**A screenshot of a browser

Description automatically generatedDatabase Design – 2023W**   
**Student ID:** 901142 **Student Name:** Roshan Shrestha **Practical Activity#21#22#23#24**  
---------------------------------------------------------------------------------------------------------------------  
The list of all the created tables after executing the provided SQL script are listed below:  
  
**Table: aliases**

A screenshot of a computer

Description automatically generated**Table : criminals**

A screenshot of a computer

Description automatically generated**Table : crimes**

A screenshot of a computer

Description automatically generated**Table : sentences**

**A screenshot of a computer

Description automatically generatedTable : prob\_officers**

**A screenshot of a computer

Description automatically generatedTable: officers**

A screenshot of a computer

Description automatically generated**Table: crime\_codes**

A screenshot of a computer

Description automatically generated**Table: appeals**

A screenshot of a computer

Description automatically generated**Table: crime\_officers**

A screenshot of a computer

Description automatically generated**Table: crime\_charges**

**A screenshot of a computer

Description automatically generatedTable: prob\_contact**

**A screenshot of a computer

Description automatically generatedTable: criminals\_dw**

**Practical Activity #1  
1. List all criminal aliases beginning with the letter B.** In order to list all the criminal aliases that begins with the letter **B**, we need to use **LIKE** operator which uses pattern matching method to search for the value in respective column’s value. The query I used is as below:  
 **SELECT alias\_id, criminal\_id, alias  
FROM aliases  
WHERE alias LIKE 'B%'**The **LIKE** keyword does a pattern matching search where the pattern 'B%' specifies that we need to look for aliases starting with the letter B, followed by any number of characters. The '%' symbol is a wildcard character that matches any number of characters. The output of the above A screenshot of a browser

Description automatically generatedscript is below:

**2. List all crimes that occurred (were charged) during the month November 2008. List the crime ID, criminal ID, date charged, and classification.**To list all the crimes that occurred during the given month and year, we need to convert the dates into the **DD-MON-YY** format using the below query:

**Alter session set nls\_date\_format='DD-MON-YY';**This is used to modify the default date format in Oracle for the current session. Dates are displayed in the format 'day-month-year' when the **NLS\_DATE\_FORMAT** parameter is set to **'DD-MON-YY'**.

After doing the necessary changes we can use the below query to reterive the required data:

**SELECT crime\_id AS roshan\_Crime\_ID, criminal\_id, date\_charged, classification  
FROM crimes  
WHERE date\_charged BETWEEN DATE '2008-11-01' AND DATE '2008-11-30';**

The SQL query above retrieves the **crime\_id** (renamed as **roshan\_Crime\_ID** in the output), **criminal\_id**, **date\_charged,** and **classification** from the **crimes** table. It narrows the results to only those rows where the date charged occurs between November 1, 2008, and November 30, 2008.The output after executing the above query can be visualized in the snippet below:

A screenshot of a computer

Description automatically generated

**3. List all crimes with a status of CA (can appeal) or IA (in appeal). List the crime ID, criminal ID, date charged, and status.**In order to view the list of all crimes that has status of **CA(can appeal)** or **IA(In appeal)**. We can execute the query below:

**SELECT crime\_id AS roshan\_Crime\_ID, criminal\_id, date\_charged, status  
FROM crimes  
WHERE status IN ('CA', 'IA')**

Here using the **AS** keyword, we are selecting the **crime\_id** and storing it as **roshan\_Crime\_ID** which will be new column name while displaying the result, along with**, criminal\_id, date\_charged, status**. The **FROM** keyword specifies that from which table we are going to select data, here we are taking data from **crimes** table, and finally using the **WHERE** keyword we are applying the condition where the data will be filtered if it has value as **CA** or **IA**, in the status column. The output from the query execution can be observed below:

**A screenshot of a computer

Description automatically generated  
Practical Activity #2**   
**4. List all crimes classified as a felony. List the crime ID, criminal ID, date charged, and classification. (Classification is ‘F’).**   
To retrieve the list of all crimes with classification as **felony** we can execute the below query:

**SELECT crime\_id AS roshan\_Crime\_ID, criminal\_id, date\_charged, classification  
FROM crimes  
WHERE classification = 'F'**

A screenshot of a computer

Description automatically generatedInitially the **SELECT** statement will specifies the columns that we want to include in the result, here we are trying to retrieve the data from **criminal\_id, date\_charged, classification** columns. And using the **FROM** statement we are representing the table in which we will apply the search query i.e.  **crimes** table. Then the **WHERE** statement will specity the condition that make sure that the result will have **classification** as **'F'**. The output from the query is attached below:

**5. List all crimes with a hearing date more than 14 days after the date charged. List the crime ID, criminal ID, date charged, and hearing date.**

To list all the crimes with a hearing date more than 14 days after the charged date, we can use the query below:

**SELECT crime\_id AS roshan\_Crime\_ID, criminal\_id, date\_charged, hearing\_date  
FROM crimes  
WHERE hearing\_date > date\_charged + 14**

**A screenshot of a computer

Description automatically generated**In this query we are trying to retrieve the list of crimes from **crimes** table where the value of **hearing\_date**, is more than 14 days after the **date\_charged**. And we want our result output to include the **crime\_id** with name **roshan\_Crime\_ID**, alomg with **criminal\_id, date\_charged, hearing\_date**. The output from the query can be visualized below:

**6. List all criminals with the zip code 23510. List the criminal ID, last name, and zip code. Sort the list by criminal ID.**

To list all the criminals where the zip code is 23510, we can run the below query:

**SELECT criminal\_id AS roshan\_Criminal\_ID, last, zip  
FROM criminals  
WHERE zip = '23510'  
ORDER BY criminal\_id**

A screenshot of a computer

Description automatically generatedThe query above retrieves the **criminal\_id** from the criminals table and renames it **roshan\_Criminal\_ID**. It also chooses the columns for **last** and **zip**. The query narrows the results to only those with the **zip code '23510'**. Finally, the results are ordered ascendingly by the **criminal\_id** column. The output of the executed query is represented as below:

**Practical Activity #3**   
**7. List all crimes that don’t have a hearing date scheduled. List the crime ID, criminal ID, date charged, and hearing date.**

To retrieve the data of all crimes that don’t have a hearing date scheduled we can execute the query below:

**SELECT crime\_id AS roshan\_Crime\_ID, criminal\_id, date\_charged, hearing\_date  
FROM crimes  
WHERE hearing\_date IS NULL**

The SQL query above retrieves the **crime\_id** (renamed **roshan\_Crime\_ID** in result), criminal ID, date charged, and hearing date from the crimes table. It restricts the results to entries where the hearing date is NULL (not scheduled). The output of the query is attached below:

A screenshot of a computer

Description automatically generated

**8. List all sentences with a probation officer assigned. List the sentence ID, criminal  
ID, and probation officer ID. Sort the list by probation officer ID and then criminal ID.**In order to list down all the sentences which has a probation officer assigned, we can use the query below:

**SELECT sentence\_id AS roshan\_Sentence\_ID, criminal\_id, prob\_id  
FROM sentences  
WHERE prob\_id IS NOT NULL  
ORDER BY prob\_id, criminal\_id**

**A screenshot of a computer

Description automatically generated**From the sentences table, the SQL query pulls the **sentence\_id** (renamed **roshan\_Sentence\_ID** in the output), **criminal\_id**, and **prob\_id**. It restricts the results to rows where the **prob\_id** is not **NULL**. The results are then ordered by **prob\_id,** first, followed by **criminal\_id**. The output is attached bwlow:

**Practical Activity #4**   
**9. List all crimes that are classified as misdemeanors (classification =’M’) and are currently in appeal (stat is ‘IA’). List the crime ID, criminal ID, classification, and status.**

To list down all the crimes that are under the classification as misdemeanors and are currently under the status as in appeal(IA), we can execute the query below:

**SELECT crime\_id AS roshan\_Crime\_ID, criminal\_id, classification, status  
FROM crimes  
WHERE classification = 'M' AND status = 'IA'**

**A screenshot of a computer

Description automatically generated**The SQL query above retrieves the **crime\_id** (renamed as **roshan\_Crime\_ID** in the output), **criminal\_id, classification and status** from the **crimes** table. It narrows the results to only those with the categorization 'M' (misdemeanour) and the status 'IA' (in appeal). The output from the query is attached below:

**10. List all crime charges with a balance owed. List the charge ID, crime ID, fine amount, court fee, amount paid, and amount owed.**

To list down the all the charges with a balance owed we can use the query below:

**SELECT charge\_id AS roshan\_Charge\_ID, crime\_id, fine\_amount, court\_fee, amount\_paid, (fine\_amount + court\_fee - amount\_paid) AS amount\_owed  
FROM crime\_charges  
WHERE (fine\_amount + court\_fee - amount\_paid) > 0**

**A screenshot of a computer

Description automatically generated**The SQL query above picks the **charge\_id** (renamed as **roshan\_Charge\_ID**), **crime\_id**, **fine\_amount**, **court\_fee**, and **amount\_paid**, and calculates the amount owed as the difference between the total of the fine amount and the amount paid. It returns rows when the calculated amount owed exceeds zero. The output is attached below: