CIT6234 - Advanced Database

Assignment 1 (30%)

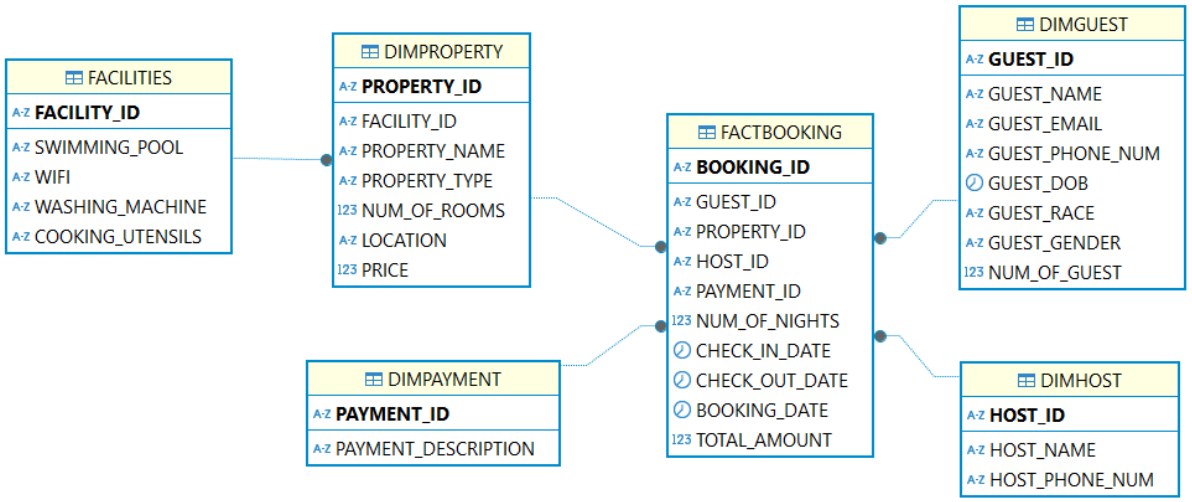
Title : Data Warehouse for Airbnb Property Booking

Tutorial Section : TT2L

	Student ID	Name
Leader	1211111809	Maryam Binti Norazman
Member	1211112284	Nur Dania Iman Binti Desman Desa
Member	1221103588	Nurul Syahafiza Binti Naziron
Member	1231303620	Nur Iman Binti Mohamad Idros

TABLE OF CONTENTS

1.0 Conceptual Schema



2.0 Data Dictionary

Table Name	Attribute Name	Contents	Type	Format	Range	Required	PK or FK	FK Reference Table
Facilities	Facility_ID	Unique ID for the facility	Varchar (10)	X999	F001-F999	Y	PK	
	Swimming_Pool	Yes/No	Varchar (3)	Xxx	Yes, No			
	Wifi	Yes/No	Varchar (3)	Xxx	Yes, No			
	Washing_Machine	Yes/No	Varchar (3)	Xxx	Yes, No			
	Cooking_Utensils	Yes/No	Varchar (3)	Xxx	Yes, No			
DimProperty	Property_ID	Unique identifier for each property	Varchar (10)	X999	P001-P999	Y	PK	
	Property_Name	Name of the property	Varchar (100)	Xxxxxxxxxxxxx		Y		
	Property_Type	Type (e.g., apartment, house)	Varchar (20)	Xxxxxxxxxxxxx		Y		
	Num_of_Rooms	Total number of rooms	Int(2)	99	1-99	Y		
	Location	Address/location of the property	Varchar (100)	Xxxxxxxxxxxxx		Y		
	Price	Base price per night	Decimal (10,2)	99999.99		Y		
	Facility_ID	Link to available facilities	Varchar (10)	X999	F001-F999	Y	FK	Facilities

DimPayment	Payment_ID	Unique ID for each payment method	Varchar (10)	XX999	PM001-PM999	Y	PK	
	Payment_Description	e.g., Cash, Debit Card, Credit Card	Varchar (20)	Xxxxxxxxxxxxx		Y		
FactBooking	Booking_ID	Unique ID for each booking	Varchar(10)	X999	B001-B999	Y	PK	
	Guest_ID	Guest who made the booking	Varchar (10)	X999	G001-G999	Y	FK	DimGuest
	Property_ID	Booked property	Varchar (10)	X999	P001-P999	Y	FK	DimProperty
	Host_ID	Host offering the property	Varchar (10)	X999	H001-H999	Y	FK	DimHost
	Payment_ID	Payment method used	Varchar (10)	XX999	PM001-PM999	Y	FK	DimPayment
	Total_Amount	Total price for the booking	Decimal (10,2)	99999.99		Y		
	Num_of_Nights	Duration of stay in nights	Int(2)	99	1-99	Y		
	Check_In_Date	Start of stay	Date	YYYY-MM-DD		Y		
	Check_Out_Date	End of stay	Date	YYYY-MM-DD		Y		
	Booking Date	Date the booking was made	Date	YYYY-MM-DD		Y		
DimGuest	Guest_ID	Unique identifier for each guest	Varchar (10)	X999	G001-G999	Y	PK	
	Guest_Name	Full name of the guest	Varchar (100)	Xxxxxxxxxxxxx		Y		
	Guest_Email	Guest's email address	Varchar (100)	Xxxxxxxxxxxxx		Y		

	Guest_Phone_Num	Guest's contact number	Varchar (15)	999-999-9999		Y		
	Guest_DOB	Guest's date of birth	Date	YYYY-MM-DD				
	Guest_Race	Race of the guest	Varchar (20)	Xxxxxxxxxxxxx				
	Guest_Gender	Gender of the guest	Varchar (6)	X		Y		
	Num_of_Guest	Number of guests for the booking	Int(2)	99	1-99	Y		
DimHost	Host_ID	Unique ID for the host	Varchar (10)	X999	H001-H999	Y	PK	
	Host_Name	Name of the host	Varchar (100)	Xxxxxxxxxxxxx		Y		
	Host_Phone_Num	Contact number of the host	Varchar (15)	999-999-9999		Y		

3.0 Calculation fact table size and storage in the database schema.

- DimGuest = 15 records
- DimProperty = 8 records
- DimHost = 5 records
- DimPayment = 6 records

Total rows of Fact Table (FactBooking)

DimGuest records × DimProperty records × DimHost records × DimPayment records

= 15 × 8 × 5 × 6

= 3,600 rows

Average bytes per field (FactBooking)

Each estimated sizes:

- Booking_ID (VARCHAR(10)) = 11 bytes
- Guest_ID (VARCHAR(10)) = 11 bytes
- Property_ID (VARCHAR(10)) = 11 bytes
- Host_ID (VARCHAR(10)) = 11 bytes
- Payment_ID (VARCHAR(10)) = 11 bytes
- Num_of_Nights (INT) = 5 bytes
- Check_In_Date (DATE) = 4 bytes
- Check_Out_Date (DATE) = 4 bytes
- Booking_Date (DATE) = 4 bytes
- Total_Amount (DECIMAL(10,2)) = 5 bytes

Total estimated sizes

= 11 + 11 + 11 + 11 + 11 + 5 + 4 + 4 + 4 + 5

= 77 bytes

Average Bytes per Field

= Total Bytes / Number of Attributes
= 77 / 10
= **7.7** bytes

Total Storage Size for Fact Table (FactBooking)

Total Storage

= Total Rows × Number of Attributes × Average Bytes per Field

= 3,600 × 10 × 7.7

= 277,200 bytes

= 0.277 MB

4.0 Data Definition Language (DDL)

4.1 Table Name : DimGuest

```
CREATE TABLE DimGuest (
    Guest_ID VARCHAR(10) PRIMARY KEY NOT NULL,
    Guest_Name VARCHAR(100) NOT NULL,
    Guest_Email VARCHAR(100) NOT NULL,
    Guest_Phone_Num VARCHAR(15) NOT NULL,
    Guest_DOB DATE NOT NULL ,
    Guest_Race VARCHAR(20),
    Guest_Gender VARCHAR(6) CHECK (Guest_Gender IN ('Male', 'Female')),
    Num_of_Guest INT NOT NULL CHECK (Num_of_Guest > 0)
);
```

4.2 Table Name : DimHost

```
CREATE TABLE DimHost (
    Host_ID  VARCHAR(10) PRIMARY KEY NOT NULL,
    Host_Name VARCHAR(100) NOT NULL,
    Host_Phone_Num VARCHAR(15) NOT NULL
);
```

4.3 Table Name : DimPayment

```
CREATE TABLE DimPayment (
    Payment_ID VARCHAR(10) PRIMARY KEY NOT NULL,
    Payment_Description VARCHAR(20) NOT NULL CHECK (Payment_Description IN ('Credit Card', 'Debit Card', 'E-Wallet', 'Online Banking', 'Cash', 'Touch n Go'))
);
```

4.4 Table Name : Facilities

```
CREATE TABLE Facilities (
    Facility_ID VARCHAR(10) PRIMARY KEY NOT NULL,
    Swimming_Pool VARCHAR(3) NOT NULL CHECK (Swimming_Pool IN ('Yes', 'No')),
    Wifi VARCHAR(3) NOT NULL CHECK (Wifi IN ('Yes', 'No')),
    Washing_Machine VARCHAR(3) NOT NULL CHECK (Washing_Machine IN ('Yes', 'No')),
    Cooking_Utensils VARCHAR(3) NOT NULL CHECK (Cooking_Utensils IN ('Yes', 'No'))
);
```

4.5 Table Name : DimProperty

```
CREATE TABLE DimProperty (
    Property_ID VARCHAR(10) PRIMARY KEY NOT NULL,
    Facility_ID VARCHAR (10) NOT NULL,
    Property_Name VARCHAR(100) NOT NULL,
    Property_Type VARCHAR(20) NOT NULL,
    Num_of_Rooms INT NOT NULL,
    Location VARCHAR(100) NOT NULL,
    Price DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (Facility_ID) REFERENCES Facilities
);
```

4.6 Table Name : FactBooking

```
CREATE TABLE FactBooking (
    Booking_ID VARCHAR(10) PRIMARY KEY NOT NULL,
    Guest_ID VARCHAR(10) NOT NULL,
    Property_ID VARCHAR(10) NOT NULL,
    Host_ID VARCHAR(10) NOT NULL,
    Payment_ID VARCHAR(10) NOT NULL,
    Num_of_Nights INT NOT NULL,
    Check_In_Date DATE NOT NULL,
    Check_Out_Date DATE NOT NULL,
    Booking_Date DATE NOT NULL,
```

```
    Total_Amount DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (Guest_ID) REFERENCES DimGuest,
    FOREIGN KEY (Property_ID) REFERENCES DimProperty,
    FOREIGN KEY (Host_ID) REFERENCES DimHost,
    FOREIGN KEY (Payment_ID) REFERENCES DimPayment
);
```

5.0 Data Manipulation Language

5.1 Table Name : DimGuest

```
INSERT INTO DimGuest VALUES
('G001', 'Alice Tan', 'alicetan@gmail.com', '0123456789', '1995-04-12', 'Chinese', 'Female', 2),
('G002', 'Muhamad Adam bin Muhamad Samad', 'adamsamad@gmail.com', '0198765432', '1988-06-23', 'Malay', 'Male', 10),
('G003', 'Sarah Kumar', 'sarahkumar@gmail.com', '0172223333', '1992-09-11', 'Indian', 'Female', 3),
('G004', 'Michael Wong', 'michaelwong@yahoo.com', '0111234567', '1980-12-05', 'Chinese', 'Male', 4),
('G005', 'Nur Aisyah binti Kamal', 'aisyahkamal@gmail.com', '0169988776', '1997-03-09', 'Malay', 'Female', 7),
('G006', 'Daniel Harriz bin Junaidi', 'danielharriz@yahoo.com', '0141122334', '1993-01-22', 'Malay', 'Male', 5),
('G007', 'Farah Zain binti Rahmat', 'farah@gmail.com', '0183344556', '1985-07-14', 'Malay', 'Female', 8),
('G008', 'Kumar Raj', 'kumar@gmail.com', '0132233445', '1978-11-30', 'Indian', 'Male', 4),
('G009', 'Lim Mei', 'mei@yahoo.com', '0123451111', '2000-05-20', 'Chinese', 'Female', 2),
('G010', 'Ariff Ismail bin Azfar Imran', 'ariff@yahoo.com', '0191112223', '1990-10-17', 'Malay', 'Male', 3),
('G011', 'Sophia Chen', 'sophia.chen@yahoo.com', '0165556677', '1991-08-25', 'Chinese', 'Female', 2),
('G012', 'Ahmad Faisal bin Ismail', 'ahmadfaisal@gmail.com', '0178899001', '1987-11-14', 'Malay', 'Male', 4),
('G013', 'Priya Devi', 'priya.devi@gmail.com', '0192233445', '1994-02-28', 'Indian', 'Female', 3),
('G014', 'Jason Lim', 'jasonlim@yahoo.com', '0119988776', '1983-07-19', 'Chinese', 'Male', 5),
('G015', 'Nurul Huda binti Mazlan', 'nurulhuda@gmail.com', '0181122334', '1998-04-05', 'Malay', 'Female', 2);
```

5.2 Table Name : DimHost

INSERT INTO DimHost VALUES

('H001', 'Linda Lee', '0131234567'),

('H002', 'Rahim Aziz', '0149876543'),

('H003', 'Kenny Ong', '0123344556'),

('H004', 'Faridah Musa', '0167788990'),

('H005', 'Vinod Singh', '0191122334');

5.4 Table Name : DimPayment

INSERT INTO DimPayment VALUES

('PM001', 'Touch n Go'),

('PM002', 'Online Banking'),

('PM003', 'E-Wallet'),

('PM004', 'Cash'),

('PM005', 'Debit Card'),

('PM006', 'Credit Card');

5.5 Table Name : Facilities

INSERT INTO Facilities VALUES

('F001', 'No', 'Yes', 'Yes', 'Yes'),

('F002', 'Yes', 'Yes', 'Yes', 'Yes'),

('F003', 'Yes', 'No', 'No', 'Yes'),

('F004', 'No', 'Yes', 'No', 'No'),

('F005', 'Yes', 'Yes', 'No', 'Yes'),

('F006', 'No', 'Yes', 'No', 'Yes'),

('F007', 'Yes', 'Yes', 'Yes', 'Yes'),

('F008', 'Yes', 'Yes', 'No', 'Yes');

5.6 Table Name : DimProperty

INSERT INTO DimProperty VALUES

('P001', 'F001', 'Hilltop Bungalow', 'Bungalow', 5, 'Kuala Lumpur', 950.00),

('P002', 'F002', 'Seaview Condo', 'Condominium', 2, 'Penang', 280.00),

('P003', 'F003', 'Sunset Villa', 'Villa', 3, 'Putrajaya', 1100.00),

('P004', 'F004', 'Garden Bungalow', 'Bungalow', 3, 'Johor Bahru', 870.00),

('P005', 'F005', 'Skyline Residence', 'Condominium', 3, 'Melaka', 300.00),

('P006', 'F006', 'Ocean Breeze Villa', 'Villa', 5, 'Langkawi', 1050.00),

('P007', 'F007', 'Serene Bungalow', 'Bungalow', 4, 'Kuala Lumpur', 920.00),

('P008', 'F008', 'Urban Heights Condo', 'Condominium', 3, 'Putrajaya', 310.00);

5.7 Table Name : FactBooking

INSERT INTO FactBooking VALUES

('B001', 'G003', 'P005', 'H002', 'PM004', 3, '2023-03-10', '2023-03-13', '2023-02-25', 900.00),

('B002', 'G007', 'P002', 'H004', 'PM001', 2, '2023-05-15', '2023-05-17', '2023-04-28', 560.00),

('B003', 'G001', 'P007', 'H001', 'PM006', 5, '2023-01-22', '2023-01-27', '2022-12-15', 4600.00),

('B004', 'G010', 'P004', 'H003', 'PM002', 4, '2023-06-08', '2023-06-12', '2023-05-20', 3480.00),

('B005', 'G005', 'P008', 'H005', 'PM005', 1, '2023-04-03', '2023-04-04', '2023-03-20', 310.00),

('B006', 'G002', 'P003', 'H002', 'PM003', 7, '2023-07-14', '2023-07-21', '2023-06-01', 7700.00),

('B007', 'G008', 'P006', 'H004', 'PM001', 2, '2023-02-18', '2023-02-20', '2023-01-30', 2100.00),

('B008', 'G004', 'P001', 'H001', 'PM004', 3, '2023-08-05', '2023-08-08', '2023-07-15', 2850.00),

('B009', 'G009', 'P005', 'H003', 'PM006', 4, '2023-09-12', '2023-09-16', '2023-08-22', 1200.00),

('B010', 'G006', 'P002', 'H005', 'PM002', 5, '2023-10-01', '2023-10-06', '2023-09-10', 1400.00),

('B011', 'G011', 'P003', 'H003', 'PM005', 2, '2023-11-08', '2023-11-10', '2023-10-15', 2200.00),

('B012', 'G012', 'P006', 'H004', 'PM003', 3, '2023-12-12', '2023-12-15', '2023-11-20', 3150.00),

('B013', 'G013', 'P001', 'H005', 'PM002', 4, '2024-01-05', '2024-01-09', '2023-12-10', 3800.00),

('B014', 'G014', 'P007', 'H001', 'PM004', 1, '2024-02-14', '2024-02-15', '2024-01-25', 920.00),

('B015', 'G015', 'P004', 'H002', 'PM006', 5, '2024-03-20', '2024-03-25', '2024-02-28', 4350.00),

('B016', 'G003', 'P008', 'H003', 'PM001', 2, '2024-04-10', '2024-04-12', '2024-03-15', 620.00),

('B017', 'G007', 'P005', 'H004', 'PM002', 3, '2024-05-18', '2024-05-21', '2024-04-20', 900.00),

('B018', 'G010', 'P002', 'H005', 'PM005', 4, '2024-06-22', '2024-06-26', '2024-05-30', 1120.00),

('B019', 'G002', 'P001', 'H001', 'PM003', 2, '2024-07-07', '2024-07-09', '2024-06-10', 1900.00),

('B020', 'G009', 'P003', 'H002', 'PM004', 6, '2024-08-15', '2024-08-21', '2024-07-25', 6600.00);

6.0 Procedural SQL

6.1 Stored Procedure

6.1.1 Total Bookings by Host

Stored Procedure :	Host_Booking_Count
Purpose :	Show how many bookings each host has handled.
Query :	CREATE PROCEDURE Host_Booking_Count () LANGUAGE SQL DYNAMIC RESULT SETS 1 BEGIN DECLARE result CURSOR WITH RETURN FOR SELECT H.Host_ID, H.Host_Name, COUNT(B.Booking_ID) AS Total_Bookings FROM DimHost H LEFT JOIN FactBooking B ON H.Host_ID = B.Host_ID GROUP BY H.Host_ID, H.Host_Name ORDER BY Total_Bookings DESC; OPEN result; END
Example Calls :	CALL Host_Booking_Count();
Output :	DIMHOST 1 × CALL Host_Booking_Count() Enter a SQL expression to filter results Grid Text

	HOST_ID	HOST_NAME	TOTAL_BOOKINGS
1	H001	Linda Lee	4
2	H002	Rahim Aziz	4
3	H003	Kenny Ong	4
4	H004	Faridah Musa	4
5	H005	Vinod Singh	4

Explanation :	It joins DimHost with FactBooking using a LEFT JOIN then uses COUNT(B.Booking_ID) to count the number of bookings per host. It use GROUP BY host to get results per host.Then using ORDER BY Total_Bookings DESC to get the result in descending order.

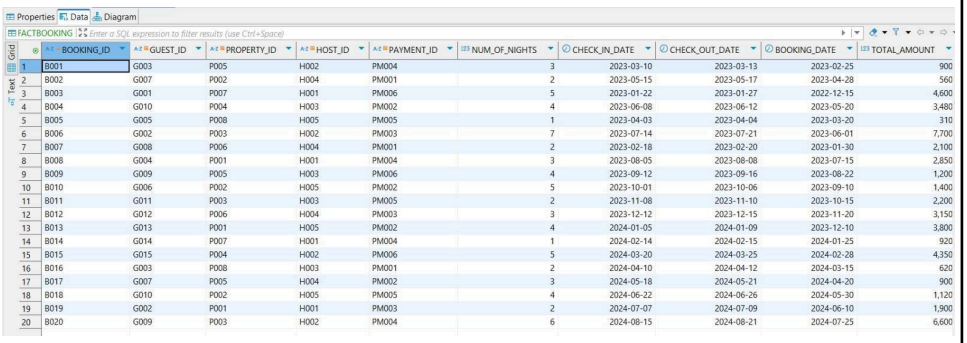
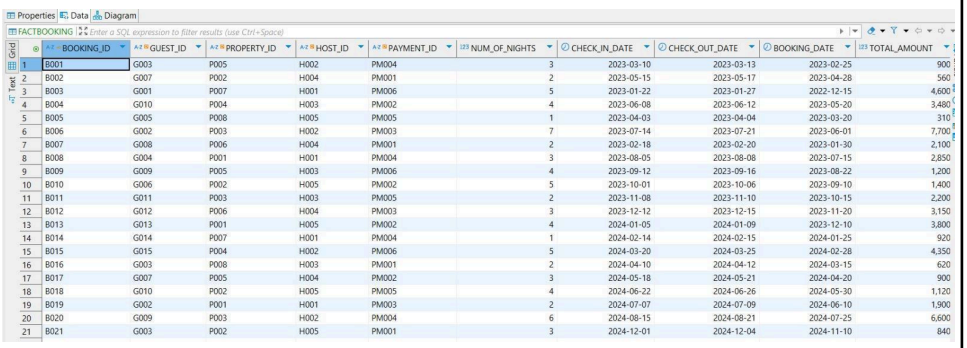
6.1.2 Available Properties By Location

Stored Procedure :	Available_Properties_By_Location
Purpose :	Returns a list of available Airbnb properties in a given location and date range, including their prices and facility details. It filters out properties that are already booked during that period.
Query :	CREATE PROCEDURE Available_Properties_By_Location (IN input_location VARCHAR(100), IN start_date DATE, IN end_date DATE) LANGUAGE SQL DYNAMIC RESULT SETS 1 BEGIN DECLARE result CURSOR WITH RETURN FOR SELECT P.Property_ID, P.Property_Name, P.Price, F.Swimming_Pool, F.Wifi, F.Washing_Machine, F.Cooking_Utensils FROM DimProperty P JOIN Facilities F ON P.Facility_ID = F.Facility_ID WHERE P.Location = input_location AND P.Property_ID NOT IN (SELECT Property_ID

	FROM FactBooking FB WHERE FB.Check_In_Date <= end_date AND FB.Check_Out_Date >= start_date) ORDER BY P.Price; OPEN result; END
Example Calls :	CALL Available_Properties_By_Location('Kuala Lumpur', '2024-06-01', '2024-06-05');
Output :	
Explanation :	It checks all properties in the given location using WHERE P.Location = input_location then filters out those where bookings overlap with the input date range using AND P.Property_ID NOT IN and then joins with FACILITIES using JOIN Facilities F ON P.Facility_ID = F.Facility_ID to show facility features in the result lastly using ORDER BY P.Price to have ascending order according to the price.

6.2 Trigger

Trigger :	Calc_Total_Amount
Purpose :	Automatically calculates Total_Amount before inserting a new booking, based on the property's price and number of nights.
Query :	CREATE TRIGGER Calc_Total_Amount BEFORE INSERT ON FactBooking REFERENCING NEW AS newRow FOR EACH ROW MODE DB2SQL BEGIN DECLARE nightly_rate DECIMAL(10, 2); SELECT Price INTO nightly_rate FROM DimProperty WHERE Property_ID = newRow.Property_ID;

	SET newRow.Total_Amount = nightly_rate * newRow.Num_of_Nights; END
Example Calls :	INSERT INTO FactBooking (Booking_ID, Guest_ID, Property_ID, Host_ID, Payment_ID, Num_of_Nights, Check_In_Date, Check_Out_Date, Booking_Date) VALUES ('B021', 'G003', 'P002', 'H005', 'PM001', 3, '2024-12-01', '2024-12-04', '2024-11-10');
Output :	Before :  After : 
Explanation :	Triggers before a booking is inserted using BEFORE INSERT ON FactBooking then fetches property price using the Property_ID then multiplies it by Num_of_Nights and assigns it to Total_Amount.

6.3 User-defined Function

User-Defined Function :	GetAgeGroup
Purpose :	Returns a guest's age group based on their date of birth (e.g. 18–25, 26–35, etc.).
Quer	CREATE FUNCTION GetAgeGroup (dob DATE) RETURNS VARCHAR(20) DETERMINISTIC LANGUAGE SQL BEGIN RETURN CASE WHEN (YEAR(CURRENT DATE) - YEAR(dob) - CASE WHEN MONTH(CURRENT DATE) < MONTH(dob) OR (MONTH(CURRENT DATE) = MONTH(dob) AND DAY(CURRENT DATE) < DAY(dob)) THEN 1 ELSE 0 END) BETWEEN 18 AND 25 THEN '18-25' WHEN (YEAR(CURRENT DATE) - YEAR(dob) - CASE WHEN MONTH(CURRENT DATE) < MONTH(dob) OR (MONTH(CURRENT DATE) = MONTH(dob) AND DAY(CURRENT DATE) < DAY(dob)) THEN 1 ELSE 0 END) BETWEEN 26 AND 35 THEN '26-35' WHEN (YEAR(CURRENT DATE) - YEAR(dob) - CASE WHEN MONTH(CURRENT DATE) < MONTH(dob) OR (MONTH(CURRENT DATE) = MONTH(dob) AND DAY(CURRENT DATE) < DAY(dob)) THEN 1 ELSE 0 END) BETWEEN 36 AND 45 THEN '36-45' WHEN (YEAR(CURRENT DATE) - YEAR(dob) - CASE WHEN MONTH(CURRENT DATE) < MONTH(dob) OR (MONTH(CURRENT DATE) = MONTH(dob) AND DAY(CURRENT DATE) < DAY(dob))

```
            THEN 1 ELSE 0 END) BETWEEN 46 AND 60 THEN '46-60'


        WHEN (YEAR(CURRENT DATE) - YEAR(dob)

            - CASE WHEN MONTH(CURRENT DATE) < MONTH(dob)

                OR (MONTH(CURRENT DATE) = MONTH(dob) AND
DAY(CURRENT DATE) < DAY(dob) )

            THEN 1 ELSE 0 END) > 60 THEN '60+'

        ELSE 'Under 18'

    END;

END
```

Example Calls :

```
SELECT
    Age_Group,
    Guest_Name,
    Guest_DOB
FROM (
    SELECT
        Guest_Name,
        Guest_DOB,
        GetAgeGroup(Guest_DOB) AS Age_Group
    FROM DimGuest
) AS AgeCategorized
ORDER BY Age_Group, Guest_DOB;
```

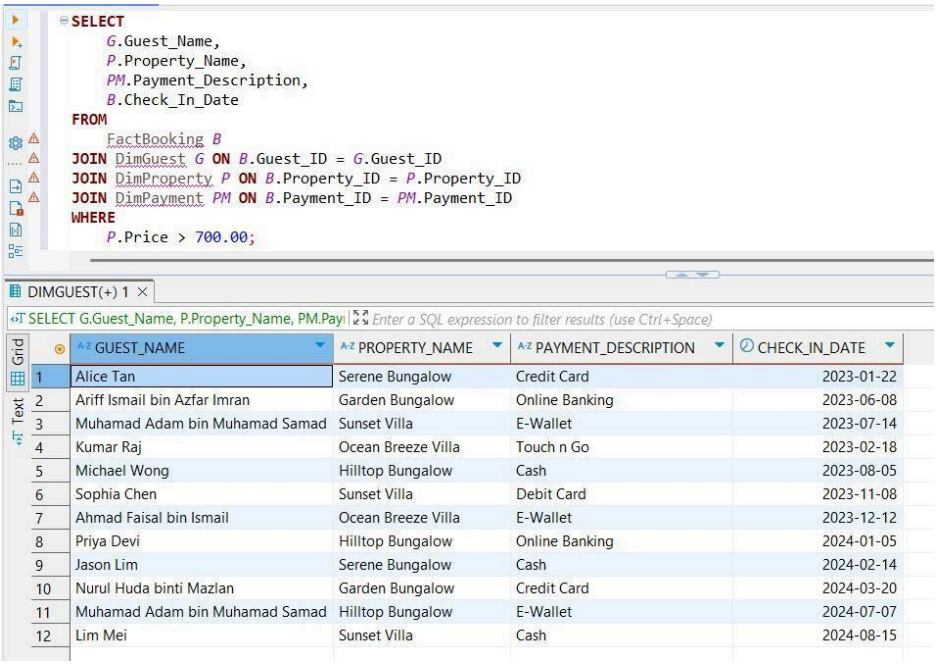
Output :

	AGE_GROUP	GUEST_NAME	GUEST_DOB
1	18-25	Lim Mei	2000-05-20
2	26-35	Ariff Ismail bin Azfar Imran	1990-10-17
3	26-35	Sophia Chen	1991-08-25
4	26-35	Sarah Kumar	1992-09-11
5	26-35	Daniel Harriz bin Junaidi	1993-01-22
6	26-35	Priya Devi	1994-02-28
7	26-35	Alice Tan	1995-04-12
8	26-35	Nur Aisyah binti Kamal	1997-03-09
9	26-35	Nurul Huda binti Mazlan	1998-04-05
10	36-45	Michael Wong	1980-12-05
11	36-45	Jason Lim	1983-07-19
12	36-45	Farah Zain binti Rahmat	1985-07-14
13	36-45	Ahmad Faisal bin Ismail	1987-11-14
14	36-45	Adam bin Muhamad Samad	1988-06-23
15	46-60	Kumar Raj	1978-11-30

Explanation :	It calculates the age by subtracting dob from current date while adjusting for month/day then maps the age to a group. The example call used to call the function then used ORDER BY Age_Group, Guest_DOB to have it in ascending order when forming the result.

7.0 Special SQL Command

7.1 Complex query with joins of at least 3 tables

Query :	SELECT G.Guest_Name, P.Property_Name, PM.Payment_Description, B.Check_In_Date FROM FactBooking B JOIN DimGuest G ON B.Guest_ID = G.Guest_ID JOIN DimProperty P ON B.Property_ID = P.Property_ID JOIN DimPayment PM ON B.Payment_ID = PM.Payment_ID WHERE P.Price > 700.00;
Output :	
Explanation :	This query joins four tables (Guest, Booking, Property and Payment Method) to list all booking for properties priced above RM700. It shows the guest's name, property name, payment description, and check-in date. This query is useful for identifying high-value bookings made by guests.

	GUEST_NAME	PROPERTY_NAME	PAYMENT_DESCRIPTION	CHECK_IN_DATE
1	Alice Tan	Serene Bungalow	Credit Card	2023-01-22
2	Ariff Ismail bin Azfar Imran	Garden Bungalow	Online Banking	2023-06-08
3	Muhamad Adam bin Muhamad Samad	Sunset Villa	E-Wallet	2023-07-14
4	Kumar Raj	Ocean Breeze Villa	Touch n Go	2023-02-18
5	Michael Wong	Hilltop Bungalow	Cash	2023-08-05
6	Sophia Chen	Sunset Villa	Debit Card	2023-11-08
7	Ahmad Faisal bin Ismail	Ocean Breeze Villa	E-Wallet	2023-12-12
8	Priya Devi	Hilltop Bungalow	Online Banking	2024-01-05
9	Jason Lim	Serene Bungalow	Cash	2024-02-14
10	Nurul Huda binti Mazlan	Garden Bungalow	Credit Card	2024-03-20
11	Muhamad Adam bin Muhamad Samad	Hilltop Bungalow	E-Wallet	2024-07-07
12	Lim Mei	Sunset Villa	Cash	2024-08-15

7.2 Group by/Group by Rollup/Group by Cube and having clause

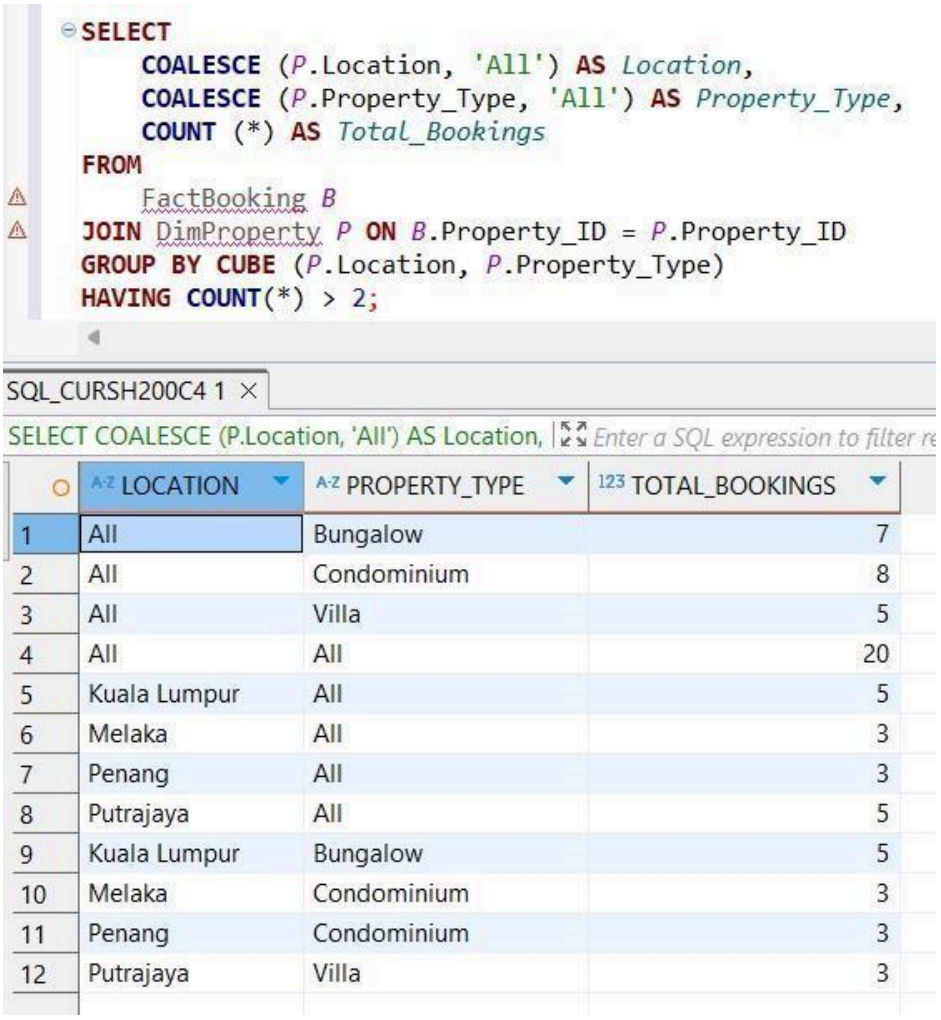
Query :	SELECT COALESCE (P.Location, 'All') AS Location, COALESCE (P.Property_Type, 'All') AS Property_Type, COUNT (*) AS Total_Bookings FROM FactBooking B JOIN DimProperty P ON B.Property_ID = P.Property_ID GROUP BY CUBE (P.Location, P.Property_Type) HAVING COUNT(*) > 2;
Output :	
Explanation :	This query uses GROUP BY CUBE to create grouped results by all combinations of Location and Property Type, including subtotals and grand totals. The COALESCE function replaces NULL values (from cube totals) with 'All' for readability. The HAVING clause filters out combinations with fewer than 3 bookings, focusing the result on more popular or frequently booked categories.

HIGHVALUEGUESTS | Enter a SQL expression to filter results (use Ctrl+Space)

	GUEST_ID	GUEST_NAME	NUMOFBOOKINGS	MAXPRICEBOOKED
1	G001	Alice Tan	1	920
2	G002	Muhamad Adam bin Muhamad Samad	2	1,100
3	G004	Michael Wong	1	950
4	G008	Kumar Raj	1	1,050
5	G009	Lim Mei	2	1,100
6	G010	Ariff Ismail bin Azfar Imran	2	870
7	G011	Sophia Chen	1	1,100
8	G012	Ahmad Faisal bin Ismail	1	1,050
9	G013	Priya Devi	1	950
10	G014	Jason Lim	1	920
11	G015	Nurul Huda binti Mazlan	1	870

7.3 View

Query :	CREATE VIEW HighValueGuests AS SELECT G.Guest_ID, G.Guest_Name, COUNT(*) AS NumofBookings,

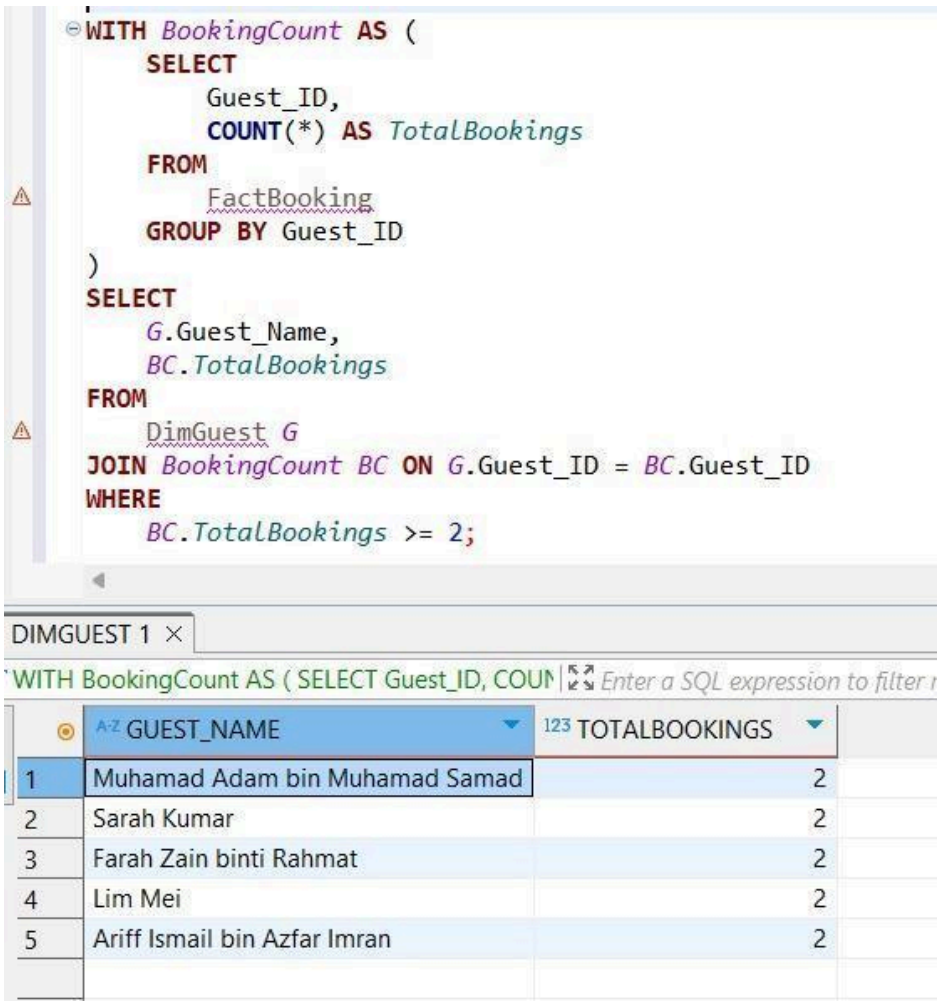
	MAX(P.Price) AS MaxPriceBooked FROM FactBooking B JOIN DimGuest G ON B.Guest_ID = G.Guest_ID JOIN DimProperty P ON B.Property_ID = P.Property_ID GROUP BY G.Guest_ID, G.Guest_Name HAVING MAX(P.Price) > 500;
Output :	
Explanation :	This view defines a virtual table called HighValueGuests that stored data for guests who have booked at least one property priced above RM500. It includes their guest ID, name, total number of bookings, and the highest price they booked. This is helpful for identifying premium customers for loyalty programs or targeted promotions.

	LOCATION	PROPERTY_TYPE	TOTAL_BOOKINGS
1	All	Bungalow	7
2	All	Condominium	8
3	All	Villa	5
4	All	All	20
5	Kuala Lumpur	All	5
6	Melaka	All	3
7	Penang	All	3
8	Putrajaya	All	5
9	Kuala Lumpur	Bungalow	5
10	Melaka	Condominium	3
11	Penang	Condominium	3
12	Putrajaya	Villa	3

7.4 TWO SQL not covered in lecture

7.4.1 Common Table Expression (CTE)

Query :	WITH BookingCount AS (SELECT Guest_ID, COUNT(*) AS TotalBookings FROM FactBooking GROUP BY Guest_ID) SELECT G.Guest_Name, BC.TotalBookings FROM DimGuest G JOIN BookingCount BC ON G.Guest_ID = BC.Guest_ID WHERE BC.TotalBookings >= 2;

Output :	
Explanation :	This query uses a Common Table Expression (CTE) named BookingCount to compute the number of bookings per guest. It then filters and displays guests who have booked 2 or more times. CTEs improve readability and are ideal for breaking complex queries into modular steps.

7.4.2 Window Function (ROW_NUMBER)

<table>
<tr><td>Query :</td><td>SELECT

B.Booking_ID,

G.Guest_Name,

ROW_NUMBER() OVER (PARTITION BY G.Guest_ID ORDER BY B.Check_In_Date) AS BookingOrder

FROM

FactBooking B

JOIN DimGuest G ON B.Guest_ID = G.Guest_ID;</td></tr>
<tr><td>Output :</td><td>

```
SELECT
    B.Booking_ID,
    G.Guest_Name,
    ROW_NUMBER() OVER (PARTITION BY G.Guest_ID ORDER BY B.Check_In_Date) AS BookingOrder
FROM
    FactBooking B
JOIN DimGuest G ON B.Guest_ID = G.Guest_ID;
```

FACTBOOKING(+) 1

SELECT B.Booking_ID, G.Guest_Name, ROW_NUM | Enter a SQL expression to filter results (use Ctrl+Space)

	BOOKING_ID	GUEST_NAME	BOOKINGORDER
1	B003	Alice Tan	1
2	B006	Muhamad Adam bin Muhamad Samad	1
3	B019	Muhamad Adam bin Muhamad Samad	2
4	B001	Sarah Kumar	1
5	B016	Sarah Kumar	2
6	B008	Michael Wong	1
7	B005	Nur Aisyah binti Kamal	1
8	B010	Daniel Harriz bin Junaidi	1
9	B002	Farah Zain binti Rahmat	1
10	B017	Farah Zain binti Rahmat	2
11	B007	Kumar Raj	1
12	B009	Lim Mei	1
13	B020	Lim Mei	2
14	B004	Ariff Ismail bin Azfar Imran	1
15	B018	Ariff Ismail bin Azfar Imran	2
16	B011	Sophia Chen	1
17	B012	Ahmad Faisal bin Ismail	1
18	B013	Priya Devi	1
19	B014	Jason Lim	1
20	B015	Nurul Huda binti Mazlan	1

</td></tr>
<tr><td>Explanation :</td><td>This query uses ROW_NUMBER(), a window function, to assign a sequential order to each guest’s bookings, based on the check-in date. The PARTITION BY clause ensures the numbering resets for each guest, making it useful for analyzing booking history and patterns over time.</td></tr>
</table>