SANDMILES GROUP DATABASE PROJECT

Abstract

Sandmiles Group is a Canadian wholesale electronics company founded in 2019. Since inception, the company has used a file-based database system. In the early years of business, this was effective and efficient. However, the company has grown and so has its data. The manager has noticed that with the file-based database system, keeping track of the company's operations has become more time-consuming and there have been a lot of errors. There have also been security issues as well as data redundancy issues. All these have prevented the company from using its data efficiently to drive decision making.

The manager wants to improve:

- data access
- Data integrity
- End user productivity
- Data security

Overall, she wants Sandmiles Group to develop a data driven culture in order to improve efficiency, customer satisfaction, staff engagement and make better informed decisions. To achieve this, the manager has requested that the company's file-based database be converted to a Relational Database Management System. We are therefore designing a database for the Employees, Department, Customers, Products and Sales.

MISSION STATEMENT

The purpose of Sandmiles Group Database is to maintain the data generated by the company's daily operations in order to improve the retail sales business, customer satisfaction, staff engagement and aid business leaders in making evidence-based decisions.

OBJECTIVES

- Maintain complete information on customer orders.
- Maintain complete information on employees responsible for the orders.
- To keep track of customer orders from the point of order to delivery.
- To produce information on customer orders

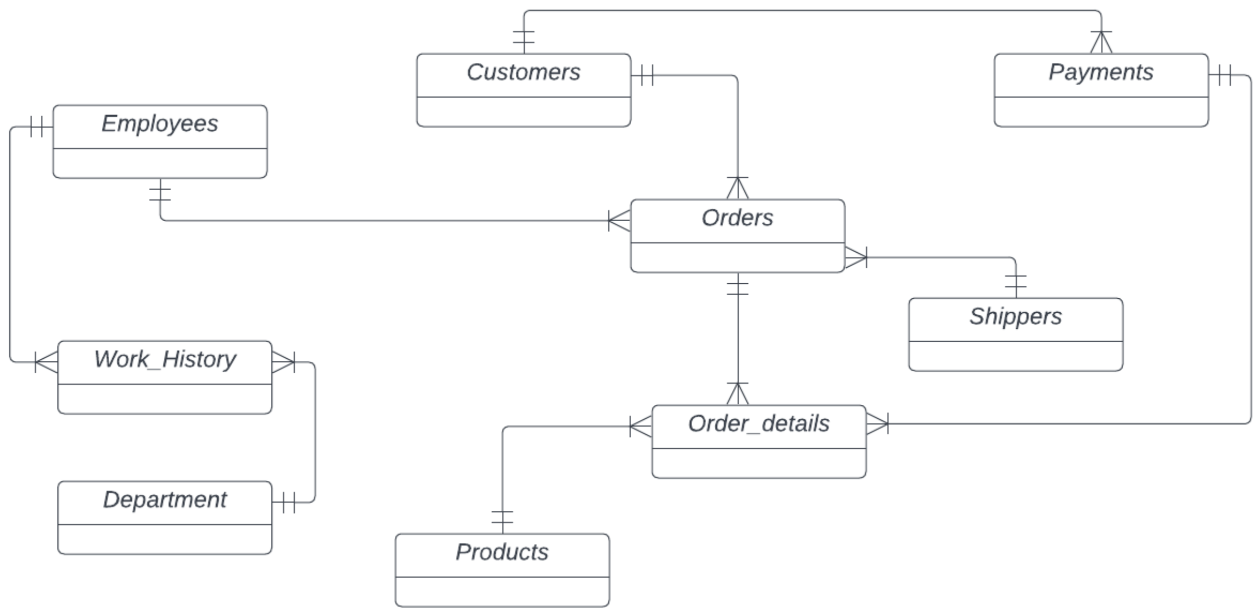
REQUIREMENTS

- To maintain (add, update and delete) data on orders
- To maintain (add, update and delete) data on products
- To maintain (add, update and delete) data on shippers
- To maintain (add, update and delete) data on our customers
- To maintain (add, update and delete) data on employees
- To maintain (add, update and delete) data on payment

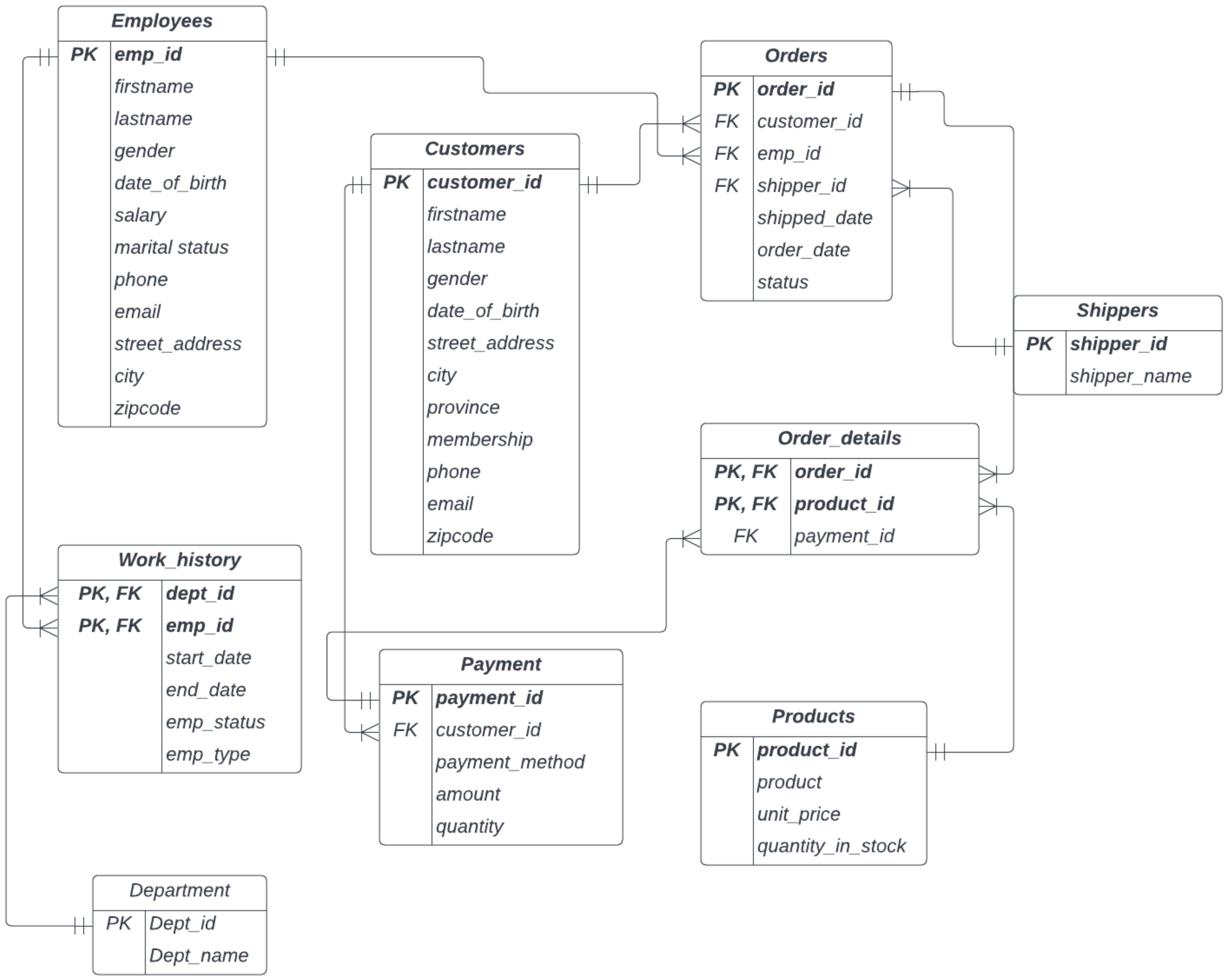
- To perform searches/details on orders
- To perform searches/details on products
- To perform searches/details on shippers
- To perform searches/ details on customers
- To perform searches/details on employees
- To perform searches/details on payment

- To report on orders
- To report on products
- To report on shippers
- To report on customers
- To report on employees
- To report on payment

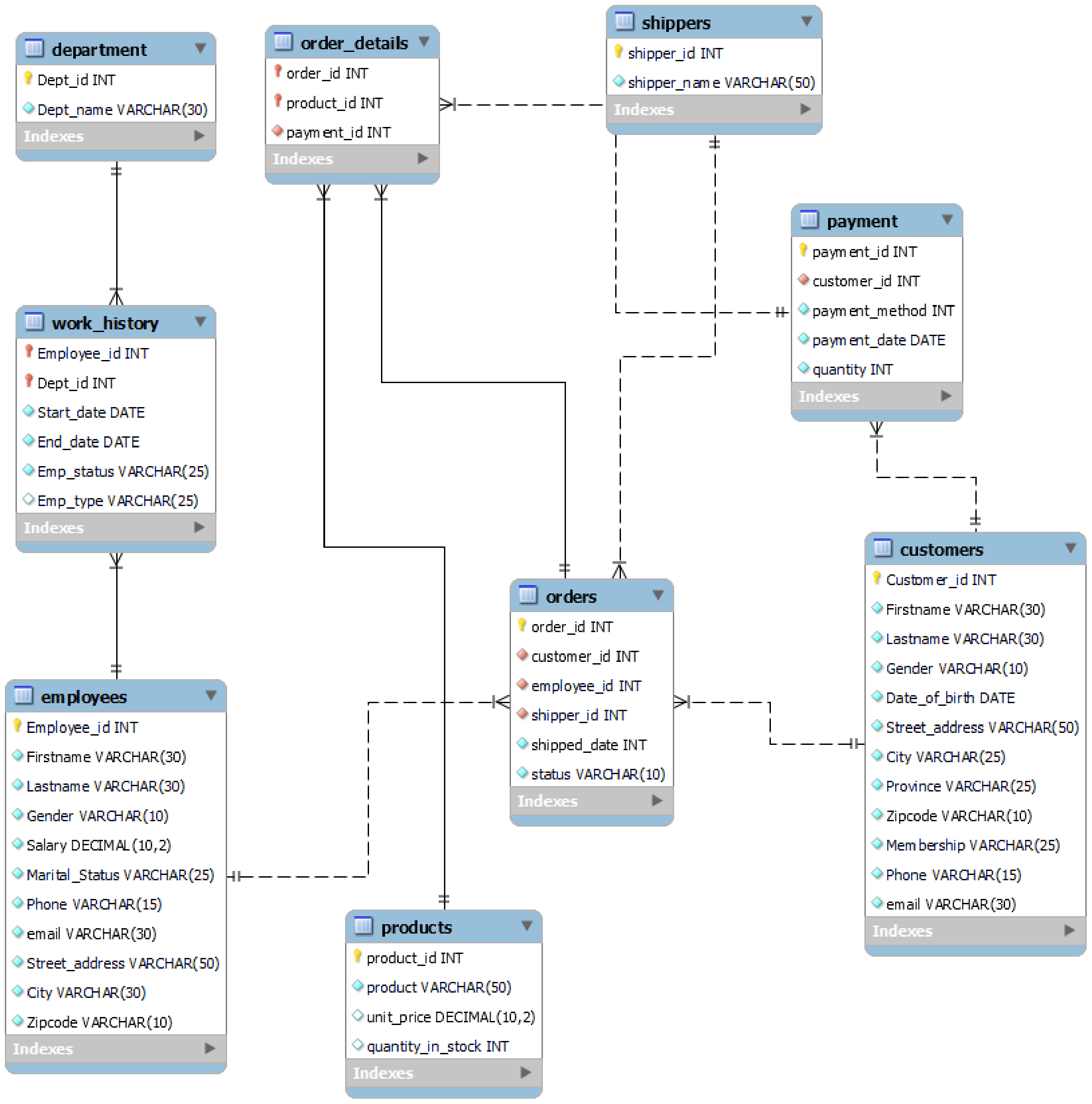
CONCEPTUAL MODEL



ER DIAGRAM (LOGICAL MODEL)



MySQL ER DIAGRAM



LOGICAL DESIGN (Relational Model)

The schemas below show the entities for the Sandmiles Group database system. The primary keys are underlined, and the foreign keys are in red.

Employees (employee_id, firstname, lastname, gender, salary, marital status, phone, email, street_address, city, zipcode)

Customers (customer_id, firstname, lastname, gender, date_of_birth, street_address, city, province, zipcode, membership, phone, email)

Work_history (employee_id, dept_id, start_date, end_date, emp_status, emp_type)

Department (dept_id, dept_name)

Orders (order_id, customer_id, employee_id, shipper_id, order_date, status)

Order_details (order_id, product_id, payment_id)

Payment (payment_id, customer_id, payment_method, payment_date, quantity)

Products (product_id, product, unit_price, quantity_in_stock)

Shippers (shipper_id, shipper_name)

Employee Relation

Employee relations contains records of all employees in Sandmiles Group.

Employees (emp_id, firstname, lastname, gender, date_of_birth, salary, marital status, phone, email, street_address, city, zipcode)

Key constraints: Primary Key is emp_id , phone is a candidate key.

Referential integrity constraints: This relation contains no foreign keys.

NULL constraints: The emp_id, firstname, lastname, zipcode, salary and phone cannot be null.

EMPLOYEE RELATIONS		
ATTRIBUTE	**DOMAIN MEANING**	**DATA TYPE**
Employee_id	All employee identification numbers	Integer
Firstname	All employees' first names	Varchar 30
Lastname	All employee last names	Varchar 30
Gender	All employees' gender	Varchar 10
Date_of_birth	All employee birth date	Date
Salary	All employees' annual gross pay	Decimal (10, 2)
Marital status	All employees' marital status	Varchar 25
Phone	All employees' phone numbers	Varchar 15
email	All employees' email addresses	Varchar 30
Street_address	All employees' home addresses	Varchar 50
City	All employees' city of residence	Varchar 30
Zipcode	All employees' residential zipcode	Varchar 10

Customers Relation

Customer Relations contains records of all customers of Sandmiles Group.

Customers (customer_id, firstname, lastname, gender, date_of_birth, street_address, city, province, membership, phone, email, zipcode)

Key constraints: customer_id is the primary key of this relation. Phone is a candidate key
Referential integrity constraints: There is no foreign key in this table.
NULL constraints: customer_id, firstname, lastname, zipcode and phone cannot be null.

CUSTOMERS RELATIONS		
ATTRIBUTE	**DOMAIN MEANING**	**DATA TYPE**
Customer_id	All customers' identification numbers	Integer
Firstname	All customers' first names	Varchar 30
Lastname	All customers' last names	Varchar 30
Gender	All customers' gender	Varchar 10
Date_of_birth	All customers' birth date	Date
Street_address	All customers' home addresses (street)	Varchar 50
City	All customers' city of residence	Varchar 25
Province	All customers' province of residence	Varchar 25
Zipcode	All customers' residential address zipcode	Varchar 10
Membership	All customers' membership type	Varchar 25
Phone	All customers' phone numbers	Varchar 25
Email	All customers' email addresses	Varchar 50

Work_History Relation

Work_history relations contain records of all employee's work history in the Sandmiles Group.
Work_history (emp_id, dept_id, start_date, end_date, emp_status, emp_type)
Key constraints: emp_id and dept_id form the composite primary key in the relation. There is no candidate key for work_history relations.
Referential integrity constraints: emp_id and dept_id are the foreign keys in this relation.
NULL constraints: emp_id and dept_id cannot be Null.

WORK_HISTORY RELATIONS		
ATTRIBUTE	**DOMAIN MEANING**	**DATA TYPE**
Employee_id	All employees' identification numbers	Integer
Dept_id	All departments' identification numbers	Integer
Start_date	All possible employee resumption dates	Date
End_date	All possible employee termination of employment date	Date
Emp_status	All employment status showing active or inactive	Varchar 25
Emp_type	All employment types	Varchar 25

Department Relation

Department relations contains records of all the departments in Sandmiles Group.
Departments (dept_id, dept_name)
Key constraints: Dept_id is the primary key. No candidate key in Department relation.
Referential constraints: there is no foreign key in this relation.

NULL constraint: The dept_id and dept_name cannot be Null.

DEPARTMENT RELATIONS		
ATTRIBUTE	DOMAIN MEANING	DATA TYPE
Dept_id	All department identification numbers	Integer
Dept_name	All department names	Varchar 50

Orders Relation

Orders relations contains records of all orders from customers in Sandmiles Group
Orders (order_id, customer_id, emp_id, shipper_id, shipped_date, order_date, status)
Key constraints: order_id is the primary key.
Referential constraints: customer_id, emp_id and shipper_id are foreign keys in the Orders Relation.
NULL constraint: order_id, customer_id, emp_id, shipper_id, payment_id, order_date, shipped_date and status cannot be Null.

ORDERS RELATIONS		
ATTRIBUTE	DOMAIN MEANING	DATA TYPE
Order_id	All order identification numbers	Integer
Customer_id	All customer identification numbers	Integer
Employee_id	All employee identification numbers	Integer
Shipper_id	All shipper identification numbers	Integer
Shipped_date	All shipping dates	Date
Order_date	All order dates	Date
status	All status of orders placed	Varchar 50

Orders Details Relation

Order Details contains a record of the order details of Sandmiles Group’s customers.
Order_details (order_id, product_id, payment_id)
Key constraints: order_id and product_id form the composite primary key.
Referential constraints: order_id, product_id and payment_id are foreign keys in the Orders_Details Relation.
NULL constraint: order_id, product_id, payment_id cannot be Null.

ORDER DETAILS RELATIONS		
ATTRIBUTE	DOMAIN MEANING	DATA TYPE
Order_id	All order identification numbers	Integer
Product_id	All product identification numbers	Integer
Payment_id	All payment identification numbers	Integer

Payment Relation

Payment Relations contains records of all customer payments for orders in Sandmiles Group.
Payment (payment_id, customer_id, payment_method, payment_date, quantity)

Key constraints: payment_id is the primary key.
Referential constraints: customer_id is the Foreign Key in Payment Relations.
NULL constraint: payment_id, customer_id, payment_date cannot be Null.

PAYMENT RELATIONS		
ATTRIBUTE	**DOMAIN MEANING**	**DATA TYPE**
Payment_id	All payment identification numbers	Integer
Customer_id	All customer identification numbers	Integer
Payment_method	All payment methods	Varchar 50
Payment_date	All dates of payments	Date
Quantity	All possible quantities of products ordered	Integer

Product Relation

Payment Relations contains a record of all products stocked and sold by Sandmiles Group.
Products (product_id, product, unit_price, quantity_in_stock)
Key constraints: product_id is the primary key.
Referential constraints: No Foreign Keys in Product Relations.
NULL constraint: payment_id, customer_id, amount, payment_date cannot be Null.

PRODUCT RELATIONS		
ATTRIBUTE	**DOMAIN MEANING**	**DATA TYPE**
Product_id	All product identification numbers	Integer
product	All products	Varchar 50
Unit_price	All prices for products	Decimal (10, 2)
Quantity_in_stock	All available quantity of products	Integer

Shippers Relation

Shippers Relations contains records of all shippers working with Sandmiles Group.
Shippers (shipper_id, shipper_name)
Key constraints: shipper_id is the primary key.
Referential constraints: No Foreign Keys in Product Relations.
NULL constraint: shipper_id and shipper_name cannot be Null.

SHIPPER RELATIONS		
ATTRIBUTES	**DOMAIN MEANING**	**DATA TYPE**
Shipper_id	All shipper identification numbers	Integer
Shipper_name	All shipper names	Varchar 50

NORMALIZATION: SCHEMA REFINEMENT

Employee Relation

Employees (emp_id, firstname, lastname, gender, date_of_birth, salary, marital status, phone, email, street_address, city, zipcode)

Employee Relation is in the 2NF and not the 3NF because zipcode, street_address and city have transitive functional dependency.

Zipcode -> city -> street_address

To therefore convert this table to the 3rd Normal Form, we would place both attributes in a new table called employee_address. The new tables are:

Employees(emp_id, firstname, lastname, gender, date_of_birth, salary, marital status, phone, email, zipcode

Employee_address (zipcode, street_address, city)

Customers Relation

Customers (customer_id, firstname, lastname, gender, date_of_birth, street_address, city, province, membership, phone, email, zipcode)

Customers Relation is in the 2NF and not the 3NF because zipcode, street_address, city and province have transitive functional dependency.

Zipcode -> province -> city -> street_address

To therefore convert this table to the 3rd Normal Form, we would place both attributes in a new table called customers_address. The new tables are:

Customers (customer_id, firstname, lastname, gender, date_of_birth, membership, phone, email)

Customers_address (zipcode, street_address, city, province)

Work_History Relation

Work_history relations is in the 3NF.

Work_history (emp_id, dept_id, start_date, end_date, emp_status, emp_type)

Department Relation

Department Relation is in the 3NF.

Department (dept_id, dept_name)

Orders Relation

Orders Relations is in the 3NF.

Orders (order_id, customer_id, emp_id, shipper_id, shipped_date, order_date, status)

Orders Details Relation

Order_details Relation is in the 3NF

Order_details (order_id, product_id, payment_id)

Payment Relation

Payment Relation is in the 3NF.

Payment (<u>payment_id,</u> customer_id, payment_method, amount, payment_date, quantity)

Product Relation

Product Relation is in the 3NF.

Products (<u>product_id,</u> product, unit_price, quantity_in_stock)

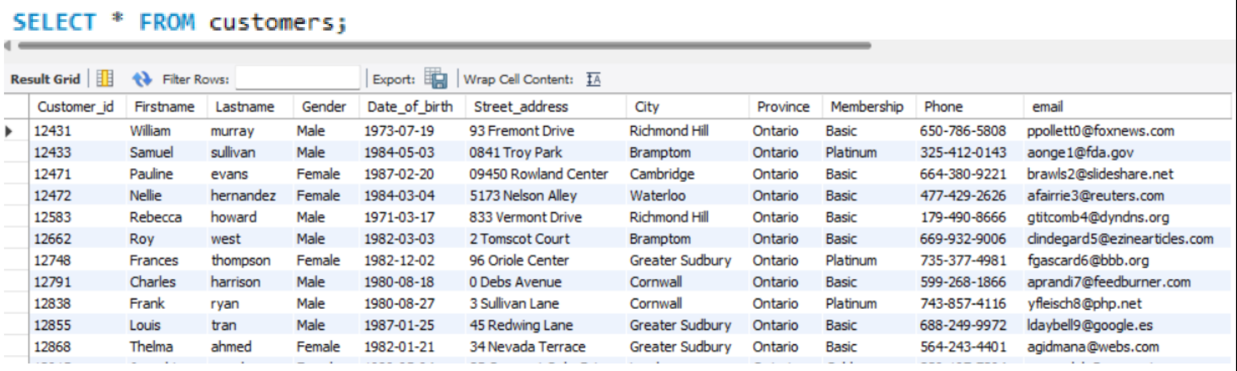
Shippers Relation

Shippers Relation is in the 3NF.

Shippers <u>(shipper_id,</u> shipper_name)

RELATIONS IN MySQL

CUSTOMERS RELATIONS

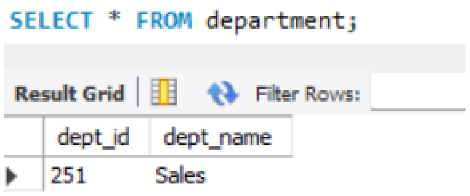


```
SELECT * FROM customers;
```

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

Customer_id	Firstname	Lastname	Gender	Date_of_birth	Street_address	City	Province	Membership	Phone	email
12431	William	murray	Male	1973-07-19	93 Fremont Drive	Richmond Hill	Ontario	Basic	650-786-5808	ppollett0@foxnews.com
12433	Samuel	sullivan	Male	1984-05-03	0841 Troy Park	Brampton	Ontario	Platinum	325-412-0143	aonge1@fda.gov
12471	Pauline	evans	Female	1987-02-20	09450 Rowland Center	Cambridge	Ontario	Basic	664-380-9221	brawls2@slideshare.net
12472	Nellie	hernandez	Female	1984-03-04	5173 Nelson Alley	Waterloo	Ontario	Basic	477-429-2626	afairrie3@reuters.com
12583	Rebecca	howard	Male	1971-03-17	833 Vermont Drive	Richmond Hill	Ontario	Basic	179-490-8666	gtitcomb4@dyndns.org
12662	Roy	west	Male	1982-03-03	2 Tomscot Court	Brampton	Ontario	Basic	669-932-9006	clindegard5@ezinearticles.com
12748	Frances	thompson	Female	1982-12-02	96 Oriole Center	Greater Sudbury	Ontario	Platinum	735-377-4981	fgascard6@bbb.org
12791	Charles	harrison	Male	1980-08-18	0 Debs Avenue	Cornwall	Ontario	Basic	599-268-1866	aprandi7@feedburner.com
12838	Frank	ryan	Male	1980-08-27	3 Sullivan Lane	Cornwall	Ontario	Platinum	743-857-4116	yfleisch8@php.net
12855	Louis	tran	Male	1987-01-25	45 Redwing Lane	Greater Sudbury	Ontario	Basic	688-249-9972	ldaybell9@google.es
12868	Thelma	ahmed	Female	1982-01-21	34 Nevada Terrace	Greater Sudbury	Ontario	Basic	564-243-4401	agidmana@webs.com

DEPARTMENT RELATIONS

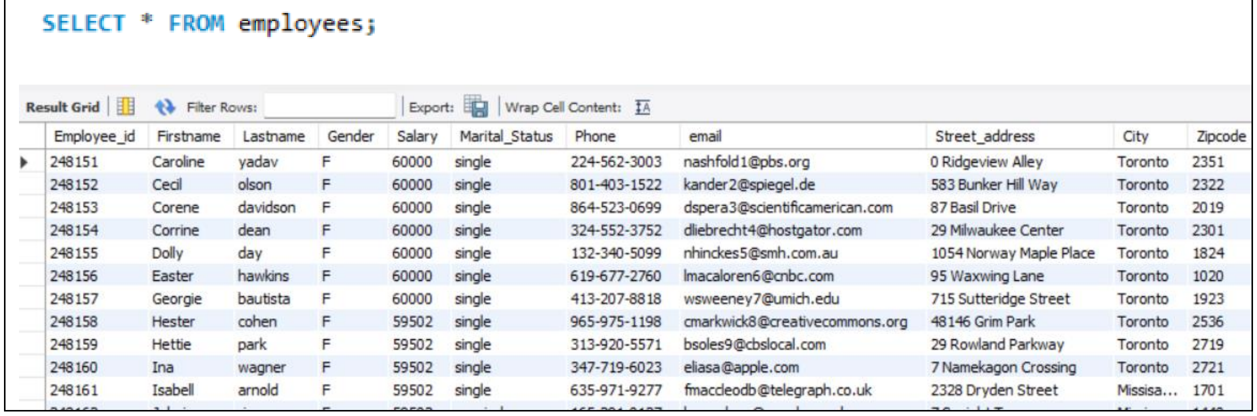


```
SELECT * FROM department;
```

Result Grid | Filter Rows:

dept_id	dept_name
251	Sales

EMPLOYEES RELATIONS

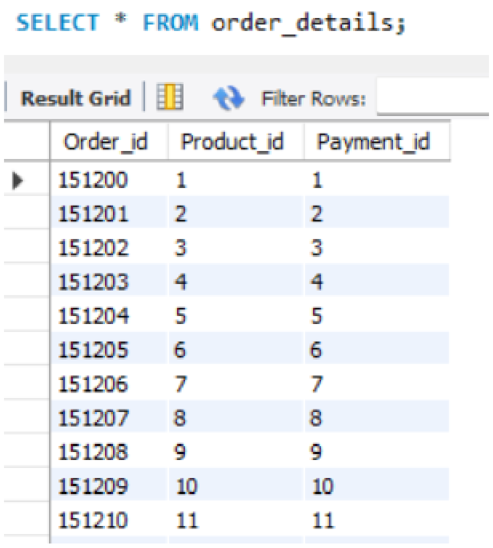


```sql
SELECT * FROM employees;
```

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

Employee_id	Firstname	Lastname	Gender	Salary	Marital_Status	Phone	email	Street_address	City	Zipcode
248151	Caroline	yadav	F	60000	single	224-562-3003	nashfold1@pbs.org	0 Ridgeview Alley	Toronto	2351
248152	Cecil	olson	F	60000	single	801-403-1522	kander2@spiegel.de	583 Bunker Hill Way	Toronto	2322
248153	Corene	davidson	F	60000	single	864-523-0699	dspera3@scientificamerican.com	87 Basil Drive	Toronto	2019
248154	Corrine	dean	F	60000	single	324-552-3752	dliebrecht4@hostgator.com	29 Milwaukee Center	Toronto	2301
248155	Dolly	day	F	60000	single	132-340-5099	nhinckes5@smh.com.au	1054 Norway Maple Place	Toronto	1824
248156	Easter	hawkins	F	60000	single	619-677-2760	lmacaloren6@cnbc.com	95 Waxwing Lane	Toronto	1020
248157	Georgie	bautista	F	60000	single	413-207-8818	wsweeney7@umich.edu	715 Sutteridge Street	Toronto	1923
248158	Hester	cohen	F	59502	single	965-975-1198	cmarkwick8@creativecommons.org	48146 Grim Park	Toronto	2536
248159	Hettie	park	F	59502	single	313-920-5571	bsoles9@cbslocal.com	29 Rowland Parkway	Toronto	2719
248160	Ina	wagner	F	59502	single	347-719-6023	eliasa@apple.com	7 Namekagon Crossing	Toronto	2721
248161	Isabell	arnold	F	59502	single	635-971-9277	fmaccleodb@telegraph.co.uk	2328 Dryden Street	Missisa...	1701

ORDER_DETAILS RELATIONS

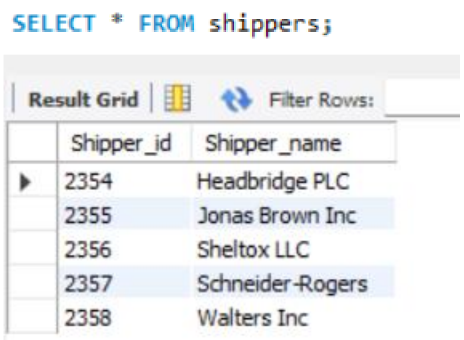


```sql
SELECT * FROM order_details;
```

Result Grid | Filter Rows:

Order_id	Product_id	Payment_id
151200	1	1
151201	2	2
151202	3	3
151203	4	4
151204	5	5
151205	6	6
151206	7	7
151207	8	8
151208	9	9
151209	10	10
151210	11	11

SHIPPERS RELATIONS

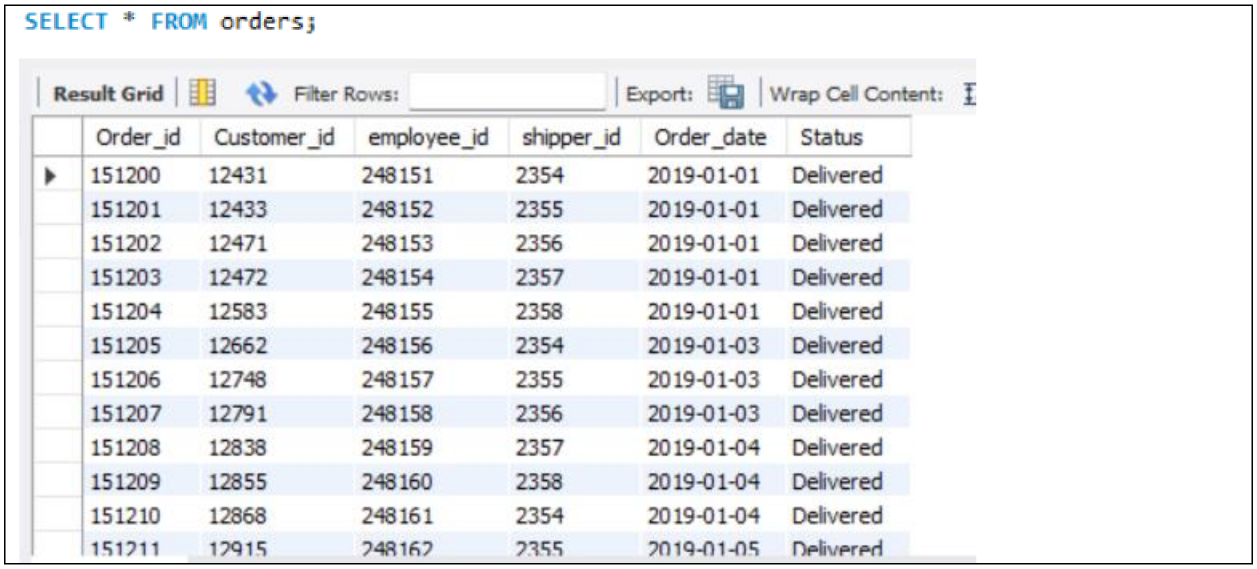


```sql
SELECT * FROM shippers;
```

Result Grid | Filter Rows:

Shipper_id	Shipper_name
2354	Headbridge PLC
2355	Jonas Brown Inc
2356	Sheltox LLC
2357	Schneider-Rogers
2358	Walters Inc

ORDERS RELATIONS

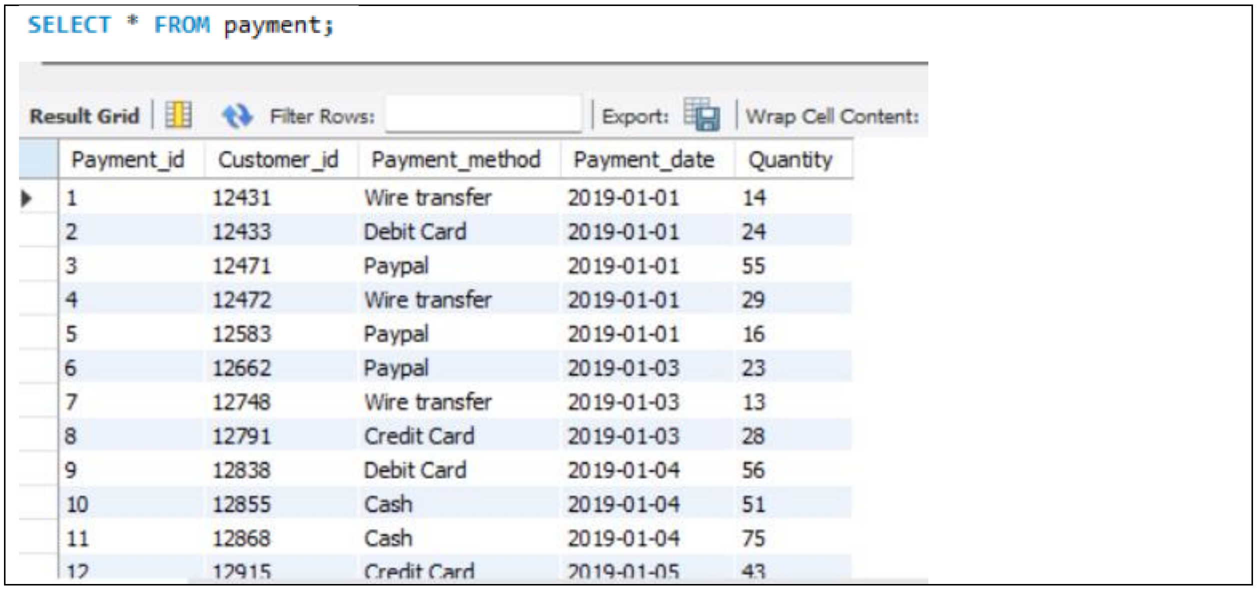


```
SELECT * FROM orders;
```

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

Order_id	Customer_id	employee_id	shipper_id	Order_date	Status
151200	12431	248151	2354	2019-01-01	Delivered
151201	12433	248152	2355	2019-01-01	Delivered
151202	12471	248153	2356	2019-01-01	Delivered
151203	12472	248154	2357	2019-01-01	Delivered
151204	12583	248155	2358	2019-01-01	Delivered
151205	12662	248156	2354	2019-01-03	Delivered
151206	12748	248157	2355	2019-01-03	Delivered
151207	12791	248158	2356	2019-01-03	Delivered
151208	12838	248159	2357	2019-01-04	Delivered
151209	12855	248160	2358	2019-01-04	Delivered
151210	12868	248161	2354	2019-01-04	Delivered
151211	12915	248162	2355	2019-01-05	Delivered

PAYMENT RELATIONS

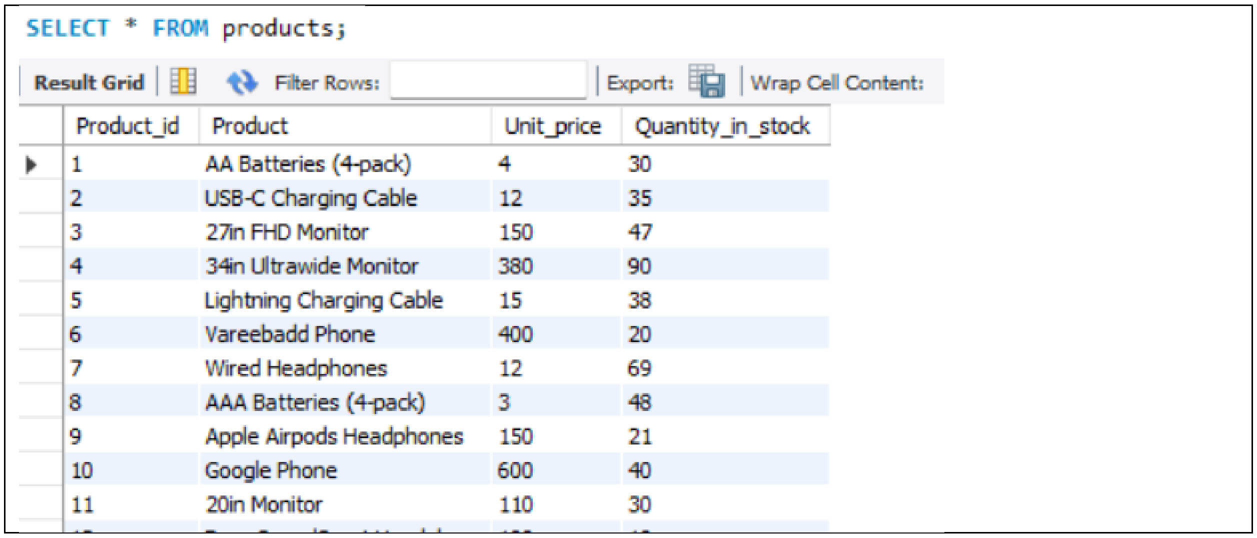


```
SELECT * FROM payment;
```

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

Payment_id	Customer_id	Payment_method	Payment_date	Quantity
1	12431	Wire transfer	2019-01-01	14
2	12433	Debit Card	2019-01-01	24
3	12471	Paypal	2019-01-01	55
4	12472	Wire transfer	2019-01-01	29
5	12583	Paypal	2019-01-01	16
6	12662	Paypal	2019-01-03	23
7	12748	Wire transfer	2019-01-03	13
8	12791	Credit Card	2019-01-03	28
9	12838	Debit Card	2019-01-04	56
10	12855	Cash	2019-01-04	51
11	12868	Cash	2019-01-04	75
12	12915	Credit Card	2019-01-05	43

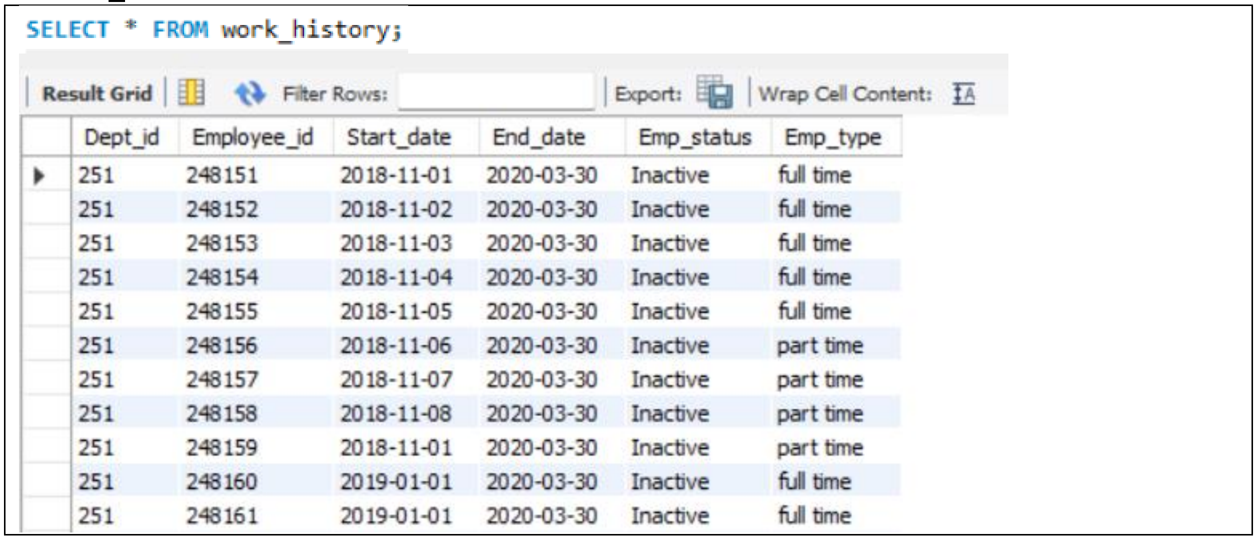
PRODUCT RELATIONS



```
SELECT * FROM products;
```

Product_id	Product	Unit_price	Quantity_in_stock
1	AA Batteries (4-pack)	4	30
2	USB-C Charging Cable	12	35
3	27in FHD Monitor	150	47
4	34in Ultrawide Monitor	380	90
5	Lightning Charging Cable	15	38
6	Vareebadd Phone	400	20
7	Wired Headphones	12	69
8	AAA Batteries (4-pack)	3	48
9	Apple Airpods Headphones	150	21
10	Google Phone	600	40
11	20in Monitor	110	30

WORK_HISTORY RELATIOSN



```
SELECT * FROM work_history;
```

Dept_id	Employee_id	Start_date	End_date	Emp_status	Emp_type
251	248151	2018-11-01	2020-03-30	Inactive	full time
251	248152	2018-11-02	2020-03-30	Inactive	full time
251	248153	2018-11-03	2020-03-30	Inactive	full time
251	248154	2018-11-04	2020-03-30	Inactive	full time
251	248155	2018-11-05	2020-03-30	Inactive	full time
251	248156	2018-11-06	2020-03-30	Inactive	part time
251	248157	2018-11-07	2020-03-30	Inactive	part time
251	248158	2018-11-08	2020-03-30	Inactive	part time
251	248159	2018-11-01	2020-03-30	Inactive	part time
251	248160	2019-01-01	2020-03-30	Inactive	full time
251	248161	2019-01-01	2020-03-30	Inactive	full time

USE CASE

Use Case 1	
Use Case Name	Add new employee information
Use Case Description	User adds new employee information to the database
Actor/User	Human Resource Manager
Pre-condition	Must be connected to the server
Post-Condition	The new employee information is returned when database is queried
Steps: 1. User goes to 'employee' table. 2. User clicks 'create new employee' 3. Database generates an employee ID 4. User enters the following required fields 'First Name, Last Name, Gender, Date of Birth, Salary, Marital Status, Phone, Email, Zipcode' in the Database. 5. User clicks 'confirm' to save the information 6. New staff information is returned when database is queried.	

Use Case 2	
Use Case Name	Update employee information
Use Case Description	User updates the employee information in the database
Actor/User	Human Resource Manager
Pre-condition	Must be connected to the server
Post-condition	Updated employee information is returned when database is queried
Steps: 1. User goes to 'employee' table 2. User enters the employee ID 3. User chooses the option to edit the following employee information 'First Name, Last Name, Gender, Date of Birth, Salary, Marital Status, Phone, Email, Zipcode' in the Database 4. System displays all inputted information and requests confirmation 5. User clicks on 'confirm' 6. Updated employee information is saved in database	

USE CASE 3	
Use Case Name	Delete employee information
Use Case Description	User deletes employee information in the database
Actor/User	Human Resource Manager
Pre-Condition	Must be connected to the server
Post-Condition	Deleted employee information is not returned when database is queried

Steps:

1. User goes to employee table
2. User enters the employee ID
3. User selects the employee profile
4. User clicks ‘delete’
5. Database asks for confirmation
6. User clicks ‘confirm’
7. Selected employee information is deleted from the database

USE CASE 4

Use Case Name	Add new customer information
Use Case Description	User adds new customer information to the database
Actor/User	Sales Manager
Pre-Condition	Must be connected to the server
Post-Condition	The new employee information is returned when database is queried

Steps:

1. User goes to ‘customers’ table.
2. User clicks ‘create new customer’
3. Database generates a customer ID
4. User enters the following required fields ‘First Name, Last Name, Gender, Date of Birth, Membership, Phone, Email, Zip code’ in the Database.
5. User clicks ‘confirm’ to save the information
6. New staff information is returned when database is queried.

USE CASE 5

Use Case Name	Update customer information
Use Case Description	User updates customer information in the database
Actor/User	Sales Manager
Pre-Condition	Must be connected to the server
Post-Condition	Updated customer information is returned when database is queried

Steps:

1. User goes to ‘customers’ table
2. User enters the customer ID
3. User chooses the option to edit the following employee information ‘First Name, Last Name, Gender, Date of Birth, Membership, Phone, Email, Zip code’ in the Database
4. System displays all inputted information and requests confirmation
5. User clicks on ‘confirm’.
6. Updated customer information is saved in database

USE CASE 5	
Use Case Name	Delete customer information
Use Case Description	User deletes customer information from the database
Actor/User	Sales Manager
Pre-Condition	Must be connected to server
Post-Condition	Selected customer is deleted from the database

Steps:

1. User goes to 'customers' table
2. User enters the customer's ID
3. User selects the customer's profile
4. User clicks 'delete'
5. Database asks for confirmation
6. User clicks 'confirm'
7. Selected customer information is deleted from the database

USE CASE 7	
Use Case Name	Add new department information
Use Case Description	User enters new department information into the database
Actor/User	Operations Manager
Pre-Condition	Must be connected to the server
Post- Condition	New department information is returned when database is queried

Steps:

1. User goes to 'department' table.
2. User clicks 'create new department'
3. Database generates a department ID
4. User enters the following required fields 'Department Name' in the Database.
5. User clicks 'confirm' to save the information
6. New department information is returned when database is queried.

USE CASE 8	
Use Case Name	Update department information
Use Case Description	User updates department information in the database
Actor/User	Operations Manager
Pre-Condition	Must be connected to the server
Post-Condition	Updated department is returned when database is queried

Steps:

1. User goes to 'department' table
2. User enters the department ID or department name
3. User chooses the option to edit the following employee information 'Department Name' in the Database
4. System displays all inputted information and requests confirmation

5. User clicks on ‘confirm’.
6. Updated department information is saved in database

USE CASE 9

Use Case Name	Delete department information
Use Case Description	User deletes department information from the database
Actor/User	Operations Manager
Pre-Condition	Must be connected to the server
Post-Condition	Selected department is deleted from the database

1. **Steps:**
2. User goes to ‘department’ table
3. User enters the department’s ID or department name
4. User selects the department profile
5. User clicks ‘delete’
6. Database asks for confirmation
7. User clicks ‘confirm’
8. Selected department information is deleted from the database

USE CASE 10

Use Case Name	Enter new order information
Use Case Description	User enters new order information into the database
Actor/User	Sales Manager
Pre-Condition	Must be connected to the server
Post-Condition	New order information is returned when database is queried

Steps:

1. User goes to ‘order’ table.
2. User clicks ‘create new order’
3. Database generates an order ID
4. User enters the following required fields ‘Customer ID, Employee ID, Shipper ID, Shipped Date, Order Date, Status’ in the Database.
5. User clicks ‘confirm’ to save the information
6. New order information is returned when the database is queried.

USE CASE 11:

Use Case Name	Update order information
Use Case Description	User updates new order information in the database
Actor/User	Sales Manager
Pre-Condition	Must be connected to the server
Post-Condition	New order information is returned when database is queried

Steps:

1. User goes to ‘orders’ table

2. User enters the order ID
3. User chooses the option to edit the following employee information 'Customer ID, Employee ID, Shipper ID, Shipped Date, Order Date, Status' in the Database
4. System displays all inputted information and requests confirmation
5. User clicks on 'confirm'.
6. Updated department information is saved in database

USE CASE 12

Use Case Name	Delete order information
Use Case Description	User deletes order information from the database
Actor/User	Sales Manager
Pre-Condition	Must be connected to the server
Post-Condition	Selected order information is deleted from database

Steps:

1. User goes to 'orders' table
2. User enters the order ID
3. User selects the order profile
4. User clicks 'delete'
5. Database asks for confirmation
6. User clicks 'confirm'
7. Selected order information is deleted from the database

USE CASE 13

Use Case Name	Enter new payment information
Use Case Description	User enters new payment information
Actor/User	Accounts Manager
Pre-Condition	Must be connected to the server
Post-Condition	New order details information is returned when database is queried

Steps:

1. User goes to 'payment' table.
2. User clicks 'create new payment'
3. Database generates a payment ID
4. User enters the following required fields 'Customer ID, Payment method, Amount, Payment Date, Quantity' in the Database.
5. User clicks 'confirm' to save the information
6. New order details information is returned when the database is queried.

USE CASE 14

Use Case Name	Update payment information
Use Case Description	User updates payment information in the database
Actor/User	Accounts Manager
Pre-Condition	Must be connected to the server

Post-Condition	New payment information is returned when database is queried

Steps:

1. User goes to ‘payment’ table
2. User enters the payment ID
3. User chooses the option to edit the following employee information ‘Customer ID, Payment method, Amount, Payment Date, Quantity’ in the Database
4. System displays all inputted information and requests confirmation
5. User clicks on ‘confirm’.
6. Updated payment information is saved in database

USE CASE 15

Use Case Name	Delete Payment information
Use Case Description	User deletes payment information from the database
Actor/User	Accounts Manager
Pre-Condition	Must be connected to the server
Post-Condition	Selected payment is deleted from the database

Steps:

1. User goes to ‘payment’ table
2. User enters the payment ID
3. User selects the order profile
4. User clicks ‘delete’
5. Database asks for confirmation
6. User clicks ‘confirm’
7. Selected payment information is deleted from the database

USE CASE 16

Use Case Name	Enter new shipper information
Use Case Description	User Enters new shipper information
Actor/User	Store Manager
Pre-Condition	Must be connected to server
Post-Condition	New order information is returned when database is queried

Steps:

1. User goes to ‘shipper’ table.
2. User clicks ‘create new shipper’
3. Database generates a shipper ID
4. User enters the following required fields ‘Shipper Name’ in the Database.
5. User clicks ‘confirm’ to save the information
6. New shipper information is returned when the database is queried.

USE CASE 17

Use Case Name	Update shipper information
Use Case Description	User updates shipper information

Actor/User	Store Manager
Pre-Condition	Must be connected to server
Post-Condition	Updated shipper information is returned when database is queried

Steps:

1. User goes to 'shipper' table
2. User enters the shipper ID
3. User chooses the option to edit the following employee information 'shipper name' in the Database
4. System displays all inputted information and requests confirmation
5. User clicks on 'confirm'.
6. Updated shipper information is saved in database

USE CASE 18

Use Case Name	Delete shipper information
Use Case Description	User Deletes shipper information
Actor/User	Store Manager
Pre-Condition	Must be connected to server
Post-Condition	Selected shipper information is deleted from the database

Steps:

1. User goes to 'shipper' table
2. User enters the shipper ID
3. User selects the order profile
4. User clicks 'delete'
5. Database asks for confirmation
6. User clicks 'confirm'
7. Selected shipper information is deleted from the database

USE CASE 19

Use Case Name	Enter Product information
Use Case Description	User enters product information
Actor/User	Store Manager
Pre-Condition	Must be connected to server
Post-Condition	New product information is returned when database is queried

Steps:

1. User goes to 'product' table.
2. User clicks 'create new product'
3. Database generates a product ID
4. User enters the following required fields 'product, unit price, quantity in stock' in the Database.
5. User clicks 'confirm' to save the information
6. New product information is returned when the database is queried.

USE CASE 20:

Use Case Name	Update product information
Use Case Description	User updates product information in the database
Actor/User	Store Manager
Pre-Condition	Must be connected to the server
Post-Condition	Updated product information is returned when database is queried

Steps:

1. User goes to ‘product’ table
2. User enters the product ID
3. User chooses the option to edit the following employee information ‘product, unit price, quantity in stock’ in the Database
4. System displays all inputted information and requests confirmation
5. User clicks on ‘confirm’.
6. Updated product information is saved in database

USE CASE 21

Use Case Name	Delete product information
Use Case Description	User deleted product information from the database
Actor/User	Store Manager
Pre-Condition	Must be connected to the server
Post-Condition	Selected product information is deleted from the database

Steps:

1. User goes to ‘product’ table
2. User enters the product ID
3. User selects the product profile
4. User clicks ‘delete’
5. Database asks for confirmation
6. User clicks ‘confirm’
7. Selected product information is deleted from the database

USE CASE 22

Use Case Name	Generate a report for the HR Manager which shows the top 2 employees with the highest sales volume and the lowest sales volume for the year 2020 and 2021.
Use Case Description	The report is generated using Join functions between the employees table and the orders table, and Group By the Employee
Actor/User	Data Analyst
Pre-Condition	All required tables must be available in the database
Post-Condition	The report shows the top 2 employees with the highest sales for the year 2020 and 2021 combined.

Steps: 1. Join conditions are entered for employee table and orders table 2. User enters the filtering conditions 3. User selects relevant attributes 4. User enters group by employee 5. The report showing the top 2 employees with the highest sales volume for 2020 and 2021 combined are generated.	

USE CASE 23

Use Case Name	Generate a monthly sales report for 2020 for the Sales Manager which shows the sales volume (number of sales made) and Gross Sales (total sales in dollars)
Use Case Description	Join functions are used to generate the sales volume and the gross sales
Actor/User	Data Analyst
Pre-Condition	All required tables must be available in the database
Post-Condition	The report showing the sales volume and gross sales is created.

Steps:
1. Join conditions are entered for orders table, payment table, order details table and product table.
2. User enters the filtering conditions
3. User selects relevant attributes
4. User enters group by order date
5. The report showing the sales volume and gross sales is generated.

USE CASE 24

Use Case Name	Create a report for the Sales Manager that shows the top 3 highest selling products in 2020.
Use Case Description	Join functions are used to generate the report
Actor/User	Data Analyst
Pre-Condition	All required tables must be available in the database
Post-Condition	The report showing the top 3 highest selling products in 2020 is generated

Steps:
1. Join conditions are entered for orders table, payment table, order details table and product table.
2. User enters the filtering conditions
3. User selects relevant attributes
4. User enters group by product
5. The report showing the sales volume and gross sales is generated.

USE CASE 25

Use Case Name	Create a report for the Marketing Manager that shows the most profitable customers in 2020.
Use Case Description	Join functions are used to generate the report

Actor/User	Data Analyst
Pre-Condition	All required tables must be available in the database
Post-Condition	The report showing the most profitable customers in 2020 is generated

Steps:

1. Join conditions are entered for orders table, product, payment table, order details table and customers table.
2. User enters the filtering conditions
3. User selects relevant attributes
4. User enters group by customer
5. The report showing the sales volume and gross sales is generated.

USE CASE 26

Use Case Name	Generate a report for the Sales Manager that shows the most frequent means of payment by customers.
Use Case Description	Generate the report using the payment table
Actor/User	Data Analyst
Pre-Condition	All required tables must be available in the database
Post-Condition	The report showing the most frequent means of payment by customers is generated.

Steps:

1. User queries the payment table
2. User enters the filtering conditions
3. User selects relevant attributes
4. User enters group by product
5. The report showing the sales volume and gross sales is generated.

USE CASE 27

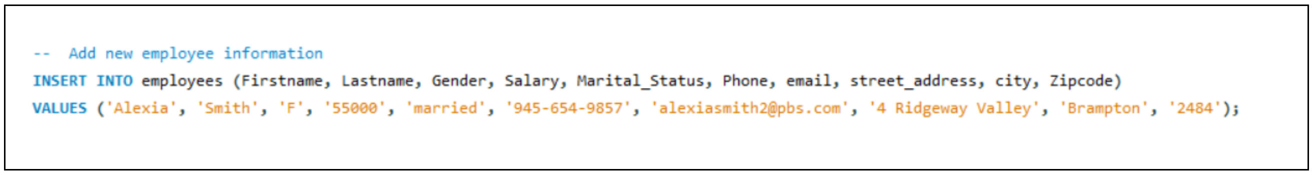
Use Case Name	Generate a report for the marketing manager that shows the most common age demographic of customers
Use Case Description	Generate the report using the customer table
Actor/User	Data Analyst
Pre-Condition	All required tables must be available
Post-Condition	The report showing the most common age demographic of customers is generated

Steps:

1. User queries the customers table
2. User enters the filtering conditions
3. User selects relevant attributes
4. User enters group by date of birth
5. The report showing the sales volume and gross sales is generated.

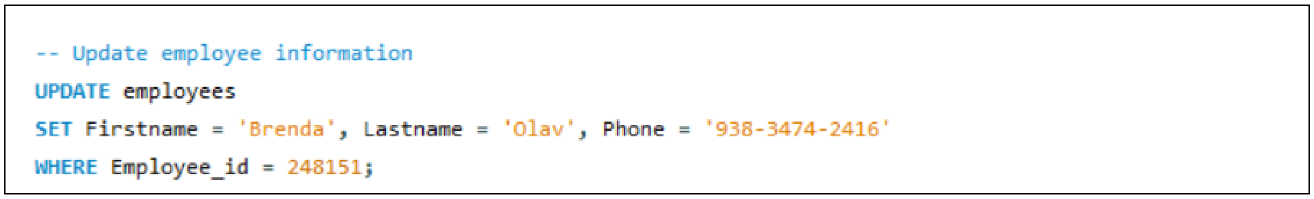
USE CASE REALIZATION- SQL STATEMENTS

USE CASE 1



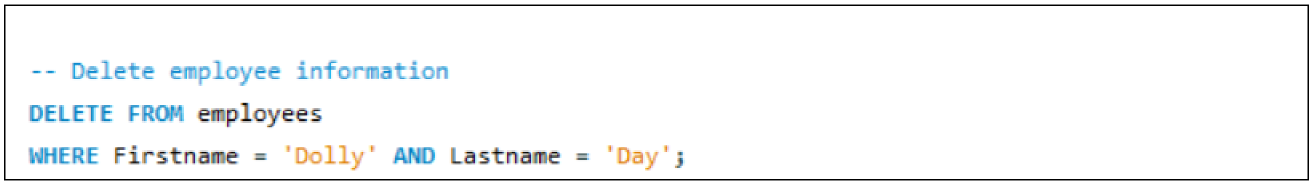
```sql
--  Add new employee information
INSERT INTO employees (Firstname, Lastname, Gender, Salary, Marital_Status, Phone, email, street_address, city, Zipcode)
VALUES ('Alexia', 'Smith', 'F', '55000', 'married', '945-654-9857', 'alexiasmith2@pbs.com', '4 Ridgeway Valley', 'Brampton', '2484');
```

USE CASE 2



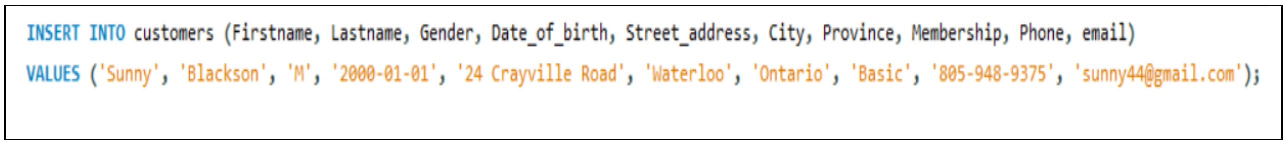
```sql
-- Update employee information
UPDATE employees
SET Firstname = 'Brenda', Lastname = 'Olav', Phone = '938-3474-2416'
WHERE Employee_id = 248151;
```

USE CASE 3



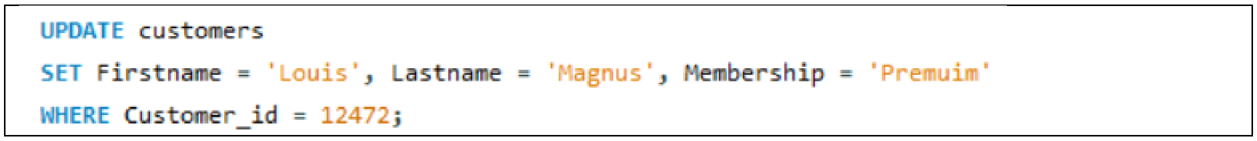
```sql
-- Delete employee information
DELETE FROM employees
WHERE Firstname = 'Dolly' AND Lastname = 'Day';
```

USE CASE 4



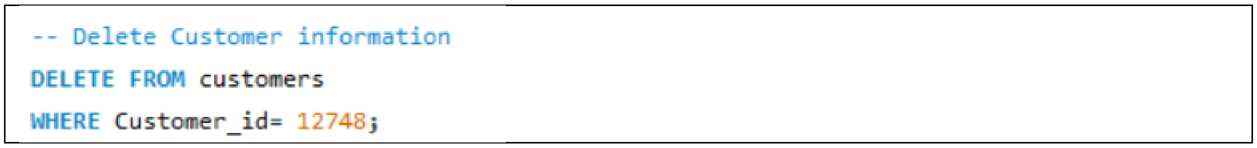
```sql
INSERT INTO customers (Firstname, Lastname, Gender, Date_of_birth, Street_address, City, Province, Membership, Phone, email)
VALUES ('Sunny', 'Blackson', 'M', '2000-01-01', '24 Crayville Road', 'Waterloo', 'Ontario', 'Basic', '805-948-9375', 'sunny44@gmail.com');
```

USE CASE 5



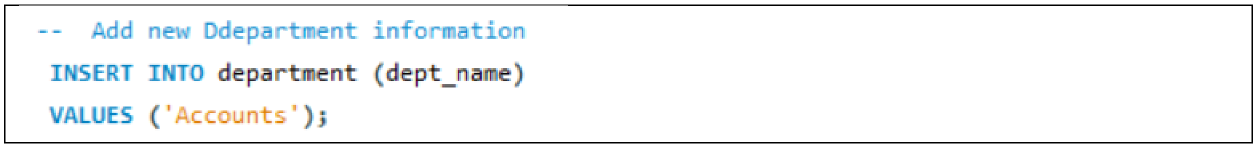
```sql
UPDATE customers
SET Firstname = 'Louis', Lastname = 'Magnus', Membership = 'Premuim'
WHERE Customer_id = 12472;
```

USE CASE 6



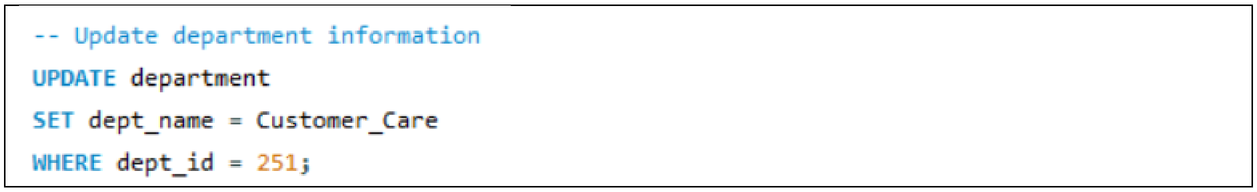
```sql
-- Delete Customer information
DELETE FROM customers
WHERE Customer_id= 12748;
```

USE CASE 7



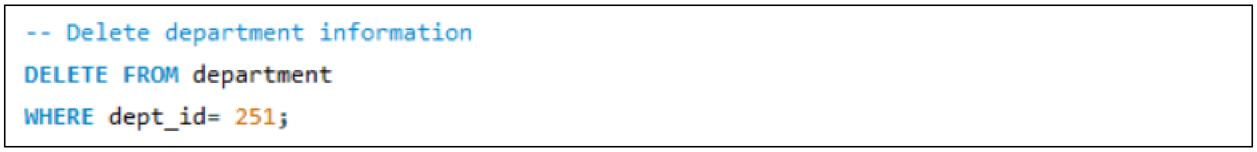
```sql
--  Add new Ddepartment information
 INSERT INTO department (dept_name)
 VALUES ('Accounts');
```

USE CASE 8



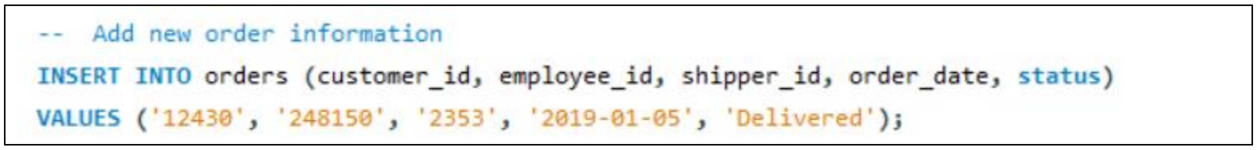
```sql
-- Update department information
UPDATE department
SET dept_name = Customer_Care
WHERE dept_id = 251;
```

USE CASE 9



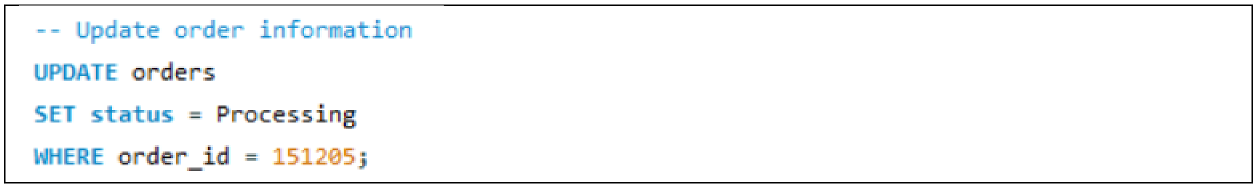
```sql
-- Delete department information
DELETE FROM department
WHERE dept_id= 251;
```

USE CASE 10



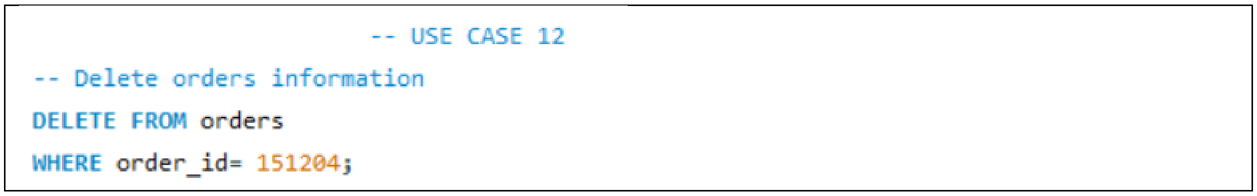
```sql
--  Add new order information
INSERT INTO orders (customer_id, employee_id, shipper_id, order_date, status)
VALUES ('12430', '248150', '2353', '2019-01-05', 'Delivered');
```

USE CASE 11



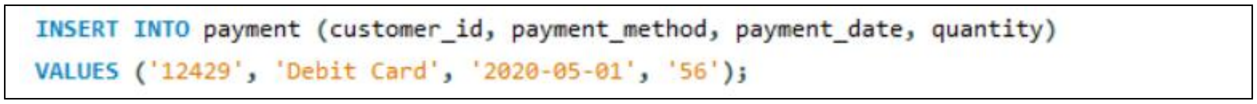
```sql
-- Update order information
UPDATE orders
SET status = Processing
WHERE order_id = 151205;
```

USE CASE 12



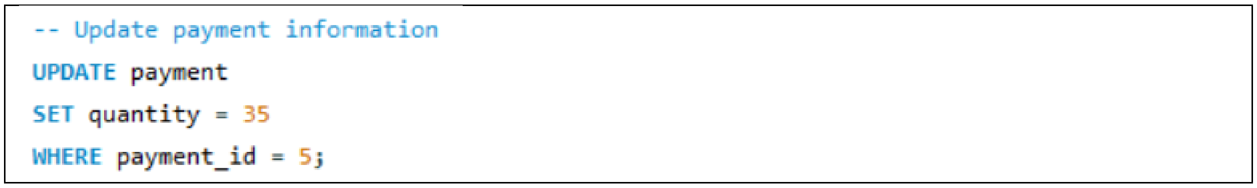
```
                         -- USE CASE 12
-- Delete orders information
DELETE FROM orders
WHERE order_id= 151204;
```

USE CASE 13



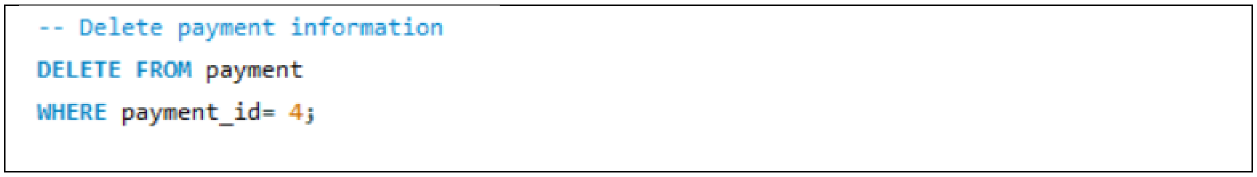
```
INSERT INTO payment (customer_id, payment_method, payment_date, quantity)
VALUES ('12429', 'Debit Card', '2020-05-01', '56');
```

USE CASE 14



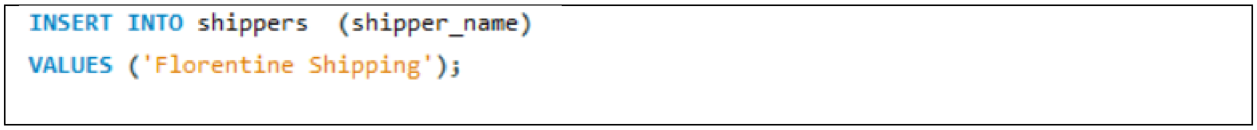
```
-- Update payment information
UPDATE payment
SET quantity = 35
WHERE payment_id = 5;
```

USE CASE 15



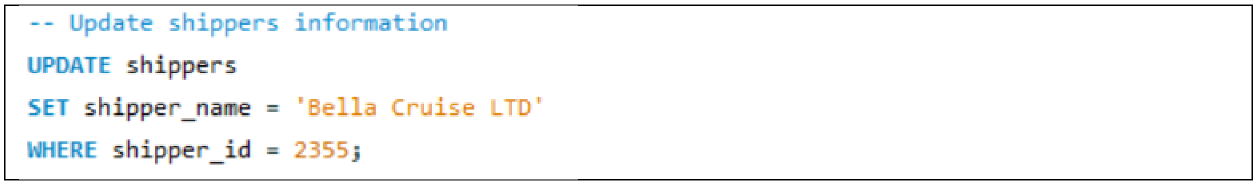
```
-- Delete payment information
DELETE FROM payment
WHERE payment_id= 4;
```

USE CASE 16



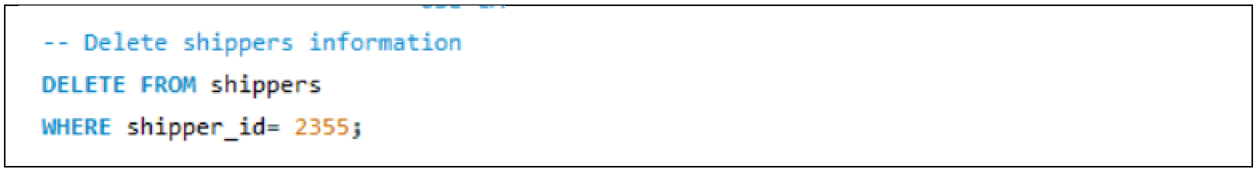
```
INSERT INTO shippers  (shipper_name)
VALUES ('Florentine Shipping');
```

USE CASE 17



```
-- Update shippers information
UPDATE shippers
SET shipper_name = 'Bella Cruise LTD'
WHERE shipper_id = 2355;
```

USE CASE 18



```
-- Delete shippers information
DELETE FROM shippers
WHERE shipper_id= 2355;
```

USE CASE 19

```
--  Add new product information
INSERT INTO products (Product, Unit_price, Quantity_in_stock)
VALUES ('Samsung Dishwasher', '463', '40');
```

USE CASE 20

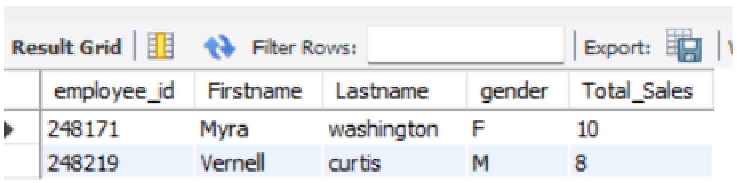
```
UPDATE products
SET Product = 'Samsung Dryer'
WHERE product_id = 17;
```

USE CASE 21

```
-- Delete products information
DELETE FROM products
WHERE product_id= 19;
```

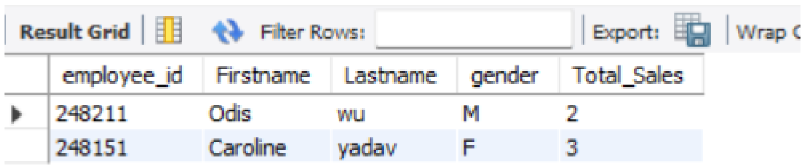
USE CASE 22

```
-- Generate a report for the HR Manager which shows the top 2 employees with the highest
-- sales volume for the year 2020 and 2021. To get the names of the top 2 employees with
-- the highest sales, use Join to combine the employees table and the orders table (this is to get the name of the
-- employees as well as the orders)
-- Also, use GROUP BY, COUNT, ORDER BY, and LIMIT

SELECT o.employee_id, e.Firstname, e.Lastname, e.gender, COUNT(*) AS 'Total_Sales'
FROM orders o
JOIN employees e
ON o.employee_id = e.employee_id
WHERE o.order_date BETWEEN '2020-01-01' AND '2021-12-31'
GROUP BY o.employee_id
ORDER BY Total_sales DESC
LIMIT 2;
```



employee_id	Firstname	Lastname	gender	Total_Sales
248171	Myra	washington	F	10
248219	Vernell	curtis	M	8

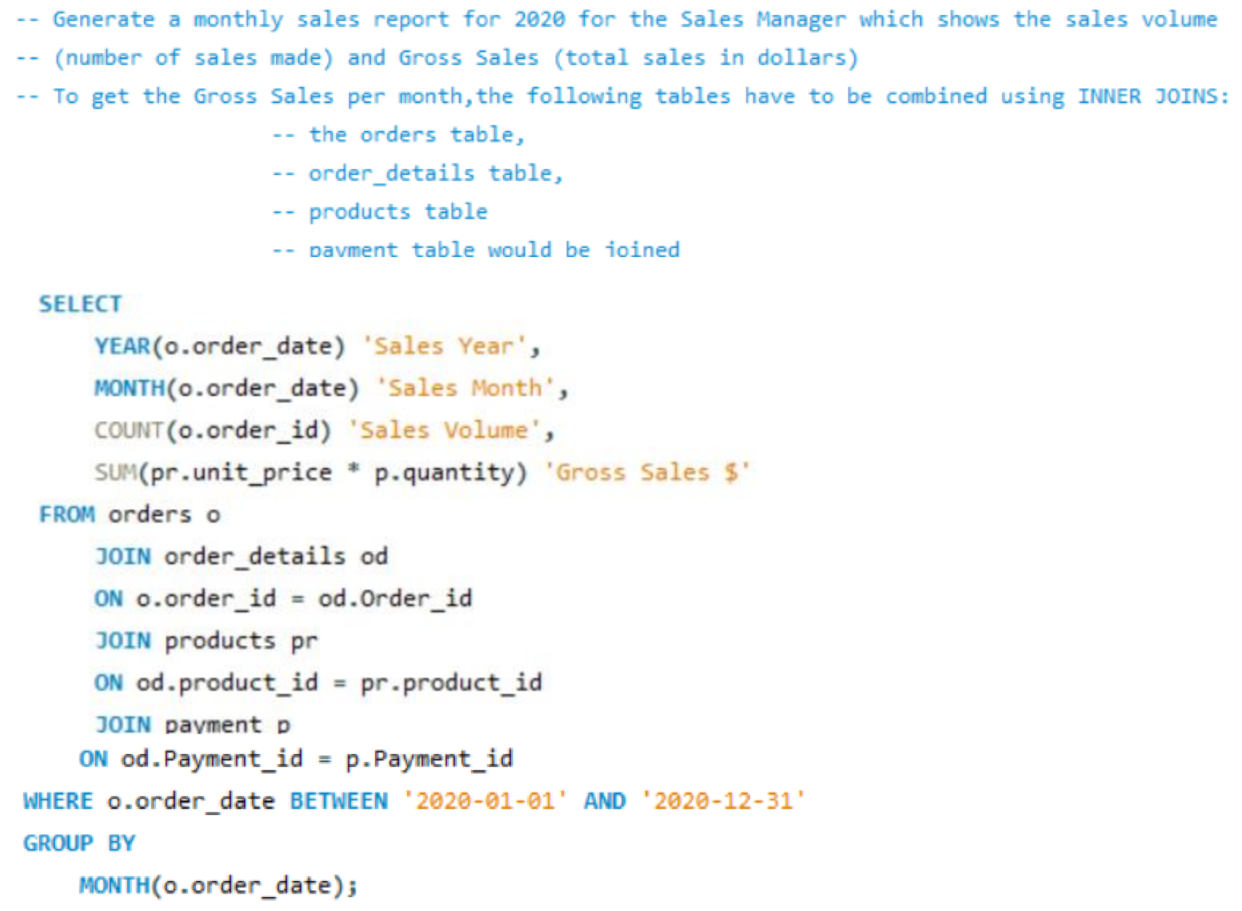
```
-- To get the 2 lowest performing employees, run the same query but this time Order by ascending as shown below:

SELECT o.employee_id, e.Firstname, e.Lastname, e.gender, COUNT(*) AS 'Total_Sales'
FROM orders o
JOIN employees e
ON o.employee_id = e.employee_id
WHERE o.order_date BETWEEN '2020-01-01' AND '2021-12-31'
GROUP BY o.employee_id
ORDER BY Total_sales
```



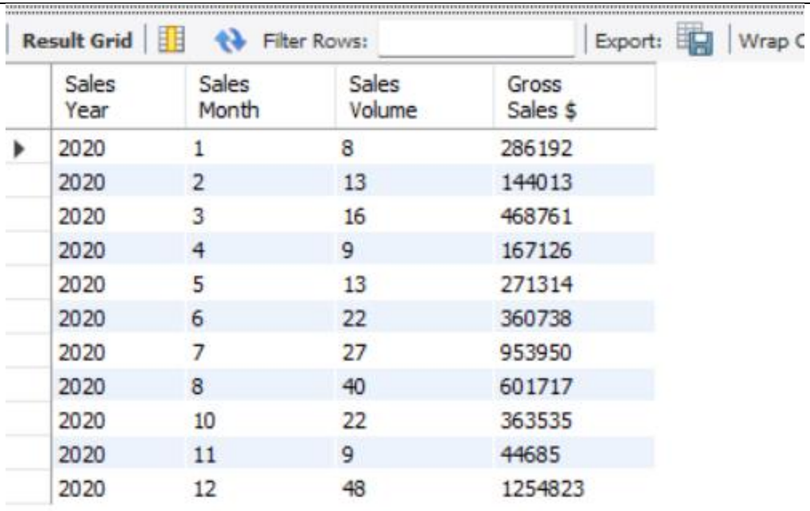
Result Grid | Filter Rows: | Export: | Wrap C

	employee_id	Firstname	Lastname	gender	Total_Sales
▶	248211	Odis	wu	M	2
	248151	Caroline	yadav	F	3

USE CASE 23



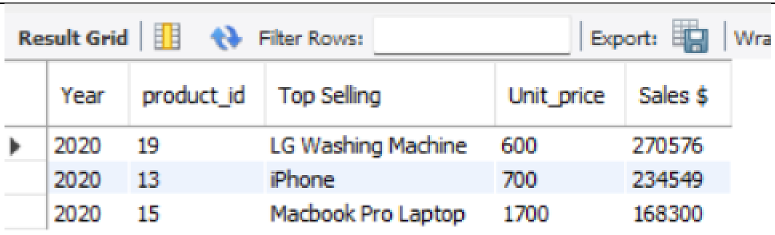
```
-- Generate a monthly sales report for 2020 for the Sales Manager which shows the sales volume
-- (number of sales made) and Gross Sales (total sales in dollars)
-- To get the Gross Sales per month,the following tables have to be combined using INNER JOINS:
                   -- the orders table,
                   -- order_details table,
                   -- products table
                   -- payment table would be joined

 SELECT
     YEAR(o.order_date) 'Sales Year',
     MONTH(o.order_date) 'Sales Month',
     COUNT(o.order_id) 'Sales Volume',
     SUM(pr.unit_price * p.quantity) 'Gross Sales $'
 FROM orders o
     JOIN order_details od
     ON o.order_id = od.Order_id
     JOIN products pr
     ON od.product_id = pr.product_id
     JOIN payment p
    ON od.Payment_id = p.Payment_id
WHERE o.order_date BETWEEN '2020-01-01' AND '2020-12-31'
GROUP BY
    MONTH(o.order_date);
```



Sales Year	Sales Month	Sales Volume	Gross Sales $
2020	1	8	286192
2020	2	13	144013
2020	3	16	468761
2020	4	9	167126
2020	5	13	271314
2020	6	22	360738
2020	7	27	953950
2020	8	40	601717
2020	10	22	363535
2020	11	9	44685
2020	12	48	1254823

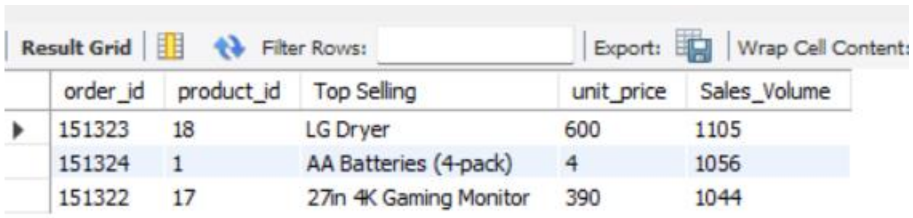
USE CASE 24

```
-- Create a report for the Sales Manager that shows the top 3 highest selling products in 2020.
-- To create this report, the following tables would be joined using an INNER JOIN:
        -- orders table
        -- order_details table
        -- products table
 SELECT
     YEAR(o.order_date) 'Year',
     pr.product_id,
     pr.product AS 'Top Selling',
     pr.Unit_price,
     SUM(pr.unit_price * p.quantity) 'Sales $'
 FROM orders o
     JOIN order_details od
     ON o.order_id = od.Order_id
     JOIN products pr
     ON od.product_id = pr.product_id
     JOIN payment p
     ON od.Payment_id = p.Payment_id
    JOIN customers c
    ON p.customer_id = c.customer_id
WHERE o.order_date BETWEEN '2020-01-01' AND '2020-12-31'
GROUP BY
    c.customer_id
ORDER BY SUM(pr.unit_price * p.quantity) DESC
LIMIT 3 ;
```



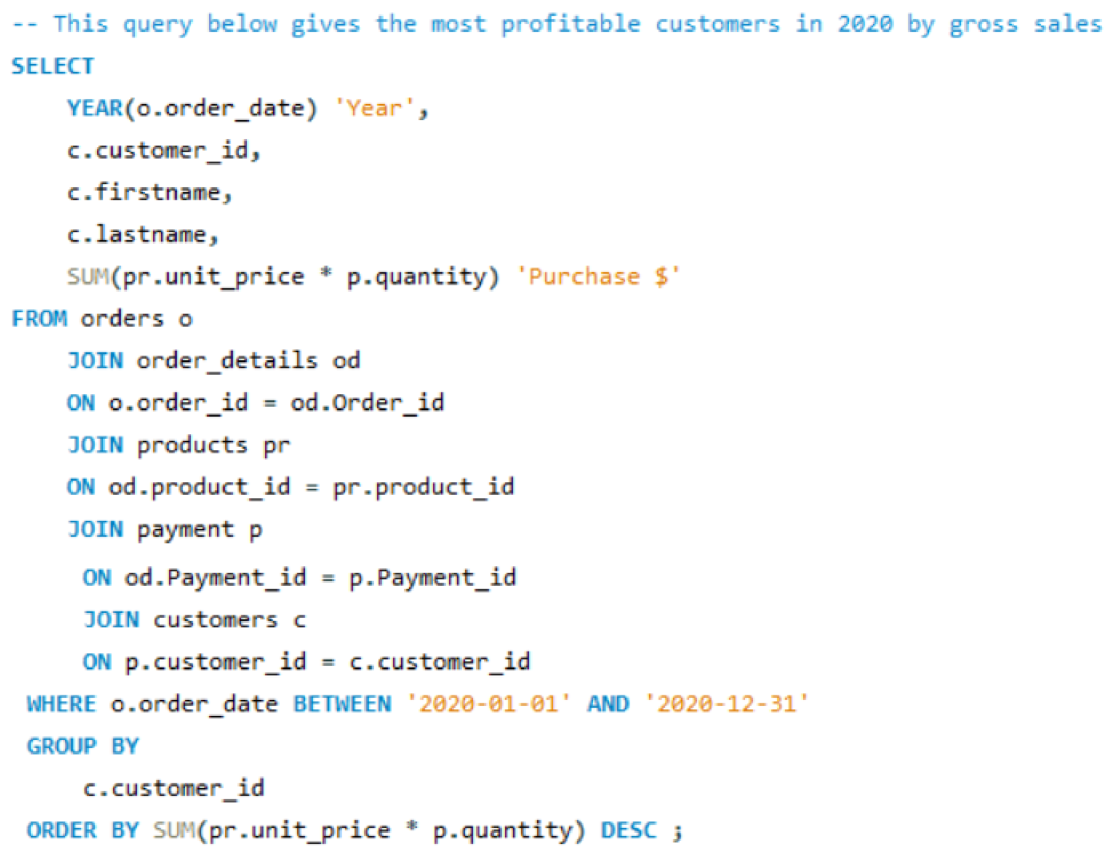
Year	product_id	Top Selling	Unit_price	Sales $
2020	19	LG Washing Machine	600	270576
2020	13	iPhone	700	234549
2020	15	Macbook Pro Laptop	1700	168300

```
-- This query below gives the top 3 highest selling product by sales volume.
SELECT
    o.order_id,
    pr.product_id,
    pr.product AS 'Top Selling',
    pr.unit_price,
    SUM(p.quantity) AS Sales_Volume
FROM orders o
    JOIN order_details od
    ON o.Order_id = od.Order_id
    JOIN products pr
    ON od.Product_id = pr.Product_id
WHERE o.order_date BETWEEN '2020-01-01' AND '2020-12-31'
GROUP BY pr.Product
ORDER BY SUM(p.quantity) DESC
LIMIT 3;
-- The above query returns LG Dryer, Iphone and AA Batteries as the Top 3 Selling Products in 2020.
```

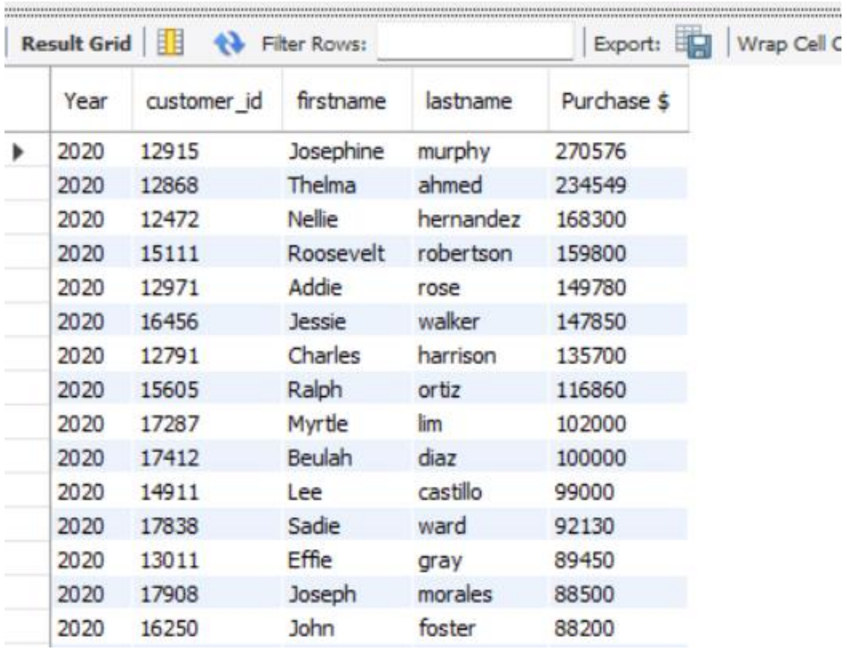


order_id	product_id	Top Selling	unit_price	Sales_Volume
151323	18	LG Dryer	600	1105
151324	1	AA Batteries (4-pack)	4	1056
151322	17	27in 4K Gaming Monitor	390	1044

USE CASE 25

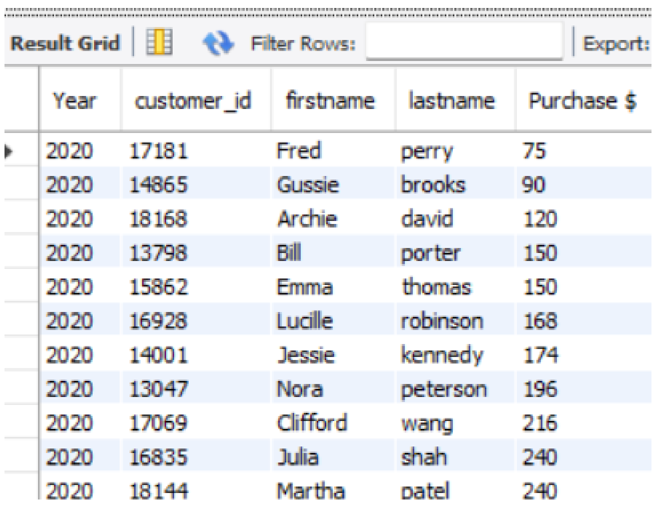


```
-- This query below gives the most profitable customers in 2020 by gross sales
SELECT
    YEAR(o.order_date) 'Year',
    c.customer_id,
    c.firstname,
    c.lastname,
    SUM(pr.unit_price * p.quantity) 'Purchase $'
FROM orders o
    JOIN order_details od
    ON o.order_id = od.Order_id
    JOIN products pr
    ON od.product_id = pr.product_id
    JOIN payment p
     ON od.Payment_id = p.Payment_id
     JOIN customers c
     ON p.customer_id = c.customer_id
 WHERE o.order_date BETWEEN '2020-01-01' AND '2020-12-31'
 GROUP BY
     c.customer_id
 ORDER BY SUM(pr.unit_price * p.quantity) DESC ;
```



Year	customer_id	firstname	lastname	Purchase $
2020	12915	Josephine	murphy	270576
2020	12868	Thelma	ahmed	234549
2020	12472	Nellie	hernandez	168300
2020	15111	Roosevelt	robertson	159800
2020	12971	Addie	rose	149780
2020	16456	Jessie	walker	147850
2020	12791	Charles	harrison	135700
2020	15605	Ralph	ortiz	116860
2020	17287	Myrtle	lim	102000
2020	17412	Beulah	diaz	100000
2020	14911	Lee	castillo	99000
2020	17838	Sadie	ward	92130
2020	13011	Effie	gray	89450
2020	17908	Joseph	morales	88500
2020	16250	John	foster	88200

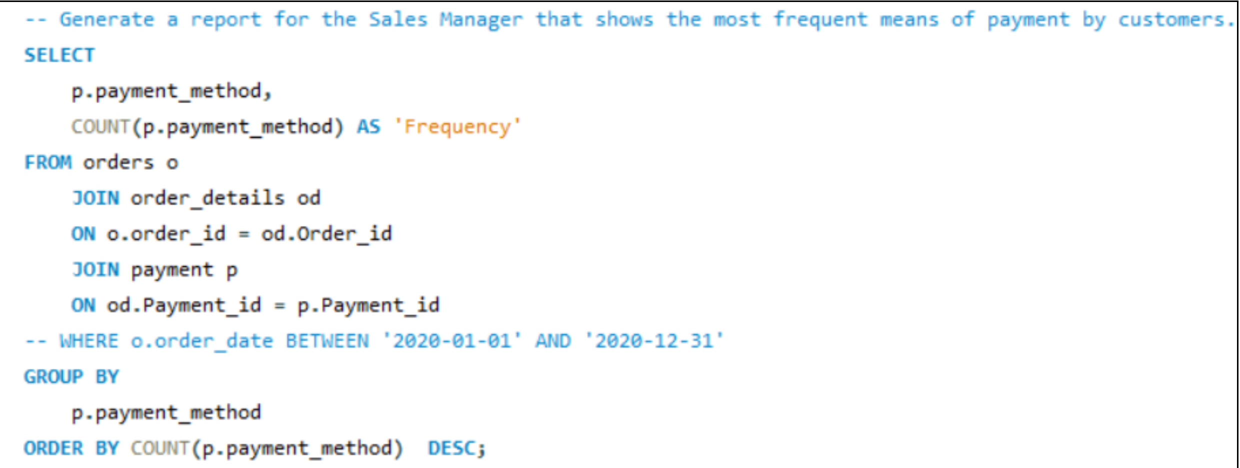
```sql
-- To get the least profitable customers by gross sales, use the query below
SELECT
    YEAR(o.order_date) 'Year',
    c.customer_id,
    c.firstname,
    c.lastname,
    SUM(pr.unit_price * p.quantity) 'Purchase $'
FROM orders o
    JOIN order_details od
    ON o.order_id = od.Order_id
    JOIN products pr
    ON od.product_id = pr.product_id
    JOIN payment p
    ON od.Payment_id = p.Payment_id
    JOIN customers c
    ON p.customer_id = c.customer_id
WHERE o.order_date BETWEEN '2020-01-01' AND '2020-12-31'
GROUP BY
    c.customer_id
ORDER BY SUM(pr.unit_price * p.quantity);
```



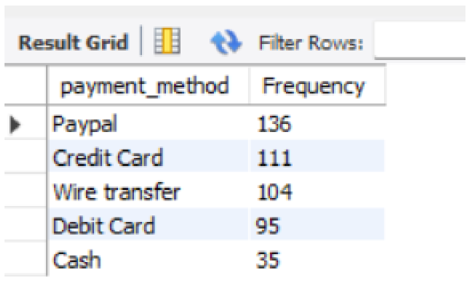
Result Grid | Filter Rows: | Export:

Year	customer_id	firstname	lastname	Purchase $
2020	17181	Fred	perry	75
2020	14865	Gussie	brooks	90
2020	18168	Archie	david	120
2020	13798	Bill	porter	150
2020	15862	Emma	thomas	150
2020	16928	Lucille	robinson	168
2020	14001	Jessie	kennedy	174
2020	13047	Nora	peterson	196
2020	17069	Clifford	wang	216
2020	16835	Julia	shah	240
2020	18144	Martha	patel	240

USE CASE 26



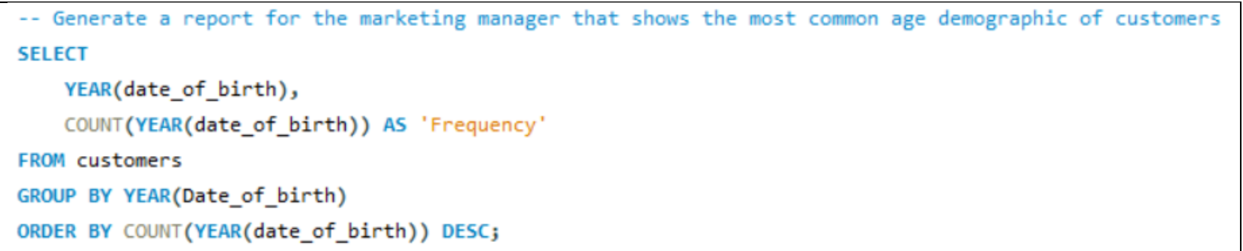
```
-- Generate a report for the Sales Manager that shows the most frequent means of payment by customers.
SELECT
    p.payment_method,
    COUNT(p.payment_method) AS 'Frequency'
FROM orders o
    JOIN order_details od
    ON o.order_id = od.Order_id
    JOIN payment p
    ON od.Payment_id = p.Payment_id
-- WHERE o.order_date BETWEEN '2020-01-01' AND '2020-12-31'
GROUP BY
    p.payment_method
ORDER BY COUNT(p.payment_method)  DESC;
```



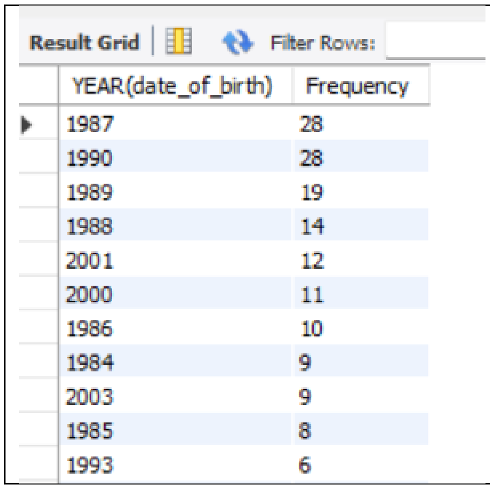
Result Grid | Filter Rows:

payment_method	Frequency
Paypal	136
Credit Card	111
Wire transfer	104
Debit Card	95
Cash	35

USE CASE 27



```
-- Generate a report for the marketing manager that shows the most common age demographic of customers
SELECT
    YEAR(date_of_birth),
    COUNT(YEAR(date_of_birth)) AS 'Frequency'
FROM customers
GROUP BY YEAR(Date_of_birth)
ORDER BY COUNT(YEAR(date_of_birth)) DESC;
```


Result Grid | Filter Rows:

```
YEAR(date_of_birth)   Frequency
1987                  28
1990                  28
1989                  19
1988                  14
2001                  12
2000                  11
1986                  10
1984                  9
2003                  9
1985                  8
1993                  6
```

TEST PLANS AND RECORD				
USE CASE	**INPUT DATA**	**EXPECTED OUTPUT**	**ACTUAL OUTPUT**	**RESULT**

1	**INSERT INTO** employees (Firstname, Lastname, Gender, Salary, Marital_Status, Phone, email, street_address, city, Zipcode)
	
	VALUES ('Alexia', 'Smith', 'F', '55000', 'married', '945-654-9857', 'alexiasmith2@pbs.com', '4 Ridgeway Valley', 'Brampton', '2484');
		Inserted a new employee information into employee table; employee_ID was auto incremented. New employee with Firstname 'Alexia' was added to the employee table.	An employee with Firstname 'Alexia' was added to the employee table.	**Pass**

2	**UPDATE** employees
	SET Firstname = 'Brenda', Lastname = 'Olav', Phone = '938-3474-2416'
	WHERE employee_id = 248151;
		Updated the employee with employee_id: 248151. the first name to Brenda, Last name to Olav and phone number to 938-3474-2416 in the employee table	The phone number of employee with employee_id: 248151 was updated to 938-3474-2416, firstname updated to Brenda and last name updated to Olav.	**Pass**

3	**DELETE FROM** employees **WHERE** Firstname = 'Dolly' **AND** Lastname = 'Day';	Deleted employee information from employee table where the firstname of the employee is ‘Dolly’ and lastname is ‘Day’	Employee with firstname Dolly and lastname Day was deleted and no longer exists in the employee table.	**Pass**

4	**INSERT INTO** customers (Firstname, Lastname, Gender, Date_of_birth, Street_address, City, Province, Membership, Phone, email)
	
	VALUES ('Sunny', 'Blackson', 'M', '2000-01-01', '24 Crayville Road', 'Waterloo', 'Ontario', 'Basic', '805-948-9375', 'sunny44@gmail.com');
		Inserted a new customer information into customer table; customer_ID was auto incremented. New customer with Firstname ‘Sunny’ was added to the customer table.	A customer with Firstname ‘Sunny’ was added to the customer table.	**Pass**

5	**UPDATE** customers **SET** Firstname = 'Louis', Lastname = 'Magnus', Membership = 'Premuim' **WHERE** Customer_id = 12472;	Updated the customer with customer_id: 12472. the first name to Louis, Last name to Magnus and Membership to ‘premium’ in the customer table	The firstname of employee with customer_id: 12472 was updated to Louis, lastname updated to Magnus and membership updated to Premuim	**Pass**
6	**DELETE FROM** customers **WHERE** Customer_id= 12748**;**	Deleted customer information from customer table where customer_id is 12748	Customer with customer_id 12748 was deleted and no longer exists in the customer table	**Pass**

7	**INSERT INTO** department (dept_name)
	
	VALUES ('Accounts');
		Inserted new department information into department table; dept_ID was auto incremented. New department with dept_name ‘Accounts’ was	A department with dept_name ‘Accounts’ was added to the department table	**Pass**

		added to the department table.		

8	**UPDATE** department
	SET dept_name = Customer_Care
	WHERE dept_id = 251;
		Updated the department with dept_id: 251. the dept_name to Customer_Care in the department table	The dept_name of the department with dept_id 251 was updated to Customer_Care.	**Pass**

9	**DELETE FROM** department
	WHERE dept_id= 251;
		Deleted department information from department table where dept_id is 251	Department with dept_id 251 was deleted and no longer exists on the department table.	**Pass**

10	**INSERT INTO** orders (customer_id, employee_id, shipper_id, order_date, status**)**
	VALUES ('12430', '248150', '2353', '2019-01-05', 'Delivered');
		Inserted new orders information into orders table; order_ID was auto incremented. New order with customer_id 12430 and employee_id 248150 was added to the orders table.	An order with customer_id 12430 and employee_id 248150 was added to the orders table.	**Pass**

11	**UPDATE** orders
	SET status = Processing
	WHERE order_id = 151205;
		Updated the order with order_id 151205. Status was updated to processing.	The order status of the order_id 151205 was updated to processing.	**Pass**

12	**DELETE FROM** orders
	WHERE order_id= 151204;
		Deleted orders with order_id 151204 from the orders table	Order with order_id 151204 was deleted from the orders table and no longer exists in the orders table	**Pass**

13	**INSERT INTO** payment (customer_id, payment_method, payment_date, quantity)
	VALUES ('12429', 'Debit Card', '2020-05-01', '56');
		Inserted new payment information into payment table; payment_ID was auto incremented.	A payment with customer_id 12429 was added to the payment table.	**Pass**

		New payment with customer_id 12429 was added to the orders table.		

14	**UPDATE payment**
	SET quantity = 35
	WHERE payment_id = 5;
		Updated the payment with payment_id 5. Quantity was updated to 35	The quantity was updated to 35 for payment with payment_id 5.	**Pass**

15	**DELETE FROM** payment
	WHERE payment_id= 4;
		Delete payment with payment_id 4 from payment table.	The payment with payment_id 4 was deleted and no longer exists on the payment_table	**Pass**

16	**INSERT INTO** shippers (shipper_name)
	VALUES ('Florentine Shipping');
		Inserted new shipper information into the shippers table. Shipper with shipper name 'Florentine shipping' was added to the shippers table.	A shipper with shipper name 'Florentine shipping' was added to the shippers table.	**Pass**

17	**UPDATE** shippers
	SET shipper_name = 'Bella Cruise LTD'
	WHERE shipper_id = 2355;
		Updated the shipper with shipper_id 2355. Shipper name was updated to 'Bella Cruise LTD'	The shipper name was updated to 'Bella Cruise LTD' for shipper with shipper_id 2355 in the shippers table.	**Pass**

18	**DELETE FROM** shippers
	WHERE shipper_id= 2355;
		Delete shipper with shipper_id 2355 from shippers table.	The shipper with shipper id 2355 was deleted from the shippers_table and no longer exists in the shippers table.	**Pass**

19	**INSERT INTO** products (Product, Unit_price, Quantity_in_stock)
	VALUES ('Samsung Dishwasher', '463', '40');
		Inserted new product information into the products table. Product with product name 'Samsung Dishwasher' was	A Product with product name 'Samsung Dishwasher' was added to the products table.	**Pass**

		added to the products table.		

20	**UPDATE** products
	SET Product = 'Samsung Dryer'
	WHERE product_id = 17;
		Updated the product with product_id 17. Product name was updated to 'Samsung Dryer'	The product name was updated to 'Samsung Dryer' in the products table.	**Pass**

21	**DELETE FROM** products
	WHERE product_id= 19;
		Delete product with product_id 19 from products table.	Product with product_id 19 was deleted from the products table and no longer exists in the products table.	**Pass**

22	**SELECT** o.employee_id, e.Firstname, e.Lastname, e.gender, COUNT(*) AS 'Total_Sales'
	FROM orders o
	JOIN employees e
	ON o.employee_id = e.employee_id
	WHERE o.order_date
	BETWEEN '2020-01-01' AND '2021-12-31'
	GROUP BY o.employee_id
	ORDER BY Total_sales **DESC**
	LIMIT 2**;**
	
	SELECT o.employee_id, e.Firstname, e.Lastname, e.gender, COUNT(*) AS 'Total_Sales'
	FROM orders o
	JOIN employees e
	ON o.employee_id = e.employee_id
	WHERE o.order_date
	BETWEEN '2020-01-01' AND '2021-12-31'
	GROUP BY o.employee_id
	ORDER BY Total_sales
	LIMIT 2**;**
		Generate a report which shows the top 2 employees with the highest sales volume for the year 2020 and 2021. Also show the employees with the lowest sales volume for 2020 and 2021**.**	A report which showed the top 2 employees with the highest sales volume and the bottom 2 sales volume for the year 2020 and 2021 was generated	**Pass**

23	**SELECT** **YEAR**(o.order_date) 'Sales Year',	Generate a monthly sales report for 2020	A monthly sales report for 2020 for the Sales Manager	**Pass**

	MONTH(o.order_date) 'Sales Month', **COUNT**(o.order_id) 'Sales Volume', **SUM**(pr.unit_price * p.quantity) 'Gross Sales $' **FROM** orders o **JOIN** order_details od **ON** o.order_id = od.Order_id **JOIN** products pr **ON** od.product_id = pr.product_id **JOIN** payment p **ON** od.Payment_id = p.Payment_id **WHERE** o.order_date **BETWEEN** '2020-01-01' AND '2020-12-31' **GROUP BY** **MONTH**(o.order_date);	for the Sales Manager which shows the sales volume -- (number of sales made) and Gross Sales (total sales in dollars)	which shows the sales volume and Gross Sales was generated	
24	**SELECT** **YEAR**(o.order_date) 'Year', pr.product_id, pr.product AS 'Top Selling', pr.Unit_price, **SUM**(pr.unit_price * p.quantity) 'Sales $' **FROM** orders o **JOIN** order_details od **ON** o.order_id = od.Order_id **JOIN** products pr **ON** od.product_id = pr.product_id **JOIN** payment p **ON** od.Payment_id = p.Payment_id **JOIN** customers c **ON** p.customer_id = c.customer_id **WHERE** o.order_date **BETWEEN** '2020-01-01' AND '2020-12-31' **GROUP BY**	Create a report that shows the top 3 highest selling products in 2020.	A report for the Sales Manager that shows the top 3 highest selling products in 2020 by gross sales and by sales volume	**Pass**

	c.customer_id **ORDER BY SUM**(pr.unit_price * p.quantity) **DESC** **LIMIT** 3**;** This query below gives the top 3 highest selling product by sales volume. SELECT o.order_id, pr.product_id, pr.product AS 'Top Selling', pr.unit_price, **SUM**(p.quantity) **AS** Sales_Volume **FROM** orders o **JOIN** order_details od **ON** o.Order_id = od.Order_id **JOIN** products pr **ON** od.Product_id = pr.Product_id **JOIN** payment p **ON** od.payment_id = p.payment_id **WHERE** o.order_date **BETWEEN** '2020-01-01' AND '2020-12-31' **GROUP BY** pr.Product **ORDER BY SUM(**p.quantity) **DESC** **LIMIT** 3**;**			
25	**SELECT** **YEAR**(o.order_date) 'Year', c.customer_id, c.firstname, c.lastname, **SUM**(pr.unit_price * p.quantity) 'Purchase $' **FROM** orders o **JOIN** order_details od **ON** o.order_id = od.Order_id **JOIN** products pr	Create a report for the that shows the most profitable customers in 2020 by gross sales and sales volume.	A report for the that shows the most profitable customers in 2020 by gross sales and by sales volume was generated.	**Pass**

```
    ON od.product_id =
pr.product_id
    JOIN payment p
    ON od.Payment_id =
p.Payment_id
    JOIN customers c
    ON p.customer_id =
c.customer_id
WHERE o.order_date
BETWEEN '2020-01-01' AND
'2020-12-31'
GROUP BY
    c.customer_id
ORDER BY SUM(pr.unit_price
* p.quantity) DESC ;

-- To get the least profitable
customers by gross sales, use the
query below

SELECT
            YEAR(o.order_date)
'Year',
    c.customer_id,
    c.firstname,
    c.lastname,
    SUM(pr.unit_price *
p.quantity) 'Purchase $'
FROM orders o
            JOIN order_details od
            ON o.order_id =
od.Order_id
    JOIN products pr
    ON od.product_id =
pr.product_id
    JOIN payment p
    ON od.Payment_id =
p.Payment_id
    JOIN customers c
    ON p.customer_id =
c.customer_id
WHERE o.order_date
BETWEEN '2020-01-01' AND
'2020-12-31'
GROUP BY
    c.customer_id
```

	ORDER BY SUM(pr.unit_price * p.quantity);			
26	**SELECT** p.payment_method, **COUNT**(p.payment_method) AS 'Frequency' **FROM** orders o **JOIN** order_details od **ON** o.order_id = od.Order_id **JOIN** payment p **ON** od.Payment_id = p.Payment_id **GROUP BY** p.payment_method **ORDER BY COUNT**(p.payment_method) **DESC;**	Generate a report that shows the most frequent means of payment by customers.	A report that shows the most frequent means of payment by customers.	**Pass**
27	**SELECT YEAR**(date_of_birth), **COUNT(YEAR(date_of_birth)) AS** 'Frequency' **FROM** customers **GROUP BY YEAR**(Date_of_birth) **ORDER BY COUNT**(**YEAR**(date_of_birth)) **DESC;**	Generate a report that shows the most common age demographic of customers	A report that shows the most common age demographic of customers	**Pass**

References

Teorey T., Lightstone S & Nadeau T. Database *Modeling and Design, Fifth Edition: Logical Design (The Morgan Kaufmann Series in Data Management Systems)* Morgan Kaufmann Publisher 2011.

https://www.lucidchart.com/