DROP TABLE STUDENTINFO PURGE; CREATE TABLE STUDENTINFO(

STUDENT\_ID NUMBER(10) PRIMARY KEY, FIRST\_NAME VARCHAR2(200), LAST\_NAME

VARCHAR2(200), DATE\_OF\_BIRTH DATE, GENDER VARCHAR2(200),

EMAIL VARCHAR2(250) UNIQUE, PHONE\_NUMBER VARCHAR2(10)

);

INSERT INTO STUDENTINFO

VALUES(101,'JOHN','SMITH','15-MAY-1998','M','john.smith@email.com',1234 567890);

INSERT INTO STUDENTINFO

VALUES(102,'EMILY','DAVIS','20-MARCH-1999','M','emily.davis@email.com', 9876543210);

INSERT INTO STUDENTINFO

VALUES(103,'MICHAEL','JOHNSON','10-JULY-1997','M','michael.johnson@emai l.com',5551234567);

INSERT INTO STUDENTINFO

VALUES(104,'SARAH','WILSON','05-JAN-2000','F','sarah.wilson@email.com', 7899876543);

INSERT INTO STUDENTINFO

VALUES(105,'DAVID','BROWN','30-SEP-1996','M','david.brown@email.com',11 12223333);

INSERT INTO STUDENTINFO

VALUES(106,'OLIVIA','LEE','18-DEC-1999','F','olivia.lee@email.com',4445 556666);

INSERT INTO STUDENTINFO

[VALUES(107,'ETHAN','MARTINEZ','25-NOV-1998','M','ethan.martinez@email.c](mailto:VALUES(107%2C%27ETHAN%27%2C%27MARTINEZ%27%2C%2725-NOV-1998%27%2C%27M%27%2C%27ethan.martinez@email.c) om',7778889999);

INSERT INTO STUDENTINFO

[VALUES(108,'SOPHIA','TAYLOR','14-FEB-2002','F','sophia.taylor@email.com](mailto:VALUES(108%2C%27SOPHIA%27%2C%27TAYLOR%27%2C%2714-FEB-2002%27%2C%27F%27%2C%27sophia.taylor@email.com) ',2223334444);

INSERT INTO STUDENTINFO

VALUES(109,'AIDEN','MILLER','12-APR-1997','M','aiden.miller@email.com', 6667778888);

INSERT INTO STUDENTINFO

[VALUES(110,'EMMA','ANDERSON','07-AUG-2002','F','emma.anderson@email.com](mailto:VALUES(110%2C%27EMMA%27%2C%27ANDERSON%27%2C%2707-AUG-2002%27%2C%27F%27%2C%27emma.anderson@email.com) ',3334445555);

INSERT INTO STUDENTINFO

VALUES(111,'BENJAMIN','HARRIS','22-JUN-1995','M','benjamin.harris@email

.com',8889990000); INSERT INTO STUDENTINFO

VALUES(112,'MIA','JOHNSON','01-OCT-1998','F','mia.johnson@email.com',99 90001111);

INSERT INTO STUDENTINFO

[VALUES(113,'WILLIAM','WHITE','12-MARCH-2003','M','willain.white@email.c](mailto:VALUES(113%2C%27WILLIAM%27%2C%27WHITE%27%2C%2712-MARCH-2003%27%2C%27M%27%2C%27willain.white@email.c) om',0001112222);

INSERT INTO STUDENTINFO

VALUES(114,'AVA','ROBINSON','28-APR-1999','F','ava.robinson@email.com', 1112223333);

INSERT INTO STUDENTINFO

VALUES(115,'JAMES','TURNER','05-DEC-1996','M','james.turner@email.com', 2223334444);

1 row(s) inserted.

SELECT \* FROM STUDENTINFO;

STUDENT\_ID FIRST\_NAME LAST\_NAME

DATE\_OF\_BIRTH GENDER EMAIL PHONE\_NUMBER

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| 1. **JOHN** [**john.smith@email.com**](mailto:john.smith@email.com) 2. **EMILY** | **SMITH 1234567890**  **DAVIS** | **15-05-98**  **20-03-99** |  |  | **M**  **M** |
| [**emily.davis@email.com**](mailto:emily.davis@email.com) | **9876543210** |  |  |  |  |
| **103 MICHAEL** | **JOHNSON** | **10-07-97** |  |  | **M** |
| [**michael.johnson@email.com**](mailto:michael.johnson@email.com) | **5551234567** |  |  |  |  |
| **104 SARAH** | **WILSON** | **05-01-00** |  |  | **F** |
| [**sarah.wilson@email.com**](mailto:sarah.wilson@email.com) | **7899876543** |  |  |  |  |
| **105 DAVID** | **BROWN** | **30-09-96** |  |  | **M** |
| [**david.brown@email.com**](mailto:david.brown@email.com) | **1112223333** |  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 1. **OLIVIA** [**olivia.lee@email.com**](mailto:olivia.lee@email.com) 2. **ETHAN** | **LEE**  **4445556666**  **MARTINEZ** | **18-12-99**  **25-11-98** | **F**  **M** |
| [**ethan.martinez@email.com**](mailto:ethan.martinez@email.com) | **7778889999** |  |  |
| **108 SOPHIA** | **TAYLOR** | **14-02-02** | **F** |
| [**sophia.taylor@email.com**](mailto:sophia.taylor@email.com) | **2223334444** |  |  |
| **109 AIDEN** | **MILLER** | **12-04-97** | **M** |
| [**aiden.miller@email.com**](mailto:aiden.miller@email.com) | **6667778888** |  |  |
| **110 EMMA** | **ANDERSON** | **07-08-02** | **F** |
| [**emma.anderson@email.com**](mailto:emma.anderson@email.com) | **3334445555** |  |  |
| **111 BENJAMIN** | **HARRIS** | **22-06-95** | **M** |
| [**benjamin.harris@email.com**](mailto:benjamin.harris@email.com) | **8889990000** |  |  |
| **112 MIA** | **JOHNSON** | **01-10-98** | **F** |
| [**mia.johnson@email.com**](mailto:mia.johnson@email.com) | **9990001111** |  |  |
| **113 WILLIAM** | **WHITE** | **12-03-03** | **M** |
| [**willain.white@email.com**](mailto:willain.white@email.com) | **1112222** |  |  |
| **114 AVA** | **ROBINSON** | **28-04-99** | **F** |
| [**ava.robinson@email.com**](mailto:ava.robinson@email.com) | **1112223333** |  |  |

STUDENT\_ID FIRST\_NAME LAST\_NAME

DATE\_OF\_BIRTH GENDER EMAIL PHONE\_NUMBER

115 JAMES TURNER 05-12-96 M [james.turner@email.com](mailto:james.turner@email.com) 2223334444

15 rows selected.

ASCII and CHR Functions:

1. **Retrieve the student\_id, first\_name, and LAST\_NAME**

from the StudentInfo table. Use the ASCII function to find the ASCII values of the first characters of both first and last names for each student.

SELECT STUDENT\_ID,ASCII(FIRST\_NAME)AS FIRST\_NAME,ASCII(LAST\_NAME

)AS LAST\_NAME FROM STUDENTINFO; STUDENT\_ID FIRST\_NAME LAST\_NAME

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **101** |  | **74** |  | **83** |
| **102** |  | **69** |  | **68** |
| **103** |  | **77** |  | **74** |
| **104** |  | **83** |  | **87** |
| **105** |  | **68** |  | **66** |
| **106** |  | **79** |  | **76** |
| **107** |  | **69** |  | **77** |
| **108** |  | **83** |  | **84** |
| **109** |  | **65** |  | **77** |
| **110** |  | **69** |  | **65** |
| **111** |  | **66** |  | **72** |
| **112** |  | **77** |  | **74** |
| **113** |  | **87** |  | **87** |
| **114**  **STUDENT\_ID** |  | **65**  **FIRST\_NAME** |  | **82**  **LAST\_NAME** |

115 74 84

15 rows selected. SET PAGE SIZE 50;

SELECT

student\_id, email,

ASCII('@') AS ascii\_at FROM StudentInfo;

1. **SELECT STUDENT\_ID,EMAIL,ASCII('@')AS ASCII\_EMAIL FROM STUDENTINFO;**

STUDENT\_ID EMAIL ASCII\_EMAIL

1. [**john.smith@email.com**](mailto:john.smith@email.com) **64**
2. [**emily.davis@email.com**](mailto:emily.davis@email.com) **64**
3. [**michael.johnson@email.com**](mailto:michael.johnson@email.com) **64**
4. [**sarah.wilson@email.com**](mailto:sarah.wilson@email.com) **64**
5. [**david.brown@email.com**](mailto:david.brown@email.com) **64**
6. [**olivia.lee@email.com**](mailto:olivia.lee@email.com) **64**
7. [**ethan.martinez@email.com**](mailto:ethan.martinez@email.com) **64**
8. [**sophia.taylor@email.com**](mailto:sophia.taylor@email.com) **64**
9. [**aiden.miller@email.com**](mailto:aiden.miller@email.com) **64**
10. [**emma.anderson@email.com**](mailto:emma.anderson@email.com) **64**
11. [**benjamin.harris@email.com**](mailto:benjamin.harris@email.com) **64**
12. [**mia.johnson@email.com**](mailto:mia.johnson@email.com) **64**
13. [**willain.white@email.com**](mailto:willain.white@email.com) **64**
14. [**ava.robinson@email.com**](mailto:ava.robinson@email.com) **64**
15. [**james.turner@email.com**](mailto:james.turner@email.com) **64**

15 rows selected.

STUDENT\_ID EMAIL ASCII\_EMAIL

1. [**john.smith@email.com**](mailto:john.smith@email.com) **64**
2. [**emily.davis@email.com**](mailto:emily.davis@email.com) **64**
3. [**michael.johnson@email.com**](mailto:michael.johnson@email.com) **64**
4. [**sarah.wilson@email.com**](mailto:sarah.wilson@email.com) **64**
5. [**david.brown@email.com**](mailto:david.brown@email.com) **64**
6. [**olivia.lee@email.com**](mailto:olivia.lee@email.com) **64**
7. [**ethan.martinez@email.com**](mailto:ethan.martinez@email.com) **64**
8. [**sophia.taylor@email.com**](mailto:sophia.taylor@email.com) **64**
9. [**aiden.miller@email.com**](mailto:aiden.miller@email.com) **64**
10. [**emma.anderson@email.com**](mailto:emma.anderson@email.com) **64**
11. [**benjamin.harris@email.com**](mailto:benjamin.harris@email.com) **64**
12. [**mia.johnson@email.com**](mailto:mia.johnson@email.com) **64**
13. [**willain.white@email.com**](mailto:willain.white@email.com) **64**
14. [**ava.robinson@email.com**](mailto:ava.robinson@email.com) **64**
15. [**james.turner@email.com**](mailto:james.turner@email.com) **64**

15 rows selected.

1. **Display the STUDENT\_ID and first\_name from the StudentInfo table. Use the CHR function to create a new column containing a special character for each student, such as a heart symbol (♥).**

SELECT STUDENT\_ID, FIRST\_NAME , CHR(9829) AS SPECIAL\_CHARACTER FROM STUDENTINFO;

STUDENT\_ID FIRST\_NAME SPECIAL\_CHARACTER

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **101** |  | **JOHN** |  | **&e** |
| **102** |  | **EMILY** |  | **&e** |
| **103** |  | **MICHAEL** |  | **&e** |
| **104** |  | **SARAH** |  | **&e** |
| **105** |  | **DAVID** |  | **&e** |
| **106** |  | **OLIVIA** |  | **&e** |
| **107** |  | **ETHAN** |  | **&e** |
| **108** |  | **SOPHIA** |  | **&e** |
| **109** |  | **AIDEN** |  | **&e** |
| **110** |  | **EMMA** |  | **&e** |
| **111** |  | **BENJAMIN** |  | **&e** |
| **112** |  | **MIA** |  | **&e** |
| **113** |  | **WILLIAM** |  | **&e** |
| **114** |  | **AVA** |  | **&e** |
| **115** |  | **JAMES** |  | **&e** |

15 rows selected.

1. **Calculate the sum of ASCII values for the characters in each students first name. Retrieve the student\_id, first\_name, and the calculated sum using the ASCII function and aggregation.**

SELECT STUDENT\_ID FIRST\_NAME,

|  |  |  |  |
| --- | --- | --- | --- |
| **SUM(ASCII(SUBSTR(FIRST\_NAME,** | **1,** | **1))** | **+** |
| **ASCII(SUBSTR(FIRST\_NAME,** | **2,** | **1))** | **+** |
| **ASCII(SUBSTR(FIRST\_NAME,** | **3,** | **1))** | **+** |
| **ASCII(SUBSTR(FIRST\_NAME,** | **4,** | **1))** | **+** |
| **ASCII(SUBSTR(FIRST\_NAME,** | **5,** | **1))** |  |
| **) AS sum\_of\_ascii\_values** |  |  |  |

FROM StudentInfo GROUP BY STUDENT\_ID, FIRST\_NAME;

STUDENT\_ID FIRST\_NAME SUM\_OF\_ASCII\_VALUES

|  |  |  |  |
| --- | --- | --- | --- |
| **101** |  | **JOHN** |  |
| **102** |  | **EMILY** | **384** |
| **103** |  | **MICHAEL** | **354** |
| **104** |  | **SARAH** | **367** |
| **105** |  | **DAVID** | **360** |
| **106** |  | **OLIVIA** | **387** |
| **107** |  | **ETHAN** | **368** |
| **108** |  | **SOPHIA** | **387** |
| **109** |  | **AIDEN** | **353** |
| **110** |  | **EMMA** |  |
| **111** |  | **BENJAMIN** | **352** |
| **112** |  | **MIA** |  |
| **113** |  | **WILLIAM** | **385** |
| **114** |  | **AVA** |  |
| **115** |  | **JAMES** | **368** |

15 rows selected.

1. **Retrieve the student\_id, LAST\_NAME**

, and the ASCII value of the last character in the last name for each student using the ASCII function.

//SELECT STUDENT\_ID,LAST\_NAME

,ASCII(SUBSTR(RIGHT(LAST\_NAME ,1)))AS ASCII\_VALUE FROM STUDENTINFO;

CONCAT Function:

1. **Create a query that retrieves the student\_id, first\_name, and LAST\_NAME**

from the StudentInfo table. Use the CONCAT function to display the full names in the format "Last Name, First Name."

SELECT STUDENT\_ID, CONCAT(LAST\_NAME

,FIRST\_NAME)AS FULL\_NAME FROM STUDENTINFO; STUDENT\_ID FULL\_NAME

1. **SMITH JOHN**
2. **DAVISEMILY**
3. **JOHNSON MICHAEL**
4. **WILSON SARAH**
5. **BROWN DAVID**
6. **LEE OLIVIA**
7. **MARTINEZ ETHAN**
8. **TAYLOR SOPHIA**
9. **MILLER AIDEN**
10. **ANDERSON MMA**
11. **HARRIS BENJAMIN**
12. **JOHNSON MIA**
13. **WHITE WILLIAM**
14. **ROBINSON AVE**
15. **TURNERJAMES**

15 rows selected.

1. **You want to create email addresses for students based on their first names. Retrieve the student\_id, first\_name, and a new column with email addresses using the CONCAT function. Assume the email domain is '@example.com'.**

SELECT STUDENT\_ID,FIRST\_NAME, CONCAT(FIRST\_NAME,'@EXAMPLE.COM')AS EMAIL\_ADDRESS FROM STUDENTINFO;

STUDENT\_ID FIRST\_NAME EMAIL\_ADDRESS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **101** |  | **JOHN** |  | [**JOHN@example.com**](mailto:JOHN@example.com) |
| **102** |  | **EMILY** |  | [**EMILY@example.com**](mailto:EMILY@example.com) |
| **103** |  | **MICHAEL** |  | [**MICHAEL@example.com**](mailto:MICHAEL@example.com) |
| **104** |  | **SARAH** |  | [**SARAH@example.com**](mailto:SARAH@example.com) |
| **105** |  | **DAVID** |  | [**DAVID@example.com**](mailto:DAVID@example.com) |
| **106** |  | **OLIVIA** |  | [**OLIVIA@example.com**](mailto:OLIVIA@example.com) |
| **107** |  | **ETHAN** |  | [**ETHAN@example.com**](mailto:ETHAN@example.com) |
| **108** |  | **SOPHIA** |  | [**SOPHIA@example.com**](mailto:SOPHIA@example.com) |
| **109** |  | **AIDEN** |  | [**AIDEN@example.com**](mailto:AIDEN@example.com) |
| **110** |  | **EMMA** |  | [**EMMA@example.com**](mailto:EMMA@example.com) |
| **111** |  | **BENJAMIN** |  | [**BENJAMIN@example.com**](mailto:BENJAMIN@example.com) |
| **112** |  | **MIA** |  | [**MIA@example.com**](mailto:MIA@example.com) |
| **113** |  | **WILLIAM** |  | [**WILLIAM@example.com**](mailto:WILLIAM@example.com) |
| **114** |  | **AVA** |  | [**AVA@example.com**](mailto:AVA@example.com) |

115 JAMES [JAMES@example.com](mailto:JAMES@example.com)

15 rows selected.

1. **Display the student\_id, email, and a new email address for each student created by concatenating their student\_id with '@university.com' using the CONCAT function.**

SELECT STUDENT\_ID, EMAIL, CONCAT(STUDENT\_ID, '@university.com') AS

new\_email FROM STUDENT\_ID;

STUDENT\_ID EMAIL NEW\_EMAIL

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **101** |  | [**john.smith@email.com**](mailto:john.smith@email.com) |  | [**101@university.com**](mailto:101@university.com) |
| **102** |  | [**emily.davis@email.com**](mailto:emily.davis@email.com) |  | [**102@university.com**](mailto:102@university.com) |
| **103** |  | [**michael.johnson@email.com**](mailto:michael.johnson@email.com) |  | [**103@university.com**](mailto:103@university.com) |
| **104** |  | [**sarah.wilson@email.com**](mailto:sarah.wilson@email.com) |  | [**104@university.com**](mailto:104@university.com) |
| **105** |  | [**david.brown@email.com**](mailto:david.brown@email.com) |  | [**105@university.com**](mailto:105@university.com) |
| **106** |  | [**olivia.lee@email.com**](mailto:olivia.lee@email.com) |  | [**106@university.com**](mailto:106@university.com) |
| **107** |  | [**ethan.martinez@email.com**](mailto:ethan.martinez@email.com) |  | [**107@university.com**](mailto:107@university.com) |
| **108** |  | [**sophia.taylor@email.com**](mailto:sophia.taylor@email.com) |  | [**108@university.com**](mailto:108@university.com) |
| **109** |  | [**aiden.miller@email.com**](mailto:aiden.miller@email.com) |  | [**109@university.com**](mailto:109@university.com) |
| **110** |  | [**emma.anderson@email.com**](mailto:emma.anderson@email.com) |  | [**110@university.com**](mailto:110@university.com) |
| **111** |  | [**benjamin.harris@email.com**](mailto:benjamin.harris@email.com) |  | [**111@university.com**](mailto:111@university.com) |
| **112** |  | [**mia.johnson@email.com**](mailto:mia.johnson@email.com) |  | [**112@university.com**](mailto:112@university.com) |
| **113** |  | [**willain.white@email.com**](mailto:willain.white@email.com) |  | [**113@university.com**](mailto:113@university.com) |
| **114** |  | [**ava.robinson@email.com**](mailto:ava.robinson@email.com) |  | [**114@university.com**](mailto:114@university.com) |
| **115** |  | [**james.turner@email.com**](mailto:james.turner@email.com) |  | [**115@university.com**](mailto:115@university.com) |

15 rows selected.

1. **Retrieve the student\_id, first\_name, and LAST\_NAME**

from the StudentInfo table. Use the CONCAT function to create a new column displaying the first name followed by the last name without a space.

SELECT STUDENT\_ID, FIRST\_NAME, LAST\_NAME

, CONCAT(FIRST\_NAME, LAST\_NAME ) AS full\_name FROM STUDENT\_ID; STUDENT\_ID FIRST\_NAME LAST\_NAME

FULL\_NAME

|  |  |  |  |
| --- | --- | --- | --- |
| **101** | **JOHN** | **SMITH** | **JOHNSMITH** |
| **102** | **EMILY** | **DAVIS** | **EMILYDAVIS** |
| **103** | **MICHAEL** | **JOHNSON** | **MICHAEL JOHNSON** |
| **104** | **SARAH** | **WILSON** | **SARAH WILSON** |
| **105** | **DAVID** | **BROWN** | **DAVID BROWN** |
| **106** | **OLIVIA** | **LEE** | **OLIVIA LEE** |
| **107** | **ETHAN** | **MARTINEZ** | **ETHAN MARTINEZ** |
| **108** | **SOPHIA** | **TAYLOR** | **SOPHIATAYLOR** |
| **109** | **AIDEN** | **MILLER** | **AIDENMILLER** |
| **110** | **EMMA** | **ANDERSON** | **EMMAANDERSON** |
| **111** | **BENJAMIN** | **HARRIS** | **BENJAMIN HARRIS** |
| **112** | **MIA** | **JOHNSON** | **MIAJOHNSON** |
| **113** | **WILLIAM** | **WHITE** | **WILLIAM WHITE** |
| **114** | **AVA** | **ROBINSON** | **AVAROBINSON** |
| **115** | **JAMES** | **TURNER** | **JAMES TURNER** |

15 rows selected.

1. **You need to generate usernames for students by combining their first names and the last two digits of their student\_id. Retrieve the student\_id, first\_name, and the generated usernames using the CONCAT function.**

SELECT STUDENT\_ID, FIRST\_NAME, CONCAT(FIRST\_NAME, SUBSTR(STUDENT\_ID,

1)) AS username FROM StudentInfo;

STUDENT\_ID FIRST\_NAME USERNAME

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **101** |  | **JOHN** |  | **JOHN01** |
| **102** |  | **EMILY** |  | **EMILY02** |
| **103** |  | **MICHAEL** |  | **MICHAEL03** |
| **104** |  | **SARAH** |  | **SARAH04** |
| **105** |  | **DAVID** |  | **DAVID05** |
| **106** |  | **OLIVIA** |  | **OLIVIA06** |
| **107** |  | **ETHAN** |  | **ETHAN07** |
| **108** |  | **SOPHIA** |  | **SOPHIA08** |
| **109** |  | **AIDEN** |  | **AIDEN09** |
| **110** |  | **EMMA** |  | **EMMA10** |
| **111** |  | **BENJAMIN** |  | **BENJAMIN11** |

|  |  |  |
| --- | --- | --- |
| **112** | **MIA** | **MIA12** |
| **113** | **WILLIAM** | **WILLIAM13** |
| **114** | **AVA** | **AVA14** |
| **115** | **JAMES** | **JAMES15** |

15 rows selected.

LOWER and UPPER Functions:

1. **Display the student\_id and email from the StudentInfo table. Convert the email addresses to lowercase using the LOWER function.**

SELECT STUDENT\_ID, LOWER(EMAIL) AS lower\_email FROM STUDENTINFO;

STUDENT\_ID LOWER\_EMAIL

1. [**john.smith@email.com**](mailto:john.smith@email.com)
2. [**emily.davis@email.com**](mailto:emily.davis@email.com)
3. [**michael.johnson@email.com**](mailto:michael.johnson@email.com)
4. [**sarah.wilson@email.com**](mailto:sarah.wilson@email.com)
5. [**david.brown@email.com**](mailto:david.brown@email.com)
6. [**olivia.lee@email.com**](mailto:olivia.lee@email.com)
7. [**ethan.martinez@email.com**](mailto:ethan.martinez@email.com)
8. [**sophia.taylor@email.com**](mailto:sophia.taylor@email.com)
9. [**aiden.miller@email.com**](mailto:aiden.miller@email.com)
10. [**emma.anderson@email.com**](mailto:emma.anderson@email.com)
11. [**benjamin.harris@email.com**](mailto:benjamin.harris@email.com)
12. [**mia.johnson@email.com**](mailto:mia.johnson@email.com)
13. [**willain.white@email.com**](mailto:willain.white@email.com)
14. [**ava.robinson@email.com**](mailto:ava.robinson@email.com)
15. [**james.turner@email.com**](mailto:james.turner@email.com)

15 rows selected.

1. **Retrieve the student\_id, first\_name, and LAST\_NAME**

from the StudentInfo table. Use the UPPER function to display the full names in uppercase.

SELECT STUDENT\_ID, UPPER(CONCAT(FIRST\_NAME, LAST\_NAME

)) AS full\_name\_in\_uppercase FROM StudentInfo;

STUDENT\_ID FULL\_NAME\_IN\_UPPERCASE

1. **JOHNSMITH**
2. **EMILY DAVIS**
3. **MICHAEL JOHNSON**
4. **SARAH WILSON**
5. **DAVID BROWN**
6. **OLIVIA LEE**
7. **ETHAN MARTINEZ**
8. **SOPHIA TAYLOR**
9. **AIDEN MILLER**
10. **EMMA ANDERSON**
11. **BENJAMIN HARRIS**
12. **MIA JOHNSON**
13. **WILLIAM WHITE**
14. **AVA ROBINSON**
15. **JAMES TURNER**

15 rows selected.

1. **Calculate the total number of students with lowercase email addresses in the StudentInfo table using the LOWER function and COUNT aggregation.**

SELECT COUNT(\*) AS total\_students\_lowercase\_emails FROM STUDENTINFO WHERE EMAIL = LOWER(EMAIL);

TOTAL\_STUDENTS\_LOWERCASE\_EMAILS

15

1. **Retrieve the student\_id, email, and first\_name. Convert the email addresses to uppercase and display them alongside the original first names using the UPPER function.**

SELECT STUDENT\_ID, UPPER(EMAIL) AS upper\_email, FIRST\_NAME FROM STUDENTINFO;

STUDENT\_ID UPPER\_EMAIL FIRST\_NAME

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **101** |  | [**JOHN.SMITH@EMAIL.COM**](mailto:JOHN.SMITH@EMAIL.COM) |  | **JOHN** |
| **102** |  | [**EMILY.DAVIS@EMAIL.COM**](mailto:EMILY.DAVIS@EMAIL.COM) |  | **EMILY** |
| **103** |  | [**MICHAEL.JOHNSON@EMAIL.COM**](mailto:MICHAEL.JOHNSON@EMAIL.COM) |  | **MICHAEL** |
| **104** |  | [**SARAH.WILSON@EMAIL.COM**](mailto:SARAH.WILSON@EMAIL.COM) |  | **SARAH** |
| **105** |  | [**DAVID.BROWN@EMAIL.COM**](mailto:DAVID.BROWN@EMAIL.COM) |  | **DAVID** |
| **106** |  | [**OLIVIA.LEE@EMAIL.COM**](mailto:OLIVIA.LEE@EMAIL.COM) |  | **OLIVIA** |
| **107** |  | [**ETHAN.MARTINEZ@EMAIL.COM**](mailto:ETHAN.MARTINEZ@EMAIL.COM) |  | **ETHAN** |
| **108** |  | [**SOPHIA.TAYLOR@EMAIL.COM**](mailto:SOPHIA.TAYLOR@EMAIL.COM) |  | **SOPHIA** |
| **109** |  | [**AIDEN.MILLER@EMAIL.COM**](mailto:AIDEN.MILLER@EMAIL.COM) |  | **AIDEN** |
| **110** |  | [**EMMA.ANDERSON@EMAIL.COM**](mailto:EMMA.ANDERSON@EMAIL.COM) |  | **EMMA** |
| **111** |  | [**BENJAMIN.HARRIS@EMAIL.COM**](mailto:BENJAMIN.HARRIS@EMAIL.COM) |  | **BENJAMIN** |
| **112** |  | [**MIA.JOHNSON@EMAIL.COM**](mailto:MIA.JOHNSON@EMAIL.COM) |  | **MIA** |
| **113** |  | [**WILLAIN.WHITE@EMAIL.COM**](mailto:WILLAIN.WHITE@EMAIL.COM) |  | **WILLIAM** |
| **114** |  | [**AVA.ROBINSON@EMAIL.COM**](mailto:AVA.ROBINSON@EMAIL.COM) |  | **AVA** |
| **115** |  | [**JAMES.TURNER@EMAIL.COM**](mailto:JAMES.TURNER@EMAIL.COM) |  | **JAMES** |

15 rows selected.

1. **You want to display the student\_id, email, and LAST\_NAME**

from the StudentInfo table. Convert the email addresses to uppercase and remove any leading and trailing spaces using the UPPER function and TRIM function.

SELECT STUDENT\_ID, UPPER(TRIM(EMAIL)) AS upper\_email, LAST\_NAME FROM STUDENTINFO;

STUDENT\_ID UPPER\_EMAIL LAST\_NAME

|  |  |  |  |
| --- | --- | --- | --- |
| **101** | [**JOHN.SMITH@EMAIL.COM**](mailto:JOHN.SMITH@EMAIL.COM) |  | **SMITH** |
| **102** | [**EMILY.DAVIS@EMAIL.COM**](mailto:EMILY.DAVIS@EMAIL.COM) |  | **DAVIS** |
| **103** | [**MICHAEL.JOHNSON@EMAIL.COM**](mailto:MICHAEL.JOHNSON@EMAIL.COM) |  | **JOHNSON** |
| **104** | [**SARAH.WILSON@EMAIL.COM**](mailto:SARAH.WILSON@EMAIL.COM) |  | **WILSON** |
| **105** | [**DAVID.BROWN@EMAIL.COM**](mailto:DAVID.BROWN@EMAIL.COM) |  | **BROWN** |
| **106** | [**OLIVIA.LEE@EMAIL.COM**](mailto:OLIVIA.LEE@EMAIL.COM) |  | **LEE** |
| **107** | [**ETHAN.MARTINEZ@EMAIL.COM**](mailto:ETHAN.MARTINEZ@EMAIL.COM) |  | **MARTINEZ** |
| **108** | [**SOPHIA.TAYLOR@EMAIL.COM**](mailto:SOPHIA.TAYLOR@EMAIL.COM) |  | **TAYLOR** |

|  |  |  |
| --- | --- | --- |
| **109** | [**AIDEN.MILLER@EMAIL.COM**](mailto:AIDEN.MILLER@EMAIL.COM) | **MILLER** |
| **110** | [**EMMA.ANDERSON@EMAIL.COM**](mailto:EMMA.ANDERSON@EMAIL.COM) | **ANDERSON** |
| **111** | [**BENJAMIN.HARRIS@EMAIL.COM**](mailto:BENJAMIN.HARRIS@EMAIL.COM) | **HARRIS** |
| **112** | [**MIA.JOHNSON@EMAIL.COM**](mailto:MIA.JOHNSON@EMAIL.COM) | **JOHNSON** |
| **113** | [**WILLAIN.WHITE@EMAIL.COM**](mailto:WILLAIN.WHITE@EMAIL.COM) | **WHITE** |
| **114** | [**AVA.ROBINSON@EMAIL.COM**](mailto:AVA.ROBINSON@EMAIL.COM) | **ROBINSON** |
| **115** | [**JAMES.TURNER@EMAIL.COM**](mailto:JAMES.TURNER@EMAIL.COM) | **TURNER** |

15 rows selected.

COUNT, AVG, MAX, MEDIAN, MIN, and SUM Functions:-

1. **Calculate the total count of students in the STUDENTINFO table.**

SELECT COUNT(\*) AS total\_students FROM STUDENTINFO; TOTAL\_STUDENTS

15

1. **Determine the average age of students based on their date of birth and display it.**

SELECT AVG(TRUNC(MONTHS\_BETWEEN(SYSDATE, date\_of\_birth) / 12)) AS

average\_age FROM STUDENTINFO;

AVERAGE\_AGE

24.2

1. **Find the maximum and minimum lengths of students email addresses and display these values.**

SELECT MAX(LENGTH(EMAIL)) AS max\_email\_length, MIN(LENGTH(EMAIL)) AS

min\_email\_length FROM STUDENTINFO;

MAX\_EMAIL\_LENGTH MIN\_EMAIL\_LENGTH

25 20

1. **Determine the sum of ASCII values of the first character of each student s last name and display the result.**

SELECT SUM(ASCII(SUBSTR(LAST\_NAME, 1, 1))) AS sum\_of\_ascii\_values FROM STUDENTINFO;

SUM\_OF\_ASCII\_VALUES

1156

TRIM Function:

* 1. **You have a column named DESCRIPTION in a table that contains text data. You want to remove any leading and trailing spaces from the values in this column. Write an SQL query using the TRIM function to achieve this.**

UPDATE STUDENTINFO SET DESCRIPTION = TRIM(BOTH FROM DESCRIPTION); SELECT TRIM('DESCRIPTION' ) AS PRODUCT\_NAME FROM STUDENTINFO;

PRODUCT\_NAME

DESCRIPTION DESCRIPTION DESCRIPTION DESCRIPTION DESCRIPTION DESCRIPTION DESCRIPTION DESCRIPTION DESCRIPTION DESCRIPTION DESCRIPTION

DESCRIPTION DESCRIPTION DESCRIPTION

PRODUCT\_NAME DESCRIPTION

15 rows selected.

* 1. **Retrieve the names of all students in the STUDENTINFO table. Some names have extra spaces at the beginning and end. Write an SQL query using the TRIM function to display the names without leading and trailing spaces.**

SELECT TRIM() AS NAME FROM STUDENTINFO;

* 1. **In a table that phone\_number, you notice that some phone\_number have unnecessary spaces. Write an SQL query using the TRIM function to remove all leading and trailing spaces from the phone\_number.**

UPDATE STUDENTINFO SET PHONE\_NUMBER = TRIM(BOTH ' ' FROM PHONE\_NUMBER);

15 rows updated.

* 1. **You need to list all gender from a table of STUDENTINFO. However, gender have leading spaces. Write an SQL query using the TRIM function to display the gender without any leading spaces.**

SELECT TRIM(LEADING ' ' FROM GENDER) AS GENDER FROM STUDENTINFO;

GENDER

M M M F M

F M F M F M F M F

GENDER

M

15 rows selected.

* 1. **Retrieve a list of email addresses from the STUDENTINFO table. Some email addresses have extra spaces in them. Use the TRIM function to remove any leading and trailing spaces from the email addresses in your query.**

SELECT TRIM(BOTH ' ' FROM EMAIL) AS EMAIL FROM STUDENTINFO;

EMAIL

[john.smith@email.com](mailto:john.smith@email.com) [emily.davis@email.com](mailto:emily.davis@email.com) [michael.johnson@email.com](mailto:michael.johnson@email.com) [sarah.wilson@email.com](mailto:sarah.wilson@email.com) [david.brown@email.com](mailto:david.brown@email.com) [olivia.lee@email.com](mailto:olivia.lee@email.com) [ethan.martinez@email.com](mailto:ethan.martinez@email.com) [sophia.taylor@email.com](mailto:sophia.taylor@email.com) [aiden.miller@email.com](mailto:aiden.miller@email.com) [emma.anderson@email.com](mailto:emma.anderson@email.com) [benjamin.harris@email.com](mailto:benjamin.harris@email.com) [mia.johnson@email.com](mailto:mia.johnson@email.com) [willain.white@email.com](mailto:willain.white@email.com) [ava.robinson@email.com](mailto:ava.robinson@email.com)

EMAIL

[james.turner@email.com](mailto:james.turner@email.com)

15 rows selected.

LTRIM Function:

1. **You have a column called STUDENT\_ID in a table where some values have extra spaces at the beginning. Write an SQL query using the LTRIM function to remove leading spaces from the STUDENT\_ID.**

SELECT LTRIM(STUDENT\_ID) AS StudentID FROM STUDENTINFO;

STUDENTID

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

15 rows selected.

1. **In a table containing date\_of\_birth, some date\_of\_birth have leading spaces. Write an SQL query using the LTRIM function to display the date\_of\_birth without any leading spaces.**

SELECT LTRIM(DATE\_OF\_BIRTH) AS DateOfBirth FROM STUDENTINFO;

DATE OF BIRTH 15-05-98

20-03-99

10-07-97

05-01-00

30-09-96

18-12-99

25-11-98

14-02-02

12-04-97

07-08-02

22-06-95

01-10-98

12-03-03

28-04-99

05-12-96

15 rows selected.

1. **Retrieve a list of first\_name from a table. Some first\_name have leading spaces. Use the LTRIM function to remove these leading spaces in your query.**

SELECT LTRIM(FIRST\_NAME) AS FIRST\_NAME FROM STUDENTINFO;

FIRSTNAME JOHN

EMILY MICHAEL SARAH

DAVID OLIVIA ETHAN SOPHIA AIDEN EMMA BENJAMIN MIA WILLIAM AVA JAMES

15 rows selected.

1. **You are working with data from a STUDENTINFO, and the STUDENT\_IDs sometimes have extra spaces at the beginning. Write an SQL query using the LTRIM function to remove any leading spaces from the student\_ IDs.**

SELECT LTRIM(STUDENT\_ID) AS StudentID FROM STUDENTINFO;

STUDENTID

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

15 rows selected.

1. **In a table that LAST\_NAME some names have leading spaces that need to be removed. Write an SQL query using the LTRIM function to clean the LAST\_NAME .**

SELECT LTRIM(LAST\_NAME) AS LastName FROM STUDENTINFO;

LAST NAME SMITH

DAVIS JOHNSON WILSON BROWN LEE MARTINEZ TAYLOR MILLER ANDERSON HARRIS JOHNSON WHITE ROBINSON TURNER

15 rows selected.

RTRIM Function:

1. **You are dealing with a table that contains email\_id, and some of them have trailing spaces. Write an SQL query using the RTRIM function to remove any trailing spaces from the email\_id.**

SELECT RTRIM(EMAIL) AS Email FROM STUDENTINFO;

EMAIL

[john.smith@email.com](mailto:john.smith@email.com) [emily.davis@email.com](mailto:emily.davis@email.com) [michael.johnson@email.com](mailto:michael.johnson@email.com) [sarah.wilson@email.com](mailto:sarah.wilson@email.com) [david.brown@email.com](mailto:david.brown@email.com) [olivia.lee@email.com](mailto:olivia.lee@email.com) [ethan.martinez@email.com](mailto:ethan.martinez@email.com) [sophia.taylor@email.com](mailto:sophia.taylor@email.com) [aiden.miller@email.com](mailto:aiden.miller@email.com) [emma.anderson@email.com](mailto:emma.anderson@email.com) [benjamin.harris@email.com](mailto:benjamin.harris@email.com) [mia.johnson@email.com](mailto:mia.johnson@email.com) [willain.white@email.com](mailto:willain.white@email.com) [ava.robinson@email.com](mailto:ava.robinson@email.com) [james.turner@email.com](mailto:james.turner@email.com)

15 rows selected.

1. **In a table that date\_of\_birth, some date have trailing spaces that need to be eliminated. Write an SQL query using the RTRIM function to display the date\_of\_birth without trailing spaces.**

SELECT RTRIM(DATE\_OF\_BIRTH) AS DateOfBirth FROM STUDENTINFO;

DATE OF BIRTH 15-05-98

20-03-99

10-07-97

05-01-00

30-09-96

18-12-99

25-11-98

14-02-02

12-04-97

07-08-02

22-06-95

01-10-98

12-03-03

28-04-99

05-12-96

15 rows selected.

1. **Retrieve a list of first\_name from a table. Some first names have trailing spaces. Use the RTRIM function to remove these trailing spaces in your query.**

SELECT RTRIM(FIRSTNAME) AS FIRSTNAME FROM STUDENTINFO;

FIRSTNAME JOHN

EMILY MICHAEL SARAH DAVID OLIVIA ETHAN SOPHIA AIDEN EMMA BENJAMIN MIA WILLIAM AVA JAMES

15 rows selected.

1. **You have a table with Students names, and some names have trailing spaces. Write an SQL query using the RTRIM function to display the student s names without any trailing spaces.**

SELECT RTRIM(FIRST\_NAME,LAST\_NAME) AS STUDENTNAME FROM STUDENTINFO;

STUDENTNAME JOHN

EMILY MICHAEL SARAH DAVID OLIVIA ETH SOPHI AIDEN EMM BENJAMIN MIA WILLIAM AVA JAMES

15 rows selected.

1. **In a table containing phone number, some number have trailing spaces that need to be cleaned. Write an SQL query using the RTRIM function to remove the trailing spaces from the phone number.**

SELECT RTRIM(PHONE\_NUMBER) AS PhoneNumber FROM STUDENTINFO;

PHONE NUMBER 1234567890

9876543210

5551234567

7899876543

1112223333

4445556666

7778889999

2223334444

6667778888

3334445555

8889990000

9990001111

1112222

1112223333

2223334444

15 rows selected.

TRIM Function:-

1. **You are dealing with a table that contains Student’s first name. Some first\_names have both leading and trailing spaces. Write an SQL query using the TRIM function to retrieve the STUDENT\_ID and cleaned first\_names for all students.**

SELECT STUDENT\_ID, TRIM(BOTH ' ' FROM FIRST\_NAME) AS FirstName FROM STUDENTINFO;

STUDENT\_ID FIRST NAME

|  |  |  |
| --- | --- | --- |
| **101** |  | **JOHN** |
| **102** |  | **EMILY** |
| **103** |  | **MICHAEL** |
| **104** |  | **SARAH** |
| **105** |  | **DAVID** |
| **106** |  | **OLIVIA** |
| **107** |  | **ETHAN** |
| **108** |  | **SOPHIA** |
| **109** |  | **AIDEN** |
| **110** |  | **EMMA** |
| **111** |  | **BENJAMIN** |
| **112** |  | **MIA** |
| **113** |  | **WILLIAM** |
| **114** |  | **AVA** |
| **115** |  | **JAMES** |

15 rows selected.

1. **In a table of last names, you notice that some names have both leading and trailing spaces. Write an SQL query using the TRIM function to display the last names without leading and trailing spaces.**

SELECT TRIM(BOTH ' ' FROM LAST\_NAME) AS LastName FROM STUDENTINFO;

LAST NAME SMITH

DAVIS JOHNSON WILSON BROWN LEE MARTINEZ TAYLOR MILLER ANDERSON HARRIS JOHNSON WHITE ROBINSON TURNER

15 rows selected.

1. **Retrieve a list of Student id from the STUDENTINFO table. Some Student id have both leading and trailing spaces. Use the TRIM function to clean the student id in your query.**

SELECT TRIM(BOTH ' ' FROM STUDENT\_ID) AS StudentID FROM STUDENTINFO;

STUDENTID

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

15 rows selected.

1. **In a table containing phone numbers, some numbers have both leading and trailing spaces that need to be removed. Write an SQL query using the TRIM function to clean the phone number.**

SELECT TRIM(BOTH ' ' FROM PHONE\_NUMBER) AS PhoneNumber FROM STUDENTINFO;

PHONE NUMBER 1234567890

9876543210

5551234567

7899876543

1112223333

4445556666

7778889999

2223334444

6667778888

3334445555

8889990000

9990001111

1112222

1112223333

2223334444

15 rows selected.

1. **You have a table with Student names, and some names have both leading and trailing spaces. Write an SQL query using the TRIM function to display the Student names without any leading and trailing spaces.**

SUBSTR Function:

1. **Retrieve the first three characters of each student's first name from the STUDENTINFO table using the SUBSTR function. Display the STUDENT\_ID and the extracted substrings.**

SELECT STUDENT\_ID, SUBSTR(FIRST\_NAME, 1, 4) AS Extracted\_Substring FROM STUDENTINFO;

STUDENT\_ID EXTRACTED\_SUBSTRING

|  |  |  |
| --- | --- | --- |
| **101** |  | **JOH** |
| **102** |  | **EMI** |
| **103** |  | **MIC** |
| **104** |  | **SAR** |
| **105** |  | **DAV** |
| **106** |  | **OLI** |
| **107** |  | **ETH** |
| **108** |  | **SOP** |
| **109** |  | **AID** |
| **110** |  | **EMM** |
| **111** |  | **BEN** |
| **112** |  | **MIA** |
| **113** |  | **WIL** |
| **114** |  | **AVA** |
| **115** |  | **JAM** |

15 rows selected.

1. **You need to extract the last two characters from each student's last name. Write an SQL query using the SUBSTR function to retrieve the STUDENT\_ID, LAST\_NAME and the extracted substrings.**

SELECT STUDENT\_ID, LAST\_NAME, SUBSTR(LAST\_NAME , LENGTH(LAST\_NAME ) -

1, 2) AS ExtractedSubstring FROM STUDENTINFO;

STUDENT\_ID LAST\_NAME EXTRACT SUBSTRING

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **101** |  | **SMITH** |  | **TH** |
| **102** |  | **DAVIS** |  | **IS** |
| **103** |  | **JOHNSON** |  | **ON** |
| **104** |  | **WILSON** |  | **ON** |
| **105** |  | **BROWN** |  | **WN** |
| **106** |  | **LEE** |  | **EE** |
| **107** |  | **MARTINEZ** |  | **EZ** |
| **108** |  | **TAYLOR** |  | **OR** |
| **109** |  | **MILLER** |  | **ER** |
| **110** |  | **ANDERSON** |  | **ON** |
| **111** |  | **HARRIS** |  | **IS** |
| **112** |  | **JOHNSON** |  | **ON** |
| **113** |  | **WHITE** |  | **TE** |
| **114** |  | **ROBINSON** |  | **ON** |
| **115** |  | **TURNER** |  | **ER** |

15 rows selected.

1. **Display the STUDENT\_ID, email, and only the domain part of each student's email address using the SUBSTR function. Assume that the domain is the character after the '@' symbol.**

SELECT STUDENT\_ID, EMAIL, SUBSTR(EMAIL, INSTR(EMAIL, '@') + 1) AS Domain FROM STUDENTINFO;

STUDENT\_ID EMAIL DOMAIN

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **101** |  | [**john.smith@email.com**](mailto:john.smith@email.com) |  | **email.com** |
| **102** |  | [**emily.davis@email.com**](mailto:emily.davis@email.com) |  | **email.com** |
| **103** |  | [**michael.johnson@email.com**](mailto:michael.johnson@email.com) |  | **email.com** |
| **104** |  | [**sarah.wilson@email.com**](mailto:sarah.wilson@email.com) |  | **email.com** |
| **105** |  | [**david.brown@email.com**](mailto:david.brown@email.com) |  | **email.com** |
| **106** |  | [**olivia.lee@email.com**](mailto:olivia.lee@email.com) |  | **email.com** |
| **107** |  | [**ethan.martinez@email.com**](mailto:ethan.martinez@email.com) |  | **email.com** |
| **108** |  | [**sophia.taylor@email.com**](mailto:sophia.taylor@email.com) |  | **email.com** |
| **109** |  | [**aiden.miller@email.com**](mailto:aiden.miller@email.com) |  | **email.com** |
| **110** |  | [**emma.anderson@email.com**](mailto:emma.anderson@email.com) |  | **email.com** |

|  |  |  |
| --- | --- | --- |
| **111** | [**benjamin.harris@email.com**](mailto:benjamin.harris@email.com) | **email.com** |
| **112** | [**mia.johnson@email.com**](mailto:mia.johnson@email.com) | **email.com** |
| **113** | [**willain.white@email.com**](mailto:willain.white@email.com) | **email.com** |
| **114** | [**ava.robinson@email.com**](mailto:ava.robinson@email.com) | **email.com** |
| **115** | [**james.turner@email.com**](mailto:james.turner@email.com) | **email.com** |

15 rows selected.

1. **Calculate the length of the first five characters in each student's email address. Retrieve the STUDENT\_ID, email, and the length of the substrings using the SUBSTR and LENGTH functions.**

SELECT STUDENT\_ID, EMAIL, LENGTH(SUBSTR(EMAIL, 1, 5)) AS

SubstringLength FROM STUDENTINFO;

STUDENT\_ID EMAIL SUBSTRING LENGTH

1. [**john.smith@email.com**](mailto:john.smith@email.com) **5**
2. [**emily.davis@email.com**](mailto:emily.davis@email.com) **5**
3. [**michael.johnson@email.com**](mailto:michael.johnson@email.com) **5**
4. [**sarah.wilson@email.com**](mailto:sarah.wilson@email.com) **5**
5. [**david.brown@email.com**](mailto:david.brown@email.com) **5**
6. [**olivia.lee@email.com**](mailto:olivia.lee@email.com) **5**
7. [**ethan.martinez@email.com**](mailto:ethan.martinez@email.com) **5**
8. [**sophia.taylor@email.com**](mailto:sophia.taylor@email.com) **5**
9. [**aiden.miller@email.com**](mailto:aiden.miller@email.com) **5**
10. [**emma.anderson@email.com**](mailto:emma.anderson@email.com) **5**
11. [**benjamin.harris@email.com**](mailto:benjamin.harris@email.com) **5**
12. [**mia.johnson@email.com**](mailto:mia.johnson@email.com) **5**
13. [**willain.white@email.com**](mailto:willain.white@email.com) **5**
14. [**ava.robinson@email.com**](mailto:ava.robinson@email.com) **5**
15. [**james.turner@email.com**](mailto:james.turner@email.com) **5**

15 rows selected.

1. **Retrieve the STUDENT\_ID, first\_name, and the third to fifth characters of each students first name using the SUBSTR function.**

SELECT STUDENT\_ID, first\_name, SUBSTR(first\_name, 3, 3) AS ExtractedSubstring FROM STUDENTINFO;

STUDENT\_ID FIRST\_NAME EXTRACT SUBSTRING

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **101** |  | **JOHN** |  | **HN** |
| **102** |  | **EMILY** |  | **ILY** |
| **103** |  | **MICHAEL** |  | **CHA** |
| **104** |  | **SARAH** |  | **RAH** |
| **105** |  | **DAVID** |  | **VID** |
| **106** |  | **OLIVIA** |  | **IVI** |
| **107** |  | **ETHAN** |  | **HAN** |
| **108** |  | **SOPHIA** |  | **PHI** |
| **109** |  | **AIDEN** |  | **DEN** |
| **110** |  | **EMMA** |  | **MA** |
| **111** |  | **BENJAMIN** |  | **NJA** |
| **112** |  | **MIA** |  | **A** |
| **113** |  | **WILLIAM** |  | **LLI** |
| **114** |  | **AVA** |  | **A** |
| **115** |  | **JAMES** |  | **MES** |

15 rows selected.

---

NVL Function:-

1. In the STUDENTINFO table, some students have missing values for their phone numbers (NULL). Use the NVL function to display 'Not Available' for students with no phone number. Retrieve the STUDENT\_ID, first\_name, and phone numbers.

SELECT STUDENT\_ID, FIRST\_NAME, NVL(phone\_number, 'Not Available') AS Phone FROM STUDENTINFO;

STUDENT\_ID FIRST\_NAME PHONE

|  |  |  |
| --- | --- | --- |
| **101 JOHN** | **1234567890** |  |
| **102 EMILY** | **9876543210** |  |
| **103 MICHAEL** | **5551234567** |  |
| **104 SARAH** | **7899876543** |  |
| **105 DAVID** | **1112223333** |  |
| **106 OLIVIA** | **4445556666** |  |
| **107 ETHAN** | **7778889999** |  |
| **108 SOPHIA** | **2223334444** |  |
| **109 AIDEN** | **6667778888** |  |
| **110 EMMA** | **3334445555** |  |
| **111 BENJAMIN** | **8889990000** |  |
| **112 MIA** | **9990001111** |  |
| **113 WILLIAM** | **1112222** |  |
| **114 AVA** | **1112223333** |  |
| **115 JAMES** | **2223334444** |  |
| **15 rows selected.** |  |  |
| **2. Calculate the age of** | **each student based** | **on their date of birth, and** |

for students with missing birthdates (NULL), display 'Age Unknown' using the NVL function. Retrieve the STUDENT\_ID, first\_name, and the calculated age.

1. **You want to categorise students as 'Male' or 'Female' based on their gender, but some have NULL values. Use the NVL function to categorise students with NULL gender values as 'Unknown'. Retrieve the STUDENT\_ID, first\_name, and the categorised gender.**

SELECT STUDENT\_ID, FIRST\_NAME, NVL(

CASE

WHEN GENDER = 'Male' THEN 'Male'

WHEN GENDER = 'Female' THEN 'Female' ELSE 'Unknown'

END, 'Unknown') AS CategorizedGender FROM STUDENTINFO;

STUDENT\_ID FIRST\_NAME CATEGORIZED GENDER

101 JOHN Unknown

|  |  |  |
| --- | --- | --- |
| **102** | **EMILY** | **Unknown** |
| **103** | **MICHAEL** | **Unknown** |
| **104** | **SARAH** | **Unknown** |
| **105** | **DAVID** | **Unknown** |
| **106** | **OLIVIA** | **Unknown** |
| **107** | **ETHAN** | **Unknown** |
| **108** | **SOPHIA** | **Unknown** |
| **109** | **AIDEN** | **Unknown** |
| **110** | **EMMA** | **Unknown** |
| **111** | **BENJAMIN** | **Unknown** |
| **112** | **MIA** | **Unknown** |
| **113** | **WILLIAM** | **Unknown** |
| **114** | **AVA** | **Unknown** |
| **115** | **JAMES** | **Unknown** |

15 rows selected.

1. **Display the STUDENT\_ID, email, and for students with NULL email addresses, show 'No Email' using the NVL function.**

SELECT STUDENT\_ID, NVL(email, 'No Email') AS Email FROM STUDENTINFO;

STUDENT\_ID EMAIL

1. [**john.smith@email.com**](mailto:john.smith@email.com)
2. [**emily.davis@email.com**](mailto:emily.davis@email.com)
3. [**michael.johnson@email.com**](mailto:michael.johnson@email.com)
4. [**sarah.wilson@email.com**](mailto:sarah.wilson@email.com)
5. [**david.brown@email.com**](mailto:david.brown@email.com)
6. [**olivia.lee@email.com**](mailto:olivia.lee@email.com)
7. [**ethan.martinez@email.com**](mailto:ethan.martinez@email.com)
8. [**sophia.taylor@email.com**](mailto:sophia.taylor@email.com)
9. [**aiden.miller@email.com**](mailto:aiden.miller@email.com)
10. [**emma.anderson@email.com**](mailto:emma.anderson@email.com)
11. [**benjamin.harris@email.com**](mailto:benjamin.harris@email.com)
12. [**mia.johnson@email.com**](mailto:mia.johnson@email.com)
13. [**willain.white@email.com**](mailto:willain.white@email.com)
14. [**ava.robinson@email.com**](mailto:ava.robinson@email.com)
15. [**james.turner@email.com**](mailto:james.turner@email.com)

15 rows selected.

1. **You have a table that date\_of\_birth, and some dates are missing (NULL). Use the NVL function to display 'date\_of\_birth Not Available' for student with null dates. Retrieve the student name and date.**

SELECT FIRST\_NAME, NVL(TO\_CHAR(DATE\_OF\_BIRTH , 'DD-MM-YYYY'), 'Date of

Birth Not Available') AS DateOfBirth FROM STUDENTINFO;

FIRST\_NAME DATE OF BIRTH

|  |  |  |
| --- | --- | --- |
| **JOHN** |  | **15-05-1998** |
| **EMILY** |  | **20-03-1999** |
| **MICHAEL** |  | **10-07-1997** |
| **SARAH** |  | **05-01-2000** |
| **DAVID** |  | **30-09-1996** |
| **OLIVIA** |  | **18-12-1999** |
| **ETHAN** |  | **25-11-1998** |
| **SOPHIA** |  | **14-02-2002** |
| **AIDEN** |  | **12-04-1997** |
| **EMMA** |  | **07-08-2002** |
| **BENJAMIN** |  | **22-06-1995** |
| **MIA** |  | **01-10-1998** |
| **WILLIAM** |  | **12-03-2003** |
| **AVA** |  | **28-04-1999** |
| **JAMES** |  | **05-12-1996** |

15 rows selected.

NVL2 Function:-

1. **You want to calculate the age for a student.Otherwise, its 0. Use the NVL2 function to calculate the bonus. Retrieve the STUDENT\_ID, student\_name, age, and the calculated age.**

SELECT

STUDENT\_ID, FIRST\_NAME,

NVL2(date\_of\_birth, TRUNC(MONTHS\_BETWEEN(SYSDATE, date\_of\_birth) / 12), 0) AS age,

NVL2(date\_of\_birth, TRUNC(MONTHS\_BETWEEN(SYSDATE, date\_of\_birth) /

12) \* 100, 0) AS bonus FROM STUDENTINFO;

STUDENT\_ID FIRST\_NAME AGE BONUS

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **101** |  | **JOHN** |  | **25** |  | **2500** |
| **102** |  | **EMILY** |  | **24** |  | **2400** |
| **103** |  | **MICHAEL** |  | **26** |  | **2600** |
| **104** |  | **SARAH** |  | **23** |  | **2300** |
| **105** |  | **DAVID** |  | **27** |  | **2700** |
| **106** |  | **OLIVIA** |  | **23** |  | **2300** |
| **107** |  | **ETHAN** |  | **24** |  | **2400** |
| **108** |  | **SOPHIA** |  | **21** |  | **2100** |
| **109** |  | **AIDEN** |  | **26** |  | **2600** |
| **110** |  | **EMMA** |  | **21** |  | **2100** |
| **111** |  | **BENJAMIN** |  | **28** |  | **2800** |
| **112** |  | **MIA** |  | **25** |  | **2500** |
| **113** |  | **WILLIAM** |  | **20** |  | **2000** |
| **114** |  | **AVA** |  | **24** |  | **2400** |
| **115** |  | **JAMES** |  | **26** |  | **2600** |

15 rows selected.

1. **In a table that last name, some quantities are missing (NULL). Use the NVL2 function to calculate the . If the quantity is missing, assume its 0. Retrieve the order IDs and adjusted quantities.**

SELECT

STUDENT\_ID, NVL2(FIRST\_NAME,LAST\_NAME

, 0) AS adjusted\_quantity FRO STUDENTINFO;

STUDENT\_ID ADJUSTED\_QUANTITY

101 SMITH

|  |  |
| --- | --- |
| **102** | **DAVIS** |
| **103** | **JOHNSON** |
| **104** | **WILSON** |
| **105** | **BROWN** |
| **106** | **LEE** |
| **107** | **MARTINEZ** |
| **108** | **TAYLOR** |
| **109** | **MILLER** |
| **110** | **ANDERSON** |
| **111** | **HARRIS** |
| **112** | **JOHNSON** |
| **113** | **WHITE** |
| **114** | **ROBINSON** |
| **115** | **TURNER** |

15 rows selected.

1. **You need to categorise students based on their age. If a student is**

18 or older, categorise them as 'Adult'; otherwise, categorise them as 'Minor.' Use the NVL2 function to categorise students with NULL birthdates as 'Unknown.' Retrieve the STUDENT\_ID, first\_name, and the categorised age.

SELECT STUDENT\_ID, first\_name, NVL2(DATE\_OF\_BIRTH,

CASE

WHEN FLOOR(MONTHS\_BETWEEN(SYSDATE, DATE\_OF\_BIRTH) / 12) >= 1 THEN

'Adult'

ELSE 'Minor' END,

'Unknown'

) AS CategorizedAge FROM STUDENTINFO;

STUDENT\_ID FIRST\_NAME CATEGORIZED AGE

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **101** |  | **JOHN** |  | **Adult** |
| **102** |  | **EMILY** |  | **Adult** |
| **103** |  | **MICHAEL** |  | **Adult** |
| **104** |  | **SARAH** |  | **Adult** |

1. **DAVID Adult**
2. **OLIVIA Adult**
3. **ETHAN Adult**
4. **SOPHIA Adult**
5. **AIDEN Adult**
6. **EMMA Adult**
7. **BENJAMIN Adult**
8. **MIA Adult**
9. **WILLIAM Adult**
10. **AVA Adult**
11. **JAMES Adult**

15 rows select

1. **Calculate the age for students. If a students age is missing (NULL), use the NVL2 function to assume their age is null Retrieve the STUDENT\_ID, first\_name, and the age.**

SELECT STUDENT\_ID, first\_name, NVL2(DATE\_OF\_BIRTH,FLOOR(MONTHS\_BETWEEN(SYSDATE, date\_of\_birth)

/ 12),NULL) AS Age FROM STUDENTINFO;

STUDENT\_ID FIRST\_NAME AGE

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **101** |  | **JOHN** |  | **25** |
| **102** |  | **EMILY** |  | **24** |
| **103** |  | **MICHAEL** |  | **26** |
| **104** |  | **SARAH** |  | **23** |
| **105** |  | **DAVID** |  | **27** |
| **106** |  | **OLIVIA** |  | **23** |
| **107** |  | **ETHAN** |  | **24** |
| **108** |  | **SOPHIA** |  | **21** |
| **109** |  | **AIDEN** |  | **26** |
| **110** |  | **EMMA** |  | **21** |
| **111** |  | **BENJAMIN** |  | **28** |
| **112** |  | **MIA** |  | **25** |
| **113** |  | **WILLIAM** |  | **20** |
| **114** |  | **AVA** |  | **24** |
| **115** |  | **JAMES** |  | **26** |

15 rows selected.

1. **In a table storing gender, some values are missing (NULL). Use the NVL2 function to calculate the adjusted values. If the values is missing, assume its Not null. Retrieve the student’s names and adjusted values.**

SELECT FIRST\_NAME, NVL2(GENDER, GENDER, 'Not Available') AS

AdjustedGender FROM STUDENTINFO;

FIRST\_NAME ADJUSTED GENDER JOHN M

EMILY M

MICHAEL M

SARAH F

DAVID M

OLIVIA F

ETHAN M

SOPHIA F

AIDEN M

EMMA F

BENJAMIN M

MIA F

WILLIAM M

AVA F

JAMES M

15 rows selected.

LENGTH Function:-

1. **Calculate the length of each students first name in the STUDENTINFO table. Retrieve the STUDENT\_ID, first\_name, and the length of the first names.**

SELECT STUDENT\_ID, FIRST\_NAME, LENGTH(FIRST\_NAME) AS FirstNameLength FROM STUDENTINFO;

STUDENT\_ID FIRST\_NAME FIRSTNAME LENGTH

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **101** |  | **JOHN** |  | **4** |
| **102** |  | **EMILY** |  | **5** |
| **103** |  | **MICHAEL** |  | **7** |
| **104** |  | **SARAH** |  | **5** |
| **105** |  | **DAVID** |  | **5** |
| **106** |  | **OLIVIA** |  | **6** |
| **107** |  | **ETHAN** |  | **5** |
| **108** |  | **SOPHIA** |  | **6** |
| **109** |  | **AIDEN** |  | **5** |
| **110** |  | **EMMA** |  | **4** |
| **111** |  | **BENJAMIN** |  | **8** |
| **112** |  | **MIA** |  | **3** |
| **113** |  | **WILLIAM** |  | **7** |
| **114** |  | **AVA** |  | **3** |
| **115** |  | **JAMES** |  | **5** |

15 rows selected.

1. **You have a table that stores email addresses, and you want to find the length of each email address. Retrieve the email addresses and their lengths using the LENGTH function.**

SELECT EMAIL, LENGTH(EMAIL) AS EmailLength FROM STUDENTINFO;

EMAIL LENGTH

[john.smith@email.com](mailto:john.smith@email.com) 20

[emily.davis@email.com](mailto:emily.davis@email.com) 21

[michael.johnson@email.com](mailto:michael.johnson@email.com) 25

[sarah.wilson@email.com](mailto:sarah.wilson@email.com) 22

[david.brown@email.com](mailto:david.brown@email.com) 21

[olivia.lee@email.com](mailto:olivia.lee@email.com) 20

[ethan.martinez@email.com](mailto:ethan.martinez@email.com) 24

[sophia.taylor@email.com](mailto:sophia.taylor@email.com) 23

[aiden.miller@email.com](mailto:aiden.miller@email.com) 22

[emma.anderson@email.com](mailto:emma.anderson@email.com) 23

[benjamin.harris@email.com](mailto:benjamin.harris@email.com) 25

[mia.johnson@email.com](mailto:mia.johnson@email.com) 21

[willain.white@email.com](mailto:willain.white@email.com) 23

[ava.robinson@email.com](mailto:ava.robinson@email.com) 22

[james.turner@email.com](mailto:james.turner@email.com) 22

15 rows selected.

1. **Determine the number of characters in each students last name in the STUDENTINFO table. Retrieve the STUDENT\_ID, LAST\_NAME and the length of the last names.**

SELECT STUDENT\_ID, LAST\_NAME, LENGTH(LAST\_NAME ) AS LastNameLength FROM STUDENTINFO;

STUDENT\_ID LAST\_NAME LASTNAMELENGTH

15 rows selected.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **101** |  | **SMITH** |  | **5** |
| **102** |  | **DAVIS** |  | **5** |
| **103** |  | **JOHNSON** |  | **7** |
| **104** |  | **WILSON** |  | **6** |
| **105** |  | **BROWN** |  | **5** |
| **106** |  | **LEE** |  | **3** |
| **107** |  | **MARTINEZ** |  | **8** |
| **108** |  | **TAYLOR** |  | **6** |
| **109** |  | **MILLER** |  | **6** |
| **110** |  | **ANDERSON** |  | **8** |
| **111** |  | **HARRIS** |  | **6** |
| **112** |  | **JOHNSON** |  | **7** |
| **113** |  | **WHITE** |  | **5** |
| **114** |  | **ROBINSON** |  | **8** |
| **115** |  | **TURNER** |  | **6** |

1. **Calculate the length of each phone number in a table that number. Retrieve the phone number and their lengths using the LENGTH function.**

SELECT PHONE\_NUMBER, LENGTH(PHONE\_NUMBER) AS PhoneNumberLength FROM STUDENTINFO;

|  |  |  |  |
| --- | --- | --- | --- |
| **PHONE\_NUMBER** | **PHONE** | **NUMBER** | **LENGTH** |
| **1234567890** |  |  | **10** |
| **9876543210** |  |  | **10** |
| **5551234567** |  |  | **10** |
| **7899876543** |  |  | **10** |
| **1112223333** |  |  | **10** |
| **4445556666** |  |  | **10** |
| **7778889999** |  |  | **10** |
| **2223334444** |  |  | **10** |
| **6667778888** |  |  | **10** |
| **3334445555** |  |  | **10** |
| **8889990000** |  |  | **10** |
| **9990001111** |  |  | **10** |
| **1112222** |  |  | **7** |
| **1112223333** |  |  | **10** |
| **2223334444** |  |  | **10** |
| **15 rows selecte** | **d.** |  |  |

1. **You want to find the length of each Student’s full name in a table. Retrieve the First names,Last Name, Students Full Name and their lengths using the LENGTH function.**

SELECT

first\_name, LAST\_NAME

,

CONCAT(first\_name, LAST\_NAME

) AS Full\_Name, LENGTH(CONCAT(first\_name,LAST\_NAME

)) AS Full\_Name\_Length FROM STUDENTINFO;

FIRST\_NAME LAST\_NAME

FULL\_NAME FULL\_NAME\_LENGTH

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **JOHN** |  | **SMITH** |  | **JOHNSMITH** |  | **9** |
| **EMILY** |  | **DAVIS** |  | **EMILYDAVIS** |  | **10** |
| **MICHAEL** |  | **JOHNSON** |  | **MICHAEL JOHNSON** |  | **14** |
| **SARAH** |  | **WILSON** |  | **SARAH WILSON** |  | **11** |
| **DAVID** |  | **BROWN** |  | **DAVID BROWN** |  | **10** |
| **OLIVIA** |  | **LEE** |  | **OLIVIALEE** |  | **9** |
| **ETHAN** |  | **MARTINEZ** |  | **ETHANMARTINEZ** |  | **13** |
| **SOPHIA** |  | **TAYLOR** |  | **SOPHIATAYLOR** |  | **12** |
| **AIDEN** |  | **MILLER** |  | **AIDENMILLER** |  | **11** |
| **EMMA** |  | **ANDERSON** |  | **EMMAANDERSON** |  | **12** |
| **BENJAMIN** |  | **HARRIS** |  | **BENJAMIN HARRIS** |  | **14** |
| **MIA** |  | **JOHNSON** |  | **MIAJOHNSON** |  | **10** |
| **WILLIAM** |  | **WHITE** |  | **WILLIAM WHITE** |  | **12** |
| **AVA** |  | **ROBINSON** |  | **AVAROBINSON** |  | **11** |
| **JAMES** |  | **TURNER** |  | **JAMES TURNER** |  | **11** |

15 rows selected.

SOUNDEX Function STUDENTINFO Table):

1. **You have a requirement to find students in the STUDENTINFO table who may have similar-sounding last names. Write an SQL query using theSoundex function to display the STUDENT\_ID, LAST\_NAME and Soundex codes for students with last names that sound alike.**

SELECT STUDENT\_ID,FIRST\_NAME,LENGTH(FIRST\_NAME)AS NAME\_LENGTH FROM STUDENTINFO

STUDENT\_ID FIRST\_NAME SOUNDEXCODE

101 JOHN J500

|  |  |  |
| --- | --- | --- |
| **102** | **EMILY** | **E540** |
| **103** | **MICHAEL** | **M240** |
| **104** | **SARAH** | **S600** |
| **105** | **DAVID** | **D130** |
| **106** | **OLIVIA** | **O410** |
| **107** | **ETHAN** | **E350** |
| **108** | **SOPHIA** | **S100** |
| **109** | **AIDEN** | **A350** |
| **110** | **EMMA** | **E500** |
| **111** | **BENJAMIN** | **B525** |
| **112** | **MIA** | **M000** |
| **113** | **WILLIAM** | **W450** |
| **114** | **AVA** | **A100** |
| **115** | **JAMES** | **J520** |

15 rows selected.

1. **Use the Soundex function in the STUDENTINFO table to calculate the Soundex codes for each student's last name. Retrieve the STUDENT\_ID, LAST\_NAMEand Soundex codes.**

SELECT STUDENT\_ID, LAST\_NAME, SOUNDEX(LAST\_NAME) AS SoundexCode FROM STUDENTINFO;

STUDENT\_ID LAST\_NAME SOUNDEXCODE

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **101** |  | **SMITH** |  | **S530** |
| **102** |  | **DAVIS** |  | **D120** |
| **103** |  | **JOHNSON** |  | **J525** |
| **104** |  | **WILSON** |  | **W425** |
| **105** |  | **BROWN** |  | **B650** |
| **106** |  | **LEE** |  | **L000** |
| **107** |  | **MARTINEZ** |  | **M635** |
| **108** |  | **TAYLOR** |  | **T460** |
| **109** |  | **MILLER** |  | **M460** |
| **110** |  | **ANDERSON** |  | **A536** |
| **111** |  | **HARRIS** |  | **H620** |
| **112** |  | **JOHNSON** |  | **J525** |

|  |  |  |
| --- | --- | --- |
| **113** | **WHITE** | **W300** |
| **114** | **ROBINSON** | **R152** |
| **115** | **TURNER** | **T656** |

15 rows selected.

1. **In the STUDENTINFO table, some students may have last names that sound similar but are spelled differently. Write an SQL query with the Soundex function to identify such students. Display the STUDENT\_ID, LAST\_NAME , and Soundex codes.**

SELECT SOUNDEX(FIRST\_NAME),SOUNDEX(LAST\_NAME),SOUNDEX(GENDER)FROM STUDENTINFO;

SOUNDEX(FIRST\_NAME) SOUNDEX(LAST\_NAME ) SOUNDEX(GENDER)

J500 S530 M000

E540 D120 M000

M240 J525 M000

S600 W425 F000

D130 B650 M000

O410 L000 F000

E350 M635 M000

S100 T460 F000

A350 M460 M000

E500 A536 F000

B525 H620 M000

M000 J525 F000

W450 W300 M000

A100 R152 F000

J520 T656 M000

15 rows selected.

1. **You are tasked with finding potential duplicate student records in the STUDENTINFO table based on similar-sounding last names. Write an SQL query using the Soundex function to retrieve the STUDENT\_ID,**

LAST\_NAMEand Soundex codes for students with last names that sound alike.