

---

***MID-TERM ASSIGNMENT REPORT***

---

**CBDB4103**

**INTERMEDIATE DATABASE**

**SEPTEMBER 2024**

**HUTECH**

---

**STUDENT'S NAME:** TA DUY THANH TAI

**STUDENT ID:** 2254030103

**CLASS:** 22BOIT02

**EMAIL:** [thanhtai12122004@gmail.com](mailto:thanhtai12122004@gmail.com)

**INSTRUCTOR:** VU THI HANH

1.

```
CREATE TABLE ARTIST (  
    artistName VARCHAR(100) PRIMARY KEY,  
    nationality VARCHAR(50)  
);
```

```
CREATE TABLE LABEL (  
    labelName VARCHAR(100) PRIMARY KEY,  
    revenue DECIMAL(15, 2)  
);
```

```
CREATE TABLE ALBUM (  
    albumTitle VARCHAR(100) PRIMARY KEY,  
    releaseYear NUMBER,  
    producedBy VARCHAR(100),  
    playedBy VARCHAR(100),  
    FOREIGN KEY (producedBy) REFERENCES LABEL(labelName),  
    FOREIGN KEY (playedBy) REFERENCES ARTIST(artistName)  
);
```

```
CREATE TABLE SONG (  
    songTitle VARCHAR(100) PRIMARY KEY,  
    length DATE,  
    writtenBy VARCHAR(100),  
    writtenYear NUMBER,
```

```
FOREIGN KEY (writtenBy) REFERENCES ARTIST(artistName)
);
```

```
CREATE TABLE SONG_INALBUM (
    albumSong VARCHAR(100),
    album VARCHAR(100),
    trackNumber NUMBER,
    PRIMARY KEY (albumSong, album),
    FOREIGN KEY (albumSong) REFERENCES SONG(songTitle),
    FOREIGN KEY (album) REFERENCES ALBUM(albumTitle)
);
```

2.

```
INSERT INTO ARTIST VALUES ('David Louis', 'french');
INSERT INTO ARTIST VALUES ('Baker Switzerland', 'swiss');
INSERT INTO ARTIST VALUES ('Dave Witmuller', 'swiss');
```

```
INSERT INTO LABEL VALUES ('Son', 50000000);
INSERT INTO LABEL VALUES ('ECC', 3000000);
INSERT INTO LABEL VALUES ('LabelBlue', 150000);
INSERT INTO LABEL VALUES ('GROW', 10000);
```

```
INSERT INTO ALBUM VALUES ('Rain', 2008, 'ECC', 'David Louis');
INSERT INTO ALBUM VALUES ('Carney de Roy', 1995, 'LabelBlue', 'David Louis');
INSERT INTO ALBUM VALUES ('WestWest', 2005, 'GROW', 'Baker Switzerland');
```

INSERT INTO SONG VALUES ('After Tomorrow', TO\_DATE('01-JAN-2000 05:35', 'DD-MON-YYYY HH24:MI'), 'David Louis', 2008);

INSERT INTO SONG VALUES ('I Do', TO\_DATE('01-JAN-2000 03:40', 'DD-MON-YYYY HH24:MI'), 'David Louis', 2008);

INSERT INTO SONG VALUES ('Standing', TO\_DATE('01-JAN-2000 04:26', 'DD-MON-YYYY HH24:MI'), 'David Louis', 1995);

INSERT INTO SONG VALUES ('Introducing', TO\_DATE('01-JAN-2000 06:00', 'DD-MON-YYYY HH24:MI'), 'Dave Witmuller', 2000);

INSERT INTO SONG\_INALBUM VALUES ('After Tomorrow', 'Rain', 1);

INSERT INTO SONG\_INALBUM VALUES ('I Do', 'Rain', 9);

INSERT INTO SONG\_INALBUM VALUES ('Standing', 'Carney de Roy', 1);

INSERT INTO SONG\_INALBUM VALUES ('Introducing', 'WestWest', 4);

3.

CREATE TABLE REVIEWS (

magazine VARCHAR2(50),

releaseYear INT,

issue INT,

criticName VARCHAR2(50) NOT NULL,

albumTitle VARCHAR2(50),

rating VARCHAR2(10) CHECK (rating IN ('positive', 'negative', 'neutral')) NOT NULL,

reviewText CLOB NOT NULL,

PRIMARY KEY (magazine, albumTitle),

FOREIGN KEY (albumTitle) REFERENCES ALBUM(albumTitle)

);

4.

```

SELECT album,
        TO_CHAR(SUM(TO_NUMBER(TO_CHAR(length, 'SSSSS'))), 'FM99999') AS
totalSeconds
FROM SONG_INALBUM sa
JOIN SONG s ON sa.albumSong = s.songTitle
GROUP BY album;

```

5.

```

<!DOCTYPE html>

<html>

<head>

    <title>Song Entry Form</title>

</head>

<body>

    <form action="submit_song.php" method="post">

        <label for="songTitle">Song Title:</label>

        <input type="text" id="songTitle" name="songTitle" required><br>

        <label for="length">Length (hh:mm:ss):</label>

        <input type="text" id="length" name="length" required><br>

        <label for="writtenBy">Written By:</label>

        <input type="text" id="writtenBy" name="writtenBy" required><br>

        <label for="writtenYear">Written Year:</label>

        <input type="number" id="writtenYear" name="writtenYear" required><br>

```

```
<input type="submit" value="Submit">

</form>

</body>

</html>
```

6.

a) Checks that each artist has only performed songs they themselves wrote. Here's how it works:

**1. Subquery Logic:**

- The inner query (SELECT a.artistName ... WHERE s.writtenBy != a.artistName) fetches artist names who performed at least one song they did **not** write.
- The condition s.writtenBy != a.artistName ensures we only get artists who played songs written by others.

**2. Main Query Logic:**

- The outer query (SELECT artistName FROM ARTIST WHERE artistName NOT IN (...)) selects artists who are **not** in the result set of the inner query.
- This means that if an artist's name does not appear in the inner result, they have only performed songs they wrote.

b) Identifies artists who have performed on at least one album from every available label. Here's the breakdown:

**1. Nested Subquery Logic:**

- The innermost query (SELECT \* FROM ALBUM ... m.producedBy = l.labelName AND m.playedBy = a.artistName) checks for each album where a particular label (l.labelName) produced it, and the artist (a.artistName) performed on it.
- The middle query (WHERE NOT EXISTS (...)) verifies that this album exists for every label.
- If it finds even one label with no such album, the artist is excluded.

**2. Main Query Logic:**

- The outer query (SELECT artistName FROM ARTIST a WHERE NOT EXISTS (...)) retrieves only the artists who have albums for **every label**.

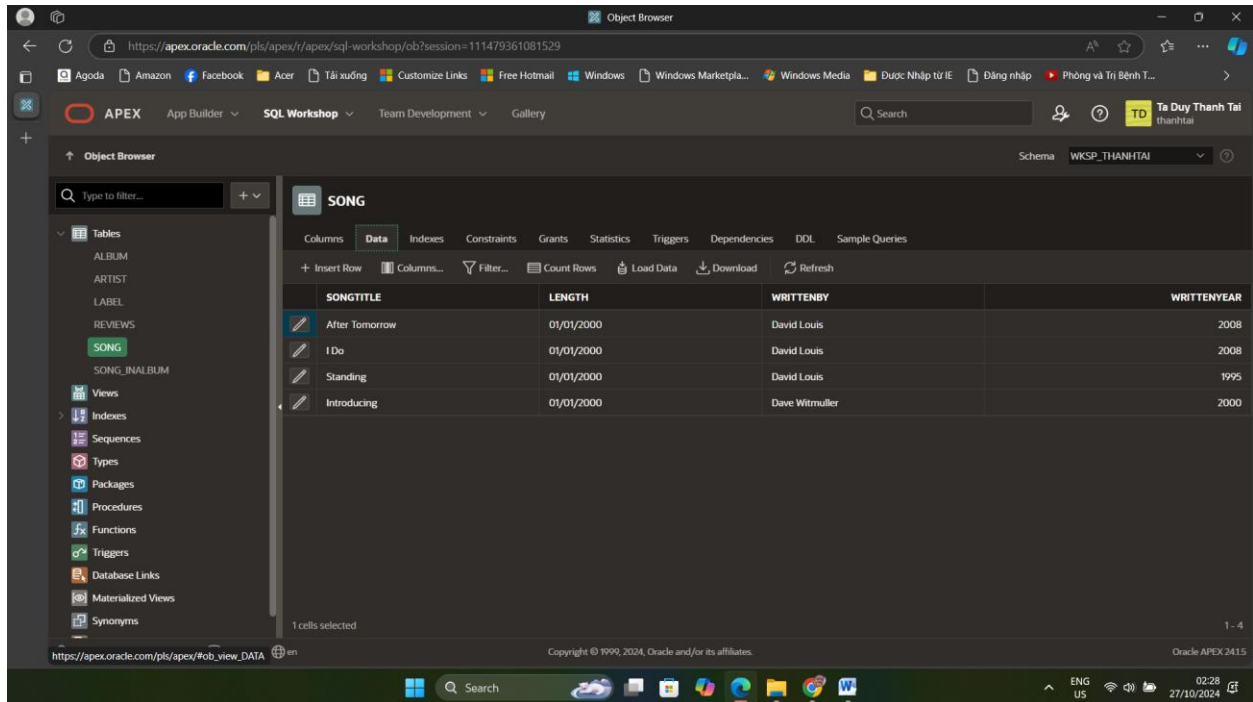
c) Identifies the year in which the sum total of song lengths is the highest.

**1. Inner Query Logic:**

- The innermost subquery (`SELECT SUM(length) AS ttl FROM SONG GROUP BY writtenYear`) groups songs by `writtenYear` and calculates the total length of songs for each year.
- The middle query (`SELECT MAX(ttl) ...`) then finds the maximum sum from the calculated yearly totals.

## 2. Main Query Logic:

- The outer query (`SELECT writtenYear FROM SONG GROUP BY writtenYear HAVING SUM(length) = ...`) groups the data by `writtenYear` and keeps only the year(s) where the sum equals the maximum length.



The screenshot shows the Oracle APEX Object Browser interface. The left sidebar displays a tree view of database objects, with 'SONG' selected under the 'Tables' category. The main panel shows the 'SONG' table data with columns: SONGTITLE, LENGTH, WRITTENBY, and WRITTENYEAR. The table contains four rows of data. The bottom status bar indicates '1 cells selected' and '1 - 4'.

SONGTITLE	LENGTH	WRITTENBY	WRITTENYEAR
After Tomorrow	01/01/2000	David Louis	2008
I Do	01/01/2000	David Louis	2008
Standing	01/01/2000	David Louis	1995
Introducing	01/01/2000	Dave Wittmuller	2000

**Figure 1: Display of Song table in Oracle APEX**

The screenshot shows the Oracle APEX Object Browser interface. The left sidebar displays a tree view of database objects, with 'SONG\_INALBUM' selected under the 'Tables' category. The main panel shows the 'Data' tab for the 'SONG\_INALBUM' table. The table has three columns: 'ALBUMSONG', 'ALBUM', and 'TRACKNUMBER'. The data is as follows:

ALBUMSONG	ALBUM	TRACKNUMBER
After Tomorrow	Rain	1
Introducing	WestWest	4
I Do	Rain	9
Standing	Carney de Roy	1

Figure 2: Display of Song\_inalbum table in Oracle APEX

The screenshot shows the Oracle APEX Object Browser interface. The left sidebar displays a tree view of database objects, with 'ARTIST' selected under the 'Tables' category. The main panel shows the 'Data' tab for the 'ARTIST' table. The table has two columns: 'ARTISTNAME' and 'NATIONALITY'. The data is as follows:

ARTISTNAME	NATIONALITY
Dave Witmuller	swiss
Baker Switzerland	swiss
David Louis	french

Figure 3: Display of Artist table in Oracle APEX



The screenshot shows the Oracle APEX Object Browser interface. The left sidebar lists database objects, with 'LABEL' selected under the 'Tables' category. The main panel displays the 'LABEL' table data in a tabular format. The table has two columns: 'LABELNAME' and 'REVENUE'. The data rows are as follows:

LABELNAME	REVENUE
Son	5000000
LabelBlue	150000
ECC	3000000
GROW	10000

The interface includes a search bar, a schema dropdown set to 'WKSP\_THANHAI', and a bottom status bar showing the user 'thanhai12122004@gmail.com' and the date '27/10/2024'.

Figure 4: Display of Label table in Oracle APEX

The screenshot shows the Oracle APEX Object Browser interface. The left sidebar lists database objects, with 'ALBUM' selected under the 'Tables' category. The main panel displays the 'ALBUM' table data in a tabular format. The table has four columns: 'ALBUMTITLE', 'RELEASEYEAR', 'PRODUCEDBY', and 'PLAYEDBY'. The data rows are as follows:

ALBUMTITLE	RELEASEYEAR	PRODUCEDBY	PLAYEDBY
Carney de Roy	1995	LabelBlue	David Louis
Rain	2008	ECC	David Louis
WestWest	2005	GROW	Baker Switzerland

The interface includes a search bar, a schema dropdown set to 'WKSP\_THANHAI', and a bottom status bar showing the user 'thanhai12122004@gmail.com' and the date '27/10/2024'.

Figure 5: Display of Album table in Oracle APEX

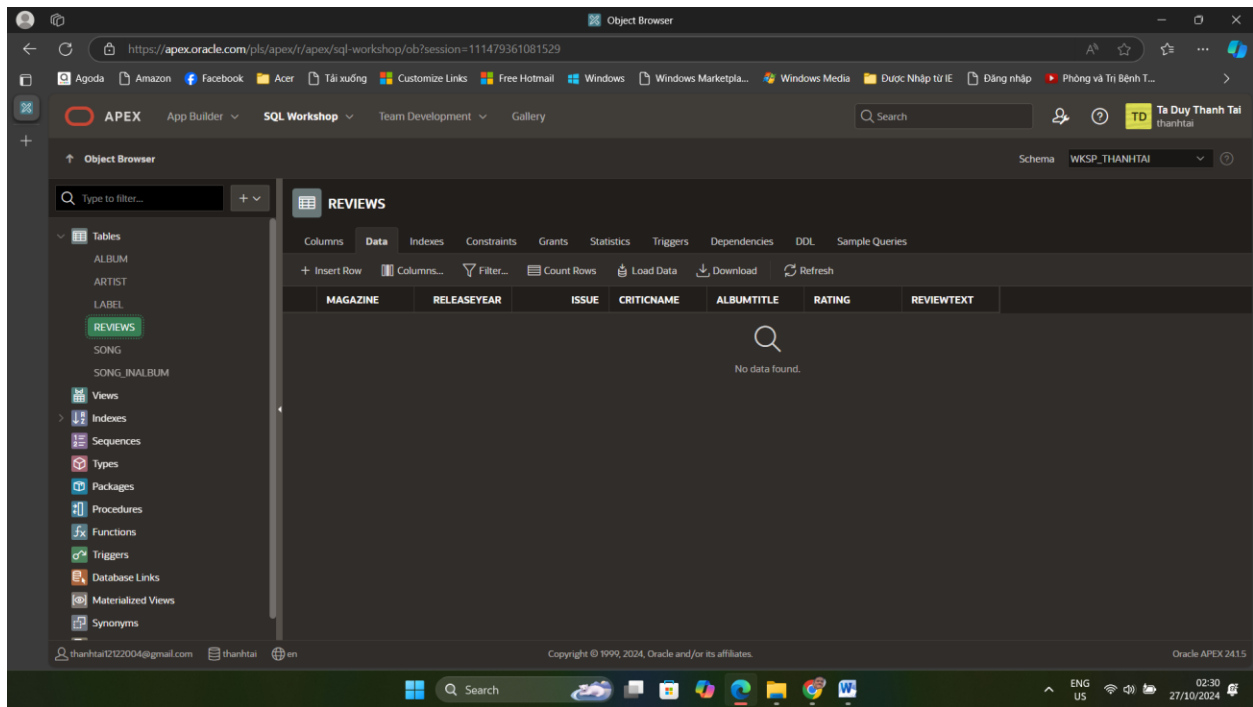


Figure 6: Display of Reviews table in Oracle APEX

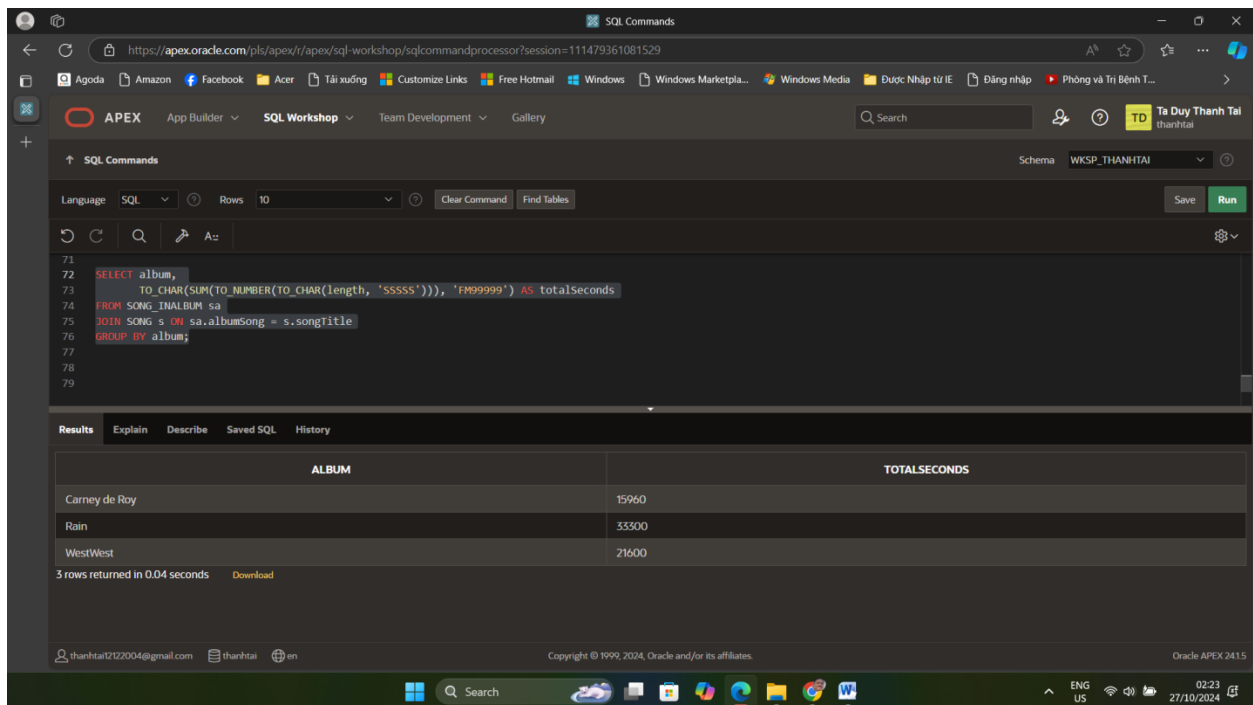


Figure 7: Display of calculation album title and the total length in Oracle APEX

The screenshot displays an Oracle APEX web application interface. The browser's address bar shows the URL: `https://apex.oracle.com/pls/apex/r/tahtai/form?form?session=116265291582914`. The application has a blue header bar with the title "Form" and a user profile icon for "tahtai12122004@gmail.com". A dark sidebar on the left contains a "Home" link and a "Form" link, which is currently selected. The main content area, titled "Form", contains a form with the following fields and values:

Field Name	Value
SONGTITLE	T
Length	2
Writtenby	Tai
Writtenyear	2024

At the bottom of the form are two buttons: "Cancel" and "Submit". The bottom of the browser window shows the Windows taskbar with the search bar, taskbar icons, and system tray information including the date "27/10/2024" and time "03:17".

**Figure 7: Display of Form in Oracle APEX**