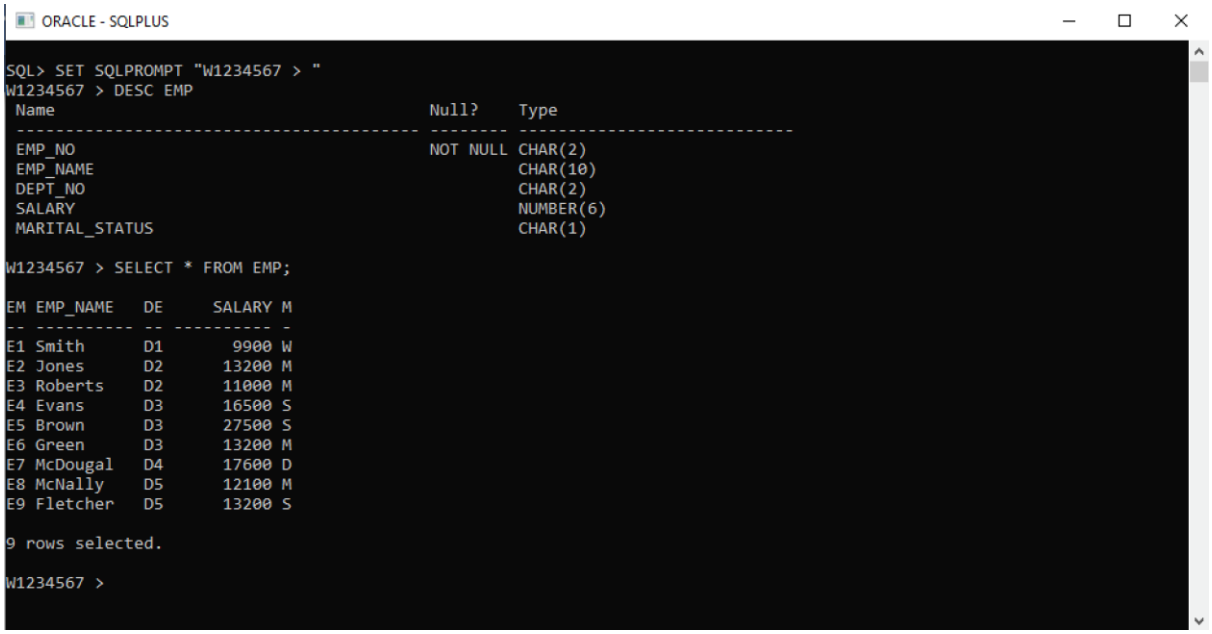
Module Title:	Database Modelling
Module Code:	KC7013
Academic Year / Semester:	2022-23 / Semester 2
Module Tutor / Email (all queries):	Akhtar Ali akhtar.ali@northumbria.ac.uk
% Weighting (to overall module):	50%
Assessment Title:	Assignment 2: Design and implementation of an object-relational database and a data warehouse
Date of Handout to Students:	Tuesday 21st March 2023
Mechanism for Handout:	Module Blackboard Site & Live Session in Week 9
Deadline for Submission Attempt by Students:	Thursday 18th May 2023 @ 23:59
Mechanism for Submission:	Document upload to Module Blackboard Site
Submission Format / Word Count	Please upload your written report as a single PDF document
Date by which Work, Feedback and Marks will be returned:	Monday 19th June 2023
Mechanism for return of Feedback and Marks:	Mark and individual written feedback will be uploaded to the Module Site on Blackboard. For further queries please email module tutor.
Student ID **Oracle Username** **Data Warehouse Username**	W22046071 dbmUsr91 DWU390
Student Name	RABBANI YAKHUBJI

ASSIGNMENT # 2

Personalising your SQL output/prompt

Before executing any **SQL code** for this assignment, you should personalise your SQL output / prompt by running SET SQLPROMPT “UniversityUserName > ”, i.e., *double-quote* followed by your UniversityUserName followed by > and then a *space* and *double-quote* as shown in the screenshot below:



Assignment Questions

Part 1 (50 marks)

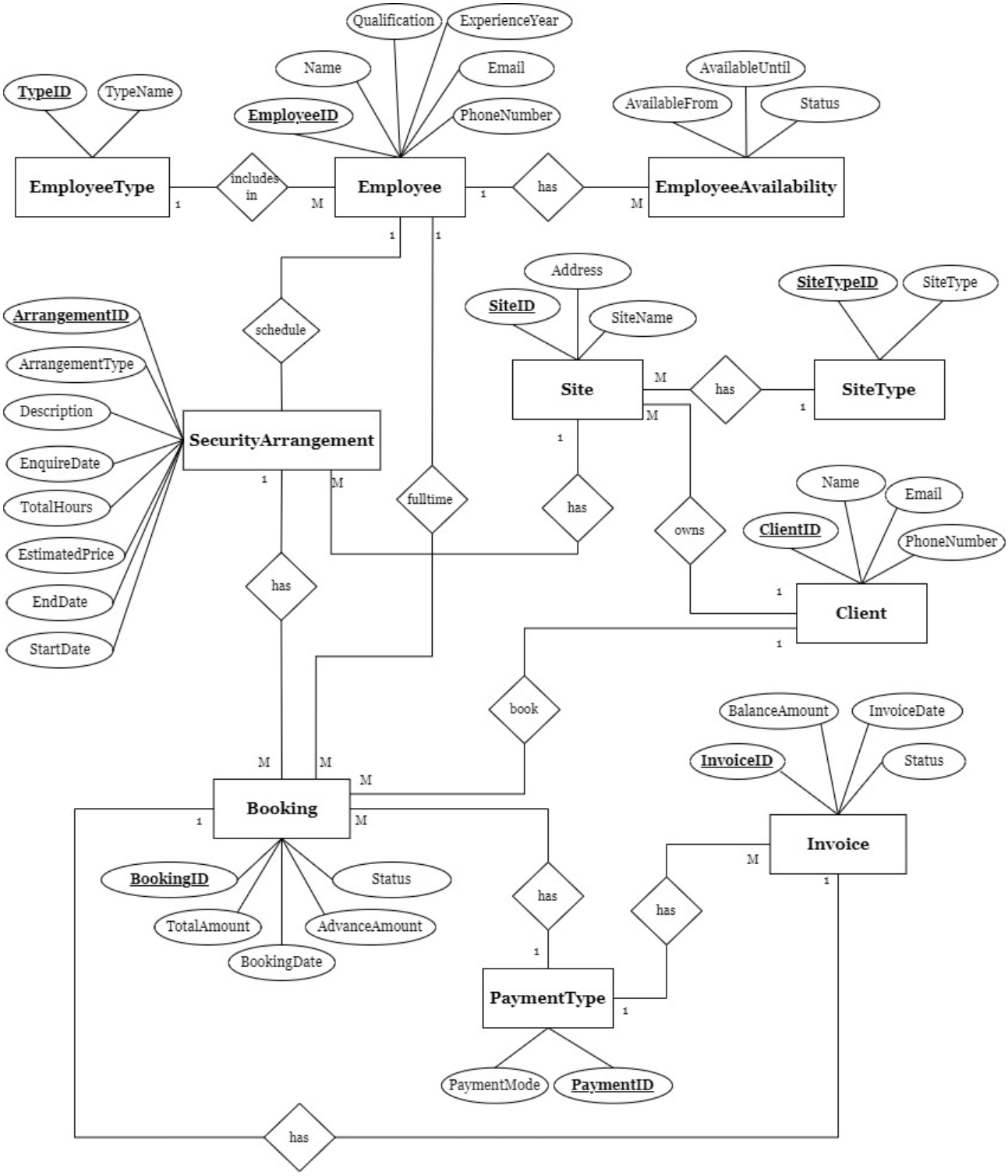
A) Implementing Object-Relational version of a subset of the NorthEastNinjas Database (26 marks)

You need to choose and justify a subset of the **NorthEastNinjas Database** from your assignment 1 submission (2 marks). The subset should include four entity types / relations who have relationships with each other.

Implement **your chosen subset** the NorthEastNinjas Database using object-relational (OR) features of Oracle 11g/12c/19c (Kannan, 2019). You need to provide a logical design for the OR version of your chosen subset (4 marks), and provide complete SQL code for implementing your design using OR features of Oracle database system (20 marks).

Note that all relationships (e.g., one-to-one, one-to-many, many-to-many) must be bi-directional. Moreover, each to-many side of a relationship (e.g., in case of many-to-one and many-to-many) should be implemented using nested tables. In addition, your implementation should allow all objects to be ***shareable*** *(i.e., all relationships should be* **REF** *based).*

Provide Revised ERD Diagram or Logical Design of the **NorthEastNinjas Database** from your assignment 1 for Reference purpose here.



Answer Part 1 A:

1) Provide below your choice and justification of what aspects (subset) of the **NorthEastNinjas Database** (2 marks)

In this scenario, all the eight entities such as employee type, employee, employee availability, security arrangement, payment type, booking, invoice, and client are used. In order to save all the data without missing any of them, all the eight entities was used and hence justified.

2) Logical Design of your object-relational subset of the **NorthEastNinjas Database** (4 marks)

TabClient

ClientID	Name	Address	PhoneNumber	Email
1	Shaiksha	36 Rowland Rd	07061188519	shaiksha@gmail.com

TabSiteType

SiteTypeID	TabSiteType
1	Property

TabSite

SiteID	ClientID	SiteTypeID	SiteName	Address
1	CleRef*	SiteRef*	Wonderful Homes	Newcastle upon Tyne

3) SQL Code for creating Object types including **nested table types, make sure it is TEXT rather than image or screenshot** (14 marks):

```
CREATE TYPE TEmployeeType AS OBJECT (
      TypeID NUMBER(5),
      TypeName VARCHAR(20)
)
/

CREATE TYPE NeEmployeeType AS TABLE OF TEmployeeType
/

CREATE TYPE TEmployees AS OBJECT (
      EmployeeID NUMBER(5),
      TypeID NeEmployeeType,
      Name VARCHAR(20),
      PhoneNumber NUMBER(10),
      Email VARCHAR(30),
      Qualification VARCHAR(20),
      ExperienceYear NUMBER(5)
)
/

CREATE TYPE NeEmployees AS TABLE OF TEmployees

CREATE TYPE TEmployeeAvailability AS OBJECT(
      MasterCalendarID NUMBER(5),
      EmployeeID REF TEmployees,
      AvailableFrom DATE,
      AvailableUntil DATE,
      Status VARCHAR(20)
)
/

CREATE TYPE NEmployeeAvailability AS TABLE OF TEmployeeAvailability
/

CREATE TYPE TClient AS OBJECT (
      ClientID NUMBER(5),
      Name VARCHAR(20),
      Address VARCHAR(20),
```

```
      PhoneNumber NUMBER(10),
      Email VARCHAR(30)
)
/

CREATE TYPE NClient AS TABLE OF TClient
/

CREATE TYPE TSiteType AS OBJECT (
      SiteTypeID NUMBER(5),
      TabSiteType VARCHAR(20)
)
/

CREATE TYPE NSiteType AS TABLE OF TSiteType
/

CREATE TYPE TSite AS OBJECT (
      SiteID NUMBER(5),
      ClientID REF TClient,
      SiteTypeID REF TSiteType,
      SiteName VARCHAR(20),
      Address VARCHAR(20)
)
/

CREATE TYPE TSecurityArrangement AS OBJECT (
      ArrangementID NUMBER(5),
      FullTimeStaffID REF TEmployees,
      SiteID REF TSite,
      ArrangementType VARCHAR(20),
      Description VARCHAR(40),
      TotalHours NUMBER(5),
      StartDate DATE,
      EndDate DATE,
      EstimatedPrice DECIMAL(5,2)
)
/

CREATE TYPE TPaymentType AS OBJECT (
      PaymentID NUMBER(5),
```

```
      PaymentMode VARCHAR(20)
)
/

CREATE TYPE NPaymentType AS TABLE OF TPaymentType
/

CREATE TYPE TBooking AS OBJECT (
      BookingID NUMBER(5),
      ArrangementID REF TSecurityArrangement,
      FullTimeStaffID REF TEmployees,
      IndividualsStaffID REF TEmployees,
      ClientID REF TClient,
      PaymentID REF TPaymentType,
      TotalAmount DECIMAL(5,2),
      AdvanceAmount DECIMAL(5,2),
      BookingDate DATE,
      Status VARCHAR(20)
)
/

CREATE TYPE NBooking AS TABLE OF TBooking
/

CREATE TYPE TInvoice AS OBJECT (
      InvoiceID NUMBER(5),
      BookingID REF TBooking,
      PaymentID REF TPaymentType,
      BalanceAmount DECIMAL(5,2),
      InvoiceDate DATE,
      Status VARCHAR(20)
)
/
```

4) SQL Code for creating Object tables including **nested tables, make sure it is TEXT rather than image or screenshot** (6 marks):

```
CREATE TABLE TabEmployeeType OF TEmployeeType(
      PRIMARY KEY(TypeID)
)
/

CREATE TABLE TabEmployees OF TEmployees(
      PRIMARY KEY(EmployeeID)
)
Nested Table TypeID Store as TypeID_Obj
/

CREATE TABLE TabEmployeeAvailability OF TEmployeeAvailability(
      PRIMARY KEY(MasterCalendarID),
      FOREIGN KEY(EmployeeID) REFERENCES TabEmployees
)
/

CREATE TABLE TabClient OF TClient(
      PRIMARY KEY(ClientID)
)
/

CREATE TABLE TabSiteType OF TSiteType(
      PRIMARY KEY(SiteTypeID)
)
/

CREATE TABLE TabSite OF TSite(
      PRIMARY KEY(SiteID),
      FOREIGN KEY(ClientID) REFERENCES TabClient,
      FOREIGN KEY(SiteTypeID) REFERENCES TabSiteType
)
/

CREATE TABLE TabSecurityArrangement OF TSecurityArrangement(
      PRIMARY KEY(ArrangementID),
```

```
      FOREIGN KEY(FullTimeStaffID) REFERENCES TabEmployees,
      FOREIGN KEY(SiteID) REFERENCES TabSite
)
/

CREATE TABLE TabPaymentType OF TPaymentType(
      PRIMARY KEY(PaymentID)
)
/

CREATE TABLE TabBooking OF TBooking(
      PRIMARY KEY(BookingID),
      FOREIGN KEY(ArrangementID) REFERENCES TabSecurityArrangement,
      FOREIGN KEY(FullTimeStaffID) REFERENCES TabEmployees,
      FOREIGN KEY(individualsStaffID) REFERENCES TabEmployees,
      FOREIGN KEY(ClientID) REFERENCES TabClient,
      FOREIGN KEY(PaymentID) REFERENCES TabPaymentType
)
/

CREATE TABLE TabInvoice OF TInvoice(
      PRIMARY KEY(InvoiceID),
      FOREIGN KEY(BookingID) REFERENCES TabBooking,
      FOREIGN KEY(PaymentID) REFERENCES TabPaymentType
)
/
```

5) **SQL Output (e.g., SPOOL file contents or output you got when you executed your above SQL code, this should show the SQL code as well as its output / result). Make sure the output is NOT TEXT but rather images or screenshots. If missing or inadequate, then up to 4 marks will be deducted from the above 26 marks for this.**

```
W22046071 >
W22046071 > CREATE TYPE TEmployeeType AS OBJECT (
  2  TypeID NUMBER(5),
  3  TypeName VARCHAR(20)
  4  )
  5  /

Type created.

W22046071 >
W22046071 > CREATE TYPE NeEmployeeType AS TABLE OF TEmployeeType
  2  /

Type created.

W22046071 >
```

```
W22046071 >
W22046071 > CREATE TYPE TEmployees AS OBJECT (
  2  EmployeeID NUMBER(5),
  3  TypeID NeEmployeeType,
  4  Name VARCHAR(20),
  5  PhoneNumber NUMBER(10),
  6  Email VARCHAR(30),
  7  Qualification VARCHAR(20),
  8  ExperienceYear NUMBER(5)
  9  )
 10  /

Type created.

W22046071 >
W22046071 > CREATE TYPE NeEmployees AS TABLE OF TEmployees
  2  /

Type created.

W22046071 >
```

```
W22046071 >
W22046071 > CREATE TYPE TEmployeeAvailability AS OBJECT(
  2  MasterCalendarID NUMBER(5),
  3  EmployeeID REF TEmployees,
  4  AvailableFrom DATE,
  5  AvailableUntil DATE,
  6  Status VARCHAR(20)
  7  )
  8  /

Type created.

W22046071 >
W22046071 > CREATE TYPE NEmployeeAvailability AS TABLE OF TEmployeeAvailability
  2  /

Type created.

W22046071 >
```

```
W22046071 >
W22046071 > CREATE TYPE TClient AS OBJECT (
  2  ClientID NUMBER(5),
  3  Name VARCHAR(20),
  4  Address VARCHAR(20),
  5  PhoneNumber NUMBER(10),
  6  Email VARCHAR(30)
  7  )
  8  /

Type created.

W22046071 >
W22046071 > CREATE TYPE NClient AS TABLE OF TClient
  2  /

Type created.

W22046071 >
```

```
W22046071 >
W22046071 > CREATE TYPE TSiteType AS OBJECT (
  2  SiteTypeID NUMBER(5),
  3  TabSiteType VARCHAR(20)
  4  )
  5  /

Type created.

W22046071 >
W22046071 > CREATE TYPE NSiteType AS TABLE OF TSiteType
  2  /

Type created.

W22046071 >
```

```
W22046071 >
W22046071 > CREATE TYPE TSite AS OBJECT (
  2  SiteID NUMBER(5),
  3  ClientID REF TClient,
  4  SiteTypeID REF TSiteType,
  5  SiteName VARCHAR(20),
  6  Address VARCHAR(20)
  7  )
  8  /

Type created.

W22046071 >
W22046071 > CREATE TYPE TSecurityArrangement AS OBJECT (
  2  ArrangementID NUMBER(5),
  3  FullTimeStaffID REF TEmployees,
  4  SiteID REF TSite,
  5  ArrangementType VARCHAR(20),
  6  Description VARCHAR(40),
  7  TotalHours NUMBER(5),
  8  StartDate DATE,
  9  EndDate DATE,
 10  EstimatedPrice DECIMAL(5,2)
 11  )
 12  /

Type created.

W22046071 >
W22046071 >
```

```
W22046071 >
W22046071 > CREATE TYPE TPaymentType AS OBJECT (
  2  PaymentID NUMBER(5),
  3  PaymentMode VARCHAR(20)
  4  )
  5  /

Type created.

W22046071 >
W22046071 > CREATE TYPE NPaymentType AS TABLE OF TPaymentType
  2  /

Type created.

W22046071 >
```

```
W22046071 >
W22046071 > CREATE TYPE TBooking AS OBJECT (
  2  BookingID NUMBER(5),
  3  ArrangementID REF TSecurityArrangement,
  4  FullTimeStaffID REF TEmployees,
  5  IndividualsStaffID REF TEmployees,
  6  ClientID REF TClient,
  7  PaymentID REF TPaymentType,
  8  TotalAmount DECIMAL(5,2),
  9  AdvanceAmount DECIMAL(5,2),
 10  BookingDate DATE,
 11  Status VARCHAR(20)
 12  )
 13  /

Type created.

W22046071 >
W22046071 > CREATE TYPE NBooking AS TABLE OF TBooking
  2  /

Type created.

W22046071 >
W22046071 > CREATE TYPE TInvoice AS OBJECT (
  2  InvoiceID NUMBER(5),
  3  BookingID REF TBooking,
  4  PaymentID REF TPaymentType,
  5  BalanceAmount DECIMAL(5,2),
  6  InvoiceDate DATE,
  7  Status VARCHAR(20)
  8  )
  9  /

Type created.

W22046071 >
```

```
W22046071 >
W22046071 > CREATE TABLE TabEmployeeType OF TEmployeeType(
  2  PRIMARY KEY(TypeID)
  3  )
  4  /

Table created.

W22046071 >
W22046071 >
W22046071 > CREATE TABLE TabEmployees OF TEmployees(
  2  PRIMARY KEY(EmployeeID)
  3  )
  4  Nested Table TypeID Store as TypeID_Obj
  5  /

Table created.

W22046071 >
W22046071 > CREATE TABLE TabEmployeeAvailability OF TEmployeeAvailability(
  2  PRIMARY KEY(MasterCalendarID),
  3  FOREIGN KEY(EmployeeID) REFERENCES TabEmployees
  4  )
  5  /

Table created.

W22046071 >
```

```
W22046071 >
W22046071 > CREATE TABLE TabClient OF TClient(
  2  PRIMARY KEY(ClientID)
  3  )
  4  /

Table created.

W22046071 >
W22046071 > CREATE TABLE TabSiteType OF TSiteType(
  2  PRIMARY KEY(SiteTypeID)
  3  )
  4  /

Table created.

W22046071 >
W22046071 > CREATE TABLE TabSite OF TSite(
  2  PRIMARY KEY(SiteID),
  3  FOREIGN KEY(ClientID) REFERENCES TabClient,
  4  FOREIGN KEY(SiteTypeID) REFERENCES TabSiteType
  5  )
  6  /

Table created.

W22046071 >
W22046071 >
```

```
W22046071 >
W22046071 >
W22046071 > CREATE TABLE TabSecurityArrangement OF TSecurityArrangement(
  2  PRIMARY KEY(ArrangementID),
  3  FOREIGN KEY(FullTimeStaffID) REFERENCES TabEmployees,
  4  FOREIGN KEY(SiteID) REFERENCES TabSite
  5  )
  6  /

Table created.

W22046071 >
W22046071 >
W22046071 > CREATE TABLE TabPaymentType OF TPaymentType(
  2  PRIMARY KEY(PaymentID)
  3  )
  4  /

Table created.

W22046071 >
```

```
W22046071 >
W22046071 > CREATE TABLE TabBooking OF TBooking(
  2  PRIMARY KEY(BookingID),
  3  FOREIGN KEY(ArrangementID) REFERENCES TabSecurityArrangement,
  4  FOREIGN KEY(FullTimeStaffID) REFERENCES TabEmployees,
  5  FOREIGN KEY(individualsStaffID) REFERENCES TabEmployees,
  6  FOREIGN KEY(ClientID) REFERENCES TabClient,
  7  FOREIGN KEY(PaymentID) REFERENCES TabPaymentType
  8  )
  9  /

Table created.

W22046071 >
W22046071 >
W22046071 > CREATE TABLE TabInvoice OF TInvoice(
  2  PRIMARY KEY(InvoiceID),
  3  FOREIGN KEY(BookingID) REFERENCES TabBooking,
  4  FOREIGN KEY(PaymentID) REFERENCES TabPaymentType
  5  )
  6  /

Table created.

W22046071 >
W22046071 > _
```

B) Populating the OR version of the subset of the NorthEastNinjas Database (10 marks)

Using PL/SQL and/or SQL, populate your OR version of the subset of the NorthEastNinjas Database with some sample data. This part can be seen to contain the following sub-tasks:

(a) Creating / inserting objects in your object tables
(b) Populating the one-to-many, many-to-one, and many-to-many relationships among the objects created in sub-task (a).

Answer Part 1 B:

1) Provide PL/SQL and/or SQL code for **creating / inserting objects in object tables** and **populating** relevant one-to-many, many-to-one, and many-to-many **relationships** among the objects in the database (10 Marks)

Inserting object tables

```
INSERT INTO TabEmployeeType VALUES (1, 'Full Time');
INSERT INTO TabEmployeeType VALUES (2, 'Individuals');
INSERT INTO TabEmployeeType VALUES (3, 'Managers');
INSERT INTO TabEmployeeType VALUES (4, 'Individuals');
INSERT INTO TabEmployeeType VALUES (5, 'Full Time');
INSERT INTO TabEmployeeType VALUES (6, 'Managers');
INSERT INTO TabEmployeeType VALUES (7, 'Full Time');
INSERT INTO TabEmployeeType VALUES (8, 'Full Time');
INSERT INTO TabEmployeeType VALUES (9, 'Managers');
INSERT INTO TabEmployeeType VALUES (10, 'Individuals');

INSERT INTO TabEmployees VALUES (1, NeEmployeeType(TEmployeeType(1, 'Full
Time')), 'Yakhubji', 123456789, 'yakhubji@gmail.com', 'MBA', 5);
INSERT INTO TabEmployees VALUES (2, NeEmployeeType(TEmployeeType(2,
'Individuals')), 'Shahid', 987654321, 'shahid@gmail.com', 'B.Sc.', 10);
INSERT INTO TabEmployees VALUES (3, NeEmployeeType(TEmployeeType(1, 'Full
Time')), 'Irfan', 789456321, 'irfan@gmail.com', 'M.Tech', 8);
INSERT INTO TabEmployees VALUES (4, NeEmployeeType(TEmployeeType(2,
'Individuals')), 'Arif', 124567890, 'arif@gmail.com', 'M.Sc.', 3);
INSERT INTO TabEmployees VALUES (5, NeEmployeeType(TEmployeeType(3,
'Managers')), 'Inthiyaz', 321456709, 'inthiyaz@gmail.com', 'B.Tech', 4);
INSERT INTO TabEmployees VALUES (6, NeEmployeeType(TEmployeeType(1, 'Full
Time')), 'Ghouse', 543219876, 'ghouse@gmail.com', 'MBA', 2);
INSERT INTO TabEmployees VALUES (7, NeEmployeeType(TEmployeeType(3,
'Managers')), 'Jeelan', 908765432, 'jeelan@gmail.com', 'B.Sc.', 6);
INSERT INTO TabEmployees VALUES (8, NeEmployeeType(TEmployeeType(2,
'Individuals')), 'Vahid', 765432109, 'vahid@gmail.com', 'M.Sc.', 7);
INSERT INTO TabEmployees VALUES (9, NeEmployeeType(TEmployeeType(3,
'Managers')), 'Mainu', 876543210, 'mainu@gmail.com', 'B.Tech', 9);
INSERT INTO TabEmployees VALUES (10, NeEmployeeType(TEmployeeType(1, 'Full
Time')), 'Kasif', 654321987, 'kasif@gmail.com', 'MBA', 1);

INSERT INTO TabEmployeeAvailability VALUES (1,NULL, '01-JUN-2023', '19-JULY-
2023', 'Vacation');
UPDATE TabEmployeeAvailability SET EmployeeID = (SELECT REF(E) FROM TabEmployees
E WHERE EmployeeID = 4)WHERE MasterCalendarID = 1;
INSERT INTO TabEmployeeAvailability VALUES (2,NULL, '20-MAR-2023', '05-APR-
2023', 'Not available');
```

```
UPDATE TabEmployeeAvailability SET EmployeeID = (SELECT REF(E) FROM TabEmployees
E WHERE EmployeeID = 2)WHERE MasterCalendarID = 2;
INSERT INTO TabEmployeeAvailability VALUES (3,NULL, '15-MAY-2023', '23-JUN-
2023', 'Available');
UPDATE TabEmployeeAvailability SET EmployeeID = (SELECT REF(E) FROM TabEmployees
E WHERE EmployeeID = 1)WHERE MasterCalendarID = 3;
INSERT INTO TabEmployeeAvailability VALUES (4,NULL, '05-SEP-2023', '04-OCT-
2023', 'Working');
UPDATE TabEmployeeAvailability SET EmployeeID = (SELECT REF(E) FROM TabEmployees
E WHERE EmployeeID = 1)WHERE MasterCalendarID = 4;
INSERT INTO TabEmployeeAvailability VALUES (5,NULL, '11-JUN-2023', '07-JULY-
2023', 'Vacation');
UPDATE TabEmployeeAvailability SET EmployeeID = (SELECT REF(E) FROM TabEmployees
E WHERE EmployeeID = 5)WHERE MasterCalendarID = 5;

INSERT INTO TabClient VALUES (1, 'Shaiksha', '36 Rowland Rd', 07061188519,
'shaiksha@gmail.com');
INSERT INTO TabClient VALUES (2, 'Shahul', '41 Stone Cellar Road', 07017170994,
'shahul@gmail.com');
INSERT INTO TabClient VALUES (3, 'Nikith', '25 Withers Close', 0708544795,
'davewilson@gmail.com');
INSERT INTO TabClient VALUES (4, 'Aadhira', '59 Henley Road', 07858417732,
'aadhira@gmail.com');
INSERT INTO TabClient VALUES (5, 'Jagath', '38 Earls Avenue', 07069616402,
'jagath@gmail.com');
INSERT INTO TabClient VALUES (6, 'Loshith', '45 Park End St', 07038550069,
'loshith@gmail.com');
INSERT INTO TabClient VALUES (7, 'Nithin', '16 Fulford Road', 07917388035,
'nithin@gmail.com');
INSERT INTO TabClient VALUES (8, 'Saswin', '55 Tonbridge Rd', 07933281321,
'saswin@gmail.com');

INSERT INTO TabSiteType VALUES (1, 'Property');
INSERT INTO TabSiteType VALUES (2, 'Business Premises');
INSERT INTO TabSiteType VALUES (3, 'Construction');
INSERT INTO TabSiteType VALUES (4, 'Industrial');
INSERT INTO TabSiteType VALUES (5, 'Private Individual');
INSERT INTO TabSiteType VALUES (6, 'Private Business');
INSERT INTO TabSiteType VALUES (7, 'Personal');
INSERT INTO TabSiteType VALUES (8, 'Commercial');
```

```
INSERT INTO TabSite VALUES (1, NULL, NULL, 'Wonderful Homes', 'Newcastle upon
Tyne');
UPDATE TabSite SET ClientID = (SELECT REF(C) FROM TabClient C WHERE ClientID =
1)WHERE SiteID = 1;
UPDATE TabSite SET SiteTypeID = (SELECT REF(ST) FROM TabSiteType ST WHERE
SiteTypeID = 1)WHERE SiteID = 1;
INSERT INTO TabSite VALUES (2, NULL, NULL, 'Luxury Ohio', 'Newcastle upon
Tyne');
UPDATE TabSite SET ClientID = (SELECT REF(C) FROM TabClient C WHERE ClientID =
1)WHERE SiteID = 2;
UPDATE TabSite SET SiteTypeID = (SELECT REF(ST) FROM TabSiteType ST WHERE
SiteTypeID = 2)WHERE SiteID = 2;
INSERT INTO TabSite VALUES (3, NULL, NULL, 'Texas Waterfront', 'Newcastle upon
Tyne');
UPDATE TabSite SET ClientID = (SELECT REF(C) FROM TabClient C WHERE ClientID =
2)WHERE SiteID = 3;
UPDATE TabSite SET SiteTypeID = (SELECT REF(ST) FROM TabSiteType ST WHERE
SiteTypeID = 3)WHERE SiteID = 3;
INSERT INTO TabSite VALUES (4, NULL, NULL, 'Sarah Sells Homes', 'Newcastle upon
Tyne');
UPDATE TabSite SET ClientID = (SELECT REF(C) FROM TabClient C WHERE ClientID =
3)WHERE SiteID = 4;
UPDATE TabSite SET SiteTypeID = (SELECT REF(ST) FROM TabSiteType ST WHERE
SiteTypeID = 4)WHERE SiteID = 4;
INSERT INTO TabSite VALUES (5, NULL, NULL, 'Tom Smith Realty', 'Newcastle upon
Tyne');
UPDATE TabSite SET ClientID = (SELECT REF(C) FROM TabClient C WHERE ClientID =
4)WHERE SiteID = 5;
UPDATE TabSite SET SiteTypeID = (SELECT REF(ST) FROM TabSiteType ST WHERE
SiteTypeID = 1)WHERE SiteID = 5;
INSERT INTO TabSite VALUES (6, NULL, NULL, 'Johnand Jacob Realty', '7
Whatlington Road');
UPDATE TabSite SET ClientID = (SELECT REF(C) FROM TabClient C WHERE ClientID =
5)WHERE SiteID = 6;
UPDATE TabSite SET SiteTypeID = (SELECT REF(ST) FROM TabSiteType ST WHERE
SiteTypeID = 2)WHERE SiteID = 6;
INSERT INTO TabSite VALUES (7, NULL, NULL, 'Anderson Realty', '50 Withers
Close');
UPDATE TabSite SET ClientID = (SELECT REF(C) FROM TabClient C WHERE ClientID =
6)WHERE SiteID = 7;
```

```
UPDATE TabSite SET SiteTypeID = (SELECT REF(ST) FROM TabSiteType ST WHERE
SiteTypeID = 3)WHERE SiteID = 7;
INSERT INTO TabSite VALUES (8, NULL, NULL, 'BuyHome4 Family', '80 Nether
Crescent');
UPDATE TabSite SET ClientID = (SELECT REF(C) FROM TabClient C WHERE ClientID =
7)WHERE SiteID = 8;
UPDATE TabSite SET SiteTypeID = (SELECT REF(ST) FROM TabSiteType ST WHERE
SiteTypeID = 4)WHERE SiteID = 8;

INSERT INTO TabSecurityArrangement VALUES (1, NULL, NULL, 'Weekdays only',
'Security for the office', 80, '15-MAR-2023', '01-APR-2023', 160.00);
UPDATE TabSecurityArrangement SET FullTimeStaffID = (SELECT REF(E) FROM
TabEmployees E WHERE EmployeeID = 5)WHERE ArrangementID = 1;
UPDATE TabSecurityArrangement SET SiteID = (SELECT REF(S) FROM TabSite S WHERE
SiteID = 3)WHERE ArrangementID = 1;
INSERT INTO TabSecurityArrangement VALUES (2, NULL, NULL, 'Weekend only',
'Security for the entrance', 48, '01-FEB-2023', '15-APR-2023', 240.00);
UPDATE TabSecurityArrangement SET FullTimeStaffID = (SELECT REF(E) FROM
TabEmployees E WHERE EmployeeID = 7)WHERE ArrangementID = 2;
UPDATE TabSecurityArrangement SET SiteID = (SELECT REF(S) FROM TabSite S WHERE
SiteID = 4)WHERE ArrangementID = 1;
INSERT INTO TabSecurityArrangement VALUES (3, NULL, NULL, 'Weekdays and
weekend', 'Security for the parking lot', 128, '25-JUL-2023', '20-AUG-2023',
120.00);
UPDATE TabSecurityArrangement SET FullTimeStaffID = (SELECT REF(E) FROM
TabEmployees E WHERE EmployeeID = 9)WHERE ArrangementID = 3;
UPDATE TabSecurityArrangement SET SiteID = (SELECT REF(S) FROM TabSite S WHERE
SiteID = 2)WHERE ArrangementID = 1;
INSERT INTO TabSecurityArrangement VALUES (4, NULL, NULL, 'Weekdays only',
'Security for the main gate', 80, '10-MAY-2023', '15-JUN-2023', 160.00);
UPDATE TabSecurityArrangement SET FullTimeStaffID = (SELECT REF(E) FROM
TabEmployees E WHERE EmployeeID = 5)WHERE ArrangementID = 4;
UPDATE TabSecurityArrangement SET SiteID = (SELECT REF(S) FROM TabSite S WHERE
SiteID = 1)WHERE ArrangementID = 1;
INSERT INTO TabSecurityArrangement VALUES (5, NULL, NULL, 'Weekend only',
'Security for the reception area', 48, '20-APR-2023', '01-APR-2023', 185.00);
UPDATE TabSecurityArrangement SET FullTimeStaffID = (SELECT REF(E) FROM
TabEmployees E WHERE EmployeeID = 7)WHERE ArrangementID = 5;
UPDATE TabSecurityArrangement SET SiteID = (SELECT REF(S) FROM TabSite S WHERE
SiteID = 3)WHERE ArrangementID = 1;
```

```
INSERT INTO TabSecurityArrangement VALUES (6, NULL, NULL, 'Weekdays and
weekend', 'Security for the storeroom', 128, '12-FEB-2023', '15-MAR-2023',
196.00);
UPDATE TabSecurityArrangement SET FullTimeStaffID = (SELECT REF(E) FROM
TabEmployees E WHERE EmployeeID = 9)WHERE ArrangementID = 6;
UPDATE TabSecurityArrangement SET SiteID = (SELECT REF(S) FROM TabSite S WHERE
SiteID = 4)WHERE ArrangementID = 1;
INSERT INTO TabSecurityArrangement VALUES (7, NULL, NULL, 'Weekdays only',
'Security for the building', 80, '25-MAR-2023', '25-MAR-2023', 196.00);
UPDATE TabSecurityArrangement SET FullTimeStaffID = (SELECT REF(E) FROM
TabEmployees E WHERE EmployeeID = 7)WHERE ArrangementID = 7;
UPDATE TabSecurityArrangement SET SiteID = (SELECT REF(S) FROM TabSite S WHERE
SiteID = 1)WHERE ArrangementID = 1;
INSERT INTO TabSecurityArrangement VALUES (8, NULL, NULL, 'Weekend only',
'Security for the conference room', 48, '10-JAN-2023', '10-JAN-2023', 178.00);
UPDATE TabSecurityArrangement SET FullTimeStaffID = (SELECT REF(E) FROM
TabEmployees E WHERE EmployeeID = 5)WHERE ArrangementID = 8;
UPDATE TabSecurityArrangement SET SiteID = (SELECT REF(S) FROM TabSite S WHERE
SiteID = 3)WHERE ArrangementID = 1;

INSERT INTO TabPaymentType VALUES (1, 'Credit card');
INSERT INTO TabPaymentType VALUES (2, 'Cash');
INSERT INTO TabPaymentType VALUES (3, 'Cheque');

INSERT INTO TabBooking VALUES (1, NULL, NULL, NULL, NULL, NULL, 150.00, 25.00,
'14-MAR-2023', 'Advance Paid');
UPDATE TabBooking SET ArrangementID = (SELECT REF(E) FROM TabSecurityArrangement
E WHERE ArrangementID = 4)WHERE BookingID = 1;
UPDATE TabBooking SET FullTimeStaffID = (SELECT REF(E) FROM TabEmployees E WHERE
EmployeeID = 1)WHERE BookingID = 1;
UPDATE TabBooking SET individualsStaffID = (SELECT REF(E) FROM TabEmployees E
WHERE EmployeeID = 2)WHERE BookingID = 1;
UPDATE TabBooking SET ClientID = (SELECT REF(C) FROM TabClient C WHERE ClientID
= 3)WHERE BookingID = 1;
UPDATE TabBooking SET PaymentID = (SELECT REF(P) FROM TabPaymentType P WHERE
PaymentID = 1)WHERE BookingID = 1;
INSERT INTO TabBooking VALUES (2, NULL, NULL, NULL, NULL, NULL, 200.00, 40.00,
'15-APR-2023', 'Booked');
UPDATE TabBooking SET ArrangementID = (SELECT REF(E) FROM TabSecurityArrangement
E WHERE ArrangementID = 1)WHERE BookingID = 2;
```

```
UPDATE TabBooking SET FullTimeStaffID = (SELECT REF(E) FROM TabEmployees E WHERE
EmployeeID = 3)WHERE BookingID = 2;
UPDATE TabBooking SET individualsStaffID = (SELECT REF(E) FROM TabEmployees E
WHERE EmployeeID = 4)WHERE BookingID = 2;
UPDATE TabBooking SET ClientID = (SELECT REF(C) FROM TabClient C WHERE ClientID
= 5)WHERE BookingID = 2;
UPDATE TabBooking SET PaymentID = (SELECT REF(P) FROM TabPaymentType P WHERE
PaymentID = 1)WHERE BookingID = 2;
INSERT INTO TabBooking VALUES (3, NULL, NULL, NULL, NULL, NULL, 320.00, 60.00,
'10-JAN-2023', 'Advance Paid');
UPDATE TabBooking SET ArrangementID = (SELECT REF(E) FROM TabSecurityArrangement
E WHERE ArrangementID = 6)WHERE BookingID = 3;
UPDATE TabBooking SET FullTimeStaffID = (SELECT REF(E) FROM TabEmployees E WHERE
EmployeeID = 6)WHERE BookingID = 3;
UPDATE TabBooking SET individualsStaffID = (SELECT REF(E) FROM TabEmployees E
WHERE EmployeeID = 8)WHERE BookingID = 3;
UPDATE TabBooking SET ClientID = (SELECT REF(C) FROM TabClient C WHERE ClientID
= 6)WHERE BookingID = 3;
UPDATE TabBooking SET PaymentID = (SELECT REF(P) FROM TabPaymentType P WHERE
PaymentID = 2)WHERE BookingID = 3;
INSERT INTO TabBooking VALUES (4, NULL, NULL, NULL, NULL, NULL, 175.00, 30.00,
'12-FEB-2023', 'Booked');
UPDATE TabBooking SET ArrangementID = (SELECT REF(E) FROM TabSecurityArrangement
E WHERE ArrangementID = 2)WHERE BookingID = 4;
UPDATE TabBooking SET FullTimeStaffID = (SELECT REF(E) FROM TabEmployees E WHERE
EmployeeID = 10)WHERE BookingID = 4;
UPDATE TabBooking SET individualsStaffID = (SELECT REF(E) FROM TabEmployees E
WHERE EmployeeID = 2)WHERE BookingID = 4;
UPDATE TabBooking SET ClientID = (SELECT REF(C) FROM TabClient C WHERE ClientID
= 4)WHERE BookingID = 4;
UPDATE TabBooking SET PaymentID = (SELECT REF(P) FROM TabPaymentType P WHERE
PaymentID = 1)WHERE BookingID = 4;
INSERT INTO TabBooking VALUES (5, NULL, NULL, NULL, NULL, NULL, 350.00, 70.00,
'18-JAN-2023', 'Advance Paid');
UPDATE TabBooking SET ArrangementID = (SELECT REF(E) FROM TabSecurityArrangement
E WHERE ArrangementID = 1)WHERE BookingID = 5;
UPDATE TabBooking SET FullTimeStaffID = (SELECT REF(E) FROM TabEmployees E WHERE
EmployeeID = 3)WHERE BookingID = 5;
UPDATE TabBooking SET individualsStaffID = (SELECT REF(E) FROM TabEmployees E
WHERE EmployeeID = 4)WHERE BookingID = 5;
```

```
UPDATE TabBooking SET ClientID = (SELECT REF(C) FROM TabClient C WHERE ClientID
= 3)WHERE BookingID = 5;
UPDATE TabBooking SET PaymentID = (SELECT REF(P) FROM TabPaymentType P WHERE
PaymentID = 3)WHERE BookingID = 5;
INSERT INTO TabBooking VALUES (6, NULL, NULL, NULL, NULL, NULL, 300.00, 50.00,
'14-MAR-2023', 'Booked');
UPDATE TabBooking SET ArrangementID = (SELECT REF(E) FROM TabSecurityArrangement
E WHERE ArrangementID = 5)WHERE BookingID = 6;
UPDATE TabBooking SET FullTimeStaffID = (SELECT REF(E) FROM TabEmployees E WHERE
EmployeeID = 6)WHERE BookingID = 6;
UPDATE TabBooking SET individualsStaffID = (SELECT REF(E) FROM TabEmployees E
WHERE EmployeeID = 8)WHERE BookingID = 6;
UPDATE TabBooking SET ClientID = (SELECT REF(C) FROM TabClient C WHERE ClientID
= 5)WHERE BookingID = 6;
UPDATE TabBooking SET PaymentID = (SELECT REF(P) FROM TabPaymentType P WHERE
PaymentID = 1)WHERE BookingID = 6;
INSERT INTO TabBooking VALUES (7, NULL, NULL, NULL, NULL, NULL, 400.00, 85.00,
'11-FEB-2023', 'Advance Paid');
UPDATE TabBooking SET ArrangementID = (SELECT REF(E) FROM TabSecurityArrangement
E WHERE ArrangementID = 6)WHERE BookingID = 7;
UPDATE TabBooking SET FullTimeStaffID = (SELECT REF(E) FROM TabEmployees E WHERE
EmployeeID = 1)WHERE BookingID = 7;
UPDATE TabBooking SET individualsStaffID = (SELECT REF(E) FROM TabEmployees E
WHERE EmployeeID = 8)WHERE BookingID = 7;
UPDATE TabBooking SET ClientID = (SELECT REF(C) FROM TabClient C WHERE ClientID
= 6)WHERE BookingID = 7;
UPDATE TabBooking SET PaymentID = (SELECT REF(P) FROM TabPaymentType P WHERE
PaymentID = 2)WHERE BookingID = 7;
INSERT INTO TabBooking VALUES (8, NULL, NULL, NULL, NULL, NULL, 250.00, 40.00,
'16-FEB-2023', 'Booked');
UPDATE TabBooking SET ArrangementID = (SELECT REF(E) FROM TabSecurityArrangement
E WHERE ArrangementID = 2)WHERE BookingID = 8;
UPDATE TabBooking SET FullTimeStaffID = (SELECT REF(E) FROM TabEmployees E WHERE
EmployeeID = 3)WHERE BookingID = 8;
UPDATE TabBooking SET individualsStaffID = (SELECT REF(E) FROM TabEmployees E
WHERE EmployeeID = 4)WHERE BookingID = 8;
UPDATE TabBooking SET ClientID = (SELECT REF(C) FROM TabClient C WHERE ClientID
= 4)WHERE BookingID = 8;
UPDATE TabBooking SET PaymentID = (SELECT REF(P) FROM TabPaymentType P WHERE
PaymentID = 1)WHERE BookingID = 8;
```

```
INSERT INTO TabBooking VALUES (9, NULL, NULL, NULL, NULL, NULL, 500.00, 100.00,
'20-FEB-2023', 'Advance Paid');
UPDATE TabBooking SET ArrangementID = (SELECT REF(E) FROM TabSecurityArrangement
E WHERE ArrangementID = 1)WHERE BookingID = 9;
UPDATE TabBooking SET FullTimeStaffID = (SELECT REF(E) FROM TabEmployees E WHERE
EmployeeID = 10)WHERE BookingID = 9;
UPDATE TabBooking SET individualsStaffID = (SELECT REF(E) FROM TabEmployees E
WHERE EmployeeID = 2)WHERE BookingID = 9;
UPDATE TabBooking SET ClientID = (SELECT REF(C) FROM TabClient C WHERE ClientID
= 3)WHERE BookingID = 9;
UPDATE TabBooking SET PaymentID = (SELECT REF(P) FROM TabPaymentType P WHERE
PaymentID = 3)WHERE BookingID = 9;


INSERT INTO TabInvoice VALUES (1, NULL, NULL, 50.00, '18-APR-2023', 'Fully
Paid');
UPDATE TabInvoice SET BookingID = (SELECT REF(B) FROM TabBooking B WHERE
BookingID = 2)WHERE InvoiceID = 1;
UPDATE TabInvoice SET PaymentID = (SELECT REF(P) FROM TabPaymentType P WHERE
PaymentID = 1)WHERE InvoiceID = 1;
INSERT INTO TabInvoice VALUES (2, NULL, NULL, 40.00, '17-MAR-2023', 'Fully
Paid');
UPDATE TabInvoice SET BookingID = (SELECT REF(B) FROM TabBooking B WHERE
BookingID = 1)WHERE InvoiceID = 2;
UPDATE TabInvoice SET PaymentID = (SELECT REF(P) FROM TabPaymentType P WHERE
PaymentID = 2)WHERE InvoiceID = 2;
INSERT INTO TabInvoice VALUES (3, NULL, NULL, 25.00, '22-JAN-2023', 'Fully
Paid');
UPDATE TabInvoice SET BookingID = (SELECT REF(B) FROM TabBooking B WHERE
BookingID = 5)WHERE InvoiceID = 3;
UPDATE TabInvoice SET PaymentID = (SELECT REF(P) FROM TabPaymentType P WHERE
PaymentID = 3)WHERE InvoiceID = 3;
INSERT INTO TabInvoice VALUES (4, NULL, NULL, 70.00, '27-FEB-2023', 'Fully
Paid');
UPDATE TabInvoice SET BookingID = (SELECT REF(B) FROM TabBooking B WHERE
BookingID = 9)WHERE InvoiceID = 4;
UPDATE TabInvoice SET PaymentID = (SELECT REF(P) FROM TabPaymentType P WHERE
PaymentID = 1)WHERE InvoiceID = 4;
INSERT INTO TabInvoice VALUES (5, NULL, NULL, 50.00, '15-FEB-2023', 'Fully
Paid');
```

```
UPDATE TabInvoice SET BookingID = (SELECT REF(B) FROM TabBooking B WHERE
BookingID = 4)WHERE InvoiceID = 5;
UPDATE TabInvoice SET PaymentID = (SELECT REF(P) FROM TabPaymentType P WHERE
PaymentID = 2)WHERE InvoiceID = 5;
INSERT INTO TabInvoice VALUES (6, NULL, NULL, 30.00, '19-FEB-2023', 'Fully
Paid');
UPDATE TabInvoice SET BookingID = (SELECT REF(B) FROM TabBooking B WHERE
BookingID = 7)WHERE InvoiceID = 6;
UPDATE TabInvoice SET PaymentID = (SELECT REF(P) FROM TabPaymentType P WHERE
PaymentID = 3)WHERE InvoiceID = 6;
INSERT INTO TabInvoice VALUES (7, NULL, NULL, 60.00, '17-FEB-2023', 'Fully
Paid');
UPDATE TabInvoice SET BookingID = (SELECT REF(B) FROM TabBooking B WHERE
BookingID = 8)WHERE InvoiceID = 7;
UPDATE TabInvoice SET PaymentID = (SELECT REF(P) FROM TabPaymentType P WHERE
PaymentID = 1)WHERE InvoiceID = 7;
INSERT INTO TabInvoice VALUES (8, NULL, NULL, 40.00, '25-JAN-2023', 'Fully
Paid');
UPDATE TabInvoice SET BookingID = (SELECT REF(B) FROM TabBooking B WHERE
BookingID = 3)WHERE InvoiceID = 8;
UPDATE TabInvoice SET PaymentID = (SELECT REF(P) FROM TabPaymentType P WHERE
PaymentID = 2)WHERE InvoiceID = 8;
INSERT INTO TabInvoice VALUES (9, NULL, NULL, 35.00, '20-MAR-2023', 'Fully
Paid');
UPDATE TabInvoice SET BookingID = (SELECT REF(B) FROM TabBooking B WHERE
BookingID = 6)WHERE InvoiceID = 9;
UPDATE TabInvoice SET PaymentID = (SELECT REF(P) FROM TabPaymentType P WHERE
PaymentID = 3)WHERE InvoiceID = 9;
```

2) SQL Output (e.g., SPOOL file contents or output you got when you executed your above SQL code, this should show the SQL code as well as its output / result). Make sure the output is NOT TEXT but rather images or screenshots. If missing or inadequate, then up to 2 marks will be deducted from the above 10 marks for this part.

```
W22046071 >
W22046071 > INSERT INTO TabEmployeeType VALUES (1, 'Full Time');

1 row created.

W22046071 > INSERT INTO TabEmployeeType VALUES (2, 'Individuals');

1 row created.

W22046071 > INSERT INTO TabEmployeeType VALUES (3, 'Managers');

1 row created.

W22046071 > INSERT INTO TabEmployeeType VALUES (4, 'Individuals');

1 row created.

W22046071 > INSERT INTO TabEmployeeType VALUES (5, 'Full Time');

1 row created.

W22046071 > INSERT INTO TabEmployeeType VALUES (6, 'Managers');

1 row created.

W22046071 > INSERT INTO TabEmployeeType VALUES (7, 'Full Time');

1 row created.

W22046071 > INSERT INTO TabEmployeeType VALUES (8, 'Full Time');

1 row created.

W22046071 > INSERT INTO TabEmployeeType VALUES (9, 'Managers');

1 row created.

W22046071 > INSERT INTO TabEmployeeType VALUES (10, 'Individuals');

1 row created.

W22046071 >
```

```
W22046071 >
W22046071 > INSERT INTO TabEmployees VALUES (1, NeEmployeeType(TEmployeeType(1, 'Full Time')), 'Yakhubji', 123456789, 'yakhubji@gmail.com', 'MBA', 5);

1 row created.

W22046071 > INSERT INTO TabEmployees VALUES (2, NeEmployeeType(TEmployeeType(2, 'Individuals')), 'Shahid', 987654321, 'shahid@gmail.com', 'B.Sc.', 10);

1 row created.

W22046071 > INSERT INTO TabEmployees VALUES (3, NeEmployeeType(TEmployeeType(1, 'Full Time')), 'Irfan', 789456321, 'irfan@gmail.com', 'M.Tech', 8);

1 row created.

W22046071 > INSERT INTO TabEmployees VALUES (4, NeEmployeeType(TEmployeeType(2, 'Individuals')), 'Arif', 124567890, 'arif@gmail.com', 'M.Sc.', 3);

1 row created.

W22046071 > INSERT INTO TabEmployees VALUES (5, NeEmployeeType(TEmployeeType(3, 'Managers')), 'Inthiyaz', 321456709, 'inthiyaz@gmail.com', 'B.Tech', 4);

1 row created.

W22046071 >
```

```
W22046071 >
W22046071 > INSERT INTO TabEmployees VALUES (6, NeEmployeeType(TEmployeeType(1, 'Full Time')), 'Ghouse', 543219876, 'ghouse@gmail.com', 'MBA', 2);

1 row created.

W22046071 > INSERT INTO TabEmployees VALUES (7, NeEmployeeType(TEmployeeType(3, 'Managers')), 'Jeelan', 908765432, 'jeelan@gmail.com', 'B.Sc.', 6);

1 row created.

W22046071 > INSERT INTO TabEmployees VALUES (8, NeEmployeeType(TEmployeeType(2, 'Individuals')), 'Vahid', 765432109, 'vahid@gmail.com', 'M.Sc.', 7);

1 row created.

W22046071 > INSERT INTO TabEmployees VALUES (9, NeEmployeeType(TEmployeeType(3, 'Managers')), 'Mainu', 876543210, 'mainu@gmail.com', 'B.Tech', 9);

1 row created.

W22046071 > INSERT INTO TabEmployees VALUES (10, NeEmployeeType(TEmployeeType(1, 'Full Time')), 'Kasif', 654321987, 'kasif@gmail.com', 'MBA', 1);

1 row created.

W22046071 >
```

```
W22046071 >
W22046071 > INSERT INTO TabEmployeeAvailability VALUES (1,NULL, '01-JUN-2023', '19-JULY-2023', 'Vacation');

1 row created.

W22046071 > UPDATE TabEmployeeAvailability SET EmployeeID = (SELECT REF(E) FROM TabEmployees E WHERE EmployeeID = 4)WHERE MasterCalendarID = 1;

1 row updated.

W22046071 > INSERT INTO TabEmployeeAvailability VALUES (2,NULL, '20-MAR-2023', '05-APR-2023', 'Not available');

1 row created.

W22046071 > UPDATE TabEmployeeAvailability SET EmployeeID = (SELECT REF(E) FROM TabEmployees E WHERE EmployeeID = 2)WHERE MasterCalendarID = 2;

1 row updated.

W22046071 > INSERT INTO TabEmployeeAvailability VALUES (3,NULL, '15-MAY-2023', '23-JUN-2023', 'Available');

1 row created.

W22046071 > UPDATE TabEmployeeAvailability SET EmployeeID = (SELECT REF(E) FROM TabEmployees E WHERE EmployeeID = 1)WHERE MasterCalendarID = 3;

1 row updated.

W22046071 >
```

```
W22046071 >
W22046071 > INSERT INTO TabEmployeeAvailability VALUES (4,NULL, '05-SEP-2023', '04-OCT-2023', 'Working');

1 row created.

W22046071 > UPDATE TabEmployeeAvailability SET EmployeeID = (SELECT REF(E) FROM TabEmployees E WHERE EmployeeID = 1)WHERE MasterCalendarID = 4;

1 row updated.

W22046071 > INSERT INTO TabEmployeeAvailability VALUES (5,NULL, '11-JUN-2023', '07-JULY-2023', 'Vacation');

1 row created.

W22046071 > UPDATE TabEmployeeAvailability SET EmployeeID = (SELECT REF(E) FROM TabEmployees E WHERE EmployeeID = 5)WHERE MasterCalendarID = 5;

1 row updated.

W22046071 >
```

```
W22046071 >
W22046071 > INSERT INTO TabClient VALUES (1, 'Shaiksha', '36 Rowland Rd', 07061188519, 'shaiksha@gmail.com');

1 row created.

W22046071 > INSERT INTO TabClient VALUES (2, 'Shahul', '41 Stone Cellar Road', 07017170994, 'shahul@gmail.com');

1 row created.

W22046071 > INSERT INTO TabClient VALUES (3, 'Nikith', '25 Withers Close', 0708544795, 'davewilson@gmail.com');

1 row created.

W22046071 > INSERT INTO TabClient VALUES (4, 'Aadhira', '59 Henley Road', 07858417732, 'aadhira@gmail.com');

1 row created.

W22046071 > INSERT INTO TabClient VALUES (5, 'Jagath', '38 Earls Avenue', 07069616402, 'jagath@gmail.com');

1 row created.

W22046071 > INSERT INTO TabClient VALUES (6, 'Loshith', '45 Park End St', 07038550069, 'loshith@gmail.com');

1 row created.

W22046071 > INSERT INTO TabClient VALUES (7, 'Nithin', '16 Fulford Road', 07917388035, 'nithin@gmail.com');

1 row created.

W22046071 > INSERT INTO TabClient VALUES (8, 'Saswin', '55 Tonbridge Rd', 07933281321, 'saswin@gmail.com');

1 row created.

W22046071 >
```

```
W22046071 >
W22046071 > INSERT INTO TabSiteType VALUES (1, 'Property');

1 row created.

W22046071 > INSERT INTO TabSiteType VALUES (2, 'Business Premises');

1 row created.

W22046071 > INSERT INTO TabSiteType VALUES (3, 'Construction');

1 row created.

W22046071 > INSERT INTO TabSiteType VALUES (4, 'Industrial');

1 row created.

W22046071 > INSERT INTO TabSiteType VALUES (5, 'Private Individual');

1 row created.

W22046071 > INSERT INTO TabSiteType VALUES (6, 'Private Business');

1 row created.

W22046071 > INSERT INTO TabSiteType VALUES (7, 'Personal');

1 row created.

W22046071 > INSERT INTO TabSiteType VALUES (8, 'Commercial');

1 row created.

W22046071 >
```

```
W22046071 >
W22046071 > INSERT INTO TabSite VALUES (1, NULL, NULL, 'Wonderful Homes', 'Newcastle upon Tyne');

1 row created.

W22046071 > UPDATE TabSite SET ClientID = (SELECT REF(C) FROM TabClient C WHERE ClientID = 1)WHERE SiteID = 1;

1 row updated.

W22046071 > UPDATE TabSite SET SiteTypeID = (SELECT REF(ST) FROM TabSiteType ST WHERE SiteTypeID = 1)WHERE SiteID = 1;

1 row updated.

W22046071 > INSERT INTO TabSite VALUES (2, NULL, NULL, 'Luxury Ohio', 'Newcastle upon Tyne');

1 row created.

W22046071 > UPDATE TabSite SET ClientID = (SELECT REF(C) FROM TabClient C WHERE ClientID = 1)WHERE SiteID = 2;

1 row updated.

W22046071 > UPDATE TabSite SET SiteTypeID = (SELECT REF(ST) FROM TabSiteType ST WHERE SiteTypeID = 2)WHERE SiteID = 2;

1 row updated.

W22046071 > INSERT INTO TabSite VALUES (3, NULL, NULL, 'Texas Waterfront', 'Newcastle upon Tyne');

1 row created.

W22046071 > UPDATE TabSite SET ClientID = (SELECT REF(C) FROM TabClient C WHERE ClientID = 2)WHERE SiteID = 3;

1 row updated.

W22046071 > UPDATE TabSite SET SiteTypeID = (SELECT REF(ST) FROM TabSiteType ST WHERE SiteTypeID = 3)WHERE SiteID = 3;

1 row updated.

W22046071 >
```

```
W22046071 >
W22046071 > INSERT INTO TabSite VALUES (4, NULL, NULL, 'Sarah Sells Homes', 'Newcastle upon Tyne');

1 row created.

W22046071 > UPDATE TabSite SET ClientID = (SELECT REF(C) FROM TabClient C WHERE ClientID = 3)WHERE SiteID = 4;

1 row updated.

W22046071 > UPDATE TabSite SET SiteTypeID = (SELECT REF(ST) FROM TabSiteType ST WHERE SiteTypeID = 4)WHERE SiteID = 4;

1 row updated.

W22046071 > INSERT INTO TabSite VALUES (5, NULL, NULL, 'Tom Smith Realty', 'Newcastle upon Tyne');

1 row created.

W22046071 > UPDATE TabSite SET ClientID = (SELECT REF(C) FROM TabClient C WHERE ClientID = 4)WHERE SiteID = 5;

1 row updated.

W22046071 > UPDATE TabSite SET SiteTypeID = (SELECT REF(ST) FROM TabSiteType ST WHERE SiteTypeID = 1)WHERE SiteID = 5;

1 row updated.

W22046071 > INSERT INTO TabSite VALUES (6, NULL, NULL, 'Johnand Jacob Realty', '7 Whatlington Road');

1 row created.

W22046071 > UPDATE TabSite SET ClientID = (SELECT REF(C) FROM TabClient C WHERE ClientID = 5)WHERE SiteID = 6;

1 row updated.

W22046071 > UPDATE TabSite SET SiteTypeID = (SELECT REF(ST) FROM TabSiteType ST WHERE SiteTypeID = 2)WHERE SiteID = 6;

1 row updated.

W22046071 >
```

```
W22046071 >
W22046071 > INSERT INTO TabSite VALUES (7, NULL, NULL, 'Anderson Realty', '50 Withers Close');

1 row created.

W22046071 > UPDATE TabSite SET ClientID = (SELECT REF(C) FROM TabClient C WHERE ClientID = 6)WHERE SiteID = 7;

1 row updated.

W22046071 > UPDATE TabSite SET SiteTypeID = (SELECT REF(ST) FROM TabSiteType ST WHERE SiteTypeID = 3)WHERE SiteID = 7;

1 row updated.

W22046071 > INSERT INTO TabSite VALUES (8, NULL, NULL, 'BuyHome4 Family', '80 Nether Crescent');

1 row created.

W22046071 > UPDATE TabSite SET ClientID = (SELECT REF(C) FROM TabClient C WHERE ClientID = 7)WHERE SiteID = 8;

1 row updated.

W22046071 > UPDATE TabSite SET SiteTypeID = (SELECT REF(ST) FROM TabSiteType ST WHERE SiteTypeID = 4)WHERE SiteID = 8;

1 row updated.

W22046071 >
W22046071 >
```

```
W22046071 >
W22046071 > INSERT INTO TabSecurityArrangement VALUES (1, NULL, NULL, 'Weekdays only', 'Security for the office', 80, '15-MAR-2023', '01-APR-2023', 160.00);

1 row created.

W22046071 > UPDATE TabSecurityArrangement SET FullTimeStaffID = (SELECT REF(E) FROM TabEmployees E WHERE EmployeeID = 5)WHERE ArrangementID = 1;

1 row updated.

W22046071 > UPDATE TabSecurityArrangement SET SiteID = (SELECT REF(S) FROM TabSite S WHERE SiteID = 3)WHERE ArrangementID = 1;

1 row updated.

W22046071 > INSERT INTO TabSecurityArrangement VALUES (2, NULL, NULL, 'Weekend only', 'Security for the entrance', 48, '01-FEB-2023', '15-APR-2023', 240.00);

1 row created.

W22046071 > UPDATE TabSecurityArrangement SET FullTimeStaffID = (SELECT REF(E) FROM TabEmployees E WHERE EmployeeID = 7)WHERE ArrangementID = 2;

1 row updated.

W22046071 > UPDATE TabSecurityArrangement SET SiteID = (SELECT REF(S) FROM TabSite S WHERE SiteID = 4)WHERE ArrangementID = 2;

1 row updated.

W22046071 > INSERT INTO TabSecurityArrangement VALUES (3, NULL, NULL, 'Weekdays and weekend', 'Security for the parking lot', 128, '25-JUL-2023', '20-AUG-2023', 120.00);

1 row created.

W22046071 > UPDATE TabSecurityArrangement SET FullTimeStaffID = (SELECT REF(E) FROM TabEmployees E WHERE EmployeeID = 9)WHERE ArrangementID = 3;

1 row updated.

W22046071 > UPDATE TabSecurityArrangement SET SiteID = (SELECT REF(S) FROM TabSite S WHERE SiteID = 2)WHERE ArrangementID = 3;

1 row updated.

W22046071 >
```

```
W22046071 >
W22046071 > INSERT INTO TabSecurityArrangement VALUES (4, NULL, NULL, 'Weekdays only', 'Security for the main gate', 80, '10-MAY-2023', '15-JUN-2023', 160.00);

1 row created.

W22046071 > UPDATE TabSecurityArrangement SET FullTimeStaffID = (SELECT REF(E) FROM TabEmployees E WHERE EmployeeID = 5)WHERE ArrangementID = 4;

1 row updated.

W22046071 > UPDATE TabSecurityArrangement SET SiteID = (SELECT REF(S) FROM TabSite S WHERE SiteID = 1)WHERE ArrangementID = 4;

1 row updated.

W22046071 > INSERT INTO TabSecurityArrangement VALUES (5, NULL, NULL, 'Weekend only', 'Security for the reception area', 48, '20-APR-2023', '01-APR-2023', 185.00);

1 row created.

W22046071 > UPDATE TabSecurityArrangement SET FullTimeStaffID = (SELECT REF(E) FROM TabEmployees E WHERE EmployeeID = 7)WHERE ArrangementID = 5;

1 row updated.

W22046071 > UPDATE TabSecurityArrangement SET SiteID = (SELECT REF(S) FROM TabSite S WHERE SiteID = 3)WHERE ArrangementID = 5;

1 row updated.

W22046071 > INSERT INTO TabSecurityArrangement VALUES (6, NULL, NULL, 'Weekdays and weekend', 'Security for the storeroom', 128, '12-FEB-2023', '15-MAR-2023', 196.00);

1 row created.

W22046071 > UPDATE TabSecurityArrangement SET FullTimeStaffID = (SELECT REF(E) FROM TabEmployees E WHERE EmployeeID = 9)WHERE ArrangementID = 6;

1 row updated.

W22046071 > UPDATE TabSecurityArrangement SET SiteID = (SELECT REF(S) FROM TabSite S WHERE SiteID = 4)WHERE ArrangementID = 6;

1 row updated.

W22046071 >
```

```
W22046071 >
W22046071 > INSERT INTO TabSecurityArrangement VALUES (7, NULL, NULL, 'Weekdays only', 'Security for the building', 80, '25-MAR-2023', '25-MAR-2023', 196.00);

1 row created.

W22046071 > UPDATE TabSecurityArrangement SET FullTimeStaffID = (SELECT REF(E) FROM TabEmployees E WHERE EmployeeID = 7)WHERE ArrangementID = 7;

1 row updated.

W22046071 > UPDATE TabSecurityArrangement SET SiteID = (SELECT REF(S) FROM TabSite S WHERE SiteID = 1)WHERE ArrangementID = 7;

1 row updated.

W22046071 > INSERT INTO TabSecurityArrangement VALUES (8, NULL, NULL, 'Weekend only', 'Security for the conference room', 48, '10-JAN-2023', '10-JAN-2023', 178.00);

1 row created.

W22046071 > UPDATE TabSecurityArrangement SET FullTimeStaffID = (SELECT REF(E) FROM TabEmployees E WHERE EmployeeID = 5)WHERE ArrangementID = 8;

1 row updated.

W22046071 > UPDATE TabSecurityArrangement SET SiteID = (SELECT REF(S) FROM TabSite S WHERE SiteID = 3)WHERE ArrangementID = 8;

1 row updated.

W22046071 >
```

```
W22046071 >
W22046071 > INSERT INTO TabPaymentType VALUES (1, 'Credit card');

1 row created.

W22046071 > INSERT INTO TabPaymentType VALUES (2, 'Cash');

1 row created.

W22046071 > INSERT INTO TabPaymentType VALUES (3, 'Cheque');

1 row created.

W22046071 >
```

```
W22046071 >
W22046071 > INSERT INTO TabBooking VALUES (1, NULL, NULL, NULL, NULL, NULL, 150.00, 25.00, '14-MAR-2023', 'Advance Paid');

1 row created.

W22046071 > UPDATE TabBooking SET ArrangementID = (SELECT REF(E) FROM TabSecurityArrangement E WHERE ArrangementID = 4)WHERE BookingID = 1;

1 row updated.

W22046071 > UPDATE TabBooking SET FullTimeStaffID = (SELECT REF(E) FROM TabEmployees E WHERE EmployeeID = 1)WHERE BookingID = 1;

1 row updated.

W22046071 > UPDATE TabBooking SET individualsStaffID = (SELECT REF(E) FROM TabEmployees E WHERE EmployeeID = 2)WHERE BookingID = 1;

1 row updated.

W22046071 > UPDATE TabBooking SET ClientID = (SELECT REF(C) FROM TabClient C WHERE ClientID = 3)WHERE BookingID = 1;

1 row updated.

W22046071 > UPDATE TabBooking SET PaymentID = (SELECT REF(P) FROM TabPaymentType P WHERE PaymentID = 1)WHERE BookingID = 1;

1 row updated.
```

```
W22046071 > INSERT INTO TabBooking VALUES (2, NULL, NULL, NULL, NULL, NULL, 200.00, 40.00, '15-APR-2023', 'Booked');

1 row created.

W22046071 > UPDATE TabBooking SET ArrangementID = (SELECT REF(E) FROM TabSecurityArrangement E WHERE ArrangementID = 1)WHERE BookingID = 2;

1 row updated.

W22046071 > UPDATE TabBooking SET FullTimeStaffID = (SELECT REF(E) FROM TabEmployees E WHERE EmployeeID = 3)WHERE BookingID = 2;

1 row updated.

W22046071 > UPDATE TabBooking SET individualsStaffID = (SELECT REF(E) FROM TabEmployees E WHERE EmployeeID = 4)WHERE BookingID = 2;

1 row updated.

W22046071 > UPDATE TabBooking SET ClientID = (SELECT REF(C) FROM TabClient C WHERE ClientID = 5)WHERE BookingID = 2;

1 row updated.

W22046071 > UPDATE TabBooking SET PaymentID = (SELECT REF(P) FROM TabPaymentType P WHERE PaymentID = 1)WHERE BookingID = 2;

1 row updated.

W22046071 >
```

```
W22046071 >
W22046071 > INSERT INTO TabBooking VALUES (3, NULL, NULL, NULL, NULL, NULL, 320.00, 60.00, '10-JAN-2023', 'Advance Paid');

1 row created.

W22046071 > UPDATE TabBooking SET ArrangementID = (SELECT REF(E) FROM TabSecurityArrangement E WHERE ArrangementID = 6)WHERE BookingID = 3;

1 row updated.

W22046071 > UPDATE TabBooking SET FullTimeStaffID = (SELECT REF(E) FROM TabEmployees E WHERE EmployeeID = 6)WHERE BookingID = 3;

1 row updated.

W22046071 > UPDATE TabBooking SET individualsStaffID = (SELECT REF(E) FROM TabEmployees E WHERE EmployeeID = 8)WHERE BookingID = 3;

1 row updated.

W22046071 > UPDATE TabBooking SET ClientID = (SELECT REF(C) FROM TabClient C WHERE ClientID = 6)WHERE BookingID = 3;

1 row updated.

W22046071 > UPDATE TabBooking SET PaymentID = (SELECT REF(P) FROM TabPaymentType P WHERE PaymentID = 2)WHERE BookingID = 3;

1 row updated.

W22046071 >
```

```
W22046071 >
W22046071 > INSERT INTO TabBooking VALUES (4, NULL, NULL, NULL, NULL, NULL, 175.00, 30.00, '12-FEB-2023', 'Booked');

1 row created.

W22046071 > UPDATE TabBooking SET ArrangementID = (SELECT REF(E) FROM TabSecurityArrangement E WHERE ArrangementID = 2)WHERE BookingID = 4;

1 row updated.

W22046071 > UPDATE TabBooking SET FullTimeStaffID = (SELECT REF(E) FROM TabEmployees E WHERE EmployeeID = 10)WHERE BookingID = 4;

1 row updated.

W22046071 > UPDATE TabBooking SET individualsStaffID = (SELECT REF(E) FROM TabEmployees E WHERE EmployeeID = 2)WHERE BookingID = 4;

1 row updated.

W22046071 > UPDATE TabBooking SET ClientID = (SELECT REF(C) FROM TabClient C WHERE ClientID = 4)WHERE BookingID = 4;

1 row updated.

W22046071 > UPDATE TabBooking SET PaymentID = (SELECT REF(P) FROM TabPaymentType P WHERE PaymentID = 1)WHERE BookingID = 4;

1 row updated.

W22046071 >
W22046071 >
```

```
W22046071 >
W22046071 > INSERT INTO TabBooking VALUES (5, NULL, NULL, NULL, NULL, NULL, 350.00, 70.00, '18-JAN-2023', 'Advance Paid');

1 row created.

W22046071 > UPDATE TabBooking SET ArrangementID = (SELECT REF(E) FROM TabSecurityArrangement E WHERE ArrangementID = 1)WHERE BookingID = 5;

1 row updated.

W22046071 > UPDATE TabBooking SET FullTimeStaffID = (SELECT REF(E) FROM TabEmployees E WHERE EmployeeID = 3)WHERE BookingID = 5;

1 row updated.

W22046071 > UPDATE TabBooking SET individualsStaffID = (SELECT REF(E) FROM TabEmployees E WHERE EmployeeID = 4)WHERE BookingID = 5;

1 row updated.

W22046071 > UPDATE TabBooking SET ClientID = (SELECT REF(C) FROM TabClient C WHERE ClientID = 3)WHERE BookingID = 5;

1 row updated.

W22046071 > UPDATE TabBooking SET PaymentID = (SELECT REF(P) FROM TabPaymentType P WHERE PaymentID = 3)WHERE BookingID = 5;

1 row updated.

W22046071 >
W22046071 >
```

```
W22046071 >
W22046071 > INSERT INTO TabBooking VALUES (6, NULL, NULL, NULL, NULL, NULL, 300.00, 50.00, '14-MAR-2023', 'Booked');

1 row created.

W22046071 > UPDATE TabBooking SET ArrangementID = (SELECT REF(E) FROM TabSecurityArrangement E WHERE ArrangementID = 5)WHERE BookingID = 6;

1 row updated.

W22046071 > UPDATE TabBooking SET FullTimeStaffID = (SELECT REF(E) FROM TabEmployees E WHERE EmployeeID = 6)WHERE BookingID = 6;

1 row updated.

W22046071 > UPDATE TabBooking SET individualsStaffID = (SELECT REF(E) FROM TabEmployees E WHERE EmployeeID = 8)WHERE BookingID = 6;

1 row updated.

W22046071 > UPDATE TabBooking SET ClientID = (SELECT REF(C) FROM TabClient C WHERE ClientID = 5)WHERE BookingID = 6;

1 row updated.

W22046071 > UPDATE TabBooking SET PaymentID = (SELECT REF(P) FROM TabPaymentType P WHERE PaymentID = 1)WHERE BookingID = 6;

1 row updated.

W22046071 >
W22046071 >
W22046071 >
```

```
W22046071 >
W22046071 > INSERT INTO TabBooking VALUES (7, NULL, NULL, NULL, NULL, NULL, 400.00, 85.00, '11-FEB-2023', 'Advance Paid');

1 row created.

W22046071 > UPDATE TabBooking SET ArrangementID = (SELECT REF(E) FROM TabSecurityArrangement E WHERE ArrangementID = 6)WHERE BookingID = 7;

1 row updated.

W22046071 > UPDATE TabBooking SET FullTimeStaffID = (SELECT REF(E) FROM TabEmployees E WHERE EmployeeID = 1)WHERE BookingID = 7;

1 row updated.

W22046071 > UPDATE TabBooking SET individualsStaffID = (SELECT REF(E) FROM TabEmployees E WHERE EmployeeID = 8)WHERE BookingID = 7;

1 row updated.

W22046071 > UPDATE TabBooking SET ClientID = (SELECT REF(C) FROM TabClient C WHERE ClientID = 6)WHERE BookingID = 7;

1 row updated.

W22046071 > UPDATE TabBooking SET PaymentID = (SELECT REF(P) FROM TabPaymentType P WHERE PaymentID = 2)WHERE BookingID = 7;

1 row updated.

W22046071 >
W22046071 >
W22046071 >
```

```
W22046071 >
W22046071 > INSERT INTO TabBooking VALUES (8, NULL, NULL, NULL, NULL, NULL, 250.00, 40.00, '16-FEB-2023', 'Booked');

1 row created.

W22046071 > UPDATE TabBooking SET ArrangementID = (SELECT REF(E) FROM TabSecurityArrangement E WHERE ArrangementID = 2)WHERE BookingID = 8;

1 row updated.

W22046071 > UPDATE TabBooking SET FullTimeStaffID = (SELECT REF(E) FROM TabEmployees E WHERE EmployeeID = 3)WHERE BookingID = 8;

1 row updated.

W22046071 > UPDATE TabBooking SET individualsStaffID = (SELECT REF(E) FROM TabEmployees E WHERE EmployeeID = 4)WHERE BookingID = 8;

1 row updated.

W22046071 > UPDATE TabBooking SET ClientID = (SELECT REF(C) FROM TabClient C WHERE ClientID = 4)WHERE BookingID = 8;

1 row updated.

W22046071 > UPDATE TabBooking SET PaymentID = (SELECT REF(P) FROM TabPaymentType P WHERE PaymentID = 1)WHERE BookingID = 8;

1 row updated.

W22046071 >
```

```
W22046071 >
W22046071 > INSERT INTO TabBooking VALUES (9, NULL, NULL, NULL, NULL, NULL, 500.00, 100.00, '20-FEB-2023', 'Advance Paid');

1 row created.

W22046071 > UPDATE TabBooking SET ArrangementID = (SELECT REF(E) FROM TabSecurityArrangement E WHERE ArrangementID = 1)WHERE BookingID = 9;

1 row updated.

W22046071 > UPDATE TabBooking SET FullTimeStaffID = (SELECT REF(E) FROM TabEmployees E WHERE EmployeeID = 10)WHERE BookingID = 9;

1 row updated.

W22046071 > UPDATE TabBooking SET individualsStaffID = (SELECT REF(E) FROM TabEmployees E WHERE EmployeeID = 2)WHERE BookingID = 9;

1 row updated.

W22046071 > UPDATE TabBooking SET ClientID = (SELECT REF(C) FROM TabClient C WHERE ClientID = 3)WHERE BookingID = 9;

1 row updated.

W22046071 > UPDATE TabBooking SET PaymentID = (SELECT REF(P) FROM TabPaymentType P WHERE PaymentID = 3)WHERE BookingID = 9;

1 row updated.

W22046071 >
W22046071 >
```

```
W22046071 >
W22046071 > INSERT INTO TabInvoice VALUES (1, NULL, NULL, 50.00, '18-APR-2023', 'Fully Paid');

1 row created.

W22046071 > UPDATE TabInvoice SET BookingID = (SELECT REF(B) FROM TabBooking B WHERE BookingID = 2)WHERE InvoiceID = 1;

1 row updated.

W22046071 > UPDATE TabInvoice SET PaymentID = (SELECT REF(P) FROM TabPaymentType P WHERE PaymentID = 1)WHERE InvoiceID = 1;

1 row updated.

W22046071 > INSERT INTO TabInvoice VALUES (2, NULL, NULL, 40.00, '17-MAR-2023', 'Fully Paid');

1 row created.

W22046071 > UPDATE TabInvoice SET BookingID = (SELECT REF(B) FROM TabBooking B WHERE BookingID = 1)WHERE InvoiceID = 2;

1 row updated.

W22046071 > UPDATE TabInvoice SET PaymentID = (SELECT REF(P) FROM TabPaymentType P WHERE PaymentID = 2)WHERE InvoiceID = 2;

1 row updated.

W22046071 > INSERT INTO TabInvoice VALUES (3, NULL, NULL, 25.00, '22-JAN-2023', 'Fully Paid');

1 row created.

W22046071 > UPDATE TabInvoice SET BookingID = (SELECT REF(B) FROM TabBooking B WHERE BookingID = 5)WHERE InvoiceID = 3;

1 row updated.

W22046071 > UPDATE TabInvoice SET PaymentID = (SELECT REF(P) FROM TabPaymentType P WHERE PaymentID = 3)WHERE InvoiceID = 3;

1 row updated.

W22046071 >
W22046071 >
```

```
W22046071 >
W22046071 > INSERT INTO TabInvoice VALUES (4, NULL, NULL, 70.00, '27-FEB-2023', 'Fully Paid');

1 row created.

W22046071 > UPDATE TabInvoice SET BookingID = (SELECT REF(B) FROM TabBooking B WHERE BookingID = 9)WHERE InvoiceID = 4;

1 row updated.

W22046071 > UPDATE TabInvoice SET PaymentID = (SELECT REF(P) FROM TabPaymentType P WHERE PaymentID = 1)WHERE InvoiceID = 4;

1 row updated.

W22046071 > INSERT INTO TabInvoice VALUES (5, NULL, NULL, 50.00, '15-FEB-2023', 'Fully Paid');

1 row created.

W22046071 > UPDATE TabInvoice SET BookingID = (SELECT REF(B) FROM TabBooking B WHERE BookingID = 4)WHERE InvoiceID = 5;

1 row updated.

W22046071 > UPDATE TabInvoice SET PaymentID = (SELECT REF(P) FROM TabPaymentType P WHERE PaymentID = 2)WHERE InvoiceID = 5;

1 row updated.

W22046071 > INSERT INTO TabInvoice VALUES (6, NULL, NULL, 30.00, '19-FEB-2023', 'Fully Paid');

1 row created.

W22046071 > UPDATE TabInvoice SET BookingID = (SELECT REF(B) FROM TabBooking B WHERE BookingID = 7)WHERE InvoiceID = 6;

1 row updated.

W22046071 > UPDATE TabInvoice SET PaymentID = (SELECT REF(P) FROM TabPaymentType P WHERE PaymentID = 3)WHERE InvoiceID = 6;

1 row updated.

W22046071 >
W22046071 >
```

```
W22046071 >
W22046071 > INSERT INTO TabInvoice VALUES (7, NULL, NULL, 60.00, '17-FEB-2023', 'Fully Paid');

1 row created.

W22046071 > UPDATE TabInvoice SET BookingID = (SELECT REF(B) FROM TabBooking B WHERE BookingID = 8)WHERE InvoiceID = 7;

1 row updated.

W22046071 > UPDATE TabInvoice SET PaymentID = (SELECT REF(P) FROM TabPaymentType P WHERE PaymentID = 1)WHERE InvoiceID = 7;

1 row updated.

W22046071 > INSERT INTO TabInvoice VALUES (8, NULL, NULL, 40.00, '25-JAN-2023', 'Fully Paid');

1 row created.

W22046071 > UPDATE TabInvoice SET BookingID = (SELECT REF(B) FROM TabBooking B WHERE BookingID = 3)WHERE InvoiceID = 8;

1 row updated.

W22046071 > UPDATE TabInvoice SET PaymentID = (SELECT REF(P) FROM TabPaymentType P WHERE PaymentID = 2)WHERE InvoiceID = 8;

1 row updated.

W22046071 > INSERT INTO TabInvoice VALUES (9, NULL, NULL, 35.00, '20-MAR-2023', 'Fully Paid');

1 row created.

W22046071 > UPDATE TabInvoice SET BookingID = (SELECT REF(B) FROM TabBooking B WHERE BookingID = 6)WHERE InvoiceID = 9;

1 row updated.

W22046071 > UPDATE TabInvoice SET PaymentID = (SELECT REF(P) FROM TabPaymentType P WHERE PaymentID = 3)WHERE InvoiceID = 9;

1 row updated.

W22046071 >
```

C) Writing Queries on the OR version of the subset of the NorthEastNinjas Database (10 marks)

For this part, you must use PL/SQL procedures to query / display the required data. Answer the following questions after populating the object-relational database (after completing part 3.2):

Q1) Display details of clients, their sites and whatever security arrangements they have for their sites in Newcastle upon Tyne for the months of January, February and March 2023. The output should not be tabular but structured like a report on client by client basis, e.g., details of each client, followed by their sites and security arrangements in some chronological order.

1) Answer Part 1 C Query a: Provide the PL/SQL code (5 marks):

```
SELECT c.Name, s.SiteName, sa.ArrangementType, sa.Description, sa.StartDate,
sa.EndDate
FROM TabClient c
INNER JOIN TabSite s ON c.ClientID = s.ClientID.ClientID
INNER JOIN TabSecurityArrangement sa ON s.SiteID = sa.SiteID.SiteID
WHERE s.Address = 'Newcastle upon Tyne'
AND (sa.StartDate BETWEEN '01-JAN-2023' AND '31-MAR-2023' OR sa.EndDate BETWEEN
'01-JAN-2023' AND '31-MAR-2023')
ORDER BY sa.ArrangementType ASC;
```

2) Output Part 1 C Query a: Provide the Output for executing the above PL/SQL code. Make sure the output is NOT TEXT but rather images or screenshots. If no correct output provided, then 2 marks will be subtracted (-2 marks):

```
W22046071 >
W22046071 >
W22046071 > SELECT c.Name, s.SiteName, sa.ArrangementType, sa.Description, sa.StartDate, sa.EndDate
  2  FROM TabClient c
  3  INNER JOIN TabSite s ON c.ClientID = s.ClientID.ClientID
  4  INNER JOIN TabSecurityArrangement sa ON s.SiteID = sa.SiteID.SiteID
  5  WHERE s.Address = 'Newcastle upon Tyne'
  6  AND (sa.StartDate BETWEEN '01-JAN-2023' AND '31-MAR-2023' OR sa.EndDate BETWEEN '01-JAN-2023' AND '31-MAR-2023')
  7  ORDER BY sa.ArrangementType ASC;

NAME                SITENAME            ARRANGEMENTTYPE      DESCRIPTION                                STARTDATE ENDDATE
------------------- ------------------- -------------------- ------------------------------------------ --------- ---------
Nikith              Sarah Sells Homes   Weekdays and weekend Security for the storeroom                 12-FEB-23 15-MAR-23
Shaiksha            Wonderful Homes     Weekdays only        Security for the building                  25-MAR-23 25-MAR-23
Shahul              Texas Waterfront    Weekdays only        Security for the office                    15-MAR-23 01-APR-23
Nikith              Sarah Sells Homes   Weekend only         Security for the entrance                  01-FEB-23 15-APR-23
Shahul              Texas Waterfront    Weekend only         Security for the conference room           10-JAN-23 10-JAN-23

W22046071 >
W22046071 >
```

Q2) Display details of the client(s) who brought the most business in terms of most money paid to *NorthEastNinjas* during 2023. The output should not be tabular but structured like a report on client by client basis, e.g., details of the client(s), followed by their total money brought the company with a breakdown of the total on site by site basis.

1) Answer Part 1 C Query b: Provide the PL/SQL code (5 marks):

```
SELECT c.Name, SUM(b.TotalAmount) AS TotalAmount
FROM TabClient c
INNER JOIN TabBooking b ON c.ClientID = b.ClientID.ClientID
WHERE b.BookingDate BETWEEN '01-JAN-2023' AND '31-DEC-2023'
GROUP BY c.Name
ORDER BY SUM(b.TotalAmount) DESC
FETCH FIRST 1 ROW ONLY;
```

2) Output Part 1 C Query b: Provide the Output for executing the above PL/SQL code. Make sure the output is NOT TEXT but rather images or screenshots. If no correct output provided, then 2 marks will be subtracted (-2 marks):

```
W22046071 >
W22046071 > SELECT c.Name, SUM(b.TotalAmount) AS TotalAmount
  2  FROM TabClient c
  3  INNER JOIN TabBooking b ON c.ClientID = b.ClientID.ClientID
  4  WHERE b.BookingDate BETWEEN '01-JAN-2023' AND '31-DEC-2023'
  5  GROUP BY c.Name
  6  ORDER BY SUM(b.TotalAmount) DESC
  7  FETCH FIRST 1 ROW ONLY;

NAME                TOTALAMOUNT
------------------- -----------
Nikith                     1000

W22046071 >
```

D) Contrasting your OR version with the relational version of the NorthEastNinjas Database from your assignment 1

(4 marks)

You should highlight the advantages and disadvantages of both versions. Comment on which version will best suit the NorthEastNinjas system.

Answer Part 1 D: Provide your comparison below

S.No		**Object Relation version**	**Entity Framework**
1	**Advantages**	Data modelling is enhanced in OR version with improved performance.	Development process is simplified
2		Data is very consistent and data integrity is maintained and reduces the amount of code used (Mishra, 2022).	Automatic handling of database operations
3		Helps in data prevention or inconsistent data.	Helps in object-relational mapping along with query optimisation.
4		Developing and maintaining an application is easier because of the integration with object oriented relational database.	It is very flexible as it adapts with the requirements of the business. (Griff, 2015)
5	**Disadvantages**	There is complexity and potential performance overhead in the OR version (Mishra, 2022).	There will be a performance issues when compared with the OR version (Griff, 2015).

For this scenario, **OR version** is best suitable because of its quick performance over the entity framework. The performance was high and there will be more security to the data along with quick referencing.

Part 2: Data Warehousing Tasks (50 Marks)

This part is based on the Sales History scenario as described in Appendix 2.

You must <u>submit</u> all the SQL queries and any other code that you wrote in answering any of the tasks / questions (e.g., the use of Explain Plan statements for the queries and their outputs as screenshots).

(A) Study the index definitions in sh_idx.sql. These indexes have already been created in SH2. Whatever indexes you decide to create for this task should be the result of your own research and thinking, and be different than those already existing in SH2 or those indexes defined in the Oracle Data Warehousing Guide (Potineni, 2017) or those of other students.

You need to design *two* queries such that each query involves at least *three* different tables and at least *one* aggregate function. You need to ensure that your queries have adequate *selectivity* such that if suitable indexes were available in your DWU version of the database, the queries would have performed more efficiently.

You need to identify and justify at least two indexes to improve the performance of your queries. Then create your proposed indexes in your DWU version of the database. You need to run your queries before and after creating your proposed indexes and report EXPLAIN PLAN outputs and make sure that your proposed indexes have been used by your queries and have improved their performance significantly.

Then discuss the differences in the performance of your queries with and without the proposed indexes. You need to cite relevant database literature to support your choice of indexes and how you dealt with the issue of selectivity in your queries.

(25 marks)

Answer Part 2 (A)

Provide the SQL Code and output for the 2 new indexes you have created on your DWU database for comparing their performance impact on DWU (i.e., these indexes must not exist in SH2) (4 Marks). *Make sure the SQL code you provide is plain text and the output is a screenshot.*

Index 1:

```
CREATE BITMAP INDEX country_subregion_bix
ON DWU390.countries (country_subregion)
NOLOGGING COMPUTE STATISTICS ;
```

```
W22046071 >
W22046071 > CREATE BITMAP INDEX country_subregion_bix
  2         ON DWU390.countries (country_subregion)
  3         NOLOGGING COMPUTE STATISTICS ;

Index created.

W22046071 >
```

Index 2:

```
CREATE BITMAP INDEX promo_subcategory_bix
ON DWU390.promotions (promo_subcategory)
NOLOGGING COMPUTE STATISTICS ;
```

```
W22046071 >
W22046071 >
W22046071 > CREATE BITMAP INDEX promo_subcategory_bix
  2         ON DWU390.promotions (promo_subcategory)
  3         NOLOGGING COMPUTE STATISTICS ;

Index created.

W22046071 >
W22046071 >
```

Provide the rationale and justification of creating the above indexes based on your own research and citing appropriate literature here and providing references in the "References and Bibliography" section at the end of the report (5 Marks):

Relevant data used in the database was easily located because of the indexed column was included in query and the execution time was very faster. Creating indexes helps in maintaining the performance of the system and the response time was also improved. With indexes, there is no need for the database to scan the entire table instead a particular and desired dataset should be chosen for scanning. With the help of index values, the dataset which was relevant to the database

was accessed by performing a range scan or index scan. Because of the easy access of the dataset, the scanning process was minimised by improving the efficiency of query. The throughput of the system and parallel processing was improved without any conflicts. Data was retrieved with the help of index structures by eliminating the sorting operations.

Provide 2 SQL queries you are going to run to compare the performance impact of your own 2 new indexes on DWU (6 marks). *Make sure the SQL code you provide is plain text.*

Index 1:

```
SELECT CST.cust_id, CST.cust_first_name, CST.cust_gender, COUNT(S.cust_id) AS
CustomerCount
FROM DWU390.countries C, DWU390.customers CST, DWU390.sales S
WHERE C.country_id = C.country_id
AND CST.cust_id = S.cust_id
AND C.country_subregion = 'Asia'
GROUP BY CST.cust_id, CST.cust_first_name, CST.cust_gender
FETCH FIRST 5 ROWS ONLY;
```

```
W22046071 >
W22046071 > SELECT CST.cust_id, CST.cust_first_name, CST.cust_gender, COUNT(S.cust_id) AS CustomerCount
  2  FROM DWU390.countries C, DWU390.customers CST, DWU390.sales S
  3  WHERE C.country_id = C.country_id
  4  AND CST.cust_id = S.cust_id
  5  AND C.country_subregion = 'Asia'
  6  GROUP BY CST.cust_id, CST.cust_first_name, CST.cust_gender
  7  FETCH FIRST 5 ROWS ONLY;

   CUST_ID CUST_FIRST_NAME      C CUSTOMERCOUNT
---------- -------------------- - -------------
     48290 Bailey               M          2142
    129080 Calvert              M          2814
     43680 Lola                 M          3090
      9200 Carlos               M          2433
     13150 Pia                  M          3090

W22046071 >
W22046071 >
```

Index 2:

```
SELECT S.cust_id, P.promo_category, C.channel_class, AVG(S.AMOUNT_SOLD) AS
SalesAmount
FROM DWU390.promotions P , DWU390.Sales S, DWU390.channels C
WHERE P.promo_id = S.promo_id
AND C.channel_id = S.channel_id
AND P.promo_subcategory = 'radio program sponsorship'
```

```
GROUP BY S.cust_id, P.promo_category, C.channel_class
FETCH FIRST 5 ROW ONLY;
```

```
W22046071 >
W22046071 >
W22046071 > SELECT S.cust_id, P.promo_category, C.channel_class, AVG(S.AMOUNT_SOLD) AS SalesAmount
  2  FROM DWU390.promotions P , DWU390.Sales S, DWU390.channels C
  3  WHERE P.promo_id = S.promo_id
  4  AND C.channel_id = S.channel_id
  5  AND P.promo_subcategory = 'radio program sponsorship'
  6  GROUP BY S.cust_id, P.promo_category, C.channel_class
  7  FETCH FIRST 5 ROW ONLY;

   CUST_ID PROMO_CATEGORY                CHANNEL_CLASS        SALESAMOUNT
---------- ------------------------------ -------------------- -----------
     35640 radio                          Direct                    368.55
     35180 radio                          Direct                   356.225
    120760 radio                          Direct                    631.94
    117200 radio                          Direct                    653.72
     56990 radio                          Indirect                   578.7

W22046071 >
W22046071 >
```

Provide Explain Plan statements & outputs for the above 2 SQL queries you have run to compare the performance impact of your 2 indexes on DWU before and after creating your proposed indexes (4 marks). *Make sure the SQL code you provide is plain text and the output is a screenshot.*

Before Index 1:

```
EXPLAIN PLAN FOR
SELECT CST.cust_id, CST.cust_first_name, CST.cust_gender, COUNT(S.cust_id) AS
CustomerCount
FROM DWU390.countries C, DWU390.customers CST, DWU390.sales S
WHERE C.country_id = C.country_id
AND CST.cust_id = S.cust_id
AND C.country_subregion = 'Asia'
GROUP BY CST.cust_id, CST.cust_first_name, CST.cust_gender;
```

```
W22046071 >
W22046071 >
W22046071 >
W22046071 > EXPLAIN PLAN FOR
  2  SELECT CST.cust_id, CST.cust_first_name, CST.cust_gender, COUNT(S.cust_id) AS CustomerCount
  3  FROM DWU390.countries C, DWU390.customers CST, DWU390.sales S
  4  WHERE C.country_id = C.country_id
  5  AND CST.cust_id = S.cust_id
  6  AND C.country_subregion = 'Asia'
  7  GROUP BY CST.cust_id, CST.cust_first_name, CST.cust_gender;

Explained.

W22046071 >
```

```
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY());
```

```
W22046071 >
W22046071 > SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY());

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------
-----------
Plan hash value: 1319077086

-----------------------------------------------------------------------------------------------------------
| Id  | Operation              | Name      | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     | Pstart| Pstop |
-----------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |           |  1255K|    41M|       | 20495   (2)| 00:00:01 |       |       |
|   1 |  HASH GROUP BY         |           |  1255K|    41M|   101M| 20495   (2)| 00:00:01 |       |       |
|*  2 |   HASH JOIN            |           |  2413K|    80M|  4872K|  4924   (2)| 00:00:01 |       |       |
|   3 |    MERGE JOIN CARTESIAN|           |   118K|  3479K|       |   549   (1)| 00:00:01 |       |       |
|*  4 |     TABLE ACCESS FULL  | COUNTRIES |     2 |    32 |       |     3   (0)| 00:00:01 |       |       |
|   5 |     BUFFER SORT        |           | 50000 |   683K|       |   546   (1)| 00:00:01 |       |       |
|   6 |      TABLE ACCESS FULL | CUSTOMERS | 50000 |   683K|       |   273   (1)| 00:00:01 |       |       |
|   7 |    PARTITION RANGE ALL |           |  1016K|  4962K|       |  3294   (1)| 00:00:01 |     1 |    17 |
|   8 |     TABLE ACCESS FULL  | SALES     |  1016K|  4962K|       |  3294   (1)| 00:00:01 |     1 |    17 |
-----------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("CST"."CUST_ID"="S"."CUST_ID")
   4 - filter("C"."COUNTRY_SUBREGION"='Asia')

21 rows selected.

W22046071 >
W22046071 >
```

After Index 1:

```
EXPLAIN PLAN FOR
SELECT CST.cust_id, CST.cust_first_name, CST.cust_gender, COUNT(S.cust_id) AS
CustomerCount
FROM DWU390.countries C, DWU390.customers CST, DWU390.sales S
WHERE C.country_id = C.country_id
AND CST.cust_id = S.cust_id
AND C.country_subregion = 'Asia'
GROUP BY CST.cust_id, CST.cust_first_name, CST.cust_gender;
```

```
W22046071 >
W22046071 > EXPLAIN PLAN FOR
  2  SELECT CST.cust_id, CST.cust_first_name, CST.cust_gender, COUNT(S.cust_id) AS CustomerCount
  3  FROM DWU390.countries C, DWU390.customers CST, DWU390.sales S
  4  WHERE C.country_id = C.country_id
  5  AND CST.cust_id = S.cust_id
  6  AND C.country_subregion = 'Asia'
  7  GROUP BY CST.cust_id, CST.cust_first_name, CST.cust_gender;

Explained.

W22046071 >
W22046071 >
```

```
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY());
```

```
W22046071 >
W22046071 > SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY());

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------------------
Plan hash value: 2389774383

-----------------------------------------------------------------------------------------------------------------------
| Id  | Operation                       | Name                 | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     | Pstart| Pstop |
-----------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                |                      |  1255K|    41M|       | 20494   (2)| 00:00:01 |       |       |
|   1 |  HASH GROUP BY                  |                      |  1255K|    41M|   101M| 20494   (2)| 00:00:01 |       |       |
|*  2 |   HASH JOIN                     |                      |  2413K|    80M|  4872K|  4923   (2)| 00:00:01 |       |       |
|   3 |    MERGE JOIN CARTESIAN         |                      |   118K|  3479K|       |   548   (1)| 00:00:01 |       |       |
|*  4 |     VIEW                        | index$_join$_001     |     2 |    32 |       |     2   (0)| 00:00:01 |       |       |
|*  5 |      HASH JOIN                  |                      |       |       |       |            |          |       |       |
|   6 |       BITMAP CONVERSION TO ROWIDS|                     |     2 |    32 |       |     1   (0)| 00:00:01 |       |       |
|*  7 |        BITMAP INDEX SINGLE VALUE | COUNTRY_SUBREGION_BIX |      |       |       |            |          |       |       |
|   8 |       INDEX FAST FULL SCAN      | COUNTRY_PK           |     2 |    32 |       |     1   (0)| 00:00:01 |       |       |
|   9 |     BUFFER SORT                 |                      | 50000 |   683K|       |   546   (1)| 00:00:01 |       |       |
|  10 |      TABLE ACCESS FULL          | CUSTOMERS            | 50000 |   683K|       |   273   (1)| 00:00:01 |       |       |
|  11 |    PARTITION RANGE ALL          |                      |  1016K|  4962K|       |  3294   (1)| 00:00:01 |     1 |    17 |
|  12 |     TABLE ACCESS FULL           | SALES                |  1016K|  4962K|       |  3294   (1)| 00:00:01 |     1 |    17 |
-----------------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("CST"."CUST_ID"="S"."CUST_ID")
   4 - filter("C"."COUNTRY_SUBREGION"='Asia')
   5 - access(ROWID=ROWID)
   7 - access("C"."COUNTRY_SUBREGION"='Asia')

27 rows selected.

W22046071 >
```

Before Index 2:

```
EXPLAIN PLAN FOR
SELECT S.cust_id, P.promo_category, C.channel_class, AVG(S.AMOUNT_SOLD) AS
SalesAmount
FROM DWU390.promotions P , DWU390.Sales S, DWU390.channels C
WHERE P.promo_id = S.promo_id
AND C.channel_id = S.channel_id
AND P.promo_subcategory = 'radio program sponsorship'
GROUP BY S.cust_id, P.promo_category, C.channel_class;
```

```
W22046071 >
W22046071 >
W22046071 > EXPLAIN PLAN FOR
  2  SELECT S.cust_id, P.promo_category, C.channel_class, AVG(S.AMOUNT_SOLD) AS SalesAmount
  3  FROM DWU390.promotions P , DWU390.Sales S, DWU390.channels C
  4  WHERE P.promo_id = S.promo_id
  5  AND C.channel_id = S.channel_id
  6  AND P.promo_subcategory = 'radio program sponsorship'
  7  GROUP BY S.cust_id, P.promo_category, C.channel_class;

Explained.

W22046071 >
W22046071 >
```

```
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY());
```

```
W22046071 >
W22046071 > SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY());

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------------------------------------------
Plan hash value: 4230855687

-------------------------------------------------------------------------------------------------
| Id  | Operation              | Name       | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     | Pstart| Pstop |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |            | 18828 |   790K|       |  3704   (2)| 00:00:01 |       |       |
|   1 |  HASH GROUP BY         |            | 18828 |   790K|  3320K|  3704   (2)| 00:00:01 |       |       |
|*  2 |   HASH JOIN            |            | 64868 |  2723K|       |  3313   (2)| 00:00:01 |       |       |
|   3 |    TABLE ACCESS FULL   | CHANNELS   |     5 |    55 |       |     3   (0)| 00:00:01 |       |       |
|*  4 |    HASH JOIN           |            | 64868 |  2027K|       |  3309   (2)| 00:00:01 |       |       |
|   5 |     VIEW               | VW_GBF_10  |    24 |   408 |       |     3   (0)| 00:00:01 |       |       |
|*  6 |      TABLE ACCESS FULL | PROMOTIONS |    24 |   672 |       |     3   (0)| 00:00:01 |       |       |
|   7 |     PARTITION RANGE ALL|            |  1016K|    14M|       |  3298   (1)| 00:00:01 |     1 |    17 |
|   8 |      TABLE ACCESS FULL | SALES      |  1016K|    14M|       |  3298   (1)| 00:00:01 |     1 |    17 |
-------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("C"."CHANNEL_ID"="S"."CHANNEL_ID")
   4 - access("ITEM_1"="S"."PROMO_ID")
   6 - filter("P"."PROMO_SUBCATEGORY"='radio program sponsorship')

22 rows selected.

W22046071 >
W22046071 >
```

After Index 2:

```
EXPLAIN PLAN FOR
SELECT S.cust_id, P.promo_category, C.channel_class, AVG(S.AMOUNT_SOLD) AS
SalesAmount
FROM DWU390.promotions P , DWU390.Sales S, DWU390.channels C
WHERE P.promo_id = S.promo_id
AND C.channel_id = S.channel_id
AND P.promo_subcategory = 'radio program sponsorship'
GROUP BY S.cust_id, P.promo_category, C.channel_class;
```

```
W22046071 >
W22046071 > EXPLAIN PLAN FOR
  2  SELECT S.cust_id, P.promo_category, C.channel_class, AVG(S.AMOUNT_SOLD) AS SalesAmount
  3  FROM DWU390.promotions P , DWU390.Sales S, DWU390.channels C
  4  WHERE P.promo_id = S.promo_id
  5  AND C.channel_id = S.channel_id
  6  AND P.promo_subcategory = 'radio program sponsorship'
  7  GROUP BY S.cust_id, P.promo_category, C.channel_class;

Explained.

W22046071 >
W22046071 >
```

```
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY());
```

```
W22046071 >
W22046071 > SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY());

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------------------------------
Plan hash value: 2395442496

-------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                                | Name                 | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     | Pstart| Pstop |
-------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                         |                      | 18828 |   790K|       |  3703   (2)| 00:00:01 |       |       |
|   1 |  HASH GROUP BY                           |                      | 18828 |   790K|  3320K|  3703   (2)| 00:00:01 |       |       |
|*  2 |   HASH JOIN                              |                      | 64868 |  2723K|       |  3312   (2)| 00:00:01 |       |       |
|   3 |    TABLE ACCESS FULL                     | CHANNELS             |     5 |    55 |       |     3   (0)| 00:00:01 |       |       |
|*  4 |    HASH JOIN                             |                      | 64868 |  2027K|       |  3309   (2)| 00:00:01 |       |       |
|   5 |     VIEW                                 | VW_GBF_10            |    24 |   408 |       |     2   (0)| 00:00:01 |       |       |
|   6 |      TABLE ACCESS BY INDEX ROWID BATCHED| PROMOTIONS           |    24 |   672 |       |     2   (0)| 00:00:01 |       |       |
|   7 |       BITMAP CONVERSION TO ROWIDS        |                      |       |       |       |            |          |       |       |
|*  8 |        BITMAP INDEX SINGLE VALUE         | PROMO_SUBCATEGORY_BIX|       |       |       |            |          |       |       |
|   9 |     PARTITION RANGE ALL                  |                      | 1016K |    14M|       |  3298   (1)| 00:00:01 |     1 |    17 |
|  10 |      TABLE ACCESS FULL                   | SALES                | 1016K |    14M|       |  3298   (1)| 00:00:01 |     1 |    17 |
-------------------------------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("C"."CHANNEL_ID"="S"."CHANNEL_ID")
   4 - access("ITEM_1"="S"."PROMO_ID")
   8 - access("P"."PROMO_SUBCATEGORY"='radio program sponsorship')

24 rows selected.

W22046071 >
W22046071 >
```

Provide discussion of the cost-based comparison of the above 2 sets of queries and their explain plan cost figures/values (6 marks):

Before Index 1: The cost here is 3 because index was not used here. The cost is 3 here because index was not used and hence it reduces the performance by slow down the time for processing the queries or dataset.
After Index 1: The cost after index is 1 because of the usage of index. Here, the index was used and the cost was very low when compared with before index1. The usage of index reduces the cost because the performance is very quick after using the index.

Before Index 2: Without the use of index, the cost was 3 because of the slow performance. To reduce the cost, index must be used.
After Index 2: The cost is 1 because index is used and hence the performance was so quick in this and the cost was reduced.

(B) There are two materialized views (MVs) defined in sh_cremv.sql and these MVs have already been created under SH2 shared schema. You should study these two MVs and understand their benefits to the user of the SH2 data warehouse.

You need to design and create two new MVs on the base tables in your DWU schema. Each of your proposed MV should involve at least *three* different tables and at least *one* aggregate function. Justify why these *two new* MVs would be useful for the users of your data warehouse. Note that you must create brand new and unique MVs, based on your own research and thinking, and these should be completely different than those of SH2 or those MVs defined in the Oracle Data Warehousing Guide (Potineni, 2017) or those of other students.

Then design *two* queries such that when you run these queries, the database optimizer will re-write these queries and instead of the tables named in your queries, the system will use the *two new* MVs to answer the queries. Note that the queries should return subsets of the values contained in these MVs. Moreover, you must not query your MVs directly in the FROM clause; let the database optimizer re-write these queries and answer them using the new MVs.

You need to run your queries on both the SH2 schema and on your DWU schema and report EXPLAIN PLAN outputs. You should make sure that the queries on the DWU schema use the new MVs and have significantly better performance compared to the same queries' performance when ran on the SH2 data warehouse as the newly proposed MVs would not exist in the SH2 schema.

Then discuss the differences in the performance of your queries with (in the case of DWU schema) and without (in the case of SH2 schema) the proposed MVs. You need to cite relevant database literature to support your choice of MVs and queries.

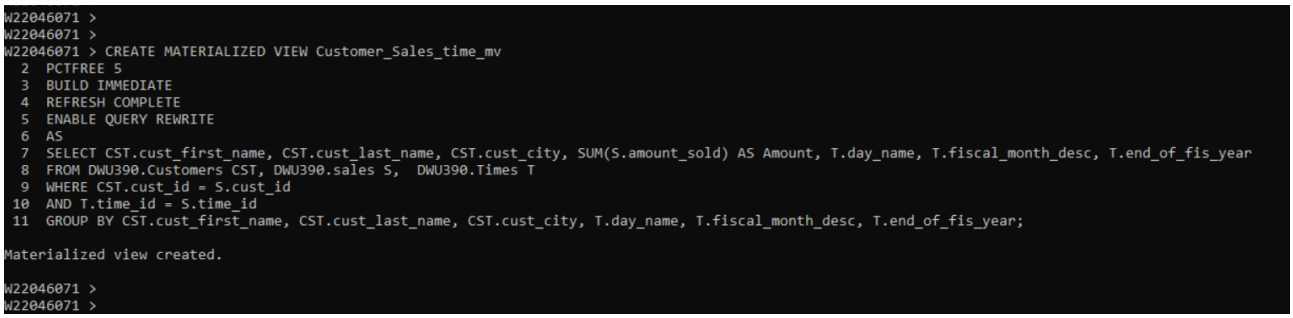
(25 marks)

Answer Part 2 (B)

Provide SQL code and output you used to create the 2 new MVs in your own DWU database (i.e., these MVs *must not* exist in SH2) (6 marks). *Make sure the SQL code you provide is plain text and the output is a screenshot.*

MV 1:

```
CREATE MATERIALIZED VIEW Customer_Sales_time_mv
PCTFREE 5
BUILD IMMEDIATE
REFRESH COMPLETE
ENABLE QUERY REWRITE
AS
SELECT CST.cust_first_name, CST.cust_last_name, CST.cust_city,
SUM(S.amount_sold) AS Amount, T.day_name, T.fiscal_month_desc, T.end_of_fis_year
FROM DWU390.Customers CST, DWU390.sales S,  DWU390.Times T
WHERE CST.cust_id = S.cust_id
AND T.time_id = S.time_id
GROUP BY CST.cust_first_name, CST.cust_last_name, CST.cust_city, T.day_name,
T.fiscal_month_desc, T.end_of_fis_year;
```



MV 2:

```
CREATE MATERIALIZED VIEW sales_prmo_channel_mv
PCTFREE 5
BUILD IMMEDIATE
REFRESH COMPLETE
ENABLE QUERY REWRITE
AS
SELECT COUNT(S.cust_id) AS SalesCustCount,C.channel_class, S.time_id,
P.supplier_id, P.prod_category, S.prod_id
```

```
FROM DWU390.products P, DWU390.Channels C, DWU390.sales S
WHERE P.prod_id = S.prod_id
AND C.channel_id = S.channel_id
GROUP BY C.channel_class, S.time_id, P.supplier_id, P.prod_category, S.prod_id;
```

```
W22046071 >
W22046071 > CREATE MATERIALIZED VIEW sales_prmo_channel_mv
  2  PCTFREE 5
  3  BUILD IMMEDIATE
  4  REFRESH COMPLETE
  5  ENABLE QUERY REWRITE
  6  AS
  7  SELECT COUNT(S.cust_id) AS SalesCustCount,C.channel_class, S.time_id, P.supplier_id, P.prod_category, S.prod_id
  8  FROM DWU390.products P, DWU390.Channels C, DWU390.sales S
  9  WHERE P.prod_id = S.prod_id
 10  AND C.channel_id = S.channel_id
 11  GROUP BY C.channel_class, S.time_id, P.supplier_id, P.prod_category, S.prod_id;

Materialized view created.

W22046071 >
W22046071 >
```

Provide the rationale and justification of creating the above MVs based on your own research and citing appropriate literature here and providing references in the "References and Bibliography" section at the end of the report (5 Marks):

Materialised view was created because of the various advantages of it such as quick performance along with reducing time and need for complex queries and reducing the time for running in performing calculations. The resources used for making queries should also be very less and has more capacity in handling queries. This MV also makes sure about the data consistency and the queries are stored cased on the tables and it gets updated automatically in order to reflect the modifications along with data integrity maintenance. Large datasets was processed and data retrieval was very easy with the help of materialised view.

Provide the 2 SQL queries you are going to run to compare the performance impact of your own 2 new MVs on DWU and the version of the same queries on SH2 (4 marks).
Make sure the SQL code you provide is plain text and the output is a screenshot.

MV 1 in SH2:

```
SELECT CST.cust_first_name, CST.cust_last_name, CST.cust_city,
SUM(S.amount_sold) AS Amount, T.day_name, T.fiscal_month_desc, T.end_of_fis_year
FROM SH2.Customers CST, SH2.sales S,  SH2.Times T
WHERE CST.cust_id = S.cust_id
AND T.time_id = S.time_id
GROUP BY CST.cust_first_name, CST.cust_last_name, CST.cust_city, T.day_name,
T.fiscal_month_desc, T.end_of_fis_year
FETCH FIRST 5 ROW ONLY;
```

```
W22046071 >
W22046071 > SELECT CST.cust_first_name, CST.cust_last_name, CST.cust_city, SUM(S.amount_sold) AS Amount, T.day_name, T.fiscal_month_desc, T.end_of_fis_year
  2  FROM SH2.Customers CST, SH2.sales S,  SH2.Times T
  3  WHERE CST.cust_id = S.cust_id
  4  AND T.time_id = S.time_id
  5  GROUP BY CST.cust_first_name, CST.cust_last_name, CST.cust_city, T.day_name, T.fiscal_month_desc, T.end_of_fis_year
  6  FETCH FIRST 5 ROW ONLY;

CUST_FIRST_NAME      CUST_LAST_NAME                           CUST_CITY                      AMOUNT DAY_NAME  FISCAL_M END_OF_FI
-------------------- ---------------------------------------- ------------------------------ ------ --------- -------- ---------
Travis               Kimball                                  Arbuckle                         2256 Thursday  2001-02  30-DEC-01
Lynna                Ogletree                                 Rosenheim                        1243 Thursday  2001-02  30-DEC-01
Ada                  Nenninger                                Gif-sur-Yvette                   1084 Thursday  2001-02  30-DEC-01
Eustace              Dwyer                                    Thame                            1144 Thursday  2001-02  30-DEC-01
Carey                Orr                                      Nieuwegein                       1750 Thursday  2001-02  30-DEC-01

W22046071 >
W22046071 >
```

MV 1 in DWU:

```
SELECT CST.cust_first_name, CST.cust_last_name, CST.cust_city,
SUM(S.amount_sold) AS Amount, T.day_name, T.fiscal_month_desc, T.end_of_fis_year
FROM DWU390.Customers CST, DWU390.sales S,  DWU390.Times T
WHERE CST.cust_id = S.cust_id
AND T.time_id = S.time_id
GROUP BY CST.cust_first_name, CST.cust_last_name, CST.cust_city, T.day_name,
T.fiscal_month_desc, T.end_of_fis_year
FETCH FIRST 5 ROW ONLY;
```

```
W22046071 >
W22046071 > SELECT CST.cust_first_name, CST.cust_last_name, CST.cust_city, SUM(S.amount_sold) AS Amount, T.day_name, T.fiscal_month_desc, T.end_of_fis_year
  2  FROM DWU390.Customers CST, DWU390.sales S,  DWU390.Times T
  3  WHERE CST.cust_id = S.cust_id
  4  AND T.time_id = S.time_id
  5  GROUP BY CST.cust_first_name, CST.cust_last_name, CST.cust_city, T.day_name, T.fiscal_month_desc, T.end_of_fis_year
  6  FETCH FIRST 5 ROW ONLY;

CUST_FIRST_NAME      CUST_LAST_NAME                      CUST_CITY                      AMOUNT DAY_NAME  FISCAL_M END_OF_FI
-------------------- ----------------------------------- ------------------------------ ------ --------- -------- ---------
Benedict             Elgin                               Wakefield                         747 Tuesday   2001-02  30-DEC-01
Ines                 Legard                              Didcot                           2134 Monday    2001-03  30-DEC-01
Mara                 Daley                               Edgewood                         5669 Saturday  2001-02  30-DEC-01
Martin               Osgode                              Saint Marks                      4877 Tuesday   2001-02  30-DEC-01
Jonathan             Osborne                             Killarney                        5730 Monday    2001-03  30-DEC-01

W22046071 >
W22046071 >
W22046071 >
```

MV 2 in SH2:

```
SELECT COUNT(S.cust_id) AS SalesCustCount,C.channel_class, S.time_id,
P.supplier_id, P.prod_category, S.prod_id
FROM SH2.products P, SH2.Channels C, SH2.sales S
WHERE P.prod_id = S.prod_id
AND C.channel_id = S.channel_id
GROUP BY C.channel_class, S.time_id, P.supplier_id, P.prod_category, S.prod_id
FETCH FIRST 5 ROWS ONLY;
```

```
W22046071 >
W22046071 >
W22046071 > SELECT COUNT(S.cust_id) AS SalesCustCount,C.channel_class, S.time_id, P.supplier_id, P.prod_category, S.prod_id
  2  FROM SH2.products P, SH2.Channels C, SH2.sales S
  3  WHERE P.prod_id = S.prod_id
  4  AND C.channel_id = S.channel_id
  5  GROUP BY C.channel_class, S.time_id, P.supplier_id, P.prod_category, S.prod_id
  6  FETCH FIRST 5 ROWS ONLY;

SALESCUSTCOUNT CHANNEL_CLASS        TIME_ID   SUPPLIER_ID PROD_CATEGORY                                      PROD_ID
-------------- -------------------- --------- ----------- -------------------------------------------------- ----------
             1 Franchise            01-FEB-01          83 Boys                                                5810
             1 Indirect             01-FEB-01         166 Women                                               5070
             1 Franchise            01-FEB-01          37 Girls                                               4635
             1 Indirect             01-FEB-01          52 Boys                                                3935
             1 Indirect             01-FEB-01         122 Men                                                 3930

W22046071 >
W22046071 >
```

MV 2 in DWU:

```
SELECT COUNT(S.cust_id) AS SalesCustCount,C.channel_class, S.time_id,
P.supplier_id, P.prod_category, S.prod_id
FROM DWU390.products P, DWU390.Channels C, DWU390.sales S
WHERE P.prod_id = S.prod_id
AND C.channel_id = S.channel_id
GROUP BY C.channel_class, S.time_id, P.supplier_id, P.prod_category, S.prod_id
FETCH FIRST 5 ROWS ONLY;
```

```
W22046071 >
W22046071 > SELECT COUNT(S.cust_id) AS SalesCustCount,C.channel_class, S.time_id, P.supplier_id, P.prod_category, S.prod_id
  2  FROM DWU390.products P, DWU390.Channels C, DWU390.sales S
  3  WHERE P.prod_id = S.prod_id
  4  AND C.channel_id = S.channel_id
  5  GROUP BY C.channel_class, S.time_id, P.supplier_id, P.prod_category, S.prod_id
  6  FETCH FIRST 5 ROWS ONLY;

SALESCUSTCOUNT CHANNEL_CLASS        TIME_ID   SUPPLIER_ID PROD_CATEGORY                                          PROD_ID
-------------- -------------------- --------- ----------- -------------------------------------------------- ----------
             1 Direct               20-FEB-01          13 Girls                                                    6200
             1 Direct               21-FEB-01         115 Men                                                     18560
             1 Indirect             21-FEB-01         228 Women                                                   48830
             1 Direct               26-FEB-01          91 Girls                                                    1825
             1 Indirect             26-FEB-01         120 Women                                                    1655

W22046071 >
W22046071 >
```

Provide Explain Plan statements & outputs for the above 2 SQL queries you have run to compare the performance impact of your 2 MVs on DWU and their version of the same queries on SH2 (4 marks):

MV 1 in SH2:

```
EXPLAIN PLAN FOR
SELECT CST.cust_first_name, CST.cust_last_name, CST.cust_city,
SUM(S.amount_sold) AS Amount, T.day_name, T.fiscal_month_desc, T.end_of_fis_year
FROM DWU390.Customers CST, DWU390.sales S,  DWU390.Times T
WHERE CST.cust_id = S.cust_id
AND T.time_id = S.time_id
GROUP BY CST.cust_first_name, CST.cust_last_name, CST.cust_city, T.day_name,
T.fiscal_month_desc, T.end_of_fis_year;
```

```
W22046071 >
W22046071 > EXPLAIN PLAN FOR
  2  SELECT CST.cust_first_name, CST.cust_last_name, CST.cust_city, SUM(S.amount_sold) AS Amount, T.day_name, T.fiscal_month_desc, T.end_of_fis_year
  3  FROM SH2.Customers CST, SH2.sales S,  SH2.Times T
  4  WHERE CST.cust_id = S.cust_id
  5  AND T.time_id = S.time_id
  6  GROUP BY CST.cust_first_name, CST.cust_last_name, CST.cust_city, T.day_name, T.fiscal_month_desc, T.end_of_fis_year;

Explained.

W22046071 >
```

```
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY());
```

```
W22046071 >
W22046071 > SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY());

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------------------
-----------
Plan hash value: 3113689673

-----------------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name      | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     | Pstart| Pstop |
-----------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |           |  1016K|    78M|       | 23179   (1)| 00:00:01 |       |       |
|   1 |  HASH GROUP BY                |           |  1016K|    78M|    93M| 23179   (1)| 00:00:01 |       |       |
|*  2 |   HASH JOIN                   |           |  1016K|    78M|       |  3900   (2)| 00:00:01 |       |       |
|   3 |    PART JOIN FILTER CREATE    | :BF0000   |  1461 | 49674 |       |    13   (0)| 00:00:01 |       |       |
|   4 |     TABLE ACCESS FULL         | TIMES     |  1461 | 49674 |       |    13   (0)| 00:00:01 |       |       |
|*  5 |    HASH JOIN                  |           |  1016K|    45M|  2056K|  3879   (2)| 00:00:01 |       |       |
|   6 |     TABLE ACCESS FULL         | CUSTOMERS | 50000 |  1464K|       |   267   (1)| 00:00:01 |       |       |
|   7 |     PARTITION RANGE JOIN-FILTER|          |  1016K|    16M|       |  2102   (2)| 00:00:01 |:BF0000|:BF0000|
|   8 |      TABLE ACCESS FULL        | SALES     |  1016K|    16M|       |  2102   (2)| 00:00:01 |:BF0000|:BF0000|
-----------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("T"."TIME_ID"="S"."TIME_ID")
   5 - access("CST"."CUST_ID"="S"."CUST_ID")

Note
-----
   - this is an adaptive plan

25 rows selected.

W22046071 >
```

MV 1 in DWU:

```
EXPLAIN PLAN FOR
SELECT CST.cust_first_name, CST.cust_last_name, CST.cust_city,
SUM(S.amount_sold) AS Amount, T.day_name, T.fiscal_month_desc, T.end_of_fis_year
FROM DWU390.Customers CST, DWU390.sales S,  DWU390.Times T
WHERE CST.cust_id = S.cust_id
AND T.time_id = S.time_id
GROUP BY CST.cust_first_name, CST.cust_last_name, CST.cust_city, T.day_name,
T.fiscal_month_desc, T.end_of_fis_year;
```

```
W22046071 >
W22046071 > EXPLAIN PLAN FOR
  2  SELECT CST.cust_first_name, CST.cust_last_name, CST.cust_city, SUM(S.amount_sold) AS Amount, T.day_name, T.fiscal_month_desc, T.end_of_fis_year
  3  FROM DWU390.Customers CST, DWU390.sales S,  DWU390.Times T
  4  WHERE CST.cust_id = S.cust_id
  5  AND T.time_id = S.time_id
  6  GROUP BY CST.cust_first_name, CST.cust_last_name, CST.cust_city, T.day_name, T.fiscal_month_desc, T.end_of_fis_year;

Explained.

W22046071 >
W22046071 >
```

```
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY());
```

```
W22046071 >
W22046071 > SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY());

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------
-----------
Plan hash value: 2124558393

-----------------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name                 | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                      |   261K|    13M|   569   (2)| 00:00:01 |
|   1 |  MAT_VIEW REWRITE ACCESS FULL| CUSTOMER_SALES_TIME_MV |   261K|    13M|   569   (2)| 00:00:01 |
-----------------------------------------------------------------------------------------------------------

8 rows selected.

W22046071 >
W22046071 >
```

MV 2 in SH2:

```
EXPLAIN PLAN FOR
SELECT COUNT(S.cust_id) AS SalesCustCount,C.channel_class, S.time_id,
P.supplier_id, P.prod_category, S.prod_id
FROM SH2.products P, SH2.Channels C, SH2.sales S
WHERE P.prod_id = S.prod_id
AND C.channel_id = S.channel_id
GROUP BY C.channel_class, S.time_id, P.supplier_id, P.prod_category, S.prod_id;
```

```
W22046071 >
W22046071 > EXPLAIN PLAN FOR
  2  SELECT COUNT(S.cust_id) AS SalesCustCount,C.channel_class, S.time_id, P.supplier_id, P.prod_category, S.prod_id
  3  FROM SH2.products P, SH2.Channels C, SH2.sales S
  4  WHERE P.prod_id = S.prod_id
  5  AND C.channel_id = S.channel_id
  6  GROUP BY C.channel_class, S.time_id, P.supplier_id, P.prod_category, S.prod_id;

Explained.

W22046071 >
W22046071 >
```

```
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY());
```

```
W22046071 > SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY());

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------------------
-----------
Plan hash value: 4120225454

-----------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                          | Name                  | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     | Pstart| Pstop |
-----------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |                       |  1016K|    39M|       | 13021   (1)| 00:00:01 |       |       |
|   1 |  HASH GROUP BY                     |                       |  1016K|    39M|    50M| 13021   (1)| 00:00:01 |       |       |
|*  2 |   HASH JOIN                        |                       |  1016K|    39M|       |  2194   (2)| 00:00:01 |       |       |
|   3 |    VIEW                            | index$_join$_001      | 10000 |   146K|       |    74   (2)| 00:00:01 |       |       |
|*  4 |     HASH JOIN                      |                       |       |       |       |            |          |       |       |
|*  5 |      HASH JOIN                     |                       |       |       |       |            |          |       |       |
|   6 |       INDEX FAST FULL SCAN         | PRODUCTS_PK           | 10000 |   146K|       |    28   (0)| 00:00:01 |       |       |
|   7 |       INDEX FAST FULL SCAN         | PRODUCTS_PROD_CAT_IX  | 10000 |   146K|       |    30   (0)| 00:00:01 |       |       |
|   8 |      BITMAP CONVERSION TO ROWIDS   |                       | 10000 |   146K|       |     5   (0)| 00:00:01 |       |       |
|   9 |       BITMAP INDEX FULL SCAN       | PRODUCTS_SUPPLIER_BIX |       |       |       |            |          |       |       |
|* 10 |    HASH JOIN                       |                       |  1016K|    25M|       |  2112   (2)| 00:00:01 |       |       |
|  11 |     VIEW                           | index$_join$_002      |     5 |    55 |       |     2   (0)| 00:00:01 |       |       |
|* 12 |      HASH JOIN                     |                       |       |       |       |            |          |       |       |
|  13 |       BITMAP CONVERSION TO ROWIDS  |                       |     5 |    55 |       |     1   (0)| 00:00:01 |       |       |
|  14 |        BITMAP INDEX FULL SCAN      | IDX_CHANNEL_CLASS     |       |       |       |            |          |       |       |
|  15 |       INDEX FAST FULL SCAN         | CHAN_PK               |     5 |    55 |       |     1   (0)| 00:00:01 |       |       |
|  16 |     PARTITION RANGE ALL            |                       |  1016K|    14M|       |  2102   (2)| 00:00:01 |     1 |    17 |
|  17 |      TABLE ACCESS FULL             | SALES                 |  1016K|    14M|       |  2102   (2)| 00:00:01 |     1 |    17 |
-----------------------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("P"."PROD_ID"="S"."PROD_ID")
   4 - access(ROWID=ROWID)
   5 - access(ROWID=ROWID)
  10 - access("C"."CHANNEL_ID"="S"."CHANNEL_ID")
  12 - access(ROWID=ROWID)

33 rows selected.

W22046071 >
```

MV 2 in DWU:

```
EXPLAIN PLAN FOR
SELECT COUNT(S.cust_id) AS SalesCustCount,C.channel_class, S.time_id,
P.supplier_id, P.prod_category, S.prod_id
FROM DWU390.products P, DWU390.Channels C, DWU390.sales S
WHERE P.prod_id = S.prod_id
AND C.channel_id = S.channel_id
GROUP BY C.channel_class, S.time_id, P.supplier_id, P.prod_category, S.prod_id;
```

```
W22046071 >
W22046071 > EXPLAIN PLAN FOR
  2  SELECT COUNT(S.cust_id) AS SalesCustCount,C.channel_class, S.time_id, P.supplier_id, P.prod_category, S.prod_id
  3  FROM DWU390.products P, DWU390.Channels C, DWU390.sales S
  4  WHERE P.prod_id = S.prod_id
  5  AND C.channel_id = S.channel_id
  6  GROUP BY C.channel_class, S.time_id, P.supplier_id, P.prod_category, S.prod_id;

Explained.

W22046071 >
W22046071 >
```

```
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY());
```

```
W22046071 >
W22046071 > SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY());

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------
-----------
Plan hash value: 3986459439

-----------------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                     |   908K|    27M|  1245   (2)| 00:00:01 |
|   1 |  MAT_VIEW REWRITE ACCESS FULL| SALES_PRMO_CHANNEL_MV |   908K|    27M|  1245   (2)| 00:00:01 |
-----------------------------------------------------------------------------------------------------------

8 rows selected.

W22046071 >
W22046071 > _
```

Provide Discussion of the cost-based comparison of the above 2 sets of queries and their explain plan cost figures / values (6 marks):

MV1:

In this Materialized view 1, the cost of SH2 is 2102 and the cost of DWU is 569. The cost is high in SH2 since materialised view is not created whereas the cost is low in DWU because of the creation of materialised view called the customer sales time.

MV2:

In this MV2, materialised view was not created for SH2 and the cost is 2012. The cost of DWU is 1245 because of the materialised view created for sales promotion channel in DWU.

References

1. Griff, O. (2015). *Advantages and disadvantages of using Entity Framework - CodeProject*. [online] Codeproject.com. Available at: https://www.codeproject.com/Questions/1013270/Advantages-and-disadvantages-of-using-Entity-Frame.

2. Kannan, P.K. (2019) Oracle Database Object-Relational Developer's Guide -19c. Part Number E96436-01. Available at: https://docs.oracle.com/en/database/oracle/oracle-database/19/adobj/index.html (Accessed: 25 January 2023).

3. Mishra, S. (2022). *Object-Oriented Databases And Their Advantages | HackerNoon*. [online] hackernoon.com. Available at: https://hackernoon.com/object-oriented-databases-and-their-advantages.

4. Morin, L. (2017) Oracle Database PL/SQL Language Reference, 12*c* Release 1 (12.1). Available at: https://docs.oracle.com/database/121/LNPLS/title.htm (Accessed: 17 January 2023).

5. Northumbria (2020) Academic Regulations for Taught Awards. Available at: https://www.northumbria.ac.uk/about-us/university-services/student-library-and-academic-services/quality-and-teaching-excellence/assessment/guidance-for-students/ (Accessed: 27 January 2023).

6. Pears, R. and Shields, G. (2008) *Cite them right: the essential referencing guide*. Newcastle upon Tyne: Pear Tree Books. Available at: https://www.citethemrightonline.com/ (Accessed: 27 January 2023).

7. Potineni, P. (2017) Oracle Database Data Warehousing Guide, 12c Release 1 (12.1). Part Number E41670-11. Available at: https://docs.oracle.com/database/121/DWHSG/ (Accessed: 16 October 2022).