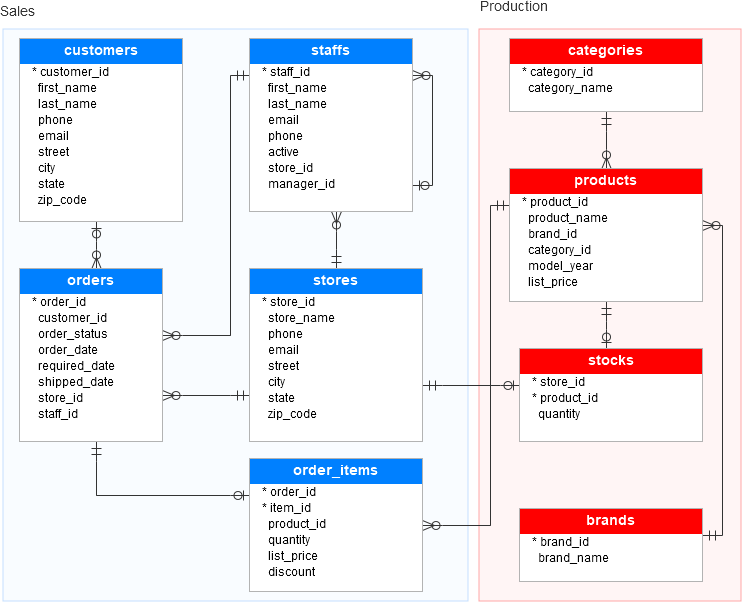
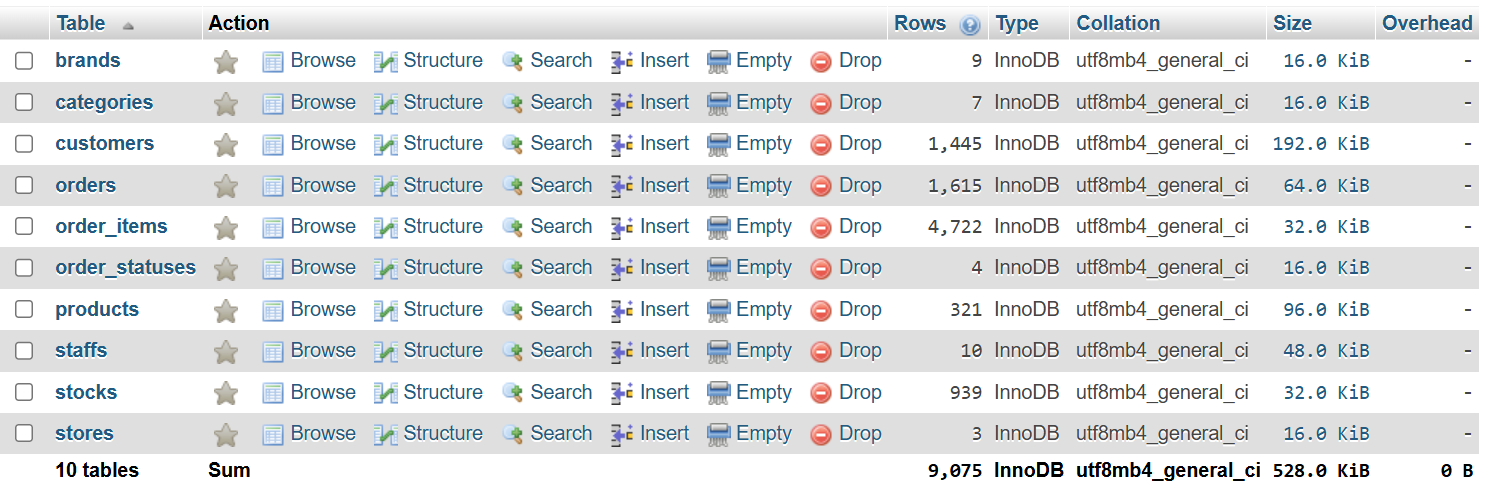
### [**Workshop:** Data analysis using SQL](https://drive.google.com/file/d/12uz6Bcv7P9YRfTRndxH6rMQqRCEpC4bh/view?usp=sharing)

This database serves **as a workshop environment** designed for practicing and experimenting with **data analysis using SQL**. It provides a structured schema that simulates a real-world business scenario, focusing on sales and production data. This setup allows users to write and test SQL queries for tasks such as data retrieval, filtering, aggregations, joins, and advanced analysis.

### [**Kaggle Link: Bike Store Relational Database | SQL**](https://www.kaggle.com/datasets/dillonmyrick/bike-store-sample-database/data)





### **Database Structure**

The database consists of two main areas:

1. Sales: Tracks customer orders, store operations, and order details.
2. Production: Manages product information, brands, categories, and inventory stock.

#### **Tables Overview:**

1. customers  
   Stores information about customers (name, contact details, location).
2. stores  
   Contains details about store locations and contact information.
3. staffs  
   Includes staff details (name, store assignment, and active status).
4. orders  
   Tracks customer orders, including order status, dates, and store/staff responsible.
5. order\_items  
   Details individual items in each order, including quantities, price, and discounts.
6. products  
   Manages product details like names, prices, categories, and brands.
7. categories  
   Defines product categories (e.g., Children Bicycles, Comfort Bicycles).
8. brands  
   Stores brand names associated with products.
9. stocks  
   Tracks product inventory levels at different stores.

### **Step-by-Step Process: Converting CSV Files to a MySQL Database**

### **1. Database Schema Creation**

* We created the database and tables using SQL based on the provided schema diagram.
* We defined Primary Keys and Foreign Keys to establish relationships between the tables.
* The tables were structured logically so that dependencies were clear (e.g., the products table depends on brands and categories).
* **Link for Sql file to tables creation:** [bikestore.sql](https://drive.google.com/file/d/1MXbya5wAH5LMYFgoPO_vrLn6Tpo5Eat0/view?usp=sharing)  to import in your DBMS.

### **2. CSV Files Preparation**[**bikestore .sql**](https://drive.google.com/file/d/1MXbya5wAH5LMYFgoPO_vrLn6Tpo5Eat0/view?usp=sharing)

* We had multiple CSV files (brands.csv, categories.csv, customers.csv, etc.), each corresponding to a table in the database.

### **3. Writing the Python Code for Data Insertion**

* We wrote Python code using the Pandas library to read CSV files and the mysql-connector-python library to connect to the MySQL database and execute SQL commands.
* **Link for Python Code:** [Converting CSV Files to a MySQL Database.ipynb](https://drive.google.com/file/d/1EJzudNC-HwA-XfwlveZYkFtoHNi9NeSL/view?usp=sharing)

### **4. Inserting Data in the Correct Order** To avoid Foreign Key conflicts, we inserted the data in the proper sequence:

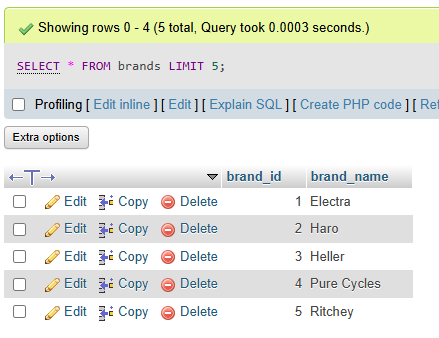
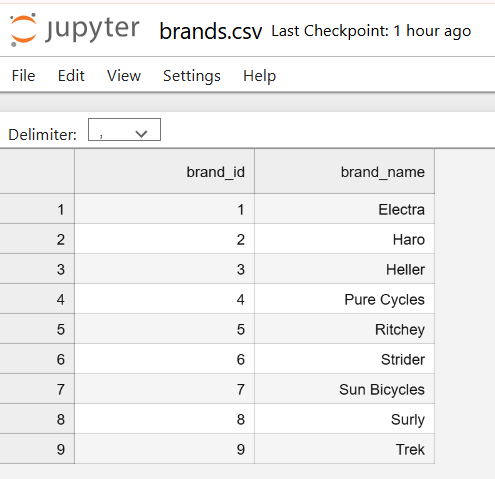
1. brands → No dependencies.
2. categories → No dependencies.
3. products → Depends on brands and categories.
4. stores → Independent.
5. customers → Independent.
6. staffs → Depends on stores.
7. orders → Depends on customers, stores, and staffs.
8. stocks → Depends on stores and products.
9. order\_items → Depends on orders and products.

### **5. Executing Data Insertion**

* We read each CSV file row-by-row and inserted the data into the corresponding table using SQL INSERT commands:

### **7. Verifying the Data**

* After the data insertion, we ran simple SQL queries to ensure the data was successfully inserted:



**SQL Data Analysis Workshop**

[**Link for Jupyter File:** **SQL Data Analysis Workshop**](https://drive.google.com/file/d/1viS8Uq0Va_uEKJv9jn6bVEEG_Db2ufDV/view?usp=sharing)

This workshop is designed to help you build practical SQL skills by analyzing real-world business data. The tasks are divided into 4 levels to gradually progress from basic data retrieval to advanced analysis. Each level focuses on specific SQL concepts and challenges, allowing you to practice query writing, data filtering, aggregations, joins, and subqueries.

* **Level 1:** Basic SQL queries (SELECT, WHERE, ORDER BY, COUNT).
* **Level 2:** Aggregations (SUM, AVG, GROUP BY) and basic joins.
* **Level 3:** Complex joins, subqueries, and multi-table analysis.
* **Level 4:** Advanced analytical tasks focused on complex queries, subqueries, and aggregations..

This structured approach ensures you build confidence and expertise in SQL, preparing you for real-world data analysis challenges. 🚀

**Level 1: Beginner – Basic Queries (using SQL Magic)**

These tasks focus on learning how to retrieve data, apply simple filters, and perform sorting.

* **Task 1: Retrieve All Data**
  + Retrieve all rows and columns from the customers table.
* **Task 2: Filter Customers by City**
  + Retrieve customers who live in the city "New York".
* **Task 3: Sort Products by Price**
  + Retrieve all products sorted by their price in descending order.
* **Task 4: Find Orders by Status**
  + Retrieve all orders with status "Completed".
  + Retrieve count of orders with status "Rejected".
* **Task 5: Count the Number of Customers**
  + Count how many customers exist in the **customers table.**

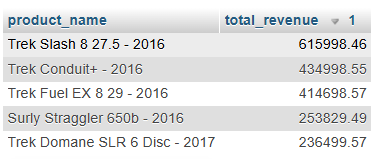
## **Level 2: Intermediate – Aggregations and Basic Joins (Task 1,2,3 using Pandas read\_sql and Task 4,5 using execute and fetchall() in mysql.connector)**

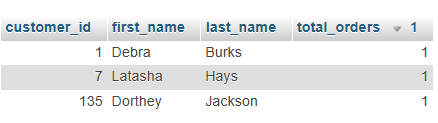
This level introduces aggregations like SUM, AVG, GROUP BY, and joins to combine data across tables.

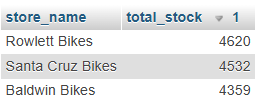
* **Task 1: Total Revenue**
  + Calculate the total revenue (sum of **list\_price \* quantity) from the order\_items table.**
* **Task 2: Average Product Price**
  + Calculate the average price of products in the **products table.**
* **Task 3: Count Orders Per Customer**
  + Retrieve the number of orders placed by each customer.
* **Task 4: Join Orders with Customers**
  + Retrieve customer names along with their order IDs.
  + Retrieve Customer Names with Order IDs and Total Orders
* **Task 5: Products with Low Stock**
  + Retrieve products with stock quantities less than 50.

## **Level 3: Advanced – Complex Joins and Subqueries (using Pandas read\_sql)**

**Tasks at this level involve multiple joins, subqueries, and advanced filtering.**

* **Task 1:** Retrieve the top 5 products with the highest total sales revenue.  
  
* **Task 2:** Find the customers who placed the most orders in June, November 2018



* **Task 3:** List all stores with their total stock quantities for all products.
* **Task 4:** Retrieve staff members who work at stores located in a specific state.
  + California (CA) , New York (NY) , Texas (TX)

For example CA: 

* **Task 5:** Identify the categories of products with the highest total revenue.  
  

### 

### 

### 

### 

### 

### 

### 

### 

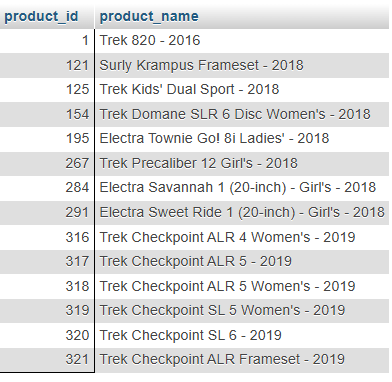
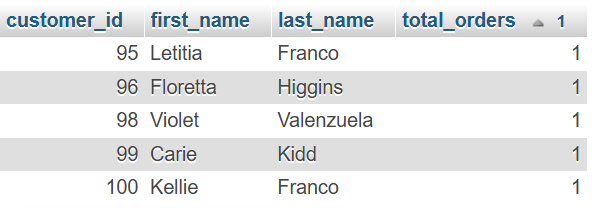
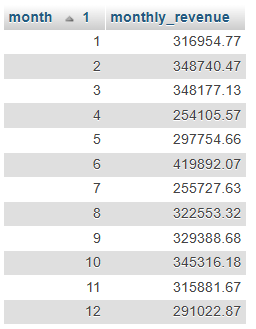
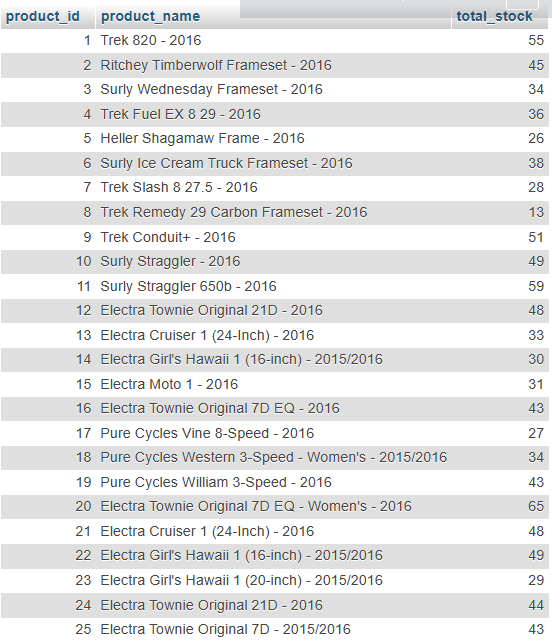
### 

### 

### 

### 

### **Level 4: Expert – Advanced Analytical Queries (using Pandas read\_sql)**

* **Task 1: Find the Store with the Highest Total Revenue** 
  + Retrieve the store name and total revenue (sum of list\_price \* quantity) across all orders.  
    
* **Task 2: Retrieve Products with No Orders**
  + Find all products that have never been ordered.  
    
* **Task 3: Identify the Customers Who Placed the Fewest Orders**
  + Retrieve customer names and their total order counts, ordered in ascending order of order count.  
    The output below: ORDER BY total\_orders,c.customer\_id ASC LIMIT 5;  
    
* **Task 4: Analyze Monthly Revenue for the Last Year (Specific to 2017)  
  **
* **Task 5: Find Products That Are Low in Stock Across All Stores**
  + Identify products where the total stock quantity (across all stores) is less than 100.  
      
    sample:  
    

### **Level 5: Additional Advanced SQL (using Pandas read\_sql)**

#### **Task 1: Aggregate Sales by City and State**

### Write a query to calculate the total sales, average order value, and maximum order value for customers in each city and state.

The output below: ORDER BY total\_sales DESC;



#### **Task 2: Find Top-Selling Products in 2018**

* + Write a query to identify the products with the highest sales volume in the year 2018.

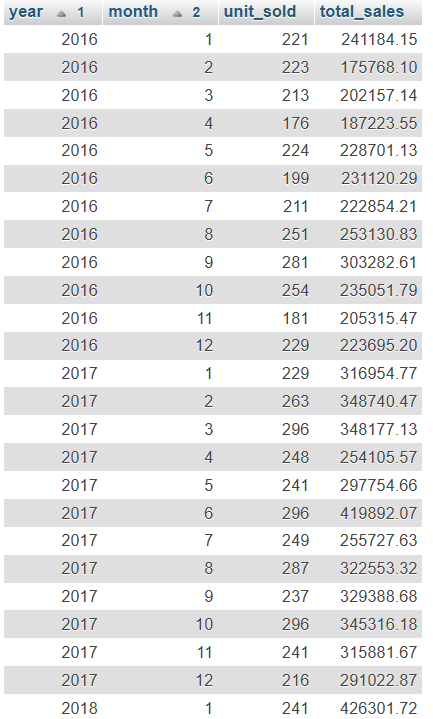
The output below: ORDER BY total\_unit\_sold DESC;



#### **Task 3: Calculate Month-over-Month Sales Growth**

* + Write a query to calculate the month-over-month sales growth for the past years.

The output below: ORDER BY year, month;

****