

---

# **CSI 4142: Phase 3 Deliverable**

OLAP Queries & BI Dashboard

---

<https://github.com/c-stev/CSI-4142-Project>

Cole Stevens (300171413)

William Beaupre (300174392)

Emiliano Bustamante (300229811)

# 1 Standard OLAP Operations

## a. Drill down and roll up

*Note: since our data is already the lowest-possible granularity, we could not perform any drill-down operations to analyze a section of data. To compensate, we've included two roll up queries instead.*

### Explanation

The following query rolls up from day to month, and views the dim\_financial data info as aggregates (e.g., the high is now the highest high of the month, the low is the lowest low of the month, etc.). It is used to monitor the activity for different companies.

### SQL

```
SELECT fd.ticker, dd.year, dd.month, AVG(fd.open) AS avg_open, AVG(fd.close) AS
avg_close,
        MAX(fd.high) AS max_high, MIN(fd.low) AS min_low, SUM(fd.volume) AS
total_volume
FROM fact_stock_analysis fsv
JOIN dim_financial fd ON fsv.financial_data_id = fd.financial_data_id
JOIN dim_date dd ON fsv.date_id = dd.date_id
GROUP BY fd.ticker, dd.year, dd.month;
```

### Output

Data Output Messages Notifications										
	ticker character varying (5)	year integer	month integer	avg_open double precision	avg_close double precision	max_high double precision	min_low double precision	total_volume numeric		
17	A	2001	5	22.93568161772074	22.93816375732422	24.98473918893457	19.56672680901672	66941412		
18	A	2001	6	19.198362996436984	19.287059602283296	22.11495595274612	16.53312536778465	65128346		
19	A	2001	7	17.93754291402874	17.887849626087007	20.32513076152064	15.896077753503953	48483900		
20	A	2001	8	17.134308441135317	17.155148091523543	19.83975708081718	14.743306304667032	54079674		
21	A	2001	9	13.187672052153783	13.222457377115886	16.351112092245074	10.920964751394916	46768693		
22	A	2001	10	13.503484217212822	13.687346748683764	15.040599365663743	11.3578020167263	54138807		
23	A	2001	11	14.983102227512049	15.218278430757069	16.73941952040729	13.384249039301729	103317793		
24	A	2001	12	17.41347871926768	17.523295402526855	18.729453537226416	16.32077967208375	68254415		
25	A	2002	1	18.177917548087514	18.274414970761253	20.20378646076515	16.332905849740424	63783748		
26	A	2002	2	17.01882506406216	16.95911111329731	19.46965788790478	15.06486260149314	62379040		

## Explanation

The following query rolls up country data to view yearly data for each country rather than daily data. It is used to monitor the change for each country's population, gdp, employment, etc.

## SQL

```
SELECT code AS country_code,  
       country,  
       EXTRACT(year FROM date) AS year,  
       SUM(population) AS total_population,  
       AVG(gdp) AS average_gdp,  
       AVG(inflation) AS average_inflation,  
       AVG(employment) AS average_employment,  
       AVG(unemployment) AS average_unemployment  
FROM dim_country  
GROUP BY code, country, EXTRACT(year FROM date)  
ORDER BY code, year;
```

## Output

	country_code character varying (4)	country character varying (64)	year numeric	total_population bigint	average_gdp double precision	average_inflation double precision	average_employment double precision	average_unemployment double precision
17	CH	Switzerland	2016	3078968141	691538166977.9615	0.04825956284153002	65.17555737704927	4.860163934426228
18	CH	Switzerland	2017	3096294416	710343175435.6317	0.734512328767123	65.20486301369857	4.755123287671233
19	CH	Switzerland	2018	3118822985	723474667980.5697	0.6504027397260272	65.24502739726067	4.550438356164383
20	CH	Switzerland	2019	3141422452	731655998690.4509	-0.17999999999999998	64.9178712328767	4.60441095890411

## b. Slice

## Explanation

The following query obtains the record of the United States' GDP movement. It is used to monitor activity.

## SQL

```
SELECT cd.country, cd.date, cd.gdp  
FROM dim_country cd  
WHERE cd.code = 'US'
```

## Output

	country character varying (64)	date date	gdp double precision
1	United States	2000-01-01	10250947997000
2	United States	2000-01-02	10251852318795.082
3	United States	2000-01-03	10252756640590.164
4	United States	2000-01-04	10253660962385.246
5	United States	2000-01-05	10254555084100.328

## c. Dice

### Explanation

The following function obtains a cube of all dates for Apple (AAPL) in 2015 where the closing price is higher than the opening price. It is used to monitor activity.

## SQL

```
SELECT fd.date, cd.company
FROM fact_stock_analysis fsv
JOIN dim_financial fd ON fd.financial_data_id = fsv.financial_data_id
JOIN dim_company cd ON cd.company_id = fsv.company_id
WHERE cd.ticker = 'AAPL' AND (fd.open - fd.close > 0)
```

## Output

	date date	company character varying (64)
1	2000-01-04	Apple Inc.
2	2000-01-06	Apple Inc.
3	2000-01-10	Apple Inc.
4	2000-01-11	Apple Inc.
5	2000-01-12	Apple Inc.

## Explanation

The following query extracts the financial transaction information for every US-based Industrials company in the 3rd quarter of 2005. It is used to monitor activity.

## SQL








```
SELECT fd.ticker, fd.date, fd.open, fd.close, fd.high, fd.low, fd.volume
FROM fact_stock_analysis fsv
JOIN dim_financial fd ON fd.financial_data_id = fsv.financial_data_id
JOIN dim_company cd ON cd.company_id = fsv.company_id
JOIN dim_country cod ON cod.country_id = fsv.country_id
JOIN dim_date dd ON dd.date_id = fsv.date_id
WHERE cd.sector = 'Industrials' AND cod.code = 'US' AND dd.year = 2005 AND
dd.quarter = 3;
```

## Output

Data Output

Messages

Notifications

	ticker character varying (5) 	date date 	open double precision 	close double precision 	high double precision 	low double precision 	volume bigint 
1	MMM	2005-07-01	41.20006941653443	41.59321212768555	41.66158609610324	40.47076447474713	8866000
2	AOS	2005-07-01	3.2922797766352074	3.329230308532715	3.339083384334144	3.2910478338219398	984000
3	AYI	2005-07-01	18.59243120097791	18.52729606628418	18.59966859154289	18.25228074637929	127500
4	ALK	2005-07-01	6.72197718706626	6.679262161254883	6.780429214807845	6.614065949143862	914000
5	AME	2005-07-01	10.999666641113752	11.123169898986816	11.20462974653781	10.96550584161851	1127250

## d. Combining OLAP operations

## Explanation

The following query rolls up to the monthly level, and lists the most profitable month/year combos for companies in the Health Care sector. It is used to monitor activity.

## SQL

```
SELECT dd.year, dd.month_string AS month, SUM(fsa.returns) AS total_returns
FROM fact_stock_analysis fsa
```

```

JOIN dim_date dd ON fsa.date_id = dd.date_id
JOIN dim_company cd ON fsa.company_id = cd.company_id
WHERE cd.sector = 'Health Care'
GROUP BY dd.year, dd.month_string
ORDER BY total_returns DESC;

```

## Output

Data Output Messages Notifications			
	year integer	month character varying (32)	total_returns double precision
1	2021	September	1055.4117289164287
2	2018	October	770.3272513371296
3	2018	December	641.527966961085
4	2021	November	534.8351016477072
5	2021	February	495.9305554526693

## Explanation

The following query rolls up to the year level, and extracts the average yearly volatility for all US-based companies. It is used to monitor activity.

## SQL

```

SELECT cd.country, dd.year, AVG(fsa.volatility)
FROM fact_stock_analysis fsa
JOIN dim_country cd ON fsa.country_id = cd.country_id
JOIN dim_date dd ON fsa.date_id = dd.date_id
WHERE cd.code = 'US'
GROUP BY cd.country, dd.year

```

## Output

Data Output Messages Notifications			
	country character varying (64)	year integer	avg double precision
1	United States	2000	0.02579242891317247
2	United States	2001	0.021978679167251835
3	United States	2002	0.022095822089804103
4	United States	2003	0.015663401909658814
5	United States	2004	0.01307435184363344
6	United States	2005	0.012489741978309753
7	United States	2006	0.012849197412085584
8	United States	2007	0.014745016949459124

## Explanation

The following query extracts the number of 'good days' (days where the daily returns were above 0) for all stocks that belong to the Industrials, Materials, and Telecommunication Services sectors. It is used to monitor activity.

## SQL

```
SELECT cd.sector, COUNT(DISTINCT dd.date_id) good_days
FROM fact_stock_analysis fsa
JOIN dim_company cd ON fsa.company_id = cd.company_id
JOIN dim_date dd ON fsa.date_id = dd.date_id
WHERE fsa.returns > 0 AND cd.sector IN ('Industrials', 'Materials',
'Telecommunication Services')
GROUP BY cd.sector
```

## Output

Data Output Messages Notifications		
	sector character varying (64)	good_days bigint
1	Industrials	5507
2	Materials	5409
3	Telecommunication Services	3488

## Explanation

The following query rolls up from company to sector and extracts all sectors that have had a positive return year in 2018. It is used to monitor activity.

## SQL

```
SELECT cd.sector, SUM(fsa.returns) AS sector_returns
FROM fact_stock_analysis fsa
JOIN dim_date dd ON fsa.date_id = dd.date_id
JOIN dim_company cd ON fsa.company_id = cd.company_id
WHERE dd.year = 2018
GROUP BY cd.sector
HAVING SUM(fsa.returns) > 0;
```

## Output

	sector character varying (64)	sector_returns double precision
1	Consumer Discretionary	1144.0983662751705
2	Consumer Staples	232.14784236315856
3	Energy	424.92983718431367
4	Financials	1238.9279078670688
5	Health Care	486.93363767501694

# 2 Explorative Operations

## a. Iceberg queries

## Explanation

The following iceberg query retrieves the top 10 performing companies (in terms of returns) on January 3rd, 2000. This query could be used to aid potential investors in making more informed decisions or conducting analysis on investment opportunities. It is used to monitor activity as well as potentially identifying potential outliers (if a particular company vastly outperforms every other company).



## SQL

```
SELECT cd.company, fd.ticker, fsa.returns
FROM fact_stock_analysis fsa
JOIN dim_financial fd ON fsa.financial_data_id = fd.financial_data_id
JOIN dim_date dd ON fsa.date_id = dd.date_id
JOIN dim_company cd ON fsa.company_id = cd.company_id
WHERE dd.day = 3 AND dd.month = 1 AND dd.year = 2000
ORDER BY fsa.returns DESC
LIMIT 10;
```

## Output

	company character varying (64)	ticker character varying (5)	returns double precision
1	American International Group, Inc.	AIG	25.99873493173061
2	Akamai Technologies Inc	AKAM	22.9375
3	Verisign Inc.	VRSN	10.932820537700053
4	Citigroup Inc.	C	10.865924943168267
5	QUALCOMM Inc.	QCOM	6.405369421878149
6	Amgen Inc	AMGN	5.077048122705271
7	Goldman Sachs Group	GS	4.236131353574024
8	Agilent Technologies Inc	A	4.095359062567759
9	General Electric	GE	3.2884986221779116
10	Mettler Toledo	MTD	2.625

## b. Windowing queries

### Explanation

The following query compares each company's volatility with the average volatility in its sector and provides a rank. This query provides researchers with the ability to identify the relative volatility of companies within their respective sectors, and allow for wiser investment choices to be made. It is used to monitor activity.

## SQL

```
SELECT sector, company, ticker, avg_volatility,
       AVG(avg_volatility) OVER (PARTITION BY sector) AS avg_sector_volatility,
```

```

RANK() OVER (PARTITION BY sector ORDER BY avg_volatility DESC) AS
volatility_rank
FROM (
    SELECT cd.sector, cd.company, fd.ticker,
    AVG(fsa.volatility) AS avg_volatility
    FROM fact_stock_analysis fsa
    JOIN dim_financial fd ON fsa.financial_data_id = fd.financial_data_id
    JOIN dim_date dd ON fsa.date_id = dd.date_id
    JOIN dim_company cd ON fsa.company_id = cd.company_id
    GROUP BY cd.sector, cd.company, fd.ticker
);

```

## Output

	sector character varying (64)	company character varying (64)	ticker character varying (5)	avg_volatility double precision	avg_sector_volatility double precision	volatility_rank bigint
1	Consumer Discretionary	Goodyear Tire & Rubber	GT	0.023006068673811525	0.01708198861900384	1
2	Consumer Discretionary	Under Armour Class A	UAA	0.022840895128027294	0.01708198861900384	2
3	Consumer Discretionary	Pulte Homes Inc.	PHM	0.0224458143887764	0.01708198861900384	3
4	Consumer Discretionary	Wynn Resorts Ltd	WYNN	0.022417843744846577	0.01708198861900384	4
5	Consumer Discretionary	D. R. Horton	DHI	0.02233154597093887	0.01708198861900384	5

## c. Using the Window clause

### Explanation

The following query calculates the average return over a rolling window for each company's financial data. This particular query could be used in analyzing the short-term performance of each company's stock returns. It is used to monitor activity.

### SQL

```

SELECT cd.company, fd.ticker, fd.date,
AVG(fsa.returns) OVER(PARTITION BY cd.company ORDER BY fd.date ROWS BETWEEN 29
PRECEDING AND 29 FOLLOWING) as avg_return_30d
FROM dim_financial fd
JOIN fact_stock_analysis fsa on fsa.financial_data_id = fd.financial_data_id

```

## Output

Data Output

Messages

Notifications

	<div>company</div> <div>character varying (64)</div>	<div>ticker</div> <div>character varying (5)</div>	<div>date</div> <div>date</div>	<div>avg_return_30d</div> <div>double precision</div>
1	3M Company	MMM	2000-01-03	0.15728912353921712
2	3M Company	MMM	2000-01-04	0.1081394908726696
3	3M Company	MMM	2000-01-05	0.10525662531276725
4	3M Company	MMM	2000-01-06	0.09725254763571922
5	3M Company	MMM	2000-01-07	0.11168183318365872

### 3 BI Dashboard and Information Visualization

## Page 1: Top-N / Bottom-N Returns

## Companies by return

### Top 10 companies by sum of returns in millions

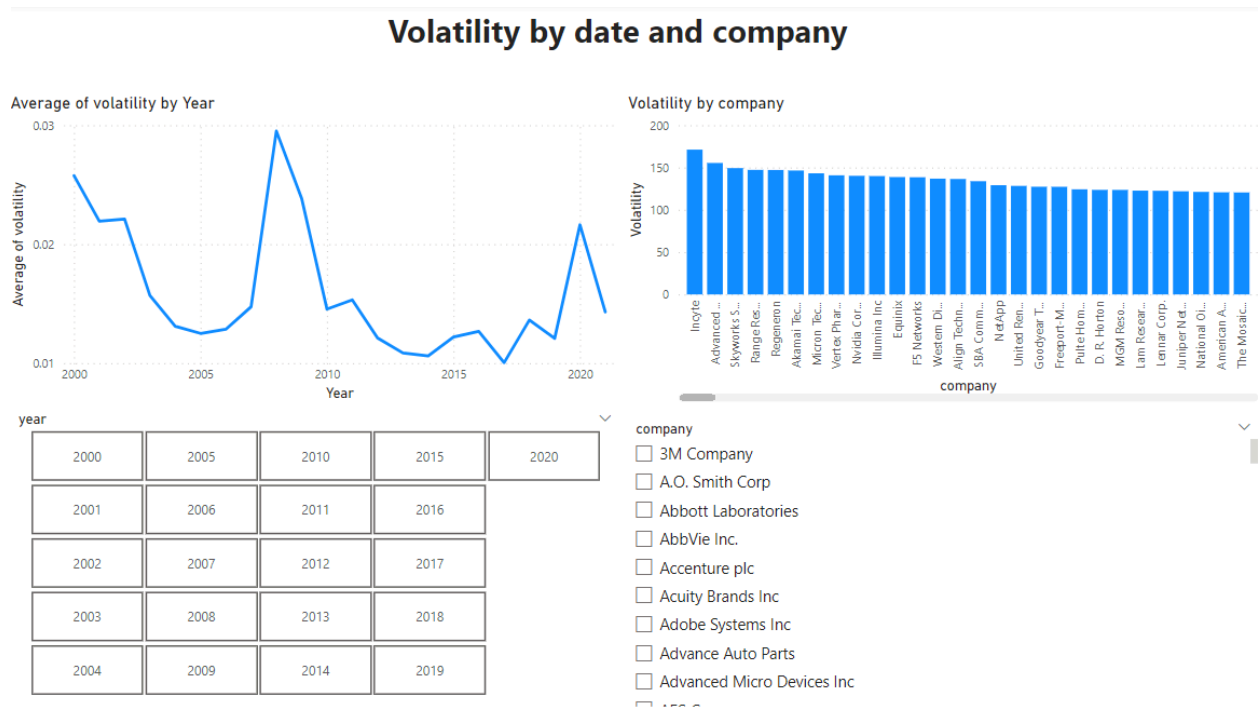
company	Sum of returns
American International Group, Inc.	1,275.16
Citigroup Inc.	768.95
Equinix	567.13
Akamai Technologies Inc	501.60
General Electric	282.45
Regeneron	275.02
Wynn Resorts Ltd	253.68
Boeing Company	210.81
Micron Technology	181.62
PVH Corp.	172.55
<b>Total</b>	<b>4,488.97</b>

### Bottom 10 companies by sum of returns in millions

company	Sum of returns
Intuit Inc.	-268.64
Grainger (W.W.) Inc.	-288.12
ANSYS	-292.77
Accenture plc	-331.41
Costco Wholesale Corp.	-347.64
IDEXX Laboratories	-398.60
Charter Communications	-426.21
O'Reilly Automotive	-455.37
AutoZone Inc	-570.07
Mettler Toledo	-832.15
<b>Total</b>	<b>-4,210.97</b>

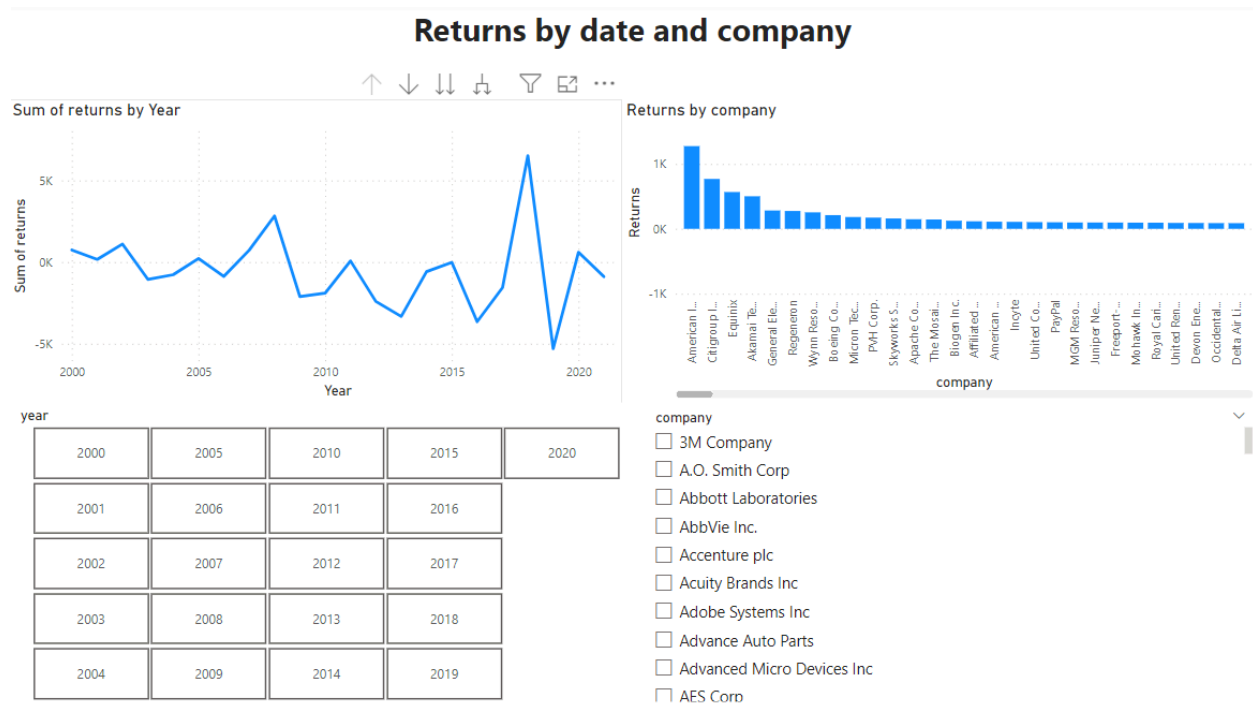
The page above permits users to visualize the top and bottom 10 companies in terms of total returns.

## Page 2: Volatility by Date & Country



The page above permits users to view the average volatility for each company. The page permits cubing operations, by allowing users to filter by year and companies. Furthermore concept hierarchy traversal is permitted through the top left line chart; the user can select whether they wish to view the data by day, month, quarter, or year.

## Page 3: Returns by Date and Company



The page above permits users to view the total returns for each company. The page permits cubing operations, by allowing users to filter by year and companies. Furthermore concept hierarchy traversal is permitted through the top left line chart; the user can select whether they wish to view the data by day, month, quarter, or year.

Page 4: Volume by Country

