SQL Queries for 370 Project - Store Management System

CREATING THE TABLE:

--------------------------------------------------------------------------------------------------------

CREATE DATABASE `storemanagerdb`;

USE `storemanagerdb`;

CREATE TABLE `customer`(

`customer\_id` INT,

`name` VARCHAR(180),

`birth\_date` DATE,

`phone\_number` VARCHAR(20),

`email` VARCHAR(180),

`address` VARCHAR(255)

);

LOAD DATA LOCAL INFILE '/path/customer.csv'

INTO TABLE customer

FIELDS TERMINATED BY ','

OPTIONALLY ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 LINES;

CREATE TABLE `item`(

`item\_id` INT,

`item\_name` VARCHAR(180),

`price` INT,

`cost` FLOAT

);

LOAD DATA LOCAL INFILE '/path/item.csv'

INTO TABLE `item`

FIELDS TERMINATED BY ','

OPTIONALLY ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 LINES;

CREATE TABLE `store`(

`store\_id` INT,

`store\_name` VARCHAR(180),

`manager\_id` INT,

`location` VARCHAR(250)

);

LOAD DATA LOCAL INFILE '/path/store.csv'

INTO TABLE `store`

FIELDS TERMINATED BY ','

OPTIONALLY ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 LINES;

CREATE TABLE `purchases`(

`purchase\_id` INT,

`customer\_id` INT,

`item\_id` INT,

`store\_id` INT,

`date` DATE

);

LOAD DATA LOCAL INFILE '/path/purchases.csv'

INTO TABLE `purchases`

FIELDS TERMINATED BY ','

OPTIONALLY ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 LINES;

CREATE TABLE `stock`(

`store\_id` INT,

`item\_id` INT,

`quantity` INT

);

LOAD DATA LOCAL INFILE '/path/stock.csv'

INTO TABLE `stock`

FIELDS TERMINATED BY ','

OPTIONALLY ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 LINES;

CREATE TABLE `managers`(

`manager\_id` INT,

`store\_id` INT,

`manager\_name` VARCHAR(180),

`mysql\_username` VARCHAR(180)

);

LOAD DATA LOCAL INFILE '/path/managers.csv'

INTO TABLE `managers`

FIELDS TERMINATED BY ','

OPTIONALLY ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 LINES;

GENERATING KEYS:

--------------------------------------------------------------------------------------------------------

// Primary Keys::

ALTER TABLE `item`

ADD PRIMARY KEY (`item\_id`);

ALTER TABLE `customer`

ADD PRIMARY KEY (`customer\_id`);

ALTER TABLE `store`

ADD PRIMARY KEY (`store\_id`);

// Foreign Keys::

// for purchases

ALTER TABLE `purchases`

ADD CONSTRAINT `fk\_customer\_id`

FOREIGN KEY (`customer\_id`)

REFERENCES `customer`(`customer\_id`);

ALTER TABLE `purchases`

ADD CONSTRAINT `fk\_item\_id`

FOREIGN KEY (`item\_id`)

REFERENCES `item`(`item\_id`);

ALTER TABLE `purchases`

ADD CONSTRAINT `fk\_store\_id`

FOREIGN KEY (`store\_id`)

REFERENCES `store`(`store\_id`);

// for stocks

ALTER TABLE `stock`

ADD CONSTRAINT `fk\_store\_id\_stock`

FOREIGN KEY (`store\_id`)

REFERENCES `store`(`store\_id`);

ALTER TABLE `stock`

ADD CONSTRAINT `fk\_item\_id\_stock`

FOREIGN KEY (`item\_id`)

REFERENCES `item`(`item\_id`);

DML(INSERT, UPDATE, DELETE) QUERIES:

--------------------------------------------------------------------------------------------------------

// Customer

//INSERT (Adding a customer to the database)

INSERT INTO `customer` (customer\_id, name, birth\_date, phone\_number, email, address)

VALUES (1, 'Larry Peters', '2001-07-01', '123-456-7890', 'larry.pete@gmail.com', '123 Elms Street');

//UPDATE (Updating a customers information in the database)

// example: updating name

UPDATE `customer`

SET `name` = 'Maria Lamb'

WHERE `customer\_id` = 1;

// DELETE (Deleting a customer from the database)

// Due to Foreign Key constrains we can't delete a customer from the database, because that customer is linked to a purchase history

DELETE FROM customer

WHERE customer\_id = 1;

---------------------------------------------

// Store

// INSERT (Adding a store to the database)

INSERT INTO `store` (store\_id, store\_name, manager\_id, location, revenue)

VALUES(20, 'Water Equipment Galore', 511,'89 Hilltop Rd Saanich BC A2B2B3', 33000);

//UPDATE (Updating a store information in the database)

// example: updating address

UPDATE `store`

SET `location` = '456 Elms Street'

WHERE `store\_id` = 501;

---------------------------------------------

// Items

// INSERT (Adding an item to the database)

INSERT INTO `item` (item\_id, item\_name, price, cost)

VALUES(9998, 'Rubber Ducky Yellow', 5, 2);

//UPDATE (Updating an item information in the database)

// example: updating price

UPDATE `item`

SET `price` = 6

WHERE `store\_id` = 9998;

COMPLEX QUERIES FOR OUR SYSTEM:

--------------------------------------------------------------------------------------------------------

// Top selling products::

SELECT `item\_name`, `price`, `cost`, (price - cost) AS 'profit'

FROM `item`

ORDER BY profit DESC

LIMIT 5;

// Highest-yielding purchases::

SELECT `purchases`.`purchase\_id`,

SUM(`item`.`price` - `item`.`cost`) AS total\_profit

FROM `purchases`

JOIN `item` ON `purchases`.`item\_id` = `item`.`item\_id`

GROUP BY `purchases`.`purchase\_id`

ORDER BY total\_profit DESC

LIMIT 5;

// Highest-yielding customers::

SELECT `customer`.`customer\_id`, `customer`.`name`,

SUM(`item`.`price` - `item`.`cost`) AS total\_spent

FROM `customer`

JOIN `purchases` ON `customer`.`customer\_id` = `purchases`.`customer\_id`

JOIN `item` ON `purchases`.`item\_id` = `item`.`item\_id`

GROUP BY `customer`.`customer\_id`, `customer`.`name`

ORDER BY total\_spent DESC

LIMIT 5;

// Calculate store revenue (by store\_id)::

SELECT `store`.`store\_id`, `store`.`store\_name`,

SUM(`item`.`price`) AS total\_revenue

FROM `store`

JOIN `purchases` ON `store`.`store\_id` = `purchases`.`store\_id`

JOIN `item` ON `purchases`.`item\_id` = `item`.`item\_id`

WHERE `store`.`store\_id` = 1

GROUP BY `store`.`store\_id`, `store`.`store\_name`;

// Monthly sales trends::

SELECT

YEAR(`purchases`.`date`) AS sales\_year,

MONTH(`purchases`.`date`) AS sales\_month,

SUM(`item`.price) AS total\_sales

FROM `purchases`

JOIN `item` ON `purchases`.`item\_id` = `item`.`item\_id`

GROUP BY YEAR(`purchases`.`date`), MONTH(`purchases`.`date`)

ORDER BY total\_sales DESC;

// Highest-yielding customers by date::

SELECT `customer`.`customer\_id`, `customer`.`name`,

SUM(`item`.`price` - `item`.`cost`) AS total\_spent

FROM `customer`

JOIN `purchases` ON `customer`.`customer\_id` = `purchases`.`customer\_id`

JOIN `item` ON `purchases`.`item\_id` = `item`.`item\_id`

WHERE `purchases`.`date` BETWEEN '2023-01-01' AND '2023-12-31'

GROUP BY `customer`.`customer\_id`, `customer`.`name`

ORDER BY total\_spent DESC

LIMIT 5;

// Looking up a customer purchase history (by customer id)::

SELECT `purchases`.`purchase\_id`, `purchases`.`purchase\_date`,

SUM(`item`.`price`) AS total\_price

FROM `purchases`

JOIN `item` ON `purchases`.`item\_id` = `item`.`item\_id`

JOIN `customer` ON `purchases`.`customer\_id` = `customer`.`customer\_id`

WHERE `customer`.`customer\_id` = ?

GROUP BY `purchases`.`purchase\_id`, `purchases`.`purchase\_date`

ORDER BY `purchases`.`purchase\_date` DESC;

// Looking customers that spend over a certain amount::

SELECT `customer`.`customer\_id`, `customer`.`name`,

SUM(`item`.`price`) AS total\_spent

FROM `customer`

JOIN `purchases` ON `customer`.`customer\_id` = `purchases`.`customer\_id`

JOIN `item` ON `purchases`.`item\_id` = `item`.`item\_id`

GROUP BY `customer`.`customer\_id`, `customer`.`name`

HAVING SUM(`item`.`price`) > 1000

ORDER BY total\_spent DESC;

SUB QUERIES FOR OUR SYSTEM:

--------------------------------------------------------------------------------------------------------

// Customers who have made purchases in a specific store::

SELECT `customer\_id`, `name`

FROM `customer`

WHERE `customer`.`customer\_id` IN (

SELECT `purchases`.`customer\_id`

FROM `purchases`

WHERE `purchases`.`store\_id` = 1

);

// Average order value::

SELECT AVG(order\_total) AS average\_order\_value

FROM (

SELECT `purchases`.`purchase\_id`, SUM(`item`.`price`) AS order\_total

FROM `purchases`

JOIN `item` ON `purchases`.`item\_id` = `item`.`item\_id`

GROUP BY `purchases`.`purchase\_id`

) AS order\_totals;

// Customers who have never made a purchase::

SELECT `customer\_id`, `name`

FROM `customer`

WHERE NOT EXISTS (

SELECT 1

FROM `purchases`

WHERE `purchases`.`customer\_id` = `customer`.`customer\_id`

);