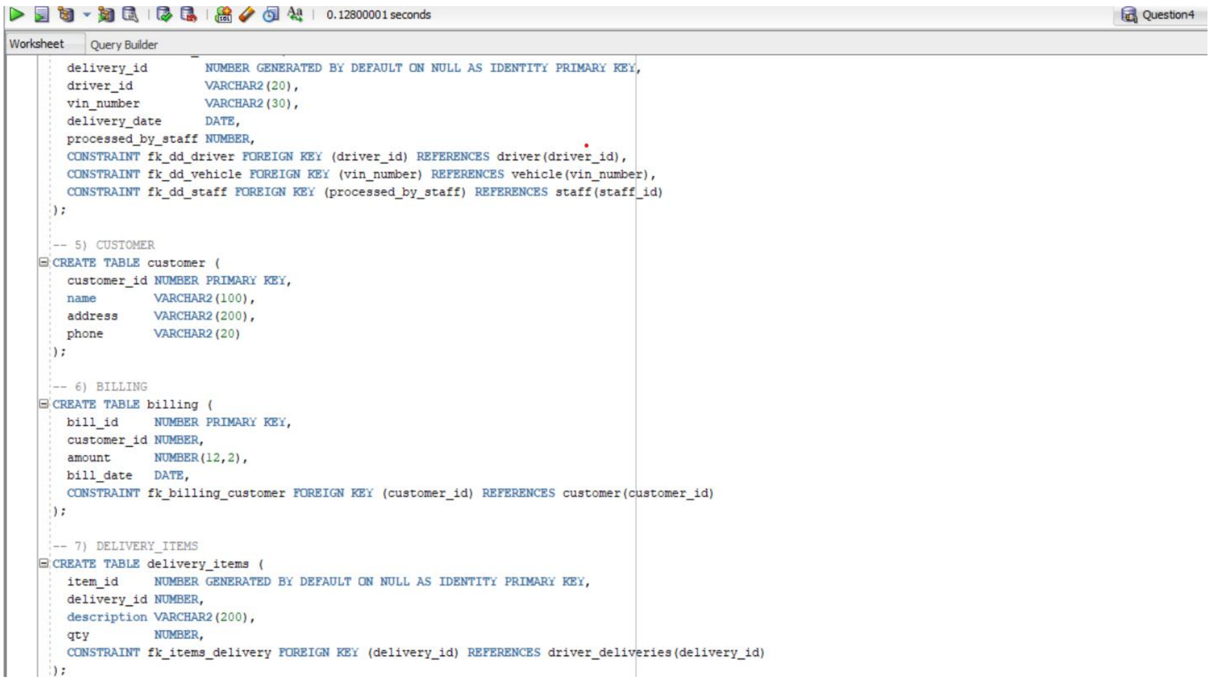
Isaac Phiri
ST10074970
INSY7

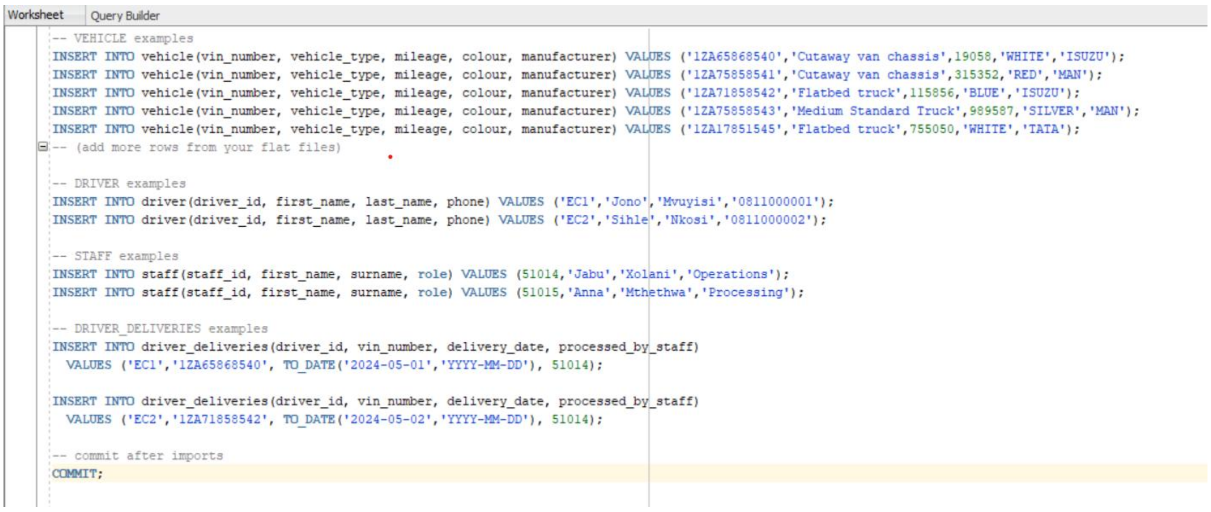
Question 1



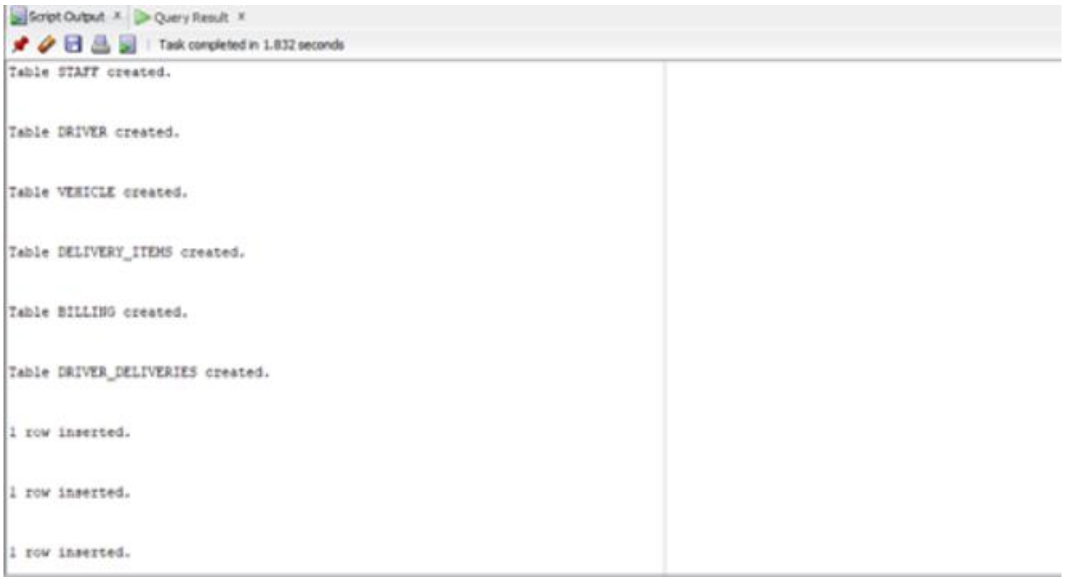
Qiestion 2



```
  delivery_id        NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY,
  driver_id          VARCHAR2(20),
  vin_number         VARCHAR2(30),
  delivery_date      DATE,
  processed_by_staff NUMBER,
  CONSTRAINT fk_dd_driver FOREIGN KEY (driver_id) REFERENCES driver(driver_id),
  CONSTRAINT fk_dd_vehicle FOREIGN KEY (vin_number) REFERENCES vehicle(vin_number),
  CONSTRAINT fk_dd_staff FOREIGN KEY (processed_by_staff) REFERENCES staff(staff_id)
);

-- 5) CUSTOMER
CREATE TABLE customer (
  customer_id NUMBER PRIMARY KEY,
  name        VARCHAR2(100),
  address     VARCHAR2(200),
  phone       VARCHAR2(20)
);

-- 6) BILLING
CREATE TABLE billing (
  bill_id     NUMBER PRIMARY KEY,
  customer_id NUMBER,
  amount      NUMBER(12,2),
  bill_date   DATE,
  CONSTRAINT fk_billing_customer FOREIGN KEY (customer_id) REFERENCES customer(customer_id)
);

-- 7) DELIVERY_ITEMS
CREATE TABLE delivery_items (
  item_id     NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY,
  delivery_id NUMBER,
  description VARCHAR2(200),
  qty         NUMBER,
  CONSTRAINT fk_items_delivery FOREIGN KEY (delivery_id) REFERENCES driver_deliveries(delivery_id)
);
```



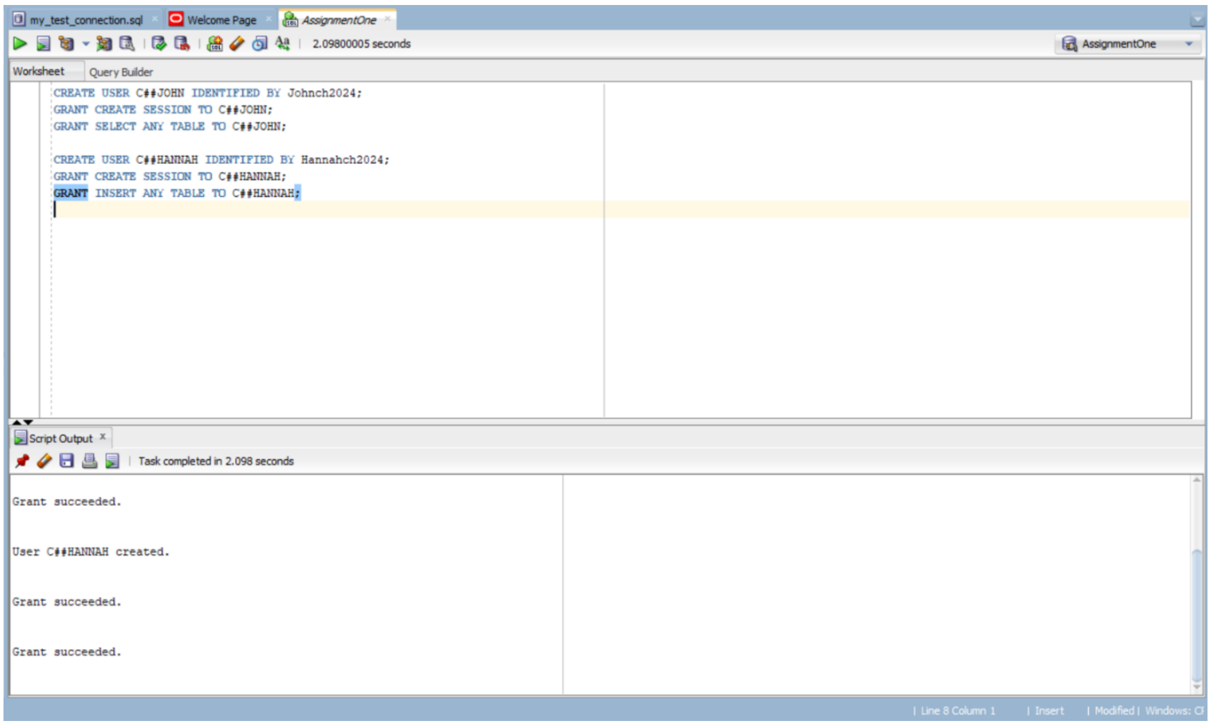
```
-- VEHICLE examples
INSERT INTO vehicle(vin_number, vehicle_type, mileage, colour, manufacturer) VALUES ('1ZA65868540','Cutaway van chassis',19058,'WHITE','ISUZU');
INSERT INTO vehicle(vin_number, vehicle_type, mileage, colour, manufacturer) VALUES ('1ZA75858541','Cutaway van chassis',315352,'RED','MAN');
INSERT INTO vehicle(vin_number, vehicle_type, mileage, colour, manufacturer) VALUES ('1ZA71858542','Flatbed truck',115856,'BLUE','ISUZU');
INSERT INTO vehicle(vin_number, vehicle_type, mileage, colour, manufacturer) VALUES ('1ZA75858543','Medium Standard Truck',989587,'SILVER','MAN');
INSERT INTO vehicle(vin_number, vehicle_type, mileage, colour, manufacturer) VALUES ('1ZA17851545','Flatbed truck',755050,'WHITE','TATA');
-- (add more rows from your flat files)

-- DRIVER examples
INSERT INTO driver(driver_id, first_name, last_name, phone) VALUES ('EC1','Jono','Mvuyisi','0811000001');
INSERT INTO driver(driver_id, first_name, last_name, phone) VALUES ('EC2','Sihle','Nkosi','0811000002');

-- STAFF examples
INSERT INTO staff(staff_id, first_name, surname, role) VALUES (51014,'Jabu','Xolani','Operations');
INSERT INTO staff(staff_id, first_name, surname, role) VALUES (51015,'Anna','Mthethwa','Processing');

-- DRIVER_DELIVERIES examples
INSERT INTO driver_deliveries(driver_id, vin_number, delivery_date, processed_by_staff)
  VALUES ('EC1','1ZA65868540', TO_DATE('2024-05-01','YYYY-MM-DD'), 51014);

INSERT INTO driver_deliveries(driver_id, vin_number, delivery_date, processed_by_staff)
  VALUES ('EC2','1ZA71858542', TO_DATE('2024-05-02','YYYY-MM-DD'), 51014);

-- commit after imports
COMMIT;
```



```
Table STAFF created.


Table DRIVER created.


Table VEHICLE created.


Table DELIVERY_ITEMS created.


Table BILLING created.


Table DRIVER_DELIVERIES created.


1 row inserted.


1 row inserted.


1 row inserted.
```

Question 3

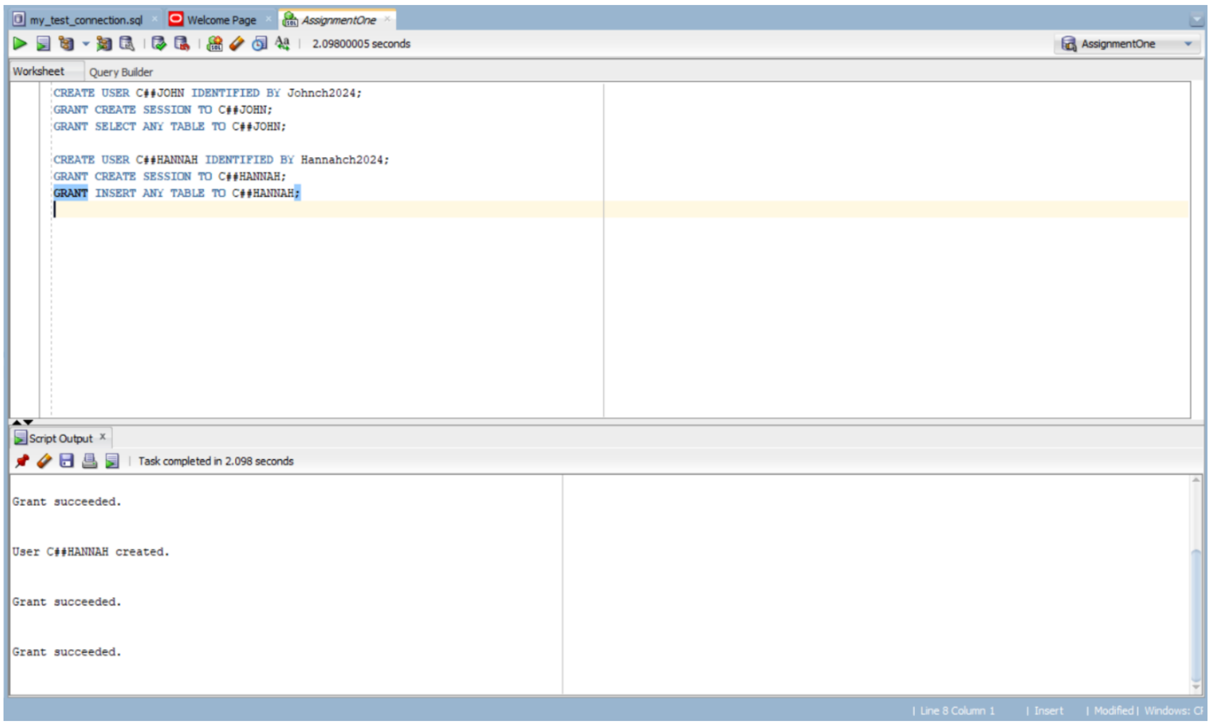
3.1)



```
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;
```

Task completed in 2.098 seconds

```
Grant succeeded.


User C##HANNAH created.


Grant succeeded.


Grant succeeded.
```



```
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;
```

Task completed in 2.098 seconds

```
Grant succeeded.


User C##HANNAH created.


Grant succeeded.


Grant succeeded.
```

3.2)

John has read-only access so he can query data but cant change it

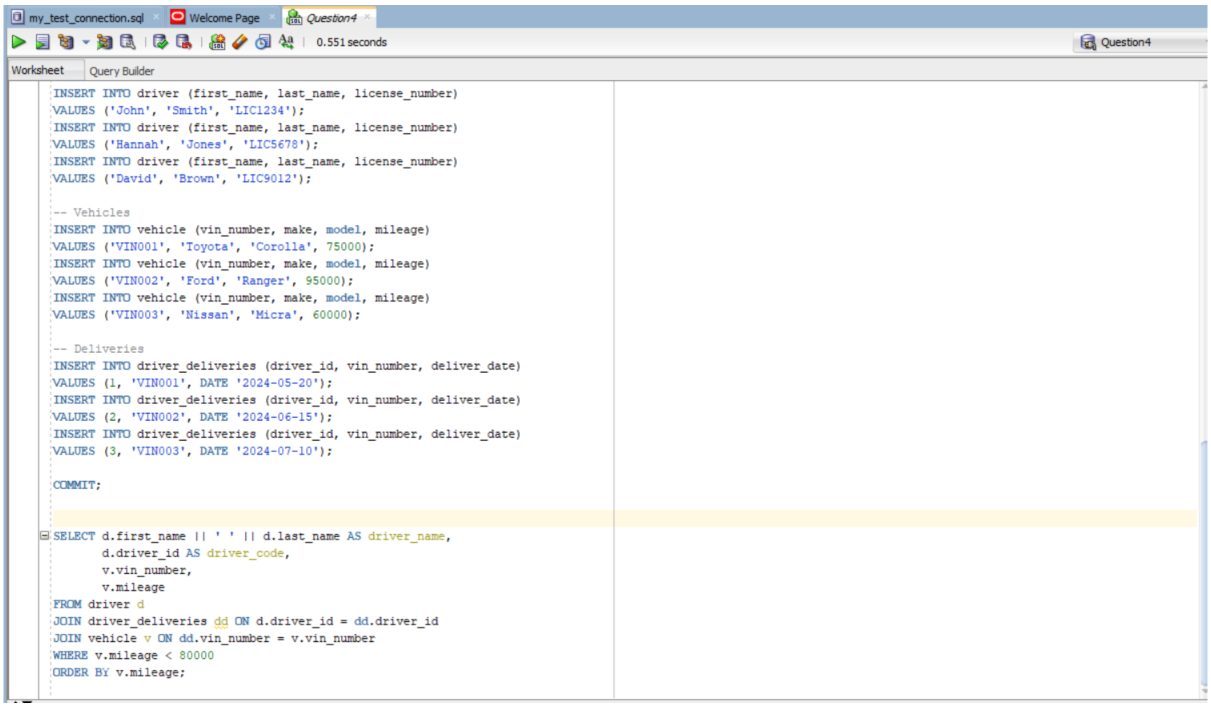
Hannah has insert privilege so that she is able to add records but cant in this configuration

Separation of duties reduces fraud risk while still preserving data integrity and still pushing accountability.

Question 4

Question 4

4.1)



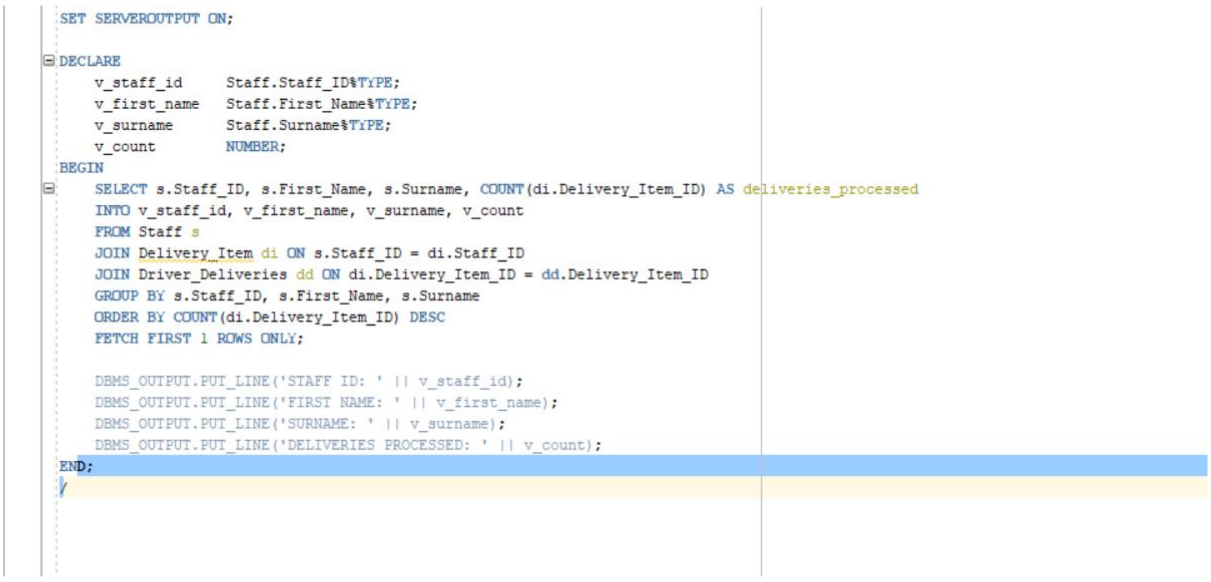
```sql
INSERT INTO driver (first_name, last_name, license_number)
VALUES ('John', 'Smith', 'LIC1234');
INSERT INTO driver (first_name, last_name, license_number)
VALUES ('Hannah', 'Jones', 'LIC5678');
INSERT INTO driver (first_name, last_name, license_number)
VALUES ('David', 'Brown', 'LIC9012');

-- Vehicles
INSERT INTO vehicle (vin_number, make, model, mileage)
VALUES ('VIN001', 'Toyota', 'Corolla', 75000);
INSERT INTO vehicle (vin_number, make, model, mileage)
VALUES ('VIN002', 'Ford', 'Ranger', 95000);
INSERT INTO vehicle (vin_number, make, model, mileage)
VALUES ('VIN003', 'Nissan', 'Micra', 60000);

-- Deliveries
INSERT INTO driver_deliveries (driver_id, vin_number, deliver_date)
VALUES (1, 'VIN001', DATE '2024-05-20');
INSERT INTO driver_deliveries (driver_id, vin_number, deliver_date)
VALUES (2, 'VIN002', DATE '2024-06-15');
INSERT INTO driver_deliveries (driver_id, vin_number, deliver_date)
VALUES (3, 'VIN003', DATE '2024-07-10');

COMMIT;


SELECT d.first_name || ' ' || d.last_name AS driver_name,
       d.driver_id AS driver_code,
       v.vin_number,
       v.mileage
FROM driver d
JOIN driver_deliveries dd ON d.driver_id = dd.driver_id
JOIN vehicle v ON dd.vin_number = v.vin_number
WHERE v.mileage < 80000
ORDER BY v.mileage;
```

4.2) • Flat files: single-table text files redundant data, duplication leads to update anomalies.

• Relational model: normalized tables, keys, and constraints help make sure data integrity and eliminate duplication, allows for powerful SQL joins, indexes, and transaction control.

- For Cheetah Deliveries, relational design supports multiple user transactions like deliveries, billing, easy reporting of driver/vehicle reports and enforcement of business rules via constraints.

Question 5



```
SET SERVEROUTPUT ON;

DECLARE
    v_staff_id      Staff.Staff_ID%TYPE;
    v_first_name    Staff.First_Name%TYPE;
    v_surname       Staff.Surname%TYPE;
    v_count         NUMBER;
BEGIN
    SELECT s.Staff_ID, s.First_Name, s.Surname, COUNT(di.Delivery_Item_ID) AS deliveries_processed
    INTO v_staff_id, v_first_name, v_surname, v_count
    FROM Staff s
    JOIN Delivery_Item di ON s.Staff_ID = di.Staff_ID
    JOIN Driver_Deliveries dd ON di.Delivery_Item_ID = dd.Delivery_Item_ID
    GROUP BY s.Staff_ID, s.First_Name, s.Surname
    ORDER BY COUNT(di.Delivery_Item_ID) DESC
    FETCH FIRST 1 ROWS ONLY;

    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_count);
END;
/
```

5.2 PL/SQL blocks have three sections:

1. Declaration section → variables, constants, cursors, data types.

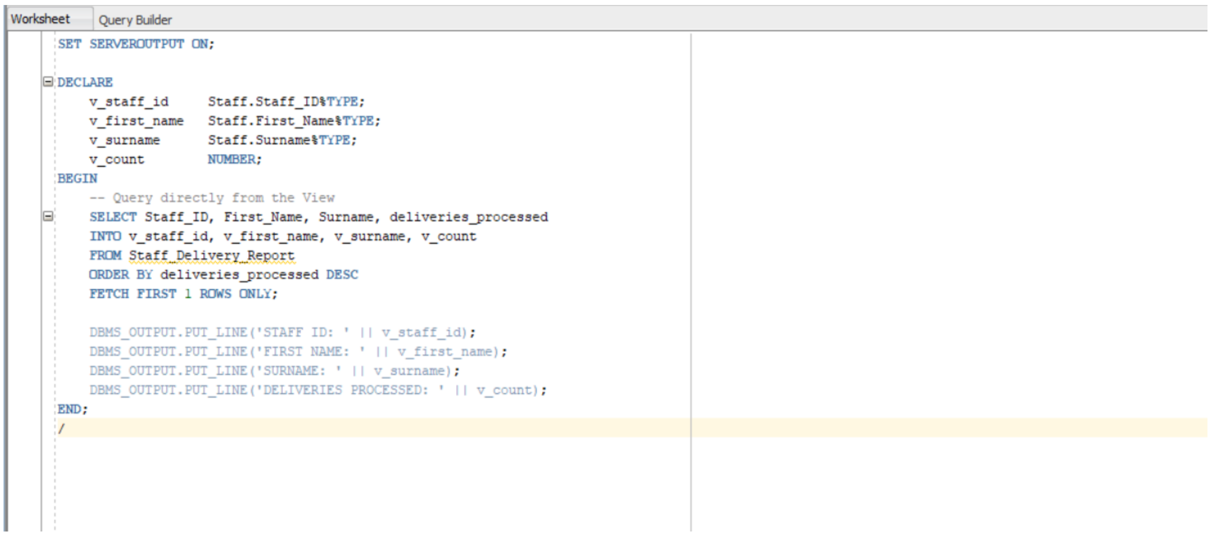
(In Q.5.1: we declared v_staff_id, v_first_name, v_surname, v_count).

2. Execution section → contains the main SQL/PLSQL logic.

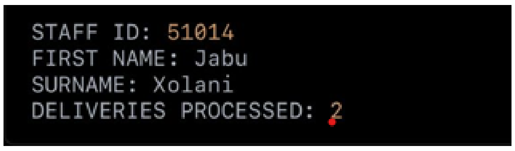
(In Q.5.1: the SELECT INTO … query and the DBMS_OUTPUT.PUT_LINE commands).

3. Exception-handling section → handles errors gracefully.

5.3.1
A view is a stored queary that also acts as if it was a virtual table.
5.3.2



```
SET SERVEROUTPUT ON;

DECLARE
    v_staff_id      Staff.Staff_ID%TYPE;
    v_first_name    Staff.First_Name%TYPE;
    v_surname       Staff.Surname%TYPE;
    v_count         NUMBER;
BEGIN
    -- Query directly from the View
    SELECT Staff_ID, First_Name, Surname, deliveries_processed
    INTO v_staff_id, v_first_name, v_surname, v_count
    FROM Staff_Delivery_Report
    ORDER BY deliveries_processed DESC
    FETCH FIRST 1 ROWS ONLY;

    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_count);
END;
/
```



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

Question 6 .1

Implicit cursor attributes

Is used when oracle does a sql statement automatically, it also proves information about what the last executed statement is.
FOUND → Returns TRUE if at least one row was affected.

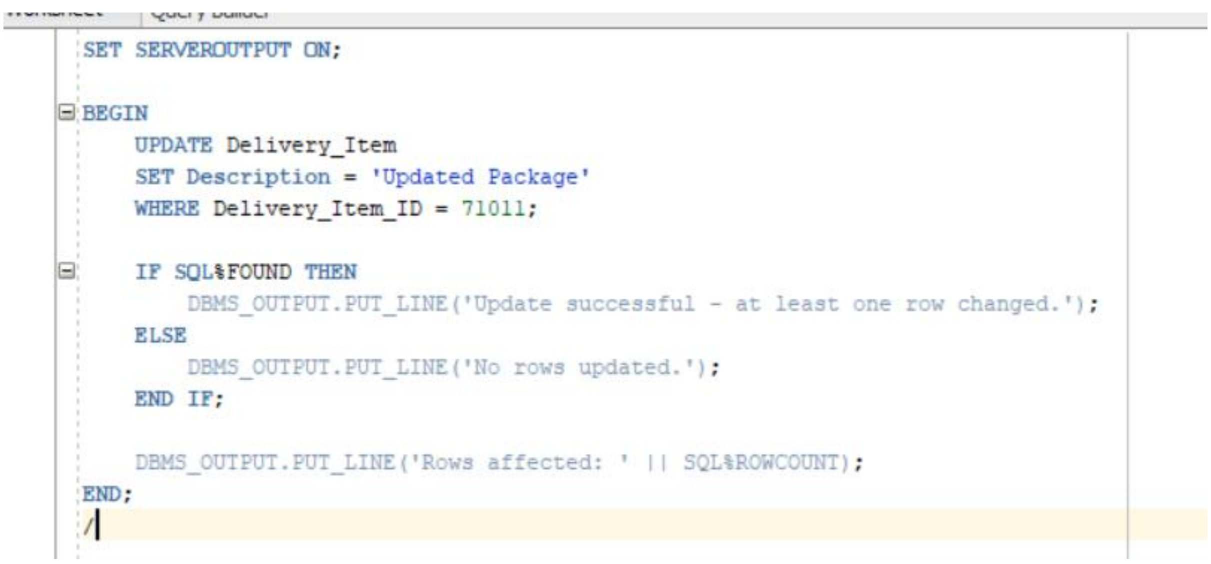
- %NOTFOUND → Returns TRUE if no rows were affected.
- %ROWCOUNT → Returns number of rows affected.
- %ISOPEN → Always FALSE for implicit cursors.

Explicit curor attributes

Declared by the user for more control when getting more rows.

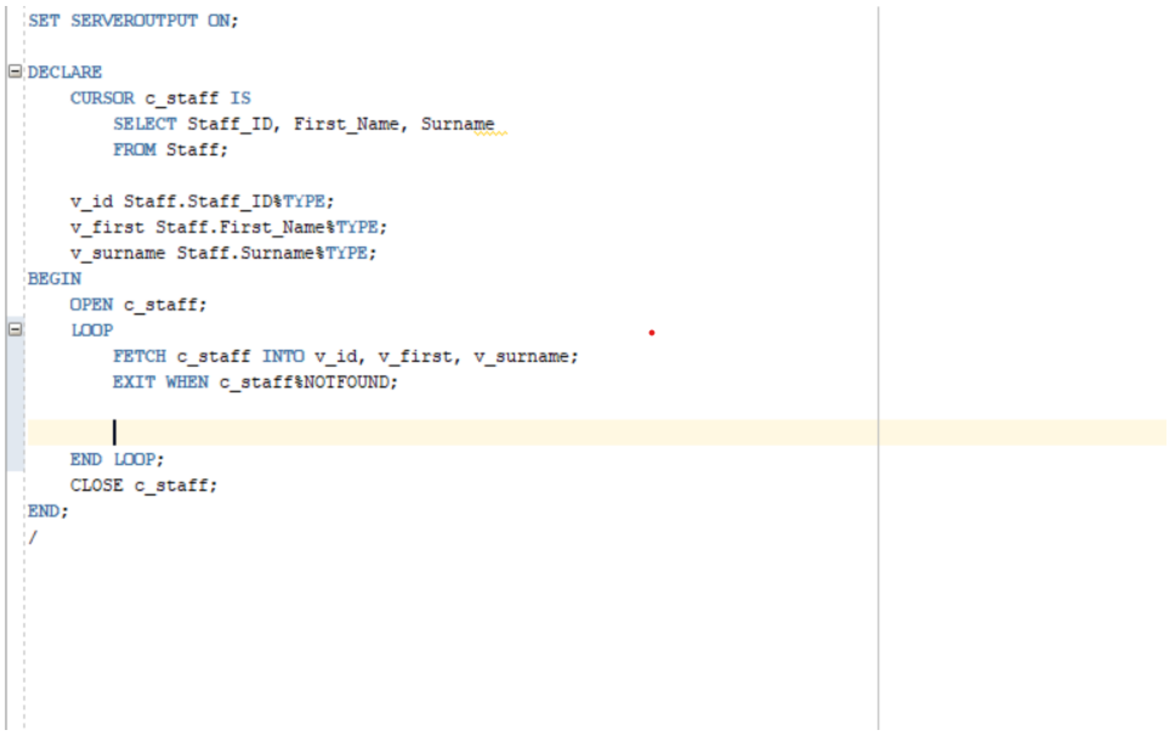
Useful when working with multiple row queries like iterating through all deliveries

A



```
SET SERVEROUTPUT ON;

BEGIN
    UPDATE Delivery_Item
    SET Description = 'Updated Package'
    WHERE Delivery_Item_ID = 71011;

    IF SQL%FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Update successful - at least one row changed.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('No rows updated.');
    END IF;

    DBMS_OUTPUT.PUT_LINE('Rows affected: ' || SQL%ROWCOUNT);
END;
/
```

B



```
SET SERVEROUTPUT ON;

DECLARE
    CURSOR c_staff IS
        SELECT Staff_ID, First_Name, Surname
        FROM Staff;

    v_id Staff.Staff_ID%TYPE;
    v_first Staff.First_Name%TYPE;
    v_surname Staff.Surname%TYPE;
BEGIN
    OPEN c_staff;
    LOOP
        FETCH c_staff INTO v_id, v_first, v_surname;
        EXIT WHEN c_staff%NOTFOUND;

    END LOOP;
    CLOSE c_staff;
END;
/
```

6.2

A sequence is a database object that generates unique numbers automatically. It is useful for generating primary keys (like Staff_ID, Delivery_Item_ID, etc.) without manual input.