

Query -1

1. procedure to update payment status

order-ID	cust-ID	order-date	order-Total	Parameter Status.
1	101	2024-02-01	250.00	
2	102	2024-02-02	400.75	
3	103	2024-02-01	150.00	

Query-2

Total events

calling the get-total-revenue
function

Output

801.25

1. NAME
 2. DATE
 3. TIME
 4. LOCATION
 5. REMARKS
 6. SIGNATURE
 7. DATE
 8. TIME
 9. LOCATION
 10. REMARKS
 11. SIGNATURE
 12. DATE
 13. TIME
 14. LOCATION
 15. REMARKS
 16. SIGNATURE
 17. DATE
 18. TIME
 19. LOCATION
 20. REMARKS
 21. SIGNATURE
 22. DATE
 23. TIME
 24. LOCATION
 25. REMARKS
 26. SIGNATURE
 27. DATE
 28. TIME
 29. LOCATION
 30. REMARKS
 31. SIGNATURE
 32. DATE
 33. TIME
 34. LOCATION
 35. REMARKS
 36. SIGNATURE
 37. DATE
 38. TIME
 39. LOCATION
 40. REMARKS
 41. SIGNATURE
 42. DATE
 43. TIME
 44. LOCATION
 45. REMARKS
 46. SIGNATURE
 47. DATE
 48. TIME
 49. LOCATION
 50. REMARKS
 51. SIGNATURE
 52. DATE
 53. TIME
 54. LOCATION
 55. REMARKS
 56. SIGNATURE
 57. DATE
 58. TIME
 59. LOCATION
 60. REMARKS
 61. SIGNATURE
 62. DATE
 63. TIME
 64. LOCATION
 65. REMARKS
 66. SIGNATURE
 67. DATE
 68. TIME
 69. LOCATION
 70. REMARKS
 71. SIGNATURE
 72. DATE
 73. TIME
 74. LOCATION
 75. REMARKS
 76. SIGNATURE
 77. DATE
 78. TIME
 79. LOCATION
 80. REMARKS
 81. SIGNATURE
 82. DATE
 83. TIME
 84. LOCATION
 85. REMARKS
 86. SIGNATURE
 87. DATE
 88. TIME
 89. LOCATION
 90. REMARKS
 91. SIGNATURE
 92. DATE
 93. TIME
 94. LOCATION
 95. REMARKS
 96. SIGNATURE
 97. DATE
 98. TIME
 99. LOCATION
 100. REMARKS
 101. SIGNATURE
 102. DATE
 103. TIME
 104. LOCATION
 105. REMARKS
 106. SIGNATURE
 107. DATE
 108. TIME
 109. LOCATION
 110. REMARKS
 111. SIGNATURE
 112. DATE
 113. TIME
 114. LOCATION
 115. REMARKS
 116. SIGNATURE
 117. DATE
 118. TIME
 119. LOCATION
 120. REMARKS
 121. SIGNATURE
 122. DATE
 123. TIME
 124. LOCATION
 125. REMARKS
 126. SIGNATURE
 127. DATE
 128. TIME
 129. LOCATION
 130. REMARKS
 131. SIGNATURE
 132. DATE
 133. TIME
 134. LOCATION
 135. REMARKS
 136. SIGNATURE
 137. DATE
 138. TIME
 139. LOCATION
 140. REMARKS
 141. SIGNATURE
 142. DATE
 143. TIME
 144. LOCATION
 145. REMARKS
 146. SIGNATURE
 147. DATE
 148. TIME
 149. LOCATION
 150. REMARKS
 151. SIGNATURE
 152. DATE
 153. TIME
 154. LOCATION
 155. REMARKS
 156. SIGNATURE
 157. DATE
 158. TIME
 159. LOCATION
 160. REMARKS
 161. SIGNATURE
 162. DATE
 163. TIME
 164. LOCATION
 165. REMARKS
 166. SIGNATURE
 167. DATE
 168. TIME
 169. LOCATION
 170. REMARKS
 171. SIGNATURE
 172. DATE
 173. TIME
 174. LOCATION
 175. REMARKS
 176. SIGNATURE
 177. DATE
 178. TIME
 179. LOCATION
 180. REMARKS
 181. SIGNATURE
 182. DATE
 183. TIME
 184. LOCATION
 185. REMARKS
 186. SIGNATURE
 187. DATE
 188. TIME
 189. LOCATION
 190. REMARKS
 191. SIGNATURE
 192. DATE
 193. TIME
 194. LOCATION
 195. REMARKS
 196. SIGNATURE
 197. DATE
 198. TIME
 199. LOCATION
 200. REMARKS
 201. SIGNATURE
 202. DATE
 203. TIME
 204. LOCATION
 205. REMARKS
 206. SIGNATURE
 207. DATE
 208. TIME
 209. LOCATION
 210. REMARKS
 211. SIGNATURE
 212. DATE
 213. TIME
 214. LOCATION
 215. REMARKS
 216. SIGNATURE
 217. DATE
 218. TIME
 219. LOCATION
 220. REMARKS
 221. SIGNATURE
 222. DATE
 223. TIME
 224. LOCATION
 225. REMARKS
 226. SIGNATURE
 227. DATE
 228. TIME
 229. LOCATION
 230. REMARKS
 231. SIGNATURE
 232. DATE
 233. TIME
 234. LOCATION
 235. REMARKS
 236. SIGNATURE
 237. DATE
 238. TIME
 239. LOCATION
 240. REMARKS
 241. SIGNATURE
 242. DATE
 243. TIME
 244. LOCATION
 245. REMARKS
 246. SIGNATURE
 247. DATE
 248. TIME
 249. LOCATION
 250. REMARKS
 251. SIGNATURE
 252. DATE
 253. TIME
 2

Task-6

Procedures, Functions, and loops in PL/SQL
(Based on online food ordering system) case

Study: online Food ordering system

objective:.

This will help in automating transactions
improving database efficiency, and enforcing
business rules in a structural manner

Step-1:- Ensure the Necessary Tables Exist
Before running the procedures and functions

Create the required tables in your
oracle database

```
DROP TABLE OrderTable PURGE;
```

```
DROP TABLE Delivery PURGE;
```

```
DROP TABLE Menu-Item PURGE;
```

```
CREATE TABLE OrderTable C
```

```
order-ID NUMBER PRIMARY KEY,
```

```
Cust-ID NUMBER;
```

```
order-Date DATE,
```

```
order-Total NUMBER(10,2);
```

```
Payment-status VARCHAR(20).
```

```
);
```

sp-119 in equal time interval (2 seconds)
we find the order book status and the

Query 3:

Loop 1: Mark all undelivered order as

"Delayed"

order-status-Delivery-status

Delayed

Delivered

Delayed


```
CREATE TABLE Delivery
Order-ID NUMBER PRIMARY KEY
Delivery-Status VARCHAR(20)
FOREIGN KEY (Order-ID) REFERENCES
OrderTable (Order-ID);
```

```
CREATE TABLE Menu-Item
Item-ID NUMBER PRIMARY KEY
Item-Name VARCHAR2(100)
Price Number(10,2)
```

```
INSERT INTO OrderTable VALUES
(1, 101, TO_DATE('2024-02-01', 'YYYY-MM-DD'),
250.50, 'Pending');
```

```
INSERT INTO OrderTable VALUES (2, 102, TO-
DATE('2024-02-02', 'YYYY-MM-DD'), 400.75,
'Paid');
```

```
INSERT INTO OrderTable VALUES (3, 103, TO-DATE
('2024-02-03', 'YYYY-MM-DD'), 150.00,
'Pending');
```

```
INSERT INTO Delivery VALUES (1, 'Pending')
```

```
INSERT INTO Delivery VALUES (2, 'Delivered');
```

```
INSERT INTO Delivery VALUES (3, 'Pending');
```

```
INSERT INTO Menu-Item VALUES (1, 'Pizza', 50);
```

```
INSERT INTO Menu-Item VALUES (2, 'Burger', 30);
```

```
INSERT INTO Menu-Item VALUES (3, 'Pasta', 45);
```

```
CREATE TABLE Delivery
Order_ID NUMBER(10) PRIMARY KEY,
Delivery_Status VARCHAR(10),
FOREIGN KEY (Order_ID) REFERENCES
OrderTable (Order_ID);
```

```
CREATE TABLE MenuItem
Item_ID NUMBER(10) PRIMARY KEY,
Item_Name VARCHAR(20),
Price NUMBER(10,2);
```

```
INSERT INTO OrderTable VALUES
(1, 101, TO_DATE('2024-02-01', 'YYYY-MM-DD'),
250.50, 'Pending');
```

```
INSERT INTO OrderTable VALUES (2, 102, TO-
DATE('2024-02-02', 'YYYY-MM-DD'), 400.35,
'Paid');
```

```
INSERT INTO OrderTable VALUES (3, 103, TO_DATE
('2024-02-03', 'YYYY-MM-DD'), 150.00,
'Pending');
```

```
INSERT INTO Delivery VALUES (1, 'Pending');
```

```
INSERT INTO Delivery VALUES (2, 'Delivered');
```

```
INSERT INTO Delivery VALUES (3, 'Pending');
```

```
INSERT INTO MenuItem VALUES (1, 'Pizza', 5.00);
```

```
INSERT INTO MenuItem VALUES (2, 'Burger', 3.00);
```

```
INSERT INTO MenuItem VALUES (3, 'Pasta', 3.50);
```



```
CREATE TABLE Delivery(  
  order-ID NUMBER PRIMARY KEY,  
  Delivery-status VARCHAR(  
    FOREIGN KEY (order-ID) REFERENCES  
    orderTable (order-ID).
```

```
);  
  
CREATE TABLE Menu-Item(  
  Item-ID NUMBER PRIMARY KEY,  
  Item-Name VARCHAR2(100),  
  Price Number(10,2)
```

```
);  
  
INSERT INTO orderTable VALUES  
  (1, 101, TO_DATE('2024-02-01', 'YYYY-MM-DD'),  
  250.50, 'Pending');
```

```
INSERT INTO orderTable VALUES (2, 102, TO-  
  DATE('2024-02-02', 'YYYY-MM-DD'), 400.75,  
  'Paid');
```

```
INSERT INTO orderTable VALUES (3, 103, TO-DATE  
  ('2024-02-03', 'YYYY-MM-DD'), 150.00,  
  'Pending');
```

```
INSERT INTO Delivery VALUES (1, 'pending');
```

```
INSERT INTO Delivery VALUES (2, 'delivered');
```

```
INSERT INTO Delivery VALUES (3, 'pending');
```

```
INSERT INTO Menu-Item VALUES (1, 'Pizza', 500);
```

```
INSERT INTO Menu-Item VALUES (2, 'Burger', 300);
```

```
INSERT INTO Menu-Item VALUES (3, 'Pasta', 450);
```

Query-4

output (cust, ID=102)

order-ID	cust-ID	order-Date	order-Total	Payment-status
2	102	2024-02-02	100.75	paid

Query-5

output:

Item-ID	Item-name	price
1	pizza	150
2	Burger	270
3	Pasta	105

1. Procedure to update Payment - status

Step-1: Create a procedure

```
CREATE OR REPLACE PROCEDURE update-payment-  
status (
```

```
    P_order-ID IN NUMBER,
```

```
    P_new-status IN VARCHAR2
```

```
  )AS
```

```
  BEGIN
```

```
    UPDATE order.Table
```

```
      SET Payment-status = P_new-status
```

```
    WHERE order-ID = P_order-ID;
```

```
  COMMIT;
```

```
    DBMS-output.put_line ('Payment status  
updated successfully for order ID: ' || P_order-ID);
```

```
  END;
```

Expected output:-

Procedure Created

Step-2:- Execution

```
BEGIN
```

```
  update-payment-status (1, 'Paid');
```

```
END;
```

output

Payment status updated successfully

for order ID: 1

Statement processed

Query-2:- Function to calculate Total Revenue

Step-1:- create a Function.

CREATE OR REPLACE FUNCTION get_Total -
Revenue

RETURN NUMBER AS v_Total - Revenue . number;
BEGIN

SELECT SUM (order - total) INTO v - Total - Revenue
FROM order Table;

Return v - Total - Revenue

END;

Expected output,
Function created

Step-2:- Execution

get - Total - Revenue;

801.25

Query-3:- Loop:- Mark All undelivered orders as
"Delayed"

DECLARE

v - order - ID order Table - order - ID % TYPE;

CURSOR cur IS SELECT order - ID . FROM
Delivery WHERE Delivery - status = "Pending";

BEGIN

open cur;

loop

FETCH cur INTO v - order - ID;

EXIT WHEN cur % NOT FOUND;

UPDATE Delivery

SET Delivery - status = &'Delayed'

WHERE order_ID = V-order_ID;

DBMS - output . put_line ('order' || V-order_ID ||
'marked as Delayed');

END LOOP;

close cur;

COMMIT;

END;

Expected output;

1 (rows) updated

Query-u:- procedure to get order details by
customer ID

Step-1:- Create a procedure

CREATE OR REPLACE PROCEDURE get_custo-
mer_order .

(P - Cust_ID . IN NUMBER) - AS

BEGIN

FOR order_rec IN (SELECT order_ID, order_Date
Order - Total , payment - status . FROM Order
Table

WHERE cust_ID = P . Cust_ID) LOOP

DBMS - output . put_line ('order ID;' || order_rec
order_ID || , Date;' || order_rec . order - total,
Total ; || . order_rec . order - Total ; status
: || order_rec . payment - status);

Expected output

order ID : 1, Date: 2024-02-01, Total = 250.5,

Payment: paid

statement processed

Query 5:- procedure to Apply discount on Menu Items

step-1:- Create a procedure

CREATE OR REPLACE PROCEDURE Apply-Discout

(discount_percent IN NUMBER)

IS BEGIN

UPDATE Menu-Item

set price = price - (price * discount_percent / 100);

COMMIT

DBMS-output.put_line ('discount Applied' ||

"discount_percent" || "%");

END;

VEL TECH - CCE	
EX NO.	
PERFORMANCE (5)	
RESULT AND ANALYSIS (3)	
VIVA VOCE (3)	
RECORD (4)	
TOTAL (15)	
DATE	