Create databases, tables and insert data.

1. Create the tables for your selected systems.

Small system - Groceries
Large system - Canvas

- Show Databases

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| canvas             |
| db_book            |
| groceries          |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
7 rows in set (0.00 sec)
```

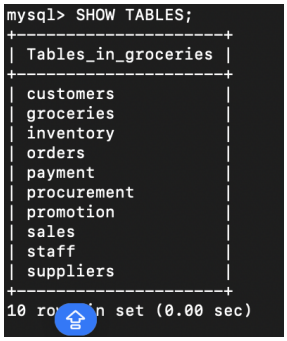
```
mysql> SHOW TABLES;
+----------------------+
| Tables_in_canvas     |
+----------------------+
| assignment           |
| attendance_records   |
| course               |
| course_announcements |
| discussion_groups    |
| exams                |
| feedback             |
| grades               |
| instructor           |
| modules              |
| notifications        |
| office_hours         |
| profile              |
| resources            |
| student              |
| teaching_assistant   |
+----------------------+
16 rows in set (0.00 sec)

mysql>
```

- Select database

```
mysql> SELECT DATABASE();
+------------+
| DATABASE() |
+------------+
| groceries  |
+------------+
```

- Display all tables in the groceries database



- Describe all tables from groceries database

```
mysql> DESCRIBE customers;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| id         | int          | NO   | PRI | NULL    |       |
| fname      | varchar(50)  | YES  |     | NULL    |       |
| lname      | varchar(50)  | YES  |     | NULL    |       |
| email      | varchar(255) | YES  |     | NULL    |       |
| SSN        | varchar(11)  | YES  |     | NULL    |       |
| phone_num  | char(10)     | YES  |     | NULL    |       |
| birth_date | date         | YES  |     | NULL    |       |
| address    | varchar(255) | YES  |     | NULL    |       |
| start_date | date         | YES  |     | NULL    |       |
| end_date   | date         | YES  |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
10 rows in set (0.00 sec)

mysql> DESCRIBE groceries;
+-------------------+--------------+------+-----+---------+-------+
| Field             | Type         | Null | Key | Default | Extra |
+-------------------+--------------+------+-----+---------+-------+
| id                | int          | NO   | PRI | NULL    |       |
| name              | varchar(50)  | YES  |     | NULL    |       |
| brand             | varchar(50)  | YES  |     | NULL    |       |
| tags              | varchar(50)  | YES  |     | NULL    |       |
| nutri_infor       | varchar(50)  | YES  |     | NULL    |       |
| category          | varchar(50)  | YES  |     | NULL    |       |
| packaging_type    | varchar(50)  | YES  |     | NULL    |       |
| date_manufactured | date         | YES  |     | NULL    |       |
| expirary_date     | date         | YES  |     | NULL    |       |
| price             | decimal(5,2) | YES  |     | NULL    |       |
+-------------------+--------------+------+-----+---------+-------+
10 rows in set (0.00 sec)

mysql> DESCRIBE inventory;
+--------------+--------------+------+-----+---------+-------+
| Field        | Type         | Null | Key | Default | Extra |
+--------------+--------------+------+-----+---------+-------+
| id           | int          | NO   | PRI | NULL    |       |
| stock_amount | int          | YES  |     | NULL    |       |
| status       | varchar(50)  | YES  |     | NULL    |       |
| location     | varchar(255) | YES  |     | NULL    |       |
| date_added   | date         | YES  |     | NULL    |       |
| last_updated | date         | YES  |     | NULL    |       |
| supplier_id  | int          | NO   | MUL | NULL    |       |
+--------------+--------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

```
mysql> DESCRIBE orders;
+------------------+--------------+------+-----+---------+-------+
| Field            | Type         | Null | Key | Default | Extra |
+------------------+--------------+------+-----+---------+-------+
| id               | int          | NO   | PRI | NULL    |       |
| status           | varchar(50)  | YES  |     | NULL    |       |
| payment_method   | varchar(50)  | YES  |     | NULL    |       |
| ordered_date     | date         | YES  |     | NULL    |       |
| cost             | decimal(5,2) | YES  |     | NULL    |       |
| delivery_method  | varchar(50)  | YES  |     | NULL    |       |
| shipping_address | varchar(255) | YES  |     | NULL    |       |
| customer_id      | int          | YES  | MUL | NULL    |       |
+------------------+--------------+------+-----+---------+-------+
8 rows in set (0.00 sec)

mysql> DESCRIBE payment;
+------------------+--------------+------+-----+---------+-------+
| Field            | Type         | Null | Key | Default | Extra |
+------------------+--------------+------+-----+---------+-------+
| id               | int          | NO   | PRI | NULL    |       |
| payment_method   | varchar(50)  | YES  |     | NULL    |       |
| payment_date     | date         | YES  |     | NULL    |       |
| payment_amount   | decimal(5,2) | YES  |     | NULL    |       |
| status           | varchar(50)  | YES  |     | NULL    |       |
| purchase_history | varchar(50)  | YES  |     | NULL    |       |
| customer_id      | int          | NO   | MUL | NULL    |       |
| grocery_id       | int          | NO   | MUL | NULL    |       |
| staff_id         | int          | NO   | MUL | NULL    |       |
| order_id         | int          | NO   | MUL | NULL    |       |
+------------------+--------------+------+-----+---------+-------+
10 rows in set (0.00 sec)

mysql> DESCRIBE procurement;
+-----------------+--------------+------+-----+---------+-------+
| Field           | Type         | Null | Key | Default | Extra |
+-----------------+--------------+------+-----+---------+-------+
| id              | int          | NO   | PRI | NULL    |       |
| notes           | varchar(50)  | YES  |     | NULL    |       |
| date_arrived    | date         | YES  |     | NULL    |       |
| date_expired    | date         | YES  |     | NULL    |       |
| payment_status  | varchar(50)  | YES  |     | NULL    |       |
| delivery_status | varchar(50)  | YES  |     | NULL    |       |
| cost            | decimal(5,2) | YES  |     | NULL    |       |
| quantity        | int          | YES  |     | NULL    |       |
+-----------------+--------------+------+-----+---------+-------+
8 rows in set (0.00 sec)
```

```
mysql> DESCRIBE promotion;
+---------------------+--------------+------+-----+---------+-------+
| Field               | Type         | Null | Key | Default | Extra |
+---------------------+--------------+------+-----+---------+-------+
| id                  | int          | NO   | PRI | NULL    |       |
| name                | varchar(50)  | YES  |     | NULL    |       |
| discount_percentage | decimal(5,2) | YES  |     | NULL    |       |
| max_discount        | decimal(5,2) | YES  |     | NULL    |       |
| promotion_type      | varchar(50)  | YES  |     | NULL    |       |
| start_date          | datetime(3)  | YES  |     | NULL    |       |
| end_date            | datetime(3)  | YES  |     | NULL    |       |
| applicable_products | varchar(50)  | YES  |     | NULL    |       |
+---------------------+--------------+------+-----+---------+-------+
8 rows in set (0.00 sec)

mysql> DESCRIBE sales;
+------------------+-------------+------+-----+---------+-------+
| Field            | Type        | Null | Key | Default | Extra |
+------------------+-------------+------+-----+---------+-------+
| id               | int         | NO   | PRI | NULL    |       |
| quantity         | int         | YES  |     | NULL    |       |
| purchase_history | varchar(50) | YES  |     | NULL    |       |
| revenue          | int         | YES  |     | NULL    |       |
| birth_date       | date        | YES  |     | NULL    |       |
| start_date       | date        | YES  |     | NULL    |       |
| customer_id      | int         | NO   | MUL | NULL    |       |
| staff_id         | int         | NO   | MUL | NULL    |       |
| grocery_id       | int         | NO   | MUL | NULL    |       |
+------------------+-------------+------+-----+---------+-------+
9 rows in set (0.00 sec)

mysql> DESCRIBE staff;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| id         | int          | NO   | PRI | NULL    |       |
| fname      | varchar(50)  | YES  |     | NULL    |       |
| lname      | varchar(50)  | YES  |     | NULL    |       |
| salary     | decimal(8,2) | YES  |     | NULL    |       |
| location   | varchar(50)  | YES  |     | NULL    |       |
| work_start | datetime(3)  | YES  |     | NULL    |       |
| work_end   | datetime(3)  | YES  |     | NULL    |       |
| role       | varchar(30)  | YES  |     | NULL    |       |
| hire_date  | date         | YES  |     | NULL    |       |
| grocery_id | int          | NO   | MUL | NULL    |       |
+------------+--------------+------+-----+---------+-------+
10 rows in set (0.00 sec)
```

```
mysql> DESCRIBE suppliers;
+--------------------+--------------+------+-----+---------+-------+
| Field              | Type         | Null | Key | Default | Extra |
+--------------------+--------------+------+-----+---------+-------+
| id                 | int          | NO   | PRI | NULL    |       |
| name               | varchar(50)  | YES  |     | NULL    |       |
| phone_num          | char(10)     | YES  |     | NULL    |       |
| rating             | int          | YES  |     | NULL    |       |
| email              | varchar(255) | YES  |     | NULL    |       |
| address            | varchar(255) | YES  |     | NULL    |       |
| products_supplied  | varchar(50)  | YES  |     | NULL    |       |
| delivery_per_month | int          | YES  |     | NULL    |       |
| payment_term       | int          | YES  |     | NULL    |       |
+--------------------+--------------+------+-----+---------+-------+
9 rows in set (0.00 sec)
```

- Show row count in promotion, customers and orders tables

```
mysql> SELECT COUNT(*)
    -> FROM promotion;
+----------+
| COUNT(*) |
+----------+
|       12 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(*)
    -> FROM customers;
+----------+
| COUNT(*) |
+----------+
|       12 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(*)
    -> FROM orders;
+----------+
| COUNT(*) |
+----------+
|       12 |
+----------+
1 row in set (0.00 sec)
```

- Show queries where the customers have the first name starting with “R”

```
mysql> SELECT fname, lname, phone_num from customers
    -> WHERE fname like 'R%';
+--------+-------+------------+
| fname  | lname | phone_num  |
+--------+-------+------------+
| Robert | Brown | 4567890123 |
+--------+-------+------------+
1 row in set (0.00 sec)
```

- Filter queries to only customers who competed payment

```
mysql> SELECT customer_id, status
    -> FROM promotion
    -> WHERE status = 'Completed';
ERROR 1054 (42S22): Unknown column 'customer_id' in 'field list'
mysql> SELECT customer_id, status, payment_method
    -> FROM payment
    -> WHERE status = 'Completed';
+-------------+-----------+----------------+
| customer_id | status    | payment_method |
+-------------+-----------+----------------+
|           1 | Completed | Credit Card    |
|           2 | Completed | Debit Card     |
|           4 | Completed | Credit Card    |
|           6 | Completed | Cash           |
|           8 | Completed | Credit Card    |
|           9 | Completed | Cash           |
|          11 | Completed | Credit Card    |
+-------------+-----------+----------------+
7 rows in set (0.00 sec)
```

a. Primary keys, foreign keys

- Keys in table Sales
  - ❖ Primary key - id (sales_id)
  - ❖ Foreign keys - customer_id
    staff_id
    grocery_id

b. Define necessary constraints for each table, such as NOT NULL, UNIQUE, DEFAULT, and CHECK Constraints.

- Describe sales tables

```
mysql> DESCRIBE sales;
+------------------+-------------+------+-----+---------+-------+
| Field            | Type        | Null | Key | Default | Extra |
+------------------+-------------+------+-----+---------+-------+
| id               | int         | NO   | PRI | NULL    |       |
| quantity         | int         | YES  |     | NULL    |       |
| purchase_history | varchar(50) | YES  |     | NULL    |       |
| revenue          | int         | YES  |     | NULL    |       |
| birth_date       | date        | YES  |     | NULL    |       |
| start_date       | date        | YES  |     | NULL    |       |
| customer_id      | int         | NO   | MUL | NULL    |       |
| staff_id         | int         | NO   | MUL | NULL    |       |
| grocery_id       | int         | NO   | MUL | NULL    |       |
+------------------+-------------+------+-----+---------+-------+
9 rows in set (0.00 sec)
```

```
mysql> SELECT *
    -> FROM sales;
+----+----------+------------------+---------+------------+------------+-------------+----------+-------------+
| id | quantity | purchase_history | revenue | birth_date | start_date | customer_id | staff_id | grocery_id |
+----+----------+------------------+---------+------------+------------+-------------+----------+-------------+
|  1 |        5 | Groceries        |     100 | 1990-01-15 | 2023-08-01 |           2 |        1 |           1 |
|  2 |        2 | Household Items  |     200 | 1985-05-20 | 2023-08-02 |           3 |        2 |           2 |
|  3 |        3 | Beverages        |      75 | 1992-11-12 | 2023-08-03 |           1 |        3 |           3 |
|  4 |        1 | Frozen Foods     |      90 | 1988-07-07 | 2023-08-04 |           6 |        4 |           4 |
|  5 |        4 | Personal Care    |     120 | 1975-02-28 | 2023-08-05 |           4 |        5 |           5 |
|  6 |       10 | Snacks           |      50 | 2000-03-17 | 2023-08-06 |           5 |        6 |           6 |
|  7 |        7 | Produce          |     130 | 1995-09-25 | 2023-08-07 |           7 |        7 |           7 |
|  8 |        6 | Bakery Items     |      85 | 1982-04-30 | 2023-08-08 |           8 |        8 |           8 |
|  9 |        9 | Canned Goods     |     110 | 1993-06-21 | 2023-08-09 |           9 |        9 |           9 |
| 10 |        8 | Dairy Products   |      75 | 1987-08-13 | 2023-08-10 |          12 |       10 |          10 |
| 11 |       12 | Condiments       |     140 | 1999-10-03 | 2023-08-11 |          11 |       11 |          11 |
| 12 |        5 | Seafood          |      95 | 1978-12-19 | 2023-08-12 |          10 |       12 |          12 |
| 13 |        5 | NULL             |     100 | 1990-01-15 | 2023-08-01 |           2 |        3 |           9 |
+----+----------+------------------+---------+------------+------------+-------------+----------+-------------+
13 rows in set (0.00 sec)
```

Basic Queries

a. Write a query to retrieve all the records from one of the tables in your database.

- All data from groceries table

```
mysql> SELECT * from groceries;
+----+---------------------------+-----------+--------------------------+-----------------------------------------------------------------------+-----------+--------------+-------------------+-----------------+-------+
| id | name                      | brand     | tags                     | nutri_infor                                                           | category  | packaging_type | date_manufactured | expiry_date     | price |
+----+---------------------------+-----------+--------------------------+-----------------------------------------------------------------------+-----------+--------------+-------------------+-----------------+-------+
|  1 | Organic Avocados          | GuacGuys  | organic, fresh, healthy  | {"calories":160,"fat":"15g","carbs":"9g"}                             | Produce   | Bag          | 2024-08-01        | 2024-09-15      |  4.99 |
|  2 | Almond Milk               | Kirkland  | vegan, dairy-free        | {"calories":30,"fat":"2.5g","carbs":"1g"}                             | Dairy     | Carton       | 2024-07-01        | 2025-07-01      |  2.49 |
|  3 | Cage-Free Eggs            | Utopihen  | protein, cage-free       | {"calories":70,"fat":"5g","protein":"6g"}                             | Dairy     | Carton       | 2024-09-01        | 2024-11-01      |  3.29 |
|  4 | Whole Wheat Bread         | Nature's  | whole grain, fiber       | {"calories":90,"fat":"1g","fiber":"4g"}                               | Bakery    | Loaf         | 2024-10-01        | 2024-11-01      |  2.99 |
|  5 | Peanut Butter             | Skippy    | protein, spread          | {"calories":190,"fat":"16g","protein":"7g"}                           | Pantry    | Jar          | 2024-03-01        | 2025-03-01      |  2.59 |
|  6 | Honey Nut Cereal          | Cheerios  | honey, nuts, crunchy     | {"calories":110,"fat":"2g","sugar":"9g"}                              | Breakfast | Box          | 2024-06-01        | 2025-06-01      |  3.79 |
|  7 | Greek Yogurt              | Fage      | protein, probiotic       | {"calories":100,"protein":"10g","sugar":"7g"}                         | Dairy     | Cup          | 2024-08-15        | 2024-09-30      |  1.29 |
|  8 | Boneless Chicken Breasts  | Tyson     | protein, lean            | {"calories":165,"protein":"31g","fat":"3.6g"}                         | Meat      | Pack         | 2024-10-10        | 2024-10-20      |  8.99 |
|  9 | Frozen Green Beans        | C&W       | vegan, frozen, healthy   | {"calories":30,"fiber":"4g","protein":"2g"}                           | Frozen    | Bag          | 2024-08-05        | 2025-08-05      |  2.49 |
| 10 | Organic Quinoa            | Vival Natural | organic, gluten-free | {"calories":120,"protein":"4g","fiber":"3g"}                          | Pantry    | Box          | 2024-05-15        | 2025-05-15      |  5.29 |
| 11 | Orange Juice              | Uncle Matt | vitamin C, refreshing   | {"calories":110,"sugar":"22g","vitamin_c":"100%"}                     | Beverages | Bottle       | 2024-09-15        | 2024-12-15      |  3.99 |
| 12 | Sharp Cheddar Cheese      | Kerrygold | cheese, dairy            | {"calories":120,"fat":"10g","protein":"7g"}                           | Dairy     | Block        | 2024-07-20        | 2024-12-20      |  4.49 |
+----+---------------------------+-----------+--------------------------+-----------------------------------------------------------------------+-----------+--------------+-------------------+-----------------+-------+
12 rows in set (0.00 sec)
```

b. Write a query to filter records based on a specific condition, sort it using ORDER BY, and limit the results using LIMIT.

- Retrieve a maximum of 5 products with expiration date in the year 2024, sorted from the cheapest products to the most expensive products.

```
mysql> SELECT *
    -> FROM groceries
    -> WHERE YEAR(expirary_date) = 2024
    -> ORDER BY price
    -> LIMIT 5;
+----+---------------------+-----------+--------------------------+-----------------------------------------------------------------------+-----------+--------------+-------------------+-----------------+-------+
| id | name                | brand     | tags                     | nutri_infor                                                           | category  | packaging_type | date_manufactured | expiry_date     | price |
+----+---------------------+-----------+--------------------------+-----------------------------------------------------------------------+-----------+--------------+-------------------+-----------------+-------+
|  7 | Greek Yogurt        | Fage      | protein, probiotic       | {"calories":100,"protein":"10g","sugar":"7g"}                         | Dairy     | Cup          | 2024-08-15        | 2024-09-30      |  1.29 |
|  4 | Whole Wheat Bread   | Nature's  | whole grain, fiber       | {"calories":90,"fat":"1g","fiber":"4g"}                               | Bakery    | Loaf         | 2024-10-01        | 2024-11-01      |  2.99 |
|  3 | Cage-Free Eggs      | Utopihen  | protein, cage-free       | {"calories":70,"fat":"5g","protein":"6g"}                             | Dairy     | Carton       | 2024-09-01        | 2024-11-01      |  3.29 |
| 11 | Orange Juice        | Uncle Matt | vitamin C, refreshing   | {"calories":110,"sugar":"22g","vitamin_c":"100%"}                     | Beverages | Bottle       | 2024-09-15        | 2024-12-15      |  3.99 |
| 12 | Sharp Cheddar Cheese | Kerrygold | cheese, dairy            | {"calories":120,"fat":"10g","protein":"7g"}                           | Dairy     | Block        | 2024-07-20        | 2024-12-20      |  4.49 |
+----+---------------------+-----------+--------------------------+-----------------------------------------------------------------------+-----------+--------------+-------------------+-----------------+-------+
5 rows in set (0.00 sec)
```

2. Aggregate Functions

a. Use aggregate functions to calculate the sum, average, maximum, or minimum, count of a numerical column in one of your tables.

- Total revenue if all grocery products are sold, the average price of these products, and the total number of unique product categories

```
mysql> SELECT SUM(price) AS total_sales, AVG(price) AS avg_price, COUNT(DISTINCT category) AS num_of_category
    -> FROM groceries;
+-------------+-----------+-----------------+
| total_sales | avg_price | num_of_category |
+-------------+-----------+-----------------+
|       46.68 |  3.890000 |               8 |
+-------------+-----------+-----------------+
```

3. Joins

a. Write an INNER JOIN query to join 3 tables using foreign keys.

- Joining the Groceries, Sales and Staff tables and displaying only the First names and Salaries of the staff members, and the total revenue HEB made

```
mysql> SELECT s2.fname, s2.salary, s1.revenue
    -> FROM groceries AS g
    -> INNER JOIN sales AS s1 ON g.id = s1.grocery_id
    -> INNER JOIN staff AS s2 ON g.id = s2.grocery_id;
+---------+----------+---------+
| fname   | salary   | revenue |
+---------+----------+---------+
| John    | 45000.00 |     100 |
| Emma    | 50000.00 |     200 |
| Michael | 47000.00 |     100 |
| Emily   | 43000.00 |      75 |
| Daniel  | 55000.00 |      95 |
| Sophia  | 46000.00 |     100 |
| James   | 47000.00 |      75 |
| Olivia  | 52000.00 |     140 |
| Liam    | 44000.00 |     200 |
| Ava     | 48000.00 |     100 |
| Noah    | 53000.00 |      75 |
| Mia     | 45000.00 |      75 |
+---------+----------+---------+
12 rows in set (0.00 sec)
```

b. Write a LEFT JOIN query that displays all the records from one table and only the matching records from another table.

- Left join the Groceries and the Sales tables

SELECT g.*, s1.*
FROM groceries
LEFT JOIN sales AS s1 ON g.id = s1.grocery_id;

```
mysql> SELECT g.*, s1.*
    -> FROM groceries AS g
    -> LEFT JOIN sales AS s1 ON g.id = s1.grocery_id;
```

id	name	brand	tags	nutri_infor	category	packaging_type	date_manufactured	expirary_date	price	id	quantity	purchase_history	revenue	birth_date	start_date	customer_id	staff_id	grocery_id
1	Organic Avocados	GuacGuys	organic, fresh, healthy	{"calories":160,"fat":"15g","carbs":"9g"}	Produce	Bag	2024-08-01	2024-09-15	4.99	1	5	Groceries	100	1990-01-15	2023-08-01	2	1	1
2	Almond Milk	Kirkland	vegan, dairy-free	{"calories":30,"fat":"2.5g","carbs":"1g"}	Dairy	Carton	2024-07-01	2025-07-01	2.49	2	2	Household Items	200	1985-05-20	2023-08-02	3	2	2
3	Cage-Free Eggs	Utopihen	protein, cage-free	{"calories":70,"fat":"5g","protein":"6g"}	Dairy	Carton	2024-09-01	2024-11-01	3.29	3	3	Beverages	75	1992-11-12	2023-08-03	1	3	3
4	Whole Wheat Bread	Nature's	whole grain, fiber	{"calories":90,"fat":"1g","fiber":"4g"}	Bakery	Loaf	2024-10-01	2024-11-01	2.99	4	1	Frozen Foods	90	1988-07-07	2023-08-04	6	4	4
5	Peanut Butter	Skippy	protein, spread	{"calories":190,"fat":"16g","protein":"7g"}	Pantry	Jar	2024-03-01	2025-03-01	2.59	5	4	Personal Care	120	1975-02-28	2023-08-05	4	5	5
6	Honey Nut Cereal	Cheerios	honey, nuts, crunchy	{"calories":110,"fat":"2g","sugar":"9g"}	Breakfast	Box	2024-06-01	2025-06-01	3.79	6	10	Snacks	50	2000-03-17	2023-08-06	5	6	6
7	Greek Yogurt	Fage	protein, probiotic	{"calories":100,"protein":"10g","sugar":"7g"}	Dairy	Cup	2024-08-15	2024-09-30	1.29	7	7	Produce	130	1995-09-25	2023-08-07	7	7	7
8	Boneless Chicken Breasts	Tyson	protein, lean	{"calories":165,"protein":"31g","fat":"3.6g"}	Meat	Pack	2024-10-10	2024-10-20	8.99	8	6	Bakery Items	85	1982-04-30	2023-08-08	8	8	8
9	Frozen Green Beans	C&W	vegan, frozen, healthy	{"calories":30,"fiber":"4g","protein":"2g"}	Frozen	Bag	2024-08-05	2025-08-05	2.49	9	9	Canned Goods	110	1993-06-21	2023-08-09	9	9	9
10	Organic Quinoa	Vival Natural	organic, gluten-free	{"calories":120,"protein":"4g","fiber":"3g"}	Pantry	Box	2024-05-15	2025-05-15	5.29	10	8	Dairy Products	75	1987-08-13	2023-08-10	12	10	10
11	Orange Juice	Uncle Matt	vitamin C, refreshing	{"calories":110,"sugar":"22g","vitamin_c":"100%"}	Beverages	Bottle	2024-09-15	2024-12-15	3.99	11	12	Condiments	140	1999-10-03	2023-08-11	11	11	11
12	Sharp Cheddar Cheese	Kerrygold	cheese, dairy	{"calories":120,"fat":"10g","protein":"7g"}	Dairy	Block	2024-07-20	2024-12-20	4.49	12	5	Seafood	95	1978-12-19	2023-08-12	10	12	12

```
12 rows in set (0.00 sec)
```

c. Write a query to find records that do not have matching entries in another table using a LEFT JOIN combined with a WHERE clause (e.g., finding customers who haven't placed any orders).

- Display all records from the Sales table, joining it with the Customers table, and filter to show only those customers who have not made any purchases at the store yet

```
mysql> SELECT s.*
    -> FROM sales AS s
    -> LEFT JOIN customers AS c ON c.id = s.customer_id
    -> WHERE s.purchase_history IS NULL;
```

id	quantity	purchase_history	revenue	birth_date	start_date	customer_id	staff_id	grocery_id
13	5	NULL	100	1990-01-15	2023-08-01	2	3	9

```
1 row in set (0.00 sec)
```

The records show that the customer with the customer_id '13' has not purchased anything from the HEB store yet.

Advanced Queries

a. Write a query using the UNION operation to combine the results of two different SELECT statements.

```
mysql> SELECT fname, lname
    -> FROM staff
    -> UNION
    -> SELECT fname, lname
    -> FROM customers;
+---------+-----------+
| fname   | lname     |
+---------+-----------+
| John    | Doe       |
| Emma    | Smith     |
| Michael | Johnson   |
| Emily   | Brown     |
| Daniel  | Davis     |
| Sophia  | Wilson    |
| James   | Martinez  |
| Olivia  | Anderson  |
| Liam    | Garcia    |
| Ava     | Lee       |
| Noah    | Hernandez |
| Mia     | Rodriguez |
| Jane    | Smith     |
| Alice   | Johnson   |
| Robert  | Brown     |
| Emily   | Davis     |
| Michael | Wilson    |
| Sarah   | Miller    |
| David   | Garcia    |
| Laura   | Martinez  |
| James   | Rodriguez |
| Sophia  | Martinez  |
| Carlos  | Hernandez |
+---------+-----------+
23 rows in set (0.00 sec)
```

b. Write a query using the INTERSECT operation to find common data that appears in both tables from two different SELECT statements.

```
SELECT fname, lname
FROM staff
INTERSECT
SELECT fname, lname
FROM customers;
```

```
mysql> SELECT fname, lname
    -> FROM staff
    -> INTERSECT
    -> SELECT fname, lname
    -> FROM customers;
+-------+-------+
| fname | lname |
+-------+-------+
| John  | Doe   |
+-------+-------+
1 row in set (0.00 sec)
```

c. Write a query that uses a JOIN and an aggregate function together to display summarized data (e.g., the total number of orders per customer).

- Display the total number of customers who purchased the products from the grocery store, and the total revenue the grocery store made, and the total number of customers

```
SELECT COUNT(s.purchase_history) AS total_customer_purchased,
        SUM(s.revenue) AS total_sales,
        DISTINCT COUNT(c.fname) AS total_customer_num
FROM sales AS s
INNER JOIN customers AS c ON c.id = s.customer_id;
```

```
mysql> SELECT COUNT(s.purchase_history) AS total_customer_purchased,
    -> SUM(s.revenue) AS total_sales,
    -> COUNT(DISTINCT c.fname) AS total_customer_num
    -> FROM sales AS s
    -> INNER JOIN customers AS c ON c.id = s.customer_id;
+--------------------------+-------------+--------------------+
| total_customer_purchased | total_sales | total_customer_num |
+--------------------------+-------------+--------------------+
|                       12 |        1370 |                 12 |
+--------------------------+-------------+--------------------+
1 row in set (0.00 sec)
```

Data Modification

a. Write an UPDATE statement to modify data in one of your tables and describe the impact of the update.

- Original Orders table

```
mysql> SELECT * from orders;
+----+------------+----------------+-------------+--------+------------------+-----------------------------+-------------+
| id | status     | payment_method | ordered_date | cost   | delivery_method  | shipping_address            | customer_id |
+----+------------+----------------+-------------+--------+------------------+-----------------------------+-------------+
|  1 | Shipped    | Credit Card    | 2024-10-01  |  89.99 | Standard Shipping | 123 Main St, Austin, TX     |           1 |
|  2 | Delivered  | PayPal         | 2024-09-15  |  45.50 | Express Delivery  | 456 Oak St, Houston, TX     |           2 |
|  3 | Processing | Credit Card    | 2024-10-12  | 150.25 | Standard Shipping | 789 Pine St, Dallas, TX     |           3 |
|  4 | Delivered  | Debit Card     | 2024-09-22  |  65.75 | Pickup            | 101 Cedar St, San Antonio, TX |           4 |
|  5 | Shipped    | Credit Card    | 2024-10-05  | 120.99 | Express Delivery  | 202 Elm St, Austin, TX      |           5 |
|  6 | Processing | PayPal         | 2024-10-08  | 200.00 | Standard Shipping | 303 Maple St, Houston, TX   |           6 |
|  7 | Delivered  | Credit Card    | 2024-09-30  |  75.49 | Pickup            | 404 Birch St, Dallas, TX    |           7 |
|  8 | Shipped    | Debit Card     | 2024-10-03  |  95.60 | Standard Shipping | 505 Walnut St, San Antonio, TX |           8 |
|  9 | Processing | Credit Card    | 2024-10-10  | 110.00 | Express Delivery  | 606 Cherry St, Austin, TX   |           9 |
| 10 | Delivered  | PayPal         | 2024-09-28  |  58.99 | Pickup            | 707 Ash St, Houston, TX     |          10 |
| 11 | Shipped    | Debit Card     | 2024-10-06  |  90.75 | Standard Shipping | 808 Willow St, Dallas, TX   |          11 |
| 12 | Processing | Credit Card    | 2024-10-09  | 130.50 | Express Delivery  | 909 Poplar St, San Antonio, TX |          12 |
+----+------------+----------------+-------------+--------+------------------+-----------------------------+-------------+
12 rows in set (0.00 sec)
```

- Change the status from ‘Delivered’ to ‘Processing’ and delivery method from ‘Pickup’ to ‘Express Delivery’ for order id ‘7’

```
mysql> UPDATE orders
    -> SET status = 'Processing', delivery_method = 'Express Delivery'
    -> WHERE id = 7;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT *  from orders;
+----+------------+----------------+-------------+--------+------------------+-----------------------------+-------------+
| id | status     | payment_method | ordered_date | cost   | delivery_method  | shipping_address            | customer_id |
+----+------------+----------------+-------------+--------+------------------+-----------------------------+-------------+
|  1 | Shipped    | Credit Card    | 2024-10-01  |  89.99 | Standard Shipping | 123 Main St, Austin, TX     |           1 |
|  2 | Delivered  | PayPal         | 2024-09-15  |  45.50 | Express Delivery  | 456 Oak St, Houston, TX     |           2 |
|  3 | Processing | Credit Card    | 2024-10-12  | 150.25 | Standard Shipping | 789 Pine St, Dallas, TX     |           3 |
|  4 | Delivered  | Debit Card     | 2024-09-22  |  65.75 | Pickup            | 101 Cedar St, San Antonio, TX |           4 |
|  5 | Shipped    | Credit Card    | 2024-10-05  | 120.99 | Express Delivery  | 202 Elm St, Austin, TX      |           5 |
|  6 | Processing | PayPal         | 2024-10-08  | 200.00 | Standard Shipping | 303 Maple St, Houston, TX   |           6 |
|  7 | Processing | Credit Card    | 2024-09-30  |  75.49 | Express Delivery  | 404 Birch St, Dallas, TX    |           7 |
|  8 | Shipped    | Debit Card     | 2024-10-03  |  95.60 | Standard Shipping | 505 Walnut St, San Antonio, TX |           8 |
|  9 | Processing | Credit Card    | 2024-10-10  | 110.00 | Express Delivery  | 606 Cherry St, Austin, TX   |           9 |
| 10 | Delivered  | PayPal         | 2024-09-28  |  58.99 | Pickup            | 707 Ash St, Houston, TX     |          10 |
| 11 | Shipped    | Debit Card     | 2024-10-06  |  90.75 | Standard Shipping | 808 Willow St, Dallas, TX   |          11 |
| 12 | Processing | Credit Card    | 2024-10-09  | 130.50 | Express Delivery  | 909 Poplar St, San Antonio, TX |          12 |
+----+------------+----------------+-------------+--------+------------------+-----------------------------+-------------+
12 rows in set (0.00 sec)
```

b. Write a DELETE statement to remove specific records from a table and explain how to ensure that data integrity is maintained.

- Original suppliers table

```
mysql> SELECT * from suppliers;
+-----+--------------------+------------+--------+-------------------------------+-------------------------------+-----------------------+----------------------+--------------+
| id  | name               | phone_num  | rating | email                         | address                       | products_supplied     | delivery_per_month   | payment_term |
+-----+--------------------+------------+--------+-------------------------------+-------------------------------+-----------------------+----------------------+--------------+
| 101 | Fresh Produce Inc. | 5123456789 |      5 | contact@freshproduce.com      | 123 Farm Rd, Austin, TX       | Fruits & Vegetables   |                    8 |           30 |
| 102 | DairyBest LLC      | 5122345678 |      4 | sales@dairybest.com           | 456 Milk St, Houston, TX      | Dairy Products        |                    6 |           45 |
| 103 | Bakers Delight     | 7139876543 |      5 | info@bakersdelight.com        | 789 Dough St, Dallas, TX      | Bakery Items          |                   10 |           30 |
| 104 | MeatMasters        | 7138765432 |      4 | orders@meatmasters.com        | 101 Steak St, San Antonio, TX | Meat Products         |                    7 |           60 |
| 105 | Grain Goods        | 2107654321 |      3 | support@graingoods.com        | 202 Wheat Blvd, Houston, TX   | Grains & Cereals      |                    5 |           45 |
| 106 | Beverage World     | 2106543210 |      4 | sales@beverageworld.com       | 303 Drink Rd, Austin, TX      | Beverages             |                   12 |           30 |
| 107 | SnackCo Foods      | 5125432109 |      4 | contact@snackco.com           | 404 Chips Ave, Dallas, TX     | Snacks                |                    8 |           30 |
| 108 | Organic Farms      | 7134321098 |      5 | orders@organicfarms.com       | 505 Organic Ln, San Antonio, TX | Organic Products    |                    6 |           60 |
| 109 | Frozen Goods Inc.  | 5123210987 |      3 | info@frozengoods.com          | 606 Freeze St, Austin, TX     | Frozen Foods          |                   10 |           45 |
| 110 | Seafood Suppliers  | 7132109876 |      5 | support@seafoodsuppliers.com  | 707 Ocean Blvd, Houston, TX   | Seafood               |                    4 |           30 |
| 111 | Healthy Oils Ltd.  | 2109876543 |      4 | sales@healthyoils.com         | 808 Oil St, Dallas, TX        | Cooking Oils          |                    9 |           30 |
| 112 | NutriEssentials    | 2108765432 |      4 | contact@nutriessentials.com   | 909 Vital Rd, San Antonio, TX | Nutritional Supplements |                  7 |           45 |
+-----+--------------------+------------+--------+-------------------------------+-------------------------------+-----------------------+----------------------+--------------+
12 rows in set (0.00 sec)
```

- Delete the suppliers that have low rating like ‘3’ from the table

```
mysql> DELETE FROM suppliers
    -> WHERE rating < 4;
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails (`groceries`.`inventory`, CONSTRAINT `inventory_ibfk_1` FOREIGN KEY (`supplier_id`) REFERENCES `suppliers` (`id`))
mysql> ALTER TABLE inventory
    -> DROP FOREIGN KEY inventory_ibfk_1;
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> DELETE FROM suppliers
    -> WHERE rating < 4;
Query OK, 2 rows affected (0.01 sec)

mysql> SELECT * from suppliers;
+-----+--------------------+------------+--------+-------------------------------+-------------------------------+-----------------------+----------------------+--------------+
| id  | name               | phone_num  | rating | email                         | address                       | products_supplied     | delivery_per_month   | payment_term |
+-----+--------------------+------------+--------+-------------------------------+-------------------------------+-----------------------+----------------------+--------------+
| 101 | Fresh Produce Inc. | 5123456789 |      5 | contact@freshproduce.com      | 123 Farm Rd, Austin, TX       | Fruits & Vegetables   |                    8 |           30 |
| 102 | DairyBest LLC      | 5122345678 |      4 | sales@dairybest.com           | 456 Milk St, Houston, TX      | Dairy Products        |                    6 |           45 |
| 103 | Bakers Delight     | 7139876543 |      5 | info@bakersdelight.com        | 789 Dough St, Dallas, TX      | Bakery Items          |                   10 |           30 |
| 104 | MeatMasters        | 7138765432 |      4 | orders@meatmasters.com        | 101 Steak St, San Antonio, TX | Meat Products         |                    7 |           60 |
| 106 | Beverage World     | 2106543210 |      4 | sales@beverageworld.com       | 303 Drink Rd, Austin, TX      | Beverages             |                   12 |           30 |
| 107 | SnackCo Foods      | 5125432109 |      4 | contact@snackco.com           | 404 Chips Ave, Dallas, TX     | Snacks                |                    8 |           30 |
| 108 | Organic Farms      | 7134321098 |      5 | orders@organicfarms.com       | 505 Organic Ln, San Antonio, TX | Organic Products    |                    6 |           60 |
| 110 | Seafood Suppliers  | 7132109876 |      5 | support@seafoodsuppliers.com  | 707 Ocean Blvd, Houston, TX   | Seafood               |                    4 |           30 |
| 111 | Healthy Oils Ltd.  | 2109876543 |      4 | sales@healthyoils.com         | 808 Oil St, Dallas, TX        | Cooking Oils          |                    9 |           30 |
| 112 | NutriEssentials    | 2108765432 |      4 | contact@nutriessentials.com   | 909 Vital Rd, San Antonio, TX | Nutritional Supplements |                  7 |           45 |
+-----+--------------------+------------+--------+-------------------------------+-------------------------------+-----------------------+----------------------+--------------+
10 rows in set (0.00 sec)
```

To ensure data integrity, before executing a delete statement, it is always good to back up data in case there is any unintentional delete operation performed. Other than that, if there is a foreign key reference to the table, ensure that the constraints are respected. In this case, since the inventory table has a foreign key for supplier_id, the operation to drop the existing foreign key could be used to alter table inventory.

Student Table:

```
mysql> SELECT *
    -> FROM student;
+----+----------+---------+-------------------------+--------------+-----+---------------+-------------+---------------------+-----------------------+-------------------------+-----------------+
| id | fname    | lname   | email                   | phone_number | age | date_of_birth | year_joined | year_classification | date_joined_university | major                   | minor           |
+----+----------+---------+-------------------------+--------------+-----+---------------+-------------+---------------------+-----------------------+-------------------------+-----------------+
|  1 | Alice    | Johnson | alice.johnson@example.com |   1234567890 |  20 | 2004-06-15    |        2023 | Sophomore           | 2023-08-20            | Computer Science        | Mathematics     |
|  2 | Bob      | Smith   | bob.smith@example.com   |   1234567891 |  22 | 2002-12-05    |        2021 | Senior              | 2021-08-20            | Mechanical Engineering  | Physics         |
|  3 | Catherine| Green   | catherine.green@example.com |   1234567892 |  19 | 2005-03-12    |        2023 | Freshman            | 2023-08-22            | Data Science            | Economics       |
|  4 | David    | Brown   | david.brown@example.com |   1234567893 |  21 | 2003-10-25    |        2022 | Junior              | 2022-08-19            | Electrical Engineering  | Computer Science |
|  5 | Emma     | White   | emma.white@example.com  |   1234567894 |  23 | 2000-01-14    |        2020 | Senior              | 2020-08-18            | Biology                 | Chemistry       |
|  6 | Frank    | Blue    | frank.blue@example.com  |   1234567895 |  24 | 1999-09-05    |        2019 | Graduate            | 2019-08-21            | Physics                 | Mathematics     |
|  7 | Grace    | Red     | grace.red@example.com   |   1234567896 |  18 | 2006-04-20    |        2024 | Freshman            | 2024-08-23            | Economics               | Statistics      |
|  8 | Hannah   | Yellow  | hannah.yellow@example.com |   1234567897 |  22 | 2002-07-18    |        2021 | Senior              | 2021-08-21            | Psychology              | Sociology       |
|  9 | Ian      | Gray    | ian.gray@example.com    |   1234567898 |  20 | 2004-02-10    |        2022 | Sophomore           | 2022-08-20            | Data Science            | Computer Science |
| 10 | Jack     | Orange  | jack.orange@example.com |   1234567899 |  21 | 2003-11-30    |        2022 | Junior              | 2022-08-21            | Electrical Engineering  | Robotics        |
| 11 | Karen    | Violet  | karen.violet@example.com |   1234567800 |  23 | 2000-05-25    |        2020 | Senior              | 2020-08-22            | Medicine                | Public Health   |
| 12 | Liam     | Indigo  | liam.indigo@example.com |   1234567801 |  19 | 2005-01-03    |        2023 | Sophomore           | 2023-08-20            | Software Engineering    | Mathematics     |
| 13 | Mia      | Teal    | mia.teal@example.com    |   1234567802 |  20 | 2004-12-10    |        2022 | Sophomore           | 2022-08-19            | Environmental Science   | Geology         |
| 14 | Nathan   | Maroon  | nathan.maroon@example.com |   1234567803 |  24 | 1999-03-15    |        2019 | Graduate            | 2019-08-21            | Mechanical Engineering  | Business        |
| 15 | Olivia   | Pink    | olivia.pink@example.com |   1234567804 |  21 | 2003-09-29    |        2021 | Junior              | 2021-08-22            | Biotechnology           | Chemistry       |
| 16 | Peter    | Black   | peter.black@example.com |   1234567805 |  25 | 1998-06-18    |        2018 | Graduate            | 2018-08-20            | Physics                 | Astronomy       |
| 17 | Quinn    | White   | quinn.white@example.com |   1234567806 |  20 | 2004-08-12    |        2023 | Sophomore           | 2023-08-23            | Artificial Intelligence | Statistics      |
+----+----------+---------+-------------------------+--------------+-----+---------------+-------------+---------------------+-----------------------+-------------------------+-----------------+
17 rows in set (0.00 sec)
```

Normalize Old Tables

Normalization is performed to ensure data integrity and successful database design. In this case, we chose the Student table. For starters, 1st NF ensures there is no redundancy in the values. Since the student table already has atomic values and each record is unique, it is 1NF.

Next, 2nd NF requires all non-key attributes to be partially dependent on the primary key, and there are no foreign keys in the table, each non-key attribute is fully dependent on the primary key.

Then, 3rd NF ensures no transitive dependencies, meaning that all key-attributes have to depend on the primary key. However, as shown in the last section, the functional dependencies in the Student table also include age as the subset of date_of_birth, and year_classification as the subset of year_joined. Therefore, we have to create new tables to break the transitive dependencies to achieve 3NF.

Creating student_age Table:

```
mysql> CREATE TABLE student_age (
    -> student_id INT NOT NULL,
    -> age INT CHECK (age > 0),
    -> PRIMARY KEY (student_id),
    -> FOREIGN KEY (student_id) REFERENCES student(id)
    -> );
Query OK, 0 rows affected (0.09 sec)

mysql> SELECT *
    -> FROM student_age;
+------------+------+
| student_id | age  |
+------------+------+
|          1 |   20 |
|          2 |   22 |
|          3 |   19 |
|          4 |   21 |
|          5 |   23 |
|          6 |   24 |
|          7 |   18 |
|          8 |   22 |
|          9 |   20 |
|         10 |   21 |
|         11 |   23 |
|         12 |   19 |
|         13 |   20 |
|         14 |   24 |
|         15 |   21 |
|         16 |   25 |
|         17 |   20 |
+------------+------+
17 rows in set (0.00 sec)
```

Creating student_year Table:

```
mysql> CREATE TABLE student_year(
    -> student_id INT NOT NULL,
    -> year_joined INT CHECK (year_joined > 2000),
    -> year_classification VARCHAR(50),
    -> PRIMARY KEY (student_id),
    -> FOREIGN KEY (student_id) REFERENCES student(id)
    -> );
Query OK, 0 rows affected (0.06 sec)
```

```
mysql> SELECT *
    -> FROM student_year;
+------------+-------------+----------------------+
| student_id | year_joined | year_classification |
+------------+-------------+----------------------+
|          1 |        2023 | Sophomore            |
|          2 |        2021 | Senior               |
|          3 |        2023 | Freshman             |
|          4 |        2022 | Sophomore            |
|          5 |        2020 | Graduate             |
|          6 |        2019 | Graduate             |
|          7 |        2024 | Freshman             |
|          8 |        2021 | Senior               |
|          9 |        2022 | Sophomore            |
|         10 |        2022 | Junior               |
|         11 |        2020 | Senior               |
|         12 |        2023 | Sophomore            |
|         13 |        2022 | Sophomore            |
|         14 |        2019 | Graduate             |
|         15 |        2021 | Junior               |
|         16 |        2018 | Graduate             |
|         17 |        2023 | Sophomore            |
+------------+-------------+----------------------+
17 rows in set (0.00 sec)
```

Final Students Table:

```
mysql> SELECT *
    -> FROM students;
+----+----------+---------+---------------------------+-------------+-----------------------+-------------------------+------------------+
| id | fname    | lname   | email                     | phone_number | date_joined_university | major                   | minor            |
+----+----------+---------+---------------------------+-------------+-----------------------+-------------------------+------------------+
|  1 | Alice    | Johnson | alice.johnson@example.com |   1234567890 | 2023-08-20            | Computer Science        | Mathematics      |
|  2 | Bob      | Smith   | bob.smith@example.com     |   1234567891 | 2021-08-20            | Mechanical Engineering  | Physics          |
|  3 | Catherine | Green  | catherine.green@example.com |  1234567892 | 2023-08-22            | Data Science            | Economics        |
|  4 | David    | Brown   | david.brown@example.com   |   1234567893 | 2022-08-19            | Electrical Engineering  | Computer Science |
|  5 | Emma     | White   | emma.white@example.com    |   1234567894 | 2020-08-18            | Biology                 | Chemistry        |
|  6 | Frank    | Blue    | frank.blue@example.com    |   1234567895 | 2019-08-21            | Physics                 | Mathematics      |
|  7 | Grace    | Red     | grace.red@example.com     |   1234567896 | 2024-08-23            | Economics               | Statistics       |
|  8 | Hannah   | Yellow  | hannah.yellow@example.com |   1234567897 | 2021-08-21            | Psychology              | Sociology        |
|  9 | Ian      | Gray    | ian.gray@example.com      |   1234567898 | 2022-08-20            | Data Science            | Computer Science |
| 10 | Jack     | Orange  | jack.orange@example.com   |   1234567899 | 2022-08-21            | Electrical Engineering  | Robotics         |
| 11 | Karen    | Violet  | karen.violet@example.com  |   1234567800 | 2020-08-22            | Medicine                | Public Health    |
| 12 | Liam     | Indigo  | liam.indigo@example.com   |   1234567801 | 2023-08-20            | Software Engineering    | Mathematics      |
| 13 | Mia      | Teal    | mia.teal@example.com      |   1234567802 | 2022-08-19            | Environmental Science   | Geology          |
| 14 | Nathan   | Maroon  | nathan.maroon@example.com |   1234567803 | 2019-08-21            | Mechanical Engineering  | Business         |
| 15 | Olivia   | Pink    | olivia.pink@example.com   |   1234567804 | 2021-08-22            | Biotechnology           | Chemistry        |
| 16 | Peter    | Black   | peter.black@example.com   |   1234567805 | 2018-08-20            | Physics                 | Astronomy        |
| 17 | Quinn    | White   | quinn.white@example.com   |   1234567806 | 2023-08-23            | Artificial Intelligence | Statistics       |
+----+----------+---------+---------------------------+-------------+-----------------------+-------------------------+------------------+
17 rows in set (0.00 sec)
```

Final student_age Table:

```
mysql> SELECT *
    -> FROM student_age;
+------------+------+
| student_id | age  |
+------------+------+
|          1 |   20 |
|          2 |   22 |
|          3 |   19 |
|          4 |   21 |
|          5 |   23 |
|          6 |   24 |
|          7 |   18 |
|          8 |   22 |
|          9 |   20 |
|         10 |   21 |
|         11 |   23 |
|         12 |   19 |
|         13 |   20 |
|         14 |   24 |
|         15 |   21 |
|         16 |   25 |
|         17 |   20 |
+------------+------+
17 rows in set (0.00 sec)
```

Final student_year Table:

```
mysql> SELECT *
    -> FROM student_year;
+------------+-------------+--------------------+
| student_id | year_joined | year_classification |
+------------+-------------+--------------------+
|          1 |        2023 | Sophomore          |
|          2 |        2021 | Senior             |
|          3 |        2023 | Freshman           |
|          4 |        2022 | Sophomore          |
|          5 |        2020 | Graduate           |
|          6 |        2019 | Graduate           |
|          7 |        2024 | Freshman           |
|          8 |        2021 | Senior             |
|          9 |        2022 | Sophomore          |
|         10 |        2022 | Junior             |
|         11 |        2020 | Senior             |
|         12 |        2023 | Sophomore          |
|         13 |        2022 | Sophomore          |
|         14 |        2019 | Graduate           |
|         15 |        2021 | Junior             |
|         16 |        2018 | Graduate           |
|         17 |        2023 | Sophomore          |
+------------+-------------+--------------------+
17 rows in set (0.00 sec)
```

Simple Queries

Simple Queries Figure 1

```
mysql> SELECT feedback_date, student_id
    -> FROM feedback
    -> WHERE urgency_level = 'High';
+---------------+------------+
| feedback_date | student_id |
+---------------+------------+
| 2024-09-10    |          1 |
| 2024-09-16    |          4 |
| 2024-09-22    |          7 |
| 2024-09-28    |         10 |
| 2024-10-04    |         13 |
| 2024-10-10    |         16 |
+---------------+------------+
6 rows in set (0.00 sec)
```

This query assists the professor in identifying students who require urgent feedback, along with the feedback date.

Simple Queries Figure 2

```
mysql> SELECT DISTINCT student_id
    -> FROM exams
    -> WHERE course_id = 11
    -> ;
+------------+
| student_id |
+------------+
|         12 |
+------------+
1 row in set (0.00 sec)
```

This query displays all Student IDs who have given an exam for course_id = 11

Simple Queries Figure 3

```
mysql> SELECT name, email, office_hours
    -> FROM teaching_assistant
    -> WHERE academic_background LIKE 'MSc%'
    -> AND section = 'C';
+-------------+-------------------------+-----------------+
| name        | email                   | office_hours    |
+-------------+-------------------------+-----------------+
| David Black | david.black@example.com | Mon 9:00-11:00  |
| Jack Pink   | jack.pink@example.com   | Mon 10:00-12:00 |
| Mia Teal    | mia.teal@example.com    | Thu 10:00-12:00 |
| Peter Black | peter.black@example.com | Tue 9:00-11:00  |
+-------------+-------------------------+-----------------+
4 rows in set (0.00 sec)
```

This query identifies teaching assistants that have a masters degree and are in section C. It shows their name, email, and office hours.

Simple Queries Figure 4

```
mysql> SELECT id AS E_ID, email, contact, biography
    -> FROM profile
    -> WHERE (role = 'TA'
    ->     AND interests LIKE '%AI%');
+------+-----------------+--------------+-----------------------------------------------------------------------+
| E_ID | email           | contact      | biography                                                             |
+------+-----------------+--------------+-----------------------------------------------------------------------+
|    6 | ta1@example.com | 345-555-8901 | Teaching assistant with a focus on neural networks and robotics. |
+------+-----------------+--------------+-----------------------------------------------------------------------+
1 row in set (0.00 sec)
```

This query Identifies teaching assistant with an interest in AI.

Simple Queries Figure 5

```
mysql> SELECT COUNT(*) AS total_num_of_rows
    -> FROM attendance_records;
+--------------------+
| total_num_of_rows |
+--------------------+
|                 17 |
+--------------------+
1 row in set (0.01 sec)
```

This is a simple query that showcases the row count for attendance_records. It can be applied to other tables as well to identify their row count.

Complex Queries

Complex Queries Figure 1

```
mysql> SELECT attendance_records.course_id,
    ->        course.name,
    ->        COUNT(attendance_records.attendance_status) AS attendance_count
    -> FROM attendance_records
    -> INNER JOIN course ON course.course_id = attendance_records.course_id
    -> GROUP BY attendance_records.course_id, course.name;
+-----------+----------------------------+------------------+
| course_id | name                       | attendance_count |
+-----------+----------------------------+------------------+
|       102 | Data Structures            |                5 |
|       103 | Algorithms Basics          |                1 |
|       104 | Database Design            |                1 |
|       105 | Web Development            |                1 |
|       106 | Data Analysis              |                1 |
|       107 | Machine Learning Basics    |                1 |
|       108 | Statistics for Data Science |                1 |
|       109 | Advanced Programming       |                1 |
|       110 | Project Management         |                1 |
|       111 | Cloud Computing            |                1 |
|       112 | Cybersecurity Basics       |                1 |
+-----------+----------------------------+------------------+
11 rows in set (0.00 sec)
```

This query displays the course ID and course name through attendance records.

Complex Queries Figure 2

```
mysql> SELECT s.fname, s.lname, a.attendance_percentage, a.reason_for_absence
    -> FROM attendance_records AS a
    -> JOIN student AS s ON a.student_id = s.id
    -> WHERE a.reason_for_absence = 'Work Commitment';
+-------+--------+----------------------+--------------------+
| fname | lname  | attendance_percentage | reason_for_absence |
+-------+--------+----------------------+--------------------+
| Liam  | Indigo |                   16 | Work Commitment    |
+-------+--------+----------------------+--------------------+
1 row in set (0.00 sec)
```

This Query check which student fail to attend class because of work commitment

Complex Queries Figure 3

```
mysql> SELECT s.fname, s.lname, s.major, s.year_classification, g.letter_grade
    -> FROM grades AS g JOIN student AS s ON g.student_id = s.id
    -> WHERE g.letter_grade LIKE 'A%'
    -> LIMIT 5;
+--------+--------+------------------------+--------------------+--------------+
| fname  | lname  | major                  | year_classification | letter_grade |
+--------+--------+------------------------+--------------------+--------------+
| Liam   | Indigo | Software Engineering   | Sophomore          | A            |
| Olivia | Pink   | Biotechnology          | Junior             | A-           |
| Quinn  | White  | Artificial Intelligence | Sophomore         | A            |
| David  | Brown  | Electrical Engineering | Junior             | A            |
| Jack   | Orange | Electrical Engineering | Junior             | A-           |
+--------+--------+------------------------+--------------------+--------------+
5 rows in set (0.01 sec)
```

This query Identifies the top 5 students with a grade of A or A-, and displays their first and last names, major, and year classifications.

Complex Queries Figure 4

```
mysql> SELECT c.name AS course_name, AVG(a.attendance_percentage) AS average_attendance_percentage
    -> FROM attendance_records AS a
    -> JOIN course AS c ON a.course_id = c.id
    -> WHERE a.year = 2024
    -> GROUP BY c.id;
+----------------------------+-------------------------------+
| course_name                | average_attendance_percentage |
+----------------------------+-------------------------------+
| Introduction to Programming |                        8.0000 |
| Data Structures            |                       17.0000 |
| Machine Learning           |                       14.0000 |
| Database Management Systems |                        3.0000 |
| Operating Systems          |                        2.0000 |
| Computer Networks          |                        1.0000 |
| Artificial Intelligence    |                       11.0000 |
| Cybersecurity Fundamentals |                       13.0000 |
| Cloud Computing            |                        4.0000 |
| Data Science Fundamentals  |                       95.0000 |
| Ethics in AI               |                        5.0000 |
| Advanced Statistics        |                       16.0000 |
| Robotics                   |                       15.0000 |
| Software Engineering       |                        6.0000 |
| Big Data Analytics         |                        9.0000 |
| Natural Language Processing |                        7.0000 |
| Environmental Data Science |                       10.0000 |
+----------------------------+-------------------------------+
```

This query finds the average attendance percentage of students for each course in 2024.

Complex Queries Figure 5

```
mysql> SELECT s.fname, s.lname, c.name AS course_name,
    ->        COUNT(a.id) AS assignment_count,
    ->        SUM(CASE WHEN a.extra_credit = TRUE THEN 1 ELSE 0 END) AS extra_credit_assignments
    -> FROM student AS s
    -> JOIN assignment AS a ON s.id = a.student_id
    -> JOIN course AS c ON a.course_id = c.id
    -> GROUP BY s.id, c.id
    -> HAVING COUNT(a.id) = 0 OR SUM(CASE WHEN a.extra_credit = TRUE THEN 1 ELSE 0 END) > 0;
+--------+--------+----------------------------+------------------+--------------------------+
| fname  | lname  | course_name                | assignment_count | extra_credit_assignments |
+--------+--------+----------------------------+------------------+--------------------------+
| David  | Brown  | Database Management Systems |                1 |                        1 |
| Grace  | Red    | Artificial Intelligence    |                1 |                        1 |
| Jack   | Orange | Data Science Fundamentals  |                1 |                        1 |
| Mia    | Teal   | Robotics                   |                1 |                        1 |
| Olivia | Pink   | Big Data Analytics         |                1 |                        1 |
+--------+--------+----------------------------+------------------+--------------------------+
5 rows set (0.01 sec)
```

This query finds the students who have not completed any assignments or have an extra credit assignment.

Complex Queries Figure 6

```
mysql> SELECT i.id, i.name, i.department, AVG(i.years_of_experience) AS avg_experience, COUNT(o.id) AS total_office_hours
    -> FROM instructors AS i
    -> JOIN office_hours AS o ON o.instructor_id = i.id
    -> WHERE i.starting_year > 2010
    -> GROUP BY i.id, i.name, i.department
    -> HAVING COUNT(o.id) > 0
    -> ORDER BY avg_experience DESC;
+----+--------------------+----------------------+----------------+--------------------+
| id | name               | department           | avg_experience | total_office_hours |
+----+--------------------+----------------------+----------------+--------------------+
|  3 | Prof. Alice Johnson | Statistics           |        12.0000 |                  1 |
| 10 | Prof. Ivy Yellow    | Sociology            |        12.0000 |                  1 |
|  1 | Dr. John Smith      | Data Science         |        10.0000 |                  1 |
|  7 | Dr. Frank Black     | Blockchain Technology |        10.0000 |                  1 |
| 15 | Dr. Nathan Teal     | AI Ethics            |        10.0000 |                  1 |
| 11 | Dr. Jack Gray       | Data Visualization   |         9.0000 |                  1 |
|  2 | Dr. Jane Doe        | Computer Science     |         8.0000 |                  1 |
| 14 | Dr. Maya Pink       | DevOps               |         8.0000 |                  1 |
|  6 | Dr. Emily White     | Cloud Computing      |         7.0000 |                  1 |
| 13 | Dr. Liam Violet     | Cloud Security       |         7.0000 |                  1 |
|  8 | Dr. Grace Blue      | Cybersecurity        |         6.0000 |                  1 |
|  5 | Prof. Carol Green   | Big Data             |         5.0000 |                  1 |
| 17 | Dr. Peter Maroon    | Machine Learning     |         5.0000 |                  1 |
+----+--------------------+----------------------+----------------+--------------------+
13 rows in set (0.01 sec)
```

This query checks the grades in the canvas system.

Aggregate Functions and Subqueries

Aggregate Functions and Subqueries Figure 1

```
mysql> SELECT COUNT(DISTINCT student_id) AS Number_of_students_that_are_present
FROM attendance_records
    -> WHERE attendance_status = 'Present';
+-------------------------------------+
| Number_of_students_that_are_present |
+-------------------------------------+
|                                   7 |
+-------------------------------------+
```

This figure counts the distinct student IDs in the attendance records table where the attendance status is marked as 'Present'.

Aggregate Functions and Subqueries Figure 2

```
mysql> SELECT i.name AS instructor_name, m.name AS module_name, AVG(m.module_du
ation) AS avg_module_duration
    -> FROM instructors AS i
    -> JOIN modules AS m ON i.id = m.instructor_id
    -> WHERE i.years_of_experience > 10
    -> GROUP BY i.id, i.name, m.name
    -> HAVING AVG(m.module_duration) > 7
    -> ORDER by avg_module_duration DESC;
+-----------------+-----------------+---------------------+
| instructor_name | module_name     | avg_module_duration |
+-----------------+-----------------+---------------------+
| Dr. Henry Red   | Cybersecurity   |              9.0000 |
| Dr. Karen Orange | Machine Learning |              8.0000 |
+-----------------+-----------------+---------------------+
2 rows in set (0.02 sec)
```

This figure identifies instructors with at least 10 years of teaching experience and an average module duration exceeding 7 hours. It displays and ranks them in descending order based on their average module duration.

Aggregate Functions and Subqueries Figure 3

```
mysql> SELECT * FROM assignment
    -> WHERE assignment.student_id IN (SELECT DISTINCT exams.student_id
    -> FROM exams
    -> WHERE exams.course_id = 101);
+---------------+------------+-----------+-----------------------------------+-------------------------------+-----------------+-------------+------------+---------------+----------------------+
| assignment_id | student_id | course_id | assignment_question               | assignment_answer             | grade_weightage | extra_credit | date_due   | date_assigned | assignment_attempts |
+---------------+------------+-----------+-----------------------------------+-------------------------------+-----------------+-------------+------------+---------------+----------------------+
|             1 |        301 |       101 | Write a Python program for Hello World. | print("Hello World")          |              10 |           0 | 2024-01-15 | 2024-01-10    |                    1 |
|             2 |        302 |       102 | Implement a linked list in Java.  | LinkedList implementation in Java |              15 |           0 | 2024-02-01 | 2024-01-20    |                    2 |
+---------------+------------+-----------+-----------------------------------+-------------------------------+-----------------+-------------+------------+---------------+----------------------+
2 rows in set (0.00 sec)
```

In this figure, the subquery takes all of the assignments completed for all students who have taken an exam in a specific course. More specifically, this identifies the assignments a student has done if they completed an exam for course_id = 101.

Aggregate Functions and Subqueries Figure 4

```
mysql> SELECT s.id, s.fname, s.lname, s.year_classification, s.major, COUNT(a.id) AS total_assignments, AVG(g.percentage_grade) AS avg_percentage_grade
    -> FROM student AS s
    -> JOIN grades AS g ON g.student_id = s.id
    -> JOIN assignment AS a ON a.student_id = s.id
    -> GROUP BY s.id
    -> HAVING AVG(g.percentage_grade) IS NOT NULL
    -> ORDER BY avg_percentage_grade DESC;
+----+-----------+---------+---------------------+-------------------------+-------------------+----------------------+
| id | fname     | lname   | year_classification | major                   | total_assignments | avg_percentage_grade |
+----+-----------+---------+---------------------+-------------------------+-------------------+----------------------+
|  4 | David     | Brown   | Junior              | Electrical Engineering  |                 1 |              95.0000 |
|  9 | Ian       | Gray    | Sophomore           | Data Science            |                 1 |              94.0000 |
| 17 | Quinn     | White   | Sophomore           | Artificial Intelligence |                 1 |              93.0000 |
| 12 | Liam      | Indigo  | Sophomore           | Software Engineering    |                 1 |              92.0000 |
| 10 | Jack      | Orange  | Junior              | Electrical Engineering  |                 1 |              91.0000 |
| 15 | Olivia    | Pink    | Junior              | Biotechnology           |                 1 |              90.0000 |
|  3 | Catherine | Green   | Freshman            | Data Science            |                 1 |              89.0000 |
|  8 | Hannah    | Yellow  | Senior              | Psychology              |                 1 |              88.0000 |
| 14 | Nathan    | Maroon  | Graduate            | Mechanical Engineering  |                 1 |              88.0000 |
|  6 | Frank     | Blue    | Graduate            | Physics                 |                 1 |              86.0000 |
| 11 | Karen     | Violet  | Senior              | Medicine                |                 1 |              85.0000 |
|  2 | Bob       | Smith   | Senior              | Mechanical Engineering  |                 1 |              84.0000 |
| 16 | Peter     | Black   | Graduate            | Physics                 |                 1 |              82.0000 |
|  5 | Emma      | White   | Senior              | Biology                 |                 1 |              79.0000 |
|  7 | Grace     | Red     | Freshman            | Economics               |                 1 |              79.0000 |
|  1 | Alice     | Johnson | Sophomore           | Computer Science        |                 1 |              78.0000 |
| 13 | Mia       | Teal    | Sophomore           | Environmental Science   |                 1 |              75.0000 |
+----+-----------+---------+---------------------+-------------------------+-------------------+----------------------+
17 rows in set (0.00 sec)
```

This figure displays academic reports in the canvas system, starting from students with the highest grades.

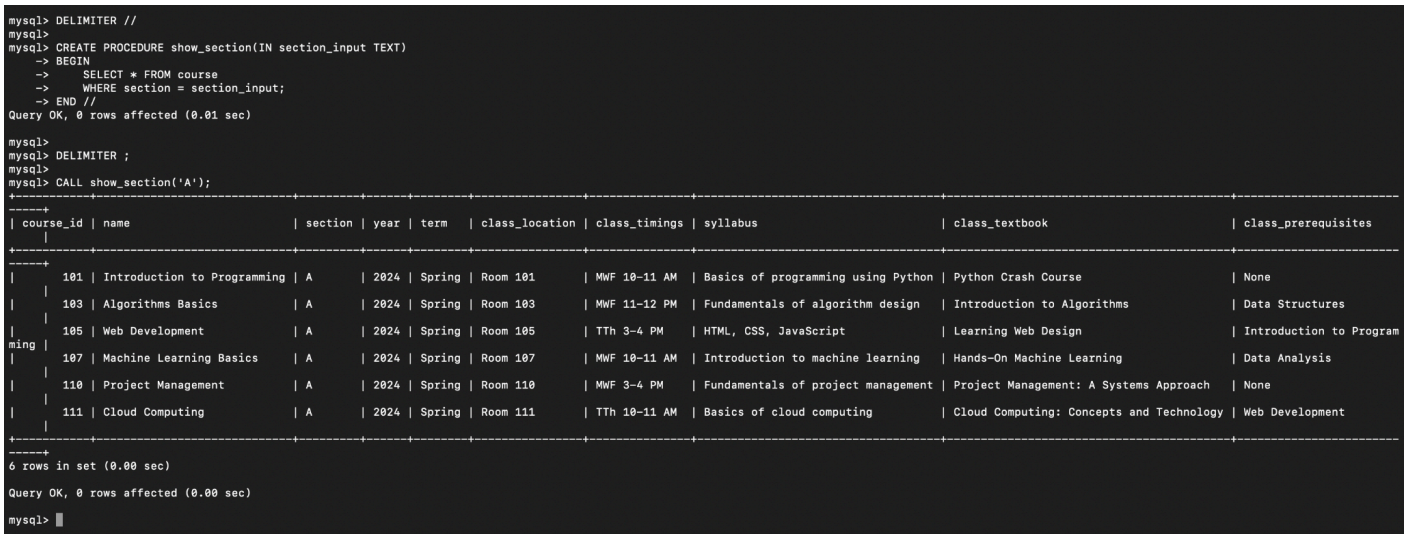
Built-in functions and stored procedures

Built-in functions and stored procedures Figure 1

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE show_course()
    -> BEGIN
    -> SELECT *
    -> FROM course;
    -> END //
ERROR 1304 (42000): PROCEDURE show_course already exists
mysql> DELIMITER ;
mysql> Call show_course();
+----+-----------------------------+---------+------+--------+----------------+---------------------+-------------------------------------------------------------------------+--------------------------------------
-----------+--------------------------------------+
| id | name                        | section | year | term   | class_location | class_timings       | syllabus                                                                | class_textbook
           | class_prerequisites                  |
+----+-----------------------------+---------+------+--------+----------------+---------------------+-------------------------------------------------------------------------+--------------------------------------
-----------+--------------------------------------+
|  1 | Introduction to Programming | A       | 2024 | Fall   | Room 101       | Mon-Wed 10:00-11:30 | Basics of programming, syntax, and algorithms.                          | Programming in Python
           | None                                 |
|  2 | Data Structures             | B       | 2024 | Spring | Room 102       | Tue-Thu 1:00-2:30   | Study of data structures including arrays, lists, stacks, queues.       | Algorithms Unlocked
           | Introduction to Programming          |
|  3 | Machine Learning            | A       | 2024 | Fall   | Room 103       | Mon-Wed 2:00-3:30   | Supervised, unsupervised learning techniques, and real-world applications. | Deep Learning by Goodfellow
           | Data Structures, Linear Algebra      |
|  4 | Database Management Systems | C       | 2024 | Spring | Room 104       | Tue-Thu 3:00-4:30   | Relational databases, SQL, and optimization techniques.                 | Database System Concepts
           | Introduction to Programming          |
|  5 | Operating Systems           | B       | 2024 | Fall   | Room 105       | Mon-Wed 9:00-10:30  | Processes, memory management, file systems, and concurrency.            | Modern Operating Systems
           | Data Structures                      |
|  6 | Computer Networks           | A       | 2024 | Spring | Room 106       | Tue-Thu 11:00-12:30 | Networking fundamentals, protocols, and security.                       | Computer Networking: A Top-Down Appr
oach       | Operating Systems                    |
|  7 | Artificial Intelligence     | C       | 2024 | Fall   | Room 107       | Mon-Wed 4:00-5:30   | Introduction to AI concepts, search algorithms, and reasoning.          | Artificial Intelligence: A Modern Ap
proach     | Data Structures                      |
|  8 | Cybersecurity Fundamentals  | B       | 2024 | Spring | Room 108       | Tue-Thu 2:30-4:00   | Cyber threats, encryption, and secure system design.                    | Cybersecurity Essentials
           | Operating Systems                    |
|  9 | Cloud Computing             | A       | 2024 | Fall   | Room 109       | Mon-Wed 3:00-4:30   | Cloud architecture, virtualization, and storage systems.                | Cloud Computing: Concepts and Techno
logy       | Operating Systems, Networking        |
| 10 | Data Science Fundamentals   | C       | 2024 | Spring | Room 110       | Tue-Thu 9:00-10:30  | Data manipulation, analysis, and visualization techniques.              | Data Science for Beginners
           | Linear Algebra, Programming Basics   |
| 11 | Ethics in AI                | B       | 2024 | Fall   | Room 111       | Mon-Wed 11:00-12:30 | Bias, fairness, and accountability in AI systems.                       | AI Ethics in Practice
           | Artificial Intelligence              |
| 12 | Advanced Statistics         | A       | 2024 | Spring | Room 112       | Tue-Thu 10:00-11:30 | Probability distributions, hypothesis testing, and regression.          | Introduction to Statistical Learning
           | Basic Statistics, Linear Algebra     |
| 13 | Robotics                    | C       | 2024 | Fall   | Room 113       | Mon-Wed 1:00-2:30   | Kinematics, sensors, and autonomous systems.                            | Introduction to Robotics
           | Linear Algebra, Programming Basics   |
| 14 | Software Engineering        | B       | 2024 | Spring | Room 114       | Tue-Thu 4:00-5:30   | Software development lifecycle, Agile methodologies.                    | Software Engineering: Principles and
 Practices | Programming Basics                   |
| 15 | Big Data Analytics          | A       | 2024 | Fall   | Room 115       | Mon-Wed 9:00-10:30  | Big data frameworks, Hadoop, and Spark.                                 | Big Data Analytics for Dummies
           | Database Management Systems          |
| 16 | Natural Language Processing | C       | 2024 | Spring | Room 116       | Tue-Thu 1:00-2:30   | Text processing, sentiment analysis, and machine translation.           | Speech and Language Processing
           | Machine Learning, Programming Basics |
| 17 | Environmental Data Science  | B       | 2024 | Fall   | Room 117       | Mon-Wed 3:00-4:30   | Environmental data analysis and visualization.                          | Environmental Data Analysis
           | Data Science Fundamentals, Statistics |
+----+-----------------------------+---------+------+--------+----------------+---------------------+-------------------------------------------------------------------------+--------------------------------------
```

This figure defines and executes a stored procedure show_course() that retrieves all rows from the course table using SELECT * FROM course;.

Built-in functions and stored procedures Figure 2



```
mysql> DELIMITER //
mysql>
mysql> CREATE PROCEDURE show_section(IN section_input TEXT)
    -> BEGIN
    ->     SELECT * FROM course
    ->     WHERE section = section_input;
    -> END //
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql>
mysql> CALL show_section('A');
+-----------+----------------------------+---------+------+--------+---------------+---------------+--------------------------------+----------------------------------------+-----------------------
-----+
| course_id | name                       | section | year | term   | class_location | class_timings | syllabus                       | class_textbook                         | class_prerequisites
     |
+-----------+----------------------------+---------+------+--------+---------------+---------------+--------------------------------+----------------------------------------+-----------------------
-----+
|       101 | Introduction to Programming | A       | 2024 | Spring | Room 101       | MWF 10-11 AM  | Basics of programming using Python | Python Crash Course                    | None
     |
|       103 | Algorithms Basics           | A       | 2024 | Spring | Room 103       | MWF 11-12 PM  | Fundamentals of algorithm design   | Introduction to Algorithms             | Data Structures
     |
|       105 | Web Development             | A       | 2024 | Spring | Room 105       | TTh 3-4 PM    | HTML, CSS, JavaScript              | Learning Web Design                    | Introduction to Program
ming |
|       107 | Machine Learning Basics     | A       | 2024 | Spring | Room 107       | MWF 10-11 AM  | Introduction to machine learning   | Hands-On Machine Learning              | Data Analysis
     |
|       110 | Project Management          | A       | 2024 | Spring | Room 110       | MWF 3-4 PM    | Fundamentals of project management | Project Management: A Systems Approach | None
     |
|       111 | Cloud Computing             | A       | 2024 | Spring | Room 111       | TTh 10-11 AM  | Basics of cloud computing          | Cloud Computing: Concepts and Technology | Web Development
     |
+-----------+----------------------------+---------+------+--------+---------------+---------------+--------------------------------+----------------------------------------+-----------------------
-----+
6 rows in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql>
```

This figure executes a stored procedure that retrieves all rows from the course table for a specified section. In this example, the section is ‘A’.

Built-in functions and stored procedures Figure 3

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE  add_resources(
    -> IN p_id INT,
    -> IN p_format VARCHAR(50),
    -> IN p_name VARCHAR(50),
    -> IN p_size INT,
    -> IN p_subject_area VARCHAR(50),
    -> IN p_access_links VARCHAR (50),
    -> IN p_source VARCHAR (50)
    -> )
    -> BEGIN
    -> INSERT INTO resources (id, format, name, size, subject_area, access_links, source) VALUES (p_id, p_format, p_name, p_size, p_subject_area, p_access_links, p_source);
    -> END //

mysql> CALL add_resources(
    -> 20,
    -> 'HTML',
    -> 'UX Design',
    -> 800,
    -> 'Web Development',
    -> 'https://blog.uxtweak.com/best-ux-resources/',
    -> 'Google'
    -> );
    -> //
Query OK, 1 row affected (0.01 sec)

mysql> SELECT *
    -> FROM resources;
    -> //
+----+---------+--------------------------------+------+-----------------------+-----------------------------------------+--------------------------+
| id | format  | name                           | size | subject_area          | access_links                            | source                   |
+----+---------+--------------------------------+------+-----------------------+-----------------------------------------+--------------------------+
|  1 | PDF     | Data Engineering Handbook      | 2048 | Data Engineering      | http://example.com/handbook             | O'Reilly                 |
|  2 | Video   | Introduction to Python         | 1500 | Programming           | http://example.com/python_intro         | YouTube                  |
|  3 | Dataset | Water Quality Dataset          |  500 | Environmental Science | http://example.com/water_dataset        | Kaggle                   |
|  4 | Ebook   | Deep Learning with TensorFlow  | 3072 | Artificial Intelligence | http://example.com/tensorflow_book    | Packt                    |
|  5 | HTML    | SQL Tutorial                   | 1024 | Database Management   | http://example.com/sql_tutorial         | W3Schools                |
|  6 | PDF     | Machine Learning Crash Course  | 2048 | Machine Learning      | http://example.com/ml_course            | Google AI                |
|  7 | Dataset | Global Homelessness Data       |  700 | Sociology             | http://example.com/homeless_data        | UN Data                  |
|  8 | Video   | Big Data Basics                | 1200 | Data Science          | http://example.com/big_data_basics      | Coursera                 |
|  9 | PDF     | Cloud Computing Essentials     | 1500 | Cloud Computing       | http://example.com/cloud_essentials     | AWS                      |
| 10 | HTML    | Docker Documentation           |  512 | DevOps                | http://example.com/docker_docs          | Docker                   |
| 11 | PDF     | Cybersecurity Fundamentals     | 1800 | Information Security  | http://example.com/cybersecurity        | NIST                     |
| 12 | Video   | Data Visualization Techniques  |  900 | Data Visualization    | http://example.com/data_viz             | Udemy                    |
| 13 | Dataset | Air Pollution Levels           |  800 | Environmental Science | http://example.com/air_pollution        | NASA                     |
| 14 | Ebook   | Statistics for Data Science    | 2560 | Statistics            | http://example.com/stats_ds             | Springer                 |
| 15 | PDF     | Blockchain Technology Overview | 1700 | Blockchain            | http://example.com/blockchain           | IBM                      |
| 16 | HTML    | Git Tutorial                   |  400 | Version Control       | http://example.com/git_tutorial         | GitHub                   |
| 17 | Dataset | COVID-19 Data Repository       | 1000 | Epidemiology          | http://example.com/covid_data           | Johns Hopkins University |
| 20 | HTML    | UX Design                      |  800 | Web Development       | https://blog.uxtweak.com/best-ux-resources/ | Google                   |
+----+---------+--------------------------------+------+-----------------------+-----------------------------------------+--------------------------+
18 rows in set (0.00 sec)
```

This figure code defines a stored procedure that inserts a new row into the resources table using the provided parameters.

Built-in functions and stored procedures Figure 4

```
mysql> CREATE PROCEDURE GetExamStats(IN course_id_input INT)
    -> BEGIN
    -> SELECT COUNT(id), AVG(average_score), SUM(total_marks)
    -> FROM exams
    -> WHERE course_id = course_id_input;
    -> END //
Query OK, 0 rows affected (0.02 sec)
```

```
mysql> CALL GetExamStats(3);
+-----------+--------------------+------------------+
| COUNT(id) | AVG(average_score) | SUM(total_marks) |
+-----------+--------------------+------------------+
|         1 |            90.0000 |              100 |
+-----------+--------------------+------------------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)
```

This Procedure calculates the average score, and the total_marks based on the given course_id.