SQL for Business Intelligence

Turning Data into Insights

0. Table of Contents

1. Introduction

1.1 Question:

```
        What kind of world we live in?
```

1.2 Answer:

```
        1. In the data-driven world we live in,

        2. businesses are constantly collecting vast amounts of data.
```

1.3 BUT

```
But what’s the use of all that data if
you can’t turn it into actionable insights?
```

```
This is where Business Intelligence (BI)
comes into play.
```

```
At the core of BI is SQL (Structured Query
Language), a powerful tool for querying and
analyzing data. In this article, we’ll explore
how SQL is used for Business Intelligence
with practical code examples.
```

2. What is a Business Intelligence?

1. What is business intelligence?

   ```
   Business intelligence combines
       -- business analytics,
       -- data mining,
       -- data visualization,
       -- data tools and infrastructure, and
       -- best practices
   to help organizations make more
   data-driven decisions.
   ```

2. What Is Business Intelligence (BI)?

```
Business intelligence refers to the
technology that enables businesses to

    1. organize, analyze, and

    2. contextualize business data

    from around the company.

BI includes

   1. multiple tools and techniques

   2. to transform raw data into

   3. meaningful and actionable information.
```

2.1 Examples of Business Intelligence

1. Why are sales dropping in a particular region?
2. What factors contribute to high profits in a specific year?
3. Which employee is performing better in this quarter?
4. What are the top-10 products sold in California
5. How many orders were NOT FILLED IN last week?
6. What was the totals sales of Nike shoes in 58 counties of California for the past 3 years?
7. Which are the top-5 states for selling Electric bikes?

2.2 Business Intelligence Main Parts

BI systems have 4 main parts:

1. A data warehouse stores company information from a variety of sources in a centralized and accessible location.
2. Business analytics or data management tools mine and analyze data in the data warehouse.
3. Business performance management (BPM) tools monitor and analyze progress towards business goals.
4. A user interface (usually an interactive dashboard with data visualization reporting tools) provides quick access the information.

2.3 Companies using Business INtelligence

- Starbucks

```
BI lets them provide loyalty card
programs and a mobile app to track
individual purchase data. It also
helps predict purchases and send
personalized offers via app or email
to entice customers and boost sales.
```

```
Starbucks carefully analyses a number of
data points to identify what might make
a successful store location. This includes:
```

1. Consumer demographics
2. Population density
3. Average income levels
4. Traffic patterns
5. Public transport hubs
6. Types of businesses in the location under consideration.

- Netflix

```
As the largest subscription streaming service,
Netflix leverages business intelligence and
data science to gain new subscribers and enhance
user experiences.
```

```
BI solutions create personalized watch lists and
compelling artwork while persuading users to watch
their favorite genres. The recommendation system
promotes content to the right audience at the right time.
```

- Uber

```
This California-based transportation company uses
business intelligence to monitor traffic conditions,
journey times and durations, driver availability and
customer demand in real time.
```

```
It leverages surge pricing and a rating system as a
part of its business model to manage fares and maintain
trust between drivers and customers.
```

- Walmart

  ```
  Walmart uses business intelligence to gauge customer
  behavior online and in-store based on customer purchases.
  It helps them make data-driven decisions around assortments,
  inventory and merchandising, resources, promotions and more
  to drive profits. Data-based apps shine a light on customer
  preferences and curate specific loyalty programs for
  different segments.
  ```

- Amazon

  ```
  Amazon is a retail giant with more than 100 million
  prime customers worldwide and over 12 million products
  for sale.
  ```

  ```
  The global marketplace is all about concentrating huge
  amounts of data that should be processed quickly and
  effectively to ensure all personalized user recommendations,
  targeted marketing, and pricing optimizations are up to
  date. All these analytic efforts result in the extended
  collection of various metrics about every customer and help
  bring the user experience to such a new, high-quality level
  that makes Amazon a leading e-retailer in the US.
  ```

  ```
  Amazon uses BI systems to optimize its logistics and
  streamline the workflow of the supply chain. Utilizing
  modern BI tools, Amazon speeds up order deliveries
  thanks to predictive analysis, choosing the best shipping
  schedules, routes, and groupings of products.
  ```

2.4 What are the Techniques for Business Intelligence

1. **Analytics**:

   ```
   Extracts and identifies trends,
   patterns and correlations from
   historical data to drive decisions.
   ```

2. **Dashboards**:

```
Collects and displays intuitive
visualizations, KPIs and metrics
to monitor business performance.
```

3. **Data Mining**:

```
Employs statistical and machine
learning techniques to unearth anomalies,
patterns and correlations within large
datasets to predict outcomes.
```

4. **ETL**:

```
Extracts data from multiple sources,
cleans it to improve quality and loads
it into systems for access and manipulation.
-> Build a Data Warehouse
```

5. **OLAP**:

```
Performs complex analytical queries
on Data Warehouses to solve problems
from different perspectives.
```

6. **Predictive Modeling**:

```
Uses machine learning and data mining
techniques to predict future outcomes
by analyzing historical data.
```

3. Introduction to SQL in BI

```
> Business Intelligence (BI) involves the use
of data analysis tools and techniques to transform
raw data into valuable insights.
```

```
> SQL, as a querying language, plays a pivotal role
in BI.
```

```
> SQL allows BI professionals to interact with
```

```
databases, retrieve relevant data, and perform
various data transformations.
```

4. Setting up Your BI Environment

```
Before we dive into SQL for BI, you need
the right tools and environment. Typically,
this involves setting up a data warehouse or
connecting to existing databases.
```

```
Popular BI tools like Tableau, Power BI, and
Looker often integrate seamlessly with SQL
databases.
```

4.1 Code Example: Connecting to a Database

```sql
-- Connect to a SQL Server database
USE employee_db;

-- Select data from a table
SELECT FirstName, LastName
  FROM Employees
    WHERE Department = 'Sales';
```

4.2 Code Example: Find Number of Employees per Department

```sql
-- Connect to a SQL Server database
USE employee_db;

-- Select data from a table and Group them
SELECT Department, count(*) as total
  FROM Employees
    GROUP BY Department;
```

5. Basic SQL Queries for BI

5.1 SELECT Statements

```
The SELECT statement is the workhorse
of SQL. It allows you to retrieve data
from one or more tables.
```

```
In BI, this is the starting point for
fetching data.
```

5.2 Filtering Data

```
Filtering data is crucial for BI.
You want to focus on the data that’s
relevant to your analysis.
```

```
SQL’s WHERE clause helps you do that.
```

5.3 Aggregating Data

```
Aggregation functions like SUM , AVG , and
COUNT are essential for summarizing data.
```

```
These functions are used extensively in creating
BI reports.
```

5.4 Code Example: Basic SQL Queries

- Retrieve all product names and their prices

```
SELECT ProductName, Price
  FROM Products;
```

- Filter orders for a specific customer

```
SELECT OrderID, OrderDate
  FROM Orders
     WHERE CustomerID = 'C2007';
```

- Calculate total sales for each year

```
SELECT YEAR(OrderDate) AS Year,
       SUM(TotalAmount) AS TotalSales
  FROM Orders
     GROUP BY YEAR(OrderDate);
```

6. Advanced SQL Techniques for BI

6.1 JOIN Operations

```
In BI, data often resides in multiple
tables.  SQL□s JOIN operations (e.g.,
INNER JOIN, LEFT JOIN, ... )
allow you to combine data from different
tables.
```

```
       -- books:  (book_id, book_title, author_id)
       -- authors:(author_id, first_name, last_name)
       --
       SELECT b.book_id, b.book_title, a.first_name, a.last_name
          FROM books b
             INNER JOIN authors a  ON b.author_id = a.author_id
             ORDER BY b.book_id;
```

6.2 Subqueries

```
Subqueries (or nested queries) are powerful
tools for BI. They let you create complex
queries step by step.
```

```
    SELECT * FROM CUSTOMERS
       WHERE ID IN (
                    SELECT ID
                       FROM CUSTOMER_DETAILS
                          WHERE SALARY > 4500
                    );
```

6.3 GROUP BY year WITH ROLLUP;

```
mysql> SELECT * from product_sales;
+------+---------+---------+--------+
| year | country | product | profit |
+------+---------+---------+--------+
| 2000 | USA     | PC      |    700 |
| 2000 | USA     | IPAD    |    800 |
| 2000 | USA     | MAC     |    900 |
| 2000 | CANADA  | PC      |    900 |
| 2000 | CANADA  | IPAD    |    300 |
| 2001 | USA     | PC      |    200 |
| 2001 | USA     | IPAD    |    300 |
| 2001 | USA     | MAC     |    400 |
| 2001 | CANADA  | PC      |    600 |
| 2001 | CANADA  | IPAD    |    400 |
+------+---------+---------+--------+
10 rows in set (0.00 sec)
```

```
mysql> SELECT year, country, SUM(profit) AS profit
    ->        FROM product_sales
    ->        GROUP BY year, country;
+------+---------+--------+
| year | country | profit |
+------+---------+--------+
| 2000 | USA     |   2400 |
| 2000 | CANADA  |   1200 |
| 2001 | USA     |    900 |
| 2001 | CANADA  |   1000 |
+------+---------+--------+
4 rows in set (0.00 sec)
```

```
mysql> SELECT year, country, SUM(profit) AS profit
    ->        FROM product_sales
    ->        GROUP BY year, country with ROLLUP
    ->   order by year;
+------+---------+--------+
| year | country | profit |
+------+---------+--------+
| NULL | NULL    |   5500 | <-- 5500 = 2400+1200+900+1000
| 2000 | CANADA  |   1200 |
| 2000 | USA     |   2400 |
| 2000 | NULL    |   3600 | <-- 3600 = 2400+1200
| 2001 | CANADA  |   1000 |
| 2001 | USA     |    900 |
| 2001 | NULL    |   1900 | <-- 1900 = 1000+900
+------+---------+--------+
7 rows in set (0.01 sec)

-- NOTE: The NULL value in the (year, country) columns
--       identifies the grand total super-aggregate line.
```

6.4 GROUP by CUBE

```
       GROUP BY CUBE is an extension of the GROUP BY
       clause similar to GROUP BY ROLLUP. In addition
       to producing all the rows of a GROUP BY ROLLUP,
       GROUP BY CUBE adds all the □cross-tabulations□
       rows. Sub-total rows are rows that further aggregate
       whose values are derived by computing the same
       aggregate functions that were used to produce
       the grouped rows.
```

[6315 group by cube rollup grouping sets sql server.006 2] | *https://www.mssqltips.com/tipimages2/6315_group-by-cube-rollup-grouping-sets-sql-server.006-2.png*

```
        A CUBE grouping is equivalent to a series of
        grouping sets and is essentially a shorter
        specification. The N elements of a CUBE specification
        correspond to 2^N GROUPING SETS.
```

[1*Gq5Ec6OsqrI0UTgdxAhemQ] | *https://miro.medium.com/v2/resize:fit:1093/1*Gq5Ec6OsqrI0UTgdxAhemQ.png*

Snowflake Examples

```
-- Create some tables and insert some rows.
CREATE TABLE products (product_ID INTEGER, wholesale_price REAL);

INSERT INTO products (product_ID, wholesale_price) VALUES
    (1, 1.00),
    (2, 2.00);

CREATE TABLE sales (product_ID INTEGER, retail_price REAL,
    quantity INTEGER, city VARCHAR, state VARCHAR);
INSERT INTO sales (product_id, retail_price, quantity, city, state) VALUES
    (1, 2.00,  1, 'SF', 'CA'),
    (1, 2.00,  2, 'SJ', 'CA'),
    (2, 5.00,  4, 'SF', 'CA'),
    (2, 5.00,  8, 'SJ', 'CA'),
    (2, 5.00, 16, 'Miami', 'FL'),
    (2, 5.00, 32, 'Orlando', 'FL'),
    (2, 5.00, 64, 'SJ', 'PR');
```

```
Run a cube query that shows profit by city, state,
and total across all states. The example below shows
a query that has three “levels”:
```

- Each city.
- Each state.
- All revenue combined.

```
This example uses ORDER BY state, city NULLS LAST to
ensure that each state’s rollup comes immediately after
all of the cities in that state, and that the final
rollup appears at the end of the output.
```

```
SELECT state, city,
       SUM((s.retail_price - p.wholesale_price) * s.quantity) AS profit
 FROM products AS p, sales AS s
 WHERE s.product_ID = p.product_ID
 GROUP BY CUBE (state, city)
 ORDER BY state, city NULLS LAST;


+-------+---------+--------+
| STATE | CITY    | PROFIT |
|-------+---------+--------|
| CA    | SF      |     13 |
| CA    | SJ      |     26 |
| CA    | NULL    |     39 |
| FL    | Miami   |     48 |
| FL    | Orlando |     96 |
| FL    | NULL    |    144 |
| PR    | SJ      |    192 |
| PR    | NULL    |    192 |
| NULL  | Miami   |     48 |
| NULL  | Orlando |     96 |
| NULL  | SF      |     13 |
| NULL  | SJ      |    218 |
| NULL  | NULL    |    375 |
+-------+---------+--------+
```

6.4 Window Functions

```
Window functions provide a way to perform
calculations across a set of rows related
to the current row.
```

```
This is extremely valuable for BI analytics.
```

Unlike aggregate functions, window functions do not collapse rows.

[aggregate vs window functions] | *https://learnsql.com/blog/sql-window-functions-cheat-sheet/aggregate-vs-window-functions.png*

6.5 Window Function Concepts and Syntax

```
This section describes how to use window functions.
Examples use the same sales information data set.
```

```
mysql> SELECT *
           FROM sales
               ORDER BY country, year, product;
+------+---------+------------+--------+
| year | country | product    | profit |
+------+---------+------------+--------+
| 2000 | Finland | Computer   |   1500 |
| 2000 | Finland | Phone      |    100 |
| 2001 | Finland | Phone      |     10 |
| 2000 | India   | Calculator |     75 |
| 2000 | India   | Calculator |     75 |
| 2000 | India   | Computer   |   1200 |
| 2000 | USA     | Calculator |     75 |
| 2000 | USA     | Computer   |   1500 |
| 2001 | USA     | Calculator |     50 |
| 2001 | USA     | Computer   |   1500 |
| 2001 | USA     | Computer   |   1200 |
| 2001 | USA     | TV         |    150 |
| 2001 | USA     | TV         |    100 |
+------+---------+------------+--------+
```

```
A window function performs an aggregate-like
operation on a set of query rows. However, whereas
an aggregate operation groups query rows into a
single result row, a window function produces
```

```
a result for each query row:
```

```
The row for which function evaluation occurs
is called the current row.
```

```
The query rows related to the current row over
which function evaluation occurs comprise the
window for the current row.
```

```
For example, using the sales information table,
these two queries perform aggregate operations
that produce a single global sum for all rows
taken as a group, and sums grouped per country:
```

```
mysql> SELECT SUM(profit) AS total_profit
       FROM sales;
+--------------+
| total_profit |
+--------------+
|         7535 |
+--------------+
```

```
mysql> SELECT country, SUM(profit) AS country_profit
       FROM sales
       GROUP BY country
       ORDER BY country;
+---------+----------------+
| country | country_profit |
+---------+----------------+
| Finland |           1610 |
| India   |           1350 |
| USA     |           4575 |
+---------+----------------+
```

```
By contrast, window operations do NOT collapse
groups of query rows to a single output row.
Instead, they produce a result for each row.
Like the preceding queries, the following query
uses SUM(), but this time as a window function:
```

```
mysql> SELECT
         year, country, product, profit,
         SUM(profit) OVER() AS total_profit,
         SUM(profit) OVER(PARTITION BY country) AS country_profit
       FROM sales
       ORDER BY country, year, product, profit;
+------+---------+------------+--------+--------------+----------------+
| year | country | product    | profit | total_profit | country_profit |
+------+---------+------------+--------+--------------+----------------+
| 2000 | Finland | Computer   |   1500 |         7535 |           1610 |
| 2000 | Finland | Phone      |    100 |         7535 |           1610 |
| 2001 | Finland | Phone      |     10 |         7535 |           1610 |
| 2000 | India   | Calculator |     75 |         7535 |           1350 |
| 2000 | India   | Calculator |     75 |         7535 |           1350 |
| 2000 | India   | Computer   |   1200 |         7535 |           1350 |
| 2000 | USA     | Calculator |     75 |         7535 |           4575 |
| 2000 | USA     | Computer   |   1500 |         7535 |           4575 |
| 2001 | USA     | Calculator |     50 |         7535 |           4575 |
| 2001 | USA     | Computer   |   1200 |         7535 |           4575 |
| 2001 | USA     | Computer   |   1500 |         7535 |           4575 |
| 2001 | USA     | TV         |    100 |         7535 |           4575 |
| 2001 | USA     | TV         |    150 |         7535 |           4575 |
+------+---------+------------+--------+--------------+----------------+
```

```
Each window operation in the query is signified
by inclusion of an OVER clause that specifies how
to partition query rows into groups for processing
by the window function:
```

```
The first OVER clause is empty, which treats the
entire set of query rows as a single partition.
The window function thus produces a global sum,
but does so for each row.
```

```
The second OVER clause partitions rows by country,
producing a sum per partition (per country). The
function produces this sum for each partition row.
```

6.4 Code Example: Advanced SQL Queries

- Combine customer and order information:

```
SELECT Customers.CustomerName, Orders.OrderID, Orders.OrderDate
   FROM Customers
   INNER JOIN Orders ON Customers.CustomerID = Orders.CustomerID;
```

- Find customers with high total purchases:

```
SELECT CustomerName
   FROM Customers
      WHERE CustomerID IN
          (
            SELECT CustomerID
                FROM Orders
                   GROUP BY CustomerID
                     HAVING SUM(TotalAmount) > 10000
          );
```

- Calculate a moving average of sales

```
SELECT OrderDate,
       TotalAmount,
       AVG(TotalAmount) OVER (ORDER BY OrderDate) AS MovingAvg
       FROM Orders;
```

7. Creating Interactive Reports

```
BI reports are interactive, allowing
users to explore data. SQL can be used
to create parameterized queries for
dynamic reports.
```

7.1 Code Example: Creating Parameterized Query

```
-- Create a parameterized query for sales by region
DECLARE @Region VARCHAR(255) = 'West';

SELECT Region, ProductName, SUM(Quantity) AS TotalQuantity
   FROM Sales
      WHERE Region = @Region
      GROUP BY Region, ProductName;
```

8. Data Visualization with SQL

```
BI isn□t just about numbers;
```

```
it’s about SQL and Visuals.
```

```
SQL can be used to retrieve data
for visualization tools like
charts and dashboards.
```

8.1 Code Example: Retrieving Data for Visualization

```
-- Retrieve data for a bar chart of monthly sales
SELECT DATE_FORMAT(OrderDate, '%Y-%m') AS Month,
       SUM(TotalAmount) AS MonthlySales
   FROM Orders
      GROUP BY Month
         ORDER BY Month;
```

9. Security and Access Control

```
In a BI environment, data security is paramount.
SQL can enforce access controls to ensure only
authorized users can access certain data.
```

10. Performance Optimization

```
BI often deals with large datasets.
SQL optimization techniques, like indexing
and query tuning, can significantly boost
performance.
```

11. Real-World BI Examples

```
Let’s look at some real-world scenarios
where SQL is used in BI:
```

1. Sales Analysis: SQL helps analyze sales data to identify trends, topselling products, and regions with high demand.

2. Customer Segmentation: Segmenting customers based on buying behavior using SQL's aggregation functions.
3. Financial Reporting: Generating financial reports like balance sheets and income statements using SQL.
4. Inventory Management: Using SQL to optimize inventory levels and avoid overstocking or understocking.
5. Marketing Campaign Analysis: Measuring the effectiveness of marketing campaigns through SQL queries.

12. Conclusion

1. SQL is the backbone of Business Intelligence.
2. SQL empowers professionals to access, manipulate, and analyze data, turning it into valuable insights.
3. Whether you're a BI analyst, data engineer, or business leader, understanding SQL is essential for making datadriven decisions.
4. So, roll up your sleeves and start querying — your data has stories to tell.
5. This article has provided an overview of how SQL is used in Business Intelligence.
6. To become proficient, practice is key. So, set up your BI environment, load some data, and start exploring. The world of data driven insights awaits!

References

1. SQL for Business Intelligence: Turning Data into Insights
2. SQL Window Functions Cheat Sheet
3. GROUP BY Modifiers
4. SQL Cube
5. Group By in SQL Server with CUBE, ROLLUP and GROUPING SETS Examples