

# Unit 12 Problem Set Submission Form

## Overview

Your Name	Jainish Savaliya
Your SU Email	jsavaliy@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 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.

Run Cancel Disconnect Change Connection payroll Estimated Plan Disable Actual Plan Enable SQL

```

1 USE payroll
2 GO
3 DROP INDEX IF EXISTS i_employee_jobtitle ON employees
4 CREATE INDEX i_employee_jobtitle ON employees(employee_jobtitle)
5 INCLUDE (employee_id, employee_firstname, employee_lastname)
6 GO
7
8 SELECT employee_id, employee_firstname, employee_lastname, employee_jobtitle FROM employees
9 WHERE employee_jobtitle='Store Manager'
10 OR employee_jobtitle='Owner'
11

```

Results Messages **Query Plan (Preview)** Plan Tree (Preview) Top Operations (Preview)

Query 1: Query cost (relative to the script): 89.50%

insert [dbo].[employees] select \* from [dbo].[employees] option (maxdop 1)

```

graph RL
    TSQL[T-SQL  
INSERT  
Cost: 0 %] --> IndexInsert[Index Insert  
[employees].[i_employee_jobtitle]  
Cost: 45 %  
0.002s  
0 of 67 (0%)]
    IndexInsert --> Sort[Sort  
Cost: 43 %  
0.000s  
67 of 67 (100%)]
    Sort --> ClusteredIndexScan[Clustered Index Scan  
[employees].[pk_employees_employee_...]  
Cost: 12 %  
0.000s  
67 of 67 (100%)]

```

T-SQL  
INSERT  
Cost: 0 %

Index Insert  
[employees].[i\_employee\_jobtitle]  
Cost: 45 %  
0.002s  
0 of 67 (0%)

Sort  
Cost: 43 %  
0.000s  
67 of 67 (100%)

Clustered Index Scan  
[employees].[pk\_employees\_employee\_...]  
Cost: 12 %  
0.000s  
67 of 67 (100%)

jsavaliy@s

File Edit View

jsavaliy@syr.edu

Ln 1, Col 17 100% Windows (CRLF) UTF-8

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

Run Cancel Disconnect Change Connection payroll Estimated Plan Disable Actual Plan

```

1  USE payroll
2  GO
3  DROP INDEX IF EXISTS i_employee_jobtitle ON employees
4  CREATE INDEX i_employee_jobtitle ON employees(employee_jobtitle)
5  INCLUDE (employee_id, employee_firstname, employee_lastname)
6  GO
7
8  SELECT employee_id, employee_firstname, employee_lastname, employee_jobtitle FROM employees
9  WHERE employee_jobtitle='Store Manager'
10 OR employee_jobtitle='Owner'
11
12 SELECT COUNT (employee_id) AS employee_count, employee_jobtitle FROM employees_history
13 WHERE employee_jobtitle = 'Store Manager'
14 OR employee_jobtitle = 'Owner'
15 Group BY employee_jobtitle

```

Results Messages Query Plan (Preview) Plan Tree (Preview) Top Operations (Preview)

Query 1: Query cost (relative to the script): 60.38%

insert [dbo].[employees] select \* from [dbo].[employees] option (maxdop 1)

T-SQL  
Cost: 0 %

Index Insert  
[employees].[i\_employee\_jobtitle]  
Cost: 45 %  
0.002s  
67 of 67 (100%)

Sort  
Cost: 43 %  
0.000s  
67 of 67 (100%)

Clustered Index Scan  
[employees].[pk\_employees\_employee\_...]  
Cost: 12 %  
0.000s  
67 of 67 (100%)

jsavaliy@s

File Edit View

jsavaliy@syr.edu

Ln 1, Col 17 100% Windows (CRLF) UTF-8

3. 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.

SQLQuery\_2 - localh...ay (sa) • localhost • SQLQuery\_1 - disconnected •

Run Cancel Disconnect Change Connection vbay Estimated Plan Disable Actual Plan Enable SQLCMD Export as Notebook

```

1 SELECT item_id, item_name,
2 dense_rank()
3 OVER (partition by item_name order by bid_datetime) as bid_order, bid_amount, lag(user_firstname + ' ' + user_lastname)
4 over (partition by item_name order by bid_datetime) as prev_bidder, user_firstname + ' ' + user_lastname as bidder, lead(user_firstname + ' ' + user_lastname)
5 over (partition by item_name order by bid_datetime) as next_bidder from vb_items
6 join vb_bids on item_id = bid_item_id
7 join vb_users on bid_user_id = user_id
8 WHERE bid_status = 'ok'

```

Results Messages Query Plan (Preview) Plan Tree (Preview) Top Operations (Preview)

Query 1: Query cost (relative to the script): 100.00%

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 b

Segment Cost: 0% 0.000s 83 of 34 (185%)

Sequence Project (Compute Scalar) Cost: 0.01% 0.000s 83 of 34 (185%)

Segment Cost: 0% 0.000s 83 of 34 (185%)

Sort Cost: 47.93% 0.000s 83 of 34 (185%)

Nested Loops (Inner Join) Cost: 0.16% 0.000s 83 of 34 (185%)

Compute Scalar Cost: 0%

Nested Loops (Inner Join) Cost: 0.29% 0.000s 83 of 8 (787%)

Clustered Index Scan [vb\_bids].[pk\_bid\_id] Cost: 13.95% 0.000s 83 of 8 (787%)

Clustered Index Seek [vb\_items].[pk\_item\_id] Cost: 18.41% 0.000s 83 of 8 (787%)

Clustered Index Seek [vb\_users].[pk\_user\_id] Cost: 18.41% 0.000s 83 of 8 (787%)

jsavaliy@s + - □ ×

File Edit View

jsavaliy@syr.edu

Ln 1, Col 17 100% Windows (CRLF) UTF-8

- 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.

Run Cancel Disconnect Change Connection vbay Estimated Plan Disable Actual Plan Enable SQLCMD Export as Notebook

```

1 USE vbay
2 GO
3 DROP INDEX IF EXISTS i_bid_status on vb_bids
4 CREATE INDEX i_bid_status on vb_bids(bid_status)
5 INCLUDE (bid_id, bid_item_id, bid_user_id, bid_datetime, bid_amount)
6 GO
7
8 SELECT item_id, item_name,
9 dense_rank()
10 OVER (partition by item_name order by bid_datetime) as bid_order, bid_amount, lag(user_firstname + ' ' + user_lastname
11 over (partition by item_name order by bid_datetime) as prev_bidder, user_firstname + ' ' + user_lastname as bidder, lead(user_firstname + ' ' + user_lastname)
12 over (partition by item_name order by bid_datetime) as next_bidder from vb_items
13 join vb_bids on item_id = bid_item_id
14 join vb_users on bid_user_id = user_id
15 WHERE bid_status = 'ok'

```

Results Messages Query Plan (Preview) Plan Tree (Preview) Top Operations (Preview)

Query 1: Query cost (relative to the script): 37.52%

Insert [dbo].[vb\_bids] select \* from [dbo].[vb\_bids] option (maxdop 1)

T-SQL  
INSERT  
Cost: 0 %

Index Insert  
[vb\_bids] [i\_bid\_status]  
Cost: 40 %  
0.004s  
0 of  
72 (0%)

Sort  
Cost: 47 %  
0.000s  
72 of  
72 (100%)

Clustered Index Scan  
[vb\_bids] [pk\_bid\_id]  
Cost: 13 %  
0.000s  
72 of  
72 (100%)

jsavaliy@s

File Edit View

jsavaliy@syr.edu

Ln 1, Col 17 | 100% Windows (CRLF) UTF-8

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.

Run Cancel Disconnect Change Connection fudgemart\_v3 Estimated Plan Disable Actual Plan Enable SQLCMD Export as Notebook

```

1 USE fudgemart_v3
2 GO
3 DROP VIEW IF EXISTS v_orders
4 GO
5 CREATE VIEW v_orders with SCHEMABINDING AS
6 (
7     SELECT c.customer_state, c.customer_firstname + ' ' + c.customer_lastname as customer_name,
8     datepart(year,order_date) as order_year, o.order_id, o.ship_via,
9     od.order_qty AS order_detail_qty, od.order_qty * p.product_retail_price AS order_detail_extd_price,
10    p.product_id, p.product_name, p.product_department
11    FROM dbo.fm_orders o
12    join dbo.fm_customers c on o.customer_id = c.customer_id
13    join dbo.fm_order_details od ON o.order_id = od.order_id
14    join dbo.fm_products p on p.product_id = od.product_id
15 )
16 GO
17
18 SELECT TOP 5 * FROM v_orders

```

Results Messages Query Plan (Preview) Plan Tree (Preview) Top Operations (Preview)

jsavaliy@s

File Edit View

jsavaliy@syr.edu

Ln 1, Col 17 100% Windows (CRLF) UTF-8

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

Run Cancel Disconnect Change Connection fudgemart\_v3 Estimated Plan Disable Actual Plan

```

1
2 USE fudgemart_v3
3 GO
4 DROP VIEW IF EXISTS i_v_orders ON dbo.v_orders
5 GO
6 CREATE UNIQUE CLUSTERED INDEX i_v_orders ON dbo.v_orders(order_id,product_id)
7 GO
8 SELECT * FROM v_orders with (NOEXPAND)

```

jsavaliy@s + - □ ×

File Edit View

jsavaliy@syr.edu

Ln 1, Col 17 100% Windows (CRLF) UTF-8

Results Messages Query Plan (Preview) Plan Tree (Preview) Top Operations (Preview)

Query 1: Query cost (relative to the script): 100.00%

SELECT \* FROM v\_orders with (NOEXPAND)

SELECT  
Cost: 0 %

Clustered Index Scan  
[v\_orders].[i\_v\_orders]  
Cost: 100 %  
0.011s  
10466 of  
10466 (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)

```



Run Cancel Disconnect Change Connection fudgemart\_v3 Estimated Plan Disable Actual Plan Enable SQL

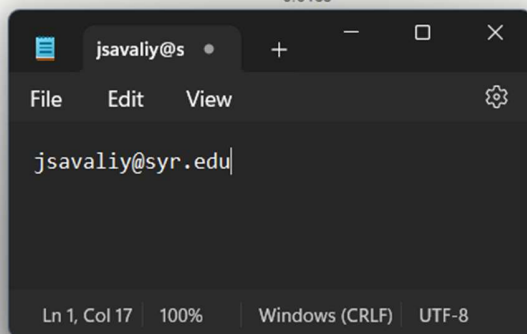
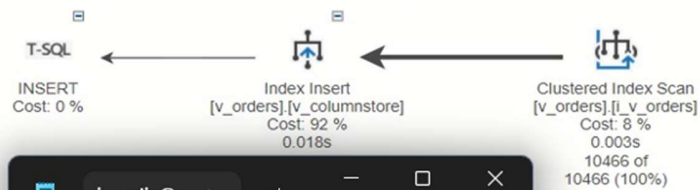
```

1 USE fudgemart_v3
2 GO
3 DROP INDEX IF EXISTS v_columnstore on dbo.v_orders
4 CREATE INDEX v_columnstore on dbo.v_orders(order_id)
5 INCLUDE (customer_state,customer_name,order_year,ship_via,order_detail_extd_price,product_name,product_department)
6 GO
7 SELECT product_name, sum(order_details_qty) from v_orders with (noexpand)
8 GROUP BY product_name
9

```

Results Messages **Query Plan (Preview)** Plan Tree (Preview) Top Operations (Preview)

Query 1: Query cost (relative to the script): 100.00%  
 Insert [dbo].[v\_orders] select \* from [dbo].[v\_orders] option (maxdop 1)





Run Cancel Disconnect Change Connection fudgemart\_v3 Estimated Plan Disable Actual Plan Enable SQ

```

1 USE fudgemart_v3
2 GO
3 DROP INDEX IF EXISTS v_columnstore on dbo.v_orders
4 CREATE INDEX v_columnstore on dbo.v_orders(order_id)
5 INCLUDE (customer_state,customer_name,order_year,ship_via,order_detail_extd_price,product_name,product_department)
6 GO
7
8 SELECT DISTINCT customer_name,product_department
9 FROM v_orders with (noexpand)
10

```

Results Messages **Query Plan (Preview)** Plan Tree (Preview) Top Operations (Preview)

Query 1: Query cost (relative to the script): 87.34%

insert [dbo].[v\_orders] select \* from [dbo].[v\_orders] option (maxdop 1)

The diagram illustrates the query execution plan. It starts with 'T-SQL' (INSERT) with a cost of 0%. This leads to an 'Index Insert' operation on '[v\_orders].[v\_columnstore]' with a cost of 92%. Finally, it leads to a 'Clustered Index Scan' on '[v\_orders].[i\_v\_orders]' with a cost of 8%, taking 0.005s to execute and scanning 10466 of 10466 rows (100%).

jsavaliy@s

File Edit View

jsavaliy@syr.edu

Ln 1, Col 17 100% Windows (CRLF) UTF-8

## 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.

- What are the key things you learned through the process of completing this assignment?  
-> I learnt About performance and Indexing.
- What were the challenges or roadblocks (if any) you encountered on the way to completing it?  
-> I was getting a lot of errors while executing.
- Were you prepared for this assignment? What can you do to be better prepared?  
-> I watched the video so, was prepared for the assignment.
- Now that you have completed the assignment rate your comfort level with this week's material. This should be an honest assessment: (choose one)  
4 ==> I understand this material and can explain it to others.