BID3000

Group exam

fall 2023

Candidate numbers
7004
7006
7007
7027
7044

Part 1: Building the data warehouse using SQL

The aim of this part is to create the dimension and the fact tables and fill them with data from the source database. These aims will be achieved using SQL and by carrying out the following tasks:

1.1. Create a data warehouse calssicmodels_dw**.** Provide the SQL code for this task.

```
CREATE DATABASE classicmodels_dw;

USE classicmodels_dw;
```

1.2. Provide the SQL code that create the Dim_Customer table that belongs to the data warehouse calssicmodels_dw. The fields of this table are shown in Figure 2.

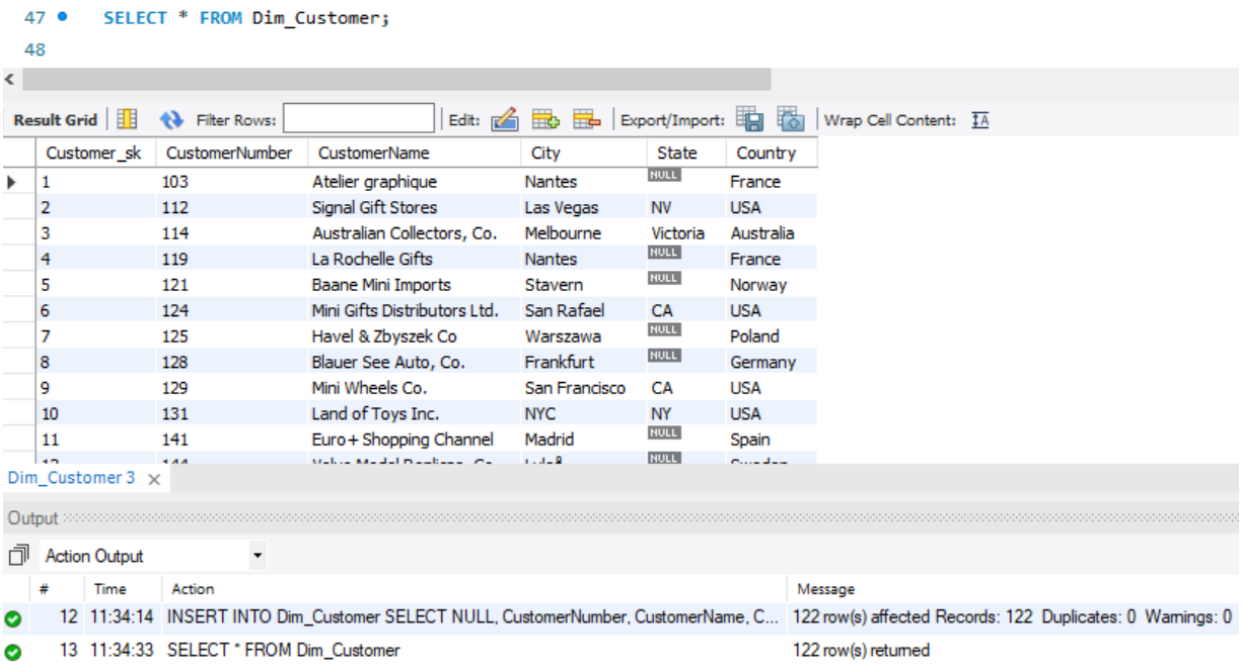
```
CREATE TABLE Dim_Customer(
   Customer_sk INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
   CustomerNumber INT,
   CustomerName VARCHAR(50),
   City VARCHAR(50),
   State VARCHAR(50),
   Country VARCHAR(50));
```

1.3. Fill the **Dim_Customer** table from the source database. Provide the SQL code for this task.

```
INSERT INTO Dim_Customer
SELECT
NULL,
CustomerNumber,
CustomerName,
City,
State,
Country
FROM classicmodels.customers;
```

1.4. Provide a screenshot of: (1) the **Dim_Customer** table filled with data, and (2) the SQL code that display the content of the table **Dim_Customer**. If the result is too long and does not hold in a single screenshot, it is OK to present part of the result (This remark is valid for all the required screenshots in this exam).

```
SELECT * FROM Dim_Customer;
```



1.5. Provide the SQL code that create the **Dim_Employee** table that belongs to the data warehouse calssicmodels_dw. The fields of this table are shown in Figure 2.

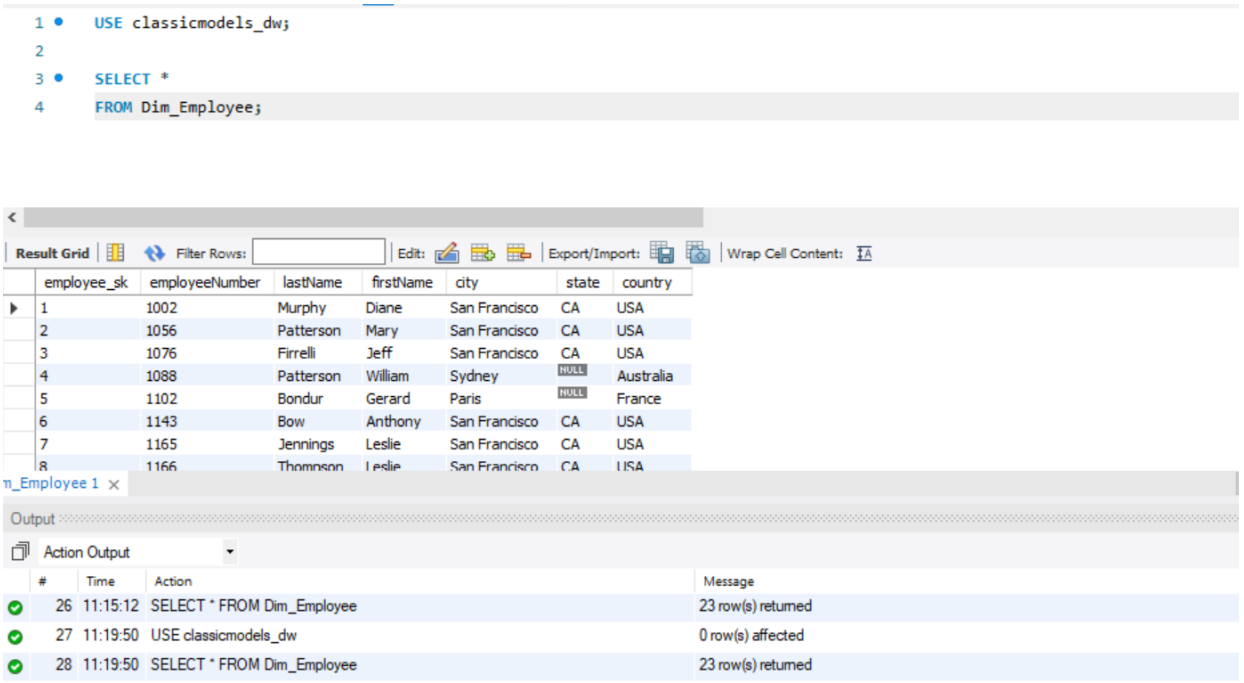
```
DROP TABLE IF EXISTS Dim_Employee;

CREATE TABLE Dim_Employee (

    employee_sk INT NOT NULL AUTO_INCREMENT PRIMARY KEY,

  employeeNumber INT,

  lastName VARCHAR(50),

  firstName VARCHAR(50),

  city VARCHAR(50),

  state VARCHAR(50),

  country VARCHAR(50)

);
```

1.6. Fill the **Dim_Employee** table from the source database. Provide the SQL code for this task.

```
INSERT INTO Dim_Employee
SELECT
   NULL,
   employeeNumber,
   lastName,
   firstName,
   city,
   state,
   country
FROM classicmodels.employees LEFT JOIN classicmodels.offices
ON classicmodels.employees.officeCode = classicmodels.offices.officeCode;
```

1.7. Provide a screenshot of: (1) the **Dim_Employee** table filled with data, and (2) the SQL code that display the content of the table **Dim_Employee**.

```
SELECT *
FROM Dim_Employee;
```



```
1  USE classicmodels_dw;
2
3  SELECT *
4  FROM Dim_Employee;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

employee_sk	employeeNumber	lastName	firstName	city	state	country
1	1002	Murphy	Diane	San Francisco	CA	USA
2	1056	Patterson	Mary	San Francisco	CA	USA
3	1076	Firrelli	Jeff	San Francisco	CA	USA
4	1088	Patterson	William	Sydney	NULL	Australia
5	1102	Bondur	Gerard	Paris	NULL	France
6	1143	Bow	Anthony	San Francisco	CA	USA
7	1165	Jennings	Leslie	San Francisco	CA	USA
8	1166	Thompson	Leslie	San Francisco	CA	USA

n_Employee 1

Output

Action Output

#	Time	Action	Message
26	11:15:12	SELECT * FROM Dim_Employee	23 row(s) returned
27	11:19:50	USE classicmodels_dw	0 row(s) affected
28	11:19:50	SELECT * FROM Dim_Employee	23 row(s) returned

1.8. Provide the SQL code that create the **Dim_Product** table that belongs to the data warehouse calssicmodels_dw. The fields of this table are shown in Figure 2.

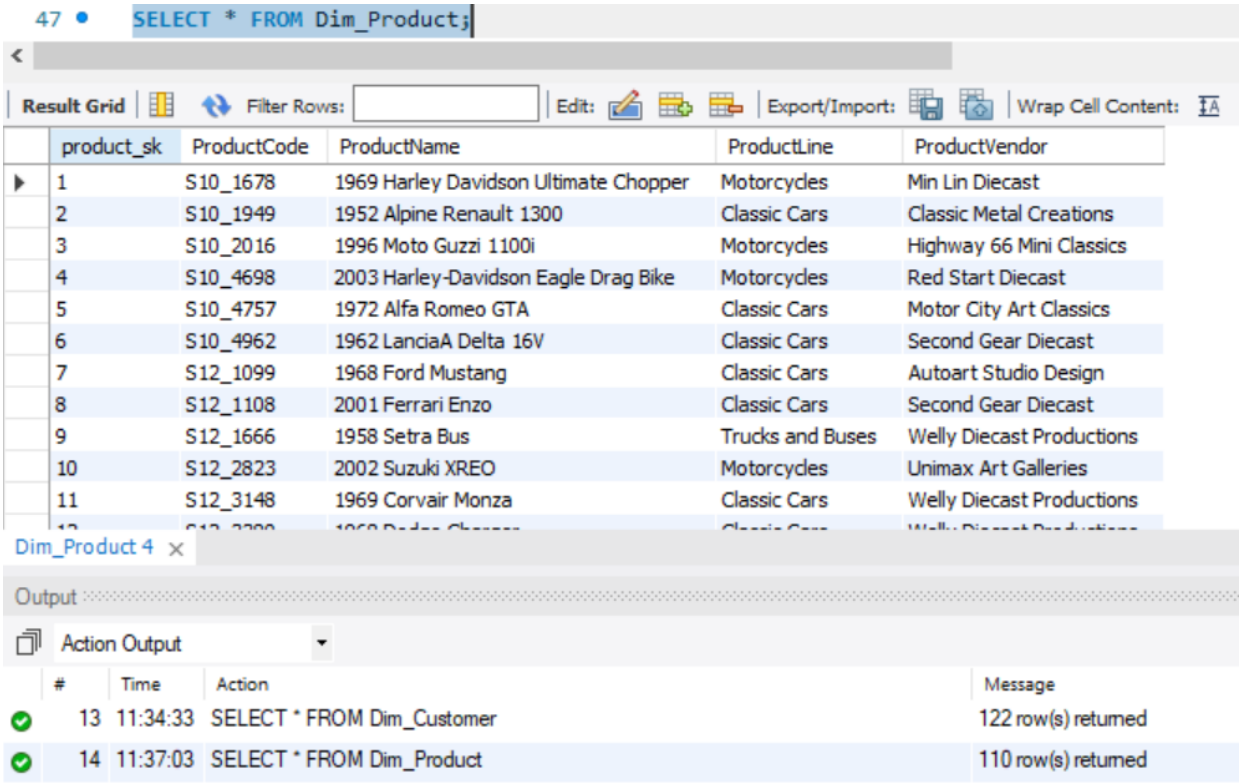
```
CREATE TABLE Dim_Product (
product_sk INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
ProductCode VARCHAR(15),
ProductName VARCHAR(70),
ProductLine VARCHAR(50),
ProductVendor VARCHAR(50));
```

1.9. Fill the **Dim_Product** table from the source database. Provide the SQL code for this task.

```
INSERT INTO Dim_Product
SELECT
NULL,
productCode ,
productName,
productLine ,
productVendor
FROM classicmodels.products;
```

1.10. Provide a screenshot of: (1) the **Dim_Product** table filled with data, and (2) the SQL code that display the content of the table **Dim_Product**.

```
SELECT * FROM Dim_Product;
```



1.11. Provide the SQL code that create the **Dim_Time** table that belongs to the data warehouse calssicmodels_dw. The fields of this table are shown in Figure 2.

```
USE classicmodels_dw;

DROP TABLE IF EXISTS Dim_Time;
CREATE TABLE Dim_Time (
      time_sk INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
   date DATE,
   month VARCHAR(9),
   quarter INT(1),
   year INT(4)
   );
```

1.12. Create a procedure named "**Fill_timedimension**" that fills the **Dim_Time** table. This procedure takes the start and end date as input parameters. Provide the SQL code for this procedure.

```
DROP PROCEDURE IF EXISTS Fill_timedimention;

DELIMITER //
CREATE PROCEDURE Fill_timedimention(IN start_date DATE, in end_date DATE)
BEGIN
   WHILE start_date < end_date do
            INSERT INTO dim_time VALUES (
            NULL,
        start_date,
        month(start_date),
        quarter(start_date),
        year(start_date));
        SET start_date = ADDDATE(start_date,1);
   END WHILE;
END; //
DELIMITER ;
```

1.13. Provide the value of the start date and the value of the end date by examining the source database.

```
34 •  USE classicmodels;
35
36 •  SELECT MIN(orderdate) as StartDate, MAX(orderdate) as EndDate
37    FROM orders;
```

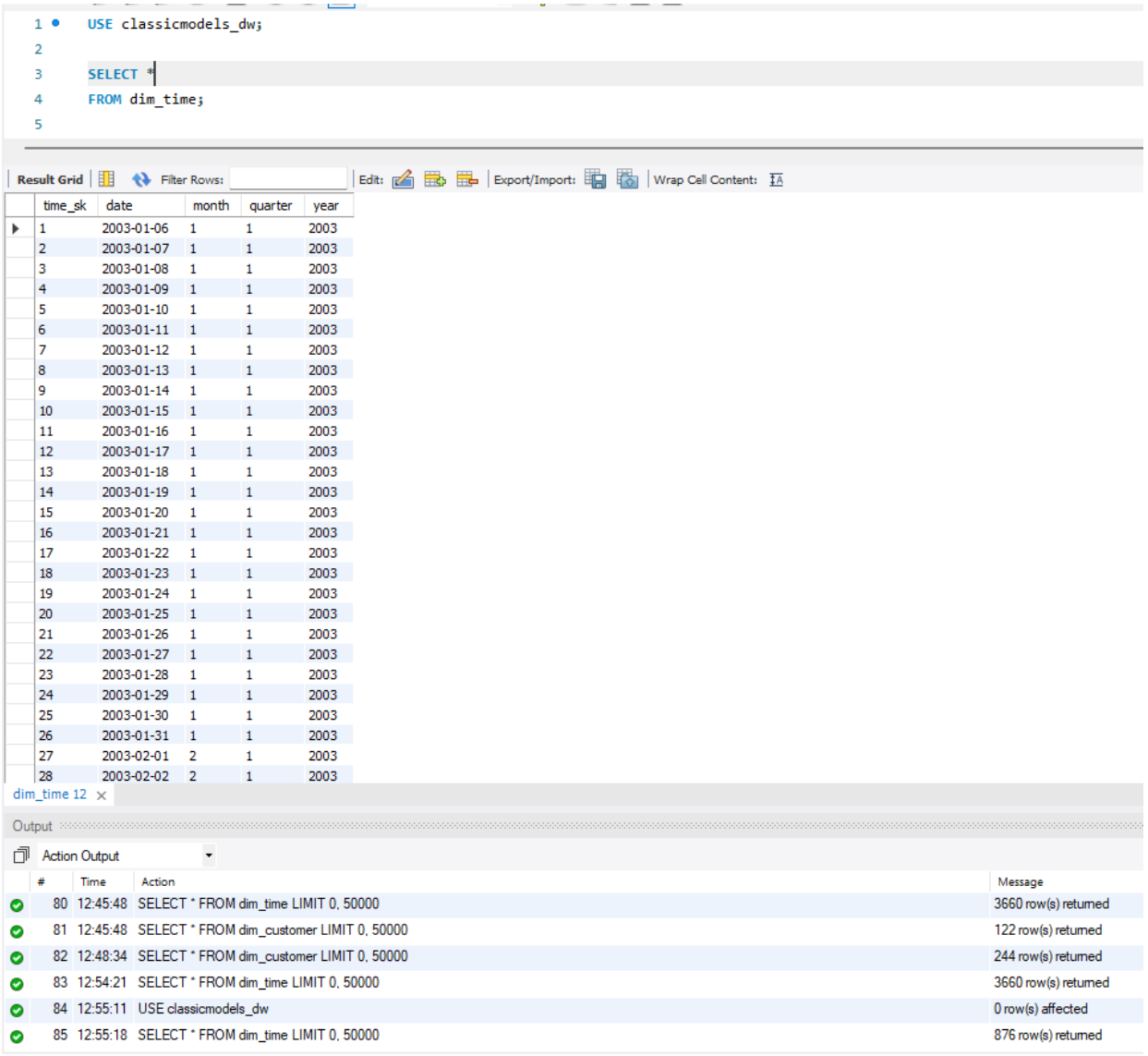
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

StartDate	EndDate
2003-01-06 00:00:00	2005-05-31 00:00:00

1.14. Call the procedure **Fill_timedimension** with the right start and end date. Provide the SQL code for this task.

```
CALL Fill_timedimention("2003-01-06","2005-05-31");
```

1.15. Provide a screenshot of: (1) the **Dim_Time** table filled with data, and (2) the SQL code that display the content of the table **Dim_Time**.



```
USE classicmodels_dw;

SELECT *
FROM dim_time;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

time_sk	date	month	quarter	year
1	2003-01-06	1	1	2003
2	2003-01-07	1	1	2003
3	2003-01-08	1	1	2003
4	2003-01-09	1	1	2003
5	2003-01-10	1	1	2003
6	2003-01-11	1	1	2003
7	2003-01-12	1	1	2003
8	2003-01-13	1	1	2003
9	2003-01-14	1	1	2003
10	2003-01-15	1	1	2003
11	2003-01-16	1	1	2003
12	2003-01-17	1	1	2003
13	2003-01-18	1	1	2003
14	2003-01-19	1	1	2003
15	2003-01-20	1	1	2003
16	2003-01-21	1	1	2003
17	2003-01-22	1	1	2003
18	2003-01-23	1	1	2003
19	2003-01-24	1	1	2003
20	2003-01-25	1	1	2003
21	2003-01-26	1	1	2003
22	2003-01-27	1	1	2003
23	2003-01-28	1	1	2003
24	2003-01-29	1	1	2003
25	2003-01-30	1	1	2003
26	2003-01-31	1	1	2003
27	2003-02-01	2	1	2003
28	2003-02-02	2	1	2003

dim_time 12

Output

Action Output

#	Time	Action	Message
80	12:45:48	SELECT * FROM dim_time LIMIT 0, 50000	3660 row(s) returned
81	12:45:48	SELECT * FROM dim_customer LIMIT 0, 50000	122 row(s) returned
82	12:48:34	SELECT * FROM dim_customer LIMIT 0, 50000	244 row(s) returned
83	12:54:21	SELECT * FROM dim_time LIMIT 0, 50000	3660 row(s) returned
84	12:55:11	USE classicmodels_dw	0 row(s) affected
85	12:55:18	SELECT * FROM dim_time LIMIT 0, 50000	876 row(s) returned

1.16. Create the **Fact_Stage_Order** table. Provide the SQL code for this task.

```
CREATE TABLE Fact_Stage_Order(
fact_pk INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
customer_sk INT,
product_sk INT,
employee_sk INT,
time_sk  INT,
orderNumber INT,
orderDate DATE,
customerNumber INT,
productCode VARCHAR(15),
salesRepEmployeeNumber INT,
quantityOrdered INT,
priceEach FLOAT(10,2),
total_sale FLOAT(10,2));
```

1.17. Provide the SQL code used to fill the **Fact_Stage_Order** table from the source database.

```
TRUNCATE Fact_Stage_Order;
INSERT INTO Fact_Stage_Order
SELECT
NULL,
NULL,
NULL,
NULL,
NULL,
orders.orderNumber,
orders.orderDate,
orders.customerNumber,
orderDetails.productCode,
customers.salesRepEmployeeNumber,
orderDetails.quantityOrdered,
orderDetails.priceEach,
(orderDetails.quantityOrdered*orderDetails.priceEach)
FROM classicmodels.orders,classicmodels.orderdetails,classicmodels.customers
WHERE orders.orderNumber = orderDetails.orderNumber AND customers.customerNumber =
orders.customerNumber;
```

1.18. Add the surrogate keys to the staging table **Fact_Stage_Order**. Provide the SQL code for this task.

```
UPDATE fact_stage_order,dim_customer
SET fact_stage_order.customer_sk = dim_customer.customer_sk
WHERE fact_stage_order.customerNumber = dim_customer.customerNumber;

UPDATE fact_stage_order,dim_product
SET fact_stage_order.product_sk = dim_product.product_sk
WHERE fact_stage_order.productCode = dim_product.ProductCode;

UPDATE fact_stage_order,dim_employee
SET fact_stage_order.employee_sk = dim_employee.employee_sk
```

```
WHERE fact_stage_order.salesRepEmployeeNumber = dim_employee.employeeNumber;

UPDATE fact_stage_order,dim_time
SET fact_stage_order.time_sk = dim_time.time_sk
WHERE fact_stage_order.orderDate = dim_time.date;
```

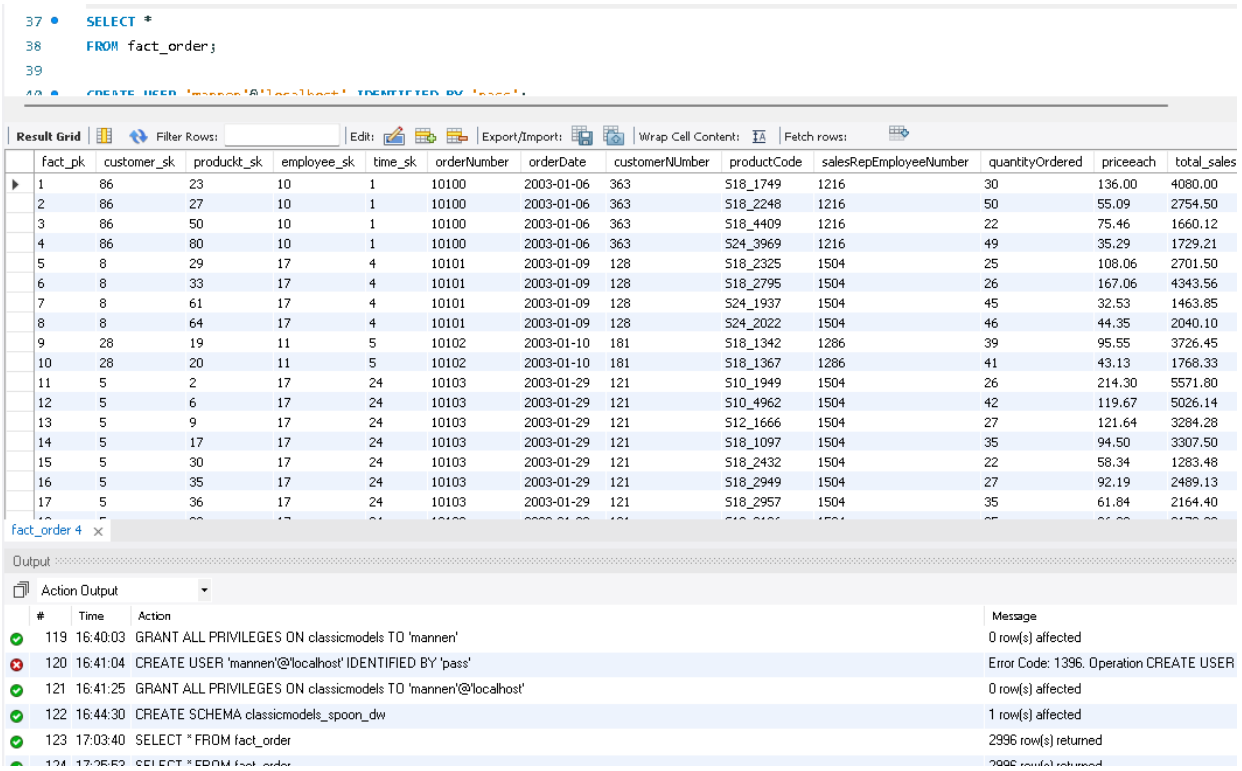
1.19. Create the **Fact_Order** table that belongs to the data warehouse calssicmodels_dw. Provide the SQL code for this task. The fields of this table are shown in Figure 2.

```
CREATE TABLE fact_order(
fact_pk INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
customer_sk INT,
produckt_sk INT,
employee_sk INT,
time_sk INT,
orderNumber INT,
orderDate DATE,
customerNUmber INT,
productCode VARCHAR(15),
salesRepEmployeeNumber INT,
quantityOrdered INT,
priceeach FLOAT(10,2),
total_sales FLOAT(10,2));
```

1.20. Provide the SQL code used to fill the **Fact_Order** table from the **Fact_Stage_Order** table.

```
INSERT INTO fact_order
SELECT
NULL,
customer_sk,
product_sk,
employee_sk,
time_sk,
orderNumber,
orderDate,
customerNumber,
productCode,
salesRepEmployeeNumber,
quantityOrdered,
priceEach,
total_sale
FROM fact_stage_order;
```

1.21. Provide a screenshot of: (1) the **Fact_Order** table filled with data, and (2) the SQL code that display the content of the table **Fact_Order**.



```
37 •  SELECT *
38    FROM fact_order;
39
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch rows:

fact_pk	customer_sk	produckt_sk	employee_sk	time_sk	orderNumber	orderDate	customerNUmber	productCode	salesRepEmployeeNumber	quantityOrdered	priceeach	total_sales
1	86	23	10	1	10100	2003-01-06	363	S18_1749	1216	30	136.00	4080.00
2	86	27	10	1	10100	2003-01-06	363	S18_2248	1216	50	55.09	2754.50
3	86	50	10	1	10100	2003-01-06	363	S18_4409	1216	22	75.46	1660.12
4	86	80	10	1	10100	2003-01-06	363	S24_3969	1216	49	35.29	1729.21
5	8	29	17	4	10101	2003-01-09	128	S18_2325	1504	25	108.06	2701.50
6	8	33	17	4	10101	2003-01-09	128	S18_2795	1504	26	167.06	4343.56
7	8	61	17	4	10101	2003-01-09	128	S24_1937	1504	45	32.53	1463.85
8	8	64	17	4	10101	2003-01-09	128	S24_2022	1504	46	44.35	2040.10
9	28	19	11	5	10102	2003-01-10	181	S18_1342	1286	39	95.55	3726.45
10	28	20	11	5	10102	2003-01-10	181	S18_1367	1286	41	43.13	1768.33
11	5	2	17	24	10103	2003-01-29	121	S10_1949	1504	26	214.30	5571.80
12	5	6	17	24	10103	2003-01-29	121	S10_4962	1504	42	119.67	5026.14
13	5	9	17	24	10103	2003-01-29	121	S12_1666	1504	27	121.64	3284.28
14	5	17	17	24	10103	2003-01-29	121	S18_1097	1504	35	94.50	3307.50
15	5	30	17	24	10103	2003-01-29	121	S18_2432	1504	22	58.34	1283.48
16	5	35	17	24	10103	2003-01-29	121	S18_2949	1504	27	92.19	2489.13
17	5	36	17	24	10103	2003-01-29	121	S18_2957	1504	35	61.84	2164.40

fact_order 4

Output

Action Output

#	Time	Action	Message
119	16:40:03	GRANT ALL PRIVILEGES ON classicmodels TO 'mannen'	0 row(s) affected
120	16:41:04	CREATE USER 'mannen'@'localhost' IDENTIFIED BY 'pass'	Error Code: 1396. Operation CREATE USER
121	16:41:25	GRANT ALL PRIVILEGES ON classicmodels TO 'mannen'@'localhost'	0 row(s) affected
122	16:44:30	CREATE SCHEMA classicmodels_spoon_dw	1 row(s) affected
123	17:03:40	SELECT * FROM fact_order	2996 row(s) returned

Part 2: Building the data warehouse using Spoon.

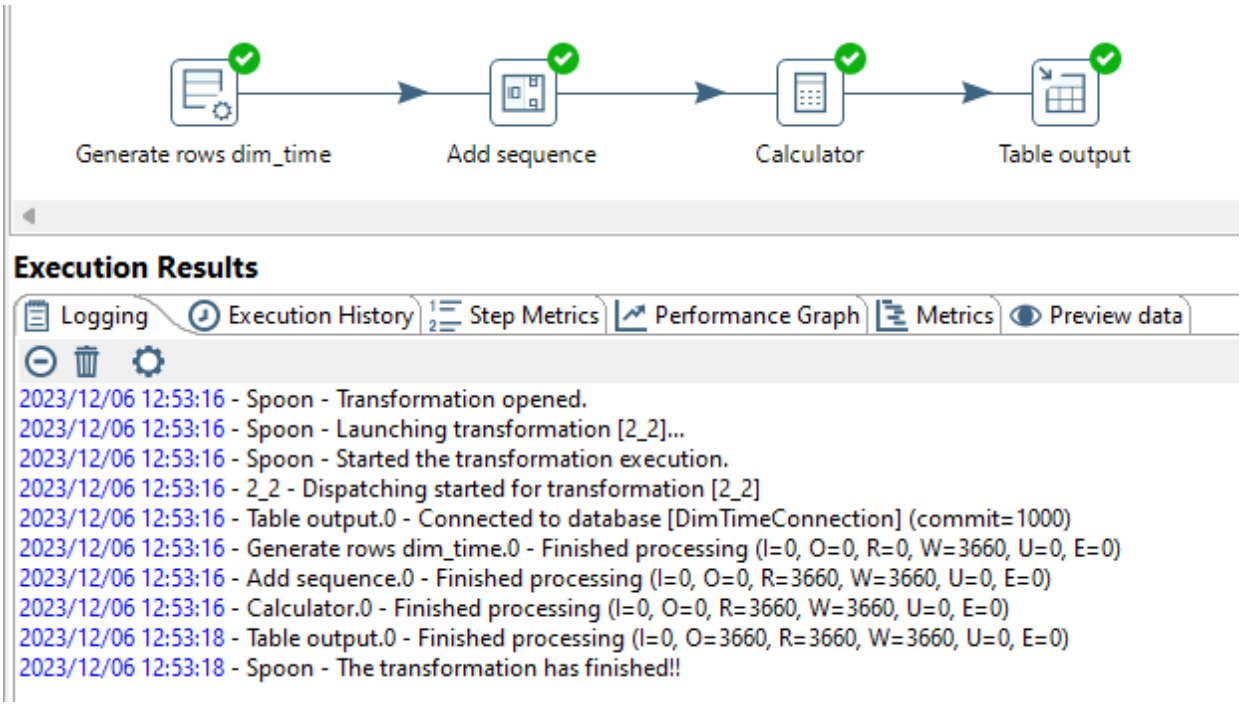
The aim of this part is to create the dimension and the fact tables and fill them with data from the source database using Spoon. This objective will be achieved by carrying out the following tasks:

2.1. Create a data warehouse **calssicmodels_spoon_dw** in MySQL. Provide the SQL code for this task.

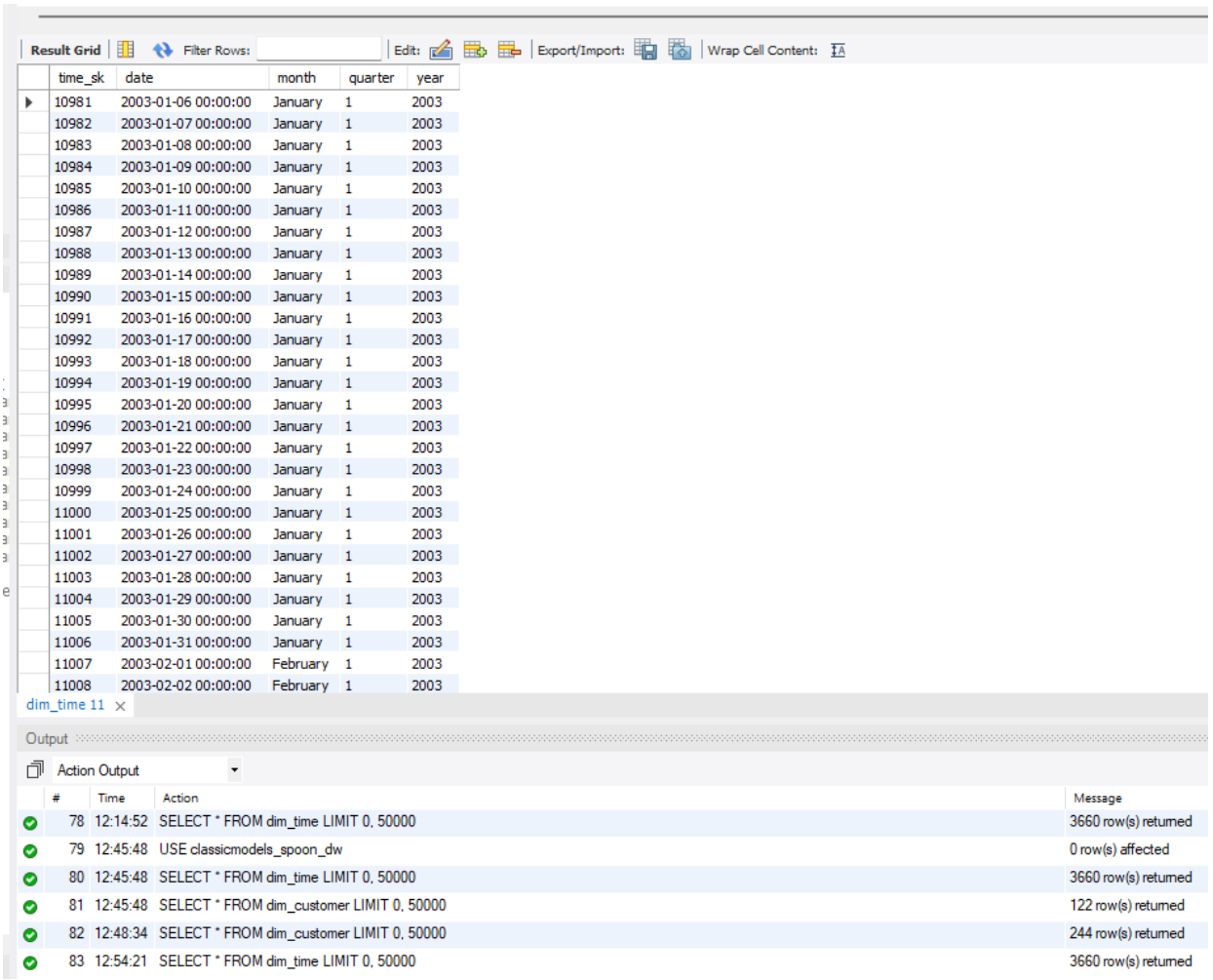
```
CREATE DATABASE classicmodels_spoon_dw;

USE classicmodels_spoon_dw;
```

2.2. Create the **Dim_Time** table and fill it with data by creating a spoon transformation. Name this transformation **2_2.ktr**. The **Dim_Time** table must belong to the data warehouse calssicmodels_spoon_dw.

2.3. Provide screenshots for the spoon transformation in step 2.2. In addition, upload the KTR file **2_2.ktr** on WiseFlow**.**



2.4. Provide a screenshot of the Dim_Time table filled with data.



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

time_sk	date	month	quarter	year
10981	2003-01-06 00:00:00	January	1	2003
10982	2003-01-07 00:00:00	January	1	2003
10983	2003-01-08 00:00:00	January	1	2003
10984	2003-01-09 00:00:00	January	1	2003
10985	2003-01-10 00:00:00	January	1	2003
10986	2003-01-11 00:00:00	January	1	2003
10987	2003-01-12 00:00:00	January	1	2003
10988	2003-01-13 00:00:00	January	1	2003
10989	2003-01-14 00:00:00	January	1	2003
10990	2003-01-15 00:00:00	January	1	2003
10991	2003-01-16 00:00:00	January	1	2003
10992	2003-01-17 00:00:00	January	1	2003
10993	2003-01-18 00:00:00	January	1	2003
10994	2003-01-19 00:00:00	January	1	2003
10995	2003-01-20 00:00:00	January	1	2003
10996	2003-01-21 00:00:00	January	1	2003
10997	2003-01-22 00:00:00	January	1	2003
10998	2003-01-23 00:00:00	January	1	2003
10999	2003-01-24 00:00:00	January	1	2003
11000	2003-01-25 00:00:00	January	1	2003
11001	2003-01-26 00:00:00	January	1	2003
11002	2003-01-27 00:00:00	January	1	2003
11003	2003-01-28 00:00:00	January	1	2003
11004	2003-01-29 00:00:00	January	1	2003
11005	2003-01-30 00:00:00	January	1	2003
11006	2003-01-31 00:00:00	January	1	2003
11007	2003-02-01 00:00:00	February	1	2003
11008	2003-02-02 00:00:00	February	1	2003

dim_time 11

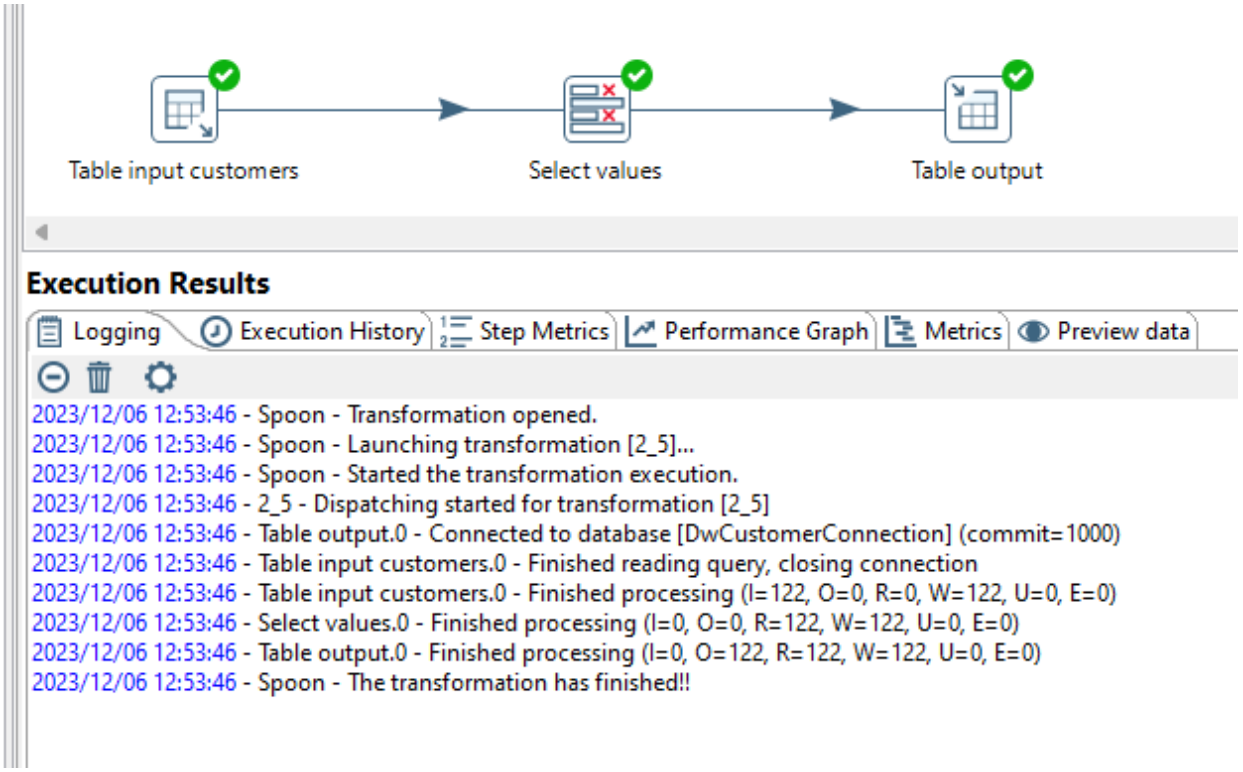
Output

Action Output

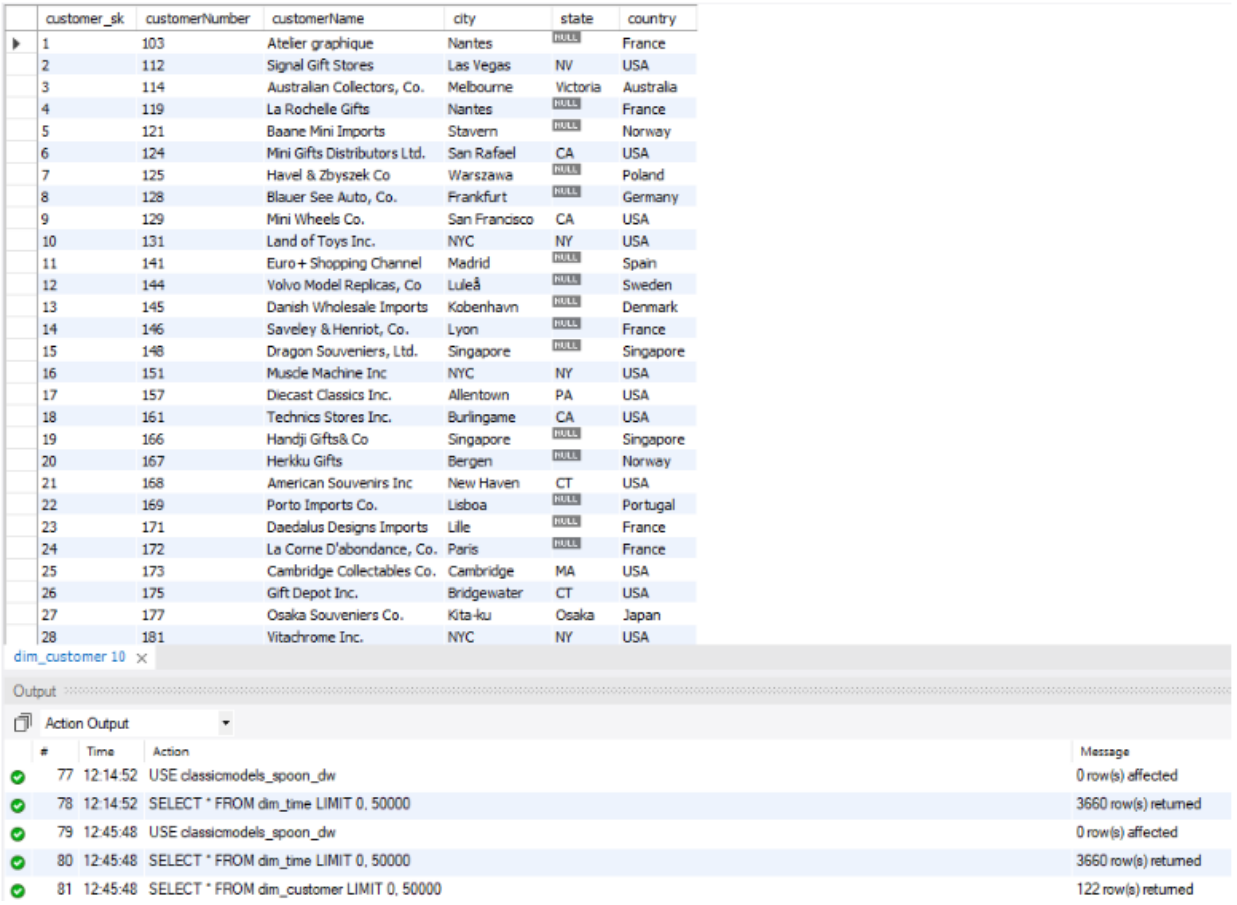
#	Time	Action	Message
78	12:14:52	SELECT * FROM dim_time LIMIT 0, 50000	3660 row(s) returned
79	12:45:48	USE classicmodels_spoon_dw	0 row(s) affected
80	12:45:48	SELECT * FROM dim_time LIMIT 0, 50000	3660 row(s) returned
81	12:45:48	SELECT * FROM dim_customer LIMIT 0, 50000	122 row(s) returned
82	12:48:34	SELECT * FROM dim_customer LIMIT 0, 50000	244 row(s) returned
83	12:54:21	SELECT * FROM dim_time LIMIT 0, 50000	3660 row(s) returned

2.5. Create the **Dim_Customer** table and fill it with data by creating a spoon transformation. Name this transformation **2_5.ktr**. The **Dim_Customer** table must belong to the data warehouse calssicmodels_spoon_dw.

2.6. Provide screenshots for the spoon transformation in step 2.5. In addition, upload the KTR file **2_5.ktr** on WiseFlow**.**



2.7. Provide a screenshot of the **Dim_Customer** filled with data.



customer_sk	customerNumber	customerName	city	state	country
1	103	Atelier graphique	Nantes	NULL	France
2	112	Signal Gift Stores	Las Vegas	NV	USA
3	114	Australian Collectors, Co.	Melbourne	Victoria	Australia
4	119	La Rochelle Gifts	Nantes	NULL	France
5	121	Baane Mini Imports	Stavern	NULL	Norway
6	124	Mini Gifts Distributors Ltd.	San Rafael	CA	USA
7	125	Havel & Zbyszek Co	Warszawa	NULL	Poland
8	128	Blauer See Auto, Co.	Frankfurt	NULL	Germany
9	129	Mini Wheels Co.	San Francisco	CA	USA
10	131	Land of Toys Inc.	NYC	NY	USA
11	141	Euro+ Shopping Channel	Madrid	NULL	Spain
12	144	Volvo Model Replicas, Co	Luleå	NULL	Sweden
13	145	Danish Wholesale Imports	Kobenhavn	NULL	Denmark
14	146	Saveley & Henriot, Co.	Lyon	NULL	France
15	148	Dragon Souveniers, Ltd.	Singapore	NULL	Singapore
16	151	Muscle Machine Inc	NYC	NY	USA
17	157	Diecast Classics Inc.	Allentown	PA	USA
18	161	Technics Stores Inc.	Burlingame	CA	USA
19	166	Handji Gifts& Co	Singapore	NULL	Singapore
20	167	Herkku Gifts	Bergen	NULL	Norway
21	168	American Souvenirs Inc	New Haven	CT	USA
22	169	Porto Imports Co.	Lisboa	NULL	Portugal
23	171	Daedalus Designs Imports	Lille	NULL	France
24	172	La Corne D'abondance, Co.	Paris	NULL	France
25	173	Cambridge Collectables Co.	Cambridge	MA	USA
26	175	Gift Depot Inc.	Bridgewater	CT	USA
27	177	Osaka Souveniers Co.	Kita-ku	Osaka	Japan
28	181	Vitachrome Inc.	NYC	NY	USA

dim_customer 10 ×

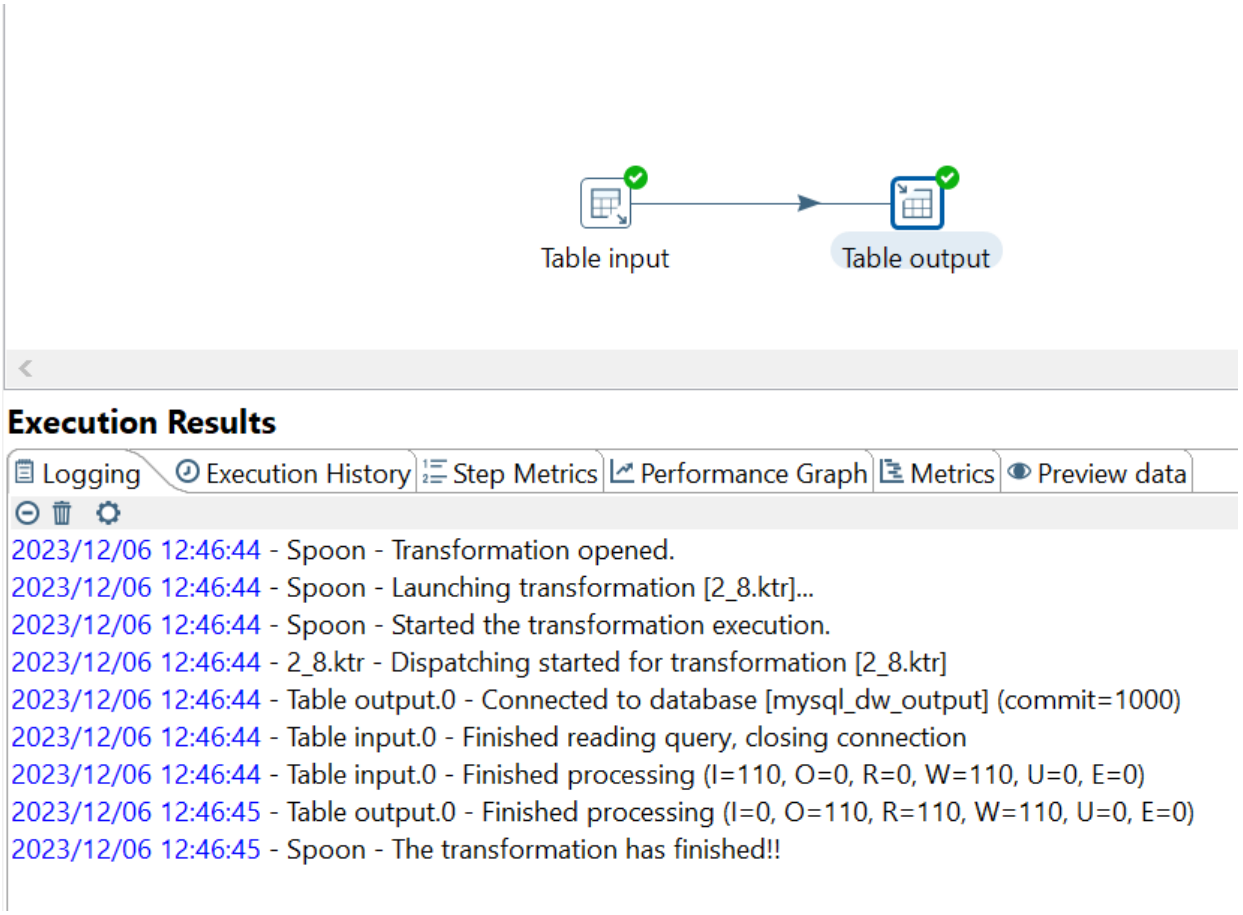
Output

Action Output

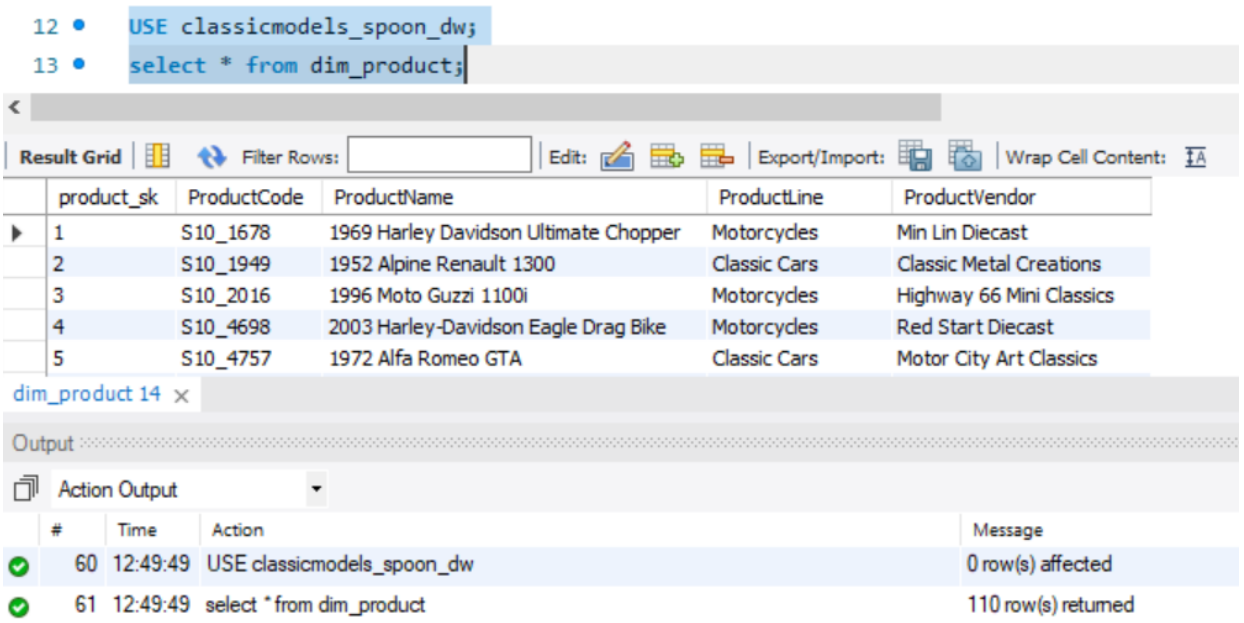
#	Time	Action	Message
77	12:14:52	USE classicmodels_spoon_dw	0 row(s) affected
78	12:14:52	SELECT * FROM dim_time LIMIT 0, 50000	3660 row(s) returned
79	12:45:48	USE classicmodels_spoon_dw	0 row(s) affected
80	12:45:48	SELECT * FROM dim_time LIMIT 0, 50000	3660 row(s) returned
81	12:45:48	SELECT * FROM dim_customer LIMIT 0, 50000	122 row(s) returned

2.8. Create the **Dim_Product** table and fill it with data by creating a spoon transformation. Name this transformation **2_8.ktr**. The **Dim_Product** table must belong to the data warehouse calssicmodels_spoon_dw.

2.9. Provide screenshots for the spoon transformation in step 2.8. In addition, upload the KTR file **2_8.ktr** on WiseFlow**.**



2.10. Provide a screenshot of the **Dim_Product** filled with data.



```
12 •  USE classicmodels_spoon_dw;
13 •  select * from dim_product;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

product_sk	ProductCode	ProductName	ProductLine	ProductVendor
1	S10_1678	1969 Harley Davidson Ultimate Chopper	Motorcycles	Min Lin Diecast
2	S10_1949	1952 Alpine Renault 1300	Classic Cars	Classic Metal Creations
3	S10_2016	1996 Moto Guzzi 1100i	Motorcycles	Highway 66 Mini Classics
4	S10_4698	2003 Harley-Davidson Eagle Drag Bike	Motorcycles	Red Start Diecast
5	S10_4757	1972 Alfa Romeo GTA	Classic Cars	Motor City Art Classics

dim_product 14

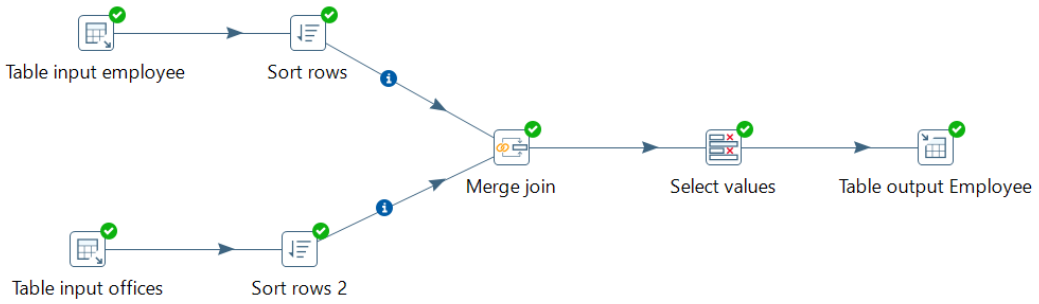
Output

Action Output

#	Time	Action	Message
60	12:49:49	USE classicmodels_spoon_dw	0 row(s) affected
61	12:49:49	select * from dim_product	110 row(s) returned

2.11. Create the **Dim_Employee** table and fill it with data by creating a spoon transformation. Name this transformation **2_11.ktr**. The **Dim_Employee** table must belong to the data warehouse calssicmodels_spoon_dw.

2.12. Provide screenshots for the spoon transformation in step 2.11. In addition, upload the KTR file **2_11.ktr** on WiseFlow**.**

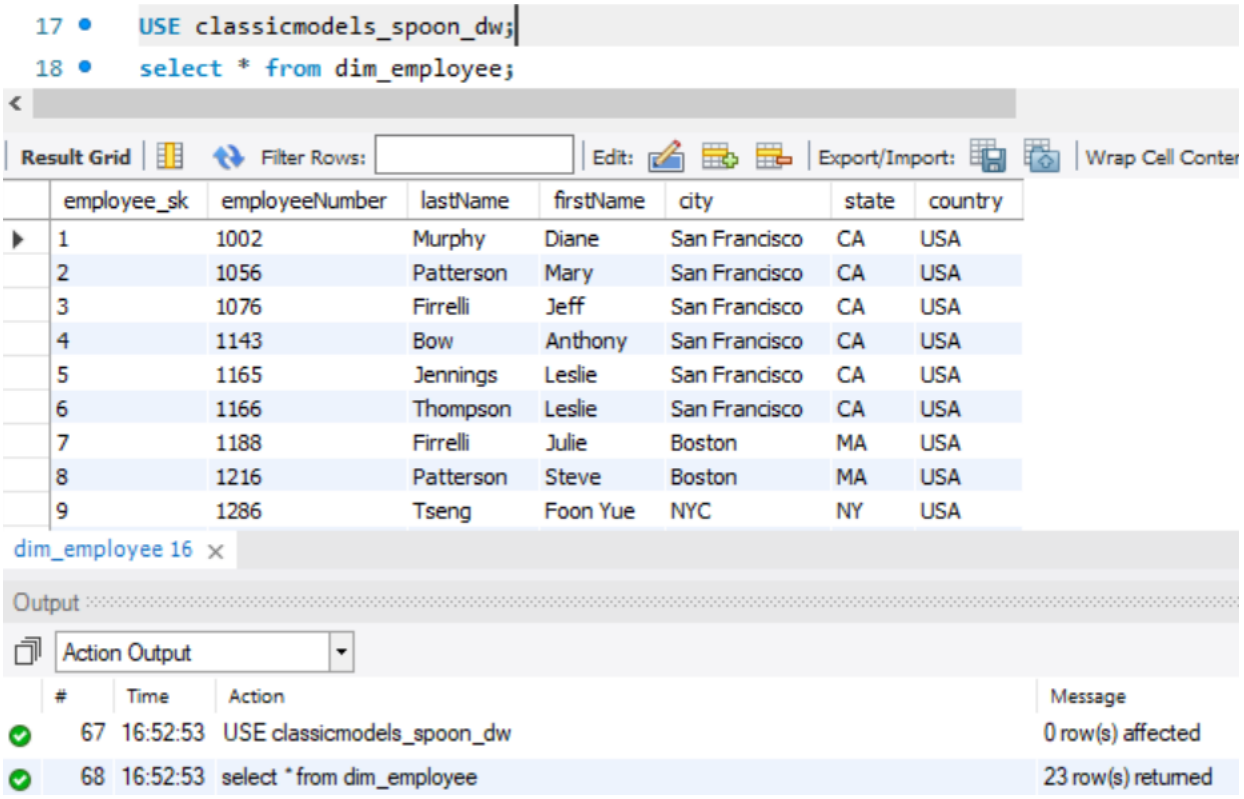


Execution Results

Logging | Execution History | Step Metrics | Performance Graph | Metrics | Preview data

2023/12/06 12:56:34 - Table input employee.0 - Finished reading query, closing connection
2023/12/06 12:56:34 - Table input employee.0 - Finished processing (I=23, O=0, R=0, W=23, U=0, E=0)
2023/12/06 12:56:34 - Table input offices.0 - Finished reading query, closing connection
2023/12/06 12:56:34 - Table input offices.0 - Finished processing (I=7, O=0, R=0, W=7, U=0, E=0)
2023/12/06 12:56:34 - Sort rows 2.0 - Finished processing (I=0, O=0, R=7, W=7, U=0, E=0)
2023/12/06 12:56:34 - Sort rows.0 - Finished processing (I=0, O=0, R=23, W=23, U=0, E=0)
2023/12/06 12:56:35 - Merge join.0 - Finished processing (I=0, O=0, R=30, W=23, U=0, E=0)
2023/12/06 12:56:35 - Select values.0 - Finished processing (I=0, O=0, R=23, W=23, U=0, E=0)
2023/12/06 12:56:35 - Table output Employee.0 - Finished processing (I=0, O=23, R=23, W=23, U=0, E=0)
2023/12/06 12:56:35 - Spoon - The transformation has finished!!

2.13. Provide a screenshot of the **Dim_Employee** filled with data.



```
17 •  USE classicmodels_spoon_dw;
18 •  select * from dim_employee;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Conter

employee_sk	employeeNumber	lastName	firstName	city	state	country
1	1002	Murphy	Diane	San Francisco	CA	USA
2	1056	Patterson	Mary	San Francisco	CA	USA
3	1076	Firrelli	Jeff	San Francisco	CA	USA
4	1143	Bow	Anthony	San Francisco	CA	USA
5	1165	Jennings	Leslie	San Francisco	CA	USA
6	1166	Thompson	Leslie	San Francisco	CA	USA
7	1188	Firrelli	Julie	Boston	MA	USA
8	1216	Patterson	Steve	Boston	MA	USA
9	1286	Tseng	Foon Yue	NYC	NY	USA

dim_employee 16

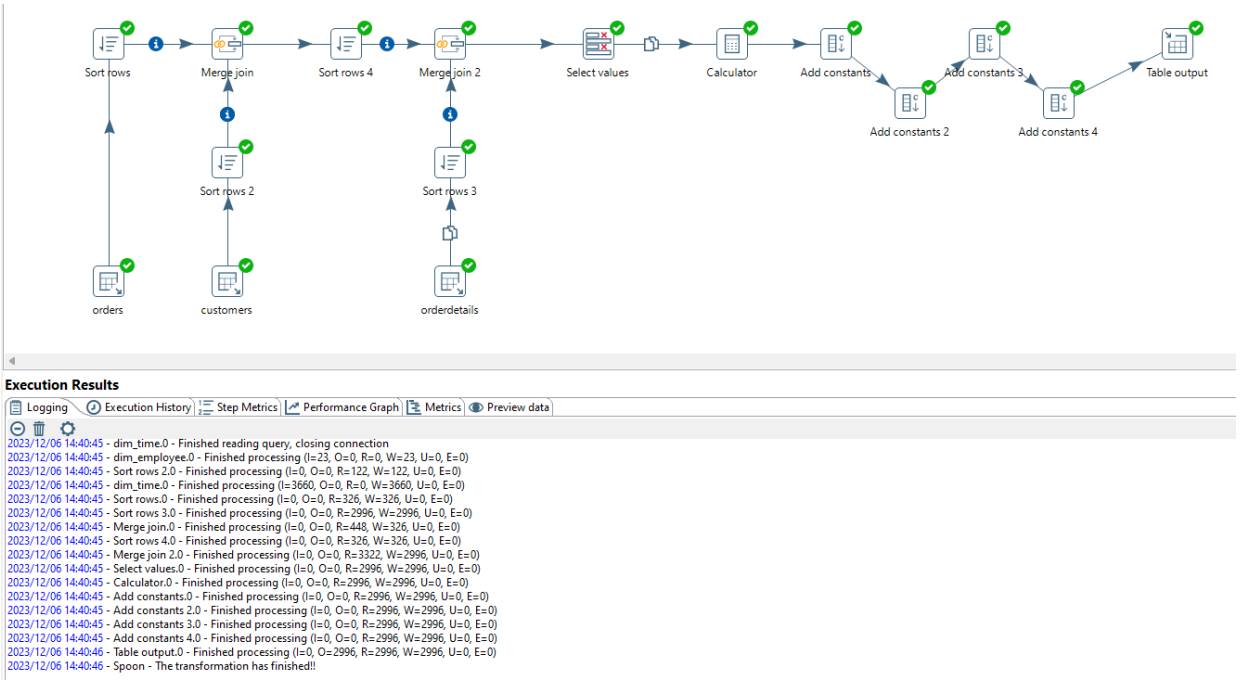
Output

Action Output

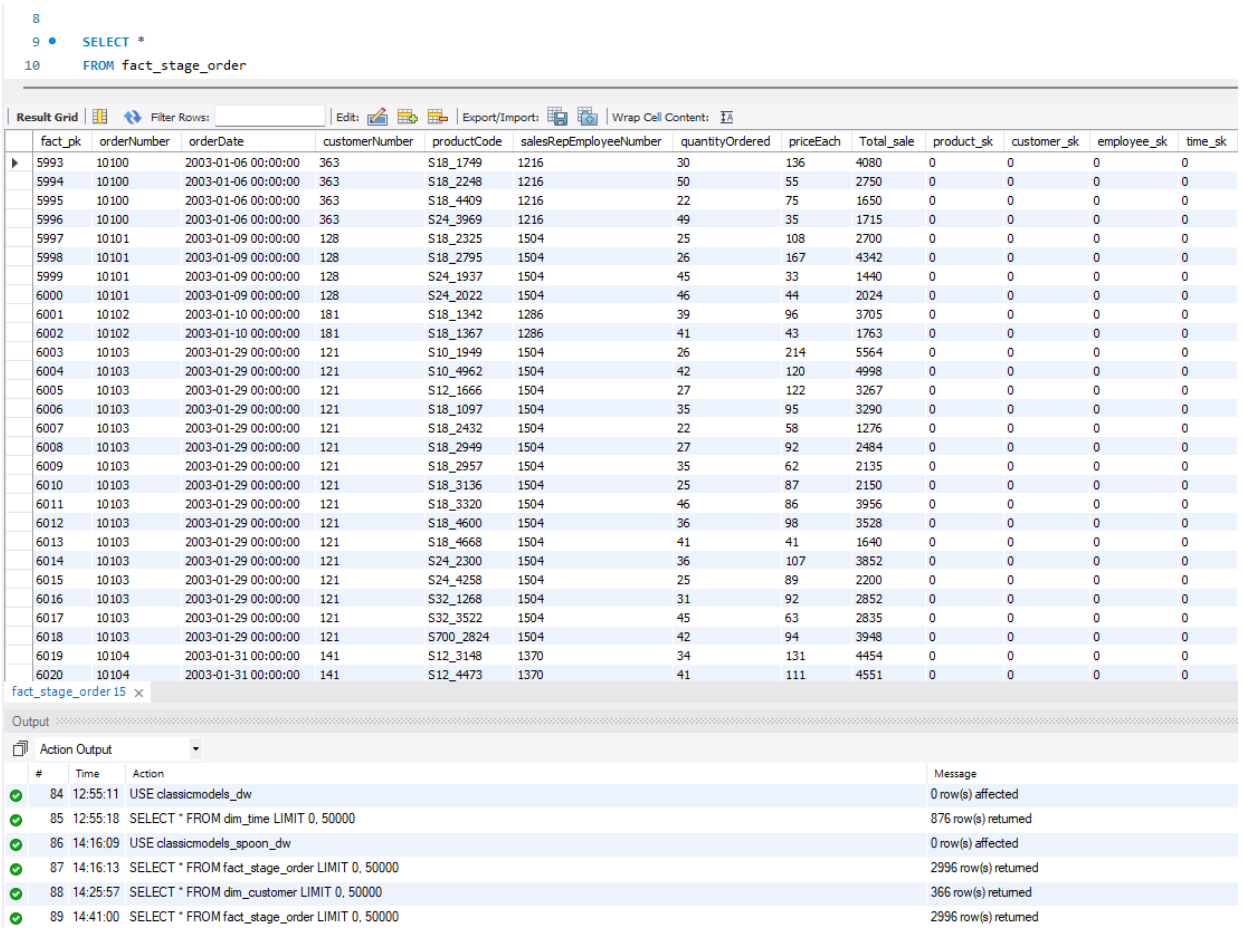
#	Time	Action	Message
67	16:52:53	USE classicmodels_spoon_dw	0 row(s) affected
68	16:52:53	select * from dim_employee	23 row(s) returned

2.14. Create the **Fact_Stage_Order** table and fill it with data by creating a spoon transformation. Name this transformation **2_14.ktr**.

2.15. Provide screenshots for the spoon transformation in step 2.14. In addition, upload the KTR file **2_14.ktr** on WiseFlow**.**



2.16. Provide a screenshot of the **Fact_Stage_Order** filled with data.



```
8
9 •  SELECT *
10   FROM fact_stage_order
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

fact_pk	orderNumber	orderDate	customerNumber	productCode	salesRepEmployeeNumber	quantityOrdered	priceEach	Total_sale	product_sk	customer_sk	employee_sk	time_sk
5993	10100	2003-01-06 00:00:00	363	S18_1749	1216	30	136	4080	0	0	0	0
5994	10100	2003-01-06 00:00:00	363	S18_2248	1216	50	55	2750	0	0	0	0
5995	10100	2003-01-06 00:00:00	363	S18_4409	1216	22	75	1650	0	0	0	0
5996	10100	2003-01-06 00:00:00	363	S24_3969	1216	49	35	1715	0	0	0	0
5997	10101	2003-01-09 00:00:00	128	S18_2325	1504	25	108	2700	0	0	0	0
5998	10101	2003-01-09 00:00:00	128	S18_2795	1504	26	167	4342	0	0	0	0
5999	10101	2003-01-09 00:00:00	128	S24_1937	1504	45	33	1440	0	0	0	0
6000	10101	2003-01-09 00:00:00	128	S24_2022	1504	46	44	2024	0	0	0	0
6001	10102	2003-01-10 00:00:00	181	S18_1342	1286	39	96	3705	0	0	0	0
6002	10102	2003-01-10 00:00:00	181	S18_1367	1286	41	43	1763	0	0	0	0
6003	10103	2003-01-29 00:00:00	121	S10_1949	1504	26	214	5564	0	0	0	0
6004	10103	2003-01-29 00:00:00	121	S10_4962	1504	42	120	4998	0	0	0	0
6005	10103	2003-01-29 00:00:00	121	S12_1666	1504	27	122	3267	0	0	0	0
6006	10103	2003-01-29 00:00:00	121	S18_1097	1504	35	95	3290	0	0	0	0
6007	10103	2003-01-29 00:00:00	121	S18_2432	1504	22	58	1276	0	0	0	0
6008	10103	2003-01-29 00:00:00	121	S18_2949	1504	27	92	2484	0	0	0	0
6009	10103	2003-01-29 00:00:00	121	S18_2957	1504	35	62	2135	0	0	0	0
6010	10103	2003-01-29 00:00:00	121	S18_3136	1504	25	87	2150	0	0	0	0
6011	10103	2003-01-29 00:00:00	121	S18_3320	1504	46	86	3956	0	0	0	0
6012	10103	2003-01-29 00:00:00	121	S18_4600	1504	36	98	3528	0	0	0	0
6013	10103	2003-01-29 00:00:00	121	S18_4668	1504	41	41	1640	0	0	0	0
6014	10103	2003-01-29 00:00:00	121	S24_2300	1504	36	107	3852	0	0	0	0
6015	10103	2003-01-29 00:00:00	121	S24_4258	1504	25	89	2200	0	0	0	0
6016	10103	2003-01-29 00:00:00	121	S32_1268	1504	31	92	2852	0	0	0	0
6017	10103	2003-01-29 00:00:00	121	S32_3522	1504	45	63	2835	0	0	0	0
6018	10103	2003-01-29 00:00:00	121	S700_2824	1504	42	94	3948	0	0	0	0
6019	10104	2003-01-31 00:00:00	141	S12_3148	1370	34	131	4454	0	0	0	0
6020	10104	2003-01-31 00:00:00	141	S12_4473	1370	41	111	4551	0	0	0	0

fact_stage_order 15 ×

Output

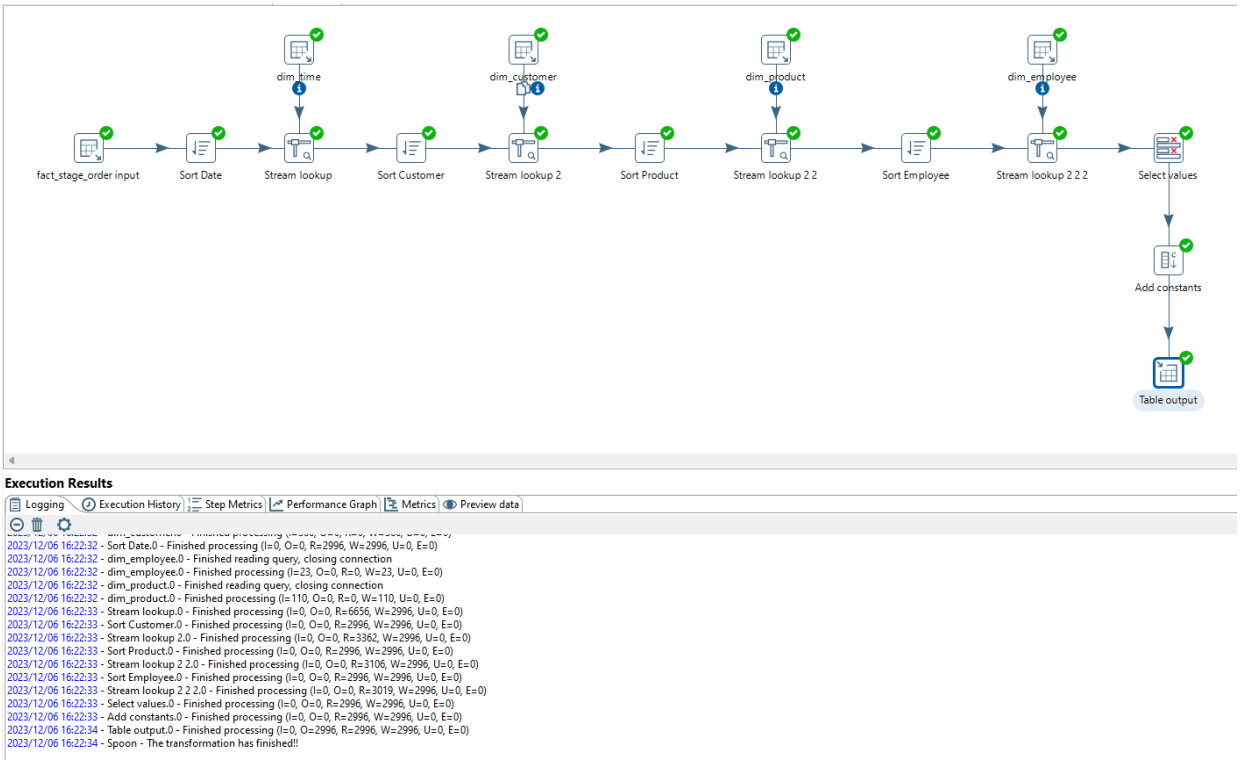
Action Output

#	Time	Action	Message
84	12:55:11	USE classicmodels_dw	0 row(s) affected
85	12:55:18	SELECT * FROM dim_time LIMIT 0, 50000	876 row(s) returned
86	14:16:09	USE classicmodels_spoon_dw	0 row(s) affected
87	14:16:13	SELECT * FROM fact_stage_order LIMIT 0, 50000	2996 row(s) returned
88	14:25:57	SELECT * FROM dim_customer LIMIT 0, 50000	366 row(s) returned
89	14:41:00	SELECT * FROM fact_stage_order LIMIT 0, 50000	2996 row(s) returned

2.17. Create the **Fact_Order** table and fill it with data using the **Fact_Stage_Order** table. Do this task by creating a spoon transformation.

2.18. In the same spoon transformation in step 2.17, add the surrogate keys to the **Fact_Order** table.

2.19. Provide the KTR file for the transformation in steps 2.17 and 2.18. Name this KTR file as **2_19.ktr.** In addition, upload the KTR file **2_19.ktr** on WiseFlow**.**



Execution Results

Logging | Execution History | Step Metrics | Performance Graph | Metrics | Preview data

```
2023/12/06 16:22:32 - Sort Date.0 - Finished processing (I=0, O=0, R=2996, W=2996, U=0, E=0)
2023/12/06 16:22:32 - dim_employee.0 - Finished reading query, closing connection
2023/12/06 16:22:32 - dim_employee.0 - Finished processing (I=23, O=0, R=0, W=23, U=0, E=0)
2023/12/06 16:22:32 - dim_product.0 - Finished reading query, closing connection
2023/12/06 16:22:32 - dim_product.0 - Finished processing (I=110, O=0, R=0, W=110, U=0, E=0)
2023/12/06 16:22:33 - Stream lookup.0 - Finished processing (I=0, O=0, R=6656, W=2996, U=0, E=0)
2023/12/06 16:22:33 - Sort Customer.0 - Finished processing (I=0, O=0, R=2996, W=2996, U=0, E=0)
2023/12/06 16:22:33 - Stream lookup 2.0 - Finished processing (I=0, O=0, R=3362, W=2996, U=0, E=0)
2023/12/06 16:22:33 - Sort Product.0 - Finished processing (I=0, O=0, R=2996, W=2996, U=0, E=0)
2023/12/06 16:22:33 - Stream lookup 2 2.0 - Finished processing (I=0, O=0, R=3106, W=2996, U=0, E=0)
2023/12/06 16:22:33 - Sort Employee.0 - Finished processing (I=0, O=0, R=2996, W=2996, U=0, E=0)
2023/12/06 16:22:33 - Stream lookup 2 2 2.0 - Finished processing (I=0, O=0, R=3019, W=2996, U=0, E=0)
2023/12/06 16:22:33 - Select values.0 - Finished processing (I=0, O=0, R=2996, W=2996, U=0, E=0)
2023/12/06 16:22:33 - Add constants.0 - Finished processing (I=0, O=0, R=2996, W=2996, U=0, E=0)
2023/12/06 16:22:34 - Table output.0 - Finished processing (I=0, O=2996, R=2996, W=2996, U=0, E=0)
2023/12/06 16:22:34 - Spoon - The transformation has finished!!
```

2.20. Provide a screenshot of the Fact_Order filled with data.

```
12 •  SELECT *
13    FROM fact_order
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

orderNumber	orderDate	customerNumber	productCode	salesRepEmployeeNumber	quantityOrdered	priceEach	Total_sale	fact_pk	product_sk	customer_sk	employee_sk	time_sk
10201	2003-12-01 00:00:00	129	S10_1678	1165	22	82	1804	13049	1	253	5	11310
10168	2003-10-28 00:00:00	161	S10_1678	1165	36	95	3384	12707	1	262	5	11276
10159	2003-10-10 00:00:00	321	S10_1678	1165	49	81	3969	12610	1	313	5	11258
10312	2004-10-21 00:00:00	124	S10_1949	1165	48	214	10272	14077	2	250	5	11635
10357	2004-12-10 00:00:00	124	S10_1949	1165	32	199	6368	14505	2	250	5	11685
10333	2004-11-18 00:00:00	129	S10_1949	1165	26	189	4888	14302	2	253	5	11663
10140	2003-07-24 00:00:00	161	S10_1949	1165	37	186	6882	12441	2	262	5	11180
10381	2005-02-17 00:00:00	321	S10_1949	1165	36	182	6552	14724	2	313	5	11754
10201	2003-12-01 00:00:00	129	S10_2016	1165	24	117	2784	13050	3	253	5	11310
10168	2003-10-28 00:00:00	161	S10_2016	1165	27	98	2619	12708	3	262	5	11276
10159	2003-10-10 00:00:00	321	S10_2016	1165	37	101	3737	12611	3	313	5	11258
10201	2003-12-01 00:00:00	129	S10_4698	1165	49	192	9359	13051	4	253	5	11310
10168	2003-10-28 00:00:00	161	S10_4698	1165	20	161	3200	12709	4	262	5	11276
10362	2005-01-05 00:00:00	161	S10_4698	1165	22	182	4004	14569	4	262	5	11711
10159	2003-10-10 00:00:00	321	S10_4698	1165	22	170	3740	12612	4	313	5	11258
10384	2005-02-23 00:00:00	321	S10_4757	1165	34	129	4386	14759	5	313	5	11760
10400	2005-04-01 00:00:00	450	S10_4757	1165	64	135	8576	14890	5	347	5	11797
10229	2004-03-11 00:00:00	124	S10_4962	1165	50	139	6900	13323	6	250	5	11411
10357	2004-12-10 00:00:00	124	S10_4962	1165	43	136	5805	14506	6	250	5	11685
10140	2003-07-24 00:00:00	161	S10_4962	1165	26	131	3406	12442	6	262	5	11180
10381	2005-02-17 00:00:00	321	S10_4962	1165	37	139	5106	14725	6	313	5	11754
10135	2003-07-02 00:00:00	124	S12_1099	1165	42	173	7266	12396	7	250	5	11158
10159	2003-10-10 00:00:00	321	S12_1099	1165	41	189	7708	12613	7	313	5	11258
10142	2003-08-08 00:00:00	124	S12_1108	1165	33	166	5478	12461	8	250	5	11195
10282	2004-08-20 00:00:00	124	S12_1108	1165	41	177	7216	13777	8	250	5	11573
10371	2005-01-23 00:00:00	124	S12_1108	1165	32	179	5696	14630	8	250	5	11729
10382	2005-02-17 00:00:00	124	S12_1108	1165	34	166	5644	14733	8	250	5	11754
10113	2003-03-26 00:00:00	124	S12_1666	1165	21	122	2541	12222	9	250	5	11060

fact_order 28 ×

Output

Action Output

#	Time	Action	Message
104	16:14:47	SELECT * FROM dim_customer LIMIT 0, 50000	366 row(s) returned
105	16:14:47	Drop table if exists fact_order	0 row(s) affected, 1 warning(s): 1051 Unknown table 'classicm
106	16:14:47	SELECT * FROM fact_stage_order LIMIT 0, 50000	2996 row(s) returned
107	16:17:34	SELECT * FROM fact_order LIMIT 0, 50000	2996 row(s) returned
108	16:20:57	SELECT * FROM fact_order LIMIT 0, 50000	2996 row(s) returned
109	16:22:37	SELECT * FROM fact_order LIMIT 0, 50000	2996 row(s) returned

Part 3. Power BI

3.1. The file **Dashboard_0.pbix** contains already some loaded data but does not contain any visuals. Your task is to create 10 different visuals using the data in **Dashboard_0.pbix**. To ensure the clarity of the visuals, the final dashboard should have 5 pages. Each page should contain 2 visuals. So, in total you will create 10 different visuals.

3.2. After creating the 10 visuals save the document as **Dashboard_final.pbix** and upload it on WiseFlow.

Save the report containing your answers as a pdf file for delivery in WiseFlow. Remember to upload the KTR files and the Power BI file as attachments in WiseFlow.