Dr. D. Y. Patil Pratishthan's

D. Y. Patil Institute of Master of Computer Applications and Management (Approved by AICTE, New Delhi & Affiliated to Savitribai Phule Pune University)

Dr. D. Y. Patil Educational Complex, Sector 29, Pradhikaran, Akurdi, Pune – 411 044

Tel No: (020)27640998, Website:www.dypimca.ac.in, E-mail : director@dypimca.ac.in

PART C Index

SR.NO	Topic	Date	Sign
1	**Create database restaurant with collections** **Restaurant_details(rest_id,rest_name,rest_address,rest_contactdetails)** **Rest_emp_details(rest_empid,rest_empname,rest_empemail,rest_empaddress,rest_empsalary)** **Invoice_details(invoice_id,invoice_itemdetails,invoice_amountdetails)** **Query to display those employee records whose salary is greater than 7000.** **Perform insert atleast 10 records** **Perform update on rest_empname,rest_contactdetails and rest_empsalary**		
2	**Create database Shopping_center with collections** **Shoppingcenter_details(shopcenter_registrationid,shopcenter_name,shopcenter_branchdetails,shopcenter_address,shop_centeremailid,shopcenter_contactdetails)** **Company(company_id,company_code,company_name,company_address,company_mail)** **emp_details(rest_empid,rest_empname,rest_empemail,rest_empaddress,rest_empsalary)** **Shop(shop_id,shop_code,shop_name,** **shopcenter_registrationid,floor_no,start_date)** **Insert 10 records for each key. Perform** **update query on floor,** **shopcenter_branchdetails,shop_code** **Perform delete operation floor_no** **Perform query to display all employee details whose name are Joe or Steev** **Perform query to display shoppingcenter_branchdetails "laystreet" and Adam road**		

3	**Create Database O'Reilly with collections** **Attendance(Attendee_id,First_name,Last_name,Phone_num, Email,VIP)** **Company(Company_id,Name,Description,Primary_contact_ attendee_id)** **Presentation(Presentation_id,Booked_companyid,Booked_ro omid,Start_time,End_time)** **Room(Room_id,Floor_number,Seat_capacity)**		

	Presentation_attendance(Ticket_id,Presentation_id,Attendee _id) **Insert 10 records for each key in collections.** **Perform query to display Company_name="Book_myspace", "Available_space","Rent_outspace"**		
4	**Create Database Library_mangement with collections** **Students(stud_id,stud_name,stud_surname,dob,gender,class)** **Borrow(borrow_id,stud_id,book_id,taken_date)** **Books(book_id,name,pagecount,point,author_id,type_id)** **Author(author_id,author_name,author_surname)** **Type(type_id,type_name)** **Insert 10 records to each collections key.** **Perform update in book name="Advance C" whose book name="Let Us C" and pagecount="201"** **List all Borrow books whose taken_date=11th -july2022 or taken_date =26th-August-2022**		
5	**Create database Nursary_management with following collections** **Supplier(Sup_id,Sup_name,Mobile,Prod_no,Email,Location_ id,Price)** **Commodities(product_no,product_price,product_name,Qua ntity,Supply)** **Customers(Cust_id,Location_id,Name,Email,Phone,Invoice)** **Category(Category_id,Category_name,Product_no)** **Employee(Employee_id,Employee_name,Email,Mobile_no,L ocation_id,Desgn_name,)** **Insert 8 records for each collection key.** **Display all supplier details whose price is greater than 8000 and less than 30000** **Display all Commodities whose product_price is less than equal to 8000.**		

6	**Create database Inventory Management with following collections:** **Product(prod_id,prod_name,part_number,product_label,inventory_received,inventory_shipped,inventory_onhand,Minimum_requirement)** **Purchases(pur_id,supplier_id,product_id,purchasedate)** **Orders(id,Title,Customer_name,prod_id,order_date)** **Supplier(supplier_id,sup_name)** **Insert 12 records for each collection key** **List all the product details whose minimum_requirement is less than 5 but greater than zero.** **Update Supplier whose sup_name is Thomas**		
7	**Write a NumPy program to convert a list of numeric values into a one-dimensional NumPy array.**		

8	**Write a NumPy program to reverse an array (the first element becomes the last).**		
9	**Write a NumPy program to append values to the end of an array.** **Expected Output:** **Original array:** **[10, 20, 30]** **After append values to the end of the array:** **[10 20 30 40 50 60 70 80 90]**		
10	**Write a NumPy program to create an empty and full array.** **Expected Output:** **[6.93270651e-310 1.59262180e-316 6.93270559e-310 6.93270665e-310]** **[6.93270667e-310 6.93270671e-310 6.93270668e-310 6.93270483e-310]** **[6.93270668e-310 6.93270671e-310 6.93270370e-310 6.93270488e-310]]** **[[6 6 6]** **[6 6 6]** **[6 6 6]]**		
11	**Write a NumPy program to convert Centigrade degrees into Fahrenheit degrees. Centigrade values are stored in a NumPy array.** **Sample Array:** **Values in Fahrenheit degrees [0, 12, 45.21, 34, 99.91]** **Values in Centigrade degrees [-17.78, -11.11, 7.34, 1.11, 37.73, 0.]**		

12	**Write a NumPy program to find common values between two arrays.** **Expected Output:** **Array1: [0 10 20 40 60]** **Array2: [10, 30, 40]** **Common values between two arrays:** **[10 40]**		
13	**Write a NumPy program to add a border (filled with 0's) around an existing array.** **Expected Output:** **Original array:** **[[1. 1. 1.]** **[1. 1. 1.]** **[1. 1. 1.]]** **1 on the border and 0 inside in the array**		
14	**Write a NumPy program to sort a given array of shape 2 along the first axis, last axis and on flattened array.** **Expected Output:**		
	Original array: **[[12 40]** **[70 30]]**		
15	**Write a NumPy program to create an element-wise comparison (greater, greater_equal, less and less_equal) of two given arrays.**		
16	**Write a python program to Check whether a Numpy array contains a specified row**		
17	**Write python program to count of occurrence of each element in array and display maximum occurrence count.**		

1. Create database restaurant with collections
Restaurant_details(rest_id,rest_name,rest_address,rest_cont actdetails)
Rest_emp_details(rest_empid,rest_empname,rest_empemail,
rest_empaddress,rest_empsalary)
Invoice_details(invoice_id,invoice_itemdetails,invoice_amoun tdetails)
Query to display those employee records whose salary is greater than 7000.
Perform insert atleast 10 records
Perform update on rest_empname,rest_contactdetails and rest_empsalary

```
import pymongo

# Connect to MongoDB
client = pymongo.MongoClient("mongodb://localhost:27017")
database = client["restaurant"]

# Get references to the collections
restaurant_details_collection = database["Restaurant_details"]
rest_emp_details_collection = database["Rest_emp_details"]
invoice_details_collection = database["Invoice_details"]

# Function to insert records into the collections
def insert_record(collection, record):
    collection.insert_one(record)

# Function to update employee details
def update_employee_details(collection, emp_id, emp_name, contact_details, salary):
    collection.update_one(
        {"rest_empid": emp_id},
        {"$set": {"rest_empname": emp_name, "rest_contactdetails": contact_details,
"rest_empsalary": salary}}
    )

# Function to query employee details by salary
def query_employee_by_salary(collection, min_salary):
    results = collection.find({"rest_empsalary": {"$gt": min_salary}})
    return list(results)

# Insert records for each collection
for i in range(1, 11):
    restaurant_detail = {
        'rest_id': i,
        'rest_name': f'Restaurant {i}',
        'rest_address': f'Address {i}',
        'rest_contactdetails': f'Contact Details {i}'
    }
    insert_record(restaurant_details_collection, restaurant_detail)

    emp_detail = {
        'rest_empid': i,
```

```
        'rest_empname': f'Employee {i}',
        'rest_empemail': f'employee{i}@example.com',
        'rest_empaddress': f'Employee Address {i}',
        'rest_empsalary': 5000 + i * 1000
    }
    insert_record(rest_emp_details_collection, emp_detail)

    invoice_detail = {
        'invoice_id': i,
        'invoice_itemdetails': f'Item Details {i}',
        'invoice_amountdetails': f'Amount Details {i}'
    }
    insert_record(invoice_details_collection, invoice_detail)

# Update employee details
update_employee_details(rest_emp_details_collection, 1, "Updated Employee Name", "Updated
Contact Details", 8000)

# Query employee details by salary
min_salary = 7000
results = query_employee_by_salary(rest_emp_details_collection, min_salary)
print(f"Employees with salary greater than {min_salary}:")
for employee in results:
    print(employee)
```

OUTPUT:

Restaurant_details

	_id ObjectId	rest_id Int32	rest_name String	rest_address String	rest_contactdetails String
1	ObjectId('64a9b9692e8c60b2d34…	1	"Restaurant 1"	"Address 1"	"Contact Details 1"
2	ObjectId('64a9b9692e8c60b2d34…	2	"Restaurant 2"	"Address 2"	"Contact Details 2"
3	ObjectId('64a9b9692e8c60b2d34…	3	"Restaurant 3"	"Address 3"	"Contact Details 3"
4	ObjectId('64a9b9692e8c60b2d34…	4	"Restaurant 4"	"Address 4"	"Contact Details 4"
5	ObjectId('64a9b9692e8c60b2d34…	5	"Restaurant 5"	"Address 5"	"Contact Details 5"
6	ObjectId('64a9b9692e8c60b2d34…	6	"Restaurant 6"	"Address 6"	"Contact Details 6"
7	ObjectId('64a9b9692e8c60b2d34…	7	"Restaurant 7"	"Address 7"	"Contact Details 7"
8	ObjectId('64a9b9692e8c60b2d34…	8	"Restaurant 8"	"Address 8"	"Contact Details 8"
9	ObjectId('64a9b9692e8c60b2d34…	9	"Restaurant 9"	"Address 9"	"Contact Details 9"
10	ObjectId('64a9b9692e8c60b2d34…	10	"Restaurant 10"	"Address 10"	"Contact Details 10"

Rest_emp_details

	_id ObjectId	rest_empid Int32	rest_empname String	rest_empemail String	rest_empaddress String	rest_empsalary Int32	rest_contactdetails String
1	ObjectId('64a9b9692e8c60b2d34…	1	"Updated Employee Name"	"employee1@example.com"	"Employee Address 1"	8000	"Updated Contact Details"
2	ObjectId('64a9b9692e8c60b2d34…	2	"Employee 2"	"employee2@example.com"	"Employee Address 2"	7000	No field
3	ObjectId('64a9b9692e8c60b2d34…	3	"Employee 3"	"employee3@example.com"	"Employee Address 3"	8000	No field
4	ObjectId('64a9b9692e8c60b2d34…	4	"Employee 4"	"employee4@example.com"	"Employee Address 4"	9000	No field
5	ObjectId('64a9b9692e8c60b2d34…	5	"Employee 5"	"employee5@example.com"	"Employee Address 5"	10000	No field
6	ObjectId('64a9b9692e8c60b2d34…	6	"Employee 6"	"employee6@example.com"	"Employee Address 6"	11000	No field
7	ObjectId('64a9b9692e8c60b2d34…	7	"Employee 7"	"employee7@example.com"	"Employee Address 7"	12000	No field
8	ObjectId('64a9b9692e8c60b2d34…	8	"Employee 8"	"employee8@example.com"	"Employee Address 8"	13000	No field
9	ObjectId('64a9b9692e8c60b2d34…	9	"Employee 9"	"employee9@example.com"	"Employee Address 9"	14000	No field
10	ObjectId('64a9b9692e8c60b2d34…	10	"Employee 10"	"employee10@example.com"	"Employee Address 10"	15000	No field

Invoice_details

	_id ObjectId	invoice_id Int32	invoice_itemdetails String	invoice_amountdetails String
1	ObjectId('64a9b9692e8c60b2d34…	1	"Item Details 1"	"Amount Details 1"
2	ObjectId('64a9b9692e8c60b2d34…	2	"Item Details 2"	"Amount Details 2"
3	ObjectId('64a9b9692e8c60b2d34…	3	"Item Details 3"	"Amount Details 3"
4	ObjectId('64a9b9692e8c60b2d34…	4	"Item Details 4"	"Amount Details 4"
5	ObjectId('64a9b9692e8c60b2d34…	5	"Item Details 5"	"Amount Details 5"
6	ObjectId('64a9b9692e8c60b2d34…	6	"Item Details 6"	"Amount Details 6"
7	ObjectId('64a9b9692e8c60b2d34…	7	"Item Details 7"	"Amount Details 7"
8	ObjectId('64a9b9692e8c60b2d34…	8	"Item Details 8"	"Amount Details 8"
9	ObjectId('64a9b9692e8c60b2d34…	9	"Item Details 9"	"Amount Details 9"
10	ObjectId('64a9b9692e8c60b2d34…	10	"Item Details 10"	"Amount Details 10"

2. Create database Shopping_center with collections

Shoppingcenter_details(shopcenter_registrationid,shopcenter _name,shopcenter_branchdetails,shopcenter_address,shop_c enteremailid,shopcenter_contactdetails)
Company(company_id,company_code,company_name,comp any_address,company_mail)
emp_details(rest_empid,rest_empname,rest_empemail,rest_e mpaddress,rest_empsalary)
Shop(shop_id,shop_code,shop_name, shopcenter_registrationid,floor_no,start_date) Insert 10 records for each key. Perform update query on floor, shopcenter_branchdetails,shop_code
Perform delete operation floor_no
Perform query to display all employee details whose name are Joe or Steev
Perform query to display shoppingcenter_branchdetails
“laystreet” and Adam road

```
import pymongo

# Connect to MongoDB
client = pymongo.MongoClient("mongodb://localhost:27017")
database = client["Shopping_center"]

# Get references to the collections
shopping_center_details_collection = database["Shoppingcenter_details"]
company_collection = database["Company"]
emp_details_collection = database["emp_details"]
shop_collection = database["Shop"]

# Function to insert records into the collections
def insert_record(collection, record):
    collection.insert_one(record)

# Function to update shop details
def update_shop_details(collection, shop_id, floor_no, branch_details, shop_code):
    collection.update_one(
        {"shop_id": shop_id},
```

```
        {"$set": {"floor_no": floor_no, "shopcenter_branchdetails": branch_details, "shop_code":
shop_code}}
    )

# Function to delete floor_no from shops
def delete_floor_no(collection, shop_id):
    collection.update_one(
        {"shop_id": shop_id},
        {"$unset": {"floor_no": ""}}
    )

# Function to query employee details by name
def query_employee_by_name(collection, names):
    results = collection.find({"rest_empname": {"$in": names}})
    return list(results)

# Function to query shopping center details by branch details and street
def query_shopping_center_by_branch(collection, branch_details, street):
    results = collection.find({"shopcenter_branchdetails": branch_details, "shopcenter_address":
{"$regex": street}})
    return list(results)

# Insert records for each collection
for i in range(1, 11):
    shopping_center_detail = {
        'shopcenter_registrationid': i,
        'shopcenter_name': f'Shopping Center {i}',
        'shopcenter_branchdetails': f'Branch Details {i}',
        'shopcenter_address': f'Street {i}, Adam Road',
        'shopcenter_emailid': f'shoppingcenter{i}@example.com',
        'shopcenter_contactdetails': f'Contact Details {i}'
    }
    insert_record(shopping_center_details_collection, shopping_center_detail)

    company_detail = {
        'company_id': i,
        'company_code': f'Company Code {i}',
        'company_name': f'Company {i}',
        'company_address': f'Company Address {i}',
        'company_mail': f'company{i}@example.com'
    }
    insert_record(company_collection, company_detail)

    emp_detail = {
        'rest_empid': i,
        'rest_empname': f'Employee Name {i}',
        'rest_empemail': f'employee{i}@example.com',
        'rest_empaddress': f'Employee Address {i}',
        'rest_empsalary': 5000 + i * 1000
```

```
    }
    insert_record(emp_details_collection, emp_detail)

    shop_detail = {
        'shop_id': i,
        'shop_code': f'Shop Code {i}',
        'shop_name': f'Shop {i}',
        'shopcenter_registrationid': i,
        'floor_no': i,
        'start_date': '2023-07-08'
    }
    insert_record(shop_collection, shop_detail)

# Update shop details
update_shop_details(shop_collection, 1, 2, "Updated Branch Details", "Updated Shop Code")

# Delete floor_no from shops
delete_floor_no(shop_collection, 2)

# Query employee details by name
employee_names = ["Joe", "Steev"]
results = query_employee_by_name(emp_details_collection, employee_names)
print("Employees with name Joe or Steev:")
for employee in results:
    print(employee)

# Query shopping center details by branch details and street
branch_details = "laystreet"
street = "Adam Road"
results = query_shopping_center_by_branch(shopping_center_details_collection, branch_details,
street)
print(f"Shopping centers with branch details '{branch_details}' and street '{street}':")
for center in results:
    print(center)
```

OUTPUT:

Shoppingcenter_details

	_id ObjectId	shopcenter_registrationid Int32	shopcenter_name String	shopcenter_branchdetails String	shopcenter_address String	shopcenter_emailid String	shopcenter_contactdetails Stri…
1	ObjectId('64aa20efda3e8b32819…	1	"Shopping Center 1"	"Branch Details 1"	"Street 1, Adam Road"	"shoppingcenter1@example.com"	"Contact Details 1"
2	ObjectId('64aa20efda3e8b32819…	2	"Shopping Center 2"	"Branch Details 2"	"Street 2, Adam Road"	"shoppingcenter2@example.com"	"Contact Details 2"
3	ObjectId('64aa20efda3e8b32819…	3	"Shopping Center 3"	"Branch Details 3"	"Street 3, Adam Road"	"shoppingcenter3@example.com"	"Contact Details 3"
4	ObjectId('64aa20efda3e8b32819…	4	"Shopping Center 4"	"Branch Details 4"	"Street 4, Adam Road"	"shoppingcenter4@example.com"	"Contact Details 4"
5	ObjectId('64aa20efda3e8b32819…	5	"Shopping Center 5"	"Branch Details 5"	"Street 5, Adam Road"	"shoppingcenter5@example.com"	"Contact Details 5"
6	ObjectId('64aa20efda3e8b32819…	6	"Shopping Center 6"	"Branch Details 6"	"Street 6, Adam Road"	"shoppingcenter6@example.com"	"Contact Details 6"
7	ObjectId('64aa20efda3e8b32819…	7	"Shopping Center 7"	"Branch Details 7"	"Street 7, Adam Road"	"shoppingcenter7@example.com"	"Contact Details 7"
8	ObjectId('64aa20efda3e8b32819…	8	"Shopping Center 8"	"Branch Details 8"	"Street 8, Adam Road"	"shoppingcenter8@example.com"	"Contact Details 8"
9	ObjectId('64aa20efda3e8b32819…	9	"Shopping Center 9"	"Branch Details 9"	"Street 9, Adam Road"	"shoppingcenter9@example.com"	"Contact Details 9"
10	ObjectId('64aa20efda3e8b32819…	10	"Shopping Center 10"	"Branch Details 10"	"Street 10, Adam Road"	"shoppingcenter10@example.com"	"Contact Details 10"

emp_details

	_id ObjectId	rest_empid Int32	rest_empname String	rest_empemail String	rest_empaddress String	rest_empsalary Int32
1	ObjectId('64aa20efda3e8b32819…	1	"Employee Name 1"	"employee1@example.com"	"Employee Address 1"	6000
2	ObjectId('64aa20efda3e8b32819…	2	"Employee Name 2"	"employee2@example.com"	"Employee Address 2"	7000
3	ObjectId('64aa20efda3e8b32819…	3	"Employee Name 3"	"employee3@example.com"	"Employee Address 3"	8000
4	ObjectId('64aa20efda3e8b32819…	4	"Employee Name 4"	"employee4@example.com"	"Employee Address 4"	9000
5	ObjectId('64aa20efda3e8b32819…	5	"Employee Name 5"	"employee5@example.com"	"Employee Address 5"	10000
6	ObjectId('64aa20efda3e8b32819…	6	"Employee Name 6"	"employee6@example.com"	"Employee Address 6"	11000
7	ObjectId('64aa20efda3e8b32819…	7	"Employee Name 7"	"employee7@example.com"	"Employee Address 7"	12000
8	ObjectId('64aa20efda3e8b32819…	8	"Employee Name 8"	"employee8@example.com"	"Employee Address 8"	13000
9	ObjectId('64aa20efda3e8b32819…	9	"Employee Name 9"	"employee9@example.com"	"Employee Address 9"	14000
10	ObjectId('64aa20efda3e8b32819…	10	"Employee Name 10"	"employee10@example.com"	"Employee Address 10"	15000

Company

	_id ObjectId	company_id Int32	company_code String	company_name String	company_address String	company_mail String
1	ObjectId('64aa20efda3e8b32819…	1	"Company Code 1"	"Company 1"	"Company Address 1"	"company1@example.com"
2	ObjectId('64aa20efda3e8b32819…	2	"Company Code 2"	"Company 2"	"Company Address 2"	"company2@example.com"
3	ObjectId('64aa20efda3e8b32819…	3	"Company Code 3"	"Company 3"	"Company Address 3"	"company3@example.com"
4	ObjectId('64aa20efda3e8b32819…	4	"Company Code 4"	"Company 4"	"Company Address 4"	"company4@example.com"
5	ObjectId('64aa20efda3e8b32819…	5	"Company Code 5"	"Company 5"	"Company Address 5"	"company5@example.com"
6	ObjectId('64aa20efda3e8b32819…	6	"Company Code 6"	"Company 6"	"Company Address 6"	"company6@example.com"
7	ObjectId('64aa20efda3e8b32819…	7	"Company Code 7"	"Company 7"	"Company Address 7"	"company7@example.com"
8	ObjectId('64aa20efda3e8b32819…	8	"Company Code 8"	"Company 8"	"Company Address 8"	"company8@example.com"
9	ObjectId('64aa20efda3e8b32819…	9	"Company Code 9"	"Company 9"	"Company Address 9"	"company9@example.com"
10	ObjectId('64aa20efda3e8b32819…	10	"Company Code 10"	"Company 10"	"Company Address 10"	"company10@example.com"

Shop

	_id ObjectId	shop_id Int32	shop_code String	shop_name String	shopcenter_registrationid Int32	floor_no Int32	start_date String	shopcenter_branchdetails String
1	ObjectId('64aa20efda3e8b32819…	1	"Updated Shop Code"	"Shop 1"	1	2	"2023-07-08"	"Updated Branch Details"
2	ObjectId('64aa20efda3e8b32819…	2	"Shop Code 2"	"Shop 2"	2	No field	"2023-07-08"	No field
3	ObjectId('64aa20efda3e8b32819…	3	"Shop Code 3"	"Shop 3"	3	3	"2023-07-08"	No field
4	ObjectId('64aa20efda3e8b32819…	4	"Shop Code 4"	"Shop 4"	4	4	"2023-07-08"	No field
5	ObjectId('64aa20efda3e8b32819…	5	"Shop Code 5"	"Shop 5"	5	5	"2023-07-08"	No field
6	ObjectId('64aa20efda3e8b32819…	6	"Shop Code 6"	"Shop 6"	6	6	"2023-07-08"	No field
7	ObjectId('64aa20efda3e8b32819…	7	"Shop Code 7"	"Shop 7"	7	7	"2023-07-08"	No field
8	ObjectId('64aa20efda3e8b32819…	8	"Shop Code 8"	"Shop 8"	8	8	"2023-07-08"	No field
9	ObjectId('64aa20efda3e8b32819…	9	"Shop Code 9"	"Shop 9"	9	9	"2023-07-08"	No field
10	ObjectId('64aa20efda3e8b32819…	10	"Shop Code 10"	"Shop 10"	10	10	"2023-07-08"	No field

3. Create Database O'Reilly with collections
Attendance(Attendee_id,First_name,Last_name,Phone_num, Email,VIP)
Company(Company_id,Name,Description,Primary_contact_ attendee_id)
Presentation(Presentation_id,Booked_companyid,Booked_ro
omid,Start_time,End_time)
Room(Room_id,Floor_number,Seat_capacity)
Insert 10 records for each key in collections.
Perform query to display Company_name=”Book_myspace”,
“Available_space”,”Rent_outspace”

```
import pymongo

# Connect to MongoDB
client = pymongo.MongoClient("mongodb://localhost:27017")
database = client["OReilly"]

# Get references to the collections
attendance_collection = database["Attendance"]
company_collection = database["Company"]
presentation_collection = database["Presentation"]
room_collection = database["Room"]
presentation_attendance_collection = database["Presentation_attendance"]

# Function to insert records into the collections
def insert_record(collection, record):
    collection.insert_one(record)

# Function to query company details
def query_company_details(collection, company_name):
    result = collection.find_one({"Name": company_name})
    return result

# Insert records for each collection
for i in range(1, 11):
    attendee = {
        'Attendee_id': i,
        'First_name': f'First Name {i}',
        'Last_name': f'Last Name {i}',
        'Phone_num': f'Phone Number {i}',
        'Email': f'attendee{i}@example.com',
        'VIP': True if i % 2 == 0 else False
    }
    insert_record(attendance_collection, attendee)

    company_record = {
        'Company_id': i,
        'Name': f'Company {i}',
        'Description': f'Description {i}',
        'Primary_contact_attendee_id': i
```

```
    }
    insert_record(company_collection, company_record)

    presentation_record = {
        'Presentation_id': i,
        'Booked_companyid': i,
        'Booked_roomid': i,
        'Start_time': f'Start Time {i}',
        'End_time': f'End Time {i}'
    }
    insert_record(presentation_collection, presentation_record)

    room_record = {
        'Room_id': i,
        'Floor_number': f'Floor Number {i}',
        'Seat_capacity': 100 + i
    }
    insert_record(room_collection, room_record)

    presentation_attendance_record = {
        'Ticket_id': i,
        'Presentation_id': i,
        'Attendee_id': i
    }
    insert_record(presentation_attendance_collection, presentation_attendance_record)

# Query company details
company_name = "Book_myspace"
result = query_company_details(company_collection, company_name)
if result:
    available_space = result['Seat_capacity'] - result['Total_tickets']
    print("Company Name:", result['Name'])
    print("Available Space:", available_space)
    print("Rent Out Space:", result['Rent_out_space'])
else:
    print("Company not found")
```

OUTPUT:

Attendance

	_id ObjectId	Attendee_id Int32	First_name String	Last_name String	Phone_num String	Email String	VIP Boolean
1	ObjectId('64aa247ada3e8b32819…	1	"First Name 1"	"Last Name 1"	"Phone Number 1"	"attendee1@example.com"	false
2	ObjectId('64aa247ada3e8b32819…	2	"First Name 2"	"Last Name 2"	"Phone Number 2"	"attendee2@example.com"	true
3	ObjectId('64aa247ada3e8b32819…	3	"First Name 3"	"Last Name 3"	"Phone Number 3"	"attendee3@example.com"	false
4	ObjectId('64aa247ada3e8b32819…	4	"First Name 4"	"Last Name 4"	"Phone Number 4"	"attendee4@example.com"	true
5	ObjectId('64aa247ada3e8b32819…	5	"First Name 5"	"Last Name 5"	"Phone Number 5"	"attendee5@example.com"	false
6	ObjectId('64aa247ada3e8b32819…	6	"First Name 6"	"Last Name 6"	"Phone Number 6"	"attendee6@example.com"	true
7	ObjectId('64aa247ada3e8b32819…	7	"First Name 7"	"Last Name 7"	"Phone Number 7"	"attendee7@example.com"	false
8	ObjectId('64aa247ada3e8b32819…	8	"First Name 8"	"Last Name 8"	"Phone Number 8"	"attendee8@example.com"	true
9	ObjectId('64aa247ada3e8b32819…	9	"First Name 9"	"Last Name 9"	"Phone Number 9"	"attendee9@example.com"	false
10	ObjectId('64aa247ada3e8b32819…	10	"First Name 10"	"Last Name 10"	"Phone Number 10"	"attendee10@example.com"	true

Company

	_id ObjectId	Company_id Int32	Name String	Description String	Primary_contact_attendee_id In…
1	ObjectId('64aa247ada3e8b32819…	1	"Company 1"	"Description 1"	1
2	ObjectId('64aa247ada3e8b32819…	2	"Company 2"	"Description 2"	2
3	ObjectId('64aa247ada3e8b32819…	3	"Company 3"	"Description 3"	3
4	ObjectId('64aa247ada3e8b32819…	4	"Company 4"	"Description 4"	4
5	ObjectId('64aa247ada3e8b32819…	5	"Company 5"	"Description 5"	5
6	ObjectId('64aa247ada3e8b32819…	6	"Company 6"	"Description 6"	6
7	ObjectId('64aa247ada3e8b32819…	7	"Company 7"	"Description 7"	7
8	ObjectId('64aa247ada3e8b32819…	8	"Company 8"	"Description 8"	8
9	ObjectId('64aa247ada3e8b32819…	9	"Company 9"	"Description 9"	9
10	ObjectId('64aa247ada3e8b32819…	10	"Company 10"	"Description 10"	10

Presentation

	_id ObjectId	Presentation_id Int32	Booked_companyid Int32	Booked_roomid Int32	Start_time String	End_time String
1	ObjectId('64aa247ada3e8b32819…	1	1	1	"Start Time 1"	"End Time 1"
2	ObjectId('64aa247ada3e8b32819…	2	2	2	"Start Time 2"	"End Time 2"
3	ObjectId('64aa247ada3e8b32819…	3	3	3	"Start Time 3"	"End Time 3"
4	ObjectId('64aa247ada3e8b32819…	4	4	4	"Start Time 4"	"End Time 4"
5	ObjectId('64aa247ada3e8b32819…	5	5	5	"Start Time 5"	"End Time 5"
6	ObjectId('64aa247ada3e8b32819…	6	6	6	"Start Time 6"	"End Time 6"
7	ObjectId('64aa247ada3e8b32819…	7	7	7	"Start Time 7"	"End Time 7"
8	ObjectId('64aa247ada3e8b32819…	8	8	8	"Start Time 8"	"End Time 8"
9	ObjectId('64aa247ada3e8b32819…	9	9	9	"Start Time 9"	"End Time 9"
10	ObjectId('64aa247ada3e8b32819…	10	10	10	"Start Time 10"	"End Time 10"

Room

	_id ObjectId	Room_id Int32	Floor_number String	Seat_capacity Int32
1	ObjectId('64aa247ada3e8b32819…	1	"Floor Number 1"	101
2	ObjectId('64aa247ada3e8b32819…	2	"Floor Number 2"	102
3	ObjectId('64aa247ada3e8b32819…	3	"Floor Number 3"	103
4	ObjectId('64aa247ada3e8b32819…	4	"Floor Number 4"	104
5	ObjectId('64aa247ada3e8b32819…	5	"Floor Number 5"	105
6	ObjectId('64aa247ada3e8b32819…	6	"Floor Number 6"	106
7	ObjectId('64aa247ada3e8b32819…	7	"Floor Number 7"	107
8	ObjectId('64aa247ada3e8b32819…	8	"Floor Number 8"	108
9	ObjectId('64aa247ada3e8b32819…	9	"Floor Number 9"	109
10	ObjectId('64aa247ada3e8b32819…	10	"Floor Number 10"	110

4. Create Database Library_mangement with collections
Students(stud_id,stud_name,stud_surname,dob,gender,class)
Borrow(borrow_id,stud_id,book_id,taken_date)
Books(book_id,name,pagecount,point,author_id,type_id)
Author(author_id,author_name,author_surname)
Type(type_id,type_name)
Insert 10 records to each collections key.
Perform update in book name=”Advance C” whose book name=”Let Us C” and pagecount=”201”
List all Borrow books whose taken_date=11th -july2022 or taken_date =26th-August-2022

```
import pymongo

# Connect to MongoDB
client = pymongo.MongoClient("mongodb://localhost:27017")
database = client["OReilly"]

# Get references to the collections
attendance_collection = database["Attendance"]
company_collection = database["Company"]
presentation_collection = database["Presentation"]
room_collection = database["Room"]
presentation_attendance_collection = database["Presentation_attendance"]

# Function to insert records into the collections
def insert_record(collection, record):
    collection.insert_one(record)

# Function to query company details
def query_company_details(collection, company_name):
    result = collection.find_one({"Name": company_name})
    return result

# Insert records for each collection
for i in range(1, 11):
    attendee = {
        'Attendee_id': i,
        'First_name': f'First Name {i}',
        'Last_name': f'Last Name {i}',
        'Phone_num': f'Phone Number {i}',
        'Email': f'attendee{i}@example.com',
        'VIP': True if i % 2 == 0 else False
    }
    insert_record(attendance_collection, attendee)

    company_record = {
        'Company_id': i,
```

```
        'Name': f'Company {i}',
        'Description': f'Description {i}',
        'Primary_contact_attendee_id': i
    }
    insert_record(company_collection, company_record)

    presentation_record = {
        'Presentation_id': i,
        'Booked_companyid': i,
        'Booked_roomid': i,
        'Start_time': f'Start Time {i}',
        'End_time': f'End Time {i}'
    }
    insert_record(presentation_collection, presentation_record)

    room_record = {
        'Room_id': i,
        'Floor_number': f'Floor Number {i}',
        'Seat_capacity': 100 + i
    }
    insert_record(room_collection, room_record)

    presentation_attendance_record = {
        'Ticket_id': i,
        'Presentation_id': i,
        'Attendee_id': i
    }
    insert_record(presentation_attendance_collection, presentation_attendance_record)

# Query company details
company_name = "Book_myspace"
result = query_company_details(company_collection, company_name)
if result:
    available_space = result['Seat_capacity'] - result['Total_tickets']
    print("Company Name:", result['Name'])
    print("Available Space:", available_space)
    print("Rent Out Space:", result['Rent_out_space'])
else:
    print("Company not found")
```

OUTPUT:

Borrow

	_id ObjectId	borrow_id Int32	stud_id Int32	book_id Int32	taken_date String	ass String
1	ObjectId('64aa264dda3e8b32819…	1	1	1	"2022-08-26"	lass 1"
2	ObjectId('64aa264dda3e8b32819…	2	2	2	"2022-07-11"	lass 2"
3	ObjectId('64aa264dda3e8b32819…	3	3	3	"2022-08-26"	lass 3"
4	ObjectId('64aa264dda3e8b32819…	4	4	4	"2022-07-11"	lass 4"
5	ObjectId('64aa264dda3e8b32819…	5	5	5	"2022-08-26"	lass 5"
6	ObjectId('64aa264dda3e8b32819…	6	6	6	"2022-07-11"	lass 6"
7	ObjectId('64aa264dda3e8b32819…	7	7	7	"2022-08-26"	lass 7"
8	ObjectId('64aa264dda3e8b32819…	8	8	8	"2022-07-11"	lass 8"
9	ObjectId('64aa264dda3e8b32819…	9	9	9	"2022-08-26"	lass 9"
10	ObjectId('64aa264dda3e8b32819…	10	10	10	"2022-07-11"	lass 10"

Books

	_id ObjectId	book_id Int32	name String	pagecount Int32	point Int32	author_id Int32	type_id Int32
1	ObjectId('64aa264dda3e8b32819…	1	"Book 1"	201	5	1	1
2	ObjectId('64aa264dda3e8b32819…	2	"Book 2"	202	10	2	2
3	ObjectId('64aa264dda3e8b32819…	3	"Book 3"	203	15	3	3
4	ObjectId('64aa264dda3e8b32819…	4	"Book 4"	204	20	4	4
5	ObjectId('64aa264dda3e8b32819…	5	"Book 5"	205	25	5	5
6	ObjectId('64aa264dda3e8b32819…	6	"Book 6"	206	30	6	6
7	ObjectId('64aa264dda3e8b32819…	7	"Book 7"	207	35	7	7
8	ObjectId('64aa264dda3e8b32819…	8	"Book 8"	208	40	8	8
9	ObjectId('64aa264dda3e8b32819…	9	"Book 9"	209	45	9	9
10	ObjectId('64aa264dda3e8b32819…	10	"Book 10"	210	50	10	10

Author

	_id ObjectId	author_id Int32	author_name String	author_surname String
1	ObjectId('64aa264dda3e8b32819…	1	"Author 1"	"Surname 1"
2	ObjectId('64aa264dda3e8b32819…	2	"Author 2"	"Surname 2"
3	ObjectId('64aa264dda3e8b32819…	3	"Author 3"	"Surname 3"
4	ObjectId('64aa264dda3e8b32819…	4	"Author 4"	"Surname 4"
5	ObjectId('64aa264dda3e8b32819…	5	"Author 5"	"Surname 5"
6	ObjectId('64aa264dda3e8b32819…	6	"Author 6"	"Surname 6"
7	ObjectId('64aa264dda3e8b32819…	7	"Author 7"	"Surname 7"
8	ObjectId('64aa264dda3e8b32819…	8	"Author 8"	"Surname 8"
9	ObjectId('64aa264dda3e8b32819…	9	"Author 9"	"Surname 9"
10	ObjectId('64aa264dda3e8b32819…	10	"Author 10"	"Surname 10"

	_id ObjectId	type_id Int32	type_name String
1	ObjectId('64aa264dda3e8b32819…	1	"Type 1"
2	ObjectId('64aa264dda3e8b32819…	2	"Type 2"
3	ObjectId('64aa264dda3e8b32819…	3	"Type 3"
4	ObjectId('64aa264dda3e8b32819…	4	"Type 4"
5	ObjectId('64aa264dda3e8b32819…	5	"Type 5"
6	ObjectId('64aa264dda3e8b32819…	6	"Type 6"
7	ObjectId('64aa264dda3e8b32819…	7	"Type 7"
8	ObjectId('64aa264dda3e8b32819…	8	"Type 8"
9	ObjectId('64aa264dda3e8b32819…	9	"Type 9"
10	ObjectId('64aa264dda3e8b32819…	10	"Type 10"

5. Create database Nursary_management with following collections
Supplier(Sup_id,Sup_name,Mobile,Prod_no,Email,Location_ id,Price)
Commodities(product_no,product_price,product_name,Qua ntity,Supply)
Customers(Cust_id,Location_id,Name,Email,Phone,Invoice)
Category(Category_id,Category_name,Product_no)
Employee(Employee_id,Employee_name,Email,Mobile_no,L ocation_id,Desgn_name,)
Insert 8 records for each collection key.
Display all supplier details whose price is greater than 8000 and less than 30000
Display all Commodities whose product_price is less than equal to 8000.

```
import pymongo

# Connect to MongoDB
client = pymongo.MongoClient("mongodb://localhost:27017")
database = client["Nursary_management"]

# Get references to the collections
suppliers_collection = database["Supplier"]
commodities_collection = database["Commodities"]
customers_collection = database["Customers"]
categories_collection = database["Category"]
employees_collection = database["Employee"]

# Function to insert records into the collections
def insert_record(collection, record):
    collection.insert_one(record)

# Function to query suppliers based on price range
def query_suppliers_by_price(collection, min_price, max_price):
    results = collection.find({"Price": {"$gt": min_price, "$lt": max_price}})
    return list(results)

# Function to query commodities based on product_price
def query_commodities_by_price(collection, max_price):
    results = collection.find({"product_price": {"$lte": max_price}})
    return list(results)

# Insert records for each collection
for i in range(1, 9):
    supplier = {
        'Sup_id': i,
        'Sup_name': f'Supplier {i}',
        'Mobile': f'123456789{i}',
        'Prod_no': f'Product {i}',
        'Email': f'supplier{i}@example.com',
        'Location_id': f'Location {i}',
        'Price': 1000 * i
    }
    insert_record(suppliers_collection, supplier)
```

```
    commodity = {
        'product_no': f'Product {i}',
        'product_price': 1000 * i,
        'product_name': f'Commodity {i}',
        'Quantity': 10 * i,
        'Supply': f'Supply {i}'
    }
    insert_record(commodities_collection, commodity)

    customer = {
        'Cust_id': i,
        'Location_id': f'Location {i}',
        'Name': f'Customer {i}',
        'Email': f'customer{i}@example.com',
        'Phone': f'987654321{i}',
        'Invoice': f'Invoice {i}'
    }
    insert_record(customers_collection, customer)

    category = {
        'Category_id': i,
        'Category_name': f'Category {i}',
        'Product_no': f'Product {i}'
    }
    insert_record(categories_collection, category)

    employee = {
        'Employee_id': i,
        'Employee_name': f'Employee {i}',
        'Email': f'employee{i}@example.com',
        'Mobile_no': f'123456789{i}',
        'Location_id': f'Location {i}',
        'Desgn_name': f'Designation {i}'
    }
    insert_record(employees_collection, employee)

# Query suppliers with price between 8000 and 30000
results = query_suppliers_by_price(suppliers_collection, 8000, 30000)
print("Suppliers with price between 8000 and 30000:")
for supplier in results:
    print(supplier)

# Query commodities with product_price less than or equal to 8000
results = query_commodities_by_price(commodities_collection, 8000)
print("Commodities with product_price less than or equal to 8000:")
for commodity in results:
    print(commodity)
```

OUTPUT:

Supplier

	_id ObjectId	Sup_id Int32	Sup_name String	Mobile String	Prod_no String	Email String	Location_id String	Price Int32
1	ObjectId('64aa274bda3e8b32819…	1	"Supplier 1"	"1234567891"	"Product 1"	"supplier1@example.com"	"Location 1"	1000
2	ObjectId('64aa274bda3e8b32819…	2	"Supplier 2"	"1234567892"	"Product 2"	"supplier2@example.com"	"Location 2"	2000
3	ObjectId('64aa274bda3e8b32819…	3	"Supplier 3"	"1234567893"	"Product 3"	"supplier3@example.com"	"Location 3"	3000
4	ObjectId('64aa274bda3e8b32819…	4	"Supplier 4"	"1234567894"	"Product 4"	"supplier4@example.com"	"Location 4"	4000
5	ObjectId('64aa274bda3e8b32819…	5	"Supplier 5"	"1234567895"	"Product 5"	"supplier5@example.com"	"Location 5"	5000
6	ObjectId('64aa274bda3e8b32819…	6	"Supplier 6"	"1234567896"	"Product 6"	"supplier6@example.com"	"Location 6"	6000
7	ObjectId('64aa274bda3e8b32819…	7	"Supplier 7"	"1234567897"	"Product 7"	"supplier7@example.com"	"Location 7"	7000
8	ObjectId('64aa274bda3e8b32819…	8	"Supplier 8"	"1234567898"	"Product 8"	"supplier8@example.com"	"Location 8"	8000

Commodities

	_id ObjectId	product_no String	product_price Int32	product_name String	Quantity Int32	Supply String
1	ObjectId('64aa274bda3e8b32819…	"Product 1"	1000	"Commodity 1"	10	"Supply 1"
2	ObjectId('64aa274bda3e8b32819…	"Product 2"	2000	"Commodity 2"	20	"Supply 2"
3	ObjectId('64aa274bda3e8b32819…	"Product 3"	3000	"Commodity 3"	30	"Supply 3"
4	ObjectId('64aa274bda3e8b32819…	"Product 4"	4000	"Commodity 4"	40	"Supply 4"
5	ObjectId('64aa274bda3e8b32819…	"Product 5"	5000	"Commodity 5"	50	"Supply 5"
6	ObjectId('64aa274bda3e8b32819…	"Product 6"	6000	"Commodity 6"	60	"Supply 6"
7	ObjectId('64aa274bda3e8b32819…	"Product 7"	7000	"Commodity 7"	70	"Supply 7"
8	ObjectId('64aa274bda3e8b32819…	"Product 8"	8000	"Commodity 8"	80	"Supply 8"

Customers

	_id ObjectId	Cust_id Int32	Location_id String	Name String	Email String	Phone String	Invoice String
1	ObjectId('64aa274bda3e8b32819…	1	"Location 1"	"Customer 1"	"customer1@example.com"	"9876543211"	"Invoice 1"
2	ObjectId('64aa274bda3e8b32819…	2	"Location 2"	"Customer 2"	"customer2@example.com"	"9876543212"	"Invoice 2"
3	ObjectId('64aa274bda3e8b32819…	3	"Location 3"	"Customer 3"	"customer3@example.com"	"9876543213"	"Invoice 3"
4	ObjectId('64aa274bda3e8b32819…	4	"Location 4"	"Customer 4"	"customer4@example.com"	"9876543214"	"Invoice 4"
5	ObjectId('64aa274bda3e8b32819…	5	"Location 5"	"Customer 5"	"customer5@example.com"	"9876543215"	"Invoice 5"
6	ObjectId('64aa274bda3e8b32819…	6	"Location 6"	"Customer 6"	"customer6@example.com"	"9876543216"	"Invoice 6"
7	ObjectId('64aa274bda3e8b32819…	7	"Location 7"	"Customer 7"	"customer7@example.com"	"9876543217"	"Invoice 7"
8	ObjectId('64aa274bda3e8b32819…	8	"Location 8"	"Customer 8"	"customer8@example.com"	"9876543218"	"Invoice 8"

Category

	_id ObjectId	Category_id Int32	Category_name String	Product_no String
1	ObjectId('64aa274bda3e8b32819…	1	"Category 1"	"Product 1"
2	ObjectId('64aa274bda3e8b32819…	2	"Category 2"	"Product 2"
3	ObjectId('64aa274bda3e8b32819…	3	"Category 3"	"Product 3"
4	ObjectId('64aa274bda3e8b32819…	4	"Category 4"	"Product 4"
5	ObjectId('64aa274bda3e8b32819…	5	"Category 5"	"Product 5"
6	ObjectId('64aa274bda3e8b32819…	6	"Category 6"	"Product 6"
7	ObjectId('64aa274bda3e8b32819…	7	"Category 7"	"Product 7"
8	ObjectId('64aa274bda3e8b32819…	8	"Category 8"	"Product 8"

Employee

	_id ObjectId	Employee_id Int32	Employee_name String	Email String	Mobile_no String	Location_id String	Desgn_name String
1	ObjectId('64aa274bda3e8b32819…	1	"Employee 1"	"employee1@example.com"	"1234567891"	"Location 1"	"Designation 1"
2	ObjectId('64aa274bda3e8b32819…	2	"Employee 2"	"employee2@example.com"	"1234567892"	"Location 2"	"Designation 2"
3	ObjectId('64aa274bda3e8b32819…	3	"Employee 3"	"employee3@example.com"	"1234567893"	"Location 3"	"Designation 3"
4	ObjectId('64aa274bda3e8b32819…	4	"Employee 4"	"employee4@example.com"	"1234567894"	"Location 4"	"Designation 4"
5	ObjectId('64aa274bda3e8b32819…	5	"Employee 5"	"employee5@example.com"	"1234567895"	"Location 5"	"Designation 5"
6	ObjectId('64aa274bda3e8b32819…	6	"Employee 6"	"employee6@example.com"	"1234567896"	"Location 6"	"Designation 6"
7	ObjectId('64aa274bda3e8b32819…	7	"Employee 7"	"employee7@example.com"	"1234567897"	"Location 7"	"Designation 7"
8	ObjectId('64aa274bda3e8b32819…	8	"Employee 8"	"employee8@example.com"	"1234567898"	"Location 8"	"Designation 8"

6. Create database Inventory Management with following collections:

Product(prod_id,prod_name,part_number,product_label,inv entory_received,inventory_shipped,inventory_onhand,Minim um_requirement)
Purchases(pur_id,supplier_id,product_id,purchasedate)
Orders(id,Title,Customer_name,prod_id,order_date)
Supplier(supplier_id,sup_name)
Insert 12 records for each collection key
List all the product details whose minimum_requirement is less than 5 but greater than zero.
Update Supplier whose sup_name is Thomas

```
import pymongo

# Connect to MongoDB
client = pymongo.MongoClient("mongodb://localhost:27017")
database = client["InventoryManagement"]

# Get references to the collections
products_collection = database["Product"]
purchases_collection = database["Purchases"]
orders_collection = database["Orders"]
suppliers_collection = database["Supplier"]

# Function to insert records into the collections
def insert_record(collection, record):
    collection.insert_one(record)

# Function to query products based on minimum requirement
def query_products_min_req(collection):
    results = collection.find({"Minimum_requirement": {"$gt": 0, "$lt": 5}})
    return list(results)

# Function to update supplier by name
def update_supplier(collection, sup_name, new_name):
    collection.update_many({"sup_name": sup_name}, {"$set": {"sup_name": new_name}})

# Insert records for each collection
```

```
for i in range(1, 13):
    product = {
        'prod_id': i,
        'prod_name': f'Product {i}',
        'part_number': f'Part {i}',
        'product_label': f'Label {i}',
        'inventory_received': 0,
        'inventory_shipped': 0,
        'inventory_onhand': 0,
        'Minimum_requirement': i % 6  # Example minimum requirement based on index
    }
    insert_record(products_collection, product)

    purchase = {
        'pur_id': i,
        'supplier_id': i,
        'product_id': i,
        'purchasedate': '2023-07-08'  # Example purchase date
    }
    insert_record(purchases_collection, purchase)

    order = {
        'id': i,
        'Title': f'Order {i}',
        'Customer_name': f'Customer {i}',
        'prod_id': i,
        'order_date': '2023-07-08'  # Example order date
    }
    insert_record(orders_collection, order)

    supplier = {
        'supplier_id': i,
        'sup_name': f'Supplier {i}'
    }
    insert_record(suppliers_collection, supplier)

# Query products with minimum requirement between 0 and 5
results = query_products_min_req(products_collection)
print("Products with minimum requirement between 0 and 5:")
for product in results:
    print(product)

# Update supplier by name
update_supplier(suppliers_collection, 'Supplier 5', 'Thomas')

# Print the updated supplier collection
print("Updated Suppliers:")
for supplier in suppliers_collection.find():
    print(supplier)
```

OUTPUT:

Product

	_id ObjectId	prod_id Int32	prod_name String	part_number String	product_label String	inventory_received Int32	inventory_shipped Int32	inventory_onhand Int32	Minimum_requirem
1	ObjectId('64aa2908da3e8b32819…	1	"Product 1"	"Part 1"	"Label 1"	0	0	0	1
2	ObjectId('64aa2908da3e8b32819…	2	"Product 2"	"Part 2"	"Label 2"	0	0	0	2
3	ObjectId('64aa2908da3e8b32819…	3	"Product 3"	"Part 3"	"Label 3"	0	0	0	3
4	ObjectId('64aa2908da3e8b32819…	4	"Product 4"	"Part 4"	"Label 4"	0	0	0	4
5	ObjectId('64aa2908da3e8b32819…	5	"Product 5"	"Part 5"	"Label 5"	0	0	0	5
6	ObjectId('64aa2908da3e8b32819…	6	"Product 6"	"Part 6"	"Label 6"	0	0	0	0
7	ObjectId('64aa2908da3e8b32819…	7	"Product 7"	"Part 7"	"Label 7"	0	0	0	1
8	ObjectId('64aa2908da3e8b32819…	8	"Product 8"	"Part 8"	"Label 8"	0	0	0	2
9	ObjectId('64aa2908da3e8b32819…	9	"Product 9"	"Part 9"	"Label 9"	0	0	0	3
10	ObjectId('64aa2908da3e8b32819…	10	"Product 10"	"Part 10"	"Label 10"	0	0	0	4
11	ObjectId('64aa2908da3e8b32819…	11	"Product 11"	"Part 11"	"Label 11"	0	0	0	5
12	ObjectId('64aa2908da3e8b32819…	12	"Product 12"	"Part 12"	"Label 12"	0	0	0	0

Purchases

	_id ObjectId	pur_id Int32	supplier_id Int32	product_id Int32	purchasedate String
1	ObjectId('64aa2908da3e8b32819…	1	1	1	"2023-07-08"
2	ObjectId('64aa2908da3e8b32819…	2	2	2	"2023-07-08"
3	ObjectId('64aa2908da3e8b32819…	3	3	3	"2023-07-08"
4	ObjectId('64aa2908da3e8b32819…	4	4	4	"2023-07-08"
5	ObjectId('64aa2908da3e8b32819…	5	5	5	"2023-07-08"
6	ObjectId('64aa2908da3e8b32819…	6	6	6	"2023-07-08"
7	ObjectId('64aa2908da3e8b32819…	7	7	7	"2023-07-08"
8	ObjectId('64aa2908da3e8b32819…	8	8	8	"2023-07-08"
9	ObjectId('64aa2908da3e8b32819…	9	9	9	"2023-07-08"
10	ObjectId('64aa2908da3e8b32819…	10	10	10	"2023-07-08"
11	ObjectId('64aa2908da3e8b32819…	11	11	11	"2023-07-08"
12	ObjectId('64aa2908da3e8b32819…	12	12	12	"2023-07-08"

Orders

	_id ObjectId	id Int32	Title String	Customer_name String	prod_id Int32	order_date String
1	ObjectId('64aa2908da3e8b32819…	1	"Order 1"	"Customer 1"	1	"2023-07-08"
2	ObjectId('64aa2908da3e8b32819…	2	"Order 2"	"Customer 2"	2	"2023-07-08"
3	ObjectId('64aa2908da3e8b32819…	3	"Order 3"	"Customer 3"	3	"2023-07-08"
4	ObjectId('64aa2908da3e8b32819…	4	"Order 4"	"Customer 4"	4	"2023-07-08"
5	ObjectId('64aa2908da3e8b32819…	5	"Order 5"	"Customer 5"	5	"2023-07-08"
6	ObjectId('64aa2908da3e8b32819…	6	"Order 6"	"Customer 6"	6	"2023-07-08"
7	ObjectId('64aa2908da3e8b32819…	7	"Order 7"	"Customer 7"	7	"2023-07-08"
8	ObjectId('64aa2908da3e8b32819…	8	"Order 8"	"Customer 8"	8	"2023-07-08"
9	ObjectId('64aa2908da3e8b32819…	9	"Order 9"	"Customer 9"	9	"2023-07-08"
10	ObjectId('64aa2908da3e8b32819…	10	"Order 10"	"Customer 10"	10	"2023-07-08"
11	ObjectId('64aa2908da3e8b32819…	11	"Order 11"	"Customer 11"	11	"2023-07-08"
12	ObjectId('64aa2908da3e8b32819…	12	"Order 12"	"Customer 12"	12	"2023-07-08"

Supplier

	_id ObjectId	supplier_id Int32	sup_name String
1	ObjectId('64aa2908da3e8b32819…	1	"Supplier 1"
2	ObjectId('64aa2908da3e8b32819…	2	"Supplier 2"
3	ObjectId('64aa2908da3e8b32819…	3	"Supplier 3"
4	ObjectId('64aa2908da3e8b32819…	4	"Supplier 4"
5	ObjectId('64aa2908da3e8b32819…	5	"Thomas"
6	ObjectId('64aa2908da3e8b32819…	6	"Supplier 6"
7	ObjectId('64aa2908da3e8b32819…	7	"Supplier 7"
8	ObjectId('64aa2908da3e8b32819…	8	"Supplier 8"
9	ObjectId('64aa2908da3e8b32819…	9	"Supplier 9"
10	ObjectId('64aa2908da3e8b32819…	10	"Supplier 10"
11	ObjectId('64aa2908da3e8b32819…	11	"Supplier 11"
12	ObjectId('64aa2908da3e8b32819…	12	"Supplier 12"

7. Write a NumPy program to convert a list of numeric values into a one-dimensional NumPy array.

```
import numpy as np

# Create a list of numeric values
numeric_list = [1, 2, 3, 4, 5]

# Convert the list to a NumPy array
numpy_array = np.array(numeric_list)

# Print the NumPy array
print(numpy_array)
```

OUTPUT:

```
[1 2 3 4 5]
```

8. Write a NumPy program to reverse an array (the first element becomes the last).

```
import numpy as np

# Create the original array
original_array = np.array([1, 2, 3, 4, 5])

# Reverse the array
reversed_array = np.flip(original_array)

# Print the results
print("Original array:")
print(original_array)
print("Reversed array:")
print(reversed_array)
```

OUTPUT:

```
Original array:
[1 2 3 4 5]
Reversed array:
[5 4 3 2 1]
```

9. Write a NumPy program to append values to the end of an array.

```
import numpy as np

# Create the original array
original_array = np.array([10, 20, 30])

# Values to append
values_to_append = np.array([40, 50, 60, 70, 80, 90])

# Append values to the end of the array
appended_array = np.append(original_array, values_to_append)

# Print the results
print("Original array:")
print(original_array)
print("After appending values to the end of the array:")
print(appended_array)
```

OUTPUT:

```
Original array:
[10 20 30]
After appending values to the end of the array:
[10 20 30 40 50 60 70 80 90]
```

10. Write a NumPy program to create an empty and full array.

```
import numpy as np

# Create an empty array
empty_array = np.empty((4, 4))
print(empty_array)

# Create a full array
full_array = np.full((3, 3), 6)
print(full_array)
```

OUTPUT:

```
[[6.23042070e-307 4.67296746e-307 1.69121096e-306 1.33511290e-306]
 [6.23058368e-307 2.22522597e-306 1.33511969e-306 1.37962320e-306]
 [9.34604358e-307 9.79101082e-307 1.78020576e-306 1.69119873e-306]
 [2.22522868e-306 1.24611809e-306 8.06632139e-308 1.60221208e-306]]
[[6 6 6]
 [6 6 6]
 [6 6 6]]
```

11. Write a NumPy program to convert Centigrade degrees into Fahrenheit degrees. Centigrade values are stored in a NumPy array.

Sample Array:
Values in Fahrenheit degrees [0, 12, 45.21, 34, 99.91]
Values in Centigrade degrees [-17.78, -11.11, 7.34, 1.11, 37.73, 0.]

```
import numpy as np

# Create the array of Centigrade degrees
centigrade_array = np.array([-17.78, -11.11, 7.34, 1.11, 37.73, 0.])

# Convert Centigrade to Fahrenheit
fahrenheit_array = centigrade_array * 9/5 + 32

# Print the results
print("Values in Centigrade degrees:", centigrade_array)
print("Values in Fahrenheit degrees:", fahrenheit_array)
```

OUTPUT:

```
Values in Centigrade degrees: [-17.78 -11.11  7.34  1.11 37.73  0. ]
Values in Fahrenheit degrees: [-4.0000e-03  1.2002e+01  4.5212e+01  3.3
998e+01  9.9914e+01  3.2000e+01]
```

12. Write a NumPy program to find common values between two arrays.

```
import numpy as np

# Create the two arrays
array1 = np.array([0, 10, 20, 40, 60])
array2 = np.array([10, 30, 40])

# Find common values
common_values = np.intersect1d(array1, array2)
print("Array1:", array1)
print("Array2:", array2)
print("Common values between two arrays:")
print(common_values)
```

OUPUT:

```
Array1: [ 0 10 20 40 60]
Array2: [10 30 40]
Common values between two arrays:
[10 40]
```

13. Write a NumPy program to add a border (filled with 0's) around an existing array.

```
import numpy as np

# Create the original array
original_array = np.array([[1, 1, 1], [1, 1, 1], [1, 1, 1]])

# Add a border filled with 0's
padded_array = np.pad(original_array, pad_width=1, mode='constant', constant_values=0)
print("Original array:")
print(original_array)
print("Array with 0's border:")
print(padded_array)
```

OUTPUT:

```
Original array:
[[1 1 1]
 [1 1 1]
 [1 1 1]]
Array with 0's border:
[[0 0 0 0 0]
 [0 1 1 1 0]
 [0 1 1 1 0]
 [0 1 1 1 0]
 [0 0 0 0 0]]
```

14. Write a NumPy program to sort a given array of shape 2 along the first axis, last axis and on flattened array.

```
import numpy as np

# Create the original array
original_array = np.array([[12, 40], [70, 30]])
print("Original array:")
print(original_array)

# Sort along the first axis
sorted_first_axis = np.sort(original_array, axis=0)
print("\nSorted along the first axis:")
print(sorted_first_axis)

# Sort along the last axis
sorted_last_axis = np.sort(original_array, axis=1)
print("\nSorted along the last axis:")
print(sorted_last_axis)

# Sort the flattened array
sorted_flattened = np.sort(original_array.flatten())
print("\nSorted flattened array:")
```

```
print(sorted_flattened)
```

OUTPUT:

```
Original array:
[[12 40]
 [70 30]]

Sorted along the first axis:
[[12 30]
 [70 40]]

Sorted along the last axis:
[[12 40]
 [30 70]]

Sorted flattened array:
[12 30 40 70]
```

15. Write a NumPy program to create an element-wise comparison (greater, greater_equal, less and less_equal) of two given arrays.

```
import numpy as np

# Create two example arrays
array1 = np.array([1, 2, 3, 4, 5])
array2 = np.array([3, 2, 5, 1, 6])

# Element-wise greater comparison
greater_comparison = np.greater(array1, array2)
print("Greater Comparison:")
print(greater_comparison)

# Element-wise greater_equal comparison
greater_equal_comparison = np.greater_equal(array1, array2)
print("Greater Equal Comparison:")
print(greater_equal_comparison)

# Element-wise less comparison
less_comparison = np.less(array1, array2)
print("Less Comparison:")
print(less_comparison)

# Element-wise less_equal comparison
less_equal_comparison = np.less_equal(array1, array2)
print("Less Equal Comparison:")
print(less_equal_comparison)
```

OUTPUT:

```
Greater Comparison:
[False False False  True False]
Greater Equal Comparison:
[False  True False  True False]
Less Comparison:
[ True False  True False  True]
Less Equal Comparison:
[ True  True  True False  True]
```

16. Write a python program to Check whether a Numpy array contains a specified row

```
import numpy as np

def contains_row(arr, row):
    for r in arr:
        if np.array_equal(r, row):
            return True
    return False

# Example usage
array = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
specified_row = np.array([4, 5, 6])

if contains_row(array, specified_row):
    print("The specified row is present in the array.")
else:
    print("The specified row is not present in the array.")
```

OUTPUT:

```
The specified row is present in the array.
```

17. Write python program to count of occurrence of each element in array and display maximum occurrence count.

```
def count_occurrence(arr):
    occurrence_count = {}
    max_count = 0

    for element in arr:
        if element in occurrence_count:
            occurrence_count[element] += 1
        else:
            occurrence_count[element] = 1

        if occurrence_count[element] > max_count:
            max_count = occurrence_count[element]

    return occurrence_count, max_count
```

```
# Example usage:
array = [1, 2, 3, 4, 2, 3, 1, 2, 4, 4, 4]
occurrence_count, max_count = count_occurrence(array)

print("Occurrence count of each element:")
for element, count in occurrence_count.items():
    print(f"{element}: {count}")

print("Maximum occurrence count:", max_count)
```

OUTPUT:

```
Occurrence count of each element:
1: 2
2: 3
3: 2
4: 4
Maximum occurrence count: 4
```