```
/* 1. Create a database named “SIIT” having five tables as shown below. All the
commands must be executed in the Command prompt (not in PHPMyAdmin).
 */
CREATE DATABASE IF NOT EXISTS SIIT;

USE SIIT;

SELECT DATABASE();

source D:\DB_326\6422770345_Lab05\siit.sql

/* (a). You need to have at least 3 data entries (3 rows
of data) for each of the tables using SQL commands.
(0.6 points) */

/* Insert into department */
INSERT INTO `department`(`dept_code`, `dept_name`)
VALUES ('ME','Machanical Engineer'),
('CPE','Computer Engineer'),
('DE','Digital Engineer');

/* Insert into instructor */
INSERT INTO `instructor`(`instructor_ID`, `first_name`, `last_name`, `dept_code`) VALUES
(1,'Gamse','Khemniwat','CPE'),
(2,'Soon','Khemniwat','CPE'),
(3,'Rider','Power','ME');

/* Insert into salary */
INSERT INTO `salary` (`instructor_ID`, `dept_code`, `salary`) VALUES ('1', 'CPE', '999999999'),
('2', 'CPE', '123456789'), ('3', 'ME', '5555555');

/* Insert into student*/
INSERT INTO `student` (`student_ID`, `first_name`, `last_name`, `dept_code`) VALUES ('1',
'Simon', 'Mawow', 'CPE'), ('2', 'Orelo', 'Uronak', 'DE'), ('3', 'Timersak', 'Joner', 'ME');

/* Insert into course*/
INSERT INTO `course` (`course_ID`, `title`, `dept_code`, `credits`) VALUES ('1', 'Quantum
Mechanical', 'CPE', '3'), ('2', 'Basic Elec', 'ME', '3'), ('3', 'BlockChain', 'DE', '3');

/* Insert into teaches*/
INSERT INTO `teaches` (`instructor_ID`, `course_ID`, `sec_ID`, `semester`, `year`) VALUES
('1', '1', '1', '1', '2023'), ('3', '2', '3', '2', '2019'), ('2', '3', '1', '1', '2022');

/* (b). The resulting relational schema should look as
```

```
shown in Figure 1 & should follow the following
rules. (Instructor is related to the department,
salary is related to the department as well). (1.4
points) */
/* - If an instructor resigns, his salary record should
be deleted and if the instructor ID is updated, it
should be updated in the salary table.
- If an instructor leaves/ updates, the teaches
table should also change accordingly.
- If a department code updates then, Instructor,
course, salary and student should be updated as
well.
- However, department entries should not be able
to delete.
- When a course is deleted/updated, then the
taches table should be changed accordingly.
 */

select COLUMN_NAME, CONSTRAINT_NAME, REFERENCED_COLUMN_NAME,
REFERENCED_TABLE_NAME
from information_schema.KEY_COLUMN_USAGE
where TABLE_NAME = 'teaches';

select COLUMN_NAME, CONSTRAINT_NAME, REFERENCED_COLUMN_NAME,
REFERENCED_TABLE_NAME
from information_schema.KEY_COLUMN_USAGE
where TABLE_NAME = 'course';

select COLUMN_NAME, CONSTRAINT_NAME, REFERENCED_COLUMN_NAME,
REFERENCED_TABLE_NAME
from information_schema.KEY_COLUMN_USAGE
where TABLE_NAME = 'department';

select COLUMN_NAME, CONSTRAINT_NAME, REFERENCED_COLUMN_NAME,
REFERENCED_TABLE_NAME
from information_schema.KEY_COLUMN_USAGE
where TABLE_NAME = 'instructor';

select COLUMN_NAME, CONSTRAINT_NAME, REFERENCED_COLUMN_NAME,
REFERENCED_TABLE_NAME
from information_schema.KEY_COLUMN_USAGE
where TABLE_NAME = 'salary';
```

```
select COLUMN_NAME, CONSTRAINT_NAME, REFERENCED_COLUMN_NAME,
REFERENCED_TABLE_NAME
from information_schema.KEY_COLUMN_USAGE
where TABLE_NAME = 'student';


/* (c). if the instructor table is the first one, you're
creating, can you still set up a foreign key
relationship with the department table? */


/* Yeah ofc,

CREATE DATABASE  SIIT_DEMO;

USE SIIT_DEMO;

SELECT DATABASE();

CREATE TABLE `instructor` (
  `instructor_ID` int(10) NOT NULL,
  `first_name` varchar(20) DEFAULT NULL,
  `last_name` varchar(20) DEFAULT NULL,
  `dept_code` varchar(20) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;


CREATE TABLE `department` (
  `dept_code` varchar(20) NOT NULL,
  `dept_name` varchar(20) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

ALTER TABLE `department`
  ADD PRIMARY KEY (`dept_code`);

ALTER TABLE `instructor`
  ADD PRIMARY KEY (`instructor_ID`),
  ADD KEY `dept` (`dept_code`),
  ADD CONSTRAINT `instructor_ibfk_1` FOREIGN KEY (`dept_code`) REFERENCES
`department` (`dept_code`) ON UPDATE CASCADE;

 */

CREATE DATABASE  SIIT_DEMO;
```

```
USE SIIT_DEMO;

SELECT DATABASE();

CREATE TABLE `instructor` (
  `instructor_ID` int(10) NOT NULL,
  `first_name` varchar(20) DEFAULT NULL,
  `last_name` varchar(20) DEFAULT NULL,
  `dept_code` varchar(20) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;


CREATE TABLE `department` (
  `dept_code` varchar(20) NOT NULL,
  `dept_name` varchar(20) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

ALTER TABLE `department`
  ADD PRIMARY KEY (`dept_code`);

ALTER TABLE `instructor`
  ADD PRIMARY KEY (`instructor_ID`),
  ADD KEY `dept` (`dept_code`),
  ADD CONSTRAINT `instructor_ibfk_1` FOREIGN KEY (`dept_code`) REFERENCES
`department` (`dept_code`) ON UPDATE CASCADE;
```

```
/* 2. Let's now create a simple Library database with tables such as 'Books',
'Authors', 'Transactions' and 'Borrowers'. Then answer the following queries.
***creating tables, inserting records & relationships (0.4 points)
*You have sample records, structure and relationships provided with figures.
 */

CREATE DATABASE IF NOT EXISTS LIBRARY;

USE LIBRARY;

SELECT DATABASE();

source D:\DB_326\6422770345_Lab05\library.sql

/* (a) List all books checked out by 'Alice Johnson'. Title and author name
should be listed, author name should be combined with last name
properly. (0.4 points) */

SELECT books.title, CONCAT(authors.first_name, ' ', authors.last_name) AS Author_Name
FROM books
INNER JOIN authors ON authors.author_id=books.author_id
INNER JOIN transactions ON transactions.book_id=books.book_id
INNER JOIN borrowers ON borrowers.borrower_id=transactions.borrower_id

WHERE borrowers.first_name = 'Alice' AND borrowers.last_name = 'Johnson';

/* (b) List all overdue books as below. (0.4 points) */

USE LIBRARY;

SELECT DATABASE();

SELECT books.title, CONCAT(borrowers.first_name, ' ', borrowers.last_name) AS
Borrower_Name, transactions.return_date
FROM transactions
INNER JOIN books ON books.book_id = transactions.book_id
INNER JOIN borrowers ON borrowers.borrower_id = transactions.borrower_id
WHERE transactions.checkout_date >= transactions.return_date;

/* (c) List all authors who have books checked out and the number of books
checked out by each as below. (0.4 points)
 */
```

```
SELECT CONCAT(authors.first_name, ' ', authors.last_name) AS Author_Name, COUNT(*) AS
books_checked_out
FROM transactions
INNER JOIN books ON books.book_id = transactions.book_id
INNER JOIN borrowers ON borrowers.borrower_id = transactions.borrower_id
INNER JOIN authors ON authors.author_id = books.author_id
WHERE transactions.checkout_date IS NOT NULL
GROUP BY authors.author_id
ORDER BY authors.first_name ASC;


/* (d)Find the borrower who has the most books taken from library (0.4
points)
 */

SELECT CONCAT(borrowers.first_name, ' ', borrowers.last_name) AS Borrower_Name,
COUNT(*) AS books_checked_out
FROM transactions
INNER JOIN books ON books.book_id = transactions.book_id
INNER JOIN borrowers ON borrowers.borrower_id = transactions.borrower_id
WHERE transactions.checkout_date IS NOT NULL
GROUP BY borrowers.borrower_id
ORDER BY COUNT(*) DESC LIMIT 1;
```

```
/* 3. Create a Coffee shop database with tables involving ‘products’, customers’,
‘orders’ and ‘order_items’. Then answer the following queries.
***creating tables, inserting records & relationships (0.4 points)
*You have sample records, structure and relationships provided with figures.
 */

CREATE DATABASE IF NOT EXISTS COFFEE_SHOP;

USE COFFEE_SHOP;

SELECT DATABASE();

source D:\DB_326\6422770345_Lab05\coffee_shop.sql

/* (a) List all orders along with the customer's name and order total as
below. (0.4 points)
 */

SELECT orders.order_id, CONCAT(customers.first_name, ' ', customers.last_name) AS
Customer_Name, SUM(products.price * order_items.quantity) AS order_total
FROM orders
INNER JOIN customers ON orders.customer_id=customers.customer_id
INNER JOIN order_items ON orders.order_id=order_items.order_id
INNER JOIN products ON order_items.product_id=products.product_id

WHERE 1
GROUP BY orders.order_id;


/* (b) Calculate the total revenue for the coffee shop as below image. (0.4
points)
 */


SELECT SUM(products.price * order_items.quantity) AS total_revenue

FROM order_items
INNER JOIN products ON order_items.product_id=products.product_id;

/* (c) Create a view to see the most popular products and list them as
below. (0.4 points)
 */
```

```
CREATE OR REPLACE VIEW Most_Pop_Pro AS

SELECT products.name, order_items.quantity AS total_quantity_sold

FROM order_items
INNER JOIN products ON order_items.product_id=products.product_id
ORDER BY order_items.quantity DESC ,products.name DESC
;

SELECT * FROM most_pop_pro;



/* (d) Find the top-spending customers as the below image. (0.4 points) */

USE COFFEE_SHOP;

SELECT DATABASE();

SELECT CONCAT(customers.first_name, ' ', customers.last_name) AS Customer_Name,
SUM(products.price * order_items.quantity) AS total_spent
FROM orders
INNER JOIN customers ON orders.customer_id=customers.customer_id
INNER JOIN order_items ON orders.order_id=order_items.order_id
INNER JOIN products ON order_items.product_id=products.product_id

WHERE 1
GROUP BY customers.customer_id
ORDER BY SUM(products.price * order_items.quantity) DESC;
```

```
/* 4. Let’s create a simple Bank database with ‘customers’, ‘accounts’, and
‘transactions’ tables.
***creating tables, inserting records & relationships (0.4 points)
*You have sample records, structure and relationships provided with figures.
 */

CREATE DATABASE IF NOT EXISTS BANK;

USE BANK;

SELECT DATABASE();

source D:\DB_326\6422770345_Lab05\bank.sql


/* (a) List all customers and their account types along with the total balance
for each customer as below. (0.8 points)
 */

SELECT CONCAT(customers.first_name, ' ', customers.last_name) AS Customers_Name,
GROUP_CONCAT(accounts.account_type ORDER BY accounts.account_type ASC
SEPARATOR ', ') as account_types,
SUM(accounts.balance) AS total_balance

FROM accounts
INNER JOIN customers ON accounts.customer_id = customers.customer_id
GROUP BY customers.customer_id
ORDER BY total_balance DESC;


/* (b) Find the top 3 customers with the highest total balance across all
accounts as the given image. (0.8 points)
 */

SELECT CONCAT(customers.first_name, ' ', customers.last_name) AS Customers_Name,
SUM(accounts.balance) AS total_balance

FROM accounts
INNER JOIN customers ON accounts.customer_id = customers.customer_id
GROUP BY customers.customer_id
ORDER BY total_balance DESC LIMIT 3;
```

```
/* 5. Let’s create a simple search engine with ‘web pages’, ‘search queries’, and
‘search results’ tables. Then answer the following queries.
***creating tables, inserting records & relationships (0.8 points)
*You have sample records, structure and relationships provided with figures.
 */

CREATE DATABASE IF NOT EXISTS search_queries;

USE search_queries;

SELECT DATABASE();

source D:\DB_326\6422770345_Lab05\search_engine.sql


/* (a) Update the content of a web page based on its URL as below. (0.4
points)
 */

UPDATE web_pages SET content='This is the updated content of page 1.' WHERE url =
'http://www.example.com/page1' ;

SELECT * FROM web_pages;

/* (b) List the web pages ranked by their appearance in search results for a
specific query as the given image. (0.8 points)
 */

SELECT search_queries.query_text, web_pages.title, web_pages.url, search_results.rank

FROM search_queries
INNER JOIN search_results ON search_results.query_id=search_queries.query_id
INNER JOIN web_pages ON web_pages.page_id=search_results.page_id

WHERE LOWER(search_queries.query_text) LIKE "%search engine%"
ORDER BY search_results.rank ASC;
```