

# Unit 12 Problem Set Submission Form

## Overview

|               |                            |
|---------------|----------------------------|
| Your Name     | Sushil Sunilkumar Deshmukh |
| Your SU Email | sdeshmuk@syr.edu           |

## Instructions

Put your name and SU email at the top. Answer these questions all from the lab. When asked to include screenshots, please follow the screen shot guidelines from the first lab.

Remember as you complete the problem sets it is not only about getting it right / correct. We will discuss the answers in class so it's important to articulate anything you would like to contribute to the discussion in your answer:

- If you feel the question is vague, include any assumptions you've made.
- If you feel the answer requires interpretation or justification provide it.
- If you do not know the answer to the question, articulate what you tried and how you are stuck.

This is how you receive credit for answering questions which might not be correct.

## Questions

Answer these questions using the problem set submission template. You will need to consult the logical model in the overview section for details. For any screenshots provided, please follow the guidelines for submitting a screenshot.

Write the following as SQL programs. For each, include the SQL as a screenshot with the output of the SQL Code.

1. Using the **payroll** database write an index to improve the performance of the following query. Your screenshot should include the created index SQL code and the query plan demonstrating the index is being used.

```

27 go
28 -- # 1
29
30 select employee_id, employee_firstname, employee_lastname, employee_jobtitle
31   from employees
32     where employee_jobtitle = 'Store Manager'
33       or employee_jobtitle = 'Owner'
34
35 GO
36 drop index if exists ix_employee_jobtitle on employees
37 go
38 create index ix_employee_jobtitle on employees(employee_jobtitle)

```

Results    Messages    **Query Plan**    Top Operations

Query 1  
select employee\_id, employee\_firstname, employee\_lastname, employee\_jobtitle from employees where employee\_jobtitle = 'Store Manager' or employee\_jobtitle = 'Owner'

Index Seek (NonClustered)  
[employees].[ix\_employee\_jobtitle]  
Cost: 100%

- Write another query using GROUP BY which also uses the index you created in the first question.

```

39 GO
40
41
42 -- # 2
43 select employee_jobtitle, count(*)
44   from employees
45   group by employee_jobtitle
46
47
48 GO
49 use vbay
50 go

```

Results    Messages    **Query Plan**    Top Operations

Query 1  
select employee\_jobtitle, count(\*) from employees group by employee\_jobtitle

Index Scan (NonClustered)  
[employees].[ix\_employee\_jobtitle]  
Cost: 99%

Stream Aggregate (Aggregate)  
Cost: 1%

Compute Scalar  
Cost: 0%

- For the following query from a previous assignment, which provides a rank of each bid on an item:

```

select item_id, item_name,
       dense_rank() over
           ( partition by item_name order by bid_datetime) as bid_order,
       bid_amount,
       lag(user_firstname + ' ' + user_lastname) over
           (partition by item_name order by bid_datetime) as prev_bidder,
       user_firstname + ' ' + user_lastname as bidder,
       lead(user_firstname + ' ' + user_lastname) over
           (partition by item_name order by bid_datetime) as next_bidder
  from vb_items
    join vb_bids on item_id=bid_item_id
    join vb_users on bid_user_id = user_id
 where bid_status='ok'

```

implement the query and run it. Provide a screenshot of the query plan and include the portion where the **vb\_bids**, **vb\_items**, and **vb\_users** tables are selected and joined together.

The screenshot shows the Azure Data Studio interface with the following details:

- File Bar:** Activities, Azure Data Studio, File, Edit, View, Help.
- Toolbar:** Includes icons for File Explorer, Database, Security, Server Objects, and more.
- Connections:** Localhost, <default> (sa).
- Query Editor:** Title: problem-set-solution(2)(1)(1).sql - localhost.demo (sa) - Azure Data Studio. Content: The provided SQL query.
- Results:** Shows a single row of data for a user named Sushil Deshmukh.
- Query Plan:** Shows the execution plan with nodes: SELECT, Compute Scalar, Stream Aggregate, Window Spool, Segment, Compute Scalar, Sequence Project, and Stream (A).

4. Write an index to improve performance of the query by replacing the clustered index scan on **vb\_bids**



with an index seek on the same table. Provide a screenshot of your index code and a screenshot of the query plan demonstrating the index is being used to draw data into the query.

```

65     join vb_bids on item_id=bid_item_id
66     join vb_users on bid_user_id = user_id
67     where bid_status='ok'
68
69 -- #4 improve performance
70
71 drop index if exists ix_bids_bid_status on vb_bids
72 go
73 create index ix_bids_bid_status on vb_bids(bid_status)
    include (bid_item_id, bid_user_id, bid_amount, bid_datetime)
74
75
76
77
78 -- #5
79 use fudgemart_v3;

```

5. Using **fudgemart\_v3**, create a schemabound view from the following query:

```

select c.customer_state, c.customer_firstname + ' ' + c.customer_lastname as customer_name,
datepart(year,order_date) as order_year, o.order_id, o.ship_via,
od.order_qty as order_detail_qty, od.order_qty * p.product_retail_price as order_detail_extd_price,
p.product_id, p.product_name, p.product_department
    from dbo.fm_orders o
    join dbo.fm_customers c on o.customer_id = c.customer_id
    join dbo.fm_order_details od on o.order_id = od.order_id
    join dbo.fm_products p on p.product_id = od.product_id

```

Name the view **v\_orders**. Provide a screenshot of the code and sample output which conveys the query ran and created the view.

```

85     with SCHEMABINDING AS
86         select c.customer_state, c.customer_firstname + ' ' + c.customer_lastname as customer_name,
87             datepart(year,order_date) as order_year, o.order_id, o.ship_via,
88             od.order_qty as order_detail_qty, od.order_qty * p.product_retail_price as order_detail_extd_price,
89             p.product_id, p.product_name, p.product_department
90         from dbo.fm_orders o
91         join dbo.fm_customers c on o.customer_id = c.customer_id
92         join dbo.fm_order_details od on o.order_id = od.order_id
93         join dbo.fm_products p on p.product_id = od.product_id
94     GO
95     SELECT * FROM v_orders
96 -- #6
97     drop index if exists cix_v_orders on v_orders

```

|   | customer_state | customer_name | order_year | order_id | ship_via       | order_detail_qty | order_detail_extd_price | product_id | product_name |
|---|----------------|---------------|------------|----------|----------------|------------------|-------------------------|------------|--------------|
| 1 | CA             | Otto Tyme     | 2009       | 1        | JiffyEx        | 1                | 38.0000                 | 12         | Work Pant    |
| 2 | CA             | Otto Tyme     | 2009       | 1        | JiffyEx        | 2                | 24.0000                 | 14         | Comfor-fi    |
| 3 | CA             | Otto Tyme     | 2009       | 1        | JiffyEx        | 1                | 75.0000                 | 16         | X-Train S    |
| 4 | CA             | Otto Tyme     | 2009       | 1        | JiffyEx        | 2                | 20.0000                 | 49         | Pedometer    |
| 5 | DC             | Sandy Beeches | 2009       | 2        | UDS            | 1                | 255.0000                | 50         | Sport Cyc    |
| 6 | AZ             | Ty Anott      | 2009       | 3        | Postal Service | 5                | 190.0000                | 12         | Work Pant    |

6. Write code to add a unique clustered index to the view **v\_orders**. Execute your view (**select \* from v\_orders**) and then observe the query plan to see if the index is being used. If the index is not being used, that's an indication there is not enough data to warrant the index. You can force the index to be used by using the **noexpand** option on the query: **select \* from v\_orders with (noexpand)**

**(noexpand)** Provide a screenshot of code to create the index and execute the view along with the query plan showing the index is used.

The screenshot shows the Azure Data Studio interface. In the top bar, it says "May 8 18:16" and "problem-set-solution(2)(1)(1).sql - localhost.fudgemart\_v3 (sa) - Azure Data Studio". The left sidebar shows "CONNECTIONS" with "localhost, <default> (sa)" selected. The main area has a tab bar with "File Edit View Help", "File", "Edit", "View", "Help", "... (1).sql - disconnected", "Fudgenbooks\_solution.sql - localh...mo (sa)", "problem-set-solution(2)(1)(1).sql - localhost.fudgemart\_v3 (sa)", "Untitled-2", and "SQLCMD". Below the tabs, there are buttons for "Run", "Cancel", "Disconnect", "Change Connection", "Fudgemart\_v3", "Explain", "Enable SQLCMD", and "Export as Notebook". The "Query Plan" tab is selected. A query window contains the following code:

```

94 GO
95 SELECT * FROM v_orders
96 -- # 6
97 drop index if exists cix_v_orders on v_orders
98 go
99 create unique clustered index cix_v_orders on v_orders(order_id,product_id)
100 go
101 Select * from v_orders with (noexpand)

```

Below the code, the "Results" tab shows the output of the query: "select \* from v\_orders". The "Query Plan" tab shows a complex execution plan with multiple stages of joins and scans across several tables: fm\_customers, fm\_products, fm\_orders, and fm\_order\_details. The plan includes Compute Scalar operations and Hash Match (Inner Join) operations. The cost of the plan is 100%.

7. Write code to add a columnstore index to **v\_orders** include all the columns from the view in the column store index. Provide screenshots with code to demonstrate you created the columnstore index and that these queries use it:

```

select product_name, sum(order_detail_qty)
    from v_orders with (noexpand)
        group by product_name

```

```

select distinct customer_name, product_department
    from v_orders with (noexpand)

```

The screenshot shows the SQL Server Management Studio interface. In the top bar, it says "Welcome" and "SQLQuery\_1 - localhost.fudgemart\_v3 (sa)". The left sidebar shows "CONNECTIONS" with "localhost, <default> (sa)" selected. The main area has a tab bar with "File", "Edit", "View", "Help", "File", "Edit", "View", "Help", "... (1).sql - disconnected", "fudgemart\_v3", "problem-set-solution(2)(1)(1).sql - localhost.fudgemart\_v3 (sa)", "Untitled-2", and "SQLCMD". Below the tabs, there are buttons for "Run", "Cancel", "Disconnect", "Change Connection", "fudgemart\_v3", "Explain", "Enable SQLCMD", and "Export as Notebook". The "Query Plan" tab is selected. A query window contains the following code:

```

119 -- # 7
120 create COLUMNSTORE
121     index ix_v_orders
122         on v_orders(customer_state, customer_name, order_year, order_id, ship_via, order_detail_qty, order_qty)
123 GO
124
125
126 select product_name, sum(order_detail_qty)

```

To the right of the query window, a small terminal window shows the command "Name: Sushil Deshmukh" and the response "Name: Sushil Deshmukh". Below the query window, the "Results" tab shows the output of the first query: "select product\_name, sum(order\_detail\_qty) from v\_orders with (noexpand) group by product\_name". The "Query Plan" tab shows a query plan with a Hash Match (Aggregate) operation and a Columnstore Index Scan (ViewNonClustered) on the v\_orders view. The cost of the plan is 68%. Below the first query, the "Messages" tab shows the creation of the columnstore index. The "Query Plan" tab also shows a second query plan for the second query: "select distinct customer\_name, product\_department from v\_orders with (noexpand)". This plan uses a Hash Match (Aggregate) operation and a Columnstore Index Scan (ViewNonClustered) on the v\_orders view. The cost of this plan is 69%.

## Reflection

Use this section to reflect on your learning. To achieve the highest grade on the assignment you must be as descriptive and personal as possible with your reflection.

1. What are the key things you learned through the process of completing this assignment?  
Performance improvement and indexing of the queries. Physical model of the sql programming
  
2. What were the challenges or roadblocks (if any) you encountered on the way to completing it?  
This topic is difficult to understand overall.
  
3. Were you prepared for this assignment? What can you do to be better prepared?  
No, I was not prepared
  
4. Now that you have completed the assignment rate your comfort level with this week's material. This should be an honest assessment: (choose one)  
1.  
  
4 ==> I understand this material and can explain it to others.  
3 ==> I understand this material.  
2 ==> I somewhat understand the material but sometimes need guidance from others.  
1 ==> I understand very little of this material and need extra help.