

Lab 9 Submittal

Step 1: Reviewing the dataset.

In the Quarterly Sales Report, sales information for each division in 2024 and 2025 is broken down. Sales for Division I in 2024 were \$33,759, with \$8,734, \$9,112, \$8,973, and \$8,117 for Quarters 1 through 4. Division I, in contrast, reported \$8,365, \$8,628, \$8,661, and \$9,234 for the same quarters in 2025, totaling \$34,888. Comparably, sales for other divisions varied from quarter to quarter. Notably, Division II showed a growing tendency with \$33,610 in 2025 as opposed to \$32,274 in 2024. Overall, Division I showed a little gain in sales from 2024 to 2025, despite the fact that various divisions had fluctuating sales statistics over time.

Step 2: Modifying the given dataset.

	2025	Sales			
	Quarter 1	Quarter 2	Quarter 3	Quarter 4	<i>totals</i>
Division I	\$8,365	\$8,628	\$8,661	\$9,234	\$34,888
Region A	\$4,500	\$4,700	\$4,800	\$4,900	\$18,900
Region B	\$3,865	\$3,928	\$3,861	\$3,934	\$15,588
Division II	\$8,428	\$8,147	\$9,224	\$7,811	\$33,610
Division III	\$7,765	\$8,749	\$8,734	\$9,491	\$34,739
Division IV	\$7,868	\$8,281	\$9,403	\$8,742	\$34,294
<i>totals</i>	\$32,426	\$33,805	\$36,022	\$35,278	\$137,531

Step 3: Loading SQL Statements.

With the given script we can see that we created 2 tables and populated them respectively.

For 2024 sales

The screenshot displays a SQL IDE interface. On the left, a 'Connections' pane shows a database named 'Harshal' with a list of tables including AUDIO, BACKUP_EMPLOY, COUNTRIES, CRAFTREGION, CRAFTSHOPITEMS, CRAFTSTORE, CUSTOMERS, EMPLOY, INVOICES, ORDER_ITEMS, ORDERS, PRODUCTS, SALES, SHIPPERS, SUPPLIERS, TBLODDATA, and TBLODDATA. The main workspace is titled 'Query Builder' and contains the following SQL script:

```
Insert into sales2025
Values ('d4', 9217, 7799,8902, 8884);

commit;

select * from sales2024;

select * from sales2025;
```

Below the script, the 'Query Result' pane shows the output of the last query, displaying 6 rows of data with columns DIVID, Q1, Q2, Q3, and Q4:

	DIVID	Q1	Q2	Q3	Q4
1	d1	8592	7591	9497	7909
2	d1a	5092	4090	5995	4406
3	d1b	3500	3501	3502	3503
4	d2	7564	8265	8503	8191
5	d3	8496	7720	7634	9120
6	d4	7753	8834	8603	9124

For 2025 sales

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays a tree view of the database schema, including tables like AUDIO, BACKUP_EMPLOY, COUNTRIES, CRAFTREGION, CRAFTSHOPITEMS, CRAFTSTORE, CUSTOMERS, EMPLOY, INVOICES, ORDER_ITEMS, ORDERS, PRODUCTS, SALES, SHIPPERS, SUPPLIERS, TBLNEWDATA, TBLOLDATA, and TEST_REIN. The right pane shows a query window with the following SQL code:

```
commit;

select * from sales2024;

select * from sales2025;
```

The Query Results pane shows the results of the query, displaying 6 rows of data with columns DIVID, Q1, Q2, Q3, and Q4.

	DIVID	Q1	Q2	Q3	Q4
1	d1	9145	7874	9138	9204
2	d1a	5119	4088	5880	4816
3	d1b	4033	3786	3251	4388
4	d2	8948	8478	8147	8697
5	d3	8100	6933	7740	8270
6	d4	9217	7799	8902	8884

Step 4: Performing BI procedures.

Row Totals for 2024

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays a tree view of the database schema. The right pane shows a query window with the following SQL code:

```
SELECT divID,
       SUM(q1 + q2 + q3 + q4) AS total_sales
FROM sales2024
GROUP BY divID;
```

The Query Results pane shows the results of the query, displaying 6 rows of data with columns DIVID and TOTAL_SALES.

	DIVID	TOTAL_SALES
1	d3	32970
2	d1	33589
3	d1a	19583
4	d4	34314
5	d1b	14006
6	d2	32523

Column Totals for 2025

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays a tree view of the database schema. The right pane shows a query window with the following SQL code:

```
SELECT
  SUM(q1) AS q1_total,
  SUM(q2) AS q2_total,
  SUM(q3) AS q3_total,
  SUM(q4) AS q4_total
FROM sales2025;
```

The Query Results pane shows the results of the query, displaying 1 row of data with columns Q1_TOTAL, Q2_TOTAL, Q3_TOTAL, and Q4_TOTAL.

	Q1_TOTAL	Q2_TOTAL	Q3_TOTAL	Q4_TOTAL
1	44562	40958	43058	44259

Drill-Down for 2024:

For Div 1

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays a tree view of the database schema. The right pane shows a query window with the following SQL code:

```
SELECT
  divID,
  q1,
  q2,
  q3,
  q4,
  q1 + q2 + q3 + q4 AS total_division_sales
FROM sales2024
WHERE divID = 'd1';
```

The Query Results pane shows the results of the query, displaying 1 row of data with columns DIVID, Q1, Q2, Q3, Q4, and TOTAL_DIVISION_SALES.

	DIVID	Q1	Q2	Q3	Q4	TOTAL_DIVISION_SALES
1	d1	8592	7591	9497	7909	33589

For Div 1 – a

Query Builder

```
SELECT
  divID,
  q1,
  q2,
  q3,
  q4,
  q1 + q2 + q3 + q4 AS total_region_sales
FROM sales2024
WHERE divID = 'dia';
```

Script Output x Query Result x All Rows Fetched: 1 in 0.019 seconds

DIVID	Q1	Q2	Q3	Q4	TOTAL_REGION_SALES
1 dia	5092	4090	5995	4406	19583

For Div 1 – b

Query Builder

```
SELECT
  divID,
  q1,
  q2,
  q3,
  q4,
  q1 + q2 + q3 + q4 AS total_region_sales
FROM sales2024
WHERE divID = 'dlb';
```

Script Output x Query Result x All Rows Fetched: 1 in 0.021 seconds

DIVID	Q1	Q2	Q3	Q4	TOTAL_REGION_SALES
1 dlb	3500	3501	3502	3503	14006

Hence, we can confirm that the quarterly sales are equal to that of sum of regional sales.

Roll-up for 2024

For total sales of Div 1 a and b

Query Builder

```
SELECT
  divID,
  SUM(q1) AS q1_total,
  SUM(q2) AS q2_total,
  SUM(q3) AS q3_total,
  SUM(q4) AS q4_total,
  SUM(q1 + q2 + q3 + q4) AS total_quarterly_sales,
  SUM(q1 + q2 + q3 + q4) AS total_region_sales
FROM sales2024
WHERE divID IN ('dia', 'dlb')
GROUP BY divID;
```

Script Output x Query Result x All Rows Fetched: 2 in 0.016 seconds

DIVID	Q1_TOTAL	Q2_TOTAL	Q3_TOTAL	Q4_TOTAL	TOTAL_REGION_SALES
1 dia	5092	4090	5995	4406	19583
2 dlb	3500	3501	3502	3503	14006

For total sales of Div 1 overall

Query Builder

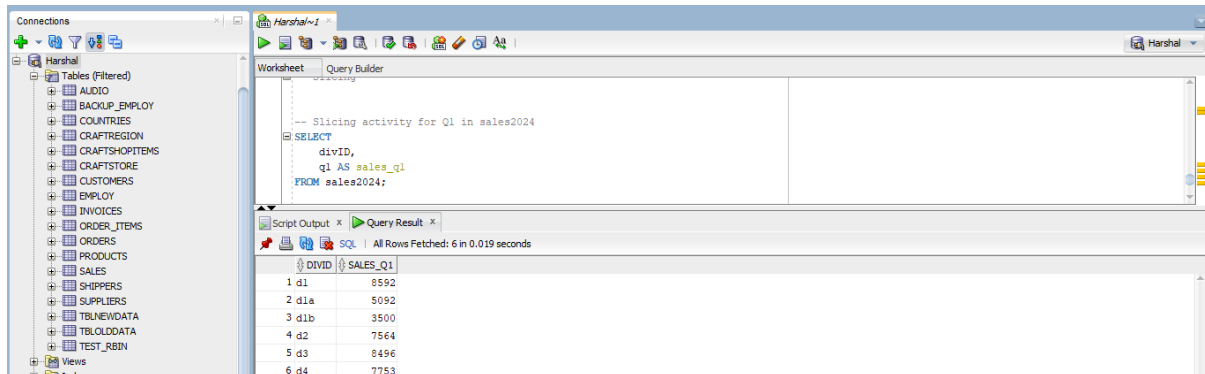
```
SELECT
  divID,
  SUM(q1) AS q1_total,
  SUM(q2) AS q2_total,
  SUM(q3) AS q3_total,
  SUM(q4) AS q4_total,
  SUM(q1 + q2 + q3 + q4) AS total_quarterly_sales,
  SUM(q1 + q2 + q3 + q4) AS total_region_sales
FROM sales2024
WHERE divID = 'd1';
```

Script Output x Query Result x All Rows Fetched: 1 in 0.019 seconds

DIVID	Q1_TOTAL	Q2_TOTAL	Q3_TOTAL	Q4_TOTAL	TOTAL_REGION_SALES
1 d1	8592	7591	9497	7909	33589

Slicing on Q1:

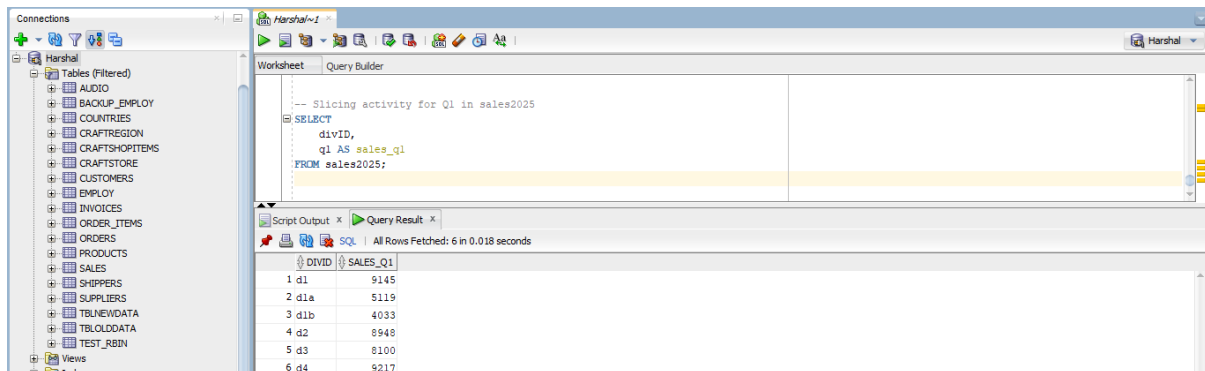
For 2024



The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays a tree view of the database schema, including tables like AUDIO, BACKUP_EMPLOY, COUNTRIES, CRAFTREGION, CRAFTSHOPITEMS, CRAFTSTORE, CUSTOMERS, EMPLOY, INVOICES, ORDER_ITEMS, ORDERS, PRODUCTS, SALES, SHIPPERS, SUPPLIERS, TBLNEWDATA, TBLOLDATA, and TEST_REBIN. The right pane shows the Query Builder with a query for Q1 sales in 2024. The query is:
-- Slicing activity for Q1 in sales2024
SELECT
divID,
q1 AS sales_q1
FROM sales2024;
The Query Result pane shows the following data:

DIVID	SALES_Q1
1 d1	8592
2 d1a	5092
3 d1b	3500
4 d2	7564
5 d3	8496
6 d4	7753

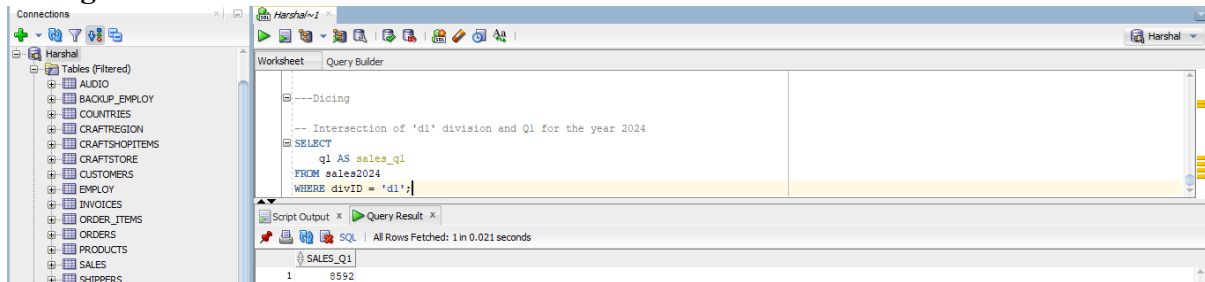
For 2025



The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays a tree view of the database schema. The right pane shows the Query Builder with a query for Q1 sales in 2025. The query is:
-- Slicing activity for Q1 in sales2025
SELECT
divID,
q1 AS sales_q1
FROM sales2025;
The Query Result pane shows the following data:

DIVID	SALES_Q1
1 d1	9145
2 d1a	5119
3 d1b	4033
4 d2	8940
5 d3	9100
6 d4	9217

Dicing:



The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays a tree view of the database schema. The right pane shows the Query Builder with a dicing query for Q1 sales in 2024. The query is:
--Dicing
-- Intersection of 'd1' division and Q1 for the year 2024
SELECT
q1 AS sales_q1
FROM sales2024
WHERE divID = 'd1';
The Query Result pane shows the following data:

SALES_Q1
1 8592

Step 5: Q and A

1)

One effective way to represent sales data across divisions, regions, districts, and offices is to use a multi-dimensional strategy, such as an OLAP cube or dimensional model. By storing pre-calculated aggregations and facilitating simple navigation across various hierarchy levels—from broad summaries to in-depth specifics—this structure guarantees quicker query performance. Users benefit from more flexibility in precisely generating hierarchical aggregations, comparing performance across several aspects at once, and assessing sales patterns. But careful preparation is necessary for the execution, taking into account things like business requirements and the complexity of the data. Nevertheless, a multi-dimensional

design greatly improves analytical insights by offering effective access to data at several levels, permitting smooth comparisons, and speeding up the examination of sales performance across several hierarchical categories.

2)

There are several benefits to arranging data in a cube inside an OLAP application:

Enhanced Query Processing: Cubes accelerate query replies, especially for complicated analytical questions, by precalculating and storing aggregated data. The computation time is greatly decreased by this pre-aggregation.

Multidimensional Analysis: Cubes let users work with data in several dimensions and hierarchies, making analysis more flexible. Without complex SQL queries, users may examine data from various perspectives, facilitating the effective extraction of insights.

Hierarchical Exploration: Cubes' hierarchical nature makes it easy to navigate between various data granularity levels. Users may easily explore more or enlarge across dimensions, examining information ranging from high-level overviews to detailed information.

Better Data Comprehension: Users can more easily understand data relationships since cubes graphically display the links between the data. Business users may more easily understand data trends and patterns by utilizing an intuitive interface that is created by grouping measurements and dimensions into cubes.

Scalability and Adaptability: Cubes can effectively manage massive amounts of data and change to meet evolving business requirements. Flexibility is ensured by adding new dimensions or metrics without affecting already-existing data.

Advanced calculations: Complex computations like as market share calculations and year-over-year comparisons are supported by OLAP cubes. End users' jobs might be made simpler by presetting these computations within the cube.

All things considered, cubes simplify data exploration, improve analytical capabilities, expedite query processing, and enable users to get valuable insights from intricate datasets in a framework that is both flexible and easy to use.

3)

The following actions would be performed in MS Excel to create the chart that is displayed, it is pretty easy:

- Decide which data, including the column headings, will be plotted.
- Next, choose the preferred chart type from the Charts group by clicking the Insert tab.
- Click Next, choose the Data Range tab, and confirm that the right data range is selected in the Chart Wizard dialog box.
- To adjust the chart as desired, select the Chart Options tab and click Finish.
- The necessary chart is prepared.

4)

The picture shows a chart that was made using data from sales reports by following a few steps:

1. Using Microsoft Excel to import data from a sales report into a spreadsheet.
2. Removing duplicates, fixing mistakes, and guaranteeing uniform formatting are all part of cleaning and formatting the data.

3. Creating a pivot table to compile and examine the sales information by quarter and division.
4. Creating a chart and choosing a column chart type for display using the data from the pivot table.
5. Adding axis labels for the divisions and quarters and personalizing the chart with a title ("Division Totals").
6. Improving the layout of the chart by adjusting the font size, colors, and decorative components.

Sales information is captured by division and quarter in the pivot table, with the overall sales shown in the "Values" column.

With the y-axis showing sales in dollars and the x-axis indicating quarters, the customizable chart shows total sales by division over the course of several quarters. "Division Totals" is the title of the chart, and "Quarter" and "Sales" are the labels on the axis.

5)

The following actions would be performed in MS Excel in order to create the chart in the image using the data from the sales report:

1. Import the data from the sales report into Excel.
2. To summarize the data by month and product, create a pivot table.
3. From the pivot table, insert a line chart.
4. Tailor the diagram.

Step 1: Open Excel and import the sales report data

You may use the Data > Get Data > From File > From Text/CSV command or copy and paste the data from the source file to import the sales report data into Excel.

Step 2: To summarize the data by month and product, use a pivot table.

Choose the data from the sales report, then click the Insert > Table > PivotTable button to create a pivot table. After choosing the Existing Worksheet option in the Create PivotTable dialog box, click the OK button.

Drag the Month field to the Rows area and the Product field to the Columns area of the PivotTable Fields window. Move the Values field over to the Sales field.

Step 3: From the pivot table, insert a line chart.

After choosing the pivot table data, click the Insert > Chart option to insert a line chart from the table. Choose the Line chart type from the Insert Chart dialog box and click OK.

Step 4: Make the chart unique

The chart title, axis labels, and other formatting choices can be changed once the line chart has been added.

For instance:

This is an illustration of a pivot table that was made to compile sales report data according to month and product:

Here is an illustration of a line chart made using the pivot table:

The graph displays the monthly totals for all of the products. Months are displayed on the x-axis, and sales in dollars are displayed on the y-axis. "Month" and "Sales" are the labels on the axis, while the title of the chart is "Product Sales by Month."