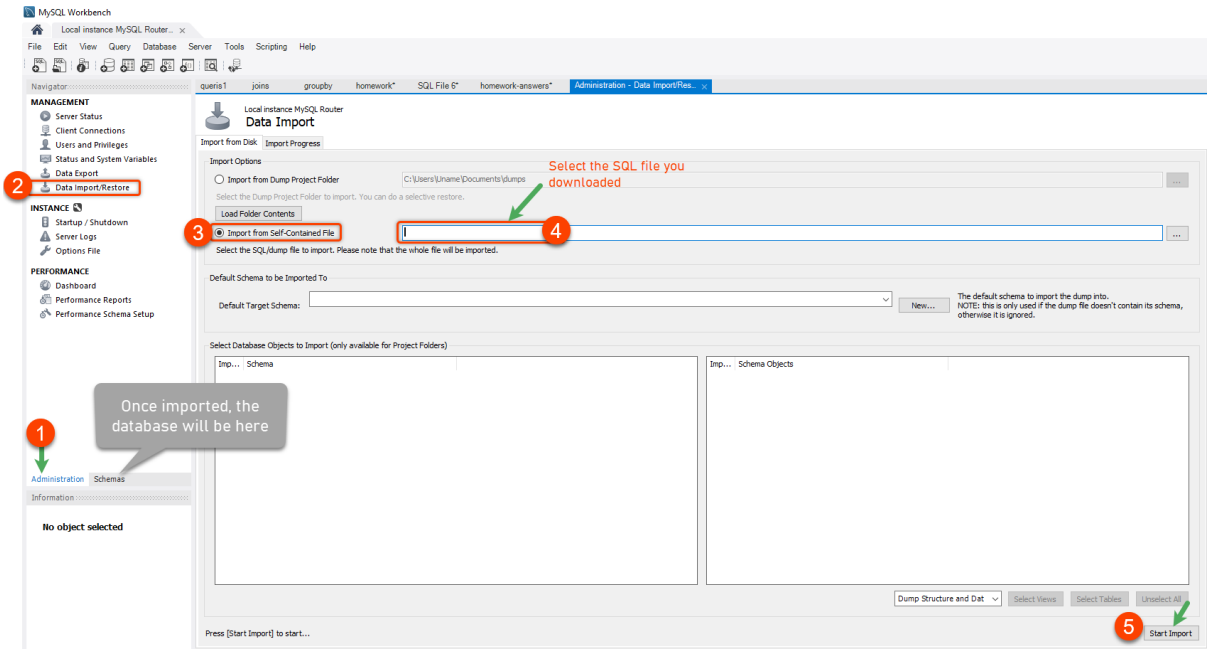
1. Open MySQL Workbench, login if necessary
2. Click on the “server administration” tab (see illustration, click to expand)
3. Click on “Data Import/Restore”
4. Select the option “Import from self-contained file”
5. Specify the path of the downloaded awesome-chocolates-data.sql file
6. Start import



-- To check what is in the database

1.

```
SHOW tables;
```

-- To check structure of the sales table

```
desc sales;
```

-- To check what is in the tables

```
SELECT * FROM sales;
SELECT * FROM people;
SELECT * FROM geo;
SELECT * FROM product;
```

-- To see some of the attributes

2.

```
SELECT
   SaleDate, Amount, Customers
FROM
   sales;
```

-- Selecting in our own order

3.

```
SELECT
  Amount, Customers, GeoID
FROM
  sales;
```

-- Calculation in SQL queries

4.

```
SELECT
  SaleDate, Amount, Boxes, Amount / Boxes 'Amount per Box'
FROM
  sales;
```

-- Impose conditions using WHERE Clause it works like filter

5.

```
SELECT
  *
FROM
  sales
WHERE
  Amount > 10000;
```

-- ORDER Clause works as sorting

6.

```
SELECT
  *
FROM
  sales
WHERE
  Amount > 10000
ORDER BY Amount;
```

-- DESC keyword to arrange the result in descending order

7.

```
SELECT
  *
FROM
  sales
WHERE
  Amount > 10000
ORDER BY Amount DESC;
```

-- Get all the records from sales table where GeoID is g1 and display the result by sorting with respect to PID and Amount (amount in descending order).

8.

```
SELECT
   *
FROM
   sales
WHERE
   GeoID = 'g1'
ORDER BY PID, Amount DESC;
```

-- Get all the records on or after 1st Jan 2022 from sales table where amount is greater than 10000.

9.

```
SELECT
   *
FROM
   sales
WHERE
   Amount > 10000
      AND SaleDate >= '2022-01-01';
```

-- Get all the records of 2022 from sales table where amount is greater than 10000, sort the result in descending order of amount.

10.

```
SELECT
   SaleDate, Amount
FROM
   sales
WHERE
   Amount > 10000 AND YEAR(SaleDate) = 2022
ORDER BY Amount DESC;
```

-- Get all the records form sales table where number of boxes are between 0 and 50.

11.

```
SELECT
   *
FROM
   sales
WHERE
   boxes >= 0 AND Boxes <= 50;
```

12.

```
SELECT
```

```
    *
FROM
    sales
WHERE
    Boxes BETWEEN 0 AND 50;
```

-- Some of Date functions
-- Get SaleDate, Amount, Boxes from sales table where day of sale is 'Friday'.

13.

```
SELECT
    SaleDate, Amount, Boxes, DAYNAME(saledate) AS 'Day of Week'
FROM
    sales
WHERE
    DAYNAME(saledate) = 'Friday';
```

-- Get all the records of Delish and Jucies teams from people table.

14.

```
SELECT
    *
FROM
    people
WHERE
    team IN ('Delish' , 'Jucies');
```

-- Get all the records of Salespersons whose names start with 'B' from people table.

15.

```
SELECT
    *
FROM
    people
WHERE
    Salesperson LIKE 'B%';
```

16.

```
SELECT
    *
FROM
    people
WHERE
    Salesperson LIKE '%B%';
```

-- Get SaleDate and Amount column from sales table and a new column to mention the amount when amount is < 1000 as 'Under 1k', when < 5000 as 'Under 5k', < 10000 as 'Under 10k' else '10k or more'.

17.

```
SELECT
  SaleDate,
  Amount,
  CASE
    WHEN amount < 1000 THEN 'Under 1k'
    WHEN Amount < 5000 THEN 'Under 5k'
    WHEN Amount < 10000 THEN 'Under 10k'
    ELSE '10k or more'
  END AS 'Amount category'
FROM
  sales;
```

-- Joins

-- Before using Joins understand your data well.

```
SELECT * FROM sales;
SELECT * FROM people;
SELECT * FROM geo;
SELECT * FROM product;
```

-- Get date of sale, amount and the name of salespersons by using Joins.

18.

```
SELECT
  s.SaleDate, s.Amount, p.Salesperson
FROM
  sales s
    JOIN
  people p on p.SPID = s.SPID;
```

-- Get date of sale, amount and the name of products by using Joins.

19.

```
SELECT
  s.SaleDate, s.Amount, pr.Product
FROM
  sales s
    LEFT JOIN
  products pr ON pr.PID = s.PID;
```

-- Get date of sale, amount, name of salesperson, name of product and team by using Joins.

20.

```
SELECT
  s.SaleDate, s.Amount, p.Salesperson, pr.Product, p.Team
FROM
   sales s
      JOIN
   people p ON p.SPID = s.SPID
      JOIN
   products pr on pr.PID = s.PID;
```

-- Get date of sale, amount, name of salesperson, name of product and team who have sales less than $500 by using Joins.

21.

```
SELECT
   s.SaleDate, s.Amount, p.Salesperson, pr.Product, p.Team
FROM
   sales s
      JOIN
   people p ON p.SPID = s.SPID
      JOIN
   products pr ON pr.PID = s.PID
WHERE
   s.Amount < 500;
```

-- Get date of sale, amount, name of salesperson, name of product, of 'Delish' team who have sales less than $500 by using Joins.

22.

```
SELECT
   s.SaleDate, s.Amount, p.Salesperson, pr.Product, p.Team
FROM
   sales s
      JOIN
   people p ON p.SPID = s.SPID
      JOIN
   products pr ON pr.PID = s.PID
WHERE
   s.Amount < 500 AND p.Team = 'Delish';
```

-- Get date of sale, amount, name of salesperson, name of product and team who have sales less than $500 and team not mentioned by using Joins.

23.

```
SELECT
   s.SaleDate, s.Amount, p.Salesperson, pr.Product, p.Team
FROM
   sales s
```

```
    JOIN
  people p ON p.SPID = s.SPID
    JOIN
  products pr ON pr.PID = s.PID
WHERE
  s.Amount < 500 AND p.Team = '';
```

-- Get date of sale, amount, name of salesperson, name of product, 'Delish' team and Geography where sales is less than $500 and country is New Zealand or India by using Joins.

24.

```
SELECT
  s.SaleDate,
  s.Amount,
  p.Salesperson,
  pr.Product,
  p.Team,
  g.Geo
FROM
  sales s
    JOIN
  people p ON p.SPID = s.SPID
    JOIN
  products pr ON pr.PID = s.PID
    JOIN
  geo g ON g.GeoID = s.GeoID
WHERE
  s.Amount < 500 AND p.Team = 'Delish'
    AND g.Geo IN ('New Zealand', 'India');
```

-- Get date of sale, amount, name of salesperson, name of product, 'Delish' team and Geography where sales is less than $500 and country is New Zealand or India by using Joins and sort the result by sale date.

25.

```
SELECT
  s.SaleDate,
  s.Amount,
  p.Salesperson,
  pr.Product,
  p.Team,
  g.Geo
FROM
  sales s
    JOIN
  people p ON p.SPID = s.SPID
```

```
        JOIN
    products pr ON pr.PID = s.PID
        JOIN
    geo g ON g.GeoID = s.GeoID
WHERE
    s.Amount < 500 AND p.Team = 'Delish'
        AND g.Geo IN ('New Zealand' , 'India')
ORDER BY s.SaleDate;
```

-- Group By is used as Pivot table in excel

-- Get total sale, average sale, total boxes and average boxes sold in each GeoID from sales table.

26.

```
SELECT
    GeoID,
    SUM(Amount),
    AVG(Amount),
    SUM(Boxes),
    AVG(Boxes)
FROM
    sales
GROUP BY GeoID;
```

-- Get total sale, average sale, total boxes and average boxes sold in each Geography.

27.

```
SELECT
    g.Geo, SUM(Amount), AVG(Amount), SUM(Boxes), AVG(Boxes)
FROM
    sales s
        JOIN
    geo g ON g.GeoID = s.GeoID
GROUP BY g.Geo;
```

-- Get total boxes sold and total sale according to each product category and team.

28.

```
SELECT
    pr.Category, p.Team, SUM(Boxes), SUM(Amount)
FROM
    sales s
        JOIN
    people p ON p.spid = s.spid
        JOIN
    products pr ON pr.pid = s.pid
group by pr.Category, p.Team;
```

-- Get total boxes sold and total sale according to each product category and team and sort the data with respect to Category and team.

29.

```
SELECT
   pr.Category, p.Team, SUM(Boxes), SUM(Amount)
FROM
   sales s
      JOIN
   people p ON p.spid = s.spid
      JOIN
   products pr ON pr.pid = s.pid
GROUP BY pr.Category , p.Team
ORDER BY pr.Category , p.Team;
```

-- Get total boxes sold and total sale according to each product category and team where team is not blank and sort the data with respect to Category and team.

30.

```
SELECT
   pr.Category, p.Team, SUM(Boxes), SUM(Amount)
FROM
   sales s
      JOIN
   people p ON p.spid = s.spid
      JOIN
   products pr ON pr.pid = s.pid
WHERE
   p.Team <> ''
GROUP BY pr.Category , p.Team
ORDER BY pr.Category , p.Team;
```

-- Get total sale as ‘Total Amount’ according to each product and sort the data with respect to ‘Total Amount’ in descending order.

31.

```
SELECT
   pr.Product, SUM(s.Amount) AS 'Total Amount'
FROM
   sales s
      JOIN
   products pr ON pr.pid = s.pid
GROUP BY pr.Product
ORDER BY 'Total Amount' DESC;
```

-- Get total boxes sold and total sale of top five products.

32.

```
SELECT
   pr.Product, SUM(s.Amount) AS 'Total Amount'
FROM
   sales s
      JOIN
   products pr ON pr.pid = s.pid
GROUP BY pr.Product
ORDER BY 'Total Amount' DESC
LIMIT 5;
```