# Homework 4

**Due: 09/27/2024 by 11:59 PM**

1. Your manager has requested a list of orders where the shipment was delayed, so the employees responsible can reach out to affected customers. In the ‘**Northwind’** database, write a query to retrieve the ‘order\_id’ from the ‘**orders’** table for orders shipped after the required delivery date. Include the corresponding ‘customer\_id’ and the first and last names of the employees from the ‘**employees’** table who processed each order. Display the ‘order\_ids’ in descending order.

## SELECT

order\_id, o.customer\_id,

first\_name AS employee\_first\_name, last\_name AS employee\_last\_name FROM orders AS O

INNER JOIN customers AS C

ON O.customer\_id = C.customer\_id INNER JOIN employees AS E

ON O.employee\_id = E.employee\_id WHERE required\_date < shipped\_date ORDER BY order\_id DESC;

***Explanation:*** *This query retrieves the order IDs for orders that were shipped after the required delivery date. It also includes customer and employee details to help follow up with affected customers. The result is ordered in descending order by order ID.*

1. Northwind company is interested in identifying which categories have the most products with high unit prices. In the ‘**Northwind’** database, write a query to find the number of products in each category from the ‘**products’** table where the unit price exceeds 15. Show the category name from the ‘**categories’** table and the corresponding count of high- priced products. Sort the results by the number of products in descending order and identify the category with the most products.

## SELECT

category\_name,

COUNT(\*) AS count\_of\_products FROM products AS P

INNER JOIN categories AS C ON P.category\_id = C.category\_id WHERE unit\_price > 15

GROUP BY category\_name ORDER BY COUNT(\*) DESC;

# ‘Condiments’ is the category with the most products which has a unit price exceeding 15

# *****Explanation:***** *This query lists product categories where the unit price exceeds 15, showing the count of products for each category. The results are sorted by the number of products in descending order, highlighting the category with the most high-priced products.*

1. In the ‘**Northwind’** database, list the suppliers' company name and contact name from the ‘**suppliers’** table that provide products in the 'Seafood' category, as defined in the ‘**categories’** and ‘**products’** tables.

## SELECT

company\_name AS supplier, contact\_name AS point\_of\_contact FROM suppliers AS S

INNER JOIN products AS P

ON S.supplier\_id = P.supplier\_id INNER JOIN categories AS C ON P.category\_id = C.category\_id

WHERE category\_name = 'Seafood';

***Explanation:*** *This query retrieves the company name and contact information for suppliers who provide products in the 'Seafood' category, based on data from the 'suppliers', 'products', and 'categories' tables.*

1. Your manager wants to know how many distinct actors have appeared in films with an 'NC-17' rating. In the ‘**dvdrental’** database, write a query to retrieve the number of distinct actors from the ‘**actor’** table who has appeared in films from the ‘**film’** table with an 'NC-17’ rating.

## SELECT

COUNT(DISTINCT A.actor\_id) AS no\_of\_actors\_in\_NC\_17\_films FROM actor AS A

INNER JOIN film\_actor AS FA ON A.actor\_id = FA.actor\_id INNER JOIN film AS F

ON FA.film\_id = F.film\_id WHERE rating = 'NC-17';

***Explanation:*** *This query counts the distinct actors who have appeared in films rated 'NC-17'. It joins the 'actor', 'film\_actor', and 'film' tables to get the necessary data*

1. In the ‘**dvdrental’** database, write a query to retrieve the ‘customer\_id’ and email of the top 5 customers with the highest number of DVD rentals, based on the ‘**rental’** and ‘**customer’** tables.

## SELECT

R.customer\_id, email

FROM rental AS R

INNER JOIN customer AS C

ON R.customer\_id = C.customer\_id GROUP BY R.customer\_id,email ORDER BY COUNT(\*) DESC LIMIT 5;

***Explanation:*** *This query finds the top 5 customers with the highest number of DVD rentals, using the 'rental' and 'customer' tables. The results are ordered by the count of rentals in descending order.*

1. In the ‘**dvdrental’** database, write a query to retrieve the ‘category name’ from the ‘**category’** table, the average film length from the ‘**film’** table, and the number of films in each category. Sort the results by ‘category name’ in ascending order.

### SELECT

name AS category\_name, AVG(length),

COUNT(\*) AS number\_of\_films FROM FILM AS F

INNER JOIN film\_category AS FC ON F.film\_id = FC.film\_id

INNER JOIN category AS C

ON FC.category\_id = C.category\_id GROUP BY name

ORDER BY name;

***Explanation:*** *This query retrieves the average film length and the number of films in each category. The results are ordered alphabetically by category name.*

1. In the ‘**dvdrental’** database, write a query that captures the ‘actor id’ and counts the number of movies each actor has made through the ‘**film\_actor’** and ‘**actor’** tables. Retrieve the top 5 actors (‘actor\_id’) with the maximum number of movies.

### SELECT

FA.actor\_id, COUNT(\*)

FROM film\_actor AS FA INNER JOIN actor AS A

ON A.actor\_id = FA.actor\_id GROUP BY FA.actor\_id ORDER BY COUNT(\*) DESC LIMIT 5;

***Explanation:*** *This query counts the number of films each actor has appeared in, showing the top 5 actors with the highest number of films. The 'film\_actor' and 'actor' tables are joined to gather the necessary data*.

1. In the ‘**Northwind’** database, write a query that returns the employee id and first name of employees from the ‘**employees’** table along with the first name of their managers. Use ‘**employee\_id’** and ‘**reports\_to’** columns from the employees table (Hint: Self-join). Make sure your output also includes if an employee does not have a manager. Use appropriate column aliases for the first name of the employee and manager.

### SELECT

E1.employee\_id,

E1.first\_name AS employee\_first\_name, E2.first\_name AS manager\_first\_name FROM employees AS E1

LEFT JOIN employees AS E2

ON E1.reports\_to = E2.employee\_id;

***Explanation:*** *This query retrieves the employee ID and first name along with their manager's first name using a self-join on the 'employees' table. The LEFT JOIN ensures employees without a manager are also included.*

1. In the ‘**dvdrental’** database, write a query to find the most rented film categories by using the ‘**category’**, ‘**film\_category’**, ‘**film’**, ‘**inventory’**, and ‘**rental’** tables.

### SELECT

name AS category FROM category AS C

INNER JOIN film\_category AS FC ON C.category\_id = FC.category\_id INNER JOIN film AS F

ON FC.film\_id = F.film\_id INNER JOIN inventory AS I ON F.film\_id = I.film\_id INNER JOIN rental AS R

ON I.inventory\_id = R.inventory\_id GROUP BY name

### ORDER BY COUNT(\*) DESC;

**The most rented film category is Sports.**

***Explanation:*** *This query identifies the most rented film categories by counting the rentals in each category. It joins the 'category', 'film\_category', 'film', 'inventory', and 'rental' tables to gather the necessary data.*

1. In the ‘**dvdrental’** database, write a query to find **unique** film titles from the ‘**film’** table that start with 'A' and have been rented, utilizing the ‘**inventory’** and ‘**rental’** tables.

SELECT DISTINCT title

FROM film as F

INNER JOIN inventory AS I ON F.film\_id = I.film\_id INNER JOIN rental AS R

ON I.inventory\_id = R.inventory\_id WHERE title LIKE 'A%';

***Explanation:*** *This query finds the distinct film titles starting with 'A' that have been rented. It uses joins between the 'film', 'inventory', and 'rental' tables.*

1. **Bonus question:** In the ‘**dvdrental’** database, write a query to find actors in the ‘**actor’** table who share the same first name but have different last names. Use a self-join for this query.

### SELECT

A1.first\_name, A1.last\_name FROM actor AS A1

INNER JOIN actor AS A2

ON A2.first\_name = A1.first\_name WHERE A1.actor\_id <> A2.actor\_id AND A1.last\_name <> A2.last\_name GROUP BY DISTINCT A1.actor\_id ORDER BY A1.first\_name;

***Explanation:*** *This query uses a self-join on the 'actor' table to find actors with the same first name but different last names. It ensures that the same actor is not compared to themselves.*