Assignment #4 Part 1: Demo of Designing Views/ Simple Queries

Arman Grewal(501100160), Gurpreet Singh Bhatti(501103368), Jasdeep Singh(501089933)

Faculty of Engineering, Toronto Metropolitan University

CPS 510: Database Systems I (Section 7, Group 8, Topic 33)

Jorge Lopez

October 6th, 2023

Queries:

Orthodontic Specialists Directory: A-Z Order:

```
SET PAGESIZE 100
SET LINESIZE 150
TTITLE "Orthodontic Specialists Directory: A-Z Order and Names starting with Dr";
COLUMN dentist_id FORMAT 99999
COLUMN email_address FORMAT A30
COLUMN phone_number FORMAT A20
COLUMN name FORMAT A30
COLUMN specialization FORMAT A20
SELECT DISTINCT dentist_id, email_address, phone_number, name, specialization
FROM dentists
WHERE specialization = 'Orthodontics' AND name LIKE 'Dr%'
ORDER BY name ASC;
```

Script Output | Query Result

Task completed in 0.093 seconds

```
Page:          1


Orthodontic Specialists Directory: A-Z Order and Names starting with Dr
DENTIST_ID EMAIL_ADDRESS                  PHONE_NUMBER         NAME                           SPECIALIZATION
---------- ------------------------------ -------------------- ------------------------------ --------------------
        12 dr.roberts@example.com         555-555-1012         Dr. Christopher Roberts        Orthodontics
         8 dr.thompson@example.com        555-555-1008         Dr. Emily Thompson             Orthodontics
        19 dr.hall@example.com            555-555-1019         Dr. Laura Hall                 Orthodontics
         2 dr.jones@example.com           555-555-1002         Dr. Sarah Jones                Orthodontics
```

SET PAGESIZE 100
SET LINESIZE 150
TTITLE "Orthodontic Specialists Directory: A-Z Order and Names starting with Dr";
COLUMN dentist_id FORMAT 99999
COLUMN email_address FORMAT A30
COLUMN phone_number FORMAT A20
COLUMN name FORMAT A30
COLUMN specialization FORMAT A20
SELECT DISTINCT dentist_id, email_address, phone_number, name, specialization
FROM dentists
WHERE specialization = 'Orthodontics' AND name LIKE 'Dr%'
ORDER BY name ASC;

This SQL query retrieves data from the “dentists” table, specifically for dentists specializing in orthodontics and their name starts with “Dr.”. It will sort the results alphabetically by their name, making accessing information about orthodontist dentists in a dental clinic’s database convenient. Sorting dentists by their specialization ensures that they can be matched with patients based on their specific dental requirements.

Upcoming Appointments with a specific dentist: Sorted by Date and Time:

Worksheet | Query Builder

```
SET PAGESIZE 100
SET LINESIZE 150
TTITLE "Upcoming Appointments with a specific dentist: Sorted by Date and Time:";
COLUMN appointment_id FORMAT 99999
COLUMN description FORMAT A30
COLUMN dentist_id FORMAT 99999
COLUMN patient_id FORMAT 99999
COLUMN datetime FORMAT A20
SELECT DISTINCT *
FROM appointments
WHERE dentist_id = 1
ORDER BY datetime ASC;
```

Script Output | Query Result

Task completed in 0.152 seconds

```
Page:           1


Upcoming Appointments with a specific dentist: Sorted by Date and Time:
APPOINTMENT_ID DESCRIPTION                    DENTIST_ID PATIENT_ID DATETIME
-------------- ------------------------------ ---------- ---------- --------------------
             1 Routine Checkup                         1          1 23-10-05
            11 Orthodontic Checkup                     1         50 23-10-30
            21 Routine Checkup                         1         40 23-11-26
            31 Orthodontic Checkup                     1         30 23-12-26
            41 Routine Checkup                         1         20 24-01-25
```

SET PAGESIZE 100
SET LINESIZE 150
TTITLE "Upcoming Appointments with a specific dentist: Sorted by Date and Time:";
COLUMN appointment_id FORMAT 99999
COLUMN description FORMAT A30
COLUMN dentist_id FORMAT 99999
COLUMN patient_id FORMAT 99999
COLUMN datetime FORMAT A20
SELECT DISTINCT *
FROM appointments
WHERE dentist_id = 1
ORDER BY datetime ASC;

This SQL query will grab unique records from the "appointments" table where the "dentist_id" equals 1. It helps identify distinct appointments scheduled with a specific dentist (dentist_id 1) and arranges them

chronologically by the "datetime" column in ascending order. This query is useful for obtaining a clear, ordered list of appointments associated with a particular dentist.

Adult Patients: Sorted by Name and Date of Birth:

Worksheet | Query Builder

```
SET PAGESIZE 100
SET LINESIZE 150
TTITLE "Adult Patients: Sorted by Name and Date of Birth:";
COLUMN patient_id FORMAT 99999
COLUMN name FORMAT A30
COLUMN date_of_birth FORMAT A13
SELECT DISTINCT patient_id, name, TO_CHAR(date_of_birth, 'YYYY-MM-DD') AS date_of_birth
FROM patients
WHERE date_of_birth > TO_DATE('1990-01-01', 'YYYY-MM-DD')
ORDER BY name ASC;
```

Script Output | Query Result

Task completed in 0.16 seconds

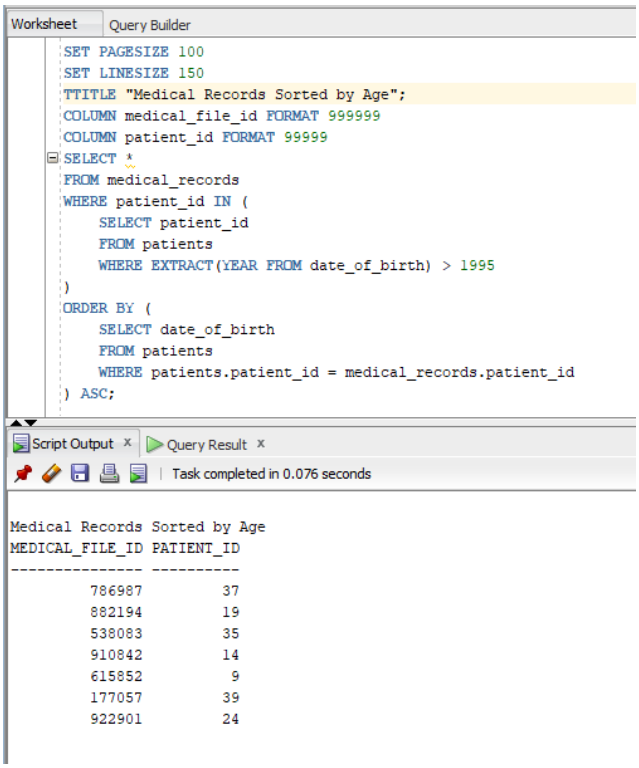
```
Page:          1


Adult Patients: Sorted by Name and Date of Birth:
PATIENT_ID NAME                           DATE_OF_BIRTH
---------- ------------------------------ -------------
        26 Amanda Smith                   1990-11-08
         3 Bob Johnson                    1995-07-10
         9 Chris White                    1998-03-02
        45 Elena Davis                    1995-11-16
        10 Ella Davis                     1995-12-10
        50 Ella Davissss                  1995-12-16
        27 Emily Johnson                  1990-06-20
        48 Emilyyyy Wilson                1995-02-24
        31 Jennifer Lee                   1991-04-15
        46 Jenniferrrr Martin             1990-11-09
         1 John Doe                       1990-01-15
        22 Karen Brown                    1993-09-03
        33 Karen Taylor                   1994-01-25
        12 Laura Martin                   1994-01-19
        35 Linda Clark                    1997-05-05
        16 Linda Johnson                  1991-08-11
        41 Margaret Rodriguez             1992-03-24
        19 Mark Jones                     1996-06-29
        14 Mary Brown                     1997-11-29
        39 Nancy Allen                    1998-08-22
        24 Nancy Johnson                  1999-05-14
        43 Patricia Lewis                 1995-04-03
        29 Sarah Wilson                   1993-09-05
         6 Susan Brown                    1992-04-30
        37 Susan Hernandez                1996-02-28
```

```
SET PAGESIZE 100
SET LINESIZE 150
TTITLE "Adult Patients: Sorted by Name and Date of Birth:";
COLUMN patient_id FORMAT 99999
COLUMN name FORMAT A30
COLUMN date_of_birth FORMAT A13
SELECT DISTINCT patient_id, name, TO_CHAR(date_of_birth, 'YYYY-MM-DD') AS date_of_birth
FROM patients
WHERE date_of_birth > TO_DATE('1990-01-01', 'YYYY-MM-DD')
ORDER BY name ASC;
```

This SQL query retrieves patient information, specifically their ID, name, and date of birth, from the "patients" table. It filters the results to only include patients born after January 1, 1990 and then arranges

them in ascending order based on their names. This query can be altered if we need to find children at a specific age by changing the date of birth. This can be useful for various purposes, such as demographic analysis or to find patients in a certain age group.

Medical Records Sorted by Age:

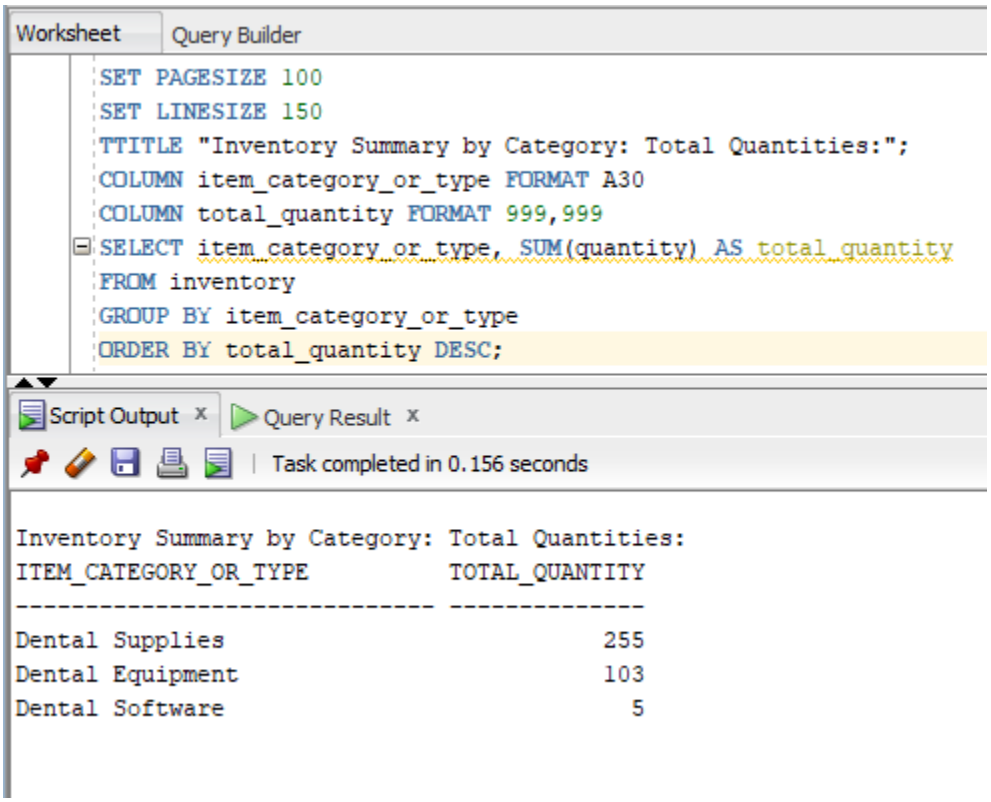



```
SET PAGESIZE 100
SET LINESIZE 150
TTITLE "Medical Records Sorted by Age";
COLUMN medical_file_id FORMAT 999999
COLUMN patient_id FORMAT 99999
SELECT *
FROM medical_records
WHERE patient_id IN (
   SELECT patient_id
   FROM patients
   WHERE EXTRACT(YEAR FROM date_of_birth) > 1995
)
ORDER BY (
   SELECT date_of_birth
   FROM patients
   WHERE patients.patient_id = medical_records.patient_id
) ASC;
```

This SQL query retrieves all records from the "medical_records" table for patients born after 1995, based on their birthdate stored in the "date_of_birth" column in the "patients" table. It uses a subquery to first

identify patient IDs meeting this criteria. The results are then sorted in ascending order based on the birthdates of these patients, providing a chronological listing of medical records for patients born after 1995.

Inventory Summary by Category: Total Quantities:




```
SET PAGESIZE 100
SET LINESIZE 150
TTITLE "Inventory Summary by Category: Total Quantities:";
COLUMN item_category_or_type FORMAT A30
COLUMN total_quantity FORMAT 999,999
SELECT item_category_or_type, SUM(quantity) AS total_quantity
FROM inventory
GROUP BY item_category_or_type
ORDER BY total_quantity DESC;
```

This SQL query retrieves data from the “inventory” table and groups it by the "item_category_or_type." It will then calculate the total quantity of items within each category or type. The results are then ordered in descending order based on the total quantity, providing a list of item categories/types with the highest total quantities at the top.

High-Cost Treatments: Sorted by Descending Cost:

Worksheet | Query Builder

```
SET PAGESIZE 100
SET LINESIZE 150
TTITLE "High-Cost Treatments: Sorted by Descending Cost:";
COLUMN treatment_id FORMAT 99999
COLUMN cost FORMAT 999,999.99
COLUMN description FORMAT A30
COLUMN name_of_treatment FORMAT A30
COLUMN inventory_inventory_id FORMAT 99999
SELECT DISTINCT *
FROM treatments
WHERE cost > 350
ORDER BY cost DESC;
```

Script Output | Query Result

Task completed in 0.209 seconds

```
High-Cost Treatments: Sorted by Descending Cost:
TREATMENT_ID        COST DESCRIPTION                    NAME_OF_TREATMENT              INVENTORY_INVENTORY_ID
------------ ----------- ------------------------------ ------------------------------ -----------------------
           6      700.00 Teeth Whitening Procedure      Teeth Whitening                                     14
           4      600.00 Root Canal Therapy             Root Canal                                          19
           9      550.00 Dental Implant Surgery         Dental Implant                                       1
           2      500.00 Tooth Extraction Procedure     Tooth Extraction                                     9
          10      450.00 Oral Surgery Consultation      Oral Surgery Consultation                           13
           7      400.00 Periodontal Treatment          Periodontal Treatment                                3
```

```
SET PAGESIZE 100
SET LINESIZE 150
TTITLE "High-Cost Treatments: Sorted by Descending Cost:";
COLUMN treatment_id FORMAT 99999
COLUMN cost FORMAT 999,999.99
COLUMN description FORMAT A30
COLUMN name_of_treatment FORMAT A30
COLUMN inventory_inventory_id FORMAT 99999
SELECT DISTINCT *
FROM treatments
WHERE cost > 350
ORDER BY cost DESC;
```

This SQL query retrieves all rows from the "treatments" table where the "cost" of a treatment exceeds $350. It then arranges the results in descending order based on the treatment cost, displaying the most expensive treatments first. Essentially, it provides a list of treatments that cost more than $350 in descending order of their costs.

Unpaid Invoices: Total Costs Grouped by Patients, Highest to Lowest:

Worksheet | Query Builder

```
SET PAGESIZE 100
SET LINESIZE 150
TTITLE "Unpaid Invoices: Total Costs Grouped by Patients, Highest to Lowest:";
COLUMN patient_id FORMAT 99999
COLUMN total_cost FORMAT 999,999.99
SELECT patient_id, SUM(cost) AS total_cost
FROM invoices
WHERE payment_status != 'Paid'
GROUP BY patient_id
ORDER BY total_cost DESC;
```

Script Output | Query Result

Task completed in 0.084 seconds

```
Unpaid Invoices: Total Costs Grouped by Patients, Highest to Lowest:
PATIENT_ID  TOTAL_COST
---------- -----------
        47      700.00
        18      550.00
        24      450.00
         5      350.00
        40      250.00
        14      200.00
```

```
SET PAGESIZE 100
SET LINESIZE 150
TTITLE "Unpaid Invoices: Total Costs Grouped by Patients, Highest to Lowest:";
COLUMN patient_id FORMAT 99999
COLUMN total_cost FORMAT 999,999.99
SELECT patient_id, SUM(cost) AS total_cost
FROM invoices
WHERE payment_status != 'Paid'
GROUP BY patient_id
ORDER BY total_cost DESC;
```

This SQL query retrieves the "patient_id" and the total cost of unpaid invoices from the "invoices" table, grouping the results by patient. It calculates the sum of the costs for each patient's unpaid invoices and assigns it the alias "total_cost." The results are then sorted in descending order based on the total unpaid costs, providing a list of patients with the highest outstanding invoice amounts at the top.