SECTION B **[MARKS: 70]**

APPLICATION SECTION: ANSWER ALL QUESTIONS.

Instructions: This section can be completed without the need for an IDE/Oracle. Emphasis is placed on the logic of the solutions provided

You have recently joined the Database team at XYZ Construction company as a Database developer. The company specialises in developing townhouses. At present, the database is small and only includes information about customers, townhouses, materials, and purchase agreements. The following set of relations have been set up. The relationships between the tables must be derived from the data in each of the tables. The tables and the information are as follows:

- CUSTOMER (cust_id, cust_fname, cust_sname, cust_address, cust_contact)
- TOWNHOUSE (house_num, build_price, bedrooms, bathrooms)
- MATERIALS (material_id, material_name)
- PURCHASE_AGREEMENT (purchase_num, purchase_date, purchase_amt, house_num, cust_id, material_id)

Sample Data is shown below:

CUSTOMER

CUST_ID	CUST_FNAME	CUST_SNAME	CUST_ADDRESS	CUST_CONTACT
C115	Philip	Willis	3 Main Road	0821253659
C116	Jeff	Watson	13 Cape Road	0769658547
C117	Sam	Smith	3 Mountain Road	0863256574
C118	Alfred	Hanson	8 Circle Road	0762356587
C119	Wayne	Bitterhout	15 Main Road	0821235258
C120	Thando	Zolani	88 Summer Road	0847541254
C121	Sedrick	Jackson	3 Long Road	0745556658
C122	Clark	Jones	7 Sea Road	0814745745

TOWNHOUSE

HOUSE_NUM	BUILD_PRICE	BEDROOMS	BATHROOMS
1	R 950 000	3	2
2	R 750 000	2	1
3	R 850 000	2	2
4	R 550 000	1	1
5	R 1 050 000	3	3

MATERIALS

MATERIAL_ID	MATERIAL_NAME
101	Roman Standard Roofing, Oak Doors, Luxury Garage Doors, Italian Style Tiling, Mizu Bath, Tri Glider Shower, Stella Staps
102	Standard Roofing, Standard Doors, Standard Garage Door, Standard Style Tiling, Standard Bath, Basic Shower, Basic Staps
103	Luxury Standard Roofing, Luxury Door, Luxury Garage Door, Luxury Style Tiling, Luxury Bath, Luxury Glider Shower, Luxury Staps

PURCHASE AGREEMENT

PURCHASE_ NUM	PURCHASE_ DATE	PURCHASE_AMT	HOUSE_NUM	CUST_ID	MATERIAL_ID
555	03-MAY-20	R 1 100 000	5	C121	103
556	05-MAY-20	R 1 050 000	1	C115	102
557	06-MAY-20	R 1 190 000	2	C120	103
558	07-MAY-20	R 1 120 000	3	C119	101

Question 1 **(Marks: 5)**

Your manager requires you to create an SQL query that will display the combined customer name, the townhouse number, materials used and the purchase agreement amount. In your query, only display the purchase agreement amounts greater than R1 100 000.

Sample Results

CUSTOMER	HOUSE_NUM	MATERIALS	PURCHASE_AMOUNT
Wayne, Bitterhout	3	Roman Standard Roofing, Oak Doors, Luxury Garage Doors, Italian Style Tiling, Mizu Bath, Tri Glider Shower, Stella Staps	R 1 120 000
Thando, Zolani	2	Luxury Standard Roofing, Luxury Door, Luxury Garage Door, Luxury Style Tiling, Luxury Bath, Luxury Glider Shower, Luxury Staps	R 1 190 000

Mark Allocation:

Requirement	Maximum Mark	Examiner's Mark	Moderator's Mark
Correct select statement used • Little or no attempt = 0 • Good – No changes required = 1 • Excellent select statement = 2	(2)		
Correct tables used • Little or no attempt = 0 • Good – Some tables used = 1 • Excellent – Tables used correctly = 2	(2)		
Correct code • Errors in code = 0 • Excellent – Code correct = 1	(1)		
TOTAL MARKS	5		

Question 2 (Marks: 10)

Your colleague, John, has been struggling with a PL/SQL query that should display the Customer ID and the House Number, including the Materials used in the construction of the townhouses. John doesn't seem to be able to determine why the query is not working and has asked for your assistance.

Identify the errors and **re-write** the correct code:

```
set serveroutput on
 c_id CUSTOMER.cust_id%Type;
 t TOWNHOUSE.house_num%Type;
 m_name MATERIALS.material_name%Type;
cursor info is
select cu.cust_id, t.house_num, m.material_name
from CUSTOMER cu, TOWNHOUSE t, MATERIALS m, PURCHASE_AGREEMENT pa
where cu.cust_id = pa.cust_id
and t.house_num = pa.house_num
and m.material_id = pa.material_id
begin
for rec in info
 c_id:=rec.cust_id;
 t:= rec.house_num;
 m_name:=rec.material_name;
 dbms_output_put_line('CUSTOMER ID:  ' || c_id || ', ' || chr(13) ||'HOUSE NUM:    ' || t ||
chr(13) || 'MATERIALS:    ' || m_name);
 dbms_output.put_line('----------');
 end loop;
```

They didn't include the declare variables and alias and variables seems to be matching with same values.

They forgot to add the semi-colon to end before the begin statement.

They forgot the semicolon at server output on.

They forgot to the End the statement after the start statement.

Question 3 **(Marks: 20)**

You would like to assist John in understanding the different explicit cursors he can use.

Using the scenario in **Question 2**, answer the following questions.

Q.3.1 Write the code solution using an explicit cursor with a simple Loop. Make sure to use your own variables. (10)

Mark Allocation:

Requirement	**Maximum Mark**	**Examiner's Mark**	**Moderator's Mark**
Variables declared correctly • Little or no attempt = 0 • Good: Variables nearly correct = 1 • Excellent: Variables correct = 2	(2)		
Correct select statement used • Little or no attempt = 0 • Good: No changes required = 1 – 2 • Excellent select statement = 3	(3)		
Correct tables used • Little or no attempt = 0 • Good: Some tables used = 1 • Excellent: Tables used correctly = 2	(2)		
Correct Execution section • Little or no attempt = 0 • Good: Some steps implemented = 1 – 2 • Excellent: All steps implemented = 3	(3)		
TOTAL MARKS	10		

Q.3.2 Using your solution in **Q.3.1**, **discuss** the steps involved when accessing the explicit cursor. (8)

Q.3.3 Give an example output of your solution. (2)

Note: Provide only one record example.

Question 4 (Marks: 10)

The Accounts department requested a single view with a specific name to be used by the department to select data from the tables. John had created a view called **House_View** to display the combined Customer Name, House Number, Material ID and the Purchase Amount as provided below. However, the department has requested the view to be modified.

Discuss the modifications required on the query to include a 2% discount to the customer and the total amount. In your query, also include the statement to run the view. Include the modified code.

Sample Result:

	CUSTOMER	HOUSE_NUM	MATERIAL_ID	PURCHASE_AMT	DISCOUNT	TOTAL
1	Philip,Willis	1	102	1050000	21000	1029000
2	Wayne,Bitterhout	3	101	1120000	22400	1097600
3	Thando,Zolani	2	103	1190000	23800	1166200
4	Sedrick,Jackson	5	103	1100000	22000	1078000

Original Code:

```
CREATE VIEW House_View
as
 select c.cust_fname || ', ' || c.cust_sname CUSTOMER, th.house_num, m.material_id,
pa.purchase_amt
 from CUSTOMER c, TOWNHOUSE th, MATERIALS m, PURCHASE_AGREEMENT pa
 where c.cust_id = pa.cust_id
 and th.house_num = pa.house_num
 and m.material_id = pa.material_id;
```

Question 5 (Marks: 5)

Explain the role and purpose of the two codes provided below.

```
Create or Replace Trigger Purchase_Entry
After Insert or Update of Purchase_Amt on PURCHASE_AGREEMENT
For Each Row
Begin
  if(:New.Purchase_Amt < 0) Then
   RAISE_APPLICATION_ERROR(-20100,
     'Cannot enter a purchase agreement amount less than zero.');
  end if;
End;
```

The role of the SQL trigger is a piece of code that executes automatically in response to the specific event that happened in a database. so in this code they create a trigger code on 'purchase_Amt on Purchase_Agreement' if user provides a value that is less than zero then it pops up the exception handling error, to make sure user mustn't enter any values that is less than zero. so the exception handling error that they implement will helps not to break the program completely.

```
insert into PURCHASE_AGREEMENT values (559, '27 May 2020', -1, 12350,'C122',1)
```

Question 6 (Marks: 10)

Below is the code for a procedure that will accept a Customer ID as an input parameter and display the Customer ID, Customer Surname and Customer Address.

Review the code, then answer the questions that follow.

```
create or replace procedure sp_customer_details (c_id IN CUSTOMER.cust_id%TYPE,
customer_details out VARCHAR2)
IS
BEGIN
SELECT 'CUSTOMER ID:' || c.cust_id || ', Surname:'|| c.cust_sname || ',ADDRESS:' ||
c.cust_address
into customer_details
from CUSTOMER c
where c.cust_id = c_id;
END;
```

Having %Type, it allows to create a unique data type that is created in a database of Customer table. so which makes user easier to generate declare data type automatically using the Customer table and it also time saving for user to search the customer ID data type

Q.6.1 Briefly discuss the advantages of using the **%TYPE** attribute in the code. (4)

Q.6.2 Create the code to execute the procedure with exception handling if no customer data is found. Use customer ID **C115** in your code. (6)

Mark Allocation

Requirement	Maximum Mark	Examiner's Mark	Moderator's Mark
DECLARATION BLOCK Correct variables created • Not attempted = 0 • Minor changes required = 1 • Good: No changes required = 2	(2)		
PROCEDURAL BLOCK Correct code to run the procedure • Not attempted = 0 • Good: Code nearly correct = 1 • Excellent: Code correct = 2	(2)		
EXCEPTION BLOCK Correct code to handle the exception • Not attempted = 0 • Good: Code nearly correct = 1 • Excellent: Code correct = 2	(2)		
TOTAL MARKS	6		

Question 7 (Marks: 10)

Review the code below then answer the questions that follow.

LINE NUMBER	
1	Create or replace Function House_Details(h_num in townhouse.house_num%Type)
2	RETURN varchar2
3	IS
4	details varchar(100);
5	cursor c1 is
6	select c.cust_id || ', R ' || t.build_price || ', ' || pa.purchase_date || ', ' || pa.purchase_amt
7	from CUSTOMER c, TOWNHOUSE t, PURCHASE_AGREEMENT pa
8	where c.cust_id = pa.cust_id
9	and t.house_num = pa.house_num
10	and t.house_num = h_num;
11	BEGIN
12	open c1;
13	fetch c1 into details;
14	if c1%notfound then
15	details := 'No house information found';
16	end if;
17	RETURN details;
18	close c1;
19	EXCEPTION
20	WHEN OTHERS THEN
21	raise_application_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR-
22	'||SQLERRM);
23	END;
24	
25	**select House_Details(3) from dual;**

The purpose of function allows to use the code for multiple times without have to rewrite the function code and which helps to reduce the redundance of the code.

It also ability to deal the complex task in SQL, were it breaks down the complex task into smaller tasks and each function performs specific task.

Q.7.1 **Briefly explain** the purpose of the function. (2)

Q.7.2 Provide the output result using any input value of your choice. (2)

Q.7.3 **Defend** the use of a function instead of a procedure using examples from the query. (6)

END OF PAPER