Open Destination Coding Test

1. **Table Definition and queries**
   1. **Provide details like table name, columns/fields, data type, constraint, indexes etc. You can make your own assumptions while creating the above database structure**.
      1. **Brand**

CREATE TABLE [dbo].[Brand](

[Id] [int] IDENTITY(1,1) NOT NULL,

[name] [varchar](128) NOT NULL,

[createdon] [datetime] NULL,

[createdby] [varchar](128) NULL,

[updatedon] [datetime] NULL,

[updatedby] [varchar](128) NULL,

[isactive] [bit] NULL,

CONSTRAINT [PK\_Brand] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

* + 1. **Products**

CREATE TABLE [dbo].[Products](

[Id] [int] IDENTITY(1,1) NOT NULL,

[name] [varchar](100) NOT NULL,

[unitprice] [decimal](8, 0) NOT NULL,

[companyid] [int] NOT NULL,

[createdon] [datetime] NULL,

[createdby] [varchar](128) NULL,

[updatedon] [datetime] NULL,

[updatedby] [varchar](128) NULL,

[isactive] [bit] NULL,

CONSTRAINT [PK\_Products] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Products] WITH CHECK ADD CONSTRAINT [FK\_Products\_Brand] FOREIGN KEY([companyid])

REFERENCES [dbo].[Brand] ([Id])

GO

ALTER TABLE [dbo].[Products] CHECK CONSTRAINT [FK\_Products\_Brand]

GO

* + 1. **ProductSales**

CREATE TABLE [dbo].[ProductSales](

[Id] [int] IDENTITY(1,1) NOT NULL,

[productid] [int] NOT NULL,

[qtysold] [int] NOT NULL,

[unitprice] [decimal](8, 2) NOT NULL,

[createdon] [datetime] NULL,

[createdby] [varchar](128) NULL,

[updatedon] [datetime] NULL,

[updatedby] [varchar](128) NULL,

[isactive] [bit] NULL,

CONSTRAINT [PK\_ProductSales] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[ProductSales] WITH CHECK ADD CONSTRAINT [FK\_ProductSales\_Products] FOREIGN KEY([productid])

REFERENCES [dbo].[Products] ([Id])

GO

ALTER TABLE [dbo].[ProductSales] CHECK CONSTRAINT [FK\_ProductSales\_Products]

GO

* + 1. **ProductRating**

CREATE TABLE [dbo].[productrating](

[Id] [int] IDENTITY(1,1) NOT NULL,

[rate] [int] NOT NULL,

[productid] [int] NOT NULL,

[createdon] [datetime] NOT NULL,

[createdby] [varchar](128) NOT NULL,

[isactive] [bit] NULL,

CONSTRAINT [PK\_productrating] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[productrating] WITH CHECK ADD CONSTRAINT [FK\_productrating\_Products] FOREIGN KEY([productid])

REFERENCES [dbo].[Products] ([Id])

GO

ALTER TABLE [dbo].[productrating] CHECK CONSTRAINT [FK\_productrating\_Products]

GO

ALTER TABLE [dbo].[productrating] WITH CHECK ADD CONSTRAINT [chk\_Ratings] CHECK (([rate]>=(0) AND [rate]<=(5)))

GO

* 1. **Write Query to return all products (1 row per product) with all product details like Name, Company name, rating and total sales for last 1 year**

select p.[name] as productname,b.[name] as brandname,sum(qtysold\*ps.unitprice) as totalSales,p.unitprice,sum(ps.qtysold) as qtysold

from dbo.Products p

join dbo.Brand b on p.companyid = b.Id

join dbo.ProductSales ps on p.Id = ps.productid

left join (

select ROUND(AVG(CAST(rate AS FLOAT)), 2) as rate,productid from dbo.productrating group by productid

) pr on pr.productid = p.id

WHERE ps.createdon > DATEADD(year,-1,GETDATE())

group by p.Id,p.[name],b.[name],p.unitprice;

* 1. **List the products along with price with rating of 3 and 4 stars with product with cheapest price displayed first.**

select p.[name],ROUND(AVG(CAST(pr.rate AS FLOAT)), 2) as rate,p.unitprice

from dbo.Products p

join dbo.productrating pr on p.id = pr.productid

group by p.Id,p.[name],p.unitprice

having ROUND(AVG(CAST(pr.rate AS FLOAT)), 2) between 3 and 4

order by p.unitprice;

1. **Search UI and Flow**
   1. **UI**
      1. **Mention if any Ajax features can be used**

I have used http ajax post request

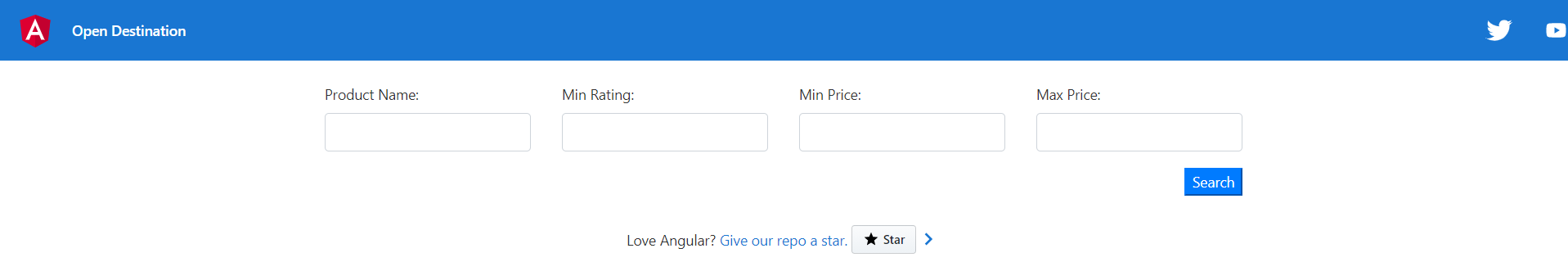
* + 1. **Mention how paging and sorting will be handled.**

Sorting I have done. Pagination I have not implemented

-Sorting: on click of header user can sort ascending and desc accordingly

-Pagination: We can pass skip size and take size number to fetch limited data to avoid load whole records

* + 1. **Mention which controls you will use on UI for the filters and for displaying results**

**-** Product Name: Input type text  
- Min Rating: Input type number  
- Min Price: Input type number  
- Max Price: Input type number  
  


* 1. **Mention flow when Search button is clicked from UI to Business and Data layer.**  
     - When user click on search it will collect data from filter form and call appservice.getProducts() service and pass data in a json format using post request  
     - Client will hit backend web api name called <https://localhost:44385/api/product>- Api will call injected repository ProdRep.SearchProduct() and based on parameter using dbcontaxt it will fetch records and return it to controller.  
     - Controller will return data to client.  
     - Client angular will subscribe search service and get data and it will bind to products object
  2. **Mention the query which you will write for Search considering all the parameters are optional. If user does not select any filters, it should show all records. If user selects 1, 2, or all 3 filters then query should filter accordingly and support all combinations.**  
     Instead of stored procedure I have used entity framework code first approach.  
     If I would write SP, it would be similar to above query which I have mentioned, except sorting and pagination.  
     Soring and pagination we can use as bellow.  
     Pagination and sorting: We can use common table expression query and apply order by with dynamic red query and pass row number between skip size and takesize+skipsize
  3. **Mention how you will ensure performance is quick and results return within 2 seconds for large database.**

Usually, we will use pagination so it will allow to fetch limited records to display on the web page. Still in certain case query is taking more time to return data we can track in sql profiler to ensure where query is taking more time and diagnostic.

I have Pushed my test to github you can access using below mentioned link.  
  
1) client : <https://github.com/ajaykvyas96/OpenDestClient>2) webapi: <https://github.com/ajaykvyas96/OpenDestination> (sql changes also added in the folder SqlChnages)