

# IST769 Homework Submission

Name: **Sathish Kumar Rajendiran**

SUID: **666555028**

Email: [srajendi@syr.edu](mailto:srajendi@syr.edu)

Date Due: **08/03/2021**

Date Due: **Performance, Security, NoSQL**

Homework #:**4**

## Exercise(s):

1. Create a **non-clustered index** on the timesheets table in the demo database. The index you create should be designed to improve the following query:

```
select employee_id, employee_firstname, employee_lastname,
sum(timesheet_hourlyrate*timesheet_hours)
from timesheets
group by employee_id, employee_firstname, employee_lastname
```

## Solution:

```
----Pre-setup command
----Use [fudgemart_v3]
----Go
----select * into demo.dbo.timesheets
----from fudgemart_employees join fudgemart_employee_timesheets
----on employee_id = timesheet_employee_id
----Go

USE [demo]
GO

SET STATISTICS TIME ON
SET STATISTICS IO ON
GO

--before Index creation
declare @startTime datetime2
declare @endTime datetime2
set @startTime = getdate()
select employee_id, employee_firstname, employee_lastname,
sum(timesheet_hourlyrate*timesheet_hours) as wages
from timesheets
group by employee_id, employee_firstname, employee_lastname
set @endTime = getdate()
---total execution time in milliseconds
```

```

select datediff(millisecond,@startTime,@endTime) as execTime
GO

SET STATISTICS TIME OFF
SET STATISTICS IO OFF
GO

---- Find an existing index named IX_Timesheets_EmployeeID_Name_Hours_Rate and delete
it if found.
IF EXISTS (SELECT name FROM sys.indexes
           WHERE name = N'IX_Timesheets_EmployeeID_Name_Hours_Rate')
DROP INDEX IX_Timesheets_EmployeeID_Name_Hours_Rate ON dbo.timesheets;
GO

-- Create a nonclustered index called IX_Timesheets_EmployeeID_Name_Hours_Rate
-- on the dbo.timesheets table using the employee_id,
employee_firstname,employee_lastname,timesheet_hourlyrate,timesheet_hours columns.
CREATE NONCLUSTERED INDEX IX_Timesheets_EmployeeID_Name_Hours_Rate
ON dbo.timesheets (employee_id)
INCLUDE (employee_firstname, employee_lastname,timesheet_hourlyrate,timesheet_hours)
GO

SET STATISTICS TIME ON
SET STATISTICS IO ON
GO

--after Index creation
declare @startTime datetime2
declare @endTime datetime2
set @startTime = getdate()
select employee_id, employee_firstname, employee_lastname,
sum(timesheet_hourlyrate*timesheet_hours) as wages
from timesheets
group by employee_id, employee_firstname, employee_lastname
set @endTime = getdate()
---total execution time in milliseconds
select datediff(millisecond,@startTime,@endTime) as execTime
GO

SET STATISTICS TIME OFF
SET STATISTICS IO OFF
GO

```

**Evidence:**

Connect ➔

localhost (SQL Server 14.0.3401.7 - sa)

- Databases
  - System Databases
  - Database Snapshots
  - demo**
- demo**
  - Database Diagrams
  - Tables
    - System Tables
    - FileTables
    - External Tables
    - Graph Tables
  - dbo.DB\_Errors
  - dbo.players
  - dbo.players\_history
  - dbo.shots
  - dbo.timesheets**
    - Columns
      - employee\_id (int, not null)
      - employee\_ssn (char(9), not null)
      - employee\_lastname (varchar(50), not null)
      - employee\_firstname (varchar(50), not null)
      - employee\_jobtitle (varchar(20), not null)
      - employee\_department (varchar(20), not null)
      - employee\_birthdate (datetime, not null)

Results Messages Execution plan

	employee_id	employee_firstname	employee_lastname	wages
1	1	Anal	Photo	32660.06300
2	2	Sal	Ladd	121135.46800
3	3	Dustin	Dawind	7831.05000
4	4	Sandi	Shores	79714.26800
5	5	Isabelle	Gunnering	161799.23200
6	6	Lee	Hmeehom	149386.36400
7	7	Allan	Wrench	97164.93600
8	8	Aly	Gator	99305.22000
9	9	Alma	Frenzergon	94169.24000
10	10	Artie	Choke	102301.74400
		execTime		
1		16		

Query executed successfully.

Results Messages Execution plan

```
(33 rows affected)
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'timesheets'. Scan count 1, logical reads 108, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.

(1 row affected)

SQL Server Execution Times:
  CPU time = 16 ms,  elapsed time = 16 ms.
```

194 % 4

Query executed successfully.

localhost (SQL Server 14.0.3401.7 - sa)

- Databases
  - System Databases
  - Database Snapshots
  - demo
    - Database Diagrams
    - Tables
      - System Tables
      - FileTables
      - External Tables
      - Graph Tables
      - dbo.DB\_Errors
      - dbo.players
      - dbo.players\_history
      - dbo.shots
      - dbo.timesheets
        - Columns
        - Keys
        - Constraints
        - Triggers
        - Indexes
          - IX\_Timesheets\_EmployeeID\_Name\_Hours\_Rate (Non-Unique, Non-Clustered)
      - Statistics

Results Messages Execution plan

	employee_id	employee_firstname	employee_lastname	wages
1	1	Anal	Photo	32660.06300
2	2	Sal	Ladd	121135.46800
3	3	Dustin	Dawind	7831.05000
4	4	Sandi	Shores	79714.26800
5	5	Isabelle	Gunnering	161799.23200
6	6	Lee	Hvmeehom	149386.36400
7	7	Allan	Wrench	97164.93600
8	8	Ally	Gator	99305.22000
9	9	Alma	Frienzergon	94169.24000
10	10	Artie	Choke	102301.74400

execTime  
1 14

Index Properties - IX\_Timesheets\_EmployeeID\_Name\_Hours\_Rate

Ready

Select a page: General, Options, Storage, Filter, Fragmentation, Extended Properties

Table name: timesheets

Index name: IX\_Timesheets\_EmployeeID\_Name\_Hours\_Rate

Index type: Nonclustered

Unique

Index key columns: employee\_id, employee\_firstname, employee\_lastname, timesheet\_hourlyrate, timesheet\_hours

Included columns:

Name	Data Type	Identity	Allow NULLS
employee_firstname	varchar(50)	No	No
employee_lastname	varchar(50)	No	No
timesheet_hourlyrate	money	No	No
timesheet_hours	decimal(3,1)	No	No

Connection: localhost [sa]

[View connection properties](#)

```

Results Messages Execution plan
(33 rows affected)
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'timesheets'. Scan count 1, logical reads 40, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.

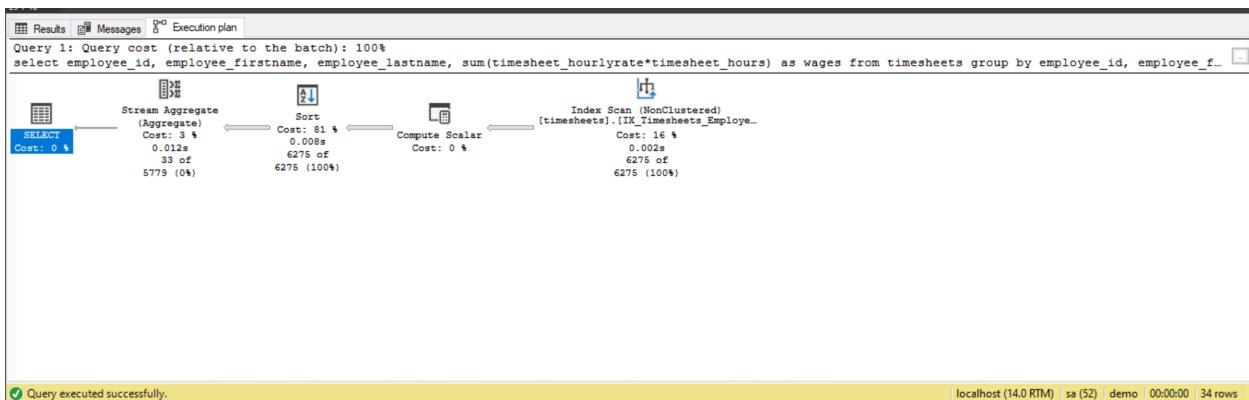
(1 row affected)

SQL Server Execution Times:
    CPU time = 13 ms, elapsed time = 14 ms.

SQL Server Execution Times:
    CPU time = 0 ms, elapsed time = 0 ms.

194 %
Query executed successfully.
localhost (14.0 RTM) | sa (52) | demo | 00:00:00 | 34 rows

```



2. Write an SQL Select query which uses the index you created in the first question but does an index seek instead of an index scan.

**Solution:**

```

SET STATISTICS TIME ON
SET STATISTICS IO ON
GO

--after Index creation
declare @startTime datetime2
declare @endTime datetime2
--declare @maxemployeeID int
set @startTime = getdate()
--select @maxemployeeID = max(employee_id) from dbo.timesheets
select employee_id, employee_firstname, employee_lastname,
sum(timesheet_hourlyrate*timesheet_hours) as wages
from timesheets
where employee_id < 50
--where employee_id < ( select max(employee_id) from dbo.timesheets)
--where employee_id < @maxemployeeID
group by employee_id, employee_firstname, employee_lastname

```

```

set @endTime = getdate()
---total execution time in milliseconds
select datediff(millisecond,@startTime,@endTime) as execTime
GO

SET STATISTICS TIME OFF
SET STATISTICS IO OFF
GO

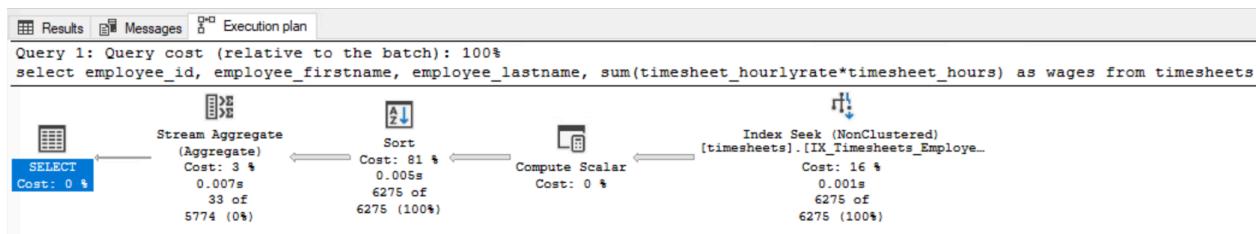
```

**Evidence:**

Execution plan				
	employee_id	employee_firstname	employee_lastname	wages
23	23	Nat	Tural	32807.29590
24	24	Otto	Moni	149814.17200
25	25	Patty	O'Furniture	93312.79600
26	26	Pete	Moss	34403.01540
27	27	Sara	Docktur-Indahaus	164367.94800
28	28	Sara	Isnomor	71287.37200
29	29	Seymour	Ofewe	115571.05600
30	30	Sonny	Shores	110862.68000
31	31	Tally	Itupp	184057.28800
32	32	Tuck	Androll	243983.41200
33	33	Mike	Fudge	589412.12400

execTime
10

Execution plan		
(33 rows affected)	Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob logical : Table 'timesheets'. Scan count 1, logical reads 40, physical reads 0, read-ahead reads 0, lob logical : (1 row affected)	SQL Server Execution Times: CPU time = 8 ms, elapsed time = 9 ms.



3. Create a single columnstore index on the timesheets table in the demo database which will improve the following queries:

```
select employee_department, sum(timesheet_hours)
from timesheets group by employee_department

select employee_jobtitle, avg(timesheet_hourlyrate)
from timesheets group by employee_jobtitle
```

**Solution:**

```
/*
SET STATISTICS TIME ON
SET STATISTICS IO ON
GO

--before Index creation
declare @startTime datetime2
declare @endTime datetime2
set @startTime = getdate()

select employee_department
, sum(timesheet_hours) agg_timesheet_hours
from timesheets
group by employee_department
set @endTime = getdate()

---total execution time in milliseconds
select datediff(millisecond,@startTime,@endTime) as execTime
GO

SET STATISTICS TIME OFF
SET STATISTICS IO OFF
GO

---- Find an existing index named IX_CLM_Timesheets_Employee_Department_Hours
---- and delete it if found.
IF EXISTS (SELECT name FROM sys.indexes
           WHERE name = N'IX_CLM_Timesheets')
DROP INDEX IX_CLM_Timesheets ON dbo.timesheets;
GO
-- Create a nonclustered COLUMNSTORE index called IX_CLM_Timesheets
-- on the dbo.timesheets table using the employee_id,
employee_department,timesheet_hours,timesheet_hourlyrate,timesheet_hours columns.

CREATE NONCLUSTERED COLUMNSTORE INDEX IX_CLM_Timesheets
ON dbo.timesheets(employee_department,timesheet_hours)
GO
```

```

SET STATISTICS TIME ON
SET STATISTICS IO ON
GO

--after Index creation
declare @startTime datetime2
declare @endTime datetime2
set @startTime = getdate()

select employee_department
, sum(timesheet_hours) agg_timesheet_hours
from timesheets
group by employee_department
set @endTime = getdate()

---total execution time in milliseconds
select datediff(millisecond,@startTime,@endTime) as execTime
GO

SET STATISTICS TIME OFF
SET STATISTICS IO OFF
GO

/***********************/

SET STATISTICS TIME ON
SET STATISTICS IO ON
GO

--before Index creation
declare @startTime datetime2
declare @endTime datetime2
set @startTime = getdate()

select employee_jobtitle
, avg(timesheet_hourlyrate) avg_hourlyrate
from timesheets
group by employee_jobtitle
set @endTime = getdate()

---total execution time in milliseconds
select datediff(millisecond,@startTime,@endTime) as execTime
GO

SET STATISTICS TIME OFF
SET STATISTICS IO OFF

```

```

GO

IF EXISTS (SELECT name FROM sys.indexes
           WHERE name = N'IX_CLM_Timesheets')
DROP INDEX IX_CLM_Timesheets ON dbo.timesheets;
GO
-- Create a nonclustered COLUMNSTORE index called IX_CLM_Timesheets
-- on the dbo.timesheets table using the employee_id,
employee_department,timesheet_hours,timesheet_hourlyrate,timesheet_hours columns.

CREATE NONCLUSTERED COLUMNSTORE INDEX IX_CLM_Timesheets
ON
dbo.timesheets(employee_department,timesheet_hours,employee_jobtitle,timesheet_hourlyr
ate)
GO

SET STATISTICS TIME ON
SET STATISTICS IO ON
GO

--after Index creation
declare @startTime datetime2
declare @endTime datetime2
set @startTime = getdate()

select employee_jobtitle
, avg(timesheet_hourlyrate) avg_hourlyrate
from timesheets
group by employee_jobtitle
set @endTime = getdate()

---total execution time in milliseconds
select datediff(millisecond,@startTime,@endTime) as execTime
GO

SET STATISTICS TIME OFF
SET STATISTICS IO OFF
GO

```

**Evidence:**

Results    Messages    Execution plan

	employee_department	agg_timesheet_hours
1	Hardware	31829.0
2	Sporting Goods	36455.0
3	Housewares	36618.0
4	Customer Service	41600.0
5	Electronics	35458.0
6	Clothing	30996.0

	execTime
1	4

(6 rows affected)

Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.

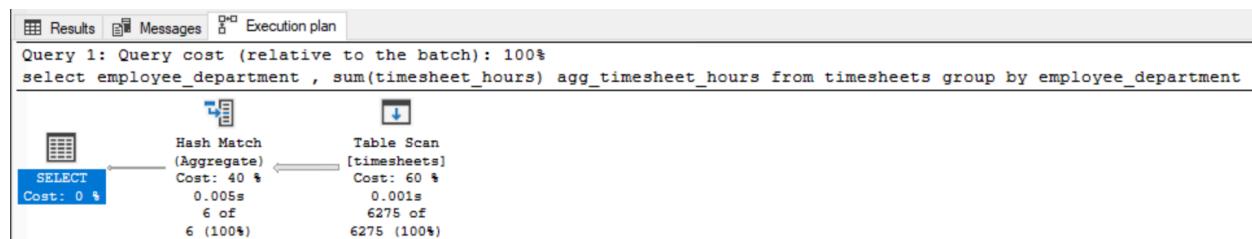
Table 'Workfile'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.

Table 'timesheets'. Scan count 1, logical reads 108, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.

(1 row affected)

SQL Server Execution Times:

CPU time = 5 ms, elapsed time = 5 ms.



localhost (SQL Server 14.0.3401.7 - sa)

- Databases
  - System Databases
  - Database Snapshots
- demo
  - Database Diagrams
  - Tables
    - System Tables
    - FileTables
    - External Tables
    - Graph Tables
    - dbo.DB\_Errors
    - dbo.players
    - dbo.players\_history
    - dbo.shots
    - dbo.timesheets
      - Columns
      - Keys
      - Constraints
      - Triggers
      - Indexes
        - IX\_CLM\_Timesheets (Non-Clustered, Columnstore)
        - IX\_Timesheets\_EmployeeID\_Name\_Hours\_Rate (Non-Unique, Non-Clustered)

Results    Messages    Execution plan

	employee_department	agg_timesheet_hours
1	Housewares	36618.0
2	Sporting Goods	36455.0
3	Clothing	30996.0
4	Customer Service	41600.0
5	Hardware	31829.0
6	Electronics	35458.0

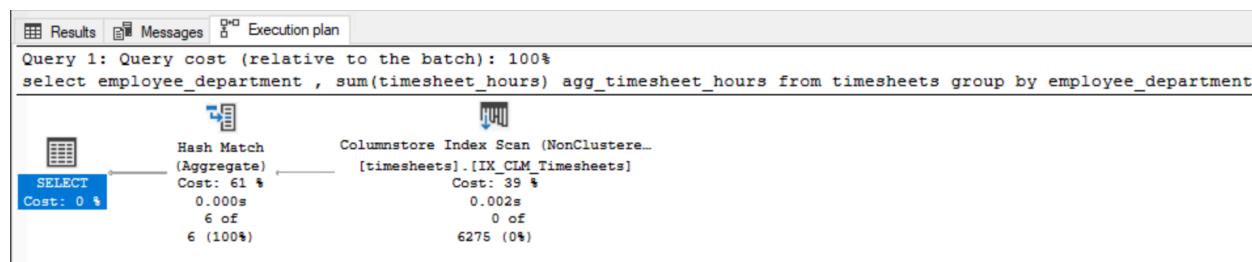
	execTime
1	0

```
(6 rows affected)
Table 'timesheets'. Scan count 2, logical reads 0, physical reads 0, read-ahead reads 0, lob logical
Table 'timesheets'. Segment reads 1, segment skipped 0.
```

```
(1 row affected)
```

SQL Server Execution Times:

CPU time = 2 ms, elapsed time = 1 ms.



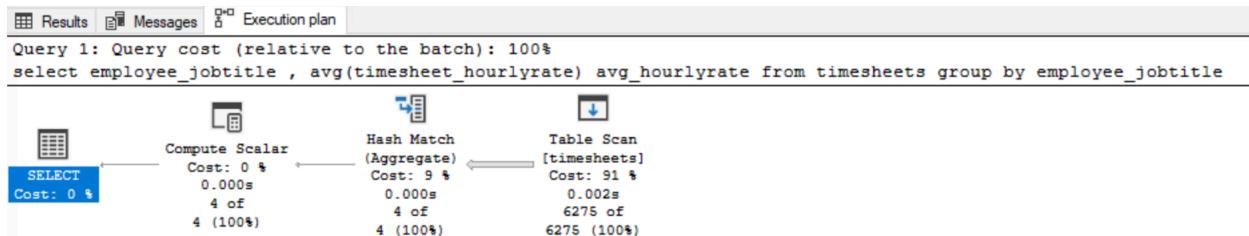
	employee_jobtitle	avg_hourlyrate
1	Department Manager	19.567
2	Store Manager	29.3249
3	CEO	70.8428
4	Sales Associate	11.8918

✓ Query executed successfully.

```
SQL Server Execution Times:
CPU time = 0 ms, elapsed time = 0 ms.
SQL Server parse and compile time:
CPU time = 3 ms, elapsed time = 3 ms.
```

```
SQL Server Execution Times:
CPU time = 0 ms, elapsed time = 0 ms.
```

```
(4 rows affected)
Table 'timesheets'. Scan count 1, logical reads 108, physical reads 0, read-ahead reads 0, lob logical
```



Index Properties - IX\_CLM\_Timesheets

Ready

Select a page Script Help

Table name: timesheets

Index name: IX\_CLM\_Timesheets

Index type: Nonclustered columnstore

Columnstore columns

Name	Data Type	Identity	Allow NULLs
employee_department	varchar(20)	No	No
timesheet_hours	decimal(3,1)	No	No
employee_jobtitle	varchar(20)	No	No
timesheet_hourlyrate	money	No	No

Connection

localhost [sa]

Results Messages Execution plan

	employee_jobtitle	avg_hourlyrate
1	Department Manager	19.567
2	Store Manager	29.3249
3	CEO	70.8428
4	Sales Associate	11.8918

	execTime
1	0

SQL Server Execution Times:

CPU time = 0 ms, elapsed time = 0 ms.

SQL Server parse and compile time:

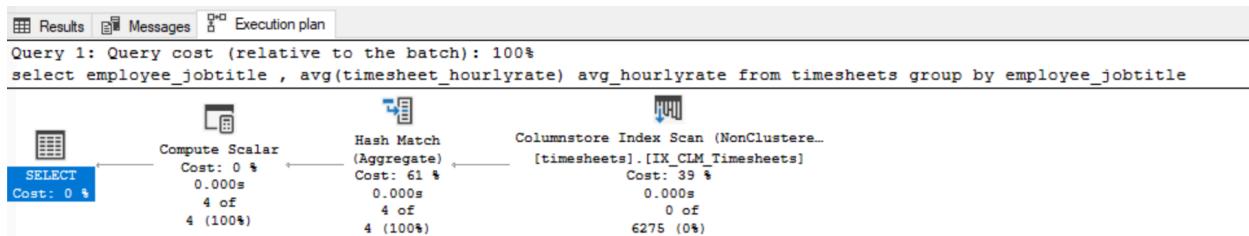
CPU time = 0 ms, elapsed time = 0 ms.

SQL Server Execution Times:

CPU time = 0 ms, elapsed time = 0 ms.

(4 rows affected)

Table 'timesheets'. Scan count 2, logical reads 0, physical reads 0, read-ahead reads 0, lob logical  
Table 'timesheets'. Segment reads 1, segment skipped 0.



4. Create an indexed view named `v_employees` on the `timesheets` table in the demo database which lists the employee id, first name, last name, job title, and department columns values and one row per employee (essentially re-building the employee table). Then set a unique clustered index on the view and finish by writing an SQL Select query which uses the indexed view.

**Solution:**

```

IF EXISTS (SELECT * FROM INFORMATION_SCHEMA.VIEWS WHERE TABLE_NAME='v_employees')
BEGIN
DROP VIEW v_employees
END
GO

CREATE VIEW v_employees WITH schemabinding
AS
SELECT employee_id
, employee_firstname
, employee_lastname
, employee_jobtitle
, employee_department
, COUNT_BIG(*) as employee_count
from dbo.timesheets
Group by
employee_id
, employee_firstname
, employee_lastname
, employee_jobtitle
, employee_department
GO

--before index creation
SELECT * FROM dbo.v_employees
GO

IF EXISTS (SELECT name FROM sys.indexes
      WHERE name ='IX_CLS_v_employees_EmployeeID')
DROP INDEX IX_CLS_v_employees_EmployeeID ON dbo.v_employees;
GO

```

```
-- Create an unique clustered index called IX_CLS_v_employees_EmployeeID  
-- on the dbo.v_employees table using the employee_id
```

```
CREATE UNIQUE CLUSTERED INDEX IX_CLS_v_employees_EmployeeID  
ON dbo.v_employees(employee_id)  
GO  
--query from the view after index creation  
SELECT * FROM dbo.v_employees  
GO
```

### Evidence:

	employee_id	employee_firstname	employee_lastname	employee_jobtitle	employee_department	employee_count
1	5	Isabelle	Gunnering	Department Manager	Electronics	208
2	12	Bill	Melator	Sales Associate	Sporting Goods	54
3	22	Mike	Rophone	Sales Associate	Housewares	208
4	1	Arial	Photo	Sales Associate	Electronics	129
5	18	Justin	Case	Sales Associate	Clothing	208
6	25	Patty	O'Furniture	Sales Associate	Housewares	208
7	8	Ally	Gator	Sales Associate	Sporting Goods	208
8	17	Kurt	Tan	Sales Associate	Clothing	208
9	29	Seymour	Ofewe	Sales Associate	Customer Service	208
10	30	Sonny	Shores	Sales Associate	Customer Service	208
11	28	Sara	Isnominator	Sales Associate	Clothing	192
12	11	Bette	Alott	Sales Associate	Sporting Goods	208
13	16	Detyers	Erin	Sales Associate	Hardware	208
14	23	Nat	Tural	Sales Associate	Sporting Goods	171
15	2	Sal	Ladd	Sales Associate	Electronics	208
16	20	Mary	Mi	Department Manager	Clothing	208
17	9	Alma	Frienzergon	Sales Associate	Housewares	208
18	14	Chris P.	Nugget	Sales Associate	Electronics	208

Query executed successfully.

```
Query 1: Query cost (relative to the batch) : 100%  
SELECT * FROM dbo.v_employees
```

Table Scan  
[timesheets]  
Cost: 100 %  
0.013s  
6275 of  
6275 (100%)

Connect ▾

- ⊕ Keys
- ⊕ Constraints
- ⊕ Triggers
- ⊖ Indexes
  - [nl] IX\_CLM\_Timesheets (Non-Clustered, Columnstore)
  - ⊖ IX\_Timesheets\_EmployeeID\_Name\_Hours\_Rate (Non-Unique, Non-Clustered)
- ⊕ Statistics

- ⊖ Views
  - ⊕ System Views
  - ⊖ dbo.v\_employees
    - ⊕ Columns
    - ⊕ Triggers
    - ⊖ Indexes
      - ⊖ IX\_CLS\_v\_employees\_EmployeeID (Clustered)
  - ⊕ Statistics

```

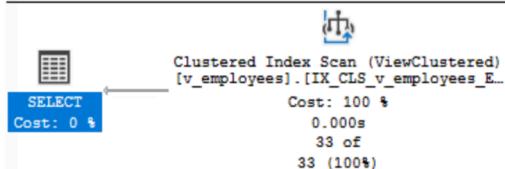
293 --query from the view after index creation
294 SELECT * FROM dbo.v_employees
295 GO
296
297 ****

```

194 % Results Messages Execution plan

Query 1: Query cost (relative to the batch): 100%

SELECT \* FROM dbo.v\_employees



	employee_id	employee_firstname	employee_lastname	employee_jobtitle	employee_department	employee_count
1	1	Arial	Photo	Sales Associate	Electronics	129
2	2	Sal	Ladd	Sales Associate	Electronics	208
3	3	Dustin	Dawind	Sales Associate	Hardware	44
4	4	Sandi	Shores	Sales Associate	Hardware	156
5	5	Isabelle	Gunnering	Department Manager	Electronics	208
6	6	Lee	Hvmeehom	Department Manager	Hardware	208
7	7	Allan	Wrench	Sales Associate	Housewares	208
8	8	Aly	Gator	Sales Associate	Sporting Goods	208
9	9	Alma	Frienzergon	Sales Associate	Housewares	208
10	10	Artie	Choke	Sales Associate	Hardware	208
11	11	Bette	Altott	Sales Associate	Sporting Goods	208
12	12	Bill	Melator	Sales Associate	Sporting Goods	54
13	13	Bob	Ernweave	Sales Associate	Sporting Goods	121
14	14	Chris P.	Nugget	Sales Associate	Electronics	208
15	15	Chuck	Iupp	Sales Associate	Sporting Goods	208
16	16	Detyers	Erin	Sales Associate	Hardware	208
17	17	Kurt	Tan	Sales Associate	Clothing	208
18	18	Justin	Case	Sales Associate	Clothing	208

Query executed successfully.

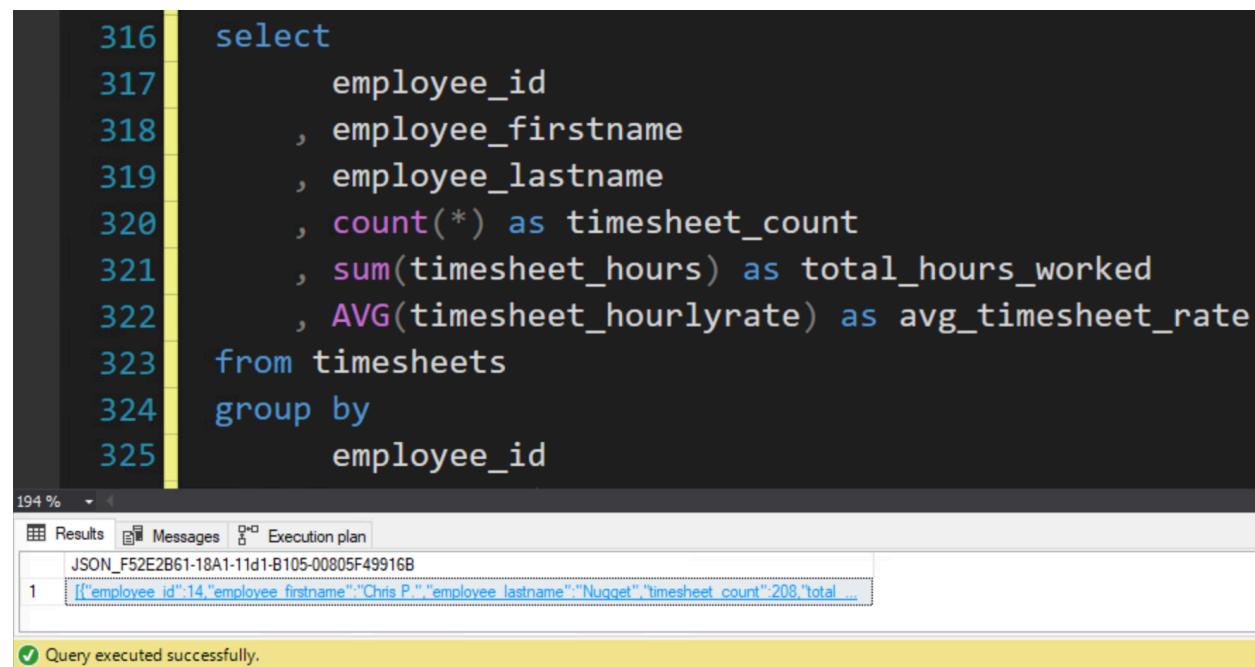
localhost (14.0 RTM) | sa (52) | demo | 00:00:00 | 33 rows

5. Output the following query in **JSON** format: Display the employee id, first name, last name, count of timesheets, total hours worked, and average timesheet hourly rate.

**Solution:**

```
--SQL statement with NOSQL Output
select
    employee_id
    , employee_firstname , employee_lastname
    , count(*) as timesheet_count
    , sum(timesheet_hours) as total_hours_worked
    , AVG(timesheet_hourlyrate) as avg_timesheet_rate
from timesheets
group by
    employee_id
    , employee_firstname
    , employee_lastname
FOR JSON AUTO
GO
```

**Evidence:**



The screenshot shows a SQL query results window with the following details:

- Query Text:** The query is identical to the one provided in the solution, selecting employee\_id, employee\_firstname, employee\_lastname, count(\*), sum(timesheet\_hours), and AVG(timesheet\_hourlyrate) from timesheets, grouped by employee\_id, and outputting as JSON.
- Execution Plan:** The plan is labeled "JSON\_F52E2B61-18A1-11d1-B105-00805F49916B".
- Results:** The result set contains one row of data:

```
[{"employee_id":14,"employee_firstname":"Chris P","employee_lastname":"Nugget","timesheet_count":208,"total_hours_worked":166.5,"avg_timesheet_rate":8.0625}
```
- Status Bar:** A yellow bar at the bottom indicates "Query executed successfully."

Results | Messages | Execution plan

	employee_id	employee_firstname	employee_lastname	timesheet_count	total_hours_worked	avg_timesheet_rate
1	14	Chris P.	Nugget	208	8320.0	15.1769
2	22	Mike	Rophone	208	3338.0	11.3184
3	24	Otto	Moni	208	8320.0	18.0065
4	30	Sonny	Shores	208	8320.0	13.3248
5	1	Arial	Photo	129	2178.0	14.9883
6	3	Dustin	Dawind	44	629.0	12.45
7	12	Bill	Melator	54	2160.0	10.2539
8	2	Sei	Ladd	208	8320.0	14.5505

✓ Query executed successfully.

```

324      group by
325          employee_id
326          , employee_firstname
327          , employee_lastname
328      FOR JSON AUTO
329      GO
330
331

```

194 %

Results | Messages | Execution plan

	JSON_F52E2B61-18A1-11d1-B105-00805F49916B
1	[{"employee_id":14,"employee_firstname":"Chris P.",...

[JSON formatter]

JSON PARSER   XML FORMATTER   JSBEAUTIFIER   SAVE   RECENT LINKS   LOGIN

Upload Data

Validate

2 Tab Space

Format / Beautify

study in style  Ashley HomeStore

Minify / Compact

Code

```

1 [
2   {
3     "employee_id": 14,
4     "employee_firstname": "Chris P.",
5     "employee_lastname": "Nugget",
6     "timesheet_count": 208,
7     "total_hours_worked": 8320,
8     "avg_timesheet_rate": 15.1769
9   },
10  [
11    {
12      "employee_id": 22,
13      "employee_firstname": "Mike",
14      "employee_lastname": "Rophone",
15      "timesheet_count": 208,
16      "total_hours_worked": 3338,
17      "avg_timesheet_rate": 11.3184
18    },
19    {
20      "employee_id": 24,
21      "employee_firstname": "Otto",
22      "employee_lastname": "Moni",
23      "timesheet_count": 208,
24    }
25

```

## Appendix

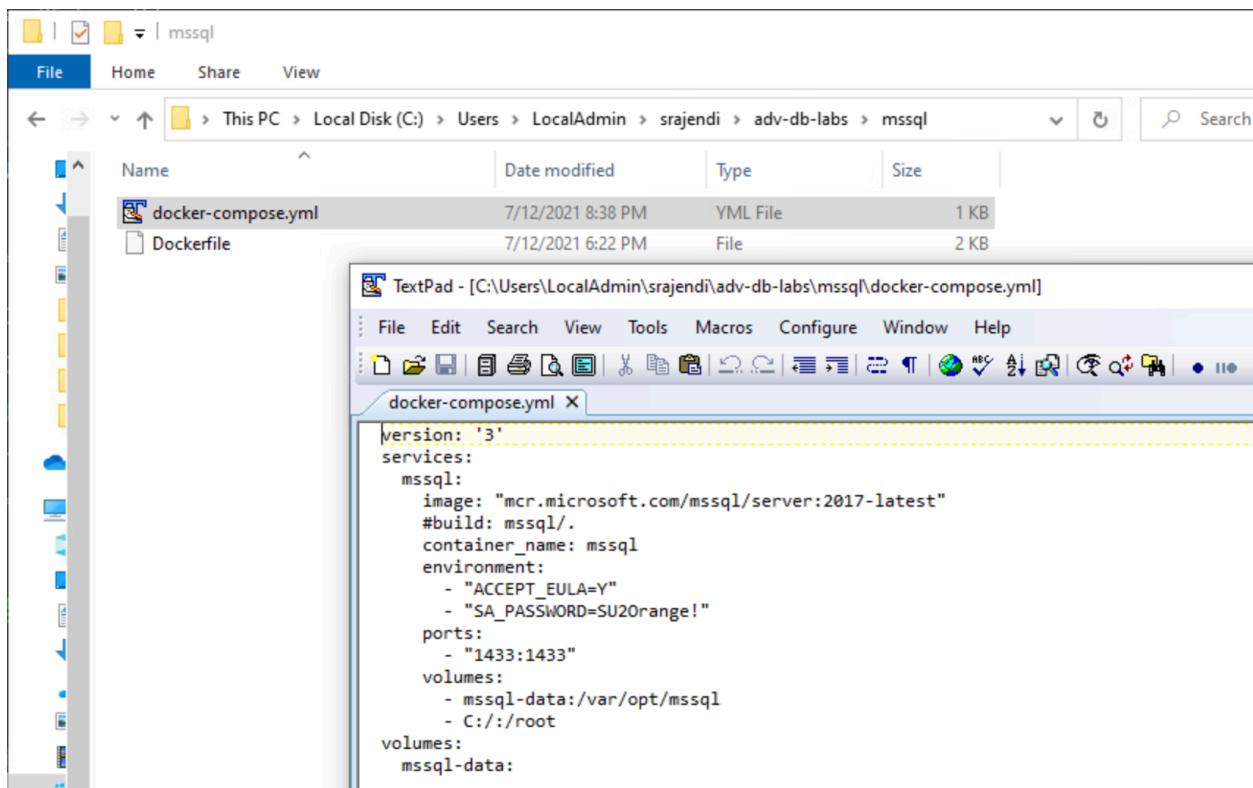
```
c:\ Administrator: Command Prompt
Microsoft Windows [Version 10.0.19041.1110]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql

C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql>docker-compose ps
Name      Command     State    Ports
-----
C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql>docker-compose up -d
Creating network "mssql_default" with the default driver
Creating mssql ... done

C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql>docker-compose ps
Name      Command     State    Ports
-----
mssql   /opt/mssql/bin/nonroot_msg ...   Up      0.0.0.0:1433->1433/tcp,:::1433->1433/tcp

C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql>
```



```
srajendi_HW4.sql -...lhost.demo (sa (52))* ✎ X
1  ****
2  IST769 Homework Submission
3  Name: Sathish Kumar Rajendiran
4  SUID: 666555028
5  Email: srajendi@syr.edu
6  Date Due: 08/03/2021
7  Date Due: Performance, Security, NoSQL
8  Homework #:4
9  ****
10 1. Create a non-clustered index on the timesheets table in the demo database
11 The index you create should be designed to improve the following query:
12
13 select employee_id, employee_firstname, employee_lastname,
14      employee_address, employee_email, employee_phone
15
16
17
18
19
20
21
22
23
24
25
26
27
28
```

194 %

Query executed successfully.

localhost (14.0 RTM) | sa (52) |

```
srajendi_HW4.sql -...lhost.demo (sa (52))* ✎ X
19  ----Pre-setup command
20  ----Use [fudgemart_v3]
21  ----Go
22  ----select * into demo.dbo.timesheets
23  ----from fudgemart_employees join fudgemart_employee_timesheets
24  ----on employee_id = timesheet_employee_id
25  ----Go
26
27  USE [demo]
28  GO
```

```
srajendi_HW4.sql -...lhost.demo (sa (52))* ✎ X
88 2. Write an SQL Select query which uses the index you created in the first question
     but does an index seek instead of an index scan.
89 */
90
91 SET STATISTICS TIME ON
92 SET STATISTICS IO ON
93 GO
94
95 --after Index creation
96 declare @startTime datetime2
97 declare @endTime datetime2
98 --declare @maxemployeeID int
99 set @startTime = getdate()
```

```
srajendi_HW4.sql -> localhost.demo (sa (52)) * X
117 ****
118 3. Create a single columnstore index on the timesheets table in the demo database
   which will improve the following queries:
119
120 select employee_department, sum(timesheet_hours)
121 from timesheets group by employee_department
122
123 select employee_jobtitle, avg(timesheet_hourlyrate)
124 from timesheets group by employee_jobtitle
125 */
126 SET STATISTICS TIME ON
127 SET STATISTICS IO ON
128 GO
```

194 % Query executed successfully. | localhost (14.0 RTM) | sa (52) | demo | 00:00:02

```
srajendi_HW4.sql -> localhost.demo (sa (52)) * X
149 ---- Find an existing index named IX_CLM_Timesheets_Employee_Department_Hours
150 ---- and delete it if found.
151 IF EXISTS (SELECT name FROM sys.indexes
152             WHERE name = N'IX_CLM_Timesheets')
153     DROP INDEX IX_CLM_Timesheets ON dbo.timesheets;
154     GO
155 -- Create a nonclustered COLUMNSTORE index called IX_CLM_Timesheets
156 -- on the dbo.timesheets table using the employee_id,
157 -- employee_department,timesheet_hours,timesheet_hourlyrate,timesheet_hours columns.
158 CREATE NONCLUSTERED COLUMNSTORE INDEX IX_CLM_Timesheets
159     ON dbo.timesheets(employee_department,timesheet_hours)
160     GO
```

194 % Query executed successfully. | localhost (14.0 RTM) | sa (52) | demo | 00:00:02 | 2

```
srajendi_HW4.sql -> localhost.demo (sa (52)) * X
245 ****
246
247 4. Create an indexed view named v_employees on the timesheets table in the demo
   database which lists the employee id, first name, last name, job title, and
   department columns values and one row per employee (essentially re-building the
   employee table). Then set a unique clustered index on the view and finish by
   writing an SQL Select query which uses the indexed view.
248 */
249
250 **** SQL Views Creation - Begins ****
251 ----- v_employees - Creation
252 ----- This view returns product details.
253 */
```

194 % Query executed successfully. | localhost (14.0 RTM) | sa (52) | demo | 00:00:02

```
srajendi_HW4.sql -...lhost.demo (sa (52))* ✘ X
261 CREATE VIEW v_employees WITH schemabinding
262 AS
263     SELECT employee_id
264         , employee_firstname
265         , employee_lastname
266         , employee_jobtitle
267         , employee_department
268         , COUNT_BIG(*) as employee_count
269     from dbo.timesheets
270     Group by
271         employee_id
272         , employee_firstname
273         , employee_lastname
274         , employee_department
194 %
Query executed successfully.
```

```
srajendi_HW4.sql -...lhost.demo (sa (52))* ✘ X
285 DROP INDEX IX_CLS_v_employees_EmployeeID ON dbo.v_employees;
286 GO
287 -- Create an unique clustered index called IX_CLS_v_employees_EmployeeID
288 -- on the dbo.v_employees table using the employee_id
289
290 CREATE UNIQUE CLUSTERED INDEX IX_CLS_v_employees_EmployeeID
291 ON dbo.v_employees(employee_id)
292 GO
293 --query from the view after index creation
294 SELECT * FROM dbo.v_employees
295 GO
296
297 ****
298 E. Output the following query in JSON format. Display the employee_id, first
194 %
Query executed successfully.    localhost (14.0 RTM) | sa (52)
```

```
srajendi.HW4.sql -> localhost.demo (sa (52)) * → X
297  ****
298 5. Output the following query in JSON format: Display the employee id, first name,
   last name, count of timesheets, total hours worked, and average timesheet hourly
   rate.
299 */
300 --simple select sql with SQL Output
301 select
302     employee_id
303     , employee_firstname
304     , employee_lastname
305     , count(*) as timesheet_count
306     , sum(timesheet_hours) as total_hours_worked
307     , AVG(timesheet_hourlyrate) as avg_timesheet_rate
308 from timesheets
194 %
```

```
Query executed successfully. | localhost (14.0 RTM) | sa (52) | demo | 00:00:02 | 22
194 %
315 --SQL statement with NOSQL Output
316 select
317     employee_id
318     , employee_firstname , employee_lastname
319     , count(*) as timesheet_count
320     , sum(timesheet_hours) as total_hours_worked
321     , AVG(timesheet_hourlyrate) as avg_timesheet_rate
322 from timesheets
323 group by
324     employee_id
325     , employee_firstname
326     , employee_lastname
327 FOR JSON AUTO
328 GO
194 %
Query executed successfully. | localhost (14.0 RTM)
```

srajendi\_HW4.sql -...lhost.demo (sa (52))

194 %

Results Messages Execution plan

	employee_id	employee_firstname	employee_lastname	employee_jobtitle	employee_department	employee_count
4	4	Sandi	Shores	Sales Associate	Hardware	156
5	5	Isabelle	Gurnering	Department Ma...	Electronics	208
6	6	Lee	Hvmeehom	Department Ma...	Hardware	208
7	7	Allan	Wrench	Sales Associate	Housewares	208
8	8	Ally	Gator	Sales Associate	Sporting Goods	208
9	9	Alma	Frenzergon	Sales Associate	Housewares	208
10	10	Atie	Choke	Sales Associate	Hardware	208
11	11	Bette	Altot	Sales Associate	Sporting Goods	208

	employee_id	employee_firstname	employee_lastname	timesheet_count	total_hours_worked	avg_timesheet_rate
22	23	Nat	Tural	171	2801.0	11.6978
23	19	Kent	Belevit	208	8320.0	9.7235
24	20	Mary	Mi	208	8320.0	20.1158
25	6	Lee	Hvmeehom	208	8320.0	17.955
26	17	Kurt	Tan	208	3338.0	9.5177
27	27	Sara	Docktur-Indahaus	208	8320.0	19.7557
28	28	Sara	Isnromor	192	7680.0	9.2822
29	25	Patty	O'Fumiture	208	8320.0	11.2154

JSON\_F52E2B61-18A1-11d1-B105-00805F49916B

1 [{"employee\_id": "14", "employee\_firstname": "Chris", "employee\_lastname": "P..."}]

Query executed successfully.

localhost (14.0 RTM) | sa (52) | demo | 00:00:02 | 228 rows

769-Win10Docker-srajendi

Enforce US Keyboard Layout | View Fuls

Lab4 - Google Drive

drive.google.com/drive/folders/1lxblF5Y-la\_kRKlhMnKNKooOSgXqGNu2

Drive Search in Drive

New Priority My Drive > IST769 > Lab4

Now you can block people in Drive. To prevent people outside your organization from sharing unwanted files with you, right-click a file shared with you, and choose Block. Learn more

Name	Owner	Last modified	File size
srajendi_HW4.sql	me	2:23 AM me	10 KB

Priority My Drive Shared drives Shared with me

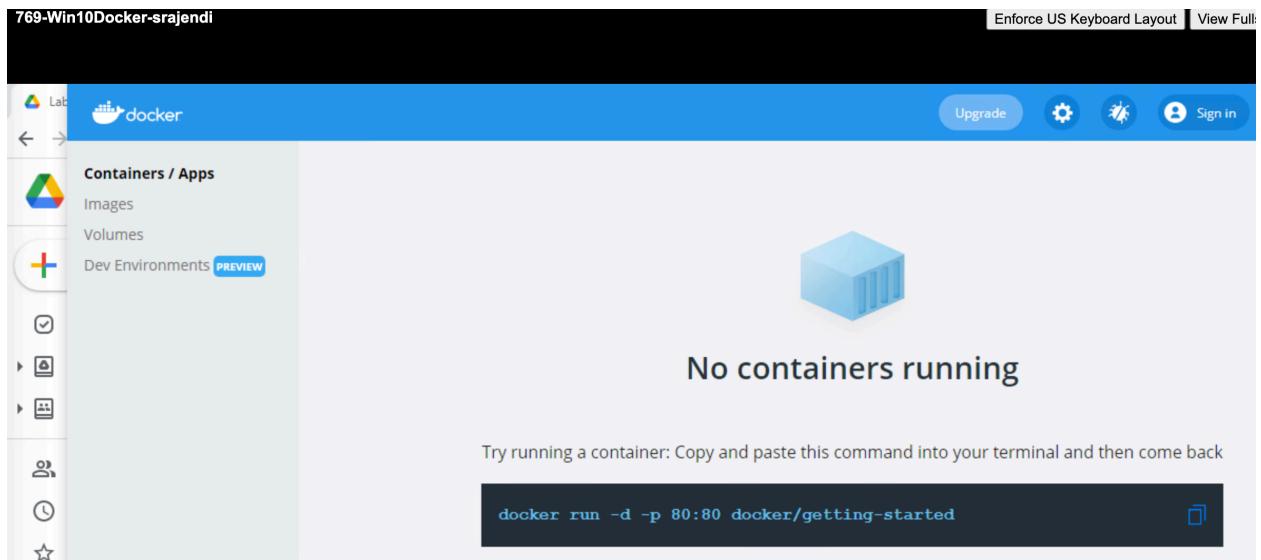
769-Win10Docker-srajendi

Containers / Apps

Images Volumes Dev Environments PREVIEW

mssql RUNNING

mssql mcr.microsoft.c... RUNNING PORT: 1433



769-Win10Docker-srajendi

The screenshot shows a Windows Command Prompt window titled 'Administrator: Command Prompt'. The user runs several Docker commands from the directory 'C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql'. The commands include 'cd', 'docker-compose ps', 'docker-compose up -d', 'docker-compose ps', 'docker-compose down', and 'docker-compose ps' again. The output shows the creation of a network named 'mssql\_default', the start of a container named 'mssql' (state 'Up'), and the stopping and removal of the container.

```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.19041.1110]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql

C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql>docker-compose ps
Name      Command      State      Ports
-----
mssql    /opt/mssql/bin/nonroot_msg ...   Up      0.0.0.0:1433->1433/tcp,:::1433->1433/tcp

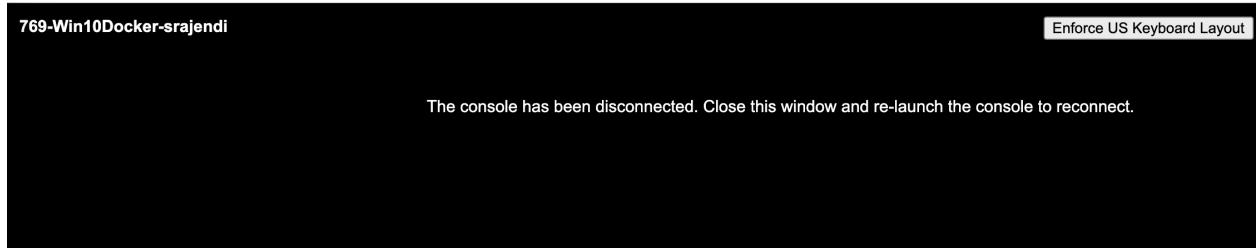
C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql>docker-compose up -d
Creating network "mssql_default" with the default driver
Creating mssql ... done

C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql>docker-compose ps
Name      Command      State      Ports
-----
mssql    /opt/mssql/bin/nonroot_msg ...   Up      0.0.0.0:1433->1433/tcp,:::1433->1433/tcp

C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql>docker-compose ps
Name      Command      State      Ports
-----
mssql    /opt/mssql/bin/nonroot_msg ...   Up      0.0.0.0:1433->1433/tcp,:::1433->1433/tcp

C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql>docker-compose down
Stopping mssql ... done
Removing mssql ... done
Removing network mssql_default

C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql>docker-compose ps
Name      Command      State      Ports
-----
```



The screenshot shows the vSphere Client interface. At the top, there are several tabs: 'vSphere - 769-Win10Docker-srajendi', 'sql server 2014 - Always comi...', and others. Below the tabs, the address bar shows the URL: [vlab.ischool.syr.edu/ui/#?extensionId=vsphere.core.inventory.serverObjectViewsExtension&objectId=urn:vmomi:VirtualMachine:vm...](http://vlab.ischool.syr.edu/ui/#?extensionId=vsphere.core.inventory.serverObjectViewsExtension&objectId=urn:vmomi:VirtualMachine:vm...). The main content area displays a summary for a virtual machine named '769-Win10Docker-srajendi'. The summary card indicates the VM is 'Powered Off'. It provides guest OS information (Microsoft Windows 10 (64-bit)), compatibility (ESXi 6.7 Update 2 and later (VM version 15)), and tools status (Not running, version:11333 (Current)). It also lists DNS Name, IP Addresses, and Host. On the right side, resource usage is shown: CPU USAGE (0 Hz), MEMORY USAGE (0 B), and STORAGE USAGE (117.15 GB). Below the summary card, sections for 'VM Hardware' and 'Related Objects' are visible, along with a 'Notes' section containing VM-specific details.