**IVY TECH COMMUNITY COLLEGE**

**DBMS130 – M08 Project Final**

Use the M04 Project ERD and the tables created for the midterm project to complete the steps listed below.

# **Setting Up Your Script (5 points)**

1. Open a text editor to begin writing your SQL script that will contain your SQL statements. (You may use any program that allow you to create .sql files (e.g., Notepad, Wordpad, Notes, TextEdit, Notepad++, etc.)
2. Name the file M08\_Project\_LastName\_FirstName.sql (e.g., M08\_Project\_Doe\_John.sql).
3. Execute your script to ensure it runs without errors. (If your code produces errors, work to resolve them before submitting.)
4. Save the script and upload it as your submission.

Points will be deducted if a .sql file is not submitted. Your script should contain 1 view, 4 sequences, 1 index, 1 synonym, 1 multi-line comment, and 3 single-line comments.

# **Creating a View (10 points)**

* 1. Create a view called TITLE\_UNAVAIL to show the movie titles and media\_id of the media not returned yet. The view should not allow any DML operations.

CREATE VIEW TITLE\_UNAVAIL  AS

SELECT MV.TITLE, MD.MEDIA\_ID

FROM MEDIA MD

JOIN MOVIES MV

ON MD.TITLE\_ID = MV.TITLE\_ID

JOIN RENTAL\_HISTORY RH

ON MD.MEDIA\_ID = RH.MEDIA\_ID

WHERE RETURN\_DATE IS NULL;

1. **Creating Sequences (5 points)**Create the following sequences to be used for primary key values:
   1. Use a sequence to generate PKs for CUSTOMER\_ID in RENTAL\_CUSTOMERS table.   
      Begin at 101 and increment by 1.

CREATE SEQUENCE CUSTOMER\_ID\_SEQ

START WITH 101

INCREMENT BY 1;

NB:

* 1. Use a sequence to generate PKs for TITLE\_ID in MOVIES table.   
     Begin at 1 and increment by 1.

CREATE SEQUENCE TITLE\_ID\_SEQ

START WITH 1

INCREMENT BY 1;

* 1. Use a sequence to generate PKs for MEDIA\_ID in MEDIA table.   
     Begin at 92 and increment by 1.

CREATE SEQUENCE MEDIA\_ID\_SEQ

START WITH 92

INCREMENT BY 1;

* 1. Use a sequence to generate PKs for ACTOR\_ID in ACTOR table.   
     Begin at 1001 and increment by 1.

CREATE SEQUENCE ACTOR\_ID\_SEQ

START WITH 1001

INCREMENT BY 1;

1. **Creating Indexes (5 points)**
   1. Create an index on the last\_name column of the Rental Customers table.

CREATE INDEX RENTAL\_CUSTOMERS\_LASTNAME\_IDX ON RENTAL\_CUSTOMERS(LAST\_NAME);

1. **Creating Synonyms (5 points)**
   1. Create a synonym called TU for the TITLE\_UNAVAIL view.

CREATE SYNONYM TU

FOR TITLE\_UNAVAIL;

1. **Writing Queries (20 points)**Note: You may need to write INSERT statements to add more data to your tables.
   * Read each of the questions listed in the left column of the table below.
   * Write the answer to your question and/or the required SQL statement in the corresponding cell in the right column that answers the question. (Test your statement to confirm the results before submitting.).

|  |  |
| --- | --- |
| 1. To determine the popularity of the movies in the OracleFlix inventory, write a query to display each movie and the number of times it was rented. Sort your results by the count of rentals in descending order. | SELECT MV.TITLE\_ID,MV.TITLE, COUNT(RH.RENTAL\_DATE) AS RENTAL\_COUNT  FROM MOVIES MV  LEFT JOIN MEDIA MD ON MV.TITLE\_ID = MD.TITLE\_ID  JOIN RENTAL\_HISTORY RH ON MD.MEDIA\_ID=RH.MEDIA\_ID  GROUP BY MV.TITLE\_ID,MV.TITLE  ORDER BY RENTAL\_COUNT DESC; |
| 1. OracleFlix will be mailing coupons to the homes of frequent renters. They are printing labels for each customer. Write a query to identify the frequent renters (i.e., customers that have rented more than once). Your results should include the customers’ first and last name concatenated, address, city, and state. Also include 2 columns – name\_length and address\_length – that display the length of the concatenation of the first and last name and the length of the address. Sort your results by the customers’ last names. | SELECT CONCAT(CONCAT(FIRST\_NAME, ' '),LAST\_NAME) AS FULL\_NAME,  CONCAT (CONCAT(CONCAT(CONCAT(ADDRESS, ' '), CITY), ' '), STATE) AS FULL\_ADDRESS,  LENGTH(CONCAT(FIRST\_NAME, LAST\_NAME)) AS NAME\_LENGTH,  LENGTH(CONCAT(CONCAT(ADDRESS,CITY),STATE)) AS ADDRESS\_LENGTH  FROM RENTAL\_CUSTOMERS  WHERE CUSTOMER\_ID IN (  SELECT CUSTOMER\_ID  FROM RENTAL\_HISTORY  GROUP BY CUSTOMER\_ID  HAVING COUNT(\*) > 1  )  ORDER BY LAST\_NAME; |
| 1. With more people buying Blu-ray products, OracleFlix is considering clearing VHS products out of their inventory. Before they do, they need to know if customers are renting that type of media. Write a query to identify the number of times each type of media has been rented. Sort your results by the count of rentals descending. | SELECT FORMAT, COUNT(\*) AS RENTAL\_COUNT  FROM MEDIA M  JOIN RENTAL\_HISTORY RH  ON M.MEDIA\_ID = RH.MEDIA\_ID  GROUP BY FORMAT  ORDER BY RENTAL\_COUNT DESC; |
| 1. Customers often ask for movies by actor. Write a query that lists the actors in the movies in the OracleFlix inventory. Your results should include the actor’s name (first and last) and the movie title. To help OracleFlix employees answer the customers’ questions quickly, sort the results by the actors’ last name and first name. | SELECT AC.FIRST\_NAME, AC.LAST\_NAME, MV.TITLE  FROM ACTORS AC  JOIN STAR\_BILLINGS SB  ON AC.ACTOR\_ID = SB.ACTOR\_ID  JOIN MOVIES MV  ON SB.TITLE\_ID = MV.TITLE\_ID  ORDER BY AC.LAST\_NAME, AC.FIRST\_NAME; |
| 1. OracleFlix would like to create a section of their inventory called “Classics.” To do this, they need to identify movies with a release date of at least 25 years ago (i.e., 1996 or before). Write a query that identifies these movies. Your results should include the release date and the movie title. Sort your results by release date. | CREATE VIEW CLASSICS AS  SELECT TITLE, RELEASE\_DATE  FROM MOVIES  WHERE EXTRACT(YEAR FROM RELEASE\_DATE) <= 1996  ORDER BY RELEASE\_DATE;  --OR—  CREATE VIEW CLASSICS AS  SELECT TITLE, RELEASE\_DATE  FROM MOVIES  WHERE EXTRACT(YEAR FROM SYSDATE) - EXTRACT(YEAR FROM RELEASE\_DATE) >= 25  ORDER BY RELEASE\_DATE; |

## **Creating Queries (10 points)**

You have ideas for 2 inventory and rentals reports that could be helpful to the leadership of the OracleFlix company. Develop the queries to support the reports you plan to propose and explain how they would be helpful to leadership.

Note: You may need to write INSERT statements to add more data to your tables.

* + Your query should include the SQL concepts, keywords, and/or clauses in column 1.
  + Test your statement to confirm the results before submitting and place your final query in the column 2.
  + Describe the output of your query as if you were presenting it to a client (e.g., what does it mean, how would it be useful to them) in column 3
  + Save this document once you have added your answers and uploaded it as part of your submission.

|  |  |  |
| --- | --- | --- |
| **Query Requirements** | **Query** | **Meaning / Value of Query Results** |
| Query 1   * At least 1 join * GROUP BY * At least 1 multiple-row / aggregate function | SELECT A.FIRST\_NAME,A.LAST\_NAME,M.TITLE,MIN(A.BIRTH\_DATE)  FROM ACTORS A  JOIN STAR\_BILLINGS SB  ON A.ACTOR\_ID=SB.ACTOR\_ID  JOIN MOVIES M  ON SB.TITLE\_ID=M.TITLE\_ID  GROUP BY A.FIRST\_NAME, A.LAST\_NAME,M.TITLE  ORDER BY A.FIRST\_NAME ASC; | This query obtains the minimum age and name of actor according to their name, and movie title. MIN is the aggregate function which is used to select minimum value of corresponding column. |
| Query 2   * At least 1 subquery * At least 1 Column Alias | SELECT CONCAT(CONCAT(RC.FIRST\_NAME, ' '),RC.LAST\_NAME) AS "FULL NAME", M.FORMAT AS "MEDIA TYPE ORDERED"  FROM RENTAL\_CUSTOMERS RC  JOIN RENTAL\_HISTORY RH  ON RC.CUSTOMER\_ID=RH.CUSTOMER\_ID  JOIN MEDIA M  ON RH.MEDIA\_ID=M.MEDIA\_ID  WHERE RC.CUSTOMER\_ID IN (  SELECT CUSTOMER\_ID  FROM RENTAL\_HISTORY  WHERE RH.CUSTOMER\_ID IS NOT NULL)  ORDER BY FIRST\_NAME; | Column alias is used to represent the aggregate function as a name which will be used to identify the column name. The query finds customers full name together with the media type they ordered. The result shows customer full name in one column and the ”Media Type they Ordered” in another column |