CIT6234 - Advanced Database
Assignment 2 (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 Idrus

TABLE OF CONTENTS

1.0 Data Modeling

document

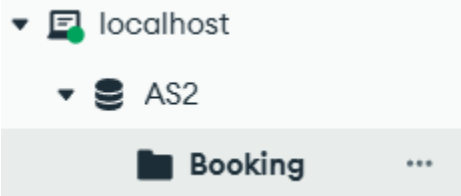
```
booking:
{
  "booking_id": "B001",
  "booking_date": ISODate("2023-02-25T00:00:00Z"),
  "guest_id": "G003",
  "property_id": "P005",
  "host_id": "H002",

  "payment_description": "Cash",

  "total_amount": 900.00,
  "num_of_nights": 3,

  "time":
    "check_in_date": ISODate("2023-03-10T00:00:00Z"),
    "check_out_date": ISODate("2023-03-13T00:00:00Z"),

"host_name": "Rahim Aziz,
  "host_phone_num": "01149876543"

guest: {
  "guest_id": "G003",
 "guest_name": "Sarah Kumar,
  "guest_email": "sarahkumar@gmail.com,
  "guest_phone_num": "01172223333",
  "guest_dob": ISODate("1992-09-11T00:00:00Z"),
  "num_of_guest": 3,
}


  property {
"property_name": "Hilltop Bungalow",
  "property_type": "Bungalow",
  "num_of_rooms": 5,
  "location": "Jalan Bukit Damansara, 50490 Kuala
Lumpur",
  "price": 950.00,

  "facilities":
    "amenities":  "Swimming_Pool, Wifi,
Washing_Machine, Cooking_Utensils"}
}
```

2.0 Data Dictionary - Dania

Collections Name	Object Name	Data Type	Unique
Booking	Booking_ID	String	True
	Booking_Date	Date	
	Guest	Object	True
	Property	Object	True
	Host_ID	String	True
	Payment_Description	String	
	Total_Amount	Double	
	Num_of_Nights	Integer	
	Check_In_Date	Date	
	Check_Out_Date	Date	
	Host_Name	String	
	Host_Phone_Num	String	True
property	Property_ID	String	True
	Property_Name	String	
	Property_Type	String	
	Num_of_Rooms	Integer	
	Location	String	
	Price	Double	
	Amenities	Array	
guest	Guest_ID	String	True
	Guest_Name	String	

	Guest_Email	String	True
	Guest_Phone_Num	String	True
	Guest_DOB	Date	
	Num_of_Guest	Integer	

3.0 Database and Respective Collections - Maryam



4.0 Sample Data - Maryam

4.1 Booking Collection

Booking

	_id ObjectId	Booking_ID String	Booking_Date Date	Guest Object	Property Object	Host_ID String
1	ObjectId('684c388aacedd36…	"B001"	2023-02-25T00:00:00.000+0…	{} 6 fields	{} 7 fields	"H002"
2	ObjectId('684c388aacedd36…	"B002"	2023-04-28T00:00:00.000+0…	{} 6 fields	{} 7 fields	"H004"
3	ObjectId('684c388aacedd36…	"B003"	2022-12-15T00:00:00.000+0…	{} 6 fields	{} 7 fields	"H001"
4	ObjectId('684c388aacedd36…	"B004"	2023-05-20T00:00:00.000+0…	{} 6 fields	{} 7 fields	"H003"
5	ObjectId('684c388aacedd36…	"B005"	2023-03-20T00:00:00.000+0…	{} 6 fields	{} 7 fields	"H005"
6	ObjectId('684c388aacedd36…	"B006"	2023-06-01T00:00:00.000+0…	{} 6 fields	{} 7 fields	"H002"
7	ObjectId('684c388aacedd36…	"B007"	2023-01-30T00:00:00.000+0…	{} 6 fields	{} 7 fields	"H004"
8	ObjectId('684c388aacedd36…	"B008"	2023-07-15T00:00:00.000+0…	{} 6 fields	{} 7 fields	"H001"
9	ObjectId('684c388aacedd36…	"B009"	2023-08-22T00:00:00.000+0…	{} 6 fields	{} 7 fields	"H003"
10	ObjectId('684c388aacedd36…	"B010"	2023-09-10T00:00:00.000+0…	{} 6 fields	{} 7 fields	"H005"

Booking

	Payment_Description String	Total_Amout Double	Num_of_Nights Int32	Check_In_Date Date	Check_Out_Date Date	Host_Name String
1	"Cash"	900	3	2023-03-10T00:00:00.000+0…	2023-03-13T00:00:00.000+0…	"Rahim Aziz"
2	"Touch n Go"	560	2	2023-05-15T00:00:00.000+0…	2023-05-17T00:00:00.000+0…	"Faridah Musa"
3	"Credit Card"	4600	5	2023-01-22T00:00:00.000+0…	2023-01-27T00:00:00.000+0…	"Linda Lee"
4	"Online Banking"	3480	4	2023-06-08T00:00:00.000+0…	2023-06-12T00:00:00.000+0…	"Kenny Ong"
5	"Debit Card"	310	1	2023-04-03T00:00:00.000+0…	2023-04-04T00:00:00.000+0…	"ViFalsed Singh"
6	"E-Wallet"	7700	7	2023-07-14T00:00:00.000+0…	2023-07-21T00:00:00.000+0…	"Rahim Aziz"
7	"Touch n Go"	2100	2	2023-02-18T00:00:00.000+0…	2023-02-20T00:00:00.000+0…	"Faridah Musa"
8	"Cash"	2850	3	2023-08-05T00:00:00.000+0…	2023-08-08T00:00:00.000+0…	"Linda Lee"
9	"Credit Card"	1200	4	2023-09-12T00:00:00.000+0…	2023-09-16T00:00:00.000+0…	"Kenny Ong"
10	"Online Banking"	1400	5	2023-10-01T00:00:00.000+0…	2023-10-06T00:00:00.000+0…	"ViFalsed Singh"

Host_Phone_Num String
"01149876543"
"01167788990"
"01131234567"
"01123344556"
"01191122334"
"01149876543"
"01167788990"
"01131234567"
"01123344556"
"01191122334"

4.2 Guest Object

Booking | Guest { }

	_id ObjectId	Guest_ID String	Guest_Name String	Guest_DOB Date	Guest_Email String	Guest_Phone_Num Strin
1	ObjectId('684c388aacedd3…	"G003"	"Sarah Kumar"	1992-09-11T00:00:00.000+0…	"sarahkumar@gmail.com"	"01172223333"
2	ObjectId('684c388aacedd3…	"G007"	"Farah Zain binti Rahmat"	1985-07-14T00:00:00.000+0…	"farah@gmail.com"	"01183344556"
3	ObjectId('684c388aacedd3…	"G001"	"Alice Tan"	1995-04-12T00:00:00.000+0…	"alicetan@gmail.com"	"01123456789"
4	ObjectId('684c388aacedd3…	"G010"	"Ariff Ismail bin Azfar I…	1990-10-17T00:00:00.000+0…	"ariff@yahoo.com"	"01191112223"
5	ObjectId('684c388aacedd3…	"G005"	"Nur Aisyah binti Kamal"	1997-03-09T00:00:00.000+0…	"aisyahkamal@gmail.com"	"01169988776"
6	ObjectId('684c388aacedd3…	"G002"	"Muhamad Adam bin Muhamad…	1988-06-23T00:00:00.000+0…	"adamsamad@gmail.com"	"01198765432"
7	ObjectId('684c388aacedd3…	"G008"	"Kumar Raj"	1978-11-30T00:00:00.000+0…	"kumar@gmail.com"	"01132233445"
8	ObjectId('684c388aacedd3…	"G004"	"Michael Wong"	1980-12-05T00:00:00.000+0…	"michaelwong@yahoo.com"	"01111234567"
9	ObjectId('684c388aacedd3…	"G009"	"Lim Mei"	2000-05-20T00:00:00.000+0…	"mei@yahoo.com"	"01123451111"
10	ObjectId('684c388aacedd3…	"G006"	"Daniel Harriz bin Junaid…	1993-01-22T00:00:00.000+0…	"danielharriz@yahoo.com"	"01141122334"

Guest_Phone_Num String	Num_of_Guest Int32	Register_Date Date
"01172223333"	3	2022-12-25T00:00:00.000+0…
"01183344556"	8	2023-01-08T00:00:00.000+0…
"01123456789"	2	2022-10-10T00:00:00.000+0…
"01191112223"	3	2023-04-06T00:00:00.000+0…
"01132233445"	4	2022-11-03T00:00:00.000+0…
"01198765432"	10	2023-02-23T00:00:00.000+0…
"01132233445"	4	2022-11-03T00:00:00.000+0…
"01111234567"	4	2023-07-01T00:00:00.000+0…
"01123451111"	2	2023-08-22T00:00:00.000+0…
"01141122334"	5	2023-03-29T00:00:00.000+0…

4.3 Property Object

Booking | Property { }

	_id ObjectId	Property_ID String	Property_Name String	Property_Type String	Num_of_Rooms Int32	Location String
1	ObjectId('684c388aacedd3…	"P005"	"Skyline Residence"	"Condominium"	3	"Jalan Bukit Beruang,
2	ObjectId('684c388aacedd3…	"P002"	"Seaview Condo"	"Condominium"	2	"Jalan Tanjung Bungah
3	ObjectId('684c388aacedd3…	"P007"	"Serene Bungalow"	"Bungalow"	4	"Jalan Ampang, 50450
4	ObjectId('684c388aacedd3…	"P004"	"Garden Bungalow"	"Bungalow"	3	"Jalan Danga Bay, 802(
5	ObjectId('684c388aacedd3…	"P008"	"Urban Heights Condo"	"Condominium"	3	"Persiaran Perdana, 6
6	ObjectId('684c388aacedd3…	"P003"	"Sunset Villa"	"Villa"	3	"Presint 8, 62250 Put
7	ObjectId('684c388aacedd3…	"P006"	"Ocean Breeze Villa"	"Villa"	5	"Pantai Cenang, 07000
8	ObjectId('684c388aacedd3…	"P001"	"Hilltop Bungalow"	"Bungalow"	5	"Jalan Bukit Damansara
9	ObjectId('684c388aacedd3…	"P005"	"Skyline Residence"	"Condominium"	3	"Jalan Bukit Beruang,
10	ObjectId('684c388aacedd3…	"P002"	"Seaview Condo"	"Condominium"	2	"Jalan Tanjung Bungah

Price Int32	Amenities Array
300	[] 3 elements
280	[] 4 elements
920	[] 1 elements
870	[] 2 elements
310	[] 2 elements
1100	[] 3 elements
1050	[] 3 elements
950	[] 2 elements
300	[] 3 elements
280	[] 4 elements

4.4 Amenities Array

Booking | Property { } | Amenities [ ]

	_id ObjectId	0 String	1 String	2 String	3 String
1	ObjectId('684c388aacedd3…	"Swimming_Pool"	"Wifi"	"Cooking_Utensils"	No field
2	ObjectId('684c388aacedd3…	"Swimming_Pool"	"Wifi"	"Washing_Machine"	"Cooking_Utensils"
3	ObjectId('684c388aacedd3…	"Wifi"	No field		
4	ObjectId('684c388aacedd3…	"Wifi"	"Cooking_Utensils"	No field	
5	ObjectId('684c388aacedd3…	"Swimming_Pool"	"Cooking_Utensils"	No field	
6	ObjectId('684c388aacedd3…	"Wifi"	"Washing_Machine"	"Cooking_Utensils"	No field
7	ObjectId('684c388aacedd3…	"Swimming_Pool"	"Wifi"	"Cooking_Utensils"	No field
8	ObjectId('684c388aacedd3…	"Wifi"	"Washing_Machine"	No field	
9	ObjectId('684c388aacedd3…	"Swimming_Pool"	"Wifi"	"Cooking_Utensils"	No field
10	ObjectId('684c388aacedd3…	"Swimming_Pool"	"Wifi"	"Washing_Machine"	"Cooking_Utensils"

5.0 NoSQL Query

5.1 One Query with Logical Operation

Purpose:	This query retrieves bookings made on or after January 1, 2023 that meet specific conditions. It selects bookings where the total amount exceeds 500 or the stay duration is between 3 and 7 nights. It also filters for properties that have either a swimming pool or a washing machine, but not both, and excludes booking paid with Cash or Touch n Go. the purpose is to identify qualified, higher-value bookings with unique property features and valid payment types for analysis or reporting.

Query:	db.Booking.find({
	  $and: [
	  { "Booking_Date": { $gte: ISODate("2023-01-01T00:00:00Z") } },
	  {
	  $or: [
	  { "Total_Amount": { $gt: 500 } },
	  {
	  $and: [
	  { "Num_of_Nights": { $gte: 3 } },
	  { "Num_of_Nights": { $lte: 7 } }
	  ]
	  }
	  ]
	  },
	  {
	  $or: [
	  {
	  $and: [
	  { "Property.Amenities": "Swimming_Pool" },
	  { "Property.Amenities": { $not: { $in: ["Washing_Machine"] } } }
	  ]
	  },
	  {
	  $and: [
	  { "Property.Amenities": "Washing_Machine" },
	  { "Property.Amenities": { $not: { $in: ["Swimming_Pool"] } } }
	  ]
	  }
	  ]
	  },
	  { "Payment_Description": { $nin: ["Cash", "Touch n Go"] } }
	  ]
	});

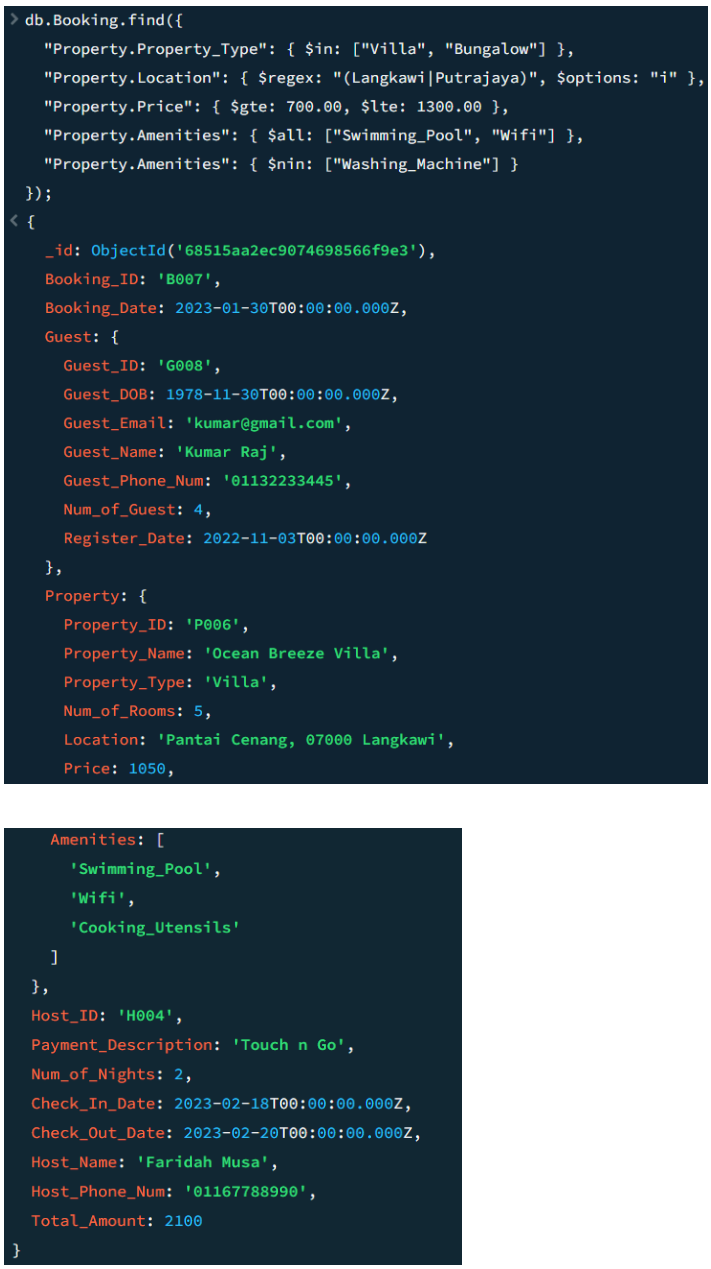
Output:


```
{
  _id: ObjectId('68515aa2ec9074698566f9e2'),
  Booking_ID: 'B006',
  Booking_Date: 2023-06-01T00:00:00.000Z,
  Guest: {
    Guest_ID: 'G002',
    Guest_DOB: 1988-06-23T00:00:00.000Z,
    Guest_Email: 'adamsamad@gmail.com',
    Guest_Name: 'Muhamad Adam bin Muhamad Sa
    Guest_Phone_Num: '01198765432',
    Num_of_Guest: 10,
    Register_Date: 2023-02-23T00:00:00.000Z
  },
  Property: {
    Property_ID: 'P003',
    Property_Name: 'Sunset Villa',
    Property_Type: 'Villa',
    Num_of_Rooms: 3,
    Location: 'Presint 8, 62250 Putrajaya',
    Price: 1100,
    Amenities: [
      'Wifi',
      'Washing_Machine',
      'Cooking_Utensils'
    ]
  },
  Host_ID: 'H002',
  Payment_Description: 'E-Wallet',
  Num_of_Nights: 7,
  Check_In_Date: 2023-07-14T00:00:00.000Z,
  Check_Out_Date: 2023-07-21T00:00:00.000Z,
  Host_Name: 'Rahim Aziz',
  Host_Phone_Num: '01149876543',
  Total_Amount: 7700
}
{
  _id: ObjectId('68515aa2ec9074698566f9e5'),
  Booking_ID: 'B009',
  Booking_Date: 2023-08-22T00:00:00.000Z,
  Guest: {
    Guest_ID: 'G009',
    Guest_DOB: 2000-05-20T00:00:00.000Z,
    Guest_Email: 'mei@yahoo.com',
    Guest_Name: 'Lim Mei',
    Guest_Phone_Num: '01123451111',
    Num_of_Guest: 2,
    Register_Date: 2023-08-22T00:00:00.000Z
  },
```

5.2 One query with Comparison operator - Maryam

Purpose	To find booking records for high-value holiday properties (specifically 'Villa' or 'Bungalow' types) located in popular resort or administrative areas like 'Langkawi' or 'Putrajaya', within a premium budget range of $700.00 to $1300.00, that consistently offer essential luxury amenities like a 'Swimming_Pool' and 'Wifi', but notably do not include a 'Washing_Machine', perhaps catering to guests preferring laundry services or shorter stays. This scenario is common for travel agencies or property management systems seeking to identify specific premium listings for targeted marketing or guest recommendations.
Query	db.Booking.find({ "Property.Property_Type": { $in: ["Villa", "Bungalow"] }, "Property.Location": { $regex: "(Langkawi|Putrajaya)", $options: "i" }, "Property.Price": { $gte: 700.00, $lte: 1300.00 }, "Property.Amenities": { $all: ["Swimming_Pool", "Wifi"] }, "Property.Amenities": { $nin: ["Washing_Machine"] } });

Output	
Explanation	db.Booking.find(): This is the primary command to query documents within the Booking collection. The assumption here is that booking records embed property details within a nested Property sub-document. "Property.Property_Type": { $in: ["Villa", "Bungalow"] }: This filter targets properties whose Property_Type (nested under the Property document) is either "Villa" or "Bungalow". The $in operator efficiently checks for multiple possible values for a single field. "Property.Location": { $regex: "(Langkawi|Putrajaya)", $options: "i" }: This condition filters for properties located in either "Langkawi" or "Putrajaya".

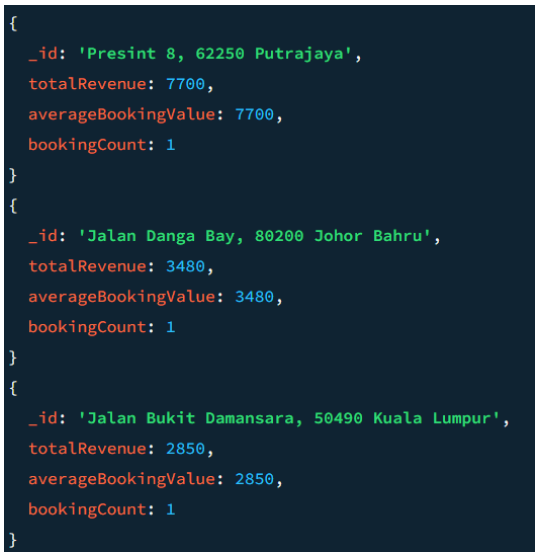
	- $regex: "(Langkawi|Putrajaya)": Uses a regular expression to match documents where the Location field contains either "Langkawi" or "Putrajaya". The | acts as an "OR" logical operator within the regex. - $options: "i": Makes the regular expression match case-insensitively, allowing for variations like "langkawi", "Langkawi", etc. "Property.Price": { $gte: 700.00, $lte: 1300.00 }: This specifies a price range for the properties. - $gte: 700.00: Filters for properties with a Price greater than or equal to $700.00. - $lte: 1300.00: Filters for properties with a Price less than or equal to $1300.00. Both conditions must be met, effectively defining an inclusive range. "Property.Amenities": { $all: ["Swimming_Pool", "Wifi"] }: This is a crucial filter for amenities. It uses the $all operator, which ensures that the Amenities array (nested under Property) must contain both "Swimming_Pool" AND "Wifi" for a document to be matched. This implies that the Amenities field in your MongoDB documents is stored as an array of strings (e.g., ["Swimming_Pool", "Wifi", "Cooking_Utensils"]). "Property.Amenities": { $nin: ["Washing_Machine"] }: This condition further refines the amenity filter. It uses the $nin (not in) operator to exclude any properties where the Amenities array contains "Washing_Machine". This ensures that the results only include properties that do not offer a washing machine.

5.3 Aggregate function

Purpose:	This aggregation query identifies the top 3 property locations based on total booking revenue from May 1, 2023 onwards. It flyers bookings by date, then groups them by location to calculate revenue, average booking value, and booking count. The results are sorted in descending order of revenue to find the highest-earning locations. This helps in understanding which locations are the most profitable for strategic planning and marketing focus.

Query:	db.Booking.aggregate([
	{
	$match: {
	"Booking_Date": { $gte: ISODate("2023-05-01T00:00:00Z") }
	}
	},
	{
	$group: {
	_id: "$Property.Location",
	totalRevenue: { $sum: "$Total_Amount" },

	averageBookingValue: { $avg: "$Total_Amount" },
	bookingCount: { $sum: 1 }
	}
	},
	{
	$sort: {
	totalRevenue: -1
	}
	},
	{
	$limit: 3
	}
	])

Output:	

5.4 One query with project command

Purpose	This query is used to show selected details from the Booking collection. It displays the guest's name in uppercase, their email and phone number, the property price, and number of guests. It also calculates the cost per person by dividing the property price by the number of guests. Lastly, it classifies each booking into a Luxury Level of "High", "Medium", or "Basic" based on the property price.

Query	db.Booking.aggregate([
	{
	$project: {
	_id: 0,
	Guest_Name_Upper: { $toUpper: "$Guest.Guest_Name" },
	Guest_Email: "$Guest.Guest_Email",

```
    Guest_Phone: "$Guest.Guest_Phone_Num",
    Property_Price: "$Property.Price",
    Num_of_Guest: "$Guest.Num_of_Guest",

    Price_Per_Person: {
      $cond: {
        if: { $gt: ["$Guest.Num_of_Guest", 0] },
        then: { $round: [{ $divide: ["$Property.Price", "$Guest.Num_of_Guest"]
}, 2] },
        else: "N/A"
      }
    },

    Luxury_Level: {
      $switch: {
        branches: [
          {
            case: { $gt: ["$Property.Price", 800] },
            then: "High"
          },
          {
            case: {
              $and: [
                { $gte: ["$Property.Price", 400] },
                { $lte: ["$Property.Price", 800] }
              ]
            },
            then: "Medium"
          }
        ],
        default: "Basic"
      }
    }
  }
 }
])
```

Output	{
	Guest_Name_Upper: 'SARAH KUMAR',
	Guest_Email: 'sarahkumar@gmail.com',
	Guest_Phone: '01172223333',
	Property_Price: 300,
	Num_of_Guest: 3,
	Price_Per_Person: 100,
	Luxury_Level: 'Basic'
	}
	{
	Guest_Name_Upper: 'FARAH ZAIN BINTI RAHMAT',
	Guest_Email: 'farah@gmail.com',
	Guest_Phone: '01183344556',
	Property_Price: 280,
	Num_of_Guest: 8,
	Price_Per_Person: 35,
	Luxury_Level: 'Basic'
	}
	{
	Guest_Name_Upper: 'ALICE TAN',
	Guest_Email: 'alicetan@gmail.com',
	Guest_Phone: '01123456789',
	Property_Price: 920,
	Num_of_Guest: 2,
	Price_Per_Person: 460,
	Luxury_Level: 'High'
	}
	{
	{
	Guest_Name_Upper: 'ARIFF ISMAIL BIN AZFAR IMRAN',
	Guest_Email: 'ariff@yahoo.com',
	Guest_Phone: '01191112223',
	Property_Price: 870,
	Num_of_Guest: 3,
	Price_Per_Person: 290,
	Luxury_Level: 'High'
	}
	{
	Guest_Name_Upper: 'NUR AISYAH BINTI KAMAL',
	Guest_Email: 'aisyahkamal@gmail.com',
	Guest_Phone: '01169988776',
	Property_Price: 310,
	Num_of_Guest: 7,
	Price_Per_Person: 44.29,
	Luxury_Level: 'Basic'
	}
	{
	Guest_Name_Upper: 'MUHAMAD ADAM BIN MUHAMAD SAMAD',
	Guest_Email: 'adamsamad@gmail.com',
	Guest_Phone: '01198765432',
	Property_Price: 1100,
	Num_of_Guest: 10,
	Price_Per_Person: 110,
	Luxury_Level: 'High'
	}

	 
Explanation	• $project: Selects and reshapes fields to be shown in the output. • $toUpper: Converts the guest's name to uppercase for consistent formatting. • $cond: Adds logic to check if the number of guests is not zero before dividing. • $divide: Calculates price per person (property price ÷ number of guests). • $round: Rounds the price per person to two decimal places.

	• $switch: Handles multiple conditions to assign a luxury level. • $gt, $gte, $lte: Used to compare property price and define price tiers. + • $and: Ensures both minimum and maximum price conditions are met for Medium level.

5.5 One query with size clause - Dania

Purpose:	This query uses the $size operator to filter documents where an array field contains an exact number of elements. In this context, it is used to identify records such as properties with a specific number of amenities by checking the length of an array field. This helps in narrowing down data based on specific structural criteria, ensuring that only documents with a precise array size are considered. It is particularly useful in scenarios where the quantity of items in a list matters more than their content.

Query:	db.Booking.find(
	{
	$and: [
	{
	$or: [
	{ "Property.Amenities": { $size: 3 } },
	{ "Property.Amenities": { $size: 4 } }
	]
	},
	{
	"Property.Amenities": { $all: ["Wifi", "Cooking_Utensils"] }
	},
	{
	$or: [
	{
	$and: [
	{ "Property.Property_Type": "Condominium" },
	{ "Property.Num_of_Rooms": 3 },
	{ "Property.Price": { $lte: 350 } }
	]
	},
	{
	$and: [
	{ "Property.Property_Type": { $in: ["Bungalow", "Villa"] } },
	{ "Property.Num_of_Rooms": { $gte: 4 } },
	{ "Property.Price": { $gte: 900 } }
	]
	}
	]

	}
	]
	},
	{
	_id: 0, // exclude MongoDB document ID
	"Property.Property_Type": 1,
	"Property.Num_of_Rooms": 1,
	"Property.Price": 1,
	"Property.Amenities": 1
	}
	)

Output:	 
Explanation	This query retrieves bookings with specific property criteria from the Booking collection. It filters documents based on the following conditions:

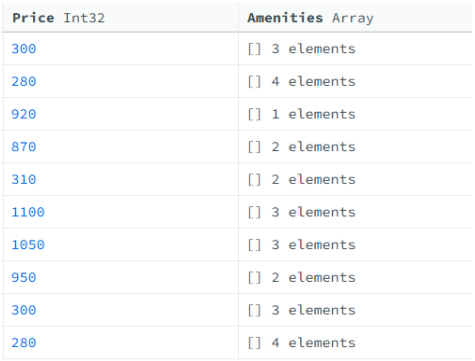
	● Amenities Size – The property must have either 3 or 4 amenities. ● Required Amenities – The amenities list must include both "Wifi" and "Cooking_Utensils". ● Property Type and Price Filter – The property must either: ○ Be a "Condominium" with exactly 3 rooms and a price less than or equal to 350, or ○ Be a "Bungalow" or "Villa" with 4 or more rooms and a price greater than or equal to 900. The output excludes the _id field and only displays the property's type, number of rooms, price, and amenities.

6.0 NoSQL command to update particular record based on certain criteria - Maryam

Purpose	To apply a broad promotional price adjustment across your booking records by offering a 5% discount and marking them as 'discount applicable'. This update targets records where the associated property meets at least one of the following conditions: it is a 'Villa', it is located in 'Johor Bahru', or it has more than 3 rooms. This scenario is useful for campaigns that target properties based on type, specific regional popularity, or capacity, without requiring all conditions to be met simultaneously.

Query	db.Booking.updateMany(
	{
	$or: [
	{ "Property.Property_Type": "Villa" },
	{ "Property.Location": { $regex: "Johor Bahru", $options: "i" } },
	{ "Property.Num_of_Rooms": { $gt: 3 } }
	]
	},
	{
	$mul: { "Property.Price": 0.95 },
	$set: {
	"Property.Discount_Applicable": true,
	"Property.Last_Price_Adjustment": new Date()
	}
	}
	);

Output	Before:

	 After: 
Explanation	db.Booking.updateMany(): This command is used to modify multiple documents within the Booking collection that match the specified filter criteria. Filter Criteria ({ $or: [...] }): The core of this command's targeting logic lies within the $or operator. This means that a booking record will be selected for an update if any of the conditions listed within the $or array are true for its nested Property details: - { "Property.Property_Type": "Villa" }: This condition matches any booking record where the Property_Type within the nested Property document is exactly "Villa". - { "Property.Location": { $regex: "Johor Bahru", $options: "i" } }: This condition matches any booking record where the Location field (within the nested Property document) contains "Johor Bahru". The $options: "i" ensures the search is case-insensitive (e.g., "johor bahru", "Johor Bahru" will all match). - { "Property.Num_of_Rooms": { $gt: 3 } }: This condition matches any booking record where the Num_of_Rooms field (within the nested Property document) is greater than 3. **Update Operations:** $mul: { "Property.Price": 0.95 }: This operator is used to modify the Price field within the Property sub-document. It multiplies the current Price by 0.95, effectively applying a 5% discount to the property's price for the matching booking records. $set: { "Property.Discount_Applicable": true, "Property.Last_Price_Adjustment": new Date() }: The $set operator adds new fields or updates existing ones within the Property sub-document.

	- "Property.Discount_Applicable": true: A new boolean field Discount_Applicable is added to the Property sub-document and set to true, indicating that a discount has been applied. - "Property.Last_Price_Adjustment": new Date(): A new field Last_Price_Adjustment is added to the Property sub-document and set to the current date and time using new Date(), providing a timestamp for when this specific price adjustment occurred.

7.0 NoSQL command to delete some specific records based on certain criteria - Maryam

Purpose	To perform a targeted data cleanup by removing booking records that are older than a specific date and are associated with guests who registered within a particular calendar year. This query specifically deletes booking entries made before June 1, 2023, where the corresponding guest registered during the entire year of 2022. This is useful for database maintenance, such as archiving or purging outdated transactional data based on specific historical criteria.

Query	db.Booking.deleteMany({
	"Booking_Date": { $lt: ISODate("2023-06-01T00:00:00Z") },
	"Guest.Register_Date": {
	$gte: ISODate("2022-01-01T00:00:00Z"),
	$lt: ISODate("2023-01-01T00:00:00Z")
	}
	});

Output	Before: Booking _id ObjectId / Booking_ID String / Booking_Date Date / Guest Object / Property Object / Host_ID String 1 ObjectId('684c388aacedd36… / "B001" / 2023-02-25T00:00:00.000+0… / {} 6 fields / {} 7 fields / "H002" 2 ObjectId('684c388aacedd36… / "B002" / 2023-04-28T00:00:00.000+0… / {} 6 fields / {} 7 fields / "H004" 3 ObjectId('684c388aacedd36… / "B003" / 2022-12-15T00:00:00.000+0… / {} 6 fields / {} 7 fields / "H001" 4 ObjectId('684c388aacedd36… / "B004" / 2023-05-20T00:00:00.000+0… / {} 6 fields / {} 7 fields / "H003" 5 ObjectId('684c388aacedd36… / "B005" / 2023-03-20T00:00:00.000+0… / {} 6 fields / {} 7 fields / "H005" 6 ObjectId('684c388aacedd36… / "B006" / 2023-06-01T00:00:00.000+0… / {} 6 fields / {} 7 fields / "H002" 7 ObjectId('684c388aacedd36… / "B007" / 2023-01-30T00:00:00.000+0… / {} 6 fields / {} 7 fields / "H004" 8 ObjectId('684c388aacedd36… / "B008" / 2023-07-15T00:00:00.000+0… / {} 6 fields / {} 7 fields / "H001" 9 ObjectId('684c388aacedd36… / "B009" / 2023-08-22T00:00:00.000+0… / {} 6 fields / {} 7 fields / "H003" 10 ObjectId('684c388aacedd36… / "B010" / 2023-09-10T00:00:00.000+0… / {} 6 fields / {} 7 fields / "H005"

Booking

	Payment_Description String	Total_Amout Double	Num_of_Nights Int32	Check_In_Date Date	Check_Out_Date Date	Host_Name String
1	"Cash"	900	3	2023-03-10T00:00:00.000+0…	2023-03-13T00:00:00.000+0…	"Rahim Aziz"
2	"Touch n Go"	560	2	2023-05-15T00:00:00.000+0…	2023-05-17T00:00:00.000+0…	"Faridah Musa"
3	"Credit Card"	4600	5	2023-01-22T00:00:00.000+0…	2023-01-27T00:00:00.000+0…	"Linda Lee"
4	"Online Banking"	3480	4	2023-06-08T00:00:00.000+0…	2023-06-12T00:00:00.000+0…	"Kenny Ong"
5	"Debit Card"	310	1	2023-04-03T00:00:00.000+0…	2023-04-04T00:00:00.000+0…	"ViFalsed Singh"
6	"E-Wallet"	7700	7	2023-07-14T00:00:00.000+0…	2023-07-21T00:00:00.000+0…	"Rahim Aziz"
7	"Touch n Go"	2100	2	2023-02-18T00:00:00.000+0…	2023-02-20T00:00:00.000+0…	"Faridah Musa"
8	"Cash"	2850	3	2023-08-05T00:00:00.000+0…	2023-08-08T00:00:00.000+0…	"Linda Lee"
9	"Credit Card"	1200	4	2023-09-12T00:00:00.000+0…	2023-09-16T00:00:00.000+0…	"Kenny Ong"
10	"Online Banking"	1400	5	2023-10-01T00:00:00.000+0…	2023-10-06T00:00:00.000+0…	"ViFalsed Singh"

Host_Phone_Num String
"01149876543"
"01167788990"
"01131234567"
"01123344556"
"01191122334"
"01149876543"
"01167788990"
"01131234567"
"01123344556"
"01191122334"

Booking | Guest { }

	_id ObjectId	Guest_ID String	Guest_Name String	Guest_DOB Date	Guest_Email String	Guest_Phone_Num Strin
1	ObjectId('684c388aacedd3…	"G003"	"Sarah Kumar"	1992-09-11T00:00:00.000+0…	"sarahkumar@gmail.com"	"01172223333"
2	ObjectId('684c388aacedd3…	"G007"	"Farah Zain binti Rahmat"	1985-07-14T00:00:00.000+0…	"farah@gmail.com"	"01183344556"
3	ObjectId('684c388aacedd3…	"G001"	"Alice Tan"	1995-04-12T00:00:00.000+0…	"alicetan@gmail.com"	"01123456789"
4	ObjectId('684c388aacedd3…	"G010"	"Ariff Ismail bin Azfar I…	1990-10-17T00:00:00.000+0…	"ariff@yahoo.com"	"01191112223"
5	ObjectId('684c388aacedd3…	"G005"	"Nur Aisyah binti Kamal"	1997-03-09T00:00:00.000+0…	"aisyahkamal@gmail.com"	"01169988776"
6	ObjectId('684c388aacedd3…	"G002"	"Muhamad Adam bin Muhamad…	1988-06-23T00:00:00.000+0…	"adamsamad@gmail.com"	"01198765432"
7	ObjectId('684c388aacedd3…	"G008"	"Kumar Raj"	1978-11-30T00:00:00.000+0…	"kumar@gmail.com"	"01132233445"
8	ObjectId('684c388aacedd3…	"G004"	"Michael Wong"	1980-12-05T00:00:00.000+0…	"michaelwong@yahoo.com"	"01111234567"
9	ObjectId('684c388aacedd3…	"G009"	"Lim Mei"	2000-05-20T00:00:00.000+0…	"mei@yahoo.com"	"01123451111"
10	ObjectId('684c388aacedd3…	"G006"	"Daniel Harriz bin Junaid…	1993-01-22T00:00:00.000+0…	"danielharriz@yahoo.com"	"01141122334"

Guest_Phone_Num String	Num_of_Guest Int32	Register_Date Date
"01172223333"	3	2022-12-25T00:00:00.000+0…
"01183344556"	8	2023-01-08T00:00:00.000+0…
"01123456789"	2	2022-10-10T00:00:00.000+0…
"01191112223"	3	2023-04-06T00:00:00.000+0…
"01132233445"	4	2022-11-03T00:00:00.000+0…
"01198765432"	10	2023-02-23T00:00:00.000+0…
"01132233445"	4	2022-11-03T00:00:00.000+0…
"01111234567"	4	2023-07-01T00:00:00.000+0…
"01123451111"	2	2023-08-22T00:00:00.000+0…
"01141122334"	5	2023-03-29T00:00:00.000+0…

After:

Booking

Booking

	_id ObjectId	Booking_ID String	Booking_Date Date	Guest Object	Property Object	Host_ID String
1	ObjectId('68515aa2ec90746…	"B002"	2023-04-28T00:00:00.000+0…	{} 7 fields	{} 7 fields	"H004"
2	ObjectId('68515aa2ec90746…	"B004"	2023-05-20T00:00:00.000+0…	{} 7 fields	{} 7 fields	"H003"
3	ObjectId('68515aa2ec90746…	"B006"	2023-06-01T00:00:00.000+0…	{} 7 fields	{} 7 fields	"H002"
4	ObjectId('68515aa2ec90746…	"B008"	2023-07-15T00:00:00.000+0…	{} 7 fields	{} 7 fields	"H001"
5	ObjectId('68515aa2ec90746…	"B009"	2023-08-22T00:00:00.000+0…	{} 7 fields	{} 7 fields	"H003"
6	ObjectId('68515aa2ec90746…	"B010"	2023-09-10T00:00:00.000+0…	{} 7 fields	{} 7 fields	"H005"

Booking

	Payment_Description String	Num_of_Nights Int32	Check_In_Date Date	Check_Out_Date Date	Host_Name String	Host_Phone_Num String
1	"Touch n Go"	2	2023-05-15T00:00:00.000+0…	2023-05-17T00:00:00.000+0…	"Faridah Musa"	"01167788990"
2	"Online Banking"	4	2023-06-08T00:00:00.000+0…	2023-06-12T00:00:00.000+0…	"Kenny Ong"	"01123344556"
3	"E-Wallet"	7	2023-07-14T00:00:00.000+0…	2023-07-21T00:00:00.000+0…	"Rahim Aziz"	"01149876543"
4	"Cash"	3	2023-08-05T00:00:00.000+0…	2023-08-08T00:00:00.000+0…	"Linda Lee"	"01131234567"
5	"Credit Card"	4	2023-09-12T00:00:00.000+0…	2023-09-16T00:00:00.000+0…	"Kenny Ong"	"01123344556"
6	"Online Banking"	5	2023-10-01T00:00:00.000+0…	2023-10-06T00:00:00.000+0…	"ViFalsed Singh"	"01191122334"

Total_Amount Double
560
3480
7700
2850
1200
1400

Guest

Booking | Guest { }

	_id ObjectId	Guest_ID String	Guest_DOB Date	Guest_Email String	Guest_Name String	Guest_Phone_Num Stri
1	ObjectId('68515aa2ec9074…	"G007"	1985-07-14T00:00:00.000+0…	"farah@gmail.com"	"Farah Zain binti Rahmat"	"01183344556"
2	ObjectId('68515aa2ec9074…	"G010"	1990-10-17T00:00:00.000+0…	"ariff@yahoo.com"	"Ariff Ismail bin Azfar I…	"01191112223"
3	ObjectId('68515aa2ec9074…	"G002"	1988-06-23T00:00:00.000+0…	"adamsamad@gmail.com"	"Muhamad Adam bin Muhamad…	"01198765432"
4	ObjectId('68515aa2ec9074…	"G004"	1980-12-05T00:00:00.000+0…	"michaelwong@yahoo.com"	"Michael Wong"	"01111234567"
5	ObjectId('68515aa2ec9074…	"G009"	2000-05-20T00:00:00.000+0…	"mei@yahoo.com"	"Lim Mei"	"01123451111"
6	ObjectId('68515aa2ec9074…	"G006"	1993-01-22T00:00:00.000+0…	"danielharriz@yahoo.com"	"Daniel Harriz bin Junaid…	"01141122334"

Num_of_Guest Int32	Register_Date Date
8	2023-01-08T00:00:00.000+0…
3	2023-04-06T00:00:00.000+0…
10	2023-02-23T00:00:00.000+0…
4	2023-07-01T00:00:00.000+0…
2	2023-08-22T00:00:00.000+0…
5	2023-03-29T00:00:00.000+0…

Property

Booking | Property { }

	_id ObjectId	Property_ID String	Property_Name String	Property_Type String	Num_of_Rooms Int32	Location String
1	ObjectId('68515aa2ec9074…	"P002"	"Seaview Condo"	"Condominium"	2	"Jalan Tanjung Bungah
2	ObjectId('68515aa2ec9074…	"P004"	"Garden Bungalow"	"Bungalow"	3	"Jalan Danga Bay, 802(
3	ObjectId('68515aa2ec9074…	"P003"	"Sunset Villa"	"Villa"	3	"Presint 8, 62250 Put
4	ObjectId('68515aa2ec9074…	"P001"	"Hilltop Bungalow"	"Bungalow"	5	"Jalan Bukit Damansar
5	ObjectId('68515aa2ec9074…	"P005"	"Skyline Residence"	"Condominium"	3	"Jalan Bukit Beruang,
6	ObjectId('68515aa2ec9074…	"P002"	"Seaview Condo"	"Condominium"	2	"Jalan Tanjung Bungah

Booking | Property { }

	_id ObjectId	rty_Type String	Num_of_Rooms Int32	Location String	Price Int32	Amenities Array
1	ObjectId('68515aa2ec9074…	lominium"	2	"Jalan Tanjung Bungah, 11…	280	[] 4 elements
2	ObjectId('68515aa2ec9074…	galow"	3	"Jalan Danga Bay, 80200 J…	870	[] 2 elements
3	ObjectId('68515aa2ec9074…	a"	3	"Presint 8, 62250 Putraja…	1100	[] 3 elements
4	ObjectId('68515aa2ec9074…	galow"	5	"Jalan Bukit Damansara, 5…	950	[] 2 elements
5	ObjectId('68515aa2ec9074…	lominium"	3	"Jalan Bukit Beruang, 754…	300	[] 3 elements
6	ObjectId('68515aa2ec9074…	lominium"	2	"Jalan Tanjung Bungah, 11…	280	[] 4 elements

Explanation	db.Booking.deleteMany(): This command is used to remove multiple documents from your Booking collection. Filter Criteria: Documents will only be deleted if they match both of the following conditions: - "Booking_Date": { $lt: ISODate("2023-06-01T00:00:00Z") }: This condition filters for booking records where the Booking_Date field is earlier than June 1, 2023. - "Guest.Register_Date": { $gte: ISODate("2022-01-01T00:00:00Z"), $lt: ISODate("2023-01-01T00:00:00Z") }: This condition targets booking records

	whose associated guest registered on or after January 1, 2022, and before January 1, 2023. In essence, it selects bookings made by guests who registered at any point during the calendar year 2022.

8.0 TWO NoSQL commands that are not covered in lecture - Iman

8.1 Identify guests who paid more per person compared to average price per person

Purpose	This query identifies guests who paid more for their booking by comparing the price per person of each booking against the average price per person across all bookings. The purpose is to find bookings where the cost per guest is above average, which may suggest poor value or overpriced properties. This is useful for pricing analysis, customer experience improvement, and identifying listings that may need adjustment.

Query	db.Booking.aggregate([
	
	{
	$project: {
	Booking_ID: 1,
	Guest_Name: "$Guest.Guest_Name",
	Guest_Email: "$Guest.Guest_Email",
	Guest_Phone: "$Guest.Guest_Phone_Num",
	Property_Name: "$Property.Property_Name",
	Total_Price: "$Property.Price",
	Num_of_Guest: "$Guest.Num_of_Guest",
	Price_Per_Person: {
	$cond: {
	if: { $gt: ["$Guest.Num_of_Guest", 0] },
	then: { $round: [{ $divide: ["$Property.Price", "$Guest.Num_of_Guest"] }, 2] },
	else: 0
	}
	}
	}
	},

```
{
 $facet: {
  bookings: [
   {
    $project: {
     Booking_ID: 1,
     Guest_Name: 1,
     Guest_Email: 1,
     Guest_Phone: 1,
     Property_Name: 1,
     Total_Price: 1,
     Num_of_Guest: 1,
     Price_Per_Person: 1
    }
   }
  ],
  average: [
   {
    $group: {
     _id: null,
     avgPricePerPerson: { $avg: "$Price_Per_Person" }
    }
   }
  ]
 }
},

{
 $project: {
  bookings: 1,
  avgPrice: { $arrayElemAt: ["$average.avgPricePerPerson", 0] }
 }
},

{
 $unwind: "$bookings"
},
{
 $replaceRoot: {
  newRoot: {
   $mergeObjects: ["$bookings", { avgPrice: "$avgPrice" }]
  }
 }
},
{
 $match: {
  $expr: { $gt: ["$Price_Per_Person", "$avgPrice"] }
 }
}
```

	])
Output	 
Explanation	• $project – Selects relevant fields and computes Price_Per_Person using division and rounding.

	• $cond – Ensures safe division (prevents division by 0). • $facet – Runs parallel pipelines to calculate both the per-booking details and the overall average. • $group – Calculates average price per person across all documents. • $arrayElemAt – Extracts average from an array. • $unwind / $mergeObjects / $replaceRoot – Flattens and merges data from parallel stages. • $expr with $gt – Filters documents where a computed value exceeds the average.

8.2 Host performance

Purpose	This query analyzes host performance by aggregating booking data based on each host. It calculates the total revenue earned, the number of bookings, the total number of guests served, and the average revenue per booking for every host. Each host is also categorized as either "Senior" or "Junior" based on their total revenue, where hosts who earn more than RM3000 are marked as "Senior". This command is useful for evaluating host contributions, identifying top performers, and managing host relationships effectively.

Query	db.Booking.aggregate([
	{
	$group: {
	_id: {
	Host_ID: "$Host_ID",
	Host_Name: "$Host_Name",
	Host_Phone_Num: "$Host_Phone_Num"
	},
	Total_Revenue: { $sum: "$Total_Amount" },
	Booking_Count: { $sum: 1 },
	Total_Guests: { $sum: "$Guest.Num_of_Guest" }
	}
	},
	{
	$addFields: {
	Average_Revenue_Per_Booking: {
	$round: [{ $divide: ["$Total_Revenue", "$Booking_Count"] }, 2]
	},
	Host_Level: {
	$cond: {

	if: { $gt: ["$Total_Revenue", 3000] },
	then: "Senior",
	else: "Junior"
	}
	}
	}
	},
	{
	$sort: { Total_Revenue: -1 }
	}
	])

Output	

<table>
<tr>
<td></td>
<td>
<pre>
{
  _id: {
    Host_ID: 'H003',
    Host_Name: 'Kenny Ong',
    Host_Phone_Num: '01123344556'
  },
  Total_Revenue: 4680,
  Booking_Count: 2,
  Total_Guests: 5,
  Average_Revenue_Per_Booking: 2340,
  Host_Level: 'Senior'
}
{
  _id: {
    Host_ID: 'H004',
    Host_Name: 'Faridah Musa',
    Host_Phone_Num: '01167788990'
  },
  Total_Revenue: 2660,
  Booking_Count: 2,
  Total_Guests: 12,
  Average_Revenue_Per_Booking: 1330,
  Host_Level: 'Junior'
}
</pre>
<pre>
{
  _id: {
    Host_ID: 'H005',
    Host_Name: 'ViFalsed Singh',
    Host_Phone_Num: '01191122334'
  },
  Total_Revenue: 1710,
  Booking_Count: 2,
  Total_Guests: 12,
  Average_Revenue_Per_Booking: 855,
  Host_Level: 'Junior'
}
</pre>
</td>
</tr>
<tr>
<td>Explanation</td>
<td>

$group – Aggregates all bookings by each host using Host_ID, Host_Name, and Host_Phone_Num. It calculates:

Total_Revenue by summing Total_Amount
Booking_Count by counting documents
Total_Guests by summing Guest.Num_of_Guest

$addFields – Adds computed fields after grouping:

Average_Revenue_Per_Booking is calculated using $divide and rounded to 2 decimal places.

</td>
</tr>
</table>

	- Host_Level is classified using $cond based on revenue: - More than 3000 → "Senior" - Otherwise → "Junior" ● $round – Ensures clean display of average revenue values. ● $cond – Provides if-else logic to assign levels. ● $sort – Orders the output by Total_Revenue in descending order to highlight top-earning hosts.

Short Essay Big Data

Big Data is transforming the hospitality industry by helping hotels, resorts, and digital platforms harness vast amounts of information to enhance guest experiences and streamline operations. It empowers businesses to personalize services, implement dynamic pricing, and improve efficiency. This essay explores Big Data's impact on pricing, operational safety, personalized marketing, and customer loyalty.

One of the most significant applications of Big Data is dynamic pricing. Platforms like Airbnb analyze search behavior, booking patterns, and external variables such as local events and seasonal trends to forecast demand. These insights power real-time pricing algorithms, allowing hosts to maximize revenue while offering competitive rates. This kind of forward-looking pricing strategy is only achievable through large-scale data analysis.

Operational efficiency and guest safety have also improved with the help of Big Data. Hospitality businesses can monitor user activity, reviews, and transaction patterns to detect fraud and flag anomalies. Predictive analytics allow providers to identify issues like potential equipment failures or staffing shortages before they impact service delivery. These back-end improvements may not be visible to guests but significantly contribute to seamless and safe hospitality experiences.

Big Data also enables hyper-personalized marketing strategies. Platforms such as Booking.com and Expedia use demographic information, browsing habits, and engagement metrics to customize offers and tailor user experiences. A frequent business traveler, for example, might receive curated city-center hotel recommendations and timely push notifications. This precision not only increases bookings but also builds meaningful connections with customers through relevant and timely communication.

Finally, real-time feedback analysis helps providers respond to guest needs more effectively. Platforms like TripAdvisor go beyond collecting reviews—they apply sentiment analysis and emotion recognition to identify emerging satisfaction trends. This allows hotels to retrain staff, upgrade services, or amplify positive experiences based on live feedback, creating a continuous improvement loop fueled by data.

In conclusion, Big Data is more than a technological tool; it is a strategic driver in the hospitality industry. It transforms operations from reactive to predictive, allowing businesses to anticipate needs, solve problems before they arise, and deliver highly personalized experiences. As the industry continues to evolve, Big Data will remain central to making travel safer, smarter, and more memorable.

References

Barten, M. (2024, June 4). *5 Ways Big Data Can Help Those in the Hospitality Industry*. Revfine.com. Retrieved June 20, 2025, from https://www.revfine.com/big-data-hospitality-industry/

Tripadvisor for Business | Reach Millions of Travelers. (n.d.). Tripadvisor. Retrieved June 20, 2025, from https://www.tripadvisor.com/business

Katal, A., Wazid, M., & Goudar, R. H. (2013). *Big data: Issues, challenges, tools and good practices*. In 2013 Sixth International Conference on Contemporary Computing (IC3) (pp. 404–409). IEEE. https://doi.org/10.1109/IC3.2013.6612229

Miah, S. J., Vu, H. Q., Gammack, J., & McGrath, M. (2017). *A big data analytics method for tourist behaviour analysis*. *Information & Management*, 54(6), 771–785. https://doi.org/10.1016/j.im.2016.11.011

Pros and Cons of Big Data. (2024, February 28). Harvard Online. Retrieved June 20, 2025, from https://www.harvardonline.harvard.edu/blog/pros-cons-big-data