/\*1. SELECT STATEMENT\*/

/\*Query data of the whole table 'customer'\*/

select \* from customer;

/\*Query for columns: first name, last name, email from table 'customer'\*/

select first\_name, last\_name, email

from customer;

/\*1.1 SORT DATA\*/

/\*Sorts customers by the first name in ascending order\*/

select first\_name

from customer

order by first\_name asc;

/\*Sort customers by the last name in descending order\*/

SELECT last\_name, first\_name

FROM customer

ORDER BY last\_name ASC;

/\*Sort customers by the first name in the ascending order first, and then sort the sorted result set by the last name in descending order\*/

SELECT first\_name, last\_name

FROM customer

ORDER BY

first\_name asc,

last\_name desc;

/\*1.2 FILTER DATA\*/

/\*Filter customers whose first names are Jamie\*/

SELECT first\_name, last\_name

FROM customer

WHERE first\_name like 'Jamie';

/\*Filter customers whose first name is Jamie and last name is Rice\*/

SELECT first\_name, last\_name

FROM customer

WHERE first\_name like 'Jamie' AND

last\_name like 'Rice';

/\*Filter customers who paid the rental with amount is either less than $2 or greater than $8\*/

SELECT distinct first\_name, last\_name

FROM customer c

INNER JOIN payment p ON c.customer\_id = p.customer\_id

WHERE p.amount > 8 OR p.amount < 2;

/\*1.3 LIMIT AND OFFSET\*/

/\*Retrieve the first 5 films ordered by film\_id\*/

SELECT title

FROM film

ORDER BY film\_id

LIMIT 5;

/\*Retrieve 4 films starting from the third one, ordered by film\_id\*/

SELECT title

FROM film

ORDER BY film\_id

LIMIT 4

OFFSET 3;

/\*Retrieve the top 10 most expensive films\*/

SELECT title, replacement\_cost

FROM film

ORDER BY replacement\_cost DESC

LIMIT 10;

/\*1.4 Advanced condition: IN\*/

/\*Retrieve rental information of customer id 1 and 2\*/

SELECT customer\_id, last\_name

FROM customer

WHERE customer\_id IN (1,2);

/\*Retrieve customer\_id of customers that has rental’s return date on 2005-05-27\*/

SELECT c.customer\_id, r.return\_date

FROM customer c

INNER JOIN rental r ON c.customer\_id = r.customer\_id

WHERE CAST(r.return\_date AS DATE) = '2005-05-27'

ORDER BY c.customer\_id;

/\*Similar to the previous task, except the returns this time are customers' first names and last names \*/

SELECT c.last\_name, c.first\_name

FROM customer c

INNER JOIN rental r ON c.customer\_id = r.customer\_id

WHERE CAST(r.return\_date AS DATE) = '2005-05-27';

/\*Retrieve payments whose date is between 2007-02-07 and 2007-02-15\*/

SELECT p.payment\_id, p.payment\_date

FROM payment p

WHERE CAST(p.payment\_date AS DATE) BETWEEN '2007-02-07' AND '2007-02-15';

/\*Retrieve customers whose first name has the 2nd and 3rd letters are 'he'\*/

SELECT first\_name

FROM customer

WHERE first\_name LIKE '\_he%';

/\*1.5 GROUP BY\*/

/\*Retrieve total payment based on customer\_id, then sort in a descending order\*/

SELECT sum(p.amount), c.customer\_id

FROM payment p

INNER JOIN customer c ON c.customer\_id = p.customer\_id

GROUP BY c.customer\_id

ORDER BY c.customer\_id DESC;

/\*Count the number of transaction each staff has done\*/

SELECT count(p.payment\_id), s.last\_name

FROM payment p

INNER JOIN staff s ON s.staff\_id = p.staff\_id

GROUP BY s.staff\_id;

select \* from staff;

/\*Retrieve customer\_id and the toal payment of only customers who have spent more than $150 \*/

SELECT c.customer\_id, sum(p.amount)

FROM customer c

INNER JOIN payment p ON p.customer\_id = c.customer\_id

GROUP BY c.customer\_id

HAVING sum(p.amount) > 150;

/\*2. JOIN TABLES\*/

/\*2.1 INNER JOIN\*/

/\*Retrieve customer id, first name, last name, email, the amount paid, and payment date for only customers who have at least 1 payment\*/

SELECT c.customer\_id, c.first\_name, c.last\_name, c.email, p.amount, p.payment\_date

FROM customer c

INNER JOIN payment p ON c.customer\_id = p.customer\_id;

/\*Similar to the previous task, this time sort the results by customer id\*/

SELECT c.customer\_id, c.first\_name, c.last\_name, c.email, p.amount, p.payment\_date

FROM customer c

INNER JOIN payment p ON c.customer\_id = p.customer\_id

ORDER BY c.customer\_id;

/\*Same task, this time retrieve only customers with id 2 and 3\*/

SELECT c.customer\_id, c.first\_name, c.last\_name, c.email, p.amount, p.payment\_date

FROM customer c

INNER JOIN payment p ON c.customer\_id = p.customer\_id

WHERE c.customer\_id IN (2,3)

ORDER BY c.customer\_id;

/\*Inner join both tables payment and staff into the customer table to retrieve customer data, staff, and payment information\*/

SELECT c.last\_name, c.first\_name, s.username, p.amount

FROM customer c

INNER JOIN payment p ON c.customer\_id = p.customer\_id

INNER JOIN staff s ON s.staff\_id = p.staff\_id;

/\*2.2 LEFT JOIN\*/

/\*Left join the film and inventory tables to retrieve film id, title, and inventory id\*/

SELECT f.film\_id, f.title, i.inventory\_id

FROM film f

LEFT OUTER JOIN inventory i ON f.film\_id = i.film\_id;

/\*Similar task, except this time only retrieve films that do not have an inventory id\*/

SELECT f.film\_id, f.title, i.inventory\_id

FROM film f

LEFT OUTER JOIN inventory i ON f.film\_id = i.film\_id

WHERE i.film\_id is NULL;

/\*3. CREATE TABLES\*/

/\*3.1 SELECT INTO\*/

/\*creates a new table named film\_r that contains all films with the rating R and rental duration 5 days from the film table\*/

/\*Creates a temporary table named short\_film that contains all films whose lengths are under 60 minutes.\*/

/\*3.2 ALTER TABLE\*/

/\*Create a new table called 'link', 3 columns: link\_id as primary key, title as text NOT NULL, url as text NOT NULL, UNIQUE\*/

/\*Now add a column named 'active' with values as yes, no, and null\*/

/\*Try to use other syntax to modify columns and tables using the ALTER TABLE syntax\*/

/\*4. IMPORT AND EXPORT\*/

/\*4.1 Import\*/

/\*Create a table named 'persons', columns: id, first\_name, last\_name, dob, email\*/

/\*Import the sample1.csv file into the persons table\*/

/\*~ the end ~ YOU KNOW SQL NOW!\*/